ANALYSIS OF SHELL-TYPE STRUCTURES SUBJECTED TO TIME-DEPENDENT MECHANICAL AND THERMAL LOADING

A SEMI-ANNUAL STATUS REPORT
SUBMITTED TO
NASA-LEWIS RESEARCH CENTER
CLEVELAND, OHIO

By

G. J. Simitses and R. Riff

School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, Georgia

October, 1988

(NASA GRANT: NAG 3-534)

INTRODUCTION

The objective of the present research is to develop a general mathematical model and solution methodologies for analyzing structural response of thin, metallic shell-type structures under large transient, cyclic or static thermomechanical loads. Among the system responses, which are associated with these load conditions, are thermal buckling, creep buckling and racheting. Thus, geometric as well as material-type nonlinearities (of high order) can be anticipated and must be considered in the development of the mathematical model. Furthermore, this must also be accommodated in the solution procedures.

SUMMARY OF PROGRESS

The progress to date has been elaborated upon in an interim scientific report submitted to the sponsor during the summer of 1986, and in a series of semiannual progress reports. The most recent of these is dated April, 1988.

A complete true ab-initio rate theory of kinematics and kinetics for continuum and curved thin structures, without any restriction on the magnitude of the strains or the deformation, was formulated. The time dependence and large strain behavior are incorporated through the introduction of the time rates of the metric and curvature in two coordinate systems; a fixed (spatial) one and a convected (material) coordinate system. The relations between the time derivative and the covariant derivatives (gradients) have been developed for curved space and motion, so that the velocity components supply the

connection between the equations of motion and the time rate of change of the metric and curvature tensors.

The metric tensor (time rate of change) in the convected material coordinate system is linearly decomposed into elastic and plastic parts. In this formulation, a yield function is assumed, which is dependent on the rate of change of stress, metric, temperature, and a set of internal variables. Moreover, a hypoelastic law was chosen to describe the thermoelastic part of the deformation.

A time and temperature dependent viscoplastic model was formulated in this convected material system to account for finite strains and rotations. The history and temperature dependence were incorporated through the introduction of internal variables. The choice of these variables, as well as their evolution, was motivated by phenomenological thermodynamic considerations.

The nonisothermal elastic-viscoplastic deformation process was described completely by "thermodynamic state" equations. Most investigators (in the area of viscoplasticity) employ plastic strains as state variables. Our study shows that, in general, use of plastic strains as state variables may lead to inconsistencies with regard to thermodynamic considerations. Furthermore, the approach and formulation employed by all previous investigators lead to the condition that all plastic work is completely dissipated. This, however, is in contradiction with experimental evidence, from which it emerges that part of the plastic work is used for producing residual stresses in the lattice, which, when phenomenologically considered, causes hardening. Both limitations are not present in our formulation, because of the inclusion of the "thermodynamic state" equations.

The obtained complete rate field equations consist of the principles of the rate of the virtual power and the rate of conservation of energy, of the constitutive relations, and of boundary and initial conditions. These formulations provide a sound basis for the formulation of the adopted finite element solution procedures.

The derived shell theory, in the least restricted form, before any simplifying assumptions are imposed, has the following desirable features:

(a) The two-dimensional, impulse-integral form of the equations of motion and the Second Law of Thermodynamics (Clausius-Duhem inequality) for a shell follow naturally and exactly from their three-dimensional counterparts.
(b) Unique and concrete definitions of shell variables such as stress resultants and couples, rate of deformation, spin and entropy resultants can be obtained in terms of weighted integrals of the three-dimensional quantities through the thickness.
(c) There are no series expansions in the thickness direction.
(d) There is no need for making use of the Kirchhoff Hypotheses in the kinematics.
(e) All approximations can be postponed until the descretization process of the integral forms of the First Law of Thermodynamics
(f) A by-product of the descent from three-dimensional theory is that the two-dimensional temperature field (that emerges) is not a through-the-thickness average, but a true point by point distribution. This is contrary to what one finds in the literature concerning thermal stresses in the shell.

To develop geometrically nonlinear, doubly curved finite shell elements the basic equations of nonlinear shell theories have to be transferred into the finite element model. As these equations in general are written in tensor notation, their implementation into the finite element matrix formulation requires considerable effort.

The nonlinear element matrices are directly derived from the incrementally formulated nonlinear shell equations, by using a tensor-oriented procedure. The classical thin shell theory based on the Kirchoff-Love hypotheses (Formulation D in Appendix A) was employed for this purpose. For this formulation, we are using the "natural" degrees of freedom per mid-surface shell node: three incremental velocities and the rates of rotations about the material coordinates in a mixed form.

A description of the developed element and related finite element code are given in Appendix B. This exposition provides information concerning the formulation, the finite element and how it is employed in the solution of shell-like configurations. Moreover a complete description including program flow chart, listing, input instructions to the user and explanation of output are also included in Appendix B.

The quasi-linear nature of the principle of the rate of virtual power suggests the adoption of an incremental approach to numerical integration with respect to time. The availability of the field formulation provides assurance of the completeness of the incremental equations and allows the use of any convenient procedure for spatial integration over the domain V. In the present instance, the choice has been made in favor of a simple first order expansion in time for the construction of incremental solutions from the results of finite element spatial integration of the governing equations.

The procedure employed permits the rates of the field formulation to be interpreted as increments in the numerical solution. This is particularly convenient for the construction of incremental boundary condition histories.

Even under the condition of static external loads and slowly growing creep effects, the presence of snap-through buckling makes the inertial effects significant. In dynamic analyses, the applied body forces include inertial forces. Assuming that the mass of the body considered is preserved, the mass matrix can be evaluated prior to the time integration using the initial configuration.

Finite element solution of any boundary-value problem involves the solution of the equilibrium equations (global) together with the constitutive equations (local). Both sets of equations are solved simultaneously in a step by step manner. The incremental form of the global and local equations can be achieved by taking the integration over the incremental time step $t=t_{j+1}-t_j$. The rectangular rule has been applied to execute the resulting time integration.

Clearly, the numerical solution involves iteration. A simplified version of the Riks-Wempner constant-arc-length method has been utilized. This iteration procedure which is a generalization of the displacement control method also allows to trace the nonlinear response beyond bifurcation points. In contrast to the conventional Newton-Raphson techniques, the iteration of the method takes place in the velocity and load rate space. The load step of the first solution in each increment is limited by controlling the length ds of the tangent. Either the length is kept constant in each step or it is adapted to the characteristics of the solution. In each step the triangular-size stiffness matrix has to be checked for negative diagonal terms, indicating that a critical point is reached.

One of the most challenging aspects of finite strain formulations is to locate analytical solution with which to compare the proposed formulation. Typically, as a first problem, a large strain uniaxial test case was analyzed. The case considered examines the rate-dependent plastic response of a bar to a deformation history that includes segments of loading, unloading, and reloading, each occurring at varying strain and temperature rates. Moreover, it was shown that the proposed formulation generates no strain energy under a pure rigid body rotation. These are surely important demonstrations but they only represent a partial test because the principal stretch directions remain constant. Finally, a problem which was discussed by Nagtegaal and de Jong, and others too, as a problem which demonstrates limitations of the constitutive models in many strain formulation, is the Couette flow problem. This problem is solved as a third example. The results of these test problems show that:

- The formulation can accommodate very large strains and rotations.
- The formulation does not display the oscillatory behavior in the stresses of the Couette flow problem.
- The model incorporates the simplification associated with rate-insensitive elastic response without losing the ability to model rate temperature dependent yield strength and plasticity.

The problem of buckling of shallow arches under transient thermomechanical load was investigated next. The analysis was performed with the aid of 24 paralinear isoparametric elements. The paralinear isoparametric element is such that the thickness is small compared to other dimensions. The characteristics of the element are defined by the geometry and interpolation functions, which are linear in the thickness direction and parabolic in the longitudinal direction. Consequently, the element allows for shear strain energy since normals to a mid-surface before deformation remain straight, but not necessarily normal to the midsurface after deformation.

The developed solution scheme is capable of predicting response which includes pre- and post-buckling with thermal creep and plastic effects. The solution procedure was demonstrated through several examples which include both creep and snap-through behavior.

The last set of problems which are under investigation consists of creep or thermal buckling, with plastic effects, of shells of revolution.

In addition, following a more traditional approach, a method was developed for bounding the response (solution) of bars and beams of (linear) visoelastic material behavior, based on nonlinear kinematic relations.

In connection with the progress to date, two papers were published by the AIAA Journal in 1986 and 1987. Moreover, a paper entitled, "Non-Isothermal Elastoviscoplastic Analysis of Planar Curved Beams" was presented at the 3rd Symposium on Nonlinear Constitutive Relations for High-Temperature Applications, held at the University of Akron, on June 11-13, 1986. A descriptive abstract of this paper was published in the meeting proceedings and the full paper appeared in a special publication (NASA CP 10010). Copies of the above have been sent to the sponsor.

In addition, the two papers presented at the 28th AIAA/ASME/ASCE/AHS SDM Conference and published in the proceedings of this conference, and the one paper presented at the 21st Conference of the Israel Society of Mechanical Engineers (keynote lecture by Dr. R. Riff) have been accepted for publication; the first two by the AIAA Journal and the last one by the International Journal of Computers and Structures. These three papers deal with applications to snap-through and creep buckling of bars and arches, and to two-dimensional problems in extension and shear. Most of this work was also presented at the NASA-Lewis Conference on Structural Integrity and Durability of Reusable Space Propulsion Systems on May 1987 in Cleveland, Copies of these papers will be forwarded to the Sponsor, as soon as they appear in print.

In connection with the more traditional approach a paper accepted for presentation and for publication in the Proceedings of the special Symposium on Constitutive Equations at the ASME Winter Annual Meeting, Chicago, IL., November 28 - December 2, 1988. The title of the paper is "Creep Analysis of Beams and Arches Based on a Hereditary Visco- Elastic- Plastic Constitutive Law". Copies of this paper will also be forwarded to the sponsor soon (in December, 1988).

Moreover, two abstracts (attached herewith) have been submitted for presentation at and publication in the proceedings of two prestigious conferences: (a) the AIAA/ASME/.../AHS 30th SDM Conference and (b) the 21st Conference of the Int'l. Committee on Aeronautical Fatigue (ICAF).

In addition to the above, two papers are in preparation and both will be completed soon (copies will be sent to the sponsor within 1988). One deals with applications to shell configurations, and it is titled "Analysis of Shell Structures Subjected to Large Mechanical and Thermal Loadings". The second deals with the time- dependent response of curved structural elements, and it is titled "The Dynamic Aspects of Thermo- Elasto- Viscoplastic Snap- Through and Creep Phenomena".

Finally, a finite element has been developed and it is currently being tested for a less restricted shell formulation (Formulation C; see Appendix A). A description of this newly developed element and the related finite element code will be submitted to the sponsor as soon as the testing is completed and reliable results have been obtained.

FUTURE TASKS

The main thrust of the additional tasks is to develop a finite element and select a code, which will be made available to all users and which will be based on the most general (but practical) nonlinear shell formulation possible and nonlinear constitutive relations to predict the response of shell-like structures, when subjected to time-dependent thermomechanical loads with large excursions.

In order to accomplish the overall objective, as stated a number of steps or tasks need be completed. These are:

Step 1: Formulation of the principle of the rate of virtual power for the less restricted shell formulation (C, B, E, A).

Step 2: Couple the general constitutive relations to shell kinematics for the aforementioned foremention formulations.

Step 3: Before developing a finite element, and before numerically testing and comparing the merits of the various formulations, some theoretical evaluation of the degree of approximation will be attempted.

Step 4: Develop a finite element for the chosen least restrictive shell formulation. In order to accomplish this some novel ideas and procedures will be employed. These include a tensor-oriented procedure for obtaining the element, and a discretization process that involves not only the nodal degrees of freedom but also an approximation to the "shift" tensors.

Step 5: Incorporate the developed finite element and related constitutive relations into a FEM code.

Step 6: Numerically test and evaluate the developed finite element procedure.

It is estimated that this entire research effort will require a minimum of three years to bring to fruition. Some of the tasks (step 1 and 2) have already been started.

As already stated, the work on Formulation C is in the testing stage (step 6).

APPENDIX A

The various shell theory approximations (formulations) are based on the use of certain simplifying assumptions regarding the geometry and kinematics of the shell configuration.

These are:

Assumption I: The material points which are on the normal to the reference surface before deformation will be on the same normal after deformation.
Assumption II: The shell is sufficiently thin so that we can assume linear dependence of the position of any material point (in the deformed state) to the normal (to the reference surface) coordinate (in the deformed state). The linear dependence can easily be changed to parabolic, cubic, or any desired degree of approximation.
Assumption III: The rate of change of the velocity gradients with respect to in-plan coordinates on the two boundary shell surfaces is negligibly small.
Assumption IV: The rate of change of the distance of a material from the reference surface is negligibly small.

On the basis of the above four simplifying assumptions, several formulations result, for the analysis of thin shells.

Formulation A: This formulation makes use of Assumption I, only.
Formulation B: This formulation employs Assumptions I and II.
Formulation C: This formulation employs Assumption I, II and III.
Formulation D: This is the classical thin shell theory based on the Kirchhoff-Love hypotheses of Assumptions I, II, III, IV, as applied to large deformation theory.

These formulations are arranged in such a manner that we move from the least restrictive (A) to the most restrictive (D).

In addition to this a fifth formulation (E) can easily be devised and this formulation in terms of order of restriction is similar to Formulation A. Formulation E makes use of Assumption II only.

APPENDIX B FINITE ELEMENT AND RELATED CODE FOR FORMULATION D

In this Appendix, a description of the developed finite element and the related finite element code is presented. First, the essentials of the element are described and then the complete solution procedure is presented with sufficient detail. This includes a flow chart, a line by line listing of the computer program, input data information, and explanation of the output.

B.1. THE SHELL ELEMENT

A brief description highlighting the essential features of the shell element development and the related code used in this work is given here.

In order to derive discrete algorithm based on the finite element displacement method we approximate the velocity field by index-oriented notation, which allows the separate representation of the shape functions (the specific expression depends on the decided upon degress - of - freedom, Lagrangian, Hermitian, etc.) for the tangential velocities v_α and for the normal velocity v_3 .

$$v_\alpha = v_\alpha^M V_M = V^M v_{\alpha M} \tag{1}$$

$$v_3 = v_3^M V_M = V^M v_{3M} \tag{2}$$

Upper indices imply the columns, lower indices the rows of a matrix expression, and the summation is carried out spanning the number of degrees-of-freedom. V^M and V_M represent, therefore, the vector of nodal velocities by the row and by the column respectively. We get the shape functions for the partial derivatives of the velocity shape functions v_α^M , v_3^M :

$$v_{\alpha,\beta} = (v_\alpha^M)_{,\beta} V_M = v_{\alpha,\beta}^M V_M \tag{3}$$

$$v_{3,\beta} = (v_3^M)_{,\beta} V_M = v_{3,\beta}^M V_M \tag{4}$$

The main idea of this formulation is the development of shape functions for further mechanical and thermal variables by the application of well-known tensor procedure on the basic shape functions (3) and (4). Taking, for example, the covariant derivative

$$v_{\alpha;\beta} = v_{\alpha,\beta} - v_\lambda \Gamma_{\alpha\beta}^\lambda \tag{5}$$

and inserting (3) and (1), we can define the shape function of the covariant derivative:

$$v_{\alpha;\beta} = \underbrace{\left(v_{\alpha,\beta}^M - v_\lambda^M \Gamma_{\alpha\beta}^\lambda \right)}_{v_{\alpha;\beta}^M} V_M = v_{\alpha;\beta}^M V_M \tag{6}$$

In the same way we receive the shape function of the internal variables, for example the rate of deformation and the spin tensors:

$$d_{(\alpha\beta)} = \frac{1}{2}\left(v_{\alpha;\beta} + v_{\beta;\alpha} - 2 b_{\alpha\beta} v_3 \right) =$$

$$\underbrace{\frac{1}{2}\left(v_{\alpha;\beta}^M + v_{\beta;\alpha}^M - 2 b_{\alpha\beta} v_3^M \right)}_{d_{(\alpha\beta)}^M} V_M = d_{(\alpha\beta)}^M V_M \tag{7}$$

$$w_{(\alpha\beta)} = -\left(v_{3;\alpha\beta} + v_{\lambda;\alpha}\, b^{\lambda}_{\beta} + v_{\lambda;\beta}\, b^{\lambda}_{\alpha} + v_{\lambda}\, b^{\lambda}_{\alpha;\beta} - v_{3}\, b_{\lambda\alpha}\, b^{\lambda}_{\beta}\right) =$$

$$\underbrace{-\left(v^{\Lambda}_{3;\alpha\beta} + v^{\Lambda}_{\lambda;\alpha}\, b^{\lambda}_{\beta} + v^{\Lambda}_{\lambda;\beta}\, b^{\lambda}_{\alpha} + v^{M}_{\lambda}\, b^{\lambda}_{\alpha;\beta} - v^{M}_{3}\, b_{\lambda\alpha}\, b^{\lambda}_{\beta}\right)}_{w^{M}_{(\alpha\beta)}} V_M = \tag{8}$$

$$= w^{M}_{(\alpha\beta)}\, V_M$$

The same procedure is now applied to the shift tensor

$$\mathcal{M}^{-1\ \Lambda}_{\ T} = \left(\delta^{\gamma}_{\beta} + \zeta_0\, b^{\gamma}_{\beta}\right) \delta^{\beta}_{T}\, \delta^{\Lambda}_{\gamma} + \delta^{3}_{T}\, \delta^{M}_{3} \tag{9}$$

which is responsible for the "exact" distributions over the thickness.

Finally the shape functions of the internal forces and temperature variables can be derived from the shape functions of the rate of deformation and spin tensor via the constitutive relations; for example:

$$N^{(\alpha\beta)} = \underbrace{D H^{\alpha\beta\rho\lambda}\, v^{M}_{(\rho\lambda)}}_{N^{(\alpha\beta)M}} V_M = N^{(\alpha\beta)M}\, V_M \tag{10}$$

All these expression are now substituted into the rate of the first law of thermodynamics to obtain the element "stiffness" equations. The developed element matrices are implemented into the global relation of the complete shell structure by standard assemblage process considering incidence and boundary conditions.

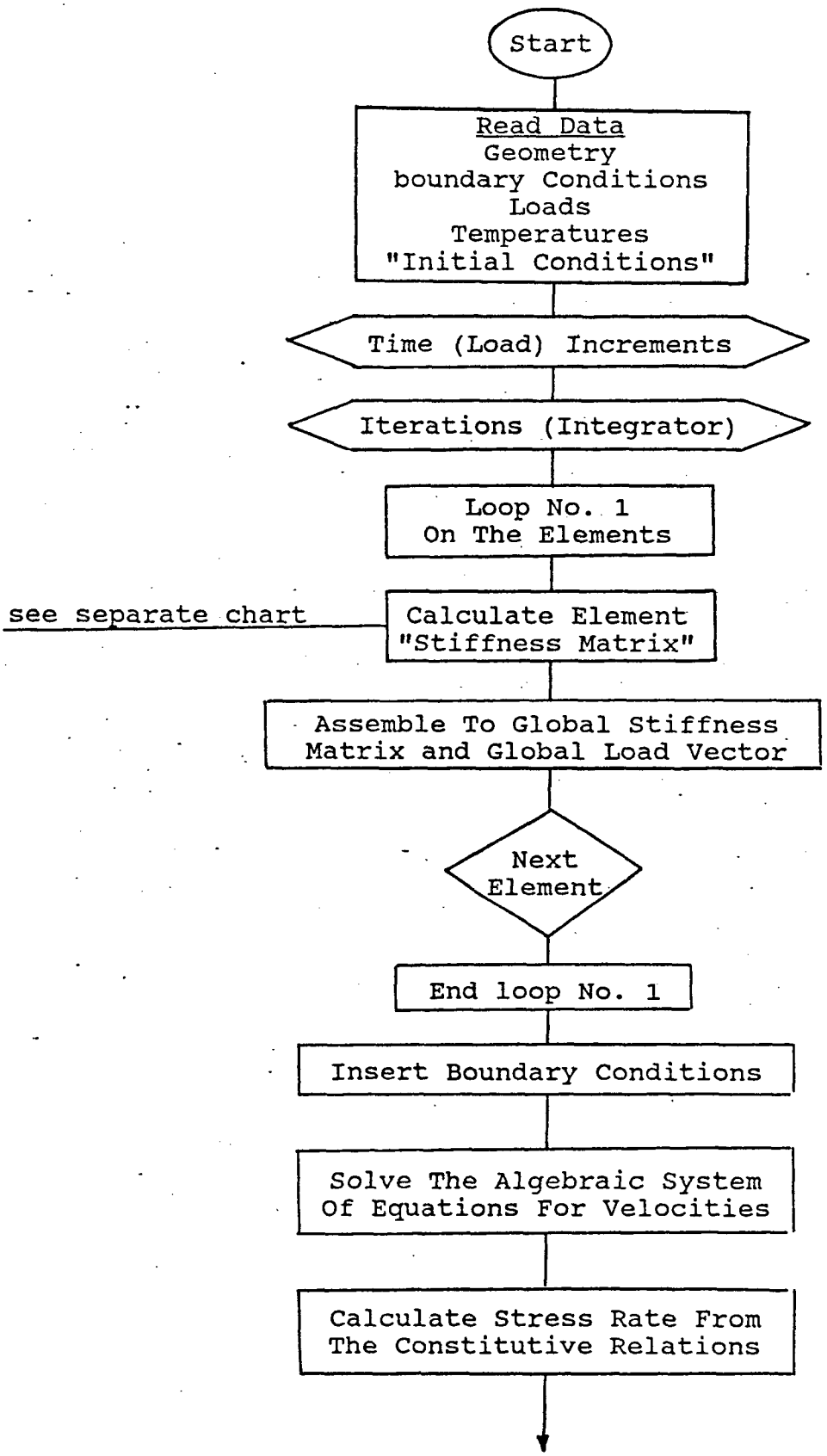
The present curvilinear formulation of the element enables the precise description of geometry, external loads and temperatures and the fulfillment of the convergence criteria, while the rigid body motion condition can only be satisfied in an approximate manner. The tensor oriented formulation renders the optional use of various shape functions for the tangential velocities v and the normal velocity v_3.

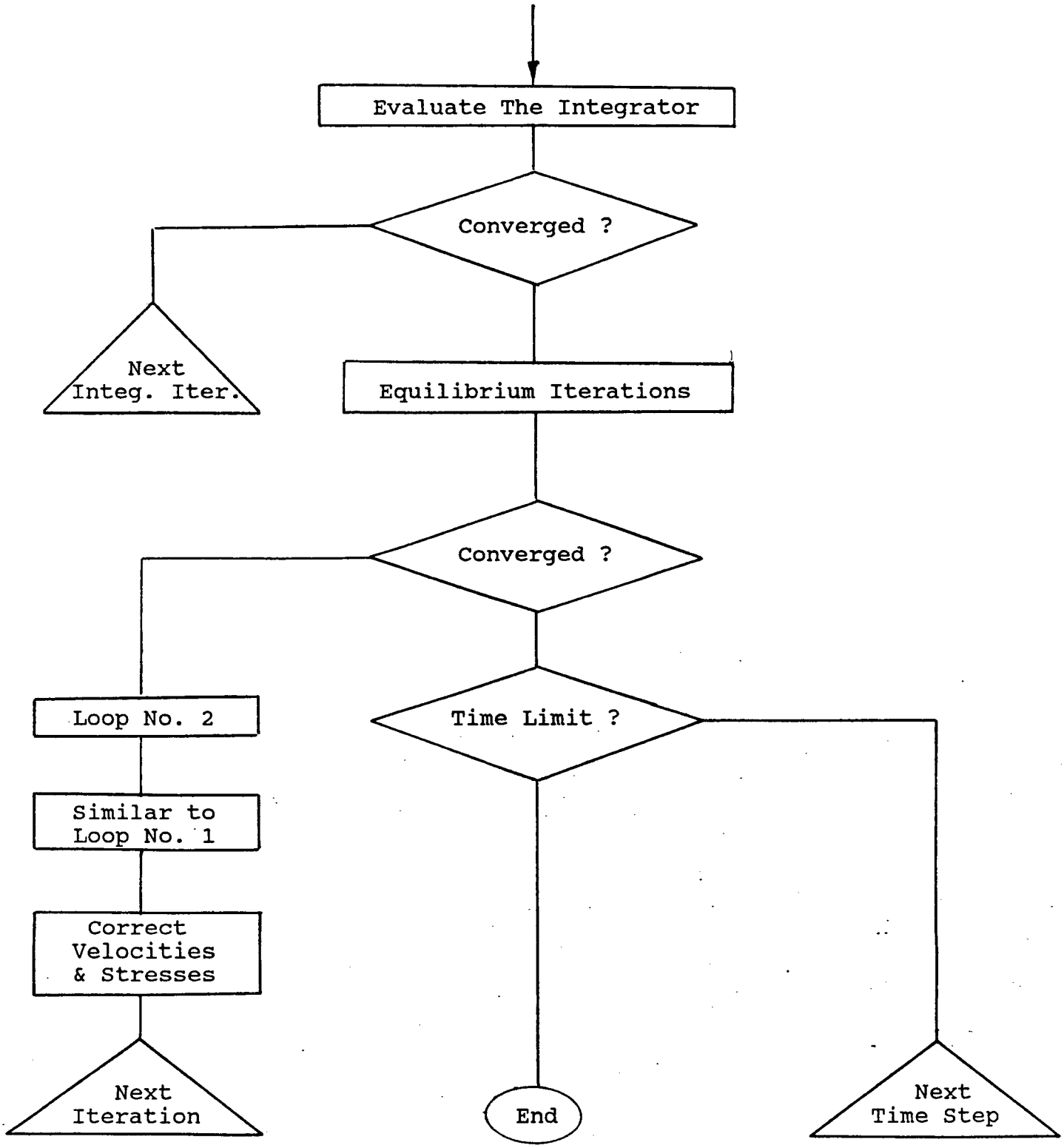
The shell element which have been used up to today is based on the bicubic Hermite polynomial with 4x12 generalized velocities and 4 temperatures. Numerical integration spanning the element domain was applied (16 points of integration), whereby area and boundary integrals were replaced by double integration with respect to the curvilinear θ^{α} - coordinates.

$$dA = \sqrt{(a)}\ d\theta^{1}\, d\theta^{2} \tag{11}$$

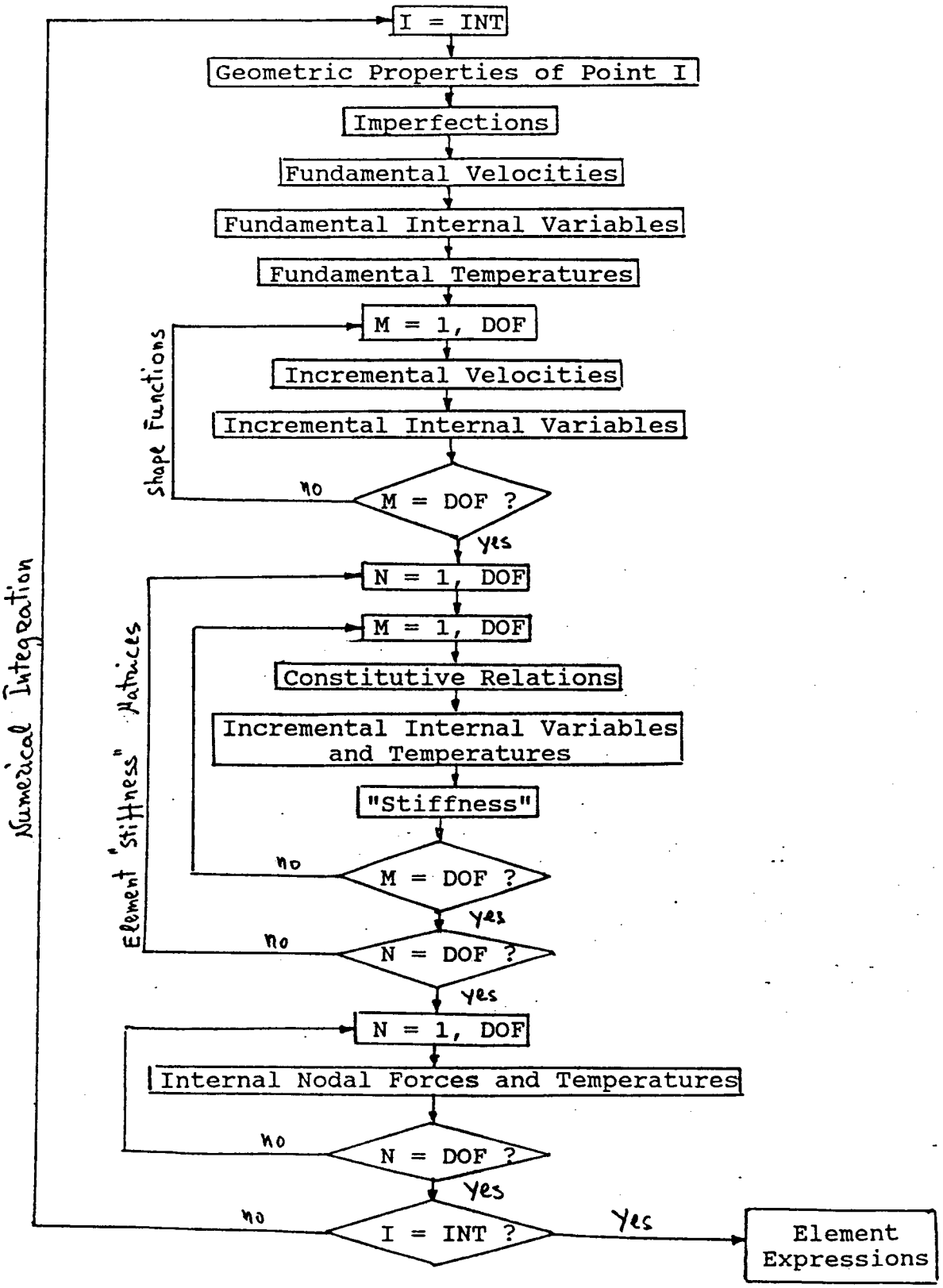
$$dS = \sqrt{(a_{\alpha\rho}\, d\theta^{\alpha}\, d\theta^{\rho})} \tag{12}$$

B.2 - FLOW CHART




Evaluate The Integrator
Converged ?
Next
Integ. Iter.
Equilibrium Iterations
Converged ?
Loop No. 2
Time Limit ?
Similar to
Loop No. 1
Correct
Velocities
& Stresses
Next
Iteration
End
Next
Time Step

FLOW CHART FOR THE SET-UP OF THE ELEMENT MATRICES



B.3.1 - LISTING OF THE MAIN PROGRAM

```
      SUBROUTINE ASEMB (INEL,NSDF,NEL,MR,MC,SKE,NOD,A,ICOL)
C
C     ROUTINE TO ASSEMBLE CONDENSED GLOBAL STIFFNESS MATRIX (A)
C     AND POINTER MATRIX (ICOL)
C
      REAL*8 A
      DIMENSION A(MR,MC)
      DIMENSION SKE(12,12)
      DIMENSION ICOL(MR,MC),NOD(4,NEL)
      IF (INEL.GT.1) GO TO 25
      DO 20 N=1,MR
      DO 20 M=1,MC
      A(N,M)=0
   20 ICOL(N,M)=0
   25 N=INEL
      I=0
      DO 80 JJ=1,4
      NROWB=NROWB+1
      I=I+1
      II=0
      DO 70 KK=1,4
      NCOLB=(NOD(KK,N)-1)*3
      DO 70 K=1,3
      NCOLB=NCOLB+1
      II=II+1
      IF(ABS(SKE(I,II)).LT.1.E-6) GO TO 70
      DO 40 M=1,MC
      IF (ICOL(NROWB,M)-NCOLB) 30,60,30
   30 IF(ICOL(NROWB,M)) 50,50,40
   40 CONTINUE
   50 ICOL(NROWB,M)=NCOLB
   60 A(NROWB,M)=A(NROWB,M)+DBLE(SKE(I,II))
   70 CONTINUE
   80 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE BOUND (MR,MC,NEL,NNP,NSDF,KODE,VAL,A,B,BF,ICOL,
     &ZTEST,FLAG,P)
C
C     SUBROUTINE TO INSERT B.C. AND REORDER MATRIX + POINTER
C
      REAL*8 A,B,AX,ZTEST,R,BF
      DIMENSION KODE(3,NNP),VAL(3,NNP)
      DIMENSION A(MR,MC),B(MR),BF(MR),R(MR)
      DIMENSION ICOL(MR,MC)
      INTEGER FLAG(7)
C
C     ZERO FORCE ARRAY
C
```

```
      DO 10 I=1,MR
      BF(I)=0.0D0
   10 B(I)=0.0D0
C
C     INSERT BOUNDARY CONDITIONS
C
      DO 50 KROW=1,3
      DO 50 KCOL=1,NNP
      IFR=3*(KCOL-1)+KROW
      IF (KODE(KROW,KCOL).EQ.0) GO TO 20
      GO TO 25
   20 B(IFR)=DBLE(VAL(KROW,KCOL))
      GO TO 40
   25 DO 35 I=2,MC
      A(IFR)=0.0D0
   35 ICOL(IFR,I)=0
      A(IFR,1)=1.D0
      ICOL(IFR,1)=IFR
      B(IFR)=DBLE(VAL(KROW,KCOL))
   40 CONTINUE
   50 CONTINUE
      DO 60 L=1,MP
   60 BF(L)=B(L)
C
C     REORDERING
C
      DO 270 IR=1,NSDF
      IF (ICOL(IR,2).EQ.0) GO TO 270
      JC=2
      DO 200 IC=3,MC
      IF (ICOL(IR,IC).EQ.0) GO TO 210
      JC=IC
  200 CONTINUE
  210 JC1=JC
      DO 240 IC=1,JC1
  220 IF(IC.GT.JC) GO TO 250
      IF (DABS(A(IR,IC)).GT.ZTEST) GO TO 240
      DO 230 K=IC,JC
      K1=K+1
      A(IR,K)=A(IP,K1)
  230 ICOL(IR,K)=ICOL(IR,K1)
      JC=JC-1
      GO TO 220
  240 CONTINUE
  250 ISTOP=0
      DO 260 IC=2,JC
      I1=IC-1
      K=ICOL(IR,IC)
      IF(ICOL(IR,I1).LT.K) GO TO 260
      ISTOP=1
      ICOL(IR,IC)=ICOL(IR,I1)
      ICOL(IR,I1)=K
      AX=A(IR,IC)
      A(IR,IC)=A(IR,I1)
```

```
      A(IR,I1)=AX
  260 CONTINUE
      IF(FLAG(6).EQ.0) GO TO 999
      DO 280 I=1,MP
  280 B(I)=R(I)
  999 RETURN
      END
C
C
      SUBROUTINE CONSTIT(INEL,NEL,NNP,NOD,GRAD,VG,RLM,RMU,GCON,
     $ S,MODUL,DR,FLAG,PLAS)
C
C     ROUTINE TO PERFORM CONSTITUTIVE CALCULATIONS
C     3D/1D FORMULATION      (JAUMANN STRESS RATES)
C
C     PARAMETER DESCRIPTION:
C         INEL - NO. OF ELEMENT
C         MODUL - ELEMENT MODULI (CF. SYMSTI)
C         NOD - ELEMENT INDICES
C         S - ELEMENT STRESS
C         DS - RESULTANT STRESS INCREMENTS
C         GRAV - RESULTANT STRAIN RATES
C         VG - GLOBAL VELOCITY VECTOR
C         SMIX - MIXED STRAIN RATES
C         DR - RESULTANT MATERIAL STRAIN RATES
C
      INTEGER FLAG(7)
      REAL*4 MODUL(3,NEL),S(3,3,NEL,DS(3,3),GRAV(3,3),VG(3,NNP)
      REAL*4 GCON(3,3),ET,GRAD(3,4,3,3),GIJ,GIK,WORK(3,3)
      DIMENSION NOD(4,DEL),SMIX(3,3),DR(3,3,NEL)
C
C     STRAIN RATE TENSOR
C
      DO 5 I=1,3
      DO 5 J=1,3
    5 GRAV(I,J)=0.
      DO 10 IA=1,4
      K=NOD(IA,INEL)
      DO 20 IS=1,3
      DO 20 JS=1,3
      D=0.0
      DO 30 I=1,3
   30 D=D+GRAD(IA,IS,JS)*VG(I,K)
   20 GRAV(IS,JS)=D+GRAV(IS,JS)
   10 CONTINUE
      D1=(GRAV(1,2)+GRAV(2,1))*0.5
      D2=(GRAV(1,3)+GRAV(3,1))*0.5
      D3=(GRAV(2,3)+GRAV(3,2))*0.5
      GRAV(1,2)=D1
      GRAV(2,1)=D1
      GRAV(1,3)=D2
      GRAV(3,1)=D2
      GRAV(2,3)=D3
      GRAV(3,2)=D3
```

```
      D=0.0
      DO 40 I=1,3
      DO 40 J=1,3
   40 D=D+GCON(I,J)*GRAV(I,J)
C
C     STRESS INCREMENTS
C
C     ELASTIC
C
      RLAM=RLM*D
      DO 50 I=1,3
      DO 50 J=1,3
      D=0.0
      GIJ=GCON(I,J)
      DO 60 K=1,3
      GIK=GCON(J,L)*RMU
      DO 60 L=1,3
   60 D=D+GIK*GCON(J,L)*GRAV(K,L)
   50 DS(I,J)=D+D+GIJ*RLAM
      IF (FLAG(6).NE.1) GO TO 56
      DO 55 I=1,3
      DO 55 J=1,3
   55 DR(I,J,NEL)=DS(I,J)
      GO TO 999
C
C     VISCOPLASTIC
C
   56 IF(PLAS.LT.0.) GO TO 99
      ET=MODUL(1,NEL)
      IF(ET.EQ.0.00) GO TO 99
      D=MODUL(2,NEL)
      DO 70 I=1,3
      DO 70 J=1,3
   70 WORK(I,J)=S(I,J,INEL)-GCON(I,J)*D
      RUM=0.00
      DO 80 I=1,3
      DO 80 J=1,3
   80 RUM=RUM+WORK(I,J)*GRAV(I,J)
      RUM=RUM*ET
      DO 90 I=1,3
      DO 90 J=1,3
   90 DS(I,J)=DS(I,J)-RUM*WORK(I,J)
   99 CONTINUE
C
C     MIXED STRAIN RATES
C
      DO 25 I=1,3
      DO 25 J=1,3
      D=0.00
      DO 15 K=1,3
   15 D=D+GCON(I,K)*GRAV(K,J)
   25 SMIX(I,J)=D
C
C
```

```
C      MATERIAL STRESS RATES
C
       DO 45 I=1,3
       DO 45 J=1,3
       D=DS(I,J)
       DO 35 K=1,3
  35   D=D-SMIX(I,K)*S(K,J,INEL)-SMIX(J,K)*S(I,K,INEL)
  45   DR(I,J,INEL)=D
 999   RETURN
       END
C
C
       SUBROUTINE DERVEX(INEL,NNP,NEL,NOD,X,FINT,XDER)
       DIMENSION XCOR(4,3),FINT(3,4),XDER(3,3)
       DIMENSION X(3,NNP),NOD(4,NEL)
C
C      TRANSFORMATION TENSOR
C
       CALL NODCOP(INEL,NNP,NEL,NOD,X,XCOR)
       DO 10 I=1,3
       DO 10 J=1,3
  10   XDER(I,J)=0.
       DO 20 L=1,3
       DO 20 I=1,3
       DO 20 J=1,4
       XDER(I,L)=XDER(I,L)+FINT(I,J)*XCOR(J,L)
  20   CONTINUE
       RETURN
       END
C
C
       SUBROUTINE DETGAU (INEL,GCOV,DET,DETG,F)
       DIMENSION GCOV(3,3),G(3,3)
C
C      METRIC DETERMINANT AND GAUSS POINTWEIGHT
C
       DO 20 I=1,3
       DO 20 J=1,3
       G(I,J)=GCOV(I,J)
  20   CONTINUE
       DET=G(1,1)*G(2,2)*G(3,3)+G(1,2)*G(2,3)*G(3,1)+
      *G(1,3)*G(2,1)*G(3,2)-G(1,1)*G(2,3)*G(3,2)-
      *G(1,2)*G(2,1)*G(3,3)-G(1,3)*G(2,2)*G(3,1)
       IF (DET.GT.0.0) GO TO 30
       STOP
  30   DETG=DET**0.5
       F=DETG*1./6.
       RETURN
       END
C
C
       SUBROUTINE EQITER(INEL,NEL,NNP,NSDF,NOD,GRAD,S,VOL,
      &                  Q,Q1,FLAG)
       DIMENSION OD(4,NEL),S(3,3,NEL),GRAD(3,4,3,3),QT(3)
```

```
      DIMENSION Q1(3,NNP)
      REAL*8 Q(NSDF)
C
C     ROUTINE TO CALCULATE RESULTANT NODAL FORCES
C     CAUSED BY THE STRESSES
C
      IF(INEL.NE.1) GO TO 50
      DO 51 I=1,NSDF
   51 Q(I)=0.D0
   50 DO 10 IA=1,4
      ND=NOD(IA,INEL)
      DO 20 I=1,3
      D=0.0
      DO 30 JS=1,3
      DO 30 IS=1,3
   30 D=D+GRAD(I,IA,JS,IS)*S(IS,JS,INEL)
      QT(I)=D*VOL
      NN=(ND-1)*3+I
   20 Q(NN)=Q(NN)+DBLE(QT(I))
   10 CONTINUE
      IF(INEL.EQ.NEL) GO TO 40
      GO TO 999
   40 DO 52 I=1,3
      DO 52 J=1,NNP
      M=3*(J-1)+I
   52 Q1(I,J)=Q(M)
  999 RETURN
      END
C
C
      SUBROUTINE EQSOLV(NPT,NNN,B,A,NCOL,AA,NPIV,IBANDW,NNCOL,
     & ZTEST,V,NNP)
C
C     SUBROUTINE EQSOLV FOR SOLUTION OF LARGE SPARSE NONSYMTRICAL
C     SYSTEM OF LINEAR EQUATIONS
C
C     SOLVES A*X=B WHERE A IS PARTIALLY PACKED MATRIX OF NON-ZERO
C     COEFFICIENTS AND B IS THE CONSTANT VECTOR
C     A      = MATRIX CONTAINING NONZERO COEFFICIENTS OF SYSTEM
C     AA     = ARRAY USED FOR PIVOTAL ROW ELEMENTS
C     B      = CONSTANT VECTOR OF EQ. TO BE SOLVED, ALSO USED
C              FOR RETURNING THE SOLUTION
C     IBANDW = MATRIX CONTAINING INDICES OF NONZERO COEFF. OF A
C     NCOL   = ARRAY FOR PIVOTAL ROW INDICES
C     NNN    = COLUMN DIMENSION IN MAIN PROGRAM FOR A AND NCOL
C     NPIV   = ARRAY TO STORE PIVOTAL COLUMN
C     NPT    = MAXIMUM NUMBER OF ROWS IN THE SYSTEM
C     ZTEST  = VALUE BELOW WHICH, ELEMENT IS MADE EQUAL TO ZERO
C
      REAL*8 A,B,AA,X,C,AAA,A1,SAVE,ZTEST,PMAX,PMIN
      DIMENSION A(NPT,NNN),NCOL(NPT,NNN),AA(NPT),NPIV(NPT)
     &IBANDW(NPT),NNCOL(NPT),B(NPT),V(3,NNP)
C
C
```

```
C       INITIALIZE THE BANDWITH COUNTER
C
        IF(DABS(ZTEST).GT.0.0001) ZTEST=0.0
        PMAX=0
        PMIN=1.Q+10
        NSTOP=0
        MAXWID=0
        DO 1 I=1,NPT
        IBANDW(I)=0
    1   CONTINUE
        DO 5 I=1,NPT
        DO 2 J=1,NNN
        NC=NCOL(I,J)
        IF(NC.EQ.0) GO TO 3
        IBANDW(I)=J
    2   CONTINUE
    3   IF(MAXWID.LT.J) MAXWID=J
        IF(J.NE.1) GO TO 5
        NSTOP=1
    5   CONTINUE
        IF(NSTOP.EQ.1) STOP
C
C       FINDING THE ROW WITH MINIMUM BANDWITH
C
        NPT1=NPT-1
        DO 23 LL=1,NPT1
C
C       FINDING THE ROW WITH MINIMUM BANDWITH
C
        KK=100000
        DO 6 I=LL,NPT
        IC=IBANDW(I)
        IF(IC.LE.0) GO TO 6
        IF(IC.GE.KK) GO TO 6
        KK=IC
    6   CONTINUE
C
C       INTERCHANGE ROWS - MINROW WITH LL
C
        LM=IBANDW(LL)
        M=MINROW
        DO 7 I=1,LM
        NNCOL(I)=NCOL(M,I)
        NCOL(M,I)=MCOL(LL,I)
        AA(I)=A(M,I)
        A(M,I)=A(LL,I)
    7   CONTINUE
        SAVE=B(LL)
        B(LL)=B(M)
        B(M)=SAVE
        IBANDW(LL)=IBANDW(M)
        IBANDW(M)=LM
        RETURN
        END
```

```
C
      SUBROUTINE INDADA(NNP,NEL,KODE,X,VAL,NOD,E,PR,POWER,FACTOR,
     &FPS,DT,XL,XW,XH,TMAX,ICMAX,ITMAX,IDT,FINT,RMU,RLM,PLAS)
      DIMENSION KODE(3,NNP),X(3,NNP),VAL(3,NNP),NOD(4,NEL),EPS(4)
     & ,FINT(3,4),TITLE(20)
C
C     READ AND PRINT PROBLEM DATA
C
      READ 101,TITLE
      PRINT 201,TITLE
      READ 102,E,PR,POWER,FACTOR
      PRINT 203
      PRINT 204,E,PR
      PRINT 205,POWER,FACTOR
      READ 105,XL,XW,XH,TMAX,DT
      PRINT 210,XL,XW,XH
      PRINT 211,TMAX,DT
      READ 102,EPS(1),EPS(2),EPS(3),EPS(4)
      READ 106,ICMAX,ITMAX
      READ 107,PLAS
      PRINT 217,PLAS
      PRINT 212
      PRINT 213,EPS(1),EPS(2),ICMAX
      PRINT 214,EPS(3),EPS(4),ITMAX
      PRINT 215
      PRINT 216,NEL,NNP
C
C     READ AND PRINT NODAL DATA
C
      PRINT 205
    5 READ 103,M,KODE(1,M),KODE(2,M),KODE(3,M),X(1,M),X(2,M),
     & X(3,M),VAL(1,M),VAL(2,M),VAL(3,M)
      IF(M.LT.NNP) GO TO 5
      DO 9 I=1,NNP
      X(1,I)=X(1,I)*XL
      X(2,I)=X(2,I)*XW
      X(3,I)=X(3,I)*XH
    9 CONTINUE
      PRINT 206,(N,KODE(1,N),KODE(2,N),KODE(3,N),X(1,N),X(2,N)
     $,X(3,N),VAL(1,N),VAL(2,N),VAL(3,N),N=1,NNP)
C
C     READ AND PRINT ELEMENT DATA
C
      PRINT 207
      DO 6 N=1,NEL
      READ 104,NOD(1,N),NOD(2,N),NOD(3,N),NOD(4,N)
    6 CONTINUE
      DO 4 L=1,NEL
    4 PRINT 208,L,(NOD(I,L),I=1,4)
      CALL INTFUN (FINT)
      CALL LAME(E,PR,RMU,RLM)
      RETURN
      END
C
```

```
C
      SUBROUTINE INTERP(XDER,FINT,GRAD)
      DIMENSION XDER(3,3),FINT(3,4),GRAD(3,4,3,3)
C
C     ROUTINE TO CALCULATE TRANSFORMATION TENSOR FOR
C     COVARIANT DERIVATIVES COMPONENTS (L)
C
      DO 29 I=1,3
      DO 29 IS=1,4
      TEMP=XDER(IS,I)
      DO 29 IA=1,4
      DO 29 J=1,3
   29 GRAD(I,IA,IS,J)=TEMP*FINT(J,IA)
      RETURN
      END
C
C
C
      SUBROUTINE INTFUN(FINT)
      DIMENSION FINT(3,4)
C
C     ROUTINE TO PROVIDE THE ELEMENT FUNCTION
C
      DO 13 I=1,3
      DO 12 J=1,4
      FINT(I,J)=0.
   12 CONTINUE
   13 CONTINUE
      DO 14 I=1,3
      FINT(I,I)=1.
      FINT(I,4)=-1.
   14 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE INVERT (A,D,N,NX,MX)
C
C     SUBROUTINE TO INVERT A GENERAL MATRIX
C     (USED FOR THE CALCULATION OF THE CONTRAVARIANT
C      METRIC TENSOR)
C
      DIMENSION A(NX,MX)
      N1=N-1
      N2=2*N
      DO 52 I=1,N
      DO 51 J=1,N
      J1=J+N
   51 A(I,J1)=0.
      J1=I+N
   52 A(I,J1)=1.
      DO 60 K=1,N1
      C=A(K,K)
```

```
      K1=K+1
      DO 56 J=K1,N2
   56 A(K,J)=A(K,J)/C
      DO 60 I=K1,M
      C=A(I,K)
   60 A(I,J)=A(I,J)-C*A(K,J)
      NP1=N+1
      DO 59 J=NP1,N2
   59 A(N,J)=A(N,J)/A(N,N)
      DO 61 L=1,N1
      K=N-L
      K1=K+1
      DO 61 I=NP1,N2
      DO 61 J=K1,N
   61 A(K,I)=A(K,I)-A(K,J)*A(J,I)
      DO 62 I=1,N
      DO 62 J=1,N
      J1=J+N
   62 A(I,J)=A(I,J1)
      D=1.
      DO 63 I=1,N
   63 D=D*A(I,I)
      RETURN
      END
C
C
      SUBROUTINE LAME(E,PR,RMO,RLM)
C
C     ROUTINE TO CALCULATE LAME CONSTANTS (ELASTIC)
C
      A=1.+PR
      B=1.-2*PR
      RLM=E*PR/(A+B)
      RMU=E/(2.*A)
      RETURN
      END
C
C
      SUBROUTINE MAIN2N
      REAL*8 A($$,$$),AA($$),B($$),ZTEST,BF($$),R($$),Q($$)
      DIMENSION XDER(3,3),GCOV(3,3),GCON(3,3),FINT(3,4)
     $ GRAD(3,4,3,3),SKE(12,12)
      DIMENSION KODE(3,12),VAL(3,12),X0(3,12),X1(3,12),V0(3,12)
     & ,V(3,12),EPS(4),Q1(3,12),DX(3,12)
      DIMENSION S(3,3,$$),S0(3,3,$$),ST(3,3,$$),DR(3,3,$$)
     & ,D0(3,3,$$),NOD(4,$$),SGL(3,3,$$)
      DIMENSION NPIV($$),IBANDW($$),NNCOL($$),ICOL($$,$$)
      INTEGER FLAG(7)
      REAL*4 MODUL(3,$$),NORM(7)
C*****************************************************************
      NSDF=
      NNP=
      NEL=
C*****************************************************************
```

```
C
C      DATA INPUT/OUTPUT
C
       DATA MR,MC,ZTEST/$$,$$,1.Q-20/
       CALL INDATA(NNP,NEL,KODE,X,VAL,NOD,E,PR,POWER,FACTOR,EPS,
      $ DT,XL,XW,XH,TMAX,ICMAX,ITMAX,FINT,RMU,RLM,PLAS)
C
C      ZERO ARRAYS
C
       CALL ZERO(FLAG,V,S,Q,MODUL,NSDF,NEL,NNP)
       CHECK=100.
C
C      LOOP OVER ELEMENTS
C
   75  FLAG(1)=0
       KKKK=1
   80  CONTINUE
       VGL=0.
       DO 10 KNEL=1,NEL
       INEL=KNEL
C
C      COMPUTATION OF KINEMATICS QUANTITIES
C
       CALL DERVEX(INEL,NNP,NEL,NOD,X,FINT,XDER)
       CALL METRIC(INEL,XDER,GCOV,DET,DETG,VOL,GCON,VGL)
       CALL INTERP(XDER,FINT,GRAD)
C
C      THE ELEMENT STIFNESS MATRIX
C
       CALLSYMSTI(INEL,NEL,RMU,RLM,VOL,GCON,GRAD,SKE,POWER,FACTOR,
      $ S,MODUL,GCOV,PLAS)
C
C      ASSEMBLY TO GLOBAL STIFFNESS MATRIX
C
       CALL ASSEMB(INEL,NSDF,NEL,MR,MC,SKE,NOD,A,ICOL)
   10  CONTINUE
       IF(FLAG(6).NE.0) GO TO 15
       IF(FLAG(3).NE.1) GO TO 15
       PRINT 400,FLAG(2),VGL
C
C      SOLUTION OF ALGEBRAIC EQUATIONS: [K]*(V)=(F)
C
   15  CALL BOUND(MR,MC,NEL,NNP,NSDF,KODE,VAL,A,B,BF,ICOL,ZTEST,
      $ FLAG,R)
       CALL EQSOLV(NSDF,MC,B,A,ICOL,AA,NPIV,IBANDW,NNCOL,ZTEST,
      $ V,NNP)
       IF(FLAG(6).EQ.1) GO TO 73
C
C      STRESS INCREMENTS (CF. CONSTITUTIVE EQUATIONS)
C
       DO 11 INEL=1,NEL
       CALL DERVEX(INEL,NNP,NEL,NOD,X,FINT,XDER)
       CALL METRIC(INEL,XDER,GCOV,DET,DETG,VOL,GCON,VGL)
       CALL INTERP(XDER,FINT,GRAD)
```

```
      CALL CONSTIT(INEL,NEL,NNP,NOD,GRAD,V,RLM,RMU,GCON,S,MODUL,
     $ DR,FLAG,PLAS)
   11 CONTINUE
C
C     INTEGRATOR
C
      DIT=DT
      CALL TRAPEZ(NNP,NEL,EPS,DIT,TMAX,ICMAX,FLAG,X,X0,XT,V,V0,
     $ S,S0,ST,DR,D0,XX,SS)
      IF(FLAG(3)) 70,80,70
C
C     ITERATIONS
C
   70 FLAG(7)=0
   72 DO 71 INEL=1,NEL
      CALL DERVEX(INEL,NNP,NEL,NOD,X,FINT,XDER)
      CALL METRIC(INEL,XDER,GCOV,DET,DETG,VOL,GCON,VGL)
      CALL INTERP(XDER,FINT,GRAD)
      CALL EQITER(INEL,NEL,NNP,NSDF,NOD,GRAD,S,VOL,Q,Q1,FLAG)
   71 CONTINUE
      CALL RW(BF,Q,DTT,FLAG,EPS,NORM,ITMAX,R,NNP,NSDF,KODE,VAL,X,
     $ V,X0,DX,KKKK,CHECK)
      IF(FLAG(6).EQ.1) GO TO 80
C
C     STRESS TRANSFORMATION TO GLOBAL SYSTEM
C
      DO 83 I=1,3
      DO 83 J=1,3
      DO 83 K=1,NEL
   83 SGL(I,J,K)=0.0
      DO 85 INEL=1,NEL
      CALL DERVEX(INEL,NNP,NEL,NOD,X,FINT,XDER)
      DO 84 IB=1,3
      DO 84 IS=1,3
      XII=XDEP(IS,IB)
      DO 84 JS=1,3
      SXJ=S(IS,JS,INEL)*XII
      DO 84 JB=1,3
      SGL(IB,JB,INEL)=SGL(IB,JB,INEL)+SXJ*XDER(JS,JB)
   84 CONTINUE
   85 CONTINUE
C
C     OUTPUT
C
      ISTEP=FLAG(2)
      PRINT 100,ISTEP
      DO 86 K=1,NEL
      PRINT 200,K
   86 PRINT 300,((SGL(I,J,K),J=1,3),I=1,3)
      PRINT 350,(J,(MODUL(I,J),I=1,3),J=1,NEL)
      IF(FLAG(3).LT.0) GO TO 999
C
      GO TO 75
   73 KKKK=0
```

```
      DO 77 INEL=1,NEL
      CALL DERVEX(INEL,NNP,NEL,NOD,X,FINT,XDER)
      CALL METRIC(INEL,XDER,GCOV,DET,DETG,VOL,GCON,VGL)
      CALL INTERP(XDER,FINT,GRAD)
      CALL CONSTIT(INEL,NEL,NNP,NOD,GRAD,V,RLM,RMU,GCON,S,MODUL,
     $ DR,FLAG,PLAS)
   77 CONTINUE
      DO 79 I=1,3
      DO 79 J=1,NNP
   79 X(I,J)=X(I,J)+V(I,J)
      DO 78 I=1,3
      DO 78 J=1,3
      DO 78 K=1,NEL
   78 S(I,J,K)=S(I,J,K)+DR(I,J,K)
      GO TO 72
  999 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE METRIC(INEL,XDER,GCOV,DET,DETG,F,GCON,VGL)
C
C     ROUTINE TO CALCULATE METRIC QUANTITIES
C     FOR THE ELEMENT
C
      DIMENSION XDER(3,3),GCOV(3,3),GCON(3,3)
      DIMENSION TDER(3,3),GC(3,6)
      IF(INEL.EQ.1) VGL=0.
C
C     THE COVARIANT METRIC TENSOR
C
      DO 22 I=1,3
      DO 22 J=1,3
   22 TDER(J,I)=XDER(I,J)
      DO 23 I=1,3
      DO 23 J=1,3
      GCOV(I,J)=0.
      DO 23 K=1,3
   23 GCOV(I,J)=GCOV(I,J)+XDER(I,K)*TDER(K,J)
C
C     THE CONTRAVARIANT METRIC TENSOR
C
      DO 26 I=1,3
      DO 26 J=1,6
      GC(I,J)=0.0
      DO 25 I=1,3
      DO 25 J=1,3
      GC(I,J)=GCOV(I,J)
      TDER(I,J)=GCOV(I,J)
   25 CONTINUE
      CALL DETGAU (INTEL,TDER,DET,DETG,F)
      VGL=VGL+F
      CALL INVERT(GC,D,3,3,6)
```

```
      DO 27 I=1,3
      DO 27 J=1,3
   27 GCON(I,J)=GC(I,J)
      RETURN
      END
C
C
      SUBROUTINE RW(BF,Q,DTT,FLAG,EPS,NORM,ITMAX,R,NNP,NSDF,
     &KODE,VAL,X,V,X0,DX,KKKK,CHECK)
C
C     ROUTINE TO PERFORM EQILIBRIUM ITERATIONS
C     USING "RIKS-WEMPNER MODIFIED METHOD"
C     STIFNESS MATRIX IS UPDATED FOR EACH ITERATION
C
      REAL*8 BF(NSDF),R(NSDF),Q(NSDF),F1
      REAL*8 NORM(7)
      INTEGER FLAG(7)
      DIMENSION EPS(4),KODE(3,NNP),VAL(3,NNP),X(3,NNP),V(3,NNP),
     &X0(3,NNP),DX(3,NNP)
      D1=0.0
      D2=0.0
      D3=0.0
      D4=0.0
      D5=0.0
C
C     RESIDUAL FORCE VECTOR
C
      DO 10 I=1,NSDF
      F1=BF(I)*DBLE(DTT)*FLAG(2)
   10 R(I)=F1-Q(I)
C
C     ERROR NORMS
C
      DO 20 I=1,NSDF
      IF(BF(I).EQ.0.D0) GO TO 20
      D1=D1+R(I)*R(I)
      D2=D2+BF(I)*DBLE(DTT)*BF(I)*DBLE(DTT)*FLAG(2)*FLAG(2)
   20 CONTINUE
      NORM(1)=D1**0.5
      NORM(2)=D2**0.5
      NORM(3)=NORM(1)/NORM(2)
      IF(FLAG(7).LT.ITMAX) GO TO 30
      IF(CHECK.GT.NORM(3)) GO TO 30
      FLAG(4)=1
      GO TO 70
   30 IF(NORM(3).GT.EPS(4)) GO TO 40
      CHECK=NORM(3)
      IF(KKKK.EQ.1) GO TO 70
      DO 60 I=1,3
      DO 60 J=1,NNP
      DX(I,J)=X(I,J)-X0(I,J)
      D3=D3+V(I,J)*V(I,J)
      D4=D4+DX(I,J)*DX(I,J)
   60 D5=D5+X0(I,J)*X0(I,J)
```

```
      NORM(4)=D3**0.5
      NORM(5)=D4**0.5
      NORM(6)=D5**0.5
      QQ=NORM(4)/NORM(5)
      CF=QQ/(1.-QQ)
      NORM(7)=CF*NORM(4)/NORM(6)
      IF(NORM(7).GT.EPS(3)) GO TO 40
C
C     TERMINATION - ITERATIONS CONVERGED
C
   70 FLAG(6)=0
      CHECK=NORM(2)
      IF(FLAG(5).EQ.0) GO TO 75
      STOP 100
   75 IF(FLAG(4).EQ.0) GO TO 999
      FLAG(5)=1
      GO TO 999
C
C     CHECK NO. OF ITERATIONS
C
   40 FLAG(7)=FLAG(7)+1
      IF(FLAG(7).GT.ITMAX) GO TO 888
C
C     NO CONVERGENCE - PREPARE FOR NEXT ITERATION
C
      DO 50 I=1,3
      DO 50 J=1,NNP
      IF(KODE(I,J).EQ.0) GO TO 50
      M=3*(J-1)+I
      R(M)=DBLE(VAL(I,J))
   50 CONTINUE
      FLAG(6)=1
      GO TO 999
  888 STOP 200
  999 CONTINUE
      RETURN
      END
C
C
      SUBROUTINE NODCOP(INEL,NNP,NEL,NOD,X,XCOR)
C
C     ROUTINE TO ASSEMBLE NODAL COORDINATES MATRIX
C     FOR EACH ELEMENT
C
      DIMENSION XCUR(4,3)
      DIMENSION X(3,NNP),NOD(4,NEL)
      DO 8 I=1,4
      DO 7 J=1,3
      XCOR(I,J)=0.
    7 CONTINUE
    8 CONTINUE
      DO 10 I=1,4
      M=NOD(I,NEL)
      DO 9 L=1,3
```

```
      XCOR(I,L)=X(L,M)
    9 CONTINUE
   10 CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE SYMSTI(INEL,NEL,RMU,RLM,VOL,GCON,GRAD,SKE,POWER,
     $FACTOR,S,MODUL,GCOV,PLAS)
      REAL*4 MODUL(3,NEL)
      DIMENSION GRAD(3,4,3,3),GCON(3,3),SKE(12,12),S(3,3,NEL),
     &DEVI(3,3),GCOV(3,3)
C
C     ROUTINE TO CALCULATE ELEMENT STIFNESS MATRIX
C
      INTEGER*2 ARI(12)/1,2,3,1,2,3,1,2,3,/,ARIA(12)/1,1,1,
     &2,2,2,3,3,3,4,4,4/
      ITOT=1
      DO 35 IA=1,4
      DO 35 I=1,3
      DO 36 JTOT=1,ITOT
      JB=ARIA(JTOT)
      D1=0.0
      D2=0.0
      D=0.0
      DO 37 JS=1,3
      DO 37 IS=1,3
      ELI=GRAD(J,IA,JS,IS)
      GIJ=RLM*GCOM(IS,JS)
      DO 37 K=1,3
      ELKL=GRAD(J,JB,K,L)
      ELLK=GRAD(J,JB,L,K)
   37 D=D+ELI*(GIK*GCON(JS,L)*(ELLK+ELKL)+GIJ*GCON(K,L)*ELKL)
      SKE(ITOT,JTOT)=D*VOL
   36 SKE(JTOT,ITOT)=D*VOL
   35 ITOT=ITOT+1
C
      DO 45 ITOT=1,12
      I=ARI(ITOT)
      IA=ARIA(ITOT)
      DO 45 JTOT=1,12
      J=ARI(JTOT)
      JB=ARIA(JTOT)
      D=0.0
      DO 46 JS=1,3
      DO 46 IS=1,3
      LI=GRAD(I,IA,JS,IS)
      SIJ=S(IS,JS,INEL)
      DO 46 K=1,3
      SJK=S(JS,K,INEL)
      SIK=S(IS,K,INEL)
      DO 46 L=1,3
      LKL=GRAD(J,JB,K,L)
```

```
      LLK=GRAD(J,JB,L,K)
   46 D=D+LI*(SIJ*GCON(K,L)*LKL-0.5*(SJK*GCON(IS,L)*(LKL+LLK)
     &-GCON(JS,L)*SIK*(LLK-LKL)))
   45 SKE(ITOT,JTOT)=SKE(ITOT,JTOT)+D*VOL
C
      IF(PLAS.GT.0) GO TO 38
      GO TO 999
   38 CONTINUE
C
      SE=0.
      TRACE=0.
      DO 39 I=1,3
      DO 39 J=1,3
      TRACE=TRACE+GCOV(I,J)*S(I,J,INEL)
      TRACE=TRACE/3
      MODUL(2,INEL)=TRACE
      DO 40 I=1,3
      DO 40 J=1,3
   40 DEVI(I,J)=S(I,J,INEL)-TRACE*GCON(I,J)
      DO 41 I=1,3
      DO 41 J=1,3
      GIJ=DEVI(I,J)
      DO 41 K=1,3
      GIK=GCOV(J,K)*GIJ
      DO 41 L=1,3
   41 SE=SE+GIK*GCOV(J,L)*DEVI(K,L)
      SE=(1.500*SE)**0.5
      MODUL(3,INEL)=SE
C
      EY=RMU*(3.00*RLM+RMU+RMU)/(RLM+RMU)
      ET=0.0
      ET=POWER*FACTOR*(SE/EY)**(POWER-1.00)/EY
      IF(ET.EQ.0.0) GO TO 111
      ET=1.00/ET
      EY=3.00*RMU
      ET=(EY/SE)**2./(EY+ET)
      EY=ET*VOL
C
C     STIFFNESS MATRIX ADJUSTMENT
C
      DO 42 ITOT=1,12
      I=ARI(ITOT)
      IA+ARIA(ITOT)
      DO 42 JTOT=1,12
      J=ARI(JTOT)
      JB=ARIA(JTOT)
      D=0.00
      DO 43 JS=1,3
      DO 43 IS=1,3
      GIJ=DEVI(IS,JS)*GRAD(I,IA,JS,IS)
      DO 43 K=1,3
      DO 43 L=1,3
   43 D=D+GIJ*DEVI(K,L)*GRAD(J,JB,K,L)
   42 SKE(JTOT,ITOT)=SKE(JTOT,ITOT)-D*EY
```

```
  111  MODUL(1,INEL)=ET
  999  CONTINUE
       RETURN
       END
C
C
C
       SUBROUTINE TRAPEZ(NNP,NEL,EPS,DTT,TMAX,ICMAX,FLAG,X,X0,XT,
      $ V,V0,S,S0,ST,DR,D0,XX,SS)
       INTEGER FLAG(7)
       DIMENSION X(3,NNP),X0(3,NNP),XT(3,NNP),V(3,NNP)V0(3,NNP)
      $ ,EPS(4),S0(3,3,NEL),S1(3,3,NEL),DR(3,3,NEL),D0(3,3,NEL)
      $ ,S(3,3,NEL)
C
C      ROUTINE TO PERFORM INTEGRATION OF THE DEPENDENT VARIABLES
C
       IF(FLAG(1).NE.0) GO TO 30
C
C      SAVE ORIGINAL DATA
C
       DO 10 I=1,3
       DO 10 J=1,NNP
       X0(I,J)=X(I,J)
  10   V0(I,J)=V(I,J)
       DO 15 I=1,3
       DO 15 J=1,3
       DO 15 K=1,NEL
       S0(I,J,K)=S(I,J,K)
  15   D0(I,J,K)=DR(I,J,K)
C
C      PREDICTOR
C
       D0 20 I=1,3
       DO 20 J=1,NNP
  20   X(I,J)=X(I,J)+DTT*V(I,J)
       DO 25 I=1,3
       DO 25 J=1,3
       DO 25 K=1,NEL
  25   S(I,J,K)=S(I,J,K)+DTT*DR(I,J,K)
       FLAG(3)=0
       FLAG(1)=1
       GO TO 999
  30   FLAG(1)=FLAG(1)+1
       DO 35 I=1,3
       DO 35 J=1,NNP
  35   XT(I,J)=X(I,J)
       DO 36 I=1,3
       DO 36 J=1,3
       DO 36 K=1,NEL
  36   ST(I,J,K)=S(I,J,K)
C
C      CORRECTOR
C
       DO 40 I=1,3
```

```
      DO 40 J=1,NNP
   40 X(I,J)=X0(I,J)+DTT*0.5*(V0(I,J)+V(I,J))
      DO 45 I=1,3
      DO 45 J=1,3
      DO 45 K=1,NEL
   45 S(I,J,K)=S0(I,J,K)+DTT*0.5(D0(I,J,K)+DR(I,J,K))
C
C     TESTING FOR CONVERGENCE
C
      D1=0.0
      D2=0.0
      D3=0.0
      D4=0.0
      DO 51 I=1,3
      DO 51 J=1,NNP
      XD=X(I,J)-X0(I,J)
      D1=D1+XD*XD
   51 D2=D2+(X(I,J)-XT(I,J))*(X(I,J)-XT(I,J))
      D1=D1**0.5
      D2=D2**0.5
      XNR=D2/D1
      D0 52 I=1,3
      DO 52 J=1,3
      DO 52 K=1,NEL
      D3=D3+S(I,J,K)*S(I,J,K)
   52 D4=D4+(S(I.J.K)-ST(I,J,K))*(S(I,J,K)-ST(I,J,K))
      D3=D3**0.5
      D4=D4**0.5
      SNR=D4/D3
      IF(XNR.GT.EPS(1)) GO TO 888
      IF(SNR.GT.EPS(2)) GO TO 888
      FLAG(3)=1
      FLAG(2)=FLAG(2)+1
      FLAG(1)=0
  998 SNUM=TMAX/DTT
      IF(FLAG(2).LT.SNUM) GO TO 999
      FLAG(3)=-1
      GO TO 999
  888 IF(FLAG(1).LT.ICMAX) GO TO 999
      XX=XNR-EPS(1)
      SS=SNR-EPS(2)
      STOP
  999 RETURN
      END
C
C
C
      SUBROUTINE ZERO(FLAG,ASX,ASS,ASBD,MSMD,NSDF,NEL,NNP)
      INTEGER FLAG(7)
      REAL*8 ASBD(NSDF)
      REAL*4 ASX(3,NNP),ASS(3,3,NEL),MSMD(2,NEL)
C
C     AUXILIARY ROUTINE TO ZERO ARRAYS
C
```

```
      DO 10 I=1,7
10    FLAG(I)=0
      DO 20 I=1,3
      DO 20 J=1,NNP
20    ASX(I,J)=0.
      DO 30 I=1,3
      DO 30 J=1,3
      DO 30 K=1,NEL
30    ASS(I,J,K)=0.
      DO 40 I=1,NSDF
40    ASBD(I)=0.D0
      DO 50 I=1,2
      DO 50 J=1,NEL
50    MSMD(I,J)=0.
      RETURN
      END
```

B.3.2 - LISTING OF THE SUBROUTINES FOR THE SHELL ELEMENT

```
      SUBROUTINE CONST(ISEG,IPOINT,S,LAYER,EP1,EP2,EC1,EC2,Z,KK,
     1SBBAR,EBARP,EPEFF,ETAN,NQ2,SGEFF,G12,EST1,EST2,FKST1,FKST2,
     2EX,EY,UP,EBARC,FN,FM,CPA,CPB,ICREEP,IFLOW)
C
C     SUBROUTINE FOR THE 2-D MATERIAL LAW
C
      COMMON/ITERS/ITER
      COMMON/FUTIME/DTIMEF
      COMMON/STSS/SIG1,SIG2
      COMMON/TEMTUR/TEMP(25),P(25)
C
      COMMON/TOMER/TOME,DTIME
      COMMON/FANDD/ALPHA1,ALPHA2,TTURE,F1,F2,IHINGE,ITEST
      COMMON/FDATA/S25,ECOEF,S26
      COMMON/MDATA/MP,E,U,U1,S24,S23,E11,E12,E22,EALP1,EALP2
      COMMON/SVTOM/TOMES
      COMMON/DEFM/E1,E2,K1,K2,JUP,IPLS(1),AT1
      COMMON/FLSTEP/KSTEP,KSTEPM
      COMMON/SVTEMP/TEMPS(25)
      COMMON/NBGSTP/NBEG(25),NSTOPP(25),INTVAL(25)
      DIMENSION SBBAR(1),EBARP(1),EPEFF(20),ETAN(20),SGEFF(20)
      DIMENSION EPS1(50),EPS2(50),EBARC(1)
      REAL K1,K2
C
C     CALCULATES NEW VALUES FOR VISCOPLASTIC STRAIN,VISCOPLASTIC
C     STRAIN RATE WITH THE USE OF THE FIRST MATERIAL LAW.
C     DIRECTIONAL HARDENING IS INCORPORATED. 'CREEP' ARRAYS
C     HAS BEEN USED FOR STORING AND ACCUMULATING THE DIRECTIONAL
C     HARDENING TERMS:BETA1,BETA2.
C     THIS HAS BEEN POSSIBLE BY SETING ICREEP=1.
C     (CREEPA=LARGE NUMBER). EBARC,HAS BEEN USED TO STORE WP,
C     (PLASTIC WORK)
C
      DOUBLE PRECISION BETA10,BETA20,WP0,EBAR,
     &          BM1,BM2,BZ0,BZ1,BZ3,BD0,BN,SIGMAP
     &          BZI0,BZD0,BZT0,BZIY,BZDY,BZTY,
     &          SIG10,SIG20,SBAR0,SIG1Y,SIG2Y,SBARY,SBARYY,
     &          S10,S20,J20,S1Y,S2Y,J2Y,SBAR,DWP,
     &          SIG1D,SIG2D,EPS1S,EPS2S,
     &          EPNEW1,EPNEW2,E1SEL,E2SEL,E1ELAS,E2ELAS,
     &          FEPS0,EPRT01,EPRTY1,EPRTY2,
     &          DE1,DE2,DEFF,H11,H12,H21,H22,
     &          CI11,CI22,CI12,CI21,C1,C2,
     &          DT,DTOT,DTPART,DELT,DELTS,
     &          DS1,DS2,DEP1,DEP2,DEBAR,DSBAR,
     &          FDOTK,FKOTK,SRAT,FEPSY,C11,C12,C21,C22,
     &          G,EP1D,DI11,DI12,DI21,DI22,ETOT,DWPY,ARG,
     &          DSBAR1,DSBAR2,HPRIME,DENOM,WPDOT,ET,
     &          A1,A2,A3,HRECIP
C
C
```

```
C      DATA FOR REPRESENTING PARTICULAR MATERIAL
C
C      UNITS IN NEWTON-MM
C
C      IN-100 PARAMETERS
C
       BZ0=895.0D0
       BZ1=1123.0D0
       BZ3=120.0D0
       BN =2.8D0
       BD0=10000.0D0
       BM1=0.038D0
       BM2=2.63D0
       SIGMAP=441.0D0
C
       BTETA=0.6D0
       IDIREC=1
C
       KTEST=MOD(KSTEP,INTVAL(ISEG))
       IPLS(1)=0
       EBAR=EBARP(KK)
       DEBAR=0.0D0
       ET=E
       H11=0.0D0
       H12=0.0D0
       H21=0.0D0
       H22=0.0D0
       C11=0.
       C12=0.
       C21=0.
       C22=0.
       JTEST=0
       KTMAX=1
       FKTMAX=KTMAX
       DEC1=0.
       DEC2=0.
       DECBAR=0.0
       SBAR=0.
       KOUNT=0
       TSTRS1=0.0
       TSTRS2=0.0
       SDIF=0.0
       DWPY=0.0
       WPDOT=0.0D0
       Q=0.999
       QR=1./(Q-1)
C
C      DIRECTIONAL HARDENING INDICATOR,INDIRC=1,MEANS THAT
C      D.H.INCLUDED
C
      IDERC=1
C
C      FIRST GET STRAINS THRU THICKNESS
C      CURRENT STRAINS AT KKTH STATION THRU THICKNESS
```

```
C
      EPS1(KK)=(E1-Z*K1)/F1
      EPS2(KK)=(E2-Z*K2)/F2
C
C     CONVERGED STRAINS AT LAST LOAD STEP
C
      EPS1S= (EST1-Z*FKST1)/F1
      EPS2S= (EST2-Z*FKST2)/F2
C
C     VISCOPLASTIC STRAIN COMPONENTS FROM LAST LOAD
C
      EPNEW1=EP1
      EPNEW2=EP2
      BETA10=0.
      BETA20=0.
C
      IF (IDEREC.NE.1) GO TO 10
      BETA10=EC1
      BETA20=EC2
   10 CONTINUE
      WP0=EBARC(KK)
C
      IF (IPOINT.NE.12) GO TO 6
      IF (KK.GT.1) GO TO6
      WRITE (6,5) BETA10,BETA20,WP0,EBAR
    5 FORMAT (//,1H,'BETA10=',G14.4,'BETA20=',G14.4,T55,'WP0=',
     1G14.4,T80,'EBAR=',G14.4)
      IF (WP0.LT.0.0D0) WP0=0.0D0
C
C     THERMAL QUANTITIES AT PRESENET AND AT LAST LOAD STEP
C     TSTR1=ALPHA1*TTURE*TEMP(ISEG)
      TSTR1=0.0D0
C     TSTR2=ALPHA2*TTURE*TEMP(ISEG)
      TSTR2=0.0D0
C     CALCULATE ELASTIC STRAINS CORRESPONDING TO LAST LOAD STEP
C
      E1SEL=EPS1S-EP1
      E2SEL=EPS2S-EP2
C
C     CALCULATE 'ELASTIC STRAINS' FOR CURRENT LOAD STEP
C
      E1ELAS=EPS1(KK)-EPNEW1
      E2ELAS=EPS2(KK)-EPNEW2
C
C     CALCULATE 'STRESSES' FOR CURRENT LOAD STEP
C
      SIG1D=EX*(E1ELAS+U*E2ELAS)
      SIG2D=EY*(U*E1ELAS+E2ELAS)
      SIG1=SIG1D
      SIG2=SIG2D
C
      SBAR=DSQRT(SIG1D*SIG1D+SIG2D*SIG2D-SIG1D*SIG2D)
      SBARY=SBAR
C
```

```
      IF (IFLOW.EQ.1) GO TO 35
      SIG10=EX*(E1SEL+U*E2SEL)
      SIG20=EY*(U*E1SEL+E2SEL)
      SBAR0=DSQRT(SIG10*SIG10+SIG20*SIG20-SIG10*SIG20)
      S1G2Y=SIG10
      SIG2Y=SIG20
      SBARY=SBAR0
C
C     NOW CALCULATE STRAIN RATE COMPONENTS CORESPONDING TO LAST
C     CONVERGED LOAD STEP
C
      BETA1Y=BETA10
      BETA2Y=BETA20
      WPY=WP0
      IDIREC=1
      CALL BZVALO(WPY,IDIREC,BM1,BZ0,BZ1,SIG10,SIG20,BETA10,
     &            BETA20,BZI0,BZD0,BZT0)
C
      S10=(2.0D0*SIG10-SIG20)/3.0D0
      S20=(2.0D0*SIG20-SIG10)/3.0D0
      J20=SBAR0*SBAR0/3.0D0
C
      BZTY=BZT0
      S1Y=S10
      S2Y=S20
      J2Y=J20
C
      EPRT01=0.0D0
      EPRT02=0.0D0
      EPRTY1=0.0D0
      EPRTY2=0.0D0
      IF (SBAR(KK).EQ.0.0)SBAR(KK)=SIGMAP
      IF (SBARY.LT.0.5D0*SIGMAP) GO TO 220
C
      CALL SBRATE(SIG10,SIG20,BD0,BZT0,BN,EPRT01,EPRT02,
     &            S10,S20,J20)
C
      EPRTY1=EPRT01
      EPRTY2=EPRT02
C
      IF(SBAR.EQ.0.0) GO TO 220
C     IF(SBAR.LT.0.5D0*SIGMAP) GO TO 220
      SRAT=DMAX1(DABS(EPRT01),DABS(EPRT02))
      IF (SRAT.LT.1.0D-8) GO TO 220
C
C     DE1,DE2,ARE THE STRAIN RATES INCREMENTS FOR CURRENT LOAD
C     STEP.
C     DETERMINE HOW MUCH IS ELASTIC AND HOW MUCH IS VISCOPLASTIC
C
      DE1=E1ELAS-E1SEL
      DE2=E2ELAS-E2SEL
      DEFF=2.0D0*DSQRT((DE1*DE1+DE2*DE2+DE1*DE2)/3.0D0)
C
      KTMAX=DEFF/.0002
```

```
      IF (KTMAX.GT.100) KTMAX=100
      IF (KTMAX.EQ..0) KTMAX=1
      DTTOT=0.0
      DELT=TOME-TOMES
      IF (DELT.EQ.0.0)DELT=1.0
C
   30 CONTINUE
      FKTMAX=KTMAX
      IFLAG=0
      DT=DELT/FKTMAX
   35 CONTINUE
C
      ETOT=EBAR+S25*SBARY
      IF (IFLOW.EQ.1) GO TO 230
C
C     CHECK IF SUBINCREMENTATION COMPLETED
C
      IF(DTTOT.GE.DELT*0.999) GO TO 200
C
C     FIND APPROXIMATE VALUES FOR THE STRAIN RATE FOR THE
C     CURRENT SUBINCREMENT.
C
   60 CONTINUE
C
      CALL BRATE(SIG1Y,SIG2Y,BD0,BZTY,BN,EPRTY1,EPRTY2,
     &           S1Y,S2Y,J2Y)
C
      CALL BDERIV(SIG1Y,SIG2Y,BD0,BN,BZTY,H11,H12,H22,H21)
C
  120 CONTINUE
      IFLAG=0
      DTTOT=DTTOT+DT
      IF(DTTOT.LE.DELT*0.999) GO TO 130
      DTPART=DTOT-DELT
      DTTOT=DTTOT-DT
      DT=DT-DTPART
      DTTOT=DTTOT+DT
      KOUNT=KTMAX
  130 CONTINUE
C
C     HAVING OBTAINED THE TIME STEP,DT,NOW CALCULATE THE CHANGES
C     IN STRESS,DS1 AND DS2 AND DSBAR.
C     FIRST CALCULATE THE NECESSARY MATRICES,DI,CI
C
      CI11=BTETA*DT*H11
      CI12=BTETA*DT*H12
      CI21=CI12
      CI22=BTETA*DT*H22
      CALL DIM(CI11,CI12,CI22,E,U,DI11,DI12,DI21,DI22)
C
      C1=DE1*DT/DELT
      C2=DE2*DT/DELT
C
      DS1=DI11*(C1-DT*EPRTY1)+DI12*(C2-DT*EPRTY2)
```

```
      DS2=DI21*(C1-DT*EPRTY1)+DI22*(C2-DT*EPRTY2)
C
      DEP1=DT*EPRTY1+CI11*DS1+CI21*DS2
      DEP2=DT*EPRTY2+CI21*DS1+CI22*DS2
      DEBAR=2.0D0*DSQRT((DEP1*DEP1+DEP2*DEP2+DEP1*DEP2)/3.0D0)
C     DEBAR=HRECIP*DSBAR
C
      IF(KTMAX.GE.100) GO TO 150
      SDOTK=SQRT(DS1*DS1+DS2*DS2)
      FKDOTK=SDOTK/SBBAR(KK)
      IF (FKDOTK.LT.0.005) GO TO 150
      KTMAX=2*KTMAX
      DTTOT=DTTOT-DT
      DT=DT/2.
      GO TO 120
C
C     NOW THE CORRECT VALUE OF DT HAS BEEN EVALUATED
C
  150 CONTINUE
      IF(IFLAG.EQ.1) GO TO 160
      IFLAG = 1
      IFLAG1 = 1
  160 CONTINUE
C
      EPNEW1=EPNEW1+DEP1
      EPNEW2=EPNEW2+DEP2
      EBAR=EBAR+DEBAR
C
      SIG1Y=SIG1Y+DS1
      SIG2Y=SIG2Y+DS2
      SBARYY=SQRT(DIG1Y*SIG2Y+SIG2Y*SIG2Y-SIG1Y*SIG2Y)
      DSBAR=SBARYY-SBARY
      SBARY=SBARYY
      S1Y=(2.0D0*SIG1Y-SIG2Y)/3.0D0
      S2Y=(2.0D0*SIG2Y-SIG1Y)/3.0D0
      J2Y=(SBARY*SBARY)/3.0D0
      DWPY=(SIG1Y-DS1/2.0D0)*DEP1+(SIG2Y-DS2/2.0D0)*DEP2
      WPDOT=DWPY/DT
      IF(DWPY.LT.0.0D0)DWPY=0.0D0
      WPY=WPY+DWPY
C
      SIG1=SIG1Y
      SIG2=SIG2Y
      SBAR=SBARY
      ETOT=EBAR+S25*SBARY
      EP1D=(ETOT-SBAR*S24)*100.0
C
      CALL BZVAL(WPY,IDEREC,BM1,BM2,BZ0,BZ1,BZ3,SIG1,SIG2,BETA1Y,
     &           BETA2Y,DWPY,BZIY,BZDY,BZTY)
C
      CALL SBRATE(SIG1Y,SIG2Y,BD0,BZTY,BN,EPRTY1,EPRTY2,
     &            S1Y,S2Y,J2Y)
C
      SBAR=SBARY
```

```
      HRECIP=DEBAR/DSBAR
      ET=1./(HRECIP+1./E)
      IF(ET.EQ.0.0D0) ET=0.001D0*E
C
      IF(IPOINT.NE.2) GO TO 177
      IF(KK.NE.3) GO TO 177
      WRITE(6,176) IPOINT,KK,DT,EPID,DEBAR,ETOT,SBAR
  176 FORMAT(2I4,1P5E11.4)
  177 CONTINUE
C
      IF(IPOINT.NE.12) GO TO 180
      IF(KK.GT.1) GO TO 180
C     WRITE(6,170) IPOINT,KK,SBARY,SIG1,SIG2,DEP1,DEP2,DWPY,DE1,
C    &DE2,BETA1Y,BETA2Y
  170 FORMAT(2I4,1P10E11.4)
C
  180 CONTINUE
C     IF(IFLAG.EQ.1AND.IFLAG1.EQ.1) GO TO 120
C     IF(IPOINT.NE.12) GO TO 191
C     IF(KK.GT.1) GO TO 191
C     WRITE(6,98) SBAR,ET,EBAR,ETOT,DS1,DS2,DSBAR,DEBAR,EPRTY1,
C    &EPRTY2,EP1D,DTTOT,DT
  98  FORMAT(1P13E10.3)
  191 CONTINUE
      GO TO 30
C
  200 CONTINUE
C
C     OBTAIN THE FOUR ELEMENTS OF THE (C) MATRIX
C
      DSBAR1=(SIG1Y-UP*SIG2Y)/SBARY
      DSBAR2=(SIG2Y-UP*SIG1Y)/SBARY
      QRR=SBAR/SBBAR(KK)
      IF(QRR.GE.1.0D0) GO TO 209
      IF(JCODB.EQ.0) GO TO 209
      IF(JCODB.EQ.2) GO TO 207
C
C     JCODB=1
      IF(ET.LE.0.0D0) GO TO 206
      IF(ET.GT.E) GO TO 206
      GO TO 209
  206 CONTINUE
      ET=0.999D0*E
      GO TO 209
C
C     JCODB=2
  207 CONTINUE
      IF(ET.LT.0.0D0) GO TO 208
      IF(ET.GT.E) ET=0.999D0*E
      GO TO 209
  208 CONTINUE
      ET=0.001D0*E
  209 CONTINUE
C
```

```
      IF(QRR.LT.Q) GO TO 210
      IF(ET.LT.E) GO TO 210
      QA=QR*(QRR-1.0)
      ET=QA*E+(1.0-QA)*0.001D0*E
  210 CONTINUE
C
      IF(ABS(E-ET)/E.LT.0.001) GO TO 220
C
      HPRIME=E*ET/(E-ET)
      DENOM=HPRIME+ECOEF*(DSBAR1*DSBAR1+2.*U*DSBAR1*DSBAR2
     &      +DSBAR2*DSBAR2)
      A1=DSBAR1+U*DSBAR2
      A2=DSBAR1*U+DSBAR2
      A3=ECOEF/DENOM
C
      C11=A1*DSBAR1*A3
      C12=A2*DSBAR1*A3
      C21=A1*DSBAR2*A3
      C22=A2*DSBAR2*A3
C
      IF(IPOINT.NE.12) GO TO 211
      IF(KK.GT.1) GO TO 211
C
C     WRITE(6,518) C11,C12,C21,C22,EPRTY1,EPRTY2
  518 FORMAT(//,1H,'C11=',F9.3,T20,'C12=',F9.3,T40,'C21='
     &,F9.3,T60,'C22=',F9.3,T80,'EPRTY1=',G10.3,T100,
     &'EPRTY2=',G10.3)
C
C     WRITE(6,219) SIG1,SIG2,BZTY,BZIY,WPY,SBAR,ET
  219 FORMAT(//,1H,7G12.4)
  211 CONTINUE
C
      IF(SBAR.GE.SBBAR(KK))IPLS(1)=1
C
  220 CONTINUE
C
      E11=EX*(1.-C11-U*C21)
      E12=ECOEF*(U-C12-U*C22)
      E22=EY*(1.-C12*U-C22)
C
      IF(IPOINT.NE.12) GO TO 223
      IF(KK.GT.1) GO TO 223
C     WRITE(6,221) E11,E12,E21,E22
  221 FORMAT(//,1H1,'MATRIX E',4G12.4)
C     WRITE(6,222)DSBAR,DEBAR,EBAR,WPDOT
  222 FORMAT(//,1H,4G12.4)
  223 CONTINUE
C
C     CALCULATE THE "EQUIVALENT THERMOMECHANICAL LOADS"
C
      FN1=-EPNEW1+EPS1(KK)*C11+EPS2(KK)*C12
      FN2=-EPNEW2+EPS1(KK)*C21+EPS2(KK)*C22
      EALP1=-EX*(FN1+U*FN2)*F2
      EALP2=-EY*(U*FN1+FN2)*F1
```

```
  230 CONTINUE
      ETOT=EBAR+S25*SBAR
      G12=0.5*E/(1.+U)
      IF(IFLOW.EQ.0) GO TO 235
      E11=EX
      E12=U*ECOEF
      E22=EY
  235 CONTINUE
C
      IF(SBAR.LT.0.98*SBBAR(KK)) GO TO 240
      EP1D=ETOT-SBAR*S24
      G=1.5*(E*EP1D/SBAR-1.0)
      G12=0.5*E/(1.+U+G)
C
      IF(IFLOW.EQ.0) GO TO 240
      GPRIME=2.25*(E/ET-E*EP1D/SBAR)/SBAR**2
      AA=EP1D/SBAR
      BB=-(U+G/3.)/E
      S1=(2.*SIG1-SIG2)/3.
      S2=(2.*SIG2-SIG1)/3.
      AS=AA+GPRIME*S1*S1/E
      BS=BB+GPRIME*S1*S2/E
      CS=AA+GPRIME*S2*S2/E
      DEN=AS*CS-BS*BS
      E11=CS/DEN
      E12=-BS/DEN
      E22=AS/DEN
      IF(UP.NE.0) GO TO 240
      E11=ET*EX/ECOEF
      E12=0.0
      E22=ET*EY/ECOEF
  240 CONTINUE
C
      IF(IFLOW.EQ.1) GO TO 320
      IF(NQ2.LT.2.AND.ITER.NE.1) GO TO 280
      IF(KTEST.NE.0) GO TO 280
      IF(IPOINT.LT.NBEG(ISEG).OR.IPOINT.GT.NSTOPP(ISEG))GO TO 280
C
C     NOW WRITE OUT ALL THE IMPORTANT QUANTITIES
C
      SK1=0.001
      SK2=0.000001
      IF (E.GT.20000.) GO TO 250
      SK1=1.0
      SK2=0.001
  250 CONTINUE
      SPBAR=SBAR*SK1
      SP1=SIG1*SK1
      SP2=SIG2*SK1
      EPR1=EPNEW1*100.0
      EPR2=EPNEW2*100.0
      ERT1=EPRTY1/SK2
      ERT2=EPRTY2/SK2
      EBARPR=EBAR*100.0
```

```
      RWPY=WPY*1.0
      SPBAR1=SBBAR(KK)*SK1
      ESP=ET*SK2
      EPP1=(EPS1(KK))*100.0
      EPP2=(EPS2(KK))*100.0
      EP1D=(ETOT-SBAR*S24)*100.0
      IF(KK.EQ.1)WRITE(6,260)
  260 FORMAT(1H)
      WRITE(6,270)ISEG,IPOINT,Z,SPBAR,SP1,SP2,EPR1,EPR2,ERT1,
     &ERT2,EBARPR,RPWY,SPBAR1,ESP,EPP1,EPP2,EP1D
  270 FORMAT(2I4,I5,14F10.4)
C
C     ESTABLISH NEW VALUES FOR THE YIELD STRESS AND HARDNING
C     PROPERTIES.
C
  280 IF(ITER.NE.1)GO TO 320
      IF(IPLS(1).EQ.0) GO TO 290
      SBBAR(KK)=SBAR
  290 CONTINUE
      EBARP(KK)=EBAR
      EP1=EPNEW1
      EP2=EPNEW2
      EBARC(KK)=WPY
      EC1=BETA1Y
      EC2=BETA2Y
      IF(IPOINT.NE.12)GO TO 296
      IF(KK.GT.1)GO TO 296
C     WRITE(6,11) SIG10,SIG20
C     WRITE(6,295) EC1,EC2,EBARC(KK0,EBARP(KK)
C  11 FORMAT(//1H,'SIG10=',G14.4,T25,'SIG20=',G14.4)
  295 FORMAT(//1H,'EC1=',G14.4,T25,'EC2=',G14.4,T55,'EBARC(KK)=',
     & G14.4,T80,'EBARP(KK)=',G14.4)
  296 CONTINUE
C
C     DT=DTIMEF
      FN1=EPNEW1+EPS1(KK)*C11+EPS2(KK)*C12
      FN2=EPNEW2+EPS1(KK)*C21+EPS2(KK)*C22
      EALPI=-EX*(FN1+U*FN2)*F2
      EALP2=-EY*(FN2+U*FN1)*F1
  320 CONTINUE
      RETURN
      END
C
C
C     SUBROUTINE TO COMPUTE THE HARDENING TERMS: BZ10,BZD0,BZT0,
C     BASED ON LAST CONVERGED LOAD STEP
C
      SUBROUTINE BZVAL0(WPY,DIREC,BM1,BZ0,BZ1,SIG10,SIG20,
     & BETA10,BETA20,BZ10,BZD0,BZT0)
      IMPLICIT REAL (A-H,O-Z)
      REAL*8 WPY,BZ0,BZ1,SIG10,SIG20,BETA10,BETA20,BZ10,BZD0,BZT0
     & ,BU1,BU2,DENOMI
C
      BZD0=0.0D0
```

```
      IF(IDIREC.EQ.0) GO TO 10
C
      DENOMI=DSQRT(SIG10*SIG10+SIG20*SIG20)
      IF(DENOMI.EQ.0.0D0)GO TO 10
      BU1=SIG10/DENOMI
      BU2=SIG20/DENOMI
C
      BZD0=BU1*BETA10+BU2*BETA20
   10 BZI0=BZ1+(BZ0-BZ1)*DEXP(-BM1*WPY)
C
      BZT0=BZ10+BZD0
      RETURN
      END
C
C
C     SUBROUTINE TO CALCULATE AND UPDATE LOCAL HARDENING
C     PROPERTIES
C
      SUBROUTINE BZVAL(WPY,IDIREC,BM1,BM2,Z0,Z1Z3SIG1,SIG2,
     & BETA1Y, BETA2Y,DWPY,BZI,BZD,BZT)
      IMPLICIT REAL (A-H,O-Z)
      REAL*8 WPY,BM1,BM2,Z0,Z1,Z3,SIG1,SIG2,BETA1Y,BETA2Y,FDBETA,
     & DWPY,BZI,BZD,BZT,U1,U2,DENOMI
      INTEGER IDIREC
C
      BZD=0.0D0
      IF(IDIREC.EQ.0)GO TO 10
      DENOMI=DSQRT(SIG1*SIG1+SIG2*SIG2)
      IF (DENOMI.EQ.0.0D0) GO TO 10
      U1=SIG1/DENOMI
      U2=SIG2/DENOMI
C
      FDBETA=1.0D0-DEXP(-BM2*DWPY)
      BETA1Y=BETA1Y+(Z3*U1-BETA1Y)*FDBETA
      BETA2Y=BETA2Y+(Z3*U2-BETA2Y)*FDBETA
      BZD=BETA1Y*U1+BETA2Y*U2
C
   10 BZI=Z1+(Z0-Z1)*DEXP(-BM1*WPY)
      BZT=BZD+BZI
      RETURN
      END
C
C
C     SUBROUTINE TO COMPUTE(DN)MATRIX
C
      SUBROUTINE DIM(CI11,CI12,CI22,E,U,DI11,DI12,DI21,DI22)
      IMPLICIT REAL*8(A-H,O-Z)
      REAL*8 CI11,CI12,CI22,DI11,DI12,DI21,DI22,E,U,B11,B12,
     & B21,B22,E11,E12,E21,E22,D11,D12,D21,D22,A11,A12,A21,
     & A22,DET
C
      D11=E/(1.0D0-U*U)
      D12=U*E/(1.0D0-U*U)
      D21=U*E/(1.0D0-U*U)
```

```
      D22=E/(1.0D0-U*U)
C
C     E=(D)(CI)
C
      E11=D11*CI11+D12*CI12
      E12=D11*CI12+D12*CI22
      E21=D21*CI11+D22*CI12
      E22=D21*CI12+D22*CI22
C
C     (I)=(D)(CI)
C
      E11=1.0D0+E11
      E22=1.0D0+E22
C
C     INVERSE[(I)+(D)(CI)]
C
      DET=E12*E21-E11*E22
      IF(DET.EQ.0.0D0)GO TO 10
      A11=E22/DET
      A12=E12/DET
      A21=E21/DET
      A22=E11/DET
C
C     INVERSE[(I)+(D)(CI)(D)]
C
      DI11=A11*D11+A12*D12
      DI12=A11*D12+A12*D22
      DI21=A21*D11+A22*D12
      DI22=A21*D12+A22*D22
C
      GO TO 20
   10 DI11=D11
      DI12=D12
      DI21=D21
      DI22=D22
C
   20 CONTINUE
      RETURN
      END
C
C
C     SUBROUTINE TO CALCULATE THE DERIVATIVE OF STRAIN RATE WITH
C     RESPECT TO THE STRESS
C
      SUBROUTINE BDERIV (SIG1,SIG2,D0,BPN,Z,H11,H12,H22,H21)
      IMPLICIT REAL *8(A-H,O-Z)
      REAL*8 H11,H12,H22,H21,SIG1,SIG2,D0,BPN,Z,J2,S1,S2,SIGEF,
     & F2,A,B,C
C
      H11=0.0D0
      H12=0.0D0
      H21=0.0D0
      H22=0.0D0
      SIGEF=DSQRT(SIG1*SIG1+SIG2*SIG2-SIG1*SIG2)
```

```
      IF(SIGEF.EQ.0.0D0) GO TO 10
      J2=SIGEF*SIGEF/3.0D0
      F2=DEXP(-(0.5D0)*(Z/SIGEF)**(2.0D0*BPN))
      A=D0*F2/DSQRT(J2)
      B=-(A/(2.0D0*J2))
      C=A*((BPN/(2.0D0*J2))*(Z/SIGEF)**(2.0D0*BPN))
      S1=(22.0D0*SIG1-SIG2)/3.0D0
      S2=(22.0D0*SIG2-SIG1)/3.0D0
C
      H11=(2.0D0/3/0D0)*A+B*S1+C*S1*S1
      H12=A/3.0D0+B*S1*S2+C*S1*S2
      H21=H12
      H22=A*(2.0D0/3.0D0)+B*S2*S2+C*S2*S2
   10 CONTINUE
C
      RETURN
      END
C
C
      SUBROUTINE STDCTY (IB,IC,IL,A)
C
C     CALLED FROM GASP
C     THIS SUBROUTINE WRITES HACKDCTY
C
      DIMENSION A(4096)
      DIMENSION IB(IL)
      IF(IL.GE.1)WRITE (2) (A(I),I=1,4096),IC,IB(I),I=1,IL)
      IF(IL.LT.1)WRITE (2) (A(I),I=14096),IC
      RETURN
      END
C
C
      ENTRY RDDCTY (IB,IC,IL,A)
C
C     CALLED FROM GASP
C     THIS SUBROUTINE READS HACKDCTY
C
      IF (IL.GE.1)READ (2) (A(I),I=1,4096),IC,(IB(I),I=1,IL)
      IF (IL.LT.1)READ (2) (A(I),I=1,4096),IC
      RETURN
      END
C
C
      SUBROUTINE SBRATE (SIG1,SIG2,D0,BZ,BN,EPRT1,EPRT2,S1,S2,J)
      IMPLICIT REAL (A-H,O-Z)
      REAL*8SIG1,SIG2,DO,BZ,BN,EPRT1,EPRT2,S1,S2,J,SIGEF,ARG,F
C
      ZERO=0.0D0
      EPRT1=ZERO
      EPRT2=ZERO
      SIGEF=DSQRT(SIG1*SIG1+SIG2*SIG2-SIG1*SIG2)
      S1=(2.0D0*SIG1-SIG2)/3.0D0
      S2=(2.0D0*SIG2-SIG1)/3.0D0
      J=SIGEF*SIGEF/3.0D0
```

```
      ARG=0.5D0*(BZ/SIGEF)**(2.0D0*BN)
C
      IF (ARG.GT.20.0D0) GO TO 10
      F=D0*DEXP(-ARG)
      EPRT1=F*S1/DSQRT(J)
      EPRT2=F*S2/DSQRT(J)
C
   10 CONTINUE
      RETURN
      END
```

B.4 - I/O INSTRUCTIONS

1. INPUT VARIABLES DEFINITION

E	- Young's modulus
PR	- Poisson's ratio
POWER	- Material parameter
FACTOR	- Material parameter
XL	- Dimension in X1 direction
XW	- Dimension in X2 direction
XH	- Dimension in X3 direction
TMAX	- Maximal time
DT	- Time increment
EPS(1)	- Convergence criterion for displacements in the integration
EPS(2)	- Convergence criterion for stresses in the integration
EPS(3)	- Convergence criterion for displacements in the iterations
EPS(4)	- Convergence criterion for forces in the iterations
ICMAX	- Maximum number of 'corrections' for one step
ITMAX	- Maximum number of iterations for one step
PLAS	- 0 - Elastic deformations only 1 - Viscoplastic deformations
M	- Node number
KODE(I,M)	- Indicates boundary condition in X_i direction at node M 0 - Force condition 1 - Displacement condition
X(I,M)	- Coordinate X_i at node M
VAL(I,M)	- Value of boundary condition in X_i direction at node M (see KODE)
NODE(J,N)	- The number of the J node of element N

2. INPUT GUIDE

TITLE

E PR POWER FACTOR

XL XW XH TMAX DT

EPS(1) EPS(2) EPS(3) EPS(4)

ICMAX ITMAX

PLAS

* One line for each node:

```
M           KODE(1,M)  KODE(2,M)  KODE(3,M)  X(1,M)    X(2,M)
X(3,M)      VAL(1,M)   VAL(2,M)   VAL(3,M)

* One line for each element:
NOD(1,N)   NOD(2,N)   NOD(3,N)   NOD(4,N)
```

3. OUTPUT EXAMPLE

```
      *****MATERIAL DATA*****
      E= 0.10000E+03      PR= 0.25000E+00
      PRGEP= 0.25000E+02      FACTOR= 0.10000E+54
LENGHT OF BAR= 0.10000E+02   WIDTH= 0.10000E+01HEIGHT= 0.10000E+01
OVERALL TIME= 0.10000E+02      TIME INCREMENT 0.10000E+00
PLAS=    0
*****CONVERGENCE CRITERIA*****
INTEGRATOR CRITERIA:  DISPLACEMENT: 0.10000E-01  STRESS: 0.10000E-01     MAX. CYCLES PER STEP=   10
EQUILIBRIUM CRITERIA:  DISPLACEMENT: 0.10000E-02  FORCE: 0.10000E-01     MAX. ITERATIONS PER STEP=    5
*****DISCRETIZATION DATA*****
NO. OF ELEMENTS=   10   NO. NODAL POINTS=,   12
      NODE      KODE       X1          X2          X3          VAL1        VAL2        VAL3
        1       1 1 0     0.0         0.0         0.10000E+01 0.0         0.0         0.0
        2       1 1 1     0.0         0.0         0.0         0.0         0.0         0.0
        3       1 0 1     0.0         0.10000E+01 0.0         0.0         0.0         0.0
        4       1 0 0     0.0         0.10000E+01 0.10000E+01 0.0         0.0         0.0
        5       0 1 0     0.50000E+01 0.0         0.10000E+01 0.0         0.0         0.0
        6       0 1 1     0.50000E+01 0.0         0.0         0.0         0.0         0.0
        7       0 0 1     0.50000E+01 0.10000E+01 0.0         0.0         0.0         0.0
        8       0 0 0     0.50000E+01 0.10000E+01 0.10000E+01 0.0         0.0         0.0
        9       0 1 0     0.10000E+02 0.0         0.10000E+01 0.25000E+00 0.0         0.0
       10       0 1 1     0.10000E+02 0.0         0.0         0.25000E+00 0.0         0.0
       11       0 0 1     0.10000E+02 0.10000E+01 0.0         0.25000E+00 0.0         0.0
       12       0 0 0     0.10000E+02 0.10000E+01 0.10000E+01 0.25000E+00 0.0         0.0
    ELEMENT     NODE1     NODE2     NODE3     NODE4
        1         6         3         1         2
        2         3         6         8         7
        3         8         1         3         4
        4         1         8         6         5
        5         1         8         6         3
        6         6         9         8         5
        7        11         9         8        12
        8        11         9         6        10
        9        11         8         6         7
       10         6        11         9         8

************MAXMID =     27
MAX PIVOT=  36.334    MIN PIVOT=  .97453
```