HIGHER-ORDER SPECTRAL/*HP* FINITE ELEMENT TECHNOLOGY FOR STRUCTURES AND FLUID FLOWS

A Dissertation

by

VENKAT PRADEEP VALLALA

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2013

Major Subject: Mechanical Engineering

ABSTRACT

This study deals with the use of high-order spectral/*hp* approximation functions in the finite element models of various nonlinear boundary-value and initial-value problems arising in the fields of structural mechanics and flows of viscous incompressible fluids. For many of these classes of problems, the high-order (typically, polynomial order p greater than or equal to 4) spectral/*hp* finite element technology offers many computational advantages over traditional low-order (i.e., $p < 3$) finite elements. For instance, higher-order spectral/*hp* finite element procedures allow us to develop robust structural elements for beams, plates, and shells in a purely displacement-based setting, which avoid all forms of *numerical locking*. The higher-order spectral/*hp* basis functions avoid the *interpolation error* in the numerical schemes, thereby making them accurate and stable. Furthermore, for fluid flows, when combined with least-squares variational principles, such technology allows us to develop efficient finite element models, that always yield a symmetric positive-definite (SPD) coefficient matrix, and thereby robust direct or iterative solvers can be used. The least-squares formulation avoids *ad-hoc* stabilization methods employed with traditional low-order weak-form Galerkin formulations. Also, the use of spectral/*hp* finite element technology results in a better conservation of physical quantities (e.g., dilatation, volume, and mass) and stable evolution of variables with time in the case of unsteady flows. The present study uses spectral/*hp* approximations in the (1) weak-form Galerkin finite element models of viscoelastic beams, (2) weak-form Galerkin displacement finite element models of shear-deformable elastic shell structures under thermal and mechanical loads, and (3) least-squares formulations for the Navier-Stokes equations governing flows of viscous incompressible fluids. Numerical simulations using the developed technology of several non-trivial benchmark problems are presented to illustrate the robustness of the higher-order spectral/hp based finite element technology.

DEDICATION

To modern temples - Texas A&M University and Osmania University

ACKNOWLEDGEMENTS

I would like to thank my teacher and advisor, Dr. J. N. Reddy, for his support and guidance during the course of my M.S. and Ph.D. studies at Texas A&M University. It has been a great pleasure taking courses and working with Professor Reddy, a dedicated and passionate teacher and researcher in the field of applied and computational mechanics.

My sincere thanks to Dr. Anastasia Muliana, Dr. Steve Suh, and Dr. Ramesh Talreja for serving on my Ph.D. Committee and for their advice and co-operation. I am grateful to Dr. Anastasia Muliana and Dr. Steve Suh for their wonderful courses on Elasticity and Advanced Computer Aided Engineering. My special thanks to Dr. Harry Hogan for graciously accepting to substitute for Dr. Steve Suh for the final defense exam.

I am fortunate to work as a database developer for my supervisor, Mr. John Roths, Associate Research Scientist and IT-Manager at the Department of Veterinary Pathobiology, Texas A&M University. A self-trained IT professional, every moment of my work with him, has been a great learning experience. I am thankful to him for his flexibility and co-operation during my course and research deadlines.

I am also thankful to all my seniors and colleagues at Advanced Computational Mechanics Laboratory (ACML), especially Dr. Gregory Payette, Dr. Roman Arciniega, Mr. Jinseok Kim, Dr. Vinu Unnikrishnan and Dr. Ginu Unnikrishnan - each for their friendship, time and help with research, courses and many other things. During my Ph.D. studies at Texas A&M University, I was supported, at different times, by the Qatar National Research Fund through Grant NPRP09-1183-2-461 and the Air Force Office of Scientific Research through MURI Grant FA-9550-09-1-0686. I would also like to acknowledge the support, training, and software resources from the Texas A&M Supercomputing Facility, which have greatly enhanced and contributed to my research work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Background

Over the last few decades, the advances made in computer hardware and mathematical models of physical phenomena, the finite element method (FEM) has evolved as a versatile computational tool for solving complex engineering problems. At present, FEM is widely used as a premiere discretization procedure for the numerical simulation of structural mechanics problems. However, the majority of commercial finite element software in use today is dominated by lower-order weak-form Galerkin finite element technology and its variants. Although the weak-form Galerkin procedure results in an *ideal* mathematical framework for the numerical analysis of structures, the use of lower-order finite element technology often necessitates *ad-hoc* techniques to *stabilize* or *fix* deficiencies like various types of *locking* associated with the resulting discrete numerical schemes [84, 85, 88, 83]. This is especially true in the finite element modeling of structural components, such as beams, plates, and shells, where lower-order finite elements often require the use of *reduced* integration techniques that inevitably require hour-glass control [11]. As a result, reliable, general purpose, lower-order finite element technology for structural components still remains an open area in the numerical discretization of structures.

Although the finite element method has become the dominant method of choice in the numerical analysis of practical engineering structures, it is yet to receive similar widespread acceptance in the field of computational fluid dynamics. In the field of fluid mechanics, much of the success and breakthroughs in the numerical discretization of the the Navier–Stokes equations governing flows of viscous incompressible fluids have come in the context of lower-order finite difference (FD) and finite volume (FV) technology. It is well known that the finite element procedures offer many advantages over finite difference and finite volume methods. In particular, the finite element method can routinely deal with complex practical geometries, material properties, boundary conditions, and possesses a rich mathematical foundation [90]. As a result, there has been a renewed interest in recent years in developing efficient finite element models of the Navier–Stokes equations.

The majority of finite element models for fluids are based on the weak-form Galerkin procedure, although the underlying weak form does not ideal variational setting. Thus, it is well-known that the application of weak-form Galerkin procedure to fluid flows can result in a non-optimal setting for a given finite element discretization [14, 88]. For instance, the use of the weak-form Galerkin formulation of the Navier–Stokes equations expressed in terms of velocities and pressure must satisfy the restrictive discrete inf-sup or Ladyzhenskaya-Babuska-Brezzi (LBB) condition [17] in selecting approximation spaces for the velocity and pressure fields; this effectively precludes the use of equal degree of lower-order approximations of the velocity and pressure fields. Even when the LBB condition is satisfied, the finite element solution may still be plagued with spurious oscillations or wiggles in convection dominated flows (i.e., for high Reynolds number flows) and conservation of various physical quantities like dilatation, volume, mass etc. may be poor. Stabilized weak-form Galerkin finite element models, such as the SUPG [41, 18], penalty [82], and Galerkin least-squares [42] have received considerable attention over the last few decades and have greatly improved the associated finite element models. Unfortunately, the success of these methods is often intertwined with *ad-hoc* parameters that must be fine-tuned for a given flow situation. In addition, they do not result in a symmetric positive-definite (SPD) coefficient matrix. As with the structures, for fluid flows, there exists no reliable, general purpose stabilization-free, lower-order finite element technology.

1.2 Motivation for the research

As discussed in the previous section, majority of previous works concerned with efficient finite element models for structures and fluids, predominantly lower-order (i.e., linear and quadratic) finite element approximations of the field variables have been employed through the use of weak-form Galerkin formulations. As stated earlier, finite element models based on lower-order approximations are often plagued with many issues, both in structures and fluid flows, that require the use of ad-hoc approaches with side effects. The aim of this research is to develop higher-order finite element technology that uses higher-order spectral/*hp* approximations of the field variables for problems of structural mechanics and fluid mechanics, and bring out the benefits of the least-squares formulations in the finite element analysis of fluid flow problems.

Variational methods (i.e. methods based on the existence of a functional whose extremum is equivalent to the weak for of the governing equations) are considered to produce the "best" approximation to the exact solution of the equations being solved [87]. For most structural mechanics problems, such a variational setting is possible; that is, the weak-form Galerkin formulation for the construction of finite element models is . The issue of *numerical locking* can be easily alleviated by using higher-order spectral/*hp* basis functions without resorting to any ad-hoc approach. Therefore, we use weak-form Galerkin formulation for all the structural mechanics problems in this research work.

A variational setting based on the weak formulation does not exist for the Navier–Stokes equations expressed in terms of pressure and velocities. Consequently, most finite element models of the Navier–Stokes equations based on the weak-form Galerkin procedure do not guarantee minimization of the error in the approximation of the solution or the differential equation. The least-squares method offer an appealing alternative to the commonly used weak-form Galerkin procedure for fluids (see for example Refs. [47, 44, 48, 46, 76, 78, 80]). They not only possess the best approximation properties but also avoid the restrictive compatibility requirements, that is, the LBB condition. Also, they always result in a symmetric positive-definite (SPD) coefficient matrix, so that robust direct and iterative solvers can be employed. However, previous applications of the least-squares method have often been plagued with *spurious solution oscillations* [75] and poor conservation of physical quantities (like dilatation, mass, volume) [26], primarily due to the use of lower-order approximations. The least-squares formulation, when combined with high-order spectral/*hp* approximation functions, results in a better conservation of the physical quantities, which in-turn, reduces the instability and *spurious oscillations* of solution variables with time. The instability can be further reduced using a iterative penalization strategy [34, 75], as done in this study.

1.3 Scope of the research

The research work reported in this thesis began at Texas A&M University in the Summer of 2009, and it is mainly focused on developing a reliable, general purpose, stabilization-free finite element technology for structures and fluid flows using higher-order spectral/*hp* basis functions. The thesis is organized as follows:

- In Section 2, some of the practical and numerical issues involved in developing a high-order finite element framework using spectral/*hp* basis functions are presented. We show ways to derive higher-order spectral/*hp* basis functions and higher-order numerical quadratures to evaluate integrals of such functions. We generate higher-order finite element grids, including body-fitted grids for simple two-dimensional curved boundaries. We show ways to implement schur complement method and also derive a line (or surface) integration method to handle general traction (flux, outflow) type of boundary conditions. Finally, higher-order finite element strategies and solution methods using direct and iterative solvers are discussed. The results presented in this section are taken from manuscripts that are currently in review for publications in journals [121, 117].

- In Section 3, we develop weak-form Galerkin finite element models for viscoelastic beams using high-order spectral/*hp* finite element models. The material of the beam is considered to be linearly viscoelastic while the beam may undergo *von Kármán* nonlinear geometric deformations. The beam is modeled using a higher-order beam theory (HBT) that admits C^0 continuous interpolation for all dependent variables of the theory. The focus of this work is more on the evaluation of the performance of the high-order spectral/*hp* approximations with respect to issues of *numerical locking*, and not much on the theory or the mechanics of viscoelasticity. We use the linear viscoelastic constitutive relations from the works of [50, 22, 72], who used lower-order beam theories (e.g., Euler-Bernoulli and Timoshenko beam theories or its variants) and lower-order finite elements. This work has resulted in two peer-reviewed journal publications [118, 120].

- In Section 4, we describe higher-order spectral/*hp* finite element procedures that allow us to develop robust beam, plate, and shell elements in a purely displacement based setting and which avoid all forms of *numerical locking*. A weak-form Galerkin finite element model is constructed based on an improved fist-order shear deformation theory (FSDT), that allows the use of fully three-dimensional constitutive equations in the numerical implementation. Also, the formulation allows the use of randomly skewed and curved quadrilateral elements, a highlight of the present study, which will be useful to represent complicated shell geometries. The formulation is suitable for the analysis of geometrically nonlinear response of elastic, isotropic, and functionally graded shell structures subjected to mechanical and thermal loads. The results presented in this section are included in a manuscript under preparation for a journal [119].

- In Section 5, we develop a stress-based least-squares finite element models of the Navier–Stokes equations governing flows of viscous incompressible fluids using higher-order spectral/*hp* basis functions. An iterative penalization approach is used to improve conservation of physical quantities, and it resulted in a smooth evolution of primary solution variables. Numerical solutions of several non-trivial benchmark problems are presented to illustrate the accuracy and robustness of the developed finite element technology. The work on the fluid flows has culminated in to peer-reviewed journal publications that are either published, in review, or currently in preparation [122, 121, 117].

- In Section 6, some conclusions on the research reported in the thesis are presented and recommendations for future research are made.

2. ART OF HIGHER-ORDER SPECTRAL/*HP* FINITE ELEMENT METHODS: MESH GENERATION, BOUNDARY CONDITIONS, SCHUR COMPLEMENT METHOD AND LINEAR SOLVERS

In this section, we address some practical and numerical issues that are critical for a successful implementation of higher-order finite elements for large systems. To begin with, a brief introduction on the advantages and the necessity to go for higher-order finite element approximations is presented. Then, we review spectral interpolation functions and describe ways to implement them in a finite element program. Also, higher-order numerical quadrature rules and recursive/iterative schemes to calculate the integrals are derived. We also show ways to generate higher-order finite element grids for one-dimension and two-dimension domains including body-fitted meshes for curved boundaries. The algorithms needed to generate the necessary data structures to efficiently apply the boundary conditions for large domains are also presented. A line (or surface) integration method to handle general traction (flux, outflow) type of boundary conditions is developed. We show the advantages of schur complement (or static node condensation) method for higher-order finite elements and discuss ways to implement it. Then, we present some parallel solution strategies for use with direct and iterative solvers. Finally, a benchmark problem is tested to demonstrate the robustness of all strategies presented in this section.

2.1 Introduction

Most of the traditional finite element implementations are typically characterized by the use of lower-order (i.e., linear or quadratic) elements. High-order finite elements offer many theoretical and numerical advantages compared to lower-order finite elements. For structures like beams, plates and shells [118, 120, 119], it is shown that, for weak-form Galerkin finite element formulations, the issues of locking can be alleviated using higher-order elements. There is no need to use *reduced* or *selective* integration techniques and equal-order interpolations can be used for all dependent variables. In fluid flows, use of higher-order elements with the least-squares method [87] leads to better mass conservation and stability of field variables with time [75]. Also, the least-squares method results in an un-constrained minimization problem, as opposed to a saddle-point problem with the weak-form Galerkin method, so the approximation function spaces for velocity and pressure are not constrained to satisfy the LBB condition [121, 122, 5, 16] and thus equal-order interpolations can be used for all variables of the mathematical model. The higher-order spectral/*hp* elements also result in spectral accuracy (i.e., exponential convergence with increased order of approximation) for smooth (C^∞) solutions. Higher-order spectral/*hp* elements are suited for problems where high resolution in solution is required. For more on the mathematical treatment of error estimates, convergence, and stability of the higher-order finite elements, we refer to [104].

Although there are few published works in the literature that use higher-order finite elements, most of them do not talk about the practical issues encountered to get those results. Developing an efficient higher-order finite element solver is not a trivial task as no algorithm can be hard-wired in the code as with the lower-order elements. Also, generating higher-order shape functions, numerical quadrature, mesh and boundary conditions data structures is not a trivial task especially for complicated domains. It demands a more scientific and holistic approach to develop necessary algorithms and data structures to characterize the mesh and to apply the boundary conditions. Not many commercial finite element softwares implement higher-order elements and there are no open-source mesh generators which further complicates the problem. In this paper, we present some ideas, algorithms, datahandling techniques that we developed during the part of this research work and address certain practical issues that are needed for a successful implementation of higher-order finite element codes. It must be mentioned that we don't implement any automatic mesh adaptive strategies in finite element code and all the higher-order elements used are of conforming type i.e. no non-matching (*hanging*) points/nodes in the mesh. We have successfully developed higher-order finite element solvers with uniform, graded, skewed and arbitrarily curved quadrilateral elements. With few changes one can easily extend it to generate higher-order two-dimensional triangular elements and fully three-dimensional elements. The present work appeals to people who have experience with lower-order finite element programming and intend to develop their own programs using higher-order elements.

2.2 Higher-order nodal base functions and quadrature

2.2.1 Spectral nodal base functions

First, we present one-dimension spectral (also called Lobatto) nodal base functions and then, derive them in higher-dimensions. Consider the well-known p'th order Lagrange interpolation functions $l_{p,i}$ in one-dimension

$$l_{p,i} = \psi_i(\xi_j) = \prod_{\substack{j=1 \\ j \neq i}}^{p+1} \frac{(\xi - \xi_i)}{(\xi_i - \xi_j)} \tag{2.1}$$

where ξ_j are the evenly spaced nodal points in the canonical interval $-1 \leqslant \xi \leqslant +1$. The above Lagrange polynomials with equally or evenly spaced nodal points are prone to oscillations (for $p > 4$), leading to a divergence known as *Runge's Phenomenon.* This behavior tends to grow with the increase in the number of nodal points ξ_j. The *Runge effect* introduces *interpolation error* into the numerical scheme there-by making it less accurate and un-stable. In the next few paragraphs we present a mathematical definition of the *interpolation error*.

In one-dimension space, we know that in a general finite element procedure, the basic idea is to divide the domain into smaller elements defined on a typical interval $a \leq x \leq b$ of an e'th element and approximate the solution function $u(x)$, as-closely-as desired by a p'th-order polynomial function $P_p(x)$. The polynomial function can be expressed by specified coefficients or *a-priori* unknown nodal values u_i and interpolation functions $\psi_i(x)$ as

$$u(x) \approx P_p(x) = \sum_{i=1}^{n} u_i \psi_i(x) \tag{2.2}$$

where $n = (p+1)$ is the number of nodes in the e'th element. The interpolation error function is defined as the difference between exact solution and approximate polynomial function

$$\varepsilon(x) = u(x) - P_p(x) \tag{2.3}$$

Note $\varepsilon(x_i) = 0$, since $u(x_i) = P_p(x_i)$ at x_i the $-i'$th node. The aim is to achieve best possible accuracy for a given order of the interpolation function and also $\varepsilon(x)$ should tend to zero in the order p tends to infinity.

$$\lim_{p \to \infty} \left(\max_{a \leq x \leq b} |u(x) - P_p(x)| \right) = 0 \tag{2.4}$$

The evenly spaced nodes of the traditional Lagrange interpolation functions, perform well at lower-orders of the interpolation functions (typically $p \leq 3$). However at higher-orders, the evenly spaced interpolation functions result in oscillations towards the edges of the canonical interval. These oscillations are referred to as *Runge's Phenomenon.* This will introduce interpolation error into the numerical scheme there-by making it less accurate and un-stable. In the canonical interval $-1 \leqslant \xi \leqslant +1$, it can be easily verified (with a simple one-dimension function) that the evenly spaced nodal points of the traditional Lagrange shape functions leads to the divergence of the above equation i.e.

$$\lim_{p\to\infty}\left(\max_{-1\leq\xi\leq+1}\left|u\left(\xi\right)-P_p\left(\xi\right)\right|\right)=\pm\infty \tag{2.5}$$

This problem can be eliminated by choosing un-evenly spaced nodal points of the spectral (also called Lobatto) interpolation functions. More details on Lobatto and Lagrange nodal base functions can be found in [104, 51, 121]. To get the Lobatto interpolation functions and to find their zeros in the canonical interval $-1\leqslant\xi\leqslant+1$, the below relationship between the Lobatto polynomials $Lo\left(\xi\right)$ and the well-known Legendre polynomials $L\left(\xi\right)$ is used.

$$Lo_p\left(\xi\right)=L'_{p+1}\left(\xi\right) \tag{2.6}$$

The above equation allows the use of well established recurrence relations and other helpful features of the Legendre polynomials in deriving the expressions for the Lobatto interpolation functions. Using these recurrence relations and properties like "partition of unity", the final expression for Lobatto interpolation functions can be easily obtained by doing little math (see [104] for details)

$$\psi_i(\xi)=\frac{(\xi-1)(\xi+1)Lo_{p-1}(\xi)}{(p)(p+1)L_p(\xi_i)(\xi-\xi_i)}=\frac{(\xi-1)(\xi+1)L'_p(\xi)}{(p)(p+1)L_p(\xi_i)(\xi-\xi_i)} \tag{2.7}$$

The Eq. (2.7) simply represents Lagrange interpolation functions corresponding to $(p-1)$ intermediate Lobatto nodes ξ_i. Hence in practice the Lobatto nodal functions are treated as standard Lagrange nodal interpolation functions with un-evenly spaced nodes, given by the zeros of

$$(\xi - 1)(\xi + 1)L_p^{'}(\xi) = 0 \tag{2.8}$$

in the canonical interval $-1 \leqslant \xi \leqslant +1$. These set of points $\{\xi_i\}_{i=1}^{i=p+1}$ are commonly referred as Gauss-Lobatto-Legendre (GLL) nodes. To find the GLL nodes from Eq. (2.8), the higher-order Legendre polynomials have to be solved for any arbitrary order. For this, the following three-point recurrence scheme of the orthogonal Legendre polynomials can be used

$$L_{p+1}\left(\xi\right) = \left[\left(2p + 1\right)\xi L_p\left(\xi\right) - pL_{p-1}\left(\xi\right)\right]/(p + 1) \tag{2.9}$$

The first derivative of the orthogonal Legendre polynomials satisfy the following recurrence relation

$$\frac{\left(\xi - 1\right)\left(\xi + 1\right)}{p}L_p^{'}\left(\xi\right) = \xi L_p\left(\xi\right) - L_{p-1}\left(\xi\right) \tag{2.10}$$

The above recurrence schemes can be easily solved by noting that the first two Legendre polynomials are $L_0\left(\xi\right) = 1$ and $L_1\left(\xi\right) = \xi$. For $p \leqslant 2$ the GLL nodes are evenly spaced and coincide with the Lagrange nodes, for $p > 2$ the GLL nodes are biased towards the ends of the canonical interval. So in practice Lobatto interpolation functions are simply treated as Lagrange interpolation functions with un-evenly spaced GLL nodes. Also in the finite element program, instead of using the complicated Eq. (2.7) of Lobatto interpolation functions, the simple formula of the classical Lagrange interpolation functions given by Eq. (2.1) is used.

Using the GLL nodes from Eq. (2.9), the one-dimension Lobatto interpolation functions are plotted in Fig. 2.1(b) for $p = 8$. For comparision, a plot of the evenly spaced one-dimension Lagrange interpolation functions is shown in Fig. 2.1(a). From Fig. 2.1(b), it is visibly evident that the GLL nodes are biased towards the ends of the interval. Also, it is evident that Lobatto interpolation functions reach a maximum value of 1 through out the canonical interval $-1 \leqslant \xi \leqslant +1$ and the *Runge's Phenomenon* is not observed. Hence the higher-order spectral/*hp* functions will reduce the *interpolation errors* in a numerical scheme and thus making it more stable. In two-dimensions, the GLL nodes will be biased towards the element sides as shown pictorially in Fig. 2.2 for polynomial orders of $p = 4$ and $p = 8$. Note, the nodes are numbered from left to right using a local numbering system that is followed in our finite element program. For higher-dimensions the Lobatto interpolation functions can be easily obtained from the tensor product of one-dimension basis functions, for example, in two-dimensions

$$\psi\left(\xi,\eta\right) = \psi\left(\xi\right)\psi\left(\eta\right) \text{ in } [\text{ - }1{,}1] \times [\text{ - }1{,}1] \tag{2.11}$$

where $\psi\left(\xi\right)$ and $\psi\left(\eta\right)$ are given by Eq. (2.1). The Fig. 2.3 shows the plot of Lagrange interpolation function and Lobatto interpolation function associated with the node at the center $(\xi = 0, \eta = 0)$ of the element for $p = 8$, i.e. $\psi_{41}\left(\xi,\eta\right)$ (see Fig. 2.2(b)). From the Fig. 2.3, it is visibly evident that both the Lagrange and spectral Lobbato interpolation functions reach a maximum value of 1 at the element centers but the evenly spaced Lagrange interpolation function over-shoots and goes beyond 1 at the corners. This is due to the inherent problem of *Runge's Phenomenon* with the evenly spaced Lagrange interpolation functions. In higher-dimensions too, these oscillations become more pronounced as the order of the polynomial is increased. So as in one-dimension case, the evenly spaced higher-order two-dimension Lagrange shape functions introduce interpolation errors in to a numerical scheme.

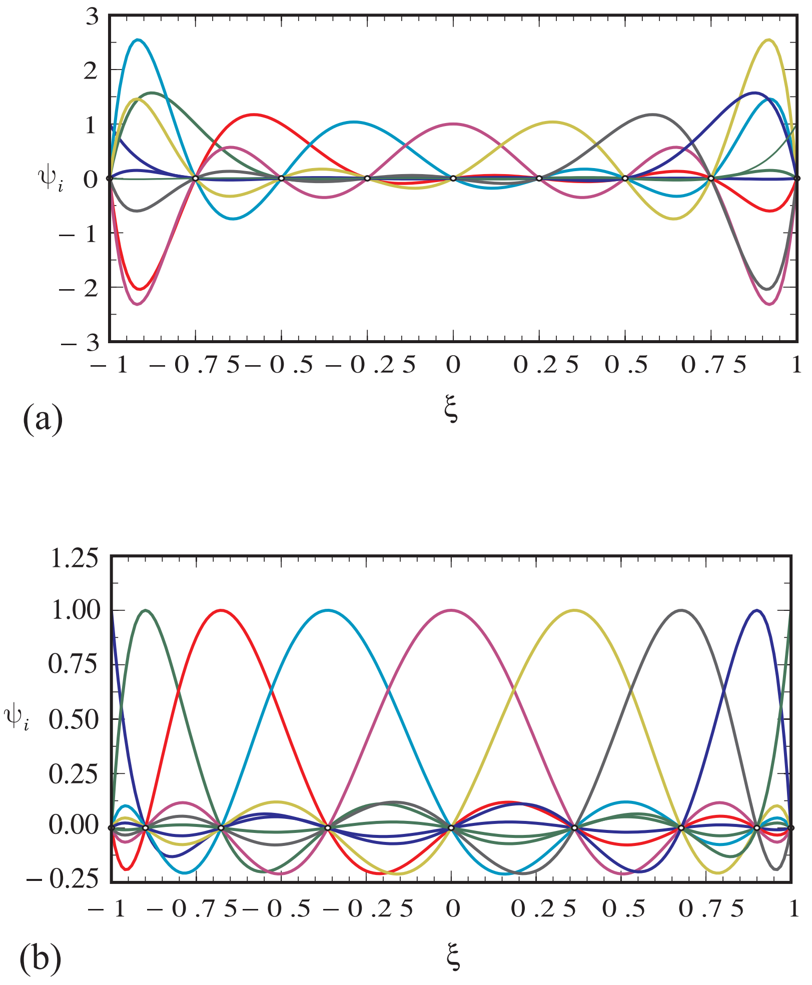


Figure 2.1: One-dimensional interpolation functions and nodal locations for a polynomial of order $p = 8$: (a) Equi-spaced and (b) Spectral (Lobatto).

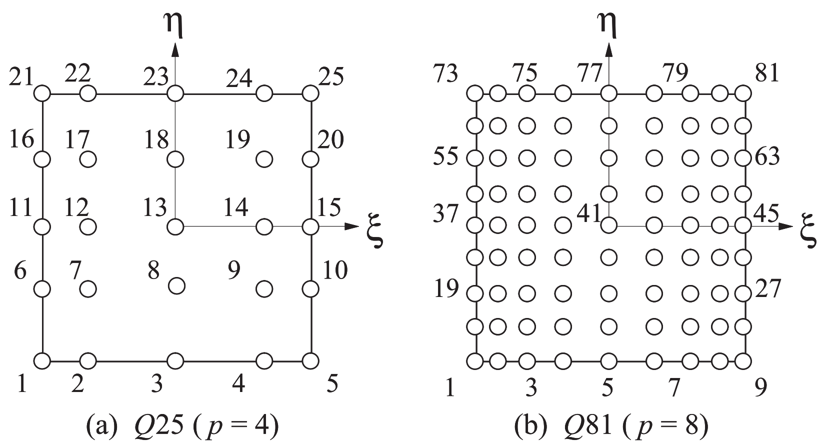


Figure 2.2: Two-dimension GLL node locations and local element numbers for a polynomial of orders: (a) $p = 4$ and (b) $p = 8$.

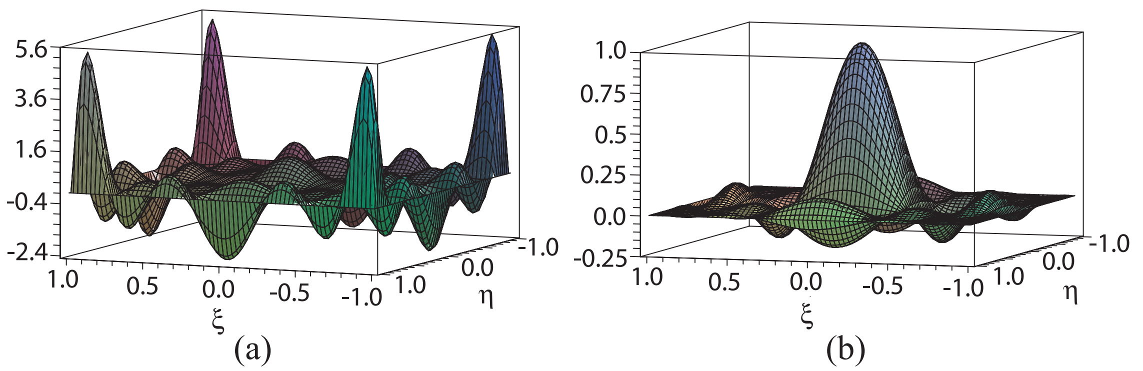


Figure 2.3: Two-dimension interpolation function $\psi_{41}(\xi, \eta)$ for polynomial of order $p = 8$: (a) Equi-spaced and (b) Spectral.

2.2.2 Numerical quadrature

There are different Gaussian integration quadratures corresponding to different sets of orthogonal polynomial basis functions, each with different Gaussian weighting functions and Gaussian points. For example, we have the Gauss–Chebyshev quadrature for Chebyshev functions, the Gauss–Lobatto quadrature for Lobatto functions, or the standard and most widely used Gauss–Legendre quadrature for Legendre functions etc. Since the nodal Lobatto functions are similar to evenly spaced Lagrange interpolations functions we use the standard Gauss–Legendre quadrature throughout this work. The higher-order elements need higher-order quadrature rules i.e. higher-order sets of Gauss–Legendre points and weights. Some standard mathematical handbooks [1] give these values, but they are often limited to lower-order quadratures and are not complete. In Section 4 on functionally graded shells, a quadrature rule of order $\approx$ 50 is employed to do the pre-integration through the thickness of the shell element. The Gauss–Legendre quadrature points $\{\xi_i\}_{i=1}^{p+1}$ are the zeros of Legendre polynomial of order 'p' in the canonical interval $-1 \leqslant \xi \leqslant +1$ and they can be obtained using a recurrence relations as explained previously. Once the quadrature points are known, the Gauss–Legendre quadrature weights can be obtained using

$$w_i = \frac{2}{(1-\xi_i^2)\left[L_p'(\xi_i)\right]^2} = \frac{2}{(1+p)^2}\frac{(1-\xi_i^2)}{\left[L_{p+1}(\xi_i)\right]^2} \tag{2.12}$$

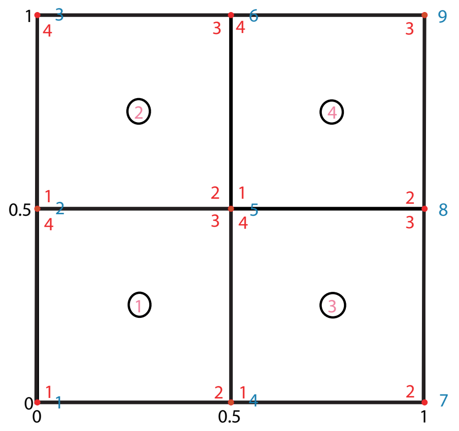
Throughout this work a quadrature rule of atleast $(p+1)$ is used in each direction. Also, all the integrands are evaluated using *full integration* rules without resorting to any *selective* or *reduced* integration techniques. The GLL nodes and the quadrature points and weights can be easily obtained by coding the above mentioned recurrence relationships in MATLAB or Maple. It is always safe to use a large precision for all of them, especially for GLL nodes, to prevent them from overlapping one towards the corners of the element.

2.3 Higher-order mesh generation

The generation of higher-order two-dimensions mesh is not so straight forward, especially, for non-rectangular domains or domains with simple curved boundaries. Also, the application of *essential* and *natural* type boundary conditions requires the development of suitable algorithms and data structures. Before discussing about them, we mention that all the higher-order mesh related data, like the global positions of the nodes, the connectivity matrix (which relates the local node numbers of each element to the unique global node numbers in the mesh), and the global *essential* and *natural* type of boundary conditions for each degree of freedom are specified in the input file to the finite element program. Although the boundary conditions are specified at global level in the input file to the finite element program, they are actually applied at element level (as opposed to the usual tradition of applying at the global level). Even though converting the global boundary conditions to element level require additional steps in the finite element program, there are some advantages which will be explained in the coming sections. For rectangular domains and shell structures this data (mesh attributes, node locations, connectivity matrix and global boundary conditions) is entirely generated using MATLAB, but for non-rectangular domains or domains with simple curves (like a quarter circular plate, plate with a circular hole) we combine the best of the commercial software programs like ABAQUS and MATLAB as explained in coming paragraphs.

To generate higher-order grids for non-rectangular domains, first a skeletal mesh with linear (polynomial of order $p = 1$) finite elements is designed in ABAQUS and then, additional nodes are inserted using a p-refinement MATLAB program. The advantage of ABAQUS is that, complicated geometries can be easily designed in its computer aided environment (CAE). The main aim to make a linear mesh is to fix the elements, their neighbors and the boundaries of the finite element domain. Once the linear model is drawn in CAE, the ABAQUS job is submitted to write all the mesh attributes like number of elements, node locations etc. to a text file. This information will greatly reduce the complexities involved in generating the higher-order grids. To understand the procedure in detail consider a simple mesh shown in Fig. 2.4(a). Although the higher-order mesh for this simple geometry can be directly generated, it is used to keep the illustration simple and understandable. The mesh has two linear elements (polynomial of order $p = 1$) across X-axis from [0, 1] and two in the Y-axis from [0, 1]. The element numbers are shown in the center of each element in small circles. The unique global node numbers are shown in Blue and the local element numbers are shown in Red. It is to be noted that ABAQUS uses an anti-clockwise local node numbering system. Note, different commercial software programmes (like ABAQUS) have different local numbering systems and they must be in-tune with the local node numbering system followed in the finite element program.

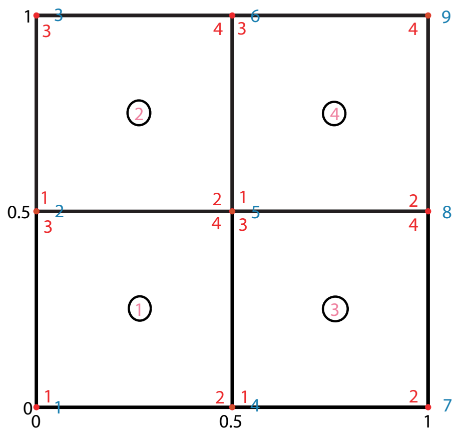
The data obtained from the ABAQUS job is given in Fig. 2.4(b). Here, ETYPE stands for element type which is one more than the polynomial order i.e. $(p+1)$. NE stands for number of elements in the mesh. ECON stands for element connectivity matrix; it relates local elemental node numbers to the unique global node numbers. In ECON $(i,\ j)$ matrix, the row-index 'i' represents the element number and the column-index 'j' represents the local element number. This matrix is very critical in many aspects of the finite element program. Note, each one of the components is qualified with 'abaq.', so that it can be used as a data structure without having



(a)

abaq.Node	abaq.X	abaq.Y	abaq.ETYPE	abaq.NE
1	0	0	2	4
2	0	0.5		
3	0	1		
4	0.5	0		
5	0.5	0.5		
6	0.5	1		
7	1	0		
8	1	0.5		
9	1	1		
abaq.ECON				
	1	2	3	4
1	1	4	5	2
2	2	5	6	3
3	4	7	8	5
4	5	8	9	6

(b)



(c)

mesh.Node	mesh.X	mesh.Y	mesh.ETYPE	mesh.NE
1	0	0	2	4
2	0	0.5		
3	0	1		
4	0.5	0		
5	0.5	0.5		
6	0.5	1		
7	1	0		
8	1	0.5		
9	1	1		
mesh.ECON				
	1	2	3	4
1	1	4	2	5
2	2	5	3	6
3	4	7	5	8
4	5	8	6	9

(d)

Figure 2.4: (a) ABAQUS linear mesh (b) ABAQUS linear mesh attributes (c) Modified linear mesh and (d) Modified linear mesh attributes

to handle the individual components. In our finite element program a different local node numbering system is followed. It is shown in the figure Fig. 2.4(c); observe that the local element numbering (in Red) is from left to right, as opposed to anti-clockwise in the ABAQUS mesh. To get the data for this system, all it takes is to swap the 3'rd column of 'abaq.ECON' with the 4'th column in Fig. 2.4(b). This new data is shown in Fig. 2.4(d); observe that in this figure everything remains the same as in Fig. 2.4(b) from ABAQUS, except the data in the above said columns is interchanged. Here, we qualify each one of them with 'mesh.' so that it can be imported directly to MATLAB and used as a data structure.

To apply boundary conditions one more data structure about the sides of the physical domain is required. It should hold the information about the elements that are on the boundary of the mesh and also the sides of each element on the boundary. To do this a local element side numbering system has to be followed. The Fig. 2.5 shows how the nodes and sides are numbered in our finite element program for a typical two-dimensions master element $[-1, +1] \times [-1, +1]$ with ETYPE $= 3$. It can be noted that side-1 is plotted in Red for $\xi = +1$ and side-2 is plotted in Green for $\eta = +1$, side-3 is plotted in Blue for $\xi = -1$ and side-4 is plotted in Black for $\eta = -1$.

Using the data structure 'mesh' from Fig. 2.4(d) and the notation from Fig. 2.5, the colored mesh of Fig. 2.4(c) is shown in Fig. 2.6(a). Note, since it quite obvious to plot the element sides in color using MATLAB, it is not explained. The 'sidesets' data structure in Fig. 2.6(b) can be easily generated from the boundary element side colors in Fig. 2.6(a). The sideset1, sideset2, sideset3 and sideset4 contain the information about the element numbers and the element sides for all the boundaries of the domain. For the present simple mesh, the 'sidesets' data structure can be created manually but for complicated domains it can be directly created in MATLAB. Here,

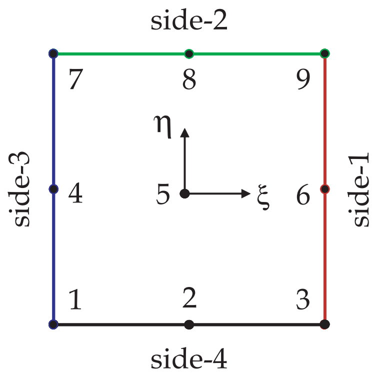
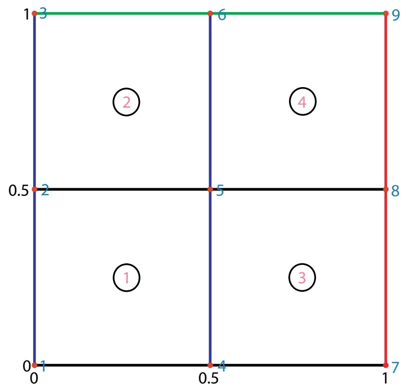


Figure 2.5: Elemental side numbering system and color scheme employed.

each one of it is qualified with 'sidesets.', so that it can be used as a data structure. Once the 'mesh' and 'sidesets' data structures are obtained, they are saved to a '.mat' file in MATLAB. This will serve as the input data to the *p*-refinement MATLAB program. This program basically inserts the spectral GLL nodes in to the linear mesh and generates the higher-order mesh using the transformation of the geometry given by Eq. (2.13). It also converts the 'sidesets' data structure to 'nodesets' that is used to apply the *essential* type of boundary conditions (see next section).

$$x = \sum_{i=1}^{n} x_i^e \psi_i^e \left(\xi, \eta\right), \; y = \sum_{i=1}^{n} y_i^e \psi_i^e \left(\xi, \eta\right) \tag{2.13}$$

The *p*-refinement program also needs information about the orientation of the neighboring elements. The Fig. 2.7(a) shows the possible combinations in which the 'side-1' of an element can be in contact with sides of a neighboring element. This indicates if the local co-ordinate systems of the two neighboring elements are oriented in the same direction or in the opposite direction. Similar combinations are possible for the other three sides of a element. This information can be compactly represented using a symmetric matrix, as in Fig. 2.7(b), where +1 represents same orientation



(a)

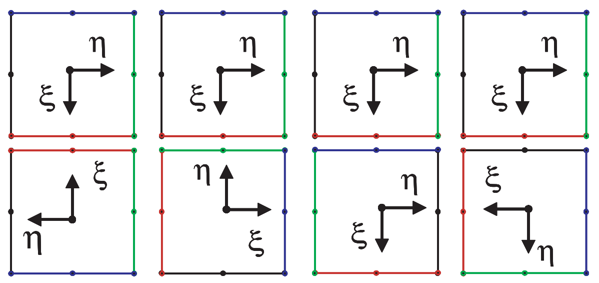
	Element No.	Side No.
sidesets.sideset1	3	1
	4	1
sidesets.sideset2	4	2
	2	2
sidesets.sideset3	2	3
	1	3
sidesets.sideset4	1	4
	3	4

(b)

Figure 2.6: (a) Mesh in color for a polynomial of order $p = 1$ and (b) Sideset info.

of the local co-ordinate systems and -1 represents the opposite orientation. Due to space limitations in this work, we only mention the steps involved in the p-refinement MATLAB program in Fig. 2.8. The p-refinement we results in a similar data as in Fig. 2.4(d), but for the refined mesh. It also generates the 'nodesets' data structure that is used to apply *essential* type of boundary conditions. The p-refined mesh generated using these steps is shown in Fig. 2.9 for $p = 2$. The p-refinement MATLAB code has the capability to automatically generate complete input files from $p = 2$ to $p = 20$. Using similar ideas a h-refinement MATLAB code is also developed, this will insert new elements into the mesh instead of nodes.

It must be mentioned that the p-refinement code inserts new GLL nodes on a straight line joining the two nodes of the initial linear mesh. When a part of the domain is curved, like a plate with a circular hole, then the higher-order refined mesh will have straight edges around the hole. To generate a body-fitted mesh around the circular arc, another strategy is developed. Consider a two-dimensional domain $[-15.5, +25.5] \times [-20.5, +20.5]$ with a circular hole of diameter 1 at the center. The skeletal linear mesh is generated in ABAQUS and the local node numbering system



(a)

$$\Xi = \begin{bmatrix} -1 & +1 & +1 & -1 \\ +1 & -1 & -1 & +1 \\ +1 & -1 & -1 & +1 \\ -1 & +1 & +1 & -1 \end{bmatrix}$$

(b)

Figure 2.7: (a) Possible combinations of side-1 and (b) Compact representation.

is changed as explained above. The colored mesh is plotted in MATLAB as shown Fig. 2.10(a). Here, the element and node numbers are suppressed for clarity. But they can be plotted as shown for the above cases. The 'sidesets' data, similar to the one in Fig. 2.6(b), can always be visually verified by the color of the element sides on the outer boundaries and on the surface of the hole.

The p-refined mesh around the cylinder is shown in Fig. 2.10(b) for $p = 2$ (nodes are shown as dots). It is clear that the mesh is not smooth around the cylinder. In some cases, like in the flow over cylinder problems, the surface needs to be smooth, otherwise it will affect the flow characteristics of the fluid. In such cases, the newly inserted GLL nodes have to be moved from the straight edges back on to the circular arc. This can be achieved by solving a two-dimensions isotropic, pseudo-elasticity problem with specified displacements as the boundary conditions on the straight edges of the hole and zero-displacements on all other boundaries of the domain. The pseudo-elasticity formulation is a standard procedure (see [11]) that is used in conjunction with the arbitrary Lagrangian Eulerian (ALE) formulation for the analysis of fluid-structure interaction (FSI) problems. To prevent excessive distortion of elements in the model the Young's modulus of the e'th finite element is specified as

(I) Loop over number of elements $i = 1$ .... mesh.NE

(a) Use the matrix from Fig. 7(b) and the connectivity matrix mesh.ECON, to extract the neighbor info of each element. It should have the neighbor element numbers, contacting sides and theorientation (+1 or -1) of local co-ordinate system.

(II) Initialize counter to track the global node numbers in the *p*-refined mesh *cntr* = 0

(III) Loop over number of elements $i = 1$ .... mesh.NE

(a) Use Eq. (8) to insert spectral GLL nodes. Increment the counter *cntr* for each node inserted. Write the new data structure mesh1.ECON, mesh1.X, mesh1.Y.

(b) Use the data from step (I) to check neighboring elements. If neighbors are already *p*-refined, then decrease the *cntr* by the suitable number of nodes on the contacting sides, if not, do nothing.

(c) From step (I), if the orientation of local co-ordinate system is +1 then transfer the global node numbers from the already *p*-refined elements directly to mesh1.ECON. If it is -1, transfer the global nodes after reversing.

(d) Using *cntr* from (b) and mesh1.ECON from (c), modify the global node numbers in mesh1.X and mesh1.Y.

(IV) Use mesh1.ECON from step (III) and extract the global nodes on each 'sidesets'. For example it will generate 'nodesets.nodeset1' from 'sidesets.sideset1', and it will have global node numbers of the *p*-refined mesh on that boundary.

(a) For *essential* type boundary conditions, these 'nodesets' can be directly assigned to the degree of freedom that is to be constrained on that boundary.

(b) For *natural* (like traction, flux, outflow) type boundary conditions, there is no need of 'nodesets' information, just the element number and the local side number are enough. Hence, we can directly use the 'sidesets' to apply these boundary conditions.

(V) Write the *p*-refined data and boundary conditions to create the input file for the finite element program.

Figure 2.8: Steps involved in p-refinement.

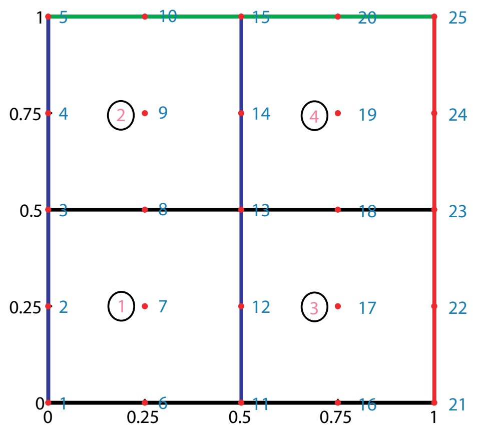


Figure 2.9: p-refined mesh for a polynomial of order $p = 2$ in color.

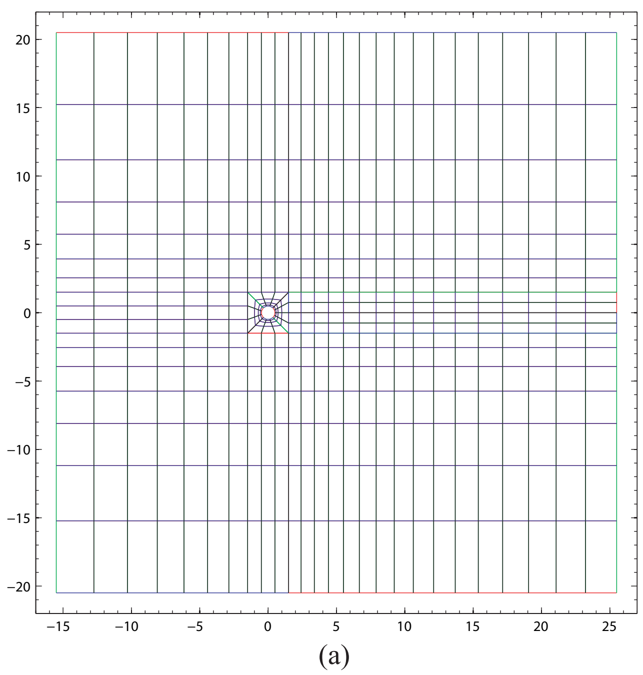
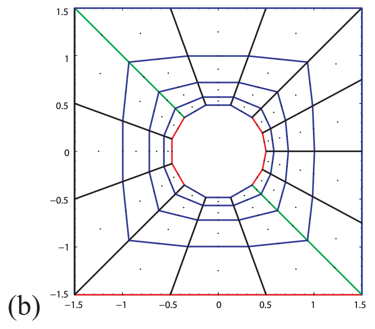
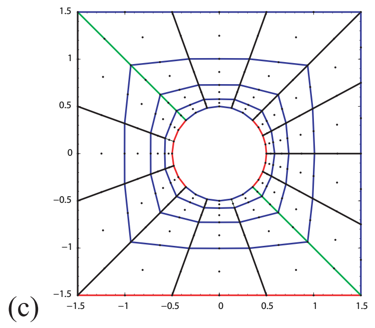




Figure 2.10: (a) Linear ABAQUS mesh (b) p-refined mesh for $p = 2$ with straight edges and (c) p-refined mesh with circular hole.

$E_e = E_0 \mu(\Omega^e)^{-0.5}$ where $\mu(\Omega^e)$ is the area of the element and E_0 is the non-negative quantity that is taken arbitrarily. Since for this case the nodes have to be on the circle of diameter 1, the **X** and **Y** displacement values can be calculated and a boundary value problem can be setup. A weak-form Galerkin finite element formulation is used for the pseudo-elasticity problem. Solving this finite element problem will move the nodes evenly and results in a smooth mesh. The mesh obtained is shown in Fig. 2.10(c).

Using the above concepts with the parametric equations, the higher-order grids for shell structures are generated as in [119]. The parametric equations make it convenient to describe curves and curved surfaces in higher-dimensional spaces. Typically, the mesh in higher-dimension space is constructed for the un-deformed mid-surface of the shell by mapping each nodal position from the parametric space $\bar{\theta} \subseteq \mathrm{R}^2$ on to the nodal locations belonging to the higher-dimension space in R^3. The coordinates of the parametric space $\bar{\theta}$ in R^2 are denoted as (θ_1, θ_2) and for most of the cases, unless stated otherwise, we take $\bar{\theta} \subseteq [0,1] \times [0,1]$. The basic idea is to first create the linear mesh on a unit square region $[0,1] \times [0,1] \subseteq \mathrm{R}^2$, then do the p-refinement and use the necessary parametric equations to get the desired shell geometry in higher-dimension space. To show this an octant of a cylindrical shell is generated as in Fig. 2.11(b). For this first, the linear colored mesh shown in Fig. 2.11(a) is generated. Then, it is p-refined to the required level (here $p = 2$ is used) as shown in Fig. 2.11(b). Then, the necessary parametric equations are used to map the p-refined nodes into higher-dimension space to different shell geometries. The parametric equations to generate a cylindrical surface are

$$\begin{aligned} x &= R\,\sin\left(\frac{\pi}{2}\theta_1\right) \\ y &= L\theta_2 \qquad\qquad \text{where} \quad (\theta_1, \theta_2) \subseteq [0,1] \times [0,1] \\ z &= R\,\cos\left(\frac{\pi}{2}\theta_1\right) \end{aligned} \tag{2.14}$$

where $R = 300$ and $L = 300$ are the radius and length of the shell required. The octant mid-surface of the cylindrical shell is shown Fig. 2.11(c). To generate a hyperboloidal shell with skewed elements, as in Fig. 2.11(d), all it takes is to have an initial linear mesh with skewed elements. Note, the data structures needed to apply the shell boundary conditions remain the same as discussed above. The ‘sidesets’ information which is converted into ‘nodesets’ by the p-refinement MATLAB code is used to apply *essential* type boundary conditions and point loads on the shells. Also, the ‘sidesets’ information is used to apply *natural* type of boundary conditions

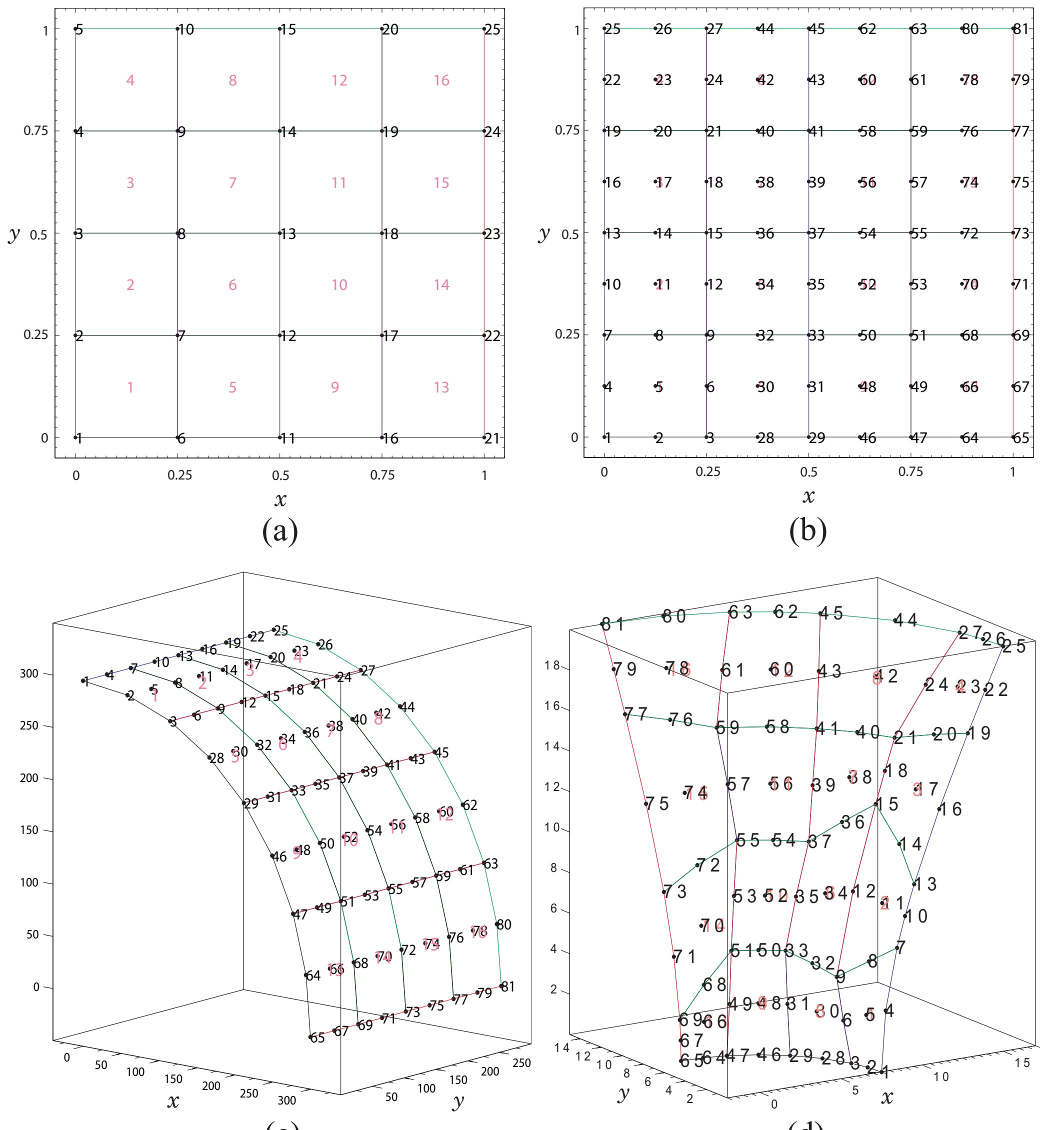


Figure 2.11: (a) Linear mesh (b) p-refined mesh (c) Octant mesh of cylindrical shell (d) Octant skewed-mesh of hyperboloidal shell.

like tractions, pressures, displacement dependent loads etc.

Using shell mechanics, vector algebra and the parametric equations in curvilinear system, the normal and tangential vectors to the mid-surface of the shell can be easily derived. For a cylindrical shell geometry they are

$$\begin{aligned} n_x &= \sin\left(\frac{\pi}{2}\theta_1\right), \quad & t_x &= 0 \\ n_y &= 0\,, & t_y &= 1 \\ n_z &= \cos\left(\frac{\pi}{2}\theta_1\right), \quad & t_z &= 0 \end{aligned} \tag{2.15}$$

Starting with the p-refined mesh shown in Fig. 2.12 along with the parametric equations from Eq. (2.34) and using the above normal and tangential equations, the normal and tangential vectors at each node are plotted in Fig. 2.13(a). If parametric equations of a hyperboloid shell geometry are used, it will result in Fig. 2.13(b). Using these concepts various higher-order grids with seven parameter continuum shell elements [119] are generated for thermo-mechanical analysis of fully nonlinear isotropic, laminated composite and functionally graded elastic shell structures.

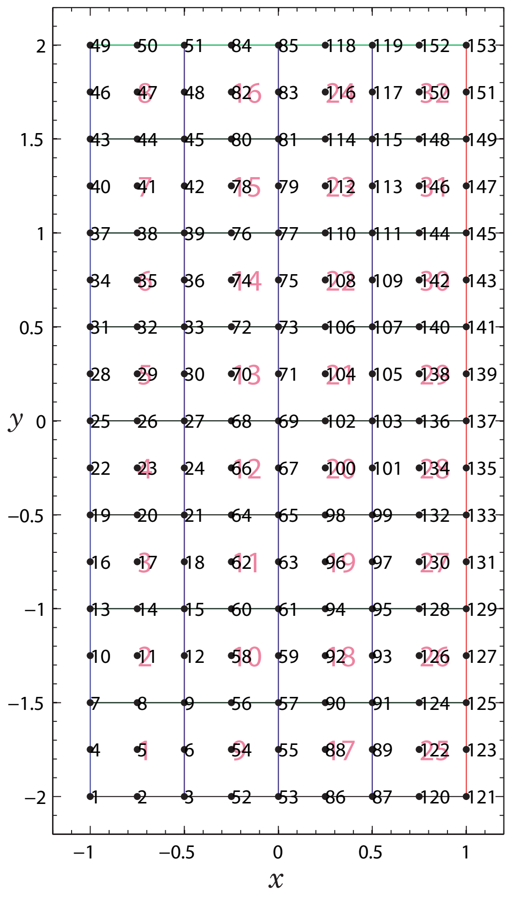
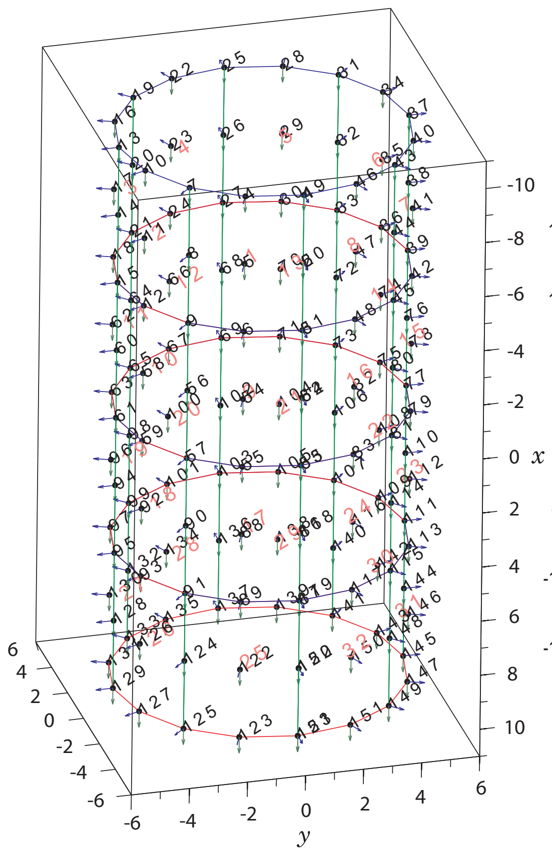


Figure 2.12: A p-refined mesh for $p = 2$.


(a)

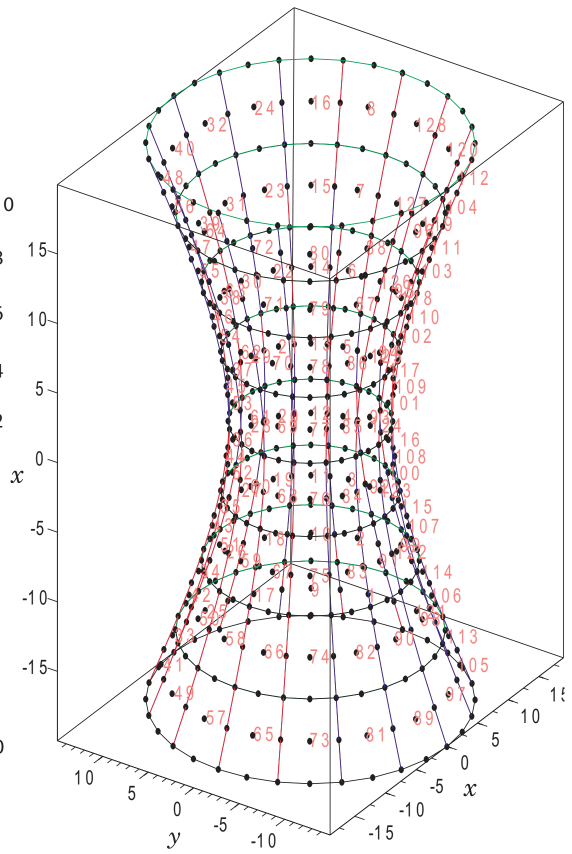

(b)

Figure 2.13: (a) Full mesh of cylindrical shell with normals and tangents and (b) Full mesh of hyperboloidal shell.

2.4 Application of boundary conditions

2.4.1 Boundary conditions

So far, we have seen how to convert the 'sidesets' information (consisting of element number and local element side number on the boundary of linear mesh) to 'nodesets' information (which consists of global node numbers of the p-refined mesh). To apply the *essential* type of boundary conditions, the total number, global node numbers and the corresponding values are specified in the input file to the finite element program for each degree of freedom. To apply *natural* (flux, traction, and outflow) type of boundary conditions, the total number of elements, the element number, the local side number (i.e. 1, 2, 3 or 4) and the value are specified in the input file for each boundary. For example, consider the mesh shown in Fig. 2.14, it is obtained by doing p-refinement. Assume there are two degrees of freedom (ess1 and ess2) at each node in the finite element model. Also, imagine that on the right-side of this domain there is a specified *natural* (traction) boundary condition with a constant value of 1 in the horizontal direction and on the left-side there is a specified *essential* boundary condition with a value of zero for both degrees of freedom. Then the data in the input file will look like in Fig. 2.14. Note, all this information is automatically written to a text file by the p-refinement MATLAB program.

Traditionally, in finite element programs which assemble the element matrices into global full-matrices or banded-matrices (like tri-diagonal, penta-diagonal forms), the *essential* type of boundary conditions are imposed at global level. However if sparse type solvers [74] or element-free solvers are used, it becomes difficult or sometimes impossible to apply these boundary conditions at global level. In this work the *essential* type of boundary conditions are applied at the element level. It has some numerical advantages and it also becomes easy to implement the schur complement

```
// Essential boundary conditions
~ess1_Number~
5
~ess1_Number_END~

~ess1_Node_Value~
1       0.0
2       0.0
3       0.0
4       0.0
5       0.0
~ess1_Node_Value_END~

// Essential boundary conditions
~ess2_Number~
5
~ess2_Number_END~

~ess2_Node_Value~
1       0.0
2       0.0
3       0.0
4       0.0
5       0.0
~ess2_Node_Value_END~

// Traction boundary conditions
~t_Number~
2
~t_Number_END~

~t_Elem_Side_tx_value_ty_value~
3       1     1.0   0.0
4       1     1.0   0.0
~t_Elem_Side_tx_value_ty_value_END~
```

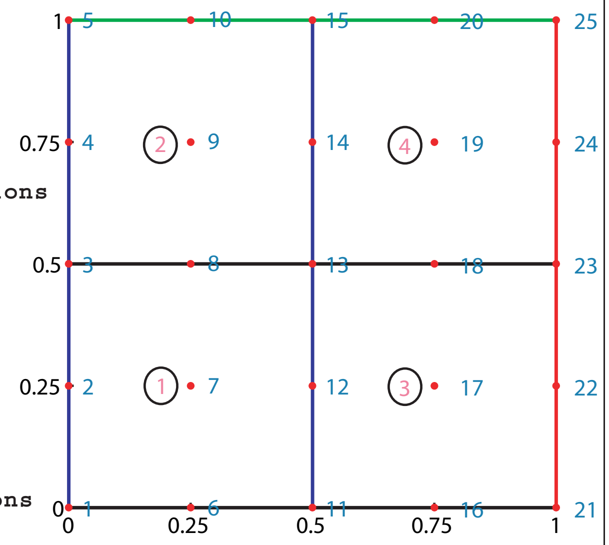


Figure 2.14: Input file with specified *essential* and *natural* boundary conditions.

method (to be discussed next). Converting the global boundary node numbers and the corresponding values for each degree of freedom to the element requires additional logic in the program. The element level information has the element total, the local node number and the element value. Also, to apply *natural* type of boundary conditions at element level, the element total, element side and the element value is required for each one of it. A simple MATLAB code given in Fig. A(a) of appendix A implements this logic.

Once the elemental information of the *natural* and *essential* boundary conditions is obtained, it needs to be applied to the element system of equations. First, a strategy to apply the *essential* type of boundary conditions is presented and next, the *natural* type of boundary conditions are presented. If the *essential* boundary conditions are applied arbitrarily the symmetry of the element matrix gets disturbed, which makes the global matrix un-symmetric and thereby difficult to solve. Few extra algebraic steps are needed to insert the known element value of the *essential* type boundary condition without disturbing the symmetry. The steps involved are explained with a simple example for the the following elemental system of equations

$$\mathrm{K}^e = \begin{bmatrix} 10 & 20 & 40 \\ 20 & 1 & 30 \\ 40 & 30 & 6 \end{bmatrix}, \ \mathrm{F}^e = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} \quad \Rightarrow \quad \begin{bmatrix} 10 & 20 & 40 \\ 20 & 1 & 30 \\ 40 & 30 & 6 \end{bmatrix} \begin{Bmatrix} \Delta_1^e \\ \Delta_1^e \\ \Delta_1^e \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} \tag{2.16}$$

Assume the elemental value of the *essential* boundary condition is known as $\Delta_2^e = 10$. The following algebraic operations show how to replace the known value of this specified boundary condition without affecting the symmetry of the element matrix.

$$\begin{bmatrix} 10 & 20 & 40 \\ 20 & 1 & 30 \\ 40 & 30 & 6 \end{bmatrix} \begin{Bmatrix} \Delta_1^e \\ 10 \\ \Delta_3^e \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} \quad \Rightarrow \begin{bmatrix} 10 & 0 & 40 \\ 20 & 0 & 30 \\ 40 & 0 & 6 \end{bmatrix} \begin{Bmatrix} \Delta_1^e \\ 10 \\ \Delta_3^e \end{Bmatrix} = \begin{Bmatrix} 1 \\ 2 \\ 3 \end{Bmatrix} - 10 \begin{Bmatrix} 20 \\ 1 \\ 30 \end{Bmatrix}$$

$$\begin{bmatrix} 10 & 0 & 40 \\ 20 & 0 & 30 \\ 40 & 0 & 6 \end{bmatrix} \begin{Bmatrix} \Delta_1^e \\ 10 \\ \Delta_3^e \end{Bmatrix} = \begin{Bmatrix} -199 \\ -8 \\ -297 \end{Bmatrix} \Rightarrow \begin{bmatrix} 10 & 0 & 40 \\ 0 & 1 & 0 \\ 40 & 0 & 6 \end{bmatrix} \begin{Bmatrix} \Delta_1^e \\ \Delta_2^e \\ \Delta_3^e \end{Bmatrix} = \begin{Bmatrix} -19 \\ 10 \\ -297 \end{Bmatrix} \tag{2.17}$$

A simple MATLAB code given in Fig. A(b) of appendix A implements this logic.

2.4.2 Integration along element boundaries

In most of the *natural* (flux, traction, outflow) type of boundary conditions, it involves the evaluation of closed-form path (or boundary) integrals like

$$\oint_{\Gamma} ()\, ds \tag{2.18}$$

In solids these integrals occur in structures with traction loads, pressure loads, displacement dependent loads like hydrostatic loads etc. In fluids these occur in the evaluation of various fluxes, volumetric/mass flow rates over control volumes (or surfaces), lift/drag forces on surfaces like airfoils etc. In two-dimensions it leads to evaluation of line integrals along the edges of the elements and in three-dimensions it leads to evaluation of surface integrals over the faces of the elements. For lower-order ($p \leq 2$) two-dimensions elements the evaluation of these boundary integrals is easy and often, they are manually evaluated and hard-wired in the finite element codes. But the evaluation of these integrals for higher-order elements with un-evenly spaced nodes is not straight forward and it needs a more general approach.

In this work, general two-dimensional higher-order spectral elements with curved boundaries are considered. A typical four sided higher-order spectral finite element is depicted in Fig. 2.15. It is important to note that the unit normal vector $\mathbf{n}$ along the element boundary depends continuously on the given location along the curve. Although a two-dimensions case is presented, its extension to three-dimensions analysis is relatively straight-forward and mostly analogous to the present discussion.

In the finite element method it is customary to represent the geometry of the physical domain in addition to the dependent variables using the standard finite element interpolation functions. As a result, the positional vector of each node

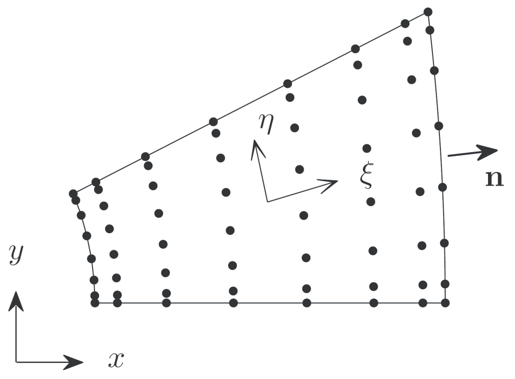


Figure 2.15: A typical two-dimensions higher-order spectral element (shown for $p = 7$).

$\mathbf{x} = (x, y)$ within the Eulerian finite element mesh can be expressed by the following formula that is applicable within a typical finite element

$$\mathbf{x} = \sum_{j=1}^{n=(p+1)} \mathbf{x}_j \psi_j (\xi, \eta) \tag{2.19}$$

In the above expressions,$\psi_j (\xi, \eta)$ represent the interpolation functions associated with a given element. The quantities ξ and η are the natural coordinates. It is important to note that a differential vector in the finite element can be expressed as

$$d\mathbf{x} = d\mathbf{x}_1 + d\mathbf{x}_2 = \mathbf{g}_1 d\xi + \mathbf{g}_2 d\eta \tag{2.20}$$

where a set of linearly independent vectors $\mathbf{g}_j$ are introduced. These are the covariant basis vectors associated with the parametric description of the geometry of the element (where ξ and η are the parameters) and are given by the following formulas

$$\mathbf{g}_1 = \frac{\partial \mathbf{x}}{\partial \xi} = \frac{\partial x}{\partial \xi}\mathbf{e}_x + \frac{\partial y}{\partial \xi}\mathbf{e}_y, \qquad \mathbf{g}_2 = \frac{\partial \mathbf{x}}{\partial \eta} = \frac{\partial x}{\partial \eta}\mathbf{e}_x + \frac{\partial y}{\partial \eta}\mathbf{e}_y \tag{2.21}$$

It is important to note that the covariant basis vectors are in general non-orthogonal, and non-unitary. An important feature of $\mathbf{g}_1$ and $\mathbf{g}_2$ is that they are point-wise tangent to curves in $\boldsymbol{B}_2^e$ sketched out by fixing ξ and η respectively.

2.4.3 Unit normal vectors along element boundaries

The boundary value problems arising in the study of continuous bodies typical require the specification of flux type quantities along at least a portion of the domain boundary. To prescribe such quantities it is necessary to know the unit normal vector along the boundary. In this section the general formulas for determining these vectors along each boundary of a typical finite element are derived. For doing that, the below sides definition is used (same as in Fig. 2.5) for a typical quadrilateral element:

$$\begin{array}{llll} \text{side}-1 \;: & (\xi=1,\eta) & \qquad \text{side}-2 \;: & (\xi,\eta=1) \\ \text{side}-3 \;: & (\xi=-1,\eta) & \qquad \text{side}-4 \;: & (\xi,\eta=-1) \end{array} \tag{2.22}$$

Before presenting the unit normal vectors, the following formulas for unit vectors that are tangent to each element sides are provided

$$\begin{aligned}
\hat{\mathbf{t}}^{(1)} &= \tfrac{1}{\sqrt{\mathbf{g}_2\cdot\mathbf{g}_2}}\mathbf{g}_2\Big|_{\xi=1} = \left[\left(\tfrac{\partial x}{\partial \eta}\right)^2+\left(\tfrac{\partial y}{\partial \eta}\right)^2\right]^{-1/2}\left(\tfrac{\partial x}{\partial \eta}\mathbf{e}_x+\tfrac{\partial y}{\partial \eta}\mathbf{e}_y\right)\Big|_{\xi=1}\\
\hat{\mathbf{t}}^{(2)} &= -\tfrac{1}{\sqrt{\mathbf{g}_1\cdot\mathbf{g}_1}}\mathbf{g}_1\Big|_{\eta=1} = -\left[\left(\tfrac{\partial x}{\partial \xi}\right)^2+\left(\tfrac{\partial y}{\partial \xi}\right)^2\right]^{-1/2}\left(\tfrac{\partial x}{\partial \xi}\mathbf{e}_x+\tfrac{\partial y}{\partial \xi}\mathbf{e}_y\right)\Big|_{\eta=1}\\
\hat{\mathbf{t}}^{(3)} &= -\tfrac{1}{\sqrt{\mathbf{g}_2\cdot\mathbf{g}_2}}\mathbf{g}_2\Big|_{\xi=-1} = -\left[\left(\tfrac{\partial x}{\partial \eta}\right)^2+\left(\tfrac{\partial y}{\partial \eta}\right)^2\right]^{-1/2}\left(\tfrac{\partial x}{\partial \eta}\mathbf{e}_x+\tfrac{\partial y}{\partial \eta}\mathbf{e}_y\right)\Big|_{\xi=-1}\\
\hat{\mathbf{t}}^{(4)} &= \tfrac{1}{\sqrt{\mathbf{g}_1\cdot\mathbf{g}_1}}\mathbf{g}_1\Big|_{\eta=-1} = \left[\left(\tfrac{\partial x}{\partial \xi}\right)^2+\left(\tfrac{\partial y}{\partial \xi}\right)^2\right]^{-1/2}\left(\tfrac{\partial x}{\partial \xi}\mathbf{e}_x+\tfrac{\partial y}{\partial \xi}\mathbf{e}_y\right)\Big|_{\eta=-1}
\end{aligned} \tag{2.23}$$

The unit normal vectors are determined by taking cross products of the above tangent vectors with $\mathbf{e}_z$ and can be expressed as

$$
\begin{aligned}
\mathbf{n}^{(1)} = \hat{\mathbf{t}}^{(1)} \times \mathbf{e}_z &= \left[\left(\tfrac{\partial x}{\partial \eta}\right)^2 + \left(\tfrac{\partial y}{\partial \eta}\right)^2\right]^{-1/2} \left(\tfrac{\partial y}{\partial \eta}\mathbf{e}_x - \tfrac{\partial x}{\partial \eta}\mathbf{e}_y\right)\Big|_{\xi=1} \\
\mathbf{n}^{(2)} = \hat{\mathbf{t}}^{(2)} \times \mathbf{e}_z &= \left[\left(\tfrac{\partial x}{\partial \xi}\right)^2 + \left(\tfrac{\partial y}{\partial \xi}\right)^2\right]^{-1/2} \left(-\tfrac{\partial y}{\partial \xi}\mathbf{e}_x + \tfrac{\partial x}{\partial \xi}\mathbf{e}_y\right)\Big|_{\eta=1} \\
\mathbf{n}^{(3)} = \hat{\mathbf{t}}^{(3)} \times \mathbf{e}_z &= \left[\left(\tfrac{\partial x}{\partial \eta}\right)^2 + \left(\tfrac{\partial y}{\partial \eta}\right)^2\right]^{-1/2} \left(-\tfrac{\partial y}{\partial \eta}\mathbf{e}_x + \tfrac{\partial x}{\partial \eta}\mathbf{e}_y\right)\Big|_{\xi=-1} \\
\mathbf{n}^{(4)} = \hat{\mathbf{t}}^{(4)} \times \mathbf{e}_z &= \left[\left(\tfrac{\partial x}{\partial \xi}\right)^2 + \left(\tfrac{\partial y}{\partial \xi}\right)^2\right]^{-1/2} \left(\tfrac{\partial y}{\partial \xi}\mathbf{e}_x - \tfrac{\partial x}{\partial \xi}\mathbf{e}_y\right)\Big|_{\eta=-1}
\end{aligned}
\tag{2.24}
$$

2.4.4 Differential arc lengths

To integrate flux like quantities such as traction vectors along an element boundary, it becomes necessary to produce an expression for the differential arc length of the boundary. It can be shown that the differential lengths of each element side of $\boldsymbol{B}_2^e$ can be expressed as

$$
\begin{aligned}
ds^{(1)} &= \sqrt{d\mathbf{x}_2 \cdot d\mathbf{x}_2}\big|_{\xi=1} = \sqrt{\left(\tfrac{\partial x}{\partial \eta}\right)^2 + \left(\tfrac{\partial y}{\partial \eta}\right)^2}\, d\eta|_{\xi=1} \\
ds^{(2)} &= \sqrt{d\mathbf{x}_1 \cdot d\mathbf{x}_1}\big|_{\eta=1} = \sqrt{\left(\tfrac{\partial x}{\partial \xi}\right)^2 + \left(\tfrac{\partial y}{\partial \xi}\right)^2}\, d\xi|_{\eta=1} \\
ds^{(3)} &= \sqrt{d\mathbf{x}_2 \cdot d\mathbf{x}_2}\big|_{\xi=-1} = \sqrt{\left(\tfrac{\partial x}{\partial \eta}\right)^2 + \left(\tfrac{\partial y}{\partial \eta}\right)^2}\, d\eta|_{\xi=-1} \\
ds^{(4)} &= \sqrt{d\mathbf{x}_1 \cdot d\mathbf{x}_1}\big|_{\eta=-1} = \sqrt{\left(\tfrac{\partial x}{\partial \xi}\right)^2 + \left(\tfrac{\partial y}{\partial \xi}\right)^2}\, d\xi|_{\eta=-1}
\end{aligned}
\tag{2.25}
$$

2.5 Schur complement method

The high-order methods are memory minimizing when compared to the lower-order methods of same accuracy. However, compared to low-order methods, they require more computations per degree of freedom. The memory requirement can be minimized by using reduction methods, like the schur complement method. It is also called static node condensation or static node substructuring [101, 24, 51, 90].

For higher-order elements the number of interior nodes are more than the boundary nodes (see Fig. 2.16). So if some how these interior nodes are removed from the global system of equations then the memory requirement can be greatly reduced.

mesh.ETYPE	Total Nodes	Boundary Nodes	Interior Nodes
2	4	4	0
4	16	12	4
8	64	28	36
12	144	44	100

Figure 2.16: Boundary and Interior nodes for higher-order elements

To make this idea clear consider the graded S-duct mesh shown in Fig. 2.17(a). Note, this mesh is generated in ABAQUS and p-refined to $p = 3$ as explained in the previous sections. The nodes are plotted as black dots and the element numbers and global node numbers are suppressed for clarity. The statically condensed mesh is shown in the Fig. 2.17(b). All the element interior nodes are removed in this mesh. In Fig. 2.18(a), the sparsity pattern of a typical mesh (with $p = 4$ and randomly numbered elements) is shown and in Fig. 2.18(b), the sparsity pattern of it's statically condensed mesh is shown. It is clear that for this mesh, the size of the shcur complement system is about 30 % of original system. Also, the system of equations are less dense with the number of non-zeros about 20-22 % of the original mesh. This greatly reduces the amount of memory required to store the global matrix which becomes prominent as the discretization of the element is increased. The other benefit is that it improves the condition number [51] of the matrix and makes the system less dense and hence robust direct and iterative solvers can be used for faster convergence.

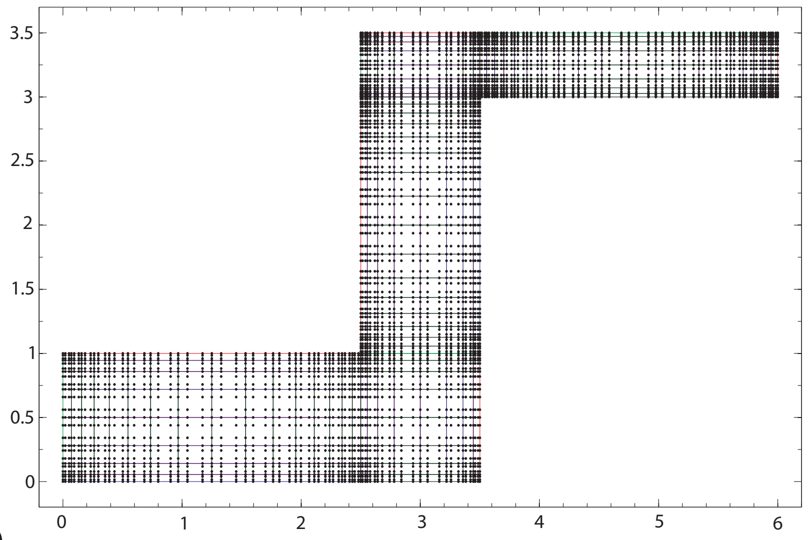
The basic idea of schur complement is to eliminate the unknowns corresponding to the interior nodes using node condensation, so that the overall size of the assembled global system gets reduced considerably. To achieve this a new connectivity matrix (ECON) needs to be generated for the statically condensed mesh. This can be achieved by selective numbering of the elemental boundary nodes followed by numbering the elemental interior nodes on the original mesh as in [51] or by creating a new node numbering scheme on the statically condensed mesh. Since the unknowns corresponding to the elemental interior nodes are not coupled to other elements, it is possible to split the elemental system of equations corresponding to the components of the interface (or boundary) nodes and the interior nodes. From this, the unknowns corresponding to the interior nodes can be explicitly removed by doing suitable algebraic operations. Once this assembled global system is solved, the unknowns corresponding to the boundary nodes of each element can be extracted and in turn the unknowns corresponding to the interior nodes can be solved. To make this idea clear, consider the typical elemental system of equations represented below

$$[\mathrm{K}^e]\,\{\Delta^e\} = \mathrm{F}^e \tag{2.26}$$

where K^e is the sparse but full elemental stiffness matrix (it can be symmetric or unsymmetric), Δ^e is the unknown elemental solution vector and F^e is the elemental force vector. The above elemental system can be separated in to blocks of matrices as shown below

$$\begin{bmatrix} [\mathrm{K}^e_{bb}] & [\mathrm{K}^e_{bi}] \\ [\mathrm{K}^e_{ib}] & [\mathrm{K}^e_{ii}] \end{bmatrix} \begin{Bmatrix} \{\Delta^e_b\} \\ \{\Delta^e_i\} \end{Bmatrix} = \begin{Bmatrix} \{\mathrm{F}^e_b\} \\ \{\mathrm{F}^e_i\} \end{Bmatrix} \tag{2.27}$$

where K^e_{bb} is the block of matrix corresponding to the couplings of pure boundary-


(a)

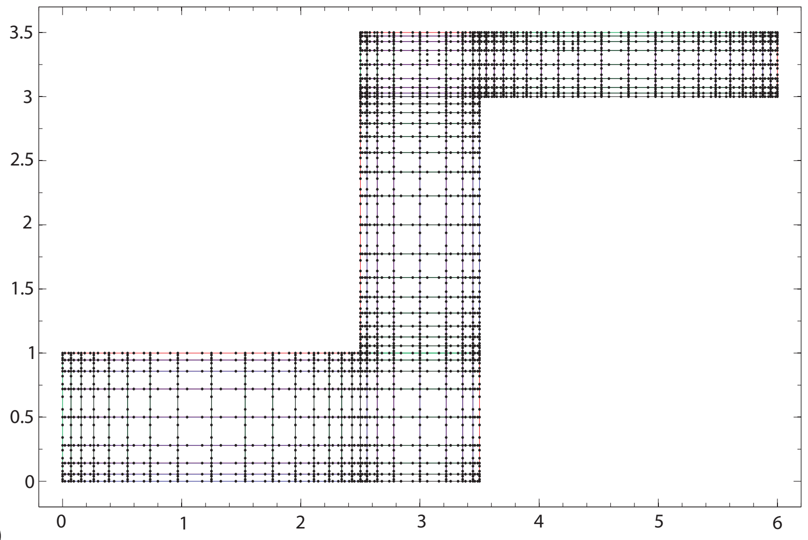

(b)

Figure 2.17: (a) S-duct mesh for $p = 3$ and (b) Statically condensed mesh.

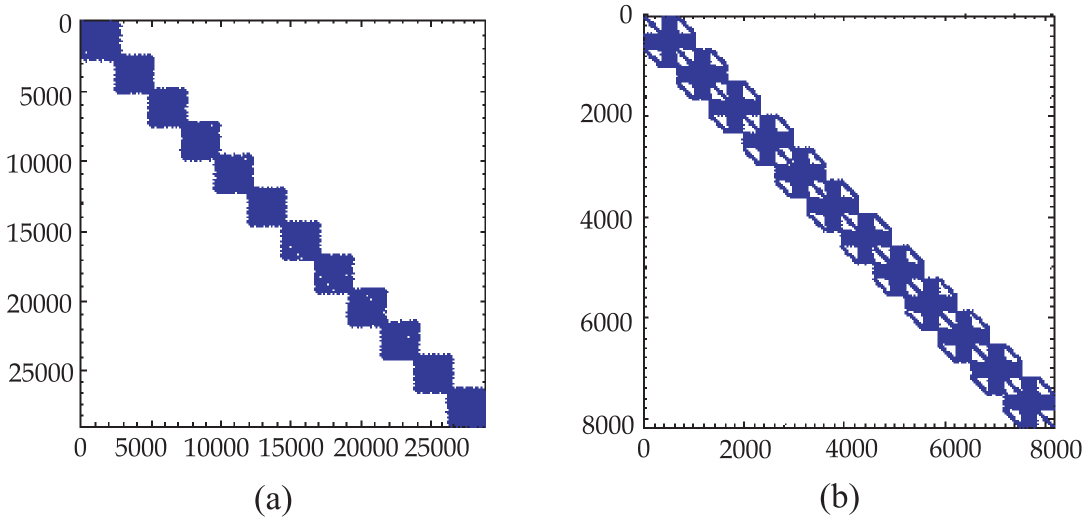


Figure 2.18: (a) Sparsity pattern of a typical mesh and (b) Sparsity pattern of statically condensed mesh.

boundary elemental nodes, K^e_{ii} is the block of matrix corresponding to the couplings of pure interior-interior elemental nodes, K^e_{bi} is the block of matrix corresponding to the couplings of boundary-interior elemental nodes, K^e_{ib} is the block of matrix corresponding to the couplings of interior-boundary elemental nodes, Δ^e_b is the unknown elemental solution vector corresponding to the boundary nodes, Δ^e_i is the unknown elemental solution vector corresponding to the interior nodes, F^e_b is the force vector corresponding to the boundary nodes and F^e_i is the force vector corresponding to the interior nodes. The Eq. (2.27) can be solved for Δ^e_b by suitable block multiplication techniques, and the final equation can be written as

$$\left[[\mathrm{K}^e_{bb}] - [\mathrm{K}^e_{bi}]\,[\mathrm{K}^e_{ii}]^{-1}\,[\mathrm{K}^e_{ib}]\right]\{\Delta^e_b\} = \{\mathrm{F}^e_b\} - [\mathrm{K}^e_{bi}]\,[\mathrm{K}^e_{ii}]^{-1}\,\{\mathrm{F}^e_i\} \tag{2.28}$$

From Eq. (2.28), it is clear that to solve for Δ^e_b first, $\left[[\mathrm{K}^e_{bb}] - [\mathrm{K}^e_{bi}]\,[\mathrm{K}^e_{ii}]^{-1}\,[\mathrm{K}^e_{ib}]\right]$ has to be constructed. This is the schur complement of element matrix K^e, it involves matrix-vector operations, matrix-matrix operations and also matrix-inverse operations which are computationally cheap. To implement the element level static node condensation in the finite element code, the `BLAS` (Basic Linear Algebra Subprograms) and `Goto BLAS` subroutines are used by linking to the `LAPACK` (Linear Algebra PACKage) library. The element inverse operations are performed via Gaussian elimination with partial pivoting using the standard `LAPACK` subroutine `dgesv` (General Matrix Factorization and Multiple Right-Hand Side Solve).

$$[\mathrm{K}_{ii}^{e}]\,\{\Delta_i^e\} = \{\mathrm{F}_i^e\} - [\mathrm{K}_{ib}^{e}]\,\{\Delta_b^e\} \tag{2.29}$$

Note, since the evaluation of Δ_b^e and Δ_i^e involves the inversion of matrices, a blind increase in the element discretization might lead to increase in the computational cost for doing these matrix-inverse operations. Hence a balance is needed for optimal results. The schur complement technique is also well suited for element-by-element methods in which the unknowns corresponding to the elemental boundary nodes are solved directly and then the interior unknowns are solved in an element-by-element fashion. The element-by-element methods and element-free methods are used for problems where large memory needed for global system equations. You can find more about these algorithms in the works of Jiang [45].

2.6 Higher-order finite element strategies

2.6.1 An abstract higher-order problem

In this section, an overview of the general steps involved in a higher-order finite element formulation are presented. Among these, the tasks that are independent (i.e., which can be executed simultaneously) of other tasks are identified. To explain these, consider a typical finite element formulation which leads to a set of linear algebraic equations for the e'th element of the form

$$[\mathrm{K}^e]\{\Delta^e\} = \{\mathrm{F}^e\} \tag{2.30}$$

where $[\mathrm{K}^e]$ is the element stiffness matrix, $\{\Delta^e\}$ is the unknown element solution vector and $\{\mathrm{F}^e\}$ is the element force vector. In traditional finite element codes, these equations are assembled to obtain the global stiffness matrix and force vectors as

$$[\mathrm{K}]\{\Delta\} = \{\mathrm{F}\} \tag{2.31}$$

where

$$[K] = \mathop{\mathbf{A}}_{e=1}^{\mathrm{NE}}[K^e], \quad \{F\} = \mathop{\mathbf{A}}_{e=1}^{\mathrm{NE}}\{F^e\}, \quad \{\Delta\} = \mathop{\mathbf{A}}_{e=1}^{\mathrm{NE}}\{\Delta^e\} \tag{2.32}$$

and $\mathbf{A}$ is a symbolic representation of the global finite element assembly operator. The boundary conditions are usually applied to these matrices at global level and then solved. Instead, as explained above, they are applied at element level to facilitate the implementation of the schur complement method. After static node condensation, assume the linear algebraic equations for the e'th element corresponding to the element boundary nodes be represented as

$$\left[\bar{\mathrm{K}}^e\right]\left\{\bar{\Delta}_b^e\right\} = \left\{\bar{\mathrm{F}}^e\right\} \tag{2.33}$$

where $\left[\bar{\mathrm{K}}^e\right]$ is the statically condensed element stiffness matrix, $\left\{\bar{\Delta}_b^e\right\}$ is the unknown elemental solution vector corresponding to the boundary nodes and $\left\{\bar{\mathrm{F}}^e\right\}$ is the statically condensed element force vector. The global system of equations are setup by combining these *statically condensed* element equations into the following expression

$$\left[\bar{\mathrm{K}}\right]\left\{\bar{\Delta}\right\}=\left\{\bar{\mathrm{F}}\right\} \tag{2.34}$$

where

$$\left[\bar{K}\right]=\mathop{\mathbf{A}}_{e=1}^{\mathrm{NE}}\left[\bar{K}^e\right], \quad \left\{\bar{F}\right\}=\mathop{\mathbf{A}}_{e=1}^{\mathrm{NE}}\left\{\bar{F}^e\right\}, \quad \left\{\bar{\Delta}\right\}=\mathop{\mathbf{A}}_{e=1}^{\mathrm{NE}}\left\{\bar{\Delta}_b^e\right\} \tag{2.35}$$

The element-level equations of a particular element are completely independent of the equations associated with any other element in the mesh. So, the element-level operations of constructing, applying boundary conditions and static node condensation to form $\left[\bar{\mathrm{K}}^e\right]$ and $\left[\bar{\mathrm{F}}^e\right]$, can be performed concurrently (in parallel) as:

(I) Loop over all finite elements: $e = 1, \mathrm{NE}$ (parallel)

- Numerically evaluate element coefficient matrix $[K^e]$ and force vector $\{F^e\}$
- Apply *essential* and *natural* boundary conditions to $[K^e]$ and $\{F^e\}$
- Perform static node condensation to construct $[\bar{K}^e]$ and $\{\bar{F}^e\}$
- Assemble components of $[\bar{K}^e]$ into the global sparse coefficient matrix $[\bar{K}]$ and $\{\bar{F}^e\}$ into the global force vector $\{\bar{F}\}$

Even after condensing out the interior nodes through static node condensation, for large scale simulations involving higher-order finite elements, it becomes impractical (in terms of memory) to construct a full or banded coefficient matrix $\left[\bar{\mathrm{K}}\right]$. To reduce this memory requirement, the global coefficient matrix $\left[\bar{\mathrm{K}}\right]$ is represented in compressed sparse row (CSR) or compressed sparse column (CSC) format. This sparse vector storage format is based on storage by row and column indices. Parallel construction of the global finite element system in compressed sparse formats, in a manner that is both fast and memory efficient, is a far-less trivial task. To keep the present discussion brief, in step (II) below, we only present the steps involved in efficient parallel global assembly operator $\mathbf{A}$, for the case when the global coefficient matrix $\left[\bar{\mathrm{K}}\right]$ is sparse (i.e., populated primarily by zeros). The complete details on how to generate compressed sparse row (CSR) and compressed sparse column (CSC) formats of the global coefficient matrix $\left[\bar{\mathrm{K}}\right]$ will be presented in our up-coming paper [74]. There are some good open-source libraries like `PETSc` (Portable, Extensible Toolkit for Scientific Computation - http://www.mcs.anl.gov/petsc/) which can also be used to accomplish the same.

(II) Sort global coefficient matrix $[\bar{K}]$ into *compressed sparse column* (CSC) or *compressed sparse row* (CSR) form (parallel)

- Sort column (or row) indices of each row (or column) of $[\bar{K}]$ in non-decreasing order
- Sum repeated entries of $[\bar{K}]$ to enforce compatibility of primary variables
- Remove "numerical" zeros from sparse matrix $[\bar{K}]$

After the above step, solve the statically condensed system of equations by linking to either direct or iterative solvers. This step can be done in parallel or serial depending on the solver selected.

(III) Solve global system of equations for all element boundary degrees of freedom using an appropriate linear solver library to get $\left\{\bar{\Delta}\right\}$ of Eq. (2.34) (parallel)

- Link to direct solvers
- Link to iterative solvers

After step (III), solve for degrees of freedom corresponding to interior nodes of each element as shown below:

(IV) Loop over all finite elements: $e = 1, \mathrm{NE}$ (parallel)

- Extract boundary degrees of freedom $\left\{\bar{\Delta}_b^e\right\}$ for each element from $\left\{\bar{\Delta}\right\}$ and solve for interior degrees of freedom $\left\{\bar{\Delta}_i^e\right\}$
- Rearrange $\left\{\bar{\Delta}_i^e\right\}$ and $\left\{\bar{\Delta}_b^e\right\}$ of each element to get solution vector $\{\Delta^e\}$
- Assemble element solution $\{\Delta^e\}$ to get global solution $\{\Delta\}$ of Eq. (2.31)

The steps (I)-(IV) briefly summarize the critical steps in an abstract higher-order finite element implementation that is used in this research work.

2.6.2 Parallel processing paradigm

Having identified the parallel and serial tasks in the higher-order finite element methodology, we focus on parallel strategies for efficient implementation of those tasks. The most commonly used and highly evolved parallel computing environments are the shared-address-space architectures (`OpenMP` paradigm) and the distributed-address-space architectures (`MPI` paradigm). Although, the Message Passing Interface (`MPI`) is more general and uses relatively cheap distributed memory, its implementation is complicated owing to the need to exchange messages across switches of a network for communication. Considerable code and algorithm changes are needed to incorporate `MPI` to parallelize a program. On the other hand, for shared-address-space architectures the communication is implicitly specified since all the processors have access to the same pool of shared memory. Hence, most of the programming techniques for shared-address-space machines focus on concurrency and synchronization thus simplifying the program.

The `OpenMP` paradigm is an Application Programming Interface (API) that supports multi-threading which can be simply interpreted as a Fork-Join model. In this model, the program begins as a single process on the master thread which executes sequentially until a "parallel region" is encountered. Then the master thread *forks* (spawns) into a team of parallel threads which execute the statements in the "parallel region" concurrently. After threads complete the work in the "parallel region", they synchronize before terminating to *join* back into the master thread. The section of the code that is meant to run in parallel, is marked with pre-processor directives which will spawn the parallel threads before the section is executed. Using just 3 or 4 simple and limited set of pre-processor directives in code, complicated tasks can be accomplished on shared memory machines. Also, these pre-processor directives are additions to the source code that can be ignored by a non `OpenMP` compiler. Serial and parallel versions share the same source code - the serial compiler simply overlooks the parallel code additions. Addition of a directive does not break the serial code. New serial code enhancements outside parallel regions will not break the parallel program. Hence changing serial code to `OpenMP` parallel code need no big modifications. The `OpenMP` parallel code can still run in serial and hence it is easy to debug. The other feature of `OpenMP`, is the capability to incrementally parallelize a serial program (very different from `MPI` which typically requires an all or nothing approach). Both *task* and *data* parallelism can be easily achieved. The number of threads can be assigned by the runtime environment based on environment variables or in the code using functions. These `OpenMP` pre-processor directives are very portable and is available in various programming languages like Fortran, C/C++. Also, it has a public forum for API and an active support (http://openmp.org). The `OpenMP` functions are included in a header file labeled `omp.h` for C/C++.

The main crux in the implementation of the `OpenMP` philosophy lies in the realization of the implicit communication between the threads. In `OpenMP` paradigm, the threads read and write shared variables, and hence there is no need for explicit communications with messages. If threads need to use local variables to do work that does not need to be globally visible outside the "parallel region", the variables can be declared as private. Each thread will have its own copy of the private variable from the processor. This means threads can "cache" their data; not required to maintain exact consistency with real (main) memory all of the time. The cache is a small memory nearer to the processor and acts as a low-latency, high-bandwidth storage and hence more efficient. When all threads must view a shared variable identically, programmer is responsible for ensuring that the variable is `FLUSH`ed by all threads to the shared memory pool to get the latest value.

For the steps (I)-(IV) in the previous subsection, the `OpenMP` parallelization is achieved with appropriate placement of pre-compiler directives prior to parallelizable for-loops using the C/C++ specific `OpenMP` syntax `#pragma omp parallel for` as shown in Fig. 2.19 (Note, the steps (I)-(IV) are marked on the left side of the figure). This syntax marks the beginning of the "parallel region" and the loop encountered immediately is executed in parallel. The code becomes serial immediately after the end of the loop. As a side-note, in the beginning of Fig. 2.19, we use the `OpenMP` syntax to do re-initialization of solution vector to zero. Here the variable "`i`" is a shared variable and it resides in the shared main memory pool. This is not efficient, as the main memory has greater latency compared to cache memory. Since no other `OpenMP` threads will have access to the loop indices, a local copy of the variable "`i`" can be created using private clause in `#pragma omp parallel for private(n)` as shown for steps (I) and (IV) in Fig. 2.19. In these steps, the variable "`n`" is local for each thread and it resides in the cache memory instead of the shared main memory. The variables that are not explicitly declared as private are by default treated as shared. Another case of importance is *synchronization*, especially when different threads are working on different parts of a shared array. To ensure that all threads in a team have a consistent view of this array object, the syntax `#pragma omp flush [(list)]` can be used. In the absence of a list, all visible variables are flushed. With these simple pre-compiler statements the `OpenMP` based finite element code can be easily developed.

The numerical implementation is done using IBM's AIX v11.1 compiler with `OpenMP` support on an IBM Cluster-1600 with IBMs 1.9 GHz RISC Power5+ processors. Each node is a symmetric multi-processor (SMP) system with 16 processors and 25 GB of usable shared memory. The number of threads can be assigned by the runtime environment based on environment variables or in code using necessary functions. In this work, we use both interactive and batch modes to run the finite element code and the threads are assigned at run time using environment variables. In Fig. 2.20, we present a batch job file for the above IBM cluster that is designed to run on 8-processors using `OpenMP` paradigm. The typical parameters (like number of processors, wall-clock time etc.) that are usually changed are highlighted in color. Note, we include commands like `module load umfpack` and `module load gotoblas` in the batch job file to link to `UMFPACK` and `BLAS`. Also, as comments at the bottom, we mention IBM's compilation directives starting with `xlC_r` for `ESSL` and `UMFPACK` subroutines. The final thing to be noted in the compilation directive is the aggressiveness of the optimization. The common compiler option `-Ox (x = 0,1...5)` specifies the level of aggressiveness. The greater the aggressiveness level, the more the code will be changed. Hence, one must use caution with `x > 3`, as it could change the results. Throughout the present work we fix the aggressiveness level at `x = 3`. The `-O3` optimizations include loop permutation, loop tiling, loop skewing, loop reversal, unimodular transformations, forward substitution, and expression simplification.

```
//---------------------------------------------------------------------
//
// Begin file
//
//---------------------------------------------------------------------

// Standard C++ header files

// OpenMP header file
#include <omp.h>

// User defined header files
#include "fluid.h"
#include "umfpackSolve.h"

// Function to solve the finite element equations using Schur
complement
void fluid_f_schur(fluid_c_mesh & mesh, schurComp & schur,
                   sparseMatrix & sparseMat, fluid_c_material &
                   material, fluid_c_boundary & boundary, gen_interp2D
                   & sfL2D, gen_interp2DS & sfL2DS, fluid_c_femSol &
                   solution)
{
      //-----------------------------------------------------------------
      //
      // Solve for variables at boundary nodes
      //
      //-----------------------------------------------------------------

      // Declaration of integers
      int n, schurBI;

      // Re-initialize delU to zero
      // solution.initializeU();
      int Equations = mesh.NN*mesh.DFPN;
      #pragma omp parallel for
      for ( int i = 0 ; i < Equations ; i++ ) {
            U[i] = 0.0;
      }


      // Set Schur condition to element boundary degrees of freedom
      schurBI = 0;

      // Loop over each element to build finite element equations
      #pragma omp parallel for private(n)
      for ( n = 0 ; n < mesh.NE ; n++ ) {
(I)         fluid_f_elMat(mesh, sparseMat, material, boundary, sfL2D,
                          sfL2DS, solution, n, schur, schurBI);
      }

      // Re-arrange sparse system of equations into compressed-row form
(II)  sparseMat.sort();

      // Call UMFPACK library to solve linear algebraic equations
      umfpackSolve(sparseMat.get_equations(),
                   sparseMat.get_k_pointer(), sparseMat.get_k_j(),
(III)              sparseMat.get_k_value(), sparseMat.get_f_value(),
                   sparseMat.get_x_value());

      // Update boundary nodes of delU
      solution.populateUboundary(schur, sparseMat.get_x_value());

      //-----------------------------------------------------------------
      //
      // Solve for variables at interior nodes
      //
      //-----------------------------------------------------------------

      // Set Schur condition to element interior degrees of freedom
      schurBI = 1;

      // Loop over each element to build finite element equations
      #pragma omp parallel for private(n)
      for ( n = 0 ; n < mesh.NE ; n++ ) {
            fluid_f_elMat(mesh, sparseMat, material, boundary, sfL2D,
(IV)                      sfL2DS, solution, n, schur, schurBI);
      }

      // End of function
      return;
}

//---------------------------------------------------------------------
//
// End of file
//
//---------------------------------------------------------------------
```

Figure 2.19: A simple OpenMP paradigm C++ code for steps (I)-(IV).

```
#------------------------- Job file: openMP.job -------------------------
#@ shell                  = /bin/ksh
#@ comment                = 8-proc OpenMP Job
#@ initialdir             = /scratch/vallala/the_iter_lid_p9_b50_g10_npc8
#@ job_name               = openMP
#@ error                  = $(job_name).o$(schedd_host).$(jobid).$(stepid)
#@ output                 = $(job_name).o$(schedd_host).$(jobid).$(stepid)
#@ job_type               = parallel
#@ resources              = ConsumableCpus(8) ConsumableMemory(8500mb)
#@ wall_clock_limit       = 3:00:00
#@ node                   = 1
#@ total_tasks            = 1
#@ notification           = error
#@ queue
#

# Copy executable and input files to temporary directory
cd $TMPDIR
cp $LOADL_STEP_INITDIR/fem.out .
cp $LOADL_STEP_INITDIR/fluid-input.inp .

# Link to UMFPACK and BLAS
module load umfpack
module load gotoblas

# Set OpenMP parameters for analysis
export OMP_NUM_THREADS=8
export OBJECT_MODE=64
export OMP_DYNAMIC=FALSE           # Most common but not always
export MALLOCMULTIHEAP=HEAPS:8
export AIXTHREAD_SCOPE=S

# Run executable file
./fem.out
# /usr/local/bin/jobinfo

# Copy solution text files back into parent directory
cp fluid-solution.out       $LOADL_STEP_INITDIR
cp fluid-summary.out        $LOADL_STEP_INITDIR
cp fluid-ls-functional.out  $LOADL_STEP_INITDIR
cp fluid-timer.out          $LOADL_STEP_INITDIR

# USEFUL INFORMATION:

# Compile (ESSL): xlC_r -qsmp=omp -q64 -O3 -lesslsmp -L$GOTOBLAS_LIB -lgoto64 -
                 lpthread -lm *.cpp

# Compile (UMFPACK): xlC_r -qsmp=omp -q64 -O3 -I$UMFPACK_INC -L$UMFPACK_LIB -
                    lamd -lumfpack -L$GOTOBLAS_LIB -lgoto64 -lpthread -lm
                    *.cpp

# To submit a job:      llsubmit openmp.job
# To check job status:  llq -u userid
# To cancel a job:      llcancel jobNumber
#------------------------- Job file: openMP.job -------------------------
```

Figure 2.20: An OpenMP batch job file.

2.6.3 A note on direct and iterative solvers

Since the global coefficient matrix is stored in compressed sparse column (CSC) or compressed sparse row (CSR) format, the solver step (III) can be linked to either direct or iterative solvers. Also, it will be shown in the next section that the least-squares formulation always yields symmetric positive-definite (SPD) coefficient matrix and hence, robust direct and iterative solvers can be used. In the problems of finding the root of an equation (or a solution of a system of equations), an iterative method uses an initial guess to generate successive approximations to a solution. In the case of a system of linear equations, the two main classes of iterative methods are the stationary iterative methods (splitting), and the more general non-stationary (Krylov subspace) methods. In splitting iterative methods, the operator `A` of the linear system of equations `Ax = b`, is usually decomposed into one of the diagonal component `D`, the remainder component `R`, the upper triangular component `U`, the lower triangular component `L` etc. and the linear system is transformed to form a approximating "correction equation", which is solved repeatedly until the error in the result (residual) is less than the tolerance limit. While these methods are simple to derive, implement, and analyze, convergence is only guaranteed for a limited class of matrices. Some of the examples of stationary iterative methods are the Jacobi method, the Gauss–Seidel method and the successive over-relaxation method.

On the other hand, the Krylov subspace methods work by forming (spanning) a basis (the Krylov subspace) of the sequence of successive matrix powers times the initial residual (the Krylov vector sequence). The approximations to the solution are then formed by minimizing the residual over the subspace formed. If the operator `A` of the linear system of equations `Ax = b` is self-adjoint, it can be solved by the conjugate gradient method (CG) and the eigenvalue system, $\mathtt{Ax} = \lambda\mathtt{x}$, can be solved by the Lanczos iterative method. If `A` is non self-adjoint, the linear system can be solved by the bi-conjugate gradient method (Bi-CG), the generalized minimal residual method (GMRES) etc. and the eigenvalue system by the Arnoldi iterative method. Since these methods have a basis, in the absence of round-off errors, these methods converge in N iterations, where N is the system size. Even for the Krylov subspace methods, the rate of convergence is not very satisfactory.

The rate of convergence of the linear system of equations `Ax = b` depends on the condition number of the operator `A`, lower the number, faster the convergence rate. All the Krylov subspace methods presented above will have faster convergence if applied to preconditioned system formed from `Ax = b` by applying various preconditioners. The classical preconditioners for any linear operator `A`, are derived from the stationary (splitting) iterative methods mentioned above and can be easily incorporated in Krylov subspace methods. The preconditioned Krylov subspace methods can be considered as accelerations of stationary iterative methods. Depending on the splitting method, one can have Jacobi preconditioner or a Symmetric Gauss-Seidel (SGS) preconditioner, etc.

For a SPD coefficient matrix, preconditioned conjugate gradient (PCG) methods are optimal choice. The convergence rate is strongly dependent on the condition number of the (preconditioned) coefficient matrix. The IBM's `ESSL` and `Parallel ESSL` libraries, provide a good number of these iterative solver subroutines. Both `ESSL` and `Parallel ESSL` libraries support 32-bit and 64-bit Fortran, C and C++ serial, `OpenMP` and `MPI` parallel applications running under AIX and Linux operating systems. We have successfully linked our finite element program with the subroutine `dsris` (Iterative Linear System Solver for a General or Symmetric Sparse Matrix Stored by Rows). This subroutine solves a general or symmetric sparse linear system of equations, using an iterative algorithm with or without preconditioning.

Although the iterative methods are cheap and suited for large scale problems they are slow and the preconditioner determines the number of iterations. Also, convergence is not always guaranteed. These short-comings can be overcome using direct methods, which are better suited for small to medium-large-scale problems. The solution is always guaranteed and unique. In the absence of roundoff errors, direct methods would deliver an exact solution by solving the linear system of equations `Ax = b` by any one of Gaussian elimination, `LU` factorization, `QR` method, Block algorithms or the more advanced multi-frontal solvers. Since we implement the schur complement method to eliminate all interior degrees of freedom, the memory required for the coefficient matrix is significantly reduced. Hence in this work we use direct solvers for relatively large problems (up to `0.5X10^6` degrees of freedom). To solve a sparse system by a direct method, one must use both the factorization and the solve subroutines. The factorization subroutine should be followed by the corresponding solve subroutine; that is, the output from the factorization subroutine should be used as input to the solve subroutine. The IBM's `ESSL` library also provides subroutines for direct solvers like `dgsf` (General Sparse Matrix Factorization Using Storage by Indices, Rows, or Columns) and `dgss` (General Sparse Matrix or Its Transpose Solve Using Storage by Indices, Rows, or Columns). However, in the current study we use a more advanced library called `UMFPACK`, which has a set of subroutines for solving sparse linear systems directly using the unsymmetric multi-frontal method and direct sparse `LU` factorization. It is a very robust and highly optimized direct serial solver suited for small to medium-large-scale problems. The solver step can also be easily linked to other advanced external libraries like `MUMPS`, `PARDISO`, `PETSc` etc. which may have serial/parallel, direct/iterative solvers for Fortran, C/C++.

2.7 Notation, finite element formulation and numerical results

2.7.1 Notation

Before proceeding to describe the high-order finite element problems utilized in this research, we find it prudent to introduce some standard notation. We assume that Ω is an open bounded subset of $\mathbb{R}^{nd}$, where nd denotes the number of spatial dimensions. The boundary of Ω is denoted by $\Gamma = \partial\Omega = \bar{\Omega} - \Omega$, where $\bar{\Omega}$ represents the closure of Ω. A typical point belonging to $\bar{\Omega}$ is denoted as $\mathbf{x}$. We employ the customary designations for the Sobolev spaces $H^s(\Omega)$ and $H^s(\Gamma)$ where $s \geqslant 0$. The corresponding norms are given as $\| \cdot \|_{\Omega,s}$ and $\| \cdot \|_{\Gamma,s}$. Likewise the inner products associated with these spaces are denoted as $(\cdot \,, \cdot)_{\Omega,s}$ and $(\cdot \,, \cdot)_{\Gamma,s}$ respectively. The product spaces $\mathbf{H}^s(\Omega) = [H^s(\Omega)]^{nd}$ are constructed in the usual way.

2.7.2 Weak formulations

In this research we are concerned with the variational or weak formulation of boundary and initial boundary-value problems. We construct these weak formulations based upon either the classical weak form Galerkin formulation and also through the use of the least-squares method. Weak formulations typically involve integral statements over Ω and Γ that are in a *generalized sense* equivalent to the original set of partial differential equations and natural boundary conditions associated with a given system. Such problems may be stated as follows: find $\mathbf{u} \in \mathcal{V}$ such that

$$\mathcal{B}(\mathbf{w}, \mathbf{u}) = \mathcal{F}(\mathbf{w}) \quad \forall\, \mathbf{w} \in \mathcal{W} \tag{2.36}$$

where $\mathcal{B}(\mathbf{w}, \mathbf{u})$ is a bilinear form, $\mathcal{F}(\mathbf{w})$ is a linear form, and $\mathcal{V}$ and $\mathcal{W}$ are appropriate function spaces (e.g., the Sobolev space $\mathbf{H}^1(\Omega)$). The quantity $\mathbf{u}$ represents the set of independent variables (associated with the variational boundary value problem), and

w represents the corresponding weighting or test function. Unlike classical solutions that are defined unambiguously point-wise, weak solutions exist with respect to test functions and are therefore understood in the context of distributions.

2.7.3 Abstract least-squares method

We consider the following abstract boundary-value problem:

$$\mathcal{L}(\mathbf{u}) = \mathbf{f} \qquad \text{in } \Omega \tag{2.37a}$$

$$\mathbf{u} = \mathbf{u}^{\mathrm{p}} \qquad \text{on } \Gamma^{\mathrm{D}} \tag{2.37b}$$

$$g(\mathbf{u}) = \mathbf{h} \qquad \text{on } \Gamma^{\mathrm{N}} \tag{2.37c}$$

where $\mathcal{L}$ is a nonlinear first-order spatial partial differential operator, **u** is the independent variable, **f** is the forcing function and $\mathbf{u}^{\mathrm{p}}$ is the prescribed essential boundary condition. The Neuman boundary condition is expressed in terms of the operator g and the prescribed function **h**. We assume that the function g is linear in **u** and that the problem is well-posed.

In the least-squares method, we construct an unconstrained convex least-squares functional $\mathcal{J}$ whose minimizer corresponds with the solution of equation (B.2). To maintain practicality [9, 10, 76, 78, 80] in the numerical implementation, we construct the least-squares functional in terms of the sum of the squares of the L_2 norms of the abstract equation residuals

$$\mathcal{J}(\mathbf{u};\mathbf{f},\mathbf{h}) = \frac{1}{2}\Big(\|\mathcal{L}(\mathbf{u}) - \mathbf{f}\|^2_{\Omega,0} + \|g(\mathbf{u}) - \mathbf{h}\|^2_{\Gamma^{\mathrm{N}},0}\Big) \tag{2.38}$$

The abstract minimization principle associated with the least-squares method may be stated as follows: find $\mathbf{u} \in \mathcal{V}$ such that

$$\mathcal{J}(\mathbf{u}; \mathbf{f}, \mathbf{h}) \leqslant \mathcal{J}(\tilde{\mathbf{u}}; \mathbf{f}, \mathbf{h}) \quad \text{for all } \tilde{\mathbf{u}} \in \mathcal{V} \tag{2.39}$$

where the function space $\mathcal{V}$ is defined as

$$\mathcal{V} = \left\{ \mathbf{u} : \mathbf{u} \in \mathbf{H}^1(\Omega),\ \mathbf{u} = \mathbf{u}^{\mathrm{p}} \text{ on } \Gamma^{\mathrm{D}} \right\} \tag{2.40}$$

The necessary condition for minimization requires that the first variation of $\mathcal{J}(\mathbf{u}; \mathbf{f}, \mathbf{h})$, denoted as $\mathcal{G}(\mathbf{u}, \delta\mathbf{u})$, be identically zero. Carrying out the minimization principle with the aid of the Gâteaux derivative yields

$$\begin{aligned} \mathcal{G}(\mathbf{u}, \delta\mathbf{u}) = \delta\mathcal{J}(\mathbf{u}, \delta\mathbf{u}; \mathbf{f}, \mathbf{h}) &= \frac{d}{d\varepsilon}\mathcal{J}(\mathbf{u} + \varepsilon\delta\mathbf{u}; \mathbf{f}, \mathbf{h})\Big|_{\varepsilon=0} \\ &= (\nabla\mathcal{L}(\mathbf{u}) \cdot \delta\mathbf{u}, \mathcal{L}(\mathbf{u}) - \mathbf{f})_{\Omega,0} + (g(\delta\mathbf{u}), g(\mathbf{u}) - \mathbf{h})_{\Gamma^{\mathrm{N}},0} = 0 \end{aligned} \tag{2.41}$$

where the symbolic derivative (or gradient) operator ∇ acts with respect to the independent variable $\mathbf{u}$. The linear vector space of kinematically admissible variations $\mathcal{W}$ is of the form

$$\mathcal{W} = \left\{ \delta\mathbf{u} : \delta\mathbf{u} \in \mathbf{H}^1(\Omega),\ \delta\mathbf{u} = \mathbf{0} \text{ on } \Gamma^{\mathrm{D}} \right\} \tag{2.42}$$

The least-squares based based weak formulation, therefore, is to find $\mathbf{u} \in \mathcal{V}$ such that equation (2.41) holds for all $\delta\mathbf{u} \in \mathcal{W}$.

2.7.4 Least-squares finite element formulation of a boundary-value problem

In the following example all the above strategies like the higher-order mesh generation, ‘sidesets’ and ‘nodesets’ generation for *essential* type of boundary conditions, line integrals for *natural* type of boundary conditions, schur complement method and others are put to test. We attempt to solve the isothermal, incompressible, steady-state Navier–Stokes flow over a cylinder in two-dimensions.

The dimensionless Navier–Stokes equation in terms of the primitive variables of pressure p and velocity $\mathbf{u}$ can be written as

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{2.43a}$$

$$(\mathbf{u} \cdot \nabla)\, \mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}} \nabla \cdot \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\mathrm{T}} \right] = \mathbf{f} \qquad \text{in } \Omega \tag{2.43b}$$

$$\mathbf{u} = \mathbf{u}^{\mathrm{P}} \qquad \text{on } \Gamma_u \tag{2.43c}$$

$$\hat{\mathbf{n}} \cdot \underline{\boldsymbol{\sigma}} = \mathbf{t}^{\mathrm{P}} \qquad \text{on } \Gamma_t \tag{2.43d}$$

where Re is the Reynolds number, $\mathbf{f}$ is the dimensionless resultant body force due to agents like gravity, magnetic effects etc., $\mathbf{u}^{\mathrm{P}}$ is the dimensionless prescribed velocity on the boundary Γ_u, $\mathbf{t}^{\mathrm{P}}$ is the dimensionless prescribed traction on the boundary Γ_t, $\hat{\mathbf{n}}$ is the outward unit normal to the boundary Γ_t and $\underline{\boldsymbol{\sigma}}$ is the total stress tensor (Cauchy stress). It must be noted that the parts of boundary with prescribed velocities and tractions satisfy $\Gamma = \Gamma_u \cup \Gamma_t$ and $\emptyset = \Gamma_u \cap \Gamma_t$. The Cauchy stress can be represented in terms of primitive variables from the constitutive relation as below

$$\underline{\boldsymbol{\sigma}} = -p\underline{\mathbf{I}} + \frac{1}{\mathrm{Re}} \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\mathrm{T}} \right] \tag{2.44}$$

To develop the finite element equations, the least-squares formulation is used instead of the traditional weak-form Galerkin formulation. For more on the least-squares formulations and its advantages see Section 5. The second-order system of equations (5.1a) - (5.1d) are recast as a first-order system by introducing stress tensor, $\underline{\mathbf{T}} = \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\mathrm{T}} \right]$, as an auxiliary variable. Note, the stress tensor is symmetric. The stress-based first-order system can be written as

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{2.45a}$$

$$(\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}} \nabla \cdot \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\mathrm{T}} \right] = \mathbf{f} \qquad \text{in } \Omega \tag{2.45b}$$

$$\underline{\mathbf{T}} = \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\mathrm{T}} \right] \qquad \text{in } \Omega \tag{2.45c}$$

$$\mathbf{u} = \mathbf{u}^{\mathrm{P}} \qquad \text{on } \Gamma_u \tag{2.45d}$$

$$\hat{\mathbf{n}} \cdot \underline{\mathbf{T}} = \underline{\mathbf{T}}^{\mathrm{P}} \qquad \text{on } \Gamma_{\mathrm{T}} \tag{2.45e}$$

For the above stress-based first-order system the outflow boundary conditions can be imposed in a strong sense, using the components of symmetric stress tensor or in a weak sense, through the least-squares functional as done in Section 5. In the present work it is applied in a weak sense and is explained in the next section. The least-squares functional for the above system of equations can be setup by taking the sum of the squares of the residual equations in L_2-normed function space as

$$\begin{aligned}\mathcal{J}\left(p, \mathbf{u}, \underline{\mathbf{T}}; \mathbf{f}\right) = \tfrac{1}{2}\Bigg(& \left\|\nabla \cdot \mathbf{u}\right\|_0^2 + \left\|(\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p - \tfrac{1}{\mathrm{Re}} \nabla \cdot \underline{\mathbf{T}} - \mathbf{f}\right\|_0^2 + \\ & \left\|\underline{\mathbf{T}} - \left[(\nabla \mathbf{u}) + (\nabla \mathbf{u})^{\mathrm{T}} \right]\right\|_0^2 \Bigg)\end{aligned} \tag{2.46}$$

The least-squares minimization problem is to find $p(x), \mathbf{u}(x), \underline{\mathbf{T}}(x) \ni \mathcal{J}\left(p, \mathbf{u}, \underline{\mathbf{T}}; \mathbf{f}\right) \leq \mathcal{J}\left(\tilde{p}, \tilde{\mathbf{u}}, \tilde{\underline{\mathbf{T}}}; \mathbf{f}\right) \forall \left(\tilde{p}, \tilde{\mathbf{u}}, \tilde{\underline{\mathbf{T}}}; \mathbf{f}\right) \in \mathrm{X}$, i.e., seek $(p,\ \mathbf{u}, \underline{\mathbf{T}})$ suchthat $\mathcal{J}\left(p, \mathbf{u}, \underline{\mathbf{T}}; \mathbf{f}\right)$ is minimized over X, where X is

$$\mathrm{X} = \left\{ (p,\ \mathbf{u},\ \underline{\mathbf{T}}) \in \mathrm{H}_0^1(\Omega) \times \mathrm{H}^1(\Omega) \times \mathrm{H}^1(\Omega) \right\} \tag{2.47}$$

After suitable linearizations by Newton's Method, the variational problem corresponding to above least-squares functional can be written as

$$\mathcal{B}\left(\left(\tilde{p}, \tilde{\mathbf{u}}, \underline{\tilde{\mathbf{T}}}\right), \left(p, \mathbf{u}, \underline{\mathbf{T}}\right)\right) = \mathcal{F}\left(\tilde{p}, \tilde{\mathbf{u}}, \underline{\tilde{\mathbf{T}}}\right) \forall \left(\tilde{p}, \tilde{\mathbf{u}}, \underline{\tilde{\mathbf{T}}}\right) \in \mathrm{X} \tag{2.48}$$

where the bi-linear and linear forms are given as

$$\begin{aligned}\mathcal{B}\left(\left(\tilde{p}, \tilde{\mathbf{u}}, \underline{\tilde{\mathbf{T}}}\right), \left(p, \mathbf{u}, \underline{\mathbf{T}}\right)\right) = \int_{\Omega} \Bigg\{ & \left(\left(\mathbf{u}_0 \cdot \nabla\right)\tilde{\mathbf{u}} + \left(\tilde{\mathbf{u}} \cdot \nabla\right)\mathbf{u}_0 + \nabla\tilde{p} - \tfrac{1}{\mathrm{Re}}\nabla \cdot \underline{\tilde{\mathbf{T}}}\right) \cdot \\ & \left(\left(\mathbf{u}_0 \cdot \nabla\right)\mathbf{u} + \left(\mathbf{u} \cdot \nabla\right)\mathbf{u}_0 + \nabla p - \tfrac{1}{\mathrm{Re}}\nabla \cdot \underline{\mathbf{T}}\right) + \\ & \left(\left(\nabla \cdot \tilde{\mathbf{u}}\right) \cdot \left(\nabla \cdot \mathbf{u}\right)\right) + \left(\underline{\tilde{\mathbf{T}}} - \left[\left(\nabla\tilde{\mathbf{u}}\right) + \left(\nabla\tilde{\mathbf{u}}\right)^{\mathrm{T}}\right]\right) \cdot \\ & \left(\underline{\mathbf{T}} - \left[\left(\nabla\mathbf{u}\right) + \left(\nabla\mathbf{u}\right)^{\mathrm{T}}\right]\right) \Bigg\} d\Omega\end{aligned} \tag{2.49}$$

$$\mathcal{F}\left(\tilde{p}, \tilde{\mathbf{u}}, \underline{\tilde{\mathbf{T}}}\right) = \int_{\Omega} \left(\left(\mathbf{u}_0 \cdot \nabla\right)\mathbf{u}_0 + \mathbf{f}\right) \cdot \left(\left(\mathbf{u}_0 \cdot \nabla\right)\tilde{\mathbf{u}} + \left(\tilde{\mathbf{u}} \cdot \nabla\right)\mathbf{u}_0 + \nabla\tilde{p} - \frac{1}{\mathrm{Re}}\nabla \cdot \underline{\tilde{\mathbf{T}}}\right) d\Omega \tag{2.50}$$

The above bi-linear and linear forms can be explicitly written as

$$\begin{bmatrix} \mathrm{K}^{11} & \mathrm{K}^{12} & \mathrm{K}^{13} & \mathrm{K}^{14} & \mathrm{K}^{15} & \mathrm{K}^{16} \\ \mathrm{K}^{21} & \mathrm{K}^{22} & \mathrm{K}^{23} & \mathrm{K}^{24} & \mathrm{K}^{25} & \mathrm{K}^{26} \\ \mathrm{K}^{31} & \mathrm{K}^{32} & \mathrm{K}^{33} & \mathrm{K}^{34} & \mathrm{K}^{35} & \mathrm{K}^{36} \\ \mathrm{K}^{41} & \mathrm{K}^{42} & \mathrm{K}^{43} & \mathrm{K}^{44} & \mathrm{K}^{45} & \mathrm{K}^{46} \\ \mathrm{K}^{51} & \mathrm{K}^{52} & \mathrm{K}^{53} & \mathrm{K}^{54} & \mathrm{K}^{55} & \mathrm{K}^{56} \\ \mathrm{K}^{61} & \mathrm{K}^{62} & \mathrm{K}^{63} & \mathrm{K}^{64} & \mathrm{K}^{65} & \mathrm{K}^{66} \end{bmatrix} \begin{Bmatrix} p \\ u_x \\ u_y \\ T_{xx} \\ T_{xy} \\ T_{yy} \end{Bmatrix} = \begin{Bmatrix} \mathrm{F}^{1} \\ \mathrm{F}^{2} \\ \mathrm{F}^{3} \\ \mathrm{F}^{4} \\ \mathrm{F}^{5} \\ \mathrm{F}^{6} \end{Bmatrix} \tag{2.51}$$

2.7.5 Outflow boundary conditions

Here we discuss in detail about the outflow boundary condition, its contribution to the least-squares functional and a method to evaluate it. The outflow boundary conditions is typically given by

$$\hat{\mathbf{t}} = \hat{\mathbf{n}} \cdot \underline{\tilde{\boldsymbol{\sigma}}} = 0 \tag{2.52}$$

where $\underline{\tilde{\boldsymbol{\sigma}}}$ is the pseudo Cauchy stress tensor defined as $\underline{\tilde{\boldsymbol{\sigma}}} = -p\underline{\mathbf{I}} + (1/\mathrm{Re})\,\nabla\mathbf{u}$. In two-dimensions cartesian coordinate system the Eq. (5.13) becomes

$$\begin{aligned}
\hat{\mathbf{t}} &= \left(-p\hat{n}_i\delta_{ij} + \tfrac{1}{\mathrm{Re}}\hat{n}_i\tfrac{\partial u_j}{\partial x_i}\right)\mathbf{e}_j \\
\hat{t}_x &= -p\hat{n}_x + \tfrac{1}{\mathrm{Re}}\left(\hat{n}_x\tfrac{\partial u_x}{\partial x} + \hat{n}_y\tfrac{\partial u_x}{\partial y}\right) \\
\hat{t}_y &= -p\hat{n}_y + \tfrac{1}{\mathrm{Re}}\left(\hat{n}_x\tfrac{\partial u_y}{\partial x} + \hat{n}_y\tfrac{\partial u_y}{\partial y}\right)
\end{aligned} \tag{2.53}$$

When the outflow boundary condition is applied in a weak sense, the least-squares functional in Eq. (2.46) needs to be modified as (see the underlined term)

$$\begin{aligned}
\mathcal{J}\left(p, \mathbf{u}, \underline{\mathbf{T}}; \mathbf{f}\right) = \tfrac{1}{2}\Bigg(&\left\|\nabla\cdot\mathbf{u}\right\|_0^2 + \left\|\left(\mathbf{u}\cdot\nabla\right)\mathbf{u} + \nabla p - \tfrac{1}{\mathrm{Re}}\nabla\cdot\underline{\mathbf{T}} - \mathbf{f}\right\|_0^2 + \\
&\left\|\underline{\mathbf{T}} - \left[\left(\nabla\mathbf{u}\right) + \left(\nabla\mathbf{u}\right)^{\mathrm{T}}\right]\right\|_0^2 + \underline{\left\|\hat{\mathbf{t}} - \hat{\mathbf{n}}\cdot\underline{\tilde{\boldsymbol{\sigma}}}\right\|_{0,\Gamma_{\mathrm{outflow}}}^2}\Bigg)
\end{aligned} \tag{2.54}$$

Using the Eq. (2.53) the contribution of the outflow boundary condition to the above least-squares functional can be written as

$$\mathcal{J}_{\mathrm{out}}\left(p, \mathbf{u}\right) = \frac{1}{2}\int_{\Gamma_{\mathrm{out}}}\left\|\hat{\mathbf{t}} + \left(p\hat{\mathbf{n}} - \frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\mathbf{u}\right)\right\|_0^2 ds \tag{2.55}$$

The first variation of the above functional becomes

$$\delta\mathcal{J}_{\text{out}} = \int_{\Gamma_{\text{out}}} \left[\left(\delta p\hat{\mathbf{n}} - \frac{1}{\text{Re}}\hat{\mathbf{n}}\cdot\nabla\delta\mathbf{u}\right)\cdot\left(p\hat{\mathbf{n}} - \frac{1}{\text{Re}}\hat{\mathbf{n}}\cdot\nabla\mathbf{u}\right) + \left(\delta p\hat{\mathbf{n}} - \frac{1}{\text{Re}}\hat{\mathbf{n}}\cdot\nabla\delta\mathbf{u}\right)\cdot\hat{\mathbf{t}}\right] ds \tag{2.56}$$

The terms of the above equation can be separated into bi-linear and linear forms as

$$\mathcal{B}_{\text{out}}\Big((\tilde{p},\tilde{\mathbf{u}}),(p,\mathbf{u})\Big) = \mathcal{F}_{\text{out}}\Big((\tilde{p},\tilde{\mathbf{u}})\Big) \tag{2.57}$$

which can be explicitly expressed as

$$\begin{aligned}\mathcal{B}_{\text{out}}\Big((\tilde{p},\tilde{\mathbf{u}}),(p,\mathbf{u})\Big) &= \int_{\Gamma_{\text{out}}} \Big(\tilde{p}\hat{\mathbf{n}} - \tfrac{1}{\text{Re}}\hat{\mathbf{n}}\cdot\nabla\tilde{u}\Big)\cdot\Big(p\hat{\mathbf{n}} - \tfrac{1}{\text{Re}}\hat{\mathbf{n}}\cdot\nabla\mathbf{u}\Big)ds\\ \mathcal{F}_{\text{out}}\Big((\tilde{p},\tilde{\mathbf{u}})\Big) &= \int_{\Gamma_{\text{out}}} \Big(\tfrac{1}{\text{Re}}\hat{\mathbf{n}}\cdot\nabla\tilde{\mathbf{u}} - \tilde{p}\hat{\mathbf{n}}\Big)\cdot\hat{\mathbf{t}}ds\end{aligned} \tag{2.58}$$

In two-dimensions, the above becomes

$$\begin{aligned}\mathcal{B}_{\text{out}}\Big((\tilde{p},\tilde{u}),(p,u)\Big) &= \int_{\Gamma_{\text{out}}} \Big\{\Big[\tilde{p}\hat{n}_x - \tfrac{1}{\text{Re}}\Big(\hat{n}_x\tfrac{\partial\tilde{u}_x}{\partial x} + \hat{n}_y\tfrac{\partial\tilde{u}_x}{\partial y}\Big)\Big]\\ &\qquad\Big[p\hat{n}_x - \tfrac{1}{\text{Re}}\Big(\hat{n}_x\tfrac{\partial u_x}{\partial x} + \hat{n}_y\tfrac{\partial u_x}{\partial y}\Big)\Big] +\\ &\qquad\Big[\tilde{p}\hat{n}_y - \tfrac{1}{\text{Re}}\Big(\hat{n}_x\tfrac{\partial\tilde{u}_y}{\partial x} + \hat{n}_y\tfrac{\partial\tilde{u}_y}{\partial y}\Big)\Big]\\ &\qquad\Big[p\hat{n}_y - \tfrac{1}{\text{Re}}\Big(\hat{n}_x\tfrac{\partial u_y}{\partial x} + \hat{n}_y\tfrac{\partial u_y}{\partial y}\Big)\Big]\Big\}\, ds\\ \mathcal{F}_{\text{out}}\Big((\tilde{p},\tilde{u})\Big) &= \int_{\Gamma_{\text{out}}} \Big\{\hat{t}_x\Big[\tfrac{1}{\text{Re}}\Big(\hat{n}_x\tfrac{\partial\tilde{u}_x}{\partial x} + \hat{n}_y\tfrac{\partial\tilde{u}_x}{\partial y}\Big) - \tilde{p}\hat{n}_x\Big] +\\ &\qquad\hat{t}_y\Big[\tfrac{1}{\text{Re}}\Big(\hat{n}_x\tfrac{\partial\tilde{u}_y}{\partial x} + \hat{n}_y\tfrac{\partial\tilde{u}_y}{\partial y}\Big) - \tilde{p}\hat{n}_y\Big]\Big\}\, ds\end{aligned} \tag{2.59}$$

The additional components from this to the finite element coefficient matrices and force vectors of Eq. (5.10) are given by

$$
\begin{aligned}
&K_{ij}^{11} = \int_{\Gamma_{\text{out}}} \psi_i\psi_j ds, && K_{ij}^{12} = -\int_{\Gamma_{\text{out}}} \tfrac{n_x}{\text{Re}}\psi_i\mathcal{N}_j ds \\
&K_{ij}^{13} = -\int_{\Gamma_{\text{out}}} \tfrac{n_y}{\text{Re}}\psi_i\mathcal{N}_j ds && \\
&K_{ij}^{21} = -\int_{\Gamma_{\text{out}}} \tfrac{n_x}{\text{Re}}\mathcal{N}_i\psi_j ds, && K_{ij}^{22} = \int_{\Gamma_{\text{out}}} \tfrac{1}{\text{Re}^2}\mathcal{N}_i\mathcal{N}_j ds \\
&K_{ij}^{31} = -\int_{\Gamma_{\text{out}}} \tfrac{n_y}{\text{Re}}\mathcal{N}_j\psi_j ds, && K_{ij}^{33} = \int_{\Gamma_{\text{out}}} \tfrac{1}{\text{Re}^2}\mathcal{N}_i\mathcal{N}_j ds \\
&F_i^{1} = -\int_{\Gamma_{\text{out}}} \left(\hat{t}_x\hat{n}_x + \tilde{t}_y\hat{n}_y\right)\psi_i ds, && F_i^{2} = \int_{\Gamma_{\text{out}}} \tfrac{\hat{t}_x}{\text{Re}}\mathcal{N}_i ds \\
&F_i^{3} = \int_{\Gamma_{\text{out}}} \tfrac{\hat{t}_y}{\text{Re}}\mathcal{N}_i ds && \\
&\qquad\qquad \text{where } \mathcal{N}_i = \hat{n}_x\tfrac{\partial\psi_i}{\partial x} + \hat{n}_y\tfrac{\partial\psi_i}{\partial y}
\end{aligned}
\tag{2.60}
$$

Note, the line integration techniques discussed in section D are used to evaluate these integrals and the values of $\hat{t}_x$ and $\hat{t}_y$ are specified in the input file as shown in Fig. 2.14.

2.7.6 Numerical results

To test all the above strategies a standard benchmark problem of flow past a circular cylinder in a long channel is considered. The cylinder is of unit diameter and is at the center of the finite domain $\Omega = [-15.5, +25.5] \times [-20.5, +20.5]$ as shown in Fig. 2.10(a). The mesh has 501 quadrilateral finite elements. For this mesh the horizontal velocity is specified as 1.0 at the inflow (left), top and bottom boundaries and the vertical velocity as 0 on these three boundaries. A no-slip boundary condition is imposed on the surface of the cylinder. And the outflow boundary condition is imposed in a weak-sense through the least-squares functional on the outflow (right) boundary of the domain. The value of Reynolds number and the placement of the computational boundaries in relation to the cylinder are critical as the flow pattern depends on them. At low Reynolds number ($5 < \text{Re} < 46.1$), the flow of an incompressible, newtonian fluid past a circular cylinder is stationary and its pattern is characterized by a pair of symmetric vortices on the downstream of the cylinder. The size of these standing vortex layers is proportional to the Reynolds number. As the Reynolds number reaches a critical value (Re $>$ 46.1), the standing vortex layers become unstable and flow can no longer be treated as two-dimension problem. A Reynolds number of Re $= 40$ is used for all the cases in this work. In finite element program we utilize the OpenMP parallel paradigm to setup the global linear system of equations in compressed sparse column (CSC) or compressed sparse row (CSR) formats for the statically condensed mesh [74]. The program can easily link to any external solver libraries like `UMFPACK`, `PARDISO`, `MUMPS`, `ESSL` etc. and the global system of equations for the statically condensed mesh can be solved using a variety of direct/iterative, serial/parallel solvers from any of these libraries. The current test problem is solved using `UMFPACK`, which is a serial direct solver on a cluster at the Texas A&M Supercomputing Facility. The hardware specifications are IBM Cluster-1600 with IBM's 1.9 GHz RISC Power5+ processors. Each node is a symmetric multi-processor (SMP) system with 16 processors and 25 GB of usable shared memory.

In Fig. 2.21(a), the crown region of cylinder is shown in yellow for a polynomial of order $p = 2$. This region has 8 elements numbered 101, 100, 99, 98, 37, 38, 39, 40. To assess the conservation of continuity equation on this region, the absolute value of the below

$$\mathcal{Q} = \sum \mathcal{Q}^e = \oint\limits_{\Gamma^e} \hat{\mathbf{n}} \cdot \mathbf{u} \, \partial\Gamma^e \tag{2.61}$$

is calculated for each of the elements in the crown region. Note, the above equation is obtained by applying the divergence theorem to the continuity equation and is evaluated using the line integral methods discussed in section D. The problem is solved with different polynomial orders of $p = 3, 5 \, \text{and} \, 7$ each with 4659, 12775 and 24899 nodes respectively. In Fig. 2.21(b), the values of total solution error ε_{all}, errors in pressure ε_p, horizontal velocity ε_{u_x}, vertical velocity ε_{u_y}, the least-squares functional $\mathcal{J}$, and the volumetric flow rate imbalance $\mathcal{Q}$ are plotted for $p = 3, 5\,\text{and}\,7$. Note, all these values decrease exponentially with increase in the polynomial order, this is called spectral convergence. This is another advantage of the least-squares method coupled with higher-order spectral/*hp* basis functions. In Figs. 2.21(c) and 2.21(d), the pressure and vertical velocity contours are shown along with the streamtraces of symmetric vortices downstream the cylinder.

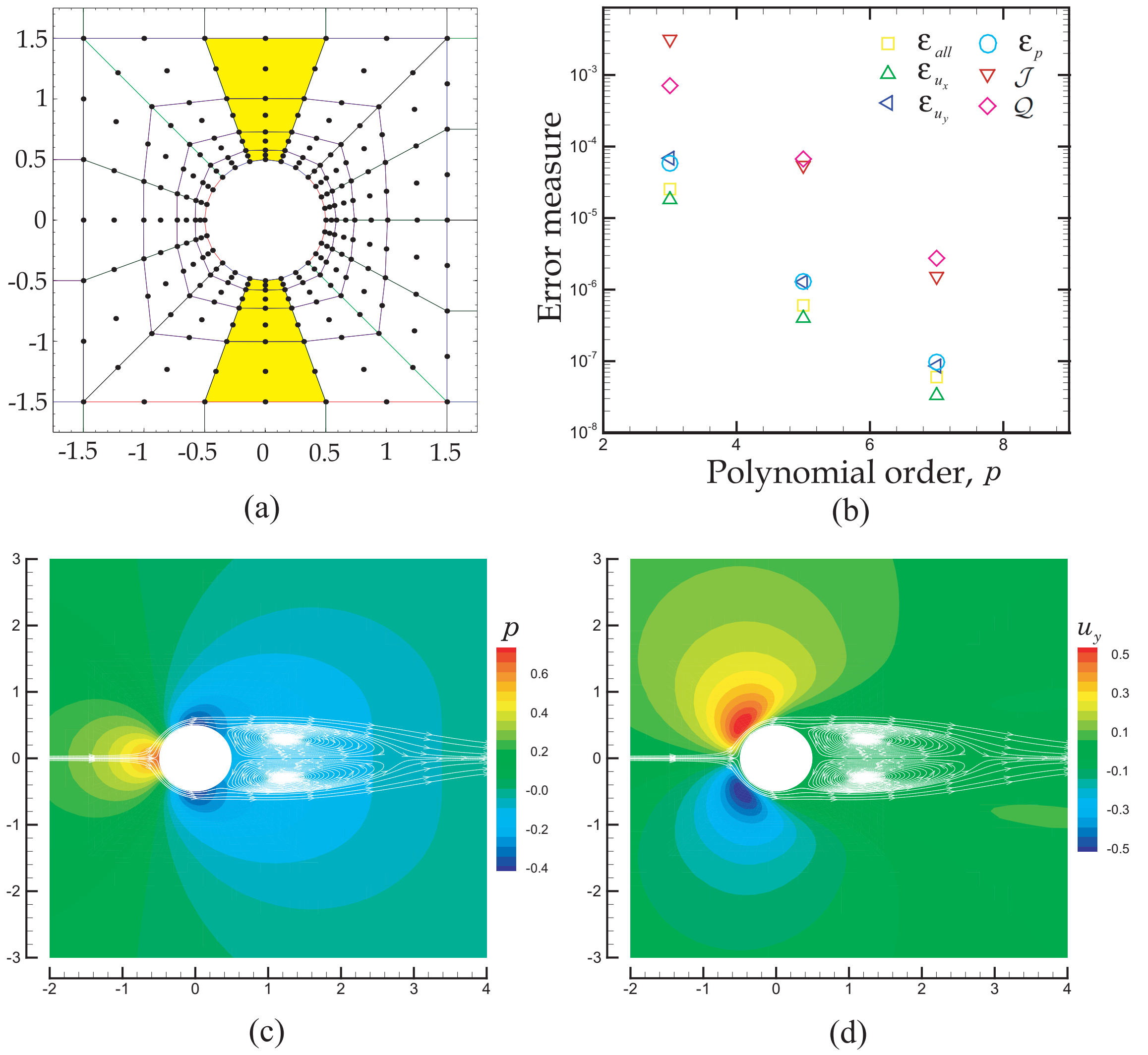


Figure 2.21: (a) Crown region (shown for mesh with $p = 2$) (b) Various error measures (c) Pressure contours and (d) Vertical velocity contours.

3. HIGHER-ORDER SPECTRAL/*HP* FINITE ELEMENT FORMULATIONS OF BEAMS WITH VISCOELASTICITY*

In this section, we develop a finite element model for efficient nonlinear analysis of the mechanical response of viscoelastic beams is presented. The principle of virtual work is utilized in conjunction with the third-order beam theory to develop displacement-based, weak-form Galerkin finite element model for both quasi-static and fully-transient analysis. The displacement field is assumed such that the third-order beam theory admits C^0 Lagrange interpolation of all dependent variables and the constitutive equation can be that of an isotropic material. Also, higher-order interpolation functions of spectral/*hp* type are employed to efficiently eliminate numerical locking. The mechanical properties are considered to be linear viscoelastic while the beam may undergo *von Kármán* nonlinear geometric deformations. The constitutive equations are modeled using the Prony exponential series with general *n-parameter* Kelvin chain as its mechanical analogy for quasi-static cases and a simple two-element Maxwell model for dynamic cases. The fully discretized finite element equations are obtained by approximating the convolution integrals from the viscous part of the constitutive relations using a trapezoidal rule. A two-point recurrence scheme is developed that uses the approximation of relaxation moduli with the Prony series. This necessitates the data storage for only the last time step and not for the entire deformation history.

*Part of the numerical results reported in this section appear in the article "A spectral/*hp* nonlinear finite element analysis of higher-order beam theory with viscoelasticity" by V. P. Vallala, G. S. Payette and J. N. Reddy, *Int. J. Appl. Mech.*, vol. 4, pp. 43–57, 2012. Copyright (2012) World Scientific.

3.1 Introduction

The phrase "viscoelastic behavior" refers to a spectrum of mechanical characteristics that are not exhibited either by pure viscous or pure elastic bodies. It is a combination of these two extremes in relative proportions. Many engineering materials (like bio-polymers, alloys, amorphous glass, metals at high temperatures) fall in this continuous spectrum of all possible properties from one extreme to the other, which makes it more important to study these materials. The theory of viscoelastic behavior is long been established, the reader can refer to [58] for a materials perspective and standard texts of Lockett [60], Flügge [31], Christensen [23], Findley [30] and Reddy [93] for a continuum perspective. Within the continuum purview there are analytical methods like integral transforms, Laplace transforms and correspondence principle, etc. to study the mechanical response of structures made of viscoelastic materials. But as with many analytical methods they are limited to simple cases of geometry and loading. In such scenarios numerical techniques like the finite element method becomes very useful in testing and predicting the properties of a material without actually fabricating them. Numerical methods can be used to obtain approximate solution with desired accuracy.

Many researchers have used the finite element method to study viscoelastic materials [112, 69, 40, 35, 36, 96]. The main difficulty with the viscoelastic finite element models is the approximation of convolution integrals that come from viscoelastic constitutive equations. Most of these finite element models try to circumvent the problem with the time dependent convolution integrals by transforming them to a set of discrete algebraic equations in space. Taylor and Oden [112, 69] used recurrence relations such that only the deformation history from last few iterations is needed to be stored instead of entire deformation from beginning. The other popular methods in literature are Laplace transform approach by [22, 2, 113], Fourier transform method by [21], anelastic displacement formulation by [115, 70], and Golla-Hughes-McTavish (GHM) method by [62, 63, 7, 6]. For more on this refer [72].

Most of the viscoelastic finite element models in the literature use lower-order theories for beams, plates or shells. Johnson et al. [50] reviewed the history integral form of Maxwell solid and derived a new differential constitutive law for it. The Prony series is used to express relaxation moduli and, as a result, the time dependent linear viscoelastic constitutive equations were expressed as set of ordinary differential equations in terms of displacements to obtain the finite element stiffness coefficients. Many of these finite element formulations are restricted to small deformations. As a result they don't account for geometric nonlinearity effect at large loads. In the present study we present quasi-static and fully-transient spectral/*hp* finite element analysis for linear viscoelastic beams based on higher-order shear deformation theory with the *von Kármán* nonlinear strains [88].

The section is organized as follows. We first review the kinematic assumptions that form the basis for each of the three beam theories considered in the present study. An effective strain tensor (a simplification of the Green–Lagrange strain) is then introduced along with the assumed linear viscoelastic constitutive model. The finite element formulation for each beam theory is then derived from the principle of virtual displacements, or equivalently through the use of the weak-form Galerkin procedure. In the fully discretized finite element models, the convolution integrals (emanating from the viscoelastic constitutive equations) are temporally approximated using the trapezoidal rule in conjunction with a two-point recurrence formula. We conclude the section by presenting numerical results for quasi-static and fully transient verification benchmark problems. We shown that all forms of locking may be avoided through the use of either: (a) low-order finite elements with selective employment of full and reduced numerical integration strategies or (b) fully integrated finite elements constructed from high-order polynomial interpolation functions of both Lagrange and Hermite type. The main ideas and some numerical results are taken from our published journals Ref. [118, 120].

3.2 Assumptions and strain-displacement relations

For the deformation of the beam the standard strain measure of *Green–Lagrange strain* **E**, is widely used in solid mechanics. The non-zero components can be expressed as

$$E_{XX} = \frac{\partial u}{\partial X} + \frac{1}{2}\left[\underline{\left(\frac{\partial u}{\partial X}\right)^2} + \left(\frac{\partial w}{\partial X}\right)^2\right] \tag{3.1}$$

$$E_{XZ} = \frac{1}{2}\left(\frac{\partial u}{\partial Z} + \frac{\partial w}{\partial X} + \underline{\frac{\partial u}{\partial X}\frac{\partial u}{\partial Z}}\right) \text{ and } E_{ZZ} = \frac{1}{2}\underline{\left(\frac{\partial u}{\partial Z}\right)^2} \tag{3.2}$$

Under the assumptions of large transverse displacements, small strains and small to moderate rotations ($< 15°$), we can omit the underlined terms from the above equation. Thus the Green–Lagrange strain tensor becomes the *reduced strain tensor* $\mathbf{E} \approx \boldsymbol{\varepsilon}$ [Reddy, 2004]. The nonzero components of this reduced Green–Lagrange strain tensor are given by

$$\begin{aligned} \varepsilon_{xx} &= \frac{\partial u}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2 \\ \varepsilon_{xz} &= \frac{1}{2}\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right) \end{aligned} \tag{3.3}$$

For the continuous deformation of beam we use *material* or *Lagrangian* description. All quantities that describe the motion, like displacements, rotations are measured in the initial or undeformed configuration. Hence the coordinates (x,y,z) used in Eq.(3.3) above and throughout this paper represent the material coordinates. The equations of motion are derived from the principle of virtual work, which is expressed in terms of quantities (like virtual displacements) measured in the undeformed configuration. Since all the displacements and strains are measured in the undeformed configuration we use second Piola–Kirchhoff stress tensor, denoted by $\boldsymbol{\sigma}$, for the stress measure.

3.3 A review of higher-order beam theories

Beam theories based on the assumed form of the displacement field are most popular. In these theories, the displacements are expanded in increasing powers of the thickness (or height) coordinate. The word "order" refers to the power of the thickness coordinate in the power series expansion of the displacement field. To make this clear, a review of beam theories is presented below.

The simplest and oldest beam theory is the Bernoulli–Euler beam theory (BET) Kirchhoff [53]. It is based on the kinematic assumptions that straight lines perpendicular to the plane of the undeformed beam remain straight and inextensible, and rotate such that they always remain perpendicular the axis of the beam after deformation. These assumptions, known as the *Kirchhoff hypothesis*, amounts to neglecting both transverse shear and transverse normal strains [88, 91]. The assumed displacement field is of the form

$$u(x,z,t) = u_0(x,t) + z\theta_x(x,t), \quad w(x,z,t) = w_0(x,t) \tag{3.4}$$

where

$$\theta_x = -\frac{\partial w_0}{\partial x} \tag{3.5}$$

and z is the coordinate perpendicular to the undeformed midplane of the beam, and (x) is the coordinate lying in the plane. The theory does not qualify to be called first-order because the first-order terms, θ_x is not independent of w_0. Finite element models of BET require C^1-continuity, that is, continuity of the transverse displacement w_0 as well as its derivatives - slopes θ_x, and the development of BET finite elements that satisfy all completeness and compatibility requirements is cumbersome.

The simplest first-order shear deformation theory (FSDT) is based on the displacement expansion

$$u(x,z,t) = u_0(x,y,t) + z\theta_x(x,y,t), \quad w(x,z,t) = w_0(x,t) \tag{3.6}$$

where θ_x is the rotation of a transverse normal line

$$\theta_x = \frac{\partial u}{\partial z} \tag{3.7}$$

The first-order theory is based on the first two assumptions of the Kirchhoff hypothesis, and the normality of the assumption is not invoked, making the rotation θ_x to be independent of (u_0, w_0). As a result the transverse shear strain γ_{xz} is nonzero but independent of z. This lead to the introduction of shear correction factors in the evaluation of the transverse shear forces. The finite element models of the theory require only C^0-continuity, i.e., the variables of the theory (u_0, w_0, θ_x) be continuous between elements; however, they can exhibit spurious transverse shear stiffness even in pure bending, known as the shear locking, as the beam becomes thin. The spurious transverse shear stiffness stems from an interpolation inconsistency that prevents the Kirchhoff conditions of Eq. (3.5) from being satisfied as the beam becomes thin. The shear locking phenomenon can be alleviated by using a reduced integration to evaluate transverse shear stiffness terms in the element stiffness matrix or by using higher-order approximations of the displacement field. Although the reduced integration solution is the most economical alternative, the process allows some elements to exhibit spurious displacement modes, i.e., deformation modes that result in zero strain at the Gaussian integration points.

Second-order and higher-order theories relax the Kirchhoff hypothesis further by allowing the straight lines normal to the midplane before deformation to become curves. However, most published theories still assume inextensibility of these lines. Second-order theories are not popular because of the fact that they too require shear correction factors and while not improving over FSDT.

The third-order theories assumes a cubic expansion of the displacement field which is optimal because it gives quadratic variation of transverse shear strain and stress, and require no "shear correction factors" compared to the FSDT beam theory, where the transverse shear strain and stress are constant through the beam height. Several third-order theories have been developed by different researchers, and some of them are claimed to be new whereas they are not new, as pointed by Reddy [84, 92], but only disguised in the form of the displacement expansions used. Various third-order theories developed over the years differ from each others in several ways. The final equations developed depend on (1) the displacement field, (2) the strain-displacement relations (linear or nonlinear, if nonlinear, nature of the nonlinearity included - small strain but large displacements and rotations or moderate rotations, etc.), and (3) equilibrium (or equations of motion) adopted.

In the present analysis we employ a cubic expansion of the displacement field on the similar-lines of the third-order beam theory by Reddy [87, 92, 84]. For a beam of rectangular cross-section, the displacement field is assumed to be of the following

$$
\begin{aligned}
u\left(x,z,t\right) &= u_0\left(x,t\right) + z\phi_x\left(x,t\right) - z^3 c_1\,\theta_x\left(x,t\right) \\
w\left(x,z,t\right) &= w_0\left(x,t\right)
\end{aligned} \tag{3.8}
$$

where $c_1 = 4/3h^2$, and h is the height of beam. The non-zero strain components of the Green–Lagrange strain tensor resulting from the displacement field Eq. (3.8) can be expressed as

$$
\begin{aligned}
\varepsilon_{xx} &= \frac{\partial u_0}{\partial x} + \frac{1}{2}\left(\frac{\partial w_0}{\partial x}\right)^2 + z\frac{\partial \phi_x}{\partial x} - z^3 c_1\left(\frac{\partial \theta_x}{\partial x}\right) \\
\gamma_{xz} &= \phi_x + \frac{\partial w_0}{\partial x} - c_2 z^2 \theta_x
\end{aligned} \tag{3.9}
$$

where $c_2 = 3c_1 = 4/h^2$, and we have retained the *von Kármán* nonlinear terms in the above equations. The main advantage of the present higher-order beam theory (HBT) over lower-order beam theories is that the HBT allows for C^0 continuity and also gives quadratic variation of the transverse shear strain through the thickness of the beam. Thus, there is no need for a shear correction factor as with the Timoshenko beam theory.

3.4 Linear viscoelastic constitutive relations

A material is said to be linearly viscoelastic, if stress is proportional to strain at a given time and linear superposition principle holds (Boltzmann's superposition principle). Considering the strains associated with relaxation and recovery and using the ideas of linearity, for isothermal deformation in one-dimension, the constitutive equation relating the nonzero components of the second Piola–Kirchhoff stress tensor $\boldsymbol{\sigma}$ to the Green–Lagrange strain tensor $\boldsymbol{\varepsilon}$ components can be expressed as

$$\begin{aligned}
\sigma_{xx}\left(\mathbf{x},t\right) &= E\left(0\right)\varepsilon_{xx}\left(\mathbf{x},t\right) + \int_0^t \dot{E}\left(t-s\right)\varepsilon_{xx}\left(\mathbf{x},s\right)ds \\
\sigma_{xz}\left(\mathbf{x},t\right) &= G\left(0\right)\gamma_{xz}\left(\mathbf{x},t\right) + \int_0^t \dot{G}\left(t-s\right)\gamma_{xz}\left(\mathbf{x},s\right)ds
\end{aligned} \tag{3.10}$$

where $E(t)$, $G(t)$ are the extensional, shear relaxation moduli and $\dot{E}(t-s)$, $\dot{G}(t-s)$ are derivatives with respect to $(t-s)$. By an appropriate change of variables and integrating by parts the above equations can be cast in the form of Boltzmann superposition integrals. The specific forms of $E(t)$ and $G(t)$ will depend upon the material model employed. In this study, we express the relaxation moduli in terms of the Prony series of order n as

$$E\left(t\right) = E_0 + \sum_{l=1}^{n} E_l e^{-\frac{t}{\tau_l^E}}, \qquad G\left(t\right) = G_0 + \sum_{l=1}^{n} G_l e^{-\frac{t}{\tau_l^G}} \tag{3.11}$$

The time derivative of the relaxation moduli can be expressed as

$$\dot{E}\left(t\right) = -\sum_{l=1}^{n} \frac{E_l}{\tau_l^E} e^{-\frac{t}{\tau_l^E}}, \qquad \dot{G}\left(t\right) = -\sum_{l=1}^{n} \frac{G_l}{\tau_l^G} e^{-\frac{t}{\tau_l^G}} \tag{3.12}$$

It is important to note that in the integral constitutive equations given by Eq. (3.10) we assume that a discontinuity exists in the response only at $t = 0$.

3.5 Weak forms and semi-discrete models

3.5.1 Galerkin weak formulation of HBT

To construct the finite element model, we begin with the principle of virtual work applied to a typical beam structure in undeformed configuration, which is given as

$$\int_V \left(\rho\delta\mathbf{u}\cdot\ddot{\mathbf{u}} + \delta\varepsilon : \sigma - \rho\delta\mathbf{u}\cdot\mathbf{b}\right) dV - \int_{\Gamma_\sigma} \delta\mathbf{u}\cdot\mathbf{t}ds = 0 \tag{3.13}$$

Over a typical beam element, the above is equivalent to the following four weak forms (see Reddy [90]):

$$0 = \int_{x_a}^{x_b} \left(I_0\,\delta u_0\,\ddot{u}_0 + \frac{\partial\delta u_0}{\partial x}\,N_{xx} - I_0\,\delta u_0\,f\right) dx$$
$$- \delta u_0\left(x_a\right) Q_1 - \delta u_0\left(x_b\right) Q_5 \tag{3.14}$$

$$0 = \int_{x_a}^{x_b} \left(I_0\,\delta w_0\,\ddot{w}_0 + N_{xx}\frac{\partial\delta w_0}{\partial x}\,\frac{\partial w_0}{\partial x} + Q_x\frac{\partial\delta w_0}{\partial x} - I_0\,\delta w_0\,q\right) dx$$
$$- Q_2\delta w_0\left(x_a\right) - Q_6\delta w_0\left(x_b\right) \tag{3.15}$$

$$0 = \int_{x_a}^{x_b} \left[\delta\,\theta\left(-I_4\,c_1\,\ddot{\phi} + I_6\,c_1^2\,\ddot{\theta}\right) - P_{xx}\,c_1\,\frac{\partial\delta\theta}{\partial x} - R_x\,c_2\,\delta\theta\right] dx$$
$$+ Q_3\,\delta\theta\left(x_a\right) + Q_7\,\delta\theta\left(x_b\right) \tag{3.16}$$

$$0 = \int_{x_a}^{x_b} \left[\delta\phi\left(I_2\,\ddot{\phi} - I_4\,c_1\,\ddot{\theta}\right) + M_{xx}\,\frac{\partial\delta\phi}{\partial x} + Q_x\,\delta\phi\right] dx$$
$$- Q_4\,\delta\phi\left(x_a\right) - Q_8\,\delta\phi\left(x_b\right) \tag{3.17}$$

Note we drop the subscript x on variables θ_x and ϕ_x for simplicity. The variational problem for the HBT can be stated as: find $(u_0, w_0, \theta, \phi) \in H^1(\Omega) \times H^1(\Omega) \times H^1(\Omega) \times H^1(\Omega)$ for all $(\delta u_0, \delta w_0, \delta\theta, \delta\phi) \in H^1(\Omega) \times H^1(\Omega) \times H^1(\Omega) \times H^1(\Omega)$ such that the equations Eqs. (3.14)-(3.17) hold true, where, $H^m(\Omega)$ is the Sobolev space of order m and $\Omega = [x_a, x_b]$. The constants used in the above equations are defined as follows:

$$I_i = \rho D_i = \rho \int_{A^e} z^i \, dA \tag{3.18}$$

The internal stress resultants $N_{xx}, M_{xx}, P_{xx}, Q_x$, and R_x for the HBT are defined as follows:

$$\left\{ \begin{array}{c} N_{xx} \\ M_{xx} \\ P_{xx} \end{array} \right\} = \int_{A^e} \left\{ \begin{array}{c} 1 \\ z \\ z^3 \end{array} \right\} \sigma_{xx} \, dA, \qquad \left\{ \begin{array}{c} Q_x \\ R_x \end{array} \right\} = \int_{A^e} \left\{ \begin{array}{c} 1 \\ z^2 \end{array} \right\} \sigma_{xz} \, dA \tag{3.19}$$

The secondary variables Q_i are defined as follows:

$$\begin{aligned} &Q_1 = -\left. N_{xx} \right|_{x=x_a}, \;\; Q_5 = \left. N_{xx} \right|_{x=x_b}, \;\; Q_2 = -\left. Q_x \right|_{x=x_a}, \;\; Q_6 = \left. Q_x \right|_{x=x_b} \\ &Q_3 = -\left. c_1 P_{xx} \right|_{x=x_a}, Q_7 = \left. c_1 P_{xx} \right|_{x=x_b}, Q_4 = -\left. M_{xx} \right|_{x=x_a}, Q_8 = \left. M_{xx} \right|_{x=x_b} \end{aligned} \tag{3.20}$$

3.5.2 Semi-discrete finite element models

Since the assumed kinematic displacement requires only the continuity of the primary variables across the element boundaries and not its derivatives, i.e. C^0 continuous, we take the following equal-order interpolation functions for all primary variables

$$\begin{aligned} u_0(x,t) &= \sum_{j=1}^{p} \Delta_j^{(1)}(t)\,\psi_j(x), \quad w_0(x,t) = \sum_{j=1}^{p} \Delta_j^{(2)}(t)\,\psi_j(x), \\ \theta_x(x,t) &= \sum_{j=1}^{p} \Delta_j^{(3)}(t)\,\psi_j(x), \quad \phi_x(x,t) = \sum_{j=1}^{p} \Delta_j^{(4)}(t)\,\psi_j(x) \end{aligned} \tag{3.21}$$

where $\Delta_j^{(1)}, \Delta_j^{(2)}, \Delta_j^{(3)}, \Delta_j^{(4)}$are the generalized displacements at the nodes and ψ_j are the one-dimensional nodal spectral interpolation functions.

Substituting Eq. (3.21) in to the weak forms in Eqs. (3.14)-(3.17) yields the semi-discrete finite element model of the HBT beam element. The finite element equations can be expressed as

$$[M]\left\{\ddot{\Delta}\right\}+[K]\{\Delta\}+\int\limits_0^t\left[\tilde{K}\right]\{\Delta(s)\}\;ds=\{F\} \tag{3.22}$$

The components in the above equation are explicitly given as

$$\begin{aligned}
M_{ij}^{11}&=\int_{x_a}^{x_b}I_0\psi_i\psi_j dx, M_{ij}^{12}=0, M_{ij}^{13}=0, M_{ij}^{14}=0\\
M_{ji}^{21}&=0, M_{ij}^{22}=\int_{x_a}^{x_b}I_0\psi_i\psi_j dx, M_{ij}^{23}=0, M_{ij}^{24}=0\\
M_{ij}^{31}&=0, M_{ij}^{32}=0, M_{ij}^{33}=\int_{x_a}^{x_b}I_6c_1^2\psi_i\psi_j dx, M_{ij}^{34}=\int_{x_a}^{x_b}-c_1I_4\psi_i\psi_j dx\\
M_{ij}^{41}&=0, M_{ij}^{42}=0, M_{ij}^{43}=\int_{x_a}^{x_b}-c_1I_4\psi_i\psi_j dx, M_{ij}^{44}=\int_{x_a}^{x_b}I_2\psi_i\psi_j dx
\end{aligned} \tag{3.23}$$

$$\begin{aligned}
K_{ij}^{11}&=\int_{x_a}^{x_b}E(0)\,D_0\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx,\;\; 2K_{ij}^{12}=K_{ji}^{21}=\int_{x_a}^{x_b}E(0)\,D_0\frac{\partial w_0}{\partial x}\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx\\
K_{ij}^{22}&=\int_{x_a}^{x_b}\left[\frac{1}{2}E(0)\,D_0\left(\frac{\partial w_0}{\partial x}\right)^2+G(0)\,D_0\right]\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx\\
K_{ij}^{23}&=K_{ji}^{32}=\int_{x_a}^{x_b}-G(0)\,D_2\;c_2\frac{d\psi_i}{dx}\psi_j dx,\\
K_{ij}^{24}&=K_{ji}^{42}=\int_{x_a}^{x_b}G(0)\,D_0\frac{d\psi_i}{dx}\psi_j dx\\
K_{ij}^{33}&=\int_{x_a}^{x_b}E(0)\,D_6\,c_1^2\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx+\int_{x_a}^{x_b}G(0)\,D_4\,c_2^2\psi_i\psi_j dx\\
K_{ij}^{34}&=K_{ji}^{43}=\int_{x_a}^{x_b}-E(0)\,D_4\,c_1\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx\;+\int_{x_a}^{x_b}-G(0)\,D_2\,c_2\psi_i\psi_j dx\\
K_{ij}^{44}&=\int_{x_a}^{x_b}E(0)\,D_2\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx+\int_{x_a}^{x_b}G(0)\,D_0\psi_i\psi_j dx
\end{aligned} \tag{3.24}$$

$$
\begin{aligned}
\tilde{K}_{ij}^{11} &= \int_{x_a}^{x_b} \dot{E}\left(t-s\right) D_0 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx \\
2\tilde{K}_{ij}^{12} = \tilde{K}_{ji}^{21} &= \int_{x_a}^{x_b} \dot{E}\left(t-s\right) D_0 \frac{\partial w_0\left(x,s\right)}{\partial x}\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx \\
\tilde{K}_{ij}^{22} &= \int_{x_a}^{x_b} \left(\frac{1}{2}\dot{E}\left(t-s\right) D_0 \frac{\partial w_0}{\partial x}\frac{\partial w_0\left(x,s\right)}{\partial x} + \dot{G}\left(t-s\right) D_0\right)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx \\
\tilde{K}_{ij}^{23} = \tilde{K}_{ji}^{32} &= \int_{x_a}^{x_b} -\dot{G}\left(t-s\right) D_2 c_2 \frac{d\psi_i}{dx}\psi_j dx \\
\tilde{K}_{ij}^{24} = \tilde{K}_{ji}^{42} &= \int_{x_a}^{x_b} \dot{G}\left(t-s\right) D_0 \frac{d\psi_i}{dx}\psi_j dx \\
\tilde{K}_{ij}^{33} &= \int_{x_a}^{x_b} \dot{E}\left(t-s\right) D_6\, c_1^2 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx + \int_{x_a}^{x_b} \dot{G}\left(t-s\right) D_4\, c_2^2 \psi_i\psi_j dx \\
\tilde{K}_{ij}^{34} = \tilde{K}_{ji}^{43} &= \int_{x_a}^{x_b} -\dot{E}\left(t-s\right) D_4\, c_1 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx + \int_{x_a}^{x_b} -\dot{G}\left(t-s\right) D_2\, c_2 \psi_i\psi_j dx \\
\tilde{K}_{ij}^{44} &= \int_{x_a}^{x_b} \dot{E}\left(t-s\right) D_2 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx + \int_{x_a}^{x_b} \dot{G}\left(t-s\right) D_0 \psi_i\psi_j dx
\end{aligned}
\tag{3.25}
$$

All other terms in Eqs. (3.24, 3.25) are zeros. The force terms are

$$
\begin{aligned}
F_i^1 &= \int_{x_a}^{x_b} I_0 f\, \psi_i dx + \psi_i\left(x_a\right) Q_1 + \psi_i\left(x_b\right) Q_5 \\
F_i^2 &= \int_{x_a}^{x_b} I_0 q\, \psi_i dx + Q_2 \psi_i\left(x_a\right) + Q_6 \psi_i\left(x_b\right) \\
F_i^3 &= -Q_3 \psi_i\left(x_a\right) - Q_7 \psi_i\left(x_b\right) \\
F_i^4 &= Q_4 \psi_i\left(x_a\right) + Q_8 \psi_i\left(x_b\right)
\end{aligned}
\tag{3.26}
$$

3.6 Full discretization : recurrence formulas and time approximations

3.6.1 Quasi-static time discretization: recurrence formula

In order to derive fully discretized finite element equations, we start with the partitioning of the time interval $[0\,,\,T] \subset \mathbb{R}$ (region of interest) into set of N non-overlapping subintervals such that

$$[0\,,\ T] = \bigcup_{k=1}^{N} [t_k\,,\ t_{k+1}] \tag{3.27}$$

The final solution is obtained by repeatedly solving an initial value problem within each subregion $[t_k\,,\ t_{k+1}]$, with the known values of solution at $t = t_k$ as initial conditions. From the Eq. (3.22) it is clear that the semi-discrete finite element equations have contributions from elastic and viscous parts of the constitutive relations. The elastic part is simple and with in each subinterval, it can easily be fully descritized using well know schemes like Newmark or its variants [68]. However, the contribution from viscous part is in the form of convolution integrals, hence full discretization is not so straight forward. In order to solve the problem in each subinterval, we can approximate these convolution integrals using two-point (trapezoidal rule) or three-point (simpson's rule) formulas. But a direct temporal integration from here, results in a computationally unattractive solution procedure which requires the storage of the entire deformation history. It also becomes a main bottle neck for storing them especially when the computational memory scarce. Another negative aspect is that, when N is large, much of the computational time expended to get solution at a subinterval, goes into the evaluation of the convolution integrals.

To circumvent these issues, we develop a recurrence scheme for two-point (trapezoidal rule) formula that can be used to approximate the convolution integrals with in each subinterval. The two-point recurrence scheme requires the storage of the generalized displacements and a set of internal variables evaluated at the Gauss points, from the previous time step only. A similar three-point simpson's scheme can also be developed which requires the storage from last two time steps as in [69]. Using these ideas the convolution integral appearing in Eq. (3.22) can be expressed as

$$\int_{0}^{t_N} \left[\tilde{K}\right] \{\Delta(s)\}\, ds = \sum_{k=1}^{N-1} \int_{t_k}^{t_{k+1}} \left[\tilde{K}\right] \{\Delta(s)\}\, ds \tag{3.28}$$

In order to develop the recurrence formulation the following multiplicative decomposition of the relaxation moduli [102] is used. These equations hold true as the relaxation moduli can be expressed in terms of the Prony series within each subinterval.

$$\begin{aligned} \dot{E}(t_{k+1}-s) &= \sum_{l=1}^{n} e^{-\Delta t_k/\tau_l^E} \dot{E}_l(t_k-s) \\ \dot{G}(t_{k+1}-s) &= \sum_{l=1}^{n} e^{-\Delta t_k/\tau_l^G} \dot{G}_l(t_k-s) \end{aligned} \tag{3.29}$$

where $\Delta t_k = t_{k+1} - t_k$. Using the above, the Eq. (3.28) can be expressed in index notion at an arbitrary time step $t = t_s$ as

$$X_i(t_s) = \sum_{k=1}^{s-1} \int_{t_k}^{t_{k+1}} \tilde{K}_{ij}\Delta_j(s)\, ds \cong \sum_{l=1}^{n} \sum_{m=1}^{NGP} \alpha_m \bar{X}_i^{lm}(t_s) \tag{3.30}$$

where Einstein's summation convention on repeated indices is implied. As noted previously, Gauss quadrature is employed in evaluation of $\tilde{K}_{ij}$, resulting in the summation over m (where NGP is the number of Gauss points). The quantity $\bar{X}_i^{lm}$ assumes the following possible forms for extensional and shear moduli

$$\bar{X}_i^{lm}(t_s) = e^{-\frac{\Delta t_{s-1}}{\tau_l^E}} \bar{X}_i^{lm}(t_{s-1}) - \frac{\Delta t_{s-1}}{2} \frac{E_l}{\tau_l^E} \left(e^{-\frac{\Delta t_{s-1}}{\tau_l^E}} f_i^m(t_{s-1}) + f_i^m(t_s) \right) \tag{3.31}$$

$$\bar{X}_i^{lm}(t_s) = e^{-\frac{\Delta t_{s-1}}{\tau_l^G}} \bar{X}_i^{lm}(t_{s-1}) - \frac{\Delta t_{s-1}}{2} \frac{G_l}{\tau_l^G} \left(e^{-\frac{\Delta t_{s-1}}{\tau_l^G}} f_i^m(t_{s-1}) + f_i^m(t_s) \right) \tag{3.32}$$

Note to get the above expressions, we replaced convolution integrals with in each subinterval with a two-point trapezoidal rule. Also, the specific forms of α_m and $f_i^m(t_s)$ depend on components of $\tilde{K}_{ij}$. In Eq. (3.30), even though there are $(s-1)$ time steps of k to get to time $t = t_s$, the above equations just need the values of solution $\{\Delta(t_s)\}$ and internal variables $\bar{X}_i^{lm}(t_{s-1})$ from $k = t_s$ and $k = t_{s-1}$. There is no need to store these values for all the $(s-1)$ time steps. Thus the above equations represent recurrence formulas in terms of the internal variables $\bar{X}_i^{lm}(t_s)$ with $\bar{X}_i^{lm}(t_1 = 0) = 0$.

Using Eqs. (3.31),(3.32) results in the following quasi-static fully-discretized equations for generalized displacements at the current time step

$$\left[\bar{K}\right]_s \{\Delta(t_s)\} = \{F\}_s - \left\{\tilde{Q}\right\}_s \tag{3.33}$$

the above in expanded form is given as

$$\begin{bmatrix} [\bar{K}^{11}] & [\bar{K}^{12}] & [\bar{K}^{13}] & [\bar{K}^{14}] \\ [\bar{K}^{21}] & [\bar{K}^{22}] & [\bar{K}^{23}] & [\bar{K}^{24}] \\ [\bar{K}^{31}] & [\bar{K}^{32}] & [\bar{K}^{33}] & [\bar{K}^{34}] \\ [\bar{K}^{41}] & [\bar{K}^{42}] & [\bar{K}^{43}] & [\bar{K}^{44}] \end{bmatrix}_s \begin{Bmatrix} \{\Delta^1(t_s)\} \\ \{\Delta^2(t_s)\} \\ \{\Delta^3(t_s)\} \\ \{\Delta^4(t_s)\} \end{Bmatrix} = \begin{Bmatrix} \{F^1\} \\ \{F^2\} \\ \{F^3\} \\ \{F^4\} \end{Bmatrix}_s - \begin{Bmatrix} \{\tilde{Q}^1\} \\ \{\tilde{Q}^2\} \\ \{\tilde{Q}^3\} \\ \{\tilde{Q}^4\} \end{Bmatrix}_s \tag{3.34}$$

where the stiffness matric is given by

$$
\begin{aligned}
\bar{K}_{ij}^{11} &= \int_{x_a}^{x_b} \left(E(0) + \frac{\Delta t_{N-1}}{2}\dot{E}(0)\right) D_0 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx \\
2\bar{K}_{ij}^{12} = \bar{K}_{ji}^{21} &= \int_{x_a}^{x_b} \left(E(0) + \frac{\Delta t_{N-1}}{2}\dot{E}(0)\right) D_0 \frac{\partial w_0}{\partial x}\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx \\
\bar{K}_{ij}^{22} &= \int_{x_a}^{x_b} \left[\frac{1}{2}\left(E(0) + \frac{\Delta t_{N-1}}{2}\dot{E}(0)\right) D_0 \left(\frac{\partial w_0}{\partial x}\right)^2 + \left(G(0) + \frac{\Delta t_{N-1}}{2}\dot{G}(0)\right) D_0\right] \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx \\
\bar{K}_{ij}^{23} = \bar{K}_{ji}^{32} &= \int_{x_a}^{x_b} -\left(G(0) + \frac{\Delta t_{N-1}}{2}\dot{G}(0)\right) D_2\, c_2 \frac{d\psi_i}{dx}\psi_j dx \\
\bar{K}_{ij}^{24} = \bar{K}_{ji}^{42} &= \int_{x_a}^{x_b} \left(G(0) + \frac{\Delta t_{N-1}}{2}\dot{G}(0)\right) D_0 \frac{d\psi_i}{dx}\psi_j dx \\
\bar{K}_{ij}^{33} &= \int_{x_a}^{x_b} \left(E(0) + \frac{\Delta t_{N-1}}{2}\dot{E}(0)\right) D_6\, c_1^2 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx + \int_{x_a}^{x_b} \left(G(0) + \frac{\Delta t_{N-1}}{2}\dot{G}(0)\right) D_4\, c_2^2 \psi_i\psi_j dx \\
\bar{K}_{ij}^{34} = \bar{K}_{ji}^{43} &= \int_{x_a}^{x_b} -\left(E(0) + \frac{\Delta t_{N-1}}{2}\dot{E}(0)\right) D_4\, c_1 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx \\
&\quad + \int_{x_a}^{x_b} -\left(G(0) + \frac{\Delta t_{N-1}}{2}\dot{G}(0)\right) D_2\, c_2 \psi_i\psi_j dx \\
\bar{K}_{ij}^{44} &= \int_{x_a}^{x_b} \left(E(0) + \frac{\Delta t_{N-1}}{2}\dot{E}(0)\right) D_2 \frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx + \int_{x_a}^{x_b} \left(G(0) + \frac{\Delta t_{N-1}}{2}\dot{G}(0)\right) D_0 \psi_i\psi_j dx
\end{aligned}
\tag{3.35}
$$

and the force vector $\tilde{Q}$ is given by

$$
\begin{aligned}
\left\{\tilde{Q}^1\right\} &= \left\{{}^1\bar{Q}^1\right\} + \left\{{}^2\bar{Q}^1\right\} \\
\left\{\tilde{Q}^2\right\} &= \left\{{}^1\bar{Q}^2\right\} + \left\{{}^2\bar{Q}^2\right\} + \left\{{}^3\bar{Q}^2\right\} + \left\{{}^4\bar{Q}^2\right\} + \left\{{}^5\bar{Q}^2\right\} \\
\left\{\tilde{Q}^3\right\} &= \left\{{}^1\bar{Q}^3\right\} + \left\{{}^2\bar{Q}^3\right\} + \left\{{}^3\bar{Q}^3\right\} + \left\{{}^4\bar{Q}^3\right\} + \left\{{}^5\bar{Q}^3\right\} \\
\left\{\tilde{Q}^4\right\} &= \left\{{}^1\bar{Q}^4\right\} + \left\{{}^2\bar{Q}^4\right\} + \left\{{}^3\bar{Q}^4\right\} + \left\{{}^4\bar{Q}^4\right\} + \left\{{}^5\bar{Q}^4\right\}
\end{aligned}
\tag{3.36}
$$

All other components of matric in Eq. (3.34) are zeros. The right hand side of Eq. (3.36) are presented in Appendix B.

3.6.2 Fully-transient time approximation: Newmark's scheme

There are many numerical schemes in literature [68, 125, 37] to convert the second-order differential equations in time to algebraic equations. We use the popular Newmark scheme from [90] for the generic time differential equation in variable $\{u\}$

$$[K]\{u\} + [C]\{\dot{u}\} + [M]\{\ddot{u}\} = \{F\} \tag{3.37}$$

subjected to initial conditions

$$\{u(0)\} = \{u_0\}, \quad \{\dot{u}(0)\} = \{v_0\} \tag{3.38}$$

where $\big[M\big]$ denotes the mass matrix, $\big[C\big]$ the damping matrix, $\big[K\big]$ the combined stiffness matrix due to quasi-static time discretization of viscoelastic terms, and $\big\{F\big\}$ the effective force vector due to the same. In the present study, we have $\big[C\big] = 0$, but is included for completeness. The Newmark parameters α and $\gamma\,(= 2\beta)$ that determine the stability and accuracy of the scheme are taken as $\alpha = \gamma = 1/2$. This makes it a stable constant-average acceleration method. The time-discretized equations can be written as

$$\big[\hat{K}\big]_{s+1}\{u\}_{s+1} = \big\{\hat{F}\big\}_{s,s+1} \tag{3.39}$$

where

$$\begin{aligned}
\left[\hat{K}\right]_{s+1} &= [K]_{s+1} + a_3\,[M]_{s+1} + a_6\,[C]_{s+1} \\
\left\{\hat{F}\right\}_{s,s+1} &= \{F\}_{s+1} + [M]_{s+1}\,\{A\}_s + [C]_{s+1}\,\{B\}_s \\
\{A\}_s &= a_3\,\{b\}_s = a_3\,\{u\}_s + a_4\,\{\dot{u}\}_s + a_5\,\{\ddot{u}\}_s \\
\{B\}_s &= a_6\,\{u\}_s + a_7\,\{\dot{u}\}_s + a_8\,\{\ddot{u}\}_s
\end{aligned} \tag{3.40}$$

and

$$a_3 = \frac{1}{\beta\,(\Delta t)^2},\;\; a_4 = a_3\Delta t,\;\; a_5 = \frac{1}{\gamma} - 1 \tag{3.41}$$

The calculation of $\left[\hat{K}\right]$ and $\left\{\hat{F}\right\}$ requires the knowledge of the initial conditions $\{u\}_0\,, \{\dot{u}\}_0\,,$ and $\{\ddot{u}\}_0$. In practice, we do not know the value of $\{\ddot{u}\}_0$. As an approximation, it can be calculated from (we assume that the applied force is zero at $t = 0$):

$$\{\ddot{u}\}_0 = [M]^{-1}\left(\{F\}_0 - [K]\,\{u\}_0 - [C]\,\{\dot{u}\}_0\right) \tag{3.42}$$

However, the initial guess for displacement and velocity are taken to be $\{u\}_0 = 0$ $\{\dot{u}\}_0 = 0$, respectively, and the applied force is zero at $t = 0$, hence the initial condition for acceleration becomes $\{\ddot{u}\}_0 = 0$. So, Eq. (3.42) is not used. At the end of each time step, the new velocity vector $\{\dot{u}\}_{s+1}$ and acceleration vector $\{\ddot{u}\}_{s+1}$ are computed using

$$\begin{aligned}
\{\ddot{u}\}_{s+1} &= a_3\left(\{u\}_{s+1} - \{u\}_s\right) - a_4\,\{\dot{u}\}_s - a_5\,\{\ddot{u}\}_s \\
\{\dot{u}\}_{s+1} &= \{\dot{u}\}_s + a_2\,\{\ddot{u}\}_s + a_1\,\{\ddot{u}\}_{s+1} \\
a_1 &= \alpha\,\Delta t, a_2 = (1-\alpha)\;\Delta t
\end{aligned} \tag{3.43}$$

The assembly, imposition of boundary conditions, and the solution procedures are the same as in static problems. The integrals in Eqs. (3.33),(3.39) are evaluated using Gauss quadrature rules. In our implementation, Gauss-Legendre rules are used with the nodal spectral basis, and *full integration* is used to evaluate all the integrals.

3.7 Numerical results and discussion

The fully discretized finite element equations are nonlinear due to inclusion of the *von Kármán* strains in the formulation. For our formulation we solve the equations iteratively using Newton-Raphson linearization procedure [9]. The linearized equations are of the form

$$\left\{\Delta^{(r)}\right\}_s = \left\{\Delta^{(r-1)}\right\}_s - \left[\bar{T}\right]_s^{-1}\left(\left[\bar{K}^{(r-1)}\right]_s\left\{\Delta^{(r-1)}\right\}_s + \left\{F^{(r-1)}\right\}_s - \left\{\tilde{Q}^{(r-1)}\right\}_s\right) \tag{3.44}$$

where $\left\{\Delta^{(r)}\right\}_s$ represents the solution at the $r'th$ iteration and time $t = t_s$. The tangent stiffness matrix $\left[\bar{T}\right]_s$ in the Eq. (3.44) is defined using Einstein's summation notation as

$$\bar{T}_{ij} = \sum_1^n \bar{K}_{ij} + \frac{\partial \bar{K}_{im}}{\partial \Delta_j}\Delta_m + \frac{\partial \tilde{Q}_i}{\partial \Delta_j} \tag{3.45}$$

All quantities in Eq. (3.45) comprising the tangent stiffness matrix are formulated using the solution from $(r-1)'th$ iteration. It is important to note that all the partial derivatives are taken with respect to the solution of the current time step. Applying the Newton's method to the fully-discretized HBT beam equations results in the following components of tangent stiffness matrix

$$
\begin{aligned}
&\bar{T}_{ij}^{11} = \bar{K}_{ij}^{11}, \qquad \bar{T}_{ij}^{12} = 2\bar{K}_{ij}^{12}, \qquad \bar{T}_{ij}^{13} = 0, \qquad \bar{T}_{ij}^{14} = 0 \\
&\bar{T}_{ij}^{21} = \bar{K}_{ij}^{21}, \qquad \bar{T}_{ij}^{23} = \bar{K}_{ij}^{23}, \qquad \bar{T}_{ij}^{24} = \bar{K}_{ij}^{24}, \\
&\bar{T}_{ij}^{22} = \bar{K}_{ij}^{22} + \int_{x_a}^{x_b} D_0 \left\{ \left(E\left(0\right) + \frac{\Delta t_{N-1}}{2} \dot{E}\left(0\right) \right) \left[\frac{\partial u_0}{\partial x} + \left(\frac{\partial w_0}{\partial x} \right)^2 \right] \right. \\
&\qquad \left. + \frac{\Delta t_{N-1}}{2} \dot{E}\left(\Delta t_{N-1}\right) \left[\frac{\partial u_0\left(x, t_{N-1}\right)}{\partial x} + \frac{1}{2} \left(\frac{\partial w_0\left(x, t_{N-1}\right)}{\partial x} \right)^2 \right] \right\} \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} dx \\
&\qquad + \sum_{l=1}^{n} \sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}} \left({}^3\bar{X}_i^{lm}\left(t_{N-1}\right) + {}^4\bar{X}_i^{lm}\left(t_{N-1}\right) \right) \frac{d\psi_j\left(x_m\right)}{dx} \\
&\bar{T}_{ij}^{31} = \bar{K}_{ij}^{31}, \qquad \bar{T}_{ij}^{32} = \bar{K}_{ij}^{32}, \qquad \bar{T}_{ij}^{33} = \bar{K}_{ij}^{33}, \qquad \bar{T}_{ij}^{34} = \bar{K}_{ij}^{34} \\
&\bar{T}_{ij}^{41} = \bar{K}_{ij}^{41}, \qquad \bar{T}_{ij}^{42} = \bar{K}_{ij}^{42}, \qquad \bar{T}_{ij}^{43} = \bar{K}_{ij}^{43}, \qquad \bar{T}_{ij}^{44} = \bar{K}_{ij}^{44}
\end{aligned}
\tag{3.46}
$$

3.7.1 Element locking

As mentioned above, the nonlinear finite element equations are linearized using Newton's procedure and the equations are solved using iterative scheme. Since a nonlinear beam becomes stiff with load, total load is divided into several smaller load steps with the solution of each step being used for the next one. For all the problems in the present study, five load steps are utilized with a maximum of 20 iterations at each load step. However, all problems in this study converged within 4 iterations for a tolerance of $\varepsilon = 10^{-6}$. At the first load step, the initial guess vector for the solution is chosen to be the zero vector. This condenses out nonlinear terms and solution is linear. At each subsequent iteration the solution vector from the previous iteration is used as the new guess vector. And at each new load step, the converged solution from the previous load step is used as the initial guess vector. At each time step following $t = 0$, the finite element equations are solved iteratively using the Newton procedure without the employment of load steps. The finite element formulations and procedures developed are tested for quasi-static, fully-transient cases and the solutions match with lower-order beam theory results from [72, 22].

The lower-order beam theories like the Euler-Bernoulli (EBT) suffer from *membrane locking* while Timoshenko (TBT) suffers from both *shear* and *membrane locking* [84, 87]. These effects are important if the beam bending is nonlinear with load. As the displacements u_0 and w_o are coupled, the beam undergoes axial displacement without any axial forces. Selective or reduced integration techniques are used to eliminate them to a certain extent. The lower-order HBT elements used in this study do suffer from membrane and shear locking in the thin-beam limit. However, they are reduced with the use of higher-order spectral interpolation functions for all the primary variables. In particular, we introduce the following full integration HBT elements: HBTLN, HBTQD, HBTCB and HBTQI, which each have 2, 3, 4 and 6 nodes, respectively.

3.7.2 Material properties

For the present analysis, we utilize a viscoelastic material model based on the experimental findings of Lai and Bakker [57] for a glassy amorphous polymer material (PMMA). The Prony series parameters for the viscoelastic relaxation modulus given in Table 3.1 were calculated by Payette and Reddy [72] from the published compliance parameters [57]. Although the finite element formulation places no restriction on the relationship between *E(t)* and *G(t)*, for the present analysis we adopt the approach taken by Chen [22] and assume that the shear and relaxation moduli are related by

$$G\left(t\right) = \frac{E\left(t\right)}{2\left(1+\nu\right)} \tag{3.47}$$

where ν is Poisson's ratio of the material, which is assumed to be time-independent and equal to $\nu = 0.4$ [55].

Table 3.1: Viscoelastic moduli of PMMA.

E_0	205.7818 Ksi		
E_1	43.1773 Ksi	τ_1^E	9.1955×10^{-1} s
E_2	9.2291 Ksi	τ_2^E	9.8120×10^{0} s
E_3	22.9546 Ksi	τ_3^E	9.5268×10^{1} s
E_4	26.2647 Ksi	τ_4^E	9.4318×10^{2} s
E_5	34.6298 Ksi	τ_5^E	9.2066×10^{3} s
E_6	40.3221 Ksi	τ_6^E	8.9974×10^{4} s
E_7	47.5275 Ksi	τ_7^E	8.6852×10^{5} s
E_8	46.8108 Ksi	τ_8^E	8.5142×10^{6} s
E_9	58.6945 Ksi	τ_9^E	7.7396×10^{7} s

3.7.3 Quasi-static: loading and boundary conditions

A viscoelastic beam of uniform rectangular cross section $1\,\text{in} \times 1\,\text{in}$, and length $L = 100\,\text{in}$, with material properties given in Table 3.1 is used for analysis. The computational domain is reduced by taking advantage of the symmetry about $x = L/2$. At $t = 0$ the beam is subjected to a time invariant uniform vertical distributed load $q = 0.25\,\text{lb}_\text{f}/\text{in}$. For the HBT, we consider 10 HBTLN elements (11 nodes), 5 HBTQD elements (11 nodes), 3 HBTCB elements (10 nodes) and 2 HBTQI elements (11 nodes). Apart from the above finite elements, p and h refinement studies are done to make sure that the solution is converged. Three sets of boundary conditions considered in the analysis are:

(i) *Hinged at both ends:*

$$w_0\,(0,t) = u_0\,(L/2,t) = \phi_x\,(L/2,t) = 0 \tag{3.48}$$

(ii) *Pinned at both ends:*

$$u_0\,(0,t) = w_0\,(0,t) = u_0\,(L/2,t) = \phi_x\,(L/2,t) = 0 \tag{3.49}$$

(iii) *Clamped at both ends:*

$$\begin{aligned} u_0\,(0,t) = w_0\,(0,t) = \theta_x\,(0,t) = \phi_x\,(0,t) = 0 \\ u_0\,(L/2,t) = \ \theta_x\,(L/2,t) = \ \phi_x\,(L/2,t) \ = 0 \end{aligned} \tag{3.50}$$

Each of the above finite elements and boundary conditions are chosen to demonstrate the geometric nonlinear capabilities of the finite element models that cannot be captured by the uncoupled linear formulation.

In Table 3.2, we present numerical results for quasi-static beam deflection using cases (i)-(iii) for the HBT finite elements. A constant time step $\Delta t = 1.0\,\mathrm{s}$ has been employed with a total simulation time of 1800 s. Results for the high-order HBT (HBTCB and HBTQI) finite elements are in excellent agreement, but differ largely from HBTLN element, which as expected, suffers excessively from shear locking [89]. To make sure that the solution of HBTQI element is fully converged h refinement studies are done on HBTQI element keeping all other parameters constant. The numerical results presented in Table 3.2 for the HBTQI element can be obtained with only one element. As expected the HBTQI elements are most optimal in terms of computational cost due to less number of global nodes than other elements.

Graphical results for HBTQD and HBTQI elements are provided in Fig. 3.1. As

Table 3.2: Quasi-static finite element results for the maximum deflection w_{max} of a viscoelastic beam under uniform distributed load q with three different sets of boundary conditions.

Time,t	HBTLN	HBTQD	HBTCB	HBTQI
Hinged-hinged beam				
0	0.7979	7.1931	7.3281	7.3496
200	0.9426	8.4586	8.6555	8.6826
400	0.9581	8.5937	8.7982	8.8260
600	0.9677	8.6760	8.8854	8.9136
800	0.9753	8.7420	8.9552	8.9837
1000	0.9817	8.7981	9.0146	9.0434
1200	0.9873	8.8465	9.0659	9.0950
1400	0.9922	8.8887	9.1107	9.1399
1600	0.9965	8.9258	9.1500	9.1795
1800	1.0003	8.9586	9.1849	9.2145
Pinned-pinned beam				
0	0.6923	1.2466	1.2465	1.2470
200	0.7874	1.3257	1.3256	1.3260
400	0.7971	1.3337	1.3335	1.3340
600	0.8030	1.3385	1.3384	1.3388
800	0.8077	1.3424	1.3422	1.3426
1000	0.8117	1.3456	1.3455	1.3459
1200	0.8151	1.3485	1.3483	1.3487
1400	0.8181	1.3509	1.3507	1.3511
1600	0.8207	1.3530	1.3529	1.3533
1800	0.8230	1.3549	1.3547	1.3552
Clamped-clamped beam				
0	0.1469	0.8941	0.9102	0.9109
200	0.1734	0.9820	0.9987	0.9997
400	0.1763	0.9909	1.0076	1.0086
600	0.1780	0.9962	1.0130	1.0140
800	0.1794	1.0005	1.0173	1.0183
1000	0.1806	1.0041	1.0209	1.0220
1200	0.1816	1.0072	1.0241	1.0251
1400	0.1825	1.0099	1.0268	1.0278
1600	0.1833	1.0123	1.0292	1.0302
1800	0.1840	1.0144	1.0313	1.0323

expected, the deflection steadily increases, then approaches a zero slope, for it to reach an equilibrium or a long-time constant value. This behavior is called creep under constant load. From the figure we can clearly see for the case of hinged-hinged beam the values of deflection for a HBTQD element are less compared to the values for a HBTQI element. This shows that there is still a small locking in HBTQD element compared to HBTQI element. The hinged-hinged beam has no end constraints on the axial displacement, so it will not develop significant axial strains due to transverse deflections. But pinned-pinned and clamped-clamped beams are constrained from axial motion at $x = 0$ and $x = L/2$. As a result, these develop axial strains and offer resistance to transverse deflection of the beam. This resistance increases with load making them more stiffer. Hence hinged-hinged beams have larger transverse deflections than other two cases. Also at $t = 0$, the results coincide with the instantaneous elastic solution where Young's modulus is given as $E = 535.4\,\text{Ksi}$.

Next we investigate the creep and recovery behavior of the viscoelastic constitutive model using a time dependent load. An important characteristic is that the beam should eventually return to its original configuration once the loads are removed. To demonstrate that the finite element models capture this effect we consider the clamped-clamped case presented above subjected to a quasi-static transverse load

$$q(t) = q_0 \left\{ H(t) - \frac{1}{\tau(\beta-\alpha)}[(t-\alpha\tau)H(t-\alpha\tau) - (t-\beta\tau)H(t-\beta\tau)] \right\} \qquad (3.51)$$

where $q_0 = 0.25\,\text{lb}_\text{f}/\text{in}$,$\tau = 1800\,\text{s}$ and $H(t)$ is the Heaviside function. The parameters $0 \leq \alpha \leq \beta \leq 1$ are constants. Equation Eq. (3.51) represents a load function that is constant in $0 < t < \alpha\tau$ and then linearly decreases to zero from $t = \alpha\tau$ to $t = \beta\tau$. For $t > \beta\tau$, the load is maintained at zero. In Fig. 3.2, we present numerical results for various values of α and β , where we have employed a constant time step

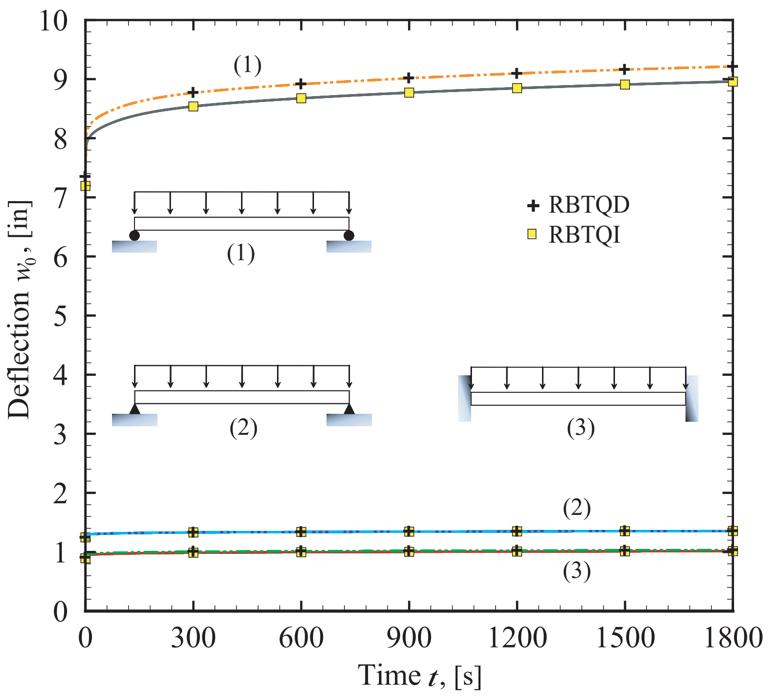


Figure 3.1: Quasi-static maximum vertical deflection w_{max}, of viscoelastic beam under uniform distributed load q.

of $\Delta t = 1.0\,\text{s}$ with two HBTQI elements. As expected each of the curves in the figure follow a path of delayed recovery from $t = \alpha\tau$ to $t = \beta\tau$ and then to its original configuration as t tends to infinity once the applied load is removed.

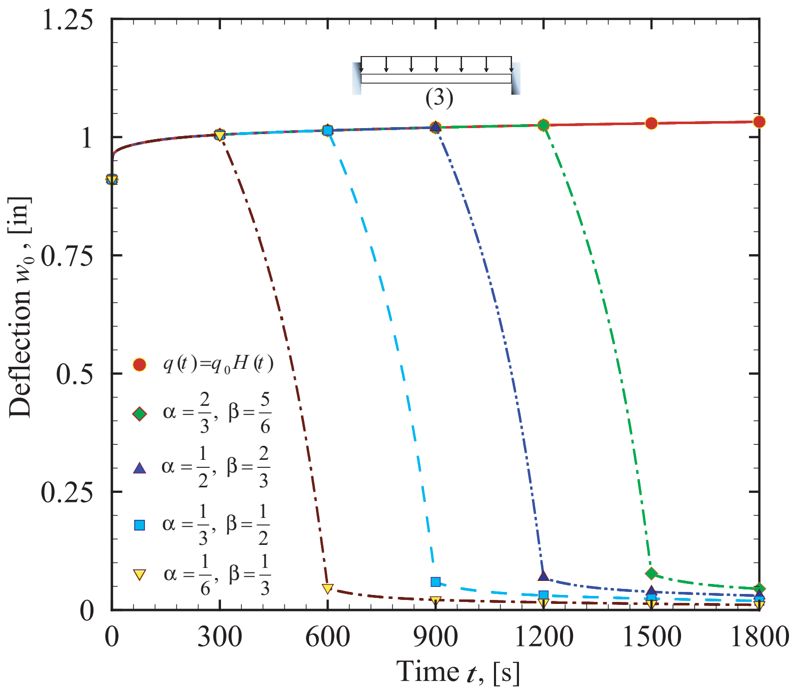


Figure 3.2: Quasi-static maximum vertical deflection w_{max}, of clamped-clamped beam under time-dependent transverse loading $q(t)$.

We also consider the effect that shear strain has on the transverse deflection of viscoelastic beams. To this end we modify the original thin beam problems by letting $L = 10\,\text{in}$, $q = 25\,\text{lb}_\text{f}/\text{in}$ and $t = 1.0\,\text{s}$. All other parameters are kept the same as in the previous examples. In Table 3.3, we present numerical results for the transverse deflection of clamped-clamped and pinned-pinned beams using HBTQD, HBTCB and HBTQI elements.

Table 3.3: Effect of transverse shear strain on the maximum quasi-static vertical deflection w_{max} of a viscoelastic beam under uniform distributed load q.

Maximum vertical Deflection w_{max}						
	clamped-clamped			pinned-pinned		
Time,t	HBTQD	HBTCB	HBTQI	HBTQD	HBTCB	HBTQI
0	0.0164	0.0165	0.0165	0.0736	0.0737	0.0737
200	0.0194	0.0195	0.0195	0.0864	0.0865	0.0865
400	0.0197	0.0198	0.0198	0.0878	0.0878	0.0878
600	0.0199	0.0200	0.0200	0.0886	0.0887	0.0887
800	0.0201	0.0202	0.0202	0.0893	0.0893	0.0893
1000	0.0202	0.0203	0.0203	0.0898	0.0899	0.0899
1200	0.0203	0.0204	0.0205	0.0903	0.0904	0.0904
1400	0.0204	0.0206	0.0206	0.0908	0.0908	0.0908
1600	0.0205	0.0206	0.0206	0.0912	0.0912	0.0911
1800	0.0206	0.0207	0.0207	0.0915	0.0915	0.0915

3.7.4 Fully-transient: loading, initial and boundary conditions

For this we consider a simple two-element Maxwell model studied by Chen [22]. It consists of a linear spring and a linear viscous dashpot connected in series as shown in the inset of Fig. 4.3. The Prony series parameters are $E_1 = 9.8 \times 10^7\,\text{N/m}^2$, $\eta = 2.744 \times 10^9\,\text{N - sec/m}^2$, with the modulus time constant as $\tau_l^E = \eta/E_1$. The relaxation modulus in the Prony series is given by $E\,(t) = E_1 e^{-t/\tau_l^E}$. The material density and Poisson ratio are taken as $\rho = 500\,\text{kg/m}^3$, $\nu = 0.3$ respectively. A simply supported viscoelastic beam of length $L = 10\,\text{m}$, breadth $b = 2\,\text{m}$ and height $h = 0.5\,\text{m}$ is considered. The computational domain is reduced by taking advantage of the symmetry about $x = L/2$. The beam is subjected to a uniformly distributed vertical load $q(t) = q_0 H(t)\,\text{N/m}$ where $q_0 = 10$. For this analysis, we consider 2 HBTQI elements (11 nodes) with boundary conditions as given in Eq. (3.48). The initial conditions are taken as $\{u\}_0 = \{\dot{u}\}_0 = \{\ddot{u}\}_0 = 0$. For Newmark's scheme we take $\alpha = 0.5$ and $\gamma = 0.5$, which gives a stable scheme (i.e.,there is no restriction on the value of the time step used). The total time is taken as 20.0 s with the number of time steps as 500. The Fig. 4.3 shows the dynamic response of the beam for viscoelastic and elastic cases with different values of damping coefficients. As expected, the dynamic response of the viscoelastic beam disappeared and reached a steady state after certain period due to damping. Also, as the damping coefficient is increased the beam deflection decreased.

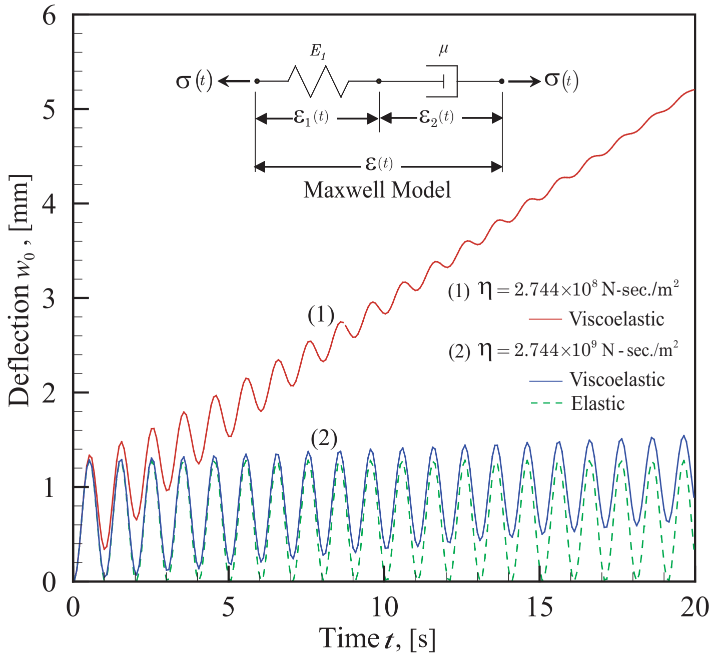


Figure 3.3: Fully-transient maximum vertical deflection w_{max}, of a hinged-hinged beam under time-dependent transverse loading $q(t)$.

4. A GEOMETRICALLY NON-LINEAR ANALYSIS OF ISOTROPIC AND FUNCTIONALLY GRADED SHELL STRUCTURES UNDER THERMAL AND MECHANICAL LOADS

In the history of finite element analysis of structures, the shells have long been a fascinating and a complicated area of research. Shells are three-dimensional bodies in which one geometric dimension is significantly smaller in comparison to the other two dimensions. Hence, they can be easily fabricated into complicated shapes and geometries and actually have very little material volume. In spite of this, the shell structures can sustain large loads. However, even small changes in loadings, boundary conditions or geometry can result in unpredictable deformations of the structure. This behavior makes it difficult to mathematically describe shell deformation geometries and also to analyse them using numerical models.

In this section, we consider large deformation analysis of isotropic and functionally graded (FG) elastic shell structures subjected to mechanical and thermal loadings. Functionally Graded Materials (FGMs) are usually bi-phasic, composite materials, microscopically inhomogeneous, in which the mechanical properties vary smoothly and continuously through the thickness from one interface to the other [54]. The gradation of the material properties through the thickness is assumed to vary continuously according to a power-law distribution based on the volume fraction of the constituents. Due to smooth variation of the material properties, the FGMs avoid severe stress concentrations, changes in displacement distributions and other singularities that are typically exhibited by composites at interfaces of lamina due to abrupt transitions in material compositions and properties. They are typically made from isotropic components, such as metals and ceramics and are designed to maximize the strength and toughness properties of the former and the thermal and corrosion resistance attributes of the latter. Hence, they are used in turbines, oil refineries, nuclear reactors, semiconductor packaging industry and high temperature aerospace environments.

Many of the recent developments in the area of locking-free shell element formulations have been in the context of lower-order elements and mixed variational principles. It is well-known that the standard displacement-based lower-order shell elements become too stiff and suffer from various forms of locking. The locking phenomena arises due to inconsistencies in representing the transverse shear energy and membrane energy. The dominant trend to overcome locking in lower-order finite element formulations of shells is to use *ad-hoc* techniques like assumed natural strain elements (Dvorkin and Bathe [28] and Hinton and Huang [38]) and the enhanced assumed strain elements (Simo and Rifai [103]). Alternate to the lower-order mixed formulations, the use of high-order interpolations have been proposed for the analysis of shells. In recent years, the higher-order finite element formulations are implemented by Pontaza and Reddy [77, 79] (using least-squares finite element formulations) and Arciniega and Reddy [4, 3] (using tensor-based weak-form Galerkin finite element formulations). In the present work, we utilize isoparametric approximation to describe the mid-surface of a given shell element using higher-order spectral/*hp* quadrilateral finite elements in a purely displacement based setting. This constitutes an important departure from the tensor-based shell finite element formulation proposed previously in the work of Arciniega and Reddy [4, 3], where a *chart* was employed to insure exact parameterization of the shell mid-surface. The use of high-order spectral/*hp* interpolants in the numerical implementation naturally leads to a finite element model that is completely locking free. Also, the use of high-order polynomial expansions in the parameterization of a given element geometry also allows for extremely accurate approximations of arbitrary shell geometries. It also, allows us to freely adopt skewed and arbitrarily curved quadrilateral shell elements in actual finite element simulations.

Here, we consider an improved first-order shear deformation theory with seven independent parameters for large deformation analysis of thin and thick isotropic and functionally graded elastic shell structures subjected to thermal and mechanical loadings. These models naturally circumvents the need for a rotation tensor in the kinematical description and allows us to use fully three-dimensional constitutive equations in the numerical implementation. As a result, complex material models may be adopted without the need for quasi-projection onto the plane-stress subspace. In this model, the transverse displacement is expanded up to a quadratic term, which essentially mitigates Poisson locking when three-dimensional constitutive equations are adopted [12]. Some of the notable early works on this is done by Sansour [99] and Bischoff and Ramm [12, 13]. The application of higher-order finite element formulations for the thermoelastic analysis of functionally graded shells with finite deformations are very limited [3].

The non-linear incremental equilibrium equations, resulting from the application of the principle of virtual work, are set using a total Lagrangian formulation, where the nodal displacements are referred to the initial structure configuration. The solution of the non-linear equation system is accomplished using the incremental/iterative Newton's method as well as some modified forms of it, like the arc-length methods for "snap-through" phenomena, where the classical Newton's method is not suitable to obtain the full structure equilibrium path (see [94, 95, 25, 56]).

4.1 Assumed seven-parameter displacement field

The mathematical background utilized in the following derivation is given in the books of Naghdi [67, 66], Bathe [20] and Wempner [124] and recently in the works of Reddy [4, 73]. The displacement of a material point from the reference configuration to the current configuration may be expressed in the usual manner as (see Fig. 4.1)

$$\mathbf{u}(\mathbf{X}, t) = \mathbf{x}(\mathbf{X}, t) - \mathbf{X} \tag{4.1}$$

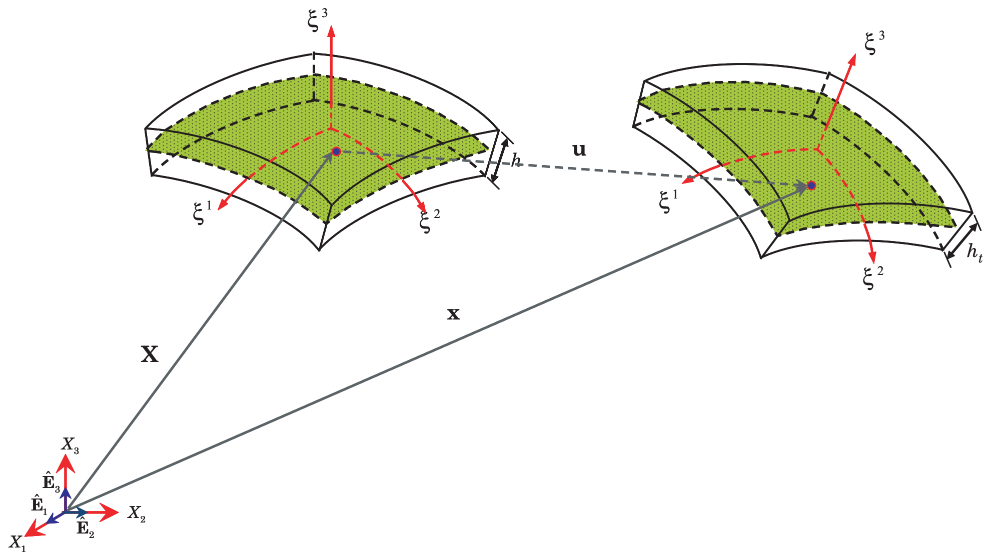


Figure 4.1: The displacement of a material point from reference configuration to the current configuration.

As done by Sansour, Bischoff, Ramm and Reddy in [99, 12, 13, 4, 73], we restrict the Taylor series approximation for **u** to the following seven-parameter expansion, so that the resulting mathematical model is consistent with three-dimensional solid mechanics [20]. The displacement field is considered as a linear expansion of the thickness coordinate around the mid-surface. The transverse displacement is parabolic through the thickness of the shell

$$\mathbf{u}(\xi^i) = \underline{\mathbf{u}}(\xi^\alpha) + \xi^3 \frac{h}{2} \boldsymbol{\varphi}(\xi^\alpha) + (\xi^3)^2 \frac{h}{2} \boldsymbol{\psi}(\xi^\alpha) \tag{4.2}$$

Note, in the above equation and throughout this section, the Latin indices (i, j) belong to the three-dimensional space ranging from 1 to 3 and the Greek indices belong to the mid-surface (α, β) ranging from 1 to 2. The generalized displacements $\underline{\mathbf{u}}$, $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$ may be expressed as

$$\underline{\mathbf{u}}(\xi^\alpha) = \underline{u}_i(\xi^\alpha)\hat{\mathbf{E}}_i, \qquad \boldsymbol{\varphi}(\xi^\alpha) = \varphi_i(\xi^\alpha)\hat{\mathbf{E}}_i, \qquad \boldsymbol{\psi}(\xi^\alpha) = \Psi(\xi^\alpha)\hat{\mathbf{n}}(\xi^\alpha) \tag{4.3}$$

The quantity $\underline{\mathbf{u}}$ represents the mid-plane displacement and $\boldsymbol{\varphi}$ is the so-called difference vector (which gives the change in the mid-surface director). The seventh parameter Ψ is included to circumvent spurious stresses in the thickness direction, caused in the six-parameter formulation by an artificial constant normal strain (a phenomena referred to as Poisson locking [12]). The configuration of the shell can be uniquely expressed in terms of the displacement vector $\underline{\mathbf{u}}$ of the mid-surface together with the difference vector $\boldsymbol{\varphi}$ and the additional variable Ψ, or by seven independent components of these vectors.

The position vector of the deformed shell at the current time t can be obtained by substituting the Eq. (4.2) into Eq. (4.1), which on rearrangement yields

$$\mathbf{x} = \mathbf{X} + \mathbf{u} = \underline{\mathbf{x}} + \xi^3 \frac{h}{2} \hat{\hat{\mathbf{n}}} + (\xi^3)^2 \frac{h}{2} \Psi \hat{\mathbf{n}} \tag{4.4}$$

where $\underline{\mathbf{x}} = \underline{\mathbf{X}} + \underline{\mathbf{u}}$ (a point on the deformed mid-surface) and $\hat{\hat{\mathbf{n}}} = \hat{\mathbf{n}} + \boldsymbol{\varphi}$ (a *pseudo*-director associated with the deformed mid-surface). It is important to note that unlike $\hat{\mathbf{n}}$; the director $\hat{\hat{\mathbf{n}}}$ is in general neither a unit vector nor is it normal to the deformed mid-surface. We define the finite element approximation of the displacement field given by Eq. (4.2) as

$$\mathbf{u}(\xi^i) = \sum_{k=1}^{n} \psi_k(\xi^1, \xi^2) \left(\underline{\mathbf{u}}^k + \xi^3 \frac{h}{2} \boldsymbol{\varphi}^k + (\xi^3)^2 \frac{h}{2} \Psi^k \hat{\mathbf{n}}(\xi^\alpha) \right) \tag{4.5}$$

where $\hat{\mathbf{n}}(\xi^\alpha)$ is the finite element approximation of the unit normal defined within a given element as

$$\hat{\mathbf{n}} = \sum_{k=1}^{n} \psi_k(\xi^1, \xi^2) \hat{\mathbf{n}}^k \tag{4.6}$$

4.2 Isoparametric characterization of shell geometry

The characterization of a shell structure is done in two steps, one by the definition of the geometry of mid-surface and next by specifying the stretch in the thickness direction. For the first step, in previous applications of higher-order finite element formulations to shell structures, Arciniega and Reddy [4, 3] have employed a *chart* to insure exact parameterization of the shell mid-surface. However, in this work we dispense with the idea of exact parametrization of mid-surface and instead use the isoparametric characterization of the mid-surface as

$$\underline{\mathbf{X}} = \boldsymbol{\phi}^e(\xi^1, \xi^2) = \sum_{k=1}^{n} \psi_k(\xi^1, \xi^2) \underline{\mathbf{X}}^k \qquad \text{in } \hat{\Omega}^e \tag{4.7}$$

within a given element, where $\underline{\mathbf{X}}$ represents a point on the approximate mid-surface and ψ_k are the two-dimensional spectral/*hp* basis functions. Next in the second step, the fully three-dimensional geometry of the undeformed configuration of a typical shell element can be obtained by

$$\mathbf{X} = \mathbf{\Phi}^e(\xi^1, \xi^2, \xi^3) = \boldsymbol{\phi}^e(\xi^1, \xi^2) + \xi^3 \frac{h}{2}\hat{\mathbf{n}} = \sum_{k=1}^{n} \psi_k(\xi^1, \xi^2)\Big(\underline{\mathbf{X}}^k + \xi^3 \frac{h}{2}\hat{\mathbf{n}}^k\Big) \tag{4.8}$$

where $\xi^3 \in [-1, +1]$ and $\hat{\mathbf{n}}$ is the finite element approximation of the unit normal defined within a given element by Eq. (4.10). Thus the present formulation requires as input the three-dimensional coordinates of the shell mid-surface ($\underline{\mathbf{X}}$) as well as a set of directors (i.e., unit normal vectors ($\hat{\mathbf{n}}$) to the mid-surface), for each node in the shell finite element model. As a result, the actual shell mid-surface as well as the unit normal to the shell mid-surface, are each approximated using the standard spectral/*hp* finite element interpolation functions within a given shell element. The use of high-order polynomial expansions in the parametrization of a given element geometry also allows for extremely accurate approximations of arbitrary shell geometries.

To describe the various kinematics of deformation of shells it is necessary to establish curvilinear basis vectors. In Fig. 4.2(a), we show the side view of three-dimensional geometry of the undeformed configuration of a typical shell element. Here, we identify a point O on the mid-surface of the shell and a point A directly above O in the direction of the unit normal $\hat{\mathbf{n}}$. At each point, like O, on the mid-surface of a given element we define a set of covariant basis vectors

$$\mathbf{a}_\alpha = \frac{\partial \underline{\mathbf{X}}}{\partial \xi^\alpha} \equiv \underline{\mathbf{X}}_{,\alpha} \tag{4.9}$$

these are tangent to the mid-surface (see Fig. 4.2(b)) and are linearly independent and thus form a local curvilinear basis. The normal vector $\mathbf{a}_3$ may be defined as $\mathbf{a}_3 = \mathbf{a}_1 \times \mathbf{a}_2$. We see that for each $(\xi^1, \xi^2) \in \hat{\Omega}^e$ (Ω^e is mid-surface), the vectors $\mathbf{a}_i$

define a basis for $\mathbb{R}^3$. In the current work, we will be largely unconcerned with $\mathbf{a}_3$ and instead utilize a finite element approximation of the unit normal defined within a given element as

$$\hat{\mathbf{n}} = \sum_{k=1}^{n} \psi_k(\xi^1, \xi^2)\hat{\mathbf{n}}^k \tag{4.10}$$

At any point in the shell element, like A, we define a set of covariant basis vectors

$$\mathbf{g}_i = \frac{\partial \mathbf{X}}{\partial \xi^i} \equiv \mathbf{X}_{,i} \tag{4.11}$$

Using Eq. (4.8) allows us to express the shell basis vectors as

$$\mathbf{g}_\alpha = \mathbf{a}_\alpha + \xi^3 \frac{h}{2}\hat{\mathbf{n}}_{,\alpha}, \qquad \mathbf{g}_3 = \frac{h}{2}\hat{\mathbf{n}} \tag{4.12}$$

In Figure 4.2(c) we provide an illustration of the vectors $\mathbf{a}_\alpha$ and $\mathbf{g}_\alpha$ at points O and A respectively, in a typical shell element.

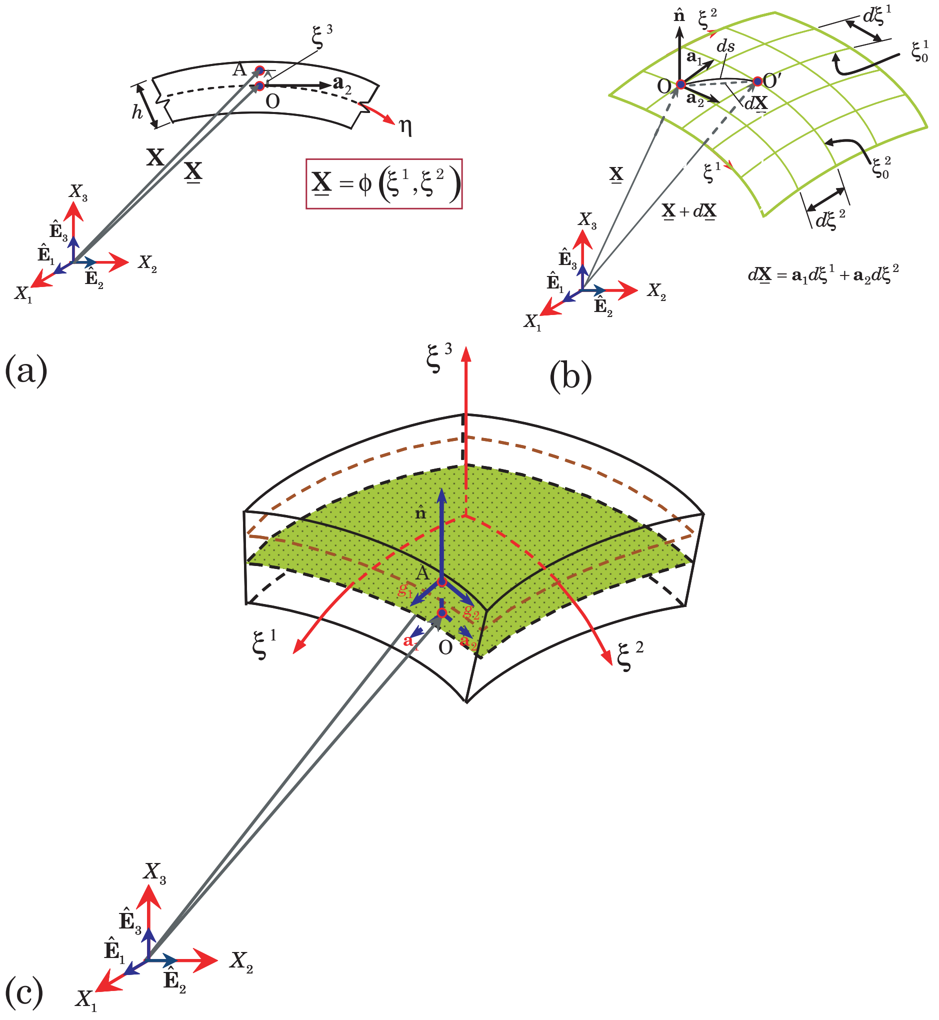


Figure 4.2: (a) Side-view (b) Mid-surface (c) Isometric-view of a typical shell finite element in the reference configuration. The basis vectors $\mathbf{a}_\alpha$ and $\mathbf{g}_\alpha$ as well as the finite element representation of the unit normal $\hat{\mathbf{n}}$ are also shown.

4.3 Strain measure

Here we use the Lagrangian description and all the kinematic variables are referred with respect to the initial configuration that the body occupies at time $t = 0$. Since the Green–Lagrange strain measure is suited for material description we introduce the strain tensor $\mathbf{E}$ as (see Reddy [93])

$$\begin{aligned}\mathbf{E} &= \frac{1}{2}\Big(\mathbf{C} - \mathbf{I}\Big)\\ &= \frac{1}{2}\Big(\mathbf{F}^{\mathrm{T}} \cdot \mathbf{F} - \mathbf{I}\Big)\\ &= \frac{1}{2}\Big(\mathbf{u}_{,i} \cdot \mathbf{g}_j + \mathbf{g}_i \cdot \mathbf{u}_{,j} + \mathbf{u}_{,i} \cdot \mathbf{u}_{,j}\Big)\mathbf{g}^i\mathbf{g}^j\end{aligned} \tag{4.13}$$

where $\mathbf{F} = (\nabla_0 \mathbf{x})^{\mathrm{T}}$ is the deformation gradient, ∇_0 is the material gradient operator, and $\mathbf{C} = \mathbf{F}^{\mathrm{T}} \cdot \mathbf{F}$, is the *right Cauchy-Green tensor* that is symmetric and positive definite (see Reddy [93]). Hence the strain becomes a symmetric tensor by definition. The strain tensor $\mathbf{E}$ appearing in Eq. (4.13) can be expanded in terms of the thickness coordinate ξ^3 as

$$E_{ij}(\xi^m) = \varepsilon_{ij}^{(0)} + \xi^3 \varepsilon_{ij}^{(1)} + \mathbf{HOT} \tag{4.14}$$

In the present formulation we neglect all covariant components of $\mathbf{E}$ that are quadratic and higher-order terms in ξ^3 (see Eq. (4.14)).

4.4 Functionally graded shells

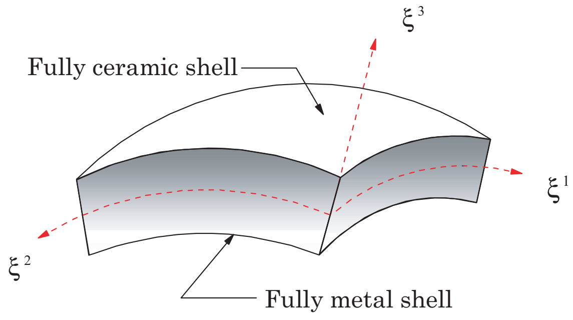
For functionally graded structures, we assume that the shell is composed of two isotropic constituents, mainly ceramics and metals. The ceramic constituent provides heat and corrosion resistance while the metallic constituent provides strength, toughness and ductility necessary to prevent fractures due to high-temperature gradients. In bi-phasic FGMs (see Fig. 4.3(a)), the properties are assumed to vary smoothly through the thickness of the shell. We also assume a rule of mixtures based on the Voigt-model [107]. In such cases, the composite modulus is given by the weighted average of the moduli of the constituents, that varies with respect to the shell thickness coordinate ξ^3 as

$$E(\xi^3) = (E^+ - E^-)f^+(\xi^3) + E^- \tag{4.15}$$

where

$$f^+(\xi^3) = \left(\frac{\xi^3 + 1}{2}\right)^n \tag{4.16}$$

The quantities E^- and E^+ constitute the moduli at the bottom ($\xi^3 = -1$) and top ($\xi^3 = +1$) surfaces of the shell respectively and f^+ is the volume fraction of the phase at top ($\xi^3 = +1$) surface of the shell. The Eq. (4.15) constitutes a power-law variation of E through the shell thickness (see Reddy [86, 81]). In Eq. (4.16), n is the constant volume fraction exponent in the range $0 \leq n \leq \infty$. The value of $n = 0$ represents a purely ceramic shell. Conversely, we have a purely metallic shell as $n \to \infty$ (see Fig.4.3(b)).



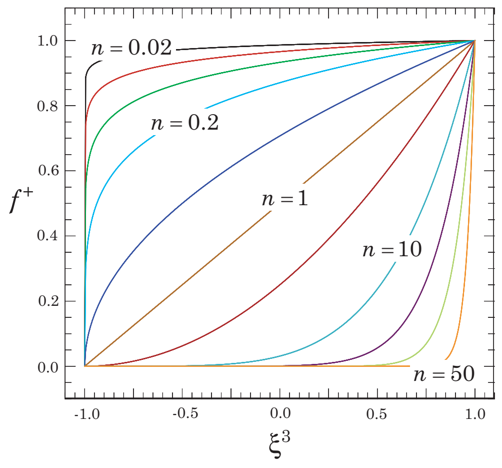


Figure 4.3: (a) Functionally graded shell and (b) Variation of volume fraction function f^+ through thickness for different values of power-law index n.

4.5 Thermal analysis

Here we consider the thermal analysis of the shell by imposing constant surface temperatures at the top and bottom surfaces of the FGM shell. The variation of temperature is assumed to occur in the thickness direction only. To determine the thermal stresses, the temperature distribution across the thickness of cylinder should be obtained. The differential equation governing the steady-state heat transfer in a FG shell can be expressed in the natural co-ordinate system as a function of ξ^3 as follows:

$$-\frac{\partial}{\partial \xi^3}\left(\frac{K(\xi^3)\partial T(\xi^1,\xi^2,\xi^3)}{\partial \xi^3}\right) = 0 \tag{4.17}$$

with $T(\xi^1,\xi^2,+1) = T_{Top}(\xi^1,\xi^2)$ and $T(\xi^1,\xi^2,-1) = T_{Bot}(\xi^1,\xi^2)$. As with the Young's moduli of FG shell, the thermal conductivity K is assumed to vary as

$$K(\xi^3) = (K^+ - K^-)f^+(\xi^3) + K^- \tag{4.18}$$

and f^+ is the volume fraction of the phase at top ($\xi^3 = +1$) surface of the shell given by Eq. (4.15).

Due to the dependency of the thermal conductivity coefficient on the thickness coordinate ξ^3, the Eq . (4.17) becomes non-linear and it yields a non-linear temperature distribution in the thickness direction of the shell element. Hence, there is no closed-form solution to it and the solution is usually obtained by a polynomial series as given in [43, 59]. Taking the first seven terms of the series, the thermal distribution across the thickness of shell is given by

$$T(\xi^3) = T_{Bot} + \frac{T_{Top} - T_{Bot}}{D} \sum_{j=0}^{5} \frac{1}{(jn+1)} \left(\frac{K^- - K^+}{K^-}\right)^j \left(\frac{1+\xi^3}{2}\right)^{(jn+1)} \tag{4.19}$$

where

$$D = \sum_{j=0}^{5} \frac{1}{(jn+1)} \left(\frac{K^- - K^+}{K^-}\right)^j \tag{4.20}$$

Once the temperature variation along the thickness of the shell is obtained the thermal analysis can be performed easily by including the thermal strains as shown in the next section.

4.6 Thermo-elastic constitutive equations

The temperature field imposed on the FGM shell gives rise to additional thermal strains. The thermal strains due to the temperature gradient is given by

$$\varepsilon^{(\mathrm{T})} = \boldsymbol{\alpha}\,\Delta T \tag{4.21}$$

where $\boldsymbol{\alpha}$ is the second-order tensor of the coefficients of the thermal expansion of the material. And $\Delta T = T(\xi^3) - T_0$, where $T(\xi^3)$ is the solution from Eq . (4.17) and T_0 is the assumed stress-free temperature of $T_0 = 0\,°\mathrm{C}$. For an isotropic Hookean material, in an arbitrary curvilinear system the components of strain tensor and thermal-expansion tensors can be represented as (see [124])

$$\varepsilon_{kl}^{(\mathrm{T})} = \alpha g_{kl} \Delta T \tag{4.22}$$

note, in the above the matrix representation of the thermal conductivity tensor is diagonal and $g_{kl} = \mathbf{g}_k \cdot \mathbf{g}_l$ are the covariant components of the *Riemannian metric* tensor $\mathbf{G}$ in the reference configuration. For a FGM, the coefficient of thermal conductivity is assumed to vary as

$$\alpha(\xi^3) = (\alpha^+ - \alpha^-) f^+(\xi^3) + \alpha^- \tag{4.23}$$

and f^+ is the volume fraction of the phase at top ($\xi^3 = +1$) surface of the shell given by Eq. (4.15).

In this study the materials are assumed to be perfectly elastic throughout the deformation and the Poisson's ratios ν is assumed to be constant. Also, all the material properties are considered to be temperature independent. Using these material properties and treating an un-coupled displacement-temperature behaviour, the stress-strain relation can be written as

$$\mathbf{S} = \mathbb{C} : (\mathbf{E} - \varepsilon^{(\mathrm{T})}) \tag{4.24}$$

where $\mathbf{S}$ is the second Piola Kirchhoff stress tensor, $\mathbf{E}$ is the Green-Lagrange strain tensor given by Eq. (4.14), $\varepsilon^{(\mathrm{T})}$ is the thermal strain tensor given by Eq. (4.22) and $\mathbb{C} = \mathbb{C}^{ijkl} \mathbf{g}_i \mathbf{g}_j \mathbf{g}_k \mathbf{g}_l$ is the fourth-order elasticity tensor. For isotropic materials, the fourth-order elasticity tensor is a function of thickness co-ordinate ξ^3 and is a function of only two independent parameters as

$$\mathbb{C}^{ijkl} = \lambda g^{ij} g^{kl} + \mu (g^{ik} g^{jl} + g^{il} g^{jk}) \tag{4.25}$$

The Lamé parameters λ and μ are related to the Young's modulus E and Poisson's ratio ν by the following expressions

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)} \tag{4.26a}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{4.26b}$$

Although $\mathbb{C}$ depends on only the Lamé parameters, the 21 contravariant components associated with the matrix $[\mathbb{C}^{ijkl}]$ are in general distinct from one another. For the homogeneous case, the Young's modulus and Poisson's ratio are constant throughout the shell structure. As in the homogeneous case, functionally graded shells may also be described using Eq.(4.25) if the Lamé parameters are taken as functions of ξ^3.

4.7 Weak-form Galerkin implementation

The finite element framework is based on the principle of virtual work. The virtual work statement is nothing but the weak-form of the equilibrium equations and it is valid for linear and nonlinear stressstrain relations. Our analysis is restricted to static cases, therefore we omit the inertial terms. The principle of virtual work may be stated as follows: find $\Phi \in \mathcal{V}$ such that for all $\delta\Phi \in \mathcal{W}$ the following weak statement holds

$$\mathcal{G}(\delta\Phi, \Phi) = \delta\mathcal{W}_{\mathrm{I}}(\delta\Phi, \Phi) + \delta\mathcal{W}_{\mathrm{E}}(\delta\Phi, \Phi) \equiv 0 \tag{4.27}$$

The quantities $\delta\mathcal{W}_{\text{I}}$ and $\delta\mathcal{W}_{\text{E}}$ are the internal and external virtual work, respectively. These quantities may be defined with respect to the undeformed configuration as

$$\delta\mathcal{W}_{\text{I}} = \int_{\mathcal{B}_0} \delta\mathbf{E} : \mathbf{S} d\mathcal{B}_0 \tag{4.28a}$$

$$\delta\mathcal{W}_{\text{E}} = -\int_{\mathcal{B}_0} \delta\mathbf{u}\cdot\rho_0\mathbf{b}_0 d\mathcal{B}_0 - \int_{\Gamma_\sigma} \delta\mathbf{u}\cdot\mathbf{t}_0 ds \tag{4.28b}$$

where ρ_0 is the density, $\mathbf{b}_0$ is the body force and $\mathbf{t}_0$ is the traction vector (which are all expressed with respect to the reference configuration). Evaluation of the internal virtual work statement for the eth element of the discrete problem yields

$$\begin{aligned}\delta\mathcal{W}_I^e &= \int_{\mathcal{B}_0^e} (\delta\varepsilon^{(0)} + \xi^3\delta\varepsilon^{(1)}) : \mathbb{C} : (\varepsilon^{(0)} + \xi^3\varepsilon^{(1)} - \varepsilon^{(\text{T})}) d\mathcal{B}_0^e \\ &= \int_{\hat{\Omega}^e}\int_{-1}^{+1} \big(\delta\varepsilon_{ij}^{(0)} + \xi^3\delta\varepsilon_{ij}^{(1)}\big)\mathbb{C}^{ijkl}\big(\varepsilon_{kl}^{(0)} + \xi^3\varepsilon_{kl}^{(1)} - \varepsilon_{kl}^{(\text{T})}\big) J d\xi^3 d\hat{\Omega}^e \\ &= \int_{\hat{\Omega}^e} \Big[\mathbb{A}^{ijkl}\delta\varepsilon_{ij}^{(0)}\varepsilon_{kl}^{(0)} + \mathbb{B}^{ijkl}\big(\delta\varepsilon_{ij}^{(0)}\varepsilon_{kl}^{(1)} + \delta\varepsilon_{ij}^{(1)}\varepsilon_{kl}^{(0)}\big) + \mathbb{D}^{ijkl}\delta\varepsilon_{ij}^{(1)}\varepsilon_{kl}^{(1)} \\ &\qquad + {}^{(\text{T})}\mathbb{A}^{ijkl}\delta\varepsilon_{ij}^{(0)} g^{kl} + {}^{(\text{T})}\mathbb{B}^{ijkl}\delta\varepsilon_{ij}^{(1)} g^{kl}\Big] d\hat{\Omega}^e\end{aligned} \tag{4.29}$$

where $\int_{\hat{\Omega}^e}(\,\cdot\,)d\hat{\Omega}^e = \int_{-1}^{+1}\int_{-1}^{+1}(\,\cdot\,)d\xi^1 d\xi^2$. The quantities $\mathbb{A}^{ijkl}$, $\mathbb{B}^{ijkl}$ and $\mathbb{D}^{ijkl}$ are the contravariant components of the effective *extensional*, *bending* and *bending-extensional coupling* fourth-order stiffness tensors respectively and ${}^{(\text{T})}\mathbb{A}^{ijkl}$, ${}^{(\text{T})}\mathbb{B}^{ijkl}$ are the corresponding components from the thermal loads. These contravariant components may be determined as

$$\{\mathbb{A}^{ijkl}, \mathbb{B}^{ijkl}, \mathbb{D}^{ijkl}\} = \int_{-1}^{+1} \{1, \xi^3, (\xi^3)^2\}\mathbb{C}^{ijkl} J d\xi^3 \tag{4.30a}$$

$$\{{}^{(\text{T})}\mathbb{A}^{ijkl}, {}^{(\text{T})}\mathbb{B}^{ijkl}\} = \int_{-1}^{+1} -\alpha\Delta T\{1, \xi^3\}\mathbb{C}^{ijkl} J d\xi^3 \tag{4.30b}$$

where J is the determinant of the Jacobian. In the computer implementation, we perform the above integration numerically using the Gauss-Legendre quadrature rule (with 50 quadrature points taken along the thickness direction). Finally, we linearize the internal virtual work terms using Newton's method.

4.8 Numerical results

In this section, numerical results obtained by the model developed herein are presented for various standard shell benchmark problems. We employ Newton's method in the solution of the resulting equations. To facilitate a numerical solution for problems involving very large deformations, we further imbed the iterative Newton procedure within an incremental load stepping algorithm. A convergence criterion of 10^{-6} is adopted in all numerical examples. Highly accurate numerical results are obtained using the proposed higher-order shell formulation without the need for *ad-hoc* fixes (e.g., reduced integration, enhanced natural strain, assumed natural strain, or mixed interpolation). To show the robustness of the proposed shell formulation, all numerical examples are tested using skewed and/or arbitrarily curved quadrilateral shell elements.

4.8.1 Cantilevered strip plate under end shear force

Here, we consider deformation of a isotropic cantilevered strip plate under end shear distributed load q as shown in Fig. 4.4, where $L = 10$, $b = 1$ and $h = 0.1$. The isotropic problem with $E = 1.2 \times 10^6$ and $\nu = 0.0$ has been considered by many authors (see for example Refs. [71, 98, 114, 110, 108, 109]).
For the analysis, we employ regular and skewed finite element meshes consisting of 4 elements, with the p-level taken as 4. Each numerical simulation is conducted using the incremental/iterative Newton procedure with 50 load steps. In Fig. 4.5, we plot the undeformed and various deformed mid-surface configurations for uniform and

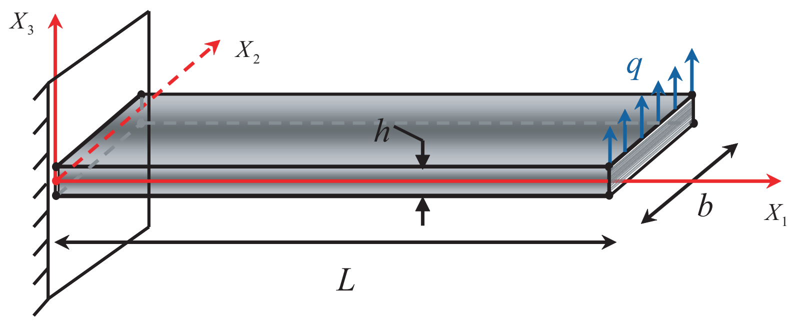


Figure 4.4: Geometry of cantilever strip plate under vertical end shear force.

skewed meshes for different shear loading stages of q. Clearly, both structures with uniform and skewed meshes undergo very large deformations which are qualitatively quite similar. For quantitative analysis, in Fig. 4.6, we plot the computed axial and transverse deflections vs. the distributed shear loading q of the plate tip using uniform and skewed meshes. The calculated deflections are in excellent agreement with the numerical results reported by Sze et al. [109]. They used ABAQUS, a commercial finite element package, which utilizes a bi-linear element with hourglass stabilization.

4.8.2 Annular slit plate under end shear force

Here, the beautiful benchmark problem considered by many in Refs. [19, 15, 8, 100, 108, 109] for isotropic case is analysed. This problem consists of a slit cantilevered annular plate as shown in Fig. 4.7 that is subjected to a line shear load q at its free end.

We take $R_i = 6$, $R_o = 10$ and $h = 0.03$. The material is isotropic with $E = 21 \times 10^6$ and $\nu = 0.0$ with $q_{max} = 0.8$. We employ uniform and arbitrarily curved quadrilateral shell elements consisting of 4 elements with the $p = 8$. Solutions obtained with the $p \leq 4$ are too stiff and suffer from locking. Each numerical simulation

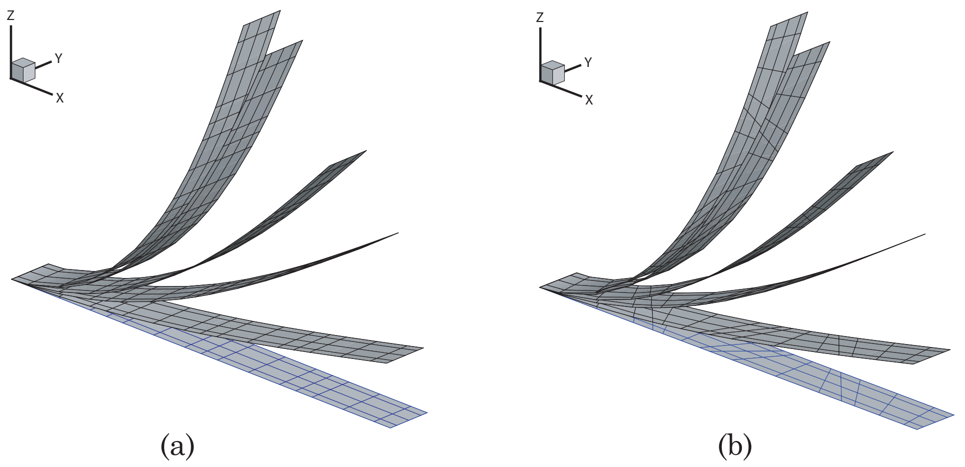


Figure 4.5: Mid-surface configurations at vertical shear force ($q = 0.0, 0.4, 1.2, 2, 4$ and 5) for (a) Uniform regular mesh (b) Skewed irregular mesh.

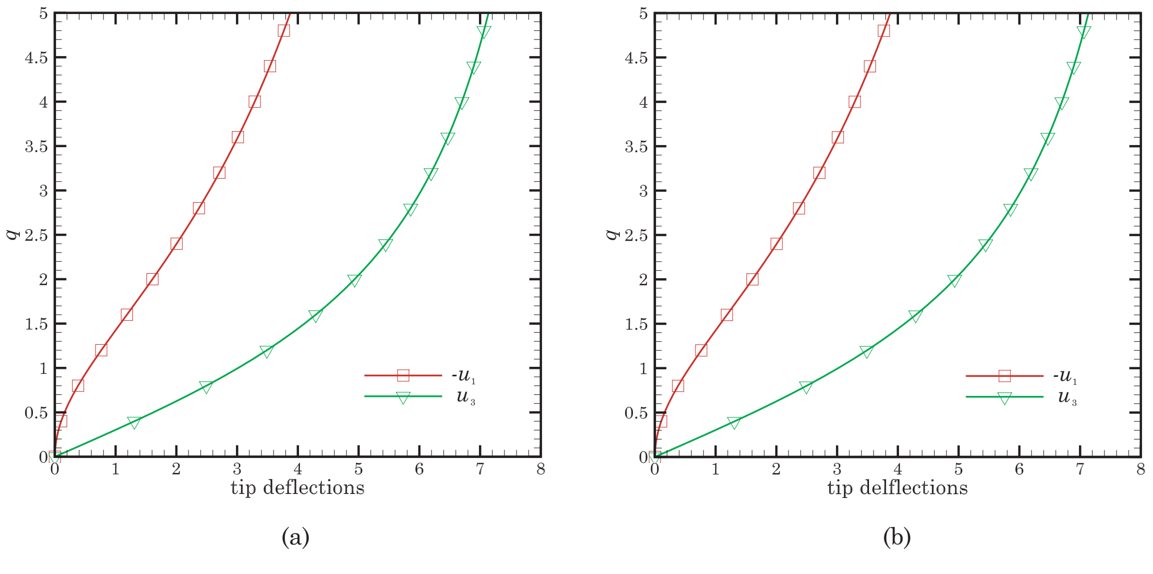


Figure 4.6: Tip deflections vs. shear load q for (a) Uniform regular mesh (b) Skewed irregular mesh.

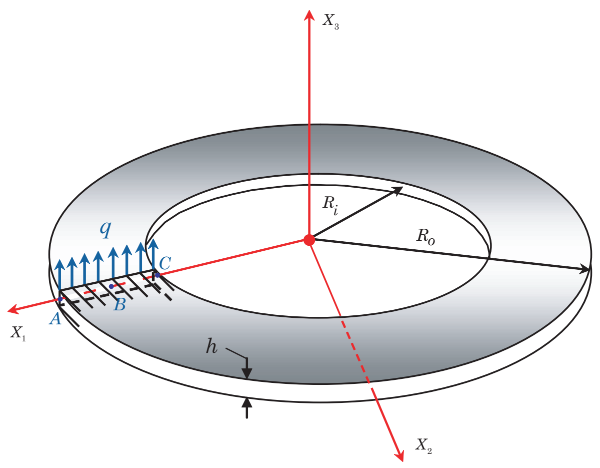


Figure 4.7: Geometry of annular slit plate under vertical shear force.

is conducted using the incremental/iterative Newton procedure with 80 equal load steps. In Fig. 4.8, we show the undeformed and various deformed mid-surface configurations for uniform and curved meshes at different stages of applied load. Clearly, both structures with uniform and skewed meshes undergo very large deformations which are qualitatively quite similar. For quantitative analysis, the transverse tip deflections vs. the net applied force $P = (R_o - R_i)q$ at three characteristic points of A, B and C are shown in Fig. 4.9 for uniform and curved meshes. The computed deflections agree very well with the tabulated displacement values reported by Sze et al. [109]. Clearly, both structures with uniform and curved meshes undergo very large deformations corresponding to maximum net applied force of $P = 3.2$. The results from uniform and curved meshes are qualitatively quite similar, this proves the robustness of the present formulation.

Next, we analyze a metal-ceramic functionally graded annular plate with the same geometry and mesh as in the above case. The metal (aluminum) is taken as the bottom material and the ceramic (zirconia) as the top constituent, with the

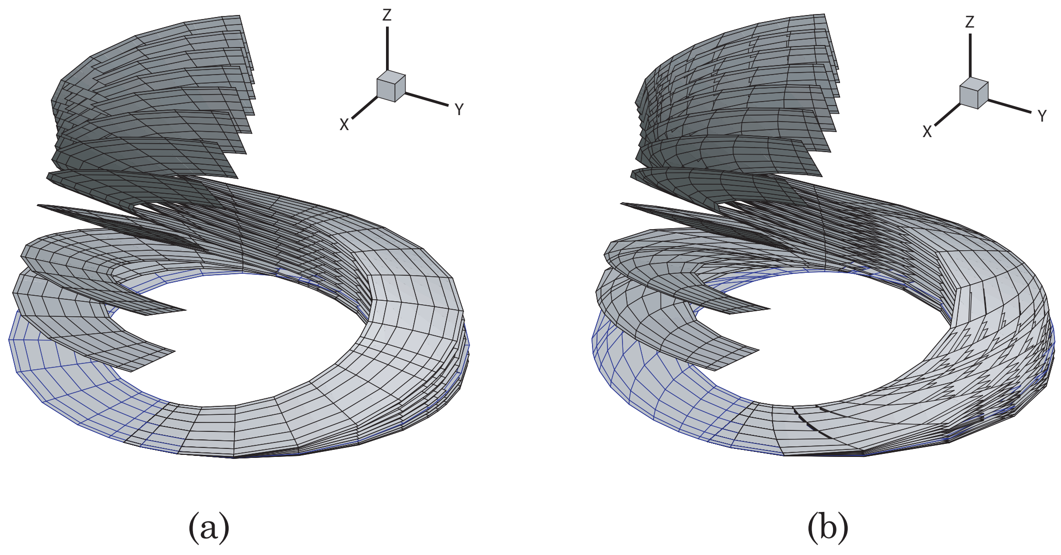


Figure 4.8: Mid-surface configurations at $P = 0.16$, 0.32, 0.64, 1.28, 1.92, 2.56 and 3.20 for (a) Uniform regular mesh (b) Curved irregular mesh.

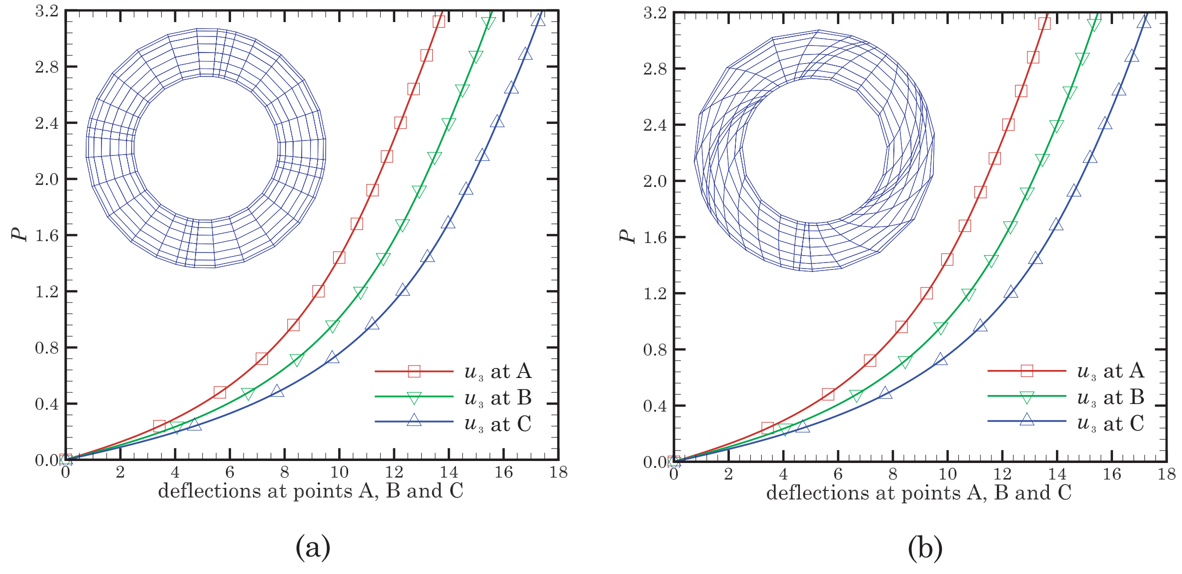
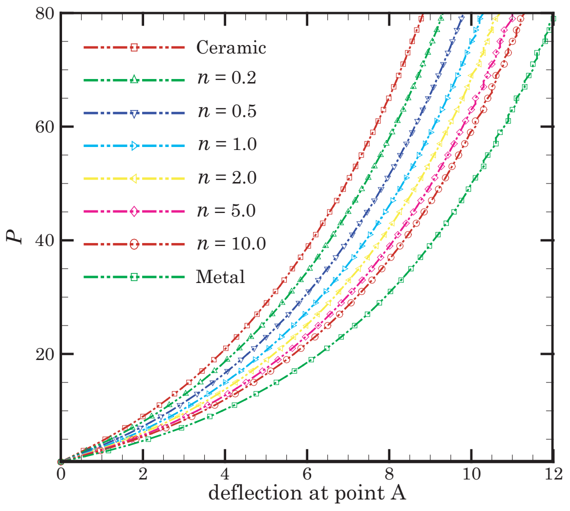


Figure 4.9: Tip deflections at points A, B and C vs. shear force P for (a) Uniform regular mesh (b) Curved irregular mesh.

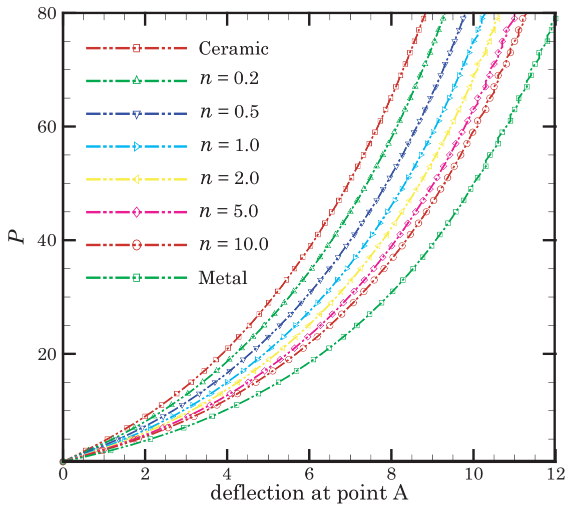
elastic properties given below

$$\begin{array}{ll} E^{-} = 70 \text{ GPa}, & E^{+} = 151 \text{ GPa} \\ \nu^{-} = 0.3, & \nu^{+} = 0.3 \end{array} ll \tag{4.31}$$

Here we consider a maximum distributed force of $q_{max} = 20$. As in the previous case the computations are performed by the NewtonRaphson method with 80 equal load steps. For quantitative analysis, the transverse tip deflections vs. the net applied force P at two extreme characteristic points of A and C are shown in Figs. 4.10 and 4.11 using uniform and curved meshes. All the load steps converged within 2-4 non-linear iterations. As expected, the bending response of FG annular plate lies in between the fully ceramic and fully metal shells. The computed deflections match exactly with the reported values by Arciniega and Reddy [4]. It is clear that the plate undergoes large deformations corresponding to maximum load of $P = 80$. The results from uniform and curved meshes for FG plate are qualitatively quite similar, this proves the robustness of the present formulation.

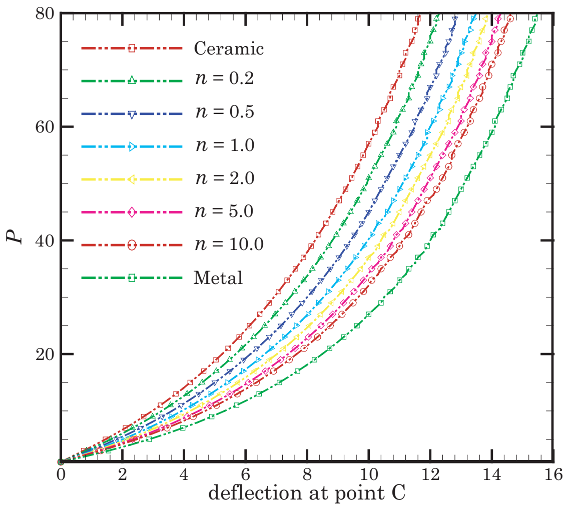


(a)

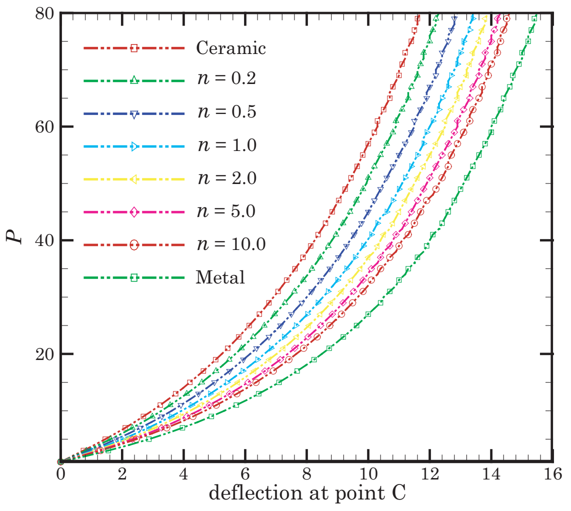


(b)

Figure 4.10: Tip deflections at point A vs. shear force P for (a) Uniform regular mesh (b) Curved irregular mesh.



(a)



(b)

Figure 4.11: Tip deflections at point C vs. shear force P for (a) Uniform regular mesh (b) Curved irregular mesh.

4.8.3 Thermoelastic analysis of a FG cantilever beam

In this example, we consider deformation of a FG cantilever beam subjected to two different thermal loading conditions as shown in Fig. 4.12, where the normalized length, thickness, and width of the beam are $L = 1$, $b = 0.05$ and $h = 0.01$.

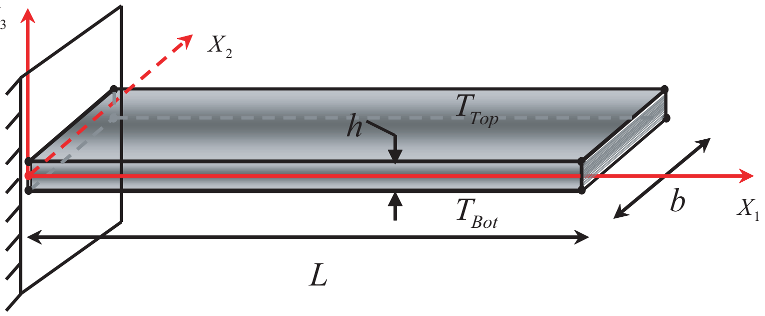


Figure 4.12: Geometry of cantilever strip plate under vertical end shear force.

The normalized thermal and mechanical properties of the bottom and top constituents of the FG beam are

$$\begin{aligned}
&E^- = 100 \times 10^6, \quad E^+ = 300 \times 10^6 \\
&\alpha^- = 5 \times 10^{-6}, \quad \alpha^+ = 2 \times 10^{-6} \quad ll \\
&K^- = 50 \times 10^{-6}, \quad K^+ = 7 \times 10^{-6}
\end{aligned} \tag{4.32}$$

The cantilever beam is subjected to through-the-thickness varying thermal fields of $T_{Top} = T_{Bot} = 50$ and $T_{Top} = -T_{Bot} = 50$. The beam considered stress free at a normalized temperature of $T = 0$. For the analysis, we employ uniform regular finite element mesh consisting of 4 elements, with the p-level taken as 4. The transverse and axial deflections variation with the volume fraction exponent n are plotted in

Figs. 4.13 and 4.14. These results match with the reported values reported in Refs. [65, 64].

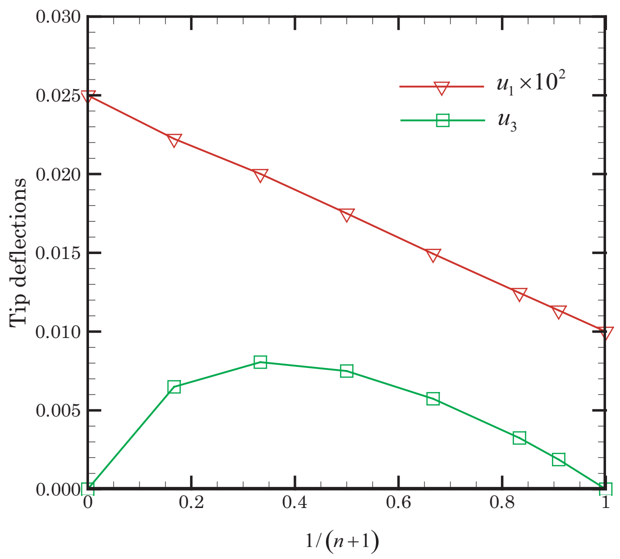


Figure 4.13: Horizontal and vertical tip deflections vs. $1/(n+1)$.

4.8.4 Thermoelastic analysis of a clamped FG circular plate

In this example, we consider deformation of a FG circular plate subjected to a through-the-thickness thermal gradient, which is a quadratic function of the radius as follows:

$$T_{Top}(r) = -T_{Bot}(r) = T_C\left[1-\left(\frac{r}{R}\right)^2\right] \tag{4.33}$$

where r is the radial parameter, $0 < r \leq R$, and $T_C = 100\,^0\mathrm{C}$. The beam considered stress free at a normalized temperature of $T = 0\,^0\mathrm{C}$. Here the radius and thickness of the plate are $R = 1.0\,\mathrm{m}$ and $h = 2\,\mathrm{mm}$. The thermal and mechanical properties

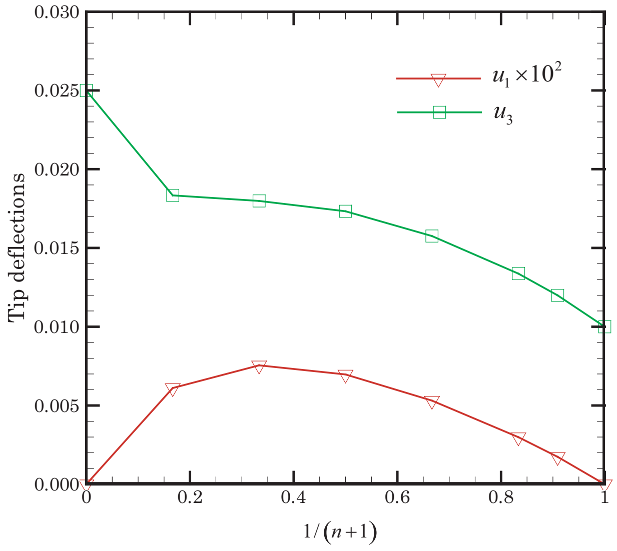


Figure 4.14: Horizontal and vertical tip deflections vs. $1/(n+1)$.

of the bottom and top constituents of the FG plate are

$$\begin{array}{ll} E^- = 100 \text{ MPa}, & E^+ = 300 \text{ MPa} \\ \alpha^- = 5\ /^0\text{C}, & \alpha^+ = 2\ /^0\text{C} \\ K^- = 50 \text{ W/m}^0\text{C}, & K^+ = 7 \text{ W/m}^0\text{C} \\ \nu^- = 0.3, & \nu^+ = 0.3 \end{array} ll \tag{4.34}$$

Since a circular plate is symmetric about its axis, all numerical simulations are conducted using one quarter of the physical domain and invoking appropriate symmetry boundary conditions as shown below in Fig. 4.15.

For the analysis, we employ non-uniform finite element mesh consisting of 8 elements along the radial direction, with the p-level taken as 5. Solutions obtained with lower-order elements are too stiff and suffer from locking. The normalized transverse displacement variation with the radius is shown in Fig. 4.16. These results match with the reported values reported in Refs. [65, 64].

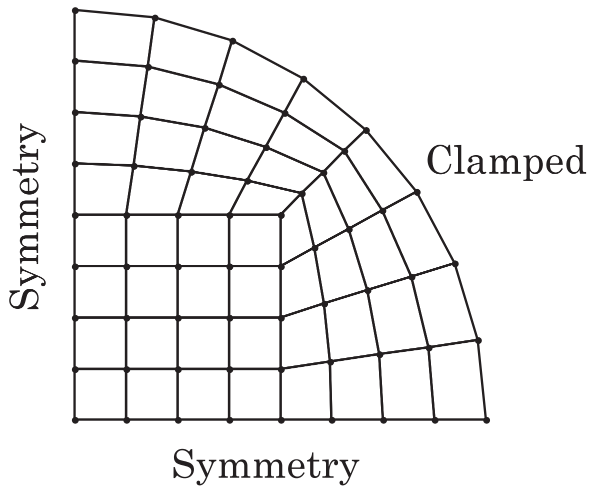


Figure 4.15: Quarter FG plate with symmetric boundary conditions.

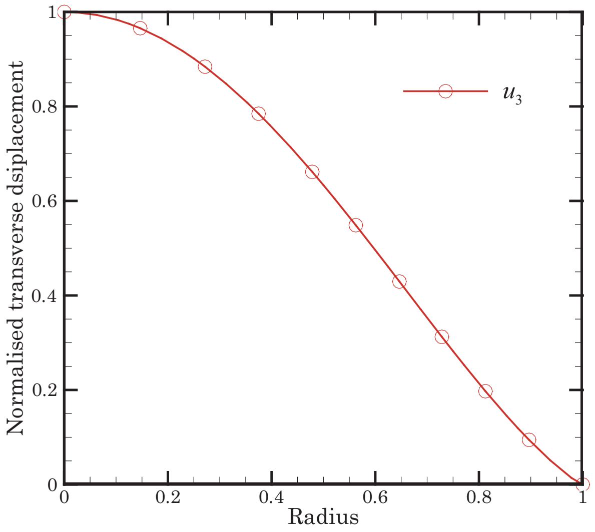


Figure 4.16: Normalized transverse tip deflections vs. radius.

4.8.5 Pull-out of an open-ended cylindrical shell

In this example, we consider the mechanical deformation of an open-ended cylinder, shown in Fig. 4.17, subjected to two pull-out point forces P. Unlike the previous example, in this problem we apply the loads such that the shell undergoes very large displacements and rotations. As a result, this problem constitutes a severe test of shell finite element formulations and has been addressed in Refs. [15, 100, 108, 109, 4] among others. The following material properties are used in the analysis of isotropic shell

$$E = 10.5 \times 10^6, \quad \nu = 0.3125 \tag{4.35}$$

The geometric parameters are taken as: $L = 10.35$, $h = 0.094$ and $R = 4.953$ (where we have taken R as the radius of the *undeformed* mid-surface as opposed to the radius of the *inner* surface of the shell).

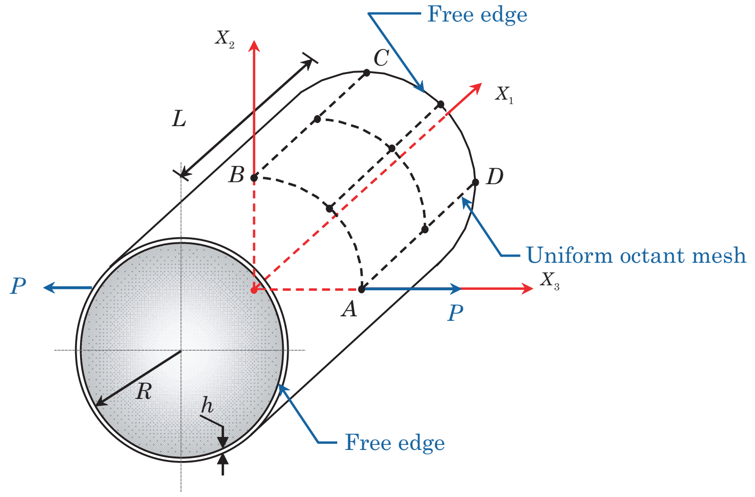


Figure 4.17: Pull-out of an open-ended cylindrical shell.

Symmetry in the geometry, material properties and loading allow us to construct the numerical model using only an octant of the actual open-ended cylinder. For the numerical model we employ a 2×2 mesh (with the p-level taken as 8) of the shell octant containing points A, B, C and D. Solutions obtained with the $p \leq 4$ are too stiff and suffer from locking. The incremental/iterative Newton procedure is adopted using a total of 80 equal load steps. For this example, all the load steps converged within 1-4 non-linear iterations. In Figs. 4.18(a)-4.18(d) we show the undeformed and various deformed mid-surface configurations for the open-ended cylindrical shell pull-out problem using uniform, skewed and curved meshes. The overall deflections and rotations are clearly quite large, especially for the final shell configuration (i.e., the case where P = 40,000). The mechanical response of the shell has two different characteristic regions: in the beginning the deformation is initially bending dominated; however, next as the load is intensified, the membrane forces clearly play an increasingly significant role, resulting in a pronounced very stiff response of the shell structure. The radial deflections vs. the net applied pulling force P are shown in Figs. 4.19(a)-4.19(c) using uniform, skewed and curved meshes for points A, B and C. The computed deflections are in excellent agreement with results of Sze et al. [109] and also Arciniega and Reddy [4].

Next, we analyze a metal-ceramic functionally graded shell with the same geometry and mesh as in the above case. The metal (aluminum) is taken as the bottom material and the ceramic (zirconia) as the top constituent, with the elastic properties given below

$$\begin{array}{ll} E^- = 70 \text{ GPa}, & E^+ = 151 \text{ GPa} \\ \nu^- = 0.3, & \nu^+ = 0.3 \end{array} ll \tag{4.36}$$

As in the previous problem due to symmetry only an octant of the actual open-ended cylinder is used for analysis. For the numerical model we employ a 2×2

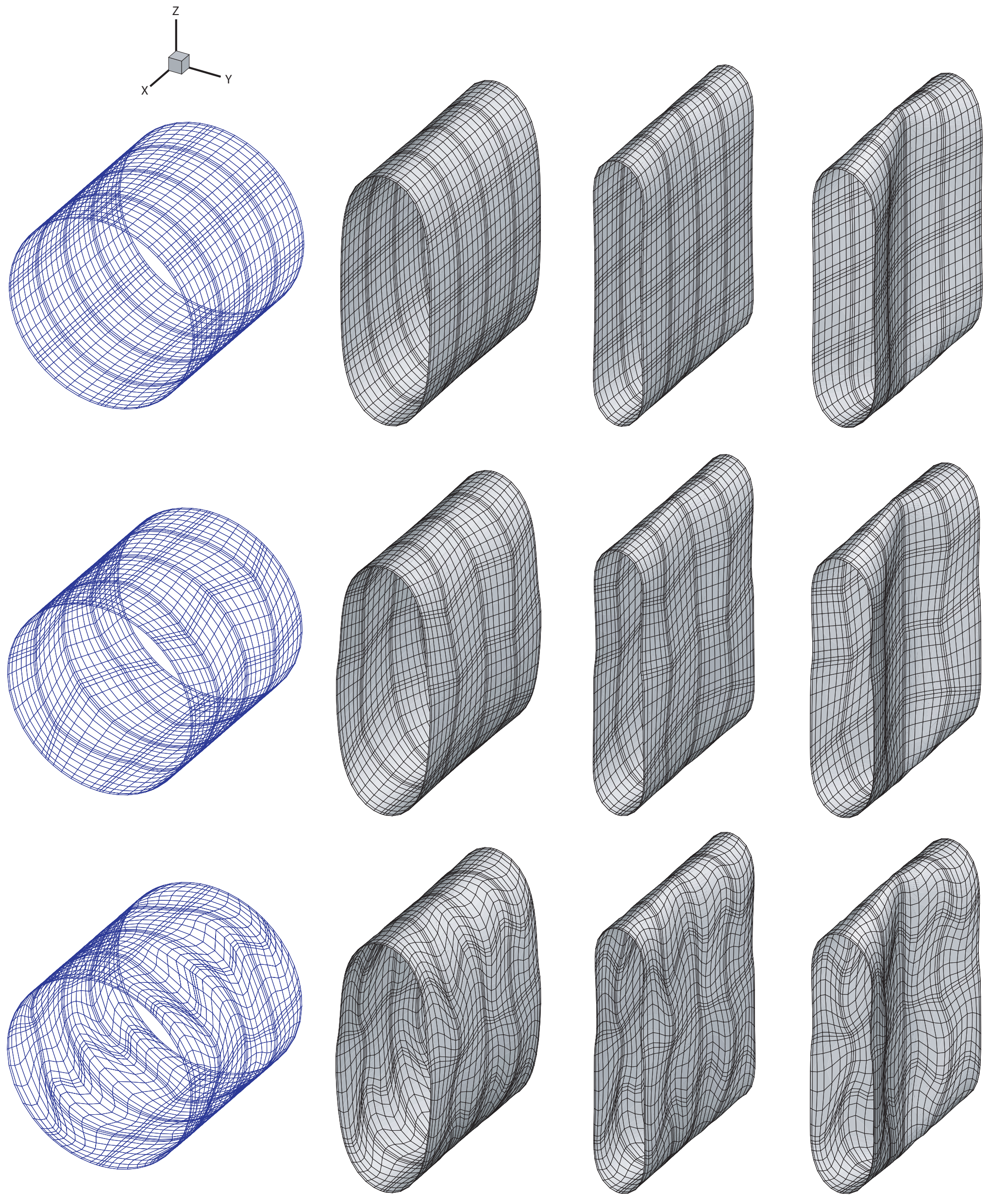
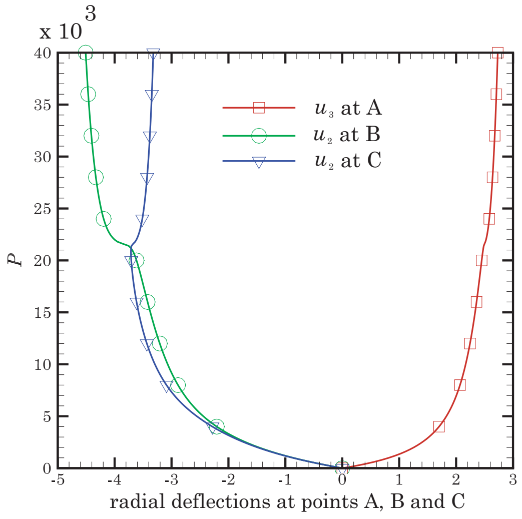
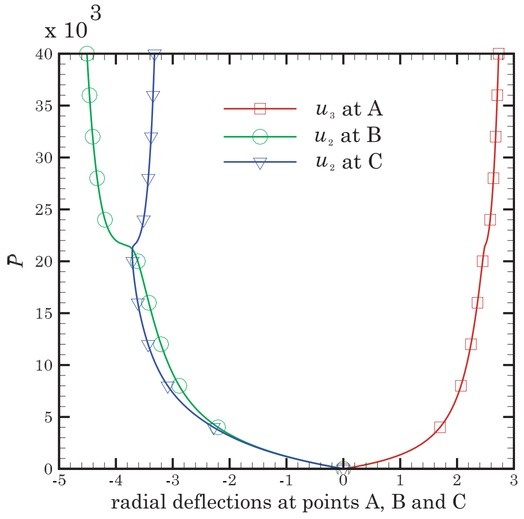


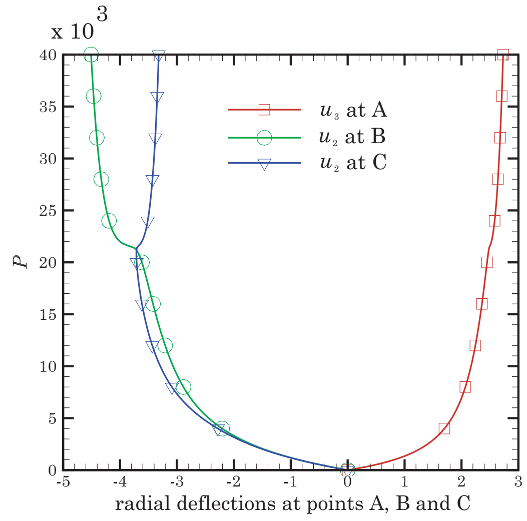
Figure 4.18: Uniform, skewed and curved mid-surface configurations at (a) $P = 0$ (b) $P = 5000$ (c) $P = 20000$ (d) $P = 40000$.



(a)



(b)



(c)

Figure 4.19: Radial deflections at points A, B and C vs. pull-out force P for (a) Uniform, (b) Skewed and (c) Curved meshes.

mesh (with the p-level taken as 8) of the shell octant containing points A, B, C and D. Solutions obtained with the $p \leq 4$ are too stiff and suffer from locking. The incremental/iterative Newton procedure is adopted using a total of 80 equal load steps. For this example, all the load steps converged within 2-5 non-linear iterations. For quantitative analysis, the radial deflections u_3 vs. the net applied pulling force P is shown in Fig. 4.20 using uniform, skewed and curved meshes for point A. In Fig. 4.21, we show the the radial deflections u_2 vs. the net applied pulling force P for point B using uniform, skewed and curved meshes. Finally in Fig. 4.22, we show the the radial deflections u_2 vs. the net applied pulling force P for point C using uniform, skewed and curved meshes. As expected, in all these cases the deformation response of FG shell lies in between the fully ceramic and fully metal shells. The computed deflections match exactly with the reported values by Sze et al. [109] and Arciniega and Reddy [4]. The results from uniform, skewed and curved meshes for FG plate are qualitatively quite similar, this proves the robustness of the present formulation.

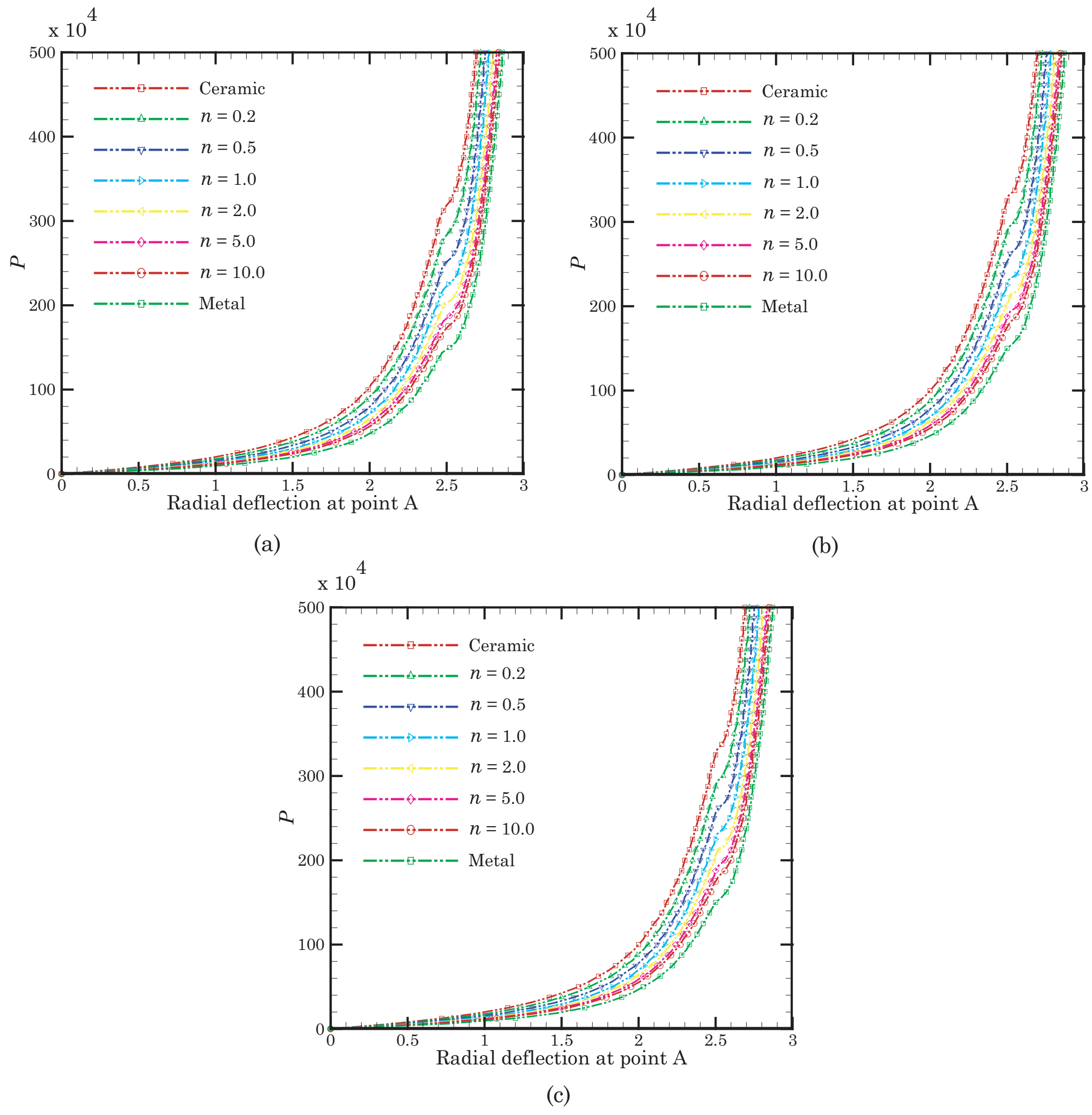
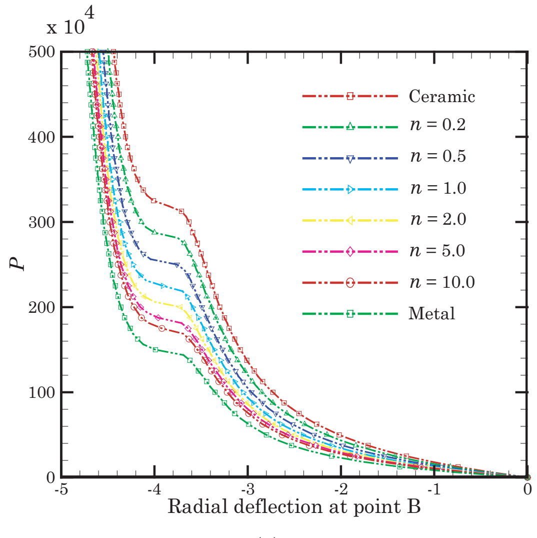
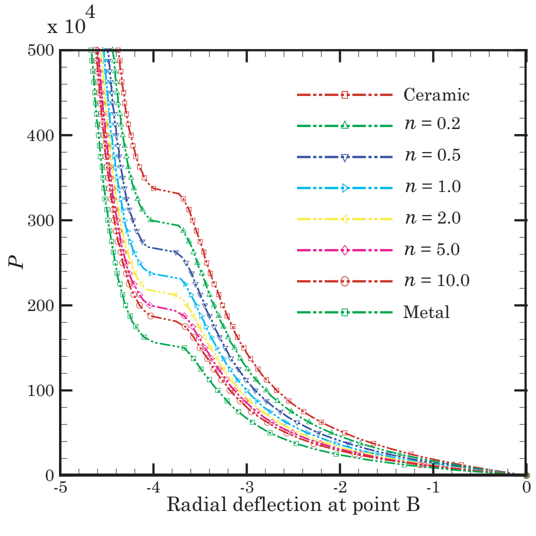


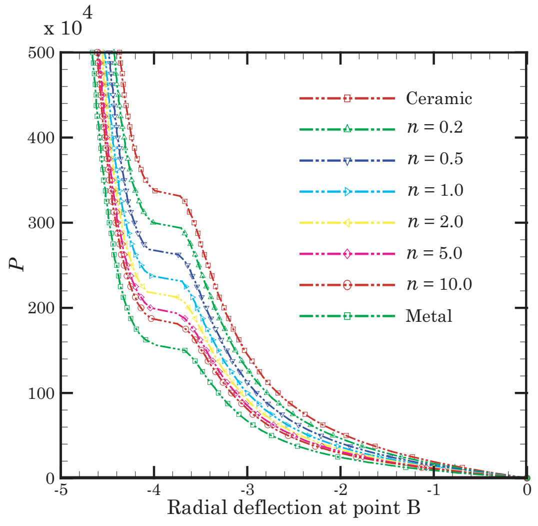
Figure 4.20: Radial deflection at point A vs. pull-out force P for (a) Uniform, (b) Skewed and (c) Curved meshes.



(a)



(b)



(c)

Figure 4.21: Radial deflection at point B vs. pull-out force P for (a) Uniform, (b) Skewed and (c) Curved meshes.

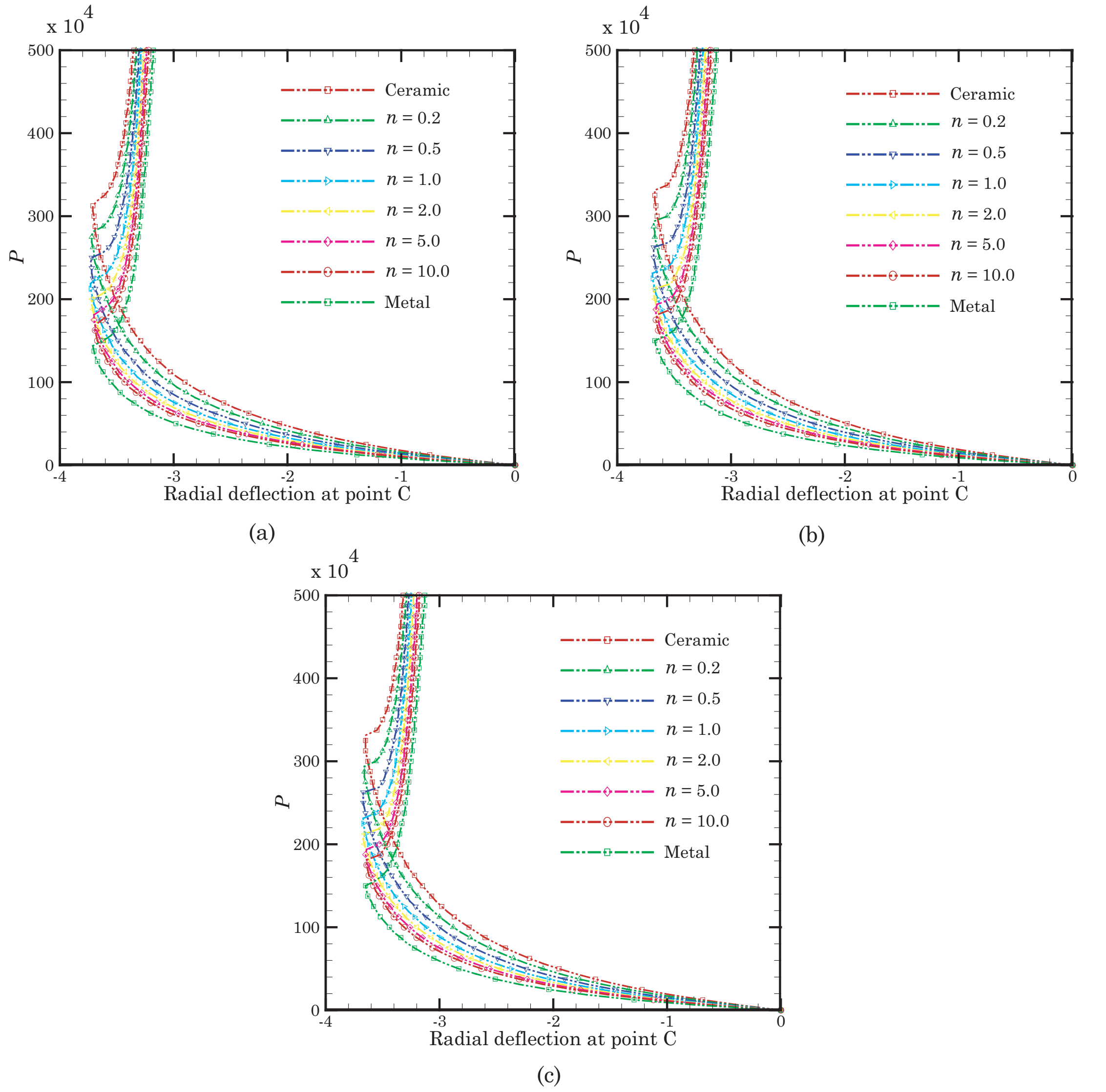


Figure 4.22: Radial deflection at point C vs. pull-out force P for (a) Uniform, (b) Skewed and (c) Curved meshes.

5. HIGHER-ORDER SPECTRAL/*HP* LEAST-SQUARES FINITE ELEMENT FORMULATIONS FOR VISCOUS INCOMPRESSIBLE FLUID FLOWS

In this section, we present least-squares finite element models for viscous, isothermal, incompressible Navier–Stokes equations using stress-based first-order system [45] and higher-order spectral/*hp* approximations. The higher-order spectral/*hp* basis functions avoid the interpolation error in the numerical schemes, thereby making them accurate and stable. For fluid flows, when combined with least-squares variational principles, the higher-order spectral/*hp* finite element technology allows us to develop efficient finite element models that always yield a symmetric positive-definite (SPD) coefficient matrix and hence, robust direct or iterative solvers can be used. Also, the use of higher-order spectral/*hp* finite element technology results in a better conservation of various physical quantities (e.g., dilatation, volume, and mass). However, due to lack of velocity and pressure coupling, the least-squares formulation in its standard form is un-stable and results in a poor evolution (with spurious oscillations) of primary variables with time. To overcome this we introduce an iterative penalization scheme, on the similar lines of [34, 75], for the transient pressure-velocity-stress first-order system of Navier–Stokes equations. Finally, numerical solutions of several non-trivial benchmark problems will be discussed.

5.1 Introduction

Although, the finite element method has become the dominant method of choice in the numerical analysis of structures, it is yet to receive such a widespread acceptance in the field of computational fluid dynamics. In the realm of fluid mechanics, much of the success and breakthroughs, in the numerical discretization of the incompressible form of the Navier–Stokes equations have come in the context of low-order finite difference and finite volume technology. It is well known, however, that finite element procedures offer many advantages over finite difference and finite volume methods*. In particular, the finite element method can naturally deal with complex regions, complicated boundary conditions and possesses a rich mathematical foundation [90]. As a result, there has been a renewed interest in recent years in developing efficient finite element models for the incompressible Navier–Stokes equations.

The majority of finite element models for fluids are based on the weak-form Galerkin procedure. It is well-known, however, that application of this method can lead to a non-optimal setting for a given finite element discretization [14, 88]. For instance, application of the Galerkin method to the incompressible Navier–Stokes equations leads to a finite element implementation in terms of the velocity and pressure which must satisfy the restrictive discrete Ladyzhenskaya-Babuska-Brezzi (LBB) condition [17]; this effectively precludes the use of equal, lower-order, interpolation of the velocity and pressure variables in the numerical implementation. Even when the LBB condition is satisfied, the finite element model may still be plagued by spurious oscillations in convection dominated flows and also conservation of various physical quantities (like dilatation, volume, mass etc.) is poor. Stabilized Galerkin based finite element formulations such as the SUPG [41, 18], penalty [82] and Galerkin least-squares [42] have received considerable attention over the last few decades and have greatly improved the discrete setting for the finite element analysis of fluid flow problems. Unfortunately, the success of these methods is often intertwined with *ad-hoc* parameters that must be fine-tuned for a given flow problem. In addition, such formulations do not result in a symmetric positive-definite (SPD) coefficient matrix. As with the structures, for fluid flows, there is no reliable, general purpose

*In fact, the finite volume method can be viewed as the finite element model based on the subdomain method.

stabilization-free, low-order finite element technology.

5.2 Least-squares finite element formulation

5.2.1 The incompressible Navier–Stokes equations

Here, we consider viscous, isothermal, incompressible Navier–Stokes fluid flows. The problem may be stated in non-dimensional form as follows: find the velocity $\mathbf{u}(\mathbf{x},t)$ and pressure $p(\mathbf{x},t)$ such that

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in } \Omega \tag{5.1a}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}}\nabla\cdot\left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right] = \mathbf{f} \qquad \text{in } \Omega \tag{5.1b}$$

$$\mathbf{u} = \mathbf{u}^{\mathrm{P}} \qquad \text{on } \Gamma_u \tag{5.1c}$$

$$\hat{\mathbf{n}}\cdot\underline{\boldsymbol{\sigma}} = \mathbf{t}^{\mathrm{P}} \qquad \text{on } \Gamma_t \tag{5.1d}$$

where Re is the Reynolds number, $\mathbf{f}$ is the dimensionless resultant body force due to agents like gravity, magnetic effects etc., $\mathbf{u}^{\mathrm{P}}$ is the dimensionless prescribed velocity on the boundary Γ_u, $\mathbf{t}^{\mathrm{P}}$ is the dimensionless prescribed traction on the boundary Γ_t, $\hat{\mathbf{n}}$ is the outward unit normal to the boundary Γ_t and $\underline{\boldsymbol{\sigma}}$ is the total stress tensor (Cauchy stress). It must be noted that the parts of boundary with prescribed velocities and tractions satisfy $\Gamma = \Gamma_u \cup \Gamma_t$ and $\emptyset = \Gamma_u \cap \Gamma_t$. From the constitutive relation, the Cauchy stress can be represented in terms of primitive variables as

$$\underline{\boldsymbol{\sigma}} = -p\underline{\mathbf{I}} + \frac{1}{\mathrm{Re}}\left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right] \tag{5.2}$$

Although, the least-squares method can be viewed as a special case of the weighted-residual method, it has its own standing as a true variational method since it involves the minimization of a functional. Also, the weighted-residual methods may or may not have a corresponding functional whose first variation is equivalent to the governing equations. Variational methods (i.e. methods based on the existence of a functional whose extremum is equivalent to the governing equations) are considered to produce the "best" approximation to the exact solution of the equations being solved. Most solid/structural mechanics problems allow the construction of a quadratic functional (based on energy considerations) whose extremum would provide the basis for the construction of associated finite element models. Unfortunately, such a functional does not exist for the Navier–Stokes equations when expressed in terms of primitive variables. Consequently, most finite element models of the Navier–Stokes equations based on the weak-form Galerkin formulation does not guarantee minimization of the error in the approximation of the solution nor in the differential equation.

If traditional weak-form Galerkin formulation is used to construct the functional for the above Navier–Stokes equations in terms of its primitive variables of pressure p and velocity $\mathbf{u}$, then the minimization of the functional with the solution results in a saddle point problem i.e. it is minimum with respect to velocity vector and maximum with respect to scalar pressure at any local or global minima. Hence, the choice of approximation function spaces for velocity and pressure are constrained to satisfy this compatibility condition, which is popularly known as LBB condition. To circumvent the saddle point problem and to get close to the variational principles, many have implemented methods like Streamline Upwind Petro Galerkin (SUPG), Lagrange multiplier methods, Penalty methods, Galerkin Least-Squares methods and other stabilized methods. But these methods are *ad-hoc*, problem-dependent and require an arbitrary choice of parameters. Also, the coefficient matrices are non-positive-definite due to the absence of pressure variable in the continuity equation and for nonlinear equations, if the coefficient matrix is non-symmetric it increases the computational cost.

The least-squares method gives a more general, flexible and robust formulation procedure than the stabilized models based on weak-form Galerkin formulations. However, if the least-squares formulation is directly applied to the above second-order Navier–Stokes equations in terms of its primitive variables of pressure p and velocity $\mathbf{u}$, then C^1 continuous approximations functions have to be used, where as, for weak-form Galerkin formulation only C^0 continuous functions are required due to weakening by the integration-by-parts. Because of this C^1 continuity requirement the least-squares formulations lost their appeal in 1970's. Recently Surana and others have started applying least-squares method directly to second-order Navier–Stokes equations in terms of its primitive variables pressure p and velocity $\mathbf{u}$ using C^1 continuous approximations functions [106]. But generating C^1 continuous approximations functions in higher-dimension spaces for higher-order elements and un-structured meshes is complicated. Analogous to *hp*-finite element frame work they call this as *hpk*-frame work.

Bo-nan Jiang introduced auxiliary variables like the vorticity, stresses, dilation, velocity gradients to reduce the second-order Navier–Stokes equations in terms of its primitive variables of pressure p and velocity $\mathbf{u}$ to first-order system of equation, so that C^0 continuous functions can be used Refs. [47, 44, 48, 46]. However, Bo-nan Jiang used lower-order finite elements, which results in collocation least-squares finite element method, hence *reduce* or *under* integration is used to get accurate solutions. Auxiliary variables introduced will increase the degree-of-freedoms per node in the finite element model and so the computation cost. But the post-processing cost can be reduced if suitable auxiliary variables are used. It is important to note that although all least-squares finite element models based on first-order equations obtained from auxiliary variables like the vorticity, stresses, dilation, velocity gradients show spectral convergence, it is not possible *a-priori* to show that every first-order system is norm-equivalent.

5.2.2 The stress-based first-order system

To over come the C^1 continuity and to allow the use of practical C^0 basis functions in the numerical implementation, we introduce the symmetric stress tensor, $\underline{\mathbf{T}} = \left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right]$ as an auxiliary variable. Using this, the second-order Navier–Stokes problem statement can be recast as the equivalent first-order problem statement: find the pressure $p(\mathbf{x},t)$, velocity $\mathbf{u}(\mathbf{x},t)$ and stress $\underline{\mathbf{T}}(\mathbf{x},t)$ such that

$$\nabla\cdot\mathbf{u} = 0 \qquad \text{in } \Omega \qquad (5.3a)$$

$$\frac{\partial\mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla p - \frac{1}{\mathrm{Re}}\nabla\cdot\underline{\mathbf{T}} = \mathbf{f} \qquad \text{in } \Omega \qquad (5.3b)$$

$$\underline{\mathbf{T}} = \left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right] \qquad \text{in } \Omega \qquad (5.3c)$$

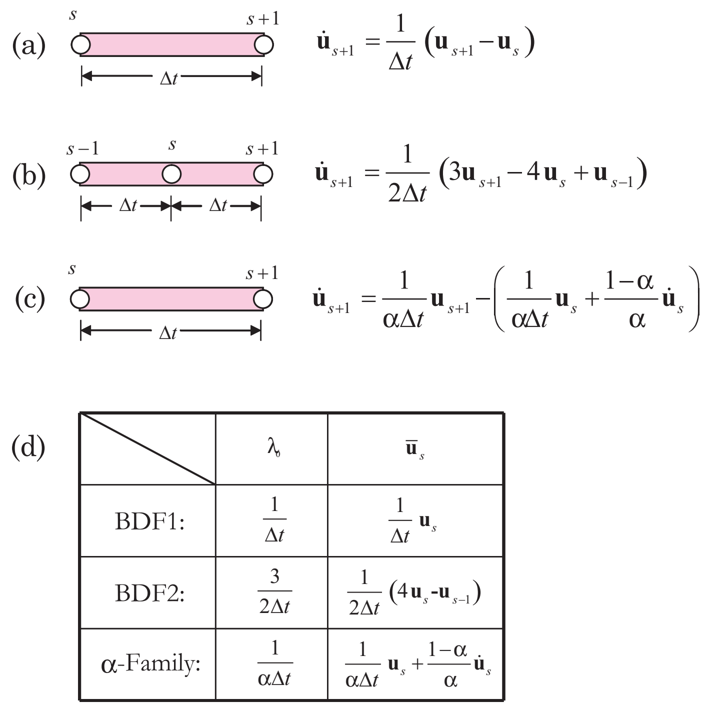
$$\mathbf{u} = \mathbf{u}^{\mathrm{P}} \qquad \text{on } \Gamma_u \qquad (5.3d)$$

$$\hat{\mathbf{n}}\cdot\underline{\mathbf{T}} = \underline{\mathbf{T}}^{\mathrm{P}} \qquad \text{on } \Gamma_{\mathrm{T}} \qquad (5.3e)$$

5.2.3 Time discretization and standard L_2-norm least-squares formulation

Adopting a space-time decoupled formulation, the above system of equations are first discretized in time and then in space to solve the transient flow simulation problems. For time discretization, we use backward difference (BDF1 and BDF2) and the α-family time approximation schemes given in Fig. 5.1.

Using these time discretization schemes, the time derivative of velocity field, at $t = t_{s+1}$, can be replaced as shown in Eq. (5.4), where $\bar{\mathbf{u}}_s$ is the history vector and



$$\dot{\mathbf{u}}_{s+1} = \frac{1}{\Delta t}\left(\mathbf{u}_{s+1} - \mathbf{u}_s\right)$$

$$\dot{\mathbf{u}}_{s+1} = \frac{1}{2\Delta t}\left(3\mathbf{u}_{s+1} - 4\,\mathbf{u}_s + \mathbf{u}_{s-1}\right)$$

$$\dot{\mathbf{u}}_{s+1} = \frac{1}{\alpha\Delta t}\mathbf{u}_{s+1} - \left(\frac{1}{\alpha\Delta t}\mathbf{u}_s + \frac{1-\alpha}{\alpha}\dot{\mathbf{u}}_s\right)$$

	λ_0	$\bar{\mathbf{u}}_s$
BDF1:	$\frac{1}{\Delta t}$	$\frac{1}{\Delta t}\mathbf{u}_s$
BDF2:	$\frac{3}{2\Delta t}$	$\frac{1}{2\Delta t}\left(4\,\mathbf{u}_s\text{-}\mathbf{u}_{s-1}\right)$
α-Family:	$\frac{1}{\alpha\Delta t}$	$\frac{1}{\alpha\Delta t}\mathbf{u}_s + \frac{1-\alpha}{\alpha}\dot{\mathbf{u}}_s$

Figure 5.1: Time discretization schemes of (a) BDF1 (b) BDF2 (c) α-Family and (d) Constant and history vector.

λ_0 is a constant, the specific forms are given in Fig. 5.1(d).

$$\begin{aligned} \dot{\mathbf{u}}_{s+1} &= \lambda_0\,\mathbf{u}_{s+1} - \bar{\mathbf{u}}_s \\ &= \tfrac{1}{\Delta t}\left(\lambda_0 \Delta t\,\mathbf{u}_{s+1} - \Delta t\,\bar{\mathbf{u}}_s\right) \end{aligned} \tag{5.4}$$

The standard least-squares functional associated with the above first-order stress-based Navier–Stokes system can be constructed by taking the sum of the squares of the L_2 norms of the residual equations. At time step $t = t_{s+1}$, the algebraic differential equation in time, allows us to define the associated least-squares functional as

$$\mathcal{J}\left(p,\mathbf{u},\underline{\mathbf{T}};\mathbf{f}\right)=\frac{1}{2}\Bigg(\left\|\nabla\cdot\mathbf{u}\right\|_0^2+\left\|\frac{1}{\Delta t}\left(\lambda_0\Delta t\mathbf{u}_{s+1}-\Delta t\bar{\mathbf{u}}_s\right)+\left(\mathbf{u}\cdot\nabla\right)\mathbf{u}+\nabla p-\frac{1}{\mathrm{Re}}\nabla\cdot\underline{\mathbf{T}}-\mathbf{f}\right\|_0^2$$
$$+\left\|\underline{\mathbf{T}}-\left[\left(\nabla\mathbf{u}\right)+\left(\nabla\mathbf{u}\right)^{\mathrm{T}}\right]\right\|_0^2+\underline{\left\|\hat{\mathbf{t}}-\hat{\mathbf{n}}\cdot\tilde{\underline{\boldsymbol{\sigma}}}\right\|_{0,\Gamma_{\text{outflow}}}^2}\Bigg) \tag{5.5}$$

Note in the above, the outflow boundary condition given by $\hat{\mathbf{t}}-\hat{\mathbf{n}}\cdot\tilde{\underline{\boldsymbol{\sigma}}}=0$, is applied in a weak sense using the least-squares functional (see the underlined term), where $\hat{\mathbf{t}}$ is the traction vector at the outflow section and $\tilde{\underline{\boldsymbol{\sigma}}}=-p\underline{\mathbf{I}}+\left(1/\mathrm{Re}\right)\nabla\mathbf{u}$ is the pseudo Cauchy stress tensor.

The least-squares minimization problem is to find variables $p(\mathbf{x},t),\mathbf{u}(\mathbf{x},t),\underline{\mathbf{T}}(\mathbf{x},t)\ni$ $\mathcal{J}\left(p,\mathbf{u},\underline{\mathbf{T}};\mathbf{f}\right)\leq\mathcal{J}\left(\tilde{p},\tilde{\mathbf{u}},\tilde{\underline{\mathbf{T}}};\mathbf{f}\right)\forall\left(\tilde{p},\tilde{\mathbf{u}},\tilde{\underline{\mathbf{T}}};\mathbf{f}\right)\in(\mathbf{x},t)$, i.e. seek $(p,\mathbf{u},\underline{\mathbf{T}})$ such that $\mathcal{J}\left(\mathrm{p},\mathbf{u},\underline{\mathbf{T}};\mathbf{f}\right)$ is minimized over $\mathbf{x}$, where $\mathbf{x}$ is

$$\mathbf{x}=\left\{\left(p,\mathbf{u},\,\underline{\mathbf{T}}\right)\in H^1\left(\Omega\right)\times H^1\left(\Omega\right)\times H^1\left(\Omega\right)\right\} \tag{5.6}$$

The variational problem (after linearizing by Newton's Method) corresponding to above least-squares functional can be written as

$$\mathcal{B}\left(\left(\tilde{p},\tilde{\mathbf{u}},\tilde{\underline{\mathbf{T}}}\right),\left(p,\mathbf{u},\underline{\mathbf{T}}\right)\right)=\mathcal{F}\left(\tilde{p},\tilde{\mathbf{u}},\tilde{\underline{\mathbf{T}}}\right)\forall\left(\tilde{p},\tilde{\mathbf{u}},\tilde{\underline{\mathbf{T}}}\right)\in\left(\mathbf{x},t\right) \tag{5.7}$$

where the bi-linear form is explicitly given as

$$
\begin{aligned}
\mathcal{B}\left(\left(\tilde{p},\tilde{\mathbf{u}},\underline{\tilde{\mathbf{T}}}\right),\left(p,\mathbf{u},\underline{\mathbf{T}}\right)\right) = \int_{\Omega} \Bigg\{ & (\nabla\cdot\tilde{\mathbf{u}})\cdot(\nabla\cdot\mathbf{u}) + \Delta t\Big(\lambda_0\tilde{\mathbf{u}}_{s+1} + (\mathbf{u}_0\cdot\nabla)\,\tilde{\mathbf{u}} \\
& + (\tilde{\mathbf{u}}\cdot\nabla)\,\mathbf{u}_0 + \nabla\tilde{p} - \frac{1}{\mathrm{Re}}\nabla\cdot\underline{\tilde{\mathbf{T}}}\Big)\cdot\Delta t\Big(\lambda_0\mathbf{u}_{s+1} \\
& + (\mathbf{u}_0\cdot\nabla)\,\mathbf{u} + (\mathbf{u}\cdot\nabla)\,\mathbf{u}_0 + \nabla p - \frac{1}{\mathrm{Re}}\nabla\cdot\underline{\mathbf{T}}\Big) \\
& + \Big(\underline{\tilde{\mathbf{T}}} - \Big[(\nabla\tilde{\mathbf{u}}) + (\nabla\tilde{\mathbf{u}})^{\mathrm{T}}\Big]\Big)\cdot\Big(\underline{\mathbf{T}} - \Big[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\Big]\Big)\Bigg\} d\Omega \\
& + \underline{\int_{\Gamma_{\mathrm{out}}} \Big(\tilde{p}\hat{\mathbf{n}} - \frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\tilde{u}\Big)\cdot\Big(p\hat{\mathbf{n}} - \frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\mathbf{u}\Big)ds}
\end{aligned}
\tag{5.8}
$$

From the above it clear that the bi-linear form is symmetric and positive definite (SPD) and the linear form is given as

$$
\begin{aligned}
\mathcal{F}\left(\tilde{p},\tilde{\mathbf{u}},\underline{\tilde{\mathbf{T}}}\right) = \int_{\Omega} \Bigg[& \Delta t\Big(\lambda_0\tilde{\mathbf{u}}_{s+1} + (\mathbf{u}_0\cdot\nabla)\,\tilde{\mathbf{u}} + (\tilde{\mathbf{u}}\cdot\nabla)\,\mathbf{u}_0 + \nabla\tilde{p} \\
& - \frac{1}{\mathrm{Re}}\nabla\cdot\underline{\tilde{\mathbf{T}}}\Big)\cdot\Delta t\Big(\bar{\mathbf{u}}_s + (\mathbf{u}_0\cdot\nabla)\,\mathbf{u}_0 + \mathbf{f}\Big)\Bigg] d\Omega \\
& + \underline{\int_{\Gamma_{\mathrm{out}}} \Big(\frac{1}{\mathrm{Re}}\hat{\mathbf{n}}\cdot\nabla\tilde{\mathbf{u}} - \tilde{p}\hat{\mathbf{n}}\Big)\cdot\hat{\mathbf{t}}ds}
\end{aligned}
\tag{5.9}
$$

In two-dimensions, the bi-linear and linear forms can be represented as

$$
\begin{bmatrix}
\mathrm{K}^{11} & \mathrm{K}^{12} & \mathrm{K}^{13} & \mathrm{K}^{14} & \mathrm{K}^{15} & \mathrm{K}^{16} \\
\mathrm{K}^{21} & \mathrm{K}^{22} & \mathrm{K}^{23} & \mathrm{K}^{24} & \mathrm{K}^{25} & \mathrm{K}^{26} \\
\mathrm{K}^{31} & \mathrm{K}^{32} & \mathrm{K}^{33} & \mathrm{K}^{34} & \mathrm{K}^{35} & \mathrm{K}^{36} \\
\mathrm{K}^{41} & \mathrm{K}^{42} & \mathrm{K}^{43} & \mathrm{K}^{44} & \mathrm{K}^{45} & \mathrm{K}^{46} \\
\mathrm{K}^{51} & \mathrm{K}^{52} & \mathrm{K}^{53} & \mathrm{K}^{54} & \mathrm{K}^{55} & \mathrm{K}^{56} \\
\mathrm{K}^{61} & \mathrm{K}^{62} & \mathrm{K}^{63} & \mathrm{K}^{64} & \mathrm{K}^{65} & \mathrm{K}^{66}
\end{bmatrix}
\begin{Bmatrix} p \\ u_x \\ u_y \\ T_{xx} \\ T_{xy} \\ T_{yy} \end{Bmatrix}
=
\begin{Bmatrix} \mathrm{F}^{1} \\ \mathrm{F}^{2} \\ \mathrm{F}^{3} \\ \mathrm{F}^{4} \\ \mathrm{F}^{5} \\ \mathrm{F}^{6} \end{Bmatrix}
\tag{5.10}
$$

5.2.4 Modified least-squares formulation with iterative penalization scheme

In weak-form Galerkin methods, stabilized methods and penalty methods the pressure (or its test function) plays the role of the Lagrange multiplier making continuity equation $\nabla \cdot \mathrm{u} = 0$ at all times and space. This kind of coupling eludes the above standard least-squares formulation. If the traditional penalization method is used to enforce the divergence-free condition (continuity equation), it results in ill-conditioning due to the requirement of high penalty parameters. It in-turn leads to numerical locking due to difference in contributions from viscous and penalty terms. Due to this, the evolution of primary variables is poor with time and is often associated with spurious oscillations. However, the coupling can be introduced into the least-squares formulation using the following iterative penalization strategy from [34, 75].

$$p^{k+1} = p^k - \gamma \left[\nabla \cdot \mathbf{u}\right] \Rightarrow \frac{\Delta p}{\gamma} = \nabla \cdot \mathbf{u} \tag{5.11a}$$

$$\Rightarrow \int\limits_{\Omega^e} \frac{\Delta p}{\gamma} d\Omega^e = \oint\limits_{\Gamma^e} \hat{\mathbf{n}} \cdot \mathbf{u}\, d\Gamma^e \tag{5.11b}$$

$$p^{k+1} = p^k - \gamma \left[\frac{1}{2}\mathrm{tr}(\underline{\mathbf{T}})\right] \Rightarrow \frac{\Delta p}{\gamma} = \frac{1}{2}\mathrm{tr}(\underline{\mathbf{T}}) \tag{5.11c}$$

here $k+1$ is the current iteration number and γ is the penalty parameter. The advantage is that it requires small magnitudes $(5 - 40)$ of penalty parameter. Using Eq. (5.11c) in momentum equation (5.3b), the pressure variable and the continuity equation can be eliminated from the system of equations. The modified least-squares functional associated with the new set of equations at current time $t = t_{s+1}$ and current iteration $k+1$ becomes:

$$\begin{aligned}\mathcal{J}\left(\mathbf{u},\underline{\mathbf{T}};\mathbf{f}\right) = \frac{1}{2}\Bigg(& \left\| \frac{1}{\Delta t}\left(\lambda_0 \Delta t \mathbf{u}_{s+1} - \Delta t \bar{\mathbf{u}}_s\right) + \left(\mathbf{u}\cdot\nabla\right)\mathbf{u} - \gamma\,\nabla\left[\tfrac{1}{2}\mathrm{tr}(\underline{\mathbf{T}})\right] - \frac{1}{\mathrm{Re}}\nabla\cdot\underline{\mathbf{T}} - \mathbf{f} + \nabla p^k \right\|_0^2 \\ & + \left\| \underline{\mathbf{T}} - \left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right] \right\|_0^2 + \underline{\left\| \hat{\mathbf{t}} - \hat{\mathbf{n}}\cdot\underline{\tilde{\boldsymbol{\sigma}}} \right\|_{0,\Gamma_{\text{outflow}}}^2} \Bigg)\end{aligned} \tag{5.12}$$

From the above modified least-squares functional the bi-linear and linear forms can be obtained as discussed above. Due to small penalty parameters, the contributions of viscous and penalty terms are comparable and it avoids ill-conditioning. This improves conservation of physical quantities like dilatation, mass, volume etc. and the stability of the numerical scheme. Also, due to improved coupling, the time evolution of variables is smooth and without any spurious oscillations. Once the solution is obtained, the pressure p can be post-computed using the above iterative relation. The above bi-linear and linear forms can be explicitly written as

$$\begin{bmatrix} \mathrm{K}^{11} & \mathrm{K}^{12} & \mathrm{K}^{13} & \mathrm{K}^{14} & \mathrm{K}^{15} \\ \mathrm{K}^{21} & \mathrm{K}^{22} & \mathrm{K}^{23} & \mathrm{K}^{24} & \mathrm{K}^{25} \\ \mathrm{K}^{31} & \mathrm{K}^{32} & \mathrm{K}^{33} & \mathrm{K}^{34} & \mathrm{K}^{35} \\ \mathrm{K}^{41} & \mathrm{K}^{42} & \mathrm{K}^{43} & \mathrm{K}^{44} & \mathrm{K}^{45} \\ \mathrm{K}^{51} & \mathrm{K}^{52} & \mathrm{K}^{53} & \mathrm{K}^{54} & \mathrm{K}^{55} \end{bmatrix} \begin{Bmatrix} u_x \\ u_y \\ T_{xx} \\ T_{xy} \\ T_{yy} \end{Bmatrix} = \begin{Bmatrix} \mathrm{F}^1 \\ \mathrm{F}^2 \\ \mathrm{F}^3 \\ \mathrm{F}^4 \\ \mathrm{F}^5 \end{Bmatrix} \tag{5.13}$$

The pressure p comes into the outflow boundary term from the pseudo Cauchy stress equation $\underline{\tilde{\boldsymbol{\sigma}}} = -p\underline{\mathbf{I}} + (1/\mathrm{Re})\,\nabla\mathbf{u}$. It can be eliminated using Eq. (5.11a) or Eq. (5.11c) as

$$\begin{aligned}
&\underline{\tilde{\boldsymbol{\sigma}}} = -p\underline{\mathbf{I}} + (1/\mathrm{Re})\,\nabla\mathbf{u}\\
&\Rightarrow \hat{\mathbf{t}} = \hat{\mathbf{n}}\cdot\underline{\tilde{\boldsymbol{\sigma}}} = \hat{\mathbf{n}}\cdot\left(-p\underline{\mathbf{I}} + \tfrac{1}{\mathrm{Re}}\nabla\mathbf{u}\right) \,\text{but}\, p^{k+1} = p^k - \gamma\,[\nabla\cdot\mathbf{u}] = p^k - \gamma\left[\tfrac{1}{2}\mathrm{tr}(\underline{\mathbf{T}})\right]\\
&\Rightarrow \hat{\mathbf{t}} = \hat{\mathbf{n}}\cdot\left(-p^k\underline{\mathbf{I}} + \gamma\,[\nabla\cdot\mathbf{u}]\,\underline{\mathbf{I}} + \tfrac{1}{\mathrm{Re}}\nabla\mathbf{u}\right) = \hat{\mathbf{n}}\cdot\left(-p^k\underline{\mathbf{I}} + \gamma\left[\tfrac{1}{2}\mathrm{tr}(\underline{\mathbf{T}})\right]\underline{\mathbf{I}} + \tfrac{1}{\mathrm{Re}}\nabla\mathbf{u}\right)
\end{aligned} \tag{5.14}$$

The contributions from the outflow terms to the stiffness matrix components is evaluated using the detailed procedure discussed in Section 2.

There are many ways in which the iterative penalization can be introduced in to the least-squares finite element formulation. Instead of using Eq. (5.11c) to eliminate pressure, as we did above, it can be directly introduced into the least-squares functional in a global discrete sense. The Eq. (5.11a) can also be directly introduced into the least-squares functional in a global discrete sense or it can be introduced by eliminating the pressure variable p from the system of equations. The Eq. (5.11b), which is obtained by taking integration over an an e'th element and applying divergence theorem, can also be introduced into the least-squares functional in a local integral sense as

$$\begin{aligned}
\mathcal{J}\left(p,\mathbf{u},\underline{\mathbf{T}};\mathbf{f}\right) = \tfrac{1}{2}\Bigg(&\left\|(\mathbf{u}\cdot\nabla)\,\mathbf{u} + \nabla p - \tfrac{1}{\mathrm{Re}}\nabla\cdot\underline{\mathbf{T}} - \mathbf{f}\right\|_0^2 + \left\|\underline{\mathbf{T}} - \left[(\nabla\mathbf{u}) + (\nabla\mathbf{u})^{\mathrm{T}}\right]\right\|_0^2\\
&\|\nabla\cdot\mathbf{u}\|_0^2 + w\left\|\oint_{\Gamma^e}\hat{\mathbf{n}}\cdot\mathbf{u}d\Gamma^e - \int_{\Omega^e}\tfrac{\Delta p}{\gamma}d\Omega^e\right\|_0^2(\mathbf{or})w\,\left\|\tfrac{\Delta p}{\gamma} - \nabla\cdot\mathbf{u}\right\|_0^2\Bigg)
\end{aligned} \tag{5.15}$$

where w is the suitable weighing parameter typically used for non-dimensionalizing or normalizing the above. No matter how the iterative penalization is introduced, it always all requires small magnitudes of the penalty parameter. With small penalty parameters the contributions of viscous and penalty terms are comparable and it avoids numerical locking. This leads to better mass conservation and stability (i.e. time evolution of variables) for non-stationary flows. There is no numerical evidence to suggest that one form works better than the other. Since, we are implementing the stress-based least-squares finite element formulation in this work, we use Eq. (5.11c) in the global discrete sense and also eliminate the pressure p from the system of equations.

5.3 Numerical results

In this section, the above stress-based least-squares finite element formulation is tested with a number of non-trivial benchmark problems for both stationary and transient flows. First, the spectral convergence of the higher-order spectral/*hp* least-squares formulation is verified for steady-state flow using lid-driven cavity problem. Next, the results are presented for flow over a backward-facing step, and flow past a circular cylinder at low Reynolds number with outflow boundary conditions. Non-linear convergence for a given numerical simulation is declared once the relative Euclidean norm of the solution residuals, $\|\Delta^{k+1} - \Delta^{k}\|/\|\Delta^{k+1}\|$, is less than 10^{-4}. All reported numerical results have been obtained using a penalty parameter of $\gamma = 10$.

5.3.1 Steady-state simulations

5.3.1.1 Lid-driven cavity flow problem

The “lid-driven” cavity is a standard test problem in the computational fluid dynamics. The problem is characterized by a unit square cavity of $\Omega = [0,1] \times [0,1]$ (see Fig. 5.2) in which the driving force for the flow is the shear created by the lid. The fluid contained inside a square cavity is set into motion by the upper wall which is sliding at constant velocity from left to right. The Dirichlet boundary conditions of $u_x = u_y = 0$ are prescribed on the left and right side-walls of the square cavity and $u_y = 0$ on the lid surface. A regularized hyperbolic tangent u_x velocity distribution is prescribed on the lid as given below:

$$u_x = u_x^{\mathrm{p}}(x) = \begin{cases} \tanh(50x) & 0 \leqslant x \leqslant 0.5 \\ -\tanh(50x - 50) & 0.5 < x \leqslant 1.0 \end{cases} \tag{5.16}$$

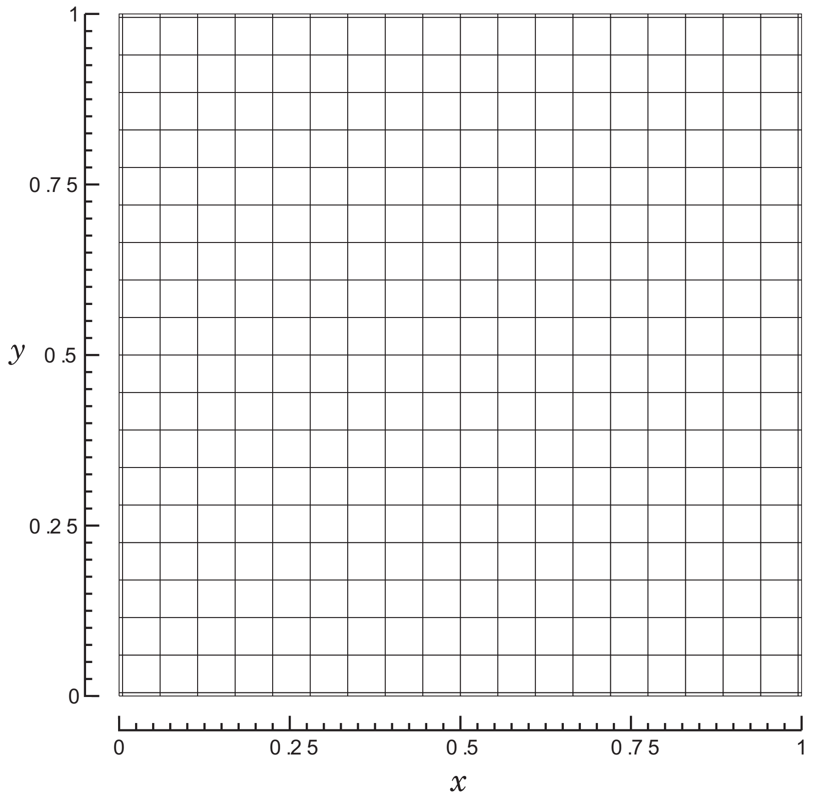


Figure 5.2: Lid-driven cavity element mesh.

The plot of the above distribution is shown in Fig. 5.3. As seen in the figure, it allows for a smooth transition from the no-slip boundary condition on the side-walls to the lid velocity on the top. The high-order finite elements are sensitive to such singularities at the corners of the lid-driven surface and can lead to an ill-posed problem if it is not taken care of.

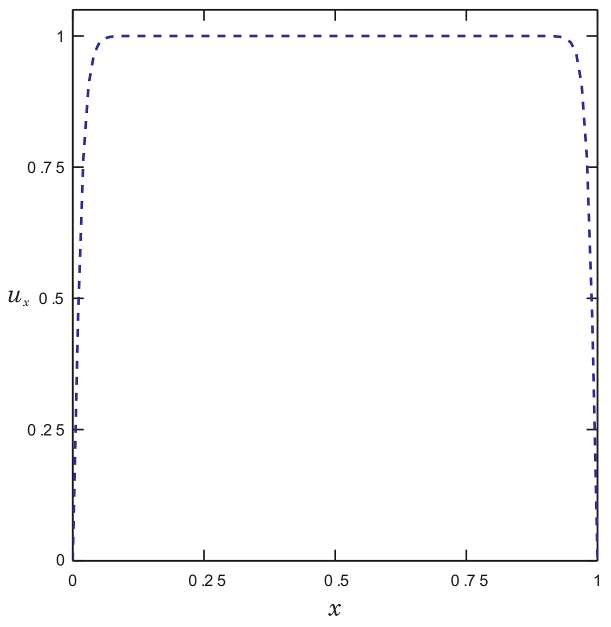


Figure 5.3: Regularized hyperbolic tangent u_x velocity distribution.

The finite element mesh in Fig. 5.2, has 20×20 non-uniform elements with adequately graded elements towards edges of the square cavity, so as to capture the flow circulations and transitions at the corners of the cavity. The Reynolds number is taken as 5×10^3, to achieve this, a load-stepping scheme is used, where, we start with Re $= 200$ and march towards Re $= 5 \times 10^3$ in a total of 25 equal load steps. At each load-step, the converged solution from the previous step is used as the initial guess. The penalty parameter is fixed as 10 for all the simulations. The problem is solved using spectral/hp elements of orders $p = 5, 7, 9$. All the problems typically converged in 2 or 3 non-linear iterations at all load steps.

The interplay of viscous and pressure forces makes the fluid to turn in the square cavity. The magnitude of these forces depends on the Reynolds number and when they outbalance one another it leads to a hierarchy of eddies, the large clockwise-rotating primary (at the geometric center of the cavity), and several small eddies such as the counter-clockwise rotating secondary eddies, the clockwise rotating tertiary eddies, at the three relevant corners of the square cavity: bottom left, bottom right, and top left. These can be seen in Fig. 5.4(a), where the streamlines are shwon along with pressure contour plots. The horizontal and vertical velocity contours are shown and in Figs. 5.4(b) and 5.4(c). These results match qualitatively very well with the published results of Jiang et al. [49]. To measure the spectral convergence of the higher-order least-squares formulation, we plot the total error in solution ε_{all}, the total least-squares functional $\mathcal{J}_{all}$, and the least-squares functional of the momentum equation $\mathcal{J}_{mom}$ in Fig. 5.4(d) for polynomial orders $p = 5, 7, 9$. Finally, to qualitatively measure the performance of the present formulation, the u_x velocity profiles along $x = 0.5$ are shown in Fig. 5.5(a) and the u_y velocity profiles along $y = 0.5$ are shown in Fig. 5.5(b) for Re = 200, 1000, 2000, 3200 and 5000 and compared with those of Jiang et al. [49]. These results perfectly match with the published results from the literature.

5.3.1.2 Flow over a backward facing step

As a second example, we consider a more rigorous example of steady-state fluid flow over a two-dimensional backward facing step at Reynolds number of 800. We utilize the simplified (truncated) step configuration proposed in the benchmark solution of Gartling [32]. The geometry of the domain, mesh and the boundary conditions are shown in Fig. 5.6(a). The computational domain for the problem is taken as

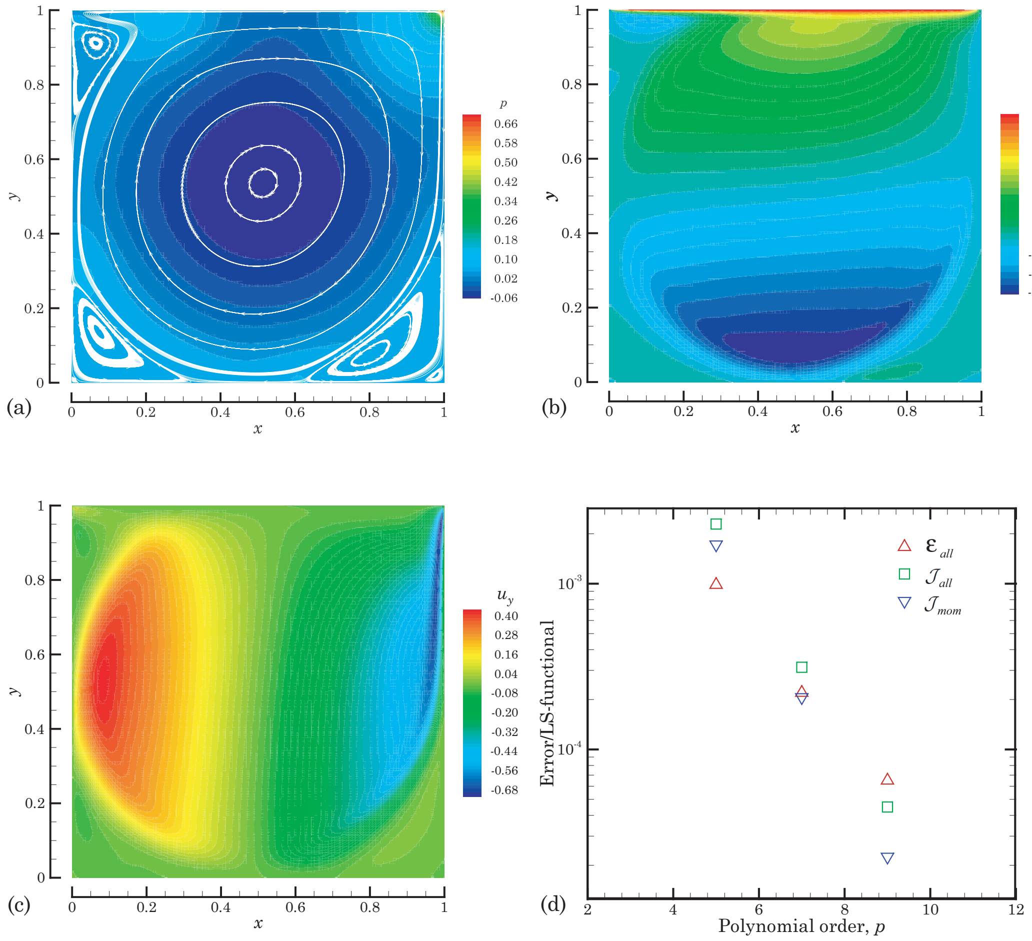


Figure 5.4: (a) Pressure contours and streamlines (b) Horizontal velocity contours (c) Vertical velocity contours and (d) Spectral convergence.

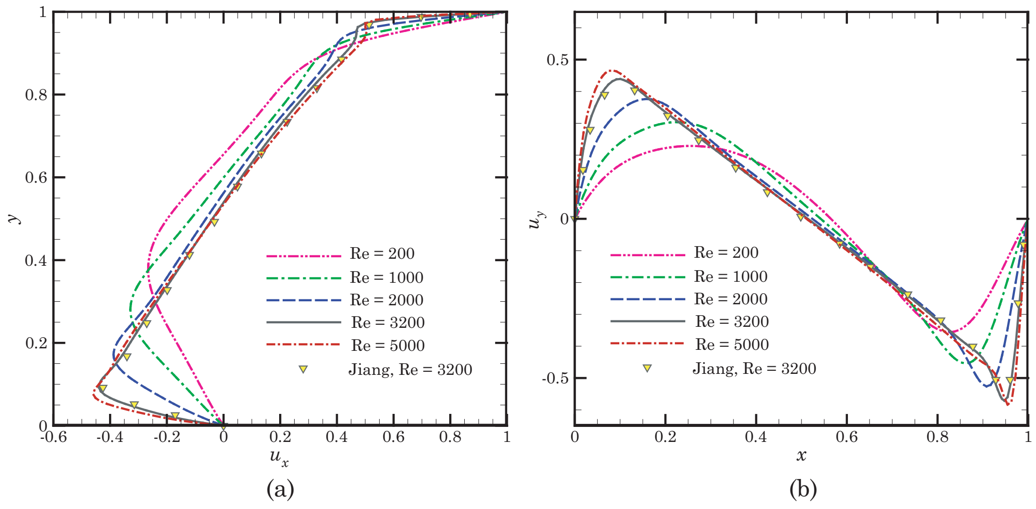


Figure 5.5: (a) u_x-velocity profiles along $x = 0.5$ and (b) u_y-velocity profiles along $y = 0.5$.

$\Omega = [0, 30] \times [-0.5, 0.5]$ and is discretized into a set of 40 rectangular finite elements, with 20 elements along the channel length and 2 along the channel height. The dimensions of the domain are selected so that the flow is completely evolved, stable and the end effects (due to termination of the problem to a geometrically finite computational domain) do not corrupt the integrity of the numerical solution. As can be seen in Fig. 5.6(a), the majority of the elements are positioned within 15 units of the channel inlet. This ensures the accurate resolution of primary and secondary circulation zones downstream of the step.

The velocity vector at the inlet is assumed to be horizontal as given by the parabolic expression $u_x = u_x^{\mathrm{p}}(y) = 24y(0.5-y)$ on $0.0 \leqslant y \leqslant 0.5$ and $u_x = u_x^{\mathrm{p}}(y) = 0.0$ on $-0.5 \leqslant y \leqslant 0.0$. The components of the velocity are taken to be zero along all solid surfaces (except on outflow) in accordance with the no-slip condition. The outflow boundary condition is enforced in a weak sense, by including the expression $\hat{\mathbf{t}} - \hat{\mathbf{n}} \cdot \underline{\tilde{\boldsymbol{\sigma}}} = 0$ in the definition of the least-squares functional, where pseudo traction vector on the outflow boundary is taken to be $\hat{\mathbf{t}} = 0$.

As with the previous example, we start with Re = 50 and march towards Re = 800 with an increment of 50 in a total of 16 load steps. At each load-step, the converged solution from the previous step is used as the initial guess. The penalty parameter is fixed as 10 for all the simulations. The problem is solved using spectral/hp elements of orders $p = 6, 8, 10, 12$. In Figs. 5.6(b), 5.6(c), 5.6(d) and 5.6(e), we plot the pressure contours, horizontal velocity contours, vertical velocity contours and the streamtraces over the entire domain. These plots qualitatively match with the previous results from the literature. Also, it can be seen that the flow is characterized by a primary separation and a recirculation zone which is directly behind the step and on the bottom-wall of the channel extending up to 6.1 units. A secondary flow separation and recirculation zone is also present on the top-wall of the channel that develops around 4.9 units downstream of the step and extends to approximately $x = 10.5$. These primary and secondary zone values match exactly with the numerical and experimental results from the literature. To do a qualitative analysis, in Figs. 5.7(a) and 5.7(b), we compare the components of the velocity vector along $x = 7$ and $x = 15$ with the results reported by Gartling [32], where a weak form Galerkin finite element model was employed. Also, in Fig. 5.7(c), we plot the pressure variation along the bottom-wall of the domain and compare with Pontaza [80]. The converged results are in excellent agreement with the published data.

To measure the conservation of various physical quantities, we make use of the incompressibility condition. The constraint that the density within a moving volume of fluid remains constant, the mass continuity equation simplifies to:

$$\nabla \cdot \mathbf{u} = 0 \tag{5.17}$$

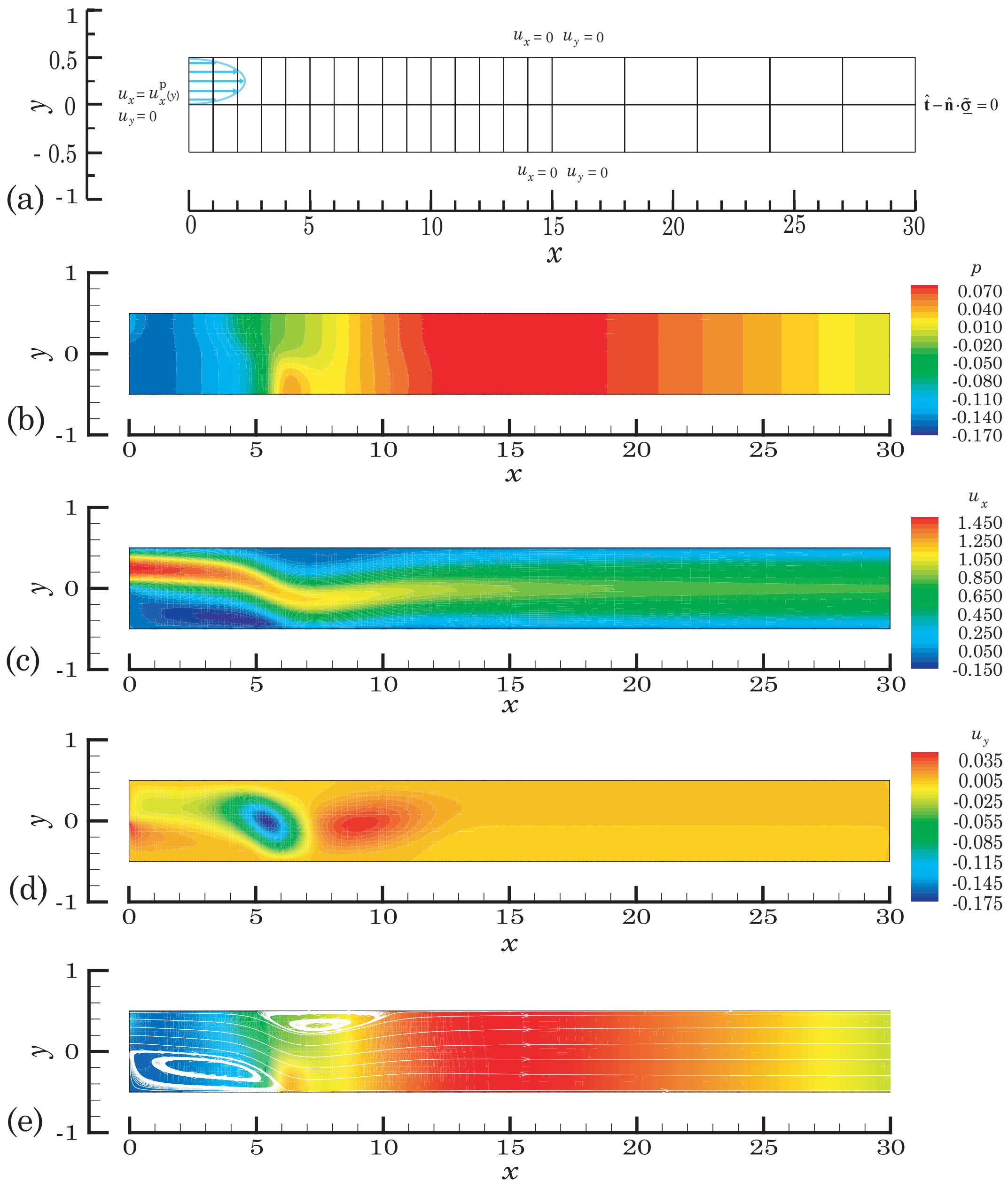


Figure 5.6: (a) Mesh and boundary conditions (b) Pressure contours (c) Horizontal velocity contours (d) Vertical velocity contours and (d) Streamtraces profile.

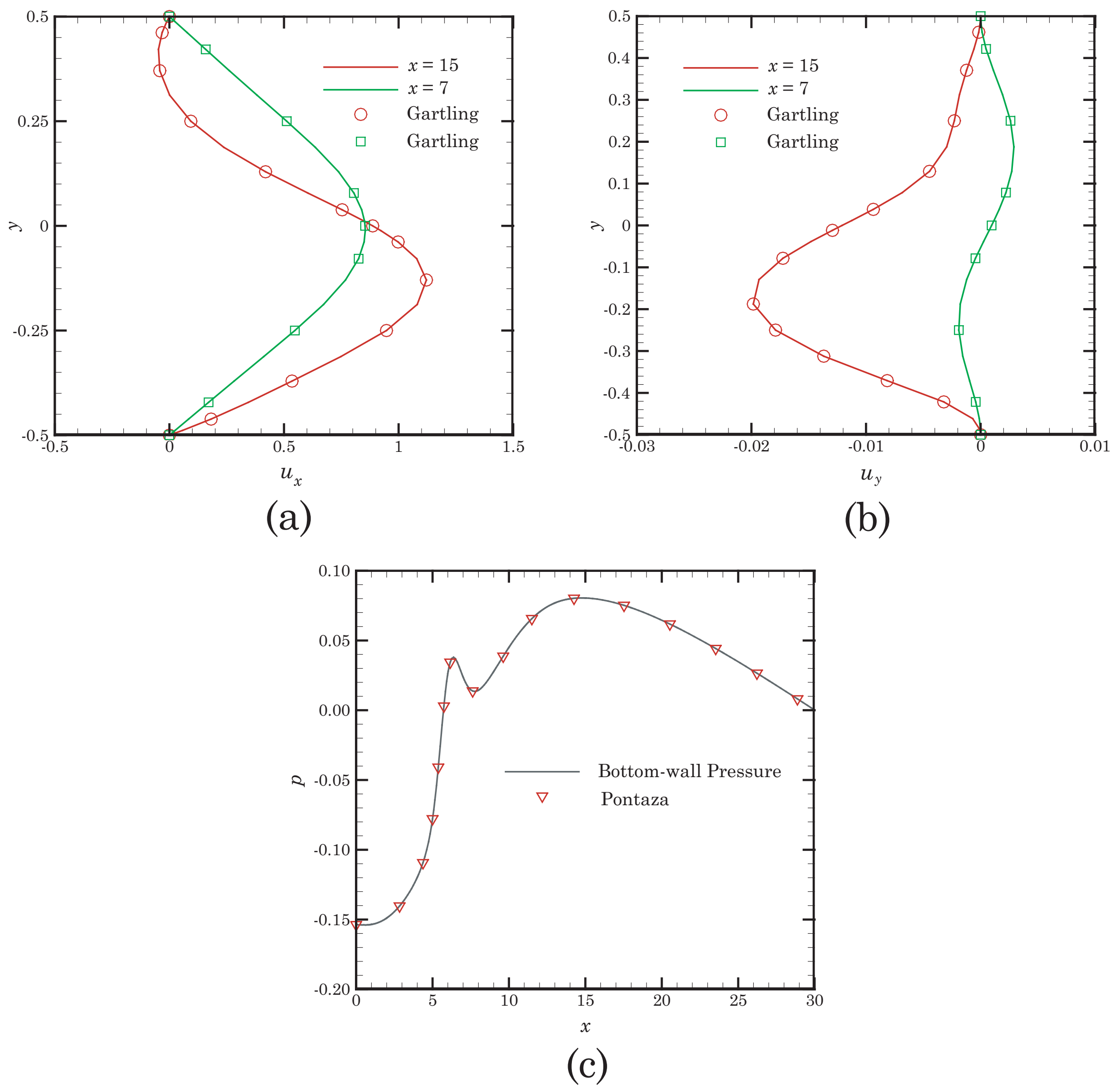


Figure 5.7: (a) Horizontal velocity at sections $x = 7$ and $x = 15$ (b) Vertical velocity at sections $x = 7$ and $x = 15$ (c) Bottom-wall pressure variation.

which means that the divergence of velocity field vanishes everywhere in the domain. Physically, this is equivalent to saying that the local volume dilation rate is zero. To see how well it is satisfied in each element of the domain, we numerically post-compute the normalized local volume dilation rate ($\mathcal{D}^e$) over the closed surface of each element

$$\mathcal{D}^e = \frac{1}{\mu^e} \oint\limits_{\Gamma^e} \hat{\mathbf{n}} \cdot \mathbf{u} \, \partial\Gamma^e \tag{5.18}$$

Note, the above equation is obtained by using divergence theorem to Eq. (5.18) over each element and normalizing with the factor μ^e, which in two-dimensions is the element area and in three-dimensions is the element volume. In Fig. 5.8, we plot the normalized local volume dilation rate for each element in the mesh for $p = 6, 10, 12$. The flow over a backward facing step is a difficult problem to simulate and with the present mesh discretization, it usually requires high polynomial orders ($p \geq 10$) of interpolation to get accurate solution. However, from Fig. 5.8(a), it is clear that even at ($p = 6$), the present iterative penalization scheme results in a better conservation of local volume dilation rate over the entire domain (i.e. better conservation). As the polynomial order is increased, it further strictly enforces the continuity equation as can be noted from Figs. 5.8(b) and 5.8(c). As expected, for the elements in the primary separation and recirculation zones and the secondary flow separation and recirculation zones, the conservation of local volume dilatation rate is relatively poor. By plotting the local element volume dilatation rate, the elements in the mesh can be flagged for further h-refinement or p-refinement.

5.3.1.3 Low Reynolds flow past a cylinder

Here we consider a steady two-dimensional flow of an incompressible fluid past a circular cylinder. The cylinder is of unit diameter and is at the center of the finite domain $\Omega = [-15.5, +25.5] \times [-20.5, +20.5]$ as shown in Fig. 5.9. The mesh has 501

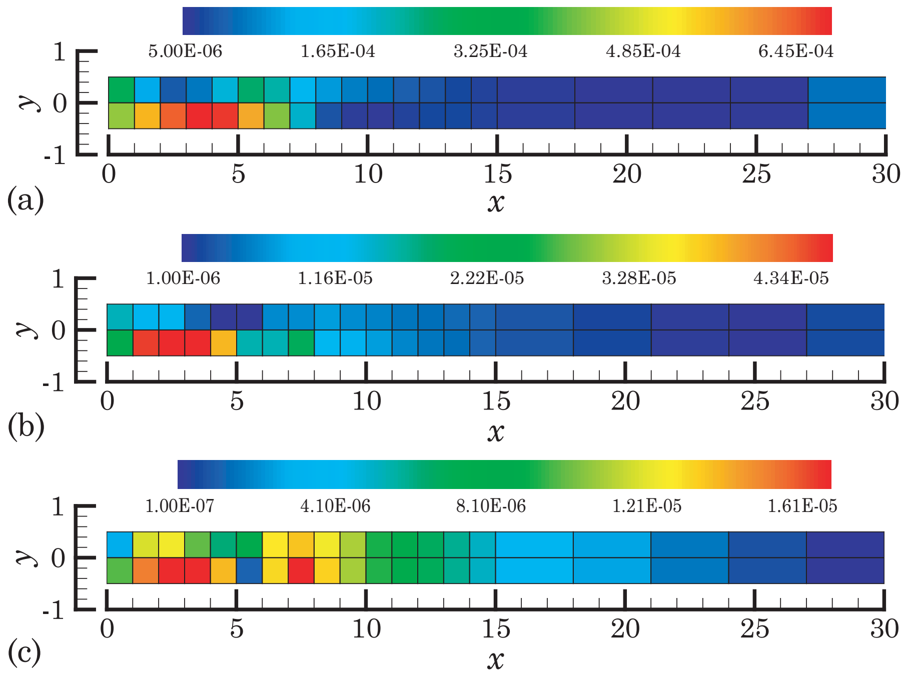


Figure 5.8: Local volume dilatation rates for (a) $p = 6$ (b) $p = 10$ and (c) $p = 12$.

quadrilateral finite elements, with body-fitting mesh around the cylinder. The value of Reynolds number and the placement of the computational boundaries in relation to the cylinder are critical as the flow pattern depends on them. At low Reynolds number ($5 < Re < 46.1$), the flow of an incompressible, newtonian fluid past a circular cylinder is stationary and its pattern is characterized by a pair of symmetric vortices on the downstream of the cylinder. The size of these standing vortex layers is proportional to the Reynolds number. As the Reynolds number reaches the critical value ($Re >= 46.1$), the standing vortex layers become unstable and flow can no longer be treated as two-dimensional flow. A Reynolds number of Re $= 40$ is used for all the cases in this work.

For this mesh, the horizontal velocity is specified as $u_x = 1.0$ at the inflow (left) and $u_x = u_\infty$ top and bottom boundaries, where u_∞ is the free-stream velocity and

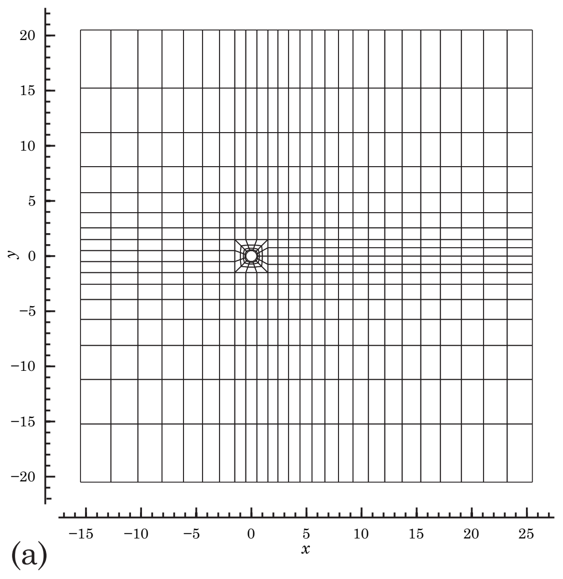
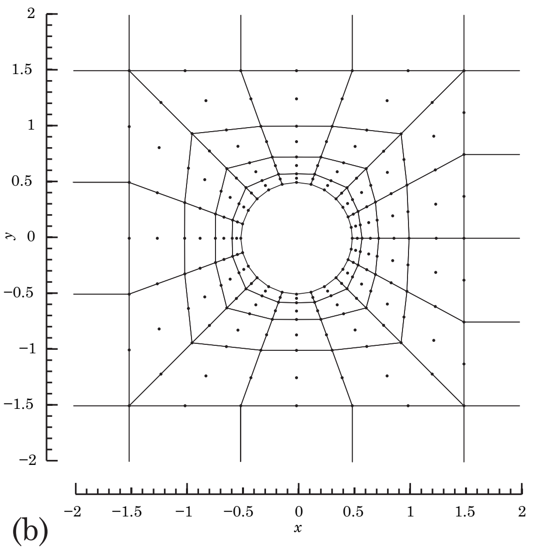
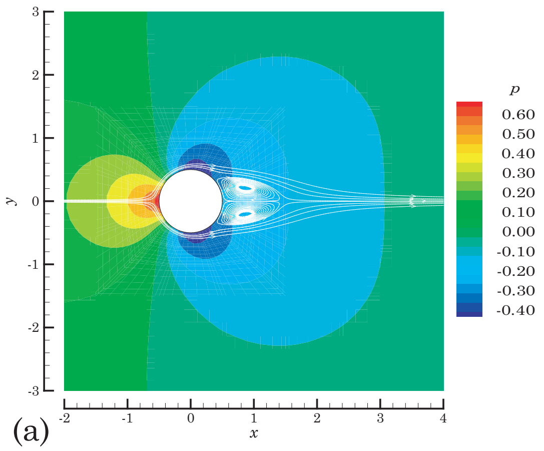
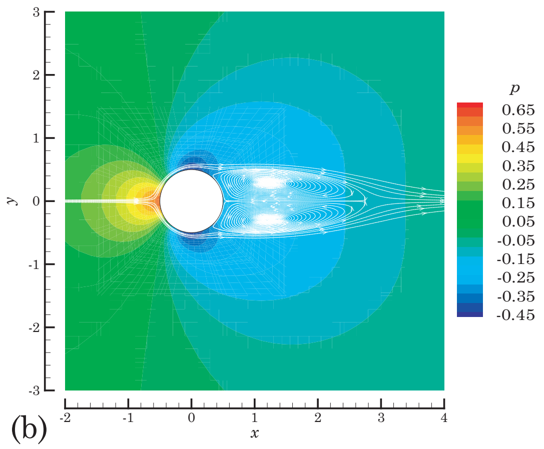
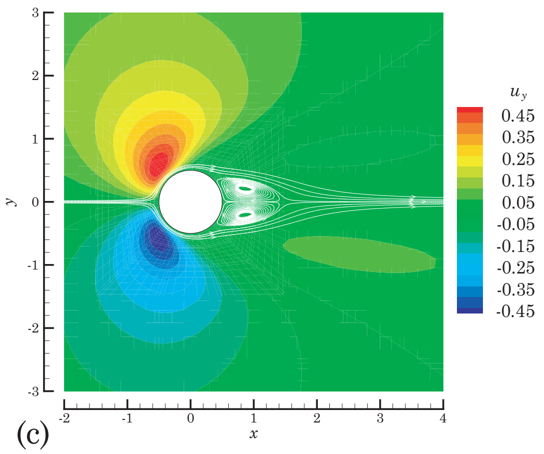



Figure 5.9: (a) Finite element mesh and (b) Close-up mesh with nodes for $p = 2$.

is taken as unity. Since the top and bottom surfaces are far from the cylinder, such boundary conditions do not influence the flow and hence do not affect the numerical solution. The vertical velocity is specified as $u_y = 0.0$ on all these three boundaries. A no-slip boundary condition of $u_x = u_y = 0.0$ is imposed on the surface of the cylinder. As in the previous problem the outflow boundary condition is enforced in a weak sense in the least-squares functional. The problem is solved with different polynomial orders of $p = 3, 5, 7$ each with 4659, 12775 and 24899 nodes respectively. In Fig. 5.10, we show the pressure and vertical velocity contour plots at Re = 20 and Re = 40. The streamlines are also shown highlighting the size of the circulation regions. It is clear that the length of the streamtraces is proportional to the Re. For Re = 40, our numerical calculations predict the wake region to extend 4.50 cylinder radii downstream of the cylinder. This is in excellent agreement with the numerical results reported by Kawaguti and Jain [52].

(a) 

(b) 

(c) 

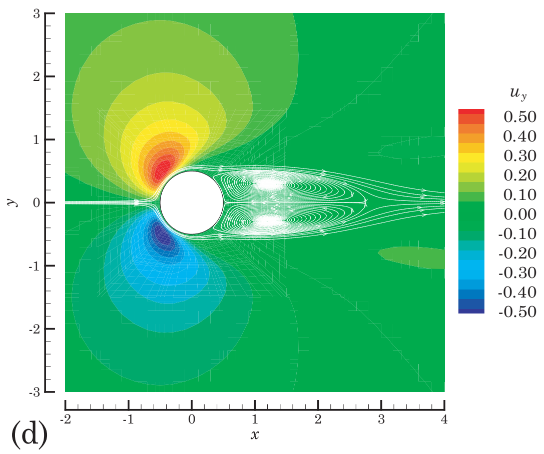
(d) 

Figure 5.10: (a) Pressure contours and streamtraces at Re = 20 (b) Pressure contours and streamtraces at Re = 40 (c) Vertical velocity contours at Re = 20 and (d) Vertical velocity contours at Re = 40.

In Fig. 5.11, we plot the post-computed values of normalized local volume dilation rate ($\mathcal{D}^e$) over the closed surface of elements around the cylinder for $p = 3, 5, 7$. As expected, for elements around the cylinder (especially on the crown and upstream region) the conservation of local volume dilatation rate is relatively poor. However, the improvement is particularly noticeable for these elements with p-refinement.

5.3.2 Non-stationary simulations

5.3.2.1 Transient lid-driven cavity flow

In this section, the transient lid-driven cavity flow problem, characterized by a unit square cavity of $\Omega = [0, 1] \times [0, 1]$, is analyzed. A 14×14 non-uniform graded mesh similar to Fig. 5.2 is used with the corner elements of 0.01×0.01 dimensions. The polynomial order $p = 7$ is used with total 196 elements and 9801 nodes. The pure-velocity Dirichlet no-slip boundary conditions, $u_x = u_y = 0$, are prescribed on the left and right side-walls of the square cavity, $u_y = 0$ on the lid surface and a regularized hyperbolic tangent u_x velocity distribution is prescribed on the lid as given below:

$$u_x(x) = u_x^{\mathrm{p}}(x) = \begin{cases} \tanh(50x) & 0 \leqslant x \leqslant 0.5 \\ -\tanh(50x - 50) & 0.5 < x \leqslant 1.0 \end{cases} \tag{5.19}$$

The horizontal velocity u_x of the lid driven surface also varies in time according to a hyperbolic tangent distribution as given by $u_x(x, t) = u_x^{\mathrm{p}}(x)\mathrm{tanh(t)}$. This problem has been solved for the penalty parameter of 10 and Reynolds number of 5000. Initial velocity conditions are taken to be zero everywhere. A uniform time step size of $\triangle t = 0.2$ with a total of 600 time steps are employed for the finite element simulation. Since the BDF2 time integration scheme is non-self-starting, we employ the BDF1 formula for the first 4 time steps and then use BDF2 for the rest.

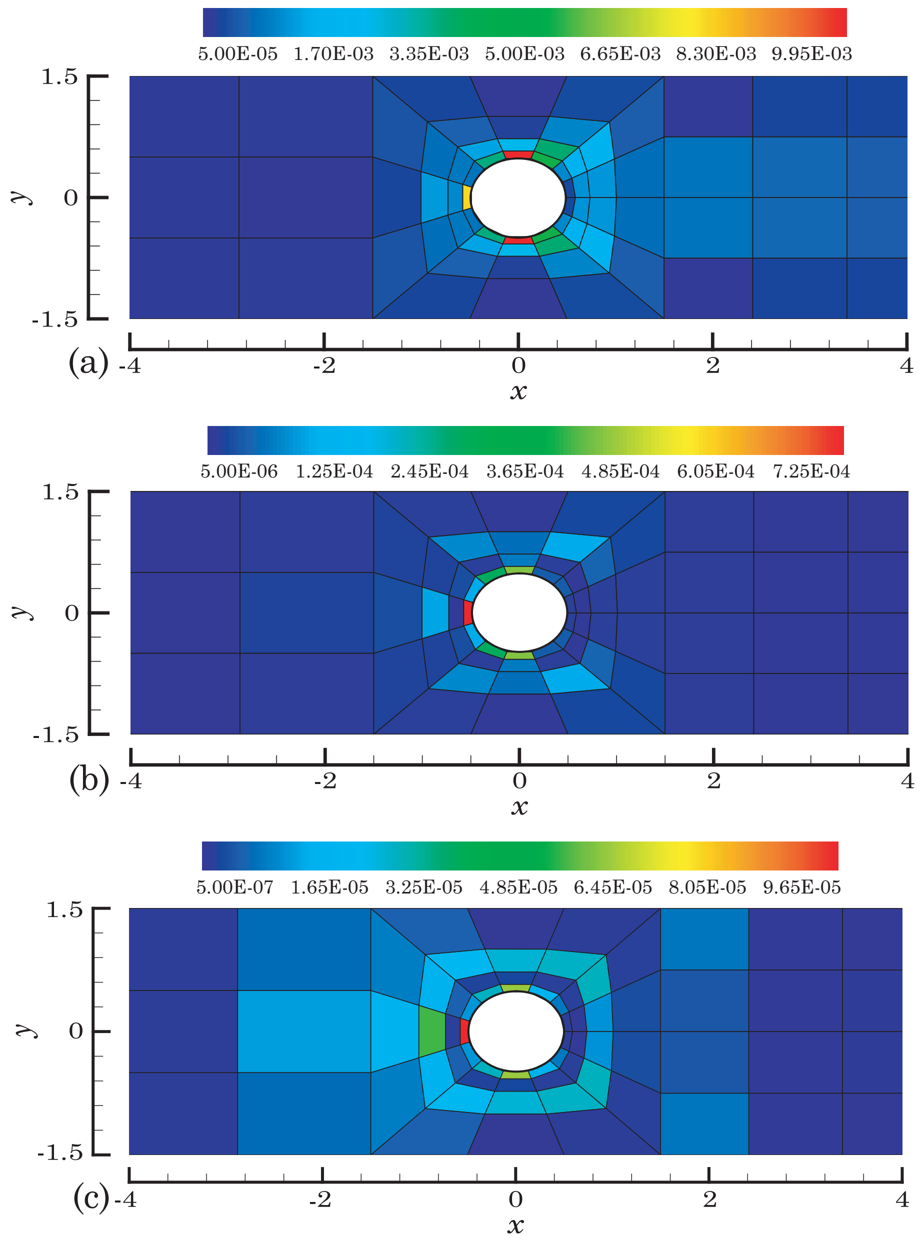


Figure 5.11: Local volume dilatation rates for (a) $p = 3$ (b) $p = 5$ and (c) $p = 7$.

The interplay of viscous and pressure forces makes the fluid to turn in the square cavity. The magnitude of these forces depends on the Reynolds number and when they outbalance each other it leads to a hierarchy of eddies, the large clockwise-rotating primary (at the geometric center of the cavity), and several small eddies such as the counter-clockwise rotating secondary eddies, the clockwise rotating tertiary eddies, at the three relevant corners of the square cavity: bottom left, bottom right, and top left. In Fig. 5.12, we plot the pressure contours in the square cavity at different times. In Fig. 5.13, we plot the time history of streamlines in the entire domain. The streamlines start close to the lid at the right corner and gradually grow to occupy the entire square domain with time until the flow reached steady-state. At steady state, there is one primary vortex, three first vortices, at the left and right bottom corners and the top left corner; two second vortices appear at the left and right bottom corners. These vortices and their centers match well with the benchmark values of Ghia et al. [33]. The steady-state horizontal and vertical velocity contours are shown and in Figs. 5.14(a) and 5.14(b). These results match qualitatively very well with the published results of Jiang et al. [49]. Finally, to qualitatively measure the performance of the present formulation, the steady-state u_x velocity profiles along $x = 0.5$ are shown in Fig. 5.14(c) and the steady-state u_y velocity profiles along $y = 0.5$ are shown in Fig. 5.14(d) and compared to Jiang et al. [49]. These results perfectly match with the published results from the literature.

5.3.2.2 Transient flow over backward facing step

In the present example, we wish to investigate the temporal behavior of the two-dimensional flow over a backward-facing step at Re = 800. Again we employ the simplified (truncated) step configuration. The computational domain is taken as $\Omega = [0, 20] \times [-0.5, 0.5]$ and is discretized into a set of 200 rectangular finite

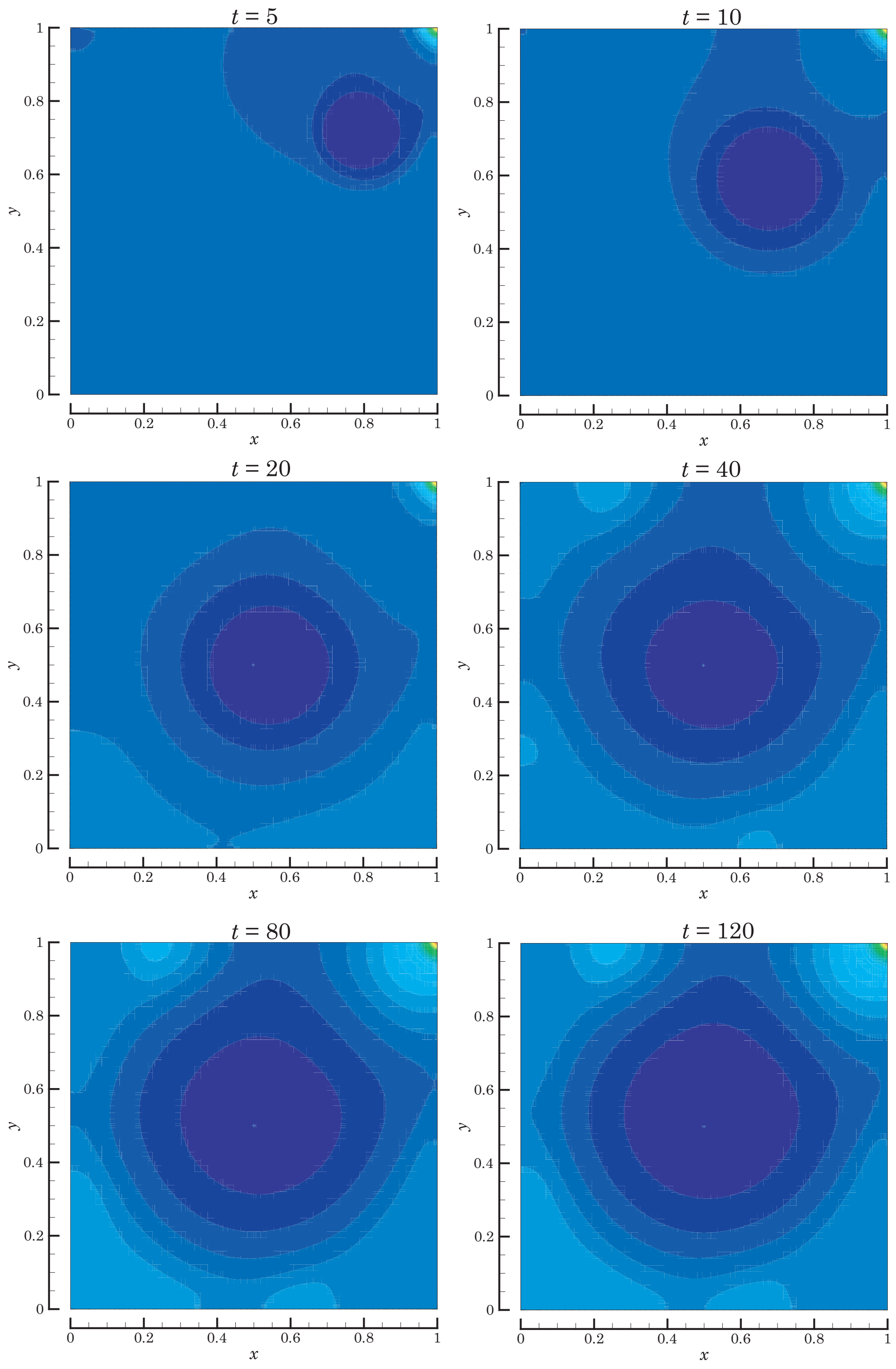


Figure 5.12: Time history of pressure contours.

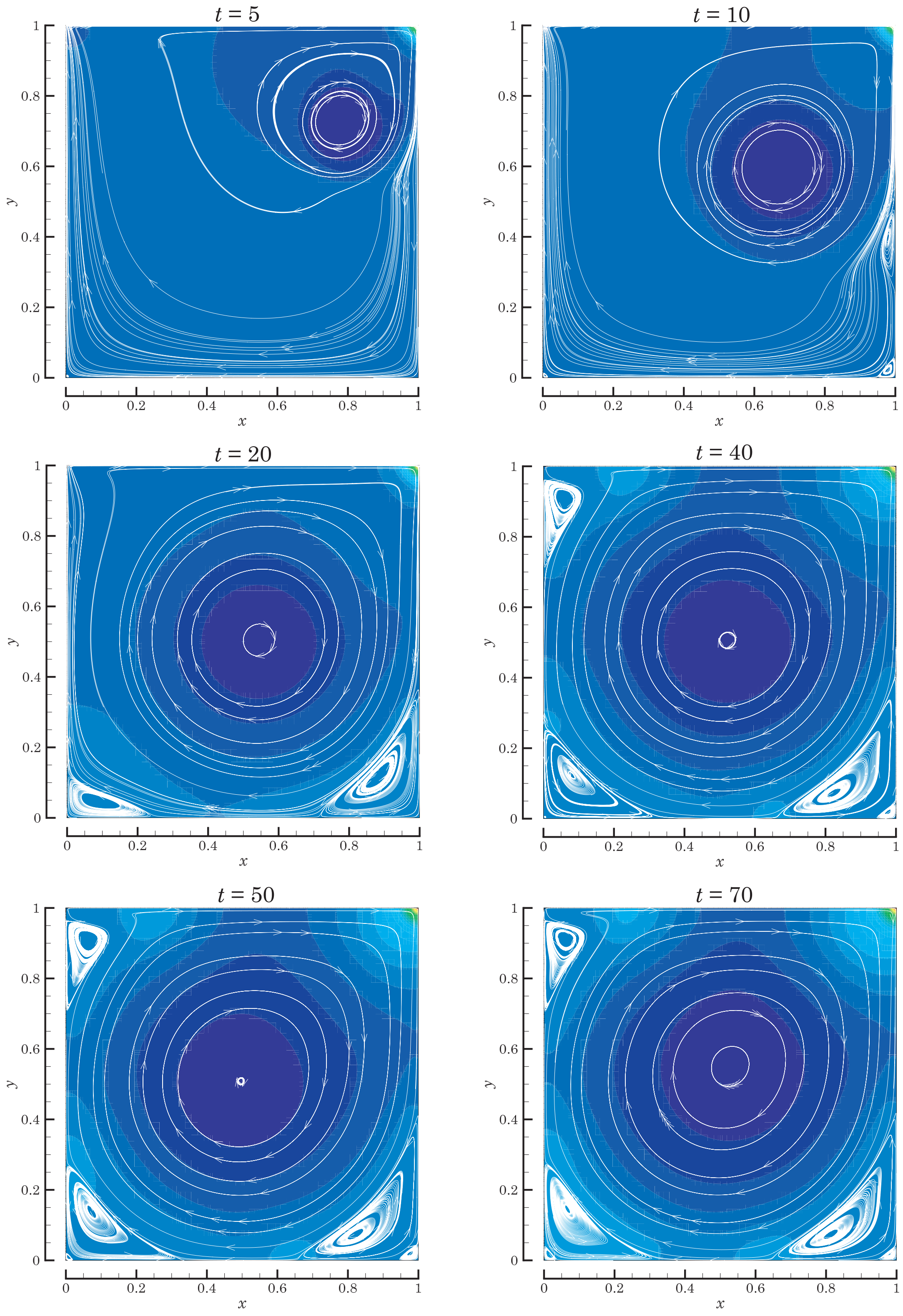


Figure 5.13: Time history of streamtraces.

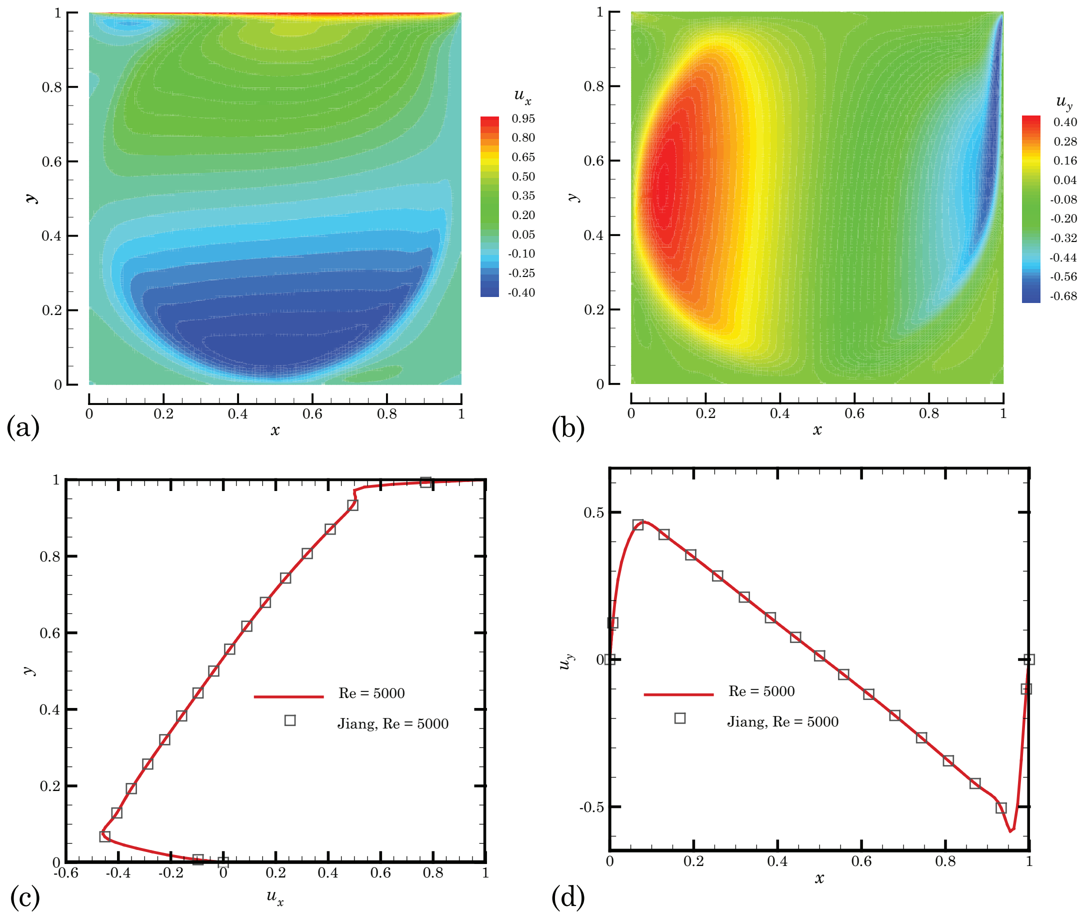


Figure 5.14: (a) Horizontal velocity contours (b) Vertical velocity contours (c) u_x-velocity profiles along $x = 0.5$ and (d) u_y-velocity profiles along $y = 0.5$.

elements, with 40 elements along the channel length and 5 along the channel height. The boundary and initial conditions are taken from the works of Gresho et al. [17] and Pontaza and Reddy [4]. The prescribed velocity vector at the inlet is assumed to be $u_x = u_x^{\mathrm{p}}(y,t) = 24y(0.5-y)[\tanh(t)] + u_x^{\mathrm{I}}(y)[1-\tanh(t)]$ on $0.0 \leqslant y \leqslant 0.5$ and $u_x = u_x^{\mathrm{p}}(y,t) = u_x^{\mathrm{I}}(y)[1-\tanh(t)]$ on $-0.5 \leqslant y \leqslant 0.0$. Here, $u_x^{\mathrm{I}}(y) = 3(0.5-y)(0.5+y)$ is the Poiseuille flow observed infinitely far downstream at steady state. This inlet condition, allows for a smooth but quick transition from Poisuille flow to flow over a backward-facing step, thus avoiding any singularities. The initial conditions are taken as $u_x = u_x^{\mathrm{I}}(y)$ and $u_y = 0$ everywhere in the computational domain. The components of the velocity are taken to be zero along the top and bottom surfaces in accordance with the no-slip condition. The outflow boundary condition is imposed in a weak-sense through the least-squares functional on the outflow (right) section of the domain. The polynomial order of $p = 7$ is used with a total of 10116 nodes. A uniform time step size of $\triangle t = 0.2$ and with a total of 500 steps is employed for the finite element simulation. As in the previous example, we employ the BDF1 formula for the first 4 time steps and then use BDF2 for the rest.

In Fig. 5.15, we plot the time history of streamtraces over the entire domain. It can be seen that the flow history is characterized by a series of primary and secondary separation zones along the top and bottom walls of the channel. At steady-state, most of the small eddies vanish except a primary separation zone on the bottom-wall extending up to 6.1 units beyond the step and a secondary separation zone on the top-wall that develops at 4.9 units downstream of the step and extends to approximately $x = 10.5$. These primary and secondary zone values are in excellent agreement with the numerical and experimental results from the literature [32]. In Fig. 5.16, the time history of pressure contours is shown. The color profiles indicate the gradients of pressure over the entire domain at any instant of time. As time progresses the pressure contours become more distributed and smooth.

As in the steady-state example, we show the time history of numerically post-computed values of the normalized local volume dilation rates ($\mathcal{D}^e$) for each element in the mesh in Fig. 5.17. The magnitude of the dilation rate represents, how well the continuity equation is satisfied over the element. As expected, when the primary and secondary eddies begin to form at top and bottom walls, the conservation is relatively poor in the order 10^{-4}. But, as time progresses it improves and at steady state it reaches an order of 10^{-6} over the entire domain. Further, to qualitatively measure this conservation, we plot the mass flow rates at sections $x = 5$ and $x = 10$ with time in Figs. 5.18(a) and 5.18(b). In the beginning, there are some small fluctuations, but they slowly die-out as time progresses till steady-state is reached.

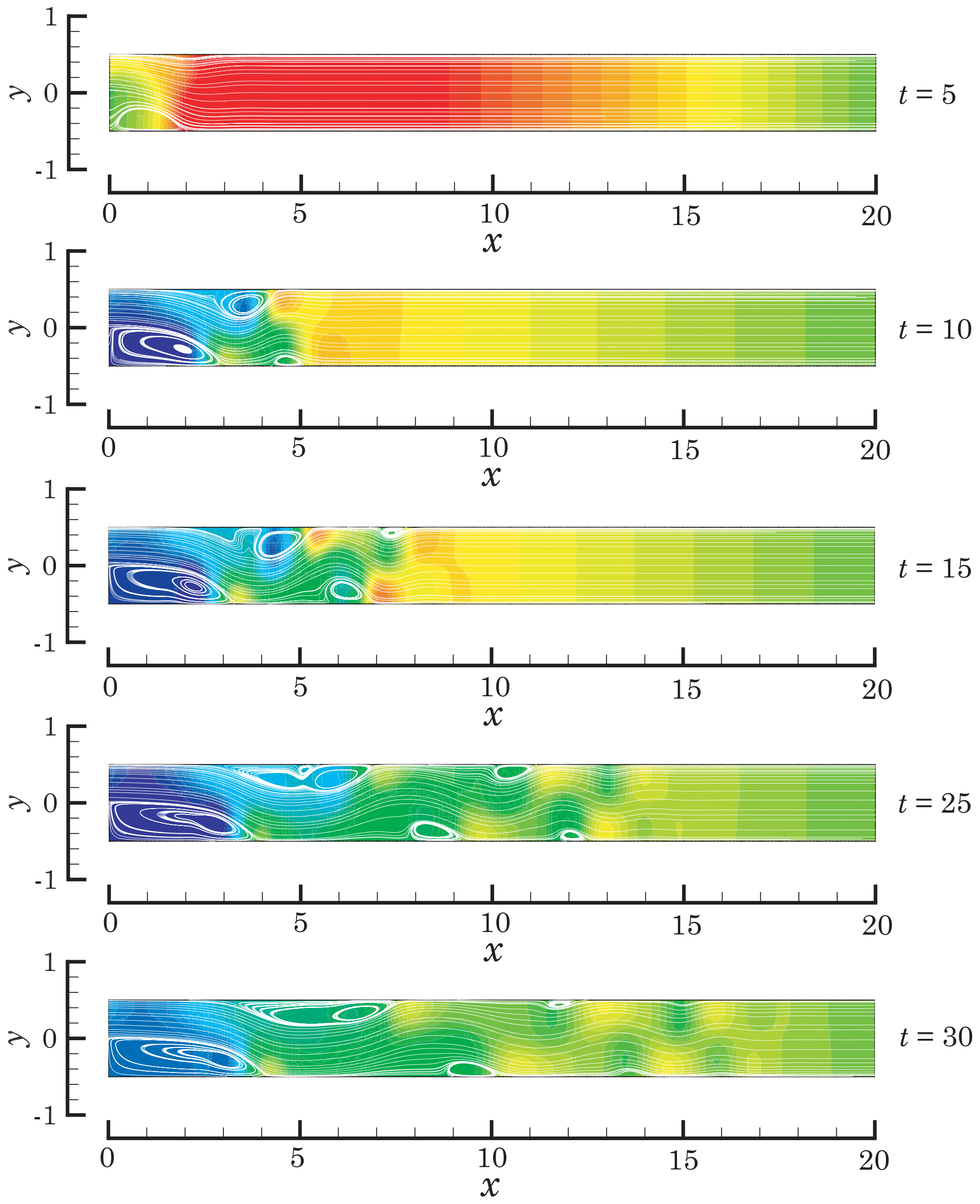


Figure 5.15: Time history of streamtraces over the domain.

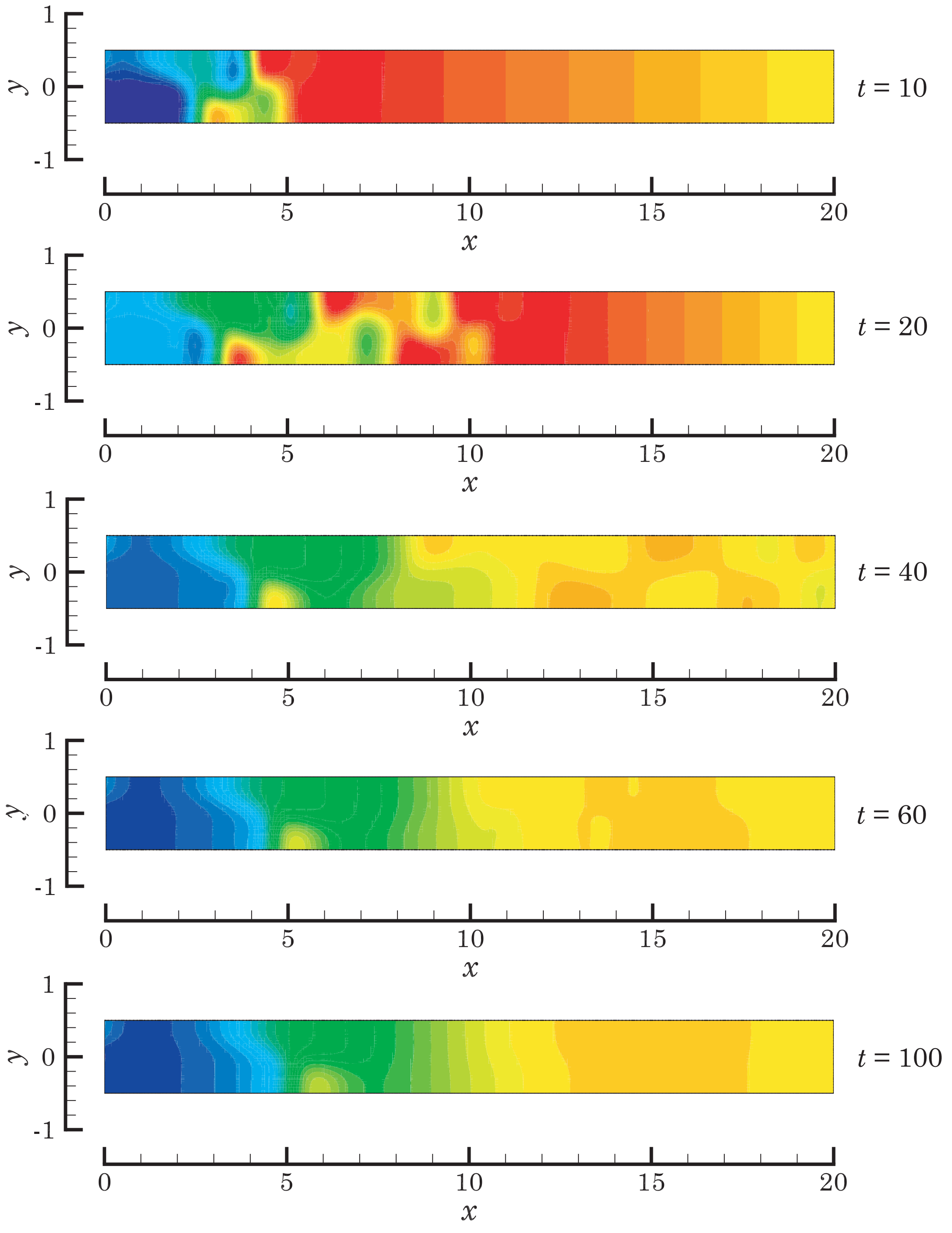


Figure 5.16: Time history of pressure contours over the domain.

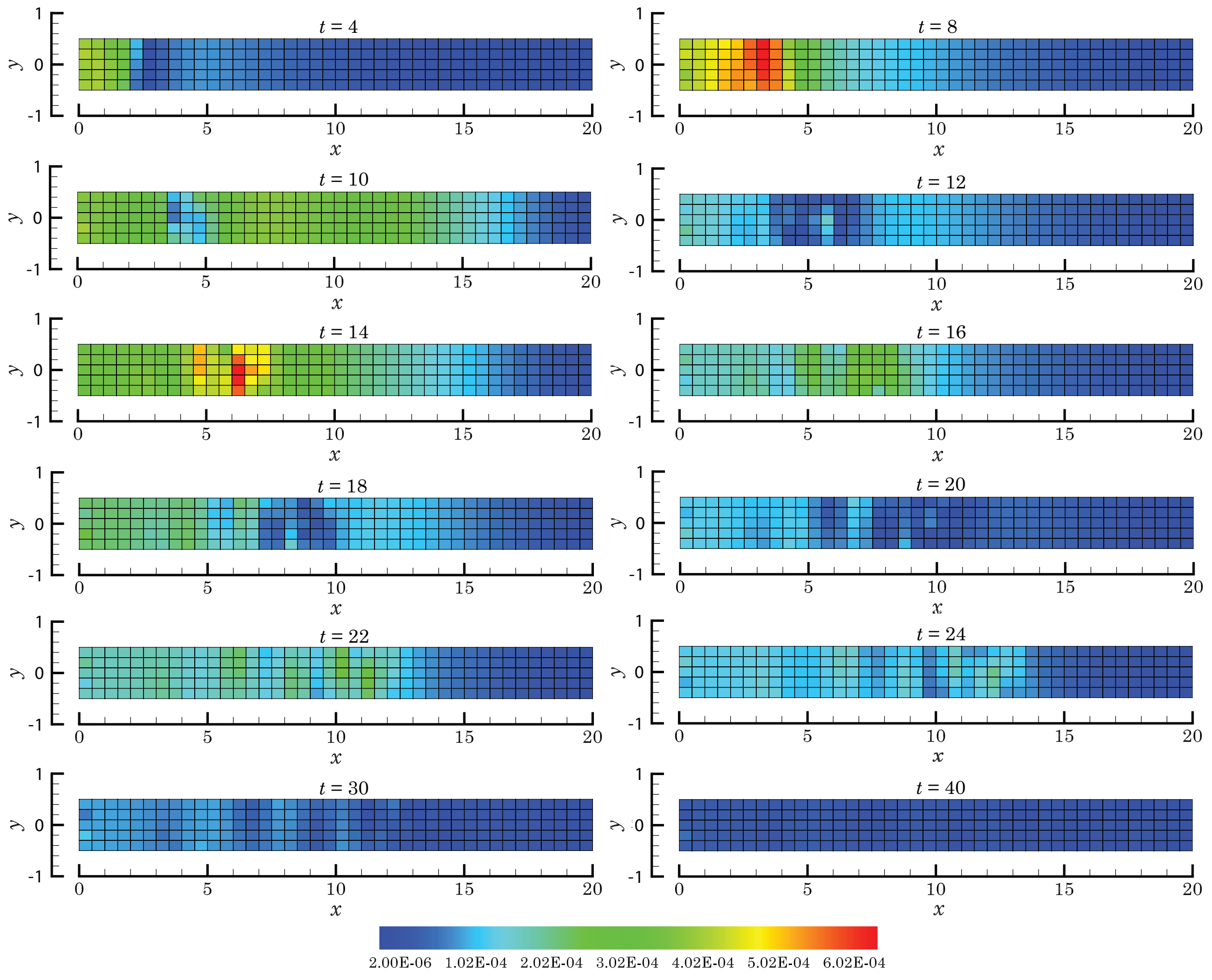
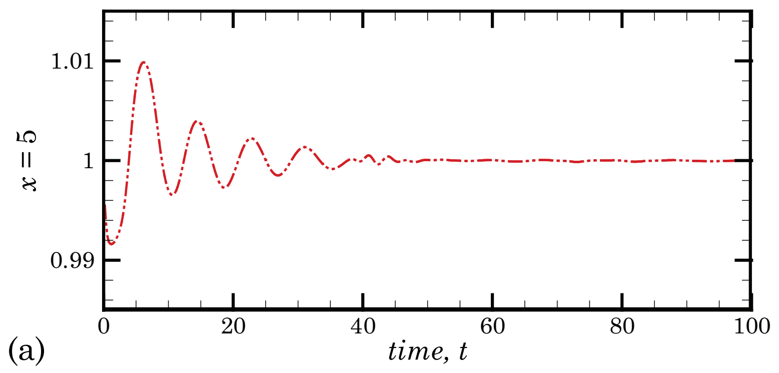


Figure 5.17: Time history of dilatation rate of each element over the domain.

The lack of pressure and velocity coupling in the least-squares formulation results in unrealistic temporal chaotic behavior, resulting in an erroneous prediction of the flow in long-term. In such cases, either the simulation diverges or the field variables fluctuate with time if at all it converges to a steady state [75]. To investigate this behavior, we probe the time history of u_y velocity at points (10,0) and (13,0) in the domain as in Figs . 5.19(b) and 5.19(a). As it is evident, the formulation results in a nice evolution of the solution variables with time and the simulation smoothly



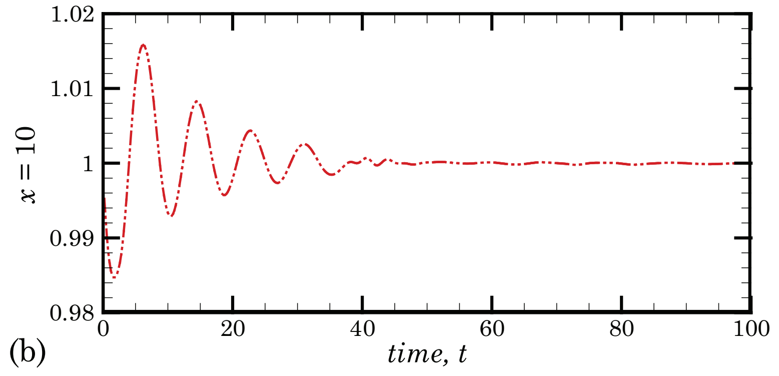
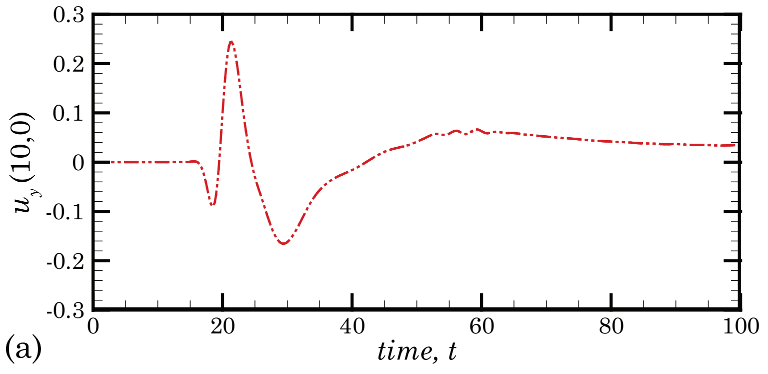


Figure 5.18: (a) Time history of mass flow rates at section $x = 5$ and (b) Time history of mass flow rates at section $x = 10$.

reaches the steady-state. Also, no erroneous fluctuations in the velocity field are observed.

5.3.2.3 Transient flow past a cylinder at Re = 100

Here we consider the case of Re = 100, for which a transient simulation is necessary. The cylinder is of unit diameter and is at the center of the finite domain $\Omega = [-15.5, +25.5] \times [-20.5, +20.5]$ as shown in Fig. 5.9. The mesh has 501 quadrilateral finite elements, with body-fitting mesh around the cylinder. Initially the fluid is at rest and the horizontal velocity is gradually increased using the hyperbolic tan-



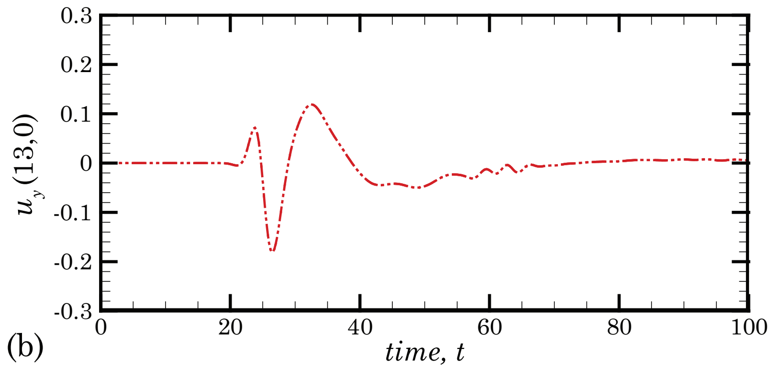


Figure 5.19: (a) Time history of vertical velocity at $(10, 0)$ and (b) Time history of vertical velocity at $(13, 0)$.

gent function in time as $u_x = u_\infty \tanh(t)$ at inflow (left), top and bottom boundaries, where u_∞ is the free-stream velocity and is taken as unity. The vertical velocity is specified as $u_y = 0$ on all these three boundaries. A no-slip boundary condition of $u_x = u_y = 0$ is imposed on the surface of the cylinder. The outflow (right) boundary condition is once again enforced in a weak sense by including the expression $\hat{\mathbf{t}} - \hat{\mathbf{n}} \cdot \underline{\tilde{\boldsymbol{\sigma}}} = 0$ in the definition of the least-squares functional, where pseudo-traction vector on the outflow boundary is taken to be $\hat{\mathbf{t}} = 0$. A uniform time step size of $\triangle t = 0.1$ with a total of 2500 time steps is used for the finite element simulation. As in the previous problem, we use the BDF1 formula for the first 4 time steps and then use BDF2 for the rest. The problem is solved using polynomial order of $p = 5$ with a total of 12775 nodes. At each time step the solution converged in only 2 or 3 nonlinear iterations.

In Fig. 5.20, we show the contour plots of pressure p, horizontal velocity u_x, and vertical velocity u_y, at an instant of $t = 200$.

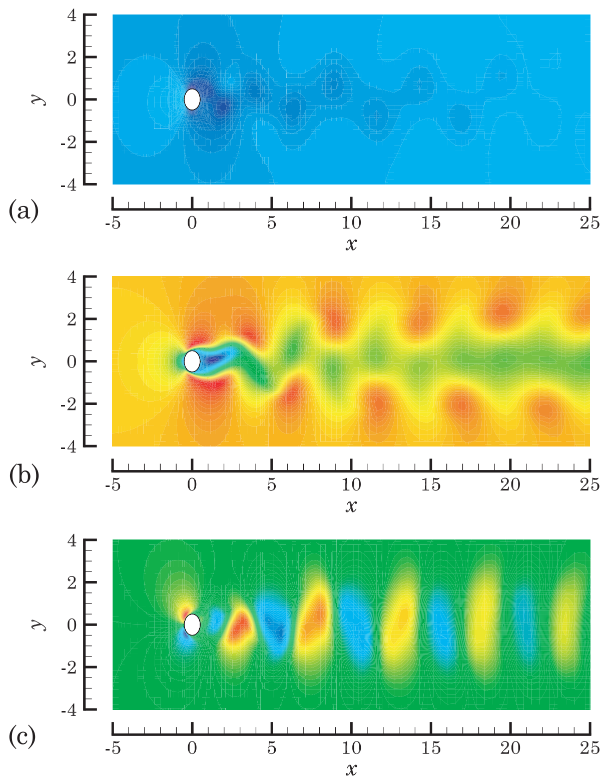
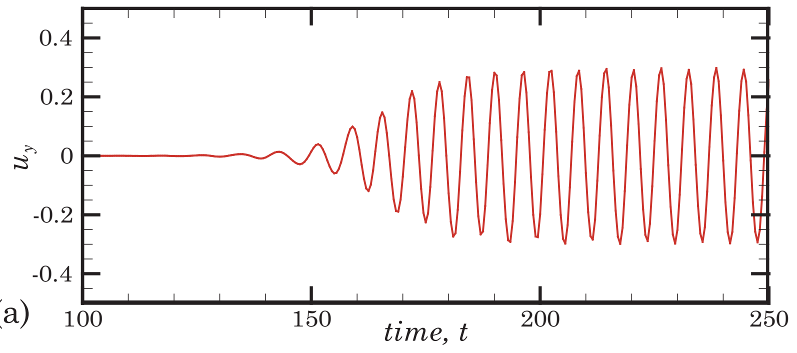


Figure 5.20: Instantaneous contour plots at $t = 200$, of (a) Pressure (b) Horizontal velocity and (c) Vertical velocity.

We probe the time history of u_y velocity at points (1,0) and (2,0) downstream the cylinder and plot in Figs. 5.21(a) and 5.21(b). The formulation results in a correct evolution of solution variables with time and the simulation smoothly reaches the steady-state. Also, no erroneous fluctuations in the field variables are observed.

(a)


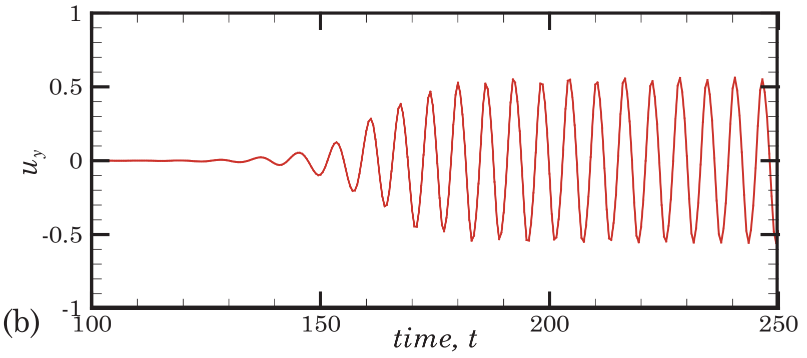
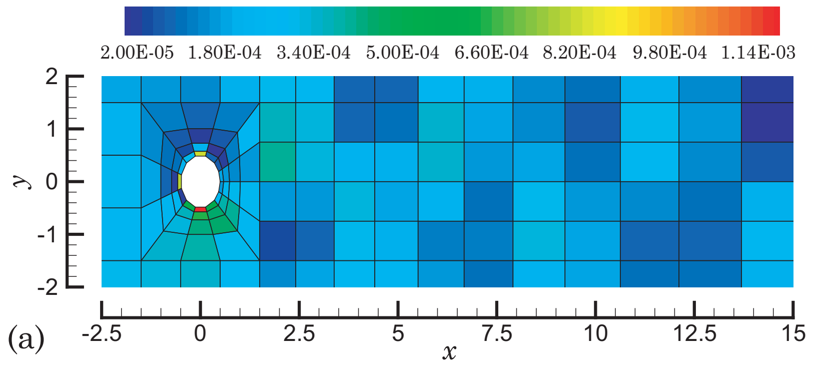
(b)


Figure 5.21: (a) Time history of velocity u_y at points (1,0) (b) Time history of velocity u_y at points (2,0).

Next in Figs. 5.22(a) and 5.22(b), we plot the normalized local volume dilation rates ($\mathcal{D}^e$) for each element in the mesh for $p = 5$ at time instants of $t = 220$ and $t = 240$. These figures, show that the local volume dilatation rate is conserved for all the elements in the space and at all times.



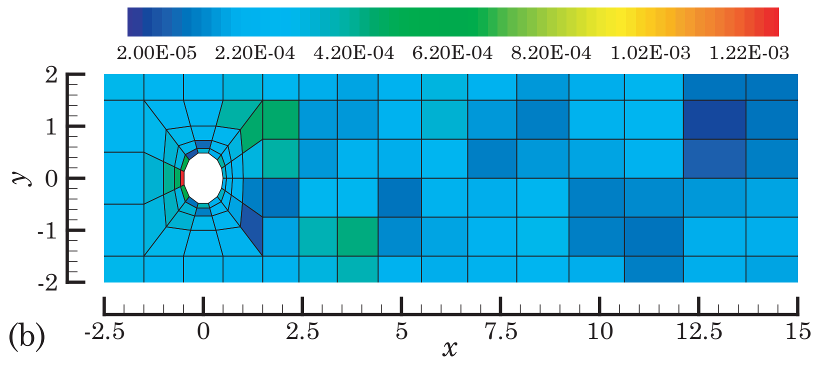


Figure 5.22: (a) Local volume dilatation rates at instants of (a) time $t = 220$ and (b) time $t = 240$.

6. CONCLUSIONS

6.1 Summary

In this study, applications of high-order spectral/*hp* approximation functions in the finite element models of various of nonlinear boundary-value and initial-value problems arising in the fields of structural mechanics and flows of viscous incompressible fluids are presented. The high-order spectral/*hp* formulations offer several theoretical and computational advantages. For structures, the higher-order spectral/*hp* finite element procedures allow us to develop robust structural elements for beams, plates, and shells in a purely displacement-based setting, which avoid all forms of *numerical locking*. For fluid flows, when the high-order spectral/*hp* procedures are combined with least-squares variational principles, such technology allows us to develop efficient finite element models that always yield a symmetric positive-definite (SPD) coefficient matrix, and thereby robust direct or iterative solvers can be used. Also, the least-squares formulations circumvent the infimum-supremum or the Ladyzhenskaya–Babuska–Brezzi (LBB) condition [121, 122, 5, 16] and equal-order interpolations can be used for all field variables of the formulation. When the spectral/*hp* approximations are used with iterative penalization methodology, they results in a better conservation of physical quantities like dilatation, volume, and mass and in stable evolution of variables with time for transient flows. Both in structures and fluid flows, the higher-order spectral/*hp* basis functions avoid the *interpolation error* in the numerical schemes, thereby making them accurate and stable. It is shown that *ad hoc* stabilized methods or *tricks* used to alleviate *numerical locking* and *spurious solution oscillations* in low-order finite elements may be largely circumvented by (a) employing high-order spectral/*hp* finite element technology and

(b) constructing the finite element model for a given physical phenomenon in the context of a true variational setting (i.e., the minimization of a quadratic functional via the potential energy considerations for structures and least-squares principles for fluid flows). Finally, it is amply illustrated that the unconstrained minimization plus high-order spectral/*hp* finite element technology offers a highly attractive numerical setting, avoiding the need for *ad-hoc* approaches that are problem dependent.

6.2 Concluding remarks

In Section 2, a framework for efficient implementation of higher-order spectral/*hp* finite element formulations is developed. The recursive/iterative relations to generate higher-order basis functions and numerical quadratures are presented. We generated higher-order spectral/*hp* grids for general two-dimension domains and domains with simple curved boundaries. Simple and efficient algorithms to generate necessary data structures to apply *natural* and *essential* type of boundary conditions are presented. A general method to perform line (or surface) integration along the boundaries of the element to apply *natural* type of boundary conditions is derived. The schur complement method to condense the element interior nodes is discussed. We presented an abstract higher-order finite element problem and identified the tasks that can be done in parallel. Then, we discussed the ways to find the solutions for large scale problems by linking to serial/parallel, direct/iterative solver libraries. Finally, we presented a numerical example to validate all these methods. The ideas presented in this work are very generic and can be easily adopted to any programming language. They can also be easily extended to higher-dimension finite element codes.

To begin with in Section 3, higher-order spectral/*hp* finite element technology is applied to one-dimension beam problems. A fully-discretized weak-form Galerkin finite element model for linear viscoelastic beam based on the higher-order beam theory (HBT) with the *von Kármán* geometric nonlinearity is developed. The beam is capable of undergoing moderate rotations and small strains (i.e., *von Kármán* geometric nonlinearity). The Higher-Order Beam Theory (HBT) admits C^0 continuous interpolants for all the dependent variables. A recurrence scheme is developed such that history data need only be stored from the previous time step. The performance of the high-order spectral/*hp* finite element technology with-respect-to issues of *numerical locking* is investigated using a variety if thick and thin beams and with different loading and boundary conditions. Both quasi-static and fully-transient problems have been considered. Various non-trivial benchmark examples are considered to demonstrate the capabilities of the developed finite element model in alleviating both *membrane* and *shear* locking.

In Section 4, higher-order spectral/*hp* finite element procedures that allow us to develop robust structural elements for beams, plates, and shells in a purely displacement based setting and which avoid all forms of *numerical locking* are developed. A weak-form Galerkin finite element model is constructed based on an improved fist-order shear deformation theory (FSDT) with seven independent parameters, that allows the use of fully three-dimensional constitutive equations in the numerical implementation. The actual shell mid-surface and the unit normal to it are each approximated using the standard spectral/*hp* finite element interpolation functions. The present formulation requires as input the three-dimensional coordinates of the shell mid-surface as well as a set of directors (i.e., unit normal vectors to the mid-surface), for each node in the shell finite element model. Hence, the formulation allows the use of randomly skewed and curved quadrilateral elements, which will be useful to mesh complicated shell geometries. The terms of virtual work statement are carefully separated to do through-thickness numerical integration at each quadrature point of the mid-surface and hence no thin-shell approximations are imposed in the numerical implementation. The formulation is extended for the analysis of geometrically non-linear response of elastic, isotropic and functionally graded shell structures subjected to mechanical and thermal loadings. Various non-trivial benchmark problems are implemented to demonstrate that the proposed shell element is free of all forms of *numerical locking* even for large geometric deformations.

In Section 5, a stress-based least-squares finite element model of the steady-state and non-stationary incompressible Navier–Stokes equations is developed using higher-order spectral/*hp* basis functions. The least-squares formulation circumvents the infimum-supremum or the Ladyzhenskaya-Babuska-Brezzi (LBB) condition and hence equal-order interpolations are used for all the primary variables. However, the standard L_2-norm least-squares formulation of the first-order stress-based Navier–Stokes equations lack the velocity and pressure coupling and results in a poor evolution (with spurious oscillations) of primary variables with time. To overcome this an iterative penalization scheme is introduced on the similar lines of [34, 75], for the transient pressure-velocity-stress first-order system of Navier–Stokes equations. It also results in a better conservation of physical quantities (like dilatation, volume, mass) and stable evolution of variables with time for transient flows. Finally, numerical solutions of several non-trivial benchmark problems are presented to verify the same.

6.3 Recommendations

The encouraging results from the application of higher-order spectral/*hp* finite element procedures opens several interesting and challenging areas of future work.

In particular the following tasks require attention.

- In Section 3, further studies can be carried out to fully understand the behavior under dynamic loadings and also with large strain capabilities. In particular, the current formulation may be modified in the context of a corotational [116] finite element formulation. It is also a research interest to combine the present formulation with non-local theories of Eringen [29] and gradient theories of Yang et al. [126] and Srinivasa and Reddy [105] for viscoelastic beams. Finally, extension of the present work to viscoelastic plates and shells is awaiting attention.

- In Section 4, the proposed formulation can be extended to do the dynamic and buckling studies of shell structures under mechanical and thermal loadings. Although, we used isotropic elastic material model, it can be applied to a more realistic material models like Cauchy elastic, hyperelastic, viscoelastic, elasto–plastic, and so on. Also, it would be of great interest to investigate how the higher-order shell element performs in the context of continuum damage evolution and fracture in shell structures.

- In Section 5, since the least-squares principle proved successful for Navier–Stokes equations, which are not derivable from variational principles. Similarity, it would be interesting to numerically investigate other equations like the Langevian equation [97]. For these type of equations, the weighted-residual integral statements used in the development of conventional weak-form Galerkin finite element models are not meaningful (i.e. they are not statements of minimization of error due to the approximation of the field variables) when applied to the Langevian equation. Since the higher-order formulation results in a better conservation of physical quantities, it would of great engineering importance to apply these methods for the analysis of flow through porous media, Darcy's flow and reservoir simulation. Also since the iterative penalization strategy [34, 75] results in a better evolution of primary variables like pressure and velocity, it can be applied to multi-phase fluid flow reservoir simulations.

- The higher-order spectral/*hp* formulations avoid *interpolation errors* in the numerical schemes and make them more accurate and stable. For typical engineering structures like beams, plates and shells it avoids all kinds of *numerical locking*. For fluid flows, the combination of higher-order spectral/*hp* technology with least-variational principles and iterative penalization schemes resulted in better conservation of physical quantities (like dilatation, volume, mass) and also in better evolution of primary variables like pressure and velocity. These in-turn make the numerical schemes more accurate and stable. Due to the improved stability and accuracy of numerical schemes for fluids and structures, the proposed higher-order spectral/*hp* formulations can be extended to study coupled fluid-structure interaction (FSI) problems [111] and moving-domain Arbitrary Lagrangian Eulerian (ALE) problems [39, 27]. They can be used in monolithic or staggered FSI schemes given in [123, 61].

REFERENCES

[1] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover Publications, New York, 1 edition, 1964.

[2] Y. Aköz and F. Kadioğlu. The mixed finite element method for the quasi-static and dynamic analysis of viscoelastic Timoshenko beams. *Int. J. Numer. Meth. Engng*, 44(12):1909–1932, 1999.

[3] R. A. Arciniega and J. N. Reddy. Large deformation analysis of functionally graded shells. *Int. J. Solids Struct.*, 44(6):2036–2052, 2007.

[4] R. A. Arciniega and J. N. Reddy. Tensor-based finite element formulation for geometrically nonlinear analysis of shell structures. *Comput. Methods Appl. Mech. Engrg.*, 196(46):1048–1073, 2007.

[5] I. Babuska. Errors bounds for finite element method. *Numerische Mathematik*, 16(4):322–333, 1971.

[6] V. Balamurugan and S. Narayanan. Active-passive hybrid damping in beams with enhanced smart constrained layer treatment. *Eng. Struct.*, 24(3):355–363, 2002.

[7] V. Balamurugan and S. Narayanan. Finite element formulation and active vibration control study on beams using smart constrained layer damping (SCLD) treatment. *J. Sound Vib.*, 249(2):227–250, 2002.

[8] A. Barut, E. Madenci, and A. Tessler. Nonlinear analysis of laminates through a Mindlin-type shear deformable shallow shell element. *Comput. Methods Appl. Mech. Engrg.*, 143(1-2):155–173, 1997.

[9] B. C. Bell and K. S. Surana. A space-time coupled p-version least-squares finite element formulation for unsteady fluid dynamics problems. *Int. J. Numer. Meth. Engng*, 37(20):3545–3569, 1994.

[10] B. C. Bell and K. S. Surana. A space-time coupled p-version least squares finite element formulation for unsteady two-dimensional Navier-Stokes equations. *Int. J. Numer. Meth. Engng*, 39(15):2593–2618, 1996.

[11] T. Belytschko, W. K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley and Sons, Ltd, New York, 2000.

[12] M. Bischoff and E. Ramm. Shear deformable shell elements for large strains and rotations. *Int. J. Numer. Meth. Engng*, 40(23):4427–4449, 1997.

[13] M. Bischoff and E. Ramm. On the physical significance of higher order kinematic and static variables in a three-dimensional shell formulation. *Int. J. Solids Struct.*, 37(4647):6933–6960, 2000.

[14] P. B. Bochev and M. D. Gunzburger. *Least-Squares Finite Element Methods*. Springer, New York, 2009.

[15] B. Brank, F. B. Damjanić, and D. Perić. On implementation of a nonlinear four node shell finite element for thin multilayered elastic shells. *Comput. Mech.*, 16:341–359, 1995.

[16] F. Brezzi and K. J. Bathe. A discourse on the stability conditions for mixed finite element formulations. *Comput. Methods Appl. Mech. Engrg.*, 82(1-3):27–57, 1990.

[17] F. Brezzi and J. Douglas. Stabilized mixed methods for the Stokes problem. *Numer. Math.*, 53:225–235, 1988.

[18] A. N. Brooks and T. J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32(1–3):199–259, 1982.

[19] N. Büechter and E. Ramm. Shell theory versus degeneration–a comparison in large rotation finite element analysis. *Int. J. Numer. Meth. Engng*, 34(1):39–59, 1992.

[20] D. Chapelle and K. J. Bathe. *The Finite Element Analysis of Shells - Fundamentals.* Springer-Verlag, New York, 2003.

[21] Q. Chen and Y. W. Chan. Integral finite element method for dynamical analysis of elastic-viscoelastic composite structures. *Comput. Struct.*, 74(1):51–64, 2000.

[22] T.-M. Chen. The hybrid Laplace transform/finite element method applied to the quasi–static and dynamic analysis of viscoelastic Timoshenko beams. *Int. J. Numer. Meth. Engng*, 38(3):509–522, 1995.

[23] R. M. Christensen. *Theory of Viscoelasticity.* Academic Press, New York, 2nd edition, 1982.

[24] W. Couzy and M. O. Deville. A fast schur complement method for the spectral element discretization of the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 116(1):135–142, 1995.

[25] M. A. Crisfield. A fast incremental/iterative solution procedure that handles "snap-through". *Comput. Struct.*, 13(13):55–62, 1981.

[26] J. M. Deang and M. D. Gunzburger. Issues related to least-squares finite element methods for the Stokes equations. *SIAM J. Sci. Comput.*, 20(3):878–906, 1998.

[27] J. Donea, S. Giuliani, and J. P. Halleux. An arbitrary lagrangian-eulerian finite element method for transient dynamic fluid-structure interactions. *Comput. Methods Appl. Mech. Engrg.*, 33(13):689–723, 1982.

[28] E. N. Dvorkin and K. J. Bathe. A continuum mechanics based four-node shell element for general non-linear analysis. *Eng. Computation*, 1(1):77–88, 1984.

[29] A. C. Eringen. Nonlocal polar elastic continua. *Int. J. Eng. Sci.*, 10:1–16, 1972.

[30] W. N. Findley, J. S. Lai, and K. Onaran. *Creep and relaxation of nonlinear viscoelastic materials.* North-Holland Pub. Co., New York, 1976.

[31] W. Flügge. *Viscoelasticity.* Springer, Berlin, Heidelberg, 2nd edition, 1975.

[32] D. K. Gartling. A test problem for outflow boundary conditions - flow over a backward-facing step. *Int. J. Numer. Meth. Fluids*, 11:953–967, 1990.

[33] U. Ghia, K. N. Ghia, and C. T. Shin. High-Re solution for incompressible flow using the Navier-Stokes equations and the multigrid method. *J. Comput. Phys.*, 48:387–411, 1982.

[34] M. D. Gunzburger. *Finite Element Methods for Viscous Incompressible Flows.* Academic Press, London, 1989.

[35] S. Hartmann. Computation in finite-strain viscoelasticity: finite elements based on the interpretation as differential-algebraic equations. *Comput. Methods Appl. Mech. Engrg.*, 191(13–14):1439–1470, 2002.

[36] M. Henriksen. Nonlinear viscoelastic stress analysis-a finite element approach. *Comput. Struct.*, 18:133–139, 1984.

[37] H. M. Hilber, T. J. R. Hughes, and R. L. Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Eng. Struct. Dyn.*, 5:283–292, 1977.

[38] E. Hinton and H. C. Huang. A family of quadrilateral Mindlin plate elements with substitute shear strain fields. *Comput. Struct.*, 23(3):409–431, 1986.

[39] C. W. Hirt, A. A. Amsden, and J. L. Cook. An arbitrary lagrangian-eulerian computing method for all flow speeds. *J. Comput. Phys.*, 14(3):227–253, 1974.

[40] G. A. Holzapfel and G. Reiter. Fully coupled thermomechanical behaviour of viscoelastic solids treated with finite elements. *Int. J. Eng. Sci.*, 33(7):1037–1058, 1995.

[41] T. J. R. Hughes and A. N. Brooks. Multi-dimensional upwind scheme with no crosswind diffusion. *American Society of Mechanical Engineers, Applied Mechanics Division, AMD*, 34:19–35, 1979.

[42] T. J. R. Hughes, L. P. Franca, and G. M. Hulbert. A new finite element formulation for computational fluid dynamics: VIII. the Galerkin/least-squares method for advective-diffusive equations. *Comput. Methods Appl. Mech. Engrg.*, 73(2):173–189, 1989.

[43] R. Javaheri and M. R. Eslami. Thermal buckling of functionally graded plates. *AIAA Journal*, 40:162–169, 2002.

[44] B. N. Jiang. A least-squares finite element method for incompressible Navier-Stokes problems. *Int. J. Numer. Meth. Fluids*, 14(7):843–859, 1992.

[45] B. N. Jiang. *The Least-Squares Finite Element Method*. Springer-Verlag, New York, 1998.

[46] B. N. Jiang. On the least-squares method. *Comput. Methods Appl. Mech. Engrg.*, 152(1–2):239–257, 1998.

[47] B. N. Jiang and C. L. Chang. Least-squares finite elements for the Stokes problem. *Comput. Methods Appl. Mech. Engrg.*, 78(3):297–311, 1990.

[48] B. N. Jiang, T. L. Lin, and L. A. Povinelli. Large-scale computation of incompressible viscous flow by least-squares finite element method. *Comput. Methods Appl. Mech. Engrg.*, 114(3-4):213 – 231, 1994.

[49] B. N. Jiang and L. A. Povinelli. Least-squares finite element method for fluid dynamics. *Comput. Methods Appl. Mech. Engrg.*, 81(1):13 – 37, 1990.

[50] A. R. Johnson, A. Tessler, and M. Dambach. Dynamics of thick viscoelastic beams. *J. Eng. Mater. Technol.*, 119(3):273–278, 1997.

[51] G. E. Karniadakis and S. J. Sherwin. *Spectral/hp Element Methods for CFD.* Oxford University Press, Oxford, 1999.

[52] M. Kawaguti and P. Jain. Numerical study of a viscous fluid past a circular cylinder. *J. Phys. Soc. Japan*, 21:2055–2062, 1966.

[53] G. Kirchhoff. Uber das gleichgwich und die bewegung einer elastischen scheibe. *Journal für die reine und angewandte Mathematik*, 40:51–88, 1850.

[54] M. Koizumi. Fgm activities in japan. *Compos. Part B: Eng.*, 28B(4):1–4, 1997.

[55] D.W. Van Krevelen. *Properties of Polymers.* Elsevier, Amsterdam, Netherlands, 3rd edition, 1990.

[56] J. H. Kweon and C. S. Hong. An improved arc-length method for postbuckling analysis of composite cylindrical panels. *Comput. Struct.*, 53(3):541–549, 1994.

[57] J. Lai and A. Bakker. 3-D Schapery representation for non-linear viscoelasticity and finite element implementation. *Comput. Mech.*, 18:182–191, 1996.

[58] R. S. Lakes. *Viscoelastic Materials.* Cambridge University Press, New York, 1st edition, 2009.

[59] W. Lanhe. Thermal buckling of a simply supported moderately thick rectangular fgm plate. *Compos. Struct.*, 64:211–218, 2004.

[60] F. J. Lockett. *Nonlinear Viscoelastic Solids.* Academic Press, London, 1972.

[61] H. G. Matthies and J. Steindorf. Partitioned but strongly coupled iteration schemes for nonlinear fluid-structure interaction. *Comput. Struct.*, 80(27-30):1991–1999, 2002.

[62] D. J. McTavish and P. C. Hughes. Finite element modeling of linear viscoelastic structures - the GHM method. In *AIAA/ASME/ASCE/AHS/ASC Structures*, Structural Dynamics and Materials Conference, pages 1753–1763, Dallas, TX; United States, April 1992.

[63] D. J. McTavish and P. C. Hughes. Modeling of linear viscoelastic space structures. *J. Vib. Acoust.*, 115(1):103–110, 1993.

[64] R. Naghdabadi and S. A. Hosseini Kordkheili. A finite element formulation for analysis of functionally graded plates and shells. *Arch. Appl. Mech.*, 74:375–386, 2005.

[65] R. Naghdabadi and S. A. Hosseini Kordkheili. Geometrically non-linear thermoelastic analysis of functionally graded shells using finite element method. *Int. J. Numer. Meth. Engng*, 72:964–986, 2007.

[66] P. M. Naghdi. *Foundations of Elastic Shell Theory, in:* I. N. Sneddon, R. Hill (Eds.), Progress in Solid Mechanics, Vol. 4. North-Holland, Amsterdam, 1963.

[67] P. M. Naghdi. *Theory of Shells and Plates, in:* S. Flügge, C. Truesdell (Eds.), Handbuch der Physik, VIa/2. Springer-Verlag, Berlin, 1972.

[68] N. M. Newmark. A method of computation for structural dynamics. *J. Eng. Mech.*, 85:67–94, 1959.

[69] J. T. Oden and W. H. Armstrong. Analysis of nonlinear, dynamic coupled thermoviscoelasticity problems by the finite element method. *Comput. Struct.*, 1(4):603–621, 1971.

[70] A. Pálfalvi. A comparison of finite element formulations for dynamics of viscoelastic beams. *Finite Elem. Anal. Des.*, 44(14):814–818, 2008.

[71] H. Parisch. Large displacements of shells including material nonlinearities. *Comput. Methods Appl. Mech. Engrg.*, 27(2):183–214, 1981.

[72] G. S. Payette and J. N. Reddy. Nonlinear quasi-static finite element formulations for viscoelastic euler–bernoulli and timoshenko beams. *Comm. Numer. Meth. Eng.*, 26(12):1736–1755, 2010.

[73] G. S. Payette and J. N. Reddy. A robust general locking-free shell element for large deformation analysis of FGM and composite structures. *Comput. Meth. Appl. Mech. Eng.*, (*accepted*), 2013.

[74] G. S. Payette, V. P. Vallala, and J. N. Reddy. An OpenMP parallel algorithm to setup compressed sparse column (CSC) and compressed sparse row (CSR) linear systems for higher-order finite elements and link to modern solvers. (*in preparation*).

[75] J. P. Pontaza. A least-squares finite element formulation for unsteady incompressible flows with improved velocity-pressure coupling. *J. Comput. Phys.*, 217(2):563–588, 2006.

[76] J. P. Pontaza and J. N. Reddy. Spectral/*hp* least-squares finite element formulation for the Navier-Stokes equations. *J. Comput. Phys.*, 190(2):523–549, 2003.

[77] J. P. Pontaza and J. N. Reddy. Mixed plate bending elements based on least-squares formulation. *Int. J. Numer. Meth. Engng*, 60(5):891–922, 2004.

[78] J. P. Pontaza and J. N. Reddy. Space-time coupled spectral/*hp* least-squares finite element formulation for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 197(2):418–459, 2004.

[79] J. P. Pontaza and J. N. Reddy. Least-squares finite element formulation for shear-deformable shells. *Comput. Methods Appl. Mech. Engrg.*, 194(2124):2464–2493, 2005.

[80] J. P. Pontaza and J. N. Reddy. Least-squares finite element formulations for viscous incompressible and compressible fluid flows. *Comput. Methods Appl. Mech. Engrg.*, 195(19–22):2454–2494, 2006.

[81] G. N. Praveen and J. N. Reddy. Nonlinear transient thermoelastic analysis of functionally graded ceramic-metal plates. *Int. J. Solids Struct.*, 35:4457–4476, 1998.

[82] J. N. Reddy. Penalty-finite-element analysis of 3-d Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 35:87–97, 1982.

[83] J. N. Reddy. A general non-linear third-order theory of plates with moderate thickness. *Int. J. Non Linear Mech.*, 25(6):677–686, 1997.

[84] J. N. Reddy. On locking-free shear deformable beam finite elements. *Comput. Methods Appl. Mech. Engrg.*, 149(1-4):113–132, 1997.

[85] J. N. Reddy. *Theory and Analysis of Elastic Plates.* Taylor and Francis, Philadelphia, 1999.

[86] J. N. Reddy. Analysis of functionally graded plates. *Int. J. Numer. Meth. Engrg.*, 47:663–684, 2000.

[87] J. N. Reddy. *Energy Principles and Variational Methods in Applied Mechanics.* John Wiley and Sons, Ltd, New York, 2nd edition, 2002.

[88] J. N. Reddy. *An Introduction to Nonlinear Finite Element Analysis.* Oxford University Press, Oxford, 2004.

[89] J. N. Reddy. *Mechanics of Laminated Composite Plates and Shells: Theory and Analysis.* CRC Press, Boca Raton, FL, 2 edition, 2004.

[90] J. N. Reddy. *An Introduction to the Finite Element Method.* McGraw-Hill, New York, 3rd edition, 2006.

[91] J. N. Reddy. Nonlocal theories for bending, buckling, and vibration of beams. *Int. J. Eng. Sci.*, 45:288–307, 2007.

[92] J. N. Reddy. A general nonlinear third-order theory of functionally graded plates. *Int. J. Aerosp. Lightweight Struct.*, 1:1–21, 2011.

[93] J. N. Reddy. *Introduction to Continuum Mechanics (covering elasticity, fluid mechanics, heat transfer, and viscoelasticity).* Cambridge University Press, New York, 2nd edition, 2013.

[94] E. Riks. An incremental approach to the solution of snapping and buckling problems. *Int. J. Solids Struct.*, 15(7):529–551, 1979.

[95] E. Riks. Some computational aspects of the stability analysis of nonlinear structures. *Comput. Methods Appl. Mech. Engrg.*, 47(3):219–259, 1984.

[96] S. Roy and J. N. Reddy. A finite element analysis of adhesively bonded composite joints with moisture diffusion and delayed failure. *Comput. Struct.*, 29:1011–1031, 1988.

[97] R. Sadr, H. Li, and M. Yoda. Impact of hindered brownian diffusion on the accuracy of particle-image velocimetry using evanescent-wave illumination. *Exp. Fluids*, 38:90–98, 2005.

[98] A. F. Saleeb, T. Y. Chang, W. Graf, and S. Yingyeunyong. Hybrid/mixed model for non-linear shell analysis and its applications to large-rotation problems. *Int. J. Numer. Meth. Engng*, 29(2):407–446, 1990.

[99] C. Sansour. A theory and finite element formulation of shells at finite deformations involving thickness change: Circumventing the use of a rotation tensor. *Arch. Appl. Mech.*, 65:194–216, 1995.

[100] C. Sansour and F. G. Kollmann. Families of 4-node and 9-node finite elements for a finite deformation shell theory. An assesment of hybrid stress, hybrid strain and enhanced strain elements. *Comput. Mech.*, 24:435–447, 2000.

[101] H. R. Schwarz. *Finite Element Methods*. Academic Press, London, 1988.

[102] J. C. Simo and T. J. R. Hughes. *Computational Inelasticity*. Springer-Verlag, Berlin, 1998.

[103] J. C. Simo and M. S. Rifai. A class of mixed assumed strain methods and the method of incompatible modes. *Int. J. Numer. Meth. Engng*, 29(8):1595–1638, 1990.

[104] P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods*. Chapman and Hall/CRC, New York, 1 edition, 2003.

[105] A. Srinivasa and J. N. Reddy. A model for a constrained, finitely deforming, elastic solid with rotation gradient dependent strain energy, and its specialization to von Kármán plates and beams. *J. Mech. Phys. Solids*, 61(3):873–885, 2013.

[106] K. S. Surana, A. R. Ahmadi, and J. N. Reddy. The k-version of finite element method for nonlinear operators in bvp. *Int. J. Comput. Eng. Sci.*, 5(1):133–207, 2004.

[107] S. Suresh and A. Mortensen. *Fundamentals of Functionally Graded Materials.* IOM Commun. Ltd, Cambridge, 1998.

[108] K. Y. Sze, W. K. Chan, and T. H. H. Pian. An eight-node hybrid-stress solid-shell element for geometric non-linear analysis of elastic shells. *Int. J. Numer. Meth. Engng*, 55(7):853–878, 2002.

[109] K. Y. Sze, X. H. Liu, and S. H. Lo. Popular benchmark problems for geometric nonlinear analysis of shells. *Finite Elem. Anal. Des.*, 40(11):1551–1569, 2004.

[110] K. Y. Sze and S. J. Zheng. A stabilized hybrid-stress solid element for geometrically nonlinear homogeneous and laminated shell analyses. *Comput. Methods Appl. Mech. Engrg.*, 191(17–18):1945–1966, 2002.

[111] P. Le Tallec and J. Mouro. Fluid structure interaction with large structural displacements. *Comput. Methods Appl. Mech. Engrg.*, 190(24-25):3039–3067, 2001.

[112] R. L. Taylor, K. S. Pister, and G. L. Goudreau. Thermomechanical analysis of viscoelastic solids. *Int. J. Numer. Meth. Engng*, 2(1):45–59, 1970.

[113] B. Temel, F. F. Calim, and N. Tütüncü. Quasi-static and dynamic response of viscoelastic helical rods. *J. Sound Vib.*, 271(3–5):921–935, 2004.

[114] C. W. S. To and M. L. Liu. Hybrid strain based three node flat triangular shell elements-II. Numerical investigation of nonlinear problems. *Comput. Struct.*, 54(6):1057–1076, 1995.

[115] M. A. Trindade, A. Benjeddou, and R. Ohayon. Finite element modelling of hybrid active-passive vibration damping of multilayer piezoelectric sandwich beams-Part I: Formulation. *Int. J. Numer. Meth. Engng*, 51(7):835–854, 2001.

[116] Y. Urthaler and J. N. Reddy. A corotational finite element formulation for the analysis of planar beams. *Commun. Numer. Methods Eng.*, 21(10):553–570, 2005.

[117] V. P. Vallala, G. S. Payette, and J. N. Reddy. On the art of higher-order spectral/*hp* finite element methods - mesh generation, boundary conditions and schur complement. *Eng. Computation* (*submitted*).

[118] V. P. Vallala, G. S. Payette, and J. N. Reddy. A spectral/*hp* nonlinear finite element analysis of higher-order beam theory with viscoelasticity. *Int. J. Appl. Mech.*, 4(1):43–57, 2012.

[119] V. P. Vallala and J. N. Reddy. Thermo-mechanical analysis of non-linear functionally graded shells using a novel 7-parameter formulation with higher-order methods. (*in preparation*).

[120] V. P. Vallala, A. Ruimi, and J. N. Reddy. Nonlinear viscoelastic analysis of orthotropic beams using a general third-order theory. *Compos. Struct.*, 94(12):3759–3768, 2012.

[121] V. P. Vallala, R. Sadr, and J. N. Reddy. Higher-order spectral/*hp* finite element models of the Navier-stokes equations. *Int. J. Comput. Fluid Dyn.* (*submitted*).

[122] V. P. Vallala, K. S. Surana, and J. N. Reddy. Alternative least-squares finite element models of Navier-Stokes equations for power-law fluids. *Eng. Computation*, 28(7):828–852, 2012.

[123] W. A. Wall, S. Genkinger, and E. Ramm. A strong coupling partitioned approach for fluid-structure interaction with free surfaces. *Comput. and Fluids*, 36(1):169–183, 2007.

[124] Gerald Wempner and G. Demosthenes Talaslidis. *Mechanics of Solids and Shells: Theories and Approximations.* CRC Press, Boca Raton, FL, 2002.

[125] E. L. Wilson. A computer program for the dynamic stress analysis of underground structures. *SESM Report No. 68-1*, University of California, Berkeley, 1968.

[126] F. Yang, A. C. M. Chong, D. C. C. Lam , and P. Tong. Couple stress based strain gradient theory for elasticity. *Int. J. Solids Struc.*, 39:2731–2743, 2002.

APPENDIX A

The Matlab code to (a) Generate element level B.C.'s and (b) Apply *essential* B.C.'s.

```matlab
%-------------------------------------------------------------------
%
% Determine total number of boundary conditions (element wise) and
% pointers to element boundary conditions
%
%-------------------------------------------------------------------

% Initialize elemental total quantities
ess1E_Total    = 0;
ess2E_Total    = 0;
tE_Total       = 0;

% Initialize pointers to boundary conditions
ess1E_Pointer    = zeros(mesh.NE+1,1);
ess2E_Pointer    = zeros(mesh.NE+1,1);
tE_Pointer       = zeros(mesh.NE+1,1);

ess1E_Pointer(1,1)    = 1;
ess2E_Pointer(1,1)    = 1;
tE_Pointer(1,1)       = 1;

% Determine number of boundary conditions
for j = 1:1:mesh.NE
    % Initialize pointers
    ess1E_Pointer(j+1,1)    = ess1E_Pointer(j,1);
    ess2E_Pointer(j+1,1)    = ess2E_Pointer(j,1);
    tE_Pointer(j+1,1)       = tE_Pointer(j,1);

    for i = 1:1:mesh.ETYPE*mesh.ETYPE
        % Number of ess1 boundary conditions
        for k = 1:1:boundary.ess1_Number
            if ( mesh.ECON(i,j) == boundary.ess1_Nodes(k,1) )
                 ess1E_Total          = ess1E_Total + 1;
                 ess1E_Pointer(j+1,1) = ess1E_Pointer(j+1,1) + 1;
            end
        end

        % Number of ess2 boundary conditions
        for k = 1:1:boundary.ess2_Number
            if ( mesh.ECON(i,j) == boundary.ess2_Nodes(k,1) )
                 ess2E_Total          = ess2E_Total + 1;
                 ess2E_Pointer(j+1,1) = ess2E_Pointer(j+1,1) + 1;
            end
        end
    end

    % Number of traction boundary conditions
    for k = 1:1:boundary.t_Number
        if ( j == boundary.t_Elements(k,1) )
            tE_Total           = tE_Total + 1;
            tE_Pointer(j+1,1) = tE_Pointer(j+1,1) + 1;
        end
    end
end
```

```matlab
%-------------------------------------------------------------------
%
% Create pointers to the boundary conditions for each element
%
%-------------------------------------------------------------------

% Initialize quantities
ess1E_Nodes    = zeros(ess1E_Total,1);
ess2E_Nodes    = zeros(ess2E_Total,1);
tE_Sides       = zeros(tE_Total,1);

% Initialize quantities
ess1E_Values    = zeros(ess1E_Total,1);
ess2E_Values    = zeros(ess2E_Total,1);
txE_Values   = zeros(tE_Total,1);
tyE_Values   = zeros(tE_Total,1);

% Initialize counters
i1 = 1; i2 = 1; i3 = 1;

% Determine number of boundary conditions
for j = 1:1:mesh.NE
    for i = 1:1:mesh.ETYPE*mesh.ETYPE
        % Number of ess1 boundary conditions
        for k = 1:1:boundary.ess1_Number
            if ( mesh.ECON(i,j) == boundary.ess1_Nodes(k,1) )
                % Assign values
                 ess1E_Values(i1,1) = boundary.ess1_Values(k,1);
                 ess1E_Nodes(i1,1)  = i;
                % Update counter
                 i1 = i1 + 1;
            end
        end

        % Number of ess2 boundary conditions
        for k = 1:1:boundary.ess2_Number
            if ( mesh.ECON(i,j) == boundary.ess2_Nodes(k,1) )
                % Assign values
                 ess2E_Values(i2,1) = boundary.ess2_Values(k,1);
                 ess2E_Nodes(i2,1)  = i;
                % Update counter
                 i2 = i2 + 1;
            end
        end

    end

    % Number of traction boundary conditions
    for k = 1:1:boundary.t_Number
        if ( j == boundary.t_Elements(k,1) )
            % Assign values
             txE_Values(i3,1) = boundary.tx_Values(k,1);
             tyE_Values(i3,1) = boundary.ty_Values(k,1);
             tE_Sides(i3,1)   = boundary.t_Sides(k,1);

            % Update counter
             i3 = i3 + 1;
        end
    end
end

%-------------------------------------------------------------------
%
% Store all boundary condition data using a Matlab data structure
%
%-------------------------------------------------------------------

% ess1 boundary conditions
boundaryElem.ess1E_Total   = ess1E_Total;
boundaryElem.ess1E_Pointer = ess1E_Pointer;
boundaryElem.ess1E_Nodes   = ess1E_Nodes;
boundaryElem.ess1E_Values  = ess1E_Values;

% ess2 boundary conditions
boundaryElem.ess2E_Total   = ess2E_Total;
boundaryElem.ess2E_Pointer = ess2E_Pointer;
boundaryElem.ess2E_Nodes   = ess2E_Nodes;
boundaryElem.ess2E_Values  = ess2E_Values;

% Traction boundary conditions
boundaryElem.tE_Total   = tE_Total;
boundaryElem.tE_Pointer = tE_Pointer;
boundaryElem.tE_Sides   = tE_Sides;
boundaryElem.txE_Values = txE_Values;
boundaryElem.tyE_Values = tyE_Values;

% End of subroutine
end
%-------------------------------------------------------------------
%
% End of subroutine
%
 -------------------------------------------------------------------
%
```

(a)

```matlab
%-------------------------------------------------------------------
%
% Apply essential boundary conditions to element equations
% mesh.DFPN =2 (DFPN: Degree of Freedom Per Node)
% KE() : Element Stiffness Matrix
% FE() : Element Force Vector
%
%-------------------------------------------------------------------

% Essential boundary conditions: ess1
val1 = boundaryElem.ess1E_Pointer(n,1);
val2 = boundaryElem.ess1E_Pointer(n+1,1) - 1;

for i = val1:1:val2
    % Temporary constants
    j  = mesh.DFPN*boundaryElem.ess1E_Nodes(i,1) - 1;
    c1 = KE(j,j);
    % Modify element matrix and force vector
    KE(j,:) = zeros(1,mesh.DFPN*mesh.ETYPE*mesh.ETYPE);
    FE      = FE- boundaryElem.ess1E_Values(i,1)*KE(:,j);
    KE(:,j) = zeros(mesh.DFPN*mesh.ETYPE*mesh.ETYPE,1);
    KE(j,j) = c1;
    FE(j,1) = c1*boundaryElem.ess1E_Values(i,1);
end

% Essential boudary conditions: ess2
val1 = boundaryElem.ess2E_Pointer(n,1);
val2 = boundaryElem.ess2E_Pointer(n+1,1) - 1;

for i = val1:1:val2
    % Temporary constants
    j  = mesh.DFPN*boundaryElem.ess2E_Nodes(i,1);
    c1 = KE(j,j);
    % Modify element matrix andforce vector
    KE(j,:) = zeros(1,mesh.DFPN*mesh.ETYPE*mesh.ETYPE);
    FE      = FE- boundaryElem.ess2E_Values(i,1)*KE(:,j);
    KE(:,j) = zeros(mesh.DFPN*mesh.ETYPE*mesh.ETYPE,1);
    KE(j,j) = c1;
    FE(j,1) = c1*boundaryElem.ess2E_Values(i,1);
end
```

(b)

APPENDIX B

The components of the viscoelastic force vector $\tilde{Q}$ on the where right hand side of Eq. (3.36) are given by

$$
\begin{aligned}
{}^1\bar{Q}_i^1 &= \frac{\Delta t_{N-1}}{2} \int_{x_a}^{x_b} \dot{E}\left(\Delta t_{N-1}\right) D_0 \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} dx \Delta_j^{(1)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n} \sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}} {}^1\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1a)} \\
{}^2\bar{Q}_i^1 &= \frac{\Delta t_{N-1}}{4} \int_{x_a}^{x_b} \dot{E}\left(\Delta t_{N-1}\right) D_0 \frac{\partial w_0\left(x, t_{N-1}\right)}{\partial x} \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} dx \Delta_j^{(2)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n} \sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}} {}^2\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1b)} \\
{}^1\bar{Q}_i^2 &= \frac{\Delta t_{N-1}}{2} \int_{x_a}^{x_b} \dot{E}\left(\Delta t_{N-1}\right) D_0 \frac{\partial w_0\left(x, t_N\right)}{\partial x} \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} dx \Delta_j^{(1)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n} \sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}} \frac{\partial w_0\left(x_m, t_N\right)}{\partial x} {}^3\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1c)} \\
{}^2\bar{Q}_i^2 &= \frac{\Delta t_{N-1}}{4} \int_{x_a}^{x_b} \dot{E}\left(\Delta t_{N-1}\right) D_0 \frac{\partial w_0\left(x, t_N\right)}{\partial x} \frac{\partial w_0\left(x, t_{N-1}\right)}{\partial x} \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} \Delta_j^{(2)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n} \sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}} \frac{\partial w_0\left(x_m, t_N\right)}{\partial x} {}^4\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1d)} \\
{}^3\bar{Q}_i^2 &= \frac{\Delta t_{N-1}}{2} \int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right) D_0 \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} dx \Delta_j^{(2)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n} \sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}} {}^5\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1e)}
\end{aligned}
$$

$$
\begin{aligned}
{}^{4}\bar{Q}_i^2 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right)\left(-D_2\ c_2\right)\frac{d\psi_i}{dx}\psi_j dx \Delta_j^{(3)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\, {}^{6}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1f)} \\
{}^{5}\bar{Q}_i^2 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right)\left(D_0\right)\frac{d\psi_i}{dx}\psi_j dx \Delta_j^{(4)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\, {}^{7}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1g)} \\
{}^{1}\bar{Q}_i^3 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right)\left(-D_2\ c_2\right)\psi_i\frac{d\psi_j}{dx} dx \Delta_j^{(2)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\, {}^{8}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1h)} \\
{}^{2}\bar{Q}_i^3 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{E}\left(\Delta t_{N-1}\right)D_6\, c_1^2\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx \Delta_j^{(3)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\, {}^{9}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1i)} \\
{}^{3}\bar{Q}_i^3 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right)D_4\, c_2^2\psi_i\psi_j dx \Delta_j^{(3)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\, {}^{10}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1j)} \\
{}^{4}\bar{Q}_i^3 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{E}\left(\Delta t_{N-1}\right)\left(-D_4\, c_1\right)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx} dx \Delta_j^{(4)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\, {}^{11}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1k)} \\
{}^{5}\bar{Q}_i^3 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right)\left(-D_2\, c_2\right)\psi_i\psi_j dx \Delta_j^{(4)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\, {}^{12}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1l)} \\
{}^{1}\bar{Q}_i^4 &= \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b} \dot{G}\left(\Delta t_{N-1}\right)\left(D_0\right)\frac{d\psi_j}{dx}\psi_i\, dx \Delta_j^{(2)}\left(t_{N-1}\right) \\
&+ \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\, {}^{13}\bar{X}_i^{lm}\left(t_{N-1}\right) \qquad \text{(B.1m)}
\end{aligned}
$$

$$
{}^{2}\bar{Q}_i^{4} = \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b}\dot{E}\left(\Delta t_{N-1}\right)\left(-D_4\,c_1\right)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx\Delta_j^{(3)}\left(t_{N-1}\right) + \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{14}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.1n}
$$

$$
{}^{3}\bar{Q}_i^{4} = \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b}\dot{G}\left(\Delta t_{N-1}\right)\left(-D_2\,c_2\right)\psi_i\psi_j dx\Delta_j^{(3)}\left(t_{N-1}\right) + \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{15}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.1o}
$$

$$
{}^{4}\bar{Q}_i^{4} = \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b}\dot{E}\left(\Delta t_{N-1}\right)\left(D_2\right)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}dx\Delta_j^{(4)}\left(t_{N-1}\right) + \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{16}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.1p}
$$

$$
{}^{5}\bar{Q}_i^{4} = \frac{\Delta t_{N-1}}{2}\int_{x_a}^{x_b}\dot{G}\left(\Delta t_{N-1}\right)\left(D_0\right)\psi_i\psi_j dx\Delta_j^{(4)}\left(t_{N-1}\right) + \sum_{l=1}^{n}\sum_{m=1}^{NGP} e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{17}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.1q}
$$

The history terms ${}^{j}\bar{X}_i^{lm}\left(t_N\right)$ introduced in the above equations can be expressed as

$$
{}^{1}\bar{X}_i^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{E_l}{\tau_l^E}D_0\frac{d\psi_i\left(x_m\right)}{dx}\frac{d\psi_j\left(x_m\right)}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(1)}\left(t_{N-1}\right)+\Delta_j^{(1)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{1}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.2a}
$$

$$
{}^{2}\bar{X}_i^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{4}\frac{E_l}{\tau_l^E}D_0\frac{d\psi_i\left(x_m\right)}{dx}\frac{d\psi_j\left(x_m\right)}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\frac{\partial w_0\left(x_m,t_{N-1}\right)}{\partial x}\Delta_j^{(2)}\left(t_{N-1}\right) + \frac{\partial w_0\left(x_m,t_N\right)}{\partial x}\Delta_j^{(2)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{2}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.2b}
$$

$$
{}^{3}\bar{X}_i^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{E_l}{\tau_l^E}D_0\frac{d\psi_i\left(x_m\right)}{dx}\frac{d\psi_j\left(x_m\right)}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(1)}\left(t_{N-1}\right)+\Delta_j^{(1)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{3}\bar{X}_i^{lm}\left(t_{N-1}\right) \tag{B.2c}
$$

$$
{}^{4}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{4}\frac{E_l}{\tau_l^E}D_0\frac{d\psi_i\left(x_m\right)}{dx}\frac{d\psi_j\left(x_m\right)}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\frac{\partial w_0\left(x_m,t_{N-1}\right)}{\partial x}\Delta_j^{(2)}\left(t_{N-1}\right) + \frac{\partial w_0\left(x_m,t_N\right)}{\partial x}\Delta_j^{(2)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{4}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2d)}
$$

$$
{}^{5}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}D_0\frac{d\psi_i\left(x_m\right)}{dx}\frac{d\psi_j\left(x_m\right)}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(2)}\left(t_{N-1}\right)+\Delta_j^{(2)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{5}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2e)}
$$

$$
{}^{6}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}\left(-D_2\ c_2\right)\frac{d\psi_i}{dx}\psi_j\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(3)}\left(t_{N-1}\right)+\Delta_j^{(3)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{6}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2f)}
$$

$$
{}^{7}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}\left(D_0\right)\frac{d\psi_i}{dx}\psi_j\left(x_m\right)\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(4)}\left(t_{N-1}\right)+\Delta_j^{(4)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{7}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2g)}
$$

$$
{}^{8}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}\left(-D_2\ c_2\right)\psi_i\frac{d\psi_j}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(2)}\left(t_{N-1}\right)+\Delta_j^{(2)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{8}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2h)}
$$

$$
{}^{9}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{E_l}{\tau_l^E}D_6\,c_1^2\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(3)}\left(t_{N-1}\right)+\Delta_j^{(3)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{9}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2i)}
$$

$$
{}^{10}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}D_4\,c_2^2\psi_i\psi_j\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(3)}\left(t_{N-1}\right)+\Delta_j^{(3)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{10}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2j)}
$$

$$
{}^{11}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{E_l}{\tau_l^E}\left(-D_4\,c_1\right)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(4)}\left(t_{N-1}\right)+\Delta_j^{(4)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{11}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2k)}
$$

$$
{}^{12}\bar{X}_{i}^{lm}\left(t_N\right) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}\left(-D_2\,c_2\right)\psi_i\psi_j\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(4)}\left(t_{N-1}\right)+\Delta_j^{(4)}\left(t_N\right)\right)W_m + e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{12}\bar{X}_{i}^{lm}\left(t_{N-1}\right) \quad \text{(B.2l)}
$$

$$
{}^{13}\bar{X}_i^{lm}(t_N) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}(D_0)\frac{d\psi_j}{dx}\psi_i\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(2)}(t_{N-1}) + \Delta_j^{(2)}(t_N)\right)W_m
+ e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{12}\bar{X}_i^{lm}(t_{N-1}) \tag{B.2m}
$$

$$
{}^{14}\bar{X}_i^{lm}(t_N) = -\frac{\Delta t_{N-1}}{2}\frac{E_l}{\tau_l^E}(-D_4\,c_1)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(3)}(t_{N-1}) + \Delta_j^{(3)}(t_N)\right)W_m
+ e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{14}\bar{X}_i^{lm}(t_{N-1}) \tag{B.2n}
$$

$$
{}^{15}\bar{X}_i^{lm}(t_N) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}(-D_2\,c_2)\psi_i\psi_j\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(3)}(t_{N-1}) + \Delta_j^{(3)}(t_N)\right)W_m
+ e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{15}\bar{X}_i^{lm}(t_{N-1}) \tag{B.2o}
$$

$$
{}^{16}\bar{X}_i^{lm}(t_N) = -\frac{\Delta t_{N-1}}{2}\frac{E_l}{\tau_l^E}(D_2)\frac{d\psi_i}{dx}\frac{d\psi_j}{dx}\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\Delta_j^{(4)}(t_{N-1}) + \Delta_j^{(4)}(t_N)\right)W_m
+ e^{-\frac{\Delta t_{N-1}}{\tau_l^E}}\,{}^{16}\bar{X}_i^{lm}(t_{N-1}) \tag{B.2p}
$$

$$
{}^{17}\bar{X}_i^{lm}(t_N) = -\frac{\Delta t_{N-1}}{2}\frac{G_l}{\tau_l^G}(D_0)\psi_i\psi_j\left(e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\Delta_j^{(4)}(t_{N-1}) + \Delta_j^{(4)}(t_N)\right)W_m
+ e^{-\frac{\Delta t_{N-1}}{\tau_l^G}}\,{}^{17}\bar{X}_i^{lm}(t_{N-1}) \tag{B.2q}
$$