Analysis and Design Optimization of Laminated Composite Structures using Symbolic Computation

Evan Summers

Submitted in fulfilment of the academic requirements for the degree of Doctor of Philosophy in the Department of Mechanical Engineering at University of Natal.

Durban, South Africa
October 1994

To Ben Coughlan

Abstract

The present study involves the analysis and design optimization of thin and thick laminated composite structures using symbolic computation.

The fibre angle and wall thickness of balanced and unbalanced thin composite pressure vessels are optimized subject to a strength criterion in order to maximise internal pressure or minimise weight, and the effects of axial and torsional forces on the optimum design are investigated.

Special purpose symbolic computation routines are developed in the C programming language for the transformation of coordinate axes, failure analysis and the calculation of design sensitivities. In the study of thin–walled laminated structures, the analytical expression for the thickness of a laminate under in-plane loading and its sensitivity with respect to the fibre orientation are determined in terms of the fibre orientation using symbolic computation. In the design optimization of thin composite pressure vessels, the computational efficiency of the optimization algorithm is improved via symbolic computation.

A new higher–order theory which includes the effects of transverse shear and normal deformation is developed for the analysis of laminated composite plates and shells with transversely isotropic layers. The *Mathematica* symbolic computation package is employed for obtaining analytical and numerical results on the basis of the higher–order theory. It is observed that these numerical results are in excellent agreement with exact three-dimensional elasticity solutions. The computational efficiency of optimization algorithms is important and therefore special purpose symbolic computation routines are developed in the C programming language for the design optimization of thick laminated structures based on the higher–order theory.

Three optimal design problems for thick laminated sandwich plates are considered, namely, the minimum weight, minimum deflection and minimum stress design. In the minimum weight problem, the core thickness and the fibre content of the surface layers are optimally determined by using equations of micromechanics to express the elastic constants. In the minimum deflection problem, the thicknesses of the surface layers are chosen as the design variables. In the minimum stress problem, the relative thicknesses of the layers are computed such that the maximum normal stress will be minimized. It is shown that this design analysis cannot be performed using a classical or shear–deformable theory for the thick panels under consideration due to the substantial effect of normal deformation on the design variables.

/Абстракт

Абстракт

Настоящее исследование включает в себя оптимизацию тонких и толстых слоистых композитных конструкций используя символьный метод вычисления.

Предметом оптимизации, с использованием критерия прочности, являются угол армирования волокон и толщина стенки сбалансированных и несбалансированных тонких композитных сосудов давления, в результате чего определяется максимальное допустимое внутреннее давление или минимальный вес. Эффект продольных и крутящих сил учитывается при оптимизации параметров конструкций.

Для преобразования координат, анализа разрушения и оценки чувствительности конструкций при оптимальном проектировании разработан метод специальных символьных вычислений с использованием алгоритмического языка программирования C. Используя процедуру символьного вычисления при исследовании тонкостенных слоистых конструкций было получено аналитическое выражение для определения толщины слоя при нагружении в плане и ее чувствительность в зависимости от угла армирования. Вычислительная эффективность алгоритма оптимизации при дизайне тонких композитных сосудов давления значительно улучшена благодаря использованию метода символьных вычислений специального назначения.

Разработана новая уточненная неклассическая теория учитывающая эффект поперечного сдвига и обжатия для анализа слоистых композитных пластин и оболочек с трансверсально–изотропными слоями. Символьный язык программирования *Mathematica* используется для получения аналитических и численных результатов на основе уточненной неклассической теории. Показано, что численные результаты прекрасно сходятся с точным трехмерным упругим решением. Вычислительная эффективность алгоритма оптимизации очень важна и поэтому для оптимизации толстых слоистых конструкций была разработана основанная на уточненной неклассической теории процедура символьного метода специального назначения с использованием алгоритмического языка C.

Рассмотрены три оптимизационные задачи для толстых трехслойных плит, а именно, дизайн минимального веса, минимальных деформаций и минимальных напряжений. При подборе оптимального веса толщина заполнителя и характеристики волокон внешних слоев определяются используя уравнения микромеханики которые определяют значение упругих констант. При рассмотрении задачи минимальных деформаций толщины внешних слоев выбираются в ка-

честве переменных. Рассматривая задачу минимальных напряжений относительные толщины слоев вычисляются так, чтобы максимальное нормальное напряжение было минимизировано. Показано, что из–за значительного влияния обжатия на переменные оптимизации этот дизайн не может быть осуществлен используя классическую или сдвиговую деформационную теорию при рассмотрении толстых плит.

Declaration

I declare that this dissertation is my own unaided work except where due acknowledgement is made to others. This dissertation is being submitted for the Degree of Doctor of Philosophy to the University of Natal, Durban, and has not been submitted previously for any other degree or examination.

Evan Summers October 1994

Acknowledgements

I express my gratitude to my supervisors, Professors Sarp Adali and Viktor Verijenko, for their guidance and encouragement; to their collaborator, Professor V. G. Piskunov, for his contributions; and to my colleague Pavel Tabakov for his support and encouragement.

Financial assistance from the Foundation of Research Development of South Africa is gratefully acknowledged.

Contents

Nomenclature

1. Thin–walled structures

$k = 1, \ldots, n$	Layer number
x, y, z	Orthogonal coordinates
N_x, N_y, N_{xy}	Force resultants
$\epsilon_x, \epsilon_y, \epsilon_{xy}$	Normal and shear strains
θ_k	Fibre angle of the k-th layer
x, ϕ, z	Cylindrical coordinates
R	Mean Radius
p	Internal pressure of a pressure vessel
T	Torque
F	Axial force
θ_{opt}	Optimal fibre angle
p_{cr}	Burst pressure
p_{max}	Maximum internal pressure
H_{cr}	Laminate thickness at failure
H_{min}	Minimum thickness
W_{min}	Minimum weight
Q_{ij}	Stiffness coefficients
$\bar{Q}_{ij}$	Reduced stiffness coefficients
E_1, E_2	Moduli of elasticity in the material coordinate system
ν_1, ν_2	Poisson's ratios in the material coordinate system
$\sigma_1, \sigma_2, \tau_{12}$	Stress components in the material coordinate system
$\varepsilon_1, \varepsilon_2, \varepsilon_{12}$	Strain components in the material coordinate system

Nomenclature

2. Higher–order theory for thick plates and shells

x_1, x_2, z	Curvilinear orthogonal coordinates
$x = (x_1, x_2)$	Orthogonal coordinate pair
k_{11}, k_{22}	Curvatures of a shell
$k = 1, \ldots, n$	Layer number
E_k, ν_k, G_k	Modulus of elasticity, Poisson's ratio and shear modulus in the k-th layer in the plane of isotropy
E'_k, ν'_k, G'_k	Modulus of elasticity, Poisson's ratio and shear modulus in the k-th layer in the transverse direction
$u_1(x), u_2(x), w(x)$	Displacements and deflection of the reference surface
$\chi_1(x)$	Shear function
$\chi_2(x)$	Compression function
$q^+(x), q^-(x)$	Normal loading on the bounding surfaces
$u_1^{(k)}, u_2^{(k)}, u_3^{(k)}$	Displacements in the k-th layer
$\alpha_{qk}(z), \varphi_{qk}(z), \psi_{gk}(z)$	Distribution functions through the thickness
$\sigma_{11}, \sigma_{12}, \sigma_{22}$	In–plane stresses
σ_{13}, σ_{23}	Transverse shear stresses
σ_{33}	Transverse normal stress
$E_{0k}, \nu_{0k},$	
$A_{11k}, A_{12k}, A_{13k}, A_{33k}$	Stiffness parameters of the k-th layer
e_{ij}, e_{i3}, e_{33}	Components of the strain tensor

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Advanced composite materials have properties which are quite different from conventional materials. In many engineering applications it is more advantageous to use composite materials rather than conventional ones. In particular, advanced composite materials are widely used in applications where a high strength–to–weight ratio is the most important criterion in the choice of material.

The cost of advanced composite materials is significantly higher than that of conventional materials and therefore the design optimization of composite structures is important in order to maximise the benefits which composites offer and to better utilise these expensive materials. In particular, an effective way to reduce the cost of such structures is via hybridization. Laminated structures may fulfil the design requirements and yet be substantially cheaper than homogeneous structures owing to the use of cheaper materials as filler layers.

The objective of the present study is the design optimization of a suite of laminated composite structures. In the first instance thin laminates are studied, in particular balanced and unbalanced laminated composite pressure vessels with specially orthotropic layers whose elastic properties depend on the angle of reinforcing fibres.

Clearly the analysis of laminated structures manufactured from different materials which may be orthotropic or transversely isotropic is a demanding area of computational solid mechanics and one well suited to the use of symbolic computation. Symbolic computation systems are able to mathematically manipulate expressions

in symbolic form and may be used to derive analytical results or formulae for numerical computations.

In the optimization study of composite pressure vessels, special purpose symbolic computation routines are developed to improve the computational efficiency of the optimization algorithm. These routines reduce the number of calculations required in each iteration of the optimization algorithm by combining the relationship between the loading parameters and the material stress into one transformation matrix.

The analysis of laminated composite structures on the basis of analytical solutions of the three–dimensional equations of elasticity is cumbersome. It is more common rather to employ a two–dimensional theory which is derived from the three–dimensional theory of elasticity via some assumptions or hypotheses. For example, the classical shell theory is based on the Kirchhoff–Love assumptions which neglect transverse stresses. Clearly a theory based on certain assumptions will lose accuracy where those assumptions are not valid. In particular, the classical shell theory is accurate for *thin* structures but not for *thick* ones. The challenge then is to derive a two–dimensional theory which is accurate for *thin* and *thick* structures. This has led to the development of *improved* or *refined* theories which include the effects of transverse shear. However, in thick laminated composite structures, there are two important effects, namely transverse shear and normal deformation. A theory which neglects normal deformation is based on the assumption that the structure is rigid in the transverse direction, and this assumption is invalid for thick structures.

Nonclassical theories which include both transverse shear and normal deformation are developed by Piskunov and Verijenko in Refs. [42, 31, 46, 45]. The approach is used in Ref. [44] to develop a higher–order theory which takes both transverse shear and normal deformation into account more comprehensively.

Clearly the computational implementation of a theory which is accurate for thick composite laminated plates and shells with layers with significantly different elastic properties, is expected to exact demanding computational effort, and indeed this is the case. The higher–order theory introduces distribution functions and integrated stiffness constants which in general are multiple piecewise integrals through the thickness of the laminate and in the general case cannot be derived in a form suitable for direct numerical implementation. Therefore the higher–order theory is implemented using symbolic computation. In the first instance, a general purpose symbolic computation system is employed. However, in design optimization studies on the basis of the higher–order theory it is necessary to integrate the symbolic computations into the optimization algorithm. This requirement together with the

unimpressive computational efficiency of the general purpose system makes such studies infeasible using this system. Therefore special purpose symbolic computation routines are developed in a conventional programming language for the implementation of the higher–order theory. These routines are two orders of magnitude more efficient than the general purpose system and are easily incorporated into the optimization algorithm.

In the present study, this new theory is employed for the analysis and design optimization of thick structures using symbolic computation. In particular, three optimization problems for thick composite sandwich plates are considered, namely, minimum weight, minimum deflection and minimum stress designs. It is shown that the design analysis cannot be performed using a classical or shear–deformable theory due to the substantial effect of normal deformation.

1.2 Symbolic Computation

In a numerical optimization technique which involves phases of design and analysis, the efficiency depends heavily on the computational time taken by the analysis. The same considerations also apply to the evaluation of the design sensitivities which may be needed in the numerical optimization algorithm to determine the sensitivity of a design with respect to the problem parameters, and in particular to the design variables.

The use of general purpose symbolic computation in a design optimization problem is computationally expensive due to the iterative nature of optimization algorithms. However, the development of special purpose symbolic computation software to perform the analysis phase leads to substantial gains in computational efficiency as compared to using a general purpose symbolic computation tool. In optimization studies, computational efficiency is of paramount importance. Therefore the implementation of special purpose symbolic computation is preferable to the use of a general purpose symbolic computation system. The efficiency of special purpose symbolic computation stems from its dedication to the analysis of a specific class of functions. In fact, the key observation which makes the development of special purpose symbolic computation software a realistic objective in a given problem is that, in general, the expressions needed in the calculations are confined to specific classes of functions.

A major motivation to develop such routines is to be able to incorporate the symbolic

computations into an iterative solution procedure. These features are particularly important when symbolic computations need to be performed within each iteration, or when the efficiency of an iterative optimization procedure may be improved by incorporating some symbolic analysis before the iterations in order to reduce the number of numerical calculations required in each iteration. Even if the symbolic computations are not essential, the increased efficiency may justify the development of special purpose symbolic computation routines for specific applications.

1.3 Optimization of Thin–Walled Structures

Fibre-reinforced composite materials are finding increased use in various engineering applications, and the optimization of such structures is a natural part of the design process in order to maximize the benefits which these materials can offer.

A major advantage of fibre reinforced composite materials is the large number of design variables available to the designer. To realize this potential and to maximize the benefits which composites can offer, the design has to be tailored to the specific requirements of the problem. Optimization of the design is an effective way of achieving this goal.

Special purpose symbolic computation routines are developed in a conventional programming language for the transformation of coordinate axes, failure analysis and the calculation of design sensitivities. In the study of thin–walled laminated structures, the analytical expression for the thickness of a laminate under in-plane loading and its sensitivity with respect to the fibre orientation are determined in terms of the fibre orientation using special purpose symbolic computation. In the design optimization of thin composite pressure vessels, the computational efficiency of the optimization algorithm is improved by using special purpose symbolic computation routines to combine the relationship between the loading parameters and the material stress into one transformation matrix.

Thin composite pressure vessels are optimized subject to a strength constraint in order to maximise the internal pressure or minimise the weight of the structure. The fibre orientation is determined for balanced and unbalanced laminations in order to maximize the internal pressure, and the effects of axial and torsional forces on the optimal design are investigated. The weight of a liquid filled pressure vessel is minimized taking both the fibre orientation and the wall thickness as design variables. Both constant and variable wall thickness cases are investigated and

comparative numerical results are presented for single and multiple layered vessels.

Simultaneous design of pressure vessels with respect to fibre orientations and thickness distributions does not seem to be considered in the literature.

1.4 Higher–Order Theory for Thick Plates and Shells

The effects of both transverse shear and normal deformation are substantial in thick structures. Therefore an improved higher–order theory is presented for the analysis of laminated transversely isotropic plates and shells subject to transverse shear and normal deformation. The theory is capable of analysing the three–dimensional stress–strain behaviour of laminated plates and shells with an arbitrary number of layers which may differ significantly in their physical and mechanical properties.

Closed form solutions on the basis of the higher–order theory are considered for the analysis of thick structures. *Mathematica* is employed to generate analytical and numerical results. The numerical results are compared to those given in the literature in order to validate the analysis presented. The features of this theory and the implications of the numerical results are discussed.

Special purpose symbolic computation routines are developed in the C programming language for a general and computationally efficient implementation of the higher–order theory. The routines process symbolic expressions and derive power series expressions for symbols. The software using these routines is able to derive the distribution functions of the higher–order theory, calculate the integrated stiffness constants exactly, and derive the stress and strain distributions through the thickness in power series form for a given laminate.

1.5 Optimization of Thick Structures

The optimal design of thick composite structures poses special challenges because of the additional effects of transverse shear and normal deformations which have to be taken into account for a realistic analysis.

Three optimal designs of thick sandwich plates are considered on the basis of the

higher–order theory, namely, minimum weight, minimum deflection and minimum stress designs. The surface layers are made of a transversely isotropic composite material and the core material may be isotropic or transversely isotropic.

In the minimum weight design problem, the core thickness and the fibre content of the surface layers are optimally determined by using equations of micromechanics to express the elastic constants. In the minimum deflection problem, the relative thickness of the surface and core layers is chosen as the design variable. In the minimum stress problem, the relative thicknesses of the layers are determined such that the maximum normal stress will be minimized.

Numerical results are given for thick sandwich plates under sinusoidal loading and the effects of various input parameters are investigated. The deflection and stress behaviour is studied and it is shown that design analysis cannot be performed using a classical theory or a shear deformable theory for the thick plates under consideration.

Design of thick sandwich structures using a higher–order theory which includes normal as well as shear deformation does not seem to be considered in the literature. In fact previous studies on the optimal design of thick laminated structures seem to be based on shear deformable theories only.

Chapter 2

Optimization of Thin Walled Structures using Special Purpose Symbolic Computation

2.1 Introduction

The present chapter addresses the problem of optimally designing thin–walled composite laminates using symbolic computation. The analysis is based on the membrane theory of shells and the optimization is carried out with respect to fibre orientations and thickness distributions subject to a quadratic failure criterion.

Symbolic computation software is developed in the C programming language for the transformation of coordinate axes, failure analysis and the calculation of design sensitivities. These computations arise in the design optimization studies of structures made of fibre reinforced composite materials. The symbolic computations are integrated into an optimization algorithm resulting in a combined symbolic and numerical approach to determine the optimal design.

In order to illustrate the approach using the *special purpose* symbolic computation for the design optimization of laminated structures, a laminate under in–plane loads is designed for minimum thickness taking the fibre orientation as the design variable. The relationship between the loading parameters and the material stress is combined and simplified into one transformation matrix using symbolic computation. The stresses are determined symbolically in terms of the fibre angle for a balanced symmetric laminate under a given loading, and substituted into a quadratic failure

criterion. The analytical expressions for the laminate thickness in terms of the fibre angle and its sensitivity with respect to the fibre angle are then determined using the symbolic computation software.

Finally, an optimal design approach is presented for laminated composite pressure vessels. The fibre orientation and wall thickness are taken as the design variables. The lamination can be balanced or unbalanced. The balanced case refers to a lamination in which the layers with the same positive and negative fibre angles balance each other out. Two examples are considered. The first one involves pressure vessels under uniform internal pressure and subjected to axial and torsional forces, and the second example concerns circular cylindrical shells filled with a liquid. The optimal thickness distribution is obtained in the case of liquid filled vessels where the pressure distribution is a function of the axial coordinate. The effect of various problem parameters on the optimal designs are investigated.

2.2 Literature Review

Previous studies involving the optimization of laminated pressure vessels include Refs. [1]–[10]. In Ref. [1], the minimum mass of fibres is determined subject to a tensile strength condition assuming inextensible fibres. Designs in Ref. [2] are based on Flügge's theory of shells and the Tsai–Hill failure criterion is employed as the strength condition. Optimal designs based on criteria other than a failure one are given in Refs. [3]–[5]. Optimum shapes of filament–wound pressure vessels are determined subject to the Tsai–Hill failure criterion in Ref. [6]. Optimal fibre orientations for cylindrical pressure vessels are obtained by Fukunaga & Chou [7] for balanced stacking sequences. Karandikar *et al.* [8] considered a multiobjective approach to the design of composite pressure vessels by including deflection, weight and volume in the performance index. In Refs. [9] and [10], Donnell's shell theory is used to investigate the effect of temperature and fuzzy strength data, respectively, on the optimal design of laminated pressure vessels. Simultaneous design of pressure vessels with respect to fibre orientations and thickness distributions does not seem to be considered in the literature.

A review of use of symbolic computation in the solution of engineering problems is given by Beltzer in Ref. [15]. Several *general purpose* symbolic computation packages are presently available for such analysis and have found use in the solution of various engineering problems such as rotor dynamics [16], flutter [17], instability [18] and

buckling [19]. Symbolic computation has also been employed in the buckling [20], stress [21] and vibration [22] analysis of composite structures. As pointed out by Graaf & Springer [21], symbolic computation provides a powerful tool for the analysis of laminated structures made of a fibre composite material in view of the complexity of axis transformations.

2.3 Laminate under In-plane Load

The approach for the design optimization of laminated composite structures using special purpose symbolic computation is presented in this section.

A laminate under in–plane loads is designed for minimum thickness taking the fibre orientation as the design variable. The relationship between the loading parameters and the material stress is combined and simplified into one transformation matrix using symbolic computation. This involves tedious matrix algebra where the entries are series of trigonometric functions of the fibre angle. The stresses are determined symbolically in terms of the fibre angle for a balanced symmetric laminate under a given loading, and substituted into the Tsai–Wu failure criterion. The analytical expressions for laminate thickness in terms of the fibre angle, and its sensitivity with respect to the fibre angle, are then determined using the symbolic computation software.

2.3.1 Basic Equations

A balanced symmetric laminate of thickness H is considered. The laminate consists of an even number of orthotropic layers of equal thickness t. The fibre angles are orientated symmetrically with respect to the middle surface such that $\theta_k = (-1)^{k-1}\theta$ for $k \leq n/2$ and $\theta_k = (-1)^k\theta$ for $k \geq n/2+1$ where k is the layer number and n is the total number of layers. The coordinate axes are x, y and z where z is perpendicular to the plate with the origin lying in the middle surface of the plate. The laminate is subjected to the normal loads N_x, N_y and the shear load N_{xy} in the xy plane.

Due to the symmetry of the lamination, the force resultants in the coordinate axes are given by

$$[N] = [A][\epsilon] \tag{2.1}$$

where

$$[N] = \begin{pmatrix} N_x \\ N_y \\ N_{xy} \end{pmatrix}, \quad [A] = \begin{pmatrix} A_{11} & A_{12} & A_{16} \\ A_{12} & A_{22} & A_{26} \\ A_{16} & A_{26} & A_{66} \end{pmatrix}, \quad [\epsilon] = \begin{pmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{pmatrix} \tag{2.2}$$

where A_{ij} are the external stiffnesses given by $A_{ij} = H\bar{Q}_{ij}(\theta)$, $H = nt$ and ϵ_x, ϵ_y and γ_{xy} denote the normal and shear strains. Here $\bar{Q}_{ij}(\theta)$ are the transformed reduced stiffness coefficients given by

$$\begin{aligned}
\bar{Q}_{11} &= Q_{11}\cos^4\theta + 2(Q_{12} + 2Q_{66})\ \cos^2\theta\ \sin^2\theta + Q_{22}\sin^4\theta \\
\bar{Q}_{12} &= (Q_{11} + Q_{22} - 4Q_{66})\ \sin^2\theta\ \cos^2\theta + Q_{12}(\sin^4\theta + \cos^4\theta) \\
\bar{Q}_{16} &= (Q_{11} - Q_{12} - 2Q_{66})\ \sin\theta\ \cos^3\theta \\
&\quad + Q_{12} - Q_{22} + 2Q_{66})\ \sin^3\theta\ \cos\theta \\
\bar{Q}_{22} &= Q_{11}\sin^4\theta + 2(Q_{12} + 2Q_{66})\ \sin^2\theta\ \cos^2\theta + Q_{22}\cos^4\theta \\
\bar{Q}_{26} &= (Q_{11} - Q_{12} - 2Q_{66})\ \sin^3\theta\ \cos\theta \\
&\quad + (Q_{12} - Q_{22} + 2Q_{66})\ \sin\theta\ \cos^3\theta \\
\bar{Q}_{66} &= (Q_{11} + Q_{22} - 2Q_{12} - 2Q_{66})\ \sin^2\theta\ \cos^2\theta \\
&\quad + Q_{66}(\sin^4\theta + \cos^4\theta)
\end{aligned} \tag{2.3}$$

where the reduced stiffness coefficients Q_{ij} are given by

$$\begin{aligned}
Q_{11} &= \frac{E_1}{1 - \nu_{12}\nu_{21}} \qquad Q_{12} = \frac{\nu_{12}E_2}{1 - \nu_{12}\nu_{21}} \\
Q_{22} &= \frac{E_2}{1 - \nu_{12}\nu_{21}} \qquad Q_{66} = G_{12}
\end{aligned} \tag{2.4}$$

It is noted that for the laminate configuration to be considered, $\bar{Q}_{11}, \bar{Q}_{12}, \bar{Q}_{22}$ and $\bar{Q}_{66}$ are independent of the layer number, since $\bar{Q}_{ij}(\theta) = \bar{Q}_{ij}(-\theta)$ for these entries. Moreover $A_{16} = A_{26} = 0$ for laminates consisting of an even number of layers of equal thickness and alternating fibre orientations since $\bar{Q}_{16}(\theta) = -\bar{Q}_{16}(-\theta)$ and $\bar{Q}_{26}(\theta) = -\bar{Q}_{26}(-\theta)$.

The stress–strain equations for the k–th orthotropic layer are given by

$$[s^{(k)}] = [\bar{Q}^{(k)}][\epsilon] \tag{2.5}$$

where $[\epsilon] = [A]^{-1}[N]$ from eqn. (2.1), and $[s^{(k)}]$ denotes the stress components $[\sigma_x^{(k)}\ \sigma_y^{(k)}\ \tau_{xy}^{(k)}]^T$ in the xy coordinate system.

The stress components in the material coordinate system, denoted by

$$[\sigma^{(k)}] = [\sigma_1^{(k)}\ \sigma_2^{(k)}\ \tau_{12}^{(k)}]^T$$

are obtained from the geometric stress components $[s^{(k)}]$ via the matrix transformation

$$[\sigma^{(k)}] = [T^{(k)}][s^{(k)}] \tag{2.6}$$

where $[T^{(k)}] = [T(\theta_k)]$ denotes the transformation matrix for the k-th layer given by

$$[T^{(k)}] = \begin{pmatrix} \cos^2\theta_k & \sin^2\theta_k & 2\cos\theta_k \sin\theta_k \\ \sin^2\theta_k & \cos^2\theta_k & -2\cos\theta_k \sin\theta_k \\ -2\cos\theta_k \sin\theta_k & 2\cos\theta_k \sin\theta_k & \cos^2\theta_k - \sin^2\theta_k \end{pmatrix} \tag{2.7}$$

From eqns. (2.1), (2.5) and (2.6) it follows that

$$[\sigma^{(k)}] = [T^{(k)}][\bar{Q}^{(k)}][A]^{-1}[N] \tag{2.8}$$

We denote the force–stress transformation matrix

$$[T_c^{(k)}] = [T^{(k)}][\bar{Q}^{(k)}][A]^{-1} \tag{2.9}$$

which is a function of the fibre angle θ_k of the k-th layer.

2.3.2 Design Optimization

The design problem involves determining the optimal fibre orientation θ to minimize the laminate thickness H subject to a strength criterion. In this study, the Tsai–Wu failure criterion is used which stipulates that the condition for non–failure is

$$\begin{aligned} &F_{11}\,\sigma_1^{(k)}\sigma_1^{(k)} + F_{22}\,\sigma_2^{(k)}\sigma_2^{(k)} + F_{66}\,\tau_{12}^{(k)}\tau_{12}^{(k)} \\ &\quad +2F_{12}\,\sigma_1^{(k)}\sigma_2^{(k)} + F_1\,\sigma_1^{(k)} + F_2\,\sigma_2^{(k)} \le 1 \end{aligned} \tag{2.10}$$

where the strength parameters F_{11}, F_{22}, F_{66}, F_{12}, F_1 and F_2 are given by

$$\begin{aligned} &F_{11} = 1/(X_t X_c)\,; \quad F_{22} = 1/(Y_t Y_c)\,; \quad F_{66} = 1/S^2 \\ &F_1 = 1/X_t - 1/X_c\,; \quad F_2 = 1/Y_t - 1/Y_c\,; \quad F_{12} = -\frac{1}{2}\sqrt{F_{11}F_{22}} \end{aligned}$$

where X_t, X_c, Y_t and Y_c are the tensile and compressive strengths of the composite material in the fibre and transverse directions, and S is the in–plane shear strength.

The optimal design problem is to determine the minimum thickness H_{min} of a laminate under the in–plane loads N_x, N_y and N_{xy} subject to the failure criterion (2.10), viz.

$$H_{min}(\theta_{opt}) \stackrel{\text{def}}{=} \min_{\theta} H(\theta) \tag{2.11}$$

subject to the constraint (2.10).

2.4 Special purpose symbolic computation

The special purpose symbolic computation routines developed in the C programming language for the optimization of laminated structures are presented in this section.

Special purpose symbolic computation is useful in optimization studies for improving the computational efficiency of the optimization algorithm. General purpose symbolic computation packages cannot, in general, be integrated into an optimization program developed in a conventional programming language. Moreover, the computational efficiency of general purpose symbolic computation systems is substantially less than that of special purpose systems which are dedicated to the specific problem at hand. Therefore in optimization studies, where computational efficiency is of paramount importance, the implementation of special purpose symbolic computation is more suitable than the use of a general purpose system.

The present study requires tedious matrix algebra where the entries are series of double trigonometric functions of the fibre angle. Special purpose routines are therefore developed to handle such expressions. The routines can perform matrix algebra involving matrices of trigonometric series and simplify the results using trigonometric identities. Since the routines manipulate a specific class of functions only, they are relatively simple and their development is a feasible objective.

Symbolic computation requires a great deal of dynamic memory allocation and access [24]. Therefore the C programming language is chosen for the development of the special purpose symbolic computation software presented in this section and a knowledge of the C language is assumed in the following discussion.

2.4.1 Data Storage

The first step is to define a storage class for the functions to be considered. Therefore, the structure `trigt` is defined by

```
typedef struct /* structure for trig series */
      {
      real  coeff; /* coefficient */
      int   fn[2]; /* function types */
      int   pow[2]; /* powers */
      int   harm[2]; /* harmonics of argument */
      char  var; /* argument */
```

```
}
trigt;
```

which contains a single term of the form

$$a\cos^m k\theta \cos^n l\theta, \quad a\sin^m k\theta \cos^n l\theta, \quad \text{or} \quad a\sin^m k\theta \sin^n l\theta$$

in a double trigonometric series, where a, m, n, k and l are constants and θ is a variable. In the laminate design application, θ denotes the fibre orientation.

A trigonometric series is stored as a null–terminated list of the `trigt` structure. Memory is dynamically allocated for each new series. A symbolic series is then accessed via a pointer to its memory address. A symbolic matrix is a two–dimensional array of such pointers to each entry of the matrix.

Various basic routines are coded to handle memory allocation for storing symbolic series and to define or duplicate a series. The routine `trig_alloc(n)` allocates memory for a series with `n` terms and returns the address of the allocated memory. The amount of memory required for a series is the size of the `trigt` structure multiplied by the number of terms in the series. When a series is no longer required, the memory it occupies is freed using the `trig_free` routine.

The routine `trig_set()` is used a define a trigonometric series in an application. For example, the expression

$$2\sin^3\theta + 4\cos^2\theta \tag{2.12}$$

is defined in a program by the code

```
trigt *ts;
ts = trig_set(2.,FnSin,3,  4.,FnCos,2,  0.);
```

The series is accessed via the pointer `ts` which contains the memory address where the defined series is stored.

2.4.2 Symbolic Processing

The routine `trig_add` adds two series by appending the two arrays of the structure `trigt` to form the sum. This routine then invokes `trig_collect` to collect the similar terms. In order to make this routine more flexible, two constants may be given for pre–multiplying the two series before they are summed.

The routine `trig_mult` multiplies two series and invokes `trig_collect`. Both the `trig_add` and `trig_mult` routines take the memory addresses of the two operand

series as arguments and return the memory address of the new resultant series created by the routine.

The routine `trig_diff` differentiates a series with respect to θ and returns the memory address of the symbolic derivative derived by the routine. The routine `trig_diff_calc` calculates the derivative of a series for a given θ without creating its symbolic derivative.

A double trigonometric series is simplified using the `trig_expand` routine, which is recursive and employs trigonometric identities to expand a given series into a series of single trigonometric functions of various harmonics, each to the power of one. This routine uses a routine `trig_binomial` to generate a symbolic binomial expansion. The trigonometric transformations that are employed by `trig_expand` are given by

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\cos^{n+m} k\theta \sin^n k\theta \\
\text{then } \texttt{result} &= a\tfrac{1}{2}\cos^m k\theta \sin^n 2k\theta
\end{aligned}$$

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\cos^{2n} k\theta \\
\text{then } \texttt{result} &= (\tfrac{a}{2})^n \ \texttt{binomial_expand}\ (\cos 2k\theta + 1)^n \\
&= (\tfrac{a}{2})^n \ \textstyle\sum_{r=0}^{n} \{ \frac{n!}{(n-r)!\,r!} \cos^{n-r} 2k\theta \}
\end{aligned}$$

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\cos^{2n+1} k\theta \\
\text{then } \texttt{result} &= (\tfrac{a}{2})^n \cos k\theta \ \texttt{binomial_expand}\ (\cos 2k\theta + 1)^n \\
&= (\tfrac{a}{2})^n \ \textstyle\sum_{r=0}^{n} \{ \frac{n!}{(n-r)!\,r!} \cos k\theta \cos^{n-r} 2k\theta \}
\end{aligned}$$

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\sin^{2n} k\theta \\
\text{then } \texttt{result} &= (\tfrac{a}{2})^n \ \texttt{binomial_expand}\ (1 - \cos 2k\theta)^n \\
&= (\tfrac{a}{2})^n \ \textstyle\sum_{r=0}^{n} \{ \frac{n!}{(n-r)!\,r!} (-\cos 2k\theta)^r \}
\end{aligned}$$

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\sin^{2n+1} k\theta \\
\text{then } \texttt{result} &= (\tfrac{a}{2})^n \sin k\theta \ \texttt{binomial_expand}\ (1 - \cos 2k\theta)^n \\
&= (\tfrac{a}{2})^n \ \textstyle\sum_{r=0}^{n} \{ \frac{n!}{(n-r)!\,r!} \sin k\theta (-\cos 2k\theta)^r \}
\end{aligned}$$

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\cos k\theta \sin l\theta \\
\text{then } \texttt{result} &= a\tfrac{1}{2}[\sin(l-k)\,\theta + \sin(l+k)\,\theta]
\end{aligned}$$

$$\begin{aligned}
\text{if} \quad \texttt{term} &= a\cos k\theta \cos l\theta \\
\text{then } \texttt{result} &= a\tfrac{1}{2}[\cos(l-k)\,\theta + \cos(l+k)\,\theta]
\end{aligned}$$

if `term` $= a \sin k\theta \sin l\theta$
then `result` $= a\frac{1}{2}[\cos(l-k)\theta - \cos(l+k)\theta]$

where the `binomial_expand` operator indicates where the routine `trig_binomial` is invoked to expand a binomial expression.

Terms with insignificant coefficients of trigonometric functions of high harmonics may appear in a series after processing by `trig_expand`. The routine `trig_significant` discards insignificant terms in a series in order to make the results more presentable.

When a series is to be manipulated by a routine, generally the routine is passed the address of the series. The routine then creates a new series for the result, without destroying the original series, and the memory address of the new series is returned by the routine. The routine `trig_op` is used when a series is to be processed into a new version and the old version discarded. This routine takes the address of the pointer as the first argument and the name of a processing routine as the second argument. The processing routine (such as `trig_collect`, `trig_expand`, `trig_significant` or `trig_diff`) is one which takes the address of a series as its only argument and returns the address of a new equivalent version of the series. The `trig_op` routine applies the processing function to the series, destroys the original version (using `trig_free`) and sets the pointer (which pointed at the original version) to the address of the new version. For example, the expression

$$2\sin^3\theta + 4\cos^2\theta$$

is differentiated and simplified by the code

```
trigt *ts;
ts = trig_set(2.,FnSin,3, 4.,FnCos,2, 0.);
trig_op(&ts,trig_diff);
trig_op(&ts,trig_expand);
trig_op(&ts,trig_significant);
```

2.4.3 Matrix Algebra

The determinant, adjoint and matrix product of symbolic matrices whose entries are double trigonometric series, are derived by the routines

```
trigt *trig_mat_det(sm1) /* determinant of matrix */
trigt **trig_mat_adj(sm1,sm2) /* adjoint of matrix */
trigt **trig_mat_mult(sm1,sm2,sm3) /* matrix product */
```

where `sm1`, `sm2` and `sm3` are two-dimensional arrays of pointers to the entries of the associated matrix. The routine `trig_mat_det` returns a pointer to the resultant series, and the first arguments of `trig_mat_adj` and `trig_mat_mult` are arrays of pointers to be assigned to the entries of the resultant matrix.

The routines in Section 2.4.2 are used by the routines which process symbolic matrices. For example, `trig_mat_mult` derives a matrix product using the routines `trig_mult` and `trig_add` to multiply entries of the operand matrices and to sum the products.

The routine `trig_mat_op` applies a processing routine to each entry of a matrix using `trig_op`, and therefore reassigns the pointer corresponding to each entry to the new versions of each entry and destroys the original versions.

2.5 Method of Solution

Since the symbolic computation is limited to series of trigonometric functions, it is necessary to restructure eqn. (2.8) so as to isolate these series. Therefore, noting that $A_{ij} = H\bar{Q}_{ij}(\theta)$, we define a matrix $[A_s]$ such that $[A] = H[A_s]$. Using the adjoint matrix, the inverse matrix $[A]^{-1}$ in eqn. (2.8) now may be expressed as

$$[A]^{-1} = \frac{1}{H}\frac{1}{\mathrm{Det}A_s}[\mathrm{Adj}A_s]^T \tag{2.13}$$

The symbolic matrix $[T_s^{(k)}]$ is defined as

$$[T_s^{(k)}] = [T^{(k)}][\bar{Q}^{(k)}][\mathrm{Adj}A_s]^T \tag{2.14}$$

and the symbolic stress vector denoted $[\sigma_s^{(k)}] = [\sigma_{s1}^{(k)}\ \sigma_{s2}^{(k)}\ \tau_{s12}^{(k)}]^T$ is defined as

$$[\sigma_s^{(k)}] = [T_s^{(k)}][N] \tag{2.15}$$

Therefore the stress $[\sigma^{(k)}]$ is given by

$$[\sigma^{(k)}] = \frac{1}{H}\frac{1}{\mathrm{Det}A_s}[\sigma_s^{(k)}] \tag{2.16}$$

Substituting the stresses (2.16) into the inequality (2.10) yields a quadratic equation in terms of H given by

$$\begin{aligned} &F_{11}\,\sigma_{s1}^{(k)}\sigma_{s1}^{(k)} + F_{22}\,\sigma_{s2}^{(k)}\sigma_{s2}^{(k)} + F_{12}\,\sigma_{s1}^{(k)}\sigma_{s2}^{(k)} + F_{66}\,\tau_{s12}^{(k)}\tau_{s12}^{(k)} \\ &+[\,F_1\,\sigma_{s1}^{(k)} + F_2\,\sigma_{s2}^{(k)}]\,(H\,\mathrm{Det}A_s) - (H\,\mathrm{Det}\,A_s)^2 = 0 \end{aligned} \tag{2.17}$$

The solution of eqn. (2.17) gives the critical thickness $H_{cr}(\theta)$ denoted by

$$H_{cr} = \frac{h_{s1} + \sqrt{h_{s2}}}{h_{s3}} \tag{2.18}$$

where the h_{si} are symbolic series expanded from the expressions

$$\begin{aligned} h_{s1} &= F_1\,\sigma_{s1}^{(k)} + F_2\,\sigma_{s2}^{(k)} \\ h_{s2} &= (F_1\,\sigma_{s1}^{(k)} + F_2\,\sigma_{s2}^{(k)})^2 + 4(\,F_{11}\,\sigma_{s1}^{(k)}\sigma_{s1}^{(k)} + F_{22}\,\sigma_{s2}^{(k)}\sigma_{s2}^{(k)} \\ &\qquad + F_{12}\,\sigma_{s1}^{(k)}\sigma_{s2}^{(k)} + F_{66}\,\tau_{s12}^{(k)}\tau_{s12}^{(k)}\,) \\ h_{s3} &= 2\,\mathrm{Det}\,A_s \end{aligned} \tag{2.19}$$

The first and second derivatives of H_{cr} with respect to θ may be determined exactly by differentiating the expression (2.18) with respect to the components h_{si} via symbolic computation.

2.5.1 Program

The procedure to derive $DetA_s$, $[T_s]$, $[\sigma_s]$ and $H_{cr}(\theta)$ is outlined below. Note that the dots represent omission. The symbolic computation routines for double trigonometric series are given in Appendix B.

First the pointers to symbolic series and matrices are defined by

```
trigt  *sym_h1,*sym_h2,*sym_h3;        /* components of H */
trigt  *sym_h1d,*sym_h2d,*sym_h3d;     /* 1st derivatives */
trigt  *sym_h1dd,*sym_h2dd,*sym_h3dd; /* 2nd derivatives */

. . .

trigt *sm1[3][3];
trigt *sym1;
```

```
trigt *sm_qb[3][3];        /* Qb matrix */
trigt *sm_as[3][3];        /* As matrix */
trigt *sm_t[3][3];         /* T matrix */

trigt *sym_det_as;         /* Det As */

trigt *sm_adj_as[3][3];    /* Adj As */
trigt *sm_ts[3][3];        /* Ts matrix */

trigt *sym_ss[3];          /* stress = Ts N */

trigt *sym_quad_b;         /* quadratic coefficients */
trigt *sym_quad_c;
```

The entries of the matrices $[\bar{Q}]$, $[T]$ and $[A_s]$ are defined as

```
/* Qb matrix */

sm_qb[0][0] = trig_set(q11,FnCos,4,
                2*(q12+2*q66),FnCosSin,2,2, q22,FnSin,4,0.);

sm_qb[0][1] = trig_set(q11+q22-4*q66,FnCosSin,2,2,
              q12,FnCos,4, q12,FnSin,4, 0.);

sm_qb[0][2] = trig_set(q11-q12-2*q66,FnCosSin,3,1,
              q12-q22+2*q66,FnCosSin,1,3, 0.);

sm_qb[1][1] = trig_set(q11,FnSin,4,
                2*(q12+2*q66),FnCosSin,2,2, q22,FnCos,4, 0.);

sm_qb[1][2] = trig_set(q11-q12-2*q66,FnCosSin,1,3,
                q12-q22+2*q66,FnCosSin,3,1, 0.);

sm_qb[2][2] = trig_set(q11+q22-2*q12-2*q66,FnCosSin,2,2,
                q66,FnCos,4, q66,FnSin,4, 0.);

sm_qb[1][0] = trig_dup(sm_qb[0][1],0); /* symmetric entries */
sm_qb[2][0] = trig_dup(sm_qb[0][2],0);
sm_qb[2][1] = trig_dup(sm_qb[1][2],0);
```

```
/* T matrix */

sm_t[0][0] = trig_set( 1., FnCos,    2,    0.);
sm_t[0][1] = trig_set( 1., FnSin,    2,    0.);
sm_t[0][2] = trig_set( 2., FnCosSin,1,1, 0.);

. . .

/* As matrix */

sm_as[0][0] = trig_dup(sm_qb[0][0],0);
sm_as[0][1] = trig_dup(sm_qb[0][1],0);
sm_as[0][2] = trig_const(0.);

. . .
```

Symbolic matrix algebra is performed to derive $[T_s]$ and $\mathrm{Det}A_s$ by the instructions

```
sym_det_as = trig_mat_det(sm_as); /* determinant of As matrix */
trig_mat_adj(sm_adj_as,sm_as); /* Adjoint of As */

trig_mat_mult(sm1,sm_qb,sm_adj_as); /* Qb Adj As */
trig_mat_op(sm1,trig_expand);

trig_mat_op(sm_t,trig_expand);

trig_mat_mult(sm_ts,sm_t,sm1); /* Ts = T (Qb Adj As) */
trig_mat_op(sm_ts,trig_expand); /* simplify matrix entries */
trig_mat_op(sm_ts,trig_significant); /* discard near zero terms */
```

The symbolic stress $[\sigma_s] = [T_s][N]$ is determined by the instructions

```
sym_ss[0] = trig_add(nn[0],sm_ts[0][0],nn[1],sm_ts[0][1]);
trig_reassign(&sym_ss[0],trig_add(1.,sym_ss[0],nn[2],sm_ts[0][2]));

sym_ss[1] = trig_add(nn[0],sm_ts[1][0],nn[1],sm_ts[1][1]);
trig_reassign(&sym_ss[1],trig_add(1.,sym_ss[1],nn[2],sm_ts[1][2]));
```

```
sym_ss[2] = trig_add(nn[0],sm_ts[2][0],nn[1],sm_ts[2][1]);
trig_reassign(&sym_ss[2],trig_add(1.,sym_ss[2],nn[2],sm_ts[2][2]));
```

The quadratic coefficients of eqn. (2.17) are derived by substituting the stresses into the inequality (2.10) in the following manner.

```
sym_quad_b = sym_quad_c = TNull; /* initialize */

for (i = 0; i < 3; i++)
   {
   trig_reassign(&sym_quad_b,
      trig_add(1.,sym_quad_b,tf[i],sym_ss[i]));

   for (j = 0; j < 3; j++)
     {
     sym1 = trig_mult(sym_ss[i],sym_ss[j]);
     trig_reassign(&sym_quad_c,
        trig_add(1.,sym_quad_c,tff[i][j],sym1));
     trig_free(sym1);
     }
   }
```

where `tf[i]`, `tf[i][j]` are the strength parameters of the failure criterion (2.10).

The components h_{s1}, h_{s2} and h_{s3} in eqn. (2.18) are derived from the quadratic coefficients as follows.

```
sym_h1 = trig_dup(sym_quad_b,0);

sym1   = trig_mult(sym_quad_b,sym_quad_b);
sym_h2 = trig_add(1.,sym1,4.,sym_quad_c); /* h2 = discriminant */

sym_h3 = trig_mult_const(sym_det_as,2.); /* h3 = 2 det As */

/* now H = (h1 + sqrt(h2))/h3 */
```

The first and second derivatives of the components h_{s1}, h_{s2} and h_{s3} in eqn. (2.18) are derived by

```
sym_h1d = trig_diff(sym_h1); /* first derivatives */
sym_h2d = trig_diff(sym_h2);
sym_h3d = trig_diff(sym_h3);

. . .

sym_h1dd = trig_diff(sym_h1d); /* second derivatives */
sym_h2dd = trig_diff(sym_h2d);
sym_h3dd = trig_diff(sym_h3d);

. . .
```

Finally, the function `calc_trig_h(the)` is defined to evaluate the symbols h_{s1}, h_{s2} and h_{s3} and their derivatives at a given fibre angle θ and calculate H_{cr}, H'_{cr} and H''_{cr}.

```
real  calc_trig_h(the) /* calculate H, H' and H" for given theta */
real  the;
{
real  h1,h2,h3;
real  h1d,h2d,h3d;
real  h1dd,h2dd,h3dd;
real  hn,hnd,hndd;

h1    = trig_calc(sym_h1,the); /* evaluate symbloic series */
h2    = trig_calc(sym_h2,the);
h3    = trig_calc(sym_h3,the);

h1d   = trig_calc(sym_h1d,the); /* evaluate symbolic derivatives */
h2d   = trig_calc(sym_h2d,the);
h3d   = trig_calc(sym_h3d,the);

h1dd  = trig_calc(sym_h1dd,the);
h2dd  = trig_calc(sym_h2dd,the);
h3dd  = trig_calc(sym_h3dd,the);

hn    = h1 + sqrt(h2);
hnd   = h1d + .5/sqrt(h2)*h2d; /* 1st derivative of hn */
hndd  = h1dd + .5/sqrt(h2)*h2dd - .25*pow(h2,-1.5)*h2d*h2d;
          /* 2nd derivative of hn = h1 + sqrt(h2) */
```

```
hcr   = hn/h3; /* h_cr */
hd    = hnd/h3 - hn*h3d/h3/h3; /* 1st derivative of h_cr */
hdd   = hndd/h3 - 2*hnd*h3d/h3/h3
            - hn*h3dd/h3/h3 + 2*hn*h3d*h3d/h3/h3/h3;
              /* 2nd derivative of h_cr */

return (hcr);
}
```

2.5.2 Results

The results of the symbolic computation are illustrated by considering a balanced symmetric laminated plate. The laminate consists of four layers of equal thickness with fibres orientated at $\theta/-\theta/-\theta/\theta$, and is made of T300/5208 graphite/epoxy. The laminate is subjected to a loading $[N] = [50\ 100\ 10]^T$ MN/m. The elastic constants of this material are taken from Ref. [12] as $E_1 = 142$GPa, $E_2 = 10.8$GPa, $G_{12} = 5.49$GPa and $\upsilon_{12} = 0.3$, and the strength values as $X_t = 1568$MPa, $X_c = 1341$MPa, $Y_t = 57$MPa, $Y_c = 212$MPa, and $S = 80$MPa.

The symbolic form of H_{cr} in terms of θ is derived by the program described in Section 2.5.1 as

$$\begin{aligned} H_{cr} \;=\; & [\,11.79 + 6.6539\cos 2\theta + 1.5328\cos 4\theta - 3.9125\cos 6\theta \\ & -5.2368\cos 8\theta - 0.58976\sin 2\theta - 0.093517\sin 6\theta + \\ & \sqrt{(\,785.63 + 722.95\cos 2\theta - 194.81\cos 4\theta} \\ & \quad -644.8\cos 6\theta - 459.63\cos 8\theta + 14.88\cos 10\theta \\ & \quad +59.553\cos 12\theta + 40.009\cos 14\theta + 27.18\cos 16\theta \\ & \quad -19.92\sin 2\theta - 10.243\sin 4\theta - 9.2797\sin 6\theta \\ & \quad +2.5187\sin 8\theta + 4.3651\sin 10\theta + 0.54685\sin 12\theta \\ & \quad +0.72858\sin 14\theta\,)\,] \\ & /\,(46.107 - 23.952\cos 4\theta - 5.2004\cos 8\theta) \end{aligned} \tag{2.20}$$

This computation is performed in under $1\frac{1}{2}$ seconds on a 386 Personal Computer. It is found that *Mathematica* [23], a general purpose symbolic computation system, is two orders of magnitude slower to derive this expression for H_{cr}.

The optimal fibre angle θ_{opt} may be computed from eqn. (2.20) using the Golden Section method. Alternatively, since the first and second derivatives of H_{cr} with

respect to θ are also determined exactly, we may equate H'_{cr} to zero and employ Newton's method to find θ_{opt}, the fibre angle at which the first derivative vanishes, using H''_{cr}. It is found that $\theta_{opt} = 54.476$ degrees and $H_{min}(\theta_{opt}) = 17.5$mm.

The stress $[\sigma_s]$, the determinant $DetA_s$ and the derivatives of the component functions h_{si} are derived as

$$\begin{aligned}
\sigma_{s1} &= 2.5176\ 10^{15} - 5.1442\ 10^{14}\cos 2\theta - 1.8999\ 10^{15}\cos 4\theta \\
&\quad +3.0248\ 10^{14}\cos 6\theta + 1.8087\ 10^{13}\cos 8\theta \\
&\quad +7.3099\ 10^{14}\sin 2\theta + 1.1591\ 10^{14}\sin 6\theta
\end{aligned}$$

$$\begin{aligned}
\sigma_{s2} &= 9.4041\ 10^{14} + 5.1442\ 10^{14}\cos 2\theta \\
&\quad +1.0351\ 10^{14}\cos 4\theta - 3.0248\ 10^{14}\cos 6\theta \\
&\quad -4.0811\ 10^{14}\cos 8\theta - 3.9826\ 10^{13}\sin 2\theta \\
&\quad -6.3151\ 10^{12}\sin 6\theta
\end{aligned}$$

$$\begin{aligned}
\tau_{s12} &= 6.3826\ 10^{14}\sin 2\theta + 5.7551\ 10^{14}\sin 4\theta \\
&\quad -1.7247\ 10^{14}\sin 6\theta - 2.131\ 10^{14}\sin 8\theta \\
&\quad +7.5665\ 10^{13}\cos 2\theta + 9.1092\ 10^{12}\cos 6\theta
\end{aligned}$$

$$\mathrm{Det}A_s = 2.3053\ 10^{13} - 1.1976\ 10^{13}\cos 4\theta - 2.6002\ 10^{12}\cos 8\theta$$

$$\begin{aligned}
h'_1 &= -1.3307\ 10^{13}\sin 2\theta - 6.1314\ 10^{12}\sin 4\theta \\
&\quad +2.3475\ 10^{13}\sin 6\theta + 4.1894\ 10^{13}\sin 8\theta \\
&\quad -1.1795\ 10^{12}\cos 2\theta - 5.611\ 10^{11}\cos 6\theta
\end{aligned}$$

$$\begin{aligned}
h'_2 &= -1.4459\ 10^{27}\sin 2\theta + 7.7924\ 10^{26}\sin 4\theta \\
&\quad +3.8688\ 10^{27}\sin 6\theta + 3.677\ 10^{27}\sin 8\theta \\
&\quad -1.488\ 10^{26}\sin 10\theta - 7.1463\ 10^{26}\sin 12\theta \\
&\quad -5.6013\ 10^{26}\sin 14\theta - 4.3489\ 10^{26}\sin 16\theta \\
&\quad -3.9841\ 10^{25}\cos 2\theta - 4.0975\ 10^{25}\cos 4\theta \\
&\quad -5.5678\ 10^{25}\cos 6\theta + 2.0149\ 10^{25}\cos 8\theta \\
&\quad +4.3651\ 10^{25}\cos 10\theta + 6.5622\ 10^{24}\cos 12\theta \\
&\quad +1.02\ 10^{25}\cos 14\theta
\end{aligned}$$

$$h'_3 = 9.5808\ 10^{13}\sin 4\theta + 4.1603\ 10^{13}\sin 8\theta$$

$$
\begin{aligned}
h_1'' &= -2.6615\ 10^{13}\cos 2\theta - 2.4525\ 10^{13}\cos 4\theta \\
&\quad +1.4085\ 10^{14}\cos 6\theta + 3.3515\ 10^{14}\cos 8\theta \\
&\quad +2.359\ 10^{12}\sin 2\theta + 3.3666\ 10^{12}\sin 6\theta
\end{aligned}
$$

$$
\begin{aligned}
h_2'' &= -2.8918\ 10^{27}\cos 2\theta 3.1169\ 10^{27}\cos 4\theta \\
&\quad +2.3213\ 10^{28}\cos 6\theta + 2.9416\ 10^{28}\cos 8\theta \\
&\quad -1.488\ 10^{27}\cos 10\theta - 8.5756\ 10^{27}\cos 12\theta \\
&\quad -7.8418\ 10^{27}\cos 14\theta - 6.9582\ 10^{27}\cos 16\theta \\
&\quad +7.9681\ 10^{25}\sin 2\theta + 1.639\ 10^{26}\sin 4\theta \\
&\quad +3.3407\ 10^{26}\sin 6\theta - 1.6119\ 10^{26}\sin 8\theta \\
&\quad -4.3651\ 10^{26}\sin 10\theta - 7.8746\ 10^{25}\sin 12\theta \\
&\quad -1.428\ 10^{26}\sin 14\theta
\end{aligned}
$$

$$
h_3'' = 3.8323\ 10^{14}\cos 4\theta + 3.3282\ 10^{14}\cos 8\theta
$$

2.6 Laminated Pressure Vessels

This section is concerned with the optimization of composite pressure vessels subject to the Tsai–Wu failure criterion and considers problems of maximum internal pressure and minimum weight. In the first problem, the fibre orientation is determined for balanced and unbalanced laminations to maximize the internal pressure. The effects of axial and torsional forces on the optimum design are discussed. It is shown that the axial force affects the optimum fibre angle differently for shells with single and multiple layers.

In the second problem, the design objective is the minimization of the weight of a liquid filled pressure vessel taking both the fibre orientation and the wall thickness as design variables. Both the constant and variable wall thickness cases are discussed. Comparative numerical results are presented for single and multiple layered vessels. It is noted that methods used in both design problems can be easily implemented in practical design situations.

In this study, the relationship between the loading parameters and the material stress is combined and simplified into one transformation matrix using the special purpose symbolic computation routines presented in the previous section, in order to improve the computational efficiency of the optimization procedure.

2.6.1 Basic Equations

The pressure vessel is modelled as a symmetrically laminated cylindrical shell of thickness H, length L and radius R where R refers to the radius of the middle surface. The shell is constructed of an even number of orthotropic layers of equal thickness t. The fibre orientation θ is defined as the angle between the fibre direction and the longitudinal axis x. The fibre angles are orientated symmetrically with respect to the middle surface such that $\theta_k = (-1)^{k-1}\theta$ for $k \leq n/2$ and $\theta_k = (-1)^k\theta$ for $k \geq n/2+1$ where $k = 1, 2, \ldots, n$ is the layer number and n is the total number of layers. It is noted that $n = 2$ corresponds to a single lamina of thickness $H = 2t$ and fibre orientation θ. The coordinate axes x,ϕ and z refer to the longitudinal, circumferential and radial directions respectively, with the origin lying in the middle surface of the shell.

Due to the symmetry of the lamination, the force resultants in the geometric coordinate axes are given by

$$[N] = [A][\epsilon] \tag{2.21}$$

where

$$[N] = \begin{pmatrix} N_x \\ N_\phi \\ N_{x\phi} \end{pmatrix}, \; [A] = \begin{pmatrix} A_{11} & A_{12} & A_{16} \\ A_{12} & A_{22} & A_{26} \\ A_{16} & A_{26} & A_{66} \end{pmatrix}, \; [\epsilon] = \begin{pmatrix} \epsilon_x \\ \epsilon_\phi \\ \gamma_{x\phi} \end{pmatrix} \tag{2.22}$$

In eqn. (2.22), A_{ij} are the extensional stiffnesses given by $A_{ij} = H\bar{Q}_{ij}(\theta)$ for $i, j = 1, 2$ and $i = j = 6$, $A_{i6} = 2t\bar{Q}_{i6}(\theta)$ for unbalanced laminates and $A_{i6} = 0$ for balanced laminates with $i = 1, 2$. Also in eqn. (2.22), ϵ_x, ϵ_ϕ and $\gamma_{x\phi}$ denote the normal and shear strains. Here $\bar{Q}_{ij}(\theta)$ is the transformed reduced stiffness component.

The stress–strain equations for the k–th orthotropic layer are given by

$$[s^{(k)}] = [\bar{Q}_{ij}^{(k)}][\epsilon] \tag{2.23}$$

where $[\epsilon] = [A]^{-1}[N]$ from eqn. (2.21), and

$$[s^{(k)}] = [\sigma_x^{(k)} \; \sigma_\phi^{(k)} \; \tau_{x\phi}^{(k)}]^T$$

denotes the stress vector in the $x\phi$ coordinate system.

The stress vector in the material coordinate system, denoted by

$$[\sigma^{(k)}] = [\sigma_1^{(k)} \; \sigma_2^{(k)} \; \tau_{12}^{(k)}]^T$$

is obtained from the *geometric* stress vector $[s^{(k)}]$ via the matrix transformation

$$[\sigma^{(k)}] = [T^{(k)}][s^{(k)}] \tag{2.24}$$

where $[T^{(k)}] = [T(\theta_k)]$ denotes the transformation matrix for the k-th layer. From eqns. (2.23) and (2.24) it follows that

$$[\sigma^{(k)}] = [T^{(k)}][\bar{Q}_{ij}^{(k)}][\epsilon] \tag{2.25}$$

The design against failure is determined by employing a suitable failure criterion. In this study, the Tsai–Wu failure criterion (2.10) is used.

The problem formulation and the performance index depend on the nature of the specific design problem. The problem statement involves maximizing or minimizing a cost function subject to the strength constraint given by the criterion (2.10). The optimization procedure is applied to two design problems.

2.6.2 Problem 2.1: Design for Maximum Internal Pressure

We consider a cylindrical pressure vessel with closed ends and subject to an internal pressure p, axial force F and torque T. The first design problem involves determining the fibre orientation θ so as to maximize the internal pressure p for a given laminate thickness H under the forces F and T such that the optimal design satisfies the strength criterion (2.10).

Method of solution

The force resultants for this problem are given by

$$N_x = \frac{pR}{2} - \frac{F}{2\pi R}, \quad N_\phi = pR, \quad N_{x\phi} = \frac{T}{2\pi R^2} \tag{2.26}$$

The vector $[N] = [N_x\ N_\phi\ N_{x\phi}]^T$ can be expressed as a sum of two components: one due to the internal pressure p, and the other due to the external forces F and T, viz.

$$[N] = [N]_p\, p + [N]_f \tag{2.27}$$

where $[N]_p$ is the coefficient vector of p, and $[N]_f$ incorporates the external forces.

From eqns. (2.26) and (2.27), it follows that

$$[N]_p = \begin{bmatrix} \frac{1}{2} \\ 1 \\ 0 \end{bmatrix} R, \quad [N]_f = \begin{bmatrix} -FR \\ 0 \\ T \end{bmatrix} \frac{1}{2\pi R^2} \tag{2.28}$$

Similarly, the strain vector $[\epsilon]$ may be expressed as

$$[\epsilon] = [\epsilon]_p \, p + [\epsilon]_f \tag{2.29}$$

where $[\epsilon]_p = [A]^{-1}[N]_p$ and $[\epsilon]_f = [A]^{-1}[N]_f$, which follows from eqns. (2.21) and (2.27). Now the stresses in the material coordinates can be computed by inserting $[\epsilon]$ from eqn. (2.29) into eqn. (2.25) which gives

$$[\sigma^{(k)}] = [\sigma^{(k)}]_p \, p + [\sigma^{(k)}]_f \tag{2.30}$$

where

$$[\sigma^{(k)}]_p = [T^{(k)}][\bar{Q}_{ij}^{(k)}][\epsilon]_p, \quad [\sigma^{(k)}]_f = [T^{(k)}][\bar{Q}_{ij}^{(k)}][\epsilon]_f \tag{2.31}$$

We substitute the stresses from eqn. (2.30) into the strength constraint (2.10) and obtain a quadratic failure criterion in terms of the internal pressure p as given by

$$\begin{aligned} &\{F_{11}(\sigma_{1p}^{(k)})^2 + F_{22}(\sigma_{2p}^{(k)})^2 + F_{66}(\tau_{12p}^{(k)})^2 + 2F_{12}\sigma_{1p}^{(k)}\sigma_{2p}^{(k)}\} \, p^2 \\ &+ \{2F_{11}\sigma_{1p}^{(k)}\sigma_{1f}^{(k)} + 2F_{22}\sigma_{2p}^{(k)}\sigma_{2f}^{(k)} + 2F_{66}\tau_{12p}^{(k)}\tau_{12f}^{(k)} \\ &\quad +2F_{12}(\sigma_{1p}^{(k)}\sigma_{2f}^{(k)} + \sigma_{2p}^{(k)}\sigma_{1f}^{(k)}) + F_1\sigma_{1p}^{(k)} + F_2\sigma_{2p}^{(k)}\} \, p \\ &+ \{F_{11}(\sigma_{1f}^{(k)})^2 + F_{22}(\sigma_{2f}^{(k)})^2 + F_{66}(\tau_{12f}^{(k)})^2 + 2F_{12}\sigma_{1f}^{(k)}\sigma_{2f}^{(k)} \\ &\quad +F_1\sigma_{1f}^{(k)} + F_2\sigma_{2f}^{(k)} - 1\} = 0 \end{aligned} \tag{2.32}$$

where $[\sigma^{(k)}]_p = [\sigma_{1p}^{(k)} \; \sigma_{2p}^{(k)} \; \tau_{12p}^{(k)}]^T$ and $[\sigma^{(k)}]_f = [\sigma_{1f}^{(k)} \; \sigma_{2f}^{(k)} \; \tau_{12f}^{(k)}]^T$.

Solving the quadratic equation (2.32) for the k-th layer yields the burst pressure $p_{cr}^{(k)} = p_{cr}^{(k)}(\theta; F, T)$ corresponding to that layer. The burst pressure of the vessel is given by

$$p_{cr} = \min_k p_{cr}^{(k)} \qquad (k = 1, 2, \ldots, n) \tag{2.33}$$

If no positive real solution of eqn. (2.32) exists, then the pressure vessel fails under external load only, and the solution of the design problem does not exist as there is no feasible design satisfying the constraint (2.10).

Optimal design problem

The design objective is the maximization of the burst pressure p_{cr} subject to the failure criterion (2.10). The optimization is carried over the fibre orientation θ. The design problem can be stated as

$$p_{max} \stackrel{\text{def}}{=} \max_{\theta} p_{cr}(\theta; F, T) = \max_{\theta} \min_{k} p_{cr}^{(k)} \tag{2.34}$$

where $p_{cr}(\theta; F, T)$ is given by eqn. (2.33). The maximum burst pressure p_{max} is determined by solving the max–min problem (2.34) which also yields the optimal fibre orientation θ_{opt}.

The optimization procedure involves the stages of evaluating the burst pressure p_{cr} for a given θ and iteratively improving θ_{opt} to maximize p_{cr}. Thus the computational solution consists of successive stages of analysis and optimization until convergence is obtained. The optimization stage employs the Golden Section method in determining θ_{opt}.

Numerical results for Problem 2.1

The optimization of the laminated pressure vessel is illustrated by considering a cylindrical shell of mean radius $R = 1$m and thickness $H = 0.01$m. The laminate is made of T300/5208 graphite/epoxy the elastic constants of which are $E_1 = 142$GPa, $E_2 = 10.8$GPa, $G_{12} = 5.49$GPa, and $\upsilon_{12} = 0.3$. The strength values are $X_t = 1568$MPa, $X_c = 1341$MPa, $Y_t = 57$MPa, $Y_c = 212$MPa, and $S = 80$MPa. The values for the material properties are taken from Ref. [12].

We first investigate the effect of fibre orientation on the burst pressure p_{cr} for different values of the axial force. Figure 2.1 on Page 36 shows the curves of p_{cr} versus θ for single–layered, four–layered and six–layered laminates with $T = 0$ for $F = 0$ and $F = 5$MN. It is noted that the results for the four–layered (balanced) laminate are applicable to balanced laminates with any number of layers. For single–layered construction, it is observed that $\theta_{opt} = 0$ for $F = 0$ and $\theta_{opt} = 90^\circ$ for $F = 5$MN. The burst pressure p_{cr} is much higher for multilayered laminates with the balanced case giving the highest burst pressure. The effects of the axial force and torque on θ_{opt} and p_{max} are investigated in Table 2.1. For single–layered laminates, $\theta_{opt} = 0$ for low values of F and jumps to 90° at a certain value of $F > 0$ which depends on the amount of torque applied. For multilayered laminates, the fibres align themselves with the longitudinal axis x as F increases. This result is to be expected on physical grounds.

Table 2.1: Optimal fibre angles and maximum pressure (Problem 2.1)

F (MN)	T (MNm)	Single layer (unbalanced)		4 layers (balanced)		6 layers (unbalanced)	
		θ_{opt}	p_{max} (MPa)	θ_{opt}	p_{max} (MPa)	θ_{opt}	p_{max} (MPa)
0	0	90.00°	1.19	54.39°	5.36	54.21°	4.17
1	0	90.00°	0.86	53.56°	5.16	53.21°	3.98
5	0	0.00°	0.59	50.25°	4.40	49.02°	3.29
10	0	0.00°	0.60	45.94°	3.67	43.80°	2.68
0	2	90.00°	1.03	54.32°	5.11	54.07°	3.77
1	2	90.00°	0.70	53.50°	4.91	52.98°	3.57
5	2	0.00°	0.52	50.00°	4.15	48.38°	2.90
10	2	0.00°	0.53	45.54°	3.44	42.77°	2.33
0	4	90.00°	0.51	54.23°	4.83	53.90°	3.34
1	4	0.00°	0.25	53.36°	4.62	52.69°	3.15
5	4	0.00°	0.26	49.69°	3.87	47.59°	2.49
10	4	0.00°	0.27	45.01°	3.19	41.56°	1.98

Failure surfaces with respect to maximum pressure are given in Figures 2.2 and 2.3 for single– and four–layered laminates, respectively. Figure 2.2 indicates that there is a sharp drop in p_{max} as F increases. Decrease in p_{max} with respect to torque is more gradual. For the balanced laminate with four layers, the failure surface as shown in Figure 2.3 is rather flat with gradual decrease in p_{max} with increasing axial force and torque.

2.6.3 Problem 2.2: Design for Minimum Weight

As our second problem, we consider a circular cylindrical shell of length L filled with a liquid of specific weight ρ_l and under a given internal pressure. The design problem involves optimizing the fibre orientation θ so as to minimize the weight of the liquid tank for a given pressure. It is noted that the weight of the tank can be evaluated in terms of the shell thickness H.

Method of solution

The force resultants for this problem are derived in Refs. [13] and [14]. For a cylindrical tank with bulkheads attached to the ends of the cylinder, these forces are

$$\begin{aligned} N_x &= \frac{p_c R}{2} + \frac{\rho_l}{8}\cos\phi(4x^2 - L^2 - 2R^2) \\ N_\phi &= p_c R - \rho_l R^2 \cos\phi \\ N_{x\phi} &= -\rho_l R x \sin\phi \end{aligned} \tag{2.35}$$

where $p_c \geq \rho_l R$ is the pressure at the center of the cylinder and x is the longitudinal axis with the origin located at the mid-point such that $-L/2 \leq x \leq L/2$.

We note that $A_{ij} = H\eta_{ij}\bar{Q}_{ij}(\theta)$ where $\eta_{ij} = 1$ for $i,j = 1,2$ and $i = j = 6$, $\eta_{i6} = 2/n$ for unbalanced laminates and $\eta_{i6} = 0$ for balanced laminates with $i = 1,2$. We define a matrix $[a]$ such that $[a] = H^{-1}[A]$. Thus $a_{ij} = \bar{Q}_{ij}(\theta)$ for $ij = 11, 12, 22$ and 66, and $a_{i6} = \eta_{i6}\bar{Q}_{i6}(\theta)$ for $i = 1,2$. From eqn. (2.21), it follows that $[\epsilon] = H^{-1}[a]^{-1}[N]$ where $[N]$ is defined by eqns. (2.22) and (2.35). Substituting $[\epsilon]$ into eqn. (2.25), we find

$$[\sigma^{(k)}] = H^{-1}[\sigma_0^{(k)}] \tag{2.36}$$

where

$$[\sigma_0^{(k)}] = [T^{(k)}][\bar{Q}_{ij}^{(k)}][a]^{-1}[N] \tag{2.37}$$

We substitute the stresses from eqn. (2.36) into the strength constraint (2.10) and obtain a quadratic failure criterion in terms of the shell thickness H as given by

$$\begin{aligned} &\{F_{11}(\sigma_{10}^{(k)})^2 + F_{22}(\sigma_{20}^{(k)})^2 + F_{66}(\tau_{120}^{(k)})^2 + 2F_{12}\sigma_{10}^{(k)}\sigma_{20}^{(k)}\} \\ &+\{F_1\sigma_{10}^{(k)} + F_2\sigma_{20}^{(k)}\}H - H^2 = 0 \end{aligned} \tag{2.38}$$

The solution of eqn. (2.38) gives for any x and ϕ the minimum shell thickness $H_{cr}^{(k)}$ corresponding to the failure of the k-th layer. From eqn. (2.38), it follows that the critical thickness $H_{cr} = H_{cr}(\theta;\ x)$ at a point x is given by

$$H_{cr} = \max_{\phi,k} H_{cr}^{(k)} \qquad (k = 1,2,\ldots,n;\ 0 \leq \phi \leq 2\pi) \tag{2.39}$$

It is noted that the critical thickness H_{cr} depends on the location x along the cylindrical shell as well as the internal pressure p_c and the specific weight ρ_l of the liquid.

Optimal design problem

The design objective for the cylindrical liquid tank problem is the minimization of the shell weight with the thickness subject to the strength condition (2.10). The weight of the shell is given by

$$W(\theta) = 2\pi R\rho_t \int_{-L/2}^{L/2} H(\theta; x)dx \tag{2.40}$$

where ρ_t is the specific weight of the fibre composite material used in the construction of the tank.

Two distinct cases depending on whether the shell thickness is constant or variable over the length $-L/2 \leq x \leq L/2$ are considered.

Case I. Constant thickness tank

In this case $H = H(\theta)$ and the weight is given by

$$W(\theta) = 2\pi RL\rho_t H(\theta) \tag{2.41}$$

Since the weight is proportional to the thickness, it is sufficient to minimize $H(\theta)$ to obtain the minimum weight design. $H_{min}(\theta)$ for a given θ valid for all x is determined from

$$H_{min}(\theta) = \max_x H_{cr} = \max_x (\max_{\phi,k} H_{cr}^{(k)})$$
$$(-L/2 \leq x \leq L/2, \quad 0 \leq \phi \leq 2\pi) \tag{2.42}$$

where $H_{cr}^{(k)}$ is determined from eqn. (2.38).

Case II. Variable thickness tank

In this case $H = H(\theta; x)$ and the minimum thickness $H_{min}(\theta; x)$ at a point x for a given θ is defined by H_{cr} in eqn. (2.39). Therefore $H_{min}(\theta; x)$ is determined as the maximum of $H_{cr}^{(k)}$ given by eqn. (2.39) at every point x producing a variable wall thickness. Thus

$$H_{min}(\theta; x) = \max_{\phi,k} H_{cr}^{(k)} \qquad (0 \leq \phi \leq 2\pi) \tag{2.43}$$

Due to symmetry, the thickness distributions are the same for $-L/2 \leq x \leq 0$ and $0 \leq x \leq L/2$. For this case, the weight is given by eqn. (2.40).

In both cases, the design problem is to determine the optimal fibre orientation θ_{opt} so as to minimize the weight of the shell, viz.

$$W_{min} = \min_\theta W(\theta) \tag{2.44}$$

with H_{min} obtained from eqn. (2.42) in Case I and from eqn. (2.43) in Case II. In eqn. (2.44), $W(\theta)$ is given by eqn. (2.41) for the constant thickness case and by eqn. (2.40) for the variable thickness case.

The minimum weight problem is solved by determining the minimum thickness H_{min} satisfying the constraint (2.38) from eqn. (2.42) (Case I), or from eqn. (2.43) (Case II). The weight is minimized over the fibre orientation θ by using a one–dimensional numerical optimization scheme, viz. the Golden Section method. Computations are continued until convergence is attained for θ.

Table 2.2: Optimal fibre angles and minimum weight for a single–layered constant thickness pressure vessel (Problem 2.2)

Single layer			$w_c(\theta)/w_{c,min}$				
p_0	θ_{opt}	$w_{c,min}$	θ				
		$(\times 10^6)$	0°	30°	45°	60°	90°
1	0.00°	90.44	1.00	1.36	1.37	1.27	1.29
2	0.00°	128.14	1.00	1.24	1.22	1.11	1.07
3	90.00°	157.93	1.08	1.23	1.20	1.07	1.00
4	90.00°	178.62	1.19	1.30	1.24	1.10	1.00
5	90.00°	199.31	1.28	1.35	1.28	1.12	1.00
6	90.00°	220.00	1.36	1.39	1.31	1.14	1.00
8	90.00°	261.38	1.47	1.46	1.35	1.17	1.00
10	90.00°	302.77	1.55	1.51	1.39	1.19	1.00
20	90.00°	509.76	1.75	1.64	1.47	1.25	1.00
30	90.00°	716.76	1.84	1.70	1.51	1.27	1.00
40	90.00°	923.77	1.89	1.73	1.53	1.28	1.00
50	90.00°	1130.79	1.92	1.75	1.54	1.29	1.00
100	90.00°	2165.87	1.99	1.79	1.57	1.31	1.00
200	90.00°	4236.06	2.02	1.81	1.58	1.32	1.00

Numerical results for Problem 2.2

Numerical results are given for single– and four–layered laminated cylinders made of the same graphite/epoxy material defined in Section 2.6.2. The numerical values are given for dimensionless quantities by introducing

$$X = x/L, \quad h = H/L, \quad r = R/L$$

Table 2.3: Optimal fibre angles and minimum weight for a single–layered variable thickness pressure vessel (Problem 2.2)

Single layer			$w_v(\theta)/w_{v,min}$				
p_0	θ_{opt}	$w_{v,min}$	θ				
		$(\times 10^6)$	0°	30°	45°	60°	90°
1	0.00°	84.71	1.00	1.22	1.27	1.25	1.09
2	90.00°	111.23	1.14	1.25	1.25	1.19	1.00
3	90.00°	131.01	1.29	1.34	1.31	1.22	1.00
4	90.00°	151.14	1.40	1.41	1.35	1.24	1.00
5	90.00°	171.47	1.48	1.46	1.38	1.25	1.00
6	90.00°	191.93	1.55	1.50	1.40	1.26	1.00
8	90.00°	233.05	1.64	1.56	1.44	1.27	1.00
10	90.00°	274.32	1.70	1.60	1.46	1.28	1.00
20	90.00°	481.18	1.86	1.70	1.52	1.30	1.00
30	90.00°	688.19	1.92	1.74	1.54	1.31	1.00
40	90.00°	895.20	1.95	1.76	1.55	1.31	1.00
50	90.00°	1102.22	1.97	1.78	1.56	1.31	1.00
100	90.00°	2137.31	2.01	1.81	1.58	1.32	1.00
200	90.00°	4207.49	2.03	1.82	1.59	1.32	1.00

$$p_0 = p_c/\rho_l R, \quad w = W/2\pi RL^2\rho_t \tag{2.45}$$

From eqns. (2.40) and (2.45), it follows that

$$w(\theta) = \int_{-1/2}^{1/2} h(\theta; X)dX \tag{2.46}$$

We note that for the constant thickness shell (Case I) $w(\theta) = h(\theta)$.

In Tables 2.2–2.5, subscripts c and v refer to the constant and variable thickness cases, respectively. In particular $w_v(\theta)$ refers to the weight of a shell with the thickness function $H_{min}(\theta; x)$ obtained from eqn. (2.43) and the fibre orientation specified as θ.

The effect of increasing the internal pressure p_0 on the optimal design is investigated in Tables 2.2 and 2.3 for single–layered constant and variable thickness shells, respectively. It is observed that θ_{opt} is 0° for low values of p_0 and jumps to 90° as p_0 increases. The weight difference between the constant and variable thickness shells decreases with increasing p_0. The right half of the tables is provided to compare the weight ratios of shells with specified and optimal fibre angles.

Table 2.4: Optimal fibre angles and minimum weight for a four–layered constant thickness pressure vessel (Problem 2.2)

Four layers			$w_c(\theta)/w_{c,min}$				
p_0	θ_{opt}	$w_{c,min}$	θ				
		$(\times 10^6)$	0°	30°	45°	60°	90°
1	44.96°	24.88	3.64	2.33	1.00	2.78	4.69
2	48.72°	30.64	4.18	2.78	1.14	2.47	4.48
3	49.84°	34.77	4.91	3.24	1.30	2.37	4.54
4	50.50°	38.52	5.53	3.63	1.43	2.31	4.64
5	50.93°	42.20	6.06	3.97	1.55	2.27	4.72
6	51.22°	45.93	6.50	4.24	1.65	2.24	4.79
8	51.60°	53.65	7.15	4.65	1.79	2.17	4.87
10	51.82°	61.76	7.59	4.92	1.89	2.11	4.90
20	52.52°	106.27	8.42	5.44	2.06	1.88	4.80
30	52.97°	152.05	8.68	5.60	2.11	1.77	4.71
40	53.30°	197.86	8.83	5.68	2.14	1.72	4.67
50	53.51°	243.64	8.92	5.74	2.16	1.69	4.64
100	53.94°	472.46	9.10	5.85	2.20	1.62	4.58
200	54.16°	929.98	9.20	5.92	2.22	1.58	4.56

Tables 2.4 and 2.5 give the same information as Tables 2.2 and 2.3 for balanced four-layered shells. It is observed that as p_0 increases, the optimal fibre angle approaches θ_{opt} of the first problem with $F = T = 0$ (see Table 2.1). This is to be expected since the contribution of the liquid to resultant forces becomes less pronounced as the internal pressure increases as is evident from eqn. (2.35) and Problems 2.1 and 2.2 converge. Comparison of Tables 2.4 and 2.5 indicates that w_{min} values differ by about 20% for constant and variable thickness shells for small values of p_0. This difference decreases as p_0 increases and drops to less than 2% for $p_0 > 20$.

Figure 2.4 shows the optimal thickness distribution of the variable thickness shell with respect to the x axis and for increasing internal pressure.

Table 2.5: Optimal fibre angles and minimum weight for a four–layered variable thickness pressure vessel (Problem 2.2)

Four layers			$w_v(\theta)/w_{v,min}$				
p_0	θ_{opt}	$w_{v,min}$	θ				
		$(\times 10^6)$	0°	30°	45°	60°	90°
1	46.42°	20.03	4.23	2.24	1.03	2.46	4.60
2	48.81°	25.05	5.06	2.88	1.20	2.23	4.44
3	49.94°	29.27	5.79	3.40	1.36	2.13	4.48
4	50.59°	33.35	6.36	3.80	1.49	2.08	4.53
5	50.98°	37.46	6.80	4.11	1.60	2.03	4.58
6	51.25°	41.66	7.13	4.36	1.68	1.99	4.61
8	51.60°	50.31	7.60	4.69	1.79	1.92	4.63
10	51.95°	59.22	7.90	4.91	1.87	1.87	4.63
20	53.08°	104.35	8.56	5.41	2.04	1.73	4.61
30	53.49°	149.83	8.81	5.59	2.10	1.68	4.59
40	53.71°	195.42	8.93	5.69	2.14	1.64	4.58
50	53.84°	241.08	9.01	5.74	2.16	1.63	4.57
100	54.11°	469.62	9.16	5.86	2.20	1.59	4.55
200	54.25°	927.00	9.23	5.92	2.22	1.57	4.54

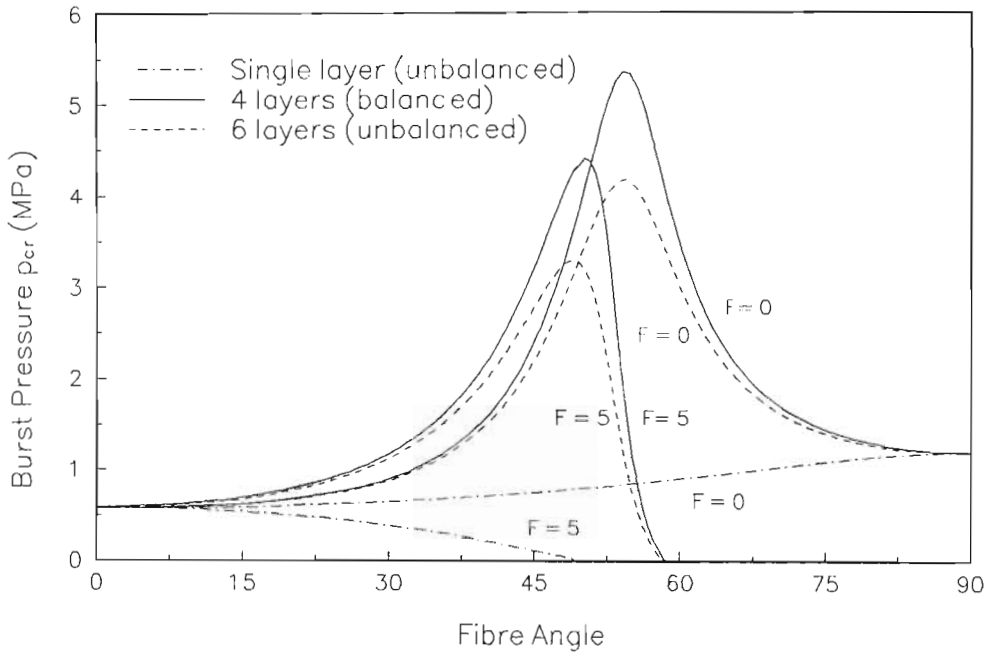


Figure 2.1. Curves of burst pressure versus fibre angle with $T = 0$ (Problem 2.1)

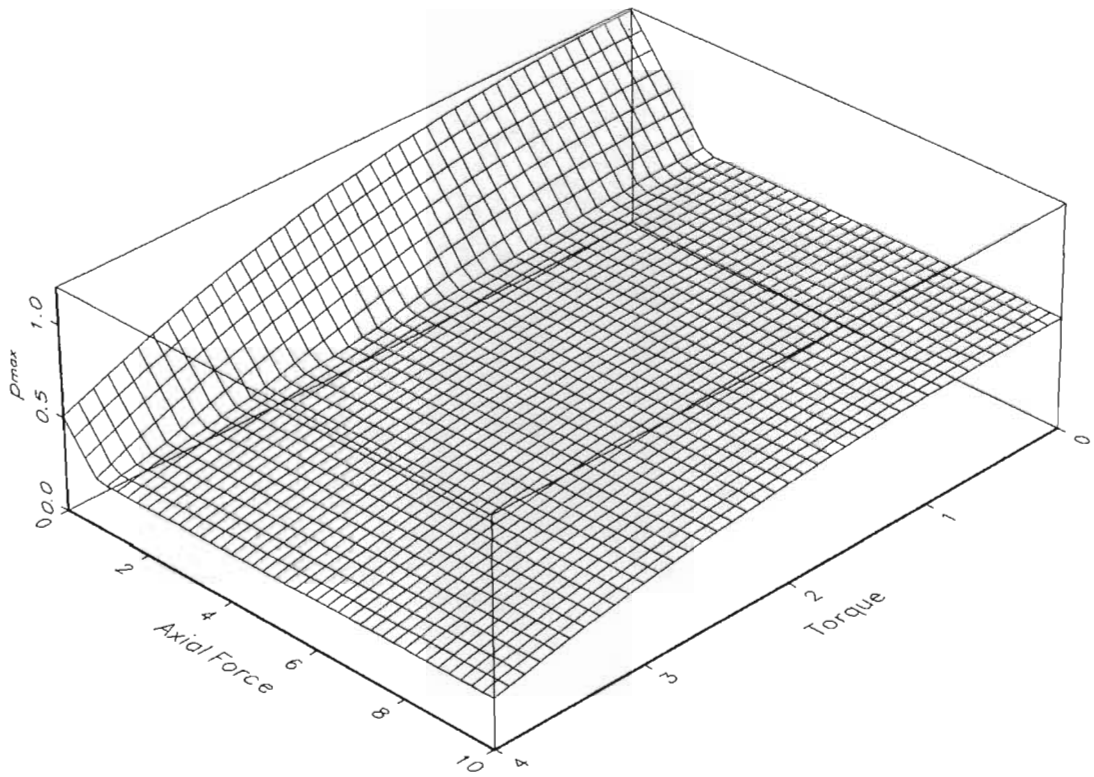


Figure 2.2. Surface of maximum pressure with respect to axial force and torque for a single–layered pressure vessel (Problem 2.1)

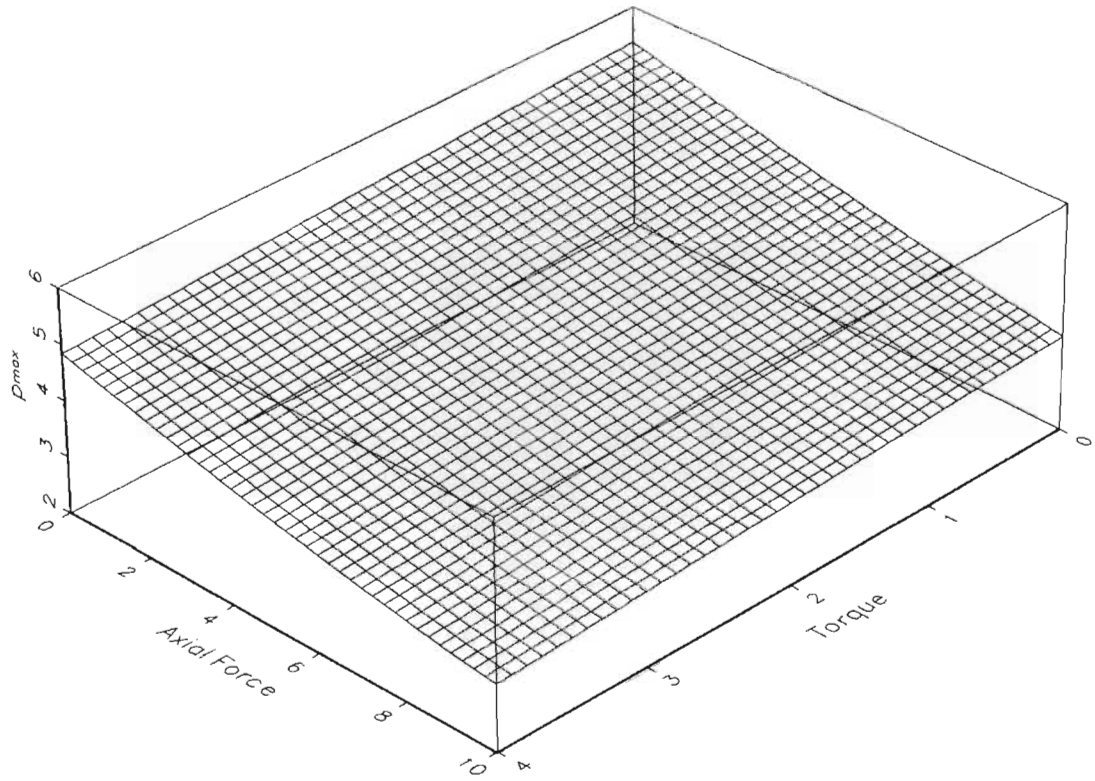


Figure 2.3. Surface of maximum pressure with respect to axial force and torque for a four–layered pressure vessel (Problem 2.1)

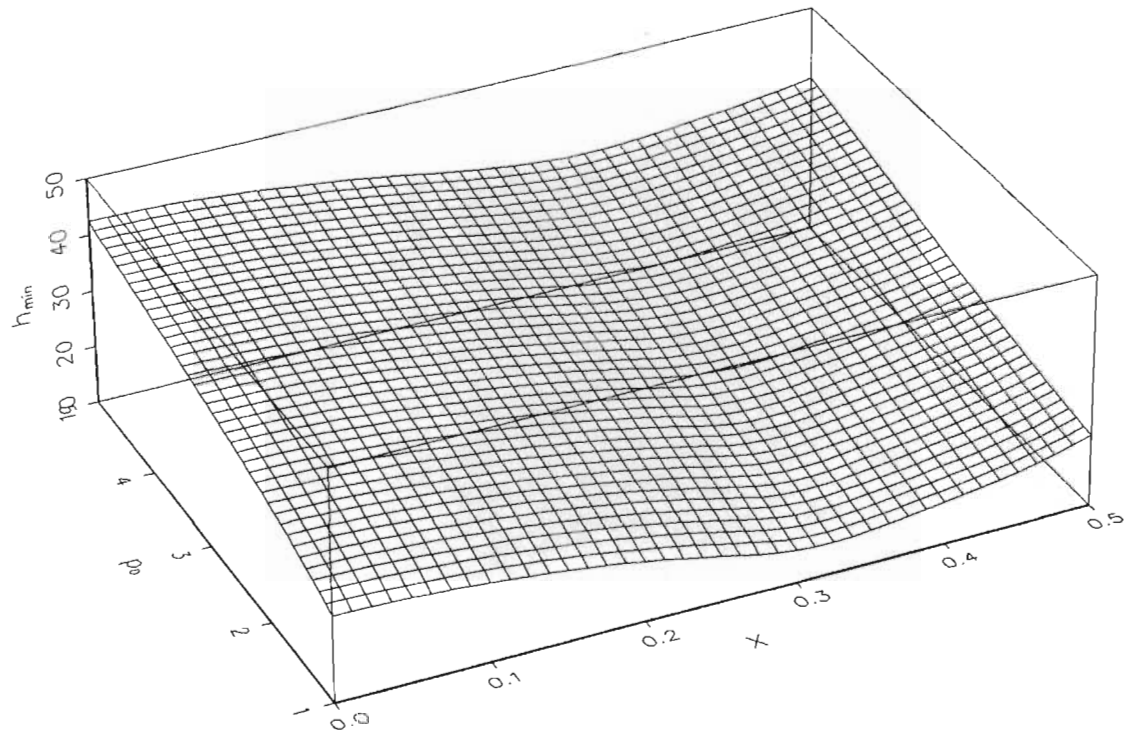


Figure 2.4. Optimal thickness distribution with respect to x axis and internal pressure for a four-layered pressure vessel (Problem 2.2)

2.7 Conclusions

In design optimization problems requiring symbolic computation, it may be necessary to integrate such computations into an iterative solution procedure to improve the computational efficiency. This is not possible using a closed general purpose symbolic computation package. In addition, proprietary general purpose symbolic computation tools often have high overheads in terms of cost, hardware requirements and processing time. These drawbacks may be overcome by developing special purpose symbolic computation software which is tailored according to the requirements of the specific problem. The development of such software can be realised for a given problem by noting that, in general, a specific class of functions will be needed in the solution of a particular problem.

The design problems studied in this Chapter require tedious matrix algebra where the entries are series of double trigonometric functions of the fibre angle. Special purpose routines are developed to process such expressions. The routines perform matrix algebra involving matrices of trigonometric series and simplify the results using trigonometric identities.

It is found that the special purpose symbolic computation is two orders of magnitude more efficient than *Mathematica*. The efficiency of special purpose symbolic computation arises from its dedication to a specific class of functions. This feature is particularly valuable in optimization studies where computational efficiency is important.

In the laminate example which is considered, the relationship between the loading parameters and the material stress is combined and simplified into one transformation matrix using symbolic computation. This involves tedious matrix algebra, where the matrix entries are series of double trigonometric functions of the fibre angle. The stress is determined symbolically and substituted into a quadratic failure criterion from which the critical thickness is obtained as an analytical function of the fibre angle via symbolic computation. The first and second derivatives of the critical thickness with respect to fibre angle are determined exactly with the aid of symbolic differentiation and may be used to determine the optimal fibre angle by means of an optimization algorithm.

A solution method is presented for the optimal design of symmetrically laminated cylindrical pressure vessels with balanced and unbalanced stacking sequences on the basis of a strength failure criterion.

Two design problems are solved. In the first problem, a cylindrical pressure vessel is optimized taking the fibre angle as the design variable to maximize the burst pressure and the effects of the axial force and torque on the optimal designs are investigated. In the second problem, a cylindrical vessel filled with a liquid and subject to an internal pressure is studied. The weight of the shell is minimized taking the fibre angle and wall thickness as the design variables. Both constant and variable thickness shells are investigated. It is shown that the results for the second problem approach those of the first problem as the internal pressure increases.

Numerical results are given for unbalanced (single– and six–layered) and balanced laminates noting that in the balanced case the number of layers does not affect the results. It is observed that fibre angles align themselves with the longitudinal axis as the axial force increases. Variable thickness shells are found to be about 20% more efficient than the constant thickness shells for low values of the internal pressure with the difference decreasing as this pressure increases. For single layer pressure vessels, the optimal fibre angle is found to be either 0° or 90° with the switch-over point depending on the magnitude of the axial force, torque or the internal pressure.

Chapter 3

Derivation of a Higher–Order Theory for Thick Laminated Plates and Shells

3.1 Introduction

The objective of the present chapter is to derive a nonclassical theory for the accurate analysis of thick laminated composite structures. The classical theory is based on the Kirchhoff–Love hypotheses of straight inextensional normals and therefore neglects the phenomena of transverse shear and normal deformation. The exact prediction of the stress and strain state of thick laminated structures made of advanced composite materials requires the use of three–dimensional elasticity models. However, quasi–three–dimensional models based on higher–order theories may accurately describe the behaviour of thick structures over some range of applicability and are considerably less computationally expensive than three–dimensional models.

In this chapter, comprehensive higher–order theory of laminated plates and shells is presented. This theory considers plates and shells with transversely isotropic layers of different thicknesses and stiffnesses, and takes into account both transverse shear and normal deformation.

The theory is based on kinematic hypotheses which are not taken *a priori,* but whose form are derived using an iterative technique where the classical Kirchhoff–Love hypotheses are assumed in the first iteration. New variables which have a clear physical meaning are introduced.

The equations of equilibrium and the boundary conditions are determined using Lagrange's variational principle, and the complete set of boundary conditions is derived. Various loading and boundary conditions which fully take into account transverse shear and normal deformation are considered.

The unknown functions in the system of governing differential equations are defined on an arbitrary reference surface. The stress/strain state of the laminated structure is determined from the solution at the reference surface and through–the–thickness distribution functions defined by the theory. The order of the governing equations is 16 and is independent of the number of layers.

3.2 Literature Survey

The accurate analysis of laminated composite plates and shells is the subject of much investigation and new higher–order theories have been developed which attempt to accurately describe the three–dimensional elastic behaviour of laminated plates and shells. Surveys of these theories may be found in the reviews by Dudchenko *et al* [25], Librescu & Reddy [26], Reddy [27], Noor & Burton [28], and in the books of Bolotin & Novichkov [29], Grigorenko & Vasilenko [30] and Piskunov & Verijenko [31].

The two main approaches for deriving two–dimensional equations of plates and shells are the analytical method introduced by Reissner [32], Mindlin [33] and Gol'denveizer [34], and the method of hypotheses.

In the second method, kinematic or static assumptions regarding the variation of displacements, strains and/or stresses through the thickness are introduced. Two approaches for the derivation of theories using the method of hypotheses have been employed, leading to *single–layer* and *discrete–layer* theories. In discrete–layer theories, the hypotheses relate to each layer such that the order of the governing differential equations is *dependent* of the number of layers, whereas in single–layer theories, the hypotheses relate to displacements, strain and/or stress through the thickness such that the order of the governing equations is *independent* of the number of layers.

The generalization of the discrete–layer approach is given by Bolotin & Novichkov [29], Grigorenko & Vasilenko [30] and more recently by Reddy [35]. In principle, discrete–layer theories can model the interlaminar stress more accurately. However, these theories are computationally expensive as compared to single–layer theories.

The single–layer approach was introduced by Ambartsumyan [36] where the classical hypotheses of Kirchoff–Love were used. He then used this approach to derive a higher–order theory of laminated plates and shells [37]. Examples of other single–layer theories are those of Reissner and Mindlin, first–order theories by Vasilenko & Savchenko [38] and Grigorenko *et al.* [39] with both transverse shear and normal deformation included, and higher–order theories by Stein [40] and Reddy [41]. However in these and other more recent publications mentioned in the reviews [25, 28], there is no compatibility between the nonlinear kinematic model which considers the distortion of the normal, and the system of internal forces and moments which are equivalent to those obtained using the *straight line* hypothesis. Theories without these disadvantages have been derived by Piskunov [42] and Rasskazov [43]. The approach introduced in Ref. [42] has been extended by Piskunov *et al.* [31, 44] and Verijenko *et al.* [45, 46, 47] to better represent the interlaminar stresses, include the direct effect of loading on transverse shear and normal deformation, and increase the range of applicability of the theory. The higher–order theory derived by Piskunov *et al.* in Ref. [44] is presented in this chapter.

A study of the reviews mentioned earlier and some other recent papers [48, 49, 50] reveals that in the design of multilayered structures in which the layers have significantly different physical characteristics, it is also necessary to consider the phenomenon of normal deformation.

3.3 Basic Equations

The shell is referenced by a curvilinear coordinate system $x_1 0\, x_2$ which is parallel to the bounding surfaces and the surfaces of contact between the layers. The axes of the curvilinear orthogonal coordinate $x_i = \text{const}$ $(i = 1, 2)$ coincide with the principle lines of curvature. The coordinate $z = x_3$ is normal to the reference surface $x_1 0\, x_2$. The reference surface $z = 0$ may be positioned arbitrarily in the package of layers. The layers are assumed to be perfectly bonded, and shells are taken as a geometry with small curvature relative to their thickness (see Figure 3.1 on Page 59). The curvatures of the shell are given by k_{ij}.

In following derivation, the index $k = 1, 2 \ldots n$ refers to the k-th layer of a laminate with n layers. The indices $i = 1, 2$ and $j = 1, 2$ refer to the coordinate directions x_1, x_2, and the index s has the range $s = 1, 2, 3$. A subscripted comma denotes differentiation with respect to the variables following the comma, and a superscripted

(k) refers to the k-th layer.

The loads applied on the external surfaces are $p^+ = p_s^+(x)$ and $p^- = p_s^-(x)$ $(s = 1,2,3)$ respectively and are functions of the curvilinear orthogonal coordinates $x = \{x_1, x_2\}$. The conditions on the external surfaces are

$$\begin{aligned} \sigma_{s3}^{(1)} &= -p_s^- \quad (z = a_0, k = 1) \\ \sigma_{s3}^{(n)} &= p_s^+ \quad (z = a_n, k = n) \end{aligned} \tag{3.1}$$

Since the layers are assumed to be rigidly bonded, the rigidity condition for an arbitrary surface $z = a_{k-1}$ is given by

$$\sigma_{s3}^{(k)} = \sigma_{s3}^{(k-1)} \quad \text{(static)} \tag{3.2}$$

$$u_s^{(k)} = u_s^{(k-1)} \quad \text{(kinematic)} \tag{3.3}$$

The components of the deformation of k-th layer $(k = 1, 2 \ldots n)$ for *small* bending are given in Ref. [51] as

$$2e_{ij}^{(k)} = u_{i,j}^{(k)} + u_{j,i}^{(k)} + 2k_{ij}u_3^{(k)} \tag{3.4}$$

$$2e_{i3}^{(k)} = u_{i,3}^{(k)} + u_{3,i}^{(k)} \tag{3.5}$$

$$e_{33}^{(k)} = u_{3,3}^{(k)} \tag{3.6}$$

The displacements of the reference surface $(z = 0, k = m)$ are expressed as

$$\begin{aligned} u_i^{(m)}(x,0) &= u_i(x) \quad (i = 1,2) \\ u_3^{(m)}(x,0) &= w(x) \end{aligned} \tag{3.7}$$

and the deformations of the reference surface as

$$\begin{aligned} \varepsilon_{ij} &= \frac{1}{2}(u_{i,j} + u_{j,i}) + k_{ij}w \\ \kappa_{ij} &= -w_{,ij} \end{aligned} \tag{3.8}$$

which satisfy the well-known equations [51]

$$\begin{aligned} 2\varepsilon_{12,12} - \varepsilon_{11,22} - \varepsilon_{22,11} &= k_{11}\kappa_{22} + k_{22}\kappa_{11} - 2k_{12}\kappa_{12} \\ \kappa_{11,2} - \kappa_{12,1} &= 0 \\ \kappa_{22,1} - \kappa_{12,2} &= 0 \end{aligned} \tag{3.9}$$

The generalized Hooke's law for the transversely isotropic k-th layer of the shell [37], where the surface of isotropy at any point (x, z) is orthogonal to the normal, is given

by

$$
\begin{aligned}
e_{11}^{(k)} &= [\sigma_{11}^{(k)} - \nu_k \sigma_{22}^{(k)}]/E_k - \nu_k' \sigma_{33}^{(k)}/E_k' \\
e_{22}^{(k)} &= [\sigma_{22}^{(k)} - \nu_k \sigma_{11}^{(k)}]/E_k - \nu_k' \sigma_{33}^{(k)}/E_k' \\
e_{33}^{(k)} &= -[\sigma_{11}^{(k)} + \sigma_{22}^{(k)}]\nu_k'/E_k' + \sigma_{33}^{(k)}/E_k' \\
2e_{12}^{(k)} &= \sigma_{12}^{(k)}/G_k \\
2e_{13}^{(k)} &= \sigma_{13}^{(k)}/G_k' \\
2e_{23}^{(k)} &= \sigma_{23}^{(k)}/G_k'
\end{aligned}
\tag{3.10}
$$

where the elastic properties are assumed to be functions of the coordinate z $(a_{k-1} \leq z \leq a_k)$. $E_k(z)$, $\nu_k(z)$ and $G_k(z) = E_k/[2(1+\nu_k)]$ are the modulus of elasticity, Poisson's ratio and shear modulus respectively in the plane of isotropy; $E_k'(z)$ and $G_k'(z)$ are the moduli of elasticity and shear respectively in the transverse direction; and $\nu_k'(z)$ is Poisson's ratio, which characterizes the reduction in the plane of isotropy when tension is applied in the transverse direction.

Classical assumptions are employed to derive basic equations for the derivation of the nonclassical higher–order theory. If the Kirchoff–Love hypotheses are assumed to be valid for each layer of the shell, then

$$2e_{i3}^{(k)} = 0\,; \quad e_{33}^{(k)} = 0 \tag{3.11}$$

$$\sigma_{33}^{(k)} = 0 \tag{3.12}$$

By substituting eqns. (3.10) into the hypotheses (3.11), integrating within the conditions (3.2) and using the notation (3.7), we obtain the classical kinematic model

$$u_i^{(k)} = u_i - w_{,i} z\,; \quad u_3^{(k)} = w \tag{3.13}$$

The tangential deformations of the k-th layer are obtained from eqns. (3.13) and (3.8) as

$$e_{ij}^{(k)} = \varepsilon_{ij} + \kappa_{ij} z \tag{3.14}$$

where it is noted that $\varepsilon_{12} = \varepsilon_{21}$, $\kappa_{12} = \kappa_{21}$ and $e_{12}^{(k)} = e_{21}^{(k)}$.

Substituting eqn. (3.14) into eqn. (3.10) and using the static hypothesis (3.12) or the assumption $E_k' = \infty$, the in–plane stresses in the k-th layer are derived as

$$
\begin{aligned}
\sigma_{11}^{(k)} &= E_{0k}[(\varepsilon_{11} + \nu_k \varepsilon_{22}) + (\kappa_{11} + \nu_k \kappa_{22}) z] \\
\sigma_{22}^{(k)} &= E_{0k}[(\varepsilon_{22} + \nu_k \varepsilon_{11}) + (\kappa_{22} + \nu_k \kappa_{11}) z] \\
\sigma_{12}^{(k)} &= E_{0k}(1 - \nu_k)(\varepsilon_{12} + \kappa_{12} z)
\end{aligned}
\tag{3.15}
$$

where $E_{0k} = E_k/(1-\nu_k^2)$.

The transverse stresses cannot be found using Hooke's law because of the hypotheses (3.11) and (3.12). Therefore, the transverse stresses are derived using the equations of equilibrium for a shell [51] which for the k-th layer may be expressed as

$$\sigma^{(k)}_{ij,j} + \sigma^{(k)}_{i3,3} = 0 \tag{3.16}$$

$$\sigma^{(k)}_{33,3} + \sigma^{(k)}_{i3,i} - k_{ij}\sigma^{(k)}_{ij} = 0 \tag{3.17}$$

Using eqn. (3.16), the transverse shear stresses are derived as

$$\sigma^{(k)}_{i3} = -\int_{a_{k-1}}^{z} \sigma^{(k)}_{ij,j} dz + A_{ik} \tag{3.18}$$

and using eqn. (3.17), the transverse normal stress are derived as

$$\sigma^{(k)}_{33} = -\int_{a_{k-1}}^{z} [\sigma^{(k)}_{i3,i} - k_{ij}\sigma^{(k)}_{ij}] dz + A_{3k} \tag{3.19}$$

where $A_{sk}(x)$ are functions of integration which may be derived from the loading conditions (3.1) and static rigidity condition (3.12).

Substituting the equations for stress (3.15) into the integrals (3.18) and calculating the functions of integration $A_{ik}(x)$ from the loading and rigidity conditions, the transverse shear stresses are derived in Ref. [31] as

$$\sigma^{(k)}_{i3} = \Delta w_{,i} f_{1k} + p_i^- f_{2k} + p_i^+ f_{3k} \tag{3.20}$$

where Δ is the Laplace operator, p_i^-, p_i^+ are the external loads and $f_{sk}(z)$ are distribution functions given by

$$\begin{aligned} f_{1k}(z) &= f_k^* - f_k B_{f1}/B_f \\ f_{2k}(z) &= f_k/B_f - 1 \\ f_{3k}(z) &= f_k/B_f \end{aligned} \tag{3.21}$$

where

$$\begin{aligned} f_k(z) &= \int_{a_0}^{z} E_{0k} dz \\ f_k^*(z) &= \int_{a_0}^{z} E_{0k} z dz \\ B_f &= \int_{a_0}^{a_n} E_{0k} dz \\ B_{f1} &= \int_{a_0}^{a_n} E_{0k} z dz \end{aligned} \tag{3.22}$$

The distribution functions $f_{sk}(z)$ enable the loading conditions on the external surfaces to be satisfied once the reference surface has been positioned in the package of layers and also take into account the influence of the elastic properties of each layer on the stress distribution of the components $\sigma_{i3}^{(k)}$ under external loading.

Substituting the derived transverse shear stress (3.20) into the integral (3.19) and calculating the functions of integration $A_{3k}(x)$ from the loading conditions (3.1) and static rigidity condition (3.12), the transverse normal stress is derived in Ref. [31] as

$$\sigma_{33}^{(k)} = p_{i,i}^{-} f_{4k} + p_{i,i}^{+} f_{5k} + p_3^{-} f_{6k} + p_3^{+} f_{7k} + B_\sigma^{(k)} \tag{3.23}$$

where p_3^{-}, p_3^{+} are the normal components of the external loads and the functions $B_\sigma^{(k)}$ are given by

$$\begin{aligned} B_\sigma^{(k)}(x,z) &= k_{ij}\varepsilon_{ij} f_{8k} + k_{ij}\kappa_{ij} f_{9k} + (k_{11}\varepsilon_{22} - 2k_{12}\varepsilon_{12} + k_{22}\varepsilon_{11}) f_{8k} \\ &\quad + (k_{11}\kappa_{22} - 2k_{12}\kappa_{12} + k_{22}\kappa_{11}) \bar{f}_{9k} \end{aligned} \tag{3.24}$$

The distribution functions in eqns. (3.23) and (3.24) are given by

$$\begin{aligned} f_{4k}(z) &= F_{1k} D_{f2}/D_{f1} - F_{2k} \\ f_{5k}(z) &= F_{1k} D_{f3}/D_{f1} - F_{3k} \\ f_{6k}(z) &= F_{1k}/D_{f1} - 1 \\ f_{7k}(z) &= F_{1k}/D_{f1} \\ f_{8k}(z) &= f_k - F_{1k} B_f/D_{f1} \\ f_{9k}(z) &= f_k^* - F_{1k} B_{f1}/D_{f1} \\ \bar{f}_{8k}(z) &= \bar{f}_k - F_{1k} \bar{B}_f/D_{f1} \\ \bar{f}_{9k}(z) &= \bar{f}_k^* - F_{1k} B_{f1}^*/D_{f1} \end{aligned} \tag{3.25}$$

where

$$\begin{aligned} F_{sk}(z) &= \int_{a_0}^{z} f_{sk} dz & D_{fs} &= \int_{a_0}^{a_n} f_{sk} dz \\ \bar{f}_k(z) &= \int_{a_0}^{z} E_{0k}\nu_k dz & \bar{f}_k^*(z) &= \int_{a_0}^{z} E_{0k}\nu_k z dz \\ \bar{B}_f &= \int_{a_0}^{a_n} E_{0k}\nu_k dz & B_{f1}^* &= \int_{a_0}^{a_n} E_{0k}\nu_k z dz \end{aligned} \tag{3.26}$$

Substituting eqns. (3.15) for the normal stresses $\sigma_{ij}^{(k)}$ in terms of the deformations of the reference surface and eqn. (3.23) for the transverse normal stress $\sigma_{33}^{(k)}$ into Hooke's law (3.10), the normal strain is obtained as

$$e_{33}^{(k)} = \nu_{0k}' z \Delta w - \nu_{0k}' u_{i,i} + \frac{1}{E_k'}(p_{i,i}^{-} f_{4k} + p_{i,i}^{+} f_{5k} + p_3^{-} f_{6k} + p_3^{+} f_{7k} + B_\sigma) \tag{3.27}$$

where $\nu'_{0k} = E_k\nu'_k/E'_k(1-\nu_k)$.

The normal strain (3.27) may be expressed as

$$e_{33}^{(k)} = \Delta w\alpha_{1k} + u_{i,i}\alpha_{2k} + p^-_{i,i}\alpha_{3k} + p^+_{i,i}\alpha_{4k} + p^-_3\alpha_{5k} + p^+_3\alpha_{6k} + B'_\sigma \tag{3.28}$$

where $B'_\sigma = B_\sigma/E'_k$ and we denote the distribution functions of the normal strain as

$$\begin{aligned} \alpha_{1k}(z) &= \nu'_{0k}z \\ \alpha_{2k}(z) &= -\nu'_{0k} \\ \alpha_{qk}(z) &= f_{gk}/E'_k \quad (q = 3\ldots 6,\ g = q+1) \end{aligned} \tag{3.29}$$

Since $e_{33}^{(k)} = u_{3,3}^{(k)}$, the equation for the normal displacements may be expressed as

$$u_3^{(k)}(x,z) = w(x) + \int_{a_{k-1}}^{z} e_{33}^{(k)}dz + C_{3k}(x) \tag{3.30}$$

The function of integration $w(x) + C_{3k}(x)$ is determined from the conditions (3.11) and (3.7), ie. $u_3^{(k)}(x,a_{k-1}) = u_3^{(k-1)}(x,a_k)$ and $u_3^{(m)}(x,0) = w(x)$. Substituting eqn. (3.28) into (3.30) and integrating, we obtain

$$u_3^{(k)}(x,z) = w + \Delta w\varphi_{1k} + u_{i,i}\varphi_{2k} + p^-_{i,i}\varphi_{3k} + p^+_{i,i}\varphi_{4k} + p^-_3\varphi_{5k} + p^+_3\varphi_{6k} + C_\sigma \tag{3.31}$$

where $C_\sigma = \int_0^z B'_\sigma dz$ and we denote the distribution functions of the normal displacement through the thickness of the laminated shell as

$$\varphi_{qk}(z) = \int_0^z \alpha_{qk}dz \qquad (q = 1\ldots 6) \tag{3.32}$$

Substituting the equation (3.20) for the the transverse shear stress $\sigma_{i3}^{(k)}$ into the expression $2e_{i3}^{(k)} = \sigma_{i3}^{(k)}/G'_k$ from eqn. (3.10), and substituting the equation in (3.31) for the normal displacement $u_3^{(k)}$ into the expression $2e_{i3} = u_{i,3}^{(k)} + u_{3,i}^{(k)}$ given in eqn. (3.5), we obtain

$$\begin{aligned} u_{i,3}^{(k)} &= 2e_{i3}^{(k)} - u_{3,i}^{(k)} = \sigma_{i3}^{(k)}/G'_k - u_{3,i}^{(k)} \\ &= -w_{,i} - \Delta w_{,i}(\varphi_{1k} - \varphi_k) - u_{i,ij}\varphi_{2k} - p^-_{i,ij}\varphi_{3k} - p^+_{i,ij}\varphi_{4k} \\ &\quad -p^-_{3,i}\varphi_{5k} - p^-_{3,i}\varphi_{6k} - p^-_i\varphi_{7k} - p^+_i\varphi_{8k} - C_{\sigma,i} \end{aligned} \tag{3.33}$$

where

$$\begin{aligned} \varphi_k(z) &= f_{1k}/G'_k \\ \varphi_{7k}(z) &= -f_{2k}/G'_k \\ \varphi_{8k}(z) &= -f_{3k}/G'_k \end{aligned} \tag{3.34}$$

Integrating eqn. (3.33) and satisfying the conditions (3.7) and (3.11), we obtain the tangential components of the displacement due to transverse shear as

$$\begin{aligned} u_i^{(k)}(x,z) &= u_i - w_{,i}z - \Delta w_{,i}\psi_{1k} - u_{j,ji}\psi_{2k} - p^-_{j,ji}\psi_{3k} - p^+_{j,ji}\psi_{4k} \\ &\quad -p^-_{3,i}\psi_{5k} - p^+_{3,i}\psi_{6k} - p^-_i\psi_{7k} - p^+_i\psi_{8k} - A_\sigma \end{aligned} \tag{3.35}$$

where $A_\sigma = \int_0^z C_{\sigma,i}dz$ and we denote

$$\begin{aligned} \psi_{1k}(z) &= \int_0^z (\varphi_{1k} - \varphi_k)dz \\ \psi_{gk}(z) &= \int_0^z \varphi_{gk}dz \qquad (g = 2\ldots 8) \end{aligned} \tag{3.36}$$

The equations derived for $e^{(k)}_{i3}, e^{(k)}_{33}$ and $\sigma^{(k)}_{i3}, \sigma^{(k)}_{33}$ are incompatible with the classical assumptions and are therefore not relevant to the classical theory. These equations, however, are important for the derivation of the higher–order theory.

3.4 Higher–Order Theory

The form the kinematic hypotheses of the higher–order theory is taken from eqn. (3.35) for the displacements $u^{(k)}_i$ and eqn. (3.31) for the deflection $u^{(k)}_3$. In particular, the higher–order theory is based on the kinematic hypotheses

$$u_i^{(k)}(x,z) = u_i - w_{,i}z - \chi_{g,i}\psi_{gk} \quad (g = 1\ldots 8) \tag{3.37}$$

$$u_3^{(k)}(x,z) = w + \chi_q\varphi_{qk} \quad (q = 1\ldots 6) \tag{3.38}$$

where $u_i(x)$, $w(x)$ are the displacements of the reference surface; $\chi_1(x)$, $\chi_2(x)$ are new functions described below; and we denote

$$\begin{aligned} &\chi_3(x) = p^-_{i,i}; \quad \chi_4(x) = p^+_{i,i}; \quad \chi_5(x) = p^-_3; \\ &\chi_6(x) = p^+_3; \quad \chi_{7,i}(x) = p^-_i; \quad \chi_{8,i}(x) = p^+_i; \end{aligned} \tag{3.39}$$

which are determined from the given loading.

The distribution functions φ_{qk} of the deflection through the thickness of the laminate are given by eqn. (3.32), and the distribution functions ψ_{gk} of the tangential components of the displacement vector through the thickness of the laminate are given by eqns. (3.36).

The first of the two new unknown functions $\chi_1(x)$ and $\chi_2(x)$ is termed the *shear function* and the second is termed the *compression function* [56, 57]. If $E'_k \neq \infty$ but

$\nu'_k = 0$ then $\varphi_{1k} = \varphi_{2k} = \psi_{2k} = 0$ and thus $\chi_1(x)$ and $\chi_2(x)$ are also connected with Poisson's transverse reduction. The interpretation of the functions $\chi_q(x)$ $(q = 1 \ldots 6)$ in eqns. (3.37) and (3.38) is now discussed. Together with the functions $\varphi_{qk}(a)$ $(q = 1 \ldots 6)$ they represent the additional normal deflection of the surface $z = a$ relative to the deflection of the reference surface. The derivatives $\chi_{g,i}$ together with the functions $\psi_{gk}(a)$ $(g = 1 \ldots 8)$ represent the angles of the tangents to the distorted normal at $z = a$ (in the directions x_i) which are added to the angles of the straight normals to the displaced reference surface, as illustrated in Figure 3.2. The other terms involving χ_g $(g = 3 \ldots 8)$, which are determined from the external loads, take into account the normal deformation caused by the direct loading.

Substituting the kinematic model given in eqns. (3.37) and (3.38) into the strain–displacement relations given by eqns. (3.4)—(3.6) and using the notation (3.8), the components of the strain tensor are obtained as

$$\begin{aligned} e_{ij}^{(k)} &= \varepsilon_{ij} + \kappa_{ij} z + \kappa_{ij}^{(g)} \psi_{gk} + k_{ij} \chi_q \varphi_{qk} \\ 2e_{i3}^{(k)} &= \chi_{t,i} \beta_{tk} \quad (t = 1, 7, 8) \\ e_{33}^{(k)} &= \chi_q \alpha_{qk} \end{aligned} \tag{3.40}$$

where $g = 1 \ldots 8$, $q = 1 \ldots 6$ and ε_{ij} and κ_{ij} are the strains of the reference surface given by eqn. (3.8); and we denote $\kappa_{ij}^{(g)} = -\chi_{g,ij}$ and

$$\begin{aligned} \beta_{1k}(z) &= f_{1k}/G'_k \\ \beta_{7k}(z) &= f_{2k}/G'_k \\ \beta_{8k}(z) &= f_{3k}/G'_k \end{aligned} \tag{3.41}$$

It is noted that $\kappa_{11,2}^{(g)} = \kappa_{12,1}^{(g)}$ and $\kappa_{22,1}^{(g)} = \kappa_{12,2}^{(g)}$.

Using Hooke's law for a transversely isotropic material (3.10), the components of the stress tensor may be determined as

$$\begin{aligned} \sigma_{11}^{(k)} &= A_{11k} e_{11}^{(k)} + A_{12k} e_{22}^{(k)} + A_{13k} e_{33}^{(k)} \\ \sigma_{22}^{(k)} &= A_{12k} e_{11}^{(k)} + A_{11k} e_{22}^{(k)} + A_{13k} e_{33}^{(k)} \\ \sigma_{33}^{(k)} &= A_{13k} [e_{11}^{(k)} + e_{22}^{(k)}] + A_{33k} e_{33}^{(k)} \\ \sigma_{12}^{(k)} &= 2G'_k e_{12}^{(k)} \\ \sigma_{13}^{(k)} &= 2G'_k e_{13}^{(k)} \\ \sigma_{23}^{(k)} &= 2G'_k e_{23}^{(k)} \end{aligned} \tag{3.42}$$

where

$$A_{11k} = \Delta_{11k}/\Delta_k$$

$$
\begin{aligned}
A_{12k} &= \Delta_{12k}/\Delta_k \\
A_{13k} &= \Delta_{13k}/\Delta_k \\
A_{33k} &= \Delta_{33k}/\Delta_k
\end{aligned} \tag{3.43}
$$

and

$$
\begin{aligned}
\Delta_k &= (1+\nu_k)[1-\nu_k-2(\nu_k')^2 E_k/E_k']/E_k^2 E_k' \\
\Delta_{11k} &= [1-(\nu_k')^2 E_k/E_k']/E_k E_k' \\
\Delta_{12k} &= [\nu_k+(\nu_k')^2 E_k/E_k']/E_k E_k' \\
\Delta_{13k} &= \nu_k'(1+\nu_k)/E_k E_k' \\
\Delta_{33k} &= (1-\nu_k^2)/E_k^2
\end{aligned} \tag{3.44}
$$

are the elastic constants for the transversely isotropic k-th layer.

Substituting the components of the strain tensor (3.40) into the stress tensor (3.42) gives

$$
\begin{aligned}
\sigma_{11}^{(k)} &= A_{11k}(\varepsilon_{11}+\kappa_{11}z+\kappa_{11}^{(g)}\psi_{gk}+k_{11}\chi_q\varphi_{qk}) \\
&\quad +A_{12k}(\varepsilon_{22}+\kappa_{22}z+\kappa_{22}^{(g)}\psi_{gk}+k_{22}\chi_q\varphi_{qk})+A_{13k}\chi_q\alpha_{qk} \\
\sigma_{22}^{(k)} &= A_{12k}(\varepsilon_{11}+\kappa_{11}z+\kappa_{11}^{(g)}\psi_{gk}+k_{11}\chi_q\varphi_{qk}) \\
&\quad +A_{11k}(\varepsilon_{22}+\kappa_{22}z+\kappa_{22}^{(g)}\psi_{gk}+k_{22}\chi_q\varphi_{qk})+A_{13k}\chi_q\alpha_{qk} \\
\sigma_{33}^{(k)} &= A_{13k}[\varepsilon_{11}+\varepsilon_{22}+(\kappa_{11}+\kappa_{22})z+(\kappa_{11}^{(g)}+\kappa_{22}^{(g)})\psi_{gk} \\
&\quad +(k_{11}+k_{22})\chi_q\varphi_{qk}]+A_{33k}\chi_q\alpha_{qk} \\
\sigma_{12}^{(k)} &= 2G_k(\varepsilon_{12}+\kappa_{12}z+\kappa_{12}^{(g)}\psi_{gk}+k_{12}\chi_q\varphi_{qk}) \\
\sigma_{i3}^{(k)} &= \chi_{t,i}f_{tk}
\end{aligned} \tag{3.45}
$$

where $g=1\ldots8$, $q=1\ldots6$ and $t=1,2,3$.

The above equations define the components of the displacement vector and the stress and strain tensors at an arbitrary point in the k-th layer. The model equations include: the system of independent unknown functions of the reference surface u_i, w, χ_p $(i,p=1,2)$; the known functions χ_g $(g=3\ldots8)$ which are determined from the given loads p^-, p^+ on the external surfaces; and the system of known functions of the normal coordinate z, incorporating the laws governing the variation of the components of the displacement vector and of the stress and strain tensors through the thickness of the package of layers. Clearly, the describing equations are not dependent on the thicknesses, stiffnesses and other properties of the layers. Moreover, the equations of this model may consider layers with elastic characteristics that are constant or variable. Thus the model is comprehensive with respect to the configuration of the package of layers.

3.5 Equilibrium & Boundary Conditions

The equations of equilibrium and the boundary conditions are determined using the Lagrange variational principle as follows:

$$\delta U = \delta\Pi + \iiint_V \{e_{33}^{(k)} + \frac{\nu_k'}{E_k'}[\sigma_{11}^{(k)} + \sigma_{22}^{(k)}] - \frac{1}{E_k'}\sigma_{33}^{(k)}\}\ \delta\sigma_{33}^{(k)}\ dV - \delta H = 0 \qquad (3.46)$$

where $\delta\Pi$ is the variation of the potential energy of deformation, and δH is the variation of the work of the external forces. Using the components of the strain tensor in eqns. (3.40), we derive

$$\begin{aligned}\delta\Pi \;=\; & \iint_S \{\int_{a_0}^{a_n} [\sigma_{ij}^{(k)}\delta(\varepsilon_{ij} + \kappa_{ij}z + \kappa_{ij}^{(g)}\psi_{gk} + k_{ij}\chi_q\varphi_q) \\ & + \sigma_{i3}^{(k)}\delta(\chi_{t,i}\beta_{tk}) + \sigma_{33}^{(k)}\delta(\chi_q\alpha_{qk})]dz\}dS \end{aligned} \qquad (3.47)$$

where $i,j = 1,2$, $t = 1,7,8$, $q = 1\ldots6$ and $g = 1\ldots8$.

Using notation similar to that of classical theory, the forces and moments are expressed as integrals of the stresses, viz.

$$N_{ij} = \int_{a_0}^{a_n} \sigma_{ij}^{(k)}\ dz\ ; \quad M_{ij} = \int_{a_0}^{a_n} \sigma_{ij}^{(k)}\ z\ dz\ ; \quad Q_i = \int_{a_0}^{a_n} \sigma_{i3}^{(k)} dz \qquad (3.48)$$

Higher–order forces and moments which describe the influence of transverse shear and normal deformation are denoted

$$\begin{aligned} N_{ij}^{(p)} &= \int_{a_0}^{a_n} \sigma_{ij}^{(k)}\varphi_{pk}dz\ ; \quad M_{ij}^{(p)} = \int_{a_0}^{a_n} \sigma_{ij}^{(k)}\psi_{pk}dz \\ Q_i^{(1)} &= \int_{a_0}^{a_n} \sigma_{i3}^{(k)}\beta_{1k}dz\ ; \quad Q_3^{(p)} = \int_{a_0}^{a_n} \sigma_{33}^{(k)}\alpha_{pk}dz \end{aligned} \qquad (3.49)$$

where $p = 1,2$.

Eqn. (3.46) may be expressed using these forces and moments as

$$\begin{aligned}\delta\Pi \;=\; & \iint_S [N_{ij}\delta u_{i,j} + N_{ij}k_{ij}\delta w - M_{ij}\delta w_{,ij} - M_{ij}^{(p)}\delta\chi_{p,ij} \\ & + N_{ij}^{(p)}k_{ij}\delta\chi_p + Q_i^{(1)}\delta\chi_{1,i} + Q_3^{(p)}\delta\chi_p]\ dS \end{aligned} \qquad (3.50)$$

where $p = 1,2$.

The variation of the work of the external loads consists of the variation of the load H_1 which is applied to the bounding surfaces and that of the boundary forces H_2. Therefore

$$\delta H_1 \;=\; \iint_S [p_s^-\delta u_s^{(1)} + p_s^+\delta u_s^{(n)}]dS \qquad (3.51)$$

$$\delta H_2 \;=\; \int_L \{\int_{a_0}^{a_n} [\sigma_{hh}^{(k)}\delta u_h^{(k)} + \sigma_{h3}^{(k)}\delta u_3^{(k)} + \sigma_{hl}^{(k)}\delta u_l^{(k)}]dz\}dL \qquad (3.52)$$

where $s = 1, 2, 3$; h and l are the normal and tangent to the boundary L of the shell; $\sigma_{hh}^{(k)}$, $\sigma_{h3}^{(k)}$ and $\sigma_{hl}^{(k)}$ are components of the stress tensor at an arbitrary point in the k-th layer on the edge L of the shell.

Using the hypotheses (3.37) and (3.38), the variation of the work of the given loading is expressed as

$$\begin{aligned} \delta H_1 &= \iint_S (p_i \delta u_i + p_3 \delta w + p_3^{(p)} \delta \chi_p)\, dS \\ &\quad - \int_L (p_h \delta w + p_h^{(p)} \delta \chi_p)\, dL \end{aligned} \tag{3.53}$$

where the summation index $p = 1, 2$, we denote $h = i$ and the generalized loads are given by

$$\begin{aligned} p_i &= p_i^- + p_i^+ \\ p_3 &= a_0 p_{i,i}^- + a_n p_{i,i}^+ + p_3^- + p_3^+ \\ p_3^{(p)} &= \psi_{p1}(a_0) p_{i,i}^- + \psi_{pn}(a_n) p_{i,i}^+ + \varphi_{p1}(a_0) p_3^- + \varphi_{pn}(a_n) p_3^- \\ p_h &= a_0 p_h^- + a_n p_h^+ \\ p_h^{(p)} &= \psi_{p1}(a_0) p_h^- + \psi_{pn}(a_n) p_h^+ \end{aligned} \tag{3.54}$$

Equating the variation of given functions to zero, we may express eqn. (3.52) as

$$\begin{aligned} \delta H_2 &= \int_L \{N_{hh}^* \delta u_h + N_{hl}^* \delta u_l - M_{hh}^* \delta w_{,h} - M_{hh}^{*(p)} \delta \chi_{p,h} \\ &\quad + (M_{hl,l}^* + Q_h^*) \delta w + [M_{hl,l}^{*(p)} + Q_h^{*(p)}] \delta \chi_p \} dL \\ &\quad - [M_{hl,l}^* \delta w + M_{hl}^{*(p)} \delta \chi_p]_{L_1}^{L_2} \end{aligned} \tag{3.55}$$

where an asterisk denotes forces acting on the boundary of the shell, and we denote

$$Q_h^{*(p)} = \int_{a_0}^{a_n} \sigma_{h3}^{(k)} \varphi_{pk} dz \tag{3.56}$$

The equations of equilibrium may now be obtained as

$$\begin{aligned} &N_{ij,j} + p_i = 0 \\ &M_{ij,ij} - k_{ij} N_{ij} + p_3 = 0 \\ &M_{ij,ij}^{(1)} + Q_{i,i}^{(1)} - Q_3^{(1)} - k_{ij} N_{ij}^{(1)} + p_3^{(1)} = 0 \\ &M_{ij,ij}^{(2)} - Q_3^{(2)} - k_{ij} N_{ij}^{(2)} + p_3^{(2)} = 0 \end{aligned} \tag{3.57}$$

and the boundary conditions are given by

$$(N_{hh} - N_{hh}^*)\, \delta u_h = 0$$

$$
\begin{aligned}
&(N_{hl} - N^*_{hl})\ \delta u_l = 0 \\
&(M_{hh,h} + 2M_{hl,l} + p_h - R^*_h)\ \delta w = 0 \\
&(M_{hh} - M^*_{hh})\delta w_{,h} = 0 \\
&(M^{(1)}_{hh} - M^{*(1)}_{hh})\ \delta \chi_{1,h} = 0 \\
&(M^{(2)}_{hh} - M^{*(2)}_{hh})\ \delta \chi_{2,h} = 0 \\
&(M^{(1)}_{hh,h} + 2M^{(1)}_{hl,l} + Q^{(1)}_h + p^{(1)}_h - R^{*(1)}_h)\ \delta \chi_1 = 0 \\
&(M^{(2)}_{hh,h} + 2M^{(2)}_{hl,l} + p^{(2)}_h - R^{*(2)}_h)\ \delta \chi_2 = 0
\end{aligned}
\tag{3.58}
$$

where we denote the generalized reactions

$$
R^*_h = Q^*_h + M^*_{hl,l}\ ; \quad R^{*(p)}_h = Q^{*(p)}_h + M^{*(p)}_{hl,l} \tag{3.59}
$$

Modelling different constraints on the boundary of the shell is now considered on the basis of these boundary conditions. The first group of constraints corresponds to the four degrees of freedom $u_h, u_l, w, w_{,h}$. These *external* boundary conditions are modelling constraints belonging to the boundary of the two–dimensional region of the reference surface of the shell ($z = 0$), and determine in general the type of support for the shell. The second group of constraints given by the conditions (3.58) models the constraints on the boundary through the thickness of the shell as shown in Figure 3.3. This group of *internal* boundary conditions concerns the modelling of transverse shear and normal deformation at the edges of the shell. The laminate is modelled more accurately when both types of boundary conditions are considered.

Substituting the components of the stress tensor (3.45) into the integrals (3.48) and (3.49) yields the equations of elasticity, viz. the equations for the generalized forces and moments. Thus the in–plane forces may be expressed as

$$
\begin{aligned}
N_{11} &= B\varepsilon_{11} + \bar{B}\varepsilon_{22} + B_0\kappa_{11} + \bar{B}_0\kappa_{22} \\
&\quad + B_g\kappa^{(g)}_{11} + \bar{B}_q\kappa^{(g)}_{22} + (k_{11}C_q + k_{22}\bar{C}_q + H_q)\chi_q \\
N_{22} &\rightleftharpoons N_{11} \\
N_{12} &= (B - \bar{B})\varepsilon_{12} + (B_0 - \bar{B}_0)\kappa_{12} + (B_g - \bar{B}_g)\kappa^{(g)}_{11} \\
&\quad + k_{12}(C_q - \bar{C}_q)\chi_q
\end{aligned}
\tag{3.60}
$$

where $q = 1 \ldots 6$ and $g = 1 \ldots 8$. The bending and twisting moments may be expressed as

$$
\begin{aligned}
M_{11} &= B_0\varepsilon_{11} + \bar{B}_0\varepsilon_{22} + D_{00}\kappa_{11} + \bar{D}_{00}\kappa_{22} \\
&\quad + D_{0g}\kappa^{(g)}_{11} + \bar{D}_{0g}\kappa^{(g)}_{22} + (k_{11}C_{0q} + k_{22}\bar{C}_{0q} + H_{0q})\chi_q \\
M_{22} &\rightleftharpoons M_{11}
\end{aligned}
$$

$$
\begin{aligned}
M_{12} &= (B_0 - \bar{B}_0)\varepsilon_{12} + (D_{00} - \bar{D}_{00})\kappa_{12} \\
&\quad +(D_{0g} - \bar{D}_{0g})\kappa_{12}^{(g)} + k_{12}(C_{0q} - \bar{C}_{0q})\chi_q
\end{aligned} \tag{3.61}
$$

the higher–order forces and moments as

$$
\begin{aligned}
N_{11}^{(p)} &= T_p\varepsilon_{11} + \bar{T}_p\varepsilon_{22} + T_{p0}\kappa_{11} + \bar{T}_{p0}\kappa_{22} + T_{pg}\kappa_{11}^{(g)} \\
&\quad +\bar{T}_{pg}\kappa_{22}^{(g)} + (k_{11}L_{pq} + k_{22}\bar{L}_{pq} + P_{pq})\chi_q \\
N_{22}^{(p)} &\rightleftharpoons N_{11}^{(p)} \\
N_{12}^{(p)} &= (T_p - \bar{T}_p)\varepsilon_{12} + (T_{p0} - \bar{T}_{p0})\kappa_{12} \\
&\quad +(T_{pq} - \bar{T}_{pq})\kappa_{12}^{(g)} + k_{12}(L_{pq} - \bar{L}_{pq})\chi_q \\
M_{11}^{(p)} &= B_p\varepsilon_{11} + \bar{B}_p\varepsilon_{22} + D_{p0}\kappa_{11} + \bar{D}_{p0}\kappa_{22} + D_{pg}\kappa_{11}^{(g)} \\
&\quad +\bar{D}_{pg}\kappa_{22}^{(g)} + (k_{11}C_{pq} + k_{22}\bar{C}_{pq} + H_{pq})\chi_q \\
M_{22}^{(p)} &\rightleftharpoons M_{11}^{(p)} \\
M_{12}^{(p)} &= (B_p - \bar{B}_p)\varepsilon_{12} + (D_{p0} - \bar{D}_{p0})\kappa_{12} \\
&\quad +(D_{pg} - \bar{D}_{pg})\kappa_{12}^{(g)} + k_{12}(C_{pq} - \bar{C}_{pq})\chi_q
\end{aligned} \tag{3.62}
$$

and the shear forces as

$$
\begin{aligned}
Q_i &= \bar{D}_t\chi_{t,i} \\
Q_i^{(1)} &= D_t\chi_{t,i} \\
Q_3^{(p)} &= \bar{P}_p(\varepsilon_{11} + \varepsilon_{22}) + \bar{P}_{p0}(\kappa_{11} + \kappa_{22}) \\
&\quad +\bar{P}_{pg}(\kappa_{11}^{(g)} + \kappa_{22}^{(g)}) + R_{pg}(k_{11} + k_{22})\chi_q + \bar{R}_{pq}\chi_q
\end{aligned} \tag{3.63}
$$

where $p = 1, 2$, $t = 1, 7, 8$, $q = 1 \ldots 6$ and $g = 1 \ldots 8$.

The equations for the generalized forces and moments include the integrated stiffnesses of heterogeneous shells given by

$$
\begin{aligned}
B &= \int_{a_0}^{a_n} A_{11k}dz & \bar{B} &= \int_{a_0}^{a_n} A_{12k}dz \\
B_0 &= \int_{a_0}^{a_n} A_{11k}zdz & \bar{B}_0 &= \int_{a_0}^{a_n} A_{12k}zdz \\
B_g &= \int_{a_0}^{a_n} A_{11k}\psi_{gk}dz & \bar{B}_g &= \int_{a_0}^{a_n} A_{12k}\psi_{gk}dz \\
C_q &= \int_{a_0}^{a_n} A_{11k}\varphi_{qk}dz & \bar{C}_q &= \int_{a_0}^{a_n} A_{12k}\varphi_{qk}dz \\
H_q &= \int_{a_0}^{a_n} A_{13k}\alpha_{qk}dz & D_{00} &= \int_{a_0}^{a_n} A_{11k}z^2dz \\
\bar{D}_{00} &= \int_{a_0}^{a_n} A_{12k}z^2dz & D_{0g} &= \int_{a_0}^{a_n} A_{11k}\psi_{gk}zdz \\
\bar{D}_{0g} &= \int_{a_0}^{a_n} A_{12k}\psi_{gk}zdz & C_{0g} &= \int_{a_0}^{a_n} A_{11k}\varphi_{qk}zdz \\
\bar{C}_{0g} &= \int_{a_0}^{a_n} A_{12k}\varphi_{qk}zdz & H_{0q} &= \int_{a_0}^{a_n} A_{13k}\alpha_{qk}zdz \\
D_{p0} &= D_{0p} & \bar{D}_{p0} &= \bar{D}_{0p} \\
D_{pg} &= \int_{a_0}^{a_n} A_{11k}\psi_{pk}\psi_{gk}dz & \bar{D}_{pg} &= \int_{a_0}^{a_n} A_{12k}\psi_{pk}\psi_{gk}dz \\
C_{pq} &= \int_{a_0}^{a_n} A_{11k}\psi_{pk}\varphi_{qk}dz & \bar{C}_{pq} &= \int_{a_0}^{a_n} A_{12k}\psi_{pk}\varphi_{qk}dz \\
H_{pq} &= \int_{a_0}^{a_n} A_{13k}\psi_{pk}\alpha_{qk}dz & T_p &= \int_{a_0}^{a_n} A_{11k}\varphi_{pk}dz \\
\bar{T}_p &= \int_{a_0}^{a_n} A_{12k}\varphi_{pk}dz & T_{p0} &= \int_{a_0}^{a_n} A_{11k}\varphi_{pk}zdz \\
\bar{T}_{p0} &= \int_{a_0}^{a_n} A_{12k}\varphi_{pk}zdz & T_{pg} &= \int_{a_0}^{a_n} A_{11k}\varphi_{pk}\psi_{gk}dz \\
\bar{T}_{pg} &= \int_{a_0}^{a_n} A_{12k}\varphi_{pk}\psi_{gk}dz & L_{pg} &= \int_{a_0}^{a_n} A_{11k}\varphi_{pk}\varphi_{gk}dz \\
\bar{L}_{pg} &= \int_{a_0}^{a_n} A_{12k}\varphi_{pk}\varphi_{gk}dz & P_{pg} &= \int_{a_0}^{a_n} A_{13k}\varphi_{pk}\alpha_{gk}dz \\
\bar{D}_t &= \int_{a_0}^{a_n} f_{tk}dz & D_t &= \int_{a_0}^{a_n} f_{tk}\beta_{1k}dz \\
\bar{P}_p &= \int_{a_0}^{a_n} A_{13k}\alpha_{pk}dz & \bar{P}_{p0} &= \int_{a_0}^{a_n} A_{13k}\alpha_{pk}zdz \\
\bar{P}_{pg} &= \int_{a_0}^{a_n} A_{13k}\alpha_{pk}\psi_{gk}dz & R_{pg} &= \int_{a_0}^{a_n} A_{13k}\alpha_{pk}\varphi_{gk}dz \\
\bar{R}_{pg} &= \int_{a_0}^{a_n} A_{33k}\alpha_{pk}\alpha_{gk}dz
\end{aligned}
\tag{3.64}
$$

The system of governing differential equations for heterogeneous composite shells is expressed in matrix form as

$$[D]\{V\} = [D_p]\{p\} \tag{3.65}$$

where $[D]$ is the matrix of differential operators on the vector of unknown functions

$$\{V\} = \{u_1, u_2, w, \chi_1, \chi_2\}^T \tag{3.66}$$

and $[D_p]$ is the matrix of differential operators on the vector of given loads

$$\{p\} = \{p_1, p_2, p_3, p_3^{(1)}, p_3^{(2)}\}^T \tag{3.67}$$

The integrated stiffnesses, all of which are given in eqn. (3.64), appear in the matrices $[D]$ and $[D_p]$, in particular, as coefficients of differential operators.

The order of the general system of differential equations (3.65) is sixteen. Therefore eight boundary conditions have to be satisfied on each edge of the shell.

One of the advantages of this higher–order theory is that the equations for the tangential components of the displacement vector and the stress and strain tensors

consist of similar terms which separately take into account the states of pure bending, transverse shear and normal deformation. This enables efficient analytical and numerical application of this theory using an independent but analogous approximation of the components of the displacement vector which belong to these states. It also allows the experience gained in using analytical and numerical methods in similar applications but on the basis of classical theory to be extended to a nonclassical approach based on higher–order theory.

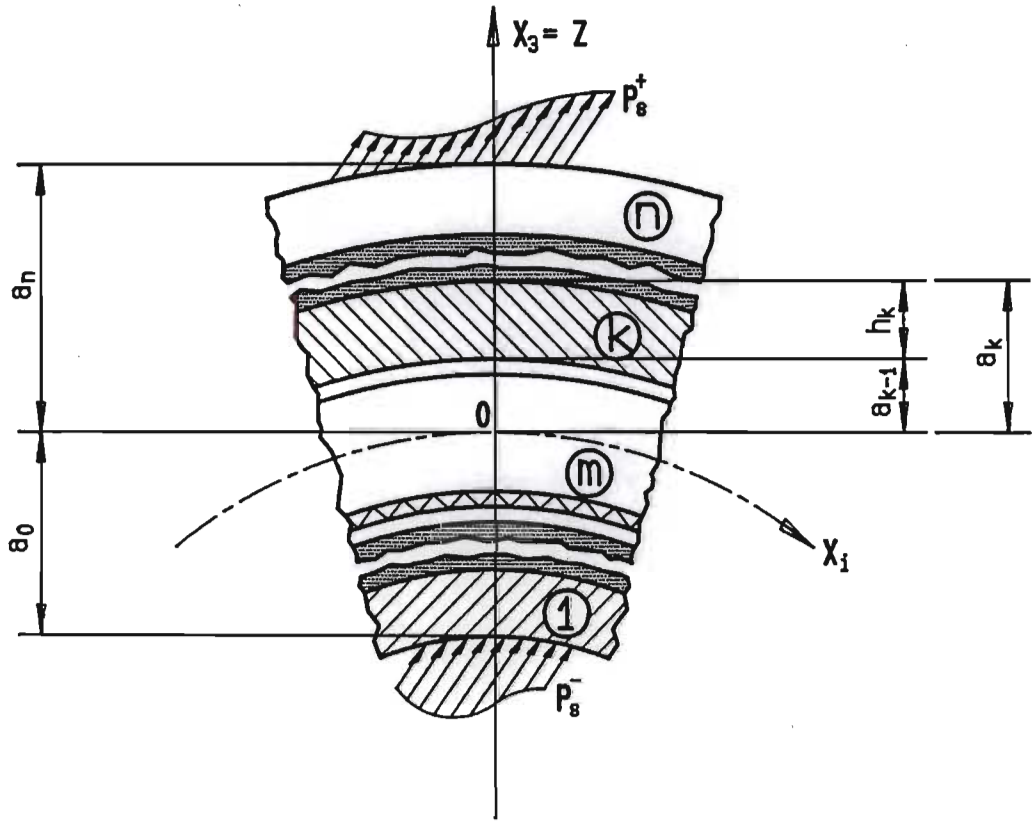


Figure 3.1. Geometry of laminated shell.

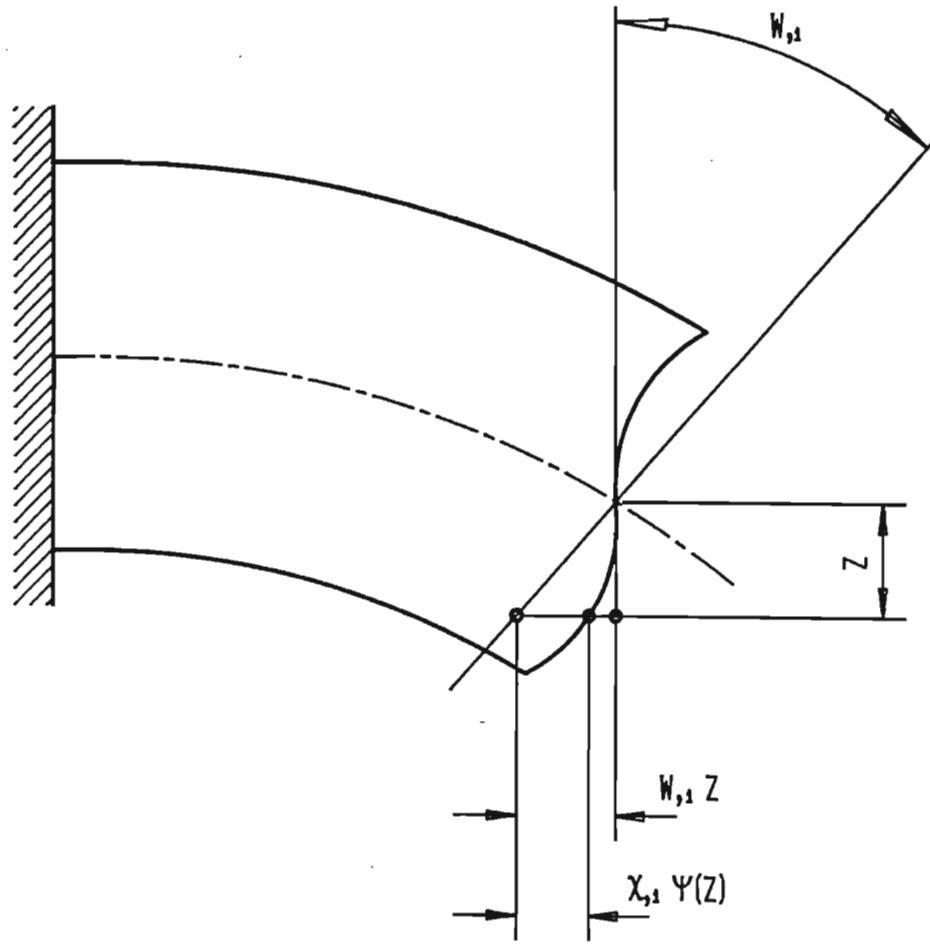
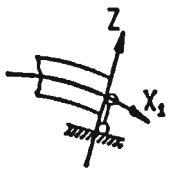
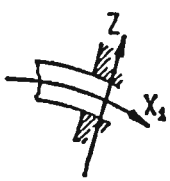
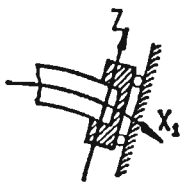
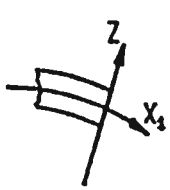


Figure 3.2. Kinematic model.

External Conditions ($z = 0$)

Moving Hinge	Clamped	Clamped, Moving	Free End
			
Constraints			
$u_2 = N_{12} = 0$; $w = M_{11} = 0$	$u_1 = u_2 = 0$; $w = w_{,1} = 0$	$u_1 = u_2 = 0$; $w_{,1} = M_{11,1} + 2M_{12,2} + p_1 = 0$	$N_{11} = N_{12} = 0$; $M_{11} = M_{11,1} + 2M_{12,2} + p_1 = 0$

Internal Conditions ($z \neq 0$)

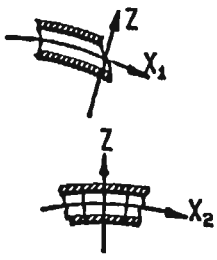
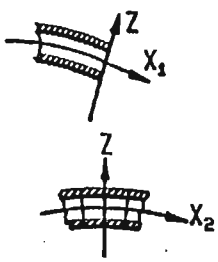
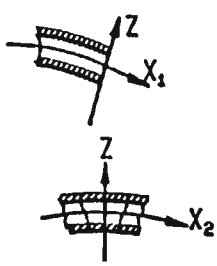
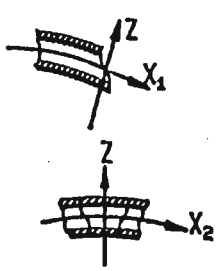
Flexible out of End Plane	Rigid	Flexible in End Plane	No Constraints
			
Constraints			
$\chi_p = M_{11}^{(p)} = 0;$ $p = 1,2$	$\chi_p = \chi_{p,l} = 0;$ $p = 1,2$	$\chi_{1,1} = M_{11,1}^{(1)} + 2M_{12,2}^{(1)} + Q_1^{(1)} + p_1^{(1)} = 0$	$M_{11}^{(1)} = M_{11,1}^{(1)} + 2M_{12,2}^{(1)} + Q_1^{(1)} + p_1^{(1)} = 0$
		$\chi_{2,1} = M_{11,1}^{(2)} + 2M_{12,2}^{(2)} + p_1^{(2)} = 0$	$M_{11}^{(2)} = M_{11,1}^{(2)} + 2M_{12,2}^{(2)} + p_1^{(2)} = 0$

Figure 3.3. Boundary Conditions.

3.6 Conclusions

A higher–order theory of laminated shells and plates subject to both transverse shear and normal deformation is derived. The proposed theory is capable of analysing thick plates and shells with an arbitrary number of transversely isotropic layers which have significantly different elastic properties. Moreover, the layers may be constructed of materials which have low transverse rigidity.

The kinematic hypotheses of the higher–order theory are not taken *a priori* but are formulated using an iterative technique where the classical Kirchhoff–Love hypotheses are assumed in the first iteration. Moreover, the new variables introduced by the theory have a clear physical meaning.

The unknown functions in the system of governing differential equations are defined on an arbitrary reference surface in the package of layers, and the order of the governing equations is independent of the number of layers.

Various loading and boundary conditions are considered which enable transverse shear and normal deformation to be fully taken into account, and the complete set of boundary conditions is derived.

Chapter 4

Implementation of the Higher–Order Theory using Symbolic Computation

4.1 Introduction

The objective of this chapter is to implement the higher–order theory presented in Chapter 3 for the numerical analysis of plates and shells based on an analytical solution.

The distribution functions and integrated stiffness constants of the higher–order theory involve multiple piecewise integrals through the thickness of the laminate, and in the general case these integrals cannot be expressed in an exact form for direct implementation into a computer program. Therefore symbolic computation is employed for the implementation of the higher–order theory.

The general purpose *Mathematica* symbolic computation system is used to derive the distribution functions, calculate the integrated stiffness constants, solve the system of governing differential equations analytically and finally to evaluate the stress/strain state for a given laminate.

In order to improve the computational efficiency of the analysis, special purpose symbolic computation routines are developed in the C programming language as an alternative to using a general purpose symbolic computation system such as *Mathematica*.

Numerical results based on the higher–order theory are obtained for thick plates and shells subject to a sinusoidal load. Both homogeneous and sandwich structures are considered and the effect of normal deformation is investigated.

4.2 Basic Equations and Some Analytical Solutions

On the basis of the higher–order theory, the system of governing differential equations for a rectangular shell with double curvature and subject to normal loading on both bounding surfaces, is derived as

$$
\begin{aligned}
&Bu_{1,11} + \tfrac{1}{2}(B-\bar{B})\,u_{1,22} + \tfrac{1}{2}(B+\bar{B})\,u_{2,12} \\
&\quad -B_0\nabla w_{,1} + (Bk_{11}+\bar{B}k_{22})\,w_{,1} \\
&\quad -B_1\nabla\chi_{1,1} + H_1\chi_{1,1} + (C_1k_{11}+\bar{C}_1k_{22})\,\chi_{1,1} \\
&\quad -B_2\nabla\chi_{2,1} + H_2\chi_{2,1} + (C_2k_{11}+\bar{C}_2k_{22})\,\chi_{2,1} \\
&\qquad = B_5\nabla q^-_{,1} - H_5q^-_{,1} - (C_5k_{11}+\bar{C}_5k_{22})\,q^-_{,1} \\
&\qquad\quad +B_6\nabla q^+_{,1} - H_6q^+_{,1} - (C_6k_{11}+\bar{C}_6k_{22})\,q^+_{,1}
\end{aligned}
\tag{4.1}
$$

$$
\begin{aligned}
&\tfrac{1}{2}(B+\bar{B})\,u_{1,12} + Bu_{2,22} + \tfrac{1}{2}(B-\bar{B})\,u_{2,11} \\
&\quad -B_0\nabla w_{,2} + (Bk_{22}+\bar{B}k_{11})\,w_{,2} \\
&\quad -B_1\nabla\chi_{1,2} + H_1\chi_{1,2} + (C_1k_{22}+\bar{C}_1k_{11})\,\chi_{1,2} \\
&\quad -B_2\nabla\chi_{2,2} + H_2\chi_{2,2} + (C_2k_{22}+\bar{C}_2k_{11})\,\chi_{2,2} \\
&\qquad = B_5\nabla q^-_{,2} - H_5q^-_{,2} - (C_5k_{11}+\bar{C}_5k_{22})\,q^-_{,2} \\
&\qquad\quad +B_6\nabla q^+_{,2} - H_6q^+_{,2} - (C_6k_{11}+\bar{C}_6k_{22})\,q^+_{,2}
\end{aligned}
\tag{4.2}
$$

$$
\begin{aligned}
&B_0\nabla u_{1,1} - (Bk_{11}+\bar{B}k_{22})\,u_{1,1} - B_0\nabla u_{2,2} - (Bk_{22}+\bar{B}k_{11})\,u_{2,2} \\
&\quad -D_{00}\nabla^2 w - [B(k_{11}^2+k_{22}^2) + 2\bar{B}k_{11}k_{22}]\,w \\
&\quad +2(B_0k_{11}+\bar{B}_0k_{22})\,w_{,11} + 2(B_0k_{22}+\bar{B}_0k_{11})\,w_{,22} \\
&\quad -D_{01}\nabla^2\chi_1 + H_{01}\nabla\chi_1 - [C_1(k_{11}^2+k_{22}^2) + 2\bar{C}_1k_{11}k_{22}]\,\chi_1 \\
&\quad +[\,(C_{01}+B_1)k_{11} + (\bar{C}_{01}+\bar{B}_1)k_{22}]\,\chi_{1,11} \\
&\quad +[\,(C_{01}+B_1)k_{22} + (\bar{C}_{01}+\bar{B}_1)k_{11}]\,\chi_{1,22} \\
&\quad -D_{02}\nabla^2\chi_2 + H_{02}\nabla\chi_2 - [C_2(k_{11}^2+k_{22}^2) + 2\bar{C}_2k_{11}k_{22}]\,\chi_2 \\
&\quad +[\,(C_{02}+B_2)k_{11} + (\bar{C}_{02}+\bar{B}_2)k_{22}]\,\chi_{2,11} \\
&\quad +[\,(C_{02}+B_2)k_{22} + (\bar{C}_{02}+\bar{B}_2)k_{11}]\,\chi_{2,22}
\end{aligned}
$$

$$
\begin{aligned}
&= D_{05}\nabla^2 q^- - H_{05}\nabla q^- \\
&\quad -[C_5(k_{11}^2+k_{22}^2)+2\bar{C}_5 k_{11}k_{22}-1]\, q^- \\
&\quad -[\,(C_{05}+B_5)k_{11}+(\bar{C}_{05}+\bar{B}_5)k_{22}]\, q^-_{,11} \\
&\quad -[\,(C_{05}+B_5)k_{22}+(\bar{C}_{05}+\bar{B}_5)k_{11}]\, q^-_{,22}
\end{aligned} \tag{4.3}
$$

$$
\begin{aligned}
&B_1\nabla u_{1,1} - H_1 u_{1,1} - (C_1 k_{11}+\bar{C}_1 k_{22})\, u_{1,1} \\
&\quad +B_1\nabla u_{2,2} - H_1 u_{2,2} - (C_1 k_{22}+\bar{C}_1 k_{11})\, u_{2,2} \\
&\quad +D_{01}\nabla^2 w - H_{01}\nabla w - [C_1(k_{11}^2+k_{22}^2)+2\bar{C}_1 k_{11}k_{22}]\, w \\
&\quad +[\,(C_{01}+B_1)k_{11}+(\bar{C}_{01}+\bar{B}_1)k_{22}]\, w_{,11} \\
&\quad +[\,(C_{01}+B_1)k_{22}+(\bar{C}_{01}+\bar{B}_1)k_{11}]\, w_{,22} \\
&\quad -D_{11}\nabla^2\chi_1 + (2H_{11}+D_1)\nabla\chi_1 \\
&\quad -[\bar{R}_{11}+L_{11}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{11}+2P_{11}(k_{11}+k_{22})]\,\chi_1 \\
&\quad +2(C_{11}k_{11}+\bar{C}_{11}k_{22})\,\chi_{1,11} \\
&\quad +2(C_{11}k_{22}+\bar{C}_{11}k_{11})\,\chi_{1,22} \\
&\quad -D_{12}\nabla^2\chi_2 + (2H_{12}+\bar{P}_{12})\nabla\chi_2 \\
&\quad -[\bar{R}_{12}+L_{12}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{12} \\
&\quad +(P_{12}+R_{12})(k_{11}+k_{22}]\,\chi_2 \\
&\quad +[\,(C_{12}+T_{12})k_{11}+(\bar{C}_{12}+\bar{T}_{12})k_{22}]\,\chi_{2,11} \\
&\quad +[\,(C_{12}+T_{12})k_{22}+(\bar{C}_{12}+\bar{T}_{12})k_{11}]\,\chi_{2,22} \\
&\qquad = D_{15}\nabla^2 q^- - (H_{15}+\bar{P}_{15})\nabla q^- \\
&\qquad\quad +[\bar{R}_{15}-\varphi_{11}(a_0)+(P_{15}+R_{15})(k_{11}+k_{22}) \\
&\qquad\qquad +L_{15}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{15}]\, q^- \\
&\qquad\quad -[\,(C_{15}+T_{15})k_{11}+(\bar{C}_{15}+\bar{T}_{15})k_{22}]\, q^-_{,11} \\
&\qquad\quad -[\,(C_{15}+T_{15})k_{22}+(\bar{C}_{15}+\bar{T}_{15})k_{11}]\, q^-_{,22} \\
&\qquad\quad +D_{16}\nabla^2 q^+ - (H_{16}+\bar{P}_{16})\nabla q^+ \\
&\qquad\quad +[\bar{R}_{16}-\varphi_{1n}(a_n)+(P_{16}+R_{16})(k_{11}+k_{22}) \\
&\qquad\qquad +L_{16}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{16}]\, q^+ \\
&\qquad\quad -[\,(C_{16}+T_{16})k_{11}+(\bar{C}_{16}+\bar{T}_{16})k_{22}]\, q^+_{,11} \\
&\qquad\quad -[\,(C_{16}+T_{16})k_{22}+(\bar{C}_{16}+\bar{T}_{16})k_{11}]\, q^+_{,22}
\end{aligned} \tag{4.4}
$$

$$
\begin{aligned}
&B_2\nabla u_{1,1} - H_2 u_{1,1} - (C_2 k_{11}+\bar{C}_2 k_{22})\, u_{1,1} \\
&\quad +B_2\nabla u_{2,2} - H_2 u_{2,2} - (C_2 k_{22}+\bar{C}_2 k_{11})\, u_{2,2} \\
&\quad +D_{02}\nabla^2 w - H_{02}\nabla w - [C_2(k_{11}^2+k_{22}^2)+2\bar{C}_2 k_{11}k_{22}]\, w
\end{aligned}
$$

$$
\begin{aligned}
&+[\,(C_{02}+B_2)k_{11}+(\bar{C}_{02}+\bar{B}_2)k_{22}]\,w_{,11}\\
&+[\,(C_{02}+B_2)k_{22}+(\bar{C}_{02}+\bar{B}_2)k_{11}]\,w_{,22}\\
&-D_{21}\nabla^2\chi_1+(H_{21}+\bar{P}_{21})\nabla\chi_1\\
&-[\bar{R}_{21}+L_{21}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{21}\\
&+(P_{21}+R_{21})(k_{11}+k_{22})]\,\chi_1\\
&+(C_{21}+T_{21})k_{11}+(\bar{C}_{21}+\bar{T}_{21})k_{22})\,\chi_{1,11}\\
&+(C_{21}+T_{21})k_{22}+(\bar{C}_{21}+\bar{T}_{21})k_{11})\,\chi_{1,22}\\
&-D_{22}\nabla^2\chi_2+(H_{22}+\bar{P}_{22})\nabla\chi_2\\
&-[\bar{R}_{22}+L_{22}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{22}+2P_{22}(k_{11}+k_{22}]\,\chi_2\\
&+2(C_{22}k_{11}+\bar{C}_{22}k_{22})\,\chi_{2,11}+2(C_{22}k_{22}+\bar{C}_{11}k_{22})\,\chi_{2,22}
\end{aligned}
$$

$$
\begin{aligned}
&= D_{25}\nabla^2q^- - (H_{25}+\bar{P}_{25})\nabla q^-\\
&\quad +[\bar{R}_{25}-\varphi_{21}(a_0)+(P_{25}+R_{25})(k_{11}+k_{22})\\
&\qquad +L_{25}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{25}]\,q^-\\
&\quad -[\,(C_{25}+T_{25})k_{11}+(\bar{C}_{25}+\bar{T}_{25})k_{22}]\,q^-_{,11}\\
&\quad -[\,(C_{25}+T_{25})k_{22}+(\bar{C}_{25}+\bar{T}_{25})k_{11}]\,q^-_{,22}\\
&\quad +D_{26}\nabla^2q^+ - (H_{26}+\bar{P}_{26})\nabla q^+\\
&\quad +[\bar{R}_{26}-\varphi_{2n}(a_n)+(P_{26}+R_{26})(k_{11}+k_{22})\\
&\qquad +L_{26}(k_{11}^2+k_{22}^2)+2k_{11}k_{22}\bar{L}_{26}]\,q^+\\
&\quad -[\,(C_{26}+T_{26})k_{11}+(\bar{C}_{26}+\bar{T}_{26})k_{22}]\,q^+_{,11}\\
&\quad -[\,(C_{26}+T_{26})k_{22}+(\bar{C}_{26}+\bar{T}_{26})k_{11}]\,q^+_{,22}
\end{aligned}
\qquad (4.5)
$$

where $u_1(x), u_2(x)$ are the displacements of the reference surface; $w(x)$ is the deflection of the reference surface; $\chi_1(x), \chi_2(x)$ are the *shear* and *compression* functions introduced by the higher–order theory; $q^+(x), q^-(x)$ are the normal loads on the bounding surfaces of the shell; k_{11}, k_{22} are the curvatures of the shell; and the integrated stiffness constants are given in Chapter 3.

In the case of a plate, i.e. $k_{11} = k_{22} = 0$, the system (4.1)—(4.5) reduces to

$$
\begin{aligned}
&Bu_{1,11}+\tfrac{1}{2}(B-\bar{B})u_{1,22}+\tfrac{1}{2}(B+\bar{B})u_{2,12}-B_0\nabla w_{,1}+H_1\chi_{1,1}-B_1\nabla\chi_{1,1}\\
&\quad +H_2\chi_{2,1}-B_2\nabla\chi_{2,1}=B_5\nabla q^-_{,1}-H_5\,q^-_{,1}+B_6\nabla q^+_{,1}-H_6\,q^+_{,1}\\
&\tfrac{1}{2}(B+\bar{B})u_{1,21}+Bu_{2,22}+\tfrac{1}{2}(B-\bar{B})u_{2,11}-B_0\nabla w_{,1}+H_1\chi_{1,2}-B_1\nabla\chi_{1,2}\\
&\quad +H_2\chi_{2,2}-B_2\nabla\chi_{2,2}=B_5\nabla q^-_{,2}-H_5\,q^-_{,2}+B_6\nabla q^+_{,2}-H_6\,q^+_{,2}\\
&-B_0\nabla u_{1,1}-B_0\nabla u_{2,2}-D_{00}\nabla^2w+D_{01}\nabla^2\chi_1-H_{01}\nabla\chi_1+D_{02}\nabla^2\chi_2\\
&\quad -H_{02}\nabla\chi_2=q^-+H_{05}\,\nabla q^-D_{05}\nabla^2q^-+q^++H_{06}\,\nabla q^+D_{06}\nabla^2q^+
\end{aligned}
\qquad (4.6)
$$

$$
\begin{aligned}
&H_1 u_{1,1} - B_1 \nabla u_{1,1} + H_1 u_{2,2} - B_1 \nabla u_{2,2} + D_{01} \nabla^2 w - H_{01} \nabla w \\
&\quad \bar{R}_{11} \chi_1 - (2H_{11} + D_1)\chi_1 + D_{11} \nabla^2 \chi_1 \bar{R}_{12} \chi_2 - (H_{12} + \bar{P}_{12})\chi_2 + D_{12} \nabla^2 \chi_2 \\
&\quad\quad = (\varphi_{11}(a_0) - \bar{R}_{15}) q^- + (H_{15} + \bar{P}_{15})\, \nabla q^- D_{15} \nabla^2 q^- \\
&\quad\quad\quad + (\varphi_{1n}(a_n) - \bar{R}_{16}) q^+ + (H_{16} + \bar{P}_{16})\, \nabla q^+ D_{16} \nabla^2 q^+
\end{aligned}
$$

$$
\begin{aligned}
&H_2 u_{1,1} - B_2 \nabla u_{1,1} + H_2 u_{2,2} - B_2 \nabla u_{2,2} + D_{02} \nabla^2 w - H_{02} \nabla w \\
&\quad \bar{R}_{21} \chi_1 - (H_{21} + \bar{P}_{21})\chi_1 + D_{21} \nabla^2 \chi_1 \bar{R}_{22} \chi_2 - (H_{22} + \bar{P}_{22})\chi_2 + D_{22} \nabla^2 \chi_2 \\
&\quad\quad = (\varphi_{21}(a_0) - \bar{R}_{25}) q^- + (H_{25} + \bar{P}_{22})\, \nabla q^- D_{25} \nabla^2 q^- \\
&\quad\quad\quad + (\varphi_{2n}(a_n) - \bar{R}_{26}) q^+ + (H_{26} + \bar{P}_{22})\, \nabla q^+ D_{26} \nabla^2 q^+
\end{aligned}
$$

In the present study, the system of governing equations is solved analytically using double trigonometric series approximations. The loading is expressed as

$$
q^{\pm}(x) = \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} a_{mn}^{\pm} \sin \lambda_m x_1 \sin \gamma_n x_2 \tag{4.7}
$$

where $\lambda_m = m\pi/b_1$, $\gamma_n = n\pi/b_2$ and $b_1 \times b_2$ are the dimensions of the shell.

The hinged–supported boundary conditions are satisfied if the unknown functions relating to the reference surface are given by

$$
\begin{aligned}
u_1 &= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} A_{mn} \cos \lambda_m x_1 \sin \gamma_n x_2 \\
u_2 &= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} B_{mn} \sin \lambda_m x_1 \cos \gamma_n x_2 \\
w &= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} C_{mn} \sin \lambda_m x_1 \sin \gamma_n x_2 \\
\chi_1 &= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} D_{mn} \sin \lambda_m x_1 \sin \gamma_n x_2 \\
\chi_2 &= \sum_{m=1}^{\infty} \sum_{n=1}^{\infty} E_{mn} \sin \lambda_m x_1 \sin \gamma_n x_2
\end{aligned} \tag{4.8}
$$

where A_{mn}, B_{mn}, C_{mn}, D_{mn} and E_{mn} are constants chosen to satisfy the system of differential equations.

Substituting eqns. (4.8) and (4.7) for a given pair (m, n) into the system of governing equations leads to a system of linear algebraic equations in terms of the constants A_{mn}, B_{mn}, C_{mn}, D_{mn} and E_{mn}. In the case of a plate with loading on the surface $z = a_n$ only, i.e. $q^- = 0$, $q^+ \neq 0$, the system of linear algebraic equations is given by

$$
-\tfrac{1}{2} A_{mn} (B\gamma_n^2 - \bar{B}\gamma_n^2 + 2B\lambda_m^2) - \tfrac{1}{2} B_{mn} (B + \bar{B}) \gamma_n \lambda_m
$$

$$
\begin{aligned}
&+E_{mn}\lambda_m(H_2 + B_2\gamma_n^2 + B_2\lambda_m^2) \\
&\quad = -a_{mn}^{+}\lambda_m(H_6 + B_6\gamma_n^2 + B_6\lambda_m^2)
\end{aligned}
$$

$$
\begin{aligned}
&-\tfrac{1}{2}A_{mn}(B+\bar{B})\gamma_n\lambda_m - \tfrac{1}{2}B_{mn}(2B\gamma_n^2 + B\lambda_m^2 - \bar{B}\lambda_m^2) \\
&\quad +E_{mn}\gamma_n(H_2 + B_2\gamma_n^2 + B_2\lambda_m^2) \\
&\quad\quad = -a_{mn}^{+}\gamma_n(H_6 + B_6\gamma_n^2 + B_6\lambda_m^2)
\end{aligned} \tag{4.9}
$$

$$
\begin{aligned}
&C_{mn}D_{00}(\gamma_n^2 + \lambda_m^2)^2 + D_{mn}(\gamma_n^2 + \lambda_m^2)(H_{01} + D_{01}\gamma_n^2 + D_{01}\lambda_m^2) \\
&\quad\quad = a_{mn}^{+}(1 - H_{06}\gamma_n^2 - D_{06}\gamma_n^4 - H_{06}\lambda_m^2 - 2D_{06}\gamma_n^2\lambda_m^2 + D_{06}\lambda_m^4)
\end{aligned}
$$

$$
\begin{aligned}
&C_{mn}(\gamma_n^2 + \lambda_m^2)(H_{01} + D_{01}\gamma_n^2 + D_{01}\lambda_m^2) + E_{mn}\bar{R}_{12} \\
&\quad +D_{mn}(\gamma_n^2 + \lambda_m^2)(2H_{11} + D_1 + D_{11}\gamma_n^2 + D_{11}\lambda_m^2) \\
&\quad\quad = a_{mn}^{+}(\varphi_{1n}(a_n) - \bar{R}_{16} - (H_{16} + \bar{P}_{16})\gamma_n^2 - D_{16}\gamma_n^4 - (H_{16} + \bar{P}_{16})\lambda_m^2 \\
&\quad\quad\quad -2D_{16}\gamma_n^2\lambda_m^2 - D_{16}\lambda_m^4)
\end{aligned}
$$

$$
\begin{aligned}
&-A_{mn}\lambda_m(H_2 + B_2\gamma_n^2 + B_2\lambda_m^2) - B_{mn}\gamma_n(H_2 + B_2\gamma_n^2 + B_2\lambda_m^2) \\
&\quad +D_{mn}\bar{R}_{21} + E_{mn}(\gamma_n^2 + \lambda_m^2)(H_{22} + \bar{P}_{22} + D_{22}\gamma_n^2 + D_{22}\lambda_m^2) \\
&\quad\quad = a_{mn}^{+}(\varphi_{2n}(a_n) - \bar{R}_{26} - (H_{26} + \bar{P}_{26})\gamma_n^2 - D_{26}\gamma_n^4 - (H_{26} + \bar{P}_{26})\lambda_m^2 \\
&\quad\quad\quad -2D_{26}\gamma_n^2\lambda_m^2 - D_{26}\lambda_m^4)
\end{aligned}
$$

The solution (4.8) is then substituted into the kinematic hypotheses of higher–order theory given in Chapter 3 as

$$
u_i^{(k)}(x,z) = u_i - w_{,i}z - \chi_{g,i}\psi_{gk} \quad (i = 1,2;\ g = 1\ldots 8) \tag{4.10}
$$

$$
u_3^{(k)}(x,z) = w + \chi_q\varphi_{qk} \quad (q = 1\ldots 6) \tag{4.11}
$$

where φ_k, φ_{qk} are distribution functions derived in Chapter 3, and for case of normal loading only we have

$$
\begin{aligned}
&\chi_3(x) = 0; \quad \chi_4(x) = 0; \quad \chi_5(x) = q^{-}; \\
&\chi_6(x) = q^{+}; \quad \chi_{7,i}(x) = 0; \quad \chi_{8,i}(x) = 0;
\end{aligned} \tag{4.12}
$$

Therefore using the hypotheses (4.10) and (4.11), the displacements and deflection at any point in the shell may be determined from the displacements and deflection of the reference surface.

The components of the strain tensor are given in Chapter 3 as

$$
\begin{aligned}
&e_{ij}^{(k)} = \varepsilon_{ij} + \kappa_{ij}z + \kappa_{ij}^{(g)}\psi_{gk} + k_{ij}\chi_q\varphi_{qk} && (g = 1\ldots 8) \\
&2e_{i3}^{(k)} = \chi_{t,i}\beta_{tk} && (t = 1,7,8) \\
&e_{33}^{(k)} = \chi_q\alpha_{qk} && (q = 1\ldots 6)
\end{aligned} \tag{4.13}
$$

where we denote $\kappa_{ij}^{(g)} = -\chi_{g,ij}$, α_{qk} and βtk are distribution functions given in Chapter 3, and ε_{ij} and κ_{ij} are the strains of the reference surface given by

$$\varepsilon_{ij} = \tfrac{1}{2}(u_{i,j} + u_{j,i}) + k_{ij}w \; ; \quad \kappa_{ij} = -w_{,ij} \tag{4.14}$$

The components of the stress tensor are determined from eqn. (4.13) using Hooke's Law and are given in Chapter 3 as

$$\begin{aligned} \sigma_{11}^{(k)} &= A_{11k}e_{11}^{(k)} + A_{12k}e_{22}^{(k)} + A_{13k}e_{33}^{(k)} & \sigma_{12}^{(k)} &= 2G'_k e_{12}^{(k)} \\ \sigma_{22}^{(k)} &= A_{12k}e_{11}^{(k)} + A_{11k}e_{22}^{(k)} + A_{13k}e_{33}^{(k)} & \sigma_{13}^{(k)} &= 2G'_k e_{13}^{(k)} \\ \sigma_{33}^{(k)} &= A_{13k}[e_{11}^{(k)} + e_{22}^{(k)}] + A_{33k}e_{33}^{(k)} & \sigma_{23}^{(k)} &= 2G'_k e_{23}^{(k)} \end{aligned} \tag{4.15}$$

where A_{11k}, A_{12k}, A_{13k}, A_{33k} are the stiffness parameters of the k-th layer (see Chapter 3).

Solution Procedure

In order to analyse a given structure on the basis of the above higher–order theory, firstly the distribution functions of the theory are derived and the integrated stiffnesses are calculated.

Using the trigonometric approximations given in eqns. (4.7) and (4.8), the system of governing equations (4.1)—(4.5) is solved analytically for the displacements $u_1(x), u_2(x)$ of the reference surface, the deflection $w(x)$ of the reference surface, and for the shear and compression functions $\chi_1(x)$ and $\chi_2(x)$, respectively.

From the kinematic hypotheses (4.10) and (4.11) the displacements and deflection through the thickness of the laminate at any point on the shell may be determined from the solution (4.8) and the distribution functions ψ_{gk}, φ_{qk}. Finally from eqns. (4.13) and (4.15) the stress–strain state of the shell may be determined.

4.3 Implementation using Mathematica

In this section, the higher–order theory is implemented using the *Mathematica* symbolic computation system. This involves four distinct tasks: the first task derives the distribution functions defined by the higher–order theory; the second calculates the integrated stiffness constants using the derived distribution functions; the third

solves the system of governing equations using the integrated stiffness constants calculated by the second task; and the fourth calculates the displacements, deflections, strains and stresses through the shell using the solution of the third task and the distribution functions derived by the first task.

4.3.1 Derivation of Distribution Functions

The first task in the *Mathematica* application is the derivation of the distribution functions defined by the higher–order theory. In general, these distribution functions are multiple piecewise integrals which involve the "layer integral" operator

$$\int_{a_m}^{z} \Longrightarrow \int_{a_{k-1}}^{z} + \sum_{r=m+1}^{k-1} \int_{a_{r-1}}^{a_r} \tag{4.16}$$

where $a_m \leq a_{k-1} \leq z < a_k$ (i.e., z is a point in the k-th layer of a laminate and $m < k$), and $a_0, a_1, \ldots a_n$ are the coordinates of the interfaces of a laminate with n layers. Note that z is the coordinate through the thickness of the laminate and $z = 0$ at the reference surface. For the case $z < a_m$ (ie. $k \leq m$), the layer integral operator is expressed as

$$\int_{a_m}^{z} \Longrightarrow \int_{a_k}^{z} + \sum_{r=k+1}^{m} \int_{a_r}^{a_{r-1}} \tag{4.17}$$

In the higher–order theory, the lower limit of the layer integrals is either $z = 0$ or $z = a_0$. In the implementation of the theory, an artificial interface is introduced at the coordinate $a_m = 0$ if the reference surface does not coincide with a *laminae interface.* This simplifies the program since the lower limit of integration may specified by an index, in particular 0 for a_0, and m for $a_m = 0$.

The distribution functions are the higher–order theory are given in Chapter 3 as

$$f_k(z) = \int_{a_0}^{z} E_{0k} dz \; ; \quad f_k^*(z) = \int_{a_0}^{z} E_{0k} z dz$$

$$B_f = \int_{a_0}^{a_n} E_{0k} dz \; ; \quad B_{f1} = \int_{a_0}^{a_n} E_{0k} z dz$$

$$f_{1k}(z) = f_k^* - f_k B_{f1}/B_f \; ; \quad f_{2k}(z) = f_k/B_f - 1 \; ; \quad f_{3k}(z) = f_k/B_f$$

$$F_{sk}(z) = \int_{a_0}^{z} f_{sk} dz \; ; \quad D_{fs} = \int_{a_0}^{a_n} f_{sk} dz \quad (s = 1, 2, 3)$$

$$f_{4k}(z) = F_{1k} D_{f2}/D_{f1} - F_{2k} \; ; \quad f_{5k}(z) = F_{1k} D_{f3}/D_{f1} - F_{3k}$$

$$f_{6k}(z) = F_{1k}/D_{f1} - 1 \; ; \quad f_{7k}(z) = F_{1k}/D_{f1}$$

$$
\begin{aligned}
&f_{8k}(z) = f_k - F_{1k}B_f/D_{f1}\ ; \quad f_{9k}(z) = f_k^* - F_{1k}B_{f1}/D_{f1} \\
&\alpha_{1k}(z) = \nu'_{0k}z\ ; \quad \alpha_{2k}(z) = -\nu'_{0k} \\
&\alpha_{qk}(z) = f_{gk}/E'_k \quad (q = 3\ldots6,\ g = q+1) \\
&\varphi_{qk}(z) = \int_0^z \alpha_{qk}dz \quad (q = 1\ldots6) \\
&\varphi_k(z) = f_{1k}/G'_k\ ; \quad \varphi_{7k}(z) = -f_{2k}/G'_k\ ; \quad \varphi_{8k}(z) = -f_{3k}/G'_k \\
&\psi_{1k}(z) = \int_0^z (\varphi_{1k} - \varphi_k)dz \\
&\psi_{gk}(z) = \int_0^z \varphi_{gk}dz \quad (g = 2\ldots8)
\end{aligned}
\tag{4.18}
$$

It is noted that the first two functions, viz. $f_k(z)$ and $f_k^*(z)$, are integrals whose integrands contain the material stiffness parameter E_{0k}. This parameter is determined from the elastic characteristics of the k-th layer. For a composite laminate, E_{0k} is constant through each layer, but depends on the layer number k. Only in the case of a homogeneous shell is E_{0k} constant through the entire thickness of the shell. Now consider the integral $f_k^*(z) = \int_{a_0}^z E_{0k}zdz$. This notation is expanded as

$$
\begin{aligned}
f_k^*(z) &= \int_{a_{k-1}}^z E_{0k}\zeta d\zeta + \sum_{r=1}^{k-1}\int_{a_{r-1}}^{a_r} E_{0r}\zeta d\zeta \\
&= \tfrac{1}{2}E_{0k}(z^2 - a_{k-1}^2) + \sum_{r=1}^{k-1}\tfrac{1}{2}E_{0r}(a_r^2 - a_{r-1}^2)
\end{aligned}
\tag{4.19}
$$

where z is a coordinate in the k-th layer. It is noted that $f_k^*(z)$ is defined only on the domain $a_{k-1} \le z \le a_k$. However, the notation $f_k^*(z)$ represents a set of n functions $f_1^*(z), f_2^*(z)\ldots f_n^*(z)$ for the n layers in the laminate. Therefore the function $f_k^*(z)$ is considered to be discontinuous with respect to the "argument" k. Moreover, the argument z may be considered to define k since $a_{k-1} \le z \le a_k$ and therefore $f_k^*(z)$ may be considered to be discontinuous with respect to its argument z, in particular, at the coordinates $a_1, a_2\ldots a_{n-1}$.

The entire set of distribution functions is built up from the functions $f_k(z)$ and $f_k^*(z)$ and it is clear that all of the distribution functions in eqns. (4.18) may be considered to be discontinuous at the laminae interfaces since each distribution function represents a set of n functions.

The distribution functions in eqns. (4.18), expressed using the layer integral notation, seem uncomplicated. However, every integral must be integrated piecewise and many of the distribution functions such as φ_{qk} and ψ_{gk} are multiple piecewise

integrals. The complexity of such integrals increases dramatically as the number of layers in the laminate increases. Even for a three–layered shell, the task of expanding the functions into exact formulas for numerical computation is too overwhelming to be attempted without the aid of symbolic computation.

In the *Mathematica* application, the layer integral operator $\int_{a_m}^{z}$ (where $a_{k-1} \leq z \leq a_k$) is defined by the code

```
LayInt[fn_,m_Integer,k_Integer,z_] := Block[{r,zeta},
    Return[If[m < k,
      Integrate[fn[k,zeta],{zeta,a[k-1],z}] +
      Sum[Integrate[fn[r,zeta],{zeta,a[r-1],a[r]}],{r,m+1,k-1}],
      If[m >= k,
      Integrate[fn[k,zeta],{zeta,a[k],z}] +
      Sum[Integrate[fn[r,zeta],{zeta,a[r],a[r-1]}],{r,k+1,m}],
      Integrate[fn[k,zeta],{zeta,a[k],z}]
      ]]];
```

where the argument `fn` is a function which takes two arguments, namely a z–coordinate ζ and the corresponding layer number r such that $a_{r-1} \leq \zeta \leq a_r$.

The distribution functions given in eqns. (4.18) are defined using the `LayInt` procedure as follows.

```
fsi [k_Integer,z_]  = e0[k] z;
fs  [k_Integer,z_]  = LayInt[fsi,0,k,z];
fi  [k_Integer,z_]  = e0[k];
f   [k_Integer,z_]  = LayInt[fi,0,k,z];

cbf = f[n,a[n]]; cbf1 = fs[n,a[n]];

f1 [k_Integer,z_]   = fs[k,z] - f[k,z] cbf1/cbf;
f2 [k_Integer,z_]   = f[k,z]/cbf - 1;
f3 [k_Integer,z_]   = f[k,z]/cbf;

F1[k_Integer,z_] = LayInt[f1,0,k,z];
F2[k_Integer,z_] = LayInt[f2,0,k,z];
F3[k_Integer,z_] = LayInt[f3,0,k,z];

cdf1 = Together[F1[n,a[n]]];
cdf2 = Together[F2[n,a[n]]];
```

```
cdf3 = Together[F3[n,a[n]]];

f4[k_Integer,z_] = F1[k,z] cdf2/cdf1 - F2[k,z];
f5[k_Integer,z_] = F1[k,z] cdf3/cdf1 - F3[k,z];
f6[k_Integer,z_] = F1[k,z]/cdf1 - 1;
f7[k_Integer,z_] = F1[k,z]/cdf1;
f8[k_Integer,z_] = f[k,z] - F1[k,z] cbf/cdf1;
f9[k_Integer,z_] = fs[k,z] - F1[k,z] cbf1/cdf1;

alp1[k_Integer,z_] = nu0[k] z;
alp2[k_Integer,z_] = -nu0[k];
alp3[k_Integer,z_] = f4[k,z]/e2[k];
alp4[k_Integer,z_] = f5[k,z]/e2[k];
alp5[k_Integer,z_] = f6[k,z]/e2[k];
alp6[k_Integer,z_] = f7[k,z]/e2[k];

vphi1[k_Integer,z_] = LayInt[alp1,m,k,z];
vphi2[k_Integer,z_] = LayInt[alp2,m,k,z];
vphi3[k_Integer,z_] = LayInt[alp3,m,k,z];
vphi4[k_Integer,z_] = LayInt[alp4,m,k,z];
vphi5[k_Integer,z_] = LayInt[alp5,m,k,z];
vphi6[k_Integer,z_] = LayInt[alp6,m,k,z];

vphi [k_Integer,z_] =  f1[k,z]/g2[k];
vphi7[k_Integer,z_] = -f2[k,z]/g2[k];
vphi8[k_Integer,z_] = -f3[k,z]/g2[k];

psi1[k_Integer,z_] = LayInt[vphi1,m,k,z] - LayInt[vphi,m,k,z];
psi2[k_Integer,z_] = LayInt[vphi2,m,k,z];
psi3[k_Integer,z_] = LayInt[vphi3,m,k,z];
psi4[k_Integer,z_] = LayInt[vphi4,m,k,z];
psi5[k_Integer,z_] = LayInt[vphi5,m,k,z];
psi6[k_Integer,z_] = LayInt[vphi6,m,k,z];
psi7[k_Integer,z_] = LayInt[vphi7,m,k,z];
psi8[k_Integer,z_] = LayInt[vphi8,m,k,z];
```

where `n` is the number of layers, `a[0]`, `a[1]` ... `a[n]` are the coordinates of the laminae interfaces, and `m` is the index of the laminae interface that coincides with the reference surface such that `a[m]`$= 0$. Also, `e0[k]`, `nu0[k]` and `g2[k]` are elastic

characteristics of the k-th layer.

The above *Mathematica* code defines the distribution functions for a general laminate. In order the derive the distribution functions for the three-layered symmetrically laminated shell, we define the following elastic and geometric constants

```
n = 4;
m = 2;

a[0]  = - h/2;
a[1]  = - h hr;
a[2]  =   0 ;
a[3]  =   h hr;
a[4]  =   h/2;

e1[1]  = be1; nu1[1] = bnu1; g1[1] = bg1;
e2[1]  = be2; nu2[1] = bnu2; g2[1] = bg2;
e1[2]  = fe1; nu1[2] = fnu1; g1[2] = fg1;
e2[2]  = fe2; nu2[2] = fnu2; g2[2] = fg2;
e1[3]  = fe1; nu1[3] = fnu1; g1[3] = fg1;
e2[3]  = fe2; nu2[3] = fnu2; g2[3] = fg2;
e1[4]  = be1; nu1[4] = bnu1; g1[4] = bg1;
e2[4]  = be2; nu2[4] = bnu2; g2[4] = bg2;

nu0   [k_Integer] = e1[k] nu2[k] / (e2[k] (1 - nu1[k]));
e0    [k_Integer] = e1[k] / (1 - nu1[k]^2);
del   [k_Integer] = (1 + nu1[k]) (1 - nu1[k] -
                      2 nu2[k]^2 e1[k]/e2[k]) / (e1[k]^2 e2[k]);
del11 [k_Integer] = (1 - nu2[k]^2 e1[k]/e2[k])/(e1[k] e2[k]);
del12 [k_Integer] = (nu1[k] + nu2[k]^2 e1[k]/e2[k]) / (e1[k] e2[k]);
del13 [k_Integer] = nu2[k] (1 + nu1[k]) / (e1[k] e2[k]);
del33 [k_Integer] = (1 - nu1[k]^2) / e1[k]^2;
a11   [k_Integer] = del11[k]/del[k];
a12   [k_Integer] = del12[k]/del[k];
a13   [k_Integer] = del13[k]/del[k];
a33   [k_Integer] = del33[k]/del[k];
```

where h is the thickness of the shell and hr is a parameter which determines the thickness of the core layer ($0 \leq$ hr $\leq \frac{1}{2}$). Also, e1[k], e2[k], nu1[k], nu2[k], g1[k], g2[k] are the elastic characteristics E, E', ν, ν', G, G' of the transversely

isotropic k-th layer; and in particular `be1`, `be2`, `bnu1`, `bnu2`, `bg1`, `bg2` are the elastic characteristics of the surface (or bearing) layers of the sandwich shell, and `fe1`, `fe2`, `fnu1`, `fnu2`, `fg1`, `fg2` are the elastic characteristics of the core (or filler) layer.

Once the material and geometric parameters have been specified, the distribution functions may be derived for each layer by the code

```
Do[f1[r,z_] = f1[r,z], {r,1,n}]
Do[f2[r,z_] = f2[r,z], {r,1,n}]
Do[f3[r,z_] = f3[r,z], {r,1,n}]

Do[F1[r,z_] = F1[r,z], {r,1,n}]
Do[F2[r,z_] = F2[r,z], {r,1,n}]
Do[F3[r,z_] = F3[r,z], {r,1,n}]

Do[f4[r,z_] = f4[r,z], {r,1,n}]
Do[f5[r,z_] = f5[r,z], {r,1,n}]
Do[f6[r,z_] = f6[r,z], {r,1,n}]

Do[alp1[r,z_]  = alp1[r,z], {r,1,n}];
Do[alp2[r,z_]  = alp2[r,z], {r,1,n}];
Do[alp3[r,z_]  = alp3[r,z], {r,1,n}];
Do[alp4[r,z_]  = alp4[r,z], {r,1,n}];
Do[alp5[r,z_]  = alp5[r,z], {r,1,n}];
Do[alp6[r,z_]  = alp6[r,z], {r,1,n}];

Do[beta1[r,z_] = beta1[r,z],{r,1,n}]
Do[beta7[r,z_] = beta7[r,z],{r,1,n}]
Do[beta8[r,z_] = beta8[r,z],{r,1,n}]

Do[vphi1[r,z_] = vphi1[r,z],{r,1,n}];
Do[vphi2[r,z_] = vphi2[r,z],{r,1,n}];
Do[vphi3[r,z_] = vphi3[r,z],{r,1,n}];
Do[vphi4[r,z_] = vphi4[r,z],{r,1,n}];
Do[vphi5[r,z_] = vphi5[r,z],{r,1,n}];
Do[vphi6[r,z_] = vphi6[r,z],{r,1,n}];

Do[psi1[r,z_]  = psi1[r,z], {r,1,n}];
Do[psi2[r,z_]  = psi2[r,z], {r,1,n}];
```

```
Do[psi3[r,z_]  = psi3[r,z], {r,1,n}];
Do[psi4[r,z_]  = psi4[r,z], {r,1,n}];
Do[psi5[r,z_]  = psi5[r,z], {r,1,n}];
Do[psi6[r,z_]  = psi6[r,z], {r,1,n}];
Do[psi7[r,z_]  = psi7[r,z], {r,1,n}];
Do[psi8[r,z_]  = psi8[r,z], {r,1,n}];
```

The computational efficiency of the above code heavily depends on the sequence of the derivations. For example, because functions ψ_{gk} are piecewise integrals of the functions φ_{qk}, the latter are derived first so that they are ready to be employed in the derivations of the functions ψ_{gk}. If the sequence were reversed, the functions φ_{qk} would be derived by *Mathematica* repetitively.

These distribution functions are required for the calculation of the integrated stiffness constants which appear in the system of governing equations, and for the determination of the displacements, deflection, strains and stresses through the thickness of the laminate.

4.3.2 Derivation of Integrated Stiffnesses

The second task of the *Mathematica* application is the calculation of the integrated stiffnesses using the distribution functions derived by the first task.

The integrated stiffness constants are definite layer integrals through the thickness of the laminate from a_0 to a_n. Their integrands contain stiffness parameters which depend on the layer number and distribution functions which are "discontinuous" at the laminae interfaces. In general, the integrated stiffnesses are multiple piecewise integrals which even for a three-layered laminate are too tedious to calculate exactly without the aid of symbolic computation.

For example, consider the integrated stiffnesses

$$
\begin{aligned}
B &= \int_{a_0}^{a_n} A_{11k}dz & \bar{B} &= \int_{a_0}^{a_n} A_{12k}dz \\
B_0 &= \int_{a_0}^{a_n} A_{11k}zdz & B_g &= \int_{a_0}^{a_n} A_{11k}\psi_{gk}dz \\
D_{00} &= \int_{a_0}^{a_n} A_{11k}z^2dz & \bar{D}_{00} &= \int_{a_0}^{a_n} A_{12k}z^2dz \\
D_{0g} &= \int_{a_0}^{a_n} A_{11k}\psi_{gk}zdz & D_{pg} &= \int_{a_0}^{a_n} A_{11k}\psi_{pk}\psi_{gk}dz \\
H_q &= \int_{a_0}^{a_n} A_{13k}\alpha_{qk}dz & H_{0q} &= \int_{a_0}^{a_n} A_{13k}\alpha_{qk}zdz \\
H_{pq} &= \int_{a_0}^{a_n} A_{13k}\psi_{pk}\alpha_{qk}dz & D_1 &= \int_{a_0}^{a_n} f_{1k}\beta_{1k}dz \\
\bar{P}_{pg} &= \int_{a_0}^{a_n} A_{13k}\alpha_{pk}\psi_{gk}dz & \bar{R}_{pg} &= \int_{a_0}^{a_n} A_{33k}\alpha_{pk}\varphi_{qk}dz
\end{aligned}
\tag{4.20}
$$

Clearly the integrals B, $\bar{B}$, B_0 and D_{00} may be readily calculated for any given laminate as

$$
\begin{aligned}
B &= \sum_{r=1}^{n} A_{11r}(a_r - a_{r-1}) \\
\bar{B} &= \sum_{r=1}^{n} A_{12r}(a_r - a_{r-1}) \\
B_0 &= \sum_{r=1}^{n} \tfrac{1}{2}A_{11r}(a_r^2 - a_{r-1}^2) \\
D_{00} &= \sum_{r=1}^{n} \tfrac{1}{3}A_{11r}(a_r^3 - a_{r-1}^3)
\end{aligned}
\tag{4.21}
$$

However, with the exception of those given in eqns. (4.21), the integrated stiffness contain distribution functions such as α_{qk}, φqk and ψ_{gk}, and in general require substantial symbolic processing in order to be calculated exactly.

In the *Mathematica* application, the integrands of the stiffness constants (4.20) which appear in the system (4.6) are defined by the code

```
cib    [k_Integer,z_]  = a11[k];
cibb   [k_Integer,z_]  = a12[k];
cib0   [k_Integer,z_]  = a11[k] z;
cib0b  [k_Integer,z_]  = a12[k] z;
cid00  [k_Integer,z_]  = a11[k] z^2;

cib1   [k_Integer,z_]  = a11[k] psi1[k,z];
cib2   [k_Integer,z_]  = a11[k] psi2[k,z];
cib5   [k_Integer,z_]  = a11[k] psi5[k,z];
cib6   [k_Integer,z_]  = a11[k] psi6[k,z];
```

```
cib1b  [k_Integer,z_]  = a12[k] psi1[k,z];
cib2b  [k_Integer,z_]  = a12[k] psi2[k,z];
cib5b  [k_Integer,z_]  = a12[k] psi5[k,z];
cib6b  [k_Integer,z_]  = a12[k] psi6[k,z];

cid01  [k_Integer,z_]  = a11[k] psi1[k,z] z;
cid02  [k_Integer,z_]  = a11[k] psi2[k,z] z;
cid05  [k_Integer,z_]  = a11[k] psi5[k,z] z;
cid06  [k_Integer,z_]  = a11[k] psi6[k,z] z;

cid11  [k_Integer,z_]  = a11[k] psi1[k,z] psi1[k,z];
cid12  [k_Integer,z_]  = a11[k] psi1[k,z] psi2[k,z];
cid22  [k_Integer,z_]  = a11[k] psi2[k,z] psi2[k,z];
cid15  [k_Integer,z_]  = a11[k] psi1[k,z] psi5[k,z];
cid16  [k_Integer,z_]  = a11[k] psi1[k,z] psi6[k,z];
cid25  [k_Integer,z_]  = a11[k] psi2[k,z] psi5[k,z];
cid26  [k_Integer,z_]  = a11[k] psi2[k,z] psi6[k,z];

cih1   [k_Integer,z_]  = a13[k] alp1[k,z];
cih2   [k_Integer,z_]  = a13[k] alp2[k,z];
cih5   [k_Integer,z_]  = a13[k] alp5[k,z];
cih6   [k_Integer,z_]  = a13[k] alp6[k,z];

cih01  [k_Integer,z_]  = a13[k] alp1[k,z] z;
cih02  [k_Integer,z_]  = a13[k] alp2[k,z] z;
cih05  [k_Integer,z_]  = a13[k] alp5[k,z] z;
cih06  [k_Integer,z_]  = a13[k] alp6[k,z] z;

cih11  [k_Integer,z_]  = a13[k] psi1[k,z] alp1[k,z];
cih12  [k_Integer,z_]  = a13[k] psi1[k,z] alp2[k,z];
cih21  [k_Integer,z_]  = a13[k] psi2[k,z] alp1[k,z];
cih22  [k_Integer,z_]  = a13[k] psi2[k,z] alp2[k,z];
cih15  [k_Integer,z_]  = a13[k] psi1[k,z] alp5[k,z];
cih16  [k_Integer,z_]  = a13[k] psi1[k,z] alp6[k,z];
cih25  [k_Integer,z_]  = a13[k] psi2[k,z] alp5[k,z];
cih26  [k_Integer,z_]  = a13[k] psi2[k,z] alp6[k,z];
```

Before any numerical computations are performed, the geometric and material pa-

rameters of a given laminate are specified. For example,

```
h    = 1/100
hr   = 1/4

be1  = 2 10^5
be2  = 2 10^5
bnu1 = 3/10
bnu2 = 3/10

fe1  = 2 10^3
fe2  = 2 10^3
fnu1 = 3/10
fnu2 = 3/10
```

The above code specifies that the sandwich shell is 1cm thick and the core layer is half the total thickness of the shell. The surface and core layers of the sandwich shell are defined to be isotropic ($E = E'$) and the surface layers are two orders of magnitude of stiffer than the core layer.

The stiffness constants may then be calculated for the given laminate using the `LayInt` operator as follows

```
cb   = N[LayInt[cib,   0,n,a[n]]];
cbb  = N[LayInt[cibb,  0,n,a[n]]];
cb0  = N[LayInt[cib0,  0,n,a[n]]];
cb1  = N[LayInt[cib1,  0,n,a[n]]];
cb2  = N[LayInt[cib2,  0,n,a[n]]];
cb5  = N[LayInt[cib5,  0,n,a[n]]];
cb6  = N[LayInt[cib6,  0,n,a[n]]];
cb0b = N[LayInt[cib0b, 0,n,a[n]]];
cb1b = N[LayInt[cib1b, 0,n,a[n]]];
cb2b = N[LayInt[cib2b, 0,n,a[n]]];
cb5b = N[LayInt[cib5b, 0,n,a[n]]];
cb6b = N[LayInt[cib6b, 0,n,a[n]]];
```

4.3.3 System of Governing Equations

The third task of the *Mathematica* application is to solve the system of governing differential equations. The trigonometric approximations given in eqns. (4.7)

and (4.8) are employed to solve the system analytically. This leads to a system of algebraic equations for the constants A_{mn}, B_{mn}, C_{mn}, D_{mn} and E_{mn} in the analytical solution (4.8).

In order to simplify the code, the following differential operators are defined:

```
Dn[f_]   := D[f,x1,x1] + D[f,x2,x2]
Dnn[f_]  := Dn[Dn[f]]
D1[f_]   := D[f,x1]
D2[f_]   := D[f,x2]
D11[f_]  := D[f,x1,x1]
D12[f_]  := D[f,x1,x2]
D21[f_]  := D[f,x2,x1]
D22[f_]  := D[f,x2,x2]
Dn1[f_]  := D[D[f,x1],x1,x1] + D[D[f,x1],x2,x2]
Dn2[f_]  := D[D[f,x2],x1,x1] + D[D[f,x2],x2,x2]
```

The trigonometric approximations (4.8) for the unknown functions in the system of governing equations are defined for a given pair (m, n) by the code

```
u1    = Amn   Cos[lam x1] Sin[gam x2]
u2    = Bmn   Sin[lam x1] Cos[gam x2]
w     = Cmn   Sin[lam x1] Sin[gam x2]
chi1  = Dmn   Sin[lam x1] Sin[gam x2]
chi2  = Emn   Sin[lam x1] Sin[gam x2]
```

where `lam` and `gam` are symbols for λ_m and γ_n respectively.

The loading $q^{\pm}(x)$ in eqn. (4.7) is defined by the code

```
q3p   = amn$p Sin[lam x1] Sin[gam x2]
q3m   = amn$m Sin[lam x1] Sin[gam x2]
```

The left–hand sides of the equations in system (4.6), for example, are defined by the code

```
eq1 = cb D11[u1] + (cb - cbb)/2 D22[u1] +
        (cb + cbb)/2 D12[u2] + (-cb0) Dn1[w] +
        (-cb1) Dn1[chi1] + ch1 D1[chi1] +
        (-cb2) Dn1[chi2] + ch2 D1[chi2]

eq2 = (cb + cbb)/2 D21[u1] + cb D22[u2] +
        (cb - cbb)/2 D11[u2] + (-cb0) Dn2[w] +
```

```
          (-cb1) Dn2[chi1] + ch1 D2[chi1] +
          (-cb2) Dn2[chi2] + ch2 D2[chi2]

eq3 = (-cb0) Dn1[u1] + (-cb0) Dn2[u2] + cd00 Dnn[w] +
          cd01 Dnn[chi1] + (-ch01) Dn[chi1] +
          cd02 Dnn[chi2] + (-ch02) Dn[chi2]

eq4 = (-cb1) Dn1[u1] + ch1 D1[u1] + (-cb1) Dn2[u2] +
          ch1 D2[u2] + cd01 Dnn[w]  + (-ch01) Dn[w] +
          cd11 Dnn[chi1] + (-chs11) Dn[chi1] + crb11 chi1 +
          cd12 Dnn[chi2] + (-chs12) Dn[chi2] + crb12 chi2

eq5 = (-cb2) Dn1[u1] + ch2 D1[u1] + (-cb2) Dn2[u2] +
          ch2 D2[u2] + cd02 Dnn[w] + (-ch02) Dn[w] +
          cd21 Dnn[chi1] + (-chs21) Dn[chi1] +
          crb21 chi1 + cd22 Dnn[chi2] +
          (-chs22) Dn[chi2] + crb22 chi2
```

and the right–hand sides by

```
r1  = cb6 Dn1[q3p] - ch6 D1[q3p]
r2  = cb6 Dn2[q3p] - ch6 D2[q3p]
r3  = q3p + ch06 Dn[q3p] - cd06 Dnn[q3p]
r4  = crbs16 q3p + chs16 Dn[q3p] - cd16 Dnn[q3p]
r5  = crbs26 q3p + chs26 Dn[q3p] - cd26 Dnn[q3p]

r1 += cb5 Dn1[q3m] - ch5 D1[q3m]
r2 += cb5 Dn2[q3m] - ch5 D2[q3m]
r3 += q3m + ch05 Dn[q3m] - cd05 Dnn[q3m]
r4 += crbs15 q3m + chs15 Dn[q3m] - cd15 Dnn[q3m]
r5 += crbs25 q3m + chs25 Dn[q3m] - cd25 Dnn[q3m]
```

The expressions for the left–hand sides of the system of equations are simplified into algebraic expressions in terms of the unknowns A_{mn}, B_{mn}, C_{mn}, D_{mn} and E_{mn} by the code

```
eqa1 = Cancel[eq1/Cos[lam x1]/Sin[gam x2]]
eqa2 = Cancel[eq2/Sin[lam x1]/Cos[gam x2]]
eqa3 = Cancel[eq3/Sin[lam x1]/Sin[gam x2]]
eqa4 = Cancel[eq4/Sin[lam x1]/Sin[gam x2]]
eqa5 = Cancel[eq5/Sin[lam x1]/Sin[gam x2]]
```

and the right-hand sides by

```
ra1  = Cancel[r1/Cos[lam x1]/Sin[gam x2]]
ra2  = Cancel[r2/Sin[lam x1]/Cos[gam x2]]
ra3  = Cancel[r3/Sin[lam x1]/Sin[gam x2]]
ra4  = Cancel[r4/Sin[lam x1]/Sin[gam x2]]
ra5  = Cancel[r5/Sin[lam x1]/Sin[gam x2]]
```

The system of algebraic equations may now be solved by the code

```
amn$p = -q0
amn$m = 0

lambda[mw_,nw_] := mw Pi/a1;
gamma [mw_,nw_] := nw Pi/a2;

mw = nw = 1;

lam = lambda[mw,nw];
gam = gamma[mw,nw]

soln = N[Solve[{eqa1 == ra1, eqa2 == ra2, eqa3 == ra3,
   eqa4 == ra4, eqa5 == ra5},
   {Amn,Bmn,Cmn,Dmn,Emn}
   ]];

{Ac[mw,nw]} = Amn /. soln;
{Bc[mw,nw]} = Bmn /. soln;
{Cc[mw,nw]} = Cmn /. soln;
{Dc[mw,nw]} = Dmn /. soln;
{Ec[mw,nw]} = Emn /. soln;
```

where it is specified that a sinusoidal load of amplitude $q0$ is acting on the top surface of the shell.

Finally, the solution is defined as

```
u1[x1_,x2_]   := Ac[mw,nw] Cos[lam x1] Sin[gam x2];
u2[x1_,x2_]   := Bc[mw,nw] Sin[lam x1] Cos[gam x2];
w[x1_,x2_]    := Cc[mw,nw] Sin[lam x1] Sin[gam x2];
chi1[x1_,x2_] := Dc[mw,nw] Sin[lam x1] Sin[gam x2];
chi2[x1_,x2_] := Ec[mw,nw] Sin[lam x1] Sin[gam x2];
```

4.3.4 Stresses and Strains

The fourth task of the *Mathematica* application is the calculation of the displacements, deflection, strains and stresses through the thickness of the laminate from the solution (4.8) using the kinematic hypotheses of the higher–order theory.

The kinematic hypotheses given in eqns. (4.10) and (4.11) and the strains given in eqns. (4.13) include the solution (4.8) which is determined by the third task of the application, and distribution functions of the higher–order theory which are derived by the first task.

In order to simplify the code, a notation for the derivatives of $u_i(x), w, \chi_{gk}$ are defined, for example

```
w$1[x1_,x2_]     := D[w[xd1,xd2],xd1]     /. {xd1 -> x1, xd2 -> x2};
w$2[x1_,x2_]     := D[w[xd1,xd2],xd2]     /. {xd1 -> x1, xd2 -> x2};
w$11[x1_,x2_]    := D[w[xd1,xd2],xd1,xd1] /. {xd1 -> x1, xd2 -> x2};
w$12[x1_,x2_]    := D[w[xd1,xd2],xd1,xd2] /. {xd1 -> x1, xd2 -> x2};
w$22[x1_,x2_]    := D[w[xd1,xd2],xd2,xd2] /. {xd1 -> x1, xd2 -> x2};
u1$1[x1_,x2_]    := D[u1[xd1,xd2],xd1]    /. {xd1 -> x1, xd2 -> x2};
u1$2[x1_,x2_]    := D[u1[xd1,xd2],xd2]    /. {xd1 -> x1, xd2 -> x2};
u2$1[x1_,x2_]    := D[u2[xd1,xd2],xd1]    /. {xd1 -> x1, xd2 -> x2};
u2$2[x1_,x2_]    := D[u2[xd1,xd2],xd2]    /. {xd1 -> x1, xd2 -> x2};
chi1$1[x1_,x2_] := D[chi1[xd1,xd2],xd1]  /. {xd1 -> x1, xd2 -> x2};
chi2$1[x1_,x2_] := D[chi2[xd1,xd2],xd1]  /. {xd1 -> x1, xd2 -> x2};
chi5$1[x1_,x2_] := D[chi5[xd1,xd2],xd1]  /. {xd1 -> x1, xd2 -> x2};
chi6$1[x1_,x2_] := D[chi6[xd1,xd2],xd1]  /. {xd1 -> x1, xd2 -> x2};
```

where, for example, `chi2$1` denotes the derivative $\chi_{2,1}$.

The displacements (4.10) and deflection (4.11) are defined by

```
u1k[x1_,x2_,k_Integer,z_] = Expand[
   u1[x1,x2] - w$1[x1,x2] z -
   chi1$1[x1,x2] psi1[k,z] - chi2$1[x1,x2] psi2[k,z] -
   chi5$1[x1,x2] psi5[k,z] - chi6$1[x1,x2] psi6[k,z]
   ];

u2k[x1_,x2_,k_Integer,z_] = Expand[
   u2[x1,x2] - w$2[x1,x2] z -
   chi1$2[x1,x2] psi1[k,z] - chi2$2[x1,x2] psi2[k,z] -
```

```
   chi5$2[x1,x2] psi5[k,z] - chi6$2[x1,x2] psi6[k,z]
   ];

u3k[x1_,x2_,k_Integer,z_] :=
   w[x1,x2] + chi1[x1,x2] vphi1[k,z] + chi2[x1,x2] vphi2[k,z] +
   chi5[x1,x2] vphi5[k,z] + chi6[x1,x2] vphi6[k,z]
```

The strains of the reference surface given by eqns. (4.14) are calculated by the code

```
ser11[x1_,x2_] := N[(u1$1[x1,x2] + u1$1[x1,x2])/2 + k11 w[x1,x2]];
ser22[x1_,x2_] := N[(u2$2[x1,x2] + u2$2[x1,x2])/2 + k22 w[x1,x2]];
ser12[x1_,x2_] := N[(u1$2[x1,x2] + u2$1[x1,x2])/2 + k12 w[x1,x2]];

skr11[x1_,x2_] := N[-w$11[x1,x2]];
skr22[x1_,x2_] := N[-w$22[x1,x2]];
skr12[x1_,x2_] := N[-w$12[x1,x2]];
skr21[x1_,x2_] := N[-w$21[x1,x2]];
```

The strains through the thickness given by eqns. (4.13) are calculated by the code

```
se11[x1_,x2_,k_Integer,z_] := Expand[N[
   ser11[x1,x2] + skr11[x1,x2] z -
   chi1$11[x1,x2] psi1[k,z] - chi2$11[x1,x2] psi2[k,z] -
   chi5$11[x1,x2] psi5[k,z] - chi6$11[x1,x2] psi6[k,z] +
   k11 (chi1[x1,x2] vphi1[k,z] + chi2[x1,x2] vphi2[k,z] +
        chi5[x1,x2] vphi5[k,z] + chi6[x1,x2] vphi6[k,z])
   ]];

se22[x1_,x2_,k_Integer,z_] := Expand[N[
   ser22[x1,x2] + skr22[x1,x2] z -
   chi1$22[x1,x2] psi1[k,z] - chi2$22[x1,x2] psi2[k,z] -
   chi5$22[x1,x2] psi5[k,z] - chi6$22[x1,x2] psi6[k,z] +
   k22 (chi1[x1,x2] vphi1[k,z] + chi2[x1,x2] vphi2[k,z] +
        chi5[x1,x2] vphi5[k,z] + chi6[x1,x2] vphi6[k,z])
   ]];

se12[x1_,x2_,k_Integer,z_] := Expand[N[
   ser12[x1,x2] + skr12[x1,x2] z -
   chi1$12[x1,x2] psi1[k,z] - chi2$12[x1,x2] psi2[k,z] -
   chi5$12[x1,x2] psi5[k,z] - chi6$12[x1,x2] psi6[k,z]
```

```
]];

se13[x1_,x2_,k_Integer,z_] := Expand[N[chi1$1[x1,x2] beta1[k,z]/2]];
se23[x1_,x2_,k_Integer,z_] := Expand[N[chi1$2[x1,x2] beta1[k,z]/2]];
se33[x1_,x2_,k_Integer,z_] := Expand[N[chi1[x1,x2] alp1[k,z] +
   chi2[x1,x2] alp2[k,z] + chi5[x1,x2] alp5[k,z] +
   chi6[x1,x2] alp6[k,z]
   ]];
```

Finally, the stresses through the thickness given by eqns. (4.15) are calculated by the code

```
ss11[x1_,x2_,k_Integer,z_] := Expand[N[a11[k]  se11[x1,x2,k,z] +
   a12[k] se22[x1,x2,k,z] + a13[k] se33[x1,x2,k,z]]];
ss22[x1_,x2_,k_Integer,z_] := Expand[N[a12[k] se11[x1,x2,k,z] +
   a11[k] se22[x1,x2,k,z] + a13[k] se33[x1,x2,k,z]]];
ss33[x1_,x2_,k_Integer,z_] := Expand[N[a13[k] se11[x1,x2,k,z] +
   a13[k] se22[x1,x2,k,z] + a33[k] se33[x1,x2,k,z]]];

ss12[x1_,x2_,k_Integer,z_] := Expand[N[2 g2[k] se12[x1,x2,k,z]]];
ss13[x1_,x2_,k_Integer,z_] := Expand[N[2 g2[k] se13[x1,x2,k,z]]];
ss23[x1_,x2_,k_Integer,z_] := Expand[N[2 g2[k] se23[x1,x2,k,z]]];
```

4.4 Homogeneous Shell

In order to illustrate the higher–order model, a transversely isotropic homogeneous shell is considered. The homogeneous shell is modelled as a special case of a sandwich shell with the same elastic constants for the surface and core layers.

As given in Appendix A, the distribution functions for the normal displacements are derived by *Mathematica* as

$$\begin{aligned}
\varphi_1(z) &= \frac{E\nu'}{2E'(1-\nu)}z^2 \\
\varphi_2(z) &= -\frac{E\nu'}{E'(1-\nu)}z \\
\varphi_3(z) &= \frac{1}{4E'}\left(\frac{z^4}{h^2} - \frac{2z^3}{3h} - \frac{z^2}{2} + \frac{hz}{2}\right)
\end{aligned}$$

$$
\begin{aligned}
\varphi_4(z) &= -\frac{1}{4E'}\left(\frac{z^4}{h^2}+\frac{2z^3}{3h}-\frac{z^2}{2}-\frac{hz}{2}\right)\\
\varphi_5(z) &= -\frac{1}{2E'}\left(\frac{z^4}{h^3}-\frac{3z^2}{2h}+z\right)\\
\varphi_6(z) &= -\frac{1}{2E'}\left(\frac{z^4}{h^3}-\frac{3z^2}{2h}-z\right)
\end{aligned}
\tag{4.22}
$$

and the distribution functions for the tangential displacements are derived as

$$
\begin{aligned}
\psi_1(z) &= \frac{E\nu'}{6E'(1-\nu)}z^3-\frac{E}{2G'(1-\nu^2)}\left(\frac{z^3}{3}-\frac{h^2z}{4}\right)\\
\psi_2(z) &= -\frac{E\nu'}{2E'(1-\nu)}z^2\\
\psi_3(z) &= \frac{1}{8E'}\left(\frac{2z^5}{5h^2}-\frac{z^4}{3h}-\frac{z^3}{3}+\frac{hz^2}{2}\right)\\
\psi_4(z) &= -\frac{1}{8E'}\left(\frac{2z^5}{5h^2}+\frac{z^4}{3h}-\frac{z^3}{3}-\frac{hz^2}{2}\right)\\
\psi_5(z) &= -\frac{1}{4E'}\left(\frac{2z^5}{5h^3}-\frac{z^3}{h}+z^2\right)\\
\psi_6(z) &= -\frac{1}{4E'}\left(\frac{2z^5}{5h^3}-\frac{z^3}{h}-z^2\right)\\
\psi_7(z) &= \frac{1}{2G'}\left(z-\frac{z^2}{h}\right)\\
\psi_8(z) &= -\frac{1}{2G'}\left(z+\frac{z^2}{h}\right)
\end{aligned}
\tag{4.23}
$$

Consider the components of the displacement vector expressed as polynomials in terms of z, viz.

$$
\begin{aligned}
u_i^{(k)} &= a_0+a_1z+a_2z^2+a_3z^3+a_4z^4+a_5z^5\\
u_3^{(k)} &= b_0+b_1z+b_2z^2+b_3z^3+b_4z^4
\end{aligned}
\tag{4.24}
$$

Substituting the distribution functions (4.22) and (4.23) for the homogeneous shell into displacements (4.10) and (4.11), the coefficients of the polynomials (4.24) are determined as

$$
\begin{aligned}
a_0 &= u_i\\
b_0 &= w\\
a_1 &= -w_{,i}-\chi_{1,i}\frac{Eh^2}{8G'(1-\nu^2)}\\
b_1 &= -\chi_2\frac{E\nu'}{E'(1-\nu)}-\frac{1}{2E'}(q^- - q^+)
\end{aligned}
$$

$$
\begin{aligned}
a_2 &= \chi_{2,i}\frac{E\nu'}{2E'(1-\nu)} + \frac{1}{4E'}(q_{,i}^- - q_{,i}^+) \\
b_2 &= \chi_1\frac{E\nu'}{2E'(1-\nu)} + \frac{3}{4hE'}(q^- + q^+) \\
a_3 &= \chi_{1,i}\left[\frac{E}{6G'(1-\nu^2)} - \frac{E\nu'}{6E'(1-\nu)}\right] - \frac{1}{4hE'}(q_{,i}^- + q_{,i}^+) \\
b_3 &= 0 \\
a_4 &= 0 \\
b_4 &= -\frac{1}{2h^3E'}(q^- + q^+) \\
a_5 &= \frac{1}{10h^3E'}(q_{,i}^- + q_{,i}^+)
\end{aligned}
\tag{4.25}
$$

Clearly the displacements vary through the thickness in a nonlinear manner: the tangential displacements as a fifth order polynomial and the normal displacements as a fourth order polynomial. In terms of the kinematic model, this implies that the normal to the reference surface is distorted and changed in length by the deformation.

If the effect of normal reduction is neglected ($E_k' = \infty$), the above equations of the higher–order theory take into account the effect of transverse shear only on the stress and strain state of the shell. If $E' = \infty$ then $b_1 = b_2 = b_3 = b_4 = 0$ and the normal displacements are constant through the thickness ($u_3^{(k)} = w$). For this case, the coefficients for the tangential displacements are given by

$$
\begin{aligned}
a_0 &= u_i \\
a_1 &= -w_{,i} - \chi_{1,i}\frac{Eh^2}{2G'(1-\nu^2)} \\
a_2 &= 0; \\
a_3 &= \chi_{1,i}\frac{E}{6G'(1-\nu^2)} \\
a_4 &= 0 \\
a_5 &= 0
\end{aligned}
\tag{4.26}
$$

Consequently for the "shear" model

$$
u_i^{(k)} = a_0 + a_1 z + a_3 z^3 \tag{4.27}
$$

If we also assume that $G_k' = \infty$ then $\varphi_{qk}(z) = 0 \ \ (q = 1 \ldots 6)$, $\psi_{gk}(z) = 0 \ \ (g = 1 \ldots 8)$ and $a_0 = u_i$, $a_1 = -w_{,i}$. Here the influence of transverse shear is not taken into account and we obtain the classical model, viz.

$$
u_3^{(k)} = w \ ; \quad u_i^{(k)} = u_i - w_{,i} z
$$

It is noted that the higher–order model described above considers the effect of normal reduction caused by the external loads as well as Poisson's ratio effects including the elongation or reduction of the reference surface.

If Poisson's ratio effect is neglected ($\chi_2 = 0$) and the influence of the external loads in the kinematic hypotheses (4.10) and (4.11) is neglected ($\chi_g = 0$, $g = 3 \ldots 8$), then the coefficients of $u_3^{(k)}$ in eqn. (4.24) vanish except for

$$\begin{aligned} b_0 &= w \\ b_2 &= \chi_1 \frac{E\nu'}{2E'(1-\nu)} \end{aligned} \tag{4.28}$$

In this simplified case, the distribution of the displacements through the thickness may be inaccurate. For example, if only one external surface of a symmetrically laminated shell is under load, then the displacements $u_3^{(k)}$ would be symmetric with respect to the middle surface, which is obviously not the true behaviour.

4.5 Special Purpose Symbolic Computation

The objective of the present study is to implement the higher–order theory for the general case, i.e. for a laminate with any number of layers, and in this respect the use of a general purpose system is found to be impractical due to the unimpressive computational efficiency of such systems.

Therefore special purpose symbolic computation software is developed in the C programming language for the computational implementation of the higher–order theory. As discussed in Chapter 2, the exceptionally high computational efficiency of special purpose symbolic computation is a result of its dedication to the analysis of a specific class of functions.

The symbolic computation system developed in this section is specifically designed to implement the higher–order theory presented in Chapter 3. Therefore the routines presented in this section handle symbolic expressions where the symbols may be defined as piecewise integrals, laminae stiffness parameters, constants or symbolic expressions. An algorithm is developed to recursively expand symbols into power series of the z–coordinate.

4.5.1 Symbols

In the application, symbols are referred to by handles which are enumerated constants. The definition of symbols are stored in an array and the symbols' handles are used as indices to this array. In addition, the special handles `SymPos1`, `SymNeg1`, `SymZ2` and `SymZ` are defined for the intrinsic "symbols" 1, -1, z^2 and z.

Symbolic expressions are null–terminated lists of these handles. The special handle `SymPlus` delimits terms to be added, where a list of symbols delimited by `SymPlus` is a term in which the listed symbols are to be multiplied.

The C structure `symt` is defined to store the definition of a symbol. The symbol may be defined as a constant, a stiffness parameter (which is a function of the layer number), a power of z, a symbolic expression, or a layer integal of a symbolic expression. An element `type` in the structure `symt` indicates which of the above classes the symbol is defined as; in particular, `type` is set to `STConst` for a a constant, `STVect` for a stiffness vector, `STZ` for a power of z, `STExpr` for a symbolic expression; and `STInt` for an integral of a symbolic expression.

Storage elements are allocated in the structure `symt` for the information associated with the symbol. There is a real number element `rv`, integer number elements `iv1` and `iv2`, and elements `rp`, `pp`, and `expr` which are pointers to arrays of real numbers, power series and symbolic expressions, respectively. Depending on the class `type` of the symbol, a subset of these storage elements is used. For example, if a symbol is defined as an integral, the limits of integration are stored in number elements and a pointer is set to the symbolic expression to be integrated. In particular, if the symbol is a constant, then the constant is stored in `rv`; if the symbol is a stiffness parameter, the pointer to the list of values is stored in `rp`; if the symbol is a power of z, the (integer) exponent is stored in `iv1`; and if the symbol is an expression or integral of an expression of other symbols, the pointer to that expression is stored in `expr`. In the integral case, the index of the lower limit, given by m in the operators (4.16) and (4.17), is stored in `iv1`, and the upper limit is taken as z. A symbol may also be defined as the type `SymDInt` which is a layer integral with lower and upper limits a_l and a_u, respectively. In this case, the indices l and u are stored in `iv1`, `iv2`, and the symbolic expression to be integrated is referenced by the pointer `expr`.

4.5.2 Power Series

By the nature of the application, symbols defined as any of the above types may be derived via symbolic computation as an n-vector of power series. After this computation, the type of symbol is changed to `STMPow` and the pointer `pp` is set to the derived array of power series. This operation is nontrivial for the types `STExpr` and `STInt`. Symbols of the type `STDInt` may be evaluated to a constant via symbolic computation.

Symbolic computation routines for storing, adding, multiplying and integrating power series in terms of z are defined next. Power series are stored in null–terminated lists of the structure `powt`. This structure contains a single term in a power series, in particular it stores a real coefficient and an integer exponent. The routines `pow_add` and `pow_mult` are defined to add two power series and multiply two power series, respectively. These routines invoke a routine `pow_collect` to collect like terms in the resultant power series. They take as arguments two pointers to the two power series and return a pointer to the resultant power series.

The routine `pow_integrate` is defined to integrate a power series from a lower limit to z, and is declared as

```
powt *pow_integrate(powt *ps, real b, real c)
```

where `ps` is a pointer to the power series to be integrated, `b` is the lower limit of integration and `c` is a constant to be added to the integral. The routine returns a pointer to the resultant power series.

4.5.3 Symbolic Processing

The primary task of the symbolic computation software is to expand symbols into power series of the coordinate z. Symbols which are constants or powers of z are trivially expanded into a power series of one term. Symbols which are stiffness constants may be expanded into a trivial power series for a given layer number k. Layer integrals must be expanded into a power series for each layer number k since we assume that their integrands contain stiffness constants (and, in general, other expressions which are discontinuous at the laminae interfaces) and therefore must be integrated piecewise.

Symbolic expressions are expanded into power series by the routine `sym_expand`, declared as

```
powt *sym_expand(int k, int sym, int *se)
```

where k is the layer number, sym is a symbol handle, and se is the pointer to a symbolic expression. The pointer se is only relevant if sym equals SymExpr or SymInt which are special handles to direct the routine to expand the symbolic expression se, or to expand and then also integrate se.

This routine invokes the routines sym_term and sym_tail to dissect an expression into its first term and into another expression comprising the second through to the last term, respectively. Consider the symbolic expression defined by

```
int se[] = {Sdf1,Sf1,Sf2,SymPlus,Sdf2,SF1,SF2,SymPlus,SymNeg1,0};
```

where those handles other than SymPlus and SymNeg1 are handles for user–defined symbols. The function call sym_term(se) returns a pointer to the symbolic expression

```
{Sdf1,Sf1,Sf2,0}
```

and sym_tail(se) returns a pointer to the symbolic expression

```
{Sdf2,SF1,SF2,SymPlus,SymNeg1,0}
```

Therefore, the routine sym_expand recursively expands symbolic expressions as follows

```
sym_expand(k,SymExpr,se)
   = pow_add(
        pow_mult(
           sym_expand(k,se[0]),
           sym_expand(k,SymExpr,sym_term(se+1))
           ),
        sym_expand(k,SymExpr,sym_tail(se))
        )
```

where, since sym_expand returns a power series, the routines pow_mult and pow_add are invoked to perform the necessary algebra. Clearly, se[0] is the handle of the first symbol in the first term of the expression se (which is a list of handles of symbols delimited by SymPlus), and sym_term(se+1) returns the symbolic expression which comprises the second through to last symbols of the first term. Since the symbols in each term are to be multiplied, pow_mult is invoked to perform that operation, and since the separate terms in the expression are to be summed, pow_add is invoked.

Layer integrals are expanded by expanding their integrands (which are symbolic expressions) into a power series and then integrating the resultant power series using pow_integrate.

The routine `lam_int` integrates symbolic expressions from a lower to an upper limit, and is declared as

```
real lam_int(int m, int k, int *se)
```

where `se` is a pointer to the symbolic expression to be integrated by the operator

$$\sum_{r=m+1}^{k} \int_{a_{r-1}}^{a_r}$$

The routine `sym_expand` expands a layer integral from a_m to z where $a_m \leq a_{k-1} \leq z \leq a_k$, as follows

```
sym_expand(k,SymInt,se)
   = pow_integrate(sym_expand(k,SymExpr,se),a[k-1],
       lam_int(m,k-1,se))
```

where `a[k-1]`$= a_{k-1}$, and m is the index of the lower limit of the layer integral. When $z < a_m$, the algorithm switches to

```
sym_expand(k,SymInt,se)
   = pow_integrate(sym_expand(k,SymExpr,se),a[k],
       lam_int(m,k,se))
```

The routine `sym_derive` invokes `sym_expand` to expand symbols of the types `STExpr` and `STInt` into power series for each $k = 1, 2, \ldots, n$, and stores the n power series associated with the symbol in the element `pp` of the `symt` structure. Symbols which are definite integrals, i.e. symbols of the type `STDInt`, evaluate to constants, so `sym_derive` evaluates these symbols using `lam_int` (which invokes `sym_expand`) and stores the result in the element `rv`.

In an application, if a symbol is contained in many symbolic expressions and is itself a symbolic expression (or an integral of a symbolic expression), the computational efficiency is improved by deriving that symbol using `sym_derive` before processing so that the symbol is expanded once only.

4.5.4 Application to Higher–Order Theory

In the application of the special purpose symbolic computation to the higher–order theory, the symbols relating to the distribution functions given in eqns. (4.18) are defined as follows:

```
f     = Int[a_0,z] E0 dz
fs    = Int[a_0,z] E0 Z dz
Bf    = Int[a_0,a_n] E0 dz
Bf1   = Int[a_0,a_n] E0 Z dz
f1    = fs - f Bf1 1/Bf
f2    = fs 1/Bf - 1
F1    = Int[a_0,z] f1 dz
F2    = Int[a_0,z] f2 dz
Df1   = Int[a_0,a_n] f1 dz
Df2   = Int[a_0,a_n] f2 dz
f4    = F1 Df2 1/Df1 - F2
alp1  = nu0 Z
alp2  = - nu0
alp3  = f4 1/E2
vphi  = f1 1/G2
vphi1 = Int[a_m,z] alp1
vphi2 = Int[a_m,z] alp2
vphi3 = Int[a_m,z] alp3
psi1  = Int[a_m,z] (vphi1 - vphi) dz
psi2  = Int[a_m,z] vphi2 dz
psi3  = Int[a_m,z] vphi3 dz
...
```

where E0, nu0, E2 and G2 are symbols for the stiffness parameters E_{0k}, ν_{0k}, E'_k and G'_k.

The symbols for the integrated stiffnesses are defined as follows:

```
B     = Int[a_0,a_n] A11 dz
B1    = Int[a_0,a_n] A11 psi1 dz
D00   = Int[a_0,a_n] A11 Z^2 dz
D01   = Int[a_0,a_n] A11 psi1 Z dz
D01b  = Int[a_0,a_n] A12 psi1 Z dz
D02   = Int[a_0,a_n] A11 psi2 Z dz
D03   = Int[a_0,a_n] A11 psi3 Z dz
D11   = Int[a_0,a_n] A11 psi1 psi1 dz
D12   = Int[a_0,a_n] A11 psi1 psi2 dz
D13   = Int[a_0,a_n] A11 psi1 psi3 dz
...
```

A listing of the symbolic computation routines described in this section is given in Appendix C. These routines are integrated into a computer program which implements the solution procedure described in Section 4.2. By means of the analytical solution (4.8), the governing differential equations are reduced to linear algebraic equations which are solved using Gauss–Jordan reduction. Then the displacements, strain and stresses may be derived at any point $(x1, x2)$ as a power series of the coordinate z through the thickness of the laminate via the symbolic computation.

4.5.5 Symbolic Results

Consider sandwich shells with isotropic core and surface layers. The stiffness of the surface of layers is $E_1 = 1$, and the core layer is one order of magnitude weaker than the surface layers, i.e. $E_1/E_2 = 10$. The shell is symmetrically laminated and the thicknesses of the surface layers are half the thickness of the core layer. Both core and surface layers have the elastic properties $\nu_k = 0.3$ and $G_k = E_k/2(1 + \nu_k)$. The application employing the special purpose symbolic computation routines derives the distribution functions as

```
...
psi1[1,z] = - 0.0753348 + 0.0892857 z - 0.404762 z^3
psi1[2,z] = 0.691964 z - 0.404762 z^3
psi1[3,z] = 0.691964 z - 0.404762 z^3
psi1[4,z] = 0.0753348 + 0.0892857 z - 0.404762 z^3
psi2[1,z] = - 0.214286 z^2
psi2[2,z] = - 0.214286 z^2
psi2[3,z] = - 0.214286 z^2
psi2[4,z] = - 0.214286 z^2
psi3[1,z] = - 0.00343798 - 0.0529369 z - 0.00224072 z^2
                - 0.125694 z^3 - 0.151515 z^4 + 0.225352 z^5
psi3[2,z] = 0.184659 z^2 - 0.258216 z^3
                - 0.151515 z^4 + 0.225352 z^5
psi3[3,z] = 0.184659 z^2 - 0.258216 z^3
                - 0.151515 z^4 + 0.225352 z^5
psi3[4,z] = - 0.00162239 + 0.0301597 z + 0.0136044 z^2
                + 0.0106701 z^3 - 0.151515 z^4 + 0.225352 z^5
psi4[1,z] = - 0.00162239 - 0.0301597 z + 0.0136044 z^2
                - 0.0106701 z^3 - 0.151515 z^4 - 0.225352 z^5
psi4[2,z] = 0.184659 z^2 + 0.258216 z^3
```

```
                - 0.151515 z^4 - 0.225352 z^5
psi4[3,z] = 0.184659 z^2 + 0.258216 z^3
                - 0.151515 z^4 - 0.225352 z^5
psi4[4,z] = - 0.00343798 + 0.0529369 z - 0.00224072 z^2
                + 0.125694 z^3 - 0.151515 z^4 - 0.225352 z^5
psi5[1,z] = 0.0505062 + 0.74868 z - 0.21831 z^2
                + 0.56338 z^3 - 0.901408 z^5
psi5[2,z] = - 2.5 z^2 + 4.3662 z^3 - 0.901408 z^5
psi5[3,z] = - 2.5 z^2 + 4.3662 z^3 - 0.901408 z^5
psi5[4,z] = 0.0198063 - 0.37632 z - 0.28169 z^2
                + 0.56338 z^3 - 0.901408 z^5
psi6[1,z] = - 0.0198063 - 0.37632 z + 0.28169 z^2
                + 0.56338 z^3 - 0.901408 z^5
psi6[2,z] = 2.5 z^2 + 4.3662 z^3 - 0.901408 z^5
psi6[3,z] = 2.5 z^2 + 4.3662 z^3 - 0.901408 z^5
psi6[4,z] = - 0.0505062 + 0.74868 z + 0.21831 z^2
                + 0.56338 z^3 - 0.901408 z^5
```

where the first argument is the layer number. The core layer is divided into two sublayers in order to introduce an artificial interface at the reference surface, and therefore the expressions for each distribution function in the layers $k = 2, 3$ are identical.

The computational efficiency of the special purpose routines is found to be more than two orders of magnitude higher than that of the *Mathematica* implementation described in Section 4.3. Moreover, it is found that the *Mathematica* implementation is impractical for a laminate with more than three layers whereas the special purpose symbolic computation implements the higher–order theory for the general case.

4.6 Numerical Results

Using the method of solution outlined in Section 4.2, numerical results are obtained for square plates and shells which are hinged–supported at their edges and subject to a normal sinusoidal load of magnitude q_0.

4.6.1 Isotropic Plates

Table 4.1 gives the deflection and normal stress at the centre of a square isotropic plate with thickness ratio of $a/h = 2$ and Poisson's ratio $\nu = 0.3$. Results are given for the full model of the higher–order theory and the shear model which neglects normal deformation. These results are compared to the exact three–dimensional solution given in Ref. [52] and to the classical theory which neglects both transverse shear and normal deformation. It is observed that the full model of the higher–order theory is in good agreement with the exact solution whereas the shear and classical models are grossly inaccurate.

Table 4.1: Deflection and stress at the centre of an isotropic plate

	$u_3 E/q_0 h$						
		Higher-order theory					
z/h	3–D Solution	Full model	Δ, %	Shear model	Δ, %	Classical theory	Δ, %
-0.5	1.215	1.221	0.4		-12		-63
0.0	0.967	0.964	-0.3	1.070	11	0.448	-54
0.5	0.772	0.784	1.6		39		-42
	$\sigma_{11}/q0$						
-0.5	-1.205	-1.186	-2	-0.973	-19	-0.790	-34
0.5	0.832	0.793	-5	0.973	17	-0.790	-5

Figure 4.1 on Page 105 shows the deflection and normal stress distributions through the thickness at the centre of a square isotropic plate subject to a sinusoidal load. In the case $a/h = 3$, the maximum deflection occurs at $z/h = -0.43$ and not at the top surface, whereas in the case $a/h = 2$ the maximum deflection occurs at the top surface where the loading is applied. This phenomenon is predicted only by three–dimensional and higher–order theories which consider the effect of normal deformation.

4.6.2 Transversely Isotropic Plates

Table 4.2 gives the deflection and normal stress at the centre of square transversely isotropic plates with thickness ratio $a/h = 5$. In the case of the shear and classical models, the modulus E' is equated to ∞ and is therefore irrelevant. It is observed that the effect of normal deformation is substantial in transversely isotropic plates and this effect increases with E/E'. The shear and classical models which neglect normal deformation are inaccurate.

Table 4.2: Deflections and normal stresses at the centre of square transversely isotropic plates ($x_1 = x_2 = a/2$) with $a/h = 5$, $\nu' = 0$ and $G' = G$.

z/h	$u_3 E/q_0 h$				
	Higher-order theory				
	Full model			Shear model	Classical model
	$E/E' = 1$	$E/E' = 50$	$E/E' = 100$	$E/E' = 1 \ldots 100$	
-0.5	21.83	40.01	58.56		
-0.25	21.59	28.19	34.93		
0	21.42	19.70	17.19	21.46	17.52
0.25	21.34	15.69	9.93		
0.5	21.33	15.01	8.56		
	$\sigma_{11}/q0$				
-0.5	-5.15	-6.60	-8.08	-5.12	-4.94
-0.25	-2.39	-2.10	-1.82	-2.39	-2.47
0	-0.01	0.59	1.17	0.00	0.00
0.25	2.38	2.40	2.41	2.39	2.47
0.5	5.11	4.25	3.38	5.12	4.94

Table 4.3 compares the deflection behaviour of isotropic and transversely isotropic square plates of various thickness ratios. As the thickness ratio of the plate increases, the effect of normal deformation decreases and the shear model (which predicts a uniform deflection through the thickness) becomes more accurate. In the transversely isotropic case ($E/E' = 10$) where the plate is 10 times weaker in the transverse direction, the effect of normal deformation is more pronounced, as expected.

Figure 4.2 shows the deflection and normal stress distributions through the thickness at the centre of a square transversely isotropic plate with $a/h = 3$, $\nu = 0.3$ and

Table 4.3: Deflection behaviour at the centre of isotropic and transversely isotropic plates ($x_1 = x_2 = a/2$) with $\nu' = \nu E'/E$ and $G' = E'/2(1+\nu')$.

	$\bar{w} = u_3 E/q_0 h$							
	$E/E' = 1$				$E/E' = 10$			
a/h	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$	$\bar{w}_{\text{shear}}$	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$	$\bar{w}_{\text{shear}}$
2	1.22	0.96	0.78	1.07	7.58	4.31	3.97	4.87
3	3.60	3.49	3.15	3.68	15.97	12.35	11.36	12.91
4	9.28	9.38	8.83	9.69	29.44	25.91	24.60	26.57
5	20.62	20.98	20.16	21.46	50.67	47.37	45.76	48.17
10	291.66	294.25	291.20	296.06	403.81	402.72	398.86	404.83

$\nu' = \nu E'/E$. In the case $E/E' = 100$, the minimum deflection occurs at $z/h = 0.21$ and not at the bottom surface as in the case $E/E' = 10$. It is noted that this effect is not observed if $\nu' = \nu$.

4.6.3 Heterogeneous Plates

Consider a plate with a modulus of elasticity which is a continuous function $E(z) = E_0 e^{-z}$ where E_0 is the modulus of elasticity of the mid-surface $z = 0$ of the plate. This plate is modelled by approximating the continuous function using a piecewise linear function through the thickness of the plate. Table 4.4 shows the deflection behaviour at the centre of the partially heterogeneous plate with distinct sublayers of constant moduli of elasticity. The modulus for elasticity for the k-th layer is taken as $E_k = E_0 e^{-z_k}$ where $z_k = \frac{1}{2}(a_k + a_{k-1})$. Deflections are given for the top, middle and bottom surfaces of the plate. It is observed that as the number of sublayers increases, the deflection converges. This problem demonstrates the ability of the software to analyse laminates with a large number of layers.

4.6.4 Sandwich Plates

Table 4.5 gives the deflections and compressive normal stresses at the centre of the top surface of various sandwich plates. The core and surface layers are isotropic with $\nu = 0.3$ and $G = E/2(1+\nu)$. The plates are symmetrically laminated and the thicknesses t_1, t_3 of the surface layers are half the thickness t_2 of the core layer.

Table 4.4: Deflection at the centre of a partially heterogeneous plate ($x_1 = x_2 = a/2$) with $E_k = E_0 e^{-z_k}$, $\nu_k = 0.3$ and $G_k = E_k/2(1+\nu_k)$.

Number of sublayers	$\bar{w} = u_3 E_0/q_0 h$		
	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$
8	3.6425	3.5514	3.1291
12	3.6408	3.5492	3.1269
16	3.6401	3.5485	3.1262
20	3.6400	3.5481	3.1258
24	3.6397	3.5479	3.1256
28	3.6396	3.5478	3.1255
32	3.6395	3.5477	3.1254
40	3.6394	3.5476	3.1253
48	3.6394	3.5476	3.1253

Young's moduli for the surface and core layers are E_1 and E_2, respectively. Therefore in the case $E_1/E_2 = 10$ the core layer is one order of magnitude weaker than the surface layers, and in the case $E_1/E_2 = 100$ the core is two orders of magnitude weaker. Three models are considered: the full model of the higher–order theory, the shear model, and a combined model where the full model is used for the core layer and the classical model is used for the surface layers. The results obtained using these three models are compared to those given by Brukker in Ref. [53] where the core layer is modelled using an exact three–dimensional elasticity solution and the surface layers are modelled using the classical hypotheses. Results are compared for the range $a/h = 3, \ldots, 10$ where for $t_2/t_1 \geq 2$ the thickness ratio of surface layers is greater than 10 and therefore Brukker's solution is considered to be exact. Brukker's model is identical to the *combined* model except that the core layer is modelled using a three–dimensional elasticity solution whereas in the combined model, the higher–order theory is used to model the core layer.

In Table 4.5 it is observed that the discrepancies between the combined model and Brukker's solution are less than 1%. All three models predict a deflection within 1% of Brukker's solution in the case of thin plates ($a/h = 10$) with $E_1/E_2 = 10$. Moreover, the deflection given by the shear model for moderately thick plates ($a/h = 4,\ 5$) with $E_1/E_2 = 10$ is within 2% of Brukker's solution.

The most important observation from Table 4.5 is that Brukker's solution is closer to both the combined and shear models than it is to the full model of the higher–

Table 4.5: Deflections and stresses at the centre of the top surface of square symmetrically laminated sandwich plates ($x_1 = x_2 = a/2$) with isotropic layers and $t_1 = t_3 = t_2/2$.

E_1/E_2	a/h	u_3E_1/q_0h						
		Brukker's Solution [53]	Higher–order theory					
			Full model	Δ, %	Combined model	Δ, %	Shear model	Δ, %
10	10	425.40	425.36	-0.0	425.32	-0.0	429.14	0.9
	5	46.96	47.22	0.6	46.94	-0.0	47.08	0.3
	4	25.47	25.76	1.1	25.45	-0.1	25.14	-1.3
	3	12.33	12.64	2.5	12.32	-0.1	11.65	-5.5
100	10	1298.0	1277.9	-1.5	1297.9	-0.0	1292.3	-0.4
	5	211.2	202.3	-4.2	211.5	0.1	202.1	-4.3
	4	115.6	110.6	-4.3	116.0	0.3	106.3	-8.0
	3	52.7	52.1	-1.1	53.1	0.8	—	—
		σ_{11}/q_0						
10	10	23.88	23.73	-0.6	23.89	0.0	23.99	0.5
	5	7.19	7.03	-2.2	7.19	0.0	7.23	0.6
	4	5.20	5.04	-3.1	5.19	-0.2	5.18	-0.4
	3	3.66	3.53	-3.6	3.65	-0.3	3.54	-0.3
100	10	37.78	35.07	-5.4	37.78	0.0	37.74	-1.8
	5	17.59	15.26	-13.2	17.60	0.1	17.02	-3.2
	4	14.14	12.16	-14.0	14.17	0.2	13.19	-6.7
	3	10.87	9.61	-11.6	10.92	0.5	—	—

order theory. This indicates that normal deformation should be modelled in both the core and surface layers in order to obtain accurate results.

Table 4.6 gives the deflection behaviour of sandwich plates of various thickness ratios. It is observed that as the thickness ratio of the plate increases, the effect of normal deformation decreases and the shear model becomes more accurate. The phenomenon of negative deflection at the bottom surface is observed for a plate with $a/h = 2$ and $E_1/E_2 = 100$. This phenomenon is caused by Poisson's effect and is observed using exact three–dimensional elasticity solutions.

Figure 4.3 gives the deflection and normal stress distributions through the thickness at the centre of a sandwich plate with $a/h = 3$. Since the core layer is weaker

Table 4.6: Deflection behaviour at the centre of square symmetrically laminated sandwich plates ($x_1 = x_2 = a/2$) with isotropic layers and $t_1 = t_3 = t_2/2$.

	$\bar{w} = u_3 E/q_0 h$							
	$E_1/E_2 = 10$				$E_1/E_2 = 100$			
a/h	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$	$\bar{w}_{\text{shear}}$	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$	$\bar{w}_{\text{shear}}$
2	5.26	3.53	2.64	4.02	22.23	5.88	-2.26	11.09
3	12.64	11.05	9.97	11.65	52.07	35.52	27.02	43.71
4	25.76	24.40	23.08	25.14	110.58	94.16	85.45	106.29
5	47.22	46.17	44.55	47.08	202.26	186.08	177.11	202.12

than the surface layers ($E_1/E_2 > 1$), it absorbs most of the normal deformation. As expected, the normal deformation of the core layer increases with E_1/E_2. The stress is substantially reduced in the core layer as compared to the stresses in the surface layers. It is observed that the stress distribution through the thickness of the surface layers becomes more symmetrical when $E_1/E_2 = 100$.

4.6.5 Isotropic Shells

Table 4.7 shows the deflections and normal stresses at the centre of a square isotropic shell with $a/h = 5$ and radius of curvature R. The shell has double curvature $k_{11} = k_{22} = 1/R$. Numerical results obtained using the higher–order theory are compared to those of the classical shell theory. It is observed that as the curvature of the shell increases, the influence of transverse shear and normal deformation on the deflection at the mid-surface tends to decrease and on the normal stress at the top surface tends to increase.

Figure 4.4 shows the relative deflection w/w_{class} at the centre of a square isotropic shell versus a/h where w_{class} is the deflection given by classical shell theory. The shell is doubly curved with a curvatures $k_{11} = k_{22} = 1/R$ and radius of curvature $R = a$. The curves show the deflection w_{shear} predicted by the shear–deformable model and the deflections given by the higher–order theory at the top, bottom and mid-surface of the shell. (Both the classical and shear–deformable theory neglect normal deformation and therefore predict a constant deflection through the thickness.) It is observed that the effect of normal deformation is more pronounced at the top surface where the load acts. Over the given range $a/h = 5 \ldots 10$, the deflection w_{shear} is close to the deflections at the centre and bottom of the shell, but deviates more

Table 4.7: Nondimensionalized deflections and normal stresses at the centre of square doubly curved isotropic shells with $a/h = 5$ where $\bar{w} = u_3 E/q_0 h$ and $\bar{\sigma} = \sigma_{11}/q_0$.

	Higher–order theory					Classical shell theory		
a/R	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$	$\bar{\sigma}_{\text{top}}$	$\bar{\sigma}_{\text{bot}}$	$\bar{w}$	$\bar{\sigma}_{\text{top}}$	$\bar{\sigma}_{\text{bot}}$
0.00	21.83	21.42	21.33	-5.15	5.11	17.52	-4.94	4.94
0.25	20.77	20.37	20.27	-4.37	5.36	16.78	-4.31	5.15
0.50	18.10	17.70	17.60	-3.33	5.09	14.91	-3.46	4.95
0.75	14.92	14.51	14.42	-2.34	4.52	12.56	-2.60	4.49
1.00	12.00	11.59	11.50	-1.55	3.89	10.30	-1.87	3.93
1.25	9.61	9.21	9.11	-0.96	3.30	8.36	-1.31	3.40
1.50	7.76	7.36	7.26	-0.54	2.81	6.80	-0.90	2.94
1.75	6.35	5.94	5.85	-0.24	2.40	5.57	-0.60	2.54
2.00	5.27	4.86	4.77	-0.03	2.06	4.61	-0.38	2.22

substantially from the deflection at the top surface (since normal deformation is neglected by the shear model). As a/h increases, i.e. as the shell becomes thinner, the effect of normal deformation is reduced and the deflections given by higher–order and shear–deformable theory approach the deflection predicted by the classical theory. At $a/h = 10$, the discrepancies of the classical theory are less than 5%, and the shear–deformable theory is accurate.

4.6.6 Laminated Shells

The effect of curvature on a square doubly curved laminated shell is considered. The shell is constructed using two identical three-layered sandwich shells separated by a cellular filler material, and therefore the shell has seven distinct layers. The three–layered surface shells have metal bearing layers and a glass/epoxy composite core material. The filler layer between the two sandwich shells is made of polystyrene. The metal layers have thickness $t_1 = 5$mm and elastic properties $E_1 = 70$GPa, $\nu_1 = 0.3$ and $G_1 = E_1/2(1+\nu_1)$. The two composite layers have a thickness $t_2 = 15$mm and elastic properties $E_2 = 26$GPa, $E_2' = 8.4$GPa, $G_2 = 11.5$GPa, $G_2' = 3$GPa and $\nu_2 = 0.13$. The polystyrene filler layer has thickness $t_4 = 150$mm and elastic properties $E_4 = 19.6$MPa, $\nu_4 = 0.4$ and $G_4 = E_4/2(1+\nu_4)$. The shell has thickness ratio $a/h = 5$ and is subjected to a sinusoidal load of magnitude $q_0 = 1$kPa on its top surface. The curvatures of the shell are $k_{11} = k_{22} = 1/R$.

Figure 4.5 and 4.6 illustrate the influence of curvature on the deflection and normal stress at the centre of the shell. The deflection and stresses are given relative to the case of zero curvature. Figure 4.5 gives the curves of the relative deflection as predicted by the shear–deformable model (in which case the deflection is uniform through the thickness) and by the higher–order theory at the top, bottom and mid-surface of the shell. Figure 4.6 shows the relative normal stresses at the top and bottom surfaces of the shell as predicted by the higher–order theory and by the shear–deformable theory.

It is observed in Figure 4.5 that as the curvature increases, the relative deflections decrease. Moreover, the relative deflection at the top (loaded) surface is least effected by the curvature, although the actual deflection at the top surface is greater than the deflection at the bottom surface and mid-surface (due to the effect of normal deformation) as is evident from Table 4.8.

Table 4.8: Nondimensionalized deflections and normal stresses at the centre of a square doubly curved laminated shell with $a/h = 5$ where $\bar{w} = 10^6 w/a$ and $\bar{\sigma} = \sigma_{11}/10^3 q0$.

	Higher–order theory					Shear–deformable theory		
a/R	$\bar{w}_{\text{top}}$	$\bar{w}_{\text{mid}}$	$\bar{w}_{\text{bot}}$	$\bar{\sigma}_{\text{top}}$	$\bar{\sigma}_{\text{bot}}$	$\bar{w}$	$\bar{\sigma}_{\text{top}}$	$\bar{\sigma}_{\text{bot}}$
0.00	12.59	9.85	8.75	-0.159	0.111	10.53	-0.133	0.133
0.02	12.49	9.76	8.66	-0.148	0.117	10.43	-0.124	0.140
0.04	12.21	9.49	8.38	-0.136	0.119	10.15	-0.113	0.144
0.06	11.78	9.05	7.95	-0.122	0.119	9.72	-0.101	0.145
0.08	11.23	8.50	7.40	-0.108	0.117	9.17	-0.088	0.144
0.10	10.61	7.88	6.78	-0.094	0.112	8.55	-0.076	0.141
0.12	9.95	7.22	6.12	-0.080	0.106	7.90	-0.064	0.136
0.14	9.29	6.56	5.46	-0.068	0.099	7.24	-0.053	0.130
0.16	8.65	5.92	4.82	-0.056	0.091	6.61	-0.044	0.124
0.18	8.05	5.32	4.22	-0.046	0.083	6.02	-0.035	0.117
0.20	7.49	4.77	3.67	-0.037	0.075	5.47	-0.028	0.111

In Figure 4.6, both the shear–deformable and higher–order theory predict that the relative stress at the bottom surface of the shell decreases with increasing curvature whereas the relative stress at the at the top surface of the shell increases initially and then decreases. Moreover, the relative stress at the bottom surface as given by the higher–order theory is more effected by curvature than that given by the

shear–deformable theory, whereas the relative stress at the top surface given by the two theories are similarly effected by curvature.

Isotropic Plate

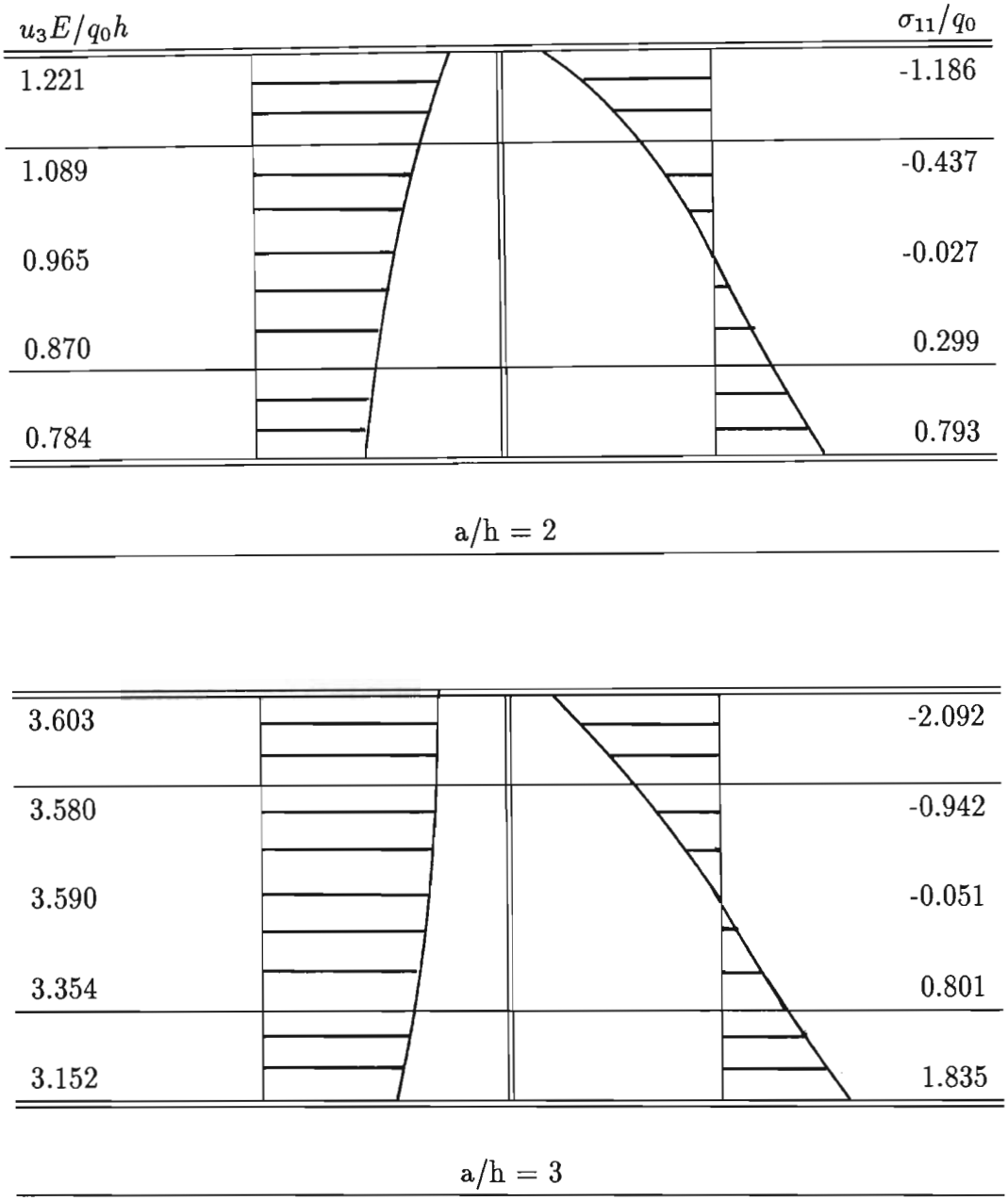


Figure 4.1. Deflection and normal stress distributions at the centre of an isotropic plate.

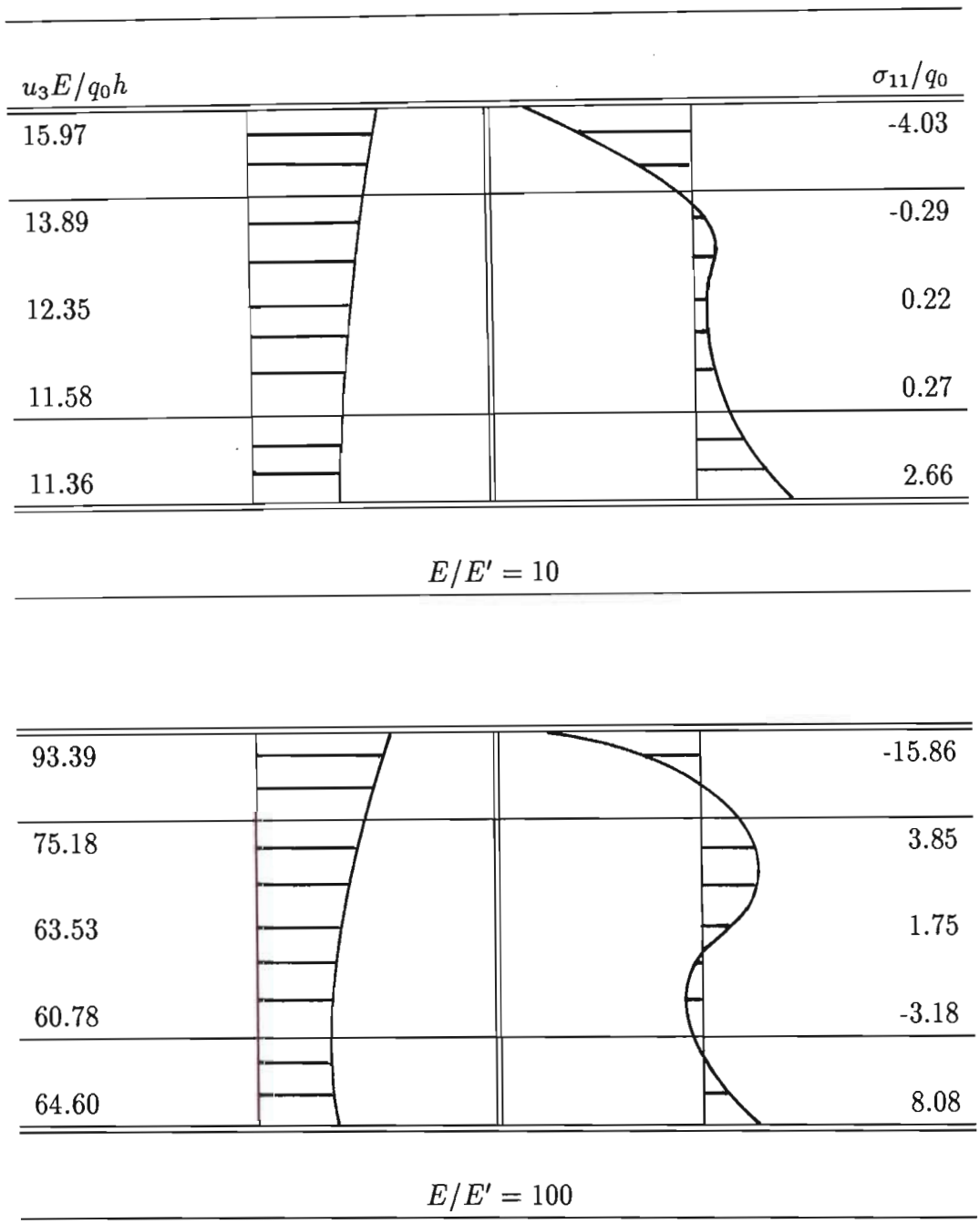


Figure 4.2. Deflection and normal stress distributions at the centre of a transversely isotropic plate with $a/h = 3$.

Sandwich Plate

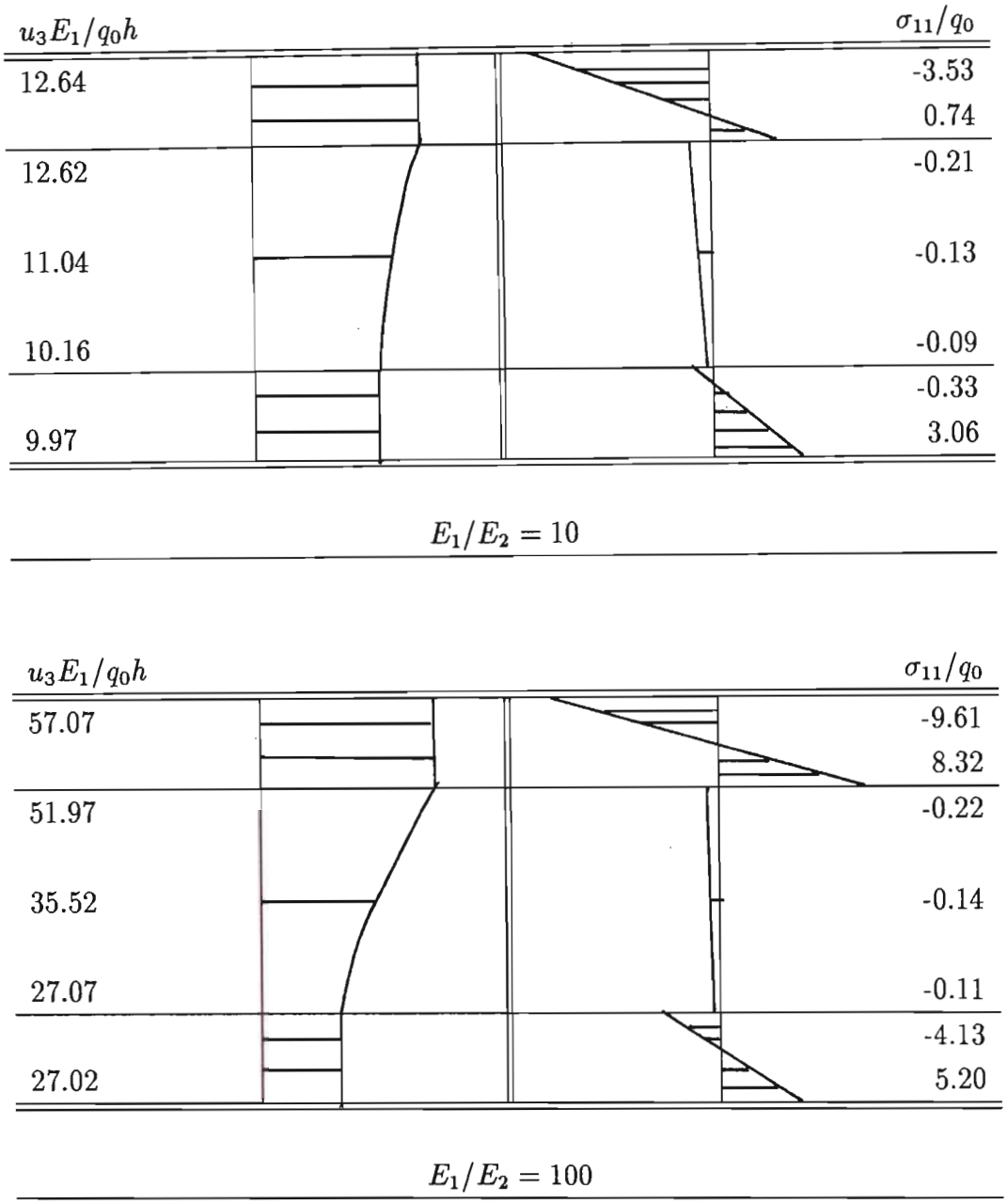


Figure 4.3. Deflection and normal stress distributions at the centre of a sandwich plate with $a/h = 3$, $t_2/t_1 = 2$ and isotropic core and surface layers.

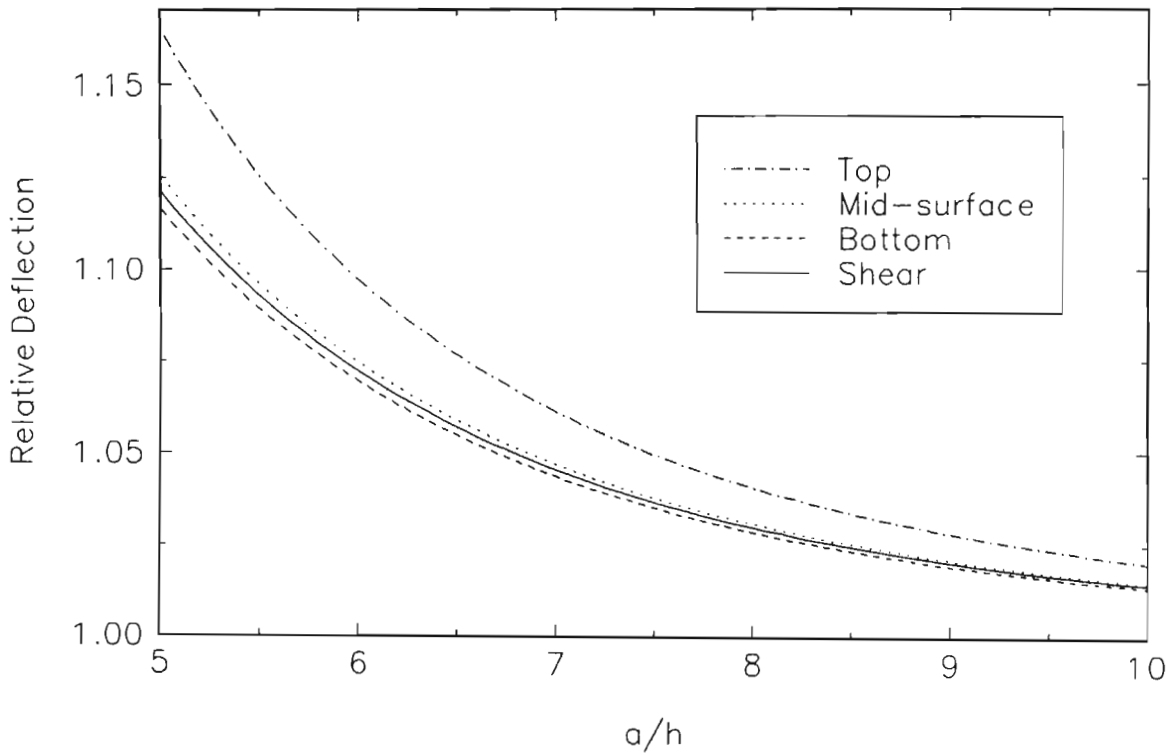


Figure 4.4. Relative deflection of a doubly curved isotropic shell.

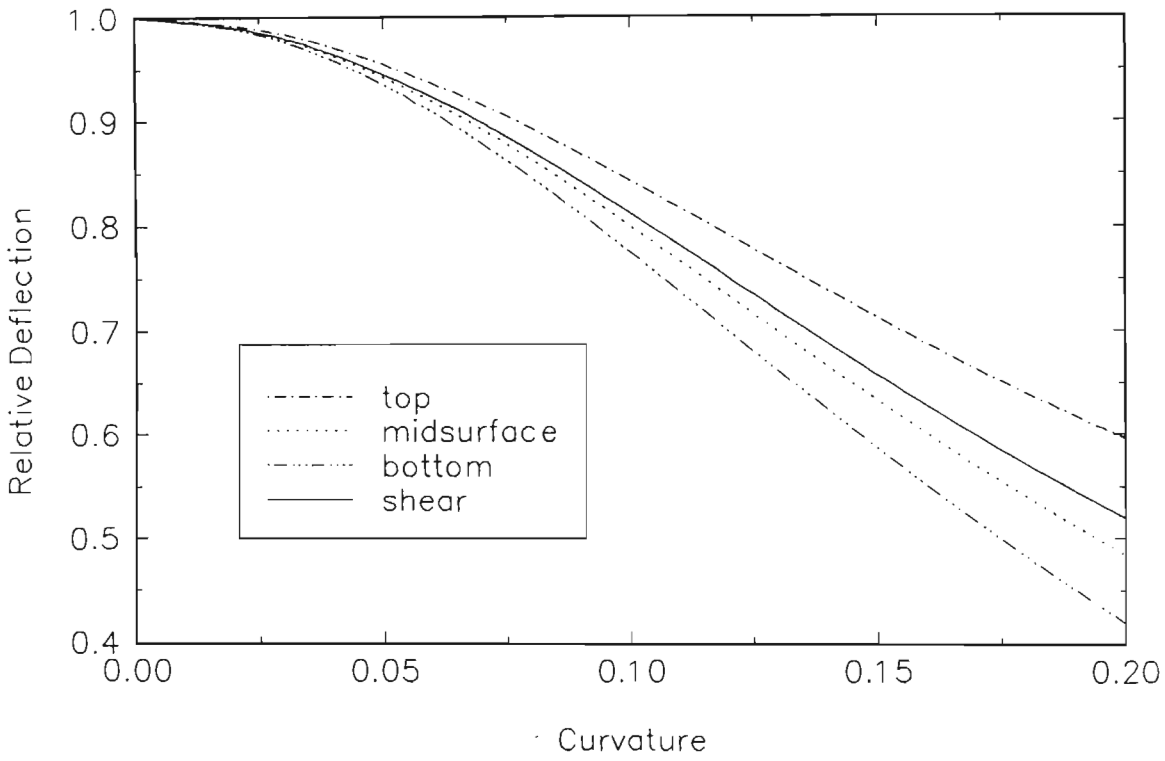


Figure 4.5. Relative deflection of a doubly curved laminated shell.

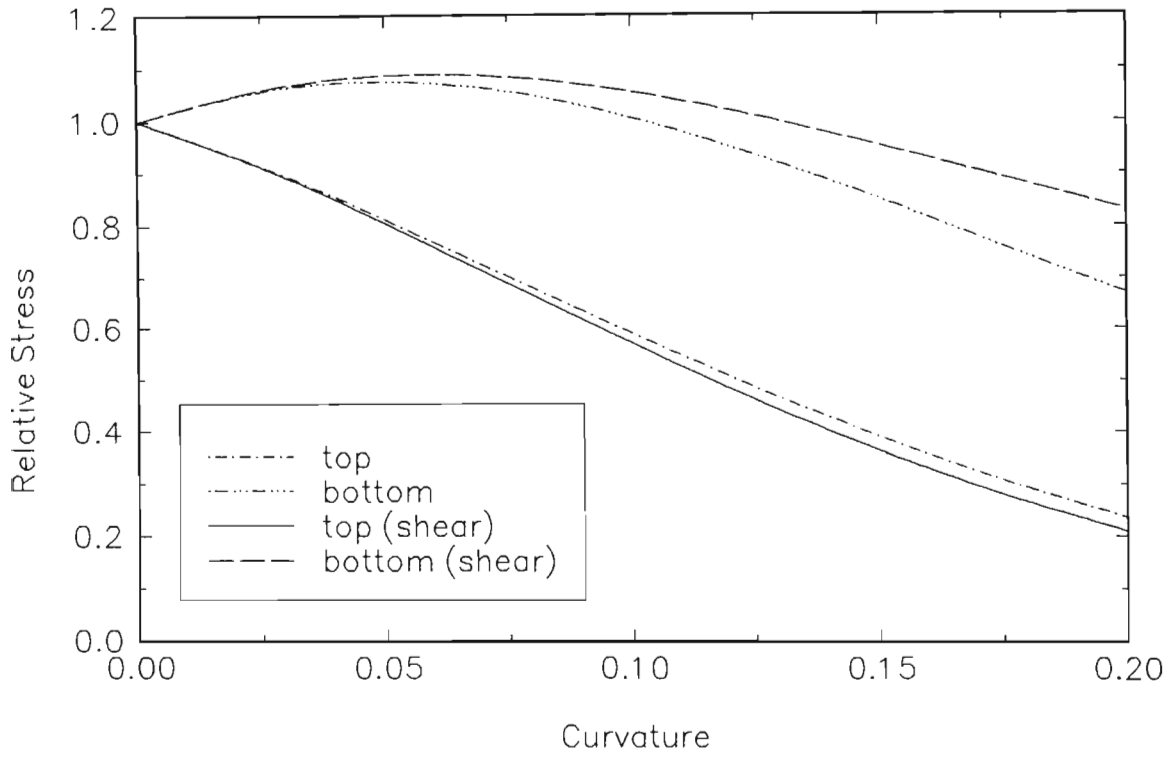


Figure 4.6. Relative stress of a doubly curved laminated shell.

4.7 Conclusions

The comprehensive higher–order theory presented in Chapter 3 is implemented for computational studies. In the general case, the distribution functions and integrated stiffnesses cannot be derived in a form suitable for direct numerical implementation, and the calculation of these functions using a numerical method detracts from the increased accuracy offered by the higher–order theory. Therefore symbolic computation is employed to derive the distribution functions, calculate the integrated stiffness constants, solve the system of governing differential equations analytically, and finally to evaluate the stress/strain state of a given laminate.

The theory is implemented using the *Mathematica* symbolic computation system, and this application is used to derive analytical results and obtain numerical results. However, in order to improve the computational efficiency of the analysis, special purpose symbolic computation routines are developed in the C programming language as an alternative to using a general purpose symbolic computation system such as *Mathematica*. The routines handle symbolic expressions where the symbols may be defined as piecewise integrals through thickness of a laminate, laminae stiffness parameters, constants or symbolic expressions. Symbols and symbolic expressions are expanded into power series using a recursion technique. It is found that the special purpose symbolic computation is more than two orders of magnitude more efficient than *Mathematica* owing to its dedication to the requirements of the specific problem.

The numerical results obtained for thick homogeneous and heterogeneous plates are compared to those in the literature in order to validate the higher–order theory. It is found that for an isotropic plate with thickness ratio $a/h = 2$, the higher–order theory predicts the deflection distribution to within 2%, and the normal stresses to within 5%, of the exact three–dimensional elasticity solution, whereas the shear–deformable model (which neglects the effect of normal deformation) and the classical model are grossly inaccurate.

The more weaker plates are in the transverse direction, the more pronounced is the effect of normal deformation, and the inaccuracy of the shear–deformable model increases. However, it is observed that as the ratio a/h increases, i.e. the structure becomes thinner, the effect of normal deformation is reduced, and for plates with $a/h \geq 10$, i.e. thin plates, the shear–deformable theory may be considered to be accurate.

Numerical results are given for doubly curved isotropic shells with thickness ratio $a/h = 5$. It is observed that as the curvature of the shell increases, the influence of transverse shear and normal deformation on the deflection at the mid-surface tends to decrease and on the normal stress at the loaded surface tends to increase.

Numerical results for sandwich plates where the higher–order theory is used to model the core layer and the classical theory is used to model the surface layers are compared to those given in the literature where the core layer is modelled as a three–dimensional elastic body. It is found that the discrepancies are at most 0.8% over the range $a/h = 3, \ldots, 10$ for the core layer one or two orders of magnitude weaker than the surface layers.

The higher–order theory predicts phenomena which can only be observed using three–dimensional elasticity solutions or a theory which considers normal deformation. In particular, in the case of an isotropic plate with $a/h = 3$, the maximum deflection occurs near the top surface where the loading is applied, whereas in the case $a/h = 2$ the maximum deflection occurs at the top surface. Moreover, for a sandwich plate with $a/h = 2$, negative deflection is observed at the bottom surface when the core layer is two orders of magnitude weaker than the surface layers, but is not observed when the core layer is only one order weaker than the surface layers. It is noted that this phenomenon is caused by Poisson's effect.

The numerical results obtained indicate that the new higher–order theory is accurate for thick structures, whereas, in general, the shear–deformable theory and the classical theory are inaccurate. Therefore in the analysis of thick structures, not only transverse shear but also normal deformation should be taken into account.

Chapter 5

Optimization of Thick Sandwich Plates based on Higher–Order Theory

5.1 Introduction

The objective of the present chapter is the optimization of thick sandwich structures on the basis of the higher–order theory presented in Chapter 3.

In the case of sandwich plates with significantly different mechanical properties of the surface and core layers, normal deformation needs to be accounted for to determine the deflection profile through the thickness in an accurate manner. The core layer absorbs some of the deformation and this leads to vastly different deflections of the top and bottom surfaces of the plate with the amount of core deformation depending on the relative stiffnesses and thicknesses of the surface and core layers. Moreover the stress distribution through the thickness of thick laminated plates is no longer symmetrical even for symmetrical structures. This is due to the fact that the load is applied on the top surface and it is physically clear that due to the non–symmetry of loading, the resulting stress distribution cannot be symmetrical as predicted by theories which fail to take normal deformation into account.

It is known that the classical theory yields inaccurate results for thick composite structures as a result of neglecting transverse shear and normal deformation. Since the accurate analysis of the stress and strain behaviour of thick laminated composite plates is essential for the optimal design of such structures, the classical theory

cannot be used for this purpose. Moreover, it is shown in this chapter that shear–deformable theories (which neglect normal deformation) are also inadequate for the design optimization of thick plates. Clearly a three–dimensional elasticity solution would provide an accurate analysis. However, such solutions are computationally demanding. Therefore, owing to the importance of both accuracy and computational efficiency in design optimization studies, the higher–order theory developed in Chapter 3 is better suited to such studies. Certainly the accuracy of this theory was demonstrated in the previous chapter, and, as discussed in Sections 3.1 and 3.2, the computational demands of single–layer theories such as this higher–order theory, are less than those of three–dimensional and discrete–layer higher–order theories.

However, the computational implementation of the proposed higher–order theory poses special computational problems due to the need to evaluate multiple piecewise integrals through the thickness of the laminate to compute stiffnesses. Moreover, due to the iterative nature of optimization solutions, these calculations need to be performed using computationally efficient algorithms. These difficulties are overcome by developing special purpose symbolic computation routines to perform the necessary calculations. The routines developed in this chapter bypass some of the symbolic processing performed by the more flexible routines developed in Section 4.5 in order to improve the efficiency of the symbolic computations.

Three optimal design problems for thick laminated sandwich plates are considered. The first problem involves the minimum weight design of a sandwich plate subject to a constraint on the deflection of the bottom surface. The design variables are chosen as the thickness of the core layer and the fibre content of the surface layers which are made of a transversely isotropic composite material. Numerical results are given for a sandwich plate with a steel honeycomb core layer. The relationship between the laminate thickness, fibre content and deflection constraint is established.

The second problem involves the minimum deflection design of a sandwich plate. In this problem the relative thickness of the surface and core layers is chosen as the design variable. The optimal design for minimum deflection is based on the observation that the deflection of the bottom surface decreases as the core thickness becomes smaller, but increases again once this thickness drops below a certain level. Moreover, the bottom surface may undergo negative deflection for certain combinations of material properties. These phenomena are peculiar to thick structures and can only be analysed using three-dimensional elasticity solutions or a higher–order theory which includes normal deformation. The effect of the relative stiffness of the surface and core layers on the optimal thickness of the surface layers is investigated.

The third problem is a minimum stress problem which involves the computation of the relative thicknesses of the layers such that the resulting lamination will reflect the stress pattern in a more realistic fashion and thereby will minimize the maximum stress. In this regard, the present study departs from conventional designs which automatically assume a symmetrical lamination. In the case of thick structures, such conventional designs cease to be optimal as shown in this chapter.

5.2 Literature survey

A number of refined theories were developed for sandwich plates to include the effect of shear deformation in the surface and core layers [67, 68, 69]. However optimum designs of sandwich plates and shells were mostly based on classical sandwich theory. Various optimization studies for sandwich structures include minimum weight beams [70], plates under compressive loads [71, 72] and bending loads [73], and acoustic sandwich panels [74]. Design of sandwich shells with fibre composite surface layers was given in Refs. [75] and [76]. Sandwich plates under uncertain bending loads were designed in Ref. [77].

The design of thick sandwich structures does not seem to be studied using a higher–order theory which includes normal as well as shear deformation. In fact previous studies on the optimal design of thick laminated structures seem to be based on shear–deformable theories only. In this regard, studies include maximum frequency design [78, 79], maximum buckling load design [79, 80], and maximum stiffness design [81].

5.3 Software

In this section, software dedicated to the implementation of the higher–order theory for design optimization is developed in the C programming language.

This software processes power series and double trigonometric series using simple programming techniques. The routines which handle power series are used for the calculation of the distribution functions and integrated stiffness constants of the higher–order theory, and the routines which handle trigonometric series are used for the calculation of the displacements (3.37) and (3.38), strains (3.40) and stresses (3.42) using the solution (4.8) of the system (4.6).

The special purpose symbolic computation routines developed in this chapter for the derivation of the distribution functions and calculation of the integrated stiffnesses differ from the more flexible but less efficient routines developed in Section 4.5. The routines developed in the present study mimic the symbolic computations of those of Section 4.5 but without the recursive expansion of symbolic expressions. Rather, the various operations to derive symbols (such as the distribution functions) as power series or to evaluate symbols (such as the integrated stiffness) are explicitly initiated via calls to routines which perform those operations. For example, if the next step of the procedure is to sum two particular symbols, the relevant *operator* routine, in this case a routine which handles summation, is called with arguments which reference the two operands. In Section 4.5, a symbolic expression could be defined and the operator routines would be initiated automatically with the appropriate arguments by the routine which expands a symbolic expression into a power series.

5.3.1 Symbols

In the application, the distribution functions $\psi_{gk}, \varphi_{qk}, \beta_{tk}$ and α_{qk} in eqns. (3.37), (3.38) and (3.40) become symbols which are referred to by handles. For example, the function ψ_{1k} is referenced by the handle `Fpsi1`, an enumerated constant. These symbols are n vectors of power series with a maximum of p_c coefficients. The coefficients of these symbols are stored in a multidimensional array and the handles to the symbols are used as indices to the first dimension of the array. The index to the second dimension is the layer number k, and the third dimension contains the p_c coefficients of the power series of the relevant symbol for the k-th layer. In the application, the value of a symbol is returned by the C function `zfn(sym,k,z)` where `sym` is the handle of the symbol to be evaluated at z where $a_{k-1} \leq z < a_k$.

5.3.2 Distribution Functions

First a set of routines is developed for the derivation of the distribution functions. This set includes routines for algebraic operations involving power series. Also, the routine `calc_layint` performs the layer integral operations (4.16) and (4.17) using the observation that

$$\int_a^z \sum_{p=0}^{n_p-1} c_p z^p dz = \sum_{p=0}^{n_p} d_p z^p \tag{5.1}$$

where

$$d_{p+1} = c_p/(p+1) \quad \text{and} \quad d_0 = -\sum_{p=0}^{n_p-1} c_p a^{p+1}/(p+1) \tag{5.2}$$

The distribution functions of the higher–order theory have a hierarchal co–dependency as is evident from eqns. (4.18). Therefore the order in which they are derived by the symbolic computation routines is dictated by their dependency on other distribution functions.

5.3.3 Integrated Stiffnesses

The higher–order theory defines a large set of integrated stiffness constants which appear in the system of governing differential equations. It is observed that these integrals may be generalized to the form

$$\int_{a_0}^{a_n} A_k \gamma_{1k}(z) \gamma_{2k}(z) z^p dz \tag{5.3}$$

where $A_k = A_{11k}, A_{12k}, A_{13k}, A_{33k}$ is a stiffness parameter of the k-th layer, $\gamma_{1k}(z)$ and $\gamma_{2k}(z)$ are either distribution functions or are equal to unity, and $p = 0, 1, 2$. In the application, a C function `calc_lamint(vm,fn0,fn1,p)` calculates any integrated stiffness, where `vm[]` is an n–vector of stiffness constants, and `fn0` and `fn1` are handles of distribution functions. Some examples of integrated stiffness constants and the corresponding C function calls which evaluate them are given in Table 5.1 where `FNull` is an intrinsic handle for $\gamma_{ik}(z) = 1$.

Table 5.1: Evaluation of integrated stiffness constants

Integrated stiffness constant	C function call
$B_1 = \int_{a_0}^{a_n} A_{11}\, \psi_{1k}\, dz$	`calc_lamint(a11,Fpsi1,FNull,0)`
$C_{02} = \int_{a_0}^{a_n} A_{11}\, \varphi_{2k}\, z\, dz$	`calc_lamint(a11,Fvphi2,FNull,1)`
$D_{11} = \int_{a_0}^{a_n} A_{11}\, \psi_{1k}^2\, dz$	`calc_lamint(a11,Fpsi1,Fpsi1,0)`
$H_{25} = \int_{a_0}^{a_n} A_{13}\, \psi_{2k}\, \alpha_{5k}\, dz$	`calc_lamint(a13,Fpsi2,Falpha5,0)`

5.3.4 Trigonometric Series

The second set of routines is dedicated to processing trigonometric series approximations of the functions

$$u_i(x)\,, \quad w(x)\,, \quad \chi_g(x) \qquad (i=1,2;\ \ g=1,\ldots,8) \tag{5.4}$$

which are required for the calculation of the displacements, stresses and strains. In the application, these series are symbols referred to by a handle which is an enumerated constant.

The coefficients of the solution (4.8) of the system (4.6) are solved for any given pair (m,n) using Gauss–Jordan reduction, and the coefficients of $\chi_3(x),\ldots,\chi_8(x)$ are determined from the given loading. These coefficients are then stored in an array which is indexed by the handle of the symbol.

The routine `eval_trig_term` differentiates and evaluates a term of the form

$$\begin{aligned} &\sin\lambda_m x_1 \cos\gamma_n x_2\,, \qquad \sin\lambda_m x_1 \sin\gamma_n x_2\,, \\ &\cos\lambda_m x_1 \sin\gamma_n x_2\,, \quad \text{or} \quad \cos\lambda_m x_1 \cos\gamma_n x_2 \end{aligned} \tag{5.5}$$

A routine `eval_trig` takes the handle of a symbol as an argument, and uses `eval_trig_term` to differentiate and evaluate a double trigonometric series whose terms are of the form (5.5) and whose coefficients have been calculated.

Macro symbols for the trigonometric approximations of the functions (5.4) are defined; for example the C directives

```
#define  xu1(x1,x2,td)      eval_trig( Cu1,   Tcos_sin,td,x1,x2)
#define  xw(x1,x2,td)       eval_trig( Cw,    Tsin_sin,td,x1,x2)
#define  xchi1(x1,x2,td)    eval_trig( Cchi1, Tsin_sin,td,x1,x2)
```

define macros for the symbols for the functions $u_1(x), w(x)$ and $\chi_1(x)$, where `Cu1`, `Cw` and `Cchi1` are their handles and `td` indicates the differential operation to be performed. For example $w_{,12}(x_1,x_2)$ is evaluated by the macro `xw(x1,x2,TD12)` as given in Table 5.2.

Using these macros, the displacements (3.37), deflection (3.38), strains (3.40) and stresses (3.42) are readily evaluated.

The entire analysis based on the higher–order theory is incorporated into an optimization algorithm where, in each iteration, the stress/strain state of the plate is determined from the configuration and material properties and of the laminate.

Table 5.2: Evaluation of functions of the reference surface

Function	Macro symbol
$u_{1,2}(x_1, x_2)$	`xu1(x1,x2,TD2)`
$u_{2,11}(x_1, x_2)$	`xu2(x1,x2,TD11)`
$\chi_{6,122}(x_1, x_2)$	`xchi6(x1,x2,TD122)`
$\chi_{g,22}(x_1, x_2)$	`xchi(g,x1,x2,TD22)`

5.4 Optimal Design Problems

Three design problems are studied, namely the minimization of weight, deflection and stress of thick laminated sandwich plates. The sandwich structure is composed of relatively stiff top and bottom surface layers of thickness t_1 and t_3, respectively, and a core layer of thickness t_2 in between the surface layers as shown in Figure 5.1 on Page 128. The surface layers are made of a transversely isotropic material and carry most of the bending loads. In the minimum weight problem the core is made of a honeycomb material, and in the minimum stress problem, results are given for isotropic and transversely isotropic core layers which can model a variety of materials including honeycomb. The plate is of rectangular shape with sides a and b in the x_1 and x_2 directions, respectively, and a normal sinusoidal load of magnitude $q0$ is applied on the top surface.

5.4.1 Minimum Weight Design

The weight W of the sandwich plate is given by

$$W = (\rho_s t_s + \rho_c t_c)ab \tag{5.6}$$

where the subscripts s and c refer to the surface and core layers. For the sandwich structure under consideration $t_s = t_1 + t_3$ and $t_c = t_2$. Let the surface layers be made of a randomly orientated fibre composite material with isotropy in the plane and transverse isotropy through the thickness. In this case, ρ_s depends on the fibre volume content v_f so that $\rho_s = \rho_s(v_f)$. In the minimum weight design problem, t_s is taken as fixed and the total thickness $h = t_s + t_c$ as variable with a constraint on the total thickness. The design variables are chosen as v_f and h.

Using eqn. (5.6) and noting that $t_c = h - t_s$, the weight is obtained as

$$W(v_f, h) = [\rho_s(v_f)t_s + \rho_c(v_f)(h - t_s)]ab \tag{5.7}$$

The design problem can be stated as

$$\min_{v_f,h} W(v_f, h) \tag{5.8}$$

subject to thickness and fibre content constraints

$$h \leq h_0\ , \quad v_{f,\min} \leq v_f \leq v_{f,\max} \tag{5.9}$$

and a deflection constraint at a given point

$$w_b(x_1, x_2;\ v_f, h) \leq w_0 \tag{5.10}$$

where h_0, $v_{f,\min}$, $v_{f,\max}$ and w_0 are specified quantities and w_b denotes the deflection of the bottom surface. As the higher–order theory employed in this study is capable of determining the deflection at any point through the thickness of the plate, the location of the deflection in the z–direction has to be specified for design purposes. The deflection of the bottom surface is chosen as a constraint because of its practical importance.

The deflection depends on the fibre content v_f through the values of the elastic constants E_s, E'_s, G_s, G'_s and ν_s. For an in–plane randomly oriented material, the following micromechanical equations are used

$$\begin{aligned} E_s &= E_f v_f/3 + E_m v_m \\ E'_s &= E'_m/(1-(1-3E'_m/E'_f)\sqrt{v_f}) \\ G_s &= G_f v_f/3 + G_m v_m \\ G'_s &= G_m/(1-(1-3G_m/G'_f)v_f) \\ \nu_s &= 0.3 \end{aligned} \tag{5.11}$$

where f and m refer to the fibre and matrix properties, respectively, and a prime indicates a property in the transverse direction.

Here the expressions E_s and G_s are taken from Ref. [82] and the factor 1/3 reflects the reduction in moduli for a randomly oriented fibre composite as compared to a unidirectional composite. The expressions are valid for sufficiently stiff fibres ($E_f >> E_m$) [82]. The expressions for E'_s and G'_s for the elastic constants through the thickness are taken from Ref. [83] and the fibre properties are multiplied by a factor 1/3 in line with the expressions for E_s and G_s to account for the random orientation of the fibres. An average value is assigned to ν_s [82].

The density is computed from

$$\rho_s = \rho_f v_f + \rho_m v_m \tag{5.12}$$

The efficiency of a minimum weight design can be assessed by defining an efficiency index given by

$$\eta = \frac{W_{\min}}{W(0.5, h)} \tag{5.13}$$

where $W(0.5, h)$ is the weight of a plate made of surface layers only with $v_f = 0.5$ and h determined such that the deflection constraint (5.10) is satisfied. The index provides a weight comparison between the optimally designed sandwich structure and its single layer counterpart with no core region.

The minimum weight design involves the computation of the fibre content v_f and the total thickness h so as to solve the optimization problem (5.8)–(5.10). For a given h and deflection constraint w_0, the minimum v_f is determined such that the inequality (5.10) is satisfied. The optimal h is obtained by minimising the weight over h. This procedure yields the minimum weight sandwich having the optimal v_f and h. A Golden Section algorithm is used to compute the minimum fibre content v_f and the optimal thickness h_{opt}.

5.4.2 Minimum Deflection Design

The deflection behaviour of thick sandwich structures differ substantially from their thin counterparts when the effects of shear and normal deformation are taken into account and the optimal design problem should be formulated accordingly. One difference involves the deflection of the bottom surface with the load applied at the top surface. In this case, as the thickness of the surface layers increase, the deflection of the bottom surface does not necessarily decrease. In fact it decreases with increasing t_1 and t_3, but starts to increase after reaching a minimum. This is observed in Figure 5.2 on Page 129 where the curves of w_b at the centre of the plate are plotted against $t_s = t_1 + t_3$ with $t_3 = t_1$ for a square laminate with $a/h = 2$ and $E_1/E_2 = 50$. In Figure 5.2, w_b is nondimensionalised with respect to the deflection w_{iso} of an isotropic plate with stiffness $E = E_1$. It is observed that the deflection of the sandwich plate approaches that of the isotropic plate as t_s increases.

The deflection behaviour observed in Figure 5.2 can be explained by noting that the core layer absorbs some of the deformation in the normal direction. However, if the core layer becomes too thin, its capacity to absorb the deflection diminishes, leading to an increase in the deflection of the bottom surface as the core thickness becomes too small. As a result the bottom surface of a structure with no core layer may deflect more than an optimally designed sandwich structure. This phenomenon can

neither be investigated nor taken into account using the classical theory of plates or a theory which excludes the effect of normal deformation.

In the light of these results the minimum deflection problem may be stated as

$$\min_{t_s} \max_{x_1,x_2} w_b(x_1, x_2, t_s) \tag{5.14}$$

where $0 \leq t_s \leq h$ with $t_s = 0$ and $t_s = h$ corresponding to a plate of core layer only and surface layer only, respectively. The efficiency of the optimally designed laminates is assessed by defining the ratio

$$\delta = \frac{w_b(\bar{x}_1, \bar{x}_2; t_{\text{opt}})}{\bar{w}_b(\bar{x}_1, \bar{x}_2; h)} \tag{5.15}$$

where $(\bar{x}_1,\ \bar{x}_2)$ is the location of maximum deflection, t_{opt} the optimal value of t_s and $\bar{w}_b(\bar{x}_1, \bar{x}_2; h)$ the deflection of a laminate composed of surface layers only. The quantity δ serves as an efficiency index to compare the optimal sandwich to its single layer counterpart with no core layer.

The solution of the minimum deflection problem is obtained by solving the minmax problem (5.14) which yields the optimal thickness of the surface layers.

5.4.3 Minimum Stress Design

In thick sandwich plates with a core region whose stiffness is relatively low, the stress distribution through the thickness is not symmetrical even if the lamination is. This is due to the fact that the transverse load applied on the top surface leads to stresses in the top layers which are different from those in the bottom layers. The resulting stress distribution cannot be symmetrical as predicted by theories which fail to take normal deformation into account. An optimal design for minimum stress involves the computation of the relative thicknesses of layers such that the maximum normal stress will be minimized.

The normal stresses are given by $\sigma_{11} = \sigma_{11}(x_1, x_2, z, t_1)$ and $\sigma_{22} = \sigma_{22}(x_1, x_2, z, t_1)$ with the core thickness t_2 being a given parameter. Then the maximum normal stress is given by

$$\sigma_{\max} = \max_{i=1,2} \left(\max_{x_1,x_2,z} \sigma_{ii}(x_1, x_2, z, t_1) \right) \tag{5.16}$$

The optimal design problem can be stated as

$$\min_{t_1} \sigma_{\max} \tag{5.17}$$

for a given total thickness $h = t_1 + t_2 + t_3$. The thickness of the top and bottom layers are subject to the practical constraint

$$t_1/h \geq 0.02\,, \quad t_3/h \geq 0.02 \tag{5.18}$$

In the present problem, the total thickness h is specified and t_2 is an input parameter. Thus $t_3 = h - t_1 - t_2$ where t_1 is the optimal thickness of the top layer.

5.5 Numerical Results

Numerical results are given for a square sandwich plate of dimensions $a \times a$ ($a = 1\text{m}$) subjected to a sinusoidal load of amplitude $q_0 = 1\,\text{MPa}$ on the top surface.

5.5.1 Minimum Weight Design

Results are given for surface layers made of T300 graphite fibres whose properties are taken as

$$\begin{aligned} E_f &= 258.6\,\text{GPa}, \quad E'_f = 18.2\,\text{GPa}, \quad \nu_f = 0.2 \\ G_f &= 36.7\,\text{GPa}, \quad G'_f = 20\,\text{GPa}, \quad \rho_f = 1750\,\text{kg/m}^3 \end{aligned} \tag{5.19}$$

and the epoxy matrix properties are taken as

$$\begin{aligned} E_m &= 3.45\,\text{GPa}, \qquad \nu_m = 0.35 \\ G_m &= E_m/2(1+\nu_m), \quad \rho_m = 1200\,\text{kg/m}^3 \end{aligned} \tag{5.20}$$

The core section is made of a 17-7 PH stainless steel honeycomb material with the elastic constants [84]

$$\begin{aligned} E_c &= 1.58\,\text{GPa}, \quad \rho_c = 124\,\text{kg/m}^3 \\ G_{xyc} &= 0.50\,\text{GPa}, \quad G_{yzc} = 0.68\,\text{GPa} \end{aligned} \tag{5.21}$$

These values correspond to a honeycomb with a cell size of 0.25in. First the dependence of the weight W on the thickness h is investigated for various values of t_s and w_0 as shown in Figure 5.3 on Page 130. It is observed that W displays a minimum point with respect to h which increases with increasing t_s. The fibre contents for various values of t_s and w_0 are shown in Figure 5.4 with $0.2 \leq v_f \leq 0.7$. As expected v_f decreases as h increases for a given t_s, but increases as the deflection constraint becomes smaller.

Next the minimum weight results are presented. The minimum weight decreases as the deflection constraint is relaxed as shown in Figure 5.5 for a given surface layer thickness $t_s = 20$mm. However, $W_{\min}$ increases as t_s increases as shown in Figure 5.6 for a given constraint $w_0 = 0.88$mm.

Table 5.3: Optimal h and v_f for the minimum weight design.

t_s (mm)	w_0 (mm)	h_{opt} (mm)	v_f (%)	$W_{\min}$ (kg)	Efficiency η
20	.88	163	47.2	46.93	0.196
22	.84	168	43.4	49.72	0.201
24	.78	175	40.4	52.90	0.206
26	.76	179	37.3	55.46	0.210
28	.72	185	35.0	58.42	0.216
30	.70	188	32.6	61.03	0.220
32	.66	195	30.9	64.10	0.225
34	.64	199	29.2	66.77	0.227
36	.62	204	27.6	69.47	0.232
38	.60	208	26.4	72.19	0.236
40	.58	213	25.1	74.94	0.240

Table 5.3 gives the h_{opt} and v_f values for a minimum weight design for various values of t_s and w_0. It is observed that the minimum weight sandwich construction provides a substantial weight saving as compared to a single layer construction with the efficiency decreasing as the deflection constraint becomes tighter.

5.5.2 Minimum Deflection Design

In this problem the design variable is the total thickness t_s of the surface layers. Results are given for a fixed h with the thickness ratio specified as $a/h = 2$.

The elastic properties of the transversely isotropic surface layers are given by $E_1 = 1\,\text{GPa}$, $\nu_1 = 0.3, G_1 = E_1/2(1+\nu_1), \nu_1' = \nu_1 E_1'/E_1, G_1' = E_1'/2(1+\nu_1')$. The core layer is taken to be isotropic with its elastic properties given by E_2, $\nu_2 = 0.3$ and $G_2 = E_2/2(1+\nu_2)$. As E_1 is fixed at 1 GPa, in the ratios E_1/E_2 and E_1/E_1' used in the figures, the denominator is varied. The total relative thickness of the surface layers is $t_s/h = (t_1 + t_3)/h$. We consider a symmetrically laminated plate, and

therefore $t_1 = t_3$ and $t_s/h = 2t_1/h$.

Figure 5.7 shows the optimal thickness t_{opt}/h vs E_1/E_2 for $E_1/E_1' = 5, 10, 20$. It is observed that t_{opt} decreases as the ratio E_1/E_2 increases, that is, as the surface layers become stronger relative to the core layer. The minimum values of the deflection, $w_{b,\text{min}}$, are given in Figure 5.8. As E_1/E_2 increases, $w_{b,\text{min}}$ decreases due to the core region absorbing more of the normal deformation. An interesting phenomenon observed in Figure 5.8 is the existence of negative deflection for $E_1/E_1' = 5$ when $E_1/E_2 \geq 74$. This phenomenon is further investigated below. Figure 5.9 gives the corresponding curves for the efficiency ratio δ. The efficiency of the design, in general, increases as E_1/E_2 increases, i.e., as the core layer becomes weaker. Negative values for δ occur as a result of $w_{b,\text{min}}$ becoming negative for certain ratios of elastic moduli as observed in Figure 5.8.

Figure 5.10 shows the curves of the t_{opt}/h plotted against E_1/E_1' for various values of E_1/E_2. It is observed that t_{opt} increases for high values of E_1/E_2 and decreases for low values of E_1/E_2 as E_1/E_1' increases, i.e., as the through–the–thickness modulus E_1' decreases. For intermediate values, i.e. for $E_1/E_2 = 50$, it increases for low values of E_1/E_1' and decreases as E_1/E_1' increases. As shown in Figure 5.11, the values of $w_{b,\text{min}}$ increase as E_1' becomes smaller. The corresponding efficiency curves are given in Figure 5.12. Again it is observed that δ is negative in some cases.

Next the behaviour of the deflection through the thickness is investigated, in particular the phenomenon of negative deflection and the effect of excluding normal deformation. Figure 5.13 shows the deflection curves through the thickness of the plate for various cases with $E_1/E_2 = 50$ and $E_1/E_1' = 10$. The deflection curve for the optimal sandwich with $t_{\text{opt}}/h = 0.803$ ($t_1 = t_2 = t_{\text{opt}}/2$) is shown in Figure 5.13a. It is observed that the top layer deflects more than the bottom one and there exists a minimum point across the thickness. The corresponding curve for a single–layered laminate is shown in Figure 5.13b. For this case the efficiency index is $\delta = 1.11/3.79 = 0.293$ indicating a 70% reduction in the deflection. Figure 5.13c shows the deflection of the sandwich plate with the effect of normal deformation neglected. It is clear that neglecting this effect leads to a completely erroneous result.

The corresponding deflection curves are given in Figures 5.14a, 5.14b and 5.14c for $E_1/E_2 = 90$ and $E_1/E_1' = 5$. In is observed from Figure 5.14a that the bottom surface of the optimal sandwich has a negative deflection. This explains the negative values for δ which for this case is $\delta = -0.443/1.11 = -0.40$. Figure 5.14c again shows the effect of neglecting the normal deformation. In this case the phe-

nomenon of negative deflection cannot be observed when the normal deformation is left unaccounted.

5.5.3 Minimum Stress Design

The elastic properties of the transversely isotropic surface layers are given by $E_1 = 1\,\text{GPa}$, $E_1' = E_1/2$, $\nu_1 = \nu_1' = 0.3$, $G_1 = E_1/2(1+\nu_1)$ and $G_1' = E_1'/2(1+\nu_1')$. The elastic properties of the core layer are given by E_2, $\nu_2 = \nu_2' = 0.15$, $G_2 = E_2/2(1+\nu_2)$ and $G_2' = E_2'/2(1+\nu_2')$. As E_1 is fixed at 1 GPa, in the ratio E_1/E_2 used in the figures, the denominator is varied.

To assess the efficiency of the designs, a comparison is made between the optimally designed and symmetrical sandwich plates. For this purpose, an efficiency index is defined as

$$\eta = \frac{\sigma_{\max}(t_1^*)}{\sigma_{\max}(\bar{t}_1)} \tag{5.22}$$

where t_1^* indicates the optimal thickness of the top surface and $\bar{t}_1 = (h - t_2)/2$, i.e. the thickness of the top layer of a symmetrical plate. In the discussion below, η is referred to as the stress ratio.

First the behaviour of the stress ratio is investigated as a function of the design variable t_1. Figure 5.15a shows the curves of η versus t_1/h for various values of the core thickness t_2/h for a thickness ratio of $a/h = 2$ and an isotropic core layer. It is observed that in general the minimum stress ratio is achieved by a nonsymmetric design. Figure 5.15b shows the same curves as in Figure 5.15a with the effect of normal deformation neglected. For this case, the results indicate that the optimal design is always a symmetric laminate and the stress ratio is greater than one for any other configuration. Therefore it is essential to include normal deformation in the analysis of thick sandwich plates to obtain the correct optimization results. Figure 5.16a shows the curves of η versus t_1/h for various thickness ratios for a transversely isotropic core with $E_2'/E_2 = 10$ and $t_2/h = 0.5$. As in the previous case, the stress ratio in minimized at certain values of t_1. Figure 5.16b shows the same curves as in Figure 5.16a with the effect of normal deformation neglected. In the vicinity of the optimal point ($t_1/h = 0.25$), the greater the thickness ratio, the greater the error as compared to the results given in Figure 5.16a. Figures 5.15 and 5.16 indicate that an optimal choice of t_1 can lead to substantial reductions in the stress ratio, i.e., the optimal plate will be considerably more efficient than the symmetrical one as determined by the efficiency index η.

The optimal values of t_1, denoted by t_1^*, and the corresponding stress ratios are plotted against t_2/h in Figure 5.17 for various values of E_1/E_2. The jump discontinuities in the values of t_1 occur due to the existence of local minima. It is observed that in all cases the top layer is thinner than the bottom layer except for $E_1/E_2 = 10$ and $t_2/h = 0.8$. However, t_1^* approaches the symmetrical case as t_2 increases. Figure 5.17b shows that the efficiency of the design drops as t_2 increases. The curves of t_1^*/h and η versus a/h are shown in Figure 5.18 for various values of t_2/h and with $E_2'/E_2 = 1$ (isotropic core). As a/h increases, i.e., as the plate becomes thinner, t_1^*/h tends to get larger, and in some cases, the top layer can be thicker than the bottom layer. This situation arises as the core layer becomes thicker. As a/h increases, the efficiency drops, except for the case $t_2/h = 0.8$ as can be seen from Figure 5.18b. Corresponding curves for a sandwich plate with a transversely isotropic core are given in Figure 5.19 where $E_2'/E_2 = 10$. In this case, the top layer can be thicker than the bottom layer in several cases. However, the general trend of t_1^*/h as a function of a/h is similar to the case with an isotropic core layer. A comparison of Figures 5.18b and 5.19b indicates that the efficiency drops in the case of a plate with a transversely isotropic core.

The effect of stiffness ratio E_1/E_2 on t_1^* and η is investigated in Figure 5.20 which shows the curves of t_1^*/h and η versus E_1/E_2 for $a/h = 4$. The results are obtained subject to the constraint $t_1/h \geq 0.02$. It is observed that t_1^* shows different trends depending on the values of t_2/h and E_1/E_2. The efficiency, in general, improves with increasing E_1/E_2.

Typical stress distributions through the thickness are shown in Figure 5.21 for the case $a/h = 3$, $E_1/E_2 = 20$, $E_1/E_1' = 2$, $E_2'/E_2 = 1$ and $t_2/h = 0.6$. Figure 5.21a shows the stress distribution for an optimal plate, Figure 5.21b for a symmetrical plate, and Figure 5.21c for a symmetrical plate with the effect of normal deformation neglected. It is observed that in the case of an optimal plate, the maximum stresses are the same at the top and bottom layers. Figure 5.21c indicates that neglecting normal deformation would yield a completely incorrect stress distribution and render the solution meaningless as is illustrated by Figures 5.15b and 5.16b.

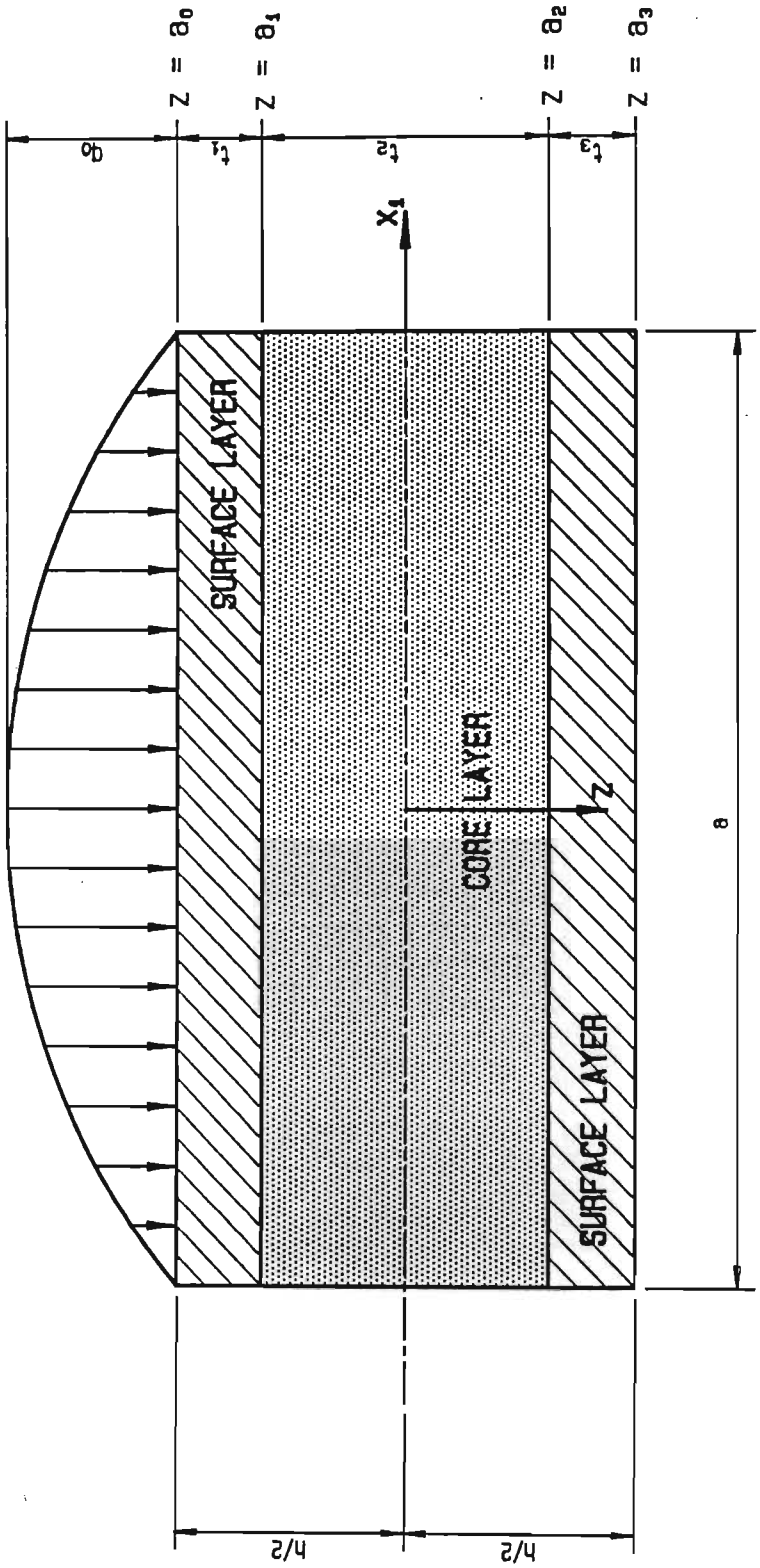


Figure 5.1. Geometry and loading of a sandwich plate

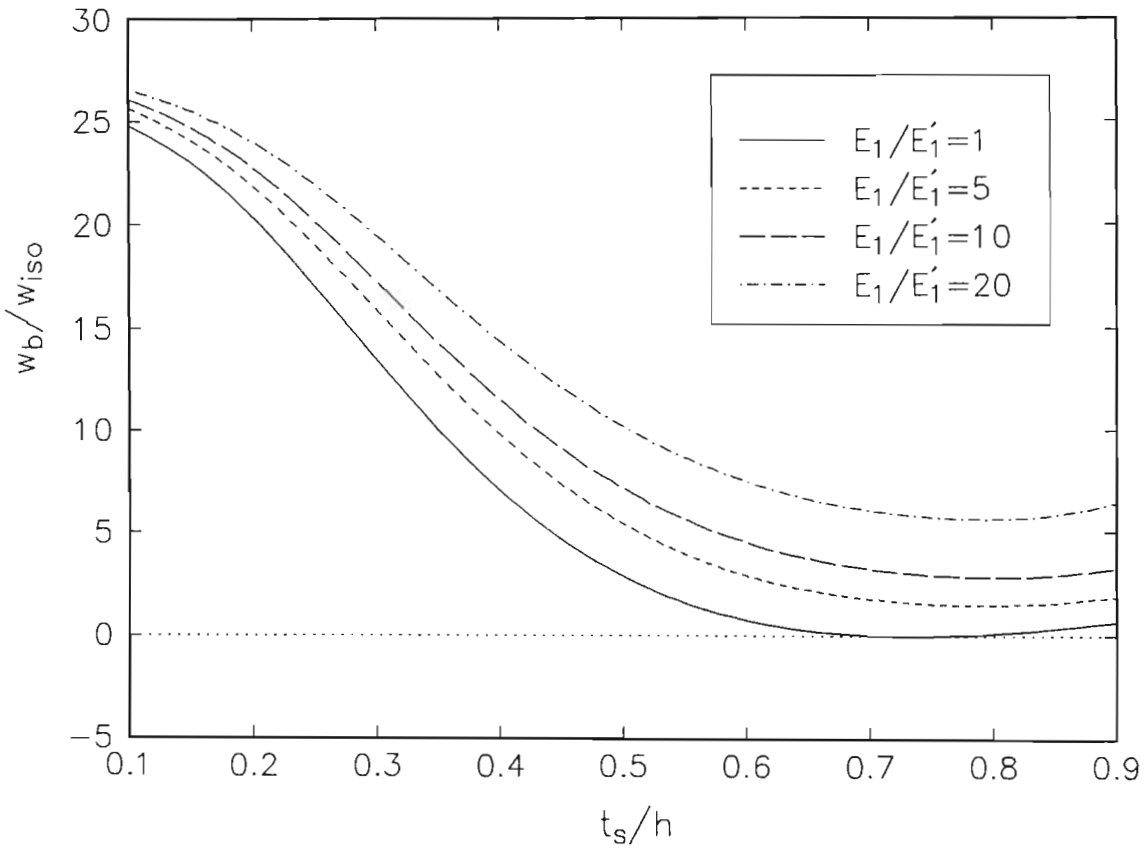


Figure 5.2. Deflection versus t_s/h with $E_1/E_2 = 50$

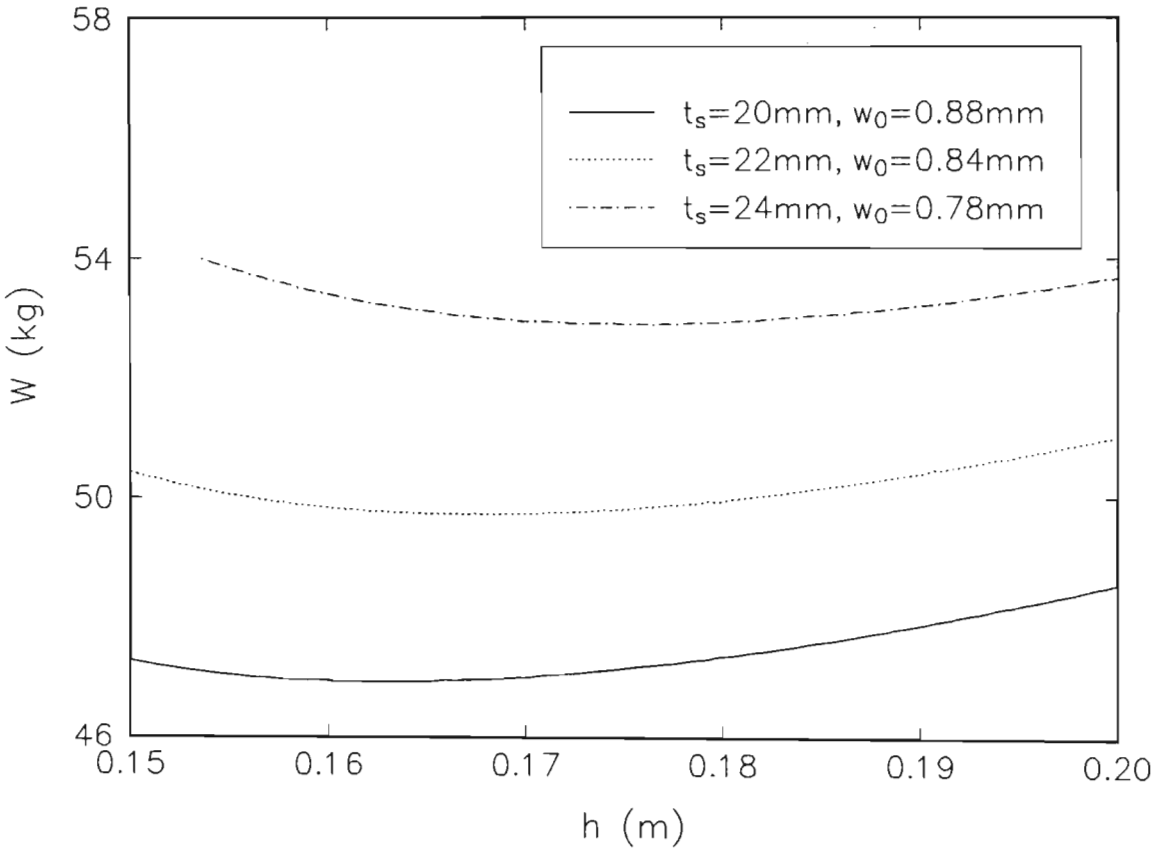


Figure 5.3. Weight versus the total thickness h

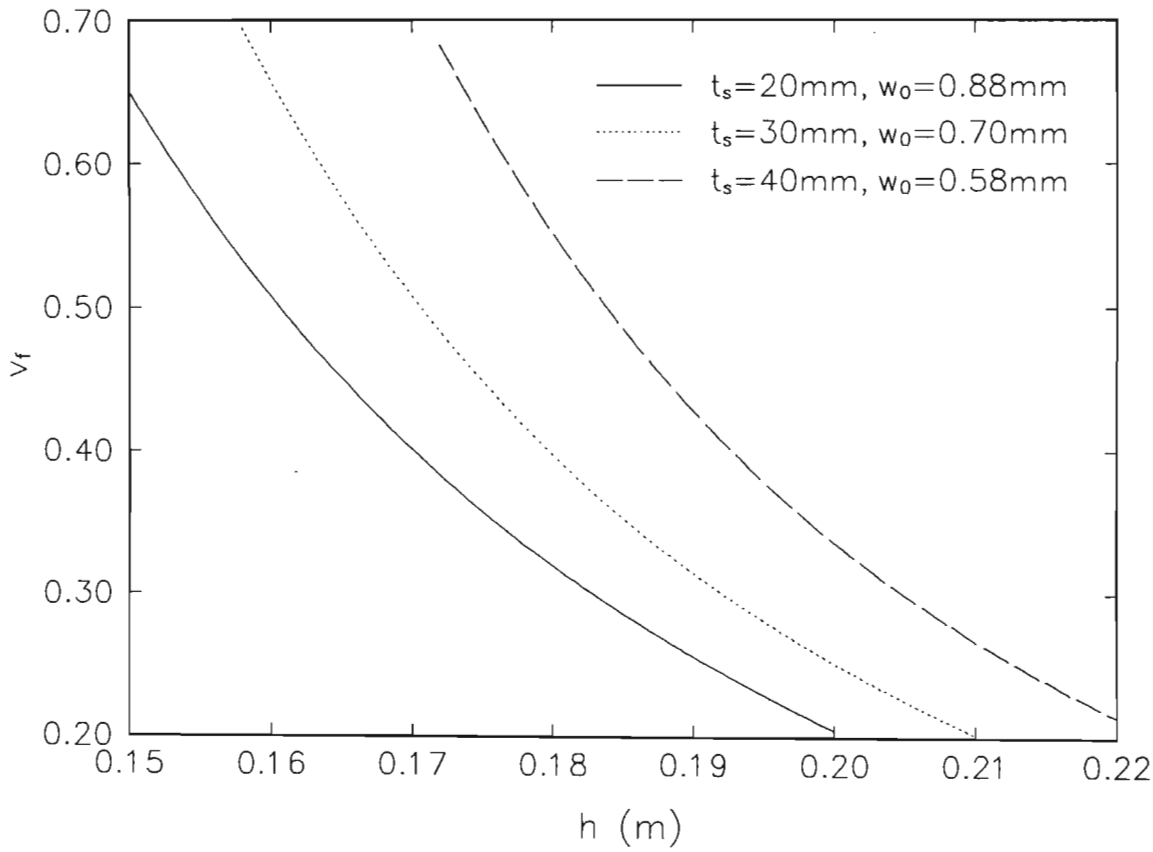


Figure 5.4. Fibre content v_f versus the total thickness h

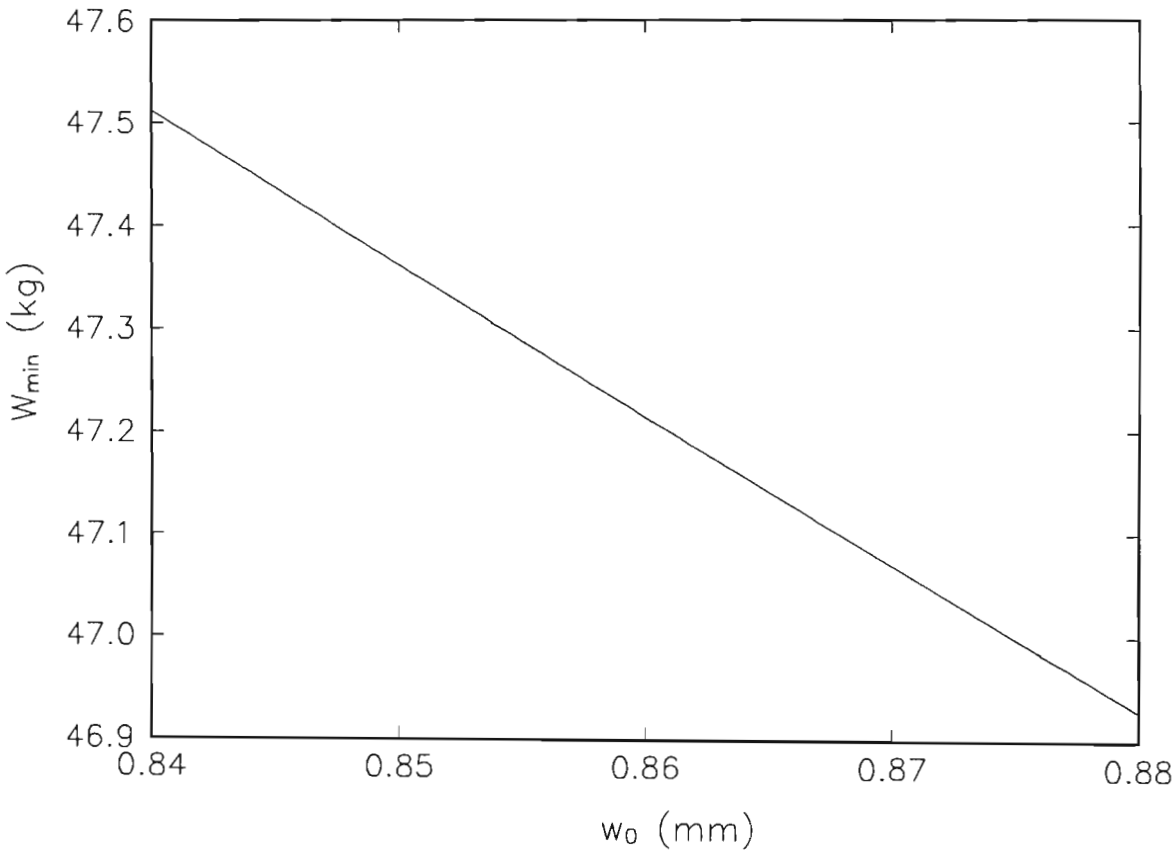


Figure 5.5. Minimum weight versus the deflection constraint with $t_s = 20$mm

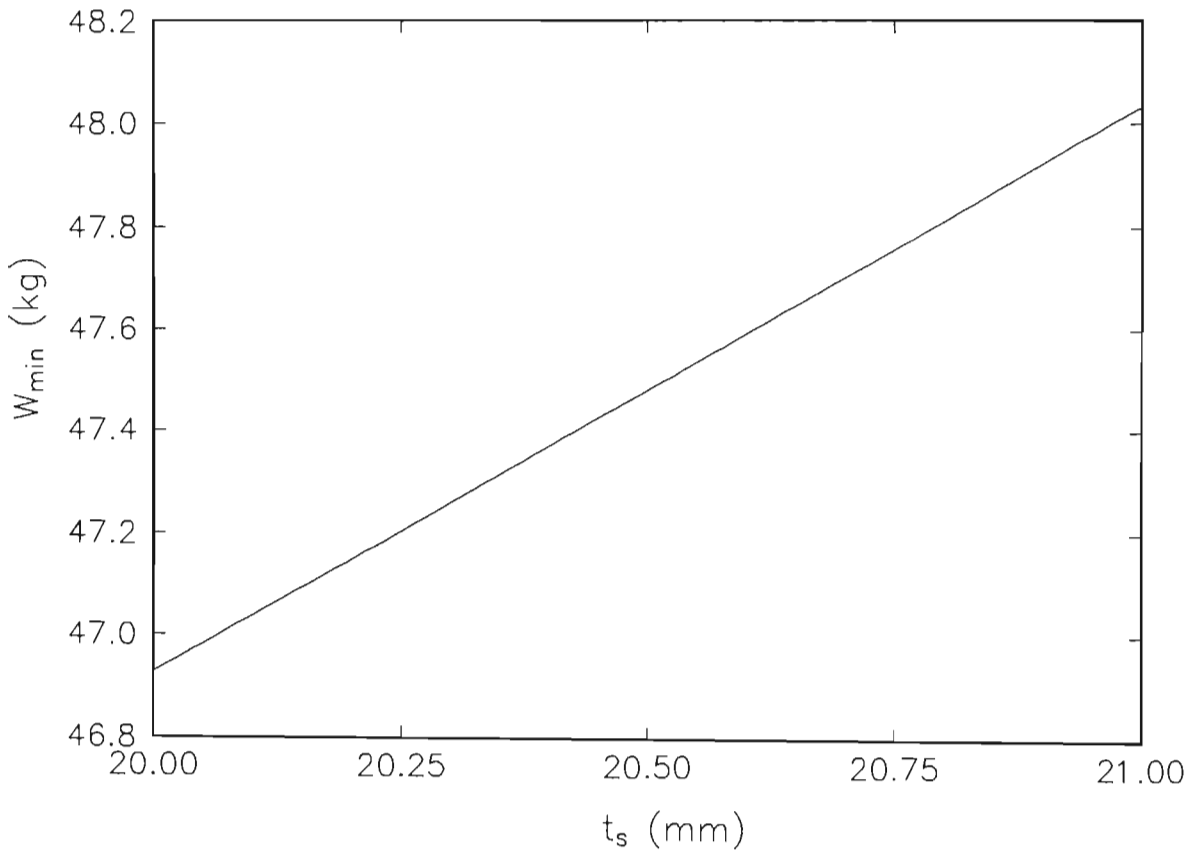


Figure 5.6. Minimum weight versus t_s with $w_0 = 0.88$mm

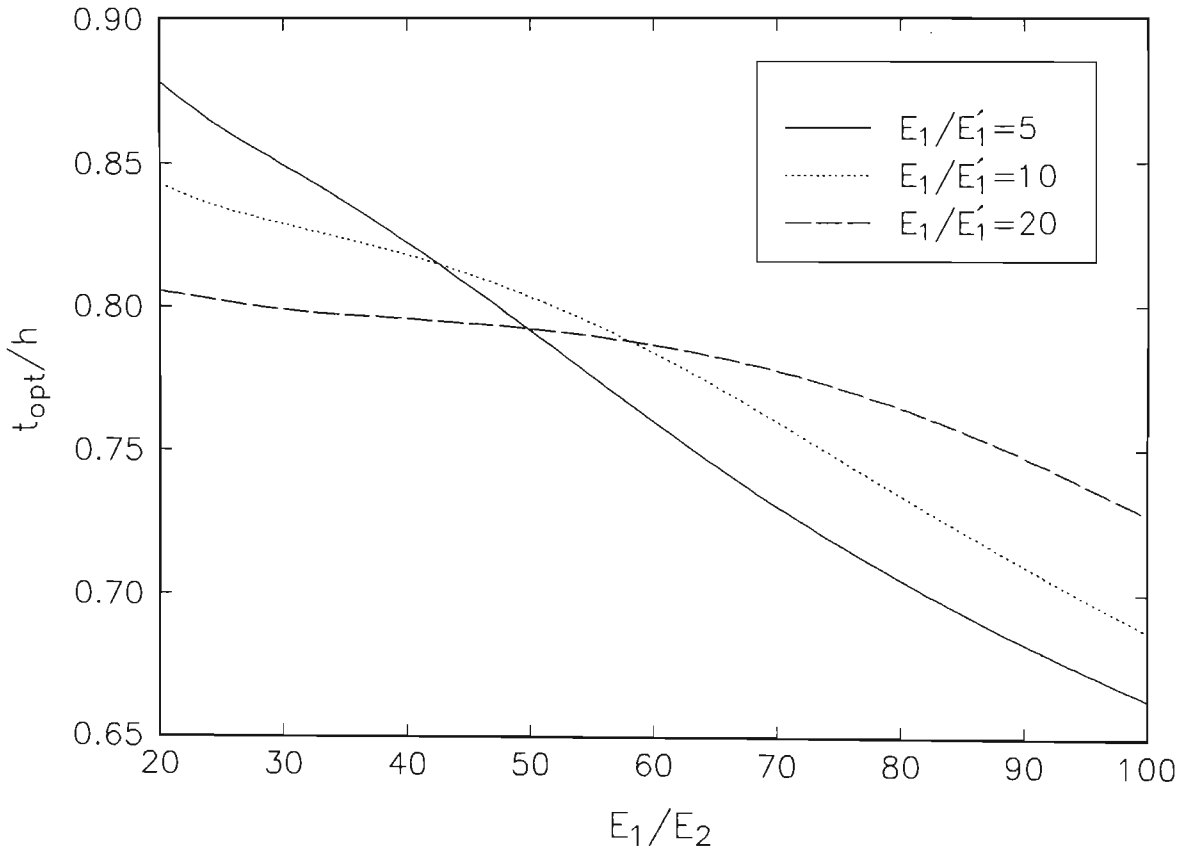


Figure 5.7. Optimal thickness ratio t_s/h versus E_1/E_2 for the minimum deflection design

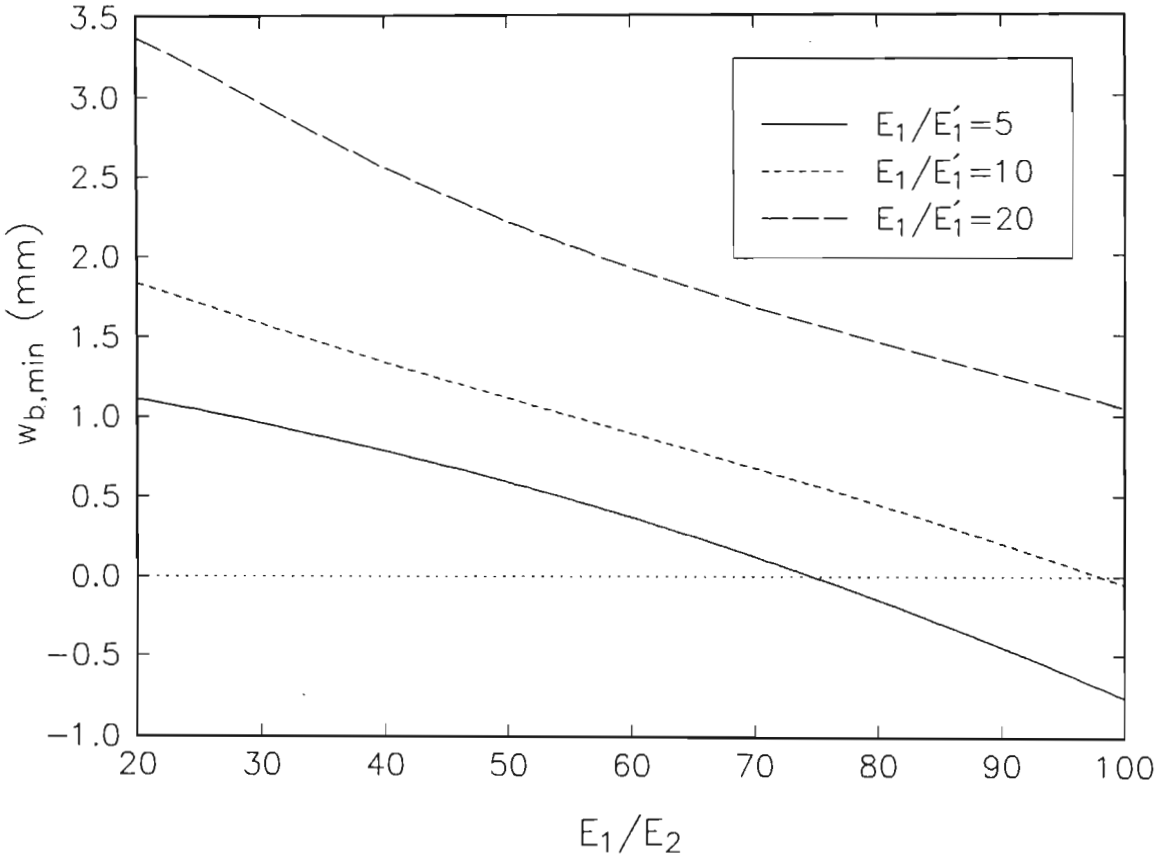


Figure 5.8. Minimum deflection $w_{b,\min}$ versus E_1/E_2

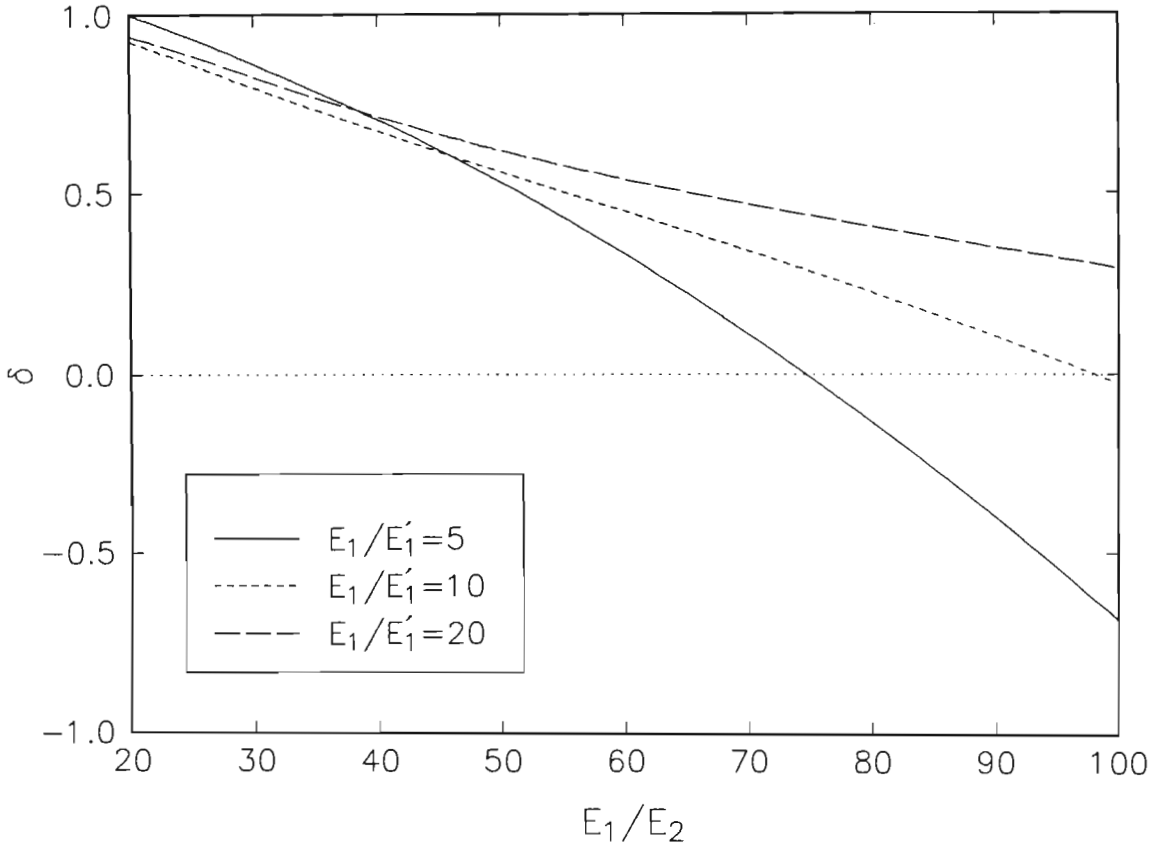


Figure 5.9. Efficiency index versus E_1/E_2

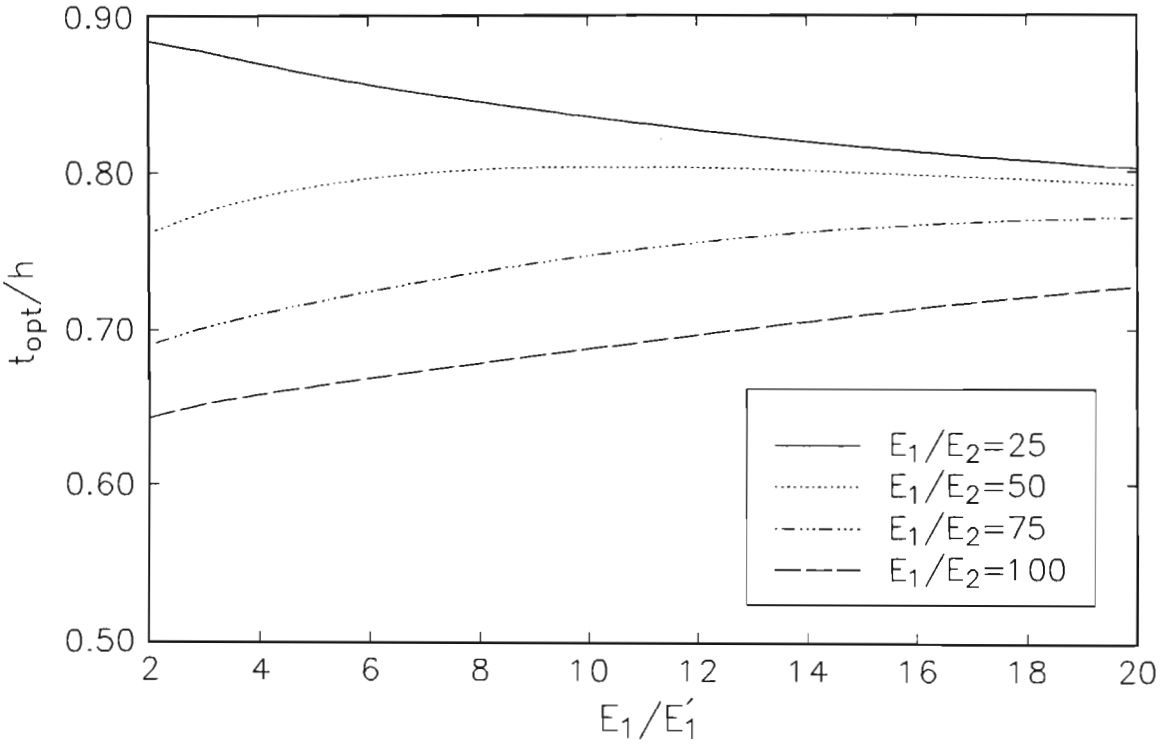


Figure 5.10. Optimal thickness ratio t_{opt}/h versus E_1/E_1' for the minimum deflection design

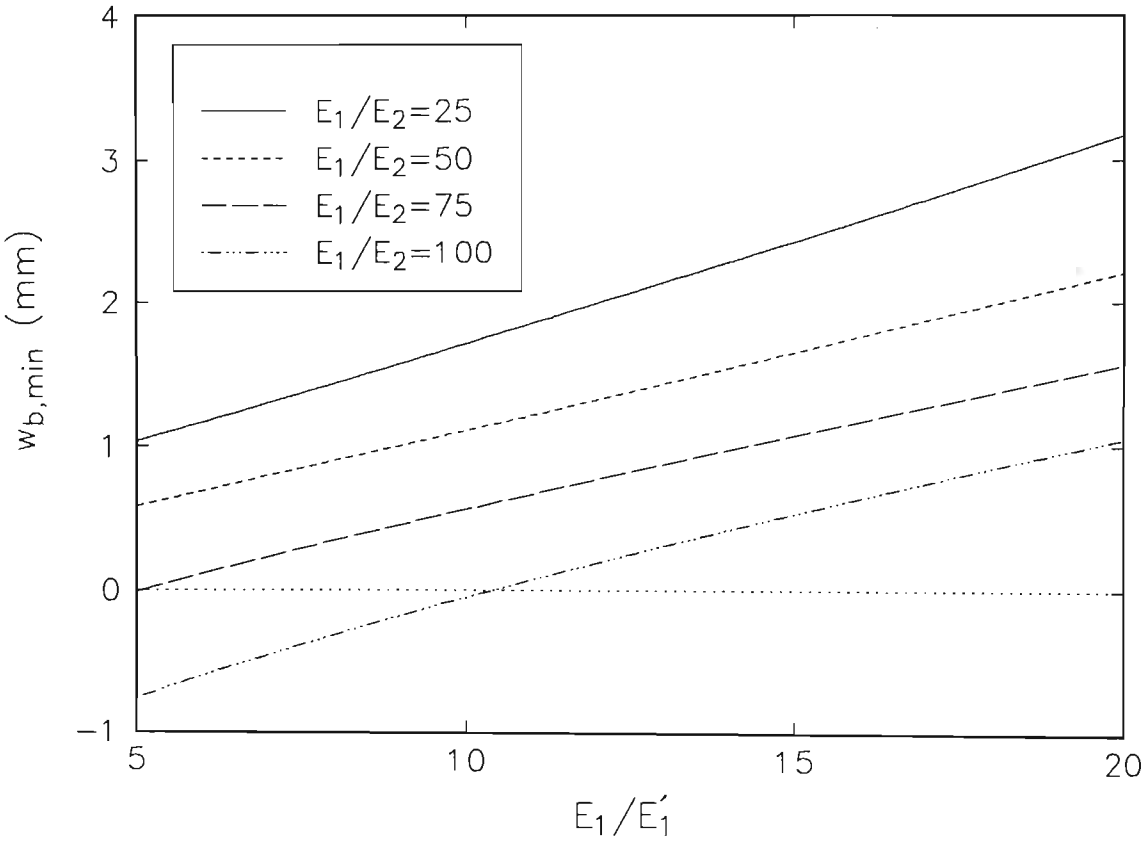


Figure 5.11. Minimum deflection $w_{b,\min}$ versus E_1/E_1'

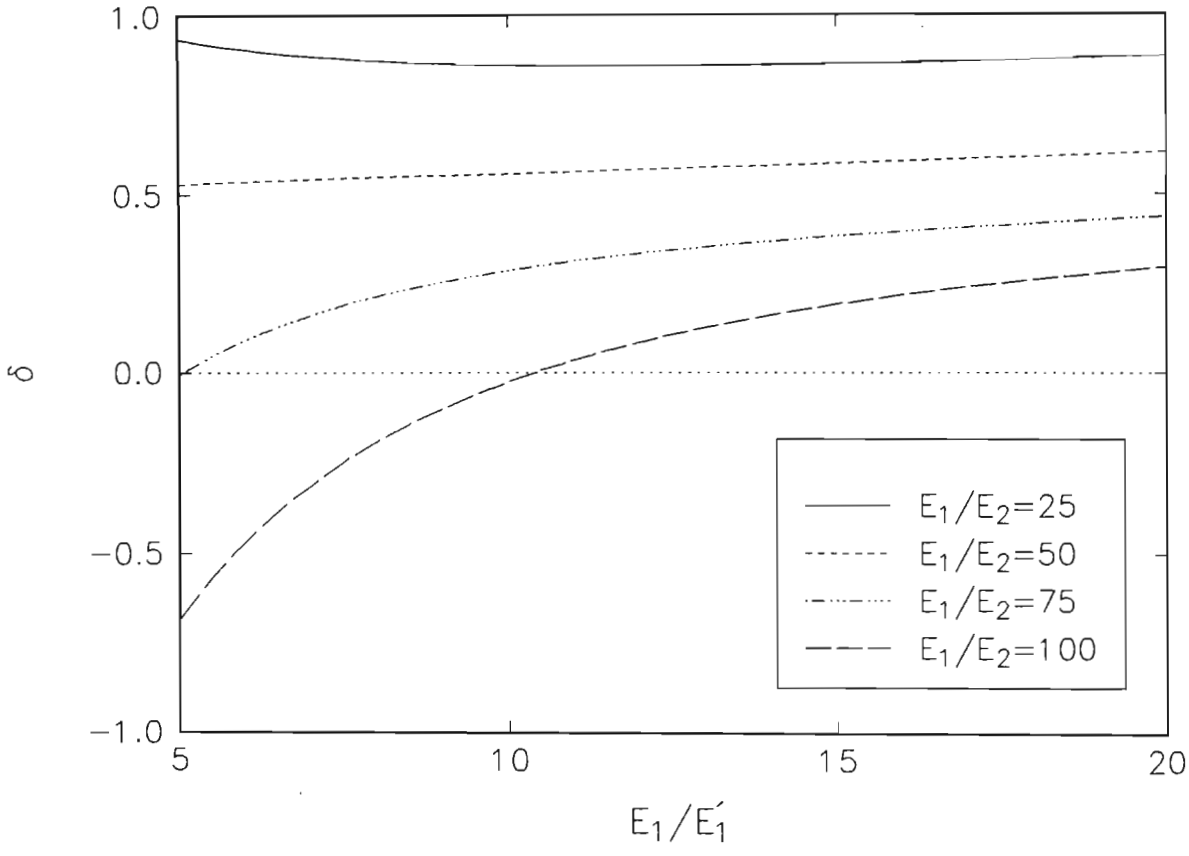


Figure 5.12. Efficiency index versus E_1/E_1'

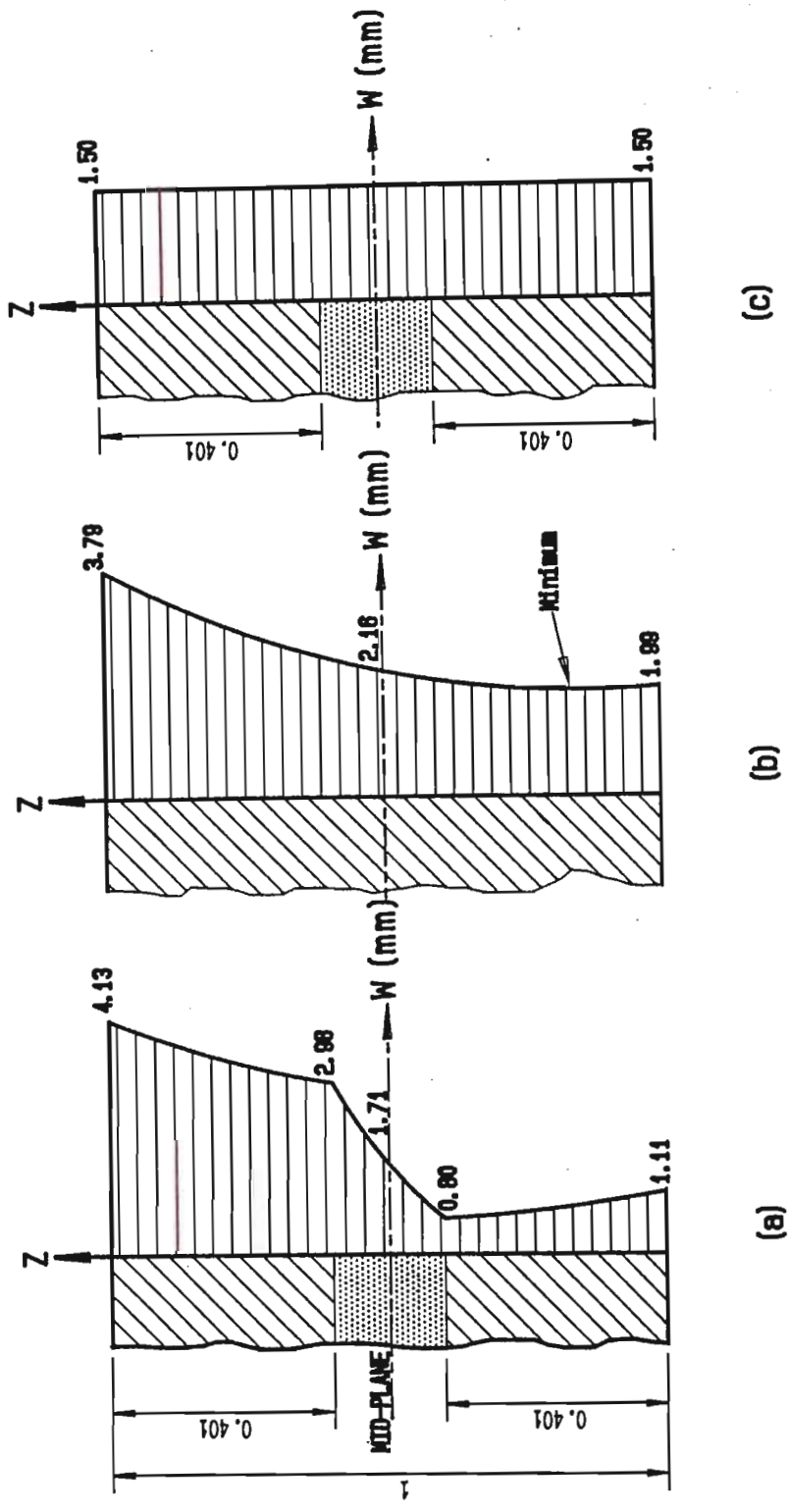


Figure 5.13. Deflection distribution for $E_1/E_2 = 50$ and $E_1/E_1' = 10$ for (a) $t_{s,\text{opt}}/h = 0.803$, (b) single–layered laminate, and (c) sandwich plate without normal deformation

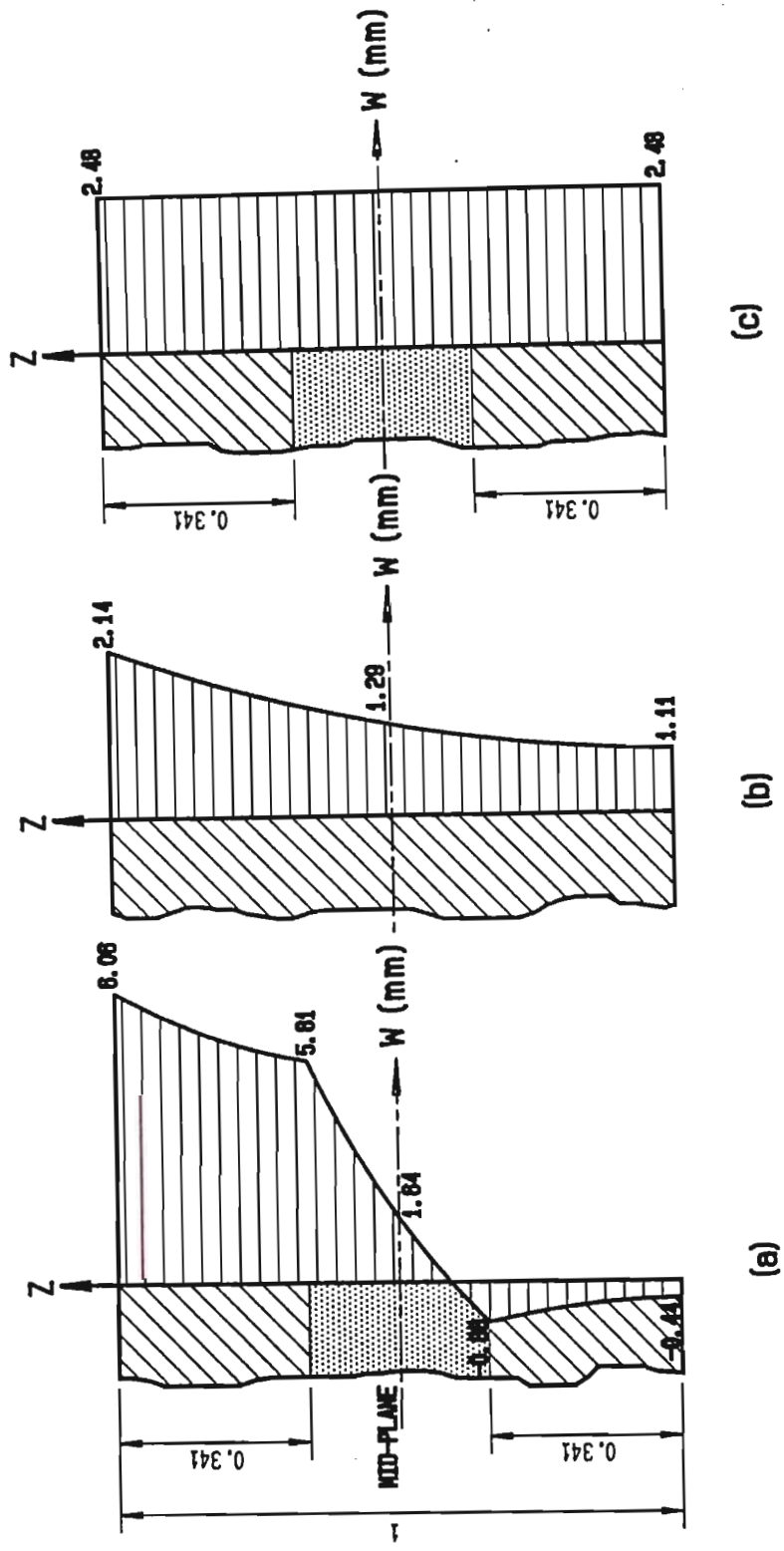


Figure 5.14. Deflection distribution for $E_1/E_2 = 90$ and $E_1/E_1' = 5$ for
(a) $t_{s,\text{opt}}/h = 0.683$, (b) single–layered laminate, and
(c) sandwich plate without normal deformation

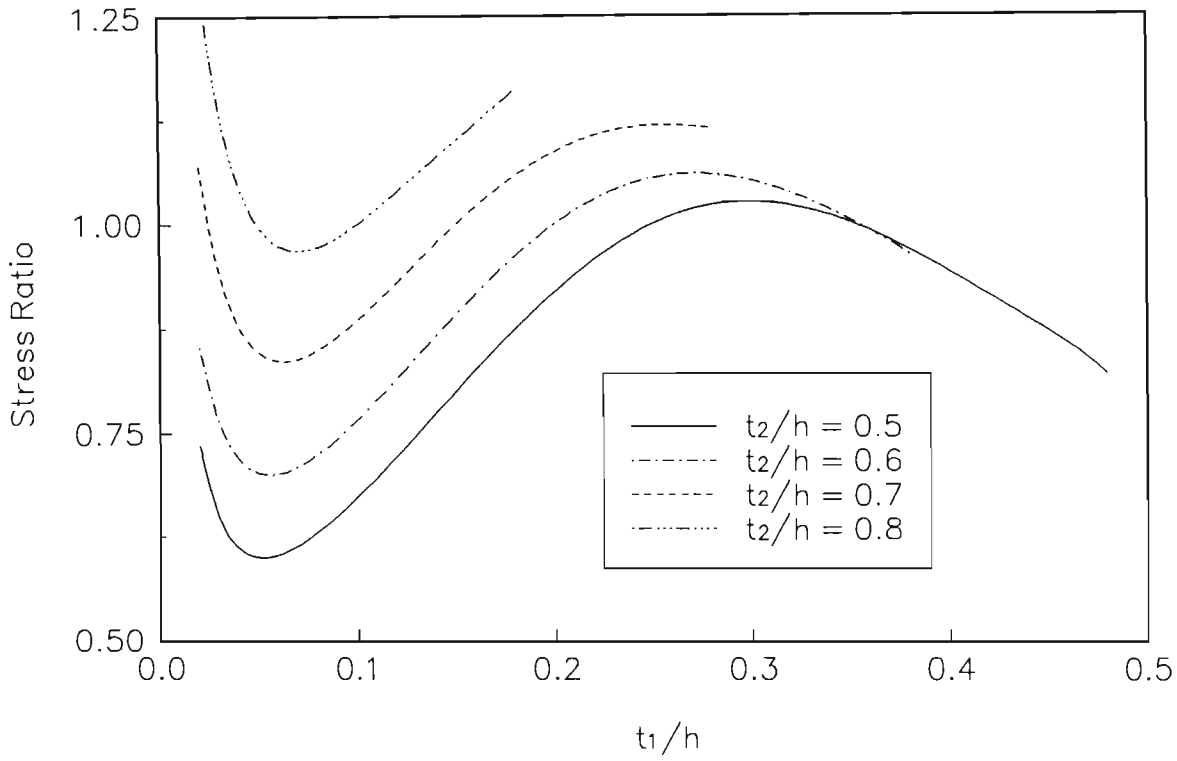


Figure 5.15a. Stress ratio η vs t_1/h with $a/h = 2$, $E_1/E_2 = 10$ and $E_2'/E_2 = 1$ (isotropic core)

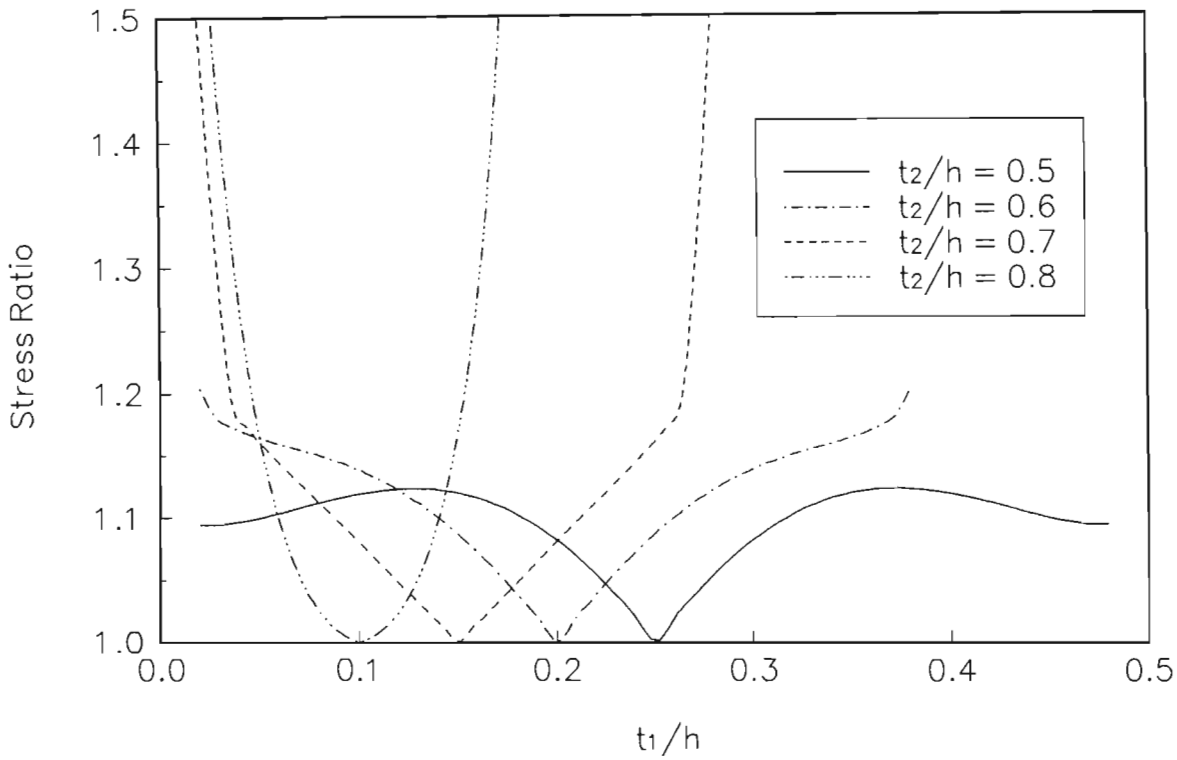


Figure 5.15b. Stress ratio η vs t_1/h with $a/h = 2$, $E_1/E_2 = 10$, $E_2'/E_2 = 1$ (isotropic core) and normal deformation neglected.

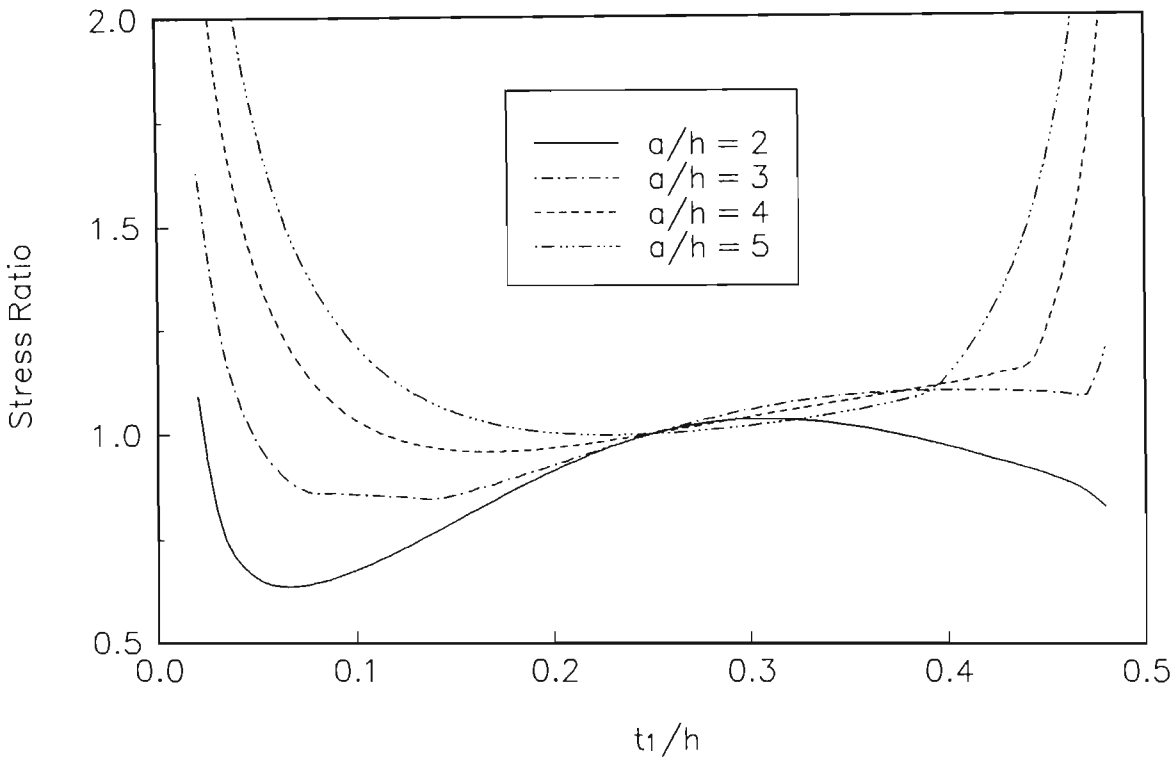


Figure 5.16a. Stress ratio η vs t_1/h with $t_2/h = 0.5$, $E_1/E_2 = 100$ and $E_2'/E_2 = 10$ (transversely isotropic core)

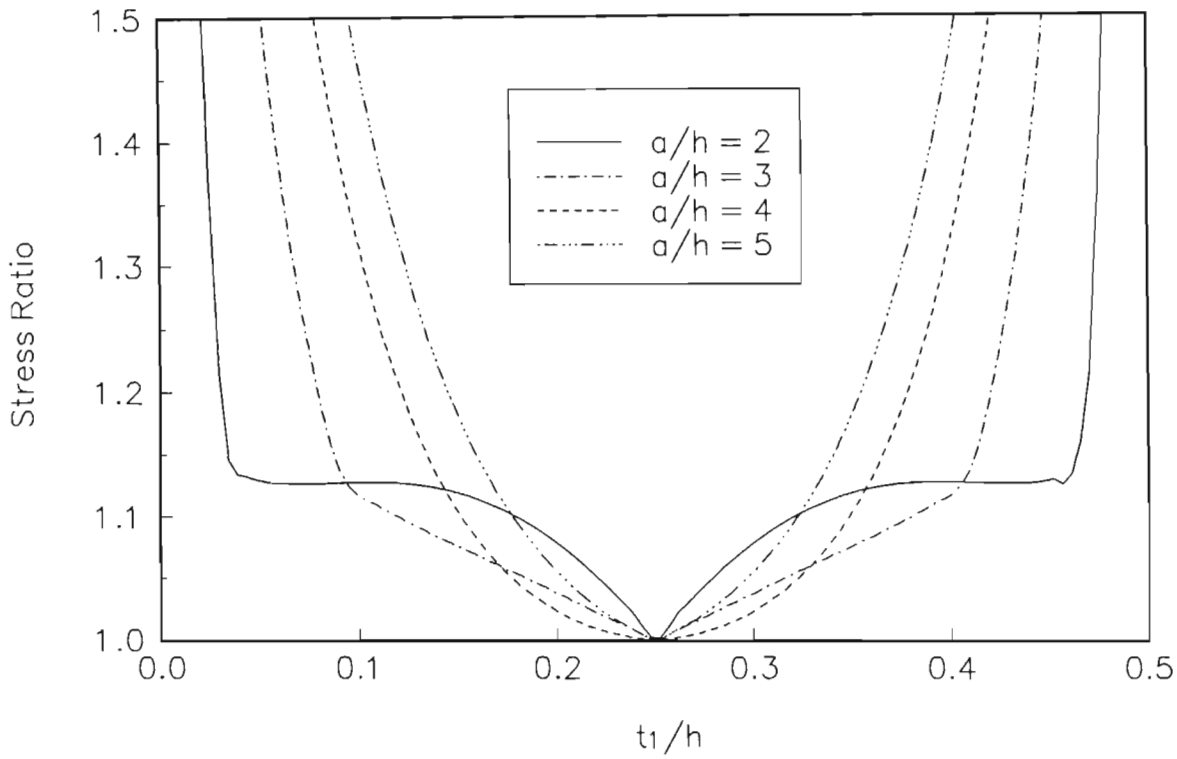


Figure 5.16b. Stress ratio η vs t_1/h with $t_2/h = 0.5$, $E_1/E_2 = 100$, $E_2'/E_2 = 10$ (transversely isotropic core) and normal deformation neglected.

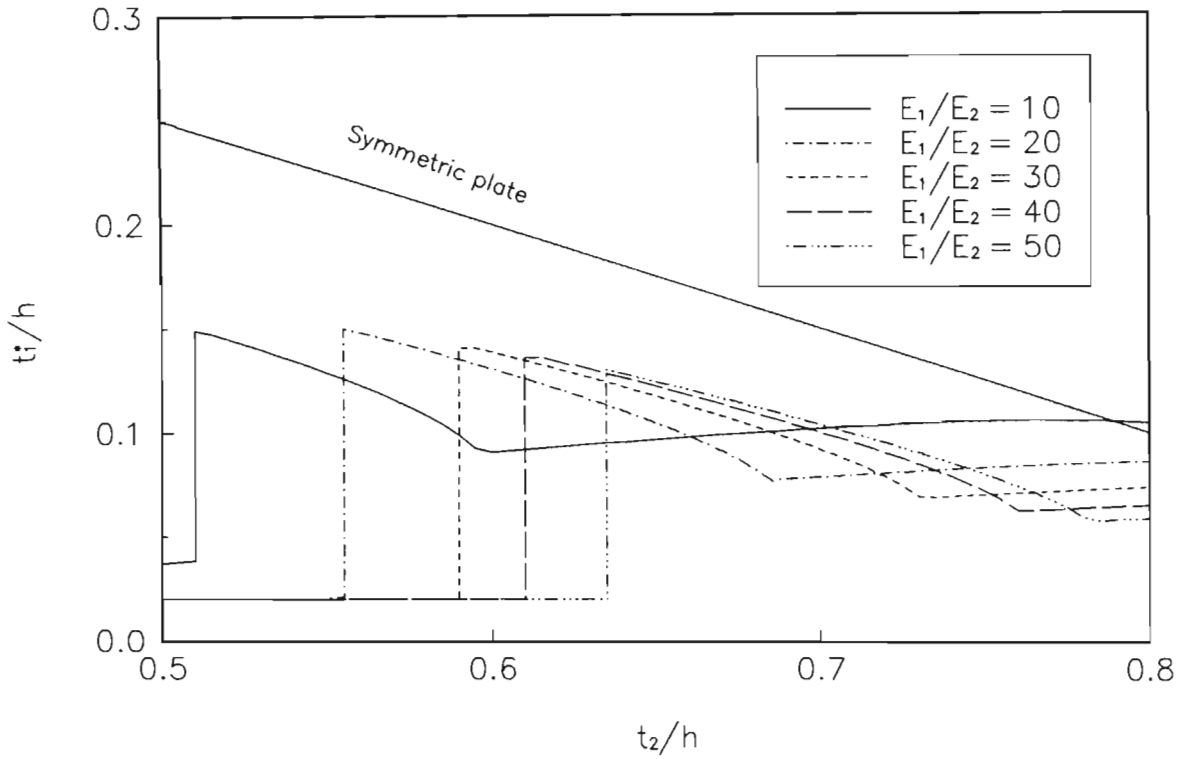


Figure 5.17a. Optimal thickness t_1^*/h vs core thickness t_2/h with $a/h = 3$ and $E_2'/E_2 = 1$

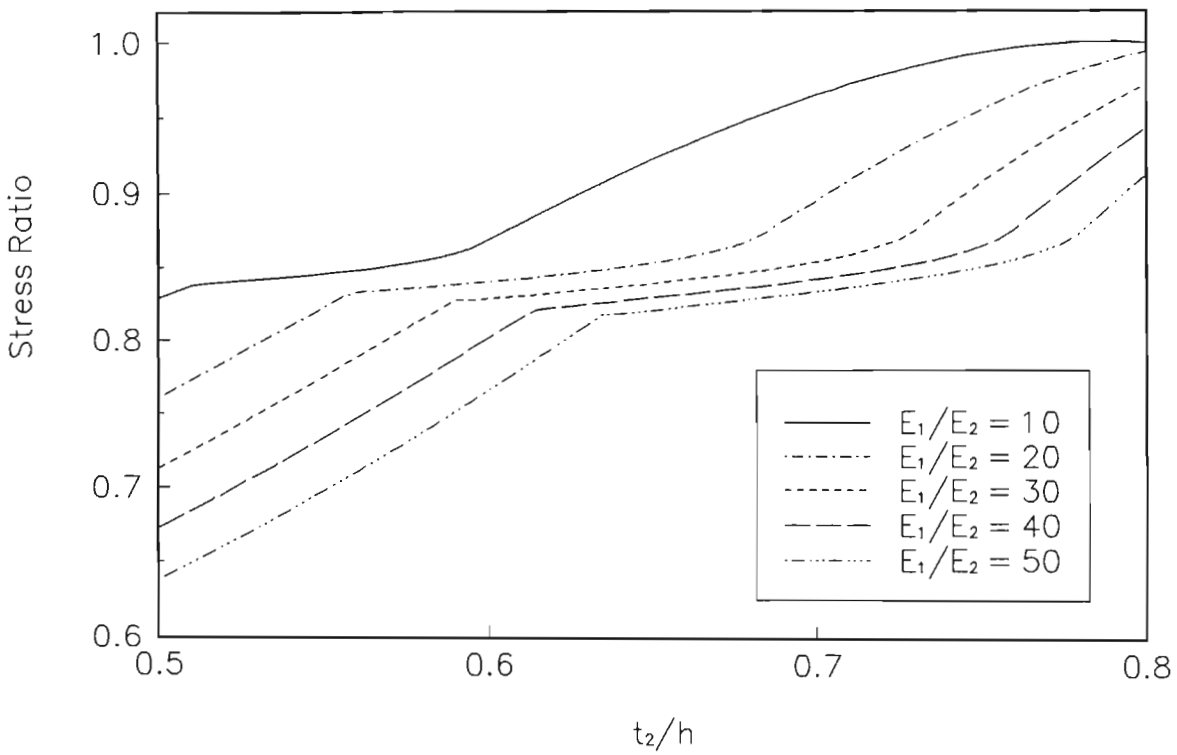


Figure 5.17b. Stress ratio η vs core thickness t_2/h with $a/h = 3$ and $E_2'/E_2 = 1$

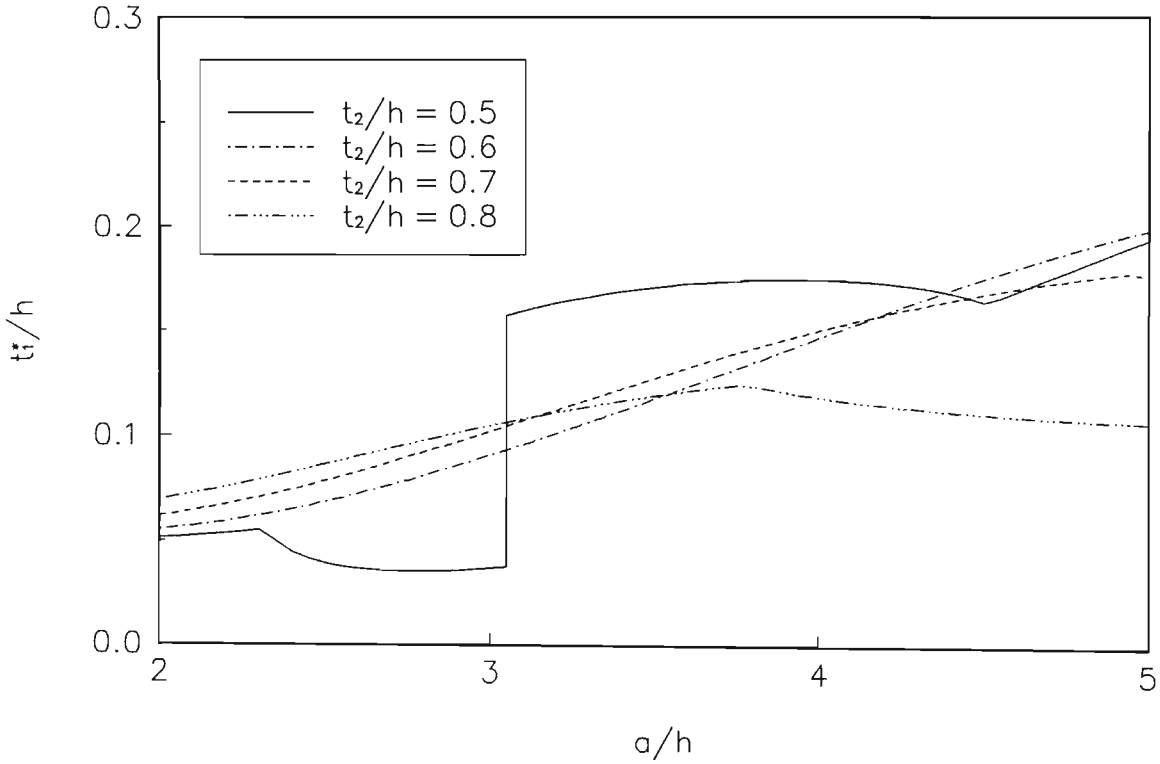


Figure 5.18a. Optimal thickness t_1^*/h vs a/h
with $E_1/E_2 = 10$ and $E_2'/E_2 = 1$

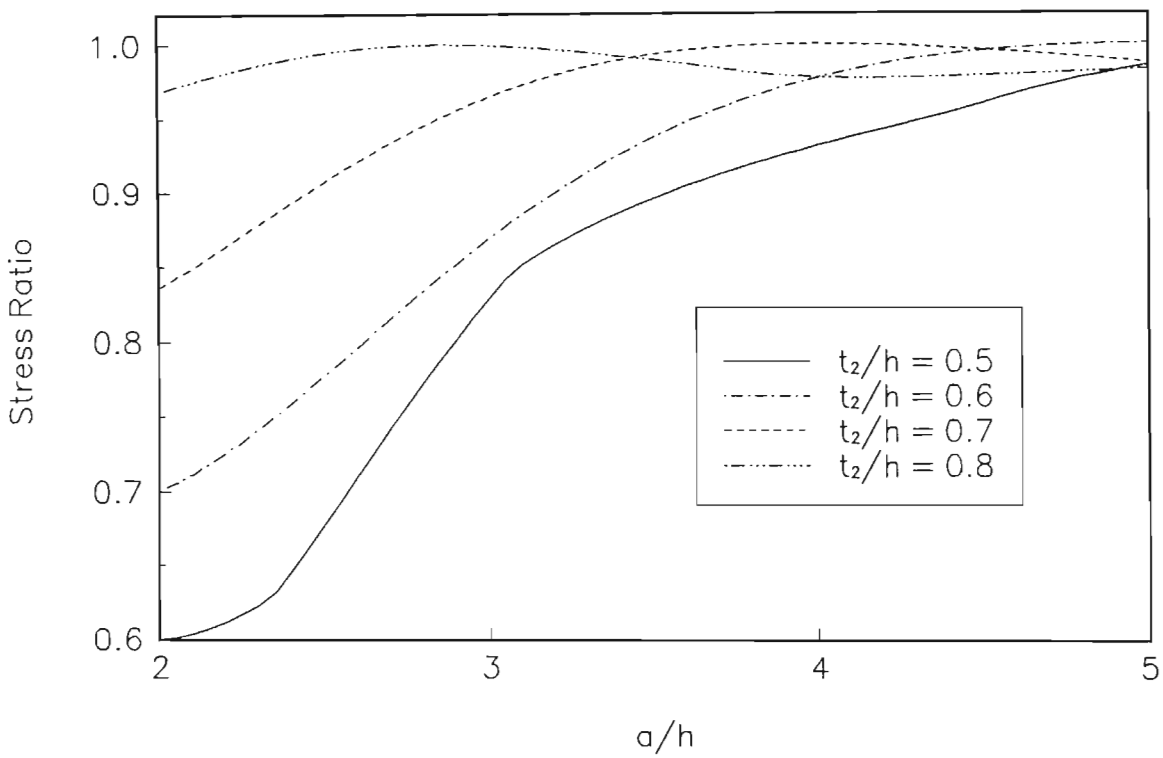


Figure 5.18b. Stress ratio η vs a/h
with $E_1/E_2 = 10$ and $E_2'/E_2 = 1$

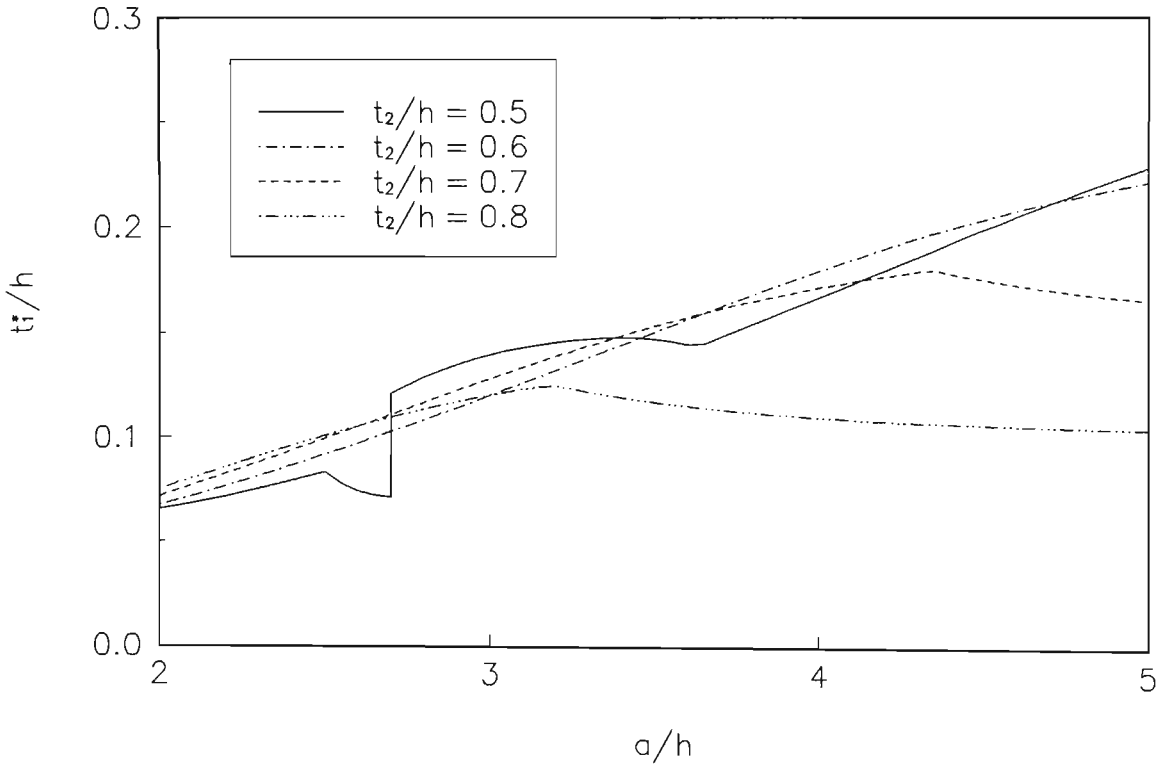


Figure 5.19a. Optimal thickness t_1^*/h vs a/h
with $E_1/E_2 = 100$ and $E_2'/E_2 = 10$

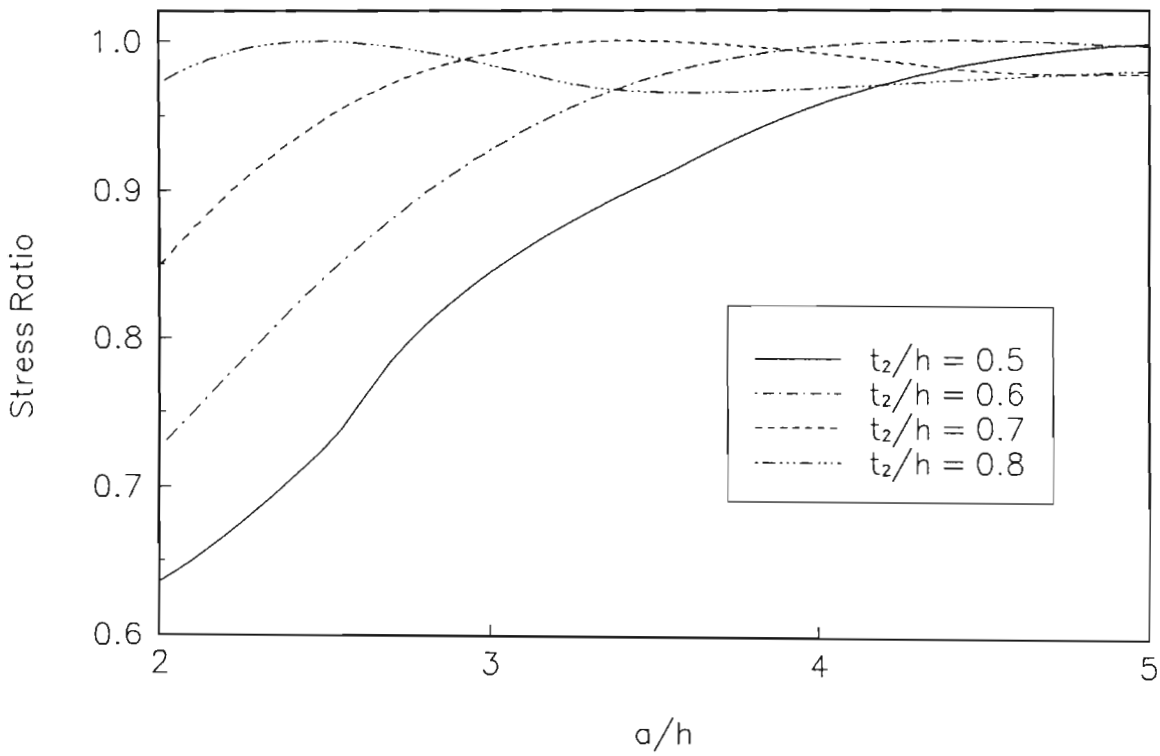


Figure 5.19b. Stress ratio η vs a/h
with $E_1/E_2 = 100$ and $E_2'/E_2 = 10$

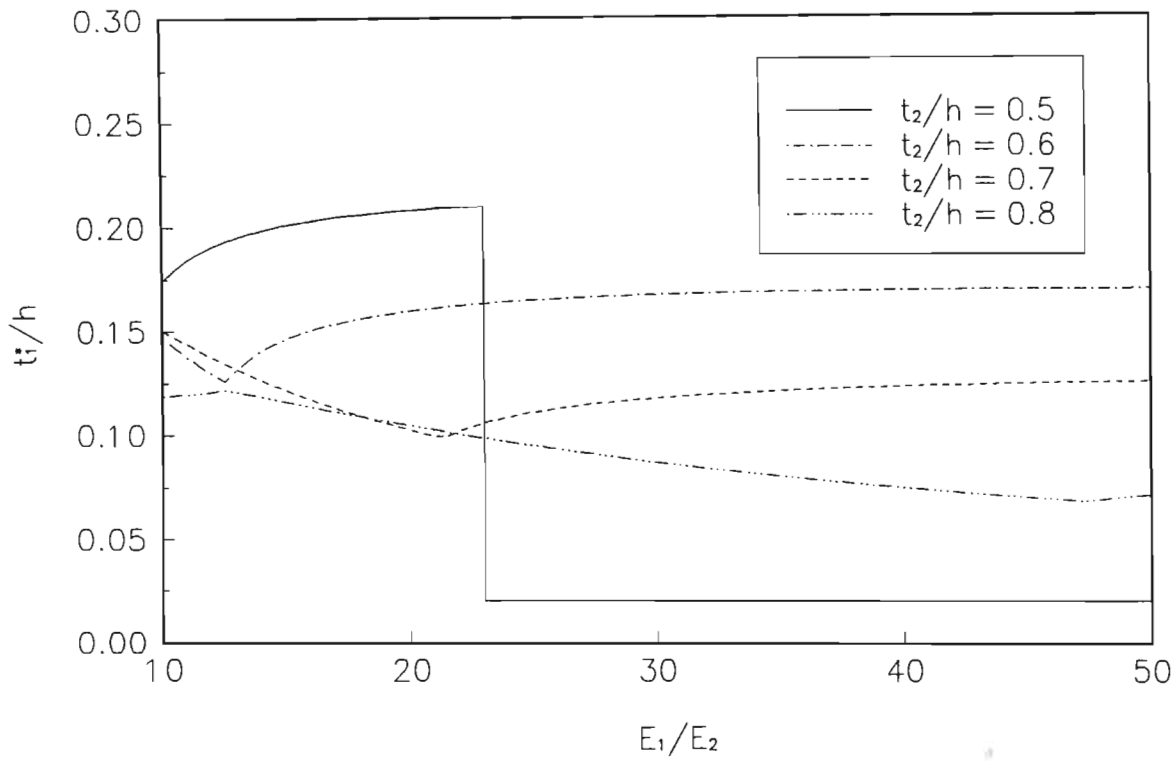


Figure 5.20a. Optimal thickness t_1^*/h vs E_1/E_2 with $a/h = 4$ and $E_2'/E_2 = 1$

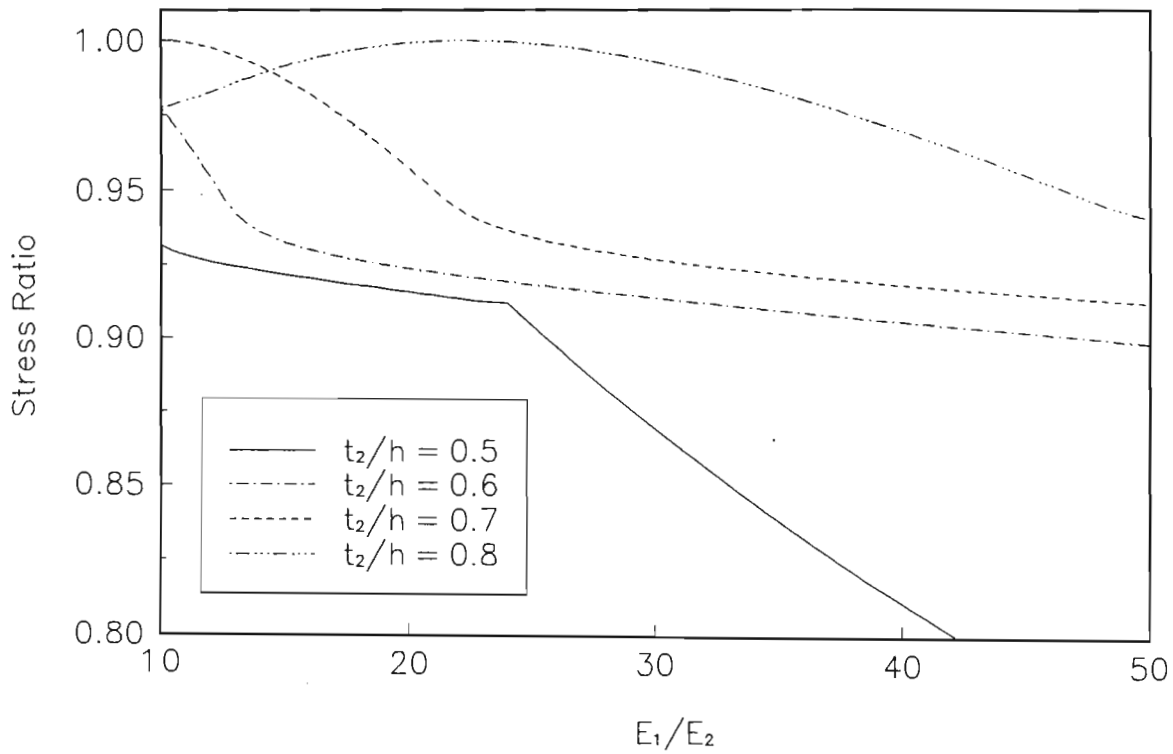


Figure 5.20b. Stress ratio η vs E_1/E_2 with $a/h = 4$ and $E_2'/E_2 = 1$

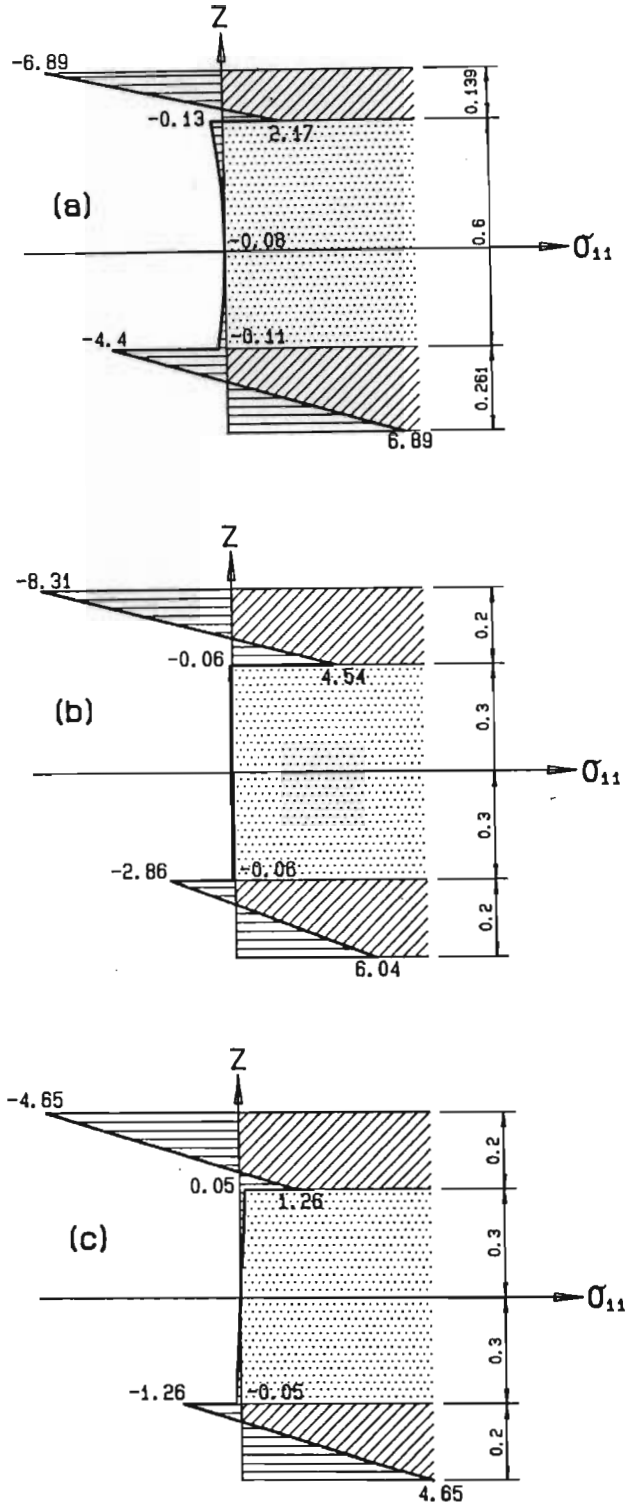


Figure 5.21. Stress distributions for $a/h = 3$, $t_2/h = 0.6$, $E_1/E_2 = 20$, $E_2'/E_2 = 1$ for (a) Optimal design ($t_1^*/h = 0.139$), (b) Symmetrically laminated plate, and (c) Symmetrically laminated plate without normal deformation

5.6 Conclusions

Optimal designs of thick sandwich plates are given using the higher–order theory, which includes the effects of normal deformation as well as transverse shear deformation. The higher–order theory can provide accurate solutions for very thick laminates with layers having significantly different mechanical properties. Therefore this approach makes it possible to optimize sandwich plates with thickness ratio $a/h \leq 5$.

The higher–order theory is implemented using symbolic computation in preference to an analytical or numerical method, since deriving the necessary analytical expressions would be extremely cumbersome while a numerical method would to some extent nullify the accuracy afforded by the higher–order theory. In order to operate at the highest possible computational efficiency, special purpose symbolic computation routines are developed and incorporated into an optimization algorithm. These routines bypass some of the symbolic processing which makes the routines developed in Section 4.5 relatively more flexible, and as such can achieve a higher degree of efficiency. It is found that the use of a general purpose symbolic computation system for the present optimization studies is infeasible as it leads to excessive computer time.

When normal deformation is considered, the through–the–thickness distribution of the normal stresses ceases to be symmetrical in the case of unsymmetrical loading, i.e. loading loading on one surface of the plate only, and conventional symmetrical designs are non–optimal. Moreover, normal deformation results in the bottom surface having a minimum deflection at a certain thickness ratio of surface and core layers and the minimum deflection design is based on this observation. Another interesting observation is the existence of negative deflection of the bottom surface at certain stiffness ratios. This phenomenon arises as a result of Poisson's effect on the normal deformation.

Three design problems are formulated and solved, namely optimal designs for minimum weight, minimum deflection and min–max normal stress. The minimum weight problem involves the determination of the fibre content of transversely isotropic surface layers made of a randomly oriented fibre composite material. The other design variable is taken as the core layer thickness. Minimum weight designs subject to a deflection constraint are obtained and the effects of problem parameters on the design variables and weight are studied. It is observed that fibre content decreases as the sandwich thickness increases for a given deflection.

In the minimum deflection problem, the thicknesses of the surface layers are chosen as the design variables. The effect of the relative stiffness E_1/E_2 of the surface and core layers is investigated and it is observed that surface layer thickness is smaller if the core stiffness E_2 is low due to more of the deflection being absorbed by a softer core region. However, surface layer thickness does not show a definite trend as the through–the–thickness stiffness E_1' of surface layers decreases. This thickness decreases for low E_1/E_2 ratios and increases for high E_1/E_2 ratios as E_1' decreases.

The optimal design for minimum stress involves the computation of the relative thicknesses of the layers such that the maximum normal stress will be minimized. The resulting design leads to vastly improved sandwich plates which have normal stresses up to 40% less than those of a symmetrical sandwich. However, the efficiency drops for a transversely isotropic core as compared to an isotropic core. It is observed that the thicknesses of the top and bottom layers of an optimal plate depend on the relative thickness of the core layer, the thickness ratio of the plate and on the relative stiffness of the core layer as compared to the surface layers. It emerges that although the top layer on which the load acts is in general thinner than the bottom layer for a minimum stress design, this is not always the case and the bottom layer can also be thinner depending on the geometric and material parameters. In particular this is the case if the core layer is thick compared to the surface layers. It is demonstrated that normal deformation has a substantial effect on the design and the effect of normal deformation becomes more pronounced when the plate becomes thicker.

Deflection and stress profiles through the thickness are studied with a view towards assessing the effect of the configuration on the bottom surface deflection and maximum normal stress and the effect of neglecting normal deformation. It is observed that if the theory used in the design analysis does not include normal deformation, the design of thick structures becomes meaningless as it leads to non–optimal sandwich plates.

Chapter 6

Conclusions

6.1 Overview

In the present study the design optimization of a suite of laminated composite structures is performed. In the first instance thin laminated composite structures are optimized, in particular balanced and unbalanced laminated composite pressure vessels with specially orthotropic layers whose elastic properties depend on the angle of the reinforcing fibres.

Special purpose symbolic computation routines are developed in a conventional programming language in order to improve the efficiency of the optimization algorithm. These routines combine the relationship between the loading parameters and the stress into one transformation matrix and thereby reduce the number of calculations required in each iteration of the optimization algorithm. It is found that the development of such routines, which are dedicated to a specific class of functions, is a feasible objective.

A new higher–order theory which includes the effects of both transverse shear and normal deformation is developed for the analysis and design optimization of thick laminated composite structures with transversely isotropic layers. The form of the kinematic hypotheses of the theory are derived using an iterative technique where the classical Kirchhoff–Love hypotheses are assumed in the first iteration and new variables which have a clear physical meaning are introduced. The governing differential equations contain stiffness constants which are integrals through the thickness of the laminate. The unknown functions in the governing equations are defined on an arbitrary reference surface and the order of the system of governing equations is

independent of the number of layers. The stress/strain state of a laminated structure is determined using the kinematic hypotheses which include the solution of the governing equations (for the reference surface) and through–the–thickness distribution functions defined by the theory.

The distribution functions and integrated stiffnesses of the theory are multiple piecewise integrals through the thickness of the laminate. In the general case, these integrals are not able to be derived in a form suitable for exact numerical calculations and the evaluation of these integrals using a numerical method detracts from the accuracy afforded by the higher–order theory. These difficulties are overcome by employing symbolic computation to derive the distribution functions, calculate the integrated stiffness constants exactly, and derive the stress and strain distributions through the thickness in power series form, for a given laminate.

In addition to using a general purpose symbolic computation system, special purpose symbolic computation routines are developed for the implementation of the theory. These routines are found to be at least two orders of magnitude more efficient in the performance of the required computations than the general purpose system. Therefore in optimization studies, where computational efficiency is of paramount importance, the development of special purpose symbolic computation software is preferable to using a general purpose symbolic computation system. Moreover, special purpose symbolic computation routines are easily incorporated into an optimization algorithm whereas this is not possible using a closed general purpose system.

The numerical results obtained indicate that the new higher–order theory is accurate for thick structures whereas, in general, the shear deformable theory and the classical theory are inaccurate. Therefore in the analysis of thick structures, not only transverse shear but also normal deformation should be taken into account.

Sandwich plates of thickness ratio $a/h \leq 5$ are optimized for minimum weight, minimum deflection and minimum stress on the basis of the higher–order theory. Such designs are not possible using a theory which neglects normal deformation.

6.2 Symbolic Computation

Special purpose symbolic computation software is developed in the C language for the transformation of coordinate axes, failure analysis and the calculation of design

sensitivities for fibre reinforced composite structures. The symbolic computation routines perform tedious matrix algebra where the entries of the matrices are series of double trigonometric functions and simplify the results using trigonometric identities. The symbolic computations are integrated into an optimization algorithm resulting in a combined symbolic and numerical approach to determine the optimal design.

Special purpose symbolic computation routines are also developed for the implementation of the higher–order theory. Symbols may be defined as laminae parameters, symbolic expressions or piecewise integrals of symbolic expressions. Symbols and symbolic expressions are recursively expanded into power series which may be integrated and evaluated.

6.3 Optimization of Thin Pressure Vessels

Two design problems for thin laminated composite pressure vessels are solved. In the first problem, a cylindrical pressure vessel is optimized taking the fibre angle as the design variable to maximize the burst pressure and the effects of the axial force and torque on the optimal designs are investigated. In the second problem, a cylindrical vessel filled with liquid and subject to an internal pressure is studied. The weight of the shell is minimized taking the fibre angle and the wall thickness as the design variables. Both constant and variable thickness shells are investigated. It is shown that the results for the second problem approach those of the first problem as the internal pressure increases.

Numerical results are given for unbalanced (single– and six–layered) and balanced laminates noting that in the balanced case the number of layers does not affect the results. It is observed that fibre angles align themselves with the longitudinal axis as the axial force increases. Variable thickness shells are found to be about 20% more efficient than the constant thickness shells for low values of the internal pressure with the difference decreasing as this pressure increases. For single–layered pressure vessels, the optimal fibre angle is found to be either 0° or 90° with the switch-over point depending on the magnitude of the axial force, torque and the internal pressure.

6.4 Higher–Order Theory

The numerical results obtained for thick homogeneous and heterogeneous plates are compared to those given in the literature to validate the higher–order theory. It is found that for an isotropic plate with thickness ratio $a/h = 2$, the higher–order theory predicts the deflection distribution to within 2% and the normal stresses to within 5% of the exact three–dimensional elasticity solution whereas the shear deformable model which neglects the effect of normal deformation and the classical theory are grossly inaccurate.

As the transverse rigidity of plates decreases, the effect of normal deformation becomes more pronounced and the inaccuracy of the shear–deformable model increases. However, it is observed that as the ratio a/h increases, the effect of normal deformation is reduced and, in general, the shear–deformable theory is accurate for plates with $a/h \geq 10$.

Numerical results for a sandwich plate where the higher–order theory is used to model the core layer and the classical theory is used to model the surface layers are compared to those given in the literature where the core layer is modelled as a three–dimensional elastic body. It is found that the discrepancies are at most 0.8% for $a/h = 3, \ldots, 10$ and a core layer up to two orders of magnitude weaker than the surface layers.

6.5 Optimization of Thick Sandwich Plates

Three design problems are formulated and solved, namely, optimal designs for minimum weight, minimum deflection and minimum normal stress. The minimum weight problem involves the determination of the optimal thickness of the core layer and the optimal fibre content of the surface layers. It is observed that fibre content decreases as the sandwich thickness increases for a given deflection constraint.

In the minimum deflection problem, the thicknesses of the surface layers are chosen as the design variables. The design optimization is based on the observation that the deflection of the bottom surface decreases as the core layer becomes thinner but increases again once the thickness of the core layer relative to the surface layers drops below a critical level. Moreover, the bottom surface may undergo negative deflection for certain combinations of material properties. These phenomena

are peculiar to thick structures and can only be analysed using three–dimensional elasticity solutions or a higher–order theory which includes normal deformation.

The optimal design for minimum stress involves the computation of the relative thicknesses of the layers such that the maximum normal stress will be minimized. It is demonstrated that normal deformation has a substantial effect on the design and this effect becomes more pronounced as the thickness of the plate increases. When normal deformation is taken into account, the maximum normal stresses in the top and bottom layers are not equal in the case of a symmetrical sandwich. The design optimization produces a configuration where these maximum stresses are equal and leads to vastly improved sandwich plates which have normal stresses up to 40% less than those of a symmetrical sandwich. It is found that if the theory used in the design analysis does not include normal deformation, the design of thick structures becomes meaningless as it leads to non–optimal sandwich plates.

6.6 Recommendations for Future Work

Higher–order theory for orthotropic materials

The present higher–order theory could be extended to orthotropic materials. This would enable the optimization of composite structures reinforced with continuous fibres where the fibre orientation is a design variable. The boundary conditions could be extended to cylindrical shells for the optimization of pressure vessels.

Object–oriented symbolic computation

The development of special purpose symbolic computation routines lends itself towards an object–oriented approach and such routines could be developed using the C++ language for various applications.

Design optimization of thick structures

A three–dimensional failure criterion could be used as a design constraint for the optimization studies of thick structures based on the higher–order theory.

The optimization studies could be extended to structures under other loading conditions such as uniformly distributed and localized loads and under various boundary conditions.

Bibliography

[1] Pipkin, A. C., & Rivlin, R. S., Minimum–weight design for pressure vessels reinforced with inextensible fibres. *J. Applied Mechanics,* **30** (1963) 103–108.

[2] Chao, C. C., Sun, C. T., & Koh, S. L., Strength optimization for cylindrical shells of laminated composites, *J. Composite Materials,* **9** (1975) 55–66.

[3] Urazgil'dyaev, K. U., Design of a equal–strength cylindrical shells of composite material. *Prikladnaya Mekhanika,* **13** (7) (1977) 121–124.

[4] Tauchert, T. R., Optimum design of a reinforced cylindrical pressure vessel. *J. Composite Materials,* **15** (1981) 390–402.

[5] Fukunaga, H., & Uemura, M., Optimum design of helically wound composite pressure vessels. *J. Composite Structures,* **1** (1983) 31–49.

[6] Eckold, G. C., A design method for filament wound GRP vessels and pipework. *Composites,* **16** (1) (1985) 41–47.

[7] Fukunaga, H., & Chou, T. W., Simplified design techniques for laminated cylindrical pressure vessels under stiffness and strength constraints. *J. Composite Materials,* **22** (1988) 1157–1169.

[8] Karandikar, H., Srinivasan, R., Mistree, F., & Fuchs, W. J., Compromise: An effective approach for the design of pressure vessels using composite materials. *Computers & Structures,* **33** (6) (1989) 1465–1477.

[9] Adali, S., & Yakar, B., Optimal design of laminated pressure vessels under internal pressure and temperature loading. In *Pressure Vessels and Components 1991,* Proc. of the 1991 Pressure Vessels and Piping Conference, PVP–Vol. 217, ASME, New York (1991) 57–63.

[10] Adali, S., Fuzzy optimization of laminated cylindrical pressure vessels. In *Composite Structures 6,* Proc. of the 6th International Conference on Composite Structures, Elsevier Applied Science, London (1991) 249–259.

[11] Datoo, M. H., *Mechanics of Fibrous Composites.* Elsevier Applied Science, London, 1991.

[12] Tsai, S. W., & Hahn, W. T., *Introduction to Composite Materials.* Technomic Publishing Company, Westport, Connecticut, 1980.

[13] Flügge, W., *Stresses in Shells.* 2nd Edition, Springer–Verlag, New York, 1973.

[14] Kraus, H., *Thin Elastic Shells.* John Wiley & Sons, New York, 1967.

[15] Beltzer, A. I. , Engineering analysis via symbolic computation – a breakthrough. *Appl. Mech. Rev.,* **43 (6)** (1990) 119–127.

[16] Crespo da Silva, M. R. M., & Hodges, D. H., The role of computerized symbolic manipulation in rotorcraft dynamic analysis. *Comp. Math. Appl.,* **12A** (1986) 161–172.

[17] Elishakoff, I., & Halkamp, J., Computerized symbolic solution for a monconservative system in which instability occurs by flutter in one range of the parameter and by divergence in the other. *Computer Meth. Appl. Mech. Eng.,* **62** (1987) 27–46.

[18] Elishakoff, I., & Couch, B., Application of computerized symbolic algebra for instability of nonconservative systems. *J. Symbolic Comp.,* **4** (1987) 391–396.

[19] Elishakoff, I., & Tang, J., Buckling of polar orthotropic circular plates on elastic foundation by computerized symbolic algebra. *Computer Meth. Appl. Mech. Eng.,* **68** (1988) 229–247.

[20] Elishakoff, I., & Pletner, B., Computerized symbolic algebraic evaluation of buckling loads by Snitko's method. *Computer Meth. Appl. Mech. Eng.,* **88** (1991) 299–309.

[21] Graaf, E., & Springer, G. S., Stress analysis of thick curved composite laminates. *Computers and Structures,* **38 (1)** (1991) 41–55.

[22] Nomura, S., & Wang, B. P., Free vibration of composite plates using symbolic algebra. *Advanced Composite Materials,* **1** (1991) 87–92.

[23] Wolfram, S., *Mathematica: A System for Doing Mathematics by Computer.* Addison–Wesley (1988).

[24] Summers, E. B., Adali S., & Verijenko, V. E., Special purpose symbolic computation software with application to the optimization of composite laminates. *Advances in Engineering Software,* **18** (1993) 199–209.

[25] Dudchenko, A. A., Lur'e, S. A., & Obraztcov, I. F., Anisotropic multilayer plates and shells. Summary of science and engineering. *Mechanica Deformiruemogo Tela,* **15** (1983) 3–68 (in Russian).

[26] Librescu, L., & Reddy, J. N., A few remarks concerning several refined theories of anisotropic composite laminated plates. *Intl. J. Engrg. Sci.,* **27** (5) (1988) 515–527.

[27] Reddy, J. N., A review of refined theories of laminated composite plates. *Shock Vib. Dig.,* **22** (7) (1990) 3–17.

[28] Noor, A. K., & Burton, W. S., Assessment of computational models for multilayered composite shells. *Appl. Mech. Rev.,* **43** (4) (1990) 67–97.

[29] Bolotin, V. V., & Novichkov, U. N., *Mechanics of Multilayered Structures.* Mashinostroenie, Moscow, 1980 (in Russian).

[30] Grigorenko, J. M., & Vasilenko, A. T., *Methods for Shell Analysis. Volume 4. Theory for Shells with Variable Stiffnesses.* Naukova Dumka, Kiev, 1981 (in Russian).

[31] Piskunov, V. G., & Verijenko, V. E., *Linear and Non-linear Problems in the Analysis of Laminated Structures.* Budivelnik, Kiev, 1986 (in Russian).

[32] Reissner, E., On bending of elastic plates. *Quart. Appl. Mech.,* **5** (1) (1947) 55–68.

[33] Mindlin, R.D., Influence of rotatory inertia and shear on flexural motions of isotropic elastic plates. *J. Appl. Mech., Trans. ASME.,* **18** (1951) 31–38.

[34] Gol'denveizer, A. L., Derivation of an approximate theory of bending of plates by the method of assymptotic integration of the equations of the theory of elasticity. *PMM,* **26** (4) (1963) 1000–1025.

[35] Reddy, J. N., A general nonlinear third–order theory of plates with moderate thickness. *J. Non-Linear Mechanics,* **25** (6) (1990) 677–686.

[36] Ambartsumyan, S. A., Some Basic Equations for a Theory of Thin Laminated Shells. *Dokl. Akad. Nauk ASSR,* **8** (5) (1948) 203—210 (in Russian).

[37] Ambartsumyan, S. A., *General Theory of Anisotropic Shells.* Nauka, Moscow. 446p (1974) (in Russian).

[38] Vasilenko, A. T., & Savchenko, P. I., Axisymmetric deformation of layered anisotropic shells of revolution with allowance for transverse shear strains and contraction. *Prikladnaya Mekhanika,* **24** (5) (1988) 499–504.

[39] Grigorenko, J. M., Vasilenko, A. T., & Golub, G. P., Computing the stressed state of orthotropic elements of shell designs with allowance for transverse shear and reduction. *Mechanics of Composite Materials,* **22** (3) (1986) 338–343.

[40] Stein, M., Nonlinear theory for plates and shells including the effects of transverse shearing. *AIAA J.,* **24** (9) (1986) 1537–1544.

[41] Reddy, J. N., On refined theories of composite laminates. *Meccanica,* **25** (4) (1990) 230–238.

[42] Piskunov, V. G., Variant of the nonclassical theory of multilayered shallow shells and plates. *Soviet Appl. Mech.,* **15** (11) (1979) 1073–1078.

[43] Rasskazov, A. O., Theory of multilaminate orthotropic sloping shells. *Soviet Appl. Mech.,* **12** (11) (1976) 50–56.

[44] Piskunov, V. G., Verijenko, V. E., Adali, S., & Summers, E. B., A higher–order theory for the analysis of laminated plates and shells with shear and normal deformation. *Intl. J. Engrg. Sci.,* **31** (6) (1993) 967–988.

[45] Verijenko, V. E., Nonclassical theory of elasto-plastic strain of laminated transversely isotropic tapered shells. *Strength of Materials,* **19** (4) (1987) 547–553.

[46] Verijenko, V. E., & Piskunov, V. G., Geometrically nonlinear theory of shallow multilayered orthotropic shells. *Strength of Materials,* **21** (1) (1989) 8–16.

[47] Verijenko, V. E., Adali, S., & Summers, E. B., Nonlinear analysis of laminated composite plates and shells including the effects of shear and normal deformation. *Composite Structures,* **25** (1993) 173–185.

[48] Blocki, J., A higher order linear theory for isotropic plates. I. Theorectical considerations. *Intl. J. Solids & Struct.,* **29** (1992) 825–836.

[49] Librescu, L., & Schmidt, R., Substantiation of a shear deformable theory of anisotropic composite laminated shells accounting for the interlaminae continuity conditions. *Intl. J. Engr. Sci.,* **29** (1991) 669–683.

[50] Tessler, A., & Saether, E., A computationally viable higher order theory for laminated composite plates. *Intl. J. Num. Meth. Engr.,* **31** (1991) 1069–1086.

[51] Novozhilov, V. V., *Theory of Thin Shells.* Gossouzizdatsudprom, Leningrad, 1962 (in Russian).

[52] Vlasov, B. F., About one case of bending a rectangular thick plate. *Vest. Moscow University.* **2** (1957) 22–34 (in Russian).

[53] Brukker, L. E., Some variants of the equations for three-layered plates. *Analysis of Aerospace Structures.* **3** (1965) 74–99 (in Russian).

[54] Bhimaraddi, A., Static and transient response of rectangular plates. *Thin–Walled Struct.,* **5** (1987) 125–143.

[55] Lo, K. H., Christensen, R. M., & Wu, E. M., A higher order theory of plate deformation. Part 2. Laminated plates. *J. Appl. Mech., Trans. ASME.,* **44** (4) (1977) 669–676.

[56] Piskunov, V. G., Verijenko, V. E., & Prisyazhnyuk, V. K., *Calculation of Inhomogeneous Shells and Plates using Finite Element Methods.* Veisha Shkola, Kiev, 1987 (in Russian).

[57] Verijenko, V. E., & Prisyazhnyuk, V. K., Linear and nonlinear models of contact interaction of laminated constructions with elastic basement. *Proceedings of International Congress IKM-X,* Weimar, Germany, **6** (1984) 175–182.

[58] Verijenko, V. E., Adali, S., & Summers, E. B., Discrete–continuous scheme of FEM for analysing laminated composite shells and plates. *Proceedings of FEMSA92,* Cape Town, South Africa (1992) 589–600.

[59] Voyiadjis, G. Z., & Baluch M. H., Refined theory for thick composite plates. *ASME J. Engr. Mech.,* **114** (4) (1988) 671–687.

[60] Verijenko, V. E., Adali, S., & Summers, E. B., Applications of refined theory in the nonlinear analysis of composite laminated structures. *Computational Mechanics,* **14** (1994) 52–67.

[61] Summers, E. B., & Verijenko, V. E., Symbolic computation in numerical analysis of composite plates and shells. *Proc. 18th SANUM Symposium,* Durban, South Africa (1992).

[62] Verijenko, V. E., Adali, S., & Summers, E. B., Investigation of the convergency of numerical methods applied to nonlinear problems in the mechanics of composites. *Proc. 18th SANUM Symposium,* Durban, South Africa (1992).

[63] Adali, S., Summers, E. B., & Verijenko, V. E., Optimization of laminated cylindrical pressure vessels under strength criterion. *Composite Structures,* **25** (1993) 305–312.

[64] Verijenko, V. E., Summers, E. B., & Adali, S., Symbolic computation for the optimization of composite structures using higher order theory. *International Congress on Computer Systems and Applied Mathematics,* St Petersburg, Russia, July 1993 (Abstract).

[65] Summers, E. B., Verijenko, V. E., & Adali, S., The development of symbolic computation routines using the C programming language. *International Congress on Computer Systems and Applied Mathematics,* St Petersburg, Russia, July 1993 (Abstract).

[66] Verijenko, V. E., Adali, S., Stewart, R. W., & Summers, E. B., Application of the method of initial functions to the analysis of anisotropic plates. *19th South African Symposium on Numerical Methods,* San Lameer, South Africa, July 1993 (Abstract).

[67] Pandya, B. N., & Kant, T., Higher–order shear deformable theories for flexure of sandwich plates – Finite element evaluations. *Int. J. Solids Structures*, **24** (12) (1988) 1267–1286.

[68] Gordaninejad, F., & Bert, C. W., A new theory for bending of thick sandwich beams. *Int J. Mech. Sciences*, **11/12** (1989) 925–934.

[69] Bert, C. W., & Cho, K. N., Uniaxial compressive and shear buckling in orthotropic sandwich plates by an improved theory. *Mechanics of Structures and Machines*, **17** (3) (1989) 283–302.

[70] Huang, S. N., & Alspaugh, D. W., Minimum weight sandwich beam design. *AIAA Journal*, **12** (12) (1974) 1617–1618.

[71] Vinson, J. R., Optimum design of composite honeycomb sandwich panels subjected to uniaxial compression. *AIAA Journal*, **24** (10) (1986) 1690–1696.

[72] Ding, Y. Optimum design of honeycomb sandwich constructions with buckling constraints. *Computers and Structures*, **33** (6) (1989) 1355–1364.

[73] Martin, P. M. J. W., Optimum design of anisotropic sandwich panels with thin faces. *Engineering Optimization.* **11** (1987) 3–12.

[74] Makris, S. E., MacGregor Smith, J., & Dym, C. L., Multiobjective optimization of acoustic sandwich panels. *Engineering Optimization*, **13** (1988) 147–172.

[75] Min, K. T., & de Charentenay, F. X., Optimum weight design of sandwich cylinders with orthotropic facings and core under combined loads. *Computers and Structures*, **24** (2) (1986) 313–322.

[76] Evans, K. E., The design of doubly curved sandwich panels with honeycomb cores. *Composite Structures*, **17** (1991) 95–111.

[77] Adali, S., Richter, A., & Verijenko, V. E., Non–probabilistic modelling and design of sandwich plates subject to uncertain loads and initial imperfections. Submitted to *Int. J. Engineering Science.*

[78] Adali, S., Design of shear–deformable antisymmetric angle–ply laminates to maximize the fundamental frequency and frequency separation. *Composite Structures*, **2** (1984) 349–369.

[79] Kam, T. Y., & Chang, R. R., Design of laminated composite plates for maximum buckling load and vibration frequency. *Computer Methods in Applied Mechanics and Engineering*, **106** (1993) 65–81.

[80] Kam, T. Y., & Chang, R. R., Buckling of shear deformable laminated composite plates. *Composite Structures*, **22** (1992) 223–234.

[81] Kam, T. Y., & Chang, R. R., Optimum layup of thick laminated composite plates for maximum stiffness. *Engineering Optimization*, **19** (1992) 237–249.

[82] Berlin, A. A., Volfson, S. A., Enikolopian, N. S., & Negmatov, S. S., *Principles of Polymer Composites.* Springer–Verlag, Heidelberg, 1986.

[83] Chamis, C. C., Simplified composite micromechanics equations for mechanical, thermal and moisture–related properties. In *Engineers' Guide to Composite Materials*, edited by J. W. Weeton, D. M. Peters & K. L. Thomas. American Society for Metals, Metals Park, Ohio, 1987, pp. 3/8–3/24.

[84] Plantema, F. J., *Sandwich Construction.* John Wiley & Sons, New York, 1966.

[85] Adali, S., Summers, E. B., & Verijenko, V. E., Minimum weight and deflection design of thick sandwich laminates via symbolic computation. To appear in *Composite Structures.*

[86] Verijenko, V. E., Adali, S., Summers, E. B., & Reiss, T., Optimal design of laminated shells with shear and normal deformation using symbolic computation. *Proceedings of 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization,* Panama City Beach, Florida, September 1994.

[87] Verijenko, V. E., Summers, E. B., & Adali, S., Minimum stress design of transversely isotropic sandwich plates based on higher–order theory. Submitted to *Composites Engineering.*

[88] Adali, S., Richter, A., Summers, E. B., & Verijenko, V. E., Optimal design of hybrid laminates with discrete ply–angles for minimum buckling load. (In preparation.)

Appendix A

Symbolic Results from Mathematica

For transversely isotropic homogeneous shells, the *Mathematica* implementation of higher–order theory derives the distribution functions as

```
f1[z_] = -(e1 h^2)/(8 (1 - nu1^2)) + (e1 z^2)/(2 (1 - nu1^2))
beta1[z_] = (e1 (h - 2 z) (h + 2 z))/(8 g2 (-1 + nu1) (1 + nu1))
beta7[z_] = -(h - 2 z)/(2 g2 h)
beta8[z_] = (h + 2 z)/(2 g2 h)
alp1[z_] = (e1 nu2 z)/(e2 (1 - nu1))
alp2[z_] = -((e1 nu2)/(e2 (1 - nu1)))
alp3[z_] = ((h - 2 z)^2 (h + 2 z))/(8 e2 h^2)
alp4[z_] = ((h - 2 z) (h + 2 z)^2)/(8 e2 h^2)
alp5[z_] = -((h - 2 z)^2 (h + z))/(2 e2 h^3)
alp6[z_] = ((h - z) (h + 2 z)^2)/(2 e2 h^3)
vphi[z_] = (-(e1 h^2)/(8 (1 - nu1^2)) + (e1 z^2)/(2 (1 - nu1^2)))/g2
vphi1[z_] = -(e1 nu2 z^2)/(2 e2 (-1 + nu1))
vphi2[z_] = (e1 nu2 z)/(e2 (-1 + nu1))
vphi3[z_] = (z (3 h^3 - 3 h^2 z - 4 h z^2 + 6 z^3))/(24 e2 h^2)
vphi4[z_] = -(z (-3 h^3 - 3 h^2 z + 4 h z^2 + 6 z^3))/(24 e2 h^2)
vphi5[z_] = -(z (2 h^3 - 3 h^2 z + 2 z^3))/(4 e2 h^3)
vphi6[z_] = -(z (-2 h^3 - 3 h^2 z + 2 z^3))/(4 e2 h^3)
vphi7[z_] = (1 - (1 - nu1^2) ((e1 h)/(2 (1 - nu1^2)) +
              (e1 z)/(1 - nu1^2))/(e1 h))/g2
vphi8[z_] = (1 - nu1^2) ((e1 h)/(2 (1 - nu1^2)) +
              (e1 z)/(1 - nu1^2))/(e1 g2 h)
psi1[z_] = (e1 z (-3 e2 h^2 + 4 e2 z^2 - 4 g2 nu2 z^2 -
              4 g2 nu1 nu2 z^2))/(24 e2 g2 (-1 + nu1) (1 + nu1))
psi2[z_] = (e1 nu2 z^2)/(2 e2 (-1 + nu1))
psi3[z_] = (z^2 (15 h^3 - 10 h^2 z - 10 h z^2 + 12 z^3))/(240 e2 h^2)
psi4[z_] = -(z^2 (-15 h^3 - 10 h^2 z + 10 h z^2 + 12 z^3))/
              (240 e2 h^2)
psi5[z_] = -(z^2 (5 h^3 - 5 h^2 z + 2 z^3))/(20 e2 h^3)
psi6[z_] = -(z^2 (-5 h^3 - 5 h^2 z + 2 z^3))/(20 e2 h^3)
psi7[z_] = -(z (-h + z))/(2 g2 h)
psi8[z_] = -(z (h + z))/(2 g2 h)
```

and the integrated stiffnesses as

```
cb = (e1 h (-e2 + e1 nu2^2))/
        ((1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cbb = -((e1 h (e2 nu1 + e1 nu2^2))/
        ((1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2)))
cb0 = 0
cb1 = 0
cb2 = (e1^2 h^3 nu2 (-e2 + e1 nu2^2))/
         (24 e2 (-1 + nu1) (1 + nu1)
   (-e2 + e2 nu1 + 2 e1 nu2^2))
cb5 = -(e1 h^3 (-e2 + e1 nu2^2))/(48 e2 (1 + nu1)
              (-e2 + e2 nu1 + 2 e1 nu2^2))
cb6 = (e1 h^3 (-e2 + e1 nu2^2))/(48 e2 (1 + nu1)
              (-e2 + e2 nu1 + 2 e1 nu2^2))
cc1 = -(e1^2 h^3 nu2 (-e2 + e1 nu2^2))/
          (24 e2 (-1 + nu1) (1 + nu1)
          (-e2 + e2 nu1 + 2 e1 nu2^2))
cc1b = (e1^2 h^3 nu2 (e2 nu1 + e1 nu2^2))/
          (24 e2 (-1 + nu1) (1 + nu1)
          (-e2 + e2 nu1 + 2 e1 nu2^2))
cc2 = 0
cc2b = 0
cc5 = (9 e1 h^2 (-e2 + e1 nu2^2))/
   (160 e2 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cc5b = (-9 e1 h^2 (e2 nu1 + e1 nu2^2))/
   (160 e2 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cc6 = (9 e1 h^2 (-e2 + e1 nu2^2))/
   (160 e2 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cc6b = (-9 e1 h^2 (e2 nu1 + e1 nu2^2))/
   (160 e2 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
ch1 = 0
ch2 = -((e1^2 h nu2^2)/((-1 + nu1)
              (-e2 + e2 nu1 + 2 e1 nu2^2)))
ch5 = (e1 h nu2)/(2 (-e2 + e2 nu1 + 2 e1 nu2^2))
ch6 = -(e1 h nu2)/(2 (-e2 + e2 nu1 + 2 e1 nu2^2))
ch01 = (e1^2 h^3 nu2^2)/(12 (-1 + nu1)
              (-e2 + e2 nu1 + 2 e1 nu2^2))
ch02 = 0
```

```
ch05 = -(e1 h^2 nu2)/(10 (-e2 + e2 nu1 + 2 e1 nu2^2))
ch06 = -(e1 h^2 nu2)/(10 (-e2 + e2 nu1 + 2 e1 nu2^2))
ch11 = -(e1^3 h^5 nu2^2 (4 e2 + g2 nu2 + g2 nu1 nu2))/
          (480 e2 g2 (-1 + nu1)^2 (1 + nu1)
     (-e2 + e2 nu1 + 2 e1 nu2^2))
ch12 = 0
ch22 = -(e1^3 h^3 nu2^3)/(24 e2 (-1 + nu1)^2
               (-e2 + e2 nu1 + 2 e1 nu2^2))
ch21 = 0
ch16 = (e1^2 h^4 nu2 (17 e2 + 4 g2 nu2 + 4 g2 nu1 nu2))/
          (1680 e2 g2 (-1 + nu1) (1 + nu1)
          (-e2 + e2 nu1 + 2 e1 nu2^2))
ch26 = -(e1^2 h^3 nu2^2)/
          (48 e2 (-1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
ch15 = (e1^2 h^4 nu2 (17 e2 + 4 g2 nu2 + 4 g2 nu1 nu2))/
          (1680 e2 g2 (-1 + nu1) (1 + nu1)
          (-e2 + e2 nu1 + 2 e1 nu2^2))
ch25 = (e1^2 h^3 nu2^2)/
           (48 e2 (-1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cd00 = (e1 h^3 (-e2 + e1 nu2^2))/
           (12 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cd01 = -(e1^2 h^5 (4 e2 + g2 nu2 + g2 nu1 nu2)
           (-e2 + e1 nu2^2))/(480 e2 g2 (-1 + nu1)
       (1 + nu1)^2 (-e2 + e2 nu1 + 2 e1 nu2^2))
cd02 = 0
cd05 = (13 e1 h^4 (-e2 + e1 nu2^2))/
   (4480 e2 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cd06 = (13 e1 h^4 (-e2 + e1 nu2^2))/
   (4480 e2 (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cd1 = (e1^2 h^5)/(120 g2 (-1 + nu1)^2 (1 + nu1)^2)
cd11 = (e1^3 h^7 (-e2 + e1 nu2^2) (68 e2^2 + 32 e2 g2 nu2 +
           32 e2 g2 nu1 nu2 + 5 g2^2 nu2^2 +
           10 g2^2 nu1 nu2^2 + 5 g2^2 nu1^2 nu2^2))/
          (80640 e2^2 g2^2 (-1 + nu1)^2 (1 + nu1)^3
             (-e2 + e2 nu1 + 2 e1 nu2^2))
cd12 = 0
cd21 = 0
cd22 = (e1^3 h^5 nu2^2 (-e2 + e1 nu2^2))/
```

```
          (320 e2^2 (-1 + nu1)^2 (1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
cd16 = -(e1^2 h^6 (268 e2 + 83 g2 nu2 + 83 g2 nu1 nu2)
            (-e2 + e1 nu2^2))/ (967680 e2^2 g2 (-1 + nu1)
                (1 + nu1)^2 (-e2 + e2 nu1 + 2 e1 nu2^2))
cd26 = (e1^2 h^5 nu2 (-e2 + e1 nu2^2))/
            (640 e2^2 (-1 + nu1) (1 + nu1)
             (-e2 + e2 nu1 + 2 e1 nu2^2))
cd15 = -(e1^2 h^6 (268 e2 + 83 g2 nu2 + 83 g2 nu1 nu2)
            (-e2 + e1 nu2^2))/(967680 e2^2 g2 (-1 + nu1)
                (1 + nu1)^2 (-e2 + e2 nu1 + 2 e1 nu2^2))
cd25 = -(e1^2 h^5 nu2 (-e2 + e1 nu2^2))/
   (640 e2^2 (-1 + nu1) (1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2))
cpb12 = 0
cpb21 = 0
cpb22 = -(e1^3 h^3 nu2^3)/(24 e2 (-1 + nu1)^2
            (-e2 + e2 nu1 + 2 e1 nu2^2))
cpb16 = (13 e1^2 h^4 nu2^2)/(4480 e2 (-1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
cpb26 = -(e1^2 h^3 nu2^2)/(48 e2 (-1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
cpb15 = (13 e1^2 h^4 nu2^2)/(4480 e2 (-1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
cpb25 = (e1^2 h^3 nu2^2)/(48 e2 (-1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
crb11 = (e1^2 h^3 nu2^2)/(12 (-1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
crb12 = 0
crb21 = 0
crb22 = (e1^2 h nu2^2)/((-1 + nu1)
              (-e2 + e2 nu1 + 2 e1 nu2^2))
crb16 = -(e1 h^2 nu2)/(10 (-e2 + e2 nu1 + 2 e1 nu2^2))
crb26 = (e1 h nu2)/(2 (-e2 + e2 nu1 + 2 e1 nu2^2))
crb15 = -(e1 h^2 nu2)/(10 (-e2 + e2 nu1 + 2 e1 nu2^2))
crb25 = -(e1 h nu2)/(2 (-e2 + e2 nu1 + 2 e1 nu2^2))
chs11 = (e1^2 h^5)/(120 g2 (-1 + nu1)^2 (1 + nu1)^2) -
            (e1^3 h^5 nu2^2 (4 e2 + g2 nu2 + g2 nu1 nu2))/
            (240 e2 g2 (-1 + nu1)^2 (1 + nu1)
```

```
    (-e2 + e2 nu1 + 2 e1 nu2^2))
chs12 = 0
chs21 = 0
chs22 = -(e1^3 h^3 nu2^3)/(12 e2 (-1 + nu1)^2
          (-e2 + e2 nu1 + 2 e1 nu2^2))
chs16 = (13 e1^2 h^4 nu2^2)/
          (4480 e2 (-1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2)) +
          (e1^2 h^4 nu2 (17 e2 + 4 g2 nu2 + 4 g2 nu1 nu2))/
          (1680 e2 g2 (-1 + nu1) (1 + nu1)
             (-e2 + e2 nu1 + 2 e1 nu2^2))
chs26 = -(e1^2 h^3 nu2^2)/(24 e2 (-1 + nu1)
          (-e2 + e2 nu1 + 2 e1 nu2^2))
chs15 = (13 e1^2 h^4 nu2^2)/
          (4480 e2 (-1 + nu1) (-e2 + e2 nu1 + 2 e1 nu2^2)) +
          (e1^2 h^4 nu2 (17 e2 + 4 g2 nu2 + 4 g2 nu1 nu2))/
          (1680 e2 g2 (-1 + nu1) (1 + nu1)
             (-e2 + e2 nu1 + 2 e1 nu2^2))
chs25 = (e1^2 h^3 nu2^2)/(24 e2 (-1 + nu1)
            (-e2 + e2 nu1 + 2 e1 nu2^2))
crbs16 = -(e1 h^2 nu2)/(8 e2 (-1 + nu1)) +
            (e1 h^2 nu2)/(10 (-e2 + e2 nu1 + 2 e1 nu2^2))
crbs26 = (e1 h nu2)/(2 e2 (-1 + nu1)) -
            (e1 h nu2)/(2 (-e2 + e2 nu1 + 2 e1 nu2^2))
crbs15 = -(e1 h^2 nu2)/(8 e2 (-1 + nu1)) +
            (e1 h^2 nu2)/(10 (-e2 + e2 nu1 + 2 e1 nu2^2))
crbs25 = -(e1 h nu2)/(2 e2 (-1 + nu1)) +
           (e1 h nu2)/(2 (-e2 + e2 nu1 + 2 e1 nu2^2))
```

Appendix B

Routines for Trigonometric Series

```
/*--- composites
 *
 *    evan summers
 *    university of natal
 *    november 1991
 *
 *    symbolic computation for laminates
 *       double trigonometric series
 *
 */

typedef struct /* structure for trig series */
      {
      real coeff;
      int  fn[2];
      int  pow[2];
      int  harm[2];
      char var;
      }
      trigt;

/*--- manifest constants ---*/

#define TNull      ((trigt*)0)

#define FnCosSin   0
#define FnCos      1
#define FnSin      2

/*--- declarations ---*/

int   trig_size(trigt *sym1); /* number of terms in series */
trigt *trig_alloc(int n); /* allocate memory for series */
void  trig_free(trigt *sym1); /* free memory */
void  trig_mfree(trigt *sym1,...); /* multiple free */
trigt *trig_clear(trigt *sym); /* clear term in series */
trigt *trig_copy(trigt *sym1,trigt *sym2,int n); /* copy series */
trigt *trig_dup(trigt *sym1,int n); /* duplicate series */
trigt *trig_realloc(trigt **sym1,int n);
          /* change size of memory allocated */
trigt *trig_reassign(trigt **sym1,trigt *sym2);
          /* reallocate pointer to series to new series */
trigt *trig_op(trigt **sym1,trigt *fn()); /* operate on series */
trigt **trig_mat_op(); /* operate on matrix */
trigt *trig_set(); /* define a series */
trigt *trig_set_const(trigt *sym1,real cnst);
         /* set series equal to 1 constant term */
trigt *trig_const(real cnst);
         /* define series equal to 1 constant term */
trigt *trig_mult_const(trigt* sym1,real cnst);
         /* multiply series by constant */
trigt *trig_add(real cnst1,trigt *sym1,real cnst2,trigt *sym2);
         /* add 2 series */
real  trig_calc(trigt *sym1,real var);
         /* calculate numerical value of series */
real  *trig_mat_calc();
         /* evaluate symbolic matrix */
int   trig_cmp(trigt *sym1,trigt *sym2);
         /* compare two terms for algebraic addition */
```

```
trigt *trig_collect(trigt *sym1); /* collect like terms in series */
trigt *trig_mult(trigt *sym1,trigt *sym2); /* multiply two series */
trigt *trig_binomial(real cnst0,real coeff,int fn0,int harm0,
          real cnst1,int pow0,...);
        /* expand binomial */
trigt *trig_expand(trigt *sym1);
        /* simplify trigonometric series into harmonics */
trigt *trig_significant(trigt *sym1); /* discard insignificant terms */
trigt *trig_diff(trigt *sym1); /* differentiate */
real trig_diff_calc(trigt *sym1,real var);
        /* evaluate derivative of symbolic matrix */
real  *trig_mat_calc_diff();
void  trig_mat_free(); /* free memory allocated for matrix */
void  trig_mat_mfree(); /* free many matrices */
trigt **trig_mat_mult_const();
         /* multiply matrix by constant */
trigt *trig_mat_minor_det();
         /* derive the determinant of a minor of a matrix */
trigt *trig_mat_det();
         /* derive and simplify determinant of matrix */
trigt **trig_mat_adj();
         /* derive and simplify adjoint of matrix */
trigt **trig_mat_mult();
         /* derive product of two matrices */
char  *trig_format(char *buf,trigt *sym1);
         /* format term for ASCII output */
void  trig_output(char *msg,trigt *sym1); /* display series */
void  trig_mat_output(); /* display symbolic matrix */
char  *trig_format_tex(char *buf,trigt *sym1);
         /* format term in TeX format */
void  trig_output_tex(FILE *stream,char *msg,trigt *sym1);
         /* output in TeX */
void  trig_mat_output_tex();
         /* output symbolic matrix in TeX */

/*--- functions ---*/

int trig_size(sym) /* number of terms in series */
trigt *sym;
{
int n;
if (!sym) return (0);
for (n = 0; sym[n].coeff != 0.; n++);
return (n);
}

int n_free = 0, n_alloc = 0; /* globals */

trigt *trig_alloc(n) /* allocate memory for series */
{
trigt *sym0;
sym0 = (trigt*) malloc((n+1)*sizeof(trigt));
if (sym0 == Null)
   {
   printf("cannot allocate memory (%d terms)\n",n);
   return (Null);
   }
n_alloc++;
memset(sym0,0,(n+1)*sizeof(trigt));
return (sym0);
}

void trig_free(sym0) /* free memory */
trigt *sym0;
```

```
{
if (sym0 == Null)
   printf("error: free: null pointer\n");
else
if (n_alloc <= n_free)
   printf("error: free: exceeded\n");
else
   {
   free(sym0);
   n_free++;
   }
}

trigt *trig_copy(sym0,sym1,n)
trigt *sym0,*sym1;
{
if (n == 0) n = trig_size(sym1) + 1;
memcpy(sym0,sym1,sizeof(trigt)*n);
return(sym0);
}

trigt *trig_dup(sym1,n) /* duplicate series */
trigt *sym1;
{
trigt *sym0;
if (n == 0) n = trig_size(sym1) + 1;
sym0 = trig_alloc(n);
trig_copy(sym0,sym1,n);
return (sym0);
}

trigt *trig_clear(sym)
trigt *sym;
{
int i;
sym[0].coeff = 0.;
for (i = 0; i < 2; i++)
   {
   sym[0].fn[i] = 0; sym[0].pow[i] = 0; sym[0].harm[i] = 0;
   }
return(sym);
}

trigt *trig_realloc(sym1,n) /* change size of memory allocated */
trigt **sym1;
{
trigt *sym0;
sym0 = trig_dup(*sym1,n);
trig_free(*sym1);
*sym1 = sym0;
return (*sym1);
}

   /* reallocate pointer to series to new series */
trigt *trig_reassign(sym0,sym1)
trigt **sym0,*sym1;
{
if (*sym0) trig_free(*sym0);
*sym0 = sym1;
return (*sym0);
}

trigt *trig_op(sym0,fn) /* operate on series */
trigt **sym0;
```

```
trigt *fn(); /* operator */
{
trigt *sym1;
if (*sym0)
   {
   sym1 = fn(*sym0);
   trig_free(*sym0);
   *sym0 = sym1;
   }
return (*sym0);
}

trigt **trig_mat_op(sm0,fn) /* operate on matrix */
trigt **sm0;
trigt *fn(); /* operator */
{
int i,j;
for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      trig_op(&sm0[i*3+j],fn);

return (sm0);
}

            /* set series equal to 1 constant term */
trigt *trig_set_const(sym0,cnst)
trigt *sym0;
real cnst;
{
int  i;
sym0[0].coeff = cnst;
for (i = 0; i < 2; i++)
   {
   sym0[0].fn[i]   = 0;
   sym0[0].pow[i]  = 0;
   sym0[0].harm[i] = 0;
   }
sym0[1].coeff = 0.;
return(sym0);
}

trigt *trig_mult_const(sym0,cnst)
trigt *sym0;
real cnst;
{
int i;
trigt *sym1;
sym1 = trig_dup(sym0,0);
for (i = 0; sym1[i].coeff; i++)
   {
   sym1[i].coeff *= cnst;
   }
return  (sym1);
}

trigt *trig_add(cnst1,sym1,cnst2,sym2) /* add 2 series */
trigt *sym1,*sym2;
real cnst1,cnst2;
{
int i,n,n1,n2,ns;
trigt *sym0;

n1 = trig_size(sym1);
n2 = trig_size(sym2);
```

```
sym0 = trig_alloc(ns = n1 + n2);

n = 0;
if (sym1 && cnst1 != 0.)
   for (i = 0; i < n1; i++)
      {
      trig_copy(&sym0[n],&sym1[i],1);
      sym0[n].coeff *= cnst1;
      n++;
      }
if (sym2 && cnst2 != 0.)
   for (i = 0; i < n2; i++)
      {
      trig_copy(&sym0[n],&sym2[i],1);
      sym0[n].coeff *= cnst2;
      n++;
      }
sym0[n].coeff = 0.;
if (n > ns) printf("error: add: size\n");
trig_op(&sym0,trig_collect);
return(sym0);
}

real trig_calc(sym0,var0) /* calculate numerical value of series */
trigt *sym0;
real var0;
{
int i,k;
real fact, term, sum = 0.;

if (sym0 == Null) return (0.);
for (i = 0; sym0[i].coeff != 0.; i++)
   {
   term = sym0[i].coeff;
   for (k = 0; k < 2; k++)
      if (sym0[i].fn[k])
         {
         if (sym0[i].fn[k] == FnCos)
            fact = cos(sym0[i].harm[k]*var0);
         else
         if (sym0[i].fn[k] == FnSin)
            fact = sin(sym0[i].harm[k]*var0);
         if (sym0[i].pow[k] != 1)
            fact = pow(fact,(real)sym0[i].pow[k]);
         term *= fact;
         }
   sum += term;
   }
return (sum);
}

real *trig_mat_calc(m0,sm0,var0) /* evaluate symbolic matrix */
real *m0;
trigt *sm0[];
real var0;
{
int i,j;
for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      m0[i*3+j] = trig_calc(sm0[i*3+j],var0);
return (m0);
}
```

```
   /* compare two terms for algebraic addition */
int trig_cmp(sym0,sym1)
trigt *sym0,*sym1;
{
int i;
for (i = 0; i < 2; i++)
   {
   if (sym0->fn[i] != sym1->fn[i]) return (0);
   if (sym0->fn[i])
      {
      if (sym0->pow[i] != sym1->pow[i]) return (0);
      if (sym0->harm[i] != sym1->harm[i]) return (0);
      }
   }
return (1);
}

void trig_switch_fn(sym0,i,j)
trigt *sym0;
{
trigt sym1;
trig_copy(&sym1,sym0,1);
sym0->fn[i]   = sym1.fn[j];
sym0->pow[i]  = sym1.pow[j];
sym0->harm[i] = sym1.harm[j];
sym0->fn[j]   = sym1.fn[i];
sym0->pow[j]  = sym1.pow[i];
sym0->harm[j] = sym1.harm[i];
}

void trig_order_fn(t0)
trigt *t0;
{
if (t0->fn[1] == FnCos && t0->fn[0] != FnCos)
 trig_switch_fn(t0,0,1);
}

trigt *trig_order(sym0)
trigt *sym0;
{
int i,j,ns;
trigt *sym1;
sym1 = trig_dup(sym0,0);
for (i = 0; sym0[i].coeff; i++) trig_order_fn(&sym0[i]);
return (sym1);
}

trigt *trig_collect(sym1) /* collect like terms in series */
trigt *sym1;
{
int i,j,n;
trigt *sym0,*sym2;

sym0 = trig_dup(sym1,0);
n = trig_size(sym0);
for (i = 0; i < n; i++)
   {
   trig_order_fn(&sym0[i]);
   if (sym0[i].fn[0] && sym0[i].fn[0] == sym0[i].fn[1]
         && sym0[i].harm[0] == sym0[i].harm[1])
      {
      if (sym0[i].fn[0] == FnSin)
         {
         sym0[i].pow[1] += sym0[i].pow[0];
```

```
        sym0[i].fn[0]   = 0;
        sym0[i].pow[0]  = 0;
        }
      else
        {
        sym0[i].pow[0] += sym0[i].pow[1];
        sym0[i].fn[1]   = 0;
        sym0[i].pow[1]  = 0;
        }
      }
   if (sym0[i].fn[0] && sym0[i].harm[0] < 0)
      {
      if (sym0[i].fn[0] == FnSin)
        {
        sym0[i].coeff   *= -1;
        sym0[i].harm[0] *= -1;
        }
      else
        {
        sym0[i].harm[0] *= -1;
        }
      }
     if (sym0[i].fn[1] && sym0[i].harm[1] < 0)
      {
      if (sym0[i].fn[1] == FnSin)
        {
        sym0[i].coeff   *= -1;
        sym0[i].harm[1] *= -1;
        }
      else
        {
        sym0[i].harm[1] *= -1;
        }
      }
   }
for (i = 0; i < n; i++)
   {
   if (sym0[i].coeff)
      for (j = 0; j < n; j++)
        {
        if (i != j && sym0[j].coeff && trig_cmp(&sym0[i],&sym0[j]))
          {
          sym0[i].coeff += sym0[j].coeff;
          sym0[j].coeff  = 0.;
          }
        }
   }
for (j = 0, i = 0; j < n; j++)
   if (sym0[j].coeff)
      {
      if (i != j) trig_copy(&sym0[i],&sym0[j],1);
      i++;
      }
sym0[i].coeff = 0.;
trig_realloc(&sym0,0);
return (sym0);
}

trigt *trig_mult(sym1,sym2) /* multiply two series */
trigt *sym1,*sym2;
{
trigt *sym0;
int i,j,k,l,n,n1,n2,ns;
trigt *t0,*t1,*t2;
```

```
real rv0,rv1,rv2;

rv1 = trig_calc(sym1,dvthe);
rv2 = trig_calc(sym2,dvthe);

n1 = trig_size(sym1);
n2 = trig_size(sym2);
sym0 = trig_alloc(ns = n1*n2);
n = 0;

for (i = 0; i < n1; i++)
   for (j = 0; j < n2; j++)
      {
      t0 = sym0 + n;
      t1 = sym1 + i;
      t2 = sym2 + j;

      trig_order_fn(t1);
      trig_order_fn(t2);
      trig_clear(t0);

      t0->coeff = t1->coeff*t2->coeff;
      for (k = 0; k < 2; k++)
         {
         if (t1->fn[k] && t2->fn[k] == 0 && !t0->fn[k])
            {
            t0->fn[k]   = t1->fn[k];
            t0->pow[k]  = t1->pow[k];
            t0->harm[k] = t1->harm[k];
            }
         else
         if (t1->fn[k] == 0 && t2->fn[k] && !t0->fn[k])
            {
            t0->fn[k]   = t2->fn[k];
            t0->pow[k]  = t2->pow[k];
            t0->harm[k] = t2->harm[k];
            }
         else
         if (t1->fn[k] && t1->fn[k] == t2->fn[k] &&
               t1->harm[k] == t2->harm[k] && !t0->fn[k])
            {
            t0->fn[k]   = t1->fn[k];
            t0->pow[k]  = t1->pow[k] + t2->pow[k];
            t0->harm[k] = t1->harm[k];
            }
         else
         if (t1->fn[k] && t2->fn[k] && t0->fn[1-k] == 0 &&
               !t0->fn[k] && !t0->fn[1-k])
            {
            t0->fn[0]   = t1->fn[k];
            t0->pow[0]  = t1->pow[k];
            t0->harm[0] = t1->harm[k];

            t0->fn[1]   = t2->fn[k];
            t0->pow[1]  = t2->pow[k];
            t0->harm[1] = t2->harm[k];
            }
         else
         if (t1->fn[k] || t2->fn[k])
            {
            char buf1[32],buf2[32];
            printf("error: mult: %s: %s\n",trig_format(buf1,t1),
                  trig_format(buf2,t2));
            }
```

```
        }
      n++;
      }
sym0[n].coeff = 0.;
if (n > ns) printf("error: mult: size\n");
trig_op(&sym0,trig_collect);
return (sym0);
}

trigt *trig_binomial(cnst0,coeff,fn0,harm0,cnst1,pow0,fn1,pow1,harm1)
real cnst0,cnst1,coeff;
{
int i,j,n,ns;
trigt *sym0;
sym0 = trig_alloc(ns = 128);
for (i = 0; i <= pow0; i++)
   {
   sym0[i].coeff = cnst0*binomial_coeff(pow0,i);
   if (i < pow0)
       sym0[i].coeff *= pow(cnst1,(real)(pow0 - i));
   if (i)
      {
      sym0[i].coeff  *= pow(coeff,(real)i);
      sym0[i].fn[0]   = fn0;
      sym0[i].pow[0]  = i;
      sym0[i].harm[0] = harm0;
      }
   else
      {
      sym0[i].fn[0]  = 0;
      sym0[i].pow[0] = 0;
      }
   if (fn1 && pow1)
      {
      sym0[i].fn[1]   = fn1;
      sym0[i].pow[1]  = pow1;
      sym0[i].harm[1] = harm1;
      }
   else
      {
      sym0[i].fn[1]   = 0;
      sym0[i].pow[1]  = 0;
      sym0[i].harm[1] = 0;
      }
   if (fn0 == FnSin) trig_switch_fn(&sym0[i],0,1);
   }
sym0[i].coeff = 0.;
if (i > ns) printf("error: binomial: size\n");
trig_realloc(&sym0,0);
return (sym0);
}

 /* simplify trigonometric series into harmonics */
trigt *trig_expand(sym1)
trigt *sym1;
{
int n = 0;
int i,j,k,n1,n2,m0,m1,m2,ns;
int stat = 0;
trigt *sym0,*sym2,*t;

sym0 = trig_alloc(ns = 256);

for (i = 0; sym1[i].coeff != 0.; i++)
```

```
{
t = &sym1[i];
if (t->fn[0] == FnSin && t->fn[1] != FnSin)
   {
   trig_switch_fn(t,0,1);
   }
if (t->fn[0] && t->fn[0] == t->fn[1] &&
       t->harm[0] == t->harm[1])
   {
   if (t->fn[0] == FnSin)
      {
      t->pow[1] += t->pow[0];
      t->fn[0] = 0;
      t->pow[0] = 0;
      }
   else
      {
      t->pow[0] += t->pow[1];
      t->fn[1] = 0;
      t->pow[1] = 0;
      }
   }
 if (t->fn[0] && t->harm[0] < 0)
   {
   if (t->fn[0] == FnSin)
      {
      t->coeff  *= -1;
      t->harm[0]  *= -1;
      }
   else
      {
      t->harm[0] *= -1;
      }
   }
 if (t->fn[1] && t->harm[1] < 0)
   {
   if (t->fn[1] == FnSin)
      {
      t->coeff  *= -1;
      t->harm[1] *= -1;
      }
   else
      {
      t->harm[1] *= -1;
      }
   }

 m0 = Min(t->pow[0],t->pow[1]);
 m1 = Max(t->pow[0],t->pow[1]);

 sym2 = Null;

 if (m0 == 0 && m1 > 1) /* cos^n ax, sin^n ax */
   {
   stat = 1;
   m2 = m1/2;
   if (Even(m1))
      {
      if (t->pow[0] == m1 && t->fn[0] == FnCos)
         sym2 = trig_binomial(t->coeff,.5,FnCos,
                   2*t->harm[0],.5,m2,0);
      else
      if (t->pow[1] == m1 && t->fn[1] == FnSin)
         sym2 = trig_binomial(t->coeff,-.5,FnCos,
```

```
                2*t->harm[1],.5,m2,0);
      else
         printf("error: expand: %s\n",trig_format(buf0,t));
      }
   else
      {
      if (t->pow[0] == m1 && t->fn[0] == FnCos)
         sym2 = trig_binomial(t->coeff,.5,FnCos,
                  2*t->harm[0],
                  .5,m2,t->fn[0],1,t->harm[0]);
      else
        if (t->pow[1] == m1 && t->fn[1] == FnSin)
         sym2 = trig_binomial(t->coeff,-.5,FnCos,
                  2*t->harm[1],
                  .5,m2,t->fn[1],1,t->harm[1]);
      else
         printf("error: expand: %s\n",trig_format(buf0,t));
      }
   }
else
if (t->fn[0] == FnCos && t->fn[1] == FnSin
   && t->harm[0] == t->harm[1] && t->pow[0] == t->pow[1])
   {
   stat = 2;
   sym0[n].coeff   = pow(.5,(real)m0)*t->coeff;
   sym0[n].fn[0]   = 0;
   sym0[n].pow[0]  = 0;
   sym0[n].fn[1]   = FnSin;
   sym0[n].pow[1]  = t->pow[0];
   sym0[n].harm[1] = 2*t->harm[0];
   n++;
   }
else
if (t->fn[0] == FnCos && t->fn[1] == FnSin
   && t->harm[0] == t->harm[1] && Even(m1 - m0))
   {
   stat = 3;
   m2 = (m1 - m0)/2;
   if (t->pow[0] == m0)
      {
      sym2 = trig_binomial(pow(.5,(real)m0)*t->coeff,
               -.5,FnCos,2*t->harm[0],.5,m2,FnSin,
               m0,2*t->harm[0]);
      }
   else
      {
      sym2 = trig_binomial(pow(.5,(real)m0)*t->coeff,
               .5,FnCos,2*t->harm[0],.5,m2,FnSin,
               m0,2*t->harm[0]);
      }
   }
else
if (m0 && Even(m1))
   {
   stat = 4;
   m2 = m1/2;
   if (m1 == t->pow[0])
      {
      if (t->fn[0] == FnCos)
         sym2 = trig_binomial(t->coeff,.5,FnCos,2*t->harm[0],
                     .5,m2,t->fn[1],t->pow[1],t->harm[1]);
      else
      if (t->fn[0] == FnSin)
         sym2 = trig_binomial(t->coeff,-.5,FnCos,2*t->harm[0],
```

```
                    .5,m2,t->fn[1],t->pow[1],t->harm[1]);
      }
   else
   if (m1 == t->pow[1])
      {
      if (t->fn[1] == FnCos)
         sym2 = trig_binomial(t->coeff,.5,FnCos,2*t->harm[1],
                    .5,m2,t->fn[0],t->pow[0],t->harm[0]);
      else
      if (t->fn[1] == FnSin)
         sym2 = trig_binomial(t->coeff,-.5,FnCos,2*t->harm[1],
                    .5,m2,t->fn[0],t->pow[0],t->harm[0]);
      }
   }
else
if (t->pow[0] == 1 && t->pow[1] == 1)
   {
   stat = 5;
   if (t->fn[0] == FnSin && t->fn[1] == FnCos)
      {
      sym0[n].coeff   = .5*t->coeff;
      sym0[n].fn[0]   = 0;
      sym0[n].pow[0]  = 0;
      sym0[n].fn[1]   = FnSin;
      sym0[n].pow[1]  = 1;
      sym0[n].harm[1] = t->harm[0] - t->harm[1];
      n++;
      sym0[n].coeff   = .5*t->coeff;
      sym0[n].fn[0]   = 0;
      sym0[n].pow[0]  = 0;
      sym0[n].fn[1]   = FnSin;
      sym0[n].pow[1]  = 1;
      sym0[n].harm[1] = t->harm[1] + t->harm[0];
      n++;
      }
   else
   if (t->fn[0] == FnCos && t->fn[1] == FnSin)
      {
      sym0[n].coeff   = .5*t->coeff;
      sym0[n].fn[0]   = 0;
      sym0[n].pow[0]  = 0;
      sym0[n].fn[1]   = FnSin;
      sym0[n].pow[1]  = 1;
      sym0[n].harm[1] = t->harm[1] - t->harm[0];
      n++;
      sym0[n].coeff   = .5*t->coeff;
      sym0[n].fn[0]   = 0;
      sym0[n].pow[0]  = 0;
      sym0[n].fn[1]   = FnSin;
      sym0[n].pow[1]  = 1;
      sym0[n].harm[1] = t->harm[1] + t->harm[0];
      n++;
      }
   else
   if (t->fn[0] == FnCos && t->fn[1] == FnCos)
      {
      sym0[n].coeff   = .5*t->coeff;
      sym0[n].fn[0]   = FnCos;
      sym0[n].pow[0]  = 1;
      sym0[n].harm[0] = abs(t->harm[1] - t->harm[0]);
      sym0[n].fn[1]   = 0;
      sym0[n].pow[1]  = 0;
      n++;
      sym0[n].coeff   = .5*t->coeff;
```

```
      sym0[n].fn[0]   = FnCos;
      sym0[n].pow[0]  = 1;
      sym0[n].harm[0] = t->harm[1] + t->harm[0];
      sym0[n].fn[1]   = 0;
      sym0[n].pow[1]  = 0;
      n++;
      }
   else
   if (t->fn[0] == FnSin && t->fn[1] == FnSin)
      {
      sym0[n].coeff   = .5*t->coeff;
      sym0[n].fn[0]   = FnCos;
      sym0[n].pow[0]  = 1;
      sym0[n].harm[0] = abs(t->harm[1] - t->harm[0]);
      sym0[n].fn[1]   = 0;
      sym0[n].pow[1]  = 0;
      n++;
      sym0[n].coeff   = -.5*t->coeff;
      sym0[n].fn[0]   = FnCos;
      sym0[n].pow[0]  = 1;
      sym0[n].harm[0] = t->harm[1] + t->harm[0];
      sym0[n].fn[1]   = 0;
      sym0[n].pow[1]  = 0;
      n++;
      }
   else
      printf("error: expand\n");
   }
else
if (t->fn[0] == FnCos && t->fn[1] == FnSin
   && t->harm[0] == t->harm[1])
   {
   stat = 6;

   sym0[n].coeff = pow(.5,(real)m0)*t->coeff;

   if (t->pow[0] == m0)
      {
      sym0[n].fn[0]   = FnSin;
      sym0[n].pow[0]  = m0;
      sym0[n].harm[0] = 2*t->harm[0];
      sym0[n].pow[1] -= m0;
      }
   else
      {
      sym0[n].pow[0] -= m0;
      sym0[n].fn[1]   = FnSin;
      sym0[n].pow[1]  = m0;
      sym0[n].harm[1] = 2*t->harm[0];
      }

   n++;
   }
else
   {
   trig_copy(&sym0[n],&sym1[i],1);
   n++;
   }

if (sym2)
   {
   n1 = trig_size(sym2);

   if (n+n1 > ns - 64)
```

```
         {
         trig_realloc(&sym0,ns += 32);
         }

      trig_copy(&sym0[n],sym2,n1); trig_free(sym2);
      n += n1;
      }
   else
   if (n > ns - 64)
      {
      trig_realloc(&sym0,ns += 32);
      }
   }
sym0[n].coeff = 0.;

trig_op(&sym0,trig_collect);
if (stat) trig_op(&sym0,trig_expand);
return (sym0);
}

trigt *trig_significant(sym1) /* discard insignificant terms */
trigt *sym1;
{
int i,j,n;
trigt *sym0;
real rm = 0.;

sym0 = trig_dup(sym1,0);

n = trig_size(sym0);

for (i = 0; i < n; i++)
   if (fabs(sym0[i].coeff) > rm)
      rm = fabs(sym0[i].coeff);

for (i = 0; i < n; i++)
   if (fabs(sym0[i].coeff/rm) < dvsig)
      sym0[i].coeff = 0.;

i = 0;
for (j = 0; j < n; j++)
   if (sym0[j].coeff)
      {
      if (i != j) trig_copy(&sym0[i],&sym0[j],1);
      i++;
      }

sym0[i].coeff = 0.;

trig_realloc(&sym0,0);

return (sym0);
}

trigt *trig_diff(t1) /* differentiate */
trigt *t1;
{
int i,k,n1,n = 0,ns;
trigt *t0;
real rv0,rv1;

n1 = trig_size(t1);
t0 = trig_alloc(ns = n1*2);
```

```
for (i = 0; i < n1; i++)
   {
   if (t1[i].fn[0] && t1[i].fn[0] == t1[i].fn[1] &&
         t1[i].harm[0] == t1[i].harm[1])
      {
      t1[i].pow[0] += t1[i].pow[1];
      t1[i].fn[1]   = 0;
      t1[i].pow[1]  = 0;
      t1[i].harm[1] = 0;
      }

   if (t1[i].fn[0] && !t1[i].fn[1])
      {
      trig_clear(&t0[n]);

      t0[n].coeff = t1[i].coeff*t1[i].pow[0]*t1[i].harm[0];

      t0[n].pow[0]  = 1;
      t0[n].harm[0] = t1[i].harm[0];

      if (t1[i].fn[0] == FnCos)
         {
         t0[n].coeff *= -1;
         t0[n].fn[0]  = FnSin;
         }
      else
      if (t1[i].fn[0] == FnSin)
         {
         t0[n].fn[0] = FnCos;
         }

      if (t1[i].pow[0] > 1)
         {
         t0[n].fn[1]   = t1[i].fn[0];
         t0[n].pow[1]  = t1[i].pow[0] - 1;
         t0[n].harm[1] = t1[i].harm[0];
         }

      n++;
      }
    else
    if (t1[i].fn[1] && !t1[i].fn[0])
      {
      trig_clear(&t0[n]);

      t0[n].coeff = t1[i].coeff*t1[i].pow[1]*t1[i].harm[1];

      t0[n].pow[0]  = 1;
      t0[n].harm[0] = t1[i].harm[1];

      if (t1[i].fn[1] == FnCos)
         {
         t0[n].coeff *= -1;
         t0[n].fn[0]  = FnSin;
         }
      else
      if (t1[i].fn[1] == FnSin)
         {
         t0[n].fn[0] = FnCos;
         }

      if (t1[i].pow[1] > 1)
         {
         t0[n].fn[1]   = t1[i].fn[1];
```

```
      t0[n].pow[1]  = t1[i].pow[1] - 1;
      t0[n].harm[1] = t1[i].harm[1];
      }

   n++;
   }
else
if (t1[i].fn[0] && t1[i].fn[1] && t1[i].harm[0] == t1[i].harm[1])
   {
   trig_clear(&t0[n]);

   t0[n].coeff = t1[i].coeff*t1[i].pow[0]*t1[i].harm[0];

   if (t1[i].pow[0] == 1)
      {
      t0[n].fn[0]   = 0;
      t0[n].pow[0]  = 0;
      t0[n].harm[0] = 0;
      }
   else
      {
      t0[n].fn[0]   = t1[i].fn[0];
      t0[n].pow[0]  = t1[i].pow[0] - 1;
      t0[n].harm[0] = t1[i].harm[0];
      }

   t0[n].fn[1]   = t1[i].fn[1];
   t0[n].pow[1]  = t1[i].pow[1] + 1;
   t0[n].harm[1] = t1[i].harm[1];

   if (t1[i].fn[0] == FnCos) t0[n].coeff *= -1;

   n++;

   trig_clear(&t0[n]);

   t0[n].coeff = t1[i].coeff*t1[i].pow[1]*t1[i].harm[1];

   t0[n].fn[0]   = t1[i].fn[0];
   t0[n].pow[0]  = t1[i].pow[0] + 1;
   t0[n].harm[0] = t1[i].harm[0];

   if (t1[i].pow[1] == 1)
      {
      t0[n].fn[1]   = 0;
      t0[n].pow[1]  = 0;
      t0[n].harm[1] = 0;
      }
   else
      {
      t0[n].fn[1]   = t1[i].fn[1];
      t0[n].pow[1]  = t1[i].pow[1] - 1;
      t0[n].harm[1] = t1[i].harm[1];
      }

   if (t1[i].fn[1] == FnCos) t0[n].coeff *= -1;

   n++;
   }
else
if (t1[i].fn[0] && t1[i].fn[1])
   {
   char buf1[32];
   printf("error: diff: %s\n",trig_format(buf1,&t1[i]));
```

```
      getch();
      }
   }
t0[n].coeff = 0;

if (n > ns) printf("error: diff: size\n");
trig_op(&t0,trig_order);
trig_realloc(&t0,0);
return (t0);
}

real trig_diff_calc(t,arg) /* differentiate */
trigt *t;
real arg;
{
int i,n;
real sum = 0, term;

n = trig_size(t);

for (i = 0; i < n; i++)
   {
   if (!t[i].fn[0] && !t[i].fn[1]) continue;

   if (t[i].fn[0])
      {
      term = t[i].coeff*t[i].harm[0];

      if (t[i].fn[0] == FnCos)
         {
         if (t[i].pow[0] != 1)
            term *= t[i].pow[0]*pow(cos(t[i].harm[0]*arg),
                        (real)t[i].pow[0] - 1.);
         term *= - sin(t[i].harm[0]*arg);
         }
      else
      if (t[i].fn[0] == FnSin)
         {
         if (t[i].pow[0] != 1)
            term *= t[i].pow[0]*pow(sin(t[i].harm[0]*arg),
                        (real)t[i].pow[0] - 1.);
         term *= cos(t[i].harm[0]*arg);
         }

      if (t[i].fn[1] == FnCos)
         term *= pow(cos(t[i].harm[1]*arg),(real)t[i].pow[1]);
      else
      if (t[i].fn[1] == FnSin)
         term *= pow(sin(t[i].harm[1]*arg),(real)t[i].pow[1]);

      sum += term;
      }

    if (t[i].fn[1])
      {
      term = t[i].coeff*t[i].harm[1];
      if (t[i].fn[1] == FnCos)
         {
         if (t[i].pow[1] != 1)
            term *= t[i].pow[1]*pow(cos(t[i].harm[1]*arg),
                           (real)t[i].pow[1] - 1.);
         term *= - sin(t[i].harm[1]*arg);
         }
      else
```

```
      if (t[i].fn[1] == FnSin)
         {
         if (t[i].pow[1] != 1)
            term *= t[i].pow[1]*pow(sin(t[i].harm[1]*arg),
                        (real)t[i].pow[1] - 1.);
         term *= cos(t[i].harm[1]*arg);
         }

      if (t[i].fn[0] == FnCos)
         term *= pow(cos(t[i].harm[0]*arg),(real)t[i].pow[0]);
      else
      if (t[i].fn[0] == FnSin)
         term *= pow(sin(t[i].harm[0]*arg),(real)t[i].pow[0]);

      sum += term;
      }
   }

return (sum);
}

/*--- matrices ---*/

         /* multiply matrix by constant */
trigt **trig_mat_mult_const(sm,cnst)
trigt *sm[];
real cnst;
{
int i,j,n;

for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      for (n = 0; sm[i*3+j][n].coeff != 0.; n++)
         sm[i*3+j][n].coeff *= cnst;

return (sm);
}

         /* derive the determinant of a minor of a matrix */
trigt *trig_mat_minor_det(sm0,i,j)
trigt *sm0[];
{
int m1,m2,n1,n2;
trigt *sym0,*sym1,*sym2;

m1 = (i+1)%3; n1 = (j+1)%3;
m2 = (i+2)%3; n2 = (j+2)%3;

sym1 = trig_mult(sm0[m1*3+n1],sm0[m2*3+n2]);
sym2 = trig_mult(sm0[m2*3+n1],sm0[m1*3+n2]);

sym0 = trig_add(1.,sym1,-1.,sym2);
trig_free(sym1); trig_free(sym2);

trig_op(&sym0,trig_collect);

return (sym0);
}

         /* derive and simplify determinant of matrix */
trigt *trig_mat_det(sm0)
trigt *sm0[];
{
int i;
```

```
trigt *sym0 = Null,*sym1;


for (i = 0; i < 3; i++)
   {
   sym1 = trig_mat_minor_det(sm0,0,i);
   trig_reassign(&sym1,trig_mult(sym1,sm0[i]));
   trig_reassign(&sym0,trig_add(1.,sym0,1.,sym1));
   trig_free(sym1);
   }

trig_op(&sym0,trig_collect);
trig_op(&sym0,trig_expand);
trig_op(&sym0,trig_significant);

return (sym0);
}

         /* derive and simplify adjoint of matrix */
trigt **trig_mat_adj(sm0,sm1)
trigt *sm0[],*sm1[];
{
int i,j;

for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      sm0[j*3+i] = trig_mat_minor_det(sm1,i,j);

return (sm0);
}

         /* derive product of two matrices */
trigt **trig_mat_mult(sm0,sm1,sm2)
trigt *sm0[],*sm1[],*sm2[];
{
int i,j,k;
trigt *sym0;
for (i = 0; i < 3; i++)
   for (j = 0; j < 3; j++)
      {
      sm0[i*3+j] = Null;
      for (k = 0; k < 3; k++)
         {
         sym0 = trig_mult(sm1[i*3+k],sm2[k*3+j]);
         trig_reassign(&sm0[i*3+j],
            trig_add(1.,sm0[i*3+j],1.,sym0));
         trig_free(sym0);
         }
      trig_op(&sm0[i*3+j],trig_collect);
      }
return (sm0);
}
```

Appendix C

Routines for Piecewise Integrals

```
/*--- higher order theory
 *
 *    Evan Bryan Summers
 *       University of Natal, Durban 4001, South Africa
 *          May 1992
 *
 *
 *       Higher Order Theory
 *          Symbolic Computation
 *
 */


/*--- types ---*/

typedef  struct /* power series ito z */
         {
         real coeff;
         int  pow;
         }
         powt;

typedef symh int; /* symbol handle */

typedef struct /* symbol definition */
         {
         char  *desc;
         uchar type;
         uchar stat;
         powt  **pp;
         symh  *expr;
         real  *rp;
         real  rv;
         int   iv1,iv2;
         }
         symt;

/*--- power series ------------------------------------------*/

#define PNull   ((powt*)0)

int pow_size(ps) /* number of terms in series */
powt *ps;
{
int i;
if (!ps) return (0);
for (i = 0; ps[i].coeff; i++);
if (i > n_pow_size) n_pow_size = i;
return (i);
}

powt *pow_alloc(n) /* allocate memory for series */
{
powt *ps;
ps = (powt*) malloc((n+1)*sizeof(powt));
if (ps == PNull)
   {
   printf("cannot allocate memory (%d terms)\n",n);
   return (PNull);
```

```
   }
memset(ps,0,(n+1)*sizeof(powt));
n_pow_nterm++; n_pow_alloc++;
return (ps);
}

void pow_free(ps) /* free memory */
powt *ps;
{
if (ps)
   {
   free(ps);
   n_pow_free++;
   }
}

powt *pow_clear(ps)
powt *ps;
{
memset(ps,0,sizeof(powt));
return(ps);
}

powt *pow_copy(ps1,ps2,n)
powt *ps1,*ps2;
{
if (n == 0) n = pow_size(ps2) + 1;
memcpy(ps1,ps2,sizeof(powt)*n);
return(ps1);
}

powt *pow_dup(ps1,n) /* duplicate series */
powt *ps1;
{
powt *ps2;
if (n == 0) n = pow_size(ps1) + 1;
ps2 = pow_alloc(n);
pow_copy(ps2,ps1,n);
return (ps2);
}

powt *pow_realloc(ps1,n) /* change size of memory allocated */
powt **ps1;
{
powt *ps2;
ps2 = pow_dup(*ps1,n);
pow_free(*ps1);
*ps1 = ps2;
return (*ps1);
}

         /* reallocate pointer to new series */
static powt *pow_reassign(ps2,ps1)
powt **ps2,*ps1;
{
if (*ps2) pow_free(*ps2);
*ps2 = ps1;
return (*ps2);
}

powt *pow_op(ps1,fn) /* operate on series */
powt **ps1;
powt *fn(); /* operating function */
{
```

```
powt *ps2;
if (*ps1)
   {
   ps2 = fn(*ps1);
   pow_free(*ps1);
   *ps1 = ps2;
   }
return (*ps1);
}

real pow_calc(ps,z)
powt *ps;
real z;
{
int i,n;
real rv = 0.;
if (ps)
   {
   for (i = 0; ps[i].coeff; i++)
      {
      if (ps[i].pow == 0) rv += ps[i].coeff; else
      if (ps[i].pow == 1) rv += ps[i].coeff*z; else
      if (fabs(z) > dvsig)
         rv += ps[i].coeff*pow(z,(double)ps[i].pow);
      }
   }
return (real_sig(rv));
}

powt *pow_collect(ps1)
powt *ps1;
{
int i,j,n;
powt *ps;

if (!ps1) return (PNull);
ps = pow_dup(ps1,n = pow_size(ps1));
for (i = 0; i < n; i++)
   {
   if (ps[i].coeff)
      {
      for (j = 0; j < n; j++)
         {
         if (i != j && ps[j].coeff && ps[i].pow == ps[j].pow)
            {
            ps[i].coeff += ps[j].coeff;
            ps[j].coeff  = 0.;
            }
         }
      }
   }
for (j = 0, i = 0; j < n; j++)
   {
   if (chk_sig(ps[j].coeff))
      {
      if (i != j) pow_copy(&ps[i],&ps[j],1);
      i++;
      }
   }
ps[i].coeff = 0.;
pow_realloc(&ps,i);
return (ps);
}
```

```
powt *pow_add(ps1,ps2)
powt *ps1,*ps2;
{
powt *ps;
int n = 0;
int n1,n2;

n1 = pow_size(ps1);
n2 = pow_size(ps2);
ps = pow_alloc(n1+n2);
if (n1)
   {
   pow_copy(ps,ps1,n1);
   n += n1;
   }
if (n2)
   {
   pow_copy(&ps[n],ps2,n2);
   n += n2;
   }
ps[n].coeff = 0.;
pow_op(&ps,pow_collect);
return (ps);
}


powt *pow_mult_const(ps1,cnst)
powt *ps1;
real cnst;
{
powt *ps2;
int i,n;
ps2 = pow_dup(ps1,n = pow_size(ps1));
for (i = 0; i < n; i++)
   ps2[i].coeff *= cnst;
return (ps2);
}


powt *pow_sum(ps1,cnst,ps2)
powt **ps1,*ps2;
real cnst;
{
powt *ps;
ps = pow_mult_const(ps2,cnst);
pow_reassign(ps1,pow_add(*ps1,ps));
pow_free(ps);
return (*ps1);
}

powt *pow_mult(ps1,ps2)
powt *ps1,*ps2;
{
powt *ps;
int i,j,n,n1,n2;

if (ps1 == PNull && ps2 == PNull) return (PNull);
if (ps1 == PNull) return (pow_dup(ps2,0));
if (ps2 == PNull) return (pow_dup(ps1,0));
n1 = pow_size(ps1);
n2 = pow_size(ps2);
if (n1 == 0 || n2 == 0) return (pow_zero());
ps = pow_alloc(n1*n2);
n = 0;
```

```
for (i = 0; i < n1; i++)
   for (j = 0; j < n2; j++)
      {
      ps[n].coeff = ps1[i].coeff*ps2[j].coeff;
      ps[n].pow = ps1[i].pow+ps2[j].pow;
      n++;
      }
ps[n].coeff = 0.;
pow_op(&ps,pow_collect);
return (ps);
}

powt *pow_integrate(ps1,z0,cnst)
powt *ps1;
real z0,cnst;
{
int  i,j,n;
powt *ps;
if (!ps1) return (PNull);
n = pow_size(ps1);
ps = pow_dup(ps1,n + 1);
for (i = 0; i < n; i++)
   {
   ps[i].pow++;
   ps[i].coeff /= (real) ps[i].pow;
   }
ps[n].coeff = - pow_calc(ps,z0) + cnst;
ps[n].pow   = 0;
n++;
ps[n].coeff = 0;
return (ps);
}

powt *pow_define(args)
char args;
{
powt *ps;
int n = 0, ns = 8;
char *arg;
ps = pow_alloc(ns);
for (arg = &args; *(real*)arg;)
   {
   ps[n].coeff = *(double*)arg; arg += sizeof(double);
   ps[n].pow = *(int*)arg;  arg += sizeof(int);
   n++;
   }
ps[n].coeff = 0.;
pow_realloc(&ps,n);
return (ps);
}

/*--- symbolic expressions ------------------------------------*/

static int sym_size(se) /* number of terms in series */
symh *se;
{
int n;
if (!se) return (0);
for (n = 0; se[n]; n++);
return (n);
}

static symh *sym_alloc(n) /* allocate memory for series */
{
```

```
symh *se;
se = (symh*) malloc((n+1)*sizeof(symh));
if (se == Null)
   {
   printf("cannot allocate memory (%d terms)\n",n);
   return (Null);
   }
n_sym_nsym++;
n_sym_alloc++;
memset(se,0,(n+1)*sizeof(symh));
return (se);
}

static void sym_free(se) /* free memory */
symh *se;
{
if (se)
   {
   free(se);
   n_sym_free++;
   }
}

static symh *sym_copy(se1,se2,n)
symh *se1,*se2;
{
if (n == 0) n = sym_size(se2) + 1;
memcpy(se1,se2,sizeof(symh)*n);
return(se1);
}

static symh *sym_dup(se1,n) /* duplicate series */
symh *se1;
{
symh *se2;
if (n == 0) n = sym_size(se1);
se2 = sym_alloc(n);
sym_copy(se2,se1,n);
se2[n] = 0;
return (se2);
}

        /* change size of memory allocated */
static symh *sym_realloc(se1,n)
symh **se1;
{
symh *se2;
se2 = sym_dup(*se1,n);
sym_free(*se1);
*se1 = se2;
return (*se1);
}

        /* reallocate pointer to new series */
static symh *sym_reassign(se2,se1)
symh **se2,*se1;
{
if (*se2) sym_free(*se2);
*se2 = se1;
return (*se2);
}

static symh sym_define(desc,is,type)
char *desc;
```

```
symh is;
int type;
{
int i;
char *arg;

arg = (char*)&type + sizeof(int);

sym[is].desc = desc;
sym[is].type = type;

if (sym[is].type == SymConst)
   {
   sym[is].rv = *(real*)arg; arg += sizeof(double);
   }
else
if (sym[is].type == SymMConst)
   {
   sym[is].rp = *(real**)arg; arg += sizeof(void*);
   }
else
if (sym[is].type == SymZ)
   {
   sym[is].iv1 = *(int*)arg; arg += sizeof(int);
   }
else
if (sym[is].type == SymExpr)
   {
   sym[is].expr = sym_dup((symh*)arg,0);
   }
else
if (sym[is].type == SymInt)
   {
   sym[is].iv1  = *(int*)arg; arg += sizeof(int);
   sym[is].expr = sym_dup((symh*)arg,0);
   }
else
if (sym[is].type == SymLamInt)
   {
   sym[is].iv1  = *(int*)arg; arg += sizeof(int);
   sym[is].iv2  = *(int*)arg; arg += sizeof(int);
   sym[is].expr = sym_dup((symh*)arg,0);
   }
else
   {
   fprintf(fdeb,"error: sym: define\n");
   n_err++;
   return (0);
   }

sym_output(flog,is);

return (is);
}

static symh *sym_term(se)
symh *se;
{
int i;
if (!se) return (Null);
for (i = 0; se[i] && se[i] != SymPlus; i++);
if (!i) return (Null);
return (sym_dup(se,i));
}
```

```
static symh *sym_tail(se)
symh *se;
{
int i;
if (se)
   {
   for (i = 0; se[i] && se[i] != SymPlus; i++);

   if (se[i] == SymPlus)
      return (sym_dup(se+i+1,0));
   }
return (Null);
}

static int sym_term_size(se)
symh *se;
{
int i;
if (!se) return (0);
for (i = 0; se[i] && se[i] != SymPlus; i++);
return (i);
}


static int sym_tail_size(se)
symh *se;
{
int i;
if (se)
   {
   for (i = 0; se[i] && se[i] != SymPlus; i++);
   if (se[i] == SymPlus)
      return (sym_size(se+i+1));
   }
return (0);
}

powt *sym_expand(k,is,se)
symh is;
symh se[];
{
powt *ret = Null;

if (k < 1) { n_err++; return (Null);}

if (is == SymNone) ret = Null; else
if (is == SymExpr && (!se || !se[0])) ret = Null; else
if (is == SymExpr)
   {
   symh *sp0,*sp1;
   powt *ps;

   sp0 = sym_term(se+1);
   sp1 = sym_tail(se+1);

   ret = sym_expand(k,se[0]);

   if (sp0)
      {
      ps = sym_expand(k,SymExpr,sp0);
      pow_reassign(&ret,pow_mult(ret,ps));
      pow_free(ps);
      sym_free(sp0);
```

```
      }

   if (sp1)
      {
      ps = sym_expand(k,SymExpr,sp1);
      pow_reassign(&ret,pow_add(ret,ps));
      pow_free(ps);
      sym_free(sp1);
      }
   }
else
if (is == SymPos1) ret = pow_define(1.,0,0.);  else
if (is == SymNeg1) ret = pow_define(-1.,0,0.); else
if (is == SymZ)    ret = pow_define(1.,1,0.);  else
if (is == SymZ2)   ret = pow_define(1.,2,0.);  else
if (is < SymUserEnd)
   {
   if (sym[is].type == SymZ)
      ret = pow_define(1.,sym[is].iv1,0.);
   else
   if (sym[is].type == SymConst)
      ret = pow_define(sym[is].rv,0,0.);
   else
   if (sym[is].type == SymMConst)
      ret = pow_define(sym[is].rp[k-1],0,0.);
   else
   if (sym[is].type == SymMPow)
      ret = pow_dup(sym[is].pp[k-1],0);
   else
   if (sym[is].stat & SSmpow)
      ret = pow_dup(sym[is].pp[k-1],0);
   else
   if (sym[is].type == SymExpr)
      ret = sym_expand(k,SymExpr,sym[is].expr);
   else
   if (sym[is].type == SymInt)
      {
      powt *ps;
      real rv;

      ps = sym_expand(k,SymExpr,sym[is].expr);

      if (k > sym[is].iv1)
         {
         rv  = lam_int(sym[is].iv1,k-1,sym[is].expr);
         ret = pow_integrate(ps,ga[k-1],rv);
         }
      else
         {
         rv  = lam_int(sym[is].iv1,k,sym[is].expr);
         ret = pow_integrate(ps,ga[k],rv);
         }

      pow_free(ps);
      }
   else
   if (sym[is].stat & SSconst)
      ret = pow_define(sym[is].rv,0,0.);
   else
   if (sym[is].type == SymLamInt)
      {
      real rv;
      sym[is].rv = lam_int(sym[is].iv1,sym[is].iv2,sym[is].expr);
      ret = pow_define(sym[is].rv,0,0.);
```

```
      }
   else
      {
      fprintf(fdeb,"error: expand: type\n");
      n_err++;
      }
   }
else
   {
   fprintf(fdeb,"error: expand\n");
   n_err++;
   }

return (ret);
}


static real lam_int(m,k,se)
symh *se;
{
real ret;
powt *ps1,*ps2;

if (k < 0 || k > n_lay)
   {
   fprintf(fdeb,"error: integrate: layer number\n");
   n_err++;
   return (0.);
   }

if (k == m) return (0.);

if (!se) se = (symh*)(&se + 1);

if (k > m)
   {
   ret  = lam_int(m,k-1,se);
   ps1  = sym_expand(k,SymExpr,se);
   ps2  = pow_integrate(ps1,ga[k-1],0.);
   ret += pow_calc(ps2,ga[k]);
   }
else
   {
   ret  = lam_int(m,k,se);
   ps1  = sym_expand(k,SymExpr,se);
   ps2  = pow_integrate(ps1,ga[k],0.);
   ret += pow_calc(ps2,ga[k-1]);
   }

pow_free(ps1);
pow_free(ps2);

return (real_sig(ret));
}


void sym_derive(is)
symh is;
{
if (!sym[is].type) return;
if (sym[is].stat & (SSmpow|SSconst)) return;

if (sym[is].type == SymExpr)
   {
```

```
   int i, n;
   sym[is].pp = (powt **) malloc(n = n_lay*sizeof(powt*));
   memset((void*)sym[is].pp,0,n);
   for (i = 0; i < n_lay; i++)
      sym[is].pp[i] = sym_expand(i+1,SymExpr,sym[is].expr);
   sym[is].stat |= SSmpow;
   }
else
if (sym[is].type == SymInt)
   {
   powt *ps;
   real rv;
   int i, n, m;

   sym[is].pp = (powt **) malloc(n = n_lay*sizeof(powt*));
   memset((void*)sym[is].pp,0,n);
   rv = 0.;
   for (m = sym[is].iv1; m > 0; m--)
      {
      ps = sym_expand(m,SymExpr,sym[is].expr);
      sym[is].pp[m-1] = pow_integrate(ps,ga[m],rv);
      rv = pow_calc(sym[is].pp[m-1],ga[m-1]);
      pow_free(ps);
      }
   rv = 0.;
   for (m = sym[is].iv1 + 1; m <= n_lay; m++)
      {
      ps = sym_expand(m,SymExpr,sym[is].expr);
      sym[is].pp[m-1] = pow_integrate(ps,ga[m-1],rv);
      rv = pow_calc(sym[is].pp[m-1],ga[m]);
      pow_free(ps);
      }
   sym[is].stat |= SSmpow;
   }
else
if (sym[is].type == SymLamInt)
   {
   sym[is].rv = lam_int(sym[is].iv1,sym[is].iv2,sym[is].expr);
   sym[is].stat |= SSconst;
   }
}


real sym_eval(is,k,z)
symh is;
real z;
{
real ret;

if (sym[is].type == SymZ)
   return (pow(z,1.*sym[is].iv1));
else
if (sym[is].type == SymConst)
   return (sym[is].rv);

if (sym[is].stat & SSconst)
   return (sym[is].rv);
else
if (sym[is].type == SymLamInt)
   return (lam_int(sym[is].iv1,sym[is].iv2,sym[is].expr));

if (sym[is].type == SymMConst)
   return (sym[is].rp[k-1]);
```

```
if (k < 1) k = layer_number(z);
if (k > n_lay)
   {
   fprintf(fdeb,"error: eval: layer\n");
   n_err++;
   }

if (sym[is].stat & SSmpow)
   ret = pow_calc(sym[is].pp[k-1],z);
else
if (sym[is].type == SymMPow)
   ret = pow_calc(sym[is].pp[k-1],z);
else
if (sym[is].type == SymExpr)
   {
   powt *ps;
   ps = sym_expand(k,SymExpr,sym[is].expr);
   ret = pow_calc(ps,z);
   pow_free(ps);
   }
else
if (sym[is].type == SymInt)
   {
   powt *ps;
   ps = sym_expand(k,is);
   ret = pow_calc(ps,z);
   pow_free(ps);
   }
else
   {
   fprintf(fdeb,"error: eval: type\n");
   n_err++;
   }
return (ret);
}
```

Appendix D

Routines for Optimization based on Higher–Order Theory

```
/*---   higher order optimization
 *
 *      optimization based on higher order theory
 *
 *      evan summers
 *         university of natal, durban
 *
 *      dedicated symbolic computation routines
 *
 */

            /* trigonometric series */
#define  TTsn1       0x10
#define  TTcs1       0x20
#define  TTsn2       0x01
#define  TTcs2       0x02
#define  TTsc        0x12
#define  TTcc        0x22
#define  TTss        0x11
#define  TTcs        0x21

            /* differential operators */
#define  TDNone      0x0000
#define  TDMask      0x0fff
#define  TD1         0x0100
#define  TD2         0x0200
#define  TD01        0x0010
#define  TD02        0x0020
#define  TD001       0x0001
#define  TD002       0x0002
#define  TD11        0x0110
#define  TD12        0x0120
#define  TD22        0x0220
#define  TD111       0x0111
#define  TD112       0x0112
#define  TD122       0x0122
#define  TD222       0x0222

            /* coefficients for trigonometic series */
#define  Cu1         0
#define  Cu2         1
#define  Cw          2
#define  Cchig       2
#define  Cchi1       3
#define  Cchi2       4
#define  Cchi3       5
#define  Cchi4       6
#define  Cchi5       7
#define  Cchi6       8

            /* distribution functions */
#define  Ff          0x0001
#define  Ffs         0x0002

#define  FF1         0x0003
#define  FF2         0x0004
#define  FF3         0x0005

#define  Ffg         0x0005
```

```
#define  Ff1        0x0006
#define  Ff2        0x0007
#define  Ff3        0x0008
#define  Ff4        0x0009
#define  Ff5        0x000a
#define  Ff6        0x000b
#define  Ff7        0x000c
#define  Ff8        0x000d
#define  Ff9        0x000e

#define  Fbeta1     0x000f
#define  Fbeta7     0x0010
#define  Fbeta8     0x0011

#define  Falpg      0x0011
#define  Falp1      0x0012
#define  Falp2      0x0013
#define  Falp3      0x0014
#define  Falp4      0x0015
#define  Falp5      0x0016
#define  Falp6      0x0017

#define  Fvphi      0x0018
#define  Fvphig     0x0018
#define  Fvphi1     0x0019
#define  Fvphi2     0x001a
#define  Fvphi3     0x001b
#define  Fvphi4     0x001c
#define  Fvphi5     0x001d
#define  Fvphi6     0x001e
#define  Fvphi7     0x001f
#define  Fvphi8     0x0020

#define  Fpsig      0x0020
#define  Fpsi1      0x0021
#define  Fpsi2      0x0022
#define  Fpsi3      0x0023
#define  Fpsi4      0x0024
#define  Fpsi5      0x0025
#define  Fpsi6      0x0026
#define  Fpsi7      0x0027
#define  Fpsi8      0x0028

/*----- types -----*/

typedef struct /* stress and strain state at some (x,z) */
        {
        real   u1,u2,u3;
        real   e11,e12,e13,e22,e23,e33;
        real   s11,s12,s13,s22,s23,s33;
        real   e33i,s11i,s13i,s33i;
        }
        sss;

/*---- macros -----*/

                /* functions of the reference surface */
#define  xu1(x1,x2,td)        eval_trig(Cu1,  TTcs,td,x1,x2)
#define  xu2(x1,x2,td)        eval_trig(Cu2,  TTsc,td,x1,x2)
#define  xw(x1,x2,td)         eval_trig(Cw,   TTss,td,x1,x2)
#define  xchi1(x1,x2,td)      eval_trig(Cchi1,TTss,td,x1,x2)
#define  xchi2(x1,x2,td)      eval_trig(Cchi2,TTss,td,x1,x2)
#define  xchi5(x1,x2,td)      eval_trig(Cchi5,TTss,td,x1,x2)
#define  xchi6(x1,x2,td)      eval_trig(Cchi6,TTss,td,x1,x2)
```

```
#define  xchi(icf,x1,x2,td)   eval_trig(Cchig+icf,TTss,td,x1,x2)


/*---- globals -----*/

                /* geometry */
int    n_lay, m_ref;
real   gh,ga[Mn+1],gb1,gb2;
real   gk11,gk12,gk22,gk21;

                /* material properties */
real   me1[Mn],me2[Mn],mg1[Mn],mg2[Mn];
real   mnu1[Mn],mnu2[Mn];
real   me0[Mn],mnu0[Mn];
real   ma11[Mn],ma12[Mn],ma13[Mn],ma33[Mn];

                /* coefficients of distribution functions */
real   cfd[41][Mn][Mp+1];


/*----- material -----*/

void mat_depend(void) /* calculate dependent material constants */
{
int  i;
real mdel,mdel11,mdel12,mdel13,mdel33;

for (i = 0; i < n_lay; i++)
   {
   me0[i]    = me1[i]/(1 - mnu1[i]*mnu1[i]);
   mnu0[i]   = me1[i]/me2[i]*mnu2[i]/(1 - mnu1[i]);

   mdel      = (1 + mnu1[i]);
   mdel     *= (1 - mnu1[i] - 2*mnu2[i]*mnu2[i]*me1[i]/me2[i]);
   mdel     /= (me1[i]*me1[i]*me2[i]);
   mdel11    = (1 - mnu2[i]*mnu2[i]*me1[i]/me2[i])/me1[i]/me2[i];
   mdel12    = (mnu1[i] + mnu2[i]*mnu2[i]*me1[i]/me2[i]);
   mdel12   /= me1[i]/me2[i];
   mdel13    = mnu2[i]*(1 + mnu1[i])/me1[i]/me2[i];
   mdel33    = (1 - mnu1[i]*mnu1[i])/me1[i]/me1[i];

   ma11[i]   = mdel11/mdel;
   ma12[i]   = mdel12/mdel;
   ma13[i]   = mdel13/mdel;
   ma33[i]   = mdel33/mdel;
   }

}

/*----- computation of coefficients -----*/

        /* set coefficents to zero */
void  coeff_clear(real cf[Mn][Mp+1])
{
int   i,k;

for (k = 1; k <= n_lay; k++)
   for (i = 0; i <= Mp; i++)
      cf[k][i] = 0;

return;
}

/* add a term to a series
```

```
      cnst is a scalar,
      vn   is n-vector of layer parameters
      pz   is power of z
      */

void   coeff_add_term(cf,cnst,vn,pz)
real   cf[Mn][Mp+1];
real   cnst,*vn;
int    pz;
{
int   k;
real   term;

for (k = 1; k <= n_lay; k++)
   {
   term = cnst;
   if (vn) term *= vn[k-1];
   cf[k][pz] += term;
   }

return;
}

            /* add two series */
                  /* premultiply by cnst*vn/vd */
                  /* where vn, vd are layer vectors */
void   coeff_add(cf1,cnst,vn,vd,cf2)
real   cf1[Mn][Mp+1],cf2[Mn][Mp+1];
real   cnst,*vn,*vd;
{
int    i,k;
real   term;

for (k = 1; k <= n_lay; k++)
   {
   term = cnst;
   if (vn) term *= vn[k-1];
   if (vd) term /= vd[k-1];
   for (i = 0; i <= Mp; i++)
      cf1[k][i] += term*cf2[k][i];
   }

return;
}

            /* calculates definite integral of series */
                  /* from a_m to a_k */
real   coeff_lamint(m,k,cf)
real   cf[Mn][Mp+1];
int    m,k;
{
int    i,l;
real   ret = 0;
real   zpu,zpl;

for (l = m+1; l <= k; l++)
   {
   zpu = ga[l]; zpl = ga[l-1];
   for (i = 0; i <= Mp; i++)
      {
      ret += cf[l][i]*(zpu - zpl)/(i+1);
      zpu *= ga[l]; zpl *= ga[l-1];
      }
   }
```

```
return (ret);
}

            /* evaluates a series */
real    coeff_eval(cf,z)
real    cf[];
real    z;
{
int     i;
real    zp = 1, ret = 0;

for (i = 0; i <= Mp; i++)
   {
   ret += cf[i]*zp;
   zp *= z;
   }

return (ret);
}

          /* integrates a series from a_m to z */
real    coeff_layint(cf0,m,cfn)
real    cf0[Mn][Mp+1],cfn[Mn][Mp+1];
int     m;
{
int     i,l;
real    term;
real    zpu,zpl;

for (term = 0, l = m; l >= 1; l--)
   {
   cf0[l][0] = 0;
   for (i = 0; i < Mp; i++)
      cf0[l][i+1] = cfn[l][i]/(i+1);
   cf0[l][0] = term - coeff_eval(cf0[l],ga[l]);
   term = coeff_eval(cf0[l],ga[l-1]);
   }

for (term = 0, l = m+1; l <= n_lay; l++)
   {
   cf0[l][0] = 0;
   for (i = 0; i < Mp; i++)
      cf0[l][i+1] = cfn[l][i]/(i+1);
   cf0[l][0] = term - coeff_eval(cf0[l],ga[l-1]);
   term = coeff_eval(cf0[l],ga[l]);
   }

return (term);
}

/*----- distribution functions -----*/

          /* evaluates distribution function */
real    zfn(int fn, int k, real z)
{
int     i;
real    ret = 0, zp = 1;

for (i = 0; i <= 6; i++)
   {
   ret += cfd[fn][k][i]*zp;
   zp *= z;
   }
```

```
return (ret);
}

            /* derives distribution functions */
void   calc_dist(void)
{
int   i,k;
real   cf[8][Mn][Mp+1] = {0};
real   cbf,cbf1,cdf1,cdf2,cdf3;

memset((void*)cfd,0,sizeof(cfd));

coeff_add_term(cf[0],1.,me0,0);
coeff_add_term(cf[1],1.,me0,1);

cbf  = coeff_layint(cfd[Ff],0,cf[0]);
cbf1 = coeff_layint(cfd[Ffs],0,cf[1]);

coeff_add(cfd[Ff1],1.,Null,Null,cfd[Ffs]);
coeff_add(cfd[Ff1],-cbf1/cbf,Null,Null,cfd[Ff]);
coeff_add(cfd[Ff2],1/cbf,Null,Null,cfd[Ff]);
coeff_add_term(cfd[Ff2],-1.,Null,0);
coeff_add(cfd[Ff3],1/cbf,Null,Null,cfd[Ff]);

cdf1 = coeff_layint(cfd[FF1],0,cfd[Ff1]);
cdf2 = coeff_layint(cfd[FF2],0,cfd[Ff2]);
cdf3 = coeff_layint(cfd[FF3],0,cfd[Ff3]);

coeff_add(cfd[Ff4],cdf2/cdf1,Null,Null,cfd[FF1]);
coeff_add(cfd[Ff4],-1.,Null,Null,cfd[FF2]);
coeff_add(cfd[Ff5],cdf3/cdf1,Null,Null,cfd[FF1]);
coeff_add(cfd[Ff5],-1.,Null,Null,cfd[FF3]);
coeff_add(cfd[Ff6],1/cdf1,Null,Null,cfd[FF1]);
coeff_add_term(cfd[Ff6],-1.,Null,0);
coeff_add(cfd[Ff7],1/cdf1,Null,Null,cfd[FF1]);
coeff_add(cfd[Ff8],1.,Null,Null,cfd[Ff]);
coeff_add(cfd[Ff8],-cbf/cdf1,Null,Null,cfd[FF1]);
coeff_add(cfd[Ff9],1.,Null,Null,cfd[Ffs]);
coeff_add(cfd[Ff9],-cbf1/cdf1,Null,Null,cfd[FF1]);

coeff_add_term(cfd[Falp1],1.,mnu0,1);
coeff_add_term(cfd[Falp2],-1.,mnu0,0);

coeff_add(cfd[Falp3],1.,Null,me2,cfd[Ff4]);
coeff_add(cfd[Falp4],1.,Null,me2,cfd[Ff5]);
coeff_add(cfd[Falp5],1.,Null,me2,cfd[Ff6]);
coeff_add(cfd[Falp6],1.,Null,me2,cfd[Ff7]);

coeff_add(cfd[Fvphi],  1.,Null,mg2,cfd[Ff1]);
coeff_add(cfd[Fvphi7],-1.,Null,mg2,cfd[Ff2]);
coeff_add(cfd[Fvphi8],-1.,Null,mg2,cfd[Ff3]);

coeff_add(cfd[Fbeta1],1.,Null,mg2,cfd[Ff1]);
coeff_add(cfd[Fbeta7],1.,Null,mg2,cfd[Ff2]);
coeff_add(cfd[Fbeta8],1.,Null,mg2,cfd[Ff3]);

coeff_layint(cfd[Fvphi1],m_ref,cfd[Falp1]);
coeff_layint(cfd[Fvphi2],m_ref,cfd[Falp2]);
coeff_layint(cfd[Fvphi3],m_ref,cfd[Falp3]);
coeff_layint(cfd[Fvphi4],m_ref,cfd[Falp4]);
coeff_layint(cfd[Fvphi5],m_ref,cfd[Falp5]);
coeff_layint(cfd[Fvphi6],m_ref,cfd[Falp6]);
```

```
coeff_add(cf[2],1.,Null,Null,cfd[Fvphi1]);
coeff_add(cf[2],-1.,Null,Null,cfd[Fvphi]);
coeff_layint(cfd[Fpsi1],m_ref,cf[2]);

coeff_layint(cfd[Fpsi2],m_ref,cfd[Fvphi2]);
coeff_layint(cfd[Fpsi3],m_ref,cfd[Fvphi3]);
coeff_layint(cfd[Fpsi4],m_ref,cfd[Fvphi4]);
coeff_layint(cfd[Fpsi5],m_ref,cfd[Fvphi5]);
coeff_layint(cfd[Fpsi6],m_ref,cfd[Fvphi6]);
coeff_layint(cfd[Fpsi7],m_ref,cfd[Fvphi7]);
coeff_layint(cfd[Fpsi8],m_ref,cfd[Fvphi8]);
}

/*----- operators on distribution functions -----*/

          /* layer integral operator to calculate stiffnesses */
real   calc_layint(m,k,zu,vm,fn0,fn1,pz)
real   vm[];
real   zu;
int    m,k;
int    fn0,fn1;
int    pz;
{
int    i,j,l;
int    np;
real   cf[16];
real   term,ret = 0;
real   zpu,zpl;

for (l = m+1; l <= k; l++)
   {
   for (i = 0; i < 16; i++) cf[i] = 0;

   if (fn0 == 0) {cf[0] = 1; np = 0;} else
   if (fn1 == 0)
      {
      for (i = 0; i <= 6; i++)
         cf[i] = cfd[fn0][l][i];
      np = 6;
      }
   else
      {
      for (i = 0; i <= 6; i++)
         for (j = 0; j <= 6; j++)
          cf[i+j] += cfd[fn0][l][i]*cfd[fn1][l][j];
      np = 12;
      }

   if (pz)
      {
      for (i = 12; i >= 0; i--) cf[i+pz] = cf[i];
      for (i = 0; i < pz; i++) cf[i] = 0;
      np += pz;
      }

   zpl = ga[l-1];
   if (l == k) zpu = zu; else zpu = ga[l];
   for (term = 0, i = 0; i <= np; i++)
      {
      term += cf[i]*(zpu - zpl)/(i+1);
      zpl *= ga[l-1];
      if (l == k) zpu *= zu; else zpu *= ga[l];
      }
```

```
   if (vm) term *= vm[l-1];

   ret += term;
   }

return (ret);
}
        /* definite layer integral from a_0 to a_n */
real   calc_lamint(vm,fn0,fn1,pz)
real   vm[];
{
int    i,j,np,k;
real   cf[16];
real   term,ret = 0;
real   zpu,zpl;

for (k = 1; k <= n_lay; k++)
   {
   for (i = 0; i < 16; i++) cf[i] = 0;

   if (fn0 == 0) {cf[0] = 1; np = 0;} else
   if (fn1 == 0)
      {
      for (i = 0; i <= 6; i++)
         cf[i] = cfd[fn0][k][i];
      np = 6;
      }
   else
      {
      for (i = 0; i <= 6; i++)
         for (j = 0; j <= 6; j++)
          cf[i+j] += cfd[fn0][k][i]*cfd[fn1][k][j];
      np = 12;
      }

   if (pz)
      {
      for (i = 12; i >= 0; i--) cf[i+pz] = cf[i];
      for (i = 0; i < pz; i++) cf[i] = 0;
      np += pz;
      }

   zpu = ga[k]; zpl = ga[k-1];
   for (term = 0, i = 0; i <= np; i++)
      {
      term += cf[i]*(zpu - zpl)/(i+1);
      zpu *= ga[k]; zpl *= ga[k-1];
      }

   if (vm) term *= vm[k-1];

   ret += term;
   }

return (ret);
}

/*----- integrated stiffnesses -----*/

#define  cbg(g)       calc_lamint(ma11,Fpsig+g,0,0)
#define  cbgb(g)      calc_lamint(ma12,Fpsig+g,0,0)
#define  ccg(g)       calc_lamint(ma11,Fvphig+g,0,0)
#define  ccgb(g)      calc_lamint(ma12,Fvphig+g,0,0)
#define  cc0g(g)      calc_lamint(ma11,Fvphig+g,0,1)
```

```
#define  cc0gb(g)     calc_lamint(ma12,Fvphig+g,0,1)
#define  ccgq(g,q)    calc_lamint(ma11,Fpsig+g,Fvphig+q,0)
#define  ccgqb(g,q)   calc_lamint(ma12,Fpsig+g,Fvphig+q,0)
#define  cd0g(g)      calc_lamint(ma11,Fpsig+g,0,1)
#define  cd0gb(g)     calc_lamint(ma12,Fpsig+g,0,1)
#define  cdgq(g,q)    calc_lamint(ma11,Fpsig+g,Fpsig+q,0)
#define  cdgqb(g,q)   calc_lamint(ma12,Fpsig+g,Fpsig+q,0)
#define  chg(g)       calc_lamint(ma13,Falpg+g,0,0)
#define  ch0g(g)      calc_lamint(ma13,Falpg+g,0,1)
#define  chgq(g,q)    calc_lamint(ma13,Fpsig+g,Falpg+q,0)
#define  clgq(g,q)    calc_lamint(ma11,Fvphig+g,Fvphig+q,0)
#define  clgqb(g,q)   calc_lamint(ma12,Fvphig+g,Fvphig+q,0)
#define  cpgq(g,q)    calc_lamint(ma13,Fvphig+g,Falpg+q,0)
#define  cpgqb(g,q)   calc_lamint(ma13,Falpg+g,Fpsig+q,0)
#define  crgq(g,q)    calc_lamint(ma33,Falpg+g,Fvphig+q,0)
#define  crgqb(g,q)   calc_lamint(ma33,Falpg+g,Falpg+q,0)
#define  ctgq(g,q)    calc_lamint(ma11,Fvphig+g,Fpsig+q,0)
#define  ctgqb(g,q)   calc_lamint(ma12,Fvphig+g,Fpsig+q,0)

        /* calculates integrated stiffnesses */
void   calc_stiff()
{
cb    = calc_lamint(ma11,0,0,0);
cbb   = calc_lamint(ma12,0,0,0);
cb0   = calc_lamint(ma11,0,0,1);
cb0b  = calc_lamint(ma12,0,0,1);
cd00  = calc_lamint(ma11,0,0,2);
cd00b = calc_lamint(ma12,0,0,2);
cd1   = calc_lamint(Null,Ff1,Fbeta1,0);

cb1 = cbg(1); cb1b = cbgb(1);
cb2 = cbg(2); cb2b = cbgb(2);
cb3 = cbg(3); cb3b = cbgb(3);
cb4 = cbg(4); cb4b = cbgb(4);
cb5 = cbg(5); cb5b = cbgb(5);
cb6 = cbg(6); cb6b = cbgb(6);

ch1 = chg(1); ch01 = ch0g(1); cd01 = cd0g(1);
ch2 = chg(2); ch02 = ch0g(2); cd02 = cd0g(2);
ch3 = chg(3); ch03 = ch0g(3); cd03 = cd0g(3);
ch4 = chg(4); ch04 = ch0g(4); cd04 = cd0g(4);
ch5 = chg(5); ch05 = ch0g(5); cd05 = cd0g(5);
ch6 = chg(6); ch06 = ch0g(6); cd06 = cd0g(6);

cc1 = ccg(1); cc1b = ccgb(1); cc01 = cc0g(1); cc01b = cc0gb(1);
cc2 = ccg(2); cc2b = ccgb(2); cc02 = cc0g(2); cc02b = cc0gb(2);
cc3 = ccg(3); cc3b = ccgb(3); cc03 = cc0g(3); cc03b = cc0gb(3);
cc4 = ccg(4); cc4b = ccgb(4); cc04 = cc0g(4); cc04b = cc0gb(4);
cc5 = ccg(5); cc5b = ccgb(5); cc05 = cc0g(5); cc05b = cc0gb(5);
cc6 = ccg(6); cc6b = ccgb(6); cc06 = cc0g(6); cc06b = cc0gb(6);

cc11 = ccgq(1,1); cc11b = ccgqb(1,1);
cc12 = ccgq(1,2); cc12b = ccgqb(1,2);
cc21 = ccgq(2,1); cc21b = ccgqb(2,1);
cc22 = ccgq(2,2); cc22b = ccgqb(2,2);
cc15 = ccgq(1,5); cc15b = ccgqb(1,5);
cc16 = ccgq(1,6); cc16b = ccgqb(1,6);
cc25 = ccgq(2,5); cc25b = ccgqb(2,5);
cc26 = ccgq(2,6); cc26b = ccgqb(2,6);

ch11 = chgq(1,1); cd11 = cdgq(1,1);
ch12 = chgq(1,2); cd12 = cdgq(1,2);
ch21 = chgq(2,1);
ch22 = chgq(2,2); cd22 = cdgq(2,2);
```

```
ch15 = chgq(1,5); cd15 = cdgq(1,5);
ch16 = chgq(1,6); cd16 = cdgq(1,6);
ch25 = chgq(2,5); cd25 = cdgq(2,5);
ch26 = chgq(2,6); cd26 = cdgq(2,6);

cl11 = clgq(1,1); cl11b = clgqb(1,1);
cl12 = clgq(1,2); cl12b = clgqb(1,2);
cl21 = clgq(2,1); cl21b = clgqb(2,1);
cl22 = clgq(2,2); cl22b = clgqb(2,2);
cl15 = clgq(1,5); cl15b = clgqb(1,5);
cl16 = clgq(1,6); cl16b = clgqb(1,6);
cl25 = clgq(2,5); cl25b = clgqb(2,5);
cl26 = clgq(2,6); cl26b = clgqb(2,6);

cp11 = cpgq(1,1); cp11b = cpgqb(1,1);
cp12 = cpgq(1,2); cp12b = cpgqb(1,2);
cp21 = cpgq(2,1); cp21b = cpgqb(2,1);
cp22 = cpgq(2,2); cp22b = cpgqb(2,2);
cp15 = cpgq(1,5); cp15b = cpgqb(1,5);
cp16 = cpgq(1,6); cp16b = cpgqb(1,6);
cp25 = cpgq(2,5); cp25b = cpgqb(2,5);
cp26 = cpgq(2,6); cp26b = cpgqb(2,6);

cr11 = crgq(1,1); cr11b = crgqb(1,1);
cr12 = crgq(1,2); cr12b = crgqb(1,2);
cr21 = crgq(2,1);
cr22 = crgq(2,2); cr22b = crgqb(2,2);
cr15 = crgq(1,5); cr15b = crgqb(1,5);
cr16 = crgq(1,6); cr16b = crgqb(1,6);
cr25 = crgq(2,5); cr25b = crgqb(2,5);
cr26 = crgq(2,6); cr26b = crgqb(2,6);

ct11 = clgq(1,1); cl11b = ctgqb(1,1);
ct12 = clgq(1,2); cl12b = ctgqb(1,2);
ct21 = clgq(2,1); cl21b = ctgqb(2,1);
ct22 = clgq(2,2); cl22b = ctgqb(2,2);
ct15 = clgq(1,5); cl15b = ctgqb(1,5);
ct16 = clgq(1,6); cl16b = ctgqb(1,6);
ct25 = clgq(2,5); cl25b = ctgqb(2,5);
ct26 = clgq(2,6); cl26b = ctgqb(2,6);
}

/*----- governing equations -----*/

          /* system of equations for plate */
void   sys_plate(tc,qc,wm,wn,q3p,q3m)
real   tc[5][5],qc[5];
int    wm,wn;
real   q3p,q3m;
{
real   lam,lam_2,lam_3,lam_4;
real   gam,gam_2,gam_3,gam_4;

lam = wm*M_PI/gb1;
gam = wn*M_PI/gb2;

lam_2 = lam*lam; lam_3 = lam_2*lam; lam_4 = lam_3*lam;
gam_2 = gam*gam; gam_3 = gam_2*gam; gam_4 = gam_3*gam;

tc[0][0] = -(cb*gam_2)/2 + (cbb*gam_2)/2 - cb*lam_2;
tc[0][1] = -(cb*gam*lam)/2 - (cbb*gam*lam)/2;
tc[0][2] = cb0*gam_2*lam + cb0*lam_3;
tc[0][3] = ch1*lam + cb1*gam_2*lam + cb1*lam_3;
tc[0][4] = ch2*lam + cb2*gam_2*lam + cb2*lam_3;
```

```
tc[1][0] = -(cb*gam*lam)/2 - (cbb*gam*lam)/2;
tc[1][1] = -(cb*gam_2) - (cb*lam_2)/2 + (cbb*lam_2)/2;
tc[1][2] = cb0*gam_3 + cb0*gam*lam_2;
tc[1][3] = ch1*gam + cb1*gam_3 + cb1*gam*lam_2;
tc[1][4] = ch2*gam + cb2*gam_3 + cb2*gam*lam_2;
tc[2][0] = -(cb0*gam_2*lam) - cb0*lam_3;
tc[2][1] = -(cb0*gam_3) - cb0*gam*lam_2;
tc[2][2] = cd00*gam_4 + 2*cd00*gam_2*lam_2 + cd00*lam_4;
tc[2][3] = ch01*gam_2 + cd01*gam_4 + ch01*lam_2
             + 2*cd01*gam_2*lam_2 + cd01*lam_4;
tc[2][4] = ch02*gam_2 + cd02*gam_4 + ch02*lam_2
             + 2*cd02*gam_2*lam_2 + cd02*lam_4;
tc[3][0] = -(ch1*lam) - cb1*gam_2*lam - cb1*lam_3;
tc[3][1] = -(ch1*gam) - cb1*gam_3 - cb1*gam*lam_2;
tc[3][2] = ch01*gam_2 + cd01*gam_4 + ch01*lam_2
             + 2*cd01*gam_2*lam_2 + cd01*lam_4;
tc[3][3] = cr11b + (2*ch11 + cd1)*(lam_2 + gam_2)
             + cd11*gam_4 + 2*cd11*gam_2*lam_2 + cd11*lam_4;
tc[3][4] = cr12b + (ch12 + cp12b)*(lam_2 + gam_2)
             + cd12*gam_4 + 2*cd12*gam_2*lam_2 + cd12*lam_4;
tc[4][0] = -(ch2*lam) - cb2*gam_2*lam - cb2*lam_3;
tc[4][1] = -(ch2*gam) - cb2*gam_3 - cb2*gam*lam_2;
tc[4][2] = ch02*gam_2 + cd02*gam_4 + ch02*lam_2
             + 2*cd02*gam_2*lam_2 + cd02*lam_4;
tc[4][3] = cr21b + (ch21 + cp21b)*(lam_2 + gam_2)
             + cd21*gam_4 + 2*cd21*gam_2*lam_2 + cd21*lam_4;
tc[4][4] = cr22b + (ch22 + cp22b)*(lam_2 + gam_2)
             + cd22*gam_4 + 2*cd22*gam_2*lam_2 + cd22*lam_4;

qc[0] = -(q3m*ch5*lam) - q3p*ch6*lam - q3m*cb5*gam_2*lam
          - q3p*cb6*gam_2*lam - q3m*cb5*lam_3 - q3p*cb6*lam_3;
qc[1] = -(q3m*ch5*gam) - q3p*ch6*gam - q3m*cb5*gam_3
          - q3p*cb6*gam_3
          - q3m*cb5*gam*lam_2 - q3p*cb6*gam*lam_2;
qc[2] = q3m + q3p - q3m*ch05*gam_2 - q3p*ch06*gam_2
          - q3m*cd05*gam_4 - q3p*cd06*gam_4 - q3m*ch05*lam_2
          - q3p*ch06*lam_2 - 2*q3m*cd05*gam_2*lam_2
          - 2*q3p*cd06*gam_2*lam_2
          - q3m*cd05*lam_4 - q3p*cd06*lam_4;
qc[3] = q3m*(zfn(Fvphi1,1,ga[0]) - cr15b)
          + q3p*(zfn(Fvphi1,n_lay,ga[n_lay]) - cr16b)
          - q3m*(ch15 + cp15b)*(lam_2+ gam_2)
          - q3m*cd15*gam_4 - q3p*cd16*gam_4
          - q3p*(ch16 + cp16b)*(lam_2 + gam_2)
          - 2*q3m*cd15*gam_2*lam_2
          - 2*q3p*cd16*gam_2*lam_2
          - q3m*cd15*lam_4 - q3p*cd16*lam_4;
qc[4] = q3m*(zfn(Fvphi2,1,ga[0]) - cr25b)
          + q3p*(zfn(Fvphi2,n_lay,ga[n_lay]) - cr26b)
          - q3m*(ch25 + cp25b)*gam_2 - q3p*(ch26 + cp26b)*gam_2
          - q3m*cd25*gam_4 - q3p*cd26*gam_4
          - q3m*(ch25 + cp25b)*lam_2
          - q3p*(ch26 + cp26b)*lam_2
          - 2*q3m*cd25*gam_2*lam_2
          - 2*q3p*cd26*gam_2*lam_2
          - q3m*cd25*lam_4 - q3p*cd26*lam_4;
}

int    sys_solve_coeff(is,dc,wm,wn,q3p,q3m)
int    is;
real   dc[];
int    wm,wn;
real   q3p,q3m;
{
```

```
int    i,j,l;
real   tc[5][5],qc[5],minv[25];
int    nc;
real   det;

if (is == 1)
   sys_shell(tc,qc,wm,wn,q3p,q3m);
else
   sys_plate(tc,qc,wm,wn,q3p,q3m);

for (i = 0; i < 5; i++)
   for (j = 0; j < 5; j++)
      tc[i][j] = real_sig(tc[i][j]);

for (nc = 2; nc < 5; nc++)
   {
   for (j = 0; j <= nc; j++)
      {
      if (real_chksig(tc[nc][j])) break;
      if (real_chksig(tc[j][nc])) break;
      }
   if (j > nc) break;
   }

for (i = nc; i < 5; i++)
   {
   dc[i] = 0;
   if (real_chksig(qc[i])) break;
   }

if (i == 5 && SolveSysGJ((real*)tc,5,qc,dc,nc,minv,&det))
   {
   return (1);
   }

lprintf("error: no solution: %d dimensions\n",nc);
return (0);
}

int    sys_solve(int is)
{
int    i;
int    ret;

for (i = 0; dcmn[i][0]; i++)
   {
   dc[i][Cchi5] = q3m[i];
   dc[i][Cchi6] = q3p[i];
   }

for (i = 0; dcmn[i][0]; i++)
   {
   ret = sys_solve_coeff(is,dc[i],
               dcmn[i][0],dcmn[i][1],q3p[i],q3m[i]);
   if (!ret) break;
   }

return (ret);
}

/*----- trigonemtric series -----*/

          /* evaluate term in double trigonometric series */
               /* tt is type of series */
```

```
                /* td is differential operator */
real   eval_trig_term(wm,wn,tt,td,x1,x2)
real   x1,x2;
int    wm,wn;
int    tt,td;
{
real   lam,gam;
real   ret = 1.;

lam = wm*M_PI/gb1;
gam = wn*M_PI/gb2;

if (td & TD1)
   {
   ret *= lam;
   if (tt & TTsn1) tt = (tt&~TTsn1)|TTcs1;
   else {ret = -ret; tt = (tt&~TTcs1)|TTsn1;}
   }
else
if (td & TD2)
   {
   ret *= gam;
   if (tt & TTsn2) tt = (tt&~TTsn2)|TTcs2;
   else {ret = -ret; tt = (tt&~TTcs2)|TTsn2;}
   }
if (td & TD01)
   {
   ret *= lam;
   if (tt & TTsn1) tt = (tt&~TTsn1)|TTcs1;
   else {ret = -ret; tt = (tt&~TTcs1)|TTsn1;}
   }
else
if (td & TD02)
   {
   ret *= gam;
   if (tt & TTsn2) tt = (tt&~TTsn2)|TTcs2;
   else {ret = -ret; tt = (tt&~TTcs2)|TTsn2;}
   }
if (td & TD001)
   {
   ret *= lam;
   if (tt & TTsn1) tt = (tt&~TTsn1)|TTcs1;
   else {ret = -ret; tt = (tt&~TTcs1)|TTsn1;}
   }
else
if (td & TD002)
   {
   ret *= gam;
   if (tt & TTsn2) tt = (tt&~TTsn2)|TTcs2;
   else {ret = -ret; tt = (tt&~TTcs2)|TTsn2;}
   }

if (tt == TTsc) ret *= sin(lam*x1)*cos(gam*x2);
if (tt == TTss) ret *= sin(lam*x1)*sin(gam*x2);
if (tt == TTcs) ret *= cos(lam*x1)*sin(gam*x2);
if (tt == TTcc) ret *= cos(lam*x1)*cos(gam*x2);

return (ret);
}
          /* evaluate
real   eval_trig(icf,tt,td,x1,x2)
int    icf;
real   x1,x2;
int    tt,td;
```

```
{
int    i;
real   ret = 0;

for (i = 0; dcmn[i][0]; i++)
   ret += dc[i][icf]*eval_trig_term(dcmn[i][0],
               dcmn[i][1],tt,td,x1,x2);

return (ret);
}

/*----- stress and strain -----*/

               /* alternative formulation for transverse shear */
real   eval_s13(x1,x2,k,z)
real   x1,x2,z;
int    k;
{
int    i;
real   beta,betas,beta0s,beta0b;
real   betags,nugs,nugbs,thegs;
real   betagb[8],nug[8],nugb[8],theg[8];
real   cbg[8],chg[8],ccg[8],ccgb[8];
real   term, ret;

betas  = calc_layint(0,k,z,ma12,0,0,0);
beta0s = calc_layint(0,k,z,ma11,0,0,1);
beta   = betas/cbb;
beta0b = cb0*beta - beta0s;

cbg[0] = cb1; chg[0] = ch1; ccg[0] = cc1; ccgb[0] = cc1b;
cbg[1] = cb2; chg[1] = ch2; ccg[1] = cc2; ccgb[1] = cc2b;
cbg[2] = cb3; chg[2] = ch3; ccg[2] = cc3; ccgb[2] = cc3b;
cbg[3] = cb4; chg[3] = ch4; ccg[3] = cc4; ccgb[3] = cc4b;
cbg[4] = cb5; chg[4] = ch5; ccg[4] = cc5; ccgb[4] = cc5b;
cbg[5] = cb6; chg[5] = ch6; ccg[5] = cc6; ccgb[5] = cc6b;

for (i = 0; i < 6; i++)
   {
   betags = calc_layint(0,k,z,ma11,Fpsi1+i,0,0);
   nugs   = calc_layint(0,k,z,ma11,Fvphi1+i,0,0);
   nugbs  = calc_layint(0,k,z,ma12,Fvphi1+i,0,0);
   thegs  = calc_layint(0,k,z,ma13,Falp1+i,0,0);

   betagb[i] = cbg[i]*beta - betags;
   nug[i]    = ccg[i]*beta - nugs;
   nugb[i]   = ccgb[i]*beta - nugbs;
   theg[i]   = chg[i]*beta - thegs;
   }
ret = -(xw(x1,x2,TD111) + xw(x1,x2,TD122))*beta0b;
for (i = 0; i < 6; i++)
   ret -= (xchi(i+1,x1,x2,TD111) + xchi(i+1,x1,x2,TD122))*betagb[i];

for (i = 0; i < 6; i++)
   {
   ret += xchi(i+1,x1,x2,TD1)*(gk11*nug[i] + gk22*nugb[i]);
   ret += xchi(i+1,x1,x2,TD2)*gk12*(nug[i] - nugb[i]);
   ret += xchi(i+1,x1,x2,TD1)*theg[i];
   }

return (ret);
}

         /* evaluates stress/strain values */
```

```
void   eval_ss(ss,x1,x2,k,z)
sss    *ss;
real   x1,x2,z;
int    k;
{
int    i;
real   xer11,xer12,xer22;
real   xkr11,xkr12,xkr22;

ss->u1 = xu1(x1,x2,0) - xw(x1,x2,TD1)*z;
for (i = 1; i <= 6; i++)
   ss->u1 += -xchi(i,x1,x2,TD1)*zfn(Fpsig+i,k,z);
ss->u2 = xu2(x1,x2,0) - xw(x1,x2,TD2)*z;
for (i = 1; i <= 6; i++)
   ss->u2 += -xchi(i,x1,x2,TD2)*zfn(Fpsig+i,k,z);
ss->u3 = xw(x1,x2,0);
for (i = 1; i <= 6; i++)
   ss->u3 += xchi(i,x1,x2,0)*zfn(Fvphig+i,k,z);

xer11 = (xu1(x1,x2,TD1) + xu1(x1,x2,TD1))/2 + gk11*xw(x1,x2,0);
xer12 = (xu1(x1,x2,TD2) + xu2(x1,x2,TD1))/2 + gk12*xw(x1,x2,0);
xer22 = (xu2(x1,x2,TD2) + xu2(x1,x2,TD2))/2 + gk22*xw(x1,x2,0);

xkr11 = -xw(x1,x2,TD11);
xkr12 = -xw(x1,x2,TD12);
xkr22 = -xw(x1,x2,TD22);

ss->e11  = xer11 + xkr11*z;
for (i = 1; i <= 6; i++)
   ss->e11 += -xchi(i,x1,x2,TD11)*zfn(Fpsig+i,k,z);
for (i = 1; i <= 6; i++)
   ss->e11 += gk11*xchi(i,x1,x2,0)*zfn(Fvphig+i,k,z);

ss->e22  = xer22 + xkr22*z;
for (i = 1; i <= 6; i++)
   ss->e22 += -xchi(i,x1,x2,TD22)*zfn(Fpsig+i,k,z);
for (i = 1; i <= 6; i++)
   ss->e22 += gk22*xchi(i,x1,x2,0)*zfn(Fvphig+i,k,z);

ss->e12  = xer12 + xkr12*z;
for (i = 1; i <= 6; i++)
   ss->e12 += -xchi(i,x1,x2,TD12)*zfn(Fpsig+i,k,z);
for (i = 1; i <= 6; i++)
   ss->e12 += gk12*xchi(i,x1,x2,0)*zfn(Fvphig+i,k,z);

ss->e13  = xchi(1,x1,x2,TD1)*zfn(Fbeta1,k,z);
ss->e13 += xchi(7,x1,x2,TD1)*zfn(Fbeta7,k,z);
ss->e13 += xchi(8,x1,x2,TD1)*zfn(Fbeta8,k,z);
ss->e13 /= 2;

ss->e23  = xchi(1,x1,x2,TD2)*zfn(Fbeta1,k,z);
ss->e23 += xchi(7,x1,x2,TD2)*zfn(Fbeta7,k,z);
ss->e23 += xchi(8,x1,x2,TD2)*zfn(Fbeta8,k,z);
ss->e23 /= 2;

ss->e33  = 0;
for (i = 1; i <= 6; i++)
   ss->e33 += xchi(i,x1,x2,0)*zfn(Falpg+i,k,z);

ss->s11   = ma11[k-1]*ss->e11 + ma12[k-1]*ss->e22 + ma13[k-1]*ss->e33;
ss->s22   = ma12[k-1]*ss->e11 + ma11[k-1]*ss->e22 + ma13[k-1]*ss->e33;
ss->s33   = ma13[k-1]*ss->e11 + ma13[k-1]*ss->e22 + ma33[k-1]*ss->e33;

ss->s12   = 2*mg2[k-1]*ss->e12;
```

```
ss->s13   = 2*mg2[k-1]*ss->e13;
ss->s23   = 2*mg2[k-1]*ss->e23;

ss->s13i  = eval_s13(x1,x2,k,z);

ss->s33i  = xchi5(x1,x2,0)*zfn(Falp5,k,z)*me2[k-1];
ss->s33i += xchi6(x1,x2,0)*zfn(Falp6,k,z)*me2[k-1];

ss->e33i  = (ss->s33i - (ss->s11 + ss->s22)*mnu2[k-1])/me2[k-1];
ss->s11i  = ma11[k-1]*ss->e11 + ma12[k-1]*ss->e22 + ma13[k-1]*ss->e33i;
}
```