Free and Forced Vibration of Linearly Elastic and St. Venant-Kirchhoff Plates using the Third Order Shear and Normal Deformable Theory

Arka P. Chattopadhyay

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

In

Engineering Mechanics

Romesh C. Batra

Mark S. Cramer

James Hanna

Shu-Ming Sun

Surot Thangjitham

9/20/2018

Blacksburg, VA

Keywords: Third Order Plate Theory, Finite Element Method, 3D Elasticity, Free Vibrations, Mode Localization, Vibration Attenuation, Material Nonlinearity, Finite Deformations

Free and Forced Vibration of Linearly Elastic and St. Venant-Kirchhoff Plates using the Third Order Shear and Normal Deformable Theory

Arka P. Chattopadhyay

ABSTRACT

Employing the Finite Element Method (FEM), we numerically study three problems involving free and forced vibrations of linearly and nonlinearly elastic plates with a third order shear and normal deformable theory (TSNDT) and the three dimensional (3D) elasticity theory. We used the commercial software ABAQUS for analyzing 3D deformations, and an in-house developed and verified software for solving the plate theory equations.

In the first problem, we consider trapezoidal load-time pulses with linearly increasing and affinely decreasing loads of total durations equal to integer multiples of the time period of the first bending mode of vibration of a plate. For arbitrary spatial distributions of loads applied to monolithic and laminated orthotropic plates, we show that plates' vibrations become miniscule after the load is removed. We call this phenomenon as vibration attenuation. It is independent of the dwell time during which the load is a constant. We hypothesize that plates exhibit this phenomenon because nearly all of plate's strain energy is due to deformations corresponding to the fundamental bending mode of vibration. Thus taking the 1^{st} bending mode shape of the plate vibration as the basis function, we reduce the problem to that of solving a single second-order ordinary differential equation. We show that this reduced-order model gives excellent results for monolithic and composite plates subjected to different loads.

Rectangular plates studied in the 2nd problem have points on either one or two normals to their midsurface constrained from translating in all three directions. We find that deformations corresponding to several modes of vibration are annulled in a region of the plate divided by a plane through the constraining points; this phenomenon is termed mode localization. New results include: (i) the localization of both in-plane and out-of-plane modes of vibration, (ii) increase in the mode localization intensity with an increase in the length/width ratio of a rectangular plate, (iii) change in the mode localization characteristics with the fiber orientation angle in unidirectional fiber- reinforced laminae, (iv) mode localization due to points on two normals constrained, and (iv) the exchange of energy during forced harmonic vibrations between two regions separated by the line of nearly stationary points that results in a beating-like phenomenon in a sub-region of the plate. This technique can help design a structure with vibrations limited to its small sub-region, and harvesting energy of vibrations of the sub-region.

In the third problem, we study finite transient deformations of rectangular plates using the TSNDT. The mathematical model includes all geometric and material nonlinearities. We compare the results of linear and nonlinear TSNDT FEM with the corresponding 3D FEM results from ABAQUS and note that the TSNDT is capable of predicting reasonably accurate results of displacements and in-plane stresses. However, the errors in computing transverse stresses are larger and the use of a two point stress recovery scheme improves their accuracy. We delineate the effects of nonlinearities by comparing results from the linear and the nonlinear theories. We observe that the linear theory over-predicts the deformations of a plate as compared to those obtained with the inclusion of geometric and material nonlinearities. We hypothesize that this is an effect of stiffening of the material

due to the nonlinearity, analogous to the strain hardening phenomenon in plasticity. Based on this observation, we propose that the consideration of nonlinearities is essential in modeling plates undergoing large deformations as linear model over-predicts the deformation resulting in conservative design criteria. We also notice that unlike linear elastic plate bending, the neutral surface of a nonlinearly elastic bending plate, defined as the plane unstretched after the deformation, does not coincide with the mid-surface of the plate. Due to this effect, use of nonlinear models may be of useful in design of sandwich structures where a soft core near the mid-surface will be subjected to large in-plane stresses.

Free and Forced Vibration of Linearly Elastic and St. Venant-Kirchhoff Plates using the Third Order Shear and Normal Deformable Theory

Arka P. Chattopadhyay

General Audience Abstract

Plates and shells are defined as structures which have thickness much smaller as compared to their length and width. These structures are extensively used in many fields of engineering such as, designing ship hulls, airplane wings and fuselage, bodies of automobile, etc. Depending on the complexity of a plate/shell deformation problem, deriving analytical solutions is not always viable and one relies on computational methods to obtain numerical solutions of the problem. However, obtaining 3-dimensional (3D) numerical solutions of deforming plates/shells often require high computational effort. To avoid this, plate/shell theories are used for modeling these structures, which, based on certain assumptions, reduce the 3D problem into an equivalent 2-dimensional (2D) problem. However, quality of the solution obtained from such a theory depends on how suitable the assumptions are for the specific problem being studied.

In this work, one such plate theory called as the Third Order Shear and Normal Deformable Theory (TSNDT) is used to model the mechanics of deforming rectangular plates under different boundary conditions (constraint conditions for the boundaries of the plate) and loading conditions (conditions of applied loads on the plate). We develop the TSNDT mathematical model of plate deformations and solve it using a computational technique called as the Finite Element Method (FEM) to analyze three different problems of mechanics of rectangular plates. These problems are briefly described below.

In the first problem, we study vibrations of rectangular plates under time dependent (dynamic) loads. When a dynamic load acts on a plate, due to the effects of inertia, the plate continues to vibrate after the removal of the load. This is analogous to ringing of a bell long after the strike of the hammer on the bell. In this study we show that such vibrations of a rectangular plate can be varied by changing time dependencies of the applied load. We observe that under certain particular loading conditions, vibrations of the plate becomes miniscule after the load removal. We call this phenomenon as *Vibration Attenuation* and investigate this computationally in different problems of plate deformation using FEM solutions.

In the second problem, we computationally investigate the effects of presence of internal fixed points (points within the volume of the plate restricted of motion) on the vibration characteristics of rectangular plate using TSNDT FEM solutions. We observe that when one or more points at locations inside a rectangular plate are fixed, vibration behavior of the plate significantly changes and the deformations are localized in certain regions of the plate. This phenomenon is called as *Mode Localization.* We study mode localization in rectangular plates under different boundary and loading conditions and analyze the effects of plate dimensions, locations of the internal fixed points and dynamic load characteristics on mode localization.

In the third problem, we investigate the effects of introduction of nonlinearities into the TSNDT mathematical model of plate deformations. Simple models in mechanics consider materials to be linearly elastic, which means that the deformations of a body are proportional to the applied loads in a linear relation. However, most materials in nature undergoing large deformations (human tissues, rubbers, and polymers, for example) do not

behave in this fashion and their deformation depends nonlinearly to applied loads. To investigate the effects of such nonlinearities, we study the behavior of nonlinearly elastic plates under different boundary and loading conditions and delineate the differences in the results of linearly elastic and nonlinearly elastic plates using the TSNDT FEM solutions. Findings of this study establishes that linear models overestimate the plate deformation under given boundary and loading conditions as compared to nonlinear models. This understanding may help in developing better design criteria for plates undergoing large deformations.

Acknowledgements

First, I humbly thank my advisor, Dr. Romesh C. Batra, for his guidance and patience throughout my dissertation work. His passion and enthusiasm for research is something that has constantly inspired me and I consider myself privileged for the opportunity to learn from him. I am grateful to my committee members, Dr. Mark Cramer, Dr. James Hanna, Dr. Shu-Ming Sun and Dr. Surot Thangjitham for their valuable time and suggestions towards my research. I would also like to thank Mr. Tim Tomlin for his help with the computational resources related to my work.

I would like to thank my friends and colleagues who have helped in making the journey easier. I especially thank Qian Li, Priyal Shah and Balachandar Guduri for their intense and often fruitful technical discussions about research problems. I would also like to thank Karthik Venkatramani, Naveen Prakash, Riju Balachandran, Mary Weidner, Jared Wilmoth and Madhumati Mukherjee who have helped me endure in this quest.

Pursuit of my ambitions thus far would not have been possible without the endless love and support of my parents. They have been a constant source of motivation and I am grateful to them for their belief in me. Finally, I thank my wife, Nabaneeta Biswas, for encouraging me and guiding me through the ups and downs in this journey. She has been an inspiration and a support pillar throughout.

This work was partially supported by US Office of Naval research (ONR) grants N00014-11-1-0594, N00014-16-1-2309 and N00014-18-1-2548 to Virginia Polytechnic Institute and State University with Dr. Y.D.S. Rajapakse as the program manager. Views expressed in the paper are those of the author and neither of ONR nor of Virginia Tech. The author is solely responsible for errors, if any, in the thesis.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

Plates and shells are defined as structures with thickness much smaller than the planar dimensions. Plate and shell theories take advantage of this small thickness to reduce equations of the three-dimensional (3D) elasticity theory to those defined on a reference surface of the plate/shell. There are numerous such theories in the literature that are distinguished from each other based on the assumptions made to derive them. Kirchhoff-Love theory [1-2], Reissner-Mindlin theory [3-4], the Higher order Shear Deformation Theory (HSDT) [5], to name a few, are some of the widely used plate theories. These differ from each other based on restrictions imposed on deformations of a line normal to the plate mid-surface; a few plate theories are briefly reviewed in Chapter 4 of this dissertation. The assumptions made in deriving the plate theory determine the quality and accuracy of the solution as compared to the exact solution obtained from the 3D elasticity theory.

Following the classical work of Mindlin [4], Batra and Vidoli [6-7] derived a compatible and a mixed K^{th} order plate theory. In the former, all three displacements of a point are expressed as complete polynomials of degree K of the thickness coordinate, z, and stresses are deduced from the constitutive relation, strain-displacement relations and the plate theory displacements. In the mixed theory, both stresses and displacements are taken as unknowns at a material point and are expressed as complete polynomials, respectively, of degree K+2 and K in z. As expected, the mixed theory gives better values of all six stress components than the compatible theory. Carrera [8] developed a variable order compatible plate theory in which the in-plane and the transverse displacements can be expressed as polynomials of different order in z.

In this work, we computationally study free and forced vibrations of rectangular plates under different initial, boundary and loading conditions using the Third order Shear and Normal Deformable Plate Theory (TSNDT) [Batra and Vidoli's compatible theory with K = 3] and employing the Finite Element Method (FEM). We develop a mathematical model of linearly/nonlinearly elastic plates, and find an approximate solution of the governing equations by the FEM with an in-house developed software in FORTRAN. The TSNDT does not require using a shear correction factor. We verify the software by comparing results obtained from it with either those available in the literature or by numerically solving the 3-dimensional (3D) elasticity equations with the commercial software, ABAQUS. Furthermore, the transverse shear and normal stresses are found by using a one-step stress recovery scheme. That is, after having solved the plate theory equations and found the three in-plane stresses by using the constitutive relation and the TSNDT displacements, we integrate with respect to z the three equations of motion to find through-the-thickness transverse shear and normal stresses. We list below problems analyzed for plates of aspect ratios (length/height) 5 and higher. In general, the number of Degrees of Freedom (DoFs) in the TSNDT is approximately $1/20^{th}$ of that in the analysis of the corresponding 3-D problem with the two solutions differing by about 5%.

In chapter 2, the phenomenon termed as v*ibration attenuation* is studied in rectangular and circular plates. In the third chapter, the *mode localization* phenomenon in plates is investigated, and in the fourth chapter, finite transient responses of nonlinearly elastic plates under different loading and boundary conditions are studied. The works in the three

chapters are briefly reviewed below with those in chapters 2 and 3 already published in peer-reviewed journals.

1.1 Vibration Attenuation

For motivating vibration attenuation in a plate, we first analytically analyze the motion of a linear spring mass system under a time-dependent trapezoidal force that increases linearly with time in time t_1, stays constant for time t_2, drops affinely to zero in time t_3, and then stays at zero. We find that the mass initially at rest returns to rest for all times greater than the loading time of $t_1 + t_2 + t_3$ when t_1 and t_3 are integer multiples of the fundamental time-period of the spring mass system. We refer to this phenomenon as load-dependent *vibration attenuation*. Subsequently, we investigate this phenomenon in linearly elastic continuous structures, by studying 3-D deformations of monolithic, fiber-reinforced laminated and functionally graded (FG) rectangular plates under trapezoidal pulse loads, using the commercial FEM software, ABAQUS. Loads were applied in the form of normal tractions either on the entire or on a part of the major surfaces of the plate inducing bending dominant deformations, and the times t_1 and t_3 were considered to be integer multiples of the time period of the first bending mode of vibrations. We also analyzed the plate deformations by taking the mass matrix normalized mode shapes of free bending vibrations of the plate as basis functions. Thus equations of motion for different mode shapes are uncoupled, and the strain energy for each mode of deformation can be computed. For each problem studied, it is found that most of the strain energy of the plate is due to its deformations in the first bending mode of vibrations with other modes contributing to less than 5% of the total strain energy. It is observed that subsequent to the load removal, the

average acceleration of the plate oscillates around zero with a significantly smaller amplitude than that prior to the load removal.

The practical significance of this work is that for trapezoidal loads of the time duration considered here, no external stimuli (e.g., dampers) is needed to significantly reduce the amplitude of free vibrations of the system after the load removal.

1.2 Mode Localization

The presence of discontinuities and irregularities in a system may result in anomalies in its physical properties. Anderson [9] first observed that irregularities in electron distributions in different lattice structures of metals vary their vibration characteristics and the material conductivity; this phenomenon is called Anderson's localization [10]. Hodges [10] investigated if the localization phenomenon is also exhibited in macroscale vibrations of continuous bodies. Subsequently, numerous works have illustrated the mode localization phenomenon in continuous structures [11-17].

In the present work, we focused on the mode localization phenomenon in free and forced vibrations of rectangular plates introduced due to the presence of constrained internal points on either one or two normals to the plate mid-surface. We found that the mode localization, including of in-plane modes of vibration, becomes more distinguishable with an increase in the length/width (i.e., aspect) ratio. For forced vibration of a plate under a harmonic excitation on the entire top surface with internal points constrained, we observed that the plate is divided into two or more sections vibrating at different frequencies. Depending on the excitation frequency, the plate exhibits a *beats* like phenomenon which

we hypothesize is due to the interactions between sections of the plate separated by the internal clamped point(s). This may enable using mode localization as a tool to design structures with desirable vibration characteristics.

1.3 Characteristics of Finite Transient Deformations of Nonlinear Elastic Plates

We study transient deformations of rectangular plates using the FEM and the TSNDT to delineate differences in responses of a plate made of either a Hookean material or a St.-Venant-Kirchhoff material. Deformations analyzed are infinitesimal for a Hookean material but finite for a St. Venant-Kirchhoff material. It is found that under otherwise identical conditions, plate's maximum deflection is less when the plate material is St. Venant-Kirchhoff than that for Hookean material. One significant difference between the plate theory and the 3-D elasticity theory results is that in the former no waves propagate in the thickness direction when tractions are applied on the plate top surface but in the latter they do. Thus the use of the TSNDT can predict premature failure of the plate under a severe shock load.

2 Load's Temporal Characteristics for Annulling Forced Vibrations of Linear Elastic Plates

This chapter constitutes the paper published in Mechanics Research Communication, 85, 5-11, 2017; https://doi.org/10.1016/j.mechrescom.2017.07.009.

ABSTRACT

For a trapezoidal load pulse with linearly increasing and affinely decreasing portions of load-time durations equal to integer multiples of the fundamental time period of the first bending mode of vibration of linearly elastic and orthotropic laminated plates, it is shown that the plate vibrations become miniscule upon removal of the load. This attenuation of forced vibrations upon load removal is independent of the dwell time (i.e., the time duration between the rising and the falling portions) during which the load is kept constant and of the spatial distribution of the transverse load on the plate major surfaces. The primary reason for this response is that for such time-dependent loads, nearly all of the plate strain energy is concentrated in deformations corresponding to the fundamental bending mode of vibration, and the plate deformations can be studied by taking the mode shape of the 1^{st} bending mode as the basis function. It thus reduces the problem to that of solving a single second-order ordinary differential equation. We have verified this postulate by comparing strain energies computed from the 3-dimensional deformations of different plate geometries and boundary conditions found by using the finite element methodology based commercial software, ABAQUS, and those determined by using the single degree of

freedom model. Thus for trapezoidal time-dependent loads applied on plates, the 1-degree of freedom model provides reasonably accurate results.

Keywords : Load duration; Fundamental bending frequency; Vibration attenuation; Linearly elastic structures

2.1 Introduction

The 2nd order ordinary differential equation (ODE) with constant coefficients governing the forced motion of a linear spring-mass system under given initial conditions can be analytically solved. For a time-dependent trapezoidal force that increases linearly with time in time t_1, stays constant for time t_2, drops affinely to zero in time t_3, and then stays at zero, the analytical solution gives that the mass initially at rest comes to rest for all times greater than the loading time of $t_1 + t_2 + t_3$. We refer to this phenomenon as *load-dependent vibration attenuation*. This observation for the spring-mass system inspired us to *investigate* if a similar result holds for linearly elastic continuous structures for which t_1 and t_3 are integer multiples of the time period of the first *bending* mode of vibrations of the plate, and the load is applied either on the entire or on a part of the major surfaces of the plate inducing bending dominant deformations. For very thick plates the fundamental mode of vibration may involve only in-plane motions with null transverse displacements, e.g., see [18-19]. However, we focus on studying problems having bending-dominant deformations. The load-dependent vibration attenuation result has been verified by analyzing 3-dimensional deformations of linearly elastic square and circular monolithic, fiber-reinforced laminated and functionally graded (FG) plates by the finite element

method (FEM) using the commercial software, ABAQUS [20]. We also analyze the plate deformations by taking the mode shapes of free bending vibrations of the plate as basis functions that uncouples equations of motion for different mode shapes, and compute the plate strain energy for deformations corresponding to a desired mode shape. For each problem studied with the above-specified load – time curve, it is found that most of the strain energy of deformations of the plate is due to its deformations in the first bending mode of vibrations with other modes contributing to less than 5% of the total strain energy. It is observed subsequent to load removal, average acceleration of the plate oscillates around zero with a significantly smaller amplitude than that prior to the load removal.

The practical significance of the result is that for trapezoidal loads of the duration considered here, there is no need for external stimuli (e.g., dampers) to reduce significantly the amplitude of free vibrations of the system after the load removal.

In several books on dynamic problems for discrete and continuous systems, e.g., see [21-22], we have not seen this result. We also have not found any experimental work for the load-time variations envisaged here.

2.2 Forced vibrations of a linear spring-mass system

For a linear spring-mass system of mass m, spring constant k, initially at rest, and subjected to a trapezoidal time-dependent load, $F(t)$, depicted in Figure 2.1 and described by

$$F(t)=\frac{1}{t_1}t\left[H(t)\text{-H}(t-t_1)\right]$$
$$+\left[H(t-T_1)-H(t-T_2))\right]-\frac{1}{t_3}(t-(t_1+t_2+t_3))\left[H(t-T_2)-H(t-T_3))\right] \quad (2.1)$$
$$T_3=t_1+t_2+t_3, T_2=t_1+t_2, T_1=t_1$$

the displacement, $x(t)$, of the spring from its initial unstretched position is given by

$$x(t)=\frac{F_{\max}}{m\omega^2}\left[\begin{array}{l}\frac{1}{t_3}H[t-T_3]\left[\begin{array}{l}\{t-T_3\}\\-\frac{1}{\omega}\sin(\omega\{t-T_3\})\end{array}\right]-\frac{1}{t_3}H[t-T_2]\left[\begin{array}{l}\{t-T_2\}\\-\frac{1}{\omega}\sin(\omega\{t-T_2\})\end{array}\right]\\-\frac{1}{t_1}H(t-T_1)\left[\begin{array}{l}(t-T_1)\\-\frac{1}{\omega}\sin(\omega(t-T_1))\end{array}\right]+\frac{1}{T_1}\left(t-\frac{1}{\omega}\sin(\omega t)\right)\end{array}\right], \quad (2.2)$$

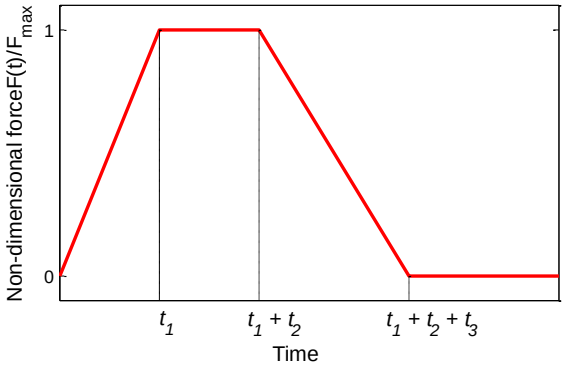


Figure 2.1 Trapezoidal load with rise time t_1, dwell time t_2, and fall time t_3

In Eqs. (2.1) and (2.2), $H(y)$ is the Heaviside step function that equals 1 for $y \geq 0$ and 0 otherwise, and $\omega=\sqrt{k/m}$ is the natural frequency of the linear spring-mass system in rad/s.

After removal of the load, the displacement of the mass in the free vibration state for time $t > T_3$ is given by

$$x(t) = \frac{F_{\max}}{m\omega^2}\left[\begin{array}{l} -\frac{1}{\omega t_3}\left[\sin\omega t\begin{pmatrix}\cos\omega T_3 - \\ \cos\omega(T_3 - t_3)\end{pmatrix} - \cos\omega t\begin{pmatrix}\sin\omega T_3 - \\ \sin\omega(T_3 - t_3)\end{pmatrix}\right] \\ +\frac{1}{\omega t_1}\left[\sin\omega t(\cos\omega t_1 - 1) - \cos\omega t\sin\omega t_1\right] \end{array}\right] \tag{2.3}$$

By using simple trigonometric identities, we write Eq. (2.3) as

$$x(t) = -\frac{2F_{\max}}{m\omega^3}\left[\begin{array}{l} \frac{\sin(\omega t_3/2)}{t_3}\left[\sin\omega t\begin{pmatrix}\cos\omega T_3\sin(\omega t_3/2) \\ -\sin\omega T_3\cos(\omega t_3/2)\end{pmatrix}\right] \\ -\cos\omega t\begin{pmatrix}\sin\omega T_3\sin(\omega t_3/2) \\ -\cos\omega T_3\cos(\omega t_3/2)\end{pmatrix}\Bigg] \\ +\frac{\sin(\omega t_1/2)}{t_1}\left[\begin{array}{l}\sin\omega t\sin(\omega t_1/2) \\ +\cos\omega t\cos(\omega t_1/2)\end{array}\right] \end{array}\right] \tag{2.4}$$

We conclude from Eq. (2.4) that $x(t) = 0$ for $t > T_3$ if and only if

$$t_1 = \frac{2n_1\pi}{\omega}, t_3 = \frac{2n_3\pi}{\omega} \tag{2.5}$$

That is, if the linearly increasing loading and the affinely decreasing unloading times are integer multiples of the time period of the fundamental frequency of free vibration, then, irrespective of the dwell time, t_2, between the loading and the unloading times, the spring mass-system ceases to vibrate after the load is removed. *We call this phenomenon of vanishing of post-load removal vibrations as load-dependent vibration attenuation.* The result holds even if t_1 and t_3 equal zero if the load duration t_2 is given by

$$t_2 = \frac{2n_2\pi}{\omega} \tag{2.6}$$

For m = 1 kg, $k = 100\pi^2$ N/m, $t_1 = t_3$ = 0.2 s, as shown in Figure 2.2, the motion of the mass ceases upon removal of the load. Since $\omega = 10\pi$ for this example problem, we have set $n_1 = n_2 = 1$ in Eqs. (2.4) and 2.5). Results plotted in Figure 2.2(d) confirm the necessity of conditions (1.5) since for $n_1 = 1.25$ the mass has a steady oscillatory motion after removal of the load.

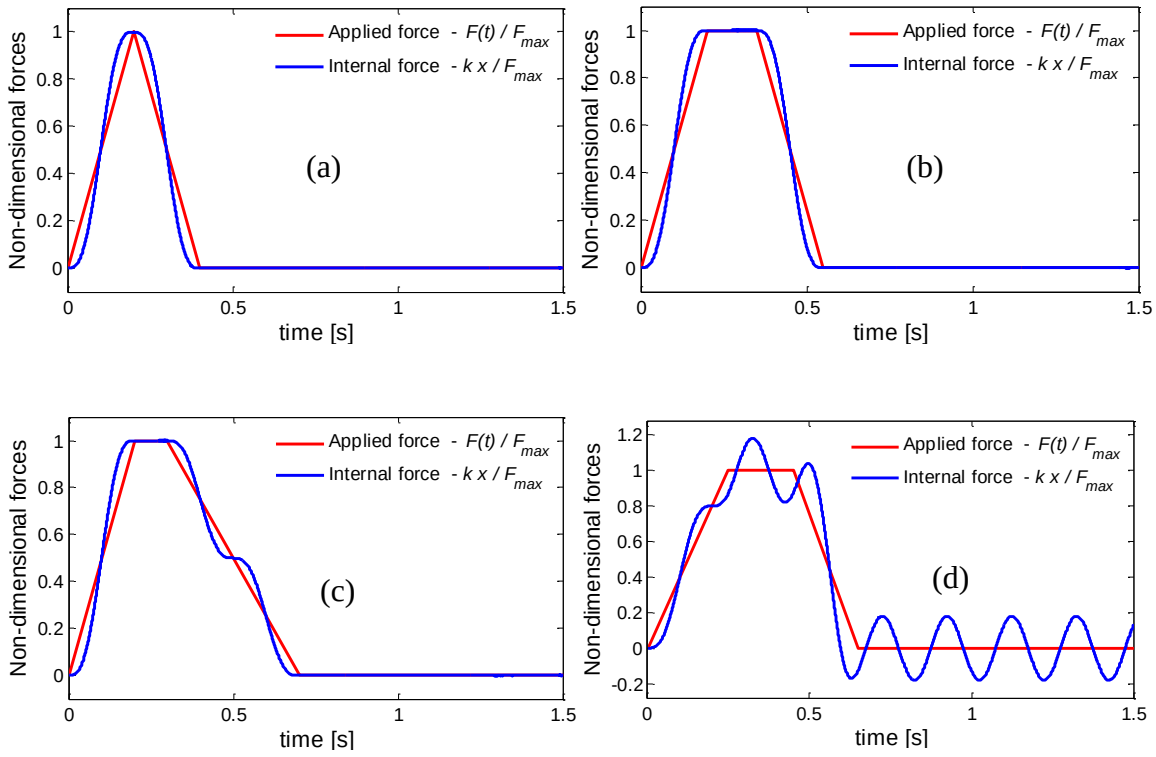


Figure 2.2 Time histories of the displacement of the mass for (a) t_1 = 0.2 s, t_2 = 0 and t_3 = 0.2 s, (b) t_1 = 0.2 s, t_2 = 0.15 s and t_3 = 0.2 s, (c) t_1 = 0.2 s, t_2 = 0.1 s and t_3 = 0.4 s, and (d) t_1 = 0.25 s, t_2 = 0.2 s and t_3 = 0.2 s

2.3 Vibration attenuation of Simply Supported and Clamped Plates

We *hypothesize* that for a linearly elastic plate with the normal traction applied on one or both of its major surfaces varying in time as shown in and the loading and the unloading time durations related to the time period of its *first frequency of free bending vibrations* by Eq. (5), the plate essentially comes to rest upon removal of the load. We note that for thick plates, the fundamental frequency may correspond to in-plane vibrations for which transverse displacements identically vanish. We exclude these by restricting ourselves to bending-dominant deformations of the plate. Furthermore, we assume that tractions applied to a major surface of the plate are of the form, $g(t)\, q(\mathbf{x})$. Whereas $g(t)$ must conform to the above-mentioned requirement, $q(\mathbf{x})$ can be an arbitrary function of points $\mathbf{x}$ on plate's major surface.

We numerically show this for a simply supported square plate (SSP), a clamped square plate (CSP), a clamped circular plate (CCP), a clamped laminated fiber-reinforced composite plate (CLP), and a rectangular clamped FG plate (CFGP). Transient 3-dimensional deformations of plates are numerically analyzed by the finite element method (FEM) with the commercial software, ABAQUS [20], using 8-node brick elements with the 2 x 2 x 2 Gauss integration rule. The FE mesh is successively refined and the time step size decreased to get a converged solution of the problem.

The primary reason for the plate response being similar to that of a linear spring-mass system is that for the function $g(t)$ satisfying the hypothesis, nearly all of the strain energy of deformations is concentrated in the first bending mode of vibration. We verify this for the example problems studied herein.

Taking the mass-normalized N eigenvectors, $\mathbf{A}^{(i)}$, of free vibrations of a linearly elastic plate as base vectors, the 3-D displacements $\mathbf{d}\left(x_1, x_2, x_3, t\right)$ of N nodes can be written as

$$\mathbf{d}\left(x_1, x_2, x_3, t\right) = \sum_{i=1}^{N} \lambda^{(i)}\left(t\right) \mathbf{A}^{(i)}\left(x_1, x_2, x_3\right) \tag{2.7}$$

where $\lambda^{(i)}\left(t\right)$ is the time-dependent component of $\mathbf{d}$ along the eigenvector $\mathbf{A}^{(i)}$. Referring the reader to Hughes' text book [23] on the FEM, the 2nd order ODE for the evolution of $\lambda^{(i)}\left(t\right)$ is

$$\ddot{\lambda}^{(i)} + \left(\omega^{(i)}\right)^2 \lambda^{(i)} = f^{(i)}\left(t\right) \tag{2.8}$$

where $f^{(i)}\left(t\right)$ is the component of the external nodal load vector along the eigenvector $\mathbf{A}^{(i)}$. For finding the strain energy of deformation corresponding to the i^{th} mode of vibration, we solve the ODE (8) for $\lambda^{(i)}$, use Eq. (2.7) to determine nodal displacements without summing on the index *i*, and then compute the strain energy from the stiffness matrix and the nodal displacements.

Results computed by taking *N*=1 in Eq. (2.7) are indicated in the Figures as mode 1. For some example problems, we show that the plate strain energy for deformations corresponding to the 2nd mode of bending vibration is negligible as compared to that of the total plate strain energy. For all example problems studied, the fundamental mode of vibration was found to due be bending.

1.1. Simply supported plate (SSP)

With reference to the coordinate axes and the plate dimensions shown in Figure 2.3, we enforce the following boundary conditions on the plate edges:

$$\begin{aligned} &\text{on } x_1 = 0, l,\ u_2 = u_3 = 0,\ \sigma_{11}=0, \\ &\text{on } x_2 = 0, b,\ u_1 = u_3 = 0, \sigma_{22} = 0 \\ &\text{on } x_3 = \pm h/2 : \sigma_{13} = \sigma_{23} = \sigma_{33} = 0 \end{aligned} \tag{2.9}$$

For a 100 mm x 100 mm x 10 mm plate with Young's modulus, E = 200 GPa, Poisson's ratio, ν = 0.3 and the mass density, ρ = 7.2 g/cc, we have listed in Table 2.1 the first five converged natural frequencies computed with a uniform mesh of 51,200 (80x80x8) 8-node elements. We have also provided the corresponding frequencies from the analytical solution for bending vibrations of Srinivas and Rao [24]. Results from the higher-order plate theory are also listed to show that in-plane modes of vibration exist for thick plates [18] and [25]. It is clear that the presently computed first three bending frequencies are within 1% of the analytical solutions of Srinivas and Rao [9], and the next two frequencies of in-plane modes of vibration agree well those from the higher-order plate theory. As pointed out by Batra and Aimamanee, the in-plane modes of vibration (modes 4 and 5) are absent in the analytical solution of Srinivas and Rao [24] since they tacitly studied bending vibrations.

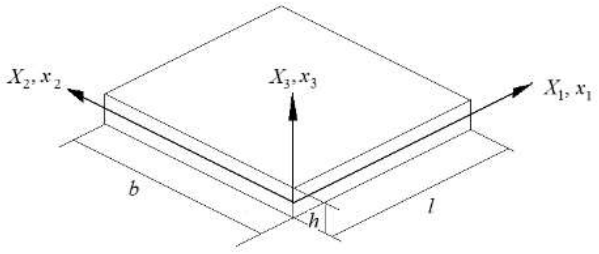


Figure 2.3 Schematic representation of the geometry of the plate

Table 2.1 Non-dimensional frequencies $\omega_n = \omega h\sqrt{\rho/E}$ of the 100 mm x 100 mm x 10 mm simply supported rectangular plate (* denotes in-plane mode of vibration).

Mode	Srinivas and Rao [24]	Qian et al [25]	Batra and Aimmanee [18]	Present 3D FEM
1	0.0578	0.0578	0.0578	0.0577
2	0.1381	0.1391	0.1391	0.1378
3	0.1381	0.1391	0.1391	0.1378
4	-	0.1948*	0.1949	0.1948*
5	-	0.1948*	0.1949	0.1978*

To study the vibration attenuation phenomena, we consider the SSP and take E = 25 GPa, ν = 0.25 and ρ = 2.5 g/cc. Hereafter, unless otherwise mentioned, we use these values of material properties in all example problems. The frequency of mode 1 vibrations of the SSP from the FE solution of the 3D problem was obtained as 2.847 Hz or equivalently the time period of the fundamental mode = 351 μs . The computed time history of the deflection of the centroid of the top surface and the strain energy of the plate with $q(\mathbf{x})$ = 0.1 GPa applied on the top surface, and $g(t)$ as a triangular pulse with rise and fall times of 351 μs is compared in Figure 2.4 with that found by taking only one basis function to solve the 3-D linear elasticity equations. These results evince that the entire plate essentially ceases to vibrate (within the scales of plotting) after the load removal and most of the plate strain energy is concentrated in the first mode of vibration. Furthermore, the plate centroidal deflections time-histories from the two approaches are very close to each other.

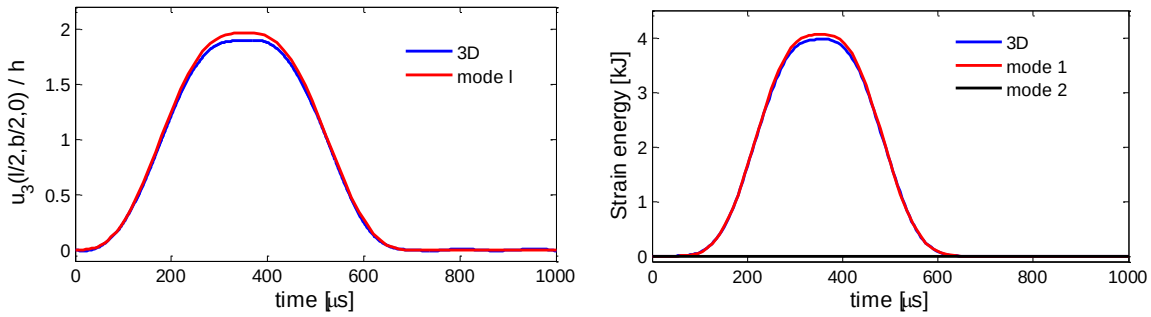


Figure 2.4 Time histories of the out-of-plane displacement of the centroid of the plate (left), and the strain energy of the plate in modes 1 and 2 of deformations, as well as in all modes of deformations (right).

1.2. Clamped Square Plate (CSP)

The converged fundamental frequency of bending vibrations and the corresponding time period of the 100 mm x 100 mm x 10 mm CSP plate equal 4.89 kHz and 204.5 μs, respectively. For $q(\mathbf{x}) = 0.1$ GPa on plate's top surface and with the load rise and fall times, $t_1 = t_3 = t_1^0 = 204.5\,\mu s$, and for $\pm 10\%$ and $\pm 20\%$ variations in t_1 and t_3, we have exhibited in Figure 2.5 time histories of the centroidal deflection found solving the 3-D problem. It is evident that the plate motion virtually ceases upon removal of the load for $t_1 = t_3 = t_1^0$ but not for the other load durations. The time along the horizontal axis for each curve is normalized by t_1.

In Figure 2.6 we have plotted time histories of the centroidal deflection and the total strain energy of the CSP for the trapezoidal load pulse, $g(t)$, with $t_1 = t_3 = t_1^0$ and dwell times t_2 of 0.5 t_1, 0.75 t_1 and 1.5 t_1 obtained from the 3-D and the mode-1 deformations. It is clear that the vibration attenuation is independent of the value of t_2, the centroidal deflection is

the maximum at the end of the linearly increasing load, stays constant during the dwell time, decreases to zero in the unloading phase and subsequent to the load removal it oscillates around zero with negligible amplitude.

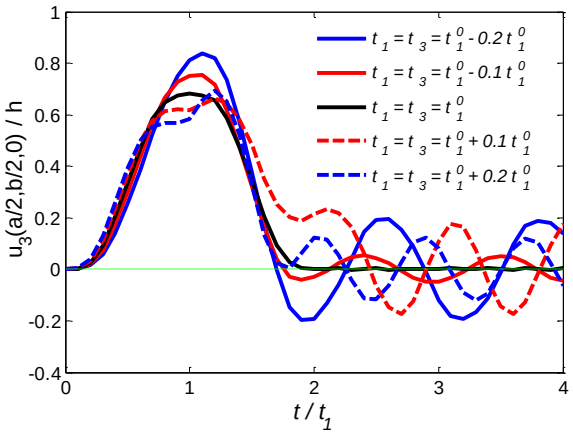


Figure 2.5 Time histories of the centroidal transverse deflection of the CSP for different rise and fall times

Results displayed in Figure 2.7 evince that the plate vibration attenuates even when the triangular load with $t_1 = t_3 = t_1^0$ is applied on a part of the plate top surface that is not

necessarily symmetrically located around the plate's centroid. The time history of the average acceleration of all nodes of the plate is presented in Figure 2.8.

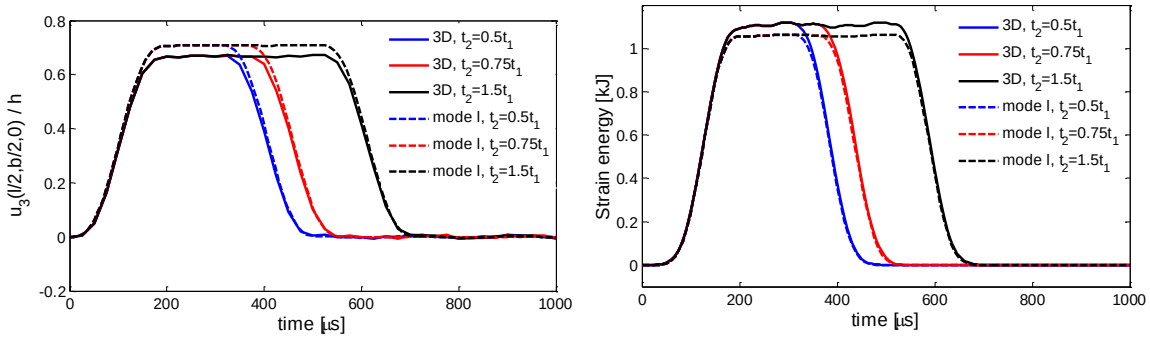


Figure 2.6 Time histories of the centroidal transverse displacement (left), and the strain energy of the plate under different trapezoidal loads obtained from the 3-D and the mode 1 deformations (right)

1.3. Clamped Circular Plate (CCP)

For a 5-mm thick clamped circular plate of radius 50 mm, the converged fundamental frequency computed with the FE mesh of 98,275 8-node elements equals 2.97 kHz. With the uniformly distributed normal traction on plate's top surface that varies in time as a triangular pulse with the rise and fall times of $2\pi/2975 = 336.7$ μs, the transient responses of the plate from the solution of the 3-D and the mode 1 and mode 2 deformations are depicted in Figure 2.9. It is clear that plate's vibrations attenuate upon removal of the load, and there is negligible strain energy associated with deformations corresponding to mode 2 vibrations.

It is clear that that the plate acceleration does not become zero subsequent to the load removal. However, the maximum amplitude of the acceleration is very small as compared to that with the load on the plate's top surface.

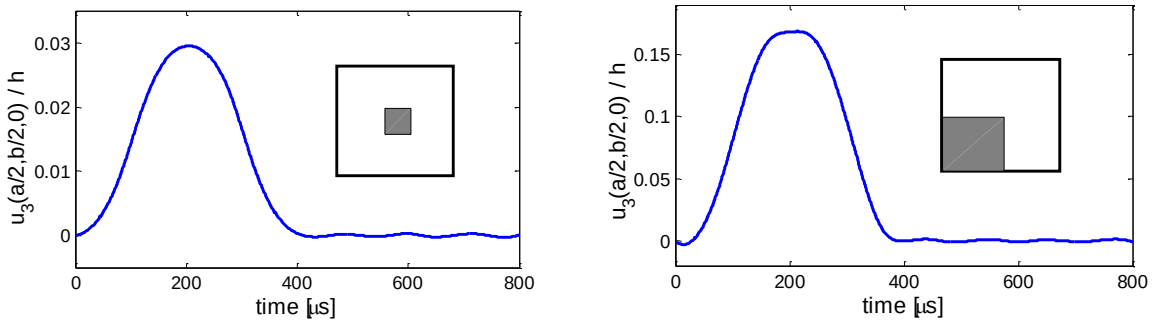


Figure 2.7. Time histories of the centroidal transverse deflection of the CSP with the normal traction applied on two different areas, represented by the shaded region in the insets, of the top surface of the plate

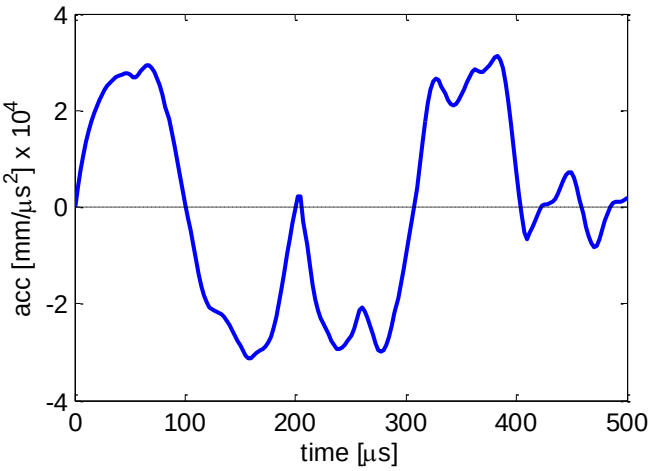


Figure 2.8. Time history of the average acceleration of the plate

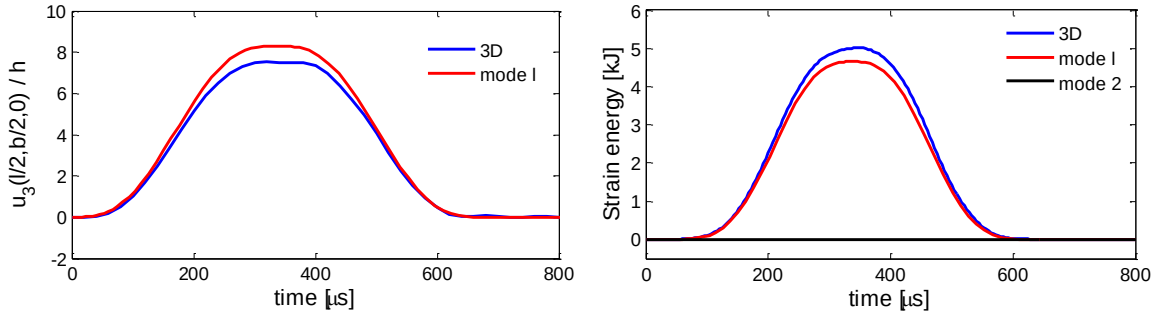


Figure **2.9** Time histories of the centroidal transverse displacement (left), and the total strain energy of the CSP (right) under a triangular time pulse and uniform normal tractions applied on the plate top surface

1.4. Clamped 4-layered Laminated Composite (CLP)

We study transient deformations of a clamped 100 mm x 100 mm four-layered laminated plate with each layer 2.5 mm thick and fibers oriented at 0^0, 15^0, 30^0 and 45^0 with respect to the global *x*-axis in the layers starting from the bottom to the top layer. The layer material is assumed to be transversely isotropic with the fiber direction as the axis of transverse isotropy and E_L = 172.4 GPa, $E_T = E_L/25$, $G_{LT} = E_T/2$, $G_{TT} = E_T/5$, $\nu_{LT} = \nu_{TT} = 0.25$, $\nu = 2.5$, $\rho = 2.5$ g/cc. Here subscripts *L* and *T* denote, respectively, the fiber direction and a direction perpendicular to the fiber and *G* the shear modulus. Each layer is discretized using 80 x 80 x 2 uniform 8-node brick elements. The converged fundamental frequency of the plate equals 4.02 *k*Hz. For a spatially uniform pressure with no dwell time and equal rise and fall times t_1, t_3 of 248.2 μs applied to the plate top surface, time histories of the transverse deflections of the centroid of the plate and of the strain energies obtained from the 3-D, mode 1 and mode 2 deformations are presented in Figure 2.10. It is evident that the

vibration of the plate attenuates once the load is removed, the plate strain energy in the 2nd bending mode is negligible and that in the first mode nearly equals the strain energy found from the solution of the 3-D deformations.

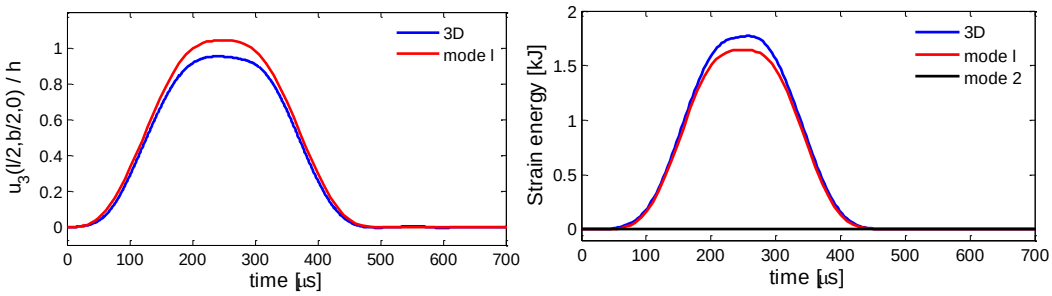


Figure 2.10 Time histories of the centroidal transverse deflection (left), and the total strain energy (right) of the CLP for a triangular time variation of the uniform normal traction from 3D and the mode 1 and mode 2 deformations

1.5. Clamped Functionally graded plate (CFGP)

For analyzing the response of a non-homogeneous plate, we consider a rectangular (200 mm x 100 mm x 10 mm) CFGP for which Young's modulus in the thickness (or *z*-) direction is given by

$$E(z) = 100\left(1+\frac{z}{2h}\right) GPa \tag{2.10}$$

where *z* varies from $-h/2$ to $h/2$. The mass density and the Poisson ratio are assumed to be constants and, respectively, equal 2.5 g/cc and 0.25. The plate is divided into 10 layers of equal thickness with each layer made of a homogeneous material having *E* deduced from Eq. (10) at its mid-point. Batra and Jin [26], amongst others, have shown that dividing an

inhomogeneous plate into 10 layers of homogeneous materials simulates well its frequencies. Each layer is meshed using 200 x 100 x 1 uniform 8-node brick elements. The fundamental frequency of the rectangular plate equals 6.80 kHz. With the spatially uniform peak pressure of 102.76 GPa with t_1, t_3 of 147μs and t_2 = 0, time histories of the plate centroidal transverse displacement and the plate strain energy computed using the 3-D, the mode 1 and the mode 2 deformations are presented in Figure 2.11. As for the other problems studied, the displacements essentially become zero and stay at zero upon the load removal, and the strain corresponding to mode 2 deformations is negligible as compared to that of the mode 1 deformations.

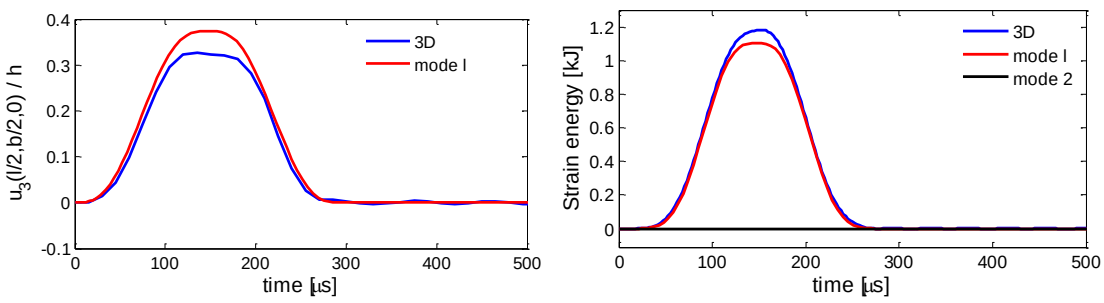


Figure 2.11 Time history of the centroidal transverse deflection (left), and the total strain energy of the rectangular CFGP (right) under uniform triangular time pulse obtained from the 3-D, mode 1 and mode 2 deformations

The time-history of the acceleration the non-homogenous plate averaged over domain $x \in [0.1l, 0.9l], y \in [0.1b, 0.9b]$ to eliminate effects of clamped boundaries is exhibited in Figure 2.12. It is evident that after the load removal at 294 μs, the plate acceleration sharply decreases and remains small. Thus, even though deformation in higher vibration modes

make negligible contributions to the strain energy of the system, they do not cease to exist once the load is removed.

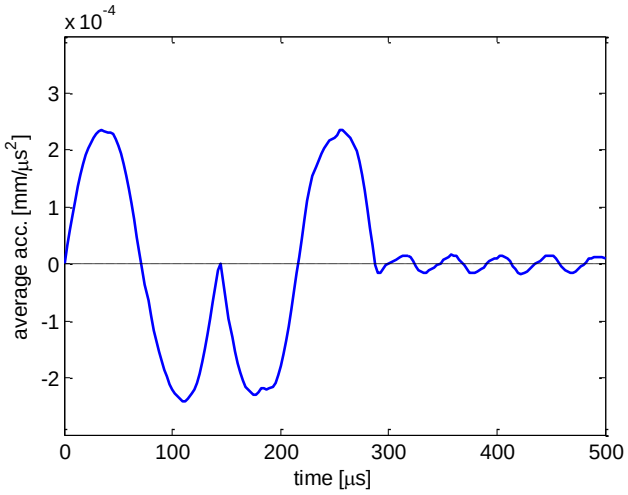


Figure 2.12 Average acceleration of the interior nodes in the domain $x \in [0.1l, 0.9l]$, $y \in [0.1b, 0.9b]$ of the clamped non-homogeneous plate from the 3D FEM solution

2.4 Conclusions

We note that for a linear spring-mass system subjected to a time-dependent trapezoidal force with linearly increasing and affinely decreasing force of duration equal to integer multiples of the time-period of the fundamental frequency of the system, the mass comes as soon as the applied force ceases to act. This motivated us to investigate if a similar result holds for linearly elastic continuous structures with the time-period of the first bending mode of vibration playing the role of the time-period of the linear spring-mass system. By studying free and forced vibration of several plates including laminates, we have found that indeed such a results holds for linearly elastic continuous structures. For all problems

studied, we found that nearly all of the strain energy of deformation is due to deformations in the fundamental bending mode of vibration. Contributions from higher vibration modes, although insignificant to the total strain energy of the plate, manifests in the small amplitude acceleration response as indicated by non-zero average acceleration of the plate after the load removal.

2.5 Acknowledgements:

This work was partially supported by the Office of Naval Research Grant N00014-16-1-2309 to Virginia Polytechnic Institute and State University with Dr. Y. D. S. Rajapakse as the Program Manager. Views expressed in the article are those of the authors and neither of the funding agency nor of authors' institutions.

3 Free and Forced Vibrations of Monolithic and Composite rectangular Plates with Interior Constrained Points

This chapter constitutes the paper published in the ASME Journal of Vibration and Acoustics, Vol. 141, Art. No. 011018, 2019; https:// doi:10.1115/1.4041216

Abstract

The restriction of deformations to a subregion of a system undergoing either free or forced vibration due to an irregularity or discontinuity in it is called mode localization. Here, we study mode localization in free and forced vibration of monolithic and unidirectional fiber-reinforced rectangular linearly elastic plates with edges either simply supported or clamped by using a third order shear and normal deformable plate theory (TSNDT) with points on either one or two normals to the plate mid-surface constrained from translating in all three directions. The plates studied are symmetric about their mid-surfaces. The in-house developed software based on the finite element method (FEM) is first verified by comparing predictions from it with either the literature results or those computed by using the linear theory of elasticity and the commercial FE software ABAQUS. New results include: (i) the localization of both in-plane and out-of-plane modes of vibration, (ii) increase in the mode localization intensity with an increase in the length/width ratio of a rectangular plate, (iii) change in the mode localization characteristics with the fiber orientation angle in unidirectional fiber reinforced laminae, (iv) mode localization due to points on two normals constrained, and (iv) the exchange of energy during forced harmonic vibrations between two regions separated by the line of fixed points that results in a beats-like phenomenon in a sub-region of the plate.

Constraining translational motion of internal points can be used to design a structure with vibrations limited to its small sub-region, and harvesting energy of vibrations of the sub-region.

3.1 Introduction

Discontinuities and irregularities in a physical system may cause anomalies in its free and forced vibrations. Anderson [9] observed that irregularities in electrons distribution in different lattice structures vary their vibration characteristics and the material conductivity; this phenomenon is called *Anderson's localization* [10]. Hodges [10] extended the vibration localization phenomenon to continuous periodic structures. Subsequently, numerous works have illustrated the mode localization phenomenon in continuous structures that include periodic structures with cyclic symmetry [11-13], multi-spanned beams [14], and irregular structures [15-17].

Early works on mode localization in continuous bodies were mostly restricted to one-dimensional (1-D) problems possibly because of difficulties in computing eigen-modes [16]. Hodges and Woodhouse [15], based on Herbert and Jones's work [17], used a statistical perturbation method to study localization phenomenon in a string by inducing an irregularity with a slidable mass. They found good agreement between their analytical and experimental results. Depending on the magnitude of internal coupling of the structure, they divided the localization phenomenon into weak and strong. Using a mathematical model closely related to Kirman and Pendry's [27] solid state physics model, Pierre [16] delineated factors for the weak and the strong localization phenomena.

Pierre and Plaut [28] used the perturbation approach to study the mode localization phenomenon in multi-span hinged beams. Due to mathematical similarities between the free vibration and the elastic buckling problems, one can also observe the mode localization phenomenon in elastic buckling of thin structures. For example, Nayfeh and Hawwa [29] used principles of mode localization to control buckling of structures, and Paik et al. [30] characterized buckling localization in composite laminae with constrained interior points. Ibrahim [31] as well as Hodges and Woodhouse [32] have reviewed the literature on the localization phenomenon published till 1987.

Nowacki [33] analytically studied, using a Levy solution, vibration of simply supported (SS) rectangular plates with multiple constrained internal points. Gorman [34] analytically investigated free vibrations of plates clamped only at symmetric points on the diagonals and used a plate theory. Gorman and Singal's [35] experimental findings agreed well with the analytical results of [34].

Bapat et al. [36-37], and Bapat and Suryanarayana [38-39] employed the flexibility function approach to study free vibration of point supported plates. Bapat and Suryanarayana [40] extended it to analyze mode localization in SS rectangular plates having internal constrained points. Lee and Lee [41] adopted the impulse function approach to analytically solve similar problems.

Rao et al. [42], Raju and Amba-Rao [43], and Utjes [44], amongst others, have numerically analyzed the mode localization phenomenon in rectangular plates with point supports by using the finite element method (FEM), whereas Kim and Dickinson [45] and Bhat [46]

used the Rayleigh-Ritz method. Filoche and Mayboroda [47] used the Kirchhoff plate theory and the FEM to show that constraining all points on a normal to the plate midsurface of a rectangular plate induced strong mode localization. Sharma et al. [48] used a first order shear deformation theory (FSDT) and the FEM to show the mode localization phenomenon in composite rectangular plates when both bending and transverse shear deformations are incorporated in the analysis. These studies [30, 31] considered only bending or out-of-plane modes of vibration, and all plate edges clamped. It seems that the localization of in-plane modes of vibration, effects of simply-supported (SS) edges on mode localization, and constraining points on two normals to the plate mid-surface have not been scrutinized. For SS plates, Batra and Aimmanee [18] analytically found these and bending vibration modes by using complete polynomials in the Levy type solutions. Other authors have studied either in-plane (e.g., see Du et al. [49]) or bending (e.g., see Srinivas and Rao [24]) modes of vibration only. Even though one can study mode localization by separately using the bending and the stretching plate theories and then combining the results, the use of a third-order shear and normal deformable plate theory (TSNDT) simultaneously gives both types of modes. It thus does not require studying the problem with two different plate theories. The TSNDT does not require a shear correction factor, frequencies and mode shapes of the first 100 modes of vibration agree well with those computed using the linear elasticity theory (LET), and the in-plane stresses computed from the TSNDT displacements and the constitutive relation agree well with those found using the LET. Furthermore, the transverse stresses computed by using a one-step stress recovery scheme (SRS) are close to those obtained using the LET. For materials with Poisson's ratio close to 0.49, the

transverse normal strains are likely to be of the same order of magnitude as the axial strains, and require plate theories that consider transverse normal strains.

The TSNDT is particularly useful for problems involving inhomogeneous materials with elastic moduli varying along the plate thickness. Vel and Batra [50] showed that many simple plate theories do not predict well stresses at critical locations. As demonstrated by Shah and Batra [51] the TSNDT solution provides reasonably accurate values of stresses everywhere in the plate.

Here we study the mode localization phenomenon in free and forced vibrations of monolithic and unidirectional fiber-reinforced laminated rectangular linearly elastic plates with internal constrained points by using a TSNDT. Christensen and Lo [52], Carrera et al. [53], Vidoli and Batra [6], and Batra and Vidoli [7], amongst others, have proposed higher-order shear and normal deformable plate theories based on Mindlin's classical work [4]. As also observed in [47] and [48] who did not consider transverse normal deformations, with an increase in the length/width ratio for a rectangular plate, the mode localization becomes stronger. One of the new results reported here is the mode localization for in-plane modes of vibration. We show that the first 100 frequencies and strain energies associated with their mass normalized mode shapes computed by using the TSNDT agree well with those found from the LET. Subsequently, we compute results with the TSNDT and study mode localization in both isotropic and composite plates. We also study forced vibrations of internally constrained plates to delineate if vibrations are localized.

For forced harmonic vibrations of an internally constrained plate at a frequency close to that of the mode for which vibrations are localized in one of the two regions, the response strongly depends upon which region exhibited mode localization. For example, we observe beats like phenomenon in the shorter region of the plate possibly due to the energy transfer between the two regions. This is similar to the steady state response observed in [54] and [55] in structures composed of vibration absorbing dampers. Spetzer el al. [56] have used this principle to design ultra-sensitive mass sensors using linked cantilever beams. Thus, the mode localization phenomenon can be both beneficial and harmful based on design requirements. It serves as a tool for designing structures with desired vibration characteristics.

3.2 Problem Formulation

We use both the LET and a TSNDT to analyze free and forced infinitesimal vibrations of linearly elastic rectangular plates with and without constraining interior points on either one or two normals to the plate mid-surface. A schematic sketch of the problem studied is exhibited in Figure 1 wherein rectangular Cartesian coordinates, used to describe plate's deformations, are also depicted.

3.2.1 The Linear Elasticity Theory

For the LET, equations governing deformations of a plate in the absence of body forces are

$$\rho \ddot{u}_i = \sigma_{ij,j} \tag{3.1}$$

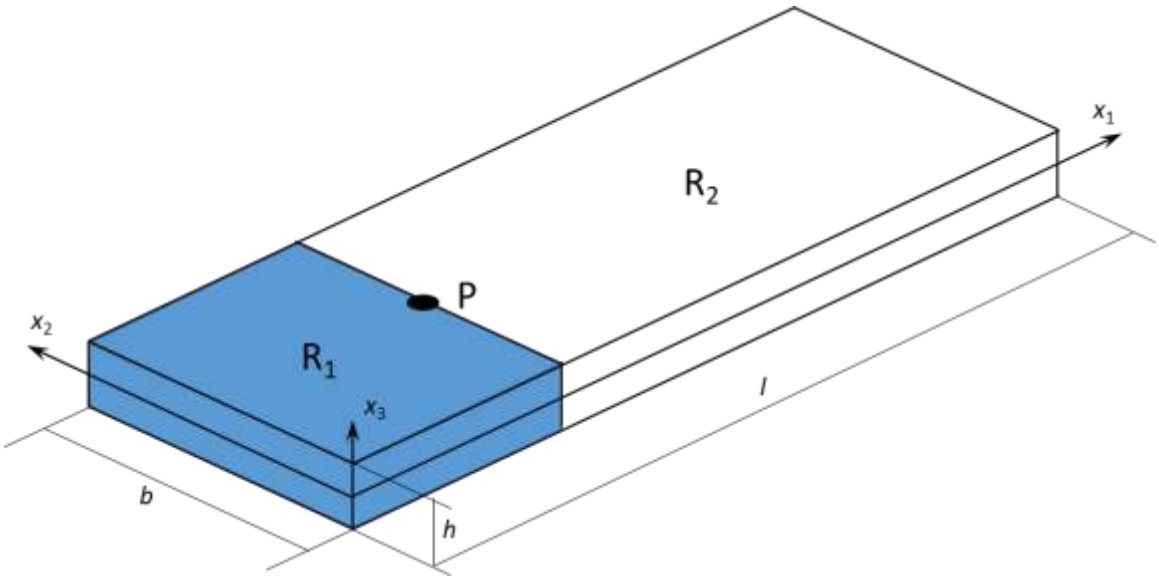


Figure 3.1 Schematic representation of a rectangular plate with the rectangular Cartesian coordinate axes. Points on the normal through the point P may be constrained from translating in all three directions

where u_i (i = 1, 2, 3) is the displacement along the x_i- axis, $\sigma_{ij,j} = \partial\sigma_{ij} / \partial x_j$, σ_{ij} is the stress tensor, $\ddot{u}_i = \partial^2 u_i / \partial t^2$, t is the time, and a repeated index implies summation over the range of the index. Eq. (3.1) is supplemented with Hooke's law

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{3.2}$$

strain-displacement relations

$$\varepsilon_{ij} = \left(u_{i,j} + u_{j,i}\right) / 2 \tag{3.3}$$

and initial/boundary conditions

$$
\begin{aligned}
&u_i\left(x_1,x_2,x_3,0\right)=u_i^0\left(x_1,x_2,x_3\right),\ \dot{u}_i\left(x_1,x_2,x_3,0\right)=\dot{u}_i^0\left(x_1,x_2,x_3\right)\\
&\sigma_{ij}n_j=t_i\left(x_1,x_2,x_3,t\right)\ \text{on}\ \Gamma_t,\ u_i\left(x_1,x_2,x_3,t\right)=u_i^{bc}\left(x_1,x_2,x_3,t\right)\ \text{on}\ \Gamma_u\\
&\Gamma_t\cup\Gamma_u=\partial\Omega,\ \Gamma_t\cap\Gamma_u=\phi
\end{aligned}
\tag{3.4}
$$

Here Γ_t and Γ_u are, respectively, parts of the boundary where surface tractions and displacements are prescribed as $t_i\left(x_1,x_2,x_3,t\right)$ and $u_i^{bc}\left(x_1,x_2,x_3,t\right)$, $\boldsymbol{n}$ is an unit outward normal to Γ_t, $u_i^0\left(x_1,x_2,x_3\right)$ is the initial displacement field, $\dot{u}_i^0\left(x_1,x_2,x_3\right)$ is the initial velocity field and Ω is the region occupied by the plate. We note that linearly independent components of $u_i\left(x_1,x_2,x_3,t\right)$ and $\sigma_{ij}n_j$ can be prescribed at a point on the plate surfaces. Substitution from Eqs. (3.2) and (3.3) into Eq. (3.1) results in three coupled linear partial differential equations for finding u_i. At an interior constrained point $\left(\bar{x}_1,\bar{x}_2,\bar{x}_3\right)$ we set u_i ($\bar{x}_1,\bar{x}_2,\bar{x}_3,t$) = 0.

3.2.2 Third-Order Shear and Normal Deformable Plate Theory (TSNDT)

In the TSNDT, the displacement field is approximated as

$$
u_i\left(x_1,x_2,x_3,t\right)=\sum_{j=0}^{3}\left(x_3\right)^j u_{ij}\left(x_1,x_2,0,t\right),\ \ i=1,2,3 \tag{3.5}
$$

where the 12 functions, u_{ij}, are defined on the plate reference surface, here taken to be the plate mid-surface, x_3 = 0. One may interpret u_{ij} as the j^{th} order partial derivative of $u_i\left(x_1,x_2,x_3,t\right)$ with respect to x_3 evaluated at x_3 = 0. Alternatively, for $j=1,2,3$ they can

be interpreted as directors proposed by the Cosserat brothers [57]. Units of u_{ij} are length/(length)^j. Substituting for displacements from Eq. (3.5) into Eq. (3.1), we get

$$\rho\left(\ddot{u}_{i0} + x_3\ddot{u}_{i1} + x_3{}^2\ddot{u}_{i2} + x_3{}^3\ddot{u}_{i3}\right) = \sigma_{ij,j} \tag{3.6}$$

Multiplying both sides of Eq. (3.6) by $\left(x_3\right)^k$ (k = 0, 1, 2, 3) and integrating resulting equations with respect to x_3 over the plate thickness, we get

$$\sum_{j=0}^{3} A_{j+1}^{(k)}\ddot{u}_{ij} = \sum_{\alpha=1}^{2} M_{i\alpha,\alpha}^{(k)} + \left(\left(x_3\right)^k \sigma_{i3}\right)\Big|_{-h/2}^{h/2} - kM_{i3}^{(k-1)},\ i=1,2,3,\ k=0,1,2,3,\ \alpha=1,2 \tag{3.7}$$

where

$$A_j^{(k)} = \int_{-h/2}^{h/2} \rho(x_3)^{k+j} dx_3\ ,\ M_{ij}^{(k)} = \int_{-h/2}^{h/2} (x_3)^k \sigma_{ij} dx_3,\quad k=0,1,2,3 \tag{3.8}$$

The element $A_j^{(k)}$ associated with $\ddot{u}_{ij}$ appears in the j^{th} row and k^{th} column of the inertia matrix **A**, and $M_{ij}^{(k)}$ is the k^{th} order moment of σ_{ij} about the plate mid-surface. $M_{ij}^{(0)}$ and $M_{ij}^{(1)}$, respectively, are the usual force per unit length and the moment per unit length; $M_{ij}^{(2)}$ and $M_{ij}^{(3)}$ are higher-order moments. Substitution from Eqs. (3.5), (3.2) and (3.3) into Eq. (3.8) gives expressions for $M_{ij}^{(k)}$ in terms of u_{ij} . These expressions when substituted in Eq. (3.7) yield 12 coupled linear partial differential equations for the 12 unknown functions, u_{ij}.

Boundary conditions for the LET and the TSNDT at a clamped and a SS edge are listed below.

SS edge $x_1 = 0$:

$u_2, u_3 = 0, \sigma_{11} = 0$ for the LET; $u_{2i}, u_{3i} = 0, M_{11}^{(i)} = 0,\ i = 0$ to 3 for the TSNDT

Clamped edge $x_1 = 0$: (3.9)

$u_1, u_2, u_3 = 0$ for the LET; $u_{1i}, u_{2i}, u_{3i} = 0,\ i = 0$ to 3 for the TSNDT

Boundary conditions (3.9) for a SS edge are the same as those employed by Srinivas and Rao [58] in their analytical solution of a linearly elastic problem, and are used when seeking a Levy type solution. Analytical solutions for static and wave propagation problems for arbitrary boundary conditions are given in [59-61].

3.3 Numerical Solution of the Problem

For the above-stated two initial-boundary-value problems (IVBPs), we first derive weak formulations by employing the Galerkin method; e.g., see [23]. We numerically solve the resulting equations by the FEM with the in-house developed software for the TSNDT equations, and the commercial software, ABAQUS, for the LET equations. We note that in each case, the resulting equations are expressed in matrix form as

$$\mathbf{M}\ddot{\mathbf{d}} + \mathbf{K}\mathbf{d} = \mathbf{F} \tag{3.10}$$

where **M** and **K**, respectively, represent the mass and the stiffness matrices, and **d** is the vector of nodal unknowns, 3 for a node in the LET and 12 for a node in the TSNDT. Nodes are distributed throughout the 3-D plate domain for the LET, and only on the 2-D plate reference surface for the TSNDT. We employ 8-node brick elements with 2 x 2 x 2 integration points in an element for the LET, and 4-node quadrilateral elements for the

TSNDT with 2 x 2 integration points in an element. The mass and the stiffness matrices, and the load vector for the TSNDT are evaluated by employing 7 uniformly spaced integration points in the thickness direction. The total number of degrees of freedom for the TSNDT is considerably less than that for the LET.

3.3.1 Free Vibrations

For the free vibration problem, $\mathbf{F} = 0$, and $\mathbf{d}(t) = \mathbf{D}e^{i\lambda t}$, no initial conditions are needed, and boundary conditions (3.4) are such that no work is done by external forces. Eq. (3.10) reduces to the following eigenvalue problem.

$$[\mathbf{M} - \lambda^2 \mathbf{K}]\mathbf{D} = \mathbf{0} \tag{3.11}$$

The cyclic frequencies f_i (in *Hz*) of the plate is given by

$$f_i = \lambda_i / 2\pi \tag{3.12}$$

The number of frequencies equals the number of unconstrained degrees of freedom or the dimensionality of the vector, **d**, minus the number of constraints including those on the plate edges.

3.3.2 Forced Vibrations

For the transient analysis, we use the conditionally stable, central-difference time integration scheme, and a lumped mass matrix in ABAQUS for the LET equations and the consistent mass matrix for the TSNDT equations. Because of the generalized displacements in the TSNDT, terms in the mass matrix have different dimensions. Thus,

one cannot employ the row sum technique of lumping the mass matrix. By non-dimensionalizing variables, one could potentially use a lumped mass matrix. The time integration scheme is stable when the time step size, Δt, satisfies the condition,

$$\Delta t \le \Delta t_{critical} = 2/\lambda_{\max} \tag{3.13}$$

where $\lambda_{\max}\,(rad/s)$ is the largest natural frequency of the system. The eigenvector for each frequency is normalized with respect to the mass matrix in both the LET and the TSNDT. The computation of stresses for the LET is straightforward. For determining in-plane stresses in the TSNDT, we find strains with the TSNDT displacements and then stresses from the constitutive relation. We use a one-step SRS to compute the transverse (out-of-plane) stresses. That is, we integrate with respect to x_3 the LET equations of motion starting from the bottom surface. For α = 1 and 2,

$$\sigma_{\alpha 3}\big|_{x_3=z} = \sigma_{\alpha 3}\big|_{x_3=-h/2} + \int_{x_3=-h/2}^{z} \left(\rho \ddot{u}_\alpha - \frac{\partial \sigma_{\alpha\beta}}{\partial x_\beta} \right) dx_3 \tag{3.14}$$

where $z = x_3$. For evaluating the integrand in Eq. (3.14) at a given point, we first find the in-plane stresses at centroids of 9 elements surrounding the point, fit a complete quadratic polynomial to each component of the stress by the least squares method, differentiate the polynomial, and then substitute in it coordinates of the point. Having found stresses σ_{13} and σ_{23}, the equation of motion in the x_3- direction is integrated with respect to x_3 to find σ_{33} that require knowing $\sigma_{3\alpha,\alpha}$ for different values of x_3.

3.4 Example Problems

3.4.1 Verification of the TSNDT Software for Free Vibrations of Rectangular Plates

Comparison of the first 100 Frequencies

We first verify the in-house developed software by comparing the 5 lowest fundamental frequencies of a linearly elastic, homogeneous and isotropic 100 mm x 100 mm x 10 mm plate having Young's modulus, E = 200 GPa, Poisson's ratio, ν = 0.3, and mass density, ρ = 7.2 g/cc with their analytical values [24], and from higher order plate theories [18, 25]. We successively refined the FE mesh of uniform elements for the TSNDT solution, and found that the difference in the first 10 frequencies using 40 x 40 (20,172 degrees of freedom (DoFs)) and 30 x 30 elements (11,532 DoFs) was less than 1 %. In Table 3.1, we have listed the presently computed first five converged frequencies using 40 x 40 uniform 4-node elements and those found by other investigators. It is clear that the TSNDT predicted first five natural frequencies differ at most by of 0.85% from their analytical values.

Table 3.1 First five non-dimensional frequencies, $\omega_n = \omega h\sqrt{\rho/E}$, of the 100 mm x 100 mm x 10 mm SS plate (* denotes in-plane mode of vibration; ω equals λ used earlier).

Mode	TSNDT	Srinivas and Rao	Qian et al. [25]	Batra and
1	0.0581	0.0578	0.0578	0.0578
2	0.1393	0.1381	0.1391	0.1391
3	0.1393	0.1381	0.1391	0.1391

4	0.1949*	-	0.1948*	0.1949*
5	0.1949*	-	0.1948*	0.1949*

We note that modes 4 and 5 marked with an * represent in-plane modes of vibration with $u_3 = 0$ that were absent from the analytical solution of Srinivas and Rao [24] since they considered only bending deformations.

When studying mode-localization, we use first 100 frequencies. Accordingly, we now compare these computed from the TSNDT with those using the LET and the commercial FE software, ABAQUS. For the 80 mm x 20 mm x 2 mm (aspect ratio = length/thickness = 40) SS and clamped plates ($E = 25$ GPa, $\nu = 0.25$, $\rho = 5$ g/cc), we have exhibited in Figure 3.2 the first 100 frequencies found from the two theories. When computing results using ABAQUS, we used 320 x 80 x 8 uniform 8-node C3D8 elements resulting in 234,009 DoFs. The use of C3DR elements in ABAQUS did not give accurate frequencies of modes greater than 50. It is clear that results from the two approaches differ by less than 4 % for both the SS and the clamped plates. A similar exercise for the 80 mm x 20 mm x 4 mm (aspect ratio = 20) plate showed, respectively, the maximum difference of 2.3% and 3.8 % for the SS and the clamped edges. Both the LET and the TSNDT gave several in-plane modes of vibrations.

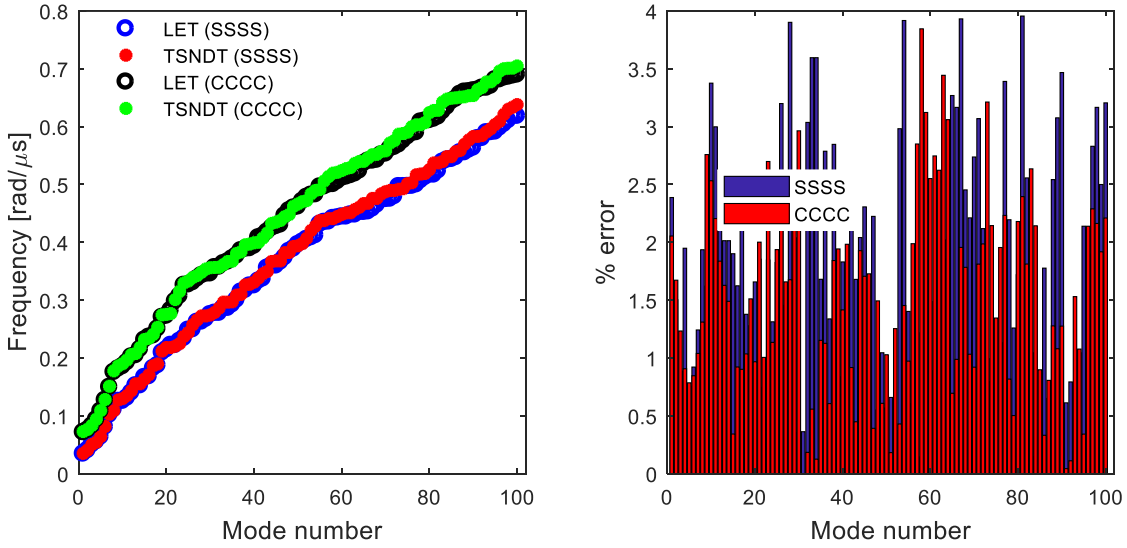


Figure 3.2 First 100 frequencies, in $\text{rad}/\mu s$, from the TSNDT and the 3D LET (left), and the relative difference between them (right) for the 80 x 20 x 2 mm SS and clamped plates

Comparison of strain energies associated with the first 100 modes of vibration

For mass normalized displacements for a mode shape, $\mathbf{D}^T\mathbf{M}\mathbf{D} = 1$, and $\mathbf{D}^T\mathbf{K}\mathbf{D}/2$ equals the strain energy of a linearly elastic body. Rayleigh's Theorem (or pre-multiplying Eq. (3.11) by $\mathbf{D}^T$) gives

$$\lambda^2 = \mathbf{D}^T\mathbf{K}\mathbf{D} / \mathbf{D}^T\mathbf{M}\mathbf{D} \tag{3.15}$$

Thus, the strain energy of deformations associated with a mode shape equals one-half of the square of the frequency (in radians/s) of the mode shape. Because of the dimensional units used here, the strain energy in J equals 0.5 x (frequency in rad/μs)^2 ; we call this as the *TSNDT (modal)* energy. One can also compute the strain energy, U , of a mode as

$$U = \frac{1}{2}\int_V \sigma_{ij}\varepsilon_{ij} dv \tag{3.16}$$

where stresses and strains are calculated from the mass normalized eigen-vectors; we call this as the *TSNDT (direct)* energy. We have exhibited in Figure 3.3 strain energies associated with the first 100 modes from the two theories for the 80 mm x 20 mm x 2 mm plate for the SS and the clamped plates. These results evince that the TSNDT and the LET give almost identical strain energies up to the first 50 modes and the TSNDT predicts slightly higher strain energies for the subsequent 50 modes.

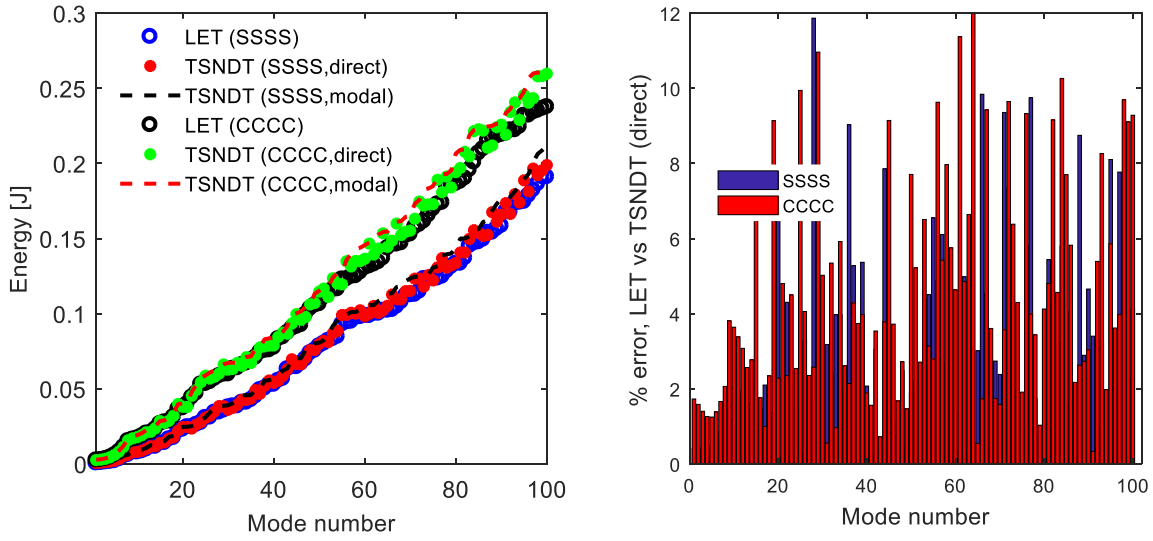


Figure 3.3 Total strain energy, in J, from the TSNDT and the 3D LET (left), and the relative error between them (right) for the 80 x 20 x 2 mm SS and clamped plates

3.4.2 Mode Localization in Clamped and SS Plates with Interior Constrained Points

Plates Made of Monolithic and Isotropic Materials

Following [47] we normalize rectangular plate's areal dimensions to $\sqrt{e}$ x $1/\sqrt{e}$ to have unit surface area, and call *e* the eccentricity (it equals the length/width). We study its free vibrations with all points on the line, $x_1 = \sqrt{e}/5$, $x_2 = 1/2\sqrt{e}$, (i.e., normal to the plate mid-surface through the point P ($\sqrt{e}/5, 1/2\sqrt{e}$, 0) constrained or equivalently, restrained from translating in all three directions. In the TSNDT, it is accomplished by setting $u_{1i}, u_{2i}, u_{3i} = 0,\ i = 0$ to 3 (see Eq. (2.9)). As shown in Figure 3.1, the plate midsurface to the left (right) of the point P is denoted by R_1 (R_2). For plates with e = 1, 4, 16 and 25, we computed results using 40 x 40, 80 x 20, 160 x 10 and 200 x 8 uniform 4-node elements using the TSNDT.

Clamped edges

We have displayed in Figure 3.4 frequencies of the first 100 vibration modes of a clamped plate of e = 16 with and without the internal constrained points. As in [47] and [48] where the Kirchhoff theory and the FSDT were used, respectively, in the TSNDT imposing an internal constraint does not affect the first 100 frequencies of a plate. However, constraining internal points strongly affects shapes of modes 2, 4, 5, 10 and 20, depicted in Figure 3.5 by using (X, Y, Z) = (x_1, x_2, x_3).We note that for a plate with the internal constrained points vibrations of either region R_1 or of region R_2 are miniscule. Although the modes 5 is essentially unaffected by constraining the internal points, mode shapes of the vibrating region are quite different.

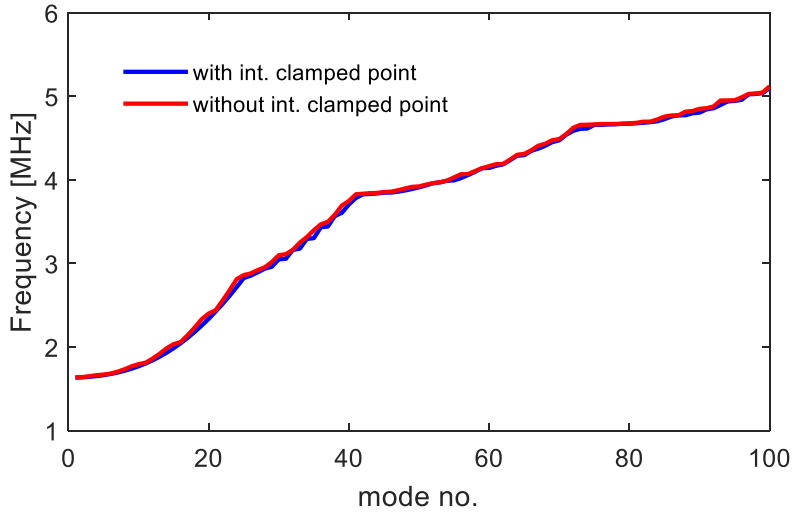


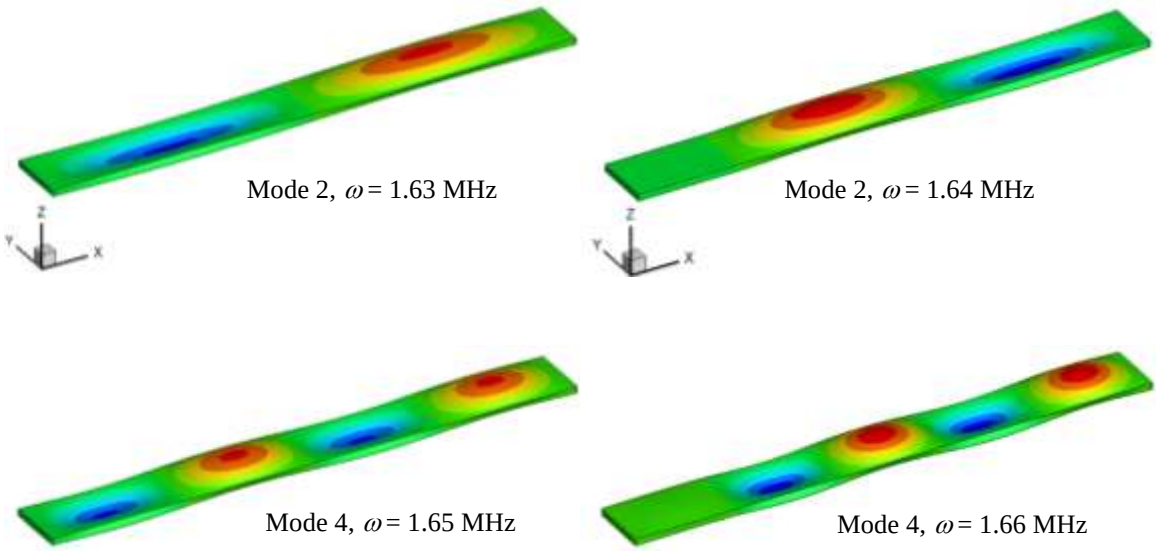
Figure 3.4 Frequencies of the first 100 modes of vibration of the e = 16 clamped plate with and without internal points constrained

Following [47] and [48], we quantify mode localization by parameter, β_1, defined by

$$\beta_1 = \frac{\sum_{i=1}^{n} [u]_i^T [k_{el}]_i [u]_i}{\sum_{i=1}^{N_{el}} [u]_i^T [k_{el}]_i [u]_i} \tag{3.17}$$

where n is the number of elements in the plate region R_1, N_{el} the total number of elements in the plate, $[k_{el}]$ the element stiffness matrix, and $\boldsymbol{u}$ the vector of nodal displacements in the mass normalized eigen-vector for the i^{th} mode. Thus β_1 equals the ratio of the total strain energy of the region R_1 to that of the entire plate. The value of β_1 near 0 implies that most of the plate deformation in region R_1 is annulled.

For the first 100 modes of vibration of plates with e = 1, 4, 16 and 25, values of β_1 for each mode and the total number of modes for a given value of β_1 are presented in Figure 3.6. These results suggest that the value of β_1 strongly depends upon the eccentricity, e, and for e = 1, $\beta_1 <$ 0.27 for 97 out of the first 100 modes of vibration implying that there is no noticeable mode localization since energies in regions R_1 and R_2 are proportional to their volumes. However, for e = 25, for the first 52 modes of vibration, points in either region R_1 or region R_2 are nearly undeformed since for them, β_1 is either near 0 or 1. For e = 1, 4, 16 and 25, the number of modes for which β_1 is either essentially 0 or 1, respectively, equals 0, 18, 41 and 52. Thus, as concluded in [47] and [48], the total number of modes with nearly null deformations increase with an increase in e.



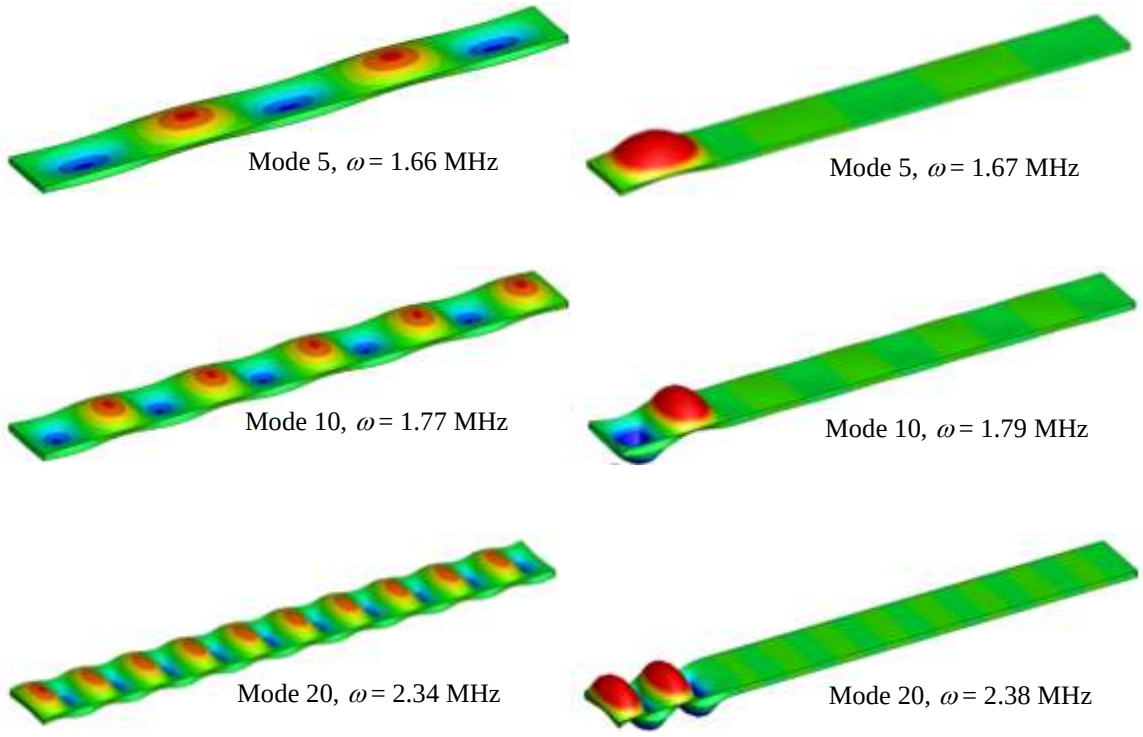


Figure 3.5 Mode shapes for free vibration of the clamped plate of e = 16 with (right) and without (left) internal constrained points. The red and the blue colors, respectively, represent magnitudes of the maximum positive and the maximum negative transverse displacement

Simply supported (SS) edges

In order to decipher whether or not an in-plane mode of vibration localizes in either region R_1 or R_2, we study free vibrations of the rectangular plate of e = 16 with and without constraining interior points. Fringe plots of the total displacement magnitude for 9 modes, not necessarily consecutive, are presented in Figure 3.7 with top views of the plate for modes 1, 3 and 4, and isometric views for other modes. Values of β_1 and the corresponding histogram is exhibited in Figure 3.8.

There is essentially no localization of deformation for the plate without any internal point constrained since $\beta_1 \cong 0.2$ for most modes. However, for the plate with internally constrained points, 37 and 5 modes, respectively, have values of β_1 close to either 0 or 1 signifying their localization in one of the two regions. The remaining 58 modes having in

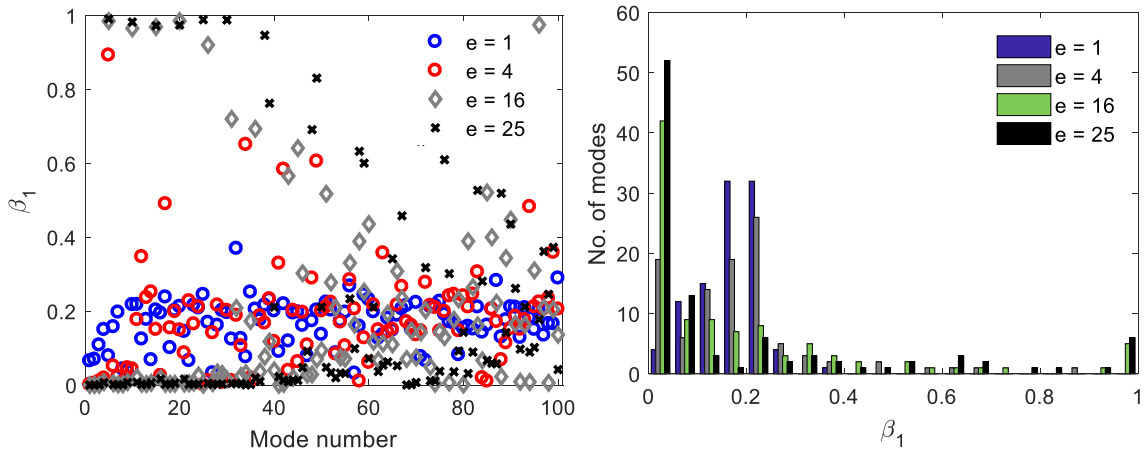
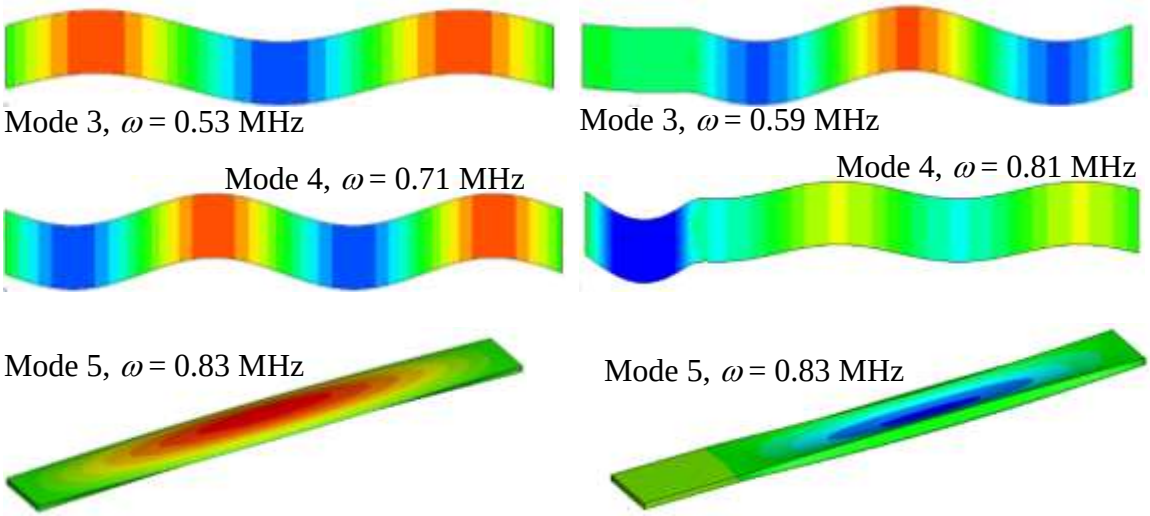


Figure 3.6. Mode localization parameter, β_1, for the first 100 modes of vibration of a clamped plate with constrained internal points (left) and distribution of modes over different values of the ratio β_1 for the first 100 vibration modes in windows of 0.05 (right)



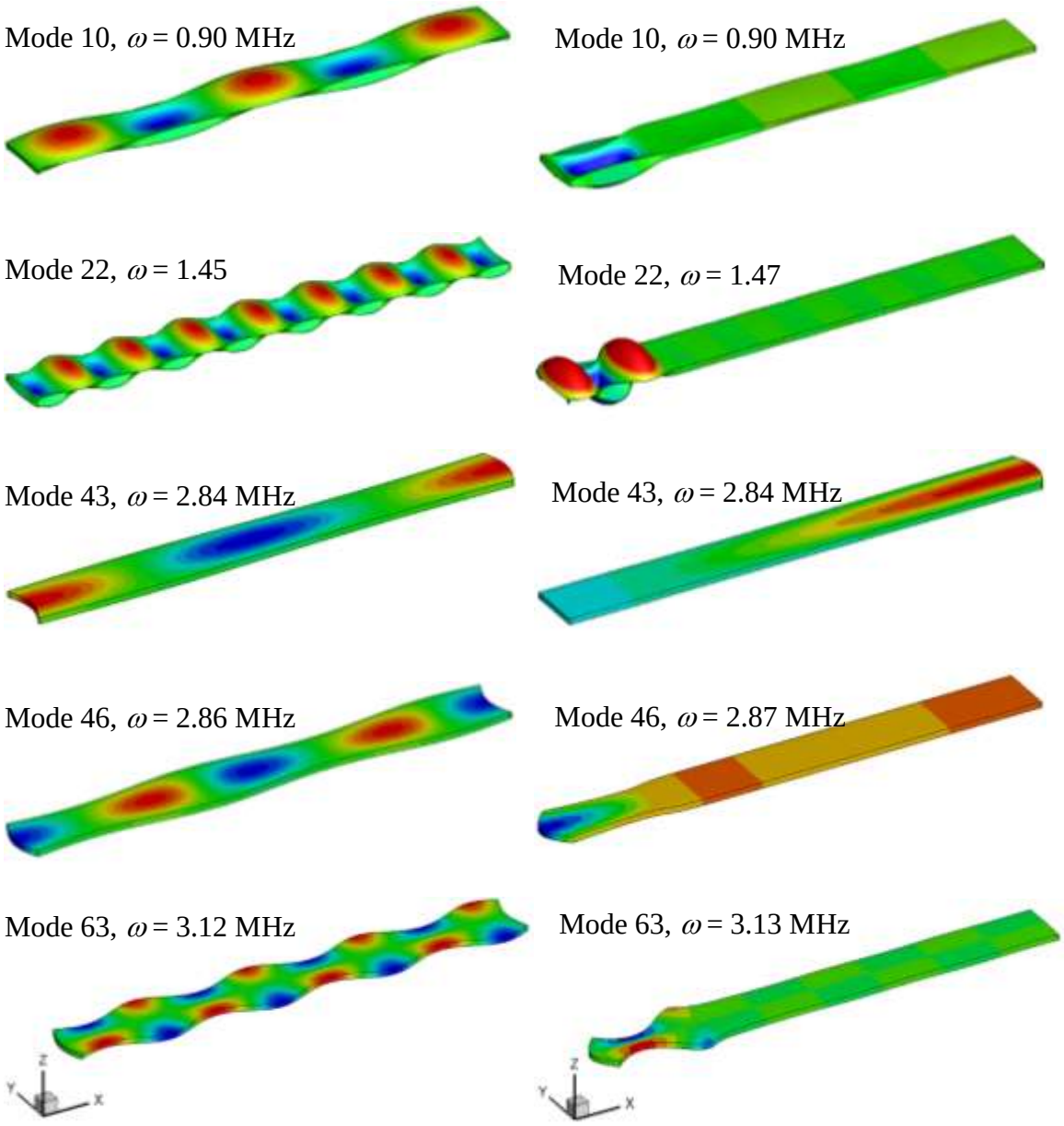


Figure 3.7 Fringe plots of the magnitude of the total displacement for different mode shapes of the SS plate of e = 16 with (right) and without (left) internal constrained points values different from 0, 0.2 and 1 are partially localized. We note that the mode localization

one region of an in-plane mode of vibration does not completely kill vibration of points in the other region as occurs for the out-of-plane (or bending) vibration modes. For example,

in the deformed shape of mode 4 displayed in Figure 3.7, in spite of the localization of the mode in region R_1, points in region R_2 significantly deform.

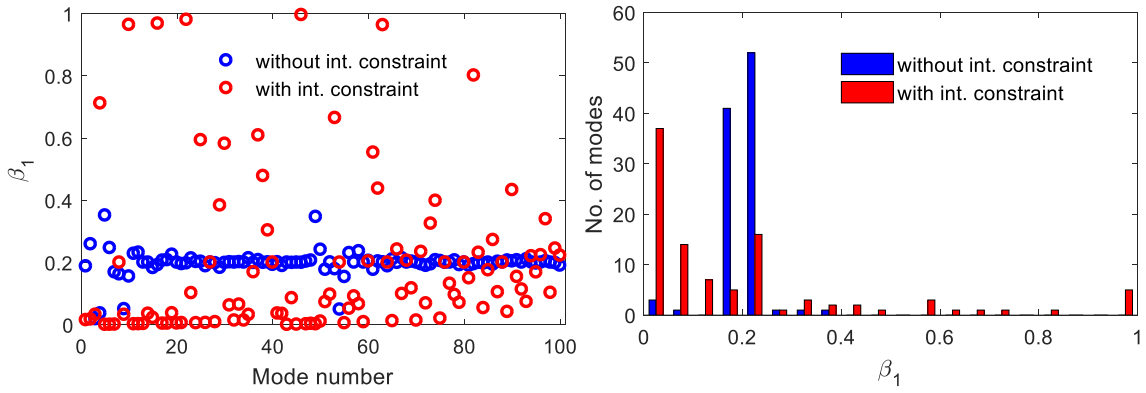


Figure 3.8 Values of β_1 for different modes (top) and the histogram of β_1 for the first 100 modes of free vibration of a SS plate of e = 16 with (right) and without (left) constraining internal points

The significant difference between vibrations of clamped and SS plates is the existence of a larger number of in-plane modes in the SS plate as compared to that in the clamped plate. Most of these modes are partially localized thus resulting in non-zero strain energies in both regions of the plate, e.g., see mode 4 in Figure 3.7.

Unidirectional fiber-Reinforced Laminated Plate

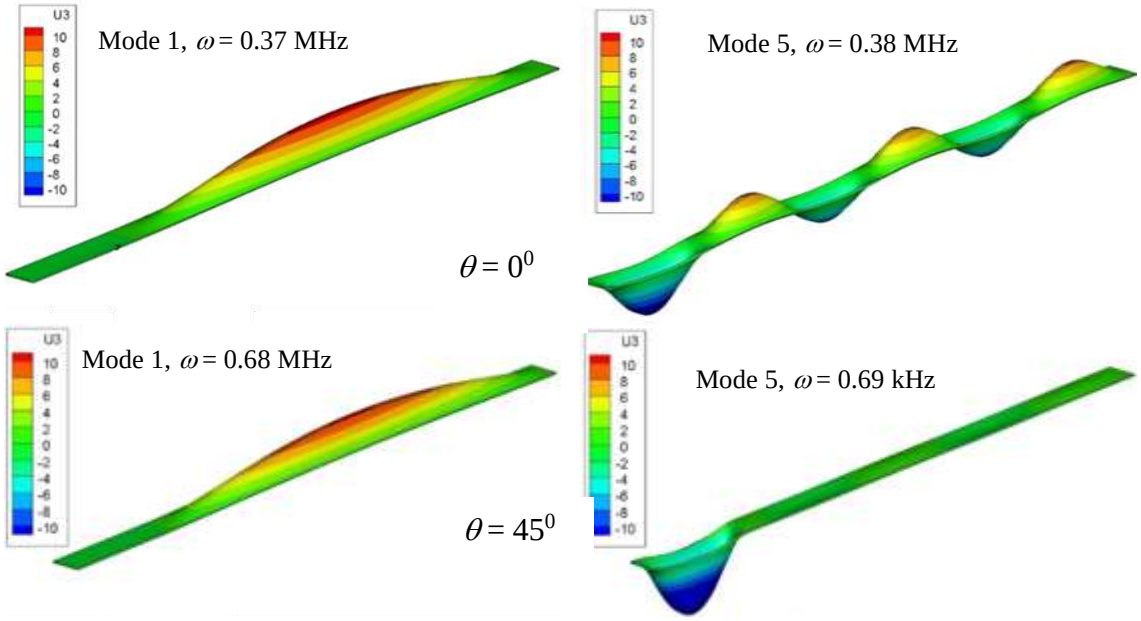
We model a unidirectional fiber-reinforced lamina as transversely isotropic with the fiber direction as the axis of transverse isotropy, and assign following values to material parameters.

$$E_L = 140 \text{ GPa}, E_T = E_L / 25, G_{LT} = E_L / 50, \\ G_{TT} = E_L / 125, \nu_{TL} = \nu_{TT} = 0.25, \rho = 5 \text{ g/cc} \tag{3.18}$$

Here subscripts *L* and *T*, respectively, describe directions parallel and perpendicular (or transverse) to the fiber direction. Material properties with respect to the global coordinate axes are deduced from these by using the tensor transformation rules for stresses and strains.

Clamped edges

For clamped thin rectangular laminates (thickness = length/400) with the axis of transverse isotropy or fiber angle, θ, in all layers of 0^o, 30^o, 45^o, 60^o and 90^o counterclockwise to the global x_1- axis, and eccentricity e = 20, we find their first 100 frequencies and the corresponding mode shapes with and without internal points on the line (l/5, b/2, z) constrained. We note that the elastic moduli with respect to the global coordinate axes depend upon θ. Hence the global stiffness matrix for the plate varies with the angle θ. Mode shapes for the first and the fifth mode of vibration for three laminae with θ = 0^o, 45^o and 90^o and internally constrained points are presented in Figure 3.9. These suggest that the



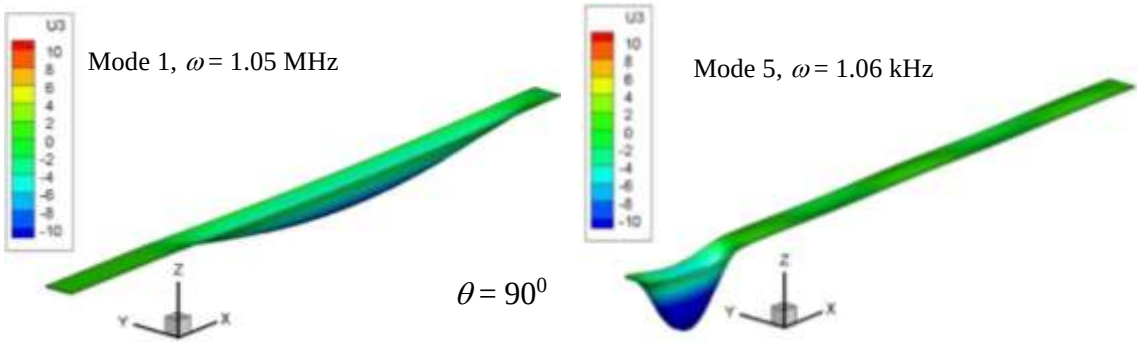


Figure 3.9 Mode shapes of mode 1 (left) and mode 5 (right) of vibration of internally constrained clamped rectangular laminae of e = 20 for fiber angles of 0^0, 45^0 and 90^0

deformation profile for mode 1 (mode 5) is virtually unaffected (significantly influenced) by the fiber orientation angle. For θ = 45º and 90º the mode shapes for mode 5 are virtually identical, similar to what was found in [48] using the FSDT. For the five values of θ, the histograms for the distribution of the mode localization parameter β_1 are given in Figure 3.10. In the θ = 0º, 45º and 90º lamina, the number of modes with β_1 ~ 0.0 equals, respectively, 30, 44 and 66 (43, 49 and 65) from the FSDT (TSNDT) solution. Thus, the number of modes localized in region R_1 increases with an increase in θ.

Simply supported edges

For SS lamina with θ = 0º, 30º, 45º, 60º and 90º and eccentricity e = 4, only 7 (23) modes are localized for θ = 0 º (90 º) when internal points are constrained. The plate exhibits an interesting behavior for the localization of the in-plane modes of vibration with the change in the fiber orientation angle. In order to see this, we have plotted in Figure 3.11 deformed shapes of the plate for the five fiber angles and have included fringe plots of the transverse displacement, u_3. We have exhibited in Figure 3.12 the top view of the deformed plate for

θ = 45º, 60º and 90º with fringe plots of the in-plane displacement u_2. We see in plots of Figure 3.11 that for θ = 0º and 30º, there is significant transverse displacement as compared to that for θ = 45º, 60º and 90º. Whereas values of in-plane displacement u_2 are negligible for θ = 0 º they are noticeable for θ = 45º and 90º. Thus, the ratio of energies, β_1, does not correctly represent the mode localization phenomenon for all values of θ. However, for θ = 45º and 90º, as shown in Figure 3.12, the interior constrained points divide the plate into regions R_1 and R_2 one of which has very little deformations as is for isotropic plates.

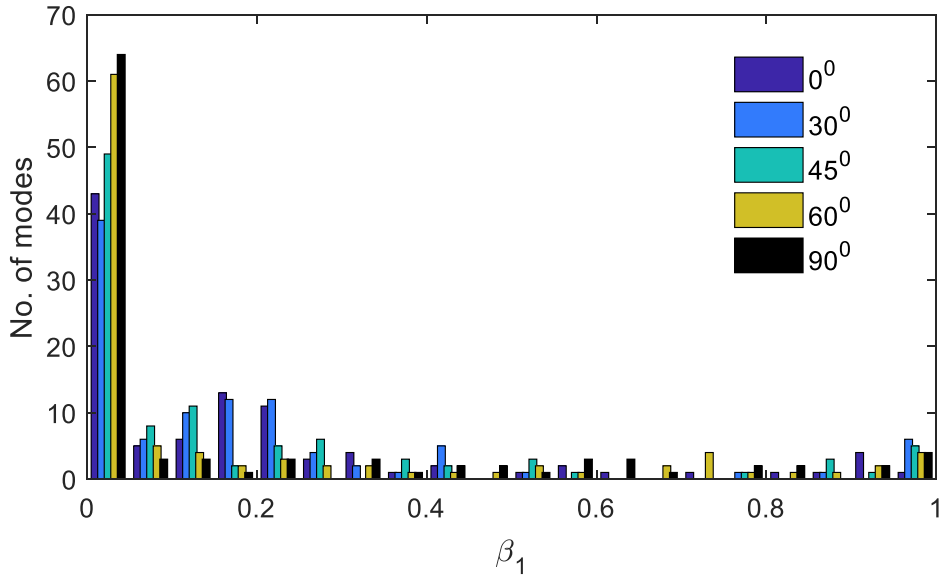


Figure 3.10 Histogram of the distribution of β_1 over the first 100 modes of vibration of the internally constrained clamped laminate with different fiber angles

We observe from results in Figure 3.13 that the plate with the 90^0 fibers has 22 localized modes that include both the out-of-plane and the in-plane modes of vibration, and the plate with the 0^0 fibers only 8 modes localized. For the 0º (90º) plate, $\beta_1 = 0.2$ for 43 (28) modes. Mode shapes for a few modes localized in region R_1 are presented in Figure 3.14. We

observe that the deformation of mode 14 is partially localized in region R_1, it is similar to that of mode 4 for the isotropic SS plate for which results are shown in Figure 3.7. Similarly, partial localization can be seen for mode 17 for which although the deformation localized in R_1, the region R_2 has significant deformations that contribute to the strain energy and accordingly β_1 is not close to 0.

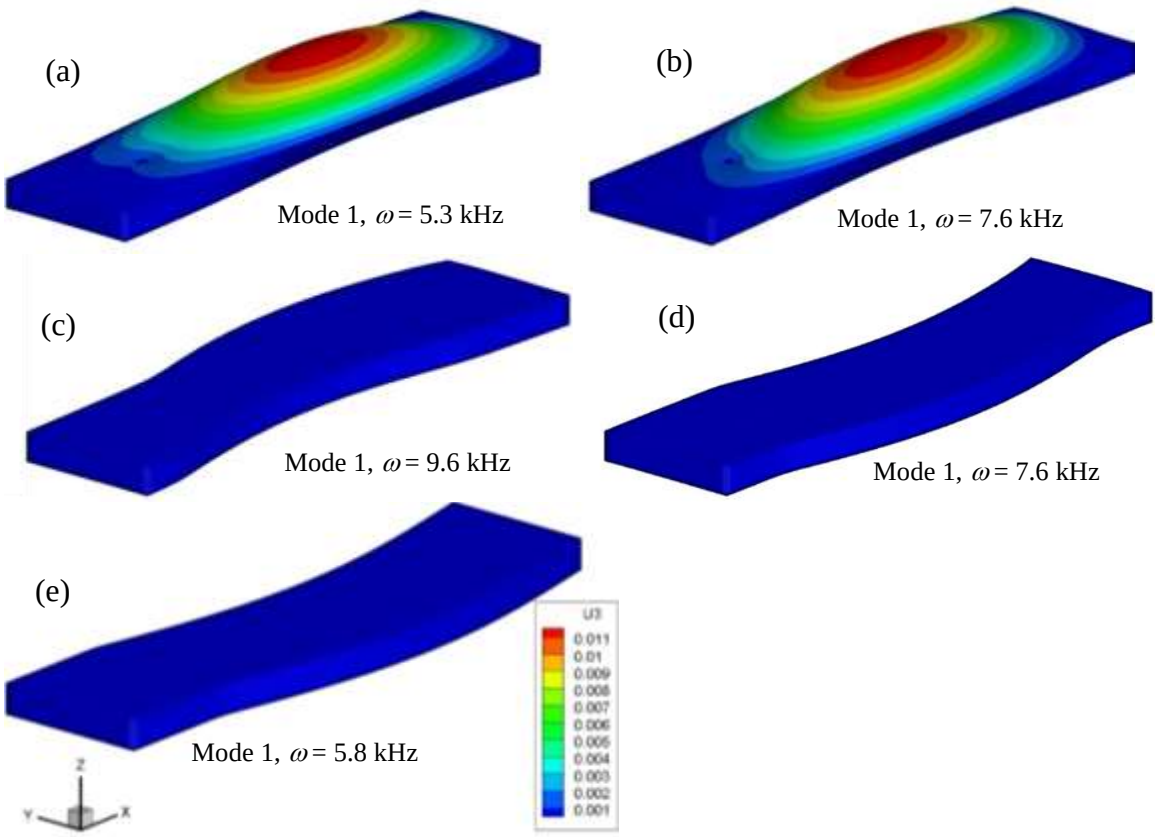


Figure 3.11. Fringe plots of the out-of-plane displacement, u3, for the fundamental mode of vibration of the internally constrained plate with fiber angles of (a) 0 ^{o}, (b) 30 ^{o}, (c) 45 ^{o}, (d) 60 ^{o}, and (e) 90 ^{o} counterclockwise to the global x1-axis. The fringe colors represent same levels of u3 (in mm) for each plot

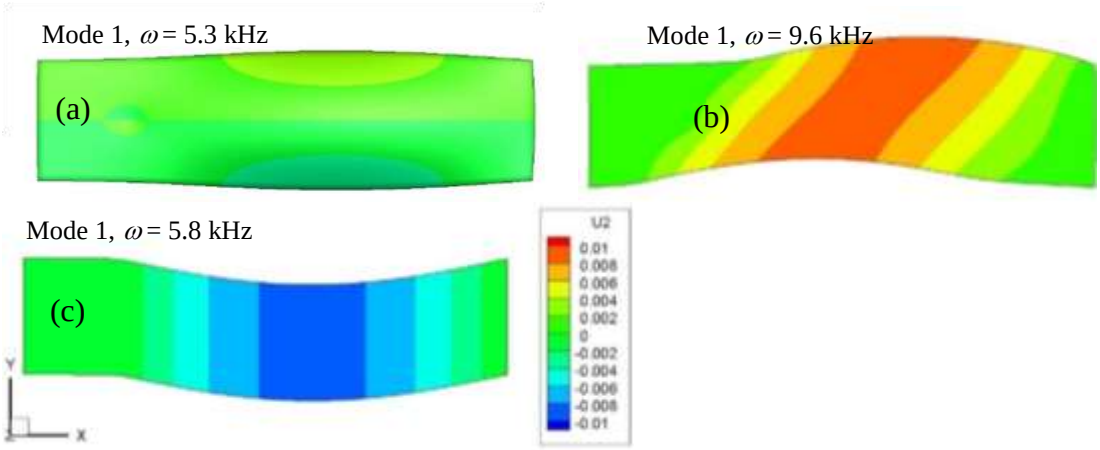


Figure 3.12 Top view of the mode shapes and fringe plots of the in-plane displacement, u_2, for fiber angles of (a) 0 °, (b) 45° and (c) 90 °. The fringe colors represent same levels of u_2 (in mm) for each plot

3.4.3 Constrained Points on Two Normals for an Isotropic SS Plate

We now explore the effect of clamping two sets of internal points on mode localization of a SS 80 mm x 20 mm x 2 mm plate with either points (l/5, b/2, z) and (4l/5, b/2, z) or points (l/10, b/2, z) and (4l/5, b/2, z) constrained. The first (second) pair of points are symmetrically (asymmetrically) located about the surface $x_1 = l/2$. Mode shapes for modes 1, 3 and 5 for the first and the second pairs of points are presented in Figure 3.15. We conclude from results for the 5th mode of free vibration that for the symmetrically located pair, the deformation is entirely localized in the shorter sections at both ends. However, for the asymmetrically located pair of points, the deformation is entirely localized in the 1/5th of the plate between $x_1 = 0.8l$ and l, and for none of the first 100 modes of vibration it localized in the (l/10)th of the left end of the plate.

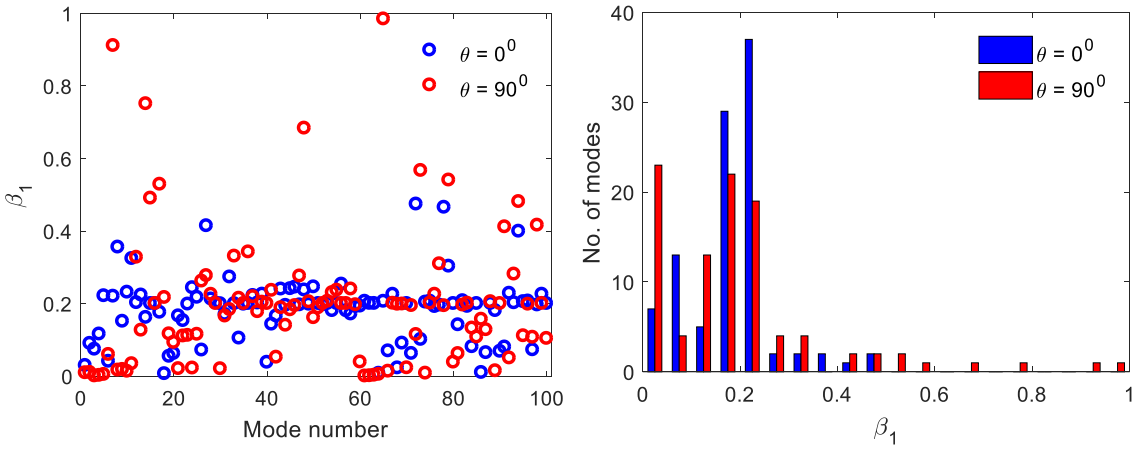


Figure 3.13 Distribution of the energy ratio β_1 over the first 100 modes of vibration of the fiber-reinforced lamina with fibers oriented at 0^0 and 90^0 to the global x_1- axis (left), and the corresponding histogram (right)

3.4.4 Transient Deformations of SS Isotropic Plates

In order to ascertain how constraining interior points affects plate's forced vibrations, we study deformations of the 80 mm x 20 mm x 2 mm SS plate (E = 25 GPa, $\nu = 0.25$ and $\rho = 5g/cc$) with and without internally constrained points (l/5, b/2, x_3) by using the 80 x 20 FE mesh of uniform elements, and the time step = 50 ns that satisfies the stability condition given in Eq. (5.13). Results for plates with e = 4 and 20 were found to be similar.

For the plate with e = 4, as seen from Figure 3.6, mode 6 is localized in region R_2 and modes 1 through 5 are localized in region R_1. In the first loading scenario, depicted in Figure 3.16, the impulsive load on the entire top surface of the plate is non-zero for $0 \le t \le 40\mu s$ and has either a triangular, or a rectangular or a half sinewave form. Thus, different impulse or linear momentum is imparted to the plate for the three loads having

possibly different dominant frequencies. In the second scenario, we scrutinize the effect of varying the frequency of the sustained applied sinusoidal load.

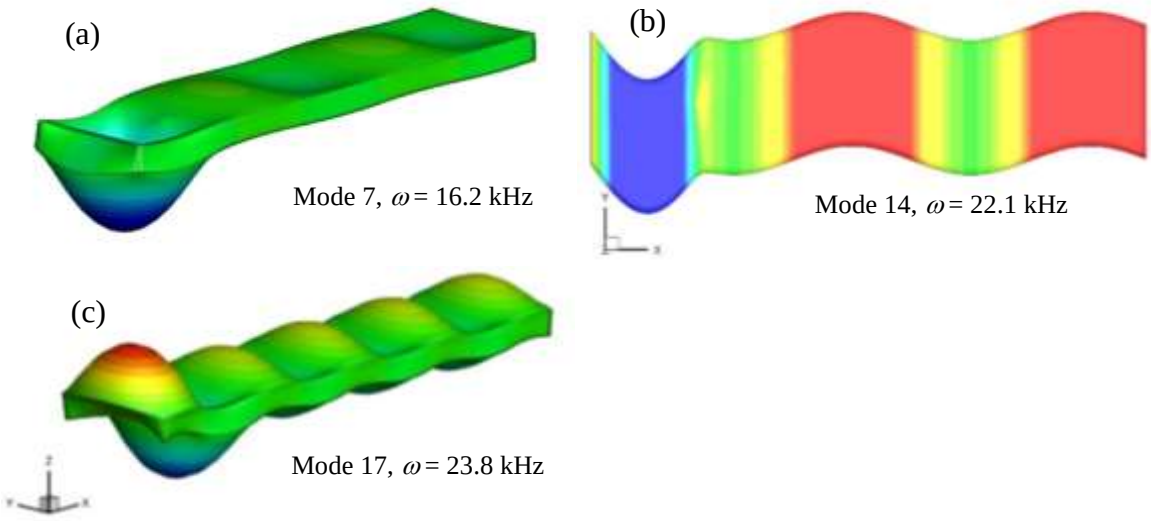


Figure 3.14 Localized mode shapes for the 90^0 composite plate; (a) mode 7, (b) top view of mode 14, and (c) mode 17

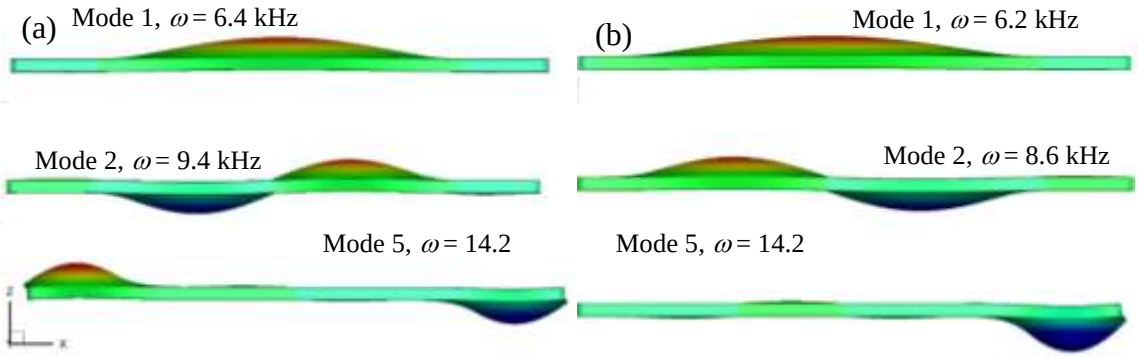


Figure 3.15 Shapes of modes 1, 2 and 5 showing deformation localization in the SS plate with two (a) symmetrically, and (b) asymmetrically located pair of constrained points.

We note that an elegant way to analyze a transient problem is to use the mode superposition method that clearly gives the contribution of a mode to the solution. We could not use it here since expressing mode shapes of an internally constrained plate in terms of either polynomials or trigonometric functions is an arduous task.

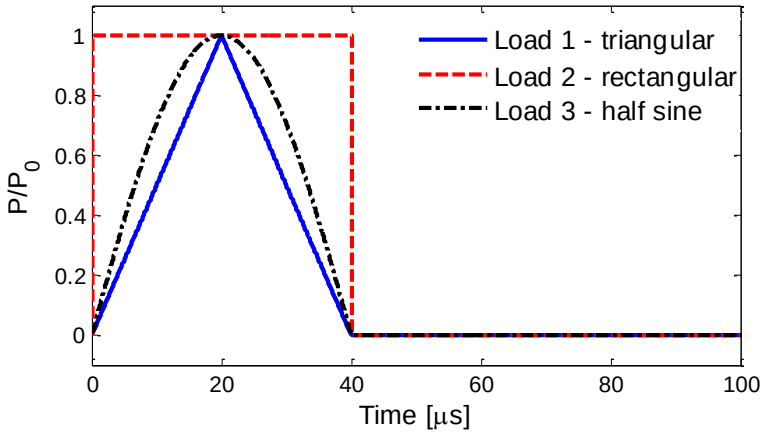


Figure 3.16 Three impulse loads considered

Impulsive Loads

We have depicted in Figure 3.17 time histories of the centroidal deflection and of the strain energy density of regions R_1 and R_2 for the three impulsive loads. It is clear that the loading function only affects the amplitude of the deflection and of the strain energy density, and the two regions vibrate essentially at different frequencies subsequent to the load removal at t = 40 μs. The dominant frequencies of vibration of regions R_1 and R_2 found using the fast Fourier transform (FFT) of the time histories of the centroidal deflection correspond, respectively, to those of modes 3 and 1 of the entire plate rather than to those of modes 6 and 1 for which free vibrations get localized in regions R_2 and R_1, respectively. It suggests that for forced vibrations the two regions deform differently from that for free vibrations.

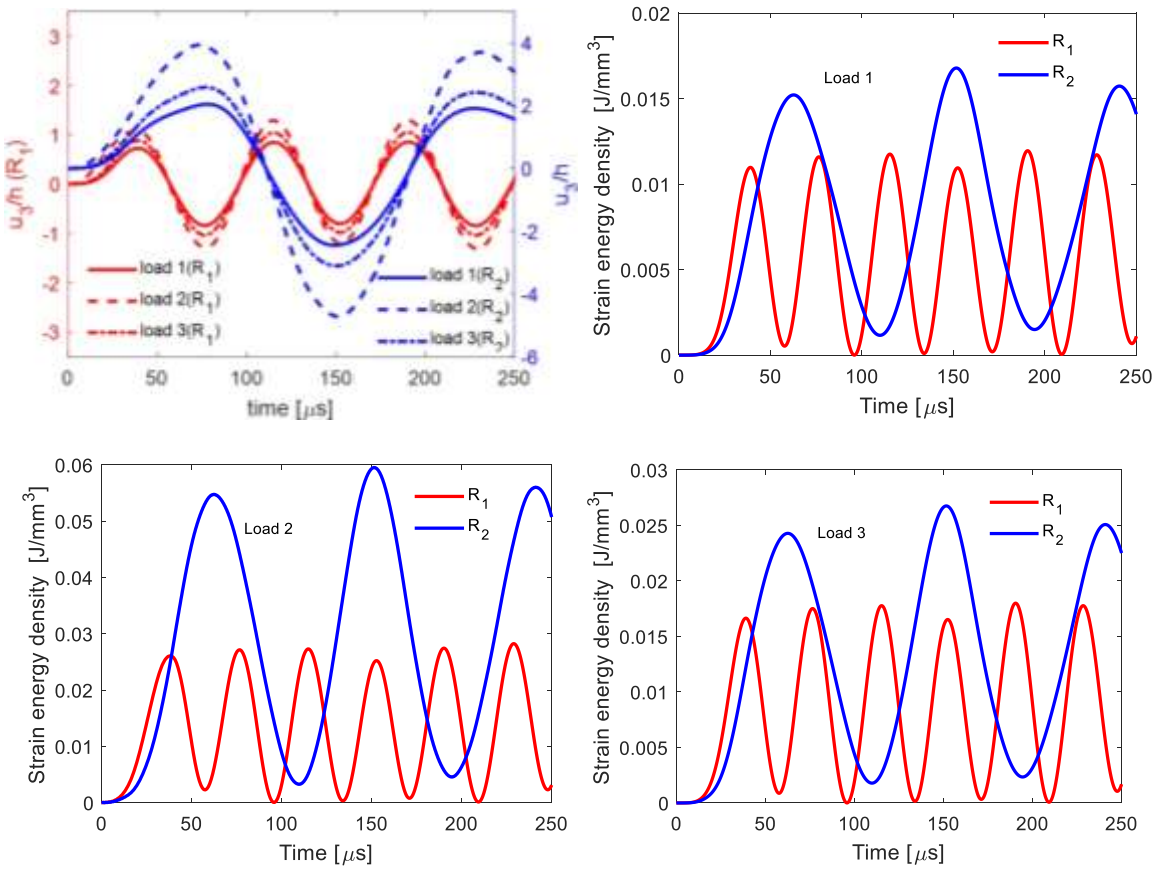


Figure 3.17 For the three transient loads, time histories of the centroidal deflection and of the strain energy densities of regions R_1 and R_2 of the plate with internal constrained points

Sustained Sinusoidal Load on a Rectangular Plate of e = 4

For the pressure load, $P(t) = P_0 \sin(2\pi\omega_p t)$, of frequency ω_p equal to 5.9 and 14.5 kHz for modes 1 and 6 of free vibration of the plate, referred henceforth to as *mode 1 excitation* and *mode 6 excitation frequency*, respectively, we have presented in Figure 3.18 time histories of the centroidal displacements and of the strain energy densities of regions R_1 and R_2. We observe that for *mode 1 excitation*, the region R_1 stays nearly at rest as was for

free vibration but the amplitude of vibration of region R_2 monotonically increases and its vibrational frequency found by the FFT analysis of its vibrational response equals approximately 5.9 kHz. For the *mode 6 excitation*, the amplitude of vibration of region R_2 stays small but that of region R_1 exhibits beats phenomenon. The FFT analysis of its vibrational response gives the dominant frequency of vibration of region R_1 equal to ~ 14.5 kHz, i.e., the frequency of mode 6 of free vibration of the entire plate or the excitation frequency of the load, and the region R_2 vibrates at the fundamental frequency, 5.9 kHz, of the plate. The time histories of the ratio of the total energies (TE = kinetic energy + strain energy) of the sections R_1 and R_2, and of the ratio of the TE of each region to the cumulative external work on the entire plate are presented in Figure 3.19. It is observed from these plots that the energy is transferred from region R_1 to region R_2 that vibrates at a much lower amplitude. It is supported by the observation that, for the first case, the strain energy of R_1 is negligible as compared to that of R_2, and in the second case most of the plate deformation is localized in region R_1. This is akin to the response exhibited by the interaction between two pendula of different frequencies explained in textbooks on vibrations (e.g., see [21]). These results are consistent with Malatkar and Nayfeh's [62] observations of the energy transfer between two widely spaced modes of vibration of a cantilever beam. To delineate the role of the internal constrained points on the phenomenon, the centroidal displacement history of the unconstrained plate under mode 6 excitation is presented in Figure 3.20. This knowledge can help design structures subjected to periodic loads for which a smaller substructure that absorbs most of the energy can be sacrificed and the larger substructure saved.

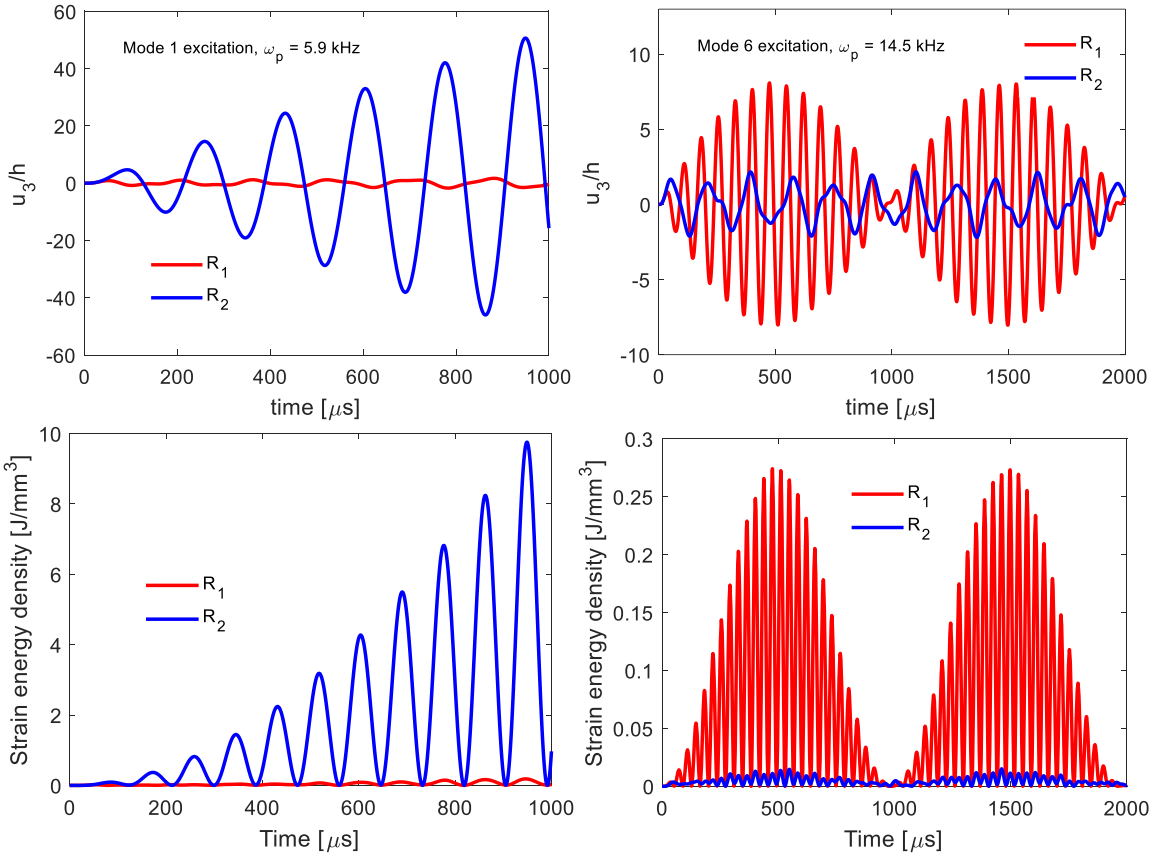


Figure 3.18 Time histories of the centroidal deflection and of the strain energy densities of the two regions of the plate for the mode 1 and the mode 6 excitation frequencies

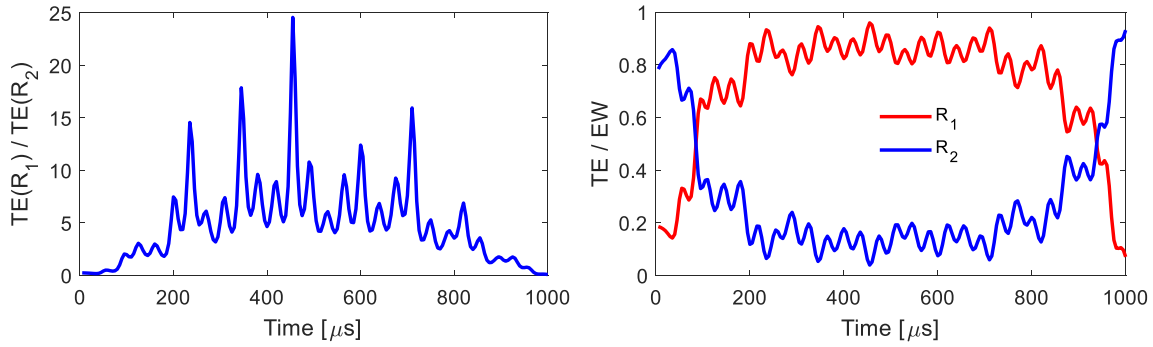


Figure 3.19 Time histories of the ratio of the total energy of regions R_1 and R_2 (left) and of the total energies of each region normalized by the external work done on the entire plate (right)

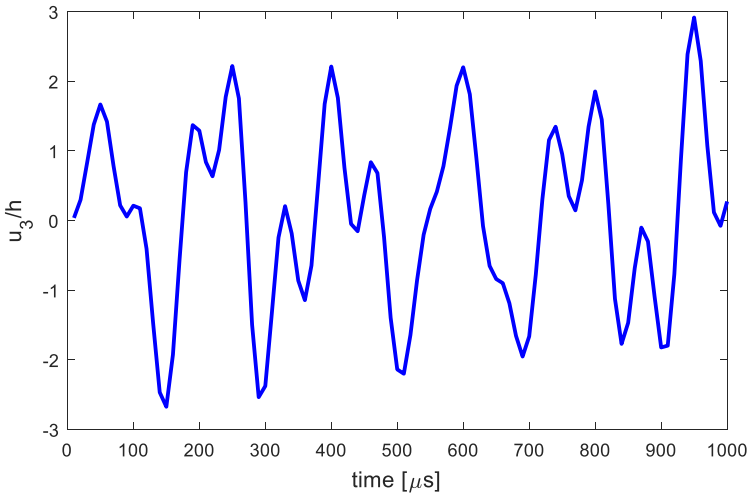


Figure 3.20 centroidal displacement history of an internally unconstrained SS plate under mode 6 harmonic loading

Sustained Sinusoidal Load on a Rectangular Plate of e = 20

For the SS plate of e = 20, mode 4 (8) is the first transverse mode of vibration for which the deformation localized in the region R_2 (R_1). The shapes and the corresponding frequencies of modes 4 and 8 of the plate with and without the internal constraint are presented in Figure 3.21.

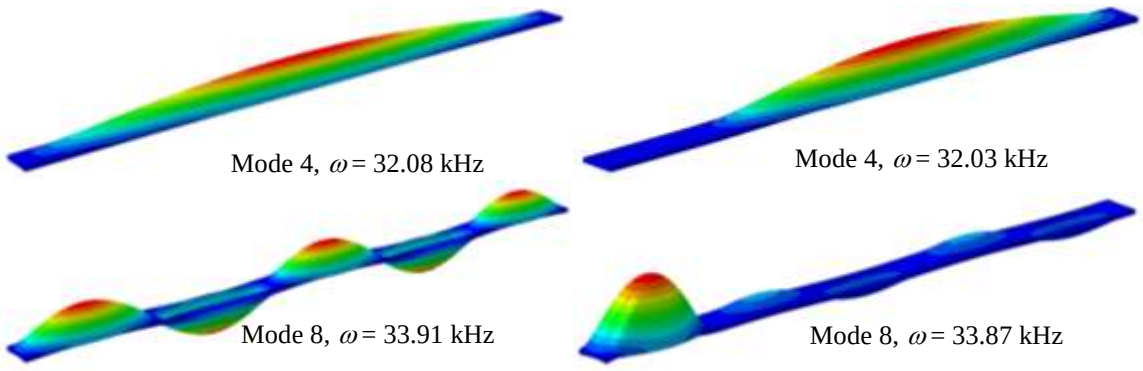


Figure 3.21 Mode shapes of transverse vibration for the SS plate of e = 20 without (left) and with (right) the internal constraint points

As mentioned above for results exhibited in Figure 3.4, the addition of the internal constraint does not noticeably affect the frequency of vibration of a particular mode but significantly changes the mode shape. Unlike the plate with e = 4 where frequencies of the modes 1 and 6 were wide apart, for the plate with e = 20 frequencies of the 4th and the 8th modes are close to each other. It is thus likely that the plate would exhibit a different phenomenon under a harmonic excitation of frequency of the 8th mode as compared to that of the e = 4 plate under mode 6 excitation. For the pressure load $P(t) = P_0 \sin\left(2\pi\omega_p t\right)$ with ω_p (in Hz) as the frequencies of modes 4 and 8, of vibration, the time histories of the centroidal displacements of regions R_1 and R_2 and their corresponding FFT are presented in Figure 3.22. It is clear that unlike for the e = 4 plate, depending on the excitation frequency, a region of the e = 20 plate resonates while the other region exhibits the beat phenomenon due to close values of frequencies of the two modes. Under the mode 4 excitation, the FFT reveals that region R_2 resonates at the mode 4 frequency of 32 kHz while the region R_1 vibrates at approximately 33.5 kHz which is close to the mode 8 frequency. Similarly, for mode 8 excitation, region R_1 resonates at 33 kHz while region R_2 exhibits beating phenomenon at 32 kHz. The rather flat region in the FFT of region R_1 is because the centroidal deflection was output at 1024 values of time.

From time histories of the ratio of the TE of the two regions, and of the ratio of the TE of each region to the work done on the entire plate by external forces, EW, exhibited in Figure 3.23, we observed that the TE of region R_1 steadily decreases from 0.25 (ratio of volumes

of regions R_1 and R_2) to 0 implying that the total energy of the plate is concentrated in R_2. Similarly, the ratio of the total energy to the EW shows that the TE of the region R_2 gradually increases and that of R_1 decreases. We hypothesize that this is due to the resonance of the region R_2.

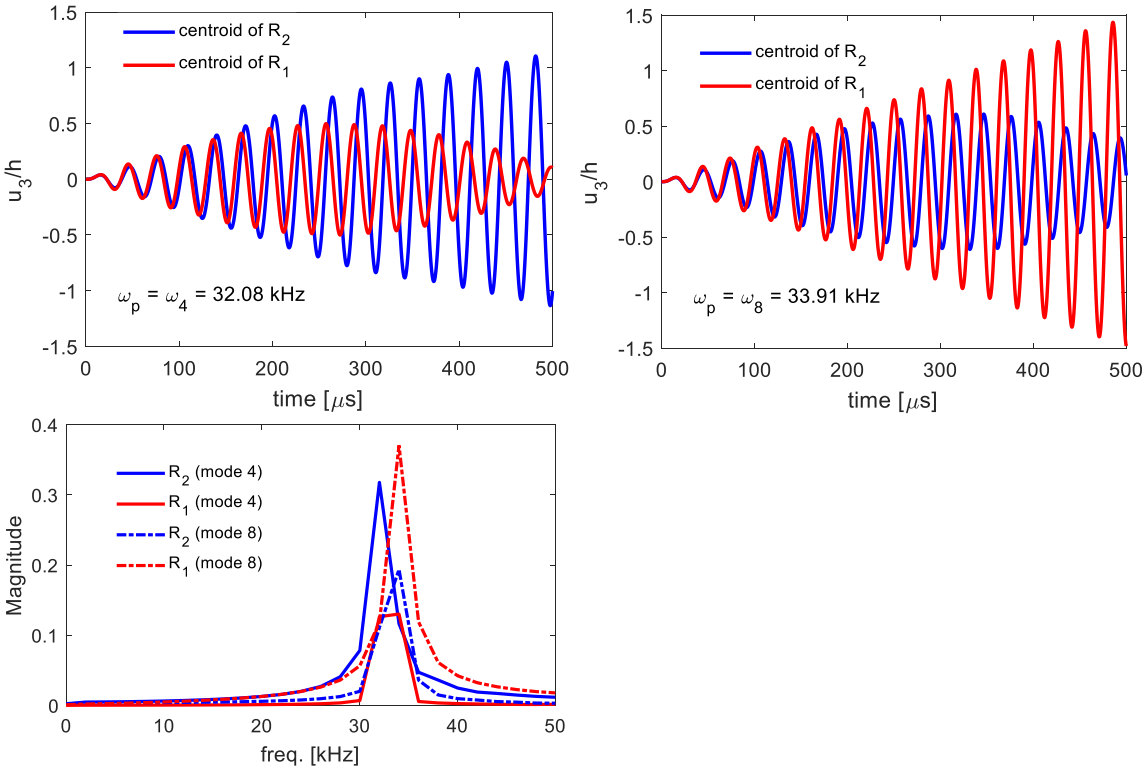


Figure 3.22 Displacement histories of centroids of regions R_1 and R_2 under harmonic loads of the two excitation frequencies

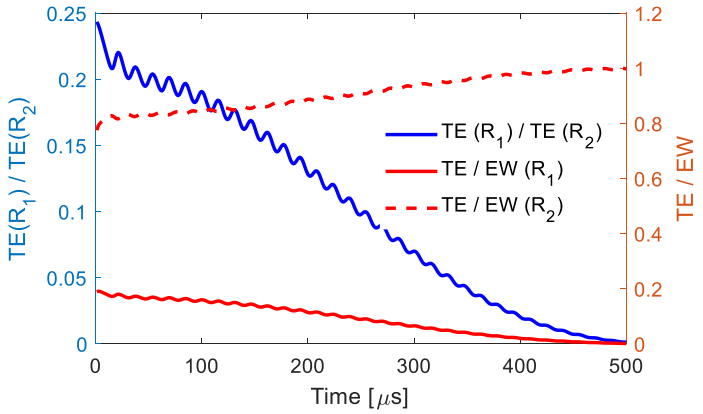


Figure 3.23 Time histories of ratio of the total energies of the regions R_1 and R_2 (left) and the ratio of the total energy of each section of the plate to the cumulative external work done on the entire plate

In order to delineate effects of internal constraints, the displacement histories of the centroid of the unconstrained plate and the results of the corresponding FFT analyses under the two excitations are presented in Figure 3.24. The displacement history of the plate under the mode 4 excitation shows a monotonic increase in the amplitude due to the resonance of the plate. However, for the mode 8 excitation, we see the beating phenomenon since the excitation frequency is close to the fundamental frequency of the plate. This behavior is different from the response of the e = 4 unconstrained plate under *mode 6 excitation* where neither the resonance nor the beats phenomenon was observed due to the large difference between the excitation and the fundamental frequencies of the plate. For the e = 20 plate, the FFTs of the displacement histories show the dominant frequency of the plate vibration for the *mode 4 (8) excitation* is about 31 (34) kHz.

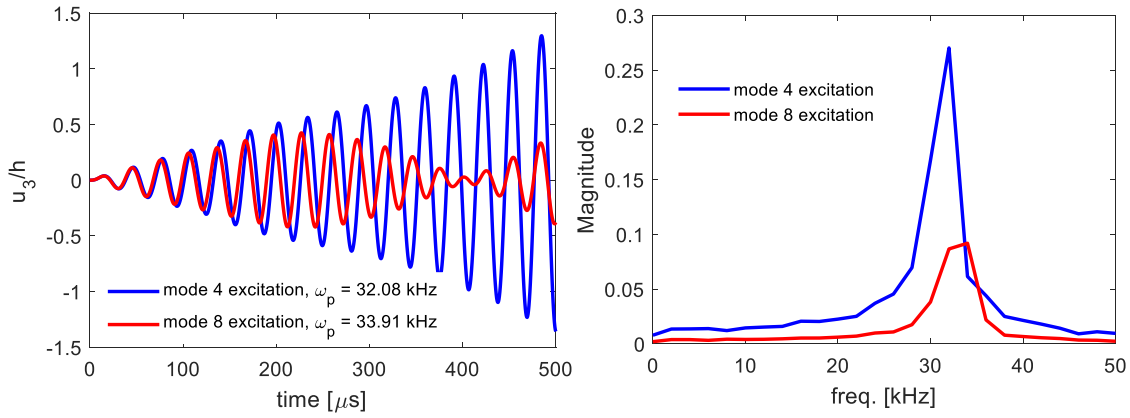


Figure 3.24 Centroidal displacement histories of the rectangular plate of e = 20 without internal constraints under mode 4 and mode 8 excitations and the corresponding FFTs of the displacement histories

3.5 Note

For forced vibrations of delaminated plates and laminates studied in [63-64], no localization of deformations was reported. Mode localization has been numerically studied in reference [65]

3.6 Conclusions

We have numerically studied free and forced vibrations of monolithic and unidirectional fiber-reinforced composite rectangular plates with edges either simply supported or clamped using a third order shear and normal deformable plate theory (TSNDT). Frequencies and strain energies of the first 100 modes of vibration are shown to agree well with those computed using the linear theory of elasticity and the commercial software, ABAQUS. By constraining all points on one or two normals to the mid-surface of a plate to have null displacements, the plate deformations are found to localize in one of the two

regions separated by the internal constrained points. Significant results from the work include the following.

- When an in-plane mode of vibration is localized, the strain energy of deformations of the other region is not small.

- For rectangular plates with points on two normals constrained from translating in all three directions, the localization occurs simultaneously in two short regions when the constrained points are equidistant from the plate edges.

- A unidirectional fiber-reinforced rectangular plate with internal constrained points switches from a transverse (bending) mode to an in-plane mode of vibration depending on the fiber orientation angle, and both modes exhibit the mode localization phenomenon.

- For forced vibrations of plates, constraining points on a normal to the plate midsurface divides the plate into two separate sections vibrating at different dominant frequencies. These regions interact with each other through energy transfer resulting in constructive/destructive interference that results in a beating-like phenomenon under suitable loading conditions and plate geometries.

- The mode localization phenomenon can help design cyclically loaded structures so a desired sub-region of the structure is significantly deformed, thereby protecting the remainder of the structure, and in maximizing energy harvested from them.

3.7 Funding

This work was partially supported by the US Office of Naval research (ONR) grant N00014-18-1-2548 to Virginia Polytechnic Institute and State University with Dr. Y.D.S. Rajapakse as the program manager. Views expressed in the paper are those of the authors and neither of ONR nor of Virginia Tech.

4 Finite Deformations of Nonlinearly Elastic Plates using the Third Order Shear and Normal Deformable Theory

Abstract

We study the response of rectangular plates subjected to different boundary and loading conditions using the TSNDT. In this approach, the displacements of a point in the directions of the three coordinate axes are expressed as complete third degree polynomials of the thickness coordinate using a 3rd order Taylor series expansions about the mid-surface of the plate. This results in transformation of the problem into solving for the unknown coefficients of different powers of the thickness coordinate at the points on the mid-surface to compute the displacements of any point in the plate, thus decreasing the complexity of the problem from a 3D domain to a two dimensional (2D) area domain. We formulate the mathematical model for finite transient deformations of rectangular plates inclusive of all geometric and material nonlinearities and solve the system of equations numerically using the finite element method (FEM). The computations are performed using an in-house developed numerical software. The efficacy of the software is verified by comparing the obtained numerical results with those from literature and from 3D FEM simulations from the commercial software ABAQUS. We study the differences in the results predicted by linear and nonlinear models to delineate the effects of nonlinearity in the response of plate structures. Different tensile and compressive behavior of St. Venant-Kirchhoff plates and its effects on characteristics of the finite deformations are analyzed. Wave propagation in the transient analysis of the plate deformation and the capability of the TSNDT in studying this is also investigated. Based on the numerical results we establish that TSNDT is capable

of analyzing the finite deformations of nonlinear elastic rectangular plates to reasonable accuracy.

4.1 Introduction

Plates and shells have a diverse range of applications in structural design, and numerous plate theories have been proposed. A primary goal of these theories is to predict deformations close to those given by the solution of equations governing three-dimensional (3-D) deformations of the structure. As far as we know, there is no single plate theory that accomplishes this for all plate geometries, edge conditions, loads and material behavior.

Analytical solutions of 3-D equations for linearly elastic plates with specific edge conditions have been given by Chree [66], Vlasov [67], Pagano [68-69], Noor [70], Noor and Rarig [71], Ren [72], Bhaskar and Vardhan [73-74], Srinivas and Rao [24] and Vel and Batra [50, 61, 75]. Difficulties in obtaining analytical solutions for plates and shells of arbitrary geometries, boundary conditions and dynamic loads have necessitated using numerical (e.g., the finite element (FE), meshless, collocation, finite-difference, generalized differential quadrature) methods. These method differ in the choice of basis functions used to approximate the displacement field and generally provide a converged solution for linear problems with an increase in the number of degrees of freedom (DoFs). Here, we do not delve into comparing the performances of different numerical schemes except to note that here we use the FEM to analyze transient finite deformations of a St. Venant-Kirchhoff plate for which the 2nd Piola-Kirchhoff stress tensor, **S**, is a linear

function of the Green-St. Venant strain tensor, **E**. Of course, the analysis and the algorithm is applicable to other material models.

The Kirchhoff-Love plate theory [1-2], usually called the classical plate theory (CPT) is the simplest one and is based on the assumption that a normal to the mid-surface in the reference configuration always stays normal to it and remains un-stretched implying null transverse normal and shear strains for infinitesimal deformations. This theory generally works well for studying many deformations of thin plates (length/height > 50). For thicker plates and especially those involving thermo-mechanical deformations, transverse shear and normal strains may significantly contribute to the strain energy of deformations.

A simple plate theory that considers transverse shear but not normal deformations is the first order shear deformation theory (FSDT) also called the Reissner-Mindlin [3-4, 76] plate theory. In the FSDT, the in-plane displacements of a point are expressed as affine functions of the thickness coordinate, and the out-of-plane deflection is assumed to be independent of the thickness coordinate. Mindlin [4] proposed a shear correction factor to partially account for the non-uniform through-the-thickness variation of shear stresses so that plate's natural frequencies are close to those found using the 3D linear elasticity theory. Mindlin, amongst others, also proposed expanding the three displacement components as power series in terms of the thickness coordinate, *z*.

Nelson and Lorch [77] retained up to 2^{nd} order terms in z in the power series expansions of the three displacements, and used a shear correction factor to get through-the-thickness shear stresses closer to those given by the solution of the 3D problem. A plate theory having

the three displacements expanded up to z^K is called K^{th} order. Lo et al. [52] considered up to z^3 (z^2) terms for the in-plane (out-of-plane) displacements. Reddy [5] reduced the number of unknowns in a higher order plate theory (HSDT) by enforcing that the problem formulation identically satisfy null shear tractions on the top and the bottom surfaces and the plate deflection be independent of z. Since the transverse normal strain identically vanishes, the transverse shear and normal stresses are computed by using a stress recovery scheme (SRS). Other HSDTs include those proposed by Pandya and Kant [78] and Carrera and Demassi [8, 53]. Books by Naghdi [79], Antman [80], Green and Naghdi [81], Flugge [82], Leissa [83-84], Ambartsumian [85] and [86], and review articles by Koiter and Simmonds [87] and Carrera [88-89], provide immense resources for learning historical developments of plate theories.

Kraus [86] has suggested that 2D shell theories which assume that a normal to the reference mid-surface does not change in length and remains normal to the deformed mid-surface can be grouped as Love's first approximation theories (LFAT). The CPT is one such theory. A shell theory not based on this assumption is grouped into Love's Second Approximation theories (LSAT). Koiter [90], based on energy considerations, recommended that any refinement of the LFAT without simultaneously considering both transverse shear and transverse normal stresses does not improve the quality of the solution over that provided by the LFAT; this is often called Koiter's recommendation (KR).

Following Mindlin and Medick's work [91], Batra and Vidoli [6-7] used Reissner-Prange-Hellinger principle to derive a mixed K^{th} order plate theory in which the three displacements and the six stresses are, respectively, expressed as power series up to z^K and

z^{K+2} . They termed the theory compatible if stresses are derived from the constitutive relations and the plate theory displacements. Both compatible and mixed plate theories do not require correction factors. Qian et al. [25, 92] used the compatible plate theory and a meshless method to numerically study free and forced vibrations of linearly elastic rectangular plates. Batra et al. [60] and Batra and Aimannee [18] used the mixed theory to analyze free vibrations and wave propagation in orthotropic plates, and reported values of *K* that give results in close agreement with those from the 3-D linear elasticity theory. These works found in-plane modes of vibrations not predicted by the analytical solution of Srinivas and Rao [58] who studied only bending deformations. The plate theory for K = 3 is called the third order shear and normal deformable theory (TSNDT). Shah and Batra [93] have used an equivalent single layer TSNDT to analyze static deformations of linearly elastic laminated plates and showed that boundary layer effects near the plate edges using the one-step SRS agree well with those given by the analytical solutions.

Works that have numerically analyzed finite deformations of the St. Venant-Kirchhoff plates/shells using the Green-St. Venant strain tensor include [94-96]. Batra and Xiao [97-98] have analyzed finite deformations of St. Venant-Kirchhoff beams using the (TSDNT); here we extend it to study transient deformations of monolithic and laminated St. Venant-Kirchhoff plates using the FEM. The developed software is verified by comparing results computed from it with those either available in the literature or found the commercial software ABAQUS in which we implemented the St. Venant-Kirchhoff material model.

We note that Cosserat and Cosserat [57] proposed a plate theory in which transverse deformations are considered by analyzing deformations of one or more linearly

independent deformable directors attached to every point the plate reference (e.g., mid-) surface but not in its tangent plane. Ericksen [99] has studied propagation of waves in homogenous solid plates using this theory. Representative works for studying deformations of Cosserat plates/shells are [100-102].

The rest of the chapter is organized as follows. The problem formulation is described in Section 2, the computational algorithm is briefly reviewed in Section 3 and the verification of the software is detailed in Section 4. Section 5 provides results for several example problems with conclusions from the work summarized in Section 6.

4.2 Problem Formulation

4.2.1 Kinematics

A schematic sketch of *a* x *b* x *h* rectangular plate is exhibited in Figure 4.1. We use rectangular Cartesian coordinate axes with X_i and x_i denoting coordinates of an arbitrary material point in the reference and the current configurations, respectively. Thus, $u_i = u_i\left(X_1, X_2, X_3, t\right) = x_i\left(X_1, X_2, X_3, t\right) - X_i$, gives the displacement of the material point **X**. The deformations gradient, F_{iL}, and the Green-St. Venant strain tensor, E_{LM}, are given by

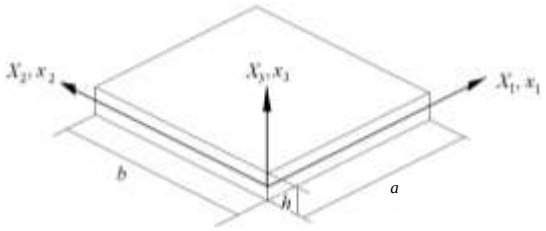


Figure 4.1 Schematic representation of a rectangular plate and the coordinate

$$F_{iL} = \frac{\partial u_i}{\partial X_L} + \delta_{iL} \tag{4.1}$$

$$E_{LM} = \frac{1}{2}\left(F_{kL}F_{kM} - \delta_{LM}\right) \tag{4.2}$$

where δ_{ij} (= 1 for $i = j$ and 0 for $i \neq j$) is the Kronecker delta, a repeated index implies summation over the range of the index, and lower (upper) case indices refer to components with respect to coordinates in the current (reference) configuration. Substitution from Eq. (4.1) into Eq. (4.2) gives

$$E_{11} = \underbrace{\underbrace{\frac{\partial u_1}{\partial X_1} + \underbrace{\frac{1}{2}\left(\frac{\partial u_3}{\partial X_1}\right)^2}_{I}}_{II} + \underbrace{\frac{1}{2}\left[\left(\frac{\partial u_1}{\partial X_1}\right)^2 + \left(\frac{\partial u_2}{\partial X_1}\right)^2\right]}_{III}}_{III} \tag{4.3.a}$$

$$E_{22} = \underbrace{\underbrace{\frac{\partial u_2}{\partial X_2} + \underbrace{\frac{1}{2}\left(\frac{\partial u_3}{\partial X_2}\right)^2}_{I}}_{II} + \underbrace{\frac{1}{2}\left[\left(\frac{\partial u_1}{\partial X_2}\right)^2 + \left(\frac{\partial u_2}{\partial X_2}\right)^2\right]}_{III}}_{III} \tag{4.3.b}$$

$$E_{33} = \underbrace{\underbrace{\frac{\partial u_3}{\partial X_3} + \underbrace{\frac{1}{2}\left[\left(\frac{\partial u_1}{\partial X_3}\right)^2 + \left(\frac{\partial u_2}{\partial X_3}\right)^2 + \left(\frac{\partial u_3}{\partial X_3}\right)^2\right]}_{I,II}}_{III}}_{III} \tag{4.3.c}$$

$$E_{12} = \underbrace{\underbrace{\underbrace{\frac{1}{2}\left(\frac{\partial u_1}{\partial X_2} + \frac{\partial u_2}{\partial X_1}\right)}_{I} + \underbrace{\frac{1}{2}\left(\frac{\partial u_3}{\partial X_1}\frac{\partial u_3}{\partial X_2}\right)}_{II}}_{II} + \underbrace{\frac{1}{2}\left(\frac{\partial u_1}{\partial X_1}\frac{\partial u_1}{\partial X_2} + \frac{\partial u_2}{\partial X_1}\frac{\partial u_2}{\partial X_2}\right)}_{III}}_{III} \tag{4.3.d}$$

$$E_{23} = \underbrace{\underbrace{\frac{1}{2}\left(\frac{\partial u_2}{\partial X_3} + \frac{\partial u_3}{\partial X_2}\right)}_{I,II} + \frac{1}{2}\left(\frac{\partial u_1}{\partial X_2}\frac{\partial u_1}{\partial X_3} + \frac{\partial u_2}{\partial X_2}\frac{\partial u_2}{\partial X_3} + \frac{\partial u_3}{\partial X_2}\frac{\partial u_3}{\partial X_3}\right)}_{III} \tag{4.3.e}$$

$$E_{31} = \underbrace{\underbrace{\frac{1}{2}\left(\frac{\partial u_3}{\partial X_1} + \frac{\partial u_1}{\partial X_3}\right)}_{I,II} + \frac{1}{2}\left(\frac{\partial u_1}{\partial X_1}\frac{\partial u_1}{\partial X_3} + \frac{\partial u_2}{\partial X_1}\frac{\partial u_2}{\partial X_3} + \frac{\partial u_3}{\partial X_1}\frac{\partial u_3}{\partial X_3}\right)}_{III} \tag{4.3.f}$$

In Eq. (4.3) terms underlined as *I, II* and *III* represent, respectively, the infinitesimal strains ε_{ij}, the von Karman strains, and the Green-St. Venant strains. It is clear that the Green-St. Venant strains include all terms present in the von Karman strains.

As noted in the Introduction, in the TSNDT, displacements of a point are expressed as complete polynomials of degree 3 in the thickness coordinate, X_3. That is,

$$u_i\left(X_1, X_2, X_3, t\right) = \sum_{j=0}^{3} (X_3)^j u_{ij}(X_1, X_2, 0, t), \quad i = 1,2,3 \tag{4.4}$$

In Eqn. (4.4) $u_{ij}(X_1, X_2, 0, t)$ may be regarded as the 12 generalized displacements (or degrees of freedom, DoFs) of a point on plate's mid-surface. Thus the problem of finding $u_i\left(X_1, X_2, X_3, t\right)$ reduces to that of finding $u_{ij}(X_1, X_2, 0, t)$. We note that the CLT and the Reissner-Mindlin (FSDT) kinematic fields can be deduced from Eq. (4.4) by assigning appropriate values to $u_{ij}(X_1, X_2, 0, t)$. For example, for the FSDT, $u_{31} = u_{32} = u_{33} = 0$, $u_{22} =$

$u_{23} = 0$, $u_{12} = u_{13} = 0$. One may also regard $u_{i1}(X_1, X_2, 0, t)$ as three directors attached to the point (X_1, X_2) and set $u_{i2} = u_{i3} = 0$ to get Cosserat's plate theory.

4.2.2 Kinetics

In the referential or the Lagrangian description of motion, transient deformations of a body are governed by

$$\rho_0 \ddot{u}_i = \frac{\partial T_{Li}}{\partial X_L} + f_i \tag{4.5}$$

where **T** is the first Piola-Kirchhoff stress tensor (nominal stress), ρ_0 the mass density in the reference configuration, and f_i the current body force per unit reference volume. Eq. (4.5) is supplemented with the following initial and boundary conditions:

$$\begin{aligned}
&u_i(X_1, X_2, X_3, 0) = u_i^0(X_1, X_2, X_3) \\
&\dot{u}_i(X_1, X_2, X_3, 0) = \dot{u}_i^0(X_1, X_2, X_3) \\
&T_{iL} N_L = t_i(X_1, X_2, X_3, t) \text{ on } \Gamma_t \\
&u_i(X_1, X_2, X_3, t) = u_i^{bc}(X_1, X_2, X_3, t) \text{ on } \Gamma_u \\
&\Gamma_t \cup \Gamma_u = A(\text{boundary surface of the plate}), \Gamma_t \cap \Gamma_u = \phi
\end{aligned} \tag{4.6}$$

In Eq. (4.6), functions u_i^0 and $\dot{u}_i^0$ are the initial displacement and velocity fields, **N** is the unit outward normal to the part Γ_t of the bounding surface on which surface tractions are specified as $t_i(X_1, X_2, X_3, t)$, and Γ_u represents the remaining boundary of the plate where displacements are specified as $u_i^{bc}(X_1, X_2, X_3, t)$. Of course, the problem formulation is

valid so long as linearly independent displacement and traction boundary conditions are prescribed at a point on the boundary.

4.2.3 Plate Theory Equations

In order to derive the plate theory equations, we multiply both sides of Eq. (4.5) with $\left(X_3\right)^k$ and integrate the resulting equation over the plate thickness to obtain the following:

$$\sum_{j=0}^{3} A_{j+1}^{(k)} \ddot{u}_{ij} = \sum_{\alpha=1}^{2} \frac{\partial M_{i\alpha}^{k}}{\partial X_{\alpha}} + \left(\left(X_3\right)^k T_{i3} \right) \Big|_{-h/2}^{h/2} - k M_{i3}^{k-1} + \int_{-h/2}^{h/2} X_3^k f_i dX_3 \qquad (4.7)$$
$$i = 1,2,3,\ j,k = 0,1,2,3,$$

where

$$A_j^{(k)} = \int_{-h/2}^{h/2} \rho_0 X_3^{k+j} dX_3 \ ,\ M_{iL}^{(k)} = \int_{-h/2}^{h/2} \left(X_3\right)^k T_{iL} dX_3, \quad k = 0,1,2,3 \qquad (4.8)$$

Eq. (4.7) is a system of 12 coupled partial differential equations, written explicitly in *Appendix A*, for the 12 unknowns $u_{ij}(X_1, X_2, 0, t)$ provided that the constitutive relation for the plate material is known.

4.2.4 Constitutive Relation for the Plate Material

We assume the plate is comprised of a St. Venant-Kirchhoff material for which the strain energy density per unit reference volume, *W*, and the second Piola-Kirchhoff stress tensor, **S,** are given by,

$$W = \frac{1}{2} E_{LM} C_{LMPQ} E_{PQ}, C_{LMPQ} = C_{PQLM} = C_{QPLM} \qquad (4.9)$$

and

$$S_{LM} = \frac{\partial W}{\partial E_{LM}} = C_{LMNP} E_{NP} \tag{4.10}$$

Here **C,** the fourth order tensor of material elasticities, has 21, 9, 5 and 2 non-vanishing independent components, respectively, for a general anisotropic, orthotropic, transversely isotropic and isotropic material. The Cauchy stress tensor, $\boldsymbol{\sigma}$, is related to **T** and **S** by,

$$\mathbf{T} = \mathbf{FS}, \boldsymbol{\sigma} = \frac{1}{J}\mathbf{FSF}^T \tag{4.11}$$

where *J* equals the determinant of **F**. We note that components of **S** and **T** are, respectively, quadratic and cubic polynomials of gradients with respect to X_α of $u_{ij}(X_1, X_2, 0, t)$, and those of $\boldsymbol{\sigma}$ are, in general, not polynomial functions. Furthermore, **S** has no physical meaning and is used for mathematical convenience. Whereas **S** is work conjugate of the Green-St. Venant strain tensor, ***E***, the von-Karman strain tensor has no work conjugate stress tensor. Ciaret [103] has used the method of asymptotic expansions by taking the plate thickness as the parameter to show that the leading term of the expansion is the solution of a system of equations equivalent to those of FvK. He also showed that displacements for the leading term are those of Kirchhoff-Love type. Justifications of membrane equations for shells are given in Refs. [104-106].

For infinitesimal deformations, E_{LM} is replaced by ε_{LM}, and **S** by $\boldsymbol{\sigma}$. Thus, the constitutive relation (4.10) reduces to Hooke's law, and the three stress tensors, **T**, **S** by $\boldsymbol{\sigma}$ coincide with each other.

With respect to the *material principal axes* of an orthotropic material, Eqn. (4.10) becomes

$$\begin{Bmatrix} S_{11} \\ S_{22} \\ S_{33} \\ S_{23} \\ S_{13} \\ S_{12} \end{Bmatrix} = \begin{bmatrix} C_{1111} & C_{1122} & C_{1133} & 0 & 0 & 0 \\ C_{2211} & C_{22222} & C_{2233} & 0 & 0 & 0 \\ C_{3311} & C_{3322} & C_{3333} & 0 & 0 & 0 \\ 0 & 0 & 0 & C_{4444} & 0 & 0 \\ 0 & 0 & 0 & 0 & C_{5555} & 0 \\ 0 & 0 & 0 & 0 & 0 & C_{6666} \end{bmatrix} \begin{Bmatrix} E_{11} \\ E_{22} \\ E_{33} \\ 2E_{23} \\ 2E_{13} \\ 2E_{12} \end{Bmatrix}$$

$$C_{1111} = \frac{1-\nu_{23}\nu_{32}}{E_2E_3\Delta}, C_{2222} = \frac{1-\nu_{13}\nu_{31}}{E_1E_3\Delta}, C_{3333} = \frac{1-\nu_{12}\nu_{21}}{E_1E_2\Delta},$$

$$C_{1122} = \frac{\nu_{21}+\nu_{31}\nu_{23}}{E_2E_3\Delta}, C_{1133} = \frac{\nu_{31}+\nu_{21}\nu_{32}}{E_2E_3\Delta}, C_{2233} = \frac{\nu_{32}+\nu_{31}\nu_{12}}{E_1E_3\Delta} \tag{4.12}$$

$$C_{2211} = \frac{\nu_{12}+\nu_{13}\nu_{32}}{E_1E_3\Delta}, C_{3311} = \frac{\nu_{13}+\nu_{12}\nu_{23}}{E_1E_2\Delta}, C_{3322} = \frac{\nu_{23}+\nu_{13}\nu_{21}}{E_1E_2\Delta}$$

$$C_{4444} = 2G_{23}, C_{5555} = 2G_{13}, C_{6666} = 2G_{12}, \Delta = \frac{1-\nu_{12}\nu_{21}-\nu_{23}\nu_{32}-\nu_{13}\nu_{31}-2\nu_{12}\nu_{21}\nu_{31}}{E_1E_2E_3}$$

In Eq. (4.12), E_1, E_2 and E_3 are Young's moduli along the three material principal axes, and ν's are Poisson's ratios. For a realistic material response, C_{4444}, C_{5555} and C_{6666} must be positive, and three eigenvalues of the remaining 3 x 3 matrix must on the right-hand side of Eq. (4.12) be positive.

For simple tension/compression and simple shear deformations of a homogeneous body, Batra [107-108] has compared the response of the St. Venant-Kirchhoff material with that of materials represented by three other relations between $\boldsymbol{\sigma}$ and $\boldsymbol{B}$. For uniaxial and simple shear deformations of an isotropic and homogeneous body, Figure 4.2 displays the relation between the nominal stress, **T,** and the relevant deformation measure. It is clear that for infinitesimal deformations, the stress-strain curves for the St. Venant-Kirchhoff

and the Hookean materials coincide with each other as they should; for finite deformations, there is significant difference in the nominal stress for the same strain. Also, for uniaxial deformations, the St. Venant-Kirchhoff material has different response in tension and compression, with an instability ensuing at a nominal strain of ~0.5.

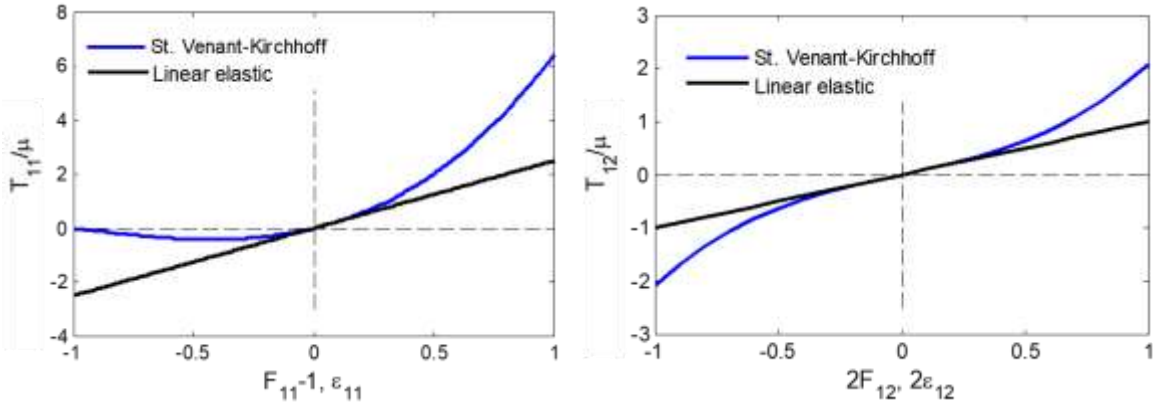


Figure 4.2 Response of a St. Venant-Kirchhoff and a linearly elastic material in uniaxial tension (left) and simple shear (right) deformations

Substitution from Eqn. (4.12) into Eqn. (4.11a), and for **T** into Eqns. (4.12) and (4.11) yields 12 coupled nonlinear partial differential equations. These 12 equations need to be solved for the 12 unknowns $u_{ij}(X_1, X_2, 0, t)$ that also satisfy the initial/boundary conditions stated as Eqn. (4.6). Eqn. (4.4) then provides the desired displacement field.

4.2.5 Boundary Conditions for the TSNDT

At the edge $X_1 = 0$, three boundary conditions used in the 3D linear elasticity theory and twelve in the TSNDT are listed below.

Simply supported (S): $u_2 = 0, u_3 = 0, T_{11} = 0$; $u_{2i} = 0, u_{3i} = 0, M_{11}^i = 0$

Clamped (C): $u_1 = 0$, $u_2 = 0$, $u_3 = 0$; $u_{1i} = 0$, $u_{2i} = 0$, $u_{3i} = 0$ (4.13)

Free (F): $T_{1i} = 0$; $M^i_{1k} = 0$

Because of our inability to analytically solve the afore-stated formulated initial-boundary-value problem, we find its approximate solution by the FEM.

4.3 Numerical Solution of the problem by the Finite Element Method

Referring the reader to *Appendix B* for details, we take the inner product of Eqn. (4.7) with the 12-dimensional test function $\phi_i^k, i = 1,2,3$ and $k = 0,1,2,3$, integrate the resulting scalar equation over plate's mid-surface, use the divergence theorem to transfer some surface integrals to line integrals, and get the weak formulation of the problem in which integrands have 1st order derivatives with respect to X_1 and X_2 of $u_{ij}(X_1, X_2, 0, t)$ and of ϕ_i^k, and 2nd order derivatives with respect to t of $u_{ij}(X_1, X_2, 0, t)$. We use a similar procedure, except for the use of the divergence theorem, to deduce initial and boundary conditions in terms of quantities defined on the plate mid-surface.

We discretize the plate mid-surface into four-node iso-parametric quadrilateral elements, and use the same FE basis functions, $\psi_k(X_1, X_2)$, $k = 1, 2, \ldots, N$, for $u_{ij}(X_1, X_2, 0, t)$ and ϕ_i^k to deduce the Galerkin approximation of the initial-boundary-value problem. That is,

$$\left.\begin{aligned} u_{ij}(X_1, X_2, t) &= \sum_{k=1}^{N} \psi_k(X_1, X_2)\, d_{ikj}(t) \\ \phi_i^k(X_1, X_2) &= \sum_{j=1}^{N} \psi_j(X_1, X_2)\, c_{ij}^k \end{aligned}\right\} \begin{matrix} i = 1,2,3 \\ j = 0,1,2,3 \end{matrix} \tag{4.14}$$

where c_{ij}^{k} are constants, and $d_{ikj}(t)$ are time-dependent DoFs of node *k*.

Substitution from Eqn. (4.14) into the weak formulation of the problem given by Eqn. (B.1-B.4) listed in Appendix B and requiring that the resulting equations hold for every c_{ij}^{k} gives the following system of 12*N* nonlinear ordinary differential equations (ODEs) for the 12*N* unknowns $d_{ikj}(t)$.

$$\mathbf{M}\ddot{\mathbf{d}}(t) = \mathbf{F}_{ext}(t) - \mathbf{F}_{\text{int}}(\mathbf{d}(t)) \tag{4.15}$$

Here **M** is the consistent mass matrix, $\mathbf{F}_{ext}$ represents nodal forces work equivalent to the external body force and surface tractions applied on plate's bounding surfaces, and $\mathbf{F}_{\text{int}}$ is the vector of nodal forces due to stresses developed in the plate at time *t*. The vector, $\mathbf{F}_{\text{int}}$, of internal forces at a given time is calculated by computing moments of stresses at that time, and substituting them in the right hand side of Eq. (B.1) – (B.4) in Appendix B. A flowchart of the algorithm for the nonlinear FEM is given in Appendix C

The system of 12*N* ODEs is numerically integrated by using the conditionally stable central difference method, i.e.,

$$\mathbf{d}_{n+1} = \mathbf{d}_n + \Delta t\dot{\mathbf{d}}_n + \frac{1}{2}\Delta t^2\ddot{\mathbf{d}}_n, \quad \ddot{\mathbf{d}}_{n+1} = \mathbf{M}^{-1}\left(\mathbf{F}_{n+1}^{ext} - \mathbf{F}_{n+1}^{\text{int}}(\mathbf{d}_{n+1})\right), \quad \dot{\mathbf{d}}_{n+1} = \dot{\mathbf{d}}_n + \frac{\Delta t}{2}\left(\ddot{\mathbf{d}}_n + \ddot{\mathbf{d}}_{n+1}\right) \tag{4.16}$$

where Δt is the time step size and $\mathbf{d}_{n+1} = \mathbf{d}(t_{n+1})$. For stability,

$$\Delta t \le \Delta t_{crit}, \Delta t_{crit} = \frac{2}{\omega_{\max}} \tag{4.16}$$

where $\omega_{\max}$ is the maximum frequency of free vibration involving infinitesimal displacements of the deformed plate at time *t*. Note that Δt_{crit} depends upon the current deformation of the plate, and the algorithm is implicit since we employ a consistent mass matrix. We have not experimented with using the mass matrix associated only with the translational degrees of freedom that can be diagonalized with either the row sum technique or the special lumping technique; e.g., see Hughes [23]. The solution accuracy can be improved by taking $\Delta t << \Delta t_{crit}$ that increases the computational time.

We solve *static problems* by using the mass scaling technique in which the mass density is reduced to a value of one millionth ($\rho_{mass\,scaling} = 10^{-6}\rho$) of the actual density of the plate material. As a consequence, waves travel at 1000 times the "real speed" and transients quickly die out due to interaction between incident and reflected waves. However, Δt_{crit} becomes extremely small and thus requires significantly higher computational cost.

4.3.1 Computation of Stresses

After computing $d_{ikj}(t)$, we find $u_{ij}(X_1, X_2, t)$, $u_i(X_1, X_2, X_3, t)$, strains, and stresses, respectively, from Eqns. (4.14), (4.4), (4.3), and (4.12). Whereas values of the in-plane stresses (T_{11}, T_{12}, T_{21}, T_{22}) are quite close to those found from the solution of the corresponding 3D problem, those of in-plane stresses (T_{13}, T_{23}) and transverse stresses (T_{31},

T_{32}, T_{33}) are not. Accordingly, we find the latter by using a one-step stress recovery scheme (SRS), i.e., using Eqn. (4.17) but do not modify T_{13} and T_{23}.

$$T_{3i} = \int_{-h/2}^{Z} \left(\rho \ddot{u}_i - T_{\alpha i,\alpha} \right) dX_3, \quad i = 1,2,3 \ \alpha = 1,2 \tag{4.17}$$

The vector $\mathbf{F}_{\text{int}}$ of nodal forces due to stresses developed in the plate at time t is determined from stresses without using the SRS scheme. We have not conducted numerical experiments to delineate improvements, if any, in results with the $\mathbf{F}_{\text{int}}$ found from the SRS stresses.

Alternatively, one could work with the symmetric Cauchy stress tensor, $\boldsymbol{\sigma}$. The expressions for surface tractions in terms of $\boldsymbol{\sigma}$ require unit normal to the deformed surface that may not be accurately computed for severely deformed plates because of finding $\boldsymbol{F}$ at surface points. We note that Hartman et al [109-110] worked in terms of $\boldsymbol{\sigma}$ to recover transverse stresses in a beam and a plate using the von Karman strains. Many authors do not distinguish between the reference and the current configurations when using the von Karman strains, as is done in linear elasticity.

Whereas for linear problems the SRS gave reasonably accurate values of σ_{13}, σ_{23} and σ_{33}, we have not succeeded in finding good values of σ_{33} (or T_{33}) for finite deformations. It could be due to their dependence upon $\sigma_{\alpha 3,\alpha}$ (α = 1, 2) that are not accurately found with a uniform FE mesh of 4-node rectangular elements.

4.4 Verification of the TSNDT software

4.4.1 Free Vibrations of Clamped Linearly Elastic Orthotropic Plate

Unless otherwise mentioned, results presented below have been found by using the 40 x 40 uniform FE mesh (20,172 DoFs) for the TSNDT, and the 80 x 80 x 8 uniform FE mesh of 8-node brick elements (177,147 DoFs) with 2 x 2 x 2 Gauss quadrature rule for the 3D linear elasticity theory (LET) using ABAQUS. These FE meshes provided the 10 lowest frequencies within an error of 1% with respect to their analytical values for a simply supported rectangular plate. For the TSNDT work, the 7-point Gauss quadrature rule is employed for integration in the X_3 – direction (e.g., see Eq. (4.8)), and the 2 x 2 Gauss quadrature rule for integrations in the X_1X_2-plane.

Batra et al. [111] have used different order plate theories to study wave propagation and free vibrations of a simply-supported orthotropic plate, Here we analyze free vibrations using the TSNDT of a 100 mm x 100 mm x 10 mm clamped orthotropic plate having ρ = 5000 kg/m^3 and following values of the moduli relative to the material principal directions: E_1 = 25 GPa, E_2 = 4.8 GPa, E_3 = 0.75 GPa, G_{12} = 1.36 GPa, G_{13} = 1.20 GPa, G_{23} = 0.46 GPa, $\nu_{12} = 0.036$, $\nu_{13} = 0.25$ and $\nu_{23} = 0.171$. The material principal axis 3 is aligned with the global x_3 - axis and the material principal axes 1 and 2 are at an angle of $+45^0$ (clockwise) to the global x_1 and x_2 axes. In Table 2 we have listed the ten lowest non-dimensional frequencies, $\bar{\omega}$, defined by

$$\bar{\omega} = 100\omega h\sqrt{\frac{\rho}{E_1}} \tag{4.19}$$

and computed using the TSNDT and the LET. It is evident that the two sets of frequencies are very close to each other with the maximum difference being 1.5%. The 8th frequency is for the in-plane mode of vibration and has null transverse displacement. Thus the present TSNDT software can accurately find the ten lowest frequencies.

Table 4.1 Comparison of the lowest 10 non-dimensional natural frequencies, $\bar{\omega} = 100\omega h\sqrt{(\rho / E_1)}$, for a clamped plate (a/h = 10) from the 3D LET and the TSNDT

Frequency	3D LET	TSNDT	Diff = $\left\|\frac{(\bar{\omega}_{3D} - \bar{\omega}_{TSNDT})}{\bar{\omega}_{3D}}\right\| \times 100$
1	1.88	1.89	0.74
2	3.03	3.05	0.92
3	3.57	3.60	0.81
4	4.17	4.21	1.03
5	5.00	5.06	1.14
6	5.36	5.42	1.24
7	5.46	5.51	0.95
8	**5.98**	**5.99**	**0.1**
9	6.25	6.33	1.31
10	6.57	6.67	1.50

4.4.2 Static infinitesimal deformations of beams/plates

Vlasov [67] and Pagano [68] have analytically analyzed static deformations of a simply-supported transversely isotropic beam having a sinusoidal load distribution on the top surface as depicted in Figure 4.3; Vel and Batra [61] solved a similar problem for a clamped

beam using the Eshelby-Stroh formalism. We set E_1 = 172 GPa, $E_2 = E_3$ = 6.9 GPa, $G_{12} = G_{13}$ = 3.4 GPa, G_{23} = 1.4 GPa and $\nu_{11} = \nu_{22} = \nu_{12}$ = 0.25. The plane strain deformations were simulated by setting u_2 = 0 for all nodes.

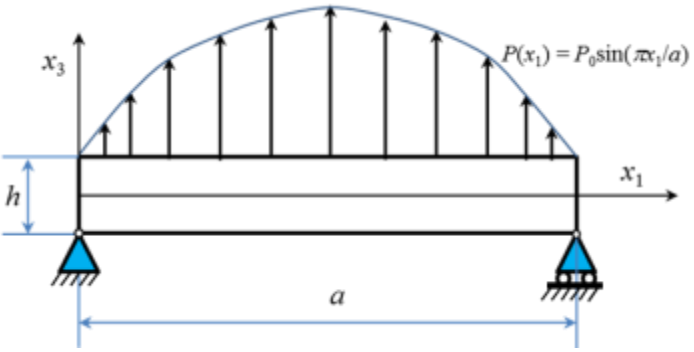


Figure 4.3 Schematic representation of a simply-supported orthotropic plate with a sinusoidal load on the top surface

From the results plotted in Figure 4.4 it is clear that the TSNDT software accurately computes the non-dimensional centroidal deflection.

$$u_3^n = \frac{100E_1h^3u_3}{P_0a^4} \tag{4.20}$$

for aspect ratio, (a/h), varying from 4 to 50. In Eqn. (4.20) P_0 is the amplitude of the sinusoidal loading. For a uniform FE mesh, the maximum value of the L^2 - error norm defined by

$$\|e\| = \sqrt{\frac{\sum_{i=1}^{N}\left(u_{exact}(i) - u_{TSNDT}(i)\right)^2}{\sum_{i=i}^{N}u_{exact}(i)^2}} \tag{4.21}$$

between the analytical solution and the TSNDT solution was computed to be 3.1%. In Eqn. (4.21), $u_{TSNDT}(i)$ equals the TSNDT deflection at node i.

In Figure 4.5 we have compared through-the-thickness distributions of the in-plane normal stress σ_{11} and the SRS-computed transverse shear stress σ_{13} for a beam with a/h = 4 and 10 and the in-plane stress σ_{11} for a beam with a/h = 10. We were unable to compare the σ_{13} for the a/h = 10 beam due to absence of results of this case in [68]

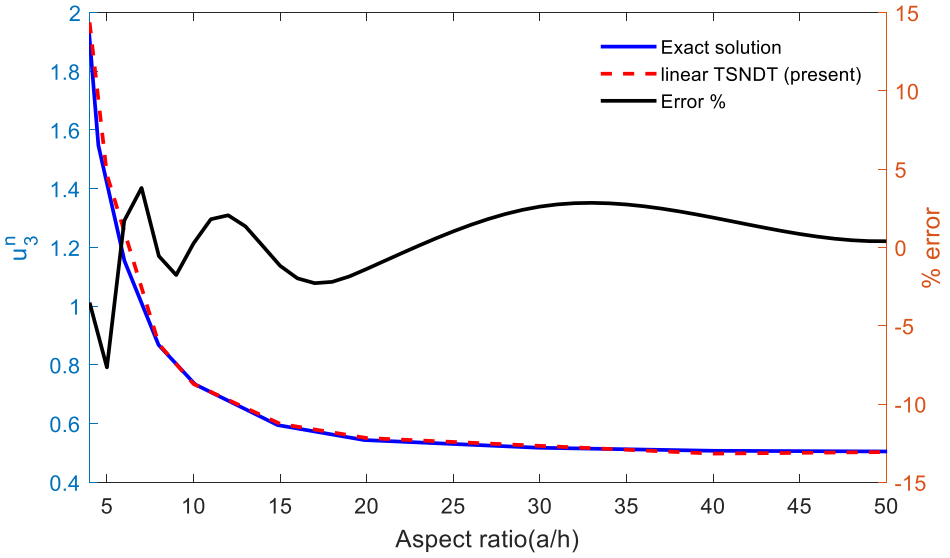


Figure 4.4 Comparison of the presently computed non-dimensional centroidal deflection of beams versus the aspect ratio (a/h) with the analytical solutions of Pagano [68]

The L^2-norm of errors for the TSNDT solutions exhibited in Figure 4.5 (a), (b) and (c) are 11.5%, 5.9 % and 1.7 %, respectively.

In order to show that the TSNDT gives good values of stresses near a clamped edge of a thick beam, we set P_0 = 0.5 GPa, a/h = 4, and compare in Figure 4.6 through-the-thickness distributions of σ_{11} and σ_{13} at four different sections from the TSNDT and the LET solutions. Near the mid-section, both stresses from the two theories are very close to each

other. However, near the clamped edge, x_1/a = 0.05, the TSNDT and the LET stresses qualitatively agree with each other but noticeably differ at locations away from the top and the bottom surfaces. We recall that for a/h = 4, the beam is quite thick and one may need a 5th order plate theory; e.g., see Vidoli et al. [7]. The L^2-error norms for σ_{11} at x_1/a = 0.1 and x_1/a = 0.5, respectively equal, 3.6% and 2.3%.

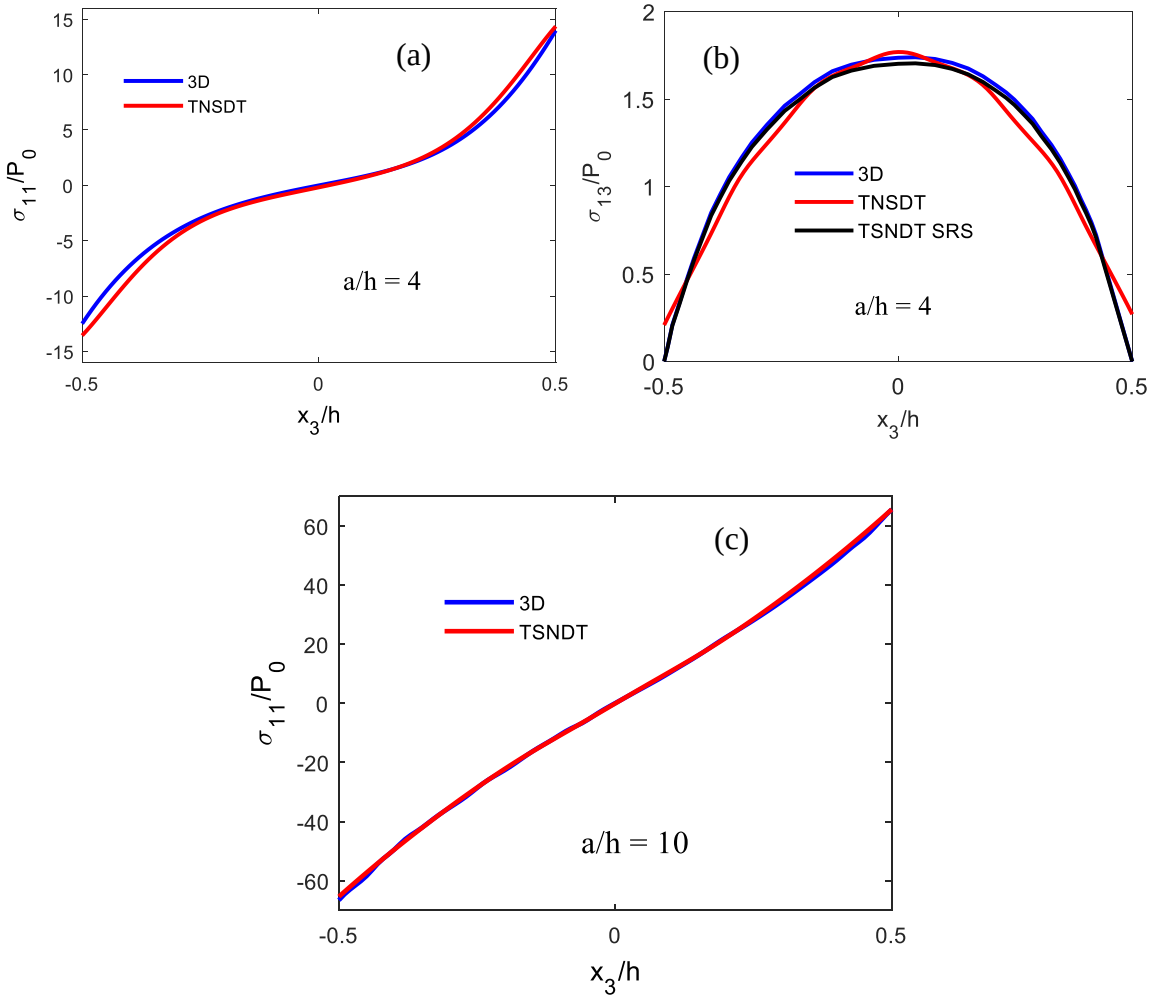


Figure 4.5 Through-the-thickness distributions of (a) σ_{11} and (b) σ_{13} from the TSNDT and the 3D analytical solutions [68] for a beam with span of 4 and (c) σ_{11} distribution for a beam with span of 10

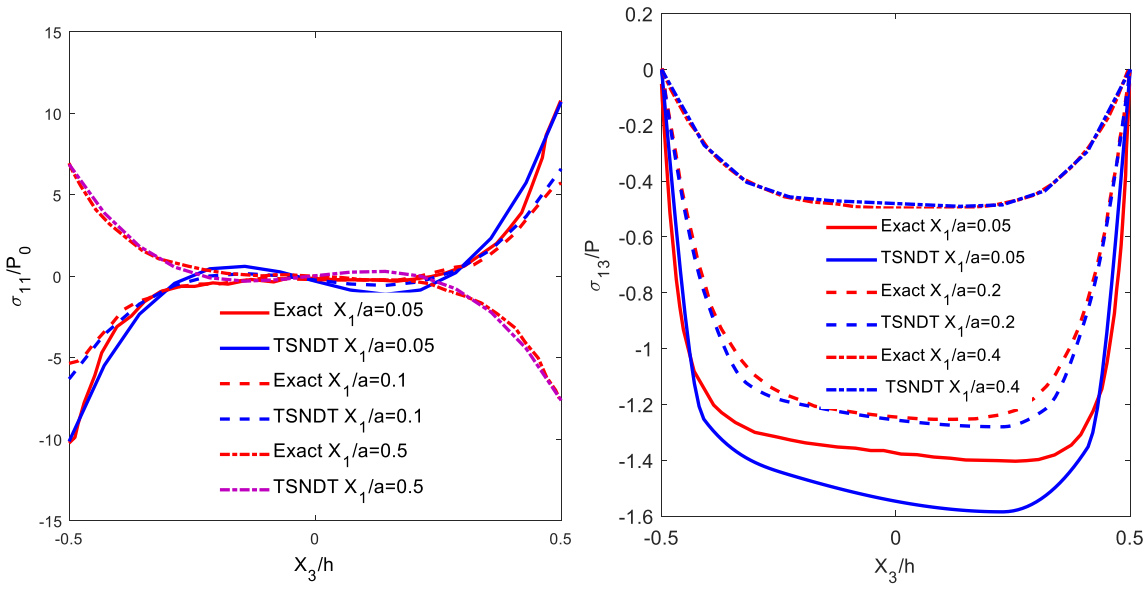


Figure 4.6 Comparison of through thickness distribution of stresses σ_{11} and the SRS found σ_{13} at three different sections of clamped-clamped orthotropic beam (a/h=4) from TSNDT and the exact solution [61].

4.4.3 Dependence of computed solution on the FE mesh

We now investigate the improvement, if any, in satisfying boundary conditions near a clamped edge by refining the FE mesh near the boundaries by employing the Chebyshev-Gauss-Lobatto (CGL) grid (e.g., see Tornabene et al[112]) given by

$$X_1^i = \left(1-\cos\left(\frac{1-i}{n_l-1}\pi\right)\right)\frac{l}{2},\ i=1,2,...,n_l,\quad X_1 \in [0,a]$$
$$X_2^i = \left(1-\cos\left(\frac{1-i}{n_b-1}\pi\right)\right)\frac{b}{2},\ i=1,2,...,n_b,\quad X_2 \in [0,b] \tag{4.22}$$

The uniform and the CGL-grid 40 x 40 FE meshes are depicted in Figure 4.7. In Table 4.2 we have listed, for different number of elements along the plate side, the centroidal deflection, the axial stress at two typical points and the transverse shear stress at a point near the plate edge. It is

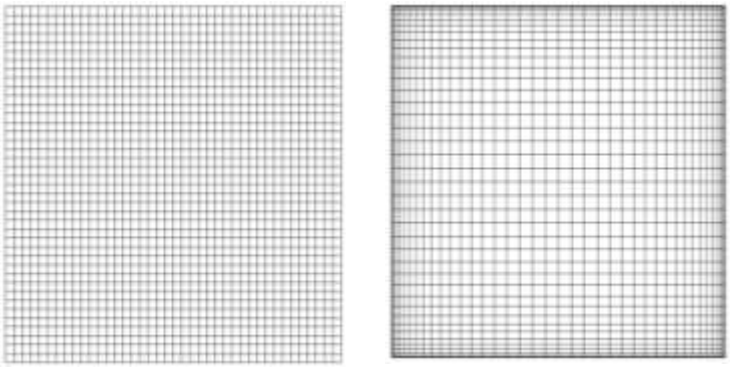


Figure 4.7 The uniform and non-uniform CGL grid used to study the mesh dependence of TSNDT solutions

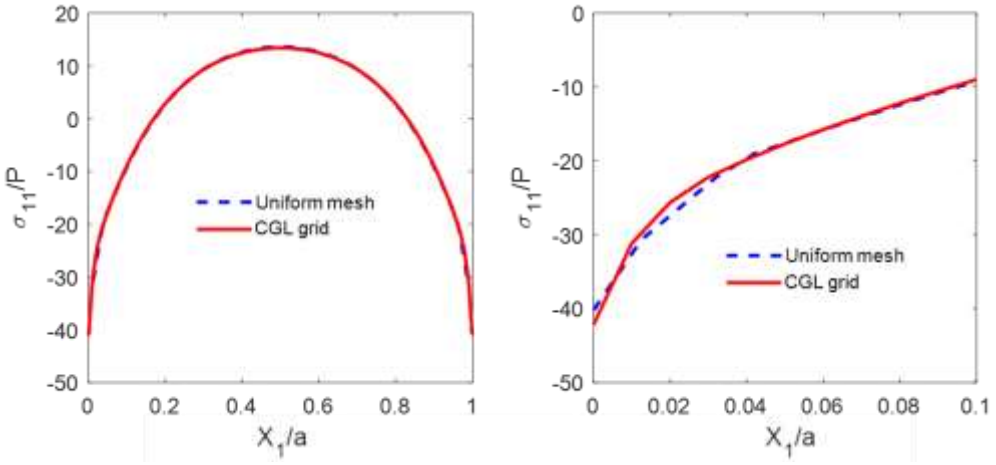
Table 4.2 Dependence upon the FE mesh of the centroidal deflection, the axial stress and the transverse shear stress for an isotropic linearly elastic 100 mm x 100 mm x 10 mm Clamped square plate (E = 25 GPa, $\nu = 0.25$) with uniform pressure P on the top surface

Uniform mesh								
No. of elements	u_3/h		σ_{11}/P				σ_{13}/P	
	At ($l/2$,$b/2$,0)	% change	At ($l/2$,$b/2$, $h/2$)	% change	At ($l/10$,$b/2$, $h/2$)	% change	At ($l/10$,$b/2$,0)	% change
25	2.83	-	6.68	-	-5.41	-	3.16	-
100	5.20	83.7	10.6	58.7	-8.61	59.1	3.74	18.3
225	5.80	11.5	12.4	16.9	-7.98	-7.31	4.49	20.1
400	6.13	5.7	12.9	4.03	-8.76	9.77	4.40	-2.01
900	6.43	4.9	13.4	3.88	-9.20	5.02	4.33	-1.59
1600	6.52	1.4	13.6	1.49	-9.24	0.4	4.35	0.46
2500	6.56	0.61	13.65	0.37	-9.25	0.11	4.37	0.46
3600	6.59	0.45	13.7	0.36	-9.26	0.11	4.37	0.02
Non-uniform mesh (CGL grid)								
No. of elements	u_3/h		σ_{11}/P				σ_{13}/P	
	At ($l/2$,$b/2$,0)	% change	At ($l/2$,$b/2$, $h/2$)	% change	At ($l/10$,$b/2$, $h/2$)	% change	At ($l/10$,$b/2$,0)	% change
25	2.44	-	4.66	-	-6.57	-	3.11	-
100	5.06	107.3	8.77	88.23	-6.48	-1.35	4.02	29.16
225	5.61	10.9	11.42	30.26	-7.77	19.74	4.25	5.80

400	6.14	9.44	12.12	6.11	-8.26	6.29	4.26	0.12
900	6.41	4.39	13.03	7.45	-8.79	6.47	4.32	1.56
1600	6.51	1.56	13.37	2.67	-8.99	2.34	4.35	0.62
2500	6.56	0.76	13.54	1.25	-9.09	1.10	4.36	0.29
3600	6.59	0.46	13.63	0.69	-9.15	0.60	4.37	0.15

clear that for both FE meshes, the centroidal deflection and stresses at the selected points converge as the mesh is refined. Stresses in the plate central region found from the two FE meshes are essentially identical but differ at points near the clamped edges as shown in Figure 4.8 wherein we have plotted variations of σ_{11} and σ_{13} along the lines $X_2 = b/2$, $X_3 = h/2$ and $X_2 = b/2$, $X_3 = 0$. The gradients of σ_{11} and σ_{13} near the clamped edge have higher values for the CGL grid than that for the uniform FE mesh; a similar variation of σ_{13} for a simply supported edge is exhibited in Figure 4.9.

Hereafter we use the uniform FE mesh since Δt_{crit} for the CGL grid is very small as compared to that for the uniform mesh.



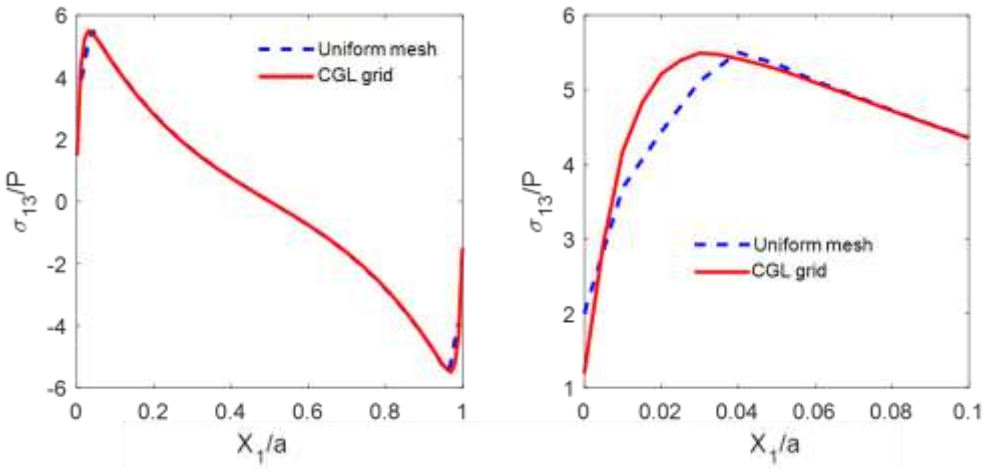


Figure 4.8 Variations of σ_{11} near the clamped edge along the line, $X_2=b/2$, $X_3=h/2$ and σ_{13} along the line, $X_2=b/2$, $X_3=0$, from the two FE meshes with figures on the right highlighting the boundary layer near the plate edge.

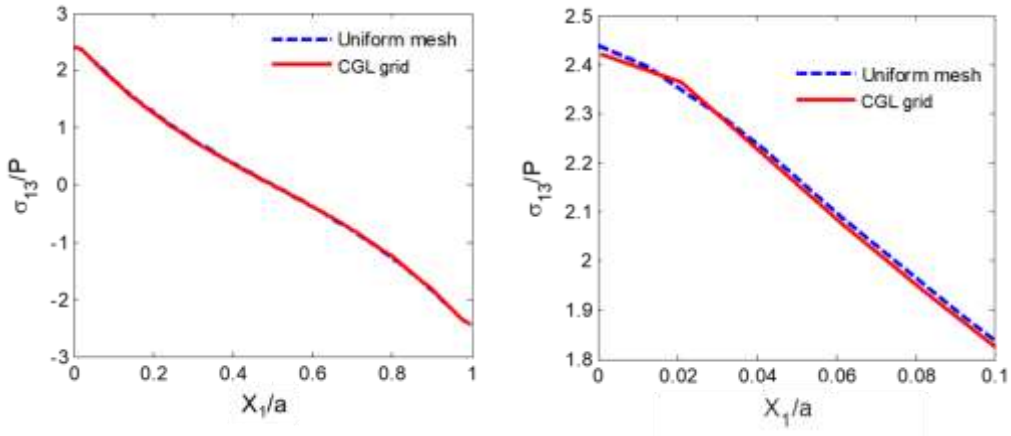


Figure 4.9 Variations of σ_{13} near a simply supported edge along the line $X_2=b/2$, $X_3=0$ from the two FE meshes

4.5 Example Problems

4.5.1 Transient deformations of linearly elastic plates

We analyze transient deformations of a clamped 100 mm x 100 mm x 10 mm plate with E = 25 GPa, ν = 0.25, ρ = 1.2 g/cc, and the uniformly distributed pressure on the top surface given by

$$P(t) = \begin{cases} P_0 \dfrac{t}{20}, & 0 < t \le 20\mu s \\ P_0\left(1 - \dfrac{1}{20}(t-20)\right), & 20\mu s \le t \le 40 \quad \mu s \\ 0, & t \ge 40\mu s \end{cases} \tag{4.23}$$

with P_0 = 0.5 GPa. The load increases linearly in time during the first 20 μs, then decreases affinely to zero in the next 20 μs, and subsequently stays at zero. The problem is also studied by assuming that the plate material is orthotropic with one material principal axis coincident with the X_3-axis and the other two making an angle of 30^0 counterclockwise with the X_1 and the X_2 axes. The moduli of the orthotropic material have values listed in subsection 4.4.1. In ABAQUS, we use the explicit algorithm with *automatic time increment* criteria for time increment step determination in ABAQUS explicit solver. For further details of this criteria, the reader is referred to in the section "Explicit dynamic analysis" in the ABAQUS user manual [20]. For the TSNDT, Δt_{crit} from Eqn. (4.17) equals approximately 1.2 μs (0.6 μs) for the isotropic (orthotropic) plate. We use Δt = 0.05 μs for both isotropic and orthotropic plates to compute a reasonably accurate solution at a higher CPU cost.

Time histories of the centroidal deflection, and of the axial stress σ_{11} at the centroid of the top surface are presented in Figure 4.10. For the first 100 μs of the isotropic (orthotropic) plate's deformations, the displacement and the stress histories from the TSNDT and the 3-D LET solutions differ at most by 3.4% and 4.6% (1.45% and 7%), respectively.

One significant difference between the TSNDT and the 3-D LET solutions is that for the former the applied load is instantaneously experienced by all plate points whereas for the latter a pressure wave travels from the top to the bottom surface that delays the load acting at points near the bottom surface. In order to clearly show this, we increase the plate thickness and the mass density, respectively, to 20 mm and 5 g/cc.

The through-the-thickness distribution at different times of the SRS-computed transverse normal stress, $\sigma_{33}(t)$, divided by $P(t)$, is presented in Figure 4.11, i.e., the ordinate at the top surface ($x_3/h = 0.5$) equals 1 in the plots. However, for the plots at 40 μs and 50 μs, the stresses are divided by the peak surface traction, P_0, as $P(t)$ equals zero at these times. The 1-D wave speed, $c = \sqrt{E/\rho}$, for the isotropic material is 2.23 mm/μs. Thus, the stress wave takes approximately 9μs to travel through the plate thickness. In the 3D LET solution, this time equals about $8\,\mu s$ because the dilatational wave speed is more than that given by $c = \sqrt{E/\rho}$. However, in the TSNDT solution, upon application of the load the stress instantaneously becomes nonzero throughout the plate.

We observe that subsequent to the time of the first reflection of the stress wave from the bottom surface, results of the 3D LET and the TSNDT match well until 20 μs (end time of

the monotonically increasing load). However, for $t > 20$ μs, the two solutions differ until the end of the computation period possibly due to the load changing instantaneously at all plate points in the TSNDT but not in the LET. Subsequent to *P(t)* becoming and staying at zero, the TSNDT and the LET results differ from each other because of a travelling σ_{33} wave in the 3D LET solution but in the TSNDT solution this effect is smeared out due to integration along the plate thickness.

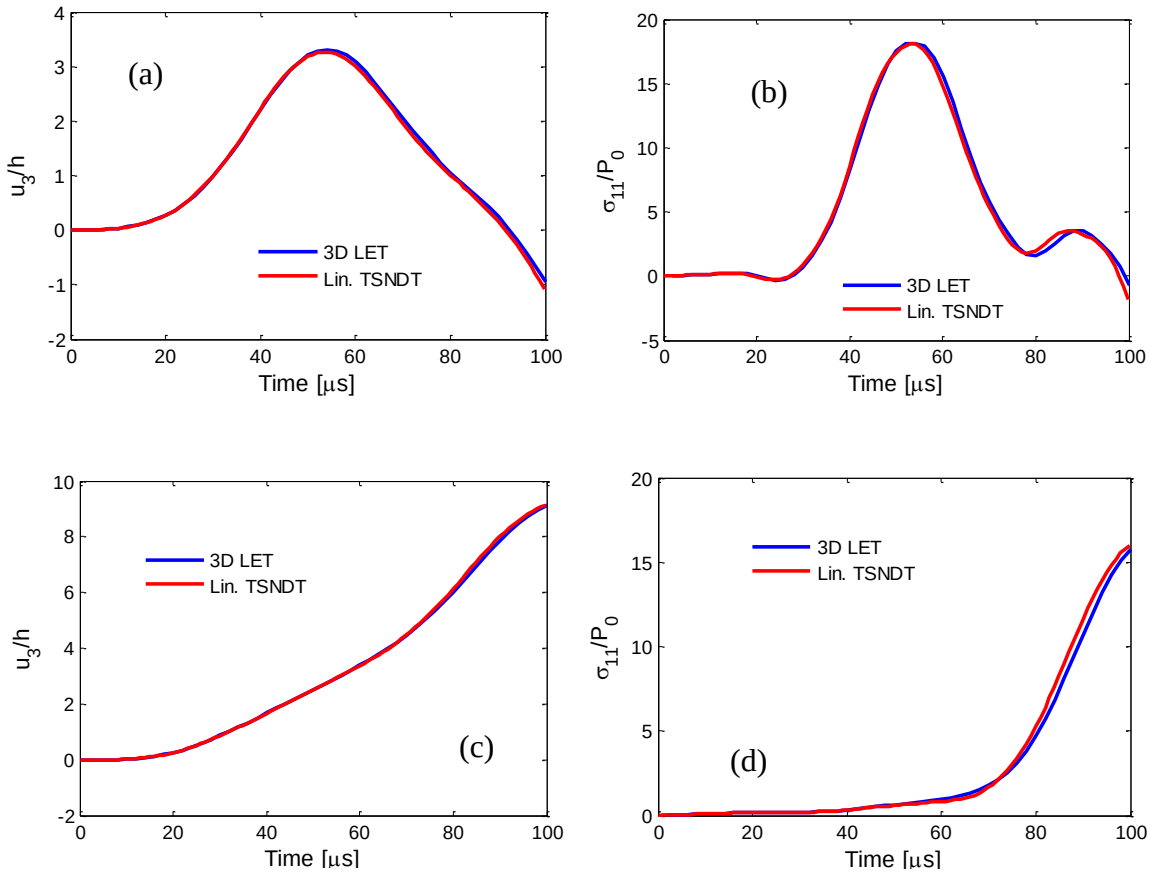


Figure 4.10 Time histories of (a), (c) centroidal deflection, and (b), (d) the axial in-plane stress σ_{11} at the centroid of the top surface of the isotropic and the orthotropic plates

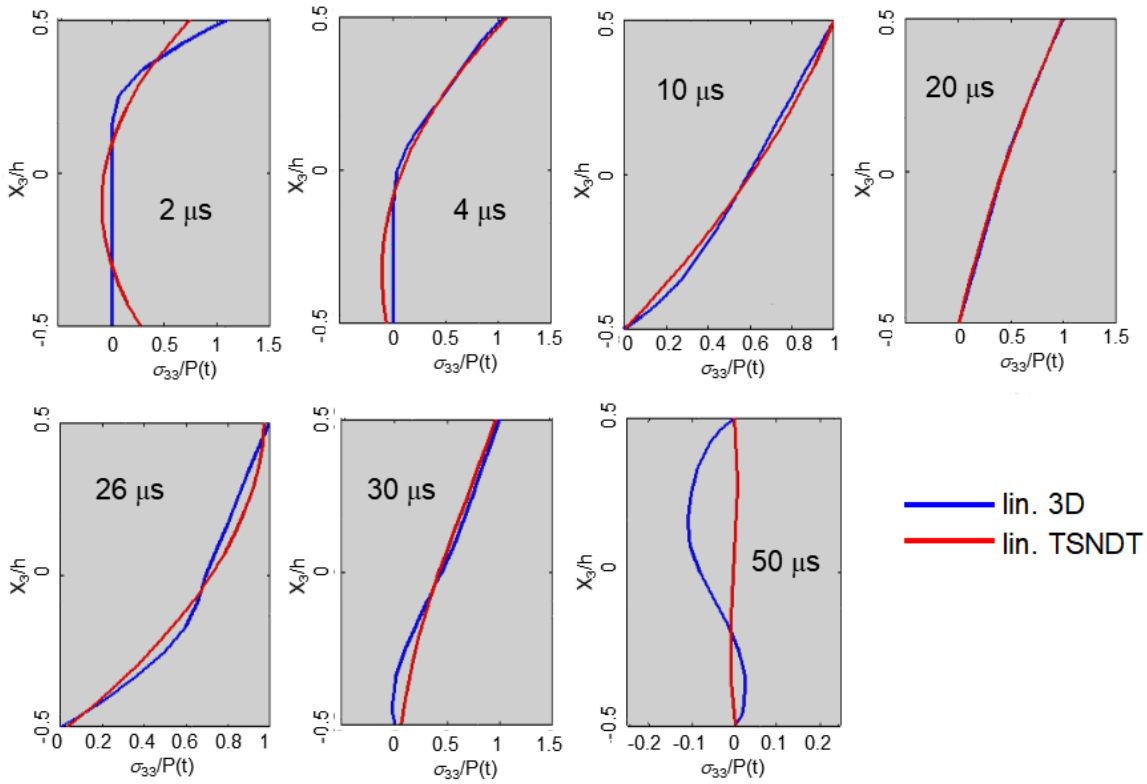


Figure 4.11 For the triangular time variation of the applied surface traction, through-the-thickness variations at different times of the transverse normal stress σ_{33} on the transverse normal passing through the plate centroid

Thus the TSNDT predictions for the displacements and the in-plane stresses agree well with those from the 3D LET except for deformations in the first few micro-seconds depending upon the plate thickness. These differences suggest that one should employ the 3D LET for shock wave problems with structures exposed to extreme loading.

The time histories of energies computed from the TSNDT solution and presented in Figure 4.12 show that the sum of the kinetic and the internal energies equals the work done by the external forces, and the numerical algorithm dissipates very little energy.

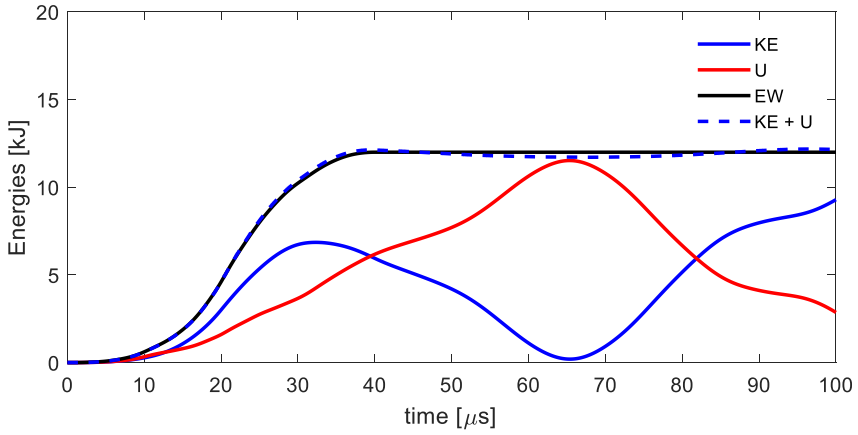


Figure 4.12 Time histories of energies and work done by external forces for the TSNDT solution; U, KE and EW, respectively, represent the total strain energy, the total kinetic energy and the work done by external forces.

The non-dimensionalized peak stress σ_{11}/P_0 and the out-of-plane displacement u_3/h at the centroid of the top surface of the isotropic plate were found to occur at approximately 65 μs equaling 5.792 and 0.55 respectively. It is clear that the peak deformations occur much after the load has been removed.

A comparison of results in Fig. 4.10 for isotropic and orthotropic plates reveals that the material symmetry significantly affects plate's response to the same load. Thus, for the same load, the response of an isotropic plate sheds no light on that of the geometrically identical orthotropic plate.

We also analyzed static deformations of the isotropic plate under identical boundary and loading conditions by reducing the mass density by a factor of 10^6. The aforementioned non-dimensional stress and displacement ratios at the centroid of the top surface were,

respectively, found to be 3.8632 and 0.29 suggesting magnification factors of 1.5 for σ_{11}/P_0 and 1.9 for the deflection.

4.5.2 Static deformations of rectangular St. Venant-Kirchhoff plates under dead and follower loads

We study finite static deformations of the clamped plate studied in subsection 4.5.1 but now consider material and geometric nonlinearities, and compare the TSNDT results with those found using the commercial FE software ABAQUS in which we have implemented an VUMAT for the St. Venant-Kirchhoff material. For the TSNDT analysis, we set ρ = 1.2 g/m^3, Δt = 0.025 nanosecond, employ the 40 x 40 uniform FE mesh to discretize the plate mid-surface, and linearly increase the uniformly distributed pressure on the top surface to 100 MPa. For the 3D analysis, we use 51,200 (80 x 80 x 8) 8-node brick elements corresponding to 177,147 DoFs with 2 x 2 x 2 Gauss integration rule in each FE. We analyze the nonlinear problem when the applied pressure is a dead load (i.e., its magnitude per unit undeformed area and direction remain unchanged) and it is a follower load (i.e., its magnitude per unit reference area remains constant but it always points along the normal to the deformed surface). That is, for pressure P_0 per unit undeformed area of the plate top surface, the traction boundary conditions, respectively, corresponding to the dead and the follower loads are

$$T_{Li}N_L = P_0\delta_{i3},$$

$$T_{Li} = pJF^{-1}_{iL} \tag{4.24}$$

Here **N** (= (0, 0, 1)) is the unit normal vector to the top surface in the reference configuration that is deformed into the vector along F_{i3}. In a linear problem, there is no distinction between the dead and the follower loads as the undeformed and the deformed configurations are assumed to be very close to each other.

For P_0 = 0.3 GPa, the plate deflection on the line $X_2 = b/2$, $X_3 = 0$ from the TSNDT and the 3D analyses are presented in Figure 4.13. It is clear that at every point on the line, the deflection predicted by the nonlinear theory is less than that found using the linear theory. Whereas, from the comparison of through-the-thickness distributions shown in Figure 4.14, it can be observed that σ_{11} for the linear theory is symmetric about the plate mid-surface, but that from the non-linear theory is not. This is partially due to the asymmetric response of the St. Venant-Kirchhoff material in uniaxial tensile and compressive deformations; cf. Figure 4.2 (left). For the nonlinear problem, the neutral surface defined by requiring the net axial force in the X_1- and the X_2- directions to be zero need not coincide with the plate mid-surface.

The L^2-norm of the difference between displacements, in-plane axial stress and transverse shear stress for the linear TSNDT and 3D models are 1.6%, 1.2% and 1.6% respectively. The corresponding values from the nonlinear and the nonlinear analyses are 1.3 %, 1.1% and 7.8% respectively. That is, the in-plane normal stress predictions from the nonlinear TSNDT and 3D solutions agree well with each other. However, the transverse shear stress has a larger deviation for the nonlinear problem.

It should be clear from plots of the total strain and the total kinetic energies (KEs) depicted in Figure 4.15 that the mass scaling indeed gives very small value of the KE relative to the

total strain energy of the body and the analyses closely approximate static deformations of the nonlinear plate. The maximum difference between energies from the TSNDT and the 3D elasticity solutions is 5%.

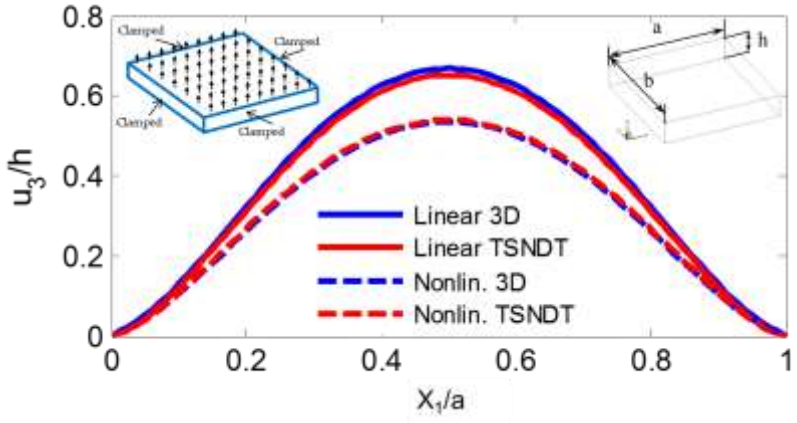


Figure 4.13 Comparison of the deflection of points on the line, $X_2 = b/2$, $X_3 = 0$, of the clamped plate obtained from the TSNDT and the 3D analyses for the linear and the nonlinear problems

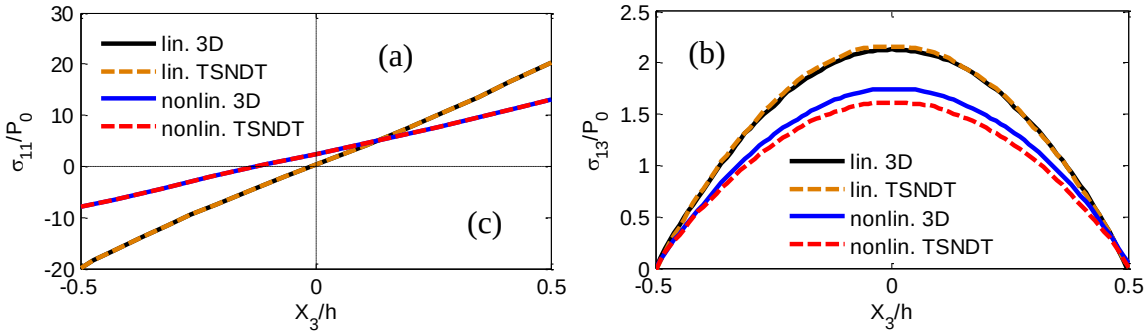


Figure 4.14 Comparison of through thickness distribution of (a) σ_{11} and (b) σ_{13} at (l/4,b/2) from linear and nonlinear 3D elasticity and TSNDT

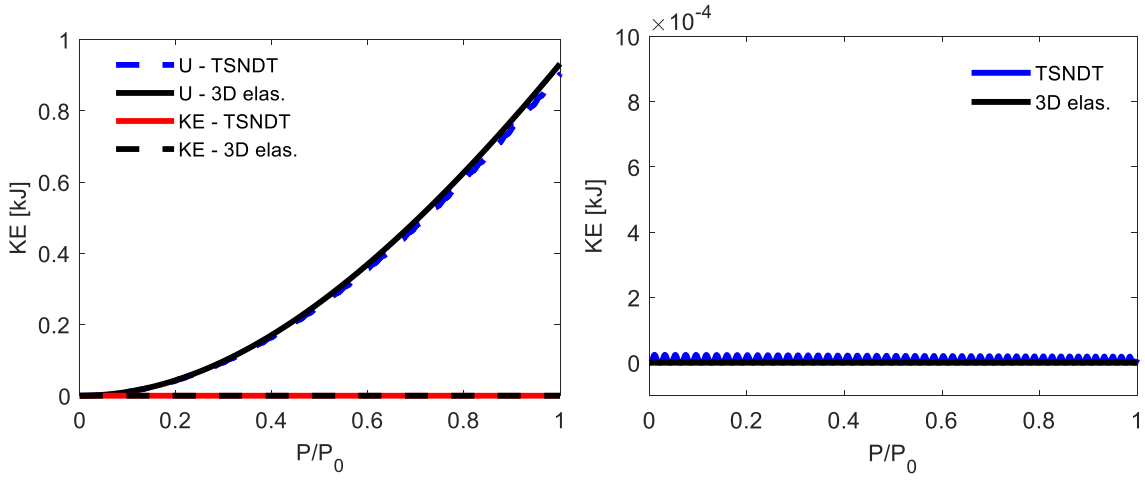


Figure 4.15 Total kinetic and strain energies (KE and U respectively) for finite deformations of the plate computed using the TSNDT and the 3D theories and zoomed in plot of KEs from TSNDT and 3D elasticity

For deformations of the cantilever beam analyzed by Xiao and Batra [97] using the layerwise TSNDT (plate thickness divided into 3 layers) using both linear and nonlinear theories, we have exhibited in Figure 4.16 the through-the-thickness distributions of the in-plane and the transverse shear stresses. The discrepancies between the two TSNDT results are attributed to the use of a layerwise theory in [97] and a single layer in the present work; the former having considerably more degrees of freedom than the latter. We observe that for the nonlinear analysis σ_{13} is non-zero at both the top and the bottom surfaces (the follower type normal traction is applied on the bottom surface). It is at least partially due to the curvature of the deformed surfaces that requires that a linear combination of σ_{13} and σ_{33} should vanish at a point.

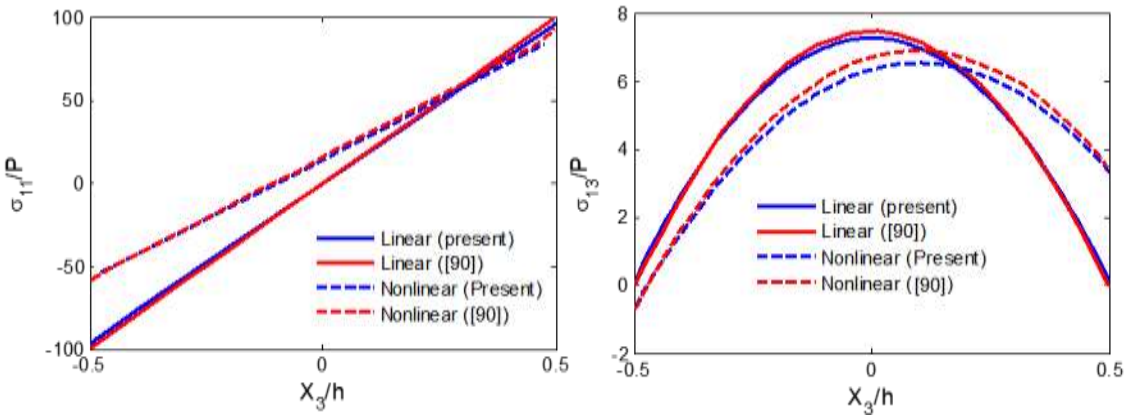


Figure 4.16 Through thickness distributions of (a) σ_{11} at the centroid of the beam and (b) σ_{13} at $X_1 = a/4$ [97]

4.5.3 Transient deformations of non-linearly elastic plates

For the plate studied in subsection 4.5.1, we now assign ρ = 1.2 g/cc, and analyze its finite deformations for the uniform follower pressure on the top surface given by Eqn. (4.23) with peak pressure P_0 of 1 GPa but equal rise and fall times of 200 μs. We set $\Delta t = 0.05$ μs ($\simeq \Delta t_{crit}/10$) for both the TSNDT and the 3D solutions. Time histories of the centroidal displacement and the in-plane stress at the centroid of the top surface are presented in Figure 4.17 and those of the total strain energy, the total kinetic energy and the work done by external forces in Figure 4.18. The L^2-norm of the relative error between the TSNDT and the 3D solutions for the plotted quantities is less than 5%. It can be observed that although the centroid of the plate continues to vibrate with significant amplitude post removal of the load, the diminishing of the energy suggests that the scale of deformation of the whole plate reduces significantly after the load is removed. We show this in Figure 4.19 where distributions of σ_{11} along the line $X_2 = b/2$ on the top surface of the plate are compared for times 200 μs and 400 μs (peak load time and load removal time respectively).

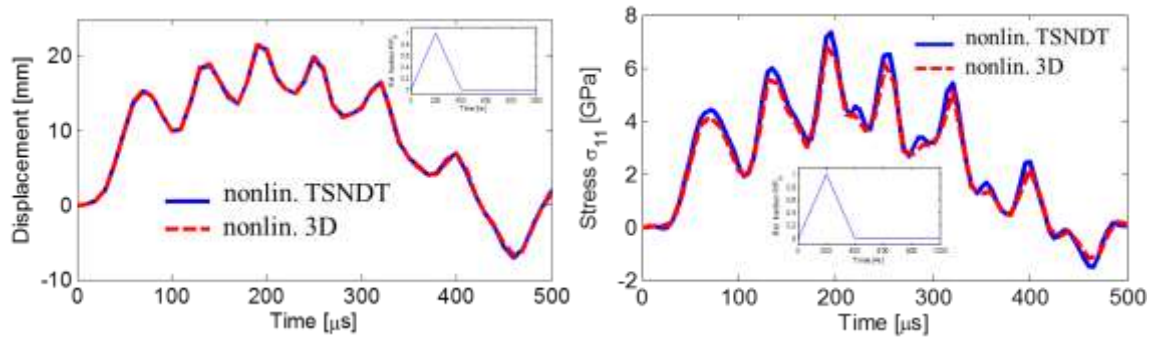


Figure 4.17 Time histories of the centroidal out-of-plane deflection and of σ_{11} at the centroid of the top face of the 100 mm x 100 mm x 10 mm clamped plate with a uniform pressure of 1 GPa applied to the top surface of the plate

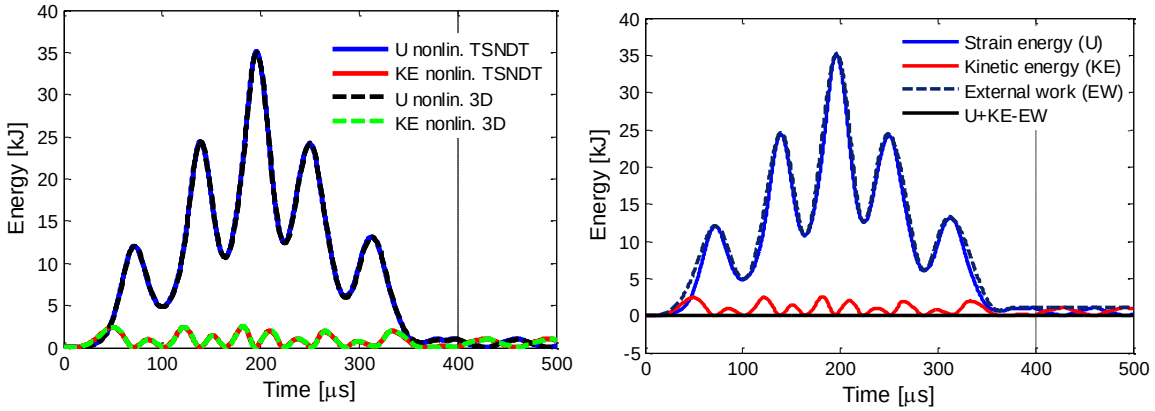


Figure 4.18 Time histories of energies from the TSNDT and the 3D solutions (left), and comparison of the total energy and the work done by external forces from the TSNDT solution (right)

This shows that although the values of σ_{11} at the centroid of the top surface of the plate at 200 μs and 400 μs have a ratio of approximately 3:1 as is seen in Figure 4.17, the whole plate undergoes much lower magnitude of deformation at 400 μs thus resulting in significantly smaller total strain energy at 400 μs as is observed in Figure 4.18.

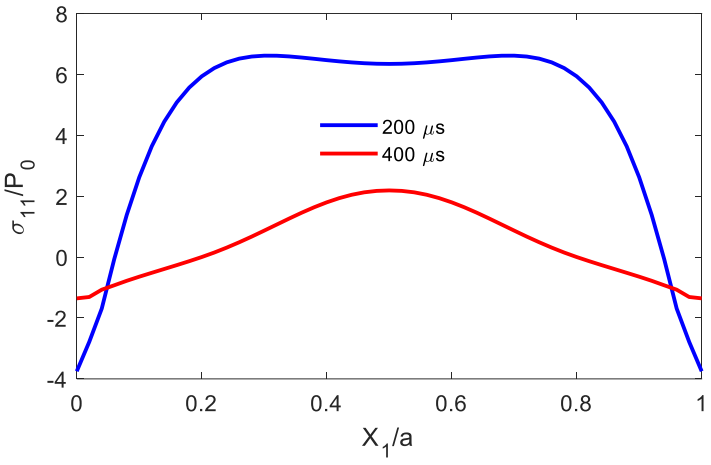


Figure 4.19 Distribution of the in-plane axial stress σ_{11} along the line $X_2 = b/2$ on the top surface of the plate at times 200 μs and 400 μs

In order to delineate effects of nonlinearities, we present time histories in Figure 4.20 for peak pressures of 10 MPa, 100 MPa, 500 MPa and 1 GPa, and also analyze deformations of a spring-mass system for linear and nonlinear springs described by Eq. (4.25).

$$f\left(x\right) = -\left(k_1 u + k_2 x\left|u\right|\right) \tag{4.25}$$

Here k_1 and k_2 are the linear and the nonlinear spring stiffnesses, and u equals the spring stretch. The equation of motion of the spring mass system is

$$m\ddot{x} + k_1 u + k_2 u\left|u\right| = F\left(t\right) \tag{4.26}$$

where $F(t)$ is the loading function. For initial conditions, $x\left(0\right) = \dot{x}\left(0\right) = 0$, and taking m = 1 kg, k_1 = 1 kN/mm, k_2 = 10 kN/mm^2 , F (t) in the form of a triangular pulse with the rise and fall times of 0.2 s and different peak values of 1 N, 10 N, 100 N and 1 kN, time histories of u found using the Runge-Kutta algorithm in MATLAB [113] are shown in Figure 4.20.

It is observed that with an increase in the applied peak pressure, the non-dimensional displacement of the spring-mass system varies in the same way as the plate centroidal deflection. It suggests that the nonlinear problem has smaller deformations than the corresponding linear one. This can be explained by the fact that the stiffness of the spring mass system is a function of the displacement and with the increase of peak loads, higher displacements result in a stiffer system. Also, it is noted that the behavior of the spring mass system under F_0 = 100 N has a large amplitude of vibration after the removal of the load and is anomalous as compared to the other loading cases. We hypothesize that this is due to the *vibration attenuation* phenomenon explained in the 2nd chapter. For the nonlinear spring-mass system considered in the present discussion, if k_2 is 0, the values of rise and fall times of the load for vibration attenuation to be exhibited is found to be 0.19s when n is set as 1 in Eq. (2.5). When the peak load F_0 = 1 N, nonlinearity of the system has minimal effect due to small values of displacements. Thus the nonlinear system behaves linearly exhibiting vibration attenuation. This is seen in the displacement history of F_0 = 1 N in Figure 4.20 represented by the black curve. However, as the value of peak load is increased, effect of the nonlinearity is more pronounced in the system resulting in deviation from a linear system exhibiting *vibration attenuation*. The case of F_0 = 100 N is analogous to the result presented in Figure 2.2 (d) where we observed a large amplitude of vibration in the linear spring-mass system after the load removal due to significant deviation of the loading time period from the *vibration attenuation* causing loading time period. For the case of F_0 = 500 *N*, the nonlinearity results in the system behaving under a load with rise and fall times of the load closer to the case of n = 2 in Eq. (2.5) thus resulting in amplitude of vibration smaller that the F_0 = 100 N case.

4.5.4 *Finite transient deformations of a rectangular plate under follower and dead loads*

We now study deformations of a clamped 220 mm x 220 mm x 22 mm St. Venant Kirchhoff plate loaded by a uniform tensile pressure on the top surface whose variation with respect to time is given by Eqn. (4.23) with rise and fall time of $200\mu s$ and P_0 = 0.5 GPa. A uniform 40 x 40 FE mesh is used for the TSNDT FEM, with Δt = $0.05\mu s$. For P_0 = 0.5 and 1.0 GPa, Figure 4.21 and Figure 4.22, respectively, exhibit time histories of the deflection at two points, (l/2, b/2, 0) and (l/4, b/4, 0). Whereas the difference in results for the dead and the follower loads are negligible for P_0 = 0.5 GPa, they are noticeable for P_0 = 1 GPa. Peak values of σ_{11} in the plate with P_0 = 1 GPa under the follower and dead load were found to be 7.35 GPa and 5.85 GPa, occurring at 110 μs and 120 μs respectively.

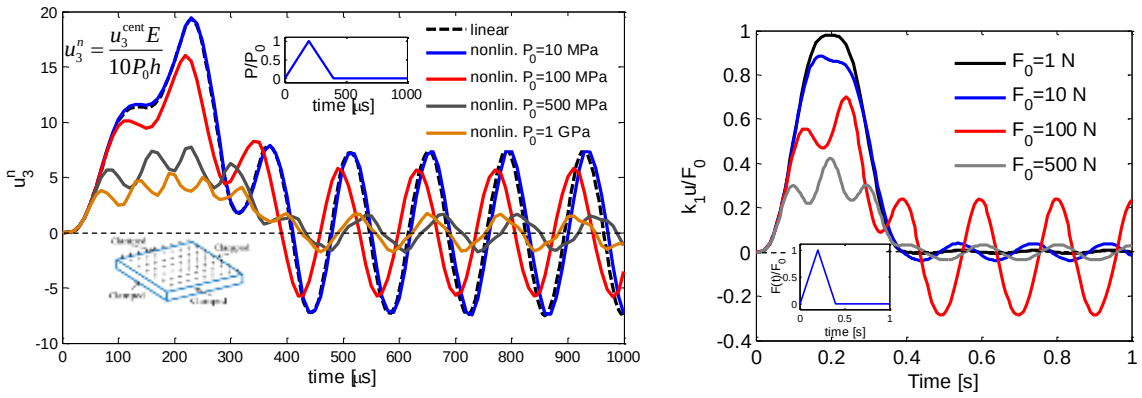


Figure 4.20 Time history of the non-dimensional centroidal deflection for the CCCC plate under different values of peak loads (left), and of the non-dimensional displacement of the mass in a single DoF spring mass system with a nonlinear spring (right)

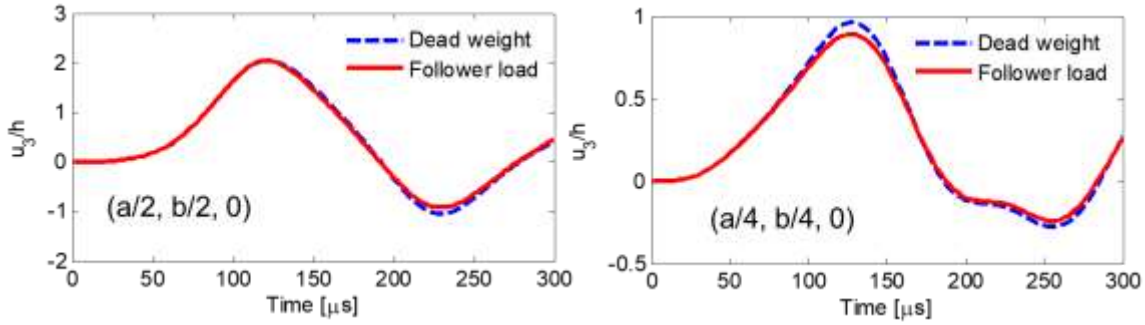


Figure 4.21. Time histories of the deflection at two points on the mid-surface of the plate for dead and follower loads having the peak value of 0.5 GPa.

4.5.5 Transient deformations of plates with pressure applied on the central region of a clamped rectangular plate

Clamped Rectangular plate

We set plate dimensions = 100 mm x 25 mm x 2.5 mm, E = 25 GPa, $\nu = 0.25$, $\rho = 5g/cc$, region loaded = $40mm \le X_1 \le 60mm$, $0 < X_2 < 25mm$, time variation = a triangular pulse of rise and fall times of $20\,\mu s$, and load type = uniformly distributed pressure. Plots presented in Figure 4.23 of the mid-surface deflection at t = 20, 40, 60, 80, 100 and 200 μs reveal that for $0 < t < 40\ \mu s$ when the external load acts on the plate, the linear and the nonlinear theories give identical profiles for the thin plate. However, after the removal of the load, the two profiles differ and this difference increases with time.

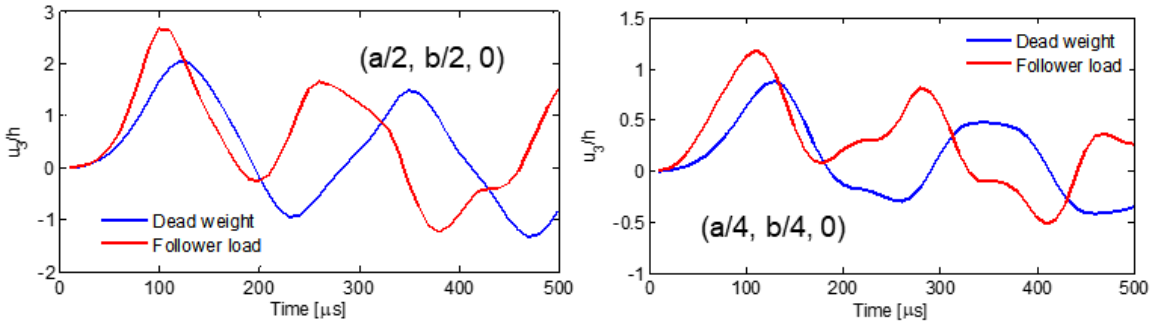


Figure 4.22. Time histories of the deflection at two points on the mid-surface of the plate for dead and follower loads having the peak value of 1 GPa.

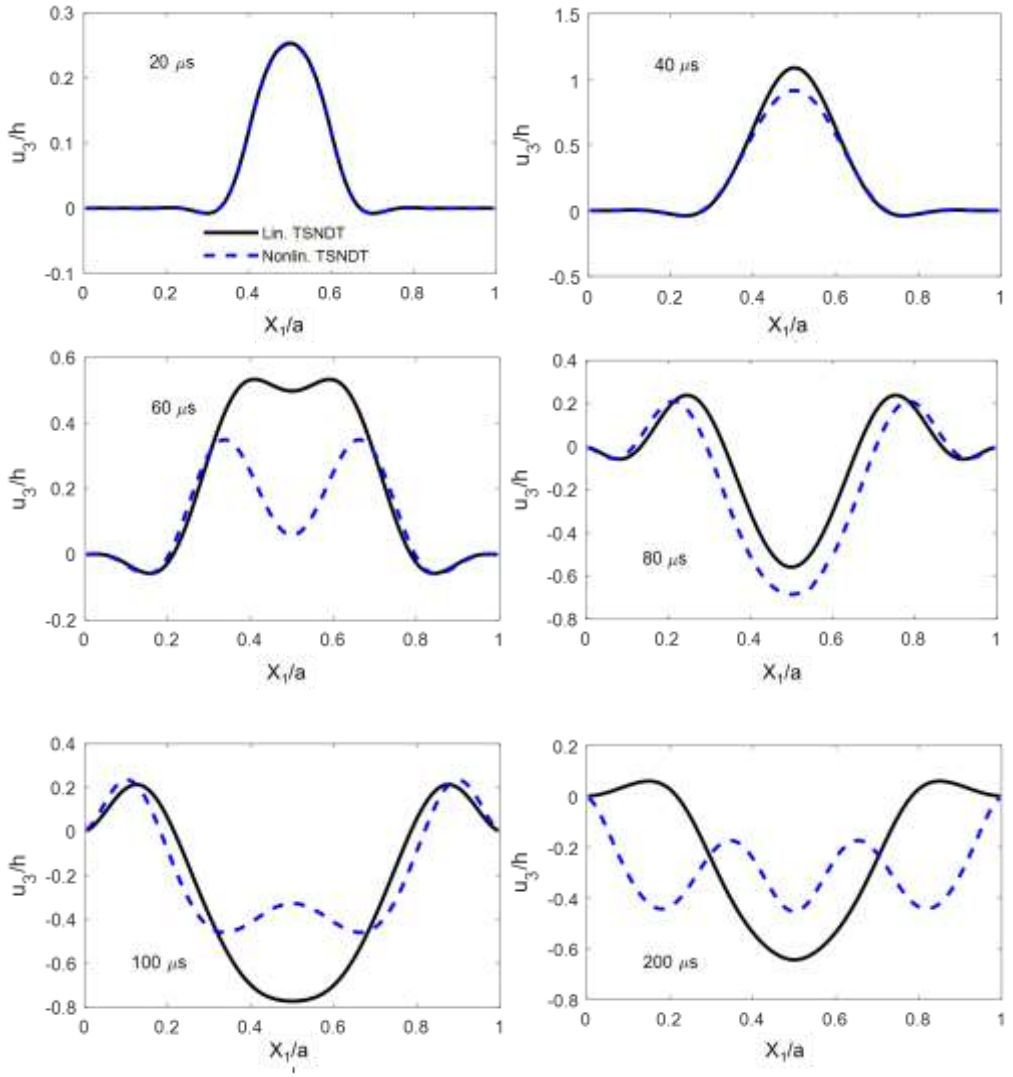


Figure 4.23 Deformation of the mid-surface along the span of the plate at different times from the linear and nonlinear analysis represented by solid red and dotted blue lines respectively

4.5.6 Isotropic and orthotropic cantilever plate deformed under tangential traction on the top surface

For isotropic and orthotropic St. Venant-Kirchhoff plates studied in subsection 4.5.3, we now analyze deformations of the plate when the left edge is clamped with the other three edges traction free, and a tangential traction with time variation described by Eq. (4.23), with equal rise and fall times of $200\mu s$. The deformed right edge surfaces of the plates at different times are presented in Figure 4.25. It is observed that the cross section of the orthotropic plate undergoes distortion about its center while that of the isotropic plate does not exhibit. Time histories of the in-plane and the out-of-plane displacements of the points (*l*, 0, 0) and (*l*, *b*, 0) for the two plates presented in Figure 4.26 clearly evince the difference in the rotations of the cross-section for the two plates.

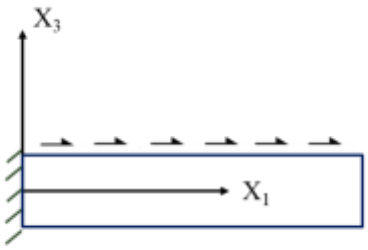
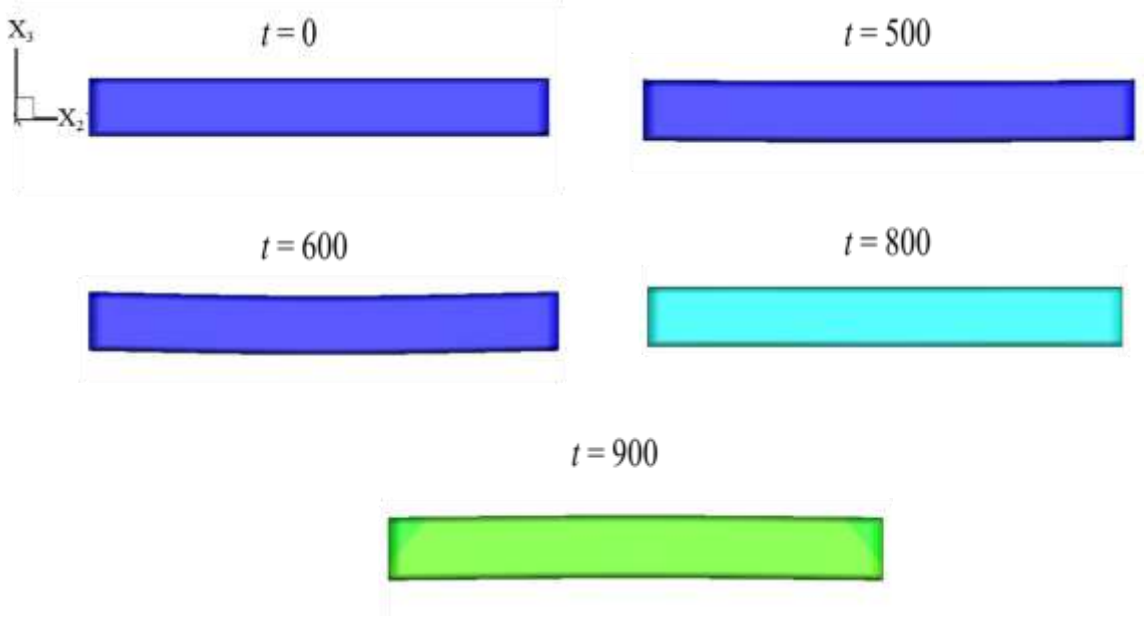
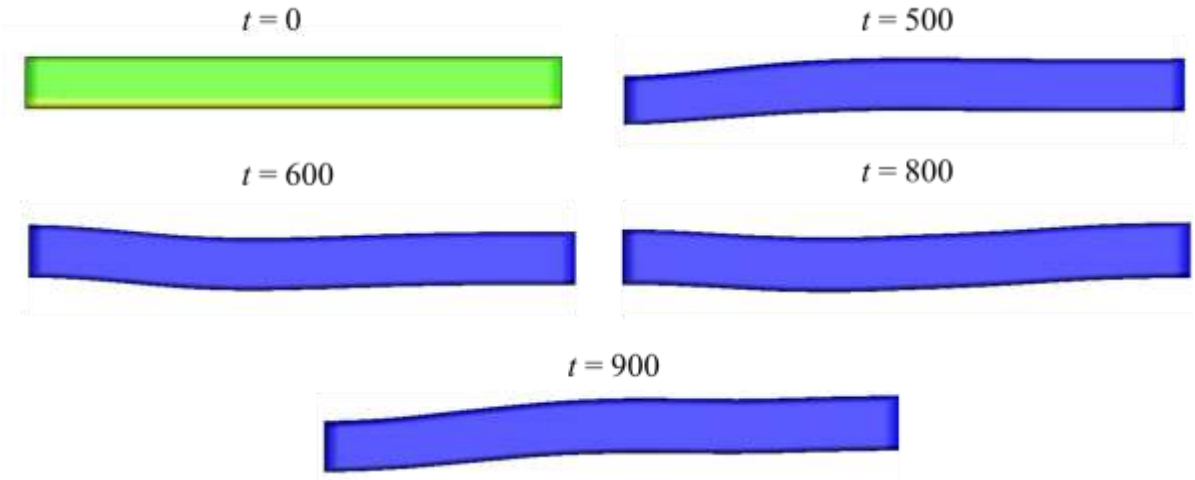


Figure 4.24 A cantilever plate under tangential traction

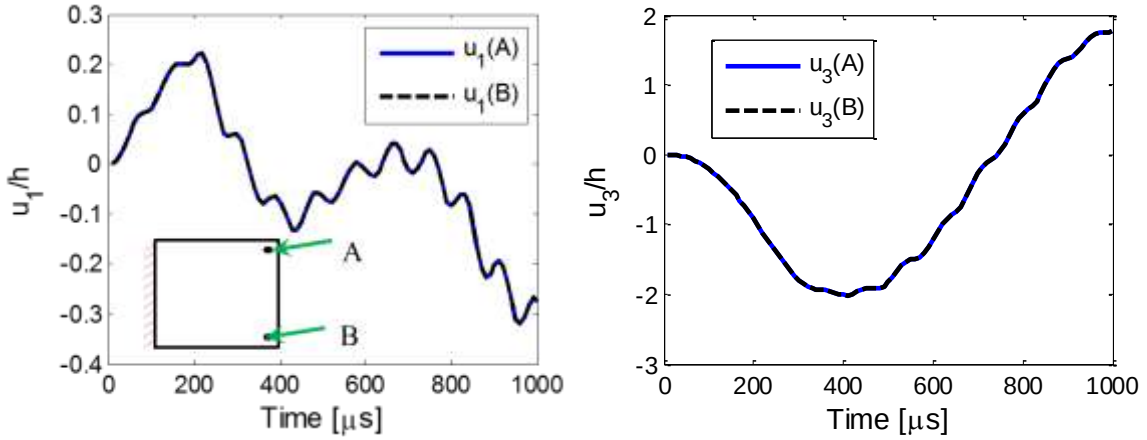


(a)

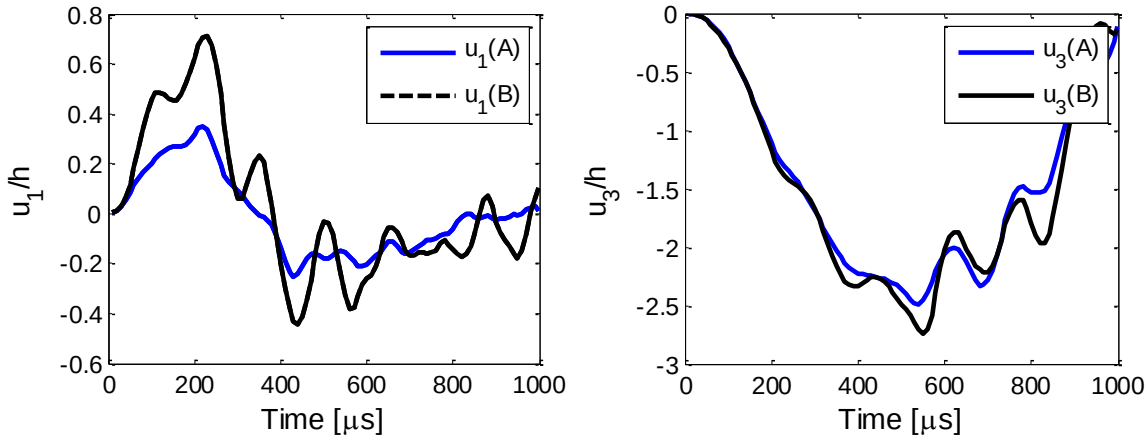


(b)

Figure 4.25 The deformed shapes of the edge $X_1 = l$ at different times for the (a) isotropic plate and (b) orthotropic plate (all times are in μs)



(a)



(b)

Figure 4.26 Time histories of the in-plane and the out-of-plane displacements (u_1 and u_3 respectively) at points A and B for the (a) isotropic and (b) orthotropic plates

4.6 Conclusions

We have developed a nonlinear mathematical model to study the infinitesimal and finite deformations of bending plates under different boundary conditions. We solved the system of equations numerically by developing an in-house FEM software. The developed software was verified by comparisons of obtained results to those from the 3D FEM

solutions and from the existing literature which were found to be in good agreement. Based on this, it is said that TSNDT is capable of predicting the transient response of linearly elastic and nonlinearly elastic plates to good accuracy. We also delineate the effects of introducing material and geometric nonlinearity in the problem and conclude that the linear theory over-predicts the deformation of the plate under a given loading and boundary condition when compared to the results of nonlinear theory. This may result in over conservative design criteria when based on the linear models. We also observe that unlike the linear theory, modeling the plate bending with nonlinear deformations results in a neutral surface different from the mid-surface of the plate pertaining to the difference in the tensile and compressive behavior of the St. Venant-Kirchhoff plate. We study a case of propagation of the in-plane wave in a linear and a nonlinear plate under a triangular pulse load acting on a partial area the top surface of the plate. From the numerical results, we observe the propagation of the in-plane wave after the removal of the load and establish that the speed of the propagation of the in-plane wave in the linear case differs from the wave speed in the nonlinear problem.

5 Conclusions

We have developed a computationally efficient validated FEM program based on the TSNDT which is capable of studying free vibrations and infinitesimal/finite static and transient deformations of linearly/nonlinearly elastic plates under a variety of boundary and loading conditions. We have established that the main advantage of using this software was using significantly lower number of DoFs as compared to 3D elasticity FEM simulations while maintaining good accuracy as compared to the 3D elasticity solutions. We have studied three different mechanics problems pertaining to plates using the developed software which enabled us to better understand the mechanics of vibrating plates. We briefly conclude the findings of these studies below. Detailed conclusions of each of the problem studied can be found at the end of the respective chapters.

The first problem we studied was called *vibration attenuation* where we observed that a simple spring-mass system undergoing transient vibrations with an acting triapezoidal pulse load comes to rest after the removal of the load when the loading and unloading time periods are multiples of the time period of vibration of the spring mass system. We extended this observation to continuous bodies by studying the same phenomenon in rectangular and circular isotropic, orthotropic and functionally graded monolithic/composite plates acted upon by either uniform or non-uniform pressures on their top surface. We observed that as long as the transient deformations of a plate are dominated by bending modes, the vibration attenuation phenomenon is also exhibited in plates. We performed a modal analysis of such plates and found that the fundamental bending mode contributes to majority of the energy of transient deformation .This is an

interesting observation as vibrations of an undamped system can be brought to rest without the use of any external dampers.

The second problem we studied is about the phenomenon of *mode localization*, where, an internal irregularity or disturbance in a structure results in localization of deformations in a part of the structure. We investigated this phenomenon in free vibrations of rectangular isotropic/orthotropic plates using points clamped within the volume of the plate to introduce intrinsic irregularity and observed mode localization of free vibrations in rectangular plates. New findings from this work included mode localization of in-plane modes of vibration and localization in multiple sub-sections of the plate due to the presence of multiple internal clamped points. We also studied the effects of localization in transient problems. We found that under a uniform pressure load applied over the whole surface of the plate, the presence of one internal clamped point divides the plate into two sections vibrating at different frequencies. The frequency of the vibration of each section was found to be close to the frequency of the corresponding mode of free vibration in which the vibration mode was localized in that particular section. We also studied the effects of harmonic excitation on an internal clamped plate and observed that due to the interaction of the two sections of the plate around the internal clamped point, the vibration behavior of the plate exhibits *beats* phenomenon, which we hypothesize is due to transfer of energy between the two sections.

In the last problem, we investigated the effects of the introduction of nonlinearity into the plate deformation formulation. We modeled the transient deformations of a plate with the inclusion of all geometric and material nonlinearities and observed that the nonlinearities in the system stiffens the system analogous to strain hardening in plastic deformation and thus results in lower deformations as compared to the results of linear elasticity under identical boundary and loading conditions. We presented the difference in the response of plate under dead weight (loading always with respect to undeformed state) and follower loads (loading with respect to the deformed shape). We also investigated how the neutral surface of a plate, defined as the surface with no axial stretching, undergoing nonlinear differs from a linear analysis where the neutral surface always coincides with the mid-surface of the plate. From this work, we conclude that it is essential to include nonlinearities to study the deformations of plates undergoing large deformations as designs based on linear models often over-predicts the deformation scale resulting in over-conservative design criteria. Also, linear models cannot always capture correct physical phenomenon in systems undergoing large deformations. .

References

1. Love AEH (1888), The small free vibrations and deformation of a thin elastic shell. Philosophical Transactions of the Royal Society of London. A: p. 491-546.

2. Kirchhoff G (1850), Über die Schwingungen einer kreisförmigen elastischen Scheibe. Annalen der Physik. **157**(10): p. 258-264.

3. Reissner E (1945), The effect of transverse shear deformation on the bending of elastic plates.

4. Mindlin RD (1951), Influence of rotary inertia and shear on flexural motions of isotropic elastic plates.

5. Reddy J and Liu C (1985), A higher-order shear deformation theory of laminated elastic shells. International Journal of Engineering Science. **23**(3): p. 319-330.

6. Vidoli S and Batra RC (2000), Derivation of plate and rod equations for a piezoelectric body from a mixed three-dimensional variational principle. Journal of Elasticity. **59**(1-3): p. 23-50.

7. Batra RC and Vidoli S (2002), Higher-order piezoelectric plate theory derived from a three-dimensional variational principle. Aiaa Journal. **40**(1): p. 91-104.

8. Carrera E and Demasi L (2002), Multilayered Finite Plate Element based on Reissner Mixed Variational Theorem. Part I: Theory. International Journal for Numerical Methods in Engineering. **55**: p. 191-231.

9. Anderson PW (1958), Absence of diffusion in certain random lattices. Physical review. **109**(5): p. 1492.

10. Hodges C (1982), Confinement of vibration by structural irregularity. Journal of sound and vibration. **82**(3): p. 411-424.

11. Valero N and Bendiksen O (1986), Vibration characteristics of mistuned shrouded blade assemblies. Journal of engineering for gas turbines and power. **108**(2): p. 293-299.

12. Wei S-T and Pierre C (1988), Localization phenomena in mistuned assemblies with cyclic symmetry part I: free vibrations. Journal of Vibration, Acoustics, Stress, and Reliability in Design. **110**(4): p. 429-438.

13. Bendiksen OO (1987), Mode localization phenomena in large space structures. AIAA journal. **25**(9): p. 1241-1248.

14. Pierre C Tang DM, and Dowell EH (1987), Localized vibrations of disordered multispan beams-Theory and experiment. AIAA journal. **25**(9): p. 1249-1257.

15. Hodges C and Woodhouse J (1983), Vibration isolation from irregularity in a nearly periodic structure: theory and measurements. The Journal of the Acoustical Society of America. **74**(3): p. 894-905.

16. Pierre C (1990), Weak and strong vibration localization in disordered structures: a statistical investigation. Journal of Sound and Vibration. **139**(1): p. 111-132.

17. Herbert D and Jones R (1971), Localized states in disordered systems. Journal of Physics C: Solid State Physics. **4**(10): p. 1145.

18. Batra R and Aimmanee S (2003), Missing frequencies in previous exact solutions of free vibrations of simply supported rectangular plates. Journal of Sound and Vibration. **265**(4): p. 887-896.

19. Batra R and Aimmanee S (2007), Vibration of an incompressible isotropic linear elastic rectangular plate with a higher-order shear and normal deformable theory. Journal of Sound and Vibration. **307**(3): p. 961-971.

20. Systèmes D (2007), Abaqus analysis user's manual. Simulia Corp. Providence, RI, USA.

21. Meirovitch L (2001), *Fundamentals of vibrations*. International Edition, McGraw-Hill.

22. Balachandran B and Magrab EB (2008), *Vibrations*. 2nd ed., Toronto, Canada: Cengage Learning.

23. Hughes TJ (2012), *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation.

24. Srinivas S and Rao A (1970), Bending, vibration and buckling of simply supported thick orthotropic rectangular plates and laminates. International Journal of Solids and Structures. **6**(11): p. 1463-1481.

25. Qian L Batra R, and Chen L (2003), Free and forced vibrations of thick rectangular plates using higher-order shear and normal deformable plate theory and meshless Petrov-Galerkin (MLPG) method. Computer Modeling in Engineering and Sciences. **4**(5): p. 519-534.

26. Batra R and Jin J (2005), Natural frequencies of a functionally graded anisotropic rectangular plate. Journal of Sound and Vibration. **282**(1): p. 509-516.

27. Kirkman P and Pendry J (1984), The statistics of one-dimensional resistances. Journal of Physics C: Solid State Physics. **17**(24): p. 4327.

28. Pierre C and Plaut RH (1989), Curve veering and mode localization in a buckling problem. Zeitschrift für Angewandte Mathematik und Physik (ZAMP). **40**(5): p. 758-761.

29. Nayfeh AH and Hawwa MA (1994), Use of mode localization in passive control of structural buckling. AIAA journal. **32**(10): p. 2131-2133.

30. Paik S Gupta S, and Batra R (2015), Localization of buckling modes in plates and laminates. Composite Structures. **120**: p. 79-89.

31. Ibrahim R (1987), Structural dynamics with parameter uncertainties. Applied Mechanics Reviews. **40**(3): p. 309-328.

32. Hodges C and Woodhouse J (1986), Theories of noise and vibration transmission in complex structures. Reports on Progress in Physics. **49**(2): p. 107.

33. Nowacki W (1953), Vibration and buckling of rectangular plates simply-supported at the periphery and at several points inside. Archiwum Mechaniki Stosowanej. **5**(3): p. 437-454.

34. Gorman D (1981), An analytical solution for the free vibration analysis of rectangular plates resting on symmetrically distributed point supports. Journal of Sound and Vibration. **79**(4): p. 561-574.

35. Gorman D and Singal R (1991), Analytical and experimental study of vibrating rectangular plates onrigid point supports. AIAA journal. **29**(5): p. 838-844.

36. Bapat A Venkatramani N, and Suryanarayan S (1988), A new approach for the representation of a point support in the analysis of plates. Journal of Sound and Vibration. **120**(1): p. 107-125.

37. Bapat A Venkatramani N, and Suryanarayan S (1988), The use of flexibility functions with negative domains in the vibration analysis of asymmetrically point-supported rectangular plates. Journal of sound and vibration. **124**(3): p. 555-576.

38. Bapat A and Suryanarayan S (1989), The flexibility function approach to vibration analysis of rectangular plates with arbitrary multiple point supports on the edges. Journal of sound and vibration. **128**(2): p. 209-233.

39. Bapat A and Suryanarayan S (1989), Free vibrations of periodically point-supported rectangular plates. Journal of sound and vibration. **132**(3): p. 491-509.

40. Bapat A and Suryanarayan S (1992), The fictitious foundation approach to vibration analysis of plates with interior point supports. Journal of sound and vibration. **155**(2): p. 325-341.

41. Lee S and Lee L (1994), Free vibration analysis of rectangular plates with interior point supports. Journal of Structural Mechanics. **22**(4): p. 505-538.

42. Rao GV Raju I, and Amba-Rao C (1973), Vibrations of point supported plates. Journal of Sound and Vibration. **29**(3): p. 387-391.

43. Raju I and Amba-Rao C (1983), Free vibrations of a square plate symmetrically supported at four points on the diagonals. Journal of Sound and Vibration. **90**(2): p. 291-297.

44. Utjes J Sarmiento GS Laura P, and Gelos R (1986), Vibrations of thin elastic plates with point supports: a comparative study. Applied Acoustics. **19**(1): p. 17-24.

45. Kim C and Dickinson S (1987), The flexural vibration of rectangular plates with point supports. Journal of Sound and Vibration. **117**(2): p. 249-261.

46. Bhat R (1991), Vibration of rectangular plates on point and line supports using characteristic orthogonal polynomials in the Rayleigh-Ritz method. Journal of sound and vibration. **149**(1): p. 170-172.

47. Filoche M and Mayboroda S (2009), Strong localization induced by one clamped point in thin plate vibrations. Physical review letters. **103**(25): p. 254301.

48. Sharma D Gupta S, and Batra R (2012), Mode localization in composite laminates. Composite Structures. **94**(8): p. 2620-2631.

49. Du J Li WL Jin G Yang T, and Liu Z (2007), An analytical method for the in-plane vibration analysis of rectangular plates with elastically restrained edges. Journal of Sound and Vibration. **306**(3-5): p. 908-927.

50. Vel SS and Batra R (1999), Analytical solution for rectangular thick laminated plates subjected to arbitrary boundary conditions. AIAA journal. **37**(11): p. 1464-1473.

51. Shah P and Batra R (2017), Stress singularities and transverse stresses near edges of doubly curved laminated shells using TSNDT and stress recovery scheme. European Journal of Mechanics-A/Solids. **63**: p. 68-83.

52. Lo K Christensen R, and Wu E (1977), A High-Order Theory of Plate Deformation. Journal of applied mechanics: p. 669.

53. Carrera E (1999), A study of transverse normal stress effect on vibration of multilayered plates and shells. Journal of Sound and Vibration. **225**(5): p. 803-829.

54. Alsuwaiyan A and Shaw SW (2003), Steady-state responses in systems of nearly-identical torsional vibration absorbers. Transactions-American Society of Mechanical Engineers Journal of Vibration and Acoustics. **125**(1): p. 80-87.

55. Alsuwaiyan AS (2013), Localization in multiple nearly-identical tuned vibration absorbers. International Journal of Engineering & Technology. **2**(3): p. 157.

56. Spletzer M Raman A Wu AQ Xu X, and Reifenberger R (2006), Ultrasensitive mass sensing using mode localization in coupled microcantilevers. Applied Physics Letters. **88**(25): p. 254102.

57. Cosserat E and Cosserat F (1909), Théorie des corps déformables. Paris.

58. Srinivas S Rao CJ, and Rao A (1970), An exact analysis for vibration of simply-supported homogeneous and laminated thick rectangular plates. Journal of sound and vibration. **12**(2): p. 187-199.

59. Vel SS and Batra R (2002), Exact solution for thermoelastic deformations of functionally graded thick rectangular plates. AIAA journal. **40**(7): p. 1421-1433.

60. Batra RC Vidoli S, and Vestroni F (2002), Plane wave solutions and modal analysis in higher order shear and normal deformable plate theories. Journal of Sound and Vibration. **257**(1): p. 63-88.

61. Vel SS and Batra R (2000), The generalized plane strain deformations of thick anisotropic composite laminated plates. International Journal of Solids and Structures. **37**(5): p. 715-733.

62. Malatkar P and Nayfeh AH (2003), On the transfer of energy between widely spaced modes in structures. Nonlinear Dynamics. **31**(2): p. 225-242.

63. Hirwani CK Panda SK, and Mahapatra TR (2018), Nonlinear finite element analysis of transient behavior of delaminated composite plate. Journal of Vibration and Acoustics. **140**(2): p. 021001.

64. Xiao J and Batra R (2014), Delamination in sandwich panels due to local water slamming loads. Journal of Fluids and Structures. **48**: p. 122-155.

65. Sapkale SL Sucheendran MM Gupta SS, and Kanade SV (2018), Vibroacoustic study of a point-constrained plate mounted in a duct. Journal of Sound and Vibration. **420**: p. 204-226.

66. Chree C (1889), The Equations of an Isotropic Elastic Solid in Polar and Cylindrical Co-ordinates their Solution and Application. Transactions of the Cambridge Philosophical Society. **14**: p. 250.

67. Vlasov B (1957), On the equations of bending of plates. Dokla Ak Nauk Azerbeijanskoi-SSR. **3**: p. 955-979.

68. Pagano N (1969), Exact solutions for composite laminates in cylindrical bending. Journal of composite materials. **3**(3): p. 398-411.

69. Pagano N (1970), Exact solutions for rectangular bidirectional composites and sandwich plates. Journal of composite materials. **4**(1): p. 20-34.

70. Noor AK (1973), Free vibrations of multilayered composite plates. AIAA journal. **11**(7): p. 1038-1039.

71. Noor AK and Rarig PL (1974), Three-dimensional solutions of laminated cylinders. Computer Methods in Applied Mechanics and Engineering. **3**(3): p. 319-334.

72. Ren J (1987), Exact solutions for laminated cylindrical shells in cylindrical bending. Composites Science and Technology. **29**(3): p. 169-187.

73. Bhaskar K and Varadan T (1993), Exact elasticity solution for laminated anisotropic cylindrical shells. Journal of applied mechanics. **60**(1): p. 41-47.

74. Bhaskar K and Varadan T (1994), Benchmark elasticity solution for locally loaded laminated orthotropic cylindrical shells. AIAA journal. **32**(3): p. 627-632.

75. Vel SS and Batra R (2001), Closure to "The generalized plane strain deformations of thick anisotropic composite laminated plates". International Journal of Solids and Structures. **38**(3): p. 483-489.

76. Reissner E (1944), On the theory of bending of elastic plates.

77. Nelson R and Lorch D (1974), A refined theory for laminated orthotropic plates. Journal of Applied Mechanics. **41**(1): p. 177-183.

78. Pandya B and Kant T (1988), Higher-order shear deformable theories for flexure of sandwich plates—finite element evaluations. International Journal of Solids and Structures. **24**(12): p. 1267-1286.

79. Naghdi PM (1973), *The theory of shells and plates*. Springer.

80. Antman SS and Marlow RS (1991), Material constraints, Lagrange multipliers, and compatibility. Applications to rod and shell theories. Archive for Rational Mechanics and Analysis. **116**(3): p. 257-299.

81. Green AE and Naghdi PM (1967), "*Shells in the Light of Generalized Continua"*. Calfornia, Univ Berkeley Div of Applied Mechanics.

82. Flügge W (2013), *Stresses in shells*. Springer Science & Business Media.

83. Leissa A (1977), Recent research in plate vibrations, 1973-1979: classical theory. Shock and Vibration Digest. **9**(10): p. 13-24.

84. Leissa A (1977), Recent research in plate vibrations: complicating effects. Shock and Vibration Digest. **9**(11): p. 21-35.

85. Ambartsumian SA (1961), *Theory of anisotropic shells*. State Publishing House for Physical and Mathematical Literature.

86. Kraus H (1967), *Thin elastic shells*. John Wiley & Sons.

87. Koiter WT and Simmonds JG (1973), *Foundations of shell theory*. Springer.

88. Carrera E (2003), Historical review of zig-zag theories for multilayered plates and shells. Applied mechanics reviews. **56**(3): p. 287-308.

89. Carrera E (2002), Theories and finite elements for multilayered, anisotropic, composite plates and shells. Archives of Computational Methods in Engineering. **9**(2): p. 87-140.

90. Koiter W (1960), A consistent first approximation in the general theory of thin elastic shells. Theory of Thin Elastic Shells: p. 12-33.

91. Mindlin R and Medick M (1959), Extensional vibrations of elastic plates. J. appl. Mech. **26**(4): p. 561-569.

92. Qian L Batra R, and Chen L (2003), Elastostatic deformations of a thick plate by using a higher-order shear and normal deformable plate theory and two meshless local Petrov-Galerkin (MLPG) methods. Computer Modeling in Engineering and Sciences. **4**(1): p. 161-176.

93. Shah P and Batra R (2015), Through-the-Thickness Stress Distributions near Edges of Composite Laminates using Stress Recovery Scheme and Third Order Shear and Normal Deformable Theory. Composite Structures.

94. Ebel H Matikainen MK Hurskainen V-V, and Mikkola A (2017), Analysis of high-order quadrilateral plate elements based on the absolute nodal coordinate formulation for three-dimensional elasticity. Advances in Mechanical Engineering. **9**(6): p. 1687814017705069.

95. Betsch P Gruttmann F, and Stein E (1996), A 4-node finite shell element for the implementation of general hyperelastic 3D-elasticity at finite strains. Computer Methods in Applied Mechanics and Engineering. **130**(1-2): p. 57-79.

96. Pascon J and Coda H (2013), High-order tetrahedral finite elements applied to large deformation analysis of functionally graded rubber-like materials. Applied Mathematical Modelling. **37**(20-21): p. 8757-8775.

97. Batra R and Xiao J (2013), Finite deformations of curved laminated St. Venant–Kirchhoff beam using layer-wise third order shear and normal deformable beam theory (TSNDT). Composite Structures. **97**: p. 147-164.

98. Batra R and Xiao J (2015), Finite deformations of full sine-wave St.-Venant beam due to tangential and normal distributed loads using nonlinear TSNDT. Meccanica. **50**(2): p. 355-365.

99. Ericksen J (1973), Plane infinitesimal waves in homogeneous elastic plates. Journal of Elasticity. **3**(3): p. 161-167.

100. Jabareen M and Mtanes E (2016), A solid-shell Cosserat point element (SSCPE) for elastic thin structures at finite deformation. Computational Mechanics. **58**(1): p. 59-89.

101. Birsan M Sadowski T, and Pietras D (2013), Thermoelastic deformations of cylindrical multi-layered shells using a direct approach. Journal of Thermal Stresses. **36**(8): p. 749-789.

102. Chróścielewski J and Witkowski W (2011), FEM analysis of Cosserat plates and shells based on some constitutive relations. ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik. **91**(5): p. 400-412.

103. Ciarlet PG (1980), A justification of the von Kármán equations. Archive for Rational Mechanics and Analysis. **73**(4): p. 349-389.

104. Ciarlet PG and Lods V (1996), Asymptotic analysis of linearly elastic shells. I. Justification of membrane shell equations. Archive for rational mechanics and analysis. **136**(2): p. 119-161.

105. Xiao Lm (1998), Asymptotic analysis of dynamic problems for linearly elastic shells–justification of equations for dynamic membrane shells. Asymptotic Analysis. **17**(2): p. 121-134.

106. Ciarlet PG (1990), Plates and junctions in elastic multi-structures. An asymptotic analysis.

107. Batra R (2001), Comparison of results from four linear constitutive relations in isotropic finite elasticity. International journal of non-linear mechanics. **36**(3): p. 421-432.

108. Batra RC (2006), *Elements of continuum mechanics*. AIAA.

109. Hartman TB Hyer MW, and Case SW (2012), Geometrically nonlinear stress recovery in composite laminates. AIAA journal. **50**(5): p. 1156-1168.

110. Hartman TB Hyer MW, and Case SW (2016), Stress recovery in composite laminates including geometrically nonlinear and dynamic effects. AIAA Journal: p. 2521-2529.

111. Batra R Vidoli S, and Vestroni F (2002), Plane wave solutions and modal analysis in higher order shear and normal deformable plate theories. Journal of Sound and Vibration. **257**(1): p. 63-88.

112. Tornabene F Fantuzzi N Viola E, and Batra RC (2015), Stress and strain recovery for functionally graded free-form and doubly-curved sandwich shells using higher-order equivalent single layer theory. Composite Structures. **119**: p. 67-89.

113. Guide MUs (1998), The mathworks. Inc., Natick, MA. **5**: p. 333.

Appendices

Appendix A

The equations of motion of a plate are,

$$\rho_0 \ddot{u}_i = \frac{\partial T_{Li}}{\partial X_L} + f_i \tag{A.1}$$

Substituting for the displacement u_i from Eqn. (4) yields,

$$\rho_0 \left(\ddot{u}_{i0} + X_3 \ddot{u}_{i1} + \left(X_3\right)^2 \ddot{u}_{i2} + \left(X_3\right)^3 \ddot{u}_{i3} \right) = \frac{\partial T_{Li}}{\partial X_L} + f_i \tag{A.2}$$

Multiplying both sides of Eqn (A.2) by powers of X_3 ranging from 0 to 3 and integrating over the domain gives,

$$\begin{aligned} &\rho_0 \left(\int_{-h/2}^{h/2} \left(X_3\right)^j \left(\ddot{u}_{i0} + X_3 \ddot{u}_{i1} + \left(X_3\right)^2 \ddot{u}_{i2} + \left(X_3\right)^3 \ddot{u}_{i3} \right) dX_3 \right) \\ &= \int_{-h/2}^{h/2} \left(X_3\right)^j \left(\frac{\partial T_{i1}}{\partial X_1} + \frac{\partial T_{i2}}{\partial X_2} \right) dX_3 + \int_{-h/2}^{h/2} \left(X_3\right)^j \frac{\partial T_{i3}}{\partial X_3} dX_3 \end{aligned} \tag{A.3}$$

where i takes values from 1 to 3 and j from of 0 to 3. This eliminates terms on the left hand side of Eqn. (A.3) multiplying $\ddot{u}_{i1}$ and $\ddot{u}_{i3}$ when j is even (including 0) and the terms multiplying $\ddot{u}_{i0}$ and $\ddot{u}_{i2}$ when j is odd.

The element A_j^k of the matrix **A** in Eqn (A.4) represents the $(j, k)^{\text{th}}$ term of the 4 x 4 matrix representing coefficients of $\ddot{u}_{ij}$

$$\mathbf{A} = \begin{bmatrix} h & 0 & \frac{h^3}{12} & 0 \\ 0 & \frac{h^3}{12} & 0 & \frac{h^5}{80} \\ \frac{h^3}{12} & 0 & \frac{h^5}{80} & 0 \\ 0 & \frac{h^5}{80} & 0 & \frac{h^7}{448} \end{bmatrix} \tag{A.4}$$

On the right hand side of Eqn. (A.3), products of different powers of X_3 multiplied by T_{Li} and integrated over the plate thickness are termed as moments of stresses and is expressed as,

$$M_{iL}^{(k)} = \int_{-h/2}^{h/2} \left(X_3\right)^k T_{iL} dX_3, \quad k = 0,1,2,3 \tag{A.5}$$

Thus $M_{Li}^{(0)}$ equals the resultant force, $M_{Li}^{(1)}$ the usual moment, and $M_{Li}^{(2)}$ and $M_{Li}^{(3)}$ are higher-order moments of the 1st Piola-Kirchhoff stresses.

Appendix B

We neglect the body force f_i, take the inner product of the 12 equations of motion listed as Eqn (4.7a) with a 12-dimensional test function, ϕ_i^k and integrate over the plate reference (taken here as the mid-) surface. With the use of the divergence theorem on the term $\sum_{\alpha=1}^{2} \frac{\partial M_{\alpha i}^{k}}{\partial X_\alpha}$ in Eqn. (4.7a), the resulting equations take the form,

$$\int_\Omega \rho_0 \phi_i^1 \left(\ddot{u}_{i0} h + \frac{h^3}{12} \ddot{u}_{i2} \right) dX_1 dX_2 = -\int_\Omega \left(M_{i1}^0 \frac{\partial \phi_i^1}{\partial X_1} + M_{i2}^0 \frac{\partial \phi_i^1}{\partial X_2} \right) dX_1 dX_2 + \int_\Omega \phi_i^1 T_{i3}\Big|_{X_3=-h/2}^{X_3=h/2} dX_1 dX_2 + \oint_\Gamma \phi_i^1 \left(M_{i1}^0 N_1 + M_{i2}^0 N_2 \right) dS \tag{B.1}$$

$$\int_\Omega \rho_0 \phi_i^2 \left(\ddot{u}_{i1} \frac{h^3}{12} + \frac{h^5}{80} \ddot{u}_{i3} \right) dX_1 dX_2 = -\int_\Omega \left(M_{i1}^1 \frac{\partial \phi_i^2}{\partial X_1} + M_{i2}^1 \frac{\partial \phi_i^2}{\partial X_2} + M_{i3}^0 \phi_i^2 \right) dX_1 dX_2 + \int_\Omega \phi_i^2 \left(X_3 T_{i3} \right)\Big|_{X_3=-h/2}^{X_3=h/2} dX_1 dX_2 + \oint_\Gamma \phi_i^2 \left(M_{i1}^1 N_1 + M_{i2}^1 N_2 \right) dS \tag{B.2}$$

$$\int_\Omega \rho_0 \phi_i^3 \left(\ddot{u}_{i0} \frac{h^3}{12} + \frac{h^5}{80} \ddot{u}_{i2} \right) dX_1 dX_2 = -\int_\Omega \left(M_{i1}^2 \frac{\partial \phi_i^3}{\partial X_1} + M_{i2}^2 \frac{\partial \phi_i^3}{\partial X_2} + 2M_{i3}^1 \phi_i^3 \right) dX_1 dX_2 + \int_\Omega \phi_i^3 \left(X_3^2 T_{i3} \right)\Big|_{X_3=-h/2}^{X_3=h/2} dX_1 dX_2 + \oint_\Gamma \phi_i^3 \left(M_{i1}^2 N_1 + M_{i2}^2 N_2 \right) dS \tag{B.3}$$

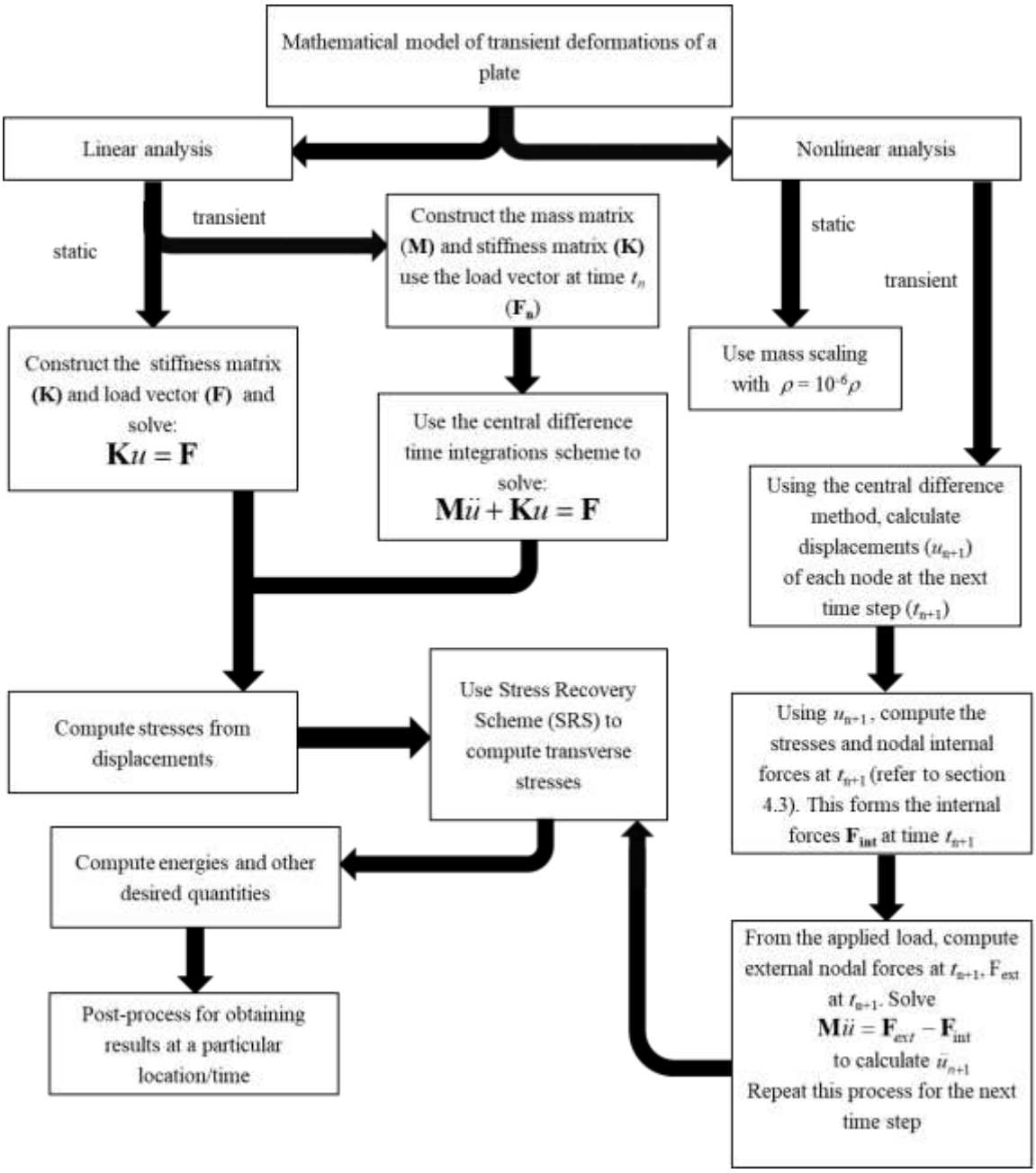
$$\int_\Omega \rho_0 \phi_i^4 \left(\ddot{u}_{i1} \frac{h^5}{80} + \frac{h^7}{448} \ddot{u}_{i3} \right) dX_1 dX_2 = -\int_\Omega \left(M_{i1}^3 \frac{\partial \phi_i^4}{\partial X_1} + M_{i2}^3 \frac{\partial \phi_i^4}{\partial X_2} + 3M_{i3}^2 \phi_i^4 \right) dX_1 dX_2 + \int_\Omega \phi_i^4 \left(X_3^3 T_{i3} \right)\Big|_{X_3=-h/2}^{X_3=h/2} dX_1 dX_2 + \oint_\Gamma \phi_i^4 \left(M_{i1}^3 N_1 + M_{i2}^3 N_2 \right) dS \tag{B.4}$$

$$i = 1, 2, 3$$

Each of the above equations represent 3 equations corresponding to the 1, 2 and 3 directions. The terms on the left hand side of these equations represent the inertial forces of the system in the directions of the three coordinate axes. Terms on the right hand side integrating moment *M* over the thickness of the plate represent the internal forces of the system and those with the boundary tractions represent the applied forces.

Appendix C

In this appendix we present the TSNDT solution algorithm solving static/transient linear/nonlinear problems. We emphasize on the nonlinear problem as the linear part is standard in FEM. The algorithm is given below. Here *u* is the nodal displacement vector.



Appendix E

The computer code of the FEM software used in this work is provided below.

The user has to compile the code in a FORTRAN compiler available and execute the executable file. The input file defining all the input parameter is also provided below. The program generates displacement and stress data in the files named "DEFORMED_SHAPE.DAT" and "STRESS_OUT.DAT". These files are output in the data format accepted by the software TecPlot and can be directly imported into TecPlot using the data import tool of TecPlot.

```
* UNITS - mass-mg,length-mm,time-micros,force-kN,stress-GPa,energy-J
*******I_CURVE, 1 - STRAIGHT PLATE, 2 - CURVED PLATE***
1
*****LENGTH OF THE PLATE [mm]********
100.0
*****WIDTH OF THE PLATE [mm]*********
100.0
*******HEIGHT OF THE PLATE [mm]******
20.0
*****MATERIAL CONSTANTS
*** E1 [kN/mm^2]
25.0
*** E2 [kN/mm^2]
25.0
*** E3
25.0
***G12
10.0
***G13
10.0
***G23
10.0
*** NU12
0.25
*** NU13
0.25
*** NU23
0.25
****FIBER ORIENTATION ALPHA (DEGREES)
0.0
**********DENSITY(mg/mm^3)*************
5.0
**********APPLIED PEAK LOAD (kN/mm^2)**********
0.5
*****SPATIAL DEPENDENT LOAD 1 - NO, 2 - YES (CURRENTLY AS FIG 4.3, CAN BE CHANGED IN THE SUBROUTINE SUB_MATRICES)
1
**1 - FOLLOWER LOAD ON DEFORMED AREA/2 - TRACTION ON UNDEFORMED AREA (FOR NONLINEAR ONLY)
2
********NUMBER OF ELEMENTS ALONG LENGTH***********
40
********NUMBER OF ELEMENTS ALONG WIDTH************
40
**I_ORDER - ORDER OF SHAPE FUNCTIONS,
** 1 - BILINEAR (4 NODED ELE.) (PROVISION TO EXPAND TO OTHER TYPES OF ELEMENTS)
1
**I_BC 1 SSSS,2 CCCC,3 CFFF,4 CCFS,5 CCSS,6 BEAM,7 INT. CLAMPED NODE (SAME NAMING AS IN THE THESIS)
2
**I_LIN - GEOMERTRIC NON-LINEARITY, 1 - LINEAR, 2 - NONLINEAR
2
**I_VON, FULL - 1, vonKarman(S=CE) - 2, (SIG=CE) - 3, LIN. ELAS. - 4 (4 EXCLUDES ALL NONLINEARITIES)
1
**THEORY ORDER, 1- 1ST ORDER THEORY, 3 - 3RD ORDER THEORY
3
**INTEGRATION POINTS IN X
2
**INTEGRATION POINTS IN Y
2
**INTEGRATION POINTS IN Z
7
** I_STATIC - PROBLEM: 1 - STATIC, 2 - DYNAMIC, 3 - FREQUENCY
2
** DECAY TIME (THETA) ** P=P_0 e^(-t/theta)
100.0
**INITITAL TIME **
0.0
**END TIME**
```

```
100.0
**INCLUDE STRESS RECOVERY: 0 - NO, 1 - YES (SRS FOR NLIN INCOMPLETE)
1
```

```
!-----PROGRAM FOR THIRD ORDER NORMAL AND SHEAR DEFORMABLE THEORY--------
!-----------------------OF A STRAIGHT PLATE-----------------------------
!-------------------------ARKA CHATTOPADHYAY----------------------------
      PROGRAM TSNDT_CODE
!     CALLING THE MODULES DEFINED IN MODULE.F
      USE PLATE_CONST
      USE FEM_CONST
      USE FEM_MATRICES
      USE NUM_INT
      USE PROG_DEBUG
      USE OMP_LIB
!
      IMPLICIT NONE
      INCLUDE 'mkl.fi'
!
      INTEGER :: I,J,K     !LOOP COUNTERS
      INTEGER :: IEL     !ELEMENT NUMBER USED IN LOOPS
      REAL(KIND=8) :: LE,BE  !LENGTH AND WIDTH OF ELEMENTS
      INTEGER :: T_P     ! TOTAL NUMBER OF POINTS USED TO CALCULATE
!                          STRESS AND STRAIN IN EACH ELEMENT
      REAL(KIND=8),ALLOCATABLE :: XE(:,:),YE(:,:)
!     NODAL COORDINATES(GLOBAL) OF EACH ELEMENT
      REAL(KIND=8),ALLOCATABLE :: NODAL_DISP(:,:),NODAL_ACC(:,:)
!     NODAL DISPLACEMENTS OF EACH ELEMENT
      INTEGER, ALLOCATABLE :: CON(:,:)
!     CONNECTIVITY MATRIX
      INTEGER :: NUM,NRHS,IELX,IELY
      REAL(KIND=8) :: E  !MAXIMUM YOUNGS MODULUS (TO CAL. WAVE SPEED)
      REAL(KIND=8) :: C  !WAVE SPEED
      REAL(KIND=8) :: TIME
      CHARACTER*1 :: UPLO
      INTEGER :: REM,QUO
      INTEGER,ALLOCATABLE :: CON2(:,:),CON3(:,:)
!     CONNECTIVITY MADE TO MAKE A 3 MESH FOR PLOTTING STRESSES IN TECPLOT
      INTEGER :: NL2,NB2,N_NODE2,NH2,SIZE_CON2,SIZE_CON3
!     NUMBER OF ELEMENTS IN THE 3D MESH FOR TECHPLOT OUTPUT

!     VARIABLES USED FOR EIGEN SOLVER
      INTEGER :: ITYPE,LWORK,VAR,NODE_ID
      REAL(KIND=8),ALLOCATABLE :: WORK(:),W(:)
      INTEGER,ALLOCATABLE :: FPM(:)  !ARRAY FOR EIGEN SOLVER
      REAL(KIND=8) :: EPSOUT !RELATIVE ERROR IN IRETATIVE SPARSE SOL.
      REAL(KIND=8) :: EMIN,EMAX  !LOWER AND UPPER BOUNDS OF THE EIG. SOL
      INTEGER :: M0          !INTIIAL GUESS OF N. OF EIG. VAL. IN DOMAIN
      INTEGER :: M           ! NO. OF EIGEN VALUES FOUND IN THE DOM.
      INTEGER :: LOOP        !NUMBER OF REFINEMENT LOOP EXECUTED
      REAL(KIND=8),ALLOCATABLE :: EIG_VAL(:) !ARRAY CONTAINING EIG. VAL.
      REAL(KIND=8), ALLOCATABLE :: EIG_VEC(:,:) !EIGEN VECTORS
      REAL(KIND=8),ALLOCATABLE :: RES (:) !RELATIVE RESIDUES OF THE VAL.
      INTEGER :: INFO           !OUTPUT STATUS OF THE EIGEN VALUE SOLVER

!     VARIABLES AND ARRAYS FOR CALCULATING ENERGIES
!     KINETIC ENERGY, WORK DONE, STRAIN ENERGY, TIME
      REAL(KIND=8) :: KINE(1,1),WD(1,1),WDI(1,1),UE(1,1),T
!     TEMP VECS FOR ENERGY
      REAL(KIND=8),ALLOCATABLE :: MV(:),FF1(:,:),FFIN1(:,:)
      REAL(KIND=8),ALLOCATABLE :: VEL_MAT(:,:),DISP_MAT(:,:)
     + ,MVMAT(:,:),DISP_MAT2(:,:)
      REAL(KIND=8) :: PI

      PI=4.0D0*ATAN(1.0D0)
      NRHS=1
!---------------------------------------------------------------------
!     CALLING THE INPUTS FROM THE INPUT FILE (SUB_INPUT.F)
      CALL INPUT (I_CURVE,L,B,H,CC,RHO,APP_LOAD,N_L,N_B,I_ORDER,I_BC,
     + I_LIN,I_THEORY,G_X,G_Y,G_Z,I_STATIC,THETA,T0,TSTOP,BETA,
     + GAMMA,E,I_LOAD,I_VON, I_TRAC, SRS_ID)
      N_FEM=N_L*N_B
!     TOTAL NUMBER OF ELEMENTS
```

```
!-----------------------------------------------------------------------
      WRITE(*,*) 'THIRD ORDER THEORY'
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!
!-----------------------------------------------------------------------
!-----WRITING THE INPUT PARAMETERS IN THE FILE INPUT_PARAMETERS.DAT
      OPEN(110,FILE='INPUT_PARA_O.DAT')
      WRITE(110,*) 'LENGTH OF THE PLATE'
      WRITE(110,21) L
      WRITE(110,*) 'WIDTH OF THE PLATE'
      WRITE(110,21) B
      WRITE(110,*) 'HEIGHT OF THE PLATE'
      WRITE(110,21) H
      WRITE(110,*) 'NUMBER OF ELEMENTS FOR FEM'
      WRITE(110,*) N_FEM
      WRITE(110,*) 'I_ORDER'
      WRITE(110,*) I_ORDER
      WRITE(110,*) 'I_LIN'
      WRITE(110,*) I_LIN
      WRITE(110,*) 'STIFFNESS MATRIX OF THE PLATE MATERIAL'
      DO I = 1,6
        WRITE(110,20) (CC(I,J),J=1,6)
      ENDDO
      WRITE(110,*) 'DECAY TIME'
      WRITE(110,21) THETA
      WRITE(110,*) 'APPLIED LOAD'
      WRITE(110,*) APP_LOAD
      CLOSE(110)
!!-------------END OF WRITING OF INPUTS--------------------------------
!-----------------------------------------------------------------------
      IF (I_CURVE == 1) THEN
        WRITE(*,*) 'STRAIGHT'
      ELSEIF (I_CURVE == 2) THEN
        WRITE(*,*) 'CURVED'
      ELSE
        WRITE(*,*) 'INVALID INPUT'
        GOTO 1000
      ENDIF
!-----------------------------------------------------------------------
      IF (I_BC==1) THEN
        WRITE(*,*) 'SSSS PLATE'
      ELSEIF (I_BC==2) THEN
        WRITE(*,*) 'CCCC PLATE'
      ELSEIF (I_BC==3) THEN
        WRITE(*,*) 'CCFF PLATE'
      ELSEIF (I_BC==4) THEN
        WRITE(*,*) 'CCFS PLATE'
      ELSEIF (I_BC==5) THEN
        WRITE(*,*) 'CCSS PLATE'
      ELSEIF (I_BC==6) THEN
        WRITE(*,*) 'BEAM PROBLEM'
      ELSEIF (I_BC==7) THEN
        WRITE(*,*) 'INTERIOR CLAMPED NODE'
      ELSE
        WRITE(*,*) 'INVALID INPUT'
        GOTO 1000
      ENDIF
!-----FEM MESH CALCULATIONS-------------------------------------------
      IF(I_ORDER==1) THEN
         E_NODE=4         !4 NODED ELEMENTS WITH LINEAR SHAPE FUNCTIONS
         NODE_L=N_L+1
         NODE_B=N_B+1
         N_NODE=NODE_B*NODE_L
      ELSE
         E_NODE=8         !8 NODED ELEMENTS WITH QUAD. SHAPE FUNCTIONS
         NODE_L=2*N_L+1
         NODE_B=2*N_B+1
         N_NODE=(NODE_L)*(NODE_B)
```

```
      ENDIF
      N_DOF=12             ! DEGREES OF FREEDOM PER NODE
      E_DOF=E_NODE*N_DOF   ! DEGREES OF FREEDOM PER ELEMENT
      TOT_DOF=N_DOF*N_NODE !TOTAL DEGEES OF FREEDOM OF THE PLATE
      WRITE(*,*) 'TOTAL NUMBER OF ELEMENTS, NODES ARE'
      WRITE(*,*) N_FEM, N_NODE
      WRITE(*,*) 'NUMBER OF INTEGRATION POINTS IN X,Y,Z'
      WRITE(*,*) G_X,G_Y,G_Z
      WRITE(*,*) 'DOF PER ELEMENT AND TOTAL RESP.'
      WRITE(*,*) E_DOF,TOT_DOF
!-----------------------------------------------------------------------
!
!
!-----------------------------------------------------------------------
!-----------------ALLOCATING SIZES OF MATRICES-------------------------
!     PLEASE SEE THE MODULE FEM_MATRICES FOR THE DEFINITIONS
      ALLOCATE(KE(E_DOF,E_DOF),ME(E_DOF,E_DOF))
      ALLOCATE(GLOB_XYZ(N_NODE,2))
      ALLOCATE(FE(E_DOF))
      ALLOCATE(FF(TOT_DOF))
      ALLOCATE(SOL(TOT_DOF))
      ALLOCATE(U10(N_NODE),U11(N_NODE),U12(N_NODE),U13(N_NODE))
      ALLOCATE(U20(N_NODE),U21(N_NODE),U22(N_NODE),U23(N_NODE))
      ALLOCATE(U30(N_NODE),U31(N_NODE),U32(N_NODE),U33(N_NODE))
      ALLOCATE(XE(N_FEM,E_NODE),YE(N_FEM,E_NODE))
      ALLOCATE(NODAL_DISP(E_DOF,1),NODAL_ACC(E_DOF,1))
      ALLOCATE(CON(N_FEM,E_DOF))
!-----------------------------------------------------------------------
!-----INITIALIZING MATRICES-----------------------------------------------
      GLOB_XYZ=0.0D0;
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
      LE=L/N_L
      BE=B/N_B
      WRITE(*,*) 'DIMENSION OF EACH ELEMENT'
      WRITE(*,*) LE,BE
!-----------------------------------------------------------------------
!-----GLOBAL NODAL COORDINATES
      DO I =1,NODE_B
         DO J=1,NODE_L
             GLOB_XYZ((I-1)*NODE_L+J,1)=(J-1)*LE;
             GLOB_XYZ((I-1)*NODE_L+J,2)=(I-1)*BE;
         ENDDO
      ENDDO
!-----------------------------------------------------------------------
      OPEN(51,FILE='DUMP.DAT')   !FILE TO DUMP ALL THE RESULTS NEEDED
!-----------------------------------------------------------------------
      OPEN(7,FILE='GLOB_COR.DAT')
      DO I=1,N_NODE
          WRITE(7,20) (GLOB_XYZ(I,J), J=1,2)
      ENDDO
      CLOSE(7)
!     ELEMENTS AND RESPECTIVE NODAL COORDINATES
!
          DO J=1,N_B
             DO I=1,N_L
                 XE((J-1)*N_L+I,1)=GLOB_XYZ(I,1)
                 XE((J-1)*N_L+I,2)=GLOB_XYZ(I+1,1)
                 XE((J-1)*N_L+I,4)=XE((J-1)*N_L+I,1)
                 XE((J-1)*N_L+I,3)=XE((J-1)*N_L+I,2)
                 YE((J-1)*N_L+I,1)=GLOB_XYZ((J-1)*N_L+J,2)
                 YE((J-1)*N_L+I,2)=GLOB_XYZ((J-1)*N_L+J,2)
                 YE((J-1)*N_L+I,4)=GLOB_XYZ((J)*N_L+J+1,2)
                 YE((J-1)*N_L+I,3)=GLOB_XYZ((J)*N_l+J+1,2)
```

```
            ENDDO
         ENDDO
         OPEN(15,FILE='ELE_CORD.DAT')
         DO I=1,N_FEM
            WRITE(15,20) (XE(I,J),J=1,E_NODE),(YE(I,J), J=1,E_NODE)
         ENDDO

       NL2=N_L-1
       NB2=N_B-1
       NH2=G_Z-1
       SIZE_CON2=NL2*NB2*NH2
       SIZE_CON3=NL2*NB2
       ALLOCATE(CON2(SIZE_CON2,8))
       ALLOCATE(CON3(SIZE_CON3,4))

       NB2=N_B-1
       NL2=N_L-1
       N_NODE2=G_Z*N_L
       CON2=0
       DO K=1,NL2
           DO I=1,NH2
              DO J=1,2
              CON2(K+I+(NH2-1)*K-NH2,4*J-3)=(J-1)*N_NODE2+K+I+
     +        (NH2-1)*K-NH2+K-1;
              CON2(K+I+(NH2-1)*K-NH2,4*J-2)=(J-1)*N_NODE2+K+I+
     +        (NH2-1)*K-NH2+1+K-1;
              CON2(K+I+(NH2-1)*K-NH2,4*J-1)=(J-1)*N_NODE2+NH2+K+
     +        I+(NH2-1)*K-NH2+2+K-1;
              CON2(K+I+(NH2-1)*K-NH2,4*J)=(J-1)*N_NODE2+NH2+K+I+
     +        (NH2-1)*K-NH2+1+K-1;
              ENDDO
           ENDDO
         ENDDO
           DO I=1,NB2-1
               DO J=1,NH2*NL2
                   DO K=1,8
                     CON2(I*NH2*NL2+J,K)=CON2((I-1)*NH2*NL2+J,K)+G_Z*N_L
                   ENDDO
               ENDDO
           ENDDO
         OPEN(927,FILE='3DCONNEC.DAT')
           DO I=1,NL2*NB2*NH2
               WRITE(927,23) (CON2(I,J),J=1,8)
           ENDDO
         CLOSE(927)

       CON3=0

       DO K=1,NB2
           DO I=1,NL2
              DO J=1,4
              CON3(K+I+(NL2-1)*K-NL2,1)=K+I+
     +        (NL2-1)*K-NL2+K-1;
              CON3(K+I+(NL2-1)*K-NL2,2)=K+I+
     +        (NL2-1)*K-NL2+1+K-1;
              CON3(K+I+(NL2-1)*K-NL2,3)=NL2+K+
     +        I+(NL2-1)*K-NL2+2+K-1;
              CON3(K+I+(NL2-1)*K-NL2,4)=NL2+K+I+
     +        (NL2-1)*K-NL2+1+K-1;
              ENDDO
           ENDDO
       ENDDO
```

```
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300         IF (I_LIN==1) THEN
301     !-----------------------------------------------------------------------
302     !-----------------------------------------------------------------------
303     !-----------------------------------------------------------------------
304     !     CALLING MATRICES AND MATRIX OPERATIONS
305     !-----------------------------------------------------------------------
306           WRITE(*,*) 'INFINITESIMAL DEFORMATIONS'
307     !     MASS MATRIX - FILE SUB_MATRICES.F
308           ALLOCATE(MM(TOT_DOF,TOT_DOF))
309
310           CALL M_MAT(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,RHO,LE,BE,ME,MM
311          + ,N_FEM,E_NODE,N_L,N_B,N_NODE,CON)
312
313     !     BOUNDARY CONDITIONS
314     !     FILE SUB_BC.F
315     !     STARTING WITH MASS MATRIX
316           CALL BC_APP_KM(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
317          +  N_NODE,I_STATIC,I_BC,MM)
318
319     !-----------------------------------------------------------------------
320     !
321     !                     MASS MATRIX OUTPUT
322     !
323     !-----------------------------------------------------------------------
324           IF(DEBUG) THEN
325           OPEN(11,FILE='MASS_MAT_OUT.DAT')
326           OPEN(66,FILE='MASS_MAT.DAT')
327           DO I=1,E_DOF
328              WRITE(66,20) (ME(I,J), J=1,E_DOF)
329           ENDDO
330           CLOSE(66)
331     !      WRITE(11,*) 'ELEMENT MASS MATRIX - LUMPED'
332     !      DO I=1,E_DOF
333     !         WRITE(11,20) (MEL(I,J), J=1,E_DOF)
334     !      ENDDO
335           DO I=1,TOT_DOF
336              WRITE(11,20) (MM(I,J), J=1,TOT_DOF)
337           ENDDO
338           CLOSE(11)
339           ENDIF
340     !-----------------------------------------------------------------------
341     !     SPARSE ASSEMBLY
342           DO I=1,TOT_DOF
343              DO J=1,TOT_DOF
344                  IF (MM(I,J) .NE. 0.0D0) THEN
345                      NUM=NUM+1
```

```
                  NZ_M=NUM
               ENDIF
            ENDDO
         ENDDO
         ALLOCATE(MG_VAL(NZ_M),COL_M(NZ_M),ROWIND_M(TOT_DOF+1))
         MG_VAL=0.0D0;ROWIND_M=0
         COL_M=0
         CALL SPARSE_ASSM(MG_VAL,TOT_DOF,NZ_M,
        + MM,ROWIND_M,COL_M)

         DEALLOCATE (MM)


!-----------------------------------------------------------------------
!     STIFFNESS MATRIX - FILE SUB_MATRICES.F
!
         ALLOCATE(KK(TOT_DOF,TOT_DOF))
         CALL K_MAT(N_FEM,N_NODE,E_DOF,TOT_DOF,N_L,N_B,E_NODE
        + ,H,LE,BE,CC,G_X,G_Y,KE,KK,I_ORDER)
!     BOUNDARY CONDITIONS
!     FILE SUB_BC.F
!     STIFFNESS MATRIX
         CALL BC_APP_KM(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
        +  N_NODE,I_STATIC,I_BC,KK)

!-----------------------------------------------------------------------
!
!                      STIFFNESS MATRIX OUTPUT
!
!-----------------------------------------------------------------------
         IF(DEBUG) THEN
         OPEN(12,FILE='STIFF_MAT_OUT.DAT')
         OPEN(67,FILE='STIFF_MAT.DAT')
         DO I=1,E_DOF
            WRITE(67,20) (KE(I,J), J=1,E_DOF)
         ENDDO
         CLOSE(67)
         DO I=1,TOT_DOF
            WRITE(12,20) (KK(I,J), J=1,TOT_DOF)
         ENDDO
         CLOSE(12)
         ENDIF
!-----------------------------------------------------------------------
!     SPARSE MATRIX ASSEMBLY
!-----------------------------------------------------------------------
         NUM=0
         DO I=1,TOT_DOF
            DO J=1,TOT_DOF
               IF (KK(I,J) .NE. 0.0D0) THEN
                  NUM=NUM+1
                  NZ_K=NUM
               ENDIF
            ENDDO
         ENDDO
         ALLOCATE(KG_VAL(NZ_K),COL_K(NZ_K),ROWIND_K(TOT_DOF+1))
         KG_VAL=0.0D0;ROWIND_K=0
         COL_K=0
         CALL SPARSE_ASSM(KG_VAL,TOT_DOF,NZ_K,
        + KK,ROWIND_K,COL_K)
         DEALLOCATE( KK )

!-----------------------------------------------------------------------
!     LOAD VECTOR - FILE SUB_MATRICES.F
c       WRITE(*,*) 'ILOAD',I_LOAD
c       WRITE(*,*) 'length','breadth',l,b
         CALL LOAD_VEC(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,LE,BE,FE,
        + FF,APP_LOAD,N_FEM,E_NODE,N_NODE,N_L,N_B,I_LOAD,XE,YE,L,B)
```

```
      CALL BC_APP_FF(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
     +  N_NODE,I_STATIC,I_BC,FF)
!-----------------------------------------------------------------------
!
!                           LOAD VECTOR
!
!-----------------------------------------------------------------------

      IF(DEBUG) THEN
      OPEN(13,FILE='LOAD_VEC.DAT')
      DO I=1,TOT_DOF
            WRITE(13,21) FF(I)
      ENDDO
      ENDIF









      WRITE(*,*) 'TOTAL NONZEROS IN K AND M',NZ_K,NZ_M
!
!
      NL2=N_L-1
      NB2=N_B-1
      NH2=G_Z-1
c      ALLOCATE(CON2(N_FEM,E_DOF))
!!!   FORMULATION OF CONNECTIVITY TO FOR 3D MESH



c      DO K=1,N_B
c          DO I=1,N_L
c             DO J=1,E_DOF/4.0D0
c             CON2(K+I+(N_L-1)*K-N_L,4*J-3)=(J-1)*N_NODE+K+I+
c     +        (N_L-1)*K-N_L+K-1;
c             CON2(K+I+(N_L-1)*K-N_L,4*J-2)=(J-1)*N_NODE+K+I+
c     +        (N_L-1)*K-N_L+1+K-1;
c             CON2(K+I+(N_L-1)*K-N_L,4*J-1)=(J-1)*N_NODE+N_L+K+
c     +        I+(N_L-1)*K-N_L+2+K-1;
c             CON2(K+I+(N_L-1)*K-N_L,4*J)=(J-1)*N_NODE+N_L+K+I+
c     +        (N_L-1)*K-N_L+1+K-1;
c             ENDDO
c          ENDDO
c        ENDDO
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------

!!    STATIC SOLUTION-----------------------------------------------------
```

```
!
!
!
      IF(I_STATIC==1) THEN
      WRITE(*,*) 'STATIC SOLUTION'
!-----------------------------------------------------------------------
c      CALL SOLVE_STAT(I_BC,KK,FF,MM,SOL,TOT_DOF,N_NODE,
c     + N_L,N_B,I_STATIC,glob_xyz)
      NRHS=1
      CALL SOLVER_F77_TEST(ROWIND_K,COL_K,KG_VAL,FF,TOT_DOF,
     + TOT_DOF, NZ_K, NRHS,SOL)
!!!!!!!!!!!!!!!!!!!!MANUFACTURED DISPLACEMENT!!!!!!!!!!!!!!!!!!!!!!!!!
         IF(MANU) THEN
!          MANUFACTURED SOLUTIONS
C           WRITE(*,*) 'MANUFACTURED SOLUTION '
!          VALUES OF THE DEGREES OF FREEDOM OF EACH NODE
           SOL=0.0d0
           DO I=1,N_NODE
            SOLN1(I)=1.0D0
            SOLN1(N_NODE+I)=GLOB_XYZ(I,1)
            SOLN1(N_NODE*2+I)=GLOB_XYZ(I,2)
            SOLN1(N_NODE*3+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)
            SOLN1(N_NODE*4+I)=1.0D0
            SOLN1(N_NODE*5+I)=GLOB_XYZ(I,1)
            SOLN1(N_NODE*6+I)=GLOB_XYZ(I,2)
            SOLN1(N_NODE*7+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)
            SOLN1(N_NODE*8+I)=1.0D0
            SOLN1(N_NODE*9+I)=GLOB_XYZ(I,1)
            SOLN1(N_NODE*10+I)=GLOB_XYZ(I,2)
            SOLN1(N_NODE*11+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)
           ENDDO
         ENDIF
!     CLASSIFYING THE RESULTS AS THE APPROPRIATE NODAL DOF
      DO I=1,N_NODE
         U10(I)=SOL(I)
         U11(I)=SOL(N_NODE+I)
         U12(I)=SOL(2*N_NODE+I)
         U13(I)=SOL(3*N_NODE+I)
         U20(I)=SOL(4*N_NODE+I)
         U21(I)=SOL(5*N_NODE+I)
         U22(I)=SOL(6*N_NODE+I)
         U23(I)=SOL(7*N_NODE+I)
         U30(I)=SOL(8*N_NODE+I)
         U31(I)=SOL(9*N_NODE+I)
         U32(I)=SOL(10*N_NODE+I)
         U33(I)=SOL(11*N_NODE+I)
      ENDDO
      TIME=0.0D0
      ALLOCATE(AN(TOT_DOF))
      AN=0.0D0
      OPEN(14,FILE='RESULT.DAT')
          WRITE(14,*) 'TITLE = "DISPLACEMENT"'
          WRITE(14,*) 'VARIABLES = x ,y, "u10", "u11", "U12", "U13",
     + "U20", "U21","U22", "U23", "U30", "U31", "U32", "U33"'
          WRITE(14,*) 'ZONE T= "Undeformed Mesh",N =',N_NODE,'E =',
     + N_FEM,', F=FEPOINT SOLUTIONTIME= ',TIME
          DO I=1,N_NODE
             WRITE(14,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),U10(I),U11(I),
     +       U12(I),U13(I),U20(I),U21(I),U22(I),U23(I),U30(I),
     +       U31(I),U32(I),U33(I)
          ENDDO
          DO I=1,N_FEM
             WRITE(14,23) (CON(I,J), J=1,4)
          ENDDO
      CLOSE(14)


!
!     STRESS CALCULATION
```

```
!
      WRITE(51,*) 'SOL IN MAIN'
      WRITE(51,20) SOL
C       WRITE(51,*) 'TRANSFERRED CONNECTIVITY IN MAIN'
C       DO I=1,N_FEM
C          WRITE(51,23) (CON(I,J), J=1,E_DOF)
C       ENDDO
      T_P=G_X*G_Y*G_Z
      ALLOCATE(ELE_STRAIN(T_P,9),ELE_STRESS(T_P,9))
      OPEN(17,FILE='STRESS.DAT')
      OPEN(16,FILE='STRAIN.DAT')
      OPEN(917,FILE='STRESS_OUT.DAT')
      IF(OUTPUT) THEN
       WRITE(17,*) 'TITLE = "STRESS"'
          WRITE(17,*) 'VARIABLES = x ,y, z, "s11", "s22", "s33", "s23",
     +"s13","s12"'
      WRITE(17,*) 'ZONE T= "Undeformed Mesh", SOLUTIONTIME= ',TIME
      ENDIF


!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      IF (SRS_ID == 0) THEN

      WRITE(917,*) 'TITLE = "STRESS"'
          WRITE(917,*) 'VARIABLES = x ,y, z, "S11", "S22", "S33", "S23",
     +"S13","S12"'
      WRITE(917,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
      WRITE(918,*) 'TITLE = "STRAIN"'
          WRITE(918,*) 'VARIABLES = x ,y, z, "E11", "E22", "E33", "E23",
     +"E13","E12"'
      WRITE(918,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'

      ELSEIF (SRS_ID==1) THEN



      WRITE(917,*) 'TITLE = "STRESS"'
          WRITE(917,*) 'VARIABLES = x ,y, z, "S11", "S22", "S33", "S23",
     +"S13","S12","S23_SRS","S13_SRS","S33_SRS"'
      WRITE(917,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
      WRITE(918,*) 'TITLE = "STRAIN"'
          WRITE(918,*) 'VARIABLES = x ,y, z, "E11", "E22", "E33", "E23",
     +"E13","E12"'
      WRITE(918,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
      ENDIF !END OF SRS_ID IF STATEMENT
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!         TECPLOT OUTPUTS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!





       DO IELY=1,N_B
          DO IELX=1,N_L
             IEL=(IELY-1)*N_L+IELX
C             WRITE(*,*) IEL
             DO J=1,E_DOF
                NODAL_DISP(J,1)=SOL(CON(IEL,J))
                NODAL_ACC(J,1)=AN(CON(IEL,J))
             ENDDO
          NODAL_ACC=0.0D0
```

```
            IF (DEBUG) THEN
                WRITE(51,*) 'LOOP NUMBER',IEL
                WRITE(51,*) 'NODAL DISPLACMENT IN MAIN'
                WRITE(51,20) NODAL_DISP
            ENDIF
            CALL STRESS_CAL (TOT_DOF,H,N_FEM,N_NODE,
     +     NODE_L,NODE_B,IEL,E_DOF,G_X,G_Y,G_Z,T_P,LE,BE,ELE_STRAIN,
     +     SOL,CON,CC,XE,YE,E_NODE,I_ORDER,T0,NODAL_ACC,RHO,SRS_ID,
     +   IELX,IELY,GLOB_XYZ,AN)
            ENDDO
          ENDDO

            DO I=1,NL2*NB2*NH2
                WRITE(917,23) (CON2(I,J),J=1,8)
            ENDDO









!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!     DYNAMIC
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
      ELSEIF(I_STATIC==2) THEN     !DYNAMIC
      WRITE(*,*) 'DYNAMIC SOLUTION'
!     ALLOCATONG DIMESIONS
!     STARTING WITH NODAL VELCOITIES AND ACCELERATIONS
      ALLOCATE(V10(N_NODE),V11(N_NODE),V12(N_NODE),V13(N_NODE))
      ALLOCATE(V20(N_NODE),V21(N_NODE),V22(N_NODE),V23(N_NODE))
      ALLOCATE(V30(N_NODE),V31(N_NODE),V32(N_NODE),V33(N_NODE))
      ALLOCATE(A10(N_NODE),A11(N_NODE),A12(N_NODE),A13(N_NODE))
      ALLOCATE(A20(N_NODE),A21(N_NODE),A22(N_NODE),A23(N_NODE))
      ALLOCATE(A30(N_NODE),A31(N_NODE),A32(N_NODE),A33(N_NODE))
      ALLOCATE(SOLN1(TOT_DOF))
!     VELOCITY AND ACCELERATION VECTORS RESPECTIVEY FOR THE NTH AND
!     (N+1)TH TIME STEP
      ALLOCATE(VN(TOT_DOF),AN(TOT_DOF),VN1(TOT_DOF),AN1(TOT_DOF))
!     LOAD VECTORS INTERNAL AND F_EXT(N+1)
      ALLOCATE(FFIN(TOT_DOF),FFN1(TOT_DOF))
      UPLO='U' !UPPER TRAINGLE OF MASS MATRIX STORED IN SPARSE FORMAT
!
!
      C=SQRT(E/RHO)          !WAVE SPEED IN MM/MICROS)
      WRITE(*,*) 'WAVE SPEED [mm/muS]', C
C      DT=0.002*LE/C           !TIME STEP SIZE
      DT=0.05D0
      WRITE(*,*) 'TIME STEP SIZE [muS]', DT
      WRITE(*,*) 'END TIME', TSTOP      !FINAL TIME
      TSTEPS=TSTOP/DT+1       !NUMBER OF TIME STEPS
c      tsteps=1001
      WRITE(*,*) 'TIME STEPS', TSTEPS
C      A1=DT*GAMMA; A2=DT*(1-GAMMA)
```

```
C      A3=1/BETA/DT**2; A4=1/BETA/DT; A5=(1-2*BETA)/2/BETA
C      WRITE(51,*) A1,A2,A3,A4,A5
       !INITIALIZING DISPLACEMENT, VELOCITY AND ACC.
       SOL=0.0D0; VN=0.0D0; AN=0.0D0
       SOLN1=0.0D0; VN1=0.0D0; AN1=0.0D0
!      LOAD = 0 AT T=0
       NRHS=1
!      FIRST STEP - TIME =0
       FFN1=0.0D0;     FFIN=0.0D0
!      TOTAL POINTS IN EACH ELEMENT FOR STRESS STRAIN CALCULATIONS
       T_P=G_X*G_Y*G_Z
       IF (SRS_ID == 0) THEN
             ALLOCATE(ELE_STRAIN(T_P,9),ELE_STRESS(T_P,9))
       ELSEIF (SRS_ID==1) THEN
             ALLOCATE(ELE_STRAIN(T_P,9),ELE_STRESS(T_P,9))
       ENDIF




!
!      TIME LOOP
!
       DO K=1,TSTEPS
          TIME=T0+(K-1)*DT

          IF (K==1) THEN
!         SOLVING FOR A0
          CALL SOLVER_F77_TEST(ROWIND_M,COL_M,MG_VAL,FFN1,TOT_DOF,
     +    TOT_DOF, NZ_M, NRHS,AN)


          ELSE
!         D(N+1)=DN+DT*VN+DT^2/2*AN
          SOLN1=SOL+DT*VN+DT**2/2.0D0*AN
!!!!!!!!!!!!!!!!!!!!MANUFACTURED DISPLACEMENT!!!!!!!!!!!!!!!!!!!!!!!!!
          IF(MANU) THEN
!           MANUFACTURED SOLUTIONS
C            WRITE(*,*) 'MANUFACTURED SOLUTION '
!           VALUES OF THE DEGREES OF FREEDOM OF EACH NODE
            SOLN1=0.0d0
            DO I=1,N_NODE
C              SOLN1(I)=1.0D0
C              SOLN1(N_NODE+I)=GLOB_XYZ(I,1)
C              SOLN1(N_NODE*2+I)=GLOB_XYZ(I,2)
C              SOLN1(N_NODE*3+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)
C              SOLN1(N_NODE*4+I)=1.0D0
C              SOLN1(N_NODE*5+I)=GLOB_XYZ(I,1)
C              SOLN1(N_NODE*6+I)=GLOB_XYZ(I,2)
C              SOLN1(N_NODE*7+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)
C              SOLN1(N_NODE*8+I)=1.0D0
C              SOLN1(N_NODE*9+I)=GLOB_XYZ(I,1)
C              SOLN1(N_NODE*10+I)=GLOB_XYZ(I,2)
C              SOLN1(N_NODE*11+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)

               SOLN1(I)=SIN(PI*GLOB_XYZ(I,1)/L)*SIN(PI*GLOB_XYZ(I,2)/B)
     +          *1.0d0
               SOLN1(N_NODE+I)=SIN(2.0D0*PI*GLOB_XYZ(I,1)/L)
     +         *SIN(2.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-1.0d0
               SOLN1(N_NODE*2+I)=SIN(3.0D0*PI*GLOB_XYZ(I,1)/L)
     +         *SIN(3.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-2.0d0
               SOLN1(N_NODE*3+I)=SIN(4.0D0*PI*GLOB_XYZ(I,1)/L)
     +         *SIN(4.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-3.0d0
               SOLN1(N_NODE*4+I)=SIN(PI*GLOB_XYZ(I,1)/L)
     +         *SIN(PI*GLOB_XYZ(I,2)/B)*1.0d0
               SOLN1(N_NODE*5+I)=SIN(2.0D0*PI*GLOB_XYZ(I,1)/L)
     +         *SIN(2.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-1.0d0
               SOLN1(N_NODE*6+I)=SIN(3.0D0*PI*GLOB_XYZ(I,1)/L)
     +         *SIN(3.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-2.0d0
```

```
760               SOLN1(N_NODE*7+I)=SIN(4.0D0*PI*GLOB_XYZ(I,1)/L)
761         +      *SIN(4.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-3.0d0
762               SOLN1(N_NODE*8+I)=SIN(PI*GLOB_XYZ(I,1)/L)
763         +      *SIN(PI*GLOB_XYZ(I,2)/B)*1.0d0
764               SOLN1(N_NODE*9+I)=SIN(2.0D0*PI*GLOB_XYZ(I,1)/L)
765         +      *SIN(2.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-1.0d0
766               SOLN1(N_NODE*10+I)=SIN(3.0D0*PI*GLOB_XYZ(I,1)/L)
767         +      *SIN(3.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-2.0d0
768               SOLN1(N_NODE*11+I)=SIN(4.0D0*PI*GLOB_XYZ(I,1)/L)
769         +      *SIN(4.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-3.0d0
770              ENDDO
771             ENDIF
772   !         GETTING F_INT=K*D(N+1)
773             CALL MKL_DCSRSYMV(UPLO,TOT_DOF,KG_VAL,ROWIND_K,COL_K,
774         +     SOLN1,FFIN)
775               IF (OUTPUT) THEN
776                   OPEN(599,FILE='INTERNAL_FORCE.DAT')
777                   DO I=1,TOT_DOF
778                   WRITE(599,20) TIME,FFIN(I)
779                ENDDO
780               ENDIF
781   !         GETTING F(N+1)
782             CALL LOAD_DYNA(K,DT,T0,FF,TOT_DOF,THETA,FFN1,TSTEPS,APP_LOAD)
783               IF (OUTPUT) THEN
784                OPEN(666,FILE='TIME_LOAD.DAT')
785                DO I=1,TOT_DOF
786                   WRITE(666,20) TIME,FFN1(I)
787                ENDDO
788               ENDIF
789             FFN1=FFN1-FFIN
790   C          IF (OUTPUT) THEN
791   C             WRITE(666,20) FFN1
792   C             WRITE(666,*) '-------------------------------------------'
793   C          ENDIF
794   !         SOLVING FOR A(N+1) M*A(N+1)=F(N+1)-K*D(N+1)
795             CALL SOLVER_F77_TEST(ROWIND_M,COL_M,MG_VAL,FFN1,TOT_DOF,
796         +   TOT_DOF, NZ_M, NRHS,AN1)
797   !         GETTING V(N+1)=VN+DT*(AN/2+A(N+1)/2)
798             VN1=VN+DT*(AN/2.0D0+AN1/2.0D0)
799             ENDIF
800   c          if (k<5) then
801   c          write(999,*) 'step', k
802   c          do i=1,tot_dof
803   c              write(999,20) sol(i),soln1(i),vn(i),vn1(i),an(i),an1(i)
804   c          enddo
805   c          endif
806
807
808
809   !         OVERWRITING Nth VALUES BY N+1th VALUES
810             AN=AN1
811             VN=VN1
812             SOL=SOLN1
813
814   c
815   !         TIME AT THE Kth STEP
816
817   c          write(*,*) 'time step and time', k,time
818   c          pause
819   !         GENERATING OUTPUTS OF DISP, VEL, ACC AND CALCULATING STRESSES
820
821
822
823              QUO=K/200
824              REM=K-QUO*200
825             IF (K==1 .OR. REM==0) THEN
826   c          write(*,*) 'output generated', k
827   c          pause
828   c          SOL=SOL*100*E*H**3.0D0/12.0D0/APP_LOAD/L**4.0D0/
```

```
c     +   (1-0.3D0**2.0D0)
          DO I=1,N_NODE
             U10(I)=SOL(I)
             U11(I)=SOL(N_NODE+I)
             U12(I)=SOL(2*N_NODE+I)
             U13(I)=SOL(3*N_NODE+I)
             U20(I)=SOL(4*N_NODE+I)
             U21(I)=SOL(5*N_NODE+I)
             U22(I)=SOL(6*N_NODE+I)
             U23(I)=SOL(7*N_NODE+I)
             U30(I)=SOL(8*N_NODE+I)
             U31(I)=SOL(9*N_NODE+I)
             U32(I)=SOL(10*N_NODE+I)
             U33(I)=SOL(11*N_NODE+I)
             V10(I)=VN(I)
             V11(I)=VN(N_NODE+I)
             V12(I)=VN(2*N_NODE+I)
             V13(I)=VN(3*N_NODE+I)
             V20(I)=VN(4*N_NODE+I)
             V21(I)=VN(5*N_NODE+I)
             V22(I)=VN(6*N_NODE+I)
             V23(I)=VN(7*N_NODE+I)
             V30(I)=VN(8*N_NODE+I)
             V31(I)=VN(9*N_NODE+I)
             V32(I)=VN(10*N_NODE+I)
             V33(I)=VN(11*N_NODE+I)
             A10(I)=AN(I)
             A11(I)=AN(N_NODE+I)
             A12(I)=AN(2*N_NODE+I)
             A13(I)=AN(3*N_NODE+I)
             A20(I)=AN(4*N_NODE+I)
             A21(I)=AN(5*N_NODE+I)
             A22(I)=AN(6*N_NODE+I)
             A23(I)=AN(7*N_NODE+I)
             A30(I)=AN(8*N_NODE+I)
             A31(I)=AN(9*N_NODE+I)
             A32(I)=AN(10*N_NODE+I)
             A33(I)=AN(11*N_NODE+I)
          ENDDO
           OPEN(14,FILE='RESULT.DAT')
           WRITE(14,*) 'TITLE = "DISPLACEMENT"'
           WRITE(14,*) 'VARIABLES = x ,y, "u10", "u11", "U12", "U13",
     + "U20", "U21","U22", "U23", "U30", "U31", "U32", "U33"'
           WRITE(14,*) 'ZONE T= "Undeformed Mesh",N =',N_NODE,'E =',
     + N_FEM,', F=FEPOINT SOLUTIONTIME= ',TIME
           DO I=1,N_NODE
              WRITE(14,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),U10(I),U11(I),
     +       U12(I),U13(I),U20(I),U21(I),U22(I),U23(I),U30(I),
     +       U31(I),U32(I),U33(I)
           ENDDO
           DO I=1,N_FEM
              WRITE(14,23) (CON(I,J), J=1,4)
           ENDDO
c           CLOSE(14)
           OPEN(39,FILE='VELOCITIES.DAT')
           DO I=1,N_NODE
              WRITE(39,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),V10(I),V11(I),
     +       V12(I),V13(I),V20(I),V21(I),V22(I),V23(I),V30(I),V31(I)
     +      ,V32(I),V33(I),TIME
           ENDDO
c           CLOSE(39)
           OPEN(40,FILE='ACCELERATION.DAT')
           DO I=1,N_NODE
              WRITE(40,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),A10(I),A11(I),
     +       A12(I),A13(I),A20(I),A21(I),A22(I),A23(I),A30(I),A31(I)
     +       ,A32(I),A33(I),TIME
           ENDDO
c           CLOSE(40)

```

```
!     STRESS CALCULATION
      WRITE(51,*) 'SOL IN MAIN'
      WRITE(51,20) SOL
      WRITE(51,*) 'TRANSFERRED CONNECTIVITY IN MAIN'
      DO I=1,N_FEM
         WRITE(51,23) (CON(I,J), J=1,E_DOF)
      ENDDO
c      pause

!     OUTPUTTING THE RESULTS OF STRESSES
      IF (OUTPUT) THEN
      OPEN(17,FILE='STRESS.DAT')
      WRITE(17,*) 'TITLE = "STRESS"'
          WRITE(17,*) 'VARIABLES = x ,y, z, "S11", "S22", "S33", "S23",
     +"s13","s12"'
      WRITE(17,*) 'ZONE T= "Undeformed Mesh", SOLUTIONTIME= ',TIME
      OPEN(16,FILE='STRAIN.DAT')
      ENDIF


      OPEN(917,FILE='STRESS_OUT.DAT')
      OPEN(918,FILE='STRAIN_OUT.DAT')

      IF (SRS_ID == 0) THEN
      WRITE(917,*) 'TITLE = "STRESS"'
          WRITE(917,*) 'VARIABLES = x ,y, z, "S11", "S22", "S33", "S23",
     +"S13","S12"'
      WRITE(917,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
      WRITE(918,*) 'TITLE = "STRAIN"'
          WRITE(918,*) 'VARIABLES = x ,y, z, "E11", "E22", "E33", "E23",
     +"E13","E12"'
      WRITE(918,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'

      ELSEIF (SRS_ID==1) THEN

      WRITE(917,*) 'TITLE = "STRESS"'
          WRITE(917,*) 'VARIABLES = x ,y, z, "S11", "S22", "S33", "S23",
     +"S13","S12","S23_SRS","S13_SRS","S33_SRS"'
      WRITE(917,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
      WRITE(918,*) 'TITLE = "STRAIN"'
          WRITE(918,*) 'VARIABLES = x ,y, z, "E11", "E22", "E33", "E23",
     +"E13","E12"'
      WRITE(918,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,',E =',
     + NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
      ENDIF !END OF SRS_ID IF STATEMENT
          DO IELY=1,N_B
            DO IELX=1,N_L
             IEL=(IELY-1)*N_L+IELX
             DO J=1,E_DOF
                 NODAL_DISP(J,1)=SOL(CON(IEL,J))
                 NODAL_ACC(J,1)=AN(CON(IEL,J))
             ENDDO
             IF (DEBUG) THEN
             WRITE(51,*) 'LOOP NUMBER',IEL
             WRITE(51,*) 'NODAL DISPLACMENT IN MAIN'
             WRITE(51,20) NODAL_DISP
             ENDIF
             CALL STRESS_CAL (TOT_DOF,H,N_FEM,N_NODE,
     +       NODE_L,NODE_B,IEL,E_DOF,G_X,G_Y,G_Z,T_P,LE,BE,ELE_STRAIN,
     +       SOL,CON,CC,XE,YE,E_NODE,I_ORDER,T0,NODAL_ACC,RHO,SRS_ID,
     +       IELX,IELY,GLOB_XYZ,AN)
            ENDDO
          ENDDO
          DO I=1,NL2*NB2*NH2
             WRITE(917,23) (CON2(I,J),J=1,8)
         ENDDO
```

```
          DO I=1,NL2*NB2*NH2
              WRITE(918,23) (CON2(I,J),J=1,8)
          ENDDO
        ENDIF  !END OF IF STATEMENT OF K (REM AND QUO LOOP)
        ENDDO


        ELSEIF(I_STATIC==3) THEN     !FREQUENCY
        ITYPE=1
        DO I=1,TOT_DOF
           KK(J,I)=KK(I,J)
           MM(J,I)=MM(I,J)
        ENDDO
        ALLOCATE(W(TOT_DOF))
        W=0.0D0
        LWORK=3*TOT_DOF-1
        ALLOCATE(WORK(LWORK))
        CALL DSYGV(ITYPE,'V','U',TOT_DOF,KK,TOT_DOF,MM,TOT_DOF,
       + W,WORK,LWORK,INFO)
        OPEN(55,FILE='EIGEN_VAL.DAT')
        DO I=1,TOT_DOF
           WRITE(55,21) W(I)
        ENDDO
        CLOSE(55)
c       DO I=1,TOT_DOF
c          IF (W(I).GT. 1.0D0) GOTO 120
c          VAR=I+1
c120    ENDDO
c       VAR=974
c       WRITE(*,*) 'FIRST FREQ', VAR

        OPEN(56,FILE='EIG_VEC.DAT')
        DO J=1,TOT_DOF
           WRITE(56,20) (KK(J,K), K=1,20)
        ENDDO
        CLOSE(56)
        WRITE(*,*) 'SOLUTION SUCCESFUL IF INFO = 0'
        WRITE(*,*) 'INFO - ',INFO

        ENDIF
        CLOSE(599)
        CLOSE(666)

!

!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!            NON-LINEAR ANALYSIS
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------


        ELSEIF (I_LIN==2) THEN    !FLAG FOR GEOMETRIC NONLINEARITY
          IF (I_VON==1) THEN
```

```
            WRITE(*,*) 'FULL GEOMETRIC NONLINEARITY'
          ELSEIF (I_VON==2) THEN
            WRITE(*,*) 'VON-KARMAN NONLINEARITY(CONST. REL. - S = C E'
          ELSEIF (I_VON==3) THEN
            WRITE(*,*) 'VON-KARMAN NONLINEARITY(CONST. REL. - SIG = C E'
          ENDIF
!-----------------------------------------------------------------------
!     CALLING MATRICES
!-----------------------------------------------------------------------
!     MASS MATRIX - FILE SUB_MATRICES.F
      ALLOCATE(MM(TOT_DOF,TOT_DOF))

      CALL M_MAT(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,RHO,LE,BE,ME,MM
     + ,N_FEM,E_NODE,N_L,N_B,N_NODE,CON)

!     BOUNDARY CONDITIONS
!     FILE SUB_BC.F
!     STARTING WITH MASS MATRIX
      CALL BC_APP_KM(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
     +  N_NODE,I_STATIC,I_BC,MM)
      OPEN(921,FILE='CONNECTIVITY.DAT')
      DO I=1,N_FEM
         WRITE(921,23) (CON(I,J),J=1,E_DOF)
      ENDDO

!-----------------------------------------------------------------------
!
!                     MASS MATRIX OUTPUT
!
!-----------------------------------------------------------------------
      IF(DEBUG) THEN
      OPEN(11,FILE='MASS_MAT_OUT.DAT')
      OPEN(66,FILE='MASS_MAT.DAT')
      DO I=1,E_DOF
         WRITE(66,20) (ME(I,J), J=1,E_DOF)
      ENDDO
      CLOSE(66)
!      WRITE(11,*) 'ELEMENT MASS MATRIX - LUMPED'
!      DO I=1,E_DOF
!         WRITE(11,20) (MEL(I,J), J=1,E_DOF)
!      ENDDO
      DO I=1,TOT_DOF
         WRITE(11,20) (MM(I,J), J=1,TOT_DOF)
      ENDDO
      CLOSE(11)
      ENDIF
!-----------------------------------------------------------------------
!     SPARSE ASSEMBLY
      NUM=0
      DO I=1,TOT_DOF
         DO J=1,TOT_DOF
             IF (MM(I,J) .NE. 0.0D0) THEN
                 NUM=NUM+1
                 NZ_M=NUM
             ENDIF
         ENDDO
      ENDDO
c      WRITE(*,*) 'NUMBER OF NONZEROS', NZ_M
      ALLOCATE(MG_VAL(NZ_M),COL_M(NZ_M),ROWIND_M(TOT_DOF+1))
      MG_VAL=0.0D0;ROWIND_M=0
      COL_M=0
      CALL SPARSE_ASSM(MG_VAL,TOT_DOF,NZ_M,
     + MM,ROWIND_M,COL_M)

      DEALLOCATE (MM)
!-----------------------------------------------------------------------


!     MASS MATRIX FOR ENERGY FILE SUB_MATRICES.F
```

```
1105         ALLOCATE(MM(TOT_DOF,TOT_DOF))
1106         MM=0.0D0
1107
1108         CALL M_MAT(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,RHO,LE,BE,ME,MM
1109        + ,N_FEM,E_NODE,N_L,N_B,N_NODE,CON)
1110
1111   !     BOUNDARY CONDITIONS
1112   !     FILE SUB_BC.F
1113   !     STARTING WITH MASS MATRIX
1114   c      CALL BC_APP_KM(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
1115   c     +  N_NODE,I_STATIC,I_BC,MM)
1116
1117   !---------------------------------------------------------------------------
1118   !     SPARSE ASSEMBLY
1119         NUM=0
1120         NZ_M2=0
1121         DO I=1,TOT_DOF
1122            DO J=1,TOT_DOF
1123                IF (MM(I,J) .NE. 0.0D0) THEN
1124                    NUM=NUM+1
1125                    NZ_M2=NUM
1126                ENDIF
1127            ENDDO
1128         ENDDO
1129   c      WRITE(*,*) 'NUMBER OF NONZEROS', NZ_M
1130         ALLOCATE(MG_VAL_ENER(NZ_M2),COL_M_ENER(NZ_M2),
1131        + ROWIND_M_ENER(TOT_DOF+1))
1132         MG_VAL_ENER=0.0D0;ROWIND_M_ENER=0
1133         COL_M_ENER=0
1134         CALL SPARSE_ASSM(MG_VAL_ENER,TOT_DOF,NZ_M2,
1135        + MM,ROWIND_M_ENER,COL_M_ENER)
1136
1137         DEALLOCATE (MM)
1138
1139
1140
1141
1142
1143
1144
1145   !     LOAD VECTOR - FILE SUB_MATRICES.F
1146         IF (I_LOAD==1) THEN
1147            WRITE(*,*) 'CONSTANT PRESSURE LOADING'
1148         ELSEIF (I_LOAD==2) THEN
1149            WRITE(*,*) 'SPATIALLY DEPENDENT LOAD'
1150         ELSE
1151            WRITE(*,*) 'INVALID LOADING TYPE'
1152            GOTO 1000
1153         ENDIF
1154         CALL LOAD_VEC(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,LE,BE,FE,
1155        + FF,APP_LOAD,N_FEM,E_NODE,N_NODE,N_L,N_B,I_LOAD,XE,YE,L,B)
1156   !     LOAD VECTOR OUTPUT
1157   C      IF (DEBUG) THEN
1158            OPEN(19,FILE='LOAD VECTOR.DAT')
1159            DO I=1,TOT_DOF
1160                WRITE(19,21) FF(I)
1161            ENDDO
1162   C      ENDIF
1163
1164
1165         IF (I_STATIC==1) THEN
1166            WRITE(*,*) 'MASS SCALING FOR NONLINEAR STATIC ANALYSIS'
1167            DT=0.0005D0
1168   !         GOTO 1125
1169
1170
1171
1172
1173
```

```
      ELSEIF (I_STATIC==2) THEN

          WRITE(*,*) 'NONLINEAR DYNAMIC SOLUTION'
!     ALLOCATONG DIMESIONS
!     STARTING WITH NODAL VELOCITIES AND ACCELERATIONS




!         TIME STEP SIZE
          DT=0.05D0
C          DT=0.000002D0
!1125      CONTINUE




!         ALLOCATING DIMENSIONS TO THE MATRICES
          ALLOCATE(V10(N_NODE),V11(N_NODE),V12(N_NODE),V13(N_NODE))
          ALLOCATE(V20(N_NODE),V21(N_NODE),V22(N_NODE),V23(N_NODE))
          ALLOCATE(V30(N_NODE),V31(N_NODE),V32(N_NODE),V33(N_NODE))
          ALLOCATE(A10(N_NODE),A11(N_NODE),A12(N_NODE),A13(N_NODE))
          ALLOCATE(A20(N_NODE),A21(N_NODE),A22(N_NODE),A23(N_NODE))
          ALLOCATE(A30(N_NODE),A31(N_NODE),A32(N_NODE),A33(N_NODE))
          ALLOCATE(SOLN1(TOT_DOF))
!     VELOCITY AND ACCELERATION VECTORS RESPECTIVEY FOR THE NTH AND
!     (N+1)TH TIME STEP
          ALLOCATE(VN(TOT_DOF),AN(TOT_DOF),VN1(TOT_DOF),AN1(TOT_DOF))
!     LOAD VECTORS INTERNAL AND F_EXT(N+1)
          ALLOCATE(FFIN(TOT_DOF),FFN1(TOT_DOF))
          UPLO='U' !UPPER TRAINGLE OF MASS MATRIX STORED IN SPARSE FORMAT
!
!
!
          WRITE(*,*) 'TIME STEP SIZE [muS]', DT
          WRITE(*,*) 'END TIME', TSTOP      !FINAL TIME
          TSTEPS=TSTOP/DT+1       !NUMBER OF TIME STEPS
c      tsteps=1
        WRITE(*,*) 'TIME STEPS', TSTEPS
C      A1=DT*GAMMA; A2=DT*(1-GAMMA)
C      A3=1/BETA/DT**2; A4=1/BETA/DT; A5=(1-2*BETA)/2/BETA
C      WRITE(51,*) A1,A2,A3,A4,A5
          !INITIALIZING DISPLACEMENT, VELOCITY AND ACC.
          SOL=0.0D0; VN=0.0D0; AN=0.0D0
          SOLN1=0.0D0; VN1=0.0D0; AN1=0.0D0
!     LOAD = 0 AT T=0
          NRHS=1
!     FIRST STEP - TIME =0
          FFN1=0.0D0;     FFIN=0.0D0
!     TOTAL POINTS IN EACH ELEMENT FOR STRESS STRAIN CALCULATIONS
          T_P=G_X*G_Y*G_Z



              IF(OUTPUT) THEN
              OPEN(16,FILE='STRAIN.DAT')
              OPEN(17,FILE='STRESS.DAT')
              ENDIF
              OPEN(917,FILE='STRESS_OUT.DAT')
              OPEN(918,FILE='STRAIN_OUT.DAT')
              OPEN(919,FILE='DEFORMED_SHAPE.DAT')
              OPEN(557,FILE='TRACTION.DAT')



      WD=0.0D0      !TOTAL EXTERNAL WORK AT THE INTIAL COND.
```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!     TIME ALGORITHM STARTS
      DO K=1,TSTEPS
         TIME=T0+(K-1)*DT

         ALLOCATE(FF1(TOT_DOF,1),FFIN1(TOT_DOF,1))

         IF (K==1) THEN
!        SOLVING FOR A0
         SOL=0.0D0
         VN=0.0D0
         CALL SOLVER_F77_TEST(ROWIND_M,COL_M,MG_VAL,FFN1,TOT_DOF,
     +   TOT_DOF, NZ_M, NRHS,AN)


         ELSE     !K=2,TSTEPS
!        D(N+1)=DN+DT*VN+DT^2/2*AN
         SOLN1=SOL+DT*VN+DT**2.0D0/2.0D0*AN
!        GETTING F_INT


!!!!!!!!!!!!!!!!!!!!MANUFACTURED DISPLACEMENT!!!!!!!!!!!!!!!!!!!!!!!!!
               IF(MANU) THEN
!          MANUFACTURED SOLUTIONS
C           WRITE(*,*) 'MANUFACTURED SOLUTION '
!          VALUES OF THE DEGREES OF FREEDOM OF EACH NODE
           SOLN1=0.0d0
C             DO I=1,N_NODE
C             SOLN1(I)=SIN(PI*GLOB_XYZ(I,1)/L)*SIN(PI*GLOB_XYZ(I,2)/B)
C      +       *1.0d0
C             SOLN1(N_NODE+I)=SIN(2.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(2.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-1.0d0
C             SOLN1(N_NODE*2+I)=SIN(3.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(3.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-2.0d0
C             SOLN1(N_NODE*3+I)=SIN(4.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(4.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-3.0d0
C             SOLN1(N_NODE*4+I)=SIN(PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(PI*GLOB_XYZ(I,2)/B)*1.0d0
C             SOLN1(N_NODE*5+I)=SIN(2.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(2.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-1.0d0
C             SOLN1(N_NODE*6+I)=SIN(3.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(3.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-2.0d0
C             SOLN1(N_NODE*7+I)=SIN(4.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(4.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-3.0d0
C             SOLN1(N_NODE*8+I)=SIN(PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(PI*GLOB_XYZ(I,2)/B)*1.0d0
C             SOLN1(N_NODE*9+I)=SIN(2.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(2.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-1.0d0
C             SOLN1(N_NODE*10+I)=SIN(3.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(3.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-2.0d0
C             SOLN1(N_NODE*11+I)=SIN(4.0D0*PI*GLOB_XYZ(I,1)/L)
C      +      *SIN(4.0D0*PI*GLOB_XYZ(I,2)/B)*10.0d0**-3.0d0
C             ENDDO
             DO I=1,N_NODE
             SOLN1(I)=10.0D0**-2
             SOLN1(N_NODE+I)=GLOB_XYZ(I,1)*10.0D0**-2
             SOLN1(N_NODE*2+I)=GLOB_XYZ(I,2)*10.0D0**-4
             SOLN1(N_NODE*3+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)*10.0D0**-6
             SOLN1(N_NODE*4+I)=10.0D0**-2
             SOLN1(N_NODE*5+I)=GLOB_XYZ(I,1)*10.0D0**-2
             SOLN1(N_NODE*6+I)=GLOB_XYZ(I,2)*10.0D0**-4
             SOLN1(N_NODE*7+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)*10.0D0**-6
             SOLN1(N_NODE*8+I)=10.0D0**-2
             SOLN1(N_NODE*9+I)=GLOB_XYZ(I,1)*10.0D0**-2
             SOLN1(N_NODE*10+I)=GLOB_XYZ(I,2)*10.0D0**-4
             SOLN1(N_NODE*11+I)=GLOB_XYZ(I,1)*GLOB_XYZ(I,2)*10.0D0**-6
```

```
              ENDDO
             ENDIF
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

                IF (DEBUG) THEN
                       OPEN(28,FILE='ELE_INT.DAT')
                ENDIF



           CALL INT_FORCE (TOT_DOF,H,N_FEM,N_NODE,
     + NODE_L,NODE_B,E_DOF,G_X,G_Y,G_Z,T_P,LE,BE,
     + NODAL_DISP,CC,XE,YE,E_NODE,I_ORDER,TIME,SOLN1,CON,NL2,NB2,NH2,
     + N_NODE2,N_L,N_B,K,CON2,CON3,FFIN,I_VON,TU,SRS_ID,AN,GLOB_XYZ)

           write(*,*) 'time step', k
                IF (OUTPUT) THEN
                       OPEN(599,FILE='INTERNAL_FORCE_N.DAT')
                       DO I=1,TOT_DOF
                              WRITE(599,20) TIME,FFIN(I)
                       ENDDO
                ENDIF
!         GETTING F(N+1)

            IF (I_TRAC==2) THEN    !IF STATEMENT FOR KIND OF LOADING
            CALL LOAD_DYNA(K,DT,T0,FF,TOT_DOF,THETA,FFN1,TSTEPS,APP_LOAD)
                IF (OUTPUT) THEN
                OPEN(666,FILE='TIME_LOAD.DAT')
                DO I=1,TOT_DOF
                       WRITE(666,20) TIME,FFN1(I)
                ENDDO
                ENDIF
            ELSEIF (I_TRAC==1) THEN
            CALL PRESS_LOAD(E_DOF,TOT_DOF,I_ORDER,H,LE,BE,CON,
     +   SOLN1,FFN1,APP_LOAD,N_FEM,E_NODE,N_NODE,N_L,N_B,I_LOAD,XE,YE
     +  ,L,B,K,T0,DT,THETA,G_X,G_Y)
            ENDIF


!       NET FORCE F_EXT - F_INT
!

          FF1(:,1)=FFN1(:)       !FOR ENERGY CALCULATION
          FFIN1(:,1)=FFIN(:)     !FOR ENERGY CALCULATION

         FFN1=FFN1-FFIN

!       SUBSTITUTING BOUNDARY CONDITIONS IN FFN1
           CALL BC_APP_FF(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
     +     N_NODE,I_STATIC,I_BC,FFN1)
!          SOLVING FOR A(N+1) M*A(N+1)=F(N+1)-Fin(N+1)
           CALL SOLVER_F77_TEST(ROWIND_M,COL_M,MG_VAL,FFN1,TOT_DOF,
     +     TOT_DOF, NZ_M, NRHS,AN1)
!          GETTING V(N+1)=VN+DT*(AN/2+A(N+1)/2)
           VN1=VN+DT*(AN/2.0D0+AN1/2.0D0)
           ENDIF
c           if (k<5) then
c           write(999,*) 'step', k
c           do i=1,tot_dof
c               write(999,20) sol(i),soln1(i),vn(i),vn1(i),an(i),an1(i)
c           enddo
c           endif


!----------------------------------------------------------------------
!          CALCULATION OF ENERGIES
!
!
!
```

```
1381
1382            ALLOCATE(MV(TOT_DOF))
1383            ALLOCATE(VEL_MAT(TOT_DOF,1),DISP_MAT(TOT_DOF,1))
1384            ALLOCATE(DISP_MAT2(TOT_DOF,1))
1385            ALLOCATE(MVMAT(TOT_DOF,1))
1386            MV=0.0D0
1387
1388
1389            OPEN(621,FILE='ENERGIES.DAT')
1390
1391
1392            DISP_MAT(:,1)=SOLN1(:)
1393            DISP_MAT2(:,1)=SOL(:)
1394            VEL_MAT(:,1)=VN1(:)
1395            CALL MKL_DCSRSYMV(UPLO,TOT_DOF,MG_VAL_ENER,ROWIND_M_ENER,
1396           +    COL_M_ENER,VN1,MV)
1397            MVMAT(:,1)=MV(:)
1398            KINE=1.0D0/2.0D0*MATMUL(TRANSPOSE(VEL_MAT),MVMAT)
1399
1400    C        WD=0.0D0
1401            WDI=0.0D0
1402
1403    C        WDI=MATMUL(TRANSPOSE(DISP_MAT),FF1)
1404            WDI=MATMUL(TRANSPOSE(DISP_MAT),FF1)
1405           +    -MATMUL(TRANSPOSE(DISP_MAT2),FF1)
1406
1407
1408            WD=WD+WDI
1409
1410    C        UE=1.0D0/2.0D0*MATMUL(TRANSPOSE(DISP_MAT),FFIN1)
1411
1412            DEALLOCATE(MV,VEL_MAT,DISP_MAT,MVMAT,FF1,FFIN1,DISP_MAT2)
1413
1414            WRITE(621,20) TIME,KINE,TU,WD
1415
1416    !-------------------------------------------------------------------------
1417    !       OVERWRITING Nth VAL. BY N+1th VAL.
1418            AN=AN1
1419            VN=VN1
1420            SOL=SOLN1
1421
1422    c
1423    !       TIME AT THE Kth STEP
1424
1425    c        write(*,*) 'time step and time', k,time
1426    c        pause
1427    !       GENERATING OUTPUTS OF DISP, VEL, ACC AND CALCULATING STRESSES
1428
1429
1430
1431             QUO=K/1
1432             REM=K-QUO*1
1433            IF (K==1 .OR. REM==0) THEN
1434    c        write(*,*) 'output generated', k
1435    c        pause
1436    c        SOL=SOL*100*E*H**3.0D0/12.0D0/APP_LOAD/L**4.0D0/
1437    c     +   (1-0.3D0**2.0D0)
1438            DO I=1,N_NODE
1439                U10(I)=SOL(I)
1440                U11(I)=SOL(N_NODE+I)
1441                U12(I)=SOL(2*N_NODE+I)
1442                U13(I)=SOL(3*N_NODE+I)
1443                U20(I)=SOL(4*N_NODE+I)
1444                U21(I)=SOL(5*N_NODE+I)
1445                U22(I)=SOL(6*N_NODE+I)
1446                U23(I)=SOL(7*N_NODE+I)
1447                U30(I)=SOL(8*N_NODE+I)
1448                U31(I)=SOL(9*N_NODE+I)
1449                U32(I)=SOL(10*N_NODE+I)
```

```
1450                U33(I)=SOL(11*N_NODE+I)
1451                V10(I)=VN(I)
1452                V11(I)=VN(N_NODE+I)
1453                V12(I)=VN(2*N_NODE+I)
1454                V13(I)=VN(3*N_NODE+I)
1455                V20(I)=VN(4*N_NODE+I)
1456                V21(I)=VN(5*N_NODE+I)
1457                V22(I)=VN(6*N_NODE+I)
1458                V23(I)=VN(7*N_NODE+I)
1459                V30(I)=VN(8*N_NODE+I)
1460                V31(I)=VN(9*N_NODE+I)
1461                V32(I)=VN(10*N_NODE+I)
1462                V33(I)=VN(11*N_NODE+I)
1463                A10(I)=AN(I)
1464                A11(I)=AN(N_NODE+I)
1465                A12(I)=AN(2*N_NODE+I)
1466                A13(I)=AN(3*N_NODE+I)
1467                A20(I)=AN(4*N_NODE+I)
1468                A21(I)=AN(5*N_NODE+I)
1469                A22(I)=AN(6*N_NODE+I)
1470                A23(I)=AN(7*N_NODE+I)
1471                A30(I)=AN(8*N_NODE+I)
1472                A31(I)=AN(9*N_NODE+I)
1473                A32(I)=AN(10*N_NODE+I)
1474                A33(I)=AN(11*N_NODE+I)
1475             ENDDO
1476              OPEN(14,FILE='RESULT.DAT')
1477              WRITE(14,*) 'TITLE = "DISPLACEMENT"'
1478              WRITE(14,*) 'VARIABLES = x ,y, "u10", "u11", "U12", "U13",
1479         + "U20", "U21","U22", "U23", "U30", "U31", "U32", "U33"'
1480              WRITE(14,*) 'ZONE T= "Undeformed Mesh",N =',N_NODE,'E =',
1481         + N_FEM,', F=FEPOINT SOLUTIONTIME= ',TIME
1482              DO I=1,N_NODE
1483                 WRITE(14,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),U10(I),U11(I),
1484         +       U12(I),U13(I),U20(I),U21(I),U22(I),U23(I),U30(I),
1485         +       U31(I),U32(I),U33(I)
1486              ENDDO
1487              DO I=1,N_FEM
1488                 WRITE(14,23) (CON(I,J), J=1,4)
1489              ENDDO
1490   c           CLOSE(14)
1491              OPEN(39,FILE='VELOCITIES.DAT')
1492              DO I=1,N_NODE
1493                 WRITE(39,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),V10(I),V11(I),
1494         +       V12(I),V13(I),V20(I),V21(I),V22(I),V23(I),V30(I),V31(I)
1495         +      ,V32(I),V33(I),TIME
1496              ENDDO
1497   c           CLOSE(39)
1498              OPEN(40,FILE='ACCELERATION.DAT')
1499              DO I=1,N_NODE
1500                 WRITE(40,20) GLOB_XYZ(I,1),GLOB_XYZ(I,2),A10(I),A11(I),
1501         +       A12(I),A13(I),A20(I),A21(I),A22(I),A23(I),A30(I),A31(I)
1502         +       ,A32(I),A33(I),TIME
1503              ENDDO
1504   c           CLOSE(40)
1505
1506
1507
1508             ENDIF
1509           ENDDO
1510   !
1511   !     TIME LOOP
1512   !
1513
1514
1515
1516
1517   !     I_STATIC END OF LOOP
1518           ENDIF
```

```
!     I_LIN END OF LOOP
      ENDIF


      CONTINUE
!
      FORMAT (9999(x,EN14.4))
      FORMAT(EN12.3)
      FORMAT(F5.0)
      FORMAT (999(2X,I5))
!

!
      WRITE(*,*) '---------------------------------'
      WRITE(*,*) '          RUN COMPLETED          '
      WRITE(*,*) '---------------------------------'
!
      END PROGRAM TSNDT_CODE




      SUBROUTINE INPUT (I_CURVE,L,B,H,CC,RHO,APP_LOAD,N_L,N_B,I_ORDER,
     + I_BC,I_LIN,I_THEORY,G_X,G_Y,G_Z,I_STATIC,THETA,T0,TSTOP
     + ,BETA,GAMMA,E,I_LOAD,I_VON,I_TRAC,SRS_ID)
!
      IMPLICIT NONE
      INTEGER :: I_CURVE,I,J
      REAL(KIND=8) :: L,B,H,CC(6,6),RHO,APP_LOAD,THETA,CM(6,6),T(6,6)
!     CC IS THE STIFFNESS MATRIX IN GLOBAL COORDINATES
!     CM IS THE STIFFNESS MATRIX IN MATERIAL COORDINATES
      INTEGER :: N_L,N_B,I_ORDER,I_BC,I_LIN,I_THEORY,G_X,G_Y,G_Z
      INTEGER :: I_STATIC,I_EIG, I_LOAD,I_VON,I_TRAC,SRS_ID
      REAL(KIND=8) :: T0,TSTOP,BETA, GAMMA
      REAL(KIND=8) :: E1,E2,E3,G12,G13,G23,NU12,NU13,NU23,NU21,NU32
     +  ,NU31,DELTA,ALPHA
      REAL(KIND=8) :: PI,E   ! VALUE OF PI AND MAXIMUM YOUNGS MOD. RESP.

      CC=0.0D0
      CM=0.0D0

!
      OPEN(100,FILE='input.dat')
      READ(100,*)
      READ(100,*)
      READ(100,*) I_CURVE       !I_CURVE  - SEE MODULE 'PLATE_CONST" FOR THE
      READ(100,*)                     !DEFINITIONS OF THESE VARIABLES
      READ(100,*) L     !LENGTH
      READ(100,*)
      READ(100,*) B     !WIDTH
      READ(100,*)
      READ(100,*) H     !HEIGHT
      READ(100,*)
      READ(100,*)
      READ(100,21) E1
      READ(100,*)
      READ(100,21) E2
      READ(100,*)
      READ(100,21) E3
      READ(100,*)
      READ(100,21) G12
      READ(100,*)
      READ(100,21) G13
      READ(100,*)
      READ(100,21) G23
      READ(100,*)
      READ(100,*) NU12
```

```
      READ(100,*)
      READ(100,*) NU13
      READ(100,*)
      READ(100,*) NU23
      READ(100,*)
      READ(100,*) ALPHA
      READ(100,*)
      READ(100,*) RHO       !DENSITY
      READ(100,*)
      READ(100,*) APP_LOAD        !APPLIED NORMAL UDL
      READ(100,*)
      READ(100,*) I_LOAD   !SPATIAL LOAD DEPENDENCY, 1 = INDEP. 2 - DEP
      READ(100,*)
      READ(100,*) I_TRAC  !FOLLOWER LOAD ON DEFORMED/ LOAD ON UNDEFORMED
      READ(100,*)
      READ(100,*) N_L   !N_L = NUMBER OF ELEMENTS ALONG LENGTH
      READ(100,*)
      READ(100,*) N_B   !N_B = NUMBER OF ELEMENTS ALONG WIDTH
      READ(100,*)
      READ(100,*)
      READ(100,*) I_ORDER   !I_ORDER - 1 - LINEAR SHAPE FUNCT. (2 NODED), 2 -
      QUADRATIC(3 NODED)
      READ(100,*)
      READ(100,*) I_BC   !I_BC - BOUNDARY CONDITIONS 1=SIMPLY SUPPORED 2-CLAMPED
      READ(100,*)
      READ(100,*) I_LIN   !I_LIN - 1 = GEOMETRICALLY LINEAR, 2 = GEOMETRIC NONLINEARITY
      READ(100,*)
      READ(100,*) I_VON
      READ(100,*)
      READ(100,*) I_THEORY   !THEORY ORDER - 1 - 1ST ORDER, 3-3RD ORDER
      READ(100,*)
      READ(100,*) G_X   !INTEGRATION POINTS IN X
      READ(100,*)
      READ(100,*) G_Y   !INTEGRATION POINTS IN Y
      READ(100,*)
      READ(100,*) G_Z   !INTEGRATION POINTS IN Z
      READ(100,*)
      READ(100,*) I_STATIC
      READ(100,*)
      READ(100,*) THETA
      READ(100,*)
      READ(100,*) T0
      READ(100,*)
      READ(100,*) TSTOP
      READ(100,*)
      READ(100,*) SRS_ID
      CLOSE(100)
!

      NU21=E2/E1*NU12
      NU31=E3/E1*NU13
      NU32=E3/E2*NU23
      DELTA=(1-NU12*NU21-NU23*NU32-NU31*NU13-2.0D0*NU21*NU32*NU13)
     + /(E1*E2*E3)

      CM(1,1)=(1.0D0-NU23*NU32)/(E2*E3*DELTA)
      CM(1,2)=(NU21+NU31*NU23)/(E2*E3*DELTA)
      CM(1,3)=(NU31+NU21*NU32)/(E2*E3*DELTA)
      CM(2,2)=(1.0D0-NU13*NU31)/(E1*E3*DELTA)
      CM(2,3)=(NU32+NU12*NU31)/(E1*E3*DELTA)
      CM(3,3)=(1.0D0-NU12*NU21)/(E1*E2*DELTA)
      CM(3,1)=CM(1,3)
      CM(3,2)=CM(2,3)
      CM(2,1)=CM(1,2)
      CM(4,4)=G23; CM(5,5)=G13; CM(6,6)=G12
!
c      CM(1,1)=100.0E+9
c      CM(1,2)=23.319E+9
c      CM(1,3)=1.07E+9
```

```
c       CM(2,2)=54.310E+9
c       CM(2,3)=9.827E+9
c       CM(3,3)=53.0172E+9
c       CM(3,1)=CM(1,3)
c       CM(3,2)=CM(2,3)
c       CM(2,1)=CM(1,2)
c       CM(4,4)=26.68E+9; CM(5,5)=15.99E+9; CM(6,6)=26.29E+9
!
        PI=4*DATAN(1.0D0)
c       ALPHA=ALPHA*PI/180.0D0
!
        T=0.0D0
        T(1,1)=DCOSD(ALPHA)**2;T(1,2)=DSIND(ALPHA)**2
        T(1,6)=-DSIND(2*ALPHA)
        T(2,1)=T(1,2); T(2,2)=T(1,1); T(2,6)=-T(1,6)
        T(3,3)=1
        T(4,4)=DCOSD(ALPHA);T(4,5)=DSIND(ALPHA)
        T(5,4)=-DSIND(ALPHA);T(5,5)=DCOSD(ALPHA)
        T(6,1)=DSIND(ALPHA)*DCOSD(ALPHA);T(6,2)=-DSIND(ALPHA)*DCOSD(ALPHA)
        T(6,6)=DCOSD(ALPHA)**2-DSIND(ALPHA)**2.0D0
!
        CC=MATMUL(T,MATMUL(CM,TRANSPOSE(T)))

        IF (E1 .NE. E2 .OR. E1 .NE. E3) THEN
           WRITE(*,*) 'ANISOTROPIC PLATE'
        ELSE
           WRITE(*,*) 'ISOTROPIC PLATE'
        ENDIF

        E=E1


!
        FORMAT(EN11.2)
        FORMAT (6F10.4)
        FORMAT (9999(x,EN14.4))

        RETURN
        END SUBROUTINE INPUT
!


!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!
!                                LOAD VECTOR
!
!
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!
        SUBROUTINE LOAD_VEC(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,LE,BE,FE,
       + FF,APP_LOAD,N_FEM,E_NODE,N_NODE,N_L,N_B,I_LOAD,XE,YE,LENGTH,
       + BREADTH)
        USE PROG_DEBUG
        IMPLICIT NONE
!-----------------------------------------------------------------------
        INTEGER :: I,J,K,L,Q     !LOOP COUNTERS
        INTEGER :: E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,N_FEM,E_NODE,N_L,N_B
       + ,N_NODE,NODE_L,NODE_B, I_LOAD
!-----E_DOF - ELEMENTAL DOF, TOT_DOF - TOTAL DOF, I_ORDER - SHAPE FUNC.
!-----G_X - NO. OF INT. POINTS, N_FEM - NUMBER OF ELE, E_NODE - NO. OF
!-----NODES PER ELEMENT, BOUNDARY CONDITION - I_BC, SPATIAL DEPENDENT
!     LOAD - I_LOAD
        REAL(KIND=8) :: H,LE,BE,APP_LOAD,LENGTH,BREADTH
!-----HEIGHT - H, LE - LENGTH OF THE ELEMENTS
!-----APP_LOAD IS THE APPLIED NORMAL UDL
        REAL(KIND=8) :: FE(E_DOF),FF(TOT_DOF)
```

```
!     FE IS ELEMENTAL LOAD VECTOR, FF IS GLOBAL LOAD VECTOR
      REAL(KIND=8),ALLOCATABLE :: X_P(:),X_W(:),Y_P(:),Y_W(:)
!-    GAUSS POINTS AND WEIGHTS IN X AND Y
      REAL(KIND=8) :: XE(N_FEM,4),YE(N_FEM,4)
      REAL(KIND=8) :: N1,N2,N3,N4,DET_J
!-----N1, N2, N3 - VALUES OF THE SHAPE FUNCTIONS AT THE GAUSS POINTS
      REAL(KIND=8), ALLOCATABLE :: FE_1(:),FF_1(:)
!-----TEMPORARY VECTORS FOR CALCULATING THE LOAD VECTOR
      REAL(KIND=8) :: S13_T,S13_B,S23_T,S23_B,S33_T,S33_B
!-----BOUNDARY TRACTIONS
      REAL(KIND=8) :: T10,T11,T12,T13,T20,T21,T22,T23,T30,T31,T32,T33
      INTEGER,ALLOCATABLE :: C(:,:),CON(:,:)    !CONNECTIVITY
      REAL(KIND=8) :: FUNCT(G_X,G_Y,4),N(4,1)
      REAL(KIND=8) :: T(12)
      REAL(KIND=8) :: W_X(1,G_X),W_Y(G_Y,1)
      REAL(KIND=8) :: M_FUN(G_X,G_Y)
      REAL(KIND=8) :: V(1,1)
      REAL(KIND=8) :: INTEGRAL(4),A(4)
      REAL(KIND=8) :: COR_X,COR_Y   !COORD. OF THE CENT. OF EACH ELE.
      REAL(KIND=8) :: PI      !VALUE OF PI
      REAL(KIND=8) :: RADI    ! RADIUS OF THE PRESSURE KERNEL

      PI=4.0D0*DATAN(1.0D0)

!---------------------------------------------------------------------
      ALLOCATE (X_P(G_X),X_W(G_X))
      ALLOCATE (Y_P(G_Y),Y_W(G_Y))
      ALLOCATE(FE_1(E_DOF))
      ALLOCATE (C(E_DOF,N_FEM))
      ALLOCATE(CON(N_FEM,E_DOF))
!
      CALL GAUSS_LEG(G_X,X_P,X_W)
      CALL GAUSS_LEG(G_Y,Y_P,Y_W)
      FE=0.0D0; FF=0.0D0


      FE_1=0.0D0;
!
!
!     SPATIAL DEPENDENCY OF LOAD
!
!
      IF (I_LOAD==1) THEN       !SPATIALLY INDEPENDENT LOAD

!     VALUES OF BOUNDARY TRACTIONS
      S13_T=APP_LOAD;S13_B=0.0D0;S33_T=0.0D0;S33_B=0.0D0
      S23_T=0.0D0; S23_B=0.0D0
c
c      s13_t=app_load;s13_b=app_load/2.0d0;s33_t=app_load
c      s33_b=app_load/2.0d0;s23_t=app_load; s23_b=app_load/2.0d0
c
      T10=S13_T-S13_B; T11=H/2.0D0*(S13_T+S13_B)
      T12=H**2/4.0D0*(S13_T-S13_B); T13=H**3/8.0D0*(S13_T+S13_B)
      T20=S23_T-S23_B; T21=H/2.0D0*(S23_T+S23_B)
      T22=H**2/4.0D0*(S23_T-S23_B); T23=H**3/8.0D0*(S23_T+S23_B)
      T30=S33_T-S33_B; T31=H/2.0D0*(S33_T+S33_B)
      T32=H**2/4.0D0*(S33_T-S33_B); T33=H**3/8.0D0*(S33_T+S33_B)
      T(1)=T10; T(2)=T11;T(3)=T12;T(4)=T13
      T(5)=T20; T(6)=T21;T(7)=T22;T(8)=T23
      T(9)=T30; T(10)=T31;T(11)=T32;T(12)=T33
c      !!!write(51,*) 'LOAD VECTOR SUBROUTINE'
c      !!!write(51,*) APP_LOAD
!-----INTEGRATING FOR THE LOCAL LOAD VECTOR
      DO I = 1,G_X
         DO J=1,G_Y
              CALL FEM_SHAPE(X_P(I),Y_P(J),I_ORDER
     +       ,N1,N2,N3,N4,DET_J,LE,BE)
              N(1,1)=N1; N(2,1)=N2; N(3,1)=N3; N(4,1)=N4
              DO K=1,4
```

```
                    FUNCT(I,J,K)=N(K,1)
                 ENDDO
              ENDDO
           ENDDO
           DO K=1,4
              DO I=1,G_X
                   write(51,20) (FUNCT(I,J,K), J=1,G_Y)
              ENDDO
           ENDDO

           DO I=1,G_X
              W_X(1,I)=X_W(I)
           ENDDO
           DO I=1,G_Y
              W_Y(I,1)=X_W(I)
           ENDDO
! DEBUGGING
c       !!!write(51,20) W_X
c       !!!write(51,20) W_Y
 !
           DO I=1,4
              M_FUN=0.0D0
              DO J=1,G_X
                   DO K=1,G_Y
                        M_FUN(J,K)=FUNCT(J,K,I)
                   ENDDO
              ENDDO
c          !!!write(51,*) 'M_FUN'
c          !!!write(51,*) I
              DO L=1,G_X
c               !!!write(51,20) (M_FUN(L,Q), Q=1,G_Y)
              ENDDO
              V=MATMUL(MATMUL(W_X,M_FUN),W_Y)
              INTEGRAL(I)=V(1,1)
           ENDDO
           ! DEBUGGING
c       !!!write(51,*) INTEGRAL
c       !!!write(51,*) DET_J
           !
           DO I=1,4
              A(I)=DET_J*INTEGRAL(I)
           ENDDO
           !DEBUGGING
c       !!!write(51,20) A
           !

           DO K=1,12
              DO I=1,4
                   FE((K-1)*4+I)=T(K)*A(I)
              ENDDO
           ENDDO

!-------------------------------------------------------------------------

!-----CONNECTIVITY MATRIX CONSTRUCTION-----------------------------------
           DO K=1,N_B
               DO I=1,N_L
                  DO J=1,E_DOF/4.0D0
                  CON(K+I+(N_L-1)*K-N_L,4*J-3)=(J-1)*N_NODE+K+I+
     +            (N_L-1)*K-N_L+K-1;
                  CON(K+I+(N_L-1)*K-N_L,4*J-2)=(J-1)*N_NODE+K+I+
     +            (N_L-1)*K-N_L+1+K-1;
                  CON(K+I+(N_L-1)*K-N_L,4*J-1)=(J-1)*N_NODE+N_L+K+
     +            I+(N_L-1)*K-N_L+2+K-1;
                  CON(K+I+(N_L-1)*K-N_L,4*J)=(J-1)*N_NODE+N_L+K+I+
     +            (N_L-1)*K-N_L+1+K-1;
                  ENDDO
               ENDDO
           ENDDO
```

```
      C=TRANSPOSE(CON)
!-----------------------------------------------------------------------
      DO I=1,E_DOF
        DO J=1,N_FEM
        FF(C(I,J))=FF(C(I,J))+FE(I)
        ENDDO
      ENDDO
!-----------------------------------------------------------------------
!      OUTPUT IN DUMP.DAT
!      IF(DEBUG) THEN
c         !!!write(51,*) 'FF_1 IN LOAD VECTOR CALC'
c            DO K=1,TOT_DOF
c               !!!write(51,*) FF_1(K)
c            ENDDO
!      ENDIF
      NODE_L=N_L+1
      NODE_B=N_B+1
      IF(.FALSE.) THEN    !IF TRUE - LOADING ON HALF PLATE (Y=B/2 TO B)
      DO I=1,12
         DO J=1,NODE_L
            FF(N_NODE*(I-1)+NODE_L*(N_B*9/20)+J)=
     +          FF(N_NODE*(I-1)+NODE_L*(N_B*9/20)+J)/2.0D0
            FF(N_NODE*(I-1)+NODE_L*(N_B*11/20)+J)=
     +          FF(N_NODE*(I-1)+NODE_L*(N_B*11/20)+J)/2.0D0
            DO K=1,(N_B*9/20)
               FF(N_NODE*(I-1)+(K-1)*NODE_L+J)=0.0D0
            ENDDO
            DO K=(11*N_B/20)+2,NODE_B
               FF(N_NODE*(I-1)+(K-1)*NODE_L+J)=0.0D0
            ENDDO
         ENDDO
      ENDDO
      ENDIF
      IF (DEBUG) THEN
      OPEN(57,FILE='ELE_LOAD.DAT')
         DO I=1,E_DOF
            WRITE(57,23) FE(I)
         ENDDO
      ENDIF
!
!
!
!-----------------------------------------------------------------------
!          SPATIALLY DEPENDEDNT LOAD FORM
!-----------------------------------------------------------------------
!
!
!
!
      ELSEIF (I_LOAD==2) THEN
!
!-----CONNECTIVITY MATRIX CONSTRUCTION----------------------------------
      DO K=1,N_B
          DO I=1,N_L
             DO J=1,E_DOF/4.0D0
             CON(K+I+(N_L-1)*K-N_L,4*J-3)=(J-1)*N_NODE+K+I+
     +        (N_L-1)*K-N_L+K-1;
             CON(K+I+(N_L-1)*K-N_L,4*J-2)=(J-1)*N_NODE+K+I+
     +        (N_L-1)*K-N_L+1+K-1;
             CON(K+I+(N_L-1)*K-N_L,4*J-1)=(J-1)*N_NODE+N_L+K+
     +        I+(N_L-1)*K-N_L+2+K-1;
             CON(K+I+(N_L-1)*K-N_L,4*J)=(J-1)*N_NODE+N_L+K+I+
     +        (N_L-1)*K-N_L+1+K-1;
             ENDDO
          ENDDO
      ENDDO
      C=TRANSPOSE(CON)
      DO Q=1,N_FEM
         COR_X=(XE(Q,1)+XE(Q,2)+XE(Q,3)+XE(Q,4))/4.0D0
```

```
          COR_Y=(YE(Q,1)+YE(Q,2)+YE(Q,3)+YE(Q,4))/4.0D0
          WRITE(922,20) COR_X,COR_Y
          RADI=SQRT((COR_X-LENGTH/2.0D0)**2.0D0+
     +     (COR_Y-BREADTH/2.0D0)**2.0D0)
c          IF (RADI+SQRT(LE**2.0D0+BE**2.0D0)<LENGTH/2.0D0) THEN
          S13_T=0.0D0;S13_B=0.0D0;
          S33_T=-APP_LOAD*SIN(PI*COR_X/LENGTH)
c     +     *SIN(2.0D0*PI*COR_Y/BREADTH)
c          RADI=RADI/10.0D0
c          S33_T=(-0.0005D0*RADI**4+0.01D0*RADI**3-0.0586D0*RADI**2-
c     +    0.001D0*RADI+1.0D0)*APP_LOAD*1.0D0
          S33_B=0.0D0
          S23_T=0.0D0; S23_B=0.0D0
c          ELSE
c          S13_T=0.0D0;S13_B=0.0D0;
c          S33_T=0.0D0
c          S33_B=0.0D0
c          S23_T=0.0D0; S23_B=0.0D0
c          ENDIF
          OPEN(974,FILE='LOAD_PATT.DAT')
          WRITE(974,20) COR_X, COR_Y ,S33_T

!-----------------------------------------------------------------------
!        load vector formation
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
      T10=S13_T-S13_B; T11=H/2.0D0*(S13_T+S13_B)
      T12=H**2/4.0D0*(S13_T-S13_B); T13=H**3/8.0D0*(S13_T+S13_B)
      T20=S23_T-S23_B; T11=H/2.0D0*(S23_T+S23_B)
      T22=H**2/4.0D0*(S23_T-S23_B); T23=H**3/8.0D0*(S23_T+S23_B)
      T30=S33_T-S33_B; T31=H/2.0D0*(S33_T+S33_B)
      T32=H**2/4.0D0*(S33_T-S33_B); T33=H**3/8.0D0*(S33_T+S33_B)
      T(1)=T10; T(2)=T11;T(3)=T12;T(4)=T13
      T(5)=T20; T(6)=T21;T(7)=T22;T(8)=T23
      T(9)=T30; T(10)=T31;T(11)=T32;T(12)=T33
c      !!!write(51,*) 'LOAD VECTOR SUBROUTINE'
c      !!!write(51,*) APP_LOAD
!-----INTEGRATING FOR THE LOCAL LOAD VECTOR
      DO I = 1,G_X
         DO J=1,G_Y
              CALL FEM_SHAPE(X_P(I),Y_P(J),I_ORDER
     +       ,N1,N2,N3,N4,DET_J,LE,BE)
              N(1,1)=N1; N(2,1)=N2; N(3,1)=N3; N(4,1)=N4
              DO K=1,4
                 FUNCT(I,J,K)=N(K,1)
              ENDDO
         ENDDO
      ENDDO
      DO K=1,4
         DO I=1,G_X
             write(51,20) (FUNCT(I,J,K), J=1,G_Y)
         ENDDO
      ENDDO

      DO I=1,G_X
         W_X(1,I)=X_W(I)
      ENDDO
      DO I=1,G_Y
         W_Y(I,1)=X_W(I)
      ENDDO
! DEBUGGING
c      !!!write(51,20) W_X
c      !!!write(51,20) W_Y
 !
      DO I=1,4
         M_FUN=0.0D0
```

```
2001            DO J=1,G_X
2002                DO K=1,G_Y
2003                    M_FUN(J,K)=FUNCT(J,K,I)
2004                ENDDO
2005            ENDDO
2006    c        !!!write(51,*) 'M_FUN'
2007    c        !!!write(51,*) I
2008    C        DO L=1,G_X
2009    c            !!!write(51,20) (M_FUN(L,Q), Q=1,G_Y)
2010    C        ENDDO
2011            V=MATMUL(MATMUL(W_X,M_FUN),W_Y)
2012            INTEGRAL(I)=V(1,1)
2013          ENDDO
2014          ! DEBUGGING
2015    c      !!!write(51,*) INTEGRAL
2016    c      !!!write(51,*) DET_J
2017          !
2018          DO I=1,4
2019             A(I)=DET_J*INTEGRAL(I)
2020          ENDDO
2021          !DEBUGGING
2022    c      !!!write(51,20) A
2023          !
2024
2025          DO K=1,12
2026             DO I=1,4
2027                 FE((K-1)*4+I)=T(K)*A(I)
2028             ENDDO
2029          ENDDO
2030          IF (DEBUG) THEN
2031          OPEN(57,FILE='ELE_LOAD.DAT')
2032          WRITE(57,*) Q
2033             DO I=1,E_DOF
2034                 WRITE(57,23) FE(I)
2035             ENDDO
2036          ENDIF
2037
2038          DO I=1,E_DOF
2039                 FF(C(I,Q))=FF(C(I,Q))+FE(I)
2040          ENDDO
2041
2042
2043          ENDDO
2044
2045          ELSE
2046
2047          WRITE(*,*) 'INVALID LOADING PARAMETER'
2048
2049          ENDIF
2050
2051
2052    !-------------------------------------------------------------------------
2053
2054
2055
2056
2057          FORMAT (999(2x,EN14.4))
2058          FORMAT (999(2X,I3))
2059          FORMAT(EN12.3)
2060          END SUBROUTINE LOAD_VEC
2061
2062    !-------------------------------------------------------------------------
2063    !-------------------------------------------------------------------------
2064    !-------------------------------------------------------------------------
2065    !-------------------------------------------------------------------------
2066    !-------------------------------------------------------------------------
2067    !                                                                       !
2068    !
2069    !                       MASS MATRIX SUBROUTINE                          !
```

```
!
!                                                                      !
!----------------------------------------------------------------------
!----------------------------------------------------------------------
!----------------------------------------------------------------------
!----------------------------------------------------------------------
!----------------------------------------------------------------------
!----------------------------------------------------------------------
      SUBROUTINE M_MAT(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,RHO,LE,BE,ME
     + ,MM,N_FEM,E_NODE,N_L,N_B,N_NODE,CON)
      USE PROG_DEBUG
      IMPLICIT NONE
!----------------------------------------------------------------------
!     REFER TO THE MODULE FEM_MATRICES, BEAM_CONST, FEM_CONST FOR THE
!     DEFINITION OF THE VARIABLES IF THE DEFINITION IS NOT DECLARED HERE
      INTEGER :: I,J,K,L,Q     !LOOP COUNTERS
      INTEGER :: E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,N_FEM,E_NODE,N_L,N_B
      INTEGER :: N_NODE
      REAL(KIND=8) :: H,RHO,LE,BE
!-----LE IS THE LENGTH OF EACH ELEMENT
      REAL(KIND=8) :: ME(E_DOF,E_DOF)
      REAL(KIND=8) :: MM(TOT_DOF,TOT_DOF)
      REAL(KIND=8),ALLOCATABLE :: X_P(:),X_W(:)
!     X_P ARE THE GAUSS POINTS IN X DIRECTION
!     X_W ARE THE WEIGHTS
      REAL(KIND=8),ALLOCATABLE :: Y_P(:),Y_W(:)
!     Y_P ARE THE GAUSS POINTS IN Y DIRECTION
!     Y_W ARE THE WEIGHTS
      REAL(KIND=8) :: N1,N2,N3,N4,DET_J
!-----N1, N2, VALUES OF THE SHAPE FUNCTIONS AT THE GAUSS POINTS
      INTEGER :: CON(N_FEM,E_DOF)
!-----CONNECTIVITY MATRIX----------------------------------------------
      REAL(KIND=8) :: N(4,1)
      REAL(KIND=8) :: NN(4,4)
      REAL(KIND=8),ALLOCATABLE :: FUNCT(:,:,:)
      REAL(KIND=8) :: INTEGRAL(16)
      REAL(KIND=8), ALLOCATABLE :: M_FUN(:,:)
      REAL(KIND=8),ALLOCATABLE :: W_X(:,:), W_Y(:,:)
      REAL(KIND=8) :: V(1,1)
      REAL(KIND=8) :: A(4,4)
!     THESE ARE TEMPORARY MATRICES CREATED TO CALCULATE THE MASS MATRIX

!----------------------------------------------------------------------
!     ALLOCATION OF SIZES TO THE MATRICES
!
      ALLOCATE (X_P(G_X),X_W(G_X))
      ALLOCATE (Y_P(G_Y),Y_W(G_Y))
      ALLOCATE (FUNCT(G_X,G_Y,16))
      ALLOCATE (M_FUN(G_X,G_Y))
      ALLOCATE(W_X(1,G_X),W_Y(G_Y,1))
!----------------------------------------------------------------------
!     INITIALLIZING THE MATRICES
      ME=0.0D0; MM=0.0D0
!----------------------------------------------------------------------
!
!     CALLING FOR GAUSS INTEGRATION POINTS AND WEIGHTS
      CALL GAUSS_LEG(G_X,X_P,X_W)
      CALL GAUSS_LEG(G_Y,Y_P,Y_W)
!----------------------------------------------------------------------
!     PRINTING THE GAUSS POINTS ABSCISSA AND WEIGHTS
!----------------------------------------------------------------------
      IF (DEBUG) THEN
      !!!write(51,*) 'GAUSS ABSCISSA AND WEIGHTS'
      !!!write(51,*) 'X GAUSS POINTS AND WEIGHTS'
      DO I=1,G_X
         !!!write(51,20) X_P(I), X_W(I)
      ENDDO
      !!write(51,*) 'Y GAUSS POINTS AND WEIGHTS'
      DO I=1,G_Y
```

```
2139            !!write(51,20) Y_P(I), Y_W(I)
2140          ENDDO
2141          ENDIF
2142    !-----------------------------------------------------------------------
2143    !     CALCULATION OF THE MASS MATRIX (ELEMENTAL)
2144          DO I = 1,G_X
2145             DO J=1,G_Y
2146                CALL FEM_SHAPE(X_P(I),Y_P(J),I_ORDER
2147         +       ,N1,N2,N3,N4,DET_J,LE,BE)
2148                N(1,1)=N1; N(2,1)=N2; N(3,1)=N3; N(4,1)=N4
2149                NN=MATMUL(N,TRANSPOSE(N))
2150                DO K=1,4
2151                   DO L=1,4
2152                      FUNCT(I,J,K+L+3*K-4)=NN(K,L)
2153                   ENDDO
2154                ENDDO
2155             ENDDO
2156          ENDDO
2157          IF(DEBUG) THEN
2158             !!!write(51,*) 'FUNCT MATRIX OF n1^2'
2159             DO I=1,G_X
2160                !!!write(51,20) (FUNCT(I,J,1), J=1,G_Y)
2161             ENDDO
2162          ENDIF
2163    !-----------------------------------------------------------------------
2164    !-----------------------------------------------------------------------
2165          DO I=1,G_X
2166             W_X(1,I)=X_W(I)
2167          ENDDO
2168          DO I=1,G_Y
2169             W_Y(I,1)=X_W(I)
2170          ENDDO
2171    !-----------------------------------------------------------------------
2172    !-----------------------------------------------------------------------
2173          IF(DEBUG) THEN
2174             !!!write(51,*) 'W_X'
2175             !!!write(51,20) W_X
2176             !!!write(51,*) 'W_Y'
2177             !!!write(51,20) W_Y
2178          ENDIF
2179    !-----------------------------------------------------------------------
2180    !-----------------------------------------------------------------------
2181          DO I=1,16
2182             M_FUN=0.0D0
2183             DO J=1,G_X
2184                DO K=1,G_Y
2185                   M_FUN(J,K)=FUNCT(J,K,I)
2186                ENDDO
2187             ENDDO
2188             !!!write(51,*) 'M_FUN'
2189             !!!write(51,*) I
2190             DO L=1,G_X
2191                !!!write(51,20) (M_FUN(L,Q), Q=1,G_Y)
2192             ENDDO
2193             V=MATMUL(MATMUL(W_X,M_FUN),W_Y)
2194             INTEGRAL(I)=V(1,1)
2195          ENDDO
2196    !-----------------------------------------------------------------------
2197    !-----------------------------------------------------------------------
2198    !       IF(DEBUG) THEN
2199             !!!write(51,*) 'INTEGRALS FOR MASS'
2200             DO I=1,16
2201                !!!write(51,20) INTEGRAL(I)
2202             ENDDO
2203    !       ENDIF
2204    !-----------------------------------------------------------------------
2205    !-----------------------------------------------------------------------
2206    !     DEFINING MASS MATRIX (ELEMENTAL)
2207    !     STARTING WITH THE MATRIX OF GLOBAL INTEGRALS
```

```
      DO I=1,4
         DO J=1,4
             A(I,J)=DET_J*INTEGRAL(I+J+3*I-4)
         ENDDO
      ENDDO
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
      IF(DEBUG) THEN
         !!!write(51,*) 'MATRIX OF GLOBAL INTEGRALS 4X4'
         DO I=1,4
             !!!write(51,20) (A(I,J),J=1,4)
         ENDDO
      ENDIF
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!      ASSEMBLING THE LOCAL MASS MATRIX
      DO I=1,4
         DO J=1,4
             ME(I,J)=RHO*H*A(I,J)
             ME(4+I,4+J)=RHO*H**3.0D0/12.0D0*A(I,J)
             ME(8+I,J)=RHO*H**3.0D0/12.0D0*A(I,J)
             ME(I,8+J)=RHO*H**3.0D0/12.0D0*A(I,J)
             ME(8+I,8+J)=RHO*H**5.0D0/80.0D0*A(I,J)
             ME(12+I,4+J)=RHO*H**5.0D0/80.0D0*A(I,J)
             ME(4+I,12+J)=RHO*H**5.0D0/80.0D0*A(I,J)
             ME(12+I,12+J)=RHO*H**7.0D0/448.0D0*A(I,J)
         ENDDO
      ENDDO
      DO I=1,16
         DO J=1,16
             ME(I+16,J+16)=ME(I,J)
             ME(I+32,J+32)=ME(I,J)
         ENDDO
      ENDDO

!-----------------------------------------------------------------------
!     PRINTING IN DUMP.DAT, UNCOMMENT IF OUTPUT NEEDED
      IF(DEBUG) THEN
      !!!write(51,*) 'CONSISTENT ELEMENT MASS MATRIX'
      DO I=1,E_DOF
         !!!write(51,20) (ME(I,J), J=1,E_DOF)
      ENDDO
      ENDIF

!-----------------------------------------------------------------------

!-----------------------------------------------------------------------
!     CONNECTIVITY MATRIX
!-----------------------------------------------------------------------
        DO K=1,N_B
          DO I=1,N_L
             DO J=1,E_DOF/4.0D0
             CON(K+I+(N_L-1)*K-N_L,4*J-3)=(J-1)*N_NODE+K+I+
     +       (N_L-1)*K-N_L+K-1;
             CON(K+I+(N_L-1)*K-N_L,4*J-2)=(J-1)*N_NODE+K+I+
     +       (N_L-1)*K-N_L+1+K-1;
             CON(K+I+(N_L-1)*K-N_L,4*J-1)=(J-1)*N_NODE+N_L+K+
     +       I+(N_L-1)*K-N_L+2+K-1;
             CON(K+I+(N_L-1)*K-N_L,4*J)=(J-1)*N_NODE+N_L+K+I+
     +       (N_L-1)*K-N_L+1+K-1;
             ENDDO
          ENDDO
        ENDDO
!
!      UNCOMMENT 150 - 153 TO OUTPUT CONNECTIVITY MATRIX
      IF(DEBUG) THEN
         OPEN(71,FILE='CONNECTIVITY.DAT')
         DO I=1,N_FEM
             WRITE(71,21) (CON(I,J), J=1,E_DOF)
```

```
2277            ENDDO
2278  !          CLOSE(71)
2279          ENDIF
2280  !
2281  !-----GLOBAL MASS MATRIX
2282          DO I=1,N_FEM    ! N_FEM IS THE TOTAL NUMBER OF ELEMENTS
2283             DO J=1,E_DOF  ! E_DOF IS THE NUMBER OF ELEMENTAL DOF
2284                 DO K=1,E_DOF
2285                     MM(CON(I,J),CON(I,K))=MM(CON(I,J),CON(I,K))+ME(J,K)
2286                 ENDDO
2287             ENDDO
2288          ENDDO
2289          DO I=1,TOT_DOF
2290             DO J=1,TOT_DOF
2291                 IF (I>J) THEN
2292                      MM(I,J)=0.0D0
2293                 ENDIF
2294             ENDDO
2295          ENDDO
2296  !-----------------------------------------------------------------------
2297  !               PRINTING THE GLOBAL MASS MATRIX
2298  !
2299  !       !!!write(51,*) 'GLOBAL MASS MATRIX'
2300  !       DO I=1,TOT_DOF
2301  !          !!!write(51,20) (MM(I,J), J=1,TOT_DOF)
2302  !       ENDDO
2303  !
2304  !
2305  !       CLOSE(51)
2306          DEALLOCATE(FUNCT,M_FUN,X_P,X_W,Y_P,Y_W,W_X,W_Y)
2307          !!!write(51,*) 'MASS MATRIX SUB COMPLETE'
2308  !
2309  !
2310          FORMAT (999(2x,EN14.4))
2311          FORMAT (999(2X,I5))
2312  !
2313          END SUBROUTINE M_MAT
2314  !-----------------------------------------------------------------------
2315  !-----------------------------------------------------------------------
2316  !-----------------------------------------------------------------------
2317  !-----------------------------------------------------------------------
2318  !-----------------------------------------------------------------------
2319  !
2320  !                         STIFFNESS MATRIX SUBROUTINE
2321  !
2322  !-----------------------------------------------------------------------
2323  !-----------------------------------------------------------------------
2324  !-----------------------------------------------------------------------
2325  !-----------------------------------------------------------------------
2326          SUBROUTINE K_MAT(N_FEM,N_NODE,E_DOF,TOT_DOF,N_L,N_B,E_NODE
2327         + ,H,LE,BE,CC,G_X,G_Y,KE,KK,I_ORDER)
2328          USE PROG_DEBUG
2329          IMPLICIT NONE
2330  !
2331          INTEGER :: I,J,K,L,Q    !LOOP COUNTERS
2332          INTEGER :: N_FEM,N_NODE,E_NODE,E_DOF,TOT_DOF,G_X,G_Y,N_L,N_B
2333         + ,I_ORDER
2334          !SAME DEFINITIONS AS MAIN PROGRAM AND MODULE
2335          REAL(KIND=8) :: H,LE,BE
2336          REAL(KIND=8) :: CC(6,6)
2337          REAL(KIND=8) :: DET_J   !JACOBIAN MATRIX DETERMINANT
2338          REAL(KIND=8) :: KE(E_DOF,E_DOF), KK(TOT_DOF,TOT_DOF)
2339          REAL(KIND=8),ALLOCATABLE :: KE1(:,:),KE2(:,:),KE3(:,:),KE4(:,:),
2340         + KE5(:,:),KE6(:,:),KE7(:,:),KE8(:,:),KE9(:,:),KE10(:,:),KE11(:,:)
2341         + ,KE12(:,:)
2342          INTEGER,ALLOCATABLE :: CON(:,:)
2343          REAL(KIND=8),ALLOCATABLE :: X_P(:),Y_P(:),X_W(:),Y_W(:),W_X(:,:)
2344         + ,W_Y(:,:),M_FUN1(:,:),M_FUN2(:,:),M_FUN3(:,:),M_FUN4(:,:)
2345         + ,M_FUN5(:,:),M_FUN6(:,:)
```

```
      REAL(KIND=8) :: FUNCT1(G_X,G_Y,16),FUNCT2(G_X,G_Y,16)
     + ,FUNCT3(G_X,G_Y,16),FUNCT4(G_X,G_Y,16),FUNCT5(G_X,G_Y,16)
     + ,FUNCT6(G_X,G_Y,16)
      REAL(KIND=8) :: N1,N2,N3,N4
      REAL(KIND=8) :: DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y
      REAL(KIND=8) :: N(4,1),DNX(4,1),DNY(4,1)
      REAL(KIND=8) :: NN(4,4),DNNX(4,4),DNNY(4,4),DNNXY(4,4),NDNX(4,4)
     + ,NDNY(4,4)
      REAL(KIND=8) :: INTEGRAL1(16),INTEGRAL2(16),INTEGRAL3(16),
     + INTEGRAL4(16),INTEGRAL5(16),INTEGRAL6(16)
      REAL(KIND=8) :: V1(1,1),V2(1,1),V3(1,1),V4(1,1),V5(1,1),V6(1,1)
      REAL(KIND=8) :: A1(4,4),A2(4,4),A3(4,4),A4(4,4),A5(4,4),A6(4,4)
      REAL(KIND=8) :: A4T(4,4),A5T(4,4),A6T(4,4)
      REAL(KIND=8) :: DIFF(E_DOF,E_DOF)
      !MATRICES USED FOR STIFF. MAT CALCULATION
!
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!     ALLOCATING MATRIX DIMENSIONS
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
      ALLOCATE(CON(N_FEM,E_DOF))
      ALLOCATE (X_P(G_X),X_W(G_X))
      ALLOCATE (Y_P(G_Y),Y_W(G_Y))
      ALLOCATE (M_FUN1(G_X,G_Y),M_FUN2(G_X,G_Y),M_FUN3(G_X,G_Y),
     + M_FUN4(G_X,G_Y),M_FUN5(G_X,G_Y),M_FUN6(G_X,G_Y))
      ALLOCATE(W_X(1,G_X),W_Y(G_Y,1))
      ALLOCATE(KE1(E_DOF,E_DOF),KE2(E_DOF,E_DOF),KE3(E_DOF,E_DOF),
     + KE4(E_DOF,E_DOF),KE5(E_DOF,E_DOF),KE6(E_DOF,E_DOF),
     + KE7(E_DOF,E_DOF),KE8(E_DOF,E_DOF),KE9(E_DOF,E_DOF),
     + KE10(E_DOF,E_DOF),KE11(E_DOF,E_DOF),KE12(E_DOF,E_DOF))
      KE=0.0D0; KK=0.0D0; KE1=0.0D0; KE2=0.0D0; KE3=0.0D0; KE4=0.0D0
      KE5=0.0D0; KE6=0.0D0; KE7=0.0D0; KE7=0.0D0; KE8=0.0D0; KE9=0.0D0
      KE10=0.0D0; KE11=0.0D0; KE12=0.0D0;KK=0.0D0
!-----------------------------------------------------------------------
!
!     CALLING FOR GAUSS INTEGRATION POINTS AND WEIGHTS
      CALL GAUSS_LEG(G_X,X_P,X_W)
      CALL GAUSS_LEG(G_Y,Y_P,Y_W)
!-----------------------------------------------------------------------
!      UNCOMMENT 62-67 TO OUTPUT THE GAUSS POINTS AND ABSCISSA IN DUMP.DAT
!-----------------------------------------------------------------------
      IF (DEBUG) THEN
      !!!write(51,*) 'GAUSS ABSCISSA AND WEIGHTS IN K SUBROUTINE'
      !!!write(51,*) 'X GAUSS POINTS AND WEIGHTS'
      DO I=1,G_X
         !!!write(51,20) X_P(I), X_W(I)
      ENDDO
      !!!write(51,*) 'Y GAUSS POINTS AND WEIGHTS'
      DO I=1,G_Y
         !!!write(51,20) Y_P(I), Y_W(I)
      ENDDO
      ENDIF

!-----------------------------------------------------------------------
      DO I = 1,G_X
         DO J=1,G_Y
             CALL FEM_SHAPE(X_P(I),Y_P(J),I_ORDER
     +       ,N1,N2,N3,N4,DET_J,LE,BE)
             CALL FEM_SHAPE_DIFF(X_P(I),Y_P(J),I_ORDER,DN1X,DN2X,DN3X
     +       ,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
             N(1,1)=N1; N(2,1)=N2; N(3,1)=N3; N(4,1)=N4
             DNX(1,1)=DN1X;DNX(2,1)=DN2X;DNX(3,1)=DN3X;DNX(4,1)=DN4X
             DNY(1,1)=DN1Y;DNY(2,1)=DN2Y;DNY(3,1)=DN3Y;DNY(4,1)=DN4Y

      OPEN(52,FILE='K_AND_A.DAT')
!             n(1,1)=1; n(2,1)=2; n(3,1)=3; n(4,1)=4
!             dnx(1,1)=5; dnx(2,1)=6; dnx(3,1)=7; dnx(4,1)=8

```

```
              NN=MATMUL(N,TRANSPOSE(N))
              DNNX=MATMUL(DNX,TRANSPOSE(DNX))
              DNNY=MATMUL(DNY,TRANSPOSE(DNY))
              DNNXY=MATMUL(DNX,TRANSPOSE(DNY))
              NDNX=MATMUL(N,TRANSPOSE(DNX))
              NDNY=MATMUL(N,TRANSPOSE(DNY))

!              write(52,*) 'multiplication veri'
!             do k=1,4
!                 write(52,20) (ndnx(k,l), l=1,4)
!              enddo

              DO K=1,4
                  DO L=1,4
                      FUNCT1(I,J,K+L+3*K-4)=NN(K,L)
                      FUNCT2(I,J,K+L+3*K-4)=DNNX(K,L)
                      FUNCT3(I,J,K+L+3*K-4)=DNNY(K,L)
                      FUNCT4(I,J,K+L+3*K-4)=DNNXY(K,L)
                      FUNCT5(I,J,K+L+3*K-4)=NDNX(K,L)
                      FUNCT6(I,J,K+L+3*K-4)=NDNY(K,L)
                  ENDDO
              ENDDO
           ENDDO
        ENDDO
!
        DO I=1,G_X
           W_X(1,I)=X_W(I)
        ENDDO
        DO I=1,G_Y
           W_Y(I,1)=Y_W(I)
        ENDDO
c       !!!write(51,*) 'FUNCTS CALCULATED IN K SUB'
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
        IF(DEBUG) THEN
           !!!write(51,*) 'W_X'
           !!!write(51,20) W_X
           !!!write(51,*) 'W_Y'
           !!!write(51,20) W_Y
        ENDIF
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
        DO I=1,16
           M_FUN1=0.0D0;M_FUN2=0.0D0;M_FUN3=0.0D0;M_FUN4=0.0D0;
           M_FUN5=0.0D0; M_FUN6=0.0D0;
           V1=0.0D0;V2=0.0D0;V3=0.0D0;V4=0.0D0;V5=0.0D0;V6=0.0D0
           DO J=1,G_X
               DO K=1,G_Y
                   M_FUN1(J,K)=FUNCT1(J,K,I)
                   M_FUN2(J,K)=FUNCT2(J,K,I)
                   M_FUN3(J,K)=FUNCT3(J,K,I)
                   M_FUN4(J,K)=FUNCT4(J,K,I)
                   M_FUN5(J,K)=FUNCT5(J,K,I)
                   M_FUN6(J,K)=FUNCT6(J,K,I)
               ENDDO
           ENDDO
c           !!!write(51,*) 'M_FUN'
c           !!!write(51,*) I
           DO L=1,G_X
c               !!!write(51,20) (M_FUN1(L,Q), Q=1,G_Y)
           ENDDO
           V1=MATMUL(MATMUL(W_X,M_FUN1),W_Y)
           V2=MATMUL(MATMUL(W_X,M_FUN2),W_Y)
           V3=MATMUL(MATMUL(W_X,M_FUN3),W_Y)
           V4=MATMUL(MATMUL(W_X,M_FUN4),W_Y)
           V5=MATMUL(MATMUL(W_X,M_FUN5),W_Y)
           V6=MATMUL(MATMUL(W_X,M_FUN6),W_Y)
           INTEGRAL1(I)=V1(1,1)
           INTEGRAL2(I)=V2(1,1)
```

```
2484            INTEGRAL3(I)=V3(1,1)
2485            INTEGRAL4(I)=V4(1,1)
2486            INTEGRAL5(I)=V5(1,1)
2487            INTEGRAL6(I)=V6(1,1)
2488         ENDDO
2489   !-----------------------------------------------------------------------
2490   !     INTEGRAL WRT DXDY
2491         DO I=1,4
2492            DO J=1,4
2493                A1(I,J)=DET_J*INTEGRAL1(I+J+3*I-4)
2494                A2(I,J)=DET_J*INTEGRAL2(I+J+3*I-4)
2495                A3(I,J)=DET_J*INTEGRAL3(I+J+3*I-4)
2496                A4(I,J)=DET_J*INTEGRAL4(I+J+3*I-4)
2497                A5(I,J)=DET_J*INTEGRAL5(I+J+3*I-4)
2498                A6(I,J)=DET_J*INTEGRAL6(I+J+3*I-4)
2499            ENDDO
2500         ENDDO
2501         A4T=TRANSPOSE(A4)
2502         A5T=TRANSPOSE(A5)
2503         A6T=TRANSPOSE(A6)
2504   !
2505         KE1=0.0D0;KE2=0.0D0;KE3=0.0D0;KE4=0.0D0;KE5=0.0D0;KE6=0.0D0
2506   !-----------------------------------------------------------------------
2507   !     ASSEMBLY OF LOCAL STIFFNESS MATRICES
2508         DO I=1,4
2509            DO J=1,4
2510                !EQUATION 1-4 NORMAL STRESSES DNNX
2511                KE1(I,J)=CC(1,1)*H*A2(I,J)
2512                KE1(I,8+J)=CC(1,1)*H**3.0D0/12.0D0*A2(I,J)
2513                KE1(4+I,4+J)=CC(1,1)*H**3.0D0/12.0D0*A2(I,J)
2514                KE1(4+I,12+J)=CC(1,1)*H**5.0D0/80.0D0*A2(I,J)
2515                KE1(8+I,J)=CC(1,1)*H**3.0D0/12.0D0*A2(I,J)
2516                KE1(8+I,8+J)=CC(1,1)*H**5.0D0/80.0D0*A2(I,J)
2517                KE1(12+I,4+J)=CC(1,1)*H**5.0D0/80.0D0*A2(I,J)
2518                KE1(12+I,12+J)=CC(1,1)*H**7.0D0/448.0D0*A2(I,J)
2519                !EQUATION 5-8 NORMAL STRESSES DNNY
2520                KE1(16+I,16+J)=CC(2,2)*H*A3(I,J)
2521                KE1(20+I,20+J)=CC(2,2)*H**3.0D0/12.0D0*A3(I,J)
2522                KE1(24+I,16+J)=CC(2,2)*H**3.0D0/12.0D0*A3(I,J)
2523                KE1(16+I,24+J)=CC(2,2)*H**3.0D0/12.0D0*A3(I,J)
2524                KE1(24+I,24+J)=CC(2,2)*H**5.0D0/80.0D0*A3(I,J)
2525                KE1(28+I,20+J)=CC(2,2)*H**5.0D0/80.0D0*A3(I,J)
2526                KE1(20+I,28+J)=CC(2,2)*H**5.0D0/80.0D0*A3(I,J)
2527                KE1(28+I,28+J)=CC(2,2)*H**7.0D0/448.0D0*A3(I,J)
2528            ENDDO
2529         ENDDO
2530         DO I=1,4
2531            DO J=1,4
2532                !EQUATION 1-4 NORMAL STRESSES DNNXY
2533                KE2(I,16+J)=CC(1,2)*H*A4(I,J)
2534                KE2(I,24+J)=CC(1,2)*H**3.0D0/12.0D0*A4(I,J)
2535                KE2(I+8,16+J)=CC(1,2)*H**3.0D0/12.0D0*A4(I,J)
2536                KE2(4+I,20+J)=CC(1,2)*H**3.0D0/12.0D0*A4(I,J)
2537                KE2(4+I,28+J)=CC(1,2)*H**5.0D0/80.0D0*A4(I,J)
2538                KE2(8+I,24+J)=CC(1,2)*H**5.0D0/80.0D0*A4(I,J)
2539                KE2(12+I,20+J)=CC(1,2)*H**5.0D0/80.0D0*A4(I,J)
2540                KE2(12+I,28+J)=CC(1,2)*H**7.0D0/448.0D0*A4(I,J)
2541                !EQUATION 5-8 NORMAL STRESSE DNYX
2542                KE2(16+I,J)=CC(1,2)*H*A4T(I,J)
2543                KE2(16+I,8+J)=CC(1,2)*H**3.0D0/12.0D0*A4T(I,J)
2544                KE2(24+I,J)=CC(1,2)*H**3.0D0/12.0D0*A4T(I,J)
2545                KE2(20+I,4+J)=CC(1,2)*H**3.0D0/12.0D0*A4T(I,J)
2546                KE2(20+I,12+J)=CC(1,2)*H**5.0D0/80.0D0*A4T(I,J)
2547                KE2(24+I,8+J)=CC(1,2)*H**5.0D0/80.0D0*A4T(I,J)
2548                KE2(28+I,4+J)=CC(1,2)*H**5.0D0/80.0D0*A4T(I,J)
2549                KE2(28+I,12+J)=CC(1,2)*H**7.0D0/448.0D0*A4T(I,J)
2550            ENDDO
2551         ENDDO
2552   !     COMPONENTS FROM SHEAR STRESSES
```

```
      DO I=1,4
         DO J=1,4
            !EQUATION 1-4 C(6,6) TERMS
!           U1 TERMS
            KE3(I,J)=CC(6,6)*H*A3(I,J)
            KE3(4+I,4+J)=CC(6,6)*H**3.0D0/12.0D0*A3(I,J)
            KE3(8+I,J)=CC(6,6)*H**3.0D0/12.0D0*A3(I,J)
            KE3(I,8+J)=CC(6,6)*H**3.0D0/12.0D0*A3(I,J)
            KE3(8+I,8+J)=CC(6,6)*H**5.0D0/80.0D0*A3(I,J)
            KE3(12+I,4+J)=CC(6,6)*H**5.0D0/80.0D0*A3(I,J)
            KE3(4+I,12+J)=CC(6,6)*H**5.0D0/80.0D0*A3(I,J)
            KE3(12+I,12+J)=CC(6,6)*H**7.0D0/448.0D0*A3(I,J)
!           U2 TERMS
            KE3(I,16+J)=CC(6,6)*H*A4T(I,J)
            KE3(I,24+J)=CC(6,6)*H**3.0D0/12.0D0*A4T(I,J)
            KE3(I+8,16+J)=CC(6,6)*H**3.0D0/12.0D0*A4T(I,J)
            KE3(4+I,20+J)=CC(6,6)*H**3.0D0/12.0D0*A4T(I,J)
            KE3(4+I,28+J)=CC(6,6)*H**5.0D0/80.0D0*A4T(I,J)
            KE3(8+I,24+J)=CC(6,6)*H**5.0D0/80.0D0*A4T(I,J)
            KE3(12+I,20+J)=CC(6,6)*H**5.0D0/80.0D0*A4T(I,J)
            KE3(12+I,28+J)=CC(6,6)*H**7.0D0/448.0D0*A4T(I,J)
            ! EQUATION 5-8 C(6,6) TERMS
            KE3(16+I,J)=CC(6,6)*H*A4(I,J)
            KE3(16+I,8+J)=CC(6,6)*H**3.0D0/12.0D0*A4(I,J)
            KE3(16+I+8,J)=CC(6,6)*H**3.0D0/12.0D0*A4(I,J)
            KE3(16+4+I,4+J)=CC(6,6)*H**3.0D0/12.0D0*A4(I,J)
            KE3(16+4+I,12+J)=CC(6,6)*H**5.0D0/80.0D0*A4(I,J)
            KE3(16+8+I,8+J)=CC(6,6)*H**5.0D0/80.0D0*A4(I,J)
            KE3(16+12+I,4+J)=CC(6,6)*H**5.0D0/80.0D0*A4(I,J)
            KE3(16+12+I,12+J)=CC(6,6)*H**7.0D0/448.0D0*A4(I,J)
            !
            KE3(16+I,16+J)=CC(6,6)*H*A2(I,J)
            KE3(20+I,20+J)=CC(6,6)*H**3.0D0/12.0D0*A2(I,J)
            KE3(24+I,16+J)=CC(6,6)*H**3.0D0/12.0D0*A2(I,J)
            KE3(16+I,24+J)=CC(6,6)*H**3.0D0/12.0D0*A2(I,J)
            KE3(24+I,24+J)=CC(6,6)*H**5.0D0/80.0D0*A2(I,J)
            KE3(28+I,20+J)=CC(6,6)*H**5.0D0/80.0D0*A2(I,J)
            KE3(20+I,28+J)=CC(6,6)*H**5.0D0/80.0D0*A2(I,J)
            KE3(28+I,28+J)=CC(6,6)*H**7.0D0/448.0D0*A2(I,J)
         ENDDO
      ENDDO
      !COMPONENTS IN THE FORM NiDNj
      DO I=1,4
         DO J=1,4
            KE4(I,36+J)=CC(1,3)*H*A5T(I,J)
            KE4(I,44+J)=CC(1,3)*H**3.0D0/4.0D0*A5T(I,J)
            KE4(I+4,40+J)=CC(1,3)*H**3.0D0/6.0D0*A5T(I,J)
            KE4(I+8,36+J)=CC(1,3)*H**3.0D0/12.0D0*A5T(I,J)
            KE4(I+8,44+J)=CC(1,3)*3.0D0*H**5.0D0/80.0D0*A5T(I,J)
            KE4(I+12,40+J)=CC(1,3)*H**5.0D0/40.0D0*A5T(I,J)
            !
            KE4(I+16,36+J)=CC(2,3)*H*A6T(I,J)
            KE4(I+16,44+J)=CC(2,3)*H**3.0D0/4.0D0*A6T(I,J)
            KE4(I+20,40+J)=CC(2,3)*H**3.0D0/6.0D0*A6T(I,J)
            KE4(I+24,36+J)=CC(2,3)*H**3.0D0/12.0D0*A6T(I,J)
            KE4(I+24,44+J)=CC(2,3)*3.0D0*H**5.0D0/80.0D0*A6T(I,J)
            KE4(I+28,40+J)=CC(2,3)*H**5.0D0/40.0D0*A6T(I,J)
         ENDDO
      ENDDO
      !COMPONENTS OF C55 IN EQNS 1-8
      DO I=1,4
         DO J=1,4
            !EQN 1-4
            KE5(4+I,4+J)=CC(5,5)*H*A1(I,J)
            KE5(4+I,12+J)=CC(5,5)*H**3.0D0/4.0D0*A1(I,J)
            KE5(4+I,32+J)=CC(5,5)*H*A5(I,J)
            KE5(4+I,40+J)=CC(5,5)*H**3.0D0/12.0D0*A5(I,J)
            KE5(8+I,8+J)=2.0D0*CC(5,5)*H**3.0D0/6.0D0*A1(I,J)
            KE5(8+I,36+J)=2.0D0*CC(5,5)*H**3.0D0/12.0D0*A5(I,J)
```

```
2622                KE5(8+I,44+J)=2.0D0*CC(5,5)*H**5.0D0/80.0D0*A5(I,J)
2623                KE5(12+I,4+J)=3.0D0*CC(5,5)*H**3.0D0/12.0D0*A1(I,J)
2624                KE5(12+I,12+J)=9.0D0*CC(5,5)*H**5.0D0/80.0D0*A1(I,J)
2625                KE5(12+I,32+J)=3.0D0*CC(5,5)*H**3.0D0/12.0D0*A5(I,J)
2626                KE5(12+I,40+J)=3.0D0*CC(5,5)*H**5.0D0/80.0D0*A5(I,J)
2627                !EQN 4-8
2628                KE5(20+I,20+J)=CC(4,4)*H*A1(I,J)
2629                KE5(20+I,28+J)=CC(4,4)*H**3.0D0/4.0D0*A1(I,J)
2630                KE5(20+I,32+J)=CC(4,4)*H*A6(I,J)
2631                KE5(20+I,40+J)=CC(4,4)*H**3.0D0/12.0D0*A6(I,J)
2632                KE5(24+I,24+J)=2.0D0*CC(4,4)*H**3.0D0/6.0D0*A1(I,J)
2633                KE5(24+I,36+J)=2.0D0*CC(4,4)*H**3.0D0/12.0D0*A6(I,J)
2634                KE5(24+I,44+J)=2.0D0*CC(4,4)*H**5.0D0/80.0D0*A6(I,J)
2635                KE5(28+I,20+J)=3.0D0*CC(4,4)*H**3.0D0/12.0D0*A1(I,J)
2636                KE5(28+I,28+J)=9.0D0*CC(4,4)*H**5.0D0/80.0D0*A1(I,J)
2637                KE5(28+I,32+J)=3.0D0*CC(4,4)*H**3.0D0/12.0D0*A6(I,J)
2638                KE5(28+I,40+J)=3.0D0*CC(4,4)*H**5.0D0/80.0D0*A6(I,J)
2639             ENDDO
2640          ENDDO
2641          ! ALL COMPONENTS OF EQNS 9 - 12
2642          DO I=1,4
2643             DO J=1,4
2644                ! EQ 9
2645                KE6(32+I,32+J)=CC(5,5)*H*A2(I,J)
2646                KE6(32+I,40+J)=CC(5,5)*H**3.0D0/12.0D0*A2(I,J)
2647                KE6(32+I,4+J)=CC(5,5)*H*A5T(I,J)
2648                KE6(32+I,12+J)=CC(5,5)*H**3.0D0/4.0D0*A5T(I,J)
2649                KE6(32+I,32+J)=KE6(32+I,32+J)+CC(4,4)*H*A3(I,J)
2650                KE6(32+I,40+J)=KE6(32+I,40+J)+
2651         +       CC(4,4)*H**3.0D0/12.0D0*A3(I,J)
2652                KE6(32+I,20+J)=CC(4,4)*H*A6T(I,J)
2653                KE6(32+I,28+J)=CC(4,4)*H**3.0D0/4.0D0*A6T(I,J)
2654                ! EQ 10
2655                KE6(36+I,36+J)=CC(5,5)*H**3.0D0/12.0D0*A2(I,J)
2656                KE6(36+I,44+J)=CC(5,5)*H**5.0D0/80.0D0*A2(I,J)
2657                KE6(36+I,8+J)=CC(5,5)*H**3.0D0/6.0D0*A5T(I,J)
2658                KE6(36+I,36+J)=KE6(36+I,36+J)
2659         +       +CC(4,4)*H**3.0D0/12.0D0*A3(I,J)
2660                KE6(36+I,44+J)=KE6(36+I,44+J)
2661         +       +CC(4,4)*H**5.0D0/80.0D0*A3(I,J)
2662                KE6(36+I,24+J)=CC(4,4)*H**3.0D0/6.0D0*A6T(I,J)
2663                KE6(36+I,J)=CC(1,3)*H*A5(I,J)
2664                KE6(36+I,8+J)=KE6(36+I,8+J)
2665         +       +CC(1,3)*H**3.0D0/12.0D0*A5(I,J)
2666                KE6(36+I,16+J)=CC(2,3)*H*A6(I,J)
2667                KE6(36+I,24+J)=KE6(36+I,24+J)
2668         +       +CC(2,3)*H**3.0D0/12.0D0*A6(I,J)
2669                KE6(36+I,36+J)=KE6(36+I,36+J)+CC(3,3)*H*A1(I,J)
2670                KE6(36+I,44+J)=KE6(36+I,44+J)
2671         +       +CC(3,3)*H**3.0D0/4.0D0*A1(I,J)
2672                ! EQ 11
2673                KE6(40+I,32+J)=CC(5,5)*H**3.0D0/12.0D0*A2(I,J)
2674                KE6(40+I,40+J)=CC(5,5)*H**5.0D0/80.0D0*A2(I,J)
2675                KE6(40+I,4+J)=CC(5,5)*H**3.0D0/12.0D0*A5T(I,J)
2676                KE6(40+I,12+J)=3.0D0*CC(5,5)*H**5.0D0/80.0D0*A5T(I,J)
2677                KE6(40+I,32+J)=KE6(40+I,32+J)
2678         +       +CC(4,4)*H**3.0D0/12.0D0*A3(I,J)
2679                KE6(40+I,40+J)=KE6(40+I,40+J)
2680         +       +CC(4,4)*H**5.0D0/80.0D0*A3(I,J)
2681                KE6(40+I,20+J)=CC(4,4)*H**3.0D0/12.0D0*A6T(I,J)
2682                KE6(40+I,28+J)=3.0D0*CC(4,4)*H**5.0D0/80.0D0*A6T(I,J)
2683                KE6(40+I,4+J)=KE6(40+I,4+J)
2684         +       +2.0D0*CC(1,3)*H**3.0D0/12.0D0*A5(I,J)
2685                KE6(40+I,12+J)=KE6(40+I,12+J)
2686         +       +2.0D0*CC(1,3)*H**5.0D0/80.0D0*A5(I,J)
2687                KE6(40+I,20+J)=KE6(40+I,20+J)
2688         +       +2.0D0*CC(2,3)*H**3.0D0/12.0D0*A6(I,J)
2689                KE6(40+I,28+J)=KE6(40+I,28+J)
2690         +       +2.0D0*CC(2,3)*H**5.0D0/80.0D0*A6(I,J)
```

```
                KE6(40+I,40+J)=KE6(40+I,40+J)
     +           +2.0D0*CC(3,3)*H**3.0D0/6.0D0*A1(I,J)
                ! EQ 12
                KE6(44+I,36+J)=CC(5,5)*H**5.0D0/80.0D0*A2(I,J)
                KE6(44+I,44+J)=CC(5,5)*H**7.0D0/448.0D0*A2(I,J)
                KE6(44+I,8+J)=CC(5,5)*H**5.0D0/40.0D0*A5T(I,J)
                KE6(44+I,36+J)=KE6(44+I,36+J)
     +           +CC(4,4)*H**5.0D0/80.0D0*A3(I,J)
                KE6(44+I,44+J)=KE6(44+I,44+J)
     +           +CC(4,4)*H**7.0D0/448.0D0*A3(I,J)
                KE6(44+I,24+J)=CC(4,4)*H**5.0D0/40.0D0*A6T(I,J)
                KE6(44+I,J)=3.0D0*CC(1,3)*H**3.0D0/12.0D0*A5(I,J)
                KE6(44+I,8+J)=KE6(44+I,8+J)
     +           +3.0D0*CC(1,3)*H**5.0D0/80.0D0*A5(I,J)
                KE6(44+I,16+J)=KE6(44+I,16+J)
     +           +3.0D0*CC(2,3)*H**3.0D0/12.0D0*A6(I,J)
                KE6(44+I,24+J)=KE6(44+I,24+J)
     +           +3.0D0*CC(2,3)*H**5.0D0/80.0D0*A6(I,J)
                KE6(44+I,36+J)=KE6(44+I,36+J)
     +           +3.0D0*CC(3,3)*H**3.0D0/12.0D0*A1(I,J)
                KE6(44+I,44+J)=KE6(44+I,44+J)
     +           +9.0D0*CC(3,3)*H**5.0D0/80.0D0*A1(I,J)
            ENDDO
          ENDDO
          DO I=1,4
            DO J=1,4
                !EQUATION 1-4 STRESSES WITH C(1,6) x GAMMA XY
                KE7(I,J)=CC(1,6)*H*A4(I,J)
                KE7(4+I,4+J)=CC(1,6)*H**3.0D0/12.0D0*A4(I,J)
                KE7(8+I,J)=CC(1,6)*H**3.0D0/12.0D0*A4(I,J)
                KE7(I,8+J)=CC(1,6)*H**3.0D0/12.0D0*A4(I,J)
                KE7(8+I,8+J)=CC(1,6)*H**5.0D0/80.0D0*A4(I,J)
                KE7(12+I,4+J)=CC(1,6)*H**5.0D0/80.0D0*A4(I,J)
                KE7(4+I,12+J)=CC(1,6)*H**5.0D0/80.0D0*A4(I,J)
                KE7(12+I,12+J)=CC(1,6)*H**7.0D0/448.0D0*A4(I,J)
!               U2 TERMS
                KE7(I,16+J)=CC(1,6)*H*A2(I,J)
                KE7(I,24+J)=CC(1,6)*H**3.0D0/12.0D0*A2(I,J)
                KE7(I+8,16+J)=CC(1,6)*H**3.0D0/12.0D0*A2(I,J)
                KE7(4+I,20+J)=CC(1,6)*H**3.0D0/12.0D0*A2(I,J)
                KE7(4+I,28+J)=CC(1,6)*H**5.0D0/80.0D0*A2(I,J)
                KE7(8+I,24+J)=CC(1,6)*H**5.0D0/80.0D0*A2(I,J)
                KE7(12+I,20+J)=CC(1,6)*H**5.0D0/80.0D0*A2(I,J)
                KE7(12+I,28+J)=CC(1,6)*H**7.0D0/448.0D0*A2(I,J)
                ! EQUATION 5-8 C(2,6) TERMS
                KE7(16+I,J)=CC(2,6)*H*A3(I,J)
                KE7(16+I,8+J)=CC(2,6)*H**3.0D0/12.0D0*A3(I,J)
                KE7(24+I,J)=CC(2,6)*H**3.0D0/12.0D0*A3(I,J)
                KE7(20+I,4+J)=CC(2,6)*H**3.0D0/12.0D0*A3(I,J)
                KE7(20+I,12+J)=CC(2,6)*H**5.0D0/80.0D0*A3(I,J)
                KE7(24+I,8+J)=CC(2,6)*H**5.0D0/80.0D0*A3(I,J)
                KE7(28+I,4+J)=CC(2,6)*H**5.0D0/80.0D0*A3(I,J)
                KE7(28+I,12+J)=CC(2,6)*H**7.0D0/448.0D0*A3(I,J)
C               !
                KE7(16+I,16+J)=CC(2,6)*H*A4T(I,J)
                KE7(20+I,20+J)=CC(2,6)*H**3.0D0/12.0D0*A4T(I,J)
                KE7(24+I,16+J)=CC(2,6)*H**3.0D0/12.0D0*A4T(I,J)
                KE7(16+I,24+J)=CC(2,6)*H**3.0D0/12.0D0*A4T(I,J)
                KE7(24+I,24+J)=CC(2,6)*H**5.0D0/80.0D0*A4T(I,J)
                KE7(28+I,20+J)=CC(2,6)*H**5.0D0/80.0D0*A4T(I,J)
                KE7(20+I,28+J)=CC(2,6)*H**5.0D0/80.0D0*A4T(I,J)
                KE7(28+I,28+J)=CC(2,6)*H**7.0D0/448.0D0*A4T(I,J)
            ENDDO
          ENDDO
          DO I=1,4
            DO J=1,4
                !EQUATION 1-4 SHEAR STRESSES DNNY COMP C62*E22
                KE8(I,16+J)=CC(6,2)*H*A3(I,J)
                KE8(4+I,20+J)=CC(6,2)*H**3.0D0/12.0D0*A3(I,J)
```

```
                KE8(8+I,16+J)=CC(6,2)*H**3.0D0/12.0D0*A3(I,J)
                KE8(I,24+J)=CC(6,2)*H**3.0D0/12.0D0*A3(I,J)
                KE8(8+I,24+J)=CC(6,2)*H**5.0D0/80.0D0*A3(I,J)
                KE8(12+I,20+J)=CC(6,2)*H**5.0D0/80.0D0*A3(I,J)
                KE8(4+I,28+J)=CC(6,2)*H**5.0D0/80.0D0*A3(I,J)
                KE8(12+I,28+J)=CC(6,2)*H**7.0D0/448.0D0*A3(I,J)
                !EQUATION 5-8 SHEAR STRESSES DNNX COMP C61*E11
                KE8(16+I,J)=CC(6,1)*H*A2(I,J)
                KE8(16+I,8+J)=CC(6,1)*H**3.0D0/12.0D0*A2(I,J)
                KE8(20+I,4+J)=CC(6,1)*H**3.0D0/12.0D0*A2(I,J)
                KE8(20+I,12+J)=CC(6,1)*H**5.0D0/80.0D0*A2(I,J)
                KE8(24+I,J)=CC(6,1)*H**3.0D0/12.0D0*A2(I,J)
                KE8(24+I,8+J)=CC(6,1)*H**5.0D0/80.0D0*A2(I,J)
                KE8(28+I,4+J)=CC(6,1)*H**5.0D0/80.0D0*A2(I,J)
                KE8(28+I,12+J)=CC(6,1)*H**7.0D0/448.0D0*A2(I,J)
             ENDDO
          ENDDO
          DO I=1,4
             DO J=1,4
                !EQUATION 1-4 SHEAR STRESSE DNYX COMP E11
                KE9(I,J)=CC(6,1)*H*A4T(I,J)
                KE9(I,8+J)=CC(6,1)*H**3.0D0/12.0D0*A4T(I,J)
                KE9(8+I,J)=CC(6,1)*H**3.0D0/12.0D0*A4T(I,J)
                KE9(4+I,4+J)=CC(6,1)*H**3.0D0/12.0D0*A4T(I,J)
                KE9(4+I,12+J)=CC(6,1)*H**5.0D0/80.0D0*A4T(I,J)
                KE9(8+I,8+J)=CC(6,1)*H**5.0D0/80.0D0*A4T(I,J)
                KE9(12+I,4+J)=CC(6,1)*H**5.0D0/80.0D0*A4T(I,J)
                KE9(12+I,12+J)=CC(6,1)*H**7.0D0/448.0D0*A4T(I,J)
                !EQUATION 5-8 SHEAR STRESSES DNNXY COMP E22
                KE9(16+I,16+J)=CC(6,2)*H*A4(I,J)
                KE9(16+I,24+J)=CC(6,2)*H**3.0D0/12.0D0*A4(I,J)
                KE9(24+I,16+J)=CC(6,2)*H**3.0D0/12.0D0*A4(I,J)
                KE9(20+I,20+J)=CC(6,2)*H**3.0D0/12.0D0*A4(I,J)
                KE9(20+I,28+J)=CC(6,2)*H**5.0D0/80.0D0*A4(I,J)
                KE9(24+I,24+J)=CC(6,2)*H**5.0D0/80.0D0*A4(I,J)
                KE9(28+I,20+J)=CC(6,2)*H**5.0D0/80.0D0*A4(I,J)
                KE9(28+I,28+J)=CC(6,2)*H**7.0D0/448.0D0*A4(I,J)
             ENDDO
          ENDDO
!
          DO I=1,4
             DO J=1,4
C               EQUATIONS 1-4 COMPONENTS C63*E33
                KE10(I,36+J)=CC(6,3)*H*A6T(I,J)
                KE10(I,44+J)=CC(6,3)*H**3.0D0/4.0D0*A6T(I,J)
                KE10(I+4,40+J)=CC(6,3)*H**3.0D0/6.0D0*A6T(I,J)
                KE10(I+8,36+J)=CC(6,3)*H**3.0D0/12.0D0*A6T(I,J)
                KE10(I+8,44+J)=CC(6,3)*3.0D0*H**5.0D0/80.0D0*A6T(I,J)
                KE10(I+12,40+J)=CC(6,3)*H**5.0D0/40.0D0*A6T(I,J)
C               EQUATIONS 5-8 COMPONENTS C63*E33
                KE10(16+I,36+J)=CC(6,3)*H*A5T(I,J)
                KE10(16+I,44+J)=CC(6,3)*H**3.0D0/4.0D0*A5T(I,J)
                KE10(I+20,40+J)=CC(6,3)*H**3.0D0/6.0D0*A5T(I,J)
                KE10(I+24,36+J)=CC(6,3)*H**3.0D0/12.0D0*A5T(I,J)
                KE10(I+24,44+J)=CC(6,3)*3.0D0*H**5.0D0/80.0D0*A5T(I,J)
                KE10(I+28,40+J)=CC(6,3)*H**5.0D0/40.0D0*A5T(I,J)
             ENDDO
          ENDDO
!         COMPONENENTS WITH COEFFICINETS C45 AND C54
          DO I=1,4
             DO J=1,4
c                !EQN 1-4
                KE11(4+I,20+J)=CC(4,5)*H*A1(I,J)
                KE11(4+I,28+J)=CC(4,5)*H**3.0D0/4.0D0*A1(I,J)
                KE11(4+I,32+J)=CC(4,5)*H*A6(I,J)
                KE11(4+I,40+J)=CC(4,5)*H**3.0D0/12.0D0*A6(I,J)
                KE11(8+I,24+J)=2.0D0*CC(4,5)*H**3.0D0/6.0D0*A1(I,J)
                KE11(8+I,36+J)=2.0D0*CC(4,5)*H**3.0D0/12.0D0*A6(I,J)
                KE11(8+I,44+J)=2.0D0*CC(4,5)*H**5.0D0/80.0D0*A6(I,J)
```

```
                KE11(12+I,20+J)=3.0D0*CC(4,5)*H**3.0D0/12.0D0*A1(I,J)
                KE11(12+I,28+J)=9.0D0*CC(4,5)*H**5.0D0/80.0D0*A1(I,J)
                KE11(12+I,32+J)=3.0D0*CC(4,5)*H**3.0D0/12.0D0*A6(I,J)
                KE11(12+I,40+J)=3.0D0*CC(4,5)*H**5.0D0/80.0D0*A6(I,J)
                !EQN 5-8
                KE11(20+I,4+J)=CC(5,4)*H*A1(I,J)
                KE11(20+I,12+J)=CC(5,4)*H**3.0D0/4.0D0*A1(I,J)
                KE11(20+I,32+J)=CC(5,4)*H*A5(I,J)
                KE11(20+I,40+J)=CC(5,4)*H**3.0D0/12.0D0*A5(I,J)
                KE11(24+I,8+J)=2.0D0*CC(5,4)*H**3.0D0/6.0D0*A1(I,J)
                KE11(24+I,36+J)=2.0D0*CC(5,4)*H**3.0D0/12.0D0*A5(I,J)
                KE11(24+I,44+J)=2.0D0*CC(5,4)*H**5.0D0/80.0D0*A5(I,J)
                KE11(28+I,4+J)=3.0D0*CC(5,4)*H**3.0D0/12.0D0*A1(I,J)
                KE11(28+I,12+J)=9.0D0*CC(5,4)*H**5.0D0/80.0D0*A1(I,J)
                KE11(28+I,32+J)=3.0D0*CC(5,4)*H**3.0D0/12.0D0*A5(I,J)
                KE11(28+I,40+J)=3.0D0*CC(5,4)*H**5.0D0/80.0D0*A5(I,J)
             ENDDO
          ENDDO
!         ADDITIONAL SHEAR TERMS IN EQUATIONS 9-12
          DO I=1,4
             DO J=1,4
                ! EQ 9
                KE12(32+I,32+J)=CC(5,4)*H*A4(I,J)
                KE12(32+I,40+J)=CC(5,4)*H**3.0D0/12.0D0*A4(I,J)
                KE12(32+I,20+J)=CC(5,4)*H*A5T(I,J)
                KE12(32+I,28+J)=CC(5,4)*H**3.0D0/4.0D0*A5T(I,J)
                KE12(32+I,32+J)=KE12(32+I,32+J)+CC(4,5)*H*A4T(I,J)
                KE12(32+I,40+J)=KE12(32+I,40+J)+
     +           CC(4,5)*H**3.0D0/12.0D0*A4T(I,J)
                KE12(32+I,4+J)=CC(4,5)*H*A6T(I,J)
                KE12(32+I,12+J)=CC(4,5)*H**3.0D0/4.0D0*A6T(I,J)
                ! EQ 10
                KE12(36+I,36+J)=CC(5,4)*H**3.0D0/12.0D0*A4(I,J)
                KE12(36+I,44+J)=CC(5,4)*H**5.0D0/80.0D0*A4(I,J)
                KE12(36+I,24+J)=CC(5,4)*H**3.0D0/6.0D0*A5T(I,J)
                KE12(36+I,36+J)=KE12(36+I,36+J)
     +           +CC(4,5)*H**3.0D0/12.0D0*A4T(I,J)
                KE12(36+I,44+J)=KE12(36+I,44+J)
     +           +CC(4,5)*H**5.0D0/80.0D0*A4T(I,J)
                KE12(36+I,8+J)=CC(4,5)*H**3.0D0/6.0D0*A6T(I,J)
!               M33 SHEAR

                KE12(36+I,J)=CC(3,6)*H*A6(I,J)
                KE12(36+I,8+J)=KE12(36+I,8+J)
     +           +CC(3,6)*H**3.0D0/12.0D0*A6(I,J)
                KE12(36+I,16+J)=CC(3,6)*H*A5(I,J)
                KE12(36+I,24+J)=KE12(36+I,24+J)
     +           +CC(3,6)*H**3.0D0/12.0D0*A5(I,J)
!
                ! EQ 11
                KE12(40+I,32+J)=CC(5,4)*H**3.0D0/12.0D0*A4(I,J)
                KE12(40+I,40+J)=CC(5,4)*H**5.0D0/80.0D0*A4(I,J)
                KE12(40+I,20+J)=CC(5,4)*H**3.0D0/12.0D0*A5T(I,J)
                KE12(40+I,28+J)=3.0D0*CC(5,4)*H**5.0D0/80.0D0*A5T(I,J)
                KE12(40+I,32+J)=KE12(40+I,32+J)
     +           +CC(4,5)*H**3.0D0/12.0D0*A4T(I,J)
                KE12(40+I,40+J)=KE12(40+I,40+J)
     +           +CC(4,5)*H**5.0D0/80.0D0*A4T(I,J)
                KE12(40+I,4+J)=CC(4,5)*H**3.0D0/12.0D0*A6T(I,J)
                KE12(40+I,12+J)=3.0D0*CC(4,5)*H**5.0D0/80.0D0*A6T(I,J)
!               M33 SHEAR TERMS
!
                KE12(40+I,4+J)=KE12(40+I,4+J)
     +           +2.0D0*CC(3,6)*H**3.0D0/12.0D0*A6(I,J)
                KE12(40+I,12+J)=KE12(40+I,12+J)
     +           +2.0D0*CC(3,6)*H**5.0D0/80.0D0*A6(I,J)
                KE12(40+I,20+J)=KE12(40+I,20+J)
     +           +2.0D0*CC(3,6)*H**3.0D0/12.0D0*A5(I,J)
                KE12(40+I,28+J)=KE12(40+I,28+J)
```

```
     +        +2.0D0*CC(3,6)*H**5.0D0/80.0D0*A5(I,J)
!
              ! EQ 12
              KE12(44+I,36+J)=CC(5,4)*H**5.0D0/80.0D0*A4(I,J)
              KE12(44+I,44+J)=CC(5,4)*H**7.0D0/448.0D0*A4(I,J)
              KE12(44+I,24+J)=CC(5,4)*H**5.0D0/40.0D0*A5T(I,J)
              KE12(44+I,36+J)=KE12(44+I,36+J)
     +        +CC(4,5)*H**5.0D0/80.0D0*A4T(I,J)
              KE12(44+I,44+J)=KE12(44+I,44+J)
     +        +CC(4,5)*H**7.0D0/448.0D0*A4T(I,J)
              KE12(44+I,8+J)=CC(4,5)*H**5.0D0/40.0D0*A6T(I,J)
!
!             M33 SHEAR TERMS
              KE12(44+I,J)=3.0D0*CC(3,6)*H**3.0D0/12.0D0*A6(I,J)
              KE12(44+I,8+J)=KE12(44+I,8+J)
     +        +3.0D0*CC(3,6)*H**5.0D0/80.0D0*A6(I,J)
              KE12(44+I,16+J)=KE12(44+I,16+J)
     +        +3.0D0*CC(3,6)*H**3.0D0/12.0D0*A5(I,J)
              KE12(44+I,24+J)=KE12(44+I,24+J)
     +        +3.0D0*CC(3,6)*H**5.0D0/80.0D0*A5(I,J)
          ENDDO
        ENDDO
!
        KE=KE1+KE2+KE3+KE4+KE5+KE6+KE7+KE8+KE9+KE10+KE11+KE12

        DEALLOCATE(KE1,KE2,KE3,KE4,KE5,KE6,KE7,KE8,KE9,KE10,KE11,KE12)
!FOR DEBUGGING
        DIFF=KE-TRANSPOSE(KE)
        IF(DEBUG) THEN
          !!!write(51,*) 'SYMMETRY TEST OF KE'
          DO I=1,E_DOF
            !!!write(51,20) (DIFF(I,J) ,J=1,E_DOF)
          ENDDO
        ENDIF
        IF(DEBUG) THEN
          WRITE(52,*) 'A1'
          DO I=1,4
            WRITE(52,20) (A1(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A2'
          DO I=1,4
            WRITE(52,20) (A2(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A3'
          DO I=1,4
            WRITE(52,20) (A3(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A4'
          DO I=1,4
            WRITE(52,20) (A4(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A4T'
          DO I=1,4
            WRITE(52,20) (A4T(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A5'
          DO I=1,4
            WRITE(52,20) (A5(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A5T'
          DO I=1,4
            WRITE(52,20) (A5T(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A6'
          DO I=1,4
            WRITE(52,20) (A6(I,J), J=1,4)
          ENDDO
          WRITE(52,*) 'A6T'
          DO I=1,4
```

```
2967                WRITE(52,20) (A6T(I,J), J=1,4)
2968             ENDDO
2969             !
2970   c          WRITE(52,*) 'KE1'
2971   c          DO I=1,E_DOF
2972   c              WRITE(52,20) (KE1(I,J), J=1,E_DOF)
2973   c          ENDDO
2974   c          WRITE(52,*) 'KE2'
2975   c          DO I=1,E_DOF
2976   c              WRITE(52,20) (KE2(I,J), J=1,E_DOF)
2977   c          ENDDO
2978   c          WRITE(52,*) 'KE3'
2979   c          DO I=1,E_DOF
2980   c              WRITE(52,20) (KE3(I,J), J=1,E_DOF)
2981   c          ENDDO
2982   c          WRITE(52,*) 'KE4'
2983   c          DO I=1,E_DOF
2984   c              WRITE(52,20) (KE4(I,J), J=1,E_DOF)
2985   c          ENDDO
2986   c          WRITE(52,*) 'KE5'
2987   c          DO I=1,E_DOF
2988   c              WRITE(52,20) (KE5(I,J), J=1,E_DOF)
2989   c          ENDDO
2990   c          WRITE(52,*) 'KE6'
2991   c          DO I=1,E_DOF
2992   c              WRITE(52,20) (KE6(I,J), J=1,E_DOF)
2993   c          ENDDO
2994         ENDIF
2995         CLOSE(52)
2996   !END OF DEBUGGING-----------------------------------------------------------
2997   !---------------------------------------------------------------------------
2998   !     CONNECTIVITY MATRIX
2999   !---------------------------------------------------------------------------
3000           DO K=1,N_B
3001             DO I=1,N_L
3002                DO J=1,E_DOF/4.0D0
3003                CON(K+I+(N_L-1)*K-N_L,4*J-3)=(J-1)*N_NODE+K+I+
3004        +        (N_L-1)*K-N_L+K-1;
3005                CON(K+I+(N_L-1)*K-N_L,4*J-2)=(J-1)*N_NODE+K+I+
3006        +        (N_L-1)*K-N_L+1+K-1;
3007                CON(K+I+(N_L-1)*K-N_L,4*J-1)=(J-1)*N_NODE+N_L+K+
3008        +        I+(N_L-1)*K-N_L+2+K-1;
3009                CON(K+I+(N_L-1)*K-N_L,4*J)=(J-1)*N_NODE+N_L+K+I+
3010        +        (N_L-1)*K-N_L+1+K-1;
3011                ENDDO
3012             ENDDO
3013           ENDDO
3014   !
3015   !     UNCOMMENT 150 - 153 TO OUTPUT CONNECTIVITY MATRIX
3016         IF(OUTPUT) THEN
3017            OPEN(71,FILE='CONNECTIVITY.DAT')
3018            DO I=1,N_FEM
3019                WRITE(71,21) (CON(I,J), J=1,E_DOF)
3020            ENDDO
3021            CLOSE(71)
3022         ENDIF
3023   !
3024   !-----GLOBAL STIFFNESS MATRIX
3025         DO I=1,N_FEM    ! N_FEM IS THE TOTAL NUMBER OF ELEMENTS
3026            DO J=1,E_DOF  ! E_DOF IS THE NUMBER OF ELEMENTAL DOF
3027                DO K=1,E_DOF
3028                    KK(CON(I,J),CON(I,K))=KK(CON(I,J),CON(I,K))+KE(J,K)
3029                ENDDO
3030            ENDDO
3031         ENDDO
3032         DO I=1,TOT_DOF
3033            DO J=1,TOT_DOF
3034                IF (I>J) THEN
3035                     KK(I,J)=0.0D0
```

```
3036                ENDIF
3037             ENDDO
3038          ENDDO
3039          DEALLOCATE(CON)
3040    !-----------------------------------------------------------------------
3041    !
3042          FORMAT (999(2x,EN14.4))
3043          FORMAT (999(2X,I5))
3044    !
3045          END SUBROUTINE K_MAT
3046
3047
3048          SUBROUTINE BC_APP_KM(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
3049         +  N_NODE,I_STATIC,I_BC,KM)
3050          USE PROG_DEBUG
3051          IMPLICIT NONE
3052    !     SUBROUTINE TO APPLY THE PRESCRIBED BOUNDARY CONDITIONS
3053          INTEGER :: N_L,N_B,NODE_L,NODE_B,TOT_DOF,I_STATIC,I_BC,N_NODE,
3054         +  I,J,K,L
3055          REAL(KIND=8) :: KM(TOT_DOF,TOT_DOF)
3056          !MATRIX EITHER KM OR MM
3057          INTEGER :: NODE_ID ! NODE NUMBER OF THE NODE AT (L/4,B/2)
3058
3059
3060          NODE_L=N_L+1
3061          NODE_B=N_B+1
3062    C      WRITE(*,*) 'I_STATIC = ',I_STATIC
3063          SELECT CASE(I_BC)
3064          CASE(1)
3065             WRITE(51,*) 'SSSS'
3066    !     EDGES X = 0 AND X = A, U_2=U_3=0
3067          DO I = 5,12
3068             DO J=1,NODE_B
3069                DO K=1,TOT_DOF
3070                   KM((I-1)*N_NODE+NODE_L*J,K)=0.0D0
3071                   KM(K,(I-1)*N_NODE+NODE_L*J)=0.0D0
3072                   KM((I-1)*N_NODE+NODE_L*J,(I-1)*N_NODE+NODE_L*J)
3073         +          =1.0D0
3074                   KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
3075                   KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3076                   KM((I-1)*N_NODE+(J-1)*NODE_L+1,
3077         +          (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0
3078                ENDDO
3079             ENDDO
3080          ENDDO
3081    !     EDGES Y = 0 AND Y = B, U_1=0
3082          DO I=1,4
3083             DO J=1,NODE_L
3084                DO K=1,TOT_DOF
3085                   KM((I-1)*N_NODE+J,K)=0.0D0
3086                   KM(K,(I-1)*N_NODE+J)=0.0D0
3087                   KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
3088                   KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
3089                   KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3090                   KM((I-1)*N_NODE+NODE_L*N_B+J,
3091         +          (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
3092                ENDDO
3093             ENDDO
3094          ENDDO
3095    !     U_3=0
3096          DO I=9,12
3097             DO J=1,NODE_L
3098                DO K=1,TOT_DOF
3099                   KM((I-1)*N_NODE+J,K)=0.0D0
3100                   KM(K,(I-1)*N_NODE+J)=0.0D0
3101                   KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
3102                   KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
3103                   KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3104                   KM((I-1)*N_NODE+NODE_L*N_B+J,
```

```
     +            (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
              ENDDO
           ENDDO
        ENDDO

        CASE(2)
           WRITE(51,*) 'CCCC'
        DO I = 1,12
           DO J=1,NODE_B
              DO K=1,TOT_DOF
                 KM((I-1)*N_NODE+NODE_L*J,K)=0.0D0
                 KM(K,(I-1)*N_NODE+NODE_L*J)=0.0D0
                 KM((I-1)*N_NODE+NODE_L*J,(I-1)*N_NODE+NODE_L*J)
     +            =1.0D0
                 KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
                 KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
                 KM((I-1)*N_NODE+(J-1)*NODE_L+1,
     +            (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0

              ENDDO
           ENDDO
        ENDDO
        DO I=1,12
           DO J=1,NODE_L
              DO K=1,TOT_DOF
                 KM((I-1)*N_NODE+J,K)=0.0D0
                 KM(K,(I-1)*N_NODE+J)=0.0D0
                 KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
                 KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
                 KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
                 KM((I-1)*N_NODE+NODE_L*N_B+J,
     +            (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0

              ENDDO
           ENDDO
        ENDDO
!
        CASE(3)
        WRITE(51,*) 'CCFF'
        DO I = 1,12
           DO J=1,NODE_B
              DO K=1,TOT_DOF
                 KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
                 KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
                 KM((I-1)*N_NODE+(J-1)*NODE_L+1,
     +            (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0

              ENDDO
           ENDDO
        ENDDO
!


!
        CASE(4)
        WRITE(51,*) 'CCFS'
        DO I = 1,12
           DO J=1,NODE_B
              DO K=1,TOT_DOF
                 KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
                 KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
                 KM((I-1)*N_NODE+(J-1)*NODE_L+1,
     +            (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0

              ENDDO
           ENDDO
        ENDDO
        DO I=1,12
           DO J=1,NODE_L
              DO K=1,TOT_DOF
```

```
                    KM((I-1)*N_NODE+J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+J)=0.0D0
                    KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
!
!

                ENDDO
            ENDDO
        ENDDO
        DO I=9,12
            DO J=1,NODE_L
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
                    KM((I-1)*N_NODE+NODE_L*N_B+J,
     +               (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
!
                ENDDO
            ENDDO
        ENDDO

        CASE(5)
        WRITE(51,*) 'CCSS'
        DO I = 1,12
            DO J=1,NODE_B
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
                    KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
                    KM((I-1)*N_NODE+(J-1)*NODE_L+1,
     +               (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0
!

                ENDDO
            ENDDO
        ENDDO
        DO I=1,12
            DO J=1,NODE_L
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+J)=0.0D0
                    KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
!

                ENDDO
            ENDDO
        ENDDO
        DO I=9,12
            DO J=1,NODE_L
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
                    KM((I-1)*N_NODE+NODE_L*N_B+J,
     +               (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
!

                ENDDO
            ENDDO
        ENDDO
        DO I = 9,12
            DO J=1,NODE_B
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+NODE_L*J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+NODE_L*J)=0.0D0
                    KM((I-1)*N_NODE+NODE_L*J,(I-1)*N_NODE+NODE_L*J)
     +               =1.0D0
!

                ENDDO
            ENDDO
```

```
3243          ENDDO
3244          CASE(6)
3245             WRITE(51,*) 'BEAM'
3246
3247
3248    !!!!!!!  UNCOMMENT FOR CC CONDITION
3249             DO I = 1,12
3250             DO J=1,NODE_B
3251                 DO K=1,TOT_DOF
3252                     KM((I-1)*N_NODE+NODE_L*J,K)=0.0D0
3253                     KM(K,(I-1)*N_NODE+NODE_L*J)=0.0D0
3254                     KM((I-1)*N_NODE+NODE_L*J,(I-1)*N_NODE+NODE_L*J)
3255         +            =1.0D0
3256                     KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
3257                     KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3258                     KM((I-1)*N_NODE+(J-1)*NODE_L+1,
3259         +            (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0
3260
3261    !
3262
3263                 ENDDO
3264             ENDDO
3265          ENDDO
3266    C        DO I=1,12
3267    C          DO J=1,NODE_L
3268    C              DO K=1,TOT_DOF
3269    C                  KM((I-1)*N_NODE+J,K)=0.0D0
3270    C                  KM(K,(I-1)*N_NODE+J)=0.0D0
3271    C                  KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
3272    C                  KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
3273    C                  KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3274    C                  KM((I-1)*N_NODE+NODE_L*N_B+J,
3275    C     +             (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
3276    C              ENDDO
3277    C          ENDDO
3278    C        ENDDO
3279
3280
3281    !!!!     SS condition
3282
3283    C          DO I = 5,12
3284    C           DO J=1,NODE_B
3285    C               DO K=1,TOT_DOF
3286    C                   KM((I-1)*N_NODE+NODE_L*J,K)=0.0D0
3287    C                   KM(K,(I-1)*N_NODE+NODE_L*J)=0.0D0
3288    C                   KM((I-1)*N_NODE+NODE_L*J,(I-1)*N_NODE+NODE_L*J)
3289    C      +             =1.0D0
3290    C                   KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
3291    C                   KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3292    C                   KM((I-1)*N_NODE+(J-1)*NODE_L+1,
3293    C      +             (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0
3294
3295    C               ENDDO
3296    C           ENDDO
3297    C        ENDDO
3298    C
3299    !        CONSTRAINING U2 = 0 FOR ALL NODES
3300             DO I=1,4*N_NODE
3301                 DO J=1,TOT_DOF
3302                     KM(4*N_NODE+I,J)=0.0D0
3303                     KM(J,4*N_NODE+I)=0.0D0
3304                     KM(4*N_NODE+I,4*N_NODE+I)=1.0D0
3305
3306
3307                 ENDDO
3308             ENDDO
3309          CASE(7)
3310    !     INTERIOR POINT CLAMPED (L/4,B/2)
3311
```

```
        NODE_ID=NODE_L*N_B/2+N_L/4+1
        WRITE(*,*) 'CCCC WITH INT. POINT CLAMPED'
        WRITE(*,*) 'NODE_ID',node_id
        DO I = 1,12
           DO J=1,NODE_B
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+NODE_L*J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+NODE_L*J)=0.0D0
                    KM((I-1)*N_NODE+NODE_L*J,(I-1)*N_NODE+NODE_L*J)
     +               =1.0D0
                    KM((I-1)*N_NODE+(J-1)*NODE_L+1,K)=0.0D0
                    KM(K,(I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
                    KM((I-1)*N_NODE+(J-1)*NODE_L+1,
     +               (I-1)*N_NODE+(J-1)*NODE_L+1)=1.0D0
!
!
!
                    KM((I-1)*N_NODE+NODE_ID,K)=0.0D0
                    KM(K,(I-1)*N_NODE+NODE_ID)=0.0D0
                    KM((I-1)*N_NODE+NODE_ID,(I-1)*N_NODE+NODE_ID)=1.0D0

                ENDDO
           ENDDO
        ENDDO
        DO I=1,12
           DO J=1,NODE_L
                DO K=1,TOT_DOF
                    KM((I-1)*N_NODE+J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+J)=0.0D0
                    KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
                    KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
                    KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
                    KM((I-1)*N_NODE+NODE_L*N_B+J,
     +               (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
!
!
!
                ENDDO
           ENDDO
        ENDDO


!!

        END SELECT
        FORMAT (9999(x,EN14.4))
        FORMAT(EN12.3)
        FORMAT(F5.0)
        FORMAT (999(2X,I5))
        END SUBROUTINE BC_APP_KM


        SUBROUTINE BC_APP_FF(N_L,N_B,NODE_L,NODE_B,TOT_DOF,
     +  N_NODE,I_STATIC,I_BC,FF)
        USE PROG_DEBUG
        IMPLICIT NONE
!       SUBROUTINE TO APPLY THE PRESCRIBED BOUNDARY CONDITIONS
        INTEGER :: N_L,N_B,NODE_L,NODE_B,TOT_DOF,I_STATIC,I_BC,N_NODE,
     +  I,J,K,L
        REAL(KIND=8) ::  FF(TOT_DOF)
        !LOAD VECTOR
        INTEGER :: NODE_ID ! NODE NUMBER OF THE NODE AT (L/4,B/2)


        NODE_L=N_L+1
        NODE_B=N_B+1
C        WRITE(*,*) 'I_STATIC = ',I_STATIC
        SELECT CASE(I_BC)
        CASE(1)
```

```
3381            WRITE(51,*) 'SSSS'
3382          DO I = 5,12
3383             DO J=1,NODE_B
3384                DO K=1,TOT_DOF
3385
3386                   FF((I-1)*N_NODE+NODE_L*J)=0.0D0
3387                   FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3388                ENDDO
3389             ENDDO
3390          ENDDO
3391          DO I=1,4
3392             DO J=1,NODE_L
3393                DO K=1,TOT_DOF
3394                   FF((I-1)*N_NODE+J)=0.0D0
3395                   FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3396                ENDDO
3397             ENDDO
3398          ENDDO
3399          DO I=9,12
3400             DO J=1,NODE_L
3401                DO K=1,TOT_DOF
3402                   FF((I-1)*N_NODE+J)=0.0D0
3403                   FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3404                ENDDO
3405             ENDDO
3406          ENDDO
3407
3408          CASE(2)
3409             WRITE(51,*) 'CCCC'
3410          DO I = 1,12
3411             DO J=1,NODE_B
3412                DO K=1,TOT_DOF
3413                   FF((I-1)*N_NODE+NODE_L*J)=0.0D0
3414                   FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3415                ENDDO
3416             ENDDO
3417          ENDDO
3418          DO I=1,12
3419             DO J=1,NODE_L
3420                DO K=1,TOT_DOF
3421                   FF((I-1)*N_NODE+J)=0.0D0
3422                   FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3423                ENDDO
3424             ENDDO
3425          ENDDO
3426    !
3427          CASE(3)
3428          WRITE(51,*) 'CCFF'
3429          DO I = 1,12
3430             DO J=1,NODE_B
3431                DO K=1,TOT_DOF
3432                     FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3433                ENDDO
3434             ENDDO
3435          ENDDO
3436          DO I=1,12
3437             DO J=1,NODE_L
3438                DO K=1,TOT_DOF
3439                   FF((I-1)*N_NODE+J)=0.0D0
3440                ENDDO
3441             ENDDO
3442          ENDDO
3443    !
3444
3445    !
3446          CASE(4)
3447          WRITE(51,*) 'CCFS'
3448          DO I = 1,12
3449             DO J=1,NODE_B
```

```
3450                DO K=1,TOT_DOF
3451                     FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3452                ENDDO
3453             ENDDO
3454          ENDDO
3455          DO I=1,12
3456             DO J=1,NODE_L
3457                DO K=1,TOT_DOF
3458                     FF((I-1)*N_NODE+J)=0.0D0
3459                ENDDO
3460             ENDDO
3461          ENDDO
3462          DO I=9,12
3463             DO J=1,NODE_L
3464                DO K=1,TOT_DOF
3465     !
3466                     FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3467                ENDDO
3468             ENDDO
3469          ENDDO
3470
3471          CASE(5)
3472          WRITE(51,*) 'CCSS'
3473          DO I = 1,12
3474             DO J=1,NODE_B
3475                DO K=1,TOT_DOF
3476                     FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3477                ENDDO
3478             ENDDO
3479          ENDDO
3480          DO I=1,12
3481             DO J=1,NODE_L
3482                DO K=1,TOT_DOF
3483                     FF((I-1)*N_NODE+J)=0.0D0
3484                ENDDO
3485             ENDDO
3486          ENDDO
3487          DO I=9,12
3488             DO J=1,NODE_L
3489                DO K=1,TOT_DOF
3490                     FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
3491                ENDDO
3492             ENDDO
3493          ENDDO
3494          DO I = 9,12
3495             DO J=1,NODE_B
3496                DO K=1,TOT_DOF
3497                     FF((I-1)*N_NODE+NODE_L*J)=0.0D0
3498                ENDDO
3499             ENDDO
3500          ENDDO
3501          CASE(6)
3502             WRITE(51,*) 'BEAM'
3503
3504     !!!!!     CC
3505             DO I = 1,12
3506             DO J=1,NODE_B
3507                DO K=1,TOT_DOF
3508                     FF((I-1)*N_NODE+NODE_L*J)=0.0D0
3509                     FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0
3510                ENDDO
3511             ENDDO
3512          ENDDO
3513     C       DO I=1,12
3514     C          DO J=1,NODE_L
3515     C              DO K=1,TOT_DOF
3516     C                  KM((I-1)*N_NODE+J,K)=0.0D0
3517     C                  KM(K,(I-1)*N_NODE+J)=0.0D0
3518     C                  KM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
```

```
C                  KM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
C                  KM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
C                  KM((I-1)*N_NODE+NODE_L*N_B+J,
C     +             (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
C!
C                  MM((I-1)*N_NODE+J,K)=0.0D0
C                  MM(K,(I-1)*N_NODE+J)=0.0D0
C                  MM((I-1)*N_NODE+J,(I-1)*N_NODE+J)=1.0D0
C                  MM((I-1)*N_NODE+NODE_L*N_B+J,K)=0.0D0
C                  MM(K,(I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
C                  MM((I-1)*N_NODE+NODE_L*N_B+J,
C     +             (I-1)*N_NODE+NODE_L*N_B+J)=1.0D0
C!
C                  FF((I-1)*N_NODE+J)=0.0D0
C                  FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
C              ENDDO
C           ENDDO
C        ENDDO

!!!!!     making all dofs zero in u2

           DO I=1,4*N_NODE
               DO J=1,TOT_DOF
                   FF(N_NODE*4+I)=0.0D0
               ENDDO
           ENDDO
        CASE(7)
!       INTERIOR POINT CLAMPED (L/4,B/2)

        NODE_ID=NODE_L*N_B/2+N_L/4+1
        WRITE(*,*) 'CCCC WITH INT. POINT CLAMPED'
        WRITE(*,*) 'NODE_ID',node_id
        DO I = 1,12
           DO J=1,NODE_B
               DO K=1,TOT_DOF
!
                   FF((I-1)*N_NODE+NODE_L*J)=0.0D0
                   FF((I-1)*N_NODE+(J-1)*NODE_L+1)=0.0D0

                   FF((I-1)*N_NODE+NODE_ID)=0.0D0
               ENDDO
           ENDDO
        ENDDO
        DO I=1,12
           DO J=1,NODE_L
               DO K=1,TOT_DOF

                   FF((I-1)*N_NODE+J)=0.0D0
                   FF((I-1)*N_NODE+NODE_L*N_B+J)=0.0D0
               ENDDO
           ENDDO
        ENDDO


!!

        END SELECT
        FORMAT (9999(x,EN14.4))
        FORMAT(EN12.3)
        FORMAT(F5.0)
        FORMAT (999(2X,I5))
        END SUBROUTINE BC_APP_FF

        SUBROUTINE GAUSS_LEG (NQ,Z,W)

C---------------------------------------
C FDLIB
C
C COPYRIGHT BY C. POZRIKIDIS, 1999
```

```
C ALL RIGHTS RESERVED.
C
C THIS PROGRAM IS TO BE USED ONLY UNDER THE
C STIPULATIONS OF THE LICENSING AGREEMENT.
C--------------------------------------

C------------------------------------------------
C THIS PROGRAM ACCOMPANIES THE BOOK:
C
C             C. POZRIKIDIS
C ``NUMERICAL COMPUTATION IN SCIENCE AND ENGINEERING''
C        OXFORD UNIVERSITY PRESS
C------------------------------------------------

C------------------------------------------------
C ABSCISSAS AND WEIGHTS FOR THE GAUSS-LEGENDRE
C QUADRATURE WITH NQ POINTS
C
C THIS TABLE CONTAINS VALUES FOR
C
C  NQ = 1,2,3,4,5,6,7,8,12,20
C
C  DEFAULT VALUE IS 20
C------------------------------------------------

      IMPLICIT NONE

      REAL(KIND=8) :: Z(20),W(20)
      INTEGER :: NQ

C-----
C TRAP
C-----

      IF(    NQ.NE. 1
     +  .AND.NQ.NE. 2
     +  .AND.NQ.NE. 3
     +  .AND.NQ.NE. 4
     +  .AND.NQ.NE. 5
     +  .AND.NQ.NE. 6
     +  .AND.NQ.NE. 7
     +  .AND.NQ.NE. 8
     +  .AND.NQ.NE.12
     +  .AND.NQ.NE.15
     +  .AND.NQ.NE.20
     +  ) THEN

         WRITE (6,*)
         WRITE (6,*) ' GAUSS_LEGENDRE:'
         WRITE (6,*)
         WRITE (6,*) '   CHOSEN NUMBER OF GAUSSIAN POINTS'
         WRITE (6,*) '   IS NOT AVAILABLE; WILL TAKE NQ=20'

         NQ = 20

      END IF

C--------------------
      IF(NQ.EQ.1) THEN
C--------------------

      Z(1) = 0.0D0

      W(1) = 2.0D0

C-------------------------
      ELSE IF(NQ.EQ.2) THEN
C-------------------------

```

```
3657         Z(1) = -0.57735 02691 8962576450
3658         Z(2) = -Z(1)
3659
3660         W(1) = 1.0D0
3661         W(2) = 1.0D0
3662
3663   C-------------------------
3664         ELSE IF(NQ.EQ.3) THEN
3665   C-------------------------
3666
3667         Z(1) = -0.77459 66692 4148337703
3668         Z(2) =  0.0D0
3669         Z(3) = -Z(1)
3670
3671         W(1) = 0.55555 55555 55555555555
3672         W(2) = 0.88888 88888 88888888888
3673         W(3) = 0.55555 55555 55555555555
3674
3675   C-------------------------
3676         ELSE IF(NQ.EQ.4) THEN
3677   C-------------------------
3678
3679         Z(1) = -0.86113 63115 9405257522
3680         Z(2) = -0.33998 10435 8485626480
3681         Z(3) = -Z(2)
3682         Z(4) = -Z(1)
3683
3684         W(1) = 0.34785 48451 3745385737
3685         W(2) = 0.65214 51548 6254614262
3686         W(3) = W(2)
3687         W(4) = W(1)
3688
3689   C-------------------------
3690         ELSE IF(NQ.EQ.5) THEN
3691   C-------------------------
3692
3693         Z(1) = -0.90617 98459 3866399279
3694         Z(2) = -0.53846 93101 0568309103
3695         Z(3) =  0.0D0
3696         Z(4) = -Z(2)
3697         Z(5) = -Z(1)
3698
3699         W(1) = 0.23692 68850 5618908751
3700         W(2) = 0.47862 86704 9936646804
3701         W(3) = 0.56888 88888 8888888889
3702         W(4) = W(2)
3703         W(5) = W(1)
3704
3705   C-------------------------
3706         ELSE IF(NQ.EQ.6) THEN
3707   C-------------------------
3708
3709         Z(1) = -0.93246 95142 03152
3710         Z(2) = -0.66120 93864 66265
3711         Z(3) = -0.23861 91860 83197
3712
3713         Z(4) = -Z(3)
3714         Z(5) = -Z(2)
3715         Z(6) = -Z(1)
3716
3717         W(1) = 0.17132 44923 79170
3718         W(2) = 0.36076 15730 48139
3719         W(3) = 0.46791 39345 72691
3720
3721         W(4) = W(3)
3722         W(5) = W(2)
3723         W(6) = W(1)
3724
3725   C-------------------------
```

```
      ELSE IF(NQ.EQ.7) THEN
C-------------------------

      Z(1) = -0.9491079123427585
      Z(2) = -0.7415311855993945
      Z(3) = -0.4058451513773972
      Z(4) = 0.0D0

      Z(5) = -Z(3)
      Z(6) = -Z(2)
      Z(7) = -Z(1)

      W(1) = 0.1294849661688697
      W(2) = 0.2797053914892766
      W(3) = 0.3818300505051189
      W(4) = 0.4179591836734694

      W(5) = W(3)
      W(6) = W(2)
      W(7) = W(1)
C-------------------------
      ELSE IF(NQ.EQ.8) THEN
C-------------------------

      Z(1) = -0.96028 98564 9753623168
      Z(2) = -0.79666 64774 1362673959
      Z(3) = -0.52553 24099 1632898581
      Z(4) = -0.18343 46424 9564980493

      Z(5) = -Z(4)
      Z(6) = -Z(3)
      Z(7) = -Z(2)
      Z(8) = -Z(1)

      W(1) = 0.10122 85362 9037625915
      W(2) = 0.22238 10344 5337447054
      W(3) = 0.31370 66458 7788728733
      W(4) = 0.36268 37833 7836198296

      W(5) = W(4)
      W(6) = W(3)
      W(7) = W(2)
      W(8) = W(1)

C-------------------------
      ELSE IF(NQ.EQ.12) THEN
C-------------------------

      Z(1) = -0.98156 06342 46719
      Z(2) = -0.90411 72563 70475
      Z(3) = -0.76990 26741 94305
      Z(4) = -0.58731 79542 86617
      Z(5) = -0.36783 14989 98180
      Z(6) = -0.12523 34085 11469

      Z(7) = -Z(6)
      Z(8) = -Z(5)
      Z(9) = -Z(4)
      Z(10)= -Z(3)
      Z(11)= -Z(2)
      Z(12)= -Z(1)

      W(1) = 0.04717 53363 86511
      W(2) = 0.10693 93259 95318
      W(3) = 0.16007 83285 43346
      W(4) = 0.20316 74267 23066
      W(5) = 0.23349 25365 38355
      W(6) = 0.24914 70458 13403
```

```
3795          W(7) = W(6)
3796          W(8) = W(5)
3797          W(9) = W(4)
3798          W(10)= W(3)
3799          W(11)= W(2)
3800          W(12)= W(1)
3801
3802    C-------------------------
3803          ELSE IF(NQ.EQ.15) THEN
3804    C-------------------------
3805
3806          Z(1) = -0.9879925180204854
3807          Z(2) = -0.9372733924007060
3808          Z(3) = -0.8482065834104272
3809          Z(4) = -0.7244177313601701
3810          Z(5) = -0.5709721726085388
3811          Z(6) = -0.3941513470775634
3812          Z(7) = -0.2011940939974345
3813          Z(8) = 0.0D0
3814
3815          Z(9) = -Z(7)
3816          Z(10) = -Z(6)
3817          Z(11) = -Z(5)
3818          Z(12) = -Z(4)
3819          Z(13) = -Z(3)
3820          Z(14) = -Z(2)
3821          Z(15) = -Z(1)
3822
3823          W(1) = 0.0307532419961173
3824          W(2) = 0.0703660474881081
3825          W(3) = 0.1071592204671719
3826          W(4) = 0.1395706779261543
3827          W(5) = 0.1662692058169939
3828          W(6) = 0.1861610000155622
3829          W(7) = 0.1984314853271116
3830          W(8) = 0.2025782419255613
3831
3832          W(9) = W(7)
3833          W(10) = W(6)
3834          W(11) = W(5)
3835          W(12)= W(4)
3836          W(13)= W(3)
3837          W(14)= W(2)
3838          W(15)= W(1)
3839    C-------------------------
3840          ELSE IF (NQ.EQ.20) THEN
3841    C-------------------------
3842
3843          Z(1) = -0.99312 85991 85094 924786
3844          Z(2) = -0.96397 19272 77913 791268
3845          Z(3) = -0.91223 44282 51325 905868
3846          Z(4) = -0.83911 69718 22218 823395
3847          Z(5) = -0.74633 19064 60150 792614
3848          Z(6) = -0.63605 36807 26515 025453
3849          Z(7) = -0.51086 70019 50827 098004
3850          Z(8) = -0.37370 60887 15419 560673
3851          Z(9) = -0.22778 58511 41645 078080
3852          Z(10)= -0.07652 65211 33497 333755
3853
3854          Z(11) = -Z(10)
3855          Z(12) = -Z(9)
3856          Z(13) = -Z(8)
3857          Z(14) = -Z(7)
3858          Z(15) = -Z(6)
3859          Z(16) = -Z(5)
3860          Z(17) = -Z(4)
3861          Z(18) = -Z(3)
3862          Z(19) = -Z(2)
3863          Z(20) = -Z(1)
```

```
3864
3865          W(1) = 0.01761 40071 39152 118312
3866          W(2) = 0.04060 14298 00386 941331
3867          W(3) = 0.06267 20483 34109 063570
3868          W(4) = 0.08327 67415 76704 748725
3869          W(5) = 0.10193 01198 17240 435037
3870          W(6) = 0.11819 45319 61518 417312
3871          W(7) = 0.13168 86384 49176 626898
3872          W(8) = 0.14209 61093 18382 051329
3873          W(9) = 0.14917 29864 72603 746788
3874          W(10)= 0.15275 33871 30725 850698
3875
3876          W(11) = W(10)
3877          W(12) = W(9)
3878          W(13) = W(8)
3879          W(14) = W(7)
3880          W(15) = W(6)
3881          W(16) = W(5)
3882          W(17) = W(4)
3883          W(18) = W(3)
3884          W(19) = W(2)
3885          W(20) = W(1)
3886
3887    C-----------
3888          END IF
3889    C-----------
3890
3891          RETURN
3892          END
3893
3894          SUBROUTINE LOAD_DYNA(I,DT,T0,FF,TOT_DOF,THETA,FF2,TSTEPS,
3895         +           APP_LOAD)
3896          USE PROG_DEBUG
3897          IMPLICIT NONE
3898          INTEGER :: N,I,J,TOT_DOF,TSTEPS
3899          REAL(KIND=8) :: FF(TOT_DOF),THETA,T,FF2(TOT_DOF),T0,DT,P,APP_LOAD
3900          REAL(KIND=8) :: COEFF
3901          LOGICAL :: TRIAN= .TRUE.   !TRIANGULAR PULSE LOAD
3902          N=I-1
3903          T=T0+N*DT
3904    c      IF (N<TSTEPS/10) THEN
3905    C         COEFF=1/(TSTEPS/10*DT)*N*DT
3906    C         COEFF=T
3907    C         FF2=FF*COEFF
3908    C         P=COEFF*APP_LOAD
3909    c      ELSE
3910    c         COEFF=EXP(-(T-TSTEPS/10*DT)/THETA)
3911    c         FF2=FF*COEFF
3912    c         P=COEFF*APP_LOAD
3913    c      ENDIF
3914          IF (TRIAN) THEN
3915             IF (T<=200) THEN
3916                 COEFF=T/200.0d0
3917                 FF2=FF*COEFF
3918                 P=APP_LOAD*COEFF
3919             ELSEIF (200<T .AND. T<=400) THEN
3920                 COEFF=-1/200.0D0*(T-200.0D0)+1.0D0
3921                 FF2=FF*COEFF
3922                 P=APP_LOAD*COEFF
3923             ELSE
3924                 FF2=0.0D0
3925                 P=0.0D0
3926             ENDIF
3927          ELSE
3928             IF (T<=1) THEN
3929                COEFF=T/1.0D0
3930    C            COEFF=1.0D0
3931                FF2=FF*COEFF
3932                P=COEFF*APP_LOAD
```

```
3933            ELSE
3934  C            COEFF=EXP(-(T-100.0D0)/THETA)
3935               COEFF=0.0D0
3936               FF2=FF*COEFF
3937               P=COEFF*APP_LOAD
3938             ENDIF
3939  C             IF (T<=100) THEN
3940  C                 COEFF=T/100.0d0
3941  C                 FF2=FF*COEFF
3942  C                 P=APP_LOAD*COEFF
3943  C           ELSEIF (20<T .AND. T<=40) THEN
3944  C               COEFF=-1/20.0D0*(T-20.0D0)+1.0D0
3945  C               FF2=FF*COEFF
3946  C               P=APP_LOAD*COEFF
3947  C            ELSE
3948  C               COEFF=EXP(-(T-100.0D0)/THETA)
3949  C             COEFF=0.0D0
3950  C               FF2=FF*COEFF
3951  C               P=COEFF*APP_LOAD
3952  C             ENDIF
3953          ENDIF
3954          OPEN(962,FILE='LOAD_PROF.DAT')
3955             WRITE(962,20) T,P
3956          FORMAT (9999(x,EN14.4))
3957          FORMAT(EN12.3)
3958          END SUBROUTINE LOAD_DYNA
3959
3960          SUBROUTINE STRESS_CAL(TOT_DOF,H,N_FEM,N_NODE,
3961         + NODE_L,NODE_B,IEL,E_DOF,G_X,G_Y,G_Z,T_P,LE,BE,ELE_STRAIN,
3962         + SOL,CON,CC,XE,YE,E_NODE,I_ORDER,TIME,NODAL_ACC,RHO,SRS_ID,
3963         + IELX,IELY,GLOB_XYZ,AN)
3964          USE PROG_DEBUG   !FOR SWITHING DEBUG MODE ON/OFF
3965          IMPLICIT NONE
3966          INTEGER :: I_BC,TOT_DOF,N_NODE,I,J,K,L,NODE_L,NODE_B
3967          INTEGER :: IEL, E_DOF !ELE. NUM. AND ELE. DOF FROM THE MAIN PROG
3968          INTEGER :: G_Z,N_FEM,T_P,E_NODE,IELX,IELY
3969          INTEGER :: CON(N_FEM,E_DOF)
3970          REAL(KIND=8) :: SOL(TOT_DOF),NODAL_ACC(E_DOF,1),AN(TOT_DOF)
3971  !       G_Z - DEFINED NUMBER OF GAUSS POINTS IN Z FOR STRESS CAL.
3972  !       T_P IS THE TOTAL NUMBER OF POINTS PER ELE. TO CAL. STRESS AND STRAIN
3973          INTEGER :: G_X,G_Y   !NUMBER OF GAUSS POINTS IN X AND Y RESP
3974          INTEGER :: G_XY      !NUMBER OF GAUSS POINTS IN THE XY PLANE
3975  !       STRAIN INCLUDING THE NODES
3976          REAL(KIND=8) :: GLOB_XYZ(N_NODE,2), H
3977  !       GAUSS POINTS AND WEIGHTS IN Z
3978          REAL(KIND=8),ALLOCATABLE :: Z_P(:), Z_W(:), X_P(:), X_W(:),
3979         +                            Y_P(:),Y_W(:)
3980  !       GLOBAL COORDINATES OF GAUSS POINTS IN Z,X AND Y
3981          REAL(KIND=8) :: X_CENT,Y_CENT
3982  !       X, Y COORDINATES OF CENTROID OF EACH ELEMENT
3983  !
3984          REAL(KIND=8) :: ELE_STRAIN(T_P,9)
3985          REAL(KIND=8) :: ELE_STRESS(T_P,9)
3986          REAL(KIND=8),ALLOCATABLE :: STRESS_OUT(:,:),STRAIN_OUT(:,:)
3987  !       GLOBAL COORDINATES OF THE GAUSS POINTS
3988          REAL(KIND=8), ALLOCATABLE :: GLOB_Z_P(:)
3989          REAL(KIND=8), ALLOCATABLE :: GLOB_X_P(:)
3990          REAL(KIND=8), ALLOCATABLE :: GLOB_Y_P(:)
3991          REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L
3992  !       COORDINATES OF THE 1ST, 2ND, 3RD AND 4TH NODE RESP. OF AN ELEMENT
3993          REAL(KIND=8) :: LE,BE
3994          REAL(KIND=8) :: NODAL_DISP(48,1),ELE_ACC(48,1)
3995          REAL(KIND=8) :: CC(6,6),RHO !MATERIAL STIFFNESS MATRIX AND DENSITY
3996          REAL(KIND=8) :: XE(N_FEM,E_NODE),YE(N_FEM,E_NODE)
3997          REAL(KIND=8),ALLOCATABLE :: COORDL(:,:),COORDG(:,:)
3998  !       COORD. IN XY PLANE (LOCAL AND GLOBAL)
3999          REAL(KIND=8),ALLOCATABLE :: LOCAL_C(:,:)
4000          INTEGER :: I_ORDER,SRS_ID
4001          REAL(KIND=8) :: DIVS
```

```
      REAL(KIND=8) :: TIME
      INTEGER :: N_L, N_B
      N_L=NODE_L-1
      N_B=NODE_B-1
      DO J=1,E_DOF
           NODAL_DISP(J,1)=SOL(CON(IEL,J))
      ENDDO
!-----------------------------------------------------------------------
!     ALLOCATING DIMENSION OF MATRICES
      ALLOCATE (Z_P(G_Z),Z_W(G_Z))
      ALLOCATE (X_P(G_X),X_W(G_X))
      ALLOCATE (Y_P(G_Y),Y_W(G_Y))
      ALLOCATE (GLOB_Z_P(G_Z))
      ALLOCATE (GLOB_X_P(G_X),GLOB_Y_P(G_Y))
      ALLOCATE (LOCAL_C(T_P,3))
!-----------------------------------------------------------------------
!     INITIALIZING THE MATRICES
      Z_P=0.0D0; Z_W=0.0D0; GLOB_Z_P=0.0D0;
!-----------------------------------------------------------------------
!     CALLING FOR GAUSS POINTS AND WEIGHTS IN X AND Z
      CALL GAUSS_LEG(G_X,X_P,X_W)
      CALL GAUSS_LEG(G_Z,Z_P,Z_W)
      CALL GAUSS_LEG(G_Y,Y_P,Y_W)
      IF (DEBUG) THEN
      WRITE(51,*) 'GAUSS ABSCISSA AND WEIGHTS'
      WRITE(51,*) 'GAUSS POINTS IN SUB_STRESS, LOOP', IEL
         DO I=1,G_Z
             WRITE(51,20) Z_P(I), Z_W(I)
         ENDDO
      ENDIF

!-----------------------------------------------------------------------
!     GLOBAL X3 COORDINATES OF GAUSS POINTS COORDINATES

      DIVS=G_Z-1
      DO I =1,G_Z
         GLOB_Z_P(I)= H/DIVS*(I-(G_Z+1.0D0)/2.0D0)
      ENDDO
      IF (DEBUG) THEN
      WRITE(51,*) 'GLOBAL COORDINATES OF GAUSS POINTS IN Z, LOOP',IEL
      DO I=1,G_Z
         WRITE(51,*) GLOB_Z_P(I)
      ENDDO
      ENDIF


!-----------------------------------------------------------------------
!
!     NODAL COORDINATES OF THE CURRENT ELEMENT
      IF(DEBUG) THEN
         WRITE(51,*) 'ELEMENT NUMBER', IEL
         DO I=1,4
             WRITE(51,20) XE(IEL,I),YE(IEL,I)
         ENDDO
      ENDIF
      X_I=XE(IEL,1);X_J=XE(IEL,2);X_K=XE(IEL,3);X_L=XE(IEL,4)
      Y_I=YE(IEL,1);Y_J=YE(IEL,2);Y_K=YE(IEL,3);Y_L=YE(IEL,4)
      IF(DEBUG) THEN
         WRITE(51,*) 'GLOBAL NODAL COORDINATES'
         WRITE(51,*) 'ELEMENT NUMBER', IEL
         WRITE(51,21) X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L
      ENDIF
      X_CENT=(X_I+X_J+X_K+X_L)/4.0D0
      Y_CENT=(Y_I+Y_J+Y_K+Y_L)/4.0D0
!
```

```
!
      G_XY=G_X*G_Y
      DO I=1,G_X
         GLOB_X_P(I)=X_I+LE/2.0D0*X_P(I)+LE/2.0D0
      ENDDO
      DO I=1,G_Y
         GLOB_Y_P(I)=Y_I+BE/2.0D0*Y_P(I)+BE/2.0D0
      ENDDO

      ALLOCATE(COORDG(G_XY,2),COORDL(G_XY,2))

      DO I=1,G_X
         DO J=1,G_Y
             COORDG((I-1)*G_Y+J,1)=GLOB_X_P(J)
             COORDG((I-1)*G_Y+J,2)=GLOB_Y_P(I)
         ENDDO
      ENDDO

      DO I=1,G_X
         DO J=1,G_Y
             COORDL((I-1)*G_Y+J,1)=X_P(J)
             COORDL((I-1)*G_Y+J,2)=Y_P(I)
         ENDDO
      ENDDO


      IF(DEBUG) THEN
          WRITE(51,*) 'ELEMENT NUMBER :',IEL
          WRITE(51,*) 'GLOBAL COORDINATES OF GAUSS POINTS IN XY'
          DO I= 1,G_XY
             WRITE(51,20) (COORDG(I,J), J=1,2)
          ENDDO
          WRITE(51,*) 'GLOBAL COORDINATES Z'
          DO I= 1,G_Z
            WRITE(51,21) GLOB_Z_P(I)
          ENDDO
      ENDIF
!
      DO I =1,G_XY
         DO J=1,G_Z
             ELE_STRAIN((J-1)*G_XY+I,1)=COORDG(I,1)
             ELE_STRAIN((J-1)*G_XY+I,2)=COORDG(I,2)
             ELE_STRAIN((J-1)*G_XY+I,3)=GLOB_Z_P(J)
             ELE_STRESS((J-1)*G_XY+I,1)=COORDG(I,1)
             ELE_STRESS((J-1)*G_XY+I,2)=COORDG(I,2)
             ELE_STRESS((J-1)*G_XY+I,3)=GLOB_Z_P(J)
             LOCAL_C((J-1)*G_XY+I,1)=COORDL(I,1)
             LOCAL_C((J-1)*G_XY+I,2)=COORDL(I,2)
             LOCAL_C((J-1)*G_XY+I,3)=GLOB_Z_P(J)
         ENDDO
      ENDDO
!
      IF(DEBUG) THEN
          WRITE(51,*) 'NODAL_DISP FROM STRESS_CAL, LOOP -',IEL
          WRITE(51,20) NODAL_DISP
          WRITE(51,*) 'TOTAL SOLUTION VECTOR'
          WRITE(51,20) SOL
      ENDIF

      IF (SRS_ID == 0) THEN
            ALLOCATE(STRESS_OUT(G_Z,9),STRAIN_OUT(G_Z,9))
      ELSEIF (SRS_ID == 1) THEN
            ALLOCATE(STRESS_OUT(G_Z,12),STRAIN_OUT(G_Z,9))
      ENDIF

      STRESS_OUT=0.0D0


      CALL STRAIN_CAL (ELE_STRESS,ELE_STRAIN,N_FEM,T_P,IEL,LE,BE
```

```
     + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,I_ORDER,LOCAL_C,
     + X_CENT,Y_CENT,G_XY,G_Z,NODAL_ACC,RHO)


      IF (SRS_ID==1) THEN
        CALL SRS(N_FEM,T_P,IELX,IELY,IEL,LE,BE,H
     +     ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,SOL,CON,CC,I_ORDER,LOCAL_C,
     +     X_CENT,Y_CENT,G_XY,G_Z,NODAL_ACC,RHO,Z_P,Z_W,STRESS_OUT,
     +     N_NODE,GLOB_Z_P,COORDG,COORDL,TOT_DOF,E_DOF,NODAL_DISP,
     +     N_L,N_B,GLOB_XYZ,AN)
      ENDIF
      DO I=1,G_Z
         STRESS_OUT(I,1)=X_CENT
         STRESS_OUT(I,2)=Y_CENT
         STRESS_OUT(I,3)=ELE_STRESS((I-1)*G_XY+1,3)
         STRAIN_OUT(I,1)=X_CENT
         STRAIN_OUT(I,2)=Y_CENT
         STRAIN_OUT(I,3)=ELE_STRESS((I-1)*G_XY+1,3)
         DO J=4,9
             STRESS_OUT(I,J)=(ELE_STRESS((I-1)*G_XY+1,J)+
     +       ELE_STRESS((I-1)*G_XY+1+1,J)+ELE_STRESS((I-1)*G_XY+1+2,J)+
     +       ELE_STRESS((I-1)*G_XY+1+3,J))/4.0D0
             STRAIN_OUT(I,J)=(ELE_STRAIN((I-1)*G_XY+1,J)+
     +       ELE_STRAIN((I-1)*G_XY+1+1,J)+ELE_STRAIN((I-1)*G_XY+1+2,J)+
     +       ELE_STRAIN((I-1)*G_XY+1+3,J))/4.0D0
         ENDDO
      ENDDO

      IF (SRS_ID==0) THEN
             DO I=1,G_Z
                    WRITE(917,20) (STRESS_OUT(I,J),J=1,9)
             ENDDO
      ELSEIF (SRS_ID==1) THEN
             DO I=1,G_Z
                    WRITE(917,20) (STRESS_OUT(I,J),J=1,12)
             ENDDO

      ENDIF
      DO I=1,G_Z
             WRITE(918,20) (STRAIN_OUT(I,J),J=1,9)
      ENDDO
      IF(OUTPUT) THEN
         DO I=1,T_P
             WRITE(16,20) (ELE_STRAIN(I,J), J=1,9),TIME
         ENDDO
         DO I=1,T_P
             WRITE(17,20) (ELE_STRESS(I,J), J=1,9)
         ENDDO
      ENDIF

      WRITE(51,*) 'END OF SUBROUTINE STRESS CAL'

!-----------------------------------------------------------------------
!-----------------------------------------------------------------------

!
      FORMAT (EN11.2)
      FORMAT (999(2x,EN14.4))
      END SUBROUTINE STRESS_CAL
!!
!!
!!    SUBROUTINE TO CALCULATE OPERATOR MATRIX
      SUBROUTINE STRAIN_CAL (ELE_STRESS,ELE_STRAIN,N_FEM,T_P,IEL,LE,BE
     + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,I_ORDER,LOCAL_C
     + ,X_CENT,Y_CENT,G_XY,G_Z)
      USE PROG_DEBUG
      IMPLICIT NONE
      INTEGER :: N_FEM, T_P, I, J, K, IEL, G_XY, G_Z
```

```
      REAL(KIND=8) :: ELE_STRESS(T_P,9), ELE_STRAIN(T_P,9)
      REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,GN1,GN2,GN3,GN4
      REAL(KIND=8) :: N1,N2,N3,N4,
     +  DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J
!     VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
      REAL(KIND=8) :: B(6,48),NODAL_DISP(48,1),NODAL_ACC(48,1)
      REAL(KIND=8) :: TEMP(6,1)
      REAL(KIND=8) :: CC(6,6),RHO
      REAL(KIND=8) :: LOCAL_C(T_P,3)
      INTEGER :: I_ORDER
!     CENTROIDAL COORDINATES
      REAL(KIND=8) :: X_CENT,Y_CENT

!
      B=0.0D0

      IF(DEBUG) THEN
          WRITE(51,*) 'NODAL_DISP FROM STRAIN_CAL, LOOP -',IEL
          WRITE(51,20) NODAL_DISP
      ENDIF
!
!
      WRITE(51,*) 'NODAL DISPLACEMENTS TRANSFERED SUCCESFULLY'
      WRITE(51,*) 'LOOP', IEL

      open(62,file='TEST.dat')
      DO I=1,T_P
         CALL FEM_SHAPE_DIFF(LOCAL_C(I,1),LOCAL_C(I,2),I_ORDER,
     +   DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
         CALL FEM_SHAPE_GLOBAL(X_I,X_J,Y_I,Y_L,ELE_STRAIN(I,1),
     +   ELE_STRAIN(I,2),GN1,GN2,GN3,GN4,LE,BE)
c         dn1x=1; dn2x=1; dn3x=1; dn4x=1;
c         dn1y=2; dn2y=2; dn3y=2; dn4y=2;
c         gn1=3; gn2=3; gn3=3; gn4=3;
         DO J=1,4
           B(1,4*J)=DN4X*ELE_STRAIN(I,3)**(J-1)
           B(1,4*J-1)=DN3X*ELE_STRAIN(I,3)**(J-1)
           B(1,4*J-2)=DN2X*ELE_STRAIN(I,3)**(J-1)
           B(1,4*J-3)=DN1X*ELE_STRAIN(I,3)**(J-1)
           B(2,4*J+16)=DN4Y*ELE_STRAIN(I,3)**(J-1)
           B(2,4*J-1+16)=DN3Y*ELE_STRAIN(I,3)**(J-1)
           B(2,4*J-2+16)=DN2Y*ELE_STRAIN(I,3)**(J-1)
           B(2,4*J-3+16)=DN1Y*ELE_STRAIN(I,3)**(J-1)
!
           B(4,4*J+32)=DN4Y*ELE_STRAIN(I,3)**(J-1)
           B(4,4*J-1+32)=DN3Y*ELE_STRAIN(I,3)**(J-1)
           B(4,4*J-2+32)=DN2Y*ELE_STRAIN(I,3)**(J-1)
           B(4,4*J-3+32)=DN1Y*ELE_STRAIN(I,3)**(J-1)
!
           B(5,4*J+32)=DN4X*ELE_STRAIN(I,3)**(J-1)
           B(5,4*J-1+32)=DN3X*ELE_STRAIN(I,3)**(J-1)
           B(5,4*J-2+32)=DN2X*ELE_STRAIN(I,3)**(J-1)
           B(5,4*J-3+32)=DN1X*ELE_STRAIN(I,3)**(J-1)
!
           B(6,4*J)=DN4Y*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J-1)=DN3Y*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J-2)=DN2Y*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J-3)=DN1Y*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J+16)=DN4X*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J-1+16)=DN3X*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J-2+16)=DN2X*ELE_STRAIN(I,3)**(J-1)
           B(6,4*J-3+16)=DN1X*ELE_STRAIN(I,3)**(J-1)
         ENDDO
         DO J=2,4
           B(3,4*J+32)=(J-1)*GN4*ELE_STRAIN(I,3)**(J-2)
           B(3,4*J-1+32)=(J-1)*GN3*ELE_STRAIN(I,3)**(J-2)
           B(3,4*J-2+32)=(J-1)*GN2*ELE_STRAIN(I,3)**(J-2)
           B(3,4*J-3+32)=(J-1)*GN1*ELE_STRAIN(I,3)**(J-2)
!          SHEAR STRESS OPERATORS
```

```
              B(4,4*J+16)=(J-1)*GN4*ELE_STRAIN(I,3)**(J-2)
              B(4,4*J-1+16)=(J-1)*GN3*ELE_STRAIN(I,3)**(J-2)
              B(4,4*J-2+16)=(J-1)*GN2*ELE_STRAIN(I,3)**(J-2)
              B(4,4*J-3+16)=(J-1)*GN1*ELE_STRAIN(I,3)**(J-2)
!
              B(5,4*J)=(J-1)*GN4*ELE_STRAIN(I,3)**(J-2)
              B(5,4*J-1)=(J-1)*GN3*ELE_STRAIN(I,3)**(J-2)
              B(5,4*J-2)=(J-1)*GN2*ELE_STRAIN(I,3)**(J-2)
              B(5,4*J-3)=(J-1)*GN1*ELE_STRAIN(I,3)**(J-2)
            ENDDO
            IF(DEBUG) THEN
            WRITE(62,*) 'POINT NO', I
            DO J=1,6
                WRITE(62,20) (B(J,K), K=1,48)
            ENDDO
            ENDIF
            TEMP=MATMUL(B,NODAL_DISP)
            ELE_STRAIN(I,4)=TEMP(1,1)
            ELE_STRAIN(I,5)=TEMP(2,1)
            ELE_STRAIN(I,6)=TEMP(3,1)
            ELE_STRAIN(I,7)=TEMP(4,1)
            ELE_STRAIN(I,8)=TEMP(5,1)
            ELE_STRAIN(I,9)=TEMP(6,1)

            ELE_STRESS(I,4)=CC(1,1)*ELE_STRAIN(I,4)
     +    + CC(1,2)*ELE_STRAIN(I,5)+CC(1,3)*ELE_STRAIN(I,6)+
     +    CC(1,4)*ELE_STRAIN(I,7)+CC(1,5)*ELE_STRAIN(I,8)+
     +    CC(1,6)*ELE_STRAIN(I,9)
            ELE_STRESS(I,5)=CC(2,1)*ELE_STRAIN(I,4)
     +    + CC(2,2)*ELE_STRAIN(I,5)+CC(2,3)*ELE_STRAIN(I,6)+
     +    CC(2,4)*ELE_STRAIN(I,7)+CC(2,5)*ELE_STRAIN(I,8)+
     +    CC(2,6)*ELE_STRAIN(I,9)

            ELE_STRESS(I,6)=CC(3,1)*ELE_STRAIN(I,4)
     +    + CC(3,2)*ELE_STRAIN(I,5)+CC(3,3)*ELE_STRAIN(I,6)+
     +    CC(3,4)*ELE_STRAIN(I,7)+CC(3,5)*ELE_STRAIN(I,8)+
     +    CC(3,6)*ELE_STRAIN(I,9)

            ELE_STRESS(I,7)=CC(4,4)*ELE_STRAIN(I,7)+CC(4,5)
     +    *ELE_STRAIN(I,8)
            ELE_STRESS(I,8)=CC(5,5)*ELE_STRAIN(I,8)+CC(5,4)
     +    *ELE_STRAIN(I,7)
            ELE_STRESS(I,9)=CC(6,1)*ELE_STRAIN(I,4)
     +    + CC(6,2)*ELE_STRAIN(I,5)+CC(6,3)*ELE_STRAIN(I,6)+
     +    CC(6,4)*ELE_STRAIN(I,7)+CC(6,5)*ELE_STRAIN(I,8)+
     +    CC(6,6)*ELE_STRAIN(I,9)
            IF(DEBUG) THEN
             WRITE(51,*) 'THE B MATRIX AT', ELE_STRAIN(I,1)
     +                                              ,ELE_STRAIN(I,2)
             DO K=1,6
                WRITE(51,20) (B(K,J),J=1,48)
             ENDDO
            ENDIF


         ENDDO



            close(62)
!
!
!
!
!
         FORMAT (EN11.2)
         FORMAT (999(x,EN14.4))
         END SUBROUTINE STRAIN_CAL

```

```
4347
4348  !-----------------------------------------------------------------------
4349  !-----------------------------------------------------------------------
4350  !-----------------------------------------------------------------------
4351  !-----------------------------------------------------------------------
4352  !      MAKING OF SPARSE MATRICES
4353  !-----------------------------------------------------------------------
4354  !-----------------------------------------------------------------------
4355  !-----------------------------------------------------------------------
4356        SUBROUTINE SPARSE_ASSM(KMG_VAL,TOT_DOF,NZ,KM,ROWIND_KM,COL_KM)
4357        USE PROG_DEBUG
4358        IMPLICIT NONE
4359
4360        INTEGER :: TOT_DOF,NZ,NUM,I,J,K,L
4361        REAL(KIND=8) :: KMG_VAL(NZ),KM(TOT_DOF,TOT_DOF)
4362        INTEGER :: COL_KM(NZ)
4363       + ,ROWIND_KM(TOT_DOF+1)
4364
4365        NUM=0
4366        DO J=1,TOT_DOF
4367           IF (KM(1,J) .NE. 0.0D0) THEN
4368               NUM=NUM+1
4369               ROWIND_KM(1)=NUM
4370               GOTO 1234
4371           ENDIF
4372        ENDDO
4373        CONTINUE
4374
4375
4376        NUM=0
4377        DO I=1,TOT_DOF
4378           DO J=1,TOT_DOF
4379               IF (KM(I,J) .NE. 0.0D0) THEN
4380               NUM=NUM+1
4381               KMG_VAL(NUM)=KM(I,J)
4382               COL_KM(NUM)=J
4383               ROWIND_KM(I+1)=NUM+1
4384               ENDIF
4385           ENDDO
4386        ENDDO
4387  !      write(*,*) 'hello',num
4388  !      write(*,*) 'bye',rowind_k(tot_dof+1)
4389  !      do i=1,tot_dof+1
4390  !         write(183,22) rowind_k
4391  !      enddo
4392  !
4393        IF (OUTPUT) THEN
4394            OPEN(123,FILE='KG_VAL.DAT')
4395            DO I=1,NZ
4396               WRITE(123,21) KMG_VAL(I)
4397            ENDDO
4398            CLOSE(123)
4399            OPEN(124,FILE='MG_VAL.DAT')
4400            DO I=1,NZ
4401               WRITE(124,21) KMG_VAL(I)
4402            ENDDO
4403            CLOSE(124)
4404            OPEN(125,FILE='COL_K.DAT')
4405            DO I=1,NZ
4406               WRITE(125,23) COL_KM(I)
4407            ENDDO
4408            CLOSE(125)
4409            OPEN(126,FILE='COL_M.DAT')
4410            DO I=1,NZ
4411               WRITE(126,23) COL_KM(I)
4412            ENDDO
4413            CLOSE(126)
4414            OPEN(127,FILE='ROWIND_K.DAT')
4415            DO I=1,TOT_DOF+1
```

```
4416                WRITE(127,23) ROWIND_KM(I)
4417             ENDDO
4418             CLOSE(127)
4419             OPEN(128,FILE='ROWIND_M.DAT')
4420             DO I=1,TOT_DOF+1
4421                WRITE(128,23) ROWIND_KM(I)
4422             ENDDO
4423             CLOSE(128)
4424          ENDIF
4425
4426          FORMAT (9999(x,EN14.4))
4427          FORMAT(EN12.3)
4428          FORMAT(I5)
4429          FORMAT (999(2X,I5))
4430          END SUBROUTINE SPARSE_ASSM
4431
4432          SUBROUTINE SOLVER_F77_TEST(ROWINDEX, COLUMNS, VALUES, RHS, NROWS,
4433         + NCOLS, NNONZEROS,NRHS, SOLUTION)
4434          IMPLICIT NONE
4435          INCLUDE 'mkl_dss.f77'
4436     C---------------------------------------------------------------------
4437     C DEFINE THE ARRAY AND RHS VECTORS
4438     C---------------------------------------------------------------------
4439          INTEGER NROWS, NCOLS, NNONZEROS, I, NRHS
4440          INTEGER ROWINDEX(NROWS + 1), COLUMNS(NNONZEROS)
4441          DOUBLE PRECISION VALUES(NNONZEROS), RHS(NROWS)
4442     c      DATA ROWINDEX / 1, 6, 8, 9, 10, 11 /
4443     c      DATA COLUMNS / 1, 2, 3, 4, 5, 2, 5, 3, 4, 5 /
4444     c      DATA VALUES / 9, 1.5, 6, .75, 3, 0.5, 0, 12, .625, 16 /
4445     c      DATA RHS / 1, 2, 3, 4, 5 /
4446     C---------------------------------------------------------------------
4447     C ALLOCATE STORAGE FOR THE SOLVER HANDLE AND THE SOLUTION VECTOR
4448     C---------------------------------------------------------------------
4449          DOUBLE PRECISION SOLUTION(NROWS)
4450          INTEGER*8 HANDLE
4451          INTEGER ERROR
4452          CHARACTER*15 STATIN
4453          DOUBLE PRECISION STATOUT(5)
4454          INTEGER BUFLEN
4455          PARAMETER(BUFLEN = 20)
4456          INTEGER BUFF(BUFLEN)
4457     C---------------------------------------------------------------
4458     C     DEBUGGING
4459     c---------------------------------------------------------------
4460     C      WRITE(375,*) '------------------------'
4461     C      WRITE(375,20) RHS
4462     C---------------------------------------------------------------------
4463     C INITIALIZE THE SOLVER
4464     C---------------------------------------------------------------------
4465          ERROR = DSS_CREATE(HANDLE, MKL_DSS_DEFAULTS)
4466          IF (ERROR .NE. MKL_DSS_SUCCESS ) GOTO 999
4467
4468     C---------------------------------------------------------------------
4469     C DEFINE THE NON-ZERO STRUCTURE OF THE MATRIX
4470     C---------------------------------------------------------------------
4471          ERROR = DSS_DEFINE_STRUCTURE( HANDLE, MKL_DSS_SYMMETRIC,
4472         + ROWINDEX, NROWS, NCOLS, COLUMNS, NNONZEROS )
4473          IF (ERROR .NE. MKL_DSS_SUCCESS ) GOTO 999
4474
4475     C---------------------------------------------------------------------
4476     C REORDER THE MATRIX
4477     C---------------------------------------------------------------------
4478          ERROR = DSS_REORDER( HANDLE, MKL_DSS_DEFAULTS, 0)
4479          IF (ERROR .NE. MKL_DSS_SUCCESS ) GOTO 999
4480
4481     C---------------------------------------------------------------------
4482     C FACTOR THE MATRIX
4483     C---------------------------------------------------------------------
4484          ERROR = DSS_FACTOR_REAL( HANDLE,
```

```
     + MKL_DSS_DEFAULTS, VALUES)
      IF (ERROR .NE. MKL_DSS_SUCCESS ) GOTO 999

C------------------------------------------------------------------------
C GET THE SOLUTION VECTOR
C------------------------------------------------------------------------
      ERROR = DSS_SOLVE_REAL( HANDLE, MKL_DSS_DEFAULTS,
     + RHS, NRHS, SOLUTION)
      IF (ERROR .NE. MKL_DSS_SUCCESS ) GOTO 999
C------------------------------------------------------------------------
C PRINT DETERMINANT OF THE MATRIX
C------------------------------------------------------------------------
      STATIN = 'DETERMINANT'
      CALL MKL_CVT_TO_NULL_TERMINATED_STR(BUFF,BUFLEN,STATIN)
      ERROR = DSS_STATISTICS(HANDLE, MKL_DSS_DEFAULTS,
     + BUFF,STATOUT)
c      WRITE(*,"(' POW OF DETERMINANT IS ', 5(F10.3))") STATOUT(1)
c      WRITE(*,"(' BASE OF DETERMINANT IS ', 5(F10.3))") STATOUT(2)
c      WRITE(*,"(' DETERMINANT IS ', 5(F10.3))")(10**STATOUT(1))*
c     + STATOUT(2)

C------------------------------------------------------------------------
C DEALLOCATE SOLVER STORAGE
C------------------------------------------------------------------------
      ERROR = DSS_DELETE( HANDLE, MKL_DSS_DEFAULTS )
      IF (ERROR .NE. MKL_DSS_SUCCESS ) GOTO 999
C------------------------------------------------------------------------
C PRINT SOLUTION VECTOR
C------------------------------------------------------------------------

      GOTO 1000
      WRITE(*,*) "SOLVER RETURNED ERROR CODE ", ERROR

      FORMAT (9999(x,EN14.4))
      FORMAT(EN12.3)
      FORMAT(F5.0)
      FORMAT (999(2X,I5))
      END



       SUBROUTINE INVERSE(A,C,N)
       !============================================================
       ! INVERSE MATRIX
       ! METHOD: BASED ON DOOLITTLE LU FACTORIZATION FOR AX=B
       ! ALEX G. DECEMBER 2009
       !------------------------------------------------------------
       ! INPUT ...
       ! A(N,N) - ARRAY OF COEFFICIENTS FOR MATRIX A
       ! N      - DIMENSION
       ! OUTPUT ...
       ! C(N,N) - INVERSE MATRIX OF A
       ! COMMENTS ...
       ! THE ORIGINAL MATRIX A(N,N) WILL BE DESTROYED
       ! DURING THE CALCULATION
       !============================================================
       IMPLICIT NONE
       INTEGER N
       DOUBLE PRECISION A(N,N), C(N,N)
       DOUBLE PRECISION L(N,N), U(N,N), B(N), D(N), X(N)
       DOUBLE PRECISION COEFF
       INTEGER I, J, K

       ! STEP 0: INITIALIZATION FOR MATRICES L AND U AND B
       ! FORTRAN 90/95 ALOOWS SUCH OPERATIONS ON MATRICES
       L=0.0
       U=0.0
       B=0.0

```

```
        ! STEP 1: FORWARD ELIMINATION
        DO K=1, N-1
           DO I=K+1,N
              COEFF=A(I,K)/A(K,K)
              L(I,K) = COEFF
              DO J=K+1,N
                 A(I,J) = A(I,J)-COEFF*A(K,J)
              END DO
           END DO
        END DO

        ! STEP 2: PREPARE L AND U MATRICES
        ! L MATRIX IS A MATRIX OF THE ELIMINATION COEFFICIENT
        ! + THE DIAGONAL ELEMENTS ARE 1.0
        DO I=1,N
          L(I,I) = 1.0
        END DO
        ! U MATRIX IS THE UPPER TRIANGULAR PART OF A
        DO J=1,N
          DO I=1,J
            U(I,J) = A(I,J)
          END DO
        END DO

        ! STEP 3: COMPUTE COLUMNS OF THE INVERSE MATRIX C
        DO K=1,N
          B(K)=1.0
          D(1) = B(1)
        ! STEP 3A: SOLVE LD=B USING THE FORWARD SUBSTITUTION
          DO I=2,N
            D(I)=B(I)
            DO J=1,I-1
              D(I) = D(I) - L(I,J)*D(J)
            END DO
          END DO
        ! STEP 3B: SOLVE UX=D USING THE BACK SUBSTITUTION
          X(N)=D(N)/U(N,N)
          DO I = N-1,1,-1
            X(I) = D(I)
            DO J=N,I+1,-1
              X(I)=X(I)-U(I,J)*X(J)
            END DO
            X(I) = X(I)/U(I,I)
          END DO
        ! STEP 3C: FILL THE SOLUTIONS X(N) INTO COLUMN K OF C
          DO I=1,N
            C(I,K) = X(I)
          END DO
          B(K)=0.0
        END DO
        END SUBROUTINE INVERSE

       SUBROUTINE INT_FORCE (TOT_DOF,H,N_FEM,N_NODE,
      + NODE_L,NODE_B,E_DOF,G_X,G_Y,G_Z,T_P,LE,BE,
      + NODAL_DISP,CC,XE,YE,E_NODE,I_ORDER,TIME,SOL,CON,NL2,NB2,NH2,
      + N_NODE2,N_L,N_B,TIME_STEP,CON2,CON3,FFIN,I_VON,TU,SRS_ID,AN,
      + GLOB_XYZ)
       USE PROG_DEBUG
       IMPLICIT NONE
       INTEGER :: I_BC,TOT_DOF,N_NODE,I,J,K,L,NODE_L,NODE_B,NL2,NB2,NH2
       INTEGER :: N_NODE2,N_B,I_VON,N_L,SRS_ID
       INTEGER :: TIME_STEP,REM,QUO
!      DEFINITIONS AS IN
       INTEGER :: G_X,G_Y,G_Z,N_FEM,G_XY,T_P,E_NODE
       INTEGER :: IEL,E_DOF,IELX,IELY
       REAL(KIND=8) :: H
       REAL(KIND=8),ALLOCATABLE :: Z_P(:), Z_W(:), X_P(:), X_W(:),
      +                            Y_P(:),Y_W(:)
!      GLOBAL COORDINATES OF GAUSS POINTS IN Z,X AND Y
```

```
      REAL(KIND=8) :: X_CENT,Y_CENT
!     X, Y COORDINATES OF CENTROID OF EACH ELEMENT
!
      REAL(KIND=8) :: ELE_STRAIN(T_P,9)
      REAL(KIND=8) :: ELE_STRESS(T_P,24)
      REAL(KIND=8) :: ELE_STRAIN2(T_P,9)
      REAL(KIND=8) :: ELE_STRESS2(T_P,24)
      REAL(KIND=8),ALLOCATABLE :: STRESS_OUT(:,:),STRAIN_OUT(:,:)
     + ,DISP_OUT(:,:)
!     OUTPUT STRESS AT THE CENTROID OF EACH ELEMENT
      REAL(KIND=8), ALLOCATABLE :: GLOB_Z_P(:),GLOB_Z_P2(:)
      REAL(KIND=8), ALLOCATABLE :: GLOB_X_P(:)
      REAL(KIND=8), ALLOCATABLE :: GLOB_Y_P(:)
      REAL(KIND=8) :: DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y
      REAL(KIND=8) :: N1,N2,N3,N4,DET_J
      REAL(KIND=8) :: N(E_NODE,1),DNX(E_NODE,1),DNY(E_NODE,1)
      REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L
!     COORDINATES OF THE 1ST, 2ND, 3RD AND 4TH NODE RESP. OF AN ELEMENT
      REAL(KIND=8) :: LE,BE
      REAL(KIND=8) :: NODAL_DISP(48,1),SOL(TOT_DOF),FFIN(TOT_DOF)
      REAL(KIND=8) :: CENT_DISP(12,1)
      REAL(KIND=8),ALLOCATABLE :: FEIN(:)
      REAL(KIND=8) :: CC(6,6),RHO  !MAT STIFF MATRIX, DEN FROM MAIN
      REAL(KIND=8) :: XE(N_FEM,E_NODE),YE(N_FEM,E_NODE)
      REAL(KIND=8),ALLOCATABLE :: COORDL(:,:),COORDG(:,:)
!     COORD. IN XY PLANE (LOCAL AND GLOBAL)
      REAL(KIND=8),ALLOCATABLE :: LOCAL_C(:,:),LOCAL_C2(:,:)
      INTEGER :: I_ORDER
      INTEGER :: CON(N_FEM,E_DOF),CON2(NL2*NB2*NH2,8),CON3(NL2*NB2,4)
      REAL(KIND=8) :: TIME
      REAL(KIND=8) :: TU,TEMP_U,T_U     !TOTAL STRAIN ENERGY
      REAL(KIND=8) :: S_CENT(G_Z,6)  !2ND PK STRESS AT CENT OF Z_P's
      REAL(KIND=8),ALLOCATABLE :: M_TEMP(:,:,:),MOM(:,:,:)
      REAL(KIND=8) :: MOM_NET(9,4)
      REAL(KIND=8) :: INT_F1(12),INT_F2(12),INT_F3(12)
      REAL(KIND=8) :: M_FUN1(G_X,G_Y),M_FUN2(G_X,G_Y),M_FUN3(G_X,G_Y)
      REAL(KIND=8) :: A1(4),A2(4),A3(4),INTEGRAL1(4),INTEGRAL2(4),
     +                INTEGRAL3(4),V1(1,1),V2(1,1),V3(1,1),W_X(1,G_X),
     +                W_Y(1,G_Y),FUNCT1(G_X,G_Y,4),FUNCT2(G_X,G_Y,4),
     +                FUNCT3(G_X,G_Y,4)
      REAL(KIND=8) :: DIVS
      REAL(KIND=8) :: AN(TOT_DOF),NODAL_ACC(E_DOF,1),GLOB_XYZ(N_NODE,2)
!     ALL COMPONENTS OF MOMENT ON THE MID PLANE
!
!
      G_XY=G_X*G_Y      !TOTAL NUMBER OF GAUSS POINTS IN PLANE
!     ALLOCATING DIMENSIONS
      ALLOCATE (Z_P(G_Z),Z_W(G_Z))
      ALLOCATE (X_P(G_X),X_W(G_X))
      ALLOCATE (Y_P(G_Y),Y_W(G_Y))
      ALLOCATE (GLOB_Z_P(G_Z),GLOB_Z_P2(G_Z))
      ALLOCATE (GLOB_X_P(G_X),GLOB_Y_P(G_Y))
      ALLOCATE(LOCAL_C(T_P,3),LOCAL_C2(T_P,3))
      ALLOCATE(COORDG(G_XY,2),COORDL(G_XY,2))
            IF (SRS_ID == 0) THEN
            ALLOCATE(STRESS_OUT(G_Z,21))
      ELSEIF (SRS_ID == 1) THEN
            ALLOCATE(STRESS_OUT(G_Z,29))
      ENDIF
      ALLOCATE(STRAIN_OUT(G_Z,12),DISP_OUT(G_Z,9))
      ALLOCATE(M_TEMP(9,4,G_XY),MOM(9,4,G_XY))
      ALLOCATE(FEIN(E_DOF))


!-----------------------------------------------------------------------
!     INITIALIZING THE MATRICES
      Z_P=0.0D0; Z_W=0.0D0; GLOB_Z_P=0.0D0;
!-----------------------------------------------------------------------
!     CALLING FOR GAUSS POINTS AND WEIGHTS IN X AND Z
```

```
      CALL GAUSS_LEG(G_X,X_P,X_W)
      CALL GAUSS_LEG(G_Z,Z_P,Z_W)
      CALL GAUSS_LEG(G_Y,Y_P,Y_W)

!
!     GLOBAL X3 COORDINATES OF GAUSS POINTS COORDINATES
      GLOB_Z_P=H/2.0D0*Z_P

      !---------------------------------------------------------------------
!     GLOBAL X3 COORDINATES OF Z POINTS FOR STRESS OUT ONLY
      DIVS=G_Z-1
      DO I =1,G_Z
         GLOB_Z_P2(I)= H/DIVS*(I-(G_Z+1)/2)
      ENDDO
!-----------------------------------------------------------------------
      ELE_STRAIN=0.0D0
      ELE_STRESS=0.0D0

!
          QUO=TIME_STEP/1
          REM=TIME_STEP-QUO*1

      IF (TIME_STEP==1 .OR. REM==0) THEN

          IF (SRS_ID==0) THEN

          WRITE(917,*) 'TITLE = "STRESS"'
          WRITE(917,*) 'VARIABLES = x ,y, z,"T11","T22","T33","T23",
     +"T13","T12","T32","T31","T21",
     +"S11", "S22", "S33", "S23","S13", "S12"
     +,"xx","yy","zz"'
c        WRITE(917,*) 'ZONE T= "Undeformed Mesh", SOLUTIONTIME= ',TIME
          WRITE(917,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,
     + ',E =',NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
          WRITE(918,*) 'TITLE = "STRAIN"'
          WRITE(918,*) 'VARIABLES = x ,y, z,"E11","E22","E33","E23",
     +"E13","E12","xx","yy","zz"'
C         WRITE(918,*) 'ZONE T= "Undeformed Mesh", SOLUTIONTIME= ',TIME
          WRITE(918,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,
     + ',E =',NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
          WRITE(919,*) 'TITLE = "DEFORMATION"'
          WRITE(919,*) 'VARIABLES = X, Y, Z, xx, yy, zz, U1, U2, U3'
          WRITE(919,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,
     + ',E =',NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
          WRITE(557,*) 'TITLE = "TRACTIONS"'
          WRITE(557,*) 'VARIABLES = x ,y, "Fx", "Fy", "Fz", "MAG"'
          WRITE(557,*) 'ZONE T= "Undeformed Mesh",N =',N_FEM,'E =',
     + NL2*NB2,', F=FEPOINT SOLUTIONTIME= ',TIME

          ELSEIF (SRS_ID==1) THEN



          WRITE(917,*) 'TITLE = "STRESS"'
          WRITE(917,*) 'VARIABLES = x ,y, z,"T11","T22","T33","T23",
     + "T13","T12","T32","T31","T21",
     + "S11", "S22", "S33", "S23","S13","S12","xx","yy","zz",
     + "T23_SRS","T13_SRS","T32_SRS","T31_SRS","T33_SRS","S23_SRS",
     + "S13_SRS","S33_SRS"'
c        WRITE(917,*) 'ZONE T= "Undeformed Mesh", SOLUTIONTIME= ',TIME
          WRITE(917,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,
     + ',E =',NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
          WRITE(918,*) 'TITLE = "STRAIN"'
          WRITE(918,*) 'VARIABLES = x ,y, z,"E11","E22","E33","E23",
     +"E13","E12","xx","yy","zz"'
C         WRITE(918,*) 'ZONE T= "Undeformed Mesh", SOLUTIONTIME= ',TIME
          WRITE(918,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,
     + ',E =',NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
          WRITE(919,*) 'TITLE = "DEFORMATION"'
```

```
4761            WRITE(919,*) 'VARIABLES = X, Y, Z, xx, yy, zz, U1, U2, U3'
4762            WRITE(919,*) 'ZONE T= "Undeformed Mesh", N =',N_NODE2*N_B,
4763         + ',E =',NL2*NB2*NH2,',F=FEPOINT, SOLUTIONTIME= ',TIME,',ET=BRICK'
4764            WRITE(557,*) 'TITLE = "TRACTIONS"'
4765            WRITE(557,*) 'VARIABLES = x ,y, "Fx", "Fy", "Fz", "MAG"'
4766            WRITE(557,*) 'ZONE T= "Undeformed Mesh",N =',N_FEM,'E =',
4767         + NL2*NB2,', F=FEPOINT SOLUTIONTIME= ',TIME
4768          ENDIF
4769         ENDIF
4770
4771
4772
4773
4774
4775
4776         FFIN=0.0D0; FEIN=0.0D0
4777
4778
4779
4780
4781
4782   !     BEGIN THE LOOP WITH ELEMENTS
4783         TU=0.0D0     !TOTAL STRAIN ENERGY
4784         DO IELY=1,N_B
4785          DO IELX=1,N_L
4786            IEL=(IELY-1)*N_L+IELX
4787
4788   !        INITIALIZING STRAIN ENERGY
4789            TEMP_U=0.0D0
4790
4791   !        COLLECTING NODAL DOF'S FOR EACH ELEMENT
4792            DO J=1,E_DOF
4793                NODAL_DISP(J,1)=SOL(CON(IEL,J))
4794                NODAL_ACC(J,1)=AN(CON(IEL,J))
4795            ENDDO
4796            DO I=1,12
4797                CENT_DISP(I,1)=(NODAL_DISP((I-1)*4+1,1)+NODAL_DISP((I-1)
4798         +       *4+2,1)+NODAL_DISP((I-1)*4+3,1)+NODAL_DISP((I-1)*4+4,1))
4799         +       /4.0D0
4800            ENDDO
4801   !        NODAL COORDINATES OF THE CURRENT ELEMENT (4 VERTICES)
4802            X_I=XE(IEL,1);X_J=XE(IEL,2);X_K=XE(IEL,3);X_L=XE(IEL,4)
4803            Y_I=YE(IEL,1);Y_J=YE(IEL,2);Y_K=YE(IEL,3);Y_L=YE(IEL,4)
4804            IF(DEBUG) THEN
4805                WRITE(51,*) 'GLOBAL NODAL COORDINATES'
4806                WRITE(51,*) 'ELEMENT NUMBER', IEL
4807                WRITE(51,20) X_I,Y_I
4808                WRITE(51,20) X_J,Y_J
4809                WRITE(51,20) X_K,Y_K
4810                WRITE(51,20) X_L,Y_L
4811            ENDIF
4812   !        CENTROIDAL COORDINATES
4813            X_CENT=(X_I+X_J+X_K+X_L)/4.0D0
4814            Y_CENT=(Y_I+Y_J+Y_K+Y_L)/4.0D0
4815
4816
4817   !        GLOBAL AND LOCAL COORDINATES OF THE GAUSS POINTS
4818
4819
4820   !         IF (DEBUG) THEN
4821   !             OPEN(923,FILE='CENTROIDS.DAT')
4822   !             WRITE(923,20) X_CENT,Y_CENT
4823   !         ENDIF
4824
4825
4826
4827
4828
4829
```

```
4830
4831
4832
4833           DO I=1,G_X
4834               GLOB_X_P(I)=X_I+LE/2.0D0*X_P(I)+LE/2.0D0
4835           ENDDO
4836           DO I=1,G_Y
4837               GLOB_Y_P(I)=Y_I+BE/2.0D0*Y_P(I)+BE/2.0D0
4838           ENDDO
4839
4840
4841           DO I=1,G_X
4842               DO J=1,G_Y
4843                   COORDG((I-1)*G_Y+J,1)=GLOB_X_P(J)
4844                   COORDG((I-1)*G_Y+J,2)=GLOB_Y_P(I)
4845               ENDDO
4846           ENDDO
4847
4848           DO I=1,G_X
4849               DO J=1,G_Y
4850                   COORDL((I-1)*G_Y+J,1)=X_P(J)
4851                   COORDL((I-1)*G_Y+J,2)=Y_P(I)
4852               ENDDO
4853           ENDDO
4854
4855   !         IF (DEBUG) THEN
4856   !             OPEN(971,FILE='COORD_G.DAT')
4857   !             DO I=1,G_XY
4858   !                 WRITE(971,20) (COORDG(I,J),J=1,2)
4859   !             ENDDO
4860   !         ENDIF
4861
4862
4863
4864   !         RECORDING COORDINATES IN THE STRAIN AND STRESS OUTPUT FILES
4865           DO I =1,G_XY
4866               DO J=1,G_Z
4867                   ELE_STRAIN((J-1)*G_XY+I,1)=COORDG(I,1)
4868                   ELE_STRAIN((J-1)*G_XY+I,2)=COORDG(I,2)
4869                   ELE_STRAIN((J-1)*G_XY+I,3)=GLOB_Z_P(J)
4870                   ELE_STRESS((J-1)*G_XY+I,1)=COORDG(I,1)
4871                   ELE_STRESS((J-1)*G_XY+I,2)=COORDG(I,2)
4872                   ELE_STRESS((J-1)*G_XY+I,3)=GLOB_Z_P(J)
4873                   LOCAL_C((J-1)*G_XY+I,1)=COORDL(I,1)
4874                   LOCAL_C((J-1)*G_XY+I,2)=COORDL(I,2)
4875                   LOCAL_C((J-1)*G_XY+I,3)=GLOB_Z_P(J)
4876               ENDDO
4877           ENDDO
4878           DO I =1,G_XY
4879               DO J=1,G_Z
4880                   ELE_STRAIN2((J-1)*G_XY+I,1)=COORDG(I,1)
4881                   ELE_STRAIN2((J-1)*G_XY+I,2)=COORDG(I,2)
4882                   ELE_STRAIN2((J-1)*G_XY+I,3)=GLOB_Z_P2(J)
4883                   ELE_STRESS2((J-1)*G_XY+I,1)=COORDG(I,1)
4884                   ELE_STRESS2((J-1)*G_XY+I,2)=COORDG(I,2)
4885                   ELE_STRESS2((J-1)*G_XY+I,3)=GLOB_Z_P2(J)
4886                   LOCAL_C2((J-1)*G_XY+I,1)=COORDL(I,1)
4887                   LOCAL_C2((J-1)*G_XY+I,2)=COORDL(I,2)
4888                   LOCAL_C2((J-1)*G_XY+I,3)=GLOB_Z_P2(J)
4889               ENDDO
4890           ENDDO
4891   !       CALLING SUBROUTINE TO GET STRAIN AND STRESS VALUES
4892            CALL NONLIN_STRAIN (ELE_STRESS,ELE_STRAIN,N_FEM,T_P,IEL,LE,BE
4893       +  ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,I_ORDER,LOCAL_C,
4894       +   TIME,I_VON)
4895
4896            CALL NONLIN_STRAIN(ELE_STRESS2,ELE_STRAIN2,N_FEM,T_P,IEL,LE,BE
4897       +  ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,I_ORDER,LOCAL_C2,
4898       +   TIME,I_VON)
```

```
4899
4900
4901
4902
4903  !         WRITING STRAIN AND STRESS VALUES AT THE CENTROID OF EACH ELE
4904               STRESS_OUT=0.0D0
4905               S_CENT=0.0D0
4906  C              DO I=1,G_Z
4907  C                 STRESS_OUT(I,1)=X_CENT
4908  C                 STRESS_OUT(I,2)=Y_CENT
4909  C                 STRESS_OUT(I,3)=ELE_STRESS((I-1)*G_XY+1,3)
4910  C                 STRAIN_OUT(I,1)=X_CENT
4911  C                 STRAIN_OUT(I,2)=Y_CENT
4912  C                 STRAIN_OUT(I,3)=ELE_STRESS((I-1)*G_XY+1,3)
4913  C                 DO J=4,18
4914  C                     STRESS_OUT(I,J)=(ELE_STRESS((I-1)*G_XY+1,J)+
4915  C      +          ELE_STRESS((I-1)*G_XY+1+1,J)+ELE_STRESS((I-1)*G_XY+1+2,J)+
4916  C      +          ELE_STRESS((I-1)*G_XY+1+3,J))/4.0D0
4917  C                 ENDDO
4918  C                 DO J=19,24
4919  C                     S_CENT(I,J-18)=(ELE_STRESS((I-1)*G_XY+1,J)+
4920  C      +          ELE_STRESS((I-1)*G_XY+1+1,J)+ELE_STRESS((I-1)*G_XY+1+2,J)+
4921  C      +          ELE_STRESS((I-1)*G_XY+1+3,J))/4.0D0
4922  C                 ENDDO
4923
4924  C                 DO J=4,9
4925  C                     STRAIN_OUT(I,J)=(ELE_STRAIN((I-1)*G_XY+1,J)+
4926  C      +          ELE_STRAIN((I-1)*G_XY+1+1,J)+ELE_STRAIN((I-1)*G_XY+1+2,J)+
4927  C      +          ELE_STRAIN((I-1)*G_XY+1+3,J))/4.0D0
4928  C                 ENDDO
4929  C                 DISP_OUT(I,1)=X_CENT
4930  C                 DISP_OUT(I,2)=Y_CENT
4931  C                 DISP_OUT(I,3)=ELE_STRESS((I-1)*G_XY+1,3)
4932  C                 DISP_OUT(I,7)=CENT_DISP(1,1)+DISP_OUT(I,3)*CENT_DISP(2,1)+
4933  C      +          DISP_OUT(I,3)**2.0D0*CENT_DISP(3,1)+
4934  C      +          DISP_OUT(I,3)**3.0D0*CENT_DISP(4,1)
4935  C                 DISP_OUT(I,8)=CENT_DISP(5,1)+DISP_OUT(I,3)*CENT_DISP(6,1)+
4936  C      +          DISP_OUT(I,3)**2.0D0*CENT_DISP(7,1)+
4937  C      +          DISP_OUT(I,3)**3.0D0*CENT_DISP(8,1)
4938  C                 DISP_OUT(I,9)=CENT_DISP(9,1)+DISP_OUT(I,3)*CENT_DISP(10,1)+
4939  C      +          DISP_OUT(I,3)**2.0D0*CENT_DISP(11,1)+
4940  C      +          DISP_OUT(I,3)**3.0D0*CENT_DISP(12,1)
4941  C                 DISP_OUT(I,4)=DISP_OUT(I,1)+DISP_OUT(I,7)
4942  C                 DISP_OUT(I,5)=DISP_OUT(I,2)+DISP_OUT(I,8)
4943  C                 DISP_OUT(I,6)=DISP_OUT(I,3)+DISP_OUT(I,9)
4944  C                 STRESS_OUT(I,19)=DISP_OUT(I,4)
4945  C                 STRESS_OUT(I,20)=DISP_OUT(I,5)
4946  C                 STRESS_OUT(I,21)=DISP_OUT(I,6)
4947  C                 STRAIN_OUT(I,10)=DISP_OUT(I,4)
4948  C                 STRAIN_OUT(I,11)=DISP_OUT(I,5)
4949  C                 STRAIN_OUT(I,12)=DISP_OUT(I,6)
4950  C              ENDDO
4951               DO I=1,G_Z
4952                  STRESS_OUT(I,1)=X_CENT
4953                  STRESS_OUT(I,2)=Y_CENT
4954                  STRESS_OUT(I,3)=ELE_STRESS2((I-1)*G_XY+1,3)
4955                  STRAIN_OUT(I,1)=X_CENT
4956                  STRAIN_OUT(I,2)=Y_CENT
4957                  STRAIN_OUT(I,3)=ELE_STRESS2((I-1)*G_XY+1,3)
4958                  DO J=4,18
4959                      STRESS_OUT(I,J)=(ELE_STRESS2((I-1)*G_XY+1,J)+
4960         +          ELE_STRESS2((I-1)*G_XY+1+1,J)+ELE_STRESS2((I-1)*G_XY+1+2,J)
4961         +          +ELE_STRESS2((I-1)*G_XY+1+3,J))/4.0D0
4962                  ENDDO
4963                  DO J=19,24
4964                      S_CENT(I,J-18)=(ELE_STRESS2((I-1)*G_XY+1,J)+
4965         +          ELE_STRESS2((I-1)*G_XY+1+1,J)+ELE_STRESS2((I-1)*G_XY+1+2,J)
4966         +          +ELE_STRESS2((I-1)*G_XY+1+3,J))/4.0D0
4967                  ENDDO
```

```
4968
4969                DO J=4,9
4970                    STRAIN_OUT(I,J)=(ELE_STRAIN2((I-1)*G_XY+1,J)+
4971       +        ELE_STRAIN2((I-1)*G_XY+1+1,J)+ELE_STRAIN2((I-1)*G_XY+1+2,J)
4972       +        +ELE_STRAIN2((I-1)*G_XY+1+3,J))/4.0D0
4973                ENDDO
4974                DISP_OUT(I,1)=X_CENT
4975                DISP_OUT(I,2)=Y_CENT
4976                DISP_OUT(I,3)=ELE_STRESS2((I-1)*G_XY+1,3)
4977
4978                DISP_OUT(I,7)=CENT_DISP(1,1)+DISP_OUT(I,3)*CENT_DISP(2,1)+
4979       +        DISP_OUT(I,3)**2.0D0*CENT_DISP(3,1)+
4980       +        DISP_OUT(I,3)**3.0D0*CENT_DISP(4,1)
4981
4982                DISP_OUT(I,8)=CENT_DISP(5,1)+DISP_OUT(I,3)*CENT_DISP(6,1)+
4983       +        DISP_OUT(I,3)**2.0D0*CENT_DISP(7,1)+
4984       +        DISP_OUT(I,3)**3.0D0*CENT_DISP(8,1)
4985
4986                DISP_OUT(I,9)=CENT_DISP(9,1)+DISP_OUT(I,3)*CENT_DISP(10,1)+
4987       +        DISP_OUT(I,3)**2.0D0*CENT_DISP(11,1)+
4988       +        DISP_OUT(I,3)**3.0D0*CENT_DISP(12,1)
4989
4990                DISP_OUT(I,4)=DISP_OUT(I,1)+DISP_OUT(I,7)
4991                DISP_OUT(I,5)=DISP_OUT(I,2)+DISP_OUT(I,8)
4992                DISP_OUT(I,6)=DISP_OUT(I,3)+DISP_OUT(I,9)
4993
4994                STRESS_OUT(I,19)=DISP_OUT(I,4)
4995                STRESS_OUT(I,20)=DISP_OUT(I,5)
4996                STRESS_OUT(I,21)=DISP_OUT(I,6)
4997                STRAIN_OUT(I,10)=DISP_OUT(I,4)
4998                STRAIN_OUT(I,11)=DISP_OUT(I,5)
4999                STRAIN_OUT(I,12)=DISP_OUT(I,6)
5000             ENDDO
5001             IF (SRS_ID==1) THEN
5002
5003
5004
5005
5006
5007             CALL SRS_NLIN(N_FEM,T_P,IELX,IELY,IEL,LE,BE,H
5008       + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,SOL,CON,CC,I_ORDER,LOCAL_C
5009       + ,X_CENT,Y_CENT,G_XY,G_Z,NODAL_ACC,RHO,Z_P,Z_W,STRESS_OUT,N_NODE,
5010       +  GLOB_Z_P2,COORDG,COORDL,TOT_DOF,E_DOF,NODAL_DISP,N_L,N_B,
5011       +  GLOB_XYZ)
5012
5013
5014             ENDIF
5015
5016   !         WRITING STRESS, STRAIN AND DISP OUTPUTS AT DESIRED TIMES
5017             IF (TIME_STEP==1 .OR. REM==0) THEN
5018              IF (SRS_ID==0) THEN
5019                DO I=1,G_Z
5020                       WRITE(917,20) (STRESS_OUT(I,J),J=1,21)
5021                ENDDO
5022                DO I=1,G_Z
5023                       WRITE(918,20) (STRAIN_OUT(I,J),J=1,12)
5024                ENDDO
5025                DO I=1,G_Z
5026                       WRITE(919,20) (DISP_OUT(I,J),J=1,9)
5027                ENDDO
5028   !            WRITING STRESS AND STRAIN OUTPUTS AT EVERY GAUSS POINT IN
5029   !            FILES STRESS.DAT AND STRAINS.DAT
5030                IF(OUTPUT) THEN
5031                       DO I=1,T_P
5032                              WRITE(16,20) (ELE_STRAIN(I,J), J=1,9),TIME
5033                       ENDDO
5034                       DO I=1,T_P
5035                              WRITE(17,20) (ELE_STRESS(I,J), J=1,24),TIME
5036                       ENDDO
```

```
5037                ENDIF
5038             ELSEIF (SRS_ID==1) THEN
5039
5040                DO I=1,G_Z
5041                       WRITE(917,20) (STRESS_OUT(I,J),J=1,29)
5042                ENDDO
5043                DO I=1,G_Z
5044                       WRITE(918,20) (STRAIN_OUT(I,J),J=1,12)
5045                ENDDO
5046                DO I=1,G_Z
5047                       WRITE(919,20) (DISP_OUT(I,J),J=1,9)
5048                ENDDO
5049             ENDIF
5050            ENDIF
5051
5052
5053
5054
5055
5056   !---------------------------------------------------------------------
5057   !          CALCULATING MOMENTS (REFER TO THEORY)
5058   !---------------------------------------------------------------------
5059             M_TEMP=0.0D0
5060             MOM=0.0D0
5061             DO I=1,G_XY
5062                DO J=1,G_Z
5063                    DO K=1,9
5064                        DO L=1,4
5065                            M_TEMP(K,L,I)=GLOB_Z_P(J)**(L-1)*
5066          +                 ELE_STRESS((J-1)*G_XY+I,K+3)
5067                        ENDDO
5068                    ENDDO
5069                    MOM(:,:,I)=MOM(:,:,I)+Z_W(J)*M_TEMP(:,:,I)
5070                ENDDO
5071             ENDDO
5072             MOM=MOM*H/2.0D0
5073             MOM_NET(:,:)=(MOM(:,:,1)+MOM(:,:,2)+MOM(:,:,3)+MOM(:,:,4))
5074          +                  /4.0D0
5075             IF(OUTPUT) THEN
5076             OPEN(824, FILE='MOMENTS.DAT')
5077             WRITE(824,*) 'ELEMENT',IEL
5078             DO I=1,G_XY
5079                WRITE(824,*) 'COORDINATES'
5080                WRITE(824,20) COORDG(I,1),COORDG(I,2)
5081                WRITE(824,*) 'MOMENTS'
5082                DO J=1,9
5083                    WRITE(824,20) (MOM(J,K,I),K=1,4)
5084                ENDDO
5085             ENDDO
5086             WRITE(824,*) 'NET MOMENT'
5087             DO I=1,9
5088                WRITE(824,20) (MOM_NET(I,J),J=1,4)
5089             ENDDO
5090             ENDIF
5091
5092
5093
5094
5095   !
5096   !          COMPONENTS MULTIPLIED BY DNXi
5097             INT_F1(1)=MOM_NET(1,1)
5098             INT_F1(2)=MOM_NET(1,2)
5099             INT_F1(3)=MOM_NET(1,3)
5100             INT_F1(4)=MOM_NET(1,4)
5101             INT_F1(5)=MOM_NET(6,1)
5102             INT_F1(6)=MOM_NET(6,2)
5103             INT_F1(7)=MOM_NET(6,3)
5104             INT_F1(8)=MOM_NET(6,4)
5105             INT_F1(9)=MOM_NET(5,1)
```

```
            INT_F1(10)=MOM_NET(5,2)
            INT_F1(11)=MOM_NET(5,3)
            INT_F1(12)=MOM_NET(5,4)
!           COMPONENTS MULTIPLIED BY DNYi
            INT_F2(1)=MOM_NET(9,1)
            INT_F2(2)=MOM_NET(9,2)
            INT_F2(3)=MOM_NET(9,3)
            INT_F2(4)=MOM_NET(9,4)
            INT_F2(5)=MOM_NET(2,1)
            INT_F2(6)=MOM_NET(2,2)
            INT_F2(7)=MOM_NET(2,3)
            INT_F2(8)=MOM_NET(2,4)
            INT_F2(9)=MOM_NET(4,1)
            INT_F2(10)=MOM_NET(4,2)
            INT_F2(11)=MOM_NET(4,3)
            INT_F2(12)=MOM_NET(4,4)
!           COMPONENTS MULTIPLIED BY Ni
            INT_F3(1)=0.0D0
            INT_F3(2)=MOM_NET(8,1)
            INT_F3(3)=2.0D0*MOM_NET(8,2)
            INT_F3(4)=3.0D0*MOM_NET(8,3)
            INT_F3(5)=0.0D0
            INT_F3(6)=MOM_NET(7,1)
            INT_F3(7)=2.0D0*MOM_NET(7,2)
            INT_F3(8)=3.0D0*MOM_NET(7,3)
            INT_F3(9)=0.0D0
            INT_F3(10)=MOM_NET(3,1)
            INT_F3(11)=2.0D0*MOM_NET(3,2)
            INT_F3(12)=3.0D0*MOM_NET(3,3)
c           write(974,*) iel
c           do i=1,12
c              write(974,20) int_f1(i),int_f2(i),int_f3(i)
c           enddo
            DO I = 1,G_X
               DO J=1,G_Y
                   CALL FEM_SHAPE(X_P(I),Y_P(J),I_ORDER
     +            ,N1,N2,N3,N4,DET_J,LE,BE)
                   CALL FEM_SHAPE_DIFF(X_P(J),Y_P(I),I_ORDER
     +             ,DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
                   DNX(1,1)=DN1X; DNX(2,1)=DN2X; DNX(3,1)=DN3X;
     +             DNX(4,1)=DN4X
                   DNY(1,1)=DN1Y; DNY(2,1)=DN2Y; DNY(3,1)=DN3Y;
     +             DNY(4,1)=DN4Y
                   N(1,1)=N1; N(2,1)=N2; N(3,1)=N3; N(4,1)=N4
                   DO K=1,4
                       FUNCT1(I,J,K)=DNX(K,1)
                   ENDDO
                   DO K=1,4
                       FUNCT2(I,J,K)=DNY(K,1)
                   ENDDO
                   DO K=1,4
```

```
                        FUNCT3(I,J,K)=N(K,1)
                    ENDDO
            ENDDO
          ENDDO



C       DO I=1,4
C          DO J=1,G_X
C          WRITE(904,20) (FUNCT1(J,K,I),K=1,G_Y)
C          ENDDO
C          WRITE(904,*) '-----'
C       ENDDO
```

```
          DO I=1,G_X
            W_X(1,I)=X_W(I)
          ENDDO
          DO I=1,G_Y
            W_Y(I,1)=X_W(I)
          ENDDO

! DEBUGGING
c       !!!write(51,20) W_X
c       !!!write(51,20) W_Y

          DO I=1,4
            M_FUN1=0.0D0
            M_FUN2=0.0D0
            M_FUN3=0.0D0
            DO J=1,G_X
                DO K=1,G_Y
                    M_FUN1(J,K)=FUNCT1(J,K,I)
                    M_FUN2(J,K)=FUNCT2(J,K,I)
                    M_FUN3(J,K)=FUNCT3(J,K,I)
                ENDDO
            ENDDO




c           !!!write(51,*) 'M_FUN'
c           !!!write(51,*) I
            DO L=1,G_X
c               !!!write(51,20) (M_FUN(L,Q), Q=1,G_Y)
            ENDDO
            V1=MATMUL(MATMUL(W_X,M_FUN1),W_Y)
            INTEGRAL1(I)=V1(1,1)
            V2=MATMUL(MATMUL(W_X,M_FUN2),W_Y)
            INTEGRAL2(I)=V2(1,1)
            V3=MATMUL(MATMUL(W_X,M_FUN3),W_Y)
```

```
5244            INTEGRAL3(I)=V3(1,1)
5245           ENDDO
5246
5247
5248
5249
5250
5251
5252           ! DEBUGGING
5253     c      !!!write(51,*) INTEGRAL
5254     c      !!!write(51,*) DET_J
5255           !
5256           DO I=1,4
5257             A1(I)=DET_J*INTEGRAL1(I)
5258             A2(I)=DET_J*INTEGRAL2(I)
5259             A3(I)=DET_J*INTEGRAL3(I)
5260           ENDDO
5261     c      WRITE(771,*) IEL
5262     c      WRITE(771,*) 'A1,A2,A3'
5263     c      WRITE(771,20) A1
5264     c      WRITE(771,20) A2
5265     c      WRITE(771,20) A3
5266
5267
5268
5269
5270
5271
5272           DO K=1,12
5273             DO I=1,4
5274              FEIN((K-1)*4+I)=INT_F1(K)*A1(I)+INT_F2(K)*A2(I)+INT_F3(K)
5275          +    *A3(I)
5276             ENDDO
5277           ENDDO
5278
5279
5280
5281
5282
5283
5284           DO I=1,E_DOF
5285                  FFIN(CON(IEL,I))=FFIN(CON(IEL,I))+FEIN(I)
5286           ENDDO
5287
5288
5289           IF (DEBUG) THEN
5290            WRITE(28,*) 'ELEMENT NUMBER',IEL
5291            DO I=1,E_DOF
5292             WRITE(28,21) FEIN(I)
5293            ENDDO
5294           ENDIF
5295
5296     C        CALL TRAC_VERI(N_FEM,T_P,IEL,LE,BE
5297     C      + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,
5298     C      +  TIME,H,I_ORDER,N_NODE,E_DOF,TIME_STEP,REM)
5299
5300
5301
5302           TEMP_U=0.0D0
5303           T_U=0.0D0
5304
5305           DO I=1,G_Z
5306                TEMP_U=1.0D0/2.0D0*(S_CENT(I,1)*STRAIN_OUT(I,4)+
5307          +     S_CENT(I,2)*STRAIN_OUT(I,5)+S_CENT(I,3)*STRAIN_OUT(I,6)+
5308          +     S_CENT(I,4)*STRAIN_OUT(I,7)+S_CENT(I,5)*STRAIN_OUT(I,8)+
5309          +     S_CENT(I,6)*STRAIN_OUT(I,9))
5310                T_U=T_U+Z_W(I)*TEMP_U
5311           ENDDO
5312
```

```
5313
5314
5315
5316
5317
5318
5319
5320          T_U=T_U*H/2.0D0*LE*BE
5321
5322          TU=TU+T_U
5323
5324
5325          ENDDO
5326
5327
5328
5329        ENDDO         !END OF IEL LOOP
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346        IF (TIME_STEP==1 .OR. REM==0) THEN
5347           DO I=1,NL2*NB2*NH2
5348               WRITE(917,23) (CON2(I,J),J=1,8)
5349               WRITE(918,23) (CON2(I,J),J=1,8)
5350               WRITE(919,23) (CON2(I,J),J=1,8)
5351           ENDDO
5352            DO I=1,NL2*NB2
5353               WRITE(557,23) (CON3(I,J), J=1,4)
5354            ENDDO
5355        ENDIF
5356
5357
5358
5359        FORMAT (9999(x,EN14.4))
5360        FORMAT(EN12.3)
5361        FORMAT(F5.0)
5362        FORMAT (999(2X,I5))
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379        END SUBROUTINE INT_FORCE
5380    !---------------------------------------------------------------------
5381    !
```

```
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!
!-----------------------------------------------------------------------

      SUBROUTINE NONLIN_STRAIN(ELE_STRESS,ELE_STRAIN,N_FEM,T_P,IEL,LE,BE
     + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,I_ORDER,LOCAL_C,
     + TIME,I_VON)
      USE PROG_DEBUG
      IMPLICIT NONE
      INTEGER :: N_FEM, T_P, I, J, K, IEL
      REAL(KIND=8) :: ELE_STRESS(T_P,24), ELE_STRAIN(T_P,9)
      REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,GN1,GN2,GN3,GN4
      REAL(KIND=8) :: N1,N2,N3,N4,TIME,
     +  DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J
!     VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
      REAL(KIND=8) :: B(9,48),NODAL_DISP(48,1),F_COL(9,1)
!     B IS THE OPERATOR MATRIX FOR CALCULATING F
!     NODAL DISP - DISP OF NODES IN THE CURENT ELEMENT
!     F_COL - ELEMENTS OF DEFORMATION GRADIENT 'F' IN A COLUMN FORM
      REAL(KIND=8) :: TEMP(6,1)
!     TEMPORARY ARRAY FOR COMPUTATIONS
      REAL(KIND=8) :: CC(6,6)
!     MATERIAL STIFFNESS MATRIX
      REAL(KIND=8) :: LOCAL_C(T_P,3)
!     LOCAL COORDINATE OF THE INTEGRATION POINTS
      INTEGER :: I_ORDER, I_VON, F_ORD
!     ORDER OF THE ELEMENTS (4 NODED/8 NODED)
!     F_ORD = 3 ORDER OF DEFORMATION GRAD
      REAL(KIND=8) :: DEF_GRAD(3,3),DISP_GRAD(3,3),F_INV(3,3)
!     DEFORMATION GRADIENT MATRIX (3X3)
!     F_INV - INVERSE OF DEF GRAD
c      REAL(KIND=8),ALLOCATABLE :: DISP_MAT(:,:)
      REAL(KIND=8) :: IDEN(3,3)   !IDENTITY MATRIX
      REAL(KIND=8) :: STRAIN_COL(6),E(3,3)
      REAL(KIND=8) :: JACOB   !JACOBIAN OF THE DEFORM. GRAD.
!     GREEN LAGRANGE STRAIN
      REAL(KIND=8) :: S(3,3),S_COL(6),T(3,3),T_COL(9),SIG(3,3),
     +  SIG_COL(9)
!     SECOND AND FIRST PIOLA KIRCHOFF STRESS RESPECTIVELY IN MATRIX AND
!     COLUMN FORM
C      REAL(KIND=8) :: TEMP_U(G_Z)  !VALUE OF STRAIN ENERGY IN IEL

c      ALLOCATE(DISP_MAT(48,3))
c      DISP_MAT(:,1)=NODAL_DISP(:,1)
c      DISP_MAT(:,2)=NODAL_DISP(:,1)
c      DISP_MAT(:,3)=NODAL_DISP(:,1)

      IDEN=0.0D0
      IDEN(1,1)=1.0D0; IDEN(2,2)=1.0D0; IDEN(3,3)=1.0D0
C      open(827,file='results_strain.dat')
C      write(827,*) '--------------------'
C      do j=1,48
C         write(827,21) nodal_disp(j,1)
C      enddo

      B=0.0D0   !INITIALIZING B
      DO I=1,T_P
       CALL FEM_SHAPE_DIFF(LOCAL_C(I,1),LOCAL_C(I,2),I_ORDER,
     + DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
       CALL FEM_SHAPE(LOCAL_C(I,1),LOCAL_C(I,2),I_ORDER
     +       ,N1,N2,N3,N4,DET_J,LE,BE)

c         dn1x=1; dn2x=1; dn3x=1; dn4x=1;
c         dn1y=2; dn2y=2; dn3y=2; dn4y=2;
c         n1=3; n2=3; n3=3; n4=3;
```

```
!         OPERATOR MATRIX
          DO J=1,4
            B(1,4*J)=DN4X*ELE_STRAIN(I,3)**(J-1)
            B(1,4*J-1)=DN3X*ELE_STRAIN(I,3)**(J-1)
            B(1,4*J-2)=DN2X*ELE_STRAIN(I,3)**(J-1)
            B(1,4*J-3)=DN1X*ELE_STRAIN(I,3)**(J-1)
            B(2,4*J+16)=DN4Y*ELE_STRAIN(I,3)**(J-1)
            B(2,4*J-1+16)=DN3Y*ELE_STRAIN(I,3)**(J-1)
            B(2,4*J-2+16)=DN2Y*ELE_STRAIN(I,3)**(J-1)
            B(2,4*J-3+16)=DN1Y*ELE_STRAIN(I,3)**(J-1)
!
            B(4,4*J+32)=DN4Y*ELE_STRAIN(I,3)**(J-1)
            B(4,4*J-1+32)=DN3Y*ELE_STRAIN(I,3)**(J-1)
            B(4,4*J-2+32)=DN2Y*ELE_STRAIN(I,3)**(J-1)
            B(4,4*J-3+32)=DN1Y*ELE_STRAIN(I,3)**(J-1)
!
            B(5,4*J+32)=DN4X*ELE_STRAIN(I,3)**(J-1)
            B(5,4*J-1+32)=DN3X*ELE_STRAIN(I,3)**(J-1)
            B(5,4*J-2+32)=DN2X*ELE_STRAIN(I,3)**(J-1)
            B(5,4*J-3+32)=DN1X*ELE_STRAIN(I,3)**(J-1)
!
            B(6,4*J)=DN4Y*ELE_STRAIN(I,3)**(J-1)
            B(6,4*J-1)=DN3Y*ELE_STRAIN(I,3)**(J-1)
            B(6,4*J-2)=DN2Y*ELE_STRAIN(I,3)**(J-1)
            B(6,4*J-3)=DN1Y*ELE_STRAIN(I,3)**(J-1)

            B(9,4*J+16)=DN4X*ELE_STRAIN(I,3)**(J-1)
            B(9,4*J-1+16)=DN3X*ELE_STRAIN(I,3)**(J-1)
            B(9,4*J-2+16)=DN2X*ELE_STRAIN(I,3)**(J-1)
            B(9,4*J-3+16)=DN1X*ELE_STRAIN(I,3)**(J-1)
          ENDDO
          DO J=2,4
            B(3,4*J+32)=(J-1)*N4*ELE_STRAIN(I,3)**(J-2)
            B(3,4*J-1+32)=(J-1)*N3*ELE_STRAIN(I,3)**(J-2)
            B(3,4*J-2+32)=(J-1)*N2*ELE_STRAIN(I,3)**(J-2)
            B(3,4*J-3+32)=(J-1)*N1*ELE_STRAIN(I,3)**(J-2)
!
            B(7,4*J+16)=(J-1)*N4*ELE_STRAIN(I,3)**(J-2)
            B(7,4*J-1+16)=(J-1)*N3*ELE_STRAIN(I,3)**(J-2)
            B(7,4*J-2+16)=(J-1)*N2*ELE_STRAIN(I,3)**(J-2)
            B(7,4*J-3+16)=(J-1)*N1*ELE_STRAIN(I,3)**(J-2)
!
            B(8,4*J)=(J-1)*N4*ELE_STRAIN(I,3)**(J-2)
            B(8,4*J-1)=(J-1)*N3*ELE_STRAIN(I,3)**(J-2)
            B(8,4*J-2)=(J-1)*N2*ELE_STRAIN(I,3)**(J-2)
            B(8,4*J-3)=(J-1)*N1*ELE_STRAIN(I,3)**(J-2)
          ENDDO
C          write(826,*) ele_strain(i,1),ele_strain(i,2),ele_strain(i,3),
C     +     local_c(i,1), local_c(i,2), i
C          do j=1,9
C              write(826,20) (B(j,k), k =1,48)
C          enddo

          F_COL=MATMUL(B,NODAL_DISP)
C          write(825,*) 'f_col'
C          write(825,20) ele_strain(i,1),ele_strain(i,2),ele_strain(i,3)
C     +     ,time
C          write(825,20) f_col


C         DISPLACEMENT GRADIENT - DUi/DXj

          DISP_GRAD(1,1)=F_COL(1,1)
          DISP_GRAD(2,2)=F_COL(2,1)
          DISP_GRAD(3,3)=F_COL(3,1)
          DISP_GRAD(3,2)=F_COL(4,1)
          DISP_GRAD(3,1)=F_COL(5,1)
          DISP_GRAD(1,2)=F_COL(6,1)
          DISP_GRAD(2,3)=F_COL(7,1)
```

```
5520            DISP_GRAD(1,3)=F_COL(8,1)
5521            DISP_GRAD(2,1)=F_COL(9,1)
5522
5523    c       DEFORMATION GRADIENT - DISPLACEMENT GRADIENT + I
5524
5525            DEF_GRAD=DISP_GRAD+IDEN
5526            JACOB=DEF_GRAD(1,1)*(DEF_GRAD(2,2)*DEF_GRAD(3,3)-
5527         +  DEF_GRAD(3,2)*DEF_GRAD(2,3))-
5528         +  DEF_GRAD(1,2)*(DEF_GRAD(2,1)*DEF_GRAD(3,3)-DEF_GRAD(3,1)
5529         +  *DEF_GRAD(2,3))+
5530         +  DEF_GRAD(1,3)*(DEF_GRAD(2,1)*DEF_GRAD(3,2)-DEF_GRAD(3,1)
5531         +  *DEF_GRAD(2,2))
5532
5533    !!!!!!!!DIFFERENT NONLINEARITIES
5534
5535            IF (I_VON==1) THEN !FULL NONLIN
5536
5537    !       GREEN ST. VENANT STRAIN
5538            E=1.0D0/2.0D0*((MATMUL(TRANSPOSE(DEF_GRAD),DEF_GRAD))-IDEN)
5539
5540    !
5541            STRAIN_COL(1)=E(1,1)
5542            STRAIN_COL(2)=E(2,2)
5543            STRAIN_COL(3)=E(3,3)
5544            STRAIN_COL(4)=2.0D0*E(2,3)
5545            STRAIN_COL(5)=2.0D0*E(1,3)
5546            STRAIN_COL(6)=2.0D0*E(1,2)
5547
5548    C        write(829,*) i
5549    C        do j=1,3
5550    C            write(829,20) (E(j,k) , k=1,3)
5551    C        enddo
5552
5553            S_COL=MATMUL(CC,STRAIN_COL)
5554
5555            S(1,1)=S_COL(1)
5556            S(2,2)=S_COL(2)
5557            S(3,3)=S_COL(3)
5558            S(1,2)=S_COL(6)
5559            S(1,3)=S_COL(5)
5560            S(2,3)=S_COL(4)
5561            S(2,1)=S(1,2)
5562            S(3,1)=S(1,3)
5563            S(3,2)=S(2,3)
5564
5565            T=MATMUL(DEF_GRAD,S)
5566    C        SIG=1.0D0/JACOB*MATMUL(T,TRANSPOSE(DEF_GRAD))
5567            T=TRANSPOSE(T)
5568            SIG=1.0D0/JACOB*MATMUL(DEF_GRAD,T)
5569
5570            IF (DEBUG) THEN
5571            OPEN(193,FILE='DEFORM_GRAD.DAT')
5572            OPEN(194,FILE='FIRST_PK.DAT')
5573            WRITE(193,*) 'COORDINATES'
5574            WRITE(193,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),ELE_STRAIN(I,3),
5575         +TIME
5576            WRITE(193,*) 'DEFORMATION GRADIENT'
5577            DO J=1,3
5578                WRITE(193,20) (DEF_GRAD(J,K),K=1,3)
5579            ENDDO
5580            WRITE(194,*) 'COORDINATES'
5581            WRITE(194,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),ELE_STRAIN(I,3),
5582         +TIME
5583            WRITE(194,*) 'STRESS'
5584            DO J=1,3
5585                WRITE(194,20) (SIG(J,K),K=1,3)
5586            ENDDO
5587            ENDIF
5588
```

```
5589
5590
5591            T_COL(1)=T(1,1)
5592            T_COL(2)=T(2,2)
5593            T_COL(3)=T(3,3)
5594            T_COL(4)=T(2,3)
5595            T_COL(5)=T(1,3)
5596            T_COL(6)=T(1,2)
5597            T_COL(7)=T(3,2)
5598            T_COL(8)=T(3,1)
5599            T_COL(9)=T(2,1)
5600
5601            SIG_COL(1)=SIG(1,1)
5602            SIG_COL(2)=SIG(2,2)
5603            SIG_COL(3)=SIG(3,3)
5604            SIG_COL(4)=SIG(2,3)
5605            SIG_COL(5)=SIG(1,3)
5606            SIG_COL(6)=SIG(1,2)
5607
5608
5609
5610
5611
5612            ELSEIF (I_VON==2) THEN !VON KARMAN NONLINEARITY WITH S=CE_vk
5613
5614                  E(1,1)=DISP_GRAD(1,1)+1.0D0/2.0D0*DISP_GRAD(3,1)**2.0D0
5615                  E(2,2)=DISP_GRAD(2,2)+1.0D0/2.0D0*DISP_GRAD(3,2)**2.0D0
5616                  E(3,3)=DISP_GRAD(3,3)
5617                  E(1,2)=1.0D0/2.0D0*(DISP_GRAD(1,2)+DISP_GRAD(2,1)+
5618           +                        DISP_GRAD(3,1)*DISP_GRAD(3,2))
5619                  E(2,1)=E(1,2)
5620                  E(2,3)=1.0D0/2.0D0*(DISP_GRAD(2,3)+DISP_GRAD(3,2))
5621                  E(3,2)=E(2,3)
5622                  E(3,1)=1.0D0/2.0D0*(DISP_GRAD(3,1)+DISP_GRAD(1,3))
5623                  E(1,3)=E(3,1)
5624   c           E=1.0D0/2.0D0*(DISP_GRAD+TRANSPOSE(DISP_GRAD))
5625
5626            STRAIN_COL(1)=E(1,1)
5627            STRAIN_COL(2)=E(2,2)
5628            STRAIN_COL(3)=E(3,3)
5629            STRAIN_COL(4)=2.0D0*E(2,3)
5630            STRAIN_COL(5)=2.0D0*E(1,3)
5631            STRAIN_COL(6)=2.0D0*E(1,2)
5632
5633   C           write(829,*) i
5634   C           do j=1,3
5635   C               write(829,20) (E(j,k) , k=1,3)
5636   C           enddo
5637
5638            S_COL=MATMUL(CC,STRAIN_COL)
5639
5640            S(1,1)=S_COL(1)
5641            S(2,2)=S_COL(2)
5642            S(3,3)=S_COL(3)
5643            S(1,2)=S_COL(6)
5644            S(1,3)=S_COL(5)
5645            S(2,3)=S_COL(4)
5646            S(2,1)=S(1,2)
5647            S(3,1)=S(1,3)
5648            S(3,2)=S(2,3)
5649
5650            T=MATMUL(DEF_GRAD,S)
5651   C           SIG=1.0D0/JACOB*MATMUL(T,TRANSPOSE(DEF_GRAD))
5652            T=TRANSPOSE(T)
5653            SIG=1.0D0/JACOB*MATMUL(DEF_GRAD,T)
5654
5655            IF (DEBUG) THEN
5656            OPEN(193,FILE='DEFORM_GRAD.DAT')
5657            OPEN(194,FILE='FIRST_PK.DAT')
```

```
5658            WRITE(193,*) 'COORDINATES'
5659            WRITE(193,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),ELE_STRAIN(I,3),
5660           +TIME
5661            WRITE(193,*) 'DEFORMATION GRADIENT'
5662            DO J=1,3
5663                WRITE(193,20) (DEF_GRAD(J,K),K=1,3)
5664            ENDDO
5665            WRITE(194,*) 'COORDINATES'
5666            WRITE(194,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),ELE_STRAIN(I,3),
5667           +TIME
5668            WRITE(194,*) 'STRESS'
5669            DO J=1,3
5670                WRITE(194,20) (SIG(J,K),K=1,3)
5671            ENDDO
5672            ENDIF
5673
5674
5675
5676            T_COL(1)=T(1,1)
5677            T_COL(2)=T(2,2)
5678            T_COL(3)=T(3,3)
5679            T_COL(4)=T(2,3)
5680            T_COL(5)=T(1,3)
5681            T_COL(6)=T(1,2)
5682            T_COL(7)=T(3,2)
5683            T_COL(8)=T(3,1)
5684            T_COL(9)=T(2,1)
5685
5686            SIG_COL(1)=SIG(1,1)
5687            SIG_COL(2)=SIG(2,2)
5688            SIG_COL(3)=SIG(3,3)
5689            SIG_COL(4)=SIG(2,3)
5690            SIG_COL(5)=SIG(1,3)
5691            SIG_COL(6)=SIG(1,2)
5692
5693            ELSEIF (I_VON==3) THEN
5694
5695                   E(1,1)=DISP_GRAD(1,1)+1.0D0/2.0D0*DISP_GRAD(3,1)**2.0D0
5696                   E(2,2)=DISP_GRAD(2,2)+1.0D0/2.0D0*DISP_GRAD(3,2)**2.0D0
5697                   E(3,3)=DISP_GRAD(3,3)
5698                   E(1,2)=1.0D0/2.0D0*(DISP_GRAD(1,2)+DISP_GRAD(2,1)+
5699           +                           DISP_GRAD(3,1)*DISP_GRAD(3,2))
5700                   E(2,1)=E(1,2)
5701                   E(2,3)=1.0D0/2.0D0*(DISP_GRAD(2,3)+DISP_GRAD(3,2))
5702                   E(3,2)=E(2,3)
5703                   E(3,1)=1.0D0/2.0D0*(DISP_GRAD(3,1)+DISP_GRAD(1,3))
5704                   E(1,3)=E(3,1)
5705
5706
5707            STRAIN_COL(1)=E(1,1)
5708            STRAIN_COL(2)=E(2,2)
5709            STRAIN_COL(3)=E(3,3)
5710            STRAIN_COL(4)=2.0D0*E(2,3)
5711            STRAIN_COL(5)=2.0D0*E(1,3)
5712            STRAIN_COL(6)=2.0D0*E(1,2)
5713
5714            S_COL=MATMUL(CC,STRAIN_COL)
5715
5716            S(1,1)=S_COL(1)
5717            S(2,2)=S_COL(2)
5718            S(3,3)=S_COL(3)
5719            S(1,2)=S_COL(6)
5720            S(1,3)=S_COL(5)
5721            S(2,3)=S_COL(4)
5722            S(2,1)=S(1,2)
5723            S(3,1)=S(1,3)
5724            S(3,2)=S(2,3)
5725
5726            T=S
```

```
5727            T_COL(1)=T(1,1)
5728            T_COL(2)=T(2,2)
5729            T_COL(3)=T(3,3)
5730            T_COL(4)=T(2,3)
5731            T_COL(5)=T(1,3)
5732            T_COL(6)=T(1,2)
5733            T_COL(7)=T(3,2)
5734            T_COL(8)=T(3,1)
5735            T_COL(9)=T(2,1)
5736
5737            SIG=T
5738
5739            SIG_COL(1)=SIG(1,1)
5740            SIG_COL(2)=SIG(2,2)
5741            SIG_COL(3)=SIG(3,3)
5742            SIG_COL(4)=SIG(2,3)
5743            SIG_COL(5)=SIG(1,3)
5744            SIG_COL(6)=SIG(1,2)
5745
5746            ELSEIF (I_VON==4) THEN
5747
5748            E=1.0D0/2.0D0*(DISP_GRAD+TRANSPOSE(DISP_GRAD))
5749
5750            STRAIN_COL(1)=E(1,1)
5751            STRAIN_COL(2)=E(2,2)
5752            STRAIN_COL(3)=E(3,3)
5753            STRAIN_COL(4)=2.0D0*E(2,3)
5754            STRAIN_COL(5)=2.0D0*E(1,3)
5755            STRAIN_COL(6)=2.0D0*E(1,2)
5756
5757            S_COL=MATMUL(CC,STRAIN_COL)
5758
5759            S(1,1)=S_COL(1)
5760            S(2,2)=S_COL(2)
5761            S(3,3)=S_COL(3)
5762            S(1,2)=S_COL(6)
5763            S(1,3)=S_COL(5)
5764            S(2,3)=S_COL(4)
5765            S(2,1)=S(1,2)
5766            S(3,1)=S(1,3)
5767            S(3,2)=S(2,3)
5768
5769            T=S
5770            T_COL(1)=T(1,1)
5771            T_COL(2)=T(2,2)
5772            T_COL(3)=T(3,3)
5773            T_COL(4)=T(2,3)
5774            T_COL(5)=T(1,3)
5775            T_COL(6)=T(1,2)
5776            T_COL(7)=T(3,2)
5777            T_COL(8)=T(3,1)
5778            T_COL(9)=T(2,1)
5779
5780            SIG=T
5781
5782            SIG_COL(1)=SIG(1,1)
5783            SIG_COL(2)=SIG(2,2)
5784            SIG_COL(3)=SIG(3,3)
5785            SIG_COL(4)=SIG(2,3)
5786            SIG_COL(5)=SIG(1,3)
5787            SIG_COL(6)=SIG(1,2)
5788
5789            ENDIF
5790
5791   !!!!!!!!!END OF NONLINEARITIES STATEMENT
5792
5793   !        INVERTING DEFORMATION GRADIENT
5794   c         F_INV=0.0D0
5795   c         F_ORD=3
```

```
c         CALL LU_INVERSE(F_ORD,DEF_GRAD,F_INV)






         DO J=1,6
             ELE_STRAIN(I,J+3)=STRAIN_COL(J)
         ENDDO
         DO J=1,9
             ELE_STRESS(I,J+3)=T_COL(J)
         ENDDO
         DO J=1,6
             ELE_STRESS(I,J+12)=SIG_COL(J)
         ENDDO
         DO J=1,6
             ELE_STRESS(I,J+18)=S_COL(J)
         ENDDO

       ENDDO



      FORMAT (9999(x,EN14.4))
      FORMAT(EN12.3)
      FORMAT(F5.0)
      FORMAT (999(2X,I5))

      END SUBROUTINE NONLIN_STRAIN



      SUBROUTINE FEM_SHAPE(X_P,Y_P,I_ORDER,N1,N2,N3,N4,DET_J,LE,BE)
      IMPLICIT NONE
!
      REAL(KIND=8) :: X_P,Y_P       !GAUSS POINTS COORDINATES
      INTEGER :: I_ORDER       !2 NODED OR 3 NODED ELEMENTS
      REAL(KIND=8) :: N1,N2,N3,N4    !VALUES OF THE SHAPE FUNCTIONS AT GAUSS POINTS
      REAL(KIND=8) :: LE       !LENGTH OF THE ELEMENT
      REAL(KIND=8) :: BE       !WIDTH OF THE ELEMENT
      REAL(KIND=8) :: J(2,2),DET_J   !ELEMENT JACOBIAN MATRIX AND DETERMINANT
!
!
      IF (I_ORDER==1) THEN
        N1=(1.0D0-X_P)*(1.0D0-Y_P)/4.0D0
        N2=(X_P+1.0D0)*(1.0D0-Y_P)/4.0D0
        N3=(X_P+1.0D0)*(Y_P+1.0D0)/4.0D0
        N4=(1.0D0-X_P)*(Y_P+1.0D0)/4.0D0
        J=0.0D0
        J(1,1)=LE/2.0D0; J(2,2)=BE/2.0D0
        DET_J=J(1,1)*J(2,2)-J(1,2)*J(2,1)
      ELSE
       WRITE(*,*) 'PROGRAM NOT READY YET'
      ENDIF
!
      END SUBROUTINE FEM_SHAPE

      SUBROUTINE FEM_SHAPE_DIFF(X_P,Y_P,I_ORDER,DN1X,DN2X,DN3X,DN4X,
     + DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
      IMPLICIT NONE
!
      REAL(KIND=8) :: X_P,Y_P       !GAUSS POINTS COORDINATES
      INTEGER :: I_ORDER       !2 NODED OR 3 NODED ELEMENTS
      REAL(KIND=8) :: DN1X,DN2X,DN3X,DN4X    !DERIVATIVES OF THE SHAPE FUNCTIONS WRT X
      REAL(KIND=8) :: DN1Y,DN2Y,DN3Y,DN4Y    !DERIVATIVES OF THE SHAPE FUNCTIONS WRT Y
      REAL(KIND=8) :: LE       !LENGTH OF THE ELEMENT
      REAL(KIND=8) :: BE       !WIDTH OF THE ELEMENT
      REAL(KIND=8) :: J(2,2),DET_J   !ELEMENT JACOBIAN MATRIX AND DETERMINANT
```

```
      REAL(KIND=8) :: DN1R,DN2R,DN3R,DN4R,DN1S,DN2S,DN3S,DN4S
      !DERIVATIVES OF MATRIX WRT R AND S RESP

      IF (I_ORDER==1) THEN
         DN1R=Y_P/4.0D0-1.0D0/4.0D0
         DN2R=-DN1R
         DN3R=Y_P/4.0D0+1.0D0/4.0D0
         DN4R=-DN3R
         DN1S=X_P/4.0D0-1.0D0/4.0D0
         DN2S=-X_P/4.0D0-1.0D0/4.0D0
         DN3S=-DN2S
         DN4S=-DN1S
         J=0.0D0
         J(1,1)=LE/2.0D0; J(2,2)=BE/2.0D0
         DET_J=J(1,1)*J(2,2)-J(1,2)*J(2,1)
      ELSE
         WRITE(*,*) 'PROGRAM NOT READY'
      ENDIF
         DN1X=1/DET_J*DN1R*J(2,2)
         DN2X=1/DET_J*DN2R*J(2,2)
         DN3X=1/DET_J*DN3R*J(2,2)
         DN4X=1/DET_J*DN4R*J(2,2)
         DN1Y=1/DET_J*DN1S*J(1,1)
         DN2Y=1/DET_J*DN2S*J(1,1)
         DN3Y=1/DET_J*DN3S*J(1,1)
         DN4Y=1/DET_J*DN4S*J(1,1)

      END SUBROUTINE FEM_SHAPE_DIFF

      SUBROUTINE FEM_SHAPE_GLOBAL(X_I,X_J,Y_I,Y_L,GLOB_X_P,GLOB_Y_P,
     + GN1,GN2,GN3,GN4,LE,BE)
      IMPLICIT NONE
      REAL(KIND=8) :: GLOB_X_P,GLOB_Y_P,X_I,X_J,Y_I,Y_L
      REAL(KIND=8) :: GN1,GN2,GN3,GN4
      REAL(KIND=8) :: LE,BE
!!
!!

      !write(117,*) 'NODAL cords from global'
      !write(117,*) x_i,x_j,y_i,y_l
      GN1=(1.0D0/2.0D0-GLOB_X_P/LE+(X_I+X_J)/2.0D0/LE)*
     + (1.0D0/2.0D0-GLOB_Y_P/BE+(Y_I+Y_L)/2.0D0/BE)
      GN2=(1.0D0/2.0D0+GLOB_X_P/LE-(X_I+X_J)/2.0D0/LE)*
     + (1.0D0/2.0D0-GLOB_Y_P/BE+(Y_I+Y_L)/2.0D0/BE)
      GN3=(1.0D0/2.0D0+GLOB_X_P/LE-(X_I+X_J)/2.0D0/LE)*
     + (1.0D0/2.0D0+GLOB_Y_P/BE-(Y_I+Y_L)/2.0D0/BE)
      GN4=(1.0D0/2.0D0-GLOB_X_P/LE+(X_I+X_J)/2.0D0/LE)*
     + (1.0D0/2.0D0+GLOB_Y_P/BE-(Y_I+Y_L)/2.0D0/BE)
!
      END SUBROUTINE FEM_SHAPE_GLOBAL




      SUBROUTINE LU_INVERSE(N,A,AINV)

C---------------------
C  CROUT DECOMPOSITION
C
C  ALGORITHM 2.6.2
C---------------------

      IMPLICIT NONE
C   REAL (KIND=8) ::       AINV(:,:)
      REAL(KIND=8),ALLOCATABLE :: L(:,:),LINV(:,:),U(:,:),UINV(:,:)
      REAL(KIND=8),ALLOCATABLE :: DIAG(:,:)

      INTEGER:: N
      REAL(KIND=8),INTENT(IN) :: A(N,N)
```

```
      REAL(KIND=8),INTENT(OUT) :: AINV(N,N)
      INTEGER I,J,K,M,I1,I2,I3,I4
      REAL(KIND=8) :: SUM,ERR,ERRMAX
      ALLOCATE(L(N,N),LINV(N,N),U(N,N),UINV(N,N))
      ALLOCATE(DIAG(N,N))

C       WRITE(*,*) 'INVERSE START'

    ! LU DECOMPOSITION

C--------------------------
C FIRST ROW AND FIRST COLUMN
C--------------------------
      AINV=0.0D0

      DO I=1,N
        U(I,I) = 1.0D0
        L(I,1) = A(I,1)
        U(1,I) = A(1,I)/L(1,1)
      END DO

C-------
C OTHERS
C-------

      DO K=2,N
        DO I=K,N
         SUM=0.0D0
         DO M=1,K-1
           SUM = SUM + L(I,M)*U(M,K)
         END DO
         L(I,K) = A(I,K) - SUM
        END DO
!
        DO J=K+1,N
          SUM = 0.0D0
          DO M=1,K-1
            SUM = SUM + L(K,M)*U(M,J)
          END DO
          U(K,J) = (A(K,J)-SUM)/L(K,K)
        END DO
      END DO
!
!
!------------------------ INVERSE LU
      CALL INV_L (N,L,LINV)
      CALL INV_U (N,U,UINV)
 !------------------------INVERSE OF A
      DO I=1,N
        DO J=1,N
          AINV(I,J) = 0.0D0
           K = I
           IF(J.LT.K) K = J
            DO M=K,N
              AINV(I,J) = AINV(I,J)+UINV(I,M)*LINV(M,J)
            END DO
        END DO
      END DO
!
C   DIAG=MATMUL(A,AINV)
      DO I1=1,N
        DIAG(I1,I1)=DOT_PRODUCT(A(I1,:),AINV(:,I1))
      ENDDO

      ERR=0.0D0; ERRMAX=0.0D0
      DO I=1,N
          ERR=ABS(DIAG(I,I)-1.0D0)
          IF (ERR>= ERRMAX) THEN
            ERRMAX=ERR
```

```
            ENDIF
          ENDDO

C         WRITE(*,*) 'ERRMAX=',ERRMAX




C DETERMINANT OF A
C-----------------

C       DET = 1.0D0

C       DO I=1,N
C        DET = DET*L(I,I)
C       END DO

C       WRITE (6,101) DET


C 101    FORMAT (/," DETERMINANT OF A :",F15.10,/)

C-----
C DONE
C-----

        RETURN
        END SUBROUTINE LU_INVERSE




        SUBROUTINE INV_L (N,A,B)


C-----------------------------------------
C INVERSE OF NXN LOWER TRIANGULAR MATRIX: A
C INVERSE IS MATRIX: B
C-----------------------------------------

        IMPLICIT NONE

          INTEGER :: N
          REAL(KIND=8),INTENT(IN) :: A(N,N)
          REAL(KIND=8),INTENT(OUT) :: B(N,N)
          INTEGER :: I,J,M
          REAL(KIND=8) :: SUM


C-------
C LAUNCH
C-------

        DO I=1,N
         B(I,I) = 1.0D0/A(I,I)
        END DO

        DO J=1,N-1
          DO I=J+1,N
            SUM = 0.0D0
            DO M=J,I-1
              SUM = SUM +A(I,M)*B(M,J)
            END DO
            B(I,J) = - SUM/A(I,I)
          END DO
        END DO
```

```
C-----
C DONE
C-----

      RETURN
      END SUBROUTINE INV_L


      SUBROUTINE INV_U (N,A,B)


C-----------------------------------------------
C COMPUTATION OF THE INVERSE OF NXN
C UPPER TRIANGULAR MATRIX: A
C
C INVERSE IS MATRIX: B
C
C ALGORITHM (2.5.8)
C-----------------------------------------------

      IMPLICIT NONE

      INTEGER:: N
      REAL(KIND=8),INTENT(IN) :: A(N,N)
      REAL(KIND=8),INTENT(OUT) :: B(N,N)
      INTEGER :: I,J,M
      REAL(KIND=8) :: SUM
C-------
C LAUNCH
C-------

      DO I=1,N
       B(I,I) = 1.0D0/A(I,I)
      END DO

      DO J=2,N
        DO I=J-1,1,-1
          SUM = 0.0D0
          DO M=I+1,J
            SUM = SUM + A(I,M)*B(M,J)
          END DO
          B(I,J) = - SUM/A(I,I)
        END DO
      END DO

C-----
C DONE
C-----

      RETURN
      END SUBROUTINE INV_U


      SUBROUTINE PRESS_LOAD(E_DOF,TOT_DOF,I_ORDER,H,LE,BE,CON,
     +   SOL,FFN1,APP_LOAD,N_FEM,E_NODE,N_NODE,N_L,N_B,I_LOAD,XE,YE
     +  ,LENGTH,BREADTH,K,T0,DT,THETA,G_X,G_Y)

      USE PROG_DEBUG     !USING THE MODULE PROG_DEBUG FOR LOG. OPERATORS
      IMPLICIT NONE

      INTEGER :: TOT_DOF,N_FEM,N_NODE,NODE_L,NODE_B,I_ORDER,I_LOAD
!     TOTAL DOF, NO OF ELE.,NO OF NODES,NODES IN LEN.,NODES IN BREADTH
      INTEGER :: IEL,I,J,K,L,N,G_X,G_Y,E_DOF,E_NODE,N_L,N_B
!     ELE #, LOOP COUNTERS, GAUSS POINTS IN X AND Y
      REAL(KIND=8) :: T,T0,DT,APP_LOAD,H
      REAL(KIND=8) :: XE(N_FEM,E_NODE),YE(N_FEM,E_NODE)
      REAL(KIND=8) :: X_I,X_J,X_K,X_L   !X COR. OF NODES OF CURRENT ELE
      REAL(KIND=8) :: Y_I,Y_J,Y_K,Y_L   !Y COR. OF NODES OF CURRENT ELE
```

```
      REAL(KIND=8) :: X_CENT,Y_CENT,Z_CENT !CENTROID OF ELEMENT AT TOP
      REAL(KIND=8) :: N1,N2,N3,N4,TIME,
     +  DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J
!     VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
      REAL(KIND=8) :: B(9,48),NODAL_DISP(48,1),F_COL(9,1)
!     B IS THE OPERATOR MATRIX
!     NODAL DISP GIVES THE NODAL DISPLACEMENTS
!     F_COL COMPONENTS OF DEFORMATION GRADIENT IN COLUMN FORM
      REAL(KIND=8) :: SOL(TOT_DOF)
      INTEGER :: CON(N_FEM,E_DOF)   !CONNECTIVIY
      REAL(KIND=8) :: FFN1(TOT_DOF)  !THE OUTPUT LOAD VECTOR AT T(N+1)
      REAL(KIND=8) :: FE(E_DOF)
      REAL(KIND=8) :: P,PP  !P IS PEAK LOAD, PP IS LOAD AT CENTROIDS
      REAL(KIND=8) :: RADI !RADIUS OF THE LOAD KERNEL FOR SPATIAL LOAD
      REAL(KIND=8) :: LENGTH, BREADTH
      REAL(KIND=8) :: DEF_GRAD(3,3), F_INV(3,3), IDEN(3,3)
      INTEGER :: F_ORD
      REAL(KIND=8) :: JACOB !DETERMINANT OF DEF_GRAD
      REAL(KIND=8) :: S13_T,S13_B,S23_T,S23_B,S33_T,S33_B
!     SURFACE TRACTIONS AT '_T' - TOP, '_B' - BOTTOM
      REAL(KIND=8) :: THETA   !DECAY CONSTANT
      REAL(KIND=8) :: X_LOC,Y_LOC   !LOCAL COOR. OF CENTROIDS


!     GET THE COORDINATES OF THE CENTER OF THE ELEMENT OF TOP
!     FIND DEFORMATION GRADIENTS OF THE TOP POINTS
!     FIND JACOBIAN AND INVERSE OF F
!     CALCULATE T13, T23, T33
!     SEND THESE VALUES TO LOAD VEC SUBROUTINE TO COMPUTE FF


      Z_CENT=H/2.0D0              !Z COORDINATE OF THE POINT ON THE TOP
      N=K-1                       ! TIME STEP NUMBER FROM T=0
      TIME=T0+N*DT                   ! CURRENT TIME

      IDEN=0.0D0                  !IDENTITY MATRIX
      IDEN(1,1)=1.0D0; IDEN(2,2)=1.0D0; IDEN(3,3)=1.0D0
      X_LOC=0.0D0
      Y_LOC=0.0D0

!     GET THE VALUE OF PRESSURE AT THE CURRENT TIME
      CALL TIME_LOAD(TIME,P,THETA,APP_LOAD)


      OPEN(962,FILE='LOAD_PROF.DAT')
      WRITE(962,*) K,TIME,P

      FFN1=0.0D0

c      open(926,file='sol.dat')
c             do i=1,tot_dof
c                write(926,23) sol(i)
c             enddo

!     RUN THE ELEMENT LOOP
      DO IEL=1,N_FEM
!        COLLECTING NODAL DOF'S FOR EACH ELEMENT
         DO J=1,E_DOF
             NODAL_DISP(J,1)=SOL(CON(IEL,J))
         ENDDO
!        NODAL COORDINATES OF THE CURRENT ELEMENT (4 VERTICES)
         X_I=XE(IEL,1);X_J=XE(IEL,2);X_K=XE(IEL,3);X_L=XE(IEL,4)
         Y_I=YE(IEL,1);Y_J=YE(IEL,2);Y_K=YE(IEL,3);Y_L=YE(IEL,4)
         if(debug) then
             write(51,*) 'global nodal coordinates'
             write(51,*) 'element number', iel
             write(51,20) x_i,y_i
             write(51,20) x_j,y_j
             write(51,20) x_k,y_k
```

```
6210                write(51,20) x_l,y_l
6211              endif
6212  !           CENTROIDAL COORDINATES
6213              X_CENT=(X_I+X_J+X_K+X_L)/4.0D0
6214              Y_CENT=(Y_I+Y_J+Y_K+Y_L)/4.0D0
6215  !           GLOBAL AND LOCAL COORDINATES OF THE GAUSS POINTS
6216  c            if (debug) then
6217  c                open(923,file='nodal_disp.dat')
6218  c                write(923,20) nodal_disp
6219  c            endif
6220
6221           IF (I_LOAD==1) THEN              !UNIFORM LOAD
6222                PP=P                        !APPLIED PRESSURE
6223           ELSEIF (I_LOAD==2) THEN          !SPATIALLY DEPENDENT LOAD
6224                RADI=SQRT((X_CENT-LENGTH/2.0D0)**2.0D0+
6225          +     (Y_CENT-BREADTH/2.0D0)**2.0D0)
6226              IF (RADI+SQRT(LE**2.0D0+BE**2.0D0)<LENGTH/2.0D0) THEN
6227              RADI=RADI/10.0D0
6228              PP=(-0.0005D0*RADI**4+0.01D0*RADI**3-0.0586D0*RADI**2-
6229          +   0.001D0*RADI+1.0D0)*P        !APP. PRES. IN SPAT. DEP. LOAD
6230              ELSE
6231              PP=0.0D0
6232              ENDIF
6233           ENDIF
6234
6235
6236           CALL FEM_SHAPE_DIFF(X_LOC,Y_LOC,I_ORDER,
6237          +  DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
6238           CALL FEM_SHAPE(X_LOC,Y_LOC,I_ORDER
6239          +       ,N1,N2,N3,N4,DET_J,LE,BE)
6240
6241  c            dn1x=1; dn2x=1; dn3x=1; dn4x=1;
6242  c            dn1y=2; dn2y=2; dn3y=2; dn4y=2;
6243  c            n1=3; n2=3; n3=3; n4=3;
6244  !           OPERATOR MATRIX
6245              DO J=1,4
6246                B(1,4*J)=DN4X*Z_CENT**(J-1)
6247                B(1,4*J-1)=DN3X*Z_CENT**(J-1)
6248                B(1,4*J-2)=DN2X*Z_CENT**(J-1)
6249                B(1,4*J-3)=DN1X*Z_CENT**(J-1)
6250                B(2,4*J+16)=DN4Y*Z_CENT**(J-1)
6251                B(2,4*J-1+16)=DN3Y*Z_CENT**(J-1)
6252                B(2,4*J-2+16)=DN2Y*Z_CENT**(J-1)
6253                B(2,4*J-3+16)=DN1Y*Z_CENT**(J-1)
6254  !
6255                B(4,4*J+32)=DN4Y*Z_CENT**(J-1)
6256                B(4,4*J-1+32)=DN3Y*Z_CENT**(J-1)
6257                B(4,4*J-2+32)=DN2Y*Z_CENT**(J-1)
6258                B(4,4*J-3+32)=DN1Y*Z_CENT**(J-1)
6259  !
6260                B(5,4*J+32)=DN4X*Z_CENT**(J-1)
6261                B(5,4*J-1+32)=DN3X*Z_CENT**(J-1)
6262                B(5,4*J-2+32)=DN2X*Z_CENT**(J-1)
6263                B(5,4*J-3+32)=DN1X*Z_CENT**(J-1)
6264  !
6265                B(6,4*J)=DN4Y*Z_CENT**(J-1)
6266                B(6,4*J-1)=DN3Y*Z_CENT**(J-1)
6267                B(6,4*J-2)=DN2Y*Z_CENT**(J-1)
6268                B(6,4*J-3)=DN1Y*Z_CENT**(J-1)
6269
6270                B(9,4*J+16)=DN4X*Z_CENT**(J-1)
6271                B(9,4*J-1+16)=DN3X*Z_CENT**(J-1)
6272                B(9,4*J-2+16)=DN2X*Z_CENT**(J-1)
6273                B(9,4*J-3+16)=DN1X*Z_CENT**(J-1)
6274              ENDDO
6275              DO J=2,4
6276                B(3,4*J+32)=(J-1)*N4*Z_CENT**(J-2)
6277                B(3,4*J-1+32)=(J-1)*N3*Z_CENT**(J-2)
6278                B(3,4*J-2+32)=(J-1)*N2*Z_CENT**(J-2)
```

```
            B(3,4*J-3+32)=(J-1)*N1*Z_CENT**(J-2)
!
            B(7,4*J+16)=(J-1)*N4*Z_CENT**(J-2)
            B(7,4*J-1+16)=(J-1)*N3*Z_CENT**(J-2)
            B(7,4*J-2+16)=(J-1)*N2*Z_CENT**(J-2)
            B(7,4*J-3+16)=(J-1)*N1*Z_CENT**(J-2)
!
            B(8,4*J)=(J-1)*N4*Z_CENT**(J-2)
            B(8,4*J-1)=(J-1)*N3*Z_CENT**(J-2)
            B(8,4*J-2)=(J-1)*N2*Z_CENT**(J-2)
            B(8,4*J-3)=(J-1)*N1*Z_CENT**(J-2)
          ENDDO



!         DISP. GRADIENT IN A COLUMN FORMAT
          F_COL=MATMUL(B,NODAL_DISP)
C          write(825,*) 'f_col'


!         DISP GRADIENT MATRIX
          DEF_GRAD(1,1)=F_COL(1,1)
          DEF_GRAD(2,2)=F_COL(2,1)
          DEF_GRAD(3,3)=F_COL(3,1)
          DEF_GRAD(3,2)=F_COL(4,1)
          DEF_GRAD(3,1)=F_COL(5,1)
          DEF_GRAD(1,2)=F_COL(6,1)
          DEF_GRAD(2,3)=F_COL(7,1)
          DEF_GRAD(1,3)=F_COL(8,1)
          DEF_GRAD(2,1)=F_COL(9,1)

!         DEFORMATION GRADIENT MATRIX = I + H
          DEF_GRAD=DEF_GRAD+IDEN
!         JACOBIAN
          JACOB=DEF_GRAD(1,1)*(DEF_GRAD(2,2)*DEF_GRAD(3,3)-
     +    DEF_GRAD(3,2)*DEF_GRAD(2,3))-
     +    DEF_GRAD(1,2)*(DEF_GRAD(2,1)*DEF_GRAD(3,3)-DEF_GRAD(3,1)
     +    *DEF_GRAD(2,3))+
     +    DEF_GRAD(1,3)*(DEF_GRAD(2,1)*DEF_GRAD(3,2)-DEF_GRAD(3,1)
     +    *DEF_GRAD(2,2))


!         INVERTING DEFORMATION GRADIENT
          F_INV=0.0D0
          F_ORD=3
          CALL LU_INVERSE(F_ORD,DEF_GRAD,F_INV)

         S13_T=PP*JACOB*F_INV(1,3)
         S23_T=PP*JACOB*F_INV(2,3)
         S33_T=PP*JACOB*F_INV(3,3)
         S13_B=0.0D0
         S23_B=0.0D0
         S33_B=0.0D0

c         open(123,file='def_grad.DAT')
c         write(123,20) x_cent,y_cent,z_cent,time
c         do j=1,3
c                write(123,20) (def_grad(j,l),l=1,3)
c         enddo
c         open(174,file='tractions.dat')
c         write(174,20) time,s13_t,s23_t,s33_t


         CALL LOAD_CALC(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,LE,BE,FE,
     + N_FEM,E_NODE,N_NODE,S13_T,S13_B,S23_T,S23_B,S33_T,S33_B)


         DO I=1,E_DOF
              FFN1(CON(IEL,I))=FFN1(CON(IEL,I))+FE(I)
```

```
          ENDDO

          ENDDO    !END OF IEL LOOP


        FORMAT (999(2x,EN14.4))
        FORMAT (999(2X,I3))
        FORMAT(EN12.3)


          END SUBROUTINE PRESS_LOAD

!--------------------------END-------------------------------------------
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
!             SUBROUTINE FOR TIME DEPENDENCY OF LOAD
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------
!-------------------------------------------------------------------------

        SUBROUTINE TIME_LOAD(T,P,THETA,APP_LOAD)
        USE PROG_DEBUG
        IMPLICIT NONE
        REAL(KIND=8) :: THETA,T,P,APP_LOAD
        REAL(KIND=8) :: COEFF
        LOGICAL :: TRIAN= .FALSE.   !TRIANGULAR PULSE LOAD
C        N=I-1
C        T=T0+N*DT
        IF (TRIAN) THEN
           IF (T<=200) THEN
               COEFF=T/200.0d0
               P=APP_LOAD*COEFF
           ELSEIF (200<T .AND. T<=400) THEN
               COEFF=-1/200.0D0*(T-200.0D0)+1.0D0
               P=APP_LOAD*COEFF
           ELSE
               P=0.0D0
           ENDIF
        ELSE
           IF (T<=100) THEN
              COEFF=T/100.0D0
              P=COEFF*APP_LOAD
           ELSE
              COEFF=EXP(-(T-100)/THETA)
              P=COEFF*APP_LOAD
          ENDIF
        ENDIF

        FORMAT (9999(x,EN14.4))
        FORMAT(EN12.3)
        END SUBROUTINE TIME_LOAD


!-------------------------------------------------------------------------
```

```
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!
!            SUBROUTINE FOR CALCULATING ELEMENTAL LOAD VECTOR
!
!
!-----------------------------------------------------------------------
!-----------------------------------------------------------------------
!
      SUBROUTINE LOAD_CALC(E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,H,LE,BE,FE,
     + N_FEM,E_NODE,N_NODE,S13_T,S13_B,S23_T,S23_B,S33_T,S33_B)

      USE PROG_DEBUG
      IMPLICIT NONE
!-----------------------------------------------------------------------
      INTEGER :: I,J,K,L,Q     !LOOP COUNTERS
      INTEGER :: E_DOF,TOT_DOF,I_ORDER,G_X,G_Y,N_FEM,E_NODE
     + ,N_NODE
!-----E_DOF - ELEMENTAL DOF, TOT_DOF - TOTAL DOF, I_ORDER - SHAPE FUNC.
!-----G_X - NO. OF INT. POINTS, N_FEM - NUMBER OF ELE, E_NODE - NO. OF
!-----NODES PER ELEMENT
      REAL(KIND=8) :: H,LE,BE
!-----HEIGHT - H, LE - LENGTH OF THE ELEMENTS
      REAL(KIND=8) :: FE(E_DOF)
!     FE IS ELEMENTAL LOAD VECTOR
      REAL(KIND=8),ALLOCATABLE :: X_P(:),X_W(:),Y_P(:),Y_W(:)
!-    GAUSS POINTS AND WEIGHTS IN X AND Y
      REAL(KIND=8) :: N1,N2,N3,N4,DET_J
!-----N1, N2, N3 - VALUES OF THE SHAPE FUNCTIONS AT THE GAUSS POINTS
!-----TEMPORARY VECTORS FOR CALCULATING THE LOAD VECTOR
      REAL(KIND=8) :: S13_T,S13_B,S23_T,S23_B,S33_T,S33_B
!-----BOUNDARY TRACTIONS
      REAL(KIND=8) :: T10,T11,T12,T13,T20,T21,T22,T23,T30,T31,T32,T33
      REAL(KIND=8) :: FUNCT(G_X,G_Y,4),N(4,1)
      REAL(KIND=8) :: T(12)
      REAL(KIND=8) :: W_X(1,G_X),W_Y(G_Y,1)
      REAL(KIND=8) :: M_FUN(G_X,G_Y)
      REAL(KIND=8) :: V(1,1)
      REAL(KIND=8) :: INTEGRAL(4),A(4)
      REAL(KIND=8) :: PI      !VALUE OF PI

      PI=4.0D0*ATAN(1.0D0)

!-----------------------------------------------------------------------
      ALLOCATE (X_P(G_X),X_W(G_X))
      ALLOCATE (Y_P(G_Y),Y_W(G_Y))

!
      CALL GAUSS_LEG(G_X,X_P,X_W)
      CALL GAUSS_LEG(G_Y,Y_P,Y_W)
      FE=0.0D0
!
!
!     SPATIAL DEPENDENCY OF LOAD
!
!
!     VALUES OF BOUNDARY TRACTIONS
C      S13_T=0.0D0;S13_B=0.0D0;S33_T=APP_LOAD;S33_B=0.0D0
C      S23_T=0.0D0; S23_B=0.0D0
c
c      s13_t=app_load;s13_b=app_load/2.0d0;s33_t=app_load
c      s33_b=app_load/2.0d0;s23_t=app_load; s23_b=app_load/2.0d0
c
      T10=S13_T-S13_B; T11=H/2.0D0*(S13_T+S13_B)
      T12=H**2/4.0D0*(S13_T-S13_B); T13=H**3/8.0D0*(S13_T+S13_B)
      T20=S23_T-S23_B; T21=H/2.0D0*(S23_T+S23_B)
      T22=H**2/4.0D0*(S23_T-S23_B); T23=H**3/8.0D0*(S23_T+S23_B)
      T30=S33_T-S33_B; T31=H/2.0D0*(S33_T+S33_B)
      T32=H**2/4.0D0*(S33_T-S33_B); T33=H**3/8.0D0*(S33_T+S33_B)
```

```
6486          T(1)=T10; T(2)=T11;T(3)=T12;T(4)=T13
6487          T(5)=T20; T(6)=T21;T(7)=T22;T(8)=T23
6488          T(9)=T30; T(10)=T31;T(11)=T32;T(12)=T33
6489    c      !!!write(51,*) 'LOAD VECTOR SUBROUTINE'
6490    c      !!!write(51,*) APP_LOAD
6491    !-----INTEGRATING FOR THE LOCAL LOAD VECTOR
6492          DO I = 1,G_X
6493             DO J=1,G_Y
6494                  CALL FEM_SHAPE(X_P(I),Y_P(J),I_ORDER
6495         +        ,N1,N2,N3,N4,DET_J,LE,BE)
6496                  N(1,1)=N1; N(2,1)=N2; N(3,1)=N3; N(4,1)=N4
6497                  DO K=1,4
6498                     FUNCT(I,J,K)=N(K,1)
6499                  ENDDO
6500             ENDDO
6501          ENDDO
6502    c      DO K=1,4
6503    c         DO I=1,G_X
6504    c             write(51,20) (FUNCT(I,J,K), J=1,G_Y)
6505    c         ENDDO
6506    c      ENDDO
6507
6508          DO I=1,G_X
6509             W_X(1,I)=X_W(I)
6510          ENDDO
6511          DO I=1,G_Y
6512             W_Y(I,1)=X_W(I)
6513          ENDDO
6514    ! DEBUGGING
6515    c      !!!write(51,20) W_X
6516    c      !!!write(51,20) W_Y
6517     !
6518          DO I=1,4
6519             M_FUN=0.0D0
6520             DO J=1,G_X
6521                 DO K=1,G_Y
6522                     M_FUN(J,K)=FUNCT(J,K,I)
6523                 ENDDO
6524             ENDDO
6525    c         !!!write(51,*) 'M_FUN'
6526    c         !!!write(51,*) I
6527             DO L=1,G_X
6528    c             !!!write(51,20) (M_FUN(L,Q), Q=1,G_Y)
6529             ENDDO
6530             V=MATMUL(MATMUL(W_X,M_FUN),W_Y)
6531             INTEGRAL(I)=V(1,1)
6532          ENDDO
6533          ! DEBUGGING
6534    c      !!!write(51,*) INTEGRAL
6535    c      !!!write(51,*) DET_J
6536          !
6537          DO I=1,4
6538             A(I)=DET_J*INTEGRAL(I)
6539          ENDDO
6540          !DEBUGGING
6541    c      !!!write(51,20) A
6542          !
6543
6544          DO K=1,12
6545             DO I=1,4
6546                 FE((K-1)*4+I)=T(K)*A(I)
6547             ENDDO
6548          ENDDO
6549
6550    !---------------------------------------------------------------------
6551    !      OUTPUT IN DUMP.DAT
6552    !      IF(DEBUG) THEN
6553    c         !!!write(51,*) 'FF_1 IN LOAD VECTOR CALC'
6554    c            DO K=1,TOT_DOF
```

```
c               !!!write(51,*) FF_1(K)
c             ENDDO
!       ENDIF
C       NODE_L=N_L+1
C       NODE_B=N_B+1


        if (debug) then
        open(57,file='ele_load.dat')
           do i=1,e_dof
               write(57,23) fe(i)
           enddo
        endif

        FORMAT (999(2x,EN14.4))
        FORMAT (999(2X,I3))
        FORMAT(EN12.3)
        END SUBROUTINE LOAD_CALC


        SUBROUTINE TRAC_VERI(N_FEM,T_P,IEL,LE,BE
       + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,NODAL_DISP,CC,
       +  TIME,H,I_ORDER,N_NODE,E_DOF,TIME_STEP,REM)


        USE PROG_DEBUG
        IMPLICIT NONE
        INTEGER :: N_FEM, T_P, I, J, K, IEL,N_NODE,E_DOF
        REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,GN1,GN2,GN3,GN4
        REAL(KIND=8) :: N1,N2,N3,N4,TIME,
       +  DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J,H
!       VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
        REAL(KIND=8) :: B(9,48),NODAL_DISP(48,1),F_COL(9,1)
!       B IS THE OPERATOR MATRIX FOR CALCULATING F
!       NODAL DISP - DISP OF NODES IN THE CURENT ELEMENT
!       F_COL - ELEMENTS OF DEFORMATION GRADIENT 'F' IN A COLUMN FORM
        REAL(KIND=8) :: TEMP(6,1)
!       TEMPORARY ARRAY FOR COMPUTATIONS
        REAL(KIND=8) :: CC(6,6)
!       MATERIAL STIFFNESS MATRIX
        REAL(KIND=8) :: X_LOC,Y_LOC,X_CENT,Y_CENT,Z_CENT
!       LOCAL COORDINATE OF THE INTEGRATION POINTS
        INTEGER :: F_ORD, I_ORDER
!       ORDER OF THE ELEMENTS (4 NODED/8 NODED)
!       F_ORD = 3 ORDER OF DEFORMATION GRAD
        REAL(KIND=8) :: DEF_GRAD(3,3),DISP_GRAD(3,3),F_INV(3,3)
!       DEFORMATION GRADIENT MATRIX (3X3)
!       F_INV - INVERSE OF DEF GRAD
c        REAL(KIND=8),ALLOCATABLE :: DISP_MAT(:,:)
        REAL(KIND=8) :: IDEN(3,3)   !IDENTITY MATRIX
        REAL(KIND=8) :: STRAIN_COL(6),E(3,3)
        REAL(KIND=8) :: JACOB   !JACOBIAN OF THE DEFORM. GRAD.
!       GREEN LAGRANGE STRAIN
        REAL(KIND=8) :: S(3,3),S_COL(6),T(3,3),T_COL(9),SIG(3,3),
       +  SIG_COL(9),TRAC(3,1),MAG(1,1)
        REAL(KIND=8) :: N(3,1)   !NORMAL VECTOR IN CURRENT CONFIG
        INTEGER :: TIME_STEP,REM


!       CENTROIDAL COORDINATES
        X_CENT=(X_I+X_J+X_K+X_L)/4.0D0
        Y_CENT=(Y_I+Y_J+Y_K+Y_L)/4.0D0
        Z_CENT=H/2.0D0


        IDEN=0.0D0                  !IDENTITY MATRIX
        IDEN(1,1)=1.0D0; IDEN(2,2)=1.0D0; IDEN(3,3)=1.0D0
```

```
6624          X_LOC=0.0D0
6625          Y_LOC=0.0D0
6626
6627
6628          CALL FEM_SHAPE_DIFF(X_LOC,Y_LOC,I_ORDER,
6629         +  DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
6630          CALL FEM_SHAPE(X_LOC,Y_LOC,I_ORDER
6631         +       ,N1,N2,N3,N4,DET_J,LE,BE)
6632
6633   c          dn1x=1; dn2x=1; dn3x=1; dn4x=1;
6634   c          dn1y=2; dn2y=2; dn3y=2; dn4y=2;
6635   c          n1=3; n2=3; n3=3; n4=3;
6636   !         OPERATOR MATRIX
6637             DO J=1,4
6638               B(1,4*J)=DN4X*Z_CENT**(J-1)
6639               B(1,4*J-1)=DN3X*Z_CENT**(J-1)
6640               B(1,4*J-2)=DN2X*Z_CENT**(J-1)
6641               B(1,4*J-3)=DN1X*Z_CENT**(J-1)
6642               B(2,4*J+16)=DN4Y*Z_CENT**(J-1)
6643               B(2,4*J-1+16)=DN3Y*Z_CENT**(J-1)
6644               B(2,4*J-2+16)=DN2Y*Z_CENT**(J-1)
6645               B(2,4*J-3+16)=DN1Y*Z_CENT**(J-1)
6646   !
6647               B(4,4*J+32)=DN4Y*Z_CENT**(J-1)
6648               B(4,4*J-1+32)=DN3Y*Z_CENT**(J-1)
6649               B(4,4*J-2+32)=DN2Y*Z_CENT**(J-1)
6650               B(4,4*J-3+32)=DN1Y*Z_CENT**(J-1)
6651   !
6652               B(5,4*J+32)=DN4X*Z_CENT**(J-1)
6653               B(5,4*J-1+32)=DN3X*Z_CENT**(J-1)
6654               B(5,4*J-2+32)=DN2X*Z_CENT**(J-1)
6655               B(5,4*J-3+32)=DN1X*Z_CENT**(J-1)
6656   !
6657               B(6,4*J)=DN4Y*Z_CENT**(J-1)
6658               B(6,4*J-1)=DN3Y*Z_CENT**(J-1)
6659               B(6,4*J-2)=DN2Y*Z_CENT**(J-1)
6660               B(6,4*J-3)=DN1Y*Z_CENT**(J-1)
6661
6662               B(9,4*J+16)=DN4X*Z_CENT**(J-1)
6663               B(9,4*J-1+16)=DN3X*Z_CENT**(J-1)
6664               B(9,4*J-2+16)=DN2X*Z_CENT**(J-1)
6665               B(9,4*J-3+16)=DN1X*Z_CENT**(J-1)
6666             ENDDO
6667             DO J=2,4
6668               B(3,4*J+32)=(J-1)*N4*Z_CENT**(J-2)
6669               B(3,4*J-1+32)=(J-1)*N3*Z_CENT**(J-2)
6670               B(3,4*J-2+32)=(J-1)*N2*Z_CENT**(J-2)
6671               B(3,4*J-3+32)=(J-1)*N1*Z_CENT**(J-2)
6672   !
6673               B(7,4*J+16)=(J-1)*N4*Z_CENT**(J-2)
6674               B(7,4*J-1+16)=(J-1)*N3*Z_CENT**(J-2)
6675               B(7,4*J-2+16)=(J-1)*N2*Z_CENT**(J-2)
6676               B(7,4*J-3+16)=(J-1)*N1*Z_CENT**(J-2)
6677   !
6678               B(8,4*J)=(J-1)*N4*Z_CENT**(J-2)
6679               B(8,4*J-1)=(J-1)*N3*Z_CENT**(J-2)
6680               B(8,4*J-2)=(J-1)*N2*Z_CENT**(J-2)
6681               B(8,4*J-3)=(J-1)*N1*Z_CENT**(J-2)
6682             ENDDO
6683
6684
6685
6686   !         DISP. GRADIENT IN A COLUMN FORMAT
6687             F_COL=MATMUL(B,NODAL_DISP)
6688   C          write(825,*) 'f_col'
6689
6690
6691   !         DISP GRADIENT MATRIX
6692             DEF_GRAD(1,1)=F_COL(1,1)
```

```
6693            DEF_GRAD(2,2)=F_COL(2,1)
6694            DEF_GRAD(3,3)=F_COL(3,1)
6695            DEF_GRAD(3,2)=F_COL(4,1)
6696            DEF_GRAD(3,1)=F_COL(5,1)
6697            DEF_GRAD(1,2)=F_COL(6,1)
6698            DEF_GRAD(2,3)=F_COL(7,1)
6699            DEF_GRAD(1,3)=F_COL(8,1)
6700            DEF_GRAD(2,1)=F_COL(9,1)
6701
6702    !       DEFORMATION GRADIENT MATRIX = I + H
6703            DEF_GRAD=DEF_GRAD+IDEN
6704    !       JACOBIAN
6705            JACOB=DEF_GRAD(1,1)*(DEF_GRAD(2,2)*DEF_GRAD(3,3)-
6706         +  DEF_GRAD(3,2)*DEF_GRAD(2,3))-
6707         +  DEF_GRAD(1,2)*(DEF_GRAD(2,1)*DEF_GRAD(3,3)-DEF_GRAD(3,1)
6708         +  *DEF_GRAD(2,3))+
6709         +  DEF_GRAD(1,3)*(DEF_GRAD(2,1)*DEF_GRAD(3,2)-DEF_GRAD(3,1)
6710         +  *DEF_GRAD(2,2))
6711
6712
6713    !       INVERTING DEFORMATION GRADIENT
6714            F_INV=0.0D0
6715            F_ORD=3
6716            CALL LU_INVERSE(F_ORD,DEF_GRAD,F_INV)
6717
6718
6719    !       STRESS CALCULATION AT THE TOP SURFACE
6720    !       GREEN ST. VENANT STRAIN
6721            E=1.0D0/2.0D0*((MATMUL(TRANSPOSE(DEF_GRAD),DEF_GRAD))-IDEN)
6722
6723    !
6724            STRAIN_COL(1)=E(1,1)
6725            STRAIN_COL(2)=E(2,2)
6726            STRAIN_COL(3)=E(3,3)
6727            STRAIN_COL(4)=2.0D0*E(2,3)
6728            STRAIN_COL(5)=2.0D0*E(1,3)
6729            STRAIN_COL(6)=2.0D0*E(1,2)
6730
6731
6732            S_COL=MATMUL(CC,STRAIN_COL)
6733
6734            S(1,1)=S_COL(1)
6735            S(2,2)=S_COL(2)
6736            S(3,3)=S_COL(3)
6737            S(1,2)=S_COL(6)
6738            S(1,3)=S_COL(5)
6739            S(2,3)=S_COL(4)
6740            S(2,1)=S(1,2)
6741            S(3,1)=S(1,3)
6742            S(3,2)=S(2,3)
6743
6744            T=MATMUL(DEF_GRAD,S)
6745    C        SIG=1.0D0/JACOB*MATMUL(T,TRANSPOSE(DEF_GRAD))
6746            T=TRANSPOSE(T)
6747            SIG=1.0D0/JACOB*MATMUL(DEF_GRAD,T)
6748
6749            N(1,1)=JACOB*F_INV(3,1)
6750            N(2,1)=JACOB*F_INV(3,2)
6751            N(3,1)=JACOB*F_INV(3,3)
6752
6753            DO I=1,3
6754                 N(I,1)=N(I,1)/(SQRT(N(1,1)**2.0D0+N(2,1)**2.0D0
6755         +              +N(3,1)**2.0D0))
6756            ENDDO
6757            TRAC = MATMUL(SIG,N)
6758            MAG=MATMUL(TRANSPOSE(TRAC),N)
6759
6760            IF (TIME_STEP==1 .OR. REM==0) THEN
6761                  WRITE(557,20) X_CENT,Y_CENT,(TRAC(I,1), I=1,3),MAG(1,1)
```

```
6762            ENDIF
6763
6764
6765
6766
6767
6768
6769          FORMAT (9999(x,EN14.4))
6770          FORMAT(EN12.3)
6771          FORMAT(F5.0)
6772          FORMAT (999(2X,I5))
6773
6774
6775           END SUBROUTINE TRAC_VERI
6776
6777          SUBROUTINE SRS(N_FEM,T_P,IELX,IELY,IEL,LE,BE,H
6778         + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,SOL,CON,CC,I_ORDER,LOCAL_C
6779         + ,X_CENT,Y_CENT,G_XY,G_Z,NODAL_ACC,RHO,Z_P,Z_W,STRESS_OUT,N_NODE,
6780         + GLOB_Z_P,COORDG,COORDL,TOT_DOF,E_DOF,NODAL_DISP,N_L,N_B,GLOB_XYZ
6781         + ,AN)
6782          USE PROG_DEBUG
6783          IMPLICIT NONE
6784          INTEGER :: N_FEM,T_P,I,J,K,IEL,IELX,IELY,G_XY,G_Z,TOT_DOF,E_DOF,
6785         + N_L,N_B,N_NODE
6786          REAL(KIND=8) :: STRESS(T_P,6,9), STRAIN(T_P,6,9)
6787          REAL(KIND=8), ALLOCATABLE :: STRESS_TEMP(:,:),COORD_TEMP(:,:)
6788          REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,GN1,GN2,GN3,GN4
6789          REAL(KIND=8) :: N1,N2,N3,N4,
6790         + DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J,H
6791          REAL(KIND=8) :: NDN1X,NDN2X,NDN3X,NDN4X,NDN1Y,NDN2Y,NDN3Y,NDN4Y
6792    !     VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
6793          REAL(KIND=8) :: B(6,48),NODAL_ACC(E_DOF,1),NODAL_DISP(E_DOF,1),
6794         +                AN(TOT_DOF)
6795          REAL(KIND=8) :: TEMP(6,9)
6796          REAL(KIND=8) :: CC(6,6),RHO
6797          REAL(KIND=8) :: LOCAL_C(T_P,3)
6798          INTEGER :: I_ORDER
6799    !     DERIVATIVES OF THE COMPONENTS OF STRESSES AT DIFF. HEIGHTS
6800          REAL(KIND=8) :: STR_DER(G_Z,9),STRESS_OUT(G_Z,12)
6801          REAL(KIND=8) :: STR_TEMP(3),STR_VAL
6802    !     CENTROIDAL COORDINATES
6803          REAL(KIND=8) :: X_CENT,Y_CENT
6804          REAL(KIND=8) :: CENT_ACC(12,1)
6805    !     RHO X ACCELERATION_i
6806          REAL(KIND=8) :: RHO_A_I(G_Z,3)
6807          INTEGER :: THICK_P,DIVS !COUNT OF THICKNESS POINTS AND NO OF DIV.
6808          REAL(KIND=8) :: S13_B, S23_B, S33_B
6809          REAL(KIND=8) :: Z_P(G_Z),Z_W(G_Z),Z_INT(G_Z)
6810          REAL(KIND=8) :: INT_COORD(T_P,3),LINT_COORD(T_P,3)
6811          REAL(KIND=8) :: COORDG(G_XY,2),COORDL(G_XY,2),GLOB_Z_P(G_Z)
6812          REAL(KIND=8) :: Z_CENT
6813          REAL(KIND=8) :: S13_COL(G_Z,1),S23_COL(G_Z,1),S33_COL(G_Z,1)
6814          REAL(KIND=8) :: W_COL(1,G_Z)
6815          REAL(KIND=8) :: S23_VAL(1,1),S13_VAL(1,1),S33_VAL(1,1),
6816         +                S13_R(G_Z,9),S23_R(G_Z,9)
6817          REAL(KIND=8) :: LOC_RS(2,1)  !LOC. CORRDINATES OF THE CENTROID
6818          REAL(KIND=8) :: GLE_X,GLE_Y  !GLOB. X,Y DIST. BETWEEN GAUSS PTS.
6819          REAL(KIND=8) :: SOL(TOT_DOF)
6820          INTEGER :: CON(N_FEM,E_DOF)
6821          REAL(KIND=8) :: DISP(E_DOF,9),ELE_ACC(E_DOF,1),GLOB_XYZ(N_NODE,2)
6822          REAL(KIND=8) :: XX(9),YY(9)   !X,Y COORDS OF 9 ELE. CENTS.
6823          INTEGER :: ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID_SET(9)
6824          REAL(KIND=8) :: COEFF_MAT(9,9),RHS11(9),RHS21(9),RHS12(9)
6825         +                ,RHS22(9),RHS23(9),RHS13(9),MAT_INV(9,9)
6826          REAL(KIND=8) :: CF11(9),CF21(9),CF12(9),CF22(9),CF13(9),CF23(9)
6827
6828    C     PARAMETERS FOR MATRIX INVERSION SUBROUTINE DGETRI.F
6829          INTEGER :: IPIV(9),INFO,LWORK,N_POINT   !N_POINT IS DIM.
6830          REAL(KIND=8) :: LU_COEFF_MAT(9,9), RES(9,9)
```

```
C     LU_COEFF_MAT IS THE LU FACTORIZED FORM OF COEFF MAT
C     WHERE COEFF_MAT=P*L*U
      REAL(KIND=8),ALLOCATABLE :: WORK(:)



!     SPECIFYING THE BOUNDARY CONDITIONS AT THE BOTTOM LAYER
!     LATER TO BE INCLUDED IN THE INPUT FILE
      S13_B=0.0D0
      S23_B=0.0D0
      S33_B=0.0D0


C-----------------------------------------------------------------------
C     INITIALIZING THE COEFFICIENT MATRIX FOR POLYNOMIAL FIT
C-----------------------------------------------------------------------
      COEFF_MAT=0.0D0
      MAT_INV=0.0D0

C-----------------------------------------------------------------------
C     GAUSS POINTS AND WEIGHTS IN Z
C-----------------------------------------------------------------------
      CALL GAUSS_LEG(G_Z,Z_P,Z_W)


C     ALLOCATING THE Z WEIGHTS IN A ROW VECTOR
      DO I=1,G_Z
          W_COL(1,I)=Z_W(I)
      ENDDO


C-----------------------------------------------------------------------
C     DEFINING THE ID NUMBER FOR ELEMENTS IN THE 9 PATCH FOR SRS
C     THE BOUNDARY ELEMENTTS ARE EXCLUDED
C-----------------------------------------------------------------------
      IF (IELX .GT. 1 .AND. IELX .LT. N_L .AND. IELY
     + .GT. 1 .AND. IELY .LT. N_B) THEN    !STATEMENTS TO ID ALL THE INT. ELE.
C         WRITE(*,*) IELY, IELX
C         WRITE(*,*) 'E_DOF',E_DOF
C         WRITE(*,*) 'TOT_DOF',TOT_DOF
C         DO I=1,N_FEM
C              WRITE(165,23) (CON(I,J),J=1,E_DOF)
C         ENDDO
            ID1=(IELY-2)*N_L+IELX-1
            ID2=(IELY-2)*N_L+IELX
            ID3=(IELY-2)*N_L+IELX+1
            ID4=(IELY-1)*N_L+IELX-1
            ID5=(IELY-1)*N_L+IELX
            ID6=(IELY-1)*N_L+IELX+1
            ID7=(IELY)*N_L+IELX-1
            ID8=(IELY)*N_L+IELX
            ID9=(IELY)*N_L+IELX+1
C-----------------------------------------------------------------------
C           ID_SET DEFINES THE ID OF THE 9 ELEMENTS IN ORDER
C-----------------------------------------------------------------------
            ID_SET=(/ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9/)

C             write(*,*) 'id_set'
C             write(*,*) id_set
C             write(*,*) '     '
C             pause


C-----------------------------------------------------------------------
C         DEFINING THE CENTROIDS OF EACH OF THE 9 ELEMENTS IN THE PATCH
C-----------------------------------------------------------------------
          DO J=1,9
            XX(J)=(GLOB_XYZ(CON(ID_SET(J),1),1)+
     +       GLOB_XYZ(CON(ID_SET(J),2),1)+GLOB_XYZ(CON(ID_SET(J),3),1)
```

```
     +       +GLOB_XYZ(CON(ID_SET(J),4),1))/4.0D0
             YY(J)=(GLOB_XYZ(CON(ID_SET(J),1),2)+
     +       GLOB_XYZ(CON(ID_SET(J),2),2)+GLOB_XYZ(CON(ID_SET(J),3),2)
     +       +GLOB_XYZ(CON(ID_SET(J),4),2))/4.0D0
             COEFF_MAT(J,1)=1.0D0
             COEFF_MAT(J,2)=XX(J)
             COEFF_MAT(J,3)=YY(J)
             COEFF_MAT(J,4)=XX(J)**2.0D0
             COEFF_MAT(J,5)=XX(J)*YY(J)
             COEFF_MAT(J,6)=YY(J)**2.0D0
             COEFF_MAT(J,7)=XX(J)**2.0D0*YY(J)
             COEFF_MAT(J,8)=YY(J)**2.0D0*XX(J)
             COEFF_MAT(J,9)=XX(J)**2.0D0*YY(J)**2.0D0
C              write(*,*) (IELY-1)*N_B+IELX, XX(J),YY(J)
           ENDDO
C            write(*,*) '              '


C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C-------------------------------------------------------------------------
C          STEPS FOR MATRIX INVERSION OF THE COEFFICENT MATRIX
C-------------------------------------------------------------------------
C-------------------------------------------------------------------------

           N_POINT=9    !DEGREE OF THE MATRIX 9X9
           LWORK=20     !QUANTITIES FOR THE INVERSION SUBROUTINE
           INFO=0
           ALLOCATE(WORK(LWORK))
           LU_COEFF_MAT=COEFF_MAT
C          CALLING LAPACK FOR LU FACTORIZATION OF COEFF MAT
           CALL DGETRF(N_POINT,N_POINT,LU_COEFF_MAT,N_POINT,IPIV,INFO)
C          INITIATIALIZING MAT_INV AS LU_COEFF_MAT
           MAT_INV=LU_COEFF_MAT
C          CALLING GENERAL MATRIX INVERSION LAPACK SUBROUTINE
           CALL DGETRI(N_POINT,MAT_INV,N_POINT,IPIV,WORK,LWORK,INFO)
C          RESIDUE FROM THE MATRIX INVERSION
           RES=MATMUL(COEFF_MAT,MAT_INV)

C-------------------------------------------------------------------------
C          CHECKING THE RESULT OF MAT INVERSION
C-------------------------------------------------------------------------
C          UN-COMMENT IF NEEDED TO CHECK
C-------------------------------------------------------------------------
C            DO I=1,9
C                WRITE(1,20) (RES(I,J),J=1,9)
C            ENDDO
C            WRITE(166,*) 'IEL',IEL
C            WRITE(166,*) 'MATRIX'
C            DO I=1,9
C                WRITE(166,22) (COEFF_MAT(I,J),J=1,9)
C            ENDDO
C            WRITE(166,*) 'INVERSE'
C            DO I=1,9
C                WRITE(166,22) (MAT_INV(I,J),J=1,9)
C            ENDDO
           DEALLOCATE(WORK)
C!!!!!!!!!!!!! INVERSION STEP COMPLETE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


C-------------------------------------------------------------------------
C          CALLING THE NODAL DISPLACEMENTS OF EACH OF THE 9 ELEMENTS
C          INDICES FOLLOW THE ORDER DISP(DEGREE OF FREEDOM, ELEMENT NO.)
C          SOL IS THE VECTOR OF GLOBAL DISPLACEMENTS OF ALL NODES
C          CON IS THE CONNECTIVITY MATRIX
C-------------------------------------------------------------------------

           DO J=1,E_DOF
             DISP(J,1)=SOL(CON((IELY-2)*N_L+IELX-1,J))
             DISP(J,2)=SOL(CON((IELY-2)*N_L+IELX,J))
```

```
              DISP(J,3)=SOL(CON((IELY-2)*N_L+IELX+1,J))
              DISP(J,4)=SOL(CON((IELY-1)*N_L+IELX-1,J))
              DISP(J,5)=SOL(CON((IELY-1)*N_L+IELX,J))
              DISP(J,6)=SOL(CON((IELY-1)*N_L+IELX+1,J))
              DISP(J,7)=SOL(CON((IELY)*N_L+IELX-1,J))
              DISP(J,8)=SOL(CON((IELY)*N_L+IELX,J))
              DISP(J,9)=SOL(CON((IELY)*N_L+IELX+1,J))
            ENDDO
C           write(*,*) (IELY-2)*N_L+IELX-1
C           write(*,*) (IELY-2)*N_L+IELX
C           write(*,*) (IELY-2)*N_L+IELX+1
C           write(*,*) (IELY-1)*N_L+IELX-1
C           write(*,*) (IELY-1)*N_L+IELX
C           write(*,*) (IELY-1)*N_L+IELX+1
C           write(*,*) (IELY)*N_L+IELX-1
C           write(*,*) (IELY)*N_L+IELX
C           write(*,*) (IELY)*N_L+IELX+1
C           pause

C           DO J=1,E_DOF
C             ELE_ACC(J,1)=NODAL_ACC(CON(IEL,J),1)
C           ENDDO

C             WRITE(*,*) 'FOR ELEMENT',IEL, 'NEIGHBOR LIST'
C             WRITE(*,*) (IELY-2)*N_B+IELX-1
C             WRITE(*,*) (IELY-2)*N_B+IELX
C             WRITE(*,*) (IELY-2)*N_B+IELX+1
C             WRITE(*,*) (IELY-1)*N_B+IELX-1
C             WRITE(*,*) (IELY-1)*N_B+IELX
C             WRITE(*,*) (IELY-1)*N_B+IELX+1
C             WRITE(*,*) (IELY)*N_B+IELX-1
C             WRITE(*,*) (IELY)*N_B+IELX
C             WRITE(*,*) (IELY)*N_B+IELX+1
C             DO I=1,G_Z
C                 W_COL(1,I)=Z_W(I)
C             ENDDO
C         ALLOCATE(STRESS_TEMP(T_P,3),COORD_TEMP(T_P,3))
!         LOGICAL IDENTIFIER DEBUG AND OUTPUT DEFINED IN MODULES.F

!         INTEGTRATION POINT FOR THICKNESS INTEGRATION IN RECOVERY FOR EACH
!         POINT IN Z


C---------------------------------------------------------------------
C         DIVS IS THE NUMBER OF DIVISIONS IN HEIGHT = TOTAL POINTS-1
C---------------------------------------------------------------------
          DIVS=G_Z-1

C         CHECKED
C---------------------------------------------------------------------
!        START OF LOOP FOR RECOVERING STRESS AT EVERY POINT IN THICKNESS
C        RECOVER THE STRESSES AT THE CENTROID LOCATION OF EACH ELEMENT
C        FOR DIFFERENT HEIGHTS. THICK_P ARE THE POINTS IN HEIGHT
C        STARTS AT THE BOTTOM SURFACE AT THICK_P=1
C---------------------------------------------------------------------
C---------------------------------------------------------------------
C---------------------------------------------------------------------
          DO THICK_P = 1,G_Z
C---------------------------------------------------------------------
C             THICK P = 1 IS THE BOTTOM SURFACE
C---------------------------------------------------------------------
              IF (THICK_P==1) THEN
C---------------------------------------------------------------------
C             BOTTOM SURFACE TRANSVERSE STRESSES = BOUNDARY CONTIONS
C             IN NON-LINEAR HAS TO BE DEPENDENT OF F
C---------------------------------------------------------------------
              STRESS_OUT(THICK_P,10)=S23_B
              STRESS_OUT(THICK_P,11)=S13_B
              STRESS_OUT(THICK_P,12)=S33_B
```

```
C-----------------------------------------------------------------------
C             FROM THICK_P=2
C-----------------------------------------------------------------------
              ELSE
C-----------------------------------------------------------------------
C             Z COORDINATE OF THE CENTROID BETWEEN THE BOTTOM SURFACE
C             AND THE CURRECT THICK_P POINT
C             NEEDED FOR GLOBAL GAUSS POINT COORDINATES
C-----------------------------------------------------------------------
              Z_CENT=(GLOB_Z_P(1)+GLOB_Z_P(THICK_P))/2.0D0

C-----------------------------------------------------------------------
C             GLOBAL GAUSS POINT COORD. IN Z FOR INTEGRATING AT THICK_P
C-----------------------------------------------------------------------
              DO J =1,G_Z
                  Z_INT(J)= Z_CENT+H/DIVS/2.0D0*(THICK_P-1.0D0)*Z_P(J)
C                 Z_INT(J)= Z_CENT+H/G_Z/2.0D0*(THICK_P-1.0D0)*Z_P(J)
              ENDDO

C             FOR DEBUGGING
C             SCREEN OUTPUT OF THE Z COORDS OF INT. POINTS FOR EACH Z_P
C             WRITE(*,*) THICK_P,'    ',Z_INT
C-----------------------------------------------------------------------
C             LOCAL AND GLOBAL COORD. OF GAUSS POINTS TOTAL OF 2X2XG_Z
C-----------------------------------------------------------------------
              DO I =1,G_XY
                 DO J=1,G_Z
                    INT_COORD((J-1)*G_XY+I,1)=COORDG(I,1)
                    INT_COORD((J-1)*G_XY+I,2)=COORDG(I,2)
                    INT_COORD((J-1)*G_XY+I,3)=Z_INT(J)
                    LINT_COORD((J-1)*G_XY+I,1)=COORDL(I,1)
                    LINT_COORD((J-1)*G_XY+I,2)=COORDL(I,2)
                    LINT_COORD((J-1)*G_XY+I,3)=Z_INT(J)
                 ENDDO
              ENDDO

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C             PROCEDURE
C             CALCULATE THE STRESSES AT THE 4 GAUSS POINTS AT EACH G_Z
C             AVERAGE THE VALUES OF STRESSES FROM THESE 4 GAUSS POINTS
C             GIVING THE CENTROIDAL STRESS AT EACH G_Z
C             INTEGRATE USING THIS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------


C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C             CALCULATION OF B FOR A GIVEN G_Z
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
              B=0.0D0
              DO I=1,T_P
C                    write(137,20) (int_coord(i,j),j=1,3)
                     CALL FEM_SHAPE_DIFF(LINT_COORD(I,1),LINT_COORD(I,2)
     +               ,I_ORDER,DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,
     +               DET_J,LE,BE)
C                    CALL FEM_SHAPE_GLOBAL(X_I,X_J,Y_I,Y_L,INT_COORD(I,1),
C      +             INT_COORD(I,2),GN1,GN2,GN3,GN4,LE,BE)
                     CALL FEM_SHAPE(LINT_COORD(I,1),LINT_COORD(I,2),
     +               I_ORDER,GN1,GN2,GN3,GN4,DET_J,LE,BE)
c                    dn1x=1; dn2x=1; dn3x=1; dn4x=1;
c                    dn1y=2; dn2y=2; dn3y=2; dn4y=2;
c                    gn1=3; gn2=3; gn3=3; gn4=3;

                     DO J=1,4
                         B(1,4*J)=DN4X*INT_COORD(I,3)**(J-1)
```

```
                            B(1,4*J-1)=DN3X*INT_COORD(I,3)**(J-1)
                            B(1,4*J-2)=DN2X*INT_COORD(I,3)**(J-1)
                            B(1,4*J-3)=DN1X*INT_COORD(I,3)**(J-1)
                            B(2,4*J+16)=DN4Y*INT_COORD(I,3)**(J-1)
                            B(2,4*J-1+16)=DN3Y*INT_COORD(I,3)**(J-1)
                            B(2,4*J-2+16)=DN2Y*INT_COORD(I,3)**(J-1)
                            B(2,4*J-3+16)=DN1Y*INT_COORD(I,3)**(J-1)
!
                            B(4,4*J+32)=DN4Y*INT_COORD(I,3)**(J-1)
                            B(4,4*J-1+32)=DN3Y*INT_COORD(I,3)**(J-1)
                            B(4,4*J-2+32)=DN2Y*INT_COORD(I,3)**(J-1)
                            B(4,4*J-3+32)=DN1Y*INT_COORD(I,3)**(J-1)
!
                            B(5,4*J+32)=DN4X*INT_COORD(I,3)**(J-1)
                            B(5,4*J-1+32)=DN3X*INT_COORD(I,3)**(J-1)
                            B(5,4*J-2+32)=DN2X*INT_COORD(I,3)**(J-1)
                            B(5,4*J-3+32)=DN1X*INT_COORD(I,3)**(J-1)
!
                            B(6,4*J)=DN4Y*INT_COORD(I,3)**(J-1)
                            B(6,4*J-1)=DN3Y*INT_COORD(I,3)**(J-1)
                            B(6,4*J-2)=DN2Y*INT_COORD(I,3)**(J-1)
                            B(6,4*J-3)=DN1Y*INT_COORD(I,3)**(J-1)
                            B(6,4*J+16)=DN4X*INT_COORD(I,3)**(J-1)
                            B(6,4*J-1+16)=DN3X*INT_COORD(I,3)**(J-1)
                            B(6,4*J-2+16)=DN2X*INT_COORD(I,3)**(J-1)
                            B(6,4*J-3+16)=DN1X*INT_COORD(I,3)**(J-1)
                        ENDDO
                        DO J=2,4
                            B(3,4*J+32)=(J-1)*GN4*INT_COORD(I,3)**(J-2)
                            B(3,4*J-1+32)=(J-1)*GN3*INT_COORD(I,3)**(J-2)
                            B(3,4*J-2+32)=(J-1)*GN2*INT_COORD(I,3)**(J-2)
                            B(3,4*J-3+32)=(J-1)*GN1*INT_COORD(I,3)**(J-2)
                !           SHEAR STRESS OPERATORS
                            B(4,4*J+16)=(J-1)*GN4*INT_COORD(I,3)**(J-2)
                            B(4,4*J-1+16)=(J-1)*GN3*INT_COORD(I,3)**(J-2)
                            B(4,4*J-2+16)=(J-1)*GN2*INT_COORD(I,3)**(J-2)
                            B(4,4*J-3+16)=(J-1)*GN1*INT_COORD(I,3)**(J-2)
                !
                            B(5,4*J)=(J-1)*GN4*INT_COORD(I,3)**(J-2)
                            B(5,4*J-1)=(J-1)*GN3*INT_COORD(I,3)**(J-2)
                            B(5,4*J-2)=(J-1)*GN2*INT_COORD(I,3)**(J-2)
                            B(5,4*J-3)=(J-1)*GN1*INT_COORD(I,3)**(J-2)
                        ENDDO
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C                       FOR DEBUGGING, OUTPUT OF B
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

                        IF(DEBUG) THEN
                               WRITE(62,*) 'POINT NO', I
                               DO J=1,6
                                      WRITE(62,20) (B(J,K), K=1,48)
                               ENDDO
                        ENDIF

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C                       TEMP(:,K)=MATMUL(B,DISP(:,K))
C-----------------------------------------------------------------------
C                       TEMP IS THE TEMPORARY STRAIN IN EACH ELEMENT
C                       ORDER OF TEMP = 6 X 9 (6 STRAINS X 9 ELEMENTS)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
                        TEMP=MATMUL(B,DISP)
                        OPEN(4,FILE='STRAIN_VERIF.DAT')
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
```

```
C-----------------------------------------------------------------------
C                     CENT. STRESSES OF THE 9 ELEMENTS IN THE PATCH
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
                      DO K=1,9     !K=ELEMENT NUMBER IN THE PATCH
                        STRAIN(I,1,K)=TEMP(1,K)
                        STRAIN(I,2,K)=TEMP(2,K)
                        STRAIN(I,3,K)=TEMP(3,K)
                        STRAIN(I,4,K)=TEMP(4,K)
                        STRAIN(I,5,K)=TEMP(5,K)
                        STRAIN(I,6,K)=TEMP(6,K)

C                       STRESS(I,:) = MATMUL(CC,STRAIN(I,:))
                        STRESS(I,1,K)=CC(1,1)*STRAIN(I,1,K)
     +                  + CC(1,2)*STRAIN(I,2,K)+CC(1,3)*STRAIN(I,3,K)+
     +                  CC(1,4)*STRAIN(I,4,K)+CC(1,5)*STRAIN(I,5,K)+
     +                  CC(1,6)*STRAIN(I,6,K)

                        STRESS(I,2,K)=CC(2,1)*STRAIN(I,1,K)
     +                  + CC(2,2)*STRAIN(I,2,K)+CC(2,3)*STRAIN(I,3,K)+
     +                  CC(2,4)*STRAIN(I,4,K)+CC(2,5)*STRAIN(I,5,K)+
     +                  CC(2,6)*STRAIN(I,6,K)

                        STRESS(I,3,K)=CC(3,1)*STRAIN(I,1,K)
     +                  + CC(3,2)*STRAIN(I,2,K)+CC(3,3)*STRAIN(I,3,K)+
     +                  CC(3,4)*STRAIN(I,4,K)+CC(3,5)*STRAIN(I,5,K)+
     +                  CC(3,6)*STRAIN(I,6,K)

                        STRESS(I,4,K)=CC(4,4)*STRAIN(I,4,K)+CC(4,5)
     +                  *STRAIN(I,5,K)

                        STRESS(I,5,K)=CC(5,5)*STRAIN(I,5,K)+CC(5,4)
     +                  *STRAIN(I,4,K)

                        STRESS(I,6,K)=CC(6,1)*STRAIN(I,1,K)
     +                  + CC(6,2)*STRAIN(I,2,K)+CC(6,3)*STRAIN(I,3,K)+
     +                  CC(6,4)*STRAIN(I,4,K)+CC(6,5)*STRAIN(I,5,K)+
     +                  CC(6,6)*STRAIN(I,6,K)
                      ENDDO
                ENDDO   !END OF LOOP IN I = 1,T_P
C                 DO I=1,9
C                    WRITE(4,20) (INT_COORD(K,1),INT_COORD(K,2),INT_COORD
C       +             (K,3),STRAIN(K,1,I),STRAIN(K,2,I),STRAIN(K,3,I),
C       +             STRAIN(K,4,I),STRAIN(K,5,I),STRAIN(K,6,I),K=1,T_P)
C                 ENDDO


C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C               ACEELERATION OF THE CENTROID OF THE ELEMENT
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
                DO I=1,12
                  CENT_ACC(I,1)=(NODAL_ACC((I-1)*4+1,1)+NODAL_ACC((I-1)
     +            *4+2,1)+NODAL_ACC((I-1)*4+3,1)+NODAL_ACC((I-1)*4+4,1))
     +            /4.0D0
                ENDDO

                STR_TEMP=0.0D0
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C               DEFINING THE STATEMENT FOR INTERNAL ELEMENTS ONLY
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C               SIGMA_Z CALCULATION FROM 3RD ELE. FROM EACH EDGE
                IF (IELX .GT. 2 .AND. IELX .LT. N_L-1 .AND. IELY
     +                .GT. 2 .AND. IELY .LT. N_B-1) THEN
                   CALL TRANS_NOR(N_FEM,IELX,IELY,IEL,LE,BE,H,
     +             T_P,GLOB_XYZ,SOL,CON,CC,I_ORDER,LOCAL_C,G_XY,G_Z,
```

```
     +           RHO,Z_INT,N_NODE,COORDG,COORDL,TOT_DOF,E_DOF,
     +           N_L,N_B,S13_B,S23_B,S33_B,S13_R,S23_R,AN,GLOB_Z_P)
               ELSE
                  S13_R=0.0D0
                  S23_R=0.0D0
               ENDIF



C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C              CALCULATION OF THE REDUCED STRESS VALUE AT THICK_P
C-----------------------------------------------------------------------
C              LOOP FOR CALCULATIONS AT GAUSS POINTS BETWEEN BOTTOM AND
C              THE THICK_P POINT
C-----------------------------------------------------------------------
               DO I=1,G_Z
C-----------------------------------------------------------------------
C                    CALCULATIONS OF DERIVTIVES OF STRESSES
C                    ORDER (I,J), J: 1=(S11,1),2=(S21,2),3=(S31,3)
C                    4=(S12,1),5=(S22,2),6=(S32,3), 7=(S13,1),8=(S23,2),9=(S33,3)

C                   LOC_RS=0.0D0
C                  CALL FEM_SHAPE_DIFF(LOC_RS(1,1),LOC_RS(2,1),I_ORDER,
C      +             DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)

!                 X AND Y DISTANCE BETWEEN GAUSS POINTS
C                   GLE_X=INT_COORD((I-1)*G_XY+2,1)-INT_COORD
C      +              ((I-1)*G_XY+1,1)
C                   GLE_Y=INT_COORD((I-1)*G_XY+3,2)-INT_COORD
C      +              ((I-1)*G_XY+1,2)

C                   NDN1X=DN1X*LE/GLE_X
C                   NDN2X=DN2X*LE/GLE_X
C                   NDN3X=DN3X*LE/GLE_X
C                   NDN4X=DN4X*LE/GLE_X

C                   NDN1Y=DN1Y*BE/GLE_Y
C                   NDN2Y=DN2Y*BE/GLE_Y
C                   NDN3Y=DN3Y*BE/GLE_Y
C                   NDN4Y=DN4Y*BE/GLE_Y

C                    write(173,*) 'gle_x', gle_x
C                    write(173,*) 'gle_y', gle_y
C                    write(173,20) Loc_rs
C                    write(173,*) 'le',le,'be',be
C                    write(173,20) Ndn1x,Ndn2x,Ndn3x,Ndn4x
C                    write(173,20) Ndn1y,Ndn2y,Ndn3y,Ndn4y

C !                 D/DX1(SIG_11)

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C                   LOOP OVER THE 9 ELEMENTS
C                   RIGHT HAND SIDE = VALUE OF THE CENTROIDAL IN-PLANE STR
C-----------------------------------------------------------------------
C                   CALCUATES THE RHS AT EACH INT. POINT G_Z
C                   OVERWRITES IN EACH LOOP
C-----------------------------------------------------------------------

                    DO K=1,9
                       RHS11(K)=(STRESS((I-1)*G_XY+1,1,K)+
     +                 STRESS((I-1)*G_XY+2,1,K)+STRESS((I-1)*G_XY+3,1,K)
     +                 +STRESS((I-1)*G_XY+4,1,K))/4.0D0

                       RHS21(K)=(STRESS((I-1)*G_XY+1,6,K)+
     +                 STRESS((I-1)*G_XY+2,6,K)+STRESS((I-1)*G_XY+3,6,K)
     +                 +STRESS((I-1)*G_XY+4,6,K))/4.0D0

```

```
                        RHS12(K)=(STRESS((I-1)*G_XY+1,6,K)+
     +                  STRESS((I-1)*G_XY+2,6,K)+STRESS((I-1)*G_XY+3,6,K)
     +                  +STRESS((I-1)*G_XY+4,6,K))/4.0D0

                        RHS22(K)=(STRESS((I-1)*G_XY+1,2,K)+
     +                  STRESS((I-1)*G_XY+2,2,K)+STRESS((I-1)*G_XY+3,2,K)
     +                  +STRESS((I-1)*G_XY+4,2,K))/4.0D0

                        RHS13(K)=S13_R(I,K)

                        RHS23(K)=S23_R(I,K)
                     ENDDO



C                      OPEN(9,FILE='STRESS_VERIF.DAT')
C                      WRITE(9,*) 'PATCH FOR',IEL
C                      WRITE(9,*) 'Z LEVEL',THICK_P, 'G_Z',I
C                      WRITE(9,20) X_CENT,Y_CENT,Z_INT(I)
C                      DO K=1,9
C                            WRITE(9,20) XX(K),YY(K),(RHS11(J),J=1,9)
C                            WRITE(9,20) XX(K),YY(K),(RHS21(J),J=1,9)
C                            WRITE(9,20) XX(K),YY(K),(RHS12(J),J=1,9)
C                            WRITE(9,20) XX(K),YY(K),(RHS22(J),J=1,9)
C                            WRITE(9,20) XX(K),YY(K),(RHS13(J),J=1,9)
C                            WRITE(9,20) XX(K),YY(K),(RHS23(J),J=1,9)
C                      ENDDO


C-----------------------------------------------------------------------
C                     COEFFICEINTS OF THE 2ND DEGREE POLYNOMIAL
C-----------------------------------------------------------------------
                      CF11=MATMUL(MAT_INV,RHS11)
                      CF21=MATMUL(MAT_INV,RHS21)
                      CF12=MATMUL(MAT_INV,RHS12)
                      CF22=MATMUL(MAT_INV,RHS22)
                      CF13=MATMUL(MAT_INV,RHS13)
                      CF23=MATMUL(MAT_INV,RHS23)
C                       write(166,*) 'cf11'
C                       write(166,22) cf11
C                       write(166,*) 'cf21'
C                       write(166,22) cf21
C                       write(166,*) 'cf12'
C                       write(166,22) cf12
C                       write(166,*) 'cf22'
C                       write(166,22) cf22


C-----------------------------------------------------------------------
C                     DERIVATIVES OF STRESSES
C                     RECORDED AT EACH I=1 TO G_Z
C                     ORDER (I,J), J: 1=(S11,1),2=(S21,2),3=(S31,3)
C                     4=(S12,1),5=(S22,2),6=(S32,3), 7=(S13,1),
C                     8=(S23,2),9=(S33,3)
C-----------------------------------------------------------------------

                      STR_DER(I,1)=CF11(2)+2.0D0*CF11(4)*X_CENT+CF11(5)
     +                   *Y_CENT+CF11(7)*2.0D0*X_CENT*Y_CENT+CF11(8)*
     +                   Y_CENT**2.0D0+2.0D0*CF11(9)*X_CENT*Y_CENT**2.0D0

                      STR_DER(I,2)=CF21(3)+CF21(5)*X_CENT+2*CF21(6)
     +                   *Y_CENT+CF21(7)*X_CENT**2.0D0+2.0D0*CF21(8)*
     +                   X_CENT*Y_CENT+2.0D0*CF21(9)*X_CENT**2.0D0*
     +                   Y_CENT

                      STR_DER(I,4)=CF12(2)+2.0D0*CF12(4)*X_CENT+CF12(5)
     +                   *Y_CENT+CF12(7)*2.0D0*X_CENT*Y_CENT+CF12(8)*
     +                   Y_CENT**2.0D0+2.0D0*CF12(9)*X_CENT*Y_CENT**2.0D0

```

```
                    STR_DER(I,5)=CF22(3)+CF22(5)*X_CENT+2*CF22(6)
     +                 *Y_CENT+CF22(7)*X_CENT**2.0D0+2.0D0*CF22(8)*
     +                 X_CENT*Y_CENT+2.0D0*CF22(9)*X_CENT**2.0D0
     +                 *Y_CENT

                    STR_DER(I,7)=CF13(2)+2.0D0*CF13(4)*X_CENT+CF13(5)
     +                 *Y_CENT+CF13(7)*2.0D0*X_CENT*Y_CENT+CF13(8)*
     +                 Y_CENT**2.0D0+2.0D0*CF13(9)*X_CENT*Y_CENT**2.0D0

                    STR_DER(I,8)=CF23(3)+CF23(5)*X_CENT+2.0D0*CF23(6)
     +                 *Y_CENT+CF23(7)*X_CENT**2.0D0+2.0D0*CF23(8)*
     +                 X_CENT*Y_CENT+2.0D0*CF23(9)*X_CENT**2.0D0
     +                 *Y_CENT

C                     OPEN(5,FILE='STRESS_GRAD_VER.DAT')
C                     WRITE(5,*) 'IEL -',IEL,'POINT THICK_P -', THICK_P,
C      +               'G_Z',I
C                     WRITE(5,20) X_CENT,Y_CENT,Z_INT(I),(STR_DER(I,J),J=1,
C      +               5)




C                    STR_DER(I,1)=STRESS((I-1)*G_XY+1,1)*NDN1X+
C      +             STRESS((I-1)*G_XY+2,1)*NDN2X+STRESS((I-1)*G_XY+3,1)*NDN4X+
C      +             STRESS((I-1)*G_XY+4,1)*NDN3X
C        !           D/DX2(SIG_12)
C                    STR_DER(I,2)=STRESS((I-1)*G_XY+1,6)*NDN1Y+
C                    +STRESS((I-1)*G_XY+2,6)*NDN2Y+STRESS((I-1)*G_XY+3,6)*NDN4Y+
C                    + STRESS((I-1)*G_XY+4,6)*NDN3Y
C        !           D/DX1(SIG_12)
C                    STR_DER(I,4)=STRESS((I-1)*G_XY+1,6)*NDN1X+
C                    + STRESS((I-1)*G_XY+2,6)*NDN2X+STRESS((I-1)*G_XY+3,6)*NDN4X+
C                    + STRESS((I-1)*G_XY+4,6)*NDN3X
C        !           D/DX2(SIG_22)
C                    STR_DER(I,5)=STRESS((I-1)*G_XY+1,2)*NDN1Y+
C                    + STRESS((I-1)*G_XY+2,2)*NDN2Y+STRESS((I-1)*G_XY+3,2)*NDN4Y+
C                    + STRESS((I-1)*G_XY+4,2)*NDN3Y

C---------------------------------------------------------------------
C               ACCELERATION AT EACH I=1 TO G_Z
C---------------------------------------------------------------------

                    RHO_A_I(I,1)=RHO*(CENT_ACC(1,1)+INT_COORD(I,3)*CENT_ACC
     +                   (2,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(3,1)+
     +                   INT_COORD(I,3)**3.0D0*CENT_ACC(4,1))
                    RHO_A_I(I,2)=RHO*(CENT_ACC(5,1)+INT_COORD(I,3)*CENT_ACC
     +                   (6,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(7,1)+
     +                   INT_COORD(I,3)**3.0D0*CENT_ACC(8,1))
                    RHO_A_I(I,3)=RHO*(CENT_ACC(9,1)+INT_COORD(I,3)*CENT_ACC
     +                   (10,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(11,1)+
     +                   INT_COORD(I,3)**3.0D0*CENT_ACC(12,1))


C---------------------------------------------------------------------
C               STRESS DERIVATIVES OF TRANSVERSE STRESSES FROM EQ. OF MOTION
C---------------------------------------------------------------------
                     STR_DER(I,3)=RHO_A_I(I,1)-STR_DER(I,1)-STR_DER(I,2)
                     STR_DER(I,6)=RHO_A_I(I,2)-STR_DER(I,4)-STR_DER(I,5)
                     STR_DER(I,9)=RHO_A_I(I,2)-STR_DER(I,7)-STR_DER(I,8)




                     S23_COL(I,1)=STR_DER(I,6)
                     S13_COL(I,1)=STR_DER(I,3)
                     S33_COL(I,1)=STR_DER(I,9)
```

```
C                   WRITE(152,*)
C                   WRITE(152,*) IEL
C                   WRITE(152,20) X_CENT,Y_CENT,INT_COORD((I-1)*G_XY+1,3)
C                   WRITE(152,20) (STR_DER(I,J),J=1,9)



!                  INTEGRATION OF STRESS GRADIENTS
C !                SIGMA_23
c                  STR_TEMP(1)=STR_TEMP(1)+Z_W(I)*STR_DER(I,6)
C !                SIGMA_13
C                  STR_TEMP(2)=STR_TEMP(2)+Z_W(I)*STR_DER(I,3)
C                  STR_TEMP(3)=STR_TEMP(3)+Z_W(I)*STR_DER(I,9)

               ENDDO    !END OF THE LOOP I=1,G_Z


C-------------------------------------------------------------------------
C              STRESS RECOVERY BY INTEGRATION OF THE RHS
C-------------------------------------------------------------------------

               S23_VAL=MATMUL(W_COL,S23_COL)
               S13_VAL=MATMUL(W_COL,S13_COL)
               S33_VAL=MATMUL(W_COL,S33_COL)
               STRESS_OUT(THICK_P,10)=H/DIVS/2.0D0*(THICK_P-1.0D0)
     +         *S23_VAL(1,1)
               STRESS_OUT(THICK_P,11)=H/DIVS/2.0D0*(THICK_P-1.0D0)
     +         *S13_VAL(1,1)




                STRESS_OUT(THICK_P,12)=H/DIVS/2.0D0*(THICK_P-1.0D0)
     +         *S33_VAL(1,1)


C              STATEMENTS TO ID ALL THE INT. ELE.


C                STRESS_OUT(THICK_P,12)=H/DIVS/2.0D0*(THICK_P-1.0D0)
C      +         *S33_VAL(1,1)



               ENDIF !END OF IF STATEMENT FOR THICK_P=2,G_Z
            ENDDO !!!END OF THICK_P LOOP
         ELSEIF (IELX==1) THEN
C        THE STATEMENT FOR IDENTIFYING THE ELEMENTS AT X=0

         ELSEIF (IELX==N_L) THEN


         ENDIF  !END OF IF STATEMENTS FOR INTERIOR ELEMENTS



         FORMAT (EN11.2)
         FORMAT (999(2x,EN14.4))
         FORMAT (999(x,EN11.2))
         FORMAT (999(X,I5))
         END SUBROUTINE SRS
```

```
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C
C
C
C          SUBROUTINE FOR TRANSVERSE NORMAL STRESS CALCULATION
C
C
C
C
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

C     SUBROUTINE TO CALCULATE THE SIGMA 33 GRADIENT ALONG THE HEIGHT
C     FOR EVERY ELEMENT, THE GRADIENT IS CALCULATED OVER 9 ELEMENT PATCH

C       SUBROUTINE TRANS_NOR(N_FEM,T_P,IELX,IELY,IEL,LE,BE,H
C      + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,SOL,CON,CC,I_ORDER,LOCAL_C
C      + ,X_CENT,Y_CENT,G_XY,G_Z,NODAL_ACC,RHO,Z_P,Z_W,STRESS_OUT,N_NODE,
C      + GLOB_Z_P,COORDG,COORDL,TOT_DOF,E_DOF,NODAL_DISP,N_L,N_B,GLOB_XYZ,
C      + S13_B,S23_B,S33_B,S13_R,S23_R)
      SUBROUTINE TRANS_NOR(N_FEM,IELX,IELY,IEL,LE,BE,H,T_P,
     + GLOB_XYZ,SOL,CON,CC,I_ORDER,LOCAL_C,G_XY,G_Z,RHO,Z_INT,
     + N_NODE,COORDG,COORDL,TOT_DOF,E_DOF,N_L,N_B,
     + S13_B,S23_B,S33_B,S13_R,S23_R,AN,GLOB_Z_P)
      USE PROG_DEBUG
      IMPLICIT NONE
      INTEGER :: N_FEM,T_P,I,J,K,IEL,IELX,IELY,G_XY,G_Z,TOT_DOF,E_DOF,
     + N_L,N_B,N_NODE
      REAL(KIND=8) :: STRESS(T_P,6,9), STRAIN(T_P,6,9)
      REAL(KIND=8), ALLOCATABLE :: STRESS_TEMP(:,:),COORD_TEMP(:,:)
      REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,GN1,GN2,GN3,GN4
      REAL(KIND=8) :: N1,N2,N3,N4,
     + DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J,H
      REAL(KIND=8) :: NDN1X,NDN2X,NDN3X,NDN4X,NDN1Y,NDN2Y,NDN3Y,NDN4Y
!     VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
      REAL(KIND=8) :: B(6,48),NODAL_ACC(E_DOF,1),AN(TOT_DOF)
      REAL(KIND=8) :: TEMP(6,9)
      REAL(KIND=8) :: CC(6,6),RHO
      REAL(KIND=8) :: LOCAL_C(T_P,3)
      INTEGER :: I_ORDER
!     DERIVATIVES OF THE COMPONENTS OF STRESSES AT DIFF. HEIGHTS
       REAL(KIND=8) :: STR_DER(G_Z,9)  !,STRESS_OUT(G_Z,12)
```

```
      REAL(KIND=8) :: STR_TEMP(3),STR_VAL
!     CENTROIDAL COORDINATES
      REAL(KIND=8) :: X_CENT,Y_CENT
      REAL(KIND=8) :: CENT_ACC(12,1)
!     RHO X ACCELERATION_i
      REAL(KIND=8) :: RHO_A_I(G_Z,3)
      INTEGER :: THICK_P,DIVS,THICK_P2 !COUNT OF THICK. PTS., NO OF DIV.
      REAL(KIND=8) :: S13_B, S23_B, S33_B
      REAL(KIND=8) :: Z_P(G_Z),Z_W(G_Z),Z_INT(G_Z),Z_INT2(G_Z)
      REAL(KIND=8) :: INT_COORD(T_P,3),LINT_COORD(T_P,3)
      REAL(KIND=8) :: COORDG(G_XY,2),COORDL(G_XY,2),GLOB_Z_P(G_Z)
      REAL(KIND=8) :: Z_CENT,Z_CENT2
      REAL(KIND=8) :: S13_COL(G_Z,1),S23_COL(G_Z,1),S33_COL(G_Z,1),
     +                S13_R(G_Z,9),S23_R(G_Z,9)
      REAL(KIND=8) :: W_COL(1,G_Z)
      REAL(KIND=8) :: S23_VAL(1,1),S13_VAL(1,1),S33_VAL(1,1)
      REAL(KIND=8) :: LOC_RS(2,1)  !LOC. CORRDINATES OF THE CENTROID
      REAL(KIND=8) :: GLE_X,GLE_Y  !GLOB. X,Y DIST. BETWEEN GAUSS PTS.
      REAL(KIND=8) :: SOL(TOT_DOF)
      INTEGER :: CON(N_FEM,E_DOF)
      REAL(KIND=8) :: DISP(E_DOF,9),ELE_ACC(E_DOF,1),GLOB_XYZ(N_NODE,2)
      REAL(KIND=8) :: XX(9),YY(9)   !X,Y COORDS OF 9 ELE. CENTS.
      INTEGER :: ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID_SET(9)
      REAL(KIND=8) :: COEFF_MAT(9,9),RHS11(9),RHS21(9),RHS12(9)
     +                ,RHS22(9),MAT_INV(9,9),RHS23(9),RHS13(9)
      REAL(KIND=8) :: CF11(9),CF21(9),CF12(9),CF22(9),CF13(9),CF23(9)

C     PARAMETERS FOR MATRIX INVERSION SUBROUTINE DGETRI.F
      INTEGER :: IPIV(9),INFO,LWORK,N_POINT   !N_POINT IS DIM.
      REAL(KIND=8) :: LU_COEFF_MAT(9,9), RES(9,9)
C     LU_COEFF_MAT IS THE LU FACTORIZED FORM OF COEFF MAT
C     WHERE COEFF_MAT=P*L*U
      REAL(KIND=8),ALLOCATABLE :: WORK(:)

C     ARRAY FOR STORING TRANSVERSE STRESSES AT THE GAUSS POINTS IN ELE.
      REAL(KIND=8), ALLOCATABLE :: STR_TRAN(:,:,:)
      INTEGER :: X_NO(9),Y_NO(9),N_PAT(9),N_COUNT
      REAL(KIND=8) :: S13_TEMP(G_Z,9),S23_TEMP(G_Z,9)


C !     SPECIFYING THE BOUNDARY CONDITIONS AT THE BOTTOM LAYER
C !     LATER TO BE INCLUDED IN THE INPUT FILE
C       S13_B=0.0D0
C       S23_B=0.0D0
C       S33_B=0.0D0

      COEFF_MAT=0.0D0
      MAT_INV=0.0D0
      RHO_A_I=0.0D0
      NODAL_ACC=0.0D0

      CALL GAUSS_LEG(G_Z,Z_P,Z_W)

      DO I=1,G_Z
          W_COL(1,I)=Z_W(I)
      ENDDO

C         WRITE(*,*) 'E_DOF',E_DOF
C         WRITE(*,*) 'TOT_DOF',TOT_DOF
C         DO I=1,N_FEM
C               WRITE(165,23) (CON(I,J),J=1,E_DOF)
C         ENDDO


C     IDEN. OF SURROUNDING ELE. X NO. FOR A CENTRAL ELE
      X_NO=(/IELX-1,IELX,IELX+1,IELX-1,IELX,IELX+1,IELX-1,IELX,
     + IELX+1/)
C     IDEN. OF SURROUNDING ELE. X NO. FOR A CENTRAL ELE
      Y_NO=(/IELY-1,IELY-1,IELY-1,IELY,IELY,IELY,IELY+1,IELY+1,
```

```
      + IELY+1/)
C     NUMBER OF THE CENTRAL ELE.
      N_PAT(:)=(Y_NO(:)-1)*N_L+X_NO(:)
C       write(*,*) '  '
C       WRITE(*,*) 'OUTPUT OF X_NO'
C       WRITE(*,23) X_NO
C       WRITE(*,*) 'OUTPUT OF Y_NO'
C       WRITE(*,23) Y_NO
C       WRITE(*,*) 'OUTPUT OF N_PAT'
C       WRITE(*,23) N_PAT



C     START OF THE LOOP FOR FINDING SHEAR STRESSES AT THE CENTERS OF
C     16 ELEMENTS SURROUNDING THE 9 ELE PATCH
      DO N_COUNT=1,9

          ID1=(Y_NO(N_COUNT)-2)*N_L+X_NO(N_COUNT)-1
          ID2=(Y_NO(N_COUNT)-2)*N_L+X_NO(N_COUNT)
          ID3=(Y_NO(N_COUNT)-2)*N_L+X_NO(N_COUNT)+1
          ID4=(Y_NO(N_COUNT)-1)*N_L+X_NO(N_COUNT)-1
          ID5=(Y_NO(N_COUNT)-1)*N_L+X_NO(N_COUNT)
          ID6=(Y_NO(N_COUNT)-1)*N_L+X_NO(N_COUNT)+1
          ID7=(Y_NO(N_COUNT))*N_L+X_NO(N_COUNT)-1
          ID8=(Y_NO(N_COUNT))*N_L+X_NO(N_COUNT)
          ID9=(Y_NO(N_COUNT))*N_L+X_NO(N_COUNT)+1
          ID_SET=(/ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9/)
C           WRITE(*,*) 'OUTPUT OF ID_SET'
C           WRITE(*,23) ID_SET

          DO J=1,E_DOF
              NODAL_ACC(J,1)=AN(CON(N_PAT(N_COUNT),J))
          ENDDO

C



C         DEFINING THE CENTROIDAL COORDINATES OF THE 9 NEIGHBORING ELE.
C         OF THE ELEMENT N_COUNT AND THE COEFF. MAT. FOR INTERPOLATION
          DO J=1,9
            XX(J)=(GLOB_XYZ(CON(ID_SET(J),1),1)+
     +       GLOB_XYZ(CON(ID_SET(J),2),1)+GLOB_XYZ(CON(ID_SET(J),3),1)
     +       +GLOB_XYZ(CON(ID_SET(J),4),1))/4.0D0
            YY(J)=(GLOB_XYZ(CON(ID_SET(J),1),2)+
     +       GLOB_XYZ(CON(ID_SET(J),2),2)+GLOB_XYZ(CON(ID_SET(J),3),2)
     +       +GLOB_XYZ(CON(ID_SET(J),4),2))/4.0D0
            COEFF_MAT(J,1)=1.0D0
            COEFF_MAT(J,2)=XX(J)
            COEFF_MAT(J,3)=YY(J)
            COEFF_MAT(J,4)=XX(J)**2.0D0
            COEFF_MAT(J,5)=XX(J)*YY(J)
            COEFF_MAT(J,6)=YY(J)**2.0D0
            COEFF_MAT(J,7)=XX(J)**2.0D0*YY(J)
            COEFF_MAT(J,8)=YY(J)**2.0D0*XX(J)
            COEFF_MAT(J,9)=XX(J)**2.0D0*YY(J)**2.0D0
          ENDDO

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C         STEPS FOR MATRIX INVERSION

          N_POINT=9    !DEGREE OF THE MATRIX
          LWORK=20     !QUANTITIES FOR THE INVERSION SUBROUTINE
          INFO=0
          ALLOCATE(WORK(LWORK))
          LU_COEFF_MAT=COEFF_MAT
C         CALLING LAPACK FOR LU FACTORIZATION OF COEFF MAT
          CALL DGETRF(N_POINT,N_POINT,LU_COEFF_MAT,N_POINT,IPIV,INFO)
```

```
C          INITIATIALIZING MAT_INV AS LU_COEFF_MAT
           MAT_INV=LU_COEFF_MAT
C          CALLING GENERAL MATRIX INVERSION LAPACK SUBROUTINE
           CALL DGETRI(N_POINT,MAT_INV,N_POINT,IPIV,WORK,LWORK,INFO)
C          RESIDUE FROM THE MATRIX INVERSION
           RES=MATMUL(COEFF_MAT,MAT_INV)


C          CHECKING THE RESULT OF MAT INVERSION
C            DO I=1,9
C                WRITE(1,20) (RES(I,J),J=1,9)
C            ENDDO
C            WRITE(166,*) 'IEL',IEL
C            WRITE(166,*) 'MATRIX'
C            DO I=1,9
C                WRITE(166,22) (COEFF_MAT(I,J),J=1,9)
C            ENDDO
C            WRITE(166,*) 'INVERSE'
C            DO I=1,9
C                WRITE(166,22) (MAT_INV(I,J),J=1,9)
C            ENDDO
           DEALLOCATE(WORK)
C!!!!!!!!!!!! INVERSION STEP COMPLETE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


C            WRITE(*,*) 'FOR ELE',IEL
C            WRITE(*,23) ID_SET
C            WRITE(*,22) XX
C            WRITE(*,22) YY
           DO J=1,E_DOF
             DISP(J,1)=SOL(CON((Y_NO(N_COUNT)-2)*N_L+X_NO(N_COUNT)-1,J))
             DISP(J,2)=SOL(CON((Y_NO(N_COUNT)-2)*N_L+X_NO(N_COUNT),J))
             DISP(J,3)=SOL(CON((Y_NO(N_COUNT)-2)*N_L+X_NO(N_COUNT)+1,J))
             DISP(J,4)=SOL(CON((Y_NO(N_COUNT)-1)*N_L+X_NO(N_COUNT)-1,J))
             DISP(J,5)=SOL(CON((Y_NO(N_COUNT)-1)*N_L+X_NO(N_COUNT),J))
             DISP(J,6)=SOL(CON((Y_NO(N_COUNT)-1)*N_L+X_NO(N_COUNT)+1,J))
             DISP(J,7)=SOL(CON((Y_NO(N_COUNT))*N_L+X_NO(N_COUNT)-1,J))
             DISP(J,8)=SOL(CON((Y_NO(N_COUNT))*N_L+X_NO(N_COUNT),J))
             DISP(J,9)=SOL(CON((Y_NO(N_COUNT))*N_L+X_NO(N_COUNT)+1,J))
           ENDDO
C            DO J=1,E_DOF
C              ELE_ACC(J,1)=NODAL_ACC(CON(IEL,J),1)
C            ENDDO

C              WRITE(*,*) 'FOR ELEMENT',IEL, 'NEIGHBOR LIST'
C              WRITE(*,*) (IELY-2)*N_B+IELX-1
C              WRITE(*,*) (IELY-2)*N_B+IELX
C              WRITE(*,*) (IELY-2)*N_B+IELX+1
C              WRITE(*,*) (IELY-1)*N_B+IELX-1
C              WRITE(*,*) (IELY-1)*N_B+IELX
C              WRITE(*,*) (IELY-1)*N_B+IELX+1
C              WRITE(*,*) (IELY)*N_B+IELX-1
C              WRITE(*,*) (IELY)*N_B+IELX
C              WRITE(*,*) (IELY)*N_B+IELX+1
C              DO I=1,G_Z
C                  W_COL(1,I)=Z_W(I)
C              ENDDO
C          ALLOCATE(STRESS_TEMP(T_P,3),COORD_TEMP(T_P,3))
!          LOGICAL IDENTIFIER DEBUG AND OUTPUT DEFINED IN MODULES.F

!          INTEGTRATION POINT FOR THICKNESS INTEGRATION IN RECOVERY FOR EACH
!          POINT IN Z


C          CHECKED
C            write(*,*) 'glob_z_p'
C            write(*,20) glob_z_p
C            write(*,*) 'z_int'
C            write(*,20) z_int
```

```
!         START OF LOOP FOR RECOVERING STRESS AT EVERY POINT IN THICKNESS
          DO THICK_P2 = 1,G_Z
C             THICK P = 1 IS THE BOTTOM SURFACE
C               IF (THICK_P2 ==1) THEN
C                 S23_R(THICK_P2,N_COUNT)=S23_B
C                 S13_R(THICK_P2,N_COUNT)=S13_B
C               ELSE
!             Z COORDINATE OF THE CENTROID BETWEEN TWO POINTS IN THICK.
                Z_CENT2=(GLOB_Z_P(1)+Z_INT(THICK_P2))/2.0D0
                DO J=1,G_Z
                  Z_INT2(J)=Z_CENT2+(Z_INT(THICK_P2)-GLOB_Z_P(1))/2.0D0
     +            *Z_P(J)
                ENDDO
C               write(*,*) 'thick_p2', thick_p2
C               write(*,*) 'z_cent2'
C               write(*,*) z_cent2
C               write(*,*) 'z_int2'
C               write(*,20) z_int2
C               pause
                DO I =1,G_XY
                  DO J=1,G_Z
                    INT_COORD((J-1)*G_XY+I,1)=COORDG(I,1)
                    INT_COORD((J-1)*G_XY+I,2)=COORDG(I,2)
                    INT_COORD((J-1)*G_XY+I,3)=Z_INT2(J)
                    LINT_COORD((J-1)*G_XY+I,1)=COORDL(I,1)
                    LINT_COORD((J-1)*G_XY+I,2)=COORDL(I,2)
                    LINT_COORD((J-1)*G_XY+I,3)=Z_INT2(J)
                  ENDDO
                ENDDO

C               DO I=1,T_P
C                   WRITE(*,20) (LINT_COORD(I,J), J=1,3)
C               ENDDO
C               PAUSE

                B=0.0D0
                DO I=1,T_P
C                 write(137,20) (int_coord(i,j),j=1,3)
                  CALL FEM_SHAPE_DIFF(LINT_COORD(I,1),LINT_COORD(I,2)
     +            ,I_ORDER,DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,
     +            DET_J,LE,BE)
C                 CALL FEM_SHAPE_GLOBAL(X_I,X_J,Y_I,Y_L,INT_COORD(I,1),
C      +          INT_COORD(I,2),GN1,GN2,GN3,GN4,LE,BE)
                  CALL FEM_SHAPE(LINT_COORD(I,1),LINT_COORD(I,2),
     +            I_ORDER,GN1,GN2,GN3,GN4,DET_J,LE,BE)
c                 dn1x=1; dn2x=1; dn3x=1; dn4x=1;
c                 dn1y=2; dn2y=2; dn3y=2; dn4y=2;
c                 gn1=3; gn2=3; gn3=3; gn4=3;

                  DO J=1,4
                         B(1,4*J)=DN4X*INT_COORD(I,3)**(J-1)
                         B(1,4*J-1)=DN3X*INT_COORD(I,3)**(J-1)
                         B(1,4*J-2)=DN2X*INT_COORD(I,3)**(J-1)
                         B(1,4*J-3)=DN1X*INT_COORD(I,3)**(J-1)
                         B(2,4*J+16)=DN4Y*INT_COORD(I,3)**(J-1)
                         B(2,4*J-1+16)=DN3Y*INT_COORD(I,3)**(J-1)
                         B(2,4*J-2+16)=DN2Y*INT_COORD(I,3)**(J-1)
                         B(2,4*J-3+16)=DN1Y*INT_COORD(I,3)**(J-1)
          !
                         B(4,4*J+32)=DN4Y*INT_COORD(I,3)**(J-1)
                         B(4,4*J-1+32)=DN3Y*INT_COORD(I,3)**(J-1)
                         B(4,4*J-2+32)=DN2Y*INT_COORD(I,3)**(J-1)
                         B(4,4*J-3+32)=DN1Y*INT_COORD(I,3)**(J-1)
          !
                         B(5,4*J+32)=DN4X*INT_COORD(I,3)**(J-1)
                         B(5,4*J-1+32)=DN3X*INT_COORD(I,3)**(J-1)
                         B(5,4*J-2+32)=DN2X*INT_COORD(I,3)**(J-1)
                         B(5,4*J-3+32)=DN1X*INT_COORD(I,3)**(J-1)
```

```
7934          !
7935                            B(6,4*J)=DN4Y*INT_COORD(I,3)**(J-1)
7936                            B(6,4*J-1)=DN3Y*INT_COORD(I,3)**(J-1)
7937                            B(6,4*J-2)=DN2Y*INT_COORD(I,3)**(J-1)
7938                            B(6,4*J-3)=DN1Y*INT_COORD(I,3)**(J-1)
7939                            B(6,4*J+16)=DN4X*INT_COORD(I,3)**(J-1)
7940                            B(6,4*J-1+16)=DN3X*INT_COORD(I,3)**(J-1)
7941                            B(6,4*J-2+16)=DN2X*INT_COORD(I,3)**(J-1)
7942                            B(6,4*J-3+16)=DN1X*INT_COORD(I,3)**(J-1)
7943                      ENDDO
7944                      DO J=2,4
7945                            B(3,4*J+32)=(J-1)*GN4*INT_COORD(I,3)**(J-2)
7946                            B(3,4*J-1+32)=(J-1)*GN3*INT_COORD(I,3)**(J-2)
7947                            B(3,4*J-2+32)=(J-1)*GN2*INT_COORD(I,3)**(J-2)
7948                            B(3,4*J-3+32)=(J-1)*GN1*INT_COORD(I,3)**(J-2)
7949                !           SHEAR STRESS OPERATORS
7950                            B(4,4*J+16)=(J-1)*GN4*INT_COORD(I,3)**(J-2)
7951                            B(4,4*J-1+16)=(J-1)*GN3*INT_COORD(I,3)**(J-2)
7952                            B(4,4*J-2+16)=(J-1)*GN2*INT_COORD(I,3)**(J-2)
7953                            B(4,4*J-3+16)=(J-1)*GN1*INT_COORD(I,3)**(J-2)
7954                !
7955                            B(5,4*J)=(J-1)*GN4*INT_COORD(I,3)**(J-2)
7956                            B(5,4*J-1)=(J-1)*GN3*INT_COORD(I,3)**(J-2)
7957                            B(5,4*J-2)=(J-1)*GN2*INT_COORD(I,3)**(J-2)
7958                            B(5,4*J-3)=(J-1)*GN1*INT_COORD(I,3)**(J-2)
7959                      ENDDO
7960
7961                          IF(DEBUG) THEN
7962                                 WRITE(62,*) 'POINT NO', I
7963                                 DO J=1,6
7964                                        WRITE(62,20) (B(J,K), K=1,48)
7965                                 ENDDO
7966                          ENDIF
7967
7968
7969   C                      TEMP(:,K)=MATMUL(B,DISP(:,K))
7970                          TEMP=MATMUL(B,DISP)
7971   C                       OPEN(4,FILE='STRAIN_VERIF.DAT')
7972                          DO K=1,9      !K=ELEMENT NUMBER IN THE PATCH
7973                            STRAIN(I,1,K)=TEMP(1,K)
7974                            STRAIN(I,2,K)=TEMP(2,K)
7975                            STRAIN(I,3,K)=TEMP(3,K)
7976                            STRAIN(I,4,K)=TEMP(4,K)
7977                            STRAIN(I,5,K)=TEMP(5,K)
7978                            STRAIN(I,6,K)=TEMP(6,K)
7979
7980   C                        STRESS(I,:) = MATMUL(CC,STRAIN(I,:))
7981                            STRESS(I,1,K)=CC(1,1)*STRAIN(I,1,K)
7982         +                  + CC(1,2)*STRAIN(I,2,K)+CC(1,3)*STRAIN(I,3,K)+
7983         +                  CC(1,4)*STRAIN(I,4,K)+CC(1,5)*STRAIN(I,5,K)+
7984         +                  CC(1,6)*STRAIN(I,6,K)
7985
7986                            STRESS(I,2,K)=CC(2,1)*STRAIN(I,1,K)
7987         +                  + CC(2,2)*STRAIN(I,2,K)+CC(2,3)*STRAIN(I,3,K)+
7988         +                  CC(2,4)*STRAIN(I,4,K)+CC(2,5)*STRAIN(I,5,K)+
7989         +                  CC(2,6)*STRAIN(I,6,K)
7990
7991                            STRESS(I,3,K)=CC(3,1)*STRAIN(I,1,K)
7992         +                  + CC(3,2)*STRAIN(I,2,K)+CC(3,3)*STRAIN(I,3,K)+
7993         +                  CC(3,4)*STRAIN(I,4,K)+CC(3,5)*STRAIN(I,5,K)+
7994         +                  CC(3,6)*STRAIN(I,6,K)
7995
7996                            STRESS(I,4,K)=CC(4,4)*STRAIN(I,4,K)+CC(4,5)
7997         +                  *STRAIN(I,5,K)
7998
7999                            STRESS(I,5,K)=CC(5,5)*STRAIN(I,5,K)+CC(5,4)
8000         +                  *STRAIN(I,4,K)
8001
8002                            STRESS(I,6,K)=CC(6,1)*STRAIN(I,1,K)
```

```
8003        +                  + CC(6,2)*STRAIN(I,2,K)+CC(6,3)*STRAIN(I,3,K)+
8004        +                  CC(6,4)*STRAIN(I,4,K)+CC(6,5)*STRAIN(I,5,K)+
8005        +                  CC(6,6)*STRAIN(I,6,K)
8006                         ENDDO
8007                       ENDDO   !END OF LOOP IN I = 1,T_P
8008    C                DO I=1,9
8009    C                   WRITE(4,20) (INT_COORD(K,1),INT_COORD(K,2),INT_COORD
8010    C       +            (K,3),STRAIN(K,1,I),STRAIN(K,2,I),STRAIN(K,3,I),
8011    C       +            STRAIN(K,4,I),STRAIN(K,5,I),STRAIN(K,6,I),K=1,T_P)
8012    C                ENDDO
8013                   DO I=1,12
8014                     CENT_ACC(I,1)=(NODAL_ACC((I-1)*4+1,1)+NODAL_ACC((I-1)
8015        +            *4+2,1)+NODAL_ACC((I-1)*4+3,1)+NODAL_ACC((I-1)*4+4,1))
8016        +            /4.0D0
8017                   ENDDO
8018                   STR_TEMP=0.0D0
8019                   DO I=1,G_Z
8020    !                    CALCULATIONS OF DERIVTIVES OF STRESSES
8021    !                    ORDER (I,J), J: 1=(S11,1),2=(S21,2),3=(S31,3)
8022    !                    4=(S12,1),5=(S22,2),6=(S32,3), 7=(S13,1),8=(S23,2),9=(S33,3)
8023
8024    C                    LOC_RS=0.0D0
8025    C                   CALL FEM_SHAPE_DIFF(LOC_RS(1,1),LOC_RS(2,1),I_ORDER,
8026    C       +            DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,DET_J,LE,BE)
8027
8028    !                  X AND Y DISTANCE BETWEEN GAUSS POINTS
8029    C                    GLE_X=INT_COORD((I-1)*G_XY+2,1)-INT_COORD
8030    C       +             ((I-1)*G_XY+1,1)
8031    C                    GLE_Y=INT_COORD((I-1)*G_XY+3,2)-INT_COORD
8032    C       +             ((I-1)*G_XY+1,2)
8033
8034    C                    NDN1X=DN1X*LE/GLE_X
8035    C                    NDN2X=DN2X*LE/GLE_X
8036    C                    NDN3X=DN3X*LE/GLE_X
8037    C                    NDN4X=DN4X*LE/GLE_X
8038
8039    C                    NDN1Y=DN1Y*BE/GLE_Y
8040    C                    NDN2Y=DN2Y*BE/GLE_Y
8041    C                    NDN3Y=DN3Y*BE/GLE_Y
8042    C                    NDN4Y=DN4Y*BE/GLE_Y
8043
8044    C                     write(173,*) 'gle_x', gle_x
8045    C                     write(173,*) 'gle_y', gle_y
8046    C                     write(173,20) Loc_rs
8047    C                     write(173,*) 'le',le,'be',be
8048    C                     write(173,20) Ndn1x,Ndn2x,Ndn3x,Ndn4x
8049    C                     write(173,20) Ndn1y,Ndn2y,Ndn3y,Ndn4y
8050
8051    C !                 D/DX1(SIG_11)
8052                        DO K=1,9
8053                           RHS11(K)=(STRESS((I-1)*G_XY+1,1,K)+
8054        +                  STRESS((I-1)*G_XY+2,1,K)+STRESS((I-1)*G_XY+3,1,K)
8055        +                  +STRESS((I-1)*G_XY+4,1,K))/4.0D0
8056
8057                           RHS21(K)=(STRESS((I-1)*G_XY+1,6,K)+
8058        +                  STRESS((I-1)*G_XY+2,6,K)+STRESS((I-1)*G_XY+3,6,K)
8059        +                  +STRESS((I-1)*G_XY+4,6,K))/4.0D0
8060
8061                           RHS12(K)=(STRESS((I-1)*G_XY+1,6,K)+
8062        +                  STRESS((I-1)*G_XY+2,6,K)+STRESS((I-1)*G_XY+3,6,K)
8063        +                  +STRESS((I-1)*G_XY+4,6,K))/4.0D0
8064
8065                           RHS22(K)=(STRESS((I-1)*G_XY+1,2,K)+
8066        +                  STRESS((I-1)*G_XY+2,2,K)+STRESS((I-1)*G_XY+3,2,K)
8067        +                  +STRESS((I-1)*G_XY+4,2,K))/4.0D0
8068                        ENDDO
8069    C                     write(*,*) 'rhs11'
8070    C                     write(*,20) rhs11
8071    C                     write(*,*) 'rhs21'
```

```
C                   write(*,20) rhs21
C                   write(*,*) 'rhs12'
C                   write(*,20) rhs12
C                   write(*,*) 'rhs22'
C                   write(*,20) rhs22
C                   OPEN(9,FILE='STRESS_VERIF.DAT')
C                   WRITE(9,*) 'PATCH FOR',IEL
C                   WRITE(9,*) 'Z LEVEL',THICK_P, 'G_Z',I
C                   WRITE(9,20) X_CENT,Y_CENT,Z_INT(I)
C                   DO K=1,9
C                         WRITE(9,20) XX(K),YY(K),(RHS11(J),J=1,9)
C                         WRITE(9,20) XX(K),YY(K),(RHS21(J),J=1,9)
C                         WRITE(9,20) XX(K),YY(K),(RHS12(J),J=1,9)
C                         WRITE(9,20) XX(K),YY(K),(RHS22(J),J=1,9)
C                   ENDDO



                  CF11=MATMUL(MAT_INV,RHS11)
                  CF21=MATMUL(MAT_INV,RHS21)
                  CF12=MATMUL(MAT_INV,RHS12)
                  CF22=MATMUL(MAT_INV,RHS22)

                  STR_DER(I,1)=CF11(2)+2.0D0*CF11(4)*X_CENT+CF11(5)
     +               *Y_CENT+CF11(7)*2.0D0*X_CENT*Y_CENT+CF11(8)*
     +               Y_CENT**2.0D0+2.0D0*CF11(9)*X_CENT*Y_CENT**2.0D0

                  STR_DER(I,2)=CF21(3)+CF21(5)*X_CENT+2.0D0*CF21(6)
     +               *Y_CENT+CF21(7)*X_CENT**2.0D0+2.0D0*CF21(8)*
     +               X_CENT*Y_CENT+2.0D0*CF21(9)*X_CENT**2.0D0*
     +               Y_CENT

                  STR_DER(I,4)=CF12(2)+2.0D0*CF12(4)*X_CENT+CF12(5)
     +               *Y_CENT+CF12(7)*2.0D0*X_CENT*Y_CENT+CF12(8)*
     +               Y_CENT**2.0D0+2.0D0*CF12(9)*X_CENT*Y_CENT**2.0D0

                  STR_DER(I,5)=CF22(3)+CF22(5)*X_CENT+2.0D0*CF22(6)
     +               *Y_CENT+CF22(7)*X_CENT**2.0D0+2.0D0*CF22(8)*
     +               X_CENT*Y_CENT+2.0D0*CF22(9)*X_CENT**2.0D0
     +               *Y_CENT

C                   OPEN(5,FILE='STRESS_GRAD_VER.DAT')
C                   WRITE(5,*) 'IEL -',IEL,'POINT THICK_P -', THICK_P,
C     +              'G_Z',I
C                   WRITE(5,20) X_CENT,Y_CENT,Z_INT(I),(STR_DER(I,J),J=1,
C     +              5)



C                   STR_DER(I,1)=STRESS((I-1)*G_XY+1,1)*NDN1X+
C     +             STRESS((I-1)*G_XY+2,1)*NDN2X+STRESS((I-1)*G_XY+3,1)*NDN4X+
C     +             STRESS((I-1)*G_XY+4,1)*NDN3X
C           !       D/DX2(SIG_12)
C                   STR_DER(I,2)=STRESS((I-1)*G_XY+1,6)*NDN1Y+
C                   +STRESS((I-1)*G_XY+2,6)*NDN2Y+STRESS((I-1)*G_XY+3,6)*NDN4Y+
C                   + STRESS((I-1)*G_XY+4,6)*NDN3Y
C           !       D/DX1(SIG_12)
C                   STR_DER(I,4)=STRESS((I-1)*G_XY+1,6)*NDN1X+
C                   + STRESS((I-1)*G_XY+2,6)*NDN2X+STRESS((I-1)*G_XY+3,6)*NDN4X+
C                   + STRESS((I-1)*G_XY+4,6)*NDN3X
C           !       D/DX2(SIG_22)
C                   STR_DER(I,5)=STRESS((I-1)*G_XY+1,2)*NDN1Y+
C                   + STRESS((I-1)*G_XY+2,2)*NDN2Y+STRESS((I-1)*G_XY+3,2)*NDN4Y+
C                   + STRESS((I-1)*G_XY+4,2)*NDN3Y

                 RHO_A_I(I,1)=RHO*(CENT_ACC(1,1)+INT_COORD(I,3)*CENT_ACC
     +                (2,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(3,1)+
     +                INT_COORD(I,3)**3.0D0*CENT_ACC(4,1))
```

```
                  RHO_A_I(I,2)=RHO*(CENT_ACC(5,1)+INT_COORD(I,3)*CENT_ACC
     +                 (6,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(7,1)+
     +                 INT_COORD(I,3)**3.0D0*CENT_ACC(8,1))
                  RHO_A_I(I,3)=RHO*(CENT_ACC(9,1)+INT_COORD(I,3)*CENT_ACC
     +                 (10,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(11,1)+
     +                 INT_COORD(I,3)**3.0D0*CENT_ACC(12,1))



                   STR_DER(I,3)=RHO_A_I(I,1)-STR_DER(I,1)-STR_DER(I,2)
                   STR_DER(I,6)=RHO_A_I(I,2)-STR_DER(I,4)-STR_DER(I,5)




                   S23_COL(I,1)=STR_DER(I,6)
                   S13_COL(I,1)=STR_DER(I,3)

C                    WRITE(152,*)
C                    WRITE(152,*) IEL
C                    WRITE(152,20) X_CENT,Y_CENT,INT_COORD((I-1)*G_XY+1,3)
C                    WRITE(152,20) (STR_DER(I,J),J=1,9)




!                    INTEGRATION OF STRESS GRADIENTS
C !                  SIGMA_23
c                    STR_TEMP(1)=STR_TEMP(1)+Z_W(I)*STR_DER(I,6)
C !                  SIGMA_13
C                    STR_TEMP(2)=STR_TEMP(2)+Z_W(I)*STR_DER(I,3)
C                    STR_TEMP(3)=STR_TEMP(3)+Z_W(I)*STR_DER(I,9)




                  ENDDO    !END OF THE LOOP I=1,G_Z



C                 STRESS_OUT(THICK_P,10)=STR_TEMP(1)*H/DIVS/2.0D0*(THICK_P-1.0D0)
C                 STRESS_OUT(THICK_P,11)=STR_TEMP(2)*H/DIVS/2.0D0*(THICK_P-1.0D0)
C                 STRESS_OUT(THICK_P,12)=STR_TEMP(3)*H/DIVS/2.0D0*(THICK_P-1.0D0)
                  S23_VAL=MATMUL(W_COL,S23_COL)
                  S13_VAL=MATMUL(W_COL,S13_COL)


C                   S23_TEMP(THICK_P2,N_COUNT)=S23_VAL
C                   S13_TEMP(THICK_P2,N_COUNT)=S23_VAL
C                   STR_TRAN(THICK_P2,1,N_COUNT)=(Z_INT(THICK_P2)-GLOB_Z_P(1))
C       +           /2.0D0*S23_VAL(1,1)
C                   STR_TRAN(THICK_P2,2,N_COUNT)=(Z_INT(THICK_P2)-GLOB_Z_P(1))
C       +           /2.0D0*S13_VAL(1,1)


                  S23_R(THICK_P2,N_COUNT)=(Z_INT(THICK_P2)-GLOB_Z_P(1))
     +            /2.0D0*S23_VAL(1,1)
                  S13_R(THICK_P2,N_COUNT)=(Z_INT(THICK_P2)-GLOB_Z_P(1))
     +            /2.0D0*S13_VAL(1,1)

C                ENDIF    ! END OF IF THICK_P2>1
              ENDDO !!!END OF THICK_P2 LOOP
            ENDDO    !END OF LOOP FOR 9 ELEMENTS (N_COUNT)


C           ID1=(IELY-2)*N_L+IELX-1
C           ID2=(IELY-2)*N_L+IELX
```

```
C         ID3=(IELY-2)*N_L+IELX+1
C         ID4=(IELY-1)*N_L+IELX-1
C         ID5=(IELY-1)*N_L+IELX
C         ID6=(IELY-1)*N_L+IELX+1
C         ID7=(IELY)*N_L+IELX-1
C         ID8=(IELY)*N_L+IELX
C         ID9=(IELY)*N_L+IELX+1
C         ID_SET=(/ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9/)

C         DO J=1,9
C           XX(J)=(GLOB_XYZ(CON(ID_SET(J),1),1)+
C      +    GLOB_XYZ(CON(ID_SET(J),2),1)+GLOB_XYZ(CON(ID_SET(J),3),1)
C      +    +GLOB_XYZ(CON(ID_SET(J),4),1))/4.0D0
C           YY(J)=(GLOB_XYZ(CON(ID_SET(J),1),2)+
C      +    GLOB_XYZ(CON(ID_SET(J),2),2)+GLOB_XYZ(CON(ID_SET(J),3),2)
C      +    +GLOB_XYZ(CON(ID_SET(J),4),2))/4.0D0
C           COEFF_MAT(J,1)=1.0D0
C           COEFF_MAT(J,2)=XX(J)
C           COEFF_MAT(J,3)=YY(J)
C           COEFF_MAT(J,4)=XX(J)**2.0D0
C           COEFF_MAT(J,5)=XX(J)*YY(J)
C           COEFF_MAT(J,6)=YY(J)**2.0D0
C           COEFF_MAT(J,7)=XX(J)**2.0D0*YY(J)
C           COEFF_MAT(J,8)=YY(J)**2.0D0*XX(J)
C           COEFF_MAT(J,9)=XX(J)**2.0D0*YY(J)**2.0D0
C         ENDDO
C         N_POINT=9   !DEGREE OF THE MATRIX
C         LWORK=20    !QUANTITIES FOR THE INVERSION SUBROUTINE
C         INFO=0
C         ALLOCATE(WORK(LWORK))
C         LU_COEFF_MAT=COEFF_MAT
CCC      CALLING LAPACK FOR LU FACTORIZATION OF COEFF MAT
C          CALL DGETRF(N_POINT,N_POINT,LU_COEFF_MAT,N_POINT,IPIV,INFO)
CCC      INITIATIALIZING MAT_INV AS LU_COEFF_MAT
C          MAT_INV=LU_COEFF_MAT
CCC      CALLING GENERAL MATRIX INVERSION LAPACK SUBROUTINE
C          CALL DGETRI(N_POINT,MAT_INV,N_POINT,IPIV,WORK,LWORK,INFO)
CCC      RESIDUE FROM THE MATRIX INVERSION
C          RES=MATMUL(COEFF_MAT,MAT_INV)

C         DEALLOCATE(WORK)


C         DO THICK_P=1,G_Z

C              IF (THICK_P==1) THEN
C                STRESS_OUT(THICK_P,12)=S33_B
C              ELSE

C                Z_CENT=(GLOB_Z_P(1)+GLOB_Z_P(THICK_P))/2.0D0
CC          WRITE(*,*) 'THICK_P',THICK_P
CC          WRITE(*,*) 'Z_CENT'
CC          WRITE(*,21) Z_CENT
C               DO J =1,G_Z
C                 Z_INT(J)= Z_CENT+H/DIVS/2.0D0*(THICK_P-1.0D0)*Z_P(J)
C               ENDDO

C               DO I=1,12
C                 CENT_ACC(I,1)=(NODAL_ACC((I-1)*4+1,1)+NODAL_ACC((I-1)
C      +          *4+2,1)+NODAL_ACC((I-1)*4+3,1)+NODAL_ACC((I-1)*4+4,1))
C      +          /4.0D0
C             ENDDO

C            DO I=1,9
C                RHS23(K)=STR_TRAN(THICK_P,1,K)
C                RHS13(K)=STR_TRAN(THICK_P,2,K)
C            ENDDO
```

```
C            CF13=MATMUL(MAT_INV,RHS13)
C            CF23=MATMUL(MAT_INV,RHS23)

C            STR_DER(THICK_P,7)=CF11(2)+2.0D0*CF11(4)*X_CENT+CF11(5)
C      +     *Y_CENT+CF11(7)*2.0D0*X_CENT*Y_CENT+CF11(8)*
C      +     Y_CENT**2.0D0+2.0D0*CF11(9)*X_CENT*Y_CENT**2.0D0

C            RHO_A_I(I,3)=RHO*(CENT_ACC(9,1)+INT_COORD(I,3)*CENT_ACC
C      +     (10,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(11,1)+
C      +     INT_COORD(I,3)**3.0D0*CENT_ACC(12,1))
C            ENDIF
C          ENDDO




      FORMAT (EN11.2)
      FORMAT (999(2x,EN14.4))
      FORMAT (999(x,EN11.2))
      FORMAT (999(X,I5))
      END SUBROUTINE TRANS_NOR

      SUBROUTINE SRS_NLIN(N_FEM,T_P,IELX,IELY,IEL,LE,BE,H
     + ,X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,SOL,CON,CC,I_ORDER,LOCAL_C
     + ,X_CENT,Y_CENT,G_XY,G_Z,NODAL_ACC,RHO,Z_P,Z_W,STRESS_OUT,N_NODE,
     + GLOB_Z_P,COORDG,COORDL,TOT_DOF,E_DOF,NODAL_DISP,N_L,N_B,GLOB_XYZ)
      USE PROG_DEBUG
      IMPLICIT NONE
      INTEGER :: N_FEM,T_P,I,J,K,IEL,IELX,IELY,G_XY,G_Z,TOT_DOF,E_DOF,
     + N_L,N_B,N_NODE
      REAL(KIND=8) :: STRESS(T_P,9,9), STRAIN(T_P,6,9),STRAIN_COL(6)
     +               ,S_COL(6),STRESS_MAT(3,3,9)
      REAL(KIND=8), ALLOCATABLE :: STRESS_TEMP(:,:),COORD_TEMP(:,:)
      REAL(KIND=8) :: X_I,X_J,X_K,X_L,Y_I,Y_J,Y_K,Y_L,GN1,GN2,GN3,GN4
      REAL(KIND=8) :: N1,N2,N3,N4,
     + DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,LE,BE,DET_J,H
      REAL(KIND=8) :: NDN1X,NDN2X,NDN3X,NDN4X,NDN1Y,NDN2Y,NDN3Y,NDN4Y
C     VALUES OF SHAPE FUNC AND ITS DERIV. AT THE GAUSS POINTS
      REAL(KIND=8) :: B(9,48),NODAL_ACC(E_DOF,1),NODAL_DISP(E_DOF,1),
     +              F_COL(9,9),DEF_GRAD(3,3,9),DISP_GRAD(3,3,9)
     +              ,F_INV(3,3,9),IDEN(3,3),JACOB(9),S(3,3),T(3,3,9),
     +               SIG(3,3,9)
      REAL(KIND=8) :: TEMP(6,9)
      REAL(KIND=8) :: CC(6,6),RHO
      REAL(KIND=8) :: LOCAL_C(T_P,3)
      INTEGER :: I_ORDER
C     DERIVATIVES OF THE COMPONENTS OF STRESSES AT DIFF. HEIGHTS
      REAL(KIND=8) :: STR_DER(G_Z,9),STRESS_OUT(G_Z,27)
      REAL(KIND=8) :: STR_TEMP(3),STR_VAL
C     CENTROIDAL COORDINATES
      REAL(KIND=8) :: X_CENT,Y_CENT
      REAL(KIND=8) :: CENT_ACC(12,1)
C     RHO X ACCELERATION_i
      REAL(KIND=8) :: RHO_A_I(G_Z,3)
      INTEGER :: THICK_P,DIVS !COUNT OF THICKNESS POINTS AND NO OF DIV.
      REAL(KIND=8) :: S13_B, S23_B, S33_B
      REAL(KIND=8) :: Z_P(G_Z),Z_W(G_Z),Z_INT(G_Z)
      REAL(KIND=8) :: INT_COORD(T_P,3),LINT_COORD(T_P,3)
      REAL(KIND=8) :: COORDG(G_XY,2),COORDL(G_XY,2),GLOB_Z_P(G_Z)
      REAL(KIND=8) :: Z_CENT
      REAL(KIND=8) :: S13_COL(G_Z,1),S23_COL(G_Z,1),S33_COL(G_Z,1),
     +               S31_COL(G_Z,1),S32_COL(G_Z,1)
      REAL(KIND=8) :: W_COL(1,G_Z)
      REAL(KIND=8) :: S32_VAL(1,1),S31_VAL(1,1),S33_VAL(1,1)
      REAL(KIND=8) :: LOC_RS(2,1)  !LOC. CORRDINATES OF THE CENTROID
      REAL(KIND=8) :: GLE_X,GLE_Y  !GLOB. X,Y DIST. BETWEEN GAUSS PTS.
      REAL(KIND=8) :: SOL(TOT_DOF)
      INTEGER :: CON(N_FEM,E_DOF)
```

```
      REAL(KIND=8) :: DISP(E_DOF,9),ELE_ACC(E_DOF,1),GLOB_XYZ(N_NODE,2)
      REAL(KIND=8) :: XX(9),YY(9)    !X,Y COORDS OF 9 ELE. CENTS.
      INTEGER :: ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9,ID_SET(9)
      REAL(KIND=8) :: COEFF_MAT(9,9),RHS11(9),RHS21(9),RHS12(9)
     +               ,RHS22(9),RHS23(9),RHS13(9),MAT_INV(9,9)
      REAL(KIND=8) :: CF11(9),CF21(9),CF12(9),CF22(9),CF13(9),CF23(9)

C     PARAMETERS FOR MATRIX INVERSION SUBROUTINE DGETRI.F
      INTEGER :: IPIV(9),INFO,LWORK,N_POINT   !N_POINT IS DIM.
      REAL(KIND=8),ALLOCATABLE :: LU_COEFF_MAT(:,:), RES(:,:)
C     LU_COEFF_MAT IS THE LU FACTORIZED FORM OF COEFF MAT
C     WHERE COEFF_MAT=P*L*U
      REAL(KIND=8),ALLOCATABLE :: WORK(:)
      REAL(KIND=8) :: F_COL2(9,1),DISP_GRAD2(3,3),DEF_GRAD2(3,3),T2(3,3)


C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     SPECIFYING THE BOUNDARY CONDITIONS AT THE BOTTOM LAYER
C     LATER TO BE INCLUDED IN THE INPUT FILE
C     BOUNDARY CONDITIONS RESPRESETING 1ST PK
C-----------------------------------------------------------------------
      S13_B=0.0D0
      S23_B=0.0D0
      S33_B=0.0D0

C-----------------------------------------------------------------------
C     INITIALIZING THE MATRIX OF COEFFS OF LEAST SQUARE
C-----------------------------------------------------------------------
      COEFF_MAT=0.0D0
      MAT_INV=0.0D0   !INVERSE OF A MATRIX


C-----------------------------------------------------------------------
C     IDENTITY MATRIX (3X3)
C-----------------------------------------------------------------------
      IDEN=0.0D0
      IDEN(1,1)=1.0D0;IDEN(2,2)=1.0D0;IDEN(3,3)=1.0D0



C-----------------------------------------------------------------------
C     GAUSS POINTS AND WEIGHTS IN Z
C-----------------------------------------------------------------------
      CALL GAUSS_LEG(G_Z,Z_P,Z_W)



C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C     ALLOCATING THE Z WEIGHTS IN A ROW VECTOR
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
      DO I=1,G_Z
          W_COL(1,I)=Z_W(I)
      ENDDO

C-----------------------------------------------------------------------
C     IDENTIFYING THE ELEMENT TO WORK ON
C-----------------------------------------------------------------------
C     DEFINING THE ID NUMBER FOR ELEMENTS IN THE 9 PATCH FOR SRS
C     THE BOUNDARY ELEMENTS ARE EXCLUDED
C     ORDER OF ELEMENT NUMBER IS BOTTOM TO TOP, LEFT TO RIGHT
C     | 7 | 8 | 9 |
C     | 4 | 5 | 6 |
C     | 1 | 2 | 3 |
C     THE POINT OF INTEREST IS THE CENTROID OF THE CENTRAL ELE (5)
C-----------------------------------------------------------------------
```

```
        IF (IELX .GT. 1 .AND. IELX .LT. N_L .AND. IELY
     + .GT. 1 .AND. IELY .LT. N_B) THEN    !STATEMENTS TO ID ALL THE INT. ELE.

C          FOR DEBUGGING, SCREEN OUTPUTS
C          WRITE(*,*) IELY, IELX
C          WRITE(*,*) 'E_DOF',E_DOF
C          WRITE(*,*) 'TOT_DOF',TOT_DOF
C          DO I=1,N_FEM
C               WRITE(165,23) (CON(I,J),J=1,E_DOF)
C          ENDDO
            ID1=(IELY-2)*N_L+IELX-1
            ID2=(IELY-2)*N_L+IELX
            ID3=(IELY-2)*N_L+IELX+1
            ID4=(IELY-1)*N_L+IELX-1
            ID5=(IELY-1)*N_L+IELX
            ID6=(IELY-1)*N_L+IELX+1
            ID7=(IELY)*N_L+IELX-1
            ID8=(IELY)*N_L+IELX
            ID9=(IELY)*N_L+IELX+1
C-----------------------------------------------------------------------
C           ID_SET DEFINES THE ID OF THE 9 ELEMENTS IN ORDER
C-----------------------------------------------------------------------
            ID_SET=(/ID1,ID2,ID3,ID4,ID5,ID6,ID7,ID8,ID9/)


C-----------------------------------------------------------------------
C          LOOP OVER THE 9 ELEMENTS OF THE PATCH TO DEFINE CENTROIDS AND
C          THE COEFFICIENT MATRIX FOR THE 2ND DEGREE POLY FIT
C-----------------------------------------------------------------------
           DO J=1,9
C-----------------------------------------------------------------------
C          DEFINING THE CENTROIDS OF EACH OF THE 9 ELEMENTS IN THE PATCH
C-----------------------------------------------------------------------
            XX(J)=(GLOB_XYZ(CON(ID_SET(J),1),1)+
     +       GLOB_XYZ(CON(ID_SET(J),2),1)+GLOB_XYZ(CON(ID_SET(J),3),1)
     +       +GLOB_XYZ(CON(ID_SET(J),4),1))/4.0D0
            YY(J)=(GLOB_XYZ(CON(ID_SET(J),1),2)+
     +       GLOB_XYZ(CON(ID_SET(J),2),2)+GLOB_XYZ(CON(ID_SET(J),3),2)
     +       +GLOB_XYZ(CON(ID_SET(J),4),2))/4.0D0
            COEFF_MAT(J,1)=1.0D0
            COEFF_MAT(J,2)=XX(J)
            COEFF_MAT(J,3)=YY(J)
            COEFF_MAT(J,4)=XX(J)**2.0D0
            COEFF_MAT(J,5)=XX(J)*YY(J)
            COEFF_MAT(J,6)=YY(J)**2.0D0
            COEFF_MAT(J,7)=XX(J)**2.0D0*YY(J)
            COEFF_MAT(J,8)=YY(J)**2.0D0*XX(J)
            COEFF_MAT(J,9)=XX(J)**2.0D0*YY(J)**2.0D0
           ENDDO

C!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C-----------------------------------------------------------------------
C          STEPS FOR MATRIX INVERSION OF THE COEFFICENT MATRIX
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

           N_POINT=9    !ORDER OF THE MATRIX
           LWORK=20     !INTRINSIC QUANTITIES FOR THE INVERSION SUBROUTINE
           INFO=0
           ALLOCATE(WORK(LWORK),LU_COEFF_MAT(9,9), RES(9,9))
           LU_COEFF_MAT=COEFF_MAT
C          CALLING LAPACK FOR LU FACTORIZATION OF COEFF MAT
           CALL DGETRF(N_POINT,N_POINT,LU_COEFF_MAT,N_POINT,IPIV,INFO)
C          INITIATIALIZING MAT_INV AS LU_COEFF_MAT
           MAT_INV=LU_COEFF_MAT
C          CALLING GENERAL MATRIX INVERSION LAPACK SUBROUTINE
           CALL DGETRI(N_POINT,MAT_INV,N_POINT,IPIV,WORK,LWORK,INFO)
C          RESIDUE FROM THE MATRIX INVERSION
           RES=MATMUL(COEFF_MAT,MAT_INV)
```

```
8486            DEALLOCATE(LU_COEFF_MAT,RES)
8487
8488
8489    C-----------------------------------------------------------------------
8490    C         CHECKING THE RESULT OF MAT INVERSION
8491    C-----------------------------------------------------------------------
8492    C         UN-COMMENT IF NEEDED TO CHECK
8493    C-----------------------------------------------------------------------
8494    C           DO I=1,9
8495    C               WRITE(1,20) (RES(I,J),J=1,9)
8496    C           ENDDO
8497    C           WRITE(166,*) 'IEL',IEL
8498    C           WRITE(166,*) 'MATRIX'
8499    C           DO I=1,9
8500    C               WRITE(166,22) (COEFF_MAT(I,J),J=1,9)
8501    C           ENDDO
8502    C           WRITE(166,*) 'INVERSE'
8503    C           DO I=1,9
8504    C               WRITE(166,22) (MAT_INV(I,J),J=1,9)
8505    C           ENDDO
8506            DEALLOCATE(WORK)
8507    C!!!!!!!!!!!! INVERSION STEP COMPLETE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
8508
8509
8510
8511
8512
8513    C-----------------------------------------------------------------------
8514    C         CALLING THE NODAL DISPLACEMENTS OF EACH OF THE 9 ELEMENTS
8515    C         INDICES FOLLOW THE ORDER DISP(DEGREE OF FREEDOM, ELEMENT NO.)
8516    C         SOL IS THE VECTOR OF GLOBAL DISPLACEMENTS OF ALL NODES TRANSF-
8517    C         ERRED FROM THE MAIN PROGRAM
8518    C         ORDER OF ELEMENT NUMBER IS BOTTOM TO TOP, LEFT TO RIGHT
8519    C         | 7 | 8 | 9 |
8520    C         | 4 | 5 | 6 |
8521    C         | 1 | 2 | 3 |
8522    C         THE POINT OF INTEREST IS THE CENTROID OF TEH CENTRAL ELE (5)
8523    C         CON IS THE CONNECTIVITY MATRIX
8524    C-----------------------------------------------------------------------
8525            DO J=1,E_DOF
8526              DISP(J,1)=SOL(CON((IELY-2)*N_B+IELX-1,J))
8527              DISP(J,2)=SOL(CON((IELY-2)*N_B+IELX,J))
8528              DISP(J,3)=SOL(CON((IELY-2)*N_B+IELX+1,J))
8529              DISP(J,4)=SOL(CON((IELY-1)*N_B+IELX-1,J))
8530              DISP(J,5)=SOL(CON((IELY-1)*N_B+IELX,J))
8531              DISP(J,6)=SOL(CON((IELY-1)*N_B+IELX+1,J))
8532              DISP(J,7)=SOL(CON((IELY)*N_B+IELX-1,J))
8533              DISP(J,8)=SOL(CON((IELY)*N_B+IELX,J))
8534              DISP(J,9)=SOL(CON((IELY)*N_B+IELX+1,J))
8535            ENDDO
8536    C           DO J=1,E_DOF
8537    C             ELE_ACC(J,1)=NODAL_ACC(CON(IEL,J),1)
8538    C           ENDDO
8539
8540    C             WRITE(*,*) 'FOR ELEMENT',IEL, 'NEIGHBOR LIST'
8541    C             WRITE(*,*) (IELY-2)*N_B+IELX-1
8542    C             WRITE(*,*) (IELY-2)*N_B+IELX
8543    C             WRITE(*,*) (IELY-2)*N_B+IELX+1
8544    C             WRITE(*,*) (IELY-1)*N_B+IELX-1
8545    C             WRITE(*,*) (IELY-1)*N_B+IELX
8546    C             WRITE(*,*) (IELY-1)*N_B+IELX+1
8547    C             WRITE(*,*) (IELY)*N_B+IELX-1
8548    C             WRITE(*,*) (IELY)*N_B+IELX
8549    C             WRITE(*,*) (IELY)*N_B+IELX+1
8550    C             DO I=1,G_Z
8551    C                 W_COL(1,I)=Z_W(I)
8552    C             ENDDO
8553    C         ALLOCATE(STRESS_TEMP(T_P,3),COORD_TEMP(T_P,3))
8554    ! C        LOGICAL IDENTIFIER DEBUG AND OUTPUT DEFINED IN MODULES.F
```

```
8555
8556  ! C        INTEGTRATION POINT FOR THICKNESS INTEGRATION IN RECOVERY FOR EACH
8557  ! C        POINT IN Z
8558
8559
8560  C-----------------------------------------------------------------------
8561  C         DIVS IS THE NUMBER OF DIVISIONS IN HEIGHT = TOTAL POINTS-1
8562  C-----------------------------------------------------------------------
8563           DIVS=G_Z-1
8564
8565
8566  C        CHECKED
8567
8568                ALLOCATE(WORK(LWORK),LU_COEFF_MAT(3,3),
8569       +                 RES(3,3))
8570  C-----------------------------------------------------------------------
8571  !        START OF LOOP FOR RECOVERING STRESS AT EVERY POINT IN THICKNESS
8572  C        RECOVER THE STRESSES AT THE CENTROID LOCATION OF EACH ELEMENT
8573  C        FOR DIFFERENT HEIGHTS. THICK_P ARE THE POINTS IN HEIGHT
8574  C        STARTS AT THE BOTTOM SURFACE AT THICK_P=1
8575  C-----------------------------------------------------------------------
8576  C-----------------------------------------------------------------------
8577  C-----------------------------------------------------------------------
8578
8579           DO THICK_P = 1,G_Z
8580
8581  C-----------------------------------------------------------------------
8582  C             BOTTOM SURFACE TRANSVERSE STRESSES = BOUNDARY CONTIONS
8583  C             IN NON-LINEAR HAS TO BE DEPENDENT OF F. FOR NOW FREE BOTTOM
8584  C             KEEPING THEM TO '0'
8585  C-----------------------------------------------------------------------

8586                IF (THICK_P==1) THEN
8587  C-----------------------------------------------------------------------
8588  C             BOTTOM SURFACE TRANSVERSE STRESSES = BOUNDARY CONDITIONS
8589  C             IN NON-LINEAR, HAS TO BE DEPENDENT OF F
8590  C             WILL INCORPORATE LATER
8591  C-----------------------------------------------------------------------
8592                STRESS_OUT(THICK_P,10)=S23_B
8593                STRESS_OUT(THICK_P,11)=S13_B
8594                STRESS_OUT(THICK_P,12)=S33_B
8595
8596
8597
8598  C-----------------------------------------------------------------------
8599  C             FROM THICK_P=2
8600  C-----------------------------------------------------------------------
8601                ELSE
8602
8603
8604
8605
8606  C-----------------------------------------------------------------------
8607  C             Z COORDINATE OF THE CENTROID BETWEEN THE BOTTOM SURFACE
8608  C             AND THE CURRENT THICK_P POINT
8609  C             NEEDED FOR GLOBAL GAUSS POINT COORDINATES
8610  C-----------------------------------------------------------------------
8611                Z_CENT=(GLOB_Z_P(1)+GLOB_Z_P(THICK_P))/2.0D0
8612
8613
8614
8615  C-----------------------------------------------------------------------
8616  C             GLOBAL GAUSS POINT COORD. IN Z FOR INTEGRATING AT THICK_P
8617  C-----------------------------------------------------------------------
8618                DO J =1,G_Z
8619                    Z_INT(J)= Z_CENT+H/DIVS/2.0D0*(THICK_P-1.0D0)*Z_P(J)
8620                ENDDO
8621
8622  C             FOR DEBUGGING
```

```
C             SCREEN OUTPUT OF THE Z COORDS OF INT. POINTS FOR EACH Z_P
C             WRITE(*,*) THICK_P,'    ',Z_INT
C-----------------------------------------------------------------------
C             LOCAL AND GLOBAL COORD. OF GAUSS POINTS TOTAL OF 2 X 2 X G_Z
C-----------------------------------------------------------------------
              DO I =1,G_XY
                 DO J=1,G_Z
                    INT_COORD((J-1)*G_XY+I,1)=COORDG(I,1)
                    INT_COORD((J-1)*G_XY+I,2)=COORDG(I,2)
                    INT_COORD((J-1)*G_XY+I,3)=Z_INT(J)
                    LINT_COORD((J-1)*G_XY+I,1)=COORDL(I,1)
                    LINT_COORD((J-1)*G_XY+I,2)=COORDL(I,2)
                    LINT_COORD((J-1)*G_XY+I,3)=Z_INT(J)
                 ENDDO
              ENDDO



C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C             PROCEDURE
C             CALCULATE THE STRESSES AT THE 4 GAUSS POINTS AT EACH G_Z
C             AVERAGE THE VALUES OF STRESSES FROM THESE 4 GAUSS POINTS
C             GIVING THE CENTROIDAL STRESS AT EACH G_Z
C             INTEGRATE USING THIS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C             CALCULATION OF B FOR A GIVEN G_Z
C             B IS AN OPERATOR MATRIX WHICH OPERATED UPON DISPS
C             GIVES THE DISPLACEMENT GRADIENT MATRIX
C-----------------------------------------------------------------------
C
C-----------------------------------------------------------------------
              B=0.0D0
              DO I=1,T_P

                    CALL FEM_SHAPE_DIFF(LINT_COORD(I,1),LINT_COORD(I,2)
     +              ,I_ORDER,DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,
     +              DET_J,LE,BE)
C                   CALL FEM_SHAPE_GLOBAL(X_I,X_J,Y_I,Y_L,INT_COORD(I,1),
C       +           INT_COORD(I,2),GN1,GN2,GN3,GN4,LE,BE)
                    CALL FEM_SHAPE(LINT_COORD(I,1),LINT_COORD(I,2),
     +              I_ORDER,GN1,GN2,GN3,GN4,DET_J,LE,BE)
C                   DN1X=1; DN2X=1; DN3X=1; DN4X=1;
C                   DN1Y=2; DN2Y=2; DN3Y=2; DN4Y=2;
C                   GN1=3; GN2=3; GN3=3; GN4=3;



C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C                   B COMPONENTS
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
                    DO J=1,4
                           B(1,4*J)=DN4X*INT_COORD(I,3)**(J-1)
                           B(1,4*J-1)=DN3X*INT_COORD(I,3)**(J-1)
                           B(1,4*J-2)=DN2X*INT_COORD(I,3)**(J-1)
                           B(1,4*J-3)=DN1X*INT_COORD(I,3)**(J-1)
                           B(2,4*J+16)=DN4Y*INT_COORD(I,3)**(J-1)
                           B(2,4*J-1+16)=DN3Y*INT_COORD(I,3)**(J-1)
                           B(2,4*J-2+16)=DN2Y*INT_COORD(I,3)**(J-1)
                           B(2,4*J-3+16)=DN1Y*INT_COORD(I,3)**(J-1)

```

```
                                B(4,4*J+32)=DN4Y*INT_COORD(I,3)**(J-1)
                                B(4,4*J-1+32)=DN3Y*INT_COORD(I,3)**(J-1)
                                B(4,4*J-2+32)=DN2Y*INT_COORD(I,3)**(J-1)
                                B(4,4*J-3+32)=DN1Y*INT_COORD(I,3)**(J-1)

                                B(5,4*J+32)=DN4X*INT_COORD(I,3)**(J-1)
                                B(5,4*J-1+32)=DN3X*INT_COORD(I,3)**(J-1)
                                B(5,4*J-2+32)=DN2X*INT_COORD(I,3)**(J-1)
                                B(5,4*J-3+32)=DN1X*INT_COORD(I,3)**(J-1)

                                B(6,4*J)=DN4Y*INT_COORD(I,3)**(J-1)
                                B(6,4*J-1)=DN3Y*INT_COORD(I,3)**(J-1)
                                B(6,4*J-2)=DN2Y*INT_COORD(I,3)**(J-1)
                                B(6,4*J-3)=DN1Y*INT_COORD(I,3)**(J-1)

                                B(9,4*J+16)=DN4X*INT_COORD(I,3)**(J-1)
                                B(9,4*J-1+16)=DN3X*INT_COORD(I,3)**(J-1)
                                B(9,4*J-2+16)=DN2X*INT_COORD(I,3)**(J-1)
                                B(9,4*J-3+16)=DN1X*INT_COORD(I,3)**(J-1)
                        ENDDO
                        DO J=2,4
                                B(3,4*J+32)=(J-1)*N4*INT_COORD(I,3)**(J-2)
                                B(3,4*J-1+32)=(J-1)*N3*INT_COORD(I,3)**(J-2)
                                B(3,4*J-2+32)=(J-1)*N2*INT_COORD(I,3)**(J-2)
                                B(3,4*J-3+32)=(J-1)*N1*INT_COORD(I,3)**(J-2)

                                B(7,4*J+16)=(J-1)*N4*INT_COORD(I,3)**(J-2)
                                B(7,4*J-1+16)=(J-1)*N3*INT_COORD(I,3)**(J-2)
                                B(7,4*J-2+16)=(J-1)*N2*INT_COORD(I,3)**(J-2)
                                B(7,4*J-3+16)=(J-1)*N1*INT_COORD(I,3)**(J-2)

                                B(8,4*J)=(J-1)*N4*INT_COORD(I,3)**(J-2)
                                B(8,4*J-1)=(J-1)*N3*INT_COORD(I,3)**(J-2)
                                B(8,4*J-2)=(J-1)*N2*INT_COORD(I,3)**(J-2)
                                B(8,4*J-3)=(J-1)*N1*INT_COORD(I,3)**(J-2)
                        ENDDO


C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C                       GRADIENTS OF DISPLACEMENTS OF 9 ELEMENTS (DUi/DXj)
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------

                        F_COL=MATMUL(B,DISP)
C                       OPEN(4,FILE='STRAIN_VERIF.DAT')

C                       WRITE(825,*) 'F_COL'
C                       WRITE(825,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),
C     +                 ELE_STRAIN(I,3),TIME
C                       WRITE(825,20) F_COL



C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
C                       ASSEMBLING THE GRADIENT IN COLUMN FORM
C-----------------------------------------------------------------------
C-----------------------------------------------------------------------
                        DO K=1,9    ! K IS THE ELE. NO. IN THE 9 ELE PATCH
                                DISP_GRAD(1,1,K)=F_COL(1,K)
                                DISP_GRAD(2,2,K)=F_COL(2,K)
                                DISP_GRAD(3,3,K)=F_COL(3,K)
                                DISP_GRAD(3,2,K)=F_COL(4,K)
                                DISP_GRAD(3,1,K)=F_COL(5,K)
                                DISP_GRAD(1,2,K)=F_COL(6,K)
                                DISP_GRAD(2,3,K)=F_COL(7,K)
                                DISP_GRAD(1,3,K)=F_COL(8,K)
                                DISP_GRAD(2,1,K)=F_COL(9,K)
```

```
C-----------------------------------------------------------------------
C                           DEFORMATION GRADIENT = DISP. GRAD. + I
C-----------------------------------------------------------------------

                            DEF_GRAD(:,:,K)=DISP_GRAD(:,:,K)+IDEN(:,:)

                            JACOB(K)=DEF_GRAD(1,1,K)*(DEF_GRAD(2,2,K)*
     +                      DEF_GRAD(3,3,K)-DEF_GRAD(3,2,K)*
     +                      DEF_GRAD(2,3,K))-DEF_GRAD(1,2,K)*
     +                      (DEF_GRAD(2,1,K)*DEF_GRAD(3,3,K)-DEF_GRAD
     +                      (3,1,K)*DEF_GRAD(2,3,K))+DEF_GRAD(1,3,K)*
     +                      (DEF_GRAD(2,1,K)*DEF_GRAD(3,2,K)-
     +                      DEF_GRAD(3,1,K)*DEF_GRAD(2,2,K))




                            N_POINT=3   !DEGREE OF THE MATRIX 3X3
                            LWORK=20    !QUANTITIES FOR THE INVERSION SUBROUTINE
                            INFO=0


C                            ALLOCATE(WORK(LWORK),LU_COEFF_MAT(3,3),
C     +                       RES(3,3))
                            LU_COEFF_MAT(:,:)=DEF_GRAD(:,:,K)




C-----------------------------------------------------------------------
C                           CALLING LAPACK FOR LU FACTORIZATION OF COEFF MAT
C-----------------------------------------------------------------------
                            CALL DGETRF(N_POINT,N_POINT,LU_COEFF_MAT,
     +                      N_POINT,IPIV,INFO)


C-----------------------------------------------------------------------
C                           INITIATIALIZING MAT_INV AS LU_COEFF_MAT
C-----------------------------------------------------------------------
                            F_INV(:,:,K)=LU_COEFF_MAT(:,:)



C-----------------------------------------------------------------------
C                           CALLING GENERAL MATRIX INVERSION LAPACK SUBROUTINE
C-----------------------------------------------------------------------
                            CALL DGETRI(N_POINT,F_INV(:,:,K),N_POINT,
     +                      IPIV,WORK,LWORK,INFO)



C-----------------------------------------------------------------------
C                           RESIDUE FROM THE MATRIX INVERSION
C-----------------------------------------------------------------------
C                             RES=MATMUL(DEF_GRAD(:,:,K),F_INV(:,:,K))
C                             DEALLOCATE(LU_COEFF_MAT,RES)

C-----------------------------------------------------------------------

C                           GREEN ST. VENANT STRAIN
C                           E=1/2*(F^T*f-I)
C-----------------------------------------------------------------------

                            STRAIN(:,:,K)=1.0D0/2.0D0*(MATMUL(TRANSPOSE
     +                      (DEF_GRAD(:,:,K)),DEF_GRAD(:,:,K))-IDEN)

```

```
8828
8829
8830   C-----------------------------------------------------------------------
8831   C                             STORING STRAIN IN COLUMN FORM FOR STRESS CAL
8832   C-----------------------------------------------------------------------
8833                                 STRAIN_COL(1)=STRAIN(1,1,K)
8834                                 STRAIN_COL(2)=STRAIN(2,2,K)
8835                                 STRAIN_COL(3)=STRAIN(3,3,K)
8836                                 STRAIN_COL(4)=2.0D0*STRAIN(2,3,K)
8837                                 STRAIN_COL(5)=2.0D0*STRAIN(1,3,K)
8838                                 STRAIN_COL(6)=2.0D0*STRAIN(1,2,K)
8839   C-----------------------------------------------------------------------
8840   C                             STRESS IN COLUMN FORM (2ND PK STRESS)
8841   C-----------------------------------------------------------------------
8842                                 S_COL=MATMUL(CC,STRAIN_COL)
8843
8844
8845
8846   C-----------------------------------------------------------------------
8847   C                             2ND PK STRESS IN 3X3 MATRIX FORM
8848   C-----------------------------------------------------------------------
8849                                 S(1,1)=S_COL(1)
8850                                 S(2,2)=S_COL(2)
8851                                 S(3,3)=S_COL(3)
8852                                 S(1,2)=S_COL(6)
8853                                 S(1,3)=S_COL(5)
8854                                 S(2,3)=S_COL(4)
8855                                 S(2,1)=S(1,2)
8856                                 S(3,1)=S(1,3)
8857                                 S(3,2)=S(2,3)
8858
8859   C-----------------------------------------------------------------------
8860   C                             1st PK STRESSES OF ALL 9 ELEMENTS ARE STORED
8861   C-----------------------------------------------------------------------
8862
8863                                 T(:,:,K)=MATMUL(DEF_GRAD(:,:,K),S)
8864   C                             SIG=1.0D0/JACOB*MATMUL(T,TRANSPOSE(DEF_GRAD))
8865
8866
8867
8868                                 T(:,:,K)=TRANSPOSE(T(:,:,K))
8869   C-----------------------------------------------------------------------
8870   C                             CAUCHY STRESS OF 9 ELEMENTS
8871   C-----------------------------------------------------------------------

8872                                 SIG(:,:,K)=1.0D0/JACOB(K)*
8873        +                        MATMUL(DEF_GRAD(:,:,K),T(:,:,K))
8874
8875
8876
8877   C-----------------------------------------------------------------------
8878   C-----------------------------------------------------------------------
8879   C                             STRESSES IN ORDERS SIG -
       11,22,33,23,13,12,32,31,12
8880   C-----------------------------------------------------------------------
8881   C-----------------------------------------------------------------------
8882                                 STRESS(I,1,K)=SIG(1,1,K)
8883                                 STRESS(I,2,K)=SIG(2,2,K)
8884                                 STRESS(I,3,K)=SIG(3,3,K)
8885                                 STRESS(I,4,K)=SIG(2,3,K)
8886                                 STRESS(I,5,K)=SIG(1,3,K)
8887                                 STRESS(I,6,K)=SIG(1,2,K)
8888                                 STRESS(I,7,K)=SIG(3,2,K)
8889                                 STRESS(I,8,K)=SIG(3,1,K)
8890                                 STRESS(I,9,K)=SIG(1,2,K)
8891                           ENDDO
8892
8893
8894
```

```
C                     IF (DEBUG) THEN
C                      OPEN(193,FILE='DEFORM_GRAD.DAT')
C                      OPEN(194,FILE='FIRST_PK.DAT')
C                       WRITE(193,*) 'COORDINATES'
C                       WRITE(193,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),ELE_
C      +                      STRAIN(I,3),TIME
C                            WRITE(193,*) 'DEFORMATION GRADIENT'
C                            DO J=1,3
C                             WRITE(193,20) (DEF_GRAD(J,K),K=1,3)
C                                  ENDDO
C                            WRITE(194,*) 'COORDINATES'
C                            WRITE(194,20) ELE_STRAIN(I,1),ELE_STRAIN(I,2),
C      +                     ELE_STRAIN(I,3), TIME
C                             WRITE(194,*) 'STRESS'
C                            DO J=1,3
C                              WRITE(194,20) (SIG(J,K),K=1,3)
C                            ENDDO
C                     ENDIF

                ENDDO   !END OF LOOP IN I = 1,T_P
                DO I=1,9
                   WRITE(4,20) (INT_COORD(K,1),INT_COORD(K,2),INT_COORD
       +            (K,3),STRAIN(K,1,I),STRAIN(K,2,I),STRAIN(K,3,I),
       +            STRAIN(K,4,I),STRAIN(K,5,I),STRAIN(K,6,I),K=1,T_P)
                ENDDO

                DO I=1,12
                  CENT_ACC(I,1)=(NODAL_ACC((I-1)*4+1,1)+NODAL_ACC((I-1)
       +          *4+2,1)+NODAL_ACC((I-1)*4+3,1)+NODAL_ACC((I-1)*4+4,1))
       +          /4.0D0
                ENDDO


                STR_TEMP=0.0D0
                DO I=1,G_Z
! C                    CALCULATIONS OF DERIVTIVES OF STRESSES
! C                    ORDER (I,J), J: 1=(S11,1),2=(S21,2),3=(S31,3)
! C                    4=(S12,1),5=(S22,2),6=(S32,3), 7=(S13,1),8=(S23,2),9=(S33,3)


C !                D/DX1(SIG_11)
                   DO K=1,9
                      RHS11(K)=(STRESS((I-1)*G_XY+1,1,K)+
       +              STRESS((I-1)*G_XY+2,1,K)+STRESS((I-1)*G_XY+3,1,K)
       +              +STRESS((I-1)*G_XY+4,1,K))/4.0D0

                      RHS21(K)=(STRESS((I-1)*G_XY+1,9,K)+
       +              STRESS((I-1)*G_XY+2,9,K)+STRESS((I-1)*G_XY+3,9,K)
       +              +STRESS((I-1)*G_XY+4,9,K))/4.0D0

                      RHS12(K)=(STRESS((I-1)*G_XY+1,6,K)+
       +              STRESS((I-1)*G_XY+2,6,K)+STRESS((I-1)*G_XY+3,6,K)
       +              +STRESS((I-1)*G_XY+4,6,K))/4.0D0

                      RHS22(K)=(STRESS((I-1)*G_XY+1,2,K)+
       +              STRESS((I-1)*G_XY+2,2,K)+STRESS((I-1)*G_XY+3,2,K)
       +              +STRESS((I-1)*G_XY+4,2,K))/4.0D0
                   ENDDO
C                    OPEN(9,FILE='STRESS_VERIF.DAT')
C                    WRITE(9,*) 'PATCH FOR',IEL
C                    WRITE(9,*) 'Z LEVEL',THICK_P, 'G_Z',I
C                    WRITE(9,20) X_CENT,Y_CENT,Z_INT(I)
C                    DO K=1,9
C                           WRITE(9,20) XX(K),YY(K),(RHS11(J),J=1,9)
C                           WRITE(9,20) XX(K),YY(K),(RHS21(J),J=1,9)
```

```
C                         WRITE(9,20) XX(K),YY(K),(RHS12(J),J=1,9)
C                         WRITE(9,20) XX(K),YY(K),(RHS22(J),J=1,9)
C                       ENDDO

                        CF11=MATMUL(MAT_INV,RHS11)
                        CF21=MATMUL(MAT_INV,RHS21)
                        CF12=MATMUL(MAT_INV,RHS12)
                        CF22=MATMUL(MAT_INV,RHS22)

                        STR_DER(I,1)=CF11(2)+2.0D0*CF11(4)*X_CENT+CF11(5)
     +                     *Y_CENT+CF11(7)*2.0D0*X_CENT*Y_CENT+CF11(8)*
     +                     Y_CENT**2.0D0+2.0D0*CF11(9)*X_CENT*Y_CENT**2.0D0

                        STR_DER(I,2)=CF21(3)+CF21(5)*X_CENT+2*CF21(6)
     +                     *Y_CENT+CF21(7)*X_CENT**2.0D0+2.0D0*CF21(8)*
     +                     X_CENT*Y_CENT+2.0D0*CF21(9)*X_CENT**2.0D0*
     +                     Y_CENT

                        STR_DER(I,4)=CF12(2)+2.0D0*CF12(4)*X_CENT+CF12(5)
     +                     *Y_CENT+CF12(7)*2.0D0*X_CENT*Y_CENT+CF12(8)*
     +                     Y_CENT**2.0D0+2.0D0*CF12(9)*X_CENT*Y_CENT**2.0D0

                        STR_DER(I,5)=CF22(3)+CF22(5)*X_CENT+2*CF22(6)
     +                     *Y_CENT+CF22(7)*X_CENT**2.0D0+2.0D0*CF22(8)*
     +                     X_CENT*Y_CENT+2.0D0*CF22(9)*X_CENT**2.0D0
     +                     *Y_CENT

C                       OPEN(5,FILE='STRESS_GRAD_VER.DAT')
C                       WRITE(5,*) 'IEL -',IEL,'POINT THICK_P -', THICK_P,
C      +                 'G_Z',I
C                       WRITE(5,20) X_CENT,Y_CENT,Z_INT(I),(STR_DER(I,J),J=1,
C      +                 5)




C                       STR_DER(I,1)=STRESS((I-1)*G_XY+1,1)*NDN1X+
C      +                STRESS((I-1)*G_XY+2,1)*NDN2X+STRESS((I-1)*G_XY+3,1)*NDN4X+
C      +                STRESS((I-1)*G_XY+4,1)*NDN3X
C        !              D/DX2(SIG_12)
C                       STR_DER(I,2)=STRESS((I-1)*G_XY+1,6)*NDN1Y+
C                       +STRESS((I-1)*G_XY+2,6)*NDN2Y+STRESS((I-1)*G_XY+3,6)*NDN4Y+
C                       + STRESS((I-1)*G_XY+4,6)*NDN3Y
C        !              D/DX1(SIG_12)
C                       STR_DER(I,4)=STRESS((I-1)*G_XY+1,6)*NDN1X+
C                       + STRESS((I-1)*G_XY+2,6)*NDN2X+STRESS((I-1)*G_XY+3,6)*NDN4X+
C                       + STRESS((I-1)*G_XY+4,6)*NDN3X
C        !              D/DX2(SIG_22)
C                       STR_DER(I,5)=STRESS((I-1)*G_XY+1,2)*NDN1Y+
C                       + STRESS((I-1)*G_XY+2,2)*NDN2Y+STRESS((I-1)*G_XY+3,2)*NDN4Y+
C                       + STRESS((I-1)*G_XY+4,2)*NDN3Y

                      RHO_A_I(I,1)=RHO*(CENT_ACC(1,1)+INT_COORD(I,3)*CENT_ACC
     +                     (2,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(3,1)+
     +                     INT_COORD(I,3)**3.0D0*CENT_ACC(4,1))
                      RHO_A_I(I,2)=RHO*(CENT_ACC(5,1)+INT_COORD(I,3)*CENT_ACC
     +                     (6,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(7,1)+
     +                     INT_COORD(I,3)**3.0D0*CENT_ACC(8,1))
                      RHO_A_I(I,3)=RHO*(CENT_ACC(9,1)+INT_COORD(I,3)*CENT_ACC
     +                     (10,1)+INT_COORD(I,3)**2.0D0*CENT_ACC(11,1)+
     +                     INT_COORD(I,3)**3.0D0*CENT_ACC(12,1))




                      STR_DER(I,3)=RHO_A_I(I,1)-STR_DER(I,1)-STR_DER(I,2)
                      STR_DER(I,6)=RHO_A_I(I,2)-STR_DER(I,4)-STR_DER(I,5)
```

```
                  S23_COL(I,1)=STR_DER(I,6)
                  S13_COL(I,1)=STR_DER(I,3)



C                   WRITE(152,*)
C                   WRITE(152,*) IEL
C                   WRITE(152,20) X_CENT,Y_CENT,INT_COORD((I-1)*G_XY+1,3)
C                   WRITE(152,20) (STR_DER(I,J),J=1,9)



! C                  INTEGRATION OF STRESS GRADIENTS
C !                T23
c                  STR_TEMP(1)=STR_TEMP(1)+Z_W(I)*STR_DER(I,6)
C !                T13
C                  STR_TEMP(2)=STR_TEMP(2)+Z_W(I)*STR_DER(I,3)
C                  T33 (NOT READY)
C                  STR_TEMP(3)=STR_TEMP(3)+Z_W(I)*STR_DER(I,9)

               ENDDO    !END OF THE LOOP I=1,G_Z



C                STRESS_OUT(THICK_P,10)=STR_TEMP(1)*H/DIVS/2.0D0*(THICK_P-1.0D0)
C                STRESS_OUT(THICK_P,11)=STR_TEMP(2)*H/DIVS/2.0D0*(THICK_P-1.0D0)
C                STRESS_OUT(THICK_P,12)=STR_TEMP(3)*H/DIVS/2.0D0*(THICK_P-1.0D0)
                 S32_VAL=MATMUL(W_COL,S23_COL)
                 S31_VAL=MATMUL(W_COL,S13_COL)
C                 S33_VAL=MATMUL(W_COL,S33_COL)
                 STRESS_OUT(THICK_P,22)=H/DIVS/2.0D0*(THICK_P-1.0D0)
     +           *S32_VAL(1,1)
                 STRESS_OUT(THICK_P,23)=H/DIVS/2.0D0*(THICK_P-1.0D0)
     +           *S31_VAL(1,1)
C                CHANGE THE EXPRESSION FOR 33 LATER
                 STRESS_OUT(THICK_P,24)=STRESS_OUT(THICK_P,6)


C                  CALL FEM_SHAPE_DIFF(LINT_COORD(I,1),LINT_COORD(I,2)
C      +           ,I_ORDER,DN1X,DN2X,DN3X,DN4X,DN1Y,DN2Y,DN3Y,DN4Y,
C      +           DET_J,LE,BE)
               CALL FEM_SHAPE_GLOBAL(X_I,X_J,Y_I,Y_L,INT_COORD(I,1),
     +         INT_COORD(I,2),GN1,GN2,GN3,GN4,LE,BE)



               ENDIF !END OF IF STATEMENT FOR THICK_P=2,G_Z
            ENDDO !!!END OF THICK_P LOOP

         ELSEIF (IELX==1) THEN
C        THE STATEMENT FOR IDENTIFYING THE ELEMENTS AT X=0

         ELSEIF (IELX==N_L) THEN


         ENDIF  !END OF IF STATEMENTS FOR INTERIOR ELEMENTS


         FORMAT (EN11.2)
         FORMAT (999(2x,EN14.4))
         FORMAT (999(x,EN11.2))
         FORMAT (999(X,I5))
         END SUBROUTINE SRS_NLIN

```

```
!-----------------------------------------------------------------------
      MODULE PLATE_CONST
      IMPLICIT NONE
      REAL(KIND=8),SAVE :: L, B, H
!     L IS THE LENGTH OF THE BEAM
!     B IS THE WIDTH OF THE BEAM
!     H IS THE THICKNESS OF THE BEAM
!-----------------------------------------------------------------------
      REAL(KIND=8),SAVE :: RHO
!     RHO IS THE DENSITY OF PLATE
!-----------------------------------------------------------------------
      REAL(KIND=8),SAVE::CC(6,6)
!     MATERIAL STIFFNESS MATRIX
!-----------------------------------------------------------------------
      INTEGER,SAVE :: I_CURVE
!     1 - STRAIGHT PLATE, 2 - CURVED PLATE
      REAL(KIND=8) :: APP_LOAD
!     APPLIED UDL OR PEAK LOAD IN SPATIAL DEPENDENT LOAD
!     DYNAMIC IN THE FORM, P=APP_LOAD*EXP^-T/THETA
      REAL(KIND=8) :: THETA
      END MODULE PLATE_CONST
!-----------------------------------------------------------------------



!-----------------------------------------------------------------------
      MODULE FEM_CONST
      IMPLICIT NONE
      INTEGER,SAVE :: N_L,N_B,N_FEM,I_ORDER,I_BC,I_LIN,G_X,G_Y
     + ,G_Z,I_THEORY,I_STATIC,NODE_B,NODE_L, I_LOAD, I_VON, I_TRAC
     + ,SRS_ID
!     N_L,N_B - ELEMENTS ALONG THE LENGTH AND WIDTH
!     N_FEM - NUMBER OF ELEMENTS FOR FEM
!     I_ORDER - ORDER OF THE SHAPE FUNCTIONS
!     1 - LINEAR SHAPE FUNCT. (4 NODED), 2 - QUADRATIC(8 NODED)
!     I_BC - BOUNDARY CONDITIONS 1-SSSS, 2-CCCC (SEE INOUT.DAT)
!     I_LINEAR 1- GEOMETRICALLY LINEAR, 2 - GEOMETRIC NONLINEARAITY
!     I_VON - 1 - VON KARMAN / 2 - FULL NONLINEARITY
!     I_THEORY - 1 - 1ST ORDER, 3-THIRD ORDER
!     I_STATIC - 1 -STATIC PROBLEM, 2 - DYNAMIC PROBLEM
!     I_LOAD - SPATIAL DEPENDENCY OF LOAD 1 - INDEP. 2 - DEP.
!     I_TRAC - 1 - FOLLOWER LOAD ON DEFORMED AREA/2 - LOAD ON UNDEFORMED
!     G_X - NUMBER OF GAUSS POINTS IN X
!     G_Y, G_X NUMBER OF POINTS IN Y AND Z RESP.
!     NODE_L, NODE_B - NUMBER OF NODES ALONG LENGTH AND WIDTH RESP.
!-----------------------------------------------------------------------
      INTEGER,SAVE :: N_NODE, E_NODE, N_DOF, E_DOF,TOT_DOF
!     N_NODE = TOTAL NUMBER OF NODES, E_NODE - NUMBER OF NODES IN AN ELEM.
!     N_DOF=DEGREES OF FREEDOM PER NODE, E_DOF - DOF PER ELEMENT------
!     TOT_DOF - TOTAL DEGREES OF FREEDOM--------------------------------
      REAL(KIND=8) :: T0,DT,TSTOP
!     INITIAL TIME, TIME STEP AND END TIME RESPECTIVELY
      INTEGER :: TSTEPS
      REAL(KIND=8) :: BETA,GAMMA
!     CONSTANTS OF NEWMARK TIME INTEGRATION ALOGRITHM
      REAL(KIND=8) :: A1,A2,A3,A4,A5,A6,A7
!     COEFFICIENTS OF NEWMARK BETA
      END MODULE FEM_CONST
!-----------------------------------------------------------------------
!
!
!
!-----------------------------------------------------------------------
      MODULE FEM_MATRICES
      REAL(KIND=8), ALLOCATABLE :: KE(:,:)
!     ELEMENT STIFFNESS MATRIX-----------------------------------------
      REAL(KIND=8), ALLOCATABLE :: ME(:,:)
!     ELEMENT MASS MATRIX - CONISSTENT AND LUMPED RESP.----------------
      REAL(KIND=8), ALLOCATABLE :: VN(:),AN(:),VN1(:),AN1(:)
```

```
!     VELOCITIES AND ACCELERATIONS OF EVERY NODE----------
      REAL(KIND=8),ALLOCATABLE :: GLOB_XYZ(:,:),KK(:,:), MM(:,:)
!     GLOB_XZ - GLOBAL COORDINATES OF EACH NODE------------------------
!     KK - GLOBAL STIFFNESS MATRIX-------------------------------------
!     MM - GLOBAL MASS MATRIX------------------------------------------
      REAL(KIND=8), ALLOCATABLE :: KG_VAL(:),MG_VAL(:)
      INTEGER, ALLOCATABLE :: COL_K(:),ROWIND_K(:),COL_M(:),ROWIND_M(:)
!     SPARSE MATRIX ARRAYS IN ORDER - VALUES, COLUMN ID, ROWINDEX FOR---
!     K AND M
      REAL(KIND=8), ALLOCATABLE :: MG_VAL_ENER(:),COL_M_ENER(:)
     + ,ROWIND_M_ENER(:)
      INTEGER :: NZ_K,NZ_M,NZ_M2   !TOTAL NUMBER OF NONZEROS IN K AND M
      REAL(KIND=8), ALLOCATABLE :: FE(:),FF(:),FFIN(:),FFN1(:)
!     ELEMENTAL AND GLOBAL LOAD VECTORS
      REAL(KIND=8),ALLOCATABLE :: SOL(:),SOLN1(:)
!     DISPLACEMENTS
      REAL(KIND=8), ALLOCATABLE :: U10(:),U11(:),U12(:),U13(:)
!     NAMED WITH SAME DEFINITIONS AS IN THE THEORY
      REAL(KIND=8), ALLOCATABLE :: U20(:),U21(:), U22(:), U23(:)
      REAL(KIND=8),ALLOCATABLE :: U30(:), U31(:), U32(:), U33(:)
      REAL(KIND=8), ALLOCATABLE :: V10(:),V11(:),V12(:),V13(:)
!     NAMED WITH SAME DEFINITIONS AS IN THE THEORY
      REAL(KIND=8), ALLOCATABLE :: V20(:),V21(:), V22(:), V23(:)
      REAL(KIND=8),ALLOCATABLE :: V30(:), V31(:), V32(:), V33(:)
      REAL(KIND=8), ALLOCATABLE :: A10(:),A11(:),A12(:),A13(:)
!     NAMED WITH SAME DEFINITIONS AS IN THE THEORY
      REAL(KIND=8), ALLOCATABLE :: A20(:),A21(:), A22(:), A23(:)
      REAL(KIND=8),ALLOCATABLE :: A30(:), A31(:), A32(:), A33(:)
      REAL(KIND=8), ALLOCATABLE :: ELE_STRESS(:,:)
      REAL(KIND=8), ALLOCATABLE :: ELE_STRAIN(:,:)
      REAL(KIND=8), ALLOCATABLE :: ELE_DISP(:,:)
!      REAL(KIND=8), ALLOCATABLE :: KK_MOD(:,:),MM_MOD(:,:),FF_MOD(:)
!     THESE ARE THE MODIFIED GLOBAL STIFFNESS AND MASS MATRIX BY
!     IMPOSING THE BOUNDARY CONDITIONS
      REAL(KIND=8) :: TU   !STRAIN ENERGY AT A GIVEN TIME
      END MODULE FEM_MATRICES
!---------------------------------------------------------------------
!
!
!
!---------------------------------------------------------------------
      MODULE NUM_INT  !MODULE FOR DEFINING GAUSS QUAD POINTS AND WEIGHTS
      REAL(KIND=8), ALLOCATABLE :: X_W(:), Y_W(:), Z_W(:)
!     X_W - WEIGHTS AT GAUSS PNT.S IN X, Z_W WEIGHTS AT GAUSS PNT.S IN Z
      REAL(KIND=8), ALLOCATABLE :: X_P(:), Y_P(:), Z_P(:)
!     LOCAL COORDINATES OF GAUSS INT. PNT.S IN X AND Z
      END MODULE NUM_INT
!---------------------------------------------------------------------
!
!---------------------------------------------------------------------
      MODULE PROG_DEBUG
      LOGICAL :: DEBUG = .FALSE.  !LOGICAL OPERATOR FOR DEBUGGING
      LOGICAL :: OUTPUT= .FALSE.  !LOGICAL OPERATOR FOR SPECIFIC OUTPUTS
      LOGICAL :: MANU= .FALSE.     !LOGICAL OPERATOR FOR MANU. SOLUTIONS
      END MODULE PROG_DEBUG
```