PROBABILISTIC FINITE ELEMENT METHODS FOR

THE EVALUATION OF WOOD COMPOSITES

By

WEI YANG

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN CIVIL ENGINEERING

WASHINGTON STATE UNIVERSITY
Department of Civil and Environmental Engineering

May 2000

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of WEI YANG find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGMENT

I would like to thank my advisor, Dr. William Cofer, for his guidance, help and encouragement throughout the course of this study.

I also wish to thank Dr. Kenneth Fridley and Dr. John Hermanson for their support and advice in completing this thesis. Special thanks are also extended to Mr. Tom Weber for his help in computer-related contents.

I am grateful for being sponsered by the Office of Naval Research, ONR 332, under the direction of Mr. James J. Kelly. I wish I had the space here to thank everyone who has supported and encouraged me, THANKS! to all of you.

PROBABILISTIC FINITE ELEMENT METHODS FOR

THE EVALUATION OF WOOD COMPOSITES

Abstract

by Wei Yang, M.S.
Washington State University
May 2000

Chair: William Cofer

Composite materials are increasingly being used in aerospace, marine, and automotive structures. The application of composite materials to engineering fields has spurred a major effort to analyze structural components made from them. Even though composite materials provide unique advantages over their constituent counterparts, they also present complex and challenging problems to analysts and designers.

The research of this thesis was directed to randomly distributed particulate-reinforced wood composites (RDPR WC), which have highly heterogeneous microstructures. Apart from spatial variations in reinforcement distribution and arrangement, the properties of the reinforcement particles are themselves orthotropic and highly variable. Computational limitations have generally dictated that two-scale models be explored in the study of particulate-reinforced composites. The effects of microscale heterogeneity may be taken into account via the formulation of a material law to be applied in a macroscale random model. To represent the random characteristics of RDPR WC with varying reinforcement ratio and reinforcement orientation, finite element

methods and probability methods were combined in the study. The material properties were then modelled as basic variables, each of which had an assumed probability distribution. The analyses gave a reliability estimate for structures and sensitivity estimates for basic variables, which are valuable results for both designers and manufacturers. In this thesis, probabilistic finite element methods were successfully introduced into the study and analysis of wood composites. Two-scale models kept the computation in a manageable range and, at the same time, were able to represent the complex microstructure of particulate-reinforced composites. The results of two simple numerical examples also provided some practical ways to improve wood products in manufacture.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

Composite materials are increasingly being used in aerospace, marine, and automotive structures. The application of composite materials to engineering fields has spurred a major effort to analyze structural components made from them. Even though composite materials provide unique advantages over their constituent counterparts, they also present complex and challenging problems to analysts and designers. With the rapid growth in the use of composite materials in many commercial products ranging from sports equipment to high-performance aircraft, literature on composite materials has proliferated.

Much attention has been focused on the finite element analysis of fiber-reinforced or particulate-reinforced metal matrix composites. Originally, they were studied by using self-consistent schemes which are based on a model consisting of a single inclusion in an infinite matrix (Duva, 1984). Deformation and failure within discontinuously reinforced composites have been investigated using axisymmetric "unit cell" models with reinforcement in different shapes (Bao, Hutchinson and McMeeking, 1991). In this approach, the composite is modelled as a periodic array of identical revolutional unit cells, each containing a reinforcement particle dimensioned to represent the overall reinforcement volume fraction. Three-dimensional arrays are also used in analyzing longitudinally aligned cylinder models, both transversely aligned and staggered (Levy and Papazian, 1990). Both models captured the basic features of the stress-strain curve of metal matrix composites, but overestimated the initial yielding. The difference between the results from the two models is diminished as the particle aspect ratio (i.e. the

ratio of cylinder length, l, over cylinder diameter, d) is decreased. A micromechanically based analytical continuum model was developed by Zhu and Zbib (1993). There, the composite was also idealized as a uniformly distributed periodic array of unit cells. Each unit cell consisted of a rigid inclusion surrounded by a plastically deforming material. The results were in good agreement with numerical analyses carried out earlier.

In practice, however, particulate-reinforced composites have highly heterogeneous microstructures. Apart from spatial variations in reinforcement distribution and arrangement, the properties of the reinforcement particles are themselves orthotropic and highly variable. Modelling a material as a self-consistent model or periodic array of identical cells implies that only one single inclusion exists in the matrix or that every single cell in the material behaves identically. This is physically unrealistic, and ignores the effects from particulate clustering and orientation. To address these issues, "random unit cell" models were developed (Brockenbrough, Suresh and Wieneche, 1991). Here, a number of particles are randomly distributed in a unit cell. But this approach is computationally intensive. High numbers of elements are required even in relatively coarse meshes.

Computational limitations have generally dictated that two-scale models be explored in the study of particulate-reinforced composites. The effects of microscale heterogeneity may be taken into account via the formulation of a material law to be applied in a macroscale random unit cell model. Leggoe, Mammoli, Bush and Hu (1998) modelled deformation in particulate-reinforced metal matrix composites with locally varying reinforcement volume fraction using a two-scale finite element approach. The

analysis results of axisymmetric unit cell models were used to define the constitutive response of mesoscale regions possessing varying volume fractions. Macroscale response was then investigated using two- and three-dimensional "random arrays" of finite elements, in which element properties were randomly assigned in line with a Gaussian distribution.

In comparison with that of metal-matrix composites, research for wood composites is underdeveloped, even though a variety of particulate-reinforced wood composites are available for use as an alternative to plain wood. Thus, the objective of this thesis is to apply the above finite element analysis methods and probabilistic methods, which will be introduced in the following, to studying randomly distributed particulate-reinforced wood composites (RDPR WC). Variations need to be introduced into the material properties at different locations to represent the characteristics of randomly reinforced composites by assigning random values to the reinforcement volume fraction and the reinforcement orientation for each element group. The analysis results showing the effects from the randomness of different properties should be able to provide designers and manufacturers some general idea about the relative importance of these factors.

Until fairly recently there has been a tendency for structural engineering to be dominated by deterministic thinking, characterized in design calculations by the use of specified minimum material properties. It is now widely recognized, however, that some risk of unacceptable structural performance must be tolerated. The main object of structural design is therefore to ensure, at an acceptable level of probability, that a structure will not become unfit for its intended purpose at any time during its specified design life. There are various sources of uncertainty in structural design. External loads,

environmental factors, material properties, and geometry of structures all possess some inherent variability.

Structural analyses which combine the finite element method and the theory of probability or statistics were initiated in the 1970's. Such analysis techniques are usually denoted as probabilistic or stochastic finite element analysis. There are three basic types of stochastic finite element methods: simulation methods, perturbation methods, and reliability methods (Liu and Der Kiureghian, 1989). The direct Monte Carlo simulation method was used in many early works in stochastic finite element analysis (Astill, Nosseir and Shinozuka, 1972; Shinozuka and Lenoe, 1976). This method has the advantage that it is adaptable to all types of problems and the results can be obtained to any desired accuracy. However, for practical problems with many random or small probabilities, this procedure is usually too expensive, since a large number of solutions are needed to obtain reliable results. The perturbation method involves the first- or second-order Taylor series expansion of the terms in the governing equation of the structure around the mean values of the random variables. Handa and Anderson (1981) applied this method to a beam and a truss structure to estimate the first two statistical moments of structural displacements and stresses. Hisada and Nakagiri (1980 and 1981) used the first- and second-order perturbation methods on linear and nonlinear problems. However, this method yields satisfactory results only when the variations of the random variables are small. Furthermore, since the perturbation methods are unable to give reliability evaluations, they are not suitable for use in the safety assessment of structures. The reliability methods aim at evaluating the failure probabilities of structures. The first-order reliability method was used by Der Kiureghian and Ke (1985, 1988) for static

analysis of linear structures with random properties. Furthermore, Liu and Der Kiureghian (1988) used the first- and second-order reliability methods for static analysis of geometrically nonlinear trusses.

In the following chapters, the two-scale finite element reliability methods were studied and applied to RDPR WC's. In Chapter 2, general formulations of the structural reliability problem and the first- and second-order reliability methods (FORM and SORM) are discussed. Chapter 3 contains the finite element formulation of the general 2-D isoparametric 8-node element, which was used in the elastic analyses in this thesis. Chapter 4 discusses the microscale models carried out in ABAQUS and the finite element reliability program CALREL-FEMCOD which was used to combine the probability calculation and the macroscale finite element analysis. In Chapter 5, the CALREL-FEMCOD program and the constitutive law from ABAQUS models were applied to two numerical examples. Conclusions and discussions are given in Chapter 6.

CHAPTER TWO

STRUCTURAL RELIABILITY

BASIC THEORY

Methods of structural reliability analysis, employing concepts of probability and statistics, account for the uncertain nature of structures and their environments. In recent years, robust and accurate techniques for computing probabilities of failure have been developed.

Two fundamental assumptions are made for the structural reliability problems considered here. First, the structure may fail in any of a finite number of modes, and with respect to each mode it is either in a safe state or in a failure state. For each mode, the state of the structure is determined by the value of a limit-state function. Second, the uncertainties in the structure and its environment are assumed to be modelled by random variables. These may include the variabilities in the material properties, the structural shape, and the external loads. In this thesis, only the first category, the effects of uncertainties in materials, will be investigated. The set of basic random variables describing these uncertainties is represented by a vector $V = [V_1, \cdots, V_n]^T$.

The limit states of a structure are usually defined in terms of structural responses and response thresholds. The response thresholds may be included in the vector V if they are also random in nature. The structural responses, denoted by a vector S, are functions of the basic random variables, i.e.,

$$S = S(V) \tag{2-1}$$

The mapping from V to S is denoted the *mechanical transformation* of the structure. Only in some special cases is it possible to get an analytical expression for this transformation. Otherwise, the transformation is in an algorithmic form, such as a finite element code.

An explicit function of V and S, $g(V,S)$, is usually used to represent the limit-state function for a given failure mode. However, it is still an implicit function of the basic variables V if the mechanical transformation in Eq. 2-1 is taken into account. Thus, the probability of failure in the mode of interest is given by

$$P_f = \int_F f_V(v)\,dv \tag{2-2}$$

$$F \equiv \{g(v,s) \le 0\} \tag{2-3}$$

where $f_V(v)$ is the joint probability density function (PDF) of V. The set $g(v,s) \le 0$ defines the failure state. Even though the expression for the failure probability appears simple, in practice it is almost impossible to perform the multi-fold integral directly, either analytically or numerically. This is because the limit-state function is implicit of V in the integration domain and the number of basic random variables is often very large. Hence, alternative methods are needed. The difference between the various reliability methods lies in the approach used for evaluating the multi-fold integral in Eq. 2-2.

Analytical integration of the convolution integral in Eq. 2-2 is possible only for some very special cases of limited practical interest (Liu and Der Kiureghian, 1989). Numerical solutions like the direct Monte Carlo simulation techniques involve ‘sampling

randomly' to simulate artificially a large number of experiments and to observe the results. In the case of analysis for structural reliability, this means that a sample value $\hat{v}_i$ is arbitrarily chosen to represent each random variable V_i. The limit state function $g(\hat{v},\hat{s}) = 0$ is then checked. If the limit state is violated (i.e. $g(\hat{v},\hat{s}) \le 0$), the structure or structural element has 'failed'. The experiment is repeated many times, each time with a randomly chosen vector V of $\hat{v}_i$ values. If N trials are conducted, the probability of failure is approximated by

$$P_f \approx \frac{n(g(\hat{v},\hat{s}) \le 0)}{N} \tag{2-4}$$

Where $n(g(\hat{v},\hat{s}) \le 0)$ denotes the number of trials for the structure failure (i.e. $g(\hat{v},\hat{s}) \le 0$). Obviously, the desired accuracy for P_f is closely related to the number N of trials performed.

On the other hand, rather than use numerical approximations to perform the integration, the first- and second-order reliability methods simplify the probability density function $f_V(\,)$ in Eq. 2-2.

THE FIRST-ORDER RELIABILITY METHOD

In the first-order reliability method (FORM), an approximation of the integral is obtained by replacing the integration boundary with a first-order approximation surface in a transformed space of standard normal variates.

The basic random variables V are transformed into a set of statistically independent, standard normal variates:

$$Y = Y(V)\,. \tag{2-5}$$

This mapping is denoted the *probability transformation.* It exists for continuous random variables and it can be inverted. In terms of the standard normal variates, the limit state function is denoted

$$G(Y) \equiv g(V(Y), S(V(Y)))\,. \tag{2-6}$$

Since the probability transformation is a one-to-one mapping, the probability content in the failure domain is preserved in the standard normal space. Hence, in this space the failure probability may be expressed by

$$P_f = \int_{G(y)\le 0} \phi(y)\, dy\,, \tag{2-7}$$

where $\phi(y)$ denotes the standard normal density of Y.

The approximation is carried out at points of the boundary which have minimum distance to the origin in the standard normal space. These points, known as design points, have maximal probability densities among all point(s) in the failure domain and, hence, their neighborhoods make the dominant contributions to the probability integral. This is also the reason why the surface approximations can be used here. The most time-consuming part of the FORM analysis is the process of finding the design point(s). Usually, a constrained optimization algorithm employing the gradient vector of the limit-state function is used for this purpose (Zhang and Der Kiureghian, 1994).

In FORM, the limit-state suface in the standard normal space is replaced by its tangent hyperplane at the point nearest to the origin (see Fig 2-1). The distance from the origin to the point, denoted β, is known as the reliability index. The first-order estimate of the failure probability is given by

$$P_f = \int_{\nabla G(y^*)(y-y^*)\le 0} \phi(y)\, dy = \Phi(-\beta) \tag{2-8}$$

where $\nabla G(y^*)$ is the gradient of $G(y)$ computed at the design point y^* and $\Phi(\,)$ is the cumulative distribution function (CDF) of the standard normal variate.

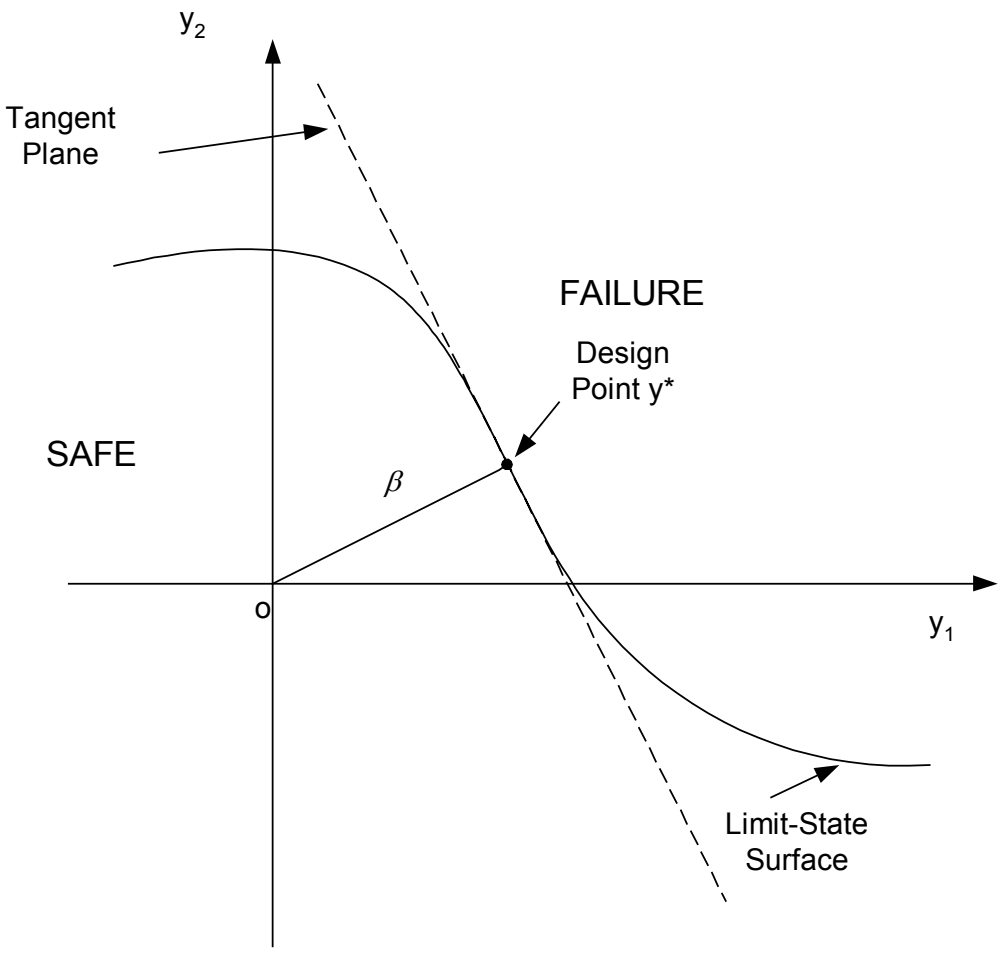


Fig. 2-1 The First-Order Reliability Method

It is possible that the limit-state surface has multiple minimum distance points. In that case, the surface is approximated by a polyhedron and system reliability techniques are employed to improve the first-order probability estimate (Liu and Der Kiureghian, 1989).

THE NORMAL TRANSFORMATION

In general, the transformation from non-normally distributed variables to equivalent standardized normal variables in Eq. 2-5 is not as simple and straightforward as it appears to be. However, once it has been done the resulting normal equivalents can be used directly in probability calculation procedures.

Consider a vector of uniformly distributed random variables, denoted R. Let these be the intermediaries between the random variables in the original space, represented by the vector X, and the standardized normal variables, represented by the vector Y. Provided the necessary data for the joint probability distribution function $F_{\mathrm{x}}(\mathrm{x})$ and its conditional distributions $F_i(x_i \mid x_1, \cdots, x_{i-1})$ are available, the *Rosenblatt Transformation* in n-dimensional space becomes:

$$\begin{aligned} &\Phi(y_1) = r_1 = F_1(x_1) \\ &\Phi(y_2) = r_2 = F_2(x_2 \mid x_1) \\ &\vdots \\ &\Phi(y_n) = r_n = F_n(x_n \mid x_1, \cdots, x_{n-1}) \end{aligned} \tag{2-9}$$

where $\Phi(\,)$ is the standard normal cumulative distribution function for Y and $F_i(x_i \mid x_1, \cdots, x_{i-1})$ is the conditional cumulative distribution function for the random variable X_i (Melchers, 1987).

Then, the component random variables x_i can be obtained from

$$\begin{aligned} x_1 &= F_1^{-1}[\Phi(y_1)] \\ x_2 &= F_2^{-1}[\Phi(y_2)|x_1] \\ &\vdots \\ x_n &= F_n^{-1}[\Phi(y_n)|x_1,\cdots,x_{n-1}] \end{aligned} \tag{2-10}$$

Before the above transformations can be incorporated into an iteration algorithm, it is necessary to determine the transformation of the limit-state function from $G(x)=0$ to $g(y)=0$. A probability density function defined in the $\mathbf{x}$ space is transformed to the $\mathbf{y}$ space

$$G(\mathbf{x}) = g(\mathbf{y})|J| \tag{2-11}$$

where the Jacobian $|J|$ has elements $j_{ij} = \partial y_i / \partial x_j$; The differential may be evaluated by

$$\frac{\partial y_i}{\partial x_j} = \frac{1}{\phi(y_i)} \frac{\partial F_i(x_i|x_1,\cdots,x_{i-1})}{\partial x_j} \tag{2-12}$$

In practice, the necessary data to allow $f_{\mathrm{x}}(\mathrm{x})$ to be described completely may not be available. If only marginal cumulative probability distribution functions $F_{X_i}(),i=1,\cdots,n$ and the correlation matrix $P=\{\rho_{ij}\}$ are available, the conditional distributions required in Eq. 2-9 are not available. As a result, it is now not possible to apply the Rosenblatt transformation.

Instead, the *Nataf transformation* may be applied to give a set of independent normal random variables. Unlike the Rosenblatt transformation, the Nataf transformation is approximate (Melchers, 1999).

Consider the marginal transformation from the random variables $\mathbf{X} = (X_1, \cdots, X_n)$ in $\mathbf{x}$ space to the standardized normal random variables $\mathbf{Y} = (Y_1, \cdots, Y_n)$ in $\mathbf{y}$ space, given by

$$Y_i = \Phi^{-1}[F_{X_i}(X_i)], \quad i = 1, \cdots, n \tag{2-13}$$

where, as before $\Phi(\,)$ is the standard normal cumulative distribution function. It is assumed that $\mathbf{Y} = (Y_1, \cdots, Y_n)$ is jointly normal and has n-dimensional standard normal probability density function $\phi_n(\mathbf{y}, P')$ with zero means, unit standard deviations and correlation matrix $P' = \{\rho'_{ij}\}$. The Nataf approximation for the joint probability density function $f_{\mathbf{X}}(\,)$ is then given by

$$f_{\mathbf{X}} = \phi_n(\mathbf{y}, P') \cdot |J| \tag{2-14}$$

where the Jacobian $|J|$ is defined by

$$|J| = \frac{\partial(y_1, \cdots, y_n)}{\partial(x_1, \cdots, x_n)} = \frac{f_{X_1}(x_1) \cdot f_{X_2}(x_2) \cdots f_{X_n}(x_n)}{\phi(y_1)\phi(y_2) \cdots \phi(y_n)} \tag{2-15}$$

Thus $f_{\mathbf{X}}(\,)$ is forced to be a unique n-dimensional joint density function defined by Eq. 2-14. The only matter left for resolution is the definition of the correlation matrix $P' = \{\rho'_{ij}\}$ in Eq. 2-14. It would be expected that this should be related to the correlation matrix $P = \{\rho_{ij}\}$ in $\mathbf{x}$ space.

For convenience, introduce the normalized random variables $Z_i = (X_i - \mu_{X_i}) / \sigma_{X_i}$. Then, for any two random variables the correlation between the X_i can be stated as

$$\rho_{ij} = \frac{\mathrm{cov}[X_i X_j]}{\sigma_{X_i}\sigma_{X_j}} = E[Z_i Z_j] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} z_i z_j \phi(y_i, y_j, \rho'_{ij})\, dy_i dy_j \tag{2-16}$$

From this expression the terms in the correlation matrix $P' = \{\rho'_{ij}\}$ can be determined for each pair of marginal distributions with known $P = \{\rho_{ij}\}$. Clearly it will be an iterative and rather tedious process. To ease the burden, Liu and Der Kiureghian (1986) have produced empirical, approximate expressions for the ratio

$$R = \frac{\rho'_{ij}}{\rho_{ij}} \tag{2-17}$$

Based on the fact that R falls into the range from 0.9 to 1.1 with the exception of combinations involving the shifted exponential distribution, Der Kiureghian and Liu (1986) further suggested that for many problems it is sufficient to use $P = \{\rho_{ij}\}$ directly as $P' = \{\rho'_{ij}\}$. It will still be able to give satisfactory results because of the fact that the structural reliability can rarely be determined with high precision in practice.

The above Nataf transformation has several useful features. First, it is applicable to an arbitrary number of random variables with prescribed marginals and covariances. Second, the resulting FORM approximations are invariant of the ordering of the basic variables. Third, the required transformation is computationally much simpler than the transformation in Eq. 2-10 when the conditional distributions are specified (Liu and Der Kiureghian, 1989).

THE SECOND-ORDER RELIABILITY METHOD

In some cases, if the limit-state surface has significant curvature it may not be accurate or desirable to approximate the limit-state surface by a linear surface through a Taylor series expansion. Furthermore, even if each limit-state function is linear in the original space, a non-linear limit state may result when the reliability problem is transformed from the original space to the standard normal space.

In the second-order reliability method (SORM), the limit-state surface is replaced by a second-order surface fitted to the design point (See Fig. 2-2). The SORM attempts to fit a paraboloid to the actual surface. It includes two sub-categories: the curvature-fitted paraboloid and the point-fitted paraboloid. The point-fitting method has the important advantage of being insensitive to the noise on the limit-state surface, which may arise when the limit-state function is in an algorithmic form (Liu and Der Kiureghian, 1989). This advantage is of particular interest in the finite element reliability methods, which are used in the present study.

The details of formulation of the SORM are beyond the scope of this thesis.

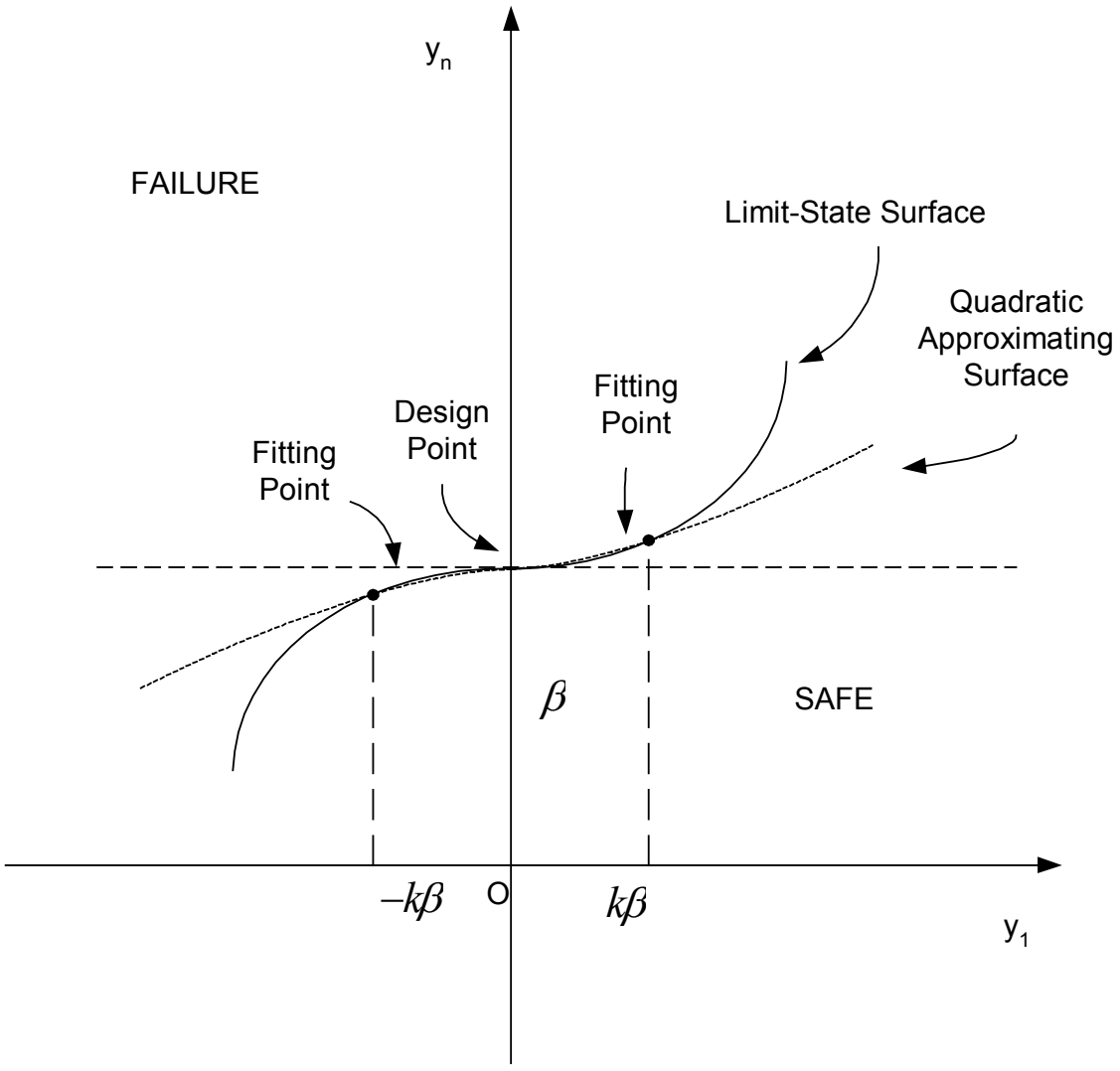


Fig. 2-2 The Second-Order Reliability Method

CHAPTER THREE

EIGHT-NODE ELEMENT FORMULATION

INTRODUCTION

The finite element method is firmly established as a powerful and popular analysis tool. It is applied to many different problems of continua and widely used for various applications in structural mechanics. The element that will be used in the present study is the eight-node isoparametric 2-D element, which needs to be added to the general-purpose finite element program FEMCOD.

The term "isoparametric" means "same parameters" and is explained as follows. Because either displacements or coordinates can be interpolated from nodal values,

1. The displacements $[u\ v]$ at any point in the element can be defined from nodal degrees of freedom (d.o.f.) $\{d\}$, i.e., $[u\ v]^T = [N]\{d\}$.

2. The coordinates $[x\ y]$ at any point in the element can be defined from nodal coordinates $\{c\}$, i.e., $[x\ y]^T = [\widetilde{N}]\{c\}$.

An element is *isoparametric* if $[N]$ and $[\widetilde{N}]$ are identical. If $[N]$ is of higher degree than $[\widetilde{N}]$, the element is called *subparametric*, but if $[N]$ is of lower degree than $[\widetilde{N}]$, the element is called *superparametric* (Cook, Malkus and Plesha, 1989).

The isoparametric formulation makes it possible to generate elements that are nonrectangular and have curved sides. In formulating isoparametric elements, a natural coordinate system must be used (system $\xi\eta$ in Fig. 3-1). Displacements are expressed in terms of natural coordinates. However, to compute the strains, the displacements must be

differentiated with respect to global coordinates x and y. Accordingly, a transformation matrix must be invoked. In addition, integrations must be done numerically rather than analytically if elements are nonrectangular. Although closed-form integrations are possible in some special cases, expressions tend to be lengthy, tedious to work out, and therefore more subject to errors of algebra and coding than numerical integration (Cook, Malkus and Plesha, 1989).

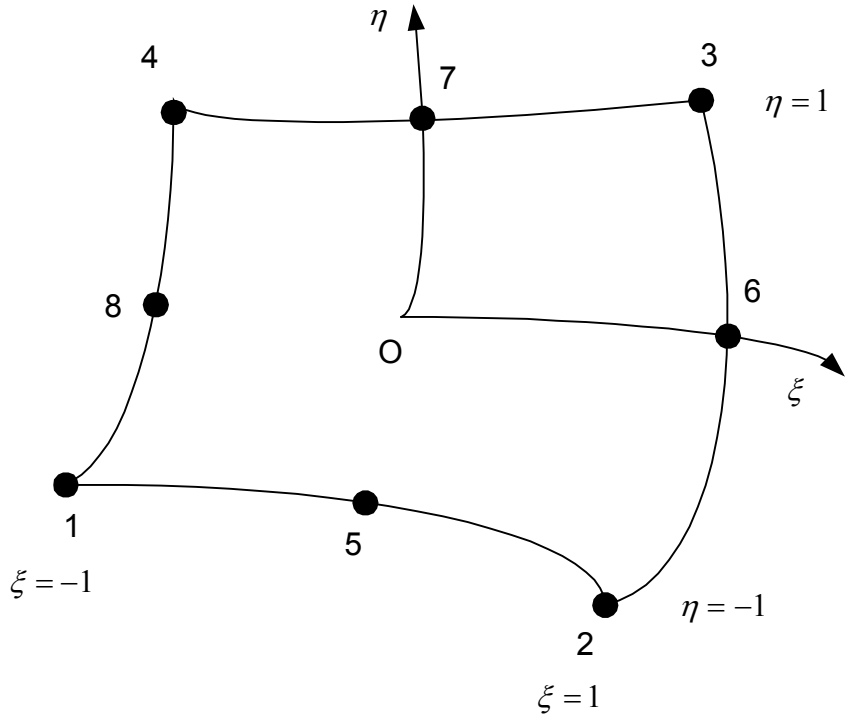


Fig. 3-1 Eight-node isoparametric elements and natural coordinates

8-NODE ISOPARAMETRIC 2-D AXISYMMETRIC ELEMENT

The formulation of axisymmetric 2-D 8-node isoparametric elements is derived first in this section. It is then modified to include the ability to analyze plane-strain and plane-stress problems.

Isoparametric coordinates in a plane are shown in Fig. 3-1. For an eight-node element, axes ξ and η need not be orthogonal, and neither do they need be parallel to the x axis or the y axis. Sides of the element are at $\xi = \pm 1$ and at $\eta = \pm 1$, each of which includes three nodes. Two of the side nodes are at $\xi = 0$ and two are at $\eta = 0$. Sides of an undeformed element may be straight lines or quadratic curves.

In this isoparametric formulation, the geometric mapping and the displacement interpolation are defined by

$$\begin{Bmatrix} r \\ z \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & \cdots & N_8 & 0 \\ 0 & N_1 & 0 & N_2 & \cdots & 0 & N_8 \end{bmatrix} \begin{Bmatrix} r_1 \\ z_1 \\ r_2 \\ z_2 \\ \vdots \\ r_8 \\ z_8 \end{Bmatrix} = \underline{\underline{N}} \cdot \underline{c} \tag{3-1}$$

and

$$\begin{Bmatrix} u \\ w \end{Bmatrix} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & \cdots & N_8 & 0 \\ 0 & N_1 & 0 & N_2 & \cdots & 0 & N_8 \end{bmatrix} \begin{Bmatrix} u_1 \\ w_1 \\ u_2 \\ w_2 \\ \vdots \\ u_8 \\ w_8 \end{Bmatrix} = \underline{\underline{N}} \cdot \underline{d} \tag{3-2}$$

where $\underline{\underline{N}}$ is the *shape function* matrix and $\underline{d}$ and $\underline{c}$ are the displacement vector and the coordinate vector, respectively. The notation "$\underline{\underline{\ }}$" and "$\underline{\ }$" refer to matrices and vectors, which will be used in the remainder of this chapter.

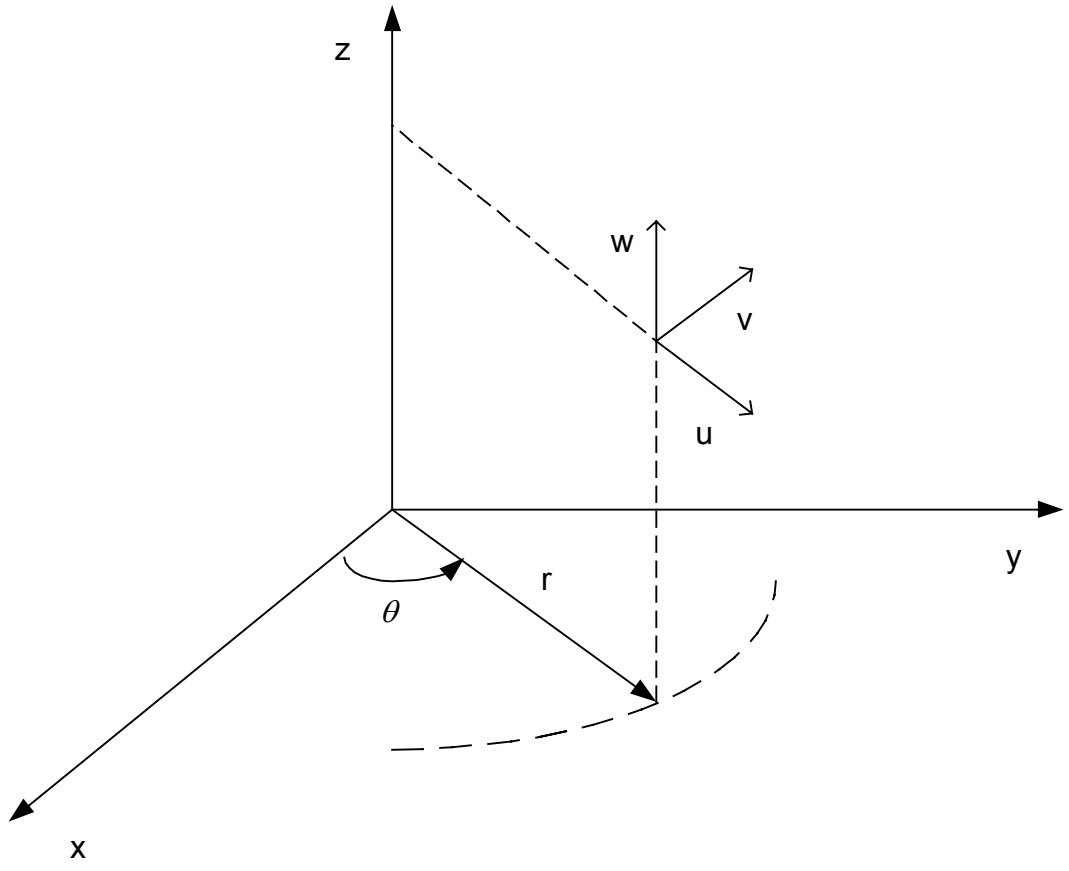


Fig. 3-2 Displacements at any point in the cylindrical coordinates

For a general 3-D problem in the cylindrical coordinate system as shown in Fig. 3-2, the strain-displacement relations at an arbitrary point in an element are

$$\begin{aligned}
\varepsilon_r &= \frac{\partial u}{\partial r} \\
\varepsilon_\theta &= \frac{u}{r} + \frac{1}{r}\frac{\partial v}{\partial \theta} \\
\varepsilon_z &= \frac{\partial w}{\partial z} \\
\gamma_{r\theta} &= \frac{1}{r}\frac{\partial u}{\partial \theta} + \frac{\partial v}{\partial r} - \frac{v}{r} \\
\gamma_{rz} &= \frac{\partial u}{\partial z} + \frac{\partial w}{\partial r} \\
\gamma_{\theta z} &= \frac{\partial v}{\partial z} + \frac{1}{r}\frac{\partial w}{\partial \theta}
\end{aligned} \tag{3-3}$$

where ε's and γ's are strain components in the cylindrical coordinate system at any particular point; u, v and w are respectively the radial, tangential, and axial displacement components at the same point.

If the geometry, material properties, and loads are axially symmetric, the problem is mathematically two-dimensional. That is, if the geometry, support conditions, loads $\underline{R}$, and material property matrix $\underline{\underline{E}}$ are all independent of θ, and if the material either is isotropic or has θ as a principal material direction, the resulting displacements and stresses are independent of θ, the circumferential displacement v is zero, and material points have only u (i.e. radial) and w (i.e. axial) displacement components

$$\frac{\partial u}{\partial \theta} = 0; \qquad \frac{\partial w}{\partial \theta} = 0; \qquad v = 0 \tag{3-4}$$

As a result, the strain-displacement relations are simplified to

$$
\begin{aligned}
\varepsilon_r &= \frac{\partial u}{\partial r} = u_{,r} \\
\varepsilon_\theta &= \frac{u}{r} \\
\varepsilon_z &= \frac{\partial w}{\partial z} = w_{,z} \\
\gamma_{r\theta} &= 0 \\
\gamma_{rz} &= \frac{\partial u}{\partial z} + \frac{\partial w}{\partial r} = u_{,z} + w_{,r} \\
\gamma_{\theta z} &= 0
\end{aligned}
\tag{3-5}
$$

where a comma denotes partial differentiation.

In a matrix format, if the zero items are omitted, Eq. 3-5 can be expressed as

$$
\underline{\varepsilon} = \begin{Bmatrix} \varepsilon_r \\ \varepsilon_\theta \\ \varepsilon_z \\ \gamma_{zr} \end{Bmatrix} = \begin{Bmatrix} u_{,r} \\ u/r \\ w_{,z} \\ u_{,z} + w_{,r} \end{Bmatrix} = \begin{bmatrix} \partial/\partial r & 0 \\ 1/r & 0 \\ 0 & \partial/\partial z \\ \partial/\partial z & \partial/\partial r \end{bmatrix} \begin{Bmatrix} u \\ w \end{Bmatrix} \tag{3-6a}
$$

or $$\underline{\varepsilon} = \underline{\underline{\partial}} \cdot \underline{\underline{N}} \cdot \underline{d} = \underline{\underline{B}} \cdot \underline{d} \tag{3-6b}$$

Since both u and w are functions of ξ and η, the partial derivatives of u and w with respect to r and z in Eq. 3-6 are impossible to compute directly. Thus, Eq. 3-6 needs to be carried out by using the *Chain Rule*.

For the sake of clarity in the derivation, the expression in Eq. 3-6 is separated into three steps

$$
\begin{Bmatrix} \varepsilon_r \\ \varepsilon_\theta \\ \varepsilon_z \\ \gamma_{zr} \end{Bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/r \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}}_{\underline{\underline{F}}} \begin{Bmatrix} u_{,r} \\ u_{,z} \\ w_{,r} \\ w_{,z} \\ u \end{Bmatrix} \tag{3-7}
$$

$$\begin{Bmatrix} u_{,r} \\ u_{,z} \\ w_{,r} \\ w_{,z} \\ u \end{Bmatrix} = \underbrace{\begin{bmatrix} \Gamma & & \\ & \Gamma & \\ & & 1 \end{bmatrix}}_{\underline{\underline{G}}} \begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ w_{,\xi} \\ w_{,\eta} \\ u \end{Bmatrix} \tag{3-8}$$

$$\begin{Bmatrix} u_{,\xi} \\ u_{,\eta} \\ w_{,\xi} \\ w_{,\eta} \\ u \end{Bmatrix} = \underbrace{\begin{bmatrix} N_{1,\xi} & 0 & N_{2,\xi} & 0 & \cdots & N_{8,\xi} & 0 \\ N_{1,\eta} & 0 & N_{2,\eta} & 0 & \cdots & N_{8,\eta} & 0 \\ 0 & N_{1,\xi} & 0 & N_{2,\xi} & \cdots & 0 & N_{8,\xi} \\ 0 & N_{1,\eta} & 0 & N_{2,\eta} & \cdots & 0 & N_{8,\eta} \\ N_1 & 0 & N_2 & 0 & \cdots & N_8 & 0 \end{bmatrix}}_{\underline{\underline{H}}} \begin{Bmatrix} u_1 \\ w_1 \\ u_2 \\ w_2 \\ \vdots \\ u_8 \\ w_8 \end{Bmatrix} \tag{3-9}$$

where the shape functions N_i, $i = 1, 8$ are

$$\begin{aligned} N_1 &= \frac{1}{4}(1-\xi)(1-\eta) - \frac{1}{2}N_5 - \frac{1}{2}N_8 \\ N_2 &= \frac{1}{4}(1+\xi)(1-\eta) - \frac{1}{2}N_5 - \frac{1}{2}N_6 \\ N_3 &= \frac{1}{4}(1+\xi)(1+\eta) - \frac{1}{2}N_6 - \frac{1}{2}N_7 \\ N_4 &= \frac{1}{4}(1-\xi)(1+\eta) - \frac{1}{2}N_7 - \frac{1}{2}N_8 \\ N_5 &= \frac{1}{2}(1-\xi^2)(1-\eta) \\ N_6 &= \frac{1}{2}(1+\xi)(1-\eta^2) \\ N_7 &= \frac{1}{2}(1-\xi^2)(1+\eta) \\ N_8 &= \frac{1}{2}(1-\xi)(1-\eta^2) \end{aligned} \tag{3-10}$$

Then, the only remaining unknown is $[\Gamma]$, which can easily be derived from Eq. 3-8

$$[\Gamma] = \begin{bmatrix} \xi_{,r} & \eta_{,r} \\ \xi_{,z} & \eta_{,z} \end{bmatrix} \tag{3-11}$$

Unfortunately, the partial derivatives of ξ and η with respect to r and z are not directly available from the known information. Therefore, the inverse of $[\Gamma]$ must be written first

$$[J] = \begin{bmatrix} r_{,\xi} & z_{,\xi} \\ r_{,\eta} & z_{,\eta} \end{bmatrix} = [\Gamma]^{-1} \tag{3-12}$$

where $[J]$ is called the *Jacobian matrix*, which is readily available. The matrix $[\Gamma]$ is then generated by inverting $[J]$

$$[\Gamma] = [J]^{-1} = \frac{1}{|J|} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix} \tag{3-13}$$

where $|J|$ is the determinant of the Jacobian matrix.

Obviously, the multiplication of the three equations, Eq. 3-7, Eq. 3-8 and Eq. 3-9, again gives the relation between nodal displacements and strains at the point of interest

$$\underline{\varepsilon} = \underline{\underline{F}} \cdot \underline{\underline{G}} \cdot \underline{\underline{H}} \cdot \underline{d} = \underline{\underline{B}} \cdot \underline{d} \tag{3-14}$$

where $\underline{\underline{B}}$ equals the multiplication of the three matrices $\underline{\underline{F}}$, $\underline{\underline{G}}$ and $\underline{\underline{H}}$.

The element stiffness matrix is formed by

$$\underline{\underline{K}} = \int_V \underline{\underline{B}}^T \cdot \underline{\underline{E}} \cdot \underline{\underline{B}} \, dV \tag{3-15}$$

where $\underline{\underline{E}}$ is the *material stiffness matrix* and dV equals $r\,d\theta\,dr\,dz$ in cylindrical coordinates. Thus, Eq. 3-15 can be expanded to

$$\begin{aligned}\underline{\underline{K}} &= \int\int\int_{-\pi}^{\pi} \underline{\underline{B}}^T \cdot \underline{\underline{E}} \cdot \underline{\underline{B}}\, r\, d\theta\, dr\, dz \\ &= \int_{-1}^{1}\int_{-1}^{1}\int_{-\pi}^{\pi} \underline{\underline{B}}^T \cdot \underline{\underline{E}} \cdot \underline{\underline{B}}\, r\, d\theta \left|J\right| d\xi\, d\eta \\ &= 2\pi\int_{-1}^{1}\int_{-1}^{1} \underline{\underline{B}}^T \cdot \underline{\underline{E}} \cdot \underline{\underline{B}}\, r\, \left|J\right| d\xi\, d\eta\end{aligned} \tag{3-16}$$

In axisymmetric problems, both the load vector $\underline{R}$ and the structure stiffness matrix $\underline{\underline{K}}$ have "2π" as a multiplier in the structural equation

$$\underline{\underline{K}} \cdot \underline{D} = \underline{R} \tag{3-17}$$

As a result, this superfluous multiplier can be avoided by letting integrals for θ integrate only from zero to one radian. With this approach, all integrals are assumed to pertain to one radian (i.e. equivalent loads give forces per radian). However, extra attention must be paid to the axial forces, which need to be modified by dividing by 2π. Thus, Eq. 3-16 becomes

$$\underline{\underline{K}} = \int_{-1}^{1}\int_{-1}^{1} \underline{\underline{B}}^T \cdot \underline{\underline{E}} \cdot \underline{\underline{B}}\, r\, \left|J\right| d\xi\, d\eta \tag{3-18}$$

which will be integrated numerically by using 3×3 Gauss quadrature (i.e. "full integration" for 8-node elements).

SURFACE TRACTIONS

In the structural equation Eq. 3-17, the structure stiffness matrix $\underline{\underline{K}}$ has been formulated in Eq. 3-18 and $\underline{D}$ is the unknown nodal displacement vector we obtained through the solution. In the following, the equivalent nodal load vector $\underline{R}$ is derived for the case of surface tractions.

The equivalent nodal load vector $\underline{R}$ may include several components of nodal forces, initial strains, initial stresses, body forces and surface tractions. The ability to handle nodal forces is part of the formulation of finite element method itself. In the present work, only the surface tractions were considered as an extra loading format in formulating the current 8-node isoparametric elements. In this thesis, the vector "$\underline{q_s}$" is used to refer to the equivalent nodal load vector solely from surface tractions.

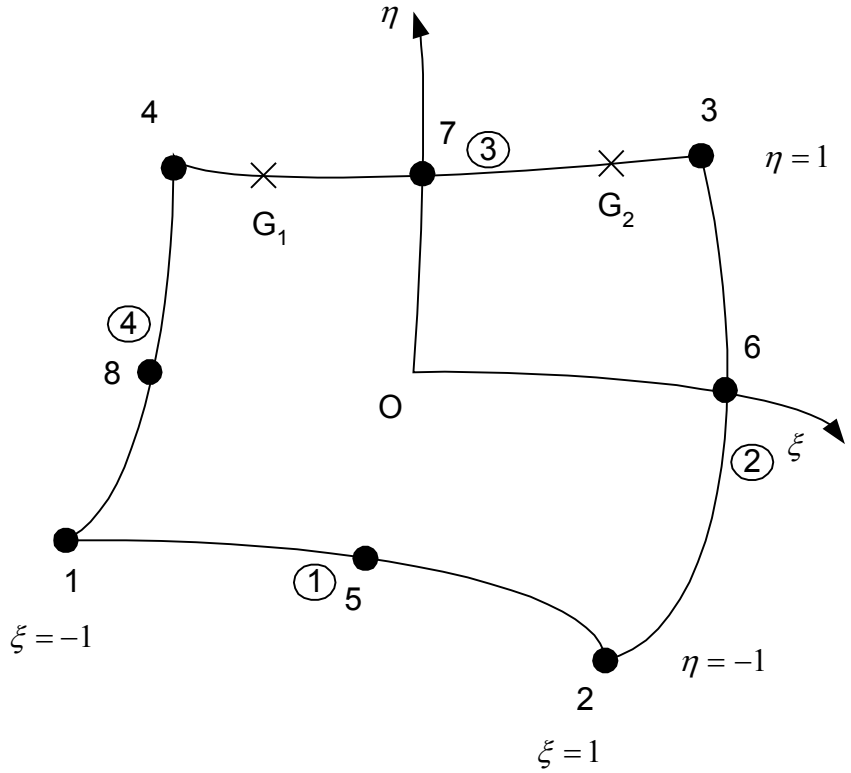


Fig. 3-3 Gauss points for edge 3 in 8-node isoparametric elements

Generally, $\underline{q_s}$ is expressed as

$$\underline{q_s} = \int_s \underline{\underline{N_s}}^T \underline{\Phi}\, ds \tag{3-19}$$

where $\underline{\Phi}$ is the surface traction, $\underline{\underline{N_s}}$ is the shape function along the loading edge, which is different for each edge, and ds is an infinitesimal length along the same edge

$$ds = \sqrt{dx^2 + dy^2} \,. \tag{3-20}$$

The equivalent load vector is formed for the surface tractions on edge 3 as an example (see Fig. 3-3). The shape function on this edge is evaluated at $\eta = 1$

$$\underline{\underline{N_s}}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & N_{3,3} & 0 & N_{4,3} & 0 & 0 & 0 & 0 & 0 & N_{7,3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & N_{3,3} & 0 & N_{4,3} & 0 & 0 & 0 & 0 & 0 & N_{7,3} & 0 & 0 \end{bmatrix} \tag{3-21}$$

where the subscript 3 means the shape functions evaluated at edge 3.

In practice, second-order Gauss quadrature (i.e. two Gauss points, G1 and G2 in Fig. 3-3, will be selected on edge 3 to integrate Eq. 3-19 numerically) is chosen to calculate $\underline{q_s}$. This strategy automatically zeros $\underline{\underline{N}}$ to $\underline{\underline{N_s}}$. Furthermore, since η is constant and equals 1 on edge 3, dx and dy in Eq. 3-20 are only functions of ξ. The relations are available from the Chain Rule,

$$\begin{aligned} dx &= \frac{dx}{d\xi} d\xi = J_{11} d\xi \\ dy &= \frac{dy}{d\xi} d\xi = J_{12} d\xi \end{aligned} \tag{3-22}$$

Generally speaking, surface tractions for a specific problem might be given in two categories: globally (i.e. pressure in the global directions) or locally (i.e. normal pressure σ and shear pressure τ on a surface). If a surface traction is given in the global coordinates, the equivalent nodal load vector can easily be defined by

$$\underline{q_s} = \int_{-1}^{1} \underline{\underline{N_s}}^T \begin{Bmatrix} \Phi_x \\ \Phi_y \end{Bmatrix} r\sqrt{J_{11}{}^2 + J_{12}{}^2}\, d\xi \tag{3-23}$$

where Φ_x and Φ_y are x and y components of the surface traction.

On the other hand, a transformation needs to be used for local pressures, as follows:

$$\begin{aligned} \underline{q_s} &= \int_s \underline{\underline{N_s}}^T \begin{Bmatrix} \Phi_x \\ \Phi_y \end{Bmatrix} ds \\ &= \int_s \underline{\underline{N_s}}^T \begin{Bmatrix} \tau\, ds \cos\beta - \sigma\, ds \sin\beta \\ \sigma\, ds \cos\beta + \tau\, ds \sin\beta \end{Bmatrix} \\ &= \int_s \underline{\underline{N_s}}^T \begin{Bmatrix} \tau\, dx - \sigma\, dy \\ \sigma\, dx + \tau\, dy \end{Bmatrix} \end{aligned} \tag{3-24}$$

where σ and τ are the local normal and shear pressures respectively, and β is the orientation angle of the infinitesimal length. Considering again $dx = J_{11} d\xi$ and $dy = J_{12} d\xi$ in Eq. 3-22, Eq. 3-24 can be simplified to

$$\underline{q_s} = \int_{-1}^{1} \underline{\underline{N_s}}^T \begin{Bmatrix} \tau J_{11} - \sigma J_{12} \\ \sigma J_{11} + \tau J_{12} \end{Bmatrix} r\, d\xi \tag{3-25}$$

Other edges are in a form similar to the above, i.e., at the edge 2,

$$\underline{q_s} = \int_{-1}^{1} \underline{\underline{N_s}}^T \begin{Bmatrix} \Phi_x \\ \Phi_y \end{Bmatrix} r\sqrt{J_{21}{}^2 + J_{22}{}^2}\, d\eta \tag{3-26}$$

or

$$\underline{q_s} = \int_{-1}^{1} \underline{\underline{N_s}}^T \begin{Bmatrix} \tau J_{21} - \sigma J_{22} \\ \sigma J_{21} + \tau J_{22} \end{Bmatrix} r\, d\eta \tag{3-27}$$

STRESS CALCULATION

For isoparametric elements, it often happens that stresses (especially shear stresses) are most accurate at Gauss points of a quadrature rule one order less than that required for "full integration" (i.e. a quadrature rule sufficient to provide the exact integrals of all terms in the element stiffness matrix if the element is undistorted, or in other words, $|J|$ is constant) of the element stiffness matrix. Thus, the 2×2 Gauss quadrature rule is used for the stress calculation of the current 8-node elements.

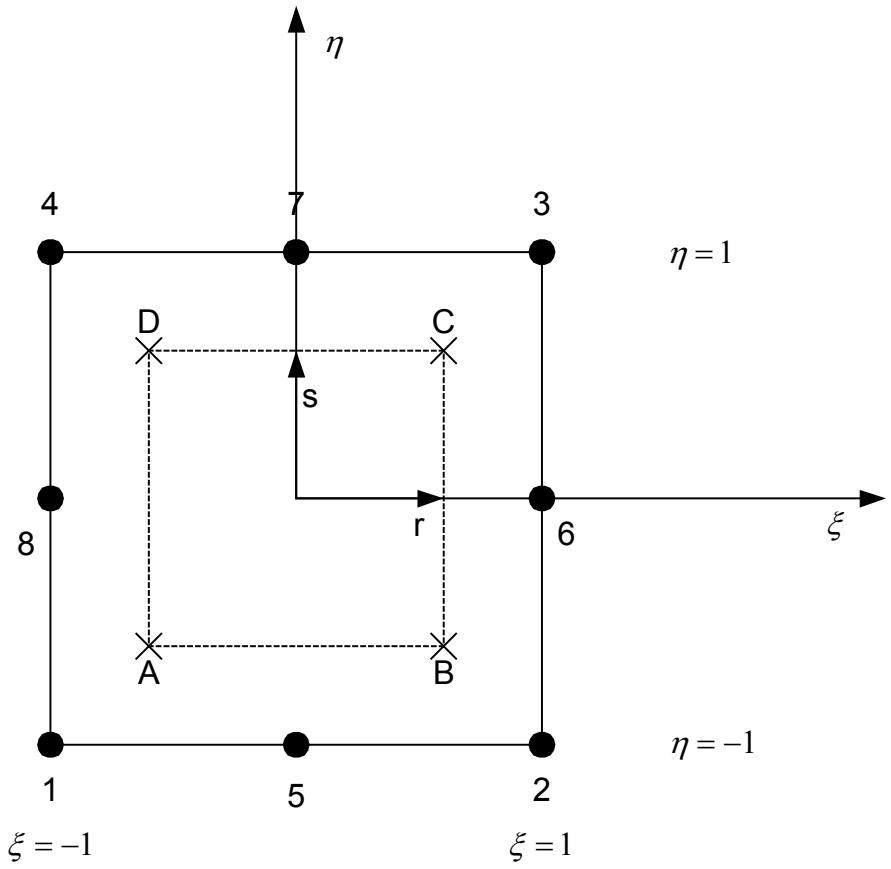


Fig. 3-4 Stress calculation in 8-node isoparametric elements

In Fig. 3-4, points A, B, C and D at $\xi,\eta = \pm\frac{1}{\sqrt{3}}$ are selected to compute the stresses for the element. Then, the results can be extrapolated to the nodal points if needed in the output. To perform the extrapolation about the points $A-D$, 4-node shape functions in terms of r and s are applied to them as if they were nodal points. That is, at any point, P,

$$\underline{\sigma}_P = N_A\underline{\sigma}_A + N_B\underline{\sigma}_B + N_C\underline{\sigma}_C + N_D\underline{\sigma}_D \tag{3-28}$$

where

$$\begin{aligned} N_A &= \frac{1}{4}(1-r)(1-s) \\ N_B &= \frac{1}{4}(1+r)(1-s) \\ N_C &= \frac{1}{4}(1+r)(1+s) \\ N_D &= \frac{1}{4}(1-r)(1+s) \end{aligned} \tag{3-29}$$

For example, at node 3, $\xi = \eta = 1,\ r = s = \sqrt{3}$

$$\begin{aligned} \underline{\sigma}_3 &= \frac{1}{4}(1-\sqrt{3})(1-\sqrt{3})\underline{\sigma}_A + \frac{1}{4}(1+\sqrt{3})(1-\sqrt{3})\underline{\sigma}_B \\ &\quad + \frac{1}{4}(1+\sqrt{3})(1+\sqrt{3})\underline{\sigma}_C + \frac{1}{4}(1-\sqrt{3})(1+\sqrt{3})\underline{\sigma}_D \\ &= 0.134\underline{\sigma}_A - 0.5\underline{\sigma}_B + 1.866\underline{\sigma}_C - 0.5\underline{\sigma}_D \end{aligned}$$

MODIFICATION FOR PLANE-STRAIN AND PLANE-STRESS APPLICATION

Only two minor changes need to be made to make the existing axisymmetric formulation useable for plane-strain and plane-stress problems.

Firstly, the $\underline{\underline{E}}$ matrices are different for each case. For isotropic materials, axisymmetric, plane-strain, and plane-stress problems have different stress-strain relations as expressed in Eq. 3-30, 3-31 and 3-32, respectively

$$\underline{\underline{E}} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \qquad \begin{Bmatrix} \sigma_r \\ \sigma_\theta \\ \sigma_z \\ \tau_{zr} \end{Bmatrix} = \underline{\underline{E}} \begin{Bmatrix} \varepsilon_r \\ \varepsilon_\theta \\ \varepsilon_z \\ \gamma_{zr} \end{Bmatrix} \tag{3-30}$$

$$\underline{\underline{E}} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \qquad \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \underline{\underline{E}} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{3-31}$$

$$\underline{\underline{E}} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \qquad \begin{Bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{Bmatrix} = \underline{\underline{E}} \begin{Bmatrix} \varepsilon_x \\ \varepsilon_y \\ \gamma_{xy} \end{Bmatrix} \tag{3-32}$$

In the present work, wood composites are highly oriented and need to be modelled as orthotropic materials, for which the above relations become more complex. Keeping the strains and the stresses in the same order as above, the $\underline{\underline{E}}$ matrices can be expressed in a general form

$$\underline{\underline{E}} = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & 0 \\ Q_{21} & Q_{22} & Q_{23} & 0 \\ Q_{31} & Q_{32} & Q_{33} & 0 \\ 0 & 0 & 0 & Q_{44} \end{bmatrix} \tag{3-33}$$

where the element Q's in the stiffness matrix for an orthotropic material in terms of the engineering constants are derived respectively as Eq. 3-34, 3-35 and 3-36 for the axisymmetric, plane-strain and plane-stress problems:

$$\begin{aligned} Q_{11} &= \frac{1-\nu_{23}\nu_{32}}{E_2E_3\Delta} \\ Q_{12} &= Q_{21} = \frac{\nu_{21}+\nu_{31}\nu_{23}}{E_2E_3\Delta} = \frac{\nu_{12}+\nu_{13}\nu_{32}}{E_1E_3\Delta} \\ Q_{13} &= Q_{31} = \frac{\nu_{31}+\nu_{21}\nu_{32}}{E_2E_3\Delta} = \frac{\nu_{13}+\nu_{12}\nu_{23}}{E_1E_3\Delta} \\ Q_{22} &= \frac{1-\nu_{13}\nu_{31}}{E_1E_3\Delta} \\ Q_{23} &= Q_{32} = \frac{\nu_{32}+\nu_{12}\nu_{31}}{E_1E_3\Delta} = \frac{\nu_{23}+\nu_{21}\nu_{13}}{E_1E_2\Delta} \\ Q_{33} &= \frac{1-\nu_{12}\nu_{21}}{E_1E_2\Delta} \\ Q_{44} &= G_{12} \end{aligned} \quad , \tag{3-34}$$

$$\begin{aligned} Q_{11} &= \frac{1-\nu_{23}\nu_{32}}{E_2E_3\Delta} \\ Q_{12} &= Q_{21} = 0 \\ Q_{13} &= Q_{31} = \frac{\nu_{31}+\nu_{21}\nu_{32}}{E_2E_3\Delta} = \frac{\nu_{13}+\nu_{12}\nu_{23}}{E_1E_3\Delta} \\ Q_{22} &= 0 \\ Q_{23} &= Q_{32} = 0 \\ Q_{33} &= \frac{1-\nu_{12}\nu_{21}}{E_1E_2\Delta} \\ Q_{44} &= G_{12} \end{aligned} \quad , \tag{3-35}$$

and

$$
\begin{aligned}
Q_{11} &= \frac{E_1}{1-\nu_{12}\nu_{21}} \\
Q_{12} &= Q_{21} = 0 \\
Q_{13} &= Q_{31} = \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{21}} = \frac{\nu_{21}E_1}{1-\nu_{12}\nu_{21}} \\
Q_{22} &= 0 \\
Q_{23} &= Q_{32} = 0 \\
Q_{33} &= \frac{E_2}{1-\nu_{12}\nu_{21}} \\
Q_{44} &= G_{12}
\end{aligned}
\quad , \tag{3-36}
$$

where $\Delta = \dfrac{1-\nu_{12}\nu_{21}-\nu_{23}\nu_{32}-\nu_{31}\nu_{13}-2\nu_{21}\nu_{32}\nu_{13}}{E_1 E_2 E_3}$.

Secondly, in the formation of the stiffness matrix and the load vector, "r" is no longer a variable of the radius from the revolution axis to the specified integration point as used in axisymmetric problems. It is taken as the thickness for the plane stress and a unit for the plane strain.

In modifying FEMCOD, three options were given according to these three situations in order to make the program general for any 2-D problems.

CHAPTER FOUR

COMPUTER IMPLEMENTATION

INTRODUCTION

A two-scale approach was used to investigate the behavior of particulate-reinforced wood composites in which the reinforcement volume fraction and the reinforcement orientation vary randomly. Conventional unit cell models were carried out in ABAQUS to generate the mesoscale properties. These properties were then applied to the CALREL-FEMCOD program as basic variables. The deviations in the basic variables were introduced to represent the orientation of the reinforcement, the reinforcement clustering, and the variations of the local volume ratio between the plastic matrix and the reinforcement particles. To be specific, the deviation of the angle between the principal axes of wood particles and the global directions of each element was used to represent the random orientation of the particles, and the deviation in the volume fraction of each element was used to represent the reinforcement clustering.

DETERMINATION OF MESOSCALE PROPERTIES

2-D models were designed to calculate the longitudinal and transversal properties separately. An axisymmetric unit cell model, representing a periodic array of cylinders each containing a single central cylindrical particle, was used to determine the longitudinal mesoscale properties. Similarly, the transversal characteristics were determined by using a plane-strain model (See Fig. 4-1). In both models, the physical fibers of wood were assumed to be oriented in the longitudinal direction.

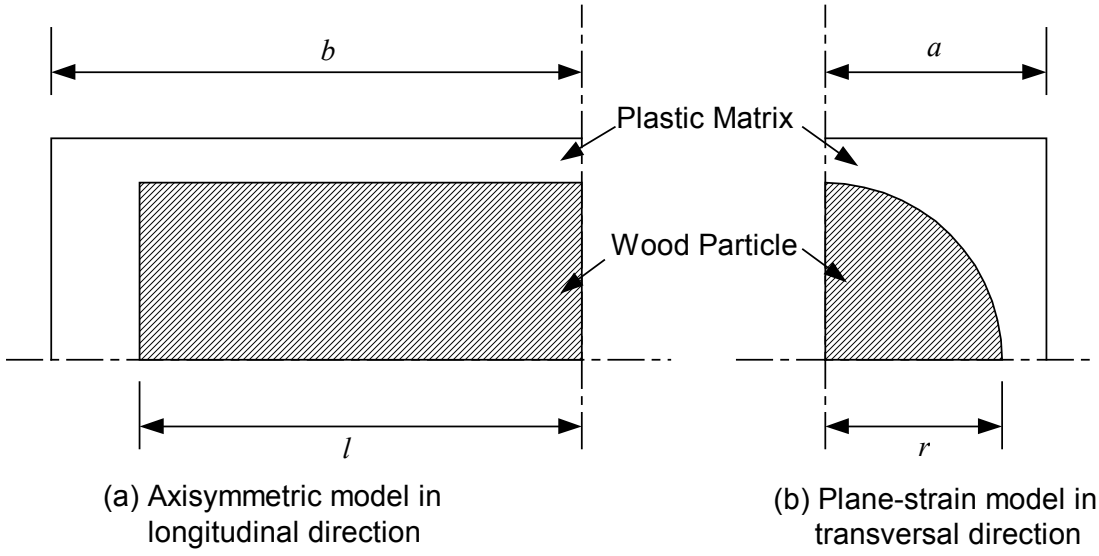


Fig. 4-1 Mesoscale models used to generate composite properties

The finite element models were meshed with isoparametric 8-node elements in ABAQUS/Standard. The properties of wood particles and the plastic matrix were assumed to be determinant in these two models. The plastic matrix material was assumed to be isotropic and elastic, with an elastic modulus of $2\times10^5\ Psi$ and a Poisson's ratio of 0.4. The wood reinforcement was assumed to be highly orthotropic and elastic, with elastic moduli of $E_1 = 1.6\times10^6\ Psi$ and $E_2 = 1.6\times10^5\ Psi$, shear moduli of $G_{12} = G_{13} = 8\times10^4\ Psi$ and $G_{23} = 1.6\times10^4\ Psi$, and all Poisson's ratios of 0.4. The volume fraction was chosen to be 30%, 50%, or 70%, which was used to represent the lower bound, average, or upper bound of values commonly encountered in practice, respectively. An aspect ratio (i.e. $\frac{b}{a}$ in Fig. 4-2) of 10:1 was used.

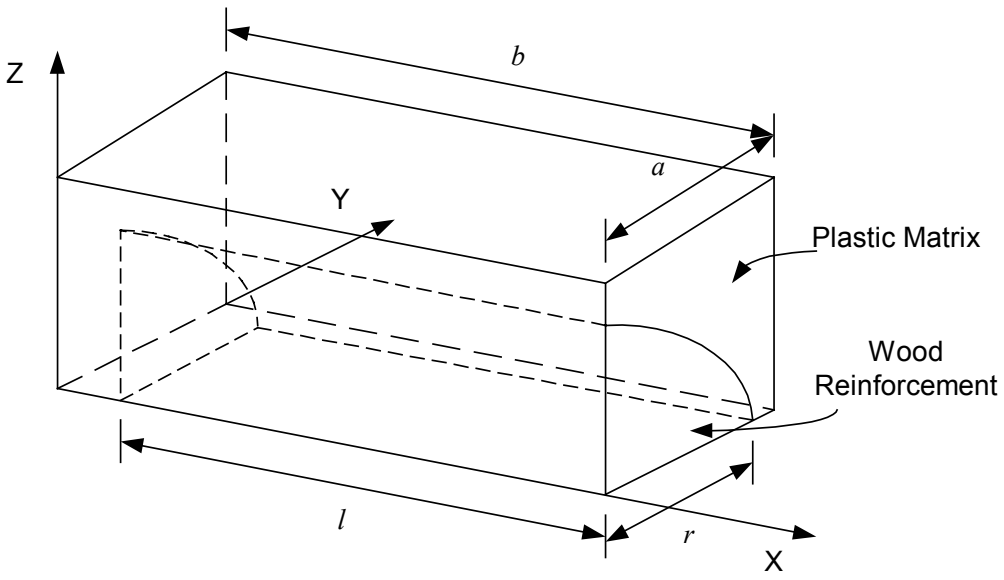


Fig. 4-2 Representation of one corner of a unit cell

To build up the ABAQUS models, the geometry must be first defined according to a prescribed volume ratio. In the 3-D representation of unit cells as shown in Fig. 4-2, the particles are transversely aligned (i.e. the arrangement of particles is identical in the transversal direction) and the volume element represents one-eighth of a single particulate-reinforced unit cell. Thus, the volume of the reinforcement at this corner and the total volume of this portion can be easily computed as

$$V_{particle} = \frac{1}{4}\pi r^2 l \tag{4-1}$$

and

$$V_{composite} = a^2 b \tag{4-2}$$

where r and l are the radius and length respectively of the cylindrical representation of the wood particle; a is the side length of the square cross section; and b is the length of

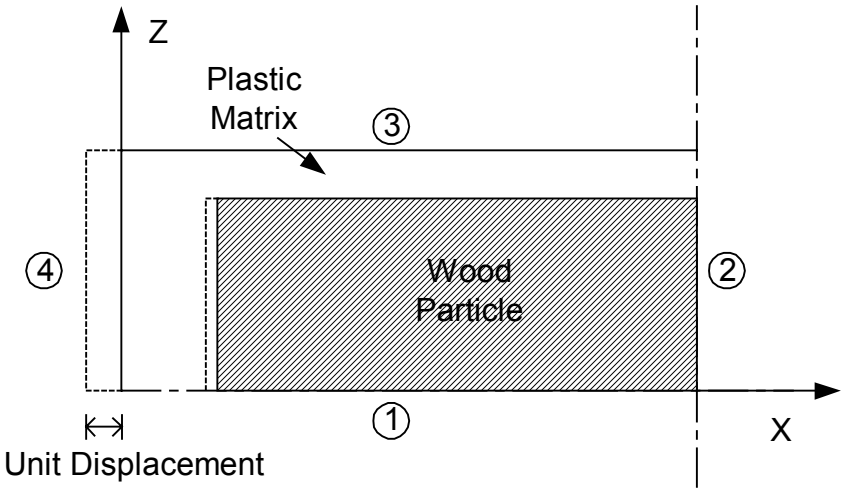
the cuboid (See Fig. 4-2). The same aspect ratio is then assumed for the particle as for the cell itself

$$\frac{l}{r} = \frac{b}{a}. \tag{4-3}$$

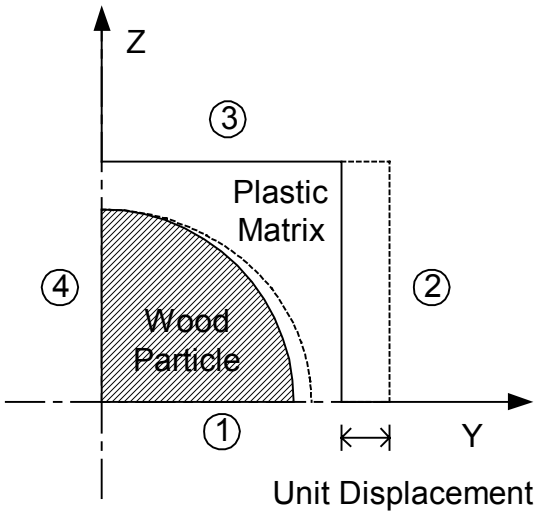
With these assumptions, enough information has been provided to set up a model. As an example, with an average volume fraction of 50% wood particle reinforcement, the relations between these geometric parameters can be easily derived:

$$\begin{aligned} & \frac{V_{Particle}}{V_{Composite}} = 50\% \\ \Rightarrow \quad & \frac{\pi}{4}\frac{r^2 l}{a^2 b} = 50\% \\ \Rightarrow \quad & \frac{\pi}{4}\frac{r^3}{a^3} = 50\% \\ \Rightarrow \quad & r = 0.86a \\ & l = 0.86b \end{aligned} \tag{4-4}$$

where $V_{Particle}$ and $V_{Composite}$ refer to the volume of the particle only and of the whole composite, respectively. The relations were then used to generate the 2-D model geometry. As a result, different models need to be generated for longitudinal and transverse cases according to each single volume fraction. In the current study, a look-up table was created. All material properties were computed for volume ratios of 30%, 40%, 50%, 60%, and 70%. For other volume ratios between the upper and lower bounds (i.e. 30% and 70%), the properties were calculated directly from linear interpolations of two adjacent cases.



(a) Axisymmetric model in longitudinal direction



(b) Plane-strain model in transversal direction

Fig. 4-3 Loading conditions in both longitudinal and transversal models

These models were loaded by applying a uniform displacement to the faces, as shown in Fig. 4-3. The boundary conditions of the other faces were specified by the symmetry requirements. The principal strain along the loading direction in each model was computed from the unit displacement of the loaded face

$$\varepsilon_{loading} = \frac{u_{uniform\ disp.}}{b} \tag{4-5}$$

$$\varepsilon_{loading} = \frac{v_{uniform\ disp.}}{a} \tag{4-6}$$

where $\varepsilon_{loading}$ refers to the *Engineering Strain* in the loading direction, and $u_{uniform\ disp.}$ and $v_{uniform\ disp.}$ are the prescribed displacements in the longitudinal and transversal models, respectively.

Other strains and all stresses were taken to be the averaged values at either Gauss points or nodal points of each element

$$\bar{\varepsilon} = \frac{1}{A}\int_A \varepsilon\, dA \tag{4-7}$$

$$\bar{\sigma} = \frac{1}{A}\int_A \sigma\, dA \tag{4-8}$$

where ε and σ are values for one particular component of strains and stresses, respectively. With the above data available, the general moduli of elasticity and Poisson's ratios of the wood composite can be easily computed.

MACROSCALE PROPERTY DETERMINATION

The macroscale analyses were performed using the Fortran program, denoted as CALREL-FEMCOD, which was developed by combining a reliability code, CALREL, and a finite element code, FEMCOD. The combination was done in a manner by which the finite element and reliability parts of the code were, to the extent possible, kept

independent of one another (See Fig. 4-4). This is very important from the point of view that it allows a researcher in either field to modify the program to implement new research results or to test new methods without knowing the other field.

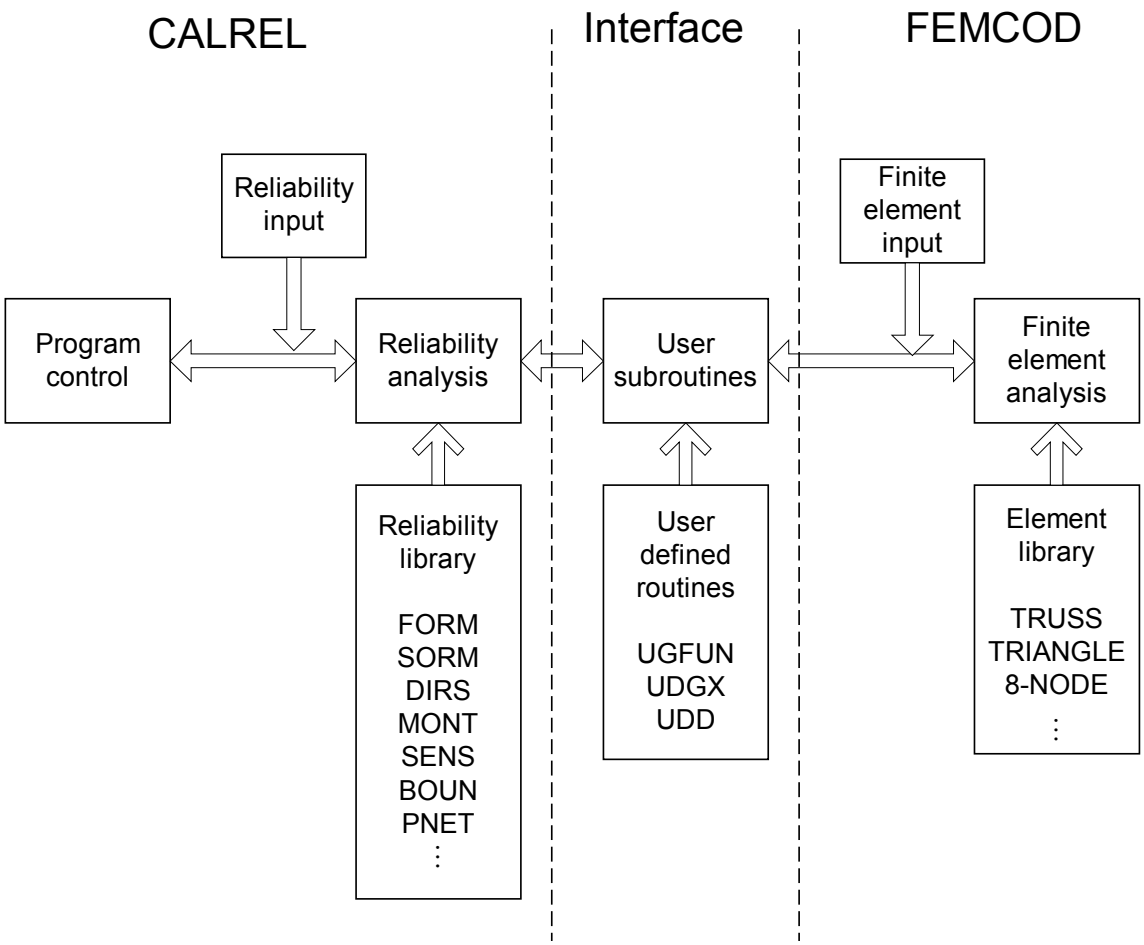


Fig. 4-4 The structure of CALREL-FEMCOD

CALREL (CAL-Reliability) is a general-purpose structural reliability program developed by Liu, Lin, and Der Kiureghian (1989). An essential component of this program is the specification of the limit-state functions that define the failure criteria and their gradients with respect to the basic variables. It was designed to compute the probability integrals in the form of

$$p_f = \int_F f_{\mathbf{X}}(x)\,dx \tag{4-1}$$

where $\mathbf{X}$ is a vector of random variables with joint probability density function $f_{\mathbf{X}}(x)$ and F is the failure domain defined by

$$F \equiv \{g(x) < 0\} \tag{4-2}$$

where $g(x)$ is a limit-state function and $g(x) < 0$ denotes a failure state. In addition to the failure probability, CALREL also computes the sensitivity estimate, which gives the information about the influence from the change of each basic variable.

In CALREL, these specifications, which are problem-dependent, are provided by user-defined subroutines. The limit-state functions are in terms of load effects computed through a mechanical transformation. Usually, the effects are implicit functions of the basic variables. As a result, a finite element code needs to be included to perform this transformation.

CALREL has a large library of probability distributions that can be used to define $f_{\mathbf{X}}(x)$ for independent as well as dependent random variables. It also incorporates four techniques for computing the above quantities according to its users' manual:

(1) First-order reliability method (FORM), where the limit-state surfaces are replaced by tangent hyperplanes at design points in a transformed standard normal space;

(2) Second-order reliability method (SORM), where the limit-state surfaces are replaced by hyperparaboloids fitted at the design points in the standard normal space;

(3) Directional simulation with exact or approximate surfaces; and

(4) Monte Carlo simulation.

Three problem-dependent and, therefore, user-defined subroutines are necessary for running CALREL. The subroutine UGFUN defines the limit-state functions. In addition, the user may define the gradients of the limit-state functions through the subroutine UDGX. However, if desired, CALREL will automatically compute the gradients by the finite difference method and the subroutine UDGX can be left blank. The third subroutine, UDD, is used to define the probability distributions that are not available in the CALREL distribution library. This routine can also be a dummy if no user-provided distributions are necessary.

CALREL executes reliability analyses according to a sequence of macro commands provided in an input file. The following is a list of macro commands currently available in CALREL. Only the first four characters are needed to identify each macro command:

CALRel	defines problem size;
RESTart	restarts a previous run;
DATA	reads input data;
END	ends input data;
FORM	performs FORM analysis;
BOUNd	computes first-order bounds for series systems;
PNET	computes PNET approximation for series systems;
SENSitivity	computes sensitivities of first-order probabilities;
SORM	performs SORM analysis;
DIRS	performs directional simulation;
MONT	performs Monte Carlo simulation;

EXIT terminates execution.

For a full description of the above commands and the input data, which are provided between macro commands DATA and END, please refer to *CALREL USER MANUAL* (Liu, Lin and Der Kiureghian, 1989).

In finite element reliability analyses, the limit-state functions in UGFUN are expressed in terms of load effects. The dependence of the load effects on the basic variables is provided through a finite element program. The finite element code that has been selected for this purpose is FEMCOD, developed by Plesha, Cook and Malkus (1989).

FEMCOD is a short but efficient finite element "program skeleton". It can be modified to implement various kinds of new elements without changing the storage strategy and solution procedure.

To provide the function proposed in the present work, three major modifications were made in FEMCOD:

(1) An isoparametric eight-node 2-D element was added to its element library;

(2) FEMCOD was converted from an independent program into a subroutine; and

(3) The input of some data, which was used as basic variables, was directed to the CALREL program instead of its own input file.

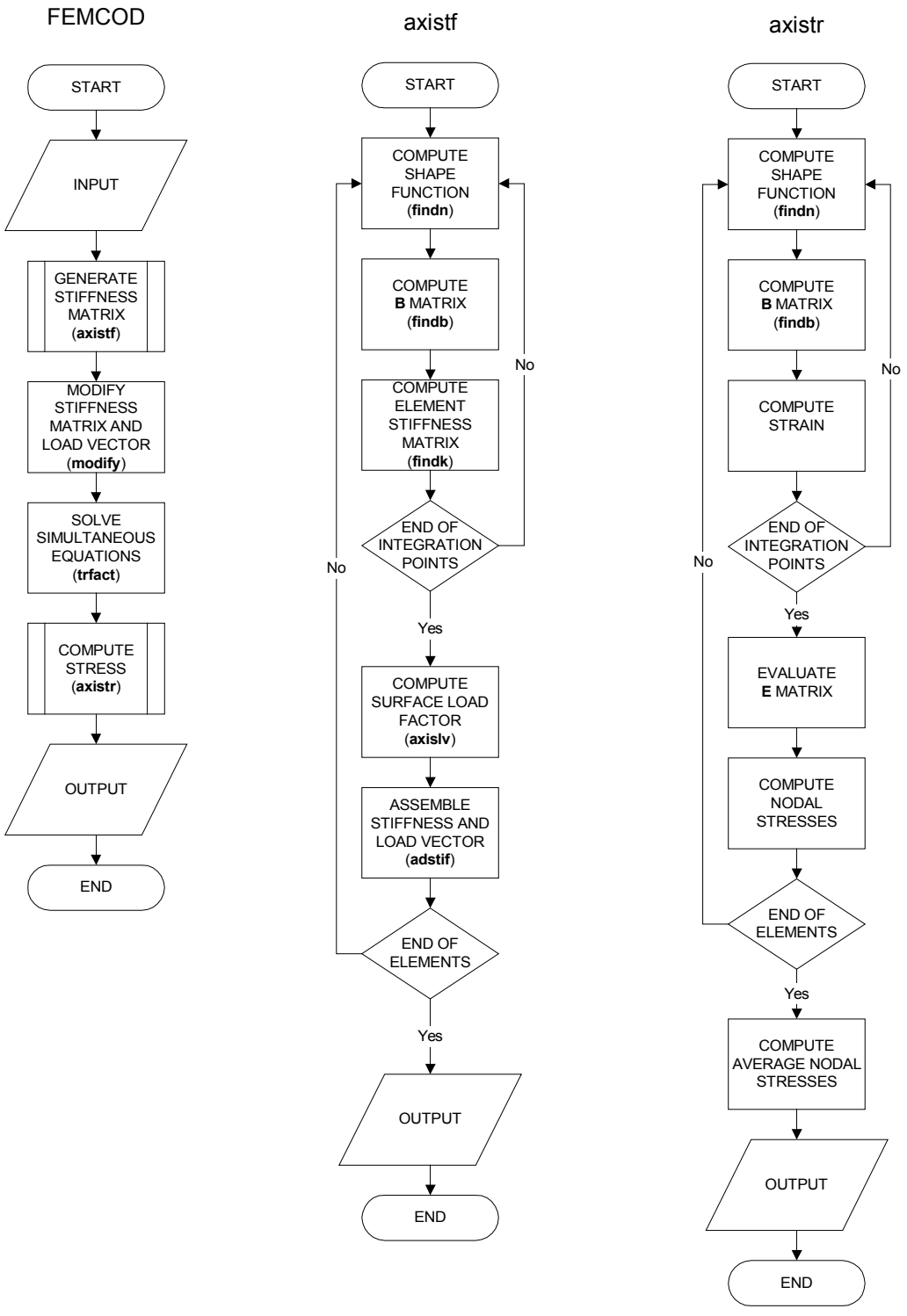


Fig. 4-5 Flowcharts for modified FEMCOD

Flowcharts for the modified version of FEMCOD are shown in Fig. 4-5. The detailed modifications in each subroutine are also given below:

(1) The following subroutines were added to FEMCOD:

axistf: computes the global stiffness matrix;

findn: computes the shape functions at integration points;

findb: computes the **B** matrix at integration points;

findk: computes the element stiffness matrix;

axislv: computes the element equivalent surface load vector;

etrans: transforms the material matrix to the global directions;

axistr: computes nodal stresses from the known nodal displacements,

(2) The following subroutines were modified to adapt new functions:

femcod: works as a subroutine and allocates additional space in storage for surface loads;

ematrix: computes the material matrix for the new 2-D element;

input: includes the input of the surface tractions and the input of basic variables from CALREL, and

(3) The following subroutines have not been changed in any means:

adstif, modify, trfact, mcheck, wi, wr, colht and locsky.

However, minor modifications are not listed above. For a detailed discription of input and the original FEMCOD, please refer to the *PROGRAM DESCRIPTION AND USER GUIDE* by Plesha, Cook and Malkus (1988).

Since the present object files (i.e. .obj files) for CALREL are 16-bit, they cannot be linked in a 32-bit compiler in Windows 95/98/NT with 32-bit libraries. As a result, the CALREL-FEMCOD program was compiled in MS-Fortran 5.1 and needs to run in MS-DOS mode. In addition, to fit into the memory in MS-DOS mode, the size of original global arrays defined in CALREL and FEMCOD was reduced.

CHAPTER FIVE

NUMERICAL APPLICATION

ABAQUS (MESOSCALE) MODEL

Conventional unit cell models were developed in ABAQUS to generate the mesoscale properties. These properties were then applied to the CALREL-FEMCOD program as basic variables.

Two ABAQUS models described in Chapter Four are shown in Fig. 5-1 and Fig. 5-2. The elastic moduli and Poisson's ratios were computed from the stress and strain output at integration points.

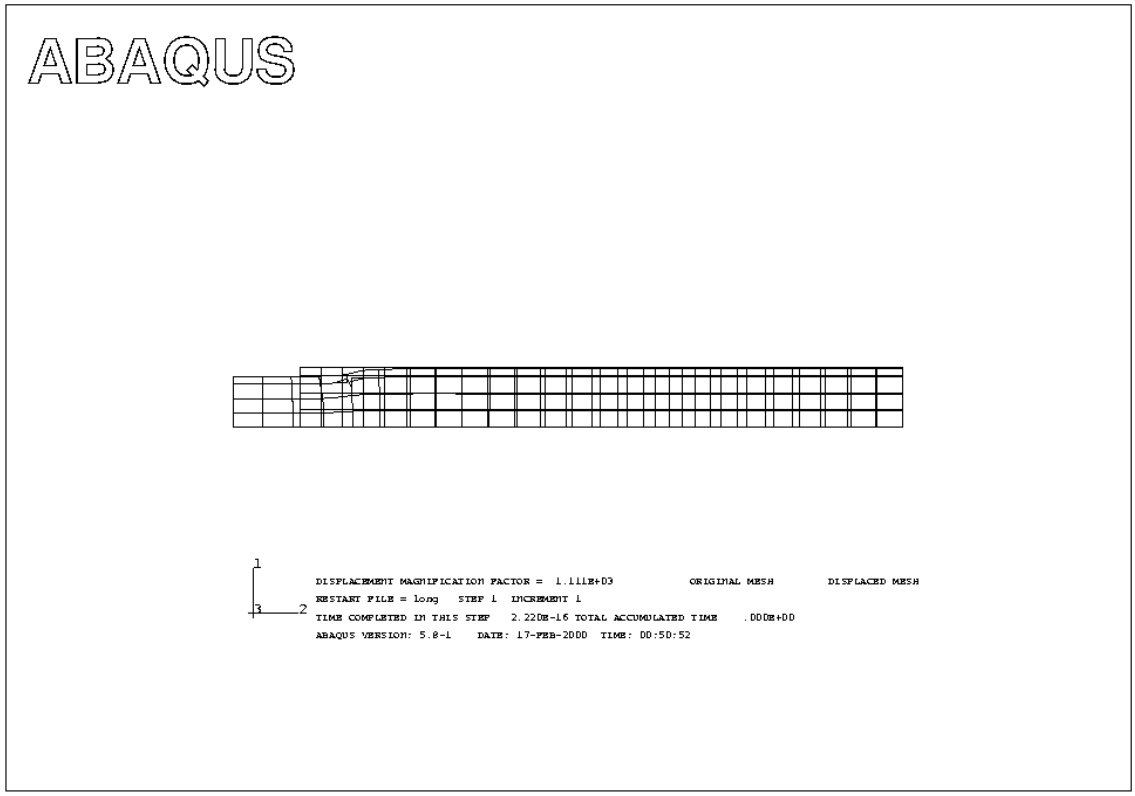


Fig. 5-1 Deformed and original shapes in the longitudinal model

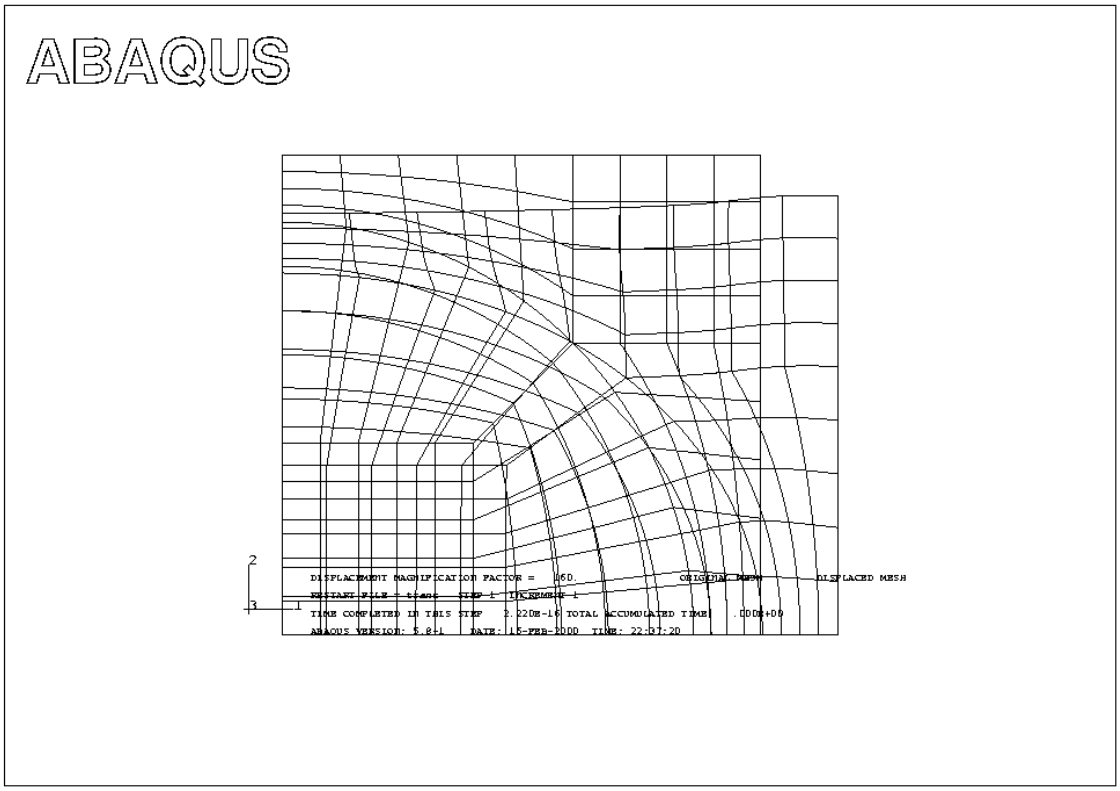


Fig. 5-2 Deformed and original shapes in the transversal model

All strains and stresses except ε_{11} and ε_{22} were taken as the averaged values at Gauss points in every finite element. The material properties of the wood composite were then easily computed, as tabulated below. ν_{12} and G_{12} are fairly constant for all volume ratios. E_1 increases with the increase of wood reinforcement, as could be expected. E_2 and ν_{23} increase first, reach their peak values at 40% reinforcement ratio, and then drop with respect to the increasing volume ratio. At low ratios of reinforcement, increasing wood particles stiffens the composite by Poisson's effects from the elastic modulus in the 1-direction. But, with higher reinforcement ratios, the elastic modulus in the 2-direction of wood particles provides more effect on the composite stiffness. Since E_2 of wood particles (i.e., $1.6\times10^5\,psi$) is assumed to be smaller than

E_2 of the plastic matrix (i.e., $2.0\times10^5\,psi$), E_2 of the composite decreases slightly with the increase of the volume ratio.

Table 5-1 Material properties for different volume ratios

Volume ratio	$E_1\,(\times10^5\,psi)$	$E_2\,(\times10^5\,psi)$	ν_{12}	ν_{23}	$G_{12}(\times10^5\,psi)$
30%	4.75	1.63	0.399	0.530	0.740
40%	5.01	1.96	0.397	0.531	0.740
50%	6.56	1.91	0.397	0.513	0.742
60%	7.47	1.89	0.399	0.509	0.746
70%	8.10	1.85	0.402	0.494	0.762

To verify the accuracy of these two models, all parameters for the 50% reinforcement ratio computed above were checked against the theoretical results. Only the procedures of comparison of E_1 and E_2 are shown below. For other theoretical solutions, one may refer to any composite book.

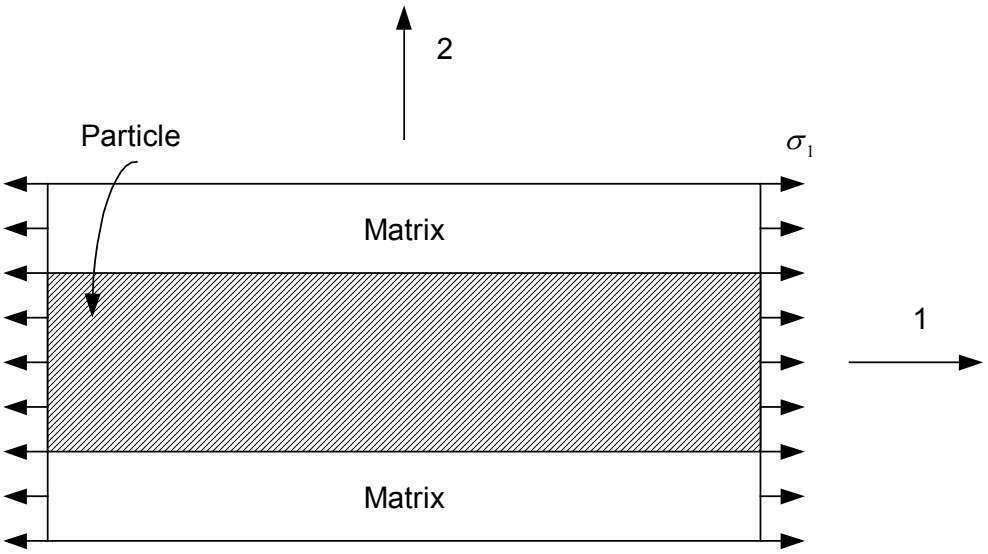
According to the basic mechanics of composite materials, the elastic moduli in the 1- and 2-directions (see Fig. 5-3) could be determined by the *rules of mixture*, given by

$$E_1 = E_pV_p + E_mV_m \tag{5-1}$$

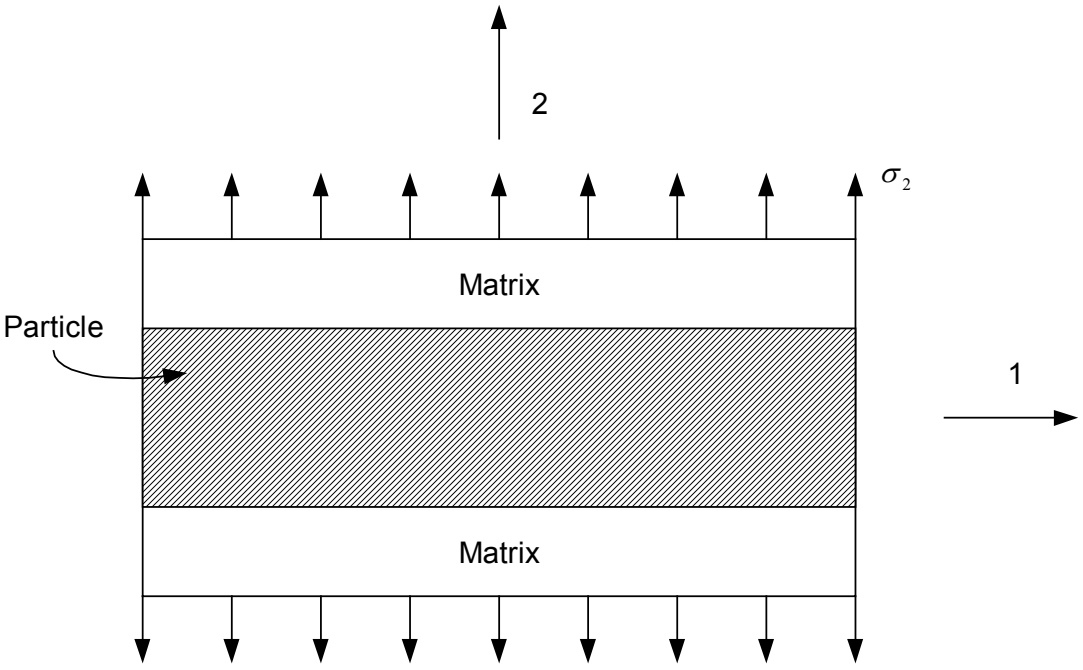
$$E_2 = \frac{E_mE_p}{E_mV_p + E_pV_m} \tag{5-2}$$

where E_p and E_m are the elastic moduli in the 1- or 2-direction for the particle and matrix, respectively, and V_p and V_m refer to the volume fractions of the particle and the matrix over the total volume of the composite. In the actual volume element as shown in

Fig. 5-4, E_1 and E_2 need to be computed in a stepwise fashion as shown, and the results were $6.55 \times 10^5\ psi$ and $1.79 \times 10^5\ psi$, respectively. Based upon the fact that the rules of mixtures actually neglect the Poisson's effects and the stiffness in the other directions, the results generated from ABAQUS are of higher accuracy. Similar to these two moduli, values of other parameters are also very close to those obtained theoretically with improvement due to the relaxation of assumptions. The actual strain and stress distributions in ABAQUS models are shown in Fig. 5-5 and 5-6.



(a) longitudinal direction



(b) transversal direction

Fig. 5-3 Representative volume element loaded in 1- and 2-directions

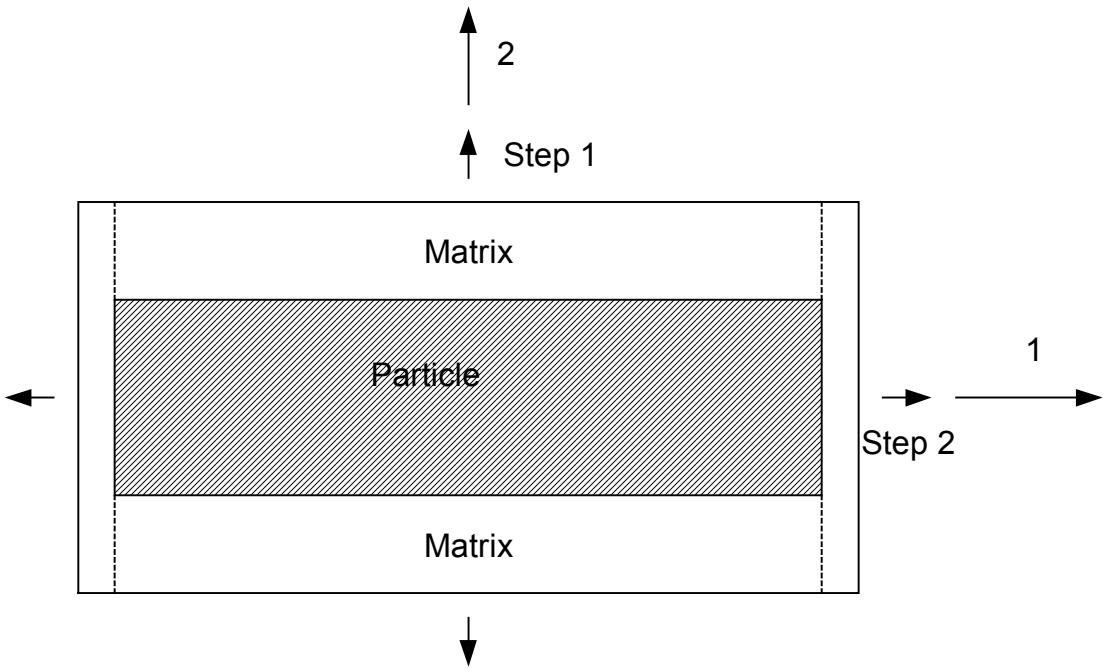


Fig. 5-4 Stepwise use of the *rules of mixtures*

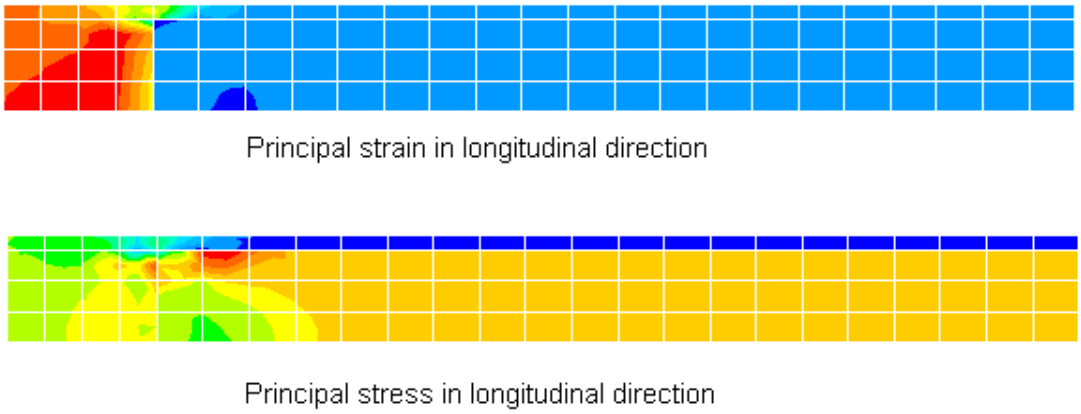
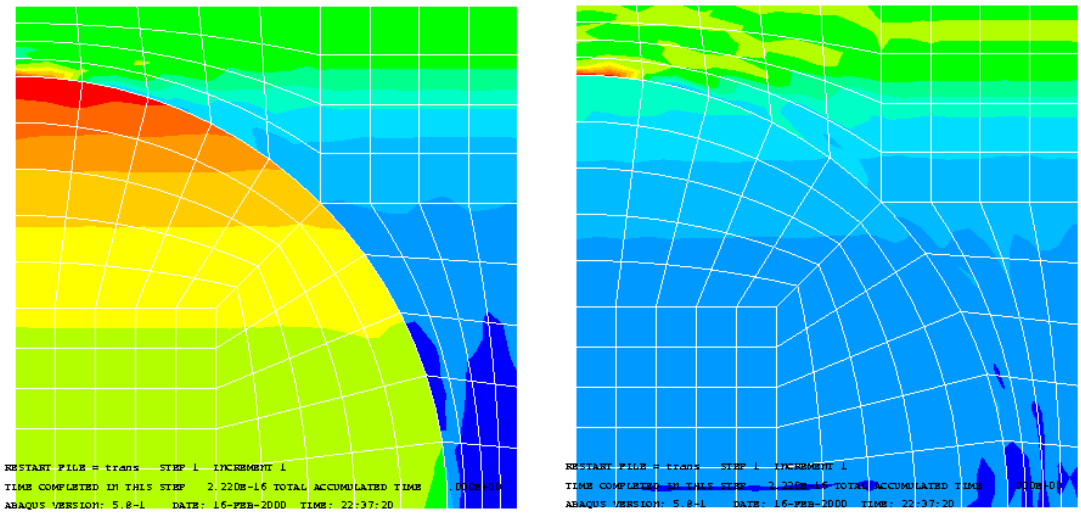


Fig. 5-5 Strain and stress in the longitudinal model



Principal strain in transversal direction

Principal stress in transversal direction

Fig. 5-6 Strain and stress in the transverse model

MACROSCALE ANALYSIS

The CALREL-FEMCOD program was used to study two simple structures. The structures were meshed with 8-node isoparametric elements as plane-stress problems. Several typical finite elements made up an element group, the elements of which possessed the same material properties. However, the properties varied between each element group. This approach represented the characteristics of randomly reinforced composite structures. Each element group had random values for the orientation angle, affecting the stiffness matrix transformation, and the reinforcement ratio, which decided basic material constants from the look-up table generated from mesoscale models (see Table 5-1).

Failure estimates were made for each structure according to a specific failure criterion. Due to the storage limitation within the current version of CALREL-FEMCOD, only fairly coarse finite element meshes and random field meshes were used

to study these two structures. In general, much finer meshes need to be used to obtain a converged result. Thus, the results of the structural reliability analyses shown in this chapter should only be used to show the overall trend of the effects from different factors. In the current study, the effects of the probability distribution of each parameter on structure reliability were studied. Sensitivity analyses were also performed for each basic variable.

NUMERICAL EXAMPLE 1

A pure tension test was performed for a 4 inch high, 10 inch wide thin plate (see Fig. 5-7). The plate was meshed with 40 elements as shown in Fig. 5-8. Since it is not efficient or computationally stable to use the same mesh for the random field discretizations as used for finite element analyses, Der Kiureghian and Ke (1988) suggested using separate meshes for them. In this study, a separate mesh for random fields of properties was then taken as a group of four finite elements (see Fig. 5-8). The bold border describes a random field mesh. As a result, 10 groups were defined and arranged as shown in Fig. 5-8. Each group had two random basic variables, namely the orientation angle and the volume ratio. Totally, 20 basic variables were used for the whole structure. The failure criterion used in this example was the maximum stress theory with an assigned value of 12000 *psi* parallel to the reinforcement direction. For simplification, the strength contribution from the transversal direction was ignored. The stress limits were adjusted to the global direction according to different mean orientation angles.

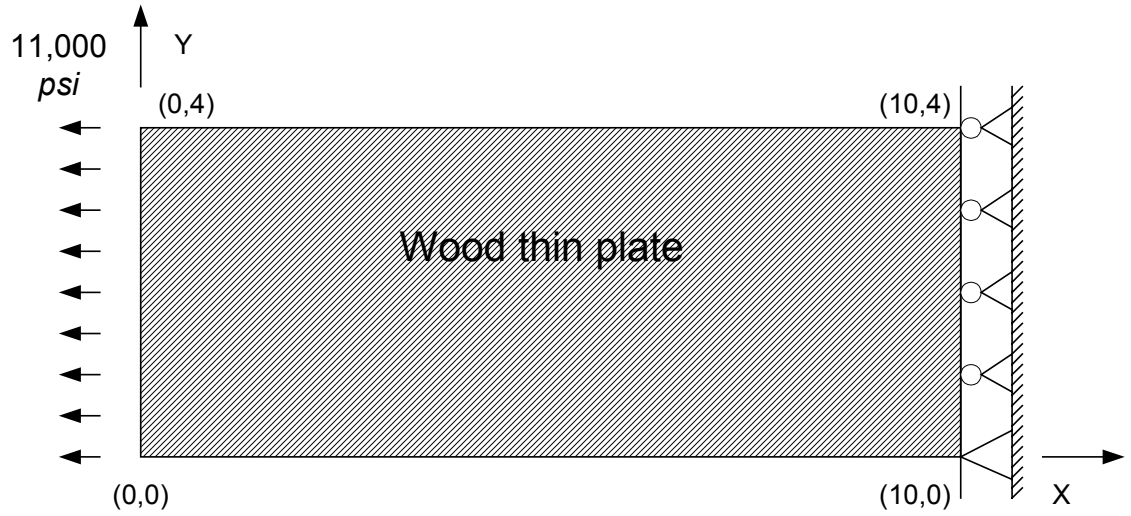


Fig. 5-7 Simple tension test of a thin plate

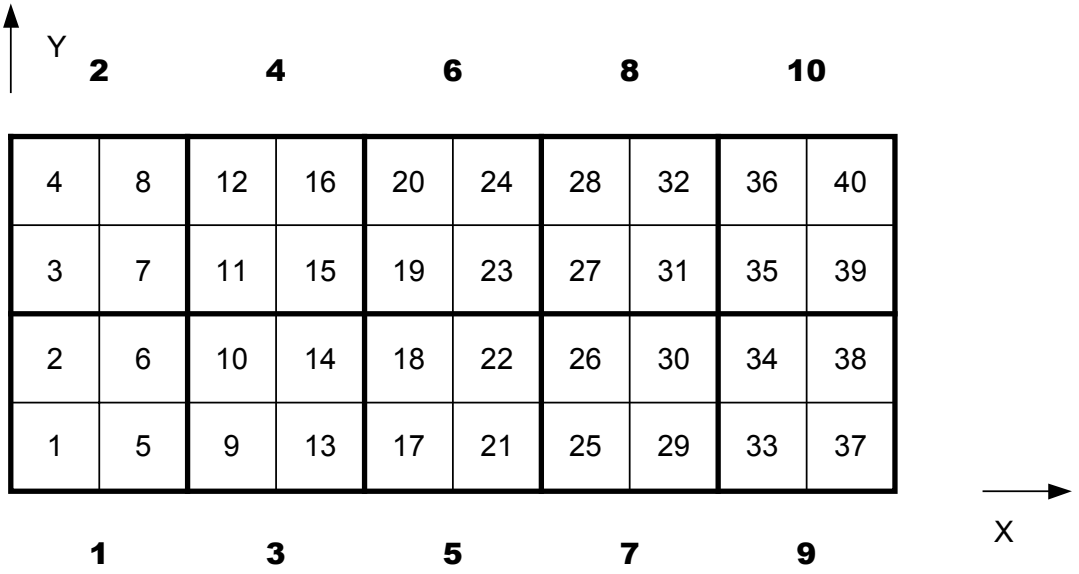


Fig. 5-8 Finite element mesh and random field mesh

The two basic variables were both assumed to have normal distributions. At the condition for which the mean volume ratio equaled 0.5 and the mean orientation angle equaled 0°, the failure probabilities and the performance indices (i.e. the reliability

indices, β) of the tension test were calculated according to different standard deviations (S. D.) of these two basic variables as shown in Table 5-2a and 5-2b:

Table 5-2a Failure probability for different standard deviations

S. D. of V. R. / S.D. of O. A.	10%	20%	30%
2.5°	9.6%	9.7%	9.9%
5°	25.7%	25.7%	25.8%
10°	37.2%	37.2%	37.2%

Note: V.R. means the volume ratio and O.A. means orientation angle.

Table 5-2b Performance indices for different standard deviations

S. D. of V. R. / S.D. of O. A.	10%	20%	30%
2.5°	1.305	1.297	1.286
5°	0.653	0.652	0.650
10°	0.327	0.327	0.326

Note: V.R. means the volume ratio and O.A. means orientation angle.

It is apparent that the structure has increased likelihood of failure with increased variation in properties (i.e. the structure is more random). Since the load was a uniform tensile pressure and the failure criterion was a stress limit in this example, the orientation angle has much more effect on the structure than does the volume ratio. The latter only affected the stiffness of the structure and, thus, resulted in different displacements. Based on this fact, further investigation was conducted on the effects on failure from distribution parameters of the orientation angle in this example. The volume ratio was

kept the same in a normal distribution of 0.5 mean and 0.1 standard deviation. The results, with respect to different means and standard deviations of the reinforcement angle, are shown in Table 5-3ab and Fig. 5-9ab. With a larger reinforcement angle (i.e. the reinforcement is further away from the tension direction), the structure was weaker and more liable to fail. Also, the failure probability was higher with the increase of variations in the reinforcement angles (i.e. larger standard deviations).

Table 5-3a Failure probability for different parameters of orientation angles

S. D. / Mean	2.5°	5°	10°	15°	20°
0°	9.6%	25.7%	37.2%	41.4%	43.5%
7.5°	19.0%	29.2%	38.4%	42.2%	44.1%
15°	40.1%	44.4%	47.1%	48.1%	48.6%

Note: All angles here are in degrees.

Table 5-3b Performance indices for different parameters of orientation angles

S. D. / Mean	2.5°	5°	10°	15°	20°
0°	1.305	0.653	0.327	0.218	0.163
7.5°	0.879	0.549	0.294	0.198	0.148
15°	0.251	0.141	0.072	0.048	0.036

Note: All angles here are in degrees.

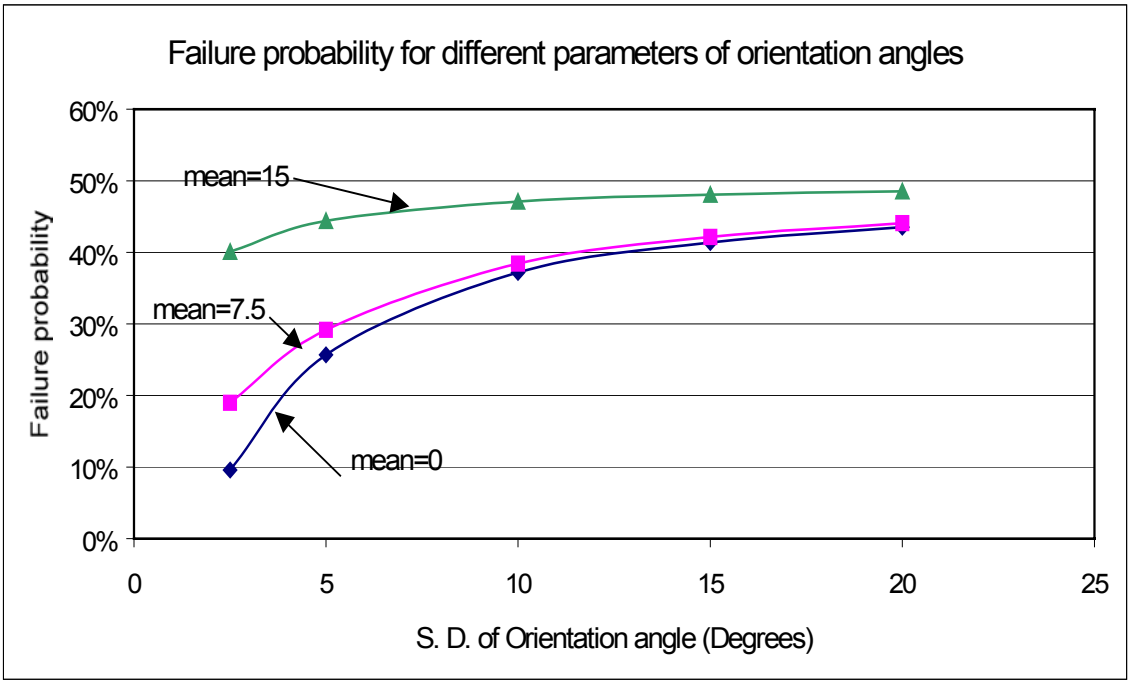


Fig. 5-9a Failure probability for different parameters of orientation angles

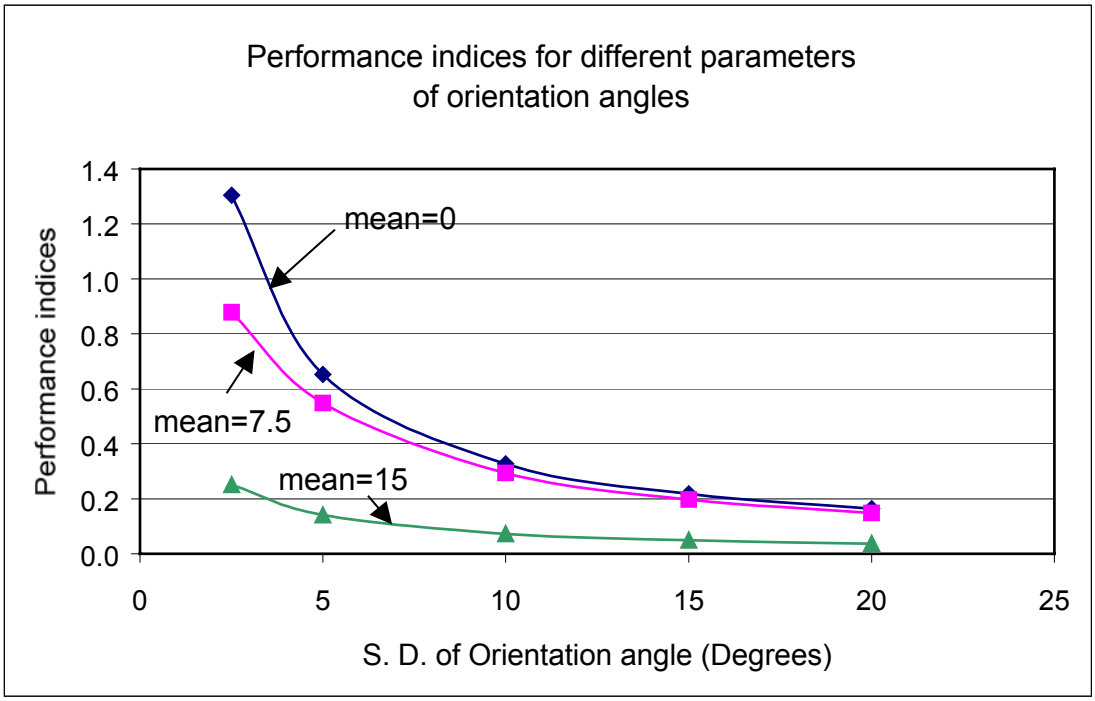


Fig. 5-9b Performance indices for different parameters of orientation angles

Sensitivity analyses were also conducted on basic variables in each situation. The results turned out to be similar to each other. Thus, only the sensitivity estimates for an arbitrary case are shown in Table 5-4. The selected case is one with a volume ratio of 50% mean and 10% standard deviation and an orientation angle of 15° mean and 15° standard deviation. Each number describes the effect of changing a specific parameter on the reliability of the structure. It could be regarded as a partial differential of the failure function over each parameter. The bigger the number, the bigger the influence. From this table, the structure was shown to be most sensitive to variation of the properties at locations furthest away from the loading. This result is fairly reasonable because when the location is very close to the loading side, the principal stress tends to be equal to the uniform pressure no matter how the properties vary.

Table 5-4 Sensitivity analysis for basic variables

Basic variable	Description	Mean	Standard deviation
x1	v. r. for group 1	-1.06E-03	1.35E-08
x2	o. a. for group 1	3.28E-03	3.40E-07
x3	v. r. for group 2	5.85E-04	4.23E-09
x4	o. a. for group 2	-2.08E-03	1.36E-07
x5	v. r. for group 3	-1.89E-02	4.32E-06
x6	o. a. for group 3	4.93E-03	7.56E-07
x7	v. r. for group 4	7.34E-03	6.22E-07
x8	o. a. for group 4	1.12E-02	3.98E-06
x9	v. r. for group 5	-8.23E-02	8.18E-05
x10	o. a. for group 5	-4.14E-02	5.48E-05
x11	v. r. for group 6	3.12E-02	1.18E-05
x12	o. a. for group 6	1.05E-01	3.50E-04
x13	v. r. for group 7	-1.34E-01	2.16E-04
x14	o. a. for group 7	-3.89E-01	4.78E-03
x15	v. r. for group 8	2.73E-02	8.68E-06
x16	o. a. for group 8	5.60E-01	9.97E-03
x17	v. r. for group 9	2.33E-01	6.53E-04
x18	o. a. for group 9	-1.08E+00	3.69E-02
x19	v. r. for group 10	-1.96E-01	4.77E-04
x20	o. a. for group 10	8.11E-01	2.09E-02

NUMERICAL EXAMPLE 2

A 20×20 in. square thin plate was loaded by a uniform pressure, 8 psi, on one side as shown in Fig. 5-10. The plate was fixed at two corners and the structure was meshed with four 8-node elements. In this study, even though the material properties were still random, they were assumed to be the same for all elements. The reliability of the structure was checked against several displacement thresholds. From another point of view, the failure criterion in this example could be regarded as a stiffness limit (i.e., $\frac{EI}{L}$).

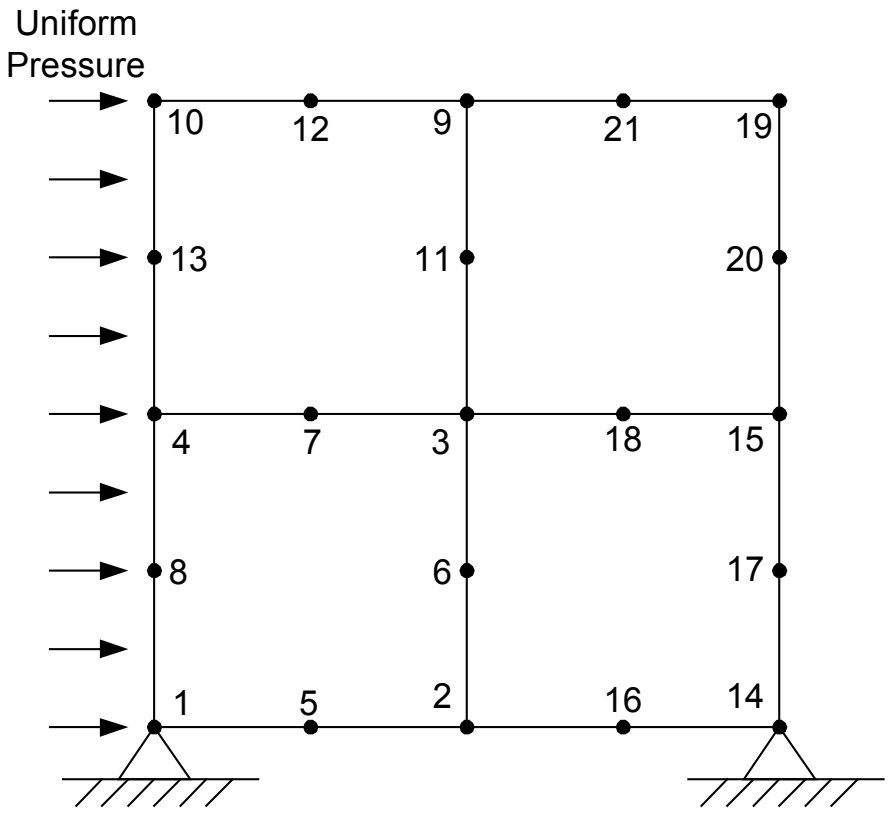


Fig. 5-10 A square plate under a uniform side pressure

Emphasis was directed to the effects from different volume ratios in this example. The orientation angle was then assumed to be a normal distribution of 90° mean with a 5° standard deviation in all cases. The structure was also assumed to be reinforced with 45%, 50%, and 55% wood particles plus a $\pm 5\%$ fluctuation. The displacement limit was evaluated at node 9, which was the middle node at the top side of the plate as shown in Fig. 5-10.

Table 5-5a Failure probability for different displacement thresholds

Disp / V. R.	0.095	0.0955	0.096	0.0965	0.097	0.0975	0.098	0.0985
45%	99.0%	98.0%	96.3%	94.0%	89.9%	84.8%	78.5%	72.2%
50%	95.5%	91.9%	86.6%	78.9%	67.3%	56.6%	45.3%	34.5%
55%	84.1%	76.1%	63.8%	49.8%	35.9%	23.1%	13.5%	8.49%

Note: Displacements are all in inches.

Table 5-5b Performance indices for different displacement thresholds

Disp / V. R.	0.095	0.0955	0.096	0.0965	0.097	0.0975	0.098	0.0985
45%	-2.315	-2.054	-1.792	-1.552	-1.275	-1.026	-0.791	-0.590
50%	-1.691	-1.401	-1.106	-0.804	-0.447	-0.165	0.118	0.400
55%	-0.999	-0.710	-0.353	0.004	0.361	0.734	1.102	1.373

Note: Displacements are all in inches.

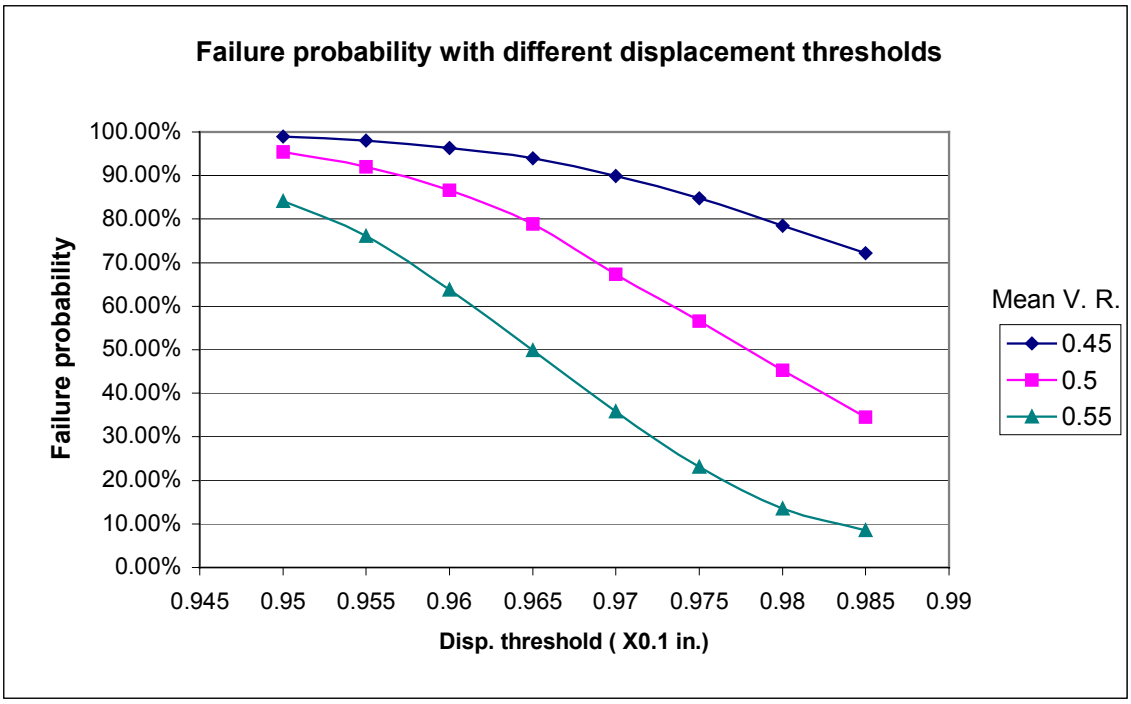


Fig. 5-11 Failure probability for different displacement thresholds

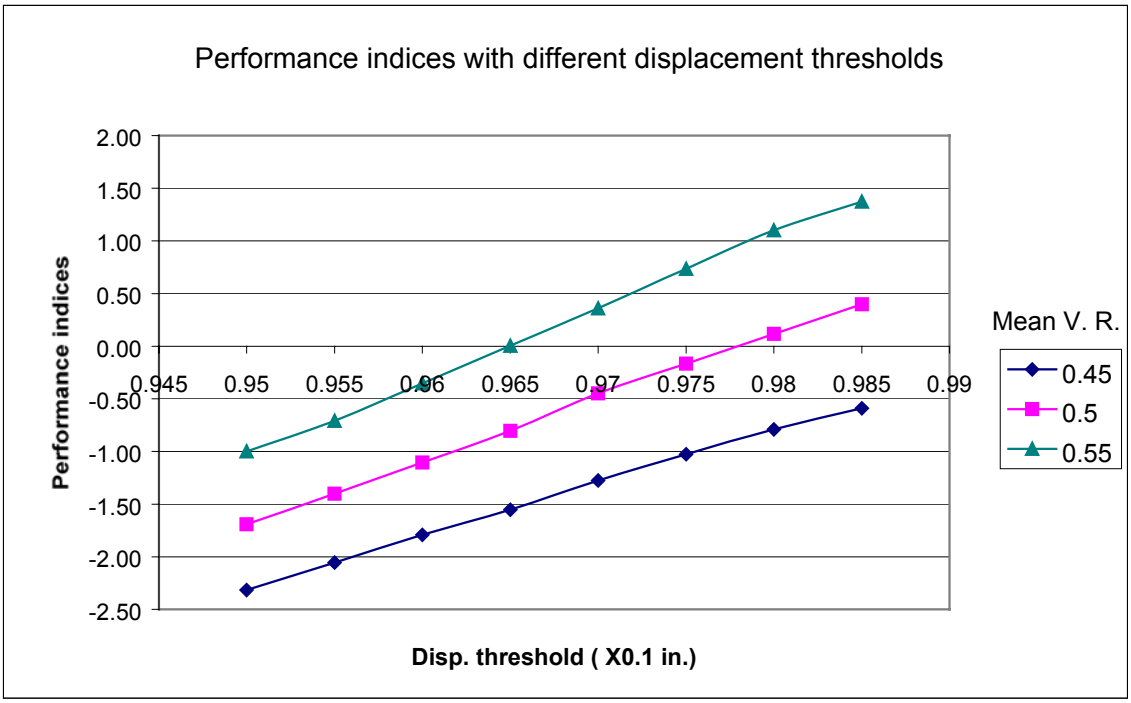


Fig. 5-11b Performance indices for different displacement thresholds

As expected, the structure was less liable to exceed the displacement limit and the overall performance was better with an increase in the displacement threshold and reinforcement fraction, as shown in Table 5-5ab and Fig. 5-11ab. Also, with a relatively large displacement limit, the reliability of the structure was considerably more sensitive to the reinforcement ratio. For instance, at the threshold of 0.0985 inch, the failure probability dropped from 72.2% to 8.49%; but it only dropped from 99.0% to 84.1% for 0.095 inch limit.

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATIONS

Probabilistic finite element methods have been successfully introduced in the study and analysis of wood composites. Two-scale models were used to keep the computation effort for composite structures in a manageable range while also representing the complex microstructure of particulate-reinforced composites.

A measure of the reliability of the structure was obtained for a specific failure criterion, which gave a better estimate with regard to how the structure could behave and how efficient the structure is, in comparison with deterministic analysis. Structures designed for different purposes usually demand different safety coefficients. In practice, a structure can only be designed to ensure that a structure will not fail at an acceptable level of probability. There are various sources of uncertainty in structural design. In this thesis, all study was directed to the uncertainty of material properties. From the aspect of probability methods, the random properties were represented by using the reinforcement ratio and the orientation angle as two basic variables for each element group. The size of the element group could be decided according to the level of randomness.

Besides safety, economy is also a major criterion for a practical design. To improve the behavior of a structure without causing a significant increase in cost, revisions need to be done in a very efficient way. Direction for this could be seen from the results of probability analyses. Sensitivity estimates indicate the relative influence of different parameters, which could direct attention toward the most important design parameters.

The models used to derive mesoscale properties and to study macroscale structures were developed using different software and separated into two independent parts. This

approach provided the advantage of being able to change one part without affecting the other. For a specific wood composite, a material-constant table can be generated for use in analyzing all structures made of it. For future work, a more realistic mesoscale model could be designed to formulate a better material law. In the mean time, existing macroscale models may be used to make runs according to the new properties in the form of the look-up table.

From the first numerical examples studied, the influence of the orientation angle was shown to be larger than that from the volume ratio in pure tension if the reinforcement is along the tension direction. This means that if the particles are well aligned in extruded wood composites, the quality of these composites can be improved without changing components or their ratios. Also, of more importance are the material properties at the locations away from loading, which should be incorporated in manufacture to improve the performance of tension members. In both examples, with higher randomness, the structures had more tendency towards failure. As a result, reducing this factor is also a practical way to raise the structural safety.

There are several shortcomings in the current study. First of all, analyses were based on elastic behavior of the material. Secondly, the wood particle and plastic matrix were assumed to deform together in the models. In practice, the gap between them could be as big as one forth of the radius of the wood particle. This interaction could be modelled as a contact problem with various initial gaps and friction coefficients. The models would then behave nonlinearly and differently in tension and compression. As a result, the material properties of each element would be problem-dependent and orientation-dependent. For instance, in bending problems, the tension side of the model behaves

differently from the compression side, which is similar to the behavior of cracked reinforced-concrete sections. Iterations are needed to determine the neutral surface, which does not undergo any deformation. Considering the random reinforcement, orientation, and orthotropic characteristics of the wood composite, the situation is even more complicated. A nonlinear ABAQUS model was developed to show this influence on basic material properties. The longitudinal model was set up by assuming no initial gap and no friction. The stress-strain relationship in the principal direction is shown in Fig. 6-1. It behaved almost linearly on both the compression and tension sides, though the elastic modulus in compression was approximately three times the one in tension. If they are compared with the results from the ABAQUS models used in Chapter 4, E_1 in compression is almost the same, but E_1 in tension is much smaller. It could make the structure considerately softer than predicted in the previous chapter. However, addressing these issues thoroughly was beyond the scope of the work of this thesis.

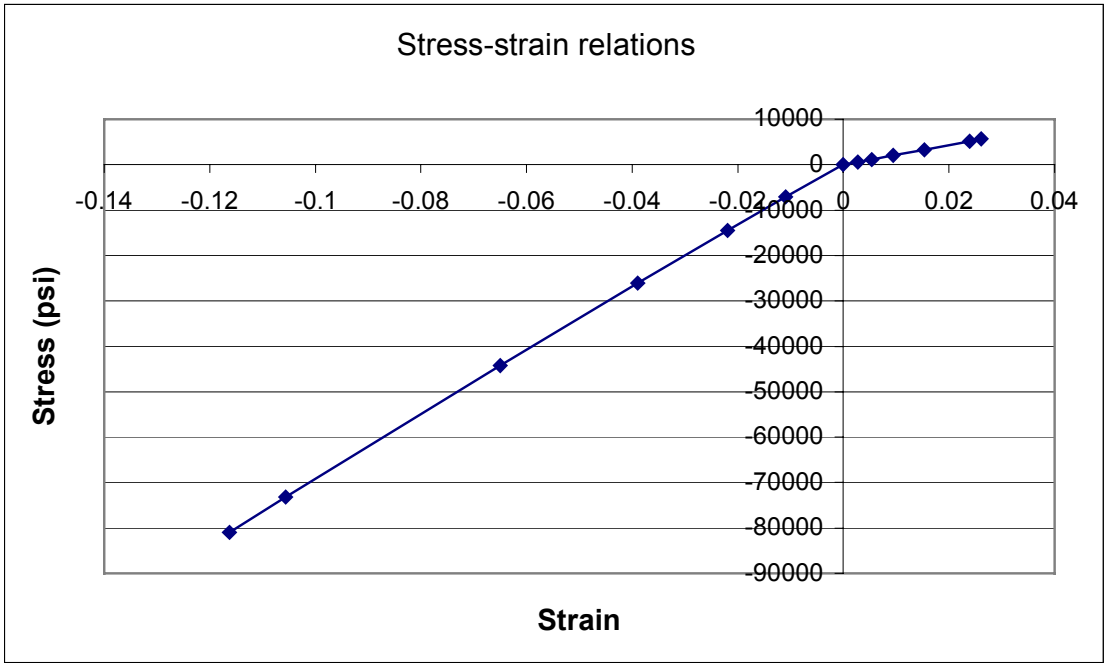


Fig. 6-1 Stress-strain relations in a nonlinear model

The CALREL-FEMCOD program currently does not possess the capability to handle nonlinear problems and the resulting iterative solution procedure. Thus, a nonlinear material law generated from ABAQUS must be linearized and equalized before it could be used in macroscale analyses. Also, the current CALREL-FEMCOD program only has a limited finite element library and has only been applied to two simple problems. All in all, a lot of continuing and interesting work could be done in the future study based on the basic idea and the program frame introduced in this thesis.

BIBLIOGRAPHY

1. Astill, C.J., Nosseir, S.B. and Shinozuka, M., "Impact Loading on Structures with Random Properties", Journal of Structural Mechanics, 1, 63 (1972)

2. Bao, G., Hutchinson, J.W. and McMeeking, R.M., "Particle Reinforcment of Ductile Matrices against Plastic Flow and Creep", Acta metallurgica et materialia, 39, 1871 (1991)

3. Cook, R.D., Malkus, D.S. and Plesha, M.E., "Concepts and Applications of Finite Element Analysis", Third edition, John Wiley & Sons (1989)

4. Der Kiureghian, A. "Finite Element Methods in Structural Safety Studies", Proceedings, ASCE-STD Symposium on Structural Safety Studies, ASCE Convention, 40 (May, 1985)

5. Der Kiureghian, A. and Ke, B.J., "The Stochastic Finite Element Method in Structural Reliability", Probabilistic Engineering Mechanics, 3, 83 (1988)

6. Duva, J.M. and Hutchinson, J.W., "Constitutive Potentials for Dilutely Voided Nonlinear Materials", Mechanics of Materials, 3, 41 (1984)

7. Handa, K. and Anderson, K., "Application of Finite Element Methods in the Statistical Analysis of Structures", Proceedings, Third International Conference on Structural Safety and Reliability, 409 (June, 1981)

8. Hisada, T. and Nakagiri, S., "A Note on Stochastic Finite Element Method (Part 3) – An Extension of the Methodology to Nonlinear Problems", Seisan-Kenkyu, 32, 14 (1980)

9. Hisada, T. and Nakagiri, S., "Stochastic Finite Element Method Developed for Structural Safety and Reliability", Proceedings, Third International Conference on Structural Safety and Reliability, 395 (June, 1981)

10. Leggoe, J.W., Mammoli, A.A., Bush, M.B. and Hu, X.Z., "Finite Element Modelling of Deformation in Particulate Reinforced Metal Matrix Composites with Random Local Microstructure Variation", Acta materialia, 46, 6075 (1998)

11. Levy, A. and Papazian, J.M., "Tensile Properties of Short Fiber-Reinforced SiC/Al Composites: Part I. Effects of Matrix Precipitates", Metallurgical transactions, 21A, 401 (1990)

12. Levy, A. and Papazian, J.M., "Tensile Properties of Short Fiber-Reinforced SiC/Al Composites: Part II. Finite-Element Analysis", Metallurgical transactions, 21A, 411 (1990)

13. Liu, P.L. and Der Kiureghian, A., "Finite-Element Reliability Method for Geometrically Nonlinear Stochastic Structures", Report No. UCB/SEMM-89/05, Structural Engineering, Mechanics, and Materials, Department of Civil Engineering, University of California, Berkeley (January, 1989)

14. Liu, P.L. and Der Kiureghian, A., "Multivariate distribution models with prescribed marginals and covariances", Probabilistic Engineering Mechanics, 1, 105 (1986)

15. Liu, P.L. and Der Kiureghian, A., "Reliability of Geometrically Nonlinear Structures", Proceedings, ASCE-EMD/GTD/STD Joint Specialty Conference on Probabilistic Methods, 164 (May, 1988)

16. Liu, P.L., Lin, H.Z. and Der Kiureghian, A., "Calrel User Manual", Department of Civil Engineering, University of California, Berkeley (1989)

17. Melchers, R.E., "Structural Reliability –Analysis and Prediction", Halsted Press (1987)

18. Melchers, R.E., "Structural Reliability –Analysis and Prediction", Second edition, Halsted Press (1999)

19. Plesha, M.E., Cook, R.D. and Malkus, D.S., "FEMCOD - Program Description and User Guide", Department of Engineering Mechanics, University of Wisconsin, Madison (1988)

20. Shinozuka, M. and Lenoe, E., "A Probabilistic Model for Spatial Distribution of Material Properties", Engineering Fracture Mechanics, 8, 217 (1976)

21. Zhang, Y. and Der Kiureghian, A., "First-excursion probability of uncertain structures", Probabilistic Engineering Mechanics, 9, 135 (1994)

22. Zhu, H.T. and Zbib, H.M., "A Macroscopic Model for Plastic Flow in Metal Matrix Composites", MME Report No. 93-02, Department of Mechanical & Material Engineering, Washington State University (1993)

APPENDIX

A. FORTRAN SOURCE FILES

```
c-------------------------------------------------------------
c*************************************************************
c     femcod.f
c-------------------------------------------------------------
c=============================================================
      subroutine femcod (var,xrv)
c
      implicit double precision (a-h,o-z),integer(i-n)
      common iaa(1200),a(12000)
      common /mpoint/ mpcord,mpfext,mpdisp,mpsurf,mptemp
     .               ,mpwork,mpstif,mpend,maxrel
     .               ,ipkfix,ipnod,ipiadr,ipmat,ipend,maxint
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      common /rmatrl / rmat(10,10)
      common /device/ iin,iout,ibug
      character*80 card,uchar
      dimension xrv(1)
c
c---- written by profs. m.e. plesha, r.d. cook & d.s. malkus
c               department of engineering mechanics
c               university of wisconsin--madison
c               sept.,1984
c               march,1988  revised
c               version  3.28.89.1
c
c---- modified by Wei Yang
c               May,2000
c
c     For 8-node isoparametric element for 2-D problem
c     This modified FEMCOD is a subroutine called by CALREL
c
      iin=10
      iout=11
      open (unit=10,status='old',file='mesh.in',form='formatted')
      open (unit=11,status='unknown',file='mesh.out',form='formatted')
      maxrel=12000
      maxint=1200
      ibug=1
c
c---- read title card and control data
      read(iin,'(a80)')card
      read(iin,'(a80)')uchar
      read(iin,'(8i5)')ietype,numnp,ndof,numel,nnpe,nsd,nmat,nedge
      write(iout,'(a80)')card
      write(iout,'(a80)')uchar
c
      write(iout,2001)
      neq=numnp*ndof
      call wi('numnp   ',numnp ,1)
      call wi('ndof    ',ndof  ,1)
      call wi('numel   ',numel ,1)
      call wi('nnpe    ',nnpe  ,1)
      call wi('nsd     ',nsd   ,1)
```

```
      call wi('nmat    ',nmat  ,1)
      call wi('neq     ',neq   ,1)
c
c---- initialize memory pointers
      mpcord=1
      mpfext=mpcord+numnp*nsd
      mpdisp=mpfext+numnp*ndof
      mpsurf=mpdisp+numel*nedge*2
      mptemp=mpsurf+numnp*ndof
      mpwork=mptemp+numnp
      mpstif=mpwork+numnp*ndof
      mpend=mpstif
c
      ipkfix=1
      ipnod=ipkfix+numnp*ndof
      ipiadr=ipnod+numel*nnpe
      ipmat=ipiadr+numnp*ndof
      ipend=ipmat+numel
c
      write(iout,2002)
      call wi('mpcord  ',mpcord,1)
      call wi('mpfext  ',mpfext,1)
      call wi('mpdisp  ',mpdisp,1)
      call wi('mpsurf  ',mpsurf,1)
      call wi('mptemp  ',mptemp,1)
      call wi('mpwork  ',mpwork,1)
      call wi('mpstif  ',mpstif,1)
      call wi('ipkfix  ',ipkfix,1)
      call wi('ipnod   ',ipnod ,1)
      call wi('ipiadr  ',ipiadr,1)
      call wi('ipmat   ',ipmat ,1)
      call wi('ipend   ',ipend ,1)
      call wi('maxrel  ',maxrel,1)
      call wi('maxint  ',maxint,1)
c
c---- check available memory
      call mcheck(mpend,ipend,maxrel,maxint)
c
c---- complete data input
c
      call inputf(a(mpcord),a(mpfext),a(mpsurf),a(mptemp),
     .            iaa(ipkfix),iaa(ipnod),iaa(ipmat),xrv)
c
c---- determine eqn-system column heights, diag addresses,length and
c     mean semi-bandwidth, mband
      call colht(iaa(ipnod),iaa(ipkfix),iaa(ipiadr),neq,numnp,numel,
     .           nnpe,ndof,length,mband)

c
c---- allocate storage for stiffness matrix
      mpend=mpstif+length
      write(iout,2003)
      call wi('mband   ',mband,1)
      call wi('length  ',length,1)
      call wi('mpend   ',mpend ,1)
c
c---- check available memory
      call mcheck(mpend,ipend,maxrel,maxint)
c
c---- clear stiffness
```

```
      do 10 i=mpstif,mpstif+length-1
   10 a(i)=0.
c
c---- form stiffness matrix
      call axistf(a(mpcord),iaa(ipkfix),iaa(ipnod),iaa(ipiadr)
     .      ,iaa(ipmat),a(mpstif),a(mpfext),a(mpsurf),a(mptemp))
c
c---- modify stiffness and load vector to account for prescribed displ
      call modify(a(mpstif),a(mpfext),iaa(ipkfix),iaa(ipiadr),
     .     neq,length)
c
c---- factorize stiffness and check for possible error in factorization
      iop=1
      tol=1.e-06
      zero=0.
      call trfact(a(mpstif),zero,a(mpwork),iaa(ipiadr),tol,neq,length,
     .            ierror,iposdf,iout,iop)
      if(iposdf.ne.0) call wi('iposdf  ',iposdf,1)
c
c---- transfer external force to displacement array
      do 30 i=1,neq
30    a(mpdisp+i-1)=a(mpfext+i-1)
c
c---- solve simultaneous equations
      iop=2
      zero=0.
      izero=0
      call trfact(a(mpstif),a(mpdisp),zero,iaa(ipiadr),zero,neq,length,
     .            izero,izero,izero,iop)
c
c---- output displacement solution
c
      write(iout,2004)
      write(iout,2005)
      write(iout,2006)
     . ((i,j,a(mpdisp+(i-1)*ndof+j-1),j=1,ndof),i=1,numnp)
c
c---- post-process data
      call axistr(a(mpcord),iaa(ipnod),iaa(ipmat),a(mpdisp),a(mptemp)
     .                  ,var)
c
      close(iin)
      close(iout)
c
      return
c
2001  format(//,21h program control data,/,1x,20(1h-))
2002  format(//,39h real and integer array memory pointers,/,
     .       1x,38(1h-))
2003  format(36h stiffness matrix storage parameters,/,1x,35(1h-))
2004  format(//,21h nodal point solution,2x,
     .       39h(node #, nodal dof #, value of the dof),/,
     .       1x,43(1h-))
2005  format(1x,37hnode        Ur                     Uz)
2006  format(2(2i4,e13.5,2x))
c
      end
```

```
c-------------------------------------------------------------
c*************************************************************
c     input.f
c-------------------------------------------------------------
c=============================================================
      subroutine inputf(x,f,surfld,temp,kfix,nod,mat,xrv)
c
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      common /rmatrl / rmat(10,10)
      common /device/ iin,iout,ibug
      dimension x(nsd,numnp),f(ndof,numnp),kk(6),kfix(ndof,numnp),
     .          nod(nnpe,numel),mat(numel),temp(numnp),
     .          surfld(2*nedge,numel),xrv(1),const(5)
      character*80 uchar
c
      read(iin,'(a80)')uchar
c
c---- read nodal data
      do 100 i=1,numnp
      read(iin,2000)n,(kk(j),j=1,6),(x(k,i),k=1,nsd),
     . (f(l,i),l=1,ndof)
      do 50 j=1,ndof
      kfix(j,n)=kk(j)
 50   continue
100   continue
c
      read(iin,'(a80)')uchar
c
c---- element data
      do 200 i=1,numel
        read(iin,3000)n,mat(i),(nod(j,i),j=1,nnpe)
200   continue
c
      read(iin,'(a80)')uchar
c
      do 250 i=1,numel
        read(iin,3500)n,(surfld(k,i),k=1,2*nedge)
250   continue
c
      read(iin,'(a80)')uchar
c
c---- read material properties
      do 300 i=1,nmat
        rmat(6,i)=xrv(2*i)
        volrat=xrv(2*i-1)
        call lookup(volrat, const)
        do 300 j=1,5
        rmat(j,i)=const(j)
300   continue
      write(*,4000)n, (rmat(j,1),j=1,5)
c
      read(iin,'(a80)')uchar
      return
c
1001  format(2x,60(1h-))
2000  format(i5,4x,6i1,2e12.4,/,15x,2e12.4)
3000  format(10i5)
3500  format(i5,8e12.4)
```

```
4000  format(i5,10e12.4)
c
      end

c=======================================================================
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Subroutine to get information from property look-up table
c     volrat: volume ratio
c     const: material constants
c
      subroutine lookup(volrat, const)
c
      implicit double precision (a-h,o-z),integer(i-n)
      common /device/ iin,iout,ibug
      dimension const(5), table(5,5)
c
      open (unit=15,status='old',file='table.txt',form='formatted')
      do 10 i=1,5
        read (15, '(5e12.4)') (table(i,j),j=1,5)
10    continue
c
      if (volrat.gt.0.7 .or. volrat.lt.0.3) then
        write(iout, *) 'Not a valid volume ratio'
        stop
      end if
c
      if (volrat.ge.0.3 .and. volrat.lt.0.4) then
         do 20 i=1,5
           const(i)=(table(1,i)*(0.4-volrat)+table(2,i)*(volrat-0.3))
     .                    /(0.4-0.3)
20       continue
      else if (volrat.ge.0.4 .and. volrat.lt.0.5) then
         do 30 i=1,5
           const(i)=(table(2,i)*(0.5-volrat)+table(3,i)*(volrat-0.4))
     .                    /(0.5-0.4)
30       continue
      else if (volrat.ge.0.5 .and. volrat.lt.0.6) then
         do 40 i=1,5
           const(i)=(table(3,i)*(0.6-volrat)+table(4,i)*(volrat-0.5))
     .                    /(0.6-0.5)
40       continue
      else if (volrat.ge.0.6 .and. volrat.le.0.7) then
         do 50 i=1,5
           const(i)=(table(4,i)*(0.7-volrat)+table(5,i)*(volrat-0.6))
     .                    /(0.7-0.6)
50       continue
      end if
c
      close(15)
      return
c
      end
```

```
c-------------------------------------------------------------
c*************************************************************
c     adstif.f
c-------------------------------------------------------------
c=============================================================
      subroutine adstif(kfix,s,fext,nod,iadres,t,re,itrow,kele,iop)
      implicit double precision (a-h,o-z), integer (i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      dimension kfix(neq),s(length),fext(neq),nod(nnpe,numel),
     .          iadres(neq),t(itrow,itrow),re(itrow)
c
c---- assemble element stiffness and optionally element load vector
c     into global stiffness matrix and global load vector.
c     homogeneous b.c. are enforced by not assembling diagonal,
c     row and column terms; it is assumed that corresponding force
c     terms have been set to zero by input data.
c     usage:
c     kfix   fixity array. for dof number m, fixity is:
c            if kfix(m)=0 unconstrained
c                      =1 zero prescribed value
c                      =2 nonzero prescribed value
c
c     s      global stiffness matrix.
c
c     fext   external force vector.
c
c     nod    element nodal connectivity array.
c
c     iadres array containing storage locations of diagonal terms of
c            stiffness matrix.  not used by this subroutine directly,
c            but necessary for subroutine locsky.
c
c     t      element stiffness matrix .
c
c     re     if iop=0, not used.
c            if iop=1, element load vector.
c
c     itrow  row and column dimension of t (and re, if used) in calling
c            program.
c
c     kele   element number being assembled (input).
c
c     iop    if iop=0, assemble element stiffness matrix only.
c            if iop=1, assemble element stiffness matrix and element
c                      load vector.
c
c
      do 100 igen=1,nnpe
      do 100 jgen=1,nnpe
      iglb=ndof*(nod(igen,kele)-1)
      jglb=ndof*(nod(jgen,kele)-1)
c
      do 100 idof=1,ndof
      do 100 jdof=1,ndof
c---- if lower triangular element, skip assembly
      if(iglb+idof.gt.jglb+jdof) go to 100
c---- enforce boundary condition:
c---- if zero-constrained dof, skip assembly
      if(kfix(iglb+idof).eq.1 .or. kfix(jglb+jdof).eq.1) go to 100
```

```
c
      jadd=locsky(iglb+idof,jglb+jdof,iadres,neq)
      s(jadd)=s(jadd)+t(ndof*(igen-1)+idof,ndof*(jgen-1)+jdof)
  100 continue
c
      if(iop.ne.1) return
c---- assemble element load vector into global load vector
      do 200 igen=1,nnpe
      iglb=ndof*(nod(igen,kele)-1)
      do 200 idof=1,ndof
c---- skip assembly if this is a zero or nonzero prescribed dof
      if(kfix(iglb+idof).ge.1) go to 200
      fext(iglb+idof)=fext(iglb+idof)+re(ndof*(igen-1)+idof)
  200 continue
c
      return
      end

c=============================================================
c     called by femcod
c     call modify(a(mpstif),a(mpfext),ia(ipkfix),ia(ipiadr),neq,length)
c
      subroutine modify(s,fext,kfix,iadres,neq,length)
      implicit double precision (a-h,o-z), integer (i-n)
      dimension kfix(neq),fext(neq),s(length),iadres(neq)
c
c---- subroutine to modify the stiffness matrix and load vector to
c     account for zero and nonzero prescribed displacements.
c     information for dof m contained in fext is assumed to be:
c
c     fext(m)= force or load if dof m is unconstrained (kfix(m)=0)
c            = 0. if dof m is zero prescribed (kfix(m)=1)
c            = value of the nonzero prescribed displacement if dof m
c              is nonzero prescribed (kfix(m)=2)
c
c---- scan all dof and modify for those that are prescribed
      do 100 k=1,neq
c
      if(kfix(k).ne.1) go to 50
c---- d.o.f. has prescribed zero value--enter unit value on diagonal
c---- of stiffness matrix (off-diag. terms made zero by sub. adstif)
      kdiag=iadres(k)
      s(kdiag)=1.0
      go to 100
c
   50 if(kfix(k).ne.2) go to 100
c---- d.o.f. has prescribed nonzero value:
c---- modify load vector then zero off-diagonal stiffness coeff.
      kdiag=iadres(k)
      ktop=kdiag
      if(k.ge.2) ktop=iadres(k-1)+1
      keqtop=k+ktop-kdiag
      mpnter=ktop-1
c---- loop over stiffness coefficients in columnwise order
      do 10 m=keqtop,k-1
      mpnter=mpnter+1
c---- skip modification if dof  m  is also a nonzero prescribed dof
      if(kfix(m).ne.2) fext(m)=fext(m)-s(mpnter)*fext(k)
      s(mpnter)=0.
   10 continue
```

```
c
c---- loop over stiffness coefficients in rowwise order using function
c     subroutine locsky to return the 1-d stiffness address
      do 20 meq=k+1,neq
      kmeqlo=locsky(k,meq,iadres,neq)
c---- if stiffness coefficient is outside skyline, skip to end of loop
      if(kmeqlo.eq.0) go to 20
c---- skip modification if dof meq is also a nonzero prescribed dof
      if(kfix(meq).ne.2) fext(meq)=fext(meq)-s(kmeqlo)*fext(k)
      s(kmeqlo)=0.
   20 continue
c---- modify diagonal term for equation k
      s(kdiag)=1.0
c
  100 continue
      return
      end

c=============================================================
c     called by femcod
c     call trfact(a(mpstif),zero,a(mpwork),ia(ipiadr),tol,neq,length,
c    .    ierror,iposdf,iout,1)
c     call trfact(a(mpstif),a(mpdisp),zero,ia(ipiadr),zero,neq,length,
c    .    izero,izero,izero,2)
c
      subroutine trfact(a,b,g,iadres,tol,n,length,ierror,iposdf,iout,
     .                  iop)
      implicit double precision (a-h,o-z), integer (i-n)
      dimension a(length),b(n),g(n),iadres(n)
c
c---- given the system of simultaneous equations ax=b, a symmetric and
c     b specified, this subroutine triple-factors a into u(trans)*d*u
c     and then performs forward/back substitution to obtain the solu-
c     tion x.  a can be positive or negative definite.  warnings are
c     issued if the subroutine detects possible singularity or lack of
c     positive definiteness.  this subroutine employs the fortran77
c     standard in which do loops are not executed if the beginning
c     index is larger than the ending index.
c
c     usage:
c     a      for iop=1 or 3, a contains, on input, the a-coefficients
c            of ax=b stored in compacted column (skyline) form accord-
c            ing to the scheme:
c
c             e.g.,   i a(1)   a(2)        a(6)    i
c                     i        a(3)  a(4)  a(7)    i
c                     i              a(5)  a(8)    i
c                     i                    a(9) ...i
c                     i                      :     i
c
c            on output, d and u overwrite the diagonal and upper-diag.
c            coefficients of a, respectively.
c            for iop=2, a must contain, on input, the coefficients of
c            d and u, which were obtained from a prior call to sub-
c            routine trfact with iop=1 or 3.
c
c     b      for iop=1, not used.
c            for iop=2 or 3, b contains, on input, the specified right-
c            hand-side and on output, the solution x overwrites b.
c
```

```
c     g      for iop=1 or 3, work vector of length n.
c            for iop=2, not used.
c
c     iadres input vector of length n that contains the addresses of
c            the diagonal coefficients of a.
c
c     tol    for iop=1 or 3, input parameter specifying tolerance for
c            decay of diagonal coefficients.  taking tol=10**(-p) will
c            indicate an error (nonzero value of ierror and a message
c            printed to i/o unit # iout) if the p leading digits of any
c            pivot element are lost compared to its original value.
c            for general purpose finite element equation solving, a
c            value for p equal to about one-half the number of digits
c            for floating point numbers is appropriate.  larger and
c            smaller values of p provide for less stringent and more
c            stringent tests on diagnonal decay, respectively.
c            for iop=2, not used.
c
c     n      number of equations (input).
c
c     length length in words of 1-d array a (input).
c
c     ierror for iop=1 or 3, on output ierror=0 if abs(d(k,k)/a(k,k))
c            is greater than tol for all k between 1 and n.  otherwise,
c            ierror is equal to the row number of the first equation
c            that did not satisfy the test.  failure of the test is
c            also printed by the subroutine on i/o unit number iout
c            if iout is greater than zero, otherwise the message is
c            suppressed.
c            for iop=2, not used.
c
c     iposdf for iop=1 or 3, on output iposdf=0 if the matrix is pos-
c            itive definite, and iposdf=-1 if the matrix is not
c            positive definite.
c
c     iout   for iop=1 or 3, input parameter specifying input/output
c            unit number for printing of warning message in the event
c            that diagonal decay is detected.  to suppress printing of
c            the message, iout should be zero or negative.
c            for iop=2, not used.
c
c     iop    input parameter specifying solution option:
c               iop=1  factorization only
c                  =2  forward/back substitution only
c                  =3  factorization and forward/back substitution
c
c
      if(iop.eq.2) go to 50
c
c---- factorization
c
      ierror=0
      iposdf=0
c
      do 40 k=2,n
      kdiag=iadres(k)
      ktop=iadres(k-1)+1
      keqtop=k+ktop-kdiag
      ipnter=ktop-1
c
```

```
      do 20 i=keqtop,k-1
      ipnter=ipnter+1
      idiag=iadres(i)
      itop=idiag
      if(i.ge.2) itop=iadres(i-1)+1
      ieqtop=i+itop-idiag
c
      sum=0.
      mpnter=itop-1
c---- in following do loop, skip over g's that are above skyline
      if(keqtop.gt.ieqtop) mpnter=mpnter+keqtop-ieqtop
      do 10 m=max(keqtop,ieqtop),i-1
      mpnter=mpnter+1
   10 sum=sum+a(mpnter)*g(m)
      a(ipnter)=(a(ipnter)-sum)/a(idiag)
   20 g(i)=a(idiag)*a(ipnter)
c
      sum=0.
      mpnter=ktop-1
      do 30 m=keqtop,k-1
      mpnter=mpnter+1
   30 sum=sum+a(mpnter)*g(m)
      temp=a(kdiag)
      a(kdiag)=a(kdiag)-sum
c---- check for decay of diag. coeff. and positive definiteness
      if(ierror.gt.0) go to 40
      ratio=a(kdiag)/temp
      if(abs(ratio).gt.tol) go to 40
      ierror=k
      if(iout.gt.0) write(iout,1000)ratio,ierror
   40 if(a(kdiag).lt.0.) iposdf=-1
 1000 format(///,34h subroutine trfact  w a r n i n g:,/,
     .       24h diagonal decay ratio of,e10.3,
     .       21h detected in equation,i6,///)
      if(iop.eq.1) return
c
c
   50 continue
c
c---- forward substitution
      do 70 k=2,n
      kdiag=iadres(k)
      ktop=iadres(k-1)+1
      keqtop=k+ktop-kdiag
      mpnter=ktop-1
      sum=0.
      do 60 m=keqtop,k-1
      mpnter=mpnter+1
   60 sum=sum+a(mpnter)*b(m)
   70 b(k)=b(k)-sum
c
c---- back substitution
      do 80 k=1,n
      kdiag=iadres(k)
   80 b(k)=b(k)/a(kdiag)
      do 90 k=n,2,-1
      kdiag=iadres(k)
      ktop=iadres(k-1)+1
      keqtop=k+ktop-kdiag
      mpnter=ktop-1
```

```
      do 90 m=keqtop,k-1
      mpnter=mpnter+1
   90 b(m)=b(m)-a(mpnter)*b(k)
      return
      end

c=============================================================
      subroutine mcheck(mpend,ipend,maxrel,maxint)
      implicit double precision (a-h,o-z),integer(i-n)
      common /device/ iin,iout,ibug
c
c---- check if memory requirements (mpend and ipend) exceed the
c     maximum storage declared in calling program (maxrel and maxint)
c     if so, issue error message and stop program execution
c
      if(mpend.le.maxrel .and. ipend.le.maxint) return
      write(iout,100)
      stop
  100 format(///,34h memory error: insufficient memory ///,
     .           50h program execution terminated by subroutine mcheck)
      end

c=============================================================
      subroutine colht(nod,kfix,iadres,neq,numnp,numel,nnpe,ndof,
     .                 length,mband)
      implicit double precision (a-h,o-z), integer (i-n)
      dimension iadres(neq),nod(nnpe,numel),kfix(ndof,numnp)
c
c---- scan element connectivity to determine the column heights and
c     diagonal storage locations for 1-dimensional compact column
c     storage according to the scheme:
c
c             e.g.,   i a(1)   a(2)        a(6)    i
c                     i        a(3)  a(4)  a(7)    i
c                     i              a(5)  a(8)    i
c                     i                    a(9) ...i
c                     i                      :     i
c
c     this subroutine first scans the dof of all elements to determine
c     column heights which are stored in iadres.  zero-dispacement dof
c     do not influence column heights.  second, the locations in a 1-d
c     array of the diagonal coefficients is computed and stored in
c     iadres.  note that the total amount of storage for the 1-d
c     compacted stiffness matrix is equal to iadres(neq).  also comput-
c     ed is the mean, or average, semi-bandwidth, mband.
c
c---- set initial column heights (i.e., heights for a diagonal matrix)
      do 10 i=1,neq
   10 iadres(i)=1
c
c---- loop over number of elements
      do 30 n=1,numel
c
c---- loop over combinations of nodes in element n
      do 30 iii=1,nnpe
      ii=nod(iii,n)
      do 30 jjj=1,nnpe
      jj=nod(jjj,n)
c
c---- loop over combinations of dof for global node numbers ii and jj
```

```
      do 30 i=1,ndof
c---- if this dof has prescribed zero value, skip to end of loop
      if(kfix(i,ii).eq.1) go to 30
      idof=(ii-1)*ndof+i
      do 20 j=1,ndof
c---- if this dof has prescribed zero value, skip to end of loop
      if(kfix(j,jj).eq.1) go to 20
      jdof=(jj-1)*ndof+j
c---- if idof,jdof is on diagonal or below, skip to end of loop
      if(idof.ge.jdof) go to 20
c---- idof,jdof location is above diagonal--adjust column height
      newhgt=jdof-idof+1
      if(newhgt.gt.iadres(jdof)) iadres(jdof)=newhgt
   20 continue
   30 continue
c
c
c---- compute addresses of diagonal entries using column heights
      do 50 i=2,neq
   50 iadres(i)=iadres(i-1)+iadres(i)
      length=iadres(neq)
c
c---- compute mean semi-bandwidth, mband
      mband=ifix((2.*float(neq)+1.-sqrt((2.*float(neq)+1.)**2-
     .      8.*float(length)))/2.)
      return
      end

c=================================================================
      function locsky(i,j,iadres,neq)
      implicit double precision (a-h,o-z), integer (i-n)
      dimension iadres(neq)
c
c---- this function subprogram provides the address in a 1-d
c     array corresponding to an i,j address in a 2-d array that
c     is stored in compact column (skyline) form.
c
c     usage:
c     i,j    row and column numbers (input).
c
c     iadres input array of length neq that contains the addresses in
c            a 1-d array for the diagonal entries in a 2-d array.
c
c     neq    number of rows (equations).
c
c     locsky on output, equal to the 1-d address for (i,j) unless
c            (i,j) is below diagonal or above skyline, in which
c            case a value of zero is returned.
c
c
c---- check if (i,j) is below diagonal
      if(i.gt.j) go to 10
c
      jdiag=iadres(j)
      jtop=jdiag
      if(j.gt.1) jtop=iadres(j-1)+1
      locsky=jdiag-j+i
c
c---- check if (i,j) is above skyline
      if(locsky.ge.jtop) return
```

```
c
c---- (i,j) location is either below diagonal or above skyline
   10 locsky=0
      return
      end

c=======================================================================
      subroutine wi(id,i,iop)
      implicit double precision (a-h,o-z), integer (i-n)
      common /device/ iin,iout,ibug
      character*8 id
      if(iop.gt.ibug) return
      write(iout,100)id,i
  100 format(1h ,a8,1h=,i10)
      return
      end

c=======================================================================
      subroutine wr(id,r,iop)
      implicit double precision (a-h,o-z), integer (i-n)
      common /device/ iin,iout,ibug
      character*8 id
      if(iop.gt.ibug) return
      write(iout,100)id,r
  100 format(1h ,a8,1h=,g15.5)
      return
      end
```

```
c-------------------------------------------------------------
c*************************************************************
c     axis.f
c-------------------------------------------------------------
c=============================================================
      subroutine axistf(xx,kfix,nod,iadres,matnum,s,fext,surfld,tt)
c
c      call axistf(a(mpcord),ia(ipkfix),ia(ipnod),ia(ipiadr),ia(ipmat),
c     .            a(mpstif),a(mpfext),a(mpsurf),a(mptemp))
c
c
c     form stiffness matrix for 8-node axisymmetric element
c
c     written by Zheng Huang and Wei Yang               April, 1999
c
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      common /rmatrl/ rmat(10,10)
      common /device/ iin,iout,ibug
      dimension xx(nsd,numnp),kfix(neq),nod(nnpe,numel),
     .          iadres(neq),matnum(numel),s(length),fext(neq),
     .          surfld(2*nedge,numel),x(2,8)
c                                      x(nsd, nnpe)
      dimension xi(3,3),eta(3,3),e(4,4),bte(16,4),eps0(4)
     .          ,stfk(16,16),ww(3),ret(16),res(16),re(16)
     .          ,b(4,16),elk(16,16),dtp(8),sfld(2,4),rett(16)
      dimension shapef(8),dsfxi(8),dsfeta(8)
      double precision jj(2,2)
c
c---- define integration point in local coordinates
      do 10 i=1,3
      eta(i,1)=-sqrt(0.6)
      eta(i,2)=0.0
      eta(i,3)=sqrt(0.6)
10    continue
      do 20 j=1,3
      xi(1,j)=-sqrt(0.6)
      xi(2,j)=0.0
      xi(3,j)=sqrt(0.6)
20    continue
c---- weight factors
      ww(1)=5.0/9.0
      ww(2)=8.0/9.0
      ww(3)=5.0/9.0
c
c---- Assemble stiffness matrix for each element
      do 100 n=1,numel
c
c---- get nodal coordinates and temperature
      do 40 i=1,nnpe
        x(1,i)=xx(1,nod(i,n))
        x(2,i)=xx(2,nod(i,n))
c        dtp(i)=tt(nod(i,n))
40    continue
c
c---- get element surface load
      do 45 i=1,nedge
        sfld(1,i)=surfld(2*i-1,n)
        sfld(2,i)=surfld(2*i,n)
```

```
45    continue
c
c---- evaluate material matrix, e, for axisymmetrix element
      mat=matnum(n)
c      call ematrx(e,mat)
c      alpha=rmat(3,mat)
ccccc
      call ematrx(e,mat)
      thick=rmat(7,mat)
c
c---- initialize element stiffness matrix to 0.
      do 60 i=1,16
        do 60 j=1,16
          elk(i,j)=0.0
60    continue
c
c---- inital thermal load vertor
      do 65 i=1,16
        ret(i)=0.
65    continue
c
c---- loop over each integration point
      do 30 i=1,3
      do 30 j=1,3
        vxi=xi(i,j)
        veta=eta(i,j)
c
c----   compute shape function and shape function derivatives
        call findn(vxi,veta,x,shapef,dsfxi,dsfeta,r,jj,detj)
c
c----   compute B matrix
        call findb(shapef,dsfxi,dsfeta,r,jj,detj,b)
c
c----   sum of element stiffness matrix at integration points
        call findk(b,e,r,detj,stfk,bte,thick)
c
c----   numerical integration (sum at integration point)
        do 50 is=1,16
        do 50 js=1,16
          elk(is,js)=elk(is,js)+stfk(is,js)*ww(i)*ww(j)
50      continue
c
c----   compute thermal load vector 'ret(16)'
c        call axitlv(alpha,bte,dtp,shapef,r,detj,rett,eps0)
c
c----   numerical integration
c        do 55 is=1,16
c          ret(is)=ret(is)+rett(is)*ww(i)*ww(j)
c55      continue
c
30    continue
c
c---- compute surface load vector
c     ip=1  surface load in normal and tangential direction
c     ip=2  surfcae load in global coordinates
c     call axislv(x,sfld,res,ip)
c
      call axislv(x,sfld,res,1,thick)
c
c---- compute total element load vector
```

```
      do 70 i=1,16
c        re(i)=ret(i)+res(i)
        re(i)=res(i)
70    continue
c
c---- assemble stiffness matrix and element load vector
      call adstif(kfix,s,fext,nod,iadres,elk,re,16,n,1)
c
100   continue
c
      return
      end

c=======================================================================
      subroutine axislv(x,sfld,res,ip,thick)
c
c     ip=1  surface load in normal and tangential direction
c     ip=2  surfcae load in global coordinates
c
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      dimension xi(2,4), eta(2,4)
      dimension shapef(8),dsfxi(8),dsfeta(8),x(2,8),ww(2)
      dimension sfld(2,4),res(16),vres(16,4)
      double precision nst(16,2),jj(2,2),js(2),jt
c
c---- Integration points on the edges
      vt=1./sqrt(3.)
      xi(1,1)=-vt
      xi(2,1)=vt
      eta(1,1)=-1.0
      eta(2,1)=-1.0
      xi(1,2)=1.
      xi(2,2)=1.
      eta(1,2)=-vt
      eta(2,2)=vt
      xi(1,3)=-vt
      xi(2,3)=vt
      eta(1,3)=1.0
      eta(2,3)=1.0
      xi(1,4)=-1.
      xi(2,4)=-1.
      eta(1,4)=-vt
      eta(2,4)=vt
c
c---- weight factor
      ww(1)=1.
      ww(2)=1.
c
c---- initialize res(j)
      do 60 j=1,16
        res(j)=0.
60    continue
c
c---- evaluate surface traction along edges
      do 80 n=1,nedge
c
c----   initialzie vres(i,n)
        do 90 j=1,16
```

```
90      vres(j,n)=0.
c
        do 30 i=1,2
c
          vxi=xi(i,n)
          veta=eta(i,n)
c----     evaluate shape functions at integration point
          call findn(vxi,veta,x,shapef,dsfxi,dsfeta,r,jj,detj)
c
c----     clear Ns(transpose)
          do 10 is=1,16
            do 10 j=1,2
              nst(is,j)=0.
10        continue
c----     computer Ns(transpose)
          do 20 is=1,8
            nst(2*is-1,1)=shapef(is)
            nst(2*is,2)=shapef(is)
20        continue
c
          if (ip.eq.1) then
c----       local surface load in
c           sfld(1,n): surface load in normal direction
c           sfld(2,n): surface load in tangential direction
c----       evaluate 'Js'
            if (n.eq.1) then
              js(1)=-sfld(2,n)*jj(1,1)+sfld(1,n)*jj(1,2)
              js(2)=-sfld(1,n)*jj(1,1)-sfld(2,n)*jj(1,2)
            else if(n.eq.2) then
              js(1)=sfld(2,n)*jj(2,1)+sfld(1,n)*jj(2,2)
              js(2)=sfld(1,n)*jj(2,1)-sfld(2,n)*jj(2,2)
            else if(n.eq.3) then
              js(1)=sfld(2,n)*jj(1,1)-sfld(1,n)*jj(1,2)
              js(2)=sfld(1,n)*jj(1,1)+sfld(2,n)*jj(1,2)
            else if(n.eq.4) then
              js(1)=-sfld(2,n)*jj(2,1)-sfld(1,n)*jj(2,2)
              js(2)=-sfld(1,n)*jj(2,1)+sfld(2,n)*jj(2,2)
            end if
c
c----       Ns(transpose)*r*Js
            do 40 j=1,16
              v=0.
              do 41 k=1,2
                    if(ietype.eq.1) then
                  v=v+nst(j,k)*r*js(k)
                    else if (ietype.eq.2) then
                  v=v+nst(j,k)*js(k)
                    else
                  v=v+nst(j,k)*thick*js(k)
                    end if
41            continue
              vres(j,n)=vres(j,n)+v*ww(i)
40          continue
          elseif (ip.eq.2) then
c----       surface load in global coordinates
c           sfld(1,n): surface load in x direction
c           sfld(2,n): surface load in y direction
            if (n.eq.1) then
              jt=sqrt(jj(1,1)**2+jj(1,2)**2)
            else if(n.eq.2) then
```

```
              jt=sqrt(jj(2,1)**2+jj(2,2)**2)
            else if(n.eq.3) then
              jt=sqrt(jj(1,1)**2+jj(1,2)**2)
            else if(n.eq.4) then
              jt=sqrt(jj(2,1)**2+jj(2,2)**2)
            end if
          end if
c
c----       Ns(transpose)*r*Js
            do 45 j=1,16
              v=0.
              do 46 k=1,2
                    if(ietype.eq.1) then
                v=v+nst(j,k)*sfld(k,n)*r*jt
                  else if (ietype.eq.2) then
                v=v+nst(j,k)*sfld(k,n)*jt
                  else
                v=v+nst(j,k)*sfld(k,n)*thick*jt
                  end if
46            continue
              vres(j,n)=vres(j,n)+v*ww(i)
45          continue
c
30    continue
c
c---- compute element surface load vector
      do 50 j=1,16
      res(j)=res(j)+vres(j,n)
50    continue
c
80    continue
c
      return
      end

c=======================================================================
      subroutine ematrx(e,mat)
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      common /rmatrl / rmat(10,10)
      dimension e(4,4),temp(4,4)
      dimension trans(4,4), transt(4,4)
c
c---- evaluate material property matrix D
c
c     this subroutine calculate material constants
c     for material type mat:
c       rmat(1,mat)=modulus of elasticity E1
c       rmat(2,mat)=modulus of elasticity E2
c       rmat(3,mat)=poisson's ratio v12
c       rmat(4,mat)=poisson's ratio v23
c       rmat(5,mat)=G12
c
      ex1=rmat(1,mat)
      ex2=rmat(2,mat)
      v12=rmat(3,mat)
      v23=rmat(4,mat)
      g12=rmat(5,mat)
      v21=v12*ex2/ex1
```

```
      v32=v23
      v13=v12
      ex3=ex2
c
      if (ietype.eq.1) then
          delta=(1-v12*v12-v23*v32-v31*v13-2*v21*v32*v13)/(ex1*ex2*ex3)
          e(1,1)=(1-v23*v32)/(ex2*ex3*delta)
            e(1,2)=(v21+v31*v23)/(ex2*ex3*delta)
            e(1,3)=(v31+v21*v32)/(ex2*ex3*delta)
            e(1,4)=0
            e(2,1)=e(1,2)
            e(2,2)=(1-v13*v31)/(ex1*ex3*delta)
            e(2,3)=(v32+v12*v31)/(ex1*ex3*delta)
            e(2,4)=0
            e(3,1)=e(1,3)
            e(3,2)=e(2,3)
            e(3,3)=(1-v12*v21)/(ex1*ex2*delta)
            e(3,4)=0
            e(4,1)=0
            e(4,2)=0
            e(4,3)=0
            e(4,4)=g12
c
      else if (ietype.eq.2) then
          delta=(1-v12*v12-v23*v32-v31*v13-2*v21*v32*v13)/(ex1*ex2*ex3)
          e(1,1)=(1-v23*v32)/(ex2*ex3*delta)
            e(1,2)=0
            e(1,3)=(v31+v21*v32)/(ex2*ex3*delta)
            e(1,4)=0
            e(2,1)=0
            e(2,2)=0
            e(2,3)=0
            e(2,4)=0
            e(3,1)=e(1,3)
            e(3,2)=0
            e(3,3)=(1-v12*v21)/(ex1*ex2*delta)
            e(3,4)=0
            e(4,1)=0
            e(4,2)=0
            e(4,3)=0
            e(4,4)=g12
c
      else
          e(1,1)=ex1/(1-v12*v21)
            e(1,2)=0
            e(1,3)=(v12*ex2)/(1-v12*v21)
            e(1,4)=0
            e(2,1)=0
            e(2,2)=0
            e(2,3)=0
            e(2,4)=0
            e(3,1)=e(1,3)
            e(3,2)=0
            e(3,3)=ex2/(1-v12*v21)
            e(3,4)=0
            e(4,1)=0
            e(4,2)=0
            e(4,3)=0
            e(4,4)=g12
      end if
```

```
c
c---- E matrix tranformation
c-----the orientation angle takes effects.
      call etrans(trans,transt,mat)
      do 20 i=1,4
        do 20 j=1,4
            temp(i,j)=0
          do 20 k=1,4
            temp(i,j)=temp(i,j)+transt(i,k)*e(k,j)
20    continue
      do 30 i=1,4
        do 30 j=1,4
            e(i,j)=0
          do 30 k=1,4
            e(i,j)=e(i,j)+temp(i,k)*trans(k,j)
30    continue
c
      return
      end

c=================================================================
      subroutine etrans(trans,transt,mat)
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .                ,neq,length,nmat,nedge
      common /rmatrl / rmat(10,10)
      dimension trans(4,4), transt(4,4)
c
c
      theta=rmat(6,mat)
      c=cos(theta)
      s=sin(theta)
      c2=c*c
      s2=s*s
      cs=c*s
c
      trans(1,1)=c2
      trans(1,2)=0
      trans(1,3)=s2
      trans(1,4)=cs
      trans(2,1)=0
      trans(2,2)=1
      trans(2,3)=0
      trans(2,4)=0
      trans(3,1)=s2
      trans(3,2)=0
      trans(3,3)=c2
      trans(3,4)=-cs
      trans(4,1)=-2*cs
      trans(4,2)=0
      trans(4,3)=2*cs
      trans(4,4)=c2-s2
c
      do 10 i=1,4
        do 10 j=1,4
          transt(i,j)=trans(j,i)
10    continue
c
      return
      end
```

```
c-------------------------------------------------------------
c*************************************************************
c     axistr.f
c-------------------------------------------------------------
c=============================================================
      subroutine axistr(xx,nod,matnum,dd,tt,var)
c
c     axistr: 8-node quad stress
c
c---- compute and output state of stress in a quad (strs=e*b*d)
c
      implicit double precision (a-h,o-z), integer (i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      common /rmatrl / rmat(10,10)
      common /device/ iin,iout,ibug
      dimension xx(nsd,numnp),nod(nnpe,numel),matnum(numel),
     .          dd(ndof,numnp),tt(numnp),x(2,8),e(4,4),
     .          strs(4,8),eta(4),xi(4),dtp(8),
     .          str(4,150),idstr(150),bte(16,4),ret(16),
     .          xind(8),etand(8),sfg(4),strnd(4,8)
      dimension d(16),b(4,16),eps0(4),eps(4),bd(4)
      dimension shapef(8),dsfxi(8),dsfeta(8)
      double precision jj(2,2)
c
c---- define integration point
      xi(1)=-1/sqrt(3.)
      xi(2)=1/sqrt(3.)
      xi(3)=1/sqrt(3.)
      xi(4)=-1/sqrt(3.)
c
      eta(1)=-1/sqrt(3.)
      eta(2)=-1/sqrt(3.)
      eta(3)=1/sqrt(3.)
      eta(4)=1/sqrt(3.)
c
      xind(1)=-1
      xind(2)=1
      xind(3)=1
      xind(4)=-1
      xind(5)=0
      xind(6)=1
      xind(7)=0
      xind(8)=-1
c
      etand(1)=-1
      etand(2)=-1
      etand(3)=1
      etand(4)=1
      etand(5)=-1
      etand(6)=0
      etand(7)=1
      etand(8)=0
c
c---- intialize nodal stress
      do 11 i=1,numnp
        idstr(i)=0
        do 11 j=1,4
          str(j,i)=0.
11    continue
```

```
c
c---- loop over elements
      do 100 n=1,numel
c
c----   evaluate material matrix for plane stress
        mat=matnum(n)
c        alpha=rmat(3,mat)
        call ematrx(e,mat)
c
c----   collect nodal displacements
        do 25 i=1,nnpe
          do 25 j=1,ndof
            d((i-1)*ndof+j)=dd(j,nod(i,n))
25      continue
c
c----   get nodal coordinates and temperature
        do 40 i=1,nnpe
          x(1,i)=xx(1,nod(i,n))
          x(2,i)=xx(2,nod(i,n))
c          dtp(i)=tt(nod(i,n))
40      continue
c
c----   intialize 'B' and 'eps0'
        do 70 is=1,4
          eps0(is)=0.
          do 70 js=1,16
            b(is,js)=0.
70      continue
c
c----   initialize nodal stresses of the element
        do 75 ind=1,nnpe
          do 75 j=1,4
              strnd(j,ind)=0.
75      continue
c
c----   evaluate stress at integration points
        do 30 i=1,4
          vxi=xi(i)
          veta=eta(i)
c
c----     compute shape function and shape function derivatives
          call findn(vxi,veta,x,shapef,dsfxi,dsfeta,r,jj,detj)
c
c----     compute B matrix
          call findb(shapef,dsfxi,dsfeta,r,jj,detj,b)
c
c----     compute thermal load vector
c          call axitlv(alpha,bte,dtp,shapef,r,detj,ret,eps0)
c
c----     compute b*d
          do 45 j=1,4
            bd(j)=0.
            do 45 k=1,16
              bd(j)=bd(j)+b(j,k)*d(k)
45        continue
c
c----     substract initial strain
          do 80 j=1,4
c            eps(j)=bd(j)-eps0(j)
            eps(j)=bd(j)
```

```
80        continue
c
c----     compute e*eps
          do 60 j=1,4
            strs(j,i)=0.
            do 60 k=1,4
              strs(j,i)=strs(j,i)+e(j,k)*eps(k)
60        continue
c
          do 65 ind=1,nnpe
            vr=xind(ind)*sqrt(3.)
            vs=etand(ind)*sqrt(3.)
            sfg(1)=(1-vr)*(1-vs)/4.
            sfg(2)=(1+vr)*(1-vs)/4.
            sfg(3)=(1+vr)*(1+vs)/4.
            sfg(4)=(1-vr)*(1+vs)/4.
            do 66 j=1,4
              strnd(j,ind)=strnd(j,ind)+strs(j,i)*sfg(i)
66          continue
65        continue
c
30      continue
c
c----   sum nodal stresses from different elements
        do 35 ind=1,nnpe
          ip=nod(ind,n)
          do 34 j=1,4
            str(j,ip)=str(j,ip)+strnd(j,ind)
34      continue
c----   A counter that tells how many nodal stress value added
        idstr(ip)=idstr(ip)+1
35      continue
c
100   continue
c
c---- compute average nodal stresses
      var=0.0
      do 10 i=1,numnp
        do 10 j=1,4
          str(j,i)=str(j,i)/idstr(i)
            if (str(1,i).gt.var) var=str(1,i)
10    continue
c
c---- output results
      write(iout,1000)
      write(iout,2000)
      do 20 n=1,numnp
        write(iout,1001)n,(str(i,n),i=1,4)
20    continue
c
      return
1000  format(//,16h  Nodal stresses,/,1x,55(1h-))
2000  format(1x
     .   ,53hNode     sigmaR     sigmaTheta     sigmaZ       taoZR)
1001  format(i5,4e13.4)
      end
```

```
c-------------------------------------------------------------
c*************************************************************
c     findb.f
c-------------------------------------------------------------
c=============================================================
      subroutine findn(xi,eta,x,shapef,dsfxi,dsfeta,r,jj,detj)
c
c---- compute shape funciton and shape function derivatives
c     at integration points
c
      implicit double precision (a-h,o-z),integer(i-n)
      dimension shapef(8),dsfxi(8),dsfeta(8),x(2,8)
      double precision jj(2,2)
c
c---- evaluate shape functions at integration point
      shapef(5)=(1-xi**2)*(1-eta)/2.
      shapef(6)=(1+xi)*(1-eta**2)/2.
      shapef(7)=(1-xi**2)*(1+eta)/2.
      shapef(8)=(1-xi)*(1-eta**2)/2.
      shapef(1)=(1-xi)*(1-eta)/4.-(shapef(8)+shapef(5))/2.
      shapef(2)=(1+xi)*(1-eta)/4.-(shapef(6)+shapef(5))/2.
      shapef(3)=(1+xi)*(1+eta)/4.-(shapef(6)+shapef(7))/2.
      shapef(4)=(1-xi)*(1+eta)/4.-(shapef(8)+shapef(7))/2.
c
c---- evaluate shape function deravitives at integration point
      dsfxi(5)=-xi*(1-eta)
      dsfxi(6)=(1-eta**2)/2.
      dsfxi(7)=-xi*(1+eta)
      dsfxi(8)=-(1-eta**2)/2.
      dsfxi(1)=-(1-eta)/4.-(dsfxi(8)+dsfxi(5))/2.
      dsfxi(2)=(1-eta)/4.-(dsfxi(6)+dsfxi(5))/2.
      dsfxi(3)=(1+eta)/4.-(dsfxi(6)+dsfxi(7))/2.
      dsfxi(4)=-(1+eta)/4.-(dsfxi(8)+dsfxi(7))/2.
c
      dsfeta(5)=-(1-xi**2)/2.
      dsfeta(6)=-(1+xi)*eta
      dsfeta(7)=(1-xi**2)/2.
      dsfeta(8)=-(1-xi)*eta
      dsfeta(1)=-(1-xi)/4.-(dsfeta(8)+dsfeta(5))/2.
      dsfeta(2)=-(1+xi)/4.-(dsfeta(6)+dsfeta(5))/2.
      dsfeta(3)=(1+xi)/4.-(dsfeta(6)+dsfeta(7))/2.
      dsfeta(4)=(1-xi)/4.-(dsfeta(8)+dsfeta(7))/2.
30    continue
c
c---- evaluate dz/deta, dr/dxi, dz/dxi,dr/deta => jj
      drdxi=0.0
      drdeta=0.0
      dzdxi=0.0
      dzdeta=0.0
      do 40 is=1,8
      drdxi=drdxi+dsfxi(is)*x(1,is)
      drdeta=drdeta+dsfeta(is)*x(1,is)
      dzdxi=dzdxi+dsfxi(is)*x(2,is)
      dzdeta=dzdeta+dsfeta(is)*x(2,is)
40    continue
c
c---- compute 'J'
      jj(1,1)=drdxi
      jj(1,2)=dzdxi
      jj(2,1)=drdeta
```

```
      jj(2,2)=dzdeta
      detj=jj(1,1)*jj(2,2)-jj(1,2)*jj(2,1)
c
      r=0.0
      do 80 is=1,8
        r=r+shapef(is)*x(1,is)
80    continue
c
      return
      end

c=====================================================================
      subroutine findb(shapef,dsfxi,dsfeta,r,jj,detj,b)
c
c---- compute "B" matrix
c
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .               ,neq,length,nmat,nedge
      dimension shapef(8),dsfxi(8),dsfeta(8),x(2,8),b(4,16),g(5,5)
     .         ,h(5,16),f(4,5),work(4,5)
      double precision jj(2,2)
c
c---- evaluate "G" (5x5)
c-    initialization
      do 50 is=1,5
        do 50 js=1,5
          g(is,js)=0.0
50    continue
c
      g(1,1)=jj(2,2)/detj
      g(3,3)=jj(2,2)/detj
      g(1,2)=-jj(1,2)/detj
      g(3,4)=-jj(1,2)/detj
      g(2,1)=-jj(2,1)/detj
      g(4,3)=-jj(2,1)/detj
      g(2,2)=jj(1,1)/detj
      g(4,4)=jj(1,1)/detj
      if(ietype.eq.1) g(5,5)=1.0
c
c---- evaluate "H" (5x16)
c-    initialization
      do 60 is=1,5
        do 60 js=1,16
          h(is,js)=0.0
60    continue
c
      do 70 is=1,8
        h(1,2*is-1)=dsfxi(is)
        h(2,2*is-1)=dsfeta(is)
        h(3,2*is)=dsfxi(is)
        h(4,2*is)=dsfeta(is)
        if(ietype.eq.1) h(5,2*is-1)=shapef(is)
70    continue
c
c---- evaluate "F"
      do 90 is=1,4
        do 90 js=1,5
          f(is,js)=0.0
90    continue
```

```
c
      f(1,1)=1.0
      f(4,2)=1.0
      f(4,3)=1.0
      f(3,4)=1.0
      if(ietype.eq.1) f(2,5)=1.0/r
c
c---- calculate "B"
c-    work(4x5) =F(4x5) x G(5x5)
      do 100 is=1,4
        do 100 js=1,5
          work(is,js)=0.0
          do 100 ks=1,5
            work(is,js)=work(is,js)+f(is,ks)*g(ks,js)
100   continue
c
c---- B(4x16) =work(4x5) x H(5x16)
      do 200 is=1,4
        do 200 js=1,16
          b(is,js)=0.0
          do 200 ks=1,5
            b(is,js)=b(is,js)+work(is,ks)*h(ks,js)
200   continue
c
      return
      end

c=====================================================================
      subroutine findk(b,e,r,detj,stfk,bte,thick)
c
c     compute element stiffness matrix evaluated
c     at integration points
c
      implicit double precision (a-h,o-z),integer(i-n)
      common /kontrl/ ietype,numnp,ndof,numel,nnpe,nsd
     .                ,neq,length,nmat,nedge
      common /rmatrl / rmat(10,10)
      dimension b(4,16),bt(16,4),e(4,4),stfk(16,16),bte(16,4)
c
c---- find B transpose-> BT
      do 10 i=1,4
        do 10 j=1,16
          bt(j,i)=b(i,j)
10    continue
c
c---- BT*E
      do 20 i=1,16
        do 20 j=1,4
          bte(i,j)=0.
          do 20 k=1,4
            bte(i,j)=bte(i,j)+bt(i,k)*e(k,j)
20    continue
c
c---- BT*E*B*r*|j|
      if (ietype.eq.1) then
c----     axisymmetric
            do 30 i=1,16
              do 30 j=1,16
                stfk(i,j)=0.
                do 30 k=1,4
```

```
                  stfk(i,j)=stfk(i,j)+bte(i,k)*b(k,j)*r*detj
30          continue
      else if (ietype.eq.2) then
c----       plane strain
            do 31 i=1,16
              do 31 j=1,16
                stfk(i,j)=0.
                do 31 k=1,4
                  stfk(i,j)=stfk(i,j)+bte(i,k)*b(k,j)*detj
31          continue
      else
c----       plane stress
            do 32 i=1,16
              do 32 j=1,16
                stfk(i,j)=0.
                do 32 k=1,4
                  stfk(i,j)=stfk(i,j)+bte(i,k)*b(k,j)*thick*detj
32          continue
      end if
c
      return
      end
```

```
c-------------------------------------------------------------
c*************************************************************
c     user.f
c-------------------------------------------------------------
c=============================================================
c---- user-defined subroutines to combine CALREL and FEMCOD
c
      subroutine ugfun(g,x,tp,ig)
      implicit real*8 (a-h,o-z)
      dimension x(1),tp(1)

c---- x vector is the random variable vector
c     x1: Volume ratio
c     x2: Orientation angle
c
c---- call femcod to finish the mechanical transformation
c     var: returned value which will be used
c          in limit-state function
c
      call femcod (var,x)
c
c---- limit-state function
      g = abs(tp(1))-abs(var)
c
      return
      end
c
c
      subroutine udgx(dgx,x,tp,ig)
      implicit real*8 (a-h,o-z)
      dimension x(1),dgx(1),tp(1)
      return
      end
c
      subroutine udd(x,par,sg,ids,cdf,pdf,bnd,ib)
      implicit real*8 (a-h,o-z)
      dimension x(1),par(4),bnd(2)
      return
      end
c
c
      subroutine usize
      common /blkrel/ mtot,np,ia(5000)
      mtot=5000
      return
      end
```

B. FEMCOD INPUT FILE

-- mesh.in

```
Numerical example 1
  ------------------------------------------------------------
    3  149    2   40    8    2   10    4
  ------------------------------------------------------------
    1    000000 0.00000E+00 0.00000E+00
                0.00000E+00 0.00000E+00 0.00000E+00
    2    000000 0.10000E+01 0.00000E+00
                0.00000E+00 0.00000E+00 0.00000E+00
    3    000000 0.10000E+01 0.10000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
    4    000000 0.00000E+00 0.10000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
    5    000000 0.50000E+00 0.00000E+00
                0.00000E+00 0.00000E+00 0.00000E+00
    6    000000 0.10000E+01 0.50000E+00
                0.00000E+00 0.00000E+00 0.00000E+00
    7    000000 0.50000E+00 0.10000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
    8    000000 0.00000E+00 0.50000E+00
                0.00000E+00 0.00000E+00 0.00000E+00
    9    000000 0.10000E+01 0.20000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   10    000000 0.00000E+00 0.20000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   11    000000 0.10000E+01 0.15000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   12    000000 0.50000E+00 0.20000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   13    000000 0.00000E+00 0.15000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   14    000000 0.10000E+01 0.30000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   15    000000 0.00000E+00 0.30000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   16    000000 0.10000E+01 0.25000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   17    000000 0.50000E+00 0.30000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   18    000000 0.00000E+00 0.25000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   19    000000 0.10000E+01 0.40000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   20    000000 0.00000E+00 0.40000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   21    000000 0.10000E+01 0.35000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   22    000000 0.50000E+00 0.40000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   23    000000 0.00000E+00 0.35000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
   24    000000 0.20000E+01 0.00000E+00
                0.00000E+00 0.00000E+00 0.00000E+00
   25    000000 0.20000E+01 0.10000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
```

```
26    000000 0.15000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
27    000000 0.20000E+01 0.50000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
28    000000 0.15000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
29    000000 0.20000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
30    000000 0.20000E+01 0.15000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
31    000000 0.15000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
32    000000 0.20000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
33    000000 0.20000E+01 0.25000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
34    000000 0.15000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
35    000000 0.20000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
36    000000 0.20000E+01 0.35000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
37    000000 0.15000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
38    000000 0.30000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
39    000000 0.30000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
40    000000 0.25000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
41    000000 0.30000E+01 0.50000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
42    000000 0.25000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
43    000000 0.30000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
44    000000 0.30000E+01 0.15000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
45    000000 0.25000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
46    000000 0.30000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
47    000000 0.30000E+01 0.25000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
48    000000 0.25000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
49    000000 0.30000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
50    000000 0.30000E+01 0.35000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
51    000000 0.25000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
52    000000 0.40000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
53    000000 0.40000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
54    000000 0.35000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
55    000000 0.40000E+01 0.50000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
```

```
56    000000 0.35000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
57    000000 0.40000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
58    000000 0.40000E+01 0.15000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
59    000000 0.35000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
60    000000 0.40000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
61    000000 0.40000E+01 0.25000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
62    000000 0.35000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
63    000000 0.40000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
64    000000 0.40000E+01 0.35000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
65    000000 0.35000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
66    000000 0.50000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
67    000000 0.50000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
68    000000 0.45000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
69    000000 0.50000E+01 0.50000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
70    000000 0.45000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
71    000000 0.50000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
72    000000 0.50000E+01 0.15000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
73    000000 0.45000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
74    000000 0.50000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
75    000000 0.50000E+01 0.25000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
76    000000 0.45000E+01 0.30000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
77    000000 0.50000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
78    000000 0.50000E+01 0.35000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
79    000000 0.45000E+01 0.40000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
80    000000 0.60000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
81    000000 0.60000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
82    000000 0.55000E+01 0.00000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
83    000000 0.60000E+01 0.50000E+00
             0.00000E+00 0.00000E+00 0.00000E+00
84    000000 0.55000E+01 0.10000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
85    000000 0.60000E+01 0.20000E+01
             0.00000E+00 0.00000E+00 0.00000E+00
```

```
 86    000000 0.60000E+01 0.15000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 87    000000 0.55000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 88    000000 0.60000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 89    000000 0.60000E+01 0.25000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 90    000000 0.55000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 91    000000 0.60000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 92    000000 0.60000E+01 0.35000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 93    000000 0.55000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 94    000000 0.70000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
 95    000000 0.70000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 96    000000 0.65000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
 97    000000 0.70000E+01 0.50000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
 98    000000 0.65000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
 99    000000 0.70000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
100    000000 0.70000E+01 0.15000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
101    000000 0.65000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
102    000000 0.70000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
103    000000 0.70000E+01 0.25000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
104    000000 0.65000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
105    000000 0.70000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
106    000000 0.70000E+01 0.35000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
107    000000 0.65000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
108    000000 0.80000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
109    000000 0.80000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
110    000000 0.75000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
111    000000 0.80000E+01 0.50000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
112    000000 0.75000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
113    000000 0.80000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
114    000000 0.80000E+01 0.15000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
115    000000 0.75000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
```

```
116    000000 0.80000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
117    000000 0.80000E+01 0.25000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
118    000000 0.75000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
119    000000 0.80000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
120    000000 0.80000E+01 0.35000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
121    000000 0.75000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
122    000000 0.90000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
123    000000 0.90000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
124    000000 0.85000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
125    000000 0.90000E+01 0.50000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
126    000000 0.85000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
127    000000 0.90000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
128    000000 0.90000E+01 0.15000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
129    000000 0.85000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
130    000000 0.90000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
131    000000 0.90000E+01 0.25000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
132    000000 0.85000E+01 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
133    000000 0.90000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
134    000000 0.90000E+01 0.35000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
135    000000 0.85000E+01 0.40000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
136    110000 0.10000E+02 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
137    100000 0.10000E+02 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
138    000000 0.95000E+01 0.00000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
139    100000 0.10000E+02 0.50000E+00
              0.00000E+00 0.00000E+00 0.00000E+00
140    000000 0.95000E+01 0.10000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
141    100000 0.10000E+02 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
142    100000 0.10000E+02 0.15000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
143    000000 0.95000E+01 0.20000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
144    100000 0.10000E+02 0.30000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
145    100000 0.10000E+02 0.25000E+01
              0.00000E+00 0.00000E+00 0.00000E+00
```

```
  146    000000 0.95000E+01 0.30000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
  147    100000 0.10000E+02 0.40000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
  148    100000 0.10000E+02 0.35000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
  149    000000 0.95000E+01 0.40000E+01
                0.00000E+00 0.00000E+00 0.00000E+00
  ------------------------------------------------------------
    1    1    1    2    3    4    5    6    7    8
    2    1    4    3    9   10    7   11   12   13
    3    2   10    9   14   15   12   16   17   18
    4    2   15   14   19   20   17   21   22   23
    5    1    2   24   25    3   26   27   28    6
    6    1    3   25   29    9   28   30   31   11
    7    2    9   29   32   14   31   33   34   16
    8    2   14   32   35   19   34   36   37   21
    9    3   24   38   39   25   40   41   42   27
   10    3   25   39   43   29   42   44   45   30
   11    4   29   43   46   32   45   47   48   33
   12    4   32   46   49   35   48   50   51   36
   13    3   38   52   53   39   54   55   56   41
   14    3   39   53   57   43   56   58   59   44
   15    4   43   57   60   46   59   61   62   47
   16    4   46   60   63   49   62   64   65   50
   17    5   52   66   67   53   68   69   70   55
   18    5   53   67   71   57   70   72   73   58
   19    6   57   71   74   60   73   75   76   61
   20    6   60   74   77   63   76   78   79   64
   21    5   66   80   81   67   82   83   84   69
   22    5   67   81   85   71   84   86   87   72
   23    6   71   85   88   74   87   89   90   75
   24    6   74   88   91   77   90   92   93   78
   25    7   80   94   95   81   96   97   98   83
   26    7   81   95   99   85   98  100  101   86
   27    8   85   99  102   88  101  103  104   89
   28    8   88  102  105   91  104  106  107   92
   29    7   94  108  109   95  110  111  112   97
   30    7   95  109  113   99  112  114  115  100
   31    8   99  113  116  102  115  117  118  103
   32    8  102  116  119  105  118  120  121  106
   33    9  108  122  123  109  124  125  126  111
   34    9  109  123  127  113  126  128  129  114
   35   10  113  127  130  116  129  131  132  117
   36   10  116  130  133  119  132  134  135  120
   37    9  122  136  137  123  138  139  140  125
   38    9  123  137  141  127  140  142  143  128
   39   10  127  141  144  130  143  145  146  131
   40   10  130  144  147  133  146  148  149  134
  ------------------------------------------------------------
    1 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.11000E+05 0.00000E+00
    2 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.11000E+05 0.00000E+00
    3 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.11000E+05 0.00000E+00
    4 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.11000E+05 0.00000E+00
    5 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
```

```
     6 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
     7 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
     8 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
     9 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    10 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    11 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    12 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    13 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    14 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    15 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    16 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    17 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    18 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    19 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    20 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    21 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    22 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    23 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    24 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    25 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    26 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    27 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    28 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    29 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    30 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    31 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    32 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    33 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    34 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
    35 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
```

```
   36 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
   37 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
   38 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
   39 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
   40 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
0.00000E+00 0.00000E+00 0.00000E+00
  -------------------------------------------------------------
```

C. CALREL INPUT FILE

-- input.txt

```
CALRel nrx=20 ntp=1
DATA
TITL nline title
1
Example 1 -- Pure tension test.
FLAG icl,igr
1 0
OPTI iop,ni1,ni2,tol,op1,op2,op3
1,50,4,0.05
STAT igt(i),nge,ngm nv,ids,ex,sg,p3,p4,x0
1 20
x1   1,1,0.5,0.1,0.0,0.0,0.5
x2   2,1,0.0,0.0873,0.0,0.0,0.0
x3   3,1,0.5,0.1,0.0,0.0,0.5
x4   4,1,0.0,0.0873,0.0,0.0,0.0
x5   5,1,0.5,0.1,0.0,0.0,0.5
x6   6,1,0.0,0.0873,0.0,0.0,0.0
x7   7,1,0.5,0.1,0.0,0.0,0.5
x8   8,1,0.0,0.0873,0.0,0.0,0.0
x9   9,1,0.5,0.1,0.0,0.0,0.5
x10 10,1,0.0,0.0873,0.0,0.0,0.0
x11 11,1,0.5,0.1,0.0,0.0,0.5
x12 12,1,0.0,0.0873,0.0,0.0,0.0
x13 13,1,0.5,0.1,0.0,0.0,0.5
x14 14,1,0.0,0.0873,0.0,0.0,0.0
x15 15,1,0.5,0.1,0.0,0.0,0.5
x16 16,1,0.0,0.0873,0.0,0.0,0.0
x17 17,1,0.5,0.1,0.0,0.0,0.5
x18 18,1,0.0,0.0873,0.0,0.0,0.0
x19 19,1,0.5,0.1,0.0,0.0,0.5
x20 20,1,0.0,0.0873,0.0,0.0,0.0
PARA tp
12000
END
FORM ini=1
SENS isc=1 isv=1
EXIT
```

D. LOOK-UP TABLE FOR DIFFERENT VOLUME RATIOS

-- table.txt

```
0.4750E+06  0.1630E+06  0.3990E+00  0.5300E+00  0.7400E+05
0.5100E+06  0.1960E+06  0.3970E+00  0.5310E+00  0.7400E+05
0.6560E+06  0.1910E+06  0.3970E+00  0.5130E+00  0.7420E+05
0.7470E+06  0.1890E+06  0.3990E+00  0.5090E+00  0.7460E+05
0.8100E+06  0.1850E+06  0.4020E+00  0.4940E+00  0.7620E+05
```

E. MAKE-FILE FOR CALREL-FEMCOD IN MS FORTRAN 5.1

-- fll.bat

```
fl /F 10000 /c /AH /G2 /Ot /Foadstif.obj adstif.for
fl /F 10000 /c /AH /G2 /Ot /Foaxis.obj axis.for
fl /F 10000 /c /AH /G2 /Ot /Foaxistr.obj axistr.for
fl /F 10000 /c /AH /G2 /Ot /Fofemcod.obj femcod.for
fl /F 10000 /c /AH /G2 /Ot /Fofindb.obj findb.for
fl /F 10000 /c /AH /G2 /Ot /Fouser1.obj user1.for
link /SEG:200 boun core data dirs form lib main mont pnet senc sens sorm
user femcod axis axistr findb adstif
```

F. CALREL-FEMCOD OUTPUT FILE

-- output.txt

```
    ************************************************************
    *      U n i v e r s i t y   o f   C a l i f o r n i a     *
    *            Department  of  Civil  Engineering            *
    *                                                          *
    *                     C  A  L  R  E  L                     *
    *                 CAL-RELiability program                  *
    *                       Developed by                       *
    *         P.-L. Liu, H.-Z. Lin and A. Der Kiureghian       *
    *                                                          *
    *                Last Revision:   December 1990            *
    *                      Copyright @ 1989                    *
    ************************************************************


>>>> NEW PROBLEM <<<<

number of limit-state functions.....ngf=      1
number of independent variable groups ...nig=      1
total number of random variables ........nrx=     20
number of limit-state parameters ........ntp=      1


>>>> INPUT DATA <<<<

Example 1 -- Pure tension test.
type of system ............................icl=      1
  icl=1 .....................................component
  icl=2 ................................series system
  icl=3 ...............................general system
flag for gradient computation ............igr=      0
  igr=0 ............................finite difference
  igr=1 ...................formulas provided by user

optimization scheme used ................iop=      1
  iop=1 ................................HL-RF method
  iop=2 ........................modified HL-RF method
  iop=3 ...................gradient projection method
  iop=4 .................sequential quadratic method
maximum number of iteration cycles ......ni1=     50
maximum steps in line search ............ni2=      4
convergence tolerance .................tol= 5.000E-02
optimization parameter 1 ..............op1= 1.000E+00
optimization parameter 2 ..............op2= 0.000E+00
optimization parameter 3 ..............op3= 0.000E+00

statistical data of basic varibles:
available probability distributions:
  determinitic ..............ids=0
  normal ....................ids=1
  lognormal .................ids=2
  gamma .....................ids=3
  shifted exponential .......ids=4
  shifted rayleigh ..........ids=5
  uniform ...................ids=6
  beta ......................ids=7
  type i largest value .....ids=11
  type i smallest value ....ids=12
  type ii largest value ....ids=13
  weibull ...................ids=14
  user defined ..............ids>50
```

```
group no.:    1          group type:    1
var   ids   mean    st. dev.   param1    param2    param3    param4   init. pt
x1      1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x2      1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x3      1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x4      1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x5      1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x6      1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x7      1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x8      1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x9      1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x10     1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x11     1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x12     1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x13     1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x14     1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x15     1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x16     1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x17     1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x18     1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00
x19     1  5.00E-01  1.00E-01  5.00E-01  1.00E-01                     5.00E-01
x20     1  0.00E+00  8.73E-02  0.00E+00  8.73E-02                     0.00E+00

deterministic parameters in limit-state function:
  tp (  1) =  1.200E+04


>>>> FIRST-ORDER RELIABILITY ANALYSIS <<<<

print interval ..........................npr=      0
  npr<0 ..........no first order results are printed
  npr=0 ........print the final step of FORM results
  npr>0 ........print the results of every npr steps
initialization flag .....................ini=      1
  ini=0 .......................start from mean point
  ini=1 ..........start from point specified by user
  ini=-1 ....start from previous linearization point
restart flag ............................ist=      0
  ist=0 .......................analyze a new problem
  ist=1 ..............continue an unconverged problem

limit-state function     1
--------------------------------------------------------------------------------
iteration number ..............iter=         5
value of limit-state function..g(x)= 2.221E-05
reliability index .............beta=     .6527
probability .....................Pf1= 2.570E-01
var          design point                    sensitivity vectors
           x*           u*           alpha     gamma     delta      eta
x1      5.000E-01     4.951E-04         .0008     .0008    -.0008     .0000
x2      4.851E-04     5.557E-03         .0085     .0085    -.0085     .0000
x3      5.000E-01     4.951E-04         .0008     .0008    -.0008     .0000
x4     -4.851E-04    -5.557E-03        -.0085    -.0085     .0085     .0000
x5      5.001E-01     9.210E-04         .0014     .0014    -.0014     .0000
x6      1.535E-03     1.759E-02         .0269     .0269    -.0269    -.0005
x7      5.001E-01     9.210E-04         .0014     .0014    -.0014     .0000
x8     -1.535E-03    -1.759E-02        -.0269    -.0269     .0269    -.0005
x9      5.003E-01     2.661E-03         .0041     .0041    -.0041     .0000
x10     4.693E-03     5.375E-02         .0824     .0824    -.0824    -.0044
x11     5.003E-01     2.661E-03         .0041     .0041    -.0041     .0000
x12    -4.693E-03    -5.375E-02        -.0824    -.0824     .0824    -.0044
x13     5.005E-01     5.322E-03         .0081     .0081    -.0081     .0000
x14     1.339E-02     1.534E-01         .2350     .2350    -.2350    -.0360
x15     5.005E-01     5.322E-03         .0081     .0081    -.0081     .0000
x16    -1.339E-02    -1.534E-01        -.2350    -.2350     .2350    -.0360
x17     5.014E-01     1.368E-02         .0209     .0209    -.0209    -.0003
x18     3.765E-02     4.313E-01         .6608     .6608    -.6608    -.2850
```

```
x19     5.014E-01     1.368E-02         .0209     .0209    -.0209    -.0003
x20    -3.765E-02    -4.313E-01        -.6608    -.6608     .6608    -.2850
-------------------------------------------------------------------------------


>>>> SENSITIVITY ANALYSIS AT COMPONENT LEVEL <<<<

type of parameters for sensitivity analysis
.........................................isv=      1
  isv=1 .....................distribution parameters
  isv=2 ...................limit-state fcn parameters
  isv=0 ..distribution and limit-state fcn parameters

sensitivity with respect to distribution parameters

limit-state function    1
-------------------------------------------------------------------------
d(beta)/d(parameter) :
var     mean      std dev     par 1      par 2      par 3      par 4
x1   -7.586E-03 -3.756E-06 -7.586E-03 -3.756E-06
x2   -9.753E-02 -5.419E-04 -9.753E-02 -5.419E-04
x3   -7.586E-03 -3.756E-06 -7.586E-03 -3.756E-06
x4    9.753E-02 -5.419E-04  9.753E-02 -5.419E-04
x5   -1.411E-02 -1.300E-05 -1.411E-02 -1.300E-05
x6   -3.087E-01 -5.428E-03 -3.087E-01 -5.428E-03
x7   -1.411E-02 -1.300E-05 -1.411E-02 -1.300E-05
x8    3.087E-01 -5.428E-03  3.087E-01 -5.428E-03
x9   -4.078E-02 -1.085E-04 -4.078E-02 -1.085E-04
x10  -9.434E-01 -5.071E-02 -9.434E-01 -5.071E-02
x11  -4.078E-02 -1.085E-04 -4.078E-02 -1.085E-04
x12   9.434E-01 -5.071E-02  9.434E-01 -5.071E-02
x13  -8.144E-02 -4.334E-04 -8.144E-02 -4.334E-04
x14  -2.692E+00 -4.128E-01 -2.692E+00 -4.128E-01
x15  -8.144E-02 -4.334E-04 -8.144E-02 -4.334E-04
x16   2.692E+00 -4.128E-01  2.692E+00 -4.128E-01
x17  -2.095E-01 -2.866E-03 -2.095E-01 -2.866E-03
x18  -7.570E+00 -3.265E+00 -7.570E+00 -3.265E+00
x19  -2.095E-01 -2.866E-03 -2.095E-01 -2.866E-03
x20   7.570E+00 -3.265E+00  7.570E+00 -3.265E+00

d(Pf1)/d(parameter) :
var     mean      std dev     par 1      par 2      par 3      par 4
x1    2.446E-03  1.211E-06  2.446E-03  1.211E-06
x2    3.144E-02  1.747E-04  3.144E-02  1.747E-04
x3    2.446E-03  1.211E-06  2.446E-03  1.211E-06
x4   -3.144E-02  1.747E-04 -3.144E-02  1.747E-04
x5    4.550E-03  4.191E-06  4.550E-03  4.191E-06
x6    9.951E-02  1.750E-03  9.951E-02  1.750E-03
x7    4.550E-03  4.191E-06  4.550E-03  4.191E-06
x8   -9.951E-02  1.750E-03 -9.951E-02  1.750E-03
x9    1.315E-02  3.499E-05  1.315E-02  3.499E-05
x10   3.042E-01  1.635E-02  3.042E-01  1.635E-02
x11   1.315E-02  3.499E-05  1.315E-02  3.499E-05
x12  -3.042E-01  1.635E-02 -3.042E-01  1.635E-02
x13   2.626E-02  1.397E-04  2.626E-02  1.397E-04
x14   8.678E-01  1.331E-01  8.678E-01  1.331E-01
x15   2.626E-02  1.397E-04  2.626E-02  1.397E-04
x16  -8.678E-01  1.331E-01 -8.678E-01  1.331E-01
x17   6.754E-02  9.242E-04  6.754E-02  9.242E-04
x18   2.440E+00  1.053E+00  2.440E+00  1.053E+00
x19   6.754E-02  9.242E-04  6.754E-02  9.242E-04
x20  -2.440E+00  1.053E+00 -2.440E+00  1.053E+00
-------------------------------------------------------------------------
Stop - Program terminated.
```

G. ABAQUS MODEL INPUT FILES FOR 50% VOLUME RATIO

-- long.inp and trans.inp

```
*HEADING
ABAQUS job created on 16-Feb-:0 at 03:01:47
**
*RESTART, WRITE, FREQUENCY=1
**
*NODE
       1,            0.,           1.4
       2,            0.,         1.615
       3,            0.,          1.83
       4,            0.,         2.045
       5,            0.,          2.26
       6,            0.,         2.475
       7,            0.,          2.69
       8,            0.,         2.905
       9,            0.,          3.12
      10,            0.,         3.335
      11,            0.,          3.55
      12,            0.,         3.765
      13,            0.,          3.98
      14,            0.,         4.195
      15,            0.,          4.41
      16,            0.,         4.625
      17,            0.,          4.84
      18,            0.,         5.055
      19,            0.,          5.27
      20,            0.,         5.485
      21,            0.,            5.7
      22,            0.,         5.915
      23,            0.,          6.13
      24,            0.,         6.345
      25,            0.,          6.56
      26,            0.,         6.775
      27,            0.,          6.99
      28,            0.,         7.205
      29,            0.,          7.42
      30,            0.,         7.635
      31,            0.,          7.85
      32,            0.,         8.065
      33,            0.,          8.28
      34,            0.,         8.495
      35,            0.,          8.71
      36,            0.,         8.925
      37,            0.,          9.14
      38,            0.,         9.355
      39,            0.,          9.57
      40,            0.,         9.785
      41,            0.,            10.
      42,      0.143333,           1.4
      43,      0.143333,          1.83
      44,      0.143333,          2.26
      45,      0.143333,          2.69
      46,      0.143333,          3.12
      47,      0.143333,          3.55
      48,      0.143333,          3.98
```

```
 49,      0.143333,          4.41
 50,      0.143333,          4.84
 51,      0.143333,          5.27
 52,      0.143333,           5.7
 53,      0.143333,          6.13
 54,      0.143333,          6.56
 55,      0.143333,          6.99
 56,      0.143333,          7.42
 57,      0.143333,          7.85
 58,      0.143333,          8.28
 59,      0.143333,          8.71
 60,      0.143333,          9.14
 61,      0.143333,          9.57
 62,      0.143333,           10.
 63,      0.286667,           1.4
 64,      0.286667,         1.615
 65,      0.286667,          1.83
 66,      0.286667,         2.045
 67,      0.286667,          2.26
 68,      0.286667,         2.475
 69,      0.286667,          2.69
 70,      0.286667,         2.905
 71,      0.286667,          3.12
 72,      0.286667,         3.335
 73,      0.286667,          3.55
 74,      0.286667,         3.765
 75,      0.286667,          3.98
 76,      0.286667,         4.195
 77,      0.286667,          4.41
 78,      0.286667,         4.625
 79,      0.286667,          4.84
 80,      0.286667,         5.055
 81,      0.286667,          5.27
 82,      0.286667,         5.485
 83,      0.286667,           5.7
 84,      0.286667,         5.915
 85,      0.286667,          6.13
 86,      0.286667,         6.345
 87,      0.286667,          6.56
 88,      0.286667,         6.775
 89,      0.286667,          6.99
 90,      0.286667,         7.205
 91,      0.286667,          7.42
 92,      0.286667,         7.635
 93,      0.286667,          7.85
 94,      0.286667,         8.065
 95,      0.286667,          8.28
 96,      0.286667,         8.495
 97,      0.286667,          8.71
 98,      0.286667,         8.925
 99,      0.286667,          9.14
100,      0.286667,         9.355
101,      0.286667,          9.57
102,      0.286667,         9.785
103,      0.286667,           10.
104,          0.43,           1.4
105,          0.43,          1.83
106,          0.43,          2.26
107,          0.43,          2.69
108,          0.43,          3.12
```

109,	0.43,	3.55
110,	0.43,	3.98
111,	0.43,	4.41
112,	0.43,	4.84
113,	0.43,	5.27
114,	0.43,	5.7
115,	0.43,	6.13
116,	0.43,	6.56
117,	0.43,	6.99
118,	0.43,	7.42
119,	0.43,	7.85
120,	0.43,	8.28
121,	0.43,	8.71
122,	0.43,	9.14
123,	0.43,	9.57
124,	0.43,	10.
125,	0.573333,	1.4
126,	0.573333,	1.615
127,	0.573333,	1.83
128,	0.573333,	2.045
129,	0.573333,	2.26
130,	0.573333,	2.475
131,	0.573333,	2.69
132,	0.573333,	2.905
133,	0.573333,	3.12
134,	0.573333,	3.335
135,	0.573333,	3.55
136,	0.573333,	3.765
137,	0.573333,	3.98
138,	0.573333,	4.195
139,	0.573333,	4.41
140,	0.573333,	4.625
141,	0.573333,	4.84
142,	0.573333,	5.055
143,	0.573333,	5.27
144,	0.573333,	5.485
145,	0.573333,	5.7
146,	0.573333,	5.915
147,	0.573333,	6.13
148,	0.573333,	6.345
149,	0.573333,	6.56
150,	0.573333,	6.775
151,	0.573333,	6.99
152,	0.573333,	7.205
153,	0.573333,	7.42
154,	0.573333,	7.635
155,	0.573333,	7.85
156,	0.573333,	8.065
157,	0.573333,	8.28
158,	0.573333,	8.495
159,	0.573333,	8.71
160,	0.573333,	8.925
161,	0.573333,	9.14
162,	0.573333,	9.355
163,	0.573333,	9.57
164,	0.573333,	9.785
165,	0.573333,	10.
166,	0.716667,	1.4
167,	0.716667,	1.83
168,	0.716667,	2.26

```
169,      0.716667,          2.69
170,      0.716667,          3.12
171,      0.716667,          3.55
172,      0.716667,          3.98
173,      0.716667,          4.41
174,      0.716667,          4.84
175,      0.716667,          5.27
176,      0.716667,           5.7
177,      0.716667,          6.13
178,      0.716667,          6.56
179,      0.716667,          6.99
180,      0.716667,          7.42
181,      0.716667,          7.85
182,      0.716667,          8.28
183,      0.716667,          8.71
184,      0.716667,          9.14
185,      0.716667,          9.57
186,      0.716667,           10.
187,          0.86,           1.4
188,          0.86,         1.615
189,          0.86,          1.83
190,          0.86,         2.045
191,          0.86,          2.26
192,          0.86,         2.475
193,          0.86,          2.69
194,          0.86,         2.905
195,          0.86,          3.12
196,          0.86,         3.335
197,          0.86,          3.55
198,          0.86,         3.765
199,          0.86,          3.98
200,          0.86,         4.195
201,          0.86,          4.41
202,          0.86,         4.625
203,          0.86,          4.84
204,          0.86,         5.055
205,          0.86,          5.27
206,          0.86,         5.485
207,          0.86,           5.7
208,          0.86,         5.915
209,          0.86,          6.13
210,          0.86,         6.345
211,          0.86,          6.56
212,          0.86,         6.775
213,          0.86,          6.99
214,          0.86,         7.205
215,          0.86,          7.42
216,          0.86,         7.635
217,          0.86,          7.85
218,          0.86,         8.065
219,          0.86,          8.28
220,          0.86,         8.495
221,          0.86,          8.71
222,          0.86,         8.925
223,          0.86,          9.14
224,          0.86,         9.355
225,          0.86,          9.57
226,          0.86,         9.785
227,          0.86,           10.
228,          0.86,           1.4
```

```
229,          0.93,           1.4
230,          0.93,          1.83
231,          0.93,          2.26
232,          0.93,          2.69
233,          0.93,          3.12
234,          0.93,          3.55
235,          0.93,          3.98
236,          0.93,          4.41
237,          0.93,          4.84
238,          0.93,          5.27
239,          0.93,           5.7
240,          0.93,          6.13
241,          0.93,          6.56
242,          0.93,          6.99
243,          0.93,          7.42
244,          0.93,          7.85
245,          0.93,          8.28
246,          0.93,          8.71
247,          0.93,          9.14
248,          0.93,          9.57
249,          0.93,           10.
250,            1.,           1.4
251,            1.,         1.615
252,            1.,          1.83
253,            1.,         2.045
254,            1.,          2.26
255,            1.,         2.475
256,            1.,          2.69
257,            1.,         2.905
258,            1.,          3.12
259,            1.,         3.335
260,            1.,          3.55
261,            1.,         3.765
262,            1.,          3.98
263,            1.,         4.195
264,            1.,          4.41
265,            1.,         4.625
266,            1.,          4.84
267,            1.,         5.055
268,            1.,          5.27
269,            1.,         5.485
270,            1.,           5.7
271,            1.,         5.915
272,            1.,          6.13
273,            1.,         6.345
274,            1.,          6.56
275,            1.,         6.775
276,            1.,          6.99
277,            1.,         7.205
278,            1.,          7.42
279,            1.,         7.635
280,            1.,          7.85
281,            1.,         8.065
282,            1.,          8.28
283,            1.,         8.495
284,            1.,          8.71
285,            1.,         8.925
286,            1.,          9.14
287,            1.,         9.355
288,            1.,          9.57
```

```
289,           1.,         9.785
290,           1.,           10.
291,         0.86,            0.
292,         0.86,         0.175
293,         0.86,          0.35
294,         0.86,         0.525
295,         0.86,           0.7
296,         0.86,         0.875
297,         0.86,          1.05
298,         0.86,         1.225
299,         0.86,           1.4
300,         0.93,            0.
301,         0.93,          0.35
302,         0.93,           0.7
303,         0.93,          1.05
304,           1.,            0.
305,           1.,         0.175
306,           1.,          0.35
307,           1.,         0.525
308,           1.,           0.7
309,           1.,         0.875
310,           1.,          1.05
311,           1.,         1.225
312,           0.,            0.
313,           0.,         0.175
314,           0.,          0.35
315,           0.,         0.525
316,           0.,           0.7
317,           0.,         0.875
318,           0.,          1.05
319,           0.,         1.225
320,     0.143333,            0.
321,     0.143333,          0.35
322,     0.143333,           0.7
323,     0.143333,          1.05
324,     0.286667,            0.
325,     0.286667,         0.175
326,     0.286667,          0.35
327,     0.286667,         0.525
328,     0.286667,           0.7
329,     0.286667,         0.875
330,     0.286667,          1.05
331,     0.286667,         1.225
332,          0.43,            0.
333,          0.43,          0.35
334,          0.43,           0.7
335,          0.43,          1.05
336,     0.573333,            0.
337,     0.573333,         0.175
338,     0.573333,          0.35
339,     0.573333,         0.525
340,     0.573333,           0.7
341,     0.573333,         0.875
342,     0.573333,          1.05
343,     0.573333,         1.225
344,     0.716667,            0.
345,     0.716667,          0.35
346,     0.716667,           0.7
347,     0.716667,          1.05
348,          0.86,           1.4
```

```
**
**
*ELEMENT, TYPE=CAX8, ELSET=PARTICLE
       1,       1,      63,      65,       3,      42,      64,
      43,       2
       2,       3,      65,      67,       5,      43,      66,
      44,       4
       3,       5,      67,      69,       7,      44,      68,
      45,       6
       4,       7,      69,      71,       9,      45,      70,
      46,       8
       5,       9,      71,      73,      11,      46,      72,
      47,      10
       6,      11,      73,      75,      13,      47,      74,
      48,      12
       7,      13,      75,      77,      15,      48,      76,
      49,      14
       8,      15,      77,      79,      17,      49,      78,
      50,      16
       9,      17,      79,      81,      19,      50,      80,
      51,      18
      10,      19,      81,      83,      21,      51,      82,
      52,      20
      11,      21,      83,      85,      23,      52,      84,
      53,      22
      12,      23,      85,      87,      25,      53,      86,
      54,      24
      13,      25,      87,      89,      27,      54,      88,
      55,      26
      14,      27,      89,      91,      29,      55,      90,
      56,      28
      15,      29,      91,      93,      31,      56,      92,
      57,      30
      16,      31,      93,      95,      33,      57,      94,
      58,      32
      17,      33,      95,      97,      35,      58,      96,
      59,      34
      18,      35,      97,      99,      37,      59,      98,
      60,      36
      19,      37,      99,     101,      39,      60,     100,
      61,      38
      20,      39,     101,     103,      41,      61,     102,
      62,      40
      21,      63,     125,     127,      65,     104,     126,
     105,      64
      22,      65,     127,     129,      67,     105,     128,
     106,      66
      23,      67,     129,     131,      69,     106,     130,
     107,      68
      24,      69,     131,     133,      71,     107,     132,
     108,      70
      25,      71,     133,     135,      73,     108,     134,
     109,      72
      26,      73,     135,     137,      75,     109,     136,
     110,      74
      27,      75,     137,     139,      77,     110,     138,
     111,      76
      28,      77,     139,     141,      79,     111,     140,
     112,      78
      29,      79,     141,     143,      81,     112,     142,
```

```
113,      80
 30,      81,     143,     145,      83,     113,     144,
114,      82
 31,      83,     145,     147,      85,     114,     146,
115,      84
 32,      85,     147,     149,      87,     115,     148,
116,      86
 33,      87,     149,     151,      89,     116,     150,
117,      88
 34,      89,     151,     153,      91,     117,     152,
118,      90
 35,      91,     153,     155,      93,     118,     154,
119,      92
 36,      93,     155,     157,      95,     119,     156,
120,      94
 37,      95,     157,     159,      97,     120,     158,
121,      96
 38,      97,     159,     161,      99,     121,     160,
122,      98
 39,      99,     161,     163,     101,     122,     162,
123,     100
 40,     101,     163,     165,     103,     123,     164,
124,     102
 41,     125,     187,     189,     127,     166,     188,
167,     126
 42,     127,     189,     191,     129,     167,     190,
168,     128
 43,     129,     191,     193,     131,     168,     192,
169,     130
 44,     131,     193,     195,     133,     169,     194,
170,     132
 45,     133,     195,     197,     135,     170,     196,
171,     134
 46,     135,     197,     199,     137,     171,     198,
172,     136
 47,     137,     199,     201,     139,     172,     200,
173,     138
 48,     139,     201,     203,     141,     173,     202,
174,     140
 49,     141,     203,     205,     143,     174,     204,
175,     142
 50,     143,     205,     207,     145,     175,     206,
176,     144
 51,     145,     207,     209,     147,     176,     208,
177,     146
 52,     147,     209,     211,     149,     177,     210,
178,     148
 53,     149,     211,     213,     151,     178,     212,
179,     150
 54,     151,     213,     215,     153,     179,     214,
180,     152
 55,     153,     215,     217,     155,     180,     216,
181,     154
 56,     155,     217,     219,     157,     181,     218,
182,     156
 57,     157,     219,     221,     159,     182,     220,
183,     158
 58,     159,     221,     223,     161,     183,     222,
184,     160
 59,     161,     223,     225,     163,     184,     224,
```

```
     185,     162
      60,     163,     225,     227,     165,     185,     226,
     186,     164
*ELEMENT, TYPE=CAX8, ELSET=MATRIX
      61,     228,     250,     252,     189,     229,     251,
     230,     188
      62,     189,     252,     254,     191,     230,     253,
     231,     190
      63,     191,     254,     256,     193,     231,     255,
     232,     192
      64,     193,     256,     258,     195,     232,     257,
     233,     194
      65,     195,     258,     260,     197,     233,     259,
     234,     196
      66,     197,     260,     262,     199,     234,     261,
     235,     198
      67,     199,     262,     264,     201,     235,     263,
     236,     200
      68,     201,     264,     266,     203,     236,     265,
     237,     202
      69,     203,     266,     268,     205,     237,     267,
     238,     204
      70,     205,     268,     270,     207,     238,     269,
     239,     206
      71,     207,     270,     272,     209,     239,     271,
     240,     208
      72,     209,     272,     274,     211,     240,     273,
     241,     210
      73,     211,     274,     276,     213,     241,     275,
     242,     212
      74,     213,     276,     278,     215,     242,     277,
     243,     214
      75,     215,     278,     280,     217,     243,     279,
     244,     216
      76,     217,     280,     282,     219,     244,     281,
     245,     218
      77,     219,     282,     284,     221,     245,     283,
     246,     220
      78,     221,     284,     286,     223,     246,     285,
     247,     222
      79,     223,     286,     288,     225,     247,     287,
     248,     224
      80,     225,     288,     290,     227,     248,     289,
     249,     226
      81,     291,     304,     306,     293,     300,     305,
     301,     292
      82,     293,     306,     308,     295,     301,     307,
     302,     294
      83,     295,     308,     310,     297,     302,     309,
     303,     296
      84,     297,     310,     250,     299,     303,     311,
     229,     298
      85,     312,     324,     326,     314,     320,     325,
     321,     313
      86,     314,     326,     328,     316,     321,     327,
     322,     315
      87,     316,     328,     330,     318,     322,     329,
     323,     317
      88,     318,     330,      63,       1,     323,     331,
      42,     319
```

```
      89,     324,     336,     338,     326,     332,     337,
     333,     325
      90,     326,     338,     340,     328,     333,     339,
     334,     327
      91,     328,     340,     342,     330,     334,     341,
     335,     329
      92,     330,     342,     125,      63,     335,     343,
     104,     331
      93,     336,     291,     293,     338,     344,     292,
     345,     337
      94,     338,     293,     295,     340,     345,     294,
     346,     339
      95,     340,     295,     297,     342,     346,     296,
     347,     341
      96,     342,     297,     348,     125,     347,     298,
     166,     343
**
** matrix
**
*ORIENTATION, SYSTEM=R, NAME=OID1
         0.,          1.,          0.,         -1.,          0.,
0.
       1,          0.
*SOLID SECTION, ELSET=MATRIX, MATERIAL=PLASTIC, ORIENTATION=OID1
         1.,
**
** particle
**
*SOLID SECTION, ELSET=PARTICLE, MATERIAL=WOOD, ORIENTATION=OID1
         1.,
**
** plastic
** Date: 16-Feb-:0           Time: 02:57:54
**
*MATERIAL, NAME=PLASTIC
**
*ELASTIC, TYPE=ISO
     200000.,         0.4
**
** wood
** Date: 16-Feb-:0           Time: 02:57:54
**
*MATERIAL, NAME=WOOD
**
*ELASTIC, TYPE=ENGINEERING CONSTANTS
     1.6+6     1.6+5     1.6+5       0.4       0.4       0.4    80000.
80000.
    16000.
**
** Step  1, step1
** LoadCase, uniform_disp
**
*STEP, AMPLITUDE=RAMP, PERTURBATION
**
*STATIC
**
**
**
** axis_top
**
```

```
*BOUNDARY, OP=NEW
      41, 1,,          0.
      41, 2,,          0.
      62, 2,,          0.
     103, 2,,          0.
     124, 2,,          0.
     165, 2,,          0.
     186, 2,,          0.
     227, 2,,          0.
     249, 2,,          0.
     290, 2,,          0.
**
** uniform_disp
**
*BOUNDARY, OP=NEW
     291, 2,,      -0.001
     300, 2,,      -0.001
     304, 2,,      -0.001
     312, 1,,          0.
     312, 2,,      -0.001
     320, 2,,      -0.001
     324, 2,,      -0.001
     332, 2,,      -0.001
     336, 2,,      -0.001
     344, 2,,      -0.001
**
** axis_left
**
*BOUNDARY, OP=NEW
       1, 1,,          0.
       2, 1,,          0.
       3, 1,,          0.
       4, 1,,          0.
       5, 1,,          0.
       6, 1,,          0.
       7, 1,,          0.
       8, 1,,          0.
       9, 1,,          0.
      10, 1,,          0.
      11, 1,,          0.
      12, 1,,          0.
      13, 1,,          0.
      14, 1,,          0.
      15, 1,,          0.
      16, 1,,          0.
      17, 1,,          0.
      18, 1,,          0.
      19, 1,,          0.
      20, 1,,          0.
      21, 1,,          0.
      22, 1,,          0.
      23, 1,,          0.
      24, 1,,          0.
      25, 1,,          0.
      26, 1,,          0.
      27, 1,,          0.
      28, 1,,          0.
      29, 1,,          0.
      30, 1,,          0.
      31, 1,,          0.
```

```
      32, 1,,          0.
      33, 1,,          0.
      34, 1,,          0.
      35, 1,,          0.
      36, 1,,          0.
      37, 1,,          0.
      38, 1,,          0.
      39, 1,,          0.
      40, 1,,          0.
     313, 1,,          0.
     314, 1,,          0.
     315, 1,,          0.
     316, 1,,          0.
     317, 1,,          0.
     318, 1,,          0.
     319, 1,,          0.
**
*CLOAD, OP=NEW
*DLOAD, OP=NEW
*TEMPERATURE, OP=NEW
**
*NODE PRINT, FREQ=1
U,
RF,
*NODE FILE, FREQ=1
U,
RF,
**
*EL PRINT, POS=INTEG, FREQ=1
S,
E,
*EL FILE,  POS=INTEG, FREQ=1
S,
E,
**
*EL PRINT, POS=NODES, FREQ=0
**
*EL FILE,  POS=NODES, FREQ=0
**
*EL PRINT, POS=CENTR, FREQ=0
**
*EL FILE,  POS=CENTR, FREQ=0
**
*EL PRINT, POS=AVERAGE, FREQ=0
**
*EL FILE,  POS=AVERAGE, FREQ=0
**
*MODAL PRINT, FREQ=99999
**
*MODAL FILE, FREQ=99999
**
*ENERGY PRINT, FREQ=0
**
*ENERGY FILE, FREQ=0
**
*PRINT, FREQ=1
**
*END STEP
```

```
*HEADING
ABAQUS job created on 15-Feb-:0 at 11:45:54
**
*RESTART, WRITE, FREQUENCY=1
**
*NODE
       1,           0.,           0.
       2,          0.04,          0.
       3,          0.08,          0.
       4,          0.12,          0.
       5,          0.16,          0.
       6,           0.2,          0.
       7,          0.24,          0.
       8,          0.28,          0.
       9,          0.32,          0.
      10,          0.36,          0.
      11,           0.4,          0.
      12,           0.,          0.04
      13,          0.08,         0.04
      14,          0.16,         0.04
      15,          0.24,         0.04
      16,          0.32,         0.04
      17,           0.4,         0.04
      18,           0.,          0.08
      19,          0.04,         0.08
      20,          0.08,         0.08
      21,          0.12,         0.08
      22,          0.16,         0.08
      23,           0.2,         0.08
      24,          0.24,         0.08
      25,          0.28,         0.08
      26,          0.32,         0.08
      27,          0.36,         0.08
      28,           0.4,         0.08
      29,           0.,          0.12
      30,          0.08,         0.12
      31,          0.16,         0.12
      32,          0.24,         0.12
      33,          0.32,         0.12
      34,           0.4,         0.12
      35,           0.,          0.16
      36,          0.04,         0.16
      37,          0.08,         0.16
      38,          0.12,         0.16
      39,          0.16,         0.16
      40,           0.2,         0.16
      41,          0.24,         0.16
      42,          0.28,         0.16
      43,          0.32,         0.16
      44,          0.36,         0.16
      45,           0.4,         0.16
      46,           0.,           0.2
      47,          0.08,          0.2
      48,          0.16,          0.2
      49,          0.24,          0.2
      50,          0.32,          0.2
      51,           0.4,          0.2
      52,           0.,          0.24
      53,          0.04,         0.24
      54,          0.08,         0.24
```

```
 55,          0.12,          0.24
 56,          0.16,          0.24
 57,           0.2,          0.24
 58,          0.24,          0.24
 59,          0.28,          0.24
 60,          0.32,          0.24
 61,          0.36,          0.24
 62,           0.4,          0.24
 63,            0.,          0.28
 64,          0.08,          0.28
 65,          0.16,          0.28
 66,          0.24,          0.28
 67,          0.32,          0.28
 68,           0.4,          0.28
 69,            0.,          0.32
 70,          0.04,          0.32
 71,          0.08,          0.32
 72,          0.12,          0.32
 73,          0.16,          0.32
 74,           0.2,          0.32
 75,          0.24,          0.32
 76,          0.28,          0.32
 77,          0.32,          0.32
 78,          0.36,          0.32
 79,           0.4,          0.32
 80,            0.,          0.36
 81,          0.08,          0.36
 82,          0.16,          0.36
 83,          0.24,          0.36
 84,          0.32,          0.36
 85,           0.4,          0.36
 86,            0.,           0.4
 87,          0.04,           0.4
 88,          0.08,           0.4
 89,          0.12,           0.4
 90,          0.16,           0.4
 91,           0.2,           0.4
 92,          0.24,           0.4
 93,          0.28,           0.4
 94,          0.32,           0.4
 95,          0.36,           0.4
 96,           0.4,           0.4
 97,      0.420811,      0.420811
 98,      0.429575,       0.33855
 99,      0.436626,      0.255045
100,       0.44179,      0.170578
101,      0.444941,     0.0854568
102,         0.446,   -1.87959E-9
103,      0.441622,      0.441622
104,      0.450824,       0.39954
105,       0.45915,      0.357101
106,       0.46668,      0.313702
107,      0.473251,       0.27009
108,      0.478925,      0.225653
109,       0.48358,      0.181157
110,      0.487246,      0.136159
111,      0.489881,     0.0909138
112,       0.49147,     0.0454992
113,         0.492,   -3.75918E-9
114,      0.462434,      0.462434
```

```
115,      0.488725,      0.375651
116,      0.509877,      0.285135
117,       0.52537,      0.191735
118,      0.534822,     0.0963708
119,         0.538,   -5.63877E-9
120,      0.483245,      0.483245
121,      0.501648,      0.439237
122,        0.5183,      0.394201
123,       0.53336,      0.347561
124,      0.546503,       0.30018
125,      0.557851,      0.251461
126,       0.56716,      0.202313
127,      0.574517,      0.152123
128,      0.579763,      0.101828
129,      0.582939,     0.0509987
130,         0.584,   -7.51836E-9
131,      0.504056,      0.504056
132,      0.547875,      0.412751
133,      0.583128,      0.315225
134,       0.60895,      0.212891
135,      0.624703,      0.107285
136,           0.63,   -9.39795E-9
137,      0.524867,      0.524867
138,      0.552471,      0.478934
139,       0.57745,      0.431302
140,      0.600041,      0.381419
141,      0.619754,       0.33027
142,      0.636776,       0.27727
143,       0.65074,       0.22347
144,      0.661738,      0.168479
145,      0.669644,      0.112742
146,      0.674408,     0.0564983
147,         0.676,   -1.12775E-8
148,      0.545678,      0.545678
149,      0.607025,      0.449852
150,       0.65638,      0.345315
151,       0.69253,      0.234048
152,      0.714585,        0.1182
153,         0.722,   -1.31571E-8
154,       0.56649,       0.56649
155,      0.603295,      0.518631
156,      0.636599,      0.468403
157,      0.666721,      0.415278
158,      0.693005,      0.360359
159,      0.715621,      0.303305
160,       0.73432,      0.244626
161,      0.748984,      0.184639
162,      0.759525,      0.123657
163,      0.765878,      0.061998
164,         0.768,   -1.50367E-8
165,      0.587301,      0.587301
166,      0.666174,      0.486953
167,      0.729631,      0.375404
168,       0.77611,      0.255204
169,      0.804465,      0.129115
170,         0.814,   -1.69163E-8
171,      0.608112,      0.608112
172,      0.653947,      0.558528
173,      0.695749,      0.505503
174,      0.733263,      0.449361
```

```
175,      0.766257,      0.390449
176,      0.794527,      0.329131
177,        0.8179,      0.265782
178,       0.83623,      0.200799
179,      0.849406,      0.134572
180,      0.857347,      0.067498
181,          0.86,   -1.87959E-8
182,            0.,         0.446
183,     0.0854525,      0.444941
184,      0.170574,      0.441792
185,      0.255041,      0.436628
186,      0.338547,      0.429577
187,            0.,         0.492
188,     0.0454944,       0.49147
189,      0.090905,      0.489883
190,       0.13615,      0.487248
191,      0.181147,      0.483583
192,      0.225643,       0.47893
193,      0.270081,      0.473256
194,      0.313694,      0.466685
195,      0.357093,      0.459154
196,      0.399537,      0.450827
197,            0.,         0.538
198,     0.0963574,      0.534824
199,      0.191721,      0.525375
200,      0.285122,      0.509884
201,       0.37564,      0.488732
202,            0.,         0.584
203,     0.0509887,       0.58294
204,       0.10181,      0.579765
205,      0.152103,      0.574521
206,      0.202294,      0.567166
207,      0.251442,      0.557859
208,      0.300162,      0.546512
209,      0.347544,       0.53337
210,      0.394186,      0.518309
211,      0.439229,      0.501653
212,            0.,          0.63
213,      0.107262,      0.624707
214,      0.212868,      0.608957
215,      0.315203,       0.58314
216,      0.412732,      0.547887
217,            0.,         0.676
218,      0.056483,       0.67441
219,      0.112715,      0.669648
220,      0.168449,      0.661745
221,      0.223442,      0.650749
222,      0.277242,      0.636788
223,      0.330243,      0.619768
224,      0.381393,      0.600056
225,      0.431279,      0.577465
226,      0.478922,      0.552481
227,            0.,         0.722
228,      0.118167,      0.714589
229,      0.234015,       0.69254
230,      0.345284,      0.656395
231,      0.449825,      0.607043
232,            0.,         0.768
233,     0.0619773,      0.765879
234,      0.123619,      0.759531
```

```
235,      0.184599,      0.748993
236,      0.244589,      0.734332
237,      0.303267,      0.715637
238,      0.360324,      0.693023
239,      0.415243,      0.666742
240,      0.468371,      0.636621
241,      0.518614,      0.603308
242,            0.,         0.814
243,      0.129072,      0.804472
244,      0.255162,      0.776123
245,      0.375365,      0.729651
246,      0.486918,      0.666199
247,            0.,          0.86
248,     0.0674715,      0.857349
249,      0.134524,      0.849414
250,      0.200749,      0.836241
251,      0.265736,      0.817915
252,      0.329084,      0.794546
253,      0.390406,      0.766279
254,      0.449318,       0.73329
255,      0.505464,      0.695778
256,      0.558508,      0.653964
257,      0.657098,      0.608112
258,      0.733777,      0.503132
259,      0.795467,      0.387268
260,      0.840655,      0.262991
261,      0.868227,      0.132977
262,        0.8775,   -1.64464E-8
263,      0.706084,      0.608112
264,      0.740588,      0.555513
265,       0.77181,      0.500754
266,      0.800047,      0.443221
267,      0.824687,      0.384068
268,      0.845965,      0.322638
269,      0.863419,       0.26017
270,      0.877169,      0.196229
271,      0.887052,      0.131355
272,       0.89301,     0.0658364
273,         0.895,   -1.40969E-8
274,       0.75507,      0.608112
275,      0.809842,      0.498377
276,      0.853907,      0.380868
277,      0.886183,      0.257349
278,      0.905877,      0.129732
279,        0.9125,   -1.17474E-8
280,      0.804056,      0.608112
281,      0.827059,      0.552697
282,      0.847874,      0.495999
283,      0.866699,      0.437295
284,      0.883126,      0.377668
285,      0.897311,      0.316365
286,      0.908947,      0.254528
287,      0.918144,      0.191383
288,      0.924702,       0.12811
289,      0.928673,     0.0641611
290,          0.93,   -9.39795E-9
291,      0.853042,      0.608112
292,      0.885906,      0.493622
293,      0.912345,      0.374468
294,      0.931711,      0.251707
```

```
295,      0.943527,      0.126488
296,        0.9475,   -7.04846E-9
297,      0.902028,      0.608112
298,       0.91353,       0.54988
299,      0.923937,      0.491245
300,       0.93335,      0.431368
301,      0.941563,      0.371267
302,      0.948656,      0.310092
303,      0.954474,      0.248886
304,      0.959057,      0.187032
305,      0.962352,      0.124866
306,      0.964337,      0.062486
307,         0.965,   -4.69897E-9
308,      0.951014,      0.608112
309,      0.961969,      0.488867
310,      0.970782,      0.368067
311,      0.977237,      0.246066
312,      0.981176,      0.123244
313,        0.9825,   -2.34949E-9
314,            1.,      0.608112
315,            1.,      0.547301
316,            1.,       0.48649
317,            1.,      0.425678
318,            1.,      0.364867
319,            1.,      0.304056
320,            1.,      0.243245
321,            1.,      0.182434
322,            1.,      0.121622
323,            1.,     0.0608112
324,            1.,            0.
325,            0.,          0.8775
326,      0.132905,      0.868238
327,      0.262907,       0.84068
328,      0.387189,      0.795505
329,      0.503072,      0.733817
330,      0.608112,      0.657098
331,            0.,         0.895
332,     0.0658039,      0.893012
333,      0.131294,      0.887061
334,       0.19616,      0.877183
335,      0.260098,       0.86344
336,      0.322567,      0.845991
337,         0.384,      0.824717
338,       0.44316,      0.800079
339,      0.500703,      0.771841
340,      0.555488,      0.740606
341,      0.608112,      0.706084
342,            0.,        0.9125
343,      0.129682,      0.905884
344,      0.257289,      0.886199
345,      0.380811,       0.85393
346,      0.498334,      0.809866
347,      0.608112,       0.75507
348,            0.,          0.93
349,     0.0641397,      0.928675
350,       0.12807,      0.924707
351,      0.191338,      0.918152
352,      0.254481,      0.908959
353,      0.316317,      0.897326
354,      0.377622,      0.883143
```

```
355,      0.437254,      0.866717
356,      0.495964,      0.847891
357,      0.552679,      0.827069
358,      0.608112,      0.804056
359,            0.,        0.9475
360,      0.126458,       0.94353
361,      0.251672,      0.931719
362,      0.374434,      0.912357
363,      0.493596,      0.885917
364,      0.608112,      0.853042
365,            0.,         0.965
366,     0.0624755,      0.964337
367,      0.124846,      0.962353
368,      0.187009,      0.959061
369,      0.248863,      0.954479
370,      0.310068,      0.948663
371,      0.371245,      0.941571
372,      0.431347,      0.933357
373,      0.491227,      0.923944
374,      0.549871,      0.913534
375,      0.608112,      0.902028
376,            0.,        0.9825
377,      0.123234,      0.981177
378,      0.246054,      0.977239
379,      0.368056,      0.970785
380,      0.488858,      0.961972
381,      0.608112,      0.951014
382,            0.,            1.
383,     0.0608112,            1.
384,      0.121622,            1.
385,      0.182434,            1.
386,      0.243245,            1.
387,      0.304056,            1.
388,      0.364867,            1.
389,      0.425678,            1.
390,       0.48649,            1.
391,      0.547301,            1.
392,      0.608112,            1.
393,      0.706084,      0.657098
394,      0.804056,      0.657098
395,      0.902028,      0.657098
396,            1.,      0.657098
397,      0.657098,      0.706084
398,      0.706084,      0.706084
399,       0.75507,      0.706084
400,      0.804056,      0.706084
401,      0.853042,      0.706084
402,      0.902028,      0.706084
403,      0.951014,      0.706084
404,            1.,      0.706084
405,      0.706084,       0.75507
406,      0.804056,       0.75507
407,      0.902028,       0.75507
408,            1.,       0.75507
409,      0.657098,      0.804056
410,      0.706084,      0.804056
411,       0.75507,      0.804056
412,      0.804056,      0.804056
413,      0.853042,      0.804056
414,      0.902028,      0.804056
```

```
     415,      0.951014,      0.804056
     416,            1.,      0.804056
     417,      0.706084,      0.853042
     418,      0.804056,      0.853042
     419,      0.902028,      0.853042
     420,            1.,      0.853042
     421,      0.657098,      0.902028
     422,      0.706084,      0.902028
     423,       0.75507,      0.902028
     424,      0.804056,      0.902028
     425,      0.853042,      0.902028
     426,      0.902028,      0.902028
     427,      0.951014,      0.902028
     428,            1.,      0.902028
     429,      0.706084,      0.951014
     430,      0.804056,      0.951014
     431,      0.902028,      0.951014
     432,            1.,      0.951014
     433,      0.657098,            1.
     434,      0.706084,            1.
     435,       0.75507,            1.
     436,      0.804056,            1.
     437,      0.853042,            1.
     438,      0.902028,            1.
     439,      0.951014,            1.
     440,            1.,            1.
**
**
*ELEMENT, TYPE=CPE8, ELSET=PARTICLE
       1,       1,       3,      20,      18,       2,      13,
      19,      12
       2,       3,       5,      22,      20,       4,      14,
      21,      13
       3,       5,       7,      24,      22,       6,      15,
      23,      14
       4,       7,       9,      26,      24,       8,      16,
      25,      15
       5,       9,      11,      28,      26,      10,      17,
      27,      16
       6,      18,      20,      37,      35,      19,      30,
      36,      29
       7,      20,      22,      39,      37,      21,      31,
      38,      30
       8,      22,      24,      41,      39,      23,      32,
      40,      31
       9,      24,      26,      43,      41,      25,      33,
      42,      32
      10,      26,      28,      45,      43,      27,      34,
      44,      33
      11,      35,      37,      54,      52,      36,      47,
      53,      46
      12,      37,      39,      56,      54,      38,      48,
      55,      47
      13,      39,      41,      58,      56,      40,      49,
      57,      48
      14,      41,      43,      60,      58,      42,      50,
      59,      49
      15,      43,      45,      62,      60,      44,      51,
      61,      50
      16,      52,      54,      71,      69,      53,      64,
```

```
 70,      63
 17,      54,      56,      73,      71,      55,      65,
 72,      64
 18,      56,      58,      75,      73,      57,      66,
 74,      65
 19,      58,      60,      77,      75,      59,      67,
 76,      66
 20,      60,      62,      79,      77,      61,      68,
 78,      67
 21,      69,      71,      88,      86,      70,      81,
 87,      80
 22,      71,      73,      90,      88,      72,      82,
 89,      81
 23,      73,      75,      92,      90,      74,      83,
 91,      82
 24,      75,      77,      94,      92,      76,      84,
 93,      83
 25,      77,      79,      96,      94,      78,      85,
 95,      84
 26,      96,      79,     105,     103,      85,      98,
104,      97
 27,      79,      62,     107,     105,      68,      99,
106,      98
 28,      62,      45,     109,     107,      51,     100,
108,      99
 29,      45,      28,     111,     109,      34,     101,
110,     100
 30,      28,      11,     113,     111,      17,     102,
112,     101
 31,     103,     105,     122,     120,     104,     115,
121,     114
 32,     105,     107,     124,     122,     106,     116,
123,     115
 33,     107,     109,     126,     124,     108,     117,
125,     116
 34,     109,     111,     128,     126,     110,     118,
127,     117
 35,     111,     113,     130,     128,     112,     119,
129,     118
 36,     120,     122,     139,     137,     121,     132,
138,     131
 37,     122,     124,     141,     139,     123,     133,
140,     132
 38,     124,     126,     143,     141,     125,     134,
142,     133
 39,     126,     128,     145,     143,     127,     135,
144,     134
 40,     128,     130,     147,     145,     129,     136,
146,     135
 41,     137,     139,     156,     154,     138,     149,
155,     148
 42,     139,     141,     158,     156,     140,     150,
157,     149
 43,     141,     143,     160,     158,     142,     151,
159,     150
 44,     143,     145,     162,     160,     144,     152,
161,     151
 45,     145,     147,     164,     162,     146,     153,
163,     152
 46,     154,     156,     173,     171,     155,     166,
```

```
     172,     165
      47,     156,     158,     175,     173,     157,     167,
     174,     166
      48,     158,     160,     177,     175,     159,     168,
     176,     167
      49,     160,     162,     179,     177,     161,     169,
     178,     168
      50,     162,     164,     181,     179,     163,     170,
     180,     169
      51,      86,      88,     189,     187,      87,     183,
     188,     182
      52,      88,      90,     191,     189,      89,     184,
     190,     183
      53,      90,      92,     193,     191,      91,     185,
     192,     184
      54,      92,      94,     195,     193,      93,     186,
     194,     185
      55,      94,      96,     103,     195,      95,      97,
     196,     186
      56,     187,     189,     204,     202,     188,     198,
     203,     197
      57,     189,     191,     206,     204,     190,     199,
     205,     198
      58,     191,     193,     208,     206,     192,     200,
     207,     199
      59,     193,     195,     210,     208,     194,     201,
     209,     200
      60,     195,     103,     120,     210,     196,     114,
     211,     201
      61,     202,     204,     219,     217,     203,     213,
     218,     212
      62,     204,     206,     221,     219,     205,     214,
     220,     213
      63,     206,     208,     223,     221,     207,     215,
     222,     214
      64,     208,     210,     225,     223,     209,     216,
     224,     215
      65,     210,     120,     137,     225,     211,     131,
     226,     216
      66,     217,     219,     234,     232,     218,     228,
     233,     227
      67,     219,     221,     236,     234,     220,     229,
     235,     228
      68,     221,     223,     238,     236,     222,     230,
     237,     229
      69,     223,     225,     240,     238,     224,     231,
     239,     230
      70,     225,     137,     154,     240,     226,     148,
     241,     231
      71,     232,     234,     249,     247,     233,     243,
     248,     242
      72,     234,     236,     251,     249,     235,     244,
     250,     243
      73,     236,     238,     253,     251,     237,     245,
     252,     244
      74,     238,     240,     255,     253,     239,     246,
     254,     245
      75,     240,     154,     171,     255,     241,     165,
     256,     246
*ELEMENT, TYPE=CPE8, ELSET=MATRIX
```

```
 76,     171,     173,     265,     263,     172,     258,
264,     257
 77,     173,     175,     267,     265,     174,     259,
266,     258
 78,     175,     177,     269,     267,     176,     260,
268,     259
 79,     177,     179,     271,     269,     178,     261,
270,     260
 80,     179,     181,     273,     271,     180,     262,
272,     261
 81,     263,     265,     282,     280,     264,     275,
281,     274
 82,     265,     267,     284,     282,     266,     276,
283,     275
 83,     267,     269,     286,     284,     268,     277,
285,     276
 84,     269,     271,     288,     286,     270,     278,
287,     277
 85,     271,     273,     290,     288,     272,     279,
289,     278
 86,     280,     282,     299,     297,     281,     292,
298,     291
 87,     282,     284,     301,     299,     283,     293,
300,     292
 88,     284,     286,     303,     301,     285,     294,
302,     293
 89,     286,     288,     305,     303,     287,     295,
304,     294
 90,     288,     290,     307,     305,     289,     296,
306,     295
 91,     297,     299,     316,     314,     298,     309,
315,     308
 92,     299,     301,     318,     316,     300,     310,
317,     309
 93,     301,     303,     320,     318,     302,     311,
319,     310
 94,     303,     305,     322,     320,     304,     312,
321,     311
 95,     305,     307,     324,     322,     306,     313,
323,     312
 96,     247,     249,     333,     331,     248,     326,
332,     325
 97,     249,     251,     335,     333,     250,     327,
334,     326
 98,     251,     253,     337,     335,     252,     328,
336,     327
 99,     253,     255,     339,     337,     254,     329,
338,     328
100,     255,     171,     341,     339,     256,     330,
340,     329
101,     331,     333,     350,     348,     332,     343,
349,     342
102,     333,     335,     352,     350,     334,     344,
351,     343
103,     335,     337,     354,     352,     336,     345,
353,     344
104,     337,     339,     356,     354,     338,     346,
355,     345
105,     339,     341,     358,     356,     340,     347,
357,     346
```

```
     106,     348,     350,     367,     365,     349,     360,
     366,     359
     107,     350,     352,     369,     367,     351,     361,
     368,     360
     108,     352,     354,     371,     369,     353,     362,
     370,     361
     109,     354,     356,     373,     371,     355,     363,
     372,     362
     110,     356,     358,     375,     373,     357,     364,
     374,     363
     111,     365,     367,     384,     382,     366,     377,
     383,     376
     112,     367,     369,     386,     384,     368,     378,
     385,     377
     113,     369,     371,     388,     386,     370,     379,
     387,     378
     114,     371,     373,     390,     388,     372,     380,
     389,     379
     115,     373,     375,     392,     390,     374,     381,
     391,     380
     116,     171,     263,     398,     341,     257,     393,
     397,     330
     117,     263,     280,     400,     398,     274,     394,
     399,     393
     118,     280,     297,     402,     400,     291,     395,
     401,     394
     119,     297,     314,     404,     402,     308,     396,
     403,     395
     120,     341,     398,     410,     358,     397,     405,
     409,     347
     121,     398,     400,     412,     410,     399,     406,
     411,     405
     122,     400,     402,     414,     412,     401,     407,
     413,     406
     123,     402,     404,     416,     414,     403,     408,
     415,     407
     124,     358,     410,     422,     375,     409,     417,
     421,     364
     125,     410,     412,     424,     422,     411,     418,
     423,     417
     126,     412,     414,     426,     424,     413,     419,
     425,     418
     127,     414,     416,     428,     426,     415,     420,
     427,     419
     128,     375,     422,     434,     392,     421,     429,
     433,     381
     129,     422,     424,     436,     434,     423,     430,
     435,     429
     130,     424,     426,     438,     436,     425,     431,
     437,     430
     131,     426,     428,     440,     438,     427,     432,
     439,     431
**
** particle
**
*ORIENTATION, SYSTEM=R, NAME=OID1
          0.,          0.,          1.,          1.,          0.,
0.
       3,          0.
*SOLID SECTION, ELSET=PARTICLE, MATERIAL=WOOD, ORIENTATION=OID1
```

```
         1.,
**
** matrix
**
*SOLID SECTION, ELSET=MATRIX, MATERIAL=PLASTIC, ORIENTATION=OID1
         1.,
**
** wood
** Date: 15-Feb-:0           Time: 11:39:27
**
*MATERIAL, NAME=WOOD
**
*ELASTIC, TYPE=ENGINEERING CONSTANTS
     1.6+6     1.6+5     1.6+5       0.4       0.4       0.4    80000.
80000.
    16000.
**
** plastic
** Date: 15-Feb-:0           Time: 11:39:27
**
*MATERIAL, NAME=PLASTIC
**
*ELASTIC, TYPE=ISO
     200000.,         0.4
**
** Step  1, step1
** LoadCase, uniform_disp
**
*STEP, AMPLITUDE=RAMP, PERTURBATION
**
*STATIC
**
**
**
** axis_left
**
*BOUNDARY, OP=NEW
       1, 1,,          0.
       1, 2,,          0.
      12, 1,,          0.
      18, 1,,          0.
      29, 1,,          0.
      35, 1,,          0.
      46, 1,,          0.
      52, 1,,          0.
      63, 1,,          0.
      69, 1,,          0.
      80, 1,,          0.
      86, 1,,          0.
     182, 1,,          0.
     187, 1,,          0.
     197, 1,,          0.
     202, 1,,          0.
     212, 1,,          0.
     217, 1,,          0.
     227, 1,,          0.
     232, 1,,          0.
     242, 1,,          0.
     247, 1,,          0.
     325, 1,,          0.
```

```
     331, 1,,          0.
     342, 1,,          0.
     348, 1,,          0.
     359, 1,,          0.
     365, 1,,          0.
     376, 1,,          0.
     382, 1,,          0.
**
** axis_bottom
**
*BOUNDARY, OP=NEW
       2, 2,,          0.
       3, 2,,          0.
       4, 2,,          0.
       5, 2,,          0.
       6, 2,,          0.
       7, 2,,          0.
       8, 2,,          0.
       9, 2,,          0.
      10, 2,,          0.
      11, 2,,          0.
     102, 2,,          0.
     113, 2,,          0.
     119, 2,,          0.
     130, 2,,          0.
     136, 2,,          0.
     147, 2,,          0.
     153, 2,,          0.
     164, 2,,          0.
     170, 2,,          0.
     181, 2,,          0.
     262, 2,,          0.
     273, 2,,          0.
     279, 2,,          0.
     290, 2,,          0.
     296, 2,,          0.
     307, 2,,          0.
     313, 2,,          0.
**
** uniform_disp
**
*BOUNDARY, OP=NEW
     314, 1,,       0.001
     315, 1,,       0.001
     316, 1,,       0.001
     317, 1,,       0.001
     318, 1,,       0.001
     319, 1,,       0.001
     320, 1,,       0.001
     321, 1,,       0.001
     322, 1,,       0.001
     323, 1,,       0.001
     324, 1,,       0.001
     324, 2,,          0.
     396, 1,,       0.001
     404, 1,,       0.001
     408, 1,,       0.001
     416, 1,,       0.001
     420, 1,,       0.001
     428, 1,,       0.001
```

```
     432, 1,,        0.001
     440, 1,,        0.001
**
*CLOAD, OP=NEW
*DLOAD, OP=NEW
*TEMPERATURE, OP=NEW
**
*NODE PRINT, FREQ=1
U,
RF,
*NODE FILE, FREQ=1
U,
RF,
**
*EL PRINT, POS=INTEG, FREQ=1
S,
E,
*EL FILE,  POS=INTEG, FREQ=1
S,
E,
**
*EL PRINT, POS=NODES, FREQ=0
**
*EL FILE,  POS=NODES, FREQ=0
**
*EL PRINT, POS=CENTR, FREQ=0
**
*EL FILE,  POS=CENTR, FREQ=0
**
*EL PRINT, POS=AVERAGE, FREQ=0
**
*EL FILE,  POS=AVERAGE, FREQ=0
**
*MODAL PRINT, FREQ=99999
**
*MODAL FILE, FREQ=99999
**
*ENERGY PRINT, FREQ=0
**
*ENERGY FILE, FREQ=0
**
*PRINT, FREQ=1
**
*END STEP
```