AN ABSTRACT OF THE THESIS OF

Min Wang for the degree of Doctor of Philosophy in Mechanical Engineering presented on June 1, 1993.

Title: Three-Dimensional, Nonlinear Viscoelastic Analysis of Laminated Composites - A Finite Element Approach

Abstract approved: _____

Timothy C. Kennedy

Polymeric composites exhibit time-dependent behavior, which raises a concern about their long term durability and leads to a viscoelastic study of these materials. Linear viscoelastic analysis has been found to be inadequate because many polymers exhibit nonlinear viscoelastic behavior. Classical laminate theory is commonly used in the study of laminated composites, but due to the plane stress/strain assumption its application has been limited to solving two dimensional, simple plate problems. A three dimensional analysis is necessary for the study of interlaminar stress and for problems involving complex geometry where certain local effects are important.

The objective of this research is to develop a fully three-dimensional, nonlinear viscoelastic analysis that can be used to model the time-dependent behavior of laminated composites. To achieve this goal, a three-dimensional finite element computer program has been developed. In this program, 20-node isoparametric solid elements are used to model the individual plies. The three-dimensional constitutive equations developed for numerical calculations are based on the Lou-Schapery one-dimensional nonlinear viscoelasticity model for the uniaxial stress case. The transient creep compliance in the viscoelastic model is represented as an exponential series plus a steady-flow term, which allows for a simplification of the numerical procedure for

handling hereditary effects. A cumulative damage law for three dimensional analysis was developed based on the Brinson-Dillard two-dimensional model to predict failure initiation.

Calculations were performed using this program in order to evaluate its performance in applications involving complex structural response. IM7/5260-H Graphite/Bismaleimide and T300/5208 Graphite/Epoxy were the materials selected for modeling the time-dependent behavior. The cases studied include: 1) Tensile loading of unnotched laminates; 2) bending of a thick laminated plate; and 3) tensile loading of notched laminates. The analysis emphasized the study of the traction-free edge-effect of laminated composites, stress distribution around a circular hole, and stress redistribution and transformation in the layers. The results indicate that the stress redistributions over time are complicated and could have a significant effect on the long-term durability of the structure.

THREE-DIMENSIONAL, NONLINEAR VISCOELASTIC ANALYSIS OF LAMINATED COMPOSITES - A FINITE ELEMENT APPROACH

by

Min Wang

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed June 1, 1993

Commencement June 1994

APPROVED:

Redacted for Privacy

Associate Professor of Mechanical Engineering in charge of major

Redacted for Privacy

Head of department of Mechanical Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented June 1, 1993

Typed by the author for Min Wang

ACKNOWLEDGEMENTS

First and foremost, I am especially thankful to my major professor, Dr. Timothy C. Kennedy, for his encouragement, advice, confidence, patience, and support throughout my five year Ph.D program. He involved me in research which is very suitable for my interests and for my future career. I thank him for all of this.

I would like to thank Dr. Charles E. Smith, Dr. Clarence A. Calder, Dr. Ernest G. Wolff and Dr. Thomas West for serving as members of my graduate committee as well as their kind advice and encouragement. I would like to thank all of the professors who taught me during my graduate education. Special thanks goes to Dr. James R. Welty. I thank him for his kind advice during my application to this department, which was very encouraging. I would also like to extend my appreciation to the Department of Mechanical Engineering for financial support during my graduate study.

I would also like to thank those around me in my seven years in the OSU Mechanical Engineering Department. Thanks especially go to Nikki Jones for her kind help and valuable discussions whenever I needed help in my English writing. Thanks also to all other fellow students for being so friendly and supportive.

Special thanks go to my host family, Aleita and Alan Holcombe. I thank them for their friendship and assistance, for spending time to read my thesis, for sharing pleasure during the happy times and providing moral support during the tough times.

Thanks are due my parents for their love, understanding and encouragement. Special thanks are also due to my parents-in-law. Without their help taking care of my child, I could not have completed my education so successfully. Finally, I would like to thank my husband, Kaijun Lin for his encouragement, patience, understanding and love.

This work was supported in part by Boeing Commercial Airplane Group with Ramesh Khanna as project monitor.

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

Page

LIST OF FIGURES (continued)

Page

LIST OF FIGURES (continued)

LIST OF APPENDICES FIGURES

THREE-DIMENSIONAL, NONLINEAR VISCOELASTIC ANALYSIS OF LAMINATED COMPOSITES - A FINITE ELEMENT APPROACH

CHAPTER 1

INTRODUCTION

Modern aerospace industries are intensely interested in using composite materials with polymeric matrices when a need exists for weight-critical and stiffness-critical structures. However, in addition to the advantages of high specific strength and stiffness, polymeric-matrix composites present time-dependent behavior in strain-stress response during a long term load. That is, under long-term loads polymer composites no longer exhibit linear elastic behavior. They exhibit a continuous increase of deformation under constant long term load, which will cause stress redistribution inside the composites and may lead to delayed failure of the structures. Thus, the long term durability of polymer composite components becomes a major concern for designers for use in critical designs. To accurately predict the long-term behavior of these structures, it is important for the designer to have analysis tools capable of modeling the viscoelastic response of complex structures composed of fiber-reinforced plastics.

1.1 Review of Viscoelasticity Study of Creep

The study of viscoelasticity has existed for more than 100 years. A systematic observation of the creep phenomenon was first reported in 1834 by Vicat. Andrade proposed the first creep law in 1910, and then empirical equations have been continuously proposed based on experimental observations. Of these empirical equations, the Norton-Bailey power law, Ludwik stress exponential law, and Prandtl-Nadai hyperbolic sine law have been widely used to calculate steady creep for a uniaxial state of stress (Findley, W.N., 1975. Shames, I.H. and Cozzarelli, F.A., 1992). These three expressions are given as follows:

$$\varepsilon_s(t) = A\tau_0^n t \tag{1.1}$$

$$\varepsilon_s(\tau_0) = Be^{\alpha\tau_0} t \tag{1.2}$$

$$\varepsilon_s(\tau_0) = C\sinh(\beta\tau_0) t \tag{1.3}$$

where the subscript s denotes steady creep, τ_0 the constant applied stress, and n in the Equation (1.1) denotes the stress power. A and B in Equation (1.1) and (1.2) are reciprocal viscosity coefficients. The Norton-Bailey power law and Ludwik stress exponential law are the most widely used equations for the steady creep components under low stress and give good agreement with experimental data. The Prandtl-Nadai hyperbolic sine law represents a behavior which is nearly linear for small stresses and nonlinear for large stresses.

However, the stress-strain-time relations reviewed above are primarily empirical. Most were developed to fit experimental creep curves obtained under constant stress and constant temperature. The actual behavior of materials has shown that the strain at a given time depends on the past history of the stress, not just on its final value. Thus the creep phenomenon is affected by the magnitude and sequence of stresses or strains in all of the past history of the material. Based on this fact, various mathematical methods have been suggested to represent the time dependence or viscoelastic behavior of materials. In the mathematical models, creep is represented by means of a linear differential method and an integral operator representation. For example, the following integral

$$\varepsilon(t) = \int_0^t J(t - \xi)\frac{\partial\sigma(\xi)}{\partial\xi} d\xi,$$

called a hereditary integral, was first suggested by Volterra (Findley, W.N, 1975), where ξ is any arbitrary time between 0 and t, representing past time. The kernel

function of the integral, $J(t-\xi)$ is a memory function which describes the stress history dependence of strain. Both the differential operator method and the integral representation can be easily generalized to a multiaxial state of stress.

The creep models discussed above are mostly limited to the linear viscoelastic range. However, many polymers exhibit nonlinear viscoelastic behavior. Thus, if a structurally efficient design is to be realized, it is necessary to provide a more accurate representation than is possible by assuming linear behavior. There has been considerable effort to develop a general constitutive equation for nonlinear viscoelastic material since late 1950's. Green, Rivlin and Spencer proposed a general constitutive equation where stress-strain relations are represented in the form of a sum of a multiple integral (Green, A.E. and Rivlin, R.S., 1957. Green, A.E., Rivlin, R.S. and Spencer, A.J.M., 1959). The advantage of the multiple integral is that it is very appealing theoretically since it is not limited to a particular material or class of materials, and permits one to construct approximations with respect to any order of nonlinearity desired. However, Schapery stated that the multiple integral representation becomes impractical with strong nonlinearities, unless one assumes the kernels can be written in terms of products of a single function, and it does not take advantage of certain simplicity that exists in the mechanical behavior of many polymers and non-polymers. Based on this later point, Schapery derived a single-integral constitutive equation from thermodynamic principles of irreversible processes (Schapery, R.A., 1966. Schapery, R.A. 1969) to describe nonlinear viscoelastic behavior of materials. In this equation the nonlinearity is contained in a reduced time and nonlinear coefficients, where the reduced time and nonlinear coefficients are function of stress and time in the creep formulation. Further discussion of the Schapery model is given in chapter 2.

1.2 Applications of Viscoelasticity in Composites Material

Along with the increased use of composite materials, especially the increased use of fiber reinforced polymeric matrices composites in aerospace and other

transportation industries, the viscoelastic study of composites becomes important due to the time dependent behavior contributed by the polymer matrix of the composites.

Studies of time-dependent behavior of composite materials have been very popular subjects. A number of studies based on different theories with different approaches have been presented. Wang Y.Z. and Tsai T. J. conducted an analysis of the quasi-static and dynamic response of a linear viscoelastic plate based on classical plate theory, using finite element method (Wang, Y.Z. and Tsai, T.J., 1988). Stango and Nelson developed an analytical procedure for representing viscoelastic constitutive equations for fiber composites, where the composite creep compliance and relaxation moduli functions are in convolutional integral form (Stango, R.J., Wang, S.S. and Nelson, C.R., 1989). Chulya and Walker developed a new integration algorithm for elasto-plastic creep and unified viscoplastic theories including continuum damage (Chulya, A. and Walker, D.P., 1991).

During the past decade, a number of these analyses have been developed based on Schapery's viscoelasticity models (Lou, Y.C., and Schapery, R.A., 1971. Schapery, R.A., 1974). Tuttle and Brinson developed a numerical procedure for the prediction of nonlinear viscoelastic response of laminated composites based on classical lamination theory (Tuttle, M.E. and Brinson, H.F., 1986). Ha and Springer developed a viscoelastic/viscoplastic analysis for composites at elevated temperature, also using classical lamination theory (Ha, S.K. and Springer, G.S., 1989). Henriksen developed a two-dimensional finite element analysis for nonlinear viscoelastic behavior of an isotropic material (Henriksen, M., 1984). Roy and Reddy presented a similar analysis including large displacements and moisture diffusion (Roy, S. and Reddy, J.N., 1988. Roy, S. and Reddy, 1988). Lin and Hwang developed a two-dimensional, finite element analysis for the linear viscoelastic response of anisotropic materials (Lin, K.Y., and Hwang, I.H., 1989). Lin and Yi presented a similar analysis for generalized plane-strain conditions (Lin, K.Y. and Yi, S., 1991). Results of Horoschenkoff's test for PEEK and Epoxy Matrices showed that the maximum difference between measured and approximated creep strain based on Schapery's uniaxial tensile stress model is less than 4% (Horoschenkoff, A., 1990).

There is very limited work presented for the study of delayed failure of composites. Brinson, Dillard and Morris have developed a linear cumulative damage rule which is based on a modification of the Tsai-Hill failure criterion for two dimensional problems (Dillard, D.A., and Brinson, H.F., 1983). This will be discussed in more detail in chapter 2.

1.3 Objectives of This Research

Based on the above discussions, it is seen that the viscoelastic behavior of composites is currently intensively studied. The primary analysis tool for composite material structural applications is the finite element method. However, there are no results published for three-dimensional, nonlinear viscoelastic analysis of laminated composites yet. Certain aspects of the response of laminated composites are beyond the modeling capabilities of classical laminated plate theory. These include modeling delamination of plies and analyzing stress states in complex structural details such as notch laminate. For these problems a fully three-dimensional analysis is required. Thus, the objective of this research is to develop a fully three-dimensional, nonlinear viscoelastic analysis that can be used to model the time-dependent behavior of laminated composites.

To achieve this goal, three-dimensional constitutive equations were developed based on Lou and Schapery's nonlinear vicoelasticity model for the uniaxial stress case. Then considerable work has been done to make these constitutive equations suitable for an efficient numerical analysis procedure. Details of the viscoelasticity theoretical development are given in Chapter 2. Due to the complexity of the analysis the finite element method approach was chosen. The basic finite element theory and its applications in laminated composites are given in Chapter 3. Chapter 4 describes the algorithms and organization of the computer program. In chapter 5, calculations were performed to evaluate the program's performance in applications involving complex structural response and to bring out certain physical features that influence design of composite structures. The cases studied in chapter 5 include: 1) Tensile

loading of unnotched laminates; 2) bending of a thick laminated plate; and 3) tensile loading of notched laminates. These problems were chosen to be analyzed because they have been popular subjects of several studies, and they are typical structures appearing in practical designs. While the conclusions drawn in each case apply to that particular problem, they are valuable in the design of similar laminates. The viscoelastic materials to be modeled are IM7/5260-Graphite/Bismileimide and T300/5208-Graphite/Epoxy. They are chosen because the availability of viscoelastic property functions, and because these materials are widely adapted in current aerospace and other advanced designs.

CHAPTER 2

TIME-DEPENDENT BEHAVIOR OF ORTHOTROPIC MATERIALS

Viscoelastic materials exhibit time-dependent behavior in their load-deformation response. This time-dependent behavior is studied in viscoelasticity. In this chapter, basic concepts in viscoelasticity are introduced in Section 2.1. Section 2.2 is devoted to developing constitutive equations to model time-dependent behavior of composite materials. In Section 2.3, the theory of delayed failure of composites is presented.

2.1 Background on Viscoelasticity Theory

As its name implies, viscoelasticity combines elasticity and viscosity (viscous flow), and is concerned with materials which exhibit time-dependent strain effects in response to applied stresses.

2.1.1 Elastic Behavior

For an elastic material, the stress-strain relationship of the material follows a linear Hooke's law. When the material is loaded, an immediate elastic strain response is obtained upon loading, and the strain then stays constant as long as the stress is fixed and disappears immediately upon removal of the load.

2.1.2 Viscoelastic Behavior

Some materials exhibit elastic action upon loading followed by a slow and continuous increase of strain at a decreasing rate. When the stress is removed, a continuously decreasing strain (delayed recovery) follows an initial elastic recovery. Such materials are significantly influenced by the rate of loading, and they are called

viscoelastic or time-dependent materials. Figure 2.1 illustrates some common phenomena of many viscoelastic materials: a) instantaneous elasticity, b) creep under constant stress, c) stress relaxation under constant strain, d) instantaneous recovery, e) delayed recovery, and f) permanent set.

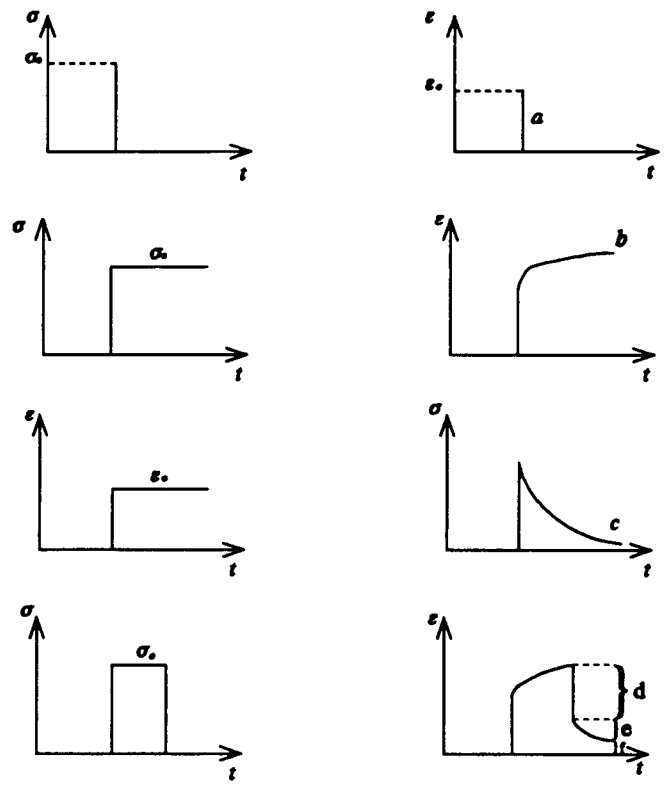


Figure 2.1 Phenomena common to many viscoelastic materials: a. Instantaneous elasticity, b. Creep under constant stress, c. Stress relaxation under constant strain, d. Instantaneous recovery, e. Delayed recovery, f. Permanent set.

2.1.3 Creep

Creep is one of the major subjects studied by viscoelasticity theory, and it can be described as a slow continuous deformation of a material under constant stress. In general, creep may be described in terms of three different stages illustrated in Figure 2.2. The first stage in which creep occurs at a decreasing rate is called primary creep. In stage two, called the secondary stage, creep proceeds at nearly constant rate. In stage three, called third or tertiary stage, creep occurs at an increasing rate and terminates in fracture.

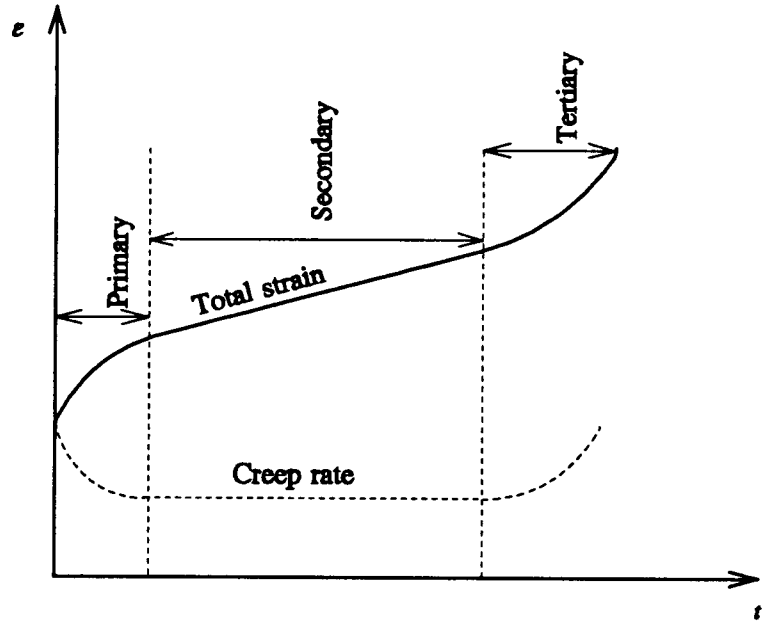


Figure 2.2 Three stages of creep.

Plastics, wood, natural and synthetic fibers, concrete and metals at elevated temperature are some of the materials showing viscoelastic behavior. Recently there have been tremendous advances in the development of composite materials. Some fiber reinforced polymer materials have been incorporated into aircraft designs. These materials occasionally work under high temperature and are subjected to long-term loads. Due to the time-dependent behavior of the polymer matrix, the components

made of composite materials may fail after serving for a period of time even though the external applied load did not exceed the ultimate strength of the materials. The objective of this research is to apply viscoelasticity theory in the development of numerical models to predict the time-dependent behavior of composite materials. It is hoped that this research will contribute to practical designs.

2.1.4 Basic Viscoelastic Elements

Viscoelastic behavior can be physically described by a model composed of a linear spring (elastic part) and a linear dash-pot (viscous part). Two very basic models are the Maxwell fluid model where spring and dashpot are connected in series (Figure 2.3a), and the Kelvin solid model where spring and dashpot are connected in parallel (Figure 2.4a). R in Figure 2.3 and Figure 2.4 can be interpreted as a linear spring constant or a Young's modulus, and η as the coefficient of viscosity.

For a Maxwell fluid model, it can be shown that under a constant stress σ_o the relationship between strain and stress has the following form

$$\varepsilon(t) = \frac{\sigma_o}{R} + \frac{\sigma_o}{\eta} t. \tag{2.1}$$

This result is shown in Figure 2.3b. Under a constant strain ε_0 for which the initial stress is σ_0, the stress relaxation for a Maxwell model can be expressed as

$$\sigma(t) = \sigma_0 e^{-R/\eta} = R\varepsilon_0 e^{-R/\eta} \tag{2.2}$$

This phenomenon is shown in Figure 2.3c.

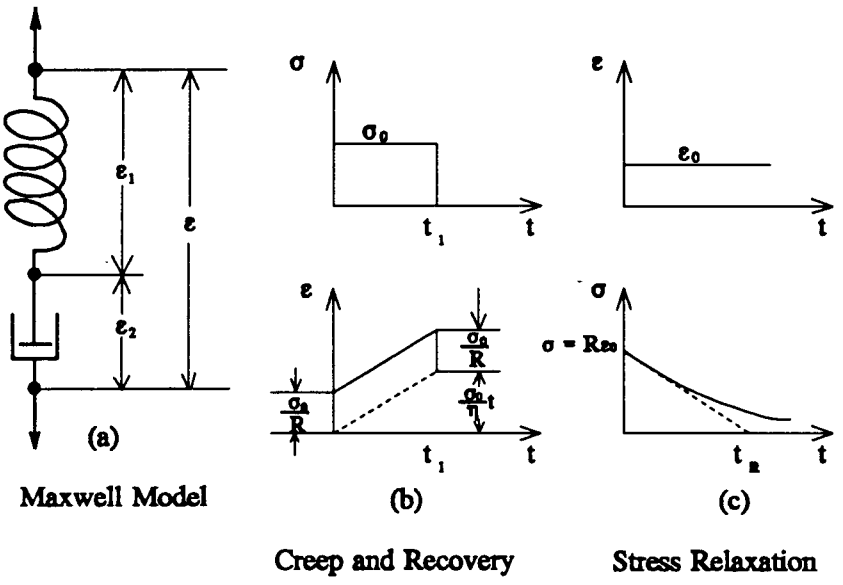


Figure 2.3 Behavior of a Maxwell model.

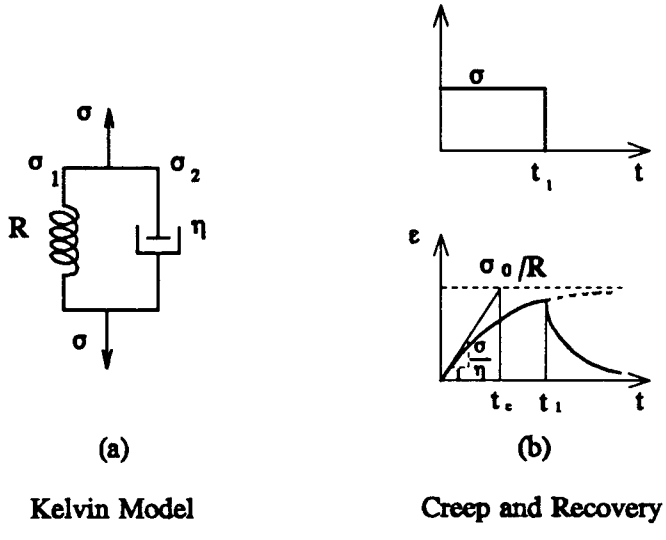


Figure 2.4 Behavior of a Kelvin model.

For a Kelvin solid model, under a constant stress σ_0 at $t = 0$, the strain response to the stress can be expressed as

$$\varepsilon = \frac{\sigma_0}{R}\left(1 - e^{-Rt/\eta}\right). \tag{2.3}$$

and is shown in Figure 2.4b. Upon the removal of the stress at $t = t_1$, strain will be recovered and the recovery can be described by

$$\varepsilon = \frac{\sigma_0}{R} e^{-Rt/\eta}\left[e^{Rt_1/\eta} - 1\right], \quad t > t_1. \tag{2.4}$$

and illustrated in Figure 2.4b. The time t_c in Figure 2.4 is called retardation time. It is the time that would take the creep strain to the asymptotic value σ_0/R if the strain were to increase linearly at its initial rate σ_0/η. It can be shown that $t_c = \eta/R$.

The response of the Kelvin model to an abruptly applied stress is as follows. The stress is at first carried entirely by the viscous element, η. Under the stress the viscous element then elongates, thus transferring a greater and greater portion of the load to the elastic element, R. Finally the entire stress is carried by the elastic element. The behavior just described is appropriately called delayed elasticity.

However, neither the Maxwell nor Kelvin model can fully represent the behavior of most viscoelastic materials when they are used individually (Findley, W.N., 1975). More complicated models, in which several Kelvin models or Maxwell models are connected in series, or in parallel, or in any other mixed combination, are often used to describe a particular material behavior. For example, several Kelvin models in series are called generalized Kelvin Models in series (Figure 2.5). The creep strain of generalized Kelvin models in series under constant stress σ_0 can be obtained by considering that the total strain is the sum of the creep strain of each individual Kelvin model. Thus the creep strain of the generalized Kelvin models has the following form (Findley, 1975)

$$\varepsilon(t) = \sigma_0 \sum_{i=1}^{n} \phi_i (1 - e^{-t/t_c^i}), \tag{2.5}$$

where $t_c^i = \eta_i/R_i$, are the retardation times. Similarly, if Maxwell models are connected in parallel, they are called generalized Maxwell models. Figure 2.5a shows generalized Kelvin models in series, and Figure 2.5b shows another generalized model for elastic response, viscous flow and delayed elasticity with multiple retardation times.

The generalized Kelvin model is more convenient than the generalized Maxwell model for viscoelastic analysis in cases where the stress history is prescribed. The generalized Maxwell model is, however, the more convenient in cases where the strain history is prescribed.

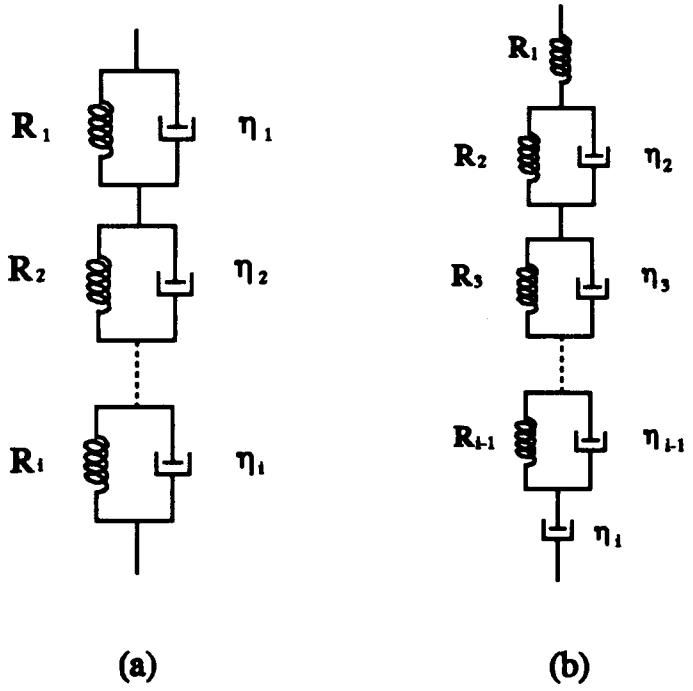


Figure 2.5. (a) Generalized Kelvin models in series. (b) Generalized model for elastic response, viscous flow and delayed elasticity.

2.2 Viscoelastic Constitutive Equations for Composites

The time-dependent behavior of viscoelastic materials must be expressed by a constitutive equation which includes time as a variable in addition to the stress and strain variables. Considerable work has been done to study the creep behavior of various materials and several equations have been proposed since the early nineteenth century. The three-dimensional constitutive equations developed in this study are based on Lou and Schapery's one dimensional nonlinear viscoelasticity model for the uniaxial stress case (Lou and Schapery, 1971).

According to Lou and Schapery, the nonlinear viscoelastic equation for strain in response to a uniaxial loading under constant temperature has the form of

$$\varepsilon^t = g_0^t D_0 \sigma^t + g_1^t \int_0^t D_c(\psi^t - \psi^\tau) \frac{\partial}{\partial \tau}\left(g_2^\tau \sigma^\tau\right) d\tau + \theta^t \qquad (2.6)$$

where D_0 is the elastic (time-independent) compliance, and $D_c(\psi^t - \psi^\tau)$ the creep (time-dependent) compliance. ψ^t and ψ^τ are called reduced times which are implicit functions of stress given by

$$\psi^t = \int_0^t \frac{du}{a}, \quad \psi^\tau = \int_0^\tau \frac{du}{a}, \qquad (2.7)$$

where a is a shift factor which physically modifies viscosity as a function of temperature and stress. θ represents the hygrothermal component of strain. The superscript t or τ denotes the time at which a quantity is evaluated. The quantity a and the compliance coefficients g_0^t, g_1^t and g_2^t are material properties that are generally functions of stress and temperature. At constant temperature, they are functions of stress which, in turn, are functions of time. Physically, g_0^t defines stress and temperature effects on elastic compliance. g_1^t and g_2^t define stress and temperature effects on transient (or creep) compliance. Equation (2.6) will be reduced to the linear viscoelastic creep equation by defining $g_0 = g_1 = g_2 = a = 1$. Equation (2.6) physically

represents the Kelvin elements in series and an elastic spring element. Based on the discussion of thermodynamic theory (Schapery, 1969), and molecular models (Ferry, 1961), and the characteristics of Kelvin models discussed previously (Equation 2.5), Schapery (Schapery, 1969) suggested that the creep compliance can be decomposed into two parts as

$$D_c(\psi^t-\psi^\tau) = \sum_{r=1}^{N} D_r\left[1-e^{-\lambda_r(\psi^t-\psi^\tau)}\right] + D_f(\psi^t-\psi^\tau) \tag{2.8}$$

where the D_r and D_f are positive constants. The term D_f is called the flow component, and leads to a residual strain after the removal of the load. Substituting Equation (2.8) into Equation (2.5), the constitutive equation becomes

$$\begin{aligned}\varepsilon^t = {} & g_0^t D_0 \sigma^t + g_1^t g_2^t \sigma^t \sum_{r=1}^{N} D_r - g_1^t \int_0^t \sum_{r=1}^{N} D_r e^{-\lambda_r(\psi^t-\psi^\tau)} \frac{\partial}{\partial\tau}\left(g_2^\tau \sigma^\tau\right) d\tau \\ & + g_1^t D_f \int_0^t (\psi^t-\psi^\tau) \frac{\partial}{\partial\tau}\left(g_2^\tau \sigma^\tau\right) d\tau + \theta^t.\end{aligned} \tag{2.9}$$

In this equation τ is any arbitrary time between *0* and *t*, representing past time. This shows that the strain at any given time depends on all that has happened before - in the entire stress history $\sigma^\tau(\tau)$. Thus, the integrations in Equation (2.9) are called hereditary integrals.

To establish a general equation similar to Equation (2.9) for a three-dimensional anisotropic case, a set of contracted single notation is introduced to define the stress and strain components as follows:

$$
\begin{aligned}
\sigma_1 &= \sigma_{11}, \quad & \varepsilon_1 &= \varepsilon_{11}, \\
\sigma_2 &= \sigma_{22}, & \varepsilon_2 &= \varepsilon_{22}, \\
\sigma_3 &= \sigma_{33}, & \varepsilon_3 &= \varepsilon_{33}, \\
\sigma_4 &= \sigma_{23}, & \varepsilon_4 &= 2\varepsilon_{23}, \\
\sigma_5 &= \sigma_{13}, & \varepsilon_5 &= 2\varepsilon_{13}, \\
\sigma_6 &= \sigma_{12}, & \varepsilon_6 &= 2\varepsilon_{12},
\end{aligned}
\tag{2.10}
$$

where σ_{ij} and ε_{ij} are tensor notations.

Now for three-dimensional anisotropic material, Equation (2.9) becomes

$$
\begin{aligned}
\varepsilon_i^t = \sum_{j=1}^{k} \Big[& g_{0ij}^t D_0 \sigma_j^t + g_{1ij}^t g_{2ij}^t \sigma_j^t \sum_{r=1}^{N} D_{rij} - g_{1ij}^t \int_0^t \sum_{r=1}^{N} D_{rij} e^{-\lambda_{rij}\left(\psi_{ij}^t - \psi_{ij}^\tau\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^\tau \sigma_j^\tau\right) \\
& + g_{1ij}^t D_{fij} \int_0^t \left(\psi_{ij}^t - \psi_{ij}^\tau\right) \frac{\partial}{\partial \tau}\left(g_{2ij}^\tau \sigma_j^\tau\right) d\tau \Big] + \theta_i^t
\end{aligned}
\tag{2.11}
$$

where

$$
\psi_{ij}^t = \int_0^t \frac{du}{a_{ij}}, \quad \psi_{ij}^\tau = \int_0^\tau \frac{du}{a_{ij}}. \tag{2.12}
$$

Equation (2.11) is difficult to deal with because it requires information from the entire stress history. The effort next is to rewrite Equation (2.11) into an equivalent form so that a numerical method can be easily applied. Consider the first integral in Equation (2.11), and name it q_{rij}^t, then

$$
q_{rij}^t = \int_0^t e^{-\lambda_{rij}\left(\psi_{ij}^t - \psi_{ij}^\tau\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^\tau \sigma_j^\tau\right) d\tau. \tag{2.13}
$$

Breaking this integral into two parts, q_{rij}^t can be written as

$$
\begin{aligned}
q_{rij}^{t} &= \int_{0}^{t-\Delta t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right) d\tau \\
&\quad + \int_{t-\Delta t}^{t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right) d\tau \\
&= I_1 + I_2
\end{aligned}
\tag{2.14}
$$

where

$$
I_1 = \int_{0}^{t-\Delta t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right) d\tau \tag{2.15}
$$

and

$$
I_2 = \int_{t-\Delta t}^{t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right) d\tau . \tag{2.16}
$$

Defining

$$
\Delta\psi^{t} = \int_{t-\Delta t}^{t} \frac{du}{a_{ij}}, \tag{2.17}
$$

I_1 can be written as

$$
\begin{aligned}
I_1 &= \int_{0}^{t-\Delta t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\Delta\psi_{ij}^{t}+\Delta\psi_{ij}^{t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right) d\tau \\
&= e^{-\lambda_{rij}^{t}\Delta\psi_{ij}^{t}} \int_{0}^{t-\Delta t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\Delta\psi_{ij}^{t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right) d\tau .
\end{aligned}
\tag{2.18}
$$

Because

$$\begin{aligned}\psi_{ij}^{t} - \Delta\psi_{ij}^{t} &= \int_0^t \frac{du}{a_{ij}} - \int_{t-\Delta t}^{t} \frac{du}{a_{ij}} \\ &= \int_0^{t-\Delta t} \frac{du}{a^{ij}} + \int_{t-\Delta t}^{t} \frac{du}{a_{ij}} - \int_{t-\Delta t}^{t} \frac{du}{a_{ij}} \\ &= \int_0^{t-\Delta t} \frac{du}{a_{ij}} \\ &= \psi_{ij}^{t-\Delta t},\end{aligned} \tag{2.19}$$

the integral I_1 now becomes

$$I_1 = e^{-\lambda_{rij}\Delta\psi_{ij}^{t}} \int_0^{t-\Delta t} e^{-\lambda_{rij}\left(\psi_{ij}^{t-\Delta t}-\psi_{ij}^{\tau}\right)} \frac{\partial}{\partial\tau}\left(g_{2ij}^{\tau}\,\sigma_j^{\tau}\right)d\tau. \tag{2.20}$$

Note that the integration part in Equation (2.20) has the identical form with the q_{ij}^{t} defined in Equation (2.13) but with different integration limits. This leads to the following results:

$$I_1 = e^{-\lambda_{rij}\Delta\psi_{ij}^{t}} q_{rij}^{t-\Delta t}. \tag{2.21}$$

Now consider the second integral I_2 in Equation (2.14). Integration by parts gives

$$I_2 = \frac{a_{ij}^{t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\psi_{ij}^{t}\right)}}{\lambda_{rij}} \frac{\partial}{\partial\tau}\left(g_{2ij}^{\tau}\sigma^{\tau}\right)\Big|_{t-\Delta t}^{t} - \int_{t-\Delta t}^{t} \frac{a_{ij}^{t} e^{-\lambda_{rij}\left(\psi_{ij}^{t}-\psi_{ij}^{t}\right)}}{\lambda_{rij}} \frac{\partial^2}{\partial\tau^2}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right)d\tau. \tag{2.22}$$

Assuming that $g_{2ij}\sigma_j$ is linear over Δt (i.e., $\partial^2(g_{2ij}^{t}\sigma_j^{t})/\partial\tau^2 = 0.$), then

$$I_2 = \frac{1 - e^{-\lambda_{rij}\Delta\psi_{ij}^{t}}}{\lambda_{rij}}\left(\frac{g_{2ij}^{t}\sigma_j^{t} - g_{2ij}^{t-\Delta t}\sigma_j^{t-\Delta t}}{\Delta\psi^{t}}\right) \tag{2.32}$$

when Δt is sufficiently small. Now, q_{rij}^{t} can be finally written as

$$q_{rij}^{t} = e^{-\lambda_{rij}\Delta\psi_{ij}^{t}} q_{rij}^{t-\Delta t} + \Gamma_{rij}\left(g_{2ij}^{t}\sigma_{j}^{t} - g_{2ij}^{t-\Delta t}\sigma_{j}^{t-\Delta t}\right) \tag{2.24}$$

where

$$\Gamma_{rij} = \frac{1 - e^{-\lambda_{rij}\Delta\psi_{ij}^{t}}}{\lambda_{rij}\Delta\psi_{ij}^{t}}. \tag{2.25}$$

From a computational standpoint, Equation (2.24) is much easier to deal with than Equation (2.11) because Equation (2.24) requires only a knowledge of quantities at the current time step t and the previous time step t-Δt, whereas Equation (2.11) requires a knowledge of quantities over the complete history of the response of the material.

Likewise, if similar procedures are applied, the second integral in Equation (2.11) can be defined as

$$q_{fij}^{t} = \int_{0}^{t}\left(\psi_{ij}^{t} - \psi_{ij}^{\tau}\right)\frac{\partial}{\partial\tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right)d\tau, \tag{2.26}$$

Breaking this integral into two parts, q_{fij}^{t} can be written as

$$\begin{aligned} q_{fij}^{t} &= \int_{0}^{t-\Delta t}\left(\psi_{ij}^{t} - \psi_{ij}^{\tau}\right)\frac{\partial}{\partial\tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right)d\tau \\ &\quad + \int_{t-\Delta t}^{t}\left(\psi_{ij}^{t} - \psi_{ij}^{\tau}\right)\frac{\partial}{\partial\tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right)d\tau \\ &= I_{3} + I_{4} \end{aligned} \tag{2.27}$$

where

$$I_{3} = \int_{0}^{t-\Delta t}\left(\psi_{ij}^{t} - \psi_{ij}^{\tau}\right)\frac{\partial}{\partial\tau}\left(g_{2ij}^{\tau}\sigma_{j}^{\tau}\right)d\tau \tag{2.28}$$

and

$$I_4 = \int_{t-\Delta t}^{t} \left(\psi_{ij}^{t} - \psi_{ij}^{\tau}\right) \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right) d\tau \ . \tag{2.29}$$

Defining

$$\psi_{ij}^{t} = \psi_{ij}^{t-\Delta t} + \Delta\psi_{ij}^{t} \ , \tag{2.30}$$

I_3 can be written as

$$\begin{aligned} I_3 &= \int_0^{t-\Delta t} \left(\psi_{ij}^{t-\Delta t} + \Delta\psi_{ij}^{t} - \psi_{ij}^{\tau}\right) \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right) d\tau \\ &= \int_0^{t-\Delta t} \left(\psi_{ij}^{t-\Delta t} - \psi_{ij}^{\tau}\right) \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right) d\tau \\ &\quad + \Delta\psi_{ij}^{t} \int_0^{t-\Delta t} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right) d\tau \ . \end{aligned} \tag{2.31}$$

Note that the first integral in the above equation can be written as $q_{fij}^{t-\Delta t}$ (See Equation (2.27)). This leads to the following results:

$$I_3 = q_{fij}^{t-\Delta t} + \Delta\psi_{ij}^{t} g_{2ij}^{t-\Delta t} \sigma_j^{t-\Delta t}. \tag{3.1}$$

Now consider the second integral I_4 in Equation (2.27) and write it into the following form

$$I_4 = \int_{t-\Delta t}^{t} \psi_{ij}^{t} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right) d\tau - \int_{t-\Delta t}^{t} \psi_{ij}^{\tau} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right) d\tau. \tag{2.33}$$

The first integral in Equation (2.33) turns out to be

$$\begin{aligned} &\psi_{ij}^{t} \int_{t-\Delta t}^{t} \frac{\partial}{\partial \tau}\left(g_{2ij}^{\tau}\sigma^{\tau}\right) d\tau \\ &= \psi_{ij}^{t}\left(g_{2ij}^{t}\sigma_j^{t} - g_{2ij}^{t-\Delta t}\sigma_j^{t-\Delta t}\right) \end{aligned} \tag{2.34}$$

Name the second integral in Equation (2.33) as I_5 and do the following:

$$I_5 = \int_{t-\Delta t}^{t} \psi_{ij}^{\tau} \frac{\partial\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right)}{\frac{\partial\tau}{a_{ij}}} \frac{d\tau}{a_{ij}} = \int_{t-\Delta t}^{t} \psi_{ij}^{\tau} \frac{\partial\left(g_{2ij}^{\tau}\sigma_j^{\tau}\right)}{\partial\psi_{ij}} d\psi_{ij} \,. \tag{2.35}$$

Integration by parts gives

$$I_5 = (\psi_{ij}^{t} - \frac{1}{2}\Delta\psi_{ij}^{t})(g_{2ij}^{t}\sigma_j^{t} - g_{2ij}^{t-\Delta t}\sigma_j^{t-\Delta t}). \tag{2.36}$$

With the combination of Equation (2.32), (2.34), (2.36), q_{fij}^{t} can be finally written as

$$q_{fij} = q_{fij}^{t-\Delta t} + \frac{1}{2}\Delta\psi_{ij}^{t}\left(g_{2ij}^{t}\sigma_j^{t} + g_{2ij}^{t-\Delta t}\sigma_j^{t-\Delta t}\right). \tag{2.37}$$

Equation (2.37) is now in a form to which numerical analysis can be easily applied because it requires only a knowledge of quantities at the current time step t and the previous time step t-Δt.

The constitutive Equation (2.11) now becomes

$$\varepsilon_i^{t} = \sum_{j=1}^{6}\left(g_{0ij}^{t}D_{0ij}\sigma_j^{t} + g_{1ij}^{t}g_{2ij}^{t}\sigma_j^{t}\sum_{r=1}^{N} D_{rij} - g_{1ij}^{t}\sum_{r=1}^{N} D_{rij}q_{rij}^{t} + g_{1ij}^{t}D_{fij}q_{fij}^{t}\right) + \Theta_i^{t} \,. \tag{2.38}$$

Substituting Equation (2.24) and Equation (2.37) into Equation (2.38) and grouping terms appropriately, Equation (2.38) becomes

$$\varepsilon_i^t = \sum_{j=1}^{6} \{ [g_{0ij}^t D_{0ij} + g_{1ij}^t g_{2ij}^t \sum_{r=1}^{N} D_{rij} (1 - \Gamma_{rij}^t) + 0.5 g_{1ij}^t g_{2ij}^t D_{fij}^t \Delta\psi_{ij}^t] \sigma_j^t$$
$$- g_{1ij}^t \sum_{r=1}^{N} D_{rij} (e^{-\lambda_{rij} \Delta\psi_{ij}^t} q_{rij}^{t-\Delta t} - \Gamma_{rij}^t g_{2ij}^{t-\Delta t} \sigma_j^{t-\Delta t}) + g_{1ij}^t D_{fij} (q_{fij}^{t-\Delta t} + 0.5 \Delta\psi_{ij}^t g_{2ij}^{t-\Delta t} \sigma_j^{t-\Delta t}) \} \quad (2.39)$$
$$+ \Theta_i^t .$$

Equation (2.39) can be written in matrix form as

$$\{\varepsilon\} = [S]\{\sigma\} + \{H\} + \{\Theta\} \quad (2.40)$$

where $\{\varepsilon\}$, $\{\sigma\}$ and $\{\Theta\}$ are column matrices containing the total strains, stresses, and hygrothermal components of strain respectively, $[S]$ is the instantaneous compliance matrix whose terms are given by

$$S_{ij} = g_{0ij}^t D_{0ij} + g_{1ij}^t g_{2ij}^t \sum_{r=1}^{N} D_{rij} \left(1 - \Gamma_{rij}^t\right) + 0.5 g_{1ij}^t g_{2ij}^t D_{fij} \Delta\psi_{ij}^t , \quad (2.41)$$

and $\{H\}$ is the hereditary strain matrix whose terms are given by

$$H_i = \sum_{j=1}^{6} \left[-g_{1ij}^t \sum_{r=1}^{N} D_{rij} \left(e^{-\lambda_{rij} \Delta\psi_{ij}^t} q_{rij}^{t-\Delta t} - \Gamma_{rij}^t g_{2ij}^{t-\Delta t} \sigma_j^{t-\Delta t} \right) + g_{1ij}^t D_{fij} \left(q_{fij}^{t-\Delta t} + 0.5 \Delta\psi_{ij}^t g_{2ij}^{t-\Delta t} \sigma_j^{t-\Delta t} \right) \right] . \quad (2.42)$$

Solving for $\{\sigma\}$ in Equation (2.39) gives

$$\{\sigma\} = [S]^{-1}\{\varepsilon\} - [S]^{-1}\left(\{H\} + \{\Theta\}\right) . \quad (2.43)$$

2.3 Delayed Failure of Composites

Failure studies of composite materials are still a wide open area and new failure criteria are continuously being proposed (Feng, 1989; Theocaris, 1990; Chamis

and Ginty, 1990). Of the failure criteria developed in the past, the maximum stress and maximum strain criteria are the simplest. A major disadvantage, however, is that they do not consider interaction between modes of failure (Jones, 1975). Tsai-Wu tensor theory has shown a very good agreement with experimental result for E-glass/epoxy. It's extension, however, has been limited because it requires an interaction term F_{12} which is relatively difficult to obtain, especially for compression. Compared with the others, Tsai-Hill theory is the most popular. It overcomes the disadvantage of maximum stress and maximum strain theories, and shows very good agreement between theory and experiment (though not as good as Tsai-Wu theory). Based on the above discussion, the study of delayed failure for time-dependent composites is an even more challenging area. There have been very limited criteria proposed in this area. The delayed failure criterion developed in this study is based on Brinson-Dillard's two dimensional model (Dillard and Brinson, 1983). In this model, Brinson and Dillard started from Tsai-Hill failure criterion, but instead of using constant material strengths, they assumed that the shear strength and transverse strength are functions of time. This causes the Tsai-Hill failure criterion to become time-dependent. Then, they introduced a linear cumulative damage law to predict the material's failure. Following similar procedures, a three dimensional delayed failure model is developed here.

In three dimensions, the Tsai-Hill criterion can be written as

$$\frac{\sigma_1^2}{X^2}+\frac{\sigma_2^2}{Y^2(t)}+\frac{\sigma_3^2}{Y^2(t)}-\frac{\sigma_1\sigma_2}{X^2}-\frac{\sigma_1\sigma_3}{X^2}-\left(\frac{2}{Y^2(t)}-\frac{1}{X^2}\right)\sigma_2\sigma_3 + \frac{\tau_{12}^2}{S_{12}^2(t)}+\frac{\tau_{13}^2}{S_{12}^2(t)}+\frac{\tau_{23}^2}{S_{23}^2(t)}=1 \tag{2.44}$$

where X, Y and S are material strengths. For elastic materials, they are material constants, while for fiber-reinforced viscoelastic composites, we assume that X is a constant due to the elastic behavior of the fibers in the X-direction, but Y and S are

time-dependent values defined as

$$\begin{aligned} Y(t) &= A - B\log t \\ S_{12}(t) &= K_{12}(A - B\log t) \\ S_{23}(t) &= K_{23}(A - B\log t) \end{aligned} \tag{2.45}$$

where A, B, K_{12} and K_{23} are material constants. Substituting $Y(t)$, $S_{12}(t)$ and $S_{23}(t)$ into Equation (44) gives

$$\begin{aligned} &\frac{\sigma_1^2}{X^2} + \frac{\sigma_2^2}{(A - B\log t)^2} + \frac{\sigma_3^2}{(A - B\log t)^2} - \frac{\sigma_1\sigma_2}{X^2} - \frac{\sigma_1\sigma_3}{X^2} - \left(\frac{2}{(A - B\log t)^2} - \frac{1}{X^2}\right)\sigma_2\sigma_3 \\ &+ \frac{\tau_{12}^2}{K_{12}^2(A - B\log t)^2} + \frac{\tau_{13}^2}{K_{12}^2(A - B\log t)^2} + \frac{\tau_{23}^2}{K_{23}^2(A - B\log t)^2} = 1 \end{aligned} \tag{2.46}$$

Rearranging the above equation gives

$$\begin{aligned} &\frac{\left(\sigma_2^2 + \sigma_3^2 - 2\sigma_2\sigma_3 + \frac{\tau_{12}^2}{K_{12}^2} + \frac{\tau_{13}^2}{K_{12}^2} + \frac{\tau_{23}^2}{K_{23}^2}\right)}{(A - B\log t)^2} \\ &= 1 - \frac{\sigma_1^2}{X^2} + \frac{\sigma_1\sigma_2}{X^2} + \frac{\sigma_1\sigma_3}{X^2} - \frac{\sigma_2\sigma_3}{X^2} \end{aligned} \tag{2.47}$$

Then t can be solved by

$$\log t = \frac{A - \left(\dfrac{\sigma_2^2 + \sigma_3^2 - 2\sigma_2\sigma_3 + \frac{\tau_{12}^2}{K_{12}^2} + \frac{\tau_{13}^2}{K_{12}^2} + \frac{\tau_{23}^2}{K_{23}^2}}{1 - \frac{\sigma_1^2}{X^2} + \frac{\sigma_1\sigma_2}{X^2} + \frac{\sigma_1\sigma_3}{X^2} - \frac{\sigma_2\sigma_3}{X^2}}\right)^{\frac{1}{2}}}{B} \tag{2.48}$$

Thus, the time to failure is given by

$$t_f = 10^{\frac{A-C}{B}} \tag{2.49}$$

where

$$C = \left(\frac{\sigma_2^2 + \sigma_3^2 - 2\sigma_2\sigma_3 + \frac{\tau_{12}^2}{K_{12}^2} + \frac{\tau_{13}^2}{K_{12}^2} + \frac{\tau_{23}^2}{K_{23}^2}}{1 - \frac{\sigma_1^2}{X^2} + \frac{\sigma_1\sigma_2}{X^2} + \frac{\sigma_1\sigma_3}{X^2} - \frac{\sigma_2\sigma_3}{X^2}} \right)^{\frac{1}{2}} . \tag{2.50}$$

t_f is the time to rupture, i.e., the time it will take the material to fail. Equation (2.49) shows that t_f is not a constant but a function of stresses and other parameters if more complicated situations are considered, for example temperature. To account for a time-varying stress state in each ply, a linear cumulative damage rule is used. To apply this law for each layer, t_f is calculated for each individual element from the current stresses, then the accumulated ratio ($\Delta t/t_f$) over the time period is evaluated as

$$\sum_{i=1}^{N} \frac{\Delta t_i}{t_{fi}}, \tag{2.51}$$

where N is the total number of time steps done by then. If this ratio is larger than 1, then the ply has failed according to this criterion. Once the ply is predicted to have failed, its stiffness will be reduced to a certain percentage of the original value. Then the contribution of this ply to the total strength of the structure is lowered accordingly. As time goes on, a strength degradation profile of the structure can be obtained.

CHAPTER 3

FINITE ELEMENT FORMULATION

The viscoelastic constitutive equations for an orthotropic material were developed and given in Chapter 2. As a method of approach, the finite element method is chosen due to its advantages in solving complex problems. In this chapter, general finite element procedures are introduced first in Section 3.1 and Section 3.2. Based on these, the application of the finite element method in viscoelasticity is then presented in Section 3.3.

3.1 Background of FEA Method

The finite element method is a numerical procedure for solving continuum mechanics problems with an accuracy acceptable to engineers (Cook, R.D., 1981). Compared to analytical methods and experimental methods, the finite element method is considered to be one of the most powerful, inexpensive, relatively simple and time saving ways for analyzing problems which involve very complicated geometry and loading conditions. The basic idea of finite element method is dividing a complicated structure into a number of relatively small, simple elements, then obtaining stress and deformation profiles of the complicated structure based upon the calculations of each individual element. The specific type of element used for current study is a 3-dimensional, 20-node isoparametric solid element.

In the finite element displacement method, displacements of nodal points are the primary unknowns which can be obtained first by solving a system of equations, where strains and stresses are the secondary unknowns which can be calculated from nodal displacements. The overall equilibrium equation for static analysis is:

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{R} \tag{3.1}$$

where: K = total stiffness matrix defined as

$$K = \sum_{i=1}^{N} K_e \tag{3.2}$$

u = nodal displacement vector,

N = number of elements,

K_e = element stiffness matrix,

R = total external nodal force, $F^a + F^r$

F^a = total applied load vector, $F^{nd} + F^{eq}$,

where

F^{nd} = applied nodal forces,

F^{eq} = equivalent applied nodal forces.

F^r = reaction load vector.

If u^e stands for nodal displacement vectors of an element, then the displacements at any point within this element can be described as

$$\hat{u} = Nu^e \tag{3.3}$$

where N is a set of interpolation functions called shape functions. In general, the shape functions have the properties of

$$N_i(x_i, y_i) = I \text{ (identity matrix)}$$

while

$$N_i(x_j, y_j) = N_i(x_m, y_m) = 0, \text{ etc.}$$

With displacements known at all points within the element, the strains and stresses within the element are calculated by

$$\varepsilon = Bu^e \tag{3.4}$$

and

$$\sigma = D\varepsilon \tag{3.5}$$

where $\boldsymbol{B}$ is the strain matrix composed of derivatives of the shape functions, and **D** the elasticity matrix which contains material properties.

The equations for calculating the element stiffness matrix $\boldsymbol{K}_e$ and equivalent applied nodal forces $\boldsymbol{F}^e$ can be derived by imposing an arbitrary virtual nodal displacement, and equating the external and internal work done by the various forces and stresses during that displacement (Zienkiewicz, O.C. and Taylor, R.L., 1989). The results are

$$\boldsymbol{K}_e = \int_{V_e} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} \, dvol \tag{3.6}$$

and

$$\boldsymbol{F}^{eq} = \sum_{m=1}^{N} -\int_{V_e} \boldsymbol{N}^T \boldsymbol{q} \, dvol - \int_{V_e} \boldsymbol{B}^T \boldsymbol{D} \varepsilon_o \, dvol + \int_{V_e} \boldsymbol{B}^T \sigma_o \, dvol \tag{3.7}$$

where $\boldsymbol{q}$ is the body forces per unit volume, $\boldsymbol{\varepsilon}_o$ and σ_o are the initial strain and stress respectively.

3.2 Three-dimensional 20-node Isoparametric Solid Element

Figure 3.1 shows a typical three-dimensional 20-node solid element, where nodes 1, 2, 3, 4, ...,20 are numbered in an anticlockwise order starting from any of the corner nodes. ξ, η and γ are local element coordinates defined as follows. The positive ξ axis is in the direction defined by moving along an element edge from the 1st to the 2nd and then the 3rd node. The positive η axis is in the direction of the element edge from the 3rd nodal connection number, through the 4th to the 5th number. The γ axis direction can be defined from the other two by applying the right hand rule to form a mutually perpendicular local coordinate system.

3.2.1 Interpolation Functions

The interpolation of a 20-node solid element can be written as

$$x(\xi,\eta,\gamma) = \sum_{i=1}^{20} N_i(\xi,\eta,\gamma)x_i,$$

$$y(\xi,\eta,\gamma) = \sum_{i=1}^{20} N_i(\xi,\eta,\gamma)y_i, \qquad (3.8)$$

$$z(\xi,\eta,\gamma) = \sum_{i=1}^{20} N_i(\xi,\eta,\gamma)z_i,$$

According to Equation (3.3), the displacement at any point within the element can be expressed as

$$\hat{u}(\xi,\eta,\gamma) = \sum_{i=1}^{20} N_i(\xi,\eta,\gamma)u_i \qquad (3.9)$$

where the shape functions N_i are given by

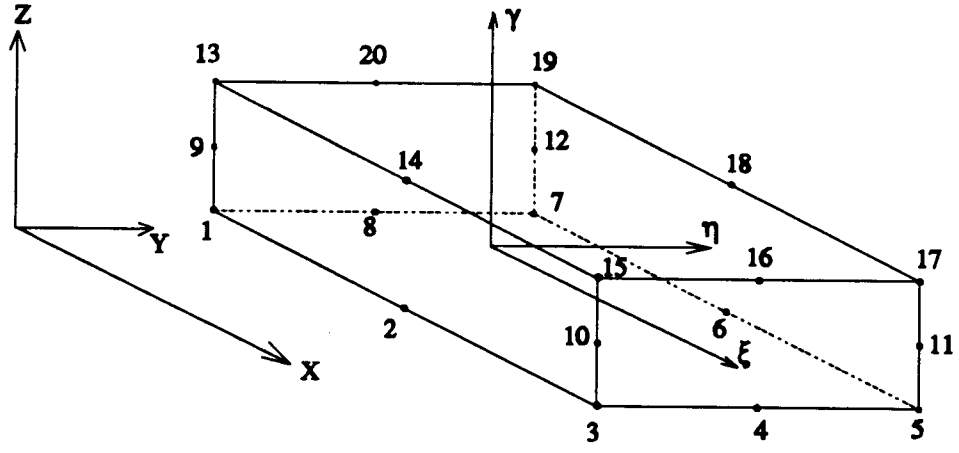


Figure 3.1 A three-dimensional 20-node isoparametric solid element.

$$
\begin{aligned}
N_1(\xi,\eta,\gamma) &= \frac{1}{8}(1-\xi)(1-\eta)(1-\gamma)(-\xi-\eta-\gamma-2)\\
N_2(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi^2)(1-\eta)(1-\gamma)\\
N_3(\xi,\eta,\gamma) &= \frac{1}{8}(1+\xi)(1-\eta)(1-\gamma)(\xi-\eta-\gamma-2)\\
N_4(\xi,\eta,\gamma) &= \frac{1}{4}(1+\xi)(1-\eta^2)(1-\gamma)\\
N_5(\xi,\eta,\gamma) &= \frac{1}{8}(1+\xi)(1+\eta)(1-\gamma)(\xi+\eta-\gamma-2)\\
N_6(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi^2)(1+\eta)(1-\gamma)\\
N_7(\xi,\eta,\gamma) &= \frac{1}{8}(1-\xi)(1+\eta)(1-\gamma)(-\xi+\eta-\gamma-2)\\
N_8(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi)(1-\eta^2)(1-\gamma)\\
N_9(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi)(1-\eta)(1-\gamma^2)\\
N_{10}(\xi,\eta,\gamma) &= \frac{1}{4}(1+\xi)(1-\eta)(1-\gamma^2)\\
N_{11}(\xi,\eta,\gamma) &= \frac{1}{4}(1+\xi)(1+\eta)(1-\gamma^2)\\
N_{12}(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi)(1+\eta)(1-\gamma^2)\\
N_{13}(\xi,\eta,\gamma) &= \frac{1}{8}(1-\xi)(1-\eta)(1+\gamma)(-\xi-\eta+\gamma-2)\\
N_{14}(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi^2)(1-\eta)(1+\gamma)\\
N_{15}(\xi,\eta,\gamma) &= \frac{1}{8}(1+\xi)(1-\eta)(1+\gamma)(\xi-\eta+\gamma-2)\\
N_{16}(\xi,\eta,\gamma) &= \frac{1}{4}(1+\xi)(1-\eta^2)(1+\gamma)\\
N_{17}(\xi,\eta,\gamma) &= \frac{1}{8}(1+\xi)(1+\eta)(1+\gamma)(\xi+\eta+\gamma-2)\\
N_{18}(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi^2)(1+\eta)(1+\gamma)\\
N_{19}(\xi,\eta,\gamma) &= \frac{1}{8}(1-\xi)(1+\eta)(1+\gamma)(-\xi+\eta+\gamma-2)\\
N_{20}(\xi,\eta,\gamma) &= \frac{1}{4}(1-\xi)(1-\eta^2)(1+\gamma).
\end{aligned}
\tag{3.10}
$$

3.2.2 Strains

With displacements known at all points within the element, the strains can be determined by Equation (3.4), with $\boldsymbol{B}$ matrix written as

$$\boldsymbol{B} = [\boldsymbol{B}_1, \boldsymbol{B}_2, \ldots\ldots, \boldsymbol{B}_i], \qquad i = 1, \ldots, 20 \tag{3.11}$$

where

$$\boldsymbol{B}_i = \begin{bmatrix} \dfrac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \dfrac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \dfrac{\partial N_i}{\partial z} \\ 0 & \dfrac{\partial N_i}{\partial z} & \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial z} & 0 & \dfrac{\partial N_i}{\partial y} \\ \dfrac{\partial N_i}{\partial y} & \dfrac{\partial N_i}{\partial x} & 0 \end{bmatrix} \tag{3.12}$$

Equation (3.4) now becomes

$$\begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{zx} \\ \gamma_{xy} \end{Bmatrix} = \sum_{i=1}^{20} \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \frac{\partial N_i}{\partial z} \\ 0 & \frac{\partial N_i}{\partial z} & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} & 0 & \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} & 0 \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}. \tag{3.13}$$

Note that $\boldsymbol{B}$ is composed of the Cartesian derivatives of N_i while N_i are expressed in terms of element coordinates ξ, η and γ rather than the Cartesian coordinates x, y and z. This can be readily solved by applying the Chain rule of differentiation. Consider the derivatives of N_i with respect to the local element coordinates

$$\begin{aligned} \frac{\partial N_i}{\partial \xi} &= \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \xi} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial \xi}, \\ \frac{\partial N_i}{\partial \eta} &= \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \eta} + \frac{\partial N_i}{\partial z}\frac{\partial z}{\partial \eta}, \\ \frac{\partial N_i}{\partial \gamma} &= \frac{\partial N_i}{\partial x}\frac{\partial x}{\partial \gamma} + \frac{\partial N_i}{\partial y}\frac{\partial y}{\partial \gamma} + \frac{\partial N_z}{\partial z}\frac{\partial z}{\partial \gamma}. \end{aligned} \tag{3.14}$$

In matrix form, Equation (3.14) becomes

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \gamma} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \gamma} & \frac{\partial y}{\partial \gamma} & \frac{\partial z}{\partial \gamma} \end{bmatrix} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} = \boldsymbol{J} \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} \tag{3.15}$$

where $\boldsymbol{J}$ is known as the *Jacobian matrix* which can be found explicitly in terms of the local coordinates. Substitution of Equation (3.8) into Equation (3.15) gives

$$\boldsymbol{J} = \sum_{i=1}^{20} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} x_i, & \frac{\partial N_i}{\partial \xi} y_i, & \frac{\partial N_i}{\partial \xi} z_i \\ \frac{\partial N_i}{\partial \eta} x_i, & \frac{\partial N_i}{\partial \eta} y_i, & \frac{\partial N_i}{\partial \eta} z_i \\ \frac{\partial N_i}{\partial \gamma} x_i, & \frac{\partial N_i}{\partial \gamma} y_i, & \frac{\partial N_i}{\partial \gamma} z_i \end{bmatrix}. \tag{3.16}$$

The Cartesian derivatives of N_i can be found now from Equation (3.15) as

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{Bmatrix} = \boldsymbol{J}^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \gamma} \end{Bmatrix}. \tag{3.17}$$

It also can be proved by vector algebra that the element volume *dvol* can be evaluated as

$$\begin{aligned} dvol &= dx\,dy\,dz \\ &= \det \boldsymbol{J}\, d\xi\, d\eta\, d\gamma \end{aligned} \tag{3.18}$$

where *det J* is the determinant of Jacobian matrix.

3.2.3 Stresses

The stress-strain relation can be expressed as

$$\sigma = D\varepsilon \tag{3.19}$$

where D is the stiffness matrix, and the strain-stress relation

$$\varepsilon = C\sigma \tag{3.20}$$

or

$$\begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \gamma_{23} \\ \gamma_{13} \\ \gamma_{12} \end{Bmatrix} = [C] \begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \tau_{23} \\ \tau_{13} \\ \tau_{12} \end{Bmatrix} \tag{3.21}$$

where C is the compliance matrix defined by the inverse of the stress-strain relation. The subscripts *1*, *2* and *3* denote the three principal material directions. For a general orthotropic material, the compliance matrix components in terms of the engineering constants are

$$[C_{ij}] = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{21}}{E_2} & -\frac{\nu_{31}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & \frac{\nu_{32}}{E_3} & 0 & 0 & 0 \\ -\frac{\nu_{13}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_3} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{G_{23}} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{31}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix} \tag{3.22}$$

where

E_1, E_2, E_3 = Young's moduli in *1*, *2* and *3* directions, respectively.

ν_{ij} = Poisson's ration for transverse strain in the j-direction when stressed in the i-direction.

G_{23}, G_{31}, G_{12} = shear moduli in the *2-3*, *3-1*, and *1-2* planes, respectively.

Due to the symmetric property of the compliance matrix and assuming that the fiber-reinforced composite materials are transversely isotropic, where

$E_2 = E_3$, $\nu_{13} = \nu_{12}$, and $G_{23} = E_2/2(1 + \nu_{23})$,

then $[C_{ij}]$ can be rewritten as

$$[C_{ij}] = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{12}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{1}{E_2} & -\frac{\nu_{23}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & -\frac{\nu_{23}}{E_2} & \frac{1}{E_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 2\left(\frac{1}{E_2}+\frac{\nu_{23}}{E_2}\right) & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{G_{12}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{G_{12}} \end{bmatrix}. \quad (3.23)$$

The stress matrix ***D*** can be obtained by

$$[D] = [C]^{-1}. \quad (3.24)$$

3.2.4 Transformation Matrix

The matrix ***D*** in the previous section has been defined in the principal material directions. However, the principal material directions are not always coincident with the global coordinate system in which the directions of displacements, applied external loads, and the rotations of the principal material axies are defined. Consider the case where the principal material direction is rotated an angle θ about z-axis, relative to x-axis in the global Cartesian coordinate system. The stress-strain relationship in the global coordinates becomes

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{Bmatrix} = [T]^{-1}[D][T_\varepsilon] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{Bmatrix} \tag{3.25}$$

where $[T]$ is called transformation matrix and it has the form of

$$[T] = \begin{bmatrix} \cos^2\theta & \sin^2\theta & 0 & 0 & 0 & 2\sin\theta\cos\theta \\ \sin^2\theta & \cos^2\theta & 0 & 0 & 0 & -2\sin\theta\cos\theta \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 & \sin\theta & \cos\theta & 0 \\ -\sin\theta\cos\theta & \sin\theta\cos\theta & 0 & 0 & 0 & \cos^2\theta - \sin^2\theta \end{bmatrix}, \tag{3.26}$$

$[T_\varepsilon]$ is strain transformation matrix, which is related with $[T]$ as (Jones, 1975)

$$[T_\varepsilon] = [T]^{-T}$$

where the superscript T denotes the matrix transpose. If $[\bar{D}] = [T]^{-1}[D][T_\varepsilon]$, the stress-strain relations in xyz coordinates becomes

$$\begin{Bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{yz} \\ \tau_{xz} \\ \tau_{xy} \end{Bmatrix} = [\bar{D}] \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{yz} \\ \gamma_{xz} \\ \gamma_{xy} \end{Bmatrix}. \tag{3.27}$$

3.2.5 Numerical Integration of Element Stiffness Matrix

With the matrix $\bar{D}$ defined by $[\bar{D}] = [T]^{-1}[D][T_\varepsilon]$ in the global coordinate system, the element stiffness matrix in Equation (3.6) becomes

$$[K_e] = \int_{v_e} [B]^T [T]^{-1} [D][T_\varepsilon][B] dvol. \tag{3.28}$$

where

$$dvol = dxdydz.$$

Switching to element coordinates, Equation (3.28) becomes

$$[K_e] = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [B]^T [T]^{-1} [D][T_\varepsilon][B] \det J d\xi d\eta d\gamma. \tag{3.29}$$

Note here that $[B]$ and *det J* in the above integral involve functions of ξ, η and γ, and so the integration has to be performed numerically. The Gauss quadrature method has proved most useful in finite element work, and it shows that the integral

$$I = \int_{-1}^{1} \phi(\xi) d\xi \tag{3.30}$$

can be approximated by sampling $\phi(\xi)$ at the midpoint of the interval and multiplying by the length of the interval, thus $I \approx 2\phi_1$. This result is exact if the function ϕ happens to be a straight line of any slope.

Generalization of Equation (3.30) leads to the formula

$$I = \int_{-1}^{1} \phi(\xi) d\xi \approx W_1\phi_1 + W_2\phi_2 + \ldots\ldots + W_n\phi_n. \tag{3.31}$$

Thus, I has been approximated by evaluating $\phi = \phi(\xi)$ at each of several locations ξ_i, multiplying the resulting ϕ_i by an appropriate weight W_i, and adding. Gauss's method locates the sampling points so that for a given number of points, greatest accuracy is achieved. In three dimensions, Gauss's method can be written as

$$I = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}\phi(\xi,\eta,\gamma)d\xi d\eta d\gamma \approx \sum_i\sum_j\sum_k W_i W_j W_k \phi(\xi_i,\eta_j,\gamma_k). \tag{3.32}$$

Consider now that $\phi(\xi,\eta,\gamma)$ is replaced by $[B]^T[T]^{-1}[D][T_\varepsilon][B]$ and I by K_e,

$$[K_e] = \sum_i\sum_j\sum_k [B]^T[T]^{-1}[D][T_\varepsilon][B]\det J W_i W_j W_k. \tag{3.33}$$

3.3 The Application of FEA Method in Viscoelasticity

3.3.1 Constitutive Equation

Recall from Chapter 2, the stress-strain relationship for transversely isotropic viscoelastic material is

$$\{\sigma\} = [S]^{-1}\{\varepsilon\} - [S]^{-1}\left(\{H\} + \{\Theta\}\right), \tag{3.34}$$

where the compliance matrix $[S]$, can be written as

$$[S] = [S_A] + [S_B] + [S_C]. \tag{3.35}$$

In terms of engineering constants and viscoelastic property quantities, the matrices $[S_A]$, $[S_B]$ and $[S_C]$ are

$$[S_A] = \begin{bmatrix} \frac{1}{E_1} & -\frac{\nu_{12}}{E_1} & -\frac{\nu_{12}}{E_1} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & \frac{g_{0t}}{E_2} & -\frac{\nu_{23} g_{0t}}{E_2} & 0 & 0 & 0 \\ -\frac{\nu_{12}}{E_1} & -\frac{\nu_{23} g_{0t}}{E_2} & \frac{g_{0t}}{E_2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{2 g_{0t}(1+\nu_{23})}{E_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{g_{0s}}{G_{12}} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{g_{0s}}{G_{12}} \end{bmatrix}, \tag{3.36}$$

$$[S_B] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & S_{B22} & S_{B23} & 0 & 0 & 0 \\ 0 & S_{B32} & S_{B33} & 0 & 0 & 0 \\ 0 & 0 & 0 & S_{B44} & 0 & 0 \\ 0 & 0 & 0 & 0 & S_{B55} & 0 \\ 0 & 0 & 0 & 0 & 0 & S_{B66} \end{bmatrix} \tag{3.37}$$

with

$$
\begin{aligned}
S_{B22} &= S_{B33} = g_{1T}^{t} g_{2T}^{t} \sum_{r=1}^{N} D_{rT}\left(1-\Gamma_{rT}\right) \\
S_{B23} &= S_{B32} = -\nu_{23} g_{1T}^{t} g_{2T}^{t} \sum_{r=1}^{N} D_{rT}\left(1-\Gamma_{rT}\right) \\
S_{B44} &= 2\left(1+\nu_{23}\right) g_{1T}^{t} g_{2T}^{t} \sum_{r=1}^{N} D_{rT}\left(1-\Gamma_{rT}\right) \\
S_{B55} &= S_{B66} = g_{1s}^{t} g_{2s}^{t} \sum_{r=1}^{N} D_{rs}\left(1-\Gamma_{rs}\right)
\end{aligned}
\tag{3.38}
$$

and

$$
\left[S_C\right] = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & S_{C22} & S_{C23} & 0 & 0 & 0 \\
0 & S_{C32} & S_{C33} & 0 & 0 & 0 \\
0 & 0 & 0 & S_{C44} & 0 & 0 \\
0 & 0 & 0 & 0 & S_{C55} & 0 \\
0 & 0 & 0 & 0 & 0 & S_{C66}
\end{bmatrix}
\tag{3.39}
$$

with

$$
\begin{aligned}
S_{C22} &= S_{C33} = \frac{1}{2} g_{1T}^{t} g_{2T}^{t} D_{fT} \Delta\psi_T^{t} \,, \\
S_{C23} &= S_{C32} = -\frac{1}{2} \nu_{23} g_{1T}^{t} g_{2T}^{t} D_{fT} \Delta\psi_T^{t} \,, \\
S_{C44} &= \left(1+\nu_{23}\right) g_{1T}^{t} g_{2T}^{t} D_{fT} \Delta\psi_T^{t} \,, \\
S_{C55} &= S_{C66} = \frac{1}{2} g_{1s}^{t} g_{2s}^{t} D_{fs} \Delta\psi_s^{t} \,.
\end{aligned}
\tag{3.40}
$$

The subscript T denotes the transverse viscoelastic properties and subscript s denotes shear viscoelastic properties. In the above equations, the nonlinear terms in the 1-direction, which is the fiber direction, are zero. This means that compared to the matrix properties, fibers behave as linear elastic materials, which dominate the

properties in 1-direction.

The hereditary strain $\{H\}$ can be written as

$$\{H\} = \left([H_A] + [H_B]\right)\{\sigma\}^{t-\Delta t} + \{P\} \ . \tag{3.41}$$

In terms of nonlinear viscoelastic quantities, the two (6 x 6) matrices $[H_A]$, $[H_B]$ are

$$[H_A] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & H_{A22} & H_{A23} & 0 & 0 & 0 \\ 0 & H_{A32} & H_{A33} & 0 & 0 & 0 \\ 0 & 0 & 0 & H_{A44} & 0 & 0 \\ 0 & 0 & 0 & 0 & H_{A55} & 0 \\ 0 & 0 & 0 & 0 & 0 & H_{A66} \end{bmatrix} \tag{3.42}$$

with the nonzero terms defined as

$$\begin{aligned} H_{A22} &= H_{A33} = g_{1T}^{t} g_{2T}^{t-\Delta t} \sum_{r=1}^{N} D_{rT} \Gamma_{rT} \ , \\ H_{A23} &= H_{A32} = -\nu_{23} g_{1T}^{t} g_{2T}^{t-\Delta t} \sum_{r=1}^{N} D_{rT} \Gamma_{rT} \ , \\ H_{A44} &= 2\left(1+\nu_{23}\right) g_{1T}^{t} g_{2T}^{t-\Delta t} \sum_{r=1}^{N} D_{rT} \Gamma_{rT} \ , \\ H_{A55} &= H_{A66} = g_{1s}^{t} g_{2s}^{t-\Delta t} \sum_{r=1}^{N} D_{rs} \Gamma_{rs} \ , \end{aligned} \tag{3.43}$$

and

$$[H_B] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & H_{B22} & H_{B23} & 0 & 0 & 0 \\ 0 & H_{B32} & H_{B33} & 0 & 0 & 0 \\ 0 & 0 & 0 & H_{B44} & 0 & 0 \\ 0 & 0 & 0 & 0 & H_{C55} & 0 \\ 0 & 0 & 0 & 0 & 0 & H_{B66} \end{bmatrix} \tag{3.44}$$

with the nonzero terms defined

$$\begin{aligned} H_{B22} &= H_{B33} = \frac{1}{2} g_{1T}^{t} g_{2T}^{t-\Delta t} D_{fT} \Delta\psi_T^t \ , \\ H_{B23} &= H_{B32} = -\frac{1}{2} \nu_{23} g_{1T}^{t} g_{2T}^{t-\Delta t} D_{fT} \Delta\psi_T^t \ , \\ H_{B44} &= (1+\nu_{23}) g_{1T}^{t} g_{2T}^{t-\Delta t} D_{fs} \Delta\psi_T^t \ , \\ H_{B55} &= H_{B66} = \frac{1}{2} g_{1s}^{t} g_{2s}^{t-\Delta t} D_{fs} \Delta\psi_s^t \ . \end{aligned} \tag{3.45}$$

The (6 x 1) column matrix $\{P\}$ is

$$\{P\} = \begin{Bmatrix} 0 \\ -g_{1T}^{t} \sum_{r=1}^{N} D_{rT} e^{-\lambda_{rT}\Delta\psi_T^t} \left(q_{2r}^{t-\Delta t} - \nu_{23} q_{3r}^{t-\Delta t} \right) + g_{1T}^{t} D_{fT} \left(q_{f2_{t-\Delta t}} - \nu_{23} q_{f3}^{t-\Delta t} \right) \\ -g_{1T} \sum_{r=1}^{N} D_{rT} e^{-\lambda_{rT}\Delta\psi_T^t} \left(q_{3r}^{t-\Delta t} - \nu_{23} g_{2r}^{t-\Delta t} \right) + g_{1T}^{t} D_{fT} \left(q_{f3}^{t-\Delta t} - \nu_{23} q_{f2}^{t-\Delta t} \right) \\ -2(1+\nu_{23}) g_{1t} \sum_{r=1}^{N} D_{rT} e^{-\lambda_{rT}\Delta\psi_T^t} q_{4r}^{t-\Delta t} + 2(1+\nu_{23}) g_{1T}^{t} D_{fT} q_{f4}^{t-\Delta t} \\ -g_{1s} \sum_{r=1}^{N} D_{rs} e^{-\lambda_{rs}\Delta\psi_s^t} q_{5r}^{t-\Delta t} + g_{1s}^{t} D_{fs} q_{f5}^{t-\Delta t} \\ -g_{1s} \sum_{r=1}^{N} D_{rs} e^{-\lambda_{rs}\Delta\psi_T^t} q_{6r}^{t-\Delta t} + g_{1s}^{t} D_{fs} q_{f6}^{t-\Delta t} \end{Bmatrix} \tag{3.46}$$

where

$$
\begin{aligned}
q_{2r} &= e^{-\lambda_{rT}\Delta\psi_T^t} q_{2r}^{t-\Delta t} + \Gamma_{rT}\left(g_{2T}^{t}\sigma_2^t - g_{2T}^{t-\Delta t}\sigma_2^{t-\Delta t}\right), \\
q_{3r} &= e^{-\lambda_{rT}\Delta\psi_T^t} q_{3r}^{t-\Delta t} + \Gamma_{rT}\left(g_{2T}^{t}\sigma_3^t - g_{2T}^{t-\Delta t}\sigma_3^{t-\Delta t}\right), \\
q_{4r} &= e^{-\lambda_{rT}\Delta\psi_T^t} q_{4r}^{t-\Delta t} + \Gamma_{rT}\left(g_{2T}^{t}\sigma_4^t - g_{2T}^{t-\Delta t}\sigma_4^{t-\Delta t}\right), \\
q_{5r} &= e^{-\lambda_{rs}\Delta\psi_s^t} q_{5r}^{t-\Delta t} + \Gamma_{rs}\left(g_{2s}^{t}\sigma_5^t - g_{2s}^{t-\Delta t}\sigma_5^{t-\Delta t}\right), \\
q_{6r} &= e^{-\lambda_{rs}\Delta\psi_s^t} q_{6r}^{t-\Delta t} + \Gamma_{rs}\left(g_{2s}^{t}\sigma_6^t - g_{2s}^{t-\Delta t}\sigma_6^{t-\Delta t}\right),
\end{aligned}
\tag{3.47}
$$

and

$$
\begin{aligned}
q_{f2} &= q_{f2}^{t-\Delta t} + \frac{1}{2}\Delta\psi_T^t\left(g_{2T}^{t}\sigma_2^t + g_{2T}^{t-\Delta t}\sigma_2^{t-\Delta t}\right), \\
q_{f3} &= q_{f3}^{t-\Delta t} + \frac{1}{2}\Delta\psi_T^t\left(g_{2T}^{t}\sigma_3^t + g_{2T}^{t-\Delta t}\sigma_3^{t-\Delta t}\right), \\
q_{f4} &= q_{f4}^{t-\Delta t} + \frac{1}{2}\Delta\psi_T^t\left(g_{2T}^{t}\sigma_4^t + q_{2T}^{t-\Delta t}\sigma_4^{t-\Delta t}\right), \\
q_{f5} &= q_{f5}^{t-\Delta t} + \frac{1}{2}\Delta\psi_s^t\left(g_{2s}^{t}\sigma_5^t + q_{2s}^{t-\Delta t}\sigma_5^{t-\Delta t}\right), \\
q_{f6} &= q_{f6}^{t-\Delta t} + \frac{1}{2}\Delta\psi_s^t\left(g_{2s}^{t}\sigma_6^t + q_{2s}^{t-\Delta t}\sigma_6^{t-\Delta t}\right).
\end{aligned}
\tag{3.48}
$$

3.3.2 Element Stiffness Matrix

The element stiffness matrix can then be written as (See Section 3.2.5, Equation (3.28))

$$
[K_e] = \int_{V_e} [B]^T [T]^{-1} [S]^{-1} [T_\varepsilon][B]\, dvol \tag{3.49}
$$

where $[B]$, $[T]$ and $[T_\varepsilon]$ are matrices defined in Section 3.2.4, and [S] is the viscoelastic compliance matrix defined in the previous section.

3.3.3 The Generalized Nodal Loads

The total generalized nodal loads for each element are

$$\{R_e\} = \{F_e^{nd}\} + \{F_e^{cr}\} + \{F_e^{th}\} \tag{3.50}$$

where F_e^{nd} is the applied nodal load vector, F_e^{cr} is the element hereditary strain load vector, and F_e^{th} is the element thermal load vector. The finite element formulation to calculate the generalized load for each element is

$$\begin{aligned}\{R_e\} &= \{F_e^{nd}\} + \int_v [B]^T[T]^{-1}[S]^{-1}\left(\{H\}+\{\Theta\}\right)dvol \\ &= \{F_e^{nd}\} + \sum_j \sum_k \sum_l [B]^T[T]^{-1}[S]^{-1}\left(\{H\} + \Delta T\{\alpha\}\right)|J|W_j W_k W_l\end{aligned} \tag{3.51}$$

3.3.4 System Equations and Numerical Solution Procedure

The structure stiffness matrix and load vectors are formed by addition of the element stiffness matrix and vectors respectively. For example

$$K = \sum_{ielem=1}^{N} K_e \tag{3.52}$$

where N is the number of total elements in the structure. The system equation to solve now has the form of

$$[K]\{U\} = \{R\} \tag{3.53}$$

where {U} is the list of nodal displacements and {R} is the total assembled nodal load vector of the structure. Each of the forces in {R} must contain the same number of components as the corresponding nodal displacement and be ordered in the

appropriate, corresponding directions. In determining a solution to Equation (3.53) Newton's method is used. The major steps of solution procedure are listed below.

1. Read external load vector from input file, initialize creep strain vector and nodal displacement vectors. Initialize stresses and strains.

2. Update old variables by replacing them with the converged solutions in previous time step. For first time step, use initialized values from step 1.

----- For each time step -----

3. Calculate nonlinear creep parameters g_o^t, g_1^t and g_2^t from previous stresses. Calculate unbalanced load vector due to nonlinear creep strain.

4. Form stiffness matrix, K.

5. Solve the system of equations for displacements.

6. Solve for new displacement increments:

$$\{U\}_i^{t+\Delta t} = \{U\}_{i-1}^{t+\Delta t} + \{\Delta U\}_i^{t+\Delta t} \tag{3.54}$$

where

$$[K]_{i-1}^{t+\Delta t}\{\Delta U\}_i^{t+\Delta t} = \{R\}_i^{t+\Delta t} - [K]_{i-1}^{t+\Delta t}\{U\}_{i-1}^{t+\Delta t} \tag{3.55}$$

7. Calculate stresses, strains from $\{U\}_i^t$.

8. Check iteration convergence to see if

$$\frac{\|\{\Delta U\}_i^{t+\Delta t}\|}{\|\{U\}_i^{t+\Delta t}\|} < 0.00001 \ . \tag{3.56}$$

If convergence is achieved, $\{u\} = \{u\}_i$, print out stresses, strains, displacements and creep strains for each element. Increase time by $t = t + \Delta t$ for next time step, and go to 2.

If no convergence is achieved, $i = i + 1$ and go to 3.

CHAPTER 4

DEVELOPMENT OF THREE-DIMENSIONAL FINITE ELEMENT PROGRAM

Based on the theoretical discussions presented in the earlier sections, a finite element computer code was developed. This section gives overall descriptions of program development and organization, which will be necessary for the reader to understand the program. Instructions for using the program is given in Appendix A.

4.1 General Description of Program

LAMCREP (LAMinate-CREeP) is a 3-D finite element program developed for analyzing nonlinear viscoelastic creep phenomena for laminate composites. A 20-node isoparametric solid element employed in the program can be used both in general purpose isotropic (or orthotropic) linear elastic analysis and in orthotropic nonlinear creep analysis. A failure model to predict long term delayed failure of laminates is available for nonlinear analysis in LAMCREP.

4.2 Program Development

LAMCREP is based upon a 2-D finite element program PLANE (in FORTRAN 77) developed by E. Hinton and D. R. J. Owen (Hinton, E. and Owen, D.R.J., 1989) in their plane stress/strain analysis. However, the original program has been dramatically modified and changed for solving the problem of nonlinear viscoelastic creep in laminated composites with orthotropic material properties. Compared to the original program, the major differences include:

1) Conversion to a three-dimensional analysis from two-dimensional plane stress/strain situations. The corresponding changes were made in shape functions (Tong, P. and Rossettos, J.N., 1977), constitutive equations and all quantities relating to coordinate dimensions.

12) Capability to handle nonlinear problems. The corresponding changes made in the program are: a) adding a time iteration loop and a convergence iteration loop in the main program; b) performing error estimation in the equation solving subroutine.

3) Development of new constitutive equations to model viscoelastic creep phenomena.

4) Capability to handle orthotropic properties.

5) Development of a failure criterion to predict long term failure of laminated composites.

6) The subroutine GAUSS in the original program was substituted by setting a gauss point table in element stiffness subroutine STIFE.

7) The subroutine LOADPS, which reads external nodal forces in the original program, was merged into the subroutine INPUT in LAMCREP.

4.3 Program Organization

4.3.1 Subroutines and Algorithms

LAMCREP can be divided in to four major phases:

1) Data input:
   - a. control information
   - b. geometry data
   - c. constraint data
   - d. material property data
   - e. load data
   - f. output table

2) Calculation of the element stiffness and stress matrices.

3) Assemblage of stiffness matrix and equation solver.

4) Strain/stress calculations and failure predictions.

Table 4.1 on this page lists all the subroutines in LAMCREP.

Table 4.1 LIST OF SUBROUTINES

(Listed in an order that each subroutine appeared in the program)

LAMCREP:	Main program.
INPUT:	The subroutine which reads the input data file, and saves all of the input information into a file named data.out.
CHECK1:	The error diagnostic subroutine.
CHECK2:	The error diagnostic subroutine.
ECHO:	The data echo subroutine.
INITIA:	The subroutine which initializes variables before time iteration starts.
STIFE:	The element stiffness subroutine which 1) forms element stiffness [K_e] and saves them into file 1, 2) saves matrix product [D][B] into file 3, and [B] into file 7 for future use in the stress and creep strain calculations, 3) calculates the force vector increments due to creep strains, and adds them on to the external nodal forces.
DMATX:	The subroutine which evaluates the linear and nonlinear stress-strain matrix [D] for each element.
SHAPE:	Shape function subroutine.
JACOB:	The subroutine which calculates: 1) the coordinates of the Gauss points, 2) the Jacobian matrix, 3) the inverse of the Jacobian matrix, 4) the Cartesian shape function derivatives.
BMATX:	The subroutine which evaluates the strain matrix [B].
DBE:	The subroutine which multiplies matrix [D] by matrix [B].
FRONT:	The equation solution subroutine which solves the system equations for displacements and reaction forces, and saves then into the file named disp.out.
STRESS:	The subroutine which 1) calculates strains and saves them into the file named strain.out. 2) calculates stresses and saves them into the file named stress.out. 3) performs failure analysis.

Several flowcharts were developed from different aspects to provide better views of the algorithms of the program. The bold boxes in the flowcharts represent related subroutines.

Figure 4.1 is a flowchart of the main program which controls the overall program.

Figure 4.2 shows the procedure to form the element stiffness matrix. In addition, because the analysis is nonlinear, there will be increments in load vectors due to nonlinear creep strains in each time step. The calculation of creep load vectors are also done in the subroutine STIFE.

The algorithm of failure analysis is shown in Figure 4.3. The flowchart in Figure 4.3 was developed in the main program level. Since failure analysis is performed on the converged values, the convergence iteration loop was omitted here. The element failure control parameters are stored in the array kfact(nelem) and are initialized to zero before the elements failed. Once an individual element is predicted to have failed, its kfact will be set equal to 1. Then in the next time step, the stiffness of this element will be reduced by a factor less than 1. The stiffness reduction is done in the subroutine DMATX. In Figure 4.3, the box for subroutine DMATX shows the procedure for stiffness reductions. d11, d22 and d12 are the stiffness reduction factors, and they are greater than zero and less than 1. For example, when kfact = 0, $E1 = E1 \times d11^{kfact} = E1$; after the element failed, kfact =1, $E1 = E1 \times d11^{kfact} = E1 \times d11$, i.e., the Young's modulus will be reduced by a factor of d11 for a failed element.

The frontal equation solution technique was adopted in program LAMCREP because it has the advantage of minimizing core storage which is very important for solving three-dimensional problems. The details about the frontal technique are given by Hinton (Hinton, E. and Owen, D.R.J.,1989). Figure 4.4 is a flowchart of the subroutine FRONT.

The maximum problem size that LAMCREP can handle is limited by the maximum frontwidth, MFRONT, which depends on the structural geometry as well as the order of element numbering. MFRONT is currently set equal to 600. If MFRONT needs to be increased for solving a larger problem, the dimensions of the

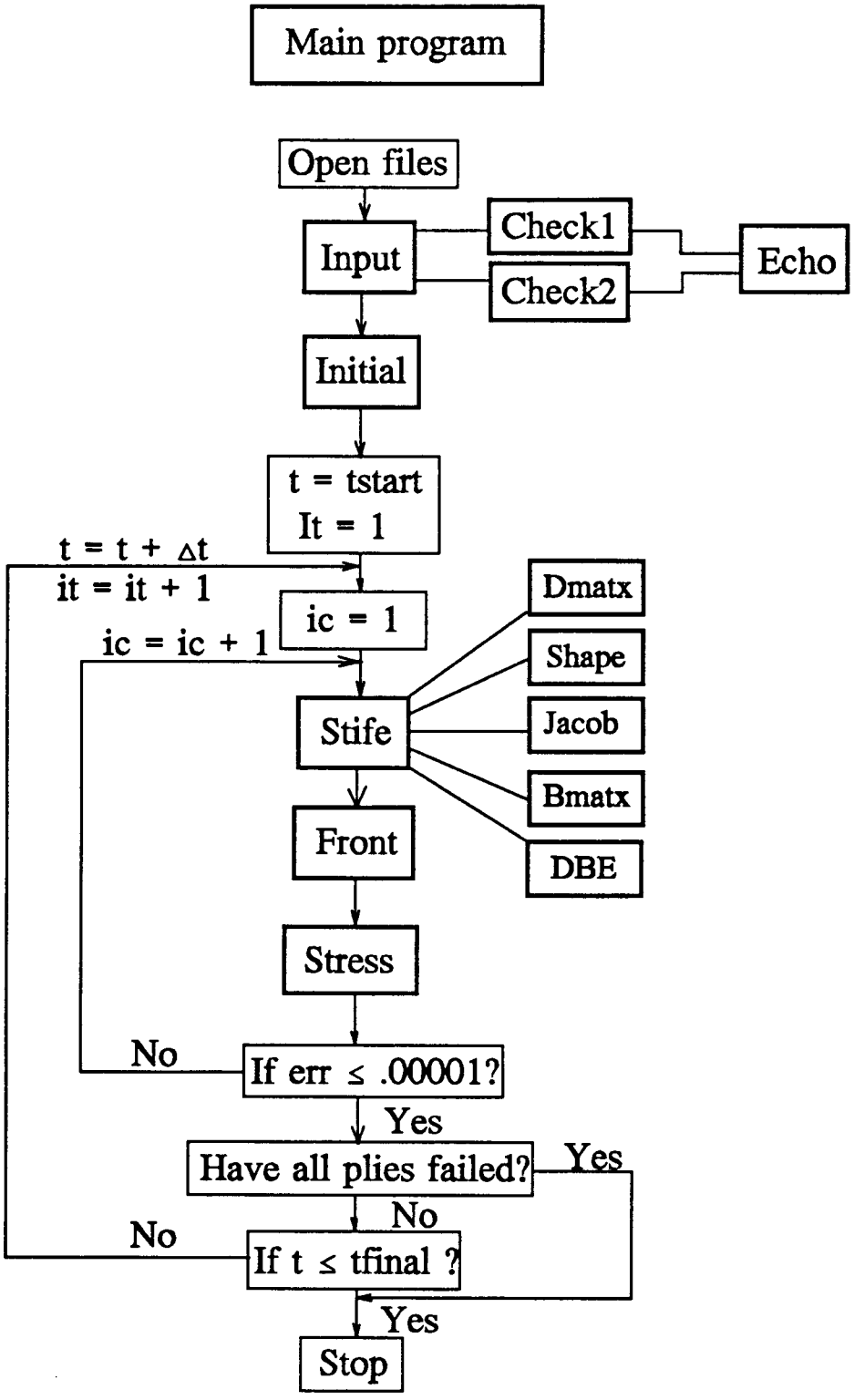


Figure 4.1 Flowchart of the main program.

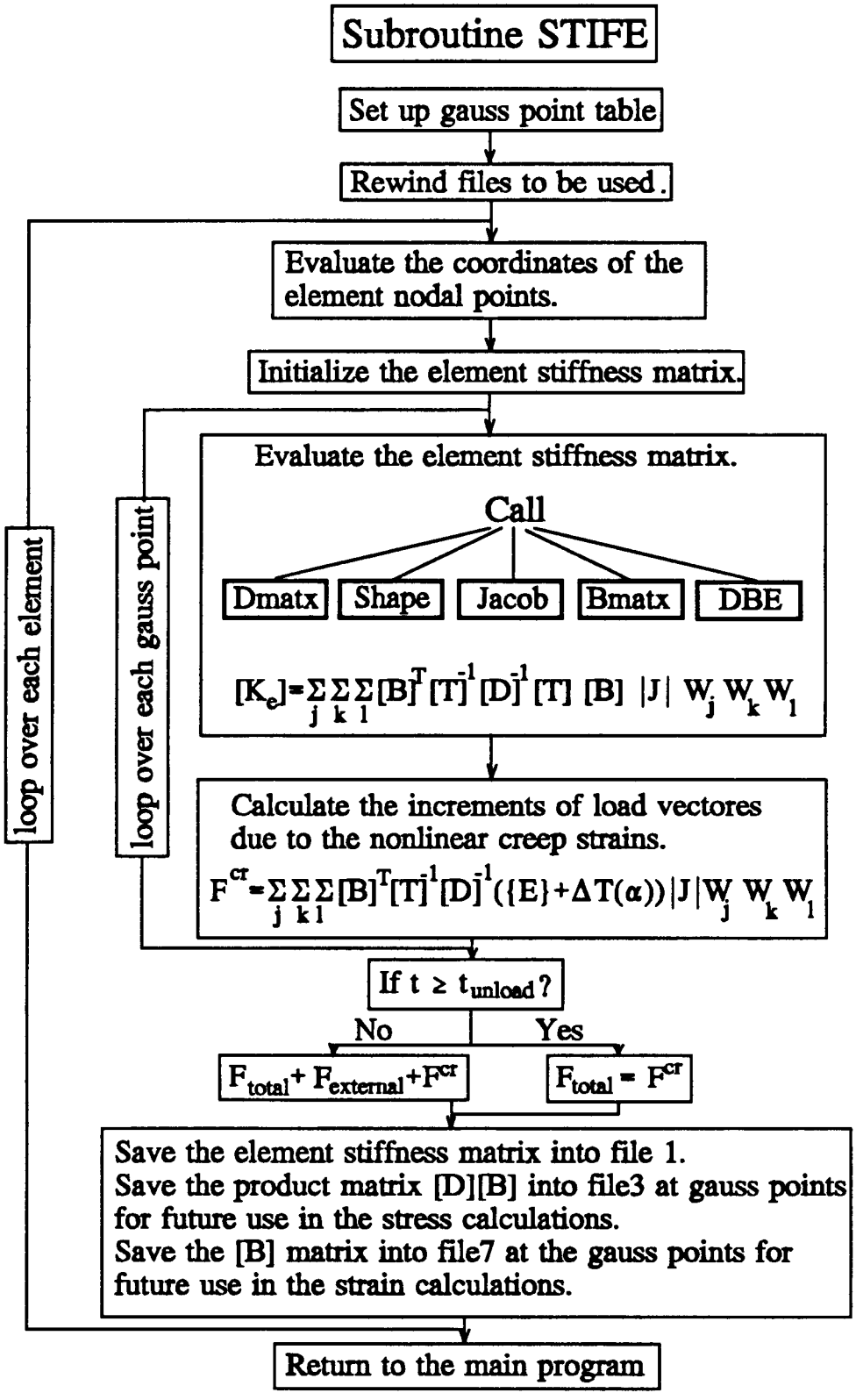


Figure 4.2 Flowchart of element stiffness subroutine STIFE.

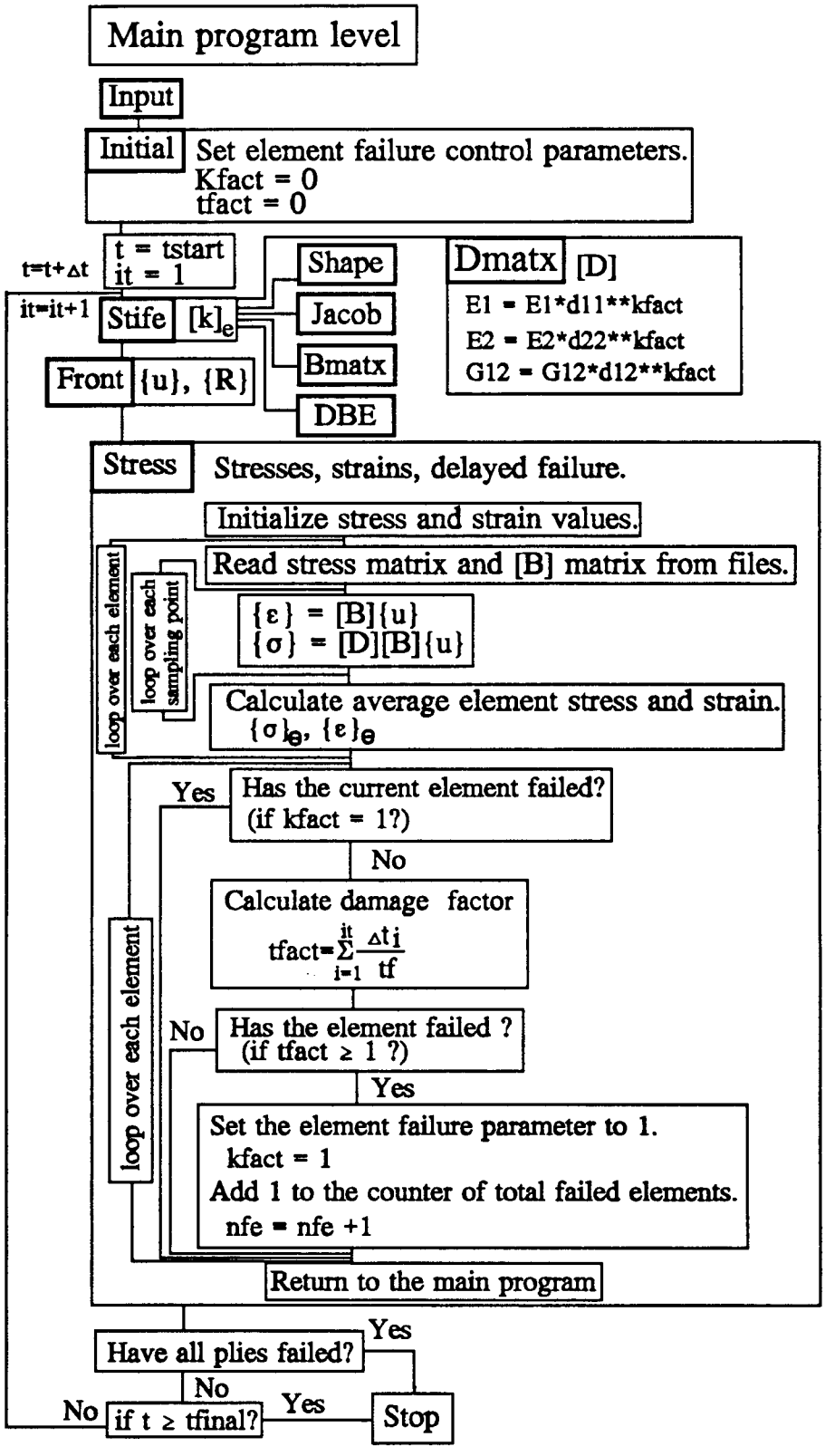


Figure 4.3 The algorithm of failure analysis.

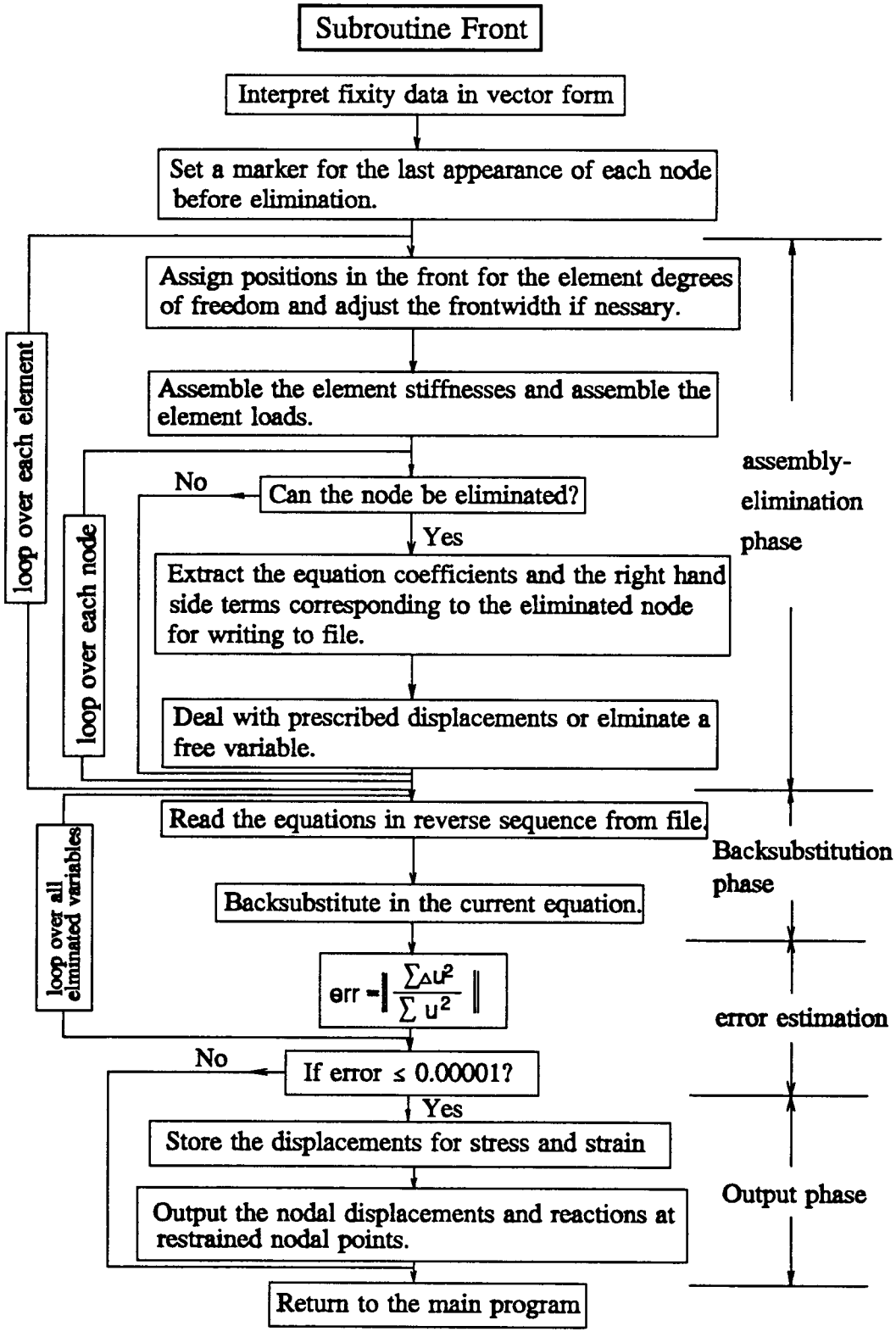


Figure 4.4 Operation sequence for frontal equation solution.

arrays listed in a DIMENSION statement in subroutine FRONT should be adjusted accordingly. Two parameters have been set in subroutine FRONT to make this adjustment easier. Appendix B gives more details about the array dimensions.

4.3.2 Common Blocks

Several common blocks are used to store the information requested in different subroutines. The often used common blocks are saved in files with the extension '.INC' in order to avoid the potential for multiple copies of the same common block to get out of step when modifications made to one of these copies are made incorrectly to one other of the copies. Then, INCLUDE *filename* statements are used so that the contents of the file named in the INCLUDE statement are placed at the point where the INCLUDE statement occurred.

File GLOBE.INC contains four common blocks named CONTRO, CONTRO2, LGDATA, and WORK. CONTRO and CONTRO2 store control parameters which are defined in subroutine INPUT and are used throughout the program. LGDATA stores a set of arrays. Care must be taken whenever the dimensions of the arrays need to be adjusted for solving a larger problem, and the DIMENSION statement in subroutine FRONT should be modified accordingly as well as the dimension of NDFRO in subroutine CHECK2. The third common block in file GLOBE.INC is called WORK. WORK stores a set of arrays which specifically relates to the element variables so that dimensions of the arrays in this common block do not change with the size of the problem.

File CRPIN.INC contains a common block named CRPIN. Common block CRPIN stores the nonlinear creep array variables to be initialized in subroutine INITIAL and to be used in subroutine DMATX. File CRP2.INC contains two common blocks named CREP and CREP2. CREP and CREP2 store two sets of nonlinear creep array variables calculated in subroutine DMATX to establish the nonlinear stress-strain relationships. Care also must be taken when the dimensions of the arrays need to be adjusted for a larger problem.

All the common blocks used in program LAMCREP are listed in appendix B. The rules for determining the dimensions of each array are also discussed in the same appendix.

CHAPTER 5

CASE STUDIES AND DISCUSSIONS OF THE RESULTS

A finite element computer program LAMCREP has been developed for analyzing time-dependent behavior of laminated composites. The algorithms and code structures of the program are described in Chapter 4. The objectives of this chapter are: a. to verify the program's solutions, and to illustrate its capabilities of solving complex problems by comparing results with calculations published by other researchers; b. to demonstrate the capabilities of the program for solving problems of three-dimensional, nonlinear viscoelastic laminated composite by analyzing time-dependent behavior of selected viscoelastic composite materials.

5.1 Material Properties

New high-temperature, graphite-fiber composite materials have been evaluated for lightweight and cost-effective aircraft structure over different types of composite materials. Besides the advantages of these materials, one major concern is their long term durability because of their viscoelastic behavior. In the current calculations, IM7/5260-H Graphite/Bismaleimide and T300/5208 Graphite/Epoxy were chosen for modeling their time-dependent viscoelastic behavior under different loads. The material properties of these composites are given below.

IM7/5260-H Graphite/Bismaleimide

The viscoelastic parameters for IM7/5260-H have been experimentally determined by the University of Washington and have the following values:

Elastic Properties

$E_{11} = 22.92 \times 10^6$ psi, $E_{22} = 1.3355 \times 10^6$ psi,

$\nu_{12} = \nu_{23} = 0.32$, $G_{12} = 0.821 \times 10^6$ psi.

$E_m = 5.46 \times 10^5$ psi, $\nu_m = 0.4$

Transverse Creep Properties

$D_{1T} = 1.073 \times 10^{-8}$ psi^{-1} $\lambda_{1T} = 0.0162$ min^{-1}

$D_{2T} = 4.675 \times 10^{-9}$ psi^{-1} $\lambda_{2T} = 0.1412$ min^{-1}

$D_{3T} = 8.895 \times 10^{-10}$ psi^{-1} $\lambda_{3T} = 0.1426$ min^{-1}

$D_{4T} = 4.895 \times 10^{-9}$ psi^{-1} $\lambda_{4T} = 0.544$ min^{-1}

$D_{fT} = 1.131 \times 10^{-11}$ psi^{-1} min^{-1}

$g_{0T} = 1 + 2.017 \times 10^{-5} \exp(\tau_{oct}/363.99)$

$g_{1T} = g_{2T} = a_T = 1$

Shear Creep Properties

$D_{1s} = 1.989 \times 10^{-8}$ psi^{-1} $\lambda_{1s} = 0.0137$ min^{-1}

$D_{2s} = 7.715 \times 10^{-10}$ psi^{-1} $\lambda_{2s} = 0.0157$ min^{-1}

$D_{3s} = 9.460 \times 10^{-9}$ psi^{-1} $\lambda_{3s} = 0.09$ min^{-1}

$D_{4s} = 1.000 \times 10^{-8}$ psi^{-1} $\lambda_{4s} = 0.279$ min^{-1}

$D_{fs} = 4.474 \times 10^{-11}$ psi^{-1}

$g_{0s} = 1 + 0.01023 \exp(\tau_{oct}/1747.73)$

$g_{1s} = 1 + 0.02747 \exp(\tau_{oct}/2878.04)$

$g_{2s} = 1 + 0.1071 \exp(\tau_{oct}/3308.2)$

$a_s = 1 - 0.0696 \exp(\tau_{oct}/2268.92)$

T300/5208 Graphite/Epoxy

The nonlinear viscoelastic properties of T300/5208 Graphite/Epoxy were determined by Tuttle and Brinson (Tuttle, M.E. and Brinson, H.F., 1986). To fit the creep model developed in Chapter 2, those values have been converted to an exponential series through a curve-fitting technique based on the Levenberg-Marquardt

method. The properties of this material are given below.

Elastic Properties

$E_{11} = 19.17 \times 10^6$ psi, $E_{22} = 1.368 \times 10^6$ psi,

$\nu_{12} = \nu_{23} = 0.273$, $G_{12} = 0.9297 \times 10^6$ psi.

Transverse Creep Properties

$D_{1T} = 9.522 \times 10^{-9}$ psi^{-1} $\lambda_{1T} = 1.861$ min^{-1}

$D_{2T} = 9.453 \times 10^{-9}$ psi^{-1} $\lambda_{2T} = 0.1128$ min^{-1}

$D_{3T} = 1.641 \times 10^{-8}$ psi^{-1} $\lambda_{3T} = 0.01184$ min^{-1}

$D_{4T} = 2.680 \times 10^{-8}$ psi^{-1} $\lambda_{4T} = 0.001523$ min^{-1}

$D_{5T} = 4.134 \times 10^{-8}$ psi^{-1} $\lambda_{5T} = 0.0002430$ min^{-1}

$D_{6T} = 8.915 \times 10^{-8}$ psi^{-1} $\lambda_{6T} = 0.0000368$ min^{-1}

$D_{fT} = 7.102 \times 10^{-13}$ psi^{-1} min^{-1}

$g_{0T} = 1$

$$g_{1T} = \begin{cases} 1 & \text{for } \tau_{oct} \le 932.6 \text{ psi} \\ 1 + 6.033 \times 10^{-4}(\tau_{oct} - 932.6) & \text{for } \tau_{oct} > 932.6 \text{ psi} \end{cases}$$

$g_{2T} = 1$

$$a_T = \begin{cases} 1 & \text{for } \tau_{oct} \le 932.6 \text{ psi} \\ \exp(-1.703 \times 10^{-3}(\tau_{oct} - 932.6)) & \text{for } \tau_{oct} > 932.6 \text{ psi.} \end{cases}$$

Shear Creep Properties

$D_{1s} = 2.297 \times 10^{-8}$ psi^{-1} $\lambda_{1s} = 2.097$ min^{-1}

$D_{2s} = 1.835 \times 10^{-8}$ psi^{-1} $\lambda_{2s} = 0.1251$ min^{-1}

$D_{3s} = 2.916 \times 10^{-8}$ psi^{-1} $\lambda_{3s} = 0.01309$ min^{-1}

$D_{4s} = 4.346 \times 10^{-8}$ psi^{-1} $\lambda_{4s} = 0.001682$ min^{-1}

$D_{5s} = 6.299 \times 10^{-8}$ psi^{-1} $\lambda_{5s} = 0.000267$ min^{-1}

$D_{6s} = 1.251 \times 10^{-7}$ psi^{-1} $\lambda_{6s} = 0.00003955$ min^{-1}

$D_{fs} = 9.39 \times 10^{-13}$ psi^{-1} min^{-1}

$$g_{0s} = \begin{cases} 1 & \text{for } \tau \le 1748 \text{ psi} \\ 1 + 3.537 \times 10^{-5}(\tau_{oct} - 1748) & \text{for } \tau_{oct} > 1748 \text{ psi} \end{cases}$$

$$g_{1s} = \begin{cases} 1 & \text{for } \tau_{oct} \le 1049 \text{ psi} \\ 1 + 6.75 \times 10^{-5}(\tau_{oct} - 1049) & \text{for } \tau_{oct} > 1049 \text{ psi} \end{cases}$$

$$g_{2s} = \begin{cases} 1 & \text{for } \tau_{oct} \le 1049 \text{ psi} \\ 1 + 8.55 \times 10^{-5}(\tau_{oct} - 1049) & \text{for } \tau_{oct} > 1049 \text{ psi} \end{cases}$$

$$a_s = \begin{cases} 1 & \text{for } \tau_{oct} \le 2103 \text{ psi} \\ \exp(-2.344 \times 10^{-4}(\tau_{oct} - 2103) & \text{for } \tau_{oct} > 2103 \text{ psi} \end{cases}$$

where τ_{oct} is the octahedral shear stress in the matrix material defined as

$$\tau_{oct} = \frac{1}{3}\left[(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_x - \sigma_z)^2 + 6\left(\tau_{xy}^2 + \tau_{xz}^2 + \tau_{yz}^2\right)\right]^{1/2} . \qquad (5.1)$$

5.2 Tensile Loading of Unnotched Laminates

Rectangular unnotched laminated plates subjected to uniform tensile stress are analyzed for different loading levels and different lay-ups.

5.2.1 Creep strain response to uniform tensile stress

5.2.1.1 Description of the Problem

To test the creep model of LAMCREP, problems of laminates subjected to constant tensile loads were chosen to be analyzed first. The first case considered here is a unidirectional laminate subjected to a constant tensile stress in the direction normal to the fibers. In this situation, the load is carried by matrix material only. Analyses were done for both IM7/5260-H and T300/5208 with constant tensile stresses

of σ = 5,000 psi and σ = 3,000 psi respectively. The analytical results for this case were obtained in a manner similar to that by Lou and Schapery (Lou, Y.C. and Schapery, R.A., 1971) creep equation, i.e.,

$$\varepsilon = g_o D_o \sigma + g_1 g_2 \sigma \left[\sum_{r=1}^{N} D_r \left(1 - e^{1 - e^{-\lambda_r t/a}}\right) + D_f t/a \right] . \qquad (5.2)$$

Comparisons of the analytical solution and program solution are given in figure 5.1 and Figure 5.2, where the creep strains are plotted versus time. The agreement between finite element solution and analytical solution is excellent.

The second case considered here is a multidirectional laminate of [90/45/-45/90]$_s$ subjected to a uniform tensile stress in global x-direction. The tensile stress for this case is σ = 10,000 psi, and the material modeled is T300/5208. Figure 5.3 shows the creep strain versus time for this case. Since a viscoelastic analytical solution is not available for the mutildirectional plate problem, the finite element solution is compared with the classical lamination theory (CLT) solution. Figure 5.3 shows excellent agreement between these two solutions.

5.2.1.2 Discussion

The nonlinear viscoelastic model in LAMCREP has been verified by the excellent agreement between the results from LAMCREP and those obtained from analytical analysis and classical laminate theory.

It is observed that the creep strain in both materials increases rapidly first and then continues to increase at a decreasing rate. Comparision of Figures 5.2 and 5.3 shows that for a T300/5208 laminate with [90/45/-45/90]$_s$ lay-up, 233% increment of the applied load leads to only 11% increment in deformation in the loading direction. This indicates that the strength of the laminate is increased by the fibers in +45 and -45-degree layers.

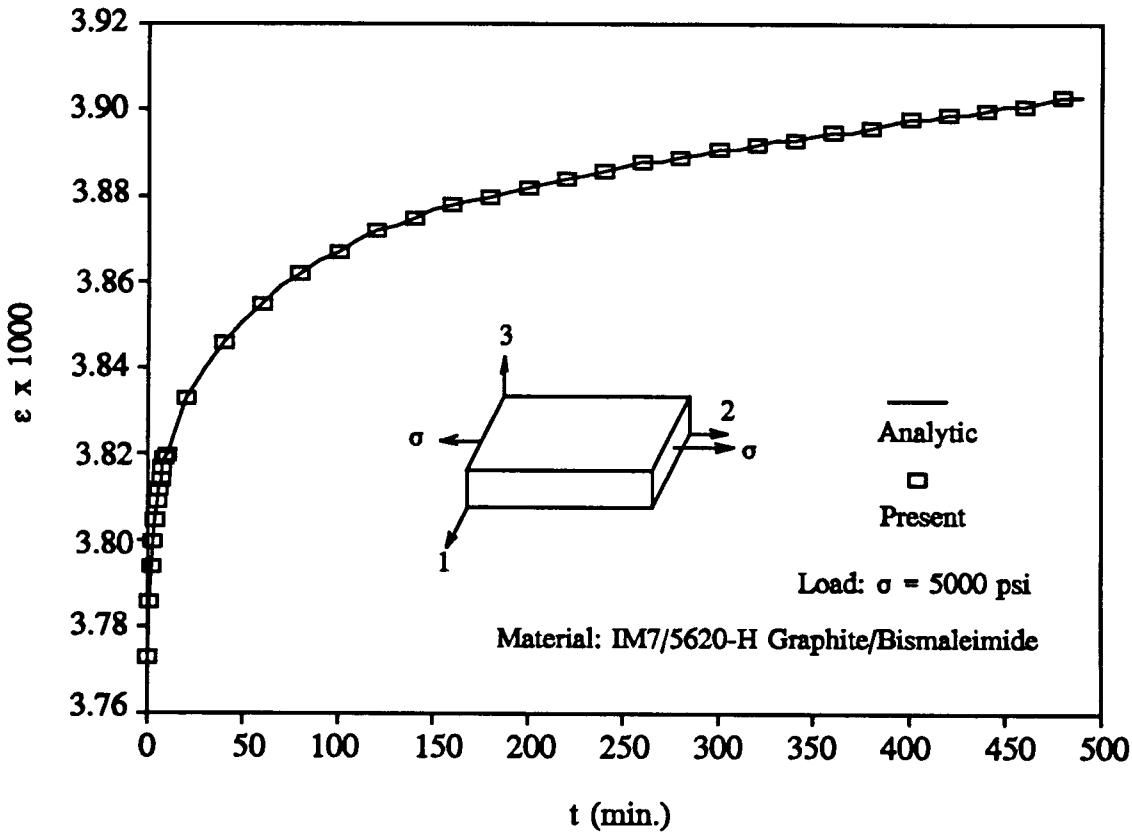


Figure 5.1 Strain versus time for a unidirectional laminate (IM7/5260-H) under tension.

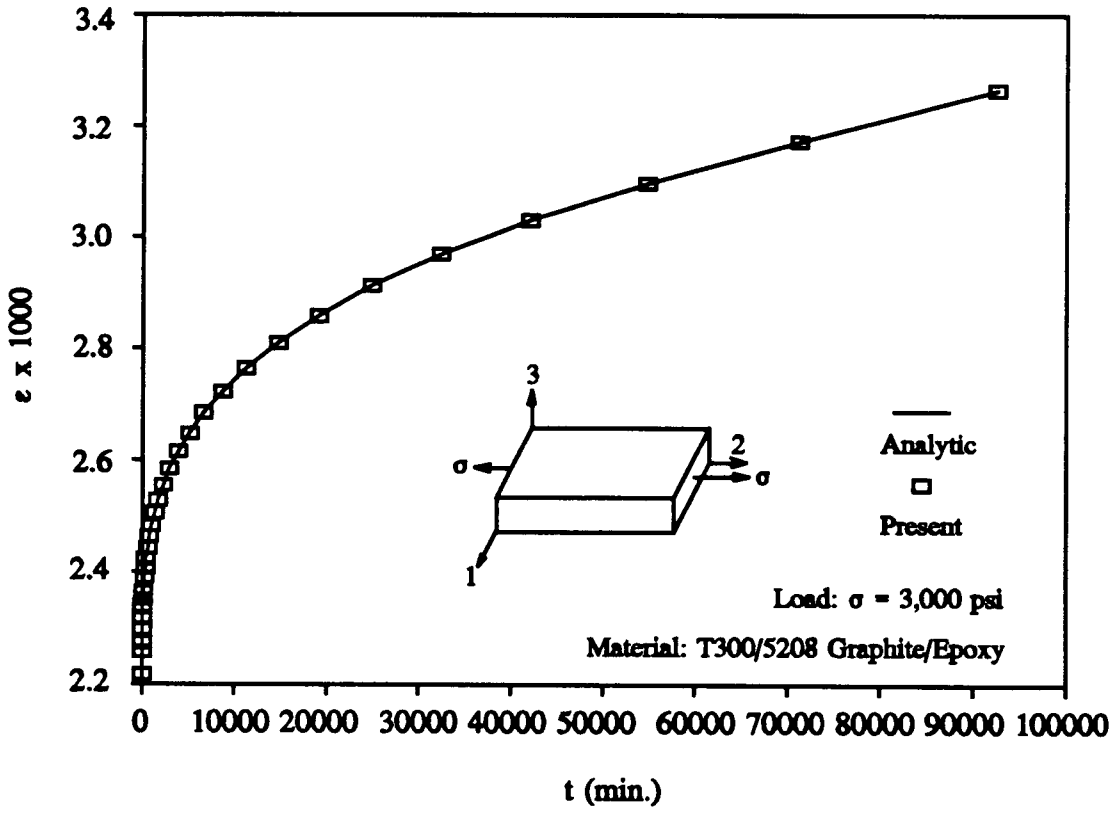


Figure 5.2 Strain versus time for a unidirectional laminate (T300/5208) under tension.

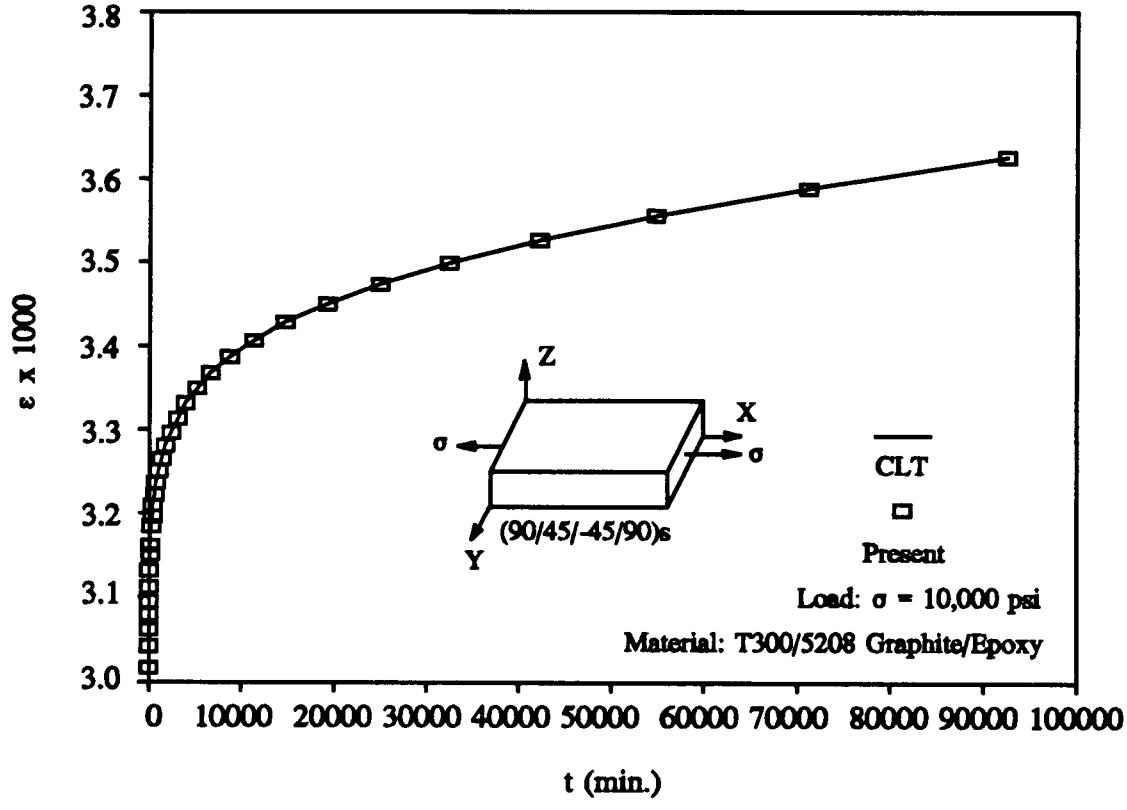


Figure 5.3 Strain versus time for a $[90/45/-45/90]_s$ laminate under tension.

5.2.2 Free Edge Effect

When a laminated plate is subjected to a tension load, the out-plane stress (interlaminar stress) profiles near the traction free edges are important in delamination predictions. The classic plate theory cannot give a realistic solution due to the plane stress and plane strain assumptions. In this situation, the finite element method is the most powerful tool for solving general three-dimensional problems. The next application of LAMCREP shown here is to analyze an unnotched multidirectional laminate under tension. By analyzing the interlaminar stresses between layers, the location of the maximum interlaminar stress is assumed as the point where delamination is most likely to occur.

5.2.2.1 Description of the Problem

An elastic analysis was performed first for verification purposes. The physical model considered in this analysis is a $[90/0]_s$ laminate subjected to constant tensile stress, as shown in Figure 5.4, where b is half the width and h_o is the thickness of each ply. Figure 5.5 shows the finite element mesh and geometries for this problem. Due to symmetry, only one-eighth of the laminate was modeled with five elements in the x-direction, two elements through the thickness of each ply and ten unequally spaced elements in y-direction. As suggested by Wang and Crossman, the thickness-to-width ratio of the laminate is taken as (2t):(2b) = 1:4. The mesh becomes finer as y near the traction free edge and the ratio between the first and last element in y-direction is 17. A comparison of LAMCREP's result with those of Wang and Crossman (Wang, A.S.D. and Crossman, F.W., 1977) is shown in Figure 5.6 where the distribution of interlaminar stress σ_z is plotted versus position y/b along the middle plane of the laminate at z=0. The agreement between the two solutions verifies the adequacy of the finite element mesh. Similar analysis is then performed for viscoelastic response of IM7/5260 and T300/5208, under tensile load of 10,000 psi and 8,000 psi, respectively. The interlaminar stress distributions for three different time

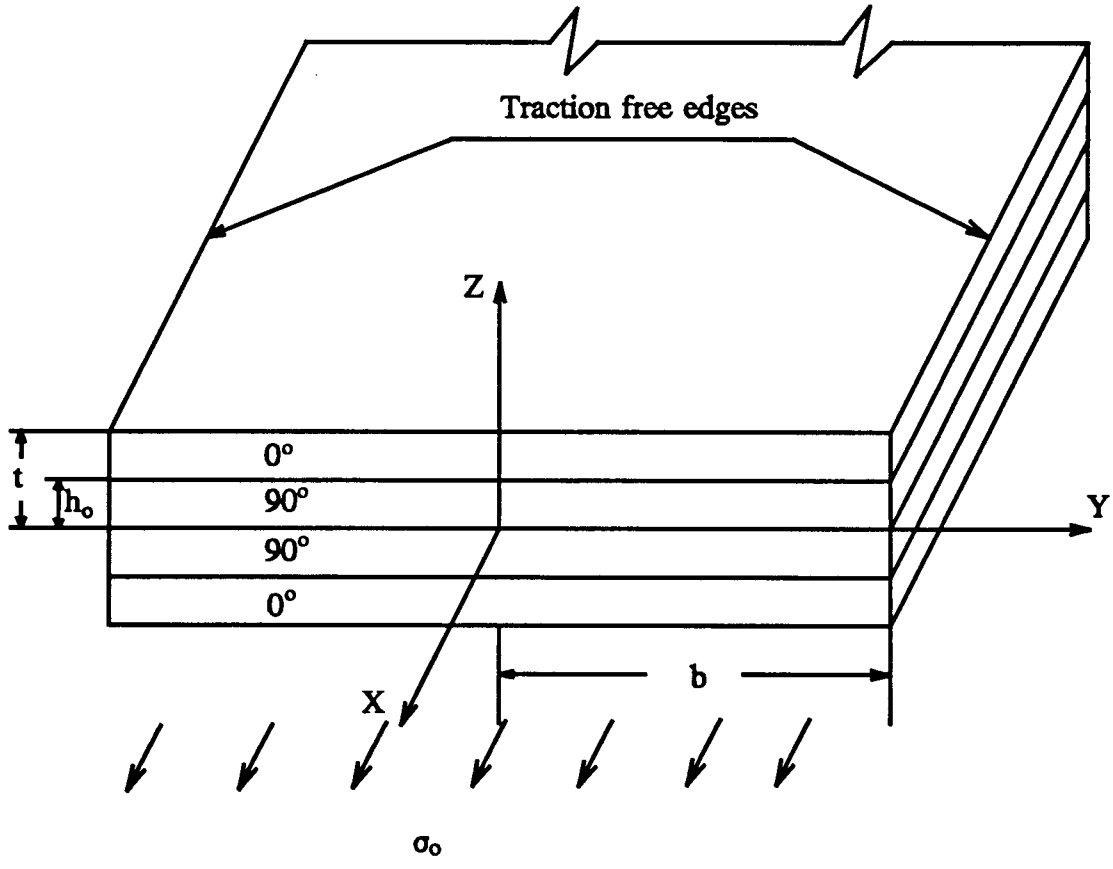


Figure 5.4 A rectangular laminated plate under tension.

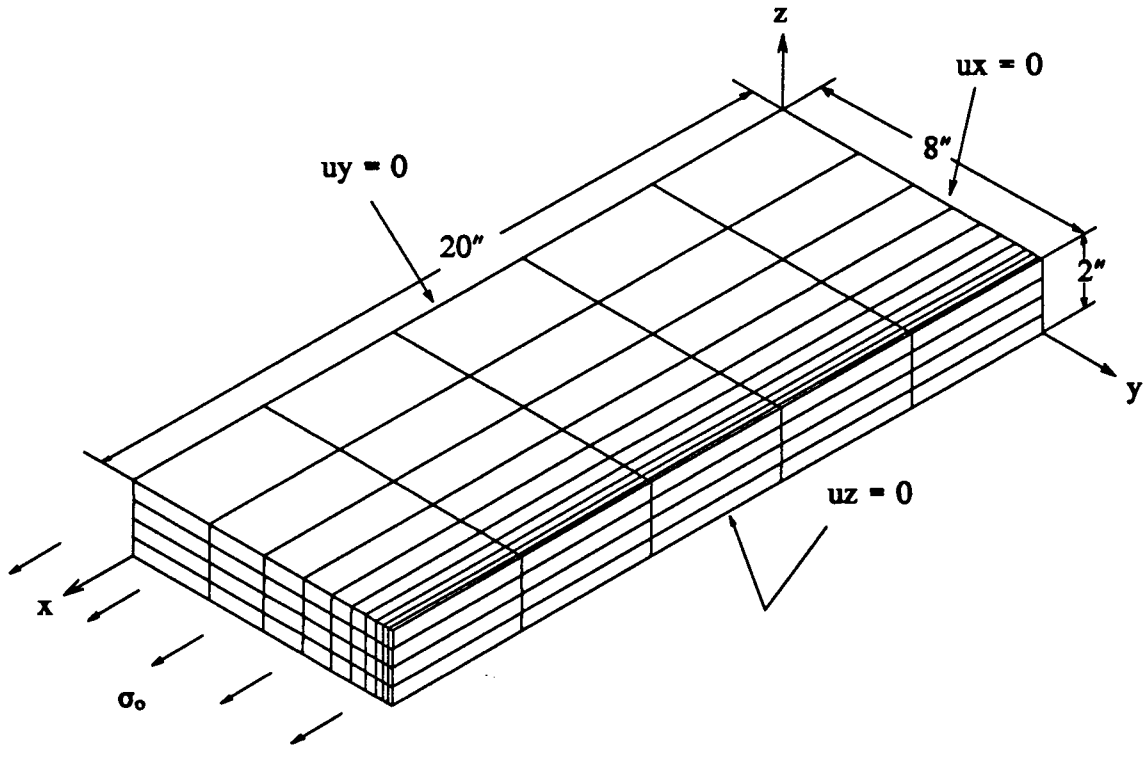


Figure 5.5 Finite element mesh for a $[0/90]_s$ laminate under tension.

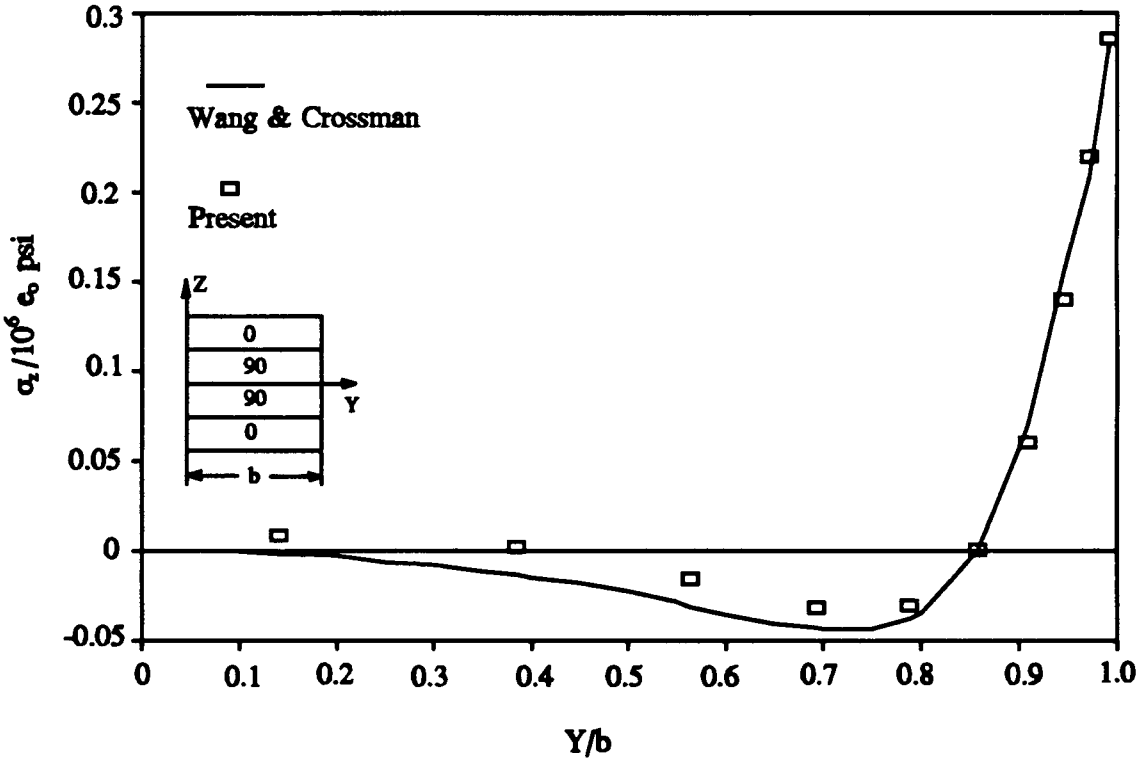


Figure 5.6 Normalized interlaminar normal stress versus position for a [0/90/90/0] laminate under tension (elastic response).

intervals, over a period of approximately two months, are shown in Figure 5.7 and Figure 5.8.

In addition, the same analysis is performed for a $[90/0/-45/45]_s$ laminated composite plate. A finite element mesh similar to the previous one was generated and is shown in Figure 5.9. The elastic verification results for this problem is shown in Figure 5.10, and the viscoelastic results in Figure 5.11.

5.2.2.2 Discussion

The correctness of LAMCREP has been verified by the elastic verification results. The interlaminar stress σ_z rises dramatically toward the free edge of laminate composites due mainly to mismatches in layer properties. This may initialize failure by delaminations at the edges of laminate structures. For laminates made of viscoelastic materials, the same results are obtained but with some additional time dependent phenomena. Figures 5.7, 5.8, and 5.11 indicate that there are reductions in the σ_z over time. Similar behavior was observed by Lin and Yi (Lin, K.Y. and Yi, S., 1991) in their linear viscoelastic analysis. Results obtained here show that over a period of approximately two months, σ_z decreases 78 percent for the IM7/5260-H $[0/90]_s$ laminate under σ of 10,000 psi, and σ_z decreases 25 percent for the T300/5208 $[90/90]_s$ laminate where a 20 percent reduction is obtained in the first 23 days. For $[90/0/-45/45]_s$ laminate, σ_z rises in value at the free edge but with opposite sign. This suggests that different lay-ups of laminates behave fundamentally differently and might be favorable for different designs.

5.3 Bending of a Thick Laminated Plate

5.3.1 Description of the Problem

To further evaluate the performance of the program, the response of a thick, simply supported laminated plate subjected to bending was analyzed. A square

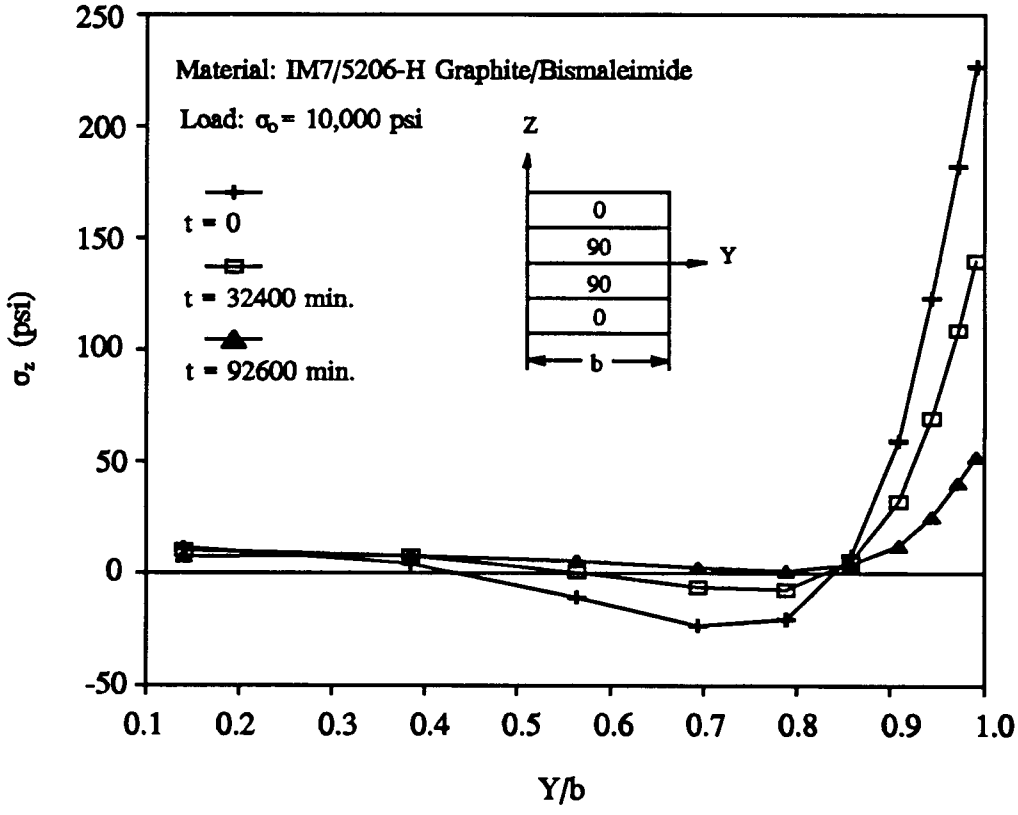


Figure 5.7 Interlaminar normal stress versus position for a $[0/90]_s$ laminate of IM7/5260-H under tension (viscoelastic response).

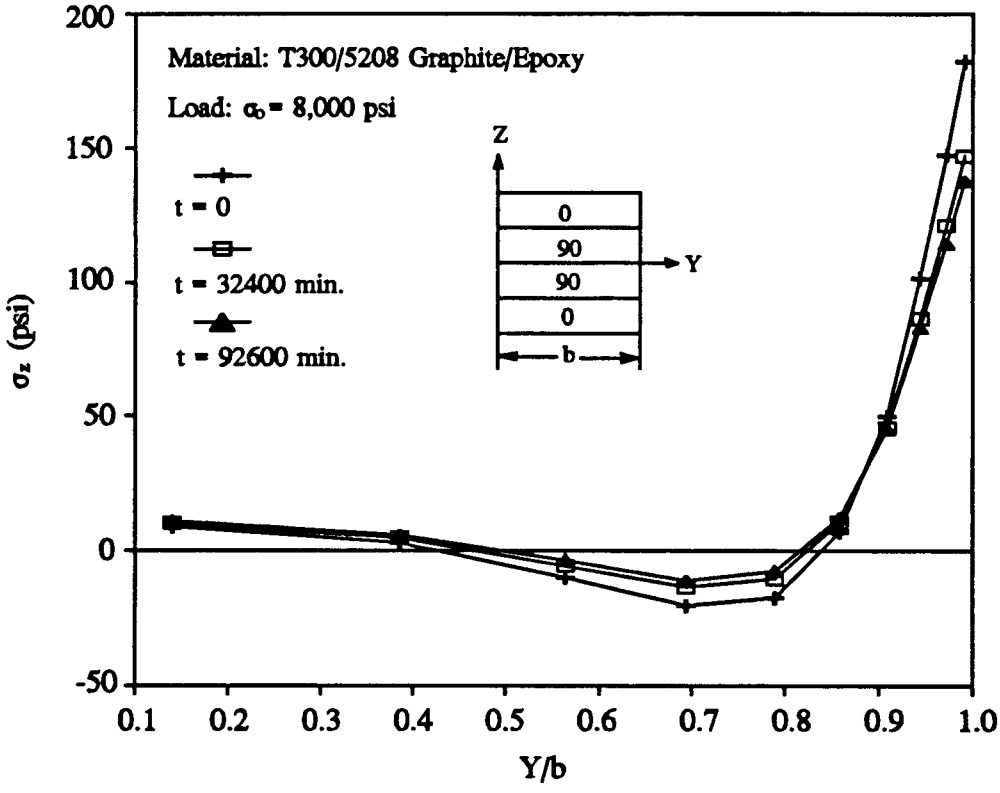


Figure 5.8 Interlaminar normal stress versus position for a $[0/90]_s$ laminate of T300/5208 under tension (viscoelastic response).

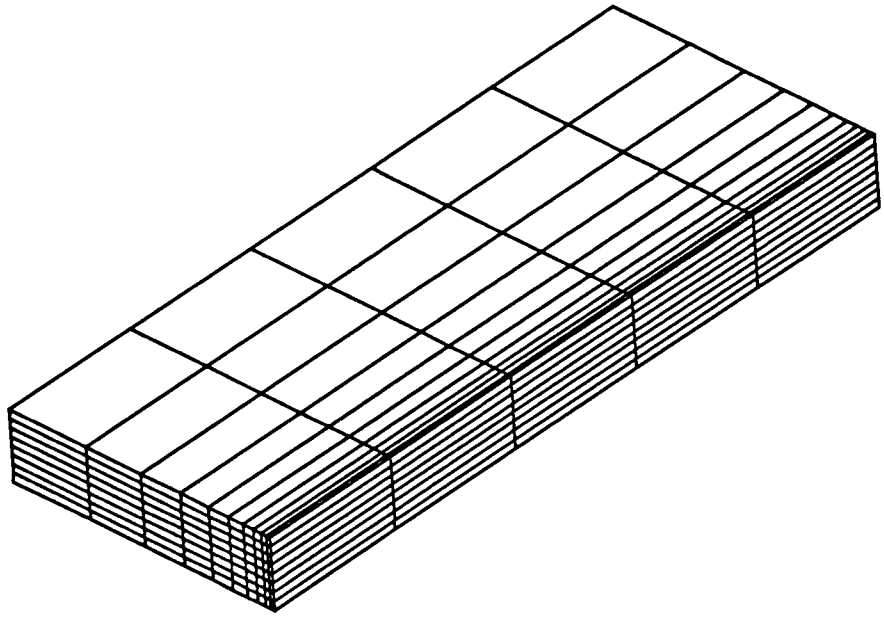


Figure 5.9 Finite element mesh for a $[90/0/-45/45]_s$ laminate.

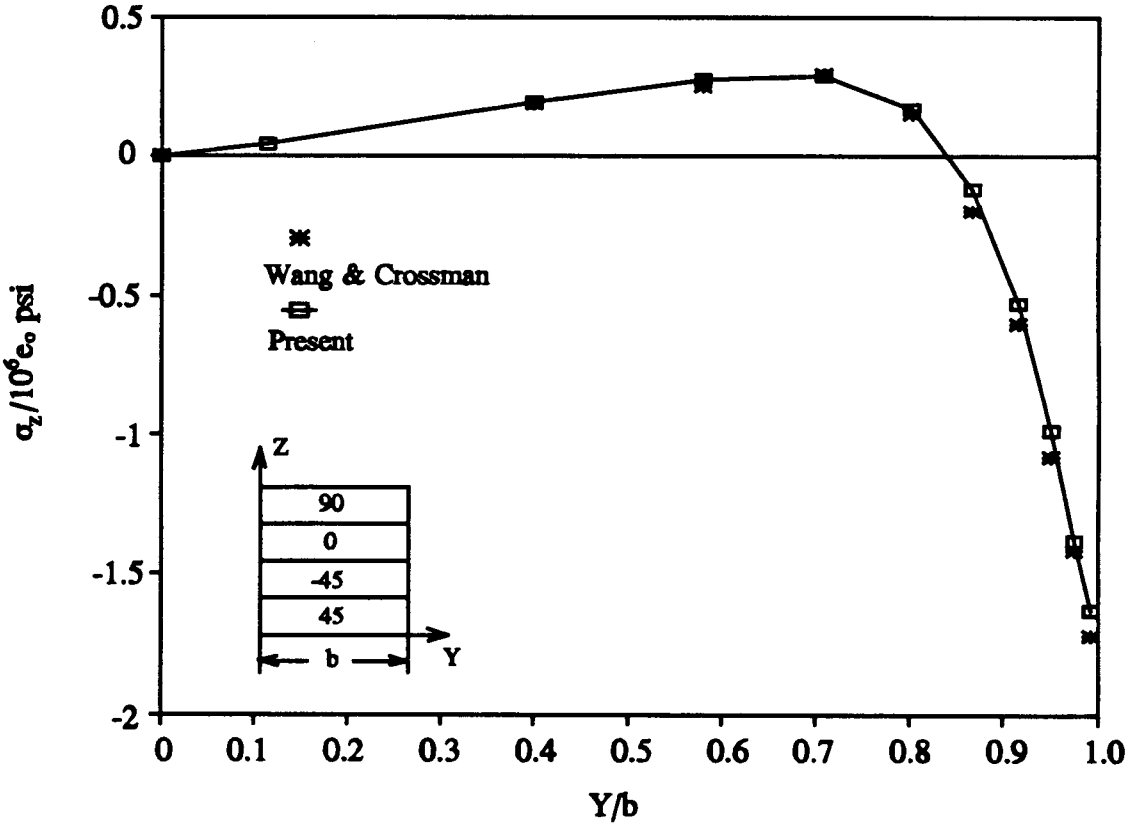


Figure 5.10 Normalized interlaminar normal stress versus position for a $[90/0/-45/45]_s$ laminate under tension (elastic response).

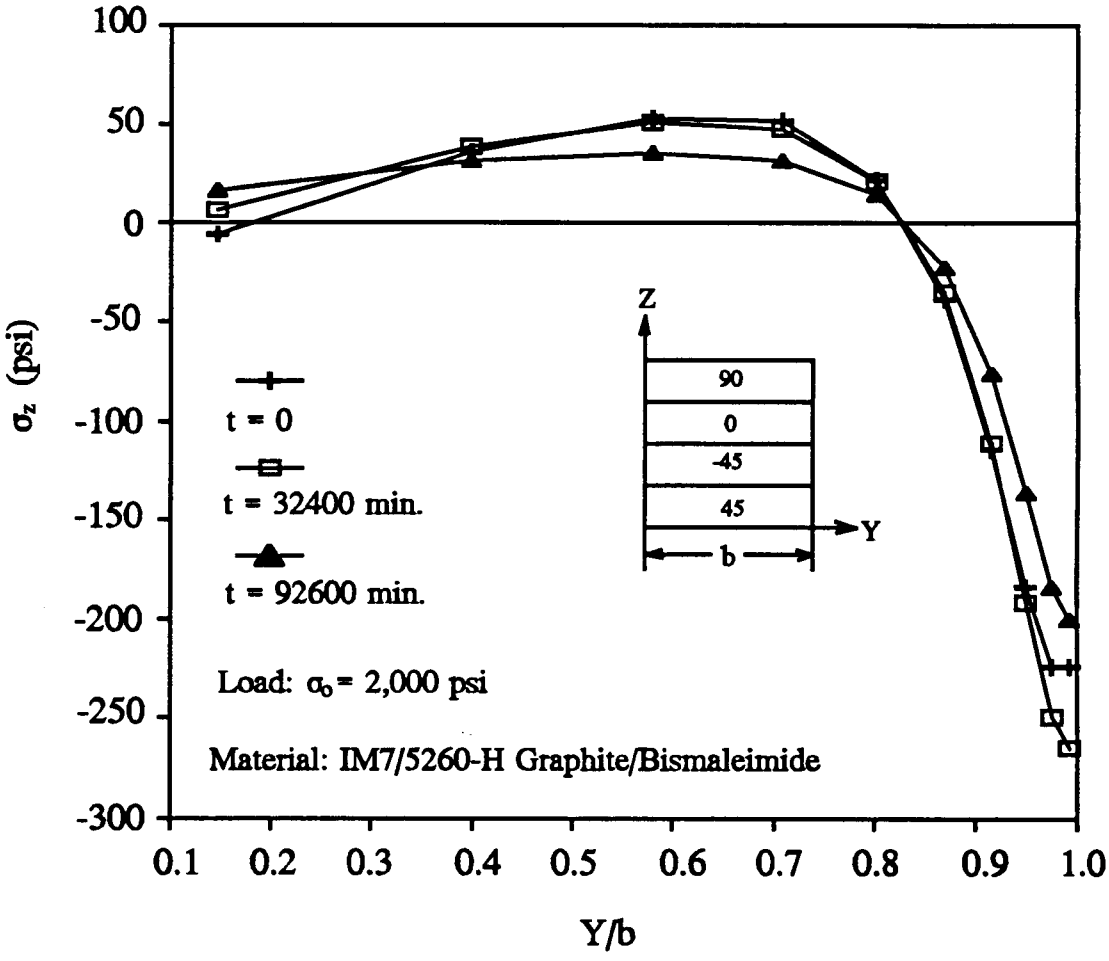


Figure 5.11 Interlaminar normal stress versus position for a $[90/0/-45/45]_s$ laminate of IM7/5260-H under tension (viscoelastic response).

[0/90/0] laminate is chosen to compare with the results of an elasticity solution given by Pagano (Pagano, N.J., 1970). The plate is simply supported along all edges and subjected to a nonuniformly distributed pressure on the top surface. The geometries and boundary conditions of this problem are shown in Figure 5.12 as

at x = 0, a, w = v = 0,

at y = 0, b, w = u = 0.

where u,v and w denote the displacements in x, y and z-directions respectively. The orientations of the fibers are defined as follows: for zero degree layers, the longitudinal direction of the fibers is parallel to the x-axis, whereas the ninety degree layer orientation designates fibers transverse to x and parallel to the y-axis.

Only one-fourth of the plate was modeled due to the symmetry . Thus, additional boundary conditions are imposed on the plane of symmetry in the finite element mesh given as

at x = a/2, w = 0,

at y = b/2, u = 0.

The finite element model is meshed with nine elements in the plane and two elements through the thickness of each ply for a total of 54 elements in the mesh (See Figure 5.13).

The pressure is sinusoidally distributed as a function of position, and can be written as

$$\sigma(x,y) = \sigma_o \sin\frac{\pi x}{a} \sin\frac{\pi y}{a}, \tag{5.3}$$

where σ_o is intensity of the sinusoidal load. Since LAMCREP takes only concentrated forces at nodal points, the pressure was converted into consistent element joint loads by performing the following integration:

$$\{P_i\} = \int_{-1}^{1}\int_{-1}^{1}\{N_i\}\{N_i\}^T\{\overline{P_i}\}\,|J|\,d\xi d\eta \tag{5.4}$$

where $\{P_i\}$ is the list of consistent joint loads on the loading surface of an element,

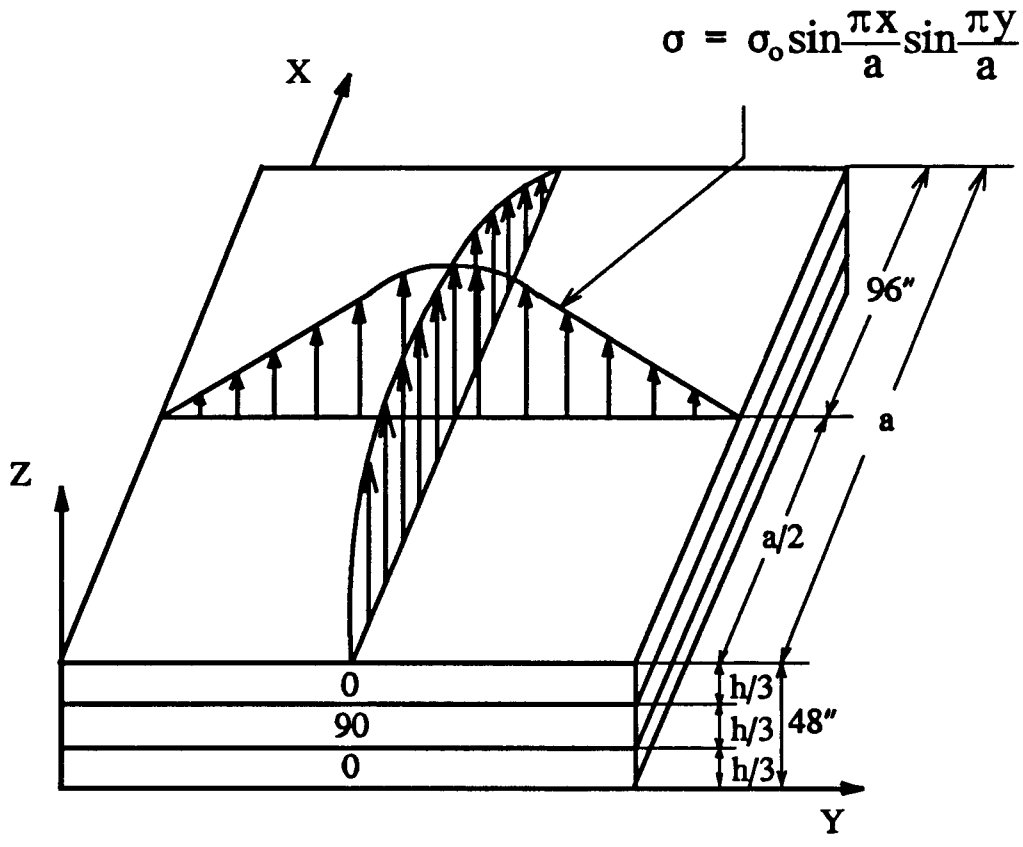


Figure 5.12 Thick, simply supported laminated ([0/90/0]) plate subjected to bending.

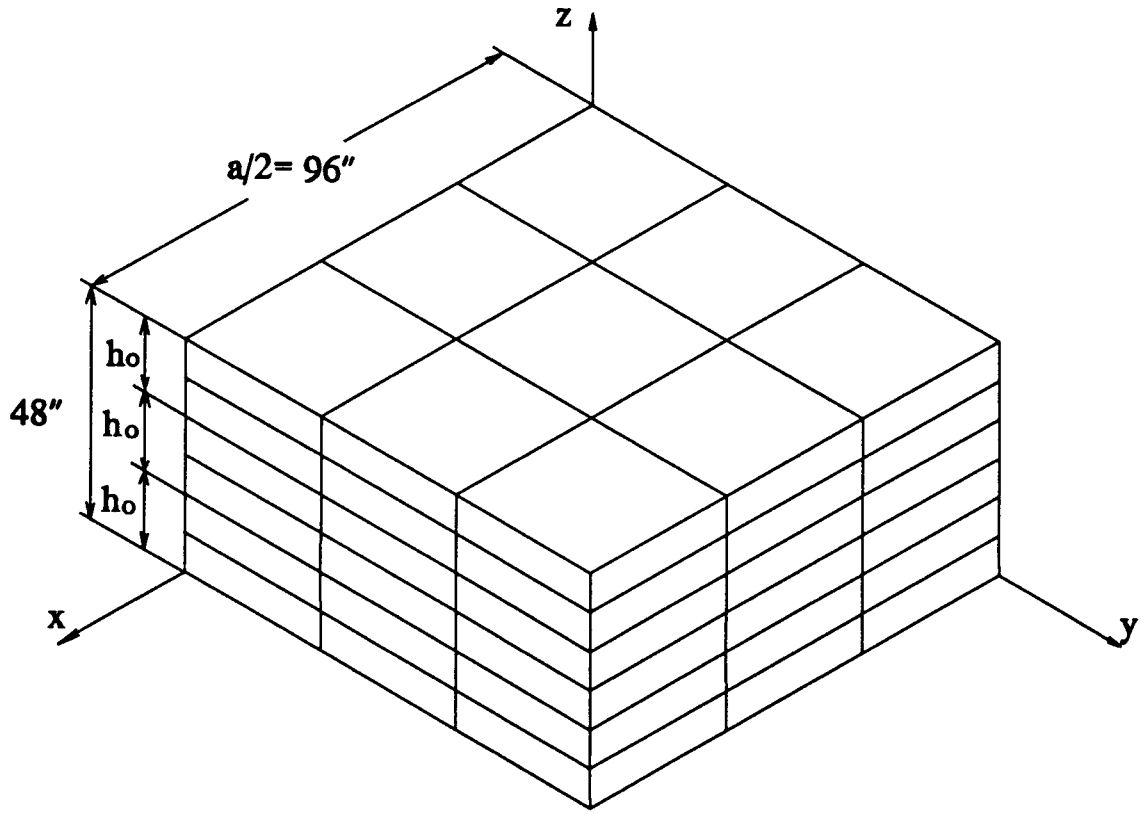


Figure 5.13 Finite element mesh for a thick, simply supported laminated plate.

$\{\overline{p}_i\}$ is the list of the pressure values evaluated at each particular node on the loading surface of the element (See Figure 5.14). For the 20-node isoparametric solid element used in the program, the integration of Equation (5.4) gives the following result which converts nonuniformly distributed pressure into consistent nodal forces

$$\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \end{bmatrix} = AB \times \tag{5.5}$$

$$\begin{bmatrix} 0.01852 & -0.0185 & 1.5E\text{-}9 & -0.037 & 0.00926 & -0.037 & 1.5E\text{-}09 & -0.0185 \\ -0.0185 & 0.1481 & -0.0185 & 0.1111 & -0.037 & 0.07407 & -0.037 & 0.1111 \\ 1.5E\text{-}09 & -0.0185 & 0.01852 & -0.0185 & 1.5E\text{-}09 & -0.037 & 0.00926 & -0.037 \\ -0.037 & 0.1111 & -0.0185 & 0.1481 & -0.0185 & .1111 & -0.037 & 0.07407 \\ 0.00926 & -0.037 & 1.5E\text{-}09 & -0.0185 & 0.01852 & -0.0185 & 1.5E\text{-}09 & -0.037 \\ -0.037 & 0.07407 & -0.037 & 0.1111 & -0.0185 & 0.1481 & -0.0185 & 0.1111 \\ 1.5E\text{-}09 & -0.037 & 0.00926 & -0.037 & 1.5E\text{-}09 & -0.0185 & 0.01852 & -0.0185 \\ -0.0185 & 0.1111 & -0.037 & 0.07407 & -0.037 & 0.1111 & -0.0185 & 0.1481 \end{bmatrix} \begin{bmatrix} \overline{p_1} \\ \overline{p_2} \\ \overline{p_3} \\ \overline{p_4} \\ \overline{p_5} \\ \overline{p_6} \\ \overline{p_7} \\ \overline{p_8} \end{bmatrix}$$

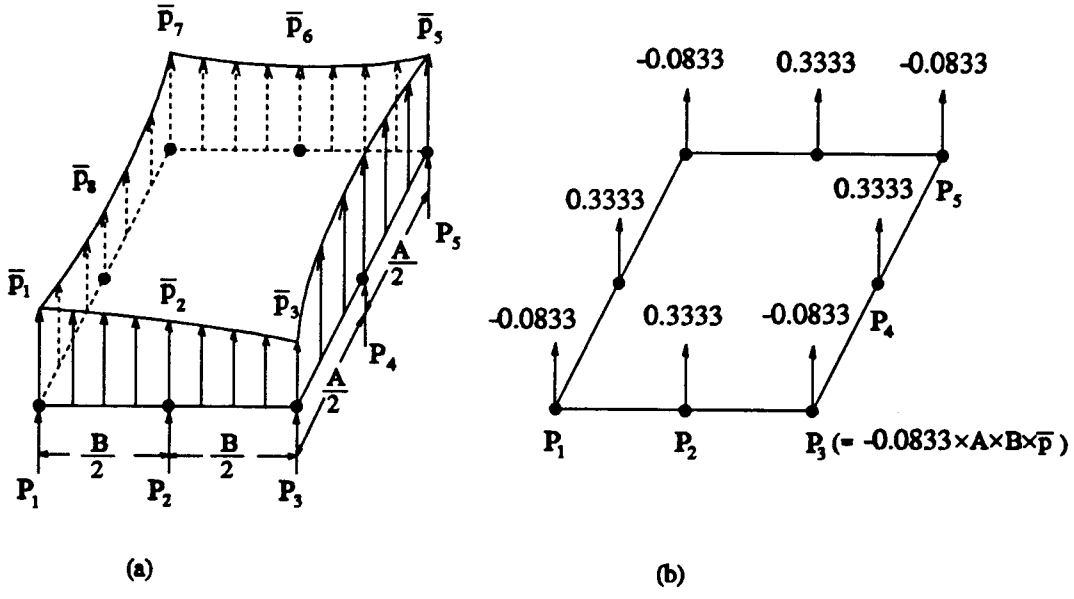


Figure 5.14 Equivalent joint loads for distributed loading.

where A and B are as shown in Figure 5.14. For a uniformly distributed load, the coefficients to calculate consistent joint loads are given in Figure 5.14b.

Again, an elastic analysis was performed first to verify the correctness of program solution and adequacy of the finite element mesh for solving this problem. The material properties used in elastic analysis are

$$E_{11} = 25 \times 10^6 \text{ psi} \qquad E_{22} = E_{33} = 1 \times 10^6 \text{ psi}$$

$$G_{12} = G_{13} = 0.5 \times 10^6 \text{ psi} \qquad G_{23} = 0.2 \times 10^6 \text{ psi}$$

$$\nu_{13} = \nu_{23} = 0.25 \qquad \nu_{12} = 0.01$$

The intensity of the sinusoidal load σ_o has the value of 1. Comparisons of the program solutions and the elastic analytical solutions given by Pagano (Pagano, N.J., 1970) are shown in Figure 5.15-5.17. As suggested by Pagano in the elasticity analysis, the following normalizations to the stresses make the results independent of the span-to-depth ratio and the magnitude of stress σ_o:

$$\left(\bar{\sigma}_x, \bar{\sigma}_y, \bar{\tau}_{xy}\right) = \frac{1}{\sigma_o S^2}\left(\sigma_x, \sigma_y, \tau_{xy}\right)$$

$$\left(\bar{\tau}_{xz}, \bar{\tau}_{yz}\right) = \frac{1}{\sigma_o S}\left(\tau_{xz}, \tau_{yz}\right) \tag{5.5}$$

$$S = \frac{a}{h}, \quad \bar{z} = \frac{z}{h}$$

where a is the width of the plate, h is the total thickness of the plate, i.e., $h = 3h_o$, and S is the span-to-depth ratio, in this particular case, S = 4. In Pagano's analysis, the distributions of σ_x at (a,a), τ_{xy} at (0,0) and τ_{xz} at (0,a) were plotted through the thickness h. Because LAMCREP calculates stresses at Gauss integration points, the distribution of σ_x at (0.993a, 0.993a), τ_{xy} at (0.007,0.007) and τ_{xz} at (0.007, 0.007a) were plotted through the thickness h for the program solution. Figures 5.15 through 5.17 show that agreement with the elasticity solution is excellent.

A nonlinear viscoelastic analysis was conducted to study the time-dependent response for bending of a thick plate. The intensity of the sinusoidal load σ_o was 3,500 psi for IM7/5260-H and 3,000 psi for T300/5208. The results are plotted in Figures 5.18-5.23.

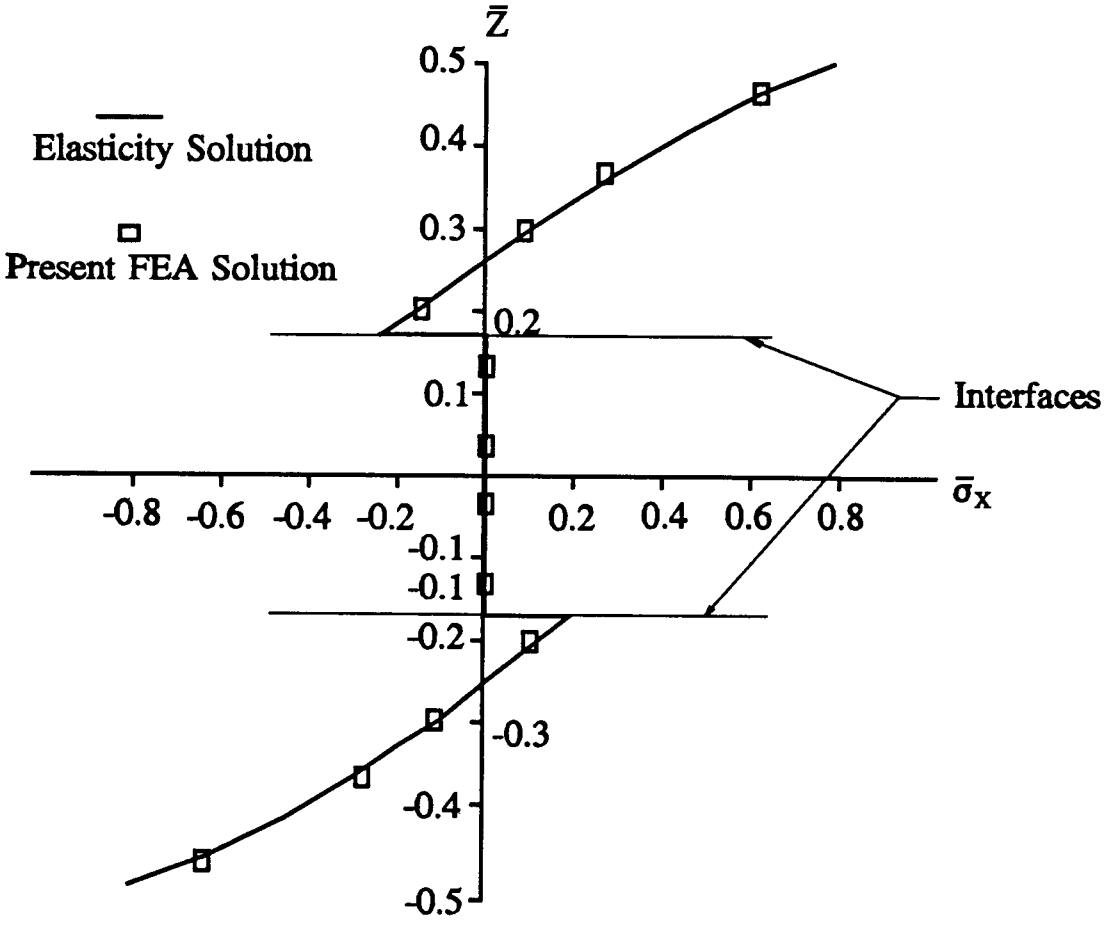


Figure 5.15 Normalized, in-plane normal stress versus position for elastic response.

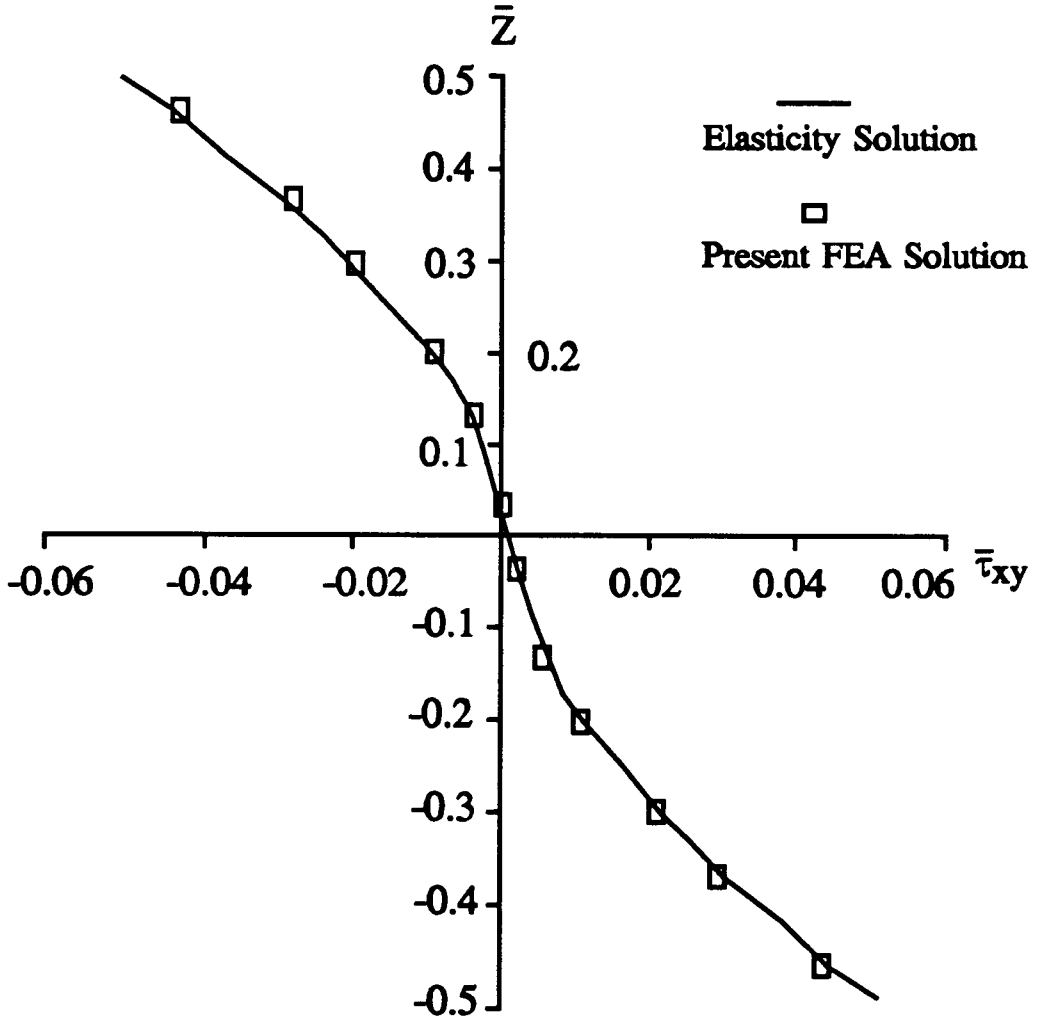


Figure 5.16 Normalized in-plane shear stress versus position for elastic response.

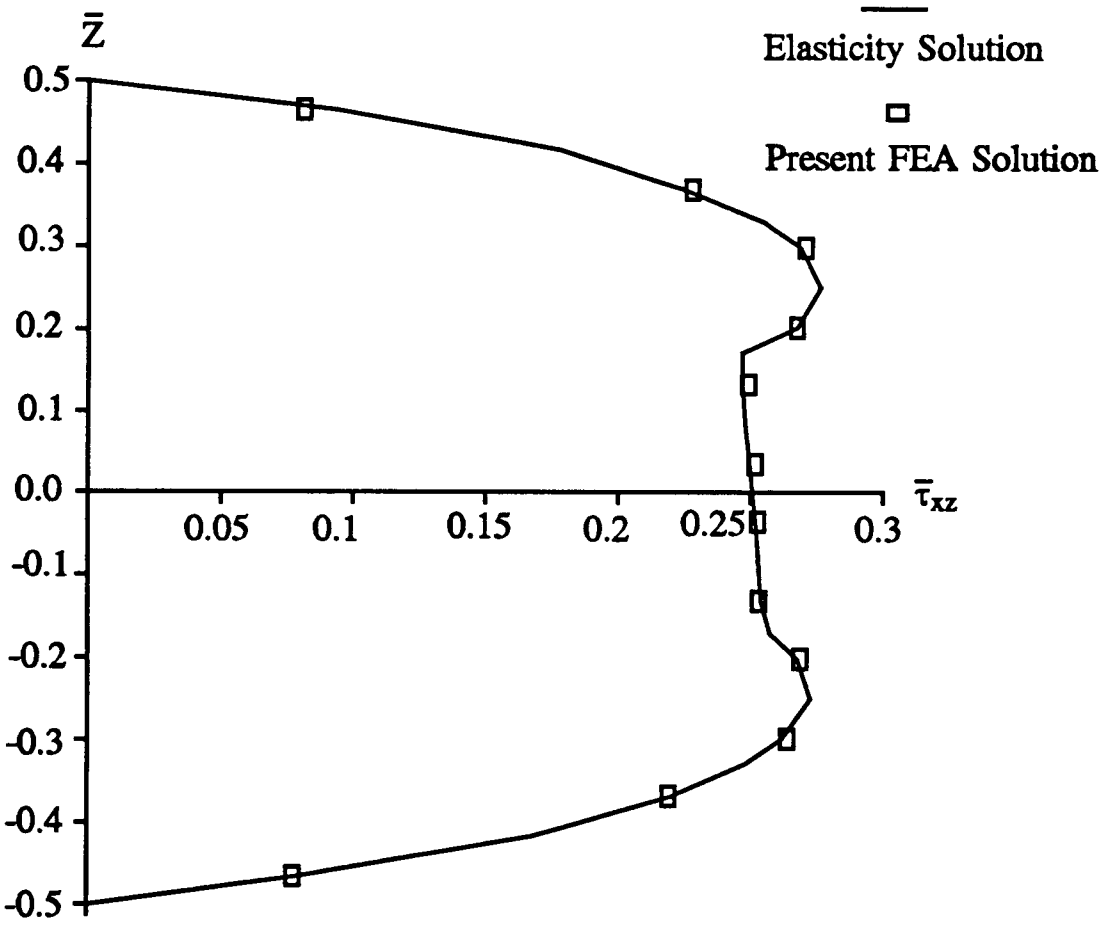


Figure 5.17 Normalized interlaminar shear stress versus position for elastic response.

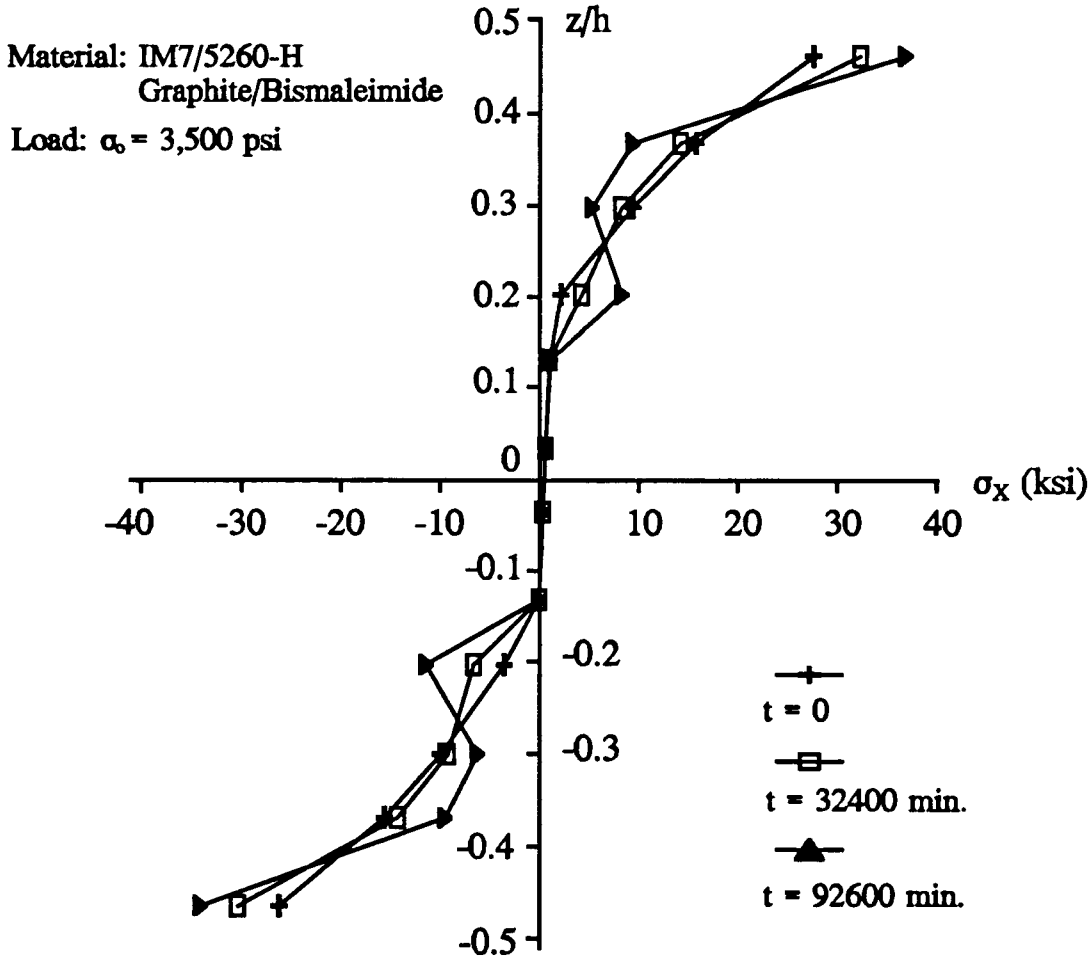


Figure 5.18 In-plane normal stress versus position for a thick, [0/90/0] laminate of IM7/5260-H under bending (viscoelastic response).

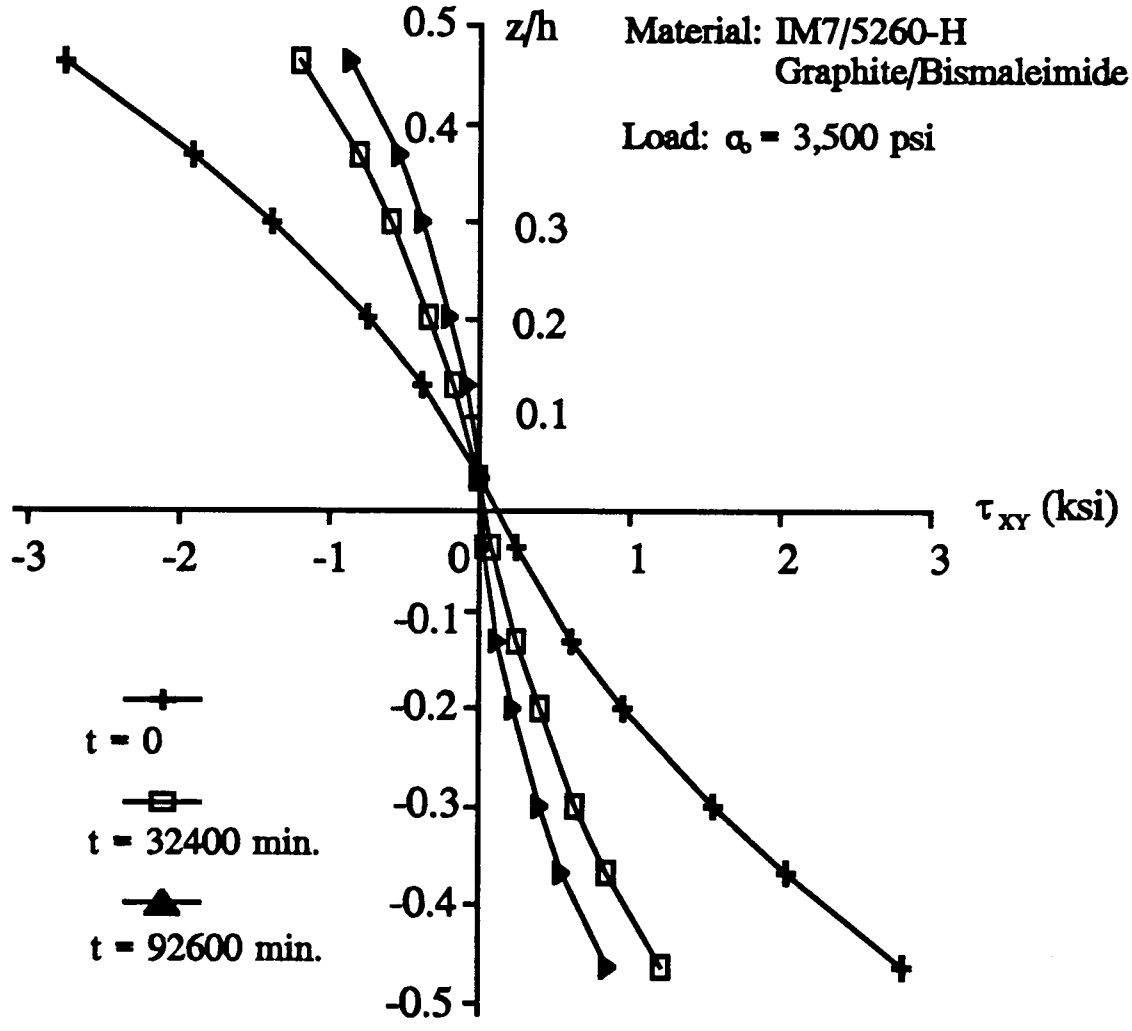


Figure 5.19 In-plane shear stress versus position for a thick [0/90/0] laminate of IM7/5260-H under bending (viscoelastic response).

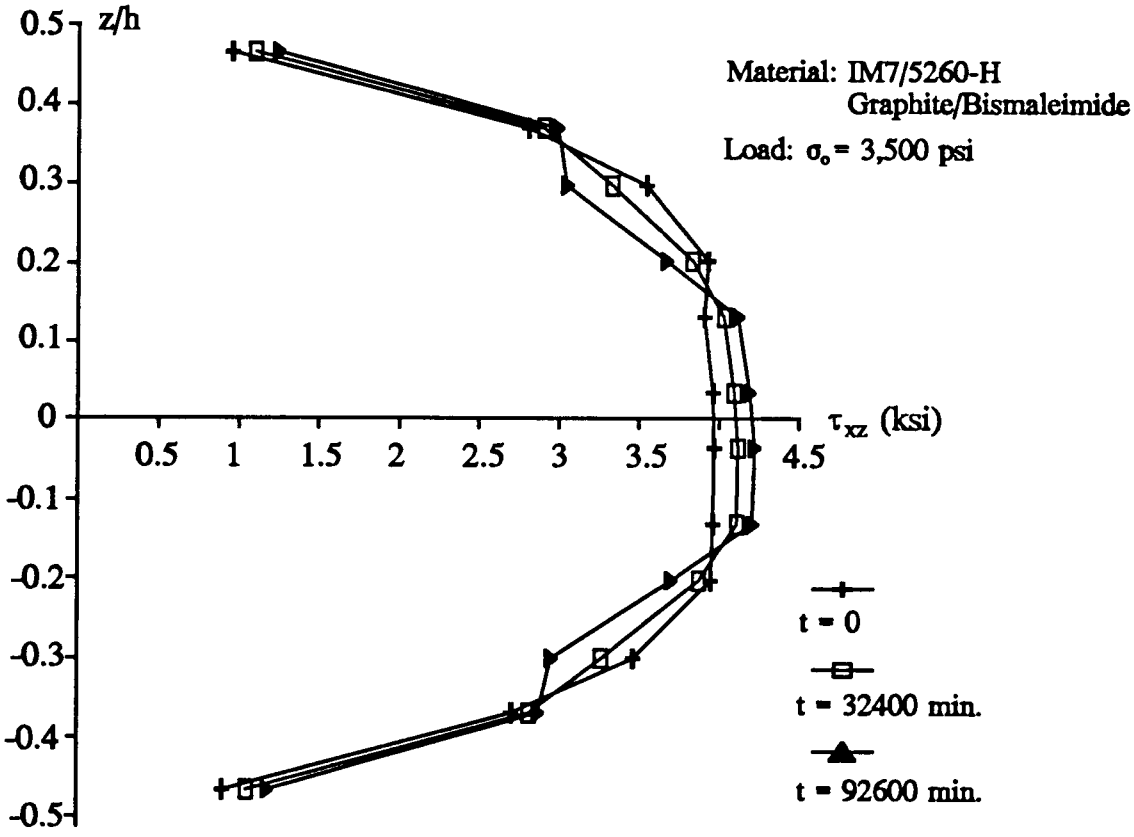


Figure 5.20 Interlaminar shear stress versus position for a thick, [0/90/0] laminated plate under bending (viscoelastic response).

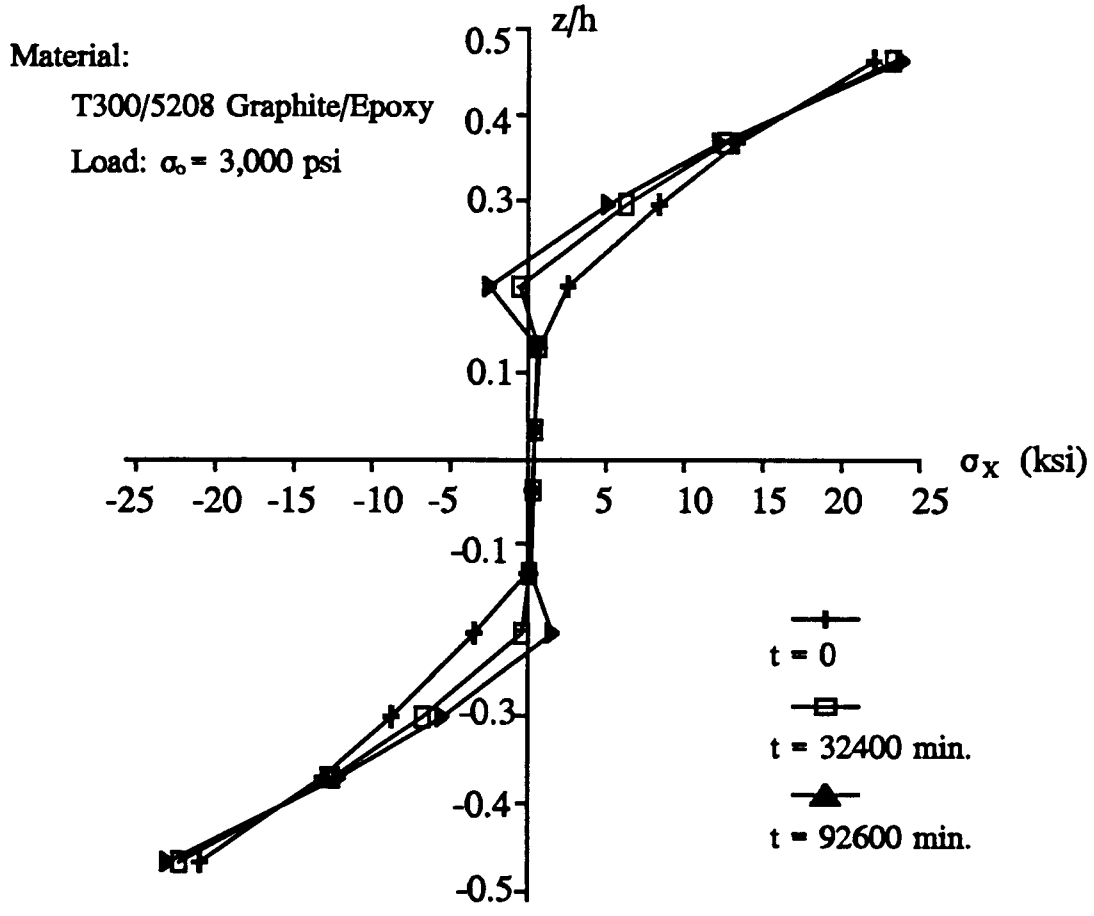


Figure 5.21 In-plane normal stress versus position for a thick, [0/90/0] laminate of T300/5208 under bending (viscoelastic response).

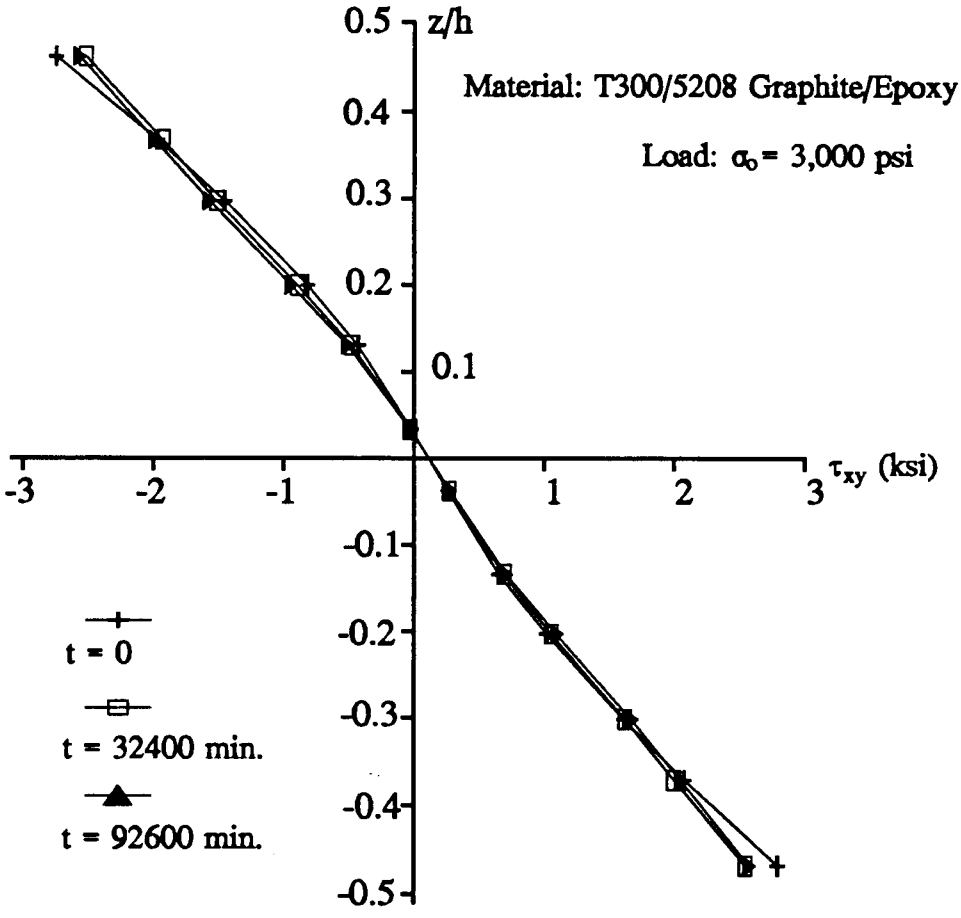


Figure 5.22 In-plane shear stress versus position for a thick, [0/90/0] laminated plate of T300/5208 under bending (viscoelastic response).

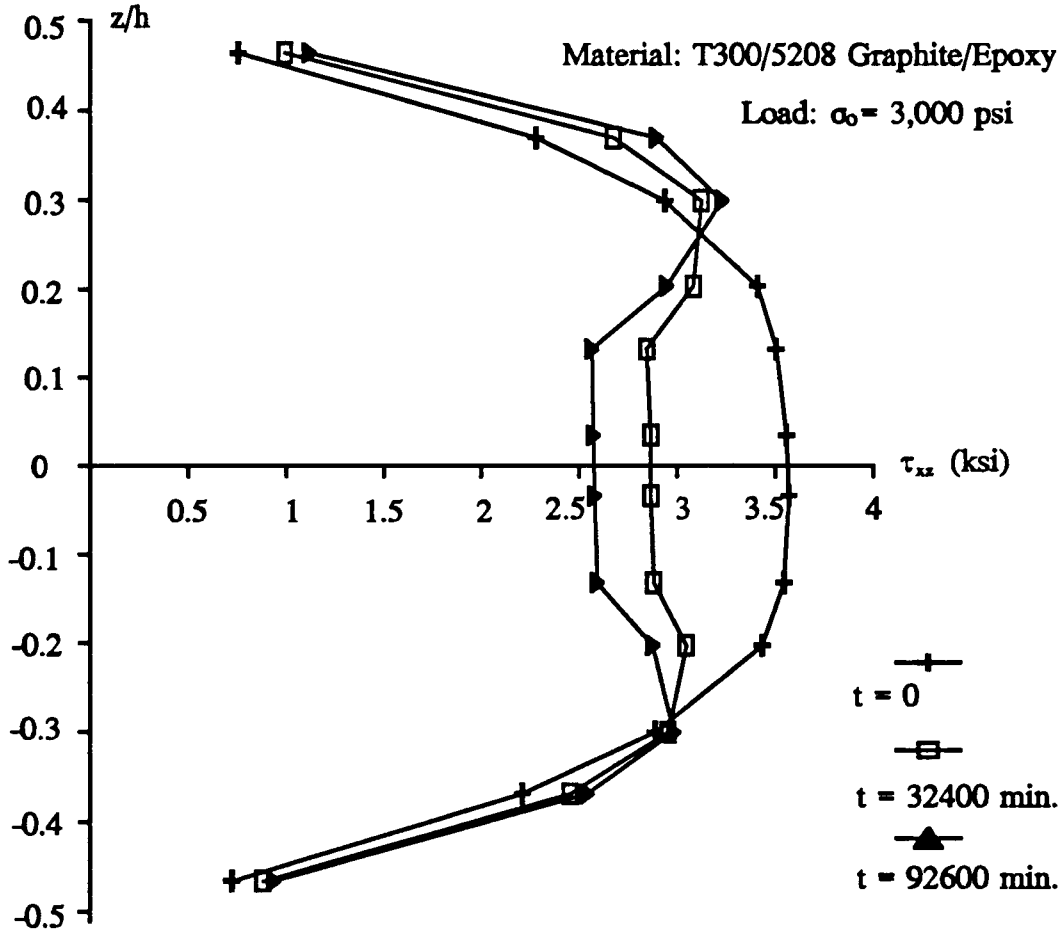


Figure 5.23 Interlaminar shear stress versus position for a thick, [0/90/0] laminated plate of T300/5208 under bending (viscoelastic response).

5.3.2 Discussion

The agreement with the analytical solution verified the adequacy of the finite element mesh.

For the nonlinear viscoelastic analysis, stresses redistribute themselves over a period of approximately two months. For IM7/5260-H, the peak value of σ_x increases 30 percent, the peak value of τ_{xy} decreases 70%, and the peak value of τ_{xz} increases 6.5%. For T300/5208, the peak value of σ_x increases by 8 percent, the peak value of τ_{xy} decreases 6.6 percent, and the peak value of τ_{xz} increases 10 percent in 0-degree layer and decreases 27.8 percent in the 90-degree layer.

It is interesting to notice that for the two different materials, the transverse shear stresses τ_{xz} redistributed themselves differently. For IM7/5260-H, τ_{xz} is a maximum in the outer layers first, then later it becomes a maximum in the middle layer. Whereas for T300/5208, τ_{xz} is a maximum in the middle layer first then the outer layer.

5.4 Tensile Loading of Notched Laminates

5.4.1 Description of the Problem

The final application considered here is the analysis of a four layer rectangular laminated plate with a circular hole at its center. The plate is subjected to a uniaxial tensile load as shown in Figure 5.24. An elastic analysis was performed first to verify the accuracy of the finite element program LAMCREP. A [90/0/0/90] laminate is chosen to compare with the results given by Nishioka and Atluri (Nishioka, T. and Atluri, S.N., 1982) and Chen and Huang (Chen, W.H. and Huang, T.F., 1989). The material used in the verification case is boron/epoxy laminate whose material properties are: $E_1 = 30 \times 10^6$ psi; $E_2 = E_3 = 3 \times 10^6$ psi; $G_{12} = G_{13} = G_{23} = 10^6$ psi; $\nu_{12} = \nu_{13} = \nu_{23} = 0.336$. The geometries are shown in Figure 5.24, where a denotes the width of the square plate, R the radius of the circular hole, h_o the thickness of each

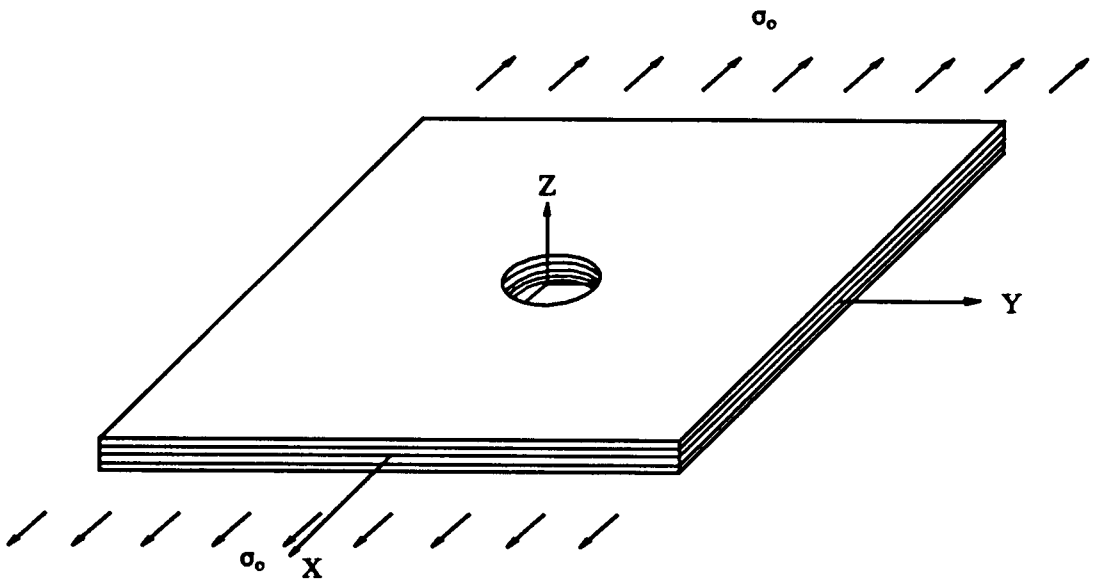


Figure 5.24 A plate with a centered hole under tension.

lamina and h the total thickness of the laminate plate, i.e., $h = 4h_o$. In the finite element analysis, only one-eighth of the plate was meshed due to symmetry. The applied load in this verification case is 1 psi. The finite element mesh is shown in Figure 5.25. Figures 5.26-5.27 are the comparisons of the elastic results from LAMCREP and the results published by others (Nishioka, T. and Alturi, S.N., 1982; Chen, W.H. and Huang, T.F., 1989). In Figure 5.26 and Figure 5.27, the circumferential stresses σ_θ at the edge of the hole in the 0-degree ply and 90-degree ply are plotted as functions of the angular position $\theta = \tan^{-1} y/x$. Again, the agreement between these results verifies the adequacy of the finite element mesh used for this analysis.

A viscoelastic analysis was conducted next. A tensile stress of 6,000 psi was applied to both materials for a time duration from t=0 to t=100,000 minute. The circumferential stresses σ_θ versus angle position θ are plotted for the 0-degree ply and the 90-degree ply at three different time steps. In addition, the interlaminar stress σ_z in the 0-degree ply was also examined by plotting it as a function of radial position at various time intervals along the line $\theta = 87.3$ (this line passes through the element integration points nearest the y-axis). Figures 5.28 - 5.30 are these results for IM7/5260-H and Figures 5.31-5.33 for T300/5208.

5.4.2 Discussion

Both elastic and viscoelastic analysis show that for a [90/0/0/90] laminate with a centered hole, there is no stress concentration observed in the 90-degree layers, i.e., the ratio of $\sigma_o/\sigma_\theta < 1$ (Figure 5.27, 5.28 and 5.31). The stress concentrations mainly occurred in the 0-degree layers. For the plate with the geometry chosen here (2R/a = 1/9), the peak values of σ_o/σ_θ are as high as 10.5 for Boron Epoxy (Figure 5.26), 15.4 for IM7/5260-H at t = 0 (Figure 5.29), and 8.4 for T300/5208 at t = 0 (Figure 5.31). As expected, the interlaminar stresses increase rapidly as they approach the edge of the hole (Figure 5.33, Figure 5.34).

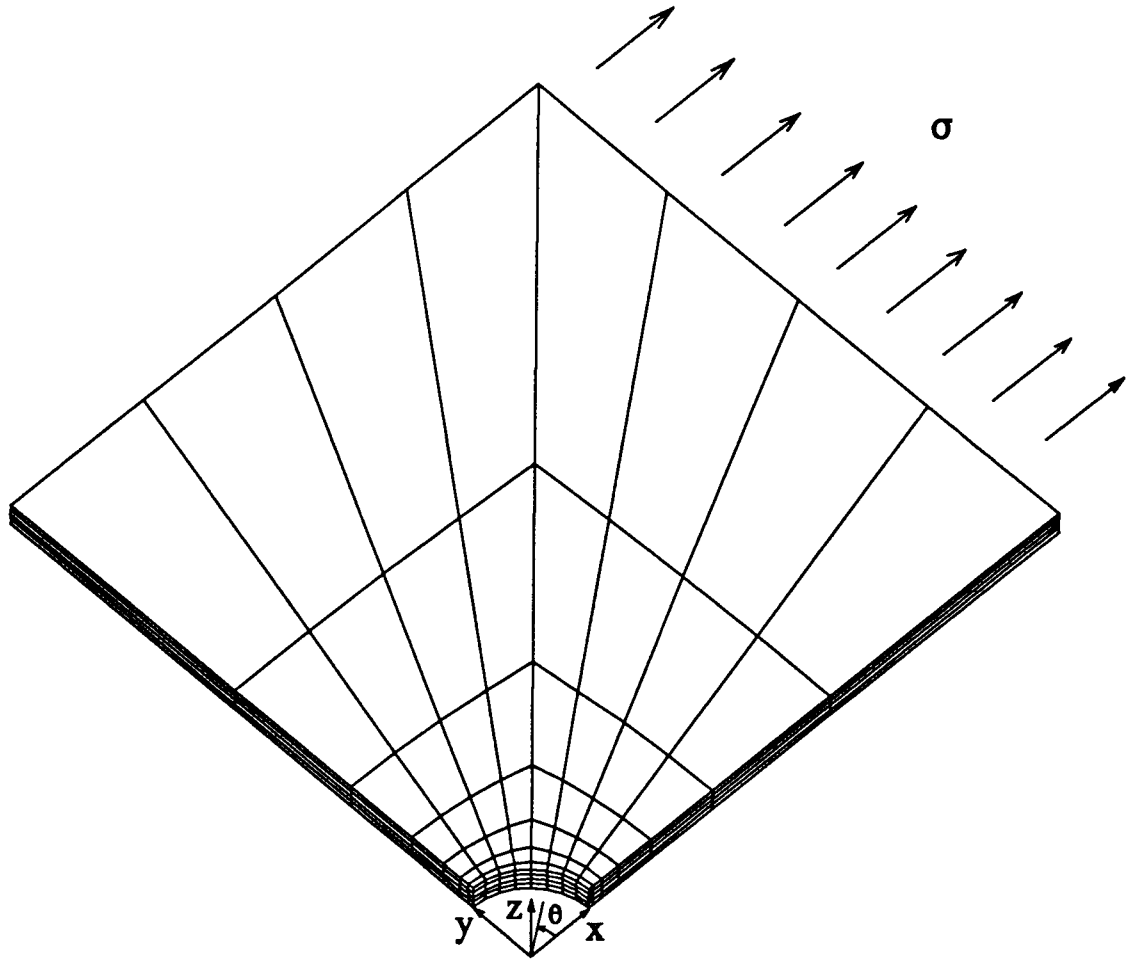


Figure 5.25 Finite element mesh for a laminated ($[90/0]_s$) plate with a circular hole under tension.

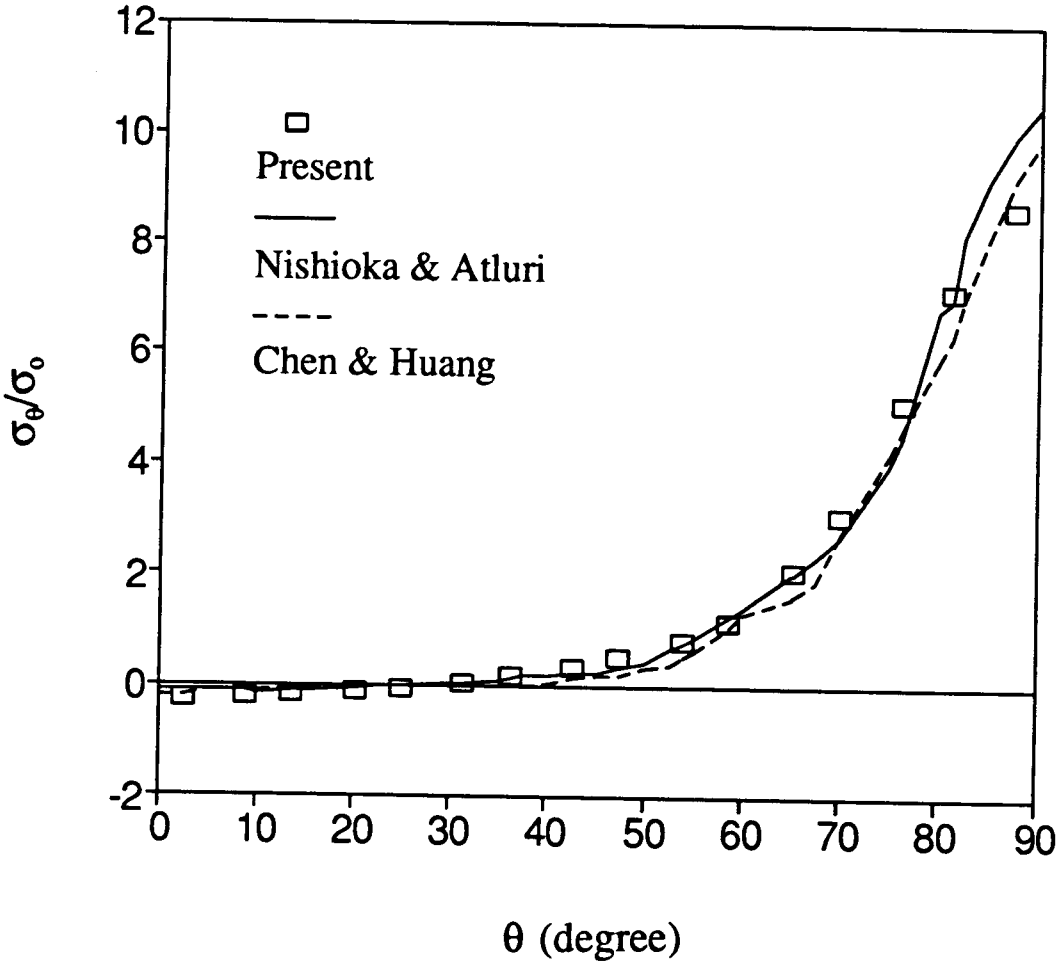


Figure 5.26 Normalized circumferential stress versus angular position in the 0-degree ply for elastic response.

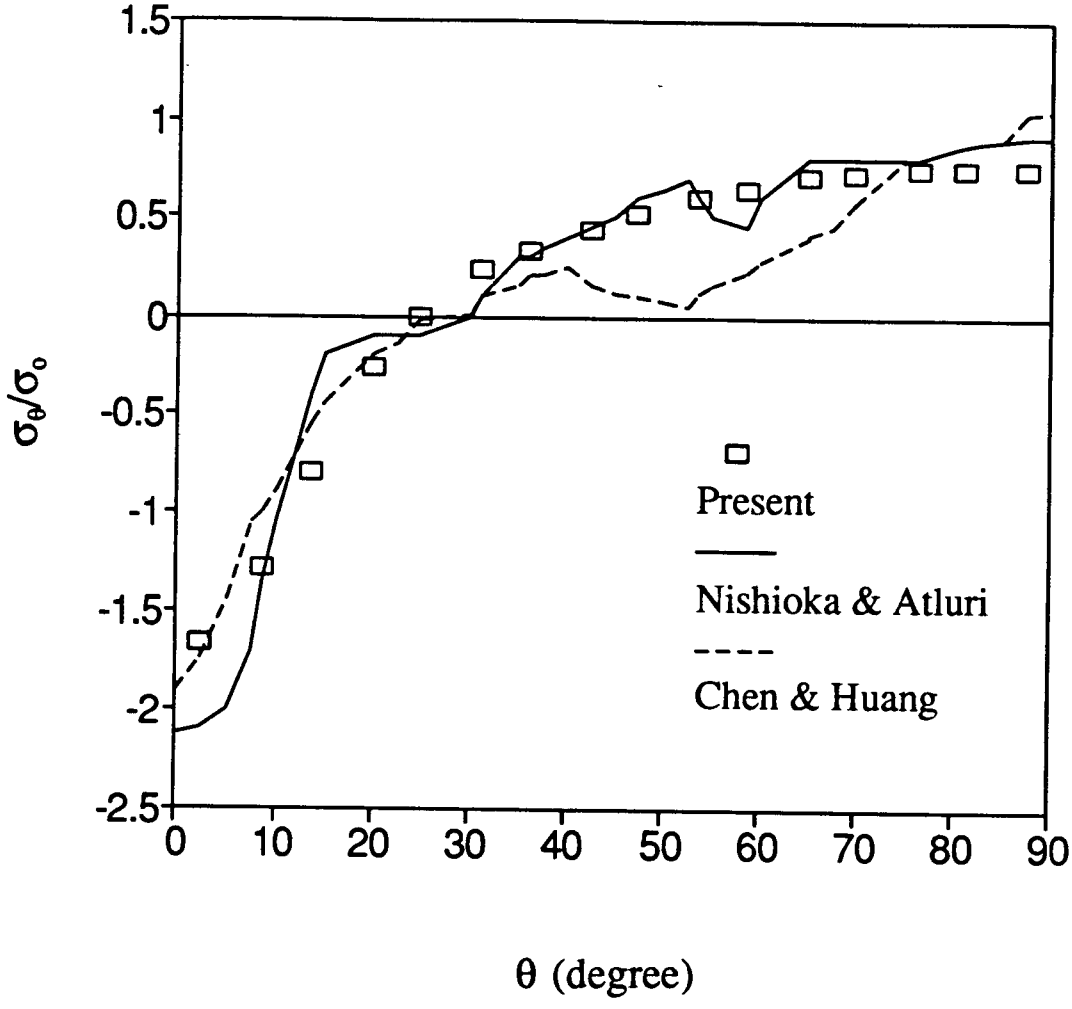


Figure 5.27 Normalized circumferential stress versus angular position in the 90-degree ply for elastic response.

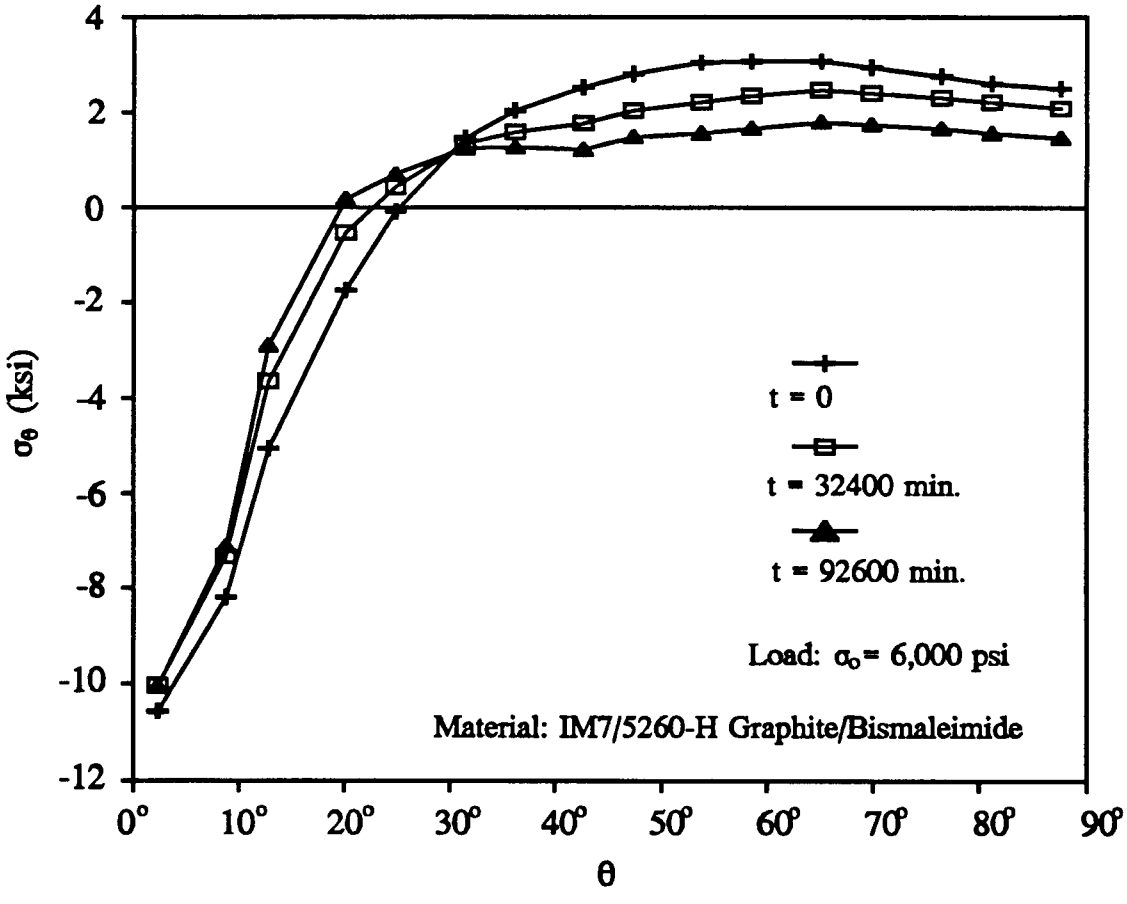


Figure 5.28 Circumferential stress versus angular position in the 90-degree ply for viscoelastic response (IM7/5260-H).

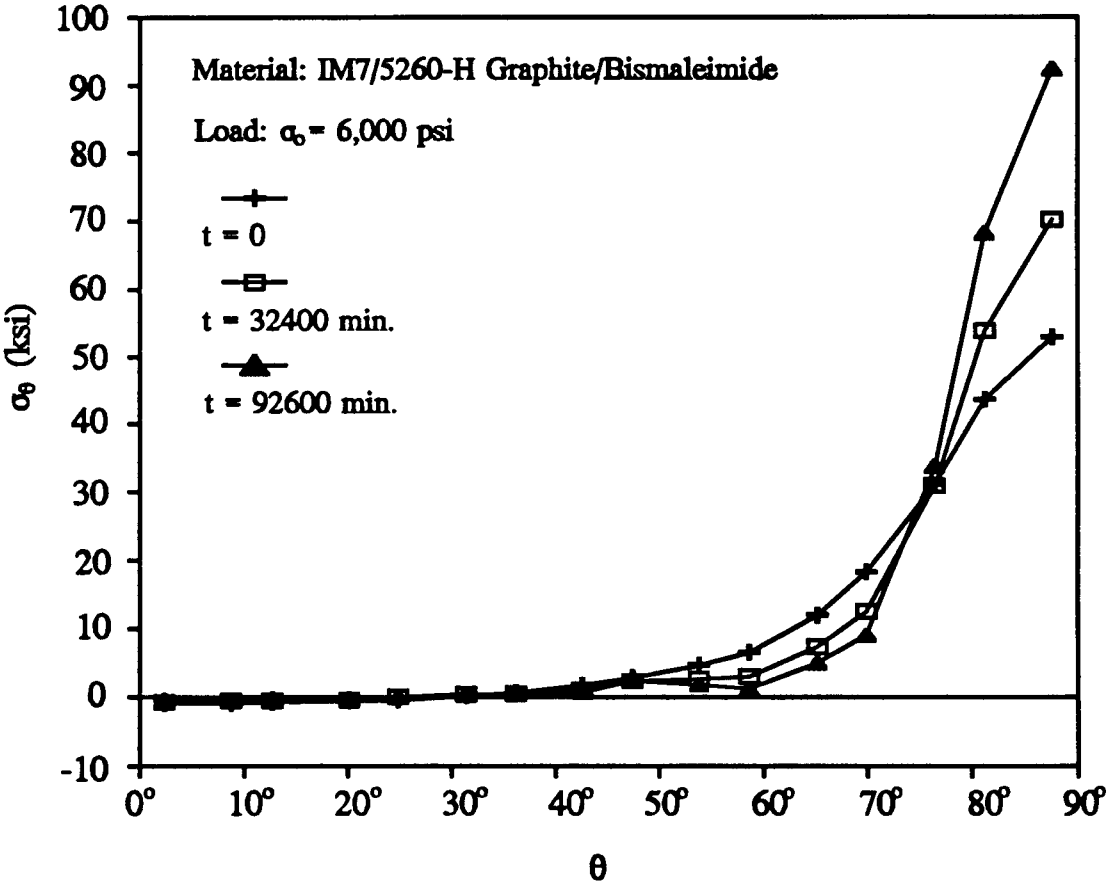


Figure 5.29 Circumferential stress versus angular position in the 0-degree ply for viscoelastic response (IM7/5260-H).

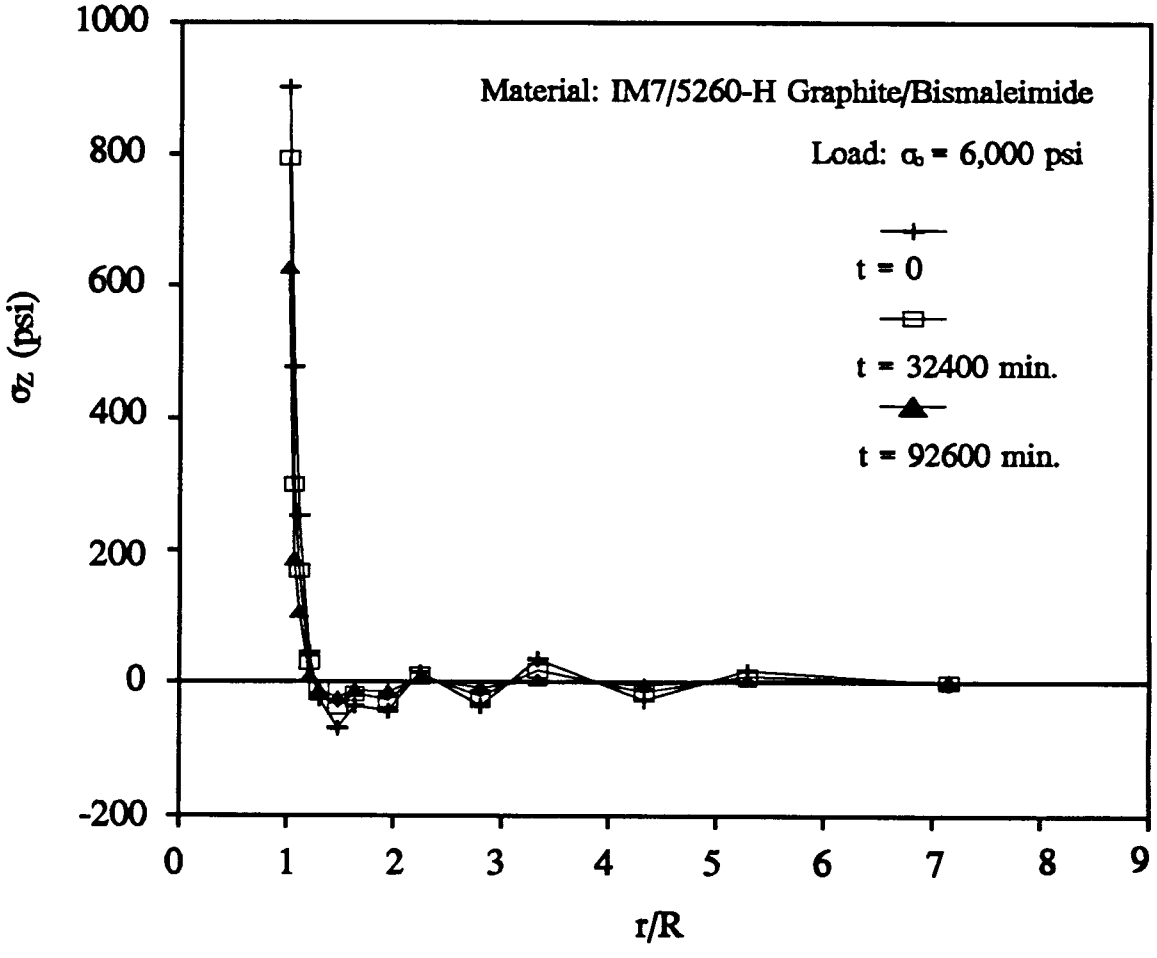


Figure 5.30 Interlaminar normal stress versus radial position in the 0-degree ply for viscoelastic response (IM7/5260-H).

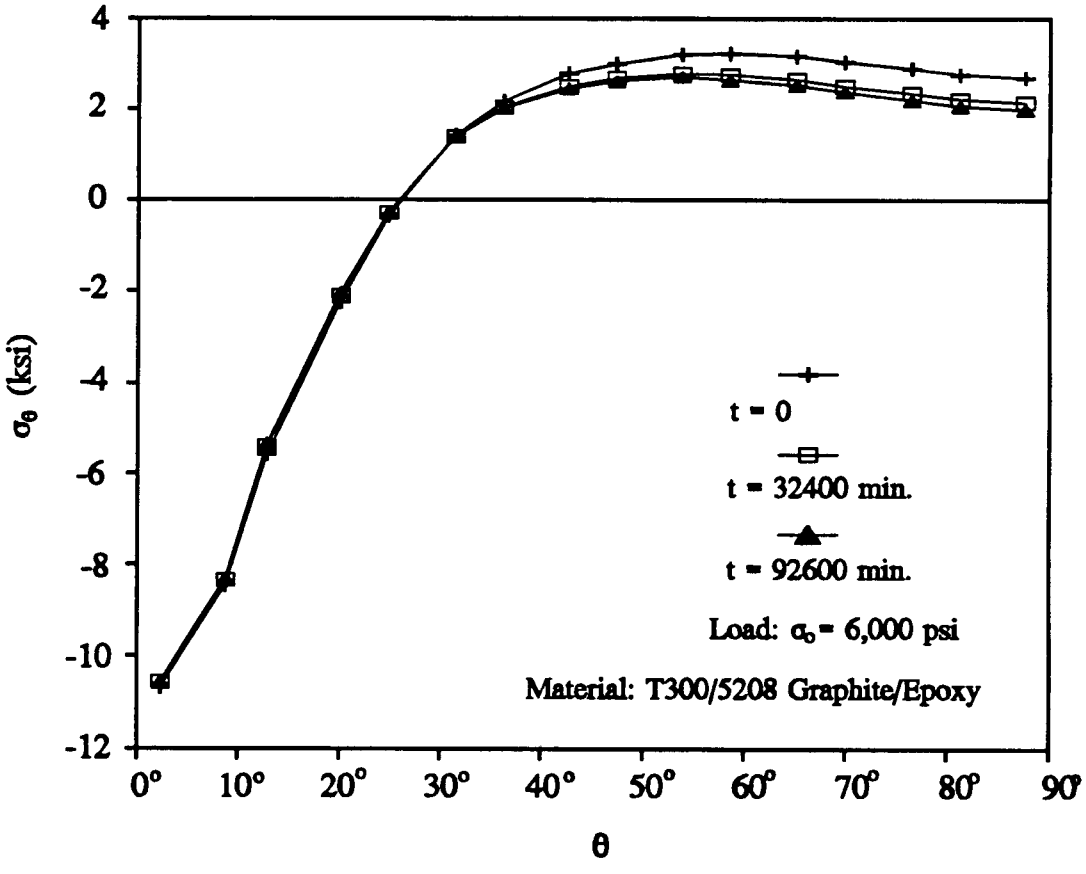


Figure 5.31 Circumferential stress versus angular position in the 90-degree ply for viscoelastic response (T300/5208).

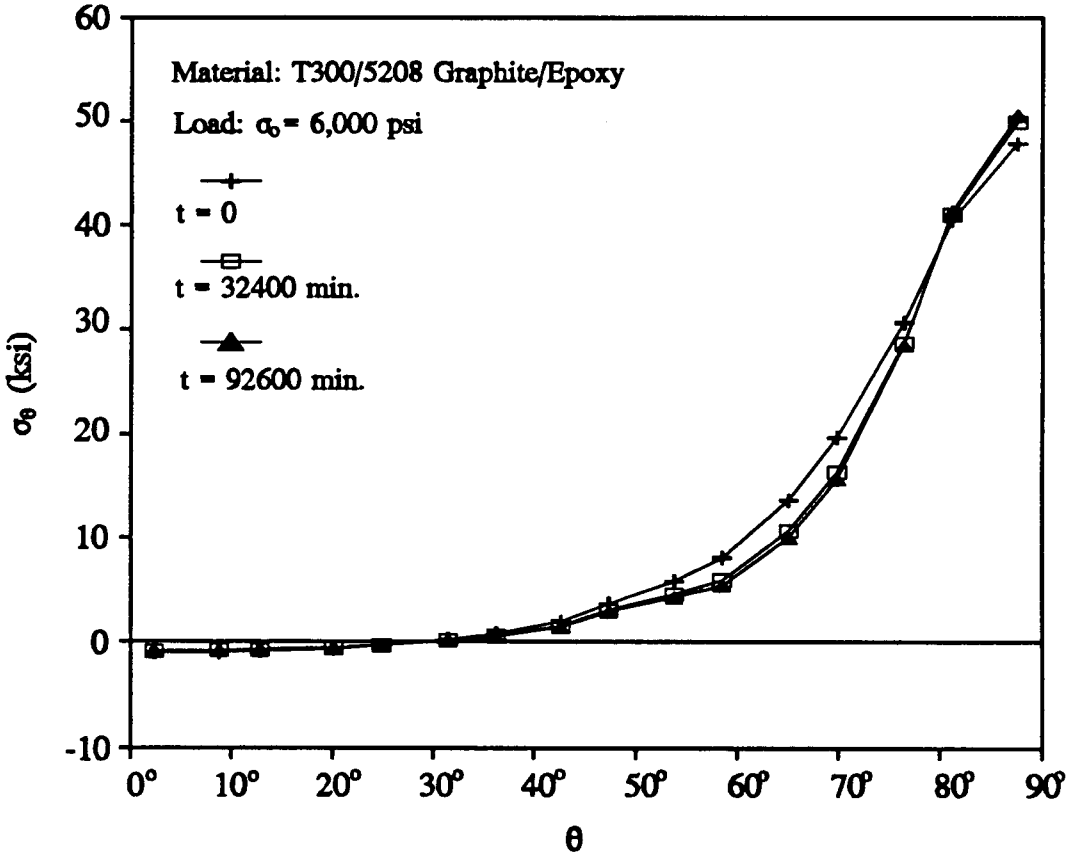


Figure 5.32 Circumferential stress versus angular position in the 0-degree ply for viscoelastic response (T300/5208).

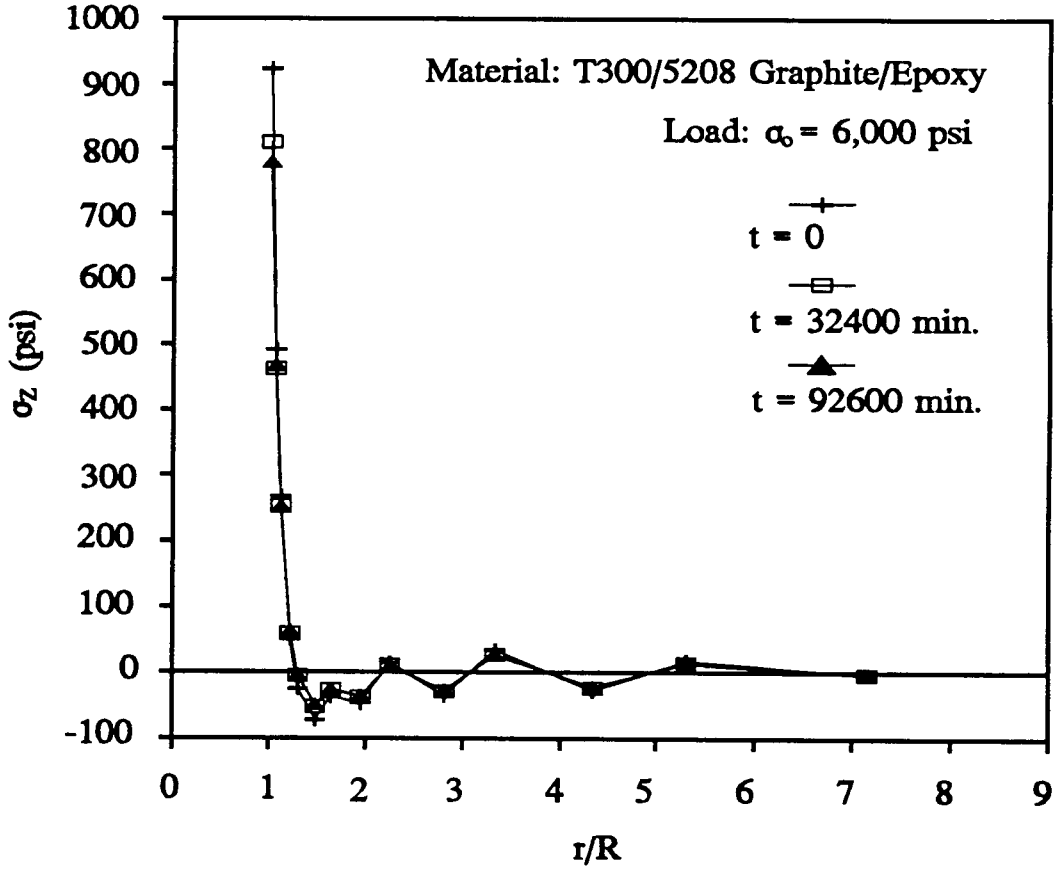


Figure 5.33 Interlaminar normal stress versus radial position in the 0-degree ply for viscoelastic response (T300/5208).

It is interesting to see that the stresses around the surface of the hole have redistributed themselves. For IM7/5260-H, the peak value of σ_θ increases 74.5 percent in the 0-degree layer, decreases 41 percent in the 90-degree layer, and the peak value of interlaminar stress σ_z decreases 30 percent over a period of approximately two months. For T300/5208, the peak value of σ_θ increases 5.7 percent in the 0-layer, decreases 26.3 percent in the 90-degree layer, and the interlaminar stress σ_z decreases 15.6 percent over a period of approximately two months. Thus, based the results presented above, it can be concluded that the load is gradually transferred from the plies with fibers transverse to the load to plies with fibers parallel to the load.

5.4.3 Failure Analysis

5.4.3.1 Description of the Problem

The failure process for the notched plate under tension was examined by using the delayed failure model developed in Section 2.3. Before analyzing the problem, the case of a $[45/-45]_s$ unnotched laminate under tension was tested to check the failure calculation of the program. In the test example, a 7,000 psi tensile load was applied on a $[45/-45]_s$ laminate (T300/5208) for a duration of 500 minutes. The comparison between LAMCREP's result and the result of classic laminate theory is shown in Figure 5.34. Both results show that all plies have failed about 70 minutes after the load was applied.

The delayed failure in the notched plate was a very slow process. First, a constant load of σ_o was applied on a $[45/-45]_s$ lamiated plate (T300/5208) with a centered hole for a period of 60,000 minutes (about 42 days). The result show that the failure initiated at the hole surface at the point where maximum stress occurred, then propagated in the radial and tangential directions near the edge of the hole (darkende area in Figure 5.35). After 60,000 minutes, only a small region of 0.5" in the radial direction normal to the loading direction and $\pm 10^\circ$ in the tangential direction had failed. However, the result reveals nothing more than failure initiation. To better

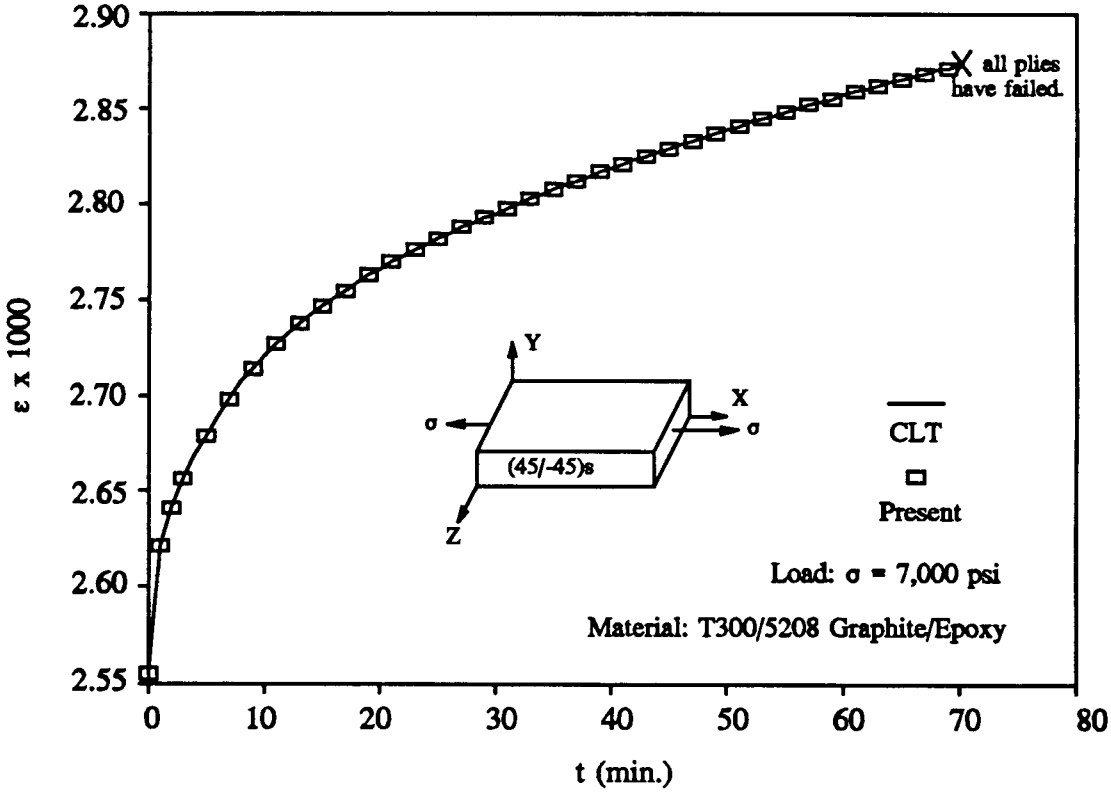
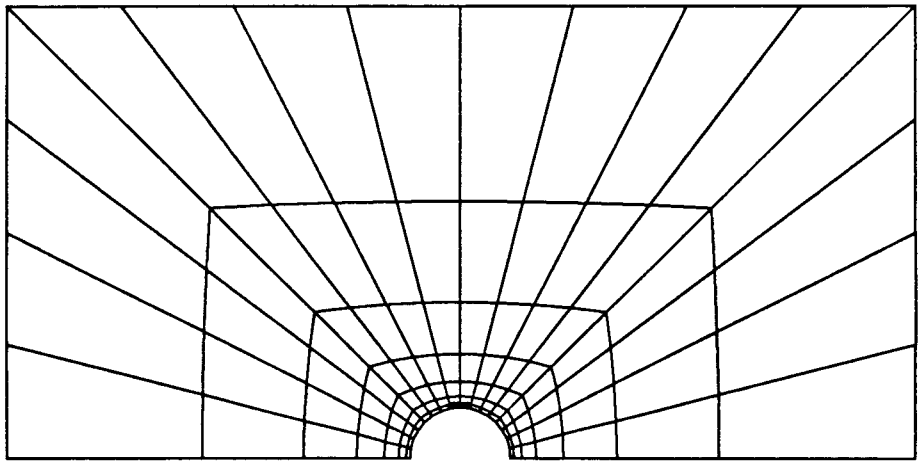
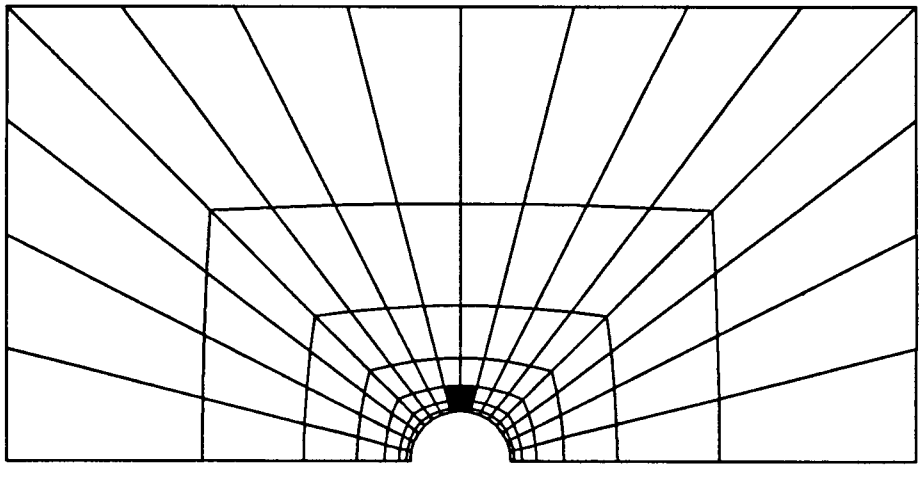


Figure 5.34 Delayed failure of a $[45/-45]_s$ laminate under tension.

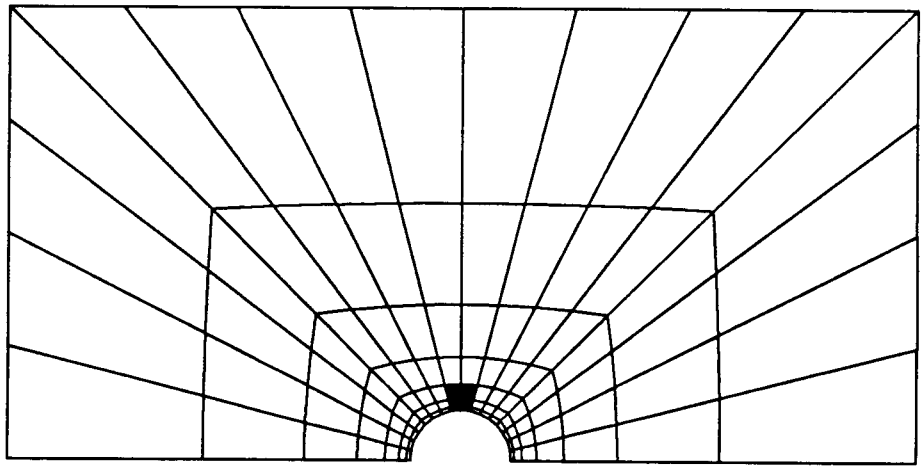


t = 2 min.

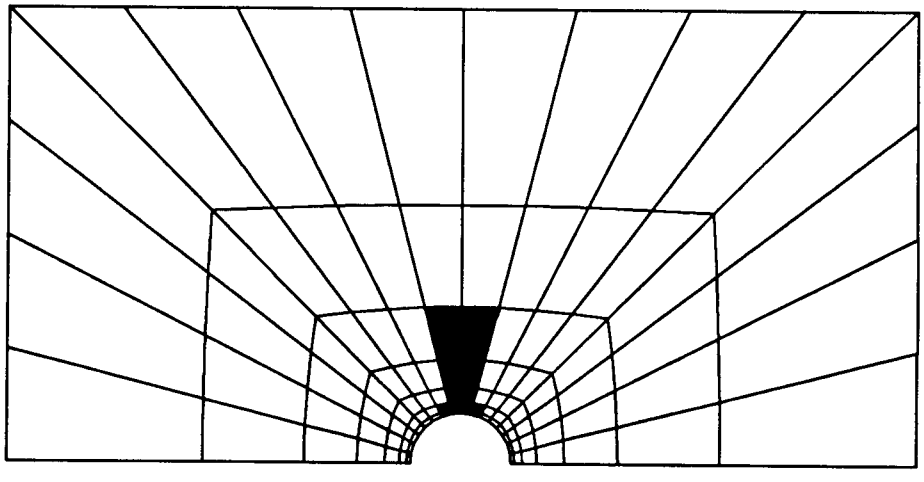


t = 6,000 min.

Figure 5.35 The predicted damage initiation in a $[45/-45]_s$ laminate with a centered hole under a constant tensile load.

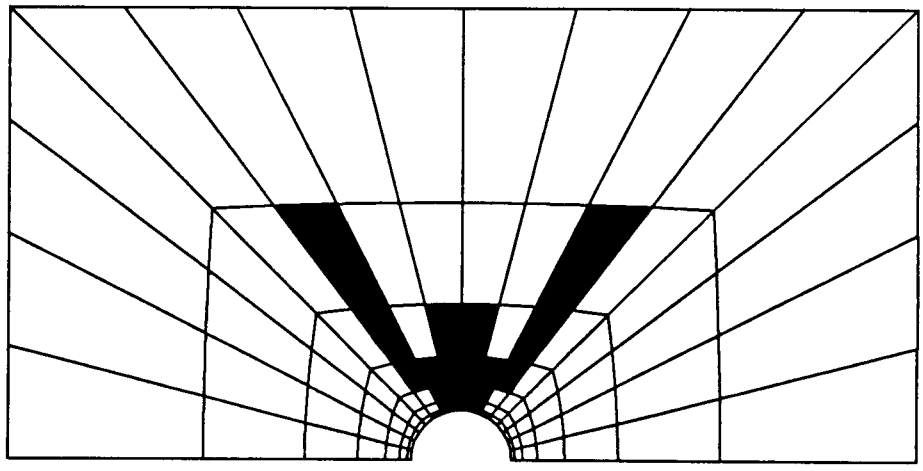


t = 185 min.
$\sigma = 3\sigma_o$

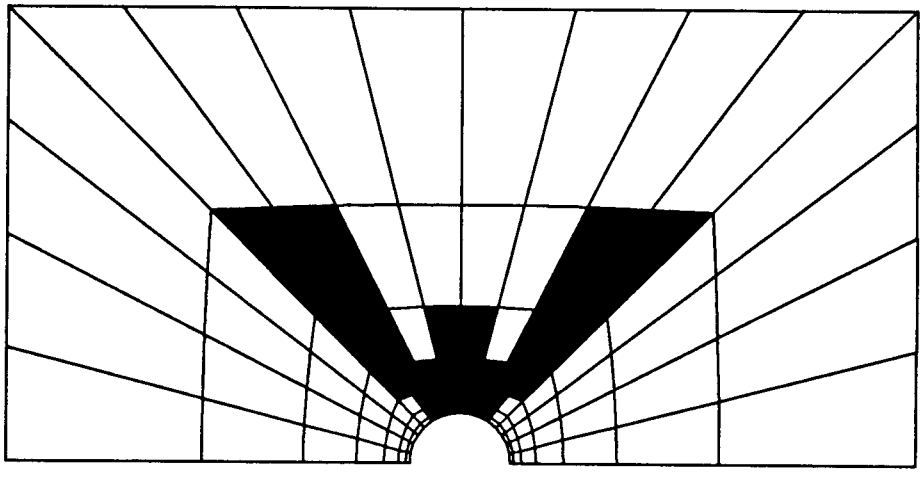


t = 2,455 min.
$\sigma = 4.2\sigma_o$

Figure 5.36 The predicted damage growth pattern in a $[45/-45]_s$ laminate with a centered hole under an increasing tensile load.



t = 2,455 min.
$\sigma = 4.2\sigma_o$



t = 3,000 min.
$\sigma = 4.2\sigma_o$

Figure 5.37 The predicted damage growth pattern in a $[45/-45]_s$ laminate with a centered hole under an increasing tensile load.

observe the failure propagation pattern of this problem, the same analysis was repeated but with an increasing load. During the time iterations, the applied load was increased by an increament of 20% of its original value between time steps, until some new elements had been predicted to have failed. Figures 5.36-5.37 plot the damage propagation profile at four different time steps.

5.4.3.2 Discussion

This application was chosen to study tensile failure of a laminated plate with a centered hole. Both Figure 5.35 and Figure 5.36 show that failure initiated from the edge of the hole where stress concentration occurred, then propagated in the radial and tangential directions. In the final stage of the failure propagation, Figure 5.37 shows that for a $[45/-45]_s$ laminate, the damage propagated mainly by approximately 45 degrees from the direction normal to the loading direction. Similar results were obtained by Chang, Lin and Chang (Chang K., Liu, S. and Chang, F., 1990) in their numerical predictions and X-radiographs. Their X-radiograph shows that the failure modes were dominated by matrix cracking and fiber-matrix shearing. The material failed finally by tearing along the fiber direction.

CHAPTER 6

CONCLUSIONS AND SUGGESTIONS FOR FUTURE WORK

CONCLUSIONS

The goals of this research can be summarized as follows: establishing three dimensional, nonlinear viscoelastic constitutive equations for orthotropic composites, developing a finite element computer program to perform the calculations, and revealing the time-dependent behavior of laminated composite structures by analyzing distributions of stress and deformation developed in these structures under different loads.

The three-dimensional nonlinear viscoelastic constitutive equations were developed based on Lou-Schapery's one dimensional model. The transient creep compliance in the constitutive equation was represented by an exponential series plus a linear term. This choice resulted in the development of a recursive relation that greatly reduces the computational effort in accounting for hereditary effects.

A three-dimensional finite element program based on the above analysis was developed using 20-node, isoparametric solid elements to model individual plies in the laminate. The correctness of the program was verified by performing several analyses for which published data are available for comparison. Based on the verification results, time-dependent behavior of selected composites was analyzed. Results indicate that the finite element program developed in this research provides very good results in both elastic analysis and viscoelastic analysis of orthotropic composites.

The time-dependent behavior of IM7/5260-H and T300/5208 Graphite/Epoxy was chosen to be analyzed because they are of interest for use in aircraft designs. The information obtained concerning characteristics of time-dependent stress/deformation states developed under long term loads, transformation of the loads between layers, and delayed failure can be summarized as follows:

1) viscoelastic materials present time-dependent deformation under constant tensile loads. The continuously increasing deformation may lead to failure of the

composite structure under a long term load which did not initially exceed the ultimate strength of the material according to an elastic analysis.

2) the strength of the material is greatly improved by the presence of the fibers. However, results have shown that different lay-ups of laminates behave fundamentally different. Thus, great care must be taken to make use of the full advantages of composites and to avoid unfavorable results.

3) the stress redistributions over time are different from most common materials, and are often very complicated and difficult to predict. The redistributions do not always lead to a higher stress value after a period of time. This indicates that the redistribution may change the failure mode of the composite structure due to the growth of the stress in one direction and reduction of the stress in other directions.

This analysis was found to be capable of accurately modeling complex, three-dimensional, time-dependent stress states.

SUGGESTIONS FOR FUTURE WORK

The analyses done so far are under a constant temperature (200°F for IM7/5260-H, 300°F for T300/5207) and a constant applied load. The long term durability of composite structures under cyclic load and temperature is not considered here, but this will surely have an influence on the final results. This consideration will extend the application of the program to the study of creep-fatigue life prediction. A micromechanics study of delamination is possible by including nonlinear link elements in the program to model bonding between layers. The program also needs improvements to be user friendly. Preparation of input data files is a difficult task, especially for a model with a large number of nodes and elements, and with complex loads. Another idea worth trying is to combine the capabilities of this program with a commercial FEA program for the analysis of a large scale problem, for example, an airplane wing, by utilizing substructures.

BIBLIOGRAPHY

Aklonis, J.J., Nacknight, W.J. and Shen, M. (1972). Introduction to Polymer Viscoelasticity, Wiley-Interscience. New York.

Baer, E., (1990). High Performance Polymers, Hanser Publishers, Munich Vienna New York.

Barker, R.M., Lin, F.T. and Dana, J.R., (1972). Three-dimensional Finite-element Analysis of Laminated Composites. Computers & Structures, 2:1013-1029.

Brinson, H.F., Griffith, W.I. and Morris, D.H., (1981). Creep Rupture of Polymer-matrix Composites. Experimental Mechanics, September 329-335.

Chang, K.Y. Liu, S. and Chang F.K., (1989). Damage Tolerance of Laminated Composites Containing an Open Hole and Subjected to Tensile Loadings. Journal of Composite Materials, 25:274-301.

Chen, W.H. and Huang, T.F., (1989). Three Dimensional Interlaminare Stress Analysis at Free Edges of Composite Laminate. Computers & Structures, 32:1275-1286.

Chulya, A. and Walker, K.P., (1991). A New Uniformly Valid Asymptotic Integration Algorithm for Elasto-Plastic Creep and Unified Viscoplastic Theories Including Continuum Damage. International Journal for Numerical Methods in Engineering, 32:385-418.

Cook, R.D., (1981). Concepts and Applications of Finite Element Analysis, John Wiley & Sons, New York.

Dillard, D.A., Straight, M.R. and Brinson, H.F., (1987). The Nonlinear Viscoelastic Characterization of Graphite/Epoxy Composites. Polymer Engineering and Science, 27:116-123.

Dillard, D.A. and Brinson, H.F., (1983). A Numerical Procedure for Predicting Creep and Delayed Failures in Laminated Composites. Long-Term Behavior of Composites, 23-27.

Dillard, D.A. Morris, D.H. and Brinson, H.F., (1982). Predicting Viscoelastic Response and Delayed Failures in General Laminated Composites. Composite Materials: Testing and Design (Sixth Conference), ASTM STP 787, I.M. Daniel, Ed., American Society for Testing and Materials, 357-370.

Douven, L.F., Douven, P.J.G. and Janssen, J.D., (1989). Analysis of Viscoelastic Behavior of Transversely Isotropic Materials. International Journal for Numerical Methods in Engineering, 28:845-860.

Feng, W.W., (1991). A Failure Criterion for Composite Materials. Journal of Composite Materials, 25:89-100.

Findley, W.N., (1975). Creep and Relaxation of Nonlinear Viscoelastic Materials, North-Holland Series in Applied Mathematics and Mechanics, Vol. 18.

Flugge, W., (1975). Viscoelasticity, Springer-Verlag, New York.

Glockner, P.G. and Szyszkowski, W., (1990). An Engineering Multiaxial Constitutive Model for Nonlinear Time-dependent Materials. International Journal of Solid Structures, 26:73-82.

Green, A.E. and Rivlin, R.S., (1957). The Mechanics of nonlinear Materials with Memory, Part I. Archive for Rational Mechanics and Analysis, 1:1.

Green, A.E., Rivlin, R.S. and Spencer, A.J.M., (1959). The Mechanics of nonlinear Materials with Memory, Part II. Archive for Rational Mechanics and Analysis, 3:82.

Green, A.E., and Rivlin, R.S., (1960). The Mechanics of Nonlinear Materials with Memory, Part III. Archive for Rational Mechanics and Analysis, 4:387.

Ha, S.K. and Springer, G.S., (1989). Time Dependent Behavior of Laminated Composites at Elevated Temperatures. Journal of Composite Materials, 23:1159-1197.

Henriksen, M., (1984). Nonlinear Viscoelastic Stress Analysis-A Finite Element Approach. Computers & Structures, 18:133-139.

Hinton, E. and Owen, D.R.J., (1989). Finite Element Programming, Academic Press, New York.

Hollaway, L.C., (1990). Polymers and Polymer Composites in Construction, Thomas Telford Ltd, London.

Horoschenkoff, A., (1990). Characterization of the Creep Compliances J_{22} and J_{66} of Orthotropic Composites with PEEK and Epoxy Matrices Using the Nonlinear Viscoelastic Response of the Neat Resins. Journal of Composite Materials, 24:879-891.

Jones, R.M., (1975). Mechanics of Composite Materials, Hemisphere Publishing Corporation, New York.

Kim, J.Y. and Hong, C.S., (1991). Three-dimensional Finite Element Analysis of Interlaminar Stresses in Thick Composite Laminates. Computer & Structures, 40:1395-1401.

Lee, J.D., (1980). Three Dimensional Finite Element Analysis of Layered Fiber-reinforced Composite Materials. Computers & Structures, 12:319-339.

Lin, K.Y. and Yi, S., (1991). Analysis of Interlaminar Stresses in Viscoelastic Composites. Int. J. Solids Structures, 27:929-945.

Lin, K.Y. and Hwang, I.H., (1989). Thermo-Viscoelastic Analysis of Composite Materials. Journal of Composite Materials, 23:554-569.

Lou, Y.C. and Schapery, R.A., (1971). Viscoelastic Characterization of a Nonlinear fiber-Reinforced Plastic. Journal of Composite Materials, 5:208-234.

Moore, R.H. and Dillard, D.A., (1990). Time-dependent Matrix Cracking in Cross-ply Laminates. Composites Science and Technology, 39:1-12.

Nishioka, T. and Atluri, S.N, (1982). Stress Analysis of Holes in Angle-ply Laminates: An Efficient Assumed Stress "Special-Hole-Element" Approach and A Simple Estimation Method. Computer & Structures, 15:135-147.

Noor, A.K. and Burton, W.S., (1992). Computational Models for High-temperature Multilayered Composite Plates and Shells. Appl. Mech. Rev., 45:419-445.

Odqvist, F.K.G. (1966). Mathematical Theory of Creep and Creep Rupture, Oxford at the clarendon Press.

Pagano, N. J., (1970). Exact Solutions for Rectangular Bidirectional Composites and Sandwich Plates. Journal of Composite Materials, 4:20-35.

Pipes, R.B., Kaminski, B.E. and Pagano, N.J., (1973). Influence of the Free Edge upon the Strength of Angle-Ply Laminates. Analysis of the Testing and Materials, 218-228.

Pomeroy, C.D., (1978). Creep of Engineering Materials, Mechanical Engineering Publications Limited, London.

Roy, S. and Reddy, J.N., (1988). A Finite Element Analysis of Adhesively Bonded Composite Joints with Moisture Diffusion and Delayed Failure. Computers & Structures, 29:1011-1988.

Schapery, R.A., (1966). A Theory of Nonlinear Thermoviscoelasticity Based on Irreversible Thermodynamics. Trans. Soc. Rheol., 7:391.

Schapery, R.A., (1969). On the Characterization of Nonlinear Viscoelastic Materials. Polymer Engineering and Science, 9:295-310.

Schapery, R.A., (1974). Viscoelastic Behavior and Analysis of Composite Materials. Mechanics of Composite Materials, Sendeckyj, G.P. (Ed.), New York, Academic Press, 86-167.

Shames, I.H., and Cozzarelli, F.A., (1992). Elastic and Inelastic Stress Analysis. Prentice Hall, Englewood Cliffs, New Jersey. p.205.

Srinatha, H.R. and Lewis, R.W., (1981). A Finite element method for Thermoviscoelastic Analysis of Plane Problems. Computer Methods in Applied Mechanics and Engineering, 25:21-33.

Stango, R.J., Wang, S.S. and Nelson, C.R., (1989). A Note on Analytical Representation of Anisotropic Viscoelastic Constitutive Equations for Fiber-reinforced Composites. Composites Science and Technology, 35:273-282.

Theocaris, P.S., (1990). Properties and Experimental Verification of the Parabolic failure Criterion with Transversely Isotropic Materials. Journal of Materials Science, 25:1076-1085.

Tuttle, M.E. and Brinson, H.F., (1986). Prediction of the Long-Term Creep Compliance of General Composite Laminates. Experimental Mechanics, 26:89-102.

Wang, A.S.D, and Crossman, F.W., (1977). Some New Results on Edge Effect in Symmetric Composite Laminates. Journal of Composite Materials, 11:92-106.

Wang, Y.Z. and Tsai, T.J., (1988). Static and Dynamic Analysis of a Viscoelastic Plate by the Finite Element Method. Applied Acoustics, 25:77-101.

Zienkiewicz, O.C. and Taylor, R.L., (1989). The Finite Element Method, McGraw-Hill Book Company, London.

APPENDICES

APPENDIX A

INSTRUCTIONS FOR PREPARING INPUT DATA FILE

Details about data in the input file are given in the following sections. Variable names listed here are the same as the names used in the program.

A.1 Control Information

input set	input parameters
1	nprob
2	title
3	npoin, nelem, nuvfx, ncase, nnode,ndofn, nmats, nreal, nprop, ngaus, ndime, nstre, ifail.
4	tstart, tfinal, tunld, dt, error.

where

nprob: Total number of problems to be solved in one run.

title: Title of the problems to be solved in one run.

npoin: Total number of nodal points.

nelem: Total number of elements.

nuvfx: Total number of fixed nodes.

ncase: 1.

nnode: Number of nodes per element (=20).

ndofn: Number of degrees of freedom per node (=3).

nmats: Total number of different materials.

nreal: Total number of different real constants.

nprop: Number of constants per material(=45).

ngaus: Order of integration formula for numerical integration (2 or 3).

ndime:	Number of coordinate dimensions (=3).
nstre:	Number of stress components (=6).
ifail:	Failure model control parameter.
	ifail=1, turn on the failure model,
	ifail=0, turn off the failure modle.
tstart:	Starting time.
tfinal:	Ending time.
tunld:	Time at which load will be removed.
dt:	Time increment.
error:	Convergence criterion.

A.2 Geometry data

Input set	Input parameters
1	dummyt
2	ipoin, coord(ipoin,1), coord(ipoin,2), coord(ipoin,3)
3	dummyt
4	numel, matno(numel), ireal(numel), lnods(numel,1), lnods(numel,2), ..., lnods(numel,20).

where

dummyt: A character variable, maximum length is 20.

ipoin: Nodal point number.

coord(ipoin,1), coord(ipoin,2), coord(ipoin,3):
x, y and z coordinates of node.

numel: Element number.

matno: Material property number.

ireal: Real constant number.

lnods(numel,1), ..., lnods(numel,20):
1st to last nodal connection number.

Notes:

1) The character variable 'dummyt' has been set to make the input file more readable. It can be any word or phrase containing 1 to 20 characters to explain the content of following data. For example, in set 1, dummyt = 'nodal point:', and in set 2, dummyt = 'elements:'.

2) Set 2 needs to be repeated for each node up to total of *npoin* nodes.

3) Set 4 needs to be repeated for each element up to total of *nelem* elements.

4) The nodal connection numbers in set 4 must be listed in an anticlockwise sequence, starting from any corner node. See figure 5.

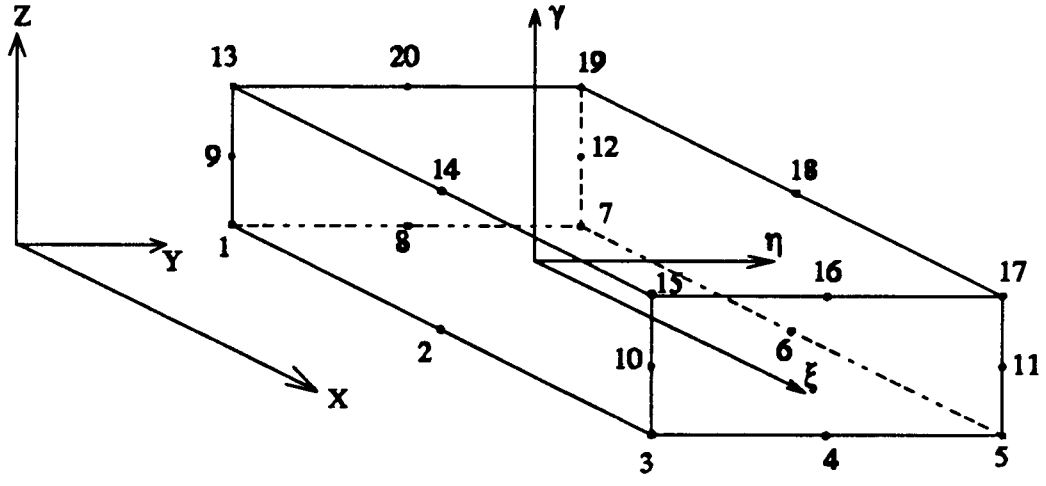


Figure A.1 A twenty-node solid element.

A.3 Constraint data

Input set	Input parameters
1	dummyt
2	nofix(ivfix), ifpre(ivfix,2),ifpre(ivfix,3), ifpre(ivfix,1), presc(ivfix,1), presc(ivfix,2), presc(ivfix,3).

where

dummyt: A character variable, maximum length is 20.

nofix: Restrained node number.

ifpre(ivfix,1), ifpre(ivfix,2), ifpre(ivfix,3):
Condition of restraint on x, y and z displacement,
0 = Unconstrained. 1 = nodal displacement restrained.

presc(ivfix,1), presc(ivfix,2), presc(ivfix,3):
The prescribed value of the x, y and z component of nodal displacement.

Example: Node 5 has been fixed in y and z direction, and it is free in x direction. In z-direction, it has a prescribed value uz = 0.005. This condition can be input as:

5,0,1,1,0,0,0.005

Note:

Set 2 need to be repeated *nufix* times.

A.4 Real constant and material property data

Input set	Input parameters
1	dummyt
2	nureal,realc(nureal)
3	dummyt
4	numat, props(numat,1), props(numat,2), ..., props(numat,8).
5	numat, props(numat,9), props(numat,10).
6	numat, props(numat,11),...., props(numat,14).
7	numat, props(numat,15),...., props(numat,18).
8	numat, props(numat,19),...., props(numat,22).
9	numat, props(numat,23),...., props(numat,26).
10	numat, props(numat,27),...., props(numat,30).
11	numat, props(numat,31),...., props(numat,33).
12	numat, props(numat,34),...., props(numat,36).
13	numat, props(numat,37),...., props(numat,41).
14	numat, lamtt(numat,ir),lamts(numat,ir).
15	numat, dtr(numat,ir),dts(numat,ir).

where

dummyt: A character variable, maximum length is 20.

nureal: Real constant identification number.

realc: The angle of the fiber specified in global coordinates x, y, and z. See figure 6.

numat: Material identification number.

prop(numat,1): Young's modulus E1 (fiber direction).

prop(numat,2): Young's modulus E2 (transverse direction).

prop(numat,3): Shear modulus G12.

prop(numat,4): Poisson's ratio v12.

prop(numat,5): Poisson's ratio v23.

prop(numat,6): Em, Young's modulus for the matrix material.

prop(numat,7): vm, Poisson's ratio for the matrix material.

prop(numat,8): itype, type of element. 3: linear elastic element. 4: element with creep nonlinearities.

prop(numat,9): Dft.

prop(numat,10): Dts.

prop(numat,11): b0t.

prop(numat,12): b1t.

prop(numat,13): b2t.

prop(numat,14): bt.

prop(numat,15): b0s.

prop(numat,16): b1s.

prop(numat,17): b2s.

prop(numat,18): bs.

prop(numat,19): k0t.

prop(numat,20): k1t.

prop(numat,21): k2t.

prop(numat,22): kt.

prop(numat,23): k0s.

prop(numat,24): k1s.

prop(numat,25): k2s.

prop(numat,26): ks.

prop(numat,27): g023.

prop(numat,28)-prop(numat,30): Thermal coefficients, alf(1), alf(2), alf(3).

prop(numat,31): Current temperature, temp.

prop(numat,32): Reference temperature, tempo.

prop(numat,33): nr, an integer constant used in later property input.

prop(numat,34): d11, coefficient used to reduce value of E1 for a particular element which has failed.

prop(numat,35): d22, coefficient used to reduce value of E2 for a particular element

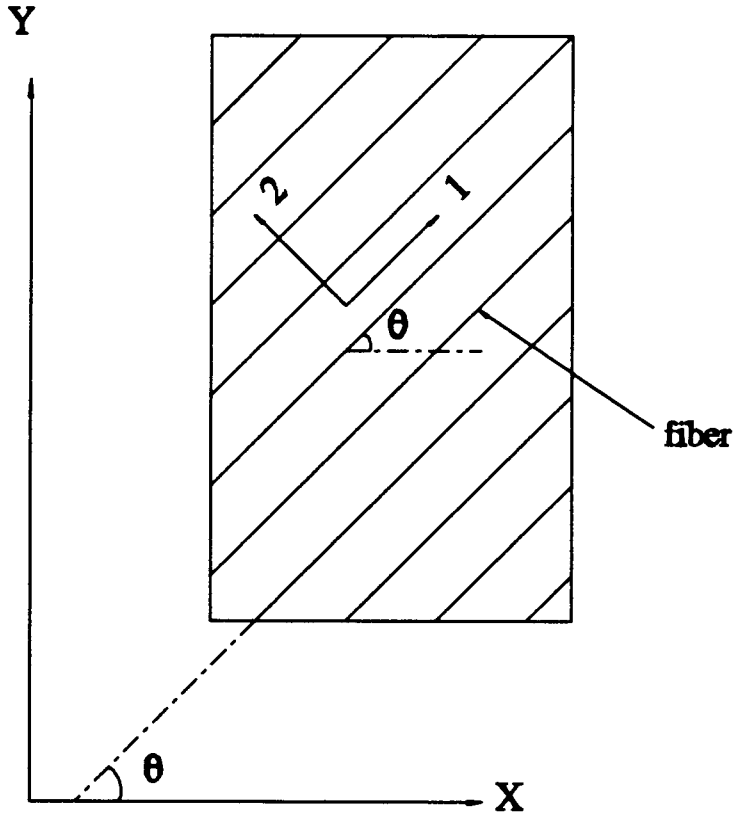


Figure A.2 The angle of fiber defined in global X,Y plane, and relative to X-direction.

which has failed.

prop(numat,36): d12, coefficient used to reduce value of G12 for a particular element which has failed.

prop(numat,37): X, ply strength in E1 direction.

prop(numat,38): a, material constant used in calculation of rupture time.

prop(numat,39): b, material constant used in calculation of rupture time.

prop(numat,40): ks12, material constant used in failure equation.

prop(numat,41): ks23, material constant used in failure equation.

lamtt, lamts: Nonlinear creep parameters in transverse and shear direction respectively.

drt, drs: Nonlinear creep parameters in transverse and shear direction respectively.

Notes:

1) To fully define one group of material properties, all 41 constants, lamtt(ir), lamts(ir), drt(ir), and drs(ir) must be input.

2) lamtt, lamts, drt and drs are array constants. The dimensions of the arrays have been input earlier in data set 37 of this part.

A.5 Load data

Input set	Input parameters
1	dummyt
2	iplod
3	lodpt, point(1), point(2), point(3)

where

dummyt: Character variable, maximum length is 20.

iplod: Applied point load control parameter, iplod=0, no applied nodal loads to be input. iplod=1, applied nodal loads to be input.

lodpt: Node number at which load has been applied.

point(1): Load component in x direction.

point(2): Load component in y direction.

point(3): Load component in z direction.

Example: 100 lb point load is applied at nodal point 1,
then:

1,100,0,0

Notes:

1) The last node number should be that for the highest numbered node whether it is loaded or not.

2) If iplod=0 in set 2, omit set 3.

A.6 Output table

Input set	Input parameters
1.	dummyt
2.	iprtrs,iprtrn,iprdis,iprforc
3.	notrs
4.	noeltrs(1),noeltrs(2),....., noeltrs(notrs).
5.	notrn
6.	noeltrn(1),noeltrn(2),....., noeltrn(notrn).
7.	nodis
8.	nodisp(1),nodisp(2),....., nodisp(nodis).

where

dummyt: A character variable, maximum length is 20.

iprtrs: Stress output control parameter.
iprtrs = 1, element stress components to be printed.
iprtrs = 0, no element stress components to be printed.

iprtrn: Strain output control parameter.
iprtrn = 1, element stress components to be printed.
iprtrn = 0, no element stress components to be printed.

iprdis: Displacement output control parameter.
iprdis = 1, nodal displacements to be printed.
iprdis = 0, no nodal displacements to be printed.

iprforc: Reaction force output control parameter.
iprforc = 1, reaction forces to be printed.
iprforc = 0, no reaction forces to be printed.

notrs: Total number of elements at which stresses are to be printed.

noeltrs: Specified element numbers at which stresses are to be printed.
If notrs = nelem, the total elements defined in current mesh, ignore data in this set.

notrn: Total number of elements at which strains are to be printed.

noeltrn: Specified element numbers at which strains are to be printed.
If notrn = nelem, the total elements defined in current mesh, ignore data in this set.

nodis: Total number of nodes at which displacements are to be printed.

nodisp: Specified nodal numbers at which displacements are to be printed.
If nodis = npoin, the total nodal points defined in current mesh, ignore data in this set.

Notes:

1) The average strains of all elements are always to be printed for the specified time steps.

2) The Strains and stresses to be printed in this part are the element strains and element stresses. Strains were calculated in global Cartesian coordinate directions. Stresses were calculated in both global Cartesian coordinates and material directions.

A.7 Sample Input File

```
    1
failure testing case, sigx=7,500 psi, [45/-45]s
56,4,40,1,20,3,1,4,12,2,3,6,1
0.0,300.0,350.,1,0.0
001
nodal points:
1,0,0,0
2,0,10,0
3,0,20,0
4,10,20,0
5,20,20,0
6,20,10,0
7,20,0,0,
8,10,0,0
9,0,0,0.0025
10,0,20,0.0025
11,20,20,0.0025
12,20,0,0.0025
13,0,0,0.005
14,0,10,0.005
15,0,20,0.005
16,10,20,0.005
17,20,20,0.005
18,20,10,0.005
19,20,0,0.005
20,10,0,0.005
21,0,0,0.0075
22,0,20,0.0075
23,20,20,0.0075
24,20,0,0.0075
25,0,0,0.01
26,0,10,0.01
27,0,20,0.01
28,10,20,0.01
29,20,20,0.01
30,20,10,0.01
31,20,0,0.01
32,10,0,0.01
33,0,0,0.0125
34,0,20,0.0125
35,20,20,0.0125
36,20,0,0.0125
```

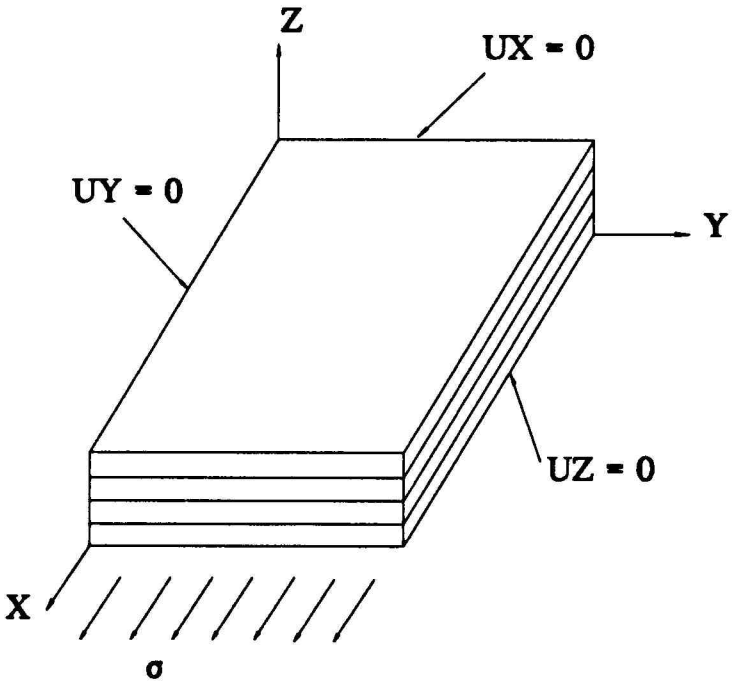


Figure A.3 One-fourth of an 8-layer laminated composite.

37,0,0,0.015
38,0,10,0.015
39,0,20,0.015
40,10,20,0.015
41,20,20,0.015
42,20,10,0.015
43,20,0,0.015
44,10,0,0.015
45,0,0,0.0175
46,0,20,0.0175
47,20,20,0.0175
48,20,0,0.0175
49,0,0,0.02
50,0,10,0.02
51,0,20,0.02
52,10,20,0.02
53,20,20,0.02
54,20,10,0.02
55,20,0,0.02
56,10,0,0.02
elements:
1,1,1,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
2,1,2,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32
3,1,3,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44
4,1,4,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56
restrained nodes:
1,1,1,1,0.,0.,0.
2,1,0,1,0.,0.,0.
3,1,0,1,0.,0.,0.
4,0,0,1,0.,0.,0.
5,0,0,1,0.,0.,0.
6,0,0,1,0.,0.,0.
7,0,1,1,0.,0.,0.
8,0,1,1,0.,0.,0.
9,1,1,0,0.,0.,0.
10,1,0,0,0.,0.,0.
13,1,1,0,0.,0.,0.
14,1,0,0,0.,0.,0.
15,1,0,0,0.,0.,0.
21,1,1,0,0.,0.,0.
22,1,0,0,0.,0.,0.
25,1,1,0,0.,0.,0.
26,1,0,0,0.,0.,0.
27,1,0,0,0.,0.,0.

33,1,1,0,0.,0.,0.
34,1,0,0,0.,0.,0.
37,1,1,0,0.,0.,0.
38,1,0,0,0.,0.,0.
39,1,0,0,0.,0.,0.
45,1,1,0,0.,0.,0.
46,1,0,0,0.,0.,0.
49,1,1,0,0.,0.,0.
50,1,0,0,0.,0.,0.
51,1,0,0,0.,0.,0.
12,0,1,0,0.,0.,0.
20,0,1,0,0.,0.,0.
19,0,1,0,0.,0.,0.
24,0,1,0,0.,0.,0.
32,0,1,0,0.,0.,0.
31,0,1,0,0.,0.,0.
36,0,1,0,0.,0.,0.
44,0,1,0,0.,0.,0.
43,0,1,0,0.,0.,0.
48,0,1,0,0.,0.,0.
56,0,1,0,0.,0.,0.
55,0,1,0,0.,0.,0.
real constant:
1,45.
2,45.
3,-45.
4,-45.
material group:
1,19.17d6,1.368d6,.9297d6,0.273,0.273,7.5d5,0.4,4
1,7.102d-13,9.391d-13
1,0.,932.6.,0.,932.6.
1,1748.,1049.,1049.,2103.
1,0.,6.033d-04,0.,-1.703d-03
1,3.537d-05,6.75d-05,8.55d-04,-2.344d-04
1,1.,0.0000001,0.000015,0.000015
1,75.,75.,6
1,.4,.2,.2
1,195600.,6800.,544.,.65,.65
1,1.861,2.097
1,0.1128,0.1251
1,0.01184,0.01309
1,0.001523,0.001682
1,0.000243,0.000267
1,0.0000368,0.00003955

```
1,9.522d-09,2.297d-08
1,9.453d-09,1.835d-08
1,1.641d-08,2.916d-08
1,2.680d-08,4.346d-08
1,4.134d-08,6.299d-08
1,8.915d-08,12.51d-08
nodel force
1,0,0,0
7,62.49997,0,0
5,62.49997,0,0
19,124.9988,0,0
17,124.9988,0,0
31,124.9988,0,0
29,124.9988,0,0
43,124.9988,0,0
41,124.9988,0,0
55,62.49997,0,0
53,62.49997,0,0
6,-249.998,0,0
12,-249.998,0,0
11,-249.998,0,0
18,-499.995,0,0
24,-249.998,0,0
23,-249.998,0,0
30,-499.995,0,0
36,-249.998,0,0
35,-249.998,0,0
42,-499.995,0,0
48,-249.998,0,0
47,-249.998,0,0
54,-249.998,0,0
56,0,0,0
output table:
1,1,0,0
4
4
END
```

APPENDIX B

COMMON BLOCKS AND DIMENSIONS OF ARRAY VARIABLES

```
common/contro/npoin,nelem,nnode,ndofn,ndime,nstre,ngaus,
  nprop,nmats,nvfix,nevab,icase,ncase,itemp,iprob,nprob,
  ifail,nreal

common/contro2/tstart,tfinal,tunld,dt,error

common/lgdata/coord(mpoin,ndime),realc(nreal),props(mmats,nprop),
  presc(mvfix,ndofn),asdis(ntotv),eload(melem,nevab),strin(nstre,
  ntotg),nofix(mvfix),ifpre(mvfix,ndofn),lnods(melem,nnode),matno
  (melem),eloadn(melem,nevab),ireal(melem)
common/work/elcod(ndime,nnode),shape(nnode),deriv(ndime,nnode),
  dmatx(nstre,nstre),cartd(ndime,nnode),dbmat(nstre,nevab),bmatx
  (nstre,nevab),smatx(nstre,nevab,ngasp),gpcod(ndime,ngasp),neror
  (24),dsmatx(nstre,nevab,ngasp)

common/crepin/g2t(ngasp,melem),g2told(ngasp,melem),g2s(ngasp,
  melem),dsit(ngasp,melem),dsis(ngasp,melem),gamt(5,ngasp,melem),
  gams(5,ngasp,melem),g2sold(ngasp,melem),qs1(ngasp,melem),qs2
  (ngasp,melem),qs3(ngasp,melem),qs4(ngasp,melem),qs5(ngasp,melem),
  qs6(ngasp,melem),qr1(5,ngasp,melem),qr2(5,ngasp,melem),qr3(5,
  ngasp,melem),qr4(5,ngasp,melem),qr5(5,ngasp,melem),qr6(5,ngasp,
  melem),sig(6,ngasp,melem),sigold(6,ngasp,melem)

common/creep/qr1old(5,ngasp,melem),qr2old(5,ngasp,melem),qr3old
  (5,ngasp,melem),qr4old(5,ngasp,melem),qr5old(5,ngasp,melem),
  qr6old(5,ngasp,melem),gamto(5,ngasp,melem),gamso(5,ngasp,melem),
  sigvold(nstre,ngasp,melem)
common/creep2/qs1old(ngasp,melem),qs2old(ngasp,melem),qs3old(ngasp,

  melem),qs4old(ngasp,melem),qs5old(ngasp,melem),qs6old(ngasp,
  melem),dsitold(ngasp,melem),dsisold(ngasp,melem),g2tvold(ngasp,
  melem),g2svold(ngasp,melem)

common/tmatx/to(nstre,nstre,melem),tde(nstre,ngasp,melem)

common/add/tfact(melem),kfact(melem)

common/print/iprtrs,iprtrn,iprdiss,iprforc,noeltrs(melem),noeltrn
  (melem),nodisp(mpoin),notrs,notrn,nodis
```

where

mpoin: The maximum number of nodal points for which the program is to be dimensioned.

ndime: Number of coordinate components.

mmats: Maximum number of material sets.

nprop: The number of material parameters required to define the characteristics of a material completely.

mvfix: Maximum number of fixed nodes.

ndofn: The number of degrees of freedom per nodal point.

ntotv: Number of total variables in the structure.
ntotv = mpoin*ndfon.

melem: Maximum number of elements.

nevab: Number of variables per element.
nevab = nnode*ndofn.

nstre: Number of stress components at any point.
ngasp: Total number of Gauss points per element.
ngasp = ngaus*ngaus*ngaus, where ngaus is the order of the Gauss integration adopted.

The dimension statement and the local arrays in subroutine FRONT are listed below.

```
DIMENSION fixed(mpoin*mdofn),equat(mfron),vecrv(mpoin*mdofn),
          gload(mfron),gstif(mstif),estif(mevab,mevab),detau
          (mpoint*mdofn),iffix(mpoin*mdofn),nacva(mfron),locel
          (mevab),ndest(mevab)
```

where

mdofn: The maximum number of degrees of freedom per node for which the program is to be dimensioned.
mevab: The maximum number of nodal variables per element for which the program is to be dimensioned.
mfron: The maximum permissible number of variables (degrees of freedom) allowed in the front.
mstif: The maximum permissible number of positions in the one-dimensional global stiffness array.

After the maximum frontwidth MFRON is defined, the maximum space required in the global stiffness array, GSTIF, can be calculated for a given maximum frontwidth as follows

```
MSTIF = MFRON*(MFRON+1)/2.
```

APPENDIX C

LIST OF THE FINITE ELEMENT PROGRAM

```
      PROGRAM lamcrep3
************************************************************************
*     This program can be used to solve nonlinear creep problem in     *
*     three-dimensional analysis.  A 3-d isoparamtic element has been  *
*     employed in the program.                                         *
************************************************************************
      include 'pres.inc'
      include 'globe.inc'
      include 'fail.inc'
      common/fail2/dforc,newfe
      character*11 result1,result2,sdate,stime
      dimension title(12)
c
      open(25,file='data.dat',status='old')
      open(26,file='creep.out',status='unknown')
      open(1,status='scratch',form='unformatted')
      open(2,status='scratch',form='unformatted')
      open(3,status='scratch',form='unformatted')
      open(4,status='scratch',form='unformatted')
      open(7,status='scratch',form='unformatted')
      open(9,file='data.out',status='unknown')
      open(10,file='stress.out',status='unknown')
      open(11,file='disp.out',status='unknown')
      open(12,file='strain.out',status='unknown')
      open(13,file='inform.dat',status='unknown')
c
c********* record the beginning time **********
      call date(result1)
      call time(result2)
c*******************************************
c
c**** read information from file 'data.dat'
      read(25,*) nprob
  900 format(i5)
      write(9,905) nprob
  905 format(1h ,5x,'Total no. of problems =',i5)
      do 20 iprob=1,nprob
      read(25,910) title
  910 format(12a6)
      write(26,915) iprob,title
      write(10,915) iprob,title
      write(9,915) iprob,title
      write(12,915) iprob,title
  915 format(//,6x,12hProblem No. ,i3,10x,12a6)
c
c***  call the subroutine which reads most of
c     the problem data
c
      write(13,*)'Read input data.....'
      call input
      ntotv=npoin*ndofn
c**** initialize variables before time iterations ***********
```

```
      write(13,*)'Initialize variables....'
      do 5 i=1,ntotv
    5    asdis(ntotv)=0.0
      err=1.0
      kf=0
      dforc=1.
      newfe=0
      call initial
      do 10 icase=1,ncase
c
c**** write headers in the output files
      write(26,987)
      write(10,907)
      write(11,988)
      write(12,986)
c***** begin time iterations***************************
      ptime=0.
      it=0
   920 continue
      it=it+1
      write(26,1001) ptime
      write(10,1001) ptime
      write(11,1001) ptime
      write(12,1001) ptime
 1001 format('t=',f14.2)
 986  format (/,7x,' XX-STRAIN',3x,
     1 'YY-STRAIN',3x,'ZZ-STRAIN',3x,'YZ-STRAIN',3x,'XZ-STRAIN',3x,
     2 'XY-STRAIN')
 987  format (/,'  TIME',2x,' XX-STRAIN',3x,
     1 'YY-STRAIN',3x,'ZZ-STRAIN',3x,'YZ-STRAIN',3x,'XZ-STRAIN',3x,
     2 'XY-STRAIN')
 907  format (4x,'11-STRESS',3x,
     1 '22-STRESS',3x,'33-STRESS',3x,'23-STRESS',3x,'13-STRESS',3x,
     2 '12-STRESS'/)
 988  format(1x,'Displacements'//,6x,4hnode,5x,7hx-disp.,
     ,7x,7hy-disp.,7x,7hz-disp./)
      ic=0
    4 continue
      ic=ic+1
      rewind 1
      rewind 2
      rewind 3
      rewind 4
      rewind 7
c***  next create the element stiffness file,
      write(13,*)'Forming element stifness.....'
      call stife(ptime,ic)
c
c***  Merge and solve the resulting equations
c     by the frontal solver
c
      write(13,*)'Solving the equations.......'
      call front(err)
c
c***  Compute the stresses in all the elements,
c
      call stress(ptime,err,kf,it)
      if (ic.ge.15) then
      write(26,*)'Nonconverging problem! ic=',ic
      write(26,*)'error=',err
      write(13,*)'Nonconverging problem! ic=',ic
      write(13,*)'error=',err
```

```
      close(1)
      close(2)
      close(3)
      close(4)
      close(7)
      close(9)
      close(10)
      close(11)
      close(12)
      close(25)
      close(26)
      stop
      endif
      write(13,990)ptime,it,ic,err
  990 format(4x,'time=',f12.2,'  at time step=',i4/
     ,,'   iteration no',i3,'  completed.','  error=',e12.4)
      if(err.gt.error) go to 4
c      write(26,1000)ic,ptime,err
      write(13,1000)ic,ptime,err
 1000 format('  Problem converged at iteration',i2,2x,'t=',f8.2/,
     ,'  error=',e12.4//)
      if(kf.eq.1) then
      write(26,*)'****** All plays have failed *******'
      go to 9000
      endif
c      if(ptime.ge.10.) dt=10.
      dt=1.2*dt
      ptime=ptime+dt
      if(ptime.le.tfinal) then
c       ptime=ptime+dt
        go to 920
      endif
   10 continue
   20 continue
9000  continue
c******record the ending time************
      sdate=result1
      stime=result2
      call date(result1)
      call time(result2)
      write(26,7000)sdate,stime
      write(26,8000)result1,result2
 7000 format(' Program began at ',a11,a11)
 8000 format(' Program stop at ',a11,a11)
      close(1)
      close(2)
      close(3)
      close(4)
      close(7)
      close(9)
      close(10)
      close(11)
      close(12)
      close(25)
      close(26)
      stop ' ******** Program completed ********'
      end
      subroutine input
      include 'pres.inc'
      character*8 selers,selern,seledis
      double precision lamtt,lamts
      include 'globe.inc'
```

```
      include 'const1.inc'
      include 'print.inc'
      dimension title(12),point(3)
      data anode/4hnode/
      data all/3hall/
      data sele/4hsele/
c*********************************************************************
c*  this subroutine reads information form the file named data.dat. *
c*********************************************************************
c***  read the first data card, and echo it
c     immediately,
c
      read(25,*) npoin,nelem,nvfix,ncase,
     , nnode,ndofn,nmats,nreal,nprop,ngaus,ndime,nstre
     ,,ifail
      read(25,*)tstart,tfinal,tunld,dt,error
      nevab=ndofn*nnode
      write(9,905) npoin,nelem,nvfix,ncase,
     , nnode,ndofn,nmats,nreal,nprop,ngaus,ndime,
     , nstre,nevab,ifail,tstart,tfinal,tunld,dt,error
  905 format(//8h npoin =,i4,4x,8h nelem =,i4,
     , 4x,8h nvfix =,i4,4x,8h ncase =,i4,4x,
     , 8h nnode =,i4,4x/,8h ndofn =,i4,4x,
     , 8h nmats =,i4,4x,8h nreal =,i4,4x,
     , 8h nprop =,i4,4x,8h ngaus =,i4,4x,/
     , 8h ndime =,i4,4x,8h nstre =,i4,4x,
     , 8h nevab =,i4,4x,8h ifail =,i4,4x,//
     , 8h tstar =,f5.1,3x,8h tfina =,f12.1,2x,
     , 7htunld =,f12.1,1x,5h dt =,f5.2,3x,/
     , 8h error = ,e10.4/)
      call check1
c
c***  zero all the nodal coordinates, prior
c     to reading some of them,
c
      do 20 ipoin=1,npoin
      do 20 idime=1,ndime
   20 coord(ipoin,idime)=0.0
c
c***  read some nodal coordinates, finishing
c     with the last node of all,
c
      read(25,926)dummyt
  926 format(a20)
      write(9,920)
  920 format(//25h  nodal point coordinates)
      write(9,925)
  925 format(6h  node,7x,1hx,9x,1hy,9x,1hz)
   30 read(25,*) ipoin,(coord(ipoin,idime),
     , idime=1,ndime)
  930 format(i5,5f10.5)
      if(ipoin.ne.npoin) go to 30
c
   40 continue
      do 50 ipoin=1,npoin
   50 write(9,935) ipoin,(coord(ipoin,idime),
     , idime=1,ndime)
  935 format(1x,i5,3f10.3)
c
c***  read the element nodal connections, and
c     the property numbers,
c
```

```
c
      read(25,926)dummyt
      write (9,2005) (anode,i,i=1,nnode)
 2005 format(///40h E L E M E N T  I N F O R M A T I O N
     1       ///17hELEM. PROP. REAL.
     2       1x,8(a4,i1,3x)/18x,a4,i1,3x,7(a4,i2,2x)/
     3       18x,8(a4,i2,2x))
      do 3000 ielem=1,nelem
      read(25,*)numel,matno(numel),ireal(numel),(lnods(numel,inode),
     1inode=1,nnode)
3000  write (9,2004) numel,matno(numel),ireal(numel),
     1               (lnods(numel,inode),inode=1,nnode)
c
 2004 format(/i4,1x,i3,1x,i4,2x,i4,7(4x,i4)/
     1       15x,i4,7(4x,i4)/15x,i4,7(4x,i4))
c
c*** read the fixed values,
c
      read(25,926)dummyt
      write(9,940)
  940 format(//17h RESTRAINED NODES)
      write(9,945)
  945 format(5h node,1x,4hcodf,6x,
     , 12hfixed values)
      do 80 ivfix=1,nvfix
      read(25,*) nofix(ivfix),(ifpre(ivfix,
     , idofn),idofn=1,ndofn),(presc(ivfix,
     , idofn),idofn=1,ndofn)
   80 write(9,955) nofix(ivfix),(ifpre(ivfix,
     , idofn),idofn=1,ndofn),(presc(ivfix,
     , idofn),idofn=1,ndofn)
  955 format(1x,i4,2x,3i1,3f10.6)
c   90 continue
c
c*** read the real constant and available selection of element
c     properties,
c
      read(25,926)dummyt
      write(9,*)' Table of real constants'
      do 99 irel=1,nreal
      read(25,*)nureal,realc(nureal)
      write(9,927)nureal,realc(nureal)
927   format(1x,'Real constant',i3,' =', f7.2)
99    continue
      read(25,926)dummyt
      write(9,960)
  960 format(//21h  Material properties)
      write(9,965)
  965 format(6hnumber,2x,10hProperties,
     /' e,et,g,v,vt,em,vm,itype:')
      do 100 imats=1,nmats
      read(25,*) numat,(props(numat,iprop),
     , iprop=1,8)
      write(9,970) numat,(props(numat,iprop),
     , iprop=1,8)
  970 format(1x,i2,2x,7(e8.2,1x),f6.2,2x,i1)
c*******iprop9-10:dft,dfs
      read(25,*)numat,(props(numat,iprop),iprop=9,10)
      write(9,975)(props(numat,iprop),iprop=9,10)
  975 format(1h ,' dft.....................=',e10.3/,
     /       1h ,' dfs.....................=',e10.3/)
c*******iprop11-14:b0t,b1t,b2t,bt
```

```
      read(25,*)numat,(props(numat,iprop),iprop=11,14)
      write(9,976)(props(numat,iprop),iprop=11,14)
  976 format(1h ,' b0t.....=',e10.3/,
     /       1h ,' b1t.....=',e10.3/,
     /       1h ,' b2t.....=',e10.3/,
     /       1h ,' bt.....=',e10.3/)
c*******iprop15-18:b0s,b1s,b2s,bs
      read(25,*)numat,(props(numat,iprop),iprop=15,18)
      write(9,977)(props(numat,iprop),iprop=15,18)
  977 format(1h ,' b0s.....=',e10.3/,
     /       1h ,' b1s.....=',e10.3/,
     /       1h ,' b2s.....=',e10.3/,
     /       1h ,' bs.....=',e10.3/)
c*******iprop19-22:k0t,k1t,k2t,kt
      read(25,*)numat,(props(numat,iprop),iprop=19,22)
      write(9,978)(props(numat,iprop),iprop=19,22)
  978 format(1h ,' k0t.....=',e10.3/,
     /       1h ,' k1t.....=',e10.3/,
     /       1h ,' k2t.....=',e10.3/,
     /       1h ,' kt.....=',e10.3/)
c*******iprop23-26:k0s,k1s,k2s,ks
      read(25,*)numat,(props(numat,iprop),iprop=23,26)
      write(9,979)(props(numat,iprop),iprop=23,26)
  979 format(1h ,' k0s.....=',e10.3/,
     /       1h ,' k1s.....=',e10.3/,
     /       1h ,' k2s.....=',e10.3/,
     /       1h ,' ks.....=',e10.3/)
c******* iprop27-30:g023,alf(1),alf(2),alf(3)
      read(25,*)numat,(props(numat,iprop),iprop=27,30)
      write(9,1055)(props(numat,iprop),iprop=27,30)
 1055 format(1h ,'g023.....=',e10.3/,
     /       1h ,'alf(1),alf(2),alf(3)=',3(e10.3,2x)/)
c*******iprop31-33:temp,tempo,nr
      read(25,*)numat,(props(numat,iprop),
     , iprop=31,33)
      nr=int(props(numat,33))
      write(9,995)  (props(numat,iprop),
     / iprop=31,32),nr
c********temp,tempo,nr
  995 format(1h ,4x,25htemp.....=,f10.3/,
     /       1h ,4x,25htempo(reference temp.)..=,f10.3/,
     /       1h ,4x,25hnr.....=,i10/)
c********iprop34-36:d11,d22,d12
      read(25,*)numat,(props(numat,iprop),
     , iprop=34,36)
      write(9,996)(props(numat,iprop),
     / iprop=34,36)
  996 format(1h ,4x,25hd11.....=,f10.3/,
     /       1h ,4x,25hd22.....=,f10.3/,
     /       1h ,4x,25hd12.....=,f10.3/)
c*******iprop37-41:x,y,s12,s13,s23
      read(25,*)numat,(props(numat,iprop),iprop=37,41)
      write(9,994)(props(numat,iprop),iprop=37,41)
  994 format(1h ,' x.....=',e10.3/,
     /       1h ,' y.....=',e10.3/,
     /       1h ,'s12.....=',e10.3/,
     /       1h ,'s23.....=',e10.3/,
     /       1h ,'s13.....=',e10.3/)
c************input creep costants ********************
      write(9,980)
  980 format(15hcreep constants)
c
```

```
      do 985 ir=1,nr
      read(25,*)numat,lamtt(numat,ir),lamts(numat,ir)
 985  write(9,1100)ir,lamtt(numat,ir),lamts(numat,ir)
 1100 format(1h ,'nr=',i2,2x/,'  lamtt,lamts:'/,
     /       5x,2(e12.5)/)
      do 990 ir=1,nr
      read(25,*)numat,dtr(numat,ir),dsr(numat,ir)
 990  write(9,1150)ir,dtr(numat,ir),dsr(numat,ir)
 1150 format(1h ,'nr=',i2,2x/,'dtr,dsr:'/,
     /       5x,2(e12.5)/)
  991 continue
  100 continue
c*******************************************************
c**** input external nodal loads
c
      do 2000 ielem=1,nelem
      do 2000 ievab=1,nevab
 2000 eloadn(ielem,ievab)=0.0
      read(25,2100) title
 2100 format(12a6)
      write(9,2200) title
 2200 format(1h ,12a6)
c
c***  read data controlling loading types
c     to be input
c
      read(25,*) iplod,igrav,iedge,itemp
      write(9,2300) iplod,igrav,iedge,itemp
 2300 format(4i5)
c
c***  read nodal point loads
c
      if(iplod.eq.0) go to 5000
   21 read(25,*) lodpt,(point(idofn),idofn=
     , 1,ndofn)
      write(9,2400) lodpt,(point(idofn),idofn=
     , 1,ndofn)
 2400 format(i5,3f15.5)
c
c***  associate the nodal point loads with
c     an element
c
      do 2500 ielem=1,nelem
      do 2500 inode=1,nnode
      nloca=lnods(ielem,inode)
      if(lodpt.eq.nloca) go to 41
 2500 continue
   41 do 51 idofn=1,ndofn
      ngash=(inode-1)*ndofn+idofn
   51 eloadn(ielem,ngash)=point(idofn)
      if(lodpt.lt.npoin) go to 21
 5000 continue
c      if(igrav.eq.0) go to 6000
c 6000 continue
c      if(iedge.eq.0) go to 7000
c 7000 continue
c      if(itemp.eq.0) go to 8000
 8000 continue
 2700 continue
 1005 format(1x,i4,5x,6e10.2/(9(10x,6e10.2/)))
c*********************************
c*       set up output tables     *
```

```
c**********************************
      read(25,926)dummyt
      read(25,*)iprtrs,iprtrn,iprdis,iprforc
      if(iprtrs.eq.0) go to 8100
      read(25,*)notrs
      if(notrs.eq.nelem) then
      do 8050 i=1,notrs
8050      noeltrs(i)=i
      else
      read(25,*)(noeltrs(i),i=1,notrs)
      endif
8100  if(iprtrn.eq.0) go to 8200
      read(25,*)notrn
      if(notrn.eq.nelem) then
      do 8150 i=1,notrn
8150      noeltrn(i)=i
      else
      read(25,*)(noeltrn(i),i=1,notrn)
      endif
8200  if(iprdis.eq.0) go to 8300
      read(25,*)nodis
      if(nodis.eq.npoin) then
      do 8250 i=1,npoin
8250      nodisp(i)=i
      else
      read(25,*)(nodisp(i),i=1,nodis)
      endif
8300  if(iprforc.eq.0)go to 8400
c*
8400  continue
      call check2
      return
      end
      subroutine check1
c
c***  to criticize the data control card and
c     print any diagnostics
c
      include 'pres.inc'
      include 'globe.inc'
c
      do 10 ieror=1,24
   10 neror(ieror)=0
c
c***  create the diagnostic messages
c
      if(npoin.le.0) neror(1)=1
      if(nelem*nnode.lt.npoin) neror(2)=1
      if(nvfix.lt.1.or.nvfix.gt.npoin) neror(3)=1
      if(ncase.le.0) neror(4)=1
c      if(ntype.lt.0.or.ntype.gt.3) neror(5)=1
      if(nnode.lt.3.or.nnode.gt.20) neror(6)=1
      if(ndofn.lt.2.or.ndofn.gt.3) neror(7)=1
      if(nmats.le.0.or.nmats.gt.nelem) neror(8)=1
      if(nprop.lt.3.or.nprop.gt.41) neror(9)=1
      if(ngaus.lt.2.or.ngaus.gt.4) neror(10)=1
      if(ndime.lt.1.or.ndime.gt.3) neror(11)=1
      if(nstre.lt.2.or.nstre.gt.6) neror(12)=1
c
c***  either return,or else print the errors
c     diagnosed
c
```

```
      keror=0
      do 20 ieror=1,12
      if(neror(ieror).eq.0) go to 20
      keror=1
      write(26,900) ieror
  900 format(//24h *** diagnosis by check1,
     , 6h error,i3)
   20 continue
      if(keror.eq.0) return
c
c***  otherwise echo all the remaining data
c     without further comment
c
      call echo
      end
      subroutine echo
      include 'pres.inc'
      include 'globe.inc'
      dimension ntitl(80)
      write(26,900)
  900 format(//25h now follows a listing of,
     , 25h post-disaster data cards/)
   10 read(25,905) ntitl
  905 format(80a1)
      write(26,910) ntitl
  910 format(20x,80a1)
      go to 10
      end
      subroutine check2
      include 'pres.inc'
      include 'globe.inc'
      dimension ndfro(400)
c
c***  to criticize the data from subroutine input
c
      mfron=720
c
      write(13,*)'Checking.....'
c***  check against two identical nonzero
c     nodal coordinates
c
      do 10 ielem=1,nelem
   10 ndfro(ielem)=0
      do 40 ipoin=2,npoin
      kpqin=ipoin-1
      do 30 jpoin=1,kpqin
      do 20 idime=1,ndime
      if(coord(ipoin,idime).ne.coord(jpoin,
     , idime)) go to 30
   20 continue
      neror(13)=neror(13)+1
   30 continue
   40 continue
c
c***  check the list of element property numbers
c
      do 50 ielem=1,nelem
   50 if(matno(ielem).le.0.or.matno(ielem).gt.
     , nmats) neror(14)=neror(14)+1
c
c***  check for impossible node numbers
c
```

```
      do 70 ielem=1,nelem
      do 60 inode=1,nnode
      if(lnods(ielem,inode).eq.0) neror(15)=
     , neror(15)+1
   60 if(lnods(ielem,inode).lt.0.or.lnods(ielem,
     , inode).gt.npoin) neror(16)=neror(16)+1
c      write(26,*)'neror16=',neror(16),ielem,inode
   70 continue
c
c***  check for any repetition of a node
c     number within an element
c
      do 140 ipoin=1,npoin
      kstar=0
      do 100 ielem=1,nelem
      kzero=0
      do 90 inode=1,nnode
      if(lnods(ielem,inode).ne.ipoin) go to 90
      kzero=kzero+1
      if(kzero.gt.1) neror(17)=neror(17)+1
c      write(26,*)'neror17=',neror(17),ielem,inode,ipoin
c
c***  seek first,last and intermediate
c     appearances of node ipoin
      if(kstar.ne.0) go to 80
c
      kstar=ielem
c
c***  calculate increase or decrease in
c     frontwidth at each element stage
c
      ndfro(ielem)=ndfro(ielem)+ndofn
   80 continue
c
c***  and change the sign of the last
c     appearance of each node
c
      klast=ielem
      nlast=inode
   90 continue
  100 continue
      if(kstar.eq.0) go to 110
      if(klast.lt.nelem) ndfro(klast+1)=
     , ndfro(klast+1)-ndofn
      lnods(klast,nlast)=-ipoin
      go to 140
c
c***  check that coordinates for an unused
c     node have not been specified
c
  110 write(26,900) ipoin
  900 format(/15h check why node,i4,
     , 14h never appears)
      neror(18)=neror(18)+1
      sigma=0.0
      do 120 idime=1,ndime
  120 sigma=sigma+abs(coord(ipoin,idime))
      if(sigma.ne.0.0) neror(19)=neror(19)+1
c
c***  check that an unused node number is not
c     a restrained node
c
```

```
      do 130 ivfix=1,nvfix
  130 if(nofix(ivfix).eq.ipoin) neror(20)=
     , neror(20)+1
  140 continue
c
c***  calculate the largest frontwidth
c
      nfron=0
      kfron=0
      do 150 ielem=1,nelem
      nfron=nfron+ndfro(ielem)
  150 if(nfron.gt.kfron) kfron=nfron
      write(26,905) kfron
      write(13,905) kfron
  905 format(//30h  Max frontwidth encountered =,i15)
      if(kfron.gt.mfron) neror(21)=1
c
c***  continue checking the data for the
c     fixed values
c
      do 170 ivfix=1,nvfix
      if(nofix(ivfix).le.0.or.nofix(ivfix)
     ,.gt.npoin) neror(22)=neror(22)+1
      kount=0
      do 160 idofn=1,ndofn
  160 if(ifpre(ivfix,idofn).gt.0) kount=1
      if(kount.eq.0) neror(23)=neror(23)+1
      kvfix=ivfix-1
      do 170 jvfix=1,kvfix
  170 if(ivfix.ne.1.and.nofix(ivfix).eq.
     , nofix(jvfix)) neror(24)=neror(24)+1
      keror=0
      do 180 ieror=13,24
      if(neror(ieror).eq.0) go to 180
      keror=1
      write(26,910) ieror,neror(ieror)
  910 format(//30h*** diagnosis by check2, error,
     , i3,6x,18h associated number,i5)
  180 continue
      if(keror.ne.0) go to 200
c
c***  return all nodal connection numbers to
c     positive values
c
      do 190 ielem=1,nelem
      do 190 inode=1,nnode
  190 lnods(ielem,inode)=iabs(lnods(ielem,inode))
      return
  200 call echo
      end
      subroutine initial
      include 'pres.inc'
      include 'globe.inc'
      include 'crpin.inc'
      include 'fail.inc'
c      common/add2/strain(6,27,40),sigb(6,27,40)
c*****************************************************************
c     this subroutine is used to initialize variables for nonlinear   *
c     analysis                                                        *
c                                                                     *
c************initialize variables*************************************
c
```

```
      lint=ngaus*ngaus*ngaus
      do 20 ielem=1,nelem
        lprop=matno(ielem)
        tfact(ielem)=0.
        kfact(ielem)=0
        itype=props(lprop,8)
        if(itype.eq.3) go to 20
        nr=props(lprop,33)
        kgasp=0
      do 11 kglsp=1,lint
      kgasp=kgasp+1
      dsit(kgasp,ielem)=0.
      dsis(kgasp,ielem)=0.
      g2t(kgasp,ielem)=0.
      g2s(kgasp,ielem)=0.
      g2told(kgasp,ielem)=0.
      g2sold(kgasp,ielem)=0.
      do 13 ir=1,nr
      gamt(ir,kgasp,ielem)=0.
      gams(ir,kgasp,ielem)=0.
      qr1(ir,kgasp,ielem)=0.
      qr2(ir,kgasp,ielem)=0.
      qr3(ir,kgasp,ielem)=0.
      qr4(ir,kgasp,ielem)=0.
      qr5(ir,kgasp,ielem)=0.
      qr6(ir,kgasp,ielem)=0.
   13 continue
      qs1(kgasp,ielem)=0.
      qs2(kgasp,ielem)=0.
      qs3(kgasp,ielem)=0.
      qs4(kgasp,ielem)=0.
      qs5(kgasp,ielem)=0.
      qs6(kgasp,ielem)=0.
      do 12 j=1,6
      sig(j,kgasp,ielem)=0.
      sigold(j,kgasp,ielem)=0.
c        strain(j,kgasp,ielem)=0.
   12 continue
   11 continue
   20 continue
      return
      end
      subroutine stife(time,ic)
      include 'pres.inc'
      include 'globe.inc'
      include 'tmatx.inc'
      dimension estif(60,60),xg(4,4),wgt(4,4)
      common/dg/dgmatx(6,6,27)
      common/fail2/dforc,newfe
c*********************************************************************
c     set up gauss point table:
c
      data xg /     0.,             0.,             0.,             0.,
     1 -.5773502691896, .5773502691896,             0.,             0.,
     2 -.7745966692415, .0000000000000, .7745966692415,             0.,
     3 -.8611363115941,-.3399810435849, .3399810435849, .8611363115941/
      data wgt / 2.000,              0.,             0.,             0.,
     1 1.0000000000000,1.0000000000000,             0.,             0.,
     2  .5555555555556, .8888888888889, .5555555555556,             0.,
     3  .3478548451375, .6521451548625, .6521451548625, .3478548451375/
c*********************************************************************
c************* file 1: element sifness matrix
```

```
c************* file 3: [d][b] matrix at each gauss point/element
c************* file 7: [b] matrix at each gauss point/element
      rewind 1
      rewind 3
      rewind 7
c      rewind 8
c***  loop over each element
c
       do 70 ielem=1,nelem
       lreal=ireal(ielem)
       lprop=matno(ielem)
       itype=props(lprop,8)
       kgasp=0.
c       write(13,*)'element.......',ielem
c
c***  evaluate the coordinates of the element
c     nodal points
c
       do 10 inode=1,nnode
       lnode=lnods(ielem,inode)
       do 10 idime=1,ndime
   10  elcod(idime,inode)=coord(lnode,idime)
       do 11 ievab=1,nevab
   11    eload(ielem,ievab)=0.0
c
c***  evaluate the d-matrix
c***  initialize the element stiffness matrix
c
      do 20 ievab=1,nevab
      do 20 jevab=1,nevab
   20 estif(ievab,jevab)=0.0
      kgasp=0
c
c***  enter loops for area numberical integration
c
      do 50 igaus=1,ngaus
      r=xg(igaus,ngaus)
      do 50 jgaus=1,ngaus
      s=xg(jgaus,ngaus)
      do 50 kgaus=1,ngaus
      t=xg(kgaus,ngaus)
      kgasp=kgasp+1
      wt=wgt(igaus,ngaus)*wgt(jgaus,ngaus)*wgt(kgaus,ngaus)
c***  evaluate the dmatx for nonliner creep materials
       call dmatrx(lprop,lreal,ielem,ic,kgasp,itype)
c   25 continue
c
c***  evaluate the shape functions,elemental
c     volume,etc.
c
c**** shap3 gives jacobian matrix and its determinat
      call shap(r,s,t)
      call jacob(ielem,djacb,kgasp)
      dvolu=djacb*wt
c
c***  evaluate the b and db matrices
c
      call bmatrx
      call dbe
c
c***  calculate the element stiffnesses
c
```

```
      do 30 ievab=1,nevab
      do 30 jevab=ievab,nevab
      do 30 istre=1,nstre
   30 estif(ievab,jevab)=estif(ievab,jevab)+
     , bmatx(istre,ievab)*dbmat(istre,
     , jevab)*dvolu
c
c***  calculate unbalanced load vector increaments due to creep
c         strain
c
      do 1060 ievab=1,nevab
      do 1060 istre=1,nstre
       eload(ielem,ievab)=eload(ielem,ievab)+
     ,    bmatx(istre,ievab)*tde(istre,kgasp,ielem)*dvolu
 1060 continue
c***  store the components of the b matrix and db matrix for
c     the element
c
      do 40 istre=1,nstre
      do 40 ievab=1,nevab
      dsmatx(istre,ievab,kgasp)=dbmat(istre,ievab)
   40 smatx(istre,ievab,kgasp)=bmatx(istre,ievab)
      do 45 istre=1,nstre
      do 45 jstre=1,nstre
   45 dgmatx(istre,jstre,kgasp)=dmatx(istre,jstre)
   50 continue
 6000 format(6e12.4)
c*** adding load increament on to the external load vector
      if (time.ge.tunld) go to 1000
      do 295 ievab=1,nevab
  295 eload(ielem,ievab)=dforc*eload(ielem,ievab)+eloadn(ielem,ievab)
 1000 continue
c
c***  construct the lower triangle of the
c     stiffness matrix
c
      do 60 ievab=1,nevab
      do 60 jevab=1,nevab
   60 estif(jevab,ievab)=estif(ievab,jevab)
c
c***  store the stiffness matrix,stress matrix
c     and sampling point coordinates for each
c     element on disc file
c
      write(1) estif
      write(3) dsmatx,gpcod
      write(7) smatx
c      write(8) dgmatx
   70 continue
      return
      end
      subroutine dmatrx(lprop,lreal,ielem,ic,kgasp,itype)
c
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c .                                                                   .
c .                                                                   .
c .   p r o g r a m                                                   .
c .                                                                   .
c .      to generate stress-strain law for isotropic or orthotropic   .
c .      linear elastic or nonlinear creep materials                  .
c .                                                                   .
c .                                                                   .
```

```
c . . . . . .
c
      include 'pres.inc'
      double precision lamtt,lamts,kt,k0s,k1s,k2s,ks,k0t,k1t,k2t
      include 'globe.inc'
      include 'const1.inc'
      include 'crpin.inc'
      include 'crp2.inc'
      include 'tmatx.inc'
      include 'fail.inc'
      common indx,dinv
      dimension d(6,6),dinv(6,6),indx(6),tbar(6,6),tinv(6,6),alf(6),
     ,p(6),sigm(6),sigmold(6),db(6,6),do(6,6),ep(6),ds(6,6),dc(6,6),
     ,td(6,6)
      n=6
      e=props(lprop,1)
      et=props(lprop,2)
      g=props(lprop,3)
      v=props(lprop,4)
      vt=props(lprop,5)
      beta=realc(lreal)
c
      pi=4.*datan(1.d+00)
      pi2=pi*2.
      gam=beta*pi/180.
      if (gam .ge. pi2) gam = gam - pi2
c
c     set the coordinate transformation for rotation of properties
c
c***** tranformation matrix t*************
      sg=dsin(gam)
      cg=dcos(gam)
      do 106 i=1,6
      do 106 j=1,6
       to(i,j,ielem)=0.0
       tinv(i,j)=0.0
       tbar(i,j)=0.0
  106 continue
      do 107 istre=1,nstre
      do 107 jstre=1,nstre
           do(istre,jstre)=0.0
  107 continue
      to(1,1,ielem)=cg**2
      to(1,2,ielem)=sg**2
      to(1,6,ielem)=2.*sg*cg
      to(2,1,ielem)=to(1,2,ielem)
      to(2,2,ielem)=to(1,1,ielem)
      to(2,6,ielem)=-to(1,6,ielem)
      to(3,3,ielem)=1.
      to(4,4,ielem)=cg
      to(4,5,ielem)=-sg
      to(5,4,ielem)=-to(4,5,ielem)
      to(5,5,ielem)=to(4,4,ielem)
      to(6,1,ielem)=-sg*cg
      to(6,2,ielem)=-to(6,1,ielem)
      to(6,6,ielem)=to(1,1,ielem)-to(1,2,ielem)
      tbar(1,1)=cg**2
      tbar(1,2)=sg**2
      tbar(1,3)=0.0
      tbar(1,4)=0.0
      tbar(1,5)=0.0
      tbar(1,6)=sg*cg
```

```
      tbar(2,1)=tbar(1,2)
      tbar(2,2)=tbar(1,1)
      tbar(2,3)=0.0
      tbar(2,4)=0.0
      tbar(2,5)=0.0
      tbar(2,6)=-tbar(1,6)
      tbar(3,1)=0.0
      tbar(3,2)=0.0
      tbar(3,3)=1.0
      tbar(3,4)=0.0
      tbar(3,5)=0.0
      tbar(3,6)=0.0
      tbar(4,1)=0.0
      tbar(4,2)=0.0
      tbar(4,3)=0.0
      tbar(4,4)=cg
      tbar(4,5)=-sg
      tbar(4,6)=0.0
      tbar(5,1)=0.0
      tbar(5,2)=0.0
      tbar(5,3)=0.0
      tbar(5,4)=-tbar(4,5)
      tbar(5,5)=tbar(4,4)
      tbar(5,6)=0.0
      tbar(6,1)=-2.*sg*cg
      tbar(6,2)=-tbar(6,1)
      tbar(6,3)=0.0
      tbar(6,4)=0.0
      tbar(6,5)=0.0
      tbar(6,6)=tbar(1,1)-tbar(1,2)
c      write(26,*)'tbar'
c      write(26,3010)((tbar(it,jt),jt=1,6),it=1,6)
c**** t(transpor)(tinvers in kennedy's note)
      tinv(1,1)=cg**2
      tinv(1,2)=sg**2
      tinv(1,3)=0.
      tinv(1,4)=0.
      tinv(1,5)=0.
      tinv(1,6)=-2.*sg*cg
      tinv(2,1)=tinv(1,2)
      tinv(2,2)=tinv(1,1)
      tinv(2,3)=0.
      tinv(2,4)=0.
      tinv(2,5)=0.
      tinv(2,6)=-tinv(1,6)
      tinv(3,1)=0.
      tinv(3,2)=0.
      tinv(3,3)=1.
      tinv(3,4)=0.
      tinv(3,5)=0.
      tinv(3,6)=0.
      tinv(4,1)=0.
      tinv(4,2)=0.
      tinv(4,3)=0.
      tinv(4,4)=cg
      tinv(4,5)=sg
      tinv(4,6)=0.
      tinv(5,1)=0.
      tinv(5,2)=0.
      tinv(5,3)=0.
      tinv(5,4)=-sg
      tinv(5,5)=cg
```

```
      tinv(5,6)=0.
      tinv(6,1)=sg*cg
      tinv(6,2)=-sg*cg
      tinv(6,3)=0.
      tinv(6,4)=0.
      tinv(6,5)=0.
      tinv(6,6)=(cg**2)-(sg**2)
c*************************************************
c     if nonlinear analysis is required, go to 202
      if(itype.eq.4) go to 202
c*************************************************
c
c     form the strain-stress law
c.... for linear elastic orthotropic material
c

      d1=e
      d2=v
      d3=et
      d4=vt
      d5=g
      do(1,1)=1./d1
      do(2,2)=1./d3
      do(3,3)=1./d3
      do(4,4)=2.*(1./d3+d4/d3)
c      do(4,4)=1./1000000.
      do(5,5)=1./d5
      do(6,6)=1./d5
      do(1,2)=-d2/d1
      do(1,3)=-d2/d1
      do(1,4)=0.0
      do(1,5)=0.0
      do(1,6)=0.0
      do(2,3)=-d4/d3
      do(2,4)=0.0
      do(2,5)=0.0
      do(2,6)=0.0
      do(3,4)=0.0
      do(3,5)=0.0
      do(3,6)=0.0
      do(4,5)=0.0
      do(4,6)=0.0
      do(5,6)=0.0
      do 3000 i=1,6
      do 3000 j=1,6
 3000  do(j,i)=do(i,j)
c*******************inverse of dmatrix*******************
      do 3014 i=1,n
       do 3015 j=1,n
          dinv(i,j)=0.
 3015    continue
      dinv(i,i)=1
 3014 continue
      call ludcmp(do,6,6,indx,c)
      do 3016 jj=1,n
       call lubksb(do,6,6,indx,dinv(1,jj))
 3016 continue
c
c     rotate the stress-strain matrix to global coordinates
c
c    t(transpose) * d(material)
c
```

```
      do 3060 irow=1,6
      do 3050 icol=1,6
      td(irow,icol)=0.0
      do 3048 ik=1,6
      td(irow,icol) = td(irow,icol)+tinv(irow,ik)*dinv(ik,icol)
 3048 continue
 3050 continue
 3060 continue
c
c      lint=ngaus*ngaus*ngaus
c      do 3200 kgasp=1,lint
c
c     t(transpose) * dinv(material) * t
c
      do 3080 irow = 1,6
      do 3075 icol = 1,6
      dmatx(irow,icol) = 0.0
      do 3070 in =1,6
      dmatx(irow,icol) = dmatx(irow,icol)
     , + td(irow,in)*tbar(in,icol)
 3070 continue
 3075 continue
 3080 continue
c 3200 continue
      return
c
c****** form nonlinear creep stress-strain matrix [d]
c
  202 continue
      em=props(lprop,6)
      vm=props(lprop,7)
      dft=props(lprop,9)
      dfs=props(lprop,10)
      b0t=props(lprop,11)
      b1t=props(lprop,12)
      b2t=props(lprop,13)
      bt=props(lprop,14)
      b0s=props(lprop,15)
      b1s=props(lprop,16)
      b2s=props(lprop,17)
      bs=props(lprop,18)
      k0t=props(lprop,19)
      k1t=props(lprop,20)
      k2t=props(lprop,21)
      kt=props(lprop,22)
      k0s=props(lprop,23)
      k1s=props(lprop,24)
      k2s=props(lprop,25)
      ks=props(lprop,26)
      g023=props(lprop,27)
      alf(1)=props(lprop,28)
      alf(2)=props(lprop,29)
      alf(3)=props(lprop,30)
      alf(4)=0.
      alf(5)=0.
      alf(6)=0.
      temp=props(lprop,31)
      tempo=props(lprop,32)
      nr=props(lprop,33)
      d11=props(lprop,34)
      d22=props(lprop,35)
      d12=props(lprop,36)
```

```
c
c*********** elastic part of the dmatrix ***************
c
      d1=e*d11**kfact(ielem)
      d2=v
      d3=et*d22**kfact(ielem)
      d4=vt
      d5=g*d12**kfact(ielem)
      do 25 io=1,6
       do 25 jo=1,6
  25        do(io,jo)=0.0
      do(1,1)=1./d1
      do(2,2)=1./d3
      do(3,3)=1./d3
      do(4,4)=2.*(1./d3+d4/d3)
      do(5,5)=1./d5
      do(6,6)=1./d5
      do(1,2)=-d2/d1
      do(1,3)=-d2/d1
      do(1,4)=0.0
      do(1,5)=0.0
      do(1,6)=0.0
      do(2,3)=-d4/d3
      do(2,4)=0.0
      do(2,5)=0.0
      do(2,6)=0.0
      do(3,4)=0.0
      do(3,5)=0.0
      do(3,6)=0.0
      do(4,5)=0.0
      do(4,6)=0.0
      do(5,6)=0.0
      do 30 i=1,6
      do 30 j=1,6
  30  do(j,i)=do(i,j)
c
c**************** define old variables ***************
c
      if(ic.eq.1) then
c      write(13,*)'  updataing old variables...,ic=',ic
      tempold=temp
c      lint=ngaus*ngaus*ngaus
c      do 151 kgasp=1,lint
       qs1old(kgasp,ielem)=qs1(kgasp,ielem)
       qs2old(kgasp,ielem)=qs2(kgasp,ielem)
       qs3old(kgasp,ielem)=qs3(kgasp,ielem)
       qs4old(kgasp,ielem)=qs4(kgasp,ielem)
       qs5old(kgasp,ielem)=qs5(kgasp,ielem)
       qs6old(kgasp,ielem)=qs6(kgasp,ielem)
       do 150 j=1,6
       sigvold(j,kgasp,ielem)=sigold(j,kgasp,ielem)
       sigold(j,kgasp,ielem)=sig(j,kgasp,ielem)
  150  continue
      g2tvold(kgasp,ielem)=g2told(kgasp,ielem)
      g2told(kgasp,ielem)=g2t(kgasp,ielem)
      g2svold(kgasp,ielem)=g2sold(kgasp,ielem)
      g2sold(kgasp,ielem)=g2s(kgasp,ielem)
      dsitold(kgasp,ielem)=dsit(kgasp,ielem)
      dsisold(kgasp,ielem)=dsis(kgasp,ielem)
      do 155 ir=1,nr
      gamto(ir,kgasp,ielem)=gamt(ir,kgasp,ielem)
      gamso(ir,kgasp,ielem)=gams(ir,kgasp,ielem)
```

```
      qr1old(ir,kgasp,ielem)=qr1(ir,kgasp,ielem)
      qr2old(ir,kgasp,ielem)=qr2(ir,kgasp,ielem)
      qr3old(ir,kgasp,ielem)=qr3(ir,kgasp,ielem)
      qr4old(ir,kgasp,ielem)=qr4(ir,kgasp,ielem)
      qr5old(ir,kgasp,ielem)=qr5(ir,kgasp,ielem)
      qr6old(ir,kgasp,ielem)=qr6(ir,kgasp,ielem)
  155 continue
c  151 continue
       tempold=temp
      endif
  156 continue
 1600 continue
c************ nonlinear part of dmatrix ******************
c****************** calculate g's ************************
c
      tauold=0.
      tau=0.
      do 1201 j=1,6
      sigmold(j)=0.0
      sigm(j)=0.0
 1201 continue
      sigmold(1)=(em/e)*sigold(1,kgasp,ielem)
     ,+(vm-(em/e)*v)*sigold(2,kgasp,ielem)+(vm-(em/e)*vt)
     ,*sigold(3,kgasp,ielem)
      sigm(1)=(em/e)*sig(1,kgasp,ielem)+(vm-(em/e)*v)
     ,*sig(2,kgasp,ielem)+(vm-(em/e)*vt)*sig(3,kgasp,ielem)
      do 1151 j=2,6
      sigmold(j)=sigold(j,kgasp,ielem)
 1151   sigm(j)=sig(j,kgasp,ielem)
       tauold=dsqrt((sigmold(1)-sigmold(2))**2+
     ,(sigmold(1)-sigmold(3))**2+(sigmold(2)
     ,-sigmold(3))**2+6.*(sigmold(4)**2+sigmold(5)**2
     ,+sigmold(6)**2))/3.
       tau=dsqrt((sigm(1)-sigm(2))**2+(sigm(1)
     ,-sigm(3))**2+(sigm(2)-sigm(3))**2+6.
     ,*(sigm(4)**2+sigm(5)**2+sigm(6)**2))/3.
c***** creep properties of T300/5208 GRIEP
c***** transvers g's
       got=1.
       if(tau.le.b1t) then
          g1t=1.
       else
          g1t=1.+k1t*(tau-b1t)
       endif
          g2t(kgasp,ielem)=1.
       if(tauold.le.bt) then
          astold=1.
       else
          astold=dexp(kt*(tauold-bt))
       endif
       if(tau.le.bt) then
          ast=1.
       else
          ast=dexp(kt*(tau-bt))
       endif
c***** shear g's
       if(tau.le.b0s) then
          gos=1.
       else
          gos=1.+k0s*(tau-b0s)
       endif
       if(tau.le.b1s) then
```

```
            g1s=1.
         else
            g1s=1.+k1s*(tau-b1s)
         endif
         if(tau.le.b2s) then
            g2s(kgasp,ielem)=1.
         else
            g2s(kgasp,ielem)=1.+k2s*(tau-b2s)
         endif
         if(tauold.le.bs) then
            assold=1.
         else
            assold=dexp(ks*(tauold-bs))
         endif
         if(tau.le.bs) then
            ass=1.
         else
            ass=dexp(ks*(tau-bs))
         endif
   19    continue
         dsit(kgasp,ielem)=0.5*(1./ast+1./astold)*dt
         dsis(kgasp,ielem)=0.5*(1./ass+1./assold)*dt
c*********************************************************
c******************* [ds] ([da] in note)*****************
c
         do 13 is=1,6
          do 13 js=1,6
         ds(is,js)=0.
    13   continue
         ds(1,1)=do(1,1)
         ds(1,2)=do(1,2)
         ds(1,3)=do(1,3)
         ds(2,1)=ds(1,2)
         ds(2,2)=got*do(2,2)
         ds(2,3)=got*do(2,3)
         ds(3,1)=ds(1,3)
         ds(3,2)=ds(2,3)
         ds(3,3)=ds(2,2)
         ds(4,4)=2.*(ds(2,2)-ds(2,3))
         ds(5,5)=gos*do(5,5)
         ds(6,6)=ds(5,5)
c*********************************************************
c
c***********calculate dc(i,j)=dr(i,j)*(1-gam(i,j))******
         do 21 idc=1,6
         do 20 jdc=1,6
         dc(idc,jdc)=0.
   20    continue
   21    continue
         do 22 ir=1,nr
         gamt(ir,kgasp,ielem)=(1.-dexp(-lamtt(lprop,ir)*
        /dsit(kgasp,ielem)))/(lamtt(lprop,ir)*dsit(kgasp,ielem))
         gams(ir,kgasp,ielem)=(1.-dexp(-lamts(lprop,ir)*
        /dsis(kgasp,ielem)))/(lamts(lprop,ir)*dsis(kgasp,ielem))
         dc(2,2)=dc(2,2)+(1.-gamt(ir,kgasp,ielem))
        ,*dtr(lprop,ir)*g1t*g2t(kgasp,ielem)
         dc(2,3)=-d4*dc(2,2)
         dc(3,2)=dc(2,3)
         dc(3,3)=dc(2,2)
         dc(4,4)=2.*(dc(2,2)-dc(2,3))
         dc(5,5)=dc(5,5)+(1.-gams(ir,kgasp,ielem))
        ,*dsr(lprop,ir)*g1s*g2s(kgasp,ielem)
```

```
      dc(6,6)=dc(5,5)
   22 continue
c***********calculate db([dc] in note)
      do 24 ib=1,6
      do 23 jb=1,6
      db(ib,jb)=0.0
   23 continue
   24 continue
      db(2,2)=0.5*g1t*g2t(kgasp,ielem)*dft*dsit(kgasp,ielem)
      db(2,3)=-d4*db(2,2)
      db(3,2)=db(2,3)
      db(3,3)=db(2,2)
      db(4,4)=2.*(db(2,2)-db(2,3))
      db(5,5)=0.5*g1s*g2s(kgasp,ielem)*dfs*dsis(kgasp,ielem)
      db(6,6)=db(5,5)
c******************calculate d(i,j)
      do 26 i=1,6
      do 26 j=1,6
       d(i,j)=0.
  26  continue
      do 41 irow=1,6
      do 41 jcol=1,6
       d(irow,jcol)=ds(irow,jcol)+dc(irow,jcol)+db(irow,jcol)
  41  continue
c      write(26,5000)((d(i,j),j=1,6),i=1,6)
 5000 format('d:',/6f12.4)
c****************** inverse of dmatrix ******************
      do 14 i=1,n
       do 15 j=1,n
          dinv(i,j)=0.
   15    continue
      dinv(i,i)=1
   14 continue
      call ludcmp(d,6,6,indx,c)
      do 16 j=1,n
       call lubksb(d,6,6,indx,dinv(1,j))
   16 continue
c     rotate the stress-strain matrix to global coordinates
c
c    t(transpose) * d(material)
c
      do 60 irow=1,6
      do 50 icol=1,6
      td(irow,icol)=0.0
      do 48 ik=1,6
      td(irow,icol) = td(irow,icol)+tinv(irow,ik)*dinv(ik,icol)
   48 continue
   50 continue
   60 continue
c
c     t(transpose) * dinv(material) * t
c
      do 80 irow = 1,6
      do 75 icol = 1,6
      dmatx(irow,icol) = 0.0
      do 70 in =1,6
      dmatx(irow,icol) = dmatx(irow,icol)
     , + td(irow,in)*tbar(in,icol)
   70 continue
   75 continue
   80 continue
c*****************calculate e(i)**********************
```

```
c       do 200 kgasp=1,lint
        do 82 i=1,6
        p(i)=0.
  82   continue
        qs1(kgasp,ielem)=0.
        qs2(kgasp,ielem)=0.
        qs3(kgasp,ielem)=0.
        qs4(kgasp,ielem)=0.
        qs5(kgasp,ielem)=0.
        qs6(kgasp,ielem)=0.
       do 86 ir=1,nr
        qr1(ir,kgasp,ielem)=0.
        qr2(ir,kgasp,ielem)=0.
        qr3(ir,kgasp,ielem)=0.
        qr4(ir,kgasp,ielem)=0.
        qr5(ir,kgasp,ielem)=0.
        qr6(ir,kgasp,ielem)=0.
   86 continue
c********* evaluate each individual term at gauss point ******
c*********[gr]([h1] in note)
      dgt=0.0
      dgs=0.0
      dqt=0.0
      dqt1=0.0
      dqt2=0.0
      dqs1=0.0
      dqs2=0.0
      do 90 ir=1,nr
      qr1(ir,kgasp,ielem)=0.
      qr2(ir,kgasp,ielem)=exp(-lamtt(lprop,ir)*dsitold(kgasp,ielem))*
     /  qr2old(ir,kgasp,ielem)+gamto(ir,kgasp,ielem)*
     /  (g2told(kgasp,ielem)*sigold(2,kgasp,ielem)-
     /  g2tvold(kgasp,ielem)*sigvold(2,kgasp,ielem))
      qr3(ir,kgasp,ielem)=exp(-lamtt(lprop,ir)*dsitold(kgasp,ielem))*
     /  qr3old(ir,kgasp,ielem)+gamto(ir,kgasp,ielem)*
     /  (g2told(kgasp,ielem)*sigold(3,kgasp,ielem)-
     /  g2tvold(kgasp,ielem)*sigvold(3,kgasp,ielem))
      qr4(ir,kgasp,ielem)=exp(-lamtt(lprop,ir)*dsitold(kgasp,ielem))*
     /  qr4old(ir,kgasp,ielem)+gamto(ir,kgasp,ielem)*
     /  (g2told(kgasp,ielem)*sigold(4,kgasp,ielem)-
     /  g2tvold(kgasp,ielem)*sigvold(4,kgasp,ielem))
      qr5(ir,kgasp,ielem)=exp(-lamts(lprop,ir)*dsisold(kgasp,ielem))*
     /  qr5old(ir,kgasp,ielem)+gamso(ir,kgasp,ielem)*
     /  (g2sold(kgasp,ielem)*sigold(5,kgasp,ielem)-
     /  g2svold(kgasp,ielem)*sigvold(5,kgasp,ielem))
      qr6(ir,kgasp,ielem)=exp(-lamts(lprop,ir)*dsisold(kgasp,ielem))*
     /  qr6old(ir,kgasp,ielem)+gamso(ir,kgasp,ielem)*
     /  (g2sold(kgasp,ielem)*sigold(6,kgasp,ielem)-
     /  g2svold(kgasp,ielem)*sigvold(6,kgasp,ielem))
      dgt=dgt+dtr(lprop,ir)*gamt(ir,kgasp,ielem)
      dgs=dgs+dsr(lprop,ir)*gams(ir,kgasp,ielem)
      dqt1=dqt1+(exp(-lamtt(lprop,ir)*dsit(kgasp,ielem)))*
     /      (qr2(ir,kgasp,ielem)-d4*qr3(ir,kgasp,ielem))*dtr(lprop,ir)
      dqt =dqt +(exp(-lamtt(lprop,ir)*dsit(kgasp,ielem)))*
     /      (d4*qr2(ir,kgasp,ielem)-qr3(ir,kgasp,ielem))*dtr(lprop,ir)
      dqt2=dqt2+(exp(-lamtt(lprop,ir)*dsit(kgasp,ielem)))*
     /      qr4(ir,kgasp,ielem)*dtr(lprop,ir)
      dqs1=dqs1+(exp(-lamts(lprop,ir)*dsis(kgasp,ielem)))*
     /      qr5(ir,kgasp,ielem)*dsr(lprop,ir)
      dqs2=dqs2+(exp(-lamts(lprop,ir)*dsis(kgasp,ielem)))*
     /      qr6(ir,kgasp,ielem)*dsr(lprop,ir)
   90 continue
```

```
c************( qf in note )*****************************
      qs1(kgasp,ielem)=0.
      qs2(kgasp,ielem)=qs2old(kgasp,ielem)+0.5*
     /  dsitold(kgasp,ielem)*(g2told(kgasp,ielem)*
     /  sigold(2,kgasp,ielem)+g2tvold(kgasp,ielem)*
     /  sigvold(2,kgasp,ielem))
      qs3(kgasp,ielem)=qs3old(kgasp,ielem)+0.5*
     /  dsitold(kgasp,ielem)*(g2told(kgasp,ielem)*
     /  sigold(3,kgasp,ielem)+g2tvold(kgasp,ielem)*
     /  sigvold(3,kgasp,ielem))
      qs4(kgasp,ielem)=qs4old(kgasp,ielem)+0.5*
     /  dsitold(kgasp,ielem)*(g2told(kgasp,ielem)*
     /  sigold(4,kgasp,ielem)+g2tvold(kgasp,ielem)*
     /  sigvold(4,kgasp,ielem))
      qs5(kgasp,ielem)=qs5old(kgasp,ielem)+0.5*
     /  dsisold(kgasp,ielem)*(g2sold(kgasp,ielem)*
     /  sigold(5,kgasp,ielem)+g2svold(kgasp,ielem)*
     /  sigvold(5,kgasp,ielem))
      qs6(kgasp,ielem)=qs6old(kgasp,ielem)+0.5*
     /  dsisold(kgasp,ielem)*(g2sold(kgasp,ielem)*
     /  sigold(6,kgasp,ielem)+g2svold(kgasp,ielem)*
     /  sigvold(6,kgasp,ielem))
      p(1)=0.
      p(2)=-g1t*(dqt1-dft*(qs2(kgasp,ielem)-d4*qs3(kgasp,ielem)))
      p(3)=g1t*(dqt + dft*(qs3(kgasp,ielem)-d4*qs2(kgasp,ielem)))
      p(4)=-2.*(1.+d4)*g1t*(dqt2-dft*qs4(kgasp,ielem))
      p(5)=-g1s*(dqs1-dfs*qs5(kgasp,ielem))
      p(6)=-g1s*(dqs2-dfs*qs6(kgasp,ielem))
c
      do 96 jcol=1,6
      ep(jcol)=0.0
   96 continue
   97 continue
      ep(1)=0.0
      ep(2)=g2told(kgasp,ielem)*g1t*(dgt+0.5*dft*
     ,      dsit(kgasp,ielem))*(sigold(2,kgasp,ielem)-
     ,      d4*sigold(3,kgasp,ielem))+p(2)
      ep(3)=g2told(kgasp,ielem)*g1t*(-dgt-0.5*dft*
     ,      dsit(kgasp,ielem))*(d4*sigold(2,kgasp,ielem)-
     ,      sigold(3,kgasp,ielem))+p(3)
      ep(4)=g2told(kgasp,ielem)*g1t*(2.*(1.+d4)*dgt+2.0*dft*
     ,      dsit(kgasp,ielem))*sigold(4,kgasp,ielem)+p(4)
      ep(5)=g2sold(kgasp,ielem)*g1s*(dgs+0.5*dfs*dsis(kgasp,ielem))
     ,      *sigold(5,kgasp,ielem)+p(5)
      ep(6)=g2sold(kgasp,ielem)*g1s*(dgs+0.5*dfs*dsis(kgasp,ielem))
     ,      *sigold(6,kgasp,ielem)+p(6)
******** calzculate t(transform)[i,j]*dinv[i,j]*(e(i)+alf*dtemp)
      do 100 i=1,6
  100    tde(i,kgasp,ielem)=0.
c      do 105 i=1,6
      do 105 j=1,6
       tde(1,kgasp,ielem)=td(1,j)*(ep(j)+alf(j)*(temp-tempo))
     ,                    +tde(1,kgasp,ielem)
       tde(2,kgasp,ielem)=td(2,j)*(ep(j)+alf(j)*(temp-tempo))
     ,                    +tde(2,kgasp,ielem)
       tde(3,kgasp,ielem)=td(3,j)*(ep(j)+alf(j)*(temp-tempo))
     ,                    +tde(3,kgasp,ielem)
       tde(4,kgasp,ielem)=td(4,j)*(ep(j)+alf(j)*(temp-tempo))
     ,                    +tde(4,kgasp,ielem)
       tde(5,kgasp,ielem)=td(5,j)*(ep(j)+alf(j)*(temp-tempo))
     ,                    +tde(5,kgasp,ielem)
       tde(6,kgasp,ielem)=td(6,j)*(ep(j)+alf(j)*(temp-tempo))
```

```
     ,                    +tde(6,kgasp,ielem)
  105 continue
c  100 continue
c  200 continue
 2000 format (10h**** error,/
     1        43h zero length between nodes 1-2 in element (,i4,1h))
c
      return
      end
      subroutine ludcmp(a,n,np,indx,d)
      implicit double precision (a-h,o-z)
c       real a,vv,d,sum,aamax,dum
       integer indx,n,np
      parameter (nmax=100,tiny=1.0e-20)
      dimension a(np,np),indx(n),vv(nmax)
      d=1.
      do 12 i=1,n
      aamax=0.
      do 11 j=1,n
        if (abs(a(i,j)).gt.aamax) aamax=abs(a(i,j))
11      continue
      if (aamax.eq.0.) pause 'singular matrix.'
      vv(i)=1./aamax
12    continue
      do 19 j=1,n
      if (j.gt.1) then
        do 14 i=1,j-1
          sum=a(i,j)
          if (i.gt.1)then
            do 13 k=1,i-1
            sum=sum-a(i,k)*a(k,j)
13            continue
            a(i,j)=sum
          endif
14        continue
      endif
      aamax=0.
      do 16 i=j,n
        sum=a(i,j)
        if (j.gt.1)then
          do 15 k=1,j-1
            sum=sum-a(i,k)*a(k,j)
15          continue
          a(i,j)=sum
        endif
        dum=vv(i)*abs(sum)
        if (dum.ge.aamax) then
          imax=i
          aamax=dum
        endif
16      continue
      if (j.ne.imax)then
        do 17 k=1,n
          dum=a(imax,k)
          a(imax,k)=a(j,k)
          a(j,k)=dum
17        continue
        d=-d
        vv(imax)=vv(j)
      endif
      indx(j)=imax
      if(j.ne.n)then
```

```
          if(a(j,j).eq.0.)a(j,j)=tiny
          dum=1./a(j,j)
          do 18 i=j+1,n
            a(i,j)=a(i,j)*dum
18          continue
        endif
19      continue
        if(a(n,n).eq.0.)a(n,n)=tiny
        return
        end
        subroutine lubksb(a,n,np,indx,b)
        implicit double precision (a-h,o-z)
c        real a,sum,b
        integer indx
        dimension a(np,np),indx(n),b(n)
        ii=0
        do 12 i=1,n
        ll=indx(i)
        sum=b(ll)
        b(ll)=b(i)
        if (ii.ne.0)then
          do 11 j=ii,i-1
            sum=sum-a(i,j)*b(j)
11          continue
        else if (sum.ne.0.) then
          ii=i
        endif
        b(i)=sum
12      continue
        do 14 i=n,1,-1
        sum=b(i)
        if(i.lt.n)then
          do 13 j=i+1,n
            sum=sum-a(i,j)*b(j)
13          continue
        endif
        b(i)=sum/a(i,i)
14      continue
        return
        end
        subroutine shap(r,s,t)
        include 'pres.inc'
        include 'globe.inc'
c***dfngt***s****************
        shape(1)=(1.0-r)*(1.0-s)*(1.0-t)
       1           *(-r-s-t-2.0)/8.0
        shape(2)=(1.0-(r)**2)*(1.0-s)*(1.0-t)/4.0
        shape(3)=(1.0+r)*(1.0-s)*(1.0-t)
       1          *(r-s-t-2.0)/8.0
        shape(4)=(1.0+r)*(1.0-(s)**2)*(1.0-t)/4.0
        shape(5)=(1.0+r)*(1.0+s)*(1.0-t)
       1          *(r+s-t-2.0)/8.0
        shape(6)=(1.0-(r)**2)*(1.0+s)*(1.0-t)/4.0
        shape(7)=(1.0-r)*(1.0+s)*(1.0-t)
       1           *(-r+s-t-2.0)/8.0
        shape(8)=(1.0-r)*(1.0-(s)**2)*(1.0-t)/4.0
        shape(9)=(1.0-r)*(1.0-s)*(1.0-(t)**2)/4.0
        shape(10)=(1.0+r)*(1.0-s)*(1.0-(t)**2)/4.0
        shape(11)=(1.0+r)*(1.0+s)*(1.0-(t)**2)/4.0
        shape(12)=(1.0-r)*(1.0+s)*(1.0-(t)**2)/4.0
        shape(13)=(1.0-r)*(1.0-s)*(1.0+t)
       1           *(-r-s+t-2.0)/8.0
```

```
      shape(14)=(1.0-(r)**2)*(1.0-s)*(1.0+t)/4.0
      shape(15)=(1.0+r)*(1.0-s)*(1.0+t)
     1          *(r-s+t-2.0)/8.0
      shape(16)=(1.0+r)*(1.0-(s)**2)*(1.0+t)/4.0
      shape(17)=(1.0+r)*(1.0+s)*(1.0+t)
     1          *(r+s+t-2.0)/8.0
      shape(18)=(1.0-(r)**2)*(1.0+s)*(1.0+t)/4.0
      shape(19)=(1.0-r)*(1.0+s)*(1.0+t)
     1           *(-r+s+t-2.0)/8.0
      shape(20)=(1.0-r)*(1.0-(s)**2)*(1.0+t)/4.0
c
      deriv(1,1)=(1.0-s)*(1.0-t)
     1            *(1.0+2.0*r+s+t)/8.0
      deriv(1,2)=-r*(1.0-s)*(1.0-t)/2.0
      deriv(1,3)=(1.0-s)*(1.0-t)
     1            *(-1.0+2.0*r-s-t)/8.0
      deriv(1,4)=(1.0-(s)**2)*(1.0-t)/4.0
      deriv(1,5)=(1.0+s)*(1.0-t)
     1            *(-1.0+2.0*r+s-t)/8.0
      deriv(1,6)=-r*(1.0+s)*(1.0-t)/2.0
      deriv(1,7)=(1.0+s)*(1.0-t)
     1            *(1.0+2.0*r-s+t)/8.0
      deriv(1,8)=-(1.0-(s)**2)*(1.0-t)/4.0
      deriv(1,9)=-(1.0-s)*(1.0-(t)**2)/4.0
      deriv(1,10)=(1.0-s)*(1.0-(t)**2)/4.0
      deriv(1,11)=(1.0+s)*(1.0-(t)**2)/4.0
      deriv(1,12)=-(1.0+s)*(1.0-(t)**2)/4.0
      deriv(1,13)=(1.0-s)*(1.0+t)
     1            *(1.0+2.0*r+s-t)/8.0
      deriv(1,14)=-r*(1.0-s)*(1.0+t)/2.0
      deriv(1,15)=(1.0-s)*(1.0+t)
     1            *(-1.0+2.0*r-s+t)/8.0
      deriv(1,16)=(1.0-(s)**2)*(1.0+t)/4.0
      deriv(1,17)=(1.0+s)*(1.0+t)
     1            *(-1.0+2.0*r+s+t)/8.0
      deriv(1,18)=-r*(1.0+s)*(1.0+t)/2.0
      deriv(1,19)=(1.0+s)*(1.0+t)
     1            *(1.0+2.0*r-s-t)/8.0
      deriv(1,20)=-(1.0-(s)**2)*(1.0+t)/4.0
c
      deriv(2,1)=(1.0-r)*(1.0-t)
     1            *(1.0+r+2.0*s+t)/8.0
      deriv(2,2)=-(1.0-(r)**2)*(1.0-t)/4.0
      deriv(2,3)=(1.0+r)*(1.0-t)
     1            *(1.0-r+2.0*s+t)/8.0
      deriv(2,4)=-s*(1.0+r)*(1.0-t)/2.0
      deriv(2,5)=(1.0+r)*(1.0-t)
     1            *(-1.0+r+2.0*s-t)/8.0
      deriv(2,6)=(1.0-(r)**2)*(1.0-t)/4.0
      deriv(2,7)=(1.0-r)*(1.0-t)
     1            *(-1.0-r+2.0*s-t)/8.0
      deriv(2,8)=-s*(1.0-r)*(1.0-t)/2.0
      deriv(2,9)=-(1.0-r)*(1.0-(t)**2)/4.0
      deriv(2,10)=-(1.0+r)*(1.0-(t)**2)/4.0
      deriv(2,11)=(1.0+r)*(1.0-(t)**2)/4.0
      deriv(2,12)=(1.0-r)*(1.0-(t)**2)/4.0
      deriv(2,13)=(1.0-r)*(1.0+t)
     1            *(1.0+r+2.0*s-t)/8.0
      deriv(2,14)=-(1.0-(r)**2)*(1.0+t)/4.0
      deriv(2,15)=(1.0+r)*(1.0+t)
     1            *(1.0-r+2.0*s-t)/8.0
      deriv(2,16)=-s*(1.0+r)*(1.0+t)/2.0
```

```
      deriv(2,17)=(1.0+r)*(1.0+t)
     1            *(-1.0+r+2.0*s+t)/8.0
      deriv(2,18)=(1.0-(r)**2)*(1.0+t)/4.0
      deriv(2,19)=(1.0-r)*(1.0+t)
     1             *(-1.0-r+2.0*s+t)/8.0
      deriv(2,20)=-s*(1.0-r)*(1.0+t)/2.0
c
      deriv(3,1)=(1.0-r)*(1.0-s)
     1            *(1.0+r+s+2.0*t)/8.0
      deriv(3,2)=-(1.0-(r)**2)*(1.0-s)/4.0
      deriv(3,3)=(1.0+r)*(1.0-s)
     1            *(1.0-r+s+2.0*t)/8.0
      deriv(3,4)=-(1.0+r)*(1.0-(s)**2)/4.0
      deriv(3,5)=(1.0+r)*(1.0+s)
     1            *(1.0-r-s+2.0*t)/8.0
      deriv(3,6)=-(1.0-(r)**2)*(1.0+s)/4.0

      deriv(3,7)=(1.0-r)*(1.0+s)
     1             *(1.0+r-s+2.0*t)/8.0
      deriv(3,8)=-(1.0-r)*(1.0-(s)**2)/4.0
      deriv(3,9)=-t*(1.0-r)*(1.0-s)/2.0
      deriv(3,10)=-t*(1.0+r)*(1.0-s)/2.0
      deriv(3,11)=-t*(1.0+r)*(1.0+s)/2.0
      deriv(3,12)=-t*(1.0-r)*(1.0+s)/2.0
      deriv(3,13)=(1.0-r)*(1.0-s)
     1             *(-1.0-r-s+2.0*t)/8.0
      deriv(3,14)=(1.0-(r)**2)*(1.0-s)/4.0
      deriv(3,15)=(1.0+r)*(1.0-s)
     1            *(-1.0+r-s+2.0*t)/8.0
      deriv(3,16)=(1.0+r)*(1.0-(s)**2)/4.0
      deriv(3,17)=(1.0+r)*(1.0+s)
     1            *(-1.0+r+s+2.0*t)/8.0
      deriv(3,18)=(1.0-(r)**2)*(1.0+s)/4.0
      deriv(3,19)=(1.0-r)*(1.0+s)
     1             *(-1.0-r+s+2.0*t)/8.0
      deriv(3,20)=(1.0-r)*(1.0-(s)**2)/4.0
      return
      end
      subroutine jacob(ielem,djacb,kgasp)
      include 'pres.inc'
      include 'globe.inc'
      dimension xjacm(3,3),xjaci(3,3)
c***bm at***[b]*******************
c
c... compute jacobian transformation from x,y,z to rg,sg,tg
c    vj:jacobian array
      do 10 idime=1,ndime
      gpcod(idime,kgasp)=0.0
      do 10 inode=1,nnode
      gpcod(idime,kgasp)=gpcod(idime,kgasp)+
     ,      elcod(idime,inode)*shape(inode)
   10 continue
c***invj***[j]**********************
      do 1301 idime=1,3
      do 1301 jdime=1,3
      xjacm(idime,jdime)=0.0
      do 1301 inode=1,20
      xjacm(idime,jdime)=xjacm(idime,jdime)+
     , deriv(idime,inode)*elcod(jdime,inode)
 1301 continue
c.... inverse of jacobian matrix
      xjaci(1,1)= (xjacm(2,2)*xjacm(3,3)-xjacm(2,3)*xjacm(3,2))
```

```
      xjaci(1,2)=-(xjacm(1,2)*xjacm(3,3)-xjacm(1,3)*xjacm(3,2))
      xjaci(1,3)= (xjacm(1,2)*xjacm(2,3)-xjacm(1,3)*xjacm(2,2))
      xjaci(2,1)=-(xjacm(2,1)*xjacm(3,3)-xjacm(2,3)*xjacm(3,1))
      xjaci(2,2)= (xjacm(1,1)*xjacm(3,3)-xjacm(1,3)*xjacm(3,1))
      xjaci(2,3)=-(xjacm(1,1)*xjacm(2,3)-xjacm(1,3)*xjacm(2,1))
      xjaci(3,1)= (xjacm(2,1)*xjacm(3,2)-xjacm(2,2)*xjacm(3,1))
      xjaci(3,2)=-(xjacm(1,1)*xjacm(3,2)-xjacm(1,2)*xjacm(3,1))
      xjaci(3,3)= (xjacm(1,1)*xjacm(2,2)-xjacm(1,2)*xjacm(2,1))
c
c    compute jacobian determinant
c
      djacb=xjacm(1,1)*xjaci(1,1)+xjacm(2,1)*xjaci(1,2)
     1         +xjacm(3,1)*xjaci(1,3)
      if(abs(djacb).gt.0e-30) goto 30
      write(26,900)ielem
  900 format(//,'program halted in jacobm'/,'zero pr megatove area',/
     ,           'element number ',i5)
   30 continue
      do 2000 i=1,3
      do 2000 j=1,3
 2000 xjaci(i,j)=xjaci(i,j)/djacb
c.... compute global derivatives of shape functions
c     cartd:dndx,dndy,dndz
       do 3000 idime=1,ndime
       do 3000 inode=1,nnode
       cartd(idime,inode)=0.
       do 3000 jdime=1,ndime
       cartd(idime,inode)=cartd(idime,inode)+
     ,  xjaci(idime,jdime)*deriv(jdime,inode)
 3000  continue
      return
      end
      subroutine bmatrx
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c .                                                                   .
c .   p r o g r a m                                                   .
c .                                                                   .
c .      evaluates strain-displacement matrix b at point (r,s,t)      .
c .                                                                   .
c .      curvilinear hexahedron 20 nodes                              .
c .                                                                   .
c . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
c
c
      include 'pres.inc'
      include 'globe.inc'
c
c     evaluate b matrix in global (x,y,z) coordinates
c
      do 130 k=1,nnode
      k1=k*3-2
      k2=k*3-1
      k3=k*3
      bmatx(1,k1)=cartd(1,k)
      bmatx(2,k1)=0.0
      bmatx(3,k1)=0.0
      bmatx(4,k1)=0.0
      bmatx(5,k1)=cartd(3,k)
      bmatx(6,k1)=cartd(2,k)
      bmatx(1,k2)=0.0
      bmatx(2,k2)=cartd(2,k)
      bmatx(3,k2)=0.0
```

```
      bmatx(4,k2)=cartd(3,k)
      bmatx(5,k2)=0.0
      bmatx(6,k2)=cartd(1,k)
      bmatx(1,k3)=0.0
      bmatx(2,k3)=0.0
      bmatx(3,k3)=cartd(3,k)
      bmatx(4,k3)=cartd(2,k)
      bmatx(5,k3)=cartd(1,k)
      bmatx(6,k3)=0.0
  130 continue
c
c
      return
c
      end
      subroutine dbe
      include 'pres.inc'
      include 'globe.inc'
c**** calculats d*b
      do 10 istre=1,6
      do 10 ievab=1,nevab
      dbmat(istre,ievab)=0.0
      do 10 jstre=1,6
      dbmat(istre,ievab)=dbmat(istre,ievab)+
     1                   dmatx(istre,jstre)*bmatx(jstre,ievab)
   10 continue
      return
      end
      subroutine front(err)
      parameter (mfron=720,kpoin=2200)
      include 'pres.inc'
      include 'print.inc'
      dimension fixed(3*kpoin),equat(mfron),vecrv(3*kpoin),
     ,gload(mfron),gstif(mfron*(mfron+1)/2),estif(60,60),detau(3*kpoin),
     ,iffix(3*kpoin),nacva(mfron),locel(60),ndest(60)
      include 'globe.inc'
      nfunc(i,j)=(j*j-j)/2+i
c      mfron=100
      mstif=259560
c
c***  interpret fixity data in vector form
c
      ntotv=npoin*ndofn
      do 100 itotv=1,ntotv
      iffix(itotv)=0
      detau(itotv)=0.0
  100 fixed(itotv)=0.0
      do 110 ivfix=1,nvfix
      nloca=(nofix(ivfix)-1)*ndofn
      do 110 idofn=1,ndofn
      ngash=nloca+idofn
      iffix(ngash)=ifpre(ivfix,idofn)
  110 fixed(ngash)=presc(ivfix,idofn)
c
c***  change the sign of the last appearance
c     of each node
c
      do 140 ipoin=1,npoin
      klast=0
      do 130 ielem=1,nelem
      do 120 inode=1,nnode
      if(lnods(ielem,inode).ne.ipoin) go to 120
```

```
      klast=ielem
      nlast=inode
  120 continue
  130 continue
      if(klast.ne.0) lnods(klast,nlast)=-ipoin
  140 continue
c
c***  start by initializing everything that
c     matters to zero
c
      do 150 istif=1,mstif
  150 gstif(istif)=0.0
      do 160 ifron=1,mfron
      gload(ifron)=0.0
      equat(ifron)=0.0
      vecrv(ifron)=0.0
  160 nacva(ifron)=0
c
c***  and prepare for disc reading and writing
c     operations
c
      rewind 1
      rewind 2
      rewind 3
      rewind 4
c
c***  enter main element assembly-reduction loop
c
      nfron=0
      kelva=0
      do 380 ielem=1,nelem
      kevab=0
      read(1) estif
      do 170 inode=1,nnode
      do 170 idofn=1,ndofn
      nposi=(inode-1)*ndofn+idofn
      locno=lnods(ielem,inode)
      if(locno.gt.0) locel(nposi)=(locno-1)*
     , ndofn+idofn
      if(locno.lt.0) locel(nposi)=(locno+1)*
     , ndofn-idofn
  170 continue
c
c***  start by lloking for existing destinations
c
      do 210 ievab=1,nevab
      nikno=iabs(locel(ievab))
      kexis=0
      do 180 ifron=1,nfron
      if(nikno.ne.nacva(ifron)) go to 180
      kevab=kevab+1
      kexis=1
      ndest(kevab)=ifron
  180 continue
      if(kexis.ne.0) go to 210
c
c***  we now seek new empty places for
c     destination vector
c
      do 190 ifron=1,mfron
      if(nacva(ifron).ne.0) go to 190
      nacva(ifron)=nikno
```

```
      kevab=kevab+1
      ndest(kevab)=ifron
      go to 200
  190 continue
c
c***  the new places may demand an increase
c     in current frontwidth
c
  200 if(ndest(kevab).gt.nfron) nfron=ndest(kevab)
  210 continue
c
c***  assemble element loads
c
      do 240 ievab=1,nevab
      idest=ndest(ievab)
      gload(idest)=gload(idest)+eload(ielem,ievab)
c
c***  assemble the element stiffnesses
c     - but not in resolution
c
      if(icase.gt.1) go to 230
      do 220 jevab=1,ievab
      jdest=ndest(jevab)
      ngash=nfunc(idest,jdest)
      ngish=nfunc(jdest,idest)
      if(jdest.ge.idest) gstif(ngash)=
     , gstif(ngash)+estif(ievab,jevab)
      if(jdest.lt.idest) gstif(ngish)=
     ,gstif(ngish)+estif(ievab,jevab)
  220 continue
  230 continue
  240 continue
 5000 format(3f15.8)
c
c***  re-examine each element node, to
c     enquire which can be eliminated
c
      do 370 ievab=1,nevab
      nikno=-locel(ievab)
      if(nikno.le.0) go to 370
c
c***  find positions of variables ready
c     for elimination
c
      do 350 ifron=1,nfron
      if(nacva(ifron).ne.nikno) go to 350
c
c***  extract the coefficients of the
c     new equation for elimination
c
      if(icase.gt.1) go to 260
      do 250 jfron=1,mfron
      if(ifron.lt.jfron) nloca=nfunc(ifron,jfron)
      if(ifron.ge.jfron) nloca=nfunc(jfron,ifron)
      equat(jfron)=gstif(nloca)
  250 gstif(nloca)=0.0
  260 continue
c
c***  and fxtract the corresponding right
c     hand sides
c
      eqrhs=gload(ifron)
```

```
      gload(ifron)=0.0
      kelva=kelva+1
c
c***  write equations to disc or to tape
c
      if(icase.gt.1) go to 270
      write(2) equat,eqrhs,ifron,nikno
      go to 280
  270 write(4) eqrhs
      read(2) equat,dummy,idumm,nikno
  280 continue
c
c***  deal with pivot
c
      pivot=equat(ifron)
      equat(ifron)=0.0
c
c***  enquire whether prfsfnt variable is
c     free or prescribed
c
      if(iffix(nikno).eq.0) go to 300
c
c***  deal with a prescribed deflection
c
      do 290 jfron=1,nfron
  290 gload(jfron)=gload(jfron)-fixed(nikno)*
     , equat(jfron)
      go to 340
c
c***  eliminate a free variable - deal with
c     the right hand side first
c
  300 do 330 jfron=1,nfron
      gload(jfron)=gload(jfron)-equat(jfron)*
     , eqrhs/pivot
c
c***  now deal with the coefficients in core
c
      if(icase.gt.1) go to 320
      if(equat(jfron).eq.0.0) go to 330
      nloca=nfunc(0,jfron)
      do 310 lfron=1,jfron
      ngash=lfron+nloca
  310 gstif(ngash)=gstif(ngash)-equat(jfron)*
     , equat(lfron)/pivot
  320 continue
  330 continue
  340 equat(ifron)=pivot
c
c***  record the new vacant space, and reduce
c     frontwidth if possible
c
      nacva(ifron)=0
      go to 360
c
c***  complete the element loop in the forward
c     elimination
c
  350 continue
  360 if(nacva(nfron).ne.0) go to 370
      nfron=nfron-1
      if(nfron.gt.0) go to 360
```

```
  370 continue
  380 continue
c
c***  enter back-substitution phase, loop
c     backwards through variables
c
      do 410 ielva=1,kelva
c
c***  read a new equation
c
      backspace 2
      read(2) equat,eqrhs,ifron,nikno
      backspace 2
      if(icase.eq.1) go to 390
      backspace 4
      read(4) eqrhs
      backspace 4
  390 continue
c
c***  prepare to back-substitute from th
c     current equation
c
      pivot=equat(ifron)
      if(iffix(nikno).eq.1) vecrv(ifron)=
     , fixed(nikno)
      if(iffix(nikno).eq.0) equat(ifron)=0.0
c
c***  back-substitute in the current equation
c
      do 400 jfron=1,mfron
  400 eqrhs=eqrhs-vecrv(jfron)*equat(jfron)
c
c***  put the final values where they belong
c
      if(iffix(nikno).eq.0) vecrv(ifron)=
     , eqrhs/pivot
      if(iffix(nikno).eq.1) fixed(nikno)=-eqrhs
      detau(nikno)=vecrv(ifron)
      detau(nikno)=detau(nikno)-asdis(nikno)
      asdis(nikno)=asdis(nikno)+detau(nikno)
  410 continue
c****** convergence iteration ***********************
c****** calculate error
      err1=0.
      err2=0.
      do 402 i=1,ntotv
        err1=err1+(detau(i))**2
        err2=err2+(asdis(i))**2
  402 continue
        err=sqrt(err1/err2)
      if(err.lt.error) then
C
c      write(11,900)
  900 format(1ho,5x,13hdisplacements)
c  420 write(11,910)
  910 format(1h0,5x,4hnode,5x,7hx-disp.,
     , 7x,7hy-disp.,7x,7hz-disp.)
  440 continue
      do 450 ipoin=1,npoin
      ngash=ipoin*ndofn
      ngish=ngash-ndofn+1
      do 455 i=1,nodis
```

```
      if(nodisp(i).eq.ipoin) then
      write(11,920) ipoin,(asdis(igash),igash=
     , ngish,ngash)
      endif
  455 continue
  450 continue
  920 format(i10,3e14.6)
c       write(11,925)
  925 format(1h0,5x,9hreactions)
c  460 write(11,935)
  935 format(1h0,5x,4hnode,5x,7hx-force,7x,
     , 7hy-force,7x,7hz-force)
  480 continue
      do 510 ipoin=1,npoin
      nloca=(ipoin-1)*ndofn
      do 490 idofn=1,ndofn
      ngush=nloca+idofn
      if(iffix(ngush).gt.0) go to 500
  490 continue
      go to 510
  500 ngash=nloca+ndofn
      ngish=nloca+1
c      write(11,945) ipoin,(fixed(igash),igash=
c     , ngish,ngash)
  510 continue
  945 format(i10,3e14.6)
      endif
c***  post front = reset all element connection
c     numbers to positive values for subsequent
c      use in stress calculation
c
      do 520 ielem=1,nelem
      do 520 inode=1,nnode
  520 lnods(ielem,inode)=iabs(lnods(ielem,inode))
      return
      end
      subroutine stress(time,err,kf,it)
c*************************************************************
c*   program                                                 *
c*   1)  to calculate strains at each gauss point/element.   *
c*   2)  to calculate stresss at each gauss point/element.   *
c*   3)  to perform failure analysis.                        *
c*   4)  to save strains and stresses into output files.      *
c*************************************************************
      include 'pres.inc'
      double precision ks12,ks23
      include 'globe.inc'
      include 'crpin.inc'
      include 'print.inc'
      include 'tmatx.inc'
      include 'fail.inc'
      dimension eldis(3,20),strain(6,27,400),sigb(6,27,400)
     ,,sig1(400),sig2(400),sig3(400),sig4(400)
     ,,sig5(400),sig6(400),strne(6,400),strn(6),sige1(400),sige2(400),
     ,sige3(400),sige4(400),sige5(400),sige6(400),f(400),tf(400)
      common/dg/dgmatx(6,6,27)
      common/fail2/dforc,newfe
      rewind 3
      rewind 7
c      rewind 8
      write(13,*)'calculating stress...'
c
```

```
c***  loop over each element
c
      do 5 i=1,6
  5     strn(i)=0.0
      do 15 ielem=1,nelem
      do 15 j=1,6
       strne(j,ielem)=0.0
       sig1(ielem)=0.0
       sig2(ielem)=0.0
       sig3(ielem)=0.0
       sig4(ielem)=0.0
       sig5(ielem)=0.0
       sig6(ielem)=0.0
       f(ielem)=0.0
       sige1(ielem)=0.0
       sige2(ielem)=0.0
       sige3(ielem)=0.0
       sige4(ielem)=0.0
       sige5(ielem)=0.0
       sige6(ielem)=0.0
  15  continue
      do 60 ielem=1,nelem
      lprop=matno(ielem)
      itype=props(lprop,8)
      x=props(lprop,37)
      a=props(lprop,38)
      b=props(lprop,39)
      ks12=props(lprop,40)
      ks23=props(lprop,41)
c
c***  read the stress matrix, sampling point
c     coordinates for the element
c
      read(3) dsmatx,gpcod
      read(7) smatx
c      read(8) dgmatx
c
c***  identify the displacements of the
c     element nodal points
c
      do 10 inode=1,nnode
      lnode=lnods(ielem,inode)
      nposn=(lnode-1)*ndofn
      do 10 idofn=1,ndofn
      nposn=nposn+1
      eldis(idofn,inode)=asdis(nposn)
   10 continue
      ngasp=0
c
      if(err.lt.error) then
c      write(26,910) ielem
c      write(26,985)
c***  enter loops over each sampling point
c      write(26,900)
      endif
c
      ngasp=0
      do 50 igaus=1,ngaus
      do 50 jgaus=1,ngaus
      do 50 kgaus=1,ngaus
      ngasp=ngasp+1
c
```

```
c***  compute the cartesian stress components
c     at the sampling points
c
      do 20 istre=1,nstre
      strain(istre,ngasp,ielem)=0.0
      kgash=0
      do 20 inode=1,nnode
      do 20 idofn=1,ndofn
      kgash=kgash+1
c**** {strain}=[b]{u}*******************************
      strain(istre,ngasp,ielem)=strain(istre,ngasp,ielem)
     , +smatx(istre,kgash,ngasp)*eldis(idofn,inode)
   20 continue
c**** calculate stresses in x,y,z coordinates (sigb)******
      do 30 istre=1,nstre
      sigb(istre,ngasp,ielem)=0.
      kgash=0.
      do 30 inode=1,nnode
      do 30 idofn=1,ndofn
      kgash=kgash+1
      sigb(istre,ngasp,ielem)=sigb(istre,ngasp,ielem)
     , +dsmatx(istre,kgash,ngasp)*eldis(idofn,inode)
c      do 30 jstre=1,nstre
c      sigb(istre,ngasp,ielem)=sigb(istre,ngasp,ielem)
c     ,  +dgmatx(istre,jstre,ngasp)*strain(jstre,ngasp,ielem)
   30 continue
 8000 format(6e12.4)
c      if(err.lt.error)then
c         write(26,*)'sigb'
c         write(26,916)ngasp,(sigb(istr1,ngasp,ielem),istr1=1,nstre)
c      endif
      if(itype.eq.3) go to 400
      do 32 istre=1,6
   32    sigb(istre,ngasp,ielem)=sigb(istre,ngasp,ielem)
     ,    -tde(istre,ngasp,ielem)
  400 continue
c      rem=amod(real(time),50.)
c      if(err.lt.error.and.rem.eq.0.0) then
       if(err.lt.error) then
c        do 1041 i=1,notrs
c          if(noeltrs(i).eq.ielem) then
c          write(10,914) ielem,ngasp,(gpcod(idime,ngasp),idime=1,ndime)
c          write(10,916)ngasp,(sigb(istr1,ngasp,ielem),istr1=1,nstre)
c          endif
c 1041 continue
      endif
      do 35 istre=1,nstre
   35    sig(istre,ngasp,ielem)=0.
c***** calculate stress components in 1,2,3 coordinates(fiber direction)
c       write(13,*)'calculate linear stress...'
      do 36 k=1,6
       sig(1,ngasp,ielem)=sig(1,ngasp,ielem)+to(1,k,ielem)
     /                       *sigb(k,ngasp,ielem)
       sig(2,ngasp,ielem)=sig(2,ngasp,ielem)+to(2,k,ielem)
     /                       *sigb(k,ngasp,ielem)
       sig(3,ngasp,ielem)=sig(3,ngasp,ielem)+to(3,k,ielem)
     /                       *sigb(k,ngasp,ielem)
       sig(4,ngasp,ielem)=sig(4,ngasp,ielem)+to(4,k,ielem)
     /                       *sigb(k,ngasp,ielem)
       sig(5,ngasp,ielem)=sig(5,ngasp,ielem)+to(5,k,ielem)
     /                       *sigb(k,ngasp,ielem)
       sig(6,ngasp,ielem)=sig(6,ngasp,ielem)+to(6,k,ielem)
```

```
     /                        *sigb(k,ngasp,ielem)
   36 continue
c
c***  output the stresses
c
c       if(err.lt.error) then
c       write(26,*)ngasp,(gpcod(idime,ngasp),idime=1,ndime)
c       write(26,915)ngasp,(strain(istrl,ngasp,ielem),istrl=1,nstre)
c       write(26,916)ngasp,(sig(istrl,ngasp,ielem),istrl=1,nstre)
c       endif
   50 continue
      if(err.lt.error) then
      lint=ngaus*ngaus*ngaus
      do 70 k=1,lint
       strne(1,ielem)=strne(1,ielem)+strain(1,k,ielem)
       strne(2,ielem)=strne(2,ielem)+strain(2,k,ielem)
       strne(3,ielem)=strne(3,ielem)+strain(3,k,ielem)
       strne(4,ielem)=strne(4,ielem)+strain(4,k,ielem)
       strne(5,ielem)=strne(5,ielem)+strain(5,k,ielem)
       strne(6,ielem)=strne(6,ielem)+strain(6,k,ielem)
c
       sig1(ielem)=sig1(ielem)+sig(1,k,ielem)
       sig2(ielem)=sig2(ielem)+sig(2,k,ielem)
       sig3(ielem)=sig3(ielem)+sig(3,k,ielem)
       sig4(ielem)=sig4(ielem)+sig(4,k,ielem)
       sig5(ielem)=sig5(ielem)+sig(5,k,ielem)
       sig6(ielem)=sig6(ielem)+sig(6,k,ielem)
c         if(itype.eq.3) then
       sige1(ielem)=sige1(ielem)+sigb(1,k,ielem)
       sige2(ielem)=sige2(ielem)+sigb(2,k,ielem)
       sige3(ielem)=sige3(ielem)+sigb(3,k,ielem)
       sige4(ielem)=sige4(ielem)+sigb(4,k,ielem)
       sige5(ielem)=sige5(ielem)+sigb(5,k,ielem)
       sige6(ielem)=sige6(ielem)+sigb(6,k,ielem)
c         endif
  70  continue
c
c******* average element strains and stresses
c
       strne(1,ielem)=strne(1,ielem)/real(lint)
       strne(2,ielem)=strne(2,ielem)/real(lint)
       strne(3,ielem)=strne(3,ielem)/real(lint)
       strne(4,ielem)=strne(4,ielem)/real(lint)
       strne(5,ielem)=strne(5,ielem)/real(lint)
       strne(6,ielem)=strne(6,ielem)/real(lint)
c
       sig1(ielem)=sig1(ielem)/real(lint)
       sig2(ielem)=sig2(ielem)/real(lint)
       sig3(ielem)=sig3(ielem)/real(lint)
       sig4(ielem)=sig4(ielem)/real(lint)
       sig5(ielem)=sig5(ielem)/real(lint)
       sig6(ielem)=sig6(ielem)/real(lint)
c         if(itype.eq.3)then
       sige1(ielem)=sige1(ielem)/real(lint)
       sige2(ielem)=sige2(ielem)/real(lint)
       sige3(ielem)=sige3(ielem)/real(lint)
       sige4(ielem)=sige4(ielem)/real(lint)
       sige5(ielem)=sige5(ielem)/real(lint)
       sige6(ielem)=sige6(ielem)/real(lint)
c         endif
c
c***************failure analysis model*****************************
```

```
      if(itype.eq.3.or.ifail.eq.0.or.kfact(ielem).ge.1) go to 1000
      f(ielem)=0.
      f(ielem)=dsqrt((sig2(ielem)**2+sig3(ielem)**2-2.*sig2(ielem)*
     ,sig3(ielem)+sig4(ielem)**2/ks23**2+sig5(ielem)**2/ks12**2+
     ,sig6(ielem)**2/ks12**2)/(1.0-sig1(ielem)**2/x**2+sig1(ielem)*
     ,sig2(ielem)/x**2+sig1(ielem)*sig3(ielem)/x**2-sig2(ielem)*
     ,sig3(ielem)/x**2))
      tf(ielem)=0.
      tf(ielem)=10.**((a-f(ielem))/b)
      tfact(ielem)=tfact(ielem)+dt/tf(ielem)
       write(13,919)ielem,tfact(ielem)
        if(tfact(ielem).ge.1.0) then
          write(26,920)ielem,time,dforc
          kfact(ielem)=kfact(ielem)+1
        endif
 1000 continue
c
c..... print element strain and stress values into files.....
c
      do 1050 i=1,notrn
      if(noeltrn(i).eq.ielem) then
      write(12,917)ielem,(strne(istr1,ielem),istr1=1,nstre)
      endif
1050  continue
      do 1060 i=1,notrs
        if(noeltrs(i).eq.ielem) then
c           write(10,915)ielem,sig1(ielem),sig2(ielem),sig3(ielem)
c     ,      ,sig4(ielem),sig5(ielem),sig6(ielem)
         write(10,915)ielem,sige1(ielem),sige2(ielem),sige3(ielem)
     ,      ,sige4(ielem),sige5(ielem),sige6(ielem)
         endif
1060  continue
      endif
c
   60 continue
c
c..... calculate the average strains of all layers.
c
      if(err.lt.error)then
        do 100 ielem=1,nelem
           strn(1)=strn(1)+strne(1,ielem)
           strn(2)=strn(2)+strne(2,ielem)
           strn(3)=strn(3)+strne(3,ielem)
           strn(4)=strn(4)+strne(4,ielem)
           strn(5)=strn(5)+strne(5,ielem)
           strn(6)=strn(6)+strne(6,ielem)
  100     continue
        do 90 j=1,6
           strn(j)=strn(j)/real(nelem)
   90     continue
c            write(26,986)
c            write(26,918)time,(strn(istr1),istr1=1,nstre)
           write(26,918)(strn(istr1),istr1=1,nstre)
      endif
c******** count how many elements have failed *******
      if(itype.eq.3.or.ifail.eq.0) go to 103
      if(err.lt.error)then
      nfe=0
      do 102 ielem=1,nelem
       if(kfact(ielem).eq.1) then
          nfe=nfe+1
       else
```

```
          go to 102
        endif
  102 continue
c***** increase external load to accelerate failure process ****
      if(nfe.eq.newfe) then
      dforc=dforc+0.2
      elseif(nfe.gt.newfe) then
        newfe=nfe
        dforc=dforc
      else
        go to 103
      endif
  103 continue
c
c....... if all plys have failed, return to the main program.
       if(nfe.eq.nelem) then
       kf=1
       go to 2000
       endif
      endif
c*****************************************************
  900 format(/,10x,8hstresses,/)
  910 format(4x,12helement no.=,i5)
  914 format(i3,i3,3f10.4)
  915 format(i3,2e13.5,3e12.4,e12.5)
  916 format(i3,6e12.4,/)
  917 format(i3,2e13.5,3e12.4,e12.5)
c  918 format(f6.1,1x,6e12.4)
  918 format(6e13.5)
  919 format(4x,'tfact( ',i3,' )=',f12.4)
  920 format(4x,'element ',i3,' failed at time=',f10.2,' dforc=',f6.1)
  985 format(4x,13hstrain/stress)
  986 format (/,4x'11-strain',3x,
     1 '22-strain',3x,'33-strain',3x,'23-strain',3x,'13-strain',3x,
     2 '12-strain')
  905 format (4x,'11-stress',3x,
     1 '22-stress',3x,'33-stress',3x,'23-stress',3x,'13-stress',3x,
     2 '12-stress'/)
  906 format(5x,5e13.5)
 2000 continue
      return
      end
```