HIERARCHICAL PLATE AND SHELL ELEMENT INCORPORATING SYMBOLIC COMPUTATIONS

SETHURAMALINGAM SUBBARAYALU

HIERARCHICAL PLATE AND SHELL ELEMENT INCORPORATING SYMBOLIC COMPUTATIONS

by

©Sethuramalingam Subbarayalu, B.E.

A thesis submitted to the school of graduate

studies in partial fulfilment of the

requirements for the degree of

Master of Engineering

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

December,1999

St.John's Newfoundland Canada

Abstract

A hierarchical nine node p-version curved shell finite element is developed incorporating symbolic computations. The element has five nodal degrees of freedom, three translations and two rotations. The displacement approximation functions which are hierarchical in nature are derived from the Lagrangian functions. The hierarchical finite elements have a distinct advantage of saving computational effort in comparison with h-version elements. However, as the order of the displacement polynomial increases, the number of gaussian points required for integration have to be increased to obtain element matrices. This increases the computational effort required for element generation. The nature of hierarchical formulation offers certain avenues for the usage of symbolic computations which substantially reduces the computational effort involved in the element generation. A number of locations where the usage of symbolic computations offers significant reduction in computational effort are identified and are incorporated. The problems associated with the development of finite element codes can be successfully addressed by the usage of Object Oriented Programming(OOP) techniques. A Finite element program for the shell element is developed using this OOP technique. The performance of the present element is demonstrated using various numerical examples.

Acknowledgements

I express my sincere thanks to my research advisor Dr.K.Munaswamy for his invaluable guidance, financial support and encouragement provided during the course of my research work.

I am thankful to Dr. R.Seshadri, Dean, Faculty of Engineering and Applied Science for his advice and support during my study. I would like to thank Dr.M.R.Haddara, Associate Dean, Graduate Studies and Research for the guidance provided during my graduate program at Memorial University.

I am grateful to the School of Graduate Studies and the Faculty of Engineering and Applied Science for providing financial assistance for the research work.

Many thanks to my parents and friends for their love and moral support. I would also like to thank Friends of India Association for their kind help during my stay in St.John's.

Contents

List of Figures

List of Tables

Symbols and Abbreviations

FE	Finite Element
FEA	Finite Element Analysis
[n]	Reference Number n
2D,3D	Two dimensional, three dimensional
CPU	Central Processing Unit
P-	Hierarchical-
{}	Column Vector
[]	Matrix
< >	Row vector
DOF	Degrees of Freedom
OOP	Object Oriented Programming
MUNSET	Finite Element program developed
SS	Simply Supported
CL	Concentrated Load
UDL	Uniformly Distributed Load

Chapter 1

INTRODUCTION

1.1 Finite Element Method

The finite element method is a computational technique for obtaining approximate numerical solution. It is used for solving physical systems subjected to external influences. The application of this method is widely used in the areas of solid mechanics, heat transfer, fluid mechanics, acoustics and electromagnetism. The steps involved in the finite element method are : the discretization of the domain, evaluation of element properties, assembly of element properties to obtain system properties and the effective solution of the resulting system of equations. These steps are the same for all types of problems.

The discretization process divides the domain into small units, each represented by an element. The discretization is suitably carried out to improve the accuracy and convergence of the solution. The density of the elements at a location in the domain depends upon the geometry and the external load distribution. A sub-domain where there is a complex geometry and sharp edges or stress raisers, needs a finer mesh

i.e higher element density. The discretization should be optimal. It should not lead to too many elements, which increases computational effort. Once the elements are created, element matrices are calculated and then assembled. The resulting system of equations is solved to obtain the solution.

It is very important in FEA to determine the accuracy of the analysis. This is done through convergence studies. There are two ways in which the convergence studies are carried out. The common method is subdivision of the original mesh to a finer mesh, which increases the number of elements and decreases the size of the element. This method is called h-refinement or h-extension. The letter h refers to the size of the element and hence the name h-refinement. The h-refinement is generally carried out at places where there is a higher stress gradient, especially at areas of stress concentrations and stress singularities. Another approach is to increase the order of the approximation polynomial for the unknown displacement field variables in the element. This is called p-refinement or p-extension. The letter p refers to the order of polynomial used for the approximation. The p-refinement can be done for the whole domain or selectively for few elements where there is a higher stress gradient.

1.2 P-Version FEM

P-version refinement can be done in two ways. One way is using the elements that use regular interpolation functions of higher order and the other way is using the elements that use hierarchical interpolation functions or shape functions. The hierarchical

shape functions have a property that the lower order approximation functions forms the subset of higher order functions and hence only the element matrices required for the additional degrees of freedom need to be evaluated and assembled, thus reducing the computational effort to a large extent. The higher order elements with non-hierarchical shape functions do not have the above said property. This necessitates the element matrices to be evaluated, assembled and solved afresh. In this work hierarchical approximation functions are used and p-version refers to hierarchical finite elements only.

The hierarchical elements have many advantages over the h-version. The p-version elements have better convergence and the mesh design is less critical because there is always a possibility to increase the element order without altering the mesh divisions. In the case of the h-version, mesh modification by division is necessary to achieve the required convergence. In the p-version, the size of the input file is smaller because there are fewer elements. These advantages contribute to the reduction in computational effort and time. Moreover, lower order element matrices need not be evaluated again as they are a subset of the higher order element matrices. The resulting matrices are better conditioned and hence they converge faster when iterative solvers are used for solving. The higher order matrices appear as the perturbation of lower order matrices and this property has a further advantage of providing an immediate estimate of the error by comparing successive solutions.

1.3 Plate and Shell Analysis

Plates and curved structural elements in the form of general shells are common in engineering practice. They can be found in roof structures, pressure vessels, nuclear reactors and space crafts. Hence, a significant effort has been directed to the development of a suitable finite element procedure for the analysis of general shell structures. Over the years, hundreds of elements have been developed. The development of the hierarchical concept has generated a great deal of interest and many research papers have been published. Its advantages over the h-version elements are well documented. Currently, a lot of research work is being carried out in the development of accurate and computationally efficient p-version shell elements.

A number of plate elements are available in the literature which do not use the concept of hierarchical analysis. These elements employ h-version analysis for convergence studies which demands more computational effort. Moreover many of these elements suffer from a problem called "Shear locking". These elements become too stiff and produce displacements far less than the actual value when the the thickness is small. This problem is overcome chiefly by modifying the transverse shear strain field, which is cumbersome and involves additional computational effort.

1.4 Symbolic Computation

The development of symbolic manipulation packages like MAPLE, MACSYMA, MATHEMATICA etc have helped to reduce much of the manual tedium involved in the symbolic manipulation of lengthy expressions. Symbolic packages have the ability to carry out various mathematical operations including integration and differentiation symbolically . This approach can be used for computing the finite element matrices, for implementation in a finite element code.

Few elements which use the hybrid/mixed formulations are suitable for carrying out the direct integration of the element stiffness matrix. Because of the complexities and manual tedium involved, this was not done in the past. The development of symbolic tools have helped in the direct integration of the element matrices. This circumvents the time consuming gaussian integration process. Moreover symbolic computations reduce the number of redundant calculations involved and simplify many calculations. These qualities of symbolic computation decreases the computational effort enormously.

In situations where symbolic integration cannot be done, due to the nature of the functions to be integrated, numerical integrations are carried out. In such cases, the stiffness coefficients are expressed symbolically which are functions of the nodal coordinates. These stiffness expressions are integrated at the gaussian points to get the stiffness matrix. In using symbolic computations, care should be taken that

there are no redundant calculations or expression growth which will increase the computational effort rather than decreasing it.

1.5 Object Oriented Programming (OOP)

The object oriented approach is the latest trend in programming practice. The finite element program developed using conventional programming practices has many inherent disadvantages. These programs, once written, are very difficult to modify later. A small modification may require the whole program to be revised. This potentially introduces even more bugs. They also lack; ease of maintenance, verification, portability and reliability. The development of large scale FE code poses many more challenges which cannot be met by conventional programming methods. The OOP has various powerful concepts like data encapsulation, inheritance, polymorphism and dynamic memory allocations, which successfully address various problems in the development of FE code.

In traditional programming languages like FORTRAN, data and subroutines are separate entities and are connected only by passing the data to a subroutine when it is called. In the case of object oriented programming languages, like C++, the data and the procedures or subroutines are intrinsically linked. The fundamentals of object oriented programming are explained in the next chapter.

1.6 Objective of the thesis

The objective of this work is to,

1. Develop a hierarchical shell element incorporating symbolic computations.

2. Show the advantages of symbolic computation in the development of the finite element code.

3. Show the merits of developing finite element code using object oriented programming.

To achieve these objectives, the major requirement is the development of a finite element program for the developed shell element. The MUNSET program is developed incorporating the symbolic computations using the object oriented programming concept. A number of numerical problems are solved to show the accuracy and superiority of the element.

1.7 Layout of the thesis

The first chapter gives an introduction to the various concepts and terminologies relevant to the current work. Chapter two gives a detailed review of the literature and defines the scope of the study. Chapter three gives the formulation part of the p-version shell element. The derivation of various equations and matrices using symbolic computations are discussed in chapter four. It also includes an account about the computational effectiveness in incorporating symbolic computations. The

computer implementation of the formulation using the object oriented approach are discussed in chapter five. Chapter six present the numerical results obtained from the analysis of test problems. Conclusion and recommendations are given in chapter seven.

Chapter 2

BACKGROUND AND SCOPE OF WORK

2.1 Literature Review

Finite element analysis is widely applied in many fields of engineering. Almost all finite element problems include the following steps. 1. Data input 2. Calculation of element stiffness matrix and load vectors. 3. Assembly of global stiffness and load matrices. 4. Application of boundary conditions. 5. Solution of equations. 6. Post processing the solutions and results output.

These fundamentals are very well explained in the textbooks by many authors like Zienkiwicz [2], Bathe [3] etc. The following sections give a detailed account of literature relevant to the current work.

2.1.1 Plate and shell elements

The wide spread use of shell structures have created a need for systematic method of analysis which can account for arbitrary geometric form, boundary conditions as well

as arbitrary loading. The classical method of solving differential equations for a particular problem is impossible for such cases. The finite element method was first used for shell analysis in 1960 [4]. The basic concept of the finite element method is the idealization of the continuum as an assemblage of discrete structural elements. Over the years, hundreds of plate and shell elements have been developed [5]. They can be put into three categories depending upon the mathematical principles employed.

1. Flat or facet elements

In this kind of formulation, the shell surface is approximated by an assembly of flat elements [6, 2]. The behavior of the shell is modelled by superposition of bending and stretching behavior. With the use of flat elements, a large number of elements must invariably be used and hence is disadvantageous in terms of computational effort and accuracy. Thus there is a need for elements which can take up curved shape.

2. Curved Shell elements based on Classical shell theories

These elements are usually based on thin shell theories. The application of these elements are limited by their corresponding theories. The elements need data such as higher order derivatives of shell surface geometry in addition to the usual nodal coordinates and thickness of the shell.

3. Degenerated Solid elements

This type of element is derived from 3-D continuum theory by using the assumption that normals to the middle surface remain normal after deformation. The strain energy related to the stresses perpendicular to the middle surface is ignored. The usage of degenerated solid shell elements have become more popular than any other methods mainly because they are not based on any particular shell theory limiting their scope. Hence their application is more versatile. The solution obtained from degenerated solid shell elements are more accurate and closer to the solution obtained using the 3-D continuum approach. Moreover the degenerated solid shell elements are based on an isoparametric technique for mapping curved shell elements. Hence they are more accurate than facet elements. The data needed are only the nodal coordinates and the thickness of the shell.

A general formulation for the curved, arbitrary shape of thick shell finite element was first presented by S.Ahmad et al.[7]. Previous attempts to develop curved shell elements were limited to shallow shell situations in which only shear deformation was neglected. The authors used the isoparametric transformation concept for the development of curved shell elements. They used the well known assumption that even for thick shells, the normals to the midsurface remain practically straight after deformation. The accuracy of the element is excellent but here the convergence to the exact solution is constrained by the fact that plane section remain plane during

deformation. In this model, the pure bending deformation modes are accompanied by some shear stress which does not exist in conventional thin shell or plate theory. Hence large elements experiencing pure bending tend to be more stiff. This phenomenon is also called shear locking. A significant effort has been directed in the development of elements free from shear locking. This can be achieved by any of the following methods.

a. Reduced integration with spurious mode control

b. Assumed strain field which defines the transverse strain field consistently

c. Hybrid/mixed formulations with special stress modes

d. Moment redistribution mechanism approach.

Although the degenerated solid shell elements have many advantages, they have a few undesirable features. The major undesirable feature is the locking phenomenon. Another problem in the degenerated shell element is the discretization of complicated surfaces. The preparation of input data by hand is time consuming, error prone and impractical. These problems are overcome by using automatic mesh generation and adaptive analysis procedures [8].

Adaptive refinement is becoming a common feature in most commercial finite element codes. It should be noted that the triangular meshes are often the preferred choice for most automatic mesh generators [9]. They are simple in modelling irregular shapes along with other types of finite elements. These elements suffer from shear

locking effect and hence a lot of work is being carried out in the improvement of triangular elements.

Shear locking can be eliminated using independent approximations in including the bending and shear effects. Bathe and Dvorkin [10] developed a 4-node quadrilateral element using this technique. This element is based on Reissner-Mindlin theory and uses mixed interpolation. They used an independent set of shape functions for the covariant components of transverse shear strains. The element performed better in both thick and thin shell applications. This element is popularly known as MITC4 (Mixed interpolation tensorial components). They further developed a 8-node element using the same approach [11].

Saleeb and Chang [12] developed an efficient quadrilateral element for plate bending based on Hellinger/Reissner mixed variational approach, where displacements, bending stress and shear stress parameters are assumed independently. However they reported inherent problems in the selection of stress fields.

A shear locking free 3-node isoparametric element for thin and thick plates is developed by Kabir [13] using Reissner/Mindlin theory. He introduces a correction term to balance the mismatch between the transverse shear strain field and the derivative of displacement. In this way he countered the locking problem.

The locking phenomenon in plate and shell elements is mainly due to the lack of consistency in defining the transverse shear strain field. This inconsistency in shear field produces spurious constraints that lead to spurious energies which stiffen the element and also cause violent oscillations of stresses. G.Prathap and Somashekar [14] devised a method to define the strain field consistently. They also found out that the jacobian transformation to the global coordinate alters the defined strain field and hence introduced the concept of edge consistency. The tangential strains along the edges between neighboring elements are also matched to have a shear locking free element. This is done by defining pseudo shear strain in the local coordinate system and then transforming to the global coordinates using the jacobian transformation at the nodes.

2.1.2 P-version FEM

The concept of p-version finite element analysis is relatively new. A large number of papers have been published on this subject and its merits over h-version are well proved. One of the first works on p-version FEM is by Peano [15]. New hierarchies of C^0 and C^1 interpolations over triangles are presented. The main characteristics of this family of the finite element is that the shape functions corresponding to an interpolation of order p, constitute the subset of higher order interpolation functions greater than p. Hence the stiffness matrix of the element of order p, forms the subset of stiffness matrices of higher orders greater than p. This development gives rise to

new families of finite elements which are computationally efficient.

The elemental arrays of higher polynomial order can be efficiently computed using hierarchical elements with precomputed arrays. These precomputed arrays are computed once and stored in a permanent file which can be used in all subsequent applications of the program. Rossow and Kutz [16] showed that the use of hierarchical elements with precomputed arrays are competitive in terms of computational efficiency compared to conventional finite element method.

The advantages of the hierarchical approach are presented by Zienkiwicz, Gago and Kelley [17]. They show the hierarchical nature of the stiffness matrices. The condition of the stiffness matrix increases because of the appearance of hierarchical variables as a perturbation on the original solution. This ensures a faster rate of iteration convergence. The perturbation nature of the hierarchical form has a further merit of providing an immediate measure of the error in the solution, by analyzing the displacement solutions of successive orders.

A proper mesh design increases the performance of p-refinement to the best performance attainable by the finite element method. Szabo [18] gives the guidelines for prior mesh design for the P-version FEM. Babuska, Griebel and Pitkaranta [19] discuss the optimal selection of shape functions for p-type finite elements. They also discuss the efficacy of the conjugate gradient and multilevel iteration methods for solving the linear system.

The hierarchical linear equation sets can be efficiently solved by using a proper solution strategy. Morris, Tsuji and Cornevali [20] developed an algorithm which has the ability to choose dynamically between iterative and direct solvers. It can also adjust the preconditioning in iterative solvers dynamically. The combination of direct and iterative solvers gives an efficient solution path, combining the advantages of both the solvers.

Woo and Basu [21] presented a new hierarchic p-version cylindrical shell element for the analysis of singular cylindrical shells. They used the Legendre polynomial shape functions for the approximation of the displacement field. A blend mapping function is used for the transformation from the standard to the real domain. A blend mapping function exactly maps the curved boundaries using the exact geometric parameters. The Legendre polynomials are able to oscillate with increased frequency near the end points and thus are better suited to approximating singular behavior. The stiffness matrix based on this element is well conditioned even at higher p-levels and hence gives faster convergence. This p-version cylindrical shell element is very efficient in terms of accuracy and computational efficiency compared to h-version cylindrical elements.

Szabo and Sahrmann[22] presented a 4-node 2-D element and an 8-node 3-D solid element for the analysis of shells. The work done by Surana and Sorem [23] is of special interest here. They developed a hierarchical three dimensional curved shell

element based on the p-version concept. The geometry of the element is described by the coordinates of the nodes in its middle surface and nodal vectors describing its top and bottom surfaces. The element displacement function can be of any arbitrary and different polynomial order. The approximation functions and their corresponding hierarchical variables are obtained by first constructing the approximation function and nodal variable for each of the three directions $\xi, \eta\ \zeta$ and then taking the tensor product. Here both the displacement functions and nodal variables are hierarchical and hence so are the element matrices. The formulation is effective for both thin and thick plates. The usage of hierarchical variables in the thickness direction increases the number of degrees of freedom greatly which increases the computational burden.

2.1.3 Symbolic computations

The development of symbolic computational packages like MAPLE, MACSYMA, MATHEMATICA etc have helped to reduce much of the computational task involved in developing new methods for finite element analysis. These symbolic packages have the ability to do various operations such as integration and differentiation. This ability can be used for deriving some closed form expressions for stiffness matrix and load vectors which reduces the computational effort substantially.

The numerical integration for hybrid/mixed element requires more computational effort than the displacement based element. The nature of the hybrid/mixed formulations is such that the analytical integration of the stiffness matrix can be done.

Chang, Tan and Zheng [24] developed a symbolic manipulation procedure leading to the analytical integration of the stiffness matrix of a hybrid/mixed elements. The development of analytical integration of the stiffness matrix has made the hybrid/mixed formulation computationally competitive and superior to the displacement based element. Moreover the analytical integration gives more accurate results than numerical integration.

Rangarajan, Knight and Aminpour [25] applied the symbolic computational approach for an assumed stress hybrid shell element with drilling degrees of freedom. The drilling degrees of freedom are rotation about the Z axis. The inplane displacement field approximations include contributions from the drilling degrees of freedom. They showed that the use of symbolic computation is twice as fast as the numerical version of the same element.

Kornkoff and Fenves [26], using symbolic computations, retained the nodal coordinates and operated in a symbolic form throughout the computation. As the coordinates are not numerical values, they may represent any set of actual coordinates. The stiffness matrix produced is expressed as a template in terms of these variables. It can be readily evaluated for a set of actual nodal coordinates during execution.

The use of computerized algebraic manipulation in the development of element stiffness matrix is advantageous. It greatly reduces the manual tedium and increases

the reliability of resulting expressions. Noor and Anderson [27] used symbolic manipulation in the development of algebraic expressions for the stiffness coefficients of non-linear finite elements. They also used the symbolic program MACSYMA for the generation FORTRAN source code for numerical evaluation of stiffness coefficients and checking for the correctness of FORTRAN statements. The expressions obtained are quite concise. Intermediate variables and symmetry of element matrices are used for expression simplifications. The incorporation of symbolic computations increase the accuracy, reliability and computational efficiency.

2.1.4 Object Oriented Programming (OOP)

Finite element programs are always error prone. They face bottleneck problems like maintenance, verification, portability, re-usability and extension. Some techniques have been used to solve a few of these problems by using programming languages like FORTRAN, which is traditionally used in FE programming. Later programming languages like PASCAL and C became famous because of their new capabilities like modularization, declaration of data types, pointers and dynamic memory allocation. In recent years the development of the OOP concept has given a new approach by which many of the above mentioned bottleneck problems can be solved.

The fundamentals of object oriented programming are explained below.

Objects: An object is an entity composed of data and procedures. OOP encapsulates specific kind of data with some specific procedures to operate on the data. This

kind of arrangement imposes certain behaviors on an object. When an object receives a message, it interprets the message and carry out some specific operations with one of its procedures. The procedures operate on the data of the object, so the internal detail of how it functions are hidden from the program that uses the object. This also means that the internal operation of an object can be changed without altering the rest of the program. Since the data are hidden in the objects, which cannot be viewed by the rest of the program, the data is safe from accidental change.

Classes: A class is an abstract data type definition. Objects are also called instances of a class. Similar objects are members of a class. The class has a list of instance variables and attached procedures. When an object is created, storage is allocated to its data described in the class. The created object maintain a link to its class so that it can access the attached procedures. Messages directed to an object are transferred to the class where the operations are performed using objects data as input.

Methods: The methods are attached to a class and they operate on the data of an object of a particular class. When a particular message is given, a class searches and finds the corresponding method . After locating the method, the class executes the operation using the object's data as input.

Inheritance: Classes can be derived from another class called parent class. The derived classes inherit both data and procedures from its parent class. In addition

to the inherited data and procedures, the derived classes can have its own data and procedures not visible to the parent class. This helps to customize the object to suite an application. Common features can be declared in a general class from which specific classes are generated. When an object receives a message, it first searches its class for the corresponding procedure, if no procedure is found, then it continues its search in its ancestor's classes.

Dynamic Memory Allocation (DMA): This feature is very useful in conserving memory. It enables the creation and deletion of data at any point in the program. In FEA, large matrices are used, which reduces the available memory and reduces the speed of computation. With DMA the objects are created dynamically and are deleted whenever they are not required further.

Polymorphism: This is the ability of a single message to activate different methods when addressed to different objects. For example, the method "*" could be redefined in a class which can carry out matrix multiplication.

Encapsulation: This is the ability to hide data inside an object. A message is sent to the object to access its data, which will activate one of its methods to operate on its own data. This kind of hiding of data by the object is called encapsulation. For example, the stiffness matrix *klocal* of an *element* object can be accessed only by the procedures of the *element* class.

The application of these concepts in FE programming proves advantageous. A lot of research is being carried out in the successful implementation OOP concepts in FE programming. Ford, Poschi and Steimer [28] analyzed the problem associated with conventional FE programming and the potential solutions offered by object oriented programming. They have laid the theoretical foundation for the implementation of object oriented concept.

Besson and Foerch [29] discuss the large scale design of object oriented finite element code. A basic library is proposed to handle the mathematical operations, repetitive input file and string manipulations. The classes reduce the independence in the code project and facilitate expandability. A large scale project based on OOP concept takes less CPU time than procedural programming. This is mainly due to the faster vector access. This aspect is very important because most of the CPU time is spent in solving the system of equations.

The object oriented programs are concise and require less time. S.P.Scholz [30] developed a finite element program for a Timoshenko beam using the object oriented programming language C++. He used classes such as matrix and vector which can perform mathematic operations by symbolic notation. This is achieved by the operator overloading feature and polymorphic methods. The data encapsulation lead to easier verification and maintenance of the program. The class concept and and the inheritance improved the data management and modularization.

Jeremic and Sture [31] presented a novel programming tool, nDarray which is designed using OOP concepts and implemented in C++. It allows building new data types such as tensors of any order. The development of tensorial objects of any order is very useful in implementing complicated tensorial formulae associated with the numerical solution of various elastic and elastoplastic problems in solid mechanics.

The development of new FE formulations is a time consuming task. Eyhermendy and Zimmermann [32, 33] proposed an object oriented approach for a symbolic environment to derive matrix forms from a strong form of an initial boundary value problem.

2.2 Scope of the Study

A displacement based hierarchical p-version plate and shell element is developed incorporating symbolic computations. The element has five nodal degrees of freedom, three translations in the global cartesian directions and two rotations in the local axes. The geometry of the element is modelled by serendipity approximation functions using the nodal coordinates and the nodal vector perpendicular to the midplane. The displacement approximation functions are hierarchical in nature and derived from the Lagrangian family. The degrees of freedom (dof) other than the corner nodes are hierarchical in nature. The dof at the corner nodes are the displacements u, v, w in the global X, Y, Z axes and the rotations in local axes. The element matrices are evaluated by using both full integration $(p+1)$ and reduced integration(p) techniques.

The element matrices are evaluated using numerical integration. As the order of the approximating polynomial function increases, the number of gaussian points have to be increased to obtain the element matrices. This increases the computational effort required for element generation. It is found that the usage of symbolic computations reduces the computational effort significantly. A number of locations where the usage of symbolic computations offers computational advantages are identified and incorporated. A finite element program for the formulated shell element is developed in C++ using the OOP concept. The OOP allows development of FE codes with good maintenance, verification, reusability and extension features.

Standard example problems from references are chosen for gauging the performance of the element. A square isotropic plate is analyzed under different boundary and loading conditions. The plate is also analyzed by varying the thickness of the plate. Results are compared with the analytical solutions. A cylindrical barrel vault under self weight is analyzed. This is a test example for shells in which the bending action is severe. The performance of the shell element is also evaluated in the standard test problem of a hemispherical shell with a hole. A thin cylindrical shell which is loaded by two centrally located and diametrically opposed concentrated forces is analyzed. The results of these tests are compared with the reference values from the literature. The effectiveness of the element is demonstrated.

Chapter 3

FINITE ELEMENT FORMULATION

3.1 Shell Element

In this study, the displacement finite element approach is used. Figure 3.1 shows the generic curved shell element in the local ξ, η and ζ coordinate system. In this formulation, it is assumed that the normal to the middle surface remains practically straight after deformation. This assumption permits the shear deformation which is very important in the thick shell situation. The strain energy corresponding to the stresses perpendicular to the middle surface is ignored for simplification. The element has 9 nodes and each node has 5 degrees of freedom. The degrees of freedom consist of displacements in the X,Y,Z directions and rotations of nodal vector v_3 about two orthogonal directions normal to it.

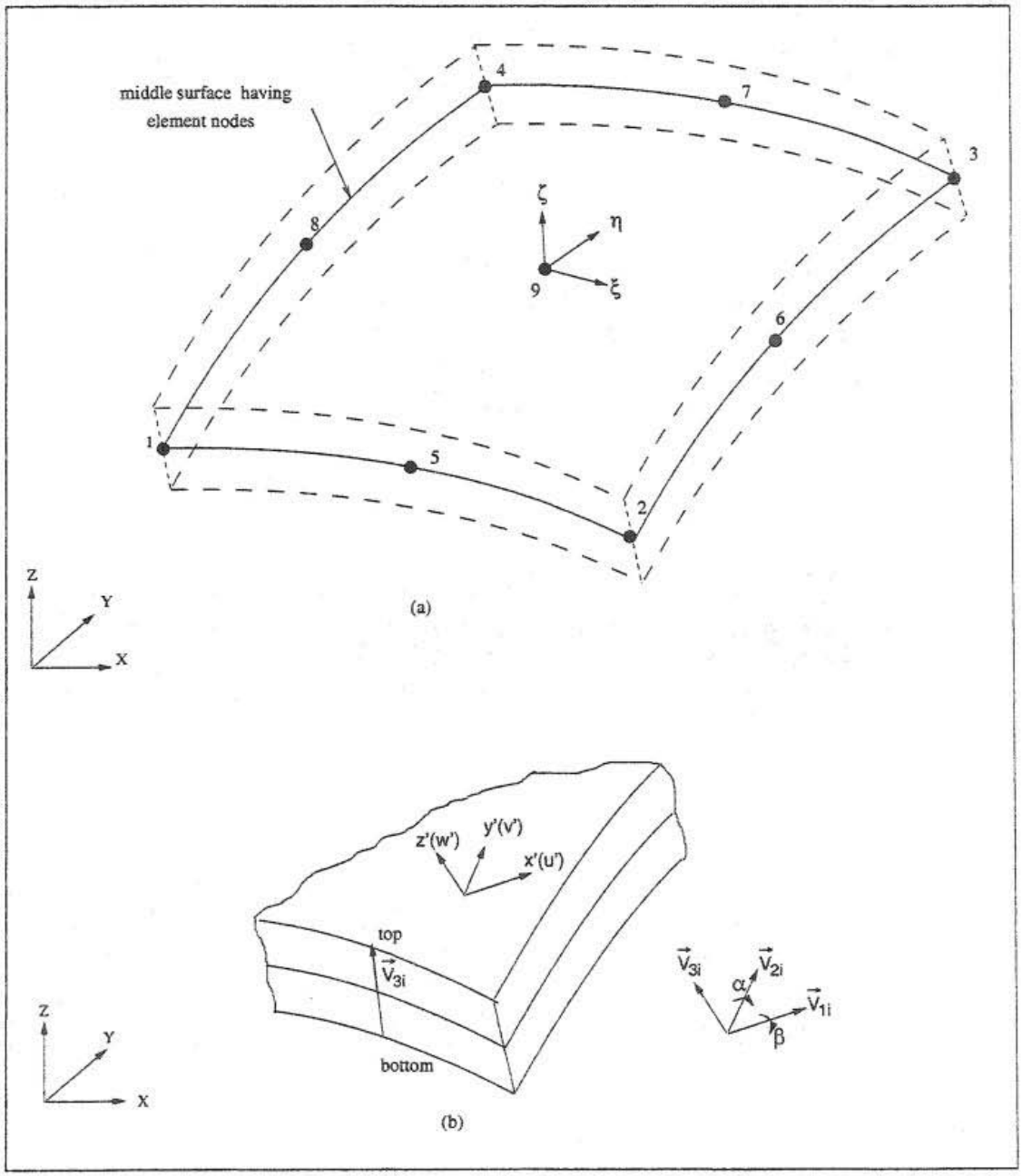

Figure 3.1: Generic nine node curved shell element

3.2 Geometric definition of the element

The nodes are located at the middle surface. The external surfaces of the element are curved and the sections across the thickness are straight. Hence, two curvilinear coordinates ξ, η in the middle plane and a linear coordinate ζ in the thickness direction are defined for the approximation of the geometry. These local coordinates ξ, η, ζ vary between -1 and 1. The top and bottom coordinates define the shape of the element. The relationship between the cartesian coordinates and the curvilinear coordinates for any point in the element is given by:

$$\begin{Bmatrix} x \\ y \\ z \end{Bmatrix} = \sum_{i=1}^{8} N_i(\xi,\eta) \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix}_{mid} + \sum_{i=1}^{8} N_i(\xi,\eta)\frac{\zeta}{2} v_{3i} \tag{3.1}$$

where,

$$v_{3i} = \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix}_{top} - \begin{Bmatrix} x_i \\ y_i \\ z_i \end{Bmatrix}_{bottom} \tag{3.2}$$

$N(\xi,\eta)$ is a serendipity approximation shape function. The subscripts *top, bottom* and *mid* indicates top, bottom and mid plane respectively. the subscript i refers to the node. The midplane coordinates are evaluated by averaging the top and bottom coordinates.

3.3 Displacement function

The displacement throughout the element is uniquely defined by three displacements in the global X, Y, Z directions and rotations of the nodal vector v_{3i} about two or-

thogonal directions perpendicular to it. The displacement function is given by:

$$\begin{Bmatrix} u \\ v \\ w \end{Bmatrix} = \sum_{i=1}^{n} N_i \begin{Bmatrix} u_i \\ v_i \\ w_i \end{Bmatrix} + \sum_{i=1}^{n} N_i \zeta \frac{t_i}{2} [v_{1i}, -v_{2i}] \begin{Bmatrix} \alpha_i \\ \beta_i \end{Bmatrix} \tag{3.3}$$

where n refers to the number of nodes in the element including the hierarchical nodes. It should be noted that the above displacement function also includes the hierarchical nodal variables. These hierarchical nodal variables and shape function are separately discussed in section 3.6.

The vectors v_{1i} and v_{2i} are uniquely defined as:

$$v_{1i} = \frac{j \times v_{3i}}{|j \times v_{3i}|}$$

$$v_{2i} = \frac{v_{3i} \times v_{1i}}{|v_{3i} \times v_{1i}|}$$

3.4 Stresses and Strains

The strains are calculated in a local coordinate system, where the z' axis is normal to the mid surface with the two other orthogonal axes x' and y' tangent to it. The strain components are given by:

$$\{\epsilon'\} = \begin{Bmatrix} \epsilon_{x'} \\ \epsilon_{y'} \\ \gamma_{x'y'} \\ \gamma_{x'z'} \\ \gamma_{y'z'} \end{Bmatrix} = \begin{Bmatrix} \frac{\partial u'}{\partial x'} \\ \frac{\partial v'}{\partial y'} \\ \frac{\partial u'}{\partial y'} + \frac{\partial v'}{\partial x'} \\ \frac{\partial w'}{\partial x'} + \frac{\partial u'}{\partial z'} \\ \frac{\partial w'}{\partial y'} + \frac{\partial v'}{\partial z'} \end{Bmatrix} \tag{3.4}$$

The normal strain component in the z' direction is neglected according to the shell assumptions. The stresses are given by the constitutive law.

$$\{\sigma'\} = \begin{Bmatrix} \sigma_{x'} \\ \sigma_{y'} \\ \tau_{x'y'} \\ \tau_{x'z'} \\ \tau_{y'z'} \end{Bmatrix} = [D'](\{\epsilon'\} - \{\epsilon_0\}) \tag{3.5}$$

where $\{\epsilon_0\}$ is the initial strain.

The constitutive matrix $[D']$ for an isotopic material is:

$$[D'] = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 & 0 & 0 \\ \nu & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{1-\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1-\nu}{2k} & 0 \\ 0 & 0 & 0 & 0 & \frac{1-\nu}{2k} \end{bmatrix} \tag{3.6}$$

The factor k is included to improve the shear displacement approximation and it is taken as 1.2 which is the ratio of relevant strain energies.

3.5 Element Matrices Evaluation

The derivatives of the displacement with respect to the global X, Y, Z coordinates are given by the relation,

$$\begin{bmatrix} u_{,x} & v_{,x} & w_{,x} \\ u_{,y} & v_{,y} & w_{,y} \\ u_{,z} & v_{,z} & w_{,z} \end{bmatrix} = [J]^{-1} \begin{bmatrix} u_{,\xi} & v_{,\xi} & w_{,\xi} \\ u_{,\eta} & v_{,\eta} & w_{,\eta} \\ u_{,\zeta} & v_{,\zeta} & w_{,\zeta} \end{bmatrix} \tag{3.7}$$

where $[J]$ is the Jacobian matrix given by:

$$[J] = \begin{bmatrix} x_{,\xi} & y_{,\xi} & z_{,\xi} \\ x_{,\eta} & y_{,\eta} & z_{,\eta} \\ x_{,\varsigma} & y_{,\varsigma} & z_{,\varsigma} \end{bmatrix} \tag{3.8}$$

Here a comma(,) followed by a subscript indicates the partial differentiation with respect to the subscript. The same notation is followed here onwards. These derivatives of the displacement are transformed to the local displacement directions x', y', z' for the evaluation of strains. The directions of the local axes are established by the following method.

A vector normal to the shell surface is found by the cross product of two vectors tangential to the surface and it is given by:

$$V_3 = \begin{Bmatrix} x_{,\xi} \\ y_{,\xi} \\ z_{,\xi} \end{Bmatrix} \times \begin{Bmatrix} x_{,\eta} \\ y_{,\eta} \\ z_{,\eta} \end{Bmatrix} \tag{3.9}$$

The other two directions are uniquely defined as:

$$V_1 = \frac{j \times V_3}{|j \times V_3|}$$

$$V_2 = \frac{V_3 \times V_1}{|V_3 \times V_1|}$$

The x', y', z' directions are obtained by reducing the above vectors to unit magnitude.

$$[\theta] = [\hat{V_1}, \hat{V_2}, \hat{V_3}] \tag{3.10}$$

The local derivatives of the displacement are given by:

$$\begin{bmatrix} u'_{,x'} & v'_{,x'} & w'_{,x'} \\ u'_{,y'} & v'_{,y'} & w'_{,y'} \\ u'_{,z'} & v'_{,z'} & w'_{,z'} \end{bmatrix} = [\theta]^T \begin{bmatrix} u_{,x} & v_{,x} & w_{,x} \\ u_{,y} & v_{,y} & w_{,y} \\ u_{,z} & v_{,z} & w_{,z} \end{bmatrix} [\theta] \tag{3.11}$$

Substitution of these components in equation 3.4 gives the strain components.

$$\{\epsilon'\} = [B']\{\delta\} = [B'] \begin{Bmatrix} \{\delta_1\} \\ \{\delta_2\} \\ \vdots \\ \{\delta_n\} \end{Bmatrix} \qquad \{\delta_i\} = \begin{Bmatrix} u_i \\ v_i \\ w_i \\ \alpha_i \\ \beta_i \end{Bmatrix} \tag{3.12}$$

Where the matrix $[B']$ is called the strain matrix. The displacement matrix $\{\delta\}$ is partitioned into sub-matrices containing the nodal variables corresponding to the particular node i. The value of n depends upon the number of nodes in the element including the hierarchical nodes.

The element stiffness matrix and load vector is evaluated by the following definitions.

$$[K^e] = \int_\Omega [B]^T [D] [B] \, d\Omega \tag{3.13}$$

$$\{F^e\} = \int_\Gamma [N]^T \{f_s\} d\Gamma + \int_\Omega [N]^T \{f_b\} d\Omega \tag{3.14}$$

The integration process is done in the local coordinate system. Changing the limits to local coordinates system gives,

$$[K^e] = \int_{-1}^{1} \int_{-1}^{1} \int_{-1}^{1} [B]^T [D] [B] \, |J(\xi, \eta, \zeta)| \, d\xi \, d\eta \, d\zeta \tag{3.15}$$

Gauss quadrature rule is used to numerically integrate the element stiffness matrix.

Gaussian integration using $NG \times NG \times NG$ points is given by:

$$[K^e] = \sum_{k=1}^{NG}\sum_{j=1}^{NG}\sum_{i=1}^{NG}[B(\xi_i,\eta_j,\zeta_k)]^T[D][B(\xi_i,\eta_j,\zeta_k)]\,|J(\xi_i,\eta_j,\zeta_k)|\,w_iw_jw_k \tag{3.16}$$

Where w_i, w_j and w_k are the gaussian weights corresponding to i, j and k^{th} gauss points. Two point integration is used along the ζ direction as ζ is a linear coordinate. The order of integration along the ξ and η directions depends upon the hierarchical order chosen along the respective directions.

Once the stiffness and load matrices for all the elements are evaluated, they are assembled to form the global stiffness matrix and load matrix.

$$[K^G] = \sum_e [K^e]$$

$$[F^G] = \sum_e [F^e]$$

The superscripts G and e indicates the global and elemental nature of the matrices. The assembled system of equations are given by:

$$[K^G]\{\delta\} = \{F^G\} \tag{3.17}$$

The above equation is solved to get the global vector of nodal displacements $\{\delta\}$.

The stresses evaluated by equation 3.5 are in the local coordinate system. Since the stresses in the local coordinate system are not easily visualized, it is conveniently transformed to the global system using the following expression.

$$\begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} = [\theta] \begin{bmatrix} \sigma_{x'} & \tau_{x'y'} & \tau_{x'z'} \\ \tau_{x'y'} & \sigma_{y'} & \tau_{y'z'} \\ \tau_{x'z'} & \tau_{y'z'} & 0 \end{bmatrix} [\theta]^T \tag{3.18}$$

3.6 P-Version Finite Element Formulation

This section presents the hierarchical shape functions used in the displacement approximations and their derivations [23]. Figure 3.2 shows the two dimensional Lagrange family element. Let p_ξ and p_η are the polynomial order of approximation along ξ and η directions respectively. The number of nodes depend upon the polynomial order. The approximation shape functions for this element is obtained by evaluating Lagrange interpolation function separately for ξ, η directions and then finding their tensor product. Since the number of nodes in the element depends upon the order, any increase in order along one side causes the change in nodal configuration and requires a new node to be created. A hierarchical formulation from a 2D lagrange element offsets this disadvantage.

Figure 3.3 shows the one dimensional Lagrange family parabolic , cubic etc. configuration and the respective nodal degrees of freedom in ξ and η directions . The approximation function for the one dimensional configuration in ξ can be written as

$$U_\xi(\xi) = \sum_{m=1}^{n_\xi} N_m(\xi) U_{\xi m} \tag{3.19}$$

Where $n_\xi = p_\xi + 1$ is the number of nodes in ξ direction. N_m is the one dimensional lagrange interpolating function given in appendix A.1.

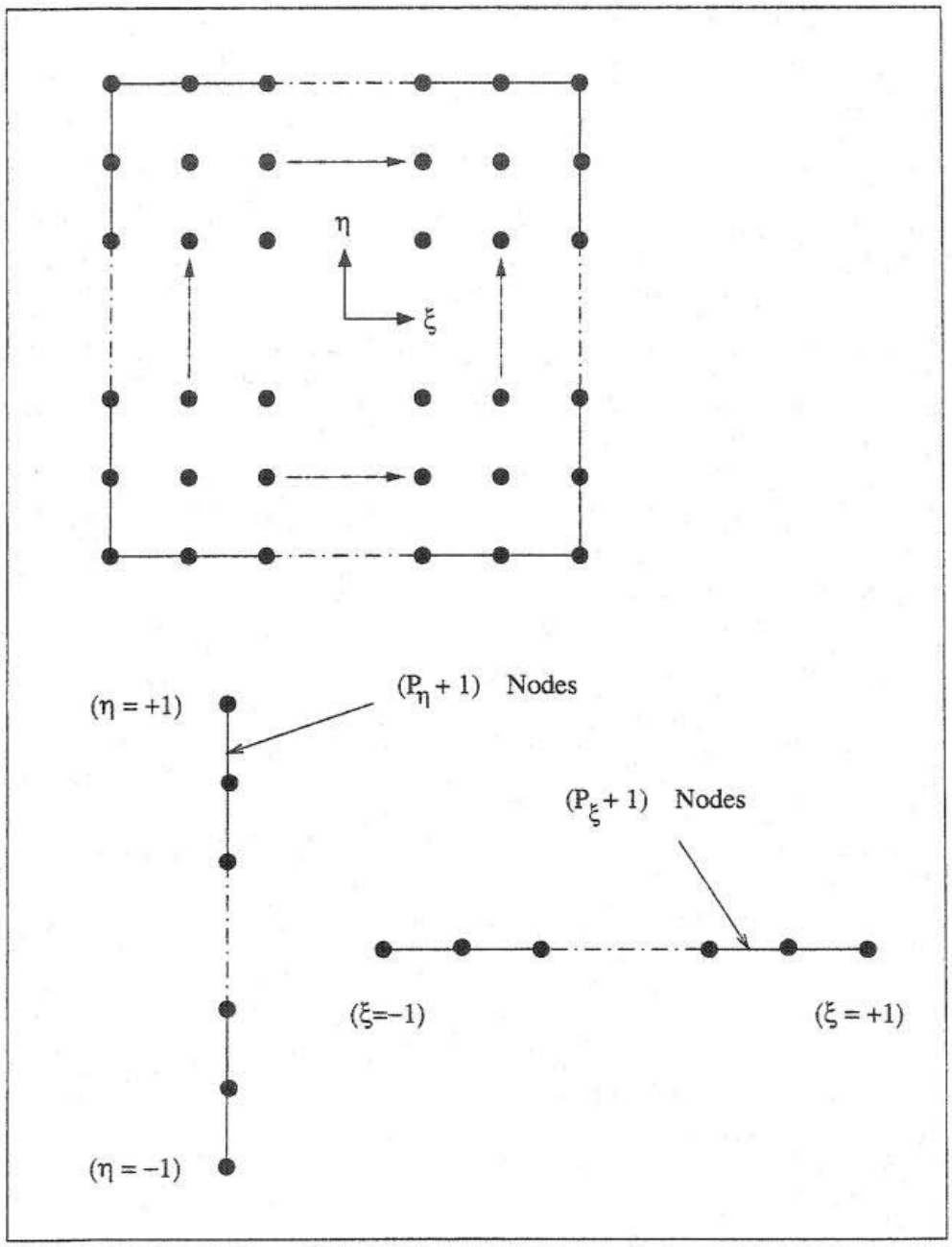

Figure 3.2: Two dimensional Lagrange element

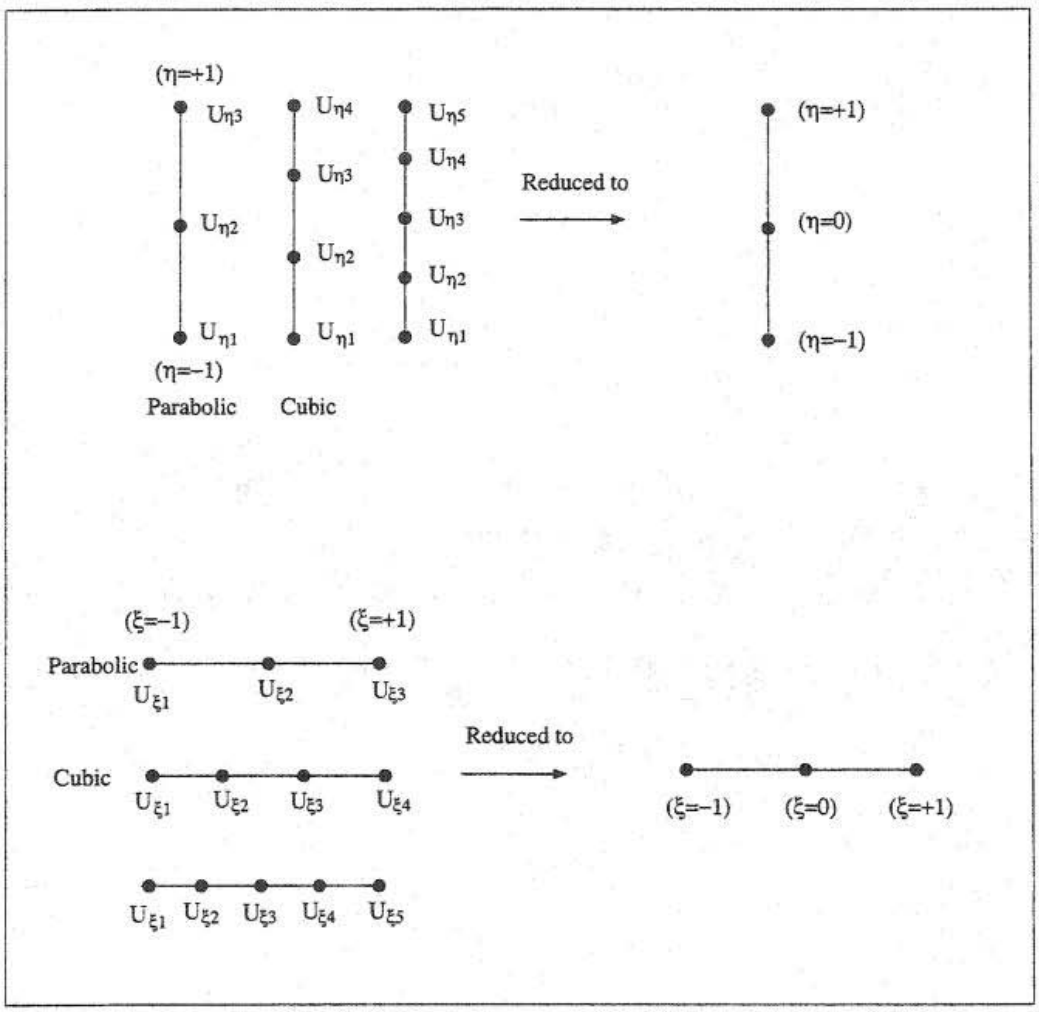

Figure 3.3: Higher order Lagrange nodal configuration and its equivalent three node hierarchical configuration

For a parabolic case, substitution of the shape functions and nodal variables in the above equation and rearranging gives:

$$U_\xi(\xi) = \frac{1}{2}(1-\xi)U_{\xi 1} + \frac{1}{2}(1+\xi)U_{\xi 3} + \left(\frac{\xi^2-1}{2}\right)(U_{\xi 1} - 2U_{\xi 2} + U_{\xi 3}) \tag{3.20}$$

Differentiating the above with respect to ξ, then evaluating these derivatives at $\xi = 0$ and substituting back in equation 3.19 gives:

$$U_\xi(\xi) = \frac{1}{2}(1-\xi)U_{\xi 1} + \frac{1}{2}(1+\xi)U_{\xi 3} + \left(\frac{\xi^2-1}{2}\right)\left(\frac{\partial^2 U_\xi}{\partial \xi^2}\right)_{\xi=0} \tag{3.21}$$

The same procedure for Lagragian cubic configuration yields:

$$U_\xi(\xi) = \frac{1}{2}(1-\xi)U_{\xi 1} + \frac{1}{2}(1+\xi)U_{\xi 4} + \left(\frac{\xi^2-1}{2}\right)\left(\frac{\partial^2 U_\xi}{\partial \xi^2}\right)_{\xi=0} + \left(\frac{\xi^3-\xi}{6}\right)\left(\frac{\partial^3 U_\xi}{\partial \xi^3}\right)_{\xi=0} \tag{3.22}$$

From above, it is seen that equation 3.21 is subset of equation 3.22. In general, for a Lagrange configuration with $p_\xi + 1$ equally spaced nodes, the equivalent three node configuration in the ξ direction is given by:

$$U_\xi(\xi) = \frac{1}{2}(1-\xi)U_{\xi 1} + \frac{1}{2}(1+\xi)U_{\xi(p_\xi+1)} + \sum_{m=2}^{p_\xi}\left(\frac{\xi^m - a}{m!}\right)\left(\frac{\partial^m U_\xi}{\partial \xi^m}\right)_{\xi=0} \tag{3.23}$$

The above equation can be concisely written as:

$$U_\xi(\xi) = N^1_{\xi 1}U_{\xi 1} + N^3_{\xi 1}U_{\xi(p_\xi+1)} + \sum_{m=2}^{p_\xi} N^2_{\xi m}U_{\xi,\xi^m} \tag{3.24}$$

where,

$$U_{\xi,\xi^m} = \left.\frac{\partial^m U_\xi}{\partial \xi^m}\right|_{\xi=0}$$

and

$$a = \begin{cases} 1 & \text{if } m \text{ is even} \\ \xi & \text{if } m \text{ is odd} \end{cases}$$

Similarly the equivalent 3 node configuration in the η direction is:

$$U_\eta(\eta) = \frac{1}{2}(1-\eta)U_{\eta 1} + \frac{1}{2}(1+\eta)U_{\eta(p_\eta+1)} + \sum_{m=2}^{p_\eta}\left(\frac{\eta^m - b}{m!}\right)\left(\frac{\partial^m U_\eta}{\partial \eta^m}\right)_{\eta=0} \tag{3.25}$$

or

$$U_\eta(\eta) = N^1_{\eta 1}U_{\eta 1} + N^3_{\eta 1}U_{\eta(p_\eta+1)} + \sum_{m=2}^{p_\eta} N^2_{\eta m}U_{\eta,\eta^m} \tag{3.26}$$

where,

$$U_{\eta,\eta^m} = \left.\frac{\partial^m U_\eta}{\partial \eta^m}\right|_{\eta=0}$$

and

$$b = \begin{cases} 1 & \text{if } m \text{ is even} \\ \eta & \text{if } m \text{ is odd} \end{cases}$$

Hierarchical variables along the ζ direction are not considered as we use a linear coordinate in ζ direction.

From the equations 3.24,3.26 , the approximation functions for the 9 node curved shell element can be evaluated by tensor product of the hierarchical one dimensional approximation functions in the ξ and η directions. Similarly the nodal variables are evaluated by the tensor product of the hierarchical one dimensional variables in ξ and η directions. The approximation functions and the corresponding nodal variables are concisely given below. The field variable U is replaced by Φ. The node number is indicated by m.

Corner nodes(nodes 1,2,3 and 4):

Approximation functions	Hierarchical variables	
$N^m_{\xi\eta}$	$(\Phi)_m$	(3.27)

where $N^m_{\xi\eta}$ are

$N^1_{\xi\eta} = N^1_{\xi 1}N^1_{\eta 1}$; $N^2_{\xi\eta} = N^3_{\xi 1}N^1_{\eta 1}$; $N^3_{\xi\eta} = N^3_{\xi 1}N^3_{\eta 1}$; and $N^4_{\xi\eta} = N^1_{\xi 1}N^3_{\eta 1}$; for nodes 1,2,3 and 4 respectively.

Midside nodes(nodes 5 and 7):

Approximation functions	Hierarchical variables	
N^m_i	$\left(\frac{\partial^i \Phi}{\partial \xi^i}\right)_m$	(3.28)

where $i = 2, 3, \ldots, p_\xi$

For node 5 the approximation function is $N^5_i = N^1_{\xi 1}N^2_{\eta i}$; and for node 7, $N^7_i = N^3_{\xi 1}N^2_{\eta i}$;

Midside nodes(nodes 6 and 8):

Approximation functions	Hierarchical variables	
N^m_j	$\left(\frac{\partial^j \Phi}{\partial \eta^j}\right)_m$	(3.29)

where $j = 2, 3, \ldots, p_\eta$

For node 6 the approximation function is $N^6_j = N^3_{\xi 1}N^2_{\eta j}$; and for node 8, $N^8_j = N^1_{\xi 1}N^2_{\eta j}$;

Center node(node 9):

Approximation functions	Hierarchical variables	
N_{ij}^{m}	$\left(\frac{\partial^j}{\partial \eta^j}\left(\frac{\partial^i \Phi}{\partial \xi^i}\right)\right)_m$	(3.30)

where $i = 2, 3, \ldots, p_\xi$

where $j = 2, 3, \ldots, p_\eta$

The shape function for the node 9 is $N_{ij}^{m} = N_{\xi i}^{2} N_{\eta j}^{2}$;

The displacement u, v, w at any point in the element is given by:

$$\left\{ \begin{array}{l} u(\xi, \eta, \zeta) \\ v(\xi, \eta, \zeta) \\ w(\xi, \eta, \zeta) \end{array} \right\} = [N]\{\delta\} \tag{3.31}$$

The hierarchical shape functions $[N]$ and the nodal variables $\{\delta\}$ for different p-levels are easily obtained from the equations 3.27 to 3.30. The equation 3.31 is the basis for the evaluation of stiffness matrix and load vectors.

Chapter 4

SYMBOLIC COMPUTATION

4.1 Incorporation of Symbolic Computations

The use of symbolic computations improve the computational efficiency of a FE analysis by decreasing the number of operations required for the evaluation of a particular matrix or parameter. A number of computations where the involvement of symbolic computations proves advantageous are located. The following sections explain the involvement of symbolic computations in the evaluation of different matrices and parameters.

4.2 Evaluation of Jacobian

The components of the jacobian matrix in equation 3.8 are,

$$J_{11} = \frac{\partial x}{\partial \xi} \qquad J_{12} = \frac{\partial y}{\partial \xi} \qquad J_{13} = \frac{\partial z}{\partial \xi}$$

$$J_{21} = \frac{\partial x}{\partial \eta} \qquad J_{22} = \frac{\partial y}{\partial \eta} \qquad J_{23} = \frac{\partial z}{\partial \eta}$$

$$J_{31} = \frac{\partial x}{\partial \zeta} \qquad J_{32} = \frac{\partial y}{\partial \zeta} \qquad J_{33} = \frac{\partial z}{\partial \zeta}$$

The substitution of equation 3.1 in the above equation gives the components of the jacobian matrix. These substitution and further simplification was carried out using MAPLE. The jacobian components are symbolically given below

$$
\begin{aligned}
J_{1i} &= a_{i1} + a_{i2}\xi + a_{i3}\eta + a_{i4}\zeta + a_{i5}\eta^2 + a_{i6}\xi\eta\zeta + a_{i7}\xi\zeta + a_{i8}\eta^2\zeta + a_{i9}\eta\zeta + a_{i10}\xi\eta \\
J_{2i} &= b_{i1} + b_{i2}\xi + b_{i3}\eta + b_{i4}\zeta + b_{i5}\xi^2 + b_{i6}\xi\eta\zeta + b_{i7}\xi\zeta + b_{i8}\eta\zeta + b_{i9}\xi\eta + b_{i10}\xi^2\zeta \\
J_{3i} &= c_{i1} + c_{i2}\xi + c_{i3}\eta + c_{i4}\xi^2 + c_{i5}\eta^2 + c_{i6}\xi\eta + c_{i7}\xi^2\eta + c_{i8}\eta^2\xi
\end{aligned}
\tag{4.1}
$$

where a_{im}, b_{in} and c_{io} are constants for an element whose value depends upon the geometrical coordinates of the nodes. The jacobian components evaluated using MAPLE are given in appendix C. The range of the subscripts i, m, n, o are,

$i = 1, 2, 3 \qquad m = 1, 2, \ldots, 10 \qquad n = 1, 2, \ldots, 10$

and $o = 1, 2, \ldots, 8$

The number of constants in equation 4.1 for the evaluation of the jacobian matrix is 84. The examination of the constants reveals that many constants are equal to each other. It is found that only 45 independent constants exist. It should also be noted that there exists a similarity within these constants. The constant $a_{1,m}, a_{2,m}$ and $a_{3,m}$ are similar in their expressions for same value of m. In the same way the constants $b_{i,n}$ and $c_{i,o}$ are also similar within them. This similarity helps in easier evaluation of the constants.

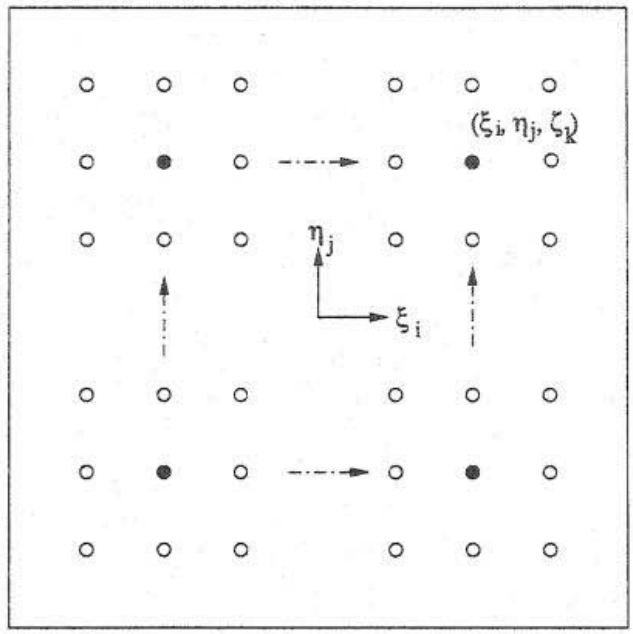

Figure 4.1: Symmetrical location of gaussian points

Figure 4.1 shows the symmetrical location of the gaussian points for a particular plane ζ_k. Let (ξ_i, η_j, ζ_k) be a point where jacobian has to be evaluated. Since the gaussian points are symmetric, it is advantageous to evaluate at all the symmetric points at once. The symmetric points for a particular value of ξ, η and ζ are:

$$(\xi_i, \eta_j, \zeta_k), (-\xi_i, \eta_j, \zeta_k), \ (\xi_i, -\eta_j, \zeta_k), (-\xi_i, -\eta_j, \zeta_k),$$

$$(\xi_i, \eta_j, -\zeta_k), (-\xi_i, \eta_j, -\zeta_k), \ (\xi_i, -\eta_j, -\zeta_k) \text{ and } (-\xi_i, -\eta_j, -\zeta_k)$$

The jacobian components are functions of ξ, η and ζ as evident from equation 4.1 and it can be written as,

$$J_{(\xi,\eta,\zeta)} = f(\xi, \eta, \zeta) \tag{4.2}$$

On evaluation of the function 4.2 at points ζ_k and $-\zeta_k$ gives the following pair of equations for the jacobian, which are now functions of ξ and η.

$$\begin{aligned} J_{(\xi,\eta,\zeta_k)} &= \Psi^1(\xi,\eta) \\ J_{(\xi,\eta,-\zeta_k)} &= \Psi^2(\xi,\eta) \end{aligned} \tag{4.3}$$

The above pair of equation are evaluated at the points η_j and $-\eta_j$, which gives the following set of equations which are now functions of ξ only.

$$\begin{aligned} J_{(\xi,\eta_j,\zeta_k)} &= \Upsilon^1(\xi) \\ J_{(\xi,-\eta_j,\zeta_k)} &= \Upsilon^2(\xi) \\ J_{(\xi,\eta_j,-\zeta_k)} &= \Upsilon^3(\xi) \\ J_{(\xi,-\eta_j,-\zeta_k)} &= \Upsilon^4(\xi) \end{aligned} \tag{4.4}$$

Further evaluation of the functions given in equation 4.4 at the points ξ_i and $-\xi_i$ gives the jacobian at all the symmetrical points.

$$\begin{aligned} J_{(\xi_i,\eta_j,\zeta_k)} &= \Phi^1 \\ J_{(-\xi_i,\eta_j,\zeta_k)} &= \Phi^2 \\ J_{(\xi_i,-\eta_j,\zeta_k)} &= \Phi^3 \\ J_{(-\xi_i,-\eta_j,\zeta_k)} &= \Phi^4 \\ J_{(\xi_i,\eta_j,-\zeta_k)} &= \Phi^5 \\ J_{(-\xi_i,\eta_j,-\zeta_k)} &= \Phi^6 \\ J_{(\xi_i,-\eta_j,-\zeta_k)} &= \Phi^7 \\ J_{(-\xi_i,-\eta_j,-\zeta_k)} &= \Phi^8 \end{aligned} \tag{4.5}$$

Where, $i,j = 1,\ldots,NG_p$ and k=1

NG_p= Number of positive gaussian points.

The functions Φ^m are just summation of constants. Many constants in the function Φ^m are equal for different m. It should be noted that the equation 4.5 can be used to calculate another set of 8 symmetrical points by just changing the value ξ_i to ξ_{i+1}. i.e, symmetry$(\xi_{i+1},\eta_j,\zeta_k)$. Thus jacobian can be evaluated at a total of $8 \times NG_p$ points.

4.3 Shape function derivatives

The derivatives of shape functions are required for the evaluation of the strain matrix $[B]$. The derivatives of shape functions with respect to the local coordinates ξ, η and ζ are independent of the geometry of the element (global coordinates x, y, z). These derivatives are constants for all the elements in the domain. Hence, it is advantageous in terms of computational effort to calculate these derivatives at all possible gaussian points at once. The evaluation can be done similar to the way in which the jacobian is evaluated. The implementation is explained in chapter 5.

4.4 Inverse of Jacobian

The evaluation of derivatives with respect to global coordinates require the evaluation of the inverse of the jacobian as seen in equation 3.7. The inversion of the matrix $[J]$ gives,

$$[J]^{-1} = \frac{[L]}{\det[J]} \tag{4.6}$$

where,

$$[L] = \begin{bmatrix} J_{22}J_{33} - J_{23}J_{32} & J_{13}J_{32} - J_{12}J_{33} & J_{12}J_{23} - J_{13}J_{22} \\ J_{23}J_{31} - J_{21}J_{33} & J_{11}J_{33} - J_{13}J_{31} & J_{13}J_{21} - J_{11}J_{23} \\ J_{21}J_{32} - J_{22}J_{31} & J_{12}J_{31} - J_{11}J_{32} & J_{11}J_{22} - J_{12}J_{21} \end{bmatrix} \tag{4.7}$$

$$\det[J] = J_{11}(J_{22}J_{33} - J_{23}J_{32}) - J_{12}(J_{33}J_{21} - J_{31}J_{23}) + J_{13}(J_{21}J_{32} - J_{31}J_{22}) \tag{4.8}$$

4.5 Shape function derivatives with respect to global coordinates

The derivatives are calculated using the following relation.

$$\begin{Bmatrix} N_{,x} \\ N_{,y} \\ N_{,z} \end{Bmatrix} = [J]^{-1} \begin{Bmatrix} N_{,\xi} \\ N_{,\eta} \\ N_{,\zeta} \end{Bmatrix} \tag{4.9}$$

Substituting equation 4.6 in equation 4.9 gives,

$$\begin{Bmatrix} N_{,x} \\ N_{,y} \\ N_{,z} \end{Bmatrix} = \frac{[L]}{\det[J]} \begin{Bmatrix} N_{,\xi} \\ N_{,\eta} \\ N_{,\zeta} \end{Bmatrix} \tag{4.10}$$

The above equation can be symbolically written as:

$$N_{,i} = \frac{1}{\det[J]} \sum_{j=1}^{3} L_{ij} N_{,j} \tag{4.11}$$

Where $i = 1, 2, 3$ and $j = 1, 2, 3$

4.6 Evaluation of the vectors of local cartesian axes

The vectors normal to the surface is given by equation 3.9. The simplification of the expression for the vectors yield,

$$V_1 = \begin{bmatrix} V_{33} \\ 0 \\ V_{31} \end{bmatrix} \frac{1}{(V_{33}^2 + V_{31}^2)^{\frac{1}{2}}} \tag{4.12}$$

$$V_1 = \begin{bmatrix} A_1 \\ A_2 \\ A_3 \end{bmatrix} \frac{1}{(A_1^2 + A_2^2 + A_3^2)^{\frac{1}{2}}} \tag{4.13}$$

where, $A_1 = V_{32}\, V_{13}$; $A_2 = V_{11}\, V_{33} - V_{31}\, V_{13}$; $A_3 = -V_{11}\, V_{32}$;

The direction cosines of the local axis is given by,

$$[\theta] = [V_1, V_2, V_3] \tag{4.14}$$

4.7 Evaluation of the Strain matrix $[B]$

The evaluation of matrix $[B]$ requires the transformation of the global derivatives of the displacement u, v and w to the local derivatives of the local orthogonal displacements.

The global derivatives of the displacements are calculated by the differentiation of equation 3.3.

$$\left\{ \begin{array}{c} u \\ v \\ w \end{array} \right\}_{,j} = \sum N_{i,j} \left\{ \begin{array}{c} u_i \\ v_i \\ w_i \end{array} \right\} + Na_{i,j} t_i [v_{1i}, -v_{2i}] \left\{ \begin{array}{c} \alpha_i \\ \beta_i \end{array} \right\} \tag{4.15}$$

$$Na = \frac{N\zeta}{2}$$

Where $j = 1, 2, 3$ and $,j$ indicates differentiation with respect to x, y, z coordinates. The substitution of the above equation in equation 3.11 gives the local derivatives of the orthogonal displacements and further substitution of these derivatives in the strain equation 3.4 gives the matrix $[B]$. The components of matrix $[B]$ are symbolically found out using MAPLE.

$$\begin{array}{l} B_{1k} = C_1 V_{1j} \\ B_{2k} = C_2 V_{2j} \\ B_{3k} = C_2 V_{1j} + C_1 V_{2j} \\ B_{4k} = C_1 V_{3j} + C_3 V_{1j} \\ B_{5k} = C_2 V_{3j} + C_3 V_{2j} \end{array} \tag{4.16}$$

$$C_1 = (V_{11} N_{i,x} + V_{12} N_{i,y} + V_{13} N_{i,z})$$

$$C_2 = (V_{21} N_{i,x} + V_{22} N_{i,y} + V_{23} N_{i,z})$$

$$C_1 = (V_{31} N_{i,x} + V_{32} N_{i,y} + V_{33} N_{i,z})$$

$$k = 5(i-1) + j$$

where $i = 1, 2, \ldots, n$, n- Number of degrees of freedom.

$j = 1, 2, 3$

Equation 4.16 evaluates only the components of matrix $[B]$ corresponding to the displacements u, v and w. The components corresponding to the rotation are evaluated using the following set of equations.

$$\begin{aligned} B_{1k} &= A_1 b_1 (-1)^j \\ B_{2k} &= A_2 b_2 (-1)^j \\ B_{3k} &= (A_2 b_1 + A_1 b_2)(-1)^j \\ B_{4k} &= (A_1 b_3 + A_3 b_1)(-1)^j \\ B_{5k} &= (A_2 b_3 + A_3 b_2)(-1)^j \end{aligned} \tag{4.17}$$

where,

$$A_1 = (V_{11} Na_{i,x} + V_{12} Na_{i,y} + V_{13} Na_{i,z})$$

$$A_2 = (V_{21} Na_{i,x} + V_{22} Na_{i,y} + V_{23} Na_{i,z})$$

$$A_3 = (V_{31} Na_{i,x} + V_{32} Na_{i,y} + V_{33} Na_{i,z})$$

$$b_1 = (V_{11} v_{j1}|_i + V_{12} v_{j2}|_i + V_{13} v_{j3}|_i)$$

$$b_2 = (V_{21} v_{j1}|_i + V_{22} v_{j2}|_i + V_{23} v_{j3}|_i)$$

$$b_3 = (V_{31} v_{j1}|_i + V_{32} v_{j2}|_i + V_{33} v_{j3}|_i)$$

$$k = (i-1)5 + 3 + j$$

$i = 1, 2, \ldots, n$ $\qquad$ $j = 1, 2$

n- Number of degrees of freedom.

4.8 Stiffness matrix computation

Equation 3.16 gives the expression for the evaluation of the stiffness matrix using gaussian integration method. Two point integration is chosen along the thickness direction and p or $p+1$ integration is chosen along the ξ and η directions. The symbolic multiplication of the matrices $[B]^T[D][B]$ using MAPLE gives the following expression.

$$[B]^T[D][B]_{ij} = A_1B_{1j} + A_2B_{2j} + B_{3i}D_{33}B(3,j) + B_{4i}D_{44}B_{4j} + B_{5i}D_{55}B_{5j} \tag{4.18}$$

where,

$$A_1 = B_{1i}\, D_{11} + B_{2i}\, D_{12}$$

$$A_2 = B_{1i}\, D_{12} + B_{2i}\, D_{22}$$

Where i, j varies up to the number of degrees of freedom.

The substitution of equation 4.18 in equation 3.16 gives the expression for the evaluation of the stiffness matrix using gaussian integration method.

4.9 Evaluation of Body force

The expression for the evaluation of the load vector due to body force is given by,

$$\{F_b\}^e = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1}[N]^T\{f_b\}\det[J]\partial\xi\,\partial\eta\,\partial\zeta \tag{4.19}$$

The shape function matrix is partitioned into submatrices corresponding to each node given by

$$[N] = [\ldots [N_i] \ldots]$$

$$[N_i] = \begin{bmatrix} N_i & 0 & 0 & Na_i t_i v_{11}|_i & -Na_i t_i v_{21}|_i \\ 0 & N_i & 0 & Na_i t_i v_{12}|_i & -Na_i t_i v_{22}|_i \\ 0 & 0 & N_i & Na_i t_i v_{13}|_i & -Na_i t_i v_{23}|_i \end{bmatrix} \tag{4.20}$$

$$\{f_b\} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}$$

Where f_1, f_2 and f_3 are the body forces acting along x, y, z directions respectively. Here $v_{11}|_i$ refers to the value of v_{11} calculated at node i.

The integration in equation 4.19 is done by the gaussian integration method.

$$\{F_b\}^e = \sum_{k=1}^{NG} \sum_{j=1}^{NG} \sum_{i=1}^{NG} [N(\xi_i, \eta_j, \zeta_k)] \{f_b\} \det[J(\xi_i, \eta_j, \zeta_k) w_i w_j w_k \tag{4.21}$$

where NG is the number of gaussian points used for integration along a a particular direction ξ, η or ζ. w_i, w_j and w_k are the gaussian weights. Symbolic evaluation of the expression $[F] = [N]^T \{f_b\}$ gives

$$F_k = f_j \, N_i \tag{4.22}$$

where,

$$i = 1, 2, \ldots, n \quad j = 1, 2, 3 \quad k = (i-1)5 + j$$

$$F_k = (f_1 v_{j1}|_i + f_2 v_{j2}|_i + f_3 v_{j3}|_i) N a_i \tag{4.23}$$

where,

$$i = 1, 2, \ldots, n \quad j = 1, 2 \quad k = (i-1)5 + 3 + j$$

and n is the total number of degrees of freedom.

The substitution of equations 4.22 and 4.23 in equation 4.21 give the expression for the evaluation of the load vector due to the body force.

4.10 Stress Strain calculation

The strain components in an element is given by equation 3.12. The stresses are calculated from the constitutive equation,

$$[\sigma'] = [D']\{\epsilon'\} \tag{4.24}$$

The evaluation of the above expression gives,

$$\begin{aligned}
\sigma_{x'} &= D'_{11}\epsilon_{x'} + D'_{12}\epsilon_{y'} \\
\sigma_{y'} &= D'_{12}\epsilon_{x'} + D'_{22}\epsilon_{y'} \\
\sigma_{x'y'} &= D'_{33}\gamma_{x'y'} \\
\sigma_{x'z'} &= D'_{44}\gamma_{x'z'} \\
\sigma_{y'z'} &= D'_{55}\gamma_{y'z'}
\end{aligned} \tag{4.25}$$

The stresses calculated above are in the local coordinate system x', y', z'. These stresses are transformed to the global system x, y, z using the equation 3.18 . This transformation was evaluated symbolically and is given by,

$$\begin{aligned}
\sigma_{ij} \;=\; & (V_{1i}\sigma_{x'} + V_{2i}\sigma_{x'y'} + V_{3i}\sigma_{x'z'})V_{1j} + (V_{1i}\sigma_{x'y'} + V_{2i}\sigma_{y'} + \\
& V_{3i}\sigma_{y'z'})V_{2j} + (V_{1i}\sigma_{x'z'} + V_{2i}\sigma_{y'z'})V_{3j}
\end{aligned} \tag{4.26}$$

4.11 Comments on Computational Effort

The incorporation of symbolic computations in the evaluation of element matrices reduces the computational effort significantly in different stages of the FE program.

1. Simplified expressions for the jacobian components are obtained with the minimum number of constants. Using these simplified expressions, the jacobian is evaluated at all the symmetrical gaussian points, which substantially reduces the computational effort.

2. Similar to the jacobian, simplified expressions are obtained for the strain matrix, stiffness matrix, vectors, body force components and shape function derivatives, which reduces the amount of computation involved.

3. All the symmetrical gaussian points have the same weight. This property is used to evaluate and sum the stiffness matrices of symmetrical points prior to multiplying by the weight.

4. Significant reduction in computational effort is achieved by computing the local shape function derivatives at all possible gaussian points and storing them in permanent arrays.

5. Intermediate variables are set up during the evaluation of many element properties, which reduces redundant computations.

6. The material property matrix $[D]$ has many zeros. The matrix multiplication in the evaluation stiffness matrix(equation 3.16) will involve unnecessary multiplication of zeros which, is avoided using a simplified stiffness expression(equation 4.18)

Chapter 5

COMPUTER IMPLEMENTATION

5.1 Finite Element Program

A p-version program MUNSET is developed for the analysis of plates and shells. The program is written in C++ incorporating object oriented programming features. The program is developed;

1. to demonstrate the successful incorporation of symbolic computations,
2. to verify the accuracy of the hierarchic shell element developed and
3. to prove the advantages of using object oriented programming along with symbolic computations.

The capabilities of the program include automatic mesh division and automatic order increment for selected elements. The major segments of the program are: element matrices evaluation, global assembly, solver and adaptive mesh division. Figure 5.1 shows the flow chart of the MUNSET program.

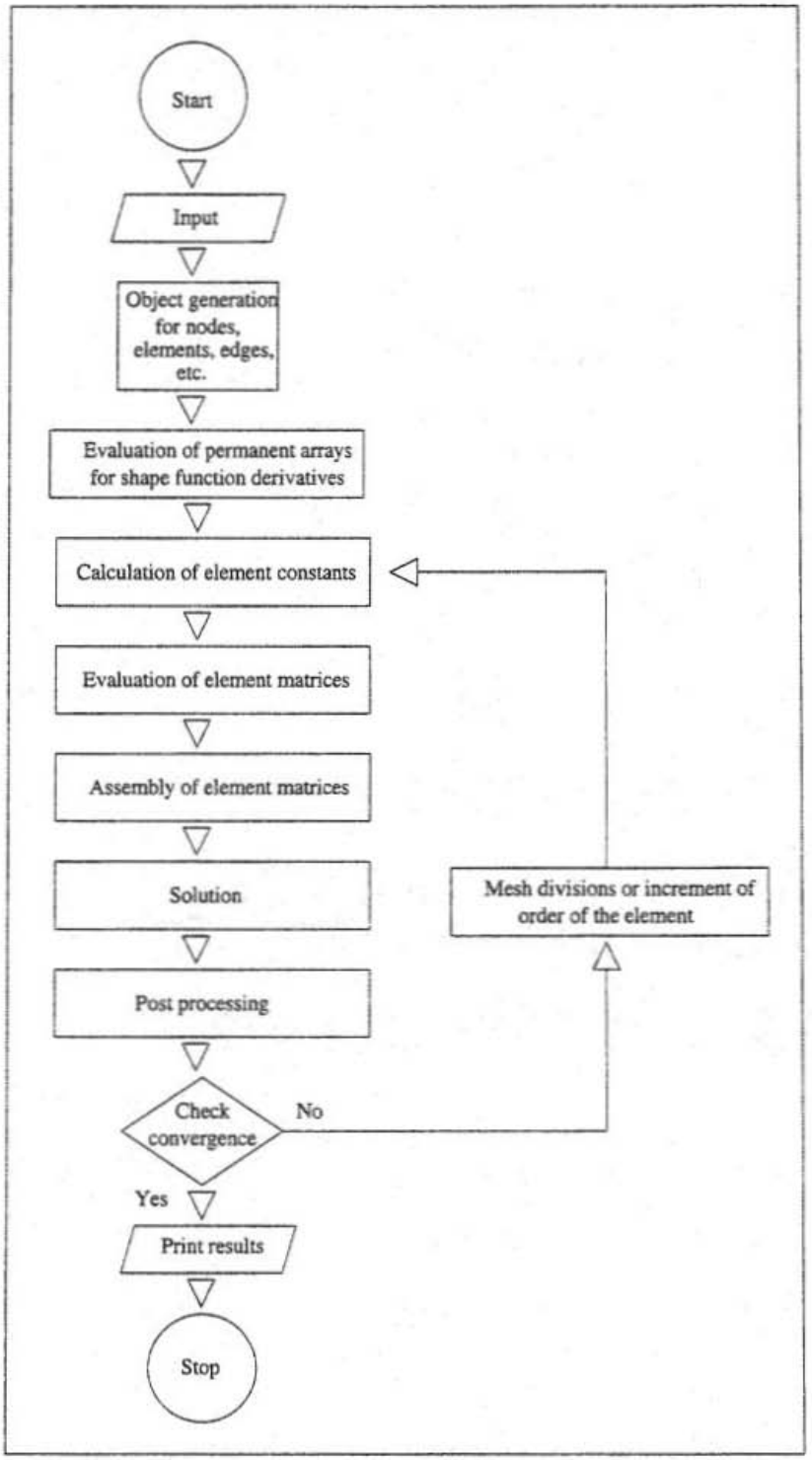

Figure 5.1: Flow chart of MUNSET program

The following subsections explain the various aspects of the program.

5.1.1 Input

The input to the program consists of material properties, geometrical coordinates of the nodes, nodal and edge connectivity of the elements, order of the elements, boundary constraints, choice of integration(p or $p+1$) and the loads applied.

5.1.2 Evaluation of element matrices

The evaluation of element matrices includes the evaluation of Jacobian, stiffness matrix and load vector due to body force. Figure 5.2 shows the flow chart for the computation of the element matrices.

First the element constants and vectors given in symbolic equations 4.1 and 4.14 are calculated. These constants do not vary for a particular element irrespective of the the hierarchical order chosen for the element. Using these constants and equations, the jacobian components are calculated at all the $8 \times NG_p$ symmetrical points for a particular gaussian point in the η direction. The symmetrical points and the functions for jacobian components are well discussed in chapter: 4. Then the stiffness matrix and load matrix are calculated for all these $8 \times NG_p$ points using the equations 4.18 and 4.21 and the jacobian evaluated previously. Using the above method of evaluation for the jacobian, stiffness and load matrices, considerable computational effort is reduced.

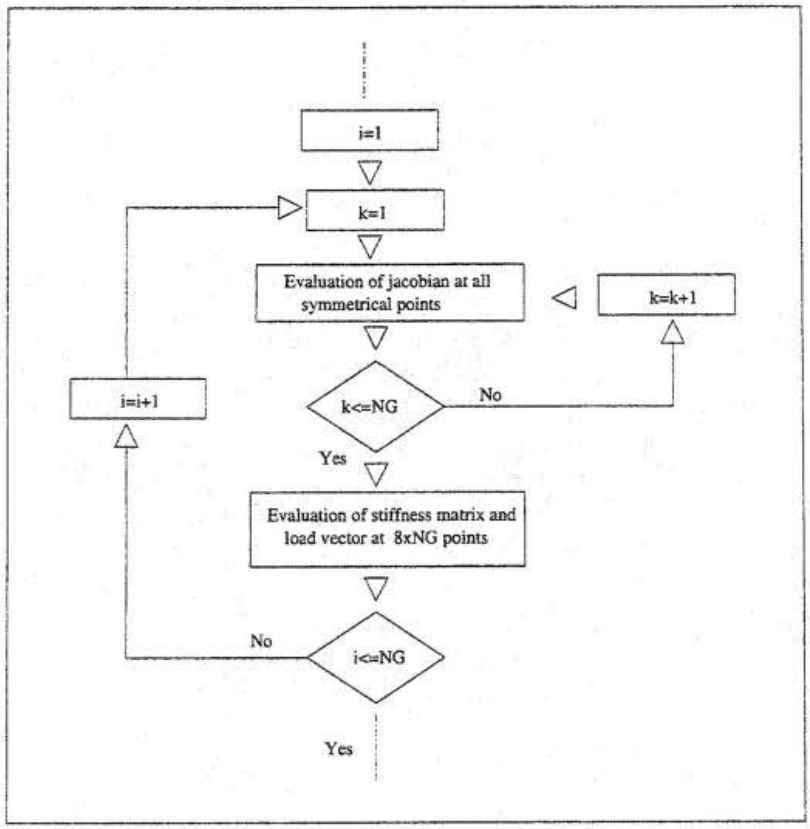

Figure 5.2: Flow chart for the evaluation of element matrices

5.1.3 Precomputed shape function derivatives

The shape function derivatives discussed in section 4.3 are constants for all the elements in the domain and hence need not be calculated at the same gaussian points repeatedly for different elements. These shape function derivatives are calculated once for all the gaussian points for different orders and stored in permanent arrays during the initial stages of the analysis. In this way a considerable reduction computational effort is achieved in the evaluation of shape function derivatives.

5.1.4 Mesh division algorithm

For convergence study, the refinement of mesh coupled with hierarchical analysis is required. The division of an element gives four new elements and the destruction of the old element. The division of an element is carried out at its second order. Figure 5.3 shows the division of an element. The division causes the creation of 16 new nodes, 12 new edges and the destruction of the 4 edges of the original element. The technique automatically identify edges of the neighboring elements and avoids the duplication of edge divisions.

5.2 Analysis Procedure

Analysis of a particular problem is started with a mesh of second order which gives reasonably good solutions. The order of the mesh is increased step by step to see the convergence of displacements and stresses. If convergence is not achieved at higher

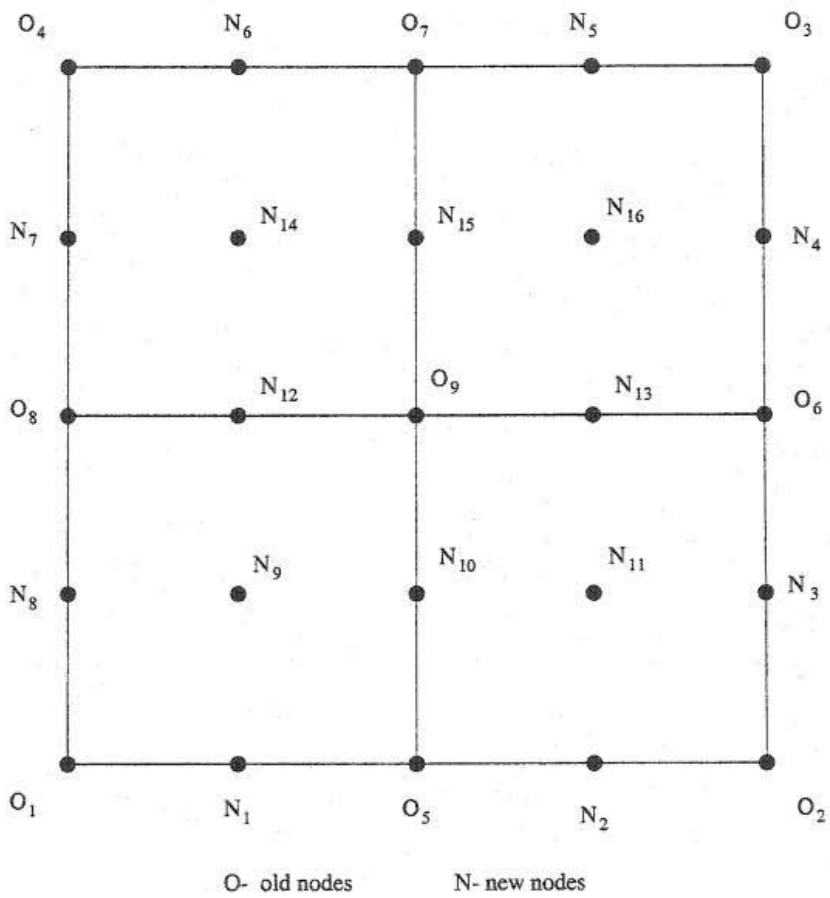

Figure 5.3: Division of an element

orders, the mesh is refined or divided and the analysis is carried out as before until the solutions converge.

5.3 Solution method

The solution is generally carried out either by direct or iterative methods. The direct methods such as gaussian elimination require the assembly of the elemental stiffness matrices, whereas the iterative solvers do not require it. The global stiffness matrix requires too much memory for a large system of equations. This limits the application of a direct solver. Even though direct solvers give exact solutions of the equation, it is limited in application for larger systems. This is due to higher solution time and the requirement of larger memory. The iterative solvers are more suitable for larger system of equations. They are faster and take lesser memory compared to direct solvers. Moreover, the iterative solvers can be much faster if the *initial guess* solution is good. In the hierarchical convergence studies, the higher orders appear as perturbation of lower order solution which makes iterative solvers preferred option.

In this work, the direct solution is carried out first by assembling the stiffness matrices in the skyline format and solving using the gaussian elimination technique. The iterative solver is used in the subsequent stages using the solution obtained at the previous stage as a guess solution. Iterative solution is carried out using the conjugate gradient method, which does not require initial assembly of stiffness matrices and thus saves time and memory. Direct solvers are used at lower orders and smaller meshes.

5.4 Object Oriented Program

The finite element program is developed using C++ , which supports object oriented programming. The basics of OOP are explained in chapter 2. A careful examination of the FE procedure reveals that the data and functions can be grouped into different classes given in appendix B.

Node:

The data and procedures related to the node are grouped under the *node* class. It has the data and procedures which are common to any node irrespective of which kind of element the node belongs to. A more specific class *Shell_node* is derived from the the *Node* class which has its own data and procedures. This inheritance property avoids needless repetition of the functions and data.

Element:

All the data and functions common to an element are grouped under the *Element* class. A more specific class *Shell_element* is derived from *element* class for shell analysis. Each object of the element class refers to an element in the mesh. The number of element objects created is equal to the number of elements used in an analysis. Whenever new elements are created by mesh divisions, corresponding objects are created dynamically. The dynamic initialization also enables the destruction of the object once it is not required further. This feature helps in the effective utilization of the memory.

Gauss:

The class *Gauss* handles the gaussian quadrature and weights for the numerical integration. The data of the class are the gaussian points and weights. An array of objects is created and each object stores the quadrature and weight for a particular order of integration.

Shape:

The class *shape* handles the shape function derivatives with respect to local coordinates ξ, η and ζ. Here an array of objects is created and each object stores the shape function derivatives required for a particular order of integration. These derivatives are evaluated at all the gaussian points required for a particular order of integration and kept in two arrays. The evaluation is carried out at the initial stages of the analysis and stored in permanent arrays. The arrays are initialized dynamically and the memory is allocated according to the requirement for a particular order of integration.

Edge:

The *Edge* class handles the data corresponding to an edge in a finite element mesh. The edge objects are initialized dynamically similar to the *Node* and *Element* objects.

Matrix:

The class *Matrix* [30] is developed to facilitate the mathematical operations using symbolic notations. Using this class, matrices of dimensions 2,3 and 4 can be created. The matrices are created dynamically using different constructors for different matrix

dimensions. The operator overloading feature helps in carrying out mathematical operations using symbolic notations. If a, b and c are matrix objects, then

$a * b$ - matrix multiplication

$a + b$ - Matrix addition

$a - b$ -Matrix subtraction

Similar to matrix class, classes for vectors can be created.

5.5 External functions

The external functions do not come under classes. They are declared in the main program. These functions carry out the manipulation of objects with its functions resulting in the creation of new data.

5.6 External data

The external data similar to external functions, do not come under the classes. These data are common and can be accessed by any object irrespective of which class the object belongs to. The main external data are:

$KG[\ ]$ -Sky line storage of global stiffness matrix. The matrix is initialized dynamically. Once the matrix is of no use after solution, it is deleted by deallocating the memory.

$neqq[\ ][\ ]$ - Used for storing the boundary conditions and degree of freedom of the nodes.

5.7 Advantages of using OOP

1. The data are hidden in the objects. Only the member functions of the object can view the data. Thus the data is safe from accidental change. This increases the reliability, data management and ease of verification of the program.

2. Any inclusion of a new element or technique in the program can be easily done using the inheritance concept. Similar to class *Shell_element*, any class can be derived from the parent *Element* class or from the already derived *Shell_element* class itself. The inherited element can use the already developed functions, so only the functions and data specific to the new element need to be developed. This feature makes the program concise and facilitates extension.

3. As the data and functions are grouped into different classes, it is easy to verify and modify the code according to the requirement during later stages. The modification do not adversely affect other parts of the program. This avoids any major modification which results in ease of maintenance and improvement of the FE code.

4. The dynamic initialization of object and data helps in the optimum usage of memory. Whenever objects and data are not required further, they are deallocated saving the memory.

5. Mathematical operations can be carried out using symbolic notations which results in concise and simple programs.

Chapter 6

NUMERICAL STUDIES AND DISCUSSIONS

Various numerical examples are presented in this chapter to demonstrate the accuracy, advantages and applications of the present element to various problems. The problems analyzed are:

1. Square plate under concentrated and distributed load with different boundary conditions.

2. A Barrel vault(Cylindrical roof) loaded by its self weight.

3. A Hemispherical shell loaded by diametrically opposed point loads in both X and Y directions.

4. A thin Cylindrical shell loaded by two centrally located and diametrically opposed concentrated forces.

The element matrices are evaluated by both reduced $(p)$and full integration $(p+1)$ techniques. The displacements and stresses obtained from the present formulations

are compared with the analytical solutions and the results available in the literature. The results obtained are in excellent agreement with the reference values and they are sometimes more accurate with fewer degrees of freedom. The effect of having an adaptive mesh is also analyzed.

6.1 Square plate problem

An isotropic square plate shown in figure 6.1 is analyzed under different loading and boundary conditions. Using the symmetry, only one quarter of the plate is modelled for the analysis. The plate deforms under bending action and the inplane displacements are constrained in the tangential directions. The analysis is carried out for varying thicknesses to study the element behavior under thick and thin shell situations. The analysis is also carried out for different meshes. The results are compared with the exact values given by Timoshenko. Different cases considered in the analysis of the plate are:

1. Simply supported plate under concentrated load at the center
2. Simply supported plate under uniformly distributed load.
3. Clamped plate under concentrated load at the center.
4. Clamped plate under uniformly distributed load.

Cases 2 and 4 are analyzed for different thicknesses whereas the cases 1 and 3 are analyzed for a refined mesh. The refined mesh has an edge length ratio of 3:7 with

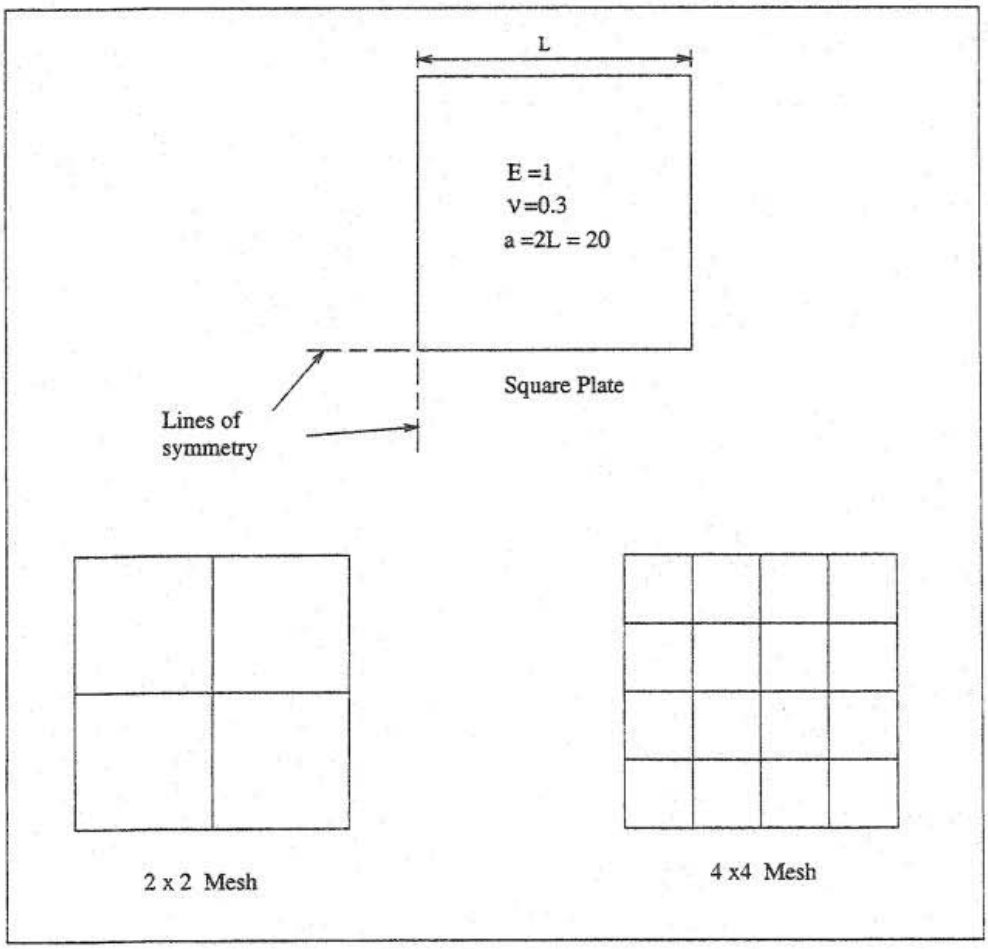

Figure 6.1: Square Plate and meshes used for the analysis

smaller elements near the center of the plate. All the cases are analyzed using 2×2 and 4×4 meshes for orders upto six. The results are given in tables from 6.2 to 6.9. The displacement solutions obtained are normalized using the following formulae.

$$\alpha = \frac{W_{max} D}{qL^4} \text{ for a uniformly distributed load } q.$$

$$\beta = \frac{W_{max} D}{PL^2} \text{ for a central concentrated load } P.$$

Von Mises equivalent stress is given by:

$$\sigma_{von} = \left(\frac{1}{2} \left((\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 \right) + 3 \left(\sigma_{xy}^2 + \sigma_{yz}^2 + \sigma_{zx}^2 \right) \right)^{\frac{1}{2}}$$

The obtained stress results can be made dimensionless using the following equations.

$$\bar{\sigma} = \frac{\sigma_{von} D}{qL^4 Et} \text{ for uniformly distributed load } q.$$

$$\bar{\sigma} = \frac{\sigma_{von} D}{PL^2 Et} \text{ for central concentrated load } P.$$

where W_{max} is the maximum displacement,D-Flexural rigidity and L-the length of the plate.

Discussions:

1. The results obtained are in excellent agreement with the exact values given by Timoshenko [1]. (See table 6.1)

2. The solution converges to the exact value when a 4×4 mesh is used. 2×2 mesh gives reasonably good results with an error of 0.8% in displacement.

3. For a particular mesh type, reasonably good results are obtained at order 4. For orders above 4, the increase in the displacement solution is marginal and it converges towards a particular value.

4. At lower orders, the p-integration gives reasonably good results compared to the p+1 integration. As the order increases, the difference in solution between p and p+1 integration techniques decreases and both converge towards the exact solution.

5. The solution obtained for thickness ratios 100 and 200 indicate (refer figure 6.2) that the element gives accurate results for all the thin plate cases. This shows that the element is free of locking in thin plate cases. However, for obtaining accurate results, the analysis should be carried using higher orders (order>3).

6. The solutions obtained for moderately thick plate (thickness ratios 10 and 20) deviate marginally from the exact thin plate solution (refer figure 6.2). This is due to the shear deformation effect which is not considered in the thin plate theory. Thus the developed element is good for both the thin and moderately thick plate analysis.

7. The analysis of cases 1 and 3 using a refined mesh near the center shows that the improvement in results are only marginal. The improvement using a 4×4 refined mesh is 0.1%.

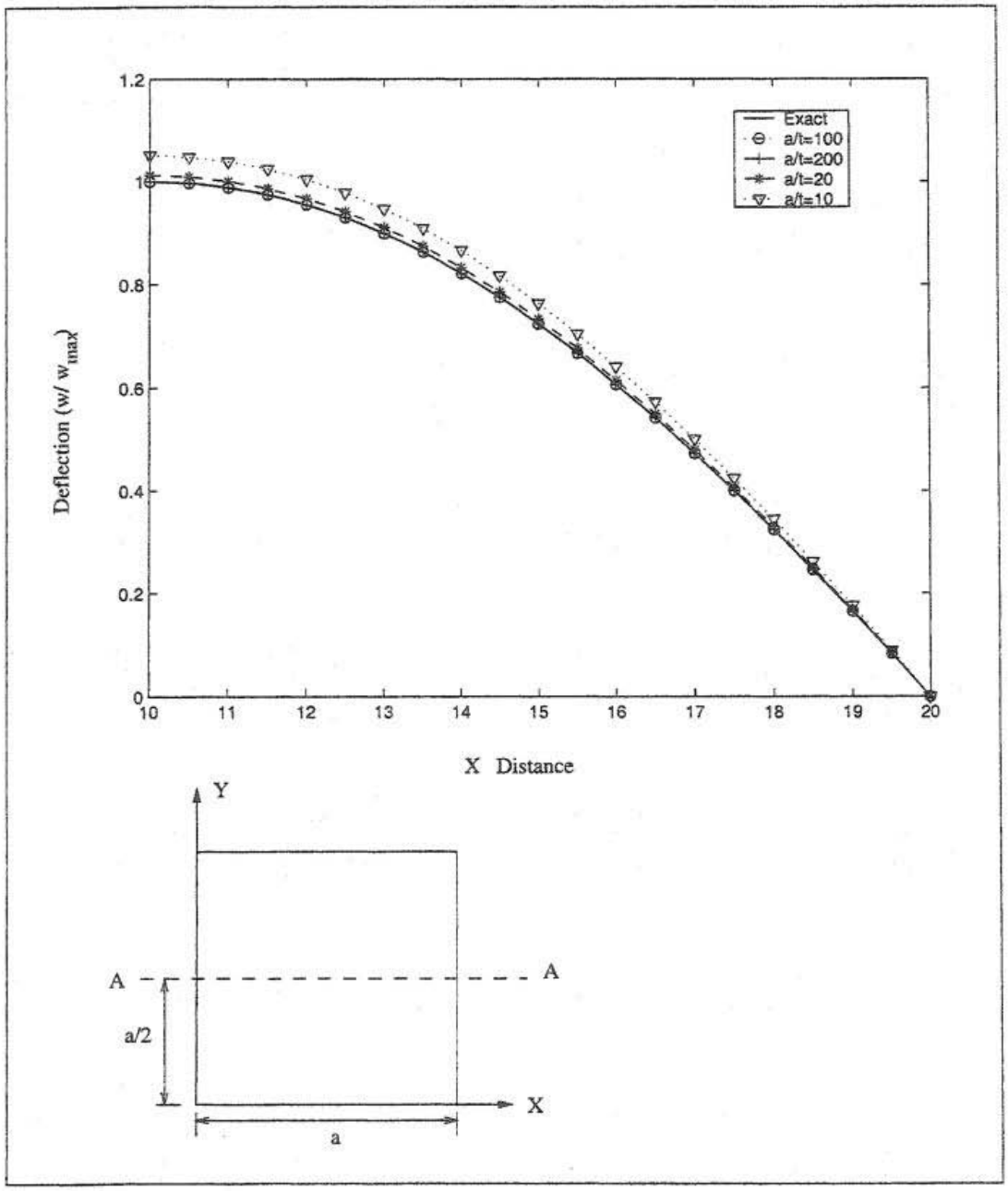

Figure 6.2: Plot of Central deflection along the section A-A of a plate under uniform load

	Uniform load (α)	Concentrated load (β)
Simply Supported Plate	40.62×10^{-04}	0.01160
Clamped Plate	12.6×10^{-04}	0.00560

Table 6.1: Exact values given by Timoshenko[1]

Mesh	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			β	$\bar{\sigma} \times 10^3$	β	$\bar{\sigma} \times 10^3$
	2	48	0.010358	0.8291	0.011640	5.2208
	3	84	0.011403	1.4381	0.011569	2.1382
2×2	4	120	0.011489	1.7435	0.011527	2.1119
	5	156	0.011529	1.9586	0.011536	2.0067
	6	192	0.011534	2.0355	0.011534	2.0326
	2	192	0.011252	1.3397	0.011632	3.5579
	3	336	0.011587	2.0098	0.011615	2.4852
4×4	4	480	0.011605	2.2980	0.011608	2.5372
	5	624	0.011608	2.4457	0.011609	2.4546
	6	768	0.011609	2.4745	0.011609	2.4725

Table 6.2: SS plate under CL: Displacements and stresses for different p-levels

Mesh	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			β	$\bar{\sigma} \times 10^3$	β	$\bar{\sigma} \times 10^3$
	2	48	0.01080	1.1961	0.011646	3.8785
	3	84	0.011515	1.8394	0.011593	2.2766
2×2	4	120	0.011556	2.1395	0.011569	2.3984
	5	156	0.011572	2.2899	0.011576	2.2846
	6	192	0.011574	2.3060	0.011574	2.2944
	2	192	0.011408	1.8438	0.011633	3.2527
	3	336	0.011614	2.4631	0.011623	2.7590
4×4	4	480	0.011620	2.7394	0.01162113	2.8439
	5	624	0.011621	2.7791	0.01162142	2.7973
	6	768	0.011621	2.8091	0.01162145	2.8057

Table 6.3: SS plate under CL(Refined mesh) : Displacements and stresses for different p-levels

a/t	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$	$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$
	2	48	42.559036	6.1782	42.811762	6.8628
	3	84	42.727699	6.6485	42.724441	6.6087
10	4	120	42.723385	6.6409	42.723188	6.6422
	5	156	42.723538	6.6356	42.723242	6.6398
	6	192	42.723285	6.6387	42.723242	6.6398
	2	48	40.691104	6.5689	41.231454	6.8001
	3	84	41.157466	6.6362	41.145936	6.5409
20	4	120	41.144756	6.5658	41.144539	6.5699
	5	156	41.144973	6.5642	41.145002	6.5613
	6	192	41.144631	6.5680	41.144571	6.5702
	2	48	39.844679	6.4319	40.725	7.1920
	3	84	40.665527	6.7625	40.641	6.5622
100	4	120	40.629848	6.5434	40.636	6.6239
	5	156	40.633957	6.5839	40.6344	6.5881
	6	192	40.634313	6.5935	40.6344	6.6037
	2	48	39.810915	6.42468	40.709952	8.3009
	3	84	40.63886	6.7707	40.622912	6.6944
200	4	120	40.590717	6.5172	50.599067	6.6626
	5	156	40.601571	6.5863	40.607379	6.5942
	6	192	40.603450	6.6149	40.603969	6.6466

Table 6.4: Simply supported plate under UDL: Displacements and stresses for different p-levels for 2×2 mesh

a/t	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$	$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$
	2	192	42.716103	6.6837	42.733357	6.6903
	3	336	42.728403	6.6504	42.728471	6.6481
10	4	480	42.728324	6.6503	42.728312	6.6503
	5	624	42.728322	6.6500	42.728338	6.6500
	6	768	42.723285	6.6387	42.723242	6.6398
	2	192	41.101802	6.5975	41154588	6.6170
	3	336	41.149904	6.5813	41.149701	6.5752
20	4	480	41.149653	6.5771	41.149641	6.5770
	5	624	41.149656	6.5769	41.149657	6.5768
	6	768	41.149640	6.5770	41.149636	6.5770
	2	192	40.4734	6.5530	40.649380	6.6218
	3	336	40.645898	6.6084	40.644534	6.5759
100	4	480	40.644481	6.5772	40.644471	6.5777
	5	624	40.644495	6.5774	40.644502	6.5769
	6	768	40.644479	6.5744	40.644475	6.5775
	2	192	40.443609	6.5290	40.633593	6.6366
	3	336	40.630299	6.6142	40.628759	6.5763
200	4	480	40.628590	6.5768	40.628597	6.5805
	5	624	40.628629	6.5776	40.628660	6.5771
	6	768	40.628619	6.5784	40.62817	6.5791

Table 6.5: Simply supported plate under UDL: Displacements and stresses for different p-levels for 4×4 mesh

Mesh	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			β	$\bar{\sigma} \times 10^3$	β	$\bar{\sigma} \times 10^3$
	2	40	0.004021	0.5500	0.005661	4.8341
	3	72	0.005365	1.1427	0.005545	1.8806
2×2	4	104	0.005448	1.4416	0.005499	1.8482
	5	136	0.005494	1.6601	0.005508	1.7292
	6	168	0.005501	1.7568	0.005503	1.7821
	2	176	0.005181	1.0694	0.005645	3.3656
	3	312	0.005599	1.7211	0.005627	2.2142
4×4	4	448	0.005616	2.0391	0.0056520	2.2492
	5	584	0.005620	2.1548	0.005621	2.1645
	6	720	0.005621	2.1839	0.005621	2.1822

Table 6.6: Clamped plate under CL: Displacements and stresses for different p-levels

Mesh	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			β	$\bar{\sigma} \times 10^3$	β	$\bar{\sigma} \times 10^3$
	2	40	0.00410698	0.9462	0.00573866	3.3065
	3	72	0.00534751	1.6047	0.00554412	2.1172
2×2	4	104	0.00540033	1.9123	0.00545614	2.2289
	5	136	0.00545640	2.0916	0.00547534	2.1239
	6	168	0.00546361	2.1400	0.00546984	2.1406
	2	176	0.00525508	1.5769	0.00564976	3.0067
	3	312	0.00562349	2.1783	0.00563388	2.4865
4×4	4	448	0.00563012	2.4539	0.005631174	2.5593
	5	584	0.00563125	2.5139	0.00563155	2.5132
	6	720	0.00563146	2.5251	0.00563152	2.5221

Table 6.7: Clamped plate under CL (Refined mesh): Displacements and stresses for different p-levels

a/t	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$	$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$
	2	40	14.402024	3.3714	15.181394	3.5497
	3	72	15.027938	3.2114	15.032938	3.1344
10	4	104	15.029661	3.1986	15.030136	3.2009
	5	136	15.030604	3.1851	15.030774	3.1818
	6	168	15.030176	3.1868	15.030103	3.1879
	2	40	11.702466	3.0850	13.410317	4.0672
	3	72	13.244227	3.2604	13.253143	3.0889
20	4	104	13.244141	3.1458	13.245671	3.1707
	5	136	13.245951	3.1476	13.246489	3.1433
	6	168	13.245753	3.1815	13.24573	3.1947
	2	40	9.98163	2.8361	12.8164	5.2150
	3	72	12.552414	3.6244	12.627861	2.9444
100	4	104	12.494106	2.9482	12.507677	3.1394
	5	136	12.518231	3.0067	12.540286	3.0913
	6	168	12.524950	3.0831	12.529525	3.1498
	2	40	9.902362	2.8252	12.797384	8.0031
	3	72	12.317853	3.9219	12.588766	4.8914
200	4	104	12.172562	2.9629	12.233130	3.4484
	5	136	12.280628	3.1511	12.254022	3.1281
	6	168	12.288732	3.0108	12.302198	3.1169

Table 6.8: Clamped plate under UDL: Displacements and stresses for different p-levels for 2×2 mesh

a/t	Order	DOF	Full integration (p+1)		Reduced integration (p)	
			$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$	$\alpha \times 10^4$	$\bar{\sigma} \times 10^5$
	2	176	14.991591	3.2754	15.054392	3.2875
	3	312	15.045586	3.2209	15.046093	3.2165
10	4	448	15.045992	3.2209	15.045999	3.2208
	5	584	15.046023	3.2203	15.046026	3.2201
	6	720	15.045995	3.2201	15.045989	3.2200
	2	176	13.090285	3.1820	13.280815	3.2231
	3	312	13.271446	3.1643	13.272364	3.1553
20	4	448	13.272145	3.1579	13.272257	3.1580
	5	584	13.272268	3.1576	13.272286	3.1576
	6	720	13.272250	3.1580	13.272244	3.1581
	2	176	12.04682	3.0760	12.686873	3.2343
	3	312	12.676513	3.1955	12.677913	3.1401
	4	448	12.677133	3.1468	12.677310	3.1496
100	5	584	12.677315	3.1460	12.677417	3.1426
	6	720	12.677316	3.1475	12.677334	3.1478
	2	176	11.975876	3.0651	12.667827	3.3115
	3	312	12.655612	3.2039	12.657885	3.1267
200	4	448	12.656142	3.1451	12.656457	3.1589
	5	584	12.656599	3.1452	12.656940	3.1379
	6	720	12.656589	3.1499	12.655694	3.1540

Table 6.9: Clamped plate under UDL: Displacements and stresses for different p-levels for 4×4 mesh

6.2 Barrel Vault problem

A barrel-vault supported by rigid diaphrams at both ends and loaded by its self weight is shown in figure 6.3. The diaphram prevents the displacement in the Y and Z directions but allows displacements in the X direction. The shell is free along the sides. Only one quarter of the shell is modelled using symmetry. The shell is analyzed for both 2×2 and 4×4 meshes for different orders . The deflection and stresses are noted at point B shown in figure 6.3. The results are given in table 6.11. The results are compared with the solutions obtained using CONSHL and 9 node isoparametric elements given in reference [34]. The reference values are given in table 6.10.

Discussions:

1. The deep shell theory solution for the vertical deflection is 3.610 given in reference [35]. The results obtained are in good agreement with the reference values.

2. The solution converges towards the reference values for both 2×2 and 4×4 cases. However, the results obtained deviates from the reference values by a margin of 0.36%.

3. For order $p = 2$, the displacement solutions obtained using p-integration is fairly accurate compared to p+1 integration. For higher orders($p > 3$), both p and p+1 integration gives accurate results. However the stress results given by

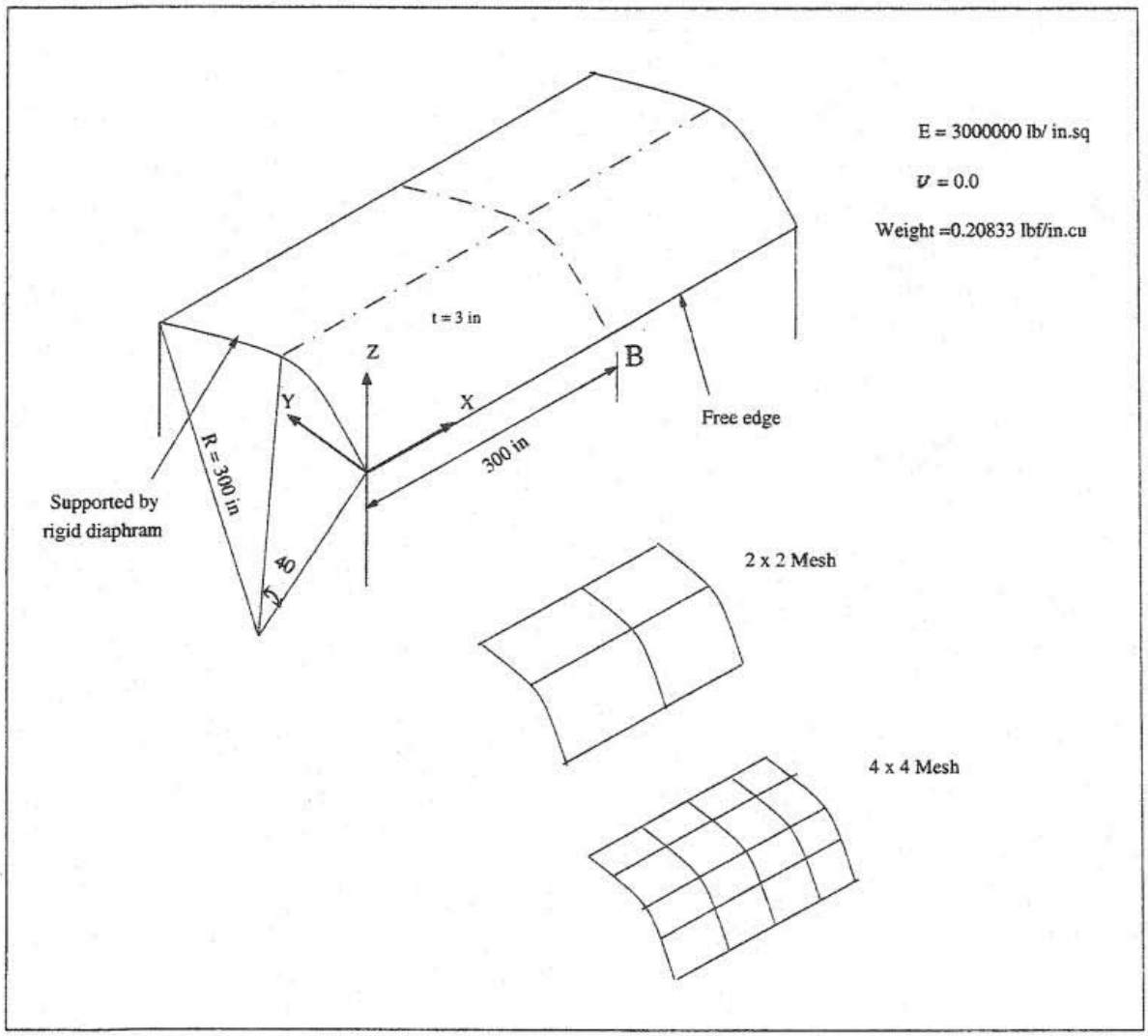

Figure 6.3: Barrel vault and its finite element meshes.

Mesh	CONSHL [34]		9 node isoparametric [34]	
	DOF	Defln(w_B)	DOF	Defln(w_B)
1×1	46	2.28660	-	-
2×2	162	3.10276	96	0.96239
4×4	604	3.51078	352	2.90531
6×6	-	-	768	3.38604
8×8	-	-	1344	3.48602

Table 6.10: Barrel-vault problem: Reference Values

p-integration at lower orders are poor.

4. Improved results are obtained in lesser number of degrees of freedom compared to the references.

6.3 Hemispherical Shell with 18^o hole

The performance of the shell element is also evaluated using a standard test problem of a hemispherical shell with a hole shown in figure 6.4. Diametrically opposed point loads are applied along both X and Y directions. The analysis is carried out using mesh sizes 6×6 and 8×8 meshes for different orders. The deflection and stresses are noted at the load application point. The results obtained are shown in table 6.12.

mesh	order	DOF	Full integration (p+1)			Reduced integration (p)		
			v_B	w_B	σ_{von}	v_B	w_B	σ_{von}
	2	92	0.372100	-0.957140	1040.506	1.987855	-3.761128	3088.372
	3	158	1.912085	-3.614304	1740.534	1.949662	-3.686610	2137.938
2×2	4	224	1.934311	-3.659151	1737.825	1.941783	-3.672662	1700.015
	5	290	1.932715	-3.658392	1681.836	1.938505	-3.667703	1670.697
	6	356	1.933390	-3.659472	1702.009	1.937836	-3.666542	1713.927
	2	344	1.541941	-2.978150	1538.454	1.932873	-3.659486	1755.367
	3	596	1.922040	-3.640431	1714.368	1.927410	-3.649818	1702.1955
4×4	4	848	1.923212	-3.643108	1680.741	1.926985	-3.649142	1680.0102
	5	1100	1.924137	-3.644597	1683.046	1.926948	-3.649085	1685.55
	6	1352	1.925004	-3.645980	1685.724	1.926945	-3.649078	1687.58

Table 6.11: Barrel Vault: Displacements and stresses for different p-levels

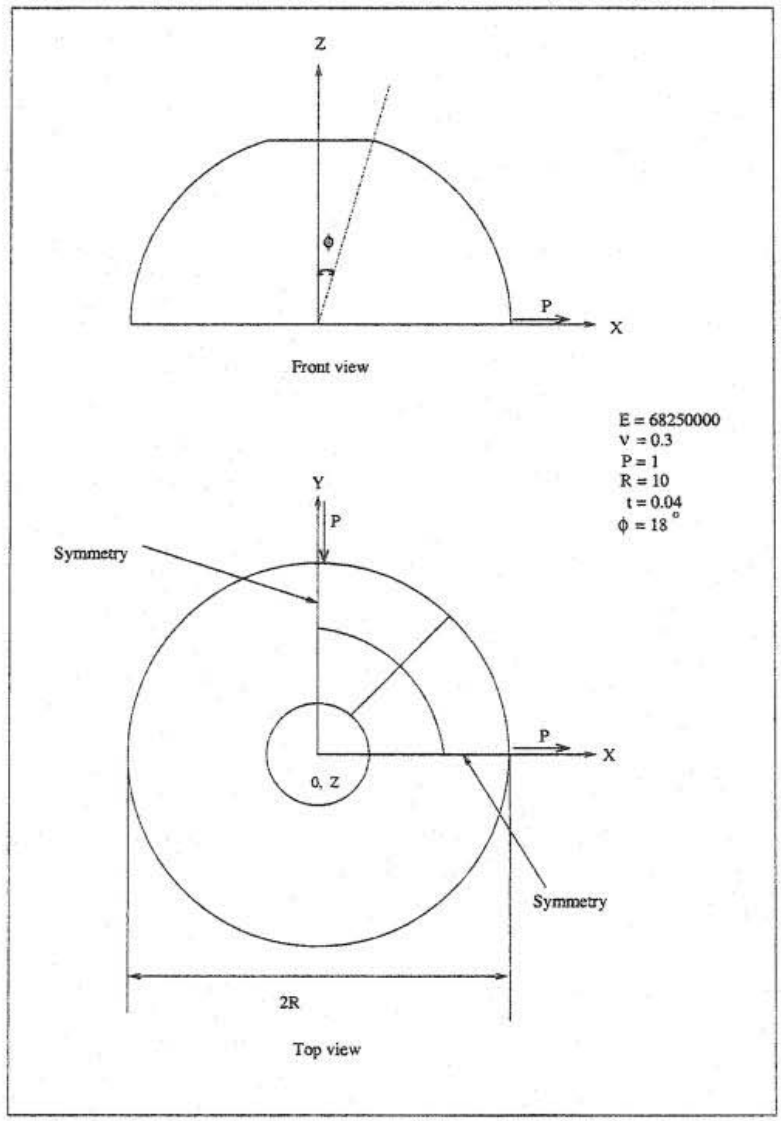

Figure 6.4: Pinched hemispherical shell with a hole and finite element model.

Discussions:

1. MacNeal and Harder [36] gave a value of 0.094 for the deflection. Simo et al suggested [37] a value of 0.093. The obtained results are in excellent agreement with the reference values. Good results are obtained for both the kind of meshes.

2. It is seen that very poor displacement solutions are obtained at lower orders when p+1 integration is used; whereas, reduced integration gives fairly accurate results. However, reduced integration gives poor results in stresses at lower orders.

3. Both p and $p+1$ integration techniques gives reasonably accurate solutions above order 3. The accurate results shows that the element is free of locking at higher orders.

6.4 Pinched Cylindrical Shell

A cylindrical shell shown in figure 6.5 is loaded by two centrally located and diametrically opposed concentrated forces is analyzed. Two types of boundary conditions are considered.

1. The ends are covered by a rigid diaphram which allow displacement only in the axial direction and rotation about the tangent to the shell boundary.

2. The ends are free.

mesh	order	DOF	Full integration (p+1)		Reduced integration (p)	
			δ	σ_{von}	δ	σ_{von}
	2	793	0.009434	3107.2894	0.094383	166479.385
	3	1369	0.090686	4684.8534	0.09323	101848.110
6×6	4	1945	0.092695	4815.711	0.093178	9181.2165
	5	2521	0.092882	5481.421	0.093084	6829.915
	6	3097	0.092922	5737.211	0.093075	6061.283
	2	1377	0.025288	5505.55	0.094329	154163.298
	3	2385	0.092883	5210.000	0.093929	213356.661
8×8	4	3393	0.093666	5519.417	0.093905	8141.598
	5	4401	0.093725	6080.657	0.093886	6781.044
	6	5409	0.093752	6322.967	0.093878	6400.492

Table 6.12: Hemispherical shell: Displacements and stresses for different p-levels

Mesh	Taylor [38]
4×4	0.086524
8×8	0.094153
12×12	0.093679
16×16	0.093501

Table 6.13: Hemispherical shell: Reference values

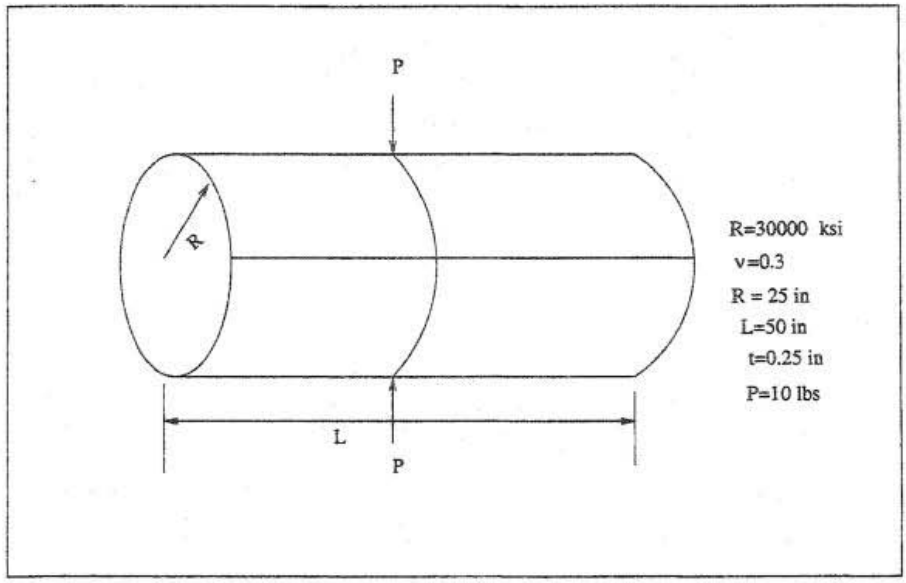

Figure 6.5: Pinched cylindrical shell, loading and dimensions.

Using the double symmetry, only one eighth of the cylinder is modelled. The deflection and the stresses are noted at the load application point. The results obtained are given in tables 6.14 and 6.17. For a rigid diaphram case the exact displacement can be taken as 0.2189866 given by Lindberg [39]. For the free ends case, a good approximation is given by [40].

$$\delta = \frac{0.0745PR^3}{2DL} = 5.4236$$

where R is the radius, L-Half length, t- thickness, D-flexural rigidity and P-magnitude of the force.

Discussions:

1. The results are in good agreement with the reference values given in tables 6.15 and 6.16. In the first case, 8×8 mesh gives a displacement of -0.216463 with an error of 1.1%. In the second case, 8×8 mesh gives a displacement of -5.702495 with 5.1% error.

2. The results obtained using a refined mesh are given in tables 6.18 and 6.19. The usage of a refined mesh near the center gives better results than a uniform mesh. A refined 4×4 mesh gives a displacement solution -0.217503 with an error of 0.67% for the first case which is better than the results obtained by a 8×8 mesh. This show the usage of an adaptive mesh can substantially reduce the computational effort in this case.

3. As seen in the previous example problems, p integration gives fairly good displacement solution at lower orders.

mesh	order	DOF	Full integration (p+1)		Reduced integration (p)	
			δ	σ_{von}	δ	σ_{von}
	2	351	-0.344940	58.25696	-6.074268	4481.965
	3	607	-5.47641	178.715	-5.826518	438.722
4×4	4	863	-5.81271	226.5698	-5.827853	330.805
	5	1119	-5.823329	259.2499	-5.826866	310.431
	6	1375	-5.825999	280.7692	-5.826972	303.431
	2	1343	-2.893403	192.237	-5.768616	1844.044
	3	2335	-5.676699	279.4197	-5.703403	406.242
8×8	4	3327	-5.700569	321.4383	-5.702514	375.915
	5	4319	-5.702098	347.7745	-5.702545	366.086
	6	5311	-5.702495	362.560	-5.702551	368.172

Table 6.14: Cylindrical shell, free ends: Displacements and stresses for different p-levels

	CONSHL [34]		9 node isoparametric [34]	
Mesh	DOF	Defln(δ)	DOF	Defln(δ)
2×2	165	3.17403	95	0.02909
4×4	607	4.88251	351	0.33731
6×6	1329	5.07955	767	1.32131
8×8	2331	5.12217	1343	2.67958
16×16	-	-	5247	4.90646
20×20	-	-	8159	5.08053

Table 6.15: Cylindrical shell, free ends: Reference Values

Mesh	CONSHL		9 node isoparametric	
	DOF	Defln(δ)	DOF	Defln(δ)
6×6	1284	0.16921	734	0.07965
10×10	3536	0.20439	2022	0.15596
14×14	6908	0.21366	3950	0.18976
18×18	11400	0.21703	6518	0.20376
22×22	-	-	9726	0.21054
26×26	-	-	13574	0.21428

Table 6.16: Cylindrical shell, diaphramed ends: Reference Values

mesh	order	DOF	Full integration (p+1)		Reduced integration (p)	
			δ	σ_{von}	δ	σ_{von}
	2	328	-0.034825	37.4533	-0.213514	648.468
	3	572	-0.118245	79.822	-0.200309	342.127
4×4	4	816	-0.186143	126.1358	-0.202316	245.497
	5	1060	-0.197897	162.8665	-0.201361	214.482
	6	1304	-0.200528	184.734	-0.201470	207.455
	2	1296	-0.123709	98.5164	-0.221637	607.251
	3	2264	-0.204047	183.2335	-0.217313	322.566
8×8	4	3232	-0.214558	231.8988	-0.216473	289.599
	5	4200	-0.216074	257.7662	-0.216504	274.844
	6	5168	-0.216463	271.34165	-0.216511	277.082

Table 6.17: Cylindrical shell, diaphramed ends: Displacements and stresses for different p-levels

mesh	order	DOF	Full integration (p+1)		Reduced integration (p)	
			δ	σ_{von}	δ	σ_{von}
	2	351	-0.36801	93.6067	-5.926577	3421.523
	3	607	-5.463178	261.2930	-5.826124	408.551
4×4	4	863	-5.815227	303.002	-5.4909	368.794
	5	1119	-5.824009	332.207	-5.825030	536.092
	6	1375	-5.824777	348.798	-5.824977	356.392

Table 6.18: Cylindrical shell, free ends: Displacements and stresses for a refined mesh

mesh	order	DOF	Full integration (p+1)		Reduced integration (p)	
			δ	σ_{von}	δ	σ_{von}
	2	328	-0.079245	76.5209	-0.221415	607.143
	3	572	-0.183349	158.8652	-0.218160	322.519
4×4	4	816	-0.213509	210.105	-0.217719	280.372
	5	1060	-0.216841	239.496	-0.217700	261.594
	6	1304	-0.217503	254.952	-0.217617	262.481

Table 6.19: Cylindrical shell, diaphramed ends: Displacements and stresses for a refined mesh

Chapter 7

Conclusion

A hierarchical nine node p-version curved shell element is successfully developed incorporating symbolic computations. The formulation is based on the assumption that the normal to the middle surface remains practically straight after deformation, which permits the shear deformation effect prominent in thick shells. The displacement approximation function can be of any order in the ξ and η directions. For plates and shells, the displacement variation across the thickness is practically linear, hence no hierarchical variation is needed in the ζ direction. The displacement approximation functions are derived from the Lagrangian family. In this formulation, both the approximation functions and nodal variables are hierarchical in nature, i.e. the element properties corresponding to a particular order p is a subset of those corresponding to order $p+1$. The element geometry is approximated using the top and bottom plane coordinates of the nodes. The lower order matrices need not be evaluated in the subsequent computation of higher order matrices. Only the additional higher order matrices need to be evaluated, thus reducing computational effort. However, this can

be done only for higher orders ($p > 3$) because the stiffness matrix do not converge at lower orders.

An attempt is made to evaluate the closed form stiffness matrix by direct integration using symbolic computations. This is possible for a rectangular flat plate where the jacobian is constant for the element, but the direct integration becomes impossible for the case where the jacobian is a function of ξ, η and ζ. An alternative general symbolic technique for the calculation of element matrices is developed. A number of locations where the usage of symbolic computations result in reduction in computational effort is identified. Effective methods for the evaluation of the jacobian, stiffness matrix, strain matrix and load vectors using symbolic computations are incorporated in the development of the FE code. The FE code is developed using the object oriented programming language C++. OOP helps to develop efficient codes with ease of maintenance, verification, re-usability and extension.

The various numerical problems presented show that the results given by the element is in good agreement with the reference values. It is also shown that the element is able to converge to the exact solution in many cases using fewer degrees of freedom. The element doesn't show any locking problem at higher orders ($p > 3$). At lower orders p-integration is able to give relatively good results compared to $p+1$ integration. The formulation is effective for both thin and moderately thick shells.

Recommendations

1. Initial mesh design does not affect the convergence rate critically. However, better results can be obtained by using a combination of mesh divisions (h-refinement) and p-refinement.

2. The direct integration of element matrices for a rectangular flat plate is possible using symbolic computation. In such case, a large reduction in computational effort is possible. In the case of curved shells and irregular plates, direct integration is not possible as such. However, direct integration can be done by making the elements smaller and rectangular with few assumptions about the jacobian components. This can be done in future work.

Bibliography

[1] S.Timoshenko and S.Woinowsky-Krieger, *Theory of plates and shells.* Mc-Grawhill, London,Toronto and Newyork, 1957.

[2] O.C.Zienkiwicz and Y.K.Cheung, *The finite element method in structural and continuum mechanics.* McGrawhill, London and Newyork, 1967.

[3] K.J.Bathe, *Finite Element Procedure in Engineering Analysis.* Prentice Hall, NJ, 1982.

[4] B.E.Greene, D.R.Strome, and R.C.Weikel, "Application of the stiffness method to the analysis of shell structures," in *In proc. Aviation conference of ASME, Los Angeles, CA*, march 1961.

[5] H.T.Y.Yang, S.Saigal, and D.G.Liaw, "Advances of thin shell finite elements and some applications ver1," *Computers and Structures*, vol. 35, pp. 481–504, 1990.

[6] R.W.Clough and C.P.Johnson, "A finite element Approximation for the analysis of thin shells," *International Journal of solids and structures*, vol. 4, pp. 43–60, 1968.

[7] S.Ahmed, B.M.Irons, and O.C.Zienkiwicz, "Analysis of thick and thin shell structures by curved finite elements," *International Journal for Numerical Methods in Engineering*, vol. 2, pp. 419–451, 1970.

[8] C.K.Lee and R.E.Hobbs, "Automatic adaptive refinement for shell analysis using nine node assumed strain element," *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 3601–3638, 1997.

[9] K.Y.Sze, D. Zhu, and D. peng Chen, "Quadratic triangular C^o plate bending element," *International Journal for Numerical Methods in Engineering*, vol. 40, pp. 937–951, 1997.

[10] K.J.Bathe and E.N.Dvorkin, "Short Communication - A four node plate bending element on Mindlin/Reissner plate theory and mixed interpolation," *International Journal for Numerical Methods in Engineering*, vol. 21, pp. 367–383, 1985.

[11] K.J.Bathe, "A formulation of general shell elements- The use of mixed interpolation of tensorial components," *International Journal for Numerical Methods in Engineering*, vol. 22, pp. 697–722, 1986.

[12] A.F.Saleeb and T.Y.Chang, "An efficient Quadrilateral element for plate bending Analysis," *International Journal for Numerical Methods in Engineering*, vol. 24, pp. 1123–1155, 1987.

[13] H.R.H.Kabir, "A shear locking free isoparametric three node triangular finite element for moderately thick and thin plates," *International Journal for Numerical Methods in Engineering*, vol. 35, pp. 503–519, 1992.

[14] G.Prathap and B.R.Somashekar, "Field and edge consistency synthesis of a 4-noded quadrilateral plate bending element," *International Journal for Numerical Methods in Engineering*, vol. 26, pp. 1693–1708, 1988.

[15] A.Peano, "Hierarchies of conforming finite elements for plane elasticity and plate bending," *Comp. and math. with Appls.*, vol. 2, pp. 211–224, 1976.

[16] M.P.Rossow and I.N.Katz, "Hierarchical finite element and precomputed arrays," *International Journal for Numerical Methods in Engineering*, vol. 12, pp. 977–999, 1978.

[17] O.C.Zienkiwicz, J.P.De, S.R.Gago, and D.W.Kelly, "The hierarchical concept in finite element analysis," *Computers and Structures*, vol. 16, pp. 53–65, 1983.

[18] B.A.Szabo, "Mesh design for the p-version of the finite element method," *Computer Methods in Applied Mechanics and Engineering*, vol. 55, pp. 181–197, 1986.

[19] I.Babuska, M.Griebel, and J.Pitkaranta, "The problem of selecting the shape functions for a p-type finite element," *International Journal for Numerical Methods in Engineering*, vol. 28, pp. 1891–1908, 1989.

[20] R.B.Morris, Y.Tsuji, and P.Carnevali, "Adaptive solution strategy for solving large systems of p-type finite element equation," *International Journal for Numerical Methods in Engineering*, vol. 33, pp. 2059–2075, 1992.

[21] K.S.Woo and P.K.Basu, "Analysis of singular cylindrical shells by p-version of FEM," *International Journal of Solids and Structures*, vol. 25, no. 2, pp. 151–165, 1989.

[22] B.A.Szabo and G.J.Sahrmann, "Hierarchic plate and shell elements," *Int. J. Numer methods Engng*, vol. 26, pp. 1855–1881, 1988.

[23] K.S.Surana and R.M.Sorem, "P-version hierarchical three dimensional curved shell element for elastostatics," *International Journal for Numerical Methods in Engineering*, vol. 31, pp. 649–676, 1991.

[24] T.Y.Chang, H.Q.Tan, D.Zheng, and M.W.Yuan, "Application of Symbolic method to hybrid/mixed finite elements and computer implementation," *Computers and Structures*, vol. 35, pp. 293–299, mar 17–19 1989.

[25] G. Rangaraju, N. F. Jr, and M. A. Aminpour, "Comparison of symbolic and numerical integration methods for an assumed stress hybrid shell element," *Communications in numerical methods in engineering*, vol. 11, pp. 307–316, 1995.

[26] A.R.Korncoff and S.J.Fenves, "Symbolic generation of finite element stiffness matrices," *Computers and Structures*, vol. 10, pp. 119–124, 1979.

[27] A.K.Noor and C.M.Andersen, "Computerized symbolic manipulation in Nonlinear Finite Element Analysis," *Computers and Structures*, vol. 13, pp. 379–403, 1981.

[28] B. W.R.Forde, R.O.Foschi, and S.F.Stiemer, "Object Oriented Finite Element Analysis," *Computers and Structures*, vol. 34, no. 3, pp. 355–374, 1990.

[29] J.Besson and R.Foerch, "Large scale Object oriented finite element code design," *Computer methods in applied mechanics in Engineering*, vol. 142, pp. 165–187, 1997.

[30] S.P.Scholz, "Elements of an object oriented FEM++ program in C++," *Computers and Structures*, vol. 43, pp. 517–529, 1992.

[31] B. Jeremic and S.Strue, "Tensor Objects in finite element programming," *International Journal for Numerical Methods in Engineering*, vol. 41, pp. 113–126, 1998.

[32] D.Eyheramendy and T.Zimmermann, "Object oriented finite elements II. A symbolic environment for automatic programming," *Computer Methods in Applied Mechanics and Engineering*, vol. 132, pp. 277–304, 1996.

[33] Y. D. Pelerin and T.Zimmermann, "Object oriented programming. III. An efficient implementation in C++," *Computer Methods in Applied Mechanics and Engineering*, vol. 108, pp. 165–183, 1993.

[34] B. L.Koxiey and F. A.Mirza, " Consistent thick shell element," *Computers and Structures*, vol. 31, pp. 531–549, 1997.

[35] H.Parish, "A critical survey of the nine node degenerated shell element with special emphasis on thin shell application and reduced integration," *Computer Methods in Applied Mechanics and Engineering*, vol. 20, pp. 323–350, 1979.

[36] R.H.MacNeal and R.L.Harder, "A proposed standard set of problems to test finite element accuracy," *Journal of Finite Elem. Anal. Des.*, vol. 1, pp. 3–20, 1985.

[37] J.C.Simo, D.D.FOx, and M.S.Rifai, "On a stress resultant geomtrically exact shell model Part II: The linear theory; Computational aspects," *Computer Methods in Applied Mechanics and Engineering*, vol. 73, pp. 53–92, 1989.

[38] R.L.Taylor, "Finite element analysis of linear shell problems," *Proceedings of the mathematics of Finite Elements and Applications, Academic Press, Newyork*, pp. 211–223, 1987.

[39] G.M.Lindberg, M.D.Olson, and G.R.Cowper, "New developments in the finite element analysis of shells," *Quat. Bul. Div. Mech. Engng and Nat. Aero. Estab, National Research Counsil of Canada, Division of Mechanical Engineering*, vol. 4, pp. 1–38, 1969.

[40] S.Lukasiewicz, *Local Loads in Plate and Shells.* Sijithoff and Noordhoff, Alphen and den Rijn, 1979.

Appendix A

Shape functions

A.1 One dimensional Lagrangian Shape functions

Linear:

$$N_1 = \frac{1}{2}(1-\xi)$$

$$N_2 = \frac{1}{2}(1+\xi)$$

Quadratic:

$$N_1 = -\frac{1}{2}\xi(1-\xi)$$

$$N_2 = (1+\xi)(1-\xi)$$

$$N_3 = \frac{1}{2}\xi(1+\xi)$$

Cubic:

$$N_1 = -\frac{9}{16}(1-\xi)(\frac{1}{3}+\xi)(\frac{1}{3}-\xi)$$

$$N_2 = \frac{27}{16}(1+\xi)(1-\xi)(\frac{1}{3}-\xi)$$

$$N_3 = \frac{27}{16}(1+\xi)(1-\xi)(\frac{1}{3}+\xi)$$

$$N_4 = -\frac{9}{16}(1+\xi)(\frac{1}{3}+\xi)(\frac{1}{3}-\xi)$$

A.2 Hierarchical Lagrangian Shape functions for second order

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta)$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta)$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta)$$

$$N_4 = \frac{1}{4}(1-\xi)(1+\eta)$$

$$N_5 = \frac{1}{4}(1-\eta)(\xi^2-1)$$

$$N_6 = \frac{1}{4}(1+\xi)(\eta^2-1)$$

$$N_7 = \frac{1}{4}(1+\eta)(\xi^2-1)$$

$$N_8 = \frac{1}{4}(1-\xi)(\eta^2-1)$$

$$N_9 = \frac{1}{4}(\xi^2-1)(\eta^2-1)$$

Appendix B

Classes used in the FE program

```
#ifndef MATRIX_HPP
#define MATRIX_HPP
#include <stdlib.h>
#include <iostream.h>
#include <stdio.h>

class Matrix {
private:
double **mm;  //base pointer
double ***pp;
double ****ff;
int r,c,ra,rb,rc,rd;
public:
//constructors and destructors
Matrix();  //creates 3x3 matrix
Matrix(int row, int col); //creates a 2-D array
Matrix(int row1, int row2, int row3); //creates a 3-D array
Matrix(int row1,int row2,int row3,int row4); //creates a 4-D array
double& operator() (int row, int col);
double& operator() (int row1, int row2, int row3);
double& operator() (int row1, int row2, int row3, int row4);
Matrix(int rows, int columns,double initval); //initialization by a value initv
Matrix(Matrix& x, int columns); //initialization by a matrix 'x'
```

```
Matrix& operator=(Matrix& x);   // Assignment operator
Matrix& operator=(double d); // Assignment operator
Matrix operator+(Matrix& x); // Addition operator
Matrix& operator+=(Matrix& x); // Addition operator
Matrix operator*(Matrix& x); // Multiplication Operator
Matrix& operator*=(Matrix& x); // Multiplication Operator
Matrix operator*(double d); // Multiplication Operator
Matrix& operator*(double d); // Multiplication Operator
~Matrix();  //destructor
int upper_row() { return (r-1); }
int upper_col() { return (c-1); }
};

class Node
{
private:
double X,Y,Z;
public:
virtual void read_coordinates()=0;
};

class Element
{
private:
int  order;
public:
virtual void read_nodconnect()=0;
virtual void jacobian()=0;
};

class Shape
{
private:
int order;
public:
```

```
};

class Material
{
private:
double mu, Elas_const;
int mat_num;
public:
virtual void read_properties()=0;
};

class Gauss
{
public:
};

class Shell_node: public  Node
{
public:
static int num_nodes; //number of nodes
double hx,hy,hz; //top and bottom  coordinates
void read_coordinates();
void create_node(int );
double get_x() {return X;}
double get_y() {return Y;}
double get_z() {return Z;}
friend void bound();
~Shell_node() { }
};

class Shell_gauss: public Gauss
{
private:
double t[6],w[6];
public:
```

```
int  g;
Shell_gauss(int);
double get_t(int q) { return t[q]; }
double get_w(int q) {return w[q]; }
friend void read_gauss();
~Shell_gauss() { }
};

class Shell_shape: public Shape
{
public:
int order;
Matrix n,m;
private:
void initialize();
\\Shape function derivatives of different order shape functions
void n1dx(int,int);
void n2dx(int,int);
void n3dx(int,int);
void n4dx(int,int);
void n5dx(int,int);
void n6dx(int,int);
public:
Shell_shape(int ord,int num) : n(10,num,35,5),m(10,num,35,5)
{
order=ord;
initialize();
}
~Shell_shape() { }
};


class Shell_element: public Element
{
private:
```

```
Matrix klocal,a,v1;   // Stiffness, constant and vector matrices
double h[10],x[10],y[10],z[10],h1[10],h2[10],h3[10];
double load_vec[201];
static int status;
int elem_no;
void eval_K(double,double);
public:
static int num_elements,RAD;
static float BL;
int EC[5];
int center_node[6];
Matrix strain,stress;
double Disp[150];
int connect[40];
int  order;
void read_nodconnect();
void make_vectors();
void constant(); //Evaluate constants
void jacobian(int,int,Shell_gauss *ptr); //Calculate jacobian
void Jacobn(double r,double s,double t);
// Stiffness matrix calculation
void stiffness(int,int,Shell_shape *sptr,Shell_gauss *ptr);
//Strain matrix calculation
void B_matrix(int,int,int,Shell_shape *sptr);
//Dynamic initialization of Matrices
Shell_element (void) :
klocal(201,201),a(4,25),v1(4,201),strain(14,7),stress(14,7)
{ order=2; modify=0; }
void assembly();  // Assembly of stiffness matrices
void body_force(Shell_shape *sptr,int p,int comb,double
DET,double weight);
void print_stiff();
void check();
void Bstrain(Shell_shape *sptr,Shell_gauss *ptr);
void stress_strain(int,double); //Calculate stresses and strains
```

```
// Functions for the Conjugate Gradient solver
void eval_diagonal(int pn);
void eval_SP(double *P,int pn);
// Functions for mesh divisions and order increment
void eval_newcoords();
void create_element(int connect[4][9], int r,int
edconn[4][4]);
void make_element(int E);
void edge_connectivity(int eno, int ec1,int ec2,int
ec3,int ec4);
void add_centernode();
void setnodal_connect();
void load_assembly(); // Assembly of load vectors
void Disp_vector(); // Displcement vetor creation
void initialize();
friend void elnode(Shell_element *eptr);
friend void read_nodloads();
friend void solution_tech_1(Shell_element *eptr);
friend void jcg_solver(int ee, Shell_element *eptr);
friend void refine_element( Shell_element *eptr, int eno);
~Shell_element() { }
};

class Edge
{
private:
};

class Shell_edge: public Edge
{
private:
Matrix STRESS; // Stresses at the centre of each edge
int edno,order; //Edge number and order
public:
static int num_edges;
```

```
static float EL; //Edge load
int bound; // Boundary edge
int div_stat,other_ed;
int node_stat,load_stat;
int Ec[3], Nc[3],c1,c2,mid[7];// nodes and elements in an edge
Shell_edge(void): STRESS(3,6)
{ ERROR=0;Ec[1]=0;Ec[2]=0;
 Nc[1]=0; Nc[2]=0; c1=0; c2=0;
bound=0;node_stat=0;order=2;load_stat=0;}
void edge_data(int E,int s, int n1, int n2, int m);
void find_error(Shell_element *eptr,int edno); // Error in stresses
between adjascent elements
//Functions for edge divisions
void create1_edge(int n1,int n2,int m,int r,int pos,int
elmno,int BOUND,int LS);
void create2_edge(int pos,int elmno);
void create_edge(int N1,int N2,int m,int E1,int E2,int
pos1,int pos2);
void create_newedges(int *old,int *New,int *E,int *N_E);
void increment_node();
void edge_load(Shell_element *eptr);
// Set the boundary constraints for the new nodes and edges
void set_neqq1();
void set_neqq2();
friend void findedge_load(Shell_element *eptr);
~Shell_edge() { }
};

#endif
```

Appendix C

Jacobian components evaluated using MAPLE

```
> n1:=1/4*(1-xi)*(1-eta)*(-xi-eta-1):
> n2:=1/4*(1+xi)*(1-eta)*(xi-eta-1):
> n3:=1/4*(1+xi)*(1+eta)*(xi+eta-1):
> n4:=1/4*(1-xi)*(1+eta)*(-xi+eta-1):
> n5:=1/2*(1-xi^2)*(1-eta):
> n6:=1/2*(1-eta^2)*(1+xi):
> n7:=1/2*(1-xi^2)*(1+eta):
> n8:=1/2*(1-eta^2)*(1-xi):
> x:=n1*x1+n2*x2+n3*x3+n4*x4+n5*x5+n6*x6+n7*x7+n8*x8+n1*ze
  ta/2*v3x1+n2*zeta/2*v3x2+n3*zeta/2*v3x3+n4*zeta/2*v3x4+n
  5*zeta/2*v3x5+n6*zeta/2*v3x6+n7*zeta/2*v3x7+n8*zeta/2*v3
  x8:
> y:=n1*y1+n2*y2+n3*y3+n4*y4+n5*y5+n6*y6+n7*y7+n8*y8+n1*ze
  ta/2*v3y1+n2*zeta/2*v3y2+n3*zeta/2*v3y3+n4*zeta/2*v3y4+n
  5*zeta/2*v3y5+n6*zeta/2*v3y6+n7*zeta/2*v3y7+n8*zeta/2*v3
  y8:
> z:=n1*z1+n2*z2+n3*z3+n4*z4+n5*z5+n6*z6+n7*z7+n8*z8+n1*ze
  ta/2*v3z1+n2*zeta/2*v3z2+n3*zeta/2*v3z3+n4*zeta/2*v3z4+n
  5*zeta/2*v3z5+n6*zeta/2*v3z6+n7*zeta/2*v3z7+n8*zeta/2*v3
  z8:
> j11:=collect(expand(diff(x,xi)),[xi,eta,zeta],distribute
  d);
```

$$
\begin{aligned}
j11 := & -\frac{1}{2}x8+\frac{1}{2}x6+\left(\frac{1}{8}v3x3-\frac{1}{8}v3x4+\frac{1}{8}v3x1-\frac{1}{8}v3x2\right)\eta\,\zeta \\
& +\left(\frac{1}{4}v3x3-\frac{1}{2}v3x5-\frac{1}{2}v3x7+\frac{1}{4}v3x1+\frac{1}{4}v3x4+\frac{1}{4}v3x2\right)\zeta\,\xi \\
& +\left(\frac{1}{2}x1-x7-x5+\frac{1}{2}x2+\frac{1}{2}x3+\frac{1}{2}x4\right)\xi+\left(-\frac{1}{4}x2+\frac{1}{4}x1+\frac{1}{4}x3-\frac{1}{4}x4\right)\eta \\
& +\left(\frac{1}{2}x8-\frac{1}{4}x1-\frac{1}{2}x6+\frac{1}{4}x2-\frac{1}{4}x4+\frac{1}{4}x3\right)\eta^2+\left(-\frac{1}{4}v3x8+\frac{1}{4}v3x6\right)\zeta \\
& +\left(\frac{1}{2}x3+x5-x7+\frac{1}{2}x4-\frac{1}{2}x2-\frac{1}{2}x1\right)\eta\,\xi \\
& +\left(-\frac{1}{4}v3x2+\frac{1}{4}v3x4+\frac{1}{2}v3x5+\frac{1}{4}v3x3-\frac{1}{2}v3x7-\frac{1}{4}v3x1\right)\xi\,\eta\,\zeta \\
& +\left(-\frac{1}{8}v3x4+\frac{1}{8}v3x3+\frac{1}{8}v3x2-\frac{1}{4}v3x6-\frac{1}{8}v3x1+\frac{1}{4}v3x8\right)\eta^2\,\zeta
\end{aligned}
$$

```
> j12:=collect(expand(diff(y,xi)),[xi,eta,zeta],distribute
  d);
```

$$j12 := \frac{1}{2}y6 - \frac{1}{2}y8 + \left(-\frac{1}{8}v3y2 - \frac{1}{8}v3y4 + \frac{1}{8}v3y1 + \frac{1}{8}v3y3\right)\eta\,\zeta$$
$$+\left(-\frac{1}{2}v3y7 + \frac{1}{4}v3y2 + \frac{1}{4}v3y1 + \frac{1}{4}v3y3 - \frac{1}{2}v3y5 + \frac{1}{4}v3y4\right)\zeta\,\xi$$
$$+\left(\frac{1}{2}y2 + \frac{1}{2}y3 - y7 + \frac{1}{2}y1 + \frac{1}{2}y4 - y5\right)\xi + \left(\frac{1}{4}y3 - \frac{1}{4}y4 + \frac{1}{4}y1 - \frac{1}{4}y2\right)\eta$$
$$+\left(-\frac{1}{2}y6 - \frac{1}{4}y4 + \frac{1}{4}y3 - \frac{1}{4}y1 + \frac{1}{2}y8 + \frac{1}{4}y2\right)\eta^2 + \left(\frac{1}{4}v3y6 - \frac{1}{4}v3y8\right)\zeta$$
$$+\left(\frac{1}{2}y4 - y7 - \frac{1}{2}y1 + \frac{1}{2}y3 - \frac{1}{2}y2 + y5\right)\eta\,\xi$$
$$+\left(\frac{1}{4}v3y4 - \frac{1}{2}v3y7 + \frac{1}{4}v3y3 + \frac{1}{2}v3y5 - \frac{1}{4}v3y1 - \frac{1}{4}v3y2\right)\xi\,\eta\,\zeta$$
$$+\left(-\frac{1}{8}v3y1 - \frac{1}{8}v3y4 + \frac{1}{8}v3y2 + \frac{1}{4}v3y8 + \frac{1}{8}v3y3 - \frac{1}{4}v3y6\right)\eta^2\,\zeta$$

```
> j13:=collect(expand(diff(z,xi)),[xi,eta,zeta],distribute
  d);
```

$$j13 := \frac{1}{2}z6 - \frac{1}{2}z8 + \left(-\frac{1}{8}v3z4 - \frac{1}{8}v3z2 + \frac{1}{8}v3z3 + \frac{1}{8}v3z1\right)\eta\,\zeta$$
$$+\left(-\frac{1}{2}v3z5 + \frac{1}{4}v3z4 + \frac{1}{4}v3z1 + \frac{1}{4}v3z2 - \frac{1}{2}v3z7 + \frac{1}{4}v3z3\right)\zeta\,\xi$$
$$+\left(\frac{1}{2}z3 + \frac{1}{2}z4 + \frac{1}{2}z1 + \frac{1}{2}z2 - z7 - z5\right)\xi + \left(\frac{1}{4}z1 - \frac{1}{4}z2 + \frac{1}{4}z3 - \frac{1}{4}z4\right)\eta$$
$$+\left(-\frac{1}{4}z4 + \frac{1}{4}z2 - \frac{1}{4}z1 + \frac{1}{4}z3 - \frac{1}{2}z6 + \frac{1}{2}z8\right)\eta^2 + \left(-\frac{1}{4}v3z8 + \frac{1}{4}v3z6\right)\zeta$$
$$+\left(-z7 - \frac{1}{2}z2 + z5 - \frac{1}{2}z1 + \frac{1}{2}z4 + \frac{1}{2}z3\right)\eta\,\xi$$
$$+\left(\frac{1}{2}v3z5 - \frac{1}{2}v3z7 - \frac{1}{4}v3z1 - \frac{1}{4}v3z2 + \frac{1}{4}v3z3 + \frac{1}{4}v3z4\right)\xi\,\eta\,\zeta$$
$$+\left(\frac{1}{8}v3z3 - \frac{1}{8}v3z1 - \frac{1}{4}v3z6 - \frac{1}{8}v3z4 + \frac{1}{4}v3z8 + \frac{1}{8}v3z2\right)\eta^2\,\zeta$$

```
> j21:=collect(expand(diff(x,eta)),[xi,eta,zeta],distribut
```

```
ed);
```

$$j21 := -\frac{1}{2}x5 + \frac{1}{2}x7 + \left(\frac{1}{4}v3x4 + \frac{1}{4}v3x2 + \frac{1}{4}v3x3 + \frac{1}{4}v3x1 - \frac{1}{2}v3x8 - \frac{1}{2}v3x6\right)\eta\,\zeta$$
$$+\left(\frac{1}{8}v3x3 - \frac{1}{8}v3x4 + \frac{1}{8}v3x1 - \frac{1}{8}v3x2\right)\zeta\,\xi + \left(-\frac{1}{4}x2 + \frac{1}{4}x1 + \frac{1}{4}x3 - \frac{1}{4}x4\right)\xi$$
$$+\left(\frac{1}{2}x2 + \frac{1}{2}x4 - x8 + \frac{1}{2}x1 + \frac{1}{2}x3 - x6\right)\eta$$
$$+\left(-\frac{1}{4}x2 - \frac{1}{2}x7 - \frac{1}{4}x1 + \frac{1}{2}x5 + \frac{1}{4}x4 + \frac{1}{4}x3\right)\xi^2 + \left(\frac{1}{4}v3x7 - \frac{1}{4}v3x5\right)\zeta$$
$$+\left(\frac{1}{2}x3 - \frac{1}{2}x1 + x8 - \frac{1}{2}x4 + \frac{1}{2}x2 - x6\right)\eta\,\xi$$
$$+\left(-\frac{1}{4}v3x4 + \frac{1}{2}v3x8 + \frac{1}{4}v3x2 - \frac{1}{2}v3x6 + \frac{1}{4}v3x3 - \frac{1}{4}v3x1\right)\xi\,\eta\,\zeta$$
$$+\left(\frac{1}{4}v3x5 - \frac{1}{8}v3x2 + \frac{1}{8}v3x3 + \frac{1}{8}v3x4 - \frac{1}{8}v3x1 - \frac{1}{4}v3x7\right)\xi^2\,\zeta$$

```
> j22:=collect(expand(diff(y,eta)),[xi,eta,zeta],distribut
  ed);
```

$$j22 := -\frac{1}{2}y5 + \frac{1}{2}y7 + \left(\frac{1}{4}v3y1 + \frac{1}{4}v3y4 + \frac{1}{4}v3y3 - \frac{1}{2}v3y8 - \frac{1}{2}v3y6 + \frac{1}{4}v3y2\right)\eta\,\zeta$$
$$+\left(-\frac{1}{8}v3y2 - \frac{1}{8}v3y4 + \frac{1}{8}v3y1 + \frac{1}{8}v3y3\right)\zeta\,\xi + \left(\frac{1}{4}y3 - \frac{1}{4}y4 + \frac{1}{4}y1 - \frac{1}{4}y2\right)\xi$$
$$+\left(-y8 + \frac{1}{2}y1 + \frac{1}{2}y4 + \frac{1}{2}y3 + \frac{1}{2}y2 - y6\right)\eta$$
$$+\left(-\frac{1}{2}y7 - \frac{1}{4}y1 - \frac{1}{4}y2 + \frac{1}{2}y5 + \frac{1}{4}y3 + \frac{1}{4}y4\right)\xi^2 + \left(-\frac{1}{4}v3y5 + \frac{1}{4}v3y7\right)\zeta$$
$$+\left(\frac{1}{2}y3 - \frac{1}{2}y4 + \frac{1}{2}y2 + y8 - \frac{1}{2}y1 - y6\right)\eta\,\xi$$
$$+\left(-\frac{1}{4}v3y1 - \frac{1}{4}v3y4 + \frac{1}{4}v3y2 + \frac{1}{2}v3y8 - \frac{1}{2}v3y6 + \frac{1}{4}v3y3\right)\xi\,\eta\,\zeta$$
$$+\left(\frac{1}{8}v3y4 + \frac{1}{4}v3y5 - \frac{1}{8}v3y1 + \frac{1}{8}v3y3 - \frac{1}{8}v3y2 - \frac{1}{4}v3y7\right)\xi^2\,\zeta$$

```
> j23:=collect(expand(diff(z,eta)),[xi,eta,zeta],distribut
  ed);
```

$$j23 := -\frac{1}{2}z5 + \frac{1}{2}z7 + \left(\frac{1}{4}v3z3 - \frac{1}{2}v3z8 - \frac{1}{2}v3z6 + \frac{1}{4}v3z4 + \frac{1}{4}v3z1 + \frac{1}{4}v3z2\right)\eta\,\zeta$$
$$+\left(-\frac{1}{8}v3z4 - \frac{1}{8}v3z2 + \frac{1}{8}v3z3 + \frac{1}{8}v3z1\right)\zeta\,\xi + \left(\frac{1}{4}z1 - \frac{1}{4}z2 + \frac{1}{4}z3 - \frac{1}{4}z4\right)\xi$$
$$+\left(\frac{1}{2}z4 + \frac{1}{2}z2 - z8 - z6 + \frac{1}{2}z3 + \frac{1}{2}z1\right)\eta$$
$$+\left(\frac{1}{4}z4 + \frac{1}{4}z3 - \frac{1}{4}z1 - \frac{1}{4}z2 - \frac{1}{2}z7 + \frac{1}{2}z5\right)\xi^2 + \left(\frac{1}{4}v3z7 - \frac{1}{4}v3z5\right)\zeta$$
$$+\left(\frac{1}{2}z3 - z6 - \frac{1}{2}z1 - \frac{1}{2}z4 + z8 + \frac{1}{2}z2\right)\eta\,\xi$$
$$+\left(-\frac{1}{2}v3z6 + \frac{1}{4}v3z3 + \frac{1}{2}v3z8 - \frac{1}{4}v3z1 + \frac{1}{4}v3z2 - \frac{1}{4}v3z4\right)\xi\,\eta\,\zeta$$
$$+\left(\frac{1}{8}v3z3 - \frac{1}{8}v3z2 + \frac{1}{8}v3z4 - \frac{1}{8}v3z1 + \frac{1}{4}v3z5 - \frac{1}{4}v3z7\right)\xi^2\,\zeta$$

```
> j31:=collect(expand(diff(x,zeta)),[xi,eta,zeta],distribu
  ted);
```

$$j31 := -\frac{1}{8}v3x4 - \frac{1}{8}v3x1 - \frac{1}{8}v3x2 - \frac{1}{8}v3x3 + \frac{1}{4}v3x8 + \frac{1}{4}v3x5 + \frac{1}{4}v3x6 + \frac{1}{4}v3x7$$
$$+\left(-\frac{1}{4}v3x8 + \frac{1}{4}v3x6\right)\xi + \left(\frac{1}{4}v3x7 - \frac{1}{4}v3x5\right)\eta$$
$$+\left(\frac{1}{8}v3x2 + \frac{1}{8}v3x1 - \frac{1}{4}v3x5 + \frac{1}{8}v3x4 - \frac{1}{4}v3x7 + \frac{1}{8}v3x3\right)\xi^2$$
$$+\left(\frac{1}{8}v3x1 - \frac{1}{4}v3x6 - \frac{1}{4}v3x8 + \frac{1}{8}v3x4 + \frac{1}{8}v3x2 + \frac{1}{8}v3x3\right)\eta^2$$
$$+\left(\frac{1}{8}v3x3 - \frac{1}{8}v3x4 + \frac{1}{8}v3x1 - \frac{1}{8}v3x2\right)\eta\,\xi$$
$$+\left(\frac{1}{4}v3x5 - \frac{1}{8}v3x2 + \frac{1}{8}v3x3 + \frac{1}{8}v3x4 - \frac{1}{8}v3x1 - \frac{1}{4}v3x7\right)\eta\,\xi^2$$
$$+\left(-\frac{1}{8}v3x4 + \frac{1}{8}v3x3 + \frac{1}{8}v3x2 - \frac{1}{4}v3x6 - \frac{1}{8}v3x1 + \frac{1}{4}v3x8\right)\eta^2\,\xi$$

```
> j32:=collect(expand(diff(y,zeta)),[xi,eta,zeta],distribu
  ted);
```

$$j32 := \frac{1}{4} v3y7 + \frac{1}{4} v3y8 - \frac{1}{8} v3y4 - \frac{1}{8} v3y1 - \frac{1}{8} v3y2 - \frac{1}{8} v3y3 + \frac{1}{4} v3y5 + \frac{1}{4} v3y6$$
$$+ \left(\frac{1}{4} v3y6 - \frac{1}{4} v3y8 \right) \xi + \left(-\frac{1}{4} v3y5 + \frac{1}{4} v3y7 \right) \eta$$
$$+ \left(-\frac{1}{4} v3y5 + \frac{1}{8} v3y2 + \frac{1}{8} v3y4 + \frac{1}{8} v3y1 - \frac{1}{4} v3y7 + \frac{1}{8} v3y3 \right) \xi^2$$
$$+ \left(\frac{1}{8} v3y1 - \frac{1}{4} v3y8 + \frac{1}{8} v3y3 - \frac{1}{4} v3y6 + \frac{1}{8} v3y4 + \frac{1}{8} v3y2 \right) \eta^2$$
$$+ \left(-\frac{1}{8} v3y2 - \frac{1}{8} v3y4 + \frac{1}{8} v3y1 + \frac{1}{8} v3y3 \right) \eta \xi$$
$$+ \left(\frac{1}{8} v3y4 + \frac{1}{4} v3y5 - \frac{1}{8} v3y1 + \frac{1}{8} v3y3 - \frac{1}{8} v3y2 - \frac{1}{4} v3y7 \right) \eta \xi^2$$
$$+ \left(-\frac{1}{8} v3y1 - \frac{1}{8} v3y4 + \frac{1}{8} v3y2 + \frac{1}{4} v3y8 + \frac{1}{8} v3y3 - \frac{1}{4} v3y6 \right) \eta^2 \xi$$

```
> j33:=collect(expand(diff(z,zeta)),[xi,eta,zeta],distribu
  ted);
```

$$j33 := \frac{1}{4} v3z7 + \frac{1}{4} v3z8 - \frac{1}{8} v3z4 - \frac{1}{8} v3z1 - \frac{1}{8} v3z2 - \frac{1}{8} v3z3 + \frac{1}{4} v3z5 + \frac{1}{4} v3z6$$
$$+ \left(-\frac{1}{4} v3z8 + \frac{1}{4} v3z6 \right) \xi + \left(\frac{1}{4} v3z7 - \frac{1}{4} v3z5 \right) \eta$$
$$+ \left(\frac{1}{8} v3z2 - \frac{1}{4} v3z5 + \frac{1}{8} v3z1 + \frac{1}{8} v3z3 - \frac{1}{4} v3z7 + \frac{1}{8} v3z4 \right) \xi^2$$
$$+ \left(\frac{1}{8} v3z3 + \frac{1}{8} v3z4 + \frac{1}{8} v3z2 - \frac{1}{4} v3z6 + \frac{1}{8} v3z1 - \frac{1}{4} v3z8 \right) \eta^2$$
$$+ \left(-\frac{1}{8} v3z4 - \frac{1}{8} v3z2 + \frac{1}{8} v3z3 + \frac{1}{8} v3z1 \right) \eta \xi$$
$$+ \left(\frac{1}{8} v3z3 - \frac{1}{8} v3z2 + \frac{1}{8} v3z4 - \frac{1}{8} v3z1 + \frac{1}{4} v3z5 - \frac{1}{4} v3z7 \right) \eta \xi^2$$
$$+ \left(\frac{1}{8} v3z3 - \frac{1}{8} v3z1 - \frac{1}{4} v3z6 - \frac{1}{8} v3z4 + \frac{1}{4} v3z8 + \frac{1}{8} v3z2 \right) \eta^2 \xi$$

```
>
```