DEVELOPMENT OF SOLUTION TECHNIQUES FOR NONLINEAR STRUCTURAL ANALYSIS

FINAL REPORT

Contract NAS8-29625

September 30, 1974

Prepared by

BOEING AEROSPACE COMPANY
RESEARCH AND ENGINEERING DIVISION
SEATTLE, WASHINGTON 98124

R. G. Vos – Technical Leader

J. S. Andrews – Program Leader

Prepared For

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER
MARSHALL SPACE FLIGHT CENTER, ALABAMA 35812

(NASA-CR-120525) DEVELOPMENT OF SOLUTION TECHNIQUES FOR NONLINEAR STRUCTURAL ANALYSIS Final Report (Boeing Aerospace Co., Seattle, Wash.)
N75-11371
CSCL 20K G3/39
Unclas
02589

THE BOEING COMPANY

D180-18325-1

DEVELOPMENT OF SOLUTION TECHNIQUES FOR NONLINEAR STRUCTURAL ANALYSIS

FINAL REPORT

Contract NAS8-29625

September 30, 1974

Prepared by

BOEING AEROSPACE COMPANY
RESEARCH AND ENGINEERING DIVISION
SEATTLE, WASHINGTON 98124

R. G. Vos - Technical Leader

J. S. Andrews - Program Leader

Prepared for

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GEORGE C. MARSHALL SPACE FLIGHT CENTER
MARSHALL SPACE FLIGHT CENTER, ALABAMA 35812

ABSTRACT

Nonlinear structural solution methods in the current research literature are classified according to order of the solution scheme, and it is shown that the analytical tools for these methods are uniformly derivable by perturbation techniques. A new perturbation formulation is developed for treating an arbitrary nonlinear material, in terms of a finite-difference generated stress-strain expansion. Nonlinear geometric effects are included in an explicit manner by appropriate definition of an applicable strain tensor. A new finite-element pilot computer program PANES (Program for Analysis of Nonlinear Equilibrium and Stability) is presented for treatment of problems involving material and geometric nonlinearities, as well as certain forms on nonconservative loading.

TABLE OF CONTENTS

SECTION PAGE

1.0 INTRODUCTION

Purpose and Scope of the Study - The present research was undertaken to develop improved techniques for solution of structures with material and geometric nonlinearities, including the limit point and bifurcation behavior which occurs in buckling and collapse problems. Because the effectiveness of such solution techniques has been found to depend strongly on the method used for generating the nonlinear equations, e.g., creation of the system Jacobian matrix, improved equation generation techniques were also emphasized. Available nonlinear analysis methods were evaluated for their current capabilities and their projected long term potentials, and the methods judged to be most promising formed a starting point for development of the techniques presented in this report. Corresponding FORTRAN subroutines were developed and incorporated into the pilot computer program PANES (acronym of the Program for Analysis of Nonlinear Equilibrium and Stability) for checkout and evaluation. The equation generation and solution techniques are within the framework of the finite element structural discretization method.

1.1 General Philosophy and Evaluation of Methods

Criteria - Structural solution methods available in the current literature were initially evaluated for this study based on four general criteria:

1. A high degree of automation which minimizes the burden on the user.
2. Cost effectiveness for large size problems.
3. The use of an effective incremental technique which allows the user to follow and plot the structural response path.
4. Achievement of accuracy by a self-correcting characteristic, which assures that the true solution is approached at each point where results are desired.

During the study it was decided that recent advancements in structural theory made it timely to broaden the applicability of the developed equation generation and solution techniques by including a fifth requirement:

5. An efficient treatment of large-strain problems, and of arbitrary nonlinear elastic or inelastic materials.

Classification of Methods - The current literature contains a very broad variety of nonlinear solution methods, and even the specialized requirements for nonlinear structural solutions have not resulted in consensus on a best method or methods. On the other hand, certain types of highly nonlinear problems are presently receiving considerable attention (for example, stability analyses and large-strain effects with arbitrary nonlinear materials), and such problems tend to eliminate certain methods from consideration while giving some direction to future research and development.

Most of the nonlinear structural solution methods can be broadly grouped into three classes:

1. Methods which use only the initial (constant) stiffness of the structure, and rely on iteration with the calculation of residual (unbalanced) forces to achieve the correct solution. The loading may be applied incrementally or in a single step.
2. Methods which form the Jacobian (tangent stiffness) matrix at a series of load increments, without iteration; also, various combined incremental and iterative algorithms which update the Jacobian at each step or periodically.
3. Higher-order methods (perturbation approaches or various numerical integration schemes) which employ higher-order derivative relations in addition to the first-order Jacobian coefficients.

In many respects, the above ordering of classes is according to increasing sophistication and greater capability. For example, the class 3 methods are especially suited for analysis of complicated limit point and postbuckling problems. As might be expected, the historical development of nonlinear solution methods has shown some tendency to progress from original class 1 techniques to those of classes 2 and 3.

1.2 Previous Developments and Present Work

Historical Development - Early finite-element work in nonlinear structural analysis began with a paper by Turner, Dill, Martin and Melosh (1960). This work incorporated nonlinear geometric effects within the so-called "geometric" stiffness matrix, and various incremental and iterative solution procedures were recognized by the authors. In the initial attempts at nonlinear solutions which followed, there was a natural tendency to generalize existing linear capabilities. This usually led to iterative approaches and use of the initial constant stiffness matrix, with calculation of the nonlinearities as additional load terms. The geometric stiffness matrix, however, formed the basis for eigenvalue buckling analyses. A more consistent and theoretical basis for the geometric stiffness matrix was investigated by Gallagher and Padlog (1963), who used a strain-energy derivation with the same displacement functions for both the linear and nonlinear stiffness terms. Several formulations for nonlinear beam and plate analysis soon followed. Mallett and Marcal (1968) presented a unifying basis for formulating large displacement problems, by deriving the total strain energy as a function of nodal displacements and including previously neglected nonlinear terms. Meanwhile, developments were proceeding in the area of material nonlinearities, for example the plasticity work of Argyris (1965) and Marcal (1968), and nonlinear elastic analysis by Oden and Kubitza (1967).

A consensus on solution methods did not appear, however, and different approaches were emphasized by different research groups. Zienkiewicz and coworkers popularized some of the class 1 solution methods, and Zienkiewicz et al. (1969) presented a particular method called the "initial stress" residual-load method. This work was followed more recently by techniques for improving the iterative convergence of such methods, e.g. Nayak and Zienkiewicz (1972). However, the paper by Zienkiewicz and Nayak (1971) presents a quite general formulation for various class 2 methods with application to combined geometric and material large-strain nonlinearities. Considerable work in geometric and material nonlinearity has been done at Brown University, for example Marcal (1969), and McNamara and Marcal (1971). Researchers there have tended to favor class 2 methods without iteration, although the use of one or two iterations in each load step has been suggested as a way of increasing accuracy. Combined incremental and iterative class 2 techniques (Newton Raphson iteration, for example) have been employed for analysis of highly nonlinear material and geometric nonlinearities, including some stability problems, by Oden and Key (1970), Sandidge (1973), and Key (1974). It would certainly appear that for such problems the class 1 methods at least are highly unsuitable. A number of nonlinear survey and development papers have been written at Texas A & M University, including Haisler et al. (1971), Stricklin et al. (1972), Stricklin et al. (1973) and Tillerson et al. (1973). These papers provide a detailed investigation of various class 1 and 2 methods, as well as certain class 3 methods which rely on numerical integration schemes. Although perturbation procedures are not tested by these authors, it is suggested in one of the early papers that perturbation techniques would be very time consuming for cases with large degrees of freedom, while a more recent paper notes that these techniques require further evaluation before they will be accepted by structural analysts. Many other researchers working in the areas of limit point and bifurcation stability problems, however, have concentrated on perturbation methods, reviving the original theoretical developments in that area by Koiter (1945). Haftka et al. (1970) use an extension of Koiter's perturbation theory in a solution approach called the "modified structure method." Morin (1970) uses perturbation techniques in developing higher-order predictor and corrector algorithms for analysis of geometrically nonlinear shells. Gallagher and Mau (1972) and Mau and Gallagher (1972) establish procedures for limit point and postbuckling analysis based on perturbation expansions and the evaluation of determinants, which employ a combination of class 1, 2 and 3 solution techniques. A number of other perturbation developments of a more theoretical nature are included in the references and bibliography section of this report.

Much of the present diversity in nonlinear solution methods can be attributed to a desire to further investigate the potentials of all methods and to compare the results obtained from them. However, the comparisons and evaluations which are presented often disagree in their conclusions as to the effectiveness of a particular method. It must be surmised that the evaluation of nonlinear solution methods is necessarily influenced by the previous experiences and preferences of the researcher, by the degree of sophistication in his various solution method tools, and by the type of problems toward which his interests are directed.

Direction of Present Work - Because the present work was directed toward obtaining techniques whose applicability included the more highly nonlinear structures, a decision was made to eliminate from consideration the constant-stiffness methods of class 1. Although schemes have been proposed for extending these methods to more severe nonlinearities, it must be said that the arguments given are not convincing. In fact, when the structural system has advanced into a highly nonlinear state, the initial constant portion of the stiffness does not really possess any more significance than that provided by an arbitrary positive definite matrix; it can not be expected that a technique based on this matrix will be of any significant value in advancing the solution beyond the current state. It was also decided in the present work to reject those methods of a non self-correcting nature, i.e., methods which do not involve an iterative calculation of the unbalanced or "residual" forces, which gives an indication of accuracy and allows the solution to be improved. Although such methods are sometimes effective, they can lead to serious errors in the computed results, especially for path dependent problems. A third group of methods eliminated from consideration were those which use the solution data generated at several previous solution points. Such methods essentially extrapolate the previous data, either by some numerical integration formula or by a curve fitting approach. These methods require storage of previous data and are usually not self-starting. However the main objection to their use would seem to be that the same type of capability is provided by perturbation methods, which more accurately evaluate the path direction and are more generally applicable to a wide range of highly nonlinear problem types (e.g., those involving path discontinuities such as bifurcation points).

With these considerations, the methods which remain for development include methods of "incremental loading", Newton Raphson iteration and its modifications involving only periodic updating of the Jacobian, and higher-order methods including various orders of predictor and corrector algorithms. In order to make the current methods applicable to cases of large strain and arbitrary nonlinear materials, the equation generation process is accomplished in the present work by a finite difference expansion procedure. It is found that generation of the nonlinear equations by this means within a perturbation context provides a unifying basis for definition of the nonlinear solution terms, including as special cases the first-order Newton Raphson and incremental loading methods, as well as almost an unlimited variety of higher-order solution techniques. The perturbation procedures have the advantage of a sound theoretical basis in classical developments, and lend themselves readily to both limit point and postbuckling problems as well as to simple nonlinear behavior without critical points.

2.0 DETERMINATION OF THE EQUILIBRIUM PATH: A GENERALIZATION OF STATIC PERTURBATION TECHNIQUES

In this Section the theory and techniques are developed for following the nonlinear equilibrium path of a structure under prescribed loading. It is assumed that the equilibrium path is continuous and unique, although limit point behavior is allowed (the non-uniqueness due to bifurcation of the equilibrium path is considered in Section 3. The development follows the "static perturbation method" which was recognized and established in concrete form by Sewell (1965). The present work generalizes previous structural solution techniques based on the method to allow effective treatment of arbitrary nonlinear materials. The resulting formulation is shown to provide a quite general and unifying basis for solution of nonlinear structures, including geometric and material nonlinearities as well as certain forms of nonconservative loading. A summary of the formulation is contained in the paper by Vos (1974).

2.1 Description of Nonlinearities

An important characteristic of the present method is a preliminary separation of the nonlinear material and geometric effects, which minimizes the required number of perturbation expansion terms, and also increases numerical accuracy.

Material Effects - The nonlinear material effects are described by expanding the stress about a known equilibrium configuration:

$$\sigma_i = \sigma^*_i + D0^*_{ij}\Delta\varepsilon_j + D1^*_{ijk}\Delta\varepsilon_j\Delta\varepsilon_k + \dots \qquad (2\text{-}1)$$

which provides the stress, σ, in terms of the incremental strain, $\Delta\varepsilon$. Here and throughout this work, an asterisk (*) denotes quantities evaluated at a known equilibrium state, and Δ denotes an incremental quantity. In (2-1) σ^* is the initial stress, while

DO* and D1* are 2nd and 3rd order incremental stress-strain tensors, respectively. This type of expansion can be developed numerically for a general nonlinear elastic or inelastic material, by an efficient finite difference or Taylor series evaluation.† Complete symmetry of the D tensors can be used to advantage if they are derivable from a strain-enery function, or in certain other cases such as that of associative plasticity. These considerations are discussed by Zienkiewicz and Nayak (1971) in a development which employs only the 2nd order (DO) tensor. In any case, the tensors can be made symmetric in the j, k and any higher order indices.

Geometric Effects - The nonlinear geometric effects are included through a definition of finite element displacement functions and an appropriate strain tensor, giving

$$\theta_i = G_{ij} q_j \tag{2-2a}$$

$$\varepsilon_i = A0_{ij}\theta_j + \frac{1}{2} A1_{ijk}\theta_j\theta_k \tag{2-2b}$$

Here q are the element generalized (nodal) displacements of an element, G is obtained by differentiating the assumed displacement functions, θ are the displacement derivatives at any point, while A0 and A1 are constant coefficients which define the strain tensor with $A1_{ijk} = A1_{ikj}$. The explicit form (2-2b) of the strain tensor will be seen to simplify later manipulations.

† The best approach developed thus far is a forward difference scheme, which requires a minimum number of function evaluations and allows an arbitrary difference size for each independent variable. Explicit coefficients have been derived for such expansions of either linear, quadratic or cubic form, and in terms of an arbitrary number of variables. Details are provided in Appendix A. It may be noted that for certain problems involving incompressible materials, the hydrostatic stress must be included as an indepdent parameter in addition to the strains.

Advantages of Present Approach - The present approach defines all nonlinearities through the form of (2-1) and (2-2), rather than through a direct expansion of the nodal displacements such as that used in the investigation of Oden and Key (1970). The present approach appears to offer substantial advantages, because it allows implementation of perturbation theories into limit point and bifurcation analysis, without involving a huge number of terms and formidable algebraic operations. As a practical matter, it should also be noted that a numerical expansion based on displacements often causes severe problems with accuracy of the expansion coefficients, due to large differences in magnitude between individual displacement limits (e.g., between the membrane and bending freedoms of a plate or shell), and the selection of accurate finite difference sizes then becomes difficult. Accuracy is more easily obtained in an expansion of the type (2-1), because the strain limits tend to be of the same order of magnitude.

2.2 Formulation of Equilibrium Equations

Virtual Work Statement - The principle of virtual work, which is valid for arbitrary nonlinear materials and nonconservative systems, is employed to obtain equilibrium equations for the system of finite elements. The formulation is developed here for a conservative system, and nonconservative effects are treated in Appendix B. The equivalence of external and internal virtual work, relates the generalized nodal forces p and displacements q of a particular finite element, in the element equilibrium equation

$$\delta q_i p_i = \int_V \delta \varepsilon_a \sigma_a \, dV \qquad (2-3)$$

which holds along any equilibrium path in the neighborhood of the reference equilibrium (*) configuration. Here $\delta\varepsilon$ and δq are kinematically consistent variations, and from equation (2-2)

$$\delta\theta_i = G_{ij}\delta q_j \tag{2-4a}$$

$$\delta\varepsilon_i = (AO_{ij} + Al_{ijk}\theta_k)\delta\theta_j \equiv A_{ij}\delta\theta_j \equiv B_{ij}\delta q_j \tag{2-4b}$$

The integral in equation (2-3) is taken over the volume of the element, and it is to be noted that a proper definition of stress and strain is required to give the correct evaluation of internal work. One approach for accomplishing this is a formulation of Lagrangian strain and second Piola-Kirchoff stress integrated over the undeformed volume, e.g. see Oden and Key (1970).

<u>Basic Equilibrium Equation</u> - Substituting for $\delta\varepsilon$, and noting that (2-3) must be satisfied for arbitrary variations δq, provides the basic equilibrium equation for the element, as

$$p_i = \int_V B_{ai}\sigma_a \, dV = \int_V^{G_{mi}} (AO_{am} + Al_{amn}\theta_n)\sigma_a \, dV \tag{2-5a}$$

In order to merge the element equations into the system equations, the usual type of finite element transformation is applied. The system forces and displacements will be denoted by the capitals P and Q, respectively, and the system basic equilibrium equation corresponding to (2-5a) is written as

$$P_i = \int_V B_{ai}\sigma_a \, dV = \int_V^{G_{mi}} (AO_{am} + Al_{amn}\theta_n)\sigma_a \, dV \tag{2-5b}$$

where now it must be understood that the integral is summed over all elements while applying the proper element-system nodal transformations. With this understanding, the element and system quantities will here be used interchangeably.

<u>Derivative Relations</u> - Equations (2-5) may now be differentiated as many times as desired with respect to some suitable path parameter. Toward that end, it is useful to record here the following typical derivative relations, where an overdot ($\dot{}$) denotes differentiation with respect to the path parameter.

$$\left.\begin{aligned}
\dot{\sigma}_a &= DO^*_{ab}\dot{\varepsilon}_b + 2D1^*_{ab}\dot{\varepsilon}_b\Delta\varepsilon_c + \dots \\
\ddot{\sigma}_a &= DO^*_{ab}\ddot{\varepsilon}_b + 2D1^*_{abc}\ddot{\varepsilon}_b\Delta\varepsilon_c + 2D1^*_{abc}\dot{\varepsilon}_b\dot{\varepsilon}_c + \dots \\
\dot{\varepsilon}_a &= B_{ai}\dot{q}_i \\
\ddot{\varepsilon}_a &= B_{ai}\ddot{q}_i + \dot{B}_{ai}\dot{q}_i = B_{ai}\ddot{q}_i + A1_{amn}\dot{\theta}_m\dot{\theta}_n
\end{aligned}\right\} \qquad (2\text{-}6a)$$

and at the reference configuration ($\Delta\varepsilon = 0$), we have

$$\left.\begin{aligned}
\dot{\sigma}^*_a &= DO^*_{ab}\dot{\varepsilon}^*_b \\
\ddot{\sigma}^*_a &= DO^*_{ab}\ddot{\varepsilon}^*_b + 2D1^*_{abc}\dot{\varepsilon}^*_b\dot{\varepsilon}^*_c
\end{aligned}\right\} \qquad (2\text{-}6b)$$

<u>First Order Equilibrium Equation</u> - Differentiating equation (2-5b) once, and evaluating at the reference equilibrium (*) configuration, gives

$$\dot{P}^*_i = \int_V (\dot{B}^*_{ai}\sigma^*_a + B^*_{ai}\dot{\sigma}^*_a)\ dV \qquad (2\text{-}7a)$$

Substituting from relations (2-6b) gives

$$\dot{P}^*_i = \int_V (G_{mi}\sigma^*_a A1_{amn}G_{nj}\dot{q}^*_j + B^*_{ai}D0^*_{ab}B^*_{bj}\dot{q}^*_j)\ dV \qquad (2\text{-}7b)$$

This is the first order equilibrium equation, which may be written in the form

$$\dot{P}^*_i = K0^*_{ij}\dot{Q}^*_j \qquad (2\text{-}7c)$$

where

$$KO^*_{ij} = \int_V (G_{mi}\sigma^*_a A1_{amn}G_{nj} + B^*_{ai}DO^*_{ab}B^*_{bj})\ dV$$

and

$$B^*_{ai} = G_{mi}A^*_{am} = G_{mi}(A0_{am} + A1_{amn}\theta^*_n)$$

The "tangent stiffness" relation (2-7c) is equivalent to the incremental matrix formulation of Zienkiewicz and Nayak (1971), although the tensor form given here shows perhaps more clearly the symmetry and differentiability properties of the tangent (Jacobian) matrix KO*. The first contribution to KO* is due to the initial stresses during changing geometry, and is always symmetric in form. The second contribution is due to the incremental stress-strain relation, and its symmetry depends on symmetry of the matrix DO*.

Second Order Equilibrium Equation - A second differentiation of (2-5b) and evaluation at the reference configuration, gives

$$\ddot{P}^*_i = \int_V (\ddot{B}^*_{ai}\sigma^*_a + 2\dot{B}^*_{ai}\dot{\sigma}^*_a + B^*_{ai}\ddot{\sigma}^*_a)dV \tag{2-8a}$$

Substituting from relations (2-6b) gives

$$\ddot{P}^*_i = \int_V \{G_{mi}\sigma^*_a Al_{amn}G_{nj}\ddot{q}^*_j + 2G_{mi}Al_{amn}\dot{\theta}^*_n\, DO^*_{ab}\dot{\varepsilon}^*_b$$

$$+ B^*_{ai}(DO^*_{ab}B^*_{bj}\ddot{q}^*_j + DO^*_{ab}Al_{brs}\dot{\theta}^*_r\dot{\theta}^*_s + 2D1^*_{abc}\dot{\varepsilon}^*_b\dot{\varepsilon}^*_c)\}\, dV \tag{2-8b}$$

This is the second order equilibrium equation, which may be written in the form

$$\ddot{P}^*_i = KO^*_{ij}\ddot{Q}^*_j + P1^*_i \tag{2-8c}$$

where P1 is a psuedo force term given by

$$P1^*_i = \int_V G_{mi}\{DO^*_{ab}(A^*_{am}Al_{brs}\dot{\theta}^*_r\dot{\theta}^*_s + 2Al_{amn}\dot{\varepsilon}^*_b\dot{\theta}^*_n)$$

$$+ 2D1^*_{abc}A^*_{am}\dot{\varepsilon}^*_b\dot{\varepsilon}^*_c\}\ dV$$

2.3 Solution of Equilibrium Equations

Incremental Load and Path Parameters - An increment of conservative loading is defined by

$$\Delta P_i = \Lambda P_i^\circ, \; P_i = \Lambda P_i^\circ, \text{ etc.} \qquad (2\text{-}9)$$

using a variable load parameter Λ and constant nodal load distribution P°.

Taylor series expansions are then used to approximate both the incremental load parameter Λ and displacements ΔQ:

$$\Lambda = \dot{\Lambda}^* S + \frac{1}{2} \ddot{\Lambda}^* S^2 + \ldots \qquad (2\text{-}10a)$$

$$\Delta Q_i = \dot{Q}_i^* S + \frac{1}{2} \ddot{Q}_i^* S^2 + \ldots \qquad (2\text{-}10b)$$

In order to handle limit point situations within the present formulation, the path parameter S is here taken as defined by

$$S^2 = i \; KO_{ij}^* \Delta Q_i \Delta Q_j \geqq 0, \; i = \pm 1 \qquad (2\text{-}11)$$

with the requirement that KO*, evaluated at the beginning of each load increment, be nonsingular (either positive or negative definite). Without any loss of generality, additional requirements imposed at every path point, S, are that

$$\dot{S} = \dot{S}^2 = 1$$

$$\ddot{S} = \dddot{S} = \ldots = 0$$

Successive differentiation of (2-11) provides the relations

$$\left.\begin{aligned} 2S\dot{S} &= i \; KO_{ij}^* (\dot{Q}_i \Delta Q_j + \Delta Q_i \dot{Q}_j) \\ 2\dot{S}^2 &= i \; KO_{ij}^* (\ddot{Q}_i \Delta Q_j + 2\dot{Q}_i \dot{Q}_j + \Delta Q_i \ddot{Q}_j) \end{aligned}\right\} \qquad (2\text{-}12a)$$

$$0 = i\ KO^*_{ij}(\dddot{Q}_i \Delta Q_i + 3\ddot{Q}_i\dot{Q}_j + 3\dot{Q}_i\ddot{Q}_j + \Delta Q_i\dddot{Q}_j) \qquad (2\text{-}12a)$$

and evaluation at the reference state ($S = \Delta Q_i = 0$), yields

$$\dot{S}^2 = 1 = i\ KO^*_{ij}\dot{Q}^*_i\dot{Q}^*_j$$

$$0 = i\ KO^*_{ij}(\ddot{Q}^*_i\dot{Q}^*_j + \dot{Q}^*_i\ddot{Q}^*_j) \qquad (2\text{-}12b)$$

It may be noted that relations (2-12) hold for the general case of an unsymmetric KO* matrix.

Determination of Rate Quantities - In order to implement various solution techniques, the equilibrium equations (2-7c) and (2-8c) must be used to determine the load and displacement rates. Multiplying (2-7c) by $\dot{Q}^*$, and making use of (2-9) and (2-12b), gives

$$i = \dot{\Lambda}^*\dot{Q}^*_i P^\circ_i \qquad (2\text{-}13a)$$

Solving (2-7c) for $\dot{Q}^*$ gives

$$KO^{*-1}_{ij}\dot{\Lambda}^* P^\circ_j = \dot{Q}^*_i \equiv \dot{\Lambda}^* Q^\circ_i \qquad (2\text{-}13b)$$

and substituting $\dot{\Lambda}^* Q^\circ$ for $\dot{Q}^*$ in (2-13a) gives

$$\dot{\Lambda}^{*2} = i/(Q^\circ_i P^\circ_i) \qquad (2\text{-}13c)$$

where now i is chosen to make $\dot{\Lambda}^{*2}$ positive. Multiplying (2-8c) by $\dot{Q}^*$, and again making use of (2-9) and (2-12b), gives

$$\dot{Q}^*_i\ddot{\Lambda}^* P^\circ_i = KO^*_{ij}\dot{Q}^*_i\ddot{Q}^*_j + \dot{Q}^*_i Pl^*_i = -KO^*_{ij}\ddot{Q}^*_i\dot{Q}^*_j + \dot{Q}^*_i Pl^*_i \qquad (2\text{-}14a)$$

Solving (2-8c) for $\ddot{Q}^*$ gives

$$\ddot{Q}^*_i = KO^{*-1}_{ij}(\ddot{\Lambda}^* P^\circ_j - P1^*_j) \equiv \ddot{\Lambda}^* Q^\circ_i - Q1^*_i \qquad (2\text{-}14b)$$

Substituting (2-14b) into (2-14a) with the use of (2-13c) then provides the result

$$\ddot{\Lambda}^* = i\dot{\Lambda}^{*2}(P1^*_i Q^\circ_i + P^\circ_i Q1^*_i)/2 \qquad (2\text{-}14c)$$

after which $\ddot{Q}^*$ is obtained directly from (2-14b).

It is to be noted that a solution for the rates $\dot{\Lambda}^*$, $\ddot{\Lambda}^*$, $\dot{Q}^*$ and $\ddot{Q}^*$ (and higher order rates if desired by similar calculations) requires only a single formation and decomposition of the matrix KO*.

Solution Procedures - Once the load and displacement rates have been determined to a desired order, many different solution procedures can be applied in tracing the nonlinear equilibrium path of the structure. The first order rates allow solution by methods of incremental loading (with or without evaluation of residual forces and corrective iterations), and Newton Raphson iteration where the Jacobian is re-evaluated at each iteration. Various combinations of incrementation and interation, with periodic updating of the Jacobian are of course possible. The second order rates allow the use of a second order predictor. The additional cost of the 2nd order predictor is associated with the P1 psuedo-force term, whose evaluation is performed at the elemental level with a cost roughly proportional to that of a single "residual force" evaluation. The cost of evaluating P1 by the form of (2-8c) is only linearly proportional to the number of integration points within an element, so that this technique is effective even for elements having complex geometry and large degrees of freedom.

Such a predictor has been found to be very useful during the present study, and although the PANES computer program allows use of various predictor-corrector options, the second order predictor almost always appears to be more efficient for cases of substantial nonlinearity. Since the second order rates are valid at any reference equilibrium configuration, they may be applied in a corrector technique, at a state where the system is in "equilibrium" under the applied loads plus a set of unbalanced residual loads. Thus convergence could be considerably accelerated if the second order relations were computed and used at each iteration, although the cost per iteration would also increase considerably. Higher order predictor-corrector relations are obviously possible as well, and the best type of solution capability would probably be a program in which more or less arbitrary options are allowed for the order of predictor and corrector, the frequency with which the Jacobian is updated, and the number of iterations to be performed per update. Although these considerations will not be discussed in any more detail here, the PANES program is at least a step in that direction, and makes available various options using the first and second order rate relations.

Limit Point and Step Size Considerations - A major advantage of a 2nd order predictor is that, with little increase in computational effort, it provides greatly increased prediction accuracy and allows larger load steps to be taken. In addition, it enables the traversing of limit points and provides various techniques for automatic selection of the load step size.

In the vicinity of a limit point, the load rate relation

$$\dot{\Lambda} = \dot{\Lambda}^* + \ddot{\Lambda}^* S \qquad (2\text{-}15a)$$

is used. Also from (2-10a) the path value for given Λ is

$$S = \{ -\dot{\Lambda}^* \pm (\dot{\Lambda}^{*2} + 2\Lambda\ddot{\Lambda}^*)^{1/2}\}/\ddot{\Lambda}^* \geq 0 \qquad (2\text{-}15b)$$

At a limit point $\dot{\Lambda} = 0$, so that from (2-15a) the critical path value is

$$S^c = -\dot{\Lambda}^*/\ddot{\Lambda}^* \qquad (2\text{-}15c)$$

Using these relations the limit point can be traversed when Λ is within some specified fraction of its critical value.

With regard to automatic selection of a general load step size, the following predictor relationships are noted.

$$\Delta\Lambda = \dot{\Lambda}S + \frac{1}{2}\ddot{\Lambda}S^2 \qquad (2\text{-}16a)$$

$$\Delta Q_i = \dot{Q}_i S + \frac{1}{2}\ddot{Q}_i S^2 \qquad (2\text{-}16b)$$

Here the quadratic terms give an indication of the accuracy of the linear predictor, but because of the truncation of higher order terms there is no indication of accuracy for the quadratic predictor. The rationale used in the PANES program implementation is therefore to select a load step size which limits the quadratic contributions to some specified factor times the linear contributions. Specifically, change in slope of the load parameter during a load step is approximated by

$$\Delta\dot{\Lambda} = \ddot{\Lambda}S \qquad (2\text{-}17a)$$

and the ratio of slope change to average slope is

$$\Delta\dot{\Lambda}/\dot{\Lambda}_{\text{average}} = \ddot{\Lambda}S/(\dot{\Lambda} + \tfrac{1}{2}\ddot{\Lambda}\ S) \qquad (2\text{-}17b)$$

This slope ratio is specified as a given allowable magnitude, in order to prevent over-prediction in (2-16a) of the behavior beyond accurate values. A similar step size restriction is employed based on (2-16b) for displacement rates.

3.0 DETERMINATION OF BIFURCATION AND THE POSTBUCKLING PATH

This section considers the identification of bifurcation points in the load-displacement path of a structure, and the prediction of the postbuckling path beyond these points. The formulation follows the appraoch of Section 2 for representing geometric nonlinearities and an arbitrary nonlinear material. The effects of nonconservative load on bifurcation and postbuckling are treated in Appendix B.

3.1 Description of the Postbuckling Path

We consider behavior of the type shown in Figure 3-1, which is a plot of the incremental load parameter Λ for a structure versus its incremental displacements ΔQ, shown here conceptually for a single degree of freedom system. The point O represents a reference equilibrium configuration ($Q = Q^*$, $\Lambda = \Delta Q = 0$). Travel along the "fundamental" and "postbuckling" paths is measured by suitable path parameters S and R, respectively. Thus S has a value of zero at point O, while R takes on a zero value at the critical bifurcation point C.

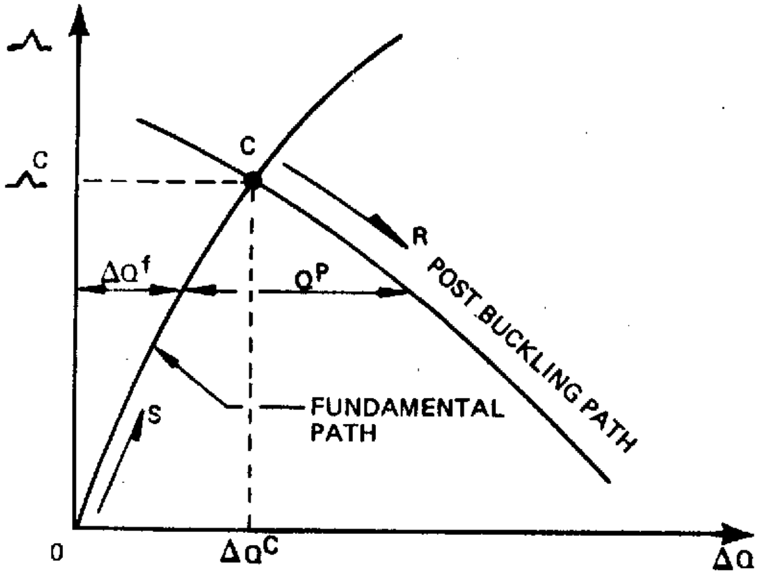


Figure 3-1: Fundamental and Postbuckling Paths

We follow the terminology of Mau and Gallagher (1972) and use a "sliding coordinate" system to describe the various fundamental and postbuckling quantities. For a given value of Λ, a point on the fundamental path has associated quantities whose values are denoted by $(\ )^f$, while <u>additional</u> values at the corresponding point on the postbuckling path are denoted by $(\ )^p$. Thus <u>total</u> values on the postbuckling path are denoted by $(\ )^f + (\ )^p$, and we write for the postbuckling path

$$\left.\begin{aligned} Q &= Q^f + Q^p \\ \Delta Q &= \Delta Q^f + Q^p \\ \Delta\varepsilon &= \Delta\varepsilon^f + \varepsilon^p \\ \sigma &= \sigma^f + \sigma^p \\ &\text{etc.} \end{aligned}\right\} \qquad (3\text{-}1)$$

where the Λ quantities are increments from the fundamental reference configuration.

We will refer to the $(\ )^f$ and $(\ )^p$ values as the "fundamental" and "postbuckling" values, respectively, and to their sums as the "total" values.

3.2 Formulation of Postbuckling Equilibrium Equations

<u>Basic Equilibrium Equation</u> - Because the postbuckling path is an equilibrium path, and equation (2-5b) is valid for a point on any equilibrium path, we may write the postbuckling equilibrium equation as

$$P_i = P_i^f + P_i^p = \int_V B_{ai}\sigma_a \, dV = \int_V (B_{ai}^f + B_{ai}^p)\,(\sigma_a^f + \sigma_a^p)\, dV \qquad (3\text{-}2)$$

Recognizing that $P_i = P_i^f$ for a given value of Λ with conservative loading, and subtracting out terms in equation (3-2) which are zero because they collectively satisfy the fundamental equilibrium equation, provides the desired form of the postbuckling equilibrium equation as

$$P_i^p = 0 = \int_V \{ B_{ai}^f \sigma_a^p + B_{ai}^p (\sigma_a^f + \sigma_a^p) \} \, dV \qquad (3\text{-}3)$$

Derivative Relations - We now record the following typical derivative relations, where a prime ( )' denotes differentiation with respect to the postbuckling path parameter R.

$$\left.\begin{aligned}
\sigma_a^p &= \sigma_a - \sigma_a^f = D0^*_{ab}\varepsilon_b^p + D1^*{}_{abc}(2\varepsilon_b^p\Delta\varepsilon_c^f + \varepsilon_b^p\varepsilon_c^p) \\
\sigma_a^{\prime p} &= D0^*_{ab}\varepsilon_b^{\prime p} + D1^*_{abc}(2\varepsilon_b^{\prime p}\Delta\varepsilon_c^f + 2\varepsilon_b^p\varepsilon_c^{\prime f} + 2\varepsilon_b^p\varepsilon_c^{\prime p}) \\
\sigma_a^{\prime\prime p} &= D0^*_{ab}\varepsilon_b^{\prime\prime p} + D1^*_{abc}(2\varepsilon_b^{\prime\prime p}\Delta\varepsilon_c^f + 4\varepsilon_b^{\prime p}\varepsilon_c^{\prime f} + 2\varepsilon_b^p\varepsilon_c^{\prime\prime f} \\
&\qquad + 2\varepsilon_b^{\prime p}\varepsilon_c^{\prime p} + 2\varepsilon_b^p\varepsilon_c^{\prime\prime p}) \\
\sigma_a^{\prime\prime\prime p} &= D0^*_{ab}\varepsilon_b^{\prime\prime\prime p} + D1^*_{abc}(2\varepsilon_b^{\prime\prime\prime p}\Delta\varepsilon_c^f + 6\varepsilon_b^{\prime\prime p}\varepsilon_c^{\prime f} + 6\varepsilon_b^{\prime p}\varepsilon_c^{\prime\prime f} \\
&\qquad + 2\varepsilon_b^p\varepsilon_c^{\prime\prime\prime f} + 6\varepsilon_b^{\prime\prime p}\varepsilon_c^{\prime p} + 2\varepsilon_b^p\varepsilon_c^{\prime\prime\prime p}) \\
\varepsilon_a^{\prime p} &= \varepsilon_a^{\prime} - \varepsilon_a^{\prime f} = B_{ai}q_i^{\prime} - B_{ai}^f q_i^{\prime f} \\
\varepsilon_a^{\prime\prime p} &= B_{ai}^{\prime}q_i^{\prime} + B_{ai}q_i^{\prime\prime} - B_{ai}^{\prime f}q_i^{\prime f} - B_{ai}^f q_i^{\prime\prime f} \\
\varepsilon^{\prime\prime\prime p} &= B_{ai}^{\prime\prime}q_i^{\prime} + 2B_{ai}^{\prime}q_i^{\prime\prime} + B_{ai}q_i^{\prime\prime\prime} - B_{ai}^{\prime\prime f}q_i^{\prime f} - 2B_{ai}^{\prime f}q_i^{\prime\prime f} \\
&\qquad - B_{ai}^f q_i^{\prime\prime\prime f}
\end{aligned}\right\} \quad (3\text{-}4a)$$

and at the critical point ($\sigma_a = \sigma_a^f$, $B_{ai} = B_{ai}^f$, $\sigma_a^p = B_{ai}^p = 0$), we have

$$\left.\begin{aligned}
\sigma_a^{\prime p} &= D0^*_{ab}\varepsilon_b^{\prime p} + 2D1^*_{abc}\varepsilon_b^{\prime p}\Delta\varepsilon_c^f = D0_{ab}\varepsilon_b^{\prime p} \\
\sigma_a^{\prime\prime p} &= D0^*_{ab}\varepsilon_b^{\prime\prime p} + D^*_{abc}(2\varepsilon_b^{\prime\prime p}\Delta\varepsilon_c^f + 4\varepsilon_b^{\prime p}\varepsilon_c^{\prime f} + 2\varepsilon_b^{\prime p}\varepsilon_c^{\prime p}) \\
&= D0_{ab}\varepsilon_b^{\prime\prime p} + D1_{abc}(4\varepsilon_b^{\prime p}\varepsilon_c^{\prime f} + 2\varepsilon_b^{\prime p}\varepsilon_c^{\prime p}) \\
\sigma_a^{\prime\prime\prime p} &= D0^*_{ab}\varepsilon_b^{\prime\prime\prime p} + D1^*_{abc}(2\varepsilon_b^{\prime\prime\prime p}\Delta\varepsilon_c^f + 6\varepsilon_b^{\prime\prime p}\varepsilon_c^{\prime f} \\
&\qquad + 6\varepsilon_b^{\prime p}\varepsilon_c^{\prime\prime f} + 6\varepsilon_b^{\prime\prime p}\varepsilon_c^{\prime p}) \\
&= D0_{ab}\varepsilon_b^{\prime\prime\prime p} + D1_{abc}(6\varepsilon_b^{\prime\prime p}\varepsilon_c^{\prime f} + 6\varepsilon_b^{\prime p}\varepsilon_c^{\prime\prime f} \\
&\quad + 6\varepsilon_b^{\prime\prime p}\varepsilon_c^{\prime p}) \\
\varepsilon_a^{\prime p} &= B^f_{ai}q_i^{\prime p} = B^*_{ai}q_i^{\prime p} + A1_{amn}\theta_m^{\prime p}\Delta\varepsilon_n^f \\
\varepsilon_a^{\prime\prime p} &= B^f_{ai}q_i^{\prime\prime p} + B'_{ai}q'_i - B'^f_{ai}q'^f = B^f_{ai}q_i^{\prime\prime p} \\
&\quad + A1_{amn}(2\theta_m^{\prime p}\theta_n^{\prime f} + \theta_m^{\prime p}\theta_n^{\prime p}) \\
\varepsilon_a^{\prime\prime\prime p} &= B^f_{ai}q_i^{\prime\prime\prime p} + B''_{ai}q'_i + 2B_{ai}q''_i - B''^f_{ai}q_i^{\prime f} - 2B'^f_{ai}q_i^{\prime\prime f} \\
&= B^f_{ai}q_i^{\prime\prime\prime p} + A1_{amn}(3\theta_m^{\prime\prime p}\theta_n^{\prime f} + 3\theta_m^{\prime p}\theta_n^{\prime\prime f} \\
&\quad + 3\theta_m^{\prime\prime p}\theta_n^{\prime p}
\end{aligned}\right\} \tag{3-4b}$$

<u>First Order Equilibrium Equation</u> - Differentiating the basic postbuckling equilibrium equation (3-3), and evaluating at the critical bifurcation point, gives

$$P_i^{\prime p} = 0 = \int_V (B^f_{ai}\sigma_a^{\prime p} + B'^p_{ai}\sigma_a)\ dV \tag{3-5a}$$

Substituting for σ'^{p} and B'^{p} gives

$$0 = \int_V \{B^f_{ai}(D0^*_{ab}\varepsilon'^{p}_b + 2D1^*_{abc}\varepsilon'^{p}_b\Delta\varepsilon^f_c) + G_{mi}A1_{amn}\theta'^{p}_n\sigma_a\}\, dV \qquad (3\text{-}5b)$$

Substituting for B^f_{ai}, and using the relations which express θ'^{p} and ε'^{p} in terms of q'^{p}, gives

$$\begin{aligned} 0 = \int_V [(B^*_{ai} + G_{mi}A1_{amn}\Delta\theta^f_n)\{D0^*_{ab}(B^*_{bj}q'^{p}_j + A1_{bns}\theta'^{p}_n\Delta\theta^f_s) \\ + 2D1^*_{abc}\varepsilon'^{p}_b\Delta\varepsilon^f_c\} + G_{mi}A1_{amn}\{G_{nj}q'^{p}_j\sigma_a{}^* \\ + \theta'^{p}_n(D0^*_{ab}\Delta\varepsilon^f_b + D1^*_{abc}\Delta\varepsilon^f_b\Delta\varepsilon^f_c)\}]\, dV \end{aligned} \qquad (3\text{-}5c)$$

This is the first order postbuckling equilibrium equation, which may be written in the form

$$0 = K0^*_{ij}\; Q'^{p}_j + P1^1_i \qquad (3\text{-}5d)$$

where again

$$K0^*_{ij} = \int_V (G_{mi}\sigma^*_a A1_{amn}G_{nj} + B^*_{ai}D0^*_{ab}B^*_{bj})\, dV$$

and

$$\begin{aligned} P1^1_i = \int_V G_{mi}\{D0^*_{ab}(A^*{}_{am}A1_{bns}\theta'^{p}_n\Delta\theta^f_s + A1_{amn}\varepsilon'^{p}_b\Delta\theta^f_n \\ + A1_{amn}\Delta\varepsilon^f_b\theta'^{p}_n) + D1^*_{abc}(2A^*_{am}\varepsilon'^{p}_b\Delta\varepsilon^f_c + 2A1_{amn}\varepsilon'^{p}_b\Delta\varepsilon^f_c\Delta\theta^f_n \\ + A1_{amn}\Delta\varepsilon^f_b\Delta\varepsilon^f_c\theta'^{p}_n)\}dV \end{aligned}$$

Equation (3-5d) is an eigenequation form, in terms of the unknown critical values $\Delta\theta$ and $\Delta\varepsilon$, which is suitable for solution by power iteration.

Alternatively, the eigenequation may be written in the form

$$0 = K0_{ij}Q'^{p}_{j} = (K0^{*}_{ij} + \Delta K0_{ij})Q'^{p}_{j} \qquad (3\text{-}5e)$$

where K0 is the Jacobian at the critical point, and is given by

$$K0_{ij} = \int_V (G_{mi}\sigma_a Al_{amn}G_{nj} + B_{ai}D0_{ab}B_{bj})\ dV$$

with the σ, D0 and B quantities evaluated at the critical point. (3-5e) may be solved directly for the eigenvector Q'^{p}, provided that the critical values of σ, D0 and B have been previously determined (as in the method proposed by Mau and Gallagher (1972). This equation may also be solved by expressing $\Delta K0$ as a Taylor series expansion in the fundamental path parameter, giving a form suitable for solution by one of the many "direct" eigensolution methods.

Second Order Equilibrium Equation - A second differentiation of the postbuckling equilibrium equation (3-3) and evaluation at the critical point, gives

$$P''^{p}_{i} = 0 = \int_V \{2B'^{f}_{ai}\sigma'^{p}_{a} + B_{ai}\sigma_a''^{p} + B''^{p}_{ai}\sigma_a + 2B'^{p}_{ai}(\sigma'^{f}_{a} + \sigma'^{p}_{a})\ dV \qquad (3\text{-}6a)$$

Substituting for σ''^{p} and B''^{p} gives

$$0 = \int_V [2G_{mi}Al_{amn}\theta'^{f}_{n}D0_{ab}\varepsilon'^{p}_{b} + B_{ai}\ D0_{ab}\varepsilon''^{p}_{b} + D1_{abc}(4\varepsilon'^{p}_{b}\varepsilon'^{f}_{c} + 2\varepsilon'^{p}_{b}\varepsilon'^{p}_{c})\} + G_{mi}Al_{amn}\theta''^{p}_{n}\sigma_a + 2G_{mi}Al_{amn}\theta'^{p}_{n} D0_{ab}(\varepsilon'^{f}_{b} + \varepsilon'^{p}_{b})]\ dV \qquad (3\text{-}6b)$$

Using the relations which express θ''^p and ε''^p in terms of q''^p gives

$$0 = \int_V \, [2G_{mi}Al_{amr}\theta_r'^f D0_{ab}\varepsilon_b'^p + B_{ai}D0_{ab}B_{bj}q_j''^p + G_{mi}A_{am}$$

$$\{D0_{ab}(2Al_{bns}\theta_n'^p\theta_s'^f + Al_{bns}\theta_n'^p\theta_s'^p) + D1_{abc}(4\varepsilon_b'^p\varepsilon_c'^f + 2\varepsilon_b'^p\varepsilon_c'^p)\}$$

$$+ G_{mi}\sigma_a Al_{amn}G_{nj}q_j''^p + 2G_{mi}Al_{amn}\theta_n'^p\{D0_{ab}(\varepsilon_b'^f + \varepsilon_b'^p)\}]\,dV \qquad (3\text{-}6c)$$

This is the second order postbuckling equilibrium equation, which may be written in the form

$$0 = K0_{ij}Q_j''^p + 2S'P2_i^1 + P2_i^2 \qquad (3\text{-}6d)$$

where K0 is again the Jacobian evaluated at the critical point, and

$$P2_i^1 = \int_V G_{mi}\,\{D0_{ab}(A_{am}Al_{bns}\theta_n'^p\dot{\theta}_s^f + Al_{amr}\varepsilon_b'^p\dot{\theta}_r^f + Al_{amr}\dot{\varepsilon}_b^f\theta_r'^p)$$

$$+ 2D1_{abc}A_{am}\varepsilon_b'^p\dot{\varepsilon}_c^f\;\}\;dV$$

$$P2_i^2 = \int_V G_{mi}\,\{D0_{ab}(A_{am}Al_{bns}\theta_n'^p\theta_s'^p + 2Al_{amr}\varepsilon_b'^p\theta_r'^p)$$

$$+ 2D1_{abc}A_{am}\varepsilon_b'^p\varepsilon_c'^p\}\;dV$$

The term S' in equation (3-6d) is the derivative of the fundamental path parameter with respect to the postbuckling path parameter, and occurs because of the substitutions

$$\theta_n'^f = \dot{\theta}_n^f S'$$

$$\varepsilon_a'^f = \dot{\varepsilon}_a^f S'$$

(3-6d) may be solved for the postbuckling displacement second derivatives Q''^p, and for the path derivative S'.

Third Order Equilibrium Equation - A third differentiation of equation (3-3) and evaluation at the critical point, gives

$$p_i^{'''p} = 0 = \int_V \{B_{ai}^{'''f}\sigma_a^p + 3B_{ai}^{''f}\sigma_a^{'p} + 3B_{ai}^{'f}\sigma_a^{''p} + B_{ai}^f\sigma_a^{'''p}$$

$$+ B_{ai}^{'''p}(\sigma_a^f+\sigma_a^p) + 3B_{ai}^{''p}(\sigma_a^{'f}+\sigma_a^{'p}) + 3B_{ai}^{'p}(\sigma_a^{''f}+\sigma_a^{''p})$$

$$+ B_{ai}^p(\sigma_a^{'''f}+\sigma_a^{'''p})\}\ dV \qquad (3\text{-}7a)$$

Substituting for $\sigma^{'''p}$ and $B^{'''p}$ gives

$$0 = \int_V [3G_{mi}\ Al_{amn}\ \theta_n^{''f}\ DO_{ab}\ \varepsilon_b^{'p} + 3G_{mi}\ Al_{amn}\ \theta_n^{'f}\{DO_{ab}\ \varepsilon_b^{''p}$$

$$+ Dl_{abc}\ (4\varepsilon_b^{'p}\varepsilon_b^{'p} + 2\varepsilon_b^{'p}\varepsilon_c^{'p})\}$$

$$+ B_{ai}\ DO_{ab}\varepsilon_b^{'''p} + Dl_{abc}\ (6\varepsilon_b^{''p}\varepsilon_c^{'f} + 6\varepsilon_b^{'p}\varepsilon_c^{''f} + 6\varepsilon_b^{''p}\varepsilon_c^{'p})\}$$

$$+ G_{mi}Al_{amn}\theta_n^{'''p}\sigma_a$$

$$+ 3G_{mi}Al_{amn}\theta_n^{''p}\ DO_{ab}(\varepsilon_b^{'f} + \varepsilon_b^{'p}) + 3G_{mi}Al_{amn}\theta_n^{'p}\{DO_{ab}(\varepsilon_b^{''f} + \varepsilon_b^{''p})$$

$$+ Dl_{abc}(2\varepsilon_b^{'f}\varepsilon_c^{'f} + 4\varepsilon_b^{'p}\varepsilon_c^{'f} + 2\varepsilon_b^{'p}\varepsilon_c^{'p})\}]\ dV \qquad (3\text{-}7b)$$

Using the relations which express $\theta^{'''p}$ and $\varepsilon^{'''p}$ in terms of $q^{'''p}$ gives

$$0 = \int_V 3G_{mi}Al_{amn}\theta_n^{''f}DO_{ab}\varepsilon_b^{'p} + 3G_{mi}Al_{amn}\theta_n^{'f}\{DO_{ab}\varepsilon_b^{''p} + Dl_{abc}(4\varepsilon_b^{'p}\varepsilon_c^{'f}$$

$$+ 2\varepsilon_b^{'p}\varepsilon_c^{'p})\} + B_{ai}DO_{ab}B_{bj}q_j^{'''p} + G_{mi}A_{am}\{DO_{ab}(3Al_{bns}\theta_n^{''p}\theta_s^{'f}$$

$$+ 3Al_{bns}\theta_n^{'p}\theta_s^{''f} + 3\theta_m^{''p}\theta_n^{'p}) + Dl_{abc}(6\varepsilon_b^{''p}\varepsilon_c^{'f} + 6\varepsilon_b^{'p}\varepsilon_c^{''f}$$

$$+ 6\varepsilon_b^{''p}\varepsilon_c^{'p})\} + G_{mi}\sigma_a Al_{amn}G_{nj}q_j^{'''p} + 3G_{mi}Al_{amn}\theta_n^{''p}\ DO(\varepsilon_b^{'f} + \varepsilon_b^{'p})$$

$$+ 3G_{mi}A1_{amn}\theta_n^{\prime p}\{DO_{ab}(\varepsilon_b^{\prime\prime f} + \varepsilon_b^{\prime\prime p}) + D1_{abc}(2\varepsilon_b^{\prime f}\varepsilon_c^{\prime f} + 4\varepsilon_b^{\prime p}\varepsilon_c^{\prime f}$$

$$+ 2\varepsilon_b^{\prime p}\varepsilon_c^{\prime p})\}]dV \qquad (3\text{-}7c)$$

This is the third order postbuckling equilibrium equation, which may be written in the form

$$0 = KO_{ij}Q_j^{\prime\prime\prime p} + 3S^{\prime\prime}P2_i^1 + 3P3_i \qquad (3\text{-}7d)$$

where

$$P3_i = \int_V G_{mi}[DO_{ab}\{S^{\prime 2}(A_{am}A1_{bns}\theta_n^{\prime p}\ddot{\theta}_s^f + A1_{amr}\varepsilon_b^{\prime p}\ddot{\theta}_r^f$$

$$+ A1_{amr}\ddot{\varepsilon}_b^f\theta_r^{\prime p}) + S^{\prime}(A_{am}A1_{bns}\dot{\theta}_n^f\theta_s^{\prime\prime p} + A1_{amr}\,\dot{\varepsilon}_b^f\theta_r^{\prime\prime p} + A1_{amr}\varepsilon_b^{\prime\prime p}\dot{\theta}_r^f)$$

$$+ (A_{am}A1_{bns}\theta_n^{\prime p}\theta_s^{\prime\prime p} + A1_{amr}\varepsilon_b^{\prime p}\theta_r^{\prime\prime p} + A1_{amr}\varepsilon_b^{\prime\prime p}\theta_r^{\prime p})$$

$$+ 2D1_{abc}\{S^{\prime 2}(A_{am}\varepsilon_b^{\prime p}\ddot{\varepsilon}_c^f + A1_{amr}\dot{\varepsilon}_b^f\dot{\varepsilon}_c^f\theta_r^{\prime p} + 2A1_{amr}\varepsilon_b^{\prime p}\dot{\varepsilon}_c^f\dot{\theta}_r^f)$$

$$+ S^{\prime}(A_{am}\dot{\varepsilon}_b^f\varepsilon_c^{\prime\prime p} + A1_{amr}\varepsilon_b^{\prime p}\varepsilon_c^{\prime p}\dot{\theta}_r^f + 2A1_{amr}\dot{\varepsilon}_b^f\varepsilon_c^{\prime p}\theta_r^{\prime p})$$

$$+ (A_{am}\varepsilon_b^{\prime p}\varepsilon_c^{\prime\prime p} + A1_{amr}\varepsilon_b^{\prime p}\varepsilon_c^{\prime p}\theta_r^{\prime p})\}]dV$$

The term S'' occurs in (3-7d) after making the substitutions

$$\theta_n^{\prime f} = \dot{\theta}_n^f S^{\prime}$$

$$\theta_n^{\prime\prime f} = \dot{\theta}_n^f S^{\prime\prime} + \ddot{\theta}_n^f S^{\prime 2}$$

$$\varepsilon_a^{\prime f} = \dot{\varepsilon}_a^f S^{\prime}$$

$$\varepsilon_a^{\prime\prime f} = \dot{\varepsilon}_a^f S^{\prime\prime} + \ddot{\varepsilon}_a^f S^{\prime 2}$$

Equation (3-7d) may be solved for the postbuckling displacement second derivatives Q'''^p, and for the path second derivative S''.

3.3 Solution of Postbuckling Equilibrium Equations

The postbuckling equilibrium equations (3-5d, 6d and 7d) may be solved sequentially to yield the displacement and load derivatives necessary for construction of the postbuckling path. These equations have been formulated here for the general case of an unsymmetric K0 Jacobian matrix, and the effects of nonconservative loading are discussed in Appendix B. The solution of the second and higher order equations for the unsymmetric case present some practical difficulties, however. Therefore, in contrast to the general solution outlined in Section 2 for the fundamental equations, the solution given here for the postbuckling equations will be presented for the case of a symmetric K0 matrix.

First Order (Bifurcation) Solution - The first order equation (3-5d) may be solved for the eigenvector Q'^p of postbuckling displacements, and for the critical value of the fundamental path parameter S. The initial step is to relate the unknown critical displacement increments ΔQ to the eigenvalue S, using the previously computed fundamental displacement derivatives:

$$Q_i^f = \dot{Q}_i^* S + \frac{1}{2}\ddot{Q}_i^* S^2 \qquad (3\text{-}8)$$

In addition to the nonlinearity inherent in this relation, the eigenequation is nonlinear for other reasons:

1. Although increments in the displacement derivatives $\Delta\theta^f$ and displacements ΔQ^f are linearly proportional, the strain increments vary nonlinearly, i.e.
$\Delta\varepsilon_a^f = A^*_{am}\Delta\theta_m^f + \frac{1}{2} A1_{amn}\Delta\theta_m^f\Delta\theta_n^f$.
2. There are Δ^2 terms ($\Delta\varepsilon^f\Delta\theta^f$, $\Delta\varepsilon^f\Delta\varepsilon^f$) in the eigenequation

due to consideration of the nonlinear material effects (effects of the D1* matrix).

Because of these nonlinearities an eigensolution by direct iteration may not converge. Particular difficulty may be expected during the first few iterations, when the estimated eigenvector contains significant proportions of higher modes for which the $\Delta\theta^f$ at some locations in the structure could be much larger than the corresponding θ^*. Also for such higher modes, the contribution to ΔQ by S^2 may be large and the Δ^2 terms may be large relative to Δ terms. It is therefore necessary to solve first the linear eigen problem, obtained by dropping all nonlinear terms. When convergence has been achieved to within a specified accuracy, iteration is continued with inclusion of all terms until convergence to the desired nonlinear eigensolution.

Higher Order Solutions - With the critical point now defined by the critical value of S, the higher order postbuckling equations may be solved by formation and decomposition of the critical point Jacobian K0. To accomplish the solution, a definition of the postbuckling path parameter R is required. We here follow the general approach of Mau and Gallagher (1972) and take R to be one of the postbuckling displacements, say Q_m^p. In the PANES program, m is taken as the index of the largest component of the eigenvector Q'^p. We then impose the requirements at every path point, R, that

$$\begin{aligned} R' &= 1 \\ R'' = R''' &= \ldots = 0 \end{aligned} \qquad (3\text{-}9)$$

Although the matrix K0 is singular, this constraint of the mth degree of freedom allows the matrix to be decomposed. A somewhat different approach than this is suggested by Haftka et al. (1970), involving the introduction of an additional constraint equation to make the K0 matrix effectively nonsingular. That approach however increases the size of the matrix, and no real advantage is seen. The present approach retains the sparsity of K0.

At this point the eigenvector Q'^{p} is again determined, using the constrained K0 matrix. This is done to achieve consistency in the calculation of Q'^{p} and the higher order derivatives determined later, as well as for greater accuracy. In terms of the symbolic inverse $K0^{-1}$:

$$0 = K0_{ij}Q'^{p}_{j} \text{ , with } Q'^{p}_{m} = 1 \tag{3-10a}$$

$$Q'^{p}_{i} = K0^{-1}_{ij}(0) \tag{3-10b}$$

The second order equation is

$$0 = K0_{ij}Q''^{p}_{j} + 2S'P2^{1}_{j} + P2^{2}_{i} \text{ , with } Q''^{p}_{m} = 0 \tag{3-11a}$$

Premultiplying by Q'^{p}, and using the symmetry of K0 with $K0_{ij}Q'^{p}_{j}$ $= 0$, results in

$$0 = Q'^{p}_{i}K0_{ij}Q''^{p}_{j} + 2S'Q'^{p}_{i}P2^{1}_{i} + Q'^{p}_{i}P2^{2}_{i} = 2S'Q'^{p}_{i}P2^{1}_{i} + Q'^{p}_{i}P2^{2}_{i} \tag{3-11b}$$

which then gives

$$S'^{c} = -Q'^{p}_{i}P2^{2}_{i}/2Q'^{p}_{i}P2^{1}_{i} \tag{3-11c}$$

and $$Q''^{p}_{i} = K0^{-1}_{ij}(-2S'P2^{1}_{j} - P2^{2}_{j}) \equiv -2S'Q2^{1}_{i} - Q2^{2}_{i} \tag{3-11d}$$

The third order equation is

$$0 = K0_{ij}Q'''^{p}_{j} + 3(S''P2^{1}_{i} + P3_{i}) \text{, with } Q'''^{p}_{m} = 0 \tag{3-12a}$$

Premultiplying by Q'^{p} as before, results in

$$0 = S''Q'^{p}_{i}P2^{1}_{i} + Q'^{p}_{i}P3_{i} \tag{3-12b}$$

which then gives

$$S'' = -Q'^{p}_{i}P3_{i}/Q'^{p}_{i}P2^{1}_{i} \qquad (3\text{-}12c)$$

and $Q'''^{p}_{i} = K0^{-1}_{ij}(-3S''P2^{1}_{i} - 3P3_{i}) \equiv -3(S''Q2^{1}_{i} + 3Q3_{i})$ (3-12d)

With the critical point derivatives of S and Q^{p} known, the postbuckling path can be constructed. The variation of load parameter Λ with postbuckling path R, is defined by

$$\begin{aligned} \Lambda' &= \dot{\Lambda}S' \\ \Lambda'' &= \dot{\Lambda}S'' + \ddot{\Lambda}S'^{2} \end{aligned} \qquad (3\text{-}13)$$

4.0 FINITE ELEMENT PROGRAM AND NONLINEAR SOLUTION ROUTINES

4.1 General Program Characteristics

The major goal of this research effort was the development of improved nonlinear solution techniques and subroutines. It was decided that the most effective way of accomplishing this goal was to develop a practical nonlinear finite element program, into which the various subroutines could be incorporated for checkout and verification. This has been accomplished, and the resulting finite element program has been given the acronym PANES (Program for Analysis of Nonlinear Equilibrium and Stability). Although PANES is a pilot program and is by no means a general structural analyzer (it utilizes only the constant strain triangle element, for 2-D in-plane or 3-D membrane analysis) it demonstrates all of the basic techniques and operations necessary for nonlinear analysis by more general types of finite elements. The program handles geometric nonlinearities and arbitrary nonlinear elastic materials (including very large strain cases), as well as certain forms of nonconservative loadings, i.e. those due to follower-force pressure loadings where the surfaces change in size and orientation. Extension of the program to cases of inelastic materials is considered to be relatively straightforward, with the introduction of appropriate stress-strain constitutive relations.

The present pilot version of PANES has three basic capabilities:

1. Analysis of nonlinear structures without critical points, i.e. tracing of simple nonlinear equilibrium paths under a specified general (non-proportional) loading. Various solution techniques are available, with automatic calculation of load step sizes.

2. Traversing of limit (maximum and minimum) type critical points, with automatic continuation of the load-path history.

3. Determination of bifurcation type critical points, and prediction of the postbuckling behavior and direction of travel, by means of path derivatives computed at the bifurcation point. Automatic switching from the fundamental path to a postbuckling path, and continuation along the postbuckling path, have not yet been included. Thus the postbuckling capability should be regarded as still in a developmental stage.

4.2 PANES Nonlinear Analysis Routines

This section describes briefly the purpose and capabilities of the program subroutines, in the order in which they appear in the PANES program. Some of these are basic finite element routines, while others are specialized routines needed for generating and solving nonlinear structural equations.

BIGS - Initializes program variables (serves as the calling subroutine for most of the input data reading routines). Also provides problem restart capability by reading or writing the restart tape.

READRS - Reads data file numbers and start or restart codes.

READO - Reads problem identification title. Also reads incremental and iterative constants, such as those relating to the predictor and corrector types, the finite difference expansions for nonlinear materials, and the techniques for continuation of the equilibrium path through limit points.

READ1 - Reads basic structural codes and values, and material constants for each material.

READC - Reads user-defined special nodal coordinate systems.

READM - Reads mesh data, including nodal locations and coordinate system codes, and element data.

READK - Reads codes to determine degrees of freedom with specified forces, displacements or constraints.

READP - Reads two load reference curves which define distribution of the applied generalized nodal loads.

READPR - Reads a pressure load reference curve which defines the distribution of the applied pressure loads (one intensity for each element).

READI - Reads incremental load data, including the nodal load and pressure load curve factors for the total load at the end of each increment.

HEAD - Writes a heading output for each load increment step, including load parameter value, number of iterations required and accuracy achieved.

OUTLIM - Predicts and outputs limit point values for the load parameter, and nodal forces and displacements.

OUTPQ - Outputs nodal forces and displacements.

OUTE - Outputs element strains.

QFILL - Uses vector of system-level nodal displacements Q to form vector of element-level nodal displacements q for an element.

PFILL - Takes vector of element-level nodal forces p and adds them to system-level force vector P.

DFILL - Uses element nodal displacement vector q to compute vector of displacement derivatives θ within the element.

EFILL - Uses element displacement derivatives vector θ to compute vector of strains ε within the element.

AFILL - Uses element displacement derivatives θ to form Lagrangian A0 or A1 matrix within the element.

GFILL - Uses element displacement functions to form the θ-q transformation matrix G.

MTRAN - Matrix transformation routine, which performs operations of the type $K_{ij} = D_{ab}B_{ai}B_{bj}$ for given D and B matrices.

ROTQ - Transforms element displacements or forces, from either nodal to element or element to nodal coordinate system.

ROTK - Transforms element stiffness matrix from element to nodal coordinate system.

FORCE - Computes internal nodal generalized forces corresponding to given nodal displacements.

PFORCE - Computes applied nodal force loadings, using nodal load reference curves and corresponding load factors.

EFORCE - Computes nodal forces due to applied pressure loadings, using pressure reference curve and factor, and the current area and orientation of each element (determined from geometry and current displacements).

ERCOMP - Computes and outputs error norm for each residual force iteration, using applied (external) forces and computed (internal) forces.

STRAIN - Computes strains for each element using geometry and nodal displacements.

ENERGY - Evaluates the strain energy density for an element at given strain components. This routine will in general be a user supplied routine based on the types of materials being used in the structure.

EVAL - Performs the function (strain energy) evaluations at the current strain state, and at the required adjacent "perturbed" states necessary to establish a strain energy expansion in terms of incremental strains. EVAL calls the STRAIN routine for evaluation, and defines the evaluation points by using the user-specified finite difference sizes. A first, second or third order expansion may be specified, and the corresponding function evaluations are returned in the form of a vector.

U2FORM - Forms coefficients for a general second order Taylor series expansion, using function values provided by EVAL. Used to develop the strain-energy related tensors σ_i and DO_{ij} for a material at current deformation state.

U3FORM - Similar to U2FORM, but forms coefficients for a general third order expansion. Develops the tensors σ_i, DO_{ij} and Dl_{ijk}.

UFILL - Calling routine which calls either U2FORM or U3FORM, depending on desired expansion order.

CFORM - Forms the contribution to the Jacobian stiffness matrix due to the nonconservative pressure loadings.

GENER8 - Generates the elemental Jacobian matrix using the current geometry and the material tensors σ_i and DO_{ij}. Also adds contributions from CFORM if loading is nonconservative.

USUM1 - Performs a summing operation between a second or third order tensor function and its vector argument, to give a vector.

USUM21 - Performs a summing operation between a third order tensor function and its two (different) vector arguments, to give a vector.

P1COMP - Computes the nonlinear load term P1*, required in generating the second order fundamental equilibrium equations.

RATES - Computes the first and second order fundamental load parameter and displacement rates.

STEP - Provides automatic calculation of a fundamental path load step size, and techniques for traversing limit points.

EIGEN1 - Computes the psuedo force term $P1^1$, for use in the inverse power iteration eigensolution process.

EIGEN - Eigensolution routine for inverse power iteration. Calls EIGEN1 routine.

POST2 - Computes the second order postbuckling psuedo force terms $P2^1$ or $P2^2$.

POST3 - Computes the third order postbuckling psuedo force term P3.

PRATES - Computes the first and second order postbuckling load and displacement rates, and third order displacement rate, at the bifurcation point.

VDOT, VCROSS, VLENTH, VNORM - Vector subroutines for computing dot product, cross product, length, and normalizing a vector, respectively.

MERGE - Merges elemental Jacobians into system Jacobian, with provision for constrained degrees of freedom. Forms general unsymmetric Jacobian matrix.

DECOMP - Decomposes unsymmetric Jacobian using Gauss wavefront type procedure. Takes advantage of sparsity but uses total square matrix for storage without packing or external storage devises. (This is a small pilot version decomposition routine.)

SOLVE - Performs forward and backward substitution for unsymmetric Jacobian matrix to provide solution vectors.

4.3 Summary of PANES Input Data

A pictorial of the PANES input deck is shown in Figure 4-1. The input data consists of the following three general types:

Type C:	Data on the usual card file. These are data which are needed for each start or restart.
Type I:	Data on File I. These are basic structural data for a given problem, such as material properties and mesh data. They are the same for all load increments and are needed only when starting.
Type II:	Data on File II. File II is not used in the current PANES version. It is provided for possible future use as a file of incremental data (e.g. additional nodal and thermal load data).

The data included on each file are described below. Formats are consistent with FORTRAN IV conventions.

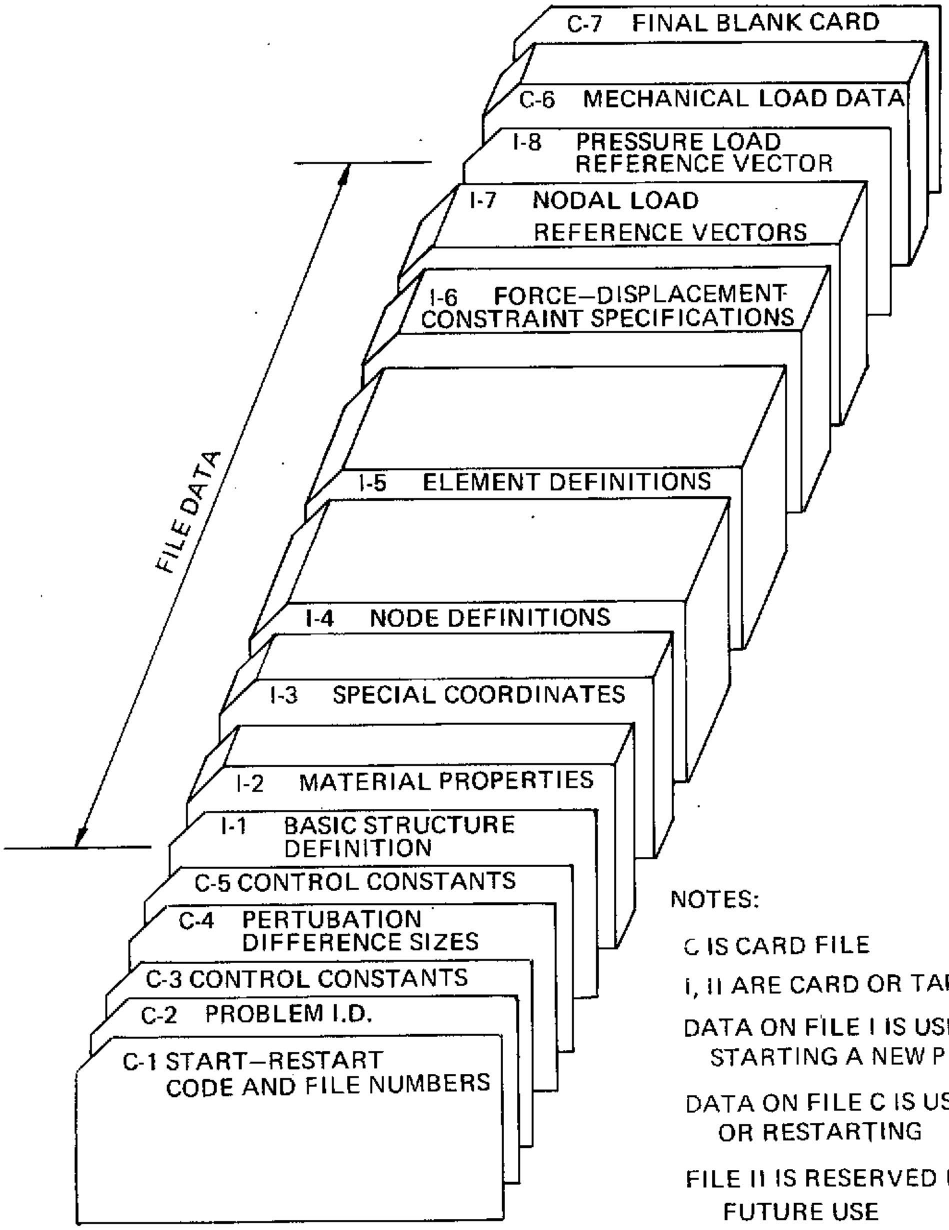
C-1. Start-restart code and data file numbers:

a. "START" if new problem, or "RESTART" if restarting.

b. If starting give unit number for file I.

c. Unit number for file II (need not be given).

d. Unit number for output file (e.g. printer).


C-7 FINAL BLANK CARD
C-6 MECHANICAL LOAD DATA
I-8 PRESSURE LOAD REFERENCE VECTOR
I-7 NODAL LOAD REFERENCE VECTORS
I-6 FORCE–DISPLACEMENT CONSTRAINT SPECIFICATIONS
I-5 ELEMENT DEFINITIONS
I-4 NODE DEFINITIONS
I-3 SPECIAL COORDINATES
I-2 MATERIAL PROPERTIES
I-1 BASIC STRUCTURE DEFINITION
C-5 CONTROL CONSTANTS
C-4 PERTUBATION DIFFERENCE SIZES
C-3 CONTROL CONSTANTS
C-2 PROBLEM I.D.
C-1 START–RESTART CODE AND FILE NUMBERS
FILE DATA

NOTES:

C IS CARD FILE

I, II ARE CARD OR TAPE FILES

DATA ON FILE I IS USED ONLY WHEN STARTING A NEW PROBLEM.

DATA ON FILE C IS USED WHEN STARTING OR RESTARTING

FILE II IS RESERVED FOR POSSIBLE FUTURE USE

Figure 4-1: PANES INPUT DECK SETUP

e. If restarting give load increment number from the end of which a restart is to be made.

f. If restarting give input restart-tape unit number.

g. If data is to be saved for future restart give output restart-tape unit number.

Format (A4, 6X, 6I5)

C-2. Problem I.D. title.

Format (20A4).

C-3. Program control constants (any constant left blank is assigned a default value):

a. Specified order of material incremental stress-strain expansion to be used (2 is exact for linear material, maximum order is 3). Default order is 3.

b. Solution predictor type. Type 1 = 1st order, Type 2 = 2nd order.

Default type = 2.

c. Maximum number of Jacobian updates per load increment step.

Default = 0.

d. Maximum number of residual force iterations per Jacobian update.

Default = 5.

e. Maximum allowable residual force error norm.

Default = 1×10^{-8}.

Format (4I10, F10.0)

C-4. Perturbation difference magnitudes for evaluating strain energy.

a. Difference for computing stiffnesses.

Default = 1×10^{-3}.

b. Difference for computing forces.

Default = 1×10^{-8}.

Format (2F10.0)

C-5. Program control constants

a. Number of increment subdivisions to be performed as load nears a limit value. Default = 3.

b. Ratio of limit load to load increment values at which limit point is to be traversed. Default = 0.1.

c. Increment step size limitation, computed from slope of load parameter versus path parameter curve, and equal to change in slope divided by average slope.

Default = 0.5.

d. Maximum load increment step size (used especially in unloading), and defined as a factor times the specified load increment.

Default = 1.0.

e. Maximum fraction of current load increment by which load is allowed to reduce after passing a maximum limit point.

Format (I10, 4F10.0)

I-1. Basic structure definition

a. Code for element pressure loads. Code 0 = no pressures, Code 1 = pressures. Default code is 0.

b. Degree of freedom per node (2 or 3). No default value.

c. Default thickness for all elements.

Format (2I10, F10.0)

I-2. Material property definitions. For each material give material I.D. number, and 2 material constants for use by the strain energy evaluation routine.

Format (I10, 2F10.0)

Blank card after data for last material.

I-3. For each special Cartesian coordinate system: the identification number (integer $\geq$ 2) and counter-clockwise angle (degrees) from basic system X-axis to special system x-axis.

Format (I10, F10.0)

Blank card after last coordinate system.

I-4. For each node: Node number; identification number of coordinate system to define location; X, Y and Z (or R, θ and Z); identification number of coordinate system to define displacements. (Coordinate I.D. number 0 implies the basic Cartesian system, 1 implies the basic cylindrical system).

Format (2I5, 3F10.0, I5)

Blank card after last node.

I-5. For each element: element number, material number, thickness, three node numbers (counter-clockwise order).

If thickness is left blank, default value from I-lc is used.

Format (2I5, F10.0, 3I5)

Blank card after last element.

I-6. For each DOF with specified displacement or constraint:

If specified displacement, give node number and component (1, 2 or 3) number;

If specified constraint, give node and component number, and independent node and component to which DOF is constrained (independent component number is + for specified force, - for specified displacement). User has option of from 1 to 4 values per card.

Format (4(4I5))

I-7. Nodal load reference vectors.

Number of vectors (for current program version must be 2)

Format (I10)

For each nonzero component of load vector: node number, component number (1 = X or R, 2 = Y or θ, 3 = Z), force or displacement value. User has option of from 1 to 4 values per card.

Format (4(2I5, F10.0))

Blank card after last value of each vector.

I-8. Pressure Load Reference Vector. (Input only if pressure code in data item I-1 is nonzero.

Number of vectors (for current program version must be 1)

Format (I10)

For each nonzero component of pressure load vector: element number, pressure intensity. User has option of from 1 to 4 values per card.

Format (4(I10, F10.0))

Blank card after last value of vector.

C-6. Incremental load data

Number of load increments

Format (I10)

For each load increment: solution predictor type (if left blank, value from C-3b is used), the cumulative factors to be applied to the nodal load reference vectors, pressure value for all elements. Pressure is applied in element positive z-coordinate direction.

Format (I10, 3F10.0)

C-7. Final blank card.

Problems may be run consecutively (first data item for each problem follows immediately after last item of preceding problem). Final blank card indicates that all problems have been read.

4.4 Summary of PANES Output

The description of PANES output is conveniently divided into two parts. The first is primarily an echo check of the input data, and the second part consists of output results for each load increment.

4.4.1 Echo Check of Input Data

Initial Output - The first page of PANES output for a problem is essentially an echo check of input items C-1 to C-5, I-1 and I-2. An indication is given as to whether the problem is being started or restarted. If it is restarted then the previous increment number is given, from the end of which the restart is progressing. Next the problem I.D. title is printed, followed by the various control constants and finite difference magnitudes (DFE and DFF). The limit point related control constants (MJUMP, JUMPR, SLOPED, FLAMAX and LAMIN) are then printed. Finally the basic structural quantities from I-1, and the material property constants from I-2 are printed.

Special Coordinate Systems - These are the user-defined direction (special Cartesian) systems of input data item I-3. Quantities printed are the system number, and counter-clockwise angle (in degrees) from the basic X axis to the special-system x axis.

Node Definitions - The information given in input item I-4 is printed. Values are the node number, location coordinate system number (0 = basic Cartesian, 1 = basic cylindrical), X or R coordinate, Y or θ (degrees) coordinate, Z coordinate, direction coordinate system number (0 = basic Cartesian, 1 = basic cylindrical, >1 = number of special user defined system).

Element Definitions - The information given in input item I-5 is printed. Values are the element number, material I.D. number, element thickness, the three element node numbers in counter-clockwise order, and the computed element area.

Force-Displacement-Constraint Prescriptions - These are the codes given in input data item I-6. Quantities printed are the dependent node and component number, and independent node and component number. (If specified displacement, no independent numbers are given).

Nodal Load Reference Vectors - For each input component of the two load vectors from input item I-7, the node number, component number, and load value are printed.

Pressure Load Reference Vector - For each input component of the pressure load vector from input item I-8, the element number and pressure intensity are printed.

Incremental Load Data - Quantities related to input data item C-6 are printed. First is printed the number of load increments to be run. Then for each increment is given the increment number, input or default value for the predictor type, and factors to be applied to the two nodal load reference vectors and the pressure load reference vector.

4.4.2 Results for Each Load Increment

Interative Error Values - An error norm computed at the end of each iteration is printed. The error norm is obtained by a ratio of unbalanced (residual) forces to total forces.

Increment Heading - The load increment and load step numbers are printed, along with the load increment and load step values at the end of the step. Following this are the nodal load reference vector factors, the element pressure vector factor, the predictor type for the increment, the maximum allowable number of Jacobian updates and the number performed during this load step, the maximum allowable number of iterations per update and the number performed since the last update, and the maximum allowable error norm and the error norm actually achieved.

Forces and Displacements - The cumulative nodal displacements and corresponding internal forces are output. The node number is printed, followed by the U, V and W (or R, θ, Z) components of force and displacement.

Strains - The cumulative element strains are output. The element number is printed, followed by the XX, YY, and XY strains in the element coordinate system.

Limit Point Output - When a limit point is traversed, the predicted value of the incremental limit load parameter is output, followed by the predicted limit forces and displacements, and strains.

Bifurcation Point Output - When an eigenvalue solution is performed to determine a critical point, the eigenvalue computed for each inverse power iteration is printed, along with the location of the maximum value in the eigenvector.

Decomposition Output - Whenever the Gauss decomposition routine is called, it prints the value (sign and base 10 logarithm) of the Jacobian stiffness determinant.

5.0 ILLUSTRATIVE PROBLEMS

Four example problems problems are presented here in order to illustrate various aspects of nonlinear equilibrium and stability theory, and to demonstrate use of the developed nonlinear subroutines and the PANES finite element program. Section 5.1 describes a snap-through truss problem with geometric nonlinearity and a maximum and a minimum limit point. Section 5.2 describes a simple pressure membrane with nonconservative type loading (changing load area), resulting in a maximum limit point. The toroidal membrane in Section 5.3 is a fairly difficult problem, involving nonlinear (Mooney) material with follower-force loads and changing load areas. It results in very large displacements and strains, and a maximum and a minimum limit point. This problem demonstrates some unique capabilities of the PANES program. Finally, Section 5.4 describes a simple bifurcation/postbuckling model, with asymmetric behavior.

5.1 Snap-Through Truss

This is a problem similar to that used as a test case by several researchers in nonlinear structural analysis, e.g., Haisler et al. (1971). The system consists of a single inclined bar (or one half of a symmetric two-member truss) as shown in Figure 5-1. The bar has length 1.0 with axial stiffness $AE = 2x10^7$ (Hookean material), is inclined initially at a slope of 1:100, and is subjected to a vertical end load P. The PANES program idealization of this system used two constant strain triangle elements, with modulas $E = 2x10^8$ and thickness = 0.1. Node 4 was constrained to have vertical displacement equal to that at node 2, so that a system with essentially one degree of freedom (vertical displacement Q) is obtained.

The expression for the axial strain, ε, is given by

$$\varepsilon = -.01Q + 0.5Q^2 \qquad (5.1\text{-}1a)$$

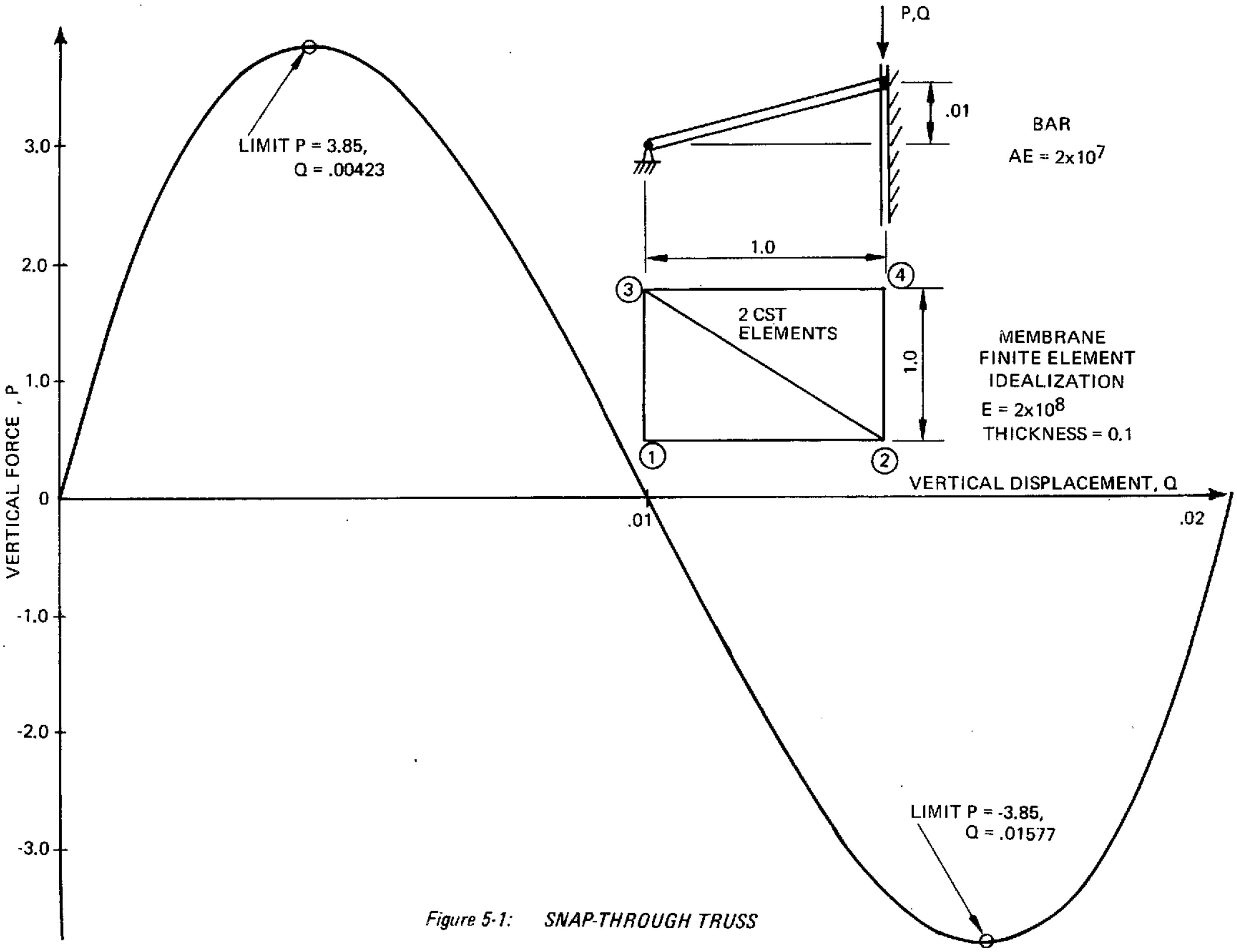


Figure 5-1: SNAP-THROUGH TRUSS

and stress, σ, is given by

$$\sigma = -2{,}000{,}000Q + 100{,}000{,}000Q^2 \qquad (5.1\text{-}1b)$$

Thus the axial force is $-200{,}000Q + 10{,}000{,}000Q^2$, from which it may by shown that the vertical applied force, P, is given by the following basic equilibrium equation.

$$P = 2000Q - 300{,}000Q^2 + 10{,}000{,}000Q^3 \qquad (5.1\text{-}2)$$

Differentiating with respect to Q, and evaluating at a reference equilibrium configuration (Q*, P*), gives the first order equilibrium equation

$$\dot{P}* = 2000\dot{Q}* - 600{,}000Q*\dot{Q}* + 30{,}000{,}000Q*^2\dot{Q}* \qquad (5.1\text{-}3a)$$

or

$$\dot{P}* = K0*\dot{Q}* \qquad (5.1\text{-}3b)$$

where the Jacobian stiffness is

$$K0* = 2000 - 600{,}000Q* + 30{,}000{,}000Q*^2$$

A second differentiation and evaluation of equation (5.1-2) results in the second order equilibrium equation

$$\ddot{P}* = K0*\ddot{Q}* + P1* \qquad (5.1\text{-}4)$$

where the psuedo force P1 is defined by

$$P1* = 600{,}000\dot{Q}*^2 + 60{,}000{,}000Q*\dot{Q}*^2$$

Using the equilibrium equations (5.1-3) and (5.1-4), the P-Q path history can be computed by various incremental and iterative approaches, including identification of limit points.

(Of course for this simple one-degree-of-freedom system, the path can be obtained immediately from the basic equilibrium equation (5.1-2)). The P-Q path history is shown in Figure 5-1. The PANES program solution was accomplished using six user-specified load increments (P = 1.0, 2.0, 3.0, 4.0, 5.0, 10.0), although about 30 additional load steps were selected automatically (mostly to achieve the desired accuracy in locating and traversing the limit point regions). Most load steps required only one or two residual force iterations, with use of a second order predictor.

5.2 Simple Pressure Membrane

One of the simplest problems which can be used to illustrate some of the effects of nonconservative loading in stability analysis is the simple pressure membrane shown in Figure 5-2. The system consists of a flat membrane 2.0 wide by 1.0 high with unit thickness, and subjected to a uniform pressure intensity Λ on one side. The ends of the membrane slide along the 45° supports, and are constrained to move together equally in the X direction. This gives a single degree-of-freedom system, with X-direction force P and displacement Q, and the membrane undergoes a uniform stretching in the Y direction. The solution was verified by a finite element analysis using the PANES program. The finite element mesh consisted of the two constant strain triangle elements shown in Figure 5-2, with X-direction displacements at nodes 2-4 constrained to equal the displacement at node 1. Zero displacements were enforced in the Z direction.

Considering large displacement and large strain effects, the stretch in the Y direction is denoted by λ, and is equal to the change in length divided by original length. The strain-energy density U, measured per unit undeformed volume, is taken to be defined by the function

$$U = C_1(\lambda-1)^2 + C_2(\lambda-1)^4 = C_1Q^2 + C_2Q^4 \qquad (5.2\text{-}1)$$

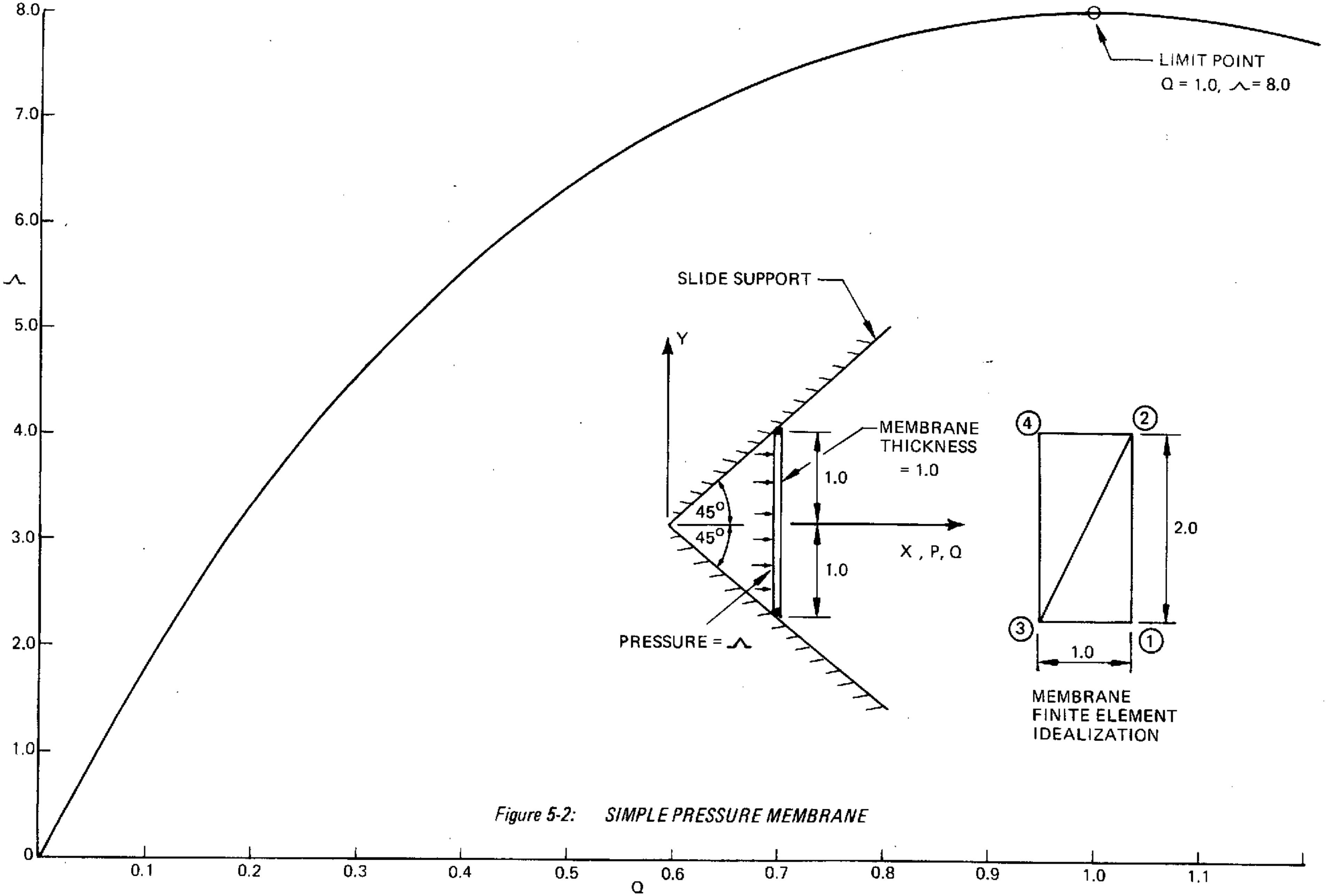


Figure 5-2: SIMPLE PRESSURE MEMBRANE

Note that we could define a Lagrangian strain ε, and stress-like quantity σ, by

$$\varepsilon = \frac{1}{2}(\lambda^2-1)$$

$$\sigma = \frac{\partial U}{\partial \lambda} = \frac{\partial U}{\partial Q} = 2C_1Q + 4C_2Q^3$$

However, this is not necessary in the present problem, for which the force can be derived directly from the strain-energy function.

The basic equilibrium equation for the system is written using the equivalence of external force (defined in terms of the pressure loading Λ) and internal force (defined as the derivative of strain energy with respect to displacement Q):

$$P = 2\Lambda(1+Q) = 2(2C_1Q + 4C_2Q^3) \qquad (5.2\text{-}2)$$

Choosing specific values for the material constants $C_1 = 10$ and $C_2 = -1$, the basic equilibrium equation may be written as

$$\Lambda(1+Q) = 20Q - 4Q^3 \qquad (5.2\text{-}3)$$

Differentiating equation (5.2-3) provides the first order equilibrium equation

$$\dot{\Lambda}(1+Q) + \Lambda\dot{Q} = 20\dot{Q} - 12Q^2\dot{Q} \qquad (5.2\text{-}4a)$$

or at an equilibrium configuration (Λ*, Q*) we can write

$$\dot{\Lambda}^* = ((20 - 12Q^{*2} - \Lambda^*)/(1+Q^*))\,\dot{Q}^* \qquad (5.2\text{-}4b)$$

From this equation it can be seen that the value of a Jacobian stiffness K0*, relating $\dot{\Lambda}^*$ with $\dot{Q}^*$, is

$$K0^* = (20 - 12Q^{*2} - \Lambda^*)/(1+Q^*) \qquad (5.2\text{-}4c)$$

(For simplicity we have here defined the Jacobian relative to $\dot{\Lambda}$ rather than $\dot{P}$).

Differentiating equation (5.2-3) a second time gives the second order equilibrium equation, as

$$\ddot{\Lambda}\,(1+Q) + 2\dot{\Lambda}\,\dot{Q} + \Lambda\,\ddot{Q} = 20\ddot{Q} - 24Q\dot{Q}^2 - 12Q^2\ddot{Q} \qquad (5.2\text{-}5a)$$

or at an equilibrium configuration we can write

$$\ddot{\Lambda}^* = K0^*\ddot{Q}^* + P1^* \qquad (5.2\text{-}5b)$$

where the psuedo force P1 is given by

$$P1^* = (-24Q^*\dot{Q}^{*2} - 2\dot{\Lambda}^*\dot{Q}^*)/(1+Q^*) \qquad (5.2\text{-}5c)$$

To illustrate the actual behavior of the system, we now choose Q as the path parameter, and without loss of generality specify at every point the conditions $\dot{Q} = 1$ and $\ddot{Q} = 0$. To aid in determination of a limit type critical point, we have the condition $\dot{\Lambda}^* = 0$ at the limit point. Using this condition, and solving equation (5.2-4b) with the use of (5.2-3), we find a limit point at ($\Lambda^* = 8.0$, $Q^* = 1.0$). The entire Λ-Q path history may be determined by various incremental and iterative predictor-corrector schemes, and is shown in Figure 5-2. The PANES program solution to this problem used a second-order predictor with residual-force corrective iterations.

5.3 Toroidal Membrane

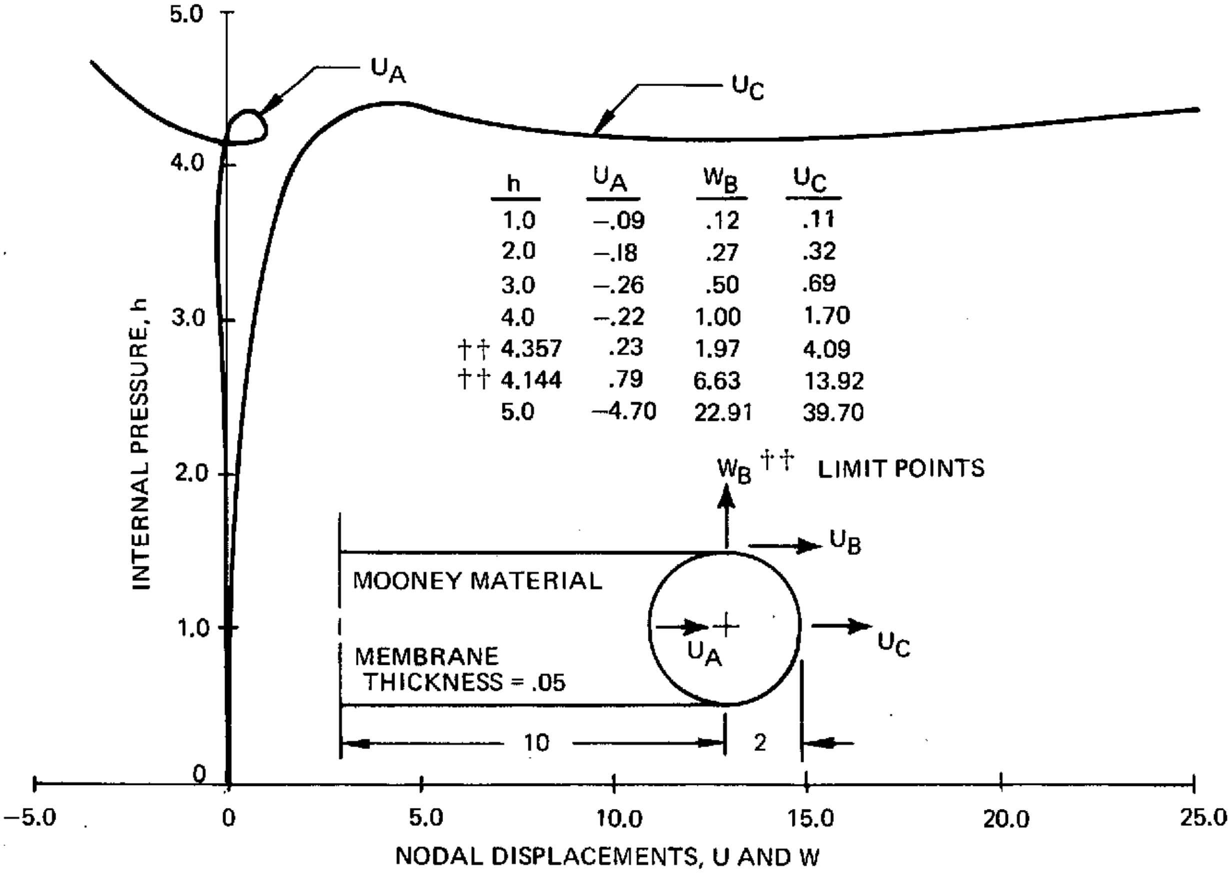
The problem illustrated here is a toroidal membrane under internal pressure, shown in Figure 5-3. This structure exhibits a highly nonlinear type of behavior with very large displacements and strains, and both a maximum and a minimum limit point. The second-order predictor was employed in a PANES program solution, along with residual-force corrective iterations to achieve equilibrium at each load step. Both the predictor and corrector incorporated all changing load area and follower-force effects. (This gave an unsymmetric K0* matrix, whose decomposition was obtained by the Gauss wavefront procedure. It was apparent that the unsymmetric effects became large enough that their inclusion was necessary for convergence).

The torus was assumed to be of Mooney material with constants $C_1 = 80$ and $C_2 = 20$, and was analyzed using plane-stress membrane elements. Geometry and displacement components are defined in Figure 5- 3. The torus has major radius 10, minor radius 2, and thickness .05. Cylindrical coordinates were employed to model a wedge-shaped segment of the major circumference, of from 2 to 10 degrees arc. Constraints were employed in the radial and vertical directions in order to equalize corresponding displacements along the two sides of the wedge.

Table 5-1 summarizes computed values of key displacements for the user-defined (input) pressure increments and the computed limit pressures, obtained with three different meshes. (N denotes the number of subdivisions over one half of the minor circumference). The indicated convergence with mesh refinement is of the kind to be expected, with finer meshes giving a more flexible structure, and resulting generally in somewhat larger displacements and lower limit point values. Computer run times ranged from 1 minute (IBM central processor time) for mesh N = 4, to 7 minutes for mesh N = 12. These times should be regarded in a qualitative fashion only, since for example much of the time was spent in solving the linear unsymmetric stiffness equations and this time could be reduced by use of a production type equation solver.

More detailed results for the fine mesh (N = 12) are shown in Table 5-2. There the data columns represent respectively the pressure, load increment number (user-specified increment), step number (where the PANES program automatically divided the specified increment into a number of smaller steps), number of residual-force type iterations performed in order to achieve the required accuracy, and values of the radial and vertical displacements at key points. Figure 5-3 gives very interesting plots of two key load-displacement paths, and indicates that no difficulties were caused by a displacement which followed an extremely irregular "doubling back" type of path, including sharp curvature sections. The basic results for this problem are corroborated by another solution to the same problem by Key (1974), who developed a finite element program with a Newton Raphson solution technique, and obtained results for pressure levels up to near the first limit point.

It may be of use for comparison/test purposes to mention corresponding results obtained by increasing the major radius from 10 to 12. Maximum and minimum limit points occurred at pressures, h, of 4.355 and 4.125, respectively, while maximum displacements (at h = 5.0) increased roughly 20 percent.



h	U_A	W_B	U_C
1.0	−.09	.12	.11
2.0	−.18	.27	.32
3.0	−.26	.50	.69
4.0	−.22	1.00	1.70
†† 4.357	.23	1.97	4.09
†† 4.144	.79	6.63	13.92
5.0	−4.70	22.91	39.70

†† LIMIT POINTS

Figure 5-3: *TORUS SOLUTION RESULTS*

TABLE 5-1: TORUS RESULTS (MESH/CONVERGENCE CHARACTERISTICS)

h	U_A	U_B	W_B	U_C	
.10	-.0065	.0015	.0120	.0076	N = 4
.50	-.0399	.0114	.0529	.0484	
1.0	-.0827	.0315	.1080	.1130	
2.0	-.162	.109	.243	.301	
3.0	-.216	.281	.436	.631	
4.0	-.168	.732	.784	1.371	
4.5	.057	1.438	1.203	2.436	
†† 4.688	.653	2.891	1.959	4.543	
†† 4.458	2.032	10.056	6.787	15.551	
5.0	-3.142	22.203	20.007	35.578	
.10	-.0062	.0013	.0135	.0067	N = 6
.50	-.0414	.0112	.0571	.0489	
1.0	-.0870	.0323	.1157	.1169	
2.0	-.171	.117	.261	.320	
3.0	-.223	.316	.476	.693	
4.0	-.129	.908	.904	1.640	
†† 4.473	.641	2.892	1.967	4.531	
†† 4.244	1.946	10.374	7.039	15.951	
4.5	-1.665	19.076	15.972	29.895	
5.0	-4.746	27.973	24.446	42.678	
.10	-.0065	.0013	.0136	.0070	N = 12
.50	-.0431	.0115	.0584	.0505	
1.0	-.0902	.0333	.1189	.1210	
2.0	-.176	.123	.270	.334	
3.0	-.226	.340	.499	.736	
4.0	-.088	1.050	.991	1.853	
†† 4.357	.636	2.889	1.964	4.518	
†† 4.127	1.903	10.607	7.199	16.239	
4.5	-2.895	22.706	18.904	34.581	
5.0	-5.510	32.327	27.005	47.404	

†† Limit points

TABLE 5-2: TORUS INCREMENTAL RESULTS (N = 12)

	h	INCR.	STEP	ITER.	U_A	U_B	W_B	U_C
	.00625	1	1	7	-.00026	.00004	.00121	.00028
†	.10		2	6	-.0065	.0013	.0136	.0070
†	.50	2	1	3	-.0431	.0115	.0584	.0505
†	1.0	3	1	2	-.0902	.0333	.1189	.1210
	1.5	4	1	2	-.135	.069	.188	.213
†	2.0		2	2	-.176	.123	.270	.334
	2.5	5	1	2	-.209	.207	.370	.499
	2.75	5	2	1	-.220	.266	.430	.606
†	3.0		3	1	-.226	.340	.499	.736
	3.360	6	1	2	-.219	.490	.621	.986
	3.632		2	2	-.193	.658	.742	1.254
	3.816		3	1	-.155	.819	.849	1.503
	3.908		4	1	-.127	.922	.914	1.661
†	4.0		5	1	-.088	1.050	.991	1.853
	4.108	7	1	1	-.023	1.248	1.106	2.147
	4.187		2	1	.049	1.446	1.216	2.439
	4.243		3	1	.123	1.641	1.322	2.723
	4.284		4	1	.197	1.829	1.421	2.995
	4.309		5	1	.268	2.005	1.513	3.250
	4.327		6	1	.335	2.167	1.596	3.483
	4.339		7	1	.395	2.313	1.671	3.692
	4.346		8	1	.448	2.441	1.736	3.875
	4.351		9	1	.494	2.549	1.791	4.031
	4.354		10	0	.532	2.640	1.837	4.161
††	4.357		-	-	.636	2.889	1.964	4.518
	4.354		11	6	.795	3.273	2.160	5.067
	4.329		12	2	1.094	4.018	2.546	6.137
	4.279		13	2	1.472	5.063	3.117	7.654
	4.229		14	1	1.784	6.117	3.740	9.210
	4.204		15	1	1.921	6.715	4.121	10.108
	4.179		16	1	2.040	7.413	4.595	11.172
	4.158		17	1	2.110	8.143	5.125	12.302
	4.145		18	1	2.124	8.725	5.577	13.218
	4.137		19	1	2.107	9.194	5.958	13.962
	4.133		20	1	2.076	9.568	6.272	14.559
	4.130		21	0	2.040	9.860	6.526	15.029
††	4.127		-	-	1.903	10.607	7.199	16.239
	4.130		22	5	1.659	11.544	8.083	17.760
	4.153		23	2	1.036	13.257	9.785	20.534
	4.327		24	3	-1.387	18.766	15.270	28.987
†	4.5		25	2	-2.895	22.706	18.904	34.581
†	5.0	8	1	5	-5.510	32.327	27.005	47.404

† Input pressure loads. Intermediate pressures were selected automatically by the program.

†† Limit points.

5.4 Asymmetric Buckling Model

The problem illustrated here is that used by several investigators of bifurcation and postbuckling behavior, e.g., Thompson (1970). The model consists of a spring and rigid bar as shown in Figure 5-4. The asymmetric postbuckling behavior is due to the decreasing resisting moment arm of the spring force about point O, as the top of the bar deflects to the right.

The conservative load Λ is applied vertically to the top of the bar. The spring is initially inclined at 45 degrees, and has constant stiffness K. This is a single degree of freedom system, defined by the horizontal displacement Q. The vertical component of distance from O to B at any time is equal to $\sqrt{1-Q^2}$, and the horizontal distance from A to B is 1+Q, from which the length of the spring is found to be $\sqrt{2(1+Q)}$. The moment arm of the spring force about point O is then determined as $\sqrt{1-(1+Q)/2}$. Equating the external applied and internal resisting moments gives the basic postbuckling equilibrium equation for the system as

$$Q = K(\sqrt{2(1+Q)} - \sqrt{2})\ \sqrt{1-(1+Q)/2} \quad = \quad K(\ \sqrt{1+Q} - 1)\ \sqrt{1-Q} \qquad (5.4\text{-}1a)$$

or

$$\Lambda = K(\sqrt{1-Q^2} - \sqrt{1-Q)}/Q \qquad (5.4\text{-}1b)$$

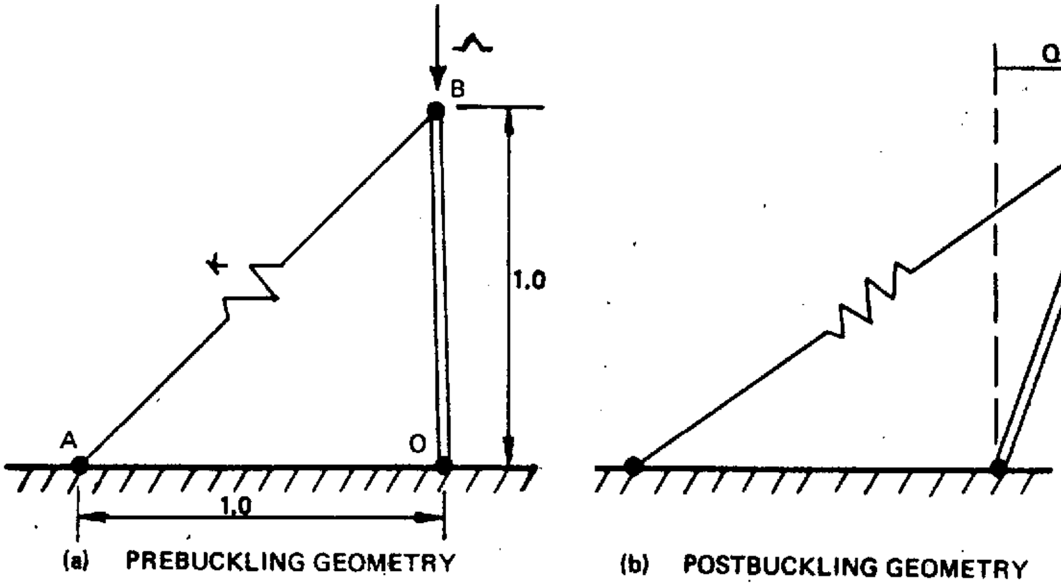
Evaluating Λ from this expression using a small finite difference in Q, gives the critical load value as

$$\Lambda = \frac{1}{2}\,K \qquad (5.4\text{-}2)$$

Similarly a second-order finite difference evaluation gives the critical asymmetric load rate as

$$\frac{\partial \Lambda}{\partial Q} = -\frac{3}{8} K \tag{5.4-3}$$

The location of the critical (bifurcation) point was verified by a PANES program solution.



(a) PREBUCKLING GEOMETRY (b) POSTBUCKLING GEOMETRY

Figure 5-4: ASYMMETRIC BUCKLING MODEL

6.0 CONCLUSIONS AND RECOMMENDATIONS

Conclusions - The present work provides improved techniques for solution of structures with material and geometric nonlinearities. FORTRAN subroutines have been developed, and incorporated into a new nonlinear finite-element program called PANES (Program for Analysis of Nonlinear Equilibrium and Stability). A new approach is developed for representing an arbitrary nonlinear material in terms of a finite-difference generated stress-strain expansion, and is considered to be of major significance. This approach leads to formulation of perturbation-type equilibrium equations of any desired order, and is effective even for numerically integrated finite elements with large degrees of freedom. The formulation should provide a unifying basis for design of many nonlinear structural analysis programs.

The present PANES program is a pilot version, capable of analyzing problems with large strain and arbitrary nonlinear elastic materials, and provides membrane finite elements (two or three degrees of freedom per node) with nonconservative pressure loading. It includes automated techniques which have been developed for selection of load step sizes, and for locating and traversing maximum and minimum limit-type critical points. Subroutines are also included for location of bifurcation-type critical points on a general nonlinear prebuckling path, and for determining the symmetric or asymmetric postbuckling behavior. The postbuckling capabilities have not yet been completely automated and tested, however, and should be regarded as being in a developmental stage.

Recommendations - The PANES program solution routines provide a significant pilot capability for analysis of structures with highly nonlinear material and geometric effects, and should now be extended and evaluated for a wider class of practical structures. Listed below are recommendations for future work.

1. The first priority should be given to improving and verifying the PANES postbuckling subroutines. This effort should include the generalization of program logic to accept additional types of finite elements, and in particular the addition of simple bar elements to the program. These elements will simplify the study of postbuckling results, and are desireable for initial verification because of the somewhat complex nature of the new nonlinear material postbuckling theory. Automated techniques should be incorporated for branching to the postbuckling path, similar to the existing PANES techniques for traversing limit points.
2. The program should be extended to incorporate a number of higher-order finite elements. Important candidate elements are plates, shells, and isoparametric-solid elements. Such elements will greatly extend the analysis capabilities of the program, and also importantly demonstrate the effectiveness of the new nonlinear solution techniques for elements which are numerically integrated and have large degrees of freedom.
3. PANES now handles an arbitrary nonlinear elastic material, by use of the proper material strain-energy definition. Formation of the stress-strain expansion relation should be generalized to cases of inelastic material, i.e., those materials for which a strain-energy function does not exist. The concept of this generalization is not difficult, but some study is required to develop an effective algorithm for forming the higher-order stress-strain expansion terms.
4. A number of largely theoretical improvements should be studied. These include the treatment of multiple and closely-spaced critical points (as often occur in an optimally designed light-weight structure), and the method

of postbuckling behavior solution for cases of an unsymmetric Jacobian stiffness matrix. Incorporation of a third-order fundamental path predictor also appears desireable, especially for use in prediction of bifurcation-type critical points. The nonlinear eigenvalue solution for these points is somewhat costly, and would have to be performed less often with the higher-order predictor.

5. The program size capability should be increased to handle the expected range of practical nonlinear structural problems. This involves some reorganization of the main program logic, and the addition of a production-type linear equation solver such as the Gauss-wavefront routines used in the BOPACE elastic-plastic-creep program (Vos and Armstrong (1973)).

7.0 REFERENCES AND BIBLIOGRAPHY

1. Argyris, J. H., Fried, I., and Scharpf, D. W., (1968), "The TET 20 and TEA 8 Elements for the Matrix Displacement Method," Journal of the Royal Aeronautical Society, Vol. 72, pp. 618-623.

2. Connor, J., and Morin, N., (1970), "Perturbation Techniques in the Analysis of Geometrically Nonlinear Shells," High Speed Computing of Elastic Structures, Proceedings of Symposium, Liege, Belgium, Vol. 2, pp. 683-705.

3. Britvec, S. J., and Chilver, A. H., (1963), "Elastic Buckling of Rigidly-Jointed Braced Frames," Journal of the Engineering Mechanics Division, ASCE, Vol. 89, pp. 217-255.

4. Ecer, A., (1973), "Finite Element Analysis of the Postbuckling Behavior of Structures," AIAA Journal, Vol. 11, pp. 1532-1538.

5. Gallagher, R. H., and Mau, S. T., (1972), "A Method of Limit Point Calculation in Finite Element Structural Analysis," NASA CR-2115, National Aeronautics and Space Administration, Washington, D.C.

6. Gallagher, R. H., and Padlog, J., (1963), "Discrete Element Approach to Structural Instability Analysis," AIAA Journal, Vol. 1, pp. 1437-1439.

7. Haftka, R. T., Mallett, R. H., and Nachbar, W., (1971), "Adaption of Koiter's Method to Finite Element Analysis of Snap-Through Buckling Behavior," International Journal of Solids and Structures, Vol. 7, pp. 1427-1445.

8. Haftka, R. T., Mallett, R. H., and Nachbar, W., (1970), "A Koiter-Type Method for Finite Element Analysis of Nonlinear Structural Behavior, Volume I: The Modified Structure Method," AFFDL-TR-70-130, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio.

9. Haisler, W. E., Stricklin, J. A., and Stebbins, F. J., (1971), "Development and Evaluation of Solution Procedures for Geometrically Nonlinear Structural Analysis by the Direct Stiffness Method," AIAA Paper No. 71-356, AIAA/ASME 12th Structures, Structural Dynamics and Materials Conference, Anaheim, California.

10. Huseyin, K., (1973), "The Multiple-Parameter Perturbation Technique for the Analysis of Non-Linear Systems," International Journal of Non-Linear Mechanics, Vol. 8, pp. 431-443.

11. Hutchinson, J. W., (1973), "Post-Bifurcation Behavior in the Plastic Range," Journal of the Mechanics and Physics of Solids, Vol. 21, pp. 163-190.

12. Hutchinson, J. W., (1973), "Imperfection Sensitivity in the Plastic Range," Journal of the Mechanics and Physics of Solids, Vol. 21, pp. 191-204.

13. Johns, K. C. and Chilver, A. H., (1971), "Multiple Path Generation at Coincident Branching Points, "International Journal of Mechanical Sciences, Vol. 13, pp. 899-910.

14. Key, J. E., (1974), "A Note on the Numerical Solution of Certain Problems in Finite Elasticity by the Finite Element Method," paper to be published in International Journal for Numerical Methods in Engineering.

15. Koiter, W. T., (1945), "Over de Stabiliteit van het Elastisch Evenwicht," Ph.D. Thesis, Technische Hogeschool, Delf, Holland. English translation (1970), "The Stability of Elastic Equilibrium," AFFDL-TR-70-25, Air Force Flight Dynamics Laboratory, Wright-Patterson Air Force Base, Ohio.

16. Levinson, M., and Burgess, I. W., (1971), "A Comparison of Some Simple Constitutive Relations for Slightly Compressible Rubber-Like Materials," International Journal of Mechanical Sciences, Vol. 13, pp. 563-572.

17. Mallett, R. H., and Marcal, P. V., (1968), "Finite Element Analysis of Nonlinear Structures," Journal of the Structural Division, ASCE, Vol. 94, pp. 2081-2105.

18. Marcal, P.V., (1969), "Large Deflection Analysis of Elastic-Plastic Plates and Shells," Brown University Technical Report No. N00014-67-A-0191-0006/1.

19. Marchenko, G. A., (1966), "Solution of Nonconservative Problems in the Theory of Elastic Stability by Ritz's Method," Isvestiya VUZ. Aviatsionnaya Tekhnika, No. 3, pp. 62-68.

20. Mau, S. T., and Gallagher, R. H., (1972), "A Finite Element Procedure for Nonlinear Prebuckling and Initial Postbuckling Analysis," NASA CR-1936, National Aeronautics and Space Administration, Washington, D.C.

21. McNamara, J. F., and Marcal, P. V., (1971), "Incremental Stiffness Method for Finite Element Analysis of the Nonlinear Dynamics Problem," Proceedings, O.N.R. International Symposium on Numerical and Computer Methods in Structural Mechanics, University of Illinois.

22. Morin, N., (1970), "Nonlinear Analysis of Thin Shells," Massachusetts Institute of Technology, Department of Civil Engineering, Report R70-43.

23. Murray, D. W., and Wilson, E. L., (1969), "Finite Element Large Deflection Analysis of Plates," Journal of the Engineering Mechanics Division, ASCE, Vol. 95, pp. 143-165.

24. Nayak, G. C., and Zienkiewicz, O. C., (1972), "Note on the 'Alpha' - Constant Stiffness Method for the Analysis of Non-Linear Problems," International Journal for Numerical Methods in Engineering, Vol. 4, pp. 579-582.

25. Nemat-Nasser, S., and Shatoff, H. D., (1973), "Numerical Analysis of Pre- and Post-Critical Response of Elastic Continua at Finite Strains," Computers and Structures, Vol. 3, pp. 983-999.

26. Oden, J. T., and Key, J. E. (1970), "Numerical Analysis of Finite Axisymmetric Deformations of Incompressible Elastic Solids of Revolution," International Journal of Solids and Structures, Vol. 6, pp. 497-518.

27. Osias, J. R., and Swedlow, J. L., (1974), "Finite Elasto-Plastic Deformation-I: Theory and Numerical Examples," International Journal of Solids and Structures, Vol. 10, pp. 321-339.

28. Prasad, S. N., and Herrmann, G., (1972), "Adjoint Variational Methods in Nonconservative Stability Problems," International Journal of Solids and Structures," Vol. 8, pp. 29-40.

29. Remmler, K. L., Cawood, D. W., Stanton, J. A., and Hill, R., (1966), "Solutions of Systems of Nonlinear Equations," Final Report to NASA Marshall Space Flight Center by Lockheed Missiles and Space Company, HREC 0178-1, A-78333.

30. Sandidge, D. W., (1973), "Stability and Postbuckling Behavior of Hyperelastic Bodies at Finite Strain by the Finite Element Method," M. S. Thesis, University of Alabama in Huntsville, Huntsville, Alabama.

31. Schmit, L. A., Bogner, F. K., and Fox, R. L., (1968), "Finite Deflection Structural Analysis Using Plate and Shell Discrete Elements," AIAA Journal, Vol. 6, pp. 781-791.

32. Sewell, M. J., (1965), "The Static Perturbation Technique in Buckling Problems," Journal of the Mechanics and Physics of Solids, Vol. 13, pp. 247-263.

33. Straight, J. W., (1968), "Solution to Beam Vibrations Problems with Mixed Response-Excitation Input Information," AIAA/ ASME 9th Structures, Structural Dynamics and Materials Conference, Palm Springs, California.

34. Stricklin, J. A., Haisler, W. E., and Von Riesemann, W. A., (1972), "Computation and Solution Procedures for Nonlinear Analysis by Combined Finite Element-Finite Difference Methods," Computers and Structures, Vol. 2, pp. 955-974.

35. Stricklin, J. A., Haisler, W. E., and Von Riesemann, W. A., (1973), "Evaluation of Solution Procedures for Material and/or Geometrically Nonlinear Structural Analysis," AIAA Journal, Vol. 11, pp. 292-299.

36. Thompson, J. M. T., (1963), "Basic Principles in the General Theory of Elastic Stability," Journal of the Mechanics and Physics of Solids, Vol. 11, pp. 13-20.

37. Thompson, J. M. T., (1969), "A General Theory for the Equilibrium and Stability of Discrete Conservative Systems," Zeitschrift für angewandte Mathematik und Physik," Vol. 20, pp. 797-846.

38. Thompson, J. M. T., (1970), "A New Approach to Elastic Branching Analysis," Journal of the Mechanics and Physics of Solids, Vol. 18, pp. 29-42.

39. Thompson, J. M. T., and Hunt, G. W., (1971), "A Theory for the Numerical Analysis of Compound Branching," Zeitschrift für angewandte Mathematik und Physik," Vol. 22, pp. 1001-1015.

40. Tillerson, J. R., Stricklin, J. A., and Haisler, W. E., (1973), "Numerical Methods for the Solution of Nonlinear Problems in Structural Analysis," ASME Symposium on Numerical Solution of Nonlinear Structural Problems, Detroit, Michigan.

41. Turner, M. J., Dill, E. H., Martin, H. C., and Melosh, R. J., (1960), "Large Deflections of Structures Subjected to Heating and External Loads," Journal of the Aero/Space Sciences, Vol. 27, pp. 97-106.

42. Vos, R. G., (1970), "Finite element analysis of plate buckling and postbuckling," Ph.D. Thesis, Rice University, Houston, Texas.

43. Vos, R. G., (1974), "Finite Element Solution of Nonlinear Structures by Perturbation Technique," First International Conference on Numerical Methods in Nonlinear Mechanics, University of Texas at Austin, Austin, Texas.

44. Vos, R. G., and Armstrong, W. H., (1973), "BOPACE Theoretical Manual," Report on Space Shuttle Main Engine Computer Program Development, to NASA Marshall Space Flight Center by Boeing Aerospace Company, D5-17266-1.

45. Vos, R. G., and Vann, W. P., (1973), "A Finite Element Tensor Approach to Plate Buckling and Postbuckling," International Journal for Numerical Methods in Engineering, Vol. 5, pp. 351-365.

46. Walker, A. C., (1969), "A Method of Solution for Nonlinear Simultaneous Algebraic Equations," International Journal for Numerical Methods in Engineering, Vol. 1, pp. 177-180.

47. Zienkiewicz, O. C., and Nayak, G. C., (1971), "A General Approach to Problems of Large Deformation, and Plasticity Using Iso-Parametric Elements," 3rd Conference on Matrix Methods in Structural Mechanics, Wright-Patterson Air Force Base, Ohio.

48. Zienkiewicz, O. C., Valliappan, S., and King, I. P., (1969), "Elasto-Plastic Solutions of Engineering Problems 'Initial Stress,' Finite Element Approach," International Journal for Numerical Methods in Engineering, Vol. 1, pp. 75-100.

APPENDIX A: FINITE DIFFERENCE EXPANSIONS

For the type of nonlinear solution techniques utilized in this work, it is necessary to generate the Jacobian stiffness matrix as well as various force-type vectors associated with residuals and nonlinear predictor quantities. In the general case involving nonlinear materials, these quantities cannot be effectively determined by an explicit process, and must be generated numerically. The numerical representation may be based on a direct expansion of the generalized forces in terms of displacements (e.g. the method for Jacobian generation used by Oden and Key (1970)), or, as in the approach used here, it may be based on an expansion of stresses in terms of strains. In any case the procedure requires the expansion of a dependent function of several independent variables, about a known reference point.

An effective expansion procedure has been developed in the present work by use of a Taylor series, in which the expansion coefficients (partial derivatives) are evaluated using finite difference expressions. After a study of various alternatives, it was concluded that the most efficient scheme involves forward differences rather than central differences, because the forward differences result in simple formulas and require a minimum number of function evaluations. In addition, if an approximate solution path increment is known, i.e. if the approximate increments which will occur in the independent variables are known, then a more accurate function representation can be obtained with the forward difference scheme by selecting the appropriate difference values. Difference coefficients are derived here for expansions of linear, quadratic and cubic form.

Linear Form - Coefficients for a linear expansion correspond to those in a two-point forward difference formula. The derivation is rather trivial, but it serves to illustrate the basic procedure.

The Taylor series expansion of an arbitrary function f, in terms of independent variables x_i, is

$$f = f^1 + f_i \, \Delta x_i \qquad \text{(A-1)}$$

where $f_i = \partial f / \partial x_i$ denotes the partial derivative of f with respect to the ith independent variable, and Δ denotes an incremental quantity. The unique types of terms may be derived by considering only one of the independent variables, which we denote simply by x. Referring to Figure A-1, we describe the values of f at points 1 and 2 by the linear expansion

$$\begin{Bmatrix} f^1 \\ f^2 \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{Bmatrix} f^1 \\ f_x \Delta x \end{Bmatrix} \qquad \text{(A-2)}$$

Inversion of this relation gives explicit definition to the difference coefficients, in the form of the matrix in the inverse relation:

$$\begin{Bmatrix} f^1 \\ f_x \Delta x \end{Bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{Bmatrix} f^1 \\ f^2 \end{Bmatrix} \qquad \text{(A-3)}$$

or

$$f_x = (-f^1 + f^2)/\Delta x \qquad \text{(A-4)}$$

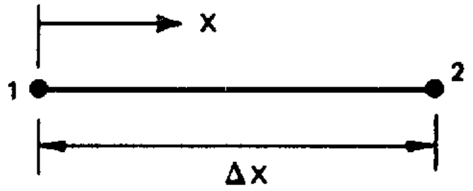


Figure A-1: Linear Difference Expansion

Quadratic Form - Coefficients for the quadratic expansion correspond to those in a three-point forward difference formula. Because the expansion involves terms no higher than second order, the unique types of difference coefficients can be derived by considering only two of the independent variables, say x and y. The corresponding Taylor series expansion for the function f(x,y) is given by the expression

$$f = f^1 + f_x \Delta x + f_y \Delta y + \frac{1}{2} f_{xx} (\Delta x)^2 + f_{yx} \Delta y \Delta x + \frac{1}{2} f_{yy} (\Delta y)^2 \qquad \text{(A-5)}$$

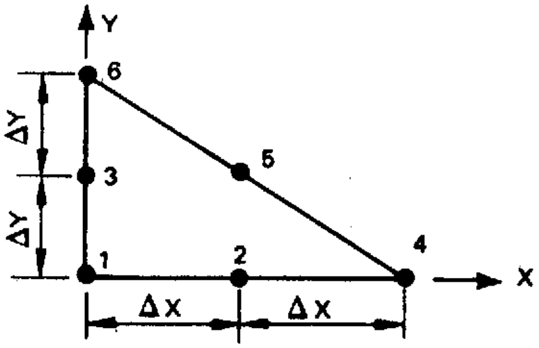


Figure A-2: Quadratic Difference Expansion

Referring to Figure A-2, we write

$$\begin{Bmatrix} f^1 \\ f^2 \\ f^3 \\ f^4 \\ f^5 \\ f^6 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1/2 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1/2 \\ 1 & 2 & 0 & 2 & 0 & 0 \\ 1 & 1 & 1 & 1/2 & 1 & 1/2 \\ 1 & 0 & 2 & 0 & 0 & 2 \end{bmatrix} \begin{Bmatrix} f^1 \\ f_x \Delta x \\ f_y \Delta y \\ f_{xx} (\Delta x)^2 \\ f_{yx} \Delta y \Delta x \\ f_{yy} (\Delta y)^2 \end{Bmatrix} \qquad \text{(A-6)}$$

and inverting the above relation gives

$$\begin{Bmatrix} f^1 \\ f_x\Delta x \\ f_y\Delta y \\ f_{xx}(\Delta x)^2 \\ f_{yx}\Delta y\Delta x \\ f_{yy}(\Delta y)^2 \end{Bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -3/2 & 2 & 0 & -1/2 & 0 & 0 \\ -3/2 & 0 & 2 & 0 & 0 & -1/2 \\ 1 & -2 & 0 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 & 0 \\ 1 & 0 & -2 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} f^1 \\ f^2 \\ f^3 \\ f^4 \\ f^5 \\ f^6 \end{Bmatrix} \qquad \text{(A-7)}$$

Cubic Form - Coefficients in the cubic expansion correspond to those in a four-point forward difference formula. Because the expansion terms are no higher than third order, the unique coefficient types can be derived by writing the function in terms of only three independent variables, say x, y and z:

$$f = f^1 + f_x\Delta x + f_y\Delta y + f_z\Delta z + \tfrac{1}{2}f_{xx}(\Delta x)^2 + \ldots + \tfrac{1}{6}f_{xxx}(\Delta x)^3$$
$$+ \ldots + \tfrac{1}{6}f_{zzz}(\Delta z)^3 \qquad \text{(A-8)}$$

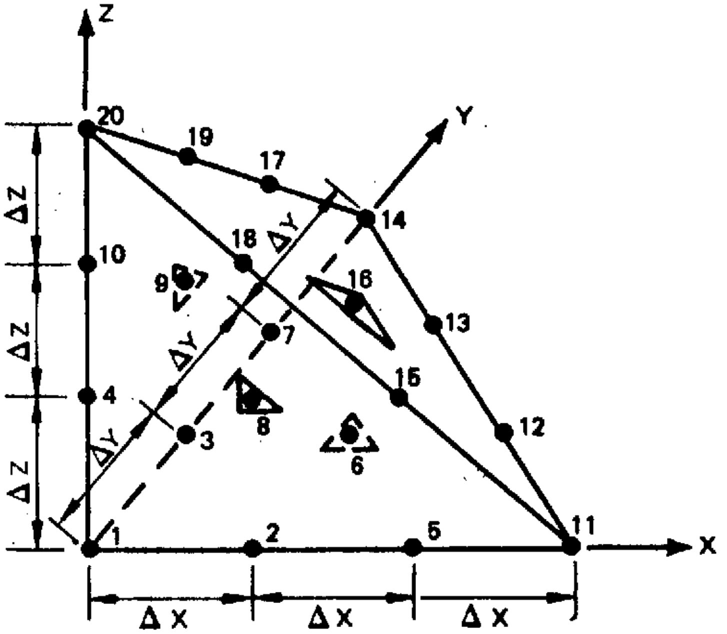


Figure A-3: Cubic Difference Expansion

Referring to Figure A-3, we write

$$\left\{\begin{array}{c} f^{1} \\ f^{2} \\ f^{3} \\ f^{4} \\ f^{5} \\ f^{6} \\ f^{7} \\ f^{8} \\ f^{9} \\ f^{10} \\ f^{11} \\ f^{12} \\ f^{13} \\ f^{14} \\ f^{15} \\ f^{16} \\ f^{17} \\ f^{18} \\ f^{19} \\ f^{20} \end{array}\right\} = \left[\begin{array}{cccccccccccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{6} \\
1 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & \frac{4}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 1 & 0 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{6} & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{4}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & \frac{1}{2} & 0 & 0 & 1 & 0 & \frac{1}{2} & \frac{1}{6} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{6} \\
1 & 0 & 1 & 1 & 0 & 0 & \frac{1}{2} & 0 & 1 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{6} \\
1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{4}{3} \\
1 & 3 & 0 & 0 & \frac{9}{2} & 0 & 0 & 0 & 0 & 0 & \frac{9}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 1 & 0 & 2 & 2 & \frac{1}{2} & 0 & 0 & 0 & \frac{4}{3} & 2 & 1 & \frac{1}{6} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 2 & 0 & \frac{1}{2} & 2 & 2 & 0 & 0 & 0 & \frac{1}{6} & 1 & 2 & \frac{4}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 3 & 0 & 0 & 0 & \frac{9}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{9}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 2 & 0 & 1 & 2 & 0 & 0 & 2 & 0 & \frac{1}{2} & \frac{4}{3} & 0 & 0 & 0 & 2 & 0 & 0 & 1 & 0 & \frac{1}{6} \\
1 & 1 & 1 & 1 & \frac{1}{2} & 1 & \frac{1}{2} & 1 & 1 & \frac{1}{2} & \frac{1}{6} & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} & \frac{1}{2} & 1 & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} \\
1 & 0 & 2 & 1 & 0 & 0 & 2 & 0 & 2 & \frac{1}{2} & 0 & 0 & 0 & \frac{4}{3} & 0 & 0 & 2 & 0 & 1 & \frac{1}{6} \\
1 & 1 & 0 & 2 & \frac{1}{2} & 0 & 0 & 2 & 0 & 2 & \frac{1}{6} & 0 & 0 & 0 & 1 & 0 & 0 & 2 & 0 & \frac{4}{3} \\
1 & 0 & 1 & 2 & 0 & 0 & \frac{1}{2} & 0 & 2 & 2 & 0 & 0 & 0 & \frac{1}{6} & 0 & 0 & 1 & 0 & 2 & \frac{4}{3} \\
1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & \frac{9}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{9}{2}
\end{array}\right] \left\{\begin{array}{l} f^{1} \\ f_{x}\Delta x \\ f_{y}\Delta y \\ f_{z}\Delta z \\ f_{xx}(\Delta x)^{2} \\ f_{yx}\Delta y\Delta x \\ f_{yy}(\Delta y)^{2} \\ f_{zx}\Delta z\Delta x \\ f_{zy}\Delta z\Delta y \\ f_{zz}(\Delta z)^{2} \\ f_{xxx}(\Delta x)^{3} \\ f_{yxx}\Delta y(\Delta x)^{2} \\ f_{yyx}(\Delta y)^{2}\Delta x \\ f_{yyy}(\Delta y)^{3} \\ f_{zxx}\Delta z(\Delta x)^{2} \\ f_{zyx}\Delta z\Delta y\Delta x \\ f_{zyy}\Delta z(\Delta y)^{2} \\ f_{zzx}(\Delta z)^{2}\Delta x \\ f_{zzy}(\Delta z)^{2}\Delta y \\ f_{zzz}(\Delta z)^{3} \end{array}\right\} \qquad (A-9)$$

The inverse of this matrix is the matrix of difference coefficients, and is given below.

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{11}{6} & 3 & 0 & 0 & -\frac{3}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{11}{6} & 0 & 3 & 0 & 0 & 0 & -\frac{3}{2} & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\
-\frac{11}{6} & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -\frac{3}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\
2 & -5 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -\frac{5}{2} & -\frac{5}{2} & 0 & \frac{1}{2} & 3 & \frac{1}{2} & 0 & 0 & 0 & 0 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & -5 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & -\frac{5}{2} & 0 & -\frac{5}{2} & \frac{1}{2} & 0 & 0 & 3 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & 0 & -\frac{1}{2} & 0 & 0 \\
2 & 0 & -\frac{5}{2} & -\frac{5}{2} & 0 & 0 & \frac{1}{2} & 0 & 3 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{2} & 0 & -\frac{1}{2} & 0 \\
2 & 0 & 0 & -5 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
-1 & 3 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & 1 & 0 & -1 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 2 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 3 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & 0 & 1 & -1 & 0 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 1 & 1 & 0 & -1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 2 & 1 & 0 & 0 & -1 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
-1 & 1 & 0 & 2 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
-1 & 0 & 1 & 2 & 0 & 0 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
-1 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

Organization of Terms - The number of unique terms in a symmetric tensor of order M and dimension N, is summarized as follows.

Order	Number of Terms
0	1
1	N
2	N(N+1)/2
3	N(N+1)(N+2)/6
M	N(N+1)(N+2) ... (N+M-1)/M!

These relations also give the number of unique Mth order partial derivatives involved in an Mth order Taylor series expansion of N variables. For example, the coefficients of a second-order expansion in three variables (x,y,z) are the 3(3+1)/2 = 6 second partial derivatives (xx,yx,yy,zx,zy,zz). The total number of terms required in the expansion is the sum of the numbers of partial derivatives of each order, for example a second-order expansion in three variables requires 1 + 3 + 3(3+1)/2 = 10 total terms, and therefore a minimum of 10 function evaluations to determine the coefficient (partial derivative) values.

The required coefficients for an expansion are conveniently organized into a one dimensional array, for example for a function of three variables the array is

$$(f^1 \; f_1 \; f_2 \; f_3 \; f_{11} \; f_{21} \; f_{22} \; f_{31} \; \cdots \; f_{111} \; \cdots)$$

The terms are then easily retrieved from this array using the tensor relations given above. Thus using L_i, L_{ij}, L_{ijk}, ... to denote the location of the respective i, ij, ijk, ... derivative terms within the array, we have for a function of N variables

$$L_0 \equiv 1$$
$$L_i = L_0 + i$$
$$L_{00} \equiv 1 + N$$

$$L_{ij} = L_{00} + (i-1)\;(i)/2 + j$$
$$L_{000} \equiv L_{00} + N(N+1)/2$$
$$L_{ijk} = L_{000} + (i-1)(i)(i+1)/6 + (j-1)(j)/2 + k$$

General Discussion - It may be observed that the forward difference scheme outlined here requires a minimum number of function evaluations, i.e. one plus the number of partial derivatives involved in the expansion. Along with its other advantages of possible improved accuracy in certain types of situations, this would seem to indicate that the forward difference formulas presented here constitute the best approach for the required function expansions.

An interesting alternative derivation of the difference coefficient matrices is possible by a procedure used in the finite element method. The function to be expanded may be thought of as the quantity (say displacement) being interpolated within the finite element. The function is then defined in terms of its values at the nodes (i.e. the independent variable values) times the corresponding shape functions. The required partial derivatives can then be evaluated explicitly at the "origin", i.e. a corner node at which the sides of the element form an orthogonal coordinate system, simply by differentiating the element shape functions. Of course the appropriate finite element shape functions must be available, but if they are then this process allows derivation of the difference formulas without inversion of a matrix. These considerations were initially responsible for the selection of the forward difference approach, and the second and third order coefficients were evaluated in this manner, for example the third order coefficients using the TET 20 element of Argyris et al. (1968).

APPENDIX B: NONCONSERVATIVE LOADING EFFECTS

General Considerations - Several types of nonconservative loading can occur in finite element analysis, for example where the applied generalized nodal forces depend on the system displacements, velocities, or other displacement or deformation parameters. The nonconservative effects considered in the present work are those due to nodal forces which are a function of the nodal displacements, and in particular the effects of pressure loadings where the pressurized surface undergoes significant changes in area and orientation. General formulations may be developed for these cases, in terms of area integrals and pressure intensities. However, in order to illustrate completely the basic effects, the nonconservative load terms are derived here for the special case of a constant-strain-triangle (CST) finite element.

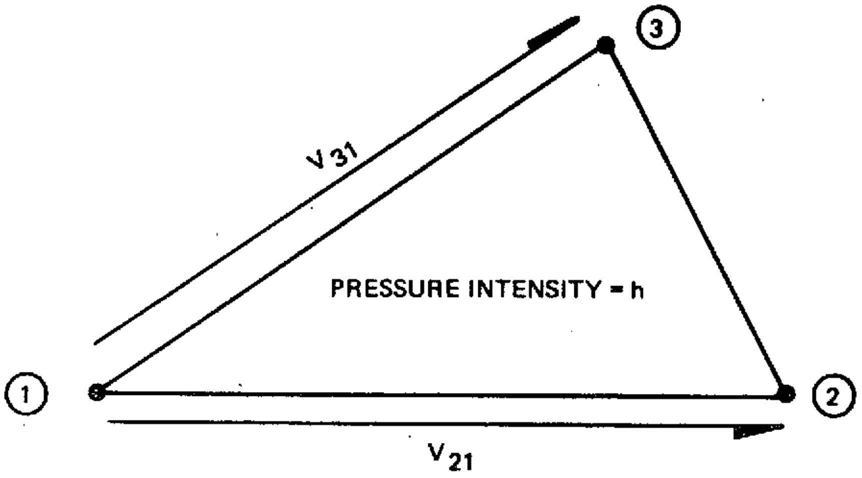


Figure B-1: CST Element for Nonconservative Loading

CST Element Definitions - Referring to Figure B-1, consider the CST element with nodes 1, 2 and 3, and subjected to a normal pressure loading of intensity h. The vectors V_{21} and V_{31} connect the nodes 1-2 and 1-3, respectively. The cross product $V_{21} \times V_{31}$ is then a vector with (positive) direction normal to the element surface and with magnitude equal to twice the element area.

Coordinates in the initial configuration are defined by the components x,y,z, and corresponding displacements by u,v,w. Then by defining for node k, the quantities

$$\left.\begin{aligned} cx_k &= (x_k + u_k) - (x_1 + u_1) \\ cy_k &= (y_k + V_k) - (y_1 + V_1) \\ Cz_k &= (z_k + w_k) - (z_1 + w_1) \end{aligned}\right\} \qquad \text{(B-1a)}$$

we may write the vectors V as

$$\left.\begin{aligned} V_{21} &= (cx_2, cy_2, cz_2) \\ V_{31} &= (cx_3, cy_3, cz_3) \end{aligned}\right\} \qquad \text{(B-1b)}$$

while their rates are given by

$$\left.\begin{aligned} \dot{V}_{21} &= (\dot{u}_2 - \dot{u}_1, \dot{v}_2 - \dot{v}_1, \dot{w}_2 - \dot{w}_1) \\ \dot{V}_{31} &= (\dot{u}_3 - \dot{u}_1, \dot{v}_3 - \dot{v}_1, \dot{w}_3 - \dot{w}_1) \end{aligned}\right\} \qquad \text{(B-1c)}$$

B.1 Fundamental Equilibrium Formulation

In this section the nonconservative loading effects are formulated for the CST element, and used to generalize the fundamental equilibrium equations in Section 2 to analysis of nonconservative systems.

Basic Load Equation - For a uniform pressure loading, each node of the CST element has an equivalent concentrated load vector p, given by

$$p_i = \frac{1}{6} h(V_{21} \times V_{31})_i \qquad \text{(B-2a)}$$

The nodal force during an increment of loading is defined by the basic nonconservative load equation

$$p_i = \frac{1}{6} (h^* + \Lambda h^\circ)(V_{21} \times V_{31})_i \qquad \text{(B-2b)}$$

where h* is the pressure intensity at the reference equilibrium configuration, h° is a constant pressure distribution and Λ is the incremental load parameter.

First-Order Load Equation - Differentiating equation (B-2b) with respect to the fundamental path parameter, gives

$$\dot{p}_i = \frac{1}{6} \dot{\Lambda} h^\circ (V_{21} \times V_{31})_i + \frac{1}{6} (h^* + \Lambda h^\circ)(\dot{V}_{21} \times V_{31} + V_{21} \times \dot{V}_{31})_i \qquad \text{(B-3a)}$$

Evaluating at the reference equilibrium configuration (Λ = 0, V = V*, etc.) gives

$$\dot{p}^*_i = \frac{1}{6} \dot{\Lambda}^* h^\circ (V^*_{21} \times V^*_{31})_i + \frac{1}{6} h^* (\dot{V}^*_{21} \times V_{31} + V^*_{21} \times \dot{V}^*_{31}) \qquad \text{(B-3b)}$$

This is the first order nonconservative load equation. To put it into the desired form, we write the vector q of element displacements as

$$q = (u_1 \; v_1 \; w_1 \; u_2 \; v_2 \; w_2 \; u_3 \; v_3 \; w_3)$$

from which it follows using equations (B-1), that

$$\frac{1}{6} h (V_{21} \times \dot{V}_{31} + \dot{V}_{21} \times V_{31})_i = C_{ij} \dot{q}_j$$

where

$$C_{ij} = \frac{1}{6} h \left[\begin{array}{c|c|c|c|c|c|c|c|c} 0 & \begin{matrix} cz_2 \\ -cz_3 \end{matrix} & \begin{matrix} -cy_2 \\ +cy_3 \end{matrix} & 0 & cz_3 & -cy_3 & 0 & -cz_2 & cy_2 \\ \hline \begin{matrix} -cz_2 \\ +cz_3 \end{matrix} & 0 & \begin{matrix} cx_2 \\ -cx_3 \end{matrix} & -cz_3 & 0 & cx_3 & cz_2 & 0 & -cx_2 \\ \hline \begin{matrix} cy_2 \\ -cy_3 \end{matrix} & \begin{matrix} -cx_2 \\ +cx_3 \end{matrix} & 0 & cy_3 & -cx_3 & 0 & -cy_2 & cx_2 & 0 \end{array} \right]$$

The desired form of the first order nonconservative load equation is then

$$\dot{p}_i^* = \dot{\Lambda}^* p_i^\circ + C_{ij}^* \dot{q}_j^* \qquad \text{(B-3c)}$$

where $\quad p_i^\circ = \frac{1}{6} h^\circ \, (V_{21}^* \times V_{31}^*)_i$

The first part of $\dot{p}^*$ in (B-3c) is a usual nodal load rate, and occurs in the nonconservative form of equilibrium equation (2-7c) as a contribution to the load term P° (see(2-9)). The second part of $\dot{p}^*$ occurs in the nonconservative form of (2-7c) as an unsymmetric contribution to the Jacobian stiffness K0* (note that since C_{ij}^* occurs on the left-hand-side of (2-7c), it must be subtracted from K0*).

<u>Second-Order Load Equation</u> - A second differentiation of (B-2b) gives

$$\ddot{p}_i = \frac{1}{6} \ddot{\Lambda} h^\circ \, (V_{21} \times V_{31})_i + \frac{1}{3} \dot{\Lambda} h^\circ \, (\dot{V}_{21} \times V_{31} + V_{21} \times \dot{V}_{31})_i$$

$$+ \frac{1}{6} (h^* + \Lambda h^\circ)(\ddot{V}_{21} \times V_{31} + V_{21} \times \ddot{V}_{31} + 2\dot{V}_{21} \times \dot{V}_{31}) \qquad \text{(B-4a)}$$

Evaluating at the reference equilibrium configuration provides

$$\ddot{p}_i^* = \frac{1}{6}\ddot{\Lambda}^* h^\circ (V_{21}^* \times V_{31}^*)_i + \frac{1}{3}\dot{\Lambda}^* h^\circ (\dot{V}_{21}^* \times V_{31}^* + V_{21}^* \times \dot{V}_{31}^*)_i$$

$$+ \frac{1}{6} h^* (\ddot{V}_{21}^* \times V_{31}^* + V_{21}^* \times \ddot{V}_{31}^* + 2\dot{V}_{21}^* \times \dot{V}_{31}^*)_i \qquad \text{(B-4b)}$$

This is the second order nonconservative load equation, which may be written in the form

$$\ddot{p}_i^* = C_{ij}^* \ddot{q}_j^* + pl_i^* \qquad \text{(B-4c)}$$

where $pl_i^* = \frac{1}{6}\ddot{\Lambda}^* h^\circ (V_{21}^* \times V_{31}^*)_i + \frac{1}{3}\dot{\Lambda}^* h^\circ (\dot{V}_{21}^* \times V_{31}^*$

$$+ V_{21}^* \times \dot{V}_{31}^*)_i + \frac{1}{3} h^* (\dot{V}_{21}^* \times \dot{V}_{31}^*)_i$$

B.2 Bifurcation and Postbuckling Formulation

Basic Load Equation - Development of the nonconservative load effects for bifurcation and postbuckling follows the fundamental relations of section B.1. Using equation (B-2b), we may write the nodal force during an increment of loading on the postbuckling path, as

$$p_i = p_i^f + p_i^p = \frac{1}{6}(h^* + \Lambda h^\circ)((V_{21}^f + V_{21}^p) \times (V_{31}^f + V_{31}^p))_i \qquad \text{(B-5a)}$$

Since this relation will be used to establish a nonconservative form of the postbuckling equilibrium equation (3-3), the fundamental load contribution must be subtracted from (B-5a). We then obtain

$$p_i^p = \frac{1}{6}(h^* + \Lambda h^\circ)(V_{21}^f \times V_{31}^p + V_{21}^p \times V_{31}^f + V_{21}^p \times V_{31}^p)_i \qquad \text{(B-5b)}$$

First-Order Load Equation - Differentiating equation (B-5b) with respect to the postbuckling path parameter, gives

$$p_i^{'p} = \frac{1}{6} \Lambda' \; h^\circ \, (V_{21}^f \times V_{31}^p + V_{21}^p \times V_{31}^f + V_{21}^p \times V_{31}^p)_i$$
$$+ \frac{1}{6} (h^* + \Lambda \, h^\circ)(V_{21}^{'f} \times V_{31}^p + V_{21}^f \times V_{31}^{'p} + V_{21}^{'p} \times V_{31}^f + V_{21}^p \times V_{31}^{'f}$$
$$+ V_{21}^{'p} \times V_{31}^p + V_{21}^p \times V_{31}^{'p})_i \qquad \text{(B-6a)}$$

Evaluating at the critical bifurcation point ($V^p = 0$), gives

$$p_i^{'p} = \frac{1}{6} (h^* + \Lambda \, h^\circ)(V_{21}^f \times V_{31}^{'p} + V_{21}^{'p} \times V_{31}^f) \qquad \text{(B-6b)}$$

Substituting for V^f gives

$$p_i^{'p} = \frac{1}{6} (h^* + \Lambda \, h^\circ)(V_{21}^* \times V_{31}^{'p} + \Delta V_{21}^f \times V_{31}^{'p} + V_{21}^{'p} \times V_{31}^*$$
$$+ V_{21}^{'p} \times \Delta V_{31}^f)_i \qquad \text{(B-6c)}$$

and using the relations which express $V^{'p}$ in terms of $q^{'p}$, gives

$$p_i^{'p} = C_{ij}^* q_j^{'p} + \frac{1}{6} h^* \, (\Delta V_{21}^f \times V_{31}^{'p} + V_{21}^{'p} \times \Delta V_{31}^f)_i$$
$$+ \frac{1}{6} \Lambda \, h^\circ \, (V_{21}^* \times V_{31}^{'p} + \Delta V_{21}^f \times V_{31}^{'p} + V_{21}^{'p} \times V_{31}^* + V_{21}^{'p} \times \Delta V_{31}^f)_i$$

(B-6d)

This is the first order postbuckling nonconservative load expression, which may be written in the form

$$p_i^{'p} = C_{ij}^* q_j^{'p} + pl_i^1 \qquad \text{(B-6e)}$$

where

$$pl_i^1 = \frac{1}{6} \Lambda \, h^\circ \, (V_{21}^* \times V_{31}^{'p} + V_{21}^{'p} \times V_{31}^*)_i$$
$$+ \frac{1}{6} (h^* + \Lambda \, h^\circ)(\Delta V_{21}^f \times V_{31}^{'p} + V_{21}^{'p} \times \Delta V_{31}^f)_i$$

Expression (B-6e) provides the necessary nonconservative addition to the load term, in the first order postbuckling equation (3-5d). The Jacobian K0* again becomes nonsymmetric due to the subtraction of the C* terms. Thus a nonlinear, nonconservative eigen equation is produced, which may be solved by the same approaches discussed in Section 3 for the symmetric problem.

Second-Order Load Equation - A second differentiation of (B-5b) with respect to the postbuckling path parameter, gives

$$
\begin{aligned}
p_i^{\prime\prime p} = {} & \tfrac{1}{6}\Lambda'' h^\circ (V_{21}^{f} x V_{31}^{p} + V_{21}^{p} x V_{31}^{f} + V_{21}^{p} x V_{31}^{p})_i \\
& + \tfrac{1}{3}\Lambda' h^\circ (V'^{f}_{21} x V_{31}^{p} + V_{21}^{f} x V'^{p}_{31} + V'^{p}_{21} x V_{31}^{f} + V_{21}^{p} x V'^{f}_{31} \\
& + V'^{p}_{21} x V_{31}^{p} + V_{21}^{p} x V'^{p}_{31})_i \\
+ {} & \tfrac{1}{6}(h^* + \Lambda h^\circ)(V''^{f}_{21} x V_{31}^{p} + 2V'^{f}_{21} x V'^{p}_{31} + V_{21}^{f} x V''^{p}_{31} + V''^{p}_{21} x V_{31}^{f} \\
& + 2V'^{p}_{21} x V'^{f}_{31} + V_{21}^{p} x V''^{f}_{31} + V''^{p}_{21} x V_{31}^{p} + 2V'^{p}_{21} x V'^{p}_{31} + V_{21}^{p} x V''^{p}_{31})_i
\end{aligned}
\tag{B-7a}
$$

Evaluating at the critical point ($V^p = 0$), with the critical value of $h = h^* + \Lambda h^\circ$, gives

$$
\begin{aligned}
p_i^{\prime\prime p} = {} & \tfrac{1}{3}\Lambda' h^\circ (V_{21}^{f} x V'^{p}_{31} + V'^{p}_{21} x V_{31}^{f})_i \\
& + \tfrac{1}{6} h (2V'^{f}_{21} x V'^{p}_{31} + V_{21}^{f} x V''^{p}_{31} + V''^{p}_{21} x V_{31}^{f} + 2V'^{p}_{21} x V'^{f}_{31} \\
& + 2V'^{p}_{21} x V'^{p}_{31})_i
\end{aligned}
\tag{B-7b}
$$

Using the relations which express V''^{p} in terms of q''^{p}, gives

$$
\begin{aligned}
p_i^{\prime\prime p} = {} & \tfrac{1}{3}\Lambda' h^\circ (V_{21}^{f} x V'^{p}_{31} + V'^{p}_{21} x V_{21}^{f})_i \\
& + C_{ij} q_j^{\prime\prime p} + \tfrac{1}{6} h (2V'^{f}_{21} x V'^{p}_{31} + 2V'^{p}_{21} x V'^{f}_{31} + 2V'^{p}_{21} x V'^{p}_{31})_i
\end{aligned}
\tag{B-7c}
$$

This is the second order postbuckling nonconservative load expression, which may be written in the form

$$p_i^{\prime\prime p} = C_{ij}q_j^{\prime\prime p} + 2S'p2_i^1 + p2_i^2 \qquad \text{(B-7d)}$$

where

$$p2_i^1 = \frac{1}{6}\dot{\Lambda}\, h^\circ (V_{21}^f x V_{31}^{\prime p} + V_{21}^{\prime p} x V_{31}^f)_i$$

$$+ \frac{1}{6}h(\dot{V}_{21}^f x V_{31}^{\prime p} + V_{21}^{\prime p} x \dot{V}_{31}^f)_i$$

and

$$p2_i^2 = \frac{1}{3}h(V_{21}^{\prime p} x V_{31}^{\prime p})_i$$

Third-Order Load Equation - A third differentiation of (B-5b) and evaluation at the critical point, provides

$$p_i^{\prime\prime p} = \frac{1}{2}\Lambda''h^\circ(V_{21}^f x V_{31}^{\prime p} + V_{21}^{\prime p} x V_{31}^f)_i$$

$$+ \frac{1}{2}\Lambda'h^\circ(V_{21}^f x V_{31}^{\prime\prime p} + 2V_{21}^{\prime f} x V_{31}^{\prime p} + V_{21}^{\prime\prime p} x V_{31}^f + 2V_{21}^{\prime p} x V_{31}^{\prime f}$$

$$+ 2V_{21}^{\prime p} x V_{31}^{\prime p})_i$$

$$+ \frac{1}{6}h(V_{21}^f x V_{31}^{\prime\prime\prime p} + 3V_{21}^{\prime\prime f} x V_{31}^{\prime p} + 3V_{21}^{\prime f} x V_{31}^{\prime\prime p} + V_{21}^{\prime\prime\prime p} x V_{31}^f + 3V_{21}^{\prime\prime p} x V_{31}^{\prime f}$$

$$+ 3V_{21}^{\prime p} x V_{31}^{\prime\prime f} + 3V_{21}^{\prime\prime p} x V_{31}^{\prime p} + 3V_{21}^{\prime p} x V_{21}^{\prime\prime p})_i \qquad \text{(B-8a)}$$

Using the relations which express V'''^{p} in terms of q'''^{p}, gives

$$
\begin{aligned}
p_i^{'''p} = {} & \frac{1}{2}\Lambda''h^\circ(V^f_{21}xV'^p_{31} + V'^p_{21}xV^f_{31})_i + \frac{1}{2}\Lambda'h^\circ(V^f_{21}xV''^p_{31} \\
& + 2V'^f_{21}xV'^p_{31} + V''^p_{21}xV^f_{31} + 2V'^p_{21}xV'^f_{31}) + 2V'^p_{21}xV'^p_{31})_i + C_{ij}q_j^{'''p} \\
& + \frac{1}{2}h(V''^f_{21}xV'^p_{31} + V'^f_{21}xV''^p_{31} + V''^p_{21}xV'^f_{31} + V'^p_{21}xV''^f_{31} + V''^p_{21}xV'^p_{31} \\
& + V'^p_{21}xV''^p_{31})_i
\end{aligned}
\tag{B-8b}
$$

This is the third order postbuckling nonconservative load expression, which may be written in the form

$$
p_i^{'''p} = C_{ij}q_j^{'''p} + 3(S''p2^1_i + p3_i)
\tag{B-8c}
$$

where

$$
\begin{aligned}
p3_i = {} & S'^2\{\frac{1}{6}\ddot{\Lambda}h^\circ(V^f_{21}xV'^p_{31} + V'^p_{21}xV^f_{31})_i + \frac{1}{3}\dot{\Lambda}h^\circ(\dot{V}^f_{21}xV'^p_{31} + V'^p_{21}x\dot{V}^f_{31})_i \\
& + \frac{1}{6}h(\ddot{V}^f_{21}xV'^p_{31} + V'^p_{21}x\ddot{V}^f_{31})_i\} \\
& + S'\{\frac{1}{6}\dot{\Lambda}h^\circ(V^f_{21}xV''^p_{31} + V''^p_{21}xV^f_{31} + 2V'^p_{21}xV'^p_{31})_i \\
& + \frac{1}{6}h(\dot{V}^f_{21}xV''^p_{31} + V''^p_{21}x\dot{V}^f_{31})_i\} \\
& + \frac{1}{6}h(V''^p_{21}xV'^p_{31} + V'^p_{21}xV''^p_{31})_i
\end{aligned}
$$

APPENDIX C: PANES PROGRAM LISTING

This appendix contains a FORTRAN IV listing of the PANES (Program for Analysis of Nonlinear Equilibrium and Stability) program. Following the program listing is a listing of input data for the torus problem described in section 5.3.

```
C     ******************************************************************
C     P A N E S(PROGRAM FOR ANALYSIS OF NONLINEAR EQUILIBRIUM/STABILITY)
C     R.G. VOS, THE BOEING COMPANY, PHONE 773-2638, KENT, WASHINGTON
C     PANES IBM 360 VERSION (78 DOF) DATED 09/30/74
C     ******************************************************************
C     APPLICABLE TO NONLINEAR NONCONSERVATIVE HYPERELASTIC SYSTEMS.
C     CST ELEMENT, 26 NODES, 78 DOF, 24 ELEMENTS, 20 LOAD INCREMENTS.
C     5 MATERIALS, 2 OR 3 DOF PER NODE.
C     NODES ARE LOCATED IN BASIC-CARTESIAN OR CYLINDRICAL COORDINATES
C     AND DISPLACEMENTS ARE IN BASIC,CYLINDRICAL, OR SPECIAL-CARTESIAN.
      INTEGER UIN1,UIN2,UOUT,UINRS,UOUTRS
      INTEGER NOD,NEL,NN,NF,NS,ORD,I,MJUMP,NJUMP
      INTEGER IPRESS,PRED,IPRED(20),MAXUP,MAXIT,NMAT,IMAT(24)
      INTEGER NINCR,INCR,ISTEP,ITER,ELNO(3,24),KFD(78),IDET
      DOUBLE PRECISION PFACT(2,20),PREF(2,78),P0(2),P1(2),PA(2)
      DOUBLE PRECISION PRFACT(20),PRREF(24),PR0,PR1,PRA
      DOUBLE PRECISION PP(78),QQ(78),P(78),Q(78),RQ(78),RRQ(78),PDUM(78)
      DOUBLE PRECISION LSIGN,RL,RRL,PATH,ERR,ERRMAX,DET
      DOUBLE PRECISION LAM,LAMS,LAMR,FLAMAX,LAMIN,JUMPR,RJUMP,SLOPED
      DOUBLE PRECISION COORDA(5),E(5),NU(5),C,DFE,DFF
      DOUBLE PRECISION GCOS(9,26),COORD(3,26),KMAT(78,78)
      DOUBLE PRECISION T(24),EGEOM(3,24),EET(3,24)
      COMMON/COMNEL/NEL/COMNS/NS/COMORD/ORD/COMDFE/DFE/COMEET/EET
      COMMON/COMNF/NF/COMMAT/IMAT/COMENU/E,NU/COMQQ/QQ
      COMMON/COMCOS/GCOS/COMCOR/COORD/COMEL/ELNO/COMEG/EGEOM/COMT/T
      COMMON/COMIPR/IPRESS/COMPR/PRA/COMPRR/PRREF
C           START PROBLEM
    1 CALL READRS(UIN1,UIN2,UOUT,INCR,UINRS,UOUTRS)
      CALL READ0(5,UOUT,ORD,PRED,MAXUP,MAXIT,ERRMAX,DFE,DFF,
     1MJUMP,JUMPR,SLOPED,FLAMAX,LAMIN)
      IF(UINRS.GT.0)GO TO 11
C           COLD START
      CALL BIGS(1,UIN1,UOUT,      IPRESS,NF,NMAT,E,NU,
     1COORDA,NOD,NEL,COORD,GCOS,IMAT,T,ELNO,EGEOM,KFD,PREF,PRREF,
     2LSIGN,PP,QQ,P1,PR1)
      NJUMP = MJUMP+1
      GO TO 16
C           READ RESTART TAPE
   11 CALL BIGS(2,UINRS,INCR,     IPRESS,NF,NMAT,E,NU,
     1COORDA,NOD,NEL,COORD,GCOS,IMAT,T,ELNO,EGEOM,KFD,PREF,PRREF,
     2LSIGN,PP,QQ,P1,PR1)
      NJUMP = MJUMP+1
   16 IF(UOUTRS.EQ.0)GO TO 17
C          WRITE RESTART TAPE
      CALL BIGS(3,UOUTRS,0,       IPRESS,NF,NMAT,E,NU,
     1COORDA,NOD,NEL,COORD,GCOS,IMAT,T,ELNO,EGEOM,KFD,PREF,PRREF,
     2LSIGN,PP,QQ,P1,PR1)
C          GENERAL PROGRAM FLOW
   17 NS = 3
      NN = NOD*NF
C     READ INCREMENTAL LOAD FACTOR DATA
      CALL READI(5,UOUT,NINCR,PRED,IPRED,PFACT,PRFACT)
C          BEGIN LOAD INCREMENT LOOP
      DO 1000 INCR=1,NINCR
C     LAM = INCREMENTAL LOAD PARAMETER (MAXIMUM VALUE 1.0).
C     LAMR = LOAD YET TO BE APPLIED = 1.0 - LAM.
C     LAMS = LOAD STEP PARAMETER = FRACTION OF LAMR TO BE APPLIED.
      LAM = 0.D0
C     LSIGN = +,- FOR LOADING,UNLOADING SITUATION.
      IF(LSIGN.LT.0.D0)STOP 101
C     P0(I),P1(I),PA(I) = LOAD FACTORS FOR LOAD REFERENCE VECTOR I.
C     PR0,PR1,PRA = PRESSURE LOAD FACTORS FOR PRESSURE REFERENCE VECTOR.
C     0,1 DENOTE VALUES AT START,END OF INCREMENT.
C     A DENOTES ACTUAL APPLIED VALUE,WHICH AT THIS POINT = 0 VALUE.
      DO 200 I=1,2
      P0(I) = P1(I)
      P1(I) = PFACT(I,INCR)
  200 PA(I) = P0(I)
      PR0 = PR1
      PR1 = PRFACT(INCR)
      PRA = PR0
      ISTEP = 0
C          BEGIN LOAD STEP
  201 ISTEP = ISTEP+1
      NUP = 0
      LAMR = 1.D0-LAM
C     SET UPPER BOUND FOR ABSOLUTE VALUE OF LOAD STEP SIZE LAMS.
      LAMS = 1.D0
      IF(LSIGN.LT.0)LAMS = FLAMAX/LAMR
C     CALL PFORCE TO GIVE APPLIED CONSERVATIVE NODAL LOADS P.
```

```
C     CALL EFORCE TO GIVE APPLIED NONCONSERVATIVE NODAL LOADS Q.
      CALL PFORCE(P1,PREF,NN,P)
      CALL EFORCE(IPRESS,PR1,PRREF,QQ,NEL,NN,NF,ELNO,Q)
C     COMPUTE LOAD STEP NODAL LOADS.  THESE = APPLIED LOAD - INTERNAL
C       LOAD FOR SPECIFIED FORCE DOF, APPLIED DISPLACEMENT - CURRENT
C       DISPLACEMENT FOR SPECIFIED DISPLACEMENT DOF.
      DO 210 I=1,NN
      IF(KFD(I).GT.0)C = P(I) + Q(I) - PP(I)
      IF(KFD(I).LT.0)C = P(I) - QQ(I)
  210 P(I) = C
C     FORM JACOBIAN AT BEGINNING OF LOAD STEP.
      CALL MERGE(KMAT,ELNO,KFD,NEL,NN,NF)
      CALL DECOMP(KMAT,NN,KFD,IDET,DET)
C          FUNDAMENTAL PATH PREDICTOR CODE
      IF(IPRED(INCR).GE.2)GO TO 241
C     APPLY LINEAR PREDICTOR FOR LOAD STEP
      CALL SOLVE(KMAT,NN,KFD,P,Q)
      LAMS = LSIGN*LAMS
      DO 230 I=1,NN
  230 Q(I) = LAMS*Q(I)
      GO TO 301
C     APPLY QUADRATIC PREDICTOR FOR LOAD STEP
  241 CALL RATES(KMAT,PDUM,NN,KFD,P,PRA,PR1,LSIGN,RL,RRL,RQ,RRQ)
      RJUMP = JUMPR/LAMR
      CALL STEP(LSIGN,RL,RRL,NN,RQ,RRQ,RJUMP,MJUMP,NJUMP,SLOPED,
     1PATH,LAMS)
C     IF LIMIT POINT WAS TRAVERSED (I.E. LAMS = 0) OUTPUT LIMIT RESULTS.
      IF(LAMS.EQ.0.D0)
     1CALL OUTLIM(6,NOD,NEL,NN,NF,ELNO,EGEOM,EET,CC,P,Q,DFF,
     2RL,RRL,RQ,RRQ,LAM,LAMR)
C     ******************************************************************
C     ******************************************************************
C     THIS SECTION OF CODE IS TEMPORY POSTBUCKLING CHECKOUT CODE.
      INTEGER BCODE,PCODE,IPOST,IIPOST
      DOUBLE PRECISION SCRIT,LCRIT,LDOT1,LDOT2,QQDOT1(78),QQDOT2(78),
     1LPOST1,LPOST2,QQPOS1(78),QQPOS2(78),QQPOS3(78),QQCRIT(78)
      DOUBLE PRECISION PDUM2(78),PDUM3(78),PDUM4(78)
C     SET BCODE=1 TO GET NONLINEAR EIGEN SOLUTION FOR BIFURCATION POINT.
C     ALSO SET PCODE=1 TO GET ADDITIONAL POSTBUCKLING PATH SOLUTION.
      BCODE = 0
      PCODE = 1
      IF(BCODE.EQ.0)GO TO 279
      CALL EIGEN(UOUT,KMAT,PDUM,Q,NN,KFD,PRA,PR1,RL,RRL,RQ,RRQ,
     25,15,1.D-2,1.D-5,SCRIT,QQPOS1,IPOST)
      CALL OUTPQ(UOUT,NOD,NF,QQPOS1,QQPOS1)
      LCRIT = RL*SCRIT + .5D0*RRL*SCRIT**2
      IF(PCODE.EQ.0)GO TO 279
C     PERFORM POSTBUCKLING SOLUTION
      LCRIT = RL*SCRIT
      LDOT1 = RL
      LDOT2 = 0.D0
      DO 265 I=1,NN
      QQCRIT(I) = QQ(I) + RQ(I)*SCRIT
      QQDOT1(I) = RQ(I)
  265 QQDOT2(I) = 0.D0
      IF(IPRED(INCR).LT.2)GO TO 271
      LCRIT = LCRIT + .5D0*RRL*SCRIT**2
      LDOT1 = LDOT1 + RRL*SCRIT
      LDOT2 = RRL
      DO 270 I=1,NN
      QQCRIT(I) = QQCRIT(I) + .5D0*RRQ(I)*SCRIT**2
      QQDOT1(I) = QQDOT1(I) + RRQ(I)*SCRIT
  270 QQDOT2(I) = RRQ(I)
      CALL OUTPQ(UOUT,NOD,NF,QQDOT1,QQDOT2)
      DO 268 I=1,NN
      C = QQ(I)
      QQ(I) = QQCRIT(I)
  268 QQCRIT(I) = C
  271 CALL MERGE(KMAT,ELNO,KFD,NEL,NN,NF)
      DO 272 I=1,NN
      C = QQ(I)
      QQ(I) = QQCRIT(I)
  272 QQCRIT(I) = C
      IIPOST = KFD(IPOST)
      KFD(IPOST) = -IPOST
      CALL DECOMP(KMAT,NN,KFD,IDET,DET)
      CALL PRATES(KMAT,PDUM,PDUM2,PDUM3,PDUM4,NN,IPOST,KFD,SCRIT,
     1QQCRIT,QQDOT1,QQDOT2,LCRIT,LDOT1,LDOT2,PRO,PRA,
     2QQPOS1,LPOST1,QQPOS2,LPOST2,QQPOS3)
      CALL OUTPQ(UOUT,NOD,NF,QQCRIT,QQPOS1)
      CALL OUTPQ(UOUT,NOD,NF,QQPOS2,QQPOS3)
      KFD(IPOST) = IIPOST
C     ******************************************************************
C     ******************************************************************
C          PATH CONTINUATION CODE
  279 CONTINUE
      DO 280 I=1,NN
  280 Q(I) = PATH*RQ(I) + .5D0*PATH**2*RRQ(I)
C     ADD STEP LOAD LAMS TO INCREMENT LOAD SUM LAM
      LAMS = LAMS*LAMR
  301 LAM = LAM + LAMS
      DO 305 I=1,2
  305 PA(I) = P0(I) + LAM*(P1(I)-P0(I))
      CALL PFORCE(PA,PREF,NN,P)
      DO 310 I=1,NN
      IF(KFD(I).GT.0)QQ(I) = QQ(I) + Q(I)
      IF(KFD(I).LT.0)QQ(I) = P(I)
  310 CONTINUE
      PRA = PR0 + LAM*(PR1-PR0)
      ITER = 0
      GO TO 451
C          BEGIN ITERATION LOOP
  311 ITER = ITER+1
      IF(ITER.GT.1)GO TO 401
C     FORM JACOBIAN AT APPROXIMATE END OF STEP
      CALL MERGE(KMAT,ELNO,KFD,NEL,NN,NF)
      CALL DECOMP(KMAT,NN,KFD,IDET,DET)
  401 CALL SOLVE(KMAT,NN,KFD,P,Q)
C     UPDATE INTERNAL FORCES PP AND DISPLACEMENTS QQ.
C     COMPUTE APPLIED EXTERNAL LOADS (CONSERVATIVE P,NONCONSERVATIVE Q)
      DO 410 I=1,NN
  410 QQ(I) = QQ(I) + Q(I)
  451 CALL FORCE(NEL,NN,NF,ELNO,DFF,QQ,PP)
      CALL PFORCE(PA,PREF,NN,P)
      CALL EFORCE(IPRESS,PRA,PRREF,QQ,NEL,NN,NF,ELNO,Q)
      DO 460 I=1,NN
      C = 0.D0
      IF(KFD(I).GT.0)C = P(I) + Q(I) - PP(I)
  460 P(I) = C
      CALL ERCOMP(UOUT,NN,KFD,PP,P,ERR)
```

```
C          END ITERATION LOOP
      IF(ERR.LE.ERRMAX)GO TO 901
      IF(ITER.LT.MAXIT)GO TO 311
C     IF THIS POINT IS REACHED, MAX. NC. ITERATIONS OCCUR
      IF(NUP.GE.MAXUP)GO TO 901
      NUP = NUP+1
      ITER = 0
      GO TO 311
C          OUTPUT INCREMENTAL STEP RESULTS
  901 CALL HEAD(UOUT,INCR,ISTEP,LAMS,LAM,PFACT(1,INCR),PFACT(2,INCR),
     1PRFACT(INCR),IPRED(INCR),MAXUP,NUP,MAXIT,ITER,ERRMAX,ERR)
      CALL OUTPQ(UOUT,NOD,NF,PP,QQ)
      CALL STRAIN(NEL,ELNO,EGEOM,NF,QQ,EET)
      CALL OUTE(UOUT,NEL,EET)
C          END LOAD STEP
      IF(LAM.LT.LAMIN)GO TO 1
      IF(LAM.LT..99900)GO TO 201
      IF(UOUTRS.EQ.0)GO TO 1000
C          WRITE INCREMENTAL RESTART TAPE
      CALL BIGS(3,UOUTRS,INCR,    IPRESS,NF,NMAT,E,NU,
     1CCORDA,NOD,NEL,COORD,GCOS,IMAT,T,ELNO,EGEOM,KFD,PREF,PRREF,
     2LSIGN,PP,QQ,P1,PR1)
 1000 CONTINUE
C          END LOAD INCREMENT LOOP
      GO TO 1
      END


      SUBROUTINE BIGS(KODE,I1,I2,IPRESS,NF,NMAT,E,NU,
     1CCORDA,NOD,NEL,COORD,GCOS,IMAT,T,ELNO,EGEOM,KFD,PREF,PRREF,
     2LSIGN,PP,QQ,P1,PR1)
C     KCDE = 1,2,3 = INITIALIZE, READ RESTART, WRITE RESTART.
C     I1 = INPUT OR RESTART INPUT-OUTPUT FILE UNIT NUMBER.
C     I2 = OUTPUT FILE UNIT NUMBER, OR INCREMENT NUMBER FOR RESTART.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     NF = NUMBER OF FREEDOMS PER NODE.
C     NMAT = NUMBER OF MATERIALS.
C     E,NU = MATERIAL CONSTANTS (E.G. ELASTIC MODULUS,POISSON'S RATIO).
C     CCORDA(I) = ANGLE FOR SPECIAL CCORDINATE SYSTEM I.
C     NOD,NEL = NUMBER OF NODES,ELEMENTS.
```

```
C     COORD(J,I) = COORDINATES OF NODE I.
C     GCOS(J,I) = DIRECTION COSINES FOR NODE I.
C     IMAT(I) = MATERIAL NUMBER FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRY FOR ELEMENT I (BASE,HEIGHT,PART BASE).
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     PREF(J,I) = NODAL LOAD AT DOF I FOR LOAD REFERECNE VECTOR J.
C     PRREF(I) = INTENSITY ON ELEMENT I FOR PRESSURE REFERENCE VECTOR.
C     LSIGN = +,- IF LOAD PARAMETER IS INCREASING,DECREASING.
C     PP(I) = CURRENT INTERNAL NODAL FORCE AT DOF I.
C     QQ(I) = CURRENT NODAL DISPLACEMENT AT DOF I.
C     P1(J) = LOAD FACTOR TO BE APPLIED TO PREF(J,I).
C     PR1 = LOAD FACTOR TO BE APPLIED TO PRREF(I).
      INTEGER KODE,I1,I2,IPRESS,NF,NMAT,NOD,NEL,IMAT(1),ELNO(3,1),KFD(1)
      DOUBLE PRECISION E(1),NU(1),COORDA(1),COORD(3,1),GCOS(9,1),
     1T(1),EGEOM(3,1),PREF(2,1),PRREF(1),LSIGN,PP(1),QQ(1),P1(1),PR1
      EQUIVALENCE (J1,UIN,UINRS,UOUTRS),(J2,UOUT,INCR)
      INTEGER J1,UIN,UINRS,UOUTRS,J2,UOUT,INCR
      INTEGER INC,I,J,NN
      DOUBLE PRECISION THICK
      J1 = I1
      J2 = I2
      IF(KODE.NE.1)GO TO 101
C          INITIALIZE VARIABLES
      CALL READ1(UIN,UOUT,IPRESS,NF,THICK,NMAT,E,NU)
      CALL READC(UIN,UOUT,COORDA)
      CALL READM(UIN,UOUT,NOD,NEL,COORDA,COORD,GCOS,
     1IMAT,THICK,T,ELNO,EGEOM)
      CALL READK(UIN,UOUT,NOD,NF,KFD)
      CALL READP(UIN,UOUT,NOD,NF,KFD,PREF)
      IF(IPRESS.GT.0)CALL READPR(UIN,UOUT,NEL,PRREF)
      LSIGN = 1.D0
      NN = NOD*NF
      DO 50 I=1,NN
      PP(I) = 0.D0
   50 QQ(I) = 0.D0
      DO 60 I=1,2
   60 P1(I) = 0.D0
      PR1 = 0.D0
      RETURN
  101 IF(KODE.NE.2)GO TO 201
C          READ VARIABLES FROM RESTART TAPE
      READ(UINRS)IPRESS,NF,NMAT,(E(I),NU(I),I=1,NMAT),
     1NOD,NEL,((COORD(J,I),J=1,3),(GCOS(J,I),J=1,9),I=1,NOD),
     2(IMAT(I),T(I),(ELNO(J,I),EGEOM(J,I),J=1,3),PRREF(I),I=1,NEL)
      NN = NOD*NF
      READ(UINRS)(KFD(I),(PREF(J,I),J=1,2),I=1,NN)
  151 READ(UINRS)INC
      IF(INC.EQ.INCR)GO TO 161
      READ(UINRS)
      GO TO 151
  161 READ(UINRS)LSIGN,(PP(I),CQ(I),I=1,NN),(P1(I),I=1,2),PR1
      REWIND UINRS
      RETURN
  201 IF(KODE.NE.3)RETURN
C          WRITE VARIABLES ONTO RESTART TAPE
      IF(INCR.GT.0)GO TO 251
      WRITE(UOUTRS)IPRESS,NF,NMAT,(E(I),NU(I),I=1,NMAT),
     1NOD,NEL,((COORD(J,I),J=1,3),(GCOS(J,I),J=1,9),I=1,NOD),
     2(IMAT(I),T(I),(ELNO(J,I),EGEOM(J,I),J=1,3),PRREF(I),I=1,NEL)
      NN = NOD*NF
      WRITE(UOUTRS)(KFD(I),(PREF(J,I),J=1,2),I=1,NN)
  251 WRITE(UOUTRS)INCR
      WRITE(UOUTRS)LSIGN,(PP(I),CQ(I),I=1,NN),(P1(I),I=1,2),PR1
      RETURN
      END


      SUBROUTINE READRS(UIN1,UIN2,UOUT,INCR,UINRS,UOUTRS)
C     READ DATA FILE NUMBERS AND START-RESTART CODES.
C     UIN1,UIN2 = FILE UNIT NUMBER FOR INPUT DATA TYPE I,II.
C     UOUT = FILE UNIT NUMBER FOR OUTPUT DATA.
C     INCR = LOAD INCREMENT NUMBER FROM END OF WHICH RESTART IS MADE.
C     UINRS = INPUT RESTART TAPE UNIT NUMBER.
C     UOUTRS = OUTPUT RESTART TAPE UNIT NUMBER.
      INTEGER UIN1,UIN2,UOUT,INCR,UINRS,UOUTRS
      INTEGER START,STAR,REST
  101 FORMAT(A4,6X,6I5)
  201 FORMAT(1H1,'STARTING PROBLEM')
  202 FORMAT(1H1,'RESTARTING PROBLEM FROM END OF LOAD INCREMENT',I5)
      DATA STAR/'STAR'/,REST/'REST'/
      READ(5,101)START,UIN1,UIN2,UOUT,INCR,UINRS,UOUTRS
      IF(START.NE.STAR.AND.START.NE.REST)STOP 9999
      IF(UIN2.LE.0.OR.UOUT.LE.0)STOP 101
      IF(START.NE.STAR)GO TO 21
C     COLD START REQUESTED
      IF(UIN1.LE.0)STOP 101
      WRITE(6,201)
      UINRS = 0
      RETURN
C     RESTART REQUESTED
   21 IF(UINRS.LE.0)STOP 101
      WRITE(6,202)INCR
      RETURN
      END


      SUBROUTINE READO(UI,UO,ORD,PRED,MAXUP,MAXIT,ERRMAX, DFE,DFF,
     1MJUMP,JUMPR,SLOPED,FLAMAX,LAMIN)
C     READ PROBLEM IDENTIFICATION AND INCREMENTAL-ITERATIVE CONSTANTS.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR STRESS-STRAIN EXPANSION.
C     PRED = DEFAULT SOLUTION PREDICTOR ORDER.
C     MAXUP = MAXIMUM NUMBER OF JACOBIAN UPDATES PER LOAD STEP.
C     MAXIT = MAXIMUM NUMBER OF RESIDUAL-LOAD ITERATIONS PER UPDATE.
C     ERRMAX = MAXIMUM ALLOWABLE ERROR NORM.
C     DFE,DFF = FINITE DIFFERENCE STEP SIZES TO BE USED IN COMPUTING
C       STRESS-STRAIN TENSORS,FORCES.
C     MJUMP = NUMBER OF INCREMENT DIVISIONS TO PERFORM WHEN NEARING A
C       LIMIT POINT.
C     JUMPR = FRACTION OF LOAD INCREMENT PRECEDING LIMIT POINT
C       AT WHICH LIMIT IS TO BE TRAVERSED.
C     SLOPED = MAXIMUM SLOPE RATIO (CHANGE/AVERAGE) DURING LOAD STEP.
C     FLAMAX = MAXIMUM FRACTICN OF LOAD INCREMENT TO BE TAKEN DURING
C       NEGATIVE LOADING (AFTER MAXIMIM LIMIT POINT).
C     LAMIN = MINIMUM (NEGATIVE) FRACTION OF LOAD INCREMENT AT WHICH
C       ANALYSIS IS TERMINATED (AFTER MAXIMUM LIMIT POINT).
      INTEGER UI,UO,ORD,PRED,MAXUP,MAXIT,MJUMP
      DOUBLE PRECISION ERRMAX,DFE,DFF,JUMPR,SLOPED,FLAMAX,LAMIN
      INTEGER I,BLANK,IDENT(20)
 101  FORMAT(20A4)
 102  FORMAT(4I10,F10.0)
 103  FORMAT(2F10.0)
 104  FORMAT(I10,4F10.0)
 201  FORMAT(1H0,20A4)
 202  FORMAT(/1H ,'TENSOR ORDER =',I5/1H ,'PREDICTOR TYPE =',I5/
     11H ,'MAXIMUM JACOBIAN UPDATES PER STEP =',I5/
     21H ,'MAXIMUM RESIDUAL LOAD CORRECTIVE ITERATIONS =',I5/
     31H ,'MAXIMUM ERROR NORM =',E12.5)
 203  FORMAT(/1H ,'DFE =',E12.5/1H ,'DFF =',E12.5)
 204  FORMAT(/1H ,'MJUMP =',I5/1H ,'JUMPR =',E12.5/
     11H ,'SLOPED =',E12.5/1H ,'FLAMAX =',E12.5/1H ,'LAMIN =',E12.5)
      DATA BLANK/'    '/
      READ(UI,101)(IDENT(I),I=1,20)
      WRITE(UO,201)(IDENT(I),I=1,20)
      READ(UI,102)ORD,PRED,MAXUP,MAXIT,ERRMAX
      IF(ORD.EQ.0)ORD = 3
      IF(PRED.EQ.0)PRED = 2
      IF(MAXUP.EQ.0)MAXUP = 0
      IF(MAXIT.EQ.0)MAXIT = 5
      IF(ERRMAX.EQ.0.D0)ERRMAX = 1.D-8
      WRITE(UO,202)ORD,PRED,MAXUP,MAXIT,ERRMAX
      READ(UI,103)DFE,DFF
      IF(DFE.EQ.0.D0)DFE = 1.D-3
      IF(DFF.EQ.0.D0)DFF = 1.D-8
      WRITE(UO,203)DFE,DFF
      READ(UI,104)MJUMP,JUMPR,SLOPED,FLAMAX,LAMIN
      IF(MJUMP.EQ.0)MJUMP = 3
      IF(JUMPR.EQ.0.D0)JUMPR = 0.1D0
      IF(SLOPED.EQ.0.D0)SLOPED = 0.5D0
      IF(FLAMAX.EQ.0.D0)FLAMAX = 1.D0
      IF(LAMIN.EQ.0.D0)LAMIN = 0.D0
      WRITE(UO,204)MJUMP,JUMPR,SLOPED,FLAMAX,LAMIN
      RETURN
      END

      SUBROUTINE READ1(UI,UO,IPRESS,NF,THICK,NMAT,E,NU)
C     READ BASIC CODES AND CONSTANTS.
```

```
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     NF = NUMBER OF FREEDOMS PER NODE.
C     THICK = DEFAULT ELEMENT THICKNESS.
C     NMAT = NUMBER OF MATERIALS.
C     E,NU = MATERIAL CONSTANTS (E.G. ELASTIC MODULUS,POISSON'S RATIO).
      INTEGER UI,UO,IPRESS,NF,NMAT
      DOUBLE PRECISION THICK,E(1),NU(1)
      DOUBLE PRECISION C1,C2
      INTEGER M,N1
  101 FORMAT(2I10,F10.0)
  102 FORMAT(I10,2F10.0)
  201 FORMAT(1H0,'PRESSURE CODE =',I5/1H ,'DOF PER NODE =',I5
     1/1H ,'DEFAULT THICKNESS =',E12.5)
  202 FORMAT(/1H ,'MATERIAL    CONSTANT     CONSTANT')
  203 FORMAT(1H ,I5,5X,E12.5,1X,E12.5)
      READ(UI,101)IPRESS,NF,THICK
      IF(NF.EQ.0)NF = 3
      IF(THICK.EQ.0.D0)THICK = 1.D0
      WRITE(UO,201)IPRESS,NF,THICK
      WRITE(UO,202)
      NMAT = 0
   40 READ(UI,102)I,C1,C2
      IF(I.LE.0)RETURN
      WRITE(UO,203)I,C1,C2
      IF(I.LE.NMAT)GO TO 51
      N1 = NMAT+1
      DO 45 M=N1,I
      E(M) = 0.D0
   45 NU(M) = 0.D0
      NMAT = I
   51 E(I) = C1
      NU(I) = C2
      GO TO 40
      END

      SUBROUTINE READC(UI,UO,CCORDA)
C     READ SPECIAL CARTESIAN COORDINATE SYSTEMS.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
```

```
C     COORDA(I) = ANGLE FOR SPECIAL COORDINATE SYSTEM I.
      INTEGER UI,UO
      DOUBLE PRECISION COORDA(1)
      INTEGER I
      DOUBLE PRECISION ANGLE,F
  101 FORMAT(I10,F10.0)
  201 FORMAT(//1H1,'CARTESIAN COORDINATE SYSTEMS DEFINED'/
     11H ,'NUMBER   X-AXIS ANGLE')
  202 FORMAT(1H ,I5,5X,F10.4)
      F = 3.14159265358979300/180.D0
      WRITE(UO,201)
      DO 5 I=1,5
    5 COORDA(I) = 0.D0
   10 READ(UI,101)I,ANGLE
      IF(I.LE.0)RETURN
      WRITE(UO,202)I,ANGLE
      COORDA(I) = ANGLE*F
      GO TO 10
      END


      SUBROUTINE READM(UI,UO,NOD,NEL,COORDA,COORD,GCOS,
     1IMAT,THICK,T,ELNO,EGEOM)
C     READ MESH DATA.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     NOD,NEL = NUMBER OF NODES,ELEMENTS.
C     COORDA(I) = ANGLE FOR SPECIAL COORDINATE SYSTEM I.
C     COORD(J,I) = COORDINATES FOR NODE I.
C     GCOS(J,I) = DIRECTION COSINES FOR NODE I.
C     IMAT(I) = MATERIAL NUMBER FOR ELEMENT I.
C     THICK = DEFAULT THICKNESS.
C     T(I) = THICKNESS OF ELEMENT I.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRY FOR ELEMENT I (BASE,HEIGHT,PART BASE).
      INTEGER UI,UO,NOD,NEL,IMAT(1),ELNO(3,1)
      DOUBLE PRECISION COORDA(1),COORD(3,1),THICK,VLENTH,VDOT
      DOUBLE PRECISION GCOS(3,3,1),EGEOM(3,1),T(1)
      INTEGER I,J,M,LCOORD,DCOORD,N1,N2,N3
      DOUBLE PRECISION F,ANGLE,X,Y,Z,R,TT,B,B1,H,L31,A,V21(3),V31(3)
  101 FORMAT(2I5,3F10.0,I5)
```

```
102 FORMAT(2I5,F10.0,3I5)
201 FORMAT(1H1,'** NODE **'/1H ,' NO.  I.D.  LCOORD   X (R)',
   17X,'Y (THETA)         Z            DCOORD')
202 FORMAT(1H ,2I5,1X,I5,2X,E12.5,1X,E12.5,1X,E12.5,2X,I5)
203 FORMAT(1H1,'  ELEMENT '/
   11H ,' NO.  I.D.       MATERIAL     THICKNESS',4X,
   2'NODE 1     NODE 2     NODE 3        AREA')
204 FORMAT(1H ,2I5,5X,I5,5X,E12.4,3X,I5,5X,I5,5X,I5,5X,E12.4)
    F = 3.14159265358979300/180.D0
    WRITE(UO,201)
    NOD = 0
  6 READ(UI,101)I,LCOORD,X,Y,Z,DCOORD
    IF(I.LE.0)GO TO 150
    NOD = NOD+1
    WRITE(UO,202)NOD,I,LCOORD,X,Y,Z,DCOORD
    COORD(3,NOD) = Z
    IF(LCOORD.EQ.0)GO TO 7
    ANGLE = Y*F
    Y = X*DSIN(ANGLE)
    X = X*DCOS(ANGLE)
  7 COORD(1,NOD) = X
    COORD(2,NOD) = Y
    IF(DCOORD-1)13,12,11
 11 ANGLE = COORDA(DCOORD)
    GCOS(1,1,NOD) = DCOS(ANGLE)
    GCOS(1,2,NOD) = DSIN(ANGLE)
    GO TO 100
 12 R = DSQRT(X**2 + Y**2)
    IF(R.EQ.0.D0)GO TO 13
    GCOS(1,1,NOD) = X/R
    GCOS(1,2,NOD) = Y/R
    GO TO 100
 13 GCOS(1,1,NOD) = 1.D0
    GCOS(1,2,NOD) = 0.D0
100 GCOS(1,3,NOD) = 0.D0
    GCOS(2,1,NOD) = -GCOS(1,2,NOD)
    GCOS(2,2,NOD) = GCOS(1,1,NOD)
    GCOS(2,3,NOD) = 0.D0
    GCOS(3,1,NOD) = 0.D0
    GCOS(3,2,NOD) = 0.D0
```

```
      GCOS(3,3,NOD) = 1.D0
      GO TO 6
  150 WRITE(UO,203)
      NEL = 0
  151 READ(UI,102)I,M,TT,N1,N2,N3
      IF(I.LE.0)GO TO 250
      IF(TT.EQ.0.D0)TT = THICK
      NEL = NEL+1
      IMAT(NEL) = M
      T(NEL) = TT
      DO 170 J=1,3
      V21(J) = COORD(J,N2) - COORD(J,N1)
  170 V31(J) = COORD(J,N3) - COORD(J,N1)
      B = VLENTH(V21)
      L31 = VLENTH(V31)
      B1 = VDOT(V21,V31)/B
      H = DSQRT(L31**2 - B1**2)
      A = .5D0*B*H
      EGEOM(1,NEL) = B
      EGEOM(2,NEL) = H
      EGEOM(3,NEL) = B1
  200 WRITE(UO,204)NEL,I,M,TT,N1,N2,N3,A
      ELNO(1,NEL) = N1
      ELNO(2,NEL) = N2
      ELNO(3,NEL) = N3
      GO TO 151
  250 RETURN
      END


      SUBROUTINE READK(UI,UO,NOD,NF,KFD)
C     READ SPECIFIED FORCE-DISPLACEMENT-CONSTRAINT DEGREES OF FREEDOM.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     NOD = NUMBER OF NODES.
C     NF = NUMBER OF FREEDOMS PER NODE.
C     KFD(I) = FORDE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
      INTEGER UI,UO,NOD,NF,KFD(1)
      INTEGER ISTOR(4),JSTOR(4),KSTOR(4),LSTOR(4)
      INTEGER I,J,K,L,M,NN,JLOC,LLOC
  101 FORMAT(4(4I5))
  201 FORMAT(//1H1,'SPECIFIED FORCE-DISPLACEMENT-CONSTRAINT DOF'
     1/1H ,'NODE I.D.  COMPONENT  NODE I.D.  COMPONENT')
  202 FORMAT(1H ,I5,6X,I5,6X,I5,6X,I5)
      NN = NOD*NF
      WRITE(UO,201)
C     SET DEFAULT CODES TO SPECIFIED FORCE.
      DO 5 I=1,NN
    5 KFD(I) = I
   10 READ(UI,101)(ISTOR(K),JSTOR(K),KSTOR(K),LSTOR(K),K=1,4)
      DO 12 M=1,4
      IF(ISTOR(M).NE.0)GO TO 13
   12 CONTINUE
      GO TO 51
   13 DO 20 M=1,4
      I = ISTOR(M)
      IF(I.LE.0)GO TO 20
      J = JSTOR(M)
      K = KSTOR(M)
      L = LSTOR(M)
      WRITE(UO,202)I,J,K,L
      IF(K.EQ.0)K = I
      IF(L.EQ.0)L = -J
      JLOC = NF*(I-1) + J
      LLOC = L
      IF(L.LT.0)LLOC = -L
      LLOC = NF*(K-1) + LLOC
      IF(L.LT.0)LLOC = -LLOC
      KFD(JLOC) = LLOC
   20 CONTINUE
      GO TO 10
   51 RETURN
      END

      SUBROUTINE READP(UI,UO,NOD,NF,KFD,PREF)
C     READ LOAD REFERENCE CURVES.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     NOD = NUMBER OF NODES.
C     NF = NUMBER OF FREEDOMS PER NODE.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
```

```
C     PREF(J,I) = NODAL LOAD AT DOF I FOR LOAD REFERENCE VECTOR J.
      INTEGER UI,UO,NOD,NF,KFD(1)
      DOUBLE PRECISION PREF(2,1)
      INTEGER ISTOR(4),JSTOR(4),ILOAD,NN,I,J,K,L
      DOUBLE PRECISION STOR(4)
  101 FORMAT(I10)
  102 FORMAT(4(2I5,F10.0))
  201 FORMAT(1H1,'NO. OF LOAD REFERENCE CURVES = ',I5)
  202 FORMAT(1H0,'LOAD REFERENCE CURVE NO. ',I5/
     11H ,'  NODE   COMPONENT  LOAD')
  203 FORMAT(1H ,I5,3X,I5,E12.5)
      NN = NF*NOD
      READ(UI,101)NLOAD
      WRITE(UO,201)NLOAD
      DO 100 ILOAD=1,NLOAD
      WRITE(UO,202)ILOAD
      DO 5 I=1,NN
    5 PREF(ILOAD,I) = 0.D0
   11 READ(UI,102)(ISTOR(K),JSTOR(K),STOR(K),K=1,4)
      DO 12 K=1,4
      IF(ISTOR(K).NE.0)GO TO 13
   12 CONTINUE
      DO 15 I=1,NN
      J = -KFD(I)
      IF(J.GT.0.AND.J.NE.I)PREF(ILOAD,I) = PREF(ILOAD,J)
   15 CONTINUE
      GO TO 100
   13 DO 20 K=1,4
      I = ISTOR(K)
      IF(I.LE.0)GO TO 20
      J = JSTOR(K)
      WRITE(UO,203)I,J,STOR(K)
      L = NF*(I-1)+J
      PREF(ILOAD,L) = STOR(K)
   20 CONTINUE
      GO TO 11
  100 CONTINUE
      RETURN
      END
      SUBROUTINE READPR(UI,UO,NEL,PRREF)
C     READ PRESSURE LOAD REFERENCE CURVE.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     NEL = NUMBER OF ELEMENTS.
C     PRREF(I) = INTENSITY ON ELEMENT I FOR PRESSURE REFERENCE VECTOR.
      INTEGER UI,UO,NEL
      DOUBLE PRECISION PRREF(1)
      INTEGER ISTOR(4),ILOAD,I,K
      DOUBLE PRECISION STOR(4)
  101 FORMAT(I10)
  102 FORMAT(4(I10,F10.0))
  201 FORMAT(1H1,'NO. OF PRESSURE LOAD REFERENCE CURVES =',I5)
  202 FORMAT(1H0,'PRESSURE LOAD REFERENCE CURVE NO.',I5/
     11H ,'ELEMENT   PRESSURE')
  203 FORMAT(1H ,I5,3X,E12.5)
      READ(UI,101)NLOAD
      WRITE(UO,201)NLOAD
      DO 100 ILOAD=1,NLOAD
      WRITE(UO,202)ILOAD
      DO 5 I=1,NEL
    5 PRREF(I) = 0.D0
   11 READ(UI,102)(ISTOR(K),STOR(K),K=1,4)
      DO 12 K=1,4
      IF(ISTOR(K).NE.0)GO TO 13
   12 CONTINUE
      GO TO 100
   13 DO 20 K=1,4
      I = ISTOR(K)
      IF(I.LE.0)GO TO 20
      WRITE(UO,203)I,STOR(K)
      PRREF(I) = STOR(K)
   20 CONTINUE
      GO TO 11
  100 CONTINUE
      RETURN
      END

      SUBROUTINE READI(UI,UO,NINCR,PRED,IPRED,PFACT,PRFACT)
```

```
C     READ INCREMENTAL LOAD DATA.
C     UI,UO = INPUT,OUTPUT FILE UNIT NUMBERS.
C     NINCR = NUMBER OF LOAD INCREMENTS.
C     PRED = DEFAULT SOLUTION PREDICTOR ORDER.
C     IPRED(I) = SOLUTION PREDICTOR ORDER FOR LOAD INCREMENT I.
C     PFACT(J,I) = NODAL LOAD FACTOR FOR INCR. I AND REFERENCE VECTOR J.
C     PRFACT(I) = ELEMENT PRESSURE INTENSITY FACTOR FOR LOAD INCR. I.
      INTEGER UI,UO,NINCR,PRED,IPRED(1)
      DOUBLE PRECISION PFACT(2,1),PRFACT(1)
  101 FORMAT(I10)
  102 FORMAT(I10,3F10.0)
  201 FORMAT(///1H1,'NO. OF LOAD INCREMENTS = ',I5/
     11H ,'INCREMENT',2X,'PREDICTOR',2X,'MECHANICAL CURVE FACTORS',
     16X,'PRESSURE')
  202 FORMAT(1H ,I5,6X,I5,6X,E12.5,2X,E12.5,2X,E12.5)
      READ(UI,101)NINCR
      WRITE(UO,201)NINCR
      DO 100 INCR=1,NINCR
      READ(UI,102)IPRED(INCR),PFACT(1,INCR),PFACT(2,INCR),PRFACT(INCR)
      IF(IPRED(INCR).EQ.0)IPRED(INCR) = PRED
      WRITE(UO,202)INCR,IPRED(INCR),PFACT(1,INCR),PFACT(2,INCR),
     1PRFACT(INCR)
  100 CONTINUE
      RETURN
      END


      SUBROUTINE HEAD(UO,INCR,ISTEP,LAMS,LAM,F1,F2,FP,
     1PRED,MAXUP,NUP,MAXIT,ITER,ERRMAX,ERR)
C     WRITE HEADING FOR LOAD INCREMENT STEP.
C     UO = OUTPUT FILE UNIT NUMBER.
C     INCR = LOAD INCREMENT NUMBER.
C     ISTEP = LOAD STEP NUMBER.
C     LAMS = LOAD STEP PARAMETER = FRACTION OF REMAINING LOAD INCREMENT.
C     LAM = INCREMENTAL LOAD PARAMETER (MAXIMUM VALUE 1.0).
C     F1,F2 = NODAL LOAD FACTORS APPLIED TO REFERENCE VECTORS.
C     FP = ELEMENT PRESSURE LOAD FACTOR APPLIED TO REFERENCE VECTOR.
C     PRED = SOLUTION PREDICTOR ORDER.
C     MAXUP = SPECIFIED MAXIMUM NUMBER OF JACOBIAN STIFFNESS UPDATES.
C     NUP = NUMBER OF JACOBIAN UPDATES PERFORMED DURING THIS LOAD STEP.
```

```
C     MAXIT = MAXIMUM NUMBER OF RESIDUAL-FORCE ITERATIONS PER UPDATE.
C     ITER = NUMBER OF ITERATIONS PERFORMED SINCE LAST UPDATE.
C     ERRMAX = SPECIFIED MAXIMUM RESIDUAL-FORCE ERROR NORM.
C     ERR = ACTUAL ERROR NORM OBTAINED.
      INTEGER UO,INCR,ISTEP,PRED,MAXUP,NUP,MAXIT,ITER
      DOUBLE PRECISION LAMS,LAM,F1,F2,FP,ERRMAX,ERR
  201 FORMAT(1H1/////1H ,'L O A D  I N C R E M E N T',I5,
     1',  L O A D  S T E P',I5,
     2//1H ,'INCREMENT LOAD PARAMETER =',E12.5,
     3',  STEP LOAD PARAMETER =',E12.5)
  202 FORMAT(//1H ,'MECHANICAL LOAD FACTORS =',2E14.5)
  207 FORMAT(1H ,'PRESSURE LOAD FACTOR =',E14.5)
  203 FORMAT(1H ,'PREDICTOR TYPE =',I5)
  204 FORMAT(1H ,'SPECIFIED MAX. NO. JACOBIAN UPDATES =',I5,
     1', NO. UPDATES PERFORMED =',I5)
  205 FORMAT(1H ,'SPECIFIED MAX. NO. ITERATIONS PER UPDATE =',I5,
     1', NO. ITERATIONS PERFORMED SINCE LAST UPDATE =',I5)
  206 FORMAT(1H ,'SPECIFIED MAX. RESIDUAL FORCE ERROR =',E12.4,
     1', ACTUAL ERROR =',E12.4)
      WRITE(UO,201)INCR,ISTEP,LAM,LAMS
      WRITE(UO,202)F1,F2
      WRITE(UO,207)FP
      WRITE(UO,203)PRED
      WRITE(UO,204)MAXUP,NUP
      WRITE(UO,205)MAXIT,ITER
      WRITE(UO,206)ERRMAX,ERR
      RETURN
      END

      SUBROUTINE OUTLIM(UO,NOD,NEL,NN,NF,ELNO,EGEOM,EET,CQ,P,Q,DFF,
     1RL,RRL,RQ,RRQ,LAM,LAMR)
C     OUTPUT LIMIT POINT DATA.
C     UO = OUTPUT FILE UNIT NUMBER.
C     NOD,NEL = NUMBER OF NODES,ELEMENTS.
C     NN,NF = SYSTEM DOF,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     EET(J,I) = PREDICTED LIMIT STRAINS FOR ELEMENT I.
C     CQ(I) = CUMULATIVE NODAL DISPLACEMENT FOR DOF I.
```

```
C     P,Q = TEMPORARY STORAGE VECTORS.
C     DFF = FINITE-DIFFERENCE SPACING USED IN COMPUTING FORCES.
C     RL,RRL = 1ST,2ND ORDER LOAD PARAMETER RATES.
C     RQ(I),RRQ(I) = 1ST,2ND ORDER DISPLACEMENT RATES FOR DOF I.
C     LAM = INCREMENTAL LOAD PARAMETER (MAXIMUM VALUE 1.0).
C     LAMR = LOAD YET TO BE APPLIED = 1.0 - LAM.
      INTEGER UO,NOD,NEL,NN,NF,ELNO(3,1)
      DOUBLE PRECISION EGEOM(3,1),EET(3,1),QQ(1),P(1),Q(1),DFF
      DOUBLE PRECISION RL,RRL,RQ(1),RRQ(1),LAM,LAMR
      DOUBLE PRECISION SLIM,LAMLIM
  201 FORMAT(//1H ,'PREDICTED LIMIT POINT OCCURS AT LOAD INCREMENT PARAM
     1ETER =',E12.5/1H ,
     2'THE FOLLOWING ARE PREDICTED LIMIT FORCES-DISPLACEMENTS-STRAINS')
      SLIM = -RL/RRL
      LAMLIM = LAM + (SLIM*RL + .5D0*SLIM**2*RRL)*LAMR
      WRITE(UO,201)LAMLIM
      DO 10 I=1,NN
   10 Q(I) = QQ(I) + SLIM*RQ(I) + .5D0*SLIM**2*RRQ(I)
      CALL FORCE(NEL,NN,NF,ELNO,DFF,Q,P)
      CALL OUTPQ(UO,NOD,NF,P,Q)
      CALL STRAIN(NEL,ELNO,EGEOM,NF,Q,EET)
      CALL OUTE(UO,NEL,EET)
      RETURN
      END


      SUBROUTINE OUTPQ(UO,NOD,NF,P,Q)
C     WRITE TOTAL FORCES AND DISPLACEMENTS.
C     UO = OUTPUT FILE UNIT NUMBER.
C     NOD,NF = NUMBER OF NODES,DOF PER NODE.
C     P(I),Q(I) = FORCE,DISPLACEMENT TO BE OUTPUT AT DOF I.
      INTEGER UO,NOD,NF
      DOUBLE PRECISION P(1),Q(1)
      INTEGER I,J,K
      DOUBLE PRECISION STOR(6)
  201 FORMAT(1H1,14X,22(1H*),'  CUMULATIVE INTERNAL FORCES AND DISPLACEM
     1ENTS',2X,23(1H*)/1H ,'** NODE **',4X,17(1H*),'  FORCES  ',16(1H*),
     27X,13(1H*),'  DISPLACEMENTS  ',13(1H*),
     3/1H ,10H NO.  I.D.,5X,1HU,14X,1HV,14X,1HW,19X,1HU,14X,1HV,14X,1HW/
     1)
  202 FORMAT(1H ,2I5,2X,3E15.7,5X,3E15.7)
      WRITE(UO,201)
      DO 10 I=1,6
   10 STOR(I) = 0.D0
      DO 100 I=1,NOD
      DO 50 J=1,NF
      K = NF*(I-1)+J
      STOR(J) = P(K)
   50 STOR(3+J) = Q(K)
  100 WRITE(UO,202)I,I,(STOR(J),J=1,6)
      RETURN
      END


      SUBROUTINE OUTE(UO,NEL,ET)
C     WRITE CUMULATIVE STRAINS ET.
C     UO = OUTPUT FILE UNIT NUMBER.
C     NEL = NUMBER OF ELEMENTS.
C     ET(J,I) = STRAINS TO BE OUTPUT FOR ELEMENT I.
      INTEGER UO,NEL
      DOUBLE PRECISION ET(3,1)
      INTEGER II,I
  201 FORMAT(1H1,'ELEMENT',3X,12(1H*),'  CUMULATIVE STRAINS  ',12(1H*),
     1/1H ,'   NO.',10X,2HXX,10X,2HYY,10X,2HZZ,10X,2HXY)
  202 FORMAT(1H ,I5,3X,2E12.4,12X,E12.4)
      WRITE(UO,201)
      DO 100 II=1,NEL
  100 WRITE(UO,202)II,(ET(I,II),I=1,3)
      RETURN
      END


      SUBROUTINE QFILL(NF,ELNO,QQ,Q)
C     FORM VECTOR OF ELEMENT DISPLACEMENTS Q FROM NODAL DISPLACEMENTS QQ
C     NF = DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     QQ(I) = NODAL DISPLACEMENT AT DOF I.
C     Q(I) = ELEMENT NODAL DISPLACEMENT AT ELEMENT DOF I.
      INTEGER NF,ELNO(1)
      DOUBLE PRECISION QQ(1),Q(1)
      INTEGER I,NI,IO,JO,K
      DO 100 I=1,3
      NI = ELNO(I)
      IO = NF*(NI-1)
      JO = NF*(I-1)
      DO 100 K=1,NF
  100 Q(JO+K) = QQ(IO+K)
      CALL ROTQ(NF,ELNO,Q,0)
      RETURN
      END


      SUBROUTINE PFILL(NF,ELNO,P,PP)
C     FORM VECTOR OF NODAL FORCES PP FROM ELEMENT FORCES P.
C     NF = DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     P(I) = ELEMENT NODAL FORCE AT ELEMENT DOF I.
C     PP(I) = NODAL FORCE AT DOF I.
      INTEGER NF,ELNO(1)
      DOUBLE PRECISION P(1),PP(1)
      INTEGER I,NI,IO,JO,K
      CALL ROTQ(NF,ELNO,P,1)
      DO 100 I=1,3
      NI = ELNO(I)
      IO = NF*(NI-1)
      JO = NF*(I-1)
      DO 100 K=1,NF
  100 PP(IO+K) = PP(IO+K) + P(JO+K)
      RETURN
      END


      SUBROUTINE DFILL(NF,EGEOM,Q,D)
C     COMPUTE ELEMENT DISPLACEMENT DERIVATIVES D FROM DISPLACEMENTS Q.
C     NF = DOF PER NODE.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     Q(I) = ELEMENT NODAL DISPLACEMENT AT ELEMENT DOF I.
C     D(I) = ITH DISPLACEMENT DERIVATIVE (UX,VX,WX,UY,VY,WY).
      INTEGER NF
      DOUBLE PRECISION EGEOM(1),Q(1),D(1)
      INTEGER NF2,I
      DOUBLE PRECISION B,H,B1,B2,A2
      NF2 = NF*2
      B = EGEOM(1)
      H = EGEOM(2)
      B1 = EGEOM(3)
      B2 = B-B1
      A2 = 1.D0/(B*H)
      DO 10 I=1,NF
      D(I) = A2*H*(-Q(I)+Q(NF+I))
   10 D(NF+I) = A2*(-B2*Q(I)-B1*Q(NF+I)+B*Q(NF2+I))
      RETURN
      END


      SUBROUTINE EFILL(NF,D,ET)
C     COMPUTE ELEMENT STRAINS ET FROM DISPLACEMENT DERIVATIVES D.
C     NF = DOF PER NODE.
C     D(I) = ITH DISPLACEMENT DERIVATIVE (UX,VX,WX,UY,VY,WY).
C     ET(I) = ITH LAGRANGIAN STRAIN COMPONENT (XX,YY,XY).
      INTEGER NF
      DOUBLE PRECISION D(1),ET(1)
      ET(1) = D(1)
      ET(2) = D(NF+2)
      ET(3) = D(2) + D(NF+1)
      DO 10 I=1,NF
      ET(1) = ET(1) + .5D0*D(I)**2
      ET(2) = ET(2) + .5D0*D(NF+I)**2
   10 ET(3) = ET(3) + D(I)*D(NF+I)
      RETURN
      END


      SUBROUTINE AFILL(NF,D,A,KODE)
C     KODE=0 FORM LAGRANGIAN A1(I,J) = A1(I,J,K)*D(K).
C     KODE=1 FORM LAGRANGIAN A(I,J) = A0(I,J) + A1(I,J,K)*D(K).
C     NF = DOF PER NODE.
C     D(I) = ITH DISPLACEMENT DERIVATIVE (UX,VX,WX,UY,VY,WY).
C     A GIVES LAGRANGIAN STRAINS (XX,YY,XY) FROM (UX,VX,WX,UY,VY,WY).
      INTEGER NF,KODE
      DOUBLE PRECISION D(1),A(3,1)
      INTEGER J,J1
      DO 10 J=1,NF
      J1 = NF+J
      A(1,J) = D(J)
      A(1,J1) = 0.D0
      A(2,J) = 0.D0
      A(2,J1) = D(J1)
      A(3,J) = D(J1)
   10 A(3,J1) = D(J)
      IF(KODE.EQ.0)RETURN
      A(1,1) = 1.D0 + A(1,1)
      A(2,NF+2) = 1.D0 + A(2,NF+2)
      A(3,2) = 1.D0 + A(3,2)
      A(3,NF+1) = 1.D0 + A(3,NF+1)
      RETURN
      END


      SUBROUTINE GFILL(NF,EGEOM,G)
C     FORM DISPLACEMENT-DERIVATIVES FROM NODAL-DISPLACEMENTS MATRIX G.
C     NF = DOF PER NODE.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     G GIVES (UX,VX,WX,UY,VY,WY) FROM ELEMENT NODAL DISPLACEMENTS.
      INTEGER NF
      DOUBLE PRECISION EGEOM(1),G(6,1)
      INTEGER NF2,NF3,I,I1,I2
      DOUBLE PRECISION B,H,B1,B2,A2
      NF2 = NF*2
      NF3 = NF*3
      B = EGEOM(1)
      H = EGEOM(2)
      B1 = EGEOM(3)
      B2 = B-B1
      A2 = 1.D0/(B*H)
      DO 5 I=1,NF2
      DO 5 J=1,NF3
    5 G(I,J) = 0.D0
      DO 10 I=1,NF
      I1 = NF+I
      I2 = NF2+I
      G(I,I) = -H*A2
      G(I,I1) = H*A2
      G(I1,I) = -B2*A2
      G(I1,I1) = -B1*A2
   10 G(I1,I2) = B*A2
      RETURN
      END


      SUBROUTINE MTRAN(A,MA,NA,B,MB,NB,D)
C     COMPUTE D(I,J) = A(M,N)*B(M,I)*B(N,J).
C     A = SQUARE MATRIX TO BE TRANSFORMED.
C     MA = MAXIMUM (FORTRAN-DIMENSIONED) SIZE OF A.
C     NA = ACTUAL SIZE OF A.
C     B = TRANSFORMATION MATRIX.
C     D = SQUARE TRANSFORMED MATRIX.
C     MB = MAXIMUM (FORTRAN-DIMENSIONED) SIZE OF D.
C     NB = ACTUAL SIZE OF D.
      INTEGER MA,NA,MB,NB
      DOUBLE PRECISION A(MA,1),B(MA,1),D(MB,1)
      INTEGER I,J,M
      DOUBLE PRECISION C,STOR(6,9)
      DO 50 I=1,NA
      DO 50 J=1,NB
      C = 0.D0
      DO 45 M=1,NA
   45 C = C + A(I,M)*B(M,J)
   50 STOR(I,J) = C
      DO 100 I=1,NB
      DO 100 J=1,NB
      C = 0.D0
      DO 95 M=1,NA
   95 C = C + STOR(M,J)*B(M,I)
  100 D(I,J) = C
      RETURN
      END


      SUBROUTINE ROTQ(NF,ELNC,Q,KODE)
```

```
C     KODE=0 ROTATE DISPLACEMENTS TO ELEMENT FROM NODAL.
C     KODE=1 ROTATE FORCES TO NODAL FROM ELEMENT.
C     NF = DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     Q(I) = ELEMENT NODAL DISPLACEMENT AT ELEMENT DOF I.
C     GCOS(J,I) = DIRECTION COSINES FOR COORDINATE SYSTEM AT NODE I.
C     COORD(J,I) = COORDINATES OF NODE I.
      COMMON/COMCOS/GCOS/COMCOR/COORD
      INTEGER NF,ELNO(1),KODE
      DOUBLE PRECISION GCOS(3,3,1),COORD(3,1),Q(1)
      INTEGER N1,N2,N3,NI,N,I,J,M,IO
      DOUBLE PRECISION V21(3),V31(3),VY(3),VZ(3)
      DOUBLE PRECISION C,REG(3,3),RNG(3,3),REN(3,3),QPART(3)
C     COMPUTE MATRIX REG TO ROTATE DISPLACEMENTS ELEMENT FROM GLOBAL
      N1 = ELNO(1)
      N2 = ELNO(2)
      N3 = ELNO(3)
      DO 10 I=1,3
      V21(I) = COORD(I,N2) - COORD(I,N1)
   10 V31(I) = COORD(I,N3) - COORD(I,N1)
      CALL VCROSS(V21,V31,VZ)
      CALL VNORM(V21,V21)
      CALL VNORM(VZ,VZ)
      CALL VCROSS(VZ,V21,VY)
      DO 20 J=1,NF
      REG(1,J) = V21(J)
      REG(2,J) = VY(J)
   20 REG(3,J) = VZ(J)
      DO 500 N=1,3
C     COMPUTE MATRIX RNG ROTATE NODE N DISPLACEMENTS NODAL FROM GLOBAL
      NI = ELNO(N)
      DO 30 I=1,NF
      DO 30 J=1,NF
   30 RNG(I,J) = GCOS(I,J,NI)
C     COMPUTE MATRIX REN ROTATE DISPLACEMENTS ELEMENT FROM NODAL
      DO 50 I=1,NF
      DO 50 J=1,NF
      C = 0.D0
      DO 45 M=1,NF
   45 C = C + REG(I,M)*RNG(J,M)
   50 REN(I,J) = C
C     ROTATE Q(N) PARTITION USING MATRIX REN FOR NODE N
      IO = NF*(N-1)
      DO 110 I=1,NF
  110 QPART(I) = Q(IO+I)
      DO 120 I=1,NF
      C = 0.D0
      IF(KODE.EQ.1)GO TO 116
      DO 115 M=1,NF
  115 C = C + REN(I,M)*QPART(M)
      GO TO 120
  116 DO 118 M=1,NF
  118 C = C + REN(M,I)*QPART(M)
  120 Q(IO+I) = C
  500 CONTINUE
      RETURN
      END

      SUBROUTINE ROTK(NF,ELNO,K)
C     ROTATE ELEMENT STIFFNESS TO NODAL FROM ELEMENT.
C     NF = DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     K = ELEMENT STIFFNESS MATRIX.
C     GCOS(J,I) = DIRECTION COSINES FOR COORDINATE SYSTEM AT NODE I.
C     COORD(J,I) = COORDINATES OF NODE I.
      COMMON/COMCOS/GCOS/COMCOR/COORD
      INTEGER NF,ELNO(1)
      DOUBLE PRECISION GCOS(3,3,1),COORD(3,1),K(9,9)
      INTEGER N1,N2,N3,NI,N,I,J,M,IO,JO,IP,JP
      DOUBLE PRECISION V21(3),V31(3),VY(3),VZ(3)
      DOUBLE PRECISION C,REG(3,3),RNG(3,3),REN(3,3),KPART(3,3)
C     COMPUTE MATRIX REG TO ROTATE DISPLACEMENTS ELEMENT FROM GLOBAL
      N1 = ELNO(1)
      N2 = ELNO(2)
      N3 = ELNO(3)
      DO 10 I=1,3
      V21(I) = COORD(I,N2) - COORD(I,N1)
   10 V31(I) = COORD(I,N3) - COORD(I,N1)
      CALL VCROSS(V21,V31,VZ)
      CALL VNORM(V21,V21)
      CALL VNORM(VZ,VZ)
      CALL VCROSS(VZ,V21,VY)
      DO 20 J=1,NF
      REG(1,J) = V21(J)
      REG(2,J) = VY(J)
   20 REG(3,J) = VZ(J)
      DO 500 N=1,3
C     COMPUTE MATRIX RNG ROTATE NODE N DISPLACEMENTS NODAL FROM GLOBAL
      NI = ELNO(N)
      DO 30 I=1,NF
      DO 30 J=1,NF
   30 RNG(I,J) = GCOS(I,J,NI)
C     COMPUTE MATRIX REN ROTATE DISPLACEMENTS ELEMENT FROM NODAL
      DO 50 I=1,NF
      DO 50 J=1,NF
      C = 0.D0
      DO 45 M=1,NF
   45 C = C + REG(I,M)*RNG(J,M)
   50 REN(I,J) = C
      DO 200 IP=1,3
C     ROTATE K(IP,N) PARTITION USING MATRIX REN FOR NODE N
      IO = NF*(IP-1)
      JO = NF*(N-1)
      DO 110 I=1,NF
      DO 110 J=1,NF
  110 KPART(I,J) = K(IO+I,JO+J)
      DO 120 I=1,NF
      DO 120 J=1,NF
      C = 0.D0
      DO 115 M=1,NF
  115 C = C + KPART(I,M)*REN(M,J)
  120 K(IO+I,JO+J) = C
  200 CONTINUE
      DO 300 JP=1,3
C     ROTATE K(N,JP) PARTITION USING MATRIX REN FOR NODE N
      IO = NF*(N-1)
      JO = NF*(JP-1)
      DO 210 I=1,NF
      DO 210 J=1,NF
  210 KPART(I,J) = K(IO+I,JO+J)
      DO 220 I=1,NF
      DO 220 J=1,NF
      C = 0.D0
      DO 215 M=1,NF
  215 C = C + REN(M,I)*KPART(M,J)
  220 K(IO+I,JO+J) = C
  300 CONTINUE
  500 CONTINUE
      RETURN
      END


      SUBROUTINE FORCE(NEL,NN,NF,ELNO,DF,QQ,PP)
C     COMPUTE INTERNAL FORCES PP CORRESPONDING TO DISPLACEMENTS QQ.
C     NEL = NUMBER OF ELEMENTS.
C     NN,NF = SYSTEM DOF,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     DF = FINITE-DIFFERENCE SPACING USED IN COMPUTING FORCES.
C     QQ(I),PP(I) = CUMULATIVE DISPLACEMENT,INTERNAL FORCE AT DOF I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     NS = NUMBER OF STRAIN COMPONENTS.
      COMMON/COMEG/EGEOM/COMT/T/COMNS/NS
      INTEGER NEL,NN,NF,ELNO(3,1),NS
      DOUBLE PRECISION QQ(1),PP(1),DF,EGEOM(3,1),T(1)
      INTEGER I,II,M,NF2,NF3
      DOUBLE PRECISION C,V,UA,UB,E0(3),E(3),S(3),D(6),PD(6),Q(9),P(9)
      DOUBLE PRECISION A(3,6),G(6,9),ENERGY
      EQUIVALENCE(A(1),G(1))
      NF2 = NF*2
      NF3 = NF*3
      DO 10 I=1,NN
   10 PP(I) = 0.D0
      DO 100 II=1,NEL
      V = .5D0*T(II)*EGEOM(1,II)*EGEOM(2,II)
      CALL QFILL(NF,ELNO(1,II),QQ,Q)
      CALL DFILL(NF,EGEOM(1,II),Q,D)
      CALL EFILL(NF,D,E0)
      DO 50 I=1,NS
      DO 35 M=1,NS
   35 E(M) = EO(M)
      C = EO(I)
      E(I) = C - DF
      UA = ENERGY(II,E)
      E(I) = C + DF
      UB = ENERGY(II,E)
   50 S(I) = (UB-UA)/(2.D0*DF)*V
      CALL AFILL(NF,D,A,1)
      DO 70 I=1,NF2
      C = 0.D0
      DO 65 M=1,NS
   65 C = C + A(M,I)*S(M)
   70 PD(I) = C
      CALL GFILL(NF,EGEOM(1,II),G)
      DO 80 I=1,NF3
      C = 0.D0
      DO 75 M=1,NF2
   75 C = C + PD(M)*G(M,I)
   80 P(I) = C
  100 CALL PFILL(NF,ELNO(1,II),P,PP)
      RETURN
      END


      SUBROUTINE PFORCE(PFACT,PREF,NN,P)
C     COMPUTE APPLIED NODAL FORCES P.
C     PFACT(J) = NODAL LOAD FACTOR FOR REFERENCE VECTOR J.
C     PREF(J,I) = NODAL LOAD AT DOF I FOR REFERENCE VECTOR J.
C     NN = TOTAL SYSTEM DOF.
C     P(I) = APPLIED CUMULATIVE LOAD AT DOF I.
      INTEGER NN
      DOUBLE PRECISION PFACT(1),PREF(2,1),P(1)
      INTEGER I
      DO 100 I=1,NN
  100 P(I) = PFACT(1)*PREF(1,I) + PFACT(2)*PREF(2,I)
      RETURN
      END
      SUBROUTINE EFORCE(IPRESS,PR,PRREF,QQ,NEL,NN,NF,ELNC,PP)
C     COMPUTE NODAL FORCES PP DUE TO ELEMENT PRESSURES PR.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     PR = ELEMENT PRESSURE INTENSITY FACTOR.
C     PRREF(I) = INTENSITY ON ELEMENT I FOR PRESSURE REFERENCE VECTOR.
C     QQ(I) = CUMULATIVE NODAL DISPLACEMENT AT DOF I.
C     NEL = NUMBER OF ELEMENTS.
C     NN,NF = TOTAL SYSTEM DOF,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     PP(I) = COMPUTED NONCONSERVATIVE PRESSURE NODAL FORCE AT DOF I.
C     EGEOM(J,I) = GEOMETRY FOR ELEMENT I (BASE,HEIGHT,PART BASE).
      COMMON/COMEG/EGEOM
      INTEGER IPRESS,NEL,NN,NF,ELNO(3,1)
      DOUBLE PRECISION PR,PRREF(1),QQ(1),PP(1),EGEOM(3,1)
      INTEGER I,II
      DOUBLE PRECISION C,V21(3),V31(3),Q(9),P(9)
      DO 10 I=1,NN
   10 PP(I) = 0.D0
      IF(IPRESS.EQ.0)RETURN
      DO 100 II=1,NEL
      CALL QFILL(NF,ELNO(1,II),QQ,Q)
      DO 20 I=1,3
      V21(I) = Q(3+I) - Q(I)
   20 V31(I) = Q(6+I) - Q(I)
      V21(1) = V21(1) + EGEOM(1,II)
      V31(1) = V31(1) + EGEOM(3,II)
      V31(2) = V31(2) + EGEOM(2,II)
      CALL VCROSS(V21,V31,P)
      C = PR*PRREF(II)/6.D0
      DO 50 I=1,3
      P(I) = P(I)*C
      P(3+I) = P(I)
   50 P(6+I) = P(I)
  100 CALL PFILL(NF,ELNO(1,II),P,PP)
      RETURN
      END

      SUBROUTINE ERCOMP(UO,NN,KFD,PP,P,ERR)
C     COMPUTE ERROR NORM USING CUMULATIVE FORCES PP AND RESIDUALS P.
```

```
C     UO = OUTPUT FILE UNIT NUMBER.
C     NN = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     PP(I) = CUMULATIVE NODAL FORCE AT DOF I.
C     P(I) = RESIDUAL (UNBALANCED) NODAL FORCE AT DOF I.
C     ERR = COMPUTED RESIDUAL-FORCE ERROR NORM.
      INTEGER UO,NN,KFD(1)
      DOUBLE PRECISION PP(1),P(1),ERR
      INTEGER I,J
      DOUBLE PRECISION C1,C2
  201 FORMAT(1H ,'ERROR NORM = ',E12.5)
      ERR = 0.D0
      C1 = 0.D0
      C2 = 0.D0
      DO 5 I=1,NN
      J = KFD(I)
      IF(J.LT.0.OR.J.EQ.I)GO TO 5
      P(J) = P(J) + P(I)
      P(I) = 0.D0
    5 CONTINUE
      DO 10 I=1,NN
      IF(KFD(I).EQ.I)C1 = C1 + DABS(P(I))
   10 C2 = C2 + DABS(PP(I))
      IF(C2.GT.0.D0)ERR = C1/C2
      WRITE(UO,201)ERR
      RETURN
      END


      SUBROUTINE STRAIN(NEL,ELNO,EGEOM,NF,QQ,EET)
C     COMPUTE STRAINS EET FROM GLOBAL DISPLACEMENTS QQ.
C     NEL = NUMBER OF ELEMENTS.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     NF = DOF PER NODE.
C     QQ(I) = CUMULATIVE NODAL DISPLACEMENT AT DOF I.
C     EET(J,I) = COMPUTED CUMULATIVE STRAINS (XX,YY,XY) FOR ELEMENT I.
      INTEGER NEL,ELNO(3,1),NF
      DOUBLE PRECISION EGEOM(3,1),QQ(1),EET(3,1)
      INTEGER II
      DOUBLE PRECISION Q(9),D(6)
      DO 10 II=1,NEL
      CALL QFILL(NF,ELNO(1,II),QQ,Q)
      CALL DFILL(NF,EGEOM(1,II),Q,D)
   10 CALL EFILL(NF,D,EET(1,II))
      RETURN
      END


      DOUBLE PRECISION FUNCTION ENERGY(II,ET)
C     EVALUATE ENERGY DENSITY FOR ELEMENT II AT STRAINS ET (MOONEY).
C     II = ELEMENT NUMBER.
C     ET(I) = CUMULATIVE STRAINS (XX,YY,XY) FOR ELEMENT.
C     IMAT(I) = MATERIAL NUMBER FOR ELEMENT I.
C     CC1(I),CC2(I) = MATERIAL PROPERTIES FOR MATERIAL I.
      COMMON/COMMAT/IMAT/COMENU/CC1,CC2
      INTEGER II,IMAT(1)
      DOUBLE PRECISION ET(1),CC1(5),CC2(5)
      INTEGER I
      DOUBLE PRECISION C1,C2,A,B,D,I1,I2
      I = IMAT(II)
      C1 = CC1(I)
      C2 = CC2(I)
      A = 2.D0*(ET(1)+ET(2))
      B = 4.D0*ET(1)*ET(2) - ET(3)**2
      D = 1.D0 + A + B
      I1 = (A*(A+B)-B)/D
      I2 = I1 + B*(A+B)/D
      ENERGY = C1*I1 + C2*I2
      RETURN
      END


      SUBROUTINE EVAL(II,ORD,N,F0,DF,ESTOR)
C     EVALUATE STRAIN ENERGY OF ELEMENT II AS FUNCTION OF STRAINS TO
C     ESTABLISH A COMPLETE INTERPOLATING POLYNOMIAL OF ORDER ORD L.E. 3.
C     II = ELEMENT NUMBER.
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR DIFFERENCE EXPANSION.
C     N = DIMENSION OF TENSORS.
C     F0(I) = CURRENT VALUE OF INDEPENDENT VARIABLE I.
```

```
C     DF(I) = FINITE DIFFERENCE IN INDEPENDENT VARIABLE I.
C     ESTOR(I) = STORAGE VECTOR FOR ENERGY EVALUATIONS I.
C     ENERGY(II,F) GIVES ENERGY FOR ELEMENT II AT VARIABLE STATE F.
      INTEGER II,ORD,N
      DOUBLE PRECISION FO(1),DF(1),ESTOR(1),ENERGY
      INTEGER I,J,K,M,IE
      DOUBLE PRECISION F(3)
      IE = 1
      ESTOR(IE) = ENERGY(II,FO)
      IF(ORD.LT.1)GO TO 100
      DO 10 I=1,N
      IE = IE+1
      DO 5 M=1,N
    5 F(M) = FO(M)
      F(I) = F(I) + DF(I)
   10 ESTOR(IE) = ENERGY(II,F)
      IF(ORD.LT.2)GO TO 100
      DO 20 I=1,N
      DO 20 J=1,I
      IE = IE+1
      DO 15 M=1,N
   15 F(M) = FO(M)
      F(I) = F(I) + DF(I)
      F(J) = F(J) + DF(J)
   20 ESTOR(IE) = ENERGY(II,F)
      IF(ORD.LT.3)GO TO 100
      DO 30 I=1,N
      DO 30 J=1,I
      DO 30 K=1,J
      IE = IE+1
      DO 25 M=1,N
   25 F(M) = FO(M)
      F(I) = F(I) + DF(I)
      F(J) = F(J) + DF(J)
      F(K) = F(K) + DF(K)
   30 ESTOR(IE) = ENERGY(II,F)
  100 RETURN
      END
      SUBROUTINE U2FORM(N,DF,ESTOR,USTOR)
C     FORM STRAIN ENERGY TENSORS USTOR USING QUADRAIIC POINTS ESTOR.
C     N = DIMENSION OF TENSORS.
C     DF(I) = FINITE DIFFERENCE IN INDEPENDENT VARIABLE I.
C     ESTOR(I) = STORAGE VECTOR FOR ENERGY EVALUATION I.
C     USTOR(I) = STORAGE VECTOR FOR TENSOR COMPONENT I.
      INTEGER N
      DOUBLE PRECISION DF(1),ESTOR(1),USTOR(1)
      INTEGER I,J,I1,I2,LOCII,N1
      DOUBLE PRECISION C,CI,CJ,CIJ,DI,DJ
      N1 = 1+N
      I1 = 1
      I2 = N1
C          FORM 0TH-ORDER TENSOR
      C = ESTOR(1)
      USTOR(1) = C
C          FORM 1ST-ORDER TENSOR
      DO 100 I=1,N
      DI = DF(I)
      I1 = I1+1
      CI = ESTOR(I1)
      LOCII = N1 + I*(I+1)/2
C     FORM X TYPE TERM
      USTOR(I1) = (- 1.5D0*C + 2.D0*CI - .5D0*ESTOR(LOCII))/DI
C          FORM 2ND-ORDER TENSOR
      DO 100 J=1,I
      DJ = DI*DF(J)
      I2 = I2+1
      CIJ = ESTOR(I2)
      IF(J.LT.I)GO TO 79
C     FORM XX TYPE TERM
      USTOR(I2) = (C - 2.D0*CI + CIJ)/DJ
      GO TO 100
   79 CJ = ESTOR(1+J)
C     FORM YX TYPE TERM
      USTOR(I2) = (C - CI - CJ + CIJ)/DJ
  100 CONTINUE
      RETURN
      END
      SUBROUTINE U3FORM(N,DF,ESTOR,USTOR)
C     FORM STRAIN ENERGY TENSORS USTOR USING CUBIC POINTS ESTOR.
C     N = DIMENSION OF TENSORS.
C     DF(I) = FINITE DIFFERENCE IN INDEPENDENT VARIABLE I.
C     ESTOR(I) = STORAGE VECTOR FOR ENERGY EVALUATIONS I.
C     USTOR(I) = STORAGE VECTOR FOR TENSOR COMPONENT I.
      INTEGER N
      DOUBLE PRECISION DF(1),ESTOR(1),USTOR(1)
      INTEGER I,J,K,I1,I2,I3,LOCII,LOCJJ,LOCIII,LOCIIJ,LOCIJJ,LOCIK,
     1LOCJK,N1,N2
      DOUBLE PRECISION C,CI,CJ,CK,CIJ,CIK,CJK,CIJK,C116,C13,DI,DJ,DK
      N1 = 1+N
      N2 = N1 + N*(N+1)/2
      C116 = 11.D0/6.D0
      C13 = 1.D0/3.D0
      I1 = 1
      I2 = N1
      I3 = N2
C          FORM 0TH-ORDER TENSOR
      C = ESTOR(1)
      USTOR(1) = C
C          FORM 1ST-ORDER TENSOR
      DO 100 I=1,N
      DI = DF(I)
      I1 = I1+1
      CI = ESTOR(I1)
      LOCII = N1 + I*(I+1)/2
      LOCIII = N2 + I*(I+1)*(I+2)/6
C     FORM X TYPE TERM
      USTOR(I1) = (- C116*C + C13*ESTOR(LOCIII) + 3.D0*CI
     1- 1.5D0*ESTOR(LOCII))/DI
C          FORM 2ND-ORDER TENSOR
      DO 100 J=1,I
      DJ = DI*DF(J)
      I2 = I2+1
      CIJ = ESTOR(I2)
      CJ = ESTOR(1+J)
      IF(J.LT.I)GO TO 79
C     FORM XX TYPE TERM
```

```
      USTOR(I2) = (2.D0*C - ESTOR(LOCIII) - 5.D0*CI + 4.D0*CIJ)/DJ
      GO TO 81
   79 LOCJJ = N1 + J*(J+1)/2
      LOCIJJ = N2 + (I-1)*I*(I+1)/6 + (J+1)*J/2
      LOCIIJ = N2 + (I-1)*I*(I+4)/6 + J
C     FORM YX TYPE TERM
      USTOR(I2) = (2.D0*C - 2.5D0*CJ + .5D0*ESTOR(LOCJJ)
     1 - .5D0*ESTOR(LOCIIJ) - .5D0*ESTOR(LOCIJJ) + .5D0*ESTOR(LOCII)
     2 - 2.5D0*CI + 3.D0*CIJ)/DJ
C            FORM 3RD-ORDER TENSOR
   81 DO 100 K=1,J
      DK = DJ*DF(K)
      I3 = I3+1
      CIJK = ESTOR(I3)
      LOCIK = N1 + (I-1)*I/2 + K
      CIK = ESTOR(LOCIK)
      CK = ESTOR(1+K)
      IF(J.LT.I)GO TO 89
C     FORM XXX TYPE TERM
      IF(K.EQ.J)USTOR(I3) = (- C + CIJK + 3.D0*CI - 3.D0*CIJ)/DK
C     FORM YYX TYPE TERM
      IF(K.LT.J)USTOR(I3) = (-C+CK+CIJK-CIJ + 2.D0*CI - 2.D0*CIK)/DK
      GO TO 100
   89 LOCJK = N1 + (J-1)*J/2 + K
      CJK = ESTOR(LOCJK)
C     FORM YXX TYPE TERM
      IF(K.EQ.J)USTOR(I3) = (-C + 2.D0*CJ -CJK+CIJK+CI - 2.D0*CIJ)/DK
C     FORM ZYX TYPE TERM
      IF(K.LT.J)USTOR(I3) = (-C + CK+CI+CJ + CIJK - CIJ-CIK-CJK)/DK
  100 CONTINUE
      RETURN
      END
      SUBROUTINE UFILL(II,ORD,ET,USTOR)
C     FILL USTOR OF ORDER ORD FOR ELEMENT II.
C     II = ELEMENT NUMBER.
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR DIFFERENCE EXPANSION.
C     ET(I) = CURRENT VALUE OF STRAIN COMPONENT I.
C     USTOR(I) = STORAGE VECTOR FOR TENSOR COMPONENT I.
C     NS = NUMBER OF STRAIN COMPONENTS.
C     DFE = FINITE DIFFERENCE SIZE FOR STRAIN VARIABLES.
```

```
      COMMON/COMNS/NS/COMDFE/DFE
      INTEGER II,ORD,NS
      DOUBLE PRECISION ET(1),USTOR(1),DFE
      INTEGER I
      DOUBLE PRECISION ESTOR(20),DF(3)
      DO 10 I=1,NS
   10 DF(I) = DFE
      CALL EVAL(II,ORD,NS,ET,DF,ESTOR)
      IF(ORD.EQ.2)CALL U2FORM(NS,DF,ESTOR,USTOR)
      IF(ORD.EQ.3)CALL U3FORM(NS,DF,ESTOR,USTOR)
      RETURN
      END


      SUBROUTINE CFORM(Q,EGEOM,CMAT)
C     FORM 3X9 PRESSURE LOAD MATRIX CMAT.
C     Q(I) = CUMULATIVE NODAL DISPLACEMENT AT DOF I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     CMAT = UNSYMMETRIC STIFFNESS PARTITION DUE TO PRESSURE LOADS.
      DOUBLE PRECISION Q(1),EGEOM(1),CMAT(3,9)
      INTEGER I
      DOUBLE PRECISION CX2,CY2,CZ2,CX3,CY3,CZ3
      CX2 = EGEOM(1) + Q(4) - Q(1)
      CY2 = Q(5) - Q(2)
      CZ2 = Q(6) - Q(3)
      CX3 = EGEOM(3) + Q(7) - Q(1)
      CY3 = EGEOM(2) + Q(8) - Q(2)
      CZ3 = Q(9) - Q(3)
      DO 10 I=1,3
      CMAT(I,I) = 0.D0
      CMAT(I,I+3) = 0.D0
   10 CMAT(I,I+6) = 0.D0
      CMAT(1,2) =  CZ2-CZ3
      CMAT(1,3) = -CY2+CY3
      CMAT(1,5) =  CZ3
      CMAT(1,6) = -CY3
      CMAT(1,8) = -CZ2
      CMAT(1,9) =  CY2
      CMAT(2,1) = -CZ2+CZ3
      CMAT(2,3) =  CX2-CX3
      CMAT(2,4) = -CZ3
      CMAT(2,6) =  CX3
      CMAT(2,7) =  CZ2
      CMAT(2,9) = -CX2
      CMAT(3,1) =  CY2-CY3
      CMAT(3,2) = -CX2+CX3
      CMAT(3,4) =  CY3
      CMAT(3,5) = -CX3
      CMAT(3,7) = -CY2
      CMAT(3,8) =  CX2
      RETURN
      END


      SUBROUTINE GENER8(II,K)
C     FORM STIFFNESS MATRIX K FOR ELEMENT II IN NODAL COORDINATES.
C     II,K = ELEMENT NUMBER,ELEMENT STIFFNESS MATRIX.
C     NS,NF = NUMBER OF STRAIN COMPONENTS,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     QQ(I) = CUMULATIVE NODAL DISPLACEMENT AT DOF I.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     PR = ELEMENT PRESSURE INTENSITY FACTOR.
C     PRREF(I) = INTENSITY ON ELEMENT I FOR PRESSURE REFERENCE VECTOR.
      COMMON/COMNS/NS/COMNF/NF/COMEL/ELNO/COMEG/EGEOM/COMT/T/COMQQ/QQ
      COMMON/COMIPR/IPRESS/COMPR/PR/COMPRR/PRREF
      INTEGER II,NS,NF,ELNO(3,1),IPRESS
      DOUBLE PRECISION K(9,9),EGEOM(3,1),T(1),QQ(1),PR,PRREF(1)
      INTEGER I,J,IU,NF2,NF3,I1
      DOUBLE PRECISION V,E(3),S(3),DO(3,3),D(6),Q(9),A(3,6),
     1H(6,6),G(6,9),USTOR(10)
      DOUBLE PRECISION C,P6,CMAT(3,9)
      EQUIVALENCE(A(1),G(1))
      V = .5D0*T(II)*EGEOM(1,II)*EGEOM(2,II)
      NF2 = NF*2
      NF3 = NF*3
      CALL QFILL(NF,ELNO(1,II),QQ,Q)
      CALL DFILL(NF,EGEOM(1,II),Q,D)
      CALL EFILL(NF,D,E)
      CALL UFILL(II,2,E,USTOR)
      IU = 1
      DO 10 I=1,NS
      IU = IU+1
   10 S(I) = USTOR(IU)*V
      DO 20 I=1,NS
      DO 20 J=1,I
      IU = IU+1
      DO(I,J) = USTOR(IU)*V
   20 DO(J,I) = DO(I,J)
      CALL AFILL(NF,D,A,1)
      CALL MTRAN(DO,3,NS,A,6,NF2,H)
      DO 50 I=1,NF
      I1 = NF+I
      H(I,I) = H(I,I) + S(1)
      H(I1,I1) = H(I1,I1) + S(2)
      H(I,I1) = H(I,I1) + S(3)
   50 H(I1,I) = H(I1,I) + S(3)
      CALL GFILL(NF,EGEOM(1,II),G)
      CALL MTRAN(H,6,NF2,G,9,NF3,K)
      IF(IPRESS.EQ.0)GO TO 101
      CALL CFORM(Q,EGEOM(1,II),CMAT)
      P6 = PR*PRREF(II)/6.D0
      DO 100 I=1,NF
      DO 100 J=1,NF3
      C = P6*CMAT(I,J)
      K(I,J) = K(I,J) - C
      K(I+3,J) = K(I+3,J) - C
  100 K(I+6,J) = K(I+6,J) - C
  101 CALL ROTK(NF,ELNO(1,II),K)
      RETURN
      END

      SUBROUTINE USUM1(ORD,USTOR,N,Q,P)
C     FOR ORD=2 COMPUTE P(I) = USTOR(I,J)*Q(J).
C     FOR ORD=3 COMPUTE P(I) = USTOR(I,J,K)*Q(J)*Q(K).
C     ORD = TENSOR ORDER TO BE USED FOR FINITE-DIFFERENCE EXPANSIONS.
C     USTOR = TENSOR STORAGE ARRAY.
C     N = TENSOR DIMENSION.
```

```
C     Q = TENSOR VECTOR ARGUMENT.
C     P = COMPUTED SUMMED VECTOR.
      INTEGER ORD,N
      DOUBLE PRECISION USTOR(1),P(1),Q(1)
      INTEGER I,J,K,IU
      DOUBLE PRECISION QI,QJ,QK,QIJ,C
      DO 100 I=1,N
  100 P(I) = 0.D0
      IU = 0
      IF(ORD.GT.2)GO TO 201
      DO 200 I=1,N
      QI = Q(I)
      DO 200 J=1,I
      IU = IU+1
      C = USTOR(IU)
      P(I) = P(I) + C*Q(J)
      IF(J.NE.I)P(J) = P(J) + C*QI
  200 CONTINUE
      RETURN
  201 DO 300 I=1,N
      QI = Q(I)
      DO 300 J=1,I
      QJ = Q(J)
      QIJ = QI*QJ
      DO 300 K=1,J
      QK = Q(K)
      IU = IU+1
      C = USTOR(IU)
      P(I) = P(I) + C*QJ*QK
      IF(K.EQ.J)GO TO 61
      P(I) = P(I) + C*QJ*QK
      GO TO 71
   61 IF(J.EQ.I)GO TO 300
      P(J) = P(J) + 2.D0*C*QIJ
      GO TO 300
   71 P(K) = P(K) + C*QIJ
      IF(J.EQ.I)GO TO 300
      P(J) = P(J) + 2.D0*C*QI*QK
      P(K) = P(K) + C*QIJ
  300 CONTINUE
```

```
      RETURN
      END

      SUBROUTINE USUM21(USTOR,N,Q1,Q2,P)
C     COMPUTE P(I) = USTOR(I,J,K)*Q1(J)*Q2(K).
C     USTOR = TENSOR STORAGE ARRAY.
C     N = TENSOR DIMENSION.
C     Q1,Q2 = TENSOR VECTOR ARGUMENTS.
C     P = COMPUTED SUMMED VECTOR.
      INTEGER N
      DOUBLE PRECISION USTOR(1),Q1(1),Q2(1),P(1)
      INTEGER I,J,K,IU
      DOUBLE PRECISION C
      DO 100 I=1,N
  100 P(I) = 0.D0
      IU = 0
      DO 200 I=1,N
      DO 200 J=1,I
      DO 200 K=1,J
      IU = IU+1
      C = USTOR(IU)
      P(I) = P(I) + C*Q1(J)*Q2(K)
      IF(K.EQ.J)GO TO 161
      P(I) = P(I) + C*Q1(K)*Q2(J)
      GO TO 171
  161 IF(J.EQ.I)GO TO 200
      P(J) = P(J) + C*(Q1(I)*Q2(J)+Q1(J)*Q2(I))
      GO TO 200
  171 P(K) = P(K) + C*Q1(I)*Q2(J)
      IF(J.EQ.I)GO TO 200
      P(J) = P(J) + C*(Q1(I)*Q2(K)+Q1(K)*Q2(I))
      P(K) = P(K) + C*Q1(J)*Q2(I)
  200 CONTINUE
      RETURN
      END

      SUBROUTINE P1COMP(NN,QQSTAR,QQDOT,PR0,PR1,PP)
C     COMPUTE 2ND ORDER FUNDAMENTAL LOAD TERM PP USING FUNDAMENTAL
```

```
C     DISPLACEMENTS (REFERENCE VALUES QQSTAR AND RATES QQDOT).
C     NN = TOTAL SYSTEM DOF.
C     QQSTAR(I) = CURRENT CUMULATIVE NODAL DISPLACEMENT AT NODE I.
C     QQDOT(I) = NODAL DISPLACEMENT RATE AT NODE I.
C     PRO,PR1 = PRESSURE FACTORS AT START,END OF LOAD STEP.
C     PP(I) = COMPUTED PSUEDO FORCE TERM AT DOF I.
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR DIFFERENCE EXPANSION.
C     NEL,NS,NF = NUMBER OF ELEMENTS,STRAIN COMPONENTS,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     PRREF(I) = PRESSURE INTENSITY ON ELEMENT I FOR PRESSURE VECTOR.
      COMMON/COMORD/ORD/COMNEL/NEL/COMNS/NS/COMNF/NF
      COMMON/COMEL/ELNO/COMEG/EGEOM/COMT/T
      COMMON/COMIPR/IPRESS/COMPRR/PRREF
      INTEGER NN,ORD,NEL,NS,NF,ELNO(3,1),IPRESS
      DOUBLE PRECISION QQSTAR(1),QQDOT(1),PRO,PR1,PP(1),EGEOM(3,1),T(1),
     1PRREF(1)
      INTEGER NF2,NF3,IU2,IU3,I,II,M
      DOUBLE PRECISION V,C,QSTAR(9),QDOT(9),DSTAR(6),DDOT(6),ESTAR(3),
     1EDOT(3),ASTAR(3,6),A1DOT(3,6),STORA(3),STORB(3),STORM(6),G(6,9),
     2P(9),USTAR(20)
      DOUBLE PRECISION C0,C1,VSTAR(6),VDOT(6),V1(3),V2(3),V3(3)
      NF2 = NF*2
      NF3 = NF*3
      IU2 = 2 + NS
      IU3 = IU2 + NS*(NS+1)/2
      DO 5 I=1,NN
    5 PP(I) = 0.D0
      DO 1000 II=1,NEL
      V = .5D0*T(II)*EGEOM(1,II)*EGEOM(2,II)
C     FORM FUNDAMENTAL REFERENCE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQSTAR,QSTAR)
      CALL DFILL(NF,EGEOM(1,II),QSTAR,DSTAR)
      CALL AFILL(NF,DSTAR,ASTAR,1)
      CALL EFILL(NF,DSTAR,ESTAR)
C     FORM STRESS-STRAIN TENSORS
      CALL UFILL(II,ORD,ESTAR,USTAR)
C     FORM FUNDAMENTAL RATE QUANTITIES
```

```
      CALL QFILL(NF,ELNO(1,II),QQDOT,QDOT)
      CALL DFILL(NF,EGEOM(1,II),QDOT,DDOT)
      CALL AFILL(NF,DDOT,A1DOT,0)
      CC 30 I=1,NS
      C = 0.DO
      DO 25 M=1,NF2
   25 C = C + ASTAR(I,M)*DDOT(M)
   30 ECOT(I) = C
C     FORM STORM = DO PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES
      CALL USUM1(2,USTAR(IU2),NS,EDOT,STORA)
      DC 110 I=1,NF2
      C = 0.DO
      DO 105 M=1,NS
  105 C = C + STORA(M)*A1DOT(M,I)
  110 STORM(I) = 2.DO*C
      DO 114 I=1,NS
      C = 0.DO
      DC 112 M=1,NF2
  112 C = C + A1DOT(I,M)*DDOT(M)
  114 STORB(I) = C
      CALL USUM1(2,USTAR(IU2),NS,STORB,STORA)
      DO 120 I=1,NF2
      C = 0.DO
      DO 115 M=1,NS
  115 C = C + STORA(M)*ASTAR(M,I)
  120 STORM(I) = STORM(I) + C
      IF(ORD.LT.3)GO TO 501
C     ACD DI PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES TO STORM
      CALL USUM1(3,USTAR(IU3),NS,EDOT,STORA)
      DO 210 I=1,NF2
      C = 0.DO
      DC 205 M=1,NS
  205 C = C + STORA(M)*ASTAR(M,I)
  210 STORM(I) = STORM(I) + C
C     COMPUTE ELEMENT FORCES P(I) = VOLUME INTEGRAL OF G(M,I)*STORM(M)
  501 CALL GFILL(NF,EGEOM(1,II),G)
      DC 510 I=1,NF3
      C = 0.DO
      DO 505 M=1,NF2
  505 C = C + G(M,I)*STORM(M)
  510 P(I) = C*V
      IF(IPRESS.EQ.0)GO TO 1000
C     ADD NONCONSERVATIVE PRESSURE FORCES TO P
      VSTAR(1) = EGEOM(1,II) + QSTAR(4) - QSTAR(1)
      VSTAR(2) = QSTAR(5) - QSTAR(2)
      VSTAR(3) = QSTAR(6) - QSTAR(3)
      VSTAR(4) = EGEOM(3,II) + QSTAR(7) - QSTAR(1)
      VSTAR(5) = EGEOM(2,II) + QSTAR(8) - QSTAR(2)
      VSTAR(6) = QSTAR(9) - QSTAR(3)
      VDOT(1) = QDOT(4) - QDOT(1)
      VDOT(2) = QDOT(5) - QDOT(2)
      VDOT(3) = QDOT(6) - QDOT(3)
      VDOT(4) = QDOT(7) - QDOT(1)
      VDOT(5) = QDOT(8) - QDOT(2)
      VDOT(6) = QDOT(9) - QDOT(3)
      C0 = PR0*PRREF(II)/3.D0
      C1 = (PR1-PR0)*PRREF(II)/3.D0
      CALL VCROSS(VDOT,VSTAR(4),V1)
      CALL VCROSS(VSTAR,VDOT(4),V2)
      CALL VCROSS(VDOT,VDOT(4),V3)
      DO 610 I=1,3
      C = C1*(V1(I)+V2(I)) + C0*V3(I)
      P(I) = P(I) - C
      P(I+3) = P(I+3) - C
  610 P(I+6) = P(I+6) - C
C     ADD ELEMENT FORCES P TO SYSTEM FORCES PP
 1000 CALL PFILL(NF,ELNC(1,II),P,PP)
      RETURN
      END


      SUBROUTINE RATES(KMAT,P1,NN,KFD,RP,PR0,PR1,LSIGN,RL,RRL,RQ,RRQ)
C     COMPUTE 1ST AND 2ND PATH RATES FOR LOAD DIRECTION VECTOR RP.
C     LOAD PARAMETER RATES = RL,RRL. DISPLACEMENT RATES = RQ,RRQ.
C     KMAT = SYSTEM JACOBIAN STIFFNESS MATRIX.
C     P1 = TEMPORARY FORCE STORAGE VECTOR.
C     NN = TOTAL SYSTEM DOF.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     RP(I) = RESIDUAL LOAD (LOAD STEP) AT DOF I.
C     PR0,PR1 = PRESSURE FACTORS AT START,END OF LOAD STEP.
```

```
C     LSIGN = +,- IF LOAD PARAMETER IS INCREASING,DECREASING.
C     RL,RRL = COMPUTED LOAD PARAMETER 1ST,2ND ORDER RATES.
C     RQ(I),RRQ(I) = COMPUTED 1ST,2ND ORDER DISPLACEMENT RATE AT DOF I.
C     QQ(I) = CURRENT CUMULATIVE DISPLACEMENT AT DOF I.
      COMMON/COMQQ/QQ
      INTEGER NN,KFD(1)
      DOUBLE PRECISION RP(1),PR0,PR1,LSIGN,RL,RRL,RQ(1),RRQ(1),QQ(1),
     1KMAT(NN,NN),P1(1)
      INTEGER I
      DOUBLE PRECISION RSIGN
      CALL SOLVE(KMAT,NN,KFD,RP,RQ)
      CALL P1COMP(NN,QQ,RQ,PR0,PR1,P1)
      RL = 0.D0
      RRL = 0.D0
      DO 50 I=1,NN
      IF(KFD(I).LT.0)GO TO 49
      RL = RL + RP(I)*RQ(I)
      RRL = RRL + P1(I)*RQ(I)
      GO TO 50
   49 P1(I) = 0.D0
   50 CONTINUE
      CALL SOLVE(KMAT,NN,KFD,P1,RRQ)
      DO 55 I=1,NN
      IF(KFD(I).GT.0)RRL = RRL + RP(I)*RRQ(I)
   55 CONTINUE
      RSIGN = 1.D0
      IF(RL.LT.0.D0)RSIGN = -1.D0
      RL = RSIGN/RL
      RRL = RSIGN*RL**2*RRL*.5D0
      DO 60 I=1,NN
   60 RRQ(I) = RRL*RQ(I) - RL*RRQ(I)
      RL = LSIGN*DSQRT(RL)
      DO 80 I=1,NN
   80 RQ(I) = RL*RQ(I)
      RETURN
      END

      SUBROUTINE STEP(LSIGN,RL,RRL,NN,RQ,RRQ,JUMPR,MJUMP,NJUMP,DSLOPE,
     1PATH,LAMS)
```

```
C     COMPUTE PATH = PATH DISTANCE, LAMS = LOAD STEP SIZE.
C     LSIGN = +,- IF LOAD PARAMETER IS INCREASING,DECREASING.
C     RL,RRL = 1ST,2ND ORDER LOAD PARAMETER RATES.
C     NN = TOTAL SYSTEM DOF.
C     RQ(I),RRQ(I) = 1ST,2ND ORDER DIAPLACEMENT RATES AT DOF I.
C     JUMPR = FRACTION OF LOAD INCREMENT PRECEEDING LIMIT POINT
C     AT WHICH LIMIT IS TO BE TRAVERSED.
C     MJUMP = MAXIMUM NUMBER OF INCREMENT DIVISIONS TO PERFORM WHEN
C     NEARING A LIMIT POINT.
C     NJUMP = CURRENT NUMBER OF INCREMENT DIVISIONS TO PERFORM WHEN
C     NEARING A LIMIT POINT.
C     DSLOPE = MAXIMUM SLOPE RATIO (CHANGE/AVERAGE) DURING LOAD STEP.
C     PATH = COMPUTED PATH STEP SIZE TO BE TAKEN.
C     LAMS = INPUT MAXIMUM,COMPUTED ACTUAL LOAD STEP SIZE.
      INTEGER NN,MJUMP,NJUMP
      DOUBLE PRECISION LSIGN,RL,RRL,RQ(1),RRQ(1),JUMPR,DSLOPE,PATH,LAMS
      INTEGER N,I
      DOUBLE PRECISION SLIM,LAMLIM,PREC,DSLOP,CR,CRR,C
      IF(LSIGN*RL.LE.0.D0)STOP 601
      IF(RL*RRL.GE.0.D0)GO TO 21
C         POSSIBLE LIMIT POINT
      SLIM = -RL/RRL
      LAMLIM = DABS(SLIM*RL + .5D0*SLIM**2*RRL)
C     CHECK CLOSENESS OF LIMIT POINT
      IF(LAMLIM.LT.JUMPR)GO TO 11
      IF(LAMS.LT.LAMLIM/MJUMP)GO TO 21
C         LIMIT IS CLOSE. TAKE FRACTIONAL STEP JUMP
      NJUMP = NJUMP-1
      IF(NJUMP.LT.2)NJUMP = 2
      IF(LAMS.LT.LAMLIM)GO TO 7
C     STEP IS LARGER THAN LIMIT. JUMP TOWARD LIMIT VALUE.
      LAMS = LAMLIM/NJUMP
      GO TO 10
C     STEP IS SMALLER THAN LIMIT. JUMP TOWARD STEP VALUE.
    7 N = NJUMP*LAMS/LAMLIM + 1
      IF(N.EQ.1)NJUMP = NJUMP+1
      LAMS = LAMS/N
   10 LAMS = LSIGN*LAMS
      PATH = (-RL+LSIGN*DSQRT(RL**2+2.D0*LAMS*RRL))/RRL
      RETURN
```

```
C          LIMIT IS VERY CLOSE. TRAVERSE THE LIMIT POINT
   11 NJUMP = MJUMP+1
      LSIGN = -LSIGN
      LAMS = 0.D0
      PATH = -2.D0*RL/RRL
      RETURN
C          LIMIT IS NOT CLOSE. CHECK SLOPE CHANGE FOR ALLOWABLE STEP
   21 NJUMP = MJUMP+1
      CR = 0.D0
      CRR = 0.D0
      DO 50 I=1,NN
      CR = CR + DABS(RQ(I))
   50 CRR = CRR + DABS(RRQ(I))
      LAMS = LSIGN*LAMS
   51 PREC = DABS(2.D0*LAMS*RRL/RL**2)
      IF(PREC.LT.1.D-8)PATH = LAMS/RL
      IF(PREC.GE.1.D-8)PATH = (-RL+LSIGN*DSQRT(RL**2+2.D0*LAMS*RRL))/RRL
      DSLOP = PATH*RRL
      DSLOP = DSLOP/(RL+.5D0*DSLOP)
      C = PATH*CRR
      C = C/(CR+.5D0*C)
      IF(DABS(DSLOP).LE.DSLOPE.AND.C.LE.DSLOPE)RETURN
      LAMS = .5D0*LAMS
      IF(DABS(LAMS).LE.1.D-3)STOP 602
      GO TO 51
      END


      SUBROUTINE EIGEN1(NN,NCODE,QQDELT,SCRIT,QQPOST,LAM,PR0,PR1,PP)
C     FORM POSTBUCKLING LOAD TERM PP USING FUNDAMENTAL DISPLACEMENTS
C     (REFERENCE VALUES QQSTAR AND CRITICAL INCREMENT QQDELT) AND
C     POSTBUCKLING DISPLACEMENT EIGENVECTOR QQPOST.
C     NN = TOTAL SYSTEM DOF.
C     NCODE = 0,1 FOR LINEAR,NONLINEAR EIGEN SOLUTION.
C     QQDELT(I) = ESTIMATED INCR. DISPLACEMENT AT DOF I TO CRITICAL PT.
C     SCRIT = ESTIMATED INCREMENTAL PATH VALUE TO CRITICAL POINT.
C     QQPOST(I) = ESTIMATED BUCKLING DISPLACEMENT AT DOF I.
C     LAM = ESTIMATED INCREMENTAL LOAD PARAMETER VALUE TO CRITICAL PT.
C     PR0,PR1 = PRESSURE FACTORS AT START,END OF LOAD STEP.
C     PP(I) = COMPUTED RIGHT-HAND-SIDE FOR INVERSE POWER ITERATION.
```

```
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR DIFFERENCE EXPANSION.
C     NEL,NS,NF = NUMBER OF ELEMENTS,STRAIN COMPONENTS,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     QQSTAR(I) = CURRENT CUMULATIVE DISPLACEMENT AT DOF I.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     PRREF(I) = PRESSURE INTENSITY ON ELEMENT I FOR PRESSURE VECTOR.
      COMMON/COMORD/ORD/COMNEL/NEL/COMNS/NS/COMNF/NF
      COMMON/COMEL/ELNO/COMEG/EGEOM/COMT/T/COMQQ/QQSTAR
      COMMON/COMIPR/IPRESS/COMPRR/PRREF
      INTEGER NN,NCODE,ORD,NEL,NS,NF,ELNO(3,1),IPRESS
      DOUBLE PRECISION QQDELT(1),SCRIT,QQPOST(1),LAM,PRO,PR1,PP(1)
      DOUBLE PRECISION EGEOM(3,1),T(1),QQSTAR(1),PRREF(1)
      INTEGER NF2,NF3,IU2,IU3,I,II,M
      DOUBLE PRECISION V,C,QSTAR(9),QDELT(9),QPOST(9),DSTAR(6),DDELT(6),
     1DPOST(6),ESTAR(3),EDELT(3),EPOST(3),ASTAR(3,6),A1DELT(3,6),
     2A1POST(3,6),STORA(3),STORB(3),STORM(6),G(6,9),P(9),USTAR(20),CA
      DOUBLE PRECISION C0,C1,VSTAR(6),VDELT(6),VPOST(6),V1(3),V2(3),
     1V3(3),V4(3)
      NF2 = NF*2
      NF3 = NF*3
      IU2 = 2 + NS
      IU3 = IU2 + NS*(NS+1)/2
      DO 5 I=1,NN
    5 PP(I) = 0.D0
      DO 1000 II=1,NEL
      V = .5D0*T(II)*EGEOM(1,II)*EGEOM(2,II)
C     FORM FUNDAMENTAL REFERENCE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQSTAR,QSTAR)
      CALL DFILL(NF,EGEOM(1,II),QSTAR,DSTAR)
      CALL AFILL(NF,DSTAR,ASTAR,1)
      CALL EFILL(NF,DSTAR,ESTAR)
C     FORM STRESS-STRAIN TENSORS
      CALL UFILL(II,ORD,ESTAR,USTAR)
C     FORM FUNDAMENTAL CRITICAL INCREMENT QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQDELT,QDELT)
      CALL DFILL(NF,EGEOM(1,II),QDELT,DDELT)
      CALL AFILL(NF,DDELT,A1DELT,0)
      DO 20 I=1,NS
      C = 0.D0
      DO 15 M=1,NF2
      CA = ASTAR(I,M)
      IF(NCODE.GT.0)CA = CA + .5D0*SCRIT*A1DELT(I,M)
   15 C = C + CA*DDELT(M)
   20 EDELT(I) = C
C     FORM POSTBUCKLING EIGEN QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQPOST,QPOST)
      CALL DFILL(NF,EGEOM(1,II),QPOST,DPOST)
      CALL AFILL(NF,DPOST,A1POST,0)
      DO 30 I=1,NS
      C = 0.D0
      DO 25 M=1,NF2
      CA = ASTAR(I,M)
      IF(NCODE.GT.0)CA = CA + SCRIT*A1DELT(I,M)
   25 C = C + CA*DPOST(M)
   30 EPOST(I) = C
C     FORM STORM = DO PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES
      DO 104 I=1,NS
      C = 0.D0
      DO 102 M=1,NF2
  102 C = C + A1DELT(I,M)*DPOST(M)
  104 STORB(I) = C
      CALL USUM1(2,USTAR(IU2),NS,STORB,STORA)
      DO 110 I=1,NF2
      C = 0.D0
      DO 105 M=1,NS
  105 C = C + STORA(M)*ASTAR(M,I)
  110 STORM(I) = C
      CALL USUM1(2,USTAR(IU2),NS,EPOST,STORA)
      DO 120 I=1,NF2
      C = 0.D0
      DO 115 M=1,NS
  115 C = C + STORA(M)*A1DELT(M,I)
  120 STORM(I) = STORM(I) + C
      CALL USUM1(2,USTAR(IU2),NS,EDELT,STORA)
      DO 130 I=1,NF2
      C = 0.D0
      DO 125 M=1,NS
  125 C = C + STORA(M)*A1POST(M,I)
  130 STORM(I) = STORM(I) + C
      IF(ORD.LT.3)GO TO 501
C     ADD D1 PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES TO STORM
      CALL USUM21(USTAR(IU3),NS,EPOST,EDELT,STORA)
      DO 210 I=1,NF2
      C = 0.D0
      DO 205 M=1,NS
  205 C = C + STORA(M)*ASTAR(M,I)
  210 STORM(I) = STORM(I) + C
      IF(NCODE.EQ.0)GO TO 501
      DO 220 I=1,NF2
      C = 0.D0
      DO 215 M=1,NS
  215 C = C + STORA(M)*A1DELT(M,I)
  220 STORM(I) = STORM(I) + SCRIT*C
      CALL USUM1(3,USTAR(IU3),NS,EDELT,STORA)
      DO 230 I=1,NF2
      C = 0.D0
      DO 225 M=1,NS
  225 C = C + STORA(M)*A1POST(M,I)
  230 STORM(I) = STORM(I) + .5D0*SCRIT*C
C     COMPUTE ELEMENT FORCES P(I) = VOLUME INTEGRAL OF G(M,I)*STORM(M)
  501 CALL GFILL(NF,EGEOM(1,II),G)
      DO 510 I=1,NF3
      C = 0.D0
      DO 505 M=1,NF2
  505 C = C + G(M,I)*STORM(M)
  510 P(I) = C*V
      IF(IPRESS.EQ.0)GO TO 1000
C     ADD NONCONSERVATIVE PRESSURE FORCES TO P
      VSTAR(1) = EGEOM(1,II) + QSTAR(4) - QSTAR(1)
      VSTAR(2) = QSTAR(5) - QSTAR(2)
      VSTAR(3) = QSTAR(6) - QSTAR(3)
      VSTAR(4) = EGEOM(3,II) + QSTAR(7) - QSTAR(1)
      VSTAR(5) = EGEOM(2,II) + QSTAR(8) - QSTAR(2)
      VSTAR(6) = QSTAR(9) - QSTAR(3)
      VDELT(1) = QDELT(4) - QDELT(1)
      VDELT(2) = QDELT(5) - QDELT(2)
      VDELT(3) = QDELT(6) - QDELT(3)
      VDELT(4) = QDELT(7) - QDELT(1)
      VDELT(5) = QDELT(8) - QDELT(2)
      VDELT(6) = QDELT(9) - QDELT(3)
      VPOST(1) = QPOST(4) - QPOST(1)
      VPOST(2) = QPOST(5) - QPOST(2)
      VPOST(3) = QPOST(6) - QPOST(3)
      VPOST(4) = QPOST(7) - QPOST(1)
      VPOST(5) = QPOST(8) - QPOST(2)
      VPOST(6) = QPOST(9) - QPOST(3)
      CALL VCROSS(VSTAR,VPOST(4),V1)
      CALL VCROSS(VPOST,VSTAR(4),V2)
      CALL VCROSS(VDELT,VPOST(4),V3)
      CALL VCROSS(VPOST,VDELT(4),V4)
      C0 = LAM*(PR1-PR0)*PRREF(II)/6.D0
      C1 = PR0*PRREF(II)/6.D0
      IF(NCODE.GT.0)C1 = C1 + SCRIT*C0
      DO 610 I=1,3
      C = C0*(V1(I)+V2(I)) + C1*(V3(I)+V4(I))
      P(I) = P(I) - C
      P(I+3) = P(I+3) - C
  610 P(I+6) = P(I+6) - C
C     ADD ELEMENT FORCES P TO SYSTEM FORCES PP
 1000 CALL PFILL(NF,ELNO(1,II),P,PP)
      RETURN
      END

      SUBROUTINE EIGEN(UO,KMAT,P1,Q0,NN,KFD,PR0,PR1,RL,RRL,RQ,RRQ,
     1MI1,MI2,ERR1,ERR2,EIG1,Q1,N)
C     SOLVE FOR EIGEN LOAD EIG1, AND VECTOR Q1 WITH MAX. INDEX VALUE N.
C     UO = OUTPUT UNIT FILE.
C     KMAT = SYSTEM JACOBIAN STIFFNESS MATRIX.
C     P1 = TEMPORARY FORCE STORAGE VECTOR.
C     Q0 = TEMPORARY DISPLACEMENT STORAGE VECTOR.
C     NN = TOTAL SYSTEM DOF.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     PR0,PR1 = PRESSURE FACTORS AT START,END OF LOAD STEP.
C     RL,RRL = 1ST,2ND ORDER LOAD PARAMETER RATES.
C     RQ(I),RRQ(I) = 1ST,2ND ORDER DISPLACEMENT RATE AT DOF I.
C     MI1,MI2 = MAXIMUM ITERATIONS FOR LINEAR,LINEAR+NONLINEAR SOLUTION.
C     ERR1,ERR2 = MAXIMUM ERROR FOR LINEAR,LINEAR+NONLINEAR SOLUTION.
```

```
C     EIG1 = COMPUTED EIGENVALUE (CRITICAL PATH VALUE).
C     Q1(I) = ITH COMPONENT OF EIGENVECTOR (BUCKLING DISPLACEMENT).
C     N = DOF OF LARGEST Q1 COMPONENT.
      COMMON/COMNF/NF
      INTEGER UO,NN,KFD(1),MI1,MI2,N
      DOUBLE PRECISION KMAT(NN,NN),P1(1),Q0(1),PRO,PR1,
     1RL,RRL,RQ(1),RRQ(1),ERR1,ERR2,EIG1,Q1(1)
      INTEGER I,NI
      DOUBLE PRECISION EIGO,MAX1,DL,C
  201 FORMAT(1H ,'EIGENVALUE = ',D15.8,5X,'DOF OF LARGEST COMPONENT OF E
     1IGEN VECTOR = ',I5)
      NI = 0
      NCODE = 0
      EIG1 = 0.D0
      DO 5 I=1,NN
    5 Q1(I) = I
   11 EIGO = EIG1
      DL = RL
      IF(NCODE.GT.0)DL = DL + .5D0*EIGO*RRL
      DO 20 I=1,NN
      Q0(I) = Q1(I)
      Q1(I) = RQ(I)
      IF(NCODE.GT.0)Q1(I) = Q1(I) + .5D0*EIGO*RRQ(I)
   20 CONTINUE
      CALL EIGEN1(NN,NCODE,Q1,EIGO,Q0,DL,PRO,PR1,P1)
      NOD = NN/NF
      CALL OUTPQ(6,NOD,NF,P1,Q0)
      DO 50 I=1,NN
      IF(KFD(I).LT.0)P1(I) = 0.D0
   50 CONTINUE
      CALL SOLVE(KMAT,NN,KFD,P1,Q1)
      N = 0
      MAX1 = 0.D0
      DO 70 I=1,NN
      IF(DABS(Q1(I)).LE.DABS(MAX1))GO TO 70
      N = I
      MAX1 = Q1(I)
   70 CONTINUE
      MAX1 = 1.D0/MAX1
      EIG1 = -MAX1
      IF(KFD(N).NE.N)N = KFD(N)
      DO 100 I=1,NN
  100 Q1(I) = Q1(I)*MAX1
      WRITE(UO,201)EIG1,N
      NI = NI+1
      C = DABS((EIG1-EIG0)/EIG1)
      IF(NCODE.GT.0.AND.(NI.GE.MI2.OR.C.LE.ERR2))RETURN
      IF(NCODE.EQ.0.AND.(NI.GE.MI1.OR.C.LE.ERR1))NCODE = 1
      GO TO 11
      END


      SUBROUTINE POST2(NN,QQCRIT,QQDOT,QQPOST,LCRIT,LDOT,PR0,PR1,PP,
     1KODE)
C     FORM 2ND ORDER POSTBUCKLING LOAD TERM PP (=P21,P22 FOR KODE=1,2).
C     NN = TOTAL SYSTEM DOF.
C     QQCRIT(I) = PREDICTED TOTAL DISPLACEMENT OF DOF I AT CRITICAL PT.
C     QQDOT(I) = PREDICTED DISPLACEMENT RATE OF DOF I AT CRITICAL PT.
C     QQPOST(I) = ITH DISPLACEMENT OF CRITICAL BUCKLING EIGENVECTOR.
C     LCRIT = PREDICTED LOAD PARAMETER VALUE AT CRITICAL POINT.
C     LDOT = PREDICTED LOAD PARAMETER RATE AT CRITICAL POINT.
C     PR0,PR1 = PRESSURE FACTORS AT START OF LOAD STEP,CRITICAL POINT.
C     PP(I) = COMPUTED PSUEDO-FORCE TERM AT DOF I.
C     KODE = CODE FOR DUAL USE OF SUBROUTINE.
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR DIFFERENCE EXPANSION.
C     NEL,NS,NF = NUMBER OF ELEMENTS,STRAIN COMPONENTS,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     PRREF(I) = PRESSURE INTENSITY ON ELEMENT I FOR PRESSURE VECTOR.
      COMMON/COMORD/ORD/COMNEL/NEL/COMNS/NS/COMNF/NF
      COMMON/COMEL/ELNO/COMEG/EGEOM/COMT/T
      COMMON/COMIPR/IPRESS/COMPRR/PRREF
      INTEGER NN,KODE,ORD,NEL,NS,NF,ELNO(3,1),IPRESS
      DOUBLE PRECISION QQCRIT(1),QQDOT(1),QQPOST(1),LCRIT,LDOT,PR0,PR1,
     1PP(1),EGEOM(3,1),T(1),PRREF(1)
      INTEGER NF2,NF3,IU2,IU3,I,II,M
      DOUBLE PRECISION V,C,QCRIT(9),QDOT(9),QPOST(9),DCRIT(6),DDOT(6),
     1DPOST(6),ECRIT(3),EDOT(3),EPOST(3),ACRIT(3,6),A1DOT(3,6),
     1A1POST(3,6),STORA(3),STORB(3),STORM(6),G(6,9),P(9),UCRIT(20)
      DOUBLE PRECISION CC,C1,VCRIT(6),VDOT(6),VPOST(6),V1(3),V2(3),
     1V3(3),V4(3)
      NF2 = NF*2
      NF3 = NF*3
      IU2 = 2 + NS
      IU3 = IU2 + NS*(NS+1)/2
      DO 5 I=1,NN
    5 PP(I) = 0.D0
      DO 1000 II=1,NEL
      V = .5D0*T(II)*EGEOM(1,II)*EGEOM(2,II)
C     FORM FUNDAMENTAL CRITICAL QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQCRIT,QCRIT)
 1901 FORMAT(1H0,9E14.7)
      WRITE(6,1901)(QCRIT(I),I=1,NF3)
      CALL DFILL(NF,EGEOM(1,II),QCRIT,DCRIT)
      WRITE(6,1901)(DCRIT(I),I=1,NF2)
      CALL AFILL(NF,DCRIT,ACRIT,1)
      DO 1910 I=1,3
 1910 WRITE(6,1901)(ACRIT(I,J),J=1,NF2)
      CALL EFILL(NF,DCRIT,ECRIT)
      WRITE(6,1901)(ECRIT(I),I=1,3)
C     FORM STRESS-STRAIN TENSORS
      CALL UFILL(II,ORD,ECRIT,UCRIT)
C     FORM FUNDAMENTAL RATE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQDOT,QDOT)
      WRITE(6,1901)(QDOT(I),I=1,NF3)
      CALL DFILL(NF,EGEOM(1,II),QDOT,DDOT)
      WRITE(6,1901)(DDOT(I),I=1,NF2)
      CALL AFILL(NF,DDOT,A1DOT,0)
      DO 1920 I=1,3
 1920 WRITE(6,1901)(A1DOT(I,J),J=1,NF2)
      DO 20 I=1,NS
      C = 0.D0
      DO 15 M=1,NF2
   15 C = C + ACRIT(I,M)*DDOT(M)
   20 EDOT(I) = C
      WRITE(6,1901)(EDOT(I),I=1,3)
C     FORM POSTBUCKLING EIGEN QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQPOST,QPOST)
      WRITE(6,1901)(QPOST(I),I=1,NF3)
      CALL DFILL(NF,EGEOM(1,II),QPOST,DPOST)
      WRITE(6,1901)(DPOST(I),I=1,NF2)
      CALL AFILL(NF,DPOST,A1POST,0)
      DO 1930 I=1,3
 1930 WRITE(6,1901)(A1POST(I,J),J=1,NF2)
      DO 30 I=1,NS
      C = 0.D0
      DO 25 M=1,NF2
   25 C = C + ACRIT(I,M)*DPOST(M)
   30 EPOST(I) = C
      WRITE(6,1901)(EPOST(I),I=1,3)
C     FORM STORM = DO PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES
      DO 104 I=1,NS
      C = 0.D0
      DO 102 M=1,NF2
  102 C = C + A1DOT(I,M)*DPOST(M)
  104 STORB(I) = C
      CALL USUM1(2,UCRIT(IU2),NS,STORB,STORA)
      DO 110 I=1,NF2
      C = 0.D0
      DO 105 M=1,NS
  105 C = C + STORA(M)*ACRIT(M,I)
  110 STORM(I) = C
      WRITE(6,1901)(STORM(I),I=1,NF2)
      CALL USUM1(2,UCRIT(IU2),NS,EDOT,STORA)
      DO 120 I=1,NF2
      C = 0.D0
      DO 115 M=1,NS
  115 C = C + STORA(M)*A1POST(M,I)
  120 STORM(I) = STORM(I) + C
      WRITE(6,1901)(STORM(I),I=1,NF2)
      CALL USUM1(2,UCRIT(IU2),NS,EPOST,STORA)
      DO 130 I=1,NF2
      C = 0.D0
      DO 125 M=1,NS
  125 C = C + STORA(M)*A1DOT(M,I)
  130 STORM(I) = STORM(I) + C
      WRITE(6,1901)(STORM(I),I=1,NF2)
      IF(ORD.LT.3)GO TO 501
```

```
C     ADD D1 PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES TO STORM
      CALL USUM21(UCRIT(IU3),NS,EDOT,EPOST,STORA)
      DO 210 I=1,NF2
      C = 0.D0
      DO 205 M=1,NS
  205 C = C + STORA(M)*ACRIT(M,I)
  210 STORM(I) = STORM(I) + C
C     COMPUTE ELEMENT FORCES P(I) = VOLUME INTEGRAL OF G(M,I)*STORM(M)
  501 CALL GFILL(NF,EGEOM(1,II),G)
      DO 510 I=1,NF3
      C = 0.D0
      DO 505 M=1,NF2
  505 C = C + G(M,I)*STORM(M)
  510 P(I) = C*V
      WRITE(6,1901)(P(I),I=1,NF3)
      IF(IPRESS.EQ.0)GO TO 1000
C     ADD NONCONSERVATIVE PRESSURE FORCES TO P
      VDOT(1) = QDOT(4) - QDOT(1)
      VDOT(2) = QDOT(5) - QDOT(2)
      VDOT(3) = QDOT(6) - QDOT(3)
      VDOT(4) = QDOT(7) - QDOT(1)
      VDOT(5) = QDOT(8) - QDOT(2)
      VDOT(6) = QDOT(9) - QDOT(3)
      VPOST(1) = QPOST(4) - QPOST(1)
      VPOST(2) = QPOST(5) - QPOST(2)
      VPOST(3) = QPOST(6) - QPOST(3)
      VPOST(4) = QPOST(7) - QPOST(1)
      VPOST(5) = QPOST(8) - QPOST(2)
      VPOST(6) = QPOST(9) - QPOST(3)
      C0 = (PR0 + LCRIT*(PR1-PR0))*PRREF(II)/6.D0
      C1 = 0.D0
      CALL VCROSS(VDOT,VPOST(4),V1)
      CALL VCROSS(VPOST,VDOT(4),V2)
      IF(KODE.EQ.2)GO TO 601
      VCRIT(1) = EGEOM(1,II) + QCRIT(4) - QCRIT(1)
      VCRIT(2) = QCRIT(5) - QCRIT(2)
      VCRIT(3) = QCRIT(6) - QCRIT(3)
      VCRIT(4) = EGEOM(3,II) + QCRIT(7) - QCRIT(1)
      VCRIT(5) = EGEOM(2,II) + QCRIT(8) - QCRIT(2)
      VCRIT(6) = QCRIT(9) - QCRIT(3)
      C1 = LDOT*(PR1-PRO)*PRREF(II)/6.D0
      CALL VCROSS(VCRIT,VPOST(4),V3)
      CALL VCROSS(VPOST,VCRIT(4),V4)
  601 DO 610 I=1,3
      C = C0*(V1(I)+V2(I)) + C1*(V3(I)+V4(I))
      P(I) = P(I) - C
      P(I+3) = P(I+3) - C
  610 P(I+6) = P(I+6) - C
C     ADD ELEMENT FORCES P TO SYSTEM FORCES PP
 1000 CALL PFILL(NF,ELNO(1,II),P,PP)
      RETURN
      END


      SUBROUTINE POST3(NN,QQCRIT,QQDOT1,QQPOS1,QQDOT2,QQPOS2,SPOST1,
     1LCRIT,LDOT1,LDOT2,PRO,PR1,PP)
C     FORM 3RD ORDER POSTBUCKLING LOAD TERM PP.
C     NN = TOTAL SYSTEM DOF.
C     QQCRIT(I) = PREDICTED TOTAL DISPLACEMENT OF DOF I AT CRITICAL PT.
C     QQDOT1(I) = CRITICAL 1ST ORDER DISPLACEMENT RATE AT DOF I.
C     QQPOS1(I) = ITH EIGENVECTOR VALUE (1ST ORDER POSTBUCKLING RATE).
C     QQDOT2(I) = CRITICAL 2ND ORDER DISPLACEMENT RATE AT DOF I.
C     QQPOS2(I) = ITH VALUE OF 2ND ORDER POSTBUCKLING DISPLACEMENT.
C     SPOST1 = 1ST ORDER FUNDAMENTAL PATH RATE.
C     LCRIT = PREDICTED LOAD PARAMETER VALUE AT CRITICAL POINT.
C     LDOT1 = 1ST ORDER LOAD PARAMETER RATE AT CRITICAL POINT.
C     LDOT2 = 2ND ORDER LOAD PARAMETER RATE AT CRITICAL POINT.
C     PRO,PR1 = PRESSURE FACTORS AT START OF LOAD STEP,CRITICAL POINT.
C     PP(I) = COMPUTED PSUEDO FORCE TERM AT DOF I.
C     ORD = MAXIMUM TENSOR ORDER TO BE USED FOR DIFFERENCE EXPANSION.
C     NEL,NS,NF = NUMBER OF ELEMENTS,STRAIN COMPONENTS,DOF PER NODE.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     EGEOM(J,I) = GEOMETRIC PROPERTIES FOR ELEMENT I.
C     T(I) = THICKNESS OF ELEMENT I.
C     IPRESS = NONCONSERVATIVE CODE = 0,1 FOR NO PRESSURE,PRESSURE.
C     PRREF(I) = PRESSURE INTENSITY ON ELEMENT I FOR PRESSURE VECTOR.
      COMMON/COMORD/ORD/COMNEL/NEL/COMNS/NS/COMNF/NF
      COMMON/COMEL/ELNO/COMEG/EGEOM/COMT/T
      COMMON/COMIPR/IPRESS/COMPRR/PRREF
      INTEGER NN,ORD,NEL,NS,NF,ELNO(3,1),IPRESS
      DOUBLE PRECISION QQCRIT(1),QQDOT1(1),QQPOS1(1),QQDOT2(1),
     1QQPOS2(1),SPOST1,LCRIT,LDOT1,LDOT2,PRO,PR1,PP(1),EGEOM(3,1),T(1),
     2PRREF(1)
      INTEGER NF2,NF3,IU2,IU3,I,II,M
      DOUBLE PRECISION V,C,QCRIT(9),QDOT1(9),QPOST1(9),QDOT2(9),
     1QPOST2(9),DCRIT(6),DDOT1(6),DPOST1(6),DDOT2(6),DPOST2(6),
     2ECRIT(3),EDOT1(3),EPOST1(3),EDOT2(3),EPOST2(3),ACRIT(3,6),
     3A1DOT1(3,6),A1POS1(3,6),A1DOT2(3,6),A1POS2(3,6),STORA(3),STORB(3),
     4STORM(6),G(6,9),P(9),UCRIT(20)
      DOUBLE PRECISION C0,C1,C2,VCRIT(6),VDOT1(6),VPOST1(6),VDOT2(6),
     1VPOST2(6),V1(3),V2(3),V3(3),V4(3),V5(3),V6(3),V7(3),V8(3),V9(3),
     2V10(3),V11(3),V12(3),V13(3)
      NF2 = NF*2
      NF3 = NF*3
      IU2 = 2 + NS
      IU3 = IU2 + NS*(NS+1)/2
      DO 5 I=1,NN
    5 PP(I) = 0.D0
      DO 1000 II=1,NEL
      V = .5D0*T(II)*EGEOM(1,II)*EGEOM(2,II)
C     FORM FUNDAMENTAL CRITICAL QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQCRIT,QCRIT)
      CALL DFILL(NF,EGEOM(1,II),QCRIT,DCRIT)
      CALL AFILL(NF,DCRIT,ACRIT,1)
      CALL EFILL(NF,DCRIT,ECRIT)
C     FORM STRESS-STRAIN TENSORS
      CALL UFILL(II,ORD,ECRIT,UCRIT)
C     FORM 1ST ORDER FUNDAMENTAL RATE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQDOT1,QDOT1)
      CALL DFILL(NF,EGEOM(1,II),QDOT1,DDOT1)
      CALL AFILL(NF,DDOT1,A1DOT1,0)
      DO 20 I=1,NS
      C = 0.D0
      DO 15 M=1,NF2
   15 C = C + ACRIT(I,M)*DDOT1(M)
   20 EDOT1(I) = C
C     FORM 1ST ORDER POSTBUCKLING RATE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQPOS1,QPOST1)
      CALL DFILL(NF,EGEOM(1,II),QPOST1,DPOST1)
      CALL AFILL(NF,DPOST1,A1POS1,0)
      DO 30 I=1,NS
      C = 0.D0
      DO 25 M=1,NF2
   25 C = C + ACRIT(I,M)*DPOST1(M)
   30 EPOST1(I) = C
C     FORM 2ND ORDER FUNDAMENTAL RATE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQDOT2,QDOT2)
      CALL DFILL(NF,EGEOM(1,II),QDOT2,DDOT2)
      CALL AFILL(NF,DDOT2,A1DOT2,0)
      DO 40 I=1,NS
      C = 0.D0
      DO 35 M=1,NF2
   35 C = C + ACRIT(I,M)*DDOT2(M) + A1DOT1(I,M)*DDOT1(M)
   40 EDOT2(I) = C
C     FORM 2ND ORDER POSTBUCKLING RATE QUANTITIES
      CALL QFILL(NF,ELNO(1,II),QQPOS2,QPOST2)
      CALL DFILL(NF,EGEOM(1,II),QPOST2,DPOST2)
      CALL AFILL(NF,DPOST2,A1POS2,0)
      DO 50 I=1,NS
      C = 0.D0
      DO 45 M=1,NF2
   45 C = C + ACRIT(I,M)*DPOST2(M) + A1POS1(I,M)*(DPOST1(M) + 2.D0*
     1SPOST1*DDOT1(M))
   50 EPOST2(I) = C
C     FORM STORM = DO PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES
      DO 104 I=1,NS
      C = 0.D0
      DO 102 M=1,NF2
  102 C = C + A1POS1(I,M)*DDOT2(M)
  104 STORB(I) = C
      CALL USUM1(2,UCRIT(IU2),NS,STORB,STORA)
      DO 110 I=1,NF2
      C = 0.D0
      DO 105 M=1,NS
  105 C = C + STORA(M)*ACRIT(M,I)
  110 STORM(I) = SPOST1**2*C
      CALL USUM1(2,UCRIT(IU2),NS,EPOST1,STORA)
      DO 120 I=1,NF2
      C = 0.D0
      DO 115 M=1,NS
  115 C = C + STORA(M)*A1DOT2(M,I)
  120 STORM(I) = STORM(I) + SPOST1**2*C
      CALL USUM1(2,UCRIT(IU2),NS,EDOT2,STORA)
      DO 130 I=1,NF2
      C = 0.D0
      DO 125 M=1,NS
  125 C = C + STORA(M)*A1POS1(M,I)
  130 STORM(I) = STORM(I) + SPOST1**2*C
      DO 134 I=1,NS
      C = 0.D0
      DO 132 M=1,NF2
  132 C = C + A1DOT1(I,M)*DPOST2(M)
  134 STORB(I) = C
      CALL USUM1(2,UCRIT(IU2),NS,STORB,STORA)
      DO 140 I=1,NF2
      C = 0.D0
      DO 135 M=1,NS
  135 C = C + STORA(M)*ACRIT(M,I)
  140 STORM(I) = STORM(I) + SPOST1*C
      CALL USUM1(2,UCRIT(IU2),NS,EDOT1,STORA)
      DO 150 I=1,NF2
      C = 0.D0
      DO 145 M=1,NS
  145 C = C + STORA(M)*A1POS2(M,I)
  150 STORM(I) = STORM(I) + SPOST1*C
      CALL USUM1(2,UCRIT(IU2),NS,EPOST2,STORA)
      DO 160 I=1,NF2
      C = 0.D0
      DO 155 M=1,NS
  155 C = C + STORA(M)*A1DOT1(M,I)
  160 STORM(I) = STORM(I) + SPOST1*C
      DO 164 I=1,NS
      C = 0.D0
      DO 162 M=1,NF2
  162 C = C + A1POS1(I,M)*DPOST2(M)
  164 STORB(I) = C
      CALL USUM1(2,UCRIT(IU2),NS,STORB,STORA)
      DO 170 I=1,NF2
      C = 0.D0
      DO 165 M=1,NS
  165 C = C + STORA(M)*ACRIT(M,I)
  170 STORM(I) = STORM(I) + C
      CALL USUM1(2,UCRIT(IU2),NS,EPOST1,STORA)
      DO 180 I=1,NF2
      C = 0.D0
      DO 175 M=1,NS
  175 C = C + STORA(M)*A1POS2(M,I)
  180 STORM(I) = STORM(I) + C
      CALL USUM1(2,UCRIT(IU2),NS,EPOST2,STORA)
      DO 190 I=1,NF2
      C = 0.D0
      DO 185 M=1,NS
  185 C = C + STORA(M)*A1POS1(M,I)
  190 STORM(I) = STORM(I) + C
      IF(ORD.LT.3)GO TO 501
C     ADD D1 PORTION OF DISPLACEMENT DERIVATIVE PSUEDO FORCES TO STORM
      CALL USUM21(UCRIT(IU3),NS,EPOST1,EDOT2,STORA)
      DO 210 I=1,NF2
      C = 0.D0
      DO 205 M=1,NS
  205 C = C + STORA(M)*ACRIT(M,I)
  210 STORM(I) = STORM(I) + SPOST1**2*C
      CALL USUM1(3,UCRIT(IU3),NS,EDOT1,STORA)
      DO 220 I=1,NF2
      C = 0.D0
      DO 215 M=1,NS
  215 C = C + STORA(M)*A1POS1(M,I)
  220 STORM(I) = STORM(I) + SPOST1**2*C
      CALL USUM21(UCRIT(IU3),NS,EPOST1,EDOT1,STORA)
      DO 230 I=1,NF2
      C = 0.D0
      DO 225 M=1,NS
  225 C = C + STORA(M)*A1DOT1(M,I)
  230 STORM(I) = STORM(I) + 2.D0*SPOST1**2*C
      CALL USUM21(UCRIT(IU3),NS,EDOT1,EPOST2,STORA)
      DO 240 I=1,NF2
      C = 0.D0
      DO 235 M=1,NS
  235 C = C + STORA(M)*ACRIT(M,I)
  240 STORM(I) = STORM(I) + SPOST1*C
```

```
      CALL USUM1(3,UCRIT(IU3),NS,EPOST1,STORA)
      DO 250 I=1,NF2
      C = 0.D0
      DC 245 M=1,NS
  245 C = C + STORA(M)*A1DOT1(M,I)
  250 STORM(I) = STORM(I) + SPOST1*C
      CALL USUM21(UCRIT(IU3),NS,EDOT1,EPOST1,STORA)
      DO 260 I=1,NF2
      C = 0.D0
      DO 255 M=1,NS
  255 C = C + STORA(M)*A1POS1(M,I)
  260 STORM(I) = STORM(I) + 2.D0*SPOST1*C
      CALL USUM21(UCRIT(IU3),NS,EPOST1,EPOST2,STORA)
      DO 270 I=1,NF2
      C = 0.D0
      DO 265 M=1,NS
  265 C = C + STORA(M)*ACRIT(M,I)
  270 STORM(I) = STORM(I) + C
      CALL USUM1(3,UCRIT(IU3),NS,EPOST1,STORA)
      DO 280 I=1,NF2
      C = 0.D0
      DO 275 M=1,NS
  275 C = C + STORA(M)*A1POS1(M,I)
  280 STORM(I) = STORM(I) + C
C     COMPUTE ELEMENT FORCES P(I) = VOLUME INTEGRAL OF G(M,I)*STORM(M)
  501 CALL GFILL(NF,EGEOM(1,II),G)
      DO 510 I=1,NF3
      C = 0.D0
      DO 505 M=1,NF2
  505 C = C + G(M,I)*STORM(M)
  510 P(I) = C*V
      IF(IPRESS.EQ.0)GO TO 1000
C     ADD NONCONSERVATIVE PRESSURE FORCES TO P
      VCRIT(1) = EGEOM(1,II) + QCRIT(4) - QCRIT(1)
      VCRIT(2) = QCRIT(5) - QCRIT(2)
      VCRIT(3) = QCRIT(6) - QCRIT(3)
      VCRIT(4) = EGEOM(3,II) + QCRIT(7) - QCRIT(1)
      VCRIT(5) = EGEOM(2,II) + QCRIT(8) - QCRIT(2)
      VCRIT(6) = QCRIT(9) - QCRIT(3)
      VDOT1(1) = QDOT1(4) - QDOT1(1)
      VDOT1(2) = QDOT1(5) - QDOT1(2)
      VDOT1(3) = QDOT1(6) - QDOT1(3)
      VDOT1(4) = QDOT1(7) - QDOT1(1)
      VDOT1(5) = QDOT1(8) - QDOT1(2)
      VDOT1(6) = QDOT1(9) - QDOT1(3)
      VPOST1(1) = QPOST1(4) - QPOST1(1)
      VPOST1(2) = QPOST1(5) - QPOST1(2)
      VPOST1(3) = QPOST1(6) - QPOST1(3)
      VPOST1(4) = QPOST1(7) - QPOST1(1)
      VPOST1(5) = QPOST1(8) - QPOST1(2)
      VPOST1(6) = QPOST1(9) - QPOST1(3)
      VDOT2(1) = QDOT1(4) - QDOT1(1)
      VDOT2(2) = QDOT1(5) - QDOT1(2)
      VDOT2(3) = QDOT1(6) - QDOT1(3)
      VDOT2(4) = QDOT1(7) - QDOT1(1)
      VDOT2(5) = QDOT1(8) - QDOT1(2)
      VDOT2(6) = QDOT1(9) - QDOT1(3)
      VPOST2(1) = QPOST2(4) - QPOST2(1)
      VPOST2(2) = QPOST2(5) - QPOST2(2)
      VPOST2(3) = QPOST2(6) - QPOST2(3)
      VPOST2(4) = QPOST2(7) - QPOST2(1)
      VPOST2(5) = QPOST2(8) - QPOST2(2)
      VPOST2(6) = QPOST2(9) - QPOST2(3)
      C0 = (PR0 + LCRIT*(PR1-PR0))*PRREF(II)/6.D0
      C1 = LDOT1*(PR1-PR0)*PRREF(II)/6.D0
      C2 = LDOT2*(PR1-PR0)*PRREF(II)/6.D0
      CALL VCROSS(VDOT2,VPOST1(4),V1)
      CALL VCROSS(VPOST1,VDOT2(4),V2)
      CALL VCROSS(VDOT1,VPOST1(4),V3)
      CALL VCROSS(VPOST1,VDOT1(4),V4)
      CALL VCROSS(VCRIT,VPOST1(4),V5)
      CALL VCROSS(VPOST1,VCRIT(4),V6)
      CALL VCROSS(VDOT1,VPOST2(4),V7)
      CALL VCROSS(VPOST2,VDOT1(4),V8)
      CALL VCROSS(VCRIT,VPOST2(4),V9)
      CALL VCROSS(VPOST2,VCRIT(4),V10)
      CALL VCROSS(VPOST1,VPOST1(4),V11)
      CALL VCROSS(VPOST2,VPOST1(4),V12)
      CALL VCROSS(VPOST1,VPOST2(4),V13)
      DO 610 I=1,3
      C = C0*(V1(I)+V2(I)+V7(I)+V8(I)+V12(I)+V13(I)) + C1*(2.D0*V3(I)
     1+2.D0*V4(I)+V9(I)+V10(I)+2.D0*V11(I)) + C2*(V5(I)+V6(I))
      P(I) = P(I) - C
      P(I+3) = P(I+3) - C
  610 P(I+6) = P(I+6) - C
C     ADD ELEMENT FORCES P TO SYSTEM FORCES PP
 1000 CALL PFILL(NF,ELNO(1,II),P,PP)
      RETURN
      END


      SUBROUTINE PRATES(KMAT,P1,Q1,P2,Q2,NN,IPOST,KFD,SCRIT,
     1QQCRIT,QQDOT1,QQDOT2,LCRIT,LDOT1,LDOT2,PR0,PR1,
     2QQPOS1,LPOST1,QQPOS2,LPOST2,QQPOS3)
C     COMPUTE POSTBUCKLING RATES FOR LOAD PARAMETER AND DISPLACEMENTS.
C     KMAT = SYSTEM JACOBIAN STIFFNESS MATRIX.
C     P1,Q1,P2,Q2 = TEMPORARY STORAGE VECTORS.
C     NN = TOTAL SYSTEM DOF.
C     IPOST = DOF OF LARGEST EIGENVECTOR COMPONENT.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     SCRIT = PREDICTED CRITICAL VALUE OF FUNDAMENTAL PATH.
C     QQCRIT(I) = PREDICTED CRITICAL VALUE OF DISPLACEMENT AT DOF I.
C     QQDOT1(I),QQDOT2(I) = 1ST,2ND ORDER CRITICAL FUNDAMENTAL
C       DISPLACEMENT RATE AT DOF I.
C     LCRIT = PREDICTED CRITICAL VALUE OF LOAD PARAMETER.
C     LDOT1,LDOT2 = 1ST,2ND ORDER CRITICAL LOAD PARAMETER RATES.
C     PR0,PR1 = PRESSURE FACTORS AT START OF LOAD STEP,CRITICAL POINT.
C     LPOST1,LPOST2 = COMPUTED 1ST,2ND ORDER POSTBUCKLING LOAD RATES.
C     QQPOS1(I),QQPOS2(I),QQPOS3(I) = 1ST,2ND,3RD ORDER POSTBUCKLING
C       DISPLACEMENT RATES AT DOF I.
C     NF = DOF PER NODE.
      COMMON/COMNF/NF
      INTEGER NN,IPOST,KFD(1)
      DOUBLE PRECISION KMAT(NN,NN),P1(1),Q1(1),P2(1),Q2(1),SCRIT,
     1QQCRIT(1),QQDOT1(1),QQDOT2(1),LCRIT,LDOT1,LDOT2,PR0,PR1,
     2QQPOS1(1),LPOST1,QQPOS2(1),LPOST2,QQPOS3(1)
      INTEGER I
      DOUBLE PRECISION CN,C0,SPOST1,SPOST2
C     SOLVE FOR 1ST ORDER DISPLACEMENTS
      DO 10 I=1,NN
```

```
   10 P1(I) = 0.D0
      P1(IPOST) = 1.D0
      CALL SOLVE(KMAT,NN,KFD,P1,QQPOS1)
      NOD = NN/NF
      CALL OUTPQ(6,NOD,NF,P1,QQPOS1)
C     SOLVE FOR 1ST ORDER LOADS AND 2ND ORDER DISPLACEMENTS
      CALL POST2(NN,QQCRIT,QQDOT1,QQPOS1,LCRIT,LDOT1,PRO,PR1,P1,1)
      CALL POST2(NN,QQCRIT,QQPOS1,QQPOS1,LCRIT,LDOT1,PRO,PR1,P2,2)
      CALL OUTPQ(6,NOD,NF,P1,P2)
      CN = 0.D0
      CD = 0.D0
      DO 110 I=1,NN
      CN = CN + QQPOS1(I)*P2(I)
  110 CD = CD + QQPOS1(I)*P1(I)
      SPOST1 = -.5D0*CN/CD
      LPOST1 = LDOT1*SPOST1
      DO 120 I=1,NN
      IF(KFD(I).GT.0)GO TO 120
      P1(I) = 0.D0
      P2(I) = 0.D0
  120 CONTINUE
      CALL SOLVE(KMAT,NN,KFD,P1,Q1)
      CALL SOLVE(KMAT,NN,KFD,P2,Q2)
      CALL OUTPQ(6,NOD,NF,P1,P2)
      CALL OUTPQ(6,NOD,NF,Q1,Q2)
      DO 130 I=1,NN
  130 QQPOS2(I) = -(2.D0*SPOST1*Q1(I) + Q2(I))
      CALL OUTPQ(6,NOD,NF,QQDOT2,QQPOS2)
C     SOLVE FOR 2ND ORDER LOADS AND 3RD ORDER DISPLACEMENTS
      CALL POST3(NN,QQCRIT,QQDOT1,QQPOS1,QQDOT2,QQPOS2,SPOST1,
     1LCRIT,LDOT1,LDOT2,PRO,PR1,P2)
      CALL OUTPQ(6,NOD,NF,QQCRIT,P2)
      CALL OUTPQ(6,NOD,NF,QQDOT1,QQPOS1)
      CN = 0.D0
      DO 210 I=1,NN
  210 CN = CN + QQPOS1(I)*P2(I)
      SPOST2 = -CN/CD
      LPOST2 = LDOT1*SPOST2 + LDOT2*SPOST1**2
      DO 220 I=1,NN
      IF(KFD(I).GT.0)GO TO 220
      P2(I) = 0.D0
  220 CONTINUE
      CALL SOLVE(KMAT,NN,KFD,P2,Q2)
      CALL OUTPQ(6,NOD,NF,P2,Q2)
      DO 230 I=1,NN
  230 QQPOS3(I) = -3.D0*(SPOST2*Q1(I) + Q2(I))
      CALL OUTPQ(6,NOD,NF,QQPOS3,QQPOS3)
      RETURN
      DEBUG INIT(SPOST1,LPOST1,SPOST2,LPOST2,CN,CD)
      END


      DOUBLE PRECISION FUNCTION VDOT(V1,V2)
C     COMPUTE VECTOR DOT PRODUCT VDOT = V1 DOT V2.
      DOUBLE PRECISION V1(1),V2(1)
      VDOT = V1(1)*V2(1) + V1(2)*V2(2) + V1(3)*V2(3)
      RETURN
      END


      SUBROUTINE VCROSS(V1,V2,VV)
C     COMPUTE VECTOR CROSS PRODUCT V1 X V2 = VV.
      DOUBLE PRECISION V1(1),V2(1),VV(1)
      VV(1) = V1(2)*V2(3) - V1(3)*V2(2)
      VV(2) = V1(3)*V2(1) - V1(1)*V2(3)
      VV(3) = V1(1)*V2(2) - V1(2)*V2(1)
      RETURN
      END


      DOUBLE PRECISION FUNCTION VLENTH(V)
C     COMPUTE VLENTH = LENGTH OF VECTOR V.
      DOUBLE PRECISION V(1)
      VLENTH = DSQRT(V(1)**2 + V(2)**2 + V(3)**2)
      RETURN
      END


      SUBROUTINE VNORM(V,VV)
C     COMPUTE NORMALIZED UNIT VECTOR VV FROM GIVEN VECTOR V.
```

```
      DOUBLE PRECISION V(1),VV(1),C
      C = DSQRT(V(1)**2 + V(2)**2 + V(3)**2)
      IF(C.LE.0.D0)C = 1.D0
      VV(1) = V(1)/C
      VV(2) = V(2)/C
      VV(3) = V(3)/C
      RETURN
      END


      SUBROUTINE MERGE(KMAT,ELNO,KFD,NEL,NN,NF)
C     GENERATE AND MERGE ELEMENTAL MATRICES INTO GLOBAL MATRIX KMAT.
C     ASSUMES ELEMENTS HAVE 3 NODES, MATRICES MAY BE UNSYMMETRIC.
C     KMAT = GLOBAL (TOTAL SYSTEM) MATRIX, WHICH MAY BE UNSYMMETRIC.
C     ELNO(J,I) = NODE NUMBERS FOR ELEMENT I.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     KFD(I) = +,- = SPECIFIED FORCE,DISPLACEMENT.
C     KFD(I) = +-I INDICATES INDEPENDENT DOF.
C     KFD(I) = +-J INDICATES DEPENDENT FREEDOM WHOSE DISPLACEMENT IS
C       CONSTRAINED TO EQUAL DISPLACEMENT AT J.
C     NEL = NUMBER OF ELEMENTS.
C     NN = TOTAL SYSTEM DOF = NUMBER OF NODES TIMES NF.
C     NF = DOF PER NODE.
C     GENER8 ROUTINE GENERATES ELEMENTAL MATRIX.
      INTEGER ELNO(3,1),KFD(1),NEL,NN,NF
      DOUBLE PRECISION KMAT(NN,NN)
      INTEGER IE,I,J,II,JJ,IO,JO,IIO,JJO,IP,JP,IN,JN,ILOC,IILOC
      DOUBLE PRECISION K(9,9)
C     ZERO OUT GLOBAL MATRIX
      DO 5 II=1,NN
      DO 5 JJ=1,NN
    5 KMAT(II,JJ) = 0.D0
      DO 100 IE=1,NEL
      CALL GENER8(IE,K)
C     IP,JP ARE PARTITION ROW,COLUMN NUMBERS.
      DO 100 IP=1,3
      IN = ELNO(IP,IE)
      IO = NF*IP - NF
      IIO = NF*IN - NF
      DO 100 JP=1,3
      JN = ELNO(JP,IE)
      JO = NF*JP - NF
      JJO = NF*JN - NF
C     MERGE PARTITION INTO GLOBAL MATRIX
      DO 50 I=1,NF
      DO 50 J=1,NF
   50 KMAT(IIO+I,JJO+J) = KMAT(IIO+I,JJO+J) + K(IO+I,JO+J)
  100 CONTINUE
C     ADD CONSTRAINED ROWS AND COLUMNS TO INDEPENDENT ROWS AND COLUMNS
      DO 200 ILOC=1,NN
      IILOC = KFD(ILOC)
      IF(IILOC.LT.0)IILOC = -IILOC
      IF(IILOC.EQ.ILOC)GO TO 200
      DO 180 I=1,NN
  180 KMAT(I,IILOC) = KMAT(I,IILOC) + KMAT(I,ILOC)
      DO 190 J=1,NN
  190 KMAT(IILOC,J) = KMAT(IILOC,J) + KMAT(ILOC,J)
  200 CONTINUE
      RETURN
      END


      SUBROUTINE DECOMP(KMAT,NN,KFD,IDET,DET)
C     DECOMPOSE GLOBAL STIFFNESS MATRIX KMAT, AND COMPUTE DETERMINANT.
C     KMAT = GLOBAL (TOTAL SYSTEM) MATRIX, WHICH MAY BE UNSYMMETRIC.
C     NN = TOTAL SYSTEM DOF.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     KFD(I) = +,- = SPECIFIED FORCE,DISPLACEMENT.
C     KFD(I) = +-I INDICATES INDEPENDENT DOF.
C     KFD(I) = +-J INDICATES DEPENDENT FREEDOM WHOSE DISPLACEMENT IS
C       CONSTRAINED TO EQUAL DISPLACEMENT AT J.
C     IDET,DET = SIGN,LOGARITHM (BASE 10) OF DETERMINANT.
      INTEGER NN,KFD(1),IDET
      DOUBLE PRECISION KMAT(NN,NN),DET
      INTEGER NM1,II,JJ1,JJ2,I,J
      DOUBLE PRECISION D,C
      IF(NN.EQ.1)RETURN
      NM1 = NN-1
      DO 200 II=1,NM1
      IF(KFD(II).NE.II)GO TO 200
      D = 1.DO/KMAT(II,II)
      JJ1 = II+1
C     DISTRIBUTE IITH ROW OF UPPER TRIANGULAR MATRIX
      DC 90 J=JJ1,NN
      C = D*KMAT(II,J)
      IF(C.EQ.0.DO)GO TO 90
C     DISTRIBUTE KMAT(II,J) TO JTH COLUMN
      DC 50 I=JJ1,J
   50 KMAT(I,J) = KMAT(I,J) - C*KMAT(I,II)
   90 CONTINUE
C     CISTRIBUTE IITH COLUMN CF LOWER TRIANGULAR MATRIX
      JJ2 = II+2
      IF(JJ2.GT.NN)GO TO 200
      DC 190 J=JJ2,NN
      C = D*KMAT(J,II)
      IF(C.EQ.0.DO)GO TO 190
C     DISTRIBUTE KMAT(J,II) TC JTH ROW
      JM1 = J-1
      DO 150 I=JJ1,JM1
  150 KMAT(J,I) = KMAT(J,I) - C*KMAT(II,I)
  190 CCNTINUE
  200 CONTINUE
C     COMPUTE DETERMINANT
      ICET = 1
      DET = 0.DO
      DO 500 II=1,NN
      IF(KFD(II).NE.II)GO TO 500
      C = KMAT(II,II)
      IF(D.LT.C.DO)IDET = -IDET
      DET = DET + DLOG10(DABS(D))
  500 CCNTINUE
 1201 FORMAT(1HO,'JACOBIAN DETERMINANT SIGN,LOGARITHM =',I5,D15.7)
      WRITE(6,1201)IDET,DET
      RETURN
      END

      SUBROUTINE SOLVE(KMAT,NN,KFD,P,Q)
C     FCRWARC-BACK SUBSTITUTE TO SOLVE FCR FORCES P AND DISPLACEMENTS Q.
C     KMAT = DECOMPOSED SYSTEM JACOBIAN MATRIX (MAY BE UNSYMMETRIC).
```

```
C     NN = TOTAL SYSTEM DOF.
C     KFD(I) = FORCE-DISPLACEMENT-CONSTRAINT SPECIFICATION FOR DOF I.
C     KFD(I) = +,- = SPECIFIED FORCE,DISPLACEMENT.
C     KFD(I) = +-I INDICATES INDEPENDENT DOF.
C     KFD(I) = +-J INDICATES DEPENDENT FREEDOM WHOSE DISPLACEMENT IS
C       CONSTRAINED TO EQUAL DISPLACEMENT AT J.
C     P(I),Q(I) = FORCE,DISPLACEMENT AT DOF I.
C     EACH DOF HAS EITHER SPECIFIED FORCE OR DISPLACEMENT.
C     INPUT P = SPECIFIED FORCES AND DISPLACEMENTS, Q = GARBAGE.
C     OUTPUT P = SPECIFIED AND COMPUTED FORCES, Q = SPECIFIED AND
C       COMPUTED DISPLACEMENTS.
      INTEGER NN,KFD(1)
      DOUBLE PRECISION KMAT(NN,NN),P(1),Q(1)
      INTEGER I,J,I1,IDUM
      DOUBLE PRECISION C
C          CORRELATE DEPENDENT-INDEPENDENT DOFS, ZERO OUT Q
      DO 5 I=1,NN
      J = IABS(KFD(I))
      IF(J.EQ.I)GO TO 5
      IF(KFD(J).GT.0)P(J) = P(J) + P(I)
      P(I) = 0.D0
    5 Q(I) = 0.D0
C          FORWARD SUBSTITUTION
      DO 100 I=1,NN
      IF(KFD(I).NE.I)GO TO 31
      C = (Q(I)+P(I))/KMAT(I,I)
      Q(I) = C
      GC TO 41
   31 C = P(I)
      P(I) = -Q(I)
      Q(I) = C
   41 IF(I.EQ.NN)GO TO 1C0
      I1 = I+1
      DC 50 J=I1,NN
   50 Q(J) = Q(J) - KMAT(J,I)*C
  100 CONTINUE
C          BACKWARD SUBSTITUTION
      I = NN+1
      DO 200 IDUM=1,NN
      I = I-1
      C = 0.D0
      IF(KFD(I).NE.I)GO TO 131
      IF(I.EQ.NN)GO TO 200
      I1 = I+1
      DO 120 J=I1,NN
  120 C = C + KMAT(I,J)*Q(J)
      Q(I) = Q(I) - C/KMAT(I,I)
      GO TO 200
  131 DO 140 J=I,NN
  140 C = C + KMAT(I,J)*Q(J)
      P(I) = P(I) + C
  200 CONTINUE
C           SET DEPENDENT DISPLACEMENTS, AND ADJUST INDEPENDENT FORCES
      DO 205 I=1,NN
      J = IABS(KFD(I))
      IF(J.EQ.I)GO TO 205
      Q(I) = Q(J)
      P(J) = P(J) - P(I)
  205 CONTINUE
      RETURN
      END
```

```
START           5    5    6
TORUS MESH-12 (RR=10, R=2, T=.05) MOONEY MATERIAL, C1=80, C2=20  R. VOS
         3         2          1          5     1.D-6
     1.D-3     1.D-6
         2 .01       1.0        0.1        -10.0
         1          3 .05
         2 80.0      20.0


    1    1 8.0       0.0        0.0           1
    2    1 8.0       2.0        0.0           1
    3    1 8.06814830.0         .517631257    1
    4    1 8.06814832.0         .517631257    1
    5    1 8.26794920.0         1.0           1
    6    1 8.26794922.0         1.0           1
    7    1 8.58578640.0         1.41421356    1
    8    1 8.58578642.0         1.41421356    1
```

```
  9     1 9.0         0.0        1.73205081     1
 10     1 9.0         2.0        1.73205081     1
 11     1 9.48236870.0           1.93185165     1
 12     1 9.48236872.0           1.93185165     1
 13     110.0         0.0        2.0            1
 14     110.0         2.0        2.0            1
 15     110.51763130.0           1.93185165     1
 16     110.51763132.0           1.93185165     1
 17     111.0         0.0        1.73205081     1
 18     111.0         2.0        1.73205081     1
 19     111.41421360.0           1.41421356     1
 20     111.41421362.0           1.41421356     1
 21     111.73205080.0           1.0            1
 22     111.73205082.0           1.0            1
 23     111.93185170.0           .517631257     1
 24     111.93185172.0           .517631257     1
 25     112.0         0.0        0.0            1
 26     112.0         2.0        0.0            1

  1     2                   2       1       3
  2     2                   3       4       2
  3     2                   4       3       6
  4     2                   5       6       3
  5     2                   6       5       7
  6     2                   7       8       6
  7     2                   8       7      10
  8     2                   9      10       7
  9     2                  10       9      11
 10     2                  11      12      10
 11     2                  12      11      14
 12     2                  13      14      11
 13     2                  14      13      15
 14     2                  15      16      14
 15     2                  16      15      18
 16     2                  17      18      15
 17     2                  18      17      19
 18     2                  19      20      18
 19     2                  20      19      22
 20     2                  21      22      19
 21     2                  22      21      23
```

```
   22    2              23   24   22
   23    2              24   23   26
   24    2              25   26   23

    1    3               2    3                  25    3                  26    3
    1    2               3    2                   5    2                   7    2
    9    2              11    2                  13    2                  15    2
   17    2              19    2                  21    2                  23    2
   25    2               2    2                   4    2                   6    2
    8    2              10    2                  12    2                  14    2
   16    2              18    2                  20    2                  22    2
   24    2              26    2
    2    1    1    1                              4    1    3    1    4    3    3    3
    6    1    5    1    6    3    5    3    8    1    7    1    8    3    7    3
   10    1    9    1   10    3    9    3   12    1   11    1   12    3   11    3
   14    1   13    1   14    3   13    3   16    1   15    1   16    3   15    3
   18    1   17    1   18    3   17    3   20    1   19    1   20    3   19    3
   22    1   21    1   22    3   21    3   24    1   23    1   24    3   23    3
   26    1   25    1

         2


         1
         1  1.0                   2  1.0                   3  1.0                   4  1.0
         5  1.0                   6  1.0                   7  1.0                   8  1.0
         9  1.0                  10  1.0                  11  1.0                  12  1.0
        13  1.0                  14  1.0                  15  1.0                  16  1.0
        17  1.0                  18  1.0                  19  1.0                  20  1.0
        21  1.0                  22  1.0                  23  1.0                  24  1.0

         8
                              0.1
                              0.5
                              1.0
                              2.0
                              3.0
                              4.0
                              4.5
                              5.0
```