FINITE DIFFERENCE METHOD ON RANDOMLY GENERATED NON-UNIFORM MESHES FOR POISSON EQUATION

SANAULLAH MASTOI

FACULTY OF SCIENCE
UNIVERSITY OF MALAYA
KUALA LUMPUR

2021

FINITE DIFFERENCE METHOD ON RANDOMLY GENERATED NON-UNIFORM MESHES FOR POISSON EQUATION

SANAULLAH MASTOI

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

**INSTITUTE OF MATHEMATICAL SCIENCE, FACULTY OF SCIENCE
UNIVERSITY OF MALAYA
KUALA LUMPUR**

2021

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **SANAULLAH MASTOI**

Registration/Matric No: 17036590/1

Name of Degree: **DOCTOR OF PHILOSOPHY**

Title of Thesis ("this Work"): **FINITE DIFFERENCE METHOD ON RANDOMLY GENERATED NON-UNIFORM MESHES FOR POISSON EQUATION**

Field of Study: **APPLIED MATHEMATICS**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date: 19/10/2021

Subscribed and solemnly declared before,

Witness's Signature Date: 19/10/2021

Name:

Designation:

ABSTRACT

In this research, a novel method based on randomly generated grids is proposed. This method enables fast convergence and improves the accuracy of the solution for a given problem. It also enhances the quality of precision by minimizing the error. The finite-difference method involving uniform grids is commonly used to solve the partial differential equation. However, it requires a higher number of iterations to reach convergence.

In addition, there is still no definite principle for the discretization of the model to generate the mesh. The newly proposed method employed randomly generated grids for mesh generation. This method is compared with the uniform grids method to check the validity and potential in minimizing the computational time and error. The comparative study is conducted for the first time by generating meshes of different sizes and boundary values.

The numerical solutions of partial differential equations and the generalized classification of fractional differential equations are obtained through various approaches, such as exact solutions, analytically, fractional differentiations, and the more generalized form of finite difference method over uniform novel method randomly generated grids. The proposed method is also known as sanaullah mastoi's method or SM's method.

The new approach is the numerical solution through the finite difference method using randomly generated grids. This study proves that the finite difference method over randomly generated grids found faster convergence iteratively, reduced computational time than uniform grids, and minimize error. A significant reduction in computational time is also noticed. Thus, this method is recommended to be used in solving the partial differential equation. However, SM's Method's performance may be increased by reshaping the mesh parameters, and broad scope of research is available.

Keywords: Fractional Differential equation, Partial differential equation, Finite difference method, Randomly generated grids, Uniform meshes, Non-uniform meshes, S.M's Method, Biological system.

ABSTRAK

Dalam penyelidikan ini, kaedah baru dicadangkan berdasarkan grid yang dihasilkan secara rawak. Kaedah ini memungkinkan penumpuan yang cepat dan meningkatkan ketepatan penyelesaian untuk masalah tertentu. Ia juga meningkatkan kualiti ketepatan dengan mengurangkan ralat. Kaedah perbezaan terhingga yang melibatkan grid seragam biasanya digunakan untuk menyelesaikan persamaan pembezaan separa. Walau bagaimanapun, ia memerlukan bilangan ulangan yang lebih tinggi untuk mencapai penumpuan.

Di samping itu, masih belum ada prinsip yang pasti untuk diskritisasi model untuk menghasilkan jaringan. Kaedah yang baru dicadangkan menggunakan grid yang dihasilkan secara rawak untuk penjanaan jaringan. Kaedah ini dibandingkan dengan kaedah grid seragam untuk memeriksa kesahan dan potensi dalam meminimumkan masa dan ralat pengiraan. Kajian perbandingan dilakukan untuk pertama kalinya dengan menghasilkan jaringan dengan pelbagai ukuran dan nilai sempadan.

Penyelesaian berangka dari persamaan pembezaan separa dan klasifikasi umum persamaan pembezaan, yang merupakan persamaan pembezaan pecahan, diperoleh melalui pelbagai pendekatan seperti penyelesaian tepat, analitik, pembezaan pecahan, dan bentuk kaedah perbezaan terhingga yang lebih umum daripada grid seragam dan kaedah baru grid yang dihasilkan secara rawak. Kaedah yang dicadangkan juga dikenali sebagai kaedah sanaullah mastoi atau kaedah SM.

Pendekatan baru ini adalah penyelesaian berangka melalui kaedah perbezaan terhingga menggunakan grid yang dihasilkan secara rawak. Kajian ini membuktikan bahawa kaedah perbezaan terhingga grid yang dihasilkan secara rawak mendapati penumpuan lebih cepat secara berulang, mengurangkan masa pengiraan daripada grid seragam, dan mengurangkan ralat. Pengurangan masa pengiraan yang ketara juga diperhatikan. Oleh itu, kaedah ini disyorkan untuk digunakan dalam menyelesaikan persamaan pembezaan separa. Namun, prestasi Kaedah SM dapat ditingkatkan dengan mengolah parameter jaringan, dan ruang skop penyelidikan yang luas tersedia.

Kata Kunci: Persamaan Pembezaan Pecahan , Persamaan Pembezaan separa, Kaedah pembezaan terhingga, Grid yang dihasilkan secara rawak, Jaringan seragam, Jaringan tidak seragam , Kaedah S.M, Sistem biologi.

ACKNOWLEDGEMENTS

In the Name of Allah, the Most Merciful, the Most Compassionate (بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ), all praises be to Allah. But, first and foremost, I acknowledge my limitless thanks to Allah, the Even Thankful, for his help and Blessings and prayers and peace be upon Prophet Mohammad (صَلَّى ٱللهُ عَلَيْهِ وَسَلَّمَ), The Messenger of Allah. At the outset, I bow before Almighty God whose benign benediction gave me the required zeal to complete the work in time.

Please recite durood (اَللّٰهُمَّ صَــــلِّ عَلٰيٰ سَيِّدِنَا مُحَمَّـــدِنِ النّبِيِ الاُمّيِ وَعَلٰيٰ اٰلِه وَاَصْحَابِه وَبَارِكْ وَسَلِّمْ). I want thanks to Saien Muhammad Ali Qalandari Baloch, Baba Saien Ahmed Ali Qureshi and Syed Ghulam Hyder Shah, the spiritual guide. They guide me right and real path of Islam.

Most importantly, I would like to express my sincere gratitude to my supervisor Prof. Dr. Wan Ainun Mior Othman, for her guidance, support, invaluable help, encouragement, and supervision throughout my research. Their understanding, patience, time, and valuable advice have been the key to the success of this study. Thanks to my ISM colleague, staff, and friends at the University of Malaya for the support, guidance, and help at several stages.

This academic pursuit would not have been achievable without the support of my Father, Respectable Rehmatullah Mastoi, Mother Jannat Khatoon, and sibling Naimatullah Mastoi, Dr. Ruqia bano, Kashif, Engr. Zabiullah, Engr. Naeemullah and Engr. Saifullah Mastoi.

To my better half, Sanya Mastoi, Daughter Maah Summayya Tabbassum and the newly joining members in our home, one of them our newborn baby on 23rd of September 2021 and BigB’s bride on October 10th, 2021. I wish you the best in this world and hereafter. Thank you all for supporting and understanding. May Allah reward you for all this.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

DE	:	Differential Equation
PDE	:	Partial differential equation
ODE	:	Ordinary Differential Equation
FDE	:	Fractional Differential Equations
FDM	:	Finite Difference Method
RGM	:	Randomly Generated meshes
RGG	:	Randomly Generated grids
SM's Method	:	Sanaullah Mastoi's Method
SD	:	Standard deviation
SSE	:	Sum of Square error
RMSE	:	Root Mean Square Error

LIST OF NOTATIONS

R-square	:	Coefficient of correlation
$u(x)$	:	function u, at x.
$u(x, y)$	:	function u, at x and y.
$u(x, y, z)$	:	function at (x, y and z).
u	:	Dependent variable
p_1	:	Regression’s Coefficient
p_2	:	Regression’s Coefficient
p_3	:	Regression’s Coefficient
p_4	:	Regression’s Coefficient
p_5	:	Regression’s Coefficient
p_6	:	Regression’s Coefficient
p_7	:	Regression’s Coefficient
p_8	:	Regression’s Coefficient
p_9	:	Coefficient of Regression
a_1, a_2	:	Constants, Boundary values
b_1, b_2	:	Constants, Boundary values
a, b, c, d	:	Constants, Boundary values
e , g , h , k	:	Grids, Meshes, Step sizes

CHAPTER 1:

INTRODUCTION

1.1 General

The solutions of the differential equations are wide-ranging in the study. The study focuses on the components in the areas of scientific applications, mathematics, and computers. These three facets are connected, or we can say that strongly tied. Numerical solutions of partial differential equations involve the applied aspect of a problem without considering mathematical and computing factors. The mathematical aspects of numerical partial differential equations can often be developed without considering applications or computing, but experience shows that this approach does not yield valuable results. The numerous techniques help to solve partial differential equations analytically and numerically. The finite differences, finite elements, spectral methods, collocation methods, domain decomposition method, adomain decomposition method, a new method (SM’s Method), etc., are various techniques. We will provide a review of an extensive range of methods. A certain amount of theory and rigor will be included, but the implementation of the methods will be stressed at all times. The goal is to come out of this course with many methods about which you have theoretical knowledge and with which you have numerical experience. Often, when numerical techniques are going to be used to solve a physical problem, it is not possible to thoroughly analyze the methods used. Whenever we use methods that have not been thoroughly investigated, we must resort to methods that become a part of numerical experimentation. As we shall see, often such investigation will also become necessary for linear problems. We usually do not even know what to prove analytically until we have run a well-designed series of experiments. The methods that we develop, analyze, and implement will usually be illustrated using standard model equations. These will often be the heat equation, the one-way wave equation, or the Poisson equation. We will generally separate the methods to correspond to the different equation types by first doing methods for parabolic equations, then methods for hyperbolic equations, and finally methods for elliptic equations. However, we reserve the right to introduce an equation of a different type and a method for equations of various types at any time.

1.2 Calculus

Calculus is the branch of mathematics in which we compute the derivatives and integrals. Most historians and mathematicians are agreed that the idea of calculus was developed autonomously by the great mathematicians Isaac Newton (1643–1727) and the German Gottfried Leibniz (1646–1716), whose images appear in Figure 1.1. The fact is that the idea of calculus (the relationship between differentials and the integrals) was first understood by Leibniz and Newton. Both mathematicians benefited from predecessors' work, such as Barrow 1669, Fermat 1629, and Cavalieri 1635. The amicable relationship between two mathematicians appears, and later, a bitter controversy exploded over whose work took precedence. Although it seems likely that Newton did, indeed, arrive at the ideas behind calculus first, we are indebted to Leibniz for the notation that we commonly use today.

Figure 1.1: The Great Mathematician Isaac Newton and Gottfried Wilhelm Leibniz are the founders of the calculus

1.3 Differential equations

The differential Equation (DE) is widely used to simulate the various physical processes (Farlow, 2006; Zwillinger, 1998). To obtain the numerical solution of the two-dimensional equation, the proper choice of mesh (alternatively called the grid) is the foremost step to achieve better accuracy and convergence. However, the mesh generation process is not unique since there is no universal rule (formula) to be discovered. Usually, the meshes are designed according to the problem and physical structures. It is often found that the better mesh quality led to a greater rate of convergence (Ang). Creating the most feasible mesh for a particular problem is challenging due to the boundary conditions and domain structure variability. The two-dimension domain is filled with structured or unstructured meshes with quadrilateral or triangular elements, respectively. The structured meshes are naturally easier to compute and implement and may require more elements or worse-shaped elements. At the same time, the unstructured meshes are often computed by Delaunay triangulation of point sets (Ang). There are pretty varied approaches for structured and unstructured meshes having their own merits and limitation.

1.4 Ordinary differential equations

In mathematics, an ordinary differential equation (ODE) is a differential equation containing one or more functions of one independent variable and the derivatives of those functions (Dinesh.) The term ordinary is used in contrast with the term partial differential equation, which may be concerning more than one independent variable.

Ordinary differential equations (ODEs) arise in many contexts of mathematics and social and natural sciences. Mathematical descriptions of change use differentials and derivatives. Various differentials, derivatives, and functions become related via equations, such that a differential equation is a result that describes dynamically changing phenomena, evolution, and variation. Often, quantities are defined as the rate of change of other quantities (for example, derivatives of displacement to time) or gradients of quantities, which is how they enter differential equations.

Specific mathematical fields include geometry and analytical mechanics. Scientific areas include much of physics and astronomy (celestial mechanics), meteorology (weather modeling), chemistry (reaction rates),(Shchepetkin & McWilliams, 2005) biology (infectious diseases, genetic variation), ecology and population modeling (population competition), economics (stock trends, interest rates, and the market equilibrium price changes).

1.5 Partial differential equation

A partial differential equation (or briefly a PDE) is a mathematical equation that involves two or more independent variables, an unknown function (dependent on those variables), and partial derivatives of the unknown function to the independent variables(Samaniego et al., 2020; Strauss, 2007; Thomas, 2013). The order of a partial differential equation is the order of the highest derivative involved. A solution (or a particular solution) to a partial differential equation is a function that solves the equation or, in other words, turns it into an identity when substituted into the equation. A solution is called general if it contains all particular solutions of the equation concerned(Chen, Fan, & Wen, 2012). The term exact solution is often used for second-and higher-order nonlinear PDEs to denote a particular solution. Partial differential equations are used to mathematically formulate and thus aid the solution of physical and other problems involving functions of several variables. (Agarwal, Agarwal, & Ruzhansky, 2020; Ahmad, Akgül, Khan, Stanimirović, & Chu, 2020; Duan & Tang, 2020; El-Ajou, Al-Smadi, Oqielat, Momani, & Hadid, 2020; Ghosh, 2020; Habeeb et al., 2020; Harir, Melliani, El Harfi, & Chadli, 2020; Hosseini, Kalhori, & Al-Jumaily, 2020; Kucharski et al., 2020; Miller, 2020; Samaniego et al., 2020; Smitha & Nagaraja, 2020; S. Sun, Gou, & Geng, 2020; Vargas-González et al., 2020; Wang & Yamamoto, 2020), such as the propagation of heat or sound, fluid flow, elasticity, electrostatics, electrodynamics, etc.

1.6 Fractional differential equations

Fractional calculus is a branch of mathematical analysis that studies the different possibilities of defining real number powers or complex number powers of the differentiation operator D.

$$Df(x) = \frac{d}{dx} f(x)$$

and of the integration operator J

$$Jf(x) = \int_0^x f(s)\, ds$$

and developing a calculus for such operators generalizing the classical one.

In this context, the term powers refers to iterative application of a linear operator D to a function f, that is, repeatedly composing D with itself, as in $D^n f(x) = (D \odot \mathrm{D} \odot \mathrm{D} \odot, \dots, \odot \mathrm{D})f = (D(D(D(\dots D(f) \dots))))$. For example, one may ask for a meaningful interpretation of $\sqrt{D} = D^{\frac{1}{2}}$, as an analogue of the functional square root for the differentiation operator, that is, an expression for some linear operator that when applied twice to any function will have the same effect as differentiation. More generally, one can look at the question of defining a linear operator D^{α}, for every real number a in such a way that, when a takes an integer value n ∈ ℤ, it coincides with the usual n-fold differentiation D if n > 0, and with the (−n)-th power of J when n < 0.

One of the motivations behind the introduction and study of these sorts of extensions of the differentiation operator D is that the sets of operator powers { Da | a ∈ ℝ } defined in this way are continuous semigroups with parameter a, of which the original discrete semigroup of { Dn | n ∈ ℤ } for integer n is a denumerable subgroup: since continuous semigroups have a well developed mathematical theory, they can be applied to other branches of mathematics.

Several researchers focused and concentrated on studying the exact and numerical solutions of the differential equations. We considered and modified the available method introduced in (Farlow; Strauss) to solve the fractional partial differential equations in the present work.

Fractional differential equations, also known as extraordinary differential equations, generalized differential equations through the application of fractional calculus. The i^{th} derivative of a function f(x) at a point x is a local property only when a is an integer; this is not the case for non-integer power derivatives. In other words, a non-integer fractional derivative of a function f (x) at x = a depends on all values of f, even those far away from a. Therefore, it is expected that the fractional derivative operation involves boundary conditions involving further information on the function. The fractional derivative of a function to order a is often defined by the Fourier or Mellin integral transforms.

1.7 Numerical Methods

A numerical method is consistent if all the derivatives' approximations (finite difference, finite element, finite volume, etc.) tend to the exact value as the step size (Δt, Δx etc.) tends to zero. Moreover, a numerical method is stable (like IVPs) if the error does not grow with time (or iteration).

1.7.1 Iterative method

In computational mathematics, an iterative method is a mathematical procedure that uses an initial value to generate a sequence of improving approximate solutions for a class of problems. The n-th approximation is derived from the previous ones. A specific implementation of an iterative method, including the termination criteria, is an algorithm of the iterative method. An iterative approach is called convergent if the corresponding sequence converges for given initial approximations. Mathematically rigorous convergence analysis of an iterative approach is usually performed; however, heuristic-based iterative methods are also common.

In contrast, direct methods attempt to solve the problem by a finite sequence of operations. In the absence of rounding errors, direct methods would deliver an exact solution (like solving a linear system of equations $Ax = b$ by Gaussian elimination). Iterative methods are often the only choice for nonlinear equations. However, iterative methods are often helpful even for linear problems involving many variables (sometimes of the order of millions), where direct methods would be prohibitively expensive (and in some cases impossible) even with the best available computing power.

The numerical method is iterative. This method is said to be consistent if approximations such as FDM, FEA. FVM. tends to the size of the exact values. (Chen et al., 2012; Schiesser, 2012).

1.8 Finite difference Method

The discussion begins with the numerical methods for the PDEs by getting solved a problem numerically.

$$S_l = AS_{mm}, m \in (a, b), l > a \qquad (1.8.1)$$

$$S(m, b) = g(m), m \in [a, b] \qquad (1.8.2)$$

$$S(a, l) = p(l), S(b, l) = q(l), l \geq a \qquad (1.8.3)$$

Where $g(a) = p(a), and\ g(b) = q(a)$. We can solve this problem numerically for the illustrations of the finite difference method (FDM). In numerical analysis, finite-difference methods (FDM) are numerical techniques for solving differential equations by approximating derivatives with finite differences. The spatial domain and time interval (if applicable) are discretized or broken into a finite number of steps, and the solution's value at these discrete points is approximated by solving algebraic equations containing finite differences and values from nearby points.

Finite difference methods convert ordinary differential equations (ODE) or partial differential equations (PDE), which may be nonlinear, into a system of linear equations that matrix algebra techniques can solve. Modern computers can perform these linear algebra computations efficiently, which, along with their relative ease of implementation, has led to the widespread use of FDM in current numerical analysis(Grossmann, Roos, &

Stynes, 2007). Today, FDM is one of the most common approaches to the numerical solution of PDE, along with finite element methods.(Dinesh.)

1.9 SM's Method

In FDM, it relies on the discretization of function on grids or meshes. There is no principle for mesh generations. Usually, FDM uses uniform or regular grids, but SM's Method focuses on randomly generated grids. There is no special treatment in solving SM's method than FDM. The only mesh generation process is different than regular meshes. In this method, we use mathematical software like MATLAB, ANSYS etc.

The name SM's method is abbreviated from Sanaullah Mastoi's Method. The research work is already published (Kumaresan., 2020; Mastoi, Kalhoro, et al., 2021; Mastoi, Mugheri, Kalhoro, & Buller, 2020; Mastoi, Mugheri, et al., 2021; Mastoi, Othman, Ali, Rajput, & Fizza, 2021; Mastoi, Othman, & Nallasamy., 2020a, 2020b; Othman, 2020) research journals.

1.10 Research Questions and Research Hypothesis

This study's prominent role and enthusiasm is the mesh generation process. However, there is no specific rule or principle for mesh generation. This thesis is based on randomly generated grids. Numerical solution of a basic differential equation with the help of the finite difference method using randomly generated grids. Therefore, it is hypothesized that "the randomly generated meshes may improve the convergence of the numerical solution." If the numerical convergence improves, then the research question occurs: "what accuracy can be obtained using randomly generated grids."

1.11 Problem statement

The execution of the FDM depends on the discretization schemes, dividing the subdomain and the nodal parameters. Usually, it is found that FDM uses regular, uniform, or equal step size or meshes. Most scholars recommended that the idea of using a randomly generated grid helpful in rapid solutions. The investigations are keen on regarding the randomly generated grids. Therefore, an inquiry into the effect of randomly generated meshes, non-uniform grids, variable step size, and FDM performance is the primary motivation of the proposed research. The idea may be implemented on the fractional partial differential equations, and a comparative study has been done. The One, Two, and Three-dimensional partial differential equations used with boundary conditions for testing and implementation purposes. Detailed research has been done on the Two-dimensional basic PDEs.

The proposed research will focus on numerical convergence by comparing the results obtained on uniform meshes with random samples of meshes.

1.12 Aim and objectives

The specific objectives of this research are as follows:

1) Introducing finite-difference random grids and notations of functions defined on grids.
2) Introducing the SM's Method for the Partial differential equations.

   2(a). By generating the uniform meshes for the finite-difference solution of differential equations.

   2(b). Generating the random meshes for finite difference solutions of the partial differential equations.

   2(c). along with an introduction of treatment of some methods and functions like fractional differential equations, domain decompositions method, and heat equation using SM's Method.
3) Implementation of SM's Method, through analyzing the effect of samples of random meshes on the convergence of the numerical solution of the differential equation.

4) To test the practicability and feasibility of using the randomly generated finite difference meshes by statistical analysis of the random samples of meshes used to obtain the numerical solution.
5) To validate the results by point-wise comparison of the numerical solution and the computational time reductions of uniform and random meshes.

1.13 Scope and limitations of the research

The proposed study is applied on one, two, and three-dimensional differential equations where the finite difference solution is achieved over samples of random meshes. The differential equation is considered stable dirichlet boundary conditions and limited to regular domains like rectangles and squares. The idea is also recommended for fractional partial differential equations.

1.14 Expected outcomes of the research

The proposed research is expected to explore the new techniques to design the best finite difference meshes for the improved numerical accuracy and convergence of the simulation of differential equations. In addition, the proposed research results may be helpful to solve the various real-world problems governed by the differential equation in general and random geometric features.

1.15 Thesis layout

This thesis consists of eight chapters and is organized so that each chapter starts with introductory remarks and ends with a summary. Chapter 1 provides the introduction of the project by highlighting the primary motivation and objective of the research. In chapter 2, the brief literature review is presented by citing the more related works. Chapter 3 presents the complete methodology to achieve the objectives of the proposed research. The extensive results are obtained, discussed, and justified. In Chapter 4, Local solutions on uniform grids, Chapter 5 Local solutions on randomly generated grids, and Chapter 6

Statistical tests of numerical solutions (Uniform Vs. Random) and Chapter 7 Point-wise comparisons (Uniform Vs. Random) and examples and Finally, Chapter 8 gives the overall conclusion and offers some directions for the future work.

CHAPTER 2:

LITERATURE REVIEW

2.1 Introduction

The proposed research's study motivations and study objectives have been discussed in detail in the previous chapters. This chapter reviews research works on numerical methods, techniques, and mesh generation. To the best of the author's knowledge, it is declared that, previously, there isn’t work has been done on the randomly generated meshes or randomly generated grids. Numerical solution of partial differential equations using SM’s method or finite difference method using randomly generated grids was not studied before or found in the literature. So, This is why it wasn't easy to find material or research work or enough stuff on this topic, but some general ideas like the mesh generation process and their applications are briefly reviewed.

2.2 History of Finite difference method

In 1592, the concept of Finite difference originated by Jost Burgis(van den Ende et al., 2015). Various scientists, including Issac Newton(Craig, 1901; Fraser, 1927; Moffatt, 1997), used the finite difference in their studies. In 1715, Brook Taylor extended the idea of Finite Differences presented in studies. The Method was simplest, helpful, and exciting by increasing interest by various scholars. George Boole in 1860, the L. M. Milne-Thomson in 1933, and Károly Jordan in 1939 were done in the finite difference method (Navarro Crespo, 2020). The Method (FDM) is a mathematical difference expression of the document $f\,(s + m) - f\,(s + n)$, which divided into the subdomains or step size tends to zero. For example, the finite difference is divided into $m - n$, becomes a difference quotient. The approximation of derivatives by finite differences play a part in finite difference methods for the numerical solution of differential equations, especially boundary value problems.

In FDM, the most important is the cell, mesh, grids, or step size. There is no specific method or formula for the mesh generation process. However, meshes are generated as per problem or conditions. Thus, we can say as per demand. Generally, uniform or equal step size is followed in mathematical, engineering, and scientific problems. This concept gives an enormous motivation to introduce SM's method. Furthermore, the particular recurrence relations can be written as difference equations by replacing iteration notation with finite differences.

2.3 Solutions of differential equations

In mathematics, A differential equation involves an unknown function $y = f(x)$ and one or more of its derivatives. A solution to a differential equation is a function $y = f(x)$ that satisfies the differential equation when function f and its derivatives are substituted into the equation. A partial differential equation (PDE) is an equation that imposes relations between the various partial derivatives of a multivariable function. An "unknown" solved for, similarly to how variable is considered an unknown number, to be solved for, in an algebraic equation. However, it is usually impossible to write down explicit formulas for solutions of partial differential equations. There is, correspondingly, a vast amount of modern mathematical and scientific research on methods to numerically approximate solutions of certain partial differential equations using computers. Partial differential equations also occupy a large sector of pure mathematical research. The usual questions are on identifying general qualitative features of solutions of the various partial differential equation.

2.4 Analytical method

A differential equation may be analytically solved with different available methods. For example, The methods are separation of variables, homogenous, non-homogenous or in-homogenous, non-liner differential equations, method of characterizations, integral transform, change of variables, fundamental solution, superposition principle, methods for non-linear equations, lie group method, semi-analytical methods, and others. First, the analytical method defines a differential equation involving an unknown function and one or more of its derivatives. Then the Analytical solution or solution to the differential equation is a function that satisfies the differential equation when the function and its derivatives are substituted into the equation.

2.5 Fractional differential equation

Fractional order differential and integral operator generalize traditional integer-order integration and derivation gained more attention from the last two decades because of physical interpretation in different fields such as biology, economics, biochemistry, medicine, and engineering science. The idea of fractional calculus is old as classical calculus; Leibniz and L'Hospital discussed it for the first time in 1965. Nonlinear fractional evolution equations describe complex phenomena in different areas such as biology, acoustics, physics, finance, and fractional dynamics(Bin, 2012). Many researchers have solved various numerical and analytical numerical and analytical methods of linear, nonlinear, integers, and fractional order problems. Numerical methods such as finite fourth-order difference methods are used by Ali et al. (Umair Ali, Sohail, Usman, et al., 2020). They solved the non-integer order sub-diffusion model and found the theoretical analysis of stability and convergence. Another literature (U Ali, Kamal, & Mohyud-Din) used the Crank-Nicolson scheme's 2D diffusion equation of fractional order. Jiang and Jingtan (Guan et al.) developed the high order finite element approach for the fractional-order differential equation and finding the convergence order rate. Srivastava et al. (Srivastava, Kung, & Wang) discussed the local meshless method for 3D convection-diffusion equation. They approximated the space derivatives based on the meshless procedure and fractional-order time derivatives are by Caputo derivative. The 2D time-fractional order differential equation is solved in ("Chapter 5 Preliminary

Review of Finite Difference Methods," ; Zwillinger). They used numerical approximation and discussed the stability and convergence analysis for the diffusion model of fractional order. Ahmad et el. (Abouelregal, Moustapha, Nofal, Rashid, & Ahmad) studied a new and simple numerical approach for the fifth-order KdV equation. Also, it compared the obtained values with Adomian's decomposition method and briefly explained the theoretical analysis to assess the accuracy. Ali and Abdullah (Umair Ali, Mastoi, Othman, Khater, & Sohail) explicitly developed the Saul'yev technique for the 2D diffusion model with stability analysis and provided test examples to demonstrate accuracy. Ahmad et al. (Abouelregal et al.) suggested an efficient local differential quadrature technique for the 2D hyperbolic telegraph equation. The time and space derivatives are approximated based on the time integration technique and multiquadric radial basis. Balasim and Mohd Ali (Mahmood, Md Basir, Ali, Mohd Kasihmuddin, & Mansor) worked on a 2D fractional-order Cable equation. They found the solution by two numerical methods, fully implicit and Crank-Nicolson method. Akgül (Ahmad et al.) developed a novel approach to reproducing kernel Hilbert space function and used Atanagana-Baleanu fractional derivatives. They solved fractional initial values problems to demonstrate the numerical results. Akgül et al. (Ahmad et al.; Saeed, Riaz, Baleanu, Akgül, & Husnine) considered the fractional-order integrator circuit model and established a unique solution. They find out the stability analysis and numerical results of the proposed model by Atanaga-Toufik scheme. The SM's method as finite random difference method and Caputo definition are discretized introduced by (Bar-Sinai, Hoyer, Hickey, & Brenner; Farlow; Folland; Strauss), having applications in control theory. Akgül (Ahmad et al.) studied the solution of the fractional-order differential model and used the Laplace transform to get the solution. The effective analytical methods to construct the solitary wave for differential equations such as Shang and Zheng (Y. Sun, Sun, & Zheng) constructed all possible exact solutions by the method. Yokus et al. (Yokus, Durur, Ahmad, Thounthong, & Zhang) solved the Bogoyavlenskii equation and used (G′/G, 1/G)-expansion and (1/G′)-expansion to find the exact traveling wave solution. Barman et al. (Barman, Seadawy, Akbar, & Baleanu) studied the interesting nonlinear equations Riemann wave and Novikov-Veselov that describe the ocean's tidal and tsunami types of waves. The author’s implemented the generalized Kudryashov technique for the exact solution of the proposed equations and obtained various solitons. Jawad et al. (Jawad, Petković, & Biswas) discussed the nonlinear evolution equations, which describe nerve propagation in biology and genetics. They applied the simple equation approach for the suggested equations and discussed the physical phenomena. In (Islam, Akbar, & Azad) presented the (G′/G)-expansion to find the solutions for the evolution equations and used Jumaire’s definition for fractional-order derivative. Bashan et al. (Bashan, Yagmurlu, Ucar, & Esen) combined the finite-uniform and random difference procedure (FUDP-FRDP) with the quadratic differential scheme (QDS) to discuss the solution of the modified two-dimensional partial differential equation as a wave-type physical phenomenon. They obtained and discussed the solitary wave nature solution. They recorded and listed the error norms, and the solution is displayed against several emerging parameters in graphs. Modified spline technique (MST) has been adopted by Bashan et al. (Bashan et al.) to compute the soliton solution of the nonlinear Schrodinger equation. They examined the efficiency and effectiveness of proposed procedure for five different problems and found an excellent agreement while computing the error norms. Few important contributions are covered in (Başhan). The three models of shallow water wave equations are determined by Wazwaz (Wazwaz). The Hirota bilinear approach was used for multiple solitons solutions and the coth-tanh for single soliton solution. Hosseini et al. (Hosseini et al.) considered the special type of mathematical model (3+1)-dimensional breaking solitons equation and used the linear superposition method. The method shown high efficiency and strongly handled the nonlinear model. In the said literature partial and fractional order differential equations are solved successfully. Fractional order is sometime a function of dependent or independent variables which are more appropriate to discuss the diffusion processes in porous medium and medium structure (Umair Ali, Sohail, & Abdullah). The reaction kinetics of proteins has been originated to show simple mechanisms that are accurately defined by fractional-order changes with temperature (Mastoi, Othman, et al.). These examples show that the variable-order operator describes some classes of physical models better than fractional order. In the review article Sun et al. (Y. Sun et al.) provided basics definitions, models, numerical techniques, and applications. So far, in the previous literature, many researchers have solved the variable-order fractional evolution equations by various numerical methods, such as Shekari (Shekari, Tayebi, & Heydari) solved the 2D time-fractional variable-order wave equation base on numerical moving least squares approach for a different domain. The resulted solution confirmed the efficiency and easy implementation of variable order models. Chen et al. (Chen et al.) focused on Stokes's first problem variable order and found the solution numerically. Also, we discussed the theoretical analysis via the Fourier series. The theoretical analysis supported the obtained numerical solution. The advection-diffusion equation of variable-order is solved explicitly and implicitly with the nonlinear forcing term by Zhuang (Samaniego et al.). Chen et al. (Gu, Wang, Chen, Zhang, & He) considered the anomalous diffusion of variable order equation with the numerical algorithm. The theoretical analysis of stability, convergence, and solvability via Fourier was discussed. The numerical solutions were effective, and the proposed scheme is powerful for such types of variable order models. The studies reported in (U Ali et al.; Umair Ali et al.; Umair Ali, Sohail, & Abdullah; Umair Ali, Sohail, Usman, et al.) discussed the chaotic analysis by using fractional operators.

This study aims to extend the closed-form traveling waves solution to the nonlinear variable-order fractional evolution equations. Here, we solve nonlinear space-time variable-order fractional MEWE based on variable-order Caputo derivative by exp method. The variable-order problems are more complicated than a constant fractional-order problem, and the evolution of a system can be more accurately described. This contribution seems natural and modeled many systems with variable-order (Lee & Kim; Sohail et al.). The closed-form solutions for variable-order evolution equations and finite random grids are unavailable to the author's knowledge, and we hope it contributes to the literature. Few significant contributions relating to the concepts of variable-order fractional operators and other related studies are reported in (Al-Shawba, Gepreel, Abdullah, & Azmi).

This study aims to extend the work in the numerical solution of finite random grids, SM's method, and fractional differential equation in partial differential equations. Moreover, the concept helps solve the problem related to various FPDE's, Laplace equations, Poisson's equations, domain decomposition method, adomain decompositions method, Legendre functions, and others.

2.6 Numerical Computing

The massive power of mathematics is, arguably, best divulged by "crunching" numbers. While an equation or a formula can provide significant insight into a physical phenomenon, its depth can only be welcomed by a limited few that already have a relatively thorough understanding of the phenomenon, to begin with, the same equation or formula. However, when put to use to generate numbers, it reveals significantly more. For example, the Navier–Stokes equations, which govern fluid flow, are not particularly appealing on paper except, perhaps, to a select few. However, when appropriately post-processed, their solution is depicted in line plots, field plots, and animations. In the realization that the numbers generated out of sophisticated equations are far more revealing than the equations themselves, for more than a century, applied mathematicians have endeavored to find ways to generate numbers from equations rapidly. The desire to create numbers has also been partly prompted by closed-form analytical solutions that only exist for a few scenarios. Even those require number crunching or computing to some degree.

Although the history of computing can be traced back to Babylon, an abacus was believed to develop over 2400 bc. It was not until the nineteenth century that the development of devices that could, according to the modern sense of the word, compute, came to be realized. While the industrial revolution created machines that made our everyday life more accessible, the nineteenth and twentieth centuries witnessed strong interest among mathematicians and scientists in building a device that could crunch numbers or compute repeatedly and rapidly. The so-called Analytical Engine, proposed by Charles Babbage around 1835, is the first computer design capable of logic-based computing. Unfortunately, it was never built due to the political and economic turn of events. In 1872, Sir William Thomson made an analog tide-predicting machine that could integrate differential equations. The Russian naval architect and mathematician Alexei Krylov (1863–1945) built a device to integrate an ordinary differential equation in 1904. These early analog machines were based on mechanical principles and made using mechanical parts. As a result, they were slow. The Second World War stimulated renewed interest in computing both on the German and British sides. The Zuse Z3, designed by Conrad Zuse (1910–1995), was built by German engineers in 1941. It is believed to be the world's first programmable electromechanical computer. The British cryptanalyst Alan Turing is also known as the father of computer science and artificial intelligence. The Imitation Game recently brought to the limelight, built an electromechanical machine to decode the Enigma machine used by the German military for their internal communication. Shortly after the war, Turing laid the theoretical foundation for the modern stored-program programmable computer. This machine does not require any rewiring to execute a different set of instructions. This so-called Turing Machine later became the academic standard for computer design, and modern computer designs, upon satisfying a set of mandatory design requirements, are referred to as “Turing complete.” With the invention of the bipolar transistor in 1947 and integrated circuits in 1952, the world witnessed a meteoric rise in computer hardware technology and computing power. In 1965, in a famous statement (Dinesh.), known today as the Moore’s Law, Gordon E. Moore predicted that the number of transistors in an integrated circuit would approximately double every two years. Over the past four decades, the growth in Very Large Scale Integrated (VLSI) circuit technology has roughly followed Moore’s law. The 62-core Xeon Phi processor, released by Intel Corporation in 2012, has 5 billion transistors, compared with 2300 transistors in the Intel 4004 processor released in 1971. The number of millions of instructions per second (MIPS), an important marker of processor speed, scales directly as the number of transistors per processor. The Intel Core i7, one of the most prevalent processors in modern (as of 2014), executes 105 MIPS.

Over the past two decades, this dramatic increase in computing power and reduction in cost has stimulated significant research, development, and educational activities that focus on computing. For example, the US National Academy of Engineering now recognizes finite element analysis and computational fluid dynamics. In mainstream industries, such as the automotive industry, the traditional methodology of “make and break” is rapidly being replaced by a paradigm in which analysis and analysis-based understanding of the working principles of a device, part, or process, is considered imperative. The focus is extended to the finite difference method, the SM’s Method, and available methods. As a result, having a working knowledge of the popular simulation tools is rapidly becoming a necessity rather than a choice for the next-generation workforce. In the academic and research communities, the drastic increase in computing power has also prompted renewed interest in developing algorithms that use the enhanced computing power to execute complex tasks efficiently.

The Finite difference method over randomly generated grids is followed the simulation process on computers through mathematical and statistical software. Thus, profile simulation and massive data can be only possible with the help of the computer—the possibilities for computer simulations to promote scientific discoveries and improve our everyday lives.

2.7 Numerical Methods

Numerical methods for ODE can also be extended to the solution of PDE. For example, methods discussed for treating initial value problems can be adopted for parabolic and hyperbolic equations. Similarly, practices discussed for treating BVPs can be adopted to solve elliptic PDEs, also boundary value problems. However, the extension of the methods to solve PDE is not straightforward.

Methods such as finite difference method (FDM), finite volume method (FVM), finite element method (FEM), boundary element method (BEM), etc., are commonly used for treating PDE numerically. All numerical methods used to solve PDEs should have consistency, stability, and convergence.

A numerical method is consistent if all the derivatives' approximations (finite difference, finite element, finite volume, etc.) tend to the exact value as the step size (Δt, Δx, etc.) tends to zero. A numerical method is stable (like IVPs) if the error does not grow with time (or iteration). The convergence of a numerical method can be ensured if the technique is consistent and stable.

In summary, the numerical methods are iterative and must be used if the problem is multidimensional and the region's geometry is too complex. Thus, they need a high degree of mathematical formulation and programming.

We shall look at different aspects of the numerical treatment of different types of PDE in the forthcoming chapters.

2.7.1 Finite Volume Method

The finite volume method derives its name from the fact that in this method, the governing PDE is satisfied over finite-sized control volumes rather than at points. The first step in this method is to split the computational domain into a set of control volumes known as cells, as shown in Fig. 2.1. In general, these cells may be of arbitrary shape and size. However, traditionally, the cells are convex polygons (in 2D) or polyhedrons (in 3D), i.e., they are bounded by straight edges (in 2D) or planar surfaces (in 3D). As a result, if the bounding surface is curved, it is approximated by straight edges or planar faces, as is evident in Fig. 2.1. These discrete bounding surfaces are known as cell faces or simply faces. The vertices of the cells, on the other hand, are called nodes and are, in fact, the same nodes that were used in the finite difference method. All information is stored at the geometric centroids of the cells, referred to as cell centers.

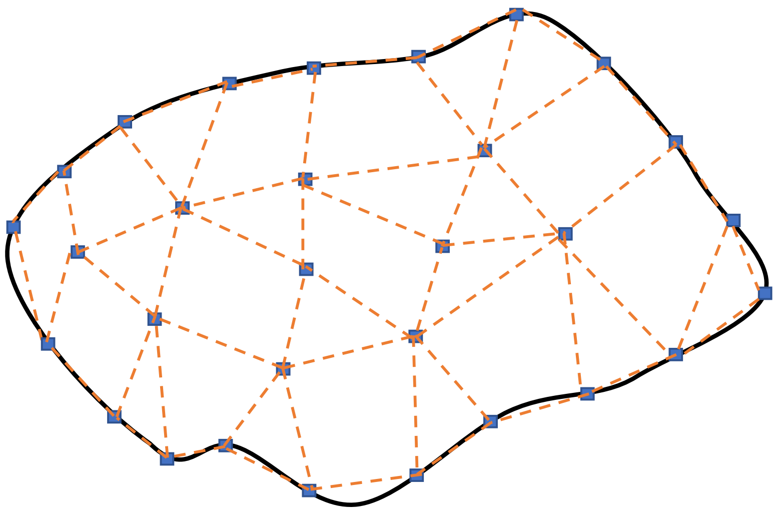


Figure 2.1: Schematic representation of Computational domain and Meshes

Schematic representation of a 2D Computational Domain and Mesh Hollow squares denote the cell centers to be used for the finite volume method. The nodes, indicated by solid squares, are used in the finite difference method and are the vertices of the cells.

In contrast with the finite difference method, the governing PDE is not solved directly in the finite volume method, as evident from the preceding discussion. Instead, it is first integrated over the control volume and then approximated and solved. Furthermore, since no cell center is located at the boundary, the boundary conditions cannot be satisfied directly. Due to these reasons, the solution to the PDE's, obtained using the finite volume method, is known as the weak form solution.

2.7.2 Finite Element Method

The finite element method is a numerical technique for solving partial differential equations. Biological systems, engineering models, solid mechanics, fluid mechanics, electromagnetics, thermodynamics, mathematical models, etc are used (Kreyszig, 2011; Deb, Babuška, & Oden, 2001). However, one of the most powerful techniques for solving PDEs with weak formulations is using a weighted residual method called the Galerkin finite element method (GFEM).

The formulation requires generating a basis function (Ainsworth & Oden, 1997) based on the elemental boundary conditions. This trial function is substituted in the partial differential equation. The first derivative of the trial function is taken for each nodal variable (Burden & Faires, 2001) to construct the residual function. The weighed form of the residual function for the whole domain is integrated by setting it equal to zero. Green's theorem can be applied over the boundary if necessary (Afzal, Sulaeman, & Okhunov, 2016). The corresponding numerical model is set up by discretizing the rectangular domain into smaller elements. Each element consists of nodal coordinates and nodal variables, which are used to perform the Galerkin approximation of the PDEs.

This study generates an element matrix and vector matrix of the boundary by integrating the total number of elements. The set of linear equations represented by the matrices are consecutively solved using the Galerkin approach. For comparison purposes, an exact solution is already available for the non-linear PDE (time-independent and no heat source) used in the 2D heat conduction rectangular domain with both Dirichlet and Neumann boundary conditions. Finally, a stiffness matrix applicable for a homogenous rectangular domain consisting of structured mesh grid elements is presented, the solution scheme of which significantly reduces the CPU performance cost

2.7.3 Finite difference method

When the appropriate finite-difference operators approximate partial derivatives apparent in the differential equations and boundary conditions, the initial-boundary value problem under study is reduced to the solution of a system of algebraic equations at all points in a defined domain. Such a discretization is called a difference scheme, which yields a different solution. A method of differential equations can be approximated by an arbitrary number of different schemes, so that it is necessary to compare their performances and to establish some criteria for checking goodness of approximation("Chapter 5 Preliminary Review of Finite Difference Methods," 1992). It has been used to solve a wide range of problems. These include linear and non-linear, time-independent, and dependent problems.

In 1950, a first mathematical definition of the stability of difference schemes was proposed, based on the requirement that roundoff error should not be amplified unboundedly with increasing step numbers. However, this definition has its disadvantages. On the one hand, it depends not only on the difference scheme adopted but also on the computer used; on the other hand, when the step sizes become smaller, the roundoff error would become larger and more prominent due to the increasing number of operations. Therefore, the definition was changed before long into a new requirement that the error in the difference solution produced by a disturbance to the initial data would not be amplified rapidly. Accordingly, stability may be defined as follows:

For a fixed step size, when the number of time steps grows unboundedly, the upper bound of the error of the difference solution can be estimated based on the disturbance. Here, the disruption is shown in two aspects: the initial value error and the degree of smoothness. It is noticeable that convergence of the difference solution of a Cauchy problem is related to the class of the initial function. Even the initially given process is sufficiently smooth (e.g., it can be expanded into a Fourier series in the Lz-norm sense), the operands treated by computer would be decidedly unsmoothed (they cannot be developed into an infinite series or even a finite sum). Therefore, it is also often required that the convergence of the difference solution can be assured for all common initial functions encountered.

Because of the inconvenience of the above definition, in the mid-l950s, Lax and Richtmyer proposed a second definition, requiring that for the calculation of a different solution at a given instant, the amplification of the solution must be bounded with vanishing step sizes. Considering that the exact solution itself may grow with time, the definition of stability may be alternatively stated as follows:

When step sizes shrink to zero, the rate of growth of the difference solution should not exceed that of the exact solution.

The new definition has two features. (i) the theoretical analysis can be made so long as the difference equation and the boundary condition have been given (making no use of the exact solution), (ii) since the amplification of the difference solution is bounded, the discretization error and roundoff error must also be bounded.

The above two definitions of stability are distinct. In the first definition, step sizes are fixed while the number of steps increases unboundedly, and so it is called step stability. In the second one, initial and final instants are fixed, while the number of steps still increases unboundedly as step sizes vanish; thus, it describes a limit error behavior at some fixed point in space and time and therefore is called point stability. Since the differential equations used in both definitions are identical, they are quite close to each other for the same number of steps.

Unfortunately, they are still unsatisfactory in some respects. For example, oscillations may sometimes appear in the numerical solution, even around an incorrect one. Such a phenomenon is indeed instability (computational instability may be classified into two types, one is strongly unstable if the solution is divergent, and the other is weakly unstable if there are spurious oscillations of finite amplitudes appearing in the solution); however, it would be judged as stable by the two definitions, since the solution and its error remain bounded. Conversely, so long as a numerical solution, although it does not converge mathematically, is accurate enough for practical purposes, it is acceptable.

From the viewpoint of functional analysis, the solution at any instant (a field) may be considered a point in some Banach space. The problem is equivalent to finding a tiny operator which transforms an initial point into a moving point. Norm (or modulus) of the operator may be smaller than, equal to, or greater than one, corresponding respectively to the cases that energy is continually dissipated, conservative, or accumulated so that stability will eventually be lost.

The distinction between convergence and stability should be emphasized. Conceptually, convergence means that the exact solution of a difference scheme approaches the exact solution of a differential equation. In contrast, stability means that the approximate solution of a difference scheme comes to the exact solution. On the other hand, theoretically, convergence requires that a prior estimate holds uniformly in step size, in a form such that a certain norm of the error in a numerical solution is equal to or smaller than a product which is obtained from summing the two norms of truncation errors in the difference equation and numerical boundary condition, and then multiplying the sum by some constant. Stability requires a uniform estimate, in a form such that the norm of the numerical solution is equal to or smaller than a product of some constant and the summed models of the initial and boundary data.

A difference problem is well-posed if it is consistent with the associated differential problem and stable. Naturally, this property is closely related to the well-posedness of the associated differential problem. In general, the condition of the well-posedness of the differential problem is more straightforward. A 1-D difference scheme may be numerically unstable, while a differential problem may be ill-posed only in at least 2-D cases. In addition, in a differential problem, wave dispersion is independent of step size, Ax, but for a different scheme, it depends on the waves with wave-length greater than $2dx$, i.e., with wavenumber satisfying the condition $kdx\ E\ (\ 0\ ,n\ )$

2.8 Mesh generation and its applications

The Mesh generation procedure is not unique. It is a practice, and the practice varies from problem to problem. Mesh is usually called grids or nodes formed per desired and physical or engineering models (S. Sun et al., 2020). It is mostly found that mesh quality depends on the convergence rate; if mesh quality is not good, the convergence rate cannot reach some cases (Duan & Tang, 2020). The challenging design of the most feasible grids for the specific model is variability in boundary conditions and the domain structure. Meshes are divided into structured, unstructured, and hybrid grids. A 2 dimension domain is filled with the structure that is regular parts and easier to implement, compute and unstructured is usually use for complex part in worse shaped and hybrid contain a combination of both grids that is structured and unstructured with quadrilateral and triangular shapes are often computed by Delaunay triangulation of given set of points (Rani & Mishra, 2020). Structured and unstructured meshes having different approaches and advantages and the limitations. The inspiration is a new technique that is to “Generate random grids” and solve a two-dimensional partial differential equation with the help of FDM. So, they hypothesized that “the randomly generated grids(meshes) improve convergence of the numerical solution.” Mesh has different types like curvilinear, cartesian, rectilinear, triangular grids, domain decomposition, and many others. These all types of grids found in the major two categories are structured and unstructured(Buller., 2020; Kumaresan., 2020; Nallasamy., 2020; Othman, 2020). Structured grids are used for regular shapes. The irregular shape we use unstructured grids or meshes with quadrilateral and triangular connectivity is regular or uniform and irregular or non-uniform grids. A hybrid grid is a scheme for joining the unstructured and structured grids (Zhong & Sheng, 2020). In a single strategy, both types were discussed and benefited (Zubar et al., 2020). The meshes or the grids or nodes are designed recalled as per mathematical model structures. Nowadays, the physical problem and model are solved using computer technology to find various algorithms and solutions in less computational time. To highest convergence with less computational time a partial differential equation of Heat equation solved, designed and random grids designed for the ease and various solution of problems. In this way, the numerical algorithm development for the partial differential equation, usually called PDEs, purely depends upon grids. The mesh generation already gained much consideration because of the applicability of physical problems, structural mechanics, CFD computational fluid dynamics, and electromagnetism (Ghosh, 2020; Zhong & Sheng, 2020). The boundaries of shapes change into meshes from finite difference methods, which can construct to the finite element meshes (Ang, 2013; Smitha & Nagaraja, 2020; X. Zhao, Mo, Guo, & Li, 2020). The technique of discretizing PDEs in various techniques is proposed in the literature. However, the numerical solution approach varies with the model as a physical phenomenon to be simulated and the type of original equation and the computational domain of the problem. Previously studies are found on equal step size on a regular mesh (J. Zhao, Yi, & Xu, 2020; Zubar et al., 2020) variable grid size(Lee & Kim, 2020), moving mesh (Duan & Tang, 2020; Uzunca, Karasözen, & Küçükseyhan, 2017; Yan, Rennie, & Mohammadian, 2021) can found on quasilinear applied in the partial differential equation in magnetohydrodynamics (Ghosh, 2020; Liu, 2019; Song & Karniadakis, 2019), porous medium, meteorology, low-speed viscous flow, free surface viscous elastic flow, multi-phase flows and so on (Ang, 2013; Duan & Tang, 2020; El-Ajou et al., 2020; Ghosh, 2020; Han, Nica, & Stinchcombe, 2020; Harir et al., 2020; Hosseini et al., 2020; Kovács, Kirchner, & Bolin, 2020; Kucharski et al., 2020; Lee & Kim, 2020; Li, Sun, & Crouseilles, 2020; Rani & Mishra, 2020; Smitha & Nagaraja, 2020; S. Sun et al., 2020; Vargas-González et al., 2020; Xiong, Wen, & Zheng, 2020; Xu, Huang, & Ma, 2020; Yan et al., 2021; J. Zhao et al., 2020; X. Zhao et al., 2020; Zhong & Sheng, 2020; Zubar et al., 2020). Hereafter the different scholars applied some techniques on strand meshes in complex flow fields and aerodynamics problem. Precision the details, mesh generation need to explore, in past two decades, regular and irregular mesh generations are solved for finite element and boundary element methods to solve the physical, engineering and scientific problems (Sankarganesh, Gowtham, Thamilarasan, & Karthik, 2018; Song & Karniadakis, 2019). Unstructured or random meshes depend upon the irregular, dispersed, or scatter points in which the partial differential equation is solved for local smoothing. Therefore, Classical FD schemes are bound or restricted on uniform grids with regular geometry. The finite random grids are frequently appropriate for FEM but need to expand on FDM (Habeeb et al., 2020). There is no appropriate data available data for the finite difference method using random grids. The idea is recommended in the literature (Habeeb et al., 2020). Therefore, the motivation and interest developed to explore the effect of random grids using the finite difference method. A physical model is chosen as1D, 2D & 3D differential equations, from which the detailed data is collected and implemented on 2D PDEs with Dirichlet boundary conditions for further investigation, implementation, and testing purposes. The methods are based on discretization of the differential equation by finite difference quotients, parameters of mesh. It is frequently observed that the finite difference method (FDM) operates on the regular step or equal step size or variable step grid size.

2.9 SM'S Method

In this thesis, the method known as SM's Method is focused. This method is an approximate method, which is abbreviated from Sanaullah Mastoi's Method. This method uses the numerical solution of partial differential equations using the finite difference method over randomly generated grids. Mesh generation does not have any specific formula or principle. In this method, numerical solutions found converging, rapid solutions with less computational time and less error.

2.10Summary

This chapter discusses some basic definitions and concepts related to the solution of differential equations using analytical method, partial differential equation, finite difference method, and thoroughly mesh generation process. Furthermore, some related works regarding mesh generation and its applications are reviewed.

The solution of PDEs is quite challenging. The number of methods available to find closed-form analytical solutions to canonical PDEs is limited. These include separation of variables, superposition, product solution methods, Fourier transforms, Laplace transforms, and perturbation methods, among a few others. Even these methods are limited by constraints such as regular geometry, linearity of the equation, constant coefficients, and others. The imposition of these constraints severely curtails the range of applicability of analytical techniques for solving PDEs, rendering them almost irrelevant for problems of practical interest. In realization of this fact, applied mathematicians and scientists have endeavored to build machines that can solve differential equations by numerical means, as outlined in the brief history of computing presented at the beginning of this chapter.

The methods for numerical solutions to PDEs can broadly be classified into two types: deterministic and stochastic. A deterministic method is one in which the output is always the same for a given input to an equation. The outcome does not depend on how many times one solves the equation, when it is solved, or what computer it is solved on (disregarding precision errors, which may be slightly different on different computers). On the other hand, a stochastic method is based on statistical principles. The output can be somewhat different for the same input depending on how many times the calculation is performed and other factors. In this case, by "slightly different," we mean within the statistical error bounds. A simple example best elucidates the difference between these two approaches. Let us consider a scenario in which a ball is released from a certain height above the horizontal ground. Upon collision with the ground, the ball bounces back to a size ho. Let us assume that based on experimental observations or other physical laws, we know that the ball always bounces back to half the height from which it is released. Following this information, we may construct the following deterministic equation: h = (1 / 2)h0 i. If this equation is used to compute the height of a bounced ball, it would always be one-half of release height. In other words, the equation (or method of calculation) has one hundred percent confidence built into it. Hence, it is termed a deterministic method.

The stochastic viewpoint of the same problem would be pretty different. In this viewpoint, one would argue that if n balls were made to bounce, by the laws of theoretical probability, n/2 balls would boot to a height slightly above half the released size. The remaining n/2 balls would leap to a height slightly below the released peak, such that in the end, when tallied, the mean height to which the balls bounce back to would be exactly half the height of release. Whether this exact result is recovered or not would depend on how many balls are bounced, i.e., the number of statistical samples used. The stochastic viewpoint may be implemented, for example, by drawing random numbers from a uniform deviate between 0 and 1. If the random number is more significant than half, then the ball is made to bounce to a height greater than half the released height, while if it is less than half, it is made to reflect a size less than half the released height. This viewpoint is more in keeping with experimental observations because, in reality, it is unlikely for each ball to bounce back to exactly half the size from which it is released. In other words, statistical variability, as prevalent in any experimental measurement, is already built into the stochastic viewpoint.

The next chapter will demonstrate the methodology of research that is mainly focused on the randomly generated meshes.

CHAPTER 3:

RESEARCH METHODOLOGY

3.1 Introduction

This chapter demonstrates the detailed methodology of the proposed research. First of all, the model problem specifications are described, and then the finite difference discretization of the governing equation is presented. After that, the mesh generation process is defined for both uniform and random spacing. Finally, for the analysis of results, sufficient data is collected in terms of uniform and random meshes, viz., different meshes of uniform spacing are generated, and meshes with random spacing are generated.

3.2 Model problem specification

The systematic procedure is followed to achieve the objectives. First, the governing equation is chosen as one, two, and three-dimensional partial differential equations and applied with initial and boundary conditions, as shown in Figure 3.1.

$$\left.\begin{aligned}
&u_{xx}(x,y,z)+u_{yy}(x,y,z)+u_{zz}(x,y,z)=-f(x,y,z)\\
&a<x\le b, a<y\le b \ \& \ a<z\le b\\
&\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}=-f(x,y,z)\\
&a\le x\le b, a\le y\le b \ \& \ a\le z\le b \text{ with } u(x,y,z)=0\\
&u(p,y,z)=a_1, u(q,y,z)=a_2, u(x,p,z)=b_1, u(x,q,z)=\\
&b_2, u(x,y,p)=c_1 \ \& \ u(x,y,q)=c_2\\
&u_{xx}+u_{yy}+u_{zz}=u_{rr}+\frac{1}{r}u_r+\frac{1}{r^2}u_{\theta\theta}+\frac{1}{r\sin\theta}u_{\vartheta\vartheta}=f(r,\theta,\vartheta),\\
&0<r<m, 0<\theta<n\pi, 0<\vartheta\le p\pi
\end{aligned}\right\} \tag{3.1}$$

where u is the dependent variable on $x, y,$ and z (in polar (r, θ and ϑ)) coordinates. f is a given forcing function, where a_1, a_2, b_1, b_2, c_1 and c_2 are the left, right, top, bottom, front and back boundary conditions, respectively; six faces, 12 edges, and 8 vertices, respectively.

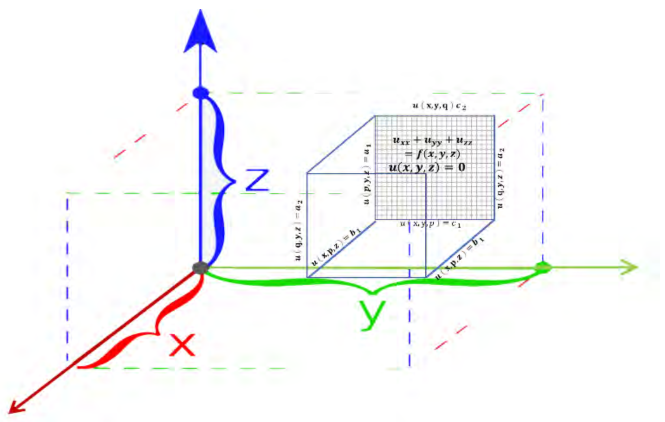


Figure 3.1: Model problem with boundary conditions

3.3 Finite difference discretization of the model

After the model problem specification, the governing equation is discretized by using the finite difference schemes. Since the problem is well-posed and does not require any special treatment regarding consistency and stability. Therefore, the second-order accurate central finite difference scheme is used, and the following Eq describes the resulting system. 3.2 (a-c) as,

$$\frac{u(i+1,j,k)-2u(i,j,k)+u(i-1,j,k)}{e^2}+\frac{u\,(i,j+1,k)-2u(i,j,k)+u(i,j-1,k)}{g^2}+\frac{u\,(i,j,k+1)-2u(i,j,k)+u(i,j,k-1)}{h^2}=-\,f\,(i\,,j,k)$$

for all $i,j,k \in \mathbb{N}, 1 \le i \le m, 1 \le j \le n, \& \; 1 \le k \le l,\; with\; u(i,j,k) = f$

$u\,(1,j,k\,) = \,a_{\,1}\,,\; u(\,m,j,k\,) = \,a_{\,2}\,,\; u(\,i,1,k) = \,b_1\,,$

$u(\,i,n,k)\; = b_2, u(\,i,j,1) = \,c_1,\; u(i\,,j,l)\; = c_2$ (3.2 a)

$$\frac{u(i+1\,,\;\,j\,)-2u\,(\,i\,,j\,)+u\,(\,i-1,\;\,j)}{e^2}+\frac{u\,(i\,,j\,+1\,)-2u\,(\,i\,,j)+u(\,i\,,j-1\,)}{g^2}=-\,f\,(i\,,j)$$

for all $i,j \in \mathbb{N}, 1 \le i \le m, 1 \le j \le n,\; with\; u(i,j) = f$

$u\,(\,1,j\,) = \,a_{\,1}\,,\; u(\,m,j\,) = \,a_{\,2}\,,\; u(\,i,1) = \,b_1,\; u(\,i,n) = b_2$ (3.2 b)

$$\frac{u(i+1\,)-2u(i)+u(i-1)}{e^2}=f(i)$$

for all $i,j \in \mathbb{N}, 1 \le i \le m, u\,(\,1\,) = \,a_{\,1}\,,\; u(\,m\,) = \,a_{\,2}$ (3.2c)

The computational domain is discretized accordingly as a finite difference mesh. A schematic of discretized mesh with uniform spacing for 1D, 2D and 3D are exhibited in Figure 3.2(a-c).

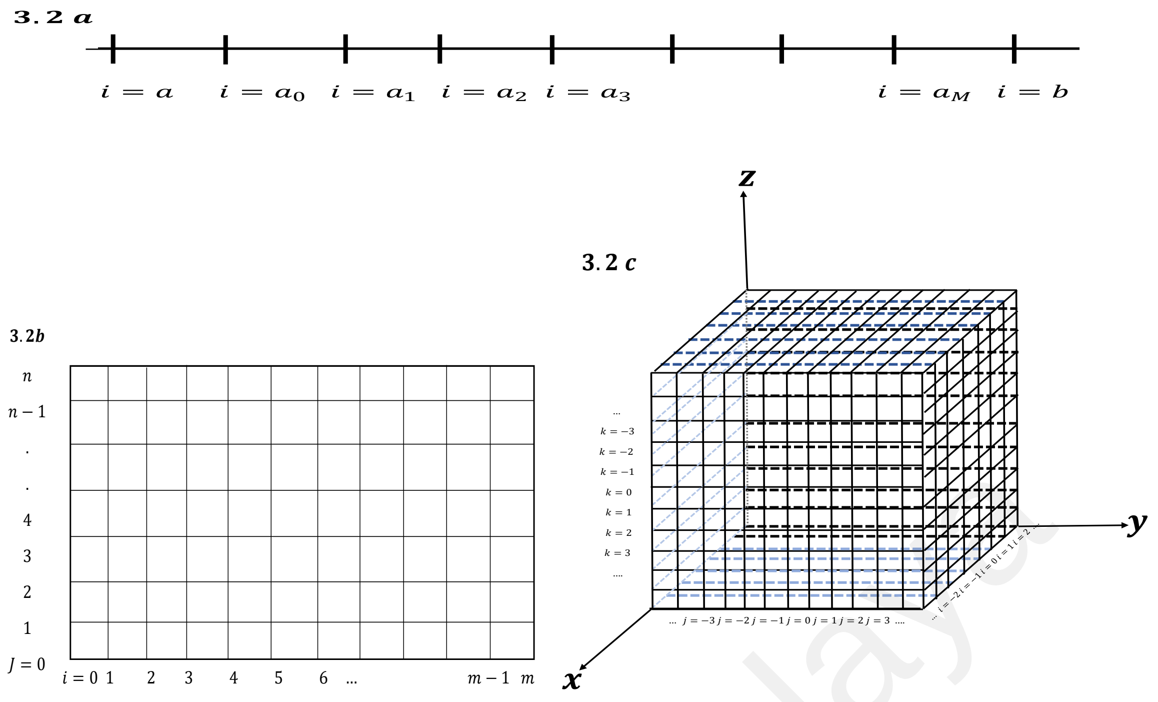


Figure 3.2: The schematic of discretized domain for (a) 1-D, (b)2-D & (c)3-D

In figure 3.2. the uniform grids are applied on the interval from a to b in the figure 3.2 a, 0 to m in x- axes, 0 to n in y-axis in figure 3.2b and in the figure 3.2c the axes are equal in length that is a to b. the uniform or regular grids.

3.3.1 Generation of uniform finite-difference grids

To solve the governing equation and obtain sufficient numerical results, the ten different sizes mesh equally spaced increments in one, two, and three dimensions along with x, x and y, and x, y *and* z axes are generated in uniform meshes with one two and three dimensions respectively. The mesh data is shown in the following Table 3.1(a-c),

Table 3.1: (a-c): Data for Uniform Mesh Generations (1-D, 2-D and 3-D)

Table 3.1 a: 1-D

Mesh size	10	20	30	40	50	60	70	80	90	100
Step size	0.1	0.05	0.03	0.025	0.020	0.017	0.014	0.013	0.011	0.01
Number of nodes	11	21	31	41	51	61	71	81	91	101
Cell Standard deviations	0	0	0	0	0	0	0	0	0	0

Table 3.1 b : 2-D

S. No	Mesh size	Step size	Number of nodes	Number of cells	Average cell size	Cell standard deviation
1	10×10	0.100	121	100	0.100	0
2	20×20	0.050	441	400	0.050	0
3	30×30	0.033	961	900	0.033	0
4	40×40	0.025	1681	1600	0.025	0
5	50×50	0.020	2601	2500	0.020	0
6	60×60	0.017	3721	3600	0.017	0
7	70×70	0.014	5041	4900	0.014	0
8	80×80	0.013	6561	6400	0.013	0
9	90×90	0.011	8281	8100	0.011	0
10	100×100	0.010	10201	10000	0.010	0

Table 3.1 c: 3-D

S. No	Mesh size	Step size	Number of nodes	Number of cells	Average cell size	Cell standard deviation
1	$10 \times 10 \times 10$	0.100	1331	1000	0.10	0
2	$20 \times 20 \times 20$	0.050	9261	8000	0.05	0
3	$30 \times 30 \times 30$	0.033	29791	27000	0.03	0
4	$40 \times 40 \times 40$	0.025	68921	64000	0.02	0
5	$50 \times 50 \times 50$	0.020	132651	125000	0.02	0
6	$60 \times 60 \times 60$	0.017	226981	216000	0.017	0
7	$70 \times 70 \times 70$	0.014	357911	343000	0.014	0
8	$80 \times 80 \times 80$	0.013	531441	512000	0.013	0
9	$90 \times 90 \times 90$	0.011	753571	729000	0.011	0
10	$100 \times 100 \times 100$	0.010	1030301	1000000	0.01	0

The uniform meshes are generated by writing a MATLAB code and meshes constructed by using the data given by Table 3.1(a-c). Then numerical solution is implemented on each interior node by explicit scheme by Eq. (3.3) as follows.

$$u_{p+1} = (\delta^2 + 2)u_p - u_{p-1} \tag{3.3a}$$

$$u^{q+1}(i,j) = \frac{k^2 \times \left(h^2 \times f(i,j) + u^q(i+1,j) - u^q(i-1,j)\right) + h^2 \times \left(u^q(i,j+1) - u^q(i,j-1)\right)}{2(h^2+k^2)} \tag{3.3b}$$

$$u^{r+1}(i,j,k) = \frac{(egh)^2 f(i,j,k)\ [\big((gh)^2(u^r(i+1,j,k)-u^r(i-1,j,k)\big) + (eh)^2\big(u^r(i,j+1,k)-u^r(i,j-1,k)\big)+(eg)^2\big(u^r(i,j,k+1)-u^r(i,j,k-1)\big)]}{2(g^2h^2+e^2h^2+g^2e^2)} \tag{3.3c}$$

where p, q, r stands for iteration, and $p + 1, q + 1$ & $r + 1$ are the successive iteration.

3.3.2 Generation of random finite difference meshes

Just like the generation of uniform meshes, the samples of random meshes are required to test their feasibility, as mentioned in the objectives of this research. It can be observed from the data of uniform meshes (Table 3.1(a-c)) that the meshes parameters like step sizes for one dimension e, for two dimensions e & g and three dimensions e, g & h, cell are same. Cell sizes may be the minimum cell size, maximum cell size and average cell size remain the same for the size of a particular grid. However, when random meshes are generated, all parameters change randomly for specific grid size, and as many times the mesh is generated, the mesh parameters vary. For testing and implementation purposes, ten samples of each mesh size ($10, 20, 30, 40, \dots 100$, $10 \times 10, 20 \times 20, \dots 100 \times 100$, $10 \times 10 \times 10, 20 \times 20 \times 20, \dots 100 \times 100 \times 100$ for one, two, and three dimensions, respectively) are obtained on MATLAB and ANSYS FLUENT software using the built-in function rand for MATLAB (given mesh size). It means for all ten different mesh sizes. We have ten different samples (or realizations) totaling 10,100 and 1000 meshes each one is different for one, two, and three dimensions. The mesh parameters are further extended as minimum cell size, maximum cell size, average cell size, the standard deviation of cells, skewness of cells, correlation in cells along axes like x, x & y, & x, y & z directions. These parameters help choose an appropriate numerical solution scheme and for selecting random samples with fast convergence. For uniform meshes, the numerical solution can be implemented using Eq. 3.3 $(a - c)$, but for random meshes, the situation changes due to instantaneous changes in the step size e, g, and h, as shown in Figure 3.3$(a - c)$. Here three different approaches can handle the unequal step size maximum, minimum and average cell size. e_{max}, g_{max} *and* h_{max} be the excellent selection, detailed discussed in the chapter of solution profile of randomly generated grids.

$$u_{p+1} = \left(\delta_{max}{}^{2} + 2\right)u_p - u_{p-1} \tag{3.4a}$$

$$u^{\,q+1}(\,i\,,j\,) = \frac{k_{max}{}^{2} \times \left(\,h_{max}{}^{2} \times\, f\,(\,i\,,j\,) + u^{\,q}\,(\,i+1\,,j\,) -\, u^{\,q}\,\,(\,i-1\,,j\,)\,\right) + h_{max}{}^{2} \times \left(\,u^{\,q}\,(\,i\,,j+1\,) -\, u^{\,q}\,\,(\,i\,,j-1)\right)}{2\left(h_{max}{}^{2}+k_{max}{}^{2}\right)} \tag{3.4b}$$

$$u^{\,r+1}(\,i\,,j,k\,) =$$

$$\frac{(e_{max}g_{max}h_{max})^2 f\,\,(\,i\,,j\,,k)\,\,\left[\left((g_{max}h_{max})^2(u^r\,(i{+}1,j,k){-}u^r(i{-}1,j,k)\right) + (e_{max}h_{max})^2\left(u^r(i,j{+}1,k){-}u^r\,(i\,,j{-}1,k)\right) + (e_{max}g_{max})^2\left(u^r(i,j,k{+}1){-}u^r\,(i\,,j,k{-}1)\right)\right]}{2\left(g_{max}{}^{2}h_{max}{}^{2}+e_{max}{}^{2}h_{max}{}^{2}+g_{max}{}^{2}e_{max}{}^{2}\right)} \tag{3.4c}$$

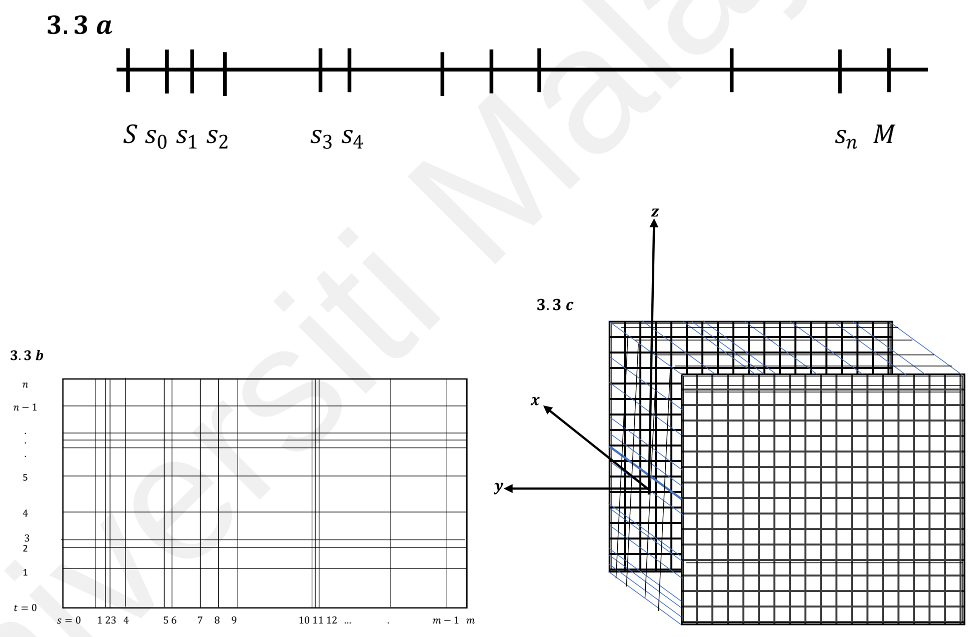


Figure 3.3 (a-c): Random spacing on sample meshes

3.4 Algorithm for the implementations of methodology

Most of the significant steps for implementing the proposed research have already been described in the previous sections. However, the systematic procedure for both the uniform and random meshes are summarized separately in Tables 3.3-3.4.

3.4.1 Algorithm for the implementation of a methodology for uniform meshes

The systematic procedure followed for generating and implementing the uniform grids. In this procedure, the most critical cell size, converging iterations, computational time, and error analysis.

Table 3.2: Algorithm using uniform meshes

Steps	Description
	Initialize the algorithm MATLAB and ANSYS
1	Generate the ***step size*** as data points with uniform step sizes for dimensions as one, two, and three dimensions as ***e, g, and h,*** respectively
2	Draw the **computational domain** and assign the **boundary conditions**
3	Assign the mesh size ***l, m, n*** and construct the uniform mesh
4	Start iteration $p = p + 1, q = q + 1, r = r + 1$ $with$ $\boldsymbol{p, q, r = 0}$ ***as initial***
5	Run the solution algorithm **Eq. 3.3($\boldsymbol{a - c}$),** with predefined error
6	**If the** solution is converged, then stop. **Else** repeat steps **four** and step **five**
7	Count the converging **iterations** and CPU as **computational time (In sec)**
8	Solution profiling

3.4.2 Algorithm for the implementation of a methodology for random meshes

The systematic procedure for the novel method (SM's Method), which is randomly generated grids, is followed for developing and implementing the finite random grids. This procedure generates maximum, minimum, and average cell size, statistical parameters, converging iterations, computational time, and error analysis.

Table 3.3: Algorithm using random meshes

Steps	Description
	Initialize the algorithm on MATLAB and ANSYS
1	Generate the ***Step size*** data points with random step sizes: ***e, g, and h,*** respectively.
2	Draw the **computational domain** and assign the **boundary conditions**
3	Construct the samples of random mesh size $\boldsymbol{l, l \times m, l \times m \times n}$ and save as .xls file for one, two, and three dimensions, respectively.
4	Import the stored (.xls) file in the MATLAB program and locate the specific **random sample**
5	Collect the statistical parameters and compute the maximum step sizes
6	Start iteration *p, q, r with p+1, q+1 and r+1, where* ***p,q,r=0 as initial***
7	Run the solution algorithm **Eq. 3.4($\boldsymbol{a - c}$),** with predefined error
8	**If** the solution is converged, then stop. **Else** repeat steps **four** & step **five**
9	Count the converging **iterations** and CPU as **computational time (In sec)**
10	Solution profiling, analysis of the accuracy

3.4 Summary

In this chapter, the complete methodology is demonstrated for obtaining the objectives of the proposed research, which is mainly associated with the analysis of uniform and random finite difference meshes. Therefore, the first model problem is briefly described, and then the procedure to generate the uniform and random meshes is explained. The input mesh data is given and shown by Tables and Figures, respectively. The step-by-step implementation of the methodology has also been illustrated that will provide the statistical and numerical solution results. The next chapter will elaborate on uniform meshes, random meshes, and their results from implementing the methodology. The concealed ideas related to the proposed research are expected to be discovered and justified.

CHAPTER 4: SOLUTIONS PROFILE ON UNIFORM GRIDS

4.1 Introduction

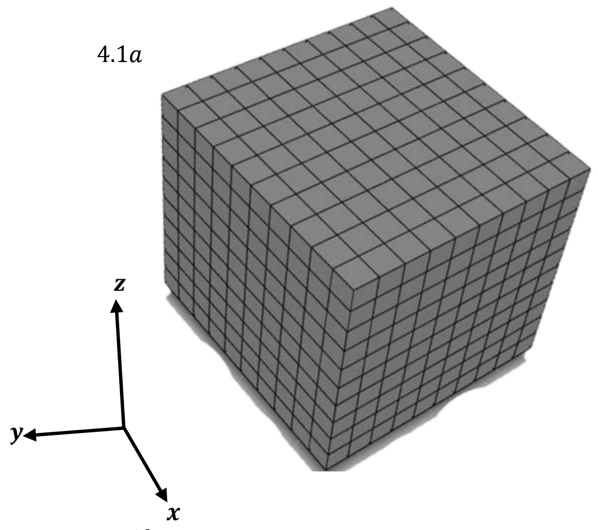
The research methodology was already described concisely in the previous chapter. This chapter demonstrates and analyzes the output results obtained from the implementation of methods through generating uniform grids. In addition, the numerical solution profiles are exhibited and interpreted. Finally, it is essential to mention that in this research, ten uniform meshes are studied.

4.2 Generation of uniform finite-difference grids

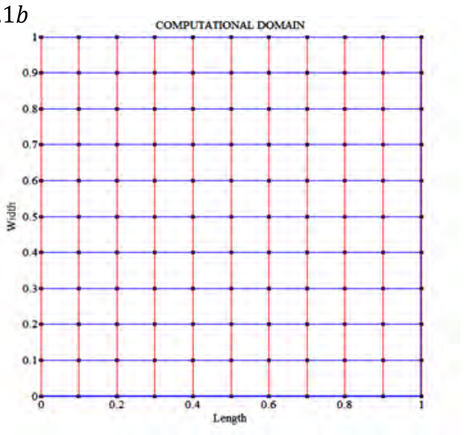
The uniform meshes are generated by writing a MATLAB code. The meshes constructed by using the data given in Table 3.1 are shown in Figure 4.1. Then the numerical solution is implemented on each interior node by the explicit method by using Eq. (3.3).

The uniform grids are generated using second-order partial differential equations in one, two, and three dimensions. The Numerical simulation of one, two, and three dimensional in uniform meshes of size $10, 20, 30, 40, \dots, 100$, $10 \times 10, 20 \times 20, \dots, 100 \times 100$ & $10 \times 10 \times 10, 20 \times 20 \times 20, \dots, 100 \times 100 \times 100$ respectively.

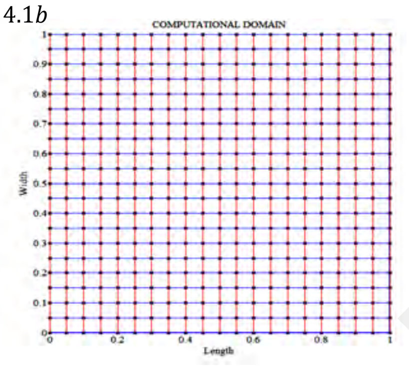
$4.1a$



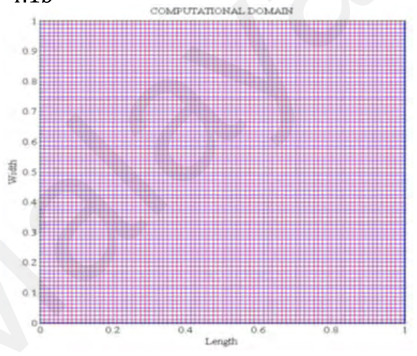
$4.1b$



$4.1b$



$4.1b$



$4.1c$

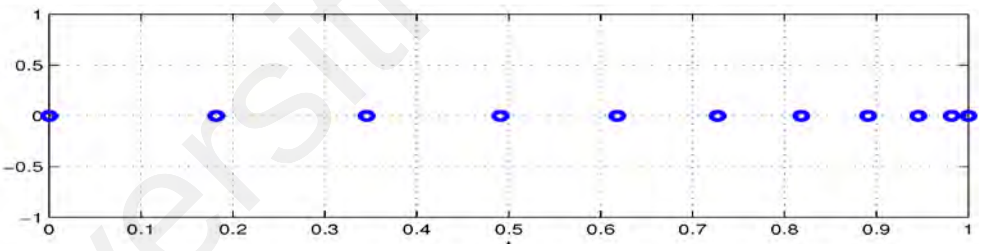


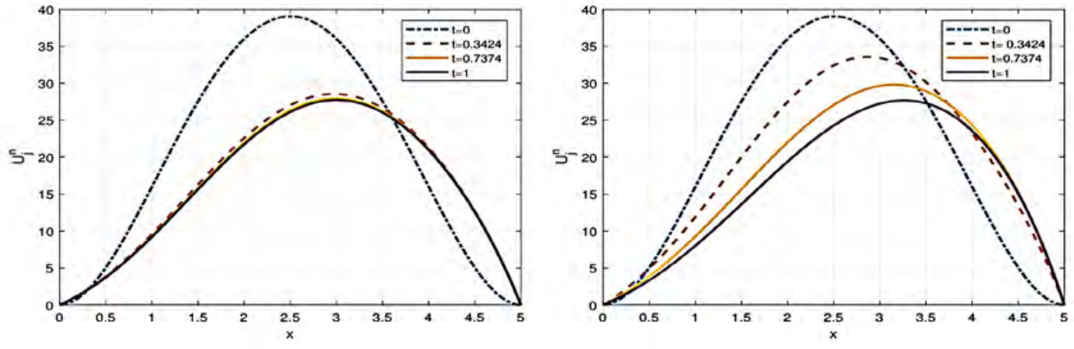
Figure 4.1: *a*- three-dimensional, *b*- two dimensional and *c*- three dimensional unfirom grids of second order PDEs.

The uniform grids are generated using MATLAB codes, and the step size can be presented graphically and tabularly. For example, in Figure 4.1 a., the three-dimensional uniform grids are shown having cell sizes from $10 \times 10 \times 10, 20 \times 20 \times 20, \ldots, 100 \times 100 \times 100$ for the three-dimensional domain, where step sizes are represented by e, g, h or g, h and k with an equal step size in all dimensions. The help of ANSYS FLUENT software generates the three-dimensional grids because of the enormous data and cell size. Therefore, it is challenging to get the results through MATLAB Program. Similarly, the two-dimensional uniform grids are generated with the help of MATLAB and ANSYS FLUENT software, Figure 4.1 b. shows the data collected for two-dimensional grids $10 \times 10, 20 \times 20, 30 \times 30, 40 \times 40, \dots, 90 \times 90, 100 \times 100$ step size in a given domain. Finally, we have stored one-dimensional step size using MATLAB and ANSYS FLUENT software, managed grids, and stored in the tabular form with steps $10, 20, 30, 40, 50, \dots, 100$.

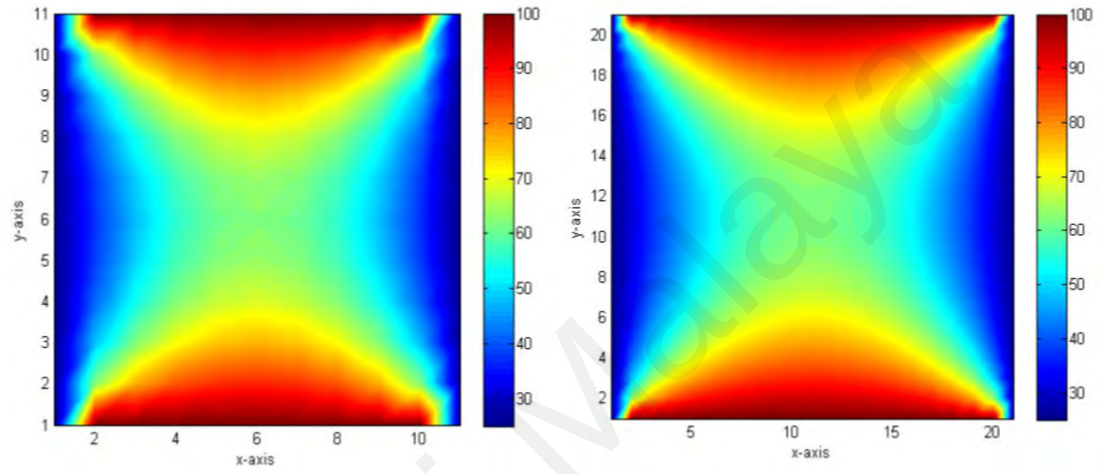
4.3 Numerical solution profiles over uniform meshes

Figures 4.2$(a - c)$ shows that the numerical profile simulation of the one-dimensional, two- dimensional and three dimensional PDEs respectively. The one dimensional time-fractional advection-diffusion equation using uniform grids $10, 20, 30, 40, 50, \dots, 100$ respectively. The steady-state temperature in the unit square over uniform meshes of size $10 \times 10, 20 \times 20, \dots, 100 \times 100$, respectively. Each figure exhibits a local solution profile where the numerical solution values vary from 25^{o}Cto 100^{o}C. The solution's smoothness verifies the problem's well-posedness and consistency, which are the best properties of Poisson’s equation. Increasing the mesh size increases the smoothness. The purpose of obtaining these simulation profiles is to use them as benchmarks for the simulation profiles obtained on random meshes. It will facilitate deciding on the feasibility of using random meshes for such problems. Three-Dimensional facet reflection in optical waveguides is solve using the finite difference method over uniform grids sizes which are $10 \times 10 \times 10, 20 \times 20 \times 20, \dots, 100 \times 100 \times 100$, respectively. The finite difference method simulates devices with a complex structure.

$4.2\ a$



$4.2\ b$



$4.2\ c$

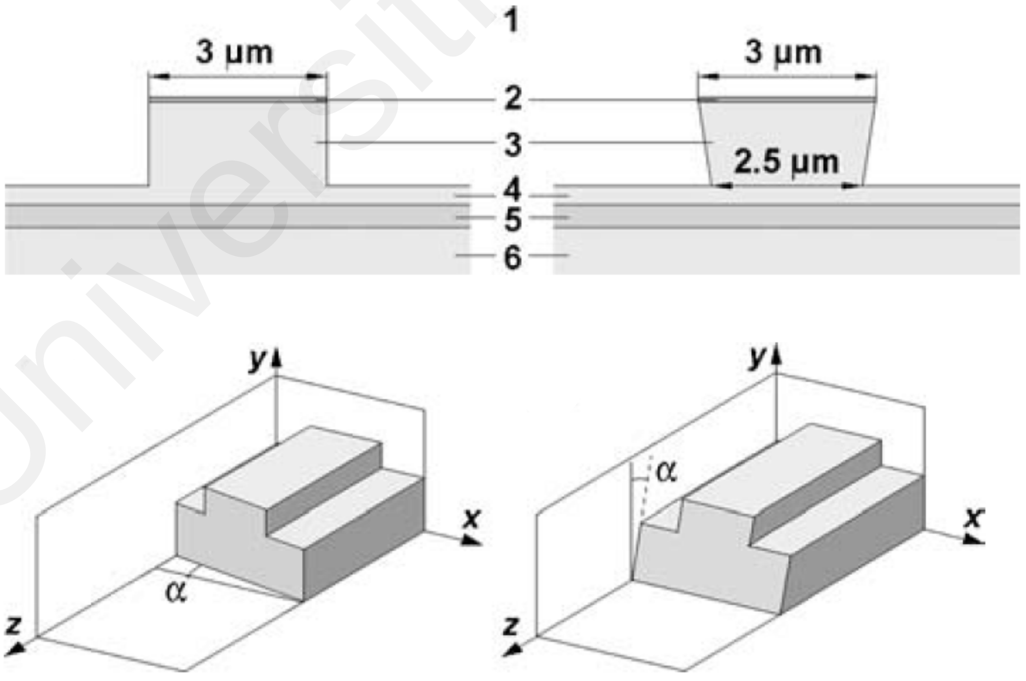


Figure 4.2: Local solution profile on Uniform mesh of size (a-1D, b-2D, and 3-D)

The finite difference method uses uniform grids to solve one-dimensional time-fractional advection-diffusion equations involving the Caputo fractional derivative. The Caputo time derivative is discretized employing a direct generalization of the well-known fractional to the case of uniform meshes in figure 4.2 a.

The 2-D PDEs is solved through the profile simulations , in Figure 4.2b, shows the profile smoothness, where the solution veries from 25^{0}C to 100^{0}C. The problem's well-posedness and consistency, which are the best properties of Poisson’s equation.

The 3D method has been applied successfully applied to the calculation of waveguide facet reflection. Parallel computing has been used to overcome large memory consumption and is expected to achieve high precision in measures while using randomly generated grids. The 3D-FDTD algorithm implemented through parallel computing can handle all the structural varieties and is sufficiently accurate to capture even subtle structural differences. The problem of facet reflection has been studied using the example of rectangular-shaped and reverse trapezoid-shaped ridge waveguides and different tilting schemes. This approach has shown that a horizontally tilted waveguide with a rectangular-shaped ridge best reduces facet reflection.

4.4 Summary

In this chapter, uniform grids are shown in the form of the figures. The complete methodology is followed, demonstrated for obtaining the analysis of uniform finite difference meshes. The purpose of bringing these simulation profiles is to use as benchmarks for the simulation profiles obtained on random meshes. Moreover, it will facilitate making the decision about the feasibility of using random meshes for such problems. The next chapter will be elaborated for random meshes and their simulation profiles obtained from implementing the methodology where the concealed ideas related to the proposed research are expected to be discovered and justified.

CHAPTER 5:

SOLUTIONS PROFILE ON RANDOM GRIDS

5.1 Introduction

In the previous chapter, the uniform grids and profile simulations are obtained by following the research methodology described. This chapter demonstrates and analyzes the output results obtained from the implementation of methods through generating randomly grids. The numerical solution profiles are exhibited and analyzed. It is essential to mention that in this research that hundred random meshes are studied.

5.2 Generation of random finite difference grids

The mesh generation process for randomly generated meshes are same as the generation of uniform meshes. The samples of random meshes are required to test their feasibility as mentioned in the objectives of this research. It can be observed from the data of uniform meshes (Table 3.1) that the mesh parameters like step sizes e, or h, k and e, g, h or g, h, k, for one dimension, two dimension and three dimension respectively. The step size has minimum cell size, maximum cell size, and average cell size remains the same for a particular grid size. However, when a random mesh is generated, all parameters change randomly for a specific grid size. As many times the mesh is generated so many times the mesh parameters vary. For testing and implementation purposes, many samples like ten, for each mesh size that is $10, 20, 30, 40, \ldots\ 90, 100$, $10 \times 10, 20 \times 20, \ldots\ 100 \times 100$, and $10 \times 10 \times 10, 20 \times 20 \times 20, 30 \times 30 \times 30, \ldots, 100 \times 100$ for 1-D, 2-D and 3-D respectively, are obtained on MATLAB, ANSYS FLUENT using the built-in function rand (given mesh size) for MATLAB. It means for all different mesh sizes, and we have various samples (or realizations), the meshes are different from each other. The mesh parameters are further extended as minimum cell size, maximum cell size, average cell size, the standard deviation of cells, skewness of cells, correlation in cells along x, x & y

, and x, y & z direction for 1D, 2D and 3D respectively. These parameters help choose an appropriate numerical solution scheme and select random samples with fast convergence.

Table 5.1: Random samples for random mesh generation of size 10×10

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$10 \times 10\ (1)$	1.7×10^{-4}	0.0995	0.01	0.0094	0.424	0.222
2	$10 \times 10\ (2)$	2.7×10^{-4}	0.0574	0.01	0.0059	-0.114	0.126
3	$10 \times 10\ (3)$	1.2×10^{-4}	0.0497	0.01	0.0047	-0.171	0.227
4	$10 \times 10\ (4)$	1.5×10^{-4}	0.0932	0.01	0.0077	-0.156	-0.057
5	$10 \times 10\ (5)$	1.4×10^{-4}	0.0625	0.01	0.0061	0.098	-0.268
6	$10 \times 10\ (6)$	1.6×10^{-4}	0.1086	0.01	0.0091	-0.189	0.291
7	$10 \times 10\ (7)$	7.6×10^{-5}	0.0945	0.01	0.0071	-0.354	0.046
8	$10 \times 10\ (8)$	2.2×10^{-4}	0.0404	0.01	0.0041	0.114	0.351
9	$10 \times 10\ (9)$	3.9×10^{-4}	0.0674	0.01	0.0061	-1.246	-0.018
10	$10 \times 10\ (10)$	8×10^{-5}	0.1414	0.01	0.0126	1.0312	0.307

Table 5.2: Random samples for random mesh generation of size 20×20

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$20 \times 20(1)$	2.3×10^{-5}	0.0283	0.0025	0.0018	-0.591	0.127
2	$20 \times 20(2)$	2×10^{-6}	0.0369	0.0023	0.0025	-1.033	0.058
3	$20 \times 20(3)$	4×10^{-6}	0.0431	0.0025	0.0028	-1.297	-0.166
4	$20 \times 20(4)$	1×10^{-6}	0.0339	0.0025	0.0024	-1.556	-0.119
5	$20 \times 20(5)$	2×10^{-6}	0.0339	0.0025	0.0022	1.515	0.156
6	$20 \times 20(6)$	5×10^{-6}	0.0252	0.0023	0.0018	0.731	0.355
7	$20 \times 20(7)$	1.9×10^{-5}	0.0225	0.0023	0.0014	0.178	0.232
8	$20 \times 20(8)$	7×10^{-6}	0.0281	0.0025	0.0018	1.587	0.03
9	$20 \times 20(9)$	1×10^{-6}	0.0334	0.0019	0.0022	-0.691	-0.054
10	$20 \times 20(10)$	1×10^{-7}	0.0298	0.0022	0.0023	-0.926	0.052

Table 5.3: Random samples for random mesh generation of size 30×30

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$30 \times 30(1)$	1×10^{-7}	0.0144	0.001	0.0008	-0.219	-0.079
2	$30 \times 30(2)$	2×10^{-6}	0.0146	0.001	0.0008	-0.474	0.167
3	$30 \times 30(3)$	1×10^{-8}	0.0146	0.001	0.0009	-1.455	0.075
4	$30 \times 30(4)$	6×10^{-6}	0.0142	0.001	0.001	1.166	-0.0072
5	$30 \times 30(5)$	1.5×10^{-7}	0.013	0.001	0.0007	-0.175	-0.279
6	$30 \times 30(6)$	1×10^{-8}	0.0396	0.001	0.001	-0.058	-0.145
7	$30 \times 30(7)$	1×10^{-9}	0.0099	0.001	0.0007	0.15	0.082
8	$30 \times 30(8)$	2×10^{-6}	0.0109	0.001	0.0007	-0.66	0.358
9	$30 \times 30(9)$	1×10^{-8}	0.0175	0.001	0.0009	1.29	-0.02
10	$30 \times 30(10)$	2×10^{-9}	0.017	0.001	0.001	-1.43	-0.007

Table 5.4: Random samples for random mesh generation of size 40×40

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$40 \times 40(1)$	1×10^{-7}	0.013	0.0006	0.000545	-0.353	0.204
2	$40 \times 40(2)$	1×10^{-8}	0.013	0.0006	0.000572	0.404	0.325
3	$40 \times 40(3)$	3×10^{-6}	0.012	0.0006	0.000473	0.154	-0.198
4	$40 \times 40(4)$	1×10^{-8}	0.016	0.0006	0.000634	0.0198	-0.068
5	$40 \times 40(5)$	1×10^{-8}	0.007	0.0006	0.000502	0.612	-0.039
6	$40 \times 40(6)$	1×10^{-6}	0.007	0.0006	0.000511	1.378	-0.163
7	$40 \times 40(7)$	2×10^{-7}	0.008	0.0006	0.000483	-0.195	0.152
8	$40 \times 40(8)$	3×10^{-8}	0.023	0.0006	0.000922	-0.091	-0.028
9	$40 \times 40(9)$	1×10^{-9}	0.014	0.0006	0.000584	-1.456	-0.127
10	$40 \times 40(10)$	1×10^{-7}	0.022	0.0006	0.000856	0.835	0.049

Table 5.5: Random samples for random mesh generation of size 50×50

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$50 \times 50(1)$	1×10^{-7}	0.0092	0.0004	0.00046	0.994	0.191
2	$50 \times 50(2)$	1×10^{-7}	0.0043	0.0004	0.00031	1.157	0.091
3	$50 \times 50(3)$	1×10^{-7}	0.0079	0.0004	0.00039	-1.414	-0.068
4	$50 \times 50(4)$	1×10^{-8}	0.0061	0.0004	0.00038	-0.805	-0.151
5	$50 \times 50(5)$	1×10^{-7}	0.0044	0.0004	0.00029	0.892	-0.034
6	$50 \times 50(6)$	1×10^{-8}	0.0096	0.0004	0.00037	-1.618	0.122
7	$50 \times 50(7)$	1×10^{-9}	0.0097	0.0004	0.00043	1.543	0.233
8	$50 \times 50(8)$	1×10^{-7}	0.0043	0.0004	0.00031	1.515	0.317
9	$50 \times 50(9)$	1×10^{-7}	0.0057	0.0004	0.00033	0.394	0.0163
10	$50 \times 50(10)$	2×10^{-8}	0.0044	0.0004	0.00022	0.767	0.272

Table 5.6: Random samples for random mesh generation of size 60×60

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$60 \times 60(1)$	1×10^{-7}	0.0035	0.00028	0.0002	-1.301	0.033
2	$60 \times 60(2)$	1×10^{-7}	0.0052	0.00028	0.0003	0.877	0.0122
3	$60 \times 60(3)$	1×10^{-7}	0.0072	0.00028	0.0002	1.429	-0.017
4	$60 \times 60(4)$	1×10^{-7}	0.0057	0.00028	0.0002	-1.316	-0.015
5	$60 \times 60(5)$	1×10^{-7}	0.0046	0.00028	0.0002	-1.456	0.0385
6	$60 \times 60(6)$	1×10^{-6}	0.0038	0.00028	0.0002	1.556	0.0313
7	$60 \times 60(7)$	1×10^{-7}	0.0089	0.00028	0.00031	-0.286	-0.235
8	$60 \times 60(8)$	1×10^{-7}	0.0046	0.00028	0.00026	1.317	0.0398
9	$60 \times 60(9)$	1×10^{-7}	0.0065	0.00028	0.00025	1.263	-0.236
10	$60 \times 60(10)$	1×10^{-7}	0.0035	0.00028	0.00023	-0.422	0.031

Table 5.7: Random samples for random mesh generation of size 70×70

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$70 \times 70(1)$	9e-8	0.007	0.00021	0.0002	-1.401	-0.0412
2	$70 \times 70(2)$	1×10^{-7}	0.003	0.00021	0.0001	-1.428	-0.0191
3	$70 \times 70(3)$	2×10^{-8}	0.005	0.00021	0.0002	-1.407	-0.0733
4	$70 \times 70(4)$	1×10^{-8}	0.004	0.00021	0.0002	-1.332	-0.0134
5	$70 \times 70(5)$	2×10^{-8}	0.003	0.00021	0.0002	-1.386	0.0394
6	$70 \times 70(6)$	1×10^{-7}	0.005	0.00021	0.0002	1.594	0.1395
7	$70 \times 70(7)$	1×10^{-7}	0.008	0.00021	0.0003	1.232	-0.1593
8	$70 \times 70(8)$	2×10^{-7}	0.003	0.00021	0.0002	-1.455	-0.0005
9	$70 \times 70(9)$	1×10^{-8}	0.005	0.00021	0.0002	-1.232	-0.0675
10	$70 \times 70(10)$	1×10^{-7}	0.004	0.00021	0.0002	-1.391	-0.0895

Table 5.8: Random samples for random mesh generation of size 80×80

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$80 \times 80(1)$	1×10^{-7}	0.0025	0.00016	0.00012	-1.4879	0.0203
2	$80 \times 80(2)$	2×10^{-8}	0.0031	0.00016	0.00016	1.1943	0.0747
3	$80 \times 80(3)$	1×10^{-8}	0.0021	0.00016	0.00011	-0.1957	-0.0607
4	$80 \times 80(4)$	2×10^{-8}	0.0031	0.00016	0.00016	0.0852	-0.0557
5	$80 \times 80(5)$	1×10^{-7}	0.0019	0.00016	0.00011	1.0831	-0.0668
6	$80 \times 80(6)$	1×10^{-7}	0.0029	0.00016	0.00011	-1.323	-0.1135
7	$80 \times 80(7)$	2×10^{-7}	0.0022	0.00016	0.00011	-1.3241	-0.1059
8	$80 \times 80(8)$	1×10^{-8}	0.0023	0.00016	0.00014	-1.4485	0.1048
9	$80 \times 80(9)$	1×10^{-7}	0.0026	0.00016	0.00012	1.0242	0.0979
10	$80 \times 80(10)$	1×10^{-7}	0.0022	0.00016	0.00012	1.3125	-0.1309

Table 5.9: Random samples for random mesh generation of size 90×90

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$90 \times 90(1)$	1×10^{-9}	0.0038	0.000123	0.000133	-1.353	0.038
2	$90 \times 90(2)$	2×10^{-9}	0.0033	0.000123	0.000124	-1.405	0.032
3	$90 \times 90(3)$	1×10^{-8}	0.0035	0.000123	0.000129	-1.314	0.153
4	$90 \times 90(4)$	2×10^{-8}	0.0055	0.000123	0.000156	0.995	0.221
5	$90 \times 90(5)$	1×10^{-7}	0.0023	0.000123	0.000107	1.361	-0.002
6	$90 \times 90(6)$	1×10^{-7}	0.0030	0.000123	0.000099	1.391	-0.096
7	$90 \times 90(7)$	2×10^{-7}	0.0026	0.000123	0.000121	1.437	0.108
8	$90 \times 90(8)$	1×10^{-8}	0.0031	0.000123	0.000117	-1.259	0.037
9	$90 \times 90(9)$	1×10^{-7}	0.0023	0.000123	0.000094	1.218	-0.047
10	$90 \times 90(10)$	1×10^{-7}	0.0033	0.000123	0.000127	-1.352	0.063

Table 5.10: Random samples for random mesh generation of size 100×100

S. No	Sample (Realization)	Min cell size	Max cell size	Average cell size	Standard deviation of cells	Skewness of cells	Correlation in cells
1	$100 \times 100(1)$	1×10^{-9}	0.002	0.0001	0.000098	-1.383	0.117
2	$100 \times 100(2)$	2×10^{-9}	0.002	0.0001	0.000096	-1.337	-0.089
3	$100 \times 100(3)$	1×10^{-8}	0.003	0.0001	0.000093	1.317	0.045
4	$100 \times 100(4)$	2×10^{-8}	0.008	0.0001	0.000087	0.525	-0.092
5	$100 \times 100(5)$	1×10^{-7}	0.003	0.0001	0.000109	1.259	-0.045
6	$100 \times 100(6)$	1×10^{-7}	0.004	0.0001	0.000099	1.132	-0.033
7	$100 \times 100(7)$	2×10^{-7}	0.003	0.0001	0.000108	1.175	0.052
8	$100 \times 100(8)$	1×10^{-8}	0.003	0.0001	0.000111	1.463	0.043
9	$100 \times 100(9)$	1×10^{-9}	0.002	0.0001	0.000087	-0.793	0.086
10	$100 \times 100(10)$	1×10^{-9}	0.002	0.0001	0.000087	1.368	-0.013

For uniform meshes, the numerical solution can be implemented using Eq. 3.3. Still, the situation changes for random meshes due to instantaneous change in the step size h and k, as shown in Figure 3.3. Here three different approaches can be used to handle the unequal step size. Consider the shaded region in Figure 3.3 contains an interior node $u_{i,j}$ where the solution is required to compute surrounded by different values of h (h_1 and h_2) and k (k_1 and k_2). h and k can be chosen as minimum, maximum or average of (h_1 and h_2) and (k_1 and k_2) respectively. It was roughly tested that among $h_{min} =$ min(h_1 and h_2), $h_{max} =$ max(h_1 and h_2)and h_{avg} =avg (h_1 and h_2) and $k_{min} =$ min(k_1 and k_2), $k_{max} =$ max(k_1 and k_2)and k_{avg} =avg (k_1 and k_2). The h_{max} and k_{max} seems to be a good selection for the converged solution. Different random meshes based on the data given in Tables 5.1-5.10 are constructed by writing a MATLAB code and are shown in Figure 5.1., In this Figure, six random samples are presented carried out from Table 5.1-5.10.

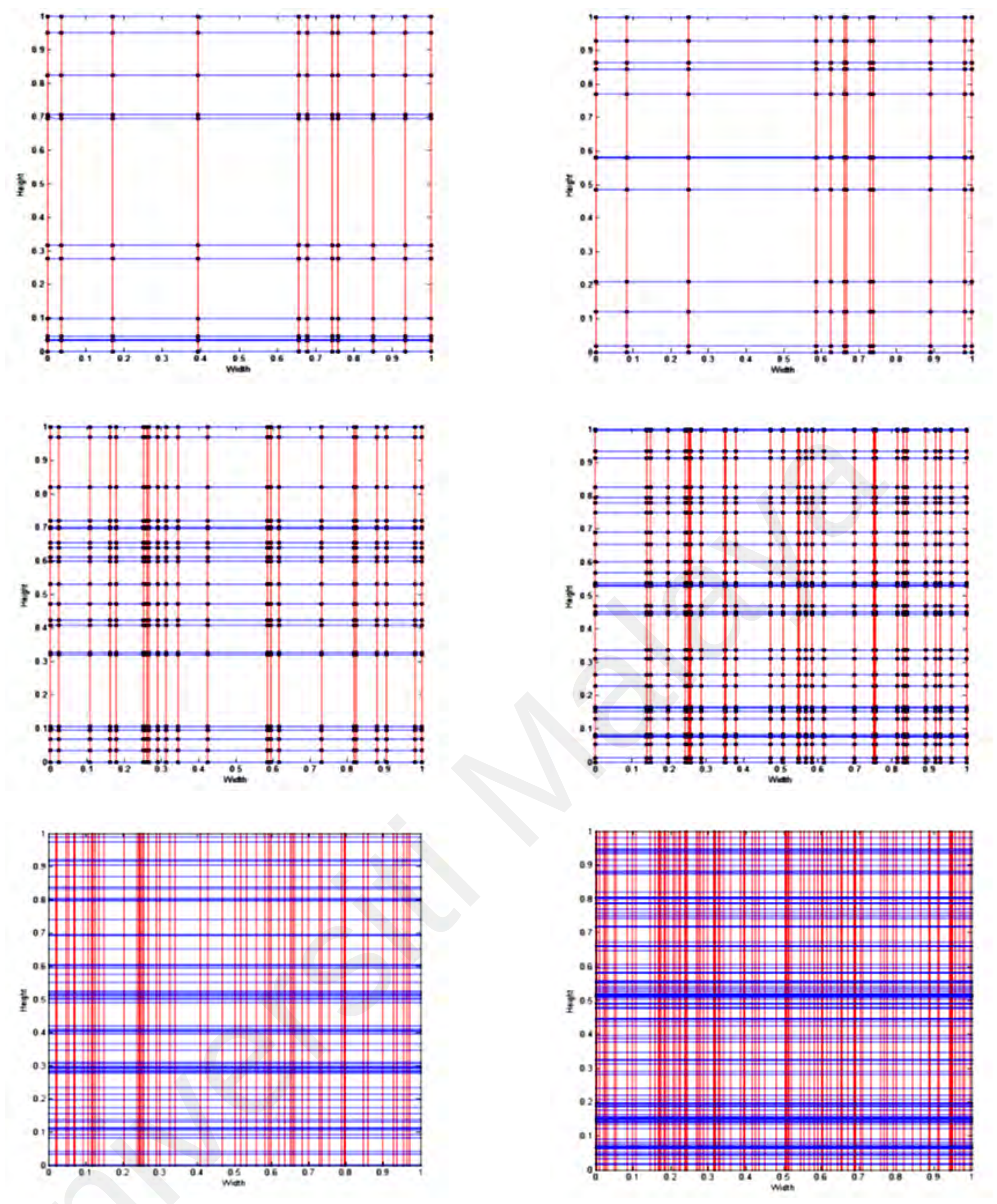


Figure 5.1: Random mesh of size realizations from 10×10 to 100×100

Various randomly generated grids are generated with the help of MATLAB code. The above random mesh size realizations were taken from Table 5.1 to Table 5.10. Although the Tabular presentations are already presented, graphical representations are offered in different realizations.

5.3 Numerical solution profiles over uniform meshes

The numerical profile solutions are obtained using random meshes. Figure 5.2 shows the numerical simulation of steady-state temperature in the unit square over randomly generated meshes of size $10 \times 10, 20 \times 20, \ldots, 100 \times 100$, respectively. Each figure exhibits a local solution profile where the numerical solution values vary from 25°Cto 100°C. In some of the figures, the solution profiles are not smooth compared to the solution on uniform meshes. However, for a specific grid size out of 10 realizations, the many are approximately the same as uniform meshes at some percentage error. In few samples, there is an error in solution profiles because of the standard deviation in the cell size of random meshes. The higher the standard deviation higher the error in the solution.

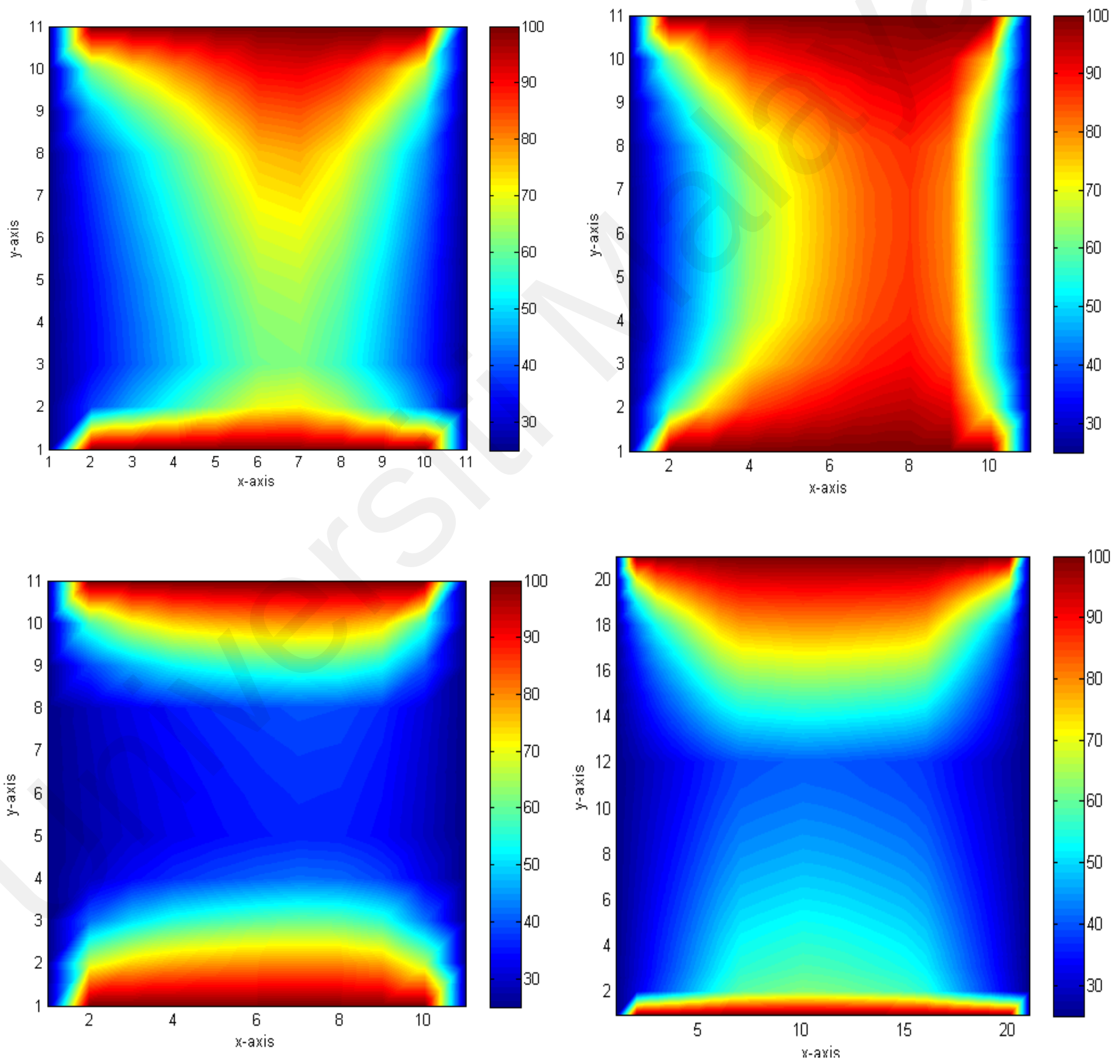


Figure 5.2: Local Solution profile on Random mesh

5.4 Summary

In this chapter, random grids are generated and shown in the form of figures. The motivation of procure these simulation profiles is to use as benchmarking for the simulation profiles prevails on random meshes. The next chapter will be elaborated for random meshes and their simulation profiles obtained from the implementation of the methodology where the concealed ideas related to the proposed research are expected to be discovered and justified. The further discussion of these profiles will be addressed more systematically in Next chapter, where the statistical and converging analysis uniform meshes Versus Random grids shown through tables and figures.

CHAPTER 6:

STATISTICAL PARAMETERS ON CONVERGING ITERATIONS (UNIFORM VS RANDOM) GRIDS

6.1 Introduction

Random samples are highlighted in this chapter. The effect of statistical parameters like, grid size, minimum cell size, maximum cell size, average cell size, standard deviation, skewness, and correlation between cell sizes are analyzed. For each mesh size, the minimum iterations are compared with the iterations of uniform mesh and highlighted. The relationship between the converging iterations over uniform and random meshes predicted the minimum iterations for a given uniform mesh size. Thus, the relationships between the uniform mesh iterations and the random meshes are established for each sample by regression fitting (used polynomial interpolation method). In last, the goodness of fit is presented.

6.2 Statistical and convergence analysis iterations (uniform vs. random meshes)

The improvement in the convergence and number of iterations are noted for each random sample. The effect of statistical parameters likes grid size, minimum cell size, maximum cell size, average cell size, standard deviation, skewness, and correlation between cell sizes are analyzed. For each mesh size, the minimum iteration is compared with the uniform mesh iterations and highlighted in the following Tables 6.1-6.10. It can be observed that for each random sample of meshes, the number of converging iterations is varied, while the iterations in uniform mesh remain the same for each trial because their step size is fixed. The number of iterations in every sample of mesh is analyzed, and found that most of the examples are converging faster than uniform meshes. It means roughly 90% of samples may converge faster than the uniform meshes. However, the accuracy of the solution with fast convergence may not be guaranteed due to the randomness. The scrutinizing also revealed that one with a large maximum cell size and a small standard deviation of cell size has fewer converging iterations among all the random mesh samples.

The effect of other statistical parameters on the converging iterations is put on view through Figures 6.1-6.10.

Table 6.1: Statistical and convergence analysis of 10×10 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$10 \times 10(1)$	**120**	**124**
$10 \times 10(2)$	**119**	**124**
$10 \times 10(3)$	**115**	**124**
$10 \times 10(4)$	**121**	**124**
$10 \times 10(5)$	**118**	**124**
$10 \times 10(6)$	**114**	**124**

Table 6.2: Statistical and convergence analysis of 20×20 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$20 \times 20(1)$	**389**	**390**
$20 \times 20(2)$	**382**	**390**
$20 \times 20(3)$	**390**	**390**
$20 \times 20(4)$	**364**	**390**

Table 6.3: Statistical and convergence analysis of 30×30 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$30 \times 30(4)$	**670**	**732**
$30 \times 30(5)$	**646**	**732**
$30 \times 30(7)$	**728**	**732**
$30 \times 30(8)$	**718**	**732**
$30 \times 30(9)$	**727**	**732**

Table 6.4: Statistical and convergence analysis of 40×40 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$40 \times 40(1)$	**1098**	**1116**
$40 \times 40(3)$	**1051**	**1116**
$40 \times 40(5)$	**1103**	**1116**
$40 \times 40(8)$	**987**	**1116**

Table 6.5: Statistical and convergence analysis of 50×50 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$50 \times 50(1)$	**1502**	**1518**
$50 \times 50(3)$	**1503**	**1518**
$50 \times 50(9)$	**1506**	**1518**

Table 6.6 : Statistical and convergence analysis of 60×60 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$60 \times 60(3)$	**1836**	**1920**
$60 \times 60(8)$	**1790**	**1920**

Table 6.7 : Statistical and convergence analysis of 70×70 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$70 \times 70(1)$	**2198**	**2306**
$70 \times 70(2)$	**2042**	**2306**
$70 \times 70(4)$	**2017**	**2306**
$70 \times 70(10)$	**2067**	**2306**

Table 6.8: Statistical and convergence analysis of 80×80 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$80 \times 80(1)$	**1979**	**2666**
$80 \times 80(2)$	**1923**	**2666**
$80 \times 80(7)$	**2414**	**2666**
$80 \times 80(8)$	**2623**	**2666**
$80 \times 80(9)$	**2515**	**2666**

Table 6.9: Statistical and convergence analysis of 90×90 random and uniform meshes

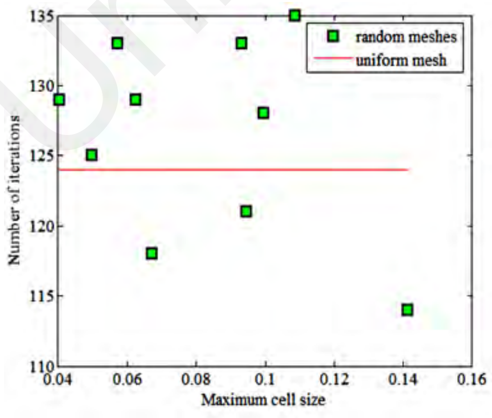
Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$90 \times 90(6)$	**2249**	**2984**

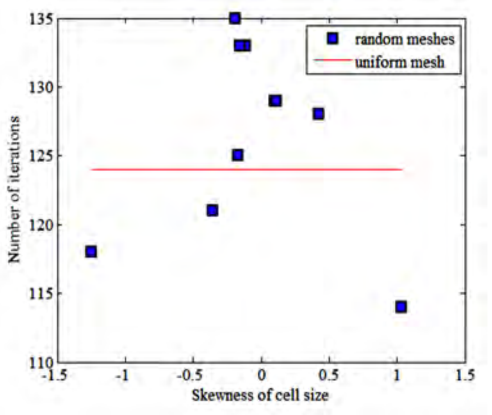
Table 6.10: Statistical and convergence analysis of 100×100 random and uniform meshes

Mesh Size	Iterations (Random mesh)	Iterations (Uniform mesh)
$100 \times 100(1)$	**3185**	**3250**
$100 \times 100(4)$	**3207**	**3250**
$100 \times 100(7)$	**2988**	**3250**
$100 \times 100(10)$	**2833**	**3250**

6.3 Effect of the statistical parameters on the converging iterations (Uniform Vs. Random meshes)

The data shown in the previous section is further analysed for random and uniform meshes simultaneously concerning the maximum cell size, the standard deviation of cell size, skewness of cell size, and cell size correlation are exhibited in Figure 6.1. These are forty Figures divided into a set of four Figures depending upon the mesh size. For example, Figure 6.1 analyzes the effect of the above-mentioned statistical parameters for the mesh samples of size for two-dimensional PDEs. The number of samples with the best results is scrutinized by graphical representation. About 6 to 9 samples of random meshes provide feasible results. It isn't easy to establish a deterministic relationship between the statistical parameters and the converging iterations. If we compare each set of four-figure, then it appears that there is no similar pattern for converging iterations under the effect of statistical parameters. The reduced iterations than uniform mesh vary chaotically. It shows that randomness does not imply any relation. A deeper insight was put into the figures. It was found that somehow the maximum cell size and standard deviation are related for the minimum number of iterations. At the same time, skewness and correlation are not more sensitive for the converging iterations as they have chaotic behavior. While the relation between the random converging iterations and uniform converging iterations can be established, that can be useful to predict the random converging iterations concerning the uniform converging iterations for given mesh size regardless of statistical parameters. The construction of such relationships can be adopted by the idea of the polynomial that will also be demonstrated.





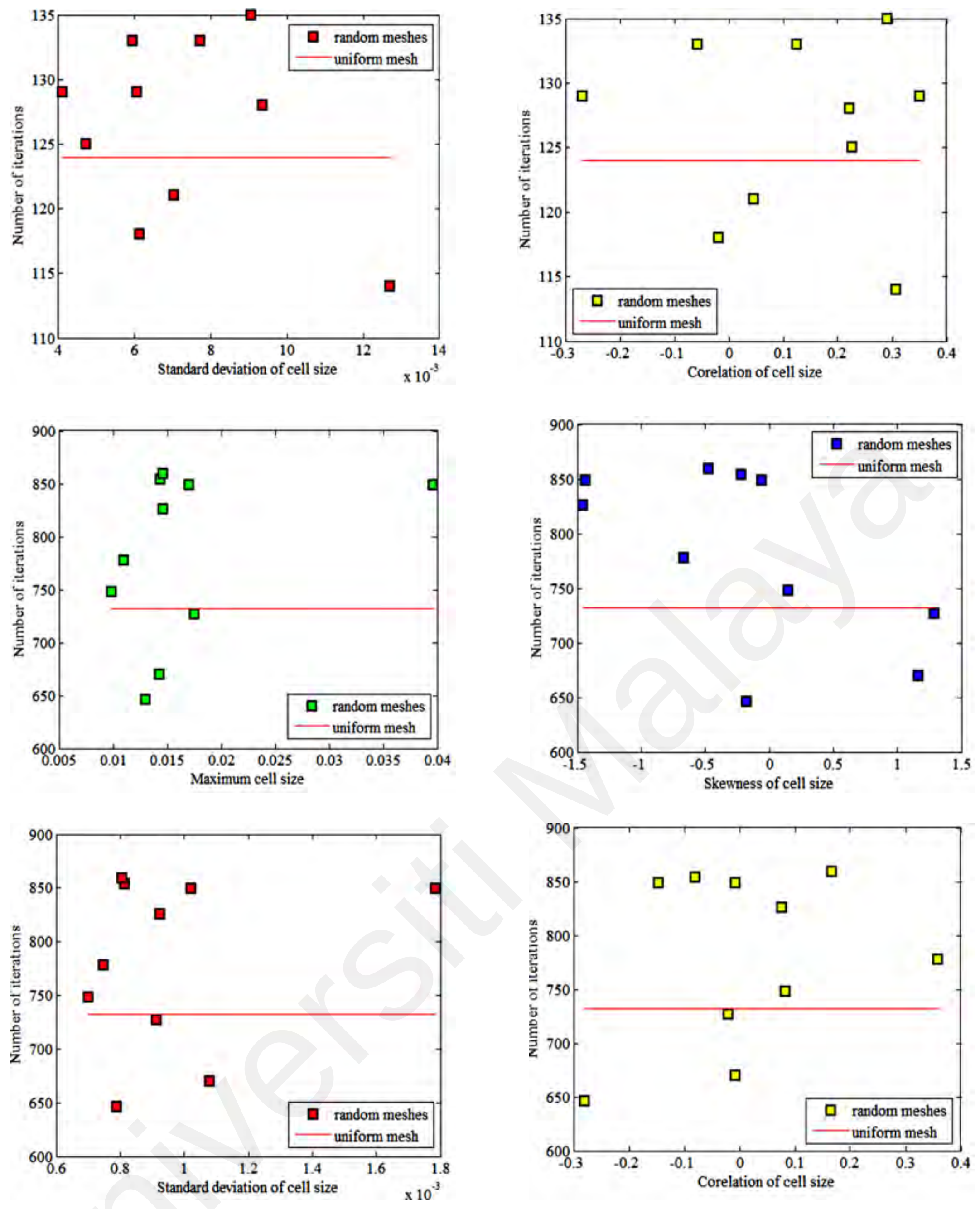


Figure 6.1: Effect of Statistical parameters (Maximum cell size, Skewness, Standard deviation, and correlation analysis) on converging iterations for cell sizes

6.4 Regression fitting for converging iteration and test of statistical significance (Uniform Vs Random meshes)

The relationship between converging iterations over uniform and random meshes can help predict the minimum iterations for a given uniform mesh size. Thus, the relationships between the uniform mesh iterations and the random meshes are established for each sample by regression fitting (used polynomial interpolation method). The regression equation is an 8th-degree polynomial whose parameters lie within the 95% confidence interval. Also, the goodness of fit is presented.

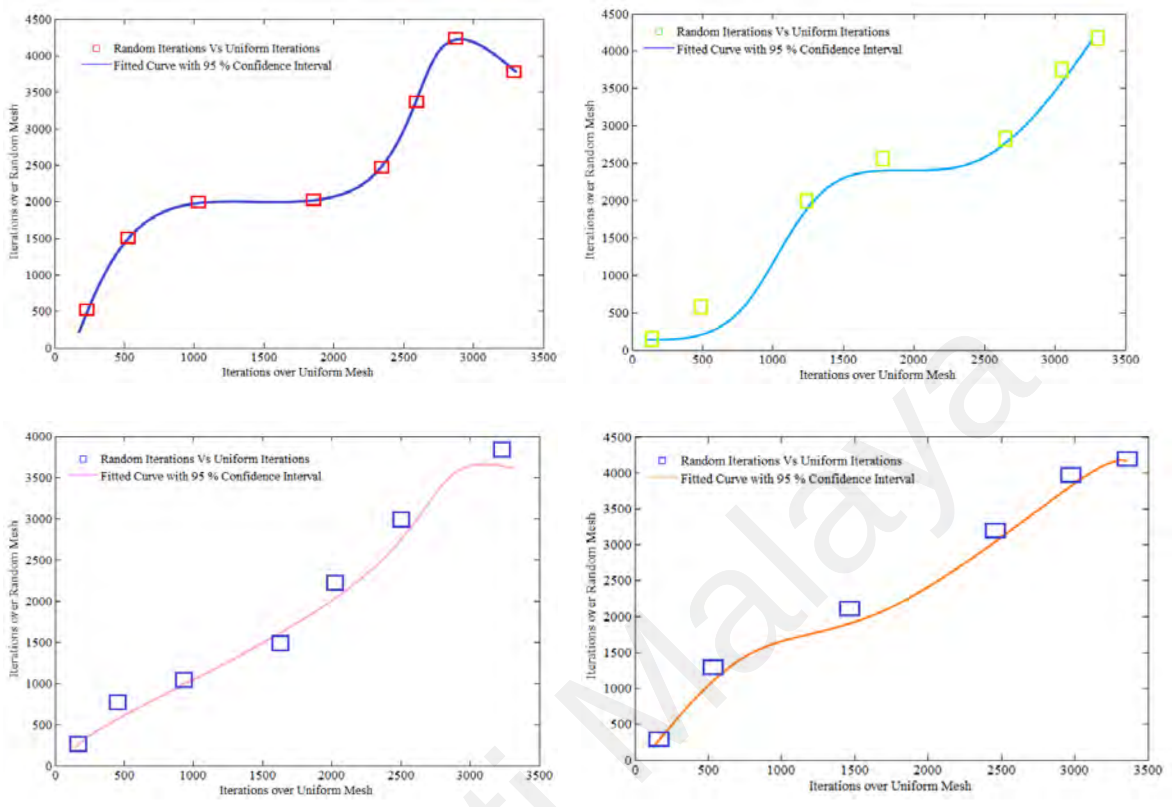


Figure 6.2: Regression fit for realisation of each cell size (Random Iteration Vs. Uniform iterations)

Regression Parameters and Goodness of fit

$$f_1(x) = q_1x^8 + qx^7 + q_3x^6 + q_4x^5 + q_5x^4 + q_6x^3 + q_7x^2 + q_8x^1 + q_9$$

Constant Coefficients (95%- Confidence bounds):

q_1 = −4.9e-022 (-1.826e-021, 8.365e-022), q_2 = 5.8e-018 (-1.209e-017, 2.381e-017)
q_3 = −2.8e-014 (-1.277e-013, 7.171e-014), q_4 = 6.9e-011 (-2.253e-010, 3.639e-010)
q_5 = −9.5e-008 (-5.933e-007, 4.027e-007), q_6 = 7.2e-005 (-0.0004092, 0.0005548)
q_7 = −0.061 (-0.2814, 0.2222), q_8 = 6.8 (-54.31, 68.03),
q_9 =−386.12 (-5103, 4331)

Goodness of fit:
SSE (Sum of Square Error): 2392
Square of Coefficient of Correlation (R-square): 0.9989
RMSE (root mean square error): 48.89

6.5 SUMMARY

In this chapter, a significant amount of results was analyzed and discussed. The results were tried to interpret with the help of comparative tables and figures. Based on the assumptions of using random meshes, many interesting facts about the random meshes are found. The next chapter will be discussed for pointwise comparisons and computational time and percentage error discussed.

CHAPTER 7:

BENCHMARK COMPARISONS

7.1 Introduction

The solution's numerical accuracy over random meshes and the point-wise comparison of the solution with uniform mesh are presented in Figures 7.1-7.10. It is observed that most of the samples are pretty near to the solution of uniform meshes while others deviate due to standard deviation in the cell size. So far, the pointwise comparison of the numerical solution and the numerical simulation profiles of random meshes are found. Still, it is a question of how to determine their accuracy compared to uniform meshes. Furthermore, investigations are carried by calculating the percentage error and percentage reductions for each sample. It can quickly be decided that how much a random sample is near the uniform sample. For example, suppose if the percentage error in the solution of any random sample of mesh is 10%, then we can say that the 90% solution is near to the solution of uniform mesh. Another critical aspect of dealing with random samples is the computational time (sec) required to find the numerical solution. The next section addresses the computational time and numerical accuracy of the random meshes about the converging iterations.

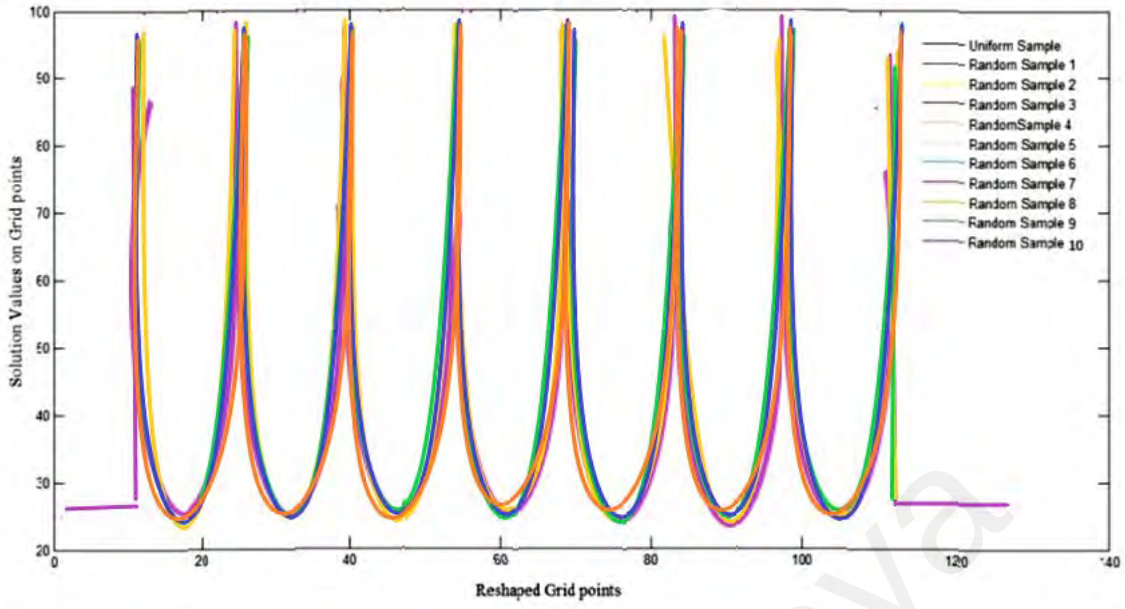


Figure 7.1: Pointwise comparison for the 2D PDEs mesh size 10×10

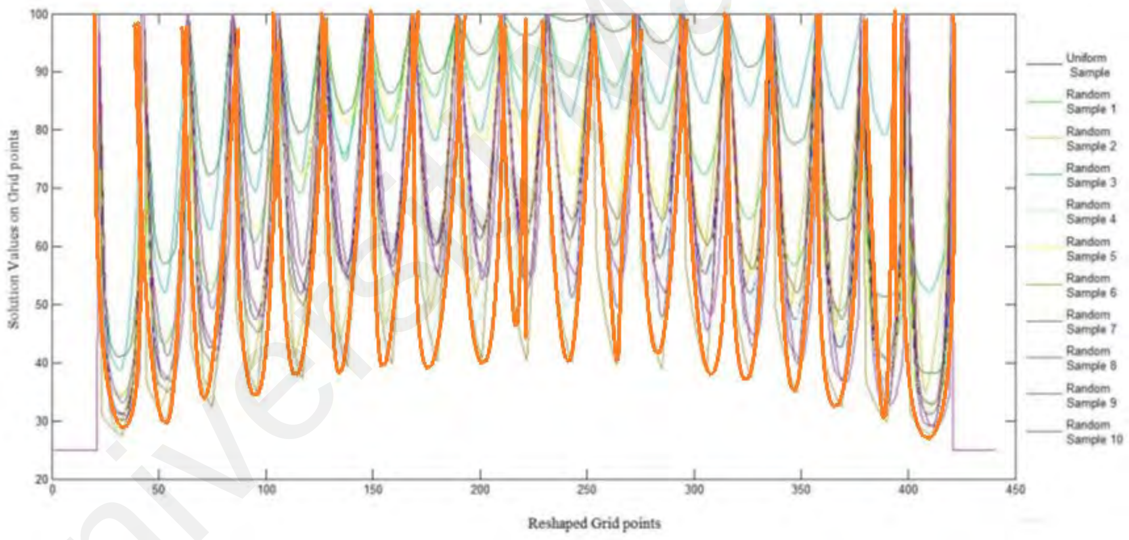


Figure 7.2: Pointwise comparison for the 2D PDEs mesh size 20×20

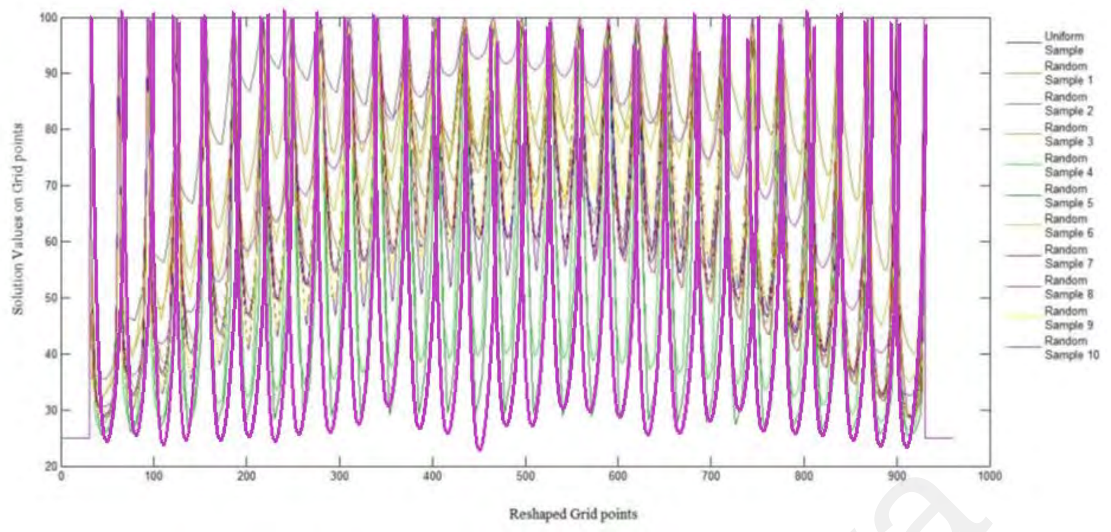


Figure 7.3: Pointwise comparison for the 2D PDEs mesh size 30×30

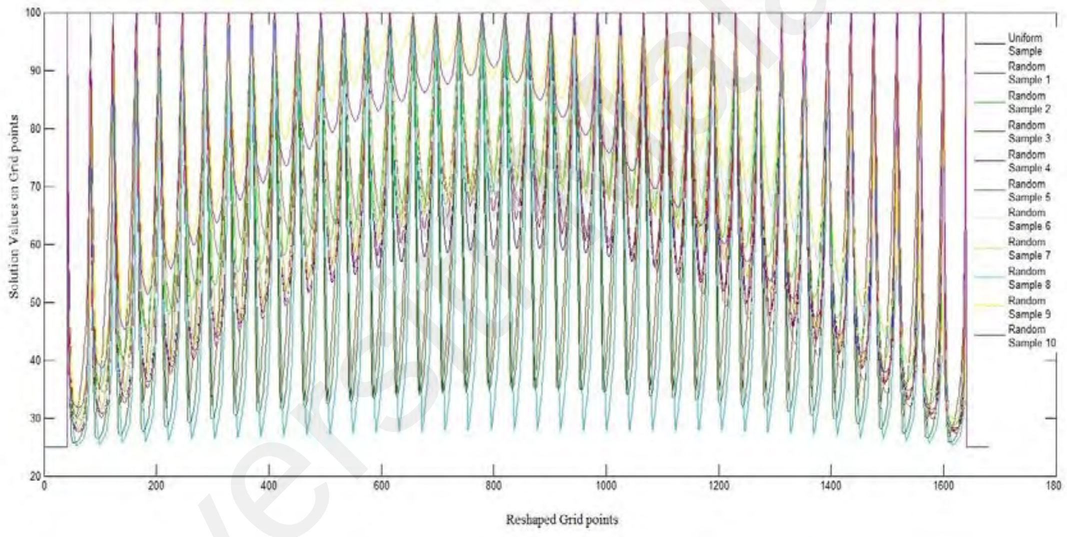


Figure 7.4: Pointwise comparison for the 2D PDEs mesh size 40×40

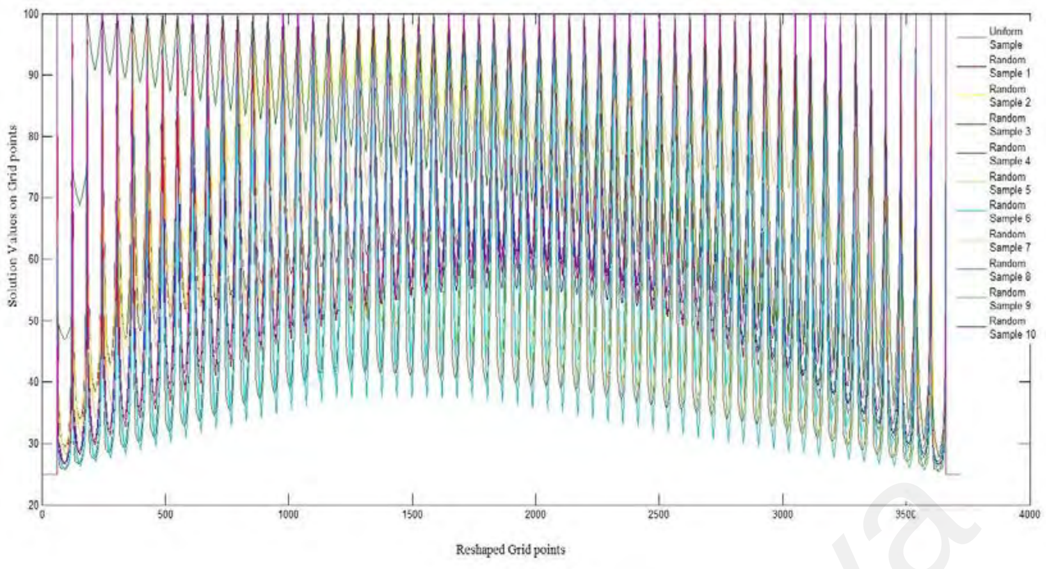


Figure 7.5: Pointwise comparison for the 2D PDEs mesh size 50×50

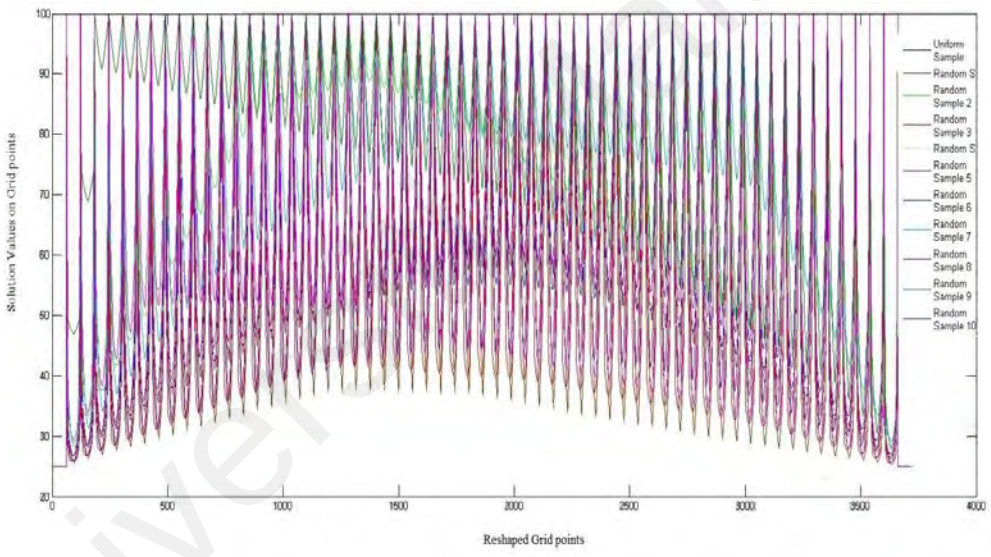


Figure 7.6: Pointwise comparison for the 2D PDEs mesh size 60×60

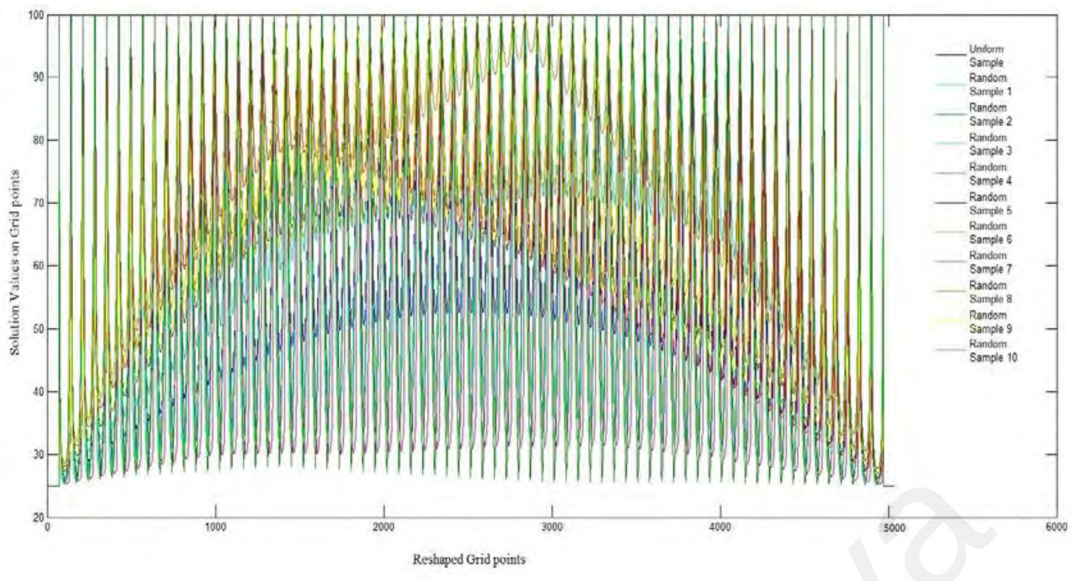


Figure 7.7: Pointwise comparison for the 2D PDEs mesh size 70×70

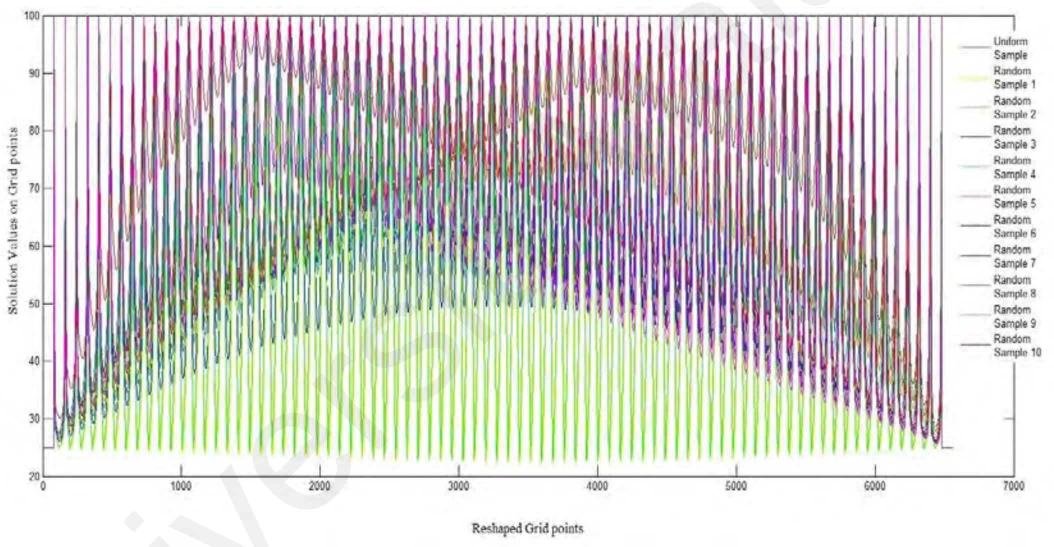


Figure 7.8: Pointwise comparison for the 2D PDEs mesh size 80×80

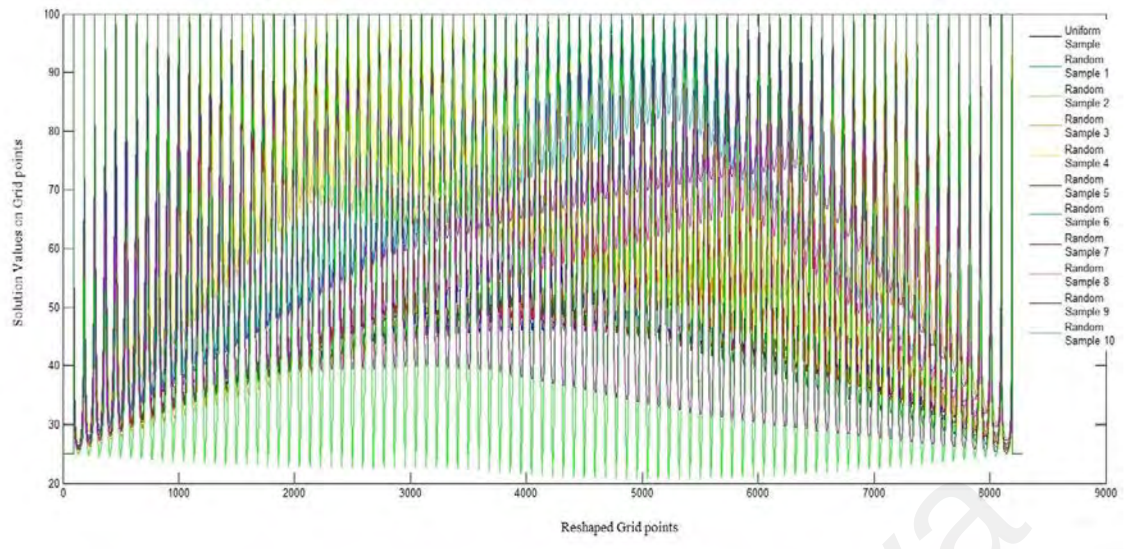


Figure 7.9: Pointwise comparison for the 2D PDEs mesh size 90×90

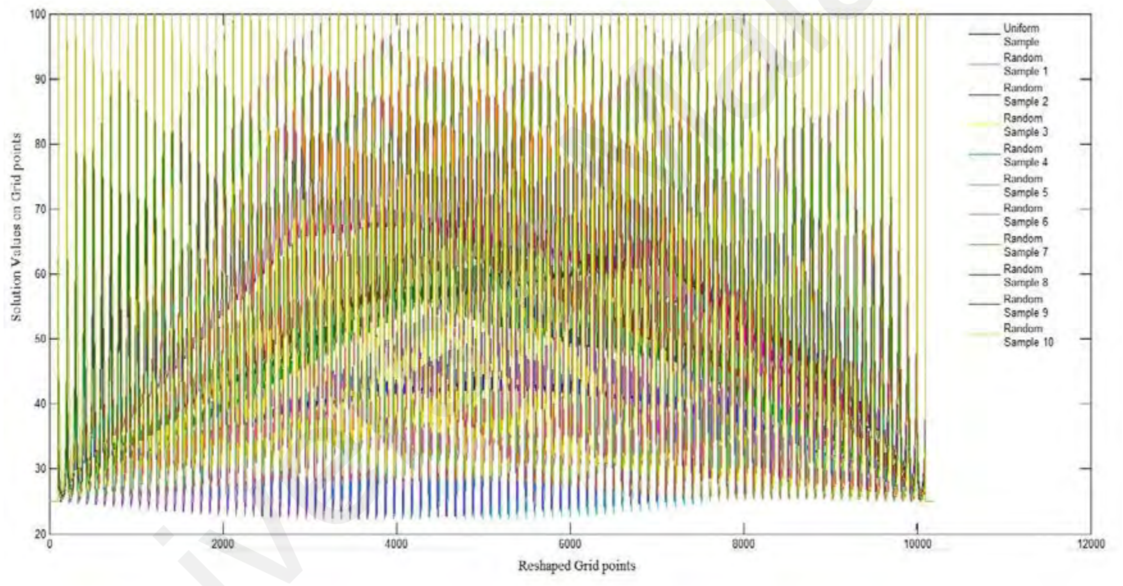


Figure 7.10: Pointwise comparison for the 2D PDEs mesh size 100×100

In this pointwise comparison, we use randomly generated grids Vs., uniform grids. The results are shown in the images on each cell size and number of random realizations. Thus, we can say that The Randomly generated grids found better and fast in convergence results than uniform grids.

7.2 Computational time and the percentage accuracy for numerical solution using uniform and random meshes

Finally, the comparative summary of the vital output parameters, namely the converging iterations, computational time, and percentage error in the numerical solution over random meshes, is presented in Tables 7.1-7.10. The improvement in the number of iterations in the random meshes is highlighted. From the scrutinizing of the Tables, it can be deduced that it is not always the case that less number of iterations will imply less computational time and less percentage error. However, at some places by chance, few specific random samples have good results in minimum iteration in minimum time with minimum error. The overall analysis reveals that there is a chance that up to 8 samples of random meshes may be found better than uniform meshes, and about 87% to 97% of numerical solution profiles on random samples may be near to that of uniform meshes.

Table 7.1: Iteration and CPU Time (In second) wise compairisons (Cell Size 10×10)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	$10 \times 10(1)$	124	128	0.0179	0.0340	7.4895
2	$10 \times 10(2)$	124	133	0.0179	0.0350	10.4352
3	$10 \times 10(3)$	124	125	0.0179	0.0291	8.7687
4	$10 \times 10(4)$	124	133	0.0179	0.032	5.8561
5	$10 \times 10(5)$	124	129	0.0179	0.0241	6.5274
6	$10 \times 10(6)$	124	135	0.0179	0.0421	7.3620
7	$10 \times 10(7)$	124	121	0.0179	0.0291	6.690
8	$10 \times 10(8)$	124	129	0.0179	0.0281	10.185
9	$10 \times 10(9)$	124	118	0.0179	0.0306	13.8284
10	$10 \times 10(10)$	124	114	0.0179	0.0215	10.0673

Table 7.2: Iteration and CPU Time (In second) wise compairisons (Cell Size 20×20)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	$20 \times 20(1)$	390	431	0.1245	0.1207	13.3565
2	$20 \times 20(2)$	390	398	0.1245	0.1723	6.3700
3	$20 \times 20(3)$	390	456	0.1245	0.1212	21.7264
4	$20 \times 20(4)$	390	382	0.1245	0.0870	6.9483
5	$20x20(5)$	390	427	0.1245	0.0451	9.2699
6	$20 \times 20(6)$	390	390	0.1245	0.0951	15.5964
7	$20 \times 20(7)$	390	405	0.1245	0.0981	5.3241
8	$20 \times 20(8)$	390	464	0.1245	0.1012	23.1439
9	$20 \times 20(9)$	390	416	0.1245	0.1109	3.4983
10	$20 \times 20(10)$	390	388	0.1245	0.1020	8.4170

Table 7.3: Iteration and CPU Time (In second) wise compairisons (Cell Size 30×30)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	$30 \times 30(1)$	732	854	0.3155	0.3267	13.8935
2	$30 \times 30\ (2)$	732	859	0.3155	0.3514	14.1853
3	$30 \times 30\ (3)$	732	826	0.3155	0.3523	10.3204
4	$30 \times 30\ (4)$	732	670	0.3155	0.2614	13.0784
5	$30 \times 30\ (5)$	732	646	0.3155	0.2632	19.8685
6	$30 \times 30\ (6)$	732	849	0.3155	0.2392	14.2445
7	$30 \times 30\ (7)$	732	748	0.3155	0.2910	2.5416
8	$30 \times 30\ (8)$	732	778	0.3155	0.3997	3.3646
9	$30 \times 30\ (9)$	732	727	0.3155	0.1910	5.4423
10	$30 \times 30\ (10)$	732	849	0.3155	0.4013	14.9839

Table 7.4: Iteration and CPU Time (In second) wise compairisons (Cell Size 40×40)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	$40 \times 40\ (1)$	1116	1098	0.6095	0.9666	12.7467
2	$40 \times 40\ (2)$	1116	1212	0.6095	0.8796	7.7098
3	$40 \times 40\ (3)$	1116	1051	0.6095	0.5013	15.4096
4	$40 \times 40\ (4)$	1116	1249	0.6095	0.6187	3.4520
5	$40 \times 40\ (5)$	1116	1183	0.6095	0.6031	6.6157
6	$40 \times 40\ (6)$	1116	1274	0.6095	0.7943	7.2821
7	$40 \times 40\ (7)$	1116	1214	0.6095	0.2991	7.8928
8	$40 \times 40\ (8)$	1116	987	0.6095	0.5813	18.4328
9	$40 \times 40\ (9)$	1116	1373	0.6095	0.8901	20.4144
10	$40 \times 40\ (10)$	1116	1334	0.6095	0.8814	13.9951

Table 7.5: Iteration and CPU Time (In second) wise compairisons (Cell Size 50×50)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	50 × 50 (1)	1518	1502	1.1354	1.3201	8.2579
2	50 × 50 (2)	1518	1708	1.1354	1.6913	18.0615
3	50 × 50 (3)	1518	1603	1.1354	1.5820	5.9155
4	50 × 50 (4)	1518	1509	1.1354	1.4710	7.2018
5	50 × 50 (5)	1518	1532	1.1354	1.2017	4.1911
6	50 × 50 (6)	1518	1782	1.1354	1.4200	9.8148
7	50 × 50 (7)	1518	1771	1.1354	1.3401	11.0597
8	50 × 50 (8)	1518	1836	1.1354	1.4235	13.8783
9	50 × 50 (9)	1518	1606	1.1354	1.4792	13.6575
10	50 × 50 (10)	1518	1857	1.1354	1.4129	13.6323

Table 7.6: Iteration and CPU Time (In second) wise compairisons (Cell Size 60×60)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	60 × 60 (1)	1920	2086	1.8222	1.9013	3.0321
2	60 × 60 (2)	1920	2436	1.8222	2.3241	16.3994
3	60 × 60 (3)	1920	1836	1.8222	2.912	15.0607
4	60 × 60 (4)	1920	2095	1.8222	2.3281	5.8969
5	60 × 60 (5)	1920	2201	1.8222	2.0980	11.6052
6	60 × 60 (6)	1920	1933	1.8222	1.6512	9.1422
7	60 × 60 (7)	1920	2232	1.8222	2.3127	18.9657
8	60 × 60 (8)	1920	1709	1.8222	1.4271	13.5796
9	60 × 60 (9)	1920	2910	1.8222	2.1432	23.2238
10	60 × 60 (10)	1920	2105	1.8222	1.9123	5.2158

Table 7.7: Iteration and CPU Time (In second) wise compairisons (Cell Size 70×70)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	70 × 70 (1)	2306	2198	3.5449	3.1188	8.9361
2	70 × 70 (2)	2306	2042	3.5449	4.6498	12.8594
3	70 × 70 (3)	2306	2731	3.5449	4.0904	13.8529
4	70 × 70 (4)	2306	2017	3.5449	2.9217	17.1358
5	70 × 70 (5)	2306	2706	3.5449	3.1599	9.8384
6	70 × 70 (6)	2306	3221	3.5449	4.5271	21.4260
7	70 × 70 (7)	2306	2902	3.5449	4.0271	11.7434
8	70 × 70 (8)	2306	2702	3.5449	3.9390	14.0924
9	70 × 70 (9)	2306	2759	3.5449	4.1275	11.8808
10	70 × 70 (10)	2306	2067	3.5449	3.0952	11.4737

Table 7.8: Iteration and CPU Time (In second) wise compairisons (Cell Size 80×80)

S.No	Size	Iteration		Time		Percentag Error
		Uniform	Random	Uniform	Random	
1	80 × 80 (1)	2666	1979	4.4224	2.7043	11.6885
2	80 × 80 (2)	2666	1923	4.4224	2.6458	12.1357
3	80 × 80 (3)	2666	3075	4.4224	4.1943	22.6994
4	80 × 80 (4)	2666	2932	4.4224	4.1415	14.8689
5	80 × 80 (5)	2666	2879	4.4224	3.8355	7.5659
6	80 × 80 (6)	2666	3368	4.4224	5.0268	20.8415
7	80 × 80 (7)	2666	2414	4.4224	3.2932	11.0635
8	80 × 80 (8)	2666	2623	4.4224	4.0040	15.4069
9	80 × 80 (9)	2666	2515	4.4224	2.7040	8.9071
10	80 × 80 (10)	2666	2847	4.4224	3.0044	12.1675

Table 7.9: Iteration and CPU Time (In second) wise compairisons (Cell Size 90×90)

S.No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	90 × 90 (1)	2984	4000	6.5926	9.9073	16.9081
2	90 × 90 (2)	2984	3648	6.5926	7.8027	7.2622
3	90 × 90 (3)	2984	3564	6.5926	7.9823	11.3747
4	90 × 90 (4)	2984	4332	6.5926	10.0660	15.3732
5	90 × 90 (5)	2984	3201	6.5926	6.8601	3.0260
6	90 × 90 (6)	2984	2249	6.5926	5.0736	10.8850
7	90 × 90 (7)	2984	3016	6.5926	6.3257	9.4133
8	90 × 90 (8)	2984	3431	6.5926	7.6009	8.2304
9	90 × 90 (9)	2984	4041	6.5926	8.6947	12.7733
10	90 × 90 (10)	2984	3025	6.5926	6.6446	3.4961

Table 7.10: Iteration and CPU Time (In second) wise compairisons (Cell Size 100×100)

S. No	Size	Iteration		Time		Percentage Error
		Uniform	Random	Uniform	Random	
1	100 × 100 (1)	3250	3185	7.6443	8.9009	9.7572
2	100 × 100 (2)	3250	3889	7.6443	9.7716	15.9726
3	100 × 100 (3)	3250	3456	7.6443	8.9317	13.7269
4	100 × 100 (4)	3250	3207	7.6443	7.2889	9.5222
5	100 × 100 (5)	3250	3228	7.6443	8.4705	14.1568
6	100 × 100 (6)	3250	4501	7.6443	12.1745	10.8340
7	100 × 100 (7)	3250	2988	7.6443	5.4988	14.5016
8	100 × 100 (8)	3250	4334	7.6443	10.1134	11.4511
9	100 × 100 (9)	3250	3964	7.6443	10.3191	8.3286
10	100 × 100 (10)	3250	2833	7.6443	7.2272	7.9895

7.3 Computational time reduction

The randomly generated grid over uniform grids is analysed, and the key output parameters (i.e., converging iterations, computational time (seconds), and percentage (%) reduction in computational time) are compared in the numerical solution. The improving iterations and the computational time (Uniform versus Randomly generated grids) are presented in Table 7.11. The computational time for the cell sizes could reduce up to 43% from uniformly generated grids to randomly generated grids

Table 7.11: Computational time and number of iterations

S. No	Cell Size	Converging Iteration		Computational time (seconds)		Percentage reductions
		Uniform grids	Random grids	Uniform grids	Random grids	
1	10×10	125	114	0.0189	0.0147	22.22%
2	20×20	382	390	0.1246	0.1102	11.55%
3	30×30	646	732	0.3154	0.2652	15.9%
4	40×40	987	1116	0.6094	0.4401	27.78%
5	50×50	1502	1518	1.1345	1.1314	0.2732%
6	60×60	1790	1920	1.8222	1.0621	41.71%
7	70×70	2017	2306	3.5448	2.0071	43%
8	80×80	1923	2666	4.442	3.0781	30.70%
9	90×90	2249	2984	6.69425	5.0012	25.39%
10	100×100	3185	3250	7.6442	5.2988	30.68%

7.4 Summary

In this chapter, a significant amount of results was analyzed, compared, and discussed. The results were tried to interpret with the help of comparative tables and figures. Based on the assumptions of using random meshes many exciting facts about the random meshes are found. That will enable us to answer the research question highlighted in chapter 1. The details of these outcomes as concluding remarks will be presented in the next chapter.

CHAPTER 8:

CONCLUSION AN FUTURE WORK

8.1 Conclusion

This study considers the numerical solutions of one, two, and three-dimensional differential equations. The numerical solutions and procedure followed for one, two, and three dimensional PDEs, the numerical profile solutions were obtained. In addition, the new method was implemented on the two-dimensional with Dirchlet boundary conditions with special treatment of randomly generated meshes. The numerical solutions used the 100, 10000, and 1000000 samples of random meshes were obtained from one, two, and three dimensions, and results are compared with available grids are uniform or regular grids. Furthermore, the feasibility and the practicality of applying the samples of randomly generated meshes are tested for each random sample's corresponding cell sizes and the statistical parameters. The hypothesis is assumed that the random meshes might improve the convergence of the numerical solution, which theory is accepted by proving the results. However, the statistical analysis helped justify the main objective, which is to test the practicability and feasibility of using the randomly generated finite difference meshes for the numerical solutions. Further, the research question regarding the accuracy, while using S.M's Method briefed in the concluding remarks, leading to justify the research objectives, hypothesis, and the research question:

1) The samples of SM's Method, a new method, from every ten different samples of random meshes using one dimensional PDEs, two dimensional PDEs, and Three dimensional PDEs and Fractional PDEs eight models of random sample provid faster convergence than uniform meshes. However , the only smoothness of the solutions is challenging because smoothness cannot guaranty owing to randomness in the mesh parameters.

2) Smoothnes using the SM’s Method, the numerical solutions are violated due to random behavior and the minimum cell size with high standard deviation in the meshes.
3) The statistical parameters, variance, cell sizes in average, minimum, and maximum are obtained. It is easy to manifest between random and uniform meshes due to statistical parameters. However, the best results were found because random meshes have maximum cell size and slight standard deviation of cells over uniform meshes. In comparison, the correlation and the skewness between the cells were found less vulnerable.
4) The statistical parameters create the relationship between converging iterations in randomly generated samples over uniform meshes.
5) Solutions are compared with the help of pointwise comparisons. The numerical solution taken by random meshes with uniformly generated solutions reveals about ninety-five percent of the samples of random meshes found near the uniformly generated solutions.
6) However, the CPU, Computational time, and percentage error are mostly found less in random samples and have good results by comparing the iterations. The number of iterations is less in number than the available method with minimum error.

Finally, it is concluded that the new method, SM’s Method, which uses randomly generated meshes to solve the one, two, and three partial differential equations using the finite difference method, is recommended and suggested.

8.2 Future work

There are some ideas that I would have liked to recommend for future study. In mathematics, the model can be chosen irregularly shaped bodies. The model chosen is composed of several different materials because the element equations are evaluated. This work can be extended for the supercomputer, where numbers of meshes and boundary conditions are used to solve the models. This study can deal with the different dynamic effects, The nonlinear behavior existing with large deformations and nonlinear materials.

In Engineering fields: An electronics and electrical engineering, the differential equations describing complex circuits containing capacitors, inductors and resistors can be replaced with finite random grids. The most important is Computer simulations of the models are used to estimate voltages and currents in the nodes of the circuits.

In Biology system: Various dynamics of growth rate-dependent and washout, Infection, susceptible, and the change in concentration, nitrate concentration with time for a column inoculated with a dilution of Nitrobacter can be used Finite random grids.

REFERENCES

Abouelregal, A. E., Moustapha, M. V., Nofal, T. A., Rashid, S., & Ahmad, H. (2021). Generalized thermoelasticity based on higher-order memory-dependent derivative with time delay. *Results in Physics, 20,* 103705.

Agarwal, P., Agarwal, R. P., & Ruzhansky, M. (2020). *Special Functions and Analysis of Differential Equations*: CRC Press.

Ahmad, H., Akgül, A., Khan, T. A., Stanimirović, P. S., & Chu, Y.-M. (2020). New perspective on the conventional solutions of the nonlinear time-fractional partial differential equations. *Complexity, 2020*.

Al-Shawba, A. A., Gepreel, K., Abdullah, F., & Azmi, A. (2018). Abundant closed form solutions of the conformable time fractional Sawada-Kotera-Ito equation using (G′/G)-expansion method. *Results in Physics, 9,* 337-343.

Ali, U., Kamal, R., & Mohyud-Din, S. (2012). On nonlinear fractional differential equations. *Int. J. Mod. Math. Sci, 3*(3).

Ali, U., Mastoi, S., Othman, W. A. M., Khater, M. M., & Sohail, M. (2021). Computation of traveling wave solution for nonlinear variable-order fractional model of modified equal width equation. *AIMS Mathematics, 6*(9), 10055-10069.

Ali, U., Sohail, M., & Abdullah, F. A. (2020). An Efficient Numerical Scheme for Variable-Order Fractional Sub-Diffusion Equation. *Symmetry, 12*(9), 1437.

Ali, U., Sohail, M., Usman, M., Abdullah, F. A., Khan, I., & Nisar, K. S. (2020). Fourth-order difference approximation for time-fractional modified sub-diffusion equation. *Symmetry, 12*(5), 691.

Ang, W.-T. (2013). Elastic crack problems, fracture mechanics, equations of elasticity and finite-part integrals. In *Hypersingular Integral Equations in Fracture Analysis* (pp. 1-24).

Bar-Sinai, Y., Hoyer, S., Hickey, J., & Brenner, M. P. (2019). Learning data-driven discretizations for partial differential equations. *Proc Natl Acad Sci U S A, 116*(31), 15344-15349. doi:10.1073/pnas.1814058116

Barman, H. K., Seadawy, A. R., Akbar, M. A., & Baleanu, D. (2020). Competent closed form soliton solutions to the Riemann wave equation and the Novikov-Veselov equation. *Results in Physics, 17*, 103131.

Başhan, A. (2019). A mixed methods approach to Schrödinger equation: Finite difference method and quartic B-spline based differential quadrature method. *An International Journal of Optimization and Control: Theories & Applications (IJOCTA), 9*(2), 223-235.

Bashan, A., Yagmurlu, N. M., Ucar, Y., & Esen, A. (2017). An effective approach to numerical soliton solutions for the Schrödinger equation via modified cubic B-spline differential quadrature method. *Chaos, Solitons & Fractals, 100,* 45-56. doi:https://doi.org/10.1016/j.chaos.2017.04.038

Bin, Z. (2012). (G′/G)-expansion method for solving fractional partial differential equations in the theory of mathematical physics. *Communications in Theoretical Physics, 58*(5), 623.

Buller., S. M. W. A. M. o. A. B. M. N. B. K. A. S. (2020). Numerical solution of Partial differential equations(PDE's) for nonlinear Local Fractional PDE's and Randomly generated grids. *International Journal of Disaster Recovery and Business Continuity, 11*(01), 07. doi:IJDRBC/article/view/27693/15261

. Chapter 5 Preliminary Review of Finite Difference Methods. (1992). In T. Weiyan (Ed.), *Elsevier Oceanography Series* (Vol. 55, pp. 161-205): Elsevier.

Chen, C.-S., Fan, C.-M., & Wen, P. (2012). The method of approximate particular solutions for solving certain partial differential equations. *Numerical Methods for Partial Differential Equations, 28*(2), 506-522.

Craig, V. J. (1901). Isaac Newton. *The American Mathematical Monthly, 8*(8-9), 157-161.

Dinesh., T. A. V. V. S. S. P. M. S. S. (2018). Potential Flow Simulation through Lagrangian

Interpolation Meshless Method Coding. *Journal of Applied Fluid Mechanics, 11*(special issue), 7. doi:10.13140/RG.2.2.17792.66567

Duan, J., & Tang, H. (2020). Entropy stable adaptive moving mesh schemes for 2D and 3D special relativistic hydrodynamics. *Journal of Computational Physics*. doi:10.1016/j.jcp.2020.109949

El-Ajou, A., Al-Smadi, M., Oqielat, M. a. N., Momani, S., & Hadid, S. (2020). Smooth expansion to solve high-order linear conformable fractional PDEs via residual power series method: Applications to physical and engineering equations. *Ain Shams Engineering Journal*, *11*(4), 1243-1254. doi:10.1016/j.asej.2020.03.016

Farlow, S. J. (2006). *An introduction to differential equations and their applications*: Courier Corporation.

Folland, G. B. (2020). *Introduction to partial differential equations*: Princeton university press.

Fraser, D. C. (1927). Newton's Interpolation Formulas. An unpublished Manuscript of Sir Isaac Newton. *Journal of the Institute of Actuaries, 58*(1), 53-95.

Ghosh, U. (2020). Electro-magneto-hydrodynamics of non-linear viscoelastic fluids. *Journal of Non-Newtonian Fluid Mechanics, 277*. doi:10.1016/j.jnnfm.2020.104234

Grossmann, C., Roos, H.-G., & Stynes, M. (2007). *Numerical treatment of partial differential equations* (Vol. 154): Springer.

Gu, Y., Wang, L., Chen, W., Zhang, C., & He, X. (2017). Application of the meshless generalized finite difference method to inverse heat source problems. *International Journal of Heat and Mass Transfer, 108*, 721-729. doi:10.1016/j.ijheatmasstransfer.2016.12.084

Guan, Z., Li, J., Wu, L., Zhang, Y., Wu, J., & Du, X. (2017). Achieving Efficient and Secure Data Acquisition for Cloud-Supported Internet of Things in Smart Grid. *IEEE Internet of Things Journal, 4*(6), 1934-1944. doi:10.1109/jiot.2017.2690522

Habeeb, T., Mokhtar, M. M., Sieda, B., Osman, G., Ibrahim, A., Metwalli, A. M., . . . Mohamed, M. B. (2020). Changing the innate consensus about mesh fixation in trans-abdominal preperitoneal laparoscopic inguinal hernioplasty in adults: Short and long term outcome. Randomized controlled clinical trial. *Int J Surg, 83*, 117-124. doi:10.1016/j.ijsu.2020.09.013

Han, J., Nica, M., & Stinchcombe, A. R. (2020). A derivative-free method for solving elliptic partial differential equations with deep neural networks. *Journal of Computational Physics, 419*. doi:10.1016/j.jcp.2020.109672

Harir, A., Melliani, S., El Harfi, H., & Chadli, L. S. (2020). Variational Iteration Method and Differential Transformation Method for Solving the SEIR Epidemic Model. *International Journal of Differential Equations, 2020*, 1-7. doi:10.1155/2020/3521936

Hosseini, S. M., Kalhori, H., & Al-Jumaily, A. (2020). Active vibration control in human forearm model using paired piezoelectric sensor and actuator. *Journal of Vibration and Control*. doi:10.1177/1077546320957533

Islam, T., Akbar, M. A., & Azad, A. K. (2018). Traveling wave solutions to some nonlinear fractional partial differential equations through the rational (G′/G)-expansion method. *Journal of Ocean Engineering and Science, 3*(1), 76-81. doi:https://doi.org/10.1016/j.joes.2017.12.003

Jawad, A. J. a. M., Petković, M. D., & Biswas, A. (2010). Modified simple equation method for nonlinear evolution equations. *Applied Mathematics and Computation, 217*(2), 869-877.

Kovács, M., Kirchner, K., & Bolin, D. (2020). Numerical solution of fractional elliptic stochastic PDEs with spatial white noise. *IMA Journal of Numerical Analysis, 40*(2), 1051-1073. doi:10.1093/imanum/dry091

Kucharski, A. J., Russell, T. W., Diamond, C., Liu, Y., Edmunds, J., Funk, S., . . . Flasche, S. (2020). Early dynamics of transmission and control of COVID-19: a mathematical modelling study. *The Lancet Infectious Diseases, 20*(5), 553-558. doi:10.1016/s1473-3099(20)30144-4

Kumaresan., S. M. W. A. M. O. N. (2020). Numerical Solutions of Second order fractionalPDE's by using Finite-difference Method over randomly generated grids. *International Journal of Advanced Science and Technology, 29*(11), 09. doi:/IJAST/article/view/19991/10143

Lee, P., & Kim, S. (2020). A variable-θ method for parabolic problems of nonsmooth data. *Computers & Mathematics with Applications, 79*(4), 962-981. doi:10.1016/j.camwa.2019.08.006

Li, Y., Sun, Y., & Crouseilles, N. (2020). Numerical simulations of one laser-plasma model based on Poisson structure. *Journal of Computational Physics, 405*. doi:10.1016/j.jcp.2019.109172

Liu, Z. (2019). Algebraic L2-decay of weak solutions to the magneto-hydrodynamic equations. *Nonlinear Analysis: Real World Applications, 50*, 267-289. doi:10.1016/j.nonrwa.2019.05.001

Mahmood, A., Md Basir, M. F., Ali, U., Mohd Kasihmuddin, M. S., & Mansor, M. (2019). Numerical solutions of heat transfer for magnetohydrodynamic jeffery-hamel flow using spectral Homotopy analysis method. *Processes, 7*(9), 626.

Mastoi, S., Kalhoro, N. B., Mugheri, A. B. M., Rajput, U. A., Mastoi, R. B., Mastoi, N., & Othman, W. A. M. (2021). Numerical solution of Partial differential equation using finite random grids. *International Journal of Advanced Research in Engineering and Technology (IJARET).*

Mastoi, S., Mugheri, A. B., Kalhoro, N. B., & Buller, A. S. (2020). Numerical solutionof Partial differential equations (PDE's) for nonlinear Local Fractional PDE'sand Randomly generated grids. *International Journal of Disaster Recovery and Business Continuity, 11*(01), 2429-2436.

Mastoi, S., Mugheri, A. B. M., Kalhoro, N. B., Rajput, U. A., Mastoi, R. B., Mastoi, N., & Othman, W. A. M. (2021). Finite difference algorithm on Finite random grids. *International Journal of Advanced Research in Engineering and Technology (IJARET).*

Mastoi, S., Othman, W. A. M., Ali, U., Rajput, U. A., & Fizza, G. (2021). Numerical Solution of the Partial Differential Equation using Randomly Generated Finite Grids and Two-Dimensional Fractional-Order Legendre Function. *Journal of Mechanics of continua and Mathematical Sciences (www.journalimcms.org), 16*(06), 39-51.

Mastoi, S., Othman, W. A. M., & Nallasamy., K. (2020a). Numerical Solution of Second order Fractional PDE's by using Finite difference Method over randomly generated grids. *International Journal of Advance Science and Technology, 29*(11), 373-381.

Mastoi, S., Othman, W. A. M., & Nallasamy., K. (2020b). Randomly generated grids and Laplace Transform for Partial differential equation. *International Journal Disaster Recovery and Business continuity, 11*(01), 1694-1702.

Miller, K. S. (2020). *Partial differential equations in engineering problems*: Courier Dover Publications.

Moffatt, H. (1997). *Numerical methods in finance* (Vol. 13): Cambridge University Press.

Nallasamy., S. M. W. A. M. O. K. (2020). Randomly generated grids and Laplace Transform for Partial differential equation. *International Journal Disaster Recovery and Business continuity, 11*(01), 09. doi:IJDRBC/article/view/24510/12892

Navarro Crespo, I. (2020). Análisis de métodos de integración numérica para problemas dinámicos. Implementación y validación mediante una aplicación de MATLAB.

Othman, S. M. W. A. M. (2020). A Finite difference method using randomly generated grids as non-uniform meshes to solve the partial differential equation. *International Journal of Disaster Recovery and Business Continuity, 11*(01), 13. doi:IJDRBC/article/view/26194/14157

Rani, D., & Mishra, V. (2020). Numerical inverse Laplace transform based on Bernoulli polynomials operational matrix for solving nonlinear differential equations. *Results in Physics, 16*. doi:10.1016/j.rinp.2019.102836

Saeed, S.-T., Riaz, M.-B., Baleanu, D., Akgül, A., & Husnine, S.-M. (2021). Exact Analysis of Second Grade Fluid with Generalized Boundary Conditions. *Intelligent Automation \& Soft Computing, 28*(2), 547--559.

Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., . . . Rabczuk, T. (2020). An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering, 362*, 112790.

Sankarganesh, P., Gowtham, P., Thamilarasan, J., & Karthik, K. (2018). Steady-state temperature distribution in a tetragonal solid. *International Journal of Ambient Energy, 41*(6), 686-690. doi:10.1080/01430750.2018.1484805

Schiesser, W. E. (2012). *The numerical method of lines: integration of partial differential equations*: Elsevier.

Shchepetkin, A. F., & McWilliams, J. C. (2005). The regional oceanic modeling system (ROMS): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean modelling, 9*(4), 347-404.

Shekari, Y., Tayebi, A., & Heydari, M. H. (2019). A meshfree approach for solving 2D variable-order fractional nonlinear diffusion-wave equation. *Computer Methods* *in Applied Mechanics and Engineering, 350*, 154-168. doi:https://doi.org/10.1016/j.cma.2019.02.035

Smitha, T. V., & Nagaraja, K. V. (2020). MATLAB automated higher-order tetrahedral mesh generator for CAD geometries and a finite element application with the subparametric mappings. *Materials Today: Proceedings*. doi:10.1016/j.matpr.2020.09.546

Sohail, M., Ali, U., Al-Mdallal, Q., Thounthong, P., Sherif, E.-S. M., Alrabaiah, H., & Abdelmalek, Z. (2020). Theoretical and numerical investigation of entropy for the variable thermophysical characteristics of couple stress material: Applications to optimization. *Alexandria Engineering Journal, 59*(6), 4365-4375. doi:https://doi.org/10.1016/j.aej.2020.07.042

Song, F., & Karniadakis, G. E. (2019). Fractional magneto-hydrodynamics: Algorithms and applications. *Journal of Computational Physics, 378*, 44-62. doi:10.1016/j.jcp.2018.10.047

Srivastava, H. M., Kung, K. Y., & Wang, K. J. (2007). Analytic solutions of a two-dimensional rectangular heat equation. *Russian Journal of Mathematical Physics, 14*(1), 115-119. doi:10.1134/s1061920807010086

Strauss, W. A. (2007). *Partial differential equations: An introduction*: John Wiley & Sons.

Sun, S., Gou, Z., & Geng, M. (2020). Simultaneous Smoothing and Untangling of 2D Meshes Based on Explicit Element Geometric Transformation and Element Stitching. *Applied Sciences, 10*(14). doi:10.3390/app10145019

Sun, Y., Sun, W., & Zheng, H. (2021). Domain decomposition method for the fully-mixed Stokes–Darcy coupled problem. *Computer Methods in Applied Mechanics and Engineering, 374*, 113578.

Thomas, J. W. (2013). *Numerical partial differential equations: finite difference methods* (Vol. 22): Springer Science & Business Media.

Uzunca, M., Karasözen, B., & Küçükseyhan, T. (2017). Moving mesh discontinuous Galerkin methods for PDEs with traveling waves. *Applied Mathematics and Computation, 292*, 9-18. doi:10.1016/j.amc.2016.07.034

van den Ende, J. A., Smets, M. M., de Jong, D. T., Brugman, S. J., Ensing, B., Tinnemans, P. T., . . . Cuppen, H. M. (2015). Do solid-to-solid polymorphic transitions in DL-norleucine proceed through nucleation? *Faraday discussions, 179*, 421-436.

Vargas-González, S., Núñez-Gómez, K. S., López-Sánchez, E., Tejero-Andrade, J. M., Ruiz-López, I. I., & García-Alvarado, M. A. (2020). Thermodynamic and mathematical analysis of modified Luikov's equations for simultaneous heat and mass transfer. *International Communications in Heat and Mass Transfer*. doi:10.1016/j.icheatmasstransfer.2020.105003

Wang, H., & Yamamoto, N. (2020). Using a partial differential equation with Google Mobility data to predict COVID-19 in Arizona. *Mathematical Biosciences and Engineering, 17*(5).

Wazwaz, A.-M. (2008). The Hirota’s direct method for multiple-soliton solutions for three model equations of shallow water waves. *Applied Mathematics and Computation, 201*(1-2), 489-503.

Xiong, J., Wen, J., & Zheng, H. (2020). An improved local radial basis function collocation method based on the domain decomposition for composite wall. *Engineering Analysis with Boundary Elements, 120*, 246-252. doi:10.1016/j.enganabound.2020.09.002

Xu, F., Huang, Q., & Ma, H. (2020). A novel domain decomposition framework for the ground state solution of Bose–Einstein condensates. *Computers & Mathematics with Applications, 80*(5), 1287-1300. doi:10.1016/j.camwa.2020.06.014

Yan, X., Rennie, C. D., & Mohammadian, A. (2021). Numerical modeling of local scour at a submerged weir with a downstream slope using a coupled moving-mesh and masked-element approach. *International Journal of Sediment Research, 36*(2), 279-290. doi:10.1016/j.ijsrc.2020.06.007

Yokus, A., Durur, H., Ahmad, H., Thounthong, P., & Zhang, Y.-F. (2020). Construction of exact traveling wave solutions of the Bogoyavlenskii equation by (G′/G, 1/G)-expansion and (1/G′)-expansion techniques. *Results in Physics, 19,* 103409.

Zhao, J., Yi, Y., & Xu, Y. (2020). Strong convergence and stability of the split-step theta method for highly nonlinear neutral stochastic delay integro differential equation. *Applied Numerical Mathematics, 157,* 385-404. doi:10.1016/j.apnum.2020.06.013

Zhao, X., Mo, Z.-L., Guo, Z.-Y., & Li, J. (2020). A modified three-dimensional virtual crack closure technique for calculating stress intensity factors with arbitrarily shaped finite element mesh arrangements across the crack front. *Theoretical and Applied Fracture Mechanics, 109*. doi:10.1016/j.tafmec.2020.102695

Zhong, D., & Sheng, C. (2020). A new method towards high-order weno schemes on structured and unstructured grids. *Computers & Fluids, 200*. doi:10.1016/j.compfluid.2020.104453

Zubar, T., Fedosyuk, V., Tishkevich, D., Kanafyev, O., Astapovich, K., Kozlovskiy, A., . . . Trukhanov, A. (2020). The Effect of Heat Treatment on the Microstructure and Mechanical Properties of 2D Nanostructured Au/NiFe System. *Nanomaterials (Basel), 10*(6). doi:10.3390/nano10061077

Zwillinger, D. (1998). *Handbook of differential equations* (Vol. 1): Gulf Professional Publishing.