Boundary Aligned Grids for Flow in Porous Media

A theoretical study and practical approach using MATLAB

Mary Abraham Eranackal

Student Number: 530137

Erasmus Mundus Masters Research Project

June 2010

"Project Dissertation submitted to Swansea University in Partial Fulfilment for the Degree of Master of Science"

Supervisor:

Prof. Michael G Edwards

BSc, PhD

School of Engineering, Swansea

MSc Computational Mechanics, Department of Civil and Computational Engineering, Swansea University

DECLARATION

This work has not previously been accepted in substance for any degree and is not being currently submitted in candidature for any degree.

STATEMENT 1

This dissertation is being submitted in partial fulfilment of the requirements for the degree of MSc.

Date: .....Signature: .....

STATEMENT 2

This dissertation is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by giving explicit references. A bibliography is appended.

Date: .....Signature: .....

STATEMENT 3

I hereby give consent for my dissertation, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Date: .....Signature: .....

ACKNOWLEDGEMENTS

During the academic year of 2009-2010, I had the opportunity to work on this project which goes towards my MSc research project fulfilment. I would like to take this opportunity to thank my project supervisor Prof. Michael G Edwards for firstly, offering such an interesting and challenging topic in reservoir modelling for research work. I thank him for all the time he spent helping me understand the nuances of each and every topic and for his continued support throughout the period. It has been an enriching and fruitful time that I got to spend on this project work which has taught me a lot about a subject which I have previously not been much informed about. I thank him sincerely for all the motivation and inspiration in challenging me to better myself during this project.

I also thank my course supervisor Dr. Antonio J. Gil for all the advice and support he has given during this time. I would also like to thank my senior Hongwen Zheng for his help and suggestions. I thank my family and friends for all their support in various ways.

I sincerely hope the work I carried out, which is detailed in this report, will do its own small bit towards this exciting field full of possibilities and challenges.

.....

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

v_h	Discrete Darcy velocity
Ω	Domain
$\partial\Omega$	Domain boundary
F	Flux
q	Flux continuity parameter
Q	Forcing term or Flow rate
T	General tensor defined via the Piola transform
∇	Gradient operator
J	Jacobian
dL_A	Outward normal vector to edge containing point A
K	Permeability tensor
ϕ	Pressure
(ξ,η)	Transform space coordinates

LIST OF ABBREVIATIONS

FDM	Finite Difference Method
FEM	Finite Element Method
FPS	Full Pressure Support
FVM	Finite Volume Method
MFEM	Mixed Finite Element Method
TPS	Triangular Pressure Support

Summary

With new petroleum reservoirs becoming rarer in the past few decades, the necessity to extract oil from the existing ones more efficiently has never before been more important. This work introduces the reader to the basic concepts of reservoirs and summarises the currently existing numerical techniques in reservoir modelling process that were researched during the project. The existing discretization techniques for modelling Darcy flow in porous media were researched extensively leading up to the most recent advancements and suggested techniques in this field. The topic in interest, that is, utilising cell centred finite volume discretization of the reservoir, is then taken up and explained in depth. The advantages and disadvantages of this scheme are critically evaluated in this thesis.

This work concentrates on development of a MATLAB application code for the physical space formulation of flux-continuous, full tensor finite volume schemes on unstructured cell-centered triangular grids. The results, typically convergence rates, show a close conformance with the expected values. The examples highlighted in this report have been chosen so for their specific types and actual application similarity. The results obtained show that the scheme works best when the symmetric approximation with the flux continuity parameter $q=2/3$ is used as opposed to other non-symmetric approximations ($q\neq2/3$, for e.g. $q=1$). Comparing the L_2 norms of pressure obtained for grids completely aligned with L_2 norms for pressure obtained for grids non-aligned to the permeability discontinuities in the domain, it is seen that the aligned grids give a more accurate representation of the reservoir, hence confirming the need for boundary aligned grids. Evaluation of a strongly discontinuous full tensor field with the presence of a source or a sink in the domain is shown to be most accurate when $q=2/3$ for the scheme as opposed to other non-symmetric approximations.

Scope of this project is limited to the triangular pressure support scheme (TPS). Further work can be done by developing the MATLAB application code for full pressure support scheme (FPS) using the currently developed TPS code. The developed code can also be combined with the existing cell-vertex based application code developed in FORTRAN [67] to create a simple application code for a hybrid gridding method which is currently showing promise in the field of reservoir simulation.

Chapter 1 Introduction

In nature some of the common porous media one encounters are the rocks, biological tissues etc. Porous media can be simply defined as a solid media with an interconnected network of pores which are filled with fluid (gaseous or liquid). The ability of porous media to contain fluid is often exploited in nature, for example, the living creature Sponge, which mainly depends on efficient cross flow of water through its body, maximises the efficiency of water flow using porous body structure. Also, fossil fuels are found in porous rocks buried deep within the earth. It is of economic importance to understand the flow of fluid through the pores of a solid material for a variety of reasons such as for developing oil recovery strategies, assessment of aquifer remediation strategies and carbon sequestration strategies. This research work concentrates on the science of flow in porous media with specific application to petroleum reservoirs. This chapter introduces the reader to the physical aspects of petroleum reservoirs and also gives a brief introduction to reservoir modelling and its objectives.

1.1 Petroleum Reservoirs

Formation

Petroleum reservoirs are hydrocarbon pools, located about 1,000 to 30,000 feet beneath the earth's surface, in porous rock structures. They may extend over hundreds of kilometres and vary in its composition of the rock, type of fuel content etc. The formation of these reservoirs can be succinctly told to be as a result of deep burial of plankton and algae matter under sand and mud followed by pressure cooking and hydrocarbon migration from the source to the reservoir rock and finally being trapped by impermeable rock. Figure 1.1 shows the typical structure of a reservoir.

The structure shown clearly depicts the reservoir abutting a salt dome, which has trapped a layer of oil and natural gas between itself and nonporous rock. A simplified outlook of the same is depicted in figure 1.2. The lower layer is usually the source rock rich in hydrocarbons. The oil which has migrated and filled the porous rock over this layer forms the oil reservoir. The uppermost part of this layer contains the lesser dense natural gas. This layer is topped by an impervious rock which traps the oil in its porous reservoir.

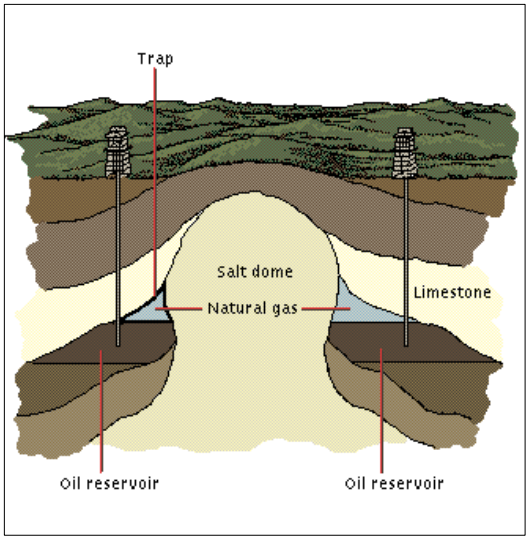


Figure 1.1 A sample reservoir physical model*

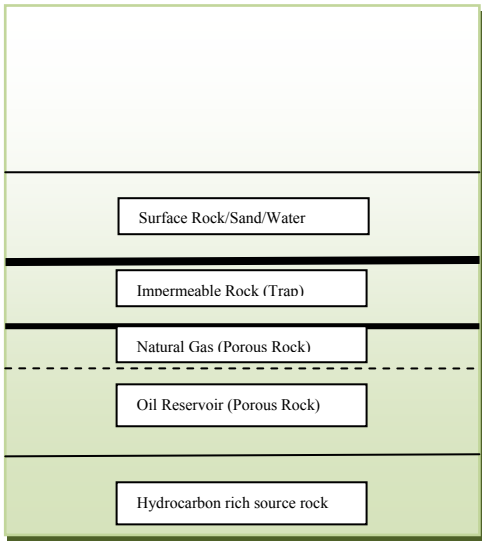


Figure 1.2 Simplified reservoir model

Because they have no place to expand, the gas and crude oil are under high pressure when trapped under the impervious rock.

Exploration

Search for petroleum reservoirs, by geologists, is done by using a number of different methods to confirm the presence of the reservoir. Mainly, seismic surveys are used to develop maps of sub-surface rock layers which can be distinguished by their density difference. By extensive surface and sub-surface surveys geologists can localise the regions where there is a high probability of reservoir existence. Once this is done, exploratory wells are dug to physically check for the presence of oil at the region. The exploratory wells dug also helps in confirming the true nature of the sub-surface features and quality of the fuel present etc [7].

Primary Production

Since the oil and natural gas trapped under the impermeable rock is already pressurised, when a well bore is drilled into the reservoir porous rock, the oil and natural gas expands into the low-pressure sink created by the well bore and will steadily move upwards to the earth's surface. This process continues on until the pressure differential exists.

*Source:http://www.emt-india.net/process/petrochemical/Petroleum.htm

Enhanced Oil Recovery

Once the pressure differential in the reservoir ceases to support the flow of the oil through the well bore to the surface, water or steam is injected into the reservoir to simultaneously, maintain the pressure differential by increasing the reservoir pressure and also to displace the oil and in case of steam, it further reduces the viscosity of oil favourably.

1.2 Reservoir Modelling

"Reservoir modelling involves the construction of a computer model of a petroleum reservoir, for the purposes of improving estimation of reserves and making decisions regarding the development of the field. A reservoir model represents the physical space of the reservoir by an array of discrete cells, delineated by a grid which may be regular or irregular." [60]

Reservoir modelling maybe done during the exploration for providing geological description of the reservoir prior to the production start and again during the production to evaluate flow properties of the oil within the reservoir. The process of reservoir modelling to determine the fuel flow properties within the porous media of the reservoir using numerical methods is specifically termed as reservoir simulation.

Advantages and Disadvantages of Reservoir Simulation

Reservoir simulation possesses the following apparent advantages;

- Simple and complex problems that otherwise cannot be solved can be easily solved.
- Reduces the overall exploration and reservoir management costs.
- Helps in deciding the optimum positioning of wells.
- Helps in analysing the effect of changed variables over the production period.
- Ease of updating the input variables when changes are found during actual production process helps in real-time production scheduling [13].

Cost of study, time required to do the study and the amount of input data required are the main disadvantages of reservoir simulation.
A typical reservoir simulation study consists of the following major steps [43]:

- Problem definition – reservoir performance problem and associated operating problem are to be defined clearly.

- Data review – data needed to construct the model must be reviewed and reorganised once they are collected.
- Data acquisition – continuous collection and updating of required data.
- Selection of approach – selecting the appropriate simulator for the defined problem.
- Reservoir description and model design – describing the region of reservoir to be modelled based on the study objective and designing the model.
- Programming support – editing the simulator program as per specific requirements and analysis of the obtained results.
- History matching – validating the obtained results by comparing it with historical production, injection data and actual reservoir performance.
- Prediction – predicting the future performance of the field.
- Reporting – assembling results and conclusion in a report.

Reservoir simulation can be tailored to each need, budget time frame etc. It is an important tool in efficient oil recovery and reservoir management now.

1.3 MATLAB

MATLAB is a high-level computer language for scientific computing and data visualization built around an interactive programming environment [37]. MATLAB was created in the late 1970s by Cleve Moler and is currently developed and sold by The MathWorks which was founded in 1984. In academia, MATLAB has gradually taken over most of the scientific programming work with its interactive easy to use features. It allows both "programming in the small" to rapidly create quick and dirty throw-away programs, and "programming in the large" to create large and complex application programs [42].

MATLAB, being an interactive tool, allows users to debug the codes faster and lets the user concentrate on the principles being coded rather than the code programming. MATLAB allows users to call predefined functions to carry out a number of tasks, for e.g., MATLAB allows users to access the software libraries LINPACK and EISPACK that are written in FORTRAN for numerical computation. Another distinctive advantage of MATLAB is that since all numerical objects are treated as double-precision arrays, there is no need to declare data types and carry out type conversions [32]. MATLAB not only provides the user with a number of solvers, for e.g. fixed-step, variable-step, continuous, discrete etc, it also provides users with a number of graphical output options.

Syntax of MATLAB is quite similar to FORTRAN. In fact, MATLAB allows users to run the codes written C and FORTRAN within its environment. This feature is often useful in developing on already written codes in FORTRAN or C using MATLAB.MATLAB also possesses the ability to carry out symbolic computation using the kernel of Maple.
The user stories in the MATLAB official product site shows an indication of how MATLAB is being gradually accepted in and used in various industries as well. It is being widely recognised for its easy interface and time saving features. Reservoir Modelling is also being carried out using MATLAB now [57].

1.4 Scope of Work

The objective of this research work is to understand the current discretization techniques for modelling Darcy flow in porous media and to develop a code to investigate the boundary aligned grid generation. The aim is to investigate and extend the current range of existing methods to more general subsurface reservoir features such as faults and fractures using cell-centered finite volume formulation.
The programming language used to code the boundary aligned scheme is MATLAB. The scheme coded for is triangular pressure support (TPS), physical space formulation of flux continuous full-tensor finite volume scheme on unstructured cell-centered triangular grids. A number of application examples have been solved. The results obtained have been compared with the sub-cell tensor formulation for cell centered finite volume scheme as well as cell vertex based TPS schemes found in [16], [19], [67].

1.5 Organisation of the Report

Chapter 2 gives an overview of the existing research work on numerical reservoir simulation, tracing it through its history to the most recent developments. The different numerical methods employed in Reservoir simulation are also briefly discussed in Chapter 2. Chapter 3 gives a brief mathematical description of the problem as well the fundamentals of grid definition. Chapter 4 focuses on the Physical Space formulation of TPS cell centered scheme on unstructured triangular grids. Description of the scheme, simple application code algorithm and computational examples with results are presented in this chapter. Chapter 5 summarises the work done in this research giving concluding remarks and also outlines the future scope of work. The application codes created are presented in the Appendix.

Chapter 2 Literature Review and Overview of Current Discretization Methods

2.1 Literature review of reservoir modelling methods

The potential of numerical reservoir simulation was recognised in the late 1940's and early 1950's by a number of companies. After the early research and development, crude, but useful simulators were available by 1950's [43]. Reliability of the reservoir simulation is an important aspect in its continued and developing use within the petroleum industry. The focus has forever been on developing more accurate and efficient forms of numerical simulation. When it comes to large-scale real life reservoir modelling it is necessary to ensure that the simulator used is cost effective for its objective as well as produce highly accurate results.

The partial differential equations that describe fluid flow in a reservoir cannot be solved analytically hence we use the reservoir simulator to solve these equations numerically. In order to carry out numerical evaluation of these partial differential equations, it is necessary to consider the reservoir region as a composition of discrete volume elements. The precision with which the reservoir can be described in a model and the accuracy with which the flow of reservoirs can be calculated will depend on the number and type of these discrete volume elements. [43], [24].

The faults and fractures often present in a reservoir are difficult to model using the uniform structured grids. Discretization of a region containing internal fractures (constraints) or sudden large variations in permeability etc, using uniform structured grids might lead to variation of rock properties within each element, which will lead to poorer accuracy. It is recognised that the discretization of the field requires some organisation for the numerical solution thereon to be efficient, i.e., it must be possible to readily identify the cells neighbouring the computation sites, also it must conform to the boundaries of the region such that the boundary conditions are accurately represented.[59]

Research over the years [1], [2], [14], [28], [30], [31] has shown it beneficial to use unstructured grids for treating discontinuous and anisotropic permeability fields. Finite volume schemes such as control volume distributed (CVD) and multi-point flux approximation (MPFA) [16], [15] have gained popularity in recent years. The main focus of this study, i.e., symmetric positive definite (SPD) flux continuous, full tensor, finite volume

schemes on unstructured cell-centered triangular girds is explained in detail in [19]. The paper details the triangular pressure support scheme for general unstructured grids in 2D for both physical space and sub-cell transform space formulation. The cell centered TPS scheme on 2D quadrilateral grids for full tensor pressure equation is discussed in [17]. The sample application runs to obtain convergence rates have been based on the examples from [17], [16], [19] and [67].

Whereas the TPS schemes are only point-wise continuous in pressure and flux, the need for pressure and flux continuity over full control volume sub-faces is realised using the Full Pressure Support (FPS) scheme. The family of full pressure support scheme was first explained in detail in [17]. These schemes have full pressure continuity imposed across control-volume faces, in contrast to the earlier families of flux-continuous schemes with point-wise continuity in pressure and flux.

This study focuses on the physical space formulation of the triangular pressure support finite volume scheme on cell-centered, unstructured triangular grids. The physical space formulation was developed by Edwards et al. [19] as a member of the sub-cell transform space family of schemes.

2.2 Numerical Discretization Methods

Numerical discretization methods refer to the methods adopted to solve complex continuous mathematical problems by creating discrete models of the problem and generating a system of algebraic equations which are solved to obtain the solution for the initially considered complex problem. There are a number of ways in which the discretization of the continuous model can be. Based on the differences in the discretization the different types of numerical analysis are:

2.2.1 Finite Difference Method

The finite difference method (FDM) was first developed by A. Thom in the 1920s under the title “the method of square” to solve nonlinear hydrodynamic equations [58]. FDM involves, dividing the problem domain into grids containing nodes and approximating the given differential equation by its finite difference equivalence at grid points. The difference equations so obtained over the domain is then subjected to the prescribed boundary conditions and/or initial conditions and solved to obtain the approximate solution over the

domain. Taylor series expansions are used to derive the finite difference equations for approximating the derivatives [25]. Whereas FDM is simple and effective as well as easy to derive, it is also limited to structured meshes.

Though FDM has been used widely in conventional reservoir simulation [51], [23], [53] its inability to accurately solve for irregular boundaries and internal constraints [39] is a major drawback since reservoir simulation usually consists of such complexities.

2.2.2 Finite Element Method (FEM)

Finite Element Method involves dividing the problem domain into a number of small, simple elements containing nodes at the connecting points between these defined elements. A field quantity is then interpolated over each element, i.e. piecewise polynomial interpolation is carried out. At the nodes, the adjacent elements share the degree of freedom and the field quantity thus becomes interpolated over the entire structure in piecewise fashion. This results in a set of simultaneous algebraic equations at the nodes which can be solved to obtain the unknown variables. Since the early 1960s, engineers used the method for approximate solutions of problems in stress analysis, fluid flow, heat transfer, and other areas. The most widely used weighted residual for the finite element method is the Galerkin Method.
FEM possesses a number of advantages over the FDM. It can readily handle the complex geometries and restraints as well as complex loading. FEM, for these reasons began to be widely used in reservoir simulation [41], [9], [26], [68]. Unfortunately, standard FEM lacks local flux continuity which is essential in reservoir simulation due to sudden changes in rock properties [67], [40]. Though still widely used in reservoir simulation, research focus is gradually being shifted to mixed finite element method and finite volume methods due to their ability to describe the reservoir model more accurately.

2.2.3 Mixed Finite Element Method (MFEM)

Finite Element Methods in which two spaces are used to approximate two different variables are called the mixed finite element methods [4]. In case of flow in porous media, the two variables will be pressure and velocity. In the classical Finite element method only potential is taken as the primary variable and velocity is obtained via a post-processing procedure using an approximation of Darcy's Law. This direct approach leads to lower-order approximations for velocity compared to potential and, additionally, the corresponding balance equation is satisfied in an extremely weak sense [12].

Mixed finite element methods while having the property to simultaneously evaluate both pressure and velocity, also possesses the advantages of the classical finite element methods in complex geometry description. Mixed finite element methods have found use in solving Darcy flow, i.e., use in reservoir simulation in the recent years [9], [10]. Extensive research work, [15], [26], [41], [45], has been carried out to adapt the favourable properties of this method to provide better simulation models.

2.2.4 Finite Volume Method (FVM)

The finite volume method is a numerical method for solving partial differential equations that calculates the values of the conserved variables averaged across a volume. The finite volume method is a discretization method which is well suited for the numerical simulation of various types (elliptic, parabolic or hyperbolic, for instance) of conservation laws; it has been extensively used in several engineering fields, such as fluid mechanics, heat and mass transfer or petroleum engineering [54].FVM can easily be used for unstructured meshes and hence description of geometry even in complex cases does not pose a difficulty. FVM is a locally conservative method. The numerical flux is conserved from one discretization cell to its neighbour. The finite volume method is locally conservative because it is based on a "balance" approach: a local balance is written on each discretization cell which is often called "control volume"; by the divergence formula, an integral formulation of the fluxes over the boundary of the control volume is then obtained. The fluxes on the boundary are discretized with respect to the discrete unknowns [54].

The FVM, which returns to the balance equation form of the equations, where one level of spatial derivatives is removed, is the method of choice; always for the pressure equation and nearly always for the saturation equation. Commercial reservoir simulators are, with the exception of streamline simulators, entirely based on the finite volume method [34]. Compared to the MFEM, FVM is computationally cheaper [67]. FVM has been developed in detail over the years considering its definite advantages in reservoir simulation over other methods, for e.g., [8], [27], [66], [61], [22], [39], [55] are some of the work which helped develop and support the current FVM formulation researched in this report.

Chapter 3 Problem Definition and Fundamentals of Grid Definition

3.1 Introduction

The ability to predict the behaviour of the reservoirs depends on how well the flow characteristics of the fluids in the reservoir are defined. Basic equations of flow in porous media for single phase flow and a brief overview of concepts like permeability, flow potential etc as well as the fundamentals of grid types are detailed in this chapter. This chapter gives a brief description of the pressure equation formulation used in the TPS physical space formulation.

3.2 Flow in Porous Media

As described in Chapter 1, porous media is a solid media with an interconnected network of pores which are filled with fluid. The measure of the ability of the porous media to transmit fluids is called *permeability*. The flow of fluid in a porous medium is defined by the Darcy's Law [3]. Darcy's Law gives a constitutive equation determined experimentally by Henry Darcy (1856) for flow of water through beds of sand. It is defined as:

$$Q = -\frac{k}{\mu}\nabla P \tag{3.1}$$

where, k is the permeability, μ is the fluid viscosity and ∇P is the pressure gradient vector. Q is the discharge volume per unit area and has dimensions of velocity (L/T). This is the fundamental equation used to solve for pressure in a reservoir. Equation (3.1) holds true for homogenous, single-phase fluids having laminar flow.

The primary objective of simulation is to numerically determine fluid pressure and velocity distribution in the reservoir. Therefore the problem is to find the pressure ϕ satisfying

$$\int_{\Omega} -\nabla\cdot(K\nabla\phi)d\tau = M \tag{3.2}$$

where, Ω : Domain in consideration

M : Specified flow rate

$$K = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \quad : \quad \text{Permeability tensor}$$

$$\nabla \quad : \quad \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)$$

The permeability tensor K is a symmetric full tensor in general (i.e. $k_{12} = k_{21}$). Since the pressure equation considered is elliptic, it follows that $k_{12}^2 \le k_{11} k_{22}$.

We can define the same problem in a general curvilinear coordinate system that is defined with respect to a uniform dimensionless transform space coordinates (ξ, η). Equation (3.2) then becomes

$$-\int_{\Omega} \tilde{\nabla} \cdot \left(T \tilde{\nabla} \phi \right) d\tilde{\tau} = M \tag{3.3}$$

where, Ω is the arbitrary control volume comprised of surfaces that are tangent to constant (ξ, η). Here, $\tilde{\nabla} = \left(\frac{\partial}{\partial \xi}, \frac{\partial}{\partial \eta} \right)$ and T is the general tensor. T is a function of both geometry and Cartesian permeability tensors and its components are given by [16],

$$\begin{aligned} T_{11} &= \left(k_{11} y_\eta^2 + k_{22} x_\eta^2 - (k_{12} + k_{21}) x_\eta y_\eta \right) / |J| \\ T_{12} &= \left(k_{12} x_\xi y_\eta + k_{21} x_\eta y_\xi - (k_{11} y_\xi y_\eta + k_{22} x_\xi x_\eta) \right) / |J| \\ T_{21} &= \left(k_{12} y_\xi x_\eta + k_{21} y_\eta x_\xi - (k_{11} y_\xi y_\eta + k_{22} x_\xi x_\eta) \right) / |J| \\ T_{22} &= \left(k_{11} y_\xi^2 + k_{22} x_\xi^2 - (k_{12} + k_{21}) x_\xi y_\xi \right) / |J| \end{aligned} \tag{3.4}$$

where, $|J| = x_\xi y_\eta - y_\xi x_\eta$ is the determinant of the Jacobian. Resolving the x, y components of velocity along the unit normals to the curvilinear coordinates (ξ, η) gives rise to the general tensor flux components

$$\begin{aligned} F &= -\int \left(T_{11} \phi_\xi + T_{12} \phi_\eta \right) d\eta, \\ G &= -\int \left(T_{21} \phi_\xi + T_{22} \phi_\eta \right) d\xi, \end{aligned} \tag{3.5}$$

Thus the equation (3.3) can be written as

$$\int_{\Omega} \left(\partial_\xi \tilde{F} + \partial_\eta \tilde{G} \right) d\xi d\eta = M \tag{3.6}$$

where, $\tilde{F} = T_{11}\phi_\xi + T_{12}\phi_\eta, \tilde{G} = T_{21}\phi_\xi + T_{22}\phi_\eta$.

When $T_{12} = T_{21}$, it is seen that $T_{12}^2 \le T_{11}T_{22}$ ensuring that (3.2) is elliptic in transform space.

The pressure ϕ in equation (3.2) is solved subject to certain boundary conditions. On domain boundary $\partial\Omega$ and solid walls, Neumann boundary conditions specifies zero flux condition with $(K\nabla\phi)\cdot\hat{n} = 0$, where $\hat{n}$ is the unit outward normal vector. In at least one location within the domain, Dirichlet boundary condition specifies pressure, $\phi = \phi_c$ (sub-script *c*-denotes constant value).

3.3 Fundamentals of Grid Definition

As mentioned earlier it is important in reservoir simulation to choose the correct discretization method to ensure accuracy and credibility of the numerical results. In finite volume, the properties of the rock can be defined either at the cell centres or the cell vertices, which gives rise to the two types of grids, namely, cell-centered scheme (Block Centered Geometry) and cell vertex scheme (Corner Point Geometry) respectively.

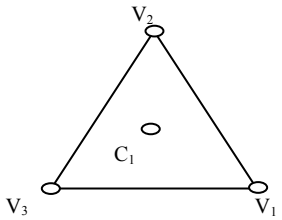


Figure 3.1 A Sample Triangular Element for FVM description.

Consider Figure (3.1). In cell-centered scheme the properties of rock will be defined at the cell centroid (or the circumcenter) C_1 of the triangular element i.e., in the cell centred approach the element mesh is used as a control volume and the centre of control volumes are considered as computational nodes. On the other hand, in cell vertex scheme, the rock properties will be defined about the 3 vertices of the triangular element, namely, V_1, V_2 and V3. i.e., the control volume is formed around each of the vertices or nodes by connecting the midpoints of the element faces and the centres of the elements and Shape functions are used to describe the variation of a variable within an element. Cell vertex scheme has certain obvious advantages over the cell centered scheme such as lesser number of unknowns in comparison. Cell-centered scheme is convenient compared to vertex-centered methods, when

considering discontinuous media properties combined with a quadrilateral or triangular primary mesh. In this case it is easy to align the grid edges, and hence, the control-volume boundaries with media discontinuities [29].

A number of schemes have been developed on the cell vertex based finite volume formulation, for e.g. [66], [67], [17]. Since, in this project, only cell centered scheme is used, it will be described in detail further using a sample 3 element mesh (Figure 3.2).

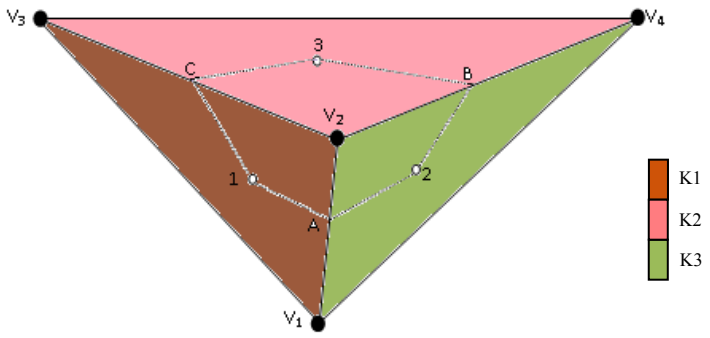


Figure 3.2 Triangular 3 element sample mesh

Consider the figure 3.2. This sample mesh can be simply described as follows:

Cluster Vertex: V_2, Cluster Elements: 1, 2 & 3,

Element 1: V_1-V_2-V_3, Element 2: V_4-V_2-V_1, Element 3: V_3-V_2-V_4

Sub-cell 1: 1-A-V_2-C, Sub-cell 2: 2-B-V_2-A, Sub-cell 3: 3-C-V_2-B.

The different shades of colour represent the control volumes over which the rock properties are constant. The grid points are considered to be the cell centres and the values for pressures are to be found at these points. Either the centroid or the circumcenter of the triangle can be considered as the cell centres. When the cell centres of each element in the cluster are connected to the mid-points of the cell edges containing the cluster vertex, we obtain a polygon encompassing the cluster vertex which is the *dual cell* (for e.g. from figure 3.2, dual cell is the domain enclosed by 1-A-2-B-3-C-1). The cell edges between neighbouring elements in a cluster are called *interfaces* (for e.g. V_1-V_2, V_3-V_2 and V_4-V_2 from figure 3.2). These interfaces act as boundaries between varying properties assigned to each element. The edge points (for e.g. A, B and C in figure 3.2) divide the interfaces into two segments called *sub-interfaces*. Each sub-cell is separated from other sub-cells by these sub-interfaces (e.g. A-V_2, B-V_2, and C-V_2.). The sub-cells when being defined can have the points lying on the cell edges at either midpoints or at any other point between the cluster vertex and the cell edge

midpoint. This option of having varying regions of sub-cell based on the point where the interface pressures or flux continuity are assumed to be leads to a family of schemes with Quadrature point q varying from 0 to 1. q = 0 being the cluster vertex and q = 1 being the cell edge midpoint.

When a cluster is considered, local flux continuity and pressure continuity needs to be defined. In case of TPS, pressure continuity is defined by assuming pressures at the right and left edge continuity points in the sub-cell. This is further explained in Chapter 4. In case of the recently developed Full Pressure Support scheme, pressure continuity is defined using pressures at the edge midpoints and an additional interface pressure is assigned to the cluster vertex [66].

In short it can be summarised that, in this research work, only unstructured triangular grids are used. The finite volume formulation adopted is the cell-centered formulation. Fluid in the reservoir is assumed to be single-phase and homogenous and obeying Darcy's Law.

Chapter 4 Triangular Pressure Support Scheme

4.1 Introduction

In the previous chapter, the basic problem to be solved in reservoir simulation has been defined. It is our aim to evaluate numerically the pressure in the reservoir. We have assumed the fluid to be in single-phase within the reservoir. In order to evaluate the pressure we thus have to solve the pressure equation that results from the Darcy's law. In this chapter, a cell-centered full tensor finite volume method of solving for the pressure is explained in detail and a simple outline of the MATLAB code algorithm created is presented here along with the computational examples and results.

4.2 Triangular Pressure Support (TPS) Formulation

In this scheme, for every cluster, the pressure within each triangular element in the cluster is considered to vary linearly over a triangular region within each sub-cell. It is necessary that for every sub-interface in the cluster, pressure continuity and flux continuity normal to the sub-interface be present. In order to ensure pressure continuity, interface pressures are introduced on each sub-interface (Figure 4.1).

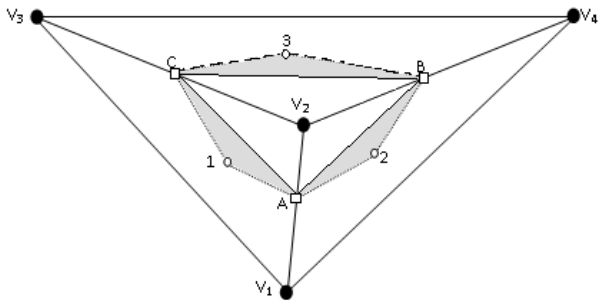


Figure 4.1 Interface pressures are indicated by squares and can be at any point between the cluster vertex and the cell edge midpoint. In this case, interface pressures are shown to be at points A, B and C.

The interface pressures are denoted as ϕ_A, ϕ_B, ϕ_C in the figure 4.1. It is important to note that the interface pressures may be discontinuous when passing along a triangle edge from one cluster to neighbouring cluster, but they are always continuous in the normal direction by construction [19].

The shaded regions in Figure 4.1 depict the regions in each sub-cell where the pressure assumes a local linear variation. These triangles, formed within each sub-cell, over which

pressure is linearly varying is called the pressure sub-triangles (e.g. 1-A-C, 2-B-A, 3-C-B). Pressure sub-triangles are essentially defined in each sub-cell with the cell grid point joined to the left and right triangle edge continuity points.

Local flux continuity is imposed across each sub-interface at each continuity point (e.g. A, B, C from figure 4.1). For a cluster *i* containing N_i elements, the number of interface pressures will be N_i (if cluster vertex does not lie on the boundary, else if the cluster vertex lies on the boundary an additional interface pressure will exist.) and the number of cell centre pressures will be N_i (irrespective of the position of the cluster vertex in the mesh). These N_i interface pressures within the dual cell can be expressed in terms of the N_i cell centre pressures using the corresponding N_i flux continuity conditions imposed on each sub-interface locally.

Once the interface pressures are assumed, at each sub-interface, flux continuity equations are written in terms of the interface pressures and the cell-centre pressures. Taking all the flux continuity equations in the cluster and solving for interface pressures, the interface pressures can be written in terms of the cell centre pressures in the cluster. The discrete flux F across each cell sub-interface maybe then written as a linear combination of the cell centre pressure values

$$F = -\sum_{j \in N_i} t_j \Phi_j \tag{4.1}$$

The coefficients t_j are called *transmissibilities* associated with the flux interface. Since the flux must be zero when Φ_j is constant for all $j \in N_i$, all consistent discretization must satisfy the condition $\sum_{j \in N_i} t_i = 0$.

In the general curvilinear coordinate system, equation (3.3), after application of Gauss divergence theorem becomes,

$$-\oint_{\delta\Omega} (T\tilde{\nabla}\phi) \cdot \vec{n}_t \, d\Gamma = M \tag{4.2}$$

where, $\vec{n}_t$ is the transform space normal vector and $\delta\Omega$ is the outer boundary of the cell in transform space. Thus, the discrete sub-interface flux can be defined as

$$F_i = -\int \sum_{j=1}^{2} T_{i,j} \phi_{\xi_j} \, d\Gamma_i \tag{4.3}$$

where, ϕ is the local piecewise linear approximation over each sub-triangle. $\phi_{\xi_1} = \phi_\xi, \phi_{\xi_2} = \phi_\eta$ are the local potential derivatives which are approximated by local potential differences between the interface pressures and cell-centered pressures.

Considering a sample sub-triangle (1, A, C) from figure 4.1,

$$\begin{pmatrix} \phi_\xi \\ \phi_\eta \end{pmatrix} = \begin{pmatrix} \phi_A - \phi_1 \\ \phi_C - \phi_1 \end{pmatrix} \tag{4.4}$$

$$\begin{pmatrix} x_\xi \\ x_\eta \end{pmatrix} = \begin{pmatrix} x_A - x_1 \\ x_C - x_1 \end{pmatrix}, \begin{pmatrix} y_\xi \\ y_\eta \end{pmatrix} = \begin{pmatrix} y_A - y_1 \\ y_C - y_1 \end{pmatrix} \tag{4.5}$$

In Equation (4.4) and (4.5) we see that coordinates of the continuity points A and C are necessary to be found. The position of the continuity points depends on the Quadrature q briefly mentioned earlier. For TPS, q can vary between 1 and 0, with q =1 being the position at the edge midpoint and q = 0 being the cluster vertex. In practice, q is never equal to 0, therefore, 0<q≤1. Thus we can find the coordinates of the continuity points which are dependent on the parameter q as,

$$\begin{aligned} x_A &= (1-q)x_{v_2} + (q)\frac{(x_{v_1} + x_{v_2})}{2} \\ y_A &= (1-q)y_{v_2} + (q)\frac{(y_{v_1} + y_{v_2})}{2} \end{aligned} \tag{4.6}$$

Discrete Darcy velocity is given by

$$v_h = -K\nabla\phi_h \tag{4.7}$$

Using equations (4.4) and (4.5), equation (4.7) becomes,

$$v_h = -\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix} \frac{1}{J} \begin{pmatrix} \phi_A - \phi_1 \\ \phi_C - \phi_1 \end{pmatrix}$$

$$v_h = \frac{1}{J} \begin{bmatrix} \phi_\xi(-k_{11}y_\eta + k_{12}x_\eta) + \phi_\eta(k_{11}y_\xi - k_{12}x_\xi) \\ \phi_\xi(-k_{21}y_\eta + k_{22}x_\eta) + \phi_\eta(k_{21}y_\xi - k_{22}x_\xi) \end{bmatrix} \tag{4.8}$$

The normal flux at the left hand side of sub-cell edge A-V_2 is given by,

$$F_A^{\ 1} = v_h.dL_A = \frac{1}{J}\begin{bmatrix} \phi_\xi(-k_{11}y_\eta + k_{12}x_\eta) + \phi_\eta(k_{11}y_\xi - k_{12}x_\xi) \\ \phi_\xi(-k_{21}y_\eta + k_{22}x_\eta) + \phi_\eta(k_{21}y_\xi - k_{22}x_\xi) \end{bmatrix} \frac{1}{2}\begin{bmatrix} y_{v2} - y_{v1} \\ -(x_{v2} - x_{v1}) \end{bmatrix} \tag{4.9}$$

Equation (4.9) can be expanded to obtain the $F_A^{\ 1}$ in terms of ϕ_ξ, ϕ_η and their coefficients as,

$$\begin{aligned} F_A^{\ 1} = \frac{1}{2J}\{&\phi_\xi(-k_{11}y_\eta(y_{v2} - y_{v1}) + k_{12}x_\eta(y_{v2} - y_{v1}) + k_{21}y_\eta(x_{v2} - x_{v1}) - (x_{v2} - x_{v1})k_{22}x_\eta) \\ &+\phi_\eta(k_{11}y_\xi(y_{v2} - y_{v1}) - k_{12}x_\xi(y_{v2} - y_{v1}) - k_{21}y_\xi(x_{v2} - x_{v1}) + k_{22}x_\xi(x_{v2} - x_{v1}))\}. \end{aligned} \tag{4.10}$$

In terms of general tensor T, the same flux is expressed as,

$$F_A^1 = -(T_{11}^1\phi_\xi + T_{12}^1\phi_\eta)\Big|_A^1 \tag{4.11}$$

Equating equations (4.10) and (4.11) we obtain values for $T_{11}^1 \& T_{12}^1$. Similarly evaluating the normal flux on sub-cell edge V_2-C, we can obtain the values of $T_{21}^1 \& T_{22}^2$. Therefore, for TPS scheme, the physical space approximation of elements of the general tensor T is given by,

$$\begin{aligned} T_{11}^1 &= \frac{1}{2J}(k_{11}y_\eta\Delta y_{v2v1} + k_{22}x_\eta\Delta x_{v2v1} - k_{12}x_\eta\Delta y_{v2v1} - k_{21}y_\eta\Delta x_{v2v1}) \\ T_{12}^1 &= \frac{1}{2J}(-k_{11}y_\xi\Delta y_{v2v1} - k_{22}x_\xi\Delta x_{v2v1} + k_{12}x_\xi\Delta y_{v2v1} + k_{21}y_\xi\Delta x_{v2v1}) \\ T_{21}^1 &= -\frac{1}{2J}(k_{11}y_\eta\Delta y_{v2v3} + k_{22}x_\eta\Delta x_{v2v3} - k_{12}x_\eta\Delta y_{v2v3} - k_{21}y_\eta\Delta x_{v2v3}) \\ T_{22}^1 &= -\frac{1}{2J}(-k_{11}y_\xi\Delta y_{v2v3} - k_{22}x_\xi\Delta x_{v2v3} + k_{12}x_\xi\Delta y_{v2v3} + k_{21}y_\xi\Delta x_{v2v3}) \end{aligned} \tag{4.12}$$

Using these, the flux continuity equations for the sample 3 cell cluster can be written as;

$$\begin{aligned} F_A &= -(T_{11}^1\Phi_{A1} + T_{12}^1\Phi_{C1}) = T_{12}^2\Phi_{B2} + T_{22}^2\Phi_{A2} \\ F_B &= -(T_{11}^2\Phi_{B2} + T_{12}^2\Phi_{A2}) = T_{12}^3\Phi_{C3} + T_{22}^3\Phi_{B3} \\ F_C &= -(T_{11}^3\Phi_{C3} + T_{12}^3\Phi_{B3}) = T_{12}^1\Phi_{A1} + T_{22}^1\Phi_{C1} \end{aligned} \tag{4.13}$$

where, $\Phi_{A1} = \phi_A - \phi_1$, and so on. F_A gives the flux leaving sub-cell 1 at continuity point A as well as the flux entering the sub-cell 2 at continuity point A.

Equation (4.13) contains 3 interface pressures $[\phi_A, \phi_B, \phi_C]$ which can be written in terms of the cell centre pressures $[\phi_1, \phi_2, \phi_3]$. Equation (4.13) basically can be written as,

$$
\begin{aligned}
&-(T_{11}^{1}\Phi_{A1}+T_{12}^{1}\Phi_{C1})-(T_{12}^{2}\Phi_{B2}+T_{22}^{2}\Phi_{A2})=0\\
&-(T_{11}^{2}\Phi_{B2}+T_{12}^{2}\Phi_{A2})-(T_{12}^{3}\Phi_{C3}+T_{22}^{3}\Phi_{B3})=0\\
&-(T_{11}^{3}\Phi_{C3}+T_{12}^{3}\Phi_{B3})-(T_{12}^{1}\Phi_{A1}+T_{22}^{1}\Phi_{C1})=0
\end{aligned}
\tag{4.14}
$$

Equations (4.14) contain 3 interface pressures which can now be considered as the unknowns and solved using the 3 equations to obtain the interface pressures in terms of cell centre pressures. This will give the discrete flux at each continuity point in terms of the cell-centre pressure values in the cluster. Each of these local fluxes is then globally assembled to form the global discrete pressure equation.

$$A\Phi = L \tag{4.15}$$

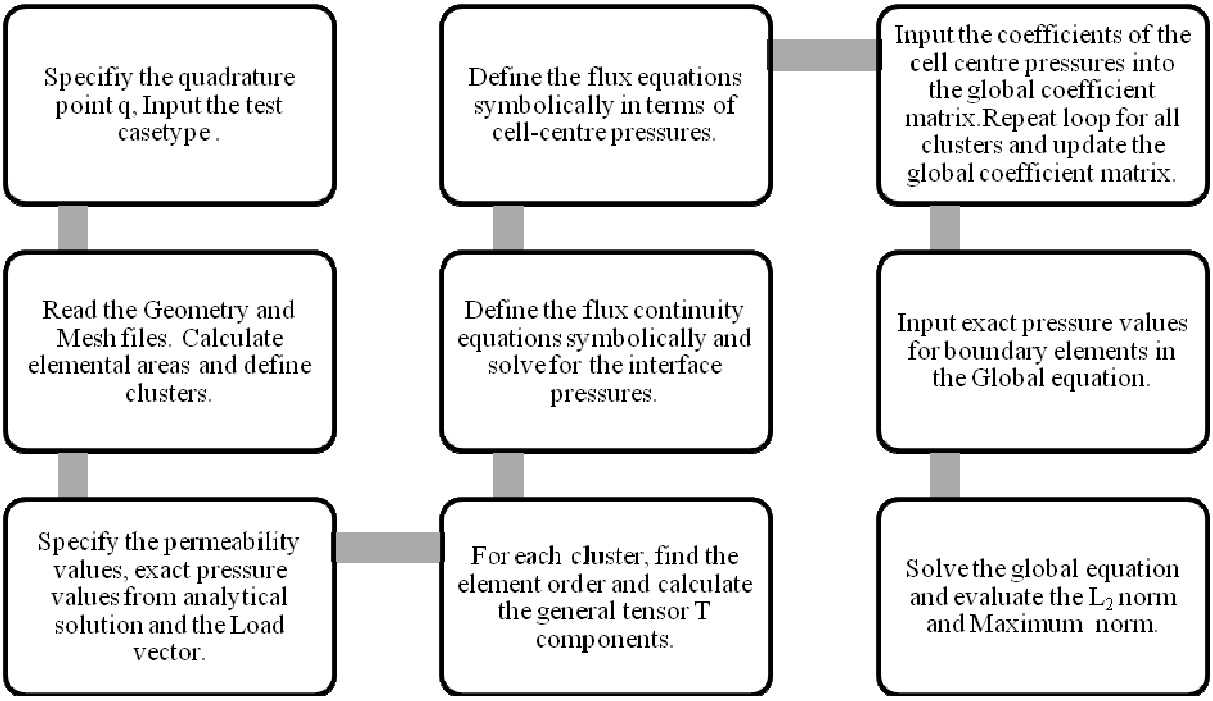
where $\Phi=[\phi_1,\phi_2,.....\phi_{nelem}]$, *nelem* is the total number of elements in the mesh. A is the coefficient matrix assembled from flux equations from each cluster. L is the load vector which is obtained by multiplying the elemental areas with the flow rates specified at the grid points.

It is to be noted that when the value of parameter q is taken as $\frac{2}{3}$, i.e. when continuity point is taken as $\frac{2}{3}$ of the half edge measured from cluster vertex, in the general tensor T (Equation (4.12), we find that $T_{12}^{1}=T_{21}^{1}$. When q=$\frac{2}{3}$, the scheme is said to be the symmetric physical space scheme. Other values of q, with 0<q≤1, leads to non symmetric general tensor T within each sub-cell. This further result in a non-symmetric global system matrix. However, the resulting discrete matrix is still conditionally positive definite and the positive definite condition for discrete ellipticity of the physical space scheme is now defined by [19],

$$((T_{12}^{1}+T_{21}^{1})/2)^2 \le T_{11}^{1}T_{22}^{1} \tag{4.16}$$

4.3 Code Algorithm Outline

The TPS scheme described so far has been coded in MATLAB. For each and every step care has been taken to keep the code simple by utilising many of MATLAB's inbuilt functions and defined variables. A simple step by step outline of the coded scheme is shown in the following process chart;



The code developed is attached in the Appendix and contains extensive commentary for each step and is self explanatory. Since MATLAB does not require variable declaration like that in FORTRAN the code is comparatively simpler and shorter. Functions are used for certain repetitive steps to shorten the main application console code. A number of different examples were solved and results and convergence graphs plotted.

The Mesh files are generated in MATLAB using a Delaunay Triangulation Algorithm. For a set P of points in a plane, triangulation DT(P) such that no point in P is inside the circum-circle of any triangle in DT(P) is called Delaunay Triangulation. Detailed explanation of the concept can be found in [36]. The mesh and boundary files are then written as .txt files and read into the main code.

4.4 Computational Examples

The following examples are solved using the physical space formulation described above and for each case, convergence rates are graphed using the log to the base of 2 of L_2 norms vs. square root of the total number of elements in the mesh. L_2 norm is found as;

$$L_2(\phi) = \left(\frac{\sum_i (A_i(\phi_{h,i} - \phi_i)^2)}{\sum_i A_i} \right)^{1/2}$$ where $A_i, \phi_i, \phi_{h,i}$ are the area, analytical pressure solution, and numerical pressure respectively for element *i*.

Case 1: Constant Pressure Case

Consider a square 2×2 domain. Let the permeability K and exact pressure ϕ be as defined below;

$$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \phi = 2.5\,.$$

For constant pressure value evaluation it is found that the numerical result is equivalent to the exact solution for both sub-cell space transform scheme and physical space scheme. The L_2 norm for pressure is found to be around $1.4e^{-016}$. Figure 4.2 depicts the numerical solution plot with points 'o' and exact solution plot with points '×'. It is seen coincident at all points in the grid. This is just a test case done mainly for debugging purposes.

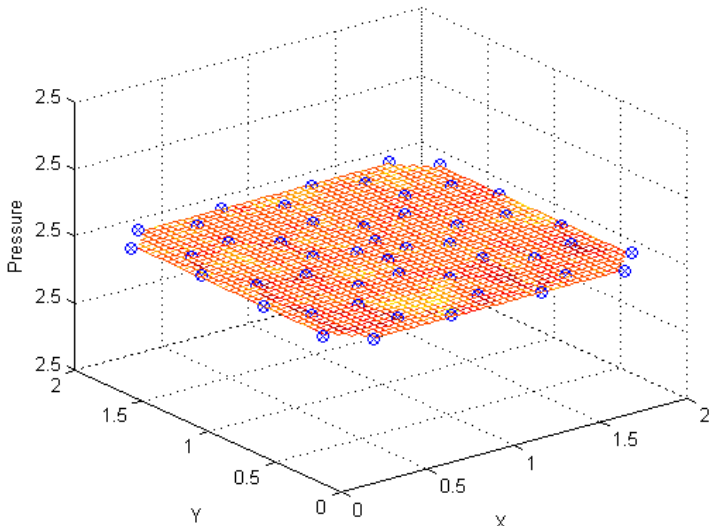


Figure 4.2 Case 1 Solution Plot

Case 2: Linear Pressure Case

The permeability K and exact pressure ϕ is taken as;

$$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \phi = x + y + 1$$

For linear pressure values, the numerical result obtained is equal to the exact solution with L_2 norms of around $1.2e^{-16}$. This is found for all cases of the sub-cell transform and physical space scheme. This is as expected, since, in the TPS numerical formulation, pressure is defined linearly in the pressure sub-triangle.

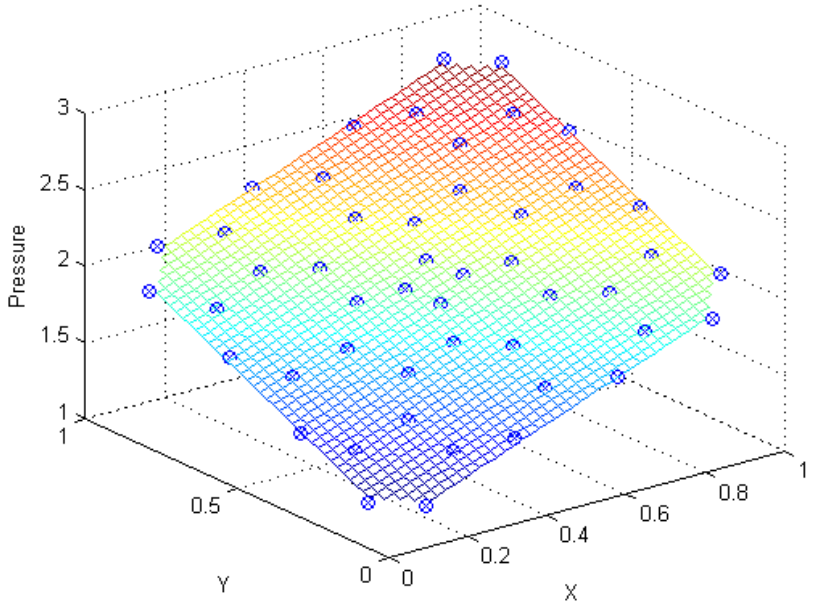


Figure 4.3 Case 2 Solution Plot

Figure 4.3 shows the numerical ('o') and exact ('×') solution to be coinciding at all points in a simple 48 element gird over a square [0, 1] domain.

Case 3: Discontinuous Bilinear Case

In this case, instead of having uniform permeability tensor throughout the unit square domain, we take different permeability tensors for $x \leq 0.5$ and $x > 0.5$ on the Cartesian grid. This case allows us to observe the effect of internal boundary aligned cell centered triangular grid on the numerical solution. The permeabilities and the exact pressure values are taken as;

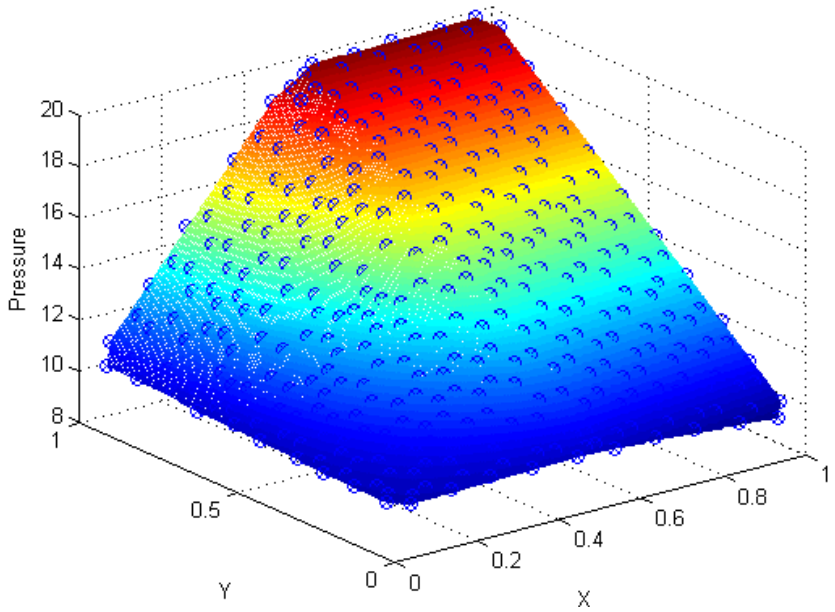


Figure 4.4 Case 3 Solution Plot

$$K = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}, \phi = 10 + 20xy, \text{ for } x \le 0.5,$$

$$K = \begin{bmatrix} 10 & 2 \\ 2 & 100 \end{bmatrix}, \phi = 10.75 - 1.5x + 9y + 2xy, \text{ for } x > 0.5.$$

Specified flow rate or forcing term Q can be found using $Q = -\nabla\cdot(K\nabla\phi)$. In this case, we obtain Q = -20 for x ≤ 0.5 and Q = -8 for x > 0.5. The maximum norm and the L_2 norms of the pressure are found. Figure 4.4 shows the numerical ('o') and exact ('×') solution of pressure as it varies over the domain for a 346 element mesh with Quadrature point at 2/3. The sudden change in pressure at x=0.5 is observable. The Figure 4.5 gives the convergence for different Quadrature points. It is clear from the obtained convergence graphs that the TPS scheme on an unstructured boundary aligned triangular grid converges at a higher rate for finer meshes. The table 4.1 below summarises the convergence rates for different Quadrature values. This example is taken from [67] where the convergence rate for cell vertex finite volume formulation of TPS scheme has been found on a uniform quadrilateral mesh, the values of which are also included in the table to compare the two schemes. The comparison shows us the advantage of using the cell centered scheme on an unstructured mesh for internal boundary (fractures etc) cases.

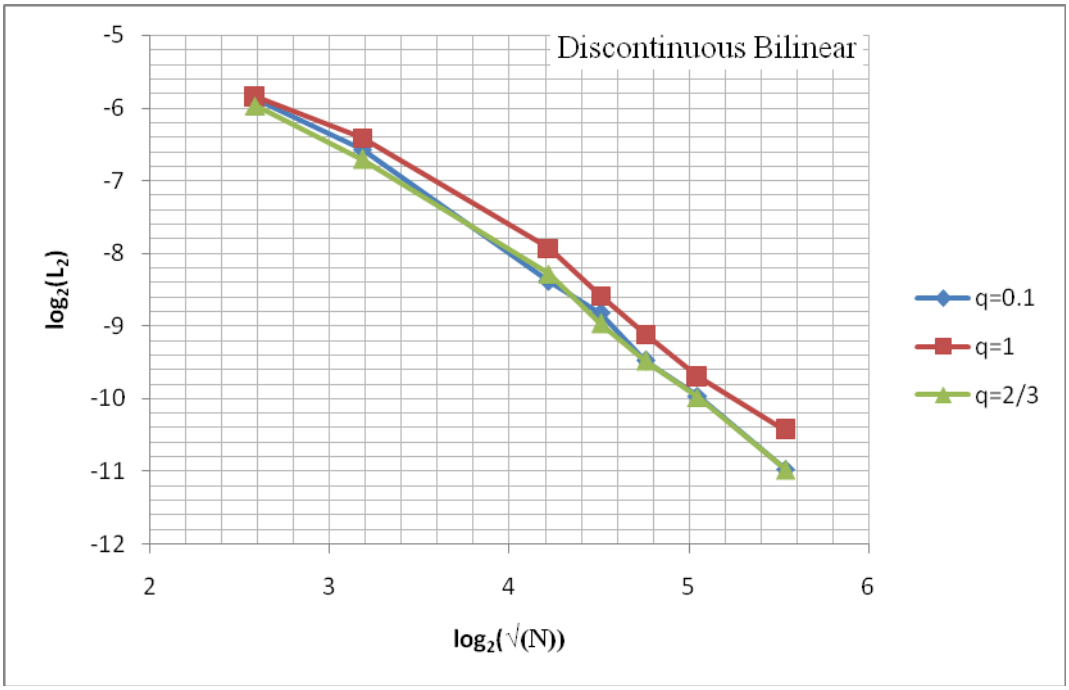


Figure 4.5 Convergence graph for Case 3

Table 4.1 Convergence rates for Case 3

Flux Continuity Point q	Coarse Unstructured mesh (<500 elements) Cell-Centered	Fine Unstructured mesh (> 500 elements) Cell-Centered	Fine Structured Mesh (>500 elements) Cell-Vertex [67]
0.1	1.8	1.94	1.010
2/3	1.5	2.16	0.999
1	1.4	1.74	0.984

It is clear that for cases with internal discontinuity, cell-centered finite volume formulation of TPS scheme on an unstructured mesh gives a better convergence rate, the highest being that of q=2/3 with 2.16 for fine meshes.

Case 4: Quadratic Pressure Case

The permeability *K* and exact pressure ϕ over the domain is taken as;

$$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \phi = x^2 + y^2$$

This is a simple quadratic pressure case taken over a square domain [0, 1]. The forcing vector is found to be -4. Figure 4.6 shows the numerical ('o') and exact ('×') solution plot on a 48 element mesh with quadrature point at 2/3. Table 4.2 gives the L_2 norm and maximum norm values for pressure for the different Quadrature point cases and figure 4.7 shows the

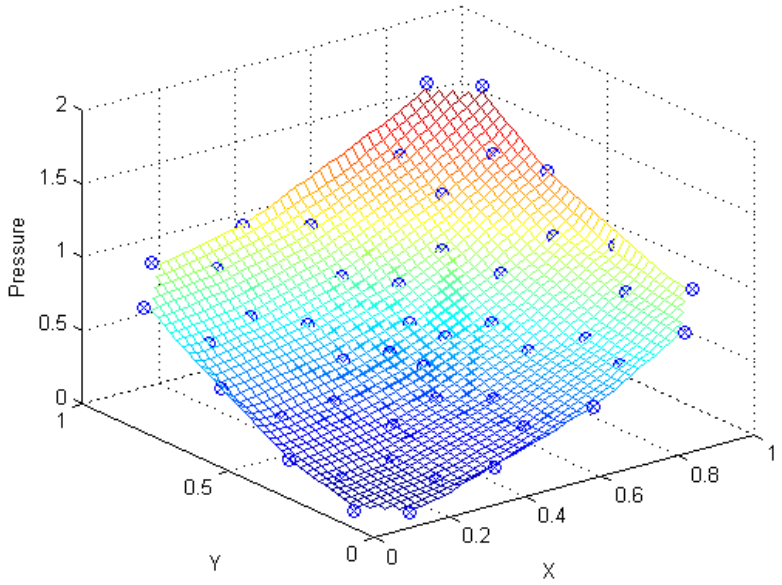


Figure 4.6 Case 4 Solution Plot

convergence of the scheme for this quadratic case for different quadrature points plotted as $\log_2$ of L_2 norm vs. $\log_2$ of square root of the total number of elements in the mesh.

The convergence rate for all the 3 cases of quadrature point plotted for is similar and for all three cases, convergence rate is seen to tend to 2 with increase in number of elements.

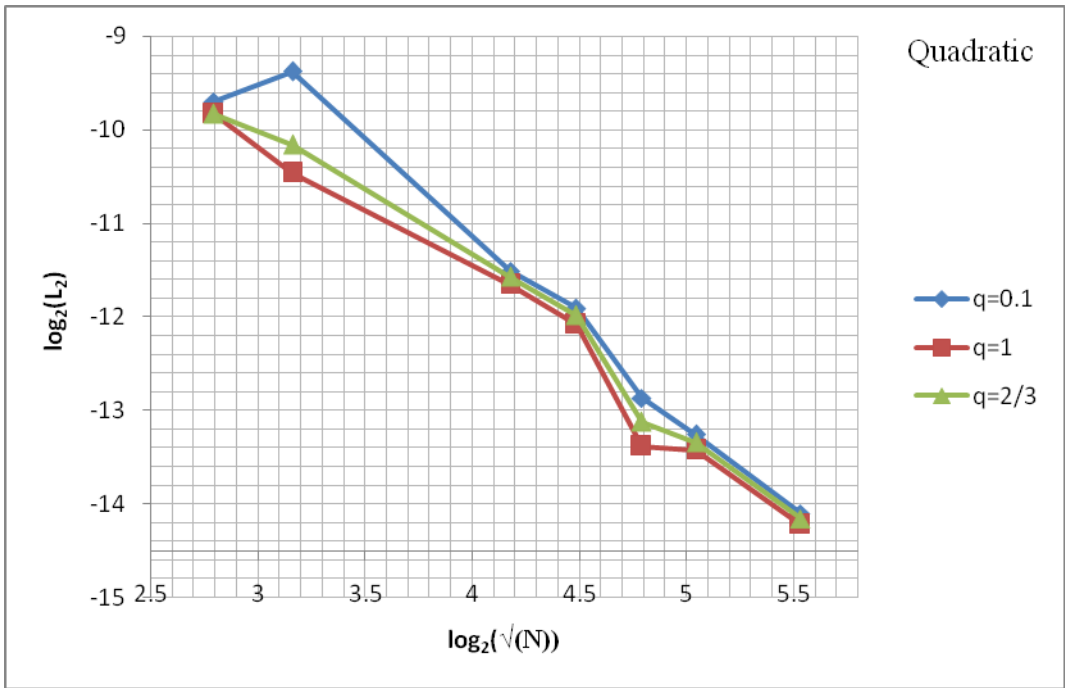


Figure 4.7 Convergence plot for Case 4

Table 4.2 L_2 Norm and Maximum Norm for Pressure for Case 4

Number of Elements in Mesh	L_2 Norm			Maximum Norm		
	q=0.1	q=2/3	q=1	q=0.1	q=2/3	q=1
768	1.34E-04	1.13E-04	9.36E-05	5.20E-04	4.68E-04	4.51E-04
1092	1.02E-04	9.65E-05	9.11E-05	3.53E-04	3.34E-04	3.05E-04
2148	5.69E-05	5.48E-05	5.26E-05	1.87E-04	1.62E-04	1.55E-04

From the table 4.2 it can be concluded that for particular case of quadratic pressure function, flux continuity point at q=1 fares better than that at q= 2/3. As the number of elements increases (>1000), the convergence rates for q = 0.1, 2/3 and 1 is found to be 1.73, 1.67, 1.62 respectively from the convergence graph plotted (Figure 4.7).

Case 5: Discontinuous Quadratic Pressure Case

This case is taken from [16] where it has been solved using cell-centered quadratic mesh. This has a piece-wise quadratically varying pressure field with a discontinuity aligned at x = 0.5. The domain considered is unit square domain [0, 1]. The permeability and exact pressure values are taken as;

$K = \begin{bmatrix} 50 & 0 \\ 0 & 1 \end{bmatrix}$, $\phi = c_l x^2 + d_l y^2$ for $x < 0.5$,

$K = \begin{bmatrix} 1 & 0 \\ 0 & 10 \end{bmatrix}$, $\phi = a_r + b_r x + c_r x^2 + d_r y^2$ for $x \geq 0.5$, where,

$\alpha = K_{\alpha\alpha}|_r / K_{\alpha\alpha}|_l$, $\beta = K_{\beta\beta}|_l / K_{\beta\beta}|_r$,

$a_r = 1$,

$f = 4a_r / ((\alpha - 2)\beta + 1)$,

$b_r = (\beta - 1) f$,

$c_r = f$,

$d_r = -c_r \, K_{\alpha\alpha}|_r / K_{\beta\beta}|_r$,

$c_l = \alpha\beta c_r$,

$d_l = d_r$.

Figure 4.8 shows the numerical ('o') and exact ('×') solution of pressure as it varies over the domain for a 346 element mesh with Quadrature point at 2/3 (Physical Space Scheme).

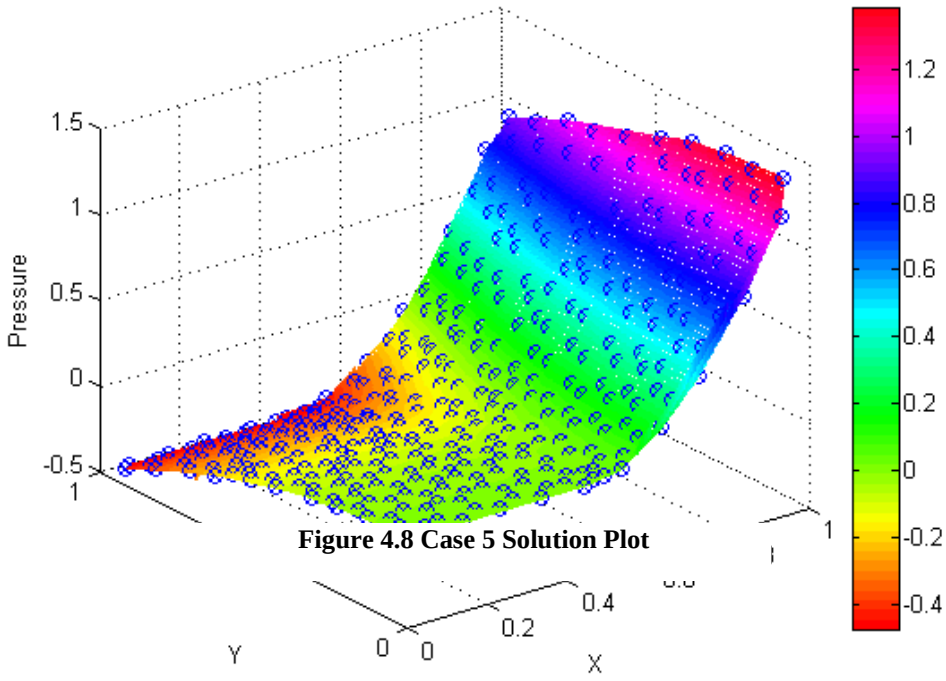


Figure 4.8 Case 5 Solution Plot

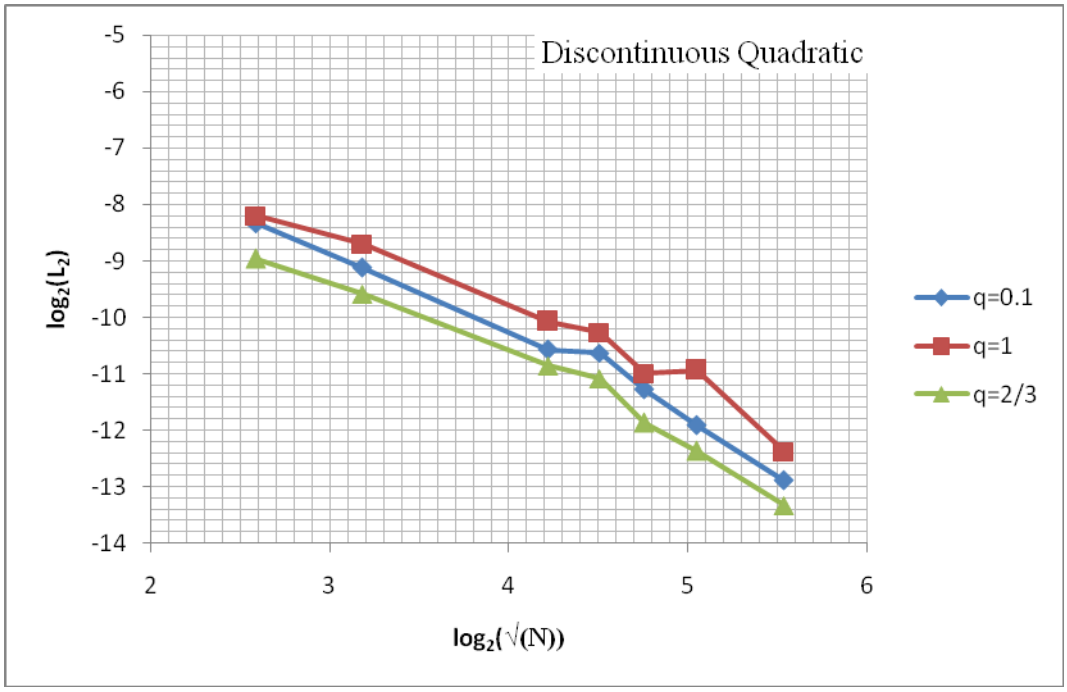


Figure 4.9 Convergence graph for Case 5

From plotting the convergence graph (Figure 4.9), it is found that the trend for convergence emerges clearly for finer mesh sizes. Convergence rates for q = 0.1, 2/3 and 1 are found to be 2.2, 1.5 and 3 respectively. Here we find that the q=1 and q=0.1 show better convergence compared to q=2/3.

Case 6: Large Anisotropy Case

This example is taken from [16]. Like in [16], domain chosen here is unit square [0, 1]. Here, TPS scheme is evaluated on cell centered unstructured triangular mesh for a large anisotropy ratio problem. The permeability is taken as;

$K = \begin{bmatrix} y^2 + \varepsilon x^2 & -(1-\varepsilon)xy \\ -(1-\varepsilon)xy & x^2 + \varepsilon y^2 \end{bmatrix}$ where, $0 < \varepsilon \le 1$. $\varepsilon = 1$ makes the problem isotropic. Degree of anisotropy of the problem increases with decreasing ε .

$\phi = \sin(\pi x)\sin(\pi y)$ is the analytical solution of pressure and the forcing vector Q is found to be,

$$Q = \{-2\varepsilon x\cos(\pi x)\pi\sin(\pi y) + (y^2 + \varepsilon x^2)\sin(\pi x)\pi^2\sin(\pi y)) - (-1+\varepsilon)y\sin(\pi x)\cos(\pi y)\pi$$
$$-2(-1+\varepsilon)xy\cos(\pi x)\pi^2\cos(\pi y) - (-1+\varepsilon)x\cos(\pi x)\pi\sin(\pi y) - 2\varepsilon y\sin(\pi x)\cos(\pi y)\pi$$
$$+(x^2 + \varepsilon y^2)\sin(\pi x)\sin(\pi y)\pi^2\}.$$

The computational solution has been found for ε=1, ε=0.1 and ε=0.01. For the physical space scheme, the numerical and exact solutions are shown in Figures 4.10, 4.11 and 4.12.

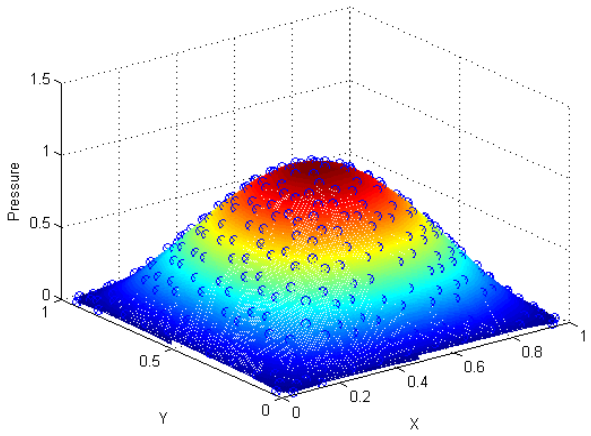


Figure 4.10 Solution plot for Case 6: =0.1, q=2/3

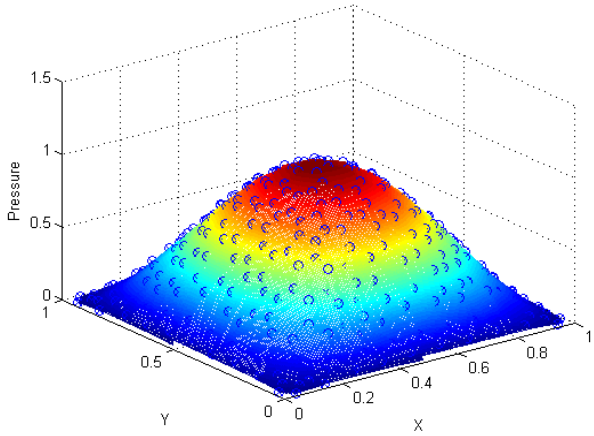


Figure 4.11 Solution plot for Case 6: =0.01, q=2/3

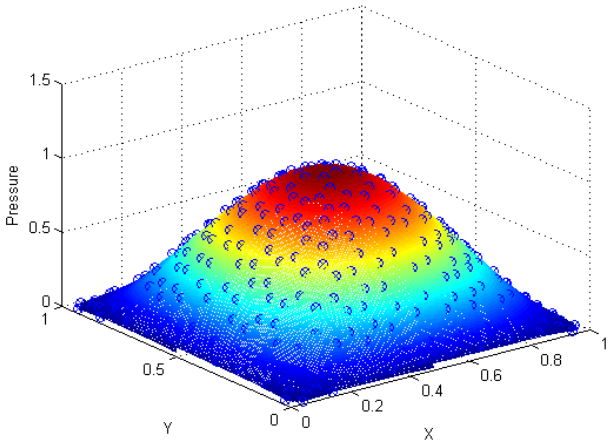


Figure 4.12 Solution plot for Case 6: =1, q=2/3

The solution plots contain both the numerical 'o' and exact 'x' solutions. Overlapping of these two solutions are seen in all the cases.

Figure 4.13 shows the convergence graph plotted for ε=0.1. In [16] for very fine mesh, convergence rate for this case has been found to be around 2.03. From the results obtained here, the convergence rates for q=0.1,0.5,2/3 and 0.95 are found to be 2.09, 2.18, 1.65 and 1.75 respectively.

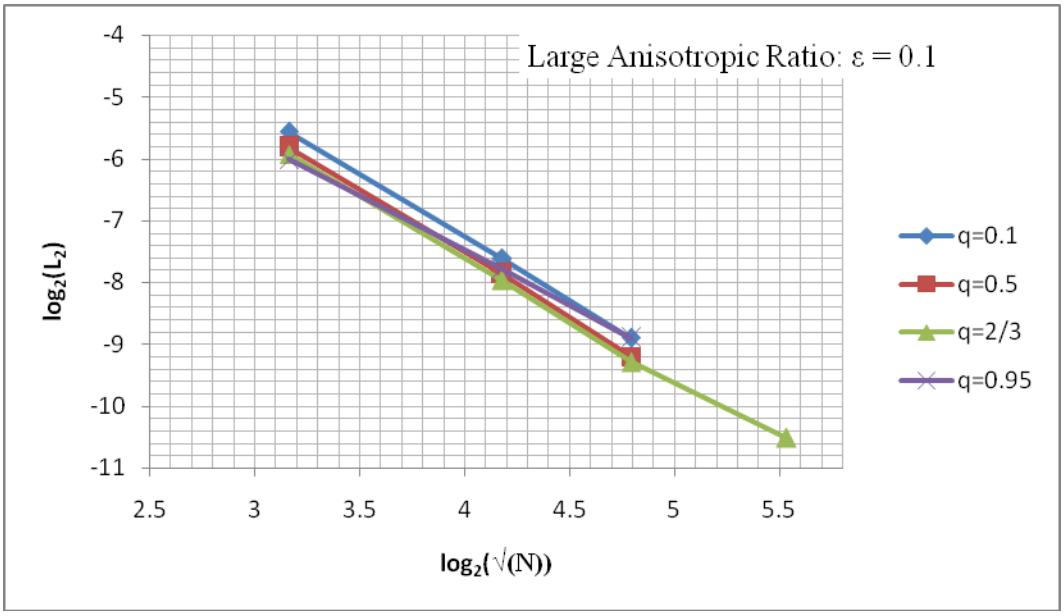


Figure 4.13 Convergence graph for Case 6: =0.1

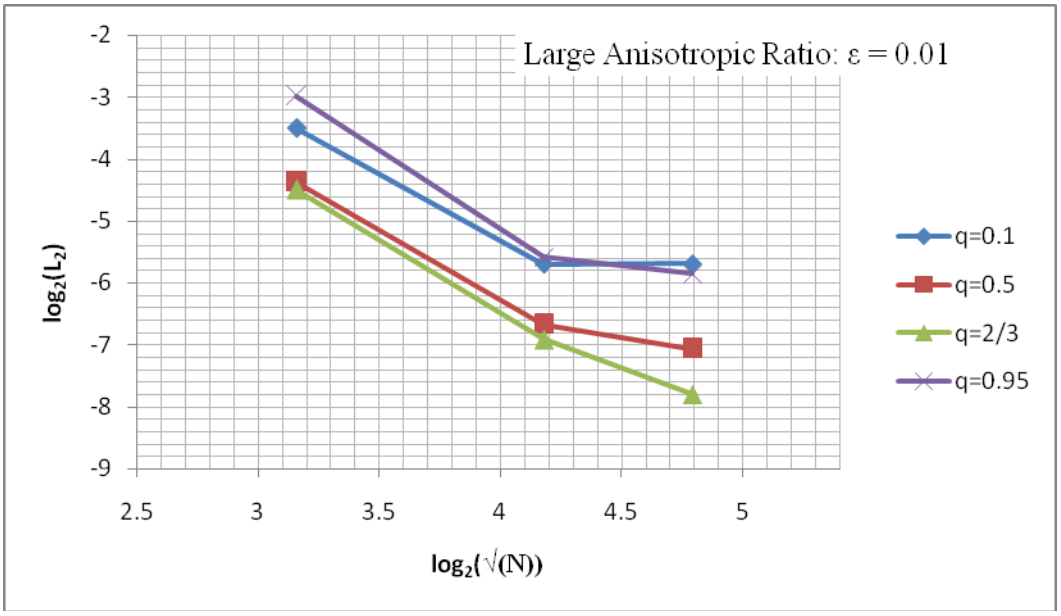


Figure 4.14 Convergence graph for Case 6: =0.01

Figure 4.14 shows the convergence graph plotted for ε=0.01, i.e. with a higher degree of anisotropy in the problem. Whereas from figure 4.13, it is evident that the numerical solution converges for lower degree of anisotropy, figure 4.14 shows that the numerical solution tends to be non-convergent for the physical space scheme. The computational solution for the same was also found to be non-convergent in case of physical space scheme in [16].

For the above examples, the domain geometries and meshes used are generated in MATLAB's interactive PDE TOOLBOX. Unit square domains were created and meshed using the inbuilt meshing function. The meshing is carried out using a Delaunay triangulation algorithm. The mesh size is determined from the shape of the geometry which in this case is a unit square for all the examples solved above.

Case 7: Circular Region with Well

This example is taken from [19]. In this case, we test a circular domain with a well included. The challenge in this case is the large difference in size of the domain and the size of the well which in reality maybe a difference of kilometres in case of reservoir domain to a 0.1 meter well.

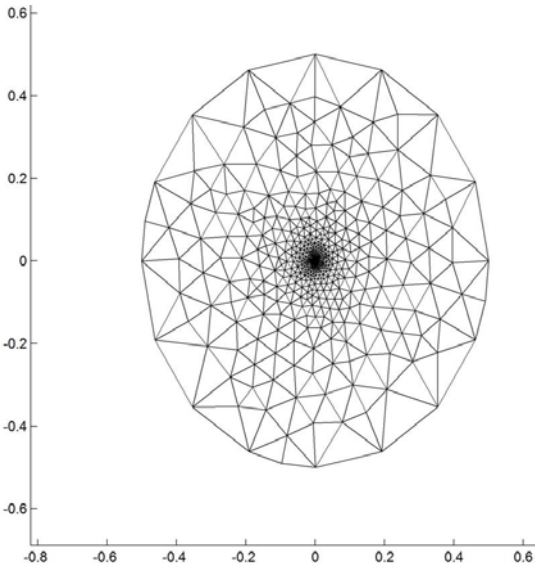


Figure 4.15 Unstructured Triangular Grid for Case 7

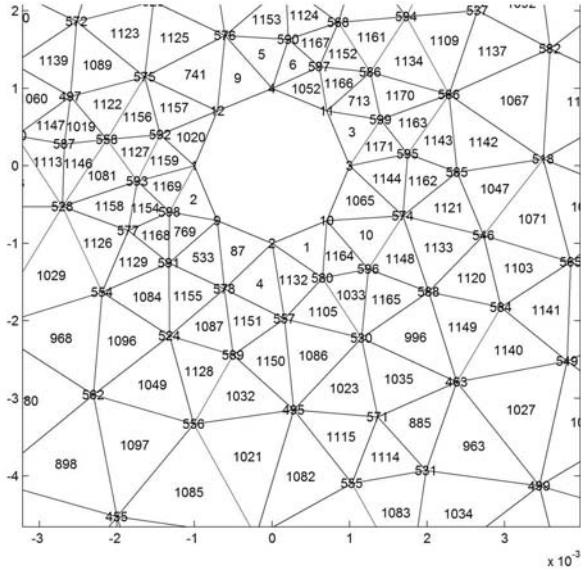


Figure 4.16 Well region in Case 7

In this case the well is considered as a geometrical object with Dirichlet Boundary conditions. A homogenous circular domain of radius R = 0.5 with a circular well with radius r = 0.001 placed at the centre of the domain is taken with permeability *K*;

$K = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and pressure equation has an analytical solution ;

$\phi(r) = \dfrac{\phi_w - \phi_B}{\log \dfrac{r_w}{R}} \log\left(\dfrac{r}{R}\right) + \phi_B$, where, $r = \sqrt{(x^2 + y^2)}, \phi_B = 1.0, \phi_w = 2.0$ and r_w is the well radius.

The Figure 4.15 shows the unstructured triangular grid used consisting of 1171 elements. The geometry and mesh both were generated using the PDE TOOLBOX in MATLAB. Figure 4.16 shows the zoomed view of the well region around which the meshing is denser in order to obtain better numerical results.

Figure 4.17 shows the exact solution plot for this case. It is evident that the pressure in the domain increases from 1 at the boundaries to 2 around the well geometry. The solution plots are for 1171 element grid shown in figure 4.15. Test runs were also done with lower number of element meshes which shows the numerical and exact solutions to be converging for finer meshes as expected.

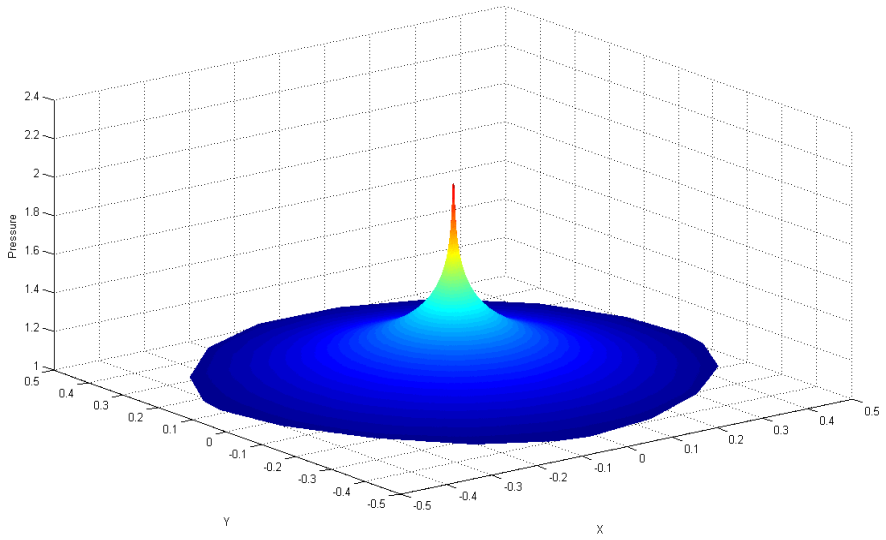


Figure 4.17 Exact Solution Plot for Case 7

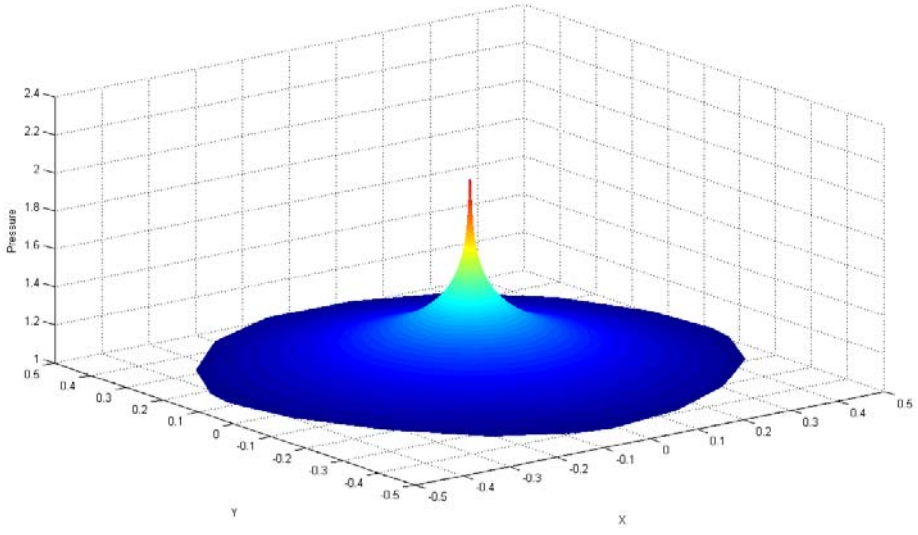


Figure 4.18 Numerical Solution Plot for Case 7: q=2/3

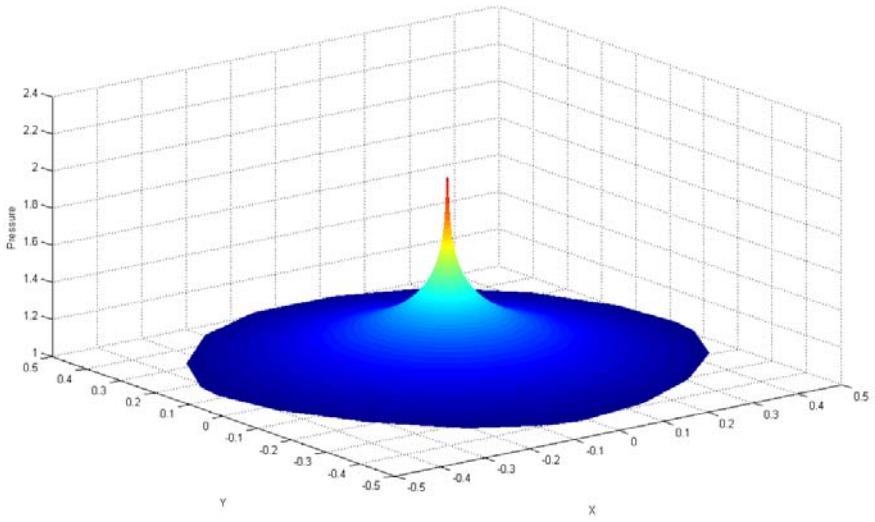


Figure 4.19 Numerical Solution Plot for Case 7: q=1

Figure 4.18 shows the numerical solution plot for q=2/3 for which the L_2 norm for pressure is found to be 1.42×10^{-4} and maximum norm of pressure 7.75×10^{-4}. Figure 4.19 shows the numerical solution plot for q=1 for which the L_2 norm for pressure is found to be 2.12×10^{-4} and maximum norm of pressure 1.4×10^{-3}. The pressure contour plot for 1171 elements and q=1 for sub-cell transform scheme is shown in figure 4.20.

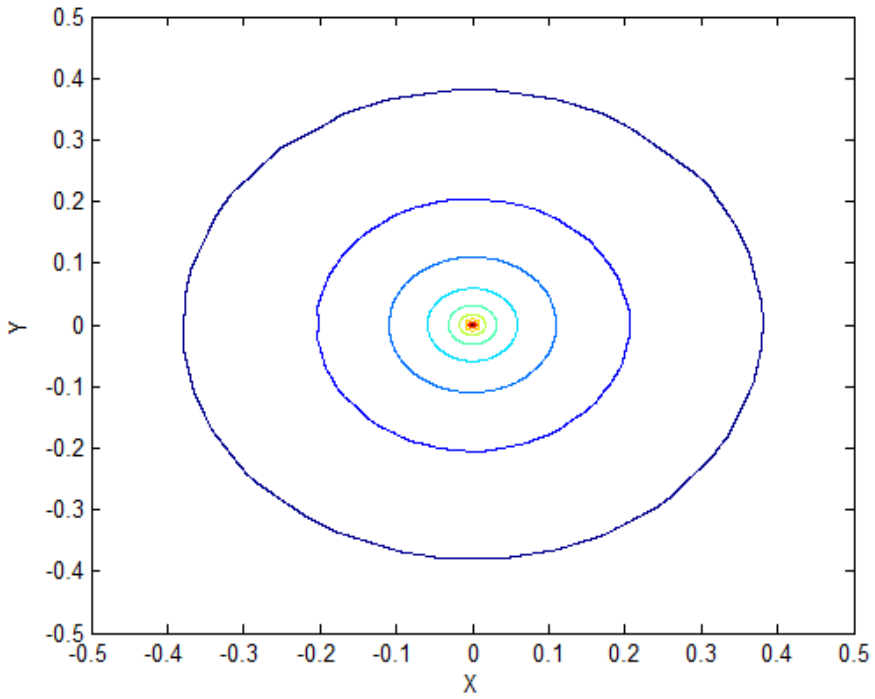


Figure 4.20 Contour Plot of Pressure for Case 7

Case 8: Polar Case with Cartesian Grid

This example is taken from [67], where it has been solved for FPS scheme on a uniform quadrilateral mesh. A unit square domain is considered with the exact solution taking the form of $\phi(r,\theta) = r^{\alpha}(a_i \sin(\alpha\theta) + b_i \cos(\alpha\theta))$. The domain is divided into 4 regions with varying permeability tensors (Figure 4.20). The values of the permeability tensors are taken as;

$$K_1 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, K_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K_3 = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}, K_4 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

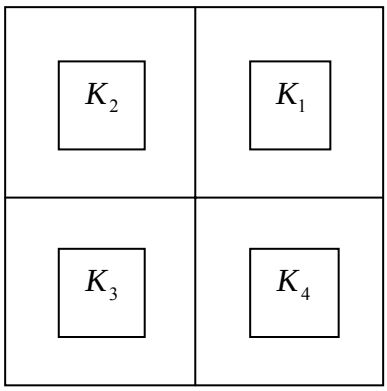


Figure 4.21 Permeability in the Domain for Case 8

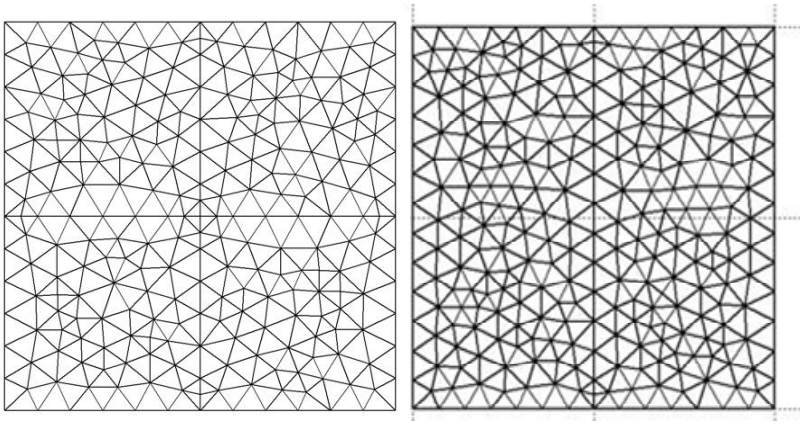


Figure 4.22 Left: Unstructured triangular mesh aligned perfectly with the permeability discontinuity in domain, Right: Mesh unaligned with the permeability discontinuities.

The domain discontinuity shown in figure 4.21 has an internal angle $\theta = \frac{\pi}{2}$. Figure 4.21 shows a 470 element mesh completely aligned with the permeability discontinuities in the domain. The constants in the analytical solution are taken as;

α=0.53544095

a_1= 0.44721360 & b_1= 1.0

a_2=-0.74535599 & b_2= 2.33333333

a_3= -0.94411759 & b_3= 0.5555556

a_4= -2.40170264 & b_4= -0.481481481

Figure 4.23 shows the exact solution plotted on a 470 element grid with unstructured triangular elements aligned with the permeability field. From the exact solution plot, the permeability discontinuities in the domain are apparent.

The computational solution of this problem is shown to produce expected convergence behaviour. In this case, an effort is made to underline the importance of the alignment of the elements with respect to the permeability discontinuities by comparing the results of L_2 norm and maximum norm for pressure obtained using an aligned unstructured grid and a non-aligned unstructured grid both using cell-centered TPS formulation. Table 4.3 summarises the comparison.

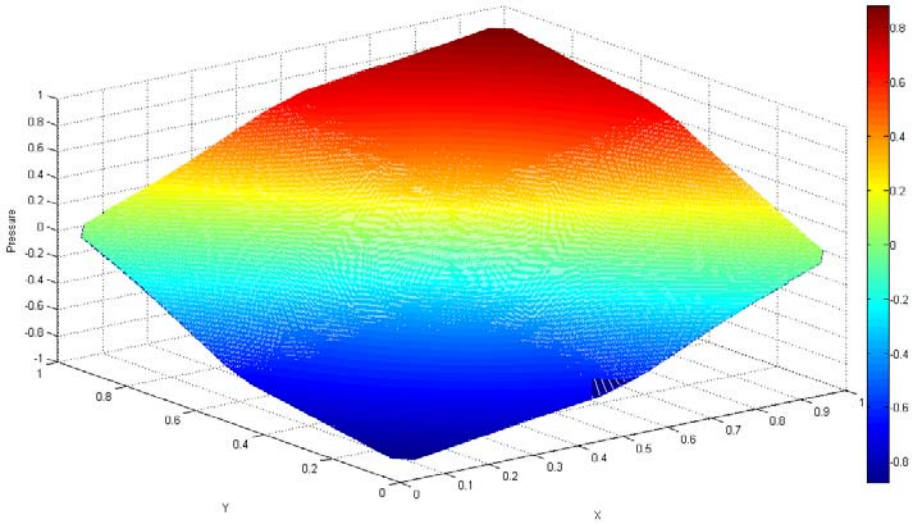


Figure 4.23 Exact Solution Plot for Case 8

Table 4.3 Comparison of L_2 norms for pressure for aligned and non-aligned unstructured grids

Elements	L_2 Norm					
	Completely Aligned Grid			Non-Aligned Grid		
	q=0.1	q=2/3	q=1	q=0.1	q=2/3	q=1
82	0.0125	0.0099	0.0129	0.0175	0.0117	0.0128
346	0.0019	0.0043	0.0059	0.0115	0.0071	0.0077
516	0.0015	0.0037	0.0051	0.0071	0.0056	0.0064

Table 4.4 Comparison of Maximum norms for pressure for aligned and non-aligned unstructured grids

Elements	Maximum Norm					
	Completely Aligned Grid			Non-Aligned Grid		
	q=0.1	q=2/3	q=1	q=0.1	q=2/3	q=1
82	0.0492	0.0391	0.0509	0.0726	0.0494	0.0572
346	0.0284	0.0346	0.0491	0.0648	0.0486	0.0474
516	0.0258	0.0317	0.0449	0.0470	0.0651	0.0692

From the results table it is evident that the L_2 norms are smaller for the grid aligned to the domain permeability discontinuities. Maximum norms are also smaller for the aligned grids in comparison. This shows the need to use boundary (be it external or internal) aligned grids while solving reservoir problems to ensure better numerical results.

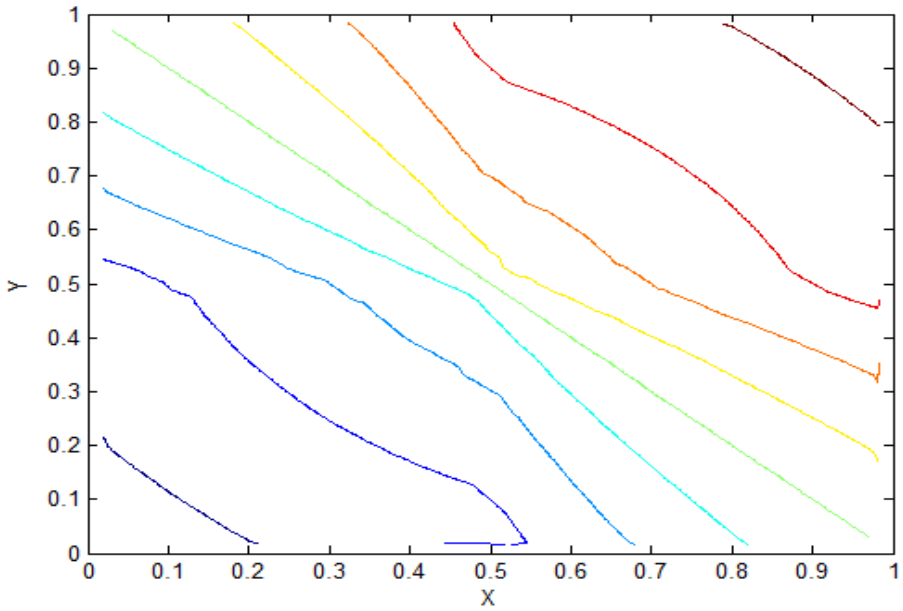


Figure 4.24 Contour Plot of Pressure for Case 8

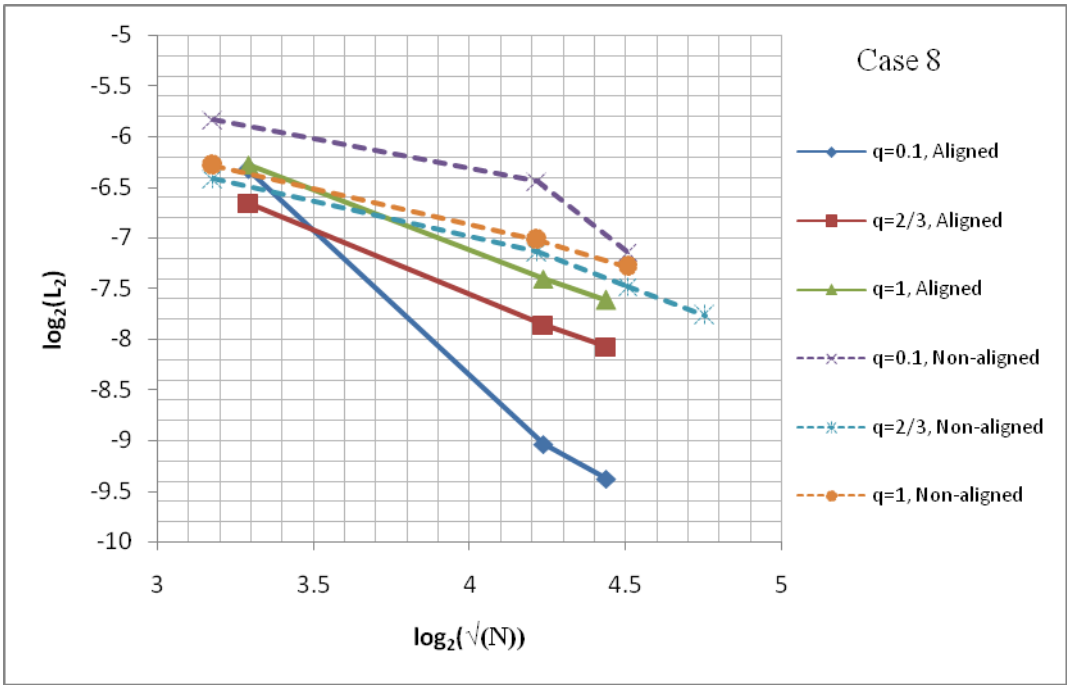


Figure 4.25 Convergence Graph for Case 8

Figure 4.24 gives the pressure contour plot for the base sub-cell space transform numerical solution on a 470 element perfectly aligned mesh. The effect of variation in pressure due to the presence of permeability discontinuities in the domain is noticeable even on this coarse

mesh. Figure 4.25 shows the convergence of aligned coarse mesh (< 1000 elements). Convergence rates for q=0.1, 2/3 and 1 are found to be 2.3, 1.2 and 1.1 respectively.

Case 9: Cartesian mesh to solve highly anisotropic $\phi = e^{(xy)}$

In this case, the permeability tensor is taken as $K = 0.001 * \begin{bmatrix} 750.25 & 432.58 \\ 432.58 & 250.75 \end{bmatrix}$ and the exact pressure in the unit square domain is given by $\phi = e^{xy}$. The forcing vector is found to be $Q=(3001/4*y^2*e^{(x*y)}+21629/25*e^{(x*y)}+21629/25*x*y*e^{(x*y)}+1003/4*x^2*e^{(x*y)})$.

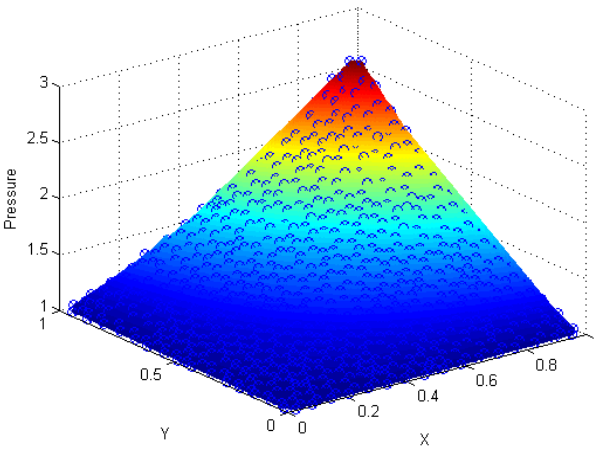


Figure 4.26 Solution Plot for Case 9: q=2/3

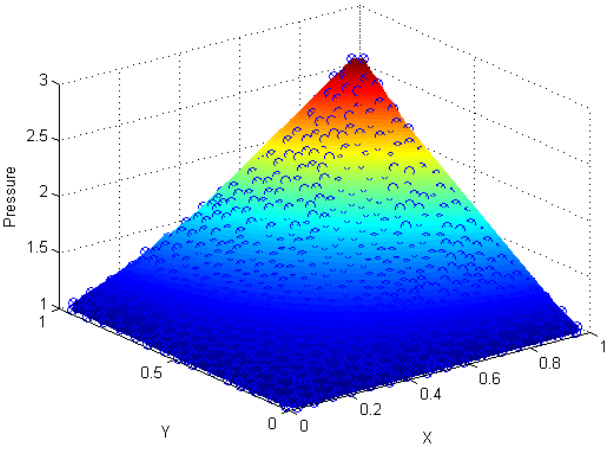


Figure 4.27 Solution Plot for Case 9: q=0.1

The exact solution, 'x', and the numerical solution, 'o', plots for the case is given in figures 4.26 and 4.27 for a 730 element mesh. For the flux continuity at q=2/3, the L_2 norm is found to be 2.8×10^{-3} and the maximum pressure norm 0.012. When the flux continuity parameter q is 0.1, L_2 norm is found to be 0.0112 and the maximum pressure norm 0.0475. Even for a coarse mesh such as this, a good conformance to the exact solution is obtained for such a highly anisotropic case.

Case 10: Strongly Discontinuous Full Tensor Field (Zigzag Field)

This computational example is sourced from [67] where it has been solved for structured quadrilateral fine meshes using FPS schemes as well for a particular case of q=1 for TPS scheme. Domain considered is a unit square with a source-sink pressure combination specified at the diagonally opposite corners of the domain. At the bottom left corner of the domain, pressure is taken at 0.0 whereas at the top right corner, pressure is taken at 200.0.

External boundary values of pressure are specified as 100.0 throughout. The permeability tensor is taken to be discontinuous in the domain (Figure 4.28).

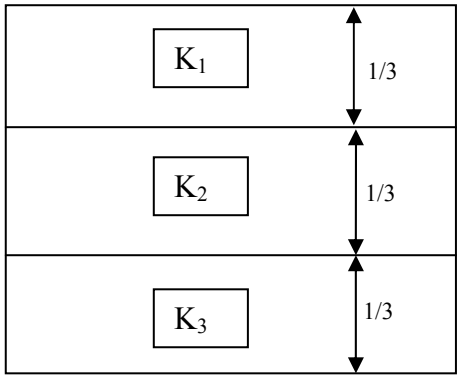


Figure 4.28 Permeability Discontinuities in the Domain

The permeabilities are taken as;

$$K_1 = \begin{bmatrix} 2464.360020 & +1148.683643 \\ +1148.683643 & 536.6399794 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} 2464.360020 & -1148.683643 \\ -1148.683643 & 536.6399794 \end{bmatrix}$$

$$K_3 = \begin{bmatrix} 2464.360020 & +1148.683643 \\ +1148.683643 & 536.6399794 \end{bmatrix}$$

The computation was carried out with 2846 elements with flux continuity point q=2/3 and q=1 for the physical space formulation of TPS scheme. Figure 4.29 shows the numerical pressure field in the domain and figure 4.30 shows the contour plot of the numerical pressure solution in the domain.

The numerical pressure solution plots clearly show the zigzag nature of the permeability field affecting the pressure in the domain. As expected from literature [67], for TPS scheme employed for this example, considerable oscillations are observed.

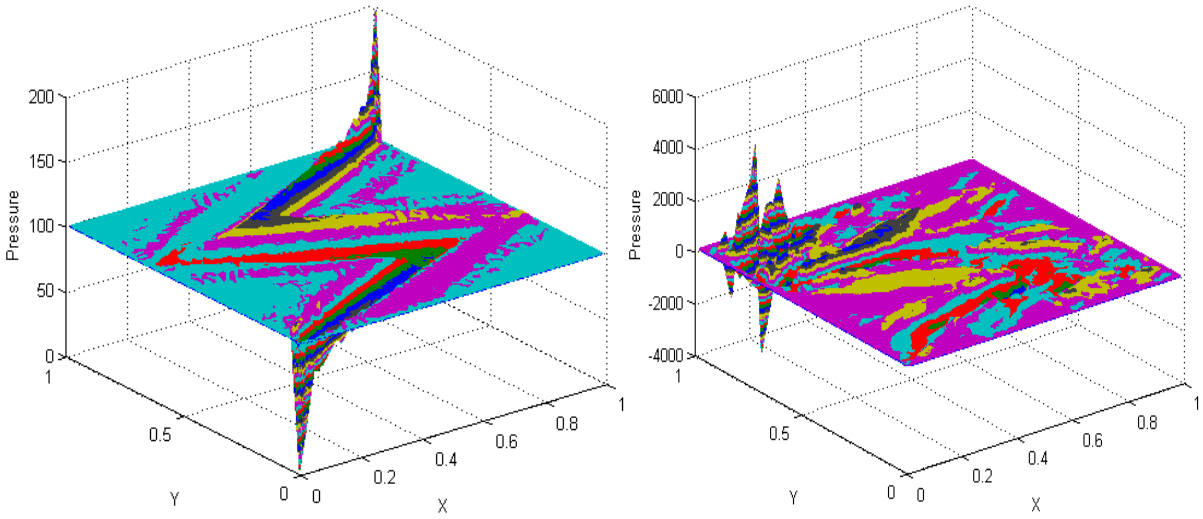


Figure 4.29 Numerical Solution Plot for Case 10: Left: q=2/3, Right: q=1

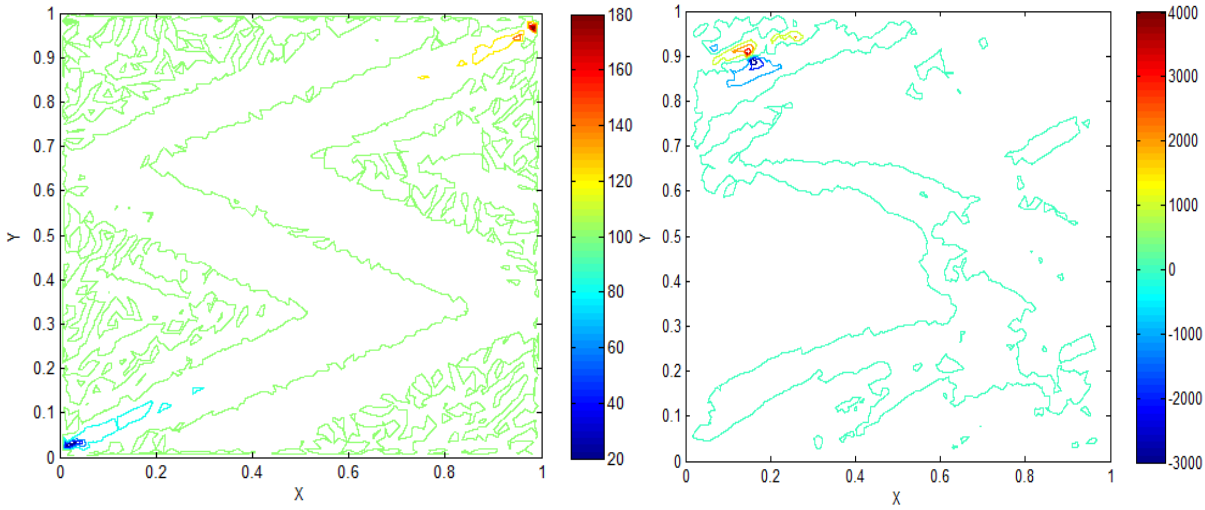


Figure 4.30 Pressure Contour Plot for Case 10: Left: q=2/3, Right: q=1

When the flux continuity point is specified by q=2/3, it is seen that the numerical solution obeys the maximum principle but the fails to satisfy the discrete maximum principle. When q=1, spurious oscillations increase considerably and even the maximum principle is shown not to be satisfied in this case.

Case 11: Square region with concentric permeability discontinuity and central source.

In this case, the domain is considered to have varying permeability as shown in figure 4.31. Domain considered is a unit square domain with centre at (0, 0). A concentric square of 0.5×0.5 is considered to be the region having permeability K_2.

Permeability K is taken in the different regions as;

$$K_1 = \begin{bmatrix} 2464.360020 & -1148.683643 \\ -1148.683643 & 536.6399794 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} 2464.360020 & 1148.683643 \\ 1148.683643 & 536.6399794 \end{bmatrix}$$

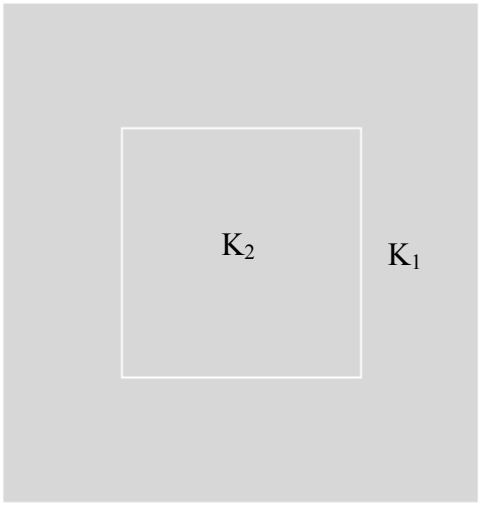


Figure 4.31 Discontinuous Permeability Field for Case 11

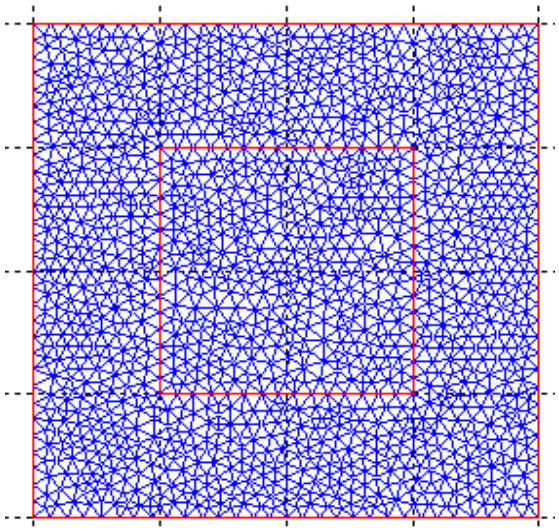


Figure 4.32 Domain Geometry and Aligned Mesh for Case 11

At centre (0, 0) of the domain pressure is specified as 1.0 and at the boundary walls pressure is specified as 0.0 uniformly. A 2900 element unstructured triangular mesh (Figure 4.32) is used to analyse the pressure equation in the domain. The case was run for physical space TPS scheme with q=2/3 and q=1. Figure 4.32 shows the numerical solution of pressure plotted in the domain. Figure 4.33 shows the numerical pressure contour in the domain. The variation in permeability leads to the zigzag nature of the pressure field in the domain.

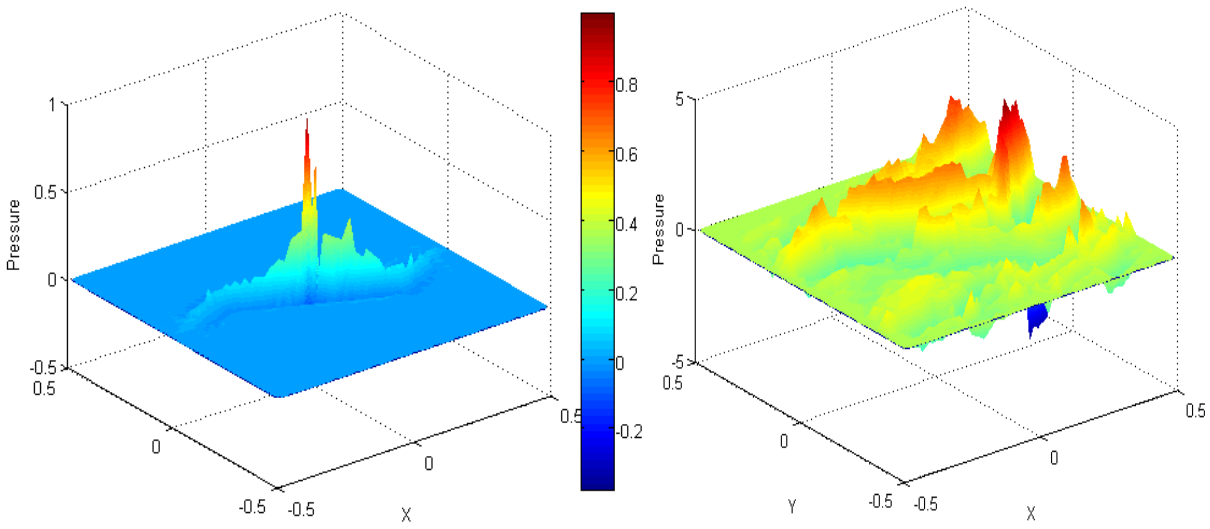


Figure 4.33 Numerical Solution Plot for Case 11: Left: q=2/3, Right: q=1

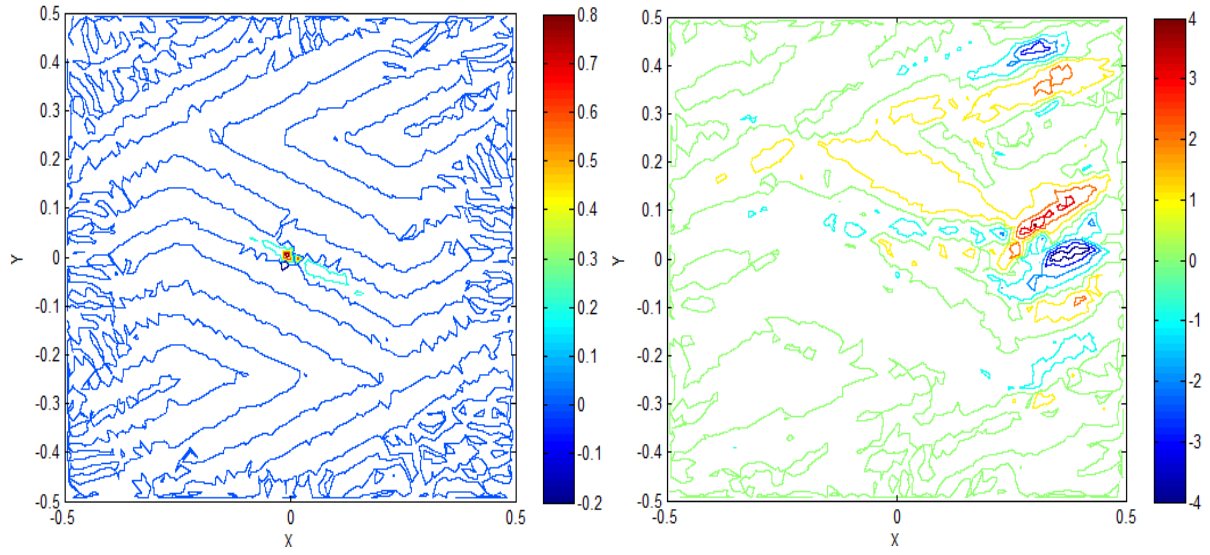


Figure 4.34 Contour Plot for Pressure in the domain for Case 11: Left: q=2/3, Right: q=1

From the contour plots it's evident that the unsymmetrical physical space scheme with q=1, produces a lot of spurious oscillation rendering the approximation unreliable in this case. It can be concluded that for cases such as this, which has the presence of the source as well as domain permeability discontinuity, the physical space scheme with symmetrical approximation provides the better results when compared to the unsymmetrical approximations for same mesh refinement.

Chapter 5 Conclusions and Future Scope

5.1 Research Conclusions

In the previous chapter, the elliptic (Darcy velocity) pressure equation for single phase fluid flow has been successfully demonstrated using cell-centered finite volume formulation with triangular pressure support. This work mainly concentrated on the physical space formulation. The code developed was used to obtain the convergence plots for test cases for both symmetric approximation with q=2/3, and a number of non-symmetric approximation (q≠2/3, e.g. q=1) thus providing a comparison of performance of the physical space scheme.

From the computational examples presented before, the following conclusions can be drawn;

1. TPS with physical space formulation gives exact results for linearly varying and constant pressure cases for all $0<q\leq1$, where q is the parameter fixing the flux continuity between the cluster vertex and cell edge mid-point.

2. For domains with discontinuous permeability, cell-centered TPS physical space formulation allows exact internal boundary alignment representation whereas cell-vertex TPS physical space formulation requires additional internal boundary alignment to ensure that the flow and rock properties within the control volume remain constant.

3. Quadratic pressure cases show promising convergence rates even for the coarse grids (< 2500 elements) used. Convergence rates are of order 1.7 for purely quadratic with unit permeability tensor whereas the convergence rates are of order 2 for discontinuous permeability case. From literature [19], the expected values of the convergence rate is of order 2 for fine meshes having more than 2500 elements in the domain. Hence, the code developed here is successfully conforming to the features of the TPS scheme.

4. For large anisotropic ratio cases, with increase in the degree of anisotropy, TPS physical space scheme has been previously found to be non-converging [19]. Figure 4.14 which shows the convergence plot for a high degree of anisotropy case indicates this trend. For physical space formulation with q=0.1, 0.5 and 0.95, no convergence has been obtained. But for q=2/3 which is the symmetric physical space scheme, which is in fact a particular member of the family of sub-cell transform space methods [19], the method is seen to be converging.

5. For challenging cases like case 7 which has the presence of a well region within the domain, the numerical solution obtained shows good conformance compared to the analytical solution in the region.

6. For cases having discontinuous permeabilities in the domain as well as the presence of source and sink, the pressure contours (Figures 4.30, 4.34) obtained, depicts the expected zigzag character of the pressure field. The pressure values in the domain lie between the maximum and minimum specified pressures in the domain when symmetric approximation is used (i.e. q=2/3) and hence can be said to be conforming to the maximum principle. However, the discrete maximum principle is violated as is indicated by the presence of spurious oscillations (seen in contours and iso-surfaces). For the non-symmetric approximation with q=1, the maximum principle is not being obeyed.

The code developed thus can be said to be working as expected from the principles of the scheme employed.

5.2 Future Scope of Work

Initial coding has already been carried out towards the end of this research work, using a similar algorithm in MATLAB, for the recently developed Full Pressure Support scheme [27], [66], [67], and [17], which are less likely to give spurious oscillations. The code requires further debugging and once done so, can be used to demonstrate the properties of this scheme for the unstructured cell-centered triangular finite volume formulation and can be compared with the current work.

Moreover, the existing cell vertex based TPS code in FORTRAN [67] can be combined with the currently developed cell centered TPS code in MATLAB to create a new hybrid numerical simulator which will draw upon the benefits of both the schemes. This will help develop a scheme which can be predicted to increase the efficiency in solving for the flow in porous media for a region containing discontinuities.

References

[1] Aavatsmark, I., Barkve, T., Boe, O., & Mannseth, T., *Discretization on Unstructured Grids for Inhomogeneous, Anisotropic Media. Part I: Derivation of the Methods*, Siam J. Sci. Comput. 19, 1998

[2] Aavatsmark, I., Barkve, T., Boe, O., & Mannseth, T., *Discretization on Unstructured Grids for Inhomogeneous, Anisotropic Media. Part II: Discussion and Numerical Results*, Siam J. Sci. Comput. 19, 1998

[3] Bear, J., *Dynamics of Fluids in Porous Media,* American Elsevier, New York, 1972.

[4] Boffi, D., and Gastaldi, L., *Mixed finite elements, compatibility conditions, and applications: lectures given at the C.I.M.E. Summer School held in Cetraro, Italy, June 26 - July 1, 2006,* Springer, 2008.

[5] Branets, L.V., Ghai, S.S., Lyons, S.L, & Wu, X., Exxon Mobil Upstream Research Company , *Efficient and accurate reservoir modelling using adaptive gridding with global scale up*, SPE, 2009.

[6] Breil, J., and Maire, P., *A cell-centered diffusion scheme on two-dimensional unstructured meshes*, Journal of Computational Physics 224, 2007.

[7] Carlson, M.R., *Practical reservoir simulation: using, assessing, and developing results*, PennWell Books, 2003.

[8] Chavent, G., Jaffre, J., and Roberts J.E., *Generalized cell-centered finite volume methods: application to two-phase flow in porous media*, 1997.

[9] Chavent, G., Jaffré, J., *Mathematical models and finite elements for reservoir simulation: single phase, multiphase, and multi-component flows through porous media*, Elsevier, 1986.

[10] Chen, Z., Huan, G., Ma, Y., *Computational methods for multiphase flows in porous media*, SIAM, 2006.

[11] Chen, Z., *Reservoir simulation: mathematical techniques in oil recovery*, Cambridge University Press, 2007.

[12] Correa, M.R., and Loula, A.F.D., *Unconditionally stable mixed finite element methods for Darcy flow*, Comput. Methods Appl. Mech. Engrg. 197, 2008.

[13] Crichlow, H.B., *Modern Reservoir Engineering – A simulation Approach*, Prentice-Hall, Inc. 1977.

[14] Durlofsky, L.J and Aziz, K., *Advanced Techniques for Reservoir Simulation and Modeling of Non-conventional Wells*, Stanford University, 2004.

[15] Edwards, M.G., and Pal, M., *Positive definite q-families of continuous sub-cell Darcy flux CVD(MPFA) Finite volume schemes and the mixed Finite element method*, International Journal for Numerical Methods in Fluids, Volume 57; Number 4, 2008. pg 355-387

[16] Edwards, M.G., and Rogers, C.F., *Finite volume discretization with imposed flux continuity for the general tensor pressure equation*, Computational Geosciences, Volume 2, Number 4, 1998. pg 259-290

[17] Edwards, M.G., and Zheng, H., *A quasi positive family of continuous Darcy-flux finite-volume schemes with full pressure support*, J. Comput. Phys 227, 2008, pg 9333-9364.

[18] Edwards, M.G., *Elimination of adaptive grid interface errors in the discrete cell centered pressure equation*, J. Comput. Phys. 126, 1996.

[19] Edwards, M.G., Friis, H.A., and Mykkeltveit, J., *Symmetric Positive Definite Flux-Continuous Full-Tensor Finite-Volume Schemes on Unstructured Cell-Centered Triangular Grids*, SIAM Journal on Scientific Computing, Volume 31, Issue 2, 2008. pg 1192-1220.

[20] Edwards, M.G., *M-matrix flux splitting for general full tensor discretization operators on structured and unstructured grids*, J. Comput. Phys. 160, 2000.

[21] Edwards M.G., and Zheng H., *Double-families of Quasi-Positive Darcy-Flux Approximations with Highly Anisotropic Tensors on Structured and Unstructured Grids"* J. Comput. Phys 229, 2010, 594-625 doi:10.1016/j.jcp.2009.09.037.

[22] Edwards, M.G., *Unstructured, control-volume distributed, full-tensor finite-volume schemes with flow based grids*, Computational Geosciences 6, 2002.

[23] Ertekin, T., Abou-Kassem, J.H., & King, G.R., *Basic Applied Reservoir Simulation*, SPE Textbook Volume 10, 2001.

[24] Fanchi, J.R., *Principles of Applied Reservoir Simulation*, Houston, Tex, Gulf Pub, 1997.

[25] Finite Difference Methods, 31-05-2010, <http://ltl.iams.sinica.edu.tw/document/training_lectures/2006/SH_Chen/Finite_Difference_Methods.pdf>

[26] Forsyth, P., *A control-volume finite element method for local mesh refinement in thermal reservoir simulation*, SPERE, 1990.

[27] Friis, H.A., and Edwards, M.G., *A family of MPFA Finite-volume schemes with full pressure support for the general tensor pressure equation on cell-centered triangular grids*, 2010.

[28] Gunasekera, D., Cox, J., and Lindsey, P., *The Generation and Application of K-Orthogonal Grid Systems*, SPE 37998, Texas, 1997.

[29] Hægland, H., Assteerawatt, A., Dahle, H.K., Eigestad, G.T., & Helmig, R., *Comparison of cell- and vertex-centered discretization methods for flow in a two-dimensional discrete-fracture–matrix system*, Advances in Water Resources Volume 32, Issue 12, December 2009.

[30] Heinemann, Z. E., & Leoben, M.U., *Interactive Generation of Irregular Simulation Grids and its Practical Applications*, SPE 27998, University of Tulsa Centennial Petroleum Engineering Symposium, 1994.

[31] Heinemann, Z.E., Brand, C.W., Munka, Margit, and Chen, Y.M., *Modelling Reservoir Geometry with Irregular Grids*, SPE Reservoir Engineering, Volume 6, Number 2, 1991. pg 225-232

[32] Hunt B., Lipsman, R., Rosenberg, J., Coombes, K., Osborn, J., Stuck, G., *A Guide to MATLAB for Beginners and Experienced Users*, Cambridge University Press, 2001.

[33] Ian D. Chivers and Jane Sleightholme, *Introduction to Programming with FORTRAN*, Springer-Verlag London Limited, 2006.

[34] Iske, A., & Randen, T., *Mathematical methods and modelling in hydrocarbon exploration and production*, Springer, 2005.

[35] Jamtveit, B., Yardley, B.W.D., *Fluid flow and transport in rocks: mechanisms and effects*, Springer, 1997.

[36] Katzmayr, M., & Ganzer, L., *An Iterative Algorithm for generating constrained Voronoi grids*, SPE, 2009.

[37] Kiusalaas, J., *Numerical Methods in Engineering with MATLAB*, Cambridge University Press, 2005.

[38] Kuzmin, D, Introduction to CFD, 31-05-2010, http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/cfd.html

[39] LeVeque, R.J., *Finite volume methods for hyperbolic problems*, Cambridge University Press, 2002.

[40] Masud, A., Hughes, T.J.R. and Wan, J., *A stabilized mixed discontinuous Galerkin method for Darcy flow*, Computer Methods in Applied Mechanics and Engineering, vol. 195, 3347-3381, 2006.

[41] Masud, A., & Hughes, T.J.R., *A stabilized mixed finite element method for Darcy flow*, Computer Methods in Applied Mechanics and Engineering, Volume 191, Issues 39-40, 2002.

[42] MATLAB V 7.1, *Full Product Family Help*, The MathWorks, Inc., 2005.

[43] Mattax, C.C., & Dalton R.L, *Reservoir Simulation.* SPE Monograph Volume 13, Richardson, TX, 1990.

[44] Morton, K.W., & Suli, E., *Finite volume methods and their analysis*, IMA J. Numer. Anal., 11, 1991.

[45] Nakshatrala.K.B., Turner, D.Z., Hjelmstad, K.D., Masud, A., *A stabilized mixed finite element method for Darcy flow based on a multi-scale decomposition of the solution*, Computer Methods in Applied Mechanics and Engineering, Volume 195, Issues 33-36, 2006.

[46] NaturalGas.org, Exploration, 31-05-2010, <http://naturalgas.org/naturalgas/exploration.asp>

[47] Norman J. Hyne, *Nontechnical guide to petroleum geology, exploration, drilling, and production*, PennWell Books, 2001

[48] Otto, S.R., & Denier, J.P., *An Introduction to Programming and Numerical Methods in MATLAB*, Springer, 2005.

[49] Pal M., & Edwards M.G., *Non-linear flux-splitting schemes with imposed discrete maximum principle for elliptic equations with highly anisotropic coefficients,* doi:10.1002/fld.2258 to appear Int. J. Numer Meth, Fluids.

[50] Pal, M., & Edwards, M.G., *Flux-Splitting Schemes for Improved Monotonicity of Discrete Solutions of Elliptic Equations with Highly Anisotropic Coefficients*, ECCOMAS CFD 2006.

[51] Peaceman, D.W., *Fundamentals of numerical reservoir simulation*, Elsevier, 1977.5

[52] Persson, P., & Strang, G., *A Simple Mesh Generator in MATLAB*, Society for Industrial and Applied Mathematics, 2004.

[53] Recktenwald, G.W., *The Control-Volume Finite-Difference Approximation to the Diffusion Equation*, 2003

[54] Robert Eymard, Thierry Gallouet and Raphaele Herbin,' Finite Volume Methods', 31-05-2010, <http://www.cmi.univ-mrs.fr/~herbin/PUBLI/bookevol.pdf>, 2006

[55] Shamsai and Vosoughifar, H.R., *Finite Volume Discretization of Flow in Porous Media by the MATLAB System*, Scientia Iranica, Vol. 11, Nos. 1&2, 2004.

[56] Staggs, H.M., & Herbeck, E.F., *Reservoir Simulation Models – An Engineering Overview*, JPT, 1971.

[57] The MathWorks, Inc., 'User Stories by Industry', 31-05-2010, <http://www.mathworks.com/company/user_stories/industry.html>

[58] Thom and Apelt, C.J., *Field Computations in Engineering and Physics*. London: D. Van Nostrand, 1961.

[59] Thompson, J.F., Warsi, Z.U.A. & Mastin, C.W., *Numerical Grid Generation – Foundations and Applications*, Elsevier Science Publishing Co., Inc, 1985.

[60] Tyson, S., *An Introduction to Reservoir Modelling*. San Francisco: Ignatius Press, 2009.

[61] Versteeg, H.K., & Malalasekera, W., *An introduction to computational fluid dynamics: the finite volume method*, Pearson Education, 2007.

[62] Wadsley, W.A., *Modelling Reservoir Geometry with Non-Rectangular Coordinate Grid*, SPE Annual Technical Conference and Exhibition, 1980.

[63] Weck, O.D and Kim, I.Y, Finite Element Method, 12-01-2004, 31-05-2010, <http://web.mit.edu/16.810/www/16.810_L4_CAE.pdf>

[64] Wheeler, M.F., & Peszynska, M., *Computational engineering and science methodologies for modelling and simulation of subsurface applications*, Advances in Water Resources 25, 2002.

[65] White R.E., *Computational Mathematics: Models, Methods and Analysis with MATLAB and MPI*, CRC Press, 2003.

[66] Zheng, H., Edwards, M.G., & Pal, M., Flux Continuous Finite Volume Schemes with Full Pressure Continuity, Civil-Comp Press, 2007.

[67] Zheng, H., *Quasi-Positive Families of Flux Continuous Finite Volumes Schemes in Two and Three Dimensions*, Swansea University, 2010.

[68] Zheng, Y., Burridge, R., & Burns, D., *Reservoir Simulation with the Finite Element Method Using Biot Poroelastic Approach,* Earth Resources Laboratory, MIT, Cambridge, MA 02139.

APPENDIX
A. Triangular pressure support application code
a. Main File

```
%       Entry point of console application.
%     Code designed for study of Cell-centered Finite Volume Method numerical
%    evaluation of Darcy's equation for flow in porous media.
%     For unstructured triangular Mesh.
%     Triangular Pressure Support Scheme.
%    Physical Space Scheme.
%     written by : Mary Abraham Eranackal
%      Erasmus Mundus Masters in Computational Mechanics 2008-2010
%         Civil & Computational Engineering Centre, University of Wales Swansea
clear all
clc
display ('*  Unstructured Cell Centered TPS FV Scheme *')
display ('*        By Mary Abraham Eranackal         *')
%*********************************************************************
%Input value of q ( q is the distance between the cluster vertex and cell
%edge midpoint,with q=1 at the midpoint and q=0 at the cluster vertex)
q = input('Please input the value of 0<q=<1 : '); % Flux Continuity at q
%Ensuring value of q is within allowed limits
while((q<=0)||(q>1))
    fprintf('Error.  q  must  be between 0 & 1\n')
    q = input('Re-Enter q: ');
end
%*********************************************************************
%Initialize variables
casetype = 0; % Type of problem selected to be solved
%*********************************************************************
%Display the casetypes (Types of Problems)
display('1. Debug 1 phy = 2.5')
display('2.  Cartesian  mesh  to solve  linear  problem:Phi=x+y+1 (diagonal tensor)')
display('3. Discontinuous Bilinear Test Case')
display('4.    Quadratic    case: x^2+y^2 with unit permeability')
display('5.          Discontinuous Quadratic Case')
display('6. Test Case for Large Anisotropy Ratios. (taken from example 5 in PDFCS')
display('7. Circular Region with Well')
display('8. theta= pi/2,    alp= 0.53544095,k1=k3=5,k2=k4=1')
display('9.  Cartesian  mesh  to solve  highly  anisotropic  phi= exp(xy)')
display('10.  Zigzag  Test  case, with permeability of 3 domains and point source and sink')
display('11.  Square  permeability discontinuity, highly anisotropic, source')
%*********************************************************************
%Selecting the required casetype
casetype = input('Please input the casetype number selecting from above display : ');
%Ensuring casetype selected is within available types
while       ((casetype<1)       ||(casetype>40))
    fprintf('Error.casetype  value must be between 1 and 40\n');
    casetype = input('Please enter again    the    casetype    number selecting from above display : ');
end
%*********************************************************************
tic % Starting stopwatch to record simulation time
%*********************************************************************
%Read the Grid Data from file
%Opening the geometry file
fid  =  fopen('D:\Master Project\My Matlab\Cellcenter\Trial\bin2.txt', 'r');
%Reads and displays the first text line of the file
str1 = native2unicode(fgetl(fid));
```

```
%Reads and displays/stores the values in the file
A = fscanf(fid, '%f %f %f %f ',[14]);
npoint = A(1) % Number of Nodes
nelem = A(2) % Number of Elements
str2 = native2unicode(fgetl(fid)); % Reads the second text line in the file
B = fscanf(fid,'%f %f %f',[3 npoint]); % Reads the coordinates of nodes table from file
B = B'; % B contains the nodes and their coordinates
coordinate = []; % coordinates of each node point
coordinate(:,1) = B(:,2);
coordinate(:,2) = B(:,3);
str3 = native2unicode(fgetl(fid));
str4 = native2unicode(fgetl(fid));
C = fscanf(fid,'%f %f %f %f %f ',[5 , nelem]); % Reading the connectivity table
C = C';
triangle = []; % connectivity of each triangle (3 vertices)
triangle(:,1) = C(:,3);
triangle(:,2) = C(:,4);
triangle(:,3) = C(:,5);
status = fclose(fid); % closing the geometry file
%Opening the boundary file
fid = fopen('D:\Master Project\My Matlab\Cellcenter\Trial\bb2.txt','r');
%Reads and displays the first text line of the file
strb1 = native2unicode(fgetl(fid));
nface = fscanf(fid,'%f') % number of boundary segments i.e. each element face which lies on the boundary of the geometry.
strb2 = native2unicode(fgetl(fid));
D = fscanf(fid,'%f %f %f %f ',[ 4 , nface]);%Reading the boundary elements and nodes
D = D';
status = fclose(fid);%Closing the boundary file
bryelems = [];%List of boundary elements (elements having atleast 2 nodes on the boundary)
bryelems = D (:,3);
bryn1 = D(:,1); % Boundary node1
bryn2 = D(:,2); % Boundary node2
bryn = [bryn1;bryn2]; %Boundary nodes
bryn = unique(bryn); %Nodes lying on the boundary
intelems = [];%List of internal elements
E = C(:,1); %Lists the element numbers
intelems=setdiff(E,bryelems); % Gives the elements not listed in boundary element list,effectively giving us the internal elements.
cfbc = [];
for sa = 1: nelem
    sano = triangle(sa,:);
    ashi = intersect(sano,bryn);
    mizu = numel(ashi);
    if (mizu > 0)
        cfbc = [cfbc;sa];%Stores all the elements having boundary nodes
    end
end
%*********************************
**********************************
******
%Calculation of Area Parameters
%Area calculation - Start
area(:,3) = 0.0; % Initialize area of each of the 3 subcells per triangle to 0.0
for i=1:nelem
    lmd(i,2)=0;% List of midpoints of the elements or in this case grid points.
    extr = triangle(i,:); % extracting the connecting nodes of the element,i.e; vertex nodes
    % x & y coordinates of the vertices of the element
    xv1 = coordinate(extr(1),1);
    xv2 = coordinate(extr(2),1);
    xv3 = coordinate(extr(3),1);
    yv1 = coordinate(extr(1),2);
    yv2 = coordinate(extr(2),2);
    yv3 = coordinate(extr(3),2);
    % x & y coordinate of the mid-point of the element
    x1 = (xv1+xv2+xv3)/3.0;
    y1 = (yv1+yv2+yv3)/3.0;
    % x & y coordinates of the triangle edge mid-points
    xe = (0.50)*( xv1 + xv2);
    xn = (0.50)*( xv2 + xv3);
    xs = (0.50)*( xv3 + xv1);
    ye = (0.50)*( yv1 + yv2);
    yn = (0.50)*( yv2 + yv3);
    ys = (0.50)*( yv3 + yv1);
    %List the midpoints
    lmd(i,1) = x1;
    lmd(i,2) = y1;
    %Finding the subcell areas of each triangle
```

```
    area(i,1)                       =
area_sub(xv3,xs,x1,xn,yv3,ys,y1,yn
);
    area(i,2)                       =
area_sub(xs,xv1,xe,x1,ys,yv1,ye,y1
);
    area(i,3)                       =
area_sub(xv2,xn,x1,xe,yv2,yn,y1,ye
);
end
%Finding    the    cluster    area
associated with each vertex
for i = 1:npoint
    ad(i)=0; % counts the number
of elements associated with each
vertex
    clusterarea(i)=0.0;   %   area
associated with each cluster
    for k = 1:nelem
        if    ((triangle(k,1)    ==
i)||(triangle(k,2)              ==
i)||(triangle(k,3) == i))
            ad(i)=ad(i)+1;
            clusterarea(i)       =
clusterarea(i) + area(k,1);% Gives
the cluster area associated with
each vertex i
        end
    end
end
%Displaying   the   Cell   Centered
Scheme Control Volumes for each
vertex
for P=1:npoint
    fprintf('For               node
%d,associated number of triangular
elements are: %d\n',P,ad(P));
    fprintf('Control   Volume   for
node     %d     is     :     %f
\n',P,clusterarea(P));
end
%Finding the total area of all the
elements in the geometry
totarea  =  0.0;  %Total  geometry
area
for i = 1:npoint
    totarea     =     totarea     +
clusterarea(i);
end
fprintf('Total    Area    of    the
Geometry is : %f\n',totarea);
%Area calculation - End
%*****************************
*****************************
*******
%Assign the Permeability and Start
Permeability calculation
unitI = [1.0 0.0 0.0 1.0];
switch casetype
    case (1)
        for I = 1:nelem
            permk(I,:)=[1,0,0,1];
        end
    case (2)
        for I = 1:nelem
            permk(I,:)=[1,0,0,1];
        end
    case (3)
        for I=1:nelem
            if (lmd(I,1)<=0.5)

permk(I,:)=[1,0.5,0.5,1];
            elseif (lmd(I,1)>0.5)

permk(I,:)=[10,2,2,100];
            end
        end
    case (4)
        for I = 1:nelem
            permk(I,:) = unitI;
        end
    case (5)
        for I=1:nelem
            if (lmd(I,1)<0.5)
                permk(I,:)       =
[50,0,0,1];
            elseif (lmd(I,1)>=0.5)

permk(I,:)=[1,0,0,10];
            end
        end
    case (6)
        %co = constant as defined
in  the  problem.  For  anisotropic
problem,
        %co  lies  between  0  and
1.At   co=   1,   problem   becomes
isotropic.As co
        %decreases,    degree    of
anisotropy increases.
        %co=1;
        %         co=0.1;
        co=0.01;
        for I = 1:nelem
            permk(I,:)             =
[(lmd(I,2)^2+co*(lmd(I,1)^2)),-(1-
co)*lmd(I,1)*lmd(I,2),-(1-
co)*lmd(I,1)*lmd(I,2),(lmd(I,1)^2+
co*(lmd(I,2)^2))];
        end
    case (7)
        for I = 1:nelem
            permk(I,:) = unitI;
        end
    case (8)
        for i = 1:nelem
            rx(i) = lmd(i,1)-0.5;
            ry(i) = lmd(i,2)-0.5;
            ra(i)   =   sqrt(rx(i)^2
+ry(i)^2);
            theta(i)               =
atan2(ry(i),rx(i));
```

```
            if ( ry(i) < 0.0 )
                theta(i)          =
theta(i)+ pi*2;
            end
        end
        for I = 1:nelem
            if ((theta(I) <pi*0.5)
& (theta(I) >= 0.0) )
                permk(I,:) = 5.0 *
unitI;
            elseif  ((theta(I)  >=
pi*0.5) & (theta(I) < pi) )
                permk(I,:)        =
unitI;
            elseif  ((theta(I)  >=
pi) & (theta(I) < pi*(3/2)) )
                permk(I,:)  =  5.0*
unitI;
            elseif  ((theta(I)  >=
pi*(3/2)) & (theta(I) < pi*2) )
                permk(I,:)        =
unitI;
            end
        end
    case(9)
        for I = 1:nelem
            permk(I,:)            =
[750.25,432.58, 432.58,250.75];
        end
    case (10)
        for I = 1:nelem
            if((     lmd(I,2)     <
0.333333)     ||     (lmd(I,2)     >
0.6666666667) )
                permk(I,:)=[
2464.360020,          1148.683643,
1148.683643, 536.6399794];
            else
                permk(I,:)=[
2464.360020,     -1148.683643,     -
1148.683643, 536.6399794];
            end
        end
    case (11)
        for I=1:nelem
            if    ((lmd(I,1)>-0.25)
&& (lmd(I,1)<0.25) && (lmd(I,2)>-
0.25) && (lmd(I,2)<0.25))
                permk(I,:)=[
2464.360020,     -1148.683643,     -
1148.683643, 536.6399794];
            else
                permk(I,:)=[
2464.360020,          1148.683643,
1148.683643, 536.6399794];
            end
        end

end
%End    of    permeability    matrix
assignment
```

```
%*********************************
*********************************
******
%Assigning the values of pressures
at  the  cell  centers  for  each
casetype
%Defines  the  analytical  pressure
values  at  the  centre  of  each
triangular  element  since  it  is  a
cell centered scheme
switch casetype
    case (1)
        Phi_exact(1:nelem) =2.50;
        Load_vector(1:nelem)      =
0.0;
    case (2)
        for I = 1:nelem
            Phi_exact(I)          =
lmd(I,1) + lmd(I,2) +1.0;
        end
        Load_vector(1:nelem) = 0;
    case (3)
        for j=1:nelem
            Area(j)=
area(j,1)+area(j,2)+area(j,3);
            if (lmd(j,1)<=0.5)
                Phi_exact(j)      =
10+20*lmd(j,1)*lmd(j,2);
                Load_vector(j)    =
20*Area(j);
            elseif (lmd(j,1)>0.5)
                Phi_exact(j)      =
10.75-
1.5*lmd(j,1)+9*lmd(j,2)+2*lmd(j,1)
*lmd(j,2);
                Load_vector(j)    =
8*Area(j);
            end
        end
    case (4)
        for I = 1:nelem
            Phi_exact(I)          =
lmd(I,1)^2+lmd(I,2)^2;
            Area(I)=
area(I,1)+area(I,2)+area(I,3);
            Load_vector(I)        =
4*Area(I);
        end
    case (5)
        qw = 1/50;
        er = 1/10;
        rt = 1;
        ty = 4*rt/((qw-2)*er+1);
        yu = (er-1)*ty;
        ui = ty;
        io = -ui*(1/10);
        cl = qw*er*ui;
        dl = io;
        for j=1:nelem
            Area(j)=
area(j,1)+area(j,2)+area(j,3);
```

```
            if (lmd(j,1)<0.5)
                Phi_exact(j)      =
cl*(lmd(j,1))^2+dl*(lmd(j,2))^2;
                Load_vector(j)    =
0*Area(j);
            elseif (lmd(j,1)>=0.5)
                Phi_exact(j)      =
rt+yu*lmd(j,1)+ui*(lmd(j,1))^2+io*
(lmd(j,2))^2;
                Load_vector(j)    =
0*Area(j);
            end
        end
    case (6)
        for I = 1:nelem
            Phi_exact(I)          =
sin(pi*lmd(I,1))*sin(pi*lmd(I,2));
            Area(I)=
area(I,1)+area(I,2)+area(I,3);
            Load_vector(I)    =    -
1*Area(I)*(-
2*co*lmd(I,1)*cos(pi*lmd(I,1))*pi*
sin(pi*lmd(I,2))+(lmd(I,2)^2+co*lm
d(I,1)^2)*sin(pi*lmd(I,1))*pi^2*si
n(pi*lmd(I,2))-(-
1+co)*lmd(I,2)*sin(pi*lmd(I,1))*co
s(pi*lmd(I,2))*pi-2*(-
1+co)*lmd(I,1)*lmd(I,2)*cos(pi*lmd
(I,1))*pi^2*cos(pi*lmd(I,2))-(-
1+co)*lmd(I,1)*cos(pi*lmd(I,1))*pi
*sin(pi*lmd(I,2))-
2*co*lmd(I,2)*sin(pi*lmd(I,1))*cos
(pi*lmd(I,2))*pi+(lmd(I,1)^2+co*lm
d(I,2)^2)*sin(pi*lmd(I,1))*sin(pi*
lmd(I,2))*pi^2);
        end
    case (7)
        for I=1:nelem
            Phi_exact(I)=(2-
1)*(log((sqrt(((lmd(I,1))^2+(lmd(I
,2))^2)/0.5))))/log(0.001/0.5)+1;
            Area(I)=
area(I,1)+area(I,2)+area(I,3);

Load_vector(I)=Area(I)*(.32182/((l
md(I,1))^2+(lmd(I,2))^2)^2*(lmd(I,
1))^2-
.32182/((lmd(I,1))^2+(lmd(I,2))^2)
+.32182/((lmd(I,1))^2+(lmd(I,2))^2
)^2*(lmd(I,2))^2);
        end
    case (8)       % this case is
for Pi/2 ,
        for I = 1:nelem
            rx(I) = lmd(I,1)-0.5;
            ry(I) = lmd(I,2)-0.5;
            ra(I) = sqrt(rx(I)^2 +
ry(I)^2 );
            ptheta(I)             =
atan2(ry(I),rx(I));
            if ( ry(I) < 0.0 )
                ptheta(I)         =
ptheta(I)+ pi*2;
            end
        end
        alp      =      0.53544095;
% k1=k3=5; k2=k4=1;
        delt_alp = alp -1.0;
        for I = 1:nelem
            if          ((ptheta(I)
<pi*0.5) & (ptheta(I) >= 0.0) )  %
K1
                m1 = 0.44721360;
                m2 = 1.0;
            elseif  ((ptheta(I)  >=
pi*0.5) & (ptheta(I) < pi) ) % K2
                m1 = -0.74535599;
                m2 = 2.33333333;
            elseif  ((ptheta(I)  >=
pi) & (ptheta(I) < pi*(3/2)) ) %
K3
                m1 = -0.94411759;
                m2 =  0.5555556;
            elseif  ((ptheta(I)  >=
pi*(3/2)) & (ptheta(I) < pi*2) ) %
K4
                m1 = -2.40170246;
                m2 = -0.481481481;
            end
            Phi_exact(I)  =  (ra(I)
^ alp)*( m1* sin(alp*ptheta(I)) +
m2* cos(alp*ptheta(I))) ;
        end
        Load_vector(1:nelem)      =
0.0;
    case (9)
        for I =1:nelem
            Phi_exact(I)          =
exp(lmd(I,1)*lmd(I,2));
            Area(I)=
area(I,1)+area(I,2)+area(I,3);
            Load_vector(I)=
Area(I)*(3001/4*lmd(I,2)^2*exp(lmd
(I,1)*lmd(I,2))+21629/25*exp(lmd(I
,1)*lmd(I,2))+21629/25*lmd(I,1)*lm
d(I,2)*exp(lmd(I,1)*lmd(I,2))+1003
/4*lmd(I,1)^2*exp(lmd(I,1)*lmd(I,2
)));
        end
    case (10)
        Phi_exact(1:nelem)= 100.0;
        Phi_exact(615) = 0.0;
        Phi_exact(538) = 200.0;
     Load_vector(1:nelem) = 0.0  ;
    case (11)
        Phi_exact(1:nelem) = 0;
        Phi_exact(648) = 1.0;
     Load_vector(1:nelem) = 0.0  ;
end
%End of exact cell center pressure
value assignment
```

```
%*********************************
*********************************
******
%Start    calculation    of    Flux
equations at each continuity point
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<
%Finding  the  elements  associated
with each cluster vertex
for i = 1:npoint
    ad(i)=0; % counts the number
of elements associated with each
vertex
    for k = 1:nelem
        if    ((triangle(k,1)    ==
i)|||(triangle(k,2)               ==
i)|||(triangle(k,3) == i))
            ad(i)=ad(i)+1;
            adel(i,ad(i))         =
k;%stores the associated elements
in a npoint row matrix
        end
    end
end
elmofv = sparse(adel);% Gives the
associated  elements  for  each  of
the npoint nodes.
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<
%  Using  external  function  to
obtain the neighbouring elements
% Create the triangle neighbour
array.
triangle_order = 3;
triangle_neighbor                 =
triangulation_neighbor_triangles  (
triangle_order,nelem,triangle);
ham = triangle_neighbor';
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>
%Initialising local parameters
Amat   =   zeros(nelem);   %Global
Coefficient Matrix
Fmat = Load_vector'; %Global Force
Vector
%Starting calculations cluster by
cluster
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<
for i=1:npoint

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<
    %Ordering adjacent elements in
cluster i
    Wam=[];%Stores     neighbouring
elements of each cluster in anti-
clockwise direction.
    if (ad(i)>2)
        for h = 1:ad(i)
            seth                  =
elmofv(i,:);%Locally    extracting
associated elements
            jam = seth(1,h);
            dam = ham(jam,:);
            dam(dam == 0) = [];
            sam                   =
intersect(dam,seth);
            tam = size(sam);
            if (tam(1,2)==2)
                wam   =   [sam(1,1)
jam sam(1,2)];
                if (ad(i)>3)
                    break
                end
            else
                continue
            end
            Wam = [Wam wam];
        end
        if (ad(i)>3)
            for n=3:ad(i)
                rat = wam(end);
                dam = ham(rat,:);
                dam(dam  ==  0)  =
[];
                sam               =
intersect(dam,seth);
                tam = size(sam);
                if (tam(1,2)==2)
                    if
(sam(1,1)==wam(1,n-1));
                        wam = [wam
sam(1,2)];
                    else
                        wam = [wam
sam(1,1)];
                    end
                end
                Wam = [Wam wam];
            end
        end
    elseif (ad(i)==2)
        wama = elmofv(i,1);
        wamb = elmofv(i,2);
        Wam = [Wam wama wamb];
    elseif (ad(i)==1)
        Wam = elmofv(i,:);
    end
    if (ad(i)>2)
        ani = ad(i)-1;
        Wam= Wam(end-(ani):end);
    end
    if (ad(i)>1)
        moch                      =
mordc(i,Wam,coordinate,triangle,q,
nelem,lmd,npoint);
        if (moch==1)
            Wam=Wam(end:-1:1);
        end
```

```
        %          newa = Wam;
    end

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>
    %Initialising Variables
    %     elco = [];
    Tl = [];
    Tr = [];

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<
    %Calculating   local   general
tensor T
    for j = 1:ad(i)
        elma = Wam(j); % Element 1
to  Element  ad(i)  in  the  cluster
taken separately
        Kel(1,:)                   =
permk(elma,:);%local        element
permeability
        [T11  T12a  T12b  T22  aref]=
flux(i,elma,coordinate,triangle,q,
nelem,lmd,npoint,Kel);
        %          elco(j,:) = [T11
T12a    T12b    T22];%Lists    the
coefficients  of  pressures  in  each
subcell
        Tl(j,:) = [T11 T12a];
        Tr(j,:) = [T12b T22];
    end

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<
    %Creating symbolic phy vector.
    phy = sym(zeros(1, ad(i)));
    for j = 1:ad(i)
        dal = Wam(j);
        phy(j)                     =
sym(sprintf('phy%d', dal));
    end

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<
    %Creating   symbolic   interface
pressure   value   vector   for   the
cluster i
    brchk=ismember(bryn,i);%Does
node i belong to boundary nodes?
    brchk = any(brchk);
    if (brchk ==1)
        phyI                       =
sym(zeros(1,(ad(i)+1)));
        for j = 1:(ad(i)+1)
            phyI(j)                =
sym(sprintf('phyI%d', j));
        end
    else
        phyI     =     sym(zeros(1,
ad(i)));
        for j = 1:(ad(i))
            phyI(j)                =
sym(sprintf('phyI%d', j));
        end
    end

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>
    Jazz=[];  %  array   of   flux
equations in a cluster
    if (ad(i)>2)
        if (brchk==0)
            for j = 1:ad(i)
                r=j+1;
                if (j==1)
                    syms a b c d e
f g h z y x l
                    jazze    =    (-
1*(a*(b-c)  +  d*(e-f)))-(g*(h-z)  +
y*(x-l));  %  Starting  Flux  in  the
cluster
                    jazze          =
subs(jazze,[a,d,g,y],[Tl(j,1),Tl(j
,2),Tr(r,1),Tr(r,2)]);
                    jazze          =
subs(jazze,[b,c,e,f,h,z,x,l],[phyI
(j),phy(j),phyI(ad(i)),phy(j),phyI
(r),phy(r),phyI(j),phy(r)]);
                    Jazz           =
[Jazz;jazze];
                elseif(j<ad(i))
                    syms a b c d e
f g h z y x l
                    jazz     =    (-
1*(a*(b-c)  +  d*(e-f)))-(g*(h-z)  +
y*(x-l));  %Middle  flux  terms  in
the cluster
                    jazz           =
subs(jazz,[a,d,g,y],[Tl(j,1),Tl(j,
2),Tr(r,1),Tr(r,2)]);
                    jazz           =
subs(jazz,[b,c,e,f,h,z,x,l],[phyI(
j),phy(j),phyI(j-
1),phy(j),phyI(r),phy(r),phyI(j),p
hy(r)]);
                    Jazz           =
[Jazz;jazz];
                elseif(j==ad(i))
                    syms a b c d e
f g h z y x l
                    jazzw    =    (-
1*(a*(b-c)  +  d*(e-f)))-(g*(h-z)  +
y*(x-l));  %  Last  flux  in  the
cluster
```

```
                    jazzw        =
subs(jazzw,[a,d,g,y],[Tl(j,1),Tl(j
,2),Tr(1,1),Tr(1,2)]);
                    jazzw        =
subs(jazzw,[b,c,e,f,h,z,x,l],[phyI
(j),phy(j),phyI(j-
1),phy(j),phyI(1),phy(1),phyI(j),p
hy(1)]);
                    Jazz         =
[Jazz;jazzw];
                end
            end
        elseif(brchk==1)
            for j = 1:(ad(i)+1)
                r = j+1;
                if (j==1)
                    syms a b c d e
f g h z y x l
                    jazze        =
((a*(b-c) + d*(e-f))); % Starting
Flux in the cluster
                    jazze        =
subs(jazze,[a,d],[Tr(j,1),Tr(j,2)]
);
                    jazze        =
subs(jazze,[b,c,e,f],[phyI(r),phy(
j),phyI(j),phy(j)]);
                    Jazz         =
[Jazz;jazze];
                elseif
(j<(ad(i)+1))
                    if (j==ad(i))
                        syms a b c
d e f g h z y x l
                        jazz  = (-
1*(a*(b-c) + d*(e-f)))-(g*(h-z) +
y*(x-l));  %Middle  flux  terms  in
the cluster
                        jazz     =
subs(jazz,[a,d,g,y],[Tl((j-
1),1),Tl((j-
1),2),Tr(j,1),Tr(j,2)]);
                        jazz     =
subs(jazz,[b,c,e,f,h,z,x,l],[phyI(
j),phy(j-1),phyI(j-1),phy(j-
1),phyI(r),phy(j),phyI(j),phy(j)])
;
                        Jazz     =
[Jazz;jazz];
                    else
                        syms a b c
d e f g h z y x l
                        jazz  = (-
1*(a*(b-c) + d*(e-f)))-(g*(h-z) +
y*(x-l));  %Middle  flux  terms  in
the cluster
                        jazz     =
subs(jazz,[a,d,g,y],[Tl((j-
1),1),Tl((j-
1),2),Tr(j,1),Tr(j,2)]);
                        jazz     =
subs(jazz,[b,c,e,f,h,z,x,l],[phyI(
j),phy(j-1),phyI(j-1),phy(j-
1),phyI(r),phy(j),phyI(j),phy(j)])
;
                        Jazz     =
[Jazz;jazz];
                end
            elseif
(j==(ad(i)+1))
                    syms a b c d e
f g h z y x l
                    jazzw   =   (-
1*(a*(b-c) + d*(e-f))); % Closing
Flux in the cluster
                    jazzw        =
subs(jazzw,[a,d],[Tl((j-
1),1),Tl((j-1),2)]);
                    jazzw        =
subs(jazzw,[b,c,e,f],[phyI(j),phy(
j-1),phyI(j-1),phy(j-1)]);
                    Jazz         =
[Jazz;jazzw];
                end
            end
        end
    elseif (ad(i)==2)
        syms a b c d e f g h z y x
l
        jazz   =   (-1*(a*(b-c)   +
d*(e-f)))+(g*(h-z) + y*(x-l));
        jazz                     =
subs(jazz,[a,d,g,y],[Tl(1,1),Tl(1,
2),Tr(2,1),Tr(2,2)]);
        jazz                     =
subs(jazz,[b,c,e,f,h,z,x,l],[phyI(
2),phy(1),phyI(1),phy(1),phyI(3),p
hy(2),phyI(2),phy(2)]);
        jazz1   =   (-1*(a*(b-c)   +
d*(e-f))); %Flux Leaving
        jazzw = (g*(h-z) + y*(x-
l)); % Flux Entering
        jazz1                    =
subs(jazz1,[a,d],[Tl(2,1),Tl(2,2)]
);
        jazz1                    =
subs(jazz1,[b,c,e,f],[phyI(3),phy(
2),phyI(2),phy(2)]);
        jazzw                    =
subs(jazzw,[g,y],[Tr(1,1),Tr(1,2)]
);
        jazzw                    =
subs(jazzw,[h,z,x,l],[phyI(2),phy(
1),phyI(1),phy(1)]);
        jazz = [jazzw;jazz;jazz1];
        Jazz=[Jazz;jazz];
    elseif (ad(i)==1)
        syms a b c d e f g h z y x
l
        jazz1   =   (-1*(a*(b-c)   +
d*(e-f))); %Flux Leaving
```

```
        jazzw = (g*(h-z) + y*(x-
l)); % Flux Entering
        jazz1                    =
subs(jazz1,[a,d],[Tl(1,1),Tl(1,2)]
);
        jazz1                    =
subs(jazz1,[b,c,e,f],[phyI(2),phy(
1),phyI(1),phy(1)]);
        jazzw                    =
subs(jazzw,[g,y],[Tr(1,1),Tr(1,2)]
);
        jazzw                    =
subs(jazzw,[h,z,x,l],[phyI(2),phy(
1),phyI(1),phy(1)]);
        jazz = [jazzw;jazz1];
        Jazz=[Jazz;jazz];
    end

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<
    %Solving     for     interface
pressures   in   terms   of   nodal
pressures.
    nochk  =  numel(Jazz);  %Number
Check
    if (nochk==2)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),phyI(1),
phyI(2));
        itnc                     =
[chat.phyI1;chat.phyI2];
    elseif (nochk==3)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),phyI(1),phyI(2),phyI(3));
        itnc                     =
[chat.phyI1;chat.phyI2;chat.phyI3]
;
            elseif (nochk==4)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),phyI(1),phyI(2),phyI(3
),phyI(4));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4];
    elseif (nochk==5)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),Jazz(5,1),phyI(1),phyI
(2),phyI(3),phyI(4),phyI(5));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4;chat.phyI5];
    elseif (nochk==6)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),Jazz(5,1),Jazz(6,1),ph
yI(1),phyI(2),phyI(3),phyI(4),phyI
(5),phyI(6));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4;chat.phyI5;chat.phyI6]
;
    elseif (nochk==7)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),Jazz(5,1),Jazz(6,1),Ja
zz(7,1),phyI(1),phyI(2),phyI(3),ph
yI(4),phyI(5),phyI(6),phyI(7));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4;chat.phyI5;chat.phyI6;
chat.phyI7];
    elseif (nochk==8)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),Jazz(5,1),Jazz(6,1),Ja
zz(7,1),Jazz(8,1),phyI(1),phyI(2),
phyI(3),phyI(4),phyI(5),phyI(6),ph
yI(7),phyI(8));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4;chat.phyI5;chat.phyI6;
chat.phyI7;chat.phyI8];
    elseif (nochk==9)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),Jazz(5,1),Jazz(6,1),Ja
zz(7,1),Jazz(8,1),Jazz(9,1),phyI(1
),phyI(2),phyI(3),phyI(4),phyI(5),
phyI(6),phyI(7),phyI(8),phyI(9));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4;chat.phyI5;chat.phyI6;
chat.phyI7;chat.phyI8;chat.phyI9];
    elseif (nochk==10)
        chat                     =
solve(Jazz(1,1),Jazz(2,1),Jazz(3,1
),Jazz(4,1),Jazz(5,1),Jazz(6,1),Ja
zz(7,1),Jazz(8,1),Jazz(9,1),Jazz(1
0,1),phyI(1),phyI(2),phyI(3),phyI(
4),phyI(5),phyI(6),phyI(7),phyI(8)
,phyI(9),phyI(10));
        itnc
=[chat.phyI1;chat.phyI2;chat.phyI3
;chat.phyI4;chat.phyI5;chat.phyI6;
chat.phyI7;chat.phyI8;chat.phyI9,c
hat.phyI10];
    end

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<
    %Defining   the   unknown   flux
equations at each continuity point
in
    %cluster.
```

```
    Fluxv=[];  % array of flux values at the interfaces in a cluster
    if (ad(i)>2)
        if (brchk==0)
            for j = 1:ad(i)
                r=j+1;
                if (j==1)
                    syms a b c d e f g h z y x l
                    fluxv   =   (-1*(a*(b-c) + d*(e-f))); % Starting Flux in the cluster
                    fluxv       = subs(fluxv,[a,d],[Tl(j,1),Tl(j,2)]);
                    fluxv       = subs(fluxv,[b,c,e,f],[itnc(j),phy(j),itnc(ad(i)),phy(j)]);
                    Fluxv       = [Fluxv;fluxv];
                elseif(j<ad(i))
                    syms a b c d e f g h z y x l
                    fluxv   =   (-1*(a*(b-c)  +  d*(e-f)));  %Middle flux terms in the cluster
                    fluxv       = subs(fluxv,[a,d],[Tl(j,1),Tl(j,2)]);
                    fluxv       = subs(fluxv,[b,c,e,f],[itnc(j),phy(j),itnc(j-1),phy(j)]);
                    Fluxv       = [Fluxv;fluxv];
                elseif(j==ad(i))
                    syms a b c d e f g h z y x l
                    fluxvw   =   (-1*(a*(b-c)  +  d*(e-f)));  %  Last flux in the cluster
                    fluxvw      = subs(fluxvw,[a,d],[Tl(j,1),Tl(j,2)]);
                    fluxvw      = subs(fluxvw,[b,c,e,f],[itnc(j),phy(j),itnc(j-1),phy(j)]);
                    Fluxv       = [Fluxv;fluxvw];
                end
            end
        elseif(brchk==1)
            for j = 1:(ad(i)+1)
                r = j+1;
                if (j==1)
                    syms a b c d e f g h z y x l
                    fluxve      = ((a*(b-c) + d*(e-f))); % Starting Flux in the cluster
                    fluxve      = subs(fluxve,[a,d],[Tr(j,1),Tr(j,2)]);
                    fluxve      = subs(fluxve,[b,c,e,f],[itnc(r),phy(j),itnc(j),phy(j)]);
                    Fluxv       = [Fluxv;fluxve];
                elseif (j<(ad(i)+1))
                    syms a b c d e f g h z y x l
                    fluxv   =   (-1*(a*(b-c)  +  d*(e-f)));  %Middle flux terms in the cluster
                    fluxv       = subs(fluxv,[a,d],[Tl((j-1),1),Tl((j-1),2)]);
                    fluxv       = subs(fluxv,[b,c,e,f],[itnc(j),phy(j-1),itnc(j-1),phy(j-1)]);
                    Fluxv       = [Fluxv;fluxv];
                elseif (j==(ad(i)+1))
                    syms a b c d e f g h z y x l
                    fluxvw   =   (-1*(a*(b-c) + d*(e-f))); % Closing Flux in the cluster
                    fluxvw      = subs(fluxvw,[a,d],[Tl((j-1),1),Tl((j-1),2)]);
                    fluxvw      = subs(fluxvw,[b,c,e,f],[itnc(j),phy(j-1),itnc(j-1),phy(j-1)]);
                    Fluxv       = [Fluxv;fluxvw];
                end
            end
        end
    elseif (ad(i)==2)
        syms a b c d e f g h z y x l
        fluxv   =   (-1*(a*(b-c)  +  d*(e-f)));
        fluxv                   = subs(fluxv,[a,d],[Tl(1,1),Tl(1,2)]);
        fluxv                   = subs(fluxv,[b,c,e,f],[itnc(2),phy(1),itnc(1),phy(1)]);
        fluxv1   =   (-1*(a*(b-c)  +  d*(e-f))); %Flux Leaving
        fluxvw = (g*(h-z) + y*(x-l)); % Flux Entering
        fluxv1                  = subs(fluxv1,[a,d],[Tl(2,1),Tl(2,2)]);
```

```
        fluxv1                  =
subs(fluxv1,[b,c,e,f],[itnc(3),phy
(2),itnc(2),phy(2)]);
        fluxvw                  =
subs(fluxvw,[g,y],[Tr(1,1),Tr(1,2)
]);
        fluxvw                  =
subs(fluxvw,[h,z,x,l],[itnc(2),phy
(1),itnc(1),phy(1)]);
        fluxv                   =
[fluxvw;fluxv;fluxv1];
        Fluxv=[Fluxv;fluxv];
    elseif (ad(i)==1)
        syms a b c d e f g h z y x
l
        fluxv1  =  (-1*(a*(b-c)  +
d*(e-f))); %Flux Leaving
        fluxvw = (g*(h-z) + y*(x-
l)); % Flux Entering
        fluxv1                  =
subs(fluxv1,[a,d],[Tl(1,1),Tl(1,2)
]);
        fluxv1                  =
subs(fluxv1,[b,c,e,f],[itnc(2),phy
(1),itnc(1),phy(1)]);
        fluxvw                  =
subs(fluxvw,[g,y],[Tr(1,1),Tr(1,2)
]);
        fluxvw                  =
subs(fluxvw,[h,z,x,l],[itnc(2),phy
(1),itnc(1),phy(1)]);
        fluxv = [fluxvw;fluxv1];
        Fluxv=[Fluxv;fluxv];
    end
    Fluxv = vpa(Fluxv,25);

%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>
    keol  =  numel(Fluxv);  %Number
of flux equations in each cluster
    %       fnox(i)  =  keol;  %For
external use.

%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<
    %Extracting   local   cluster
coeffs to global coeff matrix.
    for g=1:keol % taking all the
flux equations in each cluster one
by one
        for     r=     1:ad(i)     %
considering   all   the   elemental
pressure values one by one eg phy1
,phy13 etc
            amat = zeros(nelem); %
initialising   the   coeff   matrix
taken locally to zeros
            if (Fluxv(g,1)~=0)
                [go,mi]           =
coeffs(Fluxv(g,1),phy(r));       %
extracting   the   coefficients   of
each   pressure   in   the   flux   eqn
being considered
                if  (numel(mi)  ==
2)  %  if  the  flux  equation
considered is not merely a zero
                    if  (brchk  ==
0) % If the cluster vertex is not
on   the   boundary,   number   of
elements == number of equations
                        %  =>  keol
= ad(i)
                        if
(g+1<=ad(i))

amat(Wam(g+1),Wam(r))=go(1,2);

amat(Wam(g),Wam(r))    =    -1    *
go(1,2);
                            Amat  =
Amat+amat;
                        elseif  (g
== ad(i))

amat(Wam(g),Wam(r))=-go(1,2);

amat(Wam(1),Wam(r)) =  go(1,2);
                            Amat  =
Amat+amat;
                        end
                    elseif  (brchk
== 1) % If the cluster contains
boundary elements, keol = ad(i)+1
                        if (g==1)

amat(Wam(g),Wam(r)) = go(1,2);
                            Amat  =
Amat+amat;
                        elseif
(g<=ad(i))

amat(Wam(g),Wam(r)) = go(1,2);

amat(Wam(g-1),Wam(r))      =      -
1*go(1,2);
                            Amat  =
Amat + amat;
                        elseif  (g
== ad(i)+1)

amat(Wam(ad(i)),Wam(r))     =     -
1*go(1,2);
                            Amat  =
Amat + amat;
                        end
                    end
                else
                    continue
                end
            end
        end
    end
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>
end
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>
% chkmat = Amat;
% cfmat = Fmat;
%Initialise  approximate  pressure
vector
for d = 1 : nelem
    pmat(d,1)                    =
sym(sprintf('phy%d', d));
end
%**********************************
**********************************
*******
% Inputting Boundary Condition
% Zeros and ones
for w = 1:nelem
    brchk=ismember(cfbc,w);
    brchk = any(brchk);
    if (brchk ==1)
        Amat(w,:) = 0;
        Amat(w,w) = 1;
        Fmat(w,1)                =
Phi_exact(1,w);
    end
end
% Input Source or Sink Condition
within the domain
% Amat(648,:)=0;
% Amat(648,648)=1;
% Fmat(648,1)=Phi_exact(1,648);
%**********************************
**********************************
******
%Solving for pressure values
pmat = Amat\Fmat;
% pmat = gmres(Amat,Fmat);
%**********************************
**********************************
******
toc
%**********************************
**********************************
******
%Postprocessing
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<
%Contour Mapping of Pressure
figure(1)
X = lmd(:,1);
Y = lmd(:,2);
Zh = pmat;
[XI,YI]                          =
meshgrid(linspace(min(coordinate(:
,1)),max(coordinate(:,1)),nelem),l
inspace(min(coordinate(:,2)),max(c
oordinate(:,2)),nelem));
ZI = griddata(X,Y,Zh,XI,YI);
mesh(XI,YI,ZI), hold on
plot3(X,Y,Zh,'o'),   hold   on   %
Numerical solution Plot
xlabel('X')
ylabel('Y')
zlabel('Pressure')
Z = Phi_exact';
[XI,YI]                          =
meshgrid(linspace(min(coordinate(:
,1)),max(coordinate(:,1)),nelem),l
inspace(min(coordinate(:,2)),max(c
oordinate(:,2)),nelem));
ZI = griddata(X,Y,Z,XI,YI);
mesh(XI,YI,ZI), hold on
plot3(X,Y,Z,'x') % Exact solution
Plot
hold off
figure(2)
[XI,YI]                          =
meshgrid(linspace(min(coordinate(:
,1)),max(coordinate(:,1)),nelem),l
inspace(min(coordinate(:,2)),max(c
oordinate(:,2)),nelem));
ZI = griddata(X,Y,Zh,XI,YI);
contour(XI,YI,ZI)
xlabel('X')
ylabel('Y')
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<
%  %Plotting  Obtained  approximate
pressure values and exact pressure
values.
figure(3)
plot(pmat,('o-r'))
hold on
plot(Phi_exact,'x-b')
%
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>
%
%<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
<<<<<<<<<<<<<<<<<<<<<<<
% %L2 Norm and Max-Norm
Pe = Phi_exact';
numer = 0;
mxnorm = 0;
for c= 1: nelem
    Lerr(c)= (pmat(c)-Pe(c))^2;
    Area(c)=
area(c,1)+area(c,2)+area(c,3);
    numerL(c) = Lerr(c)*Area(c);
    numer = numer+numerL(c);
    mntemp = abs(pmat(c)-Pe(c)); %
Temporarily store the norms
    if (mntemp>mxnorm)
        mxnorm = mntemp;
    end
end
L2norm = sqrt(numer/totarea)
```

```
mxnorm
%
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
>>>>>>>>>>>>>>>>>>>>>>>
```

b. Function Files

i. To calculate the sub-cell areas

```
function [ar] = area_sub(x1,x2,x3,x4,y1,y2,y3,y4)
%Function to calculate the quad area of each subcell in a triangle.
%gives the resultant area 'ar' for given input of 4 quad corner point coordinates
var1 = 0.0;
var2 = 0.0;
var1 = 0.5d0*( (x1-x3)*(y2-y3) - (x2-x3)*(y1-y3));
var2 = 0.5d0*( (x1-x4)*(y3-y4) - (x3-x4)*(y1-y4));
if (var1 < 0.0)
    display('Area Calculation error,--var1--- the grid is not arranged anti-clockwise!!!');
end
if (var2 < 0.0)
    display('Area Calculation error,--var2--- the grid is not arranged anti-clockwise!!!');
end
ar = var1 + var2;
```

ii. To determine the neighbouring elements

External MATLAB code used.
Author : John Burkardt. The code used is distributed under the GNU LGPL license.

iii. To determine the order and assign anti-clockwise direction to cluster element order

```
function moch = mordc( i,Wam,coordinate,nelem,lmd)
%to check the counter clockwise order of cluster
% find x1*y2 - y1*x2
% where (x1, y1) is the difference calculated for point 1
% similarly for point 2, 3, etc.
% the basic idea is, no two consecutive points project an angle greater than 180 at the mid-point
% so, if they're in counter-clockwise order, the cross product would be in positive z direction
cv = [coordinate(i,1) coordinate(i,2)];
ch = [(cv(1,1)-lmd(Wam(1),1)) cv(1,2)-lmd(Wam(1),2);...
    (cv(1,1)-lmd(Wam(2),1)) cv(1,2)-lmd(Wam(2),2);...
    ];
nah12 = ch(1,1)*ch(2,2) - ch(1,2)* ch(2,1);
if (nah12 > 0)
    moch = 0;
elseif (nah12 < 0)
    moch =1;
end
```

iv. To generate the general tensor components

```
function [T11 T12a T12b T22 aref] = flux(i,elma,coordinate,triangle,q, nelem,lmd,npoint,Kel)
%Function code to calculate subcell fluxes at the continuity points on the subinterfaces.
conn = [];
conn = triangle(elma,:); % connectivity of element
arrcord(:,:) = coordinate(conn,:);% arranged coordinates
% Ensuring vertex2 is the cluster vertex always.
if (arrcord(1,:) == coordinate(i,:))
    c(1,:)=arrcord(1,:);
    b(1,:)=arrcord(2,:);
    arrcord(1,:)=arrcord(3,:);
    arrcord(2,:)=c(1,:);
    arrcord(3,:) = b(1,:);
elseif (arrcord(3,:) == coordinate(i,:))
    c(1,:)=arrcord(2,:);
    b(1,:)=arrcord(1,:);
    arrcord(2,:)=arrcord(3,:);
    arrcord(1,:)=c(1,:);
    arrcord(3,:)=b(1,:);
end
dq =1.0 - q;
xv1 = arrcord(1,1);
xv2 = arrcord(2,1);
xv3 = arrcord(3,1);
yv1 = arrcord(1,2);
yv2 = arrcord(2,2);
yv3 = arrcord(3,2);
x1 = lmd(elma,1);
x2 = (xv1+xv2)/2;
x3 = xv2;
x4 = (xv2+xv3)/2;
y1 = lmd(elma,2);
y2 = (yv1+yv2)/2;
```

```
y3 = yv2;
y4 = (yv2+yv3)/2;
%!Finding the coordinates of the
point of continuity using division
of segment internally form
%        !For   right   edge   flux
continuity
xa = dq * x3 + q * x2;
ya = dq * y3 + q * y2;
%   !For left edge flux continuity
xb = dq * x3 + q * x4;
yb = dq * y3 + q * y4;
%Inserting code for effective area
calculation
aref   =   abs((x1*(yb-ya)+xb*(ya-
y1)+xa*(y1-yb))/2);
%  !Finding  dy/dxi,dy/deta,dx/dxi
and dx/deta
%   !u = xi, v = eta
dx_v = xb - x1;
dy_v = yb - y1;
dx_u = xa - x1;
dy_u = ya - y1;
%Finding the Jacobian
% J = dx/dxi * dy/deta - dx/deta *
dy/dxi
Jaco1 = dx_u * dy_v - dx_v * dy_u;
%Warning   generation   for   low
Jacobian
if( abs(Jaco1) < 1.0E-14)
    display('Warning,  The  Jacobi
is less than 1.0E-14')
    display(Jaco1)
end
%Fluxes in the cell are
% at continuity point A, FA1 = -
T11*PhiA1 -T12*PhiC1 for element 1
[left of A]
% at continuity point C, FC1 = -
T12*PhiA1 -T22*PhiC1 for element 1
[right of C]
T11                              =
(0.5/Jaco1)*(Kel(1,1)*dy_v*(yv2-
yv1)+Kel(1,4)*dx_v*(xv2-xv1)-
Kel(1,2)*dx_v*(yv2-yv1)-
Kel(1,3)*dy_v*(xv2-xv1));
T12a                             =
(0.5/Jaco1)*(Kel(1,2)*dx_u*(yv2-
yv1)+Kel(1,3)*dy_u*(xv2-xv1)-
Kel(1,1)*dy_u*(yv2-yv1)-
Kel(1,4)*dx_u*(xv2-xv1));
T12b              =              (-
0.5/Jaco1)*(Kel(1,1)*dy_v*(yv2-
yv3)+Kel(1,4)*dx_v*(xv2-xv3)-
Kel(1,2)*dx_v*(yv2-yv3)-
Kel(1,3)*dy_v*(xv2-xv3));
T22                   =          (-
0.5/Jaco1)*(Kel(1,2)*dx_u*(yv2-
yv3)+Kel(1,3)*dy_u*(xv2-xv3)-
Kel(1,1)*dy_u*(yv2-yv3)-
Kel(1,4)*dx_u*(xv2-xv3));
```