A Total Lagrangian Finite Element Analysis for Metal Forming With Application to Metal Extrusion.

Mehrdad Foroozesh
Louisiana State University and Agricultural & Mechanical College

Order Number 9025304

A Total Lagrangian finite element analysis for metal forming with application to metal extrusion

Foroozesh, Mehrdad, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1989

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A Total Lagrangian Finite Element Analysis for Metal Forming with Application to Metal Extrusion

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Civil Engineering

by
Mehrdad Foroozesh
B.S., Louisiana State University, 1983
M.S., Louisiana State University, 1988

December 1989

ACKNOWLEDGMENTS

Many people have earned my gratitude during the course of this study. I specifically wish to thank my parents for the encouragement, moral support, and the financial support that they have given me which has helped me to advance this far and to achieve goals which would have been impossible for me to achieve without their constant guidance. I thank my wife, Mehri, for her understanding and support. Her presence by my side has been a valuable asset. I dedicate this work to her and my son, Paymon. I thank my sisters Maryam and Mahtab for encouraging me to continue my education.

I express my sincere appreciation to Dr. George Z. Voyiadjis who has guided me through the graduate school and who has provided me with a research assistantship for the final year of my graduate studies. His aid and knowledge has made this work possible. To Dr. Richard R. Avent, Dr. Mehmet T. Tumay, Dr. Fariborz Barzegar, Dr. Flora Wang, and Dr. M. Sabbaghian, I express my true appreciation for the valuable advice that they have given me and for serving as members of my advisory committee.

I wish to express my gratitude to the members of the Department of Civil Engineering at LSU and in particular to the chairman of the department, Professor Roger K. Seals, who provided me with financial assistance during the first four years of my graduate studies. I also wish to express my appreciation to the College of Engineering for providing some of the computer time needed to perform the analyses required to complete this study.

Partial support of this research was provided by the National Science Foundation under grant No. MSM-8800832.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

ABSTRACT

A detailed formulation for finite element analysis of metal forming problems is carried out in this work. It incorporates every aspect of the analysis, including the iterative solution procedures for geometric and material non-linearities, implementation of the material model, and formulation of curved contact boundaries. The finite element formulation is based on a Total Lagrangian approach which by-passes the use of the Jaumann stress rate tensor commonly used in the Updated Lagrangian formulation. The yield model used is of the von Mises type with both kinematic and isotropic hardening and is formulated in the Eulerian space. This model is then transformed to the Lagrangian reference frame. In the evaluation of stresses, yielding is first detected through the use of an elastic-predictor stress; subsequently upon detection of yielding, the consistency condition is used to evaluate the actual stress and plastic strain tensors. This method is used in conjunction with subincrementation of the strain increment tensor. The curved kinematic boundaries are modeled using the Hermite parametric formulation although other formulations such as β-splines and Bésier parametric curves may also be used with slight modifications. The above mentioned formulations are incorporated into the finite element program, *UNIFES (UNIfied Finite Element Solver)*, which is developed by the author. This program may be used for analysis of 2-D and 3-D problems. A complete listing of this program along with the details of the formulations and a users guide is provided in this work.

Applicability of the above formulations in solving metal extrusion problems is examined through several finite element analyses which are performed by using the *UNIFES* program. It is shown how the distance between the nodes on the die interface can lead to fluctuations in the extrusion pressure, and how the amplitude of these fluctuations may be reduced by mesh refinement, using multiple types of

elements. The effect of changes in the die angle as well as changes in the reduction ratio, on the extrusion pressure is also investigated. A detailed account of the solution procedures is also provided.

1. INTRODUCTION

1.1 General Remarks

Although fabrication of metals into useful shapes by deformation processes has been known to man since the end of the Neolithic Era, systematic scientific investigation of fabrication techniques did not begin until just over fifty years ago. Progress was slow because of both insufficient understanding of the fundamental mechanisms involved in metal deformation, and the inadequate techniques used to model the complex processes occurring during large deformations. Early studies employed simple models of material behavior in order to calculate the relationships between forming loads and the degree of deformation. These analyses have proved useful for design of equipment used in fabrication and for determination of the forming limits and processing schedules applicable to simple product geometries.

Within the last twenty years, the advent of large computers coupled with advances in the finite element method (FEM) have led to greater capability for analyzing processes which produce complex shapes. Simplifications of modelling metal behavior, such as assuming rigid plastic flow, the von Mises yield criterion and the Prandtl-Reuss flow law, are often adequate to describe metal behavior in deformation processes provided that no information is required on the properties of the resulting product. A particularly instructive example of the power of this method applied to metal forming problems is the design of the near net shape forging processes, where the metal flow pattern is a major concern.

Advances in the understanding of the mechanisms of metal deformation and in the ability to incorporate more accurate models of metal behavior in finite element analysis provide opportunities for a major expansion of the understanding of the relationship among material properties and process variables.

One method of describing the material behavior using the finite element technique is the flow approach (Zienkiewicz and Godbole [1974,1975], Zienkiewicz and Jain [1978]), where the metal is assumed to behave as a non-Newtonian fluid. Although large increments in strain and rotation are accommodated in this method, the elasto-plastic behavior of the material is not properly treated, leading to incorrect results for the metal flow. An alternate method for the solution of metal forming problems is the solid approach where the material is considered to be an elasto-plastic solid. The rigid-plastic formulation (Kobayashi [1977], Kobayashi and Lee [1973], Klie [1979], Roll [1978]), where the elastic deformations are ignored when compared to the large plastic strains represents an example of this approach. Generalizations of this rigid-plastic finite element technique, are used by a number of researchers to deal with the hot, rate dependent processes. The main disadvantage of the rigid-plastic formulation is its inability to predict the stress history whenever elastic loadings or plastic unloadings are encountered. The residual stresses are of critical importance in many manufacturing processes, as for example the "springback" phenomenon in the case of sheet metal forming which is governed by the residual stresses. In order to predict the residual stress in the formed product, it is essential to conduct an elasto-plastic analysis since a rigid-plastic treatment will predict stresses only in the regions currently experiencing considerable plastic flow. The elasto-plastic solid approach enables us to obtain the distribution of the residual stresses and the hardening in the form of subsequent plastic yield surfaces, which are not available from the rigid-plastic formulation. Hence, large strain elasto-plastic formulations are introduced to eliminate the shortcomings inherent in rigid-plastic formulations. The studies by Wifi [1976], Wifi and Yamada [1980], Lee and Mallett [1977], Wang and Budiansky [1978], Hibbitt, et. al. [1970], McMeeking and Rice [1975], and Lee [1976] are among many in this area.

Since metal forming involves the formation of large strains, the constitutive strain-displacement relations are non-linear. The geometric non-linearities involved along with the path dependence of the material properties in the plastic range create complex numerical problems which have to be overcome. In finite element analysis of metal forming problems, researchers have moved towards the Updated Lagrangian formulation (Shiau and Kobayashi [1988], Aravas [1986], Oh [1982], Ghosh and Kikuchi [1988], Yang and Yoon [1989], Nagtegaal [1982], Cheng and Kikuchi [1985]). In this formulation, the configuration of the body is updated after each load increment is applied. Hence, the current configuration corresponds to the initial configuration for the subsequent load increment. The popularity of the Updated Lagrangian formulation is due to the fact that the material models which are applicable to small strain problems may be applied directly to this formulation with only slight modifications. This is because in the Updated Lagrangian formulation the Cauchy stress (true stress) and the Almansi's strain tensor are used as the stress and strain measures respectively.

An impediment in the use of the Updated Lagrangian approach is the difficulty of identifying a proper co-rotational stress rate primarily for problems involving finite strains and kinematic hardening. In a number of recent papers by Nagtegaal and de Jong [1982], Lee, et. al. [1983], Dafalias [1983], and Johnson and Bammann [1984], the non-applicability of the Jaumann stress rate to kinematic hardening elasto-plastic constitutive models that display finite strains was pointed out. In these references, it was demonstrated that an oscillatory shear stress is predicted for monotonically increasing simple shear strain when the Jaumann stress rate is used in a kinematic hardening model.

A number of stress rates were proposed by the above authors in order to remedy the oscillatory behavior of the shear stress. Johnson and Bammann [1984], compared their proposed stress rate to the solution obtained using the Green-

Naghdi rate based on a Lagrangian definition of the yield criterion. This is a different yield criterion than the von Mises criterion used in the above references in conjunction with the proposed co-rotational stress rates.

Lee, et. al. [1983], developed a modified Jaumann stress rate and demonstrated its applicability to the specific problem of simple shear. The generalization of this modified Jaumann derivative to the three-dimensional case is not yet demonstrated. In a recent paper by Atluri [1984], it was pointed out that the problem with stress rate is mainly due to improper generalization of the infinitesimal strain theories to the finite strain case. Generalized stress rates are introduced in the above paper to correct the anomalies introduced by kinematic hardening plasticity models that display finite strains.

In the studies performed by Dafalias [1983,1985], Atluri [1984], and Johnson and Bammann [1984], a missing conceptual link is suggested between the microscopic and macroscopic formulations of finite strain plasticity through the plastic spin concept proposed by Dafalias [1984].

An alternate approach to the Updated Lagrangian formulation is the Total Lagrangian approach. In this method the stress and strain measures are referred to the original undeformed configuration of the body. Unfortunately, most researchers are avoiding the Total Lagrangian approach for the solution of metal forming problems because the stresses obtained through this method are the second Piola Kirchhoff stresses which have no real physical meaning. When the Total Lagrangian formulation is used, extensive transformations are needed in order to modify the material model appropriately (Voyiadjis [1984], Voyiadjis and Kiousis [1987]).

A Total Lagrangian approach is used in this work (Voyiadjis [1984]). The yield criterion is originally expressed in terms of the Cauchy stress and subsequently transformed to the Lagrangian reference frame. The associated flow rule used here preserves the normality rule in the second Piola-Kirchhoff stress space and is equivalent to that of the Cauchy stress space. Although this approach preserves the accuracy of the interpretation of the material behavior in the Eulerian reference frame, it by-passes the use of the Jaumann stress rate in the formulation of the kinematic hardening finite-strain plasticity. The resulting constitutive model is expressed in terms of three material parameters that are determined experimentally.

1.2 Objectives

A Summary of the objectives of this work is provided here. The details of these objectives are discused in the next section. These objectives are as follows:

1) Show applicability of the Total Lagrangian approach in solving large strain metal forming problems.

2) Develop an easy to use and flexibile method for modeling the contact boundaries in metal forming problems.

3) Develop a technique for efficient implementation of an elasto-plastic material model with kinematic and isotropic hardening capabilities.

4) Develop a user friendly finite element program based on the above formulations and with flexibility for future additions or modifications.

5) Perform a limited parametric study of axisymmetric extrusion problems.

1.3 Scope

A finite element program, *UNIFES (UNIfied Finite Element Solver)*, is developed in this work for analysis of general 2-D and 3-D finite strain solid mechanics problems. The geometric and material non-linearities are handled through a Newton-Raphson iterative solution procedure which is based on a Total Lagrangian formulation. In order to facilitate the solution of plane stress, plane strain, and axisymmetric metal forming problems, special routines have been added to this program for handling the kinematic boundary conditions (the present version of this program is not capable of solving 3-D problems which involve contact boundaries). Applicability of the above formulations to metal forming problems is examined through a parametric study of axisymmetric extrusion. A brief discussion of the formulations used in *UNIFES* is presented next with more details provided in the chapters that follow. A listing of the computer program *UNIFES* is provided in the appendices.

The proposed Total Lagrangian finite element approach for the solution of metal forming problems incorporates an elasto-plastic von Mises type yield criterion with both isotropic and kinematic hardening capabilities. The present formulation is derived for isothermal conditions. Use of the Lagrangian reference frame for the solution of the problem enables us to utilize the material stress rate as an objective stress rate. Consequently, the problem of identifying a correct co-rotational stress rate which is associated with the Updated Lagrangian formulation is by-passed in this case. Nevertheless, the yield criterion used in this work is expressed in terms of the Cauchy stress tensor and is subsequently transformed to the Lagrangian reference frame. This approach preserves the accuracy of the interpretation of the material behavior in the Eulerian reference frame while it by-passes the problem of the correct identification of a proper stress rate. The proposed solid approach enables us to obtain the distribution of the residual stresses and the hardening in the form of the subsequent plastic yielding, which is not possible using the rigid-plastic formulation.

A systematic method for implementation of the above mentioned model into a finite element program is developed. The details for calculation of the elasto-plastic stiffness tensor and evaluation of the stresses and plastic strains along with the corresponding FORTRAN programs are presented and discussed in detail in Chapter 3.

Frequently in metal forming analysis, curved boundaries are used as part of the extrusion die, forming presses, and rollers. In finite element implementation of these problems it is necessary to accurately simulate the geometry of these boundaries in an effective and simple way. The design engineer should be able to model a variety of curved shaped boundaries without having to modify the program.

Generation of curves and surfaces have been the subject of extensive research by solid modelers and those involved in computer graphics. Many approaches for generation of curves have been proposed, among them are the Bésier, Overhauser, Hermite and β-spline formulations. The Bésier and β-spline formulations have gained tremendous popularity for their ease of use in interactive systems. These methods of curve and surface generation have also been applied to metal forming problems.

In this work the curved contact boundaries are modeled using Hermite parametric curves. A detailed explanation of this formulation is provided which may also be applied to β-spline, Bésier and Overhauser parametric formulations with only slight modifications. Both tension free contact, and fixed rolling contact may be simulated. The details of this formulation is presented in Chapter 4.

A limited parametric study of the extrusion problem is performed to verify the applicability of the Total Lagrangian formulation to the solution of metal forming problems. Several different meshes are used in order to determine the optimum mesh type. It is shown how the distance between the nodes on the die interface can lead to fluctuations in the extrusion pressure, and how the amplitude of these fluctuations may be reduced by mesh refinement. It is also shown that for the same reduction in area the steady-state extrusion pressure increases linearly as the die angle increases. The results obtained in this study using the Total Lagrangian approach do agree with the observations made by other researchers using the Updated Lagrangian approach. A detailed discussion of this parametric study is presented in Chapter 5.

2. FINITE ELEMENT FORMULATION

2.1 Introdoction

A finite element formulation for problems in solid mechanics may in general incorporate geometric and material non-linearities. These non-linearities may occur separately or simultaneously depending on the geometry of the problem and the type of material used. Problems involving geometric non-linearities in general may be categorized into either large displacement and small strain problems, or large displacements and finite strain problems. In either case, the two most popular formulations for the solution of these problems are the Total Lagrangian and the Updated Lagrangian. The Total Lagrangian formulation refers the displacements, strains and stresses to the initial configuration of the body. In this formulation, the second Piola-Kirchhoff stress and the Green strain are used as stress and strain measures respectively. The Updated Lagrangian formulation involves updating the nodal coordinates at the end of each load increment. In this formulation displacements, strains, and stresses are referred to the previously updated configuration of the body. In the Updated Lagrangian formulation, the measures of stress and strain rates are given in terms of the co-rotational Cauchy stress rate and the spatial strain rate respectively.

The Total Lagrangian and the Updated Lagrangian formulations are different methods of handling the geometric non-linearities. If the material properties used in the analysis are linear then the Updated Lagrangian formulation requires that the problem be solved by using an incremental loading scheme. There is no need for further iterations within each load increment. The Total Lagrangian formulation, on the other hand does not require incremental loading of the material. However, an iterative solution procedure such as the Newton-Raphson method must be used.

In situations where both material and geometric non-linearities are present, both methods require incremental loading of the material as well as some form of iterative solution procedure within each load increment. In general the use of the Total Lagrangian formulation poses more difficulties due to the need for a more complex material model to relate the true stress-strain relationship of the material. Once this problem is overcome (refer to Chapter 3) then a Total Lagrangian formulation can be achieved effectively.

The present version of the program *UNIFES* developed by the author uses the Total Lagrangian formulation for modeling geometric non-linearities. A brief discussion of this formulation is presented in the following section.

2.2 Mathematical Formulation

The fundamental equation in the following arguments is the principle of virtual work. It is first expressed in the Eulerian reference frame as follows:

$$\int_V \delta\epsilon^T \mathbf{t} dV - \int_V \delta\mathbf{u}^T \mathbf{f} dV - \int_S \delta\mathbf{u}^T \mathbf{p} dS = 0 \tag{2.1}$$

where $\mathbf{t}$ is the Cauchy stress tensor, $\delta\mathbf{u}$ is the variation of displacements, $\mathbf{f}$ is the body force, $\mathbf{p}$ is the surface loading, and V and S are the current volume and surface area at time t of the deformed body. In equation (2.1) $\delta\epsilon$ is the variation of the velocity tensor and may be expressed as follows:

$$\delta\epsilon_{ij} = \delta\frac{1}{2}(\frac{\partial u_i}{\partial z_j} + \frac{\partial u_j}{\partial z_i}) \tag{2.2}$$

The corresponding form of the virtual work expression in the Larangian reference frame has been shown to be (Zienkiewicz [1977]; Bathe [1982])

$$\int_{V_0} \delta\mathbf{e}^T \mathbf{s} dV - \int_{V_0} \delta\mathbf{u}^T \mathbf{f} dV - \int_{S_0} \delta\mathbf{u}^T \mathbf{p} dS = 0 \tag{2.3}$$

where $\delta\mathbf{e}$ is the variation of the Lagrangian strain, $\mathbf{s}$ is the second Piola Kirchhoff stress tensor, $\delta\mathbf{u}$ is the variation of displacements, $\mathbf{f}$ is the body force, and $\mathbf{p}$ is the surface loading. In equation (2.3), V_0 and S_0 refer respectively to the initial volume and surface area of the body.

To solve equation (2.3) the finite element method is used. For this purpose the domain of the integration of equation (2.3) is discretized into a finite element mesh. The formulation that follows is applicable to both two dimensional and three dimensional discretizations of equation (2.3) with only minor differences in some of the matrices used.

The relation between the components of the strain and the element nodal displacements $\mathbf{q}_k$ for an element k is expressed by

$$\mathbf{e}_k = (\mathbf{B}'_k + \frac{1}{2}\mathbf{B}''_k)\mathbf{q}_k \tag{2.4}$$

where

$$\mathbf{q}_k^T = [u_1, v_1, w_1, u_2, v_2, w_2, \ldots, u_n, v_n, w_n] \tag{2.5}$$

where n is the number of nodes in the element. In equation (2.4), $\mathbf{B}'_k$ and $\mathbf{B}''_k$ represent the linear and non-linear components of the strain displacement relations.

The theory developed here requires the incremental relationship of the quantities used. Therefore, it is $d\mathbf{e}_k$ that needs to be found. Use is made of expression (2.4) to obtain

$$d\mathbf{e}_k = (\mathbf{B}'_k + \mathbf{B}''_k)d\mathbf{q}_k = \mathbf{B}_k d\mathbf{q}_k \tag{2.6}$$

or

$$d\mathbf{e}_k = \mathbf{B}_k\mathbf{T}_k d\mathbf{q} \tag{2.7}$$

where $\mathbf{q}_k = \mathbf{T}_k\mathbf{q}$ relates the nodal displacements $\mathbf{q}_k$ of the element k to the global nodal displacements $\mathbf{q}$, using the transformation matrix $\mathbf{T}_k$. Substituting equation (2.7) into the constitutive relationship

$$d\mathbf{s} = \mathbf{D}d\mathbf{e} \tag{2.8}$$

we obtain:

$$d\mathbf{s}_k = \mathbf{D}_k\mathbf{B}_k\mathbf{T}_k d\mathbf{q} \tag{2.9}$$

and

$$\mathbf{s}_k = \int_0^t \mathbf{D}_k\mathbf{B}_k\mathbf{T}_k d\mathbf{q} \tag{2.10}$$

The time integration above is along the deformation path.

The relationship between the displacement field $\mathbf{u}_k$ within the element and the element nodal diplacements $\mathbf{q}_k$ is obtained through the usual parametric shape functions, N_i, of the element as follows:

$$\mathbf{u}_k = \mathbf{N}_k\mathbf{q}_k \tag{2.11}$$

where

$$\mathbf{u}_k = \left\{ \begin{array}{c} u \\ v \\ w \end{array} \right\}_k ; \tag{2.12}$$

$$\mathbf{N}_k = \begin{pmatrix} N_1 & 0 & 0 & N_2 & 0 & 0 & \dots & N_n & 0 & 0 \\ 0 & N_1 & 0 & 0 & N_2 & 0 & \dots & 0 & N_n & 0 \\ 0 & 0 & N_1 & 0 & 0 & N_2 & \dots & 0 & 0 & N_n \end{pmatrix} \tag{2.13}$$

and $\mathbf{q}_k$ is given by equation (2.5).

Substituting equation (2.6) in (2.3) and realizing that $\delta\mathbf{e}_k^T = \delta\mathbf{q}^T\mathbf{T}_k^T\mathbf{B}_k^T$ and that $\delta\mathbf{u}_k^T = \delta\mathbf{q}^T\mathbf{T}_k^T\mathbf{N}_k^T$, we obtain:

$$\delta\mathbf{q}^T \sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} \mathbf{B}_k^T\mathbf{s}_k dV - \delta\mathbf{q}^T \sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} \mathbf{N}_k^T\mathbf{f} dV - \delta\mathbf{q}^T \sum_{k=1}^{m} \mathbf{T}_k^T \int_{S_{0k}} \mathbf{N}_k^T\mathbf{p} dS = 0 \tag{2.14}$$

where m is the number of elements in the mesh. Eliminating the virtual displacement $\delta\mathbf{q}^T$ from equation (2.14), we obtain:

$$\sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} \mathbf{B}_k^T\mathbf{s}_k dV - \sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} \mathbf{N}_k^T\mathbf{f} dV - \sum_{k=1}^{m} \mathbf{T}_k^T \int_{S_{0k}} \mathbf{N}_k^T\mathbf{p} dS = 0 \tag{2.15}$$

Note that the first term of the left hand side of (2.15) is a non-linear function of $\mathbf{q}$. This is because both $\mathbf{B}_k^T$ and $\mathbf{s}_k$ are functions of $\mathbf{q}$. Equation (2.15) can be written as:

$$\mathbf{Q}(\mathbf{q}) = \mathbf{R} \tag{2.16}$$

where

$$\mathbf{Q}(\mathbf{q}) = \sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} \mathbf{B}_k^T \mathbf{s}_k dV \tag{2.17}$$

and

$$\mathbf{R} = \sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} \mathbf{N}_k^T \mathbf{f} dV + \sum_{k=1}^{m} \mathbf{T}_k^T \int_{S_{0k}} \mathbf{N}_k^T \mathbf{p} dS \tag{2.18}$$

Differentiating equation (2.17) we obtain:

$$d\mathbf{Q}(\mathbf{q}) = \sum_{k=1}^{m} \mathbf{T}_k^T \int_{V_{0k}} (d\mathbf{B}_k^T \mathbf{s}_k + \mathbf{B}_k^T d\mathbf{s}_k) dV \tag{2.19}$$

or

$$d\mathbf{Q}(\mathbf{q}) = \sum_{k=1}^{m} \mathbf{T}_k^T \mathbf{K}_k \mathbf{T}_k d\mathbf{q} \tag{2.20}$$

where

$$\mathbf{K}_k = \mathbf{K}_k^s + \widehat{\mathbf{K}}_k \tag{2.21}$$

is known as the tangent stiffness matrix of element k.

The solution of equation (2.16) requires obtaining the derivatives

$$\frac{\partial \mathbf{Q}_i}{\partial \mathbf{q}_j} = \mathbf{K}_{ij} \tag{2.22}$$

where i =1, 2, . . . , n, and j =1, 2, . . ., n, where n is the number of degrees of freedom for the mesh. The global tangent stiffness matrix $\mathbf{K}$ is obtained from expression (2.20) where

$$\mathbf{K} = \sum_{k=1}^{m} \mathbf{T}_k^T (\mathbf{K}_k^s + \widehat{\mathbf{K}}_k) \mathbf{T}_k \tag{2.23}$$

The solution of the system of functional equations (2.16) is then numerically obtained by applying incremental steps of loading and performing iterations within each load increment. The full Newton-Raphson iterative solution procedure is used in this work, whereby the tangent stiffness matrix is evaluated for each iteration. An alternative to this method is the modified Newton-Raphson procedure, where the stiffness matrix is updated for a selected number of iterations. For reasons that would become obvious in Chapter 3, it was determined that the modified Newton-Raphson iterative solution procedure would lead to higher execution time in this study and was therefore avoided. A more detailed discussion of various iterative solution procedures is given by Zienkiewicz [1977] and Bathe [1982].

3. MATERIAL MODEL AND ITS IMPLEMENTATION

3.1 Introduction

In the last three decades or so, the theory of plasticity has been generalized for elasto-plastic solids with large deformations. Green and Naghdi [1965, 1966] generalized the theory for an elasto-plastic continuum where full use is made of the thermodynamical equations. Hill [1958], in his paper, generalized the theory to large deformations without the use of the thermodynamical equations.

There has been a consistant effort to use the Eulerian coordinate system with the Cauchy stress tensor t_{ij} in the analysis of large elasto-plastic deformations of solids (Lee [1969], and Nemat-Nasser [1979]). This is primarily because of the physical interpretation of the Cauchy stress tensor as the true stress which for the case of small strains can be approximated by the engineering stress tensor σ_{ij}. This conclusion is dependent only on the physical perceptions since mathematically both the Cauchy stress and the second Piola-Kirchhoff stress tensors reduce to the engineering stress for the case of small deformations.

In the case of small strain plasticity, the flow condition is postulated in terms of the traditional engineering stress and engineering strain rate. The corresponding yield function is expressed in terms of the engineering stress. For finite strain deformations, there is no unique approach for the extrapolation of the small strain plasticity flow rule and yield condition into the appropriate corresponding ones for finite strain plasticity.

One approach is when the flow rule is postulated in terms of the second Piola-Kirchhoff stress tensor and the material strain rate, while the yield condition is in terms of the second Piola-Kirchhoff stress tensor. The appropriate equations are obtained by direct substitution into the same functional form used for the small strain plasticity theory. This leads to the use of the material stress rate which is quite simpler to use in solution of problems when compared to the appropriate co-rotational stress rate in the Eulerian reference frame.

The commonly followed approach is the Eulerian formulation. The flow rule is postulated in terms of the Cauchy stress tensor and the spatial strain rate. The yield function is expressed in terms of the Cauchy stress. Both of these expressions are obtained by direct substitution into the same functional form used for the small strain plasticity theory. Explicit transformations of these finite-strain plasticity equations into the Lagrangian reference frame leads to a totally different functional set of equations than those described in the previous paragraph.

The ultimate choice for the appropriate formulation of the constitutive equations for finite strain plasticity does not lie on either the mathematical simplicity of the expressions or the physical interpretation of the conversion of the small-strain expressions to finite-strain expressions. The choice solely depends on the experimental evidence in the range of finite strains for which the constitutive model is to be applied (Voyiadjis [1988]).

Based on the experiments performed on metals by Voyiadjis [1984], for finite strain deformations (up to 20 percent), the von Mises yield criterion was found to be appropriate. Nevertheless, when the von Mises yield criterion was directly expressed in terms of the second Piola-Kirchhoff stress tensor in the Lagrangian reference frame, the resulting yield criterion proved to be inadequate. The additional presence of the deformation gradient in the yield criterion proved to be imperative for the case of the Lagrangian description of the yield criterion. This was achieved by first directly interpreting the von Mises yield criterion in terms of the Cauchy stress tensor and then converting tensorially the resulting expression into the second Piola-Kirchhoff stress space.

Although the proposed theory is postulated for finite strains, in this work it is assumed that the elastic component of the strain tensor is relatively small when compared to the plastic strain component. In the uniaxial load reversal experiments performed by Voyiadjis [1984], small elastic strains were encountered. Consequently, the choice of the proper linear elastic relation between the appropriate stress and appropriate strain was immaterial. No apparent physical significance will be gained in this case whether a linear relation between the Cauchy stress and the elastic spatial strain or the second Piola-Kirchhoff stress and the elastic material strain is postulated. Due to mathematical simplicity, and without violating the experimental observations, the latter choice was made.

3.2 Preliminary Definitions and Relations

The following is a brief summary of the concepts of continuum mechanics which are used in this work. This section is based on the monograph by Truesdell and Toupin [1960]. All subscripts used in this chapter refer to tensorial notation and may have values 1,2, and 3 unless otherwise stated.

Referring to Figure 3.1, let B_0 be the initial, or the undeformed, state at time $t = 0$, and B the current or the deformed state at some time t. The coordinates x_A and z_k are the initial and the deformed coordinates corresponding to the undeformed state B_0 and the deformed state B respectively. The displacement of the body is described by

$$z_k = z_k(x_1, x_2, x_3, t) \tag{3.1}$$

or

$$x_A = x_A(z_1, z_2, z_3, t) \tag{3.2}$$

The usual assumptions of single-valuedness and continuity with

$$0 < det|\frac{\partial z_k}{\partial x_A}| = J < \infty \tag{3.3}$$

are made with regard to eqations (3.1) and (3.2).

In the following discussions, two coordinate systems are employed. Spatial or Eulerian coordinates describe the location of a point in the material using the instantaneous or deformed state as reference. The quantities referred to these coordinates are indicated by lower case Latin suffixes. Material or Lagrangian coordinates describe the location of a point with respect to the original (undeformed) state. All quantities referred to the Lagrangian coordinates are indicated by capital Latin suffixes.

In Figure 3.1, the components of the displacement $\mathbf{u}$ are related to z_k and x_A by

$$u_i = z_i - x_i \tag{3.4}$$

The displacement vectors in the material and spatial forms, are given by

$$u_A = u_A(x_1, x_2, x_3, t) \tag{3.5}$$

and

$$u_k = u_k(z_1, z_2, z_3, t) \tag{3.6}$$

respectively. The expressions for the velocity vectors are

$$v_A = \frac{\partial u_A}{\partial t}(x_1, x_2, x_3, t) \tag{3.7}$$

and

$$v_k = \frac{\partial z_k}{\partial t}(x_1, x_2, x_3, t) \tag{3.8}$$

The material form of the velocity is expressed by equation (3.7), while its spatial form is obtained by using equation (3.2) to replace the x with z, hence obtaining expression (3.8).

The material strain tensor which is often referred to as the Green strain tensor is given by the following expression

$$e_{AB} = \frac{1}{2}(\delta_{k\ell}\frac{\partial z_k}{\partial x_A}\frac{\partial z_\ell}{\partial x_B} - \delta_{AB}) \tag{3.9}$$

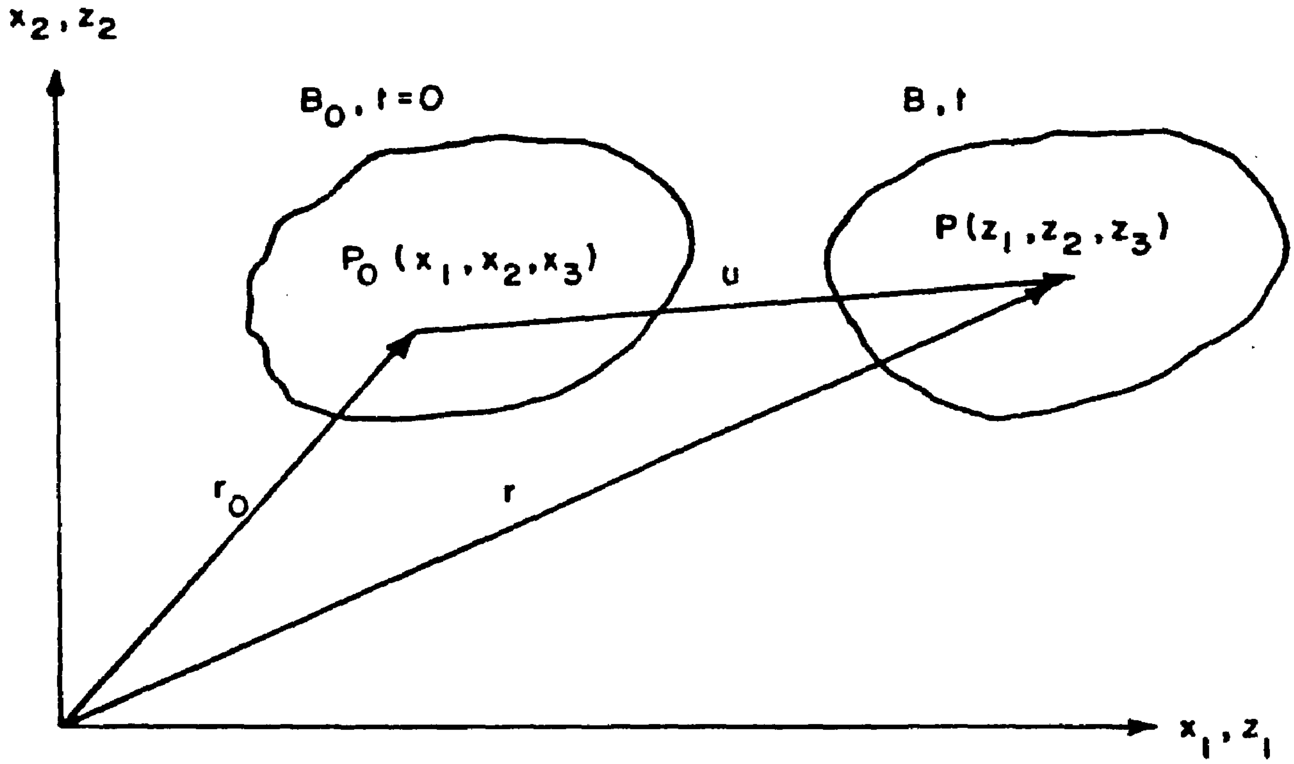


Figure 3.1 *Coordinate Systems and Description of Displacement.*

or, in terms of the displacement vector $\mathbf{u}$,

$$e_{AB} = \frac{1}{2}\left(\frac{\partial u_A}{\partial x_B} + \frac{\partial u_B}{\partial x_A} + \frac{\partial u_C}{\partial x_A}\frac{\partial u_C}{\partial x_B}\right) \tag{3.10}$$

Referring to Figure 3.1, the changes of length of a line segment $d\ell_0$ at P_0 can be computed as

$$d\ell^2 - d\ell_0^2 = 2e_{AB}dx_A dx_B \tag{3.11}$$

or

$$\frac{d\ell^2 - d\ell_0^2}{d\ell_0^2} = 2e_{AB}\ell_{0_A}\ell_{0_B} \tag{3.12}$$

where ℓ_{0_A} are the components of a unit vector along $d\ell_0$.

The spatial strain tensor $h_{k\ell}$ which is also known as the Almansi's strain tensor is defined as

$$h_{k\ell} = \frac{1}{2}\left(\delta_{k\ell} - \frac{\partial x_A}{\partial z_k}\frac{\partial x_B}{\partial z_\ell}\delta_{AB}\right) \tag{3.13}$$

or

$$h_{k\ell} = \frac{1}{2}\left(\frac{\partial u_k}{\partial z_\ell} + \frac{\partial u_\ell}{\partial z_k} - \frac{\partial u_m}{\partial z_k}\frac{\partial u_m}{\partial z_\ell}\right) \tag{3.14}$$

In terms of $h_{k\ell}$, the length changes are

$$d\ell^2 - d\ell_0^2 = 2h_{k\ell}dz_k dz_\ell \tag{3.15}$$

or

$$\frac{d\ell^2 - d\ell_0^2}{d\ell^2} = 2h_{k\ell}\ell_k\ell_\ell \tag{3.16}$$

where ℓ_k are the components of the unit vector along $d\ell$ at P.

The measure of extension which is defined as

$$\epsilon = \frac{d\ell - d\ell_0}{d\ell_0} \tag{3.17}$$

is frequently used in describing the results of uniaxial testing of various materials. ϵ can be related to the components of the material or spatial strain tensors. For example, for a line segment $d\ell_0$ whose initial direction at P_0 was parallel to x_1,

$$\epsilon = \sqrt{(1 + 2e_{11})} - 1 \tag{3.18}$$

The determinant of deformation Jacobian J expressed in equation (3.3) is equal to the volumetric strain dV/dV_0.

The material strain rate tensor is given as

$$\dot{e}_{AB} = \frac{\partial e_{AB}}{\partial t} \tag{3.19}$$

and hence

$$\frac{D}{Dt}(d\ell)^2 = 2\dot{e}_{AB}dx_A dx_B \tag{3.20}$$

where $\frac{D}{Dt}$ is the material time derivative.

The spatial strain rate tensor is given as

$$d_{k\ell} = \frac{1}{2}(\frac{\partial v_k}{\partial z_\ell} + \frac{\partial v_\ell}{\partial z_k}) \tag{3.21}$$

and hence

$$\frac{D}{Dt}(d\ell)^2 = 2d_{k\ell}dz_k dz_\ell \tag{3.22}$$

The condition of incompressability is expressed in terms of $d_{k\ell}$ by the following simple expression:

$$d_{kk} = 0 \tag{3.23}$$

The material strain rate tensor may be expressed in terms of the spatial strain rate tensor by the following relation:

$$\dot{e}_{AB} = d_{k\ell}\frac{\partial z_k}{\partial x_A}\frac{\partial z_\ell}{\partial x_B} \tag{3.24}$$

Let $t_{(n)}$ denote the stress vector, or serface traction acting on the area element P with the unit normal vector n (note: $t_{(n)}$ is force per unit area of the deformed body). In terms of the spatial or Cauchy stress tensor $t_{k\ell}$, the components of $t_{(n)}$ are

$$t_{(n)\ell} = t_{k\ell}n_k \tag{3.25}$$

In this work, the material, or the second Piola-Kirchhoff stress tensor s_{AB} will be used. Its definition is

$$s_{AB} = t_{k\ell} J \frac{\partial x_A}{\partial z_k} \frac{\partial x_B}{\partial z_\ell} \tag{3.26}$$

The components of the stress vector $p_{(n)}$ defined as the surface traction for the undeformed body, i.e.,

$$p_{(n)} = t_{(n)} \frac{dA}{dA_0} \tag{3.27}$$

can be expressed in terms of the Cauchy stress $t_{k\ell}$ or the second Piola Kirchhoff stress tensor s_{AB} in the following maner:

$$p_{(n)\ell} = t_{k\ell} J \frac{\partial x_A}{\partial z_k} n_{0_A} = s_{AB} \frac{\partial z_\ell}{\partial x_B} n_{0_A} \tag{3.29}$$

An objective stress rate tensor in terms of s_{AB} may be expressed as

$$\dot{s}_{AB} = \frac{\partial s_{AB}}{\partial t} \tag{3.30}$$

where s_{AB} in equation (3.30) is a function of the material coordinates x_i and time t.

3.3 Constitutive Model for Elasto-Plastic Behavior of Metals

The constitutive equations used in *UNIFES* are based on the general theory of plasticity at large strains in the Lagrangian reference frame (Voyiadjis [1984] and Voyiadjis and Kiousis [1986]).

A yield criterion of the von Mises type in terms of the Cauchy stresses is used. This yield function combines both isotropic and kinematic hardening of the Prager-Ziegler type (Prager [1956]; Ziegler [1959]; Shield and Ziegler [1958]) and is independent of the volumetric component of the Cauchy stress tensor. This yield function is expressed as

$$f = \frac{1}{2}(\bar{t}_{k\ell} - \bar{\alpha}_{k\ell})(\bar{t}_{k\ell} - \bar{\alpha}_{k\ell}) - c\kappa - \frac{\sigma_y^2}{3} = 0 \tag{3.31}$$

where $\bar{\alpha}_{k\ell}$ is the deviatoric component of the Eulerian shift stress tensor such that, $\bar{\alpha}_{k\ell} = \alpha_{k\ell} - \frac{1}{3}\alpha_{ii}\delta_{k\ell}$, and $\bar{t}_{k\ell}$ is the deviatoric component of the Cauchy stress tensor such that $\bar{t}_{k\ell} = t_{k\ell} - \frac{1}{3}t_{ii}\delta_{k\ell}$. The constant σ_y in equation (3.31), is the uniaxial yield strength of the material which is obtained through simple tension tests and the parameter c is a constant which controls the extend of the isotropic hardening.

The plastic work κ, used in equation (3.31) is obtained through the use of the following expression:

$$\kappa = \int_0^t t_{k\ell} d''_{k\ell} dt \tag{3.32}$$

where $d''_{k\ell}$ is the plastic component of the spatial strain rate tensor $d_{k\ell}$ given by equation (3.21).

The corresponding associated flow rule is described as

$$d''_{k\ell} = \dot{\mathrm{L}} \frac{\partial f}{\partial t_{k\ell}} = \dot{\mathrm{L}}(\bar{t}_{k\ell} - \bar{\alpha}_{k\ell}) \tag{3.33}$$

where $\dot{\mathrm{L}}$ is a scalar function. The absence of plastic volumetric strain can be verified in equation (3.33) where $d''_{kk} = 0$. Equations (3.31) and (3.33) incorporate a number of generally accepted assumptions regarding the plastic deformation of metals. This constitutive model produces no plastic volumetric strains. The hydrostatic state of stress even at large strains has no effect on the plastic behavior of metals in this model. Finally, the von Mises yield criterion and the associated flow rule are satisfactory forms of equations (3.31) and (3.33), respectively, in the small deformation theory of plasticity of metals.

The constitutive model given by equations (3.31) and (3.33) is in an Eulerian reference frame. For this model to be applied in a Lagrangian frame of reference, coordinate transformations need to be performed.

In order to obtain the equivalent form of equations (3.31) and (3.33) with respect to the Lagrangian reference frame, certain relations need to be used. Let

$$\alpha_{k\ell} = A_{AB} \frac{\partial z_k}{\partial x_A} \frac{\partial z_\ell}{\partial x_B} J^{-1} \tag{3.34}$$

where

$$A_{AB} = \int_0^t \dot{A}_{AB} dt \tag{3.35}$$

is the equivalent Lagrangian counterpart of the spatial shift stress tensor $\alpha_{k\ell}$. Also,

$$\dot{e}''_{AB} = d''_{k\ell} \frac{\partial z_k}{\partial x_A} \frac{\partial z_\ell}{\partial x_B} \tag{3.36}$$

where $\dot{e}''_{AB}$ is obtained from the following decomposition of the material strain rate tensor

$$\dot{e}_{AB} = \dot{e}'_{AB} + \dot{e}''_{AB} \tag{3.37}$$

The terms e'_{AB} and e''_{AB} are the elastic and plastic components of the strain tensor e_{AB} respectively. In general, the kinematic interpretation of these two components is not the usual one. Instead they are simple mathematical quantities defined by the constitutive law. Nevertheless, when the elastic strains are small compared to the plastic ones (an assumption that is satisfied in a considerable number of applications), the decomposition of equation (3.37) acquires the usual physical meaning.

Equation (3.31) may now be expressed in the Lagrangian reference frame as

$$f = f(1) + f(2) + f(3) \tag{3.38}$$

where

$$f(1) = \frac{1}{2}J^{-2}[s_{AB}s_{CD}C_{AC}C_{BD} - \frac{1}{3}s_{AB}s_{CD}C_{AB}C_{CD}] \tag{3.39a}$$

$$f(2) = J^{-2}[-s_{AB}A_{CD}C_{AC}C_{BD} + \frac{1}{3}s_{AB}A_{CD}C_{AB}C_{CD}]$$

$$+ \frac{1}{2}J^{-2}[A_{AB}A_{CD}C_{AC}C_{BD} - \frac{1}{3}A_{AB}A_{CD}C_{AB}C_{CD}] \tag{3.39b}$$

$$f(3) = -\frac{\sigma_y^2}{3} - c\kappa \tag{3.39c}$$

where s_{AB} is the second Piola-Kirchhoff stress tensor, given by equation (3.26) and C_{AB} is the Green's deformation tensor such that

$$C_{AB} = \frac{\partial z_k}{\partial x_A}\frac{\partial z_k}{\partial x_B} = 2e_{AB} + \delta_{AB} \tag{3.40}$$

where δ_{AB} is the Kronecker delta. The equivalent plastic work κ given by equation (3.32) may be obtained by the following relation:

$$\kappa = \int_0^t \frac{1}{J}s_{AB}\dot{e}''_{AB}\,dt \tag{3.41}$$

The yield function expressed by equation (3.38) which is an interpretation of the von Mises yield criterion in the Lagrangian reference frame, best interprets the behavior of metals at finite strains. This was demonstrated by Voyiadjis [1984] primarily for aluminum alloys (2024 T4 and 6061 T6) and steel (1180 cold rolled).

In addition, it has been shown by Voyiadjis [1984] that equation (3.33) is equivalent to the Lagrangian expression for the flow rule which is defined as

$$\dot{e}''_{AB} = \dot{\Lambda}\frac{\partial f}{\partial s_{AB}} \tag{3.42}$$

where

$$\dot{\Lambda} = \dot{L}J \tag{3.43}$$

when the yield funtion is expressed as equation (3.38).

Based on the concepts proposed by Shield and Ziegler [1958], it is assumed that the yield surface moves in the direction of the radius connecting the center of the yield surface with the current stress point P on the yield surface(Figure 3.2). Consequently, the hardening rule is expressed by

$$\dot{A}_{AB} = (s_{AB} - A_{AB})\dot{\mu} \tag{3.44}$$

where $\dot{\mu}$ is a positive scalar function and is calculated by assuming that the projection of $\dot{A}_{AB}$ on the stress gradient of the yield surface equals to $b\dot{e}''_{AB}$. The procedure to obtain $\dot{\mu}$ is outlined as follows:

$$b\dot{e}''_{AB} = \dot{A}_{CD} \frac{\dfrac{\partial f}{\partial s_{CD}} \dfrac{\partial f}{\partial s_{AB}}}{\dfrac{\partial f}{\partial s_{MN}} \dfrac{\partial f}{\partial s_{MN}}} \tag{3.45}$$

where b is a material parameter which is related to the kinematic hardening characteristics of the material. Through equations (3.44) and (3.45) the plastic component of the Lagrangian strain rate tensor is determined to be

$$\dot{e}''_{AB} = \dot{\Lambda} \frac{\partial f}{\partial s_{AB}} = \frac{1}{b}(s_{CD} - A_{CD})\dot{\mu} \frac{\dfrac{\partial f}{\partial s_{CD}} \dfrac{\partial f}{\partial s_{AB}}}{\dfrac{\partial f}{\partial s_{MN}} \dfrac{\partial f}{\partial s_{MN}}} \tag{3.46}$$

hence, the value of $\dot{\mu}$ is obtained to be

$$\dot{\mu} = \dot{\Lambda} b \frac{\dfrac{\partial f}{\partial s_{MN}} \dfrac{\partial f}{\partial s_{MN}}}{(s_{CD} - A_{CD}) \dfrac{\partial f}{\partial s_{CD}}} \tag{3.47}$$

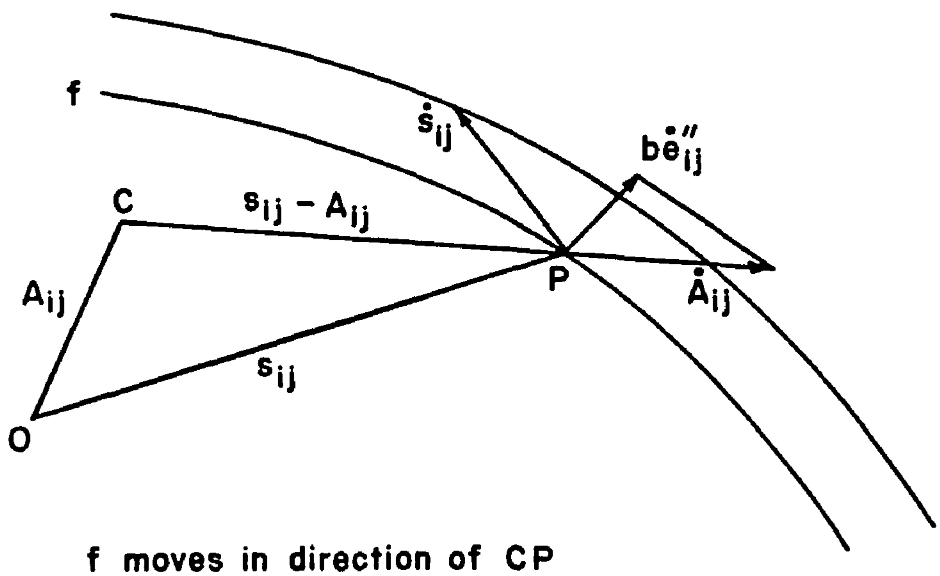


Figure 3.2 *Modification of Prager's Kinematic Hardening Rule by Shield and Ziegler [1958].*

The development of this theory is based on the concept that the yield function given by equation (3.38) is at all times equivalent to its spatial counterpart, equation (3.31). Nevertheless, the evolution of its terms, expressed by equations (3.42) and (3.44), does not necessarily yield equivalent evolution to the more usual spatial expressions. Although, in the absence of kinematic hardening, it can be shown that equation (3.33) is equivalent to the Lagrangian expression (3.42), equation (3.44) is not equivalent to the usual Ziegler type shift evolution equation $\widehat{\alpha}_{k\ell} = (t_{k\ell} - \alpha_{k\ell})\dot{\mu}$, where $\widehat{\alpha}_{k\ell}$ implies the Jaumann rate. Further study is needed in this direction so that the implications of equation (3.44) are fully understood and properly evaluated. Nevertheless, one should realize that the development of this formulation is consistently carried out in the material reference frame, where the yield function is defined by equation (3.38), and the evolution of its terms are defined by equations (3.42) and (3.44).

The parameter $\dot{\Lambda}$ is calculated from the consistency condition:

$$\dot{f} \equiv \dot{f}(\mathbf{s}, \mathbf{A}, \mathbf{e}, \kappa, J) = 0 \tag{3.48}$$

hence

$$\frac{\partial f}{\partial s_{AB}}\dot{s}_{AB} + \frac{\partial f}{\partial A_{AB}}\dot{A}_{AB} + \frac{\partial f}{\partial e_{AB}}\dot{e}_{AB} + \frac{\partial f}{\partial \kappa}\dot{\kappa} + \frac{\partial f}{\partial J}\dot{J} = 0 \tag{3.49}$$

Use of the above consistency condition requires evaluation of $\dot{J}$. This is achieved by expressing the Jacobian J in terms of the strain invariants and subsequently finding an expression for $\dot{J}$ as follows:

$$J = \frac{dV}{dV_0} = [1 + 2I(1 + I + \frac{2}{3}I^2) - 4II(1 + 2I) + 8III]^{\frac{1}{2}} \tag{3.50}$$

where

$$I = e_{AA} \tag{3.51a}$$

$$II = \frac{1}{2}e_{AB}e_{AB} \tag{3.51b}$$

$$III = \frac{1}{3}e_{AB}e_{BC}e_{AC} \tag{3.51c}$$

where I, II, III are the first, second and third strain invariants respectively. Hence, $\dot{J}$ may be evaluated by taking the time derivative of expression (3.50). Therefore,

$$\dot{J} = R_{CD}\dot{e}_{CD} \tag{3.52}$$

where

$$R_{CD} = \frac{1}{2J}[2\delta_{CD} + 4\delta_{CD}e_{KK} - 4e_{CD} + 8e_{DR}e_{RC} - 8e_{CD}e_{KK} \\ - 4\delta_{CD}e_{OP}e_{OP} + 4\delta_{CD}e_{LL}e_{KK}] \tag{3.53}$$

Following the procedure outlined by Voyiadjis [1984], the expression for $\dot{\Lambda}$ is determined to be

$$\dot{\Lambda} = [\frac{1}{Q}(E_{ABCD}\frac{\partial f}{\partial s_{AB}} + \frac{\partial f}{\partial e_{CD}} + \frac{\partial f}{\partial J}R_{CD})]\dot{e}_{CD} \tag{3.54}$$

where

$$Q = E_{ABCD}\frac{\partial f}{\partial s_{CD}}\frac{\partial f}{\partial s_{AB}} - \frac{\partial f}{\partial \kappa}s_{AB}\frac{\partial f}{\partial s_{AB}}J^{-1} \\ - \frac{\partial f}{\partial A_{AB}}(s_{AB} - A_{AB})b\frac{\frac{\partial f}{\partial s_{MN}}\frac{\partial f}{\partial s_{MN}}}{(s_{QR} - A_{QR})\frac{\partial f}{\partial s_{QR}}} \tag{3.55}$$

In expressions (3.54) and (3.55), the fourth order elastic stress-strain tensor E_{ABCD} has the following form:

$$E_{ABCD} = \lambda\delta_{AB}\delta_{CD} + G(\delta_{AC}\delta_{BD} + \delta_{AD}\delta_{BC}) \tag{3.56}$$

where λ and G are Lame's constants. If a linear elastic relation is assumed between the second Piola-Kirchhoff stress tensor and the material elastic strain tensor

$$s_{AB} = E_{ABCD}e'_{CD} \tag{3.57}$$

Equation (3.57) will be referred to as the Lagrangian linear elasticity.

The elasto-plastic stiffness tensor $\mathbf{D}^{ep}$ which corresponds to the yield function given by relation (3.38), is given by the following expression:

$$D^{ep}_{MNPQ} = E_{MNPQ} - E_{MNCD} \frac{\dfrac{\partial f}{\partial s_{AB}} \dfrac{\partial f}{\partial s_{CD}} E_{ABPQ} + \dfrac{\partial f}{\partial e_{PQ}} \dfrac{\partial f}{\partial s_{CD}} + \dfrac{\partial f}{\partial s_{CD}} \dfrac{\partial f}{\partial J} R_{PQ}}{Q} \tag{3.59}$$

The incremental elasto-plastic constitutive relation can now be expressed as follows:

$$\dot{s}_{AB} = D^{ep}_{ABCD} \dot{e}_{CD} \tag{3.60}$$

The derivatives of the yield function with respect to s_{AB}, A_{AB}, ϵ_{AB}, κ, and J used in the above derivations are obtained through the following expressions:

$$\begin{aligned} \frac{\partial f}{\partial s_{AB}} &= \frac{1}{J^2}[(A_{CD}C_{CD} - s_{CD}C_{CD})C_{AB} + s_{CD}C_{AC}C_{BD} \\ &\quad - A_{CD}C_{AC}C_{BD}] \end{aligned} \tag{3.61}$$

$$\begin{aligned} \frac{\partial f}{\partial A_{AB}} &= \frac{1}{J^2}[(s_{CD}C_{CD} - A_{CD}C_{CD})C_{AB} + A_{CD}C_{AC}C_{BD} \\ &\quad - s_{CD}C_{AC}C_{BD}] \end{aligned} \tag{3.62}$$

$$\begin{aligned} \frac{\partial f}{\partial e_{AB}} &= \frac{2}{J^2}[(s_{CD}C_{CD} - A_{CD}C_{CD})A_{AB} - (s_{CD}C_{CD} - A_{CD}C_{CD})s_{AB} \\ &\quad + s_{DB}C_{CD}s_{CA} + A_{CA}A_{DB}C_{CD} - 2A_{DB}s_{CA}C_{CD}] \end{aligned} \tag{3.63}$$

$$\frac{\partial f}{\partial J} = -\frac{2}{J}(f(1) + f(2)) \tag{3.64}$$

$$\frac{\partial f}{\partial \kappa} = -c \tag{3.65}$$

As can be seen from equation (3.58), the elasto-plastic stiffness tensor D_{ABCD} is non-symmetric. This requires the use of a non-symmetric equation solver which in turn increases the execution time and the storage requirements. The solution method used for calculation of stresses and plastic strains is based on an elastic-predictor radial-corrector algorithm with subincrementation. Due to the complexity of the model, and depending on the number of subincrementations used, the computer time required for calculation of stresses may be greater than the time required to solve the system of simultaneous equations and calculating the displacements. For this reason it is determined that a full Newton-Raphson iterative solution procedure results in greater efficiency than the modified Newton-Raphson technique. This is primarily due to the fact that full Newton-Raphson algorithms converge to the solution considerably faster than the modified Newton-Raphson algorithms. Consequently, stress calculations are performed for a fewer number of iterations.

3.4 Numerical Implementation of the Material Model

The Total Lagrangian plasticity model used in this work is successfully implemented in the finite element program *UNIFES* developed by the author. This program is primarily used for the solution of general boundary value problems in metal forming. An efficient implementation of this model is not an easy task. This is due to the extensive use of second and fourth order tensors in the principal equations describing this model (equations (3.38) through (3.65)). Furthermore, since tensors can be represented by multidimensional arrays with their subscripts ranging from one to three, due to the short vector lengths, vectorization of equations involving tensors will generate less efficient codes than the equivalent scalar operations. Later in the next section it is shown how some portions of the material model described in the previous section may be vectorized by using some characteristics of the FORTRAN language. However, we should first look at the sequence of operations required for implementing this material model.

Implementation of any material model in a finite element program involves two stages. The first stage is the calculation of the stress-strain or the incremental stress-strain stiffness matrix **D**. This is done for the purpose of assembling the global stiffness matrix and subsequent evaluation of the displacements. The second stage of implementation involves evaluation of stresses and any other vital information which is explicitly derived from the constitutive relations. In plasticity for example, calculation of the elastic and plastic components of the strain increment vector and the hardening parameters is an inherent part of the second stage of implementation. This stage of implementation follows the calculation of the strain or the strain increment vector. For both stages the **D** matrix and the stresses are sequentially evaluated at each integration point.

The control program for the plasticity module of the program *UNIFES* is the MISES subroutine. This subroutine is divided into two segments each of which operates independently and have entry names MISES1 and MISES2. Entry MISES1 is the control routine for evaluation of the elasto-plastic stiffness matrix (i.e. the first stage of implementation), whereas entry MISES2 is the control program for the second stage of implementation. A listing of the plasticity module of *UNIFES* is provided in Appendix C.

Table 3.1 lists the sequence of computations required for evaluation of the incremental elasto-plastic stiffness tensor D^{ep}_{ABCD} which is expressed by equation (3.59), and subsequent conversion of this tensor to a second order incremental elasto-plastic stiffness matrix **D**. A detailed discussion of each step is presented here with references made to the appropriate subroutines listed in Appendix C.

As seen in Table 3.1, the first step of computations requires the evaluation of the fourth order elastic stiffness tensor. This tensor is needed even if the material is determined to behave plastically (see equation (3.59)). Since the fourth order elastic tensor E_{ABCD} is not path or history dependent it is evaluated only once and used over again for all the integration points composed of the same material. If the material used at the current integration point is different from the previous point, the E_{ABCD} tensor is recalculated using the new material properties. Subroutine ADMAT is used for evaluation of the AD array which is equivalent to E_{ABCD} as presented by equation (3.56).

In the second step, the yield flag, IYIELD, is checked in order to determine if the material has previously yielded. It is important to note that the control subroutine MISES1 checks the yield flag starting with the second load increment. The first load increment is always assumed to be elastic. At the start of the program, IYIELD is initialized to zero for all integration points; its value is then appropriately altered during the course of the stress calculations (second stage of implementation). A value of one for IYIELD indicates that the material is currently yielding and that the incremental elasto-plastic stiffness matrix needs to be evaluated. A zero value for IYIELD indicates that the material is currently elastic. Hence, steps two through ten in Table 3.1, may be by-passed.

In step 3, the values of the total stress, total strain and the total shift stress tensors along with the plastic work at the end of the last iteration are retrieved from the disk or memory. Notice that these values are evaluated at the end of the previous iteration which may or may not be the converged state in the Newton-Raphson iterative solution process. Program *UNIFES* is capable of storing and retrieving these values from disk, or alternatively from a memory buffer depending on the limitations of the computer system used. The user has the option of selecting the storage method. The retrieved information is then converted to tensor form (3×3 matrix) by making appropriate calls to subroutine TENSOR. The control program MISES1 makes the necessary calls to the I/O facilities for retrieving the above mentioned information.

The Green's tensor C_{AB} is evaluated in step 4 of the computations. This task is performed by the control program MISES1. After completion of steps one through four, subroutine DEFJAC is called by the control program MISES1

Table 3.1. *Sequence of Computations Required for the First Stage of Implementation (Evaluation of the* **D** *Matrix).*

STEP 1. Evaluate the fourth order elastic stiffness tensor E_{ABCD} as defined by equation (3.56).

STEP 2. Check the yield flag IYIELD at the integration point (this flag is set equal to one if during the stress calculations yielding is detected, otherwise, it will be zero).

```
if IYIELD=0 then                (material is elastic)
        go to STEP 11;
else                            (material is plastic)
        go to STEP 3;
end;
```

STEP 3. Read s_{AB}, e_{AB}, A_{AB}, and κ from the memory or disk and convert all vectors to tensors.

STEP 4. Evaluate the Green's tensor C_{AB} through equation (3.40).

STEP 5. Evaluate the strain invariants using expressions (3.51a-c) and calculate the deformation Jacobian J through equation (3.50).

STEP 6. Evaluate the R_{AB} tensor using expression (3.53).

STEP 7. Evaluate the yield function using equations (3.38) and (3.39).

STEP 8. Evaluate the partial derivatives $\frac{\partial f}{\partial e_{AB}}, \frac{\partial f}{\partial J}, \frac{\partial f}{\partial s_{AB}}, \frac{\partial f}{\partial A_{AB}}$, and $\frac{\partial f}{\partial \kappa}$ using equations (3.61) through (3.65).

STEP 9. Evaluate Q using equation (3.55).

STEP 10. Evaluate the fourth order elasto-plastic stiffness tensor D^{ep}_{ABCD} through expression (3.59).

STEP 11. Convert the fourth order elasto-plastic tensor D^{ep}_{ABCD} from **STEP 10**, or the elastic tensor E_{ABCD} from **STEP 1**, to a second order matrix **D** for evaluation of $\mathbf{B}^T\mathbf{D}\mathbf{B}$.

---

in order to evaluate the determinant of the deformation Jacobian J. Subroutine DEFJAC first evaluates the strain invariants EINV1, EINV2, and EINV3 which correspond to I, II, and III as presented by equations (3.51a-c) respectively. After this step is completed the deformation Jacobian DJAC is evaluated through equation (3.50). Subroutine DEFJAC also completes the sixth step of computations by evaluating the RR matrix which corresponds to R_{AB} as defined by equation (3.53).

Step 7 of the first stage of implementation is performed by subroutine IYIELD which is called by MISES1. Subsequent to this step subroutine FDER is called to evaluate the derivatives of the yield function with respect to $s_{AB}, A_{AB}, e_{AB}, J$ and κ using equations (3.61) through (3.65), respectively.

In Step 9, the denominator Q of equation (3.59) is evaluated through expression (3.55). This step is performed by subroutine ELPLD which also performs the computations required to evaluate the fourth order elasto-plastic stiffness tensor as described in Step 10 (Table 3.1).

The eleventh and the final step for evaluation of the elasto-plastic stiffness matrix $\mathbf{D}$, requires conversion of the fourth order tensor D^{ep}_{ABCD} or E_{ABCD} to a second order matrix $\mathbf{D}$ for evaluation of $\mathbf{B}^T\mathbf{D}\mathbf{B}$. This conversion is performed by subroutine CONVER which is also called by the control routine MISES1.

The evaluation of stress, plastic strain, and the shift stress tensors, and also the plastic work is performed at the second stage of implementation. Evaluation of stresses and plastic strains is in general more difficult than evaluation of the elasto-plastic stiffness tensor. This is primarily because evaluation of stresses requires some form of integration algorithm for the elasto-plastic constitutive relations. The role of an integration algorithm is to correct any possible drift from the yield surface and to ensure compliance with the consistency condition. For more information on various techniques for controlling drift from the yield sur-

Table 3.2. *Sequence of Computations Required for the Second Stage of Implementation (Evaluation of the Stresses, etc.).*

STEP 1. Retrieve $\mathbf{s}, \mathbf{e}, \mathbf{e}', \mathbf{A}$ and also κ obtained at the end of the last converged load increment from memory or disk. Also set FACTOR=1 and FACSUM=0.

STEP 2. Evaluate the strain increment vector, $\dot{\mathbf{e}}$, by subtracting the total strain vector at the end of the last converged increment from the current strain vector.

STEP 3. Convert all vectors to tensors.

STEP 4. Evaluate the fourth order elastic stiffness tensor E_{ABCD} as defined by equation (3.56).

STEP 5. Evaluate the Green's tensor C_{AB} through equation (3.40).

STEP 6. Evaluate the incremental elastic-predictor stress $\dot{s}^p_{AB}$ by assuming that the loading is elastic (i.e. $\dot{s}^p_{AB} = E_{ABCD}\dot{e}_{CD}$).

STEP 7. Evaluate the strain invariants using expressions (3.51a-c) and calculate the deformation Jacobian J through equation (3.50).

STEP 8. Evaluate the R_{AB} tensor using expression (3.53).

STEP 9. Evaluate the total elastic predictor stress s^p_{AB} by adding $\dot{s}^p_{AB}$ to the stress tensor obtained at the end of the last converged load increment or at the end of **STEP 21.** (i.e., $s^p_{AB} = s_{AB} + \dot{s}^p_{AB}$).

STEP 10. Evaluate the yield function using the elastic-predictor stress s^p_{AB}, through equations (3.38) and (3.39).

if $f \leq 0$ **then** (material is elastic)
- FACSUM=FACSUM+FACTOR;
- $s_{AB} = s^p_{AB}$;
- $e_{AB} = e_{AB} + \dot{e}_{AB}$;
- $C_{AB} = 2e_{AB} + \delta_{AB}$;
- $e'_{AB} = e'_{AB} + \dot{e}_{AB}$;
- $\dot{e}''_{AB} = 0$;
- IYIELD = 0;
- **go to STEP 22**;

else if $f > 0$ and FACTOR = 1 **then** (material is plastic use subincrementation)
- FACTOR = 0.01;
- $\dot{s}^p_{AB} = FACTOR \times \dot{s}^p_{AB}$;
- $\dot{e}_{AB} = FACTOR \times \dot{e}_{AB}$;
- **go to STEP 7**;

Table 3.2. *Continued.*

else if $f > 0$ and FACTOR < 1 **then**	(material is plastic
FACSUM = FACSUM + FACTOR;	subincrementation was
go to STEP 11;	performed previously)
end;	

STEP 11. Evaluate the yield function using s_{AB} through equations (3.38) and (3.39).

STEP 12. Evaluate the partial derivatives $\frac{\partial f}{\partial e_{AB}}, \frac{\partial f}{\partial J}, \frac{\partial f}{\partial s_{AB}}, \frac{\partial f}{\partial A_{AB}}$, and $\frac{\partial f}{\partial \kappa}$ using equations (3.61) through (3.65).

STEP 13. Evaluate Q using equation (3.55).

STEP 14. Evaluate $\dot{\Lambda}$ using equation (3.54).

STEP 15. Evaluate $\dot{\mu}$ using equation (3.47).

STEP 16. Evaluate the plastic component of the strain increment as follows: $\dot{\mathbf{e}}'' = \dot{\Lambda}\frac{\partial f}{\partial \mathbf{s}}$.

STEP 17. Determine the elastic component of the strain increment as follows: $\dot{\mathbf{e}}' = \dot{\mathbf{e}} - \dot{\mathbf{e}}''$.

STEP 18. Determine the stress increment using, $\dot{\mathbf{s}} = \mathbf{E}\dot{\mathbf{e}}'$.

STEP 19. Calculate the current shift stress tensor through, $\mathbf{A} = \mathbf{A} + (\mathbf{s} - \mathbf{A})\dot{\mu}$.

STEP 20. Calculate the current plastic work through, $\kappa = \kappa + (\mathbf{s} + \dot{\mathbf{s}})\dot{\mathbf{e}}''/J$.

STEP 21. Update all tensors as shown: $\mathbf{e} = \mathbf{e} + \dot{\mathbf{e}}$; $\mathbf{s} = \mathbf{s} + \dot{\mathbf{s}}$; $\mathbf{C} = 2\mathbf{e} + \delta$.

STEP 22. Check the subincrementation flag FACSUM;

if FACSUM < 1 **then**	(more strain subincrements
go to STEP 7;	are left to be processed)
else if FACSUM=1 **then**	(all strain subincrements
go to STEP 23;	are accounted for)
end;	

STEP 23. Write $\mathbf{s}, \mathbf{e}, \mathbf{e}', \mathbf{A}$, and κ to disk or memory for future use during the next iteration.

face the reader is referred to the papers by Ortiz and Popov [1985], Potts and Gens [1985], Schreyer, et. al. [1979], Simo and Taylor [1985] and Simo and Ortiz [1985]. The integration scheme used in this work incoporates the direct use of the consistency condition along with the subincrementation of the strain increment tensor. Prior to application of this method, an elastic-predictor stress is evaluated to determine whether the material is plastically loading. The sequence of computations that need to be performed for the second stage of implementation are listed in Table 3.2. The program flow for this stage is controlled by subroutine MISES at entry MISES2. Appendix C contains the program listing for this stage of implementation.

The first step in evaluation of the stresses involves retrieving the values of the total stress, total strain, shift stress tensor, and the plastic work κ that are evaluated at the end of the last converged load increment from disk or memory buffer. Notice that the first stage of implementation requires the values of the above quantities to be from the end of the previous iteration; whereas, the second stage requires these values to be from the last converged load increment. This is because through several experiments by the author it is determined that measuring the strain increment vector from some stable equilibrium condition results in faster convergence and better results. This may be explained by the fact that errors present during the stress calculations at the previous iterations will have no impact on the current iteration.

In step 2, the strain increment vector $\dot{e}_{AB}$ is evaluated by subtracting the strain vector at the end of the last converged increment from the current strain vector. The current strain vector is evaluated from the displacement field of the continuum. In Step 3, all quantities that are in vector form are converted to tensor form. The control subroutine MISES2 makes the appropriate calls to subroutine TENSOR in order to perform the above mentioned task.

In Step 4, the elastic stiffness matrix E_{ABCD} as defined by equation (3.56) is evaluated. As explained earlier this tensor is evaluated only once as long as the material properties do not change. The elastic stiffness tensor is evaluated by subroutine ADMAT which is called by the control program MISES2. The control program MISES2 also evaluates the Green's tensor C_{AB} using equation (3.40). This task is performed in the fifth step of the computations as is indicated by Table 3.2.

The sixth step of computations involves evaluation of a trial incremental elastic stress $\dot{s}^p_{AB}$ which is commonly referred to as the elastic predictor.

Step 7 is the start of the loop for subincrementation of the incremental strain vector when plastic loading is detected. In this step the strain invariants and the Jacobian of deformation J are evaluated through appropriate expressions as indicated in Table 3.2. Subroutine DEFJAC is called by the control program MISES2 to perform the above mentioned tasks. Subroutine DEFJAC also evaluates the R_{AB} tensor through expression (3.53).

The total elastic predictor stress s^p_{AB} is evaluated in ninth step. The incremental elastic predictor stress $\dot{s}^p_{AB}$ is added to the stress tensor obtained at the end of the last converged load increment or at the end of step 21, ($s^p_{AB} = s_{AB} + \dot{s}^p_{AB}$). Next, in step 10 the yield function is evaluated by using the elastic predictor stress s^p_{AB}. This task is performed by subroutine IYIELD which is called by the control program MISES2. If the value of the yield function is less than zero the material is in its elastic range and the final stress and strain quantities are determined as given by Table 3.2. If the yield function is greater than zero, then the yield flag IYIELD is set equal to one. The variable FACTOR is checked to determine if subincrementation has been performed previously. The value of variable FACTOR is set equal to one when the control program MISES2 is first accessed. At the current step if FACTOR is equal to one, then subincrementation is performed by dividing the strain increment $\dot{e}_{AB}$ into one hundred equal subincrements and the value of FACTOR is changed to 0.01. The incremental elastic predictor stress $\dot{s}^{p}_{AB}$ is also divided into one hundred subincrements. The program control is subsequently transferred to step 7. If the variable FACTOR has a value less than one, then subincrementation has been performed previously. The program keeps track of the number of subincrements processed by using the variable FACSUM as an accumulator (see Table 3.2). When the value of FACSUM reaches one, it indicates that all subincrements have been processed. When the value of FACTOR is less than one, the program control is transferred to step 11.

In step 11, the yield function is evaluated by using the stresses obtained at the end of the last converged increment or at the end of step 21. This task is again performed by subroutine YIELD. The yield function derivatives as given by equations (3.61) through (3.65) are evaluated in step 12. Subroutine FDER is called by the control program MISES2 to obtain these derivatives.

In steps 13 through 15 the quantities Q, $\dot{\Lambda}$, and $\dot{\mu}$ are evaluated by subroutine ELPLD using equations (3.55), (3.54), and (3.47) respectively. In step 16, the control program MISES2 evaluates the plastic component of the strain increment vector $\dot{e}''_{AB}$. The elastic component of the strain increment vector is then determined by $\dot{e}'_{AB} = \dot{e}_{AB} - \dot{e}''_{AB}$. The second Piola-Kirchhoff stress increment is calculated in step 18 as shown in Table 3.2.

In steps 19 through 21 all the hardening parameters, stresses and strains are updated to reflect the current values at the end of the current strain subincrement. In step 22, the value of FACSUM described earlier is checked. If FACSUM is equal to one then all subincrements are accounted for and the program control is transferred to the final step 23. Otherwise, if FACSUM is less than one it indicates that there are some strain increments which remain to be processed. Hence, program control is transferred to step 7.

Finally in step 23, the values of the stress, strain and shift stress tensors along with the elastic strain component are converted to vectors (6×1 matrices) from tensors and are stored. The value of the plastic work is also stored. These values correspond to the values at the end of iterations and are used by the first stage of implementation to evaluate the elasto-plastic stiffness tensor for the subsequent iteration.

3.5 A Simple Vectorization Technique

Vector processing is a technique for reducing the execution time required to run a program. It can be applied to FORTRAN code such as a DO-loop that performs the same sequence of operations on successive elements of arrays. Vector processing reduces the processor time required to execute such loops by using specialized hardware which overlaps or pipelines the processing for array elements. Vector processing however, requires that the compiler generate some additional instructions (i.e., vector load instructions). If the array lengths are too short, then the processor time lost due to handling of these instructions exceeds any advantages that are gained from vectorization. The smallest effective vector length depends on the hardware used. Usually this number ranges from nine to sixteen.

In the previous section it was pointed out that due to the short vector lenghts for tensors, vectorizing the operations involving tensors lead to higher execution time than the equivalent scalar operations. Here, in order to increase the vector lengths a simple technique is introduced which takes advantage of the way which FORTRAN stores arrays in the memory registers. Hence, vectorization may be achieved more effectively.

As an example let us consider the different ways in which the expression

$$s_{AB} = E_{ABCD} e_{CD} \tag{3.66}$$

may be evaluated using two distinctly different subroutines. The program EXAMPL listed below is the main control program which calls subroutines A and B to evaluate the stress tensor as expressed by equation (3.66). The final results which are returned by each subroutine to the main program are identical.

```
      PROGRAM EXAMPL
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION E(3,3,3,3), STRS(3,3), STRN(3,3)
C
C     Initialize E and STRN to the appropriate values
      .
      .
      .
      CALL A(E,STRS,STRN)
      CALL B(E,STRS,STRN)
      STOP
      END
```

Subroutine A evaluates expression (3.66) using four nested DO-loops each ranging from one to three. Notice that in this subroutine arrays E, STRS, and STRN are dimensioned identical to the calling routine.

```
      SUBROUTINE A(E,STRS,STRN)
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION E(3,3,3,3), STRS(3,3), STRN(3,3)
C
      DO 20 J = 1 , 3
      DO 20 I = 1 , 3
      DUMMY = 0.
      DO 10 L = 1 , 3
      DO 10 K = 1 , 3
10    DUMMY = DUMMY + E(I,J,K,L)*STRN(K,L)
20    STRS(I,J) = DUMMY
      RETURN
      END
```

Subroutine B evaluates expression (3.66) using two nested DO-loops each ranging from one to nine. This is acheived by dimensioning the arrays E, STRS, and STRN differently from the calling routine.

```
          SUBROUTINE B(E,STRS,STRN)
          IMPLICIT REAL*8 (A-H,O-Z)
          DIMENSION E(9,9), STRS(9), STRN(9)
C
          DO 20 K1 = 1 , 9
          DUMMY = 0.
          DO 10 K2 = 1 , 9
10        DUMMY = DUMMY + E(K1,K2)*STRN(K2)
20        STRS(K1) = DUMMY
          RETURN
          END
```

The FORTRAN code presented in subroutine B is far more efficient than the code listed in subroutine A even when scalar operations are used. This is because addressing of one and two dimensional arrays as used in subroutine B is computationally more efficient than addressing two and four dimensional arrays as used in subroutine A. Subroutine B may also be vectorized more effectively due to the use of longer vectors. Both subroutines A and B return identical results for array STRS to the calling program.

The simple technique shown above uses the fact that in FORTRAN arrays are stored by varying the left most indices through their full range first. This is referred to as the column major storage. This technique has been used frequently in implementing the material model introduced earlier, in the program *UNIFES*. The above discussion should give the reader a basic knowledge for understanding the relevant subroutines listed in Appendix C.

3.6 Uniaxial Verification of the material model

A uniaxial finite element test of the material model is performed by using a single eight-noded axisymmetric quadrilateral element with two inch sides. Numerical results obtained are then compared to the experimental observations made by Voyiadjis [1984]. The material used is aluminum alloy 2024-T4. This material is also used in the extrusion problems presented in Chapter 5. The material properties for this aluminum alloy are listed in Table 5.1.

Referring to Figure 3.3, the material is initially loaded until it elongates 0.1854 inches by using a displacement control scheme. This displacement corresponds to an engineering strain of 9.27% and a Lagrangian strain of 9.7%. At this stage the material is unloaded elastically and reloaded in the compressive direction until a displacement of 0.0852 inches is obtained. This displacement corresponds to an engineering strain of 4.26% and a Lagrangian strain of 4.35%. It must be noted that UNIFES accepts one loading case at a time, hence for unloading the material the RESTART capability of the program is utilized. Finally the material is unloaded again until a displacement of 0.0986 inches is obtained.

Figure 3.3 illustrates the second Piola Kirchhoff versus the Lagrangian strain for this aluminum alloy. As can be seen from this figure the results obtained from experiments and from the numerical model are sufficiently close to each other. The numerical model used here does not allow a variation in the plastic modulus of the material. This explains the deviation of the numerical results from the experimental observations when loading is reversed.

As was mentioned earlier, the RESTART capability of UNIFES is used to complete the analysis presented in Figure 3.3. Tables 3.3 through 3.5 list the UNIFES input files for each stage of the analysis. For detailes on the use of each input command the reader is referred to Chapter 6.

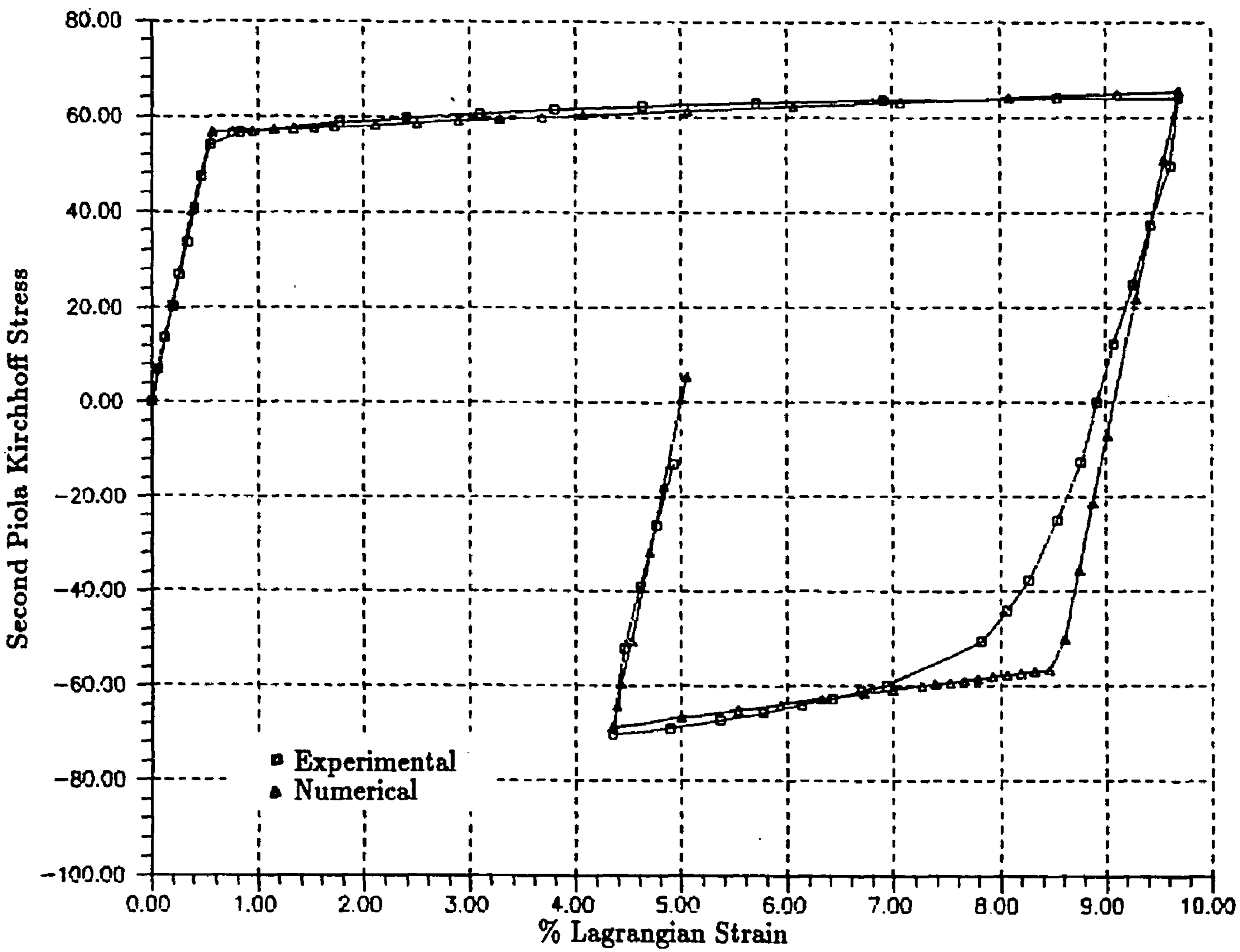


Figure 3.3 . *Experimental and Numerical Stress - Strain Diagrams for a Uniaxial Problem (Experimental Data After Voyiadjis [1984]).*

Table 3.3. *Input File for the Uniaxial Test (First Loading).*

```
TITLE 3
                    U N I A X I A L   T E S T
                              RUN 1
            Loading to a displacement of 0.1854 inches
NONLINEAR
NONSYMMETRIC
INCREMENTS 100 ITERATIONS 10 FACLOW 0.0001 FACHIGH 0.001
STOP_AFTER_DIVERGENCE 4
MATERIAL 1
          TYPE 2 E 10600. NU 0.3 KINE 40. ISOT 115.
          YIELD 32.9
ELEMENT 1 MATERIAL 1 NIPXI 3 NIPETA 3
NODES 8
          1      0.0      0.0      0
          2      0.0      1.0      0
          3      0.0      2.0      0
          4      1.0      0.0      0
          5      1.0      2.0      0
          6      2.0      0.0      0
          7      2.0      1.0      0
          8      2.0      2.0      0
INCIDENCES 1
          1 2308 8     1 6 8 3 4 7 5 2   0
DISPLACEMENTS
          NODE 1 TO 3 X 0.
          NODE 1 TO 4 BY 3 Y 0.
          NODE 6 Y 0.
          NODE 3 TO 5 BY 2 Y 0.1854
          NODE 8 Y 0.1854
END
OUTPUT_EVERY 1
```

Table 3.4. *Input File for the Uniaxial Test (Second Loading).*

```
TITLE  3
                    U N I A X I A L   T E S T
                              RUN 2
             Loading to a displacement of 0.0852 inches
RESTART
NONLINEAR
NONSYMMETRIC
INCREMENTS 100 ITERATIONS 10 FACLOW 0.0001 FACHIGH 0.001
STOP_AFTER_DIVERGENCE  4
MATERIAL 1
            TYPE  2  E  10600.  NU  0.3  KINE  40.  ISOT  115.
            YIELD  32.9
ELEMENT  1  MATERIAL  1  NIPXI  3  NIPETA  3
NODES  8
            1       0.0      0.0      0
            2       0.0      1.0      0
            3       0.0      2.0      0
            4       1.0      0.0      0
            5       1.0      2.0      0
            6       2.0      0.0      0
            7       2.0      1.0      0
            8       2.0      2.0      0
INCIDENCES 1
            1  2308  8     1  6  8  3  4  7  5  2    0
DISPLACEMENTS
            NODE 1 TO 3 X 0.
            NODE 1 TO 4 BY 3 Y 0.
            NODE 6 Y 0.
            NODE 3 TO 5 BY 2 Y -0.1002
            NODE 8 Y -0.1002
END
OUTPUT_EVERY  1
```

Table 3.5. *Input File for the Uniaxial Test (Third Loading).*

```
TITLE 3
                UNIAXIAL  TEST
                     RUN 3
           Loading to a displacement of 0.0986 inches
RESTART
NONLINEAR
NONSYMMETRIC
INCREMENTS 10 ITERATIONS 10 FACLOW 0.0001 FACHIGH 0.001
STOP_AFTER_DIVERGENCE 4
MATERIAL 1
          TYPE 2 E 10600. NU 0.3 KINE 40. ISOT 115.
          YIELD 32.9
ELEMENT 1 MATERIAL 1 NIPXI 3 NIPETA 3
NODES 8
          1     0.0    0.0    0
          2     0.0    1.0    0
          3     0.0    2.0    0
          4     1.0    0.0    0
          5     1.0    2.0    0
          6     2.0    0.0    0
          7     2.0    1.0    0
          8     2.0    2.0    0
INCIDENCES 1
          1 2308 8    1 6 8 3 4 7 5 2   0
DISPLACEMENTS
          NODE 1 TO 3 X 0.
          NODE 1 TO 4 BY 3 Y 0.
          NODE 6 Y 0.
          NODE 3 TO 5 BY 2 Y 0.0134
          NODE 8 Y 0.0134
END
OUTPUT_EVERY 1
```

4. SIMULATION OF CONTACT BOUNDARIES

4.1 Introduction

Contact problems in general may be classified into two groups, (1) contact between two deformable objects, and (2) contact between a deformable and a rigid object. Both of the above mentioned classes of contact problems have been extensively studied by a number of researchers. Papers by Hallquist, et. al. [1985], Lee and Kwak [1984], Padovan and Tovichakchaikul [1984], Wanxie and Suming [1988], Okamoto and Nakazawa [1979], and Voyiadjis and Poe [1986] extensively cover most aspects of contact between two deformable bodies. In the study by Ostachowicz [1984], rigid contact elements were used to model the rigid contacting bodies.

In this work we need to consider contact between a highly deformable material and a rigid forming press or extrusion die. Many successful formulations have been achieved by using simple geometric shapes such as straight lines and circular arcs to model the contact boundaries. However, these formulations can model a limited number of shapes and are often difficult to use. In many situations altering the shape of the forming press or the die requires additional programming by the design engineer or the analyst. In finite element solution of metal forming problems, it is necessary to accurately simulate the geometry of the contact boundaries in an effective and simple way. The design engineer should be able to model a variety of curved shaped boundaries without having to modify the program.

Generation of curves and surfaces has been the subject of extensive research by solid modelers and those involved in computer graphics. Many approaches for generation of curves have been proposed, among these are Bésier, Overhauser, Hermite, and β-spline formulations (a detailed discussion of these formulations may be found in the text book by Foley and Van Dam [1984]). The Bésier and β-spline formulations have gained tremendous popularity for their ease of use in interactive systems. These methods of curve and surface generation have also been applied to metal forming problems. Shiau and Kobayashi [1988], used the Bésier surfaces for analysis of three-dimensional open-die forging problems.

In this work the curved contact boundaries are modeled using Hermite parametric curves. A detailed explanation of this formulation is provided in the following sections which may also be applied to β-splines, Bésier and Overhauser parametric formulations with only slight modifications. Both tension free contact, and fixed rolling contact may be simulated.

4.2 Hermite Formulation of Cubic Curves

The Hermite form for a cubic parametric curve is determined from the end point coordinates and corresponding end point tangents. Referring to Figure 4.1, $\mathbf{P}$ and $\bar{\mathbf{P}}$ denote the position vectors of the starting and the finishing points of the curve, respectively. Similarly, $\mathbf{T}$ and $\bar{\mathbf{T}}$ denote the tangent vectors at the corresponding points $\mathbf{P}$ and $\bar{\mathbf{P}}$ respectively. We therefore have

$$\mathbf{P} = P_x\mathbf{i} + P_y\mathbf{j} \tag{4.1a}$$

$$\bar{\mathbf{P}} = \bar{P}_x\mathbf{i} + \bar{P}_y\mathbf{j} \tag{4.1b}$$

and

$$\mathbf{T} = T_x\mathbf{i} + T_y\mathbf{j} \tag{4.2a}$$

$$\bar{\mathbf{T}} = \bar{T}_x\mathbf{i} + \bar{T}_y\mathbf{j} \tag{4.2b}$$

where $\mathbf{i}$ and $\mathbf{j}$ are unit vectors along the coordinate axes x and y, respectively. Assuming a parametric expression for x, we have

$$x(t) = [M]\{G_x\} \tag{4.3a}$$

where

$$[M] = [t^3, t^2, t, 1] \tag{4.3b}$$

and

$$\{G_x\} = \begin{Bmatrix} a \\ b \\ c \\ d \end{Bmatrix} \tag{4.3c}$$

In equations (4.3a-b), t is a parameter such that $0 \leq t \leq 1$. The objective is to find the coefficient column vector $\{G_x\}$ of equation (4.3c) in terms of the end point position vectors and the corresponding tangents which are given by equations (4.1a-b) and (4.2a-b), respectively.

For $t = 0$, from equation (4.3a) we obtain P_x, while for $t = 1$, we obtain $\bar{P}_x$. This is expressed as follows:

$$x(0) = P_x = [0, 0, 0, 1]\{G_x\} \tag{4.4a}$$

$$x(1) = \bar{P}_x = [1, 1, 1, 1]\{G_x\} \tag{4.4b}$$

Similarly, the derivatives may be obtained by differentiating equation (4.3a) with respect to t. Consequently we have

$$x'(t) = [M]'\{G_x\} \tag{4.5}$$

where $x'(t)$ is derivative of $x(t)$ with respect to t and $[M]' = [3t^2, 2t, 1, 0]$. Therefore one obtaines the following:

$$x'(0) = T_x = [0, 0, 1, 0]\{G_x\} \tag{4.6a}$$

and

$$x'(1) = \bar{T}_x = [3, 2, 1, 0]\{G_x\} \tag{4.6b}$$

The four equations (4.4a-b) and (4.6a-b) can now be cast into a single matrix equation as follows:

$$\begin{Bmatrix} P_x \\ \bar{P}_x \\ T_x \\ \bar{T}_x \end{Bmatrix} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix} \{G_x\} \tag{4.7}$$

Figure 4.1 *Hermite Parametric Cubic Curve.*

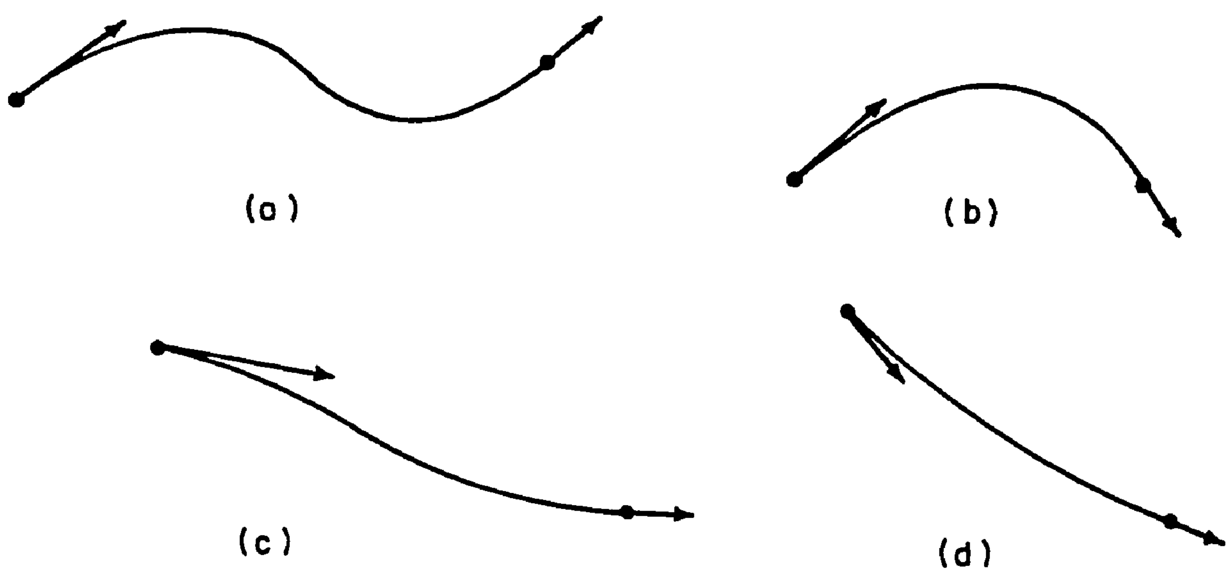


Figure 4.2 *Some Possible Shapes of the Hermite Curve.*

Inverting the 4×4 matrix and solving for the right hand side unknown coefficient vector $\{G_x\}$, we obtain

$$\{G_x\} = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{Bmatrix} P_x \\ \bar{P}_x \\ T_x \\ \bar{T}_x \end{Bmatrix} = [H]\{L_x\} \tag{4.8}$$

where $[H]$ is the 4×4 Hermite matrix and $[L_x]$ is the corresponding Hermite geometry vector which is specified by the user. Substituting equation (4.8) into equation (4.3a), we obtain

$$x(t) = [M]\{G_x\} = [M][H]\{L_x\} \tag{4.9}$$

where $[G_x]$ is equal to $[H]\{L_x\}$. A similar parametric expression for $y(t)$ may be obtained such that

$$y(t) = [M][H]\{L_y\} \tag{4.10}$$

where

$$\{L_y\} = \begin{Bmatrix} P_y \\ \bar{P}_y \\ T_y \\ \bar{T}_y \end{Bmatrix} \tag{4.11}$$

It must be noted that in using the Hermite formulation, the term $[G_x] = [H]\{L_x\}$ or $[G_y] = [H]\{L_y\}$ may be evaluated only once in the solution process. Any other point on the curve may be evaluated by simply assigning the appropriate value of the parameter t in the expressions (4.9) and (4.10). In using the Hermite parametric formulation, the user can control the shape of the curve by specifying the appropriate end point coordinates and the corresponding tangents at those points. Some of the possible curved shapes which may be obtained through this formulation are shown in Figure 4.2.

4.3 Motion of the Nodal Points on the Curved Boundaries

Unlike the constraint of motion of a nodal point along a straight boundary, simulation of motion along a curved boundary requires updating of the tangent vector to the curve at the nodal point and continuous correction for possible drift of the node away from the boundary. Figure 4.3, shows an incremental motion of the nodal point originally located on the curved boundary at A. Node A is constrained to move tangentially to the curve at A in the direction of vector $\mathbf{T}^{(1)}$ during the current load increment. However, this motion moves node A away from the curved boundary to a new location denoted by B. For the subsequent load increment, this node is projected back onto the boundary curve at point C and is constrained to move tangent to the curve in the direction of vector $\mathbf{T}^{(2)}$.

In order to locate point C for the drift correction, we need to find the appropriate parameter t corresponding to point C. Referring to Figure 4.3, since vector $\mathbf{R}$ (along BC) is perpendicular to the curve, its scalar dot product with $\mathbf{T}^{(2)}$ should equal to zero. We therefore obtain the following:

$$(x^C - x^B)T_x^{(2)} + (y^C - y^B)T_y^{(2)} = 0 \tag{4.12}$$

where (x^B, y^B) which represent the coordinates of point B are known values obtained from the equilibrium state during the last load increment. On the other hand, (x^C, y^C) and $(T_x^{(2)}, T_y^{(2)})$ are unknown and may be expressed in terms of the unknown parameter t using equations (4.9), (4.10) and (4.5) as follows:

$$x^C = [M]\{G_x\} \tag{4.13a}$$

$$y^C = [M]\{G_y\} \tag{4.13b}$$

and

$$T_x^{(2)} = [M]'\{G_x\} \tag{4.14a}$$

$$T_y^{(2)} = [M]'\{G_y\} \tag{4.14b}$$

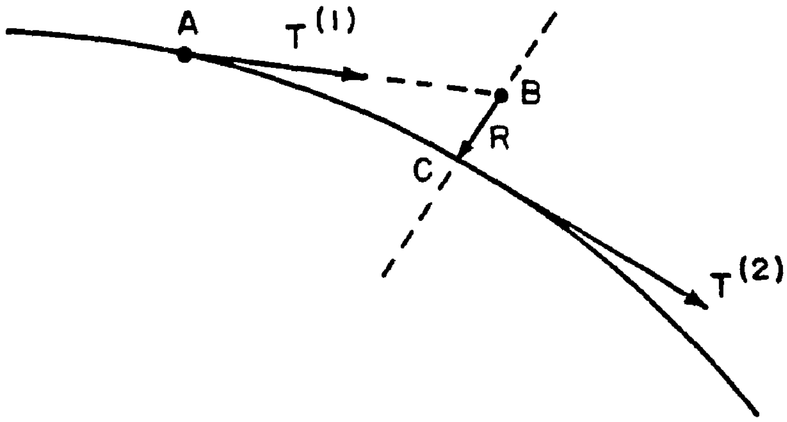


Figure 4.3 *Correction Process for the Motion of the Points Constrianed to Move on the Die.*

Substituting equations (4.13a-b) and (4.14a-b) into equation (4.12) will result in a fifth order polynomial equation in t as shown below:

$$c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0 = 0 \tag{4.15}$$

The constants $c_1, c_2, \ldots, c_5$, in equation (4.15) are given as follows:

$$c_0 = -G_x(3)G_x(4) - G_y(3)G_y(4) + x^B G_x(3) + y^B G_y(3) \tag{4.16a}$$

$$\begin{aligned} c_1 = &-(G_x(3))^2 - 2G_x(2)G_x(4) - (G_y(3))^2 \\ &- 2G_y(2)G_y(4) + 2(x^B G_x(2) + y^B G_y(2)) \end{aligned} \tag{4.16b}$$

$$\begin{aligned} c_2 = &-3G_x(1)G_x(4) - 3G_x(2)G_x(3) - 3G_y(1)G_y(4) \\ &- 3G_y(2)G_y(3) + 3(x^B G_x(1) + y^B G_y(1)) \end{aligned} \tag{4.16c}$$

$$c_3 = -2(G_x(2))^2 - 4G_x(1)G_x(3) - 2(G_y(2))^2 - 4G_y(1)G_y(3) \tag{4.16d}$$

$$c_4 = -5G_x(1)G_x(2) - 5G_y(1)G_y(2) \tag{4.16e}$$

$$c_5 = -3(G_x(1))^2 - 3(G_y(1))^2 \tag{4.16f}$$

where $G_x(n)$ implies the n-th row of the $\{G_x\}$ column vector. In equations (4.16), constants c_3, c_4 and c_5 are independent of the location of point B and their values will be calculated only once during the course of the analysis provided that the curved boundaries do not move. Similarly, the portions of the constants c_0, c_1 and c_2 which are independent of the x^B and y^B will be computed only once, hence resulting in good computational efficiency.

Once the constants c_0 through c_5 are evaluated, equation (4.15) is solved numerically for the parameter t. For the solution of the fifth order polynomial equation in t, the Muller's method is chosen (Gerald and Wheatley [1985]). This method converges to the appropriate root of the polynomial within the interval $0 \leq t \leq 1$ effectively when the first root is obtained. However, there are situations when the first root obtained may be outside the appropriate range of t, $(0 \leq t \leq 1)$.

When this situation arises, synthetic division is used to eliminate the first root and the program proceeds to find the next root. This process is continued until the appropriate root t within the required interval is obtained. In this work, it is assumed that the curved boundaries are reasonable enough and that point B is sufficiently close to the boundary so that only one normal from point B to the curve can be drawn. In other words, there is only one root of the equation (4.15) which is within the interval $0 \leq t \leq 1$. Once the appropriate value of the parameter t is determined, the location of point C and the tangent vector $\mathbf{T}^{(2)}$ are respectively evaluated through equations (4.13) and (4.14). Vector $\mathbf{R}$ in Figure 4.3 represents the correction required to bring back the nodal point to the curved boundary. For the subsequent load increment, the magnitude of vector $\mathbf{R}$ is applied to the restrained direction (direction normal to the boundary at C) as an imposed displacement and the nodal point is allowed to move along the tangent direction $\mathbf{T}^{(2)}$ freely. This process is repeated at the end of every load increment for all the nodes that are in contact with the curved boundaries.

4.4 Use of Multiple Curves in Generating Complex Boundaries

Quite frequently the curved boundaries involved in contact problems have complex shapes which may not be represented by a single cubic parametric curve. Also, there are situations where multiple boundaries need to be identified. In these situations it is necessary to use multiple splines to represent the appropriate shape of the boundaries. This task is achieved easily in *UNIFES* by simply allowing the user to define multiple geometry vectors in order to generate multiple Hermite curves.

One major problem in using multiple curve definitions for representing the boundaries is the task of identifying the closest curve to a given interface node.

The procedure described in the earlier section for controlling the motion of the nodal points on the boundaries is applicable, but additional information is needed to eliminate the unnecessary and time consuming solution of the fifth order polynomial of equation (4.15) for the curve segments which are not close to a given nodal point. Also, there are situations where there is a possiblility to draw a normal line from one nodal point to more than one curve segment. Figure 4.4, illustrates this situation. In Figure 4.4, node A should be constrained to move on curve C. However, the procedure described in the previous section can mistakenly identify curve C' as the constraining boundary for node A.

The above mentioned problems are eliminated by identifying an inclusion zone for each curve segment. If the coordinates of the node under consideration are within the bounds of the inclusion zone of a curve segment, then the node is constrained by the curve and the procedure of section 4.3 is applied to that particular curve segment.

The inclusion zone for each curve segment is identified by the user through specifying the lower-left and the uper-right hand corners of a box containing the spline. It is possible for the inclusion zone of two splines to overlap. If this situation arises, both curve segments are checked and the first curve with its normal passing through the nodal point is chosen as the constraining curve.

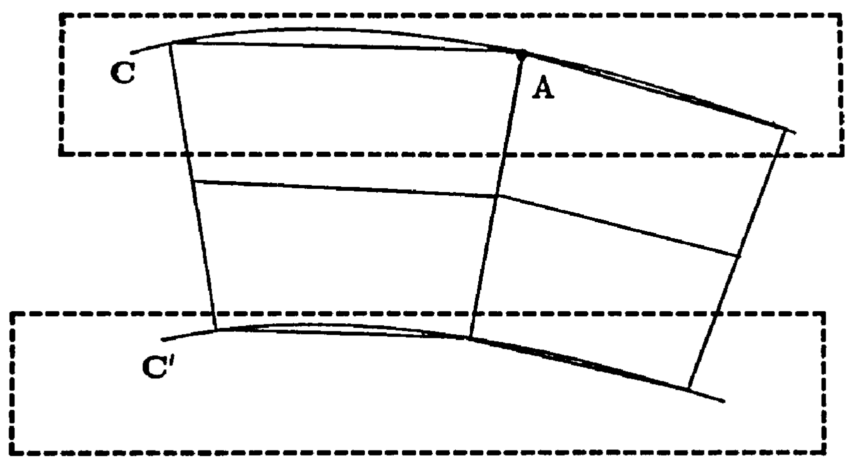


Figure 4.4 . *Zoning of the Contact Boundaries.*

4.5 Description of the Relevant Subroutines

A complete listing of the module necessary for controlling the motion of the nodal points on the curved boundaries is provided in Appendix D. The control program for this module is subroutine BOUND. This module is called by the finite element control program prior to start of each new load increment in order to update the boundary conditions and the orientation of the skew rollers. A brief description of each subroutine belonging to this module is provided next.

Subroutine BOUND. This subroutine is the main control module for determining the location of the interface node (nodes that may become in contact with the boundary during the course of the analysis). The appropriate direction cosine vectors (COSTX, COSTY), the incremental load vector (RINC), and the incremental displacement vector (UINC) are updated appropriately based on the motion and the location of the interface nodes. The parameter ICODE is set equal to zero if recalculation of the degree of freedom vector IDOF is not necessary, otherwise ICODE is set equal to unity.

Subroutine HERMITE. This subroutine is addressed only once during the course of the analysis in order to calculate the $\{G_x\}$ and $\{G_y\}$ column vectors as given by equation (4.8). This program is called by the main finite element module at the start of the program.

Subroutine COEF. This subroutine is addressed only once in order to calculate the coefficients c_3, c_4 and c_5 and the constant part of the coefficients c_0, c_1 and c_2 using equations (4.16a-f). This subroutine is called by the main finite element module at the start of the program.

Subroutine HERMXY. This sibroutine is used to evaluate the x and y coordinates of a point on the curved boundary using equations (4.9) and (4.10).

Subroutine XYPRIM. This subroutine calculates the tangent vector to the curve given by equation (4.5).

Subroutine MULLER. This subroutine evaluates the additional parts of constants c_0, c_1 and c_2 which depend on the location of the node at point B (refer to Figure 4.3). It subsequently checks the position of the nodal point against the inclusion zone for each curve in order to eliminate the curves which are not close to the point. Afterwards, the Muller's method with synthetic division is used to find the appropriate parameter t for the fifth order polynomial given by equation (4.15). If a root is not found within the interval $0 \leq \mathbf{t} \leq 1$, the flag IRET is set equal to zero. This indicates to program BOUND that the node is free.

5. PARAMETRIC STUDY OF AXISYMMETRIC EXTRUSION

5.1 Introduction

The problem of axisymmetric metal extrusion is investigated in this work. The billet is ten inches long with a radius of one inch and is made of the aluminum alloy 2024 T4. The material parameters used in the constitutive model (refer to Chapter 3) for this aluminum alloy are presented in Table 5.1. In Table 5.1, σ_y is the initial yield for uniaxial loading. The determination of parameters b and c is discussed by Voyiadjis [1984] and Voyiadjis and Kiousis [1987].

Table 5.1. *Material Properties for Aluminum 2024-T4.*

Modulus of Elasticity	$E = 10,600 ksi$
Poisson's Ratio	$\nu = 0.3$
Kinematic Hardening Parameter	$b = 40 ksi$
Isotropic Hardening Parameter	$c = 115 ksi$
Initial Yield Stress (From Uniaxial Loading)	$\sigma_y = 57 ksi$

The contact surface between the billet and the rigid die is assumed to be friction free. This in general is a valid assumption because lubricants are often used in extrusion processes. Referring to Figure 5.1, the die is modeled using three Hermite curves, namely AB, BC and CD. Portion AB represents a cylinder of constant radius that encloses the undeformed billet as shown in Figure 5.1. Region BC is the reduction region of the die and is obtained through a Hermite curve with horizontal tangents at both ends B and C. These horizontal tangents provide for a smooth transition of the material from one segment of the die to another. The exit portion of the die CD provides a region for the smooth recovering of the elastic strains and for unloading of the material. In the finite element formulation the billet is modeled such that it slides on the rigid die with tension free contact. This formulation allows the material to separate from the die as soon as tensile forces develop between the die and the billet. In the case when a previously released node penetrates the die, that node is again constrained to roll smoothly on the die surface.

Initially the billet is placed in region AB with a slight penetration of 0.05 inches into segment BC. This is to ensure that an axial force developes for the very first load increment. The billet is pushed through the die using a displacement control approach. A total displacement of 5 inches is applied to the left end of the billet using 250 load increments (0.02 inch per load increment) during the process of extrusion.

The convergence criterion used in this finite element analysis is based on the incremental internal energy obtained at the end of each iteration. The incremental internal energy defined during the ith iteration of the nth load increment is expressed as follows:

$$\Delta U_n^{(i)} = [Re_n^{(i)} - Re_n^{(i-1)}]\Delta u_n^{(i)} \tag{5.1}$$

where $Re_n^{(i)}$ and $Re_n^{(i-1)}$ are equilibrium load vectors at the nth load increment for the ith and $(i-1)$th iterations respectively. These equilibrium load vectors are obtained from the element stresses. In equation (5.1), $\Delta u_n^{(i)}$ represents the displacement increments at the ith iteration of the nth load increment. It is assumed that the convergence is obtained provided:

$$\Delta U_n^{(i)} \leq \epsilon \Delta U_n^{(1)} \tag{5.2}$$

where ϵ is a tolerance factor and $\Delta U_n^{(1)}$ is the internal energy obtained at the first iteration of the nth load increment. Similarly divergence is implied when

$$\Delta U_n^{(i)} > \Delta U_n^{(1)} \tag{5.3}$$

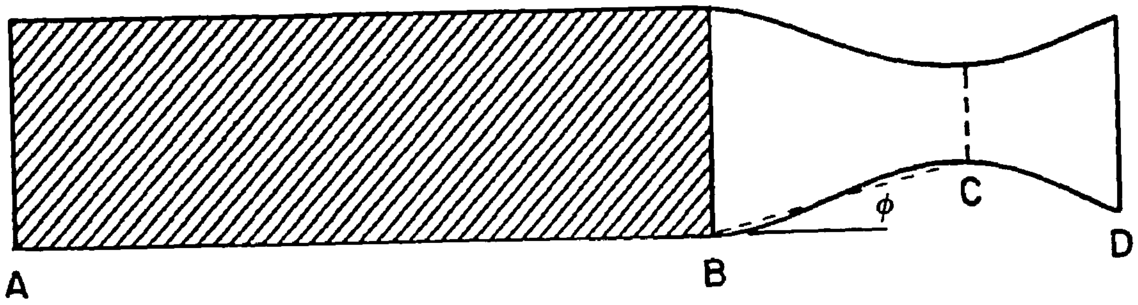


Figure 5.1. *Schematic Representation of the Die and the Initial Position of the Billet.*

During the course of the analysis, it is observed that for some load increments the proposed iterative solution procedure either diverges or does not converge to the prescribed limit within the prescribed maximum allowable number of iterations. When this occurs, the load step is reduced to one-half of the previous load increment and the procedure is repeated until all the load for this particular load increment is applied. This automatic procedure for the reduction of the load increment is only applied to those particular load increments with convergence problems; hence, it does not affect the loading procedure for any subsequent increments.

Table 5.2. *Parameters Used for the Control of the Iterative Solution Process.*

Strictest convergence tolerance	$\epsilon_{max} = 10^{-7}$
Least acceptable convergence tolerance	$\epsilon_{min} = 10^{-4}$
Allowable number of diverging iterations	$N_d = 4$
Maximum allowable number of iterations per load increment	$N = 10$

The parameters listed in Table 5.2, are utilized for the control of the iterative solution procedure used in this work. During each load increment, the tolerance ϵ_{max} is used in relation (5.2) in order to test for convergence. In case the criterion expressed by relation (5.2) using ϵ_{max} is not met, the corresponding value of ϵ required to satisfy the equality of relation (5.2) is computed. Should the computed value of ϵ fall within the bounds of ϵ_{max} and ϵ_{min}, then the solution is assumed to have converged and the next load increment is applied. Nevertheless, this relaxed convergence criterion is only accepted for a specified limited number of load increments. The parameter N_d indicates how many times relation (5.3) should be satisfied (divergence occurs) for a given load increment prior to applying the automatic subincrementation for the diverging load increment.

5.2 Comparison of Different Element Types and Meshes

A series of preliminary analyses are made using five different meshes each with different types of elements. The characteristics of the meshes used for each one of these runs are indicated in Table 5.3. The same die model is used for all five runs which are presented in this section. The initial one inch radius of the billet is reduced to 0.8 inches after extrusion. This is equivalent to a 36% reduction in the cross-sectional area of the billet. The die angle used is approximately 7.59 degrees. Referring to Figue 5.1, the die angle is approximated by connecting points B and C by a straight line and measuring the angle ϕ which this line makes with a horizontal line. The parameters used to generate each segment of the die for these runs are listed in Table 5.4. These runs are made for the purpose of determining the optimum mesh in terms of accuracy of the results, efficiency of analysis and handling of singularities which in particular arise at the exit of the die. Figure 5.2, illustrates the meshes used for each run.

Table 5.3. *Mesh Characteristics for Runs A1 Through A5*†.

Run	Number of Nodes in Element*	Number of Elements	Number of Integration Points per Element	Aspect Ratio of Elements	Total Number of Nodes	Total Number of Integration Points	Total Number of Interface Nodes
A1	4	320	4	2	369	1,280	40
A2	4	640	4	1	729	2,560	80
A3	8	320	9	2	1,057	2,880	80
A4	9	320	9	2	1,377	2,880	80
A5	4,5,8	640	4,9	1	970	2,960	160

† *All runs are based on a 36% reduction in the area and a die angle of 7.59 degrees.*
* *All elements are isoparametric quadrilateral elements.*

It is observed that run A1 using a mesh composed of 320 four-noded elements fails to complete the extrusion problem. Divergence occurs at load increment 84, that is when the second column of elements exit the die. Automatic load subincrementation procedure described earlier fails to correct the divergence. Run A2 using 640 four-noded elements, also fails to complete the analysis. However, this time divergence occurs at load increment 122, that is when the left end of the billet is displaced by 2.44 inches (6.2 cm). Further mesh refinement using the four-noded isoparametric element will be computationally inefficient when compared to meshes used utilizing eight or nine noded elements in runs A3, A4 and A5.

Table 5.4. *Parameters Used to Generate the Die for Runs A1 Through A5 Using the Hermite Formulation.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	2.0	0.8	11.45	0.0	2.0
CD	0.8	11.45	0.0	0.7	0.85	11.95	0.08	0.4

* *x-direction corresponds to the radial direction.* ** *y-direction corresponds to the axis of symmetry of the billet.*

Runs A3 and A4 are successfully completed with the material extruded as specified previously. However, use of 320 nine noded Lagrangian isoparametric elements in run A4 does not lead to improved performance or better results over the same number of eight-noded elements used for run A3. The extrusion pressure is obtained at the end of each load increment. The pressure versus the displacement of the left end of the billet is presented for runs A2, A3 and A4 in Figures 5.3 through 5.5. It is seen that the extrusion pressure increases steadily up to approximately a displacement of 1.6 inches. This range of steady increase of the extrusion pressure corresponds to the case when the billet fills the die. As the

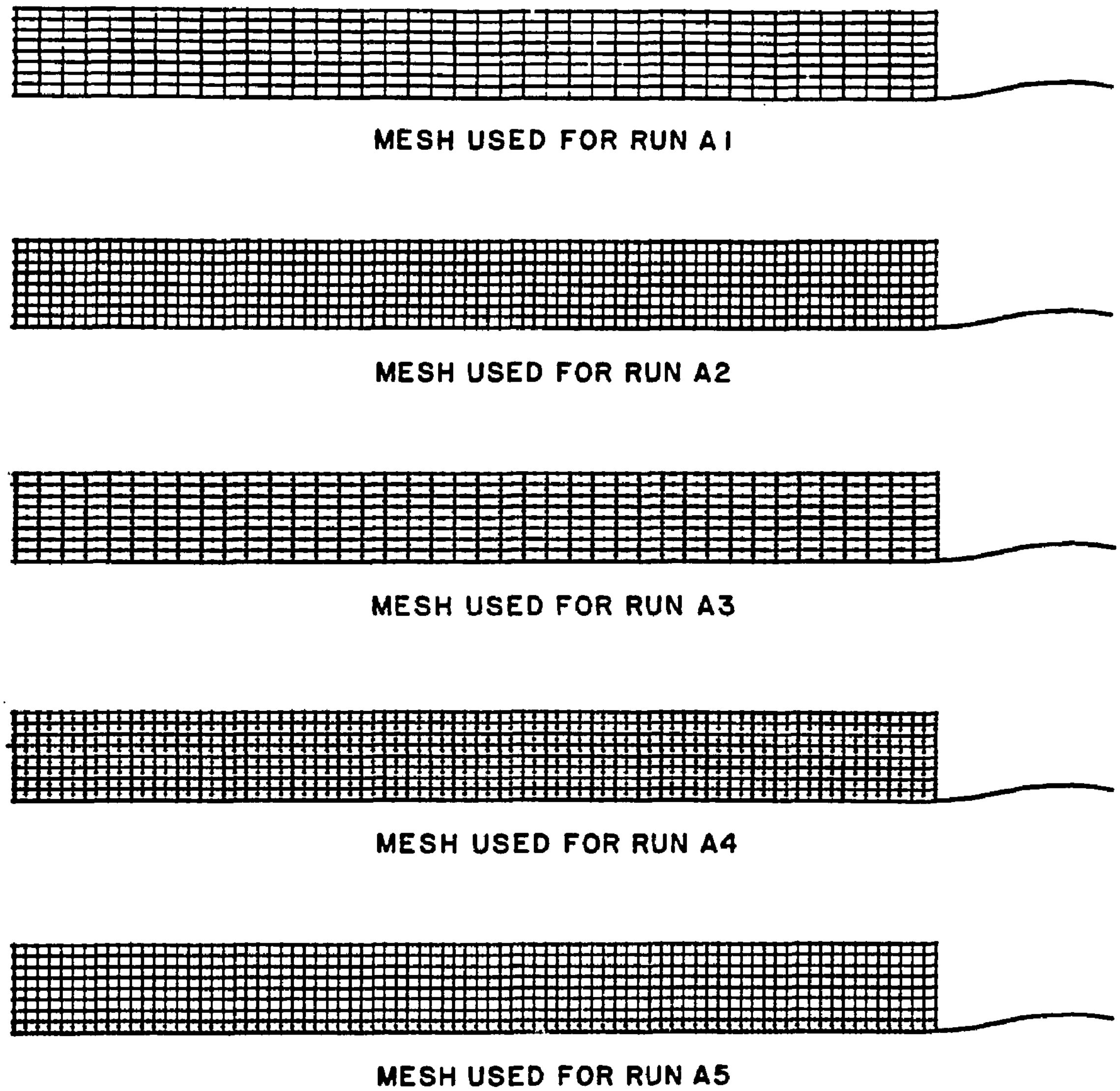


Figure 5.2. *Meshes Used for Runs A1 Through A5.*

billet exits the die, the interface nodes that are constrained to move on the die are released. This causes a drop in the elastic strain energy of the billet which in turn causes a drop in the extrusion pressure. As the material continues to enter the die, the next interface node due to be released continues to move on the die until it reaches the exit point of the die. During this period, the strain energy of the billet continues to increase hence increasing the extrusion pressure. This process will continue as shown in Figures 5.3 through 5.5 until the billet has completely exited the die. It must be noted that this condition does not exist in real extrusion problems. This is because as one material point on the surface of the billet exits the die, there is another material point infinitesimally close to the previous point which is still constrained to move on the die. In finite element analysis, the nodes that are released as the billet exits the die are a finite distance apart, consequently resulting in the above mentioned fluctuations in the extrusion pressure. Ideally, as one decreases the distance between the constrained nodes, these fluctuations are phased out. The studies performed by Aravas [1986] and Nagtegaal and Veldpaus [1980], confirm the existence of these fluctuations in the numerical solution of metal extrusion problems.

Run A5 is used to verify that these fluctuations are a function of the distance between the interface nodes. The mesh used in this run consists of 640 elements (eight elements in the radial direction and eighty elements along the length of the billet). The row containing 80 elements at the interface with the die is made of eight-noded quadratic isoparametric elements. The second row next to the interface is made of five-noded quadrilateral isoparametric transition elements. These elements are used to provide transition from a quadratic displacement field to a bi-linear displacement field. All subsequent rows are made of four-noded quadrilateral elements (refer to Figure 5.2). This mesh is computationally as efficient as the mesh for run A3 which uses 320 eight-noded quadrilateral isoparametric

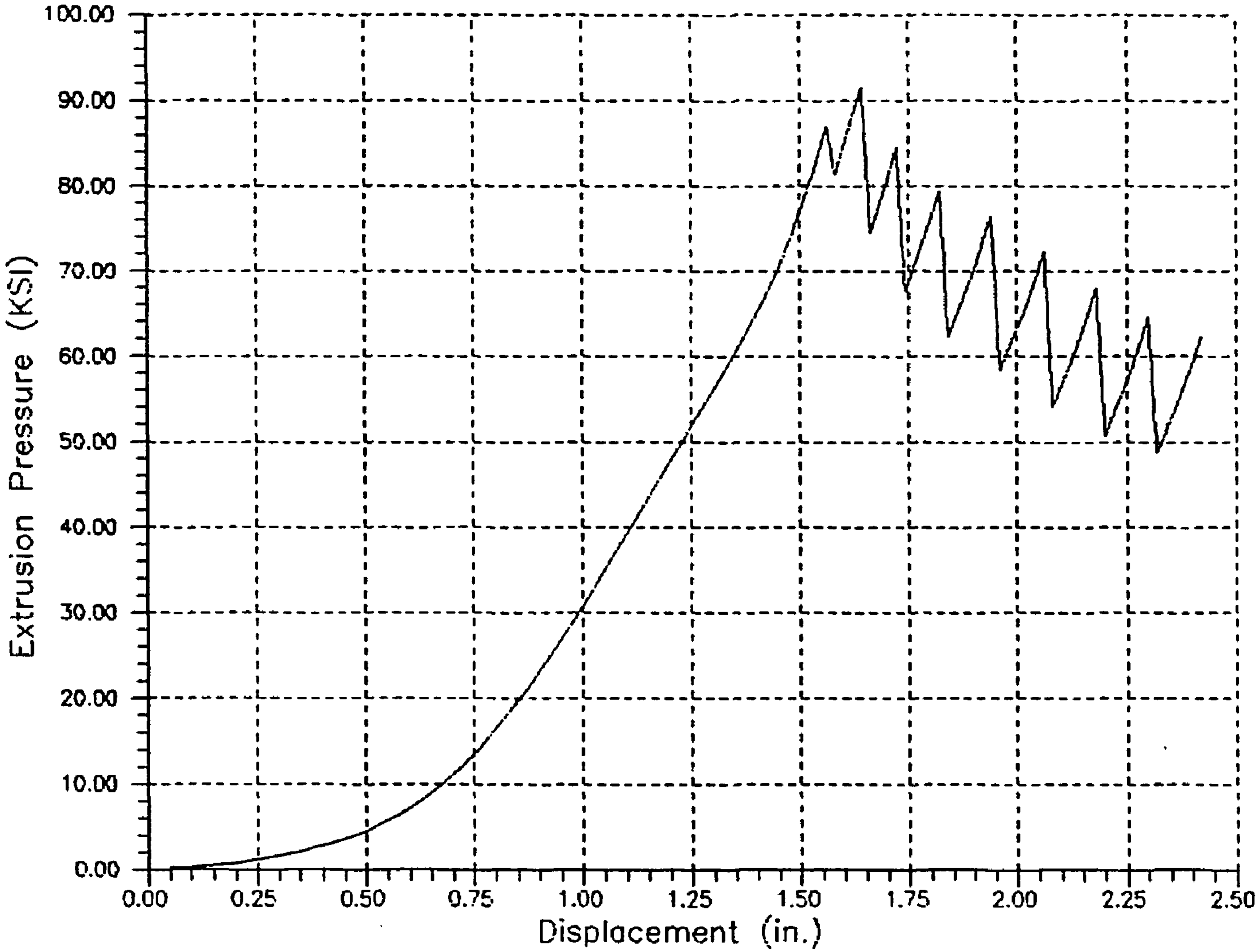


Figure 5.3. *Extrusion Pressure Versus Displacement for Run A2 (Incomplete Extrusion).*

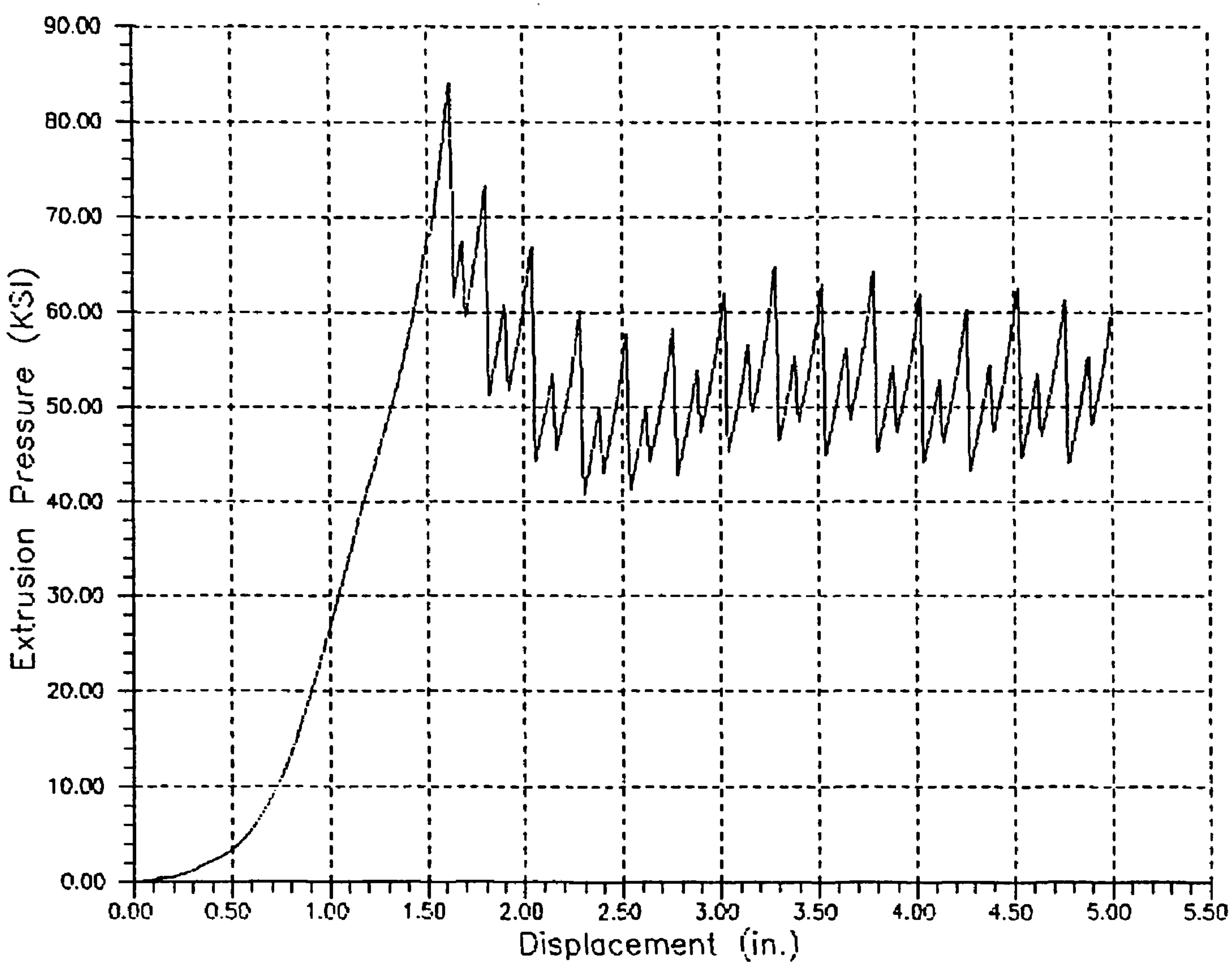


Figure 5.4. *Extrusion Pressure Versus Displacement for Run A3.*

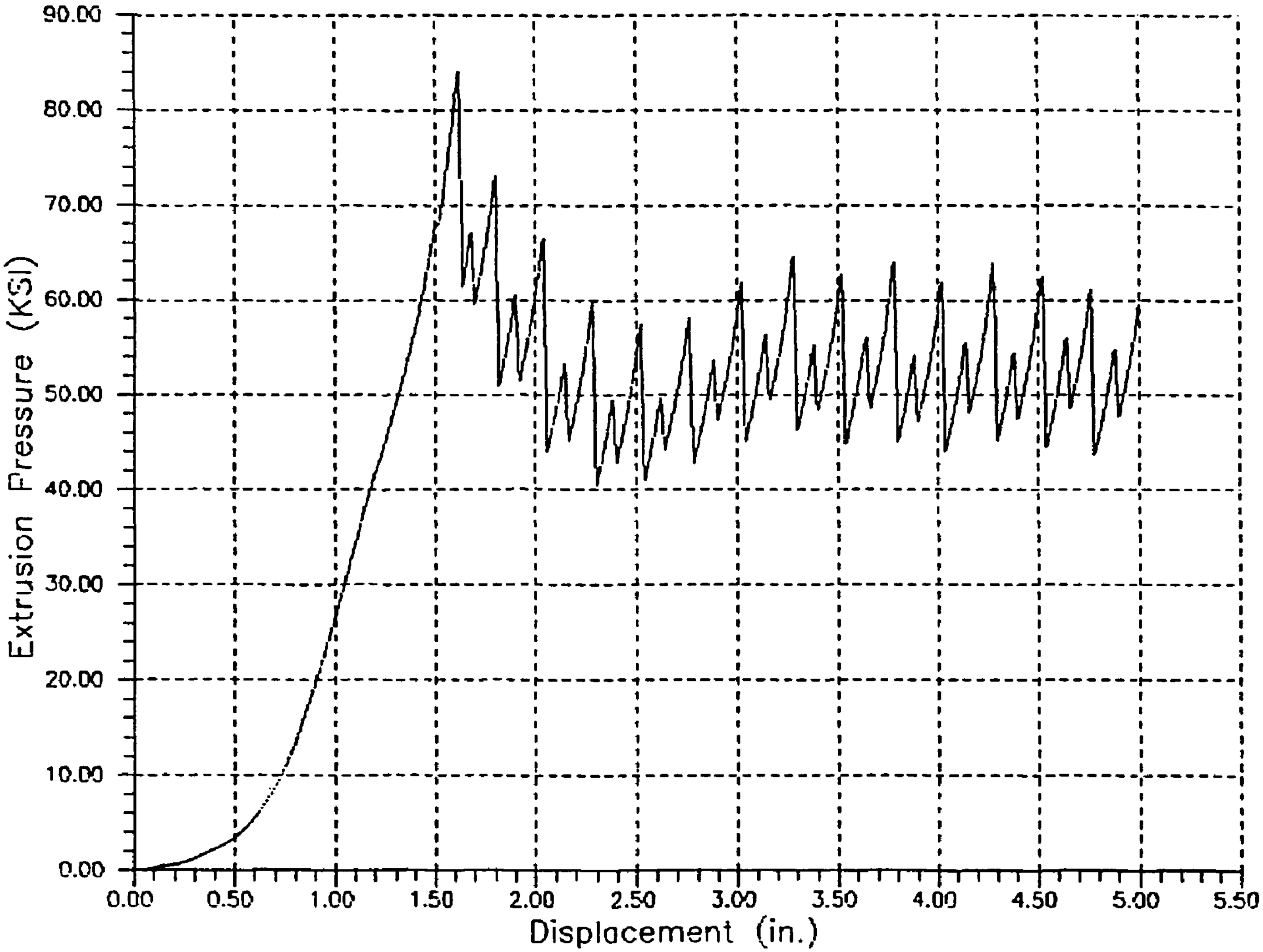


Figure 5.5. *Extrusion Pressure Versus Displacement for Run A4.*

elements. However, the distance between the interface nodes for run A5 is one-half of that used for run A3. Figure 5.6 shows the extrusion pressure versus the displacement for the left end of the billet for run A5. Comparison of this figure with Figures 5.3 through 5.5 indicates that the amplitude of the fluctuations decreases substantially when the mesh for run A5 is used.

It is also noted that the fluctuations shown in Figures 5.4 through 5.6 consist of a high peak followed immediately by a low peak. As shown in Figure 5.6, This corresponds to the release of the middle node followed by the release of the neighboring corner node of the quadratic elements respectively. Figure 5.3 which corresponds to run A2 does not show this characteristic because the elements used there at the interface with the die are four-noded isoparametric elements. Therefore, all the fluctuation peaks in Figure 5.3 are identical. We must also note that the distance between the respective peaks is identical to the distance between the neighboring nodes at the interface of the die.

The above mentioned fluctuations may be further reduced by incrementally removing the load on the released nodes as presented by Aravas [1986]. It must be noted however, that this method does not completely remove the fluctuations and will lead to an extrusion pressure which is significantly higher than the actual extrussion pressure. The higher extrusion pressure is caused by the fact that sustaining an artificial load on the exiting nodes will provide additional resistance to the flow of the material through the die.

Figures 5.3 through 5.6 also show that the extrusion pressure increases steadily until the material fills the die. Thereafter, the pressure decreases until a steady state condition is reached. This finding is identical for runs A1 through A5 regardless of element types.

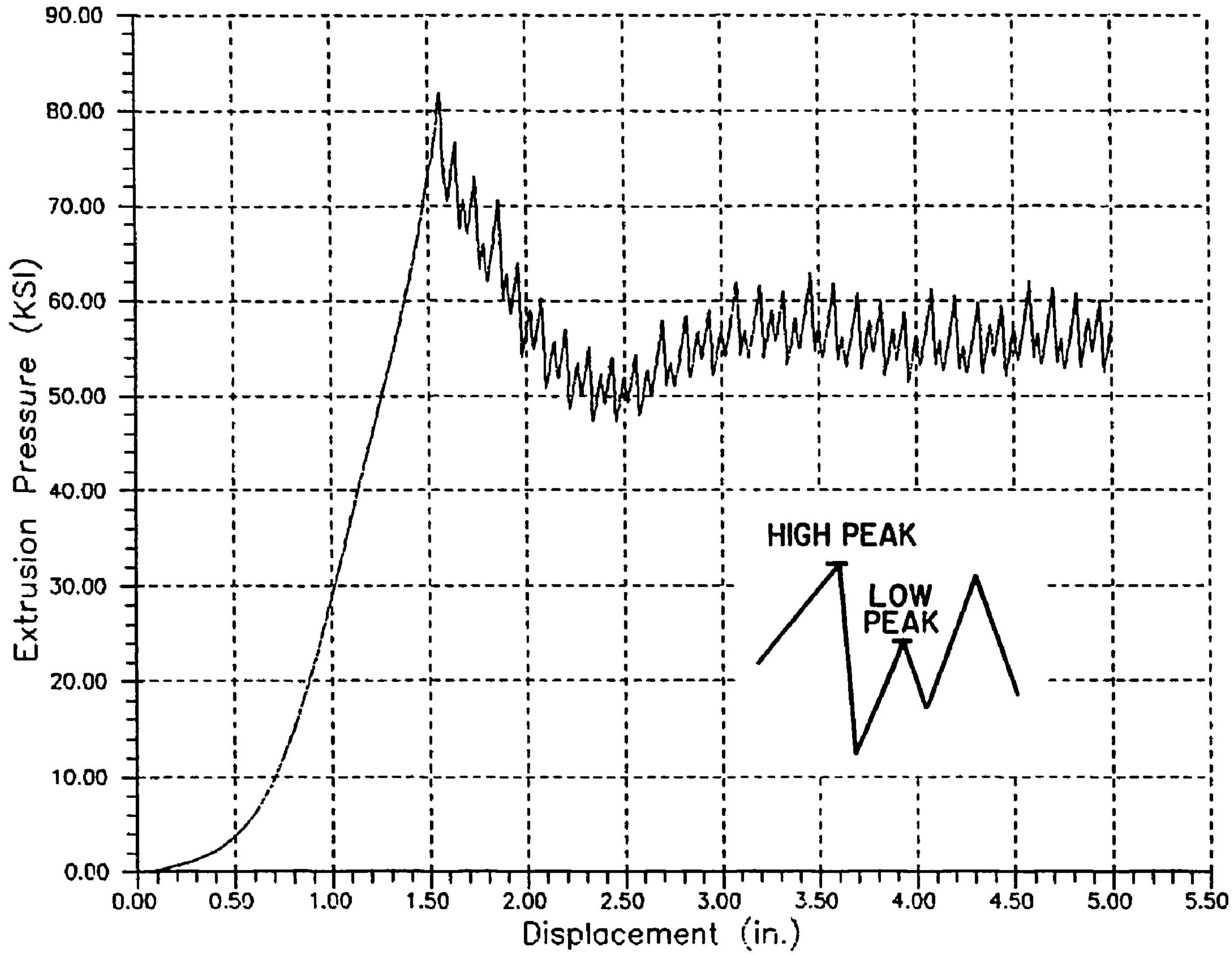


Figure 5.6. *Extrusion Pressure Versus Displacement for Run A5.*

5.3 Numerical Results Obtained Using Run A5

In Figure 5.7, the distribution of the yield zone at various stages of the extrusion is shown. It is observed that subsequent to exiting the die, yielding continues at some integration points. This is primarily due to the fact that the extruded material is seeking a final equilibrium state which may only be reached by additional yielding of the material until the residual stresses balance themselves.

In Figures 5.8 through 5.11, the stress intensity contours are plotted for both the Cauchy stress and the second Piola-Kirchhoff stress components. Two regions of heavy stress concentrations are observed in the stress contours. We note that at the entrance of the reduction region of the die (location B in Figure 5.1) and at the exit of the die, large stress variations are observed. This in particular is more intense for the axial and shear stress distributions. Variation of the axial stress distribution indicates that after extrusion, compressive axial stresses occur near the core of the billet and tension forms towards the outer radius of the billet. It is also seen that stresses at the free end of the extruded material vary significantly without showing distinct patterns.

In Figure 5.8, section A-A indicates an appropriate section for obtaining the residual stresses for the extruded material. The regions closer to the free end of the billet or the exit of the die are affected by the stress concentrations and should not be used to record the residual stresses for the final product.

Figure 5.12, shows the distribution of the volumetric Cauchy stress components. The highest compressive volumetric stresses occur in the immediate vicinity of segment BC of the die (reduction region). The maximum tensile volumetric stresses occur at the free end of the extruded billet and also at the axis of the billet after extrusion.

In Figures 5.13 and 5.14 the Lagrangian strain variations are shown. The highest shear strain value observed in Figure 5.14 is -31.96 percent which occurs

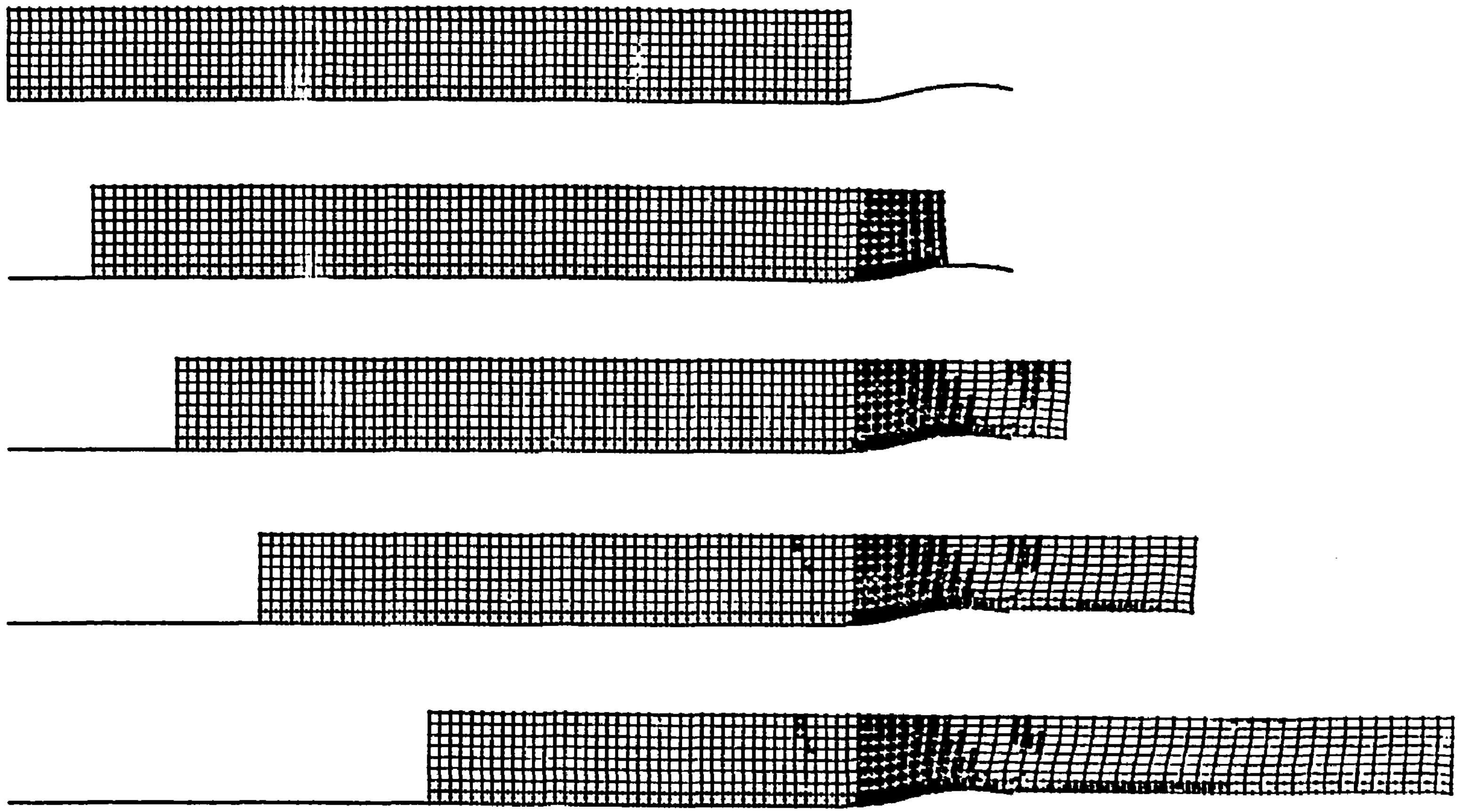
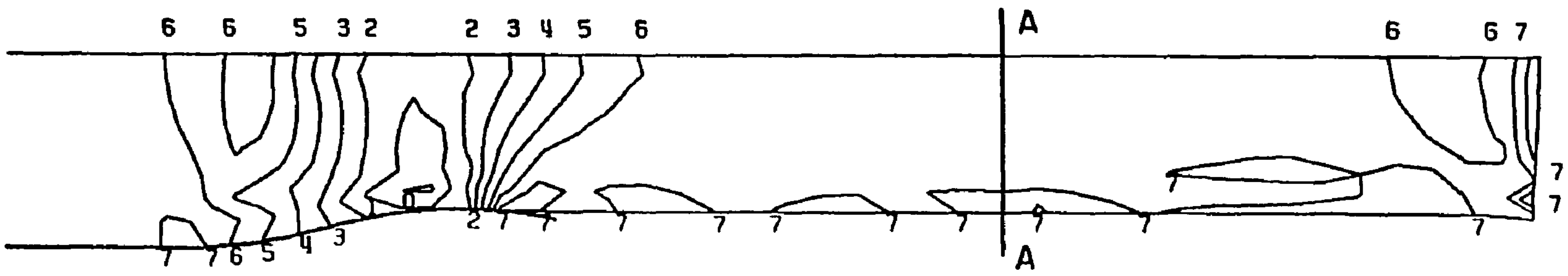
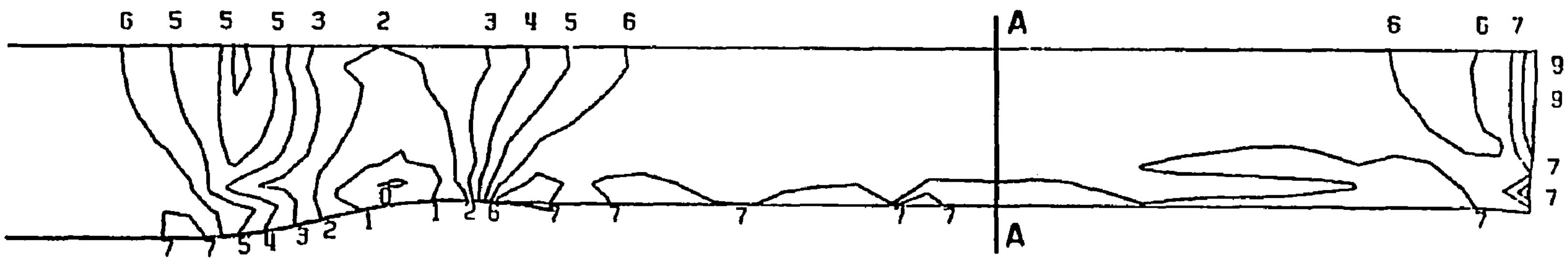


Figure 5.7. *The Yield Zone at Various Stages of the Extrusion for Run A5.*

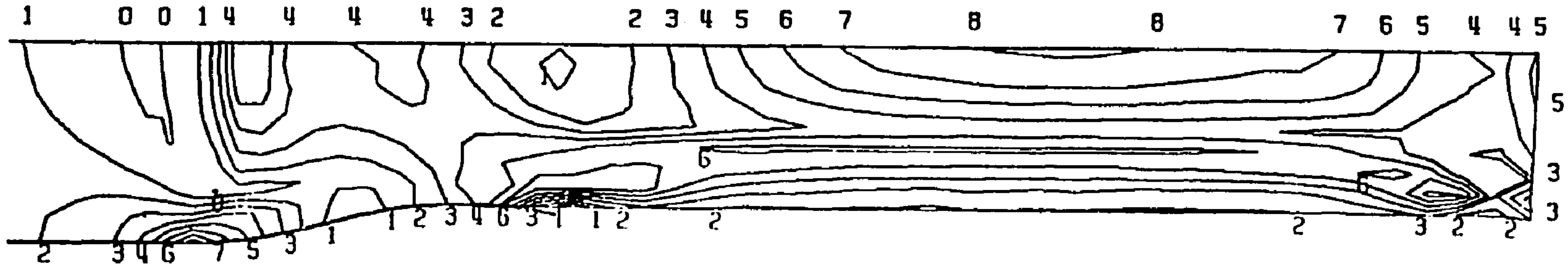


DISTRIBUTION OF THE 2ND P.K. RADIAL STRESS AT LOAD STEP 250 (KSI)
MINIMUM = -0.3660E+03   MAXIMUM = 0.9490E+02
0 = -0.3587E+03   1 = -0.3085E+03   2 = -0.2583E+03   3 = -0.2081E+03
4 = -0.1579E+03   5 = -0.1077E+03   6 = -0.5755E+02   7 = -0.7368E+01
8 = 0.4282E+02   9 = 0.9300E+02

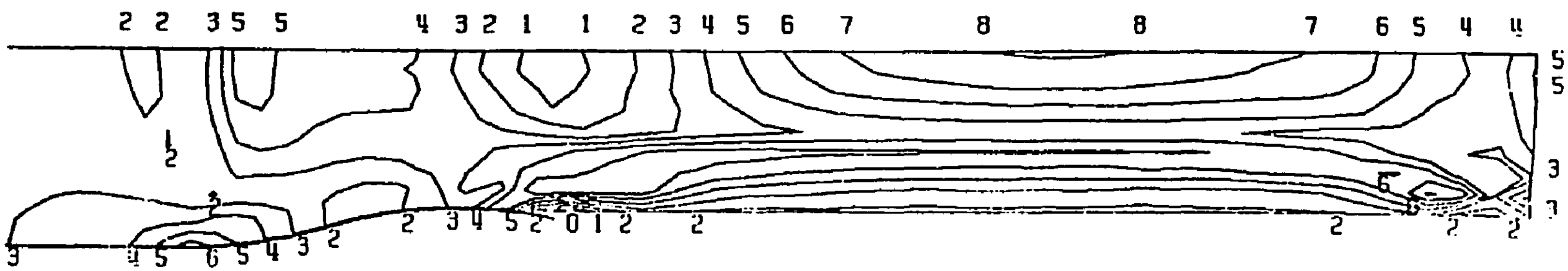


DISTRIBUTION OF THE RADIAL CAUCHY STRESS AT LOAD STEP 250 (KSI)
MINIMUM = -0.2167E+03   MAXIMUM = 0.5532E+02
0 = -0.2124E+03   1 = -0.1828E+03   2 = -0.1532E+03   3 = -0.1235E+03
4 = -0.9391E+02   5 = -0.6429E+02   6 = -0.3466E+02   7 = -0.5039E+01
8 = 0.2459E+02   9 = 0.5421E+02

Figure 5.8. *Radial Stress Distribution for Run A5.*

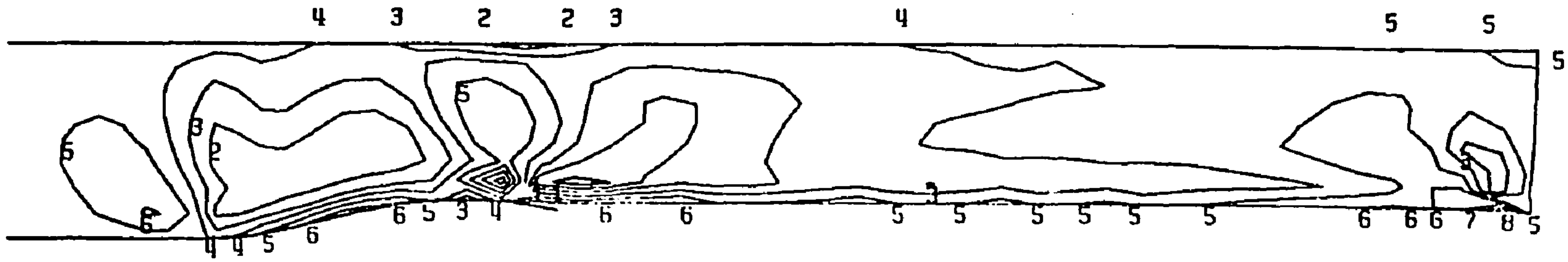


DISTRIBUTION OF THE 2ND P.K. AXIAL STRESS AT LOAD STEP 250 (KSI)
MINIMUM = -0.9018E+02 MAXIMUM = 0.8603E+02
0 = 0.8838E+02 1 = 0.6919E+02 2 = 0.5000E+02 3 = 0.3081E+02
4 = 0.1163E+02 5 = -0.7561E+01 6 = -0.2675E+02 7 = -0.4594E+02
8 = -0.6512E+02 9 = -0.8431E+02

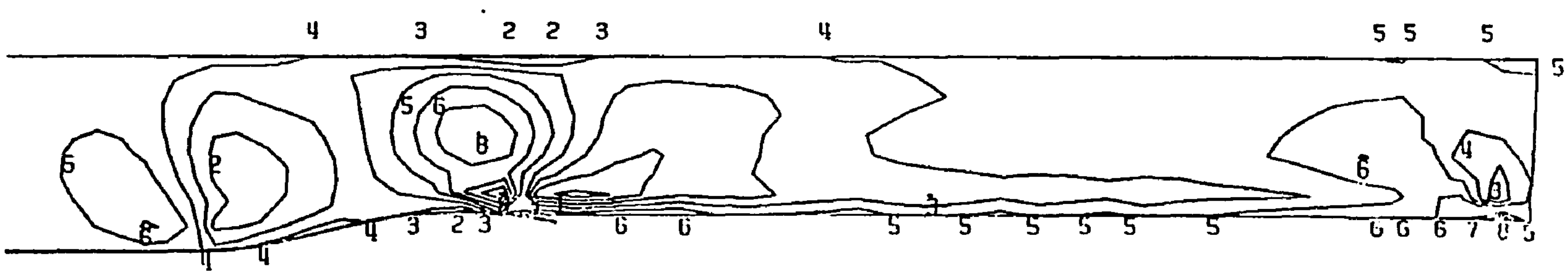


DISTRIBUTION OF THE CAUCHY AXIAL STRESS AT LOAD STEP 250 (KSI)
MINIMUM = -0.1615E+03 MAXIMUM = 0.1614E+03
0 = 0.1583E+03 1 = 0.1231E+03 2 = 0.8796E+02 3 = 0.5280E+02
4 = 0.1764E+02 5 = -0.1753E+02 6 = -0.5269E+02 7 = -0.8785E+02
8 = -0.1230E+03 9 = -0.1582E+03

Figure 5.9. *Axial Stress Distribution for Run A5.*

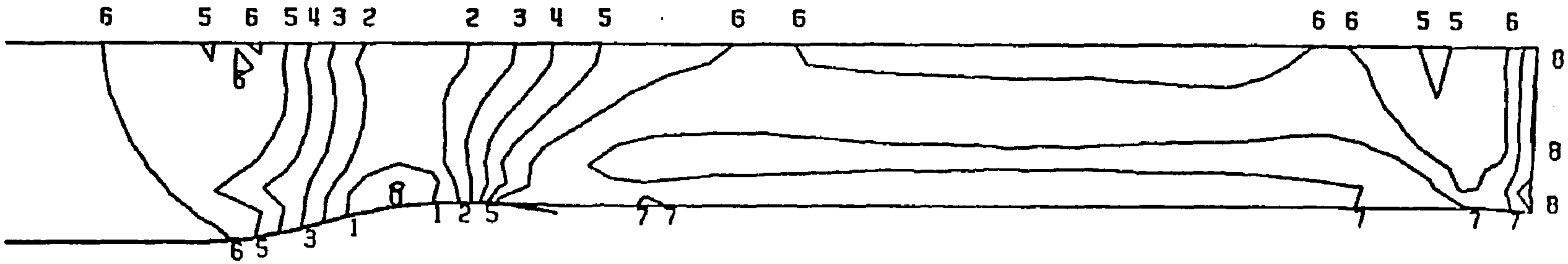


DISTRIBUTION OF THE 2ND P.K. SHEAR STRESS (RZ) AT LOAD STEP 250 (KSI)
MINIMUM = -0.5202E+02 MAXIMUM = 0.5801E+02
0 = -0.5098E+02 1 = -0.3900E+02 2 = -0.2702E+02 3 = -0.1504E+02
4 = -0.3056E+01 5 = 0.8925E+01 6 = 0.2091E+02 7 = 0.3289E+02
8 = 0.4487E+02 9 = 0.5685E+02



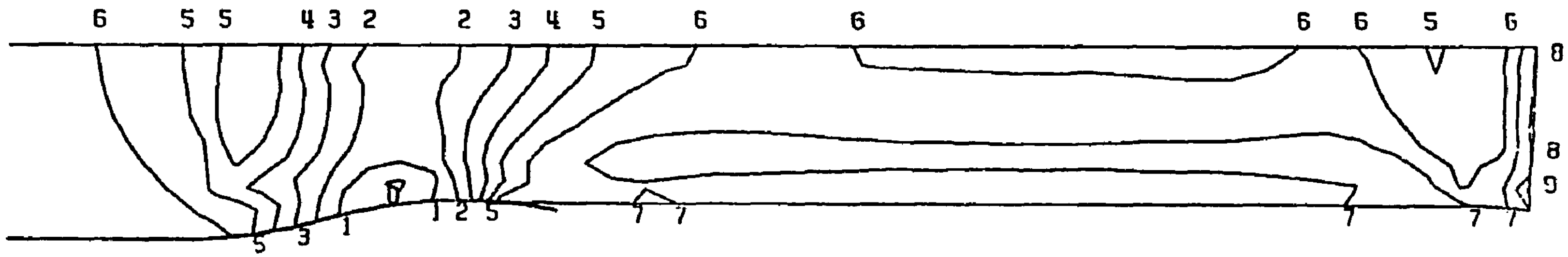
DISTRIBUTION OF THE SHEAR CAUCHY STRESS (RZ) AT LOAD STEP 250 (KSI)
MINIMUM = -0.5184E+02 MAXIMUM = 0.5804E+02
0 = -0.5081E+02 1 = -0.3884E+02 2 = -0.2688E+02 3 = -0.1491E+02
4 = -0.2945E+01 5 = 0.9020E+01 6 = 0.2098E+02 7 = 0.3295E+02
8 = 0.4492E+02 9 = 0.5688E+02

Figure 5.10. *Shear Stress Distribution for Run A5.*



DISTRIBUTION OF THE 2ND P.K. CIRCUMFERENTIAL STRESS AT LOAD STEP 250 (KSI)
MINIMUM = -0.3795E+03 MAXIMUM = 0.1458E+03

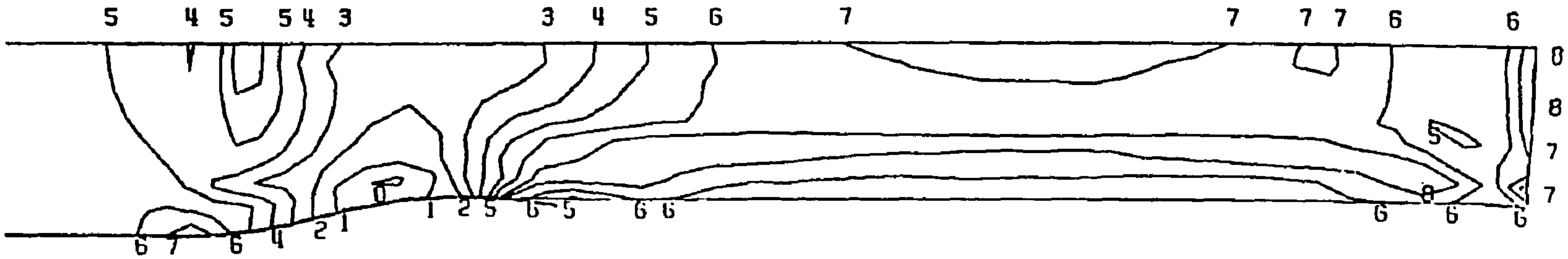
0 = -0.3719E+03	1 = -0.3147E+03	2 = -0.2575E+03	3 = -0.2003E+03
4 = -0.1431E+03	5 = -0.8589E+02	6 = -0.2869E+02	7 = 0.2850E+02
8 = 0.8570E+02	9 = 0.1429E+03		



DISTRIBUTION OF THE CIRCUMFERENTIAL CAUCHY STRESS AT LOAD STEP 250 (KSI)
MINIMUM = -0.3208E+03 MAXIMUM = 0.1192E+03

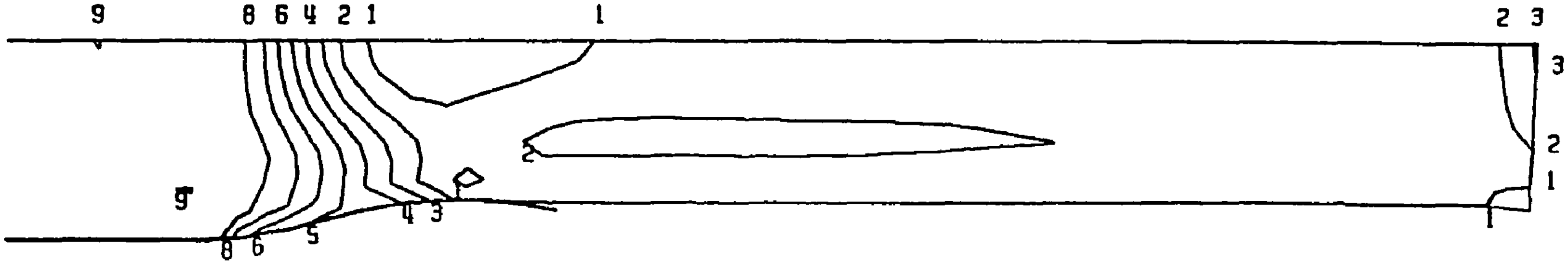
0 = -0.3144E+03	1 = -0.2665E+03	2 = -0.2186E+03	3 = -0.1706E+03
4 = -0.1227E+03	5 = -0.7481E+02	6 = -0.2690E+02	7 = 0.2102E+02
8 = 0.6894E+02	9 = 0.1169E+03		

Figure 5.11. *Circumferential Stress Distribution for Run A5.*



DISTRIBUTION OF THE VOLUMETRIC STRESSES AT LOAD STEP 250 (KSI)
MINIMUM = -0.2121E+03   MAXIMUM = 0.8023E+02
0 = -0.2078E+03   1 = -0.1760E+03   2 = -0.1442E+03   3 = -0.1123E+03
4 = -0.8051E+02   5 = -0.4869E+02   6 = -0.1686E+02   7 = 0.1497E+02
8 = 0.4679E+02   9 = 0.7862E+02

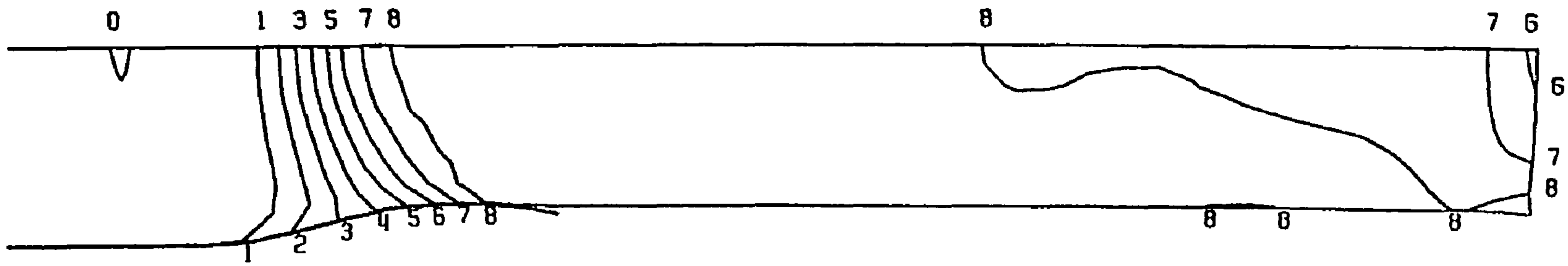
Figure 5.12. *Cauchy Volumetric Stress Distribution for Run A5.*



DISTRIBUTION OF THE RADIAL STRAIN AT LOAD STEP 250

MINIMUM = -0.1971E+00   MAXIMUM = 0.8883E-03

0 = -0.1931E+00	1 = -0.1716E+00	2 = -0.1500E+00	3 = -0.1285E+00
4 = -0.1069E+00	5 = -0.8535E-01	6 = -0.6380E-01	7 = -0.4224E-01
8 = -0.2069E-01	9 = 0.8707E-03		

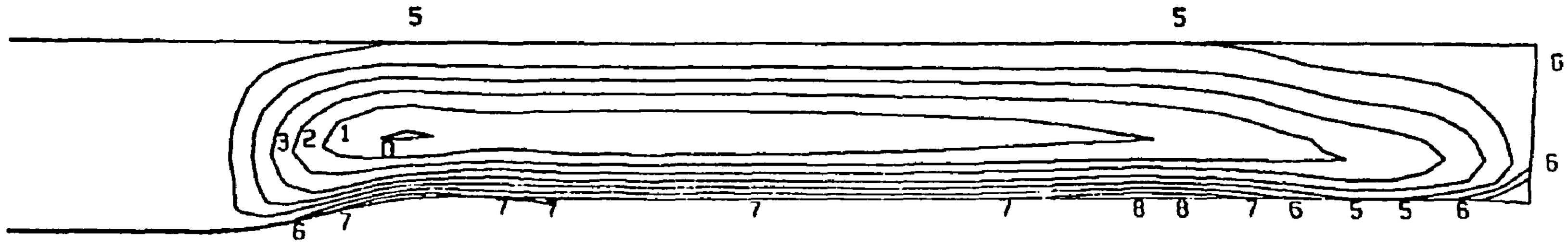


DISTRIBUTION OF THE AXIAL STRAIN AT LOAD STEP 250

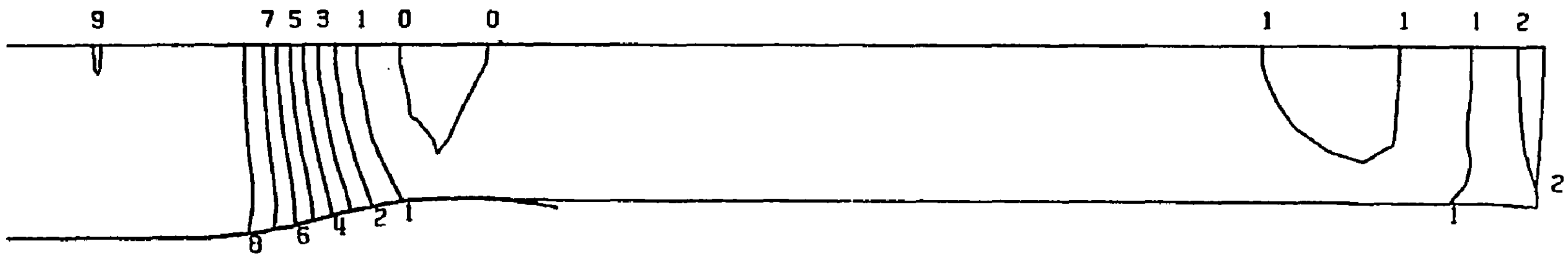
MINIMUM = -0.7304E-02   MAXIMUM = 0.7516E+00

0 = -0.7157E-02	1 = 0.7548E-01	2 = 0.1581E+00	3 = 0.2408E+00
4 = 0.3234E+00	5 = 0.4060E+00	6 = 0.4887E+00	7 = 0.5713E+00
8 = 0.6540E+00	9 = 0.7366E+00		

Figure 5.13. *Radial and Axial Strain Distributions for Run A5.*

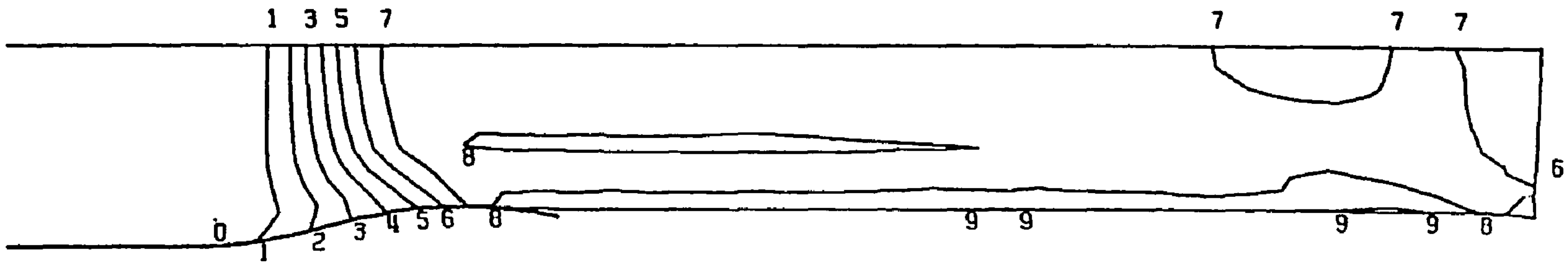


DISTRIBUTION OF THE SHEAR STRAIN (RZ) AT LOAD STEP 250
MINIMUM = -0.3196E+00   MAXIMUM = 0.1758E+00
0 = -0.3132E+00   1 = -0.2592E+00   2 = -0.2053E+00   3 = -0.1514E+00
4 = -0.9742E-01   5 = -0.4348E-01   6 = 0.1047E-01   7 = 0.6441E-01
8 = 0.1184E+00   9 = 0.1723E+00



DISTRIBUTION OF THE CIRCUMFERENTIAL STRAIN AT LOAD STEP 250
MINIMUM = -0.1883E+00   MAXIMUM = 0.8875E-03
0 = -0.1845E+00   1 = -0.1639E+00   2 = -0.1433E+00   3 = -0.1227E+00
4 = -0.1021E+00   5 = -0.8152E-01   6 = -0.6092E-01   7 = -0.4033E-01
8 = -0.1973E-01   9 = 0.8699E-03

Figure 5.14. *Shear and Circumferential strain Distributions for Run A5.*



DISTRIBUTION OF THE PLASTIC-WORK INTENSITY AT LOAD STEP 250 (KSI)
MINIMUM = 0.0 MAXIMUM = 0.6091E+02

0 = 0.7157E-01	1 = 0.6569E+01	2 = 0.1321E+02	3 = 0.1985E+02
4 = 0.2649E+02	5 = 0.3313E+02	6 = 0.3977E+02	7 = 0.4641E+02
8 = 0.5305E+02	9 = 0.5969E+02		

Figure 5.15. *Distribution of the Plastic Work Intensity for Run A5.*

at the center of the reduction region. It is observed in Figure 5.13 that the axial strain for the extruded billet has almost a constant value of 65.4 percent. However, the maximum axial strain is 75.16 percent at the free end of the extruded billet.

The distribution of the plastic work intensity κ is depicted in Figure 5.15. As anticipated, the maximum plastic work occurs at the outer radius of the extruded billet which undergoes higher shear strains than regions closer to the core of the billet. The value of κ as used in the expression for the yield surface determines the extend of the isotropic hardeneing of the material.

5.4 Study of Various Area Reduction and Die Angle Changes

In order to study the effect of changes in the percent reduction of the cross-sectional area and also the die angle on the extrusion pressure a series of nine analyses are performed. The characteristics of each analysis is shown in Table 5.5. As is seen in this table, these runs are made for 25, 30, and 35 percent reductions in the area and also for die angles of 5, 7, and 9 degrees. These numbers are selected because they represent logical values which are used in most extrusion applications. Die angles which are greater than ten degrees have been known to cause large tensile volumetric stresses in the billet which lead to the development of voids in the material. The same reasoning also applies to large reduction ratios in the cross-sectional area of the billet. Tables 5.6 through 5.14 show the parameters used to generate the die for each case.

Figures 5.16 through 5.18 show the extrusion pressure versus the displacement of the left end of the billet for analysis groups B, C and D respectively. As is seen in these figures the extrusion pressure increases as the die angle increases. This is logical because a larger angle tends to reduce the cross-sectional area of the billet over a shorter distance, therefore requiring a larger driving force. Figures

Table 5.5. *Identification Codes for Each Analysis*

	% Area Reduction		
Die Angle	25	30	35
9 Degrees	B1	C1	D1
7 Degrees	B2	C2	D2
5 Degrees	B3	C3	D3

5.16 through 5.18 also indicate that as the die angle increases the amplitude of the fluctuations in the extrusion pressure increases as well. This confirms the observations made in the preceeding section that the amplitude of these fluctuations is directly related to the elastic strain energy of the material. The larger extrusion pressure for sharper die angles induces higher levels of elastic strain energy in the billet; hence, the amplitude of the fluctuations is increased.

Figure 5.19 illustrates the change in the steady-state extrusion pressure for various reduction ratios as the die angle increases. It is observed that the variation in the steady-state pressure with respect to the die angle is linear. However, the slope of the curves in Figure 5.19 increases slightly as the reduction ratio is increased. Figure 5.20 shows the relation between the peak extrusion pressure and the die angle. In this case the curves are not perfect straight lines.

Figures 5.21 and 5.22 illustrate the steady-state and the peak extrusion pressures versus the percent reduction in area for various die angles, respectively. It is again observed that the curves in these figures represent relations which are approximately linear. It is also observed that as the die angle increases the slope of the curves increases slightly.

Table 5.6. *Parameters used to generate the die for run B1.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	0.5	0.866	10.796	0.0	0.5
CD	0.866	10.796	0.0	0.7	0.916	11.296	0.08	0.4

Table 5.7. *Parameters used to generate the die for run B2.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	0.7	0.866	11.041	0.0	0.7
CD	0.866	11.041	0.0	0.7	0.916	11.541	0.08	0.4

Table 5.8. *Parameters used to generate the die for run B3.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	1.0	0.866	11.482	0.0	1.0
CD	0.866	11.482	0.0	0.7	0.916	11.982	0.08	0.4

Table 5.9. *Parameters used to generate the die for run C1.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	0.7	0.8367	10.981	0.0	0.7
CD	0.8367	10.981	0.0	0.7	0.8867	11.481	0.08	0.4

Table 5.10. *Parameters used to generate the die for run C2.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	1.0	0.8367	11.28	0.0	1.0
CD	0.8367	11.28	0.0	0.7	0.8867	11.78	0.08	0.4

Table 5.11. *Parameters used to generate the die for run C3.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	1.5	0.8367	11.817	0.0	1.5
CD	0.8367	11.817	0.0	0.7	0.8867	12.317	0.08	0.4

Table 5.12. *Parameters used to generate the die for run D1.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	0.8	0.806	11.175	0.0	0.8
CD	0.806	11.175	0.0	0.7	0.856	11.675	0.08	0.4

Table 5.13. *Parameters used to generate the die for run D2.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	1.0	0.806	11.53	0.0	1.0
CD	0.806	11.53	0.0	0.7	0.856	12.03	0.08	0.4

Table 5.14. *Parameters used to generate the die for run D3.*

segment	P_x^*	P_y^{**}	T_x^*	T_y^{**}	$\bar{P}_x^*$	$\bar{P}_y^{**}$	$\bar{T}_x^*$	$\bar{T}_y^{**}$
AB	1.0	0.0	0.0	2.0	1.0	9.95	0.0	2.0
BC	1.0	9.95	0.0	1.5	0.806	12.167	0.0	1.5
CD	0.806	12.167	0.0	0.7	0.856	12.667	0.08	0.4

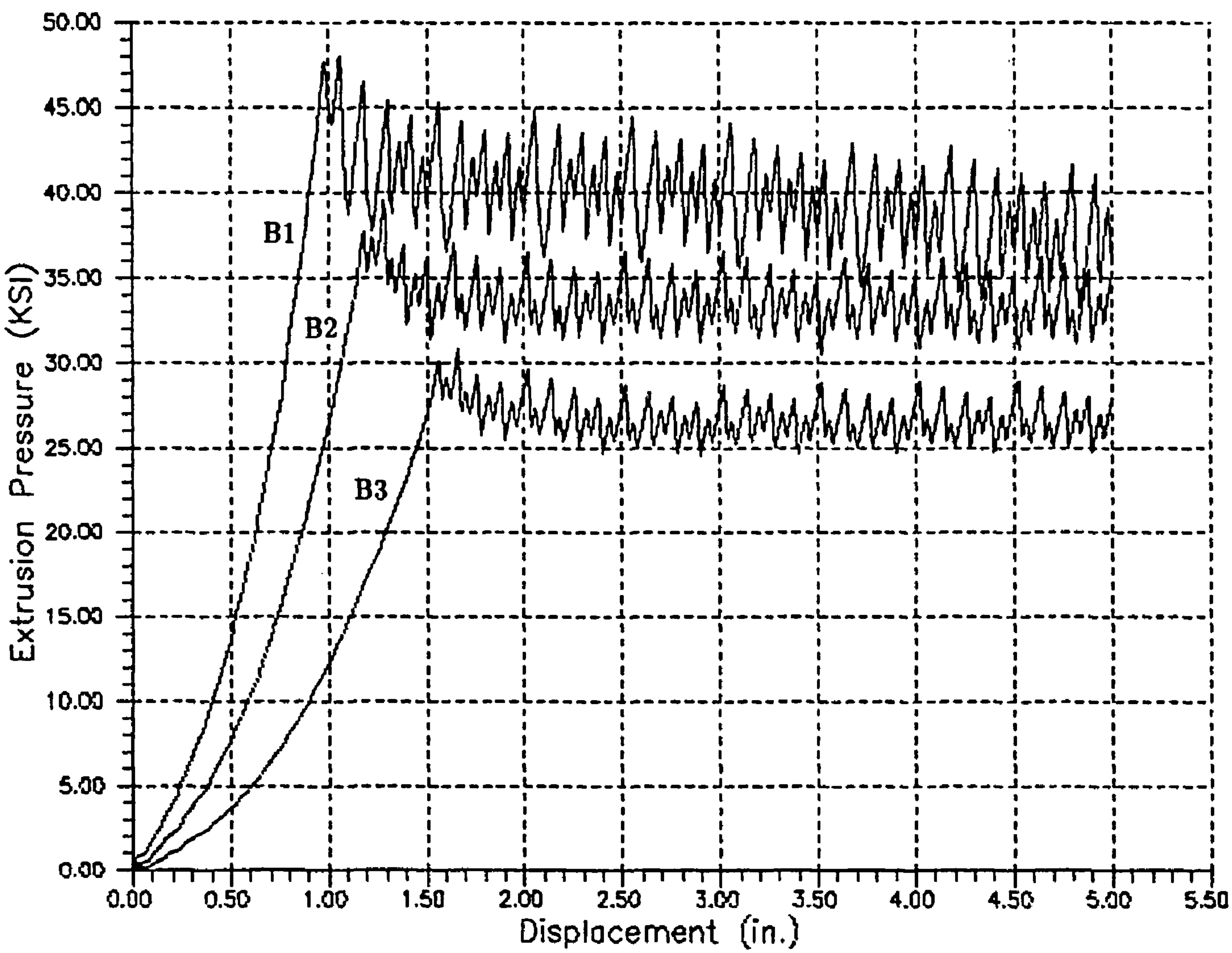


Figure 5.16. *Extrusion Pressure Versus Displacement for Runs B1, B2, and B3.*

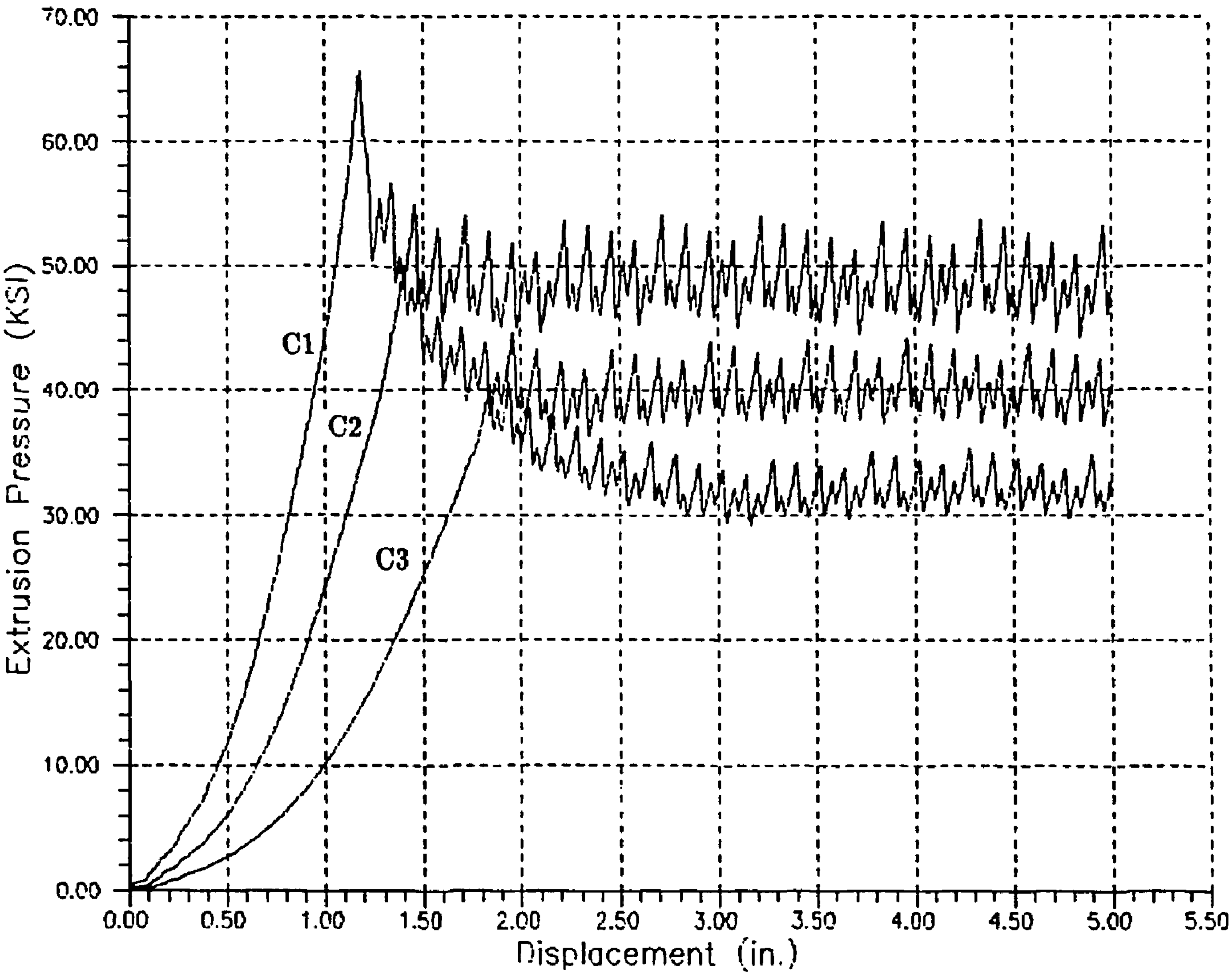


Figure 5.17. *Extrusion Pressure Versus Displacement for Runs C1, C2, and C3.*

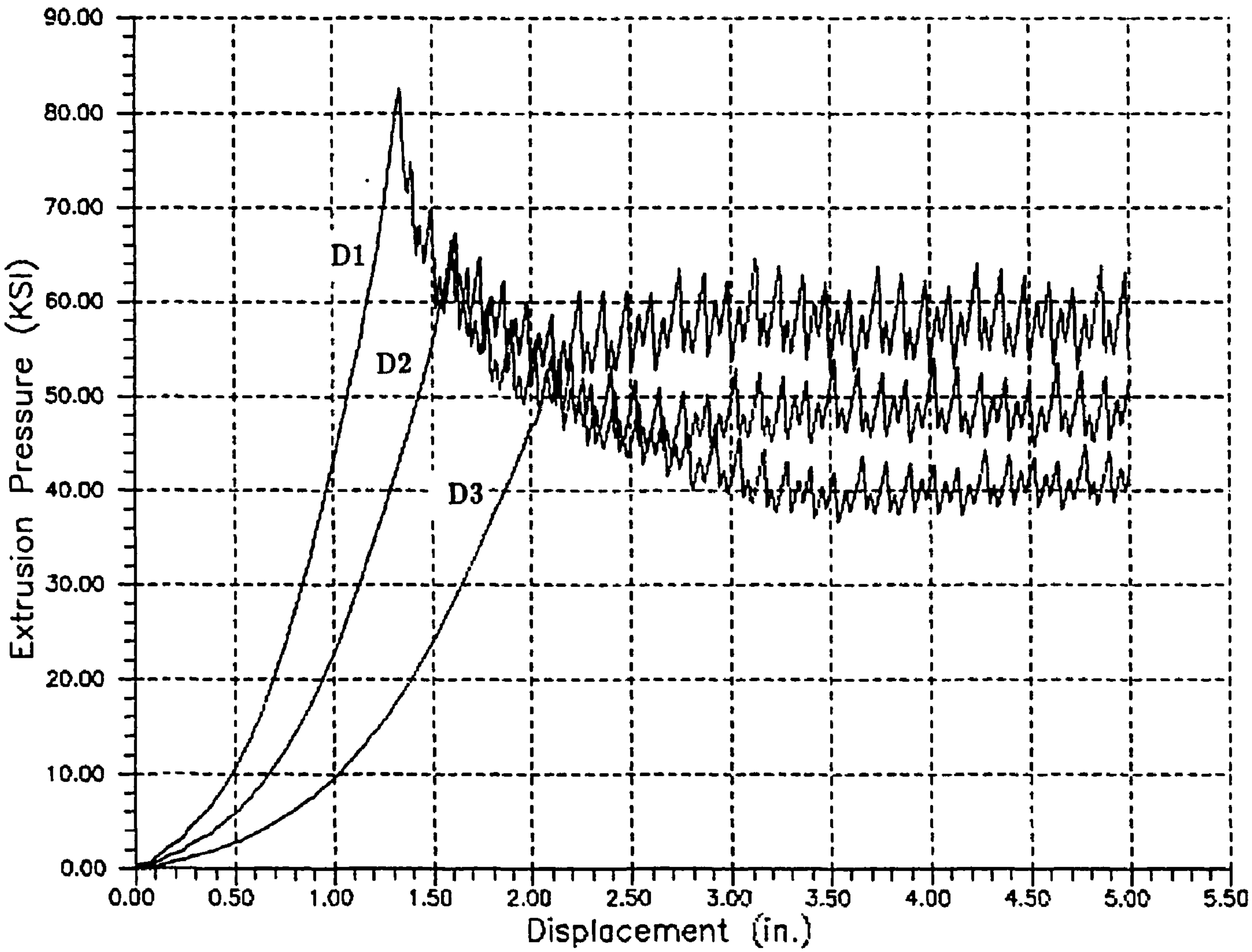


Figure 5.18. *Extrusion Pressure Versus Displacement for Runs D1, D2, and D3.*

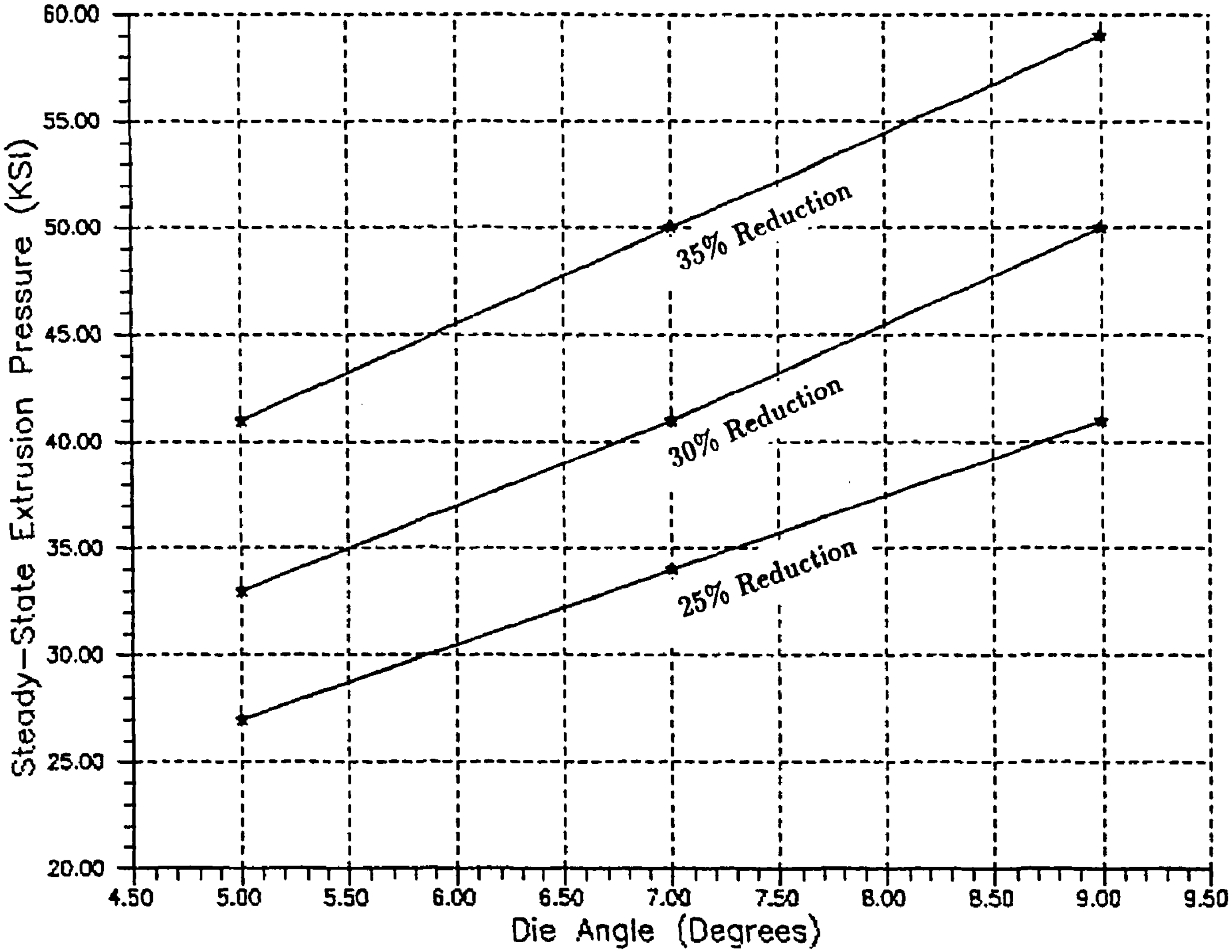


Figure 5.19. *Steady-State Extrusion Pressure Versus the Variation in the Die Angle.*

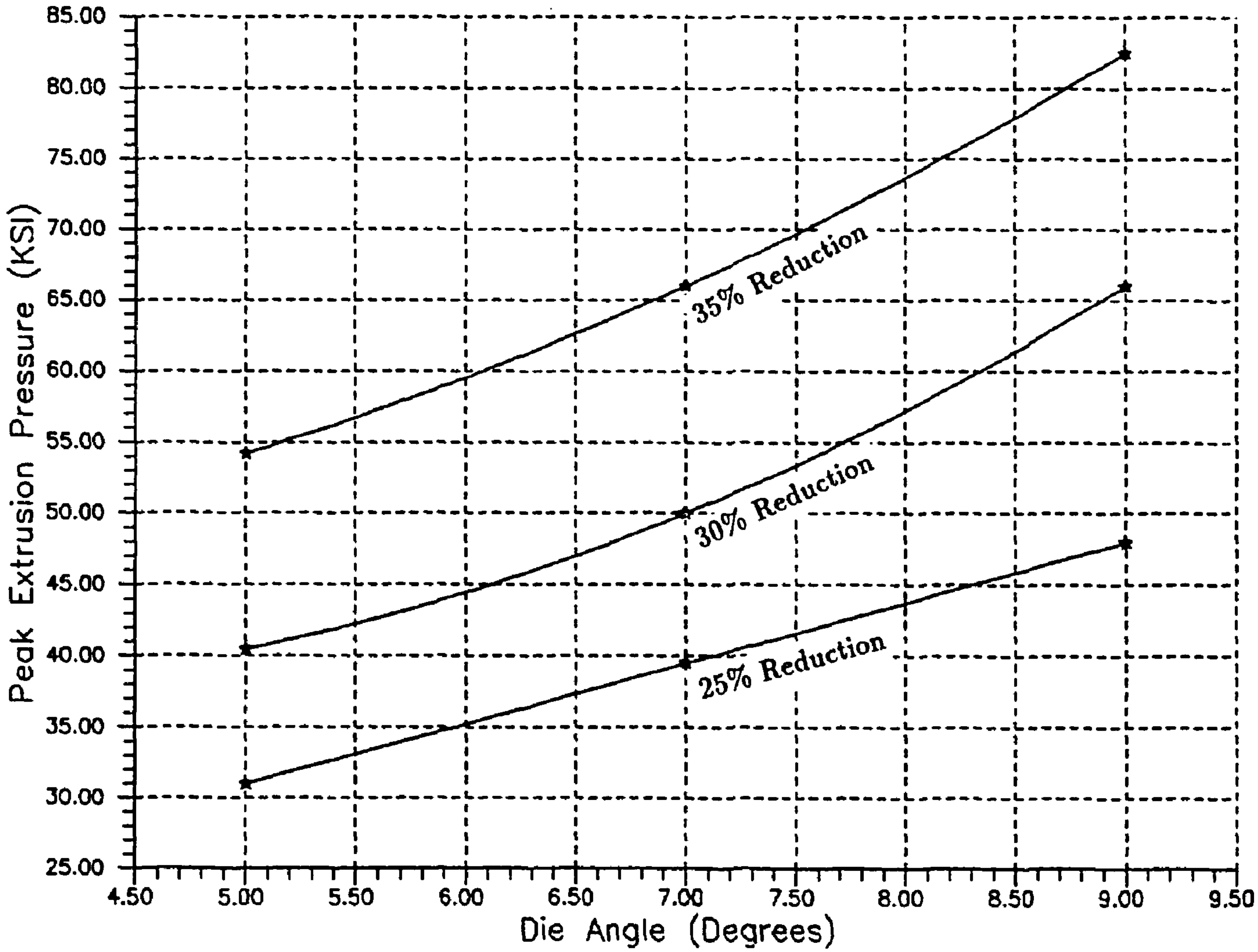


Figure 5.20. *Peak Extrusion Pressure Versus the Variation in the Die Angle.*

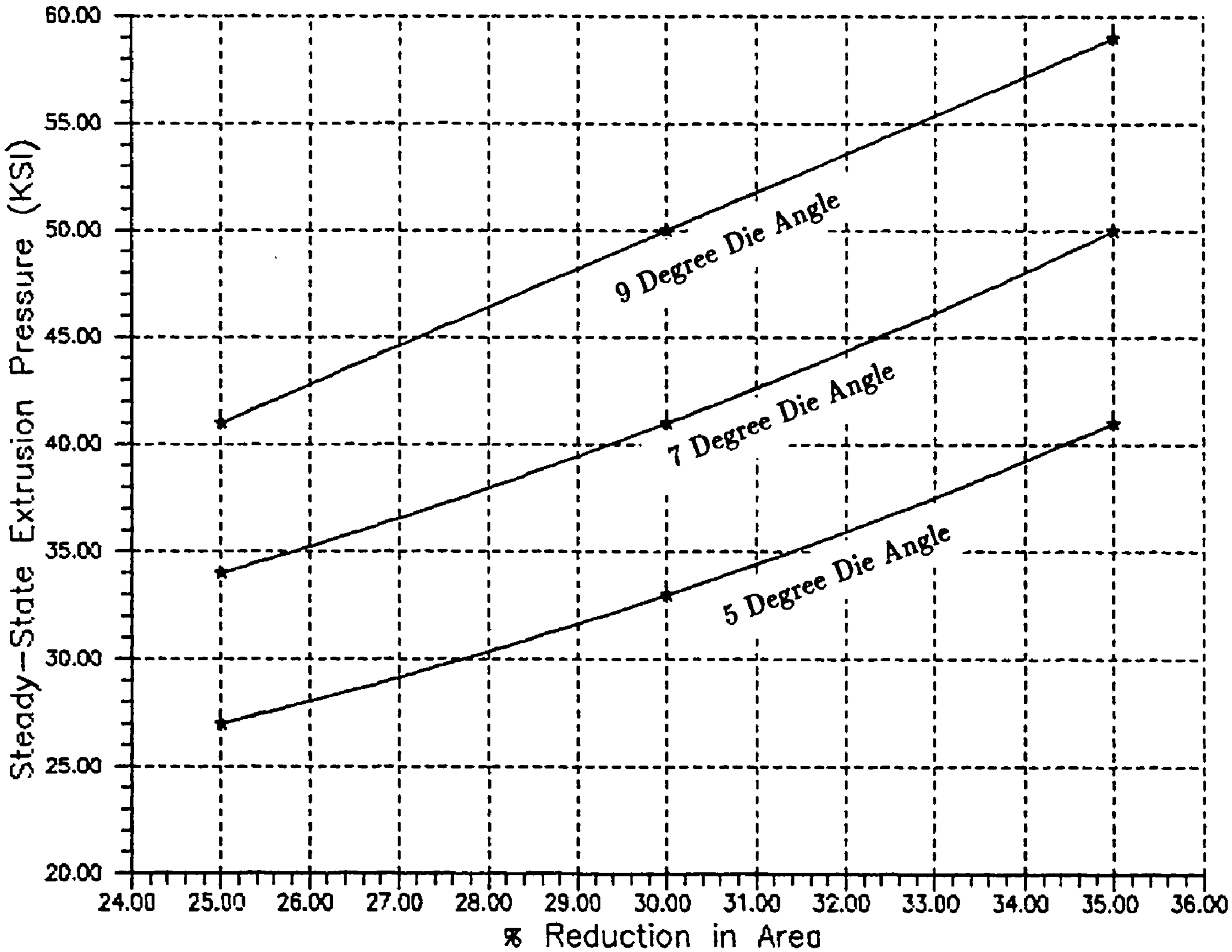


Figure 5.21. ***Steady-State Extrusion Pressure Versus the % Reduction in Area.***

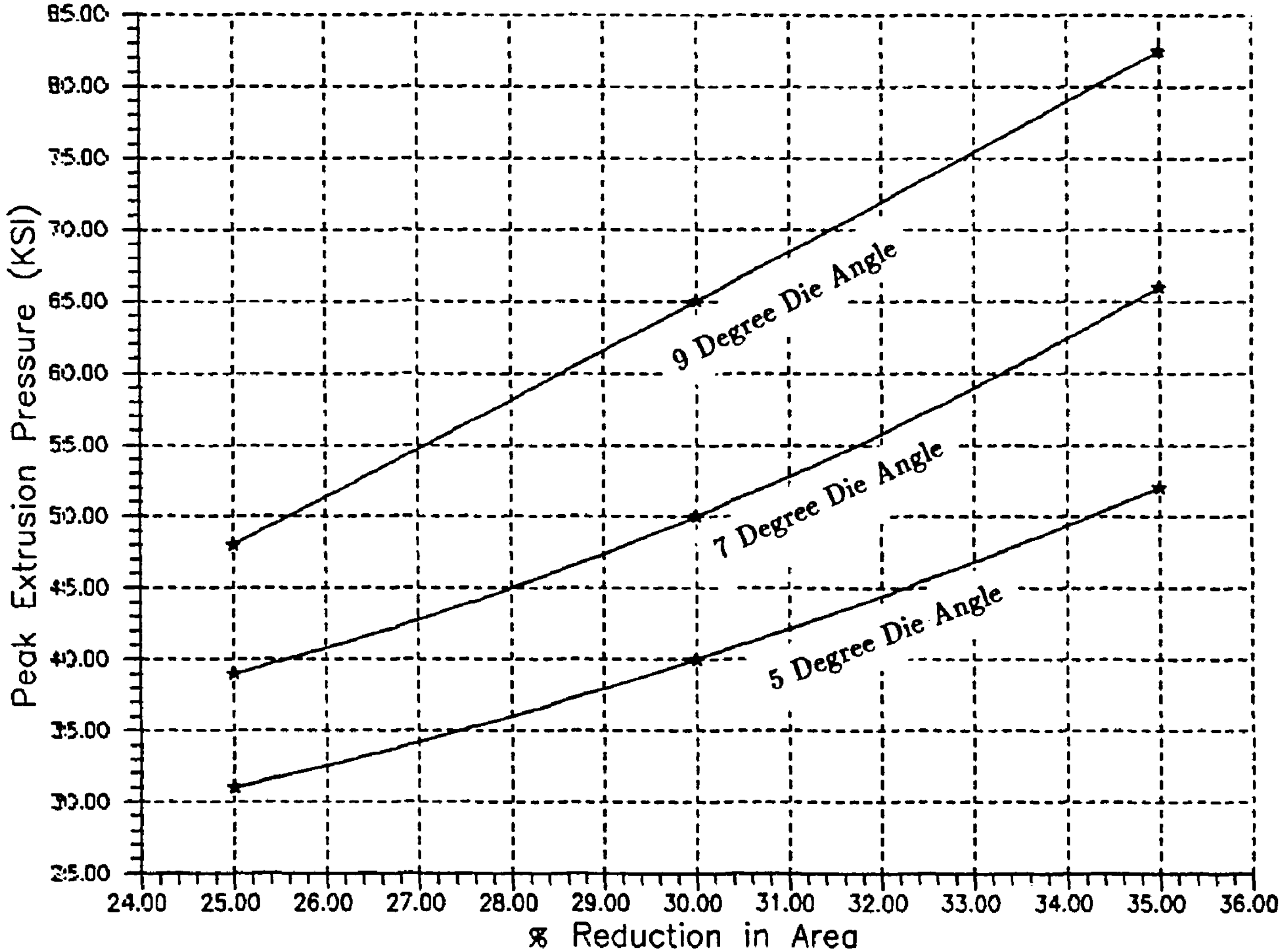


Figure 5.22. *Peak Extrusion Pressure Versus the % Reduction in Area.*

6. UNIFES USERS GUIDE

6.1. Introduction

In this chapter the reader is familiarized with the UNIFES input commands and the syntax that must be followed in order for these commands to be recognized properly by the program.

UNIFES uses unformated input files which may include character strings as well as numeric data. Character strings should not be enclosed in quotes. The command processor COMPRO selects the first four characters from each character string and uses them to form a simplified four byte long command which is then recognized by the program. A blank space may not be used in a character string which represents a single command. However, a blank space may be used in between different commands and/or numeric data. For example the character string INTERFACE_NODES is recognized as a single command. The command processor selects the first four characters INTE to have a special meaning. The character strings INTE or INTEABCDEFG are interpreted by the program to have the same meaning as the character string INTERFACE_NODES. Notice that the underline between the words INTERFACE and NODES is used to indicate to the program that the string NODES is not a separate command. The character string INTERFACE NODES with a blank space between the words INTERFACE and NODES is considered by the program as two commands INTE and NODE.

UNIFES commands may be entered on the same line or different lines as long as all the information for the commands fit on the same line. Furthermore, there is no order of sequence for most commands. The reader should refer to the description of each command for complete detailes.

6.2 Input Commands

The commands listed in this section are organized in the alphabetic order. Some commands may have a series of sub-commands associated with them. In this case the sub-commands are described as part of the overall description for the parent command.

CONNECTIVITY — CONN n

This command is equivalent to the INCIDENCES command. The CONNECTIVITY command tells the program to look for and generate element connectivities. Integer 'n' is the number of elements for which the connectivity information is provided for or is to be generated. The CONNECTIVITY command may be entered more than once. The information provided by the last incidence of the CONNECTIVITY command over rides all the preceding recurances of this command.

Example:

```
CONNECTIVITY 5
        1 2308 8   1  6  8  3  4  7  5  2      0
        4 2308 8  16 21 23 18 19 22 20 17      1
        5 2308 8  21 26 28 23 24 27 25 12      0
```

The syntax of the lines that follow the CONNECTIVITY command is

no it nn n_1 n_2 n_3 ... n_{nn} igen

where

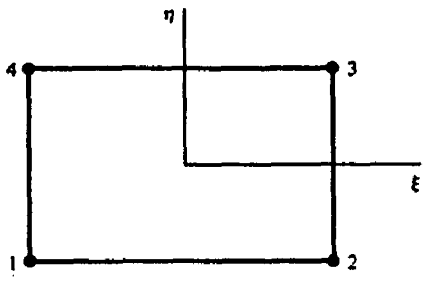
no - is the element number

it - is a four digit element identifier (refer to the note)

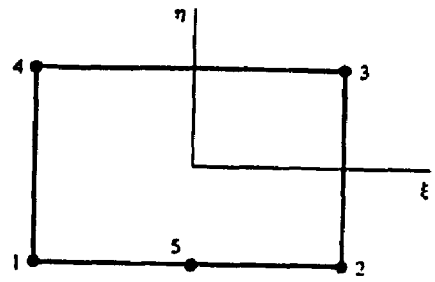
nn - is the number of nodes in the element

n_n - is the global node number associated with the element

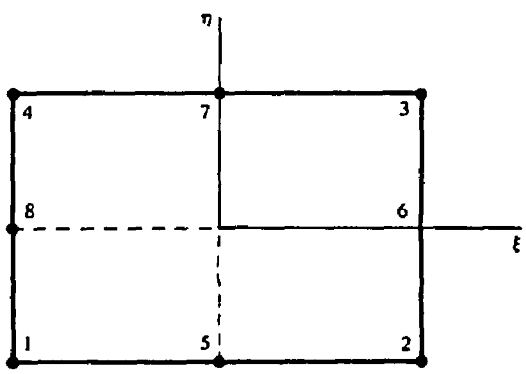
igen - is the increment of the generated element numbers



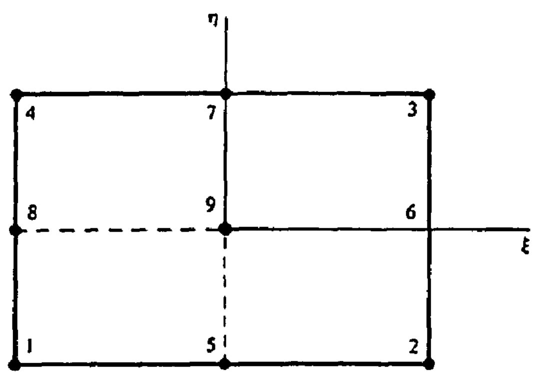
2104 - Plane Stress
2204 - Plane Strain
2304 - Axisymmetric



2105 - Plane Stress
2205 - Plane Strain
2305 - Axisymmetric

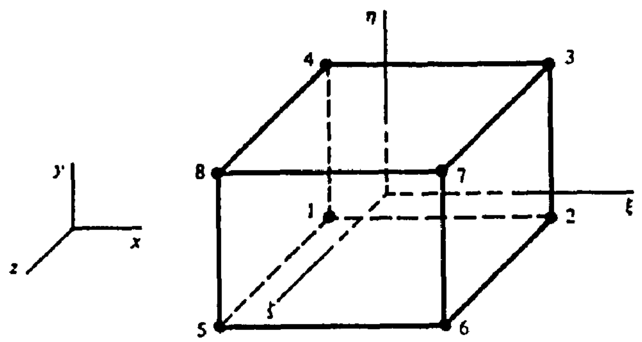


2108 - Plane Stress
2208 - Plane Strain
2308 - Axisymmetric

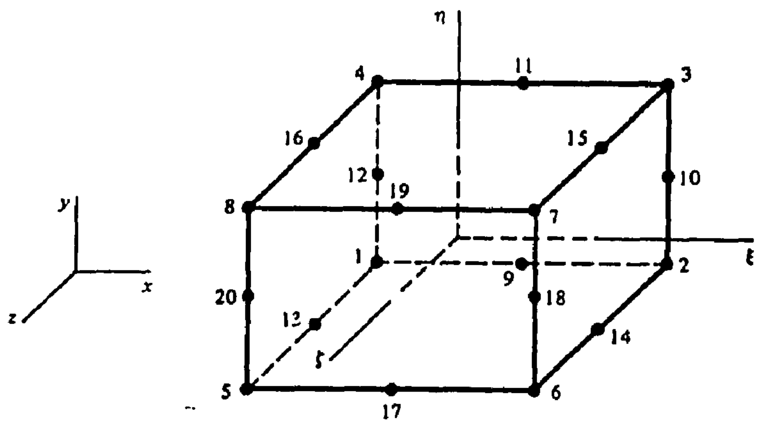


2109 - Plane Stress
2209 - Plane Strain
2309 - Axisymmetric

Figure 6.1. *Sequence of Node Numbers and Identification Codes for 2D Quadrilateral Elements.*



Element 3008



Element 3020

Figure 6.2. *Sequence of Node Numbers and Identification Codes for 3D Solid Elements.*

In the above example the CONNECTIVITY command tells the program to look for or generate five elements. Elements 1 through 4 are generated by using the information on the first two lines. This is because the 'igen'=1 on the second card. The information for element 5 is read from the third card. Value of the 'it' parameter in the above example is 2308 which corresponds to a 2D axisymmetric eight noded quadrilateral element. Parameter 'nn'=8 indicates that the element is eight noded, hence the program looks for eight node numbers subsequent to 'nn'. The sequence of node numbers for each element is shown in Figures 6.1 and 6.2. Also, notice that the parameter 'igen' must be entered even if its value is zero. If the number of elements inputted or generated does not match the value of 'n' entered after the CONNECTIVITY command an error will occur.

NOTE: The four digit integer 'it' has the form 'lmnn' where the first digit 'l' represents the dimension of the element, the second digit 'm' is zero for 3D elements and has values 1,2, or 3 for 2D elements corresponding to plane stress, plane strain, and axisymmetric problems respectively. The final two digits 'nn' correspond to the number of nodes in the element.

COORDINATES — COOR n

This command is equivalent to the NODES and JOINTS commands. The COORDINATES command tells the program to look for and generate nodal coordinates. The generated nodes are along a straight line connecting the starting node and the final node. Integer 'n' is the number of nodes for which the coordinate information is provided or is to be generated. The COORDINATES command may be entered more than once. The information provided by the last incidence of the COORDINATES command over rides all the preceding recurances of this command.

Example:

```
COORDINATES  8
          1   0.0   0.0   0
          3   0.0   2.0   1
          4   1.0   0.0   0
          5   1.0   2.0   0
          6   2.0   0.0   0
          8   2.0   2.0   1
```

In the above example the COORDINATES command is used to enter and generate data for 8 nodes in a two dimensional problem. The syntax of the lines that follow the COORDINATES command is

node   x   y   (z)   igen

The integer 'node' is the node number and real numbers x, y, and z refer to the nodal coordinates in the global reference frame. Notice that the z coordinate component is needed for 3D problems only. The parameter 'igen' is used to increment the node numbers when generating nodes. if 'igen'=0 nodes are not generated. Otherwise, if 'igen' is not zero, nodes are generated at increments 'igen' starting with the node specified on the previous input line.

In the above example node 2 is generated from the information provided by nodes 1 and 3. The same thing is also true for node 7 which is generated form the information provided by nodes 6 and 8. If the number of nodes inputed or generated does not match the value of 'n' entered after the COORDINATES command an error will occur.

CURVE — CURV n

This command is used to identify the parameters required for defining the Hermite parametric curves which are used to model the contact boundaries (for description of Hermite curves refer to Chapter 4). Integer 'n' which follows the CURVE command identifies the curve for which the information is being provided. For each Hermite spline a separate CURVE command must be issued. A maximum of 6 splines may be defined through the use of this command. Sub-commands PAX and PAY refer to the x and y coordinates of the starting point on the curve while PBX and PBY refer to the end point coordinates. In addition, sub-commands RAX, RAY, RBX and RBY are used to define the components of the tangents to the curve at the starting point and the end point respectively. Furthermore, sub-commands XZL and YZB define the x and y coordinates of the lower left corner of the inclusion zone, while XZR and YZT define the uper right corner of the inclusion zone.

Example:

```
CURVE 1
        PAX 1. PAY 1. PBX 3. PBY 2.
        RAX 2. RAY 0.5 RBX 1. RBY 1.
        XZL 1. YZB 1. XZR 4. YZT 4.
```

In the above example the CURVE command and the sub-commands associated with it are used to define the curve number one. Notice that the sub-commands of the CURVE command may be entered in any order or on multiple number of lines.

DIMENSION — DIME n

This command is used to indicate whether the problem to be solved is a two dimensional or three dimensional problem. Integer 'n' may have a value of 2 or 3. This command must be entered before the commands COORDINATES, NODES, and JOINTS. The default value for this command is n=2.

DISPLACEMENTS — DISP

This command is used to enter the displacements associated with each node. If a node is fixed the appropriate displacements must be set to zero. This command is also used to identify the orientation of the local degrees of freedom if they are different from the global orientation. The DISPLACEMENT command may be entered more than once, however the necessary information for the entire structure may be specified by using this command once.

Example:

```
DISPLACEMENTS
        NODE 1 TO 3 X 0.
        NODES 1 TO 10 BY 3 X 0. Y 0.2 Z 0.1
        NODES 6 TO 20 X 0. TX 45. TY 45. TZ 60.
        NODE 30 Y 0.
END
```

In the above example the necessary syntax for using the sub-commands of the DISPLACEMENT command is shown. Notice that the END command is necessary to terminate the DISPLACEMENT command. The sub-commands TX, TY, and TZ refer to the direction angles of the normal to the surface which containes the skew roller boundary condition. For a two dimensional problem TZ=90 degrees. The X, Y, and Z refer to the global x, y, and z displacements if TX, TY, and TZ direction angles are not specified. When the direction angles are specified X refers to the displacement in the direction normal to the inclined boundary. The BY sub-command indicates the size of the increments in the node numbers. If BY is not used the default is 'BY 1'. The TO sub-command is used to indicate the range of nodes effected by the displacement values. If TO is not used only the node identified by the NODE sub-command is effected.

ELEMENT — ELEM

This command is used to define the integration order, thickness and material type of the elements. The following example shows the necessary syntax for this command.

Example:

```
ELEMT 1 NIPXI 2 NIPETA 2 MATERIAL 1 THICK 1.
ELEM 2 TO 10 NIPXI 2 NIPETA 2 MATERIAL 1 THICK 1.
ELEM 1 TO 7 BY 2 NIPXI 3 NIPETA 3 MATERIAL 1 THICK 1.
ELEM 1 TO 7 BY 2 NIPXI 3 NIPETA 3 NIPSI 3 MATERIAL 1
ELEM 1 TO 7 BY 2 IRONS 150 MATERIAL 1
```

In the above example the sub-commands TO and BY are used to indicate the range and increment of the element numbers effected by the ELEMENT command respectively. If these sub-commands are not used, as in the first line, then only the element specified by the ELEMENT command will be effected. The sub-subcommands NIPXI, NIPETA, and NIPSI refer to the number of integration points in the element's local coordinate system ξ, η and ζ. The number of integration points specified for each direction may vary from 1 to

4. There are no defaults set for integration parameters, hence they must be specified for all elements. The IRONS sub-command specifies the alternative 14 and 15 point integration orders for 3D solid elements. If 'IRONS 150' is specified the 15 point integration rule will be used. 'IRONS 140' refers to the 14 point integration rule.

The MATERIAL sub-command identifies the material associated with each element. The material properties are defined by the global MATERIAL command. The THICKNESS sub-command assigns a thickness to the plane stress and plane strain elements.

INCIDENCES — INCI n

For information on this command refer to the CONNECTIVITY command.

INCREMENTS — INCR n

This command is used to specify the number of load increments. The default value for 'n' is one. When the INCREMENTS command is used the displacements and loads which are specified by the DISP and LOAD commands, respectively, are divided into 'n' equal portions and are then applied to the structure one portion at a time. If the ITERATIONS command is used then within each load increment iterations are performed until convergence is detected (for information on the convergence criterion used refer to chapter 5).

INTERFACE_NODES— INTE n

This command is used to define the nodes which can come in contact with the contact boundaries which are defined by the CURVE command. Integer 'n' tells the program to look for or generate 'n' interface nodes.

Example:

```
INTERFACE_NODES 11
          1  0
          2  0
          20  2
```

In the above example 11 interface nodes are defined. Each line that follows the INTE command has the syntax

no   igen

where 'no' is the node number and 'igen' is the increment at which the node numbers are generated.

ITERATIONS — ITER n

This command is used to specify the maximum number of iterations allowed for each load increment. If the problem converges to the prescribed tolerance at a fewer number of iterations, then it proceeds to the next load increment. The default value for 'n' is one. The following example illustrates the sub-commands associated with the ITERATIONS command.

Example:

```
ITERATIONS 10  FACLOW 0.0001 FACHIGH 0.001
          STOP_AFTER_DIVERG 4
```

In the above example a maximum of 10 iterations are allowed for each load increment. The FACLOW value is used to check for convergence. If convergence is detected prior to the tenth iteration, the program procedes to the next load increment. If convergence is not detected prior to the tenth increment, the value of FACHIGH is used to check for convergence. If FACHIGH is satisfied then the solution is accepted and the program procedes to the next load increment. However, if the convergence criterion is not satisfied, then the automatic subincrementation is invoked which would reduce the load increment into one half of its value. This process may be repeated up to three times. The STOP_AFTER_DIVERG sub-command tells the program to terminate execution if four consecutive divergences are detected.

JOINTS — JOIN n

For information on this command refer to the COORDINATES command.

LINEAR — LINE

This command is used to indicate that the problem to be solved is geometrically linear. Notice that material non-linearities may still exist. In that case the commands INCREMENTS and ITERATION should be used to perform the non-linear analysis. If the material used is also linear then INCREMENTS and ITERATIONS need not be specified.

LOADS — LOAD

This command is used to enter the equivalent nodal forces associated with each node. The LOADS command may be entered more than once, however the necessary information for the entire structure may be specified by using this command only once.

Example:

```
LOADS
        NODE 1 TO 3 X 20.
        NODES 1 TO 10 BY 3 X 5. Y 10. Z 14.
        NODES 6 TO 20 X 90.
        NODE 30 Y 6.
END
```

In the above example the necessary syntax for using the sub-commands of the LOADS command is shown. Notice that the END command is necessary to terminate the LOADS command. The X, Y, and Z refer to the global x, y, and z components of the nodal forces. The BY sub-command indicates the size of the increments in the node numbers. If BY is not used the default is 'BY 1'. The TO sub-command is used to indicate the range of nodes effected by the load values. If TO is not used only the node identified by the NODE sub-command is effected.

MATERIAL — MATE n

This command is used to define the material properties for each material used. Up to seven materials may be defined by the MATERIAL command. The current version of UNIFES allows two types of materials which may be defined by the TYPE sub-command of the material command. 'TYPE 1' refers to isotropic elasticity, whereas 'TYPE 2' refers to the elasto-plastic material model which is described in detail in Chapter 3. The following example illustrates use of the additional sub-commands which are associated with the MATERIAL command.

Example:

```
MATERIAL 1 TYPE 1 E 10600. NU 0.3
MATERIAL 2 TYPE 1 E 10600. NU 0.3 WX 2. WY 0. WZ 0.
MATERIAL 3 TYPE 2 E 10600. NU 0.3 ISOTROPIC 40.
        KINEMATIC 114. YIELD 56.
```

In the above example E and NU refer to the elastic modulus and the Poisson's ratio respectively. The sub-commands WX, WY and WZ represent the global components of the specific weight of the material. Materials 1 and 2 defined above are linear elastic materials. Material 3 is an elasto-plastic material with uniaxial yield strength of 56. The isotropic and kinematic hardening parameters for this material are 40 and 114 respectively.

NODES — NODE n

For information on this command refer to the COORDINATES command.

NONLINEAR — NONL

This command is used to indicate that the problem to be solved is geometrically nonlinear. When this command is used, the commands INCREMENTS and ITERATIONS should also be used to perform the non-linear analysis.

NONSYMMETRIC — NONS

This command is used to indicate that the problem to be solved requires assembly and solution of a non-symmetric system of equations. This option is required when the plasticity material model is used. The default is set to SYMMETRIC

OUTPUT — OUTP n

When this command is used, outputs are provided for every nth load increment. If this command is omitted the outputs are provided for the final load increment.

RESTART — REST

This command tells the program to use the solution obtained at the end of the last analysis as the initial condition for the current analysis. When this command is used, the values that are specified through the LOADS and the DISPLACEMENTS commands represent the additional loads or displacements, respectively, that are to be applied to the structure. User must also make sure that the proper files are provided for the analysis.

SYMMETRIC — SYMM

This command is used to indicate that the problem to be solved requires assebly and solution of a symmetric system of equations. This option is the default.

TITLE — TITL n

This command tells the program to look for 'n' title cards which immediately follow the TITLE command. Title cards will be printed at the beginning of the output file.

7. CONCLUSIONS

From this work it is observed that the solution of extrusion problems using the finite strain plasticity can be successfully achieved by means of a Total Lagrangian formulation. The proposed approach initialy formulates the constitutive model in the spatial reference frame and subsequently transforms it to the Lagrangian reference frame. It therefore by-passes the use of the Jaumann stress rate in the formulation of the problem and furthermore does not need to identify a proper co-rotational stress rate in the spatial reference frame.

It is also shown here that the computer graphics techniques used for representation of curves can be successfully applied to model curved boundaries in numerical solution of metal forming problems. In particular the Hermite formulation of parametric cubic curves provides a great flexibility in creating an assortment of differently shaped boundaries with minimal effort on the part of the analyst.

In this work it is also shown that the singularities and fluctuations in the extrusion pressure that may result at the exit of the die can be controlled by increasing the number of nodes that are in contact with the die. This can be achieved without any substantial increase in computation time by using transition elements as described in Chapter 5. The residual stress patterns and the strain distributions obtained in this work are identical to the results obtained by other researchers.

Any future research on the use of the Total Lagrangian formulation for the solution of finite strain metal forming problems, should concentrate more extensively on developing material models which are computationally more efficient and also exploring the applicability of parallel processing techniques in the finite element implementation of the constitutive relations. Each of the extrusion problems solved in this work required 7 to 10 hours of CPU time on an IBM 3090 with vector hardware. The majority of the execution time required by the program was devoted to evaluation of the stresses, plastic strains, shift stress tensor, and the plastic work.

REFERENCES

[1] Aravas, N. (1986). "The Analysis of Void Growth that Leads to General Bursts During Extrusion," *Journal of Mechanics of Physics and Solids*, Vol. 34(1), pp. 55-79.

[2] Atluri, S. N. (1984). "On Constitutive Relations at Finite Strain: Hypo-elasticity and Elasto-plasticity with Isotropic and Kinematic Hardening," *Computer Methods in Applied Mechanics and Engineering*, Vol. 43, pp.137-171.

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

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

[5] Dafalias, Y. F. (1983). "Corotational Rates for Kinematic Hardening at Large Plastic Deformations," *Journal of Applied Mechanics, Transactions of The ASME,* Vol. 50, pp. 561-565.

[6] Dafalias, Y. F. (1984). "The Plastic Spin Concept and a Simple Illustration of its Role in Finite Plastic Transformations," *Mechanics of Materials*, Vol. 3, pp. 223-233.

[7] Dafalias, Y. F. (1985). "The Plastic Spin," *Journal of Applied Mechanics, Transactions of the ASME*, Vol. 52, pp. 865-871.

[8] Foley, J. D. and Van Dam, A. (1984). *Fundamentals of Interactive Computer Graphics*, Addison Wesley Publishing Company, Inc., Philippines.

[9] Gerald, C. F. and Wheatley, P. O. (1985). *Applied Numerical Analysis*, Addison-Wesley Publishing Company, Inc., June.

[10] Ghosh, S. and Kikuchi, N. (1988). "Finite Element Formulation of the Simulation of Hot Sheet Metal Forming Processes," *International Journal of Engineering Sciences*, Vol. 26(2), pp. 143-161.

[11] Green, A. E. and Naghdi, P. M. (1965). "A General Theory of an Elastic-Plastic Continuum," *Archive for Rational Mechanics and analysis*, Vol. 18, pp. 251-281.

[12] Green, A. E. and Naghdi, P. M. (1966). "A Thermodynamic Development of Elastic-Plastic Continuua," *Proceedings, IUTAM Symposium on Irreversible Aspects of Continuum Mechanics and Transfer of Physical Characteristics in Moving Fluids*, Springer-Verlag, pp. 117-131.

[13] Hallquist, J. O., Goudreau, G. L. and Benson, D. J. (1985). "Sliding Interfaces with Contact-Impact in Large-Scale Lagrangian Computations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 51, pp. 107-137.

[14] Hibbitt, H. D., Marcal, P. W. and Rice, J. R. (1970). "A Finite Element Formulation For Problems of Large Strain and Large Displacement," *International Journal of Solids and Structures*, Vol. 6, pp. 1069-1086.

[15] Hill, R. (1958). "A General Theory of Uniqueness and Stability in Elastic Plastic Solids," *Journal of Mechanics of Physics and Solids*, Vol. 6, pp. 236

[16] Johnson, G. C. and Bammann, D. J. (1984). "A Discussion of Stress Rates in Finite Deformation Problems," *International Journal of Solids and Structures*, Vol. 20(8), pp. 725-737.

[17] Kanazawa, K. and Marcal, P. V. (1978). "Finite Element Analysis of the Steel Rolling Process," *Applied Numerical Methods to Forming Processes AMD-28*, pp. 81.

[18] Klie, W. (1979). "Plastic Deformation with Free Boundaries: A Finite Element Approach," *Metal Forming Plasticity* (edited by H. Lippmann), Springer-Verlag, Berlin, pp. 260.

[19] Kobayashi, S. and Lee, C. H. (1973). "Deformation Mechanics and Workability in Upsetting Solid Circular Cylinders," *Proceedings of the North American Metalworking Res. Conference*, McMaster University, Vol. 1, pp. 185.

[20] Kobayashi, S. (1977). "Rigid-Plastic Finite Element Analysis of Axisymmetric Metal Forming Processes,"*Numericl Modelling of Manufacturing Processing*, ASME, PVP-PB-25, pp. 49.

[21] Lee, B. C. and Kwak, B. M. (1984). " A Computational Method for Elasto-Plastic Contact Problems," *Computers & Structures*, Vol. 18(5), pp.757-765.

[22] Lee, C. H. and Kobayashi, S. (1973). "New Solutions to Rigid-Plastic Deformation Problems Using a Matrix Method," *Journal of Engineering in Industry, Transactions of the ASME,* pp. 864.

[23] Lee, E. H. (1969). " Elastic-Plastic Deformation at Finite Strains," *Journal of Applied Mechanics, Transactions of the ASME,* Vol. 91(3).

[24] Lee, E. H. (1976). "The Basics for an Elastic-Plastic Code," Stanford University, SUDAM 76-1.

[25] Lee, E. H., Mallett, R. L. and Wertheimer, T. B. (1983). "Stress Analysis for Anisotropic Hardening in Finite-Deformation Plasticity," *Journal of Applied Mechanics, Transactions of the ASME*, Vol. 50, pp. 554-560.

[26] Lee, E. H., Mallett, R. L. and Yang, W. H. (1977). "Stress and Deformation Analysis of Metal Extrusion Process," *Computer Methods in Applied Mechanics and Engineering*, Vol. 10, pp. 339.

[27] McMeeking, R. M. and Rice, J. R. (1975). "Finite Element Formulation for Problems of Large Elastic-Plastic Deformation," *International Journal of Solids and Structures*, Vol. 11, pp. 601-616.

[28] Nagtegaal, J. C. and de Jong, J. E. (1982). "Some Aspects of Non-isotropic Workhardening in Finite Strain Plasticity," *Proceedings of the Workshop on Plasticity of Metals at Finite Strain: Theory, Experiment and Computation,*

Lee, E. H. and Mallett, R. L., eds. Division of Applied Mechanics, Stanford University and Department of Mechanical Engineering, R.P.I., pp. 64-102.

[29] Nemat-Masser, S. (1979). "Decomposition of Strain Measures and their Rates in Finite Deformation Elasto-Plasticity," *International Journal of Solids and Structures*, Vol. 15, pp. 155-166.

[30] Oh, S. I. (1982). "Finite Element Analysis of Metal Forming Processes with Arbitrarily Shaped Dies," *International Journal of Mechanical Sciences*, Vol. 24(8), pp. 479-493.

[31] Okamoto, N. and Nakazawa, M. (1979). "Finite Element Incremental Contact Analysis with Various Frictional Conditions," *International Journal for Numerical Methods in Engineering,* Vol. 14, pp. 337-357.

[32] Ortiz, M. and Popov, E. P. (1985). " Accuracy and Stability of Integration Algorithms for Elastoplastic Constitutive Relations," *International Journal for Numerical Methods in Engineering*, Vol. 21, pp. 1561-1576.

[33] Ostachowicz, W. (1984). "Mixed Finite Element Method for Contact Problems," *Computers & Structures*, Vol. 18(5), pp. 937-945.

[34] Padovan, J., Tavichakchaikul, S. and Zeid, I. (1984). "Finite Element Analysis of Steadily Moving Contact Fields," *Computers & Structures*, Vol. 18(2), pp. 191-200.

[35] Potts, D. M. and Gens, A. (1985). "A Critical Assessment of Methods of Correcting for Drift from the Yield Surface in Elasto-Plastic Finite Element Analysis," *International Journal for Numerical and Analytical Methods in Geomechanics*, Vol. 9, pp. 149-159.

[36] Prager, W. (1956). "A New Method of Analyzing Stresses and Strains in Work-Hardening Plastic Solids," *Journal of Applied Mechanics, Transactions of the ASME*, pp. 493-496, December.

[37] Roll, K. (1978)."Calculation of Metal Forming Processes by Finite Element Methods," *Applied Numerical Methods to Forming Processes AMD-28*, pp. 67.

[38] Schreyer, H. L., Kulak, R. F. and Kramer, J. M. (1979). "Accurate Numerical Solutions for Elasto-Plastic Models," *Journal of Pressure Vessel Technology, Transactions of the ASME*, Vol. 101, pp. 226-234.

[39] Shiau, Y. C. and Kobayashi, S. (1988). "Three-Dimensional Finite Element Analysis of Open-Die Forging," *International Journal for Numerical Methods in Engineering*, Vol. 25, pp. 67-85.

[40] Shield, R. T. and Ziegler, H. (1958). "On Prager's Hardening Rule," *ZAMP, Zeitschrift Fur Angewand Mathematik und Physik,* Vol. 9, pp. 260-267.

[41] Simo, J. C. and Ortiz, M. (1985). "A Unified Approach to Finite Deformation Elastoplastic Analysis Based on the Use of Hyperelastic Constitutive Equations," *Computer Methods in Applied Mechanics and Engineering*, Vol. 49, pp. 221-245.

[42] Simo, J. C. and Taylor, R. L. (1985). "Consistent Tangent Operators for Rate-Independent Elastoplasticity," *Computer Methods in Applied Mechanics and Engineering*, Vol. 48, pp. 101-118.

[43] Truesdell, C. and Toupin, R. (1960). "The Classical Field Theories," *Handbuch der Physik* (S. Flugge, editor), Springer-Verlag, pp. 226-793.

[44] Voyiadjis, G. Z. (1984). "Experimental Determination of the Material Parameters of Elasto-plastic, Work-hardening Metal Alloys," *Materials Science and Engineering Journal,* Vol. 62(1), pp. 99-107.

[45] Voyiadjis, G. Z. and Kiousis, P. D. (1987). "Stress Rate and the Lagrangian Formulation of the Finite-Strain Plasticity for a von Mises Kinematic Hardening Model," *International Journal of Solids and Structures*, Vol. 23(1), pp. 95-106.

[46] Voyiadjis, G. Z. and Poe, A. A. (1986). "A Finite Element Method for Contact Problems with Finite Deformations and Mixed Boundary Conditions," *Studies in Applied Mechanics*, Vol. 11, Mechanics of Material Interfaces, Elsevier Publishing Company, pp. 111-130.

[47] Wang, N. M. and Budiansky, B. (1978). "Analysis of Sheet Metal Stamping by a Finite Element Method," *Journal of Applied Mechanics, Transactions of the ASME*, Vol. 45, pp. 73-82.

[48] Wanxie, Z. and Suming, S. (1988). "A Finite Element Method for Elasto-Plastic Structures and Contact Problems by Parametric Quadratic Programming," *International Journal for Numerical Methods in Engineering*, Vol. 26, pp. 2723-2738.

[49] Wifi, A. S. (1976). " An Incremental Complete Solution of the Stretch Forming and Deep Drawing of Circular Blank Using a Hemispherical Punch," *International Journal of Mechanical Sciences*,Vol. 18, pp. 23-31.

[50] Wifi, A. S. and Yamada, Y. (1980). "Finite Element Analysis of Metal Forming Processes," *1st Cairo University Conference on Mech. Design and Prod., Advances in Mech. Design and Prod.*, (edited by G. Shawki and S. Metwalli), Pergamon Press, Oxford.

[51] Yang, D. Y. and Yoon, J. H. (1989). "Finite Element Analysis of Steady-State Three-Dimensional Extrusion of Sections Through Curved Dies," *International Journal of Mechanical Sciences*, Vol. 31(2), pp. 145-156.

[52] Ziegler, H. (1959). "A Modification of Prager's Hardening Rule," *Quarterly of Applied Mathematics*, Vol. 17(1), pp. 55-65.

[53] Zienkiewicz, O. C. (1977). *The Finite Element Method* , 3rd edition, McGraw-Hill, New York.

[54] Zienkiewicz, O. C. and Godbole, P. N. (1974). "Flow of Plastic and Viscoplastic Solids with Special Reference to Extrusion and Forming Processes,"

International Journal for Numerical Methods in Engineering, Vol. 8, pp. 3-16.

[55] Zienkiewicz, O. C. and Godbole, P. N. (1975)."A Penalty Function Approach to Problems of Plastic Flow of Metals with Large Surface Deformations," *Journal of Strain Analysis*, Vol. 10, pp. 180-183

[56] Zienkiewicz, O. C., Jain, P. C. and Onate, E. (1978). "Flow of Solids During Forming and Extrusion: Some Aspects of Numerical Solutions," *International Journal of Solids and Structures*, Vol. 14, pp. 15.

APPENDIX A

UNIFES

```
      PROGRAM UNIFES
C     12/15/88
C =====================================================================
C I                                                                   I
C I   P R O G R A M:                                                  I
C I                        U N I F E S                                I
C I                                                                   I
C I                UNIFIED FINITE ELEMENT SOLVER                      I
C I                                                                   I
C I      VERSION 1.0                                                  I
C I                                                                   I
C I   O B J E C T I V E:                                              I
C I                                                                   I
C I      ANALYISIS OF STRESSES STRAINS AND DEFORMATIONS INDUCED ON    I
C I      ELASTO-PLASTIC SOLID CONTINUUM AS THE RESULT OF VARIOUS      I
C I      FORCE OR DISPLACEMENT LOADINGS.                              I
C I                                                                   I
C I   D E V E L O P E D   B Y:                                        I
C I                                                                   I
C I                         MEHRDAD FOROOZESH                         I
C I                                                                   I
C I      ADVISOR: DR. GEORGE Z. VOYIADJIS                             I
C I      DEPARTMENT OF CIVIL ENGINEERING                              I
C I      LOUISIANA STATE UNIVERSITY, BATON ROUGE, LA 70803            I
C I                                                                   I
C I   PARTIALLY SUPPORTED BY:                                         I
C I                                                                   I
C I      NATIONAL SCIENCE FOUNDATION GRANT NO. MSM-8800832            I
C I                                                                   I
C =====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1          NINODE
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/CONTR1/INCREM,NIT
      COMMON/SAFE1/SKG(600000),SKGL(600000)
      COMMON/SAFE2/R(8000),IDOF(8000),JDIAG(8000)
      CALL INITAL(IDOF)
C
C ----- IOUT = OUTPUT DEVICE NUMBER
C
      IOUT = 13
C
C ----- READ THE INPUT FILE
C
```

```
      CALL INPUT(IDOF,IOUT,IERROR)
      IF (NINODE.GT.0) THEN
           CALL HERMIT
           CALL COEFIC
      END IF
C
C ----- INITIATE THE PLOTTING ROUTINES
C
      IF (IFPLOT.GT.0) THEN
           CALL INPLOT( NELEM )
           IF(IFLAG3.EQ.1.AND.NLINC.EQ.0) THEN
           INCREM = 1
           MDF = NNODES*NNDF
           CALL RECOV(MDF,ISTART,NTDF,IDOF)
           CALL PLOTER(SKG,NNODES,NELEM,NNDF,IDIM,NINODE,IFLAG1,IOUT)
           GO TO 90
           END IF
      END IF
C
C ----- DEFINE THE GLOBAL DEGREES OF FREEDOM
C ----- FIND THE BANDWIDTH AND THE LOCATION OF THE DIAGNAL TERMS
C       IN THE GLOBAL STIFFNESS MATRIX
C
      IF(IFLAG3.EQ.0) THEN
           CALL GLOB1(NNODES,NNDF,NTDF,IDOF)
           CALL DIAGNL(NELEM,NNDF,NTDF,IDOF,JDIAG,NTSK,MBAND,IFLAG2,
     1     IOUT)
      END IF
C
C ----- ASSEMBLE THE LOAD VECTOR
C
      CALL LOAD(R,IERROR)
C
C ----- CALL THE SOLUTION CONTROL UNIT
C
      CALL CONTRL(SKG,SKGL,R,IDOF,JDIAG,NTSK,NTDF,IOUT,MBAND)
C
C ----- CLOSE THE PLOT FILES
C
 90   IF (IFPLOT.GT.0) THEN
           CALL ENDPLT
      END IF
C
 100  STOP
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= C O N T R L ==============================
C
      SUBROUTINE CONTRL(SKG,SKGL,R,IDOF,JDIAG,NTSK,NTDF,IOUT,MBAND)
C
C ====================================================================
C I                                                                  I
C I    P R O G R A M:                                                I
C I                                                                  I
C I    SUBROUTINE 'CONTRL' CONTROLS THE INCREMENTAL LOADING AND THE   I
C I    NEWTON RAPHSON ITERATIVE PROCESS FOR THE TOTAL LAGRANGIAN     I
C I    GEOMETRIC AND MATERIAL NONLINEARITIES.                        I
C I                                                                  I
C I    A R G U M E N T  L I S T:                                     I
C I                                                                  I
C I    SKG(I)      =  GLOBAL STIFFNESS MATRIX STORED AS A ONE        I
C I                   DIMENSIONAL ARRAY                              I
C I    R(I)        =  LOAD VECTOR                                    I
C I    IDOF(I)     =  VECTOR CONTAINING THE D.O.F. NUMBERS OF JOINTS I
C I    JDIAG(I)    =  LOCATION OF THE DIAGNAL TERMS OF EACH COLUMN   I
C I                   IN THE GLOBAL STIFFNESS MATRIX 'SKG'           I
C I    NTSK        =  TOTAL NUMBER OF TERMS IN THE 'SKG' MATRIX      I
C I    NTDF        =  NUMBER OF TOTAL D.O.F. IN THE PROBLEM          I
C I                   NOT INCLUDING THE CONSTRAINED BOUNDARIES       I
C I    IOUT        =  OUTPUT DEVICE                                  I
C I    MBAND       =  HALF BAND WIDTH OF THE STIFFNESS MATRIX        I
C I                                                                  I
C I                                                                  I
C I    C O M M O N   B L O C K S                                     I
C I                                                                  I
C I    REFFER TO THE COMMON BLOCK DISCRIPTIONS.                      I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8(A-H,O-Z)
      LOGICAL YES,NO
      COMMON/TRANS/DC(3,3)
      COMMON/INPUTE/ISPB(4000)
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/MAIN4/RE(8000)
      COMMON/INPUT7/RIT(8000),RINC(8000),UINC(8000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1             NINODE
      COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/CONTR1/INCREM,NIT
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      DIMENSION R(1),SKG(1),SKGL(1),IDOF(1),JDIAG(1),DUMMY(3)
C
      YES = .TRUE.
```

```
      NO  = .FALSE.
C
C        MDF       = MAXIMUM DEGREES OF FREEDOM INCLUDING THE SUPPORTS
C
      MDF = NNODES*NNDF
      IF (IOINTR.EQ.0) IOINTR = NLINC
C
C        IF THIS RUN IS A RESTART THEN RESTORE THE LAST CONVERGED VALUES
C        OF THE EQUILIBRIUM LOAD VECTOR AND THE TOTAL DISPLACEMENT VECTOR
C
      IF(IFLAG3.EQ.1) THEN
          CALL RECOV(MDF,ISTART,NTDF,IDOF)
          CALL DIAGNL(NELEM,NNDF,NTDF,IDOF,JDIAG,NTSK,MBAND,IFLAG2,
     1    IOUT)
          IFINAL = ISTART + NLINC
          ISTART = ISTART + 1
          ISAVE = IFLAG1
      ELSE
          ISTART = 1
          IFINAL = NLINC
C
C        FOR THE FIRST ITERATION OF THE FIRST INCREMENT USE THE
C        GEOMETRIC LINEARITY ROUTINS.
C
C        IFLAG1 = 0; GEOMETRIC LINARITY
C               = 1; GEOMETRIC NON-LINEARITY
C        ISAVE  = DUMMY VARIABLE USED TO STORE THE VALUE OF 'IFLAG1'
C
          ISAVE = IFLAG1
          IFLAG1 = 0
      END IF
C
      IF (IFPLOT.GT.0) THEN
           IF (NINODE.GT.0) CALL CURVE
           CALL PLOTER(SKG,NNODES,NELEM,NNDF,IDIM,NINODE,IFLAG1,IOUT)
      END IF
C
C        CALCULATE THE PROPER LOAD OR DISPLACEMENT INCREMENT
C
C        UINC( K ) = APPLIED INCREMENT OF DISPLACEMENT
C        U( K )    = TOTAL APPLIED DISPLACEMENTS
C        R( K )    = TOTAL APPLIED LOADS
C        RINC( K ) = INCREMENT OF APPLIED LOADS
C        RE( K )   = EQUILIBRIUM LOAD VECTOR
C        NLINC     = NUMBER OF LOAD INCREMENTS
C
C
      DLINC = DFLOAT( NLINC )
C*VDIR: PREFER VECTOR
      DO 100 K1 = 1 , MDF
      UINC( K1 ) = U( K1 )/DLINC
  100 RINC( K1 ) = R( K1 )/DLINC
```

```
C
C       ICOUNT = ITERATION COUNT FOR THE RUN
C       IOCNT  = INCREMENT COUNT FROM THE THE START OR SINCE THE LAST
C                OUTPUT. WHEN 'IOCNT' IS EQUAL TO 'IOINTR' A COMPLETE
C                OUTPUT WILL BE GENERATED.
C
      ICOUNT = 0
      IOCNT  = 0
      IPLCNT = 0
C
C                     S T A R T     O F
C              I N C R E M E N T    L O O P
C
      DO 700 INCREM = ISTART , IFINAL
      IOCNT  = IOCNT + 1
      IPLCNT = IPLCNT + 1
      RFACT  = 1.0D0
      RFSUM  = 0.0D0
      NSUB   = 0
      FAC    = FACLOW
      FACNEW = FACHIG
C
C       FAC       = CONVERGANCE FACTOR
C       FACLOW    = LOWEST ALLOWABLE CONVERGENCE FACTOR
C       FACHIG    = LARGEST ALLOWABLE CONVERGENCE FACTOR
C       FACNEW    = NEW CONVERGANCE FACTOR CALCULATED BY ROUTINE 'CHECK'
C       RFACT     = REDUCTION FACTOR FOR SUBINCREMENTATION
C       U( K )    = INCREMENT OF THE APPLIED DISPLACEMENTS USED
C                      FOR THE FIRST ITERATION
C       RIT( K ) = TOTAL APPLIED LOAD AT THE END OF THE INCREMENT
C
C
C       ADJUST THE BOUNDARY CONDITIONS FOR THE INTERFACE NODES
C
      IF (NINODE.GT.0) THEN
           CALL BOUND(IDOF,NNDF,NINODE,ICODE,RFACT,IOUT)
           IF (ICODE.EQ.1) THEN
                CALL GLOB2(NNODES,NNDF,NTDF,IDOF)
                CALL DIAGNL(NELEM,NNDF,NTDF,IDOF,JDIAG,NTSK,MBAND,
     1                            IFLAG2,IOUT)
                ICODE1 = 1
           END IF
      END IF
C150  ICODE1    = 0
C
C*VDIR: PREFER VECTOR
 150  DO 200 K1 = 1 , MDF
      U( K1 )   = RFACT*UINC( K1 )
      RIT( K1 ) = RFACT*RINC( K1 ) + RE( K1 )
 200  CONTINUE
C
C                     S T A R T     O F
```

```
C             I T E R A T I O N     L O O P
C
      DO 580 NIT = 1 , MNIT
C
C       NIT =  ITERATION NUMBER
C       MNIT = MAXIMUM NUMBER OF ITERATIONS ALLOWED
C
      DO 450 K1 = 1 , NNODES
      I = NNDF*(K1 - 1)
      ICODE = ISPB( K1 )
C*VDIR: PREFER SCALAR
      DO 410 K2 = 1 , NNDF
      IDIR = I + K2
  410 DUMMY( K2 ) = RIT( IDIR ) - RE( IDIR )
      IF (ICODE.GT.0) THEN
           CALL DIRCOS(ICODE,IDIM)
C*VDIR: PREFER SCALAR
           DO 430 K2 = 1 , IDIM
           CST = 0.
C*VDIR: PREFER SCALAR
           DO 420 K3 = 1 , IDIM
           IDIR = I + K3
  420      CST = CST + (RIT( IDIR ) - RE( IDIR ))*DC(K3 , K2)
  430      DUMMY( K2 ) = CST
      END IF
C*VDIR: PREFER SCALAR
      DO 450 K2 = 1 , NNDF
      IDIR = I + K2
      ID = IDOF( IDIR )
  450 IF(ID.GT.0) R( ID ) = DUMMY( K2 )
C
      IF (NIT.EQ.1) THEN
           LDEV = LDEV1
      ELSE
           LDEV = LDEV2
      END IF
C
      CALL ASSEMB(SKG,SKGL,R,U,IDOF,JDIAG,NTSK,MBAND,IOUT)
C
      CALL REWIN
C
      IF (IFLAG2.EQ.0) THEN
           CALL SOLVE2(SKG,R,JDIAG,NTDF,1,IOUT)
           CALL SOLVE2(SKG,R,JDIAG,NTDF,2,IOUT)
      ELSE IF(IFLAG2.EQ.1) THEN
           CALL SOLVE1(SKG,SKGL,R,JDIAG,NTDF,YES,YES)
      END IF
C
C*VDIR: PREFER SCALAR
C
      DO 500 K1 = 1 , MDF
      ID = IDOF( K1 )
```

```
  500 IF(ID.GT.0) U( K1 ) = U( K1 ) + R( ID )
C
      DO 550 K1 = 1 , NNODES
      I = NNDF*(K1 - 1)
      ICODE = ISPB( K1 )
C*VDIR: PREFER SCALAR
      DO 510 K2 = 1 , NNDF
      IDIR = I + K2
  510 DUMMY( K2 ) = U( IDIR )
      IF (ICODE.GT.0) THEN
           CALL DIRCOS(ICODE,IDIM)
C*VDIR: PREFER SCALAR
           DO 530 K2 = 1 , IDIM
           CST = 0.
C*VDIR: PREFER SCALAR
           DO 520 K3 = 1 , IDIM
           IDIR = I + K3
  520      CST = CST + DC(K2 , K3)*U( IDIR )
  530      DUMMY( K2 ) = CST
      END IF
C*VDIR: PREFER SCALAR
      DO 550 K2 = 1 , NNDF
      IDIR = I + K2
      UTOTAL( IDIR ) = UTOTAL( IDIR ) + DUMMY( K2 )
  550 U( IDIR ) = DUMMY( K2 )
C
      IFLAG1 = ISAVE
C*VDIR: PREFER VECTOR
      DO 560 K1 = 1 , MDF
      RE1( K1 ) = RE( K1 )
 560  RE( K1 ) = 0.
C
      CALL GETSTR(IOUT)
      CALL CHECK(MDF,ITEST,IOUT)
C
C*VDIR: PREFER VECTOR
      DO 570 K1 = 1 , MDF
 570  U( K1 ) = 0.
C
      CALL REWIN
C
      IF(ITEST.EQ.1) THEN
          GO TO 600
      ELSE IF (ITEST.EQ.2) THEN
          ICOND = 2
          GO TO 590
      END IF
  580 CONTINUE
C
C                        E N D        O F
C              I T E R A T I O N      L O O P
C
```

```
      ICOND = 1
 590  IF (MNIT.NE.1) THEN
           IF (ICOND.EQ.1.AND.FACNEW.LT.FACHIG) THEN
                WRITE(IOUT , 1005)INCREM,FACNEW
                GO TO 600
           ELSE IF (ICOND.EQ.1.AND.RFACT.GT.0.0625) THEN
                RFACT = 0.5D0*RFACT
                WRITE(IOUT , 1009)INCREM,FACNEW,RFACT
                IF (NSUB.EQ.0) THEN
                     CALL RESTOR(MDF,LAST,IIII,IDOF)
                ELSE IF(NSUB.EQ.1) THEN
                     CALL RESTR1(MDF,IIII,IDOF)
                END IF
                GO TO 150
           ELSE IF (ICOND.EQ.2.AND.RFACT.GT.0.0625) THEN
                RFACT = 0.5D0*RFACT
                WRITE(IOUT , 1006)INCREM,RFACT
                IF (NSUB.EQ.0) THEN
                     CALL RESTOR(MDF,LAST,IIII,IDOF)
                ELSE IF(NSUB.EQ.1) THEN
                     CALL RESTR1(MDF,IIII,IDOF)
                END IF
C               IF (ICODE1.EQ.1) THEN
C                    CALL DIAGNL(NELEM,NNDF,NTDF,IDOF,JDIAG,NTSK,MBAND,
C    1                           IFLAG2,IOUT)
C               END IF
                GO TO 150
           END IF
           IF (ICOND.EQ.1) WRITE(IOUT , 1007) INCREM
           IF (ICOND.EQ.2) WRITE(IOUT , 1008) INCREM
           WRITE(IOUT , 1003) INCREM-1
           CALL RESTOR(MDF,LAST,NTDF,IDOF)
           CALL OUTPUT(IOUT,IERROR)
           CALL REWIN
           IF (IFPLOT.GT.0) THEN
              IF (NINODE.GT.0) CALL CURVE
              CALL PLOTER(SKG,NNODES,NELEM,NNDF,IDIM,NINODE,IFLAG1,IOUT)
           END IF
           GO TO 800
      END IF
C
 600  RFSUM = RFSUM + RFACT
      ICOUNT = ICOUNT + NIT
      IF (RFSUM.LT.0.999999999) THEN
           IF (NSUB.EQ.0) THEN
                CALL SWAP1
           ELSE
                CALL SWAP
           END IF
           NSUB = 1
           CALL STORE1(MDF,NTDF,IDOF)
           GO TO 150
```

```
      ELSE IF(NSUB.EQ.1) THEN
           CALL SWAP2
      ELSE IF(NSUB.EQ.0) THEN
           CALL SWAP
      END IF
C
      CALL STORE(MDF,INCREM,NTDF,IDOF)
      IF (IOCNT.EQ.IOINTR) THEN
         WRITE(IOUT , 1004) INCREM
         CALL OUTPUT(IOUT,IERROR)
         CALL REWIN
         IOCNT = 0
      END IF
      IF (IPLCNT.EQ.IFPLOT) THEN
         IF (NINODE.GT.0) CALL CURVE
         CALL PLOTER(SKG,NNODES,NELEM,NNDF,IDIM,NINODE,IFLAG1,IOUT)
         IPLCNT = 0
      END IF

C
 700  CONTINUE
  800 CALL ARCHIV(MDF)
      WRITE(IOUT , 1002) ICOUNT
      RETURN
 1002 FORMAT(///1X,'>>>>>>> TOTAL NUMBER OF ITERATIONS FOR THIS RUN IS'
     1 ,' = ',I5)
 1003 FORMAT(//1X,
     1 '>>>>>>> OUTPUTS ARE FOR THE LAST CONVERGED INCREMENT ',I4)
 1004 FORMAT(////////1X,'>>>>>>> OUTPUTS AT INCREMENT ',I4)
 1005 FORMAT(/1X,'>>>>>>> ALLOWABLE NUMBER OF ITERATIONS EXCEEDED AT',
     1 ' LOAD STEP ',I4/9X,'THE EFFECTIVE CONVERGENCE FACTOR ',E10.3,
     2 ' IS WITHIN TOLERANCE '/9X,'EXECUTION CONTINUES')
 1006 FORMAT(/1X,'>>>>>>> ALLOWABLE NUMBER OF DIVERGING ITERATIONS',
     1 ' IS EXCEEDED AT LOAD STEP ',I4/
     2 9X,'LOAD STEP FACTOR IS REDUCED TO ',F6.4/
     3 9X,'EXECUTION CONTINUES')
 1007 FORMAT(/1X,'>>>>>>> ALLOWABLE NUMBER OF ITERATIONS IS EXCEEDED ',
     1 ' AT LOAD STEP ',I4/9X,'THE EFFECTIVE CONVERGENCE FACTOR',E10.3,
     2 ' EXCEEDS THE PRISCRIBED TOLERANCE'/
     3 9X,'SUBINCREMENTATION HAS FAILED TO CORRECT THE PROBLEM'/
     4 9X,'EXECUTION TERMINATED')
 1008 FORMAT(/1X,'>>>>>>> ALLOWABLE NUMBER OF DIVERGING ITERATIONS',
     1 ' EXCEEDED AT LOAD STEP ',I4/
     2 9X,'SUBINCREMENTATION HAS FAILED TO CORRECT THE PROBLEM'/
     3 9X,'EXECUTION TERMINATED')
 1009 FORMAT(/1X,'>>>>>>> ALLOWABLE NUMBER OF ITERATIONS EXCEEDED AT',
     1 ' LOAD STEP ',I4/9X,'THE EFFECTIVE CONVERGENCE FACTOR',E10.3,
     2 ' EXCEEDS THE PRISCRIBED TOLERANCE'/9X,'REDUCTION FACTOR IS ',
     3 'REDUCES TO ',F6.4/9X,'EXECUTION CONTINUES')
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================== C H E C K ================================
C
      SUBROUTINE CHECK(MDF,ITEST,IOUT)
      IMPLICIT REAL*8  (A-H,O-Z)
      COMMON/CONTR1/INCREM,NIT
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/MAIN4/RE(8000)
      COMMON/INPUT7/RIT(8000),RINC(8000),UINC(8000)
      COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
C
      ITEST = 0
      ENRG = 0.
C
C        CALCULATE THE INCREMENT OF THE INTERNAL ENEGRY DUE TO THE
C        OUT OF BALANCE FORCES. AND REINITIALIZE THE INCREMENTAL
C        DISPLACEMENT VECTOR 'U'.
C
C*VDIR: PREFER VECTOR
      DO 20 K = 1 , MDF
      ENRG = ENRG + U( K )*(RE( K ) - RE1( K ))
   20 CONTINUE
      WRITE(6,*)'ENRG=',ENRG,'ENRG1=',ENRG1
C
      IF (NIT.EQ.1.OR.ENRG1.EQ.0.) THEN
           ENRG1  = ENRG
           NDIVER = 0
      ELSE
           IF (ENRG.GT.ENRG1) THEN
                NDIVER = NDIVER + 1
                WRITE(IOUT , 100)INCREM,NIT
                IF (NDIVER.GE.ISTOP) THEN
                     ITEST = 2
                END IF
           ELSE IF(ENRG.LE.FAC*ENRG1) THEN
                ITEST = 1
                NDIVER = 0
           END IF
C     FACTMP = ENRG/ENRG1
C     IF (FACTMP.LT.FACNEW) FACNEW = FACTMP + FACLOW
      FACNEW = ENRG/ENRG1
      END IF
C
C      ITEST = 0; NOCONVERGANCE
C            = 1; CONVERGANCE
C            = 2; TERMINATE PROGRAM DUE TO EXCEEDING THE ALLOWED
C                 NUMBER OF DIVERGING ITERATIONS
C
      RETURN
 100  FORMAT(/1X,'>>>>>>> DIVERGANCE DETECTED',
     1 ' AT LOAD INCREMENT ',I4,' ITERATION NO. ',I4)
```

END

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================== L O A D ===================================
C
      SUBROUTINE LOAD (R,IERROR)
C
C ====================================================================
C I                                                                  I
C I    P R O G R A M                                                 I
C I                                                                  I
C I    LOAD ASSEMBLES THE LOAD VECTOR BY CONSIDERING THE             I
C I    EXTERNALY APPLIED LOADS AND THE GRAVITY LOADS WHICH ARE       I
C I    SUPPERIMPOSED ON THE STRUCTER .                               I
C I                                                                  I
C I                                                                  I
C I    A R G U M E N T   L I S T                                     I
C I                                                                  I
C I      R(I)       =  LOAD VECTOR TO BE ASSEMBLED                   I
C I      IERROR     =  ERROR CODE =0; NO ERROR  0<; ERROR            I
C I                                                                  I
C I                                                                  I
C I    C O M M O N   B L O C K S                                     I
C I                                                                  I
C I    REFFER TO THE COMMON BLOCK DISCRIPTIONS.                      I
C I      N(I,J)     =  SHAPE FUNCTION FOR NODE I AT INTEGR. POINT J  I
C I      W(I)       =  GAUSSIAN WEIGTHING FUNCTIONS                  I
C I      XGAUSS     =  X COORDINATE OF THE GAUSSIAN POINTS IN THE ELEM.I
C I      WGTX(I)    =  SPECIFIC WEIGTH OF MATERIAL I IN THE X DIR.   I
C I      WGTY(I)    =  SPECIFIC WEIGTH OF MATERIAL I IN THE Y DIR.   I
C I      WGTZ(I)    =  SPECIFIC WEIGTH OF MATERIAL I IN THE Z DIR.   I
C I      THICK      =  THICKNESS OF THE ELEMENTS FOR PLANE STR & STN I
C I                 =  2*PI*XGAUSS FOR AXISYMETRIC PROBLEMS          I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 N,NXI,NETA,NSI
      INTEGER ELNUM
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,2000)
      COMMON/INPUT6/WGTX(10),WGTY(10),WGTZ(10)
      COMMON/INPUT7/RX(8000),RY(8000),RZ(8000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      COMMON/ISHAP2/W(27)
      DIMENSION R( 1 )
C
C ----- FIND THE CONTRIBUTION OF THE GRAVITY WEIGTHS FO 2D ELEMENTS
C
      ITYPE1 = 0
      IDENT1 = 0
```

```
      DO 30 ELNUM = 1 , NELEM
C
      CALL ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
      CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      WRITE(6,*)ITYPE,NNEL,ELNUM,NIPXI,NIPETA,IDENT
      IF (ITYPE.GT.300) THEN
           IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
                IF (INTCOD.GE.140) THEN
                     CALL ISH3DI(ITYPE,NNEL,IERROR)
                ELSE
                     CALL ISH3DG(ITYPE,NNEL,IERROR)
                END IF
           END IF
           IDENT1 = IDENT
           ITYPE1 = ITYPE
           DO 10 INTGPN = 1 , NIP
           CALL JACB3D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
           CST = DETJAC*W( INTGPN )
C*VDIR: IGNORE RECRDEPS
C*VDIR: PREFER VECTOR
           DO 10 K1 = 1 , NNEL
           M1 = NOP(K1 , ELNUM)
           RX( M1 ) = RX( M1 ) + N(K1 , INTGPN)*WGTX( MATNUM )*CST
           RY( M1 ) = RY( M1 ) + N(K1 , INTGPN)*WGTY( MATNUM )*CST
           RZ( M1 ) = RZ( M1 ) + N(K1 , INTGPN)*WGTZ( MATNUM )*CST
   10      CONTINUE
      ELSE
           IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
                CALL ISH2DG(ITYPE,NNEL,IERROR)
           END IF
           IDENT1 = IDENT
           ITYPE1 = ITYPE
           DO 20 INTGPN = 1 , NIP
           CALL JACB2D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
           IF (IFLAG.EQ.3) CALL AXISYM(INTGPN,ELNUM,NNEL,RAD,THICK)
           CST = DETJAC*THICK*W( INTGPN )
C*VDIR: IGNORE RECRDEPS
C*VDIR: PREFER VECTOR
           DO 20 K1 = 1 , NNEL
           M1 = NOP(K1 , ELNUM)
           RX( M1 ) = RX( M1 ) + N(K1 , INTGPN)*WGTX( MATNUM )*CST
           RY( M1 ) = RY( M1 ) + N(K1 , INTGPN)*WGTY( MATNUM )*CST
   20      CONTINUE
      END IF
   30 CONTINUE
C
C --- PLACE RX' S AND RY' S IN THE RIGHT POSITIONS IN THE
C --- LOAD ARRAY.
C
      IF (IDIM.EQ.2) THEN
C*VDIR: PREFER VECTOR
           DO 40 K = 1 , NNODES
```

```
          K2 = 2*K
          K1 = K2 - 1
          R( K1 )=RX( K )
          R( K2 )=RY( K )
 40       CONTINUE
C
      ELSE IF(IDIM.EQ.3) THEN
C*VDIR: PREFER VECTOR
          DO 50 K = 1 , NNODES
          K3 = 3*K
          K2 = K3 - 1
          K1 = K3 - 2
          R( K1 )=RX( K )
          R( K2 )=RY( K )
          R( K3 )=RZ( K )
 50       CONTINUE
      END IF
 1000 RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ====================== A S S E M B ==============================
C
      SUBROUTINE ASSEMB(SKG,SKGL,R,U,IDOF,JDIAG,NTSK,MBAND,IOUT)
C
C ==================================================================
C I                                                                I
C I   P R O G R A M                                                I
C I                                                                I
C I   SUBROUTINE ASSEMB ASSEMBELS THE GLOBAL STIFFNESS MATRIX AND/OR I
C I   STORES THE NODE NUMBERS OF THE CORENT ELEMENT AND THE POSITION I
C I   OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.              I
C I                                                                I
C I                                                                I
C I   A R G U M E N T    L I S T                                   I
C I                                                                I
C I   SKG(I)    =  GLOBAL STIFFNESS MATRIX STORED IN A ONE DIMENSIONAL I
C I                ARRAY USING THE SKYLINE METHOD                  I
C I   R(I)      =  LOAD VECTOR                                     I
C I   U(I)      =  VECTOR OF THE IMPOSED NODAL DISPLACEMENTS       I
C I   IDOF(I)   =  VECTOR CONTAINING THE D.O.F. NUMBERS THE JOINTS I
C I   JDIAG(I) =   LOCATION OF THE DIAGNAL TERMS OF EACH COLUMN IN THE I
C I                GLOBAL STIFFNESS MATRIX 'SKG'                   I
C I   NTSK      =  NUMBER OF TERMS IN THE SKG MATRIX               I
C I   IFLAG2    =  0; SYMMETRIC STIFFNESS MATRIX                   I
C I                1; NONSYMMETRIC STIFFNESS MATRIX                I
C I   IERROR    =  ERROR CODE     >0 ; ERRORS DETECTED             I
C I                               =0 ; NO ERRORS                   I
C I                                                                I
C I      FIRST DEVELOPED: 08-26-1988                               I
C I      LAST UPDATE:  09-05-1988                                  I
C I      BY:  M. FOROOZESH                                         I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER ELNUM
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,2000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE
      COMMON/INPUT9/THICK,IFLAG
      COMMON/ELST1/SK(60,60)
      COMMON/ASSEM2/II(60)
      DIMENSION IDOF( 1 ),JDIAG( 1 ),SKG( 1 ),SKGL( 1 ),R( 1 ),U( 1 )
C
C        INITIALIZE THE GLOBAL STIFFNESS MATIX TO ZERO
C
      IF (IFLAG2.EQ.0) THEN
C*VDIR: PREFER VECTOR
           DO 10 K1 = 1 , NTSK
   10      SKG( K1 )=0.
```

```
      ELSE
C*VDIR: PREFER VECTOR
           DO 20 K1 = 1 , NTSK/2
           SKG( K1 ) = 0.
   20      SKGL( K1 ) = 0.
      END IF
C
C
C           NCB = NUMBER OF COLUMNS IN THE <B> MATRIX.
C           NRB = NUMBER OF ROWS IN THE <B> MATRIX.
C           NNEL = NUMBER OF NODES IN THE ELEMENT.
C           MBAN = FULL BANDWIDTH OF THE STIFFNESS MATRIX
C
      MBAN = MBAND*2 - 1
      ITYPE1 = 0
      IDENT1 = 0
      DO 80 ELNUM = 1 , NELEM
C
      CALL ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
      CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
           IF (ITYPE.GT.300) THEN
                NCB = 3*NNEL
                NRB = 6
                IF (INTCOD.GE.140) THEN
                    CALL ISH3DI(ITYPE,NNEL,IERROR)
                ELSE
                    CALL ISH3DG(ITYPE,NNEL,IERROR)
                END IF
           ELSE
                NCB = 2*NNEL
                CALL ISH2DG(ITYPE,NNEL,IERROR)
                IF (IFLAG.EQ.3) THEN
                     NRB = 4
                ELSE
                     NRB = 3
                END IF
           END IF
      END IF
      IDENT1 = IDENT
      ITYPE1 = ITYPE
C
C*VDIR: PREFER SCALAR
      DO 30 K1 = 1 , NNEL
      I1 = NNDF*(K1 - 1)
      I2 = NNDF*(NOP(K1 , ELNUM) - 1)
C*VDIR: PREFER SCALAR
      DO 30 K2 = 1 , NNDF
      K = I1 + K2
      II( K ) = I2 + K2
   30 CONTINUE
C
```

```
C              GEOMETRICALLY NONLINEAR PROBLEMS
C
      IF (IFLAG1.EQ.1) THEN
       IF (ITYPE.GT.300) THEN
       CALL ES3DNS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
       ELSE
       CALL ES2DNS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
       END IF
C
C              GEOMETRICALLY LINEAR PROBLEMS
C
      ELSE IF(IFLAG1.EQ.0) THEN
       IF (ITYPE.GT.300) THEN
       CALL ES3DLS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
       ELSE
       CALL ES2DLS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
       END IF
      END IF
C
      DO 70 K1 = 1 , NCB
      NDOF = IDOF(II( K1 ))
      IF (NDOF.GT.0) THEN
          LOCD = JDIAG( NDOF )
          DO 50 K2 = 1 , NCB
          JDOF = IDOF(II( K2 ))
          IF (IFLAG2.EQ.0) THEN
              IF (NDOF.GE.JDOF.AND.JDOF.GT.0) THEN
                   LOCA = LOCD + NDOF - JDOF
                   SKG( LOCA ) = SKG( LOCA ) + SK(K1 , K2)
              END IF
          ELSE
              IF (NDOF.GE.JDOF.AND.JDOF.GT.0) THEN
                   LOCA = LOCD - NDOF + JDOF
                   SKG( LOCA ) = SKG( LOCA ) + SK(K2 , K1)
                   SKGL( LOCA ) = SKGL( LOCA ) + SK(K1 , K2)
              END IF
          END IF
   50     CONTINUE
      ELSE IF(NDOF.LT.0) THEN
          DO 60 K2 = 1 , NCB
          JDOF = IDOF(II( K2 ))
          IF (JDOF.GT.0) THEN
              R( JDOF ) = R( JDOF ) - SK(K1 , K2)*U(II( K1 ))
          END IF
   60     CONTINUE
      END IF
   70 CONTINUE
   80 CONTINUE
C
  100 RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= E L S T I F =============================
C
      SUBROUTINE ELSTIF
C
C ====================================================================
C I                                                                  I
C I   P R O G R A M                                                  I
C I                                                                  I
C I   ELSTIF EVALUATES THE STIFFNESS MATIRIX OF EACH ELEM.           I
C I                                                                  I
C I   E N T R Y   P O I N T S                                        I
C I                                                                  I
C I   ES2DLS: FOR 2D PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC     I
C I           PROBLEMS WITHOUT GEOMETRIC NONLIARITY.                 I
C I                                                                  I
C I   ES2DNS: FOR 2D PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC     I
C I           PROBLEMS WITH GEOMETRIC NONLIEARITY.                   I
C I                                                                  I
C I   ES3DLS: FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY        I
C I                                                                  I
C I   ES3DNS: FOR 3D STRAIN FIELDS WITH GEOMETRIC NONLINEARITY       I
C I                                                                  I
C I                                                                  I
C I   A R G U M E N T   L I S T                                      I
C I                                                                  I
C I   ELNUM     = ELEMENT NUMBER                                     I
C I                                                                  I
C I   NNEL      = NUMBER OF NODES IN THE ELEMENT                     I
C I                                                                  I
C I   NRB       = NUMBER OF ROWS OF THE <B> MATRIX                   I
C I                                                                  I
C I   NCB       = NUMBER OF COLUMNS OF THE <B> MATRIX                I
C I                                                                  I
C I   NIP       = TOTAL NUMBER OF INTEGRATION POINTS IN THE ELEM.    I
C I                                                                  I
C I   MATNUM    = MATERIAL NUMBER FOR THE ELEMENT                    I
C I                                                                  I
C I   IERROR    = ERROR CODE                                         I
C I                                                                  I
C I   IFLAG     = 1: PLANE STRESS PROBLEM (ES2DLS ,ES2DNS ONLY)      I
C I               2: PLANE STRAIN PROBLEM (ES2DLS ,ES2DNS ONLY)      I
C I               3: AXISYMMETRIC PROBLEM (ES2DLS ,ES2DNS ONLY)      I
C I                                                                  I
C I      FIRST DEVELOPED: 09-01-1988                                 I
C I      LAST UPDATE:  09-02-1988                                    I
C I      BY:  M. FOROOZESH                                           I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
```

```
      REAL*8 N,NXI,NETA,NSI,NX,NY,NZ
      CHARACTER*48 CSTRES
      CHARACTER*153 DUMMY
      INTEGER ELNUM
      COMMON/UTIL1/STRESS(6),DUMMY
      COMMON/INPUT9/THICK,IFLAG
      COMMON/ISHAP2/W(27)
      COMMON/ELST1/SK(60,60)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      COMMON/JACOB1/NX(20),NY(20),NZ(20)
C
C
C
C
      ENTRY ES2DLS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
C
      CALL ZEROSK(NCB)
      DO 200 INTGPN = 1 , NIP
         CALL JACB2D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         IF(IFLAG.EQ.3) CALL AXISYM(INTGPN,ELNUM,NNEL,RAD,THICK)
         CALL B2DLS(INTGPN,NNEL,RAD)
         CST = DETJAC*THICK*W( INTGPN )
         CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,0)
         CALL BTDB(IFLAG2,NRB,NCB,CST)
  200 CONTINUE
      CALL SKTRAN(ELNUM,NNEL,NNDF,NCB,2)
C
      RETURN
C
C
C
C
C
C
      ENTRY ES2DNS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
C
      CALL ZEROSK(NCB)
      RAD = 1.
      DO 500 INTGPN = 1 , NIP
         CALL JACB2D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         IF(IFLAG.EQ.3) CALL AXISYM(INTGPN,ELNUM,NNEL,RAD,THICK)
         CALL B2DNS(INTGPN,NNEL,RAD)
         CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,0)
         CST = DETJAC*THICK*W( INTGPN )
         CALL BTDB(IFLAG2,NRB,NCB,CST)
C
      CALL IOGET(LDEV,48,'(A48)',5)
      SX = STRESS( 1 )
      SY = STRESS( 2 )
      SZ = STRESS( 4 )
      SXY = STRESS( 3 )
```

```
      IF(IFLAG.NE.3) SZ = 0.
C
C --- CALCULATION OF <G>TR <M><G>.
C
      CST1 = SZ*CST/RAD**2
C*VDIR: ASSUME COUNT(8)
      DO 400 K1 = 1 , NNEL
      K12 = 2*K1
      K11 = K12 - 1
      B1 = (NX(K1)*SX+NY(K1)*SXY)*CST
      B2 = (NX(K1)*SXY+NY(K1)*SY)*CST
      B3 = N(K1,INTGPN)*CST1
C
C*VDIR: ASSUME COUNT(8)
      DO 400 K2 = 1 , NNEL
      K22 = 2*K2
      K21 = K22 - 1
      B4 = NX(K2)*B1+NY(K2)*B2
      SK(K11,K21) = SK(K11,K21)+B4+N(K2,INTGPN)*B3
      SK(K12,K22) = SK(K12,K22)+B4
 400  CONTINUE
 500  CONTINUE
      CALL SKTRAN(ELNUM,NNEL,NNDF,NCB,2)
C
      RETURN
C
C
C
C
C
C
      ENTRY ES3DLS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
C
      CALL ZEROSK(NCB)
      DO 600 INTGPN = 1 , NIP
         CALL JACB3D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         CALL B3DLS(NNEL)
         CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,0)
         CST = DETJAC*W( INTGPN )
         CALL BTDB(IFLAG2,NRB,NCB,CST)
 600  CONTINUE
      CALL SKTRAN(ELNUM,NNEL,NNDF,NCB,3)
C
      RETURN
C
C
C
C
C
      ENTRY ES3DNS(ELNUM,ITYPE,NNEL,NNDF,NRB,NCB,NIP,MATNUM,IFLAG2,IOUT)
C
      CALL ZEROSK(NCB)
```

```
      DO 800 INTGPN = 1 , NIP
         CALL JACB3D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         CALL B3DNS(NNEL)
         CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,0)
         CST = DETJAC*W( INTGPN )
         CALL BTDB(IFLAG2,NRB,NCB,CST)
C
      CALL IOGET(LDEV,48,'(A48)',5)
      SX = STRESS( 1 )
      SY = STRESS( 2 )
      SZ = STRESS( 3 )
      SXY = STRESS( 4 )
      SYZ = STRESS( 5 )
      SXZ = STRESS( 6 )
C
C --- CALCULATION OF <G>TR <M><G>.
C
C*VDIR: ASSUME COUNT(20)
      DO 700 K1 = 1 , NNEL
      K13 = 3*K1
      K12 = K13 - 1
      K11 = K13 - 2
      B1 = (NX(K1)*SX + NY(K1)*SXY + NZ(K1)*SXZ)*CST
      B2 = (NX(K1)*SXY + NY(K1)*SY + NZ(K1)*SYZ)*CST
      B3 = (NX(K1)*SXZ + NY(K1)*SYZ + NZ(K1)*SZ)*CST
C
C*VDIR: ASSUME COUNT(20)
      DO 700 K2 = 1 , NNEL
      K23 = 3*K2
      K22 = K23 - 1
      K21 = K23 - 2
      B4 = NX(K2)*B1 + NY(K2)*B2 + NZ(K2)*B3
      SK(K11,K21) = SK(K11,K21) + B4
      SK(K12,K22) = SK(K12,K22) + B4
      SK(K13,K23) = SK(K13,K23) + B4
 700  CONTINUE
 800  CONTINUE
      CALL SKTRAN(ELNUM,NNEL,NNDF,NCB,3)
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================= Z E R O S K =================================
C
      SUBROUTINE ZEROSK(N)
      REAL*8 SK
      COMMON/ELST1/SK(60,60)
C
C*VDIR: PREFER VECTOR
      DO 100 K1 = 1 , N
      DO 100 K2 = 1 , N
 100  SK(K1 , K2) = 0.
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================== S K T R A N ===============================
C
      SUBROUTINE SKTRAN(ELNUM,NNEL,NNDF,NCB,IDIM)
C
C ====================================================================
C I                                                                  I
C I   P R O G R A M                                                  I
C I                                                                  I
C I   SKTRAN MODIFIES THE ELEMENT STIFFNESS MATRIX FOR THE SKEW      I
C I   BOUNDARY CONDITIONS USING <T>T<SK><T> TRANSFORMATION.          I
C I                                                                  I
C I   A R G U M E N T    L I S T                                     I
C I                                                                  I
C I   ELNUM     = ELEMENT NUMBER                                     I
C I                                                                  I
C I   NNEL      = NUMBER OF NODES IN THE ELEMENT                     I
C I                                                                  I
C I   NNDF      = NUMBER OF NODAL DEGREES OF FREEDOM                 I
C I                                                                  I
C I   NCB       = NUMBER OF COLUMNS OF THE <B> MATRIX                I
C I                                                                  I
C I   IDIM      = PHYSICAL DIMENSION OF THE PROBLEM (I.E.,2D OR 3D)  I
C I                                                                  I
C I      FIRST DEVELOPED: 10-18-1988                                 I
C I      LAST UPDATE:  10-18-1988                                    I
C I      BY:  M. FOROOZESH                                           I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER ELNUM
      COMMON/ELST1/SK(60,60)
      COMMON/TRANS/DC(3,3)
      COMMON/INPUTE/ISPB(4000)
      COMMON/INPUT2/NOP(20,2000)
      DIMENSION CST(60,3)
C
      DO 600 K1 = 1 , NNEL
      NODE = NOP(K1 , ELNUM)
      ICODE = ISPB(NODE)
      IF (ICODE.GT.0) THEN
          I = NNDF*(K1 - 1)
          CALL DIRCOS(ICODE,IDIM)
C*VDIR: PREFER VECTOR
          DO 300 K2 = 1 , NCB
C*VDIR: PREFER SCALAR
          DO 200 K3 = 1 , IDIM
          CST(K2 , K3) = 0.
C*VDIR: PREFER SCALAR
          DO 200 IDIR = 1 , IDIM
```

```
         ID = I + IDIR
 200     CST(K2 , K3) = CST(K2 , K3) + SK(K2 , ID)*DC(IDIR , K3)
C*VDIR: PREFER SCALAR
         DO 300 K3 = 1 , IDIM
         ID = I + K3
 300     SK(K2 , ID) = CST(K2 , K3)
C
C*VDIR: PREFER VECTOR
         DO 500 K2 = 1 , NCB
C*VDIR: PREFER SCALAR
         DO 400 K3 = 1 , IDIM
         CST(K2 , K3) = 0.
C*VDIR: PREFER SCALAR
         DO 400 IDIR = 1 , IDIM
         ID = I + IDIR
 400     CST(K2 , K3) = CST(K2 , K3) + SK(ID , K2)*DC(IDIR , K3)
C*VDIR: PREFER SCALAR
         DO 500 K3 = 1 , IDIM
         ID = I + K3
 500     SK(ID , K2) = CST(K2 ,  K3)
      END IF
 600  CONTINUE
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= D I R C O S ===============================
C
      SUBROUTINE DIRCOS(ICODE,IDIM)
C
C ===================================================================
C I                                                                 I
C I   P R O G R A M                                                 I
C I                                                                 I
C I   DIRCOS EVALUATES THE DIRECTION COSINES OF THE Y_PRIM AND      I
C I   THE Z_PRIM AXES FOR SKEW BOUNDARY CONDITIONS USING THE        I
C I   DIRECTION COSINES OF THE X_PRIM (WHICH IS THE AXIS NORMAL TO  I
C I   THE PLANE OF THE ROLLER.                                      I
C I                                                                 I
C I                                                                 I
C I   A R G U M E N T    L I S T                                    I
C I                                                                 I
C I   ICODE     = ADDRESS OF THE DIRECTION COSINES OF THE X_PRIM    I
C I                                                                 I
C I   IDIM      = PHYSICAL DIMENSION OF THE PROBLEM (I.E.,2D OR 3D) I
C I                                                                 I
C I                                                                 I
C I   C O M M O N    B L O C K S                                    I
C I                                                                 I
C I   COSTX(ICODE)  =  COSINE OF THETA_X                            I
C I                                                                 I
C I   COSTY(ICODE)  =  COSINE OF THETA_Y                            I
C I                                                                 I
C I   COSTZ(ICODE)  =  COSINE OF THETA_Z                            I
C I                                                                 I
C I   DC(I,J)       =  TRANSFORMATION MATRIX WHICH HAS ITS COLUMNS  I
C I                    EQUAL TO THE DIRECTION COSINES OF THE        I
C I                    X_PRIM, Y_PRIM, AND Z_PRIM AXES.             I
C I                                                                 I
C I      FIRST DEVELOPED: 10-18-1988                                I
C I      LAST UPDATE:  10-19-1988                                   I
C I      BY:  M. FOROOZESH                                          I
C I                                                                 I
C ===================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/INPUTD/COSTX(300),COSTY(300),COSTZ(300)
      COMMON/TRANS/DC(3,3)
C
      IF(IDIM.EQ.2) THEN
            DC(1 , 1) = COSTX( ICODE )
            DC(2 , 1) = COSTY( ICODE )
            DC(1 , 2) = -COSTY( ICODE )
            DC(2 , 2) = COSTX( ICODE )
      ELSE IF(IDIM.EQ.3) THEN
            TX = COSTX( ICODE )
```

```
          TY = COSTY( ICODE )
          TZ = COSTZ( ICODE )
          DC(1 , 1) = TX
          DC(2 , 1) = TY
          DC(3 , 1) = TZ
C
          CNORM = DSQRT(TY**2 + TX**2)
          DC(1 , 2) = -TY/CNORM
          DC(2 , 2) = TX/CNORM
          DC(3 , 2) = 0.
C
          CNORM = DSQRT((TX/TY)**2 + 1.+ (TX**2/(TY*TZ)+TY/TZ)**2)
          DC(1 , 3) = TX/(TY*CNORM)
          DC(2 , 3) = 1./CNORM
          DC(3 , 3) = -(TX**2/(TY*TZ) + TY/TZ)/CNORM
      END IF
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C =========================== B T D B ==================================
C
      SUBROUTINE BTDB(IFLAG2,NRB,NCB,CST)
C
C =====================================================================
C I                                                                   I
C I   SUBPROGRAM BTDB EVALUATES <BT><DEP><B>CST, WHERE                I
C I                                                                   I
C I      <BT> = TRANSPOSE OF THE <B> MATRIX                           I
C I      <DEP> = MATERIAL STIFFNESS MATRIX                            I
C I      CST   = CONSTANT VALUE TO BE MULT. WITH EACH TERM OF THE     I
C I              RESULTING MATRIX.                                    I
C I                                                                   I
C I   A R G U M E N T     L I S T                                     I
C I                                                                   I
C I      IFLAG2 = 0; FOR SYMMETRIC STIFFNESS MATRIX                   I
C I               1; FOR NONSYMMETRIC STIFFNESS MATRIX                I
C I                                                                   I
C I      NRB   = NUMBER OF ROWS IN THE <B> BATRIX                     I
C I                                                                   I
C I      NCB   = NUMBER OF COLUMNS IN THE <B> MATRIX                  I
C I                                                                   I
C I      CST   = INTEGRATION CONSTANT                                 I
C I                                                                   I
C I                                                                   I
C I      FIRST DEVELOPED: 09-01-1988                                  I
C I      LAST UPDATE:  09-02-1988                                     I
C I      BY:  M. FOROOZESH                                            I
C I                                                                   I
C =====================================================================
C
      REAL*8 SK,B,DEP,CST,DUMMY,TEMP
      COMMON/ELST1/SK(60,60)
      COMMON/B1/B(6,60)
      COMMON/MATER1/DEP(6,6)
      DIMENSION DUMMY(60,6)
C
C --- B(K3,K1) IS THE TRANSPOSE OF THE B(K1,K3)
C
      DO 10 K1 = 1 , NRB
      DO 10 K2 = 1 , NRB
 10   DEP(K1 , K2) = DEP(K1 , K2)*CST
C
      IF (IFLAG2.EQ.0) THEN
C*VDIR: PREFER VECTOR
           DO 300 K1 = 1,NCB
           DO 100 K2 = 1,NRB
           DUMMY(K1 , K2) = 0.
           DO 100 K3 = 1,NRB
 100       DUMMY(K1 , K2) = DUMMY(K1 , K2) + B(K3 , K1)*DEP(K3 , K2)
```

```
            DO 300 K4 = 1,K1
            DO 200 K5 = 1,NRB
 200        SK(K1,K4) = SK(K1,K4) + DUMMY(K1 ,  K5)*B(K5 , K4)
 300        SK(K4,K1) = SK(K1,K4)
C
      ELSE IF(IFLAG2.EQ.1) THEN
C
C*VDIR: PREFER VECTOR
            DO 600 K1 = 1,NCB
            DO 400 K2 = 1,NRB
            DUMMY(K1 , K2) = 0.
            DO 400 K3 = 1,NRB
 400        DUMMY(K1 , K2) = DUMMY(K1 , K2) + B(K3 , K1)*DEP(K3 , K2)
            DO 600 K4 = 1,NCB
            DO 600 K5 = 1,NRB
 600        SK(K1,K4) = SK(K1,K4) + DUMMY(K1 , K5)*B(K5 , K4)
      END IF
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================== B 2 D 3 D ================================
C
      SUBROUTINE B2D3D
C
C ===================================================================
C I                                                                 I
C I   SUBPROGRAM B2D3D EVALUATES THE 'B' MATRIX FOR THE 2D AND 3D   I
C I   FINITE STRAIN PROBLEMS.                                       I
C I                                                                 I
C I      ENTRY POINTS:                                              I
C I      B2DLS : FOR 2D PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC I
C I               PROBLEMS WITHOUT GEOMETRIC NONLIARITY.            I
C I                                                                 I
C I      B2DNS: FOR 2D PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC  I
C I               PROBLEMS WITH GEOMETRIC NONLIEARITY.              I
C I                                                                 I
C I      B3DLS : FOR 3D STRAIN FIELDS WITHOUT GEOMETRIC NONLINEARITY I
C I                                                                 I
C I      B3DNS: FOR 3D STRAIN FIELDS WITH GEOMETRIC NONLINEARITY    I
C I                                                                 I
C I     B(I,J)      =  VARIATIONAL STRAIN-DISPLACEMENT STIFFNESS    I
C I                    MATRIX.                                      I
C I                                                                 I
C I                                                                 I
C I   NX(K) = PARTIAL DERIVATIVE OF N(K) WITH RESPECT TO X;         I
C I   NY(K) = PARTIAL DERIVATIVE OF N(K) WITH RESPECT TO Y;         I
C I   NZ(K) = PARTIAL DERIVATIVE OF N(K) WITH RESPECT TO Z;         I
C I                                                                 I
C I                                                                 I
C I      FIRST DEVELOPED: 08-29-1988                                I
C I      LAST UPDATE:  08-30-1988                                   I
C I      BY:  M. FOROOZESH                                          I
C I                                                                 I
C ===================================================================
C
      IMPLICIT REAL*8 (A-H,N-Z)
      INTEGER ELNUM,NNEL
      COMMON/INPUT9/THICK,IFLAG
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/ASSEM2/II(60)
      COMMON/JACOB1/NX(20),NY(20),NZ(20)
      COMMON/B1/B(6,60)
      COMMON/B2/DUDX,DVDX,DWDX,DUDY,DVDY,DWDY,DUDZ,DVDZ,DWDZ,A5
C
C
C
C
      ENTRY B2DLS(INTGPN,NNEL,RAD)
C
```

```
C --- CALCULATION OF THE <B> MATRIX
C
C*VDIR: ASSUME COUNT(8)
      DO 100 K1 = 1,NNEL
      K12 = 2*K1
      K11 = K12 - 1
      B(1,K11) = NX(K1)
      B(1,K12) = 0.
      B(2,K11) = 0.
      B(2,K12) = NY(K1)
      B(3,K11) = NY(K1)
      B(3,K12) = NX(K1)
  100 CONTINUE
C
C --- CALCULATION OF THE ADDITIONAL ROW OF <B> FOR THE AXISYM. CASE.
C
      IF (IFLAG.EQ.3) THEN
C
C*VDIR: ASSUME COUNT(8)
      DO 200 K1 = 1 , NNEL
      K12 = 2*K1
      K11 = K12 - 1
      B(4,K11) = N(K1,INTGPN)/RAD
  200 B(4,K12) = 0.
C
      END IF
C
      RETURN
C
C
C
C
      ENTRY B2DNS(INTGPN,NNEL,RAD)
C
C --- CALCULATION OF THE <B> MATRIX
C
      DUDX = 0.
      DVDX = 0.
      DUDY = 0.
      DVDY = 0.
      A5 = 0.
C*VDIR: ASSUME COUNT(8)
      DO 300 K1 = 1,NNEL
      K12 = 2*K1
      K11 = K12 - 1
      DUDX = DUDX + NX(K1)*UTOTAL(II(K11))
      DVDX = DVDX + NX(K1)*UTOTAL(II(K12))
      DUDY = DUDY + NY(K1)*UTOTAL(II(K11))
      DVDY = DVDY + NY(K1)*UTOTAL(II(K12))
      A5 = A5 + N(K1,INTGPN)*UTOTAL(II(K11))
  300 CONTINUE
C
```

```
C*VDIR: ASSUME COUNT(8)
      DO 400 K1 = 1,NNEL
      K12 = 2*K1
      K11 = K12 - 1
      B(1,K11) = (1. + DUDX)*NX(K1)
      B(1,K12) = DVDX*NX(K1)
      B(2,K11) = DUDY*NY(K1)
      B(2,K12) = (1. + DVDY)*NY(K1)
      B(3,K11) = DUDY*NX(K1) + (1. + DUDX)*NY(K1)
      B(3,K12) = DVDX*NY(K1) + (1. + DVDY)*NX(K1)
 400  CONTINUE
C
C --- CALCULATION OF THE ADDITIONAL ROW OF <B> FOR THE AXISYM CASE.
C
      IF (IFLAG.EQ.3) THEN
      A5 = A5/RAD
C
C*VDIR: ASSUME COUNT(8)
      DO 500 K1 = 1 , NNEL
      K12 = 2*K1
      K11 = K12 - 1
      B(4,K11) = (A5 + 1.)*N(K1,INTGPN)/RAD
 500  B(4,K12) = 0.
C
      END IF
C
      RETURN
C
C
C
C
      ENTRY B3DLS(NNEL)
C
C --- CALCULATION OF THE <B> MATRIX
C
C*VDIR: ASSUME COUNT(20)
      DO 600 K1 = 1,NNEL
      K13 = 3*K1
      K12 = K13 - 1
      K11 = K13 - 2
      B(1,K11) = NX(K1)
      B(1,K12) = 0.
      B(1,K13) = 0.
      B(2,K11) = 0.
      B(2,K12) = NY(K1)
      B(2,K13) = 0.
      B(3,K11) = 0.
      B(3,K12) = 0.
      B(3,K13) = NZ(K1)
      B(4,K11) = NY(K1)
      B(4,K12) = NX(K1)
      B(4,K13) = 0.
```

```
      B(5,K11) = 0.
      B(5,K12) = NZ(K1)
      B(5,K13) = NY(K1)
      B(6,K11) = NZ(K1)
      B(6,K12) = 0.
      B(6,K13) = NX(K1)
 600  CONTINUE
C
      RETURN
C
C
C
C
      ENTRY B3DNS(NNEL)
C
C --- CALCULATION OF THE <B> MATRIX
C
      DUDX = 0.
      DVDX = 0.
      DWDX = 0.
      DUDY = 0.
      DVDY = 0.
      DWDY = 0.
      DUDZ = 0.
      DVDZ = 0.
      DWDZ = 0.
C
C*VDIR: ASSUME COUNT(20)
      DO 700 K1 = 1,NNEL
      K13 = 3*K1
      K12 = K13 - 1
      K11 = K13 - 2
      DUDX = DUDX + NX(K1)*UTOTAL(II(K11))
      DVDX = DVDX + NX(K1)*UTOTAL(II(K12))
      DWDX = DWDX + NX(K1)*UTOTAL(II(K13))
      DUDY = DUDY + NY(K1)*UTOTAL(II(K11))
      DVDY = DVDY + NY(K1)*UTOTAL(II(K12))
      DWDY = DWDY + NY(K1)*UTOTAL(II(K13))
      DUDZ = DUDZ + NZ(K1)*UTOTAL(II(K11))
      DVDZ = DVDZ + NZ(K1)*UTOTAL(II(K12))
      DWDZ = DWDZ + NZ(K1)*UTOTAL(II(K13))
 700  CONTINUE
C
C*VDIR: ASSUME COUNT(20)
      DO 800 K1 = 1,NNEL
      K13 = 3*K1
      K12 = K13 - 1
      K11 = K13 - 2
      B(1,K11) = (1. + DUDX)*NX(K1)
      B(1,K12) = DVDX*NX(K1)
      B(1,K13) = DWDX*NX(K1)
      B(2,K11) = DUDY*NY(K1)
```

```
      B(2,K12) = (1. + DVDY)*NY(K1)
      B(2,K13) = DWDY*NY(K1)
      B(3,K11) = DUDZ*NZ(K1)
      B(3,K12) = DVDZ*NZ(K1)
      B(3,K13) = (1. + DWDZ)*NZ(K1)
      B(4,K11) = DUDY*NX(K1) + (1. + DUDX)*NY(K1)
      B(4,K12) = DVDX*NY(K1) + (1. + DVDY)*NX(K1)
      B(4,K13) = DWDY*NX(K1) + DWDX*NY(K1)
      B(5,K11) = DUDZ*NY(K1) + DUDY*NZ(K1)
      B(5,K12) = DVDZ*NY(K1) + (1. + DVDY)*NZ(K1)
      B(5,K13) = DWDY*NZ(K1) + (1. + DWDZ)*NY(K1)
      B(6,K11) = DUDZ*NX(K1) + (1. + DUDX)*NZ(K1)
      B(6,K12) = DVDX*NZ(K1) + DVDZ*NX(K1)
      B(6,K13) = DWDX*NZ(K1) + (1. + DWDZ)*NX(K1)
  800 CONTINUE
C
      RETURN
C
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ====================== A X I S Y M ==================================
C
      SUBROUTINE AXISYM(INTGPN,ELNUM,NNEL,RAD,THICK)
C
C ======================================================================
C I                                                                    I
C I      SUBPROGRAM AXISYM EVALUATES THE RADIUS OF THE INTEGRATION     I
C I      POINT FROM THE AXIS OF SYMMETRY OF AXISYMMETRIC PROBLEMS.     I
C I      THE AXIS OF SYMMETRY IS ASSUMED TO BE THE Y AXIS.             I
C I                                                                    I
C I         INTGPN  = INTEGRATION POINT NUMBER                         I
C I         ELNUM   = ELEMENT NUMBER                                   I
C I         NNEL    = NUMBER OF NODES IN THE ELEMENT                   I
C I         RAD     = RADIUS OF THE INTEGRATION POINT                  I
C I         THICK   = SURCOMFRENCE OF THE AXISYMMETRIC SOLID           I
C I                                                                    I
C I      FIRST DEVELOPED: 09-01-1988                                   I
C I      LAST UPDATE:  09-01-1988                                      I
C I      BY:  M. FOROOZESH                                             I
C ======================================================================
      REAL*8 RAD,THICK,N,NXI,NETA,NSI
      REAL*4 X,Y,Z
      INTEGER ELNUM
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      COMMON/INPUT2/NOP(20,2000)
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
C
      RAD = 0.
C*VDIR: ASSUME COUNT(8)
      DO 10 K1 = 1 , NNEL
      RAD = RAD + N(K1 , INTGPN)*X(NOP(K1 , ELNUM))
   10 CONTINUE
C
      THICK = 6.283185307179586D0*RAD
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= G E T S T R ===============================
C
      SUBROUTINE GETSTR(IOUT)
C
C ====================================================================
C I                                                                  I
C I   SUBROUTINE GETSTR ASSEMBELS THE GLOBAL STIFFNESS MATRIX AND/OR  I
C I   STORES THE NODE NUMBERS OF THE CORENT ELEMENT AND THE POSITION  I
C I   OF THE ELEMENT MATRICES IN THE GLOBAL MATRICES.                 I
C I                                                                  I
C I                                                                  I
C I   II(J)    POSITION OF LOCAL STIFFNESS TERMS IN THE GLOBAL        I
C I            STIFFNESS MATRIX.                                      I
C I                                                                  I
C I   SKG(I) =    GLOBAL STIFFNESS MATRIX IN THE CONDENSED FORM       I
C I   SK(I,J)  =    ELEMENT STIFFNESS MATRIX                          I
C I                 (SK IS COMPUTED BY SUBPROGRAM STIFEL)             I
C I                                                                  I
C I                                                                  I
C I      FIRST DEVELOPED: 08-26-1988                                  I
C I      LAST UPDATE:  09-05-1988                                     I
C I      BY:  M. FOROOZESH                                            I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER ELNUM
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1             NINODE
      COMMON/INPUT9/THICK,IFLAG
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,2000)
      COMMON/ASSEM2/II(60)
C
C           NCB = NUMBER OF COLUMNS IN THE <B> MATRIX.
C           NRB = NUMBER OF ROWS IN THE <B> MATRIX.
C           NNEL = NUMBER OF NODES IN THE ELEMENT.
C
      ITYPE1 = 0
      IDENT1 = 0
      DO 80 ELNUM = 1 , NELEM
C
      CALL ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
      CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
           IF (ITYPE.GT.300) THEN
                NCB = 3*NNEL
                NRB = 6
                IF (INTCOD.GE.140) THEN
                    CALL ISH3DI(ITYPE,NNEL,IERROR)
                ELSE
```

```
                CALL ISH3DG(ITYPE,NNEL,IERROR)
             END IF
         ELSE
             NCB = 2*NNEL
             CALL ISH2DG(ITYPE,NNEL,IERROR)
             IF (IFLAG.EQ.3) THEN
                 NRB = 4
             ELSE
                 NRB = 3
             END IF
         END IF
      END IF
      ITYPE1 = ITYPE
      IDENT1 = IDENT
C
C*VDIR: PREFER SCALAR
      DO 30 K1 = 1 , NNEL
      I1 = NNDF*(K1 - 1)
      I2 = NNDF*(NOP(K1 , ELNUM) - 1)
C
C*VDIR: PREFER SCALAR
      DO 30 K2 = 1 , NNDF
      K = I1 + K2
      II( K ) = I2 + K2
   30 CONTINUE
C
C              GEOMETRICALLY NONLINEAR PROBLEMS
C
      IF (IFLAG1.EQ.1) THEN
          IF(ITYPE.GT.300) THEN
             CALL S3DNS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
          ELSE
             CALL S2DNS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
          END IF
C
C              GEOMETRICALLY LINEAR PROBLEMS
C
      ELSE IF(IFLAG1.EQ.0) THEN
          IF(ITYPE.GT.300) THEN
             CALL S3DLS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
          ELSE
             CALL S2DLS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
          END IF
      END IF
   80 CONTINUE
C
  100 RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= E L S T R ===============================
C
      SUBROUTINE ELSTR
C
C ====================================================================
C I                                                                  I
C I   SUBPROGRAM ELSTR EVALUATES THE STIFFNESS MATIRIX OF EACH ELEM.  I
C I                                                                  I
C I      ENTRY POINTS:                                               I
C I      S2DLS   : FOR 2D PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC I
C I                PROBLEMS WITHOUT GEOMETRIC NONLIARITY.            I
C I                                                                  I
C I      S2DNS : FOR 2D PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC   I
C I                PROBLEMS WITH GEOMETRIC NONLIEARITY.              I
C I                                                                  I
C I      S3DLS   : FOR 3D STRAIN FIELDS WITHOUT GEOM. NONLINEARITY    I
C I                                                                  I
C I      S3DNS : FOR 3D STRAIN FIELDS WITH GEOMETRIC NONLINEARITY     I
C I                                                                  I
C I                                                                  I
C I  PARAMETER LIST:                                                 I
C I                                                                  I
C I      IFLAG      = 1: PLANE STRAIN PROBLEM (ES2DLS ,ES2DNS ONLY)   I
C I                   2: PLANE STRESS PROBLEM (ES2DLS ,ES2DNS ONLY)   I
C I                   3: AXISYMMETRIC PROBLEM (ES2DLS ,ES2DNS ONLY)   I
C I                                                                  I
C I      ELNUM      = ELEMENT NUMBER                                  I
C I      NNEL       = NUMBER OF NODES IN THE ELEMENT                  I
C I      NRB        = NUMBER OF ROWS OF THE <B> MATRIX                I
C I      NCB        = NUMBER OF COLUMNS OF THE <B> MATRIX             I
C I      NIP        = TOTAL NUMBER OF INTEGRATION POINTS IN THE ELEM. I
C I      IERROR     = ERROR CODE                                      I
C I                                                                  I
C I                                                                  I
C I      FIRST DEVELOPED: 09-01-1988                                  I
C I      LAST UPDATE:  09-02-1988                                     I
C I      BY:  M. FOROOZESH                                            I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 N,NXI,NETA,NSI,NX,NY,NZ
      INTEGER ELNUM
      COMMON/ISHAP2/W(27)
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/ASSEM2/II(60)
      COMMON/JACOB1/NX(20),NY(20),NZ(20)
      COMMON/B2/DUDX,DVDX,DWDX,DUDY,DVDY,DWDY,DUDZ,DVDZ,DWDZ,A5
      COMMON/INPUT9/THICK,IFLAG
```

```
      COMMON/ELSTR1/STRN(6)
C
C
C
C
      ENTRY S2DLS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
C
      DO 200 INTGPN = 1 , NIP
         CALL JACB2D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         IF(IFLAG.EQ.3) CALL AXISYM(INTGPN,ELNUM,NNEL,RAD,THICK)
         CALL B2DLS(INTGPN,NNEL,RAD)
         CST = DETJAC*THICK*W( INTGPN )
         DUDX = 0.
         DVDX = 0.
         DUDY = 0.
         DVDY = 0.
         A5 = 0.
C*VDIR: ASSUME COUNT(8)
                DO 100 K1 = 1,NNEL
                K12 = 2*K1
                K11 = K12 - 1
                DUDX = DUDX + NX(K1)*UTOTAL(II(K11))
                DVDX = DVDX + NX(K1)*UTOTAL(II(K12))
                DUDY = DUDY + NY(K1)*UTOTAL(II(K11))
                DVDY = DVDY + NY(K1)*UTOTAL(II(K12))
                A5 = A5 + N(K1,INTGPN)*UTOTAL(II(K11))
 100            CONTINUE
C
C
         STRN( 1 ) = DUDX
         STRN( 2 ) = DVDY
         STRN( 3 ) = DUDY + DVDX
         STRN( 4 ) = 0.
         IF (IFLAG.EQ.3) STRN( 4 ) = A5/RAD
C
      CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,1)
      CALL EQUILB(CST,NCB,NRB)
 200  CONTINUE
C
 1000 RETURN
C
C
C
C
C
C
      ENTRY S2DNS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
C
      DO 400 INTGPN = 1 , NIP
         CALL JACB2D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         IF(IFLAG.EQ.3) CALL AXISYM(INTGPN,ELNUM,NNEL,RAD,THICK)
         CALL B2DNS(INTGPN,NNEL,RAD)
```

```
         CST = DETJAC*THICK*W( INTGPN )
C
         STRN( 1 ) = DUDX + 0.5*(DUDX**2 + DVDX**2 )
         STRN( 2 ) = DVDY + 0.5*(DUDY**2 + DVDY**2 )
         STRN( 3 ) = DUDY + DVDX + DUDX*DUDY + DVDX*DVDY
         STRN( 4 ) = 0.
         IF (IFLAG.EQ.3) STRN( 4 ) = A5 + 0.5*(A5**2)
C
      CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,1)
      CALL EQUILB(CST,NCB,NRB)
 400  CONTINUE
C
 2000 RETURN
C
C
C
C
C
C
      ENTRY S3DLS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
C
      DO 600 INTGPN = 1 , NIP
         CALL JACB3D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         CALL B3DLS(NNEL)
         CST = DETJAC*W( INTGPN )
C
         DUDX = 0.
         DVDX = 0.
         DWDX = 0.
         DUDY = 0.
         DVDY = 0.
         DWDY = 0.
         DUDZ = 0.
         DVDZ = 0.
         DWDZ = 0.
C
C*VDIR: ASSUME COUNT(20)
                DO 500 K1 = 1,NNEL
                K13 = 3*K1
                K12 = K13 - 1
                K11 = K13 - 2
                DUDX = DUDX + NX(K1)*UTOTAL(II(K11))
                DVDX = DVDX + NX(K1)*UTOTAL(II(K12))
                DWDX = DWDX + NX(K1)*UTOTAL(II(K13))
                DUDY = DUDY + NY(K1)*UTOTAL(II(K11))
                DVDY = DVDY + NY(K1)*UTOTAL(II(K12))
                DWDY = DWDY + NY(K1)*UTOTAL(II(K13))
                DUDZ = DUDZ + NZ(K1)*UTOTAL(II(K11))
                DVDZ = DVDZ + NZ(K1)*UTOTAL(II(K12))
                DWDZ = DWDZ + NZ(K1)*UTOTAL(II(K13))
 500            CONTINUE
C
```

```
            STRN( 1 ) = DUDX
            STRN( 2 ) = DVDY
            STRN( 3 ) = DWDZ
            STRN( 4 ) = DUDY + DVDX
            STRN( 5 ) = DWDY + DVDZ
            STRN( 6 ) = DWDX + DUDZ
C
      CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,1)
      CALL EQUILB(CST,NCB,NRB)
 600  CONTINUE
C
 3000 RETURN
C
C
C
C
C
      ENTRY S3DNS (ELNUM,ITYPE,NNEL,NRB,NCB,NIP,MATNUM,IOUT)
C
      DO 800 INTGPN = 1 , NIP
         CALL JACB3D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
         CALL B3DNS(NNEL)
         CST = DETJAC*W( INTGPN )
C
         STRN( 1 ) = DUDX + 0.5*(DUDX**2 + DVDX**2 + DWDX**2)
         STRN( 2 ) = DVDY + 0.5*(DUDY**2 + DVDY**2 + DWDY**2)
         STRN( 3 ) = DWDZ + 0.5*(DUDZ**2 + DVDZ**2 + DWDZ**2)
         STRN( 4 ) = DUDY + DVDX + DUDX*DUDY + DVDX*DVDY + DWDX*DWDY
         STRN( 5 ) = DWDY + DVDZ + DUDZ*DUDY + DVDZ*DVDY + DWDZ*DWDY
         STRN( 6 ) = DWDX + DUDZ + DUDZ*DUDX + DVDZ*DVDX + DWDZ*DWDX
C
      CALL MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,1)
      CALL EQUILB(CST,NCB,NRB)
 800  CONTINUE
C
 4000 RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= E Q U L I B ===============================
C
      SUBROUTINE EQUILB(CST,NCB,NRB)
      REAL*8 B,RE,STRESS,STRS,DS,CST,TEMP
      COMMON/ASSEM2/II(60)
      COMMON/ELSTR2/STRS(6)
      COMMON/MAIN4/RE(8000)
      COMMON/B1/B(6,60)
      DIMENSION TEMP(60)
C
C*VDIR: ASSUME COUNT(16)
C*VDIR: IGNORE RECRDEPS
      DO 200 K1 = 1 , NCB
      TEMP( K1 ) = 0.
      DO 100 K2 = 1 , NRB
  100 TEMP( K1 ) = TEMP( K1 ) + B(K2 , K1)*STRS( K2 )
  200 RE(II(K1)) = RE(II(K1)) + TEMP( K1 )*CST
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================= C A U C H Y ================================
C
      SUBROUTINE CAUCHY(ELNUM,ITYPE,NNEL,NNDF,INTGPN,STRESS,CSTR)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NX,NY,NZ,JACMAT
      INTEGER*4 ELNUM
      COMMON/JACOB1/NX(20),NY(20),NZ(20)
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/INPUT2/NOP(20,2000)
C
      DIMENSION JACMAT(3,3),PIOLA(3,3),CAUCH(3,3),STRESS(6),CSTR(6)
C
C
      DUDX = 0.
      DUDY = 0.
      DUDZ = 0.
      DVDX = 0.
      DVDY = 0.
      DVDZ = 0.
      DWDX = 0.
      DWDY = 0.
      DWDZ = 0.
C
      IF (ITYPE.GT.300) THEN
           CALL JACB3D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
C
C*VDIR: ASSUME COUNT(20)
           DO 10 K1 = 1 , NNEL
           K11 = NNDF*(NOP(K1 , ELNUM) - 1) + 1
           K12 = K11 + 1
           K13 = K11 + 2
           DUDX = DUDX + NX( K1 )*UTOTAL( K11 )
           DUDY = DUDY + NY( K1 )*UTOTAL( K11 )
           DUDZ = DUDZ + NZ( K1 )*UTOTAL( K11 )
           DVDX = DVDX + NX( K1 )*UTOTAL( K12 )
           DVDY = DVDY + NY( K1 )*UTOTAL( K12 )
           DVDZ = DVDZ + NZ( K1 )*UTOTAL( K12 )
           DWDX = DWDX + NX( K1 )*UTOTAL( K13 )
           DWDY = DWDY + NY( K1 )*UTOTAL( K13 )
           DWDZ = DWDZ + NZ( K1 )*UTOTAL( K13 )
 10        CONTINUE
C
           PIOLA(1 , 1) = STRESS( 1 )
           PIOLA(2 , 2) = STRESS( 2 )
           PIOLA(3 , 3) = STRESS( 3 )
           PIOLA(1 , 2) = STRESS( 4 )
           PIOLA(2 , 1) = STRESS( 4 )
           PIOLA(1 , 3) = STRESS( 6 )
           PIOLA(3 , 1) = STRESS( 6 )
           PIOLA(2 , 3) = STRESS( 5 )
```

```
            PIOLA(3 , 2) = STRESS( 5 )
      ELSE
            CALL JACB2D(INTGPN,ELNUM,NNEL,IERROR,DETJAC)
C
C*VDIR: ASSUME COUNT(8)
            DO 20 K1 = 1 , NNEL
            K11 = NNDF*(NOP(K1 , ELNUM) - 1) + 1
            K12 = K11 + 1
            DUDX = DUDX + NX( K1 )*UTOTAL( K11 )
            DUDY = DUDY + NY( K1 )*UTOTAL( K11 )
            DVDX = DVDX + NX( K1 )*UTOTAL( K12 )
            DVDY = DVDY + NY( K1 )*UTOTAL( K12 )
 20         CONTINUE
C
            PIOLA(1 , 1) = STRESS( 1 )
            PIOLA(2 , 2) = STRESS( 2 )
            PIOLA(1 , 2) = STRESS( 3 )
            PIOLA(2 , 1) = STRESS( 3 )
            PIOLA(3 , 3) = STRESS( 4 )
            PIOLA(1 , 3) = 0.
            PIOLA(2 , 3) = 0.
            PIOLA(3 , 1) = 0.
            PIOLA(3 , 2) = 0.
      END IF
C
      JACMAT(1 , 1) = 1. + DUDX
      JACMAT(1 , 2) = DUDY
      JACMAT(1 , 3) = DUDZ
      JACMAT(2 , 1) = DVDX
      JACMAT(2 , 2) = 1. + DVDY
      JACMAT(2 , 3) = DVDZ
      JACMAT(3 , 1) = DWDX
      JACMAT(3 , 2) = DWDY
      JACMAT(3 , 3) = 1. + DWDZ
C
      DJAC = (1. + DUDX)*((1. + DVDY)*(1. + DWDZ) - DVDZ*DWDY)-
     1       DUDY*(DVDX*(1. + DWDZ) - DVDZ*DWDX) +
     2       DUDZ*(DVDX*DWDY - (1. + DVDY)*DWDX)
C
C*VDIR: PREFER SCALAR
      DO 40 K1 = 1 , 3
C*VDIR: PREFER SCALAR
      DO 40 K2 = 1 , K1
      SUM = 0.
C*VDIR: PREFER SCALAR
      DO 30 K3 = 1 , 3
C*VDIR: PREFER SCALAR
      DO 30 K4 = 1 , 3
  30  SUM = SUM + PIOLA(K3 , K4)*JACMAT(K1 , K3)*JACMAT(K2 , K4)
  40  CAUCH(K1 , K2) = SUM/DJAC
C
      IF (ITYPE.GT.300) THEN
```

```
           CSTR( 1 ) = CAUCH(1 , 1)
           CSTR( 2 ) = CAUCH(2 , 2)
           CSTR( 3 ) = CAUCH(3 , 3)
           CSTR( 4 ) = CAUCH(2 , 1)
           CSTR( 5 ) = CAUCH(3 , 2)
           CSTR( 6 ) = CAUCH(3 , 1)
      ELSE
           CSTR( 1 ) = CAUCH(1 , 1)
           CSTR( 2 ) = CAUCH(2 , 2)
           CSTR( 3 ) = CAUCH(2 , 1)
           CSTR( 4 ) = CAUCH(3 , 3)
      END IF
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================== G L O B A L ===============================
C
      SUBROUTINE GLOBAL
C
C =====================================================================
C I                                                                   I
C I     SUBROUTINE GLOBAL IS USED TO MODIFY THE FINAL GLOBAL          I
C I     STIFFNESS MATRIX. THIS IS DONE IN ORDER TO SOLVE THE          I
C I     SET OF SIMULTINOUS EQUATIONS BY THE METHOD OF MODIFFICATION.  I
C I     THIS SUBROUTINE IS DESIGED FOR MODIFICATION OF BANDED         I
C I     NONSYMETRIC MATRICES IN THEIR CONDENSED FORM.                 I
C I                                                                   I
C I                                                                   I
C =====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION IDOF( 1 )
C
C
C                      E N T R Y     G L O B 1
C
      ENTRY GLOB1(NNODES,NNDF,NTDF,IDOF)
C
      MDOF = NNDF*NNODES
      ICOUNT = 0
C
C*VDIR: PREFER SCALAR
      DO 40 ID = 1 , MDOF
      IF (IDOF( ID ).EQ.0) THEN
          ICOUNT = ICOUNT + 1
          IDOF( ID ) = ICOUNT
      ELSE IF (IDOF( ID ).GT.0) THEN
          IDOF( ID ) = 0
      END IF
  40  CONTINUE
      NTDF = ICOUNT
C
      RETURN
C
C
C                      E N T R Y     G L O B 2
C
      ENTRY GLOB2(NNODES,NNDF,NTDF,IDOF)
C
      MDOF = NNDF*NNODES
      ICOUNT = 0
C
C*VDIR: PREFER SCALAR
      DO 50 ID = 1 , MDOF
      IF (IDOF( ID ).GT.0) THEN
```

```
            ICOUNT  = ICOUNT + 1
            IDOF( ID ) = ICOUNT
         END IF
   50    CONTINUE
         NTDF = ICOUNT
C
         RETURN
         END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================== D I A G N L ================================
C
      SUBROUTINE DIAGNL(NELEM,NNDF,NTDF,IDOF,JDIAG,NTSK,MBAND,IFLAG2,
     1 IOUT)
C ====================================================================
C I                                                                  I
C I    THIS PROGRAM COMPUTES THE VECTOR CONTAINING THE ADDRESSES     I
C I    OF THE DIAGNAL ELEMENTS OF THE STIFFNESS MATIX. IT ALSO       I
C I    CALCULATES THE BANDWIDTH AND THE AVERAGE BANDWIDTH OF THE     I
C I    OF THE STIFFNESS MATRIX AND PRINTS THESE STATISTICS.          I
C I                                                                  I
C I    A R G U M E N T     L I S T                                   I
C I                                                                  I
C I    NELEM     = TOTAL NUMBER OF ELEMENTS                          I
C I                                                                  I
C I    NNDF      = NUMBER OF NODAL DEGREES OF FREEDOM                I
C I                                                                  I
C I    NTDF      = NUMBER OF TOTAL DEGREES OF FREEDOM                I
C I                                                                  I
C I    IDOF(I)   = VECTOR CONTAINING THE D.O.F. NUMBERS OF THE NODES I
C I                                                                  I
C I    JDIAG(I)  = VECTOR CONTAINING THE ADDRESS OF THE DIAGNAL TERMS I
C I                IN THE GLOBAL STIFFNESS MATRIX 'SKG'              I
C I                                                                  I
C I    NTSK      = NUMBER OF TERMS IN THE GLOBAL STIFFNESS MATRIX    I
C I                                                                  I
C I    MBAND     = HALF BANDWIDTH OF THE STIFFNESS MATRIX            I
C I                                                                  I
C I    IOUT      = OUTPUT DEVICE NUMBER                              I
C I                                                                  I
C I                                                                  I
C I    C O M M O N     B L O C K S                                   I
C I                                                                  I
C I    NOP(I,J)  = MEMBER INCIDENCES                                 I
C I                                                                  I
C I                                                                  I
C I                                                                  I
C ====================================================================
      INTEGER ELNUM
      COMMON/INPUT2/NOP(20,2000)
      DIMENSION IDOF(1), JDIAG(1)
C
      MBAV = 0
      MBAND = 0
C*VDIR: PREFER VECTOR
      DO 5 K = 1 , NTDF
    5 JDIAG( K ) = 0
C
      DO 30 ELNUM = 1 , NELEM
      CALL ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
```

```
      MAXDOF = 0
      MINDOF = 1000000
      DO 20 NODE = 1 , NNEL
      DO 20 IDIR = 1 , NNDF
      K = NNDF*(NOP(NODE , ELNUM) - 1) + IDIR
      IF(IDOF( K ))20,20,10
 10   MAXDOF = MAX0(MAXDOF , IDOF( K ))
      MINDOF = MIN0(MINDOF , IDOF( K ))
 20   CONTINUE
C
C        AT THIS POINT THE HIGHT OF EACH COLUMN IS STORED IN JDIAG
C
           DO 26 NODE = 1 , NNEL
           DO 26 IDIR = 1 , NNDF
           ID = NNDF*(NOP(NODE , ELNUM) - 1) + IDIR
           ID = IDOF( ID )
           IF ( ID )26 , 26 , 25
 25        MHT = ID - MINDOF + 1
           IF(MHT.GT.JDIAG( ID )) JDIAG( ID ) = MHT
 26        CONTINUE
C
C        FIND THE BANDWIDTH AND THE AVERAGE BANDWIDTH
C
      MBN = MAXDOF - MINDOF
      MBAND = MAX0(MBAND , MBN)
      MBAV = MBAV + MBAND
 30   CONTINUE
C
      MBAV = MBAV/NELEM + 1
      MBAND = MBAND + 1
C
C        LOCATION OF EACH DIAGNAL TERM WILL NOW BE STORED IN JDIAG
C
      IF (IFLAG2.EQ.0) THEN
           MHT = 1
           ID  = 0
           DO 40 K = 1 , NTDF+1
           ID = ID + MHT
           MHT = JDIAG( K )
  40       JDIAG( K ) = ID
           NTSK = JDIAG(NTDF+1) - JDIAG( 1 )
      ELSE
           ID = 0
           DO 50 K = 1 , NTDF
           ID = ID + JDIAG( K )
  50       JDIAG( K ) = ID
           NTSK = 2*JDIAG( NTDF )
      END IF
C
C        NTSK = NUMBER OF TERMS IN THE GLOBAL STIFFNESS MATRIX "SKG"
C
C
```

```
      WRITE(IOUT , 100)NTDF,MBAND,MBAV,NTSK
100   FORMAT(//1X,'NUMBER OF EQUATIONS = ',I8/1X,'HALF BANDWIDTH = ',
     1 I8/1X,'AVERAGE BANDWIDTH = ',I8/1X,'SIZE OF THE STIFFNESS MATRIX'
     2 ,' = ',I8)
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================= C O O R D ==================================
C
      SUBROUTINE COORD
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 X,Y,Z
      REAL*8 N,NXI,NETA,NSI
      INTEGER ELNUM
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/INPUT2/NOP(20,2000)
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      DIMENSION U( 3 )
C
C               E N T R Y    C O O R D 1
C
      ENTRY COORD1(ELNUM,NNEL,INTGPN,X1,Y1,Z1)
      X1 = 0.
      Y1 = 0.
      Z1 = 0.
C*VDIR: ASSUME COUNT(8)
      DO 100 K = 1 , NNEL
      X1 = X1 + N(K , INTGPN)*X(NOP(K , ELNUM))
      Y1 = Y1 + N(K , INTGPN)*Y(NOP(K , ELNUM))
 100  Z1 = Z1 + N(K , INTGPN)*Z(NOP(K , ELNUM))
C
      RETURN
C
C               E N T R Y    C O O R D 2
C
      ENTRY COORD2(ELNUM,NNEL,INTGPN,NNDF,UXIP,UYIP,UZIP)
      U( 1 ) = 0.
      U( 2 ) = 0.
      U( 3 ) = 0.
C*VDIR: PREFER SCALAR
      DO 200 K = 1 , NNEL
C*VDIR: PREFER SCALAR
      DO 200 ID = 1 , NNDF
      K1 = NNDF*(NOP(K , ELNUM) - 1) + ID
      U( ID ) = U( ID ) + N(K , INTGPN)*UTOTAL( K1 )
 200  CONTINUE
      UXIP = U( 1 )
      UYIP = U( 2 )
      UZIP = U( 3 )
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ======================= M A T M O D ================================
C
      SUBROUTINE MATMOD(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,
     #                  ICODE)
      IMPLICIT REAL*8 (A-H,O-Z)
      INTEGER ELNUM
      COMMON/INPUTF/MATYPE(10)
C
      I = MATYPE( MATNUM )
      IF (I.EQ.1) THEN
           CALL ELAST(ITYPE,MATNUM,IFLAG,IOUT,ICODE)
      ELSE IF(I.EQ.2) THEN
           CALL PLASTN(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,
     #                 ICODE)
      ELSE IF(I.EQ.3) THEN
           CALL PLASTL(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,DETJAC,
     #                 ICODE)
      ELSE
           WRITE (IOUT , 100) I
           STOP
      END IF
C
      RETURN
 100  FORMAT (/1X,'INVALID MATERIAL TYPE(',I3,') SPECIFIED')
      END
```

APPENDIX B

SOLVER MODULE

```
C
C ========================= S O L V E 1 ==============================
C
      SUBROUTINE SOLVE1(A,C,B,JDIAG,NEQ,AFAC,BACK)
C
C ====================================================================
C I                                                                  I
C I   P R O G R A M:                                                 I
C I                                                                  I
C I   PROGRAM 'SOLVE1' IS USED TO SOLVE A SERIES OF BANDED           I
C I   NONSYMETRIC LINEAR EQUATIONS USING THE GAUSS ELIMINATION/BACK  I
C I   SUBSTITUTION WITH NO COLUMN PIVOTING.                          I
C I                                                                  I
C I   STORAGE:    COEFICIANT MATRIX SHOULD BE STORED IN TWO ONE      I
C I               DIMENSIONAL ARRAYS USING THE SKYLINE OR THE        I
C I               PROFILE METHOD                                     I
C I               A( K ) = UPPER TRIANGULAR MATRIX                   I
C I               C( K ) = LOWER TRIANGULAR MATRIC                   I
C I               B( K ) = RIGHT HAND SIDE VECTOR ON CALL            I
C I                      = VECTOR OF UNKNOWNS ON RETURN              I
C I                                                                  I
C I      LAST UPDATE:  12-30-1988                                    I
C I      BY:  M. FOROOZESH                                           I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      LOGICAL AFAC,BACK
      DIMENSION A( 1 ),C( 1 ),B( 1 ),JDIAG( 1 )
C
C        FACTOR A TO UT*D*U, REDUCE B TO Y
C
      JR = 0
      DO 300 J = 1 , NEQ
      JD = JDIAG( J )
      JH = JD - JR
      IF (JH.LE.1) GO TO 300
      IS = J + 1 - JH
      IE = J - 1
      IF (.NOT.AFAC) GO TO 250
      K = JR + 1
      ID = 0
C
C        REDUCE ALL EQUATIONS EXCEPT DIAGNAL
C
      DO 200 I = IS , IE
      IR = ID
```

```
      ID = JDIAG( I )
      IH = MINO(ID-IR-1 , I - IS)
      IF (IH.EQ.0) GO TO 150
      A( K ) = A( K ) - DOTPRO(A( K-IH ),C( ID-IH ),IH)
      C( K ) = C( K ) - DOTPRO(C( K-IH ),A( ID-IH ),IH)
 150  IF (A(ID).NE.0.0) C( K ) = C( K )/A( ID )
 200  K = K + 1
C
C         REDUCE THE DIAGNAL TERM
C
      A( JD ) = A( JD ) - DOTPRO(A( JR+1 ),C( JR+1 ),JH-1)
C
C         FORWARD REDUCE THE RIGHT HAND SIDE
C
 250  IF ( BACK ) B( J ) = B( J ) - DOTPRO(C( JR+1 ),B( IS ),JH-1)
 300  JR = JD
      IF(.NOT.BACK) RETURN
C
C         BACK SUBSTITUTION
C
      J = NEQ
      JD = JDIAG( J )
 500  IF (A( JD ).NE.0.0) B( J ) = B( J )/A( JD )
      D = B( J )
      J = J - 1
      IF (J.LE.0) RETURN
      JR = JDIAG( J )
      IF (JD-JR.LE.1) GO TO 700
      IS = J - JD + JR + 2
      K = JR - IS + 1
      DO 600 I = IS , J
 600  B( I ) = B( I ) - A( I+K )*D
 700  JD = JR
      GO TO 500
      END
C
C ============================= D O T =================================
C
      FUNCTION DOTPRO(A,B,N)
      REAL*8 A,B,DOTPRO,TEMP
      DIMENSION A( 1 ) , B( 1 )
      TEMP = 0.0
      DO 100 I = 1 , N
  100 TEMP = TEMP + A( I )*B( I )
      DOTPRO = TEMP
C
      RETURN
      END
```

```
C
C ========================= S O L V E 2 ==============================
C
      SUBROUTINE SOLVE2(A,R,JDIAG,NEQU,KKK,IOUT)
C
C ===================================================================
C I                                                                 I
C I   THIS PROGRAM IS USED TO SOLVE FINITE ELEMENT STATIC EQUILIB.  I
C I   EQUATIONS IN CORE, USING COMPACTED STORAGE AND COLUMN REDUCTON I
C I   SCHEME                                                        I
C I                                                                 I
C ===================================================================
C
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A( 1 ), JDIAG( 1 ), R( 1 )
C
C        PERFORM L*D*L FACTORIZATION OF THE STIFFNESS MATRIX
C
      IF (KKK - 2) 40 , 150 , 150
 40   DO 140 N = 1 , NEQU
      KN = JDIAG( N )
      KL = KN + 1
      KU = JDIAG(N+1) - 1
      KH = KU - KL
      IF (KH) 110,90,50
 50   K = N - KH
      IC = 0
      KLT = KU
      DO 80 J = 1 , KH
      IC = IC + 1
      KLT = KLT - 1
      KI = JDIAG( K )
      ND = JDIAG( K + 1 ) - KI - 1
      IF (ND) 80 , 80 , 60
 60   KK = MIN0(IC,ND)
      C = 0.
      DO 70 L = 1 , KK
 70   C = C + A(KI + L)*A(KLT + L)
      A( KLT ) = A( KLT ) - C
 80   K = K + 1
 90   K = N
      B = 0.
      DO 100 KK = KL , KU
      K = K - 1
      KI = JDIAG( K )
      C = A( KK )/ A( KI )
      B = B + C*A( KK )
100   A( KK ) = C
      A( KN ) = A( KN ) - B
110   IF (A( KN )) 120 ,120 , 140
120   WRITE(IOUT , 2000) N , A( KN )
      STOP
```

```
 140  CONTINUE
      RETURN
C
C       REDUCE THE RIGHT-HAND-SIDE LOAD VECTOR
C
 150  DO 180 N = 1 , NEQU
      KL = JDIAG( N ) + 1
      KU = JDIAG( N + 1) - 1
      IF(KU-KL) 180 , 160 , 160
 160  K = N
      C = 0.
      DO 170 KK = KL , KU
      K = K - 1
 170  C = C + A( KK )*R( K )
      R( N ) = R( N ) - C
 180  CONTINUE
C
C       BACK-SUBSTITUTE
C
      DO 200 N = 1 , NEQU
      K = JDIAG( N )
 200  R( N ) = R( N )/ A( K )
C
      IF (NEQU.EQ.1) RETURN
      N = NEQU
      DO 230 L = 2 , NEQU
      KL = JDIAG( N ) + 1
      KU = JDIAG( N + 1 ) - 1
      IF ( KU - KL ) 230 , 210 , 210
 210  K = N
      DO 220 KK = KL , KU
      K = K - 1
 220  R( K )  = R( K ) - A( KK )*R( N )
 230  N = N - 1
      RETURN
 2000 FORMAT(//1X,'STOP - STIFFNESS MATRIX NOT POSITIVE DEFINITE '//
     1 1X,' NONPOSITIVE PIVOT FOR EQUATION ',I4//1X,'PIVOT = ',E20.12)
      END
```

APPENDIX C

PLASTICITY MODULE

```
      SUBROUTINE PLAST(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,ICODE)
C
C =====================================================================
C I                                                                   I
C I   P R O G R A M:                                                  I
C I                                                                   I
C I      'PLAST' IS ENTRANCE POINT TO THE PLASTICITY MODULE.          I
C I      DEPENDING ON THE VALUE OF "ICODE" THIS ROUTINE PERFORMS      I
C I      THE FOLLOWING FUNCTIONS.                                     I
C I                                                                   I
C I    1) ICODE = 1; CALLS SUBROUTINE 'MISES1' FOR EVALUATION         I
C I                  OF THE ELASTO-PLASTIC STIFFNESS TENSOR.          I
C I    2) ICODE = 2; CALLS SUBROUTINE 'MISES2' FOR EVALUATION         I
C I                  OF THE STRESSES, PLASTIC STRAINS, ETC.           I
C I                                                                   I
C I   A R G U M E N T   L I S T S:                                    I
C I                                                                   I
C I     ELNUM   = ELEMENT NUMBER                                      I
C I     ITYPE   = ELEMENT TYPE                                        I
C I     MATNUM  = MATERIAL NUMBER                                     I
C I     INTGPN  = INTEGRATION POINT NUMBER                            I
C I     IFLAG   = ANALYSIS TYPE CODE                                  I
C I               1; PLANE STRESS                                     I
C I               2; PLANE STRAIN                                     I
C I               3; AXISYMMETRIC                                     I
C I     IOUT    = OUTPUT DEVICE NUMBER                                I
C I     ICODE   = TASK DEFINITION CODE AS DEFINED ABOVE.              I
C I                                                                   I
C =====================================================================
      INTEGER ELNUM
C
      IF(ITYPE.GT.300) THEN
           IEND = 6
      ELSE
           IEND = 4
      END IF
C
      IF (ICODE.EQ.0) THEN
           CALL MISES1(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,IEND)
      ELSE
           CALL MISES2(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,IEND)
      END IF
      RETURN
      END
```

```
C
C ====================== M I S E S ================================
C
      SUBROUTINE MISES
C
C =================================================================
C I                                                               I
C I   P R O G R A M:                                              I
C I                                                               I
C I      'MISES' IS THE CONTROL UNIT FOR:                         I
C I                                                               I
C I     1) EVALUATION OF THE STRESS-STRAIN STIFFNESS MATRIX       I
C I     2) CALCULATION OF THE STRESSES, PLASTIC STRAINS,          I
C I        PLASTIC WORK, AND THE SHIFT STRESS TENSOR.             I
C I                                                               I
C I   E N T R Y   P O I N T S:                                    I
C I                                                               I
C I     MISES1: EVALUATES THE STRESS-STRAIN STIFFNESS MATRIX      I
C I     MISES2: EVALUATES THE STRESSES, STRAINS, ETC.             I
C I                                                               I
C I   A R G U M E N T   L I S T S:                                I
C I                                                               I
C I     ELNUM   = ELEMENT NUMBER                                  I
C I     ITYPE   = ELEMENT TYPE                                    I
C I     MATNUM  = MATERIAL NUMBER                                 I
C I     INTGPN  = INTEGRATION POINT NUMBER                        I
C I     IFLAG   = ANALYSIS TYPE CODE                              I
C I               1; PLANE STRESS                                 I
C I               2; PLANE STRAIN                                 I
C I               3; AXISYMMETRIC                                 I
C I     IOUT    = OUTPUT DEVICE NUMBER                            I
C I     IEND    = 4; PLANE STRESS, PLANE STRAIN AND AXISYMMETRIC  I
C I             = 6; 3-D                                          I
C I                                                               I
C I                                                               I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NUX,NUY,NUZ
      CHARACTER*1 IYIELD,IY
      INTEGER ELNUM
      COMMON/MEMO/MATAD,MATDE
      COMMON/TEMP/PRESS,PLWORK
      COMMON/ISHAP2/W(27)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CENTER(6),WORK,IYIELD
      COMMON/CONTR1/INCREM,NIT
      COMMON/ELSTR1/STRN(6)
      COMMON/ELSTR2/STRS(6)
      COMMON/ADMAT1/AD(9,9)
      COMMON/PLAST1/IYIEL(2000)
      COMMON/FDER1/FJ,FK,FS(9),FE(9),FZ(9)
```

```
      COMMON/MATER1/DEP(6,6)
      COMMON/ELPLD1/DEPM(9,9),ALAM(9),AMU(9)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE
      COMMON/INPUT5/NUX(10),NUY(10),NUZ(10),EX(10),EY(10),EZ(10),P1X(10
     1                  ,P1Y(10),P1Z(10),P2X(10),P2Y(10),P2Z(10)
      DIMENSION S0(9),C(9),Z(9),RR(9),E(9),DEL(9),ED(9),
     1          EDOT(9),SF(9),EDOTEL(9),EDOTPL(9),DELAS(6),DE(6),SDOT(9
C
      DATA DEL/1.,0.,0.,0.,1.,0.,0.,0.,1./
C
C
C
C
C                 E N T R Y   M I S E S 1
C
      ENTRY MISES1(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,IEND)
C
      IYIELD = ' '
      IF (INCREM.GT.1) THEN
           CALL IOGET(LDEV,201,'(A201)',6)
           CALL IOBKS(LDEV)
      END IF
C
      IF (IYIELD.EQ.'Y') THEN
C
C --- CALCULATION OF THE USEFUL MATRICES
C
           CALL TENSOR(ITYPE,STRESS,S0,1.)
           CALL TENSOR(ITYPE,STRAIN,E,0.5)
           CALL TENSOR(ITYPE,CENTER,Z,1.)
           DO 200 K1 = 1 , 9
 200       C( K1 ) = 2.*E( K1 ) + DEL( K1 )
C
C --- GET THE MATERIAL PARAMETERS
C
           C3 = P1Y( MATNUM )
           SY = P1Z( MATNUM )
           BETA = P1X( MATNUM )
           YOUNG = EX( MATNUM )
           POISS = NUX( MATNUM )
C
C --- CALCULATION OF THE FOURTH ORDER ELASTIC STIFFNESS MATRIX
C
      IF (MATNUM.NE.MATAD) THEN
           CALL ADMAT(YOUNG,POISS)
           MATAD = MATNUM
      END IF
C
C --- CALCULATION OF THE JACOBIAN OF DEFORMATION
C
           CALL DEFJAC(E,DEL,RR,DJAC)
```

```
C
C --- CALCULATION OF THE YIELD FUNCTION
C
            CALL YIELD(S0,C,Z,WORK,DJAC,C3,SY,F,F1,F2)
C
C --- CALCULATION OF THE PARTIAL DERIVATIVE OF THE YIELD FUNCTION
C --- F WITH RESPECT TO THE <STRESS>,<STRAIN>, THE JACOBIAN.
C
            CALL FDER(S0,C,Z,DJAC,F1,F2,C3)
C
C --- CALCULATION OF THE ELASTOPLASTIC STIFFNESS MATRIX
C
            CALL ELPLD(S0,Z,RR,DJAC,BETA,DEN,0)
C
C --- CONVERTION OF THE FORTH ORDER STIFFNESS TENSOR TO A SECOND
C --- ORDER TENSOR
C
            CALL CONVER(DEPM,DEP,IFLAG,ITYPE)
      ELSE
            CALL DELAST(ITYPE,MATNUM,IFLAG)
      END IF
      RETURN
C
C
C
C
C                       E N T R Y   M I S E S 2
C
      ENTRY MISES2(ELNUM,ITYPE,MATNUM,INTGPN,IFLAG,IOUT,IEND)
C
      FACTOR = 1.
      FACSUM = 0.
C
      IF (INCREM.GT.1) THEN
            CALL IOGET(LDEV1,201,'(A201)',6)
      ELSE
            DO 10 K1 = 1 , IEND
            STRAIN( K1 ) = 0.
            STRESS( K1 ) = 0.
            CENTER( K1 ) = 0.
   10       STRELA( K1 ) = 0.
            WORK = 0.
      END IF
C
C --- CALCULATION OF THE STRAIN INCREMENT
C
      DO 20 K1 = 1 , IEND
   20 DE( K1 ) = STRN( K1 ) - STRAIN( K1 )
C
C --- CALCULATION OF THE USEFULL TENSORS
C
      CALL TENSOR(ITYPE,STRESS,S0,1.)
```

```
      CALL TENSOR(ITYPE,STRAIN,E,0.5)
      CALL TENSOR(ITYPE,DE,ED,0.5)
      CALL TENSOR(ITYPE,CENTER,Z,1.)
C
C --- GET THE MATERIAL PARAMETERS
C
      C3 = P1Y( MATNUM )
      SY = P1Z( MATNUM )
      BETA = P1X( MATNUM )
      YOUNG = EX( MATNUM )
      POISS = NUX( MATNUM )
C
C --- CALCULATION OF THE FOURTH ORDER ELASTIC STIFFNESS MATRIX
C
      IF (MATNUM.NE.MATAD) THEN
           CALL ADMAT(YOUNG,POISS)
           MATAD = MATNUM
      END IF
C
      DO 50 K1 = 1 , 9
      C( K1 ) = 2.*E( K1 ) + DEL( K1 )
      CST = 0.
      DO 40 K2 = 1 , 9
   40 CST = CST + AD(K1 , K2)*ED( K2 )
   50 SDOT( K1 ) = CST
C
C --- CALCULATION OF THE JACOBIAN OF DEFORMATION
C
   34 CALL DEFJAC(E,DEL,RR,DJAC)
C
C --- START OF THE INCREMENTATION LOOP
C --- CALCULATION OF THE TRIAL ELASTIC STRESS
C
      DO 35 K1 = 1 , 9
   35 SF( K1 ) = SO( K1 ) + SDOT( K1 )
C
C --- CALCULATION OF THE YIELD FUNCTION FOR THE TRIAL ELASTIC STRESS
C
      CALL YIELD(SF,C,Z,WORK,DJAC,C3,SY,F,F1,F2)
C
      IF (F.LE.0.) THEN
           FACSUM = FACSUM + FACTOR
           DO 60 K1 = 1 , IEND
 60        STRELA( K1 ) = STRELA( K1 ) + DE( K1 )*FACTOR
C
           DO 65 K1 = 1 , 9
           E( K1 ) = E( K1 ) + EDOT( K1 )
           C( K1 ) = 2.*E( K1 ) + DEL( K1 )
 65        SO( K1 ) = SF( K1 )
           IYIELD = ' '
C
      ELSE IF(F.GT.0.) THEN
```

```
          IF(FACTOR.EQ.1.) THEN
               FACTOR = 1.0D-2
C
               DO 68 K1 = 1 , 9
               SDOT( K1 ) = FACTOR*SDOT( K1 )
   68          EDOT( K1 ) = FACTOR*ED( K1 )
               GO TO 34
          END IF
          CALL YIELD(S0,C,Z,WORK,DJAC,C3,SY,F0,F1,F2)
          IF (ELNUM.EQ.320.AND.INTGPN.EQ.4.AND.FACSUM.GT.0.9) THEN
          WRITE(6 , *) 'F0= ',F0,INCREM,NIT,DJAC
          END IF
               FACSUM = FACSUM + FACTOR
C
               CALL FDER(S0,C,Z,DJAC,F1,F2,C3)
               CALL ELPLD(S0,Z,RR,DJAC,BETA,DEN,1)
C
               ALAMDA = 0.
               AMUDOT = 0.
               DO 70 K1 = 1 , 9
               AMUDOT = AMUDOT + AMU( K1 )*EDOT( K1 )
   70          ALAMDA = ALAMDA + ALAM( K1 )*EDOT( K1 )
C
               DO 80 K1 = 1 , 9
               E( K1 ) = E( K1 ) + EDOT( K1 )
               C( K1 ) = 2.*E( K1 ) + DEL( K1 )
               EDOTPL( K1 ) = ALAMDA*FS( K1 )
               EDOTEL( K1 ) = EDOT( K1 ) - EDOTPL( K1 )
   80          Z( K1 ) = Z( K1 ) + (S0( K1 )-Z( K1 ))*AMUDOT
C
               DO 100 K1 = 1 , 9
               CST = 0.
               DO 90 K2 = 1 , 9
   90          CST = CST + AD(K1 , K2)*EDOTEL( K2 )
               WORK = WORK + (S0( K1 ) + 0.5*CST)*EDOTPL( K1 )/DJAC
  100          S0( K1 ) = S0( K1 ) + CST
C
               CALL VECTOR(ITYPE,EDOTEL,DELAS,2.)
C
               DO 110 K1 = 1 , IEND
  110          STRELA( K1 ) = STRELA( K1 ) + DELAS( K1 )
C
               IYIELD = 'Y'
      END IF
      IF (FACSUM.LT.1.) GO TO 34
C
C       DEFINE THE 'IYIEL' VECTOR FOR FUTURE PLOTTING
C
  115 IF (IYIELD.EQ.'Y') THEN
           ITEMP = IBSET(IYIEL( ELNUM ) , INTGPN)
           IYIEL( ELNUM ) = ITEMP
      ELSE
```

```
            ITEMP = IBCLR(IYIEL( ELNUM ) , INTGPN)
            IYIEL( ELNUM ) = ITEMP
      END IF
C
      DO 120 K1 = 1 , IEND
 120  STRAIN( K1 ) = STRN( K1 )
C
      CALL VECTOR(ITYPE,S0,STRS,1.)
      CALL VECTOR(ITYPE,S0,STRESS,1.)
      IF (ELNUM.EQ.2.AND.INTGPN.EQ.1) PRESS = STRESS( 2 )
      CALL VECTOR(ITYPE,Z,CENTER,1.)
      CALL IOPUT(LDEV2,201,'(A201)',6)
C
      RETURN
      END
```

```
C
C ======================== T E N S O R ============================
C
      SUBROUTINE TENSOR(ITYPE,VECT,TENS,FACT)
C
C =================================================================
C I                                                               I
C I   P R O G R A M:                                              I
C I                                                               I
C I     TENSOR CALCULATES MATRICES WHICH ARE COMMON IN            I
C I     MOST OF THE SUBROUTINES THAT CONSTITUTE THE PLASTICITY    I
C I     FORMULATIONS.                                             I
C I                                                               I
C I   A R G U M E N T    L I S T:                                 I
C I                                                               I
C I       ITYPE       = ELEMENT TYPE                              I
C I       VECT(I)     = VECTOR TO BE CONVERTED TO A TENSOR        I
C I       TENS(I,J)   = TENSOR EQUIVALENT OF VECT(I)              I
C I       FACT        = FACTOR TO BE MULTIPLIES WITH THE          I
C I                     NON-DIAGONAL TERMS OF TENSOR              I
C I                                                               I
C =================================================================
C
      REAL*8 VECT,TENS
      DIMENSION VECT(6),TENS(9)
C
      TENS( 1 ) = VECT( 1 )
      TENS( 5 ) = VECT( 2 )
C
      IF (ITYPE.LT.300) THEN
           TENS( 9 ) = VECT( 4 )
           TENS( 4 ) = VECT( 3 )*FACT
           TENS( 2 ) = TENS( 4 )
           TENS( 7 ) = 0.
           TENS( 3 ) = 0.
           TENS( 8 ) = 0.
           TENS( 6 ) = 0.
      ELSE
           TENS( 9 ) = VECT( 3 )
           TENS( 4 ) = VECT( 4 )*FACT
           TENS( 2 ) = TENS( 4 )
           TENS( 7 ) = VECT( 6 )*FACT
           TENS( 3 ) = TENS( 7 )
           TENS( 8 ) = VECT( 5 )*FACT
           TENS( 6 ) = TENS( 8 )
      END IF
C
      RETURN
      END
```

```
C
C ======================= V E C T O R ==============================
C
      SUBROUTINE VECTOR(ITYPE,TENS,VECT,FACT)
C
C =================================================================
C I                                                               I
C I   P R O G R A M:                                              I
C I                                                               I
C I     VECTOR CALCULATES MATRICES WHICH ARE COMMON IN            I
C I     MOST OF THE SUBROUTINES THAT CONSTITUTE THE PLASTICITY    I
C I     FORMULATIONS.                                             I
C I                                                               I
C I   A R G U M E N T    L I S T:                                 I
C I                                                               I
C I       ITYPE       = ELEMENT TYPE                              I
C I       TENS(I,J)   = TENSOR TO BE CONVERTED TO A VECTOR        I
C I       VECT(I)     = VECTOR EQUIVALENT OT TENS(I , J)          I
C I       FACT        = FACTOR TO BE MULTIPLIES WITH THE          I
C I                     NON-DIAGONAL TERMS OF TENSOR              I
C I                                                               I
C =================================================================
C
      REAL*8 VECT,TENS
      DIMENSION VECT(6),TENS(3,3)
C
      VECT( 1 ) = TENS(1 , 1)
      VECT( 2 ) = TENS(2 , 2)
C
      IF (ITYPE.LT.300) THEN
           VECT( 4 ) = TENS(3 , 3)
           VECT( 3 ) = TENS(1 , 2)*FACT
      ELSE
           VECT( 3 ) = TENS(3 , 3)
           VECT( 4 ) = TENS(1 , 2)*FACT
           VECT( 6 ) = TENS(1 , 3)*FACT
           VECT( 5 ) = TENS(2 , 3)*FACT
      END IF
C
      RETURN
      END
```

```
C
C  ======================= A D M A T ==================================
C
      SUBROUTINE ADMAT(YOUNG,POISS)
C
C =====================================================================
C I                                                                   I
C I   P R O G R A M:                                                  I
C I                                                                   I
C I       'ADMAT' CALCULATES THE FOURTH ORDER ELASTIC STRESS-STRAIN   I
C I       TENSOR.                                                     I
C I                                                                   I
C I   A R G U M E N T    L I S T:                                     I
C I                                                                   I
C I       YOUNG  = YOUNG'S MODULUS                                    I
C I       POISS  = POISSONS RATIO                                     I
C I                                                                   I
C =====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/ADMAT1/AD(3,3,3,3)
C
C --- ALAM =  THE LAMDA LAME CONSTANT
C --- AMUE =  THE MU LAME CONSTANT (THE SHEAR MODULUS G)
C
      ALAM = POISS*YOUNG/(1. + POISS)/(1. - 2.*POISS)
      AMUE = YOUNG/2./(1. + POISS)
C
      AD(1 , 1 , 1 , 1) = ALAM + 2.*AMUE
      AD(1 , 1 , 2 , 2) = ALAM
      AD(1 , 1 , 3 , 3) = ALAM
      AD(2 , 2 , 1 , 1) = ALAM
      AD(2 , 2 , 2 , 2) = ALAM + 2.*AMUE
      AD(2 , 2 , 3 , 3) = ALAM
      AD(3 , 3 , 1 , 1) = ALAM
      AD(3 , 3 , 2 , 2) = ALAM
      AD(3 , 3 , 3 , 3) = ALAM + 2.*AMUE
      AD(1 , 2 , 1 , 2) = AMUE
      AD(2 , 1 , 2 , 1) = AMUE
      AD(1 , 3 , 1 , 3) = AMUE
      AD(3 , 1 , 3 , 1) = AMUE
      AD(2 , 3 , 2 , 3) = AMUE
      AD(3 , 2 , 3 , 2) = AMUE
      AD(1 , 2 , 2 , 1) = AMUE
      AD(2 , 1 , 1 , 2) = AMUE
      AD(1 , 3 , 3 , 1) = AMUE
      AD(3 , 1 , 1 , 3) = AMUE
      AD(2 , 3 , 3 , 2) = AMUE
      AD(3 , 2 , 2 , 3) = AMUE
C
      RETURN
      END
```

```
C
C  ======================= D E F J A C ===============================
C
      SUBROUTINE DEFJAC(E,DEL,RR,DJAC)
C
C ====================================================================
C I                                                                  I
C I   P R O G R A M:                                                 I
C I                                                                  I
C I      'DEFJAC' PERFORMS THE FOLLOWING FUNCTIONS:                  I
C I                                                                  I
C I     1) EVALUATION OF THE DEFORMATION JACOBIAN                    I
C I     2) EVALUATION OF THE MATRIX <RR> WHICH WHEN MULTIPLIED WITH  I
C I        STRAIN INCREMENT TENSOR YIELDS THE RATE OF CHANGE OF      I
C I        THE DETERMINANT OF JACOBIAN.                              I
C I                                                                  I
C I   A R G U M E N T   L I S T:                                     I
C I                                                                  I
C I     E(I,J)  = LAGRANGIAN STRAIN TENSOR                           I
C I     DELTA   = KRONECKER DELTA                                    I
C I     RR(I,J) = THIS MATRIX WHEN DOTED WITH THE STRAIN             I
C I               TENSOR WILL RESULT THE INCREMENT OF THE            I
C I               JACOBIAN.                                          I
C I     DJAC    = DETERMINANT OF DEFORMATION JACOBIAN                I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION E(3,3),DEL(3,3),RR(3,3)

C --- EINV1  = FIRST STRAIN INVARIANT
C --- EINV2  = SECOND STRAIN INVARIANT
C --- EINV3  = THIRD STRAIN INVARIANT

      EINV1 = 0.0
      EINV2 = 0.0
      EINV3 = 0.0
C
      DO 10 K1 = 1 , 3
      EINV1 = EINV1+E(K1 , K1)
      DO 10 K2 = 1 , 3
      EINV2 = EINV2+E(K1 , K2)**2
      DO 10 K3 = 1 , 3
   10 EINV3 = EINV3+E(K1 , K2)*E(K2 , K3)*E(K3 , K1)
C
      EINV2 = 0.5*EINV2
      EINV3 = 0.3333333333333333D0*EINV3
C
C --- CALCULATION OF THE DETERMINANT OF THE DEFORMATION JACOBIAN
C
      C23 = 0.6666666666666666D0
      DJAC1=1.+2.*EINV1*(1.+EINV1+C23*EINV1**2)-4.*EINV2*(1.+2.*EINV1)
```

```
     #+8.*EINV3
      DJAC = DJAC1**(0.5)
C
C --- CALCULATION OF THE MATRIX <RR>
C
      DO 30 K1 = 1 , 3
      DO 30 K2 = 1 , K1
      CST1 = 0.
      DO 20 K3 = 1 , 3
   20 CST1 = CST1+E(K1 , K3)*E(K3 , K2)
      DELTA = DEL(K1 , K2)
      RR(K1 , K2) = 2.*(DELTA*(EINV1-2.*EINV2+EINV1**2)-(1.+2.*EINV1)*
     #E(K1 , K2)+2.*CST1+0.5*DELTA)/DJAC
      RR(K2 , K1) = RR(K1 , K2)
   30 CONTINUE

      RETURN
      END
```

```
C
C  ======================== Y I E L D ==============================
C
      SUBROUTINE YIELD(S,C,Z,WORK,DJAC,C3,SY,F,F1,F2)
C
C =================================================================
C I                                                               I
C I   P R O G R A M:                                              I
C I                                                               I
C I     'YIELD' CALCULATES THE VALUE OF THE YIELD FUNCTION.       I
C I     THE PROGRAMMED YIELD FUNCTION IS AN EXTENDED FORM OF THE  I
C I     MISES YIELD CRITERION. THIS YIELD FUNCTION IS THE         I
C I     EQUIVALENT LAGRANGIAN FORMULATION OF THE EULERIAN VON MISES I
C I     TYPE YIELD CRITERION.                                     I
C I                                                               I
C I     THE YIELD FUNCTION HAS THE FOLLOWING FORM:                I
C I                                                               I
C I        F =  F1 + F2 + C3*F3 - SY**2                           I
C I                                                               I
C I     C3        IS THE ISOTROPIC WORK HARDENING COEFFICIENT     I
C I     SY        IS THE YIELD STRESS IN SIMPLE TENSION TEST/SQRT(3) I
C I     F1        IS THE PART WHICH ACCOUNTS FOR ISOTROPIC HARDENING.I
C I     F2        IS THE PART WHICH ACCOUNTS FOR THE KINEMATIC WORK I
C I                 HARDENING.                                    I
C I     F3        IS THE NEGATIVE OF THE PLASTIC WORK.            I
C I                                                               I
C I   A R G U M E N T   L I S T:                                  I
C I                                                               I
C I     S(I,J)  = SECOND PIOLA KIRCHOFF STRESS TENSOR             I
C I     C(I,J)  = GREEN'S TENSOR                                  I
C I     Z(I,J)  = SHIFT STRESS TENSOR                             I
C I     WORK    = PLASTIC WORK                                    I
C I     DJAC    = DETERMINANT OF DEFORMATION JACOBIAN             I
C I     C3      = MATERIAL PARAMETER FOR ISOTROPIC HARDENING      I
C I     SY      = INITIAL YIELD STRESS DIVIDED BY SQUARE ROOT OF 3 I
C I     F       = VALUE OF THE YIELD FUNCTION                     I
C I     F1      = VALUE OF THE F1 PART OF THE YIELD FUNCTION      I
C I     F2      = VALUE OF F2 PART OF THE YIELD FUNCTION          I
C I                                                               I
C =================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/PLAST3/TEMP1(3,3),TEMP2(3,3),CST1,CST2
      DIMENSION S(3,3),C(3,3),Z(3,3)
      DATA C12,C13,C16/0.5D0,0.3333333333333333D0,0.1666666666666666D0/
C
      DJACO = 1./DJAC**2
      CST1 = 0.
      CST2 = 0.
      CST3 = 0.
      CST4 = 0.
      CST5 = 0.
```

```
C
      DO 20 K2 = 1 , 3
      DO 20 K1 = 1 , 3
      CONST1 = 0.
      CONST2 = 0.
      DO 10 K4 = 1 , 3
      DO 10 K3 = 1 , 3
      CST = C(K1 , K3)*C(K2 , K4)
      CONST1 = CONST1 + S(K3 , K4)*CST
      CONST2 = CONST2 + Z(K3 , K4)*CST
   10 CONTINUE
C
      CST1 = CST1 + S(K1 , K2)*C(K1 , K2)
      CST2 = CST2 + Z(K1 , K2)*C(K1 , K2)
      CST3 = CST3 + S(K1 , K2)*CONST1
      CST4 = CST4 + S(K1 , K2)*CONST2
      CST5 = CST5 + Z(K1 , K2)*CONST2
      TEMP1(K1 , K2) = CONST1
      TEMP2(K1 , K2) = CONST2
   20 CONTINUE
C
      F1 = DJACO*(C12*CST3 - C16*CST1**2)
      F2 = DJACO*(C13*CST1*CST2 - CST4 + C12*CST5 - C16*CST2**2)
      F3 = -WORK
      F = F1 + F2 + C3*F3 - SY**2
C
      RETURN
      END
```

```
C
C ======================== F D E R ==================================
C
      SUBROUTINE FDER(S,C,Z,DJAC,F1,F2,C3)
C
C ===================================================================
C I                                                                 I
C I   P R O G R A M:                                                I
C I                                                                 I
C I     'FDER' CALCULATES THE DERIVATIVE OF "F" WITH RESPECT TO     I
C I     <STRESS>, <STRAIN>, JACOBIAN, AND PLASTIC WORK.WKC          I
C I                                                                 I
C I   A R G U M E N T   L I S T:                                    I
C I                                                                 I
C I     S(I,J)  = SECOND PIOLA KIRCHOFF STRESS TENSOR               I
C I     C(I,J)  = GREEN'S TENSOR                                    I
C I     Z(I,J)  = SHIFT STRESS TENSOR                               I
C I     DJAC    = DETERMINANT OF DEFORMATION JACOBIAN               I
C I     F1      = VALUE OF THE F1 PART OF THE YIELD FUNCTION        I
C I     F2      = VALUE OF F2 PART OF THE YIELD FUNCTION            I
C I     C3      = MATERIAL PARAMETER FOR ISOTROPIC HARDENING        I
C I                                                                 I
C ===================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/FDER1/FJ,FK,FS(3,3),FE(3,3),FZ(3,3)
      COMMON/PLAST3/TEMP1(3,3),TEMP2(3,3),CST1,CST2
      DIMENSION S(3,3),C(3,3),Z(3,3)
      DATA CONST/0.3333333333333333D0/
C
      DJACO = 1./DJAC**2
C
      CST1 = CONST*CST1
      CST2 = CONST*CST2
C
      DO 30 K1 = 1 , 3
      DO 30 K2 = 1 , K1
      CST3 = 0.
      CST5 = 0.
      CST6 = 0.
      DO 20 K3 = 1 , 3
      DO 20 K4 = 1 , 3
      C34 = C(K3 , K4)
      SC14 = C34*S(K3 , K1)
      CST3 = CST3 + S(K4 , K2)*SC14
      CST5 = CST5 + Z(K3 , K1)*Z(K4 , K2)*C34
      CST6 = CST6 + Z(K4 , K2)*SC14
   20 CONTINUE
C
C --- F1S(I,J) =  PARTIAL DERIVATIVE OF F1 WRT <STRESS>
C --- F2S(I,J) =  PARTIAL DERIVATIVE OF F2 WRT <STRESS>
C --- F1E(I,J) =  PARTIAL DERIVATIVE OF F1 WRT <STRAIN>
```

```
C --- F2E(I,J) =  PARTIAL DERIVATIVE OF F2 WRT <STRAIN>
C
      CST4 = TEMP1(K1 , K2)
      CST7 = TEMP2(K1 , K2)
      C12 = C(K1 , K2)
      S12 = S(K1 , K2)
      F1S = CST4 - CST1*C12
      F1E = CST3 - CST1*S12
      F2S = CST2*C12 - CST7
      F2Z = CST7 - CST4 + (CST1 - CST2)*C12
      F2E = CST5 - 2.*CST6 + (CST1 - CST2)*Z(K1 , K2) + CST2*S12
C
C --- FS(I,J) = DERIVATIVE OF F WRT <STRESS>
C --- FE(I,J) = DERIVATIVE OF F WRT <STRAIN>
C --- FZ(I,J) = DERIVATIVE OF F WRT <SHIFT TENSOR>
C
      FS(K1 , K2) = DJACO*(F1S + F2S)
      FE(K1 , K2) = 2.*DJACO*(F1E + F2E)
      FZ(K1 , K2) = F2Z*DJACO
      FS(K2 , K1) = FS(K1 , K2)
      FE(K2 , K1) = FE(K1 , K2)
      FZ(K2 , K1) = FZ(K1 , K2)
 30   CONTINUE
C
C --- F1J =  PARTIAL DERIVATIVE OF F1 WRT JACOBIAN
C --- F2J =  PARTIAL DERIVATIVE OF F2 WRT JACOBIAN
C --- FJ  = DERIVATIVE OF F WRT JACOBIAN
C --- FK  = DERIVATIVE OF F WRT PLASTIC WORK
C
      F1J = -2.*F1/DJAC
      F2J = -2.*F2/DJAC
      FJ  = F1J + F2J
      FK  = -C3
C
      RETURN
      END
```

```
C
C ======================= E L P L D ===================================
C
      SUBROUTINE ELPLD(S,Z,RR,DJAC,BETA,DEN,ICODE)
C
C ====================================================================
C I                                                                  I
C I   P R O G R A M:                                                 I
C I                                                                  I
C I     'ELPLD' PERFORMS THE FOLLOWING FUNCTIONS DEPENDING ON THE    I
C I     VALUE OF PARAMETER "ICODE".                                  I
C I                                                                  I
C I     1) ICODE = 0; IT EVALUATES THE ELASTOPLASTIC STRESS-STRAIN   I
C I                   STIFFNESS MATRIX.                              I
C I     2) ICODE > 0; IT EVALUATES THE MATRICES NEEDED TO CALCULATE  I
C I                   LAMMDA-DOT AND MU-DOT.                         I
C I                                                                  I
C I                                                                  I
C I   A R G U M E N T   L I S T:                                     I
C I                                                                  I
C I     S(I,J)  = SECOND PIOLA KIRCHOFF STRESS TENSOR                I
C I     Z(I,J)  = SHIFT STRESS TENSOR                                I
C I     RR(I,J) = THIS MATRIX WHEN DOTED WITH THE STRAIN             I
C I               TENSOR WILL RESULT THE INCREMENT OF THE            I
C I               JACOBIAN.                                          I
C I     DJAC    = DETERMINANT OF DEFORMATION JACOBIAN                I
C I     BETA    = MATERIAL PARAMETER FOR KINEMATIC HARDENING         I
C I     DEN     = DENOMINATOR Q                                      I
C I                                                                  I
C ====================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/FDER1/FJ,FK,FS(9),FE(9),FZ(9)
      COMMON/ADMAT1/AD(9,9)
      COMMON/ELPLD1/DEPM(9,9),ALAM(9),AMU(9)
      DIMENSION EFF(9),S(9),Z(9),RR(9)
C
      D1 = 0.
      D2 = 0.
      D3 = 0.
      DEN1 = 0.
      DEN2 = 0.
C
C
      DO 20 K1 = 1 , 9
      EFF( K1 ) = 0.
      D1 = D1 + FZ( K1 )*(S( K1 ) - Z( K1 ))
      D2 = D2 + FS( K1 )**2
      D3 = D3 + (S( K1 ) - Z( K1 ))*FS( K1 )
      DEN2 = DEN2 + S( K1 )*FS( K1 )
      DO 10 K2 = 1 , 9
   10 EFF( K1 ) = EFF( K1 ) + AD(K2 , K1)*FS( K2 )
```

```
   20  DEN1 = DEN1 + EFF( K1 )*FS( K1 )

       DEN = DEN1 - DEN2*FK/DJAC - BETA*D1*D2/D3
C
       IF (ICODE.EQ.0) THEN
C
            DO 30 K1 = 1 , 9
            CONST = EFF( K1 )/DEN
C
            DO 30 K2 = 1 , 9
            CST2 = FE( K2 ) + FJ*RR( K2 ) + EFF( K2 )
            DEPM(K1 , K2) = AD(K1 , K2) - CST2*CONST
   30       CONTINUE
C
       ELSE
            CST2 = BETA*D2/D3
            DO 40 K1 = 1 , 9
            ALAM( K1 ) = (EFF( K1 ) + FE( K1 ) + RR( K1 )*FJ)/DEN
            AMU( K1 ) = ALAM( K1 )*CST2
   40       CONTINUE
       END IF
C
       RETURN
       END
```

```
C
C ======================= C O N V E R ===============================
C
      SUBROUTINE CONVER(D4,D2,IFLAG,ITYPE)
C
C ==================================================================
C I                                                                I
C I   P R O G R A M:                                               I
C I                                                                I
C I     'CONVER' CONVERTS THE FOURTH ORDER ELASTOPLASTIC STIFFNESS  I
C I     TENSOR "D4" TO A SECOND ORDER MATRIX "D2".                 I
C I                                                                I
C I   A R G U M E N T   L I S T:                                   I
C I                                                                I
C I     D4      = FOURTH ORDER TENSOR TO BE CONVERTED              I
C I     D2      = THE RESULTING SECOND ORDER TENSOR                I
C I     IFLAG   = ANALYSIS TYPE CODE                               I
C I                1; PLANE STRESS                                 I
C I                2; PLANE STRAIN                                 I
C I                3; AXISYMMETRIC                                 I
C I     ITYPE   = ELEMENT TYPE                                     I
C I                                                                I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      DIMENSION D4(3,3,3,3),D2(6,6)

C     D2 = THE SECOND ORDER STIFFNESS MATRIX

      IF (ITYPE.LT.300) THEN
           D2(1,1) = D4(1,1,1,1)
           D2(1,2) = D4(1,1,2,2)
           D2(1,3) = D4(1,1,1,2)
           D2(1,4) = D4(1,1,3,3)
           D2(2,1) = D4(2,2,1,1)
           D2(2,2) = D4(2,2,2,2)
           D2(2,3) = D4(2,2,1,2)
           D2(2,4) = D4(2,2,3,3)
           D2(3,1) = D4(1,2,1,1)
           D2(3,2) = D4(1,2,2,2)
           D2(3,3) = D4(1,2,1,2)
           D2(3,4) = D4(1,2,3,3)
           D2(4,1) = D4(3,3,1,1)
           D2(4,2) = D4(3,3,2,2)
           D2(4,3) = D4(3,3,1,2)
           D2(4,4) = D4(3,3,3,3)
           IF(IFLAG.NE.1) GO TO 20
           CST1 = D4(3,3,1,1)/D4(3,3,3,3)
           CST2 = D4(3,3,2,2)/D4(3,3,3,3)
           CST3 = (D4(3,3,1,2)+D4(3,3,2,1))/D4(3,3,3,3)/2.
           D2(1,1) = D2(1,1)-CST1*D4(1,1,3,3)
           D2(1,2) = D2(1,2)-CST2*D4(1,1,3,3)
```

```
      D2(1,3) = D2(1,3)-CST3*D4(1,1,3,3)
      D2(2,1) = D2(2,1)-CST1*D4(2,2,3,3)
      D2(2,2) = D2(2,2)-CST2*D4(2,2,3,3)
      D2(2,3) = D2(2,3)-CST3*D4(2,2,3,3)
      D2(3,1) = D2(3,1)-CST1*D4(1,2,3,3)
      D2(3,2) = D2(3,2)-CST2*D4(1,2,3,3)
      D2(3,3) = D2(3,3)-CST3*D4(1,2,3,3)
      DO 10 K1 = 1,4
      D2(4,K1) = 0.
10    D2(K1,4) = 0.
   ELSE
      D2(1,1) = D4(1,1,1,1)
      D2(1,2) = D4(1,1,2,2)
      D2(1,3) = D4(1,1,3,3)
      D2(1,4) = D4(1,1,1,2)
      D2(1,5) = D4(1,1,2,3)
      D2(1,6) = D4(1,1,1,3)
      D2(2,1) = D4(2,2,1,1)
      D2(2,2) = D4(2,2,2,2)
      D2(2,3) = D4(2,2,3,3)
      D2(2,4) = D4(2,2,1,2)
      D2(2,5) = D4(2,2,2,3)
      D2(2,6) = D4(2,2,1,3)
      D2(3,1) = D4(3,3,1,1)
      D2(3,2) = D4(3,3,2,2)
      D2(3,3) = D4(3,3,3,3)
      D2(3,4) = D4(3,3,1,2)
      D2(3,5) = D4(3,3,2,3)
      D2(3,6) = D4(3,3,1,3)
      D2(4,1) = D4(1,2,1,1)
      D2(4,2) = D4(1,2,2,2)
      D2(4,3) = D4(1,2,3,3)
      D2(4,4) = D4(1,2,1,2)
      D2(4,5) = D4(1,2,2,3)
      D2(4,6) = D4(1,2,1,3)
      D2(5,1) = D4(2,3,1,1)
      D2(5,2) = D4(2,3,2,2)
      D2(5,3) = D4(2,3,3,3)
      D2(5,4) = D4(2,3,1,2)
      D2(5,5) = D4(2,3,2,3)
      D2(5,6) = D4(2,3,1,3)
      D2(6,1) = D4(1,3,1,1)
      D2(6,2) = D4(1,3,2,2)
      D2(6,3) = D4(1,3,3,3)
      D2(6,4) = D4(1,3,1,2)
      D2(6,5) = D4(1,3,2,3)
      D2(6,6) = D4(1,3,1,3)
   END IF
20 RETURN
   END
```

APPENDIX D

CONTACT BOUNDARY SIMULATION MODULE

```
      SUBROUTINE BOUND(IDOF,NNDF,NINODE,ICODE,RFACT,IOUT)
C
C ==================================================================
C I                                                                I
C I   P R O G R A M:                                               I
C I                                                                I
C I     'BOUND' CHECKS THE MOTION OF ROLLERS ON CURVED             I
C I     BOUNDARIES AND INSURES THAT THE ROLLERS STAY ON THE BOUNDARY I
C I     BY DETERMINING THE APPROPRIATE DISPLACEMENT CORRECTIONS.    I
C I                                                                I
C I   A R G U M E N T    L I S T:                                  I
C I                                                                I
C I     IDOF(I) = THE ARRAY CONTAINING THE D.O.F. NUMBERS          I
C I     NNDF    = NUMBER OF NODAL DEGREES OF FREEDOM               I
C I     NINODE  = NUMBER OF INTERFACE NODES                        I
C I     ICODE   = RETURN CODE PASSED TO THE CALLING ROUTINE        I
C I               =0; NO CHANGE IN THE D.O.F. NUMBERS              I
C I               =1; RECALCULATION OF THE 'IDOF' ARRAY IS NEEDED  I
C I     RFACT   = REDUCTION FACTOR FOR OUTOMATIC SUBINCREMENTATION  I
C I     IOUT    = OUTPUT DEVICE NUMBER                             I
C I                                                                I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 X,Y,Z
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/MAIN4/RE(8000)
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
      COMMON/INPUT7/RIT(8000),RINC(8000),UINC(8000)
      COMMON/INPUTD/COSTX(300),COSTY(300),COSTZ(300)
      COMMON/INPUTE/ISPB(4000)
      COMMON/INPUTI/INTFAC(500)
      COMMON/CONTR1/INCREM,NIT
      DIMENSION IDOF( 1 )
C
      ICODE = 0
C
C --- NINODE = NUMBER OF INTERFACE NODES
C
      DO 100 K1 = 1 , NINODE
      IRLS = 0
      NODE = INTFAC( K1 )
      ID1 = NNDF*(NODE - 1) + 1
      ID2 = ID1 + 1
```

```
C
C        YNODE = FINAL Y-COORD OF THE NODE
C
      XNODE = X( NODE ) + UTOTAL( ID1 )
      YNODE = Y( NODE ) + UTOTAL( ID2 )
C
C --- CHECK TO SEE IF THE NODE IS IN CONTACT WITH THE BOUNDARY
C
C --- FIND THE POINT ON THE DIE FOR A NORMAL RETURN CORRECTION.
C --- THE DIE IS MODELED USING THE HERMITE PARAMETRIC CURVE.
C --- SUBROUTINE MULLER FINDS THE PARAMETER 'T'.
C
      CALL MULLER(XNODE,YNODE,T,NCURVE,IRET,IOUT)
      IF (IRET.GE.2) THEN
           WRITE(IOUT , 1001)
           STOP
      ELSE IF (IRET.EQ.1) THEN
C
C --- SUBROUTINE HERMXY FINDS THE 'X' AND 'Y' COORDINATES OF THE
C --- RETURN POINT ON THE DIE.
C
           CALL XYPRIM(T,XP,YP,NCURVE)
           CALL HERMXY(T,XFINAL,YFINAL,NCURVE)
           DX = XFINAL - XNODE
           DY = YFINAL - YNODE
           IF (IDOF( ID1 ).LE.0) THEN
                RDOT = -YP*RE( ID1 ) + XP*RE( ID2 )
                DOT = -1.
           ELSE
                RDOT = -1.
                DOT = -YP*DX + XP*DY
           END IF
           IF (RDOT.GE.0.0.OR.DOT.GE.0.0) THEN
C
C --- CHANGE THE NEGATIVE "ISPB" ADDRESSES TO POSITIVE SO THAT
C     THEY ARE RECOGNIZED IN THE SOLUTION PROCESS
C
                R = DSQRT(DX**2 + DY**2)
                RP = DSQRT(XP**2 + YP**2)
                IF (ISPB( NODE ).LT.0) THEN
                     ISPB( NODE ) = - ISPB( NODE )
                ELSE IF(ISPB( NODE ).EQ.0) THEN
                     WRITE(IOUT , 1000) NODE
                     STOP
                END IF
C
C --- FIND THE DIRECTION COSINES OF THE ROLLERS ON THE DIE
C
                COSTX(ISPB( NODE )) = -YP/RP
                COSTY(ISPB( NODE )) = XP/RP
                 IF (IDOF( ID1 ).GT.0) THEN
                      ICODE = 1
```

```
                  IDOF( ID1 ) = -1
                  IDOF( ID2 ) = 1
             ELSE
                  IDOF( ID1 ) = -1
             END IF
C
C --- IMPOSE THE APPROPRIATE DISPLACEMENT FOR THE NORMAL RETURN
C     CORRECTION DURING THE NEXT ITERATION.
C
            DOT = -YP*DX + XP*DY
            IF (DOT.GE.0) THEN
                 UINC( ID1 ) = R/RFACT
            ELSE
                 UINC( ID1 ) = -R/RFACT
            END IF
       ELSE
            IRLS = 1
       END IF
  END IF
  IF (IRET.EQ.0.OR.IRLS.EQ.1) THEN
C
C --- IF THE NODE EXITS THE DIE RELEASE IT AND SET ICODE = 1 TO
C     THE CALLING ROUTINE RECALCULATES THE "IDOF" ARRAY
C
       UINC( ID1 ) = 0.
       IF(ISPB( NODE ).GT.0) ISPB( NODE ) = -ISPB( NODE )
       IF(IDOF( ID1 ).LE.0) THEN
            ICODE = 1
            IDOF( ID1 ) = 1
            RINC( ID1 ) = -RE( ID1 )/RFACT
            RINC( ID2 ) = -RE( ID2 )/RFACT
       ELSE
            RINC( ID1 ) = 0.
            RINC( ID2 ) = 0.
       END IF
C      RIT( ID1 ) = 0.
C      RIT( ID2 ) = 0.
  END IF
 100  CONTINUE
C
      RETURN
 1000 FORMAT(1X,'>>>>>>> PROGRAM STOPED IN ROUTINE "BOUND" DUE TO A'/
     1 9X,'ZERO ISPB FOR INTERFACE NODE ',I4)
 1001 FORMAT(1X,'>>>>>>> PROGRAM STOPED IN ROUTINE "BOUND" DUE TO '/
     1 9X,'EXISTANCE OF MULTIPLE CONTACT POINTS')
      END
```

```
C
C =================== M U L L E R ===================================
C
      SUBROUTINE MULLER(XD,YD,TF,NCRV,IRET,IOUT)
C
C ==================================================================
C I                                                                I
C I   P R O G R A M:                                               I
C I                                                                I
C I     'MULLER' PERFORMES THE FOLLOWING FUNCTIONS:                I
C I                                                                I
C I     1) IT USES ZONING TO DETERMINE THE BOUNDARY CURVE WHICH IS I
C I        CLOSEST TO THE INTERFACE NODE.                          I
C I                                                                I
C I     2) IT USES THE MULLERS METHOD TO SOLVE A FIFTH ORDER       I
C I        POLYNOMIAL WHICH DETERMINES PARAMETER "T" WHICH IS      I
C I        USED TO DETERMINE THE CLOSEST POINT ON A CURVE TO THE   I
C I        INTERFACE NODE.                                         I
C I                                                                I
C I   A R G U M E N T   L I S T:                                   I
C I                                                                I
C I     XD      = X-COORDINATE OF THE INTERFACE NODE               I
C I     YD      = Y-COORDINATE OF THE INTERFACE NODE               I
C I     TF      = FINAL VALUE OF PARAMETER "T"                     I
C I     NCRV    = NUMBER OF BOUNDARY CURVES TO BE CHECKED          I
C I     IRET    = CONDITION CODE RETURNED TO THE CALLING ROUTINE   I
C I               =0; NO VALUE FOR "T" IF FOUND, NODE IS FREE      I
C I               =1; A NORMAL FROM THE NODE TO ONE OF THE CURVES  I
C I                   MAY BE DRAWN INDICATING POSSIBILITY OF CONTACT I
C I               >1; TOO MANY NORMALS DETECTED, POSSIBLE ERROR    I
C I                                                                I
C I     RFACT   = REDUCTION FACTOR FOR OUTOMATIC SUBINCREMENTATION I
C I     IOUT    = OUTPUT DEVICE NUMBER                             I
C I                                                                I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/BOUND1/NCURVS
      COMMON/BOUND2/XZL(6),XZR(6),YZB(6),YZT(6)
      COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
      COMMON/CONST/C0(6),C1(6),C2(6),C3(6),C4(6),C5(6)
      DIMENSION ROOT(5)
C
      TOL = .000000001
      IRET = 0
C
C --- IRET = 0; NO CONTACT WITH BOUNDARIES
C              1; ONE CONTACT POINT DETECTED
C             <1; TOO MANY CONTACT POINTS (PROBABLE ERROR)
C
      DO 60 NCURVE = 1 , NCURVS
      IF (XD.GT.XZR(NCURVE).OR.XD.LT.XZL(NCURVE)) THEN
```

```
           GO TO 60
      ELSE IF(YD.GT.YZT(NCURVE).OR.YD.LT.YZB(NCURVE)) THEN
           GO TO 60
      END IF
C
      D5 = C5( NCURVE )
      D4 = C4( NCURVE )
      D3 = C3( NCURVE )
      D2 = C2( NCURVE ) + 3.*(XD*GX(1 , NCURVE) + YD*GY(1 , NCURVE))
      D1 = C1( NCURVE ) + 2.*(XD*GX(2 , NCURVE) + YD*GY(2 , NCURVE))
      D0 = C0( NCURVE ) + XD*GX(3 , NCURVE) + YD*GY(3 , NCURVE)
C
      NROOT = 0
   10 T1 = 0.
      T2 = 1.
      T0 = 0.5
C
      F1 = D0
      F2 = D5 + D4 + D3 + D2 + D1 + D0
C
      DO 20 K1 = 1 , NROOT
      F1 = F1/(-ROOT( K1 ))
      F2 = F2/(1. - ROOT( K1 ))
   20 CONTINUE
C
      DO 40 K1 = 1 , 20
      F0 = ((((D5*T0 + D4)*T0 + D3)*T0 + D2)*T0 + D1)*T0 + D0
C
      DO 30 K2 = 1 , NROOT
   30 F0 = F0/(T0 - ROOT( K2 ))
C
      H2 = T0 - T2
      H1 = T1 - T0
      H1S = H1**2
      GAM = H2/H1
      A = (GAM*F1 - F0*(1.+GAM) + F2)/(GAM*H1S*(1 + GAM))
      B = (F1 - F0 - A*H1S)/H1
C
      SQTRM = B**2 - 4.*A*F0
      IF (SQTRM.LT.0.) GO TO 60
      IF (B.LE.0.) THEN
           D = -DSQRT( SQTRM )
      ELSE
           D = DSQRT( SQTRM )
      END IF
C
      T = T0 - 2.*F0/(B+D)
C
      IF (DABS(T-T0).LE.TOL) GO TO 50
C
      IF (T.GT.T0.AND.T.NE.T2) THEN
           T1 = T0
```

```
          T0 = T
          F1 = F0
     ELSE IF(T.LT.T0.AND.T.NE.T1) THEN
          T2 = T0
          T0 = T
          F2 = F0
     ELSE
          GO TO 50
     END IF
 40  CONTINUE
C
 50  IF (T.GT.1..OR.T.LT.0.) THEN
          IF (NROOT.LT.4) THEN
               NROOT = NROOT + 1
               ROOT( NROOT ) = T
               GO TO 10
          END IF
     ELSE
          IRET = IRET + 1
          TF = T
          NCRV = NCURVE
          RETURN
     END IF
 60  CONTINUE
C
     RETURN
     END
```

```
C
C ==============================================================
C
      SUBROUTINE COEFIC
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/BOUND1/NCURVS
      COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
      COMMON/CONST/C0(6),C1(6),C2(6),C3(6),C4(6),C5(6)
C
      DO 10 K1 = 1 , NCURVS
C
      C0( K1 ) = 0.
      C1( K1 ) = 0.
      C2( K1 ) = 0.
      C3( K1 ) = 0.
      C4( K1 ) = 0.
      C5( K1 ) = 0.
C
      C0( K1 ) = C0( K1 ) - GX(3 , K1)*GX(4 , K1)
     1                    - GY(3 , K1)*GY(4 , K1)
      C1( K1 ) = C1( K1 ) - GX(3 , K1)**2 - 2.*GX(2 , K1)*GX(4 , K1)
     1                    - GY(3 , K1)**2 - 2.*GY(2 , K1)*GY(4 , K1)
      C2( K1 ) = C2( K1 ) -3.*GX(1,K1)*GX(4,K1)-3.*GX(2,K1)*GX(3,K1)
     1                    -3.*GY(1,K1)*GY(4,K1)-3.*GY(2,K1)*GY(3,K1)
      C3( K1 ) = C3( K1 ) - 2.*GX(2 , K1)**2 - 4.*GX(1 , K1)*GX(3 , K1)
     1                    - 2.*GY(2 , K1)**2 - 4.*GY(1 , K1)*GY(3 , K1)
      C4( K1 ) = C4( K1 ) - 5.*GX(1 , K1)*GX(2 , K1)
     1                    - 5.*GY(1 , K1)*GY(2 , K1)
      C5( K1 ) = C5( K1 ) - 3.*GX(1 , K1)**2
     1                    - 3.*GY(1 , K1)**2
   10 CONTINUE
C
      RETURN
      END
```

```
C
C ==========================================================================
C
      SUBROUTINE HERMIT
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 LX,LY
      COMMON/BOUND1/NCURVS
      COMMON/POINTS/LX(4 , 6),LY(4 , 6)
      COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
C
      DO 20 K2 = 1 , NCURVS
      DO 20 K1 = 1 , 4
      GX(K1 , K2) = 0.
      GY(K1 , K2) = 0.
      DO 20 K3 = 1 , 4
      GX(K1 , K2) = GX(K1 , K2) + H(K1 , K3)*LX(K3 , K2)
   20 GY(K1 , K2) = GY(K1 , K2) + H(K1 , K3)*LY(K3 , K2)
C
      RETURN
      END
```

```
C
C ======================================================================
C
      SUBROUTINE HERMXY(T,X,Y,NCURVE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 M
      COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
      DIMENSION M(4)
C
      M( 4 ) = 1.
      M( 3 ) = T
      M( 2 ) = T*M( 3 )
      M( 1 ) = T*M( 2 )
C
      X = 0.
      Y = 0.
      DO 20 K1 = 1 , 4
      X = X + M( K1 ) * GX(K1 , NCURVE)
   20 Y = Y + M( K1 ) * GY(K1 , NCURVE)
C
      RETURN
      END
```

```
C
C ====================================================================
C
      SUBROUTINE XYPRIM(T,XP,YP,NCURVE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 MPRIM
      COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
      DIMENSION MPRIM(4)
C
      MPRIM( 4 ) = 0.
      MPRIM( 3 ) = 1.
      MPRIM( 2 ) = 2.*T
      MPRIM( 1 ) = 3.*T**2
C
      XP = 0.
      YP = 0.
      DO 20 K1 = 1 , 4
      XP = XP + MPRIM( K1 ) * GX(K1 , NCURVE)
   20 YP = YP + MPRIM( K1 ) * GY(K1 , NCURVE)
C
      RETURN
      END
```

APPENDIX E

ELASTICITY MODULE

```
      SUBROUTINE ELAST(ITYPE,MATNUM,IFLAG,IOUT,ICODE)
C
      IF (ICODE.EQ.0) THEN
           CALL DELAST(ITYPE,MATNUM,IFLAG)
      ELSE
           CALL STRSTN(ITYPE,MATNUM,IFLAG,IOUT)
      END IF
      RETURN
      END
C
C ======================= S T R S T N ==================================
C
      SUBROUTINE STRSTN(ITYPE,MATNUM,IFLAG,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*105 DUMMY
      COMMON/UTIL1/STRESS(6),STRAIN(6),DUMMY
      COMMON/MATER1/DEP(6,6)
      COMMON/ELSTR1/STRN(6)
      COMMON/ELSTR2/STRS(6)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/CONTR1/INCREM,NIT
      DIMENSION DE(6),DS(6)
C
      IF (ITYPE.GT.300) THEN
           IEND = 6
      ELSE
           IEND = 4
      END IF
C
      IF (INCREM.GT.1) THEN
         CALL IOGET(LDEV1,96,'(A96)',5)
      ELSE
         DO 50 K1 = 1 , IEND
         STRESS( K1 ) = 0.
 50      STRAIN( K1 ) = 0.
      END IF
C
      DO 100 K1 = 1 , IEND
 100  DE( K1 ) = STRN( K1 ) - STRAIN( K1 )
C
      CALL DELAST(ITYPE,MATNUM,IFLAG)
C
      DO 300 K1= 1 , IEND
      S = 0.
      DO 200 K2 = 1 , IEND
 200  S = S + DEP(K1 , K2)*DE( K2 )
```

```
 300  DS( K1 ) = S
C
      DO 400 K1 = 1 , IEND
      STRAIN( K1 ) = STRN( K1 )
      STRESS( K1 ) = STRESS( K1 ) + DS( K1 )
 400  STRS( K1 ) = STRESS( K1 )
C
      CALL IOPUT(LDEV2,96,'(A96)',5)
C
      RETURN
 1000 FORMAT(2A48)
      END
```

```
C
C ===================== D E L A S T =================================
C
      SUBROUTINE DELAST(ITYPE,MATNUM,IFLAG)
C
C ==================================================================
C I                                                                I
C I    PROGRAM 'ELAST'EVALUATES THE STRESS-STRAIN STIFFNESS MATRIX  I
C I    FOR ISOTROPIC OR ORTHOTROPIC ELASTIC MATERIALS               I
C I                                                                I
C I    C O M M O N     B L O C K S                                 I
C I                                                                I
C I                                                                I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NUX,NUY,NUZ
      COMMON/MATER1/DEP(6,6)
      COMMON/INPUT5/NUX(10),NUY(10),NUZ(10),EX(10),EY(10),EZ(10),
     1 P1X(10),P1Y(10),P1Z(10),P2X(10),P2Y(10),P2Z(10)
C
      IF (ITYPE.GT.300) THEN
C
      DO 100 K2 = 1 , 6
      DO 100 K1 = 1 , 6
 100  DEP(K1 , K2) = 0.
C
      G = 0.5*EX( MATNUM )/(1. + NUX( MATNUM ))
      CST1 = 2.*G*(1. - NUX( MATNUM ))/(1. - 2.*NUX( MATNUM ))
      CST2 = 2.*G*NUX( MATNUM )/(1. - 2.*NUX( MATNUM ))
C
      DEP(1 , 1) = CST1
      DEP(2 , 2) = CST1
      DEP(3 , 3) = CST1
      DEP(4 , 4) = G
      DEP(5 , 5) = G
      DEP(6 , 6) = G
      DEP(1 , 2) = CST2
      DEP(1 , 3) = CST2
      DEP(2 , 1) = CST2
      DEP(2 , 3) = CST2
      DEP(3 , 1) = CST2
      DEP(3 , 2) = CST2
C
      ELSE
C
      DO 200 K2 = 1 , 4
      DO 200 K1 = 1 , 4
 200  DEP(K1 , K2) = 0.
C
C        PLAIN STRESS
C
```

```
      IF (IFLAG.EQ.1) THEN
          DEP(1 , 1) = EX( MATNUM )/(1. - NUX( MATNUM )**2)
          DEP(2 , 2) = DEP(1 , 1)
          DEP(3 , 3) = EX( MATNUM )*0.5/(1. + NUX( MATNUM ))
          DEP(1 , 2) = NUX( MATNUM )*DEP(1 , 1)
          DEP(2 , 1) = DEP(1 , 2)
C
C       AXISYMMETRIC AND PLANE STRAIN
C
      ELSE
          CST1 = EX( MATNUM )/(1. + NUX(MATNUM))/(1. - 2.*NUX(MATNUM))
          CST2 = CST1*NUX( MATNUM )
          DEP(1 , 1) = CST1 - CST2
          DEP(2 , 2) = DEP(1 , 1)
          DEP(3 , 3) = EX( MATNUM )*0.5/(1. + NUX( MATNUM ))
          DEP(4 , 4) = DEP(1 , 1)
          DEP(1 , 2) =  CST2
          DEP(2 , 1) =  CST2
          DEP(1 , 4) =  CST2
          DEP(4 , 1) =  CST2
          DEP(2 , 4) =  CST2
          DEP(4 , 2) =  CST2
C
      END IF
      END IF
C
      RETURN
      END
```

APPENDIX F

ELEMENT LIBRARY MODULE

```
C
C ======================== G A U S S ==================================
C
      SUBROUTINE GAUSS(NIP,W,GCOORD)
C
C =====================================================================
C I                                                                   I
C I     SUBPROGRAM GAUSS STORES THE COORDINATES XI AND ETA OF THE     I
C I     NUMERICAL INTEGRATION POINTS AND THEIR WEIGHTING FUNCTIONS    I
C I     FOR THE FOUR POINT AND THE NINE POINT INTEGRATION.            I
C I                                                                   I
C I         NIP       = NUMBER OF THE INTEGRATION POINTS              I
C I         W(I)      = WEIGTH FUNCTION                               I
C I         GCOORD(I) = COORDINATES OF THE GAUSSIAN POINTS            I
C I                     FROM THE NEGATIVE TO POSITIVE                 I
C I                                                                   I
C I     FIRST DEVELOPED: 08-26-1988                                   I
C I     LAST UPDATE:  08-26-1988                                      I
C I     BY:  M. FOROOZESH                                             I
C =====================================================================
      REAL*8 W,GCOORD
      DIMENSION W(4),GCOORD(4)
C
      IF (NIP.EQ.1) THEN
         W( 1 ) = 2.0D0
         GCOORD( 1 ) = 0.0D0
C
      ELSE IF (NIP.EQ.2) THEN
         W( 1 ) = 1.0D0
         W( 2 ) = 1.0D0
         GCOORD( 1 ) = -0.577350269189626D0
         GCOORD( 2 ) = 0.577350269189626D0
C
      ELSE IF (NIP.EQ.3) THEN
         W( 1 ) = 5.0D0/9.0D0
         W( 2 ) = 8.0D0/9.0D0
         W( 3 ) = W( 1 )
         GCOORD( 1 ) = -0.774596669241483D0
         GCOORD( 2 ) = 0.
         GCOORD( 3 ) = 0.774596669241483D0
C
      ELSE IF (NIP.EQ.4) THEN
         W( 1 ) = 0.347854845137454D0
         W( 2 ) = 0.652145154862546D0
         W( 3 ) = W( 2 )
         W( 4 ) = W( 1 )
```

```
         GCOORD( 1 ) = -0.861136311594053D0
         GCOORD( 2 ) = -0.339981043584856D0
         GCOORD( 3 ) = +0.339981043584856D0
         GCOORD( 4 ) = +0.861136311594053D0
C
      ELSE
         GO TO 10
C
      END IF
 10   RETURN
      END
C
C ========================= I R O N S ================================
C
      SUBROUTINE IRONS(A1,B6,C8,B,C,NIP,INTCOD)
C
C ====================================================================
C I                                                                  I
C I      SUBPROGRAM IRONS STORES THE COORDINATES RETURNS THE COORD.  I
C I      AND THE WEIGHT FUNCTIONS FOR THE OPTIMUM INTEGRATION        I
C I      POINTS INTRODUCED BY BRUCE M. IRONS.                        I
C I                                                                  I
C I      FOR THE DISCRIPTION OF VARIABLES REFER TO THE REFERENCE     I
C I      PUBLICATION.                                                I
C I                                                                  I
C I      FIRST DEVELOPED: 10-08-1988                                 I
C I      LAST UPDATE:  10-08-1988                                    I
C I      BY:  M. FOROOZESH                                           I
C ====================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
C
      IF (INTCOD.EQ.150) THEN
             NIP = 15
             A1 = 1.564444444444D0
             B6 = 0.3555555555556D0
             C8 = 0.5377777777778D0
             B = 1.0D0
             C = 0.674199862D0
      ELSE IF(INTCOD.EQ.151) THEN
             NIP = 15
             A1 = 0.712137436D0
             B6 = 0.686227234D0
             C8 = 0.396312395D0
             B = 0.848418011D0
             C = 0.727662441D0
      ELSE IF(INTCOD.EQ.140) THEN
             NIP = 14
             A1 = 0.0D0
             B6 = .886426593D0
             C8 = .335180055D0
             B = 0.795822426D0
             C = 0.758786911D0
```

```
      END IF
C
      RETURN
      END
```

```
C
C ===================== I S H A P E ==============================
C
      SUBROUTINE ISHAPE
C ================================================================
C I                                                              I
C I   THIS PROGRAM EVALUATES THE SHAPE FUNCTIONS, THEIR DERIVATIVES   I
C I   WITH RESPECT TO THE NATURAL COORDINATES, AND THE WEIGHT        I
C I   FUNCTIONS AT EACH INTEGRATION POINT.                         I
C I                                                              I
C I   ENTRY POINTS:                                              I
C I       ISH2DG     (FOR 2D ELEMENTS)                           I
C I       ISH3DG     (FOR 3D ELEMENTS)                           I
C I                                                              I
C I        FIRST DEVELOPED: 08-27-1988                           I
C I        LAST MODIFIED: 08-29-1988                             I
C I        BY: M. FOROOZESH                                      I
C I                                                              I
C ================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/ISHAP2/W(27)
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      DIMENSION F(20),FXI(20),FETA(20),FSI(20)
      DIMENSION WXI(4),WETA(4),WSI(4),XI(4),ETA(4),SI(4)
C
C         EVALUATE THE SHAPE FUNCTIONS OF THE 2D ISOPARAMETRIC ELEMENTS
C
      ENTRY ISH2DG(ITYPE,NNEL,IERROR)
C
C       GET THE NATURAL COORDINATES OF THE INTEGRATION POINTS
C
      CALL GAUSS(NIPXI,WXI,XI)
      CALL GAUSS(NIPETA,WETA,ETA)
      NIP = NIPXI*NIPETA
C
C       NIP = TOTAL NUMBER OF THE INTEGRATION POINTS FOR THE ELEMENT
C       IETA = ROW NUMBER OF THE GAUSSIAN POINT FROM THE BOTTOM
C       IXI = COLUMN NUMBER OF THE GAUSSIAN POINT FROM LEFT
C
      DO 10 IETA = 1 , NIPETA
      DO 10 IXI = 1 , NIPXI
C
         J = (IETA - 1)*NIPXI + IXI
         W( J ) = WXI( IXI )*WETA( IETA )
         AXI = XI( IXI )
         AETA = ETA( IETA )
         CALL ISOP2D(AXI,AETA,F,FXI,FETA,ITYPE,IERROR)
 10      CALL ISHEXT(NNEL,J,F,FXI,FETA,FSI)
C
      RETURN
C
C
```

```
C
C
C          EVALUATE THE SHAPE FUNCTIONS OF THE 3D ISOPARAMETRIC ELEMENTS
C
      ENTRY ISH3DG(ITYPE,NNEL,IERROR)
C
      CALL GAUSS(NIPXI,WXI,XI)
      CALL GAUSS(NIPETA,WETA,ETA)
      CALL GAUSS(NIPSI,WSI,SI)
      NIP = NIPXI*NIPETA*NIPSI
      I = NIPXI*NIPETA
C
      DO 20 ISI = 1 , NIPSI
      DO 20 IETA = 1 , NIPETA
      DO 20 IXI = 1 , NIPXI
         J = (ISI - 1)*I + (IETA - 1)*NIPXI + IXI
         W( J ) = WXI( IXI )*WETA( IETA )*WSI( ISI )
         AXI = XI( IXI )
         AETA = ETA( IETA )
         ASI = SI( ISI )
         CALL ISOP3D(AXI,AETA,ASI,F,FXI,FETA,FSI,ITYPE,IERROR)
 20      CALL ISHEXT(NNEL,J,F,FXI,FETA,FSI)
C
      RETURN
C
C
C
C
      ENTRY ISH3DI(ITYPE,NNEL,IERROR)
C
      CALL IRONS(A1,B6,C8,B,C,NIP,INTCOD)
      AXI = B
      AETA = 0.
      ASI = 0.
      DO 40 J = 1 , 6
         W( J ) = B6
         CALL ISOP3D(AXI,AETA,ASI,F,FXI,FETA,FSI,ITYPE,IERROR)
         CALL ISHEXT(NNEL,J,F,FXI,FETA,FSI)
         A = AETA
         AETA = -AXI
         AXI  = -ASI
         ASI  = -A
 40   CONTINUE
C
      AXI = C
      AETA = C
      ASI = C
      DO 60 J = 7 , 14
         W( J ) = C8
         CALL ISOP3D(AXI,AETA,ASI,F,FXI,FETA,FSI,ITYPE,IERROR)
         CALL ISHEXT(NNEL,J,F,FXI,FETA,FSI)
         AXI  = -AXI
```

```
         A = AETA
         AETA = -ASI
         IF (J.EQ.10) AETA = ASI
         ASI  =  A
 60   CONTINUE
C
      IF (INTCOD.GE.150) THEN
         W( 15 ) = A1
         AXI = 0.
         AETA = 0.
         ASI = 0.
         CALL ISOP3D(AXI,AETA,ASI,F,FXI,FETA,FSI,ITYPE,IERROR)
         CALL ISHEXT(NNEL,15,F,FXI,FETA,FSI)
      END IF
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C =========================== I S H E X T ==============================
C
C
      SUBROUTINE ISHEXT(NNEL,J,F,FXI,FETA,FSI)
      REAL*8 N,NXI,NETA,NSI,F,FXI,FETA,FSI
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      DIMENSION F(20),FXI(20),FETA(20),FSI(20)
C
C*VDIR: ASSUME COUNT(8)
      DO 70 NN = 1 , NNEL
        N(NN , J) = F( NN )
        NXI(NN , J) = FXI( NN )
        NETA(NN , J) = FETA( NN )
        NSI(NN , J) = FSI( NN )
 70   CONTINUE
C
      RETURN
      END
```

```
C
C ========================= E L M L I B ============================
C
      SUBROUTINE ELMLIB
C
C =================================================================
C I                                                               I
C I      SUBPROGRAM ELMLIB CALCULATES THE SHAPE FUNCTIONS AND THE I
C I      PARTIAL DRIVATIVES OF THE SHAPE FUNCTIONS WRT THE LOCAL  I
C I      COORDINATES 'XI', 'ETA' AND 'SI'.                        I
C I                                                               I
C I              N(I)     = SHAPE FUNCTIONS OF THE ELEMENT        I
C I              NXI(I) = PARTIAL DRIVATIVE OF 'N' WRT 'XI'       I
C I              NETA(I) =    PARTIAL DRIVATIVE OF 'N' WRT 'ETA'  I
C I              NSI(I) = PARTIAL DERIVATIVE OF 'N' WRT 'SI'      I
C I                                                               I
C I                                                               I
C I        FIRST DEVELOPED: 08-27-1988                            I
C I        LAST MODIFIED: 08-27-1988                              I
C I        BY: M. FOROOZESH                                       I
C I                                                               I
C =================================================================
      REAL*8 XI,ETA,SI,N,NXI,NETA,NSI,XI0,ETA0,SI0
      DIMENSION N(20),NXI(20),NETA(20),NSI(20)
      COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
C
C
C
      ENTRY ISOP2D(XI,ETA,N,NXI,NETA,ITYPE,IERROR)
C
C       DRIVATIVE OF SHAPE FUNCTIONS FOR 2D ISOPARAMETRIC ELEMENTS.
C
C
      NXI(1) = -0.25*(1.- ETA)
      NXI(2) =    0.25*(1.- ETA)
      NXI(3) =    0.25*(1.+ ETA)
      NXI(4) = -0.25*(1.+ ETA)
      NETA(1) = -0.25*(1.- XI)
      NETA(2) = -0.25*(1.+ XI)
      NETA(3) =    0.25*(1.+ XI)
      NETA(4) =    0.25*(1.- XI)
C
      IF (ITYPE.EQ.219) THEN
           CST = 9.D0/32.D0
           C13 = 1.D0/3.D0
           C23 = 2.D0/3.D0
           NXI( 5 ) = CST*(1. - ETA)*(9.*XI**2 - 2.*XI - 3.)
           NXI( 6 ) = CST*(1. - ETA)*(-9.*XI**2 - 2.*XI + 3.)
           NXI( 7 ) = 0.5*(1. - ETA**2)
           NXI( 8 ) = -XI*(1 + ETA)
           NXI( 9 ) = -0.5*(1. - ETA**2)
           NXI( 1 ) = NXI( 1 )-0.5*NXI( 9 )-C23*NXI( 5 )-C13*NXI( 6 )
```

```
         NXI( 2 ) = NXI( 2 )-0.5*NXI( 7 )-C23*NXI( 6 )-C13*NXI( 5 )
         NXI( 3 ) = NXI( 3 ) - 0.5*(NXI( 7 ) + NXI( 8 ))
         NXI( 4 ) = NXI( 4 ) - 0.5*(NXI( 8 ) + NXI( 9 ))
         NETA( 5 ) = -CST*(1. - XI**2)*(1. - 3.*XI)
         NETA( 6 ) = -CST*(1. - XI**2)*(1. + 3.*XI)
         NETA( 7 ) = -ETA*(1. + XI)
         NETA( 8 ) = 0.5*(1. - XI**2)
         NETA( 9 ) = -ETA*(1. - XI)
         NETA( 1 ) = NETA( 1 )-0.5*NETA( 9 )-C23*NETA(5)-C13*NETA(6)
         NETA( 2 ) = NETA( 2 )-0.5*NETA( 7 )-C23*NETA(6)-C13*NETA(5)
         NETA( 3 ) = NETA( 3 ) - 0.5*(NETA( 7 ) + NETA( 8 ))
         NETA( 4 ) = NETA( 4 ) - 0.5*(NETA( 8 ) + NETA( 9 ))
      ELSE
         IF(ITYPE.EQ.204) GO TO 10
C
C       ADDITIONAL TERMS FOR THE FIVE NODE ISOPARAMETRIC EL.
C
         NXI( 5 ) = - XI*(1. - ETA)
         NETA( 5 ) = -0.5*(1. - XI**2)
         NXI( 1 ) = NXI( 1 ) - 0.5*NXI( 5 )
         NXI( 2 ) = NXI( 2 ) - 0.5*NXI( 5 )
         NETA( 1 ) = NETA( 1 ) - 0.5*NETA( 5 )
         NETA( 2 ) = NETA( 2 ) - 0.5*NETA( 5 )
C
         IF (ITYPE.EQ.205) GO TO 10
C
C       ADDITIONAL TERMS FOR THE EIGHT NODE ISOPARAMETRIC EL.
C
C
         NXI( 6 ) = 0.5*(1. - ETA**2)
         NXI( 7 ) = - XI*(1 + ETA)
         NXI( 8 ) = -0.50*(1. - ETA**2)
         NETA( 6 ) = -(1. + XI)* ETA
         NETA( 7 ) = 0.5*(1. - XI**2)
         NETA( 8 ) = -(1. - XI)* ETA
         NXI( 1 ) = NXI( 1 ) - 0.5*NXI( 8 )
         NXI( 2 ) = NXI( 2 ) - 0.5*NXI( 6 )
         NXI( 3 ) = NXI( 3 ) - 0.5*(NXI( 7 ) + NXI( 6 ))
         NXI( 4 ) = NXI( 4 ) - 0.5*(NXI( 7 ) + NXI( 8 ))
         NETA( 1 ) = NETA( 1 ) - 0.5*NETA( 8 )
         NETA( 2 ) = NETA( 2 ) - 0.5*NETA( 6 )
         NETA( 3 ) = NETA( 3 ) - 0.5*(NETA( 7 ) + NETA( 6 ))
         NETA( 4 ) = NETA( 4 ) - 0.5*(NETA( 7 ) + NETA( 8 ))
C
         IF(ITYPE.EQ.208) GO TO 10
C
C      ADDITIONAL TERMS FOR THE NINE NODE LAGRANGIAN ELEMENT
C
         NXI(9) = -2.*XI*(1.- ETA**2)
         NXI(1) = NXI(1)+NXI(9)/4.
         NXI(2) = NXI(2)+NXI(9)/4.
         NXI(3) = NXI(3)+NXI(9)/4.
```

```
          NXI(4) = NXI(4)+NXI(9)/4.
          NXI(5) = NXI(5)-NXI(9)/2.
          NXI(6) = NXI(6)-NXI(9)/2.
          NXI(7) = NXI(7)-NXI(9)/2.
          NXI(8) = NXI(8)-NXI(9)/2.
          NETA(9) = -2.* ETA*(1.- XI**2)
          NETA(1) = NETA(1)+NETA(9)/4.
          NETA(2) = NETA(2)+NETA(9)/4.
          NETA(3) = NETA(3)+NETA(9)/4.
          NETA(4) = NETA(4)+NETA(9)/4.
          NETA(5) = NETA(5)-NETA(9)/2.
          NETA(6) = NETA(6)-NETA(9)/2.
          NETA(7) = NETA(7)-NETA(9)/2.
          NETA(8) = NETA(8)-NETA(9)/2.
      END IF
C
C
C
      ENTRY N2D(XI,ETA,N,ITYPE,IERROR)
C
C       SHAPE FUNCTIONS  FOR 2D ISOPARAMETRIC ELEMENTS.
C
C
   10 N(1) = 0.25*(1.- XI)*(1.- ETA)
      N(2) = 0.25*(1.+ XI)*(1.- ETA)
      N(3) = 0.25*(1.+ XI)*(1.+ ETA)
      N(4) = 0.25*(1.- XI)*(1.+ ETA)
C
      IF (ITYPE.EQ.219) THEN
          CST = 9.D0/32.D0
          C13 = 1.D0/3.D0
          C23 = 2.D0/3.D0
          N( 5 ) = CST*(1. - ETA)*(1. - XI**2)*(1. - 3.*XI)
          N( 6 ) = CST*(1. - ETA)*(1. - XI**2)*(1. + 3.*XI)
          N( 7 ) = 0.5*(1. + XI)*(1. - ETA**2)
          N( 8 ) = 0.5*(1. - XI**2)*(1 + ETA)
          N( 9 ) = 0.5*(1. - XI)*(1. - ETA**2)
          N( 1 ) = N( 1 ) - 0.5*N( 9 ) - C23*N( 5 ) - C13*N( 6 )
          N( 2 ) = N( 2 ) - 0.5*N( 7 ) - C23*N( 6 ) - C13*N( 5 )
          N( 3 ) = N( 3 ) - 0.5*(N( 7 ) + N( 8 ))
          N( 4 ) = N( 4 ) - 0.5*(N( 8 ) + N( 9 ))
      ELSE
          IF(ITYPE.EQ.204) GO TO 15
C
C        ADDITIONAL TERMS FOR THE FIVE NODE ISOPARAMETRIC EL.
C
          N(5) = 0.5*(1.- XI**2)*(1.- ETA)
          N(1) = N(1)-0.5*N(5)
          N(2) = N(2)-0.5*N(5)
          IF(ITYPE.EQ.205) GO TO 15
C
C        ADDITIONAL TERMS FOR THE EIGHT NODE ISOPARAMETRIC EL.
```

```
C
          N(6) = 0.5*(1.+ XI)*(1.- ETA**2)
          N(7) = 0.5*(1.- XI**2)*(1.+ ETA)
          N(8) = 0.5*(1.- XI)*(1.- ETA**2)
          N(1) = N(1)-0.5*N(8)
          N(2) = N(2)-0.5*N(6)
          N(3) = N(3)-0.5*(N(7)+N(6))
          N(4) = N(4)-0.5*(N(7)+N(8))
C
          IF(ITYPE.EQ.208) GO TO 15
C
C       ADDITIONAL TERMS FOR THE NINE NODE LAGRANGIAN ELEMENT
C
          N(9) = (1.- ETA**2)*(1.- XI**2)
          N(1) = N(1)+N(9)/4.
          N(2) = N(2)+N(9)/4.
          N(3) = N(3)+N(9)/4.
          N(4) = N(4)+N(9)/4.
          N(5) = N(5)-N(9)/2.
          N(6) = N(6)-N(9)/2.
          N(7) = N(7)-N(9)/2.
          N(8) = N(8)-N(9)/2.
      END IF
C
   15 RETURN
C
C
C
C     SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR THE 3D ISOP. EL.
C
      ENTRY ISOP3D(XI,ETA,SI,N,NXI,NETA,NSI,ITYPE,IERROR)
C
      IF(ITYPE.EQ.308) THEN
C
      DO 20 K = 1 , 8
      CALL ELMEXT(XI,ETA,SI,K,XI0,ETA0,SI0)
      N(K)     = .125*(1. + XI0)*(1. + ETA0)*(1. + SI0)
      NXI(K) = .125*(1. + ETA0)*(1. + SI0)*XII( K )
      NETA(K)= .125*(1. + XI0)*(1. + SI0)*ETAI( K )
      NSI(K) = .125*(1. + XI0)*(1. + ETA0)*SII( K )
   20 CONTINUE
C
      ELSE IF (ITYPE.EQ.320) THEN
C
C        HEXAHYDRON SOLID ELEMENT
C        SHAPE FUNCTIONS AND THIE DERIVATIVES FOR NODES 1-8.
C
      DO 30 K = 1 , 8
      CALL ELMEXT(XI,ETA,SI,K,XI0,ETA0,SI0)
      N(K) = .125*(1. + XI0)*(1. + ETA0)*(1. + SI0)*(XI0+ETA0+ SI0-2.)
      NXI(K)=.125*(1. + ETA0)*(1. + SI0)*(2.*XI0+ETA0+ SI0-1.)*XII(K)
      NETA(K)=.125*(1. + XI0)*(1. + SI0)*(XI0+2.*ETA0+ SI0-1.)*ETAI(K)
```

```
      NSI(K)=.125*(1. + ETA0)*(1. + XI0)*(XI0+ETA0+2.*SI0-1.)*SII(K)
   30 CONTINUE
C
      K1 = 9
      K2 = 10
C
C        SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 13-16.
C
      DO 40 K = 13 , 16
      CALL ELMEXT(XI,ETA,SI,K,XI0,ETA0,SI0)
      N(K) = .25*(1. + XI0)*(1. + ETA0)*(1. - SI**2)
      NXI(K)=.25*(1. + ETA0)*(1. - SI**2)*XII(K)
      NETA(K)=.25*(1. + XI0)*(1. - SI**2)*ETAI(K)
      NSI(K)=-0.5*(1. + ETA0)*(1. + XI0)*SI
C
C        SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 9,11,17,19.
C
      CALL ELMEXT(XI,ETA,SI,K1,XI0,ETA0,SI0)
      N(K1) = .25*(1. - XI**2)*(1. + ETA0)*(1. + SI0)
      NXI(K1)=-0.5*(1. + ETA0)*(1. + SI0)*XI
      NETA(K1)=.25*(1. - XI**2)*(1. + SI0)*ETAI(K1)
      NSI(K1)=.25*(1. + ETA0)*(1. - XI**2)*SII(K1)
C
C        SHAPE FUNCTIONS AND THEIR DERIVATIVES FOR NODES 10,12,18,20.
C
      CALL ELMEXT(XI,ETA,SI,K2,XI0,ETA0,SI0)
      N(K2) = .25*(1. + XI0)*(1. - ETA**2)*(1. + SI0)
      NXI(K2)=.25*(1. - ETA**2)*(1. + SI0)*XII(K2)
      NETA(K2)=-0.5*(1. + XI0)*(1. + SI0)*ETA
      NSI(K2)=.25*(1. - ETA**2)*(1. + XI0)*SII(K2)
      IF (K1.EQ.11) THEN
            K1 = 17
            K2 = 18
      ELSE
            K1 = K1 + 2
            K2 = K2 + 2
      END IF
 40   CONTINUE
C
      END IF
C
      RETURN
      END
```

```
C
C ======================= E L M E X T ===============================
C
      SUBROUTINE ELMEXT(XI,ETA,SI,K,XIO,ETAO,SIO)
C
C ==================================================================
C I                                                                I
C I        FIRST DEVELOPED: 08-29-1988                             I
C I        LAST MODIFIED: 08-29-1988                               I
C I        BY: M. FOROOZESH                                        I
C I                                                                I
C ==================================================================
C
      REAL*8 XI,ETA,SI,XIO,ETAO,SIO
      COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
C
C
      XIO = XI*XII( K )
      ETAO =  ETA*ETAI( K )
      SIO = SI*SII( K )
C
      RETURN
      END
```

```
@PROCESS DIRECTIVE('*VDIR:')
C
C ========================= J A C O B I =============================
C
      SUBROUTINE JACOBI
C
C ==================================================================
C I                                                                I
C I        THIS PROGRAM CALCULATES THE JACOBIAN OF THE             I
C I        TRANSFORMATION BETWEEN THE LOCAL COORDINATES            I
C I        XI AND ETA AND THE GLOBAL COORDINATES X AND Y           I
C I        FOR INTEGRATION POINT 'INTGPN' OF ELEMENT NUMBER 'NREL'.  I
C I                                                                I
C I   INTGPN  =    INTEGRATION POINT NUMBER                        I
C I   NREL    =    ELEMENT NUMBER                                  I
C I   DETJAC  =    DETERMINANT OF THE JACOBIAN                     I
C I   NX(I)   =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO X    I
C I   NY(I)   =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO Y    I
C I   NZ(I)   =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO Z    I
C I   IERROR  =    ERROR CODE FOR EXCESIVE ELEMENT DISTORTION      I
C I                =1  NO ERROR                                    I
C I                =-1 EXCESIVE ELEMENT DISTORTION(DETJAC<0)       I
C I   NNEL    =    NUMBER OF NODES(SHAPE FUNCTIONS) PER ELEMENT    I
C I                                                                I
C I   NXI(I)  =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO XI   I
C I   NETA(I) =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO ETA  I
C I   NSI(I)  =    PARTIAL DERIVATIVE OF N(I) WITH RESPECT TO SI   I
C I   X(IK)   =    NODE COORDINATE  X,  IK=GLOBAL NODE NUMBER,     I
C I   Y(IK)   =    NODE COORDINATE  Y,  IK=GLOBAL NODE NUMBER,     I
C I   Z(IK)   =    NODE COORDINATE  Z,  IK=GLOBAL NODE NUMBER,     I
C I   XXI     =    PARTIAL DERIVATIVE OF X WITH RESPECT TO XI      I
C I   XETA    =    PARTIAL DERIVATIVE OF X WITH RESPECT TO ETA     I
C I   XSI     =    PARTIAL DERIVATIVE OF X WITH RESPECT TO SI      I
C I   YXI     =    PARTIAL DERIVATIVE OF Y WITH RESPECT TO XI      I
C I   YETA    =    PARTIAL DERIVATIVE OF Y WITH RESPECT TO ETA     I
C I   YSI     =    PARTIAL DERIVATIVE OF Y WITH RESPECT TO SI      I
C I   ZXI     =    PARTIAL DERIVATIVE OF Z WITH RESPECT TO XI      I
C I   ZETA    =    PARTIAL DERIVATIVE OF Z WITH RESPECT TO ETA     I
C I   ZSI     =    PARTIAL DERIVATIVE OF Z WITH RESPECT TO SI      I
C I                                                                I
C I                                                                I
C I        FIRST DEVELOPED: 08-27-1988                             I
C I        LAST MODIFIED: 08-27-1988                               I
C I        BY: M. FOROOZESH                                        I
C I                                                                I
C ==================================================================
C
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 N,NXI,NETA,NSI,NX,NY,NZ
      REAL*4 X,Y,Z
      COMMON/ISHAP1/N(20,27),NXI(20,27),NETA(20,27),NSI(20,27)
      COMMON/INPUT2/NOP(20,2000)
```

```
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
      COMMON/JACOB1/NX(20),NY(20),NZ(20)
C
C
C
C
C
C
      ENTRY JACB2D(INTGPN,NREL,NNEL,IERROR,DETJAC)
C
      XXI = 0.0
      XETA = 0.0
      YXI = 0.0
      YETA = 0.0
C
C*VDIR: ASSUME COUNT(8)
      DO 10 K1 = 1 , NNEL
      NODE = NOP(K1 , NREL)
      XXI  = XXI + NXI(K1,INTGPN)*X( NODE )
      XETA = XETA + NETA(K1,INTGPN)*X( NODE )
      YXI  = YXI + NXI(K1,INTGPN)*Y( NODE )
      YETA = YETA + NETA(K1,INTGPN)*Y( NODE )
   10 CONTINUE
C
      DETJAC = XXI*YETA - YXI*XETA
C
C*VDIR: ASSUME COUNT(8)
      DO 20 K = 1 , NNEL
      NX(K) = (YETA*NXI(K,INTGPN) - YXI*NETA(K,INTGPN))/DETJAC
      NY(K) = (-XETA*NXI(K,INTGPN) + XXI*NETA(K,INTGPN))/DETJAC
  20  CONTINUE
      RETURN
C
C
C
C
C
C
      ENTRY JACB3D(INTGPN,NREL,NNEL,IERROR,DETJAC)
C
      XXI = 0.0
      XETA = 0.0
      XSI = 0.0
      YXI = 0.0
      YETA = 0.0
      YSI = 0.0
      ZXI = 0.0
      ZETA = 0.0
      ZSI = 0.0
C
C*VDIR: ASSUME COUNT(20)
      DO 30 K1=1,NNEL
```

```
      NODE = NOP(K1 , NREL)
      XXI  = XXI + NXI(K1,INTGPN)*X( NODE )
      XETA = XETA + NETA(K1,INTGPN)*X( NODE )
      XSI  = XSI + NSI(K1,INTGPN)*X( NODE )
      YXI  = YXI + NXI(K1,INTGPN)*Y( NODE )
      YETA = YETA + NETA(K1,INTGPN)*Y( NODE )
      YSI  = YSI + NSI(K1,INTGPN)*Y( NODE )
      ZXI  = ZXI + NXI(K1,INTGPN)*Z( NODE )
      ZETA = ZETA + NETA(K1,INTGPN)*Z( NODE )
      ZSI  = ZSI + NSI(K1,INTGPN)*Z( NODE )
   30 CONTINUE
C
      DETJAC= XXI*(YETA*ZSI - ZETA*YSI) - YXI*(XETA*ZSI - ZETA*XSI) +
     1        ZXI*(XETA*YSI - YETA*XSI)
C
C*VDIR: ASSUME COUNT(20)
      DO 40 K = 1 , NNEL
      NX(K) = ((YETA*ZSI - ZETA*YSI)*NXI(K,INTGPN)
     1        -(YXI*ZSI - ZXI*YSI)*NETA(K,INTGPN)
     2        +(YXI*ZETA - ZXI*YETA)*NSI(K,INTGPN))/DETJAC
      NY(K) = (-(XETA*ZSI - ZETA*XSI)*NXI(K,INTGPN)
     1        +(XXI*ZSI - ZXI*XSI)*NETA(K,INTGPN)
     2       -(XXI*ZETA - ZXI*XETA)*NSI(K,INTGPN))/DETJAC
      NZ(K) = ((XETA*YSI - YETA*XSI)*NXI(K,INTGPN)
     1       -(XXI*YSI - YXI*XSI)*NETA(K,INTGPN)
     2       +(XXI*YETA - YXI*XETA)*NSI(K,INTGPN))/DETJAC
 40   CONTINUE
C
      RETURN
      END
```

```
C
C ========================== E L I N F O ===============================
C
      SUBROUTINE ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
C
C =====================================================================
C I                                                                   I
C I    P R O G R A M                                                  I
C I                                                                   I
C I    PROGRAM 'ELINFO' EXTRACTS ELEMENT INFORMATION FROM THE ARRAY   I
C I    'INFOEL'.                                                      I
C I                                                                   I
C I    A R G U M E N T     L I S T                                    I
C I                                                                   I
C I    ELNUM      = ELEMENT NUMBER PASSED BY THE CALLING ROUTINE      I
C I                                                                   I
C I    ITYPE      = ELEMENT TYPE PASSED TO THE CALLING ROUTINE        I
C I                                                                   I
C I    NNEL       = NUMBER OF NODES IN THE ELEMNT PASSED TO THE       I
C I                   CALLING ROUTINE                                 I
C I                                                                   I
C I    MATNUM     = MATERIAL I.D. NUMBER FOR THE ELEMENT PASSED TO THE I
C I                   CALLING ROUTINE                                 I
C I                                                                   I
C I    ISTART     = STARTING POSITION OF THE LINE CONNECTIVITY DATA   I
C I                   IN ARRAYS 'IS' AND 'IE'.                        I
C I                                                                   I
C I    LINES      = NUMBER OF LINES CONNECTING THE NODES WITHIN THE   I
C I                   ELEMENT                                         I
C I                                                                   I
C =====================================================================
      INTEGER ELNUM
      REAL*8 THICK
      COMMON/INPUTA/INFOEL(2,2000),ETHICK(2000)
C
      I  = INFOEL(1 , ELNUM)
      I1 = INFOEL(2 , ELNUM)
      LINES = IAND(I1 , 8257536)/131072
      ISTART = I1/8388608
      IFLAG  = IAND(I , 1835008)/262144
      ITYPE  = IAND(I , 261632)/512
      NNEL   = IAND(I , 508)/8
C
      RETURN
C
      ENTRY ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      I  = INFOEL(1 , ELNUM)
      I1 = INFOEL(2 , ELNUM)
      MATNUM = IAND(I , 7)
      NIPSI  = I/134217728
      NIPETA = IAND(I , 117440512)/16777216
      NIPXI  = IAND(I , 14680064)/2097152
```

```
INTCOD = IAND(I1 , 255)
IDENT  = IAND(I1 , 130816)/256
THICK = ETHICK( ELNUM )
RETURN
END
```

APPENDIX G

I/O MODULE

```
C
C ====================== C O M P R O ==================================
C
      SUBROUTINE COMPRO(BUFFER,BUFF,COMM,NEXT,K)
      INTEGER SPACE,COMMA,ZERO,NINE,DOT
      CHARACTER*80 BUFFER,BUFF,BUFER1,BUFF1
      CHARACTER*1 LINE(80),COM(4),COMLIN(80)
      CHARACTER*4 COMM,COMM1
      EQUIVALENCE (BUFER1,LINE),(COM,COMM1),(BUFF1,COMLIN)
      DATA SPACE,COMMA,ZERO,NINE,DOT,MINUS/64,107,240,249,75,96/
C
      BUFER1 = BUFFER
C
      IFL = 0
      BUFF1 = ' '
      NEXT = 0
      K1 = 0
  110 K = K + 1
      K1 = K1 + 1
      NCHAR = ICHAR(LINE( K ))
      IF (K.EQ.80) THEN
           GO TO 120
      ELSE IF (NCHAR.GE.ZERO.AND.NCHAR.LE.NINE) THEN
           IFL = 1
      ELSE IF (IFL.EQ.1.AND.NCHAR.NE.SPACE.AND.NCHAR.NE.DOT) THEN
           NEXT = 1
           K = K - 1
           K1 = K1 - 1
           GO TO 120
      END IF
      COMLIN( K1 ) = LINE( K )
      GO TO 110
C
  120 IFL = 0
      COMM1 = ' '
      ICOUNT = 0
      DO 130 K2 = 1 , K1
      NCHAR = ICHAR(COMLIN( K2 ))
      IF (NCHAR.EQ.SPACE.OR.NCHAR.EQ.DOT.OR.NCHAR.EQ.MINUS) THEN
           GO TO 130
      ELSE IF (NCHAR.LT.ZERO.OR.NCHAR.GT.NINE) THEN
           ICOUNT = ICOUNT + 1
           IF (ICOUNT.LE.4) COM( ICOUNT ) = COMLIN( K2 )
           COMLIN( K2 ) = ' '
      END IF
  130 CONTINUE
```

```
C
      BUFF = BUFF1
      BUFFER = BUFER1
      COMM = COMM1
      RETURN
      END
```

```
C
C ======================= I N P U T ===================================
C
      SUBROUTINE INPUT(IDOF,IOUT,IERROR)
C
C ====================================================================
C I                                                                  I
C I      SUBROUTINE INPUT2 READS ALL THE INPUT INFORMATION FROM      I
C I      CARD SETS 2 THROUGH 10. IT ALSO READS THE INFORMATION FROM  I
C I      THE PRIVIOUS RUNS IF THE PROGRAM IS RESUBMITED.             I
C I                                                                  I
C I                                                                  I
C I                                                                  I
C ====================================================================
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 NUX,NUY,NUZ,LX,LY
      REAL*4 X,Y,Z,FMAG,DMAG,ETHICK,THICK
      CHARACTER*80 BUFFER,BUFF,TITEL
      CHARACTER*4  COMM
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/INPUT2/NOP(20,2000)
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
      COMMON/INPUT5/NUX(10),NUY(10),NUZ(10),EX(10),EY(10),EZ(10),
     1 P1X(10),P1Y(10),P1Z(10),P2X(10),P2Y(10),P2Z(10)
      COMMON/INPUT6/WGTX(10),WGTY(10),WGTZ(10)
      COMMON/INPUT7/RX(8000),RY(8000),RZ(8000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE
      COMMON/INPUTA/INFOEL(2,2000),ETHICK(2000)
      COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
      COMMON/INPUTC/TITEL
      COMMON/INPUTD/COSTX(300),COSTY(300),COSTZ(300)
      COMMON/INPUTE/ISPB(4000)
      COMMON/INPUTF/MATYPE(10)
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/BOUND1/NCURVS
      COMMON/BOUND2/XZL(6),XZR(6),YZB(6),YZT(6)
      COMMON/INPUTI/INTFAC(500)
      COMMON/POINTS/LX(4 , 6) , LY(4 , 6)
      DIMENSION IDOF(1),DUMMY(6),M(20)
C
C       ICNT = COUTER FOR THE 'ISPB' ARRAY WHICH DETERMINES WHERE
C       TO LOOK FOR THE DIRECTION COSINES OF THE SKEW BOUNDARY IN
C       THE 'COSTX', 'COSTY' AND 'COSTZ' ARRAYS.
C
      ICNT = 1
C
C           READ THE COMMAND LINE BUFFER
C
      LDEV11 = 11
  100 IPOS = 0
      READ(LDEV11 , 101 ,END=1000) BUFFER
```

```
  101 FORMAT(A80)
C
  105 CALL COMPRO(BUFFER,BUFF,COMM,N,IPOS)
      IF(N.EQ.0) THEN
          ASSIGN 100 TO NEXT
      ELSE
          ASSIGN 105 TO NEXT
      END IF
C
C
C          EXTRACT THE FIRST FOUR CHARACTERS OF THE BUFFER AND REPLACE
C          ALL OTHER CHARACERS BY A BLANK EXCEPT NUMBERS 1-9.
C
C
      IF (COMM.EQ.'TITE') THEN
           GO TO 200
      ELSE IF (COMM.EQ.'COOR'.OR.COMM.EQ.'NODE'.OR.COMM.EQ.'JOIN') THEN
           GO TO 300
      ELSE IF (COMM.EQ.'MEMB'.OR.COMM.EQ.'INCI'.OR.COMM.EQ.'CONE') THEN
           GO TO 400
      ELSE IF (COMM.EQ.'DISP') THEN
           CALL IODISP(IDOF,NNDF,IDIM,ICNT,LDEV11,IOUT)
           GO TO 100
      ELSE IF (COMM.EQ.'LOAD') THEN
           CALL IOLOAD(LDEV11,IOUT)
           GO TO 100
      ELSE IF (COMM.EQ.'GRAP') THEN
           READ(BUFF , * , END = 2000) IFPLOT
           CALL GRAPHX(LDEV11,IOUT)
           GO TO 100
      END IF
      IF (COMM.EQ.'MATE') THEN
           READ(BUFF , * , END = 2000) MATNUM
      ELSE IF (COMM.EQ.'STOP') THEN
           READ(BUFF , * , END=2000) ISTOP
      ELSE IF (COMM.EQ.'LINE') THEN
           IFLAG1 = 0
      ELSE IF (COMM.EQ.'NONL') THEN
           IFLAG1 = 1
      ELSE IF (COMM.EQ.'NONS') THEN
           IFLAG2 = 1
      ELSE IF (COMM.EQ.'SYMM') THEN
           IFLAG2 = 0
      ELSE IF (COMM.EQ.'ELEM') THEN
           GO TO 440
      ELSE IF (COMM.EQ.'DIME') THEN
           READ(BUFF , * , END = 2000) IDIM
           NNDF = IDIM
      ELSE IF (COMM.EQ.'ITER') THEN
           READ(BUFF , * , END = 2000) MNIT
      ELSE IF (COMM.EQ.'INCR') THEN
           READ(BUFF , * , END = 2000) NLINC
```

```
      ELSE IF (COMM.EQ.'FACL') THEN
            READ(BUFF , * , END = 2000) FACLOW
            FAC = FACLOW
      ELSE IF (COMM.EQ.'FACH') THEN
            READ(BUFF , * , END = 2000) FACHIG
      ELSE IF (COMM.EQ.'NU') THEN
            READ(BUFF , * , END = 2000) NUX(MATNUM)
      ELSE IF (COMM.EQ.'E') THEN
            READ(BUFF , * , END = 2000) EX(MATNUM)
      ELSE IF (COMM.EQ.'WX') THEN
            READ(BUFF , * , END = 2000) WGTX(MATNUM)
      ELSE IF (COMM.EQ.'WY') THEN
            READ(BUFF , * , END = 2000) WGTY(MATNUM)
      ELSE IF (COMM.EQ.'WZ') THEN
            READ(BUFF , * , END = 2000) WGTZ(MATNUM)
      ELSE
            GO TO 150
      END IF
      GO TO NEXT
150   IF (COMM.EQ.'TYPE') THEN
            READ(BUFF , * , END = 2000) MATYPE( MATNUM )
      ELSE IF(COMM.EQ.'YIEL') THEN
            READ(BUFF , * , END = 2000) P1Z( MATNUM )
      ELSE IF(COMM.EQ.'ISOT') THEN
            READ(BUFF , * , END = 2000) P1Y( MATNUM )
      ELSE IF(COMM.EQ.'KINE') THEN
            READ(BUFF , * , END = 2000) P1X( MATNUM )
      ELSE IF(COMM.EQ.'REST') THEN
            IFLAG3 = 1
      ELSE IF(COMM.EQ.'OUTP') THEN
            READ(BUFF , * , END = 2000) IOINTR
      ELSE IF(COMM.EQ.'INTE') THEN
            GO TO 1100
      ELSE
            GO TO 155
      END IF
      GO TO NEXT
155   IF (COMM.EQ.'PAX') THEN
            READ(BUFF , * , END = 2000) LX(1 , NCURVE)
      ELSE IF(COMM.EQ.'PAY') THEN
            READ(BUFF , * , END = 2000) LY(1 , NCURVE)
      ELSE IF(COMM.EQ.'PBX') THEN
            READ(BUFF , * , END = 2000) LX(2 , NCURVE)
      ELSE IF(COMM.EQ.'PBY') THEN
            READ(BUFF , * , END = 2000) LY(2 , NCURVE)
      ELSE IF(COMM.EQ.'RAX') THEN
            READ(BUFF , * , END = 2000) LX(3 , NCURVE)
      ELSE IF(COMM.EQ.'RAY') THEN
            READ(BUFF , * , END = 2000) LY(3 , NCURVE)
      ELSE IF(COMM.EQ.'RBX') THEN
            READ(BUFF , * , END = 2000) LX(4 , NCURVE)
      ELSE IF(COMM.EQ.'RBY') THEN
```

```
          READ(BUFF , * , END = 2000) LY(4 , NCURVE)
     ELSE IF(COMM.EQ.'CURV') THEN
          READ(BUFF , * , END = 2000) NCURVE
          IF (NCURVE.GT.NCURVS) NCURVS = NCURVE
     ELSE IF(COMM.EQ.'XZL') THEN
          READ(BUFF , * , END = 2000) XZL(NCURVE)
     ELSE IF(COMM.EQ.'XZR') THEN
          READ(BUFF , * , END = 2000) XZR(NCURVE)
     ELSE IF(COMM.EQ.'YZB') THEN
          READ(BUFF , * , END = 2000) YZB(NCURVE)
     ELSE IF(COMM.EQ.'YZT') THEN
          READ(BUFF , * , END = 2000) YZT(NCURVE)
     ELSE
          GO TO 3000
     END IF
     GO TO NEXT
C
C ----- READ THE TITEL OF THE PROGRAM (CARD SET1)
C
  200 READ(BUFF , * , END = 2000) NUMBER
      DO 210 K = 1 , NUMBER
      READ(LDEV11 , 101) TITEL
  210 WRITE(IOUT , 101) TITEL
      GO TO NEXT
C
C ----- READ AND GENERATE THE NODAL COORDINATES
C
  300 I=0
      READ(BUFF , * , END = 2000) NNODES
  310 READ (LDEV11 , *) K,(DUMMY( IDIR ) , IDIR = 1 , IDIM),INCR
C
      X( K ) = DUMMY( 1 )
      Y( K ) = DUMMY( 2 )
      Z( K ) = DUMMY( 3 )
C
      I=I+1
      IF (INCR.EQ.0) GO TO 330
      N=(K-K1)/INCR
      DX=(X(K)-X(K1))/N
      DY=(Y(K)-Y(K1))/N
      DZ=(Z(K)-Z(K1))/N
      K2=K-INCR
      DO 320 J=K1,K2,INCR
      N1=(J-K1)/INCR
      X(J)=X(K1)+N1*DX
      Y(J)=Y(K1)+N1*DY
      Z(J)=Z(K1)+N1*DZ
      I=I+1
  320 CONTINUE
      I=I-1
  330 K1=K
      IF(I.LT.NNODES) GO TO 310
```

```
C
      WRITE(IOUT , 6009)
      DO 340 K1 = 1 , NNODES
  340 WRITE(IOUT , 5004)K1,X( K1 ),Y( K1 ),Z( K1 )
      GO TO NEXT
C
C ----- READ AND WRITE AND GENERATE THE ELEMENTS
C
  400 I = 0
      READ(BUFF , * , END = 2000) NELEM
  410 READ(LDEV11 , *)K,ITYPE,NNEL,(NOP(NODE,K),NODE=1,NNEL),INCR
      ID    = ITYPE/1000
      ID1   = ITYPE - ID*1000
      IFLAG = ID1/100
      ID2   = ID1 - IFLAG*100
      ITYPE = ID*100 + ID2
      IF (ITYPE.LT.300) THEN
           IF (NNEL.EQ.4) THEN
                ISTART = 1
                LINES  = 4
           ELSE IF(NNEL.EQ.5) THEN
                ISTART = 5
                LINES  = 5
           ELSE IF(ITYPE.EQ.208.OR.ITYPE.EQ.209) THEN
                ISTART = 10
                LINES  = 8
           ELSE IF(ITYPE.EQ.219) THEN
                ISTART = 54
                LINES  = 9
           END IF
      ELSE IF(ITYPE.GT.300) THEN
           IF (NNEL.EQ.8) THEN
                ISTART = 18
                LINES  = 12
           ELSE IF(NNEL.EQ.20) THEN
                ISTART = 30
                LINES = 24
           END IF
      END IF
C
      I = I + 1
      INF1 = NNEL*8 + ITYPE*512 + IFLAG*262144
      INF2 = 8388608*ISTART + 131072*LINES
      INFOEL(1 , K) = INFOEL(1 , K) + INF1
      INFOEL(2 , K) = INFOEL(2 , K) + INF2
      IF(INCR.EQ.0) THEN
          K1 = K
      ELSE
          K2 = (K - K1)/INCR
          DO 420 NODE = 1 , NNEL
  420     M( NODE ) = (NOP(NODE , K ) - NOP(NODE , K1))/K2
C
```

```
          DO 430 IELEM = K1+INCR , K-INCR , INCR
          INFOEL(1 ,  IELEM) = INFOEL(1 ,  IELEM) + INF1
          INFOEL(2 ,  IELEM) = INFOEL(2 ,  IELEM) + INF2
          I = I + 1
          IELEM1 = IELEM - INCR
          DO 430 NODE = 1 , NNEL
  430     NOP(NODE , IELEM) = NOP(NODE , IELEM1) + M( NODE )
      END IF
      IF(I.LT.NELEM) GO TO 410
      GO TO NEXT
C
C ----- READ AND GENERATE ELEMENT INFORMATIONS
C
  440 READ(BUFF , * , END = 2000) ISTART
      IEND = ISTART
      INTR = 1
      NIPXI = 0
      NIPETA = 0
      NIPSI = 0
      INTCOD = 0
      THICK  = 0.
  450 CALL COMPRO(BUFFER,BUFF,COMM,N,IPOS)
      IF (COMM.EQ.'TO') THEN
           READ(BUFF , * , END = 2000) IEND
      ELSE IF(COMM.EQ.'BY') THEN
           READ(BUFF , * , END = 2000) INTR
      ELSE IF(COMM.EQ.'NIPX') THEN
           READ(BUFF , * , END = 2000) NIPXI
      ELSE IF(COMM.EQ.'NIPE') THEN
           READ(BUFF , * , END = 2000) NIPETA
      ELSE IF(COMM.EQ.'NIPS') THEN
           READ(BUFF , * , END = 2000) NIPSI
      ELSE IF(COMM.EQ.'IRON') THEN
           READ(BUFF , * , END = 2000) INTCOD
      ELSE IF (COMM.EQ.'THIC') THEN
           READ(BUFF , * , END = 2000) THICK
      ELSE IF (COMM.EQ.'MATE') THEN
           READ(BUFF , * , END = 2000) MAT
      ELSE
           GO TO 3000
      END IF
      IF (N.NE.0) GO TO 450
C
      INF1 = 2097152*NIPXI + 16777216*NIPETA + 134217728*NIPSI + MAT
      INF2 = INTCOD + 256*(INTCOD + NIPXI + 10*NIPETA + 100*NIPSI)
      DO 460 K = ISTART , IEND , INTR
      ETHICK( K ) = ETHICK( K ) + THICK
      INFOEL(1 , K) = INFOEL(1 , K) + INF1
 460  INFOEL(2 , K) = INFOEL(2 , K) + INF2
      GO TO 100
C
C ----- READ AND GENERATE THE INTERFACE NODES
```

```
C
 1100 I = 0
      READ(BUFF , * , END = 2000) NINODE
 1110 READ (LDEV11 , *) K,INCR
      IF(INCR.EQ.0) THEN
          I = I + 1
          INTFAC( I ) = K
          IF (ISPB( K ).EQ.0) THEN
               ISPB( K ) = -ICNT
               ICNT = ICNT + 1
          END IF
      ELSE
          ISTART = INTFAC( I ) + INCR
          IEND   = K
          DO 1130 J = ISTART , IEND , INCR
          IF (ISPB( J ).EQ.0) THEN
               ISPB( J ) = -ICNT
               ICNT = ICNT + 1
          END IF
          I = I + 1
 1130     INTFAC( I ) = J
      END IF
      IF (I.LT.NINODE) GO TO 1110
      GO TO NEXT
C
 3000      WRITE(IOUT , 3001)
 3001      FORMAT(1X,'COMMAND ENTERED IS NOT RECOGNIZED BY *SAFE*')
 2000 STOP
 1000 RETURN
 5004 FORMAT(I5,1P,3G20.10)
 6002 FORMAT(//,1X,'PHYSICAL DIMENSION = ',I3/1X,'NUMBER OF NODES = ',
     1  I6/1X,'NUMBER OF ELEMENTS = ',I6/1X,'NUMBER OF NODAL D.O.F. = ',
     2  I6/,
     3  1X,'NUMBER OF APPLIED NODAL LOADS = ',I6/1X,'NUMBER OF IMPOSED'
     4  ,' NODAL DISPLACEMENTS = ',I6/1X,'NUMBER OF SKEW BOUNDARIES = ',
     5  I6/1X,'INTEGRATION CODE = ',I6/1X,'NUMBER OF LOAD INCREMENTS = '
     6  ,I6/1X,'GEOMETRIC LINEAR/NONLINEAR CODE = ',I6/1X,'MAXIMUM ',
     7  'NUMBER OF ITERATION ALLOWED = ',I6/1X,'FACTOR = ',F14.7)
 6009 FORMAT(/,20X,'COORDINATES OF THE NODES'/' NODE NO.',11X,'X',
     #19X,'Y',19X,'Z'/)
      END
```

```
C
C ======================= I O D I S P ===============================
C
      SUBROUTINE IODISP(IDOF,NNDF,IDIM,ICNT,LDEV11,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*4 COMM
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/INPUTD/COSTX(300),COSTY(300),COSTZ(300)
      COMMON/INPUTE/ISPB(4000)
      DIMENSION IDOF(1),D(3),IDO(3),THETA(3)
C
      ITIME = 0
      CST = 3.141592653589793D0/180.D0
      IF (IDIM.EQ.2) THEN
           CST1 = 90.
      ELSE
           CST1 = 0.
      END IF
C
   10 IPOS = 0
      READ(LDEV11 , '(A80)' , END = 1000) BUFFER
   20 CALL COMPRO(BUFFER,BUFF,COMM,N,IPOS)
C
      IF (COMM.EQ.'NODE') THEN
           IF(ITIME.EQ.1) THEN
                ASSIGN 30 TO NEXT
                GO TO 60
           END IF
 30        READ(BUFF , * , END=3000) ISTART
           ITIME = 1
           ISP   = 0
           IEND  = ISTART
           INTR  = 1
           DO 40 K1 = 1 , NNDF
           D( K1 ) = 0.
           IDO( K1 ) = 0
 40        THETA( K1 ) = 0.
           THETA( 3 ) = CST1
      ELSE IF(COMM.EQ.'TO') THEN
           READ(BUFF , * , END=3000) IEND
      ELSE IF(COMM.EQ.'BY') THEN
           READ(BUFF , * , END=3000) INTR
      ELSE IF(COMM.EQ.'X') THEN
           READ(BUFF , * , END=3000) D( 1 )
           IF (D( 1 ).EQ.0.) THEN
                IDO( 1 ) = 1
           ELSE
                IDO( 1 ) = -1
           END IF
      ELSE IF(COMM.EQ.'Y') THEN
           READ(BUFF , * , END=3000) D( 2 )
```

```
          IF (D( 2 ).EQ.0.) THEN
               IDO( 2 ) = 1
          ELSE
               IDO( 2 ) = -1
          END IF
     ELSE IF(COMM.EQ.'Z') THEN
          READ(BUFF , * , END=3000) D( 3 )
          IF (D( 3 ).EQ.0.) THEN
               IDO( 3 ) = 1
          ELSE
               IDO( 3 ) = -1
          END IF
     ELSE IF(COMM.EQ.'TX') THEN
          READ(BUFF , * , END=3000) THETA( 1 )
          ISP = 1
     ELSE IF(COMM.EQ.'TY') THEN
          READ(BUFF , * , END=3000) THETA( 2 )
          ISP = 1
     ELSE IF(COMM.EQ.'TZ') THEN
          READ(BUFF , * , END=3000) THETA( 3 )
          ISP = 1
     ELSE IF (COMM.EQ.'END') THEN
          ASSIGN 50 TO NEXT
          GO TO 60
 50       RETURN
     ELSE
          GO TO 2000
     END IF
C
     IF (N.NE.0) THEN
          GO TO 20
     ELSE
          GO TO 10
     END IF
C
  60 DO 80 K1 = ISTART , IEND , INTR
     DO 70 IDIR = 1 , NNDF
     ID = NNDF*(K1 - 1) + IDIR
     U( ID ) = U( ID ) + D( IDIR )
  70 IDOF( ID ) = IDOF( ID ) + IDO( IDIR )
C
     IF (ISP.EQ.1) THEN
          IF (ISPB( K1 ).EQ.0) THEN
               ISPB( K1 ) = ICNT
               K2 = ICNT
               ICNT = ICNT + 1
          ELSE IF (ISPB( K1 ).LT.0) THEN
               ISPB( K1 ) = -ISPB( K1 )
               K2 = ISPB( K1 )
          ELSE
               WRITE(IOUT , 2002) K1
               STOP
```

```
          END IF
          COSTX( K2 ) = DCOS(THETA( 1 )*CST)
          COSTY( K2 ) = DCOS(THETA( 2 )*CST)
          COSTZ( K2 ) = DCOS(THETA( 3 )*CST)
        END IF
  80    CONTINUE
        GO TO NEXT
C
 1000 RETURN
 2000 WRITE(IOUT , 2001)
 2001 FORMAT(1X,'>>>>>>> COMMAND ENTERED IS NOT RECOGNIZED AS A ',
     1         ' DISPLACEMENT SUBCOMMAND ')
 2002 FORMAT(/1X,'>>>>>>> PROGRAM STOPED DUE TO MULTIPLE DEFINITIONS'/
     1 9X,'OF THE SKEW DIRECTION FOR NODE ',I4)
 3000 STOP
      END
```

```
C
C ====================== I O L O A D ===============================
C
      SUBROUTINE IOLOAD(LDEV11,IOUT)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*4 COMM
      COMMON/INPUT7/RX(8000),RY(8000),RZ(8000)
      DIMENSION IDOF(1),D(3),IDO(3),THETA(3)
C
      ITIME = 0
C
  10  IPOS = 0
      READ(LDEV11 , '(A80)' , END = 1000) BUFFER
  20  CALL COMPRO(BUFFER,BUFF,COMM,N,IPOS)
C
      IF (COMM.EQ.'NODE') THEN
           IF(ITIME.EQ.1) THEN
                ASSIGN 30 TO NEXT
                GO TO 60
           END IF
 30        READ(BUFF , * , END=3000) ISTART
           ITIME = 1
           IEND  = ISTART
           INTR  = 1
           PX = 0.
           PY = 0.
           PZ = 0.
      ELSE IF(COMM.EQ.'TO') THEN
           READ(BUFF , * , END=3000) IEND
      ELSE IF(COMM.EQ.'BY') THEN
           READ(BUFF , * , END=3000) INTR
      ELSE IF(COMM.EQ.'X') THEN
           READ(BUFF , * , END=3000) PX
      ELSE IF(COMM.EQ.'Y') THEN
           READ(BUFF , * , END=3000) PY
      ELSE IF(COMM.EQ.'Z') THEN
           READ(BUFF , * , END=3000) PZ
      ELSE IF (COMM.EQ.'END') THEN
           ASSIGN 50 TO NEXT
           GO TO 60
 50        RETURN
      ELSE
           GO TO 2000
      END IF
C
      IF (N.NE.0) THEN
           GO TO 20
      ELSE
           GO TO 10
      END IF
C
```

```
  60  DO 80 K1 = ISTART , IEND , INTR
      RX( K1 ) = RX( K1 ) + PX
      RY( K1 ) = RY( K1 ) + PY
      RZ( K1 ) = RZ( K1 ) + PZ
  80  CONTINUE
      GO TO NEXT
C
 1000 RETURN
 2000 WRITE(IOUT , 2001)
 2001 FORMAT(1X,'>>>>>>> COMMAND ENTERED IS NOT RECOGNIZED AS A ',
     1       ' LOAD SUBCOMMAND ')
 3000 STOP
      END
```

```
C
C ====================== G R A P H X ===============================
C
      SUBROUTINE GRAPHX(LDEV11,IOUT)
      CHARACTER*80 BUFFER,BUFF
      CHARACTER*4 COMM
      COMMON/GRAPH3/XL,XR,YB,YT,ZF,D
      COMMON/GRAPH4/XVL,XVR,YVB,YVT,SX,SY
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES
C
C          READ THE COMMAND LINE BUFFER
C
  100 IPOS = 0
      READ(LDEV11 , 101 ,END=1000) BUFFER
  101 FORMAT(A80)
C
  105 CALL COMPRO(BUFFER,BUFF,COMM,N,IPOS)
      IF(N.EQ.0) THEN
          ASSIGN 100 TO NEXT
      ELSE
          ASSIGN 105 TO NEXT
      END IF
C
      IF (COMM.EQ.'FMAG') THEN
           READ(BUFF , * , END = 2000) FMAG
      ELSE IF(COMM.EQ.'DMAG') THEN
           READ(BUFF , * , END = 2000) DMAG
      ELSE IF(COMM.EQ.'WL') THEN
           READ(BUFF , * , END = 2000) XL
      ELSE IF(COMM.EQ.'WR') THEN
           READ(BUFF , * , END = 2000) XR
      ELSE IF(COMM.EQ.'WT') THEN
           READ(BUFF , * , END = 2000) YT
      ELSE IF(COMM.EQ.'WB') THEN
           READ(BUFF , * , END = 2000) YB
      ELSE IF(COMM.EQ.'VL') THEN
           READ(BUFF , * , END = 2000) XVL
      ELSE IF(COMM.EQ.'VR') THEN
           READ(BUFF , * , END = 2000) XVR
      ELSE IF(COMM.EQ.'VT') THEN
           READ(BUFF , * , END = 2000) YVT
      ELSE IF(COMM.EQ.'VB') THEN
           READ(BUFF , * , END = 2000) YVB
      ELSE IF(COMM.EQ.'LINE') THEN
           READ(BUFF , * , END = 2000) ITHICK
      ELSE IF(COMM.EQ.'CONT') THEN
           NOLINE = 1
      ELSE IF(COMM.EQ.'HIGH') THEN
           READ(BUFF , * , END = 2000) HIGHT
      ELSE IF(COMM.EQ.'ANGL') THEN
           READ(BUFF , * , END = 2000) ANGLE
      ELSE IF(COMM.EQ.'END') THEN
```

```
          GO TO 1000
     ELSE
          WRITE(IOUT , 200) COMM
200       FORMAT(1X,'>>>>>>> COMMAND "',A4,'" IS NOT RECOGNIZED BY'
    1    ,' ROUTINE GRAPHX')
     GO TO 2000
     END IF
     GO TO NEXT
1000 RETURN
2000 STOP
     END
```

```
C
C ======================= O U T P U T ===============================
C
      SUBROUTINE OUTPUT(IOUT,IERROR)
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*48 CDUMMY,CSELA(27),CSTRS(27),CSTRN(27),CDUM
      CHARACTER*48 CSTRS1,CSTRN1,CSELA1
      CHARACTER*57 CTEMP
      INTEGER ELNUM
      COMMON/UTIL1/CSTRS1,CSTRN1,CSELA1,CTEMP
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT6/WGTX(10),WGTY(10),WGTZ(10)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE
      COMMON/INPUTF/MATYPE(10)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/MAIN4/RE(8000)
      DIMENSION STRESS(6),STRAIN(6),COORDS(3),FORCES(6),DISPL(6)
      DIMENSION CSTR(6),STRPLA(6),STRELA(6)
      EQUIVALENCE (STRESS,STRAIN,FORCES,DISPL,CDUMMY),(STRELA,CDUM)
C
      ITYPE1 = 0
      IDENT1 = 0
      DO 100 ELNUM = 1 , NELEM
      CALL ELINFO(ELNUM,ITYPE,NNEL,IFLAG,ISTART,LINES)
      CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
C
      IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
           IF (ITYPE.GT.300) THEN
                ASSIGN 3001 TO IFOR
                ASSIGN 3101 TO IFOR1
                IF (MATYPE( MATNUM ).EQ.2) THEN
                     ASSIGN 3102 TO IF1
                ELSE
                     ASSIGN 3002 TO IF1
                END IF
                IF (IFLAG1.EQ.1) THEN
                     ASSIGN 3103 TO IF2
                ELSE
                     ASSIGN 3003 TO IF2
                END IF
                IF (INTCOD.EQ.0) THEN
                     CALL ISH3DG(ITYPE,NNEL,IERROR)
                ELSE
                     CALL ISH3DI(ITYPE,NNEL,IERROR)
                END IF
                IEND = 6
           ELSE
                IEND = 4
                CALL ISH2DG(ITYPE,NNEL,IERROR)
                IF (IFLAG.EQ.3) THEN
```

```
                    ASSIGN 2001 TO IFOR
                    ASSIGN 2101 TO IFOR1
                    IF (MATYPE( MATNUM ).EQ.2) THEN
                         ASSIGN 2104 TO IF1
                    ELSE
                         ASSIGN 2004 TO IF1
                    END IF
                    IF (IFLAG1.EQ.1) THEN
                         ASSIGN 2105 TO IF2
                    ELSE
                         ASSIGN 2005 TO IF2
                    END IF
               ELSE
                    ASSIGN 2001 TO IFOR
                    ASSIGN 2101 TO IFOR1
                    IF (MATYPE( MATNUM ).EQ.2) THEN
                         ASSIGN 2102 TO IF1
                    ELSE
                         ASSIGN 2002 TO IF1
                    END IF
                    IF (IFLAG1.EQ.1) THEN
                         ASSIGN 2103 TO IF2
                    ELSE
                         ASSIGN 2003 TO IF2
                    END IF
               END IF
          END IF
     END IF
     ITYPE1 = ITYPE
     IDENT1 = IDENT
C
     IF (MATYPE( MATNUM ).EQ.1) THEN
          DO 2  K1 = 1 , NIP
          CALL IOGET(LDEV1,96,'(A96)',5)
          CSTRS( K1 ) = CSTRS1
 2        CSTRN( K1 ) = CSTRN1
     ELSE IF (MATYPE( MATNUM ).EQ.2) THEN
          DO 3  K1 = 1 , NIP
          CALL IOGET(LDEV1,144,'(A144)',6)
          CSTRS( K1 ) = CSTRS1
          CSTRN( K1 ) = CSTRN1
 3        CSELA( K1 ) = CSELA1
     END IF
C
     WRITE(IOUT , 5002) ELNUM
     WRITE(IOUT , IF1)
         DO 10 INTGPN = 1 , NIP
         CALL COORD1(ELNUM,NNEL,INTGPN,COORDS(1),COORDS(2),COORDS(3))
         CDUMMY = CSTRN( INTGPN )
         WRITE(IOUT , IFOR) INTGPN,(COORDS(K1),K1=1,IDIM),
    1    (STRAIN(K1),K1=1,IEND)
         IF (MATYPE( MATNUM ).EQ.2) THEN
```

```
                CDUM = CSELA( INTGPN )
                DO 5 K1 = 1 , IEND
  5             STRPLA( K1 ) = STRAIN( K1 ) - STRELA( K1 )
                WRITE(IOUT ,IFOR1) (STRELA(K1),K1=1,IEND)
                WRITE(IOUT ,IFOR1) (STRPLA(K1),K1=1,IEND)
          END IF
  10      CONTINUE
C
C
      WRITE(IOUT , IF2)
          DO 20 INTGPN = 1 , NIP
          CALL COORD1(ELNUM,NNEL,INTGPN,COORDS(1),COORDS(2),COORDS(3))
          CDUMMY = CSTRS( INTGPN )
          WRITE(IOUT , IFOR) INTGPN,(COORDS(K1),K1=1,IDIM),
     1    (STRESS(K1),K1=1,IEND)
          IF(IFLAG1.EQ.1) THEN
               CALL CAUCHY(ELNUM,ITYPE,NNEL,NNDF,INTGPN,STRESS,CSTR)
               WRITE(IOUT , IFOR1) (CSTR(K1),K1=1,IEND)
          END IF
  20      CONTINUE
 100  CONTINUE
C
      WRITE(IOUT , 6009)
      DO 220 K1 = 1 , NNODES
      DO 210 K2 = 1 , NNDF
      K3 = (K1 -1)*NNDF + K2
 210  FORCES( K2 ) = RE( K3 )
 220  WRITE(IOUT , 5004) K1,(FORCES(K2),K2 = 1 , NNDF)
C
      WRITE(IOUT , 6007)
      DO 240 K1 = 1 , NNODES
      DO 230 K2 = 1 , NNDF
      K3 = (K1 -1)*NNDF + K2
 230  DISPL( K2 ) = UTOTAL( K3 )
 240  WRITE(IOUT , 5004) K1,(DISPL( K2 ),K2 = 1 , NNDF)
C
      RETURN
C
 2001 FORMAT(/I4,1P,6G14.5)
C
 2101 FORMAT(32X,1P,4G14.5)
C
 2002 FORMAT(/,50X,'STRAIN COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #12X,'EXX',11X,'EYY',11X,'EXY',11X,'EZZ'/)
C
 2102 FORMAT(/,70X,'STRAIN COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #7X,' TOTAL_X    ',3X,' TOTAL_Y    ',3X,' TOTAL_XY    ',2X,
     #' TOTAL_Z    '/
     #34X,' ELAST_X',6X,' ELAST_Y',6X,' ELAST_XY',5X,' ELAST_Z'/
     #34X,' PLAST_X',6X,' PLAST_Y',6X,' PLAST_XY',5X,' PLAST_Z'/)
C
 2003 FORMAT(/,70X,'STRESS COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
```

```
     #12X,'SXX',11X,'SYY',11X,'SXY',11X,'SZZ'/)
C
 2103 FORMAT(/,70X,'STRESS COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #7X,'2ND PIOLA_X',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_XY',2X,
     #'2ND_PIOLA_Z'/
     #34X,'CAUCHY_X',6X,'CAUCHY_Y',6X,'CAUCHY_XY',5X,'CAUCHY_Z'/)
C
 2004 FORMAT(/,50X,'STRAIN COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #11X,'ER ',11X,'EY ',11X,'ERY',11X,'ET '/)
C
 2104 FORMAT(/,70X,'STRAIN COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #7X,' TOTAL_R    ',3X,' TOTAL_Y    ',3X,' TOTAL_RY    ',2X,
     #' TOTAL_T    '/
     #34X,' ELAST_R',6X,' ELAST_Y',6X,' ELAST_RY',5X,' ELAST_T'/
     #34X,' PLAST_R',6X,' PLAST_Y',6X,' PLAST_RY',5X,' PLAST_T'/)
C
 2005 FORMAT(/,50X,'STRESS COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #12X,'SR ',11X,'SY ',11X,'SRY',11X,'ST '/)
C
 2105 FORMAT(/,70X,'STRESS COMPONENTS'//1X,'POINT',5X,'X',14X,'Y',
     #7X,'2ND PIOLA_R',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_RY',2X,
     #'2ND_PIOLA_T'/
     #34X,'CAUCHY_R',6X,'CAUCHY_Y',6X,'CAUCHY_RY',5X,'CAUCHY_T'/)
C
 3001 FORMAT(I3,1P,9G14.5)
 3101 FORMAT(45X,1P,6G14.5/)
C
 3002 FORMAT(/,50X,'STRAIN COMPONENTS'//1X,'POINT',5X,'X',13X,'Y',13X,
     #'Z',11X,'EXX',11X,'EYY',11X,'EZZ',11X,'EXY',11X,'EYZ',11X,'EXZ'/)
C
 3102 FORMAT(/,50X,'STRAIN COMPONENTS'//1X,'POINT',5X,'X',13X,'Y',
     #14X,'Z',7X,' TOTAL_X    ',3X,' TOTAL_Y    ',3X,' TOTAL_Z    ',2X,
     #' TOTAL_XY    ',2X,' TOTAL_YZ    ',2X,' TOTAL_XZ    '/
     #48X,' ELAST_X',6X,' ELAST_Y',6X,' ELAST_Z',5X,' ELAST_XY',5X,
     #' ELAST_YZ',5X,' ELAST_XZ'/
     #48X,' PLAST_X',6X,' PLAST_Y',6X,' PLAST_Z',5X,' PLAST_XY',5X,
     #' PLAST_YZ',5X,' PLAST_XZ'/)
C
 3003 FORMAT(/,50X,'STRESS COMPONENTS'//1X,'POINT',5X,'X',13X,'Y',13X,
     #'Z',11X,'SXX',11X,'SYY',11X,'SZZ',11X,'SXY',11X,'SYZ',11X,'SXZ'/)
C
 3103 FORMAT(/,50X,'STRESS COMPONENTS'//1X,'POINT',5X,'X',13X,'Y',
     #14X,'Z',7X,'2ND PIOLA_X',3X,'2ND PIOLA_Y',3X,'2ND PIOLA_Z',2X,
     #'2ND_PIOLA_XY',2X,'2ND PIOLA_YZ',2X,'2ND PIOLA_XZ'/
     #48X,'CAUCHY_X',6X,'CAUCHY_Y',6X,'CAUCHY_Z',5X,'CAUCHY_XY',5X,
     #'CAUCHY_YZ',5X,'CAUCHY_XZ'/)
C
 5002 FORMAT(/20X,'**********',' ELEMENT=',I5,' **********')
 5004 FORMAT(I5,1P,3G20.10)
 5005 FORMAT(I3,9(1X,G12.9))
C
 6007 FORMAT(/,20X,'DISPLACEMENT OF THE NODES'/' NODE NO.',10X,'UX',
```

```
     #18X,'UY',18X,'UZ'/)
C
 6008 FORMAT(/,45X,'TOTAL PLASTIC WORK AT GAUSSIAN POINTS'/11X,'P1',
     #11X,'P1',11X,'P3',11X,'P4',11X,'P5',11X,'P6',11X,'P7',11X,'P8'
     #,11X,'P9'/)
C
 6009 FORMAT(/,20X,'REACTIONS AT THE NODES'/' NODE NO.',10X,'RX',
     #18X,'RY',18X,'RZ'/)
C
 6010 FORMAT(/20X,'POINTS THAT HAVE YIELDED'/12X,'P1',5X,'P2',5X,'P3',
     #5X,'P4',5X,'P5',5X,'P6',5X,'P7',5X,'P8',5X,'P9')

      END
```

APPENDIX H

REAL I/O UTILITIES MODULE

```
C
C ===================== U T I L I T ==================================
C
      SUBROUTINE UTILIT
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*240 BUFFER
      CHARACTER*6  FMAT
      COMMON/UTIL1/BUFFER
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/MAIN4/RE(8000)
      COMMON/TEMP/PRESS,PLWORK
      DIMENSION IDOF( 1 )
C
C                E N T R Y    S W A P
C
      ENTRY SWAP
C
      ISWAP = LDEV1
      LDEV1 = LDEV2
      LDEV2 = ISWAP
C
      RETURN
C
C                E N T R Y    S W A P 1
C
      ENTRY SWAP1
C
      LDKEEP= LDEV1
      LDEV1 = LDEV2
      ISWAP = LDEV2
      LDEV2 = LDEV3
      LDEV3 = ISWAP
C
      RETURN
C
C                E N T R Y    S W A P 2
C
      ENTRY SWAP2
C
      LDEV1 = LDEV2
      LDEV2 = LDKEEP
C
      RETURN
C
C                E N T R Y    R E W I N
```

```
C
      ENTRY REWIN
C
      REWIND(UNIT=LDEV1,ERR=1000,IOSTAT=IERROR)
      REWIND(UNIT=LDEV2,ERR=1000,IOSTAT=IERROR)
      RETURN
C
C                E N T R Y    R E S T O R
C
      ENTRY RESTOR(MDF,ISTART,NTDF,IDOF)
C
      READ(LDEVST , *)ISTART,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 100 K1 = 1 , MDF
  100 READ(LDEVST, *)RE( K1 ),UTOTAL( K1 )
      REWIND(UNIT=LDEVST,ERR=1000,IOSTAT=IERROR)
      RETURN
C
C                E N T R Y    S T O R E
C
      ENTRY STORE(MDF,INCREM,NTDF,IDOF)
C
      WRITE(LDEVST , *)INCREM,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 200 K1 = 1 , MDF
  200 WRITE(LDEVST , *)RE( K1 ),UTOTAL( K1 ),IDOF( K1 )
      REWIND(UNIT=LDEVST,ERR=1000,IOSTAT=IERROR)
      WRITE(LDEV5 , *)INCREM,PRESS,UTOTAL( 2 ),UTOTAL(2114),PLWORK
      PLWORK = 0
      RETURN
C
C                E N T R Y    S T O R E 1
C
      ENTRY STORE1(MDF,NTDF,IDOF)
C
      WRITE(15 , *)NTDF,LDEV1,LDEV2,LDEV3
      DO 300 K1 = 1 , MDF
  300 WRITE(15 , *)RE( K1 ),UTOTAL( K1 )
      REWIND(UNIT=15,ERR=1000,IOSTAT=IERROR)
      RETURN
C
C                E N T R Y    R E S T R 1
C
      ENTRY RESTR1(MDF,NTDF,IDOF)
C
      READ(15 , *)NTDF,LDEV1,LDEV2,LDEV3
      DO 400 K1 = 1 , MDF
  400 READ(15 , *)RE( K1 ),UTOTAL( K1 )
      REWIND(UNIT=15 ,ERR=1000,IOSTAT=IERROR)
      RETURN
C
C                E N T R Y    I O G E T
C
      ENTRY IOGET(IDEV,LENGTH,FMAT,N)
```

```
      READ(UNIT=IDEV , FMT=FMAT(1:N)) BUFFER(1:LENGTH)
      RETURN
C
C               E N T R Y    I O P U T
C
      ENTRY IOPUT(IDEV,LENGTH,FMAT,N)
      WRITE(UNIT=IDEV ,FMT=FMAT(1:N)) BUFFER(1:LENGTH)
      RETURN
C
C               E N T R Y    I O B K S
C
      ENTRY IOBKS(IDEV)
      BACKSPACE (UNIT=IDEV)
      RETURN
C
C               E N T R Y    A R C H I V
C
      ENTRY ARCHIV(MDF)
      RETURN
C
C               E N T R Y    R E C O V
C
      ENTRY RECOV(MDF,ISTART,NTDF,IDOF)
C
      READ(LDEVST , *)ISTART,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 500 K1 = 1 , MDF
  500 READ(LDEVST, *)RE( K1 ),UTOTAL( K1 ),IDOF( K1 )
      REWIND(UNIT=LDEVST,ERR=1000,IOSTAT=IERROR)
      RETURN
C
      RETURN
 1000 WRITE(IOUT , 1001)
 1001 FORMAT(1H0,1X,'ERROR IN REWINDING UTILITY FILES IS DETECTED',
     1  ' BY ROUTINE CONTRL')
      STOP
      END
```

APPENDIX I

VIRTUAL I/O UTILITIES MODULE

```
C
C ===================== U T I L I T ==================================
C
      SUBROUTINE UTILIT
      IMPLICIT REAL*8 (A-H,O-Z)
      CHARACTER*201 BUFFER,IOBUFF,TBUFF1,TBUFF2
      CHARACTER*6  FMAT
      COMMON/UTIL1/BUFFER
      COMMON/UTIL2/IOBUFF(3,18000)
      COMMON/UTIL3/NREC(3),NWMAX
      COMMON/UTIL4/TBUFF1,TBUFF2
      COMMON/UTIL5/REQ1(8000),REQ2(8000),UTL1(8000),UTL2(8000),
     1             IDOF1(8000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/MAIN4/RE(8000)
      COMMON/TEMP/PRESS,PLWORK
      DIMENSION IDOF( 1 )
C
C              E N T R Y    S W A P
C
      ENTRY SWAP
C
      ISWAP = LDEV1
      LDEV1 = LDEV2
      LDEV2 = ISWAP
C
      RETURN
C
C              E N T R Y    S W A P 1
C
      ENTRY SWAP1
C
      LDKEEP= LDEV1
      LDEV1 = LDEV2
      ISWAP = LDEV2
      LDEV2 = LDEV3
      LDEV3 = ISWAP
C
      RETURN
C
C              E N T R Y    S W A P 2
C
      ENTRY SWAP2
C
      LDEV1 = LDEV2
```

```
      LDEV2 = LDKEEP
C
      RETURN
C
C                 E N T R Y    R E W I N
C
      ENTRY REWIN
C
      NWMAX = NREC(LDEV2)
      NREC(1) = 1
      NREC(2) = 1
      NREC(3) = 1
      RETURN
C
C                 E N T R Y    R E S T O R
C
      ENTRY RESTOR(MDF,ISTART,NTDF,IDOF)
C
      READ(TBUFF1 , *)ISTART,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 100 K1 = 1 , MDF
      RE( K1 ) = REQ1( K1 )
      UTOTAL( K1 ) = UTL1( K1 )
  100 CONTINUE
      RETURN
C
C                 E N T R Y    S T O R E
C
      ENTRY STORE(MDF,INCREM,NTDF,IDOF)
C
      WRITE(TBUFF1 , *)INCREM,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 200 K1 = 1 , MDF
      REQ1( K1 ) = RE( K1 )
      UTL1( K1 ) = UTOTAL( K1 )
      IDOF1( K1 ) = IDOF( K1 )
  200 CONTINUE
      WRITE(LDEV5 , *)INCREM,PRESS,UTOTAL( 2 ),UTOTAL(2114),PLWORK
      PLWORK = 0
      RETURN
C
C                 E N T R Y    S T O R E 1
C
      ENTRY STORE1(MDF,NTDF,IDOF)
C
      WRITE(TBUFF2 , *)NTDF,LDEV1,LDEV2,LDEV3
      DO 300 K1 = 1 , MDF
      REQ2( K1 ) = RE( K1 )
      UTL2( K1 ) = UTOTAL( K1 )
  300 CONTINUE
      RETURN
C
C                 E N T R Y    R E S T R 1
C
```

```
      ENTRY RESTR1(MDF,NTDF,IDOF)
C
      READ(TBUFF2 , *)NTDF,LDEV1,LDEV2,LDEV3
      DO 400 K1 = 1 , MDF
      RE( K1 ) = REQ2( K1 )
      UTOTAL( K1 ) = UTL2( K1 )
  400 CONTINUE
      RETURN
C
C              E N T R Y    I O G E T
C
      ENTRY IOGET(IDEV,LENGTH,FMAT,N)
      BUFFER(1:LENGTH) = IOBUFF(IDEV , NREC( IDEV ))(1:LENGTH)
      NREC( IDEV ) = NREC( IDEV ) + 1
      RETURN
C
C              E N T R Y    I O P U T
C
      ENTRY IOPUT(IDEV,LENGTH,FMAT,N)
      IOBUFF(IDEV , NREC( IDEV )) = BUFFER
      NREC( IDEV ) = NREC( IDEV ) + 1
      RETURN
C
C              E N T R Y    I O B K S
C
      ENTRY IOBKS(IDEV)
      NREC( IDEV ) = NREC( IDEV ) - 1
      RETURN
C
C              E N T R Y    A R C H I V
C
      ENTRY ARCHIV(MDF)
      READ(TBUFF1 , *)ISTART,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      WRITE(LDEVST , *)ISTART,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 450 K1 = 1 , MDF
      WRITE(LDEVST , *)REQ1( K1 ),UTL1( K1 ),IDOF1( K1 )
 450  CONTINUE
C
      DO 500 K1 = 1 , NWMAX
      BUFFER = IOBUFF(LDEV1 , K1)
 500  WRITE(LDEV1,'(A201)')BUFFER
      RETURN
C
C              E N T R Y    R E C O V
C
      ENTRY RECOV(MDF,ISTART,NTDF,IDOF)
      READ(LDEVST , *)ISTART,NTDF,NWMAX,LDEV1,LDEV2,LDEV3
      DO 600 K1 = 1 , MDF
 600  READ(LDEVST , *)RE( K1 ),UTOTAL( K1 ),IDOF( K1 )
      REWIND (UNIT=LDEVST)
C
      DO 700 K1 = 1 , NWMAX
```

```
      READ(LDEV1,'(A201)')BUFFER
 700  IOBUFF(LDEV1 , K1) = BUFFER
      REWIND (UNIT=LDEV1)
      RETURN
C
      END
```

APPENDIX J

INITIALIZER MODULE

```
      SUBROUTINE INITAL(IDOF)
      IMPLICIT REAL*8 (A-H,O-Z)
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/INPUT7/RX(8000),RY(8000),RZ(8000)
      COMMON/INPUTE/ISPB(4000)
      DIMENSION IDOF( 1 )
C
      NNODS = 4000
      NELEM = 2000
      NDOF  = 8000
C
      DO 10 K1 = 1 , NNODS
      ISPB( K1 ) = 0
   10 CONTINUE
C
      DO 20 K1 = 1 , NDOF
      IDOF( K1 ) = 0
      U( K1 ) = 0.
      RX( K1 ) = 0.
      RY( K1 ) = 0.
      RZ( K1 ) = 0.
   20 CONTINUE
C
      RETURN
      END
```

APPENDIX K

GRAPHICS MODULE

```
C
C ====================== I N P L O T==============================
C
      SUBROUTINE INPLOT(NELEM)
C
C ===============================================================
C  I                                                             I
C  I   P R O G R A M:                                            I
C  I                                                             I
C  I                                                             I
C  I   INPLOT PERFORMS THE FOLLOWING FUNCTIONS                   I
C  I                                                             I
C  I       1 - INITILIZES THE PLOTTING DEVICE                    I
C  I       2 - EVALUATES THE LINE CONNECTIVITY OF THE MESH       I
C  I       3 - EVALUATES THE NODE CONNECTIVITY OF THE MESH       I
C  I                                                             I
C  I   A R G U M E N T   L I S T:                                I
C  I                                                             I
C  I                                                             I
C  I   NELEM      = NUMBER OF ELEMENTS IN THE MESH               I
C  I                                                             I
C  I   C O N S I D E R A T I O N S:                              I
C  I                                                             I
C  I   THE ARRAY IREP STORES THE REPETITION NUMBER OF THE LINES IN I
C  I   BITS 0-5 AND THE STORES THE REPETITION NUMBER OF THE      I
C  I   NODES IN BITS 6-31. THESE VALUES MUST BE EXTRACTED PROPERLY I
C  I   IF NEEDED.                                                I
C  I                                                             I
C ===============================================================
      INTEGER ELNUM
      COMMON/INPUT2/NOP(20,2000)
      COMMON/GRAPH1/IS(62),IE(62)
      COMMON/GRAPH2/IVS(8000),IVE(8000),ILS(8000),ILE(8000)
      COMMON/GRAPH3/XL,XR,YB,YT,ZF,D
      COMMON/GRAPH4/XVL,XVR,YVB,YVT,SX,SY
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES,NLIN
      COMMON/GRAPH8/LDEVP
      COMMON/IREP1/IREP(8000),LREP(8000)
C
C ----- SEARCH THROUGH ELEMENTS FOR LINE AND NODE CONNECTIVITY
C
      DO 60 ELNUM = 1 , NELEM
      CALL ELINFO(ELNUM,ITYPE,NN,IFLAG,ISTART,LINES)
C
C ----- DETERMINE THE LINE CONNECTIVITY OF THE ELEMENT
C
```

```
      ISTOP = ISTART + LINES - 1
      DO 20 K2 = ISTART , ISTOP
      J1 = NOP(IS( K2 ), ELNUM)
      J2 = NOP(IE( K2 ), ELNUM)
      ICODE = 0
      DO 10 K1 = 1 , NLINES
      IF (J2.EQ.IVS(K1).OR.J2.EQ.IVE(K1)) THEN
           IF (J1.EQ.IVE(K1).OR.J1.EQ.IVS(K1)) THEN
                IREP( K1 ) = IREP( K1 ) + 1
                ICODE = 1
           END IF
      END IF
   10 CONTINUE
C
      IF (ICODE.EQ.0) THEN
           NLINES = NLINES + 1
           IREP( NLINES ) = IREP( NLINES ) + 1
           IVS( NLINES ) = J1
           IVE( NLINES ) = J2
      END IF
   20 CONTINUE
C
C ----- DETERMINE THE NODE CONNECTIVITY OF THE ELEMENT
C
      DO 30 K1 = 1 , NN
      NODE = NOP(K1 , ELNUM)
      IREP( NODE ) = IREP( NODE ) + 32
   30 CONTINUE
C
C ----- DETERMINE THE LINE CONNECTIVITY USING THE CORNER NODES
C
      DO 50 K2 = 1 , 4
      J1 = NOP(IS( K2 ), ELNUM)
      J2 = NOP(IE( K2 ), ELNUM)
      ICODE = 0
      DO 40 K1 = 1 , NLIN
      IF (J2.EQ.ILS(K1).OR.J2.EQ.ILE(K1)) THEN
           IF (J1.EQ.ILE(K1).OR.J1.EQ.ILS(K1)) THEN
                LREP( K1 ) = LREP( K1 ) + 1
                ICODE = 1
           END IF
      END IF
   40 CONTINUE
C
      IF (ICODE.EQ.0) THEN
           NLIN = NLIN + 1
           LREP( NLIN ) = LREP( NLIN ) + 1
           ILS( NLIN ) = J1
           ILE( NLIN ) = J2
      END IF
   50 CONTINUE
   60 CONTINUE
```

```
C
C ----- DETERMINE THE FACTORS FOR THE WINDOW TO VIEWPORT MAPPING
C
      SX = (XVR - XVL)/(XR - XL)
      SY = (YVT - YVB)/(YT - YB)
C
C ----- TO PRESERVE PROPORTIONALITY USE THE SMALEST OF THE SX AND SY
C       IN BOTH X AND Y DIRECTIONS
C
      IF (SX.GT.SY) THEN
           SX = SY
      ELSE
           SY = SX
      END IF
      RETURN
C
      ENTRY ENDPLT
      ID = 30
      WRITE(LDEVP,'(I3)') ID
      RETURN
      END
```

```
C
C  ====================== P L O T E R =============================
C
      SUBROUTINE PLOTER(VALUE,NNODES,NELEM,NNDF,IDIM,NINODE,IFLAG1,
     1                  IOUT)
C
C  =================================================================
C  I                                                               I
C  I   P R O G R A M:                                              I
C  I                                                               I
C  I                                                               I
C  I   PLOTER IS THE PLOTTING MODULE, ITS FUNCTIONS CONSIST OF     I
C  I                                                               I
C  I       1 - DRAWING THE MESH                                    I
C  I       2 - IDENTIFYING AND DRAWING CONTOUR LINES               I
C  I       3 - IDENTIFYING NODES AND THE ACTIVE YIELD POINTS       I
C  I                                                               I
C  I   A R G U M E N T   L I S T:                                  I
C  I                                                               I
C  I                                                               I
C  I   VALUE( I ) = THE ARRAY WHICH WILL BE USED TO STORE THE      I
C  I                VALUES TO BE CONTOURED AT THE NODES. THIS      I
C  I                ARRAY USES THE SAME STORAGE AREA AS THE        I
C  I                STIFFNESS MATRIX.                              I
C  I                                                               I
C  I   NNODES     = NUMBER OF NODES IN THE MESH                    I
C  I                                                               I
C  I   NELEM      = NUMBER OF ELEMENTS IN THE MESH                 I
C  I                                                               I
C  I   NNDF       = NUMBER OF NODAL DEGREES OF FREEDOM             I
C  I                                                               I
C  I   IDIM       = PHYSICAL DIMENSION OF THE PROBLEM (2D OR 3D).  I
C  I                                                               I
C  I   NINODE     = NUMBER OF INTERFACE NODES FOR CONTACT PROBLEMS I
C  I                CONTACT PROBLEMS REQUIRE THE 'BOUND' MODULE    I
C  I                                                               I
C  I   IFLAG1     = 0; GEOMETRIC LINEARITY                         I
C  I                1; GEOMETRIC NON-LINEARITY                     I
C  I                                                               I
C  I   IOUT       = OUTPUT DEVICE NUMBER                           I
C  I                                                               I
C  I                                                               I
C  I   C O N S I D E R A T I O N S:                                I
C  I                                                               I
C  I   COMMON/MAIN1/ IS SHARED WITH THE 'SAFE' MODULE IN ORDER TO  I
C  I   SAVE MEMORY. THE SIZE OF THIS COMMON BLOCK IN THIS MODULE   I
C  I   SHOULD MATCH ITS SIZE IN THE 'SAFE' AND THE 'BLKDATA'       I
C  I   MODULES.                                                    I
C  I                                                               I
C  =================================================================
      INTEGER ELNUM
      REAL*8 UTOTAL,XIP,YIP,ZIP,UXIP,UYIP,UZIP,THICK
```

```
      LOGICAL*4 LOGIC
      CHARACTER*80 GTITLE
C
C          W A R N I N G    Z O N E!    (REFER TO CONSIDERATIONS)
C
      COMMON/MAIN1/XXS(1500),XXE(1500),YYS(1500),YYE(1500),XC(1500),
     1             YC(1500),LNSTR(1500),LNEND(1500)
C
C          E N D   W A R N I N G   Z O N E
C
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,2000)
      COMMON/GRAPH1/IS(62),IE(62)
      COMMON/GRAPH2/IVS(8000),IVE(8000),ILS(8000),ILE(8000)
      COMMON/IREP1/IREP(8000),LREP(8000)
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES,NLIN
      COMMON/GRAPH7/GTITLE(20)
      COMMON/GRAPH8/LDEVP
      COMMON/PLAST1/IYIEL(2000)
      COMMON/CONTR1/INCREM,NIT
      DIMENSION VALUE(1),LEGEND(10),VLEGND(10)
C
C ----- IR = MAXIMUM NUMBER OF REPETITIONS FOR SURFACE LINES
C
      IR = IDIM - 1
C
C ----- LIMIT = THE MAXIMUM NUMBER OF LINE SEGMENTS PER CONTOUR VALUE
C               ALLOWED BY THE SIZE OF THE ARRAYS IN COMMON MAIN1
C
      LIMIT = 3000
C
C ----- IDENTIFY EACH NODE BY A DIAMOND
C
      DO 20 NODE = 1 , NNODES
      ID1 = NNDF*(NODE - 1)
      XS = X( NODE )*FMAG + UTOTAL( ID1 + 1 )*DMAG
      YS = Y( NODE )*FMAG + UTOTAL( ID1 + 2 )*DMAG
      CALL VIEW2(XS,YS,.02,5)
   20 CONTINUE
C
C ----- IDENTIFY EACH ACTIVE YIELD POINT WITH AN ASTRISC
C
      ITYPE1 = 0
      IDENT1 = 0
      DO 30 ELNUM = 1 , NELEM
      CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      CALL ELINFO(ELNUM,ITYPE,NN,IFLAG,ISTART,LINES)
C
      IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
```

```
          IF (ITYPE.GT.300) THEN
               IF (INTCOD.GE.140) THEN
                    CALL ISH3DI(ITYPE,NN,IERROR)
               ELSE
                    CALL ISH3DG(ITYPE,NN,IERROR)
               END IF
          ELSE
               CALL ISH2DG(ITYPE,NN,IERROR)
          END IF
     END IF
     ITYPE1 = ITYPE
     IDENT1 = IDENT
C
     DO 30 INTGPN = 1 , NIP
     LOGIC = BTEST(IYIEL( ELNUM ) , INTGPN)
     IF (LOGIC) THEN
          CALL COORD1(ELNUM,NN,INTGPN,XIP,YIP,ZIP)
          CALL COORD2(ELNUM,NN,INTGPN,NNDF,UXIP,UYIP,UZIP)
          XS = XIP*FMAG + UXIP*DMAG
          YS = YIP*FMAG + UYIP*DMAG
          CALL VIEW2(XS,YS,0.02,11)
     END IF
 30  CONTINUE
C
C ----- DRAW THE MESH
C
     DO 40 K1 = 1 , NLINES
     NODE1 = IVS( K1 )
     NODE2 = IVE( K1 )
     ID1 = NNDF*(NODE1 - 1)
     ID2 = NNDF*(NODE2 - 1)
     XS = X( NODE1 )*FMAG + UTOTAL( ID1 + 1 )*DMAG
     YS = Y( NODE1 )*FMAG + UTOTAL( ID1 + 2 )*DMAG
     XE = X( NODE2 )*FMAG + UTOTAL( ID2 + 1 )*DMAG
     YE = Y( NODE2 )*FMAG + UTOTAL( ID2 + 2 )*DMAG
     IF (IDIM.EQ.3) THEN
          ZS = Z( NODE1 )*FMAG + UTOTAL( ID1 + 3 )*DMAG
          ZE = Z( NODE2 )*FMAG + UTOTAL( ID2 + 3 )*DMAG
     ELSE
          ZS = 0.
          ZE = 0.
     END IF
C
     CALL CLIP(XS,YS,ZS,1.,XE,YE,ZE,1.)
 40  CONTINUE
C
C ----- CLOSE THE PLOT FRAME
C
     IDPLOT = 20
     WRITE(LDEVP , '(I3)') IDPLOT
C
C    > S T A R T   OF   T H E   C O N T O U R I N G   R O U T I N E <
```

```
C
      IF (NOLINE.EQ.0.OR.INCREM.EQ.0) GO TO 1000
C
C ----- EXTRAPOLATE THE VALUES TO BE CONTOURED TO THE NODAL POINTS
C
      CALL EXTRAP(NELEM,NNODES,NNDF,IFLAG1,VALUE)
      NFRAME = 14
C
      DO 200 IFRAME = 1 , NFRAME
      CALL NNUMO
C
C ----- DRAW THE BOUNDARY OF THE MESH
C
      DO 50 K1 = 1 , NLINES
      IRLINE = IAND(IREP(K1),31)
      IF (IRLINE.LE.IR) THEN
           NODE1 = IVS( K1 )
           NODE2 = IVE( K1 )
           ID1 = NNDF*(NODE1 - 1)
           ID2 = NNDF*(NODE2 - 1)
           XS = X( NODE1 )*FMAG + UTOTAL( ID1 + 1 )*DMAG
           YS = Y( NODE1 )*FMAG + UTOTAL( ID1 + 2 )*DMAG
           XE = X( NODE2 )*FMAG + UTOTAL( ID2 + 1 )*DMAG
           YE = Y( NODE2 )*FMAG + UTOTAL( ID2 + 2 )*DMAG
           IF (IDIM.EQ.3) THEN
                ZS = Z( NODE1 )*FMAG + UTOTAL( ID1 + 3 )*DMAG
                ZE = Z( NODE2 )*FMAG + UTOTAL( ID2 + 3 )*DMAG
           ELSE
                ZS = 0.
                ZE = 0.
           END IF
           CALL CLIP(XS,YS,ZS,1.,XE,YE,ZE,1.)
      END IF
 50   CONTINUE
C
C ----- DRAW THE CURVED BOUNDARY OR THE DIE IF THERE IS ONE
C
      IF (NINODE.GT.0) CALL CURVE
C
C ----- IDENTIFY CONTOUR LINES
C
      IVSTR = (IFRAME - 1)*NNODES
      IVEND = IFRAME*NNODES
C
      VMIN = 10.0E50
      VMAX =-10.0E50
      DO 60 K1 = IVSTR + 1 , IVEND
      VMIN = AMIN1(VMIN , VALUE( K1 ))
      VMAX = AMAX1(VMAX , VALUE( K1 ))
 60   CONTINUE
C
C ----- ADJUST THE VMAX AND VMIN SO THAT CONTOURS CLOSE TO THESE VALUES
```

```
C         ARE ALSO PLOTTED.
C
      VMIN1 = VMIN
      VMAX1 = VMAX
      VMAX = VMAX - ABS( VMAX )/50.
      VMIN = VMIN + ABS( VMIN )/50.
C
      VINTR = (VMAX - VMIN)/9.
      VCONT = VMIN
C
      K3 = 0
      DO 160 NCONT = 0 , 9
      K3 = K3 + 1
      VLEGND( K3 ) = VCONT
      LEGEND( K3 ) = NCONT
      NCLS = 0
      DO 90 ELNUM = 1 , NELEM
      CALL ELINFO(ELNUM,ITYPE,NN,IFLAG,ISTART,LINES)
      IF(ITYPE.LT.300) THEN
           ISTART = 1
           ISTOP  = 4
      END IF
      ICODE = 0
      DO 70 K1 = ISTART , ISTOP
      NODE1 = NOP(IS( K1 ) , ELNUM)
      NODE2 = NOP(IE( K1 ) , ELNUM)
      ID1 = NODE1 + IVSTR
      ID2 = NODE2 + IVSTR
      V1  = VALUE( ID1 )
      V2  = VALUE( ID2 )
      VH  = AMAX1(V1 , V2)
      VL  = AMIN1(V1 , V2)
      IF (VCONT.EQ.VL.OR.VCONT.EQ.VH) THEN
           VCONT1 = VCONT + VCONT/10000.
      ELSE
           VCONT1 = VCONT
      END IF
C
      IF (VCONT.GT.VL.AND.VCONT.LT.VH) THEN
           R = (VCONT1 - V1)/(V2 - V1)
           IDENT1 = NNDF*(NODE1 - 1)
           IDENT2 = NNDF*(NODE2 - 1)
           X1 = X( NODE1 )*FMAG + UTOTAL( IDENT1 + 1 )*DMAG
           X2 = X( NODE2 )*FMAG + UTOTAL( IDENT2 + 1 )*DMAG
           Y1 = Y( NODE1 )*FMAG + UTOTAL( IDENT1 + 2 )*DMAG
           Y2 = Y( NODE2 )*FMAG + UTOTAL( IDENT2 + 2 )*DMAG
           IF (ICODE.EQ.0) THEN
                XS = X1 + R*(X2 - X1)
                YS = Y1 + R*(Y2 - Y1)
                CALL GETLIN(NODE1,NODE2,NLIN,LNUM1)
                ICODE = 1
           ELSE
```

```
               NCLS = NCLS + 1
               IF (NCLS.GT.LIMIT) THEN
                    WRITE(IOUT , 1004)LIMIT
                    STOP
               END IF
               XXS( NCLS ) = XS
               YYS( NCLS ) = YS
               XXE( NCLS ) = X1 + R*(X2 - X1)
               YYE( NCLS ) = Y1 + R*(Y2 - Y1)
               CALL GETLIN(NODE1,NODE2,NLIN,LNUM)
               LNEND( NCLS ) = LNUM
               LNSTR( NCLS ) = LNUM1
               ICODE = 0
          END IF
     END IF
70   CONTINUE
90   CONTINUE
C
C ----- SEARCH FOR THE CONTOUR LINES WHICH CROSS THE BOUNDARIES
C
     DO  120 K1 = 1 , NCLS
     IR1 = 0
     IF (LNSTR( K1 ).EQ.0) GO TO 120
     IF (IAND(LREP(LNSTR( K1 )),31).EQ.1) THEN
          XC( 1 ) = XXS( K1 )
          YC( 1 ) = YYS( K1 )
          XC( 2 ) = XXE( K1 )
          YC( 2 ) = YYE( K1 )
          IR1 = 1
          LE = LNEND( K1 )
          LSN = LNSTR( K1 )
          LNSTR( K1 ) = 0
     ELSE IF(IAND(LREP(LNEND( K1 )),31).EQ.1) THEN
          XC( 2 ) = XXS( K1 )
          YC( 2 ) = YYS( K1 )
          XC( 1 ) = XXE( K1 )
          YC( 1 ) = YYE( K1 )
          LE = LNSTR( K1 )
          LSN = LNEND( K1 )
          LNSTR( K1 ) = 0
          IR1 = 1
     END IF
     IF (IR1.EQ.1) THEN
          ICOOR = 2
100       CONTINUE
          DO 110 K2 = 1 , NCLS
          LS2 = LNSTR( K2 )
          IF (K2.EQ.K1.OR.LS2.EQ.0) GO TO 110
          LE2 = LNEND( K2 )
          IF (LS2.EQ.LE) THEN
               ICOOR = ICOOR + 1
               XC( ICOOR ) = XXE( K2 )
```

```
                YC( ICOOR ) = YYE( K2 )
                LE = LNEND( K2 )
                LNSTR( K2 ) = 0
                IF (IAND(LREP( LE ),31).EQ.1) THEN
                     CALL DLINE(XC,YC,ICOOR-1)
                     CALL NUMLIN(LSN,XC(1),YC(1),NNDF,NCONT,IOUT)
                     CALL NUMLIN(LE,XC(ICOOR),YC(ICOOR),NNDF,NCONT,IOUT)
                     GO TO 120
                END IF
                GO TO 100
           ELSE IF(LE2.EQ.LE) THEN
                ICOOR = ICOOR + 1
                XC( ICOOR ) = XXS( K2 )
                YC( ICOOR ) = YYS( K2 )
                LE = LNSTR( K2 )
                LNSTR( K2 ) = 0
                IF (IAND(LREP( LE ),31).EQ.1) THEN
                     CALL DLINE(XC,YC,ICOOR-1)
                     CALL NUMLIN(LSN,XC(1),YC(1),NNDF,NCONT,IOUT)
                     CALL NUMLIN(LE,XC(ICOOR),YC(ICOOR),NNDF,NCONT,IOUT)
                     GO TO 120
                END IF
                GO TO 100
           END IF
 110  CONTINUE
      END IF
 120  CONTINUE
C
C ----- SEARCH FOR THE CONTOUR LINES WHICH FORM A CLOSED LOOP INSIDE
C         THE MESH REGON.
C
      DO  150 K1 = 1 , NCLS
      LS = LNSTR( K1 )
      LE = LS
      IF (LS.NE.0) THEN
           XC( 1 ) = XXS( K1 )
           YC( 1 ) = YYS( K1 )
           XC( 2 ) = XXE( K1 )
           YC( 2 ) = YYE( K1 )
           LE1 = LNEND( K1 )
           LNSTR( K1 ) = 0
           ICOOR = 2
 130       CONTINUE
           DO 140 K2 = 1 , NCLS
           LS2 = LNSTR( K2 )
           IF (K2.EQ.K1.OR.LS2.EQ.0) GO TO 140
           LE2 = LNEND( K2 )
           IF (LS2.EQ.LE1) THEN
                ICOOR = ICOOR + 1
                XC( ICOOR ) = XXE( K2 )
                YC( ICOOR ) = YYE( K2 )
                LE1 = LNEND( K2 )
```

```
                LNSTR( K2 ) = 0
                IF (LE1.EQ.LE.AND.ICOOR.GT.3) THEN
                     CALL DLINE(XC,YC,ICOOR-1)
                     CALL NUMLIN(LE,XC(ICOOR),YC(ICOOR),NNDF,NCONT,IOUT)
                     GO TO 150
                END IF
                GO TO 130
           ELSE IF(LE2.EQ.LE1) THEN
                ICOOR = ICOOR + 1
                XC( ICOOR ) = XXS( K2 )
                YC( ICOOR ) = YYS( K2 )
                LE1 = LNSTR( K2 )
                LNSTR( K2 ) = 0
                IF (LE1.EQ.LE.AND.ICOOR.GT.3) THEN
                     CALL DLINE(XC,YC,ICOOR-1)
                     CALL NUMLIN(LE,XC(ICOOR),YC(ICOOR),NNDF,NCONT,IOUT)
                     GO TO 150
                END IF
                GO TO 130
           END IF
 140  CONTINUE
      END IF
 150  CONTINUE
C
C ----- PLOT THE CONTOUR NUMBERS AND WRITE THE LEGENDS IN DEVICE LDEV4.
C
      VCONT = VCONT + VINTR
 160  CONTINUE
      CALL PLTNUM
C
C ----- CLOSE THE PLOT FRAME
C
      IDPLOT = 20
      WRITE(LDEVP , '(I3)') IDPLOT
C
      WRITE(LDEV4 , 1002)IFRAME,INCREM
      WRITE(LDEV4 , 1003)VMIN1,VMAX1
      WRITE(LDEV4 , 1001)(LEGEND(K1),VLEGND(K1),K1 = 1 , 10)
  200 CONTINUE
 1000 RETURN
 1001 FORMAT(1X,I1,' = ',E11.4,3X,I1,' = ',E11.4,3X,I1,' = ',E11.4,
     1       3X,I1,' = ',E11.4/1X,I1,' = ',E11.4,
     2      3X,I1,' = ',E11.4,3X,I1,' = ',E11.4,3X,I1,' = ',E11.4/
     3       1X,I1,' = ',E11.4,3X,I1,' = ',E11.4////)
 1002 FORMAT(1X,'LEGENDS FOR FRAME NUMBER ',I3,' AT LOAD STEP ',I3)
 1003 FORMAT(1X,'MINIMUM = ',E11.4,3X,'MAXIMUM = ',E11.4)
 1004 FORMAT(1X,'>>>>>>> PROGRAM TERMINATED IN ROUTINE PLOTER DUE TO '/
     1  9X,'EXCEEDING THE ALLOWABLE NUMBER OF COUNTOUR LINE SEGMENTS'/
     2  9X,'INTERNALY SET TO (',I5,')')
      END
```

```
C
C ========================= G E T L I N ==============================
C
      SUBROUTINE GETLIN(NODE1,NODE2,NLIN,LNUM)
      COMMON/GRAPH2/IVS(8000),IVE(8000),ILS(8000),ILE(8000)
C
      DO 100 K1 = 1 , NLIN
      IF (NODE1.EQ.ILS( K1 ).OR.NODE1.EQ.ILE( K1 )) THEN
           IF (NODE2.EQ.ILE( K1 ).OR.NODE2.EQ.ILS( K1 )) THEN
                LNUM = K1
                RETURN
           END IF
      END IF
 100  CONTINUE
      RETURN
      END
```

```
C
C ========================= D L I N E ================================
C
      SUBROUTINE DLINE(XC,YC,NLIN)
      DIMENSION XC(1),YC(1)
C
      XS = XC( 1 )
      YS = YC( 1 )
      DO 100 K1 = 2 , NLIN+1
      XE = XC( K1 )
      YE = YC( K1 )
      CALL CLIP(XS,YS,0.,1.,XE,YE,0.,1.)
      XS = XE
      YS = YE
 100  CONTINUE
C
      RETURN
      END
```

```
C
C ======================== N U M L I N ==============================
C
      SUBROUTINE NUMLIN(LE,XE,YE,NNDF,NCONT,IOUT)
C
C  ==================================================================
C  I                                                                I
C  I   P R O G R A M:                                               I
C  I                                                                I
C  I                                                                I
C  I   NUMLIN PERFORMS THE FOLLOWING FUNCTIONS AT EACH ENTRY POINT  I
C  I                                                                I
C  I       1 - LOCATES AND STORS THE POSITION OF EACH CONTOUR       I
C  I           NUMBER AT (ENTRY NUMLIN)                             I
C  I                                                                I
C  I       2 - PLOTS THE NUMBERS THAT ARE NOT TOO CLOSE             I
C  I           (ENTRY PLTNUM)                                       I
C  I                                                                I
C  I       3 - INITIALIZES NNUM TO ZERO (ENTRY NNUMO)               I
C  I                                                                I
C  I   A R G U M E N T   L I S T:                                   I
C  I                                                                I
C  I                                                                I
C  I   LE    = LINE NUMBER OF WHERE THE CONTOUR LINE STARTS OR      I
C  I             TERMINATES                                         I
C  I                                                                I
C  I   XE    = X-COORDINATE OF THE START (END) OF THE CONTOUR LINE  I
C  I                                                                I
C  I   YE    = Y-COORDINATE OF THE START (END) OF THE CONTOUR LINE  I
C  I                                                                I
C  I   NNDF  = NUMBER OF NODAL DEGREES OF FREEDOM                   I
C  I                                                                I
C  I   NCONT = CONTOUR NUMBER TO BE PLOTED                          I
C  I                                                                I
C  I   IOUT  = OUTPUT DEVICE NUMBER                                 I
C  I                                                                I
C  I                                                                I
C  I   C O N S I D E R A T I O N S:                                 I
C  I                                                                I
C  I                                                                I
C  I   SOME NUMBERS THAT ARE TOO CLOSE TO THE ALREADY PLOTED        I
C  I   NUMBERS WILL NOT BE PLOTED.                                  I
C  I                                                                I
C  ==================================================================
      REAL*8 UTOTAL
      COMMON/GRAPH2/IVS(8000),IVE(8000),ILS(8000),ILE(8000)
      COMMON/INPUT3/X(4000),Y(4000),Z(4000)
      COMMON/MAIN2/UTOTAL(8000)
      COMMON/GRAPH4/XVL,XVR,YVB,YVT,SX,SY
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES,NLIN
      COMMON/GRAPH6/XNUM(100),YNUM(100),NUMVAL(100),NNUM
C
```

```
C ----- D     = DISTANCE OF THE NUMBER FROM THE BOUNDARY OF THE MESH
C ----- NNUM  = NUMBER OF CONTOUR NUMBERS
C ----- LIMIT = MAXIMUM NUMBER OF CONTOUR NUMBERS DUE TO STORAGE
C               LIMITATIONS IN COMMON GRAPH6
C
      D = 1.2*HIGHT/SX
      NNUM = NNUM + 1
      LIMIT = 100
      IF (NNUM.GT.LIMIT) THEN
           WRITE(IOUT , 1001)LIMIT
           STOP
      END IF
C
      ID1 = NNDF*(ILS( LE ) - 1)
      ID2 = NNDF*(ILE( LE ) - 1)
      X1 = X(ILS( LE ))*FMAG + UTOTAL( ID1 + 1 )*DMAG
      X2 = X(ILE( LE ))*FMAG + UTOTAL( ID2 + 1 )*DMAG
      Y1 = Y(ILS( LE ))*FMAG + UTOTAL( ID1 + 2 )*DMAG
      Y2 = Y(ILE( LE ))*FMAG + UTOTAL( ID2 + 2 )*DMAG
      V1 = Y2 - Y1
      V2 = X2 - X1
      THETA = ATAN2(V1 , V2)
      XNUM( NNUM ) = XE + D*SIN( THETA )
      YNUM( NNUM ) = YE - D*COS( THETA )
      NUMVAL( NNUM ) = NCONT
C
 1001 FORMAT(1X,'>>>>>>> PROGRAM TERMINATED IN ROUTINE NUMLIN DUE TO'/
     1   9X,'EXCEEDING THE ALLOWABLE NUMBER OF CONTOUR NUMBERS '/
     2   9X,'INTERNALY SET TO (',I5,')')
      RETURN
C
C
C                 E N T R Y    P L T N U M
C
      ENTRY PLTNUM
C
      IF (NNUM.EQ.0) RETURN
      TOL = 1.6*HIGHT/SX
C
      FPN = FLOAT(NUMVAL( 1 ))
      CALL VIEW3(XNUM( 1 ),YNUM( 1 ),HIGHT,FPN,ANGLE,-1)
C
      WRITE(6,*)'NNUM = ',NNUM
      DO 20 K1 = 2 , NNUM
      ICODE = 0
      X1 = XNUM( K1 )
      Y1 = YNUM( K1 )
      DO 10 K2 = 1 , K1 - 1
      IF (NUMVAL(K2).GE.0) THEN
           X2 = XNUM( K2 )
           Y2 = YNUM( K2 )
           XDIF = X1 - X2
```

```
          YDIF = Y1 - Y2
          TDIF = SQRT(XDIF**2 + YDIF**2)
          IF (TDIF.LT.TOL) ICODE = 1
      END IF
 10   CONTINUE
C
      IF (ICODE.EQ.0) THEN
         WRITE(6,*)'FPN= ',FPN
          FPN = FLOAT(NUMVAL( K1 ))
          CALL VIEW3(X1,Y1,HIGHT,FPN,ANGLE,-1)
      ELSE
          NUMVAL( K1 ) = -1
      END IF
  20  CONTINUE
      RETURN
C
C
C                 E N T R Y     N N U M O
C
      ENTRY NNUMO
      NNUM = 0
      RETURN
      END
```

```
C
C ========================== C L I P ==================================
C
      SUBROUTINE CLIP(X1,Y1,Z1,W1,X2,Y2,Z2,W2)
      INTEGER ZOR,ZAND
      COMMON/GRAPH3/XL,XR,YB,YT,ZF,D
      EQUIVALENCE (IZOR,ZOR),(IZAND,ZAND)
C
      IF (W1.NE.1..OR.W2.NE.1.) THEN
           IF (Z1.GT.ZF.AND.Z2.GT.ZF) THEN
                GO TO 2000
           ELSE IF(Z1.GT.ZF.OR.Z2.GT.ZF) THEN
                CNST = (ZF - Z1)/(Z1 - Z2)
                XX = X1 + CNST*(X1 - X2)
                YY = Y1 + CNST*(Y1 - Y2)
                WW = (1. - ZF/D)
                IF (Z1.GT.ZF) THEN
                     X1 = XX
                     Y1 = YY
                     W1 = WW
                ELSE IF(Z2.GT.ZF) THEN
                     X2 = XX
                     Y2 = YY
                     W2 = WW
                END IF
           END IF
      END IF
C
  90  X1 = X1/W1
      Y1 = Y1/W1
      X2 = X2/W2
      Y2 = Y2/W2
      IZ1 = IZONE(X1 , Y1)
      IZ2 = IZONE(X2 , Y2)
 100  ZOR = IOR(IZ1 , IZ2)
      IF (IZOR.NE.0) GO TO 400
 200  CALL VIEW1(X1,Y1,3)
      CALL VIEW1(X2,Y2,2)
 300  GO TO 2000
 400  ZAND = IAND(IZ1 , IZ2)
      IF (IZAND.NE.0) GO TO 300
      ZAND = IAND(ZOR ,  1)
      IF (IZAND.EQ.0) GO TO 900
      XX = XL
      ICK = 1
 500  YY = Y1 + (Y2 - Y1)/(X2 - X1)*(XX - X1)
 600  IZ = IZONE(XX , YY)
      ZAND = IAND(IZ1 , ICK)
      IF (IZAND.NE.0) GO TO 800
 700  X2 = XX
      Y2 = YY
      IZ2 = IZ
```

```
      GO TO 100
 800  X1 = XX
      Y1 = YY
      IZ1 = IZ
      GO TO 100
 900  ZAND = IAND(ZOR , 2)
      IF (IZAND.EQ.0) GO TO 1000
      XX = XR
      ICK = 2
      GO TO 500
 1000 ZAND = IAND(ZOR , 4)
      IF (IZAND.EQ.0) GO TO 1200
      YY = YB
      ICK = 4
 1100 XX = X1 + (X2 - X1)/(Y2 - Y1)*(YY - Y1)
      GO TO 600
 1200 YY = YT
      ICK = 8
      GO TO 1100
C
 2000 RETURN
      END
```

```
C
C ========================== I Z O N E =================================
C
      FUNCTION IZONE(X,Y)
      COMMON/GRAPH3/XL,XR,YB,YT,ZF,D
      IZONE = 0
      IF (X.LT.XL) IZONE = 1
      IF (X.GT.XR) IZONE = 2
      IF (Y.LT.YB) IZONE = IZONE + 4
      IF (Y.GT.YT) IZONE = IZONE + 8
      RETURN
      END
```

```
C
C ========================= V I E W ================================
C
      SUBROUTINE VIEW(X,Y,IPEN)
      COMMON/GRAPH3/XL,XR,YB,YT,ZF,D
      COMMON/GRAPH4/XVL,XVR,YVB,YVT,SX,SY
      COMMON/GRAPH8/LDEVP
C
      ENTRY VIEW1(X,Y,IPEN)
      XV = SX*(X - XL) + XVL
      YV = SY*(Y - YB) + YVB
      WRITE(LDEVP,'(I3,2F6.3)')IPEN,XV,YV
      RETURN
C
C
      ENTRY VIEW2(X,Y,HIGHT,ISYM)
      IF (X.LT.XL.OR.X.GT.XR.OR.Y.LT.YB.OR.Y.GT.YT) GO TO 100
      ID = 10
      ANGL = 0.
      XV = SX*(X - XL) + XVL
      YV = SY*(Y - YB) + YVB
      WRITE(LDEVP,'(I3,2F6.3)')ID,XV,YV
      WRITE(LDEVP,'(I3,2F6.3)')ISYM,HIGHT,ANGL
 100  RETURN
C
C
      ENTRY VIEW3(X,Y,HIGHT,FPN,ANGLE,ICODE)
      IF (X.LT.XL.OR.X.GT.XR.OR.Y.LT.YB.OR.Y.GT.YT) GO TO 200
      ID = 11
      XV = SX*(X - XL) + XVL
      YV = SY*(Y - YB) + YVB
      WRITE(LDEVP,'(I3,2F6.3)')ID,XV,YV
      WRITE(LDEVP,'(2F6.3)')FPN,ANGLE
      WRITE(LDEVP,'(F6.3,I3)')HIGHT,ICODE
  200 RETURN
C
      END
```

```
C
C ========================== E X T R A P ==============================
C
      SUBROUTINE EXTRAP(NELEM,NNODES,NNDF,IFLAG1,VALUE)
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*8 N
      REAL*4 VALUE
      CHARACTER*1 IYIELD
      INTEGER ELNUM
      COMMON/UTIL1/STRESS(6),STRAIN(6),STRELA(6),CENTER(6),WORK,YIELD
      COMMON/INPUT1/NIPXI,NIPETA,NIPSI,NIP,INTCOD
      COMMON/INPUT2/NOP(20,2000)
      COMMON/INPUTF/MATYPE(10)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/GRAPH2/IVS(8000),IVE(8000),ILS(8000),ILE(8000)
      COMMON/IREP1/IREP(8000),LREP(8000)
      COMMON/EXTRP1/INT33(9),INT22(4)
      DIMENSION SIGXI(9),SIGETA(9),VALUE( 1 ),N(20),STRN(6,9),STRS(6,9),
     1        VOLUMS(9),SHAPE(9,9),CAUCH(6),CAUC(6,9),AWORK(9)
      DATA SIGXI/-1.D0,1.D0,1.D0,-1.D0,0.D0,1.D0,0.D0,-1.D0,0.D0/
      DATA SIGETA/-1.D0,-1.D0,1.D0,1.D0,-1.D0,0.D0,1.D0,0.D0,0.D0/
C
C
      DO 50 K1 = 1 , 14*NNODES
 50   VALUE( K1 ) = 0.
C
      ITYPE1 = 0
      IDENT1 = 0
      DO 400 ELNUM = 1 , NELEM
      CALL ELINFO(ELNUM,ITYPE,NN,IFLAG,ISTART,LINES)
      CALL ELINTM(ELNUM,IDENT,INTCOD,NIPXI,NIPETA,NIPSI,MATNUM,THICK)
      IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
           IF (ITYPE.LT.300) THEN
              NIP = NIPXI*NIPETA
              IF (NIP.EQ.4) THEN
                   A = 1.7320508075688765 3D0
                   IT = 204
              ELSE
                   A = 1.2909944487358060 4D0
                   IT = 209
              END IF
C
              DO 80 K1 = 1 , NN
              XI = SIGXI( K1 )*A
              ETA = SIGETA( K1 )*A
              CALL N2D(XI,ETA,N,IT,IERROR)
C
              IF (NIP.EQ.4) THEN
                   DO 60 K2 = 1 , NIP
 60                SHAPE(INT22( K2 ) , K1) = N( K2 )
              ELSE
                   DO 70 K2 = 1 , NIP
```

```
 70                 SHAPE(INT33( K2 ) , K1) = N( K2 )
               END IF
 80            CONTINUE
               IEND = 4
            ELSE
               GO TO 2000
            END IF
      END IF
      DO 100 INTGPN = 1 , NIP
      IF (MATYPE( MATNUM ).EQ.1) THEN
            CALL IOGET(LDEV1,96,'(A96)',5)
      ELSE
            CALL IOGET(LDEV1,201,'(A201)',6)
            AWORK( INTGPN ) = WORK
      END IF
C
      IF (IFLAG1.EQ.0) THEN
            VOLUMS( INTGPN ) = (STRESS(1)+STRESS(2)+STRESS(4))/3.0D0
      ELSE
            IF (ITYPE.NE.ITYPE1.OR.IDENT.NE.IDENT1) THEN
                  CALL ISH2DG(ITYPE,NN,IERROR)
            END IF
            CALL CAUCHY(ELNUM,ITYPE,NN,NNDF,INTGPN,STRESS,CAUCH)
            VOLUMS( INTGPN ) = (CAUCH(1)+CAUCH(2)+CAUCH(4))/3.0D0
            DO 90 K1 = 1 , IEND
            CAUC(K1 , INTGPN) = CAUCH( K1 )
 90         CONTINUE
      END IF
C
      DO 100 K1 = 1 , IEND
      STRS(K1 , INTGPN) = STRESS( K1 )
      STRN(K1 , INTGPN) = STRAIN( K1 )
 100  CONTINUE
C
      DO 200 K1 = 1 , NN
      NODE = NOP(K1 , ELNUM)
      DO 200 K2 = 1 , IEND
      ID = (K2 - 1)*NNODES + NODE
      DO 200 K3 = 1 , NIP
      VALUE( ID ) = VALUE( ID ) + STRS(K2 , K3)*SHAPE(K3,K1)
 200  CONTINUE
C
      DO 300 K1 = 1 , NN
      NODE = NOP(K1 , ELNUM)
      DO 300 K2 = 1 , IEND
      ID = (IEND + K2 - 1)*NNODES + NODE
      DO 300 K3 = 1 , NIP
      VALUE( ID ) = VALUE( ID ) + STRN(K2 , K3)*SHAPE(K3,K1)
 300  CONTINUE
C
      DO 350 K1 = 1 , NN
      NODE = NOP(K1 , ELNUM)
```

```
      DO 350 K2 = 1 , IEND
      ID = (2*IEND + K2 - 1)*NNODES + NODE
      DO 350 K3 = 1 , NIP
      VALUE( ID ) = VALUE( ID ) + CAUC(K2 , K3)*SHAPE(K3,K1)
  350 CONTINUE
C
      ID1 = 3*IEND*NNODES
      DO 360 K1 = 1 , NN
      ID = ID1 + NOP(K1 , ELNUM)
      DO 360 K3 = 1 , NIP
      VALUE( ID ) = VALUE( ID ) + VOLUMS( K3 )*SHAPE(K3,K1)
  360 CONTINUE
C
      ID1 = (3*IEND + 1)*NNODES
      DO 370 K1 = 1 , NN
      ID = ID1 + NOP(K1 , ELNUM)
      DO 370 K3 = 1 , NIP
      VALUE( ID ) = VALUE( ID ) + AWORK( K3 )*SHAPE(K3,K1)
  370 CONTINUE
      IDENT1 = IDENT
      ITYPE1 = ITYPE
  400 CONTINUE
C
      DO 500 K2 = 1 , 14
      ID1 = (K2 - 1)*NNODES
      DO 500 NODE = 1 , NNODES
      IRNODE = IREP( NODE )/32
      ID = ID1 + NODE
      VALUE( ID ) = VALUE( ID )/FLOAT(IRNODE)
  500 CONTINUE
C
      CALL REWIN
 2000 RETURN
 1000 FORMAT(2A48)
 1001 FORMAT(4A48,A8,A1)
      END
```

```
C
C ==================== C U R V E ====================================
C
      SUBROUTINE CURVE
      REAL*8 T,X,Y
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES,NLIN
      COMMON/GRAPH8/LDEVP
      COMMON/BOUND1/NCURVS
C
C ----- SET THE LINE THICKNESS TO 3
C
      VTHICK = 3.
      ID = 12
      WRITE(LDEVP,'(I3,F6.3)')ID,VTHICK
C
      DO 20 NCURVE = 1 , NCURVS
      DT = 0.05
      T =  0.
      CALL HERMXY(T,X,Y,NCURVE)
      XS = X*FMAG
      YS = Y*FMAG

      DO 10 K1 = 1 , 20
      T = T + DT
      CALL HERMXY(T,X,Y,NCURVE)
      XE = X*FMAG
      YE = Y*FMAG
      CALL CLIP(XS,YS,0.,1.,XE,YE,0.,1.)
      XS = XE
 10   YS = YE
 20   CONTINUE
C
      VTHICK = 2.
      WRITE(LDEVP,'(I3,F6.3)')ID,VTHICK
      RETURN
      END
```

APPENDIX L

INITIALIZER MODULE

```
C
C ==================== B L O C K   D A T A ==========================
C
      BLOCK DATA
      IMPLICIT REAL*8 (A-H,O-Z)
      REAL*4 XII,ETAI,SII,FMAG,DMAG,ETHICK
      CHARACTER*80 GTITLE
      COMMON/MAIN1/U(8000),RE1(8000)
      COMMON/MAIN4/RE(8000)
      COMMON/INPUT7/RX(8000),RY(8000),RZ(8000)
      COMMON/INPUT8/NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM,
     1              NINODE
      COMMON/INPUTB/FAC,FACNEW,FACLOW,FACHIG,ENRG1,NDIVER,ISTOP
      COMMON/INPUTE/ISPB(4000)
      COMMON/INPUTF/MATYPE(10)
      COMMON/INPUTG/IFLAG3,IOINTR,IFPLOT
      COMMON/UTIL3/NREC(3),NWMAX
      COMMON/ADMAT1/AD(81)
      COMMON/DEVICE/LDEV1,LDEV2,LDEV3,LDEV4,LDEV5,LDKEEP,LDEV,LDEVST
      COMMON/ELLIB1/XII(20),ETAI(20),SII(20)
      COMMON/HERM/H(4 , 4),GX(4 , 6),GY(4 , 6)
      COMMON/GRAPH1/IS(62),IF(62)
      COMMON/IREP1/IREP(8000),LREP(8000)
      COMMON/GRAPH5/FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK,NLINES,NLIN
      COMMON/GRAPH7/GTITLE(20)
      COMMON/GRAPH8/LDEVP
      COMMON/EXTRP1/INT33(9),INT22(4)
      COMMON/INPUTA/INFOEL(2,2000),ETHICK(2000)
C
      DATA INFOEL/4000*0/,ETHICK/2000*0./
      DATA ((H(I,J),J=1,4),I=1,4)/2.,-2.,1.,1.,-3.,3.,-2.,-1.,0.,0.,
     1      1.,0.,1.,0.,0.,0./
      DATA U/8000*0.0/,LDEV5/16/,LDEVP/17/
      DATA RE/8000*0.0/,RX/8000*0.0/,RY/8000*0.0/,RZ/8000*0.0/
      DATA LDEV1,LDEV2,LDEV3,LDEV4,LDEVST/1,2,3,4,14/,ISPB/4000*0/
      DATA (XII(K),K=1,20)/-1.,1.,1.,-1.,-1.,1.,1.,-1.,0.,1.,0.,-1.,
     1    -1.,1.,1.,-1.,0.,1.,0.,-1./
      DATA (ETAI(K),K=1,20)/-1.,-1.,1.,1.,-1.,-1.,1.,1.,-1.,0.,1.,
     1     0.,-1.,-1.,1.,1.,-1.,0.,1.,0./
      DATA (SII(K),K=1,20)/-1.,-1.,-1.,-1.,1.,1.,1.,1.,-1.,-1.,-1.,-1.,
     1     0.,0.,0.,0.,1.,1.,1.,1./
      DATA NNODES,NELEM,NNDF,NLINC,MNIT,IFLAG1,IFLAG2,IDIM/0,0,2,1,1,0,
     1     0,2/,IFLAG3,IOINTR,IFPLOT/0,0,0/
      DATA NDIVER,FAC/1,.001/,MATYPE/10*1/,AD/81*0./
      DATA NREC/1,1,1/,NWMAX/0/
C
```

```
C ----- GRAPHICS ELEMENT LINE CONECTIVITY DATA
C
      DATA IS/1,2,3,4,1,5,2,3,4,
     1        1,5,2,6,3,7,4,8,1,2,3,4,5,6,7,8,2,3,1,4,
     2    1,9,2,10,3,11,4,12,5,17,6,18,7,19,8,20,1,13,4,16,3,15,2,14,
     3    1,5,6,2,7,3,8,4,9/
      DATA IE/2,3,4,1,5,2,3,4,1,
     1        5,2,6,3,7,4,8,1,2,3,4,1,6,7,8,5,6,7,5,8,
     2    9,2,10,3,11,4,12,1,17,6,18,7,19,8,20,5,13,5,16,8,15,7,14,6,
     3    5,6,2,7,3,8,4,9,1/
      DATA FMAG,DMAG,HIGHT,ANGLE,NOLINE,ITHICK/1.,1.,0.08,0.,0,2/
      DATA NLINES,NLIN/0,0/
C
C       GAUSSIAN POINT TO NODE CONNECTIVITY DATA FOR NODAL EXTRAPOLATION
C
      DATA INT33/1,3,9,7,2,6,8,4,5/,INT22/1,2,4,3/,IREP/8000*0/
      DATA LREP/8000*0/
      END
```

VITA

Mehrdad Foroozesh was born on September 13, 1962 in Shiraz, Iran. Mr. Foroozesh attended the Louisiana State University in January, 1980. He rceived his bachelor of science degree in civil engineering in December, 1983. Afterwards, Mr. Foroozesh remained at LSU and obtained his master of science degree in civil engineering with emphasis in structural engineering in August of 1988.Mr. Foroozesh is to receive his doctor of philosophy degree in civil engineering in the fall commencement, 1989.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Mehrdad Foroozesh

Major Field: Civil Engineering

Title of Dissertation: A Total Lagrangian Finite Element Analysis for Metal Forming with Application to Metal Extrusion

Approved:

George Z. Voyiadjis

Major Professor and Chairman

F. Glen Hamby

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

7/25/89