ADAPTIVE MESH REFINEMENT FOR A FINITE DIFFERENCE SCHEME USING A QUADTREE DECOMPOSITION APPROACH

A Thesis

by

NANDAGOPALAN AUVIUR SRINIVASA

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

MASTER OF SCIENCE

December 2006

Major Subject: Mechanical Engineering

ADAPTIVE MESH REFINEMENT FOR A FINITE DIFFERENCE SCHEME USING A QUADTREE DECOMPOSITION APPROACH

A Thesis

by

NANDAGOPALAN AUVIUR SRINIVASA

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

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Arun R. Srinivasa
Committee Members,	Chii-Der Suh
	Joe Pasciak
Head of Department,	Dennis O'Neal

December 2006

Major Subject: Mechanical Engineering

ABSTRACT

Adaptive Mesh Refinement for a Finite Difference

Scheme Using a Quadtree Decomposition Approach. (December 2006)

Nandagopalan Auviur Srinivasa, B.Tech,

Jawaharlal Nehru Technological University, Hyderabad, India

Chair of Advisory Committee: Dr. Arun R. Srinivasa

Some numerical simulations of multi-scale physical phenomena consume a significant amount of computational resources, since their domains are discretized on high resolution meshes. An enormous wastage of these resources occurs in refinement of sections of the domain where computation of the solution does not require high resolutions. This problem is effectively addressed by adaptive mesh refinement (AMR), a technique of local refinement of a mesh only in sections where needed, thus allowing concentration of effort where it is required. Sections of the domain needing high resolution are generally determined by means of a criterion which may vary depending on the nature of the problem. Fairly straightforward criteria could include comparing the solution to a threshold or the gradient of a solution, that is, its local rate of change to a threshold. While the former criterion is not particularly rigorous and hardly ever represents a physical phenomenon of interest, it is simple to implement. However, the gradient criterion is not as simple to

implement as a direct comparison of values, but it is still quick and a good indicator of the effectiveness of the AMR technique.

The objective of this thesis is to arrive at an adaptive mesh refinement algorithm for a finite difference scheme using a quadtree decomposition approach. In the AMR algorithm developed, a mesh of increasingly fine resolution permits high resolution computation in sub-domains of interest and low resolution in others. In this thesis work, the gradient of the solution has been considered as the criterion determining the regions of the domain needing refinement. Initial tests using the AMR algorithm demonstrate that the paradigm adopted has considerable promise for a variety of research problems. The tests performed thus far depict that the quantity of computational resources consumed is significantly less while maintaining the quality of the solution. Analysis included comparison of results obtained with analytical solutions for four test problems, as well as a thorough study of a contemporary problem in solid mechanics.

To

My parents and brother

ACKNOWLEDGMENTS

I am grateful to many people who helped me throughout my Master's, not only for their encouragement and feedback, but also for their moral support and their ideas on how to make things work, or work better. An enormous number of people have just listened to me rant, which in itself is reason for me to be grateful. I'd like to express my appreciation to them all, but I'm sure I'll forget a few of them, and they have my apologies in advance.

I shall remain eternally indebted to my advisor, Dr. Arun R. Srinivasa, not only for all of his guidance but also for his patience. Sir, I thank you sincerely for being extremely considerate towards my inadequacies and for making an extra effort to help me with zeroing in on my area of interest. I also extend sincere thanks to the rest of my committee, Dr. Chii-Der Suh and Dr. Joe Pasciak. I thoroughly enjoyed the experience of being their student during my Master's.

Two people deserving special mention are my lab mates, Anshul and Praveen. If any person has made AMR possible, it is Anshul. He has provided me with a lot of insight into the results from AMR. He gave finishing touches to my codes very often to help me interpret my solutions better. Added to that is the moral support he provided me with during my worst moments in Master's. Thank you, Anshul. I want to thank Praveen primarily for pepping up my spirits from time to time with his

antics and amazing sense of humor. Also, for the useful discussions and all-night sessions spent on debugging MEX codes, which was a topic of common interest and operation. Thank you, Praveen, for always being around.

I also wish to thank my family. No words would ever suffice my heartfelt thanks to my parents and grandparents for the confidence, love and trust they have placed in me. I also wish to thank my brother, Krishnan, for achieving big and setting the bar high at various stages of life, which has made me push myself towards higher pursuits, keeping in line with sibling rivalry. Thank you, all of you.

I would like to thank Seemant, Sukesh, Vijay, Saradhi, Amol, Mayuresh, Nipun, Chinmay, Satish, Sourabh and some other friends of the Fall ’04 batch for their support, encouragement, wonderful friendship and sense of camaraderie.

Among my closest circle of friends, those whose support was crucial to this endeavor of mine are Ravi, Suprasanna, Harini, Harish, Aatish and Pavan. Thank you, all of you. It has been a pleasure to be associated with each of you.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF FIGURES

Page

Page

Page

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

1.1 Numerical Analysis: Introduction and Applications

Numerical analysis deals with the study of algorithms for the problems of continuous mathematics. These algorithms are routinely applied to many problems in science and engineering. Important applications include weather forecasting, climate models, the analysis and design of molecules, the design of structures like bridges and airplanes, locating oil reservoirs and the like (Wikipedia, Numerical Analysis). In addition to mathematical axioms, theorems and proofs, numerical analysis uses empirical results of computation runs to probe new methods and analyze problems.

Some of the problems analyzed by numerical analysis can be solved exactly by an algorithm. These methods are called direct methods. Significant examples of such algorithms are the simplex method in linear programming and the Gaussian elimination method for solving systems of linear equations. However, for a majority of the problems, direct methods do not exist. For such cases, iterative methods are usually employed. An iterative method begins with a guess and finds successive approximation that hopefully converges to a solution.

This thesis follows the style and format of the International Journal of Applied Mechanics and Engineering.

The iterative procedures consume a lot of computational resources. As a consequence, efficiency plays a very significant role and a heuristic method may be preferred above a method with a solid theoretic foundation (Wikipedia, Numerical Analysis).

1.2 Adaptive Mesh Refinement

In numerical analysis, continuous problems must sometimes be replaced by a discrete problem whose solution is known to approximate that of the continuous problem. This process is called discretization. The manner of discretization of the domain of interest into a grid of many individual elements is of enormous significance and interest in numerical analysis. In discretizing a domain, consideration must be given to an accurate representation of the domain, point sources, distributed sources with discontinuities and material and geometric discontinuities. Considerations such as the need to discretize the domain into sufficiently small elements so that steep gradients of the solution can be accurately calculated require some engineering judgement, which comes from both a qualitative understanding of the behavior of the solution and an estimate of the computational costs involved in the mesh refinement (Reddy, 2003). The grid generated by discretization can be static, established once and for all at the beginning of the computation, or it may be dynamic, tracking the features of the result as the computation progresses. If the computation has features which one wants to track, then the dynamic gridding scheme must be adopted. This dynamic gridding scheme is called adaptive mesh refinement.

A mesh refinement should meet the following conditions (Reddy, 2003): (1) all previous meshes should be contained in the refined mesh; (2) every point of the domain can be included within an arbitrarily small element at any stage of the mesh refinement; and (3) the same order of approximation for the solution may be retained through all stages of the refinement process. The words ‘coarse’ and ‘fine’ are relative. In any given problem, one begins with a mesh that is believed to be adequate to solve the problem at hand. Then, as a second choice, one selects a mesh that consists of a larger number of elements to solve the problem once again. If there is a significant difference between the two solutions, one sees the benefit of mesh refinement, and further refinements may be warranted. If the difference is negligibly small, further refinements may not be necessary. Such numerical experiments with mesh refinements are not always feasible in practice, mostly because of the computational costs involved (Reddy, 2003). In cases where computational cost is the primary concern, one must depend on one’s judgement concerning what is a reasonably good mesh, which is often dictated by the geometry and qualitative understanding of the variations of the solution and its gradient. One should not be overly concerned with the numerical accuracy of the solution since most practical problems are approximated in their engineering formulations. A decision on when to stop refining a mesh further can be reached by a feel for the relative proportions and directions of various errors introduced into the analysis. Scientific knowledge and experience with a given class of problems is an essential part of any approximate analysis (Reddy, 2003).

Most problems of elasticity usually require solution of certain partial differential equations with given boundary conditions. Very often, these equations can be treated in a rigorous manner only in the case of simple boundaries and domains. This leads one to resort to approximate methods. Numerical solution of partial differential equations (PDE's) involves choosing a discrete domain on which algebraic approximations of the PDE's are solved. Some of the approximation techniques are

- The Finite Difference method
- The Methods of Successive Approximation
- The Relaxation Method

1.3 The Finite Difference Method

The finite-difference method is one of the oldest numerical methods known for solving PDE's. The first application of the finite-difference equations in elasticity is believed to have been made by C. Runge, who used this method in solving torsional problems and reduced the problem to the solution of a system of linear algebraic equations (Timoshenko and Goodier, 1970). Further progress was made by L. F. Richardson who used a certain iterative procedure to solve these algebraic equations, and obtained approximate values of the stresses produced in dams by gravity forces and water pressure (Timoshenko and Goodier, 1970). Another iteration process and the proof of its convergence was given by H. Liebmann (Timoshenko and Goodier, 1970). Subsequently, the finite-difference method was applied successfully in the theory of

plates by H. Marcus before it found wide application in various publications (Timoshenko and Goodier, 1970).

In the finite-difference approach, the continuous problem domain is discretized so that the dependent variables are considered to exist only at discrete points (Anderson et al., 1984). The finite-difference approximations are developed from Taylor series and the derivatives are approximated by differences resulting in an algebraic representation of the partial differential equation (PDE). Thus, a problem involving calculus gets transformed into an algebraic problem. The nature of the resulting algebraic system depends on the character of the problem posed by the original PDE (Anderson et al., 1984). Equilibrium problems usually result in a system of algebraic equations which must be solved simultaneously throughout the problem domain in conjunction with specified boundary values (Anderson et al., 1984).

Three forms of finite-difference equations are very commonly used. The three forms are

- Forward Difference
- Backward Difference
- Central Difference

Let us assume a function f(x) discretized in space as shown in Fig.1.

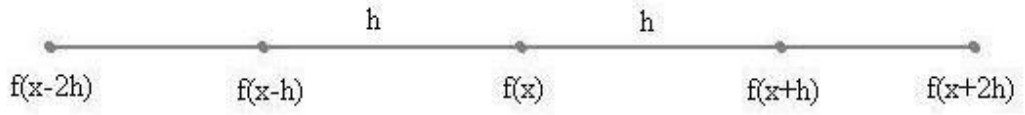


Fig. 1. Spatial discretization of the function f(x).

Forward Difference

A forward difference is defined as the difference of the functional values at the subsequent and present spatial positions. It is denoted generally by an expression of the form

$$\nabla f = f(x+h) - f(x) \tag{1.1}$$

Backward Difference

A backward difference is defined as the difference of the values of a function at the present and previous spatial positions. It is denoted generally by an expression of the form

$$\nabla f = f(x) - f(x-h) \tag{1.2}$$

Central Difference

A central difference is defined as the average of the forward and the backward differences. It can be denoted by the following expression

$$\nabla f = \frac{f(x+h) - f(x-h)}{2} \tag{1.3}$$

Using the three forms of the finite-difference equations, approximate expressions for the differential equations can be obtained in the form of equations of finite difference. Considering a rectangular boundary as shown in Fig.2, and a function w(x,y) of two variables, the first and second derivatives of the function can be approximated as follows:

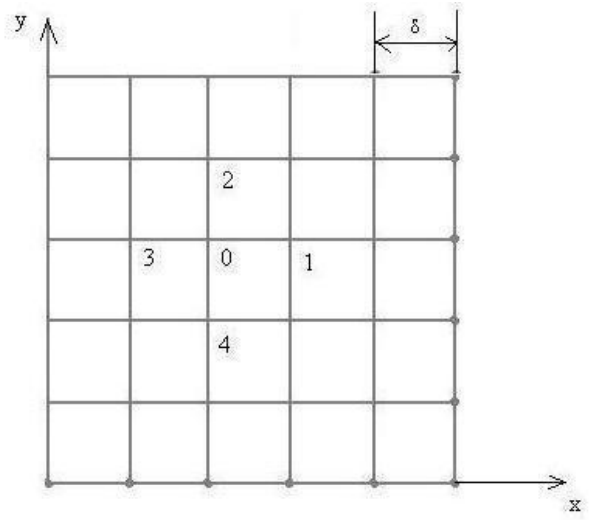


Fig. 2. Computation of derivatives at node 0.

First derivative

Considering central difference for the first derivatives both along x and y directions,

$$\frac{\partial w}{\partial x} \approx \frac{\frac{w_1 - w_0}{\delta} + \frac{w_0 - w_3}{\delta}}{2} = \frac{w_1 - w_3}{2\delta} \tag{1.4}$$

$$\frac{\partial w}{\partial y} \approx \frac{\frac{w_2 - w_0}{\delta} + \frac{w_0 - w_4}{\delta}}{2} = \frac{w_2 - w_4}{2\delta} \tag{1.5}$$

Second derivative

$$\frac{\partial^2 w}{\partial x^2} \approx \frac{w_1 - 2w_0 + w_3}{\delta^2} \tag{1.6}$$

$$\frac{\partial^2 w}{\partial y^2} \approx \frac{w_2 - 2w_0 + w_4}{\delta^2} \tag{1.7}$$

1.4 Discretization: Finite Difference Method vs. Finite Element Method

It is generally convenient to let the mesh increments be constant throughout the computational domain. However, in many instances, this is not possible due to boundaries which do not coincide with the regular mesh or the need to reduce the mesh spacing in certain regions to maintain the desired level of accuracy. It can be cumbersome to apply the finite difference method over non uniform meshes. Generally speaking, the finite-difference method in its basic form is restricted to handling rectangular shapes and simple alterations thereof, while the finite-element method, which is a numerical technique that employs the philosophy of constructing piecewise approximations of solutions to problems described by differential equations (Reddy, 2003), is flexible even with curved geometries. The piecewise approximation of the solution allows the inclusion of any discontinuous data, such as material properties (Reddy 2003). Although finite element methods now seem to dominate the scene, they have not yet made the impact on hyperbolic and other time-dependent problems that they

have achieved with elliptic equations (Mitchell and Griffiths, 1980). Finite difference methods are widely used for mixed or parabolic / hyperbolic PDE's (Anderson et al., 1984) such as those found in compressible high speed / open flows, for time-dependent problems in CFD and for reaction diffusion type of problems. For time dependent problems in CFD, finite-element method needs to solve the mass matrix at each time step whereas only a simple update is required with the finite-difference method. However, finite difference method does not do well with particularly structured boundaries, especially those with variable boundary layers like an aircraft wing.

1.5 State of the Art

The literature for adaptive mesh refinement is extensive, dating back to approximately twenty years and continuing today as a rich field of research in a number of fields like computational fluid dynamics, computational astrophysics, structural dynamics, magnetics, thermal dynamics and microwave theory among others. Several approaches (Min et al., 2006) exist to solve the finite-difference equations on uniform grids in the case of regular domains, as well as in the case of irregular domains (Gibou and Fedkiw, 2005), (Gibou et al., 2002), (Ceniceros et al., 2004), (Johansen and Colella, 1998), (LeVeque and Li, 1994), (Li, 1998), (Mayo, 1984), (McCorquodale et al., 2004), (McKenney and Greengard, 1995). The spacing of the grid points determines the local error and hence the accuracy of the solution. Many physical problems have variations in scale and when solving these problems numerically, high grid resolution in certain

portions is needed to adequately solve the equations. Uniform grids in such situations are inefficient in terms of storage and CPU requirements. Using a highly refined mesh in portions of the domain where high levels of refinement are not needed represents a waste of computational effort. Limitations on computational resources often force a compromise on grid resolution. By locally refining the mesh only where needed, adaptive mesh refinement allows concentration of effort where it is required, allowing better resolution of the problem (Min et al., 2006).

Over the years, several approaches (Berger and Colella, 1989), (Berger and Oliger, 1984) have been developed in order to realize adaptive mesh refinement. Though adaptive mesh strategies are becoming popular (Sussman et al., 1999), (Ham et al., 2002), (Ceniceros and Roma, 2004), implementations based on recursive structures, such as quadtrees / octrees are less common (Min et al., 2006).

CHAPTER II

OVERALL SCOPE AND OBJECTIVES

2.1 Overall Scope

Numerical simulations of physical phenomena require enormous computer resources, both in terms of memory storage and computing time, since their domains are discretized on high resolution meshes. Adaptive mesh refinement is a class of strategies which address this problem by performing high resolution computation only in regions that require it (Neeman, 1996). A couple of the major reasons to avoid uniformly high resolution meshes are:

- Some regions have small gradients, so the solution can be computed with sufficient accuracy on a low resolution grid.
- Some regions have very high gradient, the solution varies very rapidly among the neighboring regions. To predict the solution within a reasonable degree of accuracy, high resolution is required.

An adaptive mesh refinement technique begins with a coarse base grid. As the solution proceeds, the regions requiring more resolution are identified by means of some parameter which characterizes the solution (for e.g. the magnitude of the gradient of the solution). Finer sub-grids are then superimposed only on these regions. Finer and finer

sub-grids get added recursively until either a specified maximum level of refinement is achieved or the parameter characterizing the solution fails to satisfy the criterion for sub-division. One of the critical things the algorithm has to ensure is that abrupt transitions in the mesh are prevented. Thus, in an 'adaptive' mesh refinement technique, the grid spacing is fixed for the base grid alone, and is determined locally for the sub-grids according to the requirements of the problem. Adaptive mesh refinement algorithms are found suitable and useful for purposes like front-tracking. The current work focuses entirely on simply connected domains alone.

2.2 Objectives

The objective of this thesis is to arrive at an adaptive mesh refinement algorithm for a finite difference scheme using a quadtree decomposition approach. This thesis will deal with the solution of the torsion problem in elasticity as a test case. Analytical solutions for the torsion problem are available for a wide variety of cross-sections and in great detail. This makes comparison of the solutions easier, which influences its selection over others. We employ a hierarchical data structure called quadtree for domain discretization. Finite-difference codes require that a regular Cartesian grid be defined over the domain of the region to be modeled. Quadtree grids, which can be classified as a particular type of unstructured grid, consist of congruent but different sizes of square regions, which are constructed by recursive sub-division from an initial square according to prescribed yet flexible criteria. As a result, the grids generated from quadtrees are found to be very suitable for a finite-difference implementation.

CHAPTER III

QUADTREE DECOMPOSITION: AN INTRODUCTION AND A BRIEF EXPLANATION OF THE APPROACH

3.1 Quadtree Data Structure

The term quadtree is used to describe a class of hierarchical data structures whose common property is that they are based on the principle of recursive decomposition of space (Samet, 1990). This data structure was named a 'quadtree' by Raphael Finkel and J.L. Bentley, professors in the Department of Computer Science at the University of Kentucky and Carnegie-Mellon University respectively, in 1974 (Wikipedia, Quadtree). Quadtrees can be differentiated on the following bases (Samet, 1990): (1) the type of data they are used to represent; (2) the principle guiding the decomposition process; (3) the resolution. Currently, they are used for point data, areas, curves, surfaces and volumes. The decomposition may be into equal parts on each level or it may be governed by the input. The resolution of the decomposition may be fixed beforehand, or it may be governed by properties of the input data. One of the widely used quadtree representation of data is concerned with the representation of two-dimensional binary region data. The most studied quadtree approach to region representation, called a region quadtree, is based on the principle of successive sub-division of a bounded image array into four-equal sized quadrants (Samet, 1990). The region quadtree is easily extended to represent three-dimensional binary region data, and the resulting data structure is called a region octree.

3.2 Quadtree Decomposition

Quadtree decomposition is a domain discretization technique based on the principle of region splitting, a procedure that sub-divides a square domain into quadrants, provided the domain meets a certain pre-defined criterion. Each of these quadrants is then considered to be a 'region'. The criterion is then applied to each of these regions. If a region does not meet the criterion, it is not divided any further. However, if a region meets the criterion, it is sub-divided again into four quadrants, and the test criterion is applied to those regions. This process is repeated iteratively until every region in the domain fails to meet the criterion. The result might have regions of several different sizes. Though there are many planar decomposition methods (for e.g. triangular, hexagonal etc.), the use of a quadtree decomposition method into squares can be easily justified (Samet, 1990). Squares are used because the resulting decomposition satisfies the following two properties (Samet, 1990):

1. It yields a partition that is an infinitely repetitive pattern so that it can be used for square domains of any size.

2. It yields a partition that is infinitely decomposable into increasingly finer patterns (i.e. higher resolution).

A quadtree-like decomposition into four equilateral triangles also satisfies these criteria. However, unlike the decomposition into squares, it does not have a uniform orientation (Samet, 1990), i.e. all tiles cannot be mapped into each other by translations of the plane

that do not involve rotation and reflection. In Fig.3, regions 1 and 3 can be mapped onto each other by pure translation whereas regions 1 and 2 can be mapped onto each other only by rotation or reflection.

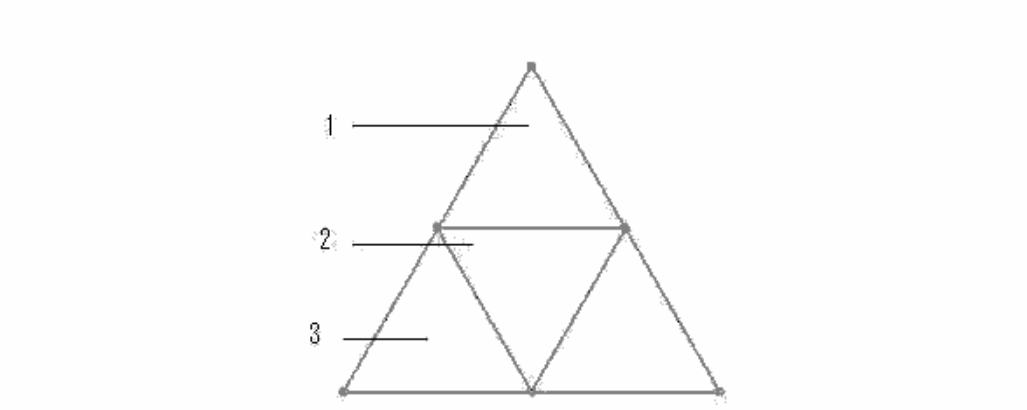


Fig. 3. Quadtree decomposition of an equilateral triangle.

In contrast, decomposition into hexagons has a uniform orientation but it does not satisfy property 2 (Samet, 1990). Fig.4. represents an example of region discretization by quadtree decomposition and the corresponding quadtree representation has been shown in Fig 5. The quadtree grid generation is fast, robust and straight-forward in concept. Mesh information is stored in simple hierarchical data structures and it is easy to obtain high resolution, dynamically adaptive grid. This is ideal when simulating free surface flows containing zones of locally high hydrodynamic gradient such as fronts.

In addition to the above, quadtrees have certain properties which are very suitable and convenient for implementation of finite-difference schemes. Finite-difference codes require that a regular Cartesian grid be defined over the domain of the region to be

modeled. Quadtree grids, which can be classified as a particular type of unstructured grid, consist of congruent but different sizes of square regions, which are constructed by recursive sub-division from an initial square according to prescribed yet flexible criteria. This makes the grids generated from quadtrees very viable for a finite-difference implementation.

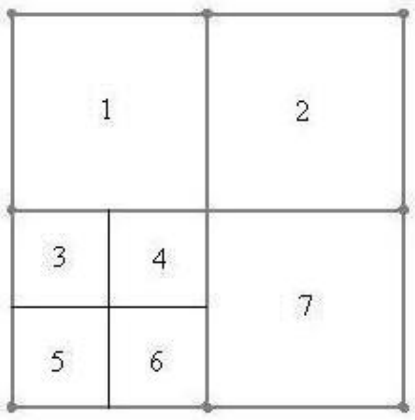


Fig. 4. A region after sub-division by quadtree decomposition.

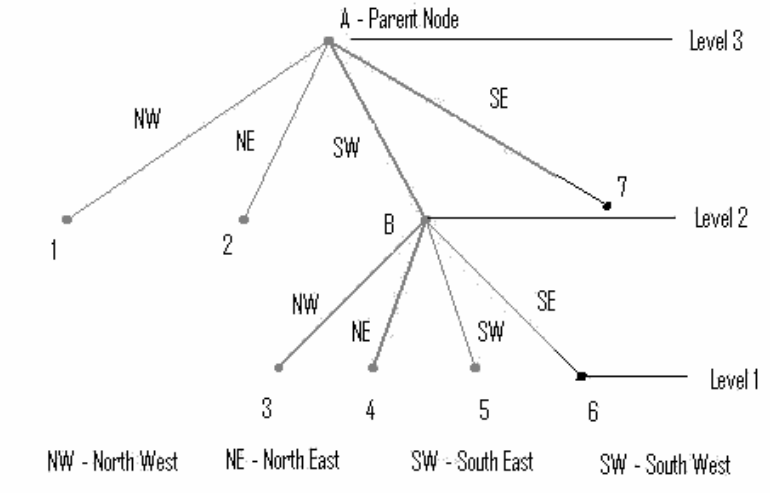


Fig. 5. The quadtree representation of Fig.4.

3.3 Finite Difference Scheme over Non Uniform Grids

As mentioned previously, domain discretization based on quadtree decomposition technique generally yields regions of several different sizes. As a result, the mode of approximation for a partial differential equation based on the finite-difference scheme is quite different when compared to the case of uniform mesh increments. A brief description of the mode of approximation for $\nabla^2\phi$ follows; starting with the computational stencil shown in Fig.6.

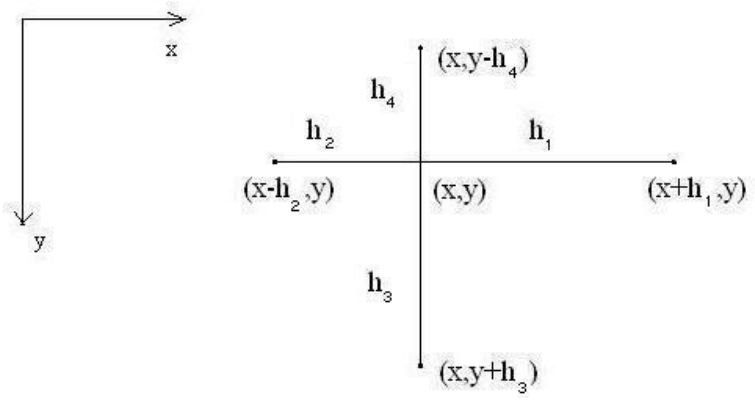


Fig. 6. The computational stencil for the finite difference scheme.

$$\phi(x+h_1,y)=\phi(x,y)+h_1\phi'(x,y)+\frac{h_1^2}{2}\phi''(x,y)+.....H.O.T \tag{3.1}$$

$$\phi(x-h_2,y)=\phi(x,y)-h_2\phi'(x,y)+\frac{h_2^2}{2}\phi''(x,y)-.....H.O.T \tag{3.2}$$

$$\phi(x,y+h_3)=\phi(x,y)+h_3\phi'(x,y)+\frac{h_3^2}{2}\phi''(x,y)+.....H.O.T \tag{3.3}$$

$$\phi(x,y-h_4)=\phi(x,y)-h_4\phi'(x,y)+\frac{h_4^2}{2}\phi''(x,y)-.....H.O.T \tag{3.4}$$

From the four equations, $\nabla^2\phi = -2$ becomes

$$\begin{aligned}\nabla^2\phi &= \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} \\ &= \frac{2}{h_1 h_2}\left[\frac{h_2}{h_1+h_2}\phi(x+h_1,y) - \phi(x,y) + \frac{h_1}{h_1+h_2}\phi(x-h_2,y)\right] \\ &+ \frac{2}{h_3 h_4}\left[\frac{h_4}{h_3+h_4}\phi(x,y+h_3) - \phi(x,y) + \frac{h_4}{h_3+h_4}\phi(x,y-h_4)\right]\end{aligned} \tag{3.5}$$

Consider a part of a domain as shown in Fig.7.

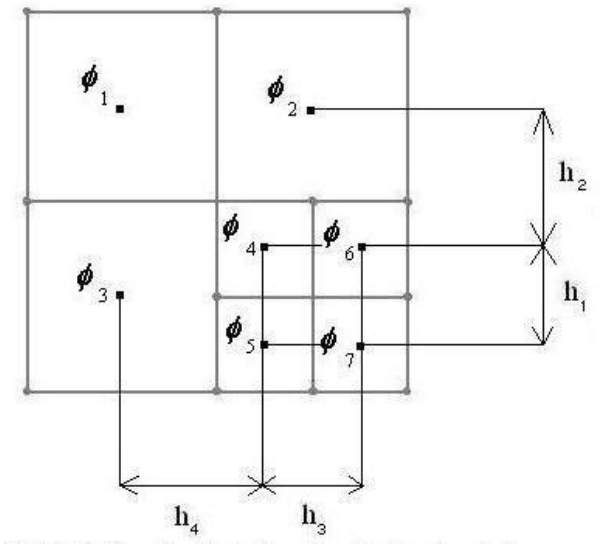


Fig. 7. Configuration representing discretization at node 4.

The discretization of $\nabla^2\phi = -2$ at node 4 is given by the expression

$$\begin{aligned}\nabla^2\phi &= \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} \\ &= \frac{2}{h_3 h_4}\left[\frac{h_4}{h_3+h_4}\phi_6 - \phi_4 + \frac{h_3}{h_3+h_4}\phi_3\right] + \frac{2}{h_1 h_2}\left[\frac{h_2}{h_1+h_2}\phi_5 - \phi_4 + \frac{h_2}{h_1+h_2}\phi_2\right]\end{aligned} \tag{3.6}$$

In addition to the general formulation, there could be specific regions in the domain which would need special considerations. The modes of approximation for some of these are shown below.

Let the domain be defined as shown in the Fig.8.

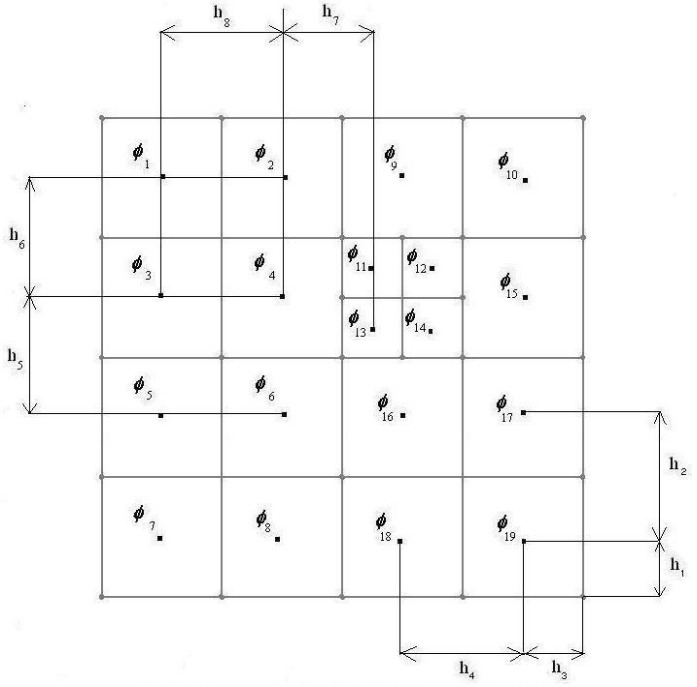


Fig. 8. Configuration illustrating discretization at nodes 4 and 19.

A couple of the regions where the principle of discretization is different when compared to what was previously shown are:

- Region 4

  Region 4 is found to have two neighbors along its right edge. The discretization for $\nabla^2\phi$ at node 4 is given by the expression

$$\begin{aligned}\nabla^2\phi &= \frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2} \\ &= \frac{2}{h_7h_8}\left[\frac{h_8}{2(h_7+h_8)}\phi_{11}+\frac{h_8}{2(h_7+h_8)}\phi_{13}-\phi_4+\frac{h_7}{h_7+h_8}\phi_3\right] \\ &+ \frac{2}{h_5h_6}\left[\frac{h_6}{h_5+h_6}\phi_6-\phi_4+\frac{h_5}{h_5+h_6}\phi_2\right]\end{aligned} \tag{3.7}$$

- Region 19

  Region 19 is a boundary region with no neighbors along its right and bottom edge. The discretization for $\nabla^2\phi$ at node 4 is given by the expression

$$\begin{aligned}\nabla^2\phi &= \frac{\partial^2\phi}{\partial x^2}+\frac{\partial^2\phi}{\partial y^2} \\ &= \frac{2}{h_3h_4}\left[-\phi_{19}+\frac{h_3}{h_3+h_4}\phi_{18}\right] + \frac{2}{h_1h_2}\left[-\phi_{19}+\frac{h_2}{h_1+h_2}\phi_{17}\right]\end{aligned} \tag{3.8}$$

3.4 A Brief Note on Center Based and Node Based Discretization

There are generally two standard choices for sampling the solution of a partial differential equation: sampling at the nodes or at the center of each region. A node-based discretization generally leads to a non-symmetric system since the relation between neighbors is non-reflective, whereas a center-based discretization often leads to a symmetric linear system (Min et al., 2006). This is explained by means of Fig.9 and Fig.10.

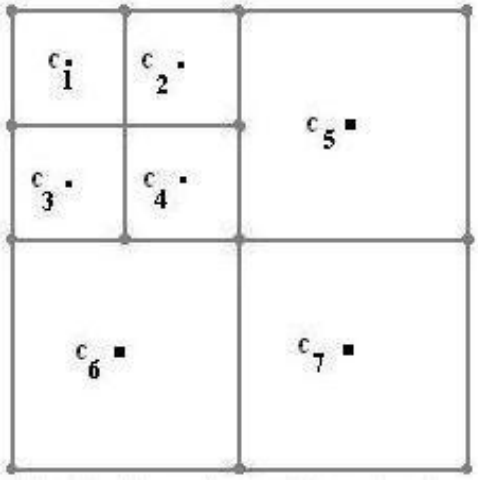


Fig. 9. Center based discretization.

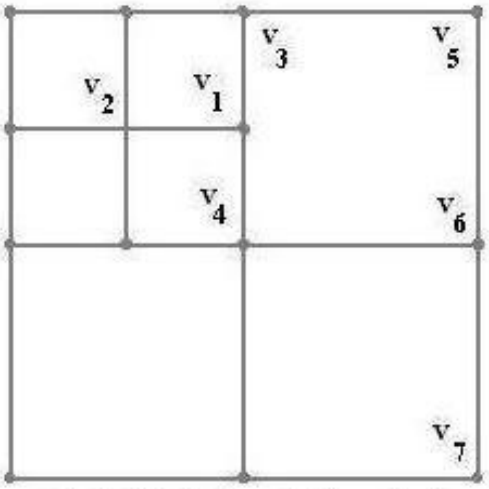


Fig. 10. Node based discretization**.**

Center based discretization is considered in the case depicted in Fig.9. The discretization at c_4 can be defined in terms of c_2, c_3, c_5 and c_6. Similarly, the discretizations at c_2, c_3, c_5 and c_6 all include c_4. Node based discretization is depicted in Fig.10. Geometrically, the discretization at v_1 can be defined in terms of v_2, v_3, v_4, v_5 and v_6. The discretization at

v_6 can be defined in terms of v_4, v_5 and v_7. As a consequence, the equation for v_1 involves v_6, but the discretization for v_6 does not involve v_1, and thus produces non-symmetric discretizations. The center-based discretization has been preferred in this study.

3.5 MATLAB: Introduction and Applications

MATLAB (an acronym for Matrix Laboratory) is a special-purpose computer program optimized to perform engineering and scientific calculations. It started out as a program designed to perform matrix mathematics (Chapman, 2000), and has grown over the years into a flexible computing system capable of solving a wide variety of technical problems. The MATLAB language provides a very extensive library of predefined functions to make technical programming tasks easier and more efficient. This exhaustive library makes MATLAB a preferred programming tool over other programming languages like FORTRAN and C (Chapman, 2000). For computations involving arithmetic mean, standard deviation and median, most languages would require the user to write subroutines and functions. However, MATLAB has these and hundreds of such other functions built right into its package. Furthermore, MATLAB has an extensive set of special-purpose toolboxes available to solve complex problems in specific areas. Some of these toolboxes serve the fields of Signal Processing, Control Systems, Communications, Image Processing and Neural Networks, among many others (Chapman, 2000).

The image processing tool box of MATLAB has an in-built function called ‘qtdecomp’ which performs the quadtree decomposition of images based on a certain predefined criteria. This function works by dividing a square image into quadrants. Each of these quadrants is referred to as a block, and each block is then tested to see if it meets some criterion of homogeneity. The blocks are further sub-divided based on whether or not the criterion is satisfied or not. This process is iteratively repeated until each block fails to meet or meets the criterion depending upon how the criterion is defined.

The present work incorporates certain features of this pre-defined function in MATLAB which performs quadtree decomposition to build its own version of a function for accomplishing quadtree decomposition. Though the underlying concepts of the functions are the same, the present work is not involved with image processing. Consequently, the function developed in the present work is not involved with reading of images or operations based on pixel intensity values unlike the predefined function in MATLAB. Also, it is observed that a vast amount of computational resources is required for the task at hand which involves a huge amount of loop-based operations. As a result, it is desired to interface MATLAB with a low-level programming language to speed up the task and reduce the amount of resources required. In this work, MATLAB interacts with C to accomplish the same through a facility called MEX which is described next. Through this interfacing, a significant amount of reduction in the computational resources has been noted. To cite an example, a code which takes about 11 hours to finish a certain

task when coded in MATLAB alone, requires about 33 seconds when the MATLAB code is interfaced with MEX, to accomplish the same.

3.6 MEX: Introduction and Applications

MEX stands for MATLAB executable. Though MATLAB provides a complete, self-contained environment for programming and manipulating data, it is advantageous if it can interact with programs external to its environment. To be able to support these external interfaces, MATLAB provides an Application Program Interface (API). The functions supported by the API include (MATLAB API Guide, 1998):

- Calling C or FORTRAN programs from MATLAB.
- Importing and exporting data to and from the MATLAB environment.
- Establishing client / server relationships between MATLAB and other software programs.

MEX-files are MATLAB's external interface which allows you to call C or FORTRAN subroutines from MATLAB. Further, certain functions which go by the name of MEX functions can be called from these invoked C or FORTRAN subroutines. These functions perform functions on data similar to the way MATLAB does. MEX-files are dynamically linked, and can be built either from the MATLAB command line or can be compiled the way C or FORTRAN programs are compiled, depending on the complexity of the function to be performed, and then called from within a MATLAB session. The

MEX-files themselves are invoked exactly the way the M-files are invoked, by referencing the name without the suffix (MATLAB API Guide, 1998).

Some of the common applications are:

- Large pre-existing C and FORTRAN programs can be called from MATLAB without having to be re-written as M-files.
- Bottleneck computations (typically loop-based code) that do not run fast enough in MATLAB can be recoded in C or FORTRAN for efficiency.

The MATLAB language works with only a single object type: the MATLAB array. All MATLAB variables, including scalars, vectors, matrices, cell arrays, structures and objects are stored as MATLAB arrays. The MATLAB array is declared to be of type 'mxArray' in C. The 'mxArray' structure, among other useful details, contains:

- The variable's name
- The dimensions of the array
- The type (array, cell, structure)
- If numeric, whether is variable is real or complex
- If sparse, its indices and nonzero maximum elements
- If a structure or object, the number of fields and field names

The mx and mex Prefixes

A large set of functions and subroutines are provided by the MATLAB Application Program Interface for the purpose of manipulating a MATLAB array data structure. These functions and subroutines begin either with the prefix ‘mx’ or ‘mex’.

- *‘mx’ Routines*

  The array access and creation routines for manipulating MATLAB arrays always start with the prefix ‘mx’. For example, ‘mxGetPr’ retrieves the pointer to the real data inside the array.

- *‘mex’ Routines*

  Routines that perform operations back in the MATLAB environment begin with the prefix ‘mex’. Useful ones include

  - ‘mexPrintf’, which prints a string at the MATLAB prompt
  - ‘mexErrMsg’, which prints a string and then kills the Mex-file call

The Components of a MEX-File

The source code for a MEX-file consists of two distinct parts:

- Computational Routine
- Gateway Routine

Computational Routine

A computational routine comprises the code for performing the computations that need to be implemented in the MEX file. These computations can be numerical computations as well as inputting and outputting data.

Gateway Routine

A gateway routine interfaces the computational routine with MATLAB by the entry point ‘mexFunction’ and its arguments. The arguments are

- plhs, an array of left-hand output arguments
- nlhs, the number of left-hand output arguments
- prhs, an array of right-hand input arguments
- nrhs, the number of right hand input arguments

The data contained in the ‘mxArray’ structure can be accessed in the Gateway routine and can be manipulated in the computational routine.

MEX – Grabbing workspace variables

It is critical for the MEX-file to be able to access the variables stored in the MATLAB workspace, which have been computed by some sequence of operations in MATLAB, when the MATLAB prompt encounters the MEX command. For this, MEX provides a very convenient feature wherein a variable defined in the MATLAB workspace can be directly read by the MEX file without passing the variable as an input argument. To be able to do this, the computational function and the gateway function are mingled.

However, the function is still defined as ‘mexFunction’ following which the ‘mxArray’ can be grabbed from the MATLAB’s workspace just by knowing its name. The general syntax for grabbing a variable ‘A’ is:

A_ptr = mexGetVariable (“A”, “base”);

On encountering the argument “base”, the mxArray ‘A’ is looked for in the current MATLAB workspace.

A potential area of confusion

Sometimes, a MEX file and an M-file with the same name can exist in the same folder. This is a potentially confusing thing. In such a situation, when the MATLAB prompt encounters the function call, it calls the MEX-file. However, this rule applies only if the files are in the same directory. Otherwise, MATLAB stops looking in its search path as soon as it finds a file with the correct name with either a .mex or an .m suffix.

Since MEX-files make a significant difference to the code in terms of the required computational resources, it is easy to feel fascinated and make extensive use of the MEX feature. However, it is important to understand that MEX-files are not appropriate for all applications. MATLAB is a high-productivity system whose specialty lies in eliminating time-consuming, low-level programming in compiled languages like FORTRAN or C. Consequently, in this thesis work, most of the programming has been done using MATLAB while the MEX facility has been utilized in those portions of the work where loop-based code is predominant.

3.7 A Quintessential Problem in Solid Mechanics

Considering the goals of this thesis work, it was important to develop a mesh refinement algorithm, be able to test the algorithm developed for a variety of cases, and determine its effectiveness by comparing the results from the simulations with the analytical solutions. To accomplish this, we needed to select a physical problem for which

- Analytical solutions are available for a variety of cross-sections
- The governing differential equation is easy to model using finite difference method
- The boundary conditions are easy to implement

Keeping the above requirements in mind, the torsion problem in elasticity, which is a quintessential problem in solid mechanics, was identified as one which would serve the purpose well. The next chapter details how each of the needs are addressed by the torsion problem. Thus, the torsion problem in elasticity has been implemented as a test case in this thesis work.

CHAPTER IV

THE TORSION PROBLEM

4.1 Introduction to the Torsion Problem in Elasticity

The torsion problem in elasticity deals with cylinders of any cross-section being twisted by couples applied at the ends. The shearing stress (Chou and Pagano, 1992) in a cylinder of circular cross-section under torsion is given by the elementary torsion formula in strength of materials. According to the elementary theory of twist of circular shafts, the shearing stress at any point of the cross-section (Fig.11) is perpendicular to the radius 'r'; proportional to the length 'r' and to the angle of twist per unit length 'α' of the shaft (Timoshenko and Goodier, 1970):

$$\tau = \mu\alpha r \tag{4.1}$$

where μ is the modulus of rigidity.

Resolving this stress into two components parallel to the x and y axes, we obtain

$$\sigma_{32} = \mu\alpha x \tag{4.2}$$

$$\sigma_{31} = -\mu\alpha y \tag{4.3}$$

The elementary theory also assumes that

$$\sigma_{11} = \sigma_{22} = \sigma_{33} = \sigma_{12} = 0 \tag{4.4}$$

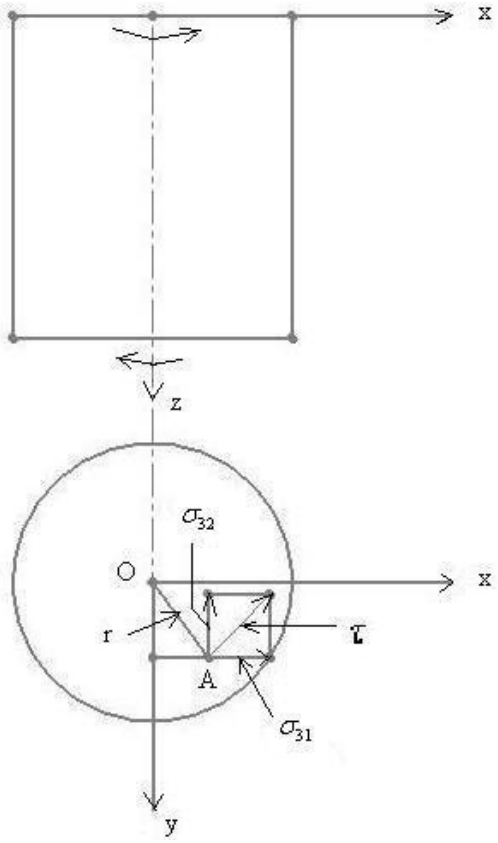


Fig. 11. Twist of a shaft of circular cross-section.

It can be shown that this elementary solution is the exact solution under certain conditions. To do this, it is important to give a brief explanation of the equations of equilibrium, the equations of compatibility and the boundary conditions.

Equations of equilibrium

To determine the equations of equilibrium, let us consider a small rectangular parallelepiped with the sides $\delta x, \delta y$ and δz as shown in Fig.12. In calculating the forces acting on the element, we consider the sides as very small, and the force is obtained by

multiplying the stress at the centroid of a side by the area of the side. Also, the body force acting on the element, which can be neglected as a small quantity of higher order in certain cases, must in this case be considered as significant since it is of the same order of magnitude as the terms due to the variations of the stress components. Let X, Y, Z denote the components of this force per unit volume of the element.

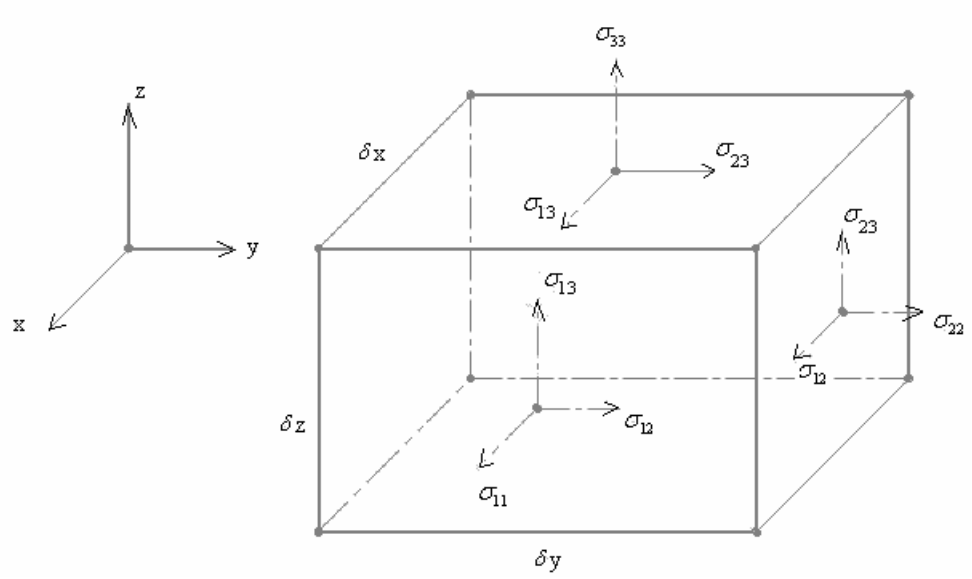


Fig. 12. Rectangular parallelepiped of dimensions $\delta x, \delta y$ and δz.

Then, in Cartesian coordinates, the equations of equilibrium of elasticity are given by the equations:

$$\begin{aligned} \frac{\partial \sigma_{11}}{\partial x}+\frac{\partial \sigma_{12}}{\partial y}+\frac{\partial \sigma_{13}}{\partial z}+X=0 \\ \frac{\partial \sigma_{21}}{\partial x}+\frac{\partial \sigma_{22}}{\partial y}+\frac{\partial \sigma_{23}}{\partial z}+Y=0 \\ \frac{\partial \sigma_{31}}{\partial x}+\frac{\partial \sigma_{32}}{\partial y}+\frac{\partial \sigma_{33}}{\partial z}+Z=0 \end{aligned} \tag{4.5}$$

The equations of equilibrium must be satisfied at all points throughout the volume of the body. The stresses vary over the volume of the body, but at the surface they must be such as to be in equilibrium with the external forces on the surface of the body. The conditions of equilibrium at the surface can be obtained by considering a tetrahedron OBCD so that the side BCD coincides with the surface of the body as shown in Fig.13.

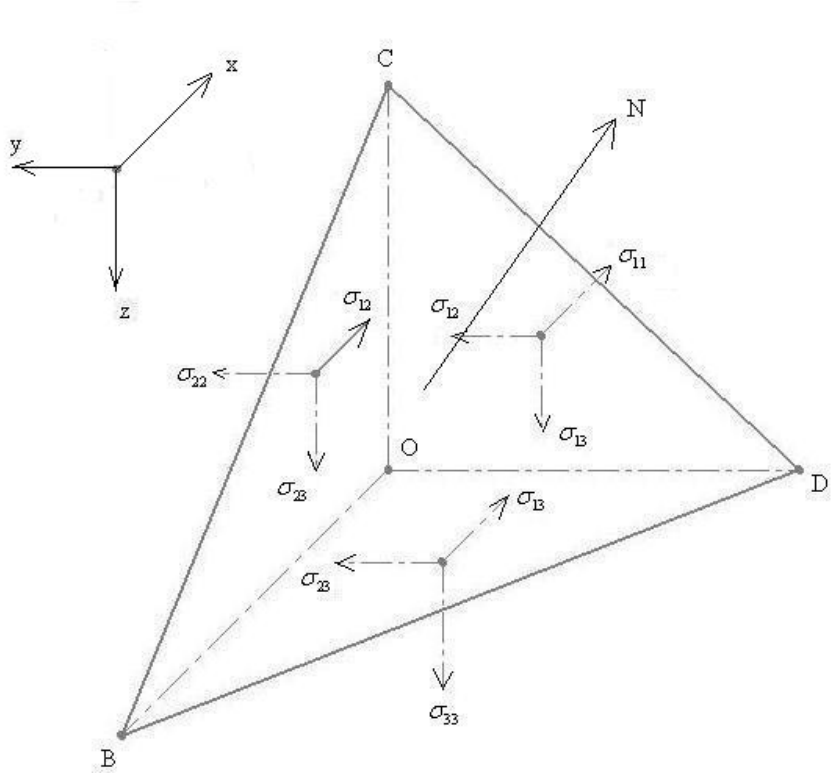


Fig. 13. BCD coincides with the surface and OBCD forms a tetrahedron.

Denoting the components of the surface forces per unit area, the equations become:

$$\begin{aligned} \bar{X} &= \sigma_{11}l + \sigma_{12}m + \sigma_{13}n \\ \bar{Y} &= \sigma_{21}l + \sigma_{22}m + \sigma_{23}n \\ \bar{Z} &= \sigma_{31}l + \sigma_{32}m + \sigma_{33}n \end{aligned} \tag{4.6}$$

in which l, m, n are the direction cosines of the outward normal to the surface of the body at the point under consideration. To determine the state of stress in a body subject to the action of given forces, it is necessary to solve the equilibrium equations, and the solution must be such as to satisfy the boundary conditions. These equations, containing six components of the stress, are not sufficient for the determination of these components. The problem is a statically determinate one, and in order to obtain the solution, the equations of compatibility are considered.

Equations of compatibility

In Cartesian coordinates, the compatibility conditions are given by the following equations:

$$\begin{aligned}
&\frac{\partial^2\varepsilon_{11}}{\partial y^2}+\frac{\partial^2\varepsilon_{22}}{\partial x^2}=2\frac{\partial^2\varepsilon_{12}}{\partial x\partial y} \qquad 2\frac{\partial^2\varepsilon_{11}}{\partial y\partial z}=\frac{\partial}{\partial x}\left(-\frac{\partial\varepsilon_{23}}{\partial x}+\frac{\partial\varepsilon_{31}}{\partial y}+\frac{\partial\varepsilon_{12}}{\partial z}\right)\\
&\frac{\partial^2\varepsilon_{22}}{\partial z^2}+\frac{\partial^2\varepsilon_{33}}{\partial y^2}=2\frac{\partial^2\varepsilon_{23}}{\partial y\partial z} \qquad 2\frac{\partial^2\varepsilon_{22}}{\partial x\partial z}=\frac{\partial}{\partial y}\left(\frac{\partial\varepsilon_{23}}{\partial x}-\frac{\partial\varepsilon_{31}}{\partial y}+\frac{\partial\varepsilon_{12}}{\partial z}\right)\\
&\frac{\partial^2\varepsilon_{33}}{\partial x^2}+\frac{\partial^2\varepsilon_{11}}{\partial z^2}=2\frac{\partial^2\varepsilon_{31}}{\partial x\partial z} \qquad 2\frac{\partial^2\varepsilon_{33}}{\partial x\partial y}=\frac{\partial}{\partial z}\left(\frac{\partial\varepsilon_{23}}{\partial x}+\frac{\partial\varepsilon_{31}}{\partial y}-\frac{\partial\varepsilon_{12}}{\partial z}\right)
\end{aligned} \tag{4.7}$$

Since the stress components are all either linear functions of the coordinates or zero, the equations of compatibility are satisfied, and it is only necessary to consider the equations of equilibrium and the boundary conditions. Substituting the expressions for the circular shaft into the equilibrium equations, it is found that the equations are satisfied provided there are no body forces. The lateral surface of the shaft is free from forces, and the boundary conditions are satisfied in the case of a cylinder of circular cross-section. Since

the stresses satisfy the governing equations of elasticity as well as the boundary conditions, therefore they represent the exact solution for a circular cylinder. The behavior of a circular cylinder under torsion is such that all cross-sections normal to the axis remain plane and the radii remain straight (Chou and Pagano, 1992).

For cylinders of other cross-sections subjected to torque, however, this condition does not prevail and warping occurs.

4.2 Torsion of Cylinders of Non Circular Cross Section

The exact solution for the problem of torsion of shafts of non-circular cross-section was first formulated by Saint-Venant by using the semi-inverse method (Chou and Pagano, 1992). In the beginning, he made certain assumptions as to the deformation of the twisted bar and showed that with these assumptions he could satisfy Eqs (4.5) and (4.6) (Timoshenko and Goodier, 1970). Then, from the uniqueness of solutions of the elasticity equations, it was shown that the assumptions made by him are correct and the solution obtained is the exact solution of the torsion problem, provided that the torques on the ends are applied as shear stresses in exactly the manner required by the solution.

Guided by the deformations which occur in a circular cylinder, Saint-Venant assumed that the displacements which occur in a twisted non-circular cylinder are of the following nature (Chou and Pagano, 1992), (Slaughter, 2002):

- Each cross-section's projection onto the X_1-X_2 plane rotates, but remains undistorted.
- The amount each cross-section rotates is proportional to its distance from the end of the cylinder; i.e. the twist of each cross-section is $\phi = \alpha X_3$, where' α ' is the twist per unit length.
- Each cross-section's out-of-plane distortion is the same, and the magnitude of the distortion is proportional to the twist per unit length' α '.

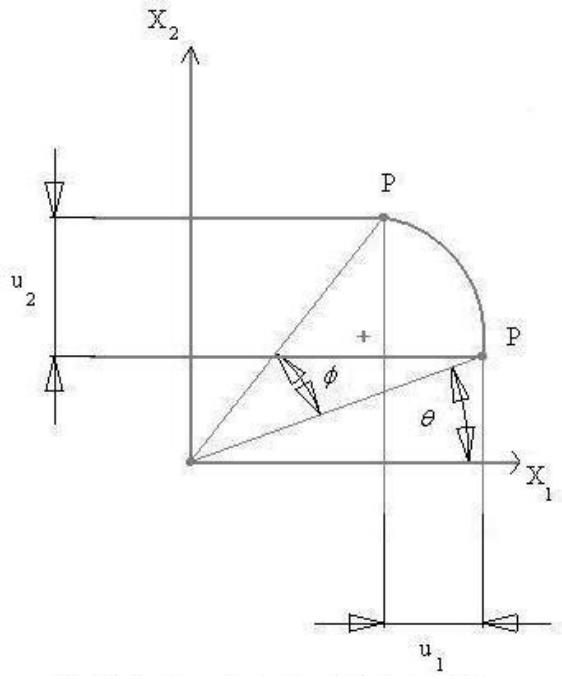


Fig. 14. In-plane displacement during torsion.

From the first two assumptions, the in-plane components of the displacement can be written as:

$$u_1 = r\cos(\phi+\theta) - r\cos\theta = X_1(\cos\phi - 1) - X_2\sin\phi$$
$$u_2 = r\sin(\phi+\theta) - r\sin\theta = X_1\sin\phi + X_2(\cos\phi - 1) \tag{4.8}$$

The out-of-plane displacement of each cross-section is given by the expression

$$u_3 = \alpha\psi(X_1, X_2) \tag{4.9}$$

$\psi(X_1, X_2)$ is called the 'warping function' and describes the out-of-plane distortion of each cross-section. If $\phi = \alpha X_3 << 1$, then the displacement components can be approximated as

$$u_1 \approx -\alpha X_2 X_3 \qquad u_2 \approx \alpha X_1 X_3 \qquad u_3 \approx \alpha\psi(X_1, X_2) \tag{4.10}$$

The components of strain can be obtained from the general expressions for the displacements:

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{4.11}$$

Hence,

$$\varepsilon_{11} = \varepsilon_{22} = \varepsilon_{33} = \varepsilon_{12} = 0 \tag{4.12}$$

$$\varepsilon_{31} = \frac{\alpha}{2}(\psi_{,1} - X_2) \qquad \varepsilon_{32} = \frac{\alpha}{2}(\psi_{,2} + X_1) \tag{4.13}$$

Further, the expressions for the components of stress can be obtained from the constitutive equations of elasticity given by Eq.(4.14).

$$\sigma_{ij} = 2\mu\varepsilon_{ij} + \lambda\varepsilon_{kk}\delta_{ij} \tag{4.14}$$

Hence,

$$\sigma_{11} = \sigma_{22} = \sigma_{33} = \sigma_{12} = 0 \tag{4.15}$$

$$\sigma_{31} = \mu\alpha(\psi_{,1} - X_2) \qquad \sigma_{32} = \mu\alpha(\psi_{,2} + X_1) \tag{4.16}$$

The warping function must now be determined in such a way that Eq.(20) is satisfied. Substituting Eqs (4.15) and (4.16) into Eq.(4.5) and neglecting the body forces, we find that the warping function must satisfy the equation

$$\frac{\partial^2\psi}{\partial X_1^{\,2}} + \frac{\partial^2\psi}{\partial X_2^{\,2}} = 0 \tag{4.17}$$

Considering Eq.(21), for the lateral surface of the bar, which is free from external forces and has normals perpendiculars to the z axes, we have X = Y = Z = 0 and cos (Nz) = n =0. The first two of Eq.(4.6) are identically satisfied and the third gives

$$\sigma_{31}l + \sigma_{32}m = 0 \tag{4.18}$$

This means that the resultant shearing stress at the boundary is directed along the tangent to the boundary.

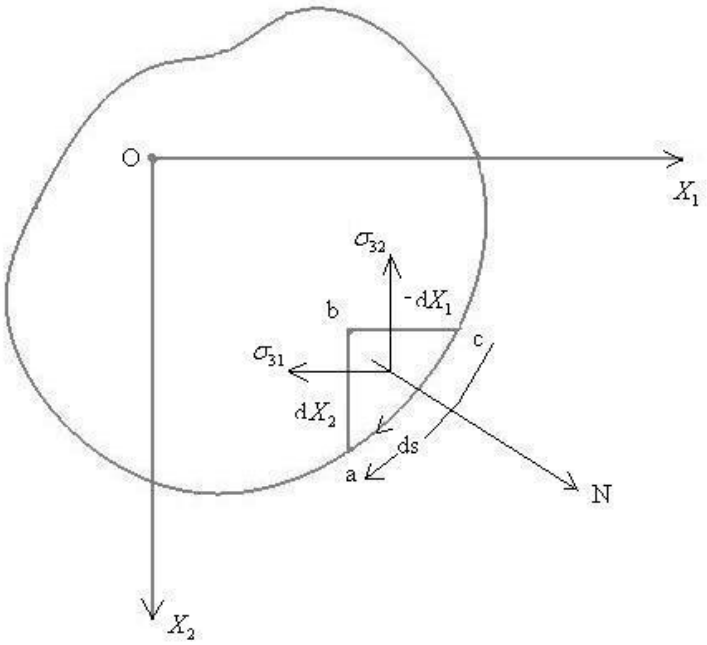


Fig. 15. Infinitesimal element 'abc' at the boundary.

Considering an infinitesimal element 'abc' at the boundary as shown in Fig.15 and assuming that's' is increasing in the direction from 'c' to 'a', we have

$$l = \cos NX_1 = \frac{dX_2}{ds} \qquad m = \cos NX_2 = -\frac{dX_1}{ds} \tag{4.19}$$

and Eq (4.18) becomes

$$\left(\frac{\partial \psi}{\partial X_1} - X_2\right)\frac{dX_2}{ds} - \left(\frac{\partial \psi}{\partial X_2} + X_1\right)\frac{dX_1}{ds} = 0 \tag{4.20}$$

Thus, each problem of torsion is reduced to the problem of finding a function satisfying Eqs (4.17) and (4.20).

An alternative procedure has the advantage of leading to a simpler boundary condition. Assuming the body force field to be negligible, the equilibrium equations reduce to the form,

$$\sigma_{31,1} + \sigma_{32,2} = 0 \tag{4.21}$$

Accordingly, the Prandtl Stress Function can be defined by the relations

$$\sigma_{31} = \phi_{,2} \qquad \sigma_{32} = -\phi_{,1} \tag{4.22}$$

Comparing Eqs (4.16) and (4.22), it is evident that

$$\psi_{,1} = \frac{\phi_{,2}}{\mu\alpha} + X_2 \qquad \psi_{,2} = -\frac{\phi_{,1}}{\mu\alpha} - X_1 \tag{4.23}$$

Using Eq.(4.22) and the Beltrami Mitchell equations, the compatibility conditions for 'ϕ'are obtained as

$$\begin{aligned} \phi_{,211} + \phi_{,222} &= \left(\nabla^2\phi\right)_{,2} = 0 \\ \phi_{,111} + \phi_{,122} &= \left(\nabla^2\phi\right)_{,1} = 0 \end{aligned} \tag{4.24}$$

Eliminating Ψ by differentiating the first part of Eq.(4.23) with respect to 'X_2' and the second part of the same with respect to 'X_1', the governing partial differential equation in the case of torsion of non-circular cylinders is obtained as

$$\nabla^2\phi = -2\mu\alpha \tag{4.25}$$

On introducing Eq.(4.22), Eq.(4.17) becomes

$$\frac{\partial \phi}{\partial X_2}\frac{dX_2}{ds}+\frac{\partial \phi}{\partial X_1}\frac{dX_1}{ds}=\frac{d\phi}{ds}=0 \tag{4.26}$$

This shows that the Prandtl stress function must be constant along the boundary of the cross-section. In the case of singly connected domains, this constant can be chosen arbitrarily. The value of 'ϕ 'at the boundary has been considered to be zero in this thesis work. Thus, the determination of the stress distribution over a cross-section of a twisted bar involves finding the function that satisfies Eq.(4.25) and is zero at the boundary.

Also, considering the conditions at the ends of the twisted bar, the normals at the end cross-sections are parallel to the z axis. Hence, $l=m=0, n=\pm 1$ and Eq.(4.5) become

$$\overline{X}=\pm\sigma_{13} \qquad \overline{Y}=\pm\sigma_{23} \tag{4.27}$$

in which the +ve sign should be taken for the end of the bar for which the outward normal has the direction of the +ve z axis. It is observed that the shearing forces over the ends are distributed in the same manner as the shearing stresses over the cross-sections of the bar. It can be proved that these forces produce a torque.

Substituting in Eq.(4.27) from Eq.(4.22), and observing that 'ϕ' at the boundary is zero, we obtain

$$\iint \bar{X}\, dX_1\, dX_2 = \iint \sigma_{13} dX_1\, dX_2 = \iint \frac{\partial \phi}{\partial X_2} dX_1\, dX_2 = \int dX_1 \int \frac{\partial \phi}{\partial X_2} dX_2 = 0$$
$$\iint \bar{Y}\, dX_1\, dX_2 = \iint \sigma_{23} dX_1\, dX_2 = -\iint \frac{\partial \phi}{\partial x} dX_1\, dX_2 = -\int dX_2 \int \frac{\partial \phi}{\partial x} dX_1 = 0 \qquad (4.28)$$

Thus, the resultant of the forces distributed over the ends of the bar is zero, and these forces represent a couple, the magnitude of which is

$$T = \iint (\bar{Y} X_1 - \bar{X} X_2)\, dX_1\, dX_2 = -\iint \frac{\partial \phi}{\partial X_1} X_1\, dX_1\, dX_2 \; - \iint \frac{\partial \phi}{\partial X_2} X_2\, dX_1\, dX_2 \qquad (4.29)$$

Integrating this by parts, and observing that ' ϕ '= 0 at the boundary, we obtain

$$T = 2 \iint \phi\, dX_1\, dX_2 \qquad (4.30)$$

Each of the integrals in the last part of Eq.(4.29) contributes one-half of the torque. Thus, half of the torque is due to the stress component ' σ_{13} 'and the other half due to ' σ_{23} '. Further, the twist per unit length ' α ' is related to the applied torque by the relation

$$T = D\alpha \qquad (4.31)$$

where D is defined as the torsional rigidity and given by the expression

$$D = 2\mu \int_C \phi\, dA \qquad (4.32)$$

The torsional rigidity provides a measure of the rigidity of a beam subjected to torsion (Sokolnikoff, 1983). It depends on the modulus of rigidity and the shape of the cross-section of the beam alone.

From the general theory of torsion, Eq.(4.26) is valid even for multiply connected cross-sections. However, unlike in the case of simply-connected domains, the constants cannot be chosen arbitrarily. Some of the relevant details are explained in the next section.

4.3 Multiply Connected Domains

Cross sections with holes in them require special treatment since it cannot be assumed that the Prandtl stress function vanishes at all the boundaries. In other words, the values of the constants associated with the boundaries need to be determined. The reason for this is that the differential equations for the Prandtl stress function is derived using the compatibility equations, which assume that the region is simply connected.

Consider a hollow shaft whose cross-section has two or more boundaries. The simplest example of this kind is a hollow shaft with an inner boundary coinciding with one of the stress lines of the solid shaft, having the same boundary as the outer boundary of the hollow shaft.

Consider an elliptic cross-section as shown in Fig.16. The stress function for the solid shaft is

$$\phi = \frac{a^2 b^2 F}{2(a^2 + b^2)} \left(\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 \right) \tag{4.33}$$

Eq.(49) represents an ellipse that is geometrically similar to the outer boundary of the cross-section.

$$\frac{x^2}{(ak)^2}+\frac{y^2}{(bk)^2}=1 \tag{4.34}$$

Along this ellipse, the stress function denoted by Eq.(4.33) remains constant, and hence, for ‘k’ less than unity, the ellipse is a stress line for the solid elliptic shaft. The shearing stress at any point of this line is in the direction of the tangent to the line.

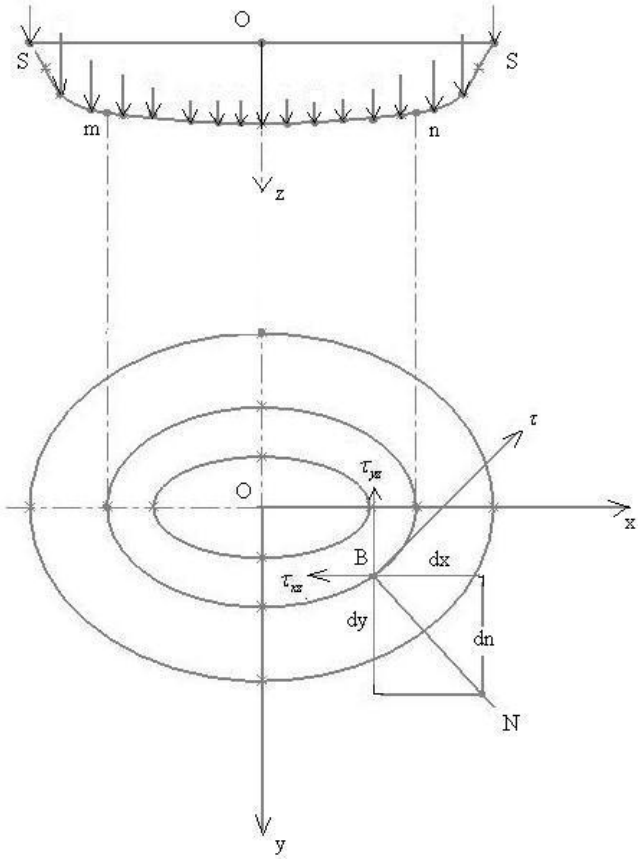


Fig. 16. Solid shaft of elliptic cross-section.

Suppose a cylindrical surface is generated by the stress line with its axis parallel to the axis of the shaft. Then, it can be concluded that there will be no stresses acting across this cylindrical surface (Timoshenko and Goodier, 1970). The material bounded by this surface can be imagined to be removed without changing the stress distribution in the outer portion of the shaft. Hence, the stress function given by Eq.(4.33) is applicable to the hollow shaft also.

For a given angle 'α'of twist, the stresses in the hollow shaft are the same as in the corresponding solid shaft. But, the torque will be smaller by the amount which in the case of the solid shaft is carried by the portion of the cross-section corresponding to the hole. The ratio of the torque carried by the part which is imagined to be removed to the total torque is determined to be $k^4 : 1$. Hence, for the hollow shaft, the Prandtl stress function is given by the expression

$$\phi = -\frac{M_t}{\pi ab(1-k^4)}\left(\frac{x^2}{a^2}+\frac{y^2}{b^2}-1\right) \tag{4.35}$$

Let us assume a case when the inner boundary which converts a simply connected domain into a multiply connected domain is no longer a stress line of the solid shaft. From the general theory of torsion, the stress function must be constant along each boundary, but these constants cannot be chosen arbitrarily unlike in the case of simply connected domains. By making some minor modifications to the expressions of displacements, the constants could be determined such that the displacements become

single-valued. A sufficient number of equations for determining these constants will then be obtained.

From Eqs (4.9), (4.15) and (4.16), we obtain

$$\sigma_{13} = \mu\left(\frac{\partial u_3}{\partial x} - \alpha y\right) \qquad \sigma_{23} = \mu\left(\frac{\partial u_3}{\partial y} + \alpha x\right) \tag{4.36}$$

By calculating the value of the integral along '$\int \tau\, ds$' each boundary by resolving the total stress into the components, we obtain

$$\begin{aligned} \int \tau\, ds &= \int\left(\sigma_{31}\frac{dx}{ds} + \sigma_{32}\frac{dy}{ds}\right)ds \\ &= \mu\int\left(\frac{\partial u_3}{\partial x}dx + \frac{\partial u_3}{\partial y}dy\right) - \mu\alpha\int(ydx - xdy) \end{aligned} \tag{4.37}$$

The first part of the integral in Eq.(4.37) vanishes from the condition that the integration is taken round a closed curve and that 'w' is a single-valued function. Hence,

$$\int \tau\, ds = \mu\alpha\int(xdy - ydx) \tag{4.38}$$

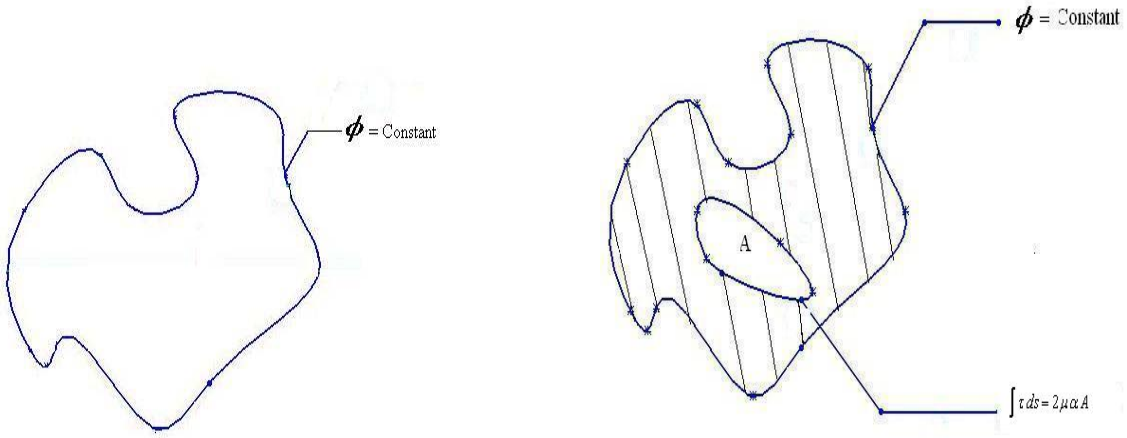
The integral on the right side in Eq.(4.38) is equal to double the area enclosed. Hence,

$$\int \tau\, ds = 2\mu\alpha A \tag{4.39}$$

Thus, it is required to determine the constant values of the stress function along the outer and inner boundaries so as to satisfy Eq.(4.39) for each boundary.

4.4 Implementation of the Boundary Conditions by FEM vs. FDM

From the theory presented in the previous two sections, it is observed that the value of the Prandtl stress function is constant along the boundary, irrespective of whether the domain is simply connected or multiply connected. However, it is crucial to recognize that a fundamental difference between the boundary conditions for the simply connected domains and the multiply connected domains exists.



(a) (b)

Fig. 17. (a) Simply connected domain. (b) Multiply connected domain.

For simply connected domains (Fig.17(a)), the Prandtl stress function can be assumed to be zero all along the boundary. For multiply connected domains (Fig.17(b)), the constant values of the Prandtl stress function on each of the boundaries cannot be arbitrarily assumed to be zero, they need to be determined. The value of this constant can be arbitrarily set to zero for one of the boundaries, but then the constant value of the Prandtl

stress function is no longer arbitrary on the other boundary. This gives rise to non-standard boundary conditions in the case of multiply connected domains.

The implementation of the boundary conditions by FEM and FDM for the two cases is an interesting case study. While there is no significant difference in the case of simply connected domains, in the case of multiply connected domains, FDM certainly holds an edge over FEM. As a result of the non-standard boundary condition for multiply connected domains, it is impossible for FEM to model multiply connected domains using the Prandtl stress function. As a result, the warping function is considered in FEM.

However, FDM can model multiply connected domains using the Prandtl stress function. Being able to model the domains using the Prandtl stress function is crucial since it allows the computation of parameters like torsional rigidity, which is defined as the factor by which the torque is divided to obtain the twist per unit length. The torsional rigidity can be expressed by means of an explicit expression in terms of the Prandtl stress function. In this thesis work, torsional rigidity is one of the factors based on which the effectiveness of the adaptive mesh refinement scheme has been determined. The next section details the computation of the analytical solution for a particular simply connected domain, a cylinder with a rectangular cross-section in this case. Based on the theory discussed in the previous two sections, it also discusses the method of computation of the constants at the boundary to satisfy the boundary conditions.

4.5 Analytical Solution for a Cylinder with a Rectangular Cross Section

Consider a uniform cylinder with rectangular cross-section of sides ‘a’ and ‘b’ with b>a as shown in Fig.18. The series solution of the torsion problem for a cylinder with rectangular cross-section is as follows. Choosing the coordinate axes through the centroid of the rectangle, a series form of the Prandtl’s stress function can be assumed as

$$\phi(x_1, x_2) = \sum_{k=0}^{\infty} Y_k(x_2) \cos(2k+1)\pi x_1 / a \tag{4.40}$$

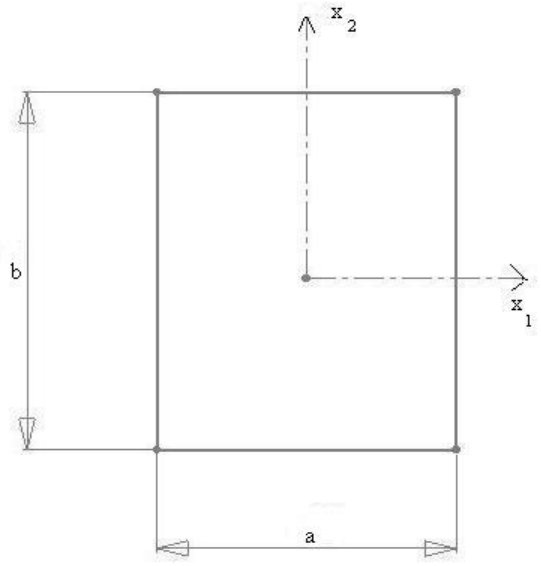


Fig. 18. Rectangular plate with sides a and b.

Boundary Conditions

$$\phi = 0 \;\; at \;\; x_1 = \pm\frac{a}{2} \tag{4.41}$$

$$\phi = 0 \;\; at \;\; x_1 = \pm\frac{b}{2} \tag{4.42}$$

The assumed series form satisfies Eq.(4.41), Eq.(4.42) requires

$$Y_k(b/2)=Y_k(-b/2)=0 \tag{4.43}$$

Substituting the series form of ϕ into the governing partial differential equation for the torsion problem $\nabla^2\phi=-2$, we obtain

$$\sum_{k=0}^{\infty}\left[Y''(x_2)-(2k+1)^2\frac{\pi^2}{a^2}Y_k(x_2)\right]\cos(2k+1)\pi x_1/a=-2 \tag{4.44}$$

From the Fourier series expansion,

$$\frac{4}{\pi}\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}\cos(2k+1)\pi x/a \;=\; 1 \tag{4.45}$$

Comparing Eqs (4.44) and (4.45) gives,

$$Y_k''-(2k+1)^2\frac{\pi^2}{a^2}Y_k=-\frac{8}{\pi}\frac{(-1)^k}{(2k+1)} \tag{4.46}$$

The solution of Eq.(4.46) is of the form shown in Eq.(4.47) with constants A_k and B_k to be determined from Eq.(4.41),

$$Y_k=A_k\cosh(2k+1)\pi\frac{x_2}{a}+B_k\sinh(2k+1)\pi\frac{x_2}{a}+\frac{8a^2}{\pi^2}\frac{(-1)^k}{(2k+1)^3} \tag{4.47}$$

$$\begin{aligned} &A_k\cosh(2k+1)\pi\frac{b}{2a}=-\frac{8a^2}{\pi^3}\frac{(-1)^k}{(2k+1)^3} \\ &B_k=0 \end{aligned} \tag{4.48}$$

Substituting the values obtained into the series form of Prandtl's function, we obtain

$$\phi = \frac{8a^2}{\pi^3}\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)^3}\left\{1-\frac{\cosh(2k+1)\pi x_2/a}{\cosh(2k+1)\pi b/2a}\right\}\cos(2k+1)\pi x_1/a \qquad (4.49)$$

Calculation of the stresses and the torsional rigidity

The non-zero stress components are given by the following expression:

$$\sigma_{13} = \sigma_{31} = \mu\alpha(\psi_{,1} - X_2) \qquad (4.50)$$

$$\sigma_{23} = \sigma_{32} = \mu\alpha(\psi_{,2} + X_1) \qquad (4.51)$$

where,

$$\psi_{,1} = -\frac{8a}{\pi^2}\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)^2}\frac{\sinh(2k+1)\pi x_2/a}{\cosh(2k+1)\pi b/2a}\cos(2k+1)\pi x_1/a + X_2 \qquad (4.52)$$

$$\psi_{,2} = \frac{8a}{\pi^2}\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)^2}\left\{1-\frac{\cosh(2k+1)\pi x_2/a}{\cosh(2k+1)\pi b/2a}\right\}\sin(2k+1)\pi x_1/a - X_1 \qquad (4.53)$$

The maximum shear stress occurs at the mid-points of the longer sides $(x_1, x_2) = (\pm\frac{a}{2}, 0)$, and its magnitude can be computed by considering Eqs (4.51) and (4.53) as,

$$(\tau_s)_{\max} = \mu\alpha a - \frac{8\mu\alpha a}{\pi^2}\sum_{k=0}^{\infty}\frac{1}{(2k+1)^2}\sec h(2k+1)\pi b/2a \qquad (4.54)$$

The series converges rapidly and can be approximated by its first term with an error of less than 0.5%. Hence,

$$(\tau_s)_{\max} = \mu\alpha a - \frac{8\mu\alpha a}{\pi^2}\left(1 - \frac{8}{\pi^2}\sec h\frac{\pi b}{2a}\right) \tag{4.55}$$

The torsional rigidity is given by

$$D = \frac{32\mu a^3}{\pi^4}\sum_{k=0}^{\infty}\left\{\frac{b}{(2k+1)^4} - \frac{2a}{\pi(2k+1)^5}\tanh(2k+1)\pi b/2a\right\} \tag{4.56}$$

On similar lines, the analytical solutions for the torsion problem are available for a wide variety of cross-sections.

The next chapter presents the results obtained by implementing the adaptive mesh refinement algorithm on a few simply connected domains.

CHAPTER V

RESULTS AND DISCUSSION

5.1 Adaptive Mesh Refinement Algorithm

The following algorithm outlines the general procedure followed, to accomplish adaptive mesh refinement, during the analysis of cylinders of different cross-section, in this thesis work.

1. Test case: Cylinder of a particular cross-section
   a) $\nabla^2\phi = -2\mu\alpha$ is solved on a coarse uniform grid using central difference scheme.
   b) Plots for 'ϕ' and '$|\nabla\phi|$' are obtained.

2. Adaptive mesh refinement code
   a) The values of '$|\nabla\phi|$' are the 'input' for the adaptive mesh refinement code.
   b) Quadtree decomposition technique is used to split every element which meets all of the following three criteria:
      (i) The size of the element is greater than a particular specified 'minimum dimension'.
      (ii) The average value of the '$|\nabla\phi|$' over the element is greater than a 'specified' value.

(iii) The element has more than two neighboring elements along any edge.

c) Splitting of elements occurs until no element in the domain satisfies all the above criteria simultaneously. An example of how this algorithm works has been provided in the appendix.

d) The new mesh generated is used to solve the Laplace equation.

e) Plots for 'ϕ' and the '$|\nabla\phi|$' are obtained.

3. Comparison of results

The results obtained from a fine uniform mesh and non-uniform mesh is compared with the analytical solution.

One of the primary parameters compared, to determine the effectiveness of the adaptive mesh refinement algorithm developed, is the torsional rigidity. Torsional rigidity is defined as the factor by which the torque is divided to obtain the twist per unit length.

In this thesis work, the torsional rigidity computed by using a fine uniform mesh as also a non-uniform mesh produced by the mesh refinement algorithm is compared with the one available from the analytical solution. The difference between the values is expressed in the form of a ratio as shown below.

$$Error\% = \frac{(TR)_{Solution basis} - (TR)_{Analytical}}{(TR)_{Analytical}} \tag{5.1}$$

where TR = Torsional Rigidity

5.2 Rectangular Domains

Test Case I: Cylinder of rectangular cross section

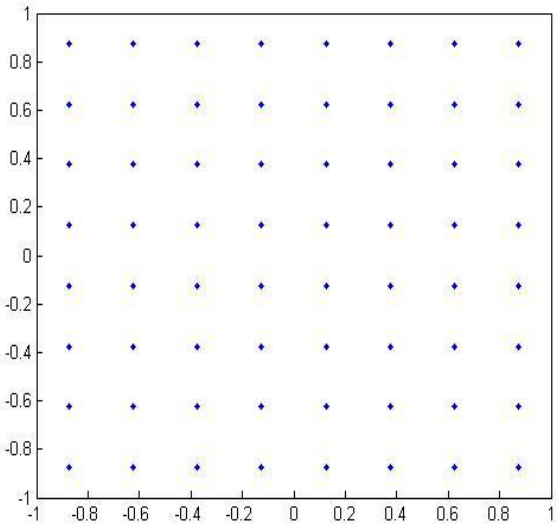


Fig. 19. Coarse uniform mesh - rectangular cross section.

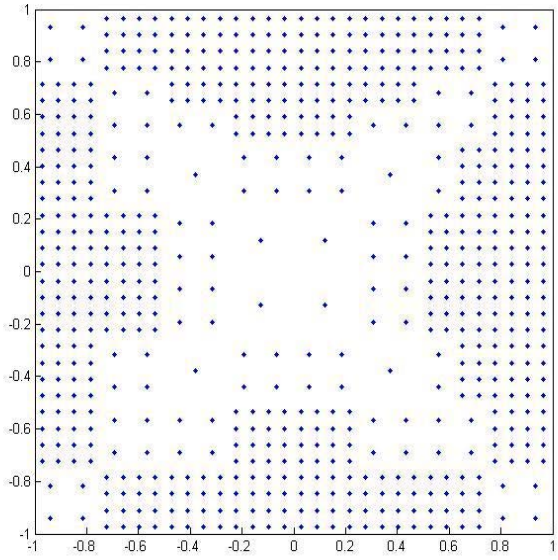


Fig. 20. Refined mesh - rectangular cross section.

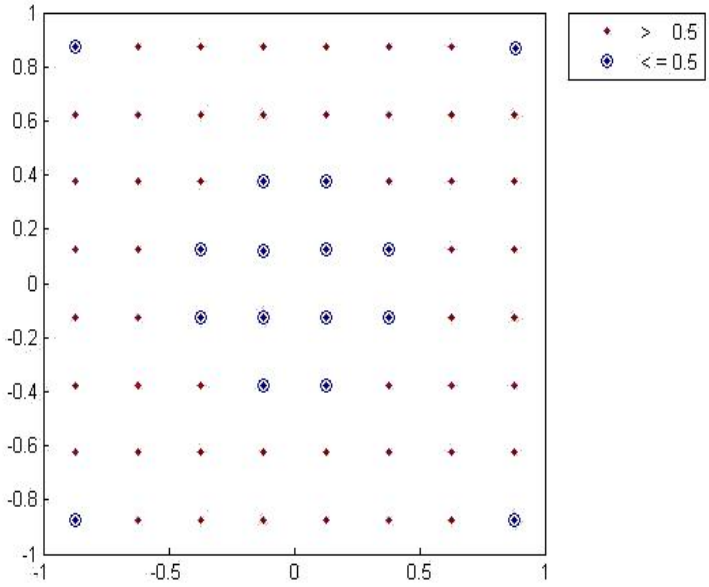


Fig. 21. '$|\nabla\phi|$' - coarse uniform mesh – rectangular cross section.

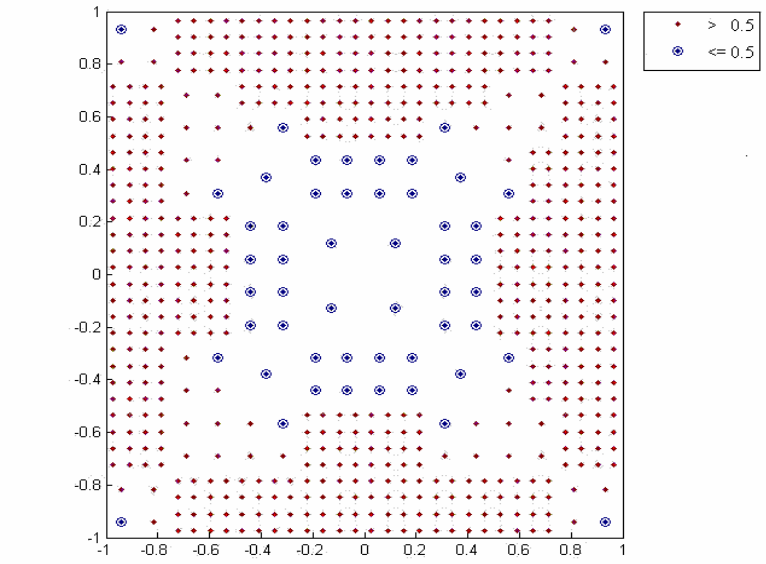


Fig. 22. '$|\nabla\phi|$' - refined mesh – rectangular cross section.

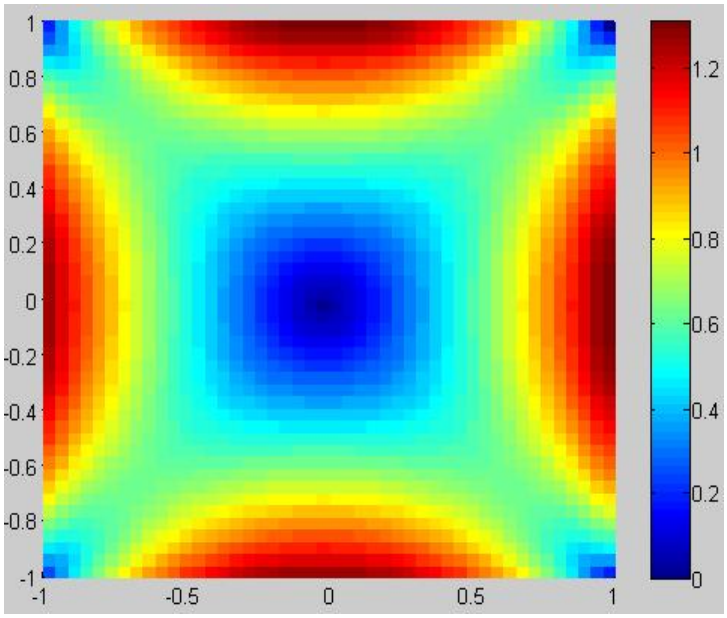


Fig. 23. Surf of '$|\nabla\phi|$' - fine uniform mesh – rectangular cross section.

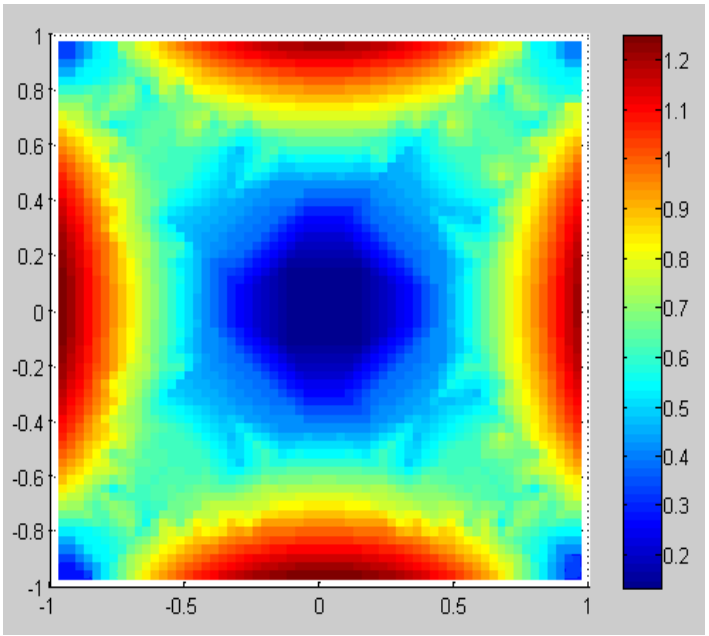


Fig. 24. Surf of '$|\nabla\phi|$' - refined mesh – rectangular cross section.

Table 1. Tabulated results for the case of a cylinder of rectangular cross section.

S. No	Solution Basis	Number of nodes	Torsional Rigidity	Error%
1	Analytical	-	2.2488	Datum
2	Fine Uniform Mesh	3600	2.2474	0.62
3	Refined Non-uniform Mesh using Mesh Refinement Algorithm	640	2.2489	0.04

The coarse uniform mesh for the rectangular cross section is shown in Fig. 19. Fig. 20 shows the refined non uniform mesh generated by the mesh refinement algorithm. Figs 21 and 22 show the corresponding comparative plots of '$|\nabla\phi|$'. The surface plots generated for the fine uniform mesh and the refined mesh are shown in Figs 23 and 24. The relevant results have been tabulated in Tab.1.

Test Case II: Cylinder of ‘C’ shaped cross section

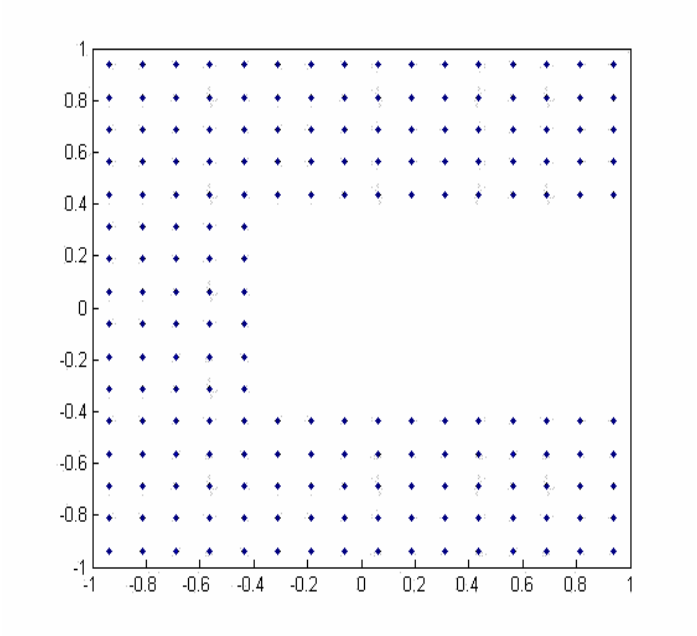


Fig. 25. Coarse uniform mesh – ‘C’ cross section.

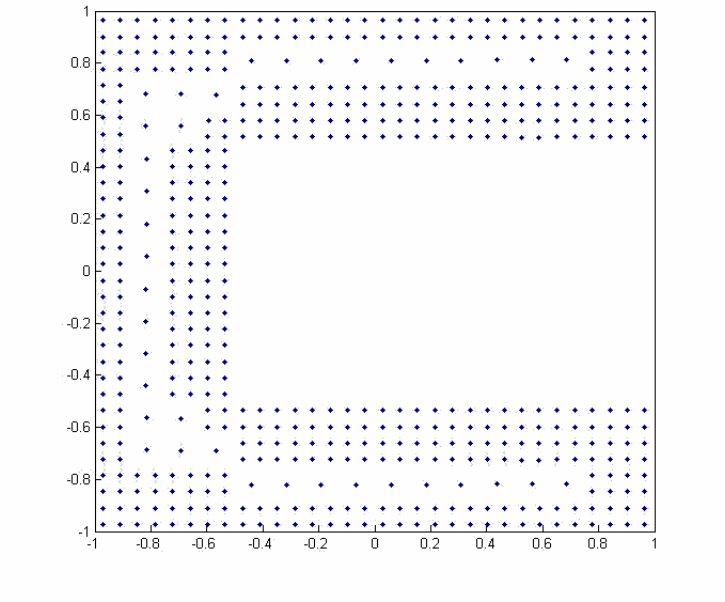


Fig. 26. Refined mesh – ‘C’ cross section.

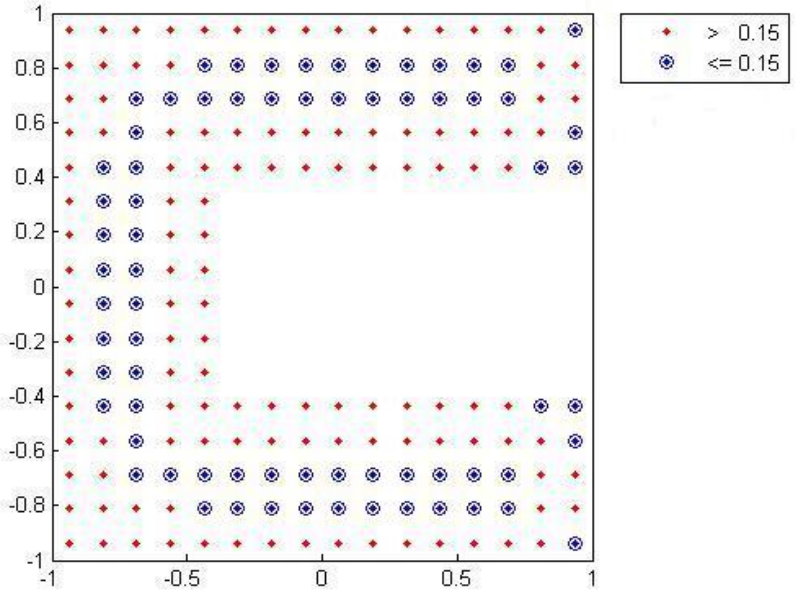


Fig. 27. '$|\nabla\phi|$' - coarse uniform mesh – 'C' cross section.

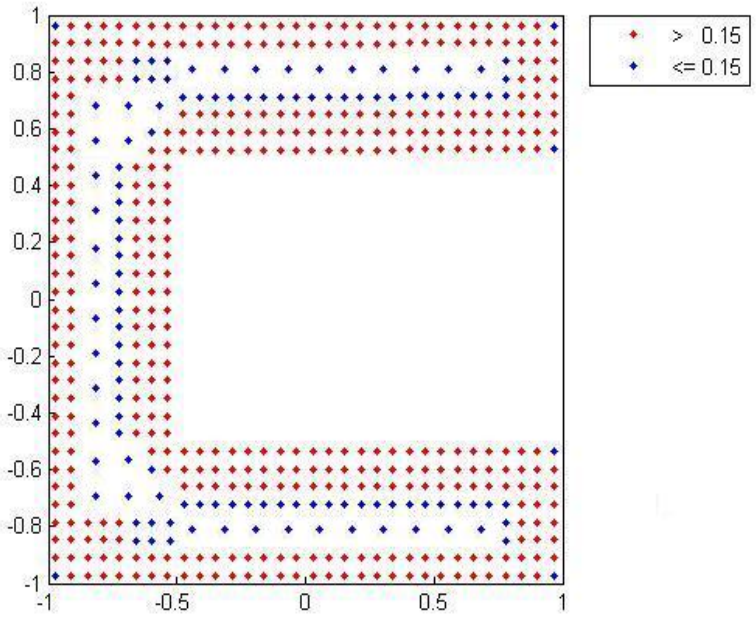


Fig. 28. '$|\nabla\phi|$' - refined mesh – 'C' cross section.

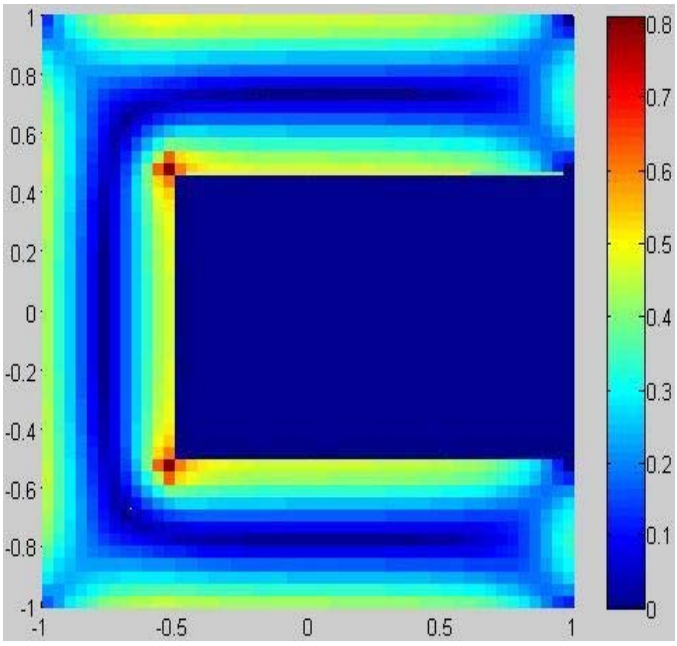


Fig. 29. Surf of '$|\nabla\phi|$' - fine uniform mesh – 'C' cross section.

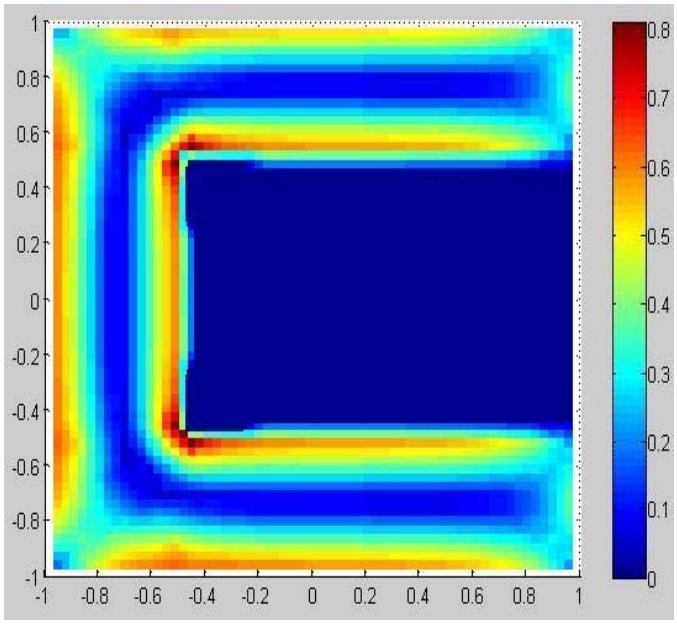


Fig. 30. Surf of '$|\nabla\phi|$' - refined mesh – 'C' cross section.

Table 2. Tabulated results for the case of a cylinder of 'C' shaped cross section.

S. No	Solution Basis	Number of nodes	Torsional Rigidity	Error%
1	Analytical	-	0.2083	Datum
2	Fine Uniform Mesh	3600	0.2028	2.64
3	Refined Non-uniform Mesh using Mesh Refinement Algorithm	461	0.2043	1.92

The coarse uniform mesh for the 'C' cross section is shown in Fig. 25. Fig. 26 shows the refined non uniform mesh generated by the mesh refinement algorithm. Figs 27 and 28 show the corresponding comparative plots of '$|\nabla\phi|$'. The surface plots generated for the fine uniform mesh and the refined mesh are shown in Figs 29 and 30.

The relevant results have been tabulated in Tab.2.

5.3 Non Rectangular Domains

Test Case III: Cylinder of circular cross section

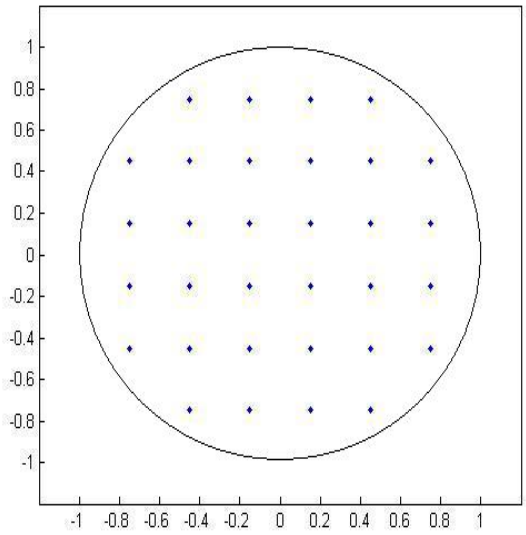


Fig. 31. Coarse uniform mesh – circular cross section.

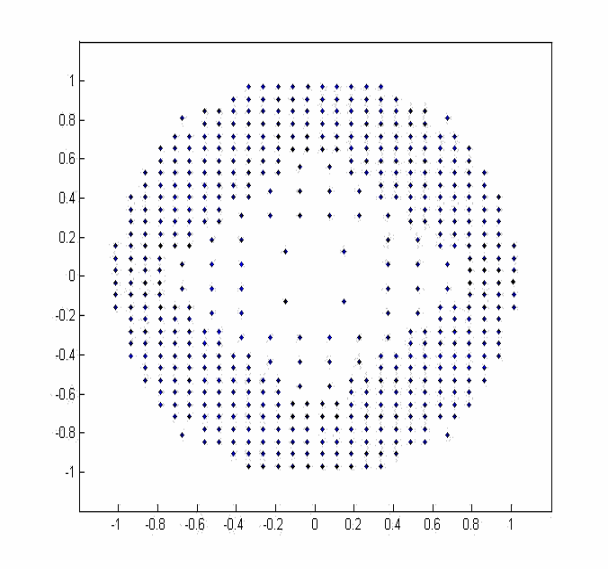


Fig. 32. Refined mesh – circular cross section.

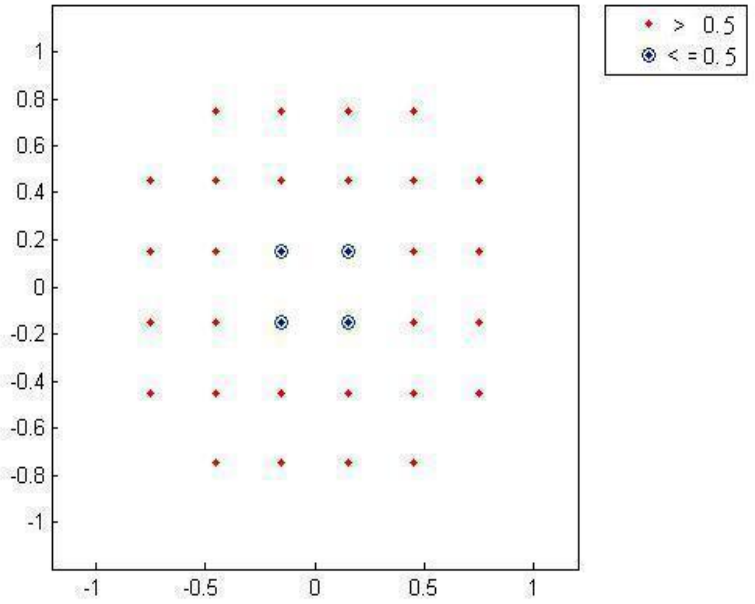


Fig. 33. '$|\nabla\phi|$' - coarse uniform mesh – circular cross section.

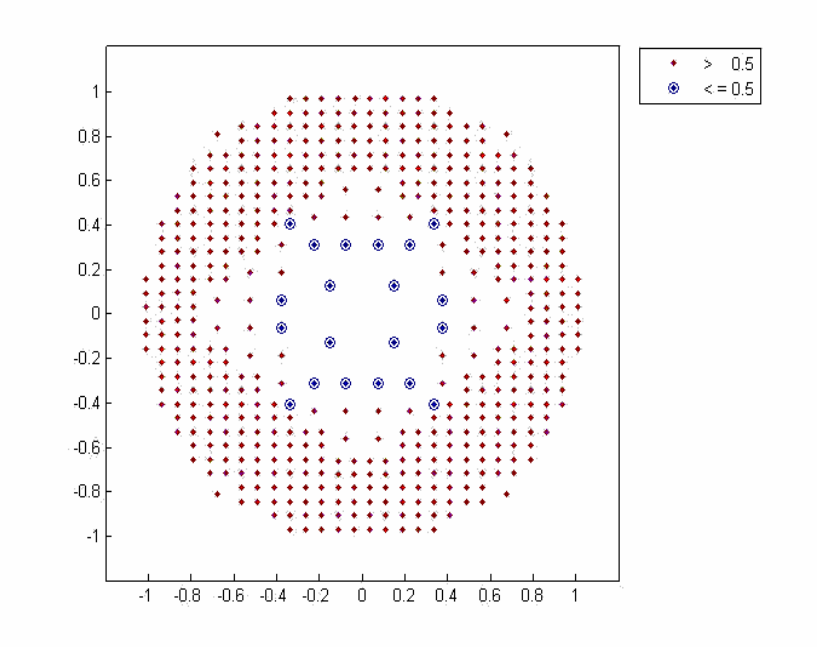


Fig. 34. '$|\nabla\phi|$' - refined mesh – circular cross section.

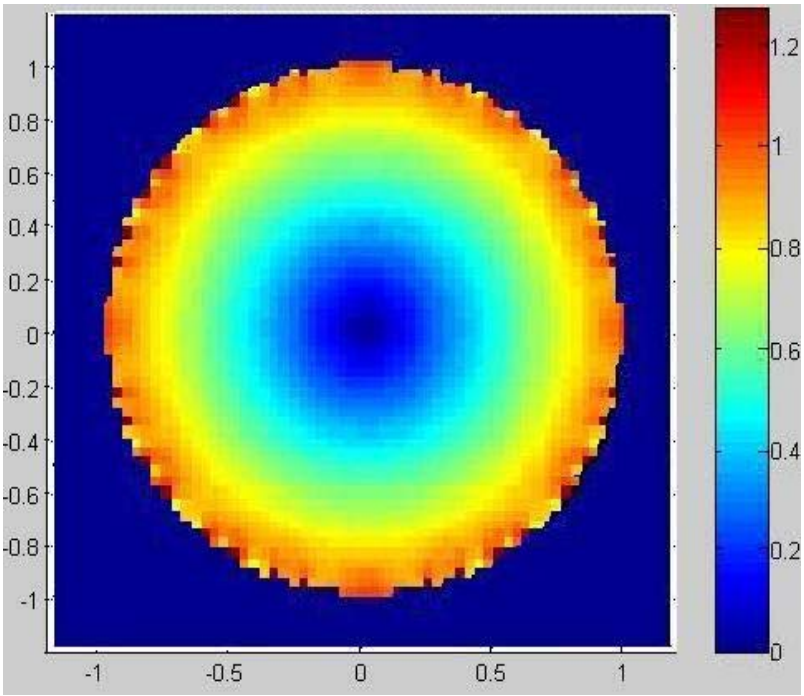


Fig. 35. Surf of '$|\nabla\phi|$' - fine uniform mesh – circular cross section.

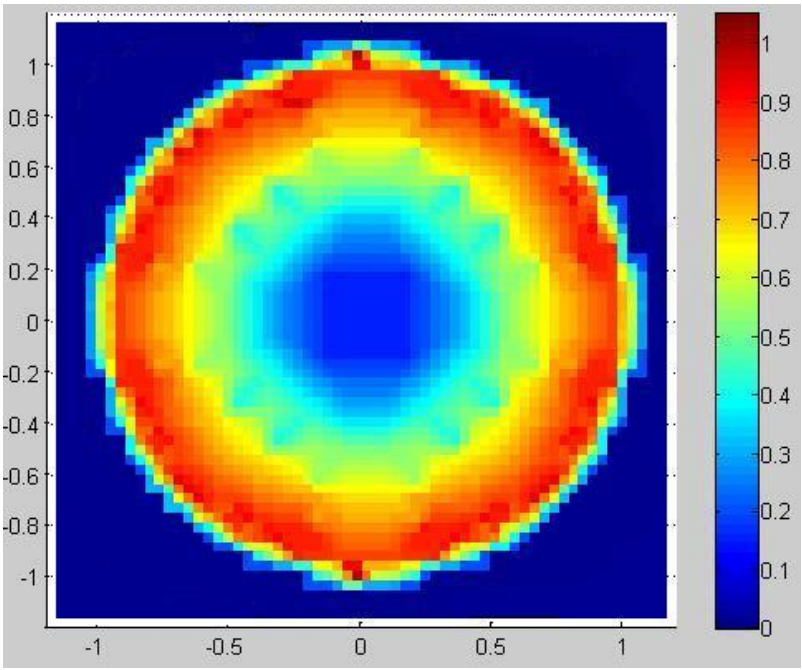


Fig. 36. Surf of '$|\nabla\phi|$' - refined mesh – circular cross section.

Table 3. Tabulated results for the case of a cylinder of circular cross section.

S. No	Solution Basis	Number of nodes	Torsional Rigidity	Error%
1	Analytical	-	1.5707	Datum
2	Fine Uniform Mesh	3600	1.5839	0.08
3	Refined Non-uniform Mesh using Mesh Refinement Algorithm	496	1.5424	1.80

The coarse uniform mesh for the circular cross section is shown in Fig. 31. Fig. 32 shows the refined non uniform mesh generated by the mesh refinement algorithm. Figs 33 and 34 show the corresponding comparative plots of '$|\nabla\phi|$'. The surface plots generated for the fine uniform mesh and the refined mesh are shown in Figs 35 and 36. The relevant results have been tabulated in Tab.3.

Test Case IV: Cylinder with elliptical cross section

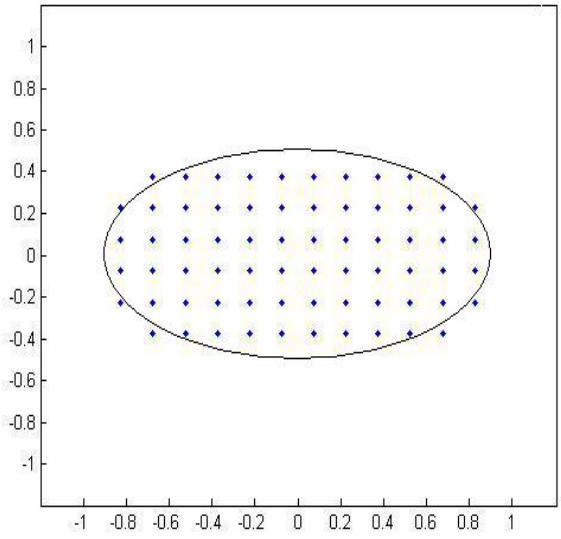


Fig. 37. Coarse uniform mesh – elliptical cross section.

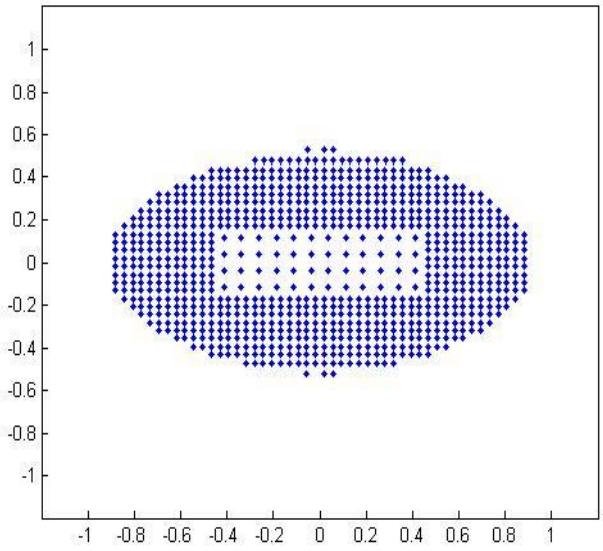


Fig. 38. Refined mesh – elliptical cross section.

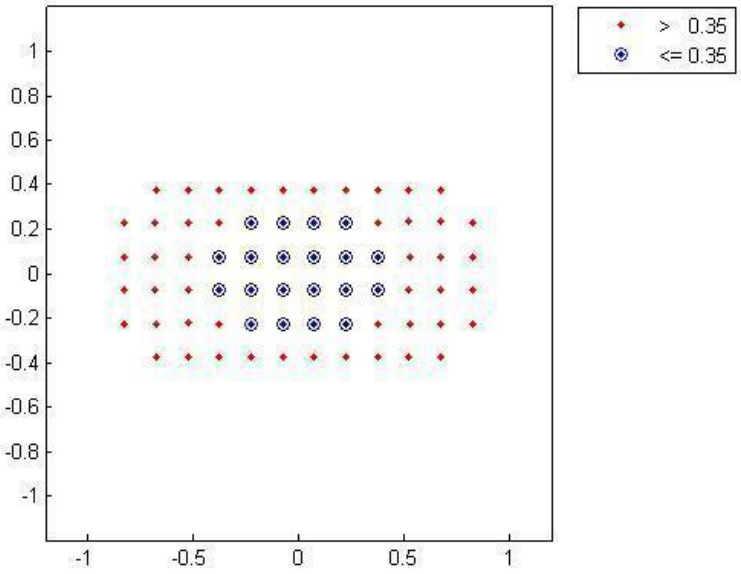


Fig. 39. '$|\nabla\phi|$' - coarse uniform mesh – elliptical cross section.

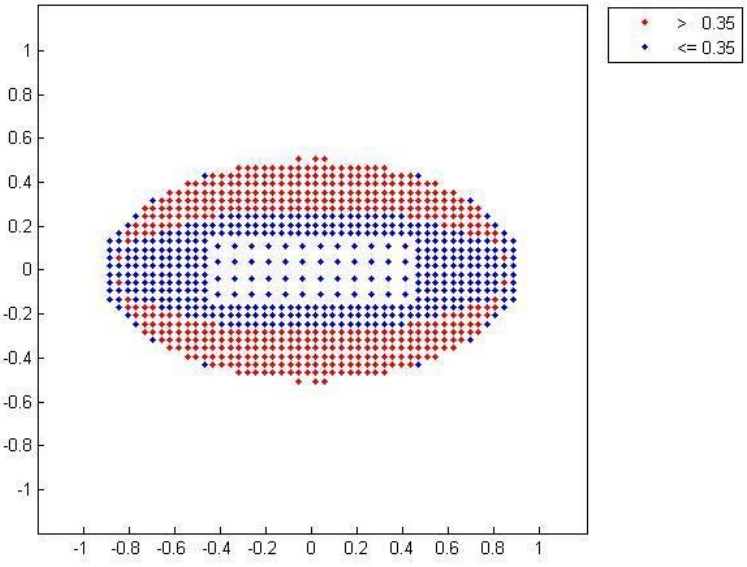


Fig. 40. '$|\nabla\phi|$' - refined mesh – elliptical cross section.

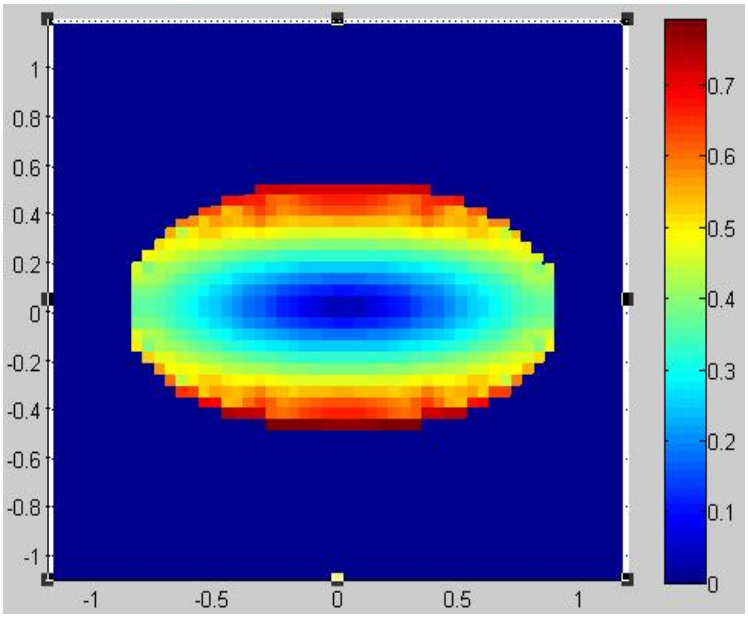


Fig. 41. Surf of '$|\nabla\phi|$' - fine uniform mesh – elliptical cross section.

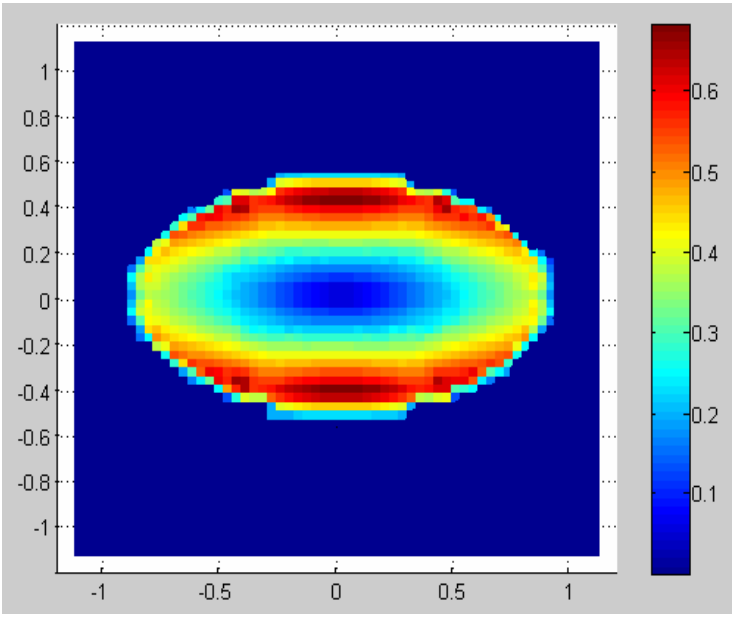


Fig. 42. Surf of '$|\nabla\phi|$' - refined mesh – elliptical cross section.

Table 4. Tabulated results for the case of a cylinder of elliptical cross section.

S. No	Solution Basis	Number of nodes	Torsional Rigidity	Error%
1	Analytical	-	0.2701	Datum
2	Fine Uniform Mesh	3600	0.2595	3.92
3	Refined Non-uniform Mesh using Mesh Refinement Algorithm	856	0.2612	3.29

The coarse uniform mesh for the elliptical cross section is shown in Fig. 37. Fig. 38 shows the refined non uniform mesh generated by the mesh refinement algorithm. Figs 39 and 40 show the corresponding comparative plots of '$|\nabla\phi|$'. The surface plots generated for the fine uniform mesh and the refined mesh are shown in Figs 41 and 42. The relevant results have been tabulated in Tab.4.

5.4 Multiply Connected Domains

Test Case: Square cross section with a square hole at the centre

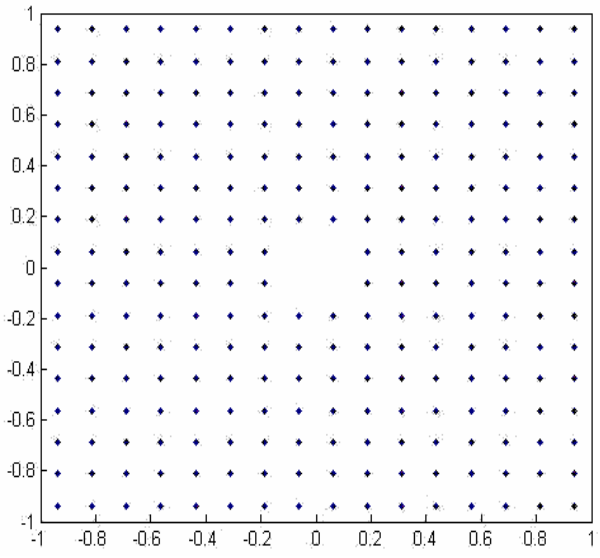


Fig. 43. Coarse uniform mesh – square cross section with hole.

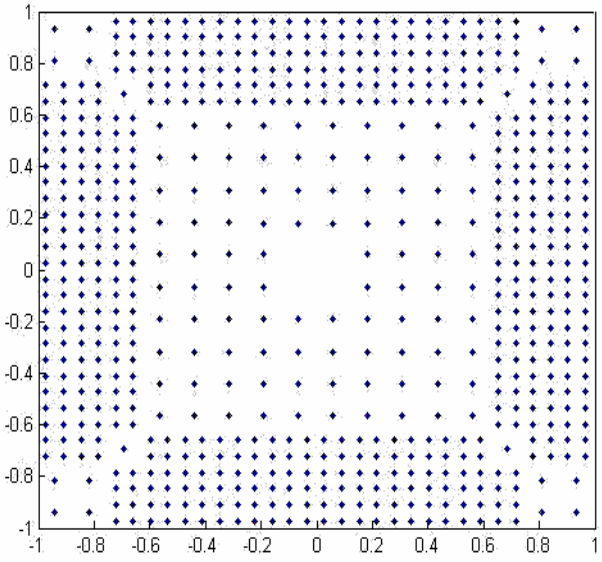


Fig. 44. Refined mesh – square cross section with hole.

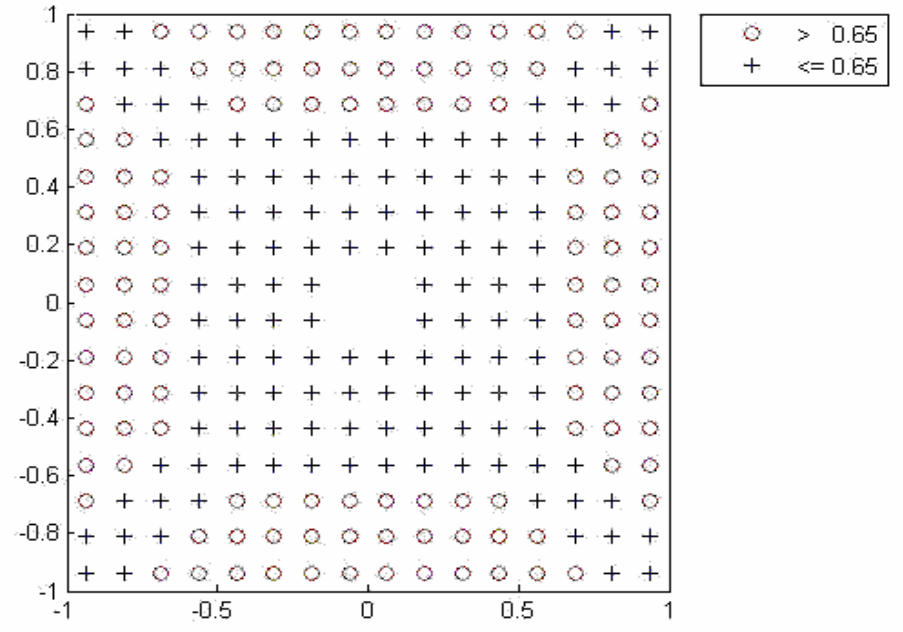


Fig. 45. '$|\nabla\phi|$' - coarse uniform mesh – square cross section with hole.

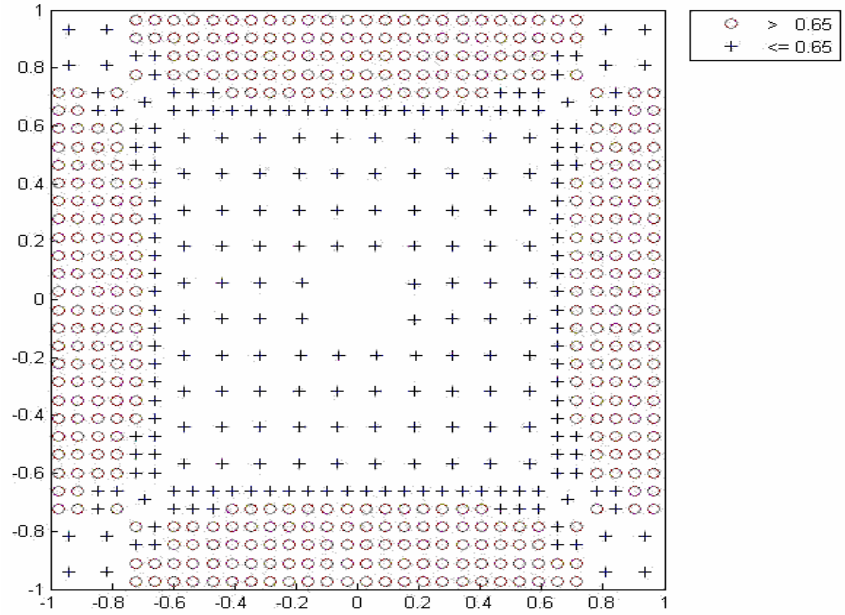


Fig. 46. '$|\nabla\phi|$' - refined mesh – square cross section with hole.

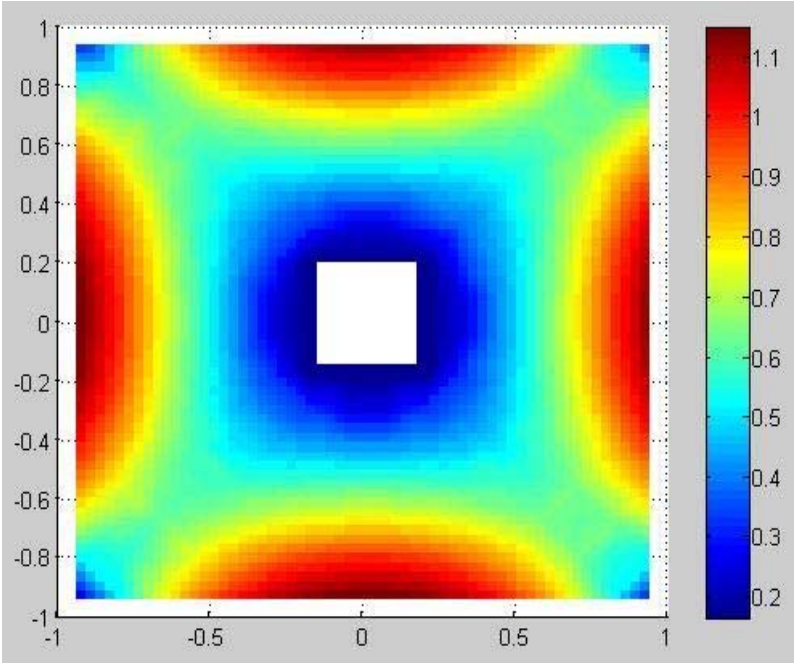


Fig. 47. Surf of '$|\nabla\phi|$' - fine uniform mesh – square cross section with hole.

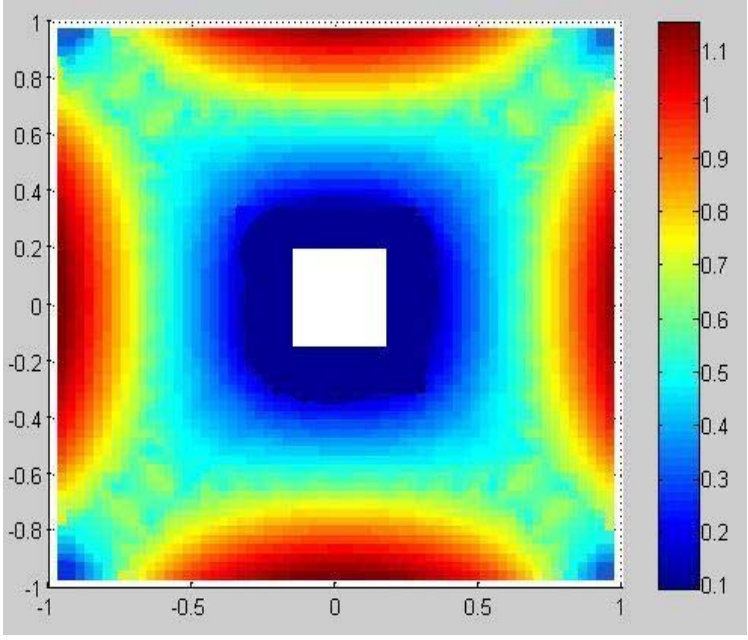


Fig. 48. Surf of '$|\nabla\phi|$' - refined mesh – square cross section with hole.

Table 5. Tabulated results for the case of a cylinder of square cross-section with a hole.

S. No	Solution Basis	Number of nodes	Torsional Rigidity	Error%
1	Analytical	-	-	Datum
2	Fine Uniform Mesh	3600	2.1265	-
3	Refined Non-uniform Mesh using Mesh Refinement Algorithm	644	2.104	-

The coarse uniform mesh for the square cross section with a hole is shown in Fig. 43. Fig. 44 shows the refined non uniform mesh generated by the mesh refinement algorithm. Figs 45 and 46 show the corresponding comparative plots of '$|\nabla\phi|$'. The surface plots generated for the fine uniform mesh and the refined mesh are shown in Figs 47 and 48. The relevant results have been tabulated in Tab.5.

5.5 Discussion

On the basis of his investigations, Saint Venant had drawn some conclusions of practical interest (Timoshenko and Goodier, 1970). The maximum stress in all cases analyzed by him was obtained at the boundary of the cross-section at the points which are nearest to the centroid of the cross-section. From Figs 22, 28 and 40, it is observed that the mesh is finer at the boundaries, which roughly corroborates these conclusions of Saint Venant. However, a more detailed investigation about Saint Venant's conclusions was later made by Filon, which showed that there are cases where the points of maximum stress, although always at the boundary, are not the nearest points to the centroid of the cross-section (Timoshenko and Goodier, 1970).

It is evident from the tabulated results that, for each of the cross-sections for which the simulations have been run, the Error % are within a reasonable range of 2-4%. In addition to the Error%, lot of insight could also be gained by proper interpretation of some other parameters. In this work, the criterion for domain decomposition is the $|\nabla\phi|$. In Fig.34, it is observed that the mesh gets finer as we move radially outward, which is a consequence of an increase in the $|\nabla\phi|$ as we move away from the centre. It is also observed that there is hardly any refinement near the centre of the circle where the Prandtl Stress Function is expected to be a maximum. There is no decomposition in this region because $|\nabla\phi|$ is expected to be almost zero.

Some of the other conclusions of Saint Venant of enormous practical importance are:

- In the case of simply connected domains, for a given cross-sectional area, the torsional rigidity increases if the polar moment of inertia of the cross-section decreases. Thus, for a given amount of material, the circular shaft gives the highest torsional rigidity.
- For a given torque and cross-sectional area, the maximum stress is the smallest for the cross-section with the smallest polar moment of inertia.
- The torsional rigidity for any singly connected cross-section can be approximated by replacing the given shaft by the shaft of an elliptic cross-section having the same cross-sectional area and the same polar moment of inertia as the given shaft.

CHAPTER VI

CONCLUSION

6.1 Future Work

The results of this thesis work are a clear indication of the potential of the implementation of adaptive mesh refinement strategies based on recursive structures such as quadtrees and octrees. The present work was limited to simply connected domains. Future work could include implementation of the strategy for simply connected domains of more complex geometries and multiply connected domains.

Further, the present work has been limited to spatial discretization. It would be interesting to introduce the aspect of discretization in time and verify the effectiveness. A simple application could be the implementation of the strategy for dynamic modeling of wave fronts. The solution at one time step could be used to refine the mesh at the wave front, and the refined mesh could be used to compute the solution at the next time interval. This method could turn out to be extremely effective with appropriate time intervals based on the type of system being modeled.

Also, one of the significant features of this thesis work is the incorporation of MEX in the simulations. However, in this work, it has not been possible to use MEX for dynamic

memory allocation. Appending this feature could have a huge impact on the computational resources required for the simulations.

6.2 Conclusion

In summary, the following were the main objectives of this thesis work:

- To produce an adaptive mesh refinement algorithm
- To implement the algorithm on problems of solid mechanics
- To determine its effectiveness by comparing the solutions obtained with the analytical solution

The listed objectives have been achieved and the related documentation has been provided in the form of the plots and the tabulated results. It was proposed that this thesis work would be limited to simply connected domains.

REFERENCES

Anderson D. A., Tannehill J. C. and Pletcher R. H. (1984): *Computational fluid mechanics and heat transfer.* - Washington, DC: McGraw-Hill Book Company.

Berger M. and Colella P. (1989): *Local adaptive mesh refinement for shock hydrodynamics.* - J. Comput. Phys., vol. 82, pp. 64-84.

Berger M. and Oliger J. (1984): *Adaptive mesh refinement for hyperbolic partial differential equation.* - J. Comput. Phys., vol. 53, pp. 484-512.

Chapman S. J. (2000): *MATLAB programming for engineers.* – Pacific Grove, CA: Brooks/Cole Publishing Company.

Ceniceros H. D. and Roma A. M. (2004): *Study of long-time dynamics of a viscous vortex sheet with a fully adaptive non-stiff method.* – J. Comput. Phys., vol. 16, No.12, pp. 4285-4318.

Chou P. C. and Pagano N. J. (1992): *Elasticity: tensor, dyadic and engineering approaches.* – New York: Dover Publications, Inc.

Gibou F. and Fedkiw R. (2005): *A fourth-order accurate discretization for the laplace and heat equations on arbitrary domains, with applications to stefan problem.* - J. Comput. Phys., vol. 202, pp. 577-601.

Gibou F., Fedkiw R., Cheng L.T. and Kang M. (2002): *A second-order accurate symmetric discretization of the poisson equation on irregular domains.* - J. Comput. Phys., vol. 176, pp. 205-227.

Gonzalez R.C., Woods R., Eddins S. L. (2005): *Digital image processing using Matlab.* – Delhi, India: Pearson Education (Singapore) Pte. Ltd.

Ham F., Lien F. and Strong A. (2002): *A fully conservative second-order finite difference scheme for incompressible flow on non-uniform grids.* - J. Comput. Phys., vol. 117, pp. 117-133.

Johansen H. and Colella P. (1998): *A Cartesian grid embedded boundary method for poisson's equation on irregular grids.* – J. Comput. Phys., vol. 147, pp. 60-85.

LeVeque R. and Li Z. (1994): *The immersed interface method for elliptic equations with discontinuous coefficients and singular sources.* – SIAM J. Numer. Anal., vol. 31, pp. 1019-1044.

Li Z. (1998): *A fast iterative algorithm for elliptic interface problems.* – SIAM J. Numer. Anal., vol. 35, pp. 230-254.

MATLAB (1998): *MATLAB application program interface guide.* – Natick, MA: The MATHWORKS Inc.

Mayo A. (1984): *The fast solution of poisson's and the biharmonic equations on irregular regions.* – SIAM J. Numer. Anal., vol. 21, pp. 285-299.

McCorquodale P., Colella P., Grote D. and Vay J. L. (2004): *A node-centered local refinement algorithm for poisson's equation in complex geometries.* – J. Comput. Phys., vol. 201, pp. 34-60.

McKenney A. and Greengard L. (1995), *A fast poisson solver for complex geometries.* – J. Comput. Phys., vol. 118, pp. 348-355.

Min C., Gibou F. and Ceniceros H. D. (2006): *A supra-convergent finite difference scheme for the variable coefficient Poisson equation on non-graded grids.* - J. Comput. Phys., submitted.

Mitchell A. R. and Griffiths D. F. (1980): *The finite difference method in partial differential equations.* – Chichester, UK: John Wiley and Sons.

Neeman H. J. (1996): *Autonomous hierarchical mesh refinement for multiscale simulations.* – PhD Thesis, University of Illinois, Urbana-Champaign.

Reddy J. N. (2003): *An introduction to the finite element method.* – New Delhi, India: Tata McGraw- Hill Publishing Company Limited.

Samet H. (1990): *Applications of spatial data structures: computer graphics, image processing, and GIS.* – Reading, MA: Addison-Wesley Publishing Company.

Slaughter W. S. (2002): The *linearized theory of elasticity.* – Boston, MA, Birkhauser.

Sokolnikoff I.S. (1983): *Mathematical theory of elasticity.* – Malabar, FL: Krieger Pub. Co.

Sussman M., Algrem A. S., Bell J. B., Colella P., Howell L. H. and Welcome M. L. (1999): *An adaptive level set approach for incompressible two-phase flow.* - J. Comput. Phys., vol. 148, pp. 81-124.

Timoshenko S. P. and Goodier J. N. (1970): *Theory of elasticity.* – New York: McGraw-Hill Book Company.

Wikipedia, Numerical Analysis, Quadtree, Accessed on 09/25/06

http://en.wikipedia.org/wiki/Numerical_analysis.

http://en.wikipedia.org/wiki/Quadtree.

APPENDIX

In this section, an example of recursive domain discretization based on the criterions of '$|\nabla\phi|$' and 'minimum dimension of the region' is presented. We begin with a sample coarse domain and define the criterions for domain decomposition. The criterions are applied to each of the regions of the domain. Regions, which meet the criterions simultaneously, undergo quadtree decomposition. This process continues iteratively until no region in the domain satisfies the criterions simultaneously. In this example, sub-division should occur if both of the following conditions are satisfied:

1. Mean > 0.38
2. Size of region > 0.125

Let us now consider the sample domain shown in Fig.49 and the relevant tabulation shown in Table 5.

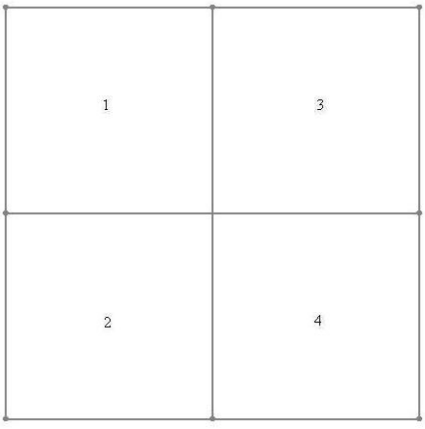


Fig. 49. Domain: Stage-1.

Table 6. Tabulated Results from Stage-1 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	$\lvert\nabla\phi\rvert$ (f)	Area of region (A)	Mean = fA / A	Further sub-division needed? YES / NO	Region Number
-0.5	0.5	1	0.32	1	0.32	NO	1
-0.5	-0.5	1	0.40	1	0.40	YES	2
0.5	0.5	1	0.40	1	0.40	YES	3
0.5	-0.5	1	0.32	1	0.32	NO	4

From the tabular column, it is evident that regions 2 and 3 satisfy both the criterions specified. Hence, quadtree decomposition of these regions occurs. The resultant domain has been shown in the Fig. 50.

One of the significant things to note is the way the regions are numbered. It is also interesting to note the way numbering proceeds when moving from one stage to another. In this work, the left top coordinates of the regions is the basis of numbering. All regions whose left top lies along one column are accessed before moving on to adjacent column of regions and the corresponding left tops. As a result, since any stage might involve decomposition of a particular region, no region has any fixed 'region number'. These 'region numbers' keep changing dynamically and have been shown in the last column of Table 6 and Table 7.

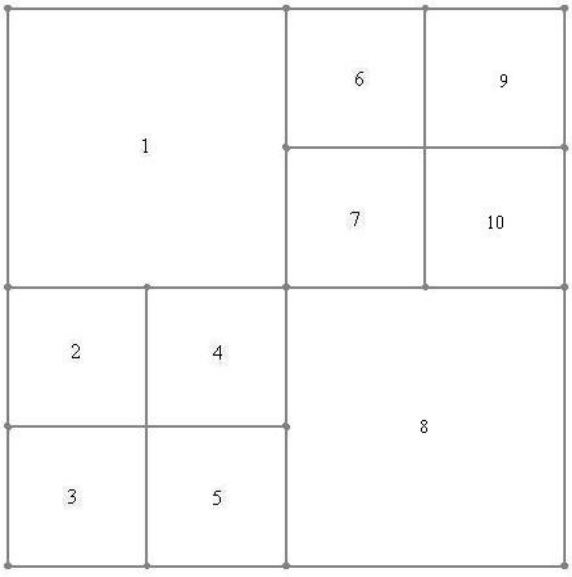


Fig. 50. Domain: Stage-2.

Table 7. Tabulated Results from Stage-2 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	$\lvert\nabla\phi\rvert$ (f)	Area of region (A)	Mean = fA / A	Further sub-division needed? YES / NO	Region Number
-0.50	0.50	1.00	0.32	1.00	0.32	NO	1
-0.75	-0.25	0.50	0.38	0.25	0.38	NO	2
-0.75	-0.75	0.50	0.36	0.25	0.36	NO	3
-0.25	-0.25	0.50	0.48	0.25	0.48	YES	4
-0.25	-0.75	0.50	0.38	0.25	0.38	NO	5
0.25	0.75	0.50	0.38	0.25	0.38	NO	6
0.25	0.25	0.50	0.48	0.25	0.48	YES	7
0.50	-0.50	1.00	0.32	1.00	0.32	NO	8
0.75	0.75	0.50	0.36	0.25	0.36	NO	9
0.75	0.25	0.50	0.38	0.25	0.38	NO	10

The mean of the regions 4 and 7 exceed the criterion set and the size of these regions exceeds the minimum limit set. As a result, these regions can get sub-divided further. The domain after discretization is shown in Fig. 51.

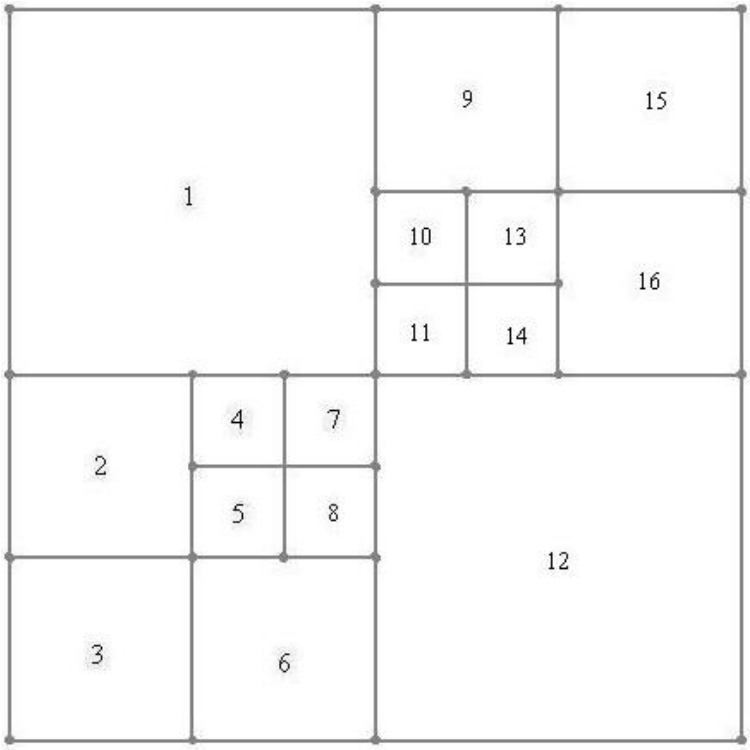


Fig. 51. Domain: Stage-3.

The details relevant to this discretized domain are presented in Table 8. The regions requiring decomposition have been identified.

Table 8. Tabulated Results from Stage-3 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	$\lvert\nabla\phi\rvert$ (f)	Area of region (A)	Mean = fA / A	Further sub-division needed? YES / NO	Region Number
-0.500	0.500	1.00	0.32	1.00	0.32	NO	1
-0.750	-0.250	0.50	0.38	0.25	0.38	NO	2
-0.750	-0.750	0.50	0.36	0.25	0.36	NO	3
-0.375	-0.125	0.25	0.37	0.0625	0.37	NO	4
-0.375	-0.375	0.25	0.37	0.0625	0.37	NO	5
-0.250	-0.750	0.50	0.38	0.25	0.38	NO	6
-0.125	-0.125	0.25	0.37	0.0625	0.37	NO	7
-0.125	-0.375	0.25	0.37	0.0625	0.37	NO	8
0.250	0.750	0.50	0.38	0.25	0.38	NO	9
0.125	0.375	0.25	0.37	0.0625	0.37	NO	10
0.125	0.125	0.25	0.81	0.0625	0.81	YES	11
0.500	-0.500	1.00	0.32	1.00	0.32	NO	12

Table 8. Continued.

X coordinate	Y coordinate	Size of region	$\lvert\nabla\phi\rvert$ (f)	Area of region (A)	Mean = fA / A	Further sub-division needed? YES / NO	Region Number
0.375	0.375	0.25	0.37	0.0625	0.37	NO	13
0.375	0.125	0.25	0.37	0.0625	0.37	NO	14
0.750	0.750	0.50	0.36	0.25	0.36	NO	15
0.750	0.250	0.50	0.38	0.25	0.38	NO	16

From the tabular column above, we can see that the mean of region 11 exceeds the mean specified and the size of the region is still greater than the minimum size allowed. As a result, region 11 will get sub-divided. The discretized domain is shown in Fig. 52.

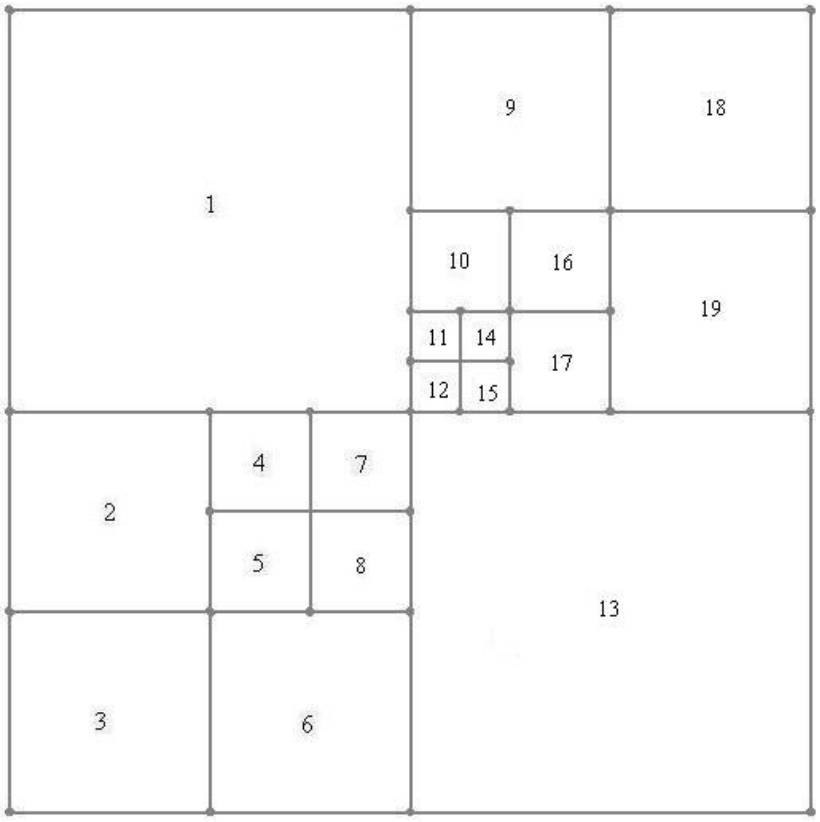


Fig. 52. Domain: Stage-4.

The details pertaining to this domain have been listed in Table 9. It is interesting to note that none of these regions have been found to be requiring decomposition.

Table 9. Tabulated Results from Stage-4 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	$\lvert\nabla\phi\rvert$ (f)	Area of region (A)	Mean = fA / A	Further sub-division needed? YES / NO	Region Number
-0.5000	0.5000	1.00	0.32	1.00	0.32	NO	1
-0.7500	-0.2500	0.50	0.38	0.25	0.38	NO	2
-0.7500	-0.7500	0.50	0.36	0.25	0.36	NO	3
-0.3750	-0.1250	0.25	0.37	0.0625	0.37	NO	4
-0.3750	-0.3750	0.25	0.37	0.0625	0.37	NO	5
-0.2500	-0.7500	0.50	0.38	0.25	0.38	NO	6
-0.1250	-0.1250	0.25	0.37	0.0625	0.37	NO	7
-0.1250	-0.3750	0.25	0.37	0.0625	0.37	NO	8
0.2500	0.7500	0.50	0.38	0.25	0.38	NO	9
0.1250	0.3750	0.25	0.37	0.0625	0.37	NO	10
0.0625	0.1875	0.125	0.38	0.015625	0.38	NO	11
0.0625	0.0625	0.125	2.1	0.015625	2.1	NO	12

Table 9. Continued.

X coordinate	Y coordinate	Size of region	$\lvert\nabla\phi\rvert$ (f)	Area of region (A)	Mean = fA / A	Further sub-division needed? YES / NO	Region Number
0.5000	-0.5000	1.00	0.32	1.00	0.32	NO	13
0.1875	0.1875	0.125	0.38	0.015625	0.38	NO	14
0.1875	0.0625	0.125	0.38	0.015625	0.38	NO	15
0.3750	0.3750	0.25	0.37	0.0625	0.37	NO	16
0.3750	0.1250	0.25	0.37	0.0625	0.37	NO	17
0.7500	0.7500	0.50	0.07	0.25	0.28	NO	18
0.7500	0.2500	0.50	0.09	0.25	0.36	NO	19

In this case, it can be seen from the tabular column that the mean of region 12 far exceeds the value specified for quadtree decomposition. However, region 12 has reached the minimum allowed size of the regions. As a result, it shall not get discretized any further.

After the domain has been discretized to an extent where no region in the domain satisfies the two criterions simultaneously any longer, regions of highly varying sizes can be found. However, abrupt transitions in the mesh are to be avoided. As a result, further domain discretization is desired. This is accomplished in this work by introducing another bunch of criterions.

As an example, let the domain which is desired to be discretized be the one obtained from Stage-4. Now, sub-division of a region should occur if both of the following conditions are satisfied:

1. Number of neighbors of any region along any of its edges > 2
2. Size of region > 0.125

With a change in the set of criterions, a different basis of numbering has also been introduced. Keeping this in mind, it is noted that Fig.53 is a modified form of the domain shown in Fig.52. Now, coming to the basis of numbering for this set of criterions, it is evident that in Stage-5, in addition to the previous basis, the regions have been numbered in the order of ascending order of ‘region sizes’.

Coming to why this has been done, let us consider Fig. 52. In this, the algorithm accesses region 2 before region 4. Let us suppose region 2 fails to satisfy the criterion set but region 4 satisfies the same. As a result, region 4 will get sub-divided and leave region 2 with three neighbors along its right edge. To avoid this, we access the regions in the order of increasing sizes. As a result, no regions other than the smallest sized ones get accessed until all the smaller sized regions have been sufficiently discretized.

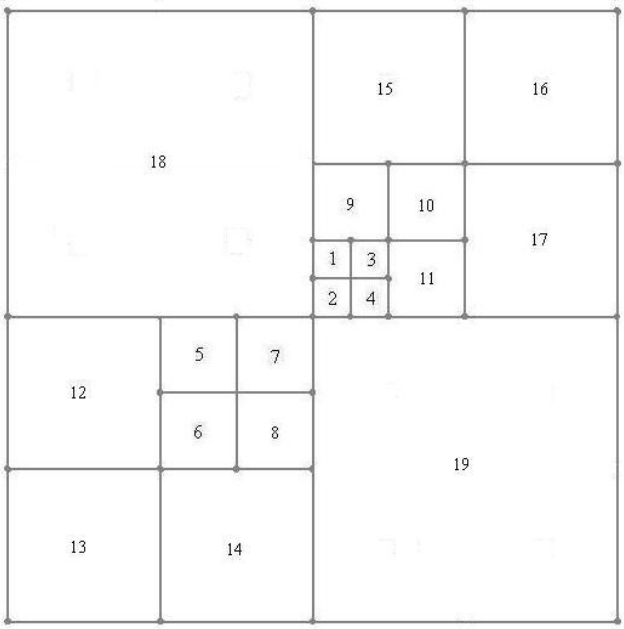


Fig. 53. Domain: Stage-5.

The relevant details can be found in Table 10. The number of neighbors for each region along each of its edges has been listed.

Table 10. Tabulated Results from Stage-5 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
0.0625	0.1875	0.125	1	1	1	1	NO	1
0.0625	0.0625	0.125	1	1	1	1	NO	2
0.1875	0.1875	0.125	1	1	1	1	NO	3
0.1875	0.0625	0.125	1	1	1	1	NO	4
-0.3750	-0.1250	0.25	1	1	1	1	NO	5
-0.3750	-0.3750	0.25	1	1	1	1	NO	6
-0.1250	-0.1250	0.25	1	1	1	1	NO	7
-0.1250	-0.3750	0.25	1	1	1	1	NO	8
0.1250	0.3750	0.25	1	1	2	1	NO	9
0.3750	0.3750	0.25	1	1	1	1	NO	10
0.3750	0.1250	0.25	2	1	1	1	NO	11

Table 10. Continued.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
-0.7500	-0.2500	0.50	0	2	1	1	NO	12
-0.7500	-0.7500	0.50	0	1	0	1	NO	13
-0.2500	-0.7500	0.50	1	1	0	2	NO	14
0.2500	0.7500	0.50	1	1	2	0	NO	15
0.7500	0.7500	0.50	1	0	1	0	NO	16
0.7500	0.2500	0.50	2	0	1	1	NO	17
-0.5000	0.5000	1.00	0	4	3	0	YES	18
0.5000	-0.5000	1.00	3	0	0	4	YES	19

From the tabular column, it can be seen that region 18 has 4 neighbors along its right edge and 3 along its bottom edge. Similarly, region 19 has 3 along its left edge and 4 along its top edge. There are no other regions having more than two neighbors along any of their regions. As a result, regions 18 and 19 have been identified as the ones requiring decomposition and the discretized domain has been shown in Fig. 54.

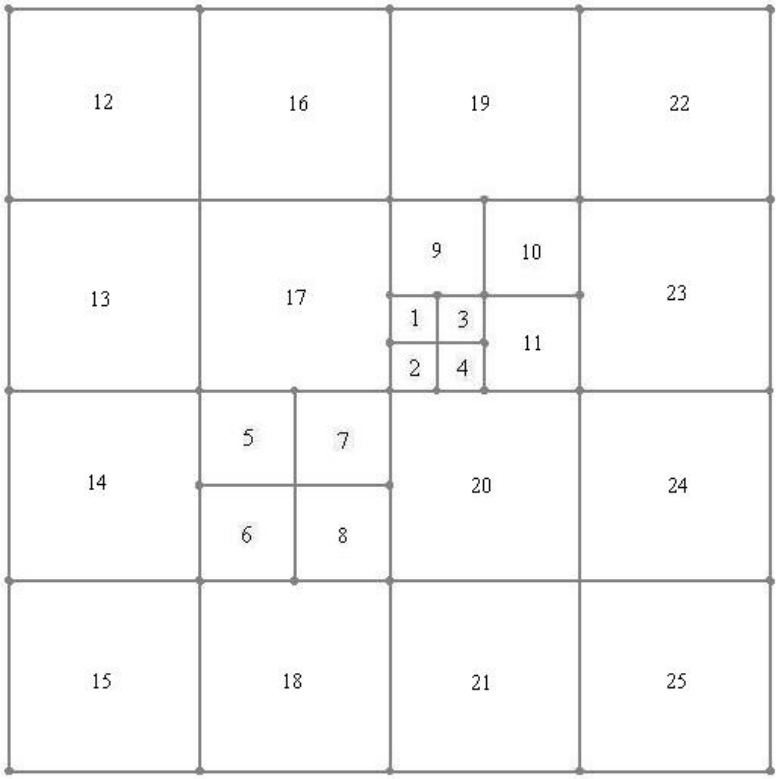


Fig. 54. Domain: Stage-6.

Table-11 shows the related details.

Table 11. Tabulated Results from Stage-6 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
0.0625	0.1875	0.125	1	1	1	1	NO	1
0.0625	0.0625	0.125	1	1	1	1	NO	2
0.1875	0.1875	0.125	1	1	1	1	NO	3
0.1875	0.0625	0.125	1	1	1	1	NO	4
-0.3750	-0.1250	0.25	1	1	1	1	NO	5
-0.3750	-0.3750	0.25	1	1	1	1	NO	6
-0.1250	-0.1250	0.25	1	1	1	1	NO	7
-0.1250	-0.3750	0.25	1	1	1	1	NO	8
0.1250	0.3750	0.25	1	1	2	1	NO	9
0.3750	0.3750	0.25	1	1	1	1	NO	10
0.3750	0.1250	0.25	2	1	1	1	NO	11
-0.7500	0.7500	0.50	0	1	1	0	NO	12

Table 11. Continued.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
-0.7500	0.2500	0.50	0	1	1	1	NO	13
-0.7500	-0.2500	0.50	0	2	1	1	NO	14
-0.7500	-0.7500	0.50	0	1	0	1	NO	15
-0.2500	0.7500	0.50	1	1	1	0	NO	16
-0.2500	0.2500	0.50	1	3	2	1	YES	17
-0.2500	-0.7500	0.50	1	1	0	2	NO	18
0.2500	0.7500	0.50	1	1	2	0	NO	19
0.2500	-0.2500	0.50	2	1	1	3	YES	20
0.2500	-0.7500	0.50	1	1	0	1	NO	21
0.7500	0.7500	0.50	1	0	1	0	NO	22
0.7500	0.2500	0.50	2	0	1	1	NO	23
0.7500	-0.2500	0.50	1	0	1	1	NO	24

Table 11. Continued.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
0.7500	-0.7500	0.50	1	0	0	1	NO	25

Regions 17 and 20 are identified as the ones requiring further decomposition. The domain after quadtree decomposition is shown in Fig. 55.

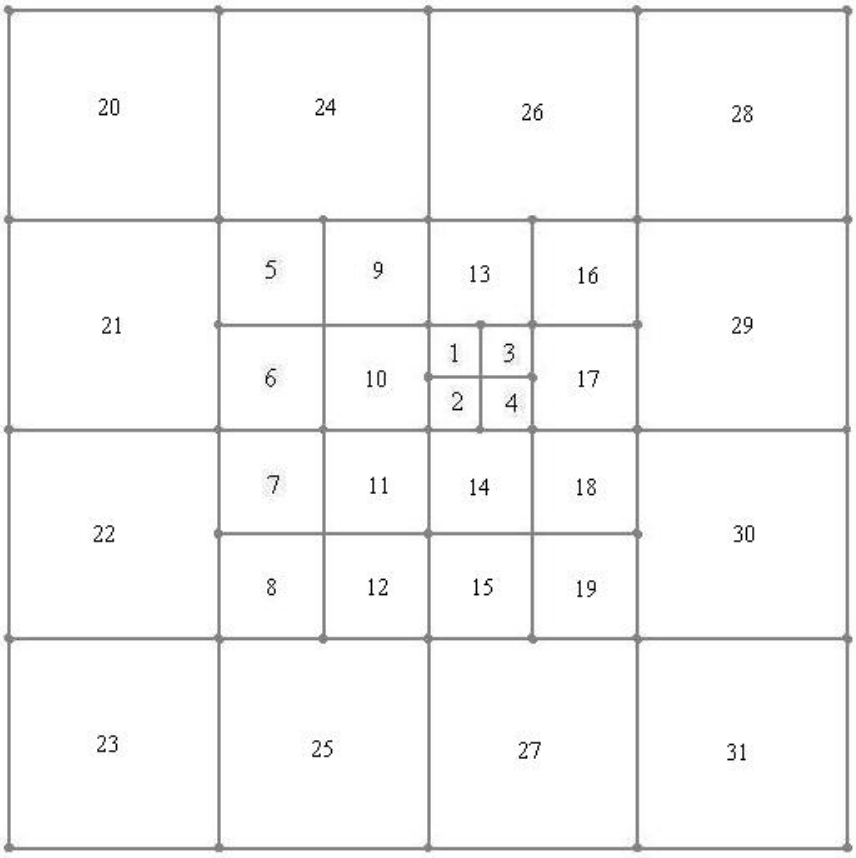


Fig. 55. Domain: Stage-7.

Table 12. Tabulated Results from Stage-7 of quadtree decomposition.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
0.0625	0.1875	0.125	1	1	1	1	NO	1
0.0625	0.0625	0.125	1	1	1	1	NO	2
0.1875	0.1875	0.125	1	1	1	1	NO	3
0.1875	0.0625	0.125	1	1	1	1	NO	4
-0.3750	0.3750	0.25	1	1	1	1	NO	5
-0.3750	0.1250	0.25	1	1	1	1	NO	6
-0.3750	-0.1250	0.25	1	1	1	1	NO	7
-0.3750	-0.3750	0.25	1	1	1	1	NO	8
-0.1250	0.3750	0.25	1	1	1	1	NO	9
-0.1250	0.1250	0.25	1	2	1	1	NO	10
-0.1250	-0.1250	0.25	1	1	1	1	NO	11
-0.1250	-0.3750	0.25	1	1	1	1	NO	12

Table 12. Continued.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
0.1250	0.3750	0.25	1	1	2	1	NO	13
0.1250	-0.1250	0.25	1	1	1	2	NO	14
0.1250	-0.3750	0.25	1	1	1	1	NO	15
0.3750	0.3750	0.25	1	1	1	1	NO	16
0.3750	0.1250	0.25	2	1	1	1	NO	17
0.3750	-0.1250	0.25	1	1	1	1	NO	18
0.3750	-0.3750	0.25	1	1	1	1	NO	19
-0.7500	0.7500	0.50	0	1	1	0	NO	20
-0.7500	0.2500	0.50	0	1	1	1	NO	21
-0.7500	-0.2500	0.50	0	2	1	1	NO	22
-0.7500	-0.7500	0.50	0	1	0	1	NO	23
-0.2500	0.7500	0.50	1	1	1	0	NO	24

Table 12. Continued.

X coordinate	Y coordinate	Size of region	No. of left neighbors	No. of right neighbors	No. of bottom neighbors	No. of top neighbors	Further sub-division required YES / NO	Region Number
-0.2500	-0.7500	0.50	1	1	0	2	NO	25
0.2500	0.7500	0.50	1	1	2	0	NO	26
0.2500	-0.7500	0.50	1	1	0	1	NO	27
0.7500	0.7500	0.50	1	0	1	0	NO	28
0.7500	0.2500	0.50	2	0	1	1	NO	29
0.7500	-0.2500	0.50	1	0	1	1	NO	30
0.7500	-0.7500	0.50	1	0	0	1	NO	31

After the domain has been sufficiently discretized by using the second set of criterions, the neighboring regions of each region along each of its edges are identified. This information is both important and useful during the application of the locally consistent finite difference scheme discussed in Sec 3.3.

Since the regions may have either one or two neighbors along their edges, provision is made to store the relevant information. The neighbors are abbreviated in the following way in Table 13 on the next page:

1. LN-1 First Left Neighbor
2. LN-2 Second Left Neighbor
3. RN-1 First Right Neighbor
4. RN-2 Second Right Neighbor
5. BN-1 First Bottom Neighbor
6. BN-2 Second Bottom Neighbor
7. TN-1 First Top Neighbor
8. TN-2 Second Top Neighbor

Table 13. Information about regions and the corresponding neighbors.

X coordinate	Y coordinate	Size of region	LN-1	LN-2	RN-1	RN-2	BN-1	BN-2	TN-1	TN-2	Region Number
0.0625	0.1875	0.125	10	0	3	0	2	0	13	0	1
0.0625	0.0625	0.125	10	0	4	0	14	0	1	0	2
0.1875	0.1875	0.125	1	0	17	0	4	0	13	0	3
0.1875	0.0625	0.125	2	0	17	0	14	0	3	0	4
-0.3750	0.3750	0.25	21	0	9	0	6	0	24	0	5
-0.3750	0.1250	0.25	21	0	10	0	7	0	5	0	6
-0.3750	-0.1250	0.25	22	0	11	0	8	0	6	0	7
-0.3750	-0.3750	0.25	22	0	12	0	25	0	7	0	8
-0.1250	0.3750	0.25	5	0	13	0	10	0	24	0	9
-0.1250	0.1250	0.25	6	0	1	2	11	0	9	0	10
-0.1250	-0.1250	0.25	7	0	14	0	12	0	10	0	11
-0.1250	-0.3750	0.25	8	0	15	0	25	0	11	0	12

Table 13. Continued.

0.1250	0.3750	0.25	9	0	16	0	1	3	26	0	13
0.1250	-0.1250	0.25	11	0	18	0	15	0	2	4	14
0.1250	-0.3750	0.25	12	0	19	0	27	0	14	0	15
0.3750	0.3750	0.25	13	0	29	0	17	0	26	0	16
0.3750	0.1250	0.25	3	4	29	0	18	0	16	0	17
0.3750	-0.1250	0.25	14	0	30	0	19	0	17	0	18
0.3750	-0.3750	0.25	15	0	30	0	27	0	18	0	19
-0.7500	0.7500	0.50	0	0	24	0	21	0	0	0	20
-0.7500	0.2500	0.50	0	0	5	6	22	0	20	0	21
-0.7500	-0.2500	0.50	0	0	7	8	23	0	21	0	22
-0.7500	-0.7500	0.50	0	0	25	0	0	0	22	0	23
-0.2500	0.7500	0.50	20	0	26	0	5	9	0	0	24
-0.2500	-0.7500	0.50	23	0	27	0	0	0	8	12	25

Table 13. Continued.

0.2500	0.7500	0.50	24	0	28	0	13	16	0	0	26
0.2500	-0.7500	0.50	25	0	31	0	0	0	30	0	27
0.7500	0.7500	0.50	26	0	28	0	29	0	0	0	28
0.7500	0.2500	0.50	16	17	0	0	30	0	28	0	29
0.7500	-0.2500	0.50	18	19	0	0	31	0	29	0	30
0.7500	-0.7500	0.50	27	0	0	0	0	0	30	0	31

VITA

Name: Nandagopalan Auviur Srinivasa

Address: 2-2-647/287/1, Srinivasa Nagar,
Plot-18, Bagh Amberpet,
Hyderabad – 500013, India

Education:

2004-2006 Texas A&M University
Master of Science
Mechanical Engineering, 2006.

2000-2004 D.V.R. College of Engineering and Technology
Jawaharlal Nehru Technological University,
Hyderabad, India.
Bachelor of Technology,
Mechanical Engineering, 2004.