Wavelet and its variants based numerical methods for Partial Differential Equations

A Thesis

submitted in partial fulfillment of the requirements for the award of the degree of

Doctor of Philosophy

in

School of Mathematics

by

Deepika Sharma
Reg no: 901511007

under the supervison of

Dr. Kavita
Assistant Professor
School of Mathematics

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA-147004, Punjab, India
October 17, 2019

.....dedicated to my parents

Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **Wavelet and its variants based numerical methods for Partial Differential Equations**, in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted to the institution is an authentic record of my own work carried out during the period **25 January, 2016** to **October 17, 2019** under the supervision of **Dr. Kavita**, Assistant Professor, School of Mathematics , Thapar Institute of Engineering and Technology, Patiala.

Date: 17-10-2019

Deepika

(**Deepika Sharma**)
Candidate

It is certified that the above statement made by the candidate is correct to the best of my knowledge.

Date: 17-10-2019

Kavita

Dr. Kavita
Supervisor

Acknowledgements

Completion of this doctoral dissertation was possible with the support of several people. I would like to express my sincere gratitude to all of them.

First of all, I would like to thank the almighty for granting perseverance. Every project is successful largely due to the effort of wonderful people who have always given their valuable advice or lent a helping hand. I would like to express my sincere gratitude to my supervisor *Dr.Kavita*, Assistant Professor, School of Mathematics, TIET, Patiala for the continuous support of my Ph.D. study and related research, for her patience, motivation, enthusiasm and immense knowledge. It has been an honor to be her first Ph.D. student. I appreciate all her contributions of time and ideas to make my Ph.D. experience productive and stimulating. The joy and enthusiasm she has for her research was contagious and motivational for me, even during tough times in the Ph.D. pursuit. On a personal level, she always inspired me by her hardworking and passionate attitude. I could not have imagined having a better advisor and mentor for my Ph.D. study. This thesis would not happen to be possible without the ardent support and care she provided me academically and personally. Her advice on both research as well as on my career have been invaluable.

It is with immense gratitude that I acknowledge Dr. Rohit Kumar Singla, Assistant Professor, TIET, Patiala. His advices and discussions were invaluable to me. His attitude towards research always inspires me. I really appreciate him for always being so supportive.

It gives me great pleasure in acknowledging the help from Dr. Harish Garg, Assistant Professor, TIET, Patiala. He has been a great source of knowledge and inspiration. Thank you for always being so helpful and making my Ph.D. journey smooth.

I am thankful to TIET, Patiala authorities for providing me the necessary facilities for the smooth completion of my Ph.D. I would like to give special thanks to my dissertation committee members: Dr. Paramjeet Singh, Dr. Vivek and Dr. Soumendu Jana for their insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives. A special thanks to Dr. Satish Kumar Sharma, Head of the Department, Dr. Arvind kumar lal, former Head of the Department for their support and for providing all the necessary facilities in the department. Their support and sincere attitude towards department have always been encouraging and helped me pave my way for the dissertation. I express my gratitude to all the faculty members and staff of the School of Mathematics, TIET Patiala, for their support.

I would like to acknowledge the Council of scientific and industrial research for providing me financial assistance. I would also like to thank anonymous reviewers and editors of our papers for their useful and enriching suggestions.

I am heartfelt to my closest friends Sonali Sethi and Nancy Verma for the stimulating discussions, for the times we work together before deadlines and for all the moments we have fun together. I greatly value their friendship and deeply appreciate their belief in me. Cheers to Japinder Kaur Saini for being great reliable person to whom I could always talk about my problems and excitements. My special thanks goes to Deepak Sharma for his motivation and support. I share the credit of my work with my friends Navjot Kaur, Prashu Bansal, Mandeep Kaur Saggi, Meenal Singhal, Jagroop Kaur, Kamal Kumar. I cannot write name of each of my friend but I would like to thank all my friends, I feel blessed to have all of you in my life.

Lastly, I owe my gratitude to my family, who have supported me spiritually throughout entire process, both by keeping me harmonious and helping me putting pieces together. Without their support and endless blessings, it is impossible for me to complete my education seamlessly. Without them, I would not be the person i am today. Words can not express how grateful I am to my mother *Mamta Sharma* and my father *Jawahar lal Sharma* for all of the sacrifices that you have made on my behalf. I would also like to express my respect and love to my brothers, *Danish Sharma, Harshit Sharma and Yogesh Sharma* for their unfailing support, unconditional love and continuous encouragement throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them. I owe my deepest gratitude to my chacha ji Jatinder Pal Sharma for his never ending motivation and inspiration and my chachi ji Sonia Sharma for being my best friend all my life. I love her dearly and thank her for all her advice and support.

Deepika Sharma

Abstract

Since the 1990s, wavelet theory has been adopted for numerically solving the partial differential equations (PDEs). There is an immense research available on wavelet based techniques for numerically solving PDEs. But the theory of wavelet to numerically solve PDEs on arbitrary manifolds is yet in its emerging phase. Moreover, handling general boundary conditions using wavelets is also a tedious task. In this thesis, fast adaptive methods based on wavelets and their variants are developed which are easily extendable to general manifolds and can handle general boundaries. Second generation wavelet, spectral graph wavelet (SPGW) and curvelet are used for this purpose. Curvelet is already used in various areas of engineering, but to best of our knowledge it has very thin appearance in the field of PDEs.

We started with Daubechies and second-generation wavelet to get a better understanding of wavelet-based methods for solving PDEs. Daubechies wavelet have been widely used for numerically solving PDEs, but we have used Daubechies wavelet for solving real life problems, *i.e.*, traffic flow problems. Furthermore, we developed a Matlab toolbox which contains the routines for the second generation wavelet transformation and inverse wavelet transformation on the space $\mathcal{L}_2([a, b])$. These wavelet transforms are further used for computing the wavelet and scaling function values ($\psi(x)$ and $\phi(x)$ respectively). We have also included the Matlab code for generating second generation wavelet based adaptive grid in our suite.

As our aim was to tackle PDEs with different boundaries and Daubechies wavelet based methods are limited to periodic boundary conditions, we used second generation wavelet and third generation wavelet. The construction of second and third-generation wavelets is unaffected by boundaries because these wavelets are built directly on the space where one needs to tackle his/her problems. These wavelets can be used for solving all types of boundaries, however in case of classical wavelets like Daubechies wavelets, additional efforts are to be made to include all types of boundaries in addition to its basic construction.

To be able to solve PDEs with solution having orientation features, we shifted to curvelets. Curvelets are attractive, because they effectively describe essential problems in which wavelet designs are far from ideals. The importance of curvelet is described in the following ways:

- Because of the bad orientation selectivity of wavelets, they do not present higher-dimensional irregularities efficiently. This makes curvelets interesting as they can

yield an ideally adapted mathematical architecture to represent functions that display smooth punctuated curve.

- Tools from hereditary multi-scale analysis such as wavelets are insufficient to detect, establish or present a compact presentation of the intermediate dimensional arrangement. Curvelet is the better product because it produces better-adapted choices by consolidating concepts from geometry with ideas from classical multiscale analysis.
- Although there is wide research available for wavelets on arbitrary manifolds, yet the theory of wavelets for numerically solving PDEs on arbitrary manifolds is in its emerging phase. The most significant characteristic of the curvelet is that it can be designed on arbitrary manifolds.

Curvelet based fast adaptive technique is constructed for numerically solving PDEs on general manifolds. This method uses closest point form for approximating the Laplacian-Beltrami (∇^2) operator and some suitable method (e.g., Crank Nicolson) for time integration of a given PDE. Curvelet is used for the following two purposes:

- For compression of the differential operators and hence for the rapid computation of the powers of the matrices associated with the PDE's numerical solution.
- Curvelet coefficients are used as an indicator function to guide the refinement of the node arrangement and in this way curvelet is used for adaptivity.

We have tested the curvelet based adaptive methods on many problems in one dimension, two dimension, three dimension and on the general manifolds. On the sphere, we have solved the non-linear Schrödinger equation (NLS) which is a standout amongst the most important physics models with several applications in various fields, for example, self-focussing in laser pulses, non-linear optics, models of protein elements, plasma material science and so on. The experimental results in MATLAB shows that in terms of computational time, the developed technique is highly efficient.

Altogether, the proposed methods are applied on a large number of test problems. Some of these test problems are elementary PDEs (Burger's equation) and some are more challenging problems (reaction-diffusion equation on the sphere, Schnakenberg model evolving on the surface of ellipsoid). These numerical tests reveal that the proposed methods can accurately capture the development of localized patterns on all scales and adapt the node arrangement accordingly. The CPU time analysis of the methods reveals that the methods are efficient as compared to the traditional methods. The convergence of the proposed methods is also numerically verified.

List of Publications

International Journals

1. D. Sharma and K. Goyal, *Second-generation wavelet optimized finite difference method (SGWOFD) for solution of Burger's equation with different boundary conditions*, International Journal of Wavelets, Multiresolution and Information Processing, Vol. 16(4), pp. 1-29, 2018, doi: 10.1142/S02196913 18500327 (**Impact Factor: 0.540**).

2. D. Sharma and K. Goyal, *Spectral graph wavelet optimized finite difference method for solution of Burger's equation with different boundary conditions*, Journal of Difference Equations and Applications, Vol. 25(3), pp. 373–395, 2019, doi: 10.1080/10236198.201 9.1576656 (**Impact Factor: 0.974**).

3. D. Sharma, R. K. Singla and K. Goyal, *An adaptive grid based curvelet optimized solution for non-linear Schrödinger equation*, International Journal of Modern Physics C, https://doi.org/10.1142/S0129183119501018 (**Impact Factor: 1.09**).

4. D. Sharma, K. Goyal and R. K. Singla, *A curvelet method for numerical solution of Partial Differential Equations*, Applied Numerical Mathematics, https://doi.org/10.1016 /j.apnum.2019.08.029 (**Impact Factor: 1.678**).

Communicated Papers

1. D. Sharma and K. Goyal, *Wavelet optimized upwind conservative method for traffic flow problems.*

2. D. Sharma and K. Goyal, *Wavelet methods for Partial Differential Equations: A review and classification.*

3. K. Goyal and D. Sharma, *A MATLAB suite for second generation wavelets on an interval and the corresponding adaptive grid.*

4. D. Sharma and K. Goyal, *Curvelet optimized method to solve partial differential equations on general manifolds.*

International Conferences

1. D. Sharma and K. Goyal, *Comparison of different numerical schemes for traffic flow problem and the corresponding Daubechies adaptive grid* presented in International Conference on Recent Advances in Theoretical and Computational Partial Differential Equations held at Punjab University, Chandigarh, India, December 5-9, 2016.

2. D. Sharma and K. Goyal, *Curvelet optimized finite difference method for partial differential equations* presented in International Conference on Current Trends in Theoretical and Computational Differential Equations with Applications held at South Asian University, New Delhi, India, December 1-5, 2017.

3. D. Sharma and K. Goyal, *A fast adaptive curvelet method for Schrödinger equation* presented in International Conference & 14th Biennial Conference of Indian Society of Industrial and Applied Mathematics (ISIAM) held at Guru Nanak Dev University, Amritsar, India, February 2-4, 2018.

4. D. Sharma and K. Goyal, *Finite difference schemes for traffic flow problem on an adaptive Daubechies grid* in International Conference on Frontiers in Industrial and Applied Mathematics (FIAM) held at National Institute of Technology, Hamirpur, India, April 26-27, 2018.

5. D. Sharma and K. Goyal, *Curvelet optimized method for solving partial differential equations on general manifolds* in International Conference on Differential Equations and Control Problems: Modeling, Analysis and Computations (ICDECP19) held at Indian Institute of Technology Mandi, Himachal Pradesh, India, June 17-19, 2019.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the late 1980s, Grossmann and Morlet [1] introduced wavelet bases as a tool for image and signal processing. Wavelet applications at the beginning include seismic signals analysis, image processing, pattern recognition [2, 3], edge-detection [4], image compression, denoising [5], speech recognition [6] and computer graphics [7]. Their potency is now well established in many of the above fields such as the U.S. Federal Department of Investigation uses wavelets in its finger impression database and are one of the constituents of the new compression standard for MPEG media. It was seen that such bases permit the objects (images, signals and turbulent fields) to be represented with singularities of complicated structures with small fraction of degrees of freedom, a feature that is especially encouraging when considering of an application for numerically solving partial differential equations (PDEs). It is verifiable truth that numerous non-linear PDEs emerging in the real world have solutions in which the shocks are developed (*e.g.*, that contains local phenomena) and there are interactions between various scales (*e.g.*, turbulence). In wavelet spaces [8, 9], these solutions are usually well-represented. In combination with the mathematical rigor of multi-resolution analysis, the capability of wavelets to recognize and isolate localized framework, for example, vortices and shocks made them interesting contestants for adaptive computational methods. Shortly, numerous appealing mathematical characteristics of wavelets (such as compact support, property of localization in wave-number and physical space, effective multi-scale decomposition, the presence of fast wavelet transformation [10] and vanishing moments etc.) along with the approaches for pre-conditioning and compressing the matrices [11] and operators motivates their utilization for the adaptive numerical solutions of PDEs.

The first attempt was made to use wavelet bases for numerically solving PDEs in the beginning of 1990s, when the first straightforward adaptive wavelet technique was developed [12]. In 1991, Beylkin executed the survey of numerical estimation in view of wavelets. The research was represented in the sort of military AD note [13], conference paper [14] and official journals [15, 16]. For the calculation purpose, Daubechies wavelet was used. Consequently, Jaffard demonstrated prevalence of resolving elliptic PDEs by using the wavelet [17, 18]. Zweig noticed that the general wave condition, whereon the continuous

wavelet transform depends, could be employed to know the phenomenon associated in the sense of hearing procedure [19]. Chen and Dahmen started associated analyzes [20, 21]. These former research reports had a grand influence and encouraged the wavelet applications in numerical estimation. From this point forward, plenty of research foundations and educational institutions started to direct the wavelet-study. These studies demonstrated that wavelet multiresolution, its characteristics, vanishing moment, short support and norm-equivalence were supreme and universal in solving the differential equation. Different wavelet approaches have been advanced for solving several types of linear and non-linear PDEs for example heat and transport equations [22], Laplace/Poisson equations [23] and reaction-diffusion equations [24–26], Burger's equation [27, 28], the nonlinear Schrödinger equations [29] and the Stokes equations [30, 31]. In 1995, a multilevel wavelet-collocation approach was developed by Vasilyev for solving PDEs [32]. This method uses the standard concept of collocation with wavelet approximation. The multilevel wavelet collocation technique suggested by Vasilyev depends on the localization characteristics of wavelets. Tchiamichian and Liandrat [27], Maday et al. [33], Ravel and Maday [34], Bertoluzza et al. [35] and Bacry et al. [36] have demonstrated that multi-resolution form of bases of wavelet is easy as well as useful structure for spatially adaptive approaches. In 1996, a vigorously adaptive multilevel wavelet collocation approach was developed by Paolucci and Vasilyev for solving PDEs in one spatial domain [37]. The method is an extension of collocation method proposed in [32]. In 1997, Paolucci and Vasilyev proposed a fast adaptive collocation approach for multi-dimensional PDEs [38]. A large portion of the wavelet calculations for tackling PDEs can deal with periodic boundaries easily [27, 33, 35, 36]. The efficient handling of general boundaries is yet an open query although various prospects to deal with this issue have been designed [39–43].

In spite of the vast literature available, the subject of wavelet based techniques for numerically solving PDEs on arbitrary manifolds is yet in its emerging phase. Numerous approaches have been developed for constructing the wavelets on arbitrary manifolds. The wavelet basis in [44, 45] is developed in light of certain forms of manifolds which could be presented as a separate union of the standard cube's continuous parametrization. It has numerous drawbacks from a practical viewpoint as its formation depends entirely on smooth parametric images of the unit cube. This issue is settled in [46], where finite-element supported bases of wavelet respecting a random initial triangularisation are developed. In [47] wavelets are built on the sphere. For solving PDEs on the sphere, an adaptive wavelet collocation technique has been proposed by M. Mehra and N. Kevlahan by using second generation spherical wavelets [48] in 2008 and this work was further extended for elliptical problems in [49]. The method is a generalization of the adaptive collocation approach to the curved manifolds. The biggest problem with the second generation wavelet is that to

approximate the manifold, an initial mesh structure is needed.

In spite of that, for estimating the general manifold, an initial mesh framework is hard to obtain. The mesh-free techniques can handle this problem. One can refer [50, 51] for details of mesh-free strategies in PDEs. In [50, 52] from the scattered data evaluation and PDEs view-point, mesh-free approximation techniques, for instance, moving least-square method and radial basis functions are considered. [51] accumulates survey of the methods that contribute to the advancement of this research area. In 2014, an adaptive meshfree diffusion wavelet technique has been proposed for tackling PDEs on the sphere [53]. In [54] spectral graph wavelet based fast adaptive and meshfree method has been proposed for numerically solving PDEs on sphere. This approach utilizes the radial basis functions for discretizing the space and some suitable method for discretizing time of a given PDE.

Bertoluzza et al. considered evaluation of error and efficiency of wavelet based numerical algorithms and wavelet collocation methods [55–58]. One variant of the wavelet collocation technique is the adaptive method for the numerical solution of PDEs on a staggered non-tensorial grid on the sphere [59–61]. This requires two families of wavelets and an appropriate way to ensure that thresholding achieves the required error control. Kaber et al. proposed wavelet multiscale algorithms and wavelet finite volume methods [62, 63]. Berrone et al. proposed wavelet-Galerkin algorithm in any solution region [64]. Amaratunga, Williams and Radha employed wavelet optimized finite element approach for exploring the microscale atomic framework [65, 66]. Micchelli and Chen proposed Galerkin algorithms of discrete wavelet [67]. Piessens et al. proposed a course on applications of wavelets considered in the numerical analysis [68]. He et al. also conducted progressive analysis of wavelet-based adaptive schemes and error estimation [69–72].

Throughout the previous two generations, the speculations of numerical techniques based on wavelet have been expanded in a various fields. In a nutshell from the prospects of algorithm formation, the primary numerical analysis approaches based on wavelets are classified as follows:

a. Wavelet Galerkin technique

b. Wavelet collocation technique

c. Wavelet optimized technique

   - Wavelet optimized finite difference technique
   - Wavelet optimized finite element technique
   - Wavelet optimized finite volume technique

d. Wavelet meshless technique

e. Wavelet boundary element (WBE) approach

f. Miscellaneous wavelet-supported numerical techniques

In the class of the previously mentioned techniques, weighted residual approach *i.e.*, wavelet-Galerkin approach and wavelet-collocation approach are the primeval numerical algorithms in computational mechanics, therefore it is important to represent these algorithms in an autonomous category despite the fact that they have common background with another techniques. In view of the Galerkin approach *i.e.*, the famous weighted residual approach, finite element method (FEM) is employed. As compared to another methods, FEM is more universal in nature. Various beneficial software is developed on the basis of FEM, in particular NASTRAN, ANSYS. In spite of that, FEM's efficiency in specific areas is not ideal. Boundary element method (BEM) is proposed for enhancing the efficiency and accuracy of FEM. Moreover, meshing is a vital and severe task in BEM or FEM for any complicated figures. To avoid such issues, the meshless approach has been developed. In the standard numerical strategies, for the purpose of approximating unknown quantities, it is critical and important for generating the shape functions that are complex, time-consumable and in fact more difficult to understand in some particular situation. Besides, the complexness of shape functions will bring about the increment in computational rate in the entire solution procedure. Determining a new technique that is naive and feasible is beneficial for developing shape functions. In addition, it appears to be a difficult job. Therefore we should turn to any other mathematical implementations. The classical strategies, in conjunction with wavelet, represent a new aspect in the execution. They behave like the fundamental foundation of characterization and conceptual basis. Wavelet based numerical methods have the following benefits:

1. The localization of wavelets, both in space and scale, governs an efficient sparse representation of pseudo differential operators (and its inverses) and functions by taking non-linear thresholding of the coefficients of wavelet and governs the matrices depicting the operators.

2. The best marvelous characteristic of study of wavelet for numerically solving PDEs is their capability to evaluate the local regularity of the result, as by local mesh refinement, it accepts self-adaptive discretizations. Moreover, the evaluation of function spaces with respect to coefficients of wavelet function and the associated standard norm equivalence [18, 73] approves preconditioning of diagonal operators in wavelet space.

3. The differential operator can be directly computed in a wavelet domain with high

speed and accuracy by setting threshold values in the domain of wavelet.

4. The presence of the FWT methods yield calculations with ideal linear complexity.

Regardless of above points of interest, wavelets have a few restrictions, for example, poor orientation sensitivity. To mitigate these limitations, curvelets [74, 75], shearlets [76], bendlets [77] and many more had been introduced. These are important as they can effectively address interesting issues where wavelet concepts are far away from ideals. They appear in terms of basic elements that exhibit excessive sensitivity to direction and are also highly anisotropic. They can be utilized to deal with PDEs on general manifolds, because they can be structured on general manifolds.

In this thesis, wavelet and its variants based fast adaptive methods are developed which are easily extendable to general manifolds. Daubechies wavelet, second-generation wavelet, spectral graph wavelet (SPGW) and curvelet are used for this purpose. Curvelets are already used in various areas of engineering, but to best of our knowledge it has very thin visibility in the field of PDEs. We have made an attempt to exploit useful properties of curvelets for numerical solutions of PDEs. We started with Daubechies and second-generation wavelet to get a better understanding of wavelet-based methods for solving PDEs. Daubechies wavelet have been widely used for numerically solving PDEs, but we have used Daubechies wavelet for solving real life problems *i.e.*, traffic flow problems. Furthermore, we developed a Matlab toolbox which contains the routines for the second generation wavelet transformation and inverse wavelet transformation on the space $\mathcal{L}_2([a, b])$. These wavelet transforms are further used for computing the wavelet and scaling function values ($\psi(x)$ and $\phi(x)$ respectively.

As our aim was to tackle PDEs with different boundaries and Daubechies wavelet based methods are limited to periodic boundary conditions, we used second generation wavelet and third generation wavelet. The construction of second and third-generation wavelets are not limited to periodic boundary conditions because these wavelets are built directly on the space where one needs to tackle his/her problems. These wavelets can be used for solving all types of boundaries, however in case of classical wavelets like Daubechies wavelets, additional efforts should be made to include all types of boundaries in addition to its basic construction. Therefore, in this thesis, second and third generation wavelet based fast adaptive methods are developed for solving PDEs.

This chapter provides a brief description of Daubechies wavelet. In section (1.2), a concise explanation of how wavelets can be utilized for numerically solving PDEs has been provided. Organization of the thesis is given in section (1.3).

1.1 Daubechies wavelet

Named after Ingrid Daubechies, the Daubechies wavelets [78] are a class of compactly supported orthogonal wavelets. These are the most famous in the class of all first generation wavelets.

1.1.1 Real line

Multi-resolution analysis (MRA) is a common structure for the wavelet theory, a mathematical structure that generally describes wavelets. MRA provides basic insights into the theory of wavelets and also leads to important algorithms. MRA's objective is to represent the random function $f \in \mathcal{L}_2(\mathbb{R})$ at different levels of detail. Daubechies in [78, 79] used the same for showing that $\exists$ an orthogonal wavelet with short support for a positive integer n such that all the derivatives upto the nth order exist. The following axioms described the MRA:

1. $\mathcal{V}^j \subset \mathcal{V}^{j+1}$ (subspaces are nested),
2. $f(x) \in \mathcal{V}^j$ iff $f(2x) \in \mathcal{V}^{j+1}$ for all $j \in \mathbb{Z}$ (invariance to dilation),
3. $\overline{\bigcup_{j\in\mathbb{Z}} \mathcal{V}^j} = \mathcal{L}_2(\mathbb{R})$,
4. $\{\phi(x-k)|k \in \mathbb{Z}\}$ is an orthonormal basis for $\mathcal{V}^0$ (invariance to translation) for a function $\phi(x) \in \mathcal{V}^0$; this is called the scaling function.

A sequence of nested approximation spaces $\mathcal{V}^j$ in $\mathcal{L}_2(\mathbb{R})$ is described by MRA, such that the closure of their union is equal to $\mathcal{L}_2(\mathbb{R})$. The approximation of f is the projection of a function $f \in \mathcal{L}_2(\mathbb{R})$ onto $\mathcal{V}^j$ that converges to f as $j \to \infty$.

Given the nested subspaces $\mathcal{V}^j$, the detail space $\mathcal{W}^j$ is defined as the orthogonal complement of $\mathcal{V}^j$ in $\mathcal{V}^{j+1}$, *i.e.*, $\mathcal{V}^j \perp \mathcal{W}^j$ and

$$\mathcal{V}^{j+1} = \mathcal{V}^j \oplus \mathcal{W}^j. \tag{1.1.1}$$

Consider now two sub-spaces $\mathcal{V}^{J_0}$ and $\mathcal{V}^J$ with $J > J_0$. Applying (1.1.1) recursively we obtain

$$\mathcal{V}^J = \mathcal{V}^{J_0} \oplus \left(\bigoplus_{j=J_0}^{J-1} \mathcal{W}^j \right). \tag{1.1.2}$$

Thus any function in $\mathcal{V}^J$ can be represented as a linearly combination of the functions in $\mathcal{V}^{J_0}$ and $\mathcal{W}^j, j = J_0, J_0+1, \cdots, J-1$; hence it can be analyzed on different scales separately. From this separation of scales, MRA received its name. Note that we will call J_0 as the coarsest approximation level and J as the finest approximation level.

As the set $\{\phi(x-k)|k \in \mathbb{Z}\}$ is an orthonormal basis for $\mathcal{V}^0$ by axiom (4) of MRA, it follows by repeated application of axiom (2) that $\{\phi_k^j(x) = 2^{j/2}\phi(2^j x - k)|k \in \mathbb{Z}\}$ is an ortho-normal basis for $\mathcal{V}^j$ and similarly, there exists a function $\psi(x) \in \mathcal{W}^0$ (which is called the mother wavelet) such that $\{\psi_k^j(x) = 2^{j/2}\psi(2^j x - k)|k \in \mathbb{Z}\}$ is an orthonormal basis for $\mathcal{W}^j$. Since $\phi_0^0(x) = \phi(x) \in \mathcal{V}^0 \subset \mathcal{V}^1$, we have

$$\phi(x) = \sum_{k=-\infty}^{\infty} h_k \phi_k^1(x), \quad h_k = \int_{-\infty}^{\infty} \phi(x)\phi_k^1(x)dx. \tag{1.1.3}$$

Only finitely many $h_k, k = 0, 1, \cdots D-1$ will be non-zero for Daubechies compactly supported scaling function, here D is an even non-negative integer and is known as the wavelet-genus. Therefore we have

$$\phi(x) = \sqrt{2}\sum_{k=0}^{D-1} h_k \phi(2x-k). \tag{1.1.4}$$

Eqn. (1.1.4) is called the dilation-equation (for the scaling function it is a two scale-relation) and $h_0, h_1, \cdots, h_{D-1}$ are called the low-pass filter coefficients. Likewise, Daubechies compactly supported wavelet $\psi(x) \in \mathcal{W}^0 \subset \mathcal{V}^1$ is defined by

$$\psi(x) = \sqrt{2}\sum_{k=0}^{D-1} g_k \phi(2x-k). \tag{1.1.5}$$

Eqn. (1.1.5) is called the wavelet equation (for the wavelet function it is a two scale relation) and $g_0, g_1, \cdots, g_{D-1}$ are called the high-pass filter coefficients. The relation which connects the filter-coefficients is as: $g_k = (-1)^k h_{D-1-k}, k = 0, 1, \cdots, D-1$. An important consequence of (1.1.4) and (1.1.5) is that $\text{supp}(\phi) = \text{supp}(\psi) = [0, D-1]$, see [78]. It follows that

$$supp(\phi_k^j) = supp(\psi_k^j) = I_k^j = \left[\frac{k}{2^j}, \frac{k+D-1}{2^j}\right].$$

It should be noted that there are generally no closed (explicit) analytical formulas for either Daubechies wavelet or scaling functions ($\psi(x)$ and $\phi(x)$ respectively) excepting the Haar scaling function ($\phi(x) = 1$ if $x \in [0,1)$, $\phi(x) = 0$ else) and the Haar wavelet function ($\psi(x) = 1$ if $x \in [0, .5)$, $\psi(x) = -1$ if $x \in [.5, 1)$, $\psi(x) = 0$ else). Furthermore, values can be calculated at dyadic points for general scaling and wavelet functions using the cascade algorithm [78, 80].

1.1.2 Periodic domain

Earlier, on the whole real line, our functions have been defined, *i.e.*, $f \in \mathcal{L}_2(\mathbb{R})$. The domain of space is a finite interval for most of the useful applications for instance data fitting, image processing or problems involving differential equations say, for simplicity, the function f is periodic on the interval $[0,1]$ *i.e.*, $f(0) = f(1)$. These situations can be managed with using the following periodic scaling and wavelet functions:
Let $\phi \in \mathcal{L}_2(\mathbb{R})$ and $\psi \in \mathcal{L}_2(\mathbb{R})$ represents the basic scaling function and the basic wavelet that from an MRA as defined in section (1.1.1). We define the 1-periodic scaling function for any $j, k \in \mathbb{Z}$ as

$$\hat{\phi}_k^j(x) = \sum_{n=-\infty}^{\infty} \phi_k^j(x+n) = 2^{j/2} \sum_{n=-\infty}^{\infty} \phi(2^j(x+n)-k), x \in \mathbb{R},$$

and the 1-periodic wavelet

$$\hat{\psi}_k^j(x) = \sum_{n=-\infty}^{\infty} \psi_k^j(x+n) = 2^{j/2} \sum_{n=-\infty}^{\infty} \psi(2^j(x+n)-k), x \in \mathbb{R}. \tag{1.1.6}$$

The 1-periodicity can be verified as follows

$$\hat{\phi}_k^j(x+1) = \sum_{n=-\infty}^{\infty} \phi_k^j(x+n+1) = \sum_{m=-\infty}^{\infty} \phi_k^j(x+m) = \hat{\phi}_k^j(x),$$

and similarly $\hat{\psi}_k^j(x+1) = \hat{\psi}_k^j(x)$. With regard to periodic scaling and wavelet functions [10], some of the useful results are as:

1. $\hat{\phi}_k^j(x)$ is constant and is equal to $2^{-j/2}$ for $j \leq 0$.
2. $\hat{\psi}_k^j(x) = 0$ for $j \leq -1$.
3. $\hat{\phi}_k^j(x)$ and $\hat{\psi}_k^j(x)$ are periodic in the shift parameter k with period 2^j for $j > 0$.

Now suppose

$$\hat{\mathcal{V}}^j = \overline{\left\langle \left\{ \hat{\phi}_k^j(x), x \in [0,1] \right\}_{k=0}^{2^j-1} \right\rangle} \text{ and } \hat{\mathcal{W}}^j = \overline{\left\langle \left\{ \hat{\psi}_k^j(x), x \in [0,1] \right\}_{k=0}^{2^j-1} \right\rangle},$$

it is observed that the $\hat{\mathcal{V}}^j$ nests in the same way as the $\mathcal{V}^j$ nests in MRA axiom 1 (see section 1.1.1), *i.e.*,

$$\hat{\mathcal{V}}^0 \subset \hat{\mathcal{V}}^1 \subset \hat{\mathcal{V}}^2 \subset \cdots \subset \mathcal{L}_2([0,1]),$$

and $\overline{\bigcup_{j=0}^{\infty} \hat{\mathcal{V}}^j} = \mathcal{L}_2([0,1])$. The orthogonality property of the non periodic wavelet and scaling functions continued to the interval limited periodic versions which indicates that

$$\hat{\mathcal{V}}^j \oplus \hat{\mathcal{W}}^j = \hat{\mathcal{V}}^{j+1}.$$

So the space $\mathcal{L}_2([0,1])$ has the decomposition

$$\mathcal{L}_2([0,1]) \approx \hat{\mathcal{V}}^{J_0} \oplus \left(\bigoplus_{j=J_0}^{\infty} \hat{\mathcal{W}}^j \right),$$

for some $J_0 > 0$.

1.1.2.1 Scaling function transformation and Inverse scaling function transformation

Assume that a function $f \in \mathcal{L}_2([0,1])$ with $f(0) = f(1)$ is given and we want to evaluate the projection of this function onto the space $\hat{\mathcal{V}}^j$, *i.e.*,

$$P_{\hat{\mathcal{V}}^j} f(x) = \sum_{k=0}^{2^j-1} c_k^j \hat{\phi}_k^j(x), \quad x \in [0,1]. \tag{1.1.7}$$

It should be noted that there are two natural ways of obtaining the scaling function coefficients c_k^j of (1.1.7).

1. **Projection:** As the basis functions are orthogonal, hence the coefficients c_k^j can be computed by utilizing the following result

   $$c_k^j = \int_I f(x) \hat{\phi}_k^j(x) dx.$$

   This is known as the orthogonal projection method. An adequately accurate quadrature technique can approximate the integral.

2. **Interpolation:** The coefficients c_k^j are selected such that the projection of f on $\mathcal{V}^j$, and f coincides at the node points at level j, *i.e.*,

   $$f\left(\frac{l}{2^r}\right) = \sum_{k=0}^{2^j-1} c_k^j \hat{\phi}_k^j \left(\frac{l}{2^r}\right), \quad l = 0, 1, \cdots, 2^r - 1,$$

   here $r \in \mathbb{N}$ is known as the function's dyadic resolution.

By using the interpolation technique for obtaining the c_k^j we can obtain from (1.1.7)

$$\begin{aligned} f\left(\frac{l}{2^r}\right) &= \sum_{k=0}^{2^j-1} c_k^j \hat{\phi}_k^j\left(\frac{l}{2^r}\right), \\ &= \sum_{k=0}^{2^j-1} c_k^j \sum_{n\in\mathbb{Z}} \phi_k^j\left(\frac{l}{2^r}+n\right), \\ &= 2^{j/2}\sum_{k=0}^{2^j-1} c_k^j \sum_{n\in\mathbb{Z}} \phi\left(\frac{m(l,k)+2^{j+q}n}{2^q}\right), \end{aligned}$$

where $m(l,k) = l2^{j+q-r} - k2^q$. Now if j is such that $2^j \geq D-1$, then we have

$$f\left(\frac{l}{2^r}\right) = 2^{j/2}\sum_{k=0}^{2^j-1} c_k^j \phi\left(\frac{\langle m(l,k)\rangle_{2^{j+q}}}{2^q}\right), \quad l = 0, 1, \cdots, 2^r-1, \tag{1.1.8}$$

(see [10] for reference). From (1.1.8) we see that $m(l,k)$ serves as an index into the vector of pre-computed values of ϕ. For this to make sense $m(l,k)$ must be an integer, which leads to the restriction

$$j+q-r \geq 0.$$

Suppose $c_j = [c_0^j, c_1^j, \cdots, c_{2^j-1}^j]^T$ and $f_r = [f(0), f(1/2^r), \cdots, f((2^r-1)/2^r)]^T$, then (1.1.8) can be written as

$$f_r = T_{r,j}c_j, \tag{1.1.9}$$

where $T_{r,j}$ is a matrix of size $2^r \times 2^j$. Given f_r, computing c_j using (1.1.9) is called the scaling function transformation (ST) and given the c_j, computing f_r using (1.1.9) is called the inverse scaling function transformation (IST).

1.1.2.2 Wavelet and Inverse wavelet transform

For $f \in \mathcal{L}_2(\mathbb{R})$, the projection onto the space $\mathcal{V}^j$ is given as

$$P_{\mathcal{V}^j} f(x) = \sum_{k=-\infty}^{\infty} c_k^j \phi_k^j(x), \tag{1.1.10}$$

which is given only in terms of scaling functions. Now since $\mathcal{V}^j = \mathcal{V}^{j-1} \oplus \mathcal{W}^{j-1}$, we can write $P_{\mathcal{V}^j} f(x)$ in terms of scaling and wavelet functions as

$$P_{\mathcal{V}^j} f(x) = \sum_{k=-\infty}^{\infty} c_k^{j-1}\phi_k^{j-1}(x) + \sum_{k=-\infty}^{\infty} d_k^{j-1}\psi_k^{j-1}(x). \tag{1.1.11}$$

There is a relationship between the coefficients $\{c_k^j\}_{k\in\mathbb{Z}}, \{c_k^{j-1}\}_{k\in\mathbb{Z}}$ and $\{d_k^{j-1}\}_{k\in\mathbb{Z}}$ given by

$$c_k^{j-1} = \sum_{l=0}^{D-1} h_k c_{2k+l}^j, \tag{1.1.12}$$

$$d_k^{j-1} = \sum_{l=0}^{D-1} g_k c_{2k+l}^j. \tag{1.1.13}$$

Computing the coefficients $\{c_k^{j-1}\}_{k\in\mathbb{Z}}$ and $\{d_k^{j-1}\}_{k\in\mathbb{Z}}$ from $\{c_k^j\}_{k\in\mathbb{Z}}$ is called partial wavelet transform (PWT). Applying (1.1.12) and (1.1.13) recursively, for $j = J, J-1, \cdots, J_0+1$ starting from the initial sequence $\{c_k^J\}_{k\in\mathbb{Z}}$ will give us the coefficients of the expansion

$$P_{\mathcal{V}^J} f(x) = \sum_{k=-\infty}^{\infty} c_k^{J_0} \phi_k^{J_0-1}(x) + \sum_{j=J_0}^{J-1} \sum_{k=-\infty}^{\infty} d_k^j \psi_k^{j-1}(x). \tag{1.1.14}$$

This process of computing $\{c_k^{J_0}\}_{k\in\mathbb{Z}}, \{d_k^j, j = J_0, \cdots, J-1\}$ from $\{c_k^J\}_{k\in\mathbb{Z}}$ is called the full wavelet transform (FWT). Similarly there exists an inverse relation

$$c_k^j = \sum_{l=n_1(k)}^{n_2(k)} c_l^{j-1} h_{k-2l} + d_l^{j-1} g_{k-2l}, \tag{1.1.15}$$

where $n_1(k) = \left\lceil \frac{k-D+1}{2} \right\rceil$ and $n_2(k) = \lfloor \frac{k}{2} \rfloor$. Obtaining $\{c_k^j\}_{k\in\mathbb{Z}}$ from $\{c_k^{j-1}\}_{k\in\mathbb{Z}}$ and $\{d_k^{j-1}\}_{k\in\mathbb{Z}}$ is called inverse partial wavelet transform (IPWT). The inverse full wavelet transform (IFWT) is obtained by the repeated application of the (1.1.15) for $j = J_0+1, J_0+2, \cdots, J$.

1.2 Wavelet-based numerical methods

Mathematical characteristics of wavelets inspires its utilization for numerically solving PDEs [81]. The localization of wavelets, both in space and scale, governs an efficient sparse representation of pseudo differential operators (and its inverses) and functions by taking non-linear thresholding of the wavelet's coefficients of the functions. The most finest characteristic of wavelet analysis for numerically solving PDEs is its capability to evaluate the local regularity of the result, that permits adaptive discretizations with natural local mesh refining [82]. Moreover, the evaluation of function spaces with respect to the coefficients of wavelet and the associated norm equivalences [73] permits diagonal preconditioning of operators into the wavelet space. At last, the continuation of the FWT leads to methods with

optimum linear complexity. The reviewer is recommended to the accompanying surveys for a more mathematical overview on wavelet algorithms for PDEs [23, 83, 84].

The present wavelet-based methods can be characterized into various forms relying upon either they take partial or full benefits of wavelet analysis, to be specific, wavelet compression, multiresolution properties, the identification of localized structures and consequent utilization for adapting the grid, interpolation based on wavelet, FWT and active error control. Wavelet based algorithms for solving PDEs are considered as:

1.2.1 Wavelet Galerkin approach

The degrees of freedom in Galerkin strategy are the extension parameters of a class of basic functions and these extension parameters are not within wavelet range (implies that not in physical range). Besides that, in wavelet Galerkin algorithms, dealing with nonlinearities is difficult which can be take care of with a couple of methods.

- Quadrature formula approach [68] (loses its exactness because of the approximating calculations).
- Pseudo approach [85] (firstly project wavelet range to physical range, calculate nonlinear terms in physical range and afterward back to wavelet range, this technique is not very reasonable in light of the fact that it demands the transformation between the physical range and wavelet range).
- Connection coefficients approach [86] (costly approach because of the summation over numerous indices).

1.2.1.1 Connection coefficients approach

Let $f \in \mathcal{V}^j$, then

$$f^{(d)}(x) = \sum_{l=-\infty}^{\infty} c_l^j \phi_l^{j(d)}(x), \quad x \in \mathbb{R}, \tag{1.2.1}$$

$f^{(d)}$ will generally have disconnections with $\mathcal{V}^j$ so $f^{(d)}$ is projected back upon $\mathcal{V}^j$

$$P_{\mathcal{V}^j} f^{(d)}(x) = \sum_{k=-\infty}^{\infty} c_k^{j(d)} \phi_k^j(x), \quad x \in \mathbb{R},$$

here

$$c_k^{j(d)} = \int_{-\infty}^{\infty} f^{(d)}(x)\phi_k^j(x)dx. \tag{1.2.2}$$

Substitute Eqn. (1.2.1) into Eqn. (1.2.2) and performing some operations, we achieve

$$c_k^{j(d)} = \sum_{n=-\infty}^{\infty} c_{n+k}^j 2^{jd}\Gamma_n^d, \quad \infty < k < \infty, \tag{1.2.3}$$

where

$$\Gamma_n^d = \int_{-\infty}^{\infty} \phi(x)\phi_n^{(d)}(x)dx, \quad n \in \mathbb{Z}, \tag{1.2.4}$$

are known as connection coefficients. Since $\phi(x)$ has compact support *i.e.*, $[0, D-1]$ (For the Daubechies wavelet, D is called the genus), it is demonstrated that the support of $\phi_n^{(d)}(x)$ and $\phi(x)$ coincide just for $-(D-2) \leq n \leq (D-2)$, thats why, they have just $2D-3$ non zero connection-coefficients. Hence Eqn. (1.2.3) depreciates as

$$c_k^{j(d)} = \sum_{n=2-D}^{D-2} c_{n+k}^j 2^{jd}\Gamma_n^d, \quad j, k \in \mathbb{Z}. \tag{1.2.5}$$

If f is a function with periodicity one, then

$$c_k^j = c_{k+2^j}^j, \ k \in \mathbb{Z},$$

and

$$c_k^{j(d)} = c_{k+2^j}^{j(d)}, \ k \in \mathbb{Z}.$$

Therefore it is necessary to take into account 2^j coefficients of either type and Eqn. (1.2.5) become

$$c_k^{j(d)} = \sum_{n=2-D}^{D-2} c_{<n+k>_{2^j}}^j 2^{jd}\Gamma_n^d, \quad k = 0, 1, \cdots, 2^j - 1, \tag{1.2.6}$$

that can be composed in the matrix frame as

$$c^{(d)} = D^{(d)}c, \tag{1.2.7}$$

here $[D^{(d)}]_{k,<n+k>_{2^j}} = 2^{jd}\Gamma_n^d$, $k = 0, 1, \cdots, 2^j - 1$; $n = 2-D, 3-D, \cdots, D-2$ and

$$c^{(d)} = [c_0^{j(d)}, c_1^{j(d)}, \cdots, c_{2^j-1}^{j(d)}],$$

where the matrix $D^{(d)}$ is the differentiation projection matrix. If differentiated function is periodic with period L then we get $c^{j(d)} = \mathcal{D}^{(d)} c^j$, here $\mathcal{D}^{(d)} = \frac{D^{(d)}}{L^d}$.

1.2.1.2 Numerical Examples

We illustrate Galerkin method for time-dependent advection-diffusion equation with periodic boundaries.

$$\begin{aligned} \frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} &= \nu \frac{\partial^2 u}{\partial x^2},\ t > 0, \\ u(x,0) &= h(x),\ x \in (0,1), \\ u(x,t) &= u(x+1,t),\ t \geq 0, \end{aligned} \tag{1.2.8}$$

here ν is a positive constant and $h(x) = \sin(kx)$. Now, we will solve this equation by expanding it in scaling function and wavelet function.

Method based on scaling function expansion:- We begin the discretization of Eqn. (1.2.8) with respect to time to achieve the Euler pattern

$$\frac{u^{n+1} - u^n}{\partial t} = \nu u_{xx}^n - a u_x^n,$$

i.e.,

$$u^{n+1} - u^n = \nu \partial t u_{xx}^n - a \partial t u_x^n. \tag{1.2.9}$$

Replace $u(x)$ with the approximation by means of

$$u^j(x) = \sum_{k=0}^{2^j-1} c_k^j \tilde{\phi}_k^j(x), \tag{1.2.10}$$

$$u_x^j(x) = \sum_{k=0}^{2^j-1} (c_u^{(1)})_k^j \tilde{\phi}_k^j(x), \tag{1.2.11}$$

$$u_{xx}^j(x) = \sum_{k=0}^{2^j-1} (c_u^{(2)})_k^j \tilde{\phi}_k^j(x), \tag{1.2.12}$$

where $(c_u^{(d)})_k^j$ is given as,

$$\begin{aligned} (c_u^{(d)})_k^j &= [\mathcal{D}^{(d)} c_u]_k, \\ &= \sum_{n=2-D}^{D-2} (c_u)^j_{<n+k>_{2^j}} 2^{jd} \gamma_n^d, \end{aligned}$$

here d is the function's derivative. Now, Galerkin discretization scheme is used for determining the coefficients $(c_u)_k^j$. Multiply Eqn. (1.2.9) by $\tilde{\phi}_l^j(x)$ and integrate over unit interval gives the relation

$$\int_0^1 (u^{n+1})^j(x)\tilde{\phi}_l^j(x)dx - \int_0^1 (u^n)^j(x)\tilde{\phi}_l^j(x)dx = \nu\partial t \int_0^1 (u_{xx})^j(x)\tilde{\phi}_l^j(x)dx + a\partial t \int_0^1 (u_x)^j(x)\tilde{\phi}_l^j(x)dx,$$

where $l = 0, 1, 2, \ldots, 2^j - 1$. Using Eqn. (1.2.9), Eqn. (1.2.10), Eqn. (2.2.6), Eqn. (1.2.12) and the orthonormality of scaling function, we get

$$\overrightarrow{c}_{u,l}^{n+1} - \overrightarrow{c}_{u,l}^{n} = \nu\partial t\mathcal{D}^{(2)}\overrightarrow{c}_{u,l}^{n} + a\partial t\mathcal{D}^{(1)}\overrightarrow{c}_{u,l}^{n},$$

where $\overrightarrow{c}_{u,l}$ stands for the scaling function's coefficients vector related to u. At last, we come to the algebraic linear system

$$\overrightarrow{c}_{u,l}^{n+1} = \mathcal{A}\overrightarrow{c}_{u,l}^{n}, \tag{1.2.13}$$

where $\mathcal{A} = I + \nu\partial t\mathcal{D}^2 - a\partial t\mathcal{D}^1$ which can be solved. The analytical solution to this problem with periodic boundary condition is:

$$u_{ana}(x,t) = \exp^{-k^2\nu t}\sin(k(x-at)), \tag{1.2.14}$$

where u_{ana} stands for analytical solution. Fig. (1.1) shows the comparison between numerical and analytical value of the Eqn. (1.2.8) at different time. Fig. (1.2) gives the relation between error between numerical and analytical value versus no. of grid points (N). It shows that as N increases, error decreases.

Method based on wavelet function expansion:- Substitute $d_u = Wc_u$ and $d_f = Wc_f$ in Eqn. (1.2.13), it yields

$$\mathcal{A}W^T d_u = W^T d_f, \tag{1.2.15}$$

Let

$$\begin{aligned} \breve{A} &= WAW^T = W(-\mathcal{D}^{(2)} + \alpha I)W^T, \\ &= -W\mathcal{D}^{(2)}W^T + \alpha I, \\ &= -\breve{\mathcal{D}}^{(2)} + \alpha I, \end{aligned}$$

where $\breve{\mathcal{D}}^{(2)} = W\mathcal{D}^{(2)}W^T$. Then

$$\breve{A}d_u = d_f, \tag{1.2.16}$$

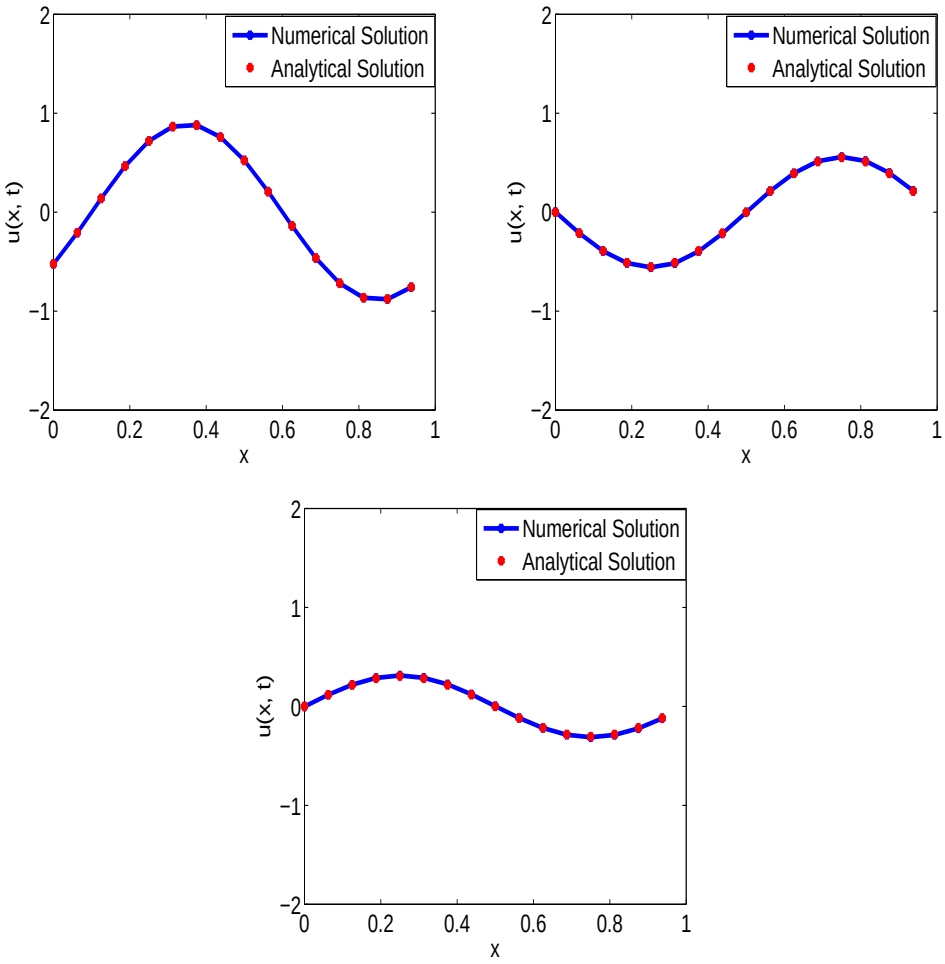


Figure 1.1: Solution of time-dependent linear advection-diffusion problem ($\nu = 0.03$, $a = 1$) at $t = 0.1s$, $0.5s$ and $1s$ using Galerkin technique.

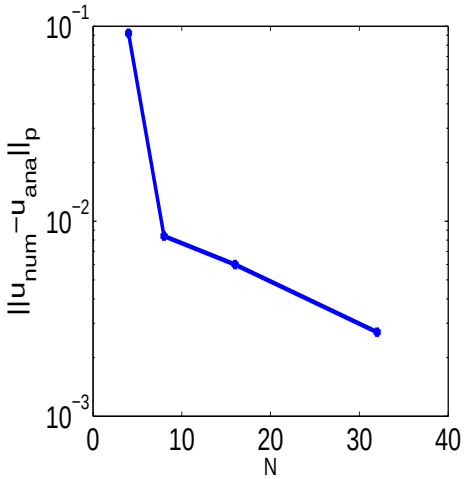


Figure 1.2: Error between numerical and analytical value versus N for Galerkin technique.

which is the wavelet discretization of advection-diffusion equation. Consequently, there is a capability of utilizing wavelet compression to decrease the mathematical complication of Eqn. (1.2.16).

1.2.2 Wavelet Collocation approach

Collocation approach include mathematical operators performing on collocation points (values of point) in the physical range [87]. Wavelet-collocation technique is generally constructed by selecting a wavelet and some kind of grid arrangement that can be adjusted numerically. In fact, one achieves finite differences over non uniform lattice. Dealing with nonlinearities in wavelet-collocation approach is simple job because of collocation nature of method. The procedure for solving this method is explained ahead:

1.2.2.1 Collocation method

In the space $\tilde{\mathcal{V}}^j$, the $f(x)$ function is estimated as

$$P_{\tilde{\mathcal{V}}^j} f(x) = \sum_{k=0}^{2^j-1} c_k^j \tilde{\phi}_k^j(x), \tag{1.2.17}$$

where c_k^j are coefficients of scaling function. Differentiating d (non-negative integer) time, Eqn. (1.2.17) becomes

$$f^{(d)}(x) = 2^{jd} \sum_{k=0}^{2^j-1} c_k^j \tilde{\phi}_k^j(x). \tag{1.2.18}$$

The estimated function will overlap with the exact function at the node points in the region at level j (collocation points) in the collocation approach, hence Eqn. (1.2.18) gives

$$f^{(d)}(l/2^j) = 2^{jd} \sum_{k=0}^{2^j-1} c_k^j \tilde{\phi}_k^j(l/2^j), \tag{1.2.19}$$

where $l = 0, 1, \cdots, 2^j - 1$. Hence by computing the scaling function coefficients, c_k^j, reduces to solve the equation of matrix

$$f^{j(d)} = \mathtt{D}^{(d)} c^j, \tag{1.2.20}$$

here $f^{j(d)} = (f^{(d)}(0), \cdots, f^{(d)}(\frac{2^j-1}{2^j}))$, $c^j = (c_0^j, \cdots, c^j_{\frac{2^j-1}{2^j}})$ and matrix $\mathtt{D}^{(d)}$ is represented as

$$\mathtt{D}^{(d)} = 2^{jd+j/2} \begin{bmatrix} 0 & 0 & \cdots & 0 & \phi^{(d)}_{D-2} & \cdots & \phi^{(d)}_2 & \phi^{(d)}_1 \\ \phi^{(d)}_1 & 0 & \cdots & \vdots & 0 & \vdots & \vdots & \phi^{(d)}_2 \\ \phi^{(d)}_2 & \phi^{(d)}_1 & \cdots & \vdots & \vdots & \vdots & \phi^{(d)}_{D-1} & \vdots \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & 0 & \phi^{(d)}_{D-2} \\ \phi^{(d)}_{D-2} & \phi^{(d)}_{D-3} & \cdots & \vdots & \vdots & \cdots & \vdots & 0 \\ 0 & \phi^{(d)}_{D-2} & \cdots & 0 & \vdots & \cdots & \vdots & \vdots \\ \vdots & 0 & \cdots & \phi^{(d)}_1 & 0 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \vdots & \phi^{(d)}_1 & \cdots & \vdots & \vdots \\ \vdots & \vdots & \cdots & \phi^{(d)}_{D-3} & \vdots & \cdots & 0 & \vdots \\ 0 & \cdots & \cdots & \phi^{(d)}_{D-2} & \phi^{(d)}_{D-3} & \cdots & \phi^{(d)}_1 & 0 \end{bmatrix}$$

For solving c_k^j by using the Eqn. (1.2.20), it is necessary to evaluate $\mathtt{D}^{(d)}$, for which the $\phi^{(d)}$ values at the dyadic rationals are required.

1.2.2.2 Numerical Computation of ϕ

The values of the general wavelet and scaling functions can be calculated by using the cascade algorithm [78, 80] at dyadic points which is described as:
Evaluating ϕ at integers: The scaling function ϕ is compactly supported on the interval $[0, D-1]$ with $\phi(D-1) = 0$ and $\phi(0) = 0$ for $D \geq 4$ [78].

By putting $x = 0, 1, \cdots, D-2$ in

$$\phi(x) = \sqrt{2} \sum_{k=0}^{D-1} h_k \phi(2x - k), \tag{1.2.21}$$

We get a linear homogeneous system of conditions. For $D = 6$ we get

$$\begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = \sqrt{2} \begin{bmatrix} h_0 & & & & \\ h_2 & h_1 & h_0 & & \\ h_4 & h_3 & h_2 & h_1 & h_0 \\ & h_5 & h_4 & h_3 & h_2 \\ & & & h_5 & h_4 \end{bmatrix} \times \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = A_0 \Phi(0), \tag{1.2.22}$$

here the function $\Phi(x)$ is represented by

$$\Phi(x) = [\phi(x), \phi(x+1), \cdots, \phi(x+D-2)]^T,$$

It is noticed that finding the solution of Eqn. (1.2.22) is similar to find the solution of the eigenvalue problem

$$A_0\Phi(0) = \lambda\Phi(0). \tag{1.2.23}$$

The solution of Eqn. (1.2.22) is the eigen-vector of A_0 analogous to the eigen-value $\lambda = 1$. *Evaluating ϕ at dyadic rationals*: After acquiring $\Phi(0)$ from Eqn. (1.2.22), we can further use Eqn. (1.2.21) to get the estimation of ϕ at all the mid-points within the integers in the interval, to be specific, a vector $\Phi(1/2)$. Substituting $x = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \cdots$ into Eqn. (1.2.21) gives

$$\phi\left(\frac{1}{2}\right) = \begin{bmatrix} \phi(\frac{1}{2}) \\ \phi(\frac{3}{2}) \\ \phi(\frac{5}{2}) \\ \phi(\frac{7}{2}) \\ \phi(\frac{9}{2}) \end{bmatrix} = \sqrt{2} \begin{bmatrix} h_0 & & & & \\ h_2 & h_1 & h_0 & & \\ h_4 & h_3 & h_2 & h_1 & h_0 \\ & h_5 & h_4 & h_3 & h_2 \\ & & & h_5 & h_4 \end{bmatrix} \times \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \\ \phi(4) \end{bmatrix} = A_1\Phi(0), \tag{1.2.24}$$

Now, we get the rationals in the form of $\frac{k}{4}$, here k is odd

$$\begin{bmatrix} \phi(\frac{1}{4}) \\ \phi(\frac{3}{4}) \\ \phi(\frac{5}{4}) \\ \phi(\frac{7}{4}) \\ \phi(\frac{9}{4}) \\ \phi(\frac{11}{4}) \\ \phi(\frac{13}{4}) \\ \phi(\frac{15}{4}) \\ \phi(\frac{17}{4}) \\ \phi(\frac{19}{4}) \end{bmatrix} = \sqrt{2} \begin{bmatrix} h_0 & & & & \\ h_1 & h_0 & & & \\ h_2 & h_1 & h_0 & & \\ h_3 & h_2 & h_1 & h_0 & \\ h_4 & h_3 & h_2 & h_1 & h_0 \\ h_5 & h_4 & h_3 & h_2 & h_1 \\ & h_5 & h_4 & h_3 & h_2 \\ & & h_5 & h_4 & h_3 \\ & & & h_5 & h_4 \\ & & & & h_5 \end{bmatrix} \times \begin{bmatrix} \phi(0) \\ \phi(\frac{1}{2}) \\ \phi(\frac{3}{2}) \\ \phi(\frac{5}{2}) \\ \phi(\frac{7}{2}) \\ \phi(\frac{9}{2}) \end{bmatrix}, \tag{1.2.25}$$

which can be written as

$$\begin{aligned} \Phi\left(\frac{1}{4}\right) &= A_0\Phi\left(\frac{1}{2}\right), \\ \Phi\left(\frac{3}{4}\right) &= A_1\Phi\left(\frac{1}{2}\right). \end{aligned}$$

We can proceed as follows by using the two similar matrices for all calculation steps until

a desired resolution 2^q is obtained, for $i = 2, 3, \cdots, q$ and $k = 1, 3, 5, \cdots, 2^{j-1} - 1$,

$$\begin{aligned} \Phi\left(\frac{k}{2^i}\right) &= A_0 \Phi\left(\frac{k}{2^{i-1}}\right), \\ \Phi\left(\frac{k}{2^i} + \frac{1}{2}\right) &= A_1 \Phi\left(\frac{k}{2^{i-1}}\right). \end{aligned}$$

Similarly, ψ function values can be calculated from the values of ϕ by utilizing the equation of wavelet

$$\psi(x) = \sqrt{2} \sum_{k=0}^{D-1} g_k \phi(2x - k), \tag{1.2.26}$$

$$\psi(m/2^q) = \sqrt{2} \sum_{k=0}^{D-1} g_k \phi(2m/2^q - k). \tag{1.2.27}$$

To compute the $\phi^{(d)}$ values at the dyadic rationals, Eqn. (1.2.21) is differentiated d times, then we obtain

$$\phi^{(d)}(x) = 2^d \sqrt{2} \sum_{k=0}^{D-1} h_k \phi^{(d)}(2x - k). \tag{1.2.28}$$

On substitution $x = 0, 1, \cdots, D - 1$ in Eqn. (1.2.28) we achieve the system

$$2^{-d} \Phi^{(d)}(0) = A_0 \Phi^{(d)}(0), \tag{1.2.29}$$

the matrix A_0 is already defined in Eqn. (1.2.22). From Eqn. (1.2.29) it is apparent that related to the eigen-value 2^{-d}, $\phi^d(0)$ is the eigen-vector of the A_0 matrix. Now substituting $x = \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \cdots$ into Eqn. (1.2.28), leads to the equation which is in the matrix form as

$$\Phi^{(d)}\left(\frac{1}{2}\right) = 2^d A_1 \Phi^{(d)}\left(\frac{1}{2}\right),$$

where A_1 is given by Eqn. (1.2.24). Similarly, we acquire the $\Phi^{(d)}$ estimations for a desired resolution. Differentiating Eqn. (1.2.26) d times yields

$$\psi^{(d)}(x) = 2^d \sqrt{2} \sum_{k=0}^{D-1} g_k \phi^{(d)}(2x - k),$$

that can be utilized for evaluating the $\psi^{(d)}(x)$ values from the $\phi^{(d)}(x)$ values.

1.2.2.3 Numerical Example

Now we demonstrate collocation method for time-dependent advection diffusion equation Eqn. (1.2.8). On discretizing the Eqn. (1.2.8) with respect to time, we get

$$u^{n+1} - u^n = \nu \partial t u^n_{xx} - a \partial t u^n_x, \tag{1.2.30}$$

Replace $u(x)$ with the approximation as

$$u(x) = \sum_{k=0}^{2^j-1} c^j_k \tilde{\phi}^j_k(x), \tag{1.2.31}$$

$$u_x(x) = \sum_{k=0}^{2^j-1} c^j_k (\tilde{\phi}^{(1)})^j_k(x), \tag{1.2.32}$$

$$u_{xx}(x) = \sum_{k=0}^{2^j-1} c^j_k (\tilde{\phi}^{(2)})^j_k(x), \tag{1.2.33}$$

Using Eqn. (1.2.31), Eqn. (1.2.32) and Eqn. (1.2.33) in Eqn. (1.2.30), we get

$$\sum_{k=0}^{2^j-1} c^{j,n+1}_k \tilde{\phi}^j_k(x) = \sum_{k=0}^{2^j-1} c^{j,n}_k \tilde{\phi}^j_k(x) + \nu dt \sum_{k=0}^{2^j-1} c^{j,n}_k (\tilde{\phi}^{(2)})^j_k(x) - adt \sum_{k=0}^{2^j-1} c^{j,n}_k (\tilde{\phi}^{(1)})^j_k(x). \tag{1.2.34}$$

On solving the above system, we get the solution for $u(x)$. Fig. (1.3) displays the relation between numerical and exact value of the Eqn. (1.2.8) at different time. Fig. (1.4) gives the relation between error between numerical and analytical value versus no. of grid points N. It shows that as N increases, error decreases.

1.2.3 Wavelet Optimized method

In wavelet optimized method, wavelets can be combined with finite difference [88], finite element [89] or finite volume method [44]. Here wavelets are used to generate adaptive grid. Out of these, wavelet optimized finite difference (WOFD) is most famous. WOFD is introduced by Leland Jameson [90]. Instead of representing the solution in the form of wavelet or scaling function expansion, the wavelet transformation can be employed to figure out where the finite-difference grid should be refined or coarsened [48, 91–93]. Wavelets give an ideal procedure for selecting grid where dispersed grids are set in areas of the domain where the solutions are continuous (for example in a fluid mechanics problem, where flow is continuous) and fine framework is put in areas of the region where-ever

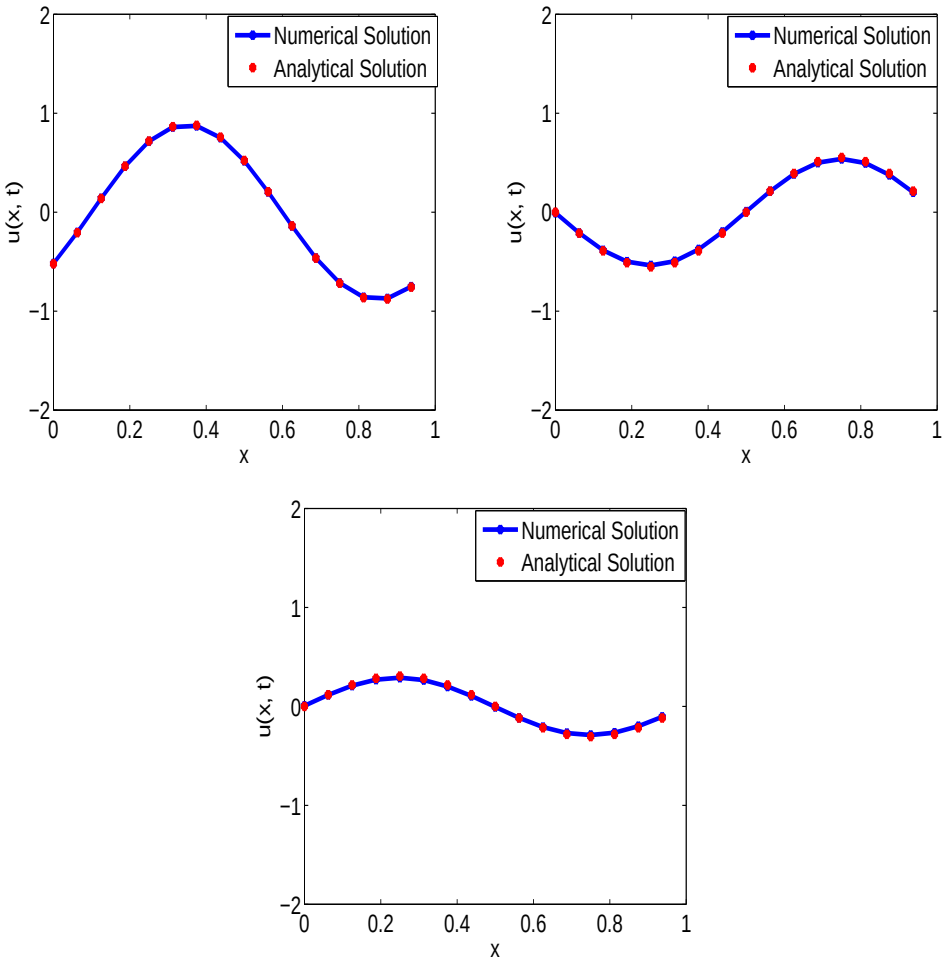


Figure 1.3: Solution of time-dependent linear advection-diffusion problem ($\nu = 0.03$, $a = 1$) at $t = 0.1s$, $0.5s$ and $1s$ using collocation technique.

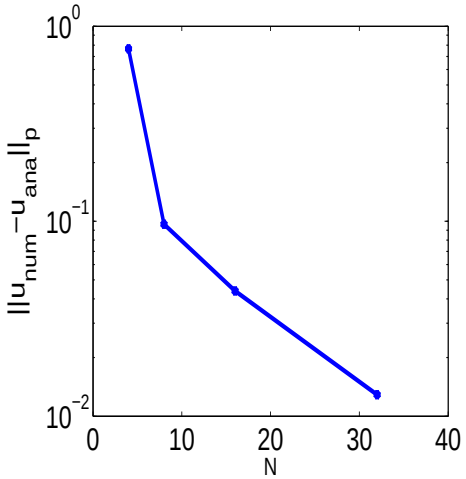


Figure 1.4: Error between numerical and analytical value versus N for collocation technique.

solutions are disorganized. A computationally effective execution of such a technique requires the advancement of a computational situation particularly tailored to adaptive multi-resolution wavelet transformations. This is one such primary reason why numerous scientists purposefully avoided adaptive wavelet techniques. Further, compressing features of wavelets and the capacity to recognize localized arrangements made it a successful candidate to be utilized together with another numerical methods [94].

A fruitful use of the WOFD is suggested by Adechi et al. [95] to numerically decide the neutral curve in the vitality plane of Lewis number-activation. Numerical results of WOFD are reliable with those presented through asymptotic and linear stability studies. Further, Bauer [96] presented the effective utilization of the WOFD in terms of conservation laws. In his investigation, he uses wavelet for defining a hybrid adaptive technique for PDEs in conservative form. Further, Schwarz et al. developed the wavelet-based finite volume technique [89]. In the above analysis Daubechies wavelet has been used for defining the grid. Diffusion wavelet for grid adaptation is used by Goyal and Mehra in [97].

1.2.4 Wavelet meshless method

Numerous issues of practical significance, like fragmentation, crack propagation and large deformations are described by a continuous change in the geometry of the domain under examination. The analysis of this class of problems by traditional finite difference and finite element strategies can be an expensive and cumbersome task. The study of large deformations issues by the finite element approach may require the constant remeshing of the area to elude the breakdown of the computation because of the over mesh distortion. In fact, in problems where only a few meshes are required for estimation, mesh generation could be a definitely more expensive and time-consumable job than developing and arranging the discrete set of equations. For the study of this class of problems, meshless techniques provide an attractive alternate. In computational mechanics, the meshless approach is a talk of the town presently. Several meshless strategies have been planned and accomplished amazing advancement, for example the element-free Galerkin (EFG) approach [98,99], the diffuse element approach (DEA) [100], the mesh-less local Petrov Galerkin (MLPG) approach [101] and the mesh free-point interpolatory approach (PIA) [102]. In the above meshless approaches, the test function and the associated form of the differential equation are distinct. He et al. observed that wavelet functions possess the characteristics of orthogonality and compactness that can overcome the redundancy in calculating other field functions and increase accuracy or lessened the calculation [103]. The fundamental concept of wavelet mesh-less strategy is presented concisely as follows:

In mesh-less approach, the shape function is known as the window, weighing or kernel function, which is indicated by Ω domain as shown in Fig. (1.5). The main feature of the window function is its compactness, where the size of support is determined by the dilation or smoothing length parameter. If in the solution area Ω, N groups of nodes are taken then $U(x) = (u_x, v_x)$ takes the form as

$$U(x) = \sum_{L=1}^{N} \sum_{k \in \mathbb{Z}} \alpha_k^L \phi_k(r_L(X)), \tag{1.2.35}$$

here $X_L = (x_L, y_L)$ are interpolatory nodes and $r_L(X) = \frac{\|X - X_L\|}{r_{def}}$. Here r_{def} stands for the consequence radius of the interpolatory nodes and α_k^L stands for the scaling function coefficients $\phi(r_L(X))$. Scaling functions are similar to the shape functions and α_k^l is similar to the variables that are related to nodes in the traditional mesh-less techniques. It should also be noted that wavelet-based techniques are not the same as conventional mesh-less approaches as they are nonessential to utilize data of nodes for creating the shape function. Therefore, no nodes or meshes are required for approximation [104].

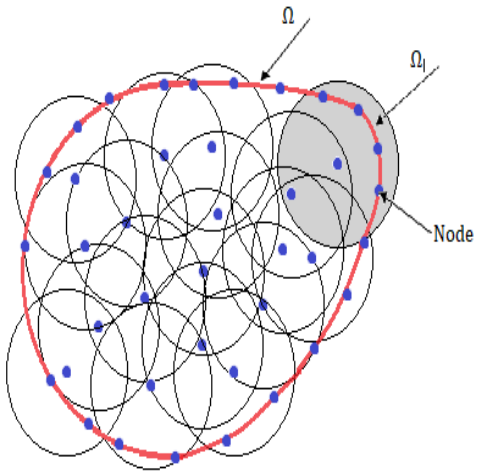


Figure 1.5: Discretization using meshless methods.

The remaining solving strategies for governing equations by using wavelet-based methods are the same as other meshless methods after the development of shape functions. For further detail, refer to [104, 105]. In these studies, Daubechies wavelet is used for solving two dimensional elastic problems. Neither nodes nor meshes are required in this approach as the scaling functions are used directly for approximating the function. In [50], meshfree estimation approach, such as moving least-square approach and the radial basis functions are constructed from the PDEs point of perspectives and approximation of the dispersed

data approximation. [51] is a bundle of research articles that add to the progress of this research field. In [53], a fast adaptive diffusion wavelet approach is proposed for numerically solving PDEs on sphere. Before that, the formulated wavelet-meshfree techniques were restricted to flat geometry. In [54], a fast adaptive spectral graph wavelet technique is proposed for numerically solving PDEs. The main highlights of these wavelets are that they can be built on arbitrary manifolds and their development does not require an original mesh that discretize the manifold. These highlights made spectral graph wavelet and diffusion wavelet a particular option for mesh free techniques on arbitrary manifolds. The constructed approach can be generalized to general manifolds. These papers were significant critical point for the researchers who used wavelets for numerically solving PDEs on general manifolds as diffusion wavelet and spectral graph wavelet have many useful advantages over the extant wavelets.

Meshless methods based on wavelets utilizes the unique features of wavelets for developing formulation for solution. Compared to conventional meshless techniques, it is more brief in shape function development. Moreover, the features of wavelet ensure the efficiency and precision when evaluating the integral, that can be shown numerically. Therefore, developing the field function of meshless approach using wavelet basis function is a new concept in theory and an improvement in algorithm [104].

1.2.5 Wavelet boundary element approach

As compared to wavelet collocation or wavelet Galerkin approach, boundary element approach is usually higher effective for reducing the problem dimension by less PC time, storage and more accuracy. It allows specialists to filter unwanted data and also to target on the concerned portion of the area [106]. Furthermore, construction of boundary element approach generally leads to fully populated matrices. It means that in accordance with square of the problem size, the computational time and storage requirements will tend to increase. Finite element matrices, on the other hand, are generally banded (elements are associated locally) and the storage requirements for system matrices generally increases quite linearly with the problem size. Compression strategies (*e.g.*, multipole expansions, hierarchic matrices or adaptive cross approximation) can therefore be utilized to enhance the boundary element approach. Several people concentrate on extending the primary thought of the wavelet-based programmes for boundary element technique based integral equations for compressing the fully populated matrices and accelerate the computational speed. The primary use of multi-scale estimation of wavelet was examined [107, 108]. The procedure of wavelet boundary element approach for the case of Neumann boundary value

problem has been explained as follows:
We consider the Neumann problem with smooth boundary Γ as follows:

$$\begin{aligned} \Delta u &= 0, \\ \frac{\partial u}{\partial n} &= u_n,\ u \in \Gamma \end{aligned} \tag{1.2.36}$$

here $\frac{\partial u}{\partial n}$ stands for the exterior normal derivative, $u_n \in H^{1/2}(\Gamma)$ is the given function.
In order to ensure that the Eqn. (1.2.36) has solutions, u_n should meet the conditions of consistency.

$$\int_\Gamma u_n ds = 0. \tag{1.2.37}$$

The Green function is stated as

$$G(p, p') = -\frac{1}{4\pi} ln \frac{(x - x') + (y + y')}{(x - x') + (y - y')}.$$

The normal boundary integral equation is

$$u_n(x) = -\frac{1}{\pi x^2} * u_0(x),$$

and the poisson function is

$$u(x, y) = \frac{y}{x^2 + y^2} * u_0(x),\ y > 0,$$

here $*$ means convolution. The integral-kernel is a vigorously singular integral-kernel that can be viewed as the Hadamard's finite-partial integration.

The bilinear model is

$$\tilde{D}(u_0, v_0) = -\int\int_{R^2} \frac{v_0(x) u_0(y)}{\pi (x - y)^2} dx dy,$$

The linear functional is:

$$\tilde{F}(v_0(x)) = \int_R v_0(x) u_n(x) dx,$$

Hence, the Eqn. (1.2.36) is stated as

$$\tilde{D} \in (u_0, v_0) = \tilde{F}(v_0), \tag{1.2.38}$$

Take the approximate value of u_0 as $u^h(x)$,

$$u_0(x) \approx u^h(x) = \sum_{k=-M}^{M} u^h(x_k) \phi_k(x), \tag{1.2.39}$$

where $\phi_k(x)$ is the wavelet function.

On substituting the Eqn. (1.2.39) into Eqn. (1.2.38), we achieve the discrete solving function

$$\sum_{k=-M}^{M} u^h(x_k)\tilde{D}(\phi_k, \phi_l) = \tilde{F}(\phi_l),$$

here $-M \leq l \leq M$. The associated form of matrix is

$$AX = b, \tag{1.2.40}$$

where,

$$\begin{aligned} A &= (\tilde{D}(\phi_k, \phi_l))_{(2M+1)\times(2M=1)}, && (1.2.41)\\ X &= (u^h_{-M}, u^h_{-M+1}, \ldots, u^h_M)^T, && (1.2.42)\\ b &= (\tilde{F}(\phi_{-M}), \tilde{F}(\phi_{-M+1}), \ldots, \tilde{F}(\phi_M)). && (1.2.43) \end{aligned}$$

Therefore, by solving a matrix function of Eqn. (1.2.40) the associated Neumann boundary value problem could be solved.

On solving the neumann boundary value problem, it has been found that the discretization of the boundary integral condition is a key advance of the boundary element approach in resolving engineering issues. This is because of the belief that the liability of the boundary element approximation is specifically linked to the model of the discrete boundary element where a suitable mesh should be used to accurately present the original problem in both its geometry as well as in state variables [109]. Despite that, in an analysis of traditional boundary element technique, mesh configuration relies on an examiner's intuition or experience. In case, an initial solution is abandoned, at that point a completely new data class showing a new mesh should be prepared. Clearly the analysis of traditional boundary element technique is very expensive and tedious task and there is no assurance that the end result is adequately precise.

Neumann problem is classified as one of the easiest problems for boundary element method. In light of the above process, Spasojevic et al. recommended that by utilizing the orthogonal Haar wavelet, wavelet boundary element method acquire inadequately boundary element matrices that emerged from Laplace condition with mixed boundaries [107]. In addition, Schneider et al. used bi-orthogonal wavelets to develop significantly sparser matrices than those acquired using the traditional hair wavelet [108]. They also noticed that a sparser boundary element lattices were prompted by the highest order of vanishing moments (decides the smoothness of wavelet basis).

In recent times, several types of wavelet boundary element techniques by utilizing the distinctive wavelets were recommended for compression of boundary element conditions for creating the fast boundary element method. Abe and Koro used an orthogonal Haar wavelet and non-orthogonal spline for developing a h-version hierarchal boundary element method for 2-D Laplace equation, separately [110, 111]. Taking into account the ultimate goal of reducing memory consumption and saving estimation time, the wavelet estimation coefficients were deducted to frame additional boundary element matrices. Harbrecht et al. proposed a bi-orthogonal wavelet estimate for the mixed approach of finite element and boundary element method to solve an external Dirichlet boundary value problem [112]. For the purpose of additional speed up the wavelet BEM, Cabeleiro and Gonzalez provided a parallel iteration solver, similar to the usually used parallel estimation of the conventional boundary element method [113]. Tausch stretched out p-version wavelet boundary element technique to deal with tokes flow utilizing quasi-vanishing moments of wavelet [114]. Abe and Koro suggested an useful deterministic method of optimum threshold constant by rejecting the estimated wavelet coefficients without affecting the accuracy [115]. With the help of BEM, Bucher et al. proposed a rapid method for immediately addressing problems with numerous load cases [116]. Eppler et al. have extended the bi-orthogonal wavelet BEM to the problems of shape optimization [117]. Pursuing Wrobel's research [116], Bucher et al. proposed another computational system to execute the necessary 2-D wavelet transformation in sub domains [117, 118]. The wavelet boundary element approach was connected to the fluid dynamics by Ravnik et al. [119]. They suggested that the governing equations in the composition of speed-vorticity and acquired solutions of the consequent system of equation by using compression of the Haar wavelet matrix. They also expanded the basic idea to the boundary element and finite element hybrid techniques [120]. Xiao et al. used Daubechies wavelet to solve 2-D Laplace condition and wavelet boundary element matrices estimated by fast Fourier transformation [121]. For solving the diffusion equation, Barmada expanded Daubechies wavelet based wavelet boundary element approach [122]. Ravnik et al. expanded Daubechies wavelet boundary element method and conventional finite element method for solving diluted flows of particles. [123]. Herbrecht and Eppler have extended the bi-orthogonal spline wavelet boundary element method to optimize the figure for external 2 and 3-dimensional electromagnetic shaping [124]. Attarnejad and Ebrahimnejad extended the Daubechies wavelet to fundamental problems of elasticity [125]. Xiao et al. epitomized the different wavelet boundary element compression methods, showed their achievements with regard to practical problems with acoustic dispersion, capacitance extraction, stokes flow and suggested a posteriori compressing technique for wavelet boundary element matrices [126]. They subsequently presented a wavelet boundary element technique with an ascending function and showed

that the complexity for new strategy never exceeds [127] and stretched out the technique to huge scale stokes flow [128]. Furthermore, Tausch and Xiao have proposed a fast-wavelet multipole technique to take full advantage of the fast multipole and wavelet compression techniques [129]. Wen et al. have improved the quasi-vanishing moment wavelet to 3-D electrostatic feature analysis [130]. Ebrahimnejad et al. proposed a Daubechies wavelet boundary element technique for inspecting 2-D elasticity difficulties [131]. To embed the fast wavelet boundary element approach to selectable CPU cores using Intel's Open Mp library, Reinauer et al. developed a boundary element approach multilevel (hierarchical) wavelet compression strategy [132]. Randrianarivony and Harbrecht improved the wavelet boundary element approach to inspect possible surface charge that hikes through salvation continuum models [133].

In summary, examination, further research and issues of wavelet boundary element approach lies in the four basic perspectives. The first is the competent wavelet boundary element strategy for estimating p version or h version at different levels using scaling functions or nascent wavelet or scaling functions at different levels. Second, wavelet determination with great properties is still a difficult task to explain explicit problems. Third, as we aware about, the coefficients of the wavelet estimation are mainly non-zero however majority of smooth areas leads to small wavelet estimation coefficients. The sensible truncation strategy is therefore the way to compress BEM matrices without affecting precision. At last, the composition of wavelet boundary element technique and the conventional fast algorithm of boundary element technique (*e.g.*, parallel computing strategy, fast-multipole strategy) or finite element method is a vital future course.

1.2.6 Miscellaneous numerical methods based on wavelet

In addition to the above mentioned numerical strategies, a few well-examined problems or approaches demonstrate a new strength on combining with wavelet, for example, the non-linear terms, non-linear boundaries, method of optimizing numerical techniques and adaptivity analysis in time. Now, we are attempting to present some of the remarkable numerical methods based on hybrid wavelets that are not included in the above classification. As an effective supplement to the present classification, they are planned to execute.

1.2.6.1 Lagrangian wavelet techniques

The traveling wavelet strategy [134] is an intriguing extension of the adaptive wavelet techniques, where wavelet size and wavelet position can change persistently with time. Despite

that, because of the wavelet collision problem, it turns out that this appealing technique is uncertain when testing the non-linear problems. Bergdorf and Koumoutsakos [41] used Lagrangian particle approach with multi-resolution wavelet-based adaptive grid to overcome the wavelet collision issue. The technique is an extension of the adaptive wavelet collocation strategy [91,135] by associating it with particle techniques [136]. This approach prompts the development of new particle methods with adaptive multi-resolution capabilities. Haefele et al. [137] designed an application for plasma simulations using particle techniques to determine the Vlasov equation and grid adaptation based on wavelets.

1.2.6.2 Space-Time wavelet techniques

A large part of the numerical strategies that are gradually adapting, both wavelet-based and non-wavelet-based, depend on spatial adaptation, while comprehensively adjusting the time stepping, either to ensure steadiness or to address the time-integration flaw. It makes these strategies far from ideal for issues which are all the while irregular in both time and space. To overcome this problem, various adaptive time-stepping methods have been pursued for space-adaptive discretization of PDEs. In terms of adaptive wavelet schemes, Mallat et al. [138] proposed a time step dependent on the scale for the first ever. They used this strategy to the Burger's equation and linear parabolic equations. To overcome this constraint, spatially variable time steps have been introduced for adaptive multi-resolution algorithms [139–141] and employed in composition with time-step control [140] for the compressible Euler conditions. The fundamental idea behind these schemes is that the time step is enforced by the stability condition at the optimal level of determination for explicit time integration approaches, while the time step can be extended in the areas of coarser resolution without degrading the demand for stability. The incomplete values between two scales are interpolized in time at the boundaries.

Regardless of the various advantages of local time stepping schemes (particularly with regard to multi-resolution methods or adaptive wavelet techniques), they are still dependent on the traditional time-stepping approach and hence this results in the aggregation of errors in time, despite the fact that flaws in spatial time integration are managed at all times. Kevlahan et al. [142] proposed a simultaneous space-time adaptive wavelet collocation scheme to address this problem, where the problem is solved in a computational space-time field, that usually adapting to space and time resolution to adequately resolve the solution's spatially and temporarily intermittent forms. This technique also allows the global time-integration error to be controlled by building a nearby optimal grid for the entire space-time arrangement. The effectiveness and accuracy of the approach is demonstrated by the combination of two-dimensional vortex [142] and two dimensional

homogeneous turbulence [143, 144] problems. The space-time approach utilizes generously less space-time grid points (in a few cases upto 20 time less) and rapid as compared to a similar adaptive time-marching approach of wavelets while achieving the identical universal accuracy. The principle downside of the simultaneous space-time approach is a significantly large memory demand that could be reduced by solving the problem using temporal slices, as Kevlahan et al. describe [142].

The split-step scheme, also known as the beam propagation method, is a well-known time-stepping scheme for finding the solution of non-linear PDEs in optical applications. It is traditionally used in association with a spatial discretizing scheme based upon a Fourier spectral scheme in which case it is called the Fourier split-step scheme or FFT split-step. If the split-stepping is used with wavelet based grid adaptation scheme then the resulting scheme is known as the wavelet split-step scheme.

1.2.6.3 Wavelet techniques on general manifolds

To develop wavelet based methods on general manifolds, the transformation of the wavelet should be generalized to non-tensorial meshes produced to conform to arbitrary manifolds. For constructing the wavelets on general manifolds, various techniques have been proposed. In [44, 145] wavelet bases are developed in light of certain form of manifolds which can be presented as separate union of smooth parametric images of a standard cube. It has numerous drawbacks from a practical viewpoint as its formation depends entirely on smooth parametrization of the unit cube. This issue is settled in [46], where finite element supported wavelet bases respecting an arbitrary initial triangularization are developed. Wavelets are developed on the sphere (a specific kind of manifold) in [47]. Swelden and his collaborators developed the second-generation wavelet by using the lifting technique [146] that has various constructive benefits over the traditional wavelets.

In spite of vast literature available, the subject of wavelet based methods for numerically solving PDEs on general manifolds is yet in its emerging phase. For solving PDEs on the sphere, an adaptive wavelet collocation method has been developed by M. Mehra and N. Kevlahan by using second generation spherical wavelets [48] in 2008 and this work was further extended for elliptic problems in [49]. The fact that curvelet and spectral graph wavelet provide us wavelet transforms on nontensorial meshes has motivated us to use these for solving PDEs.

1.3 Organisation of the thesis

The research work exhibited in this thesis is an endeavor to create a few techniques for numerically solving PDEs based on wavelets and its variants. The proposed work encapsulates these approaches which are weaved into seven chapters. Chapter 1 is the current chapter where we discussed the literature and introduce the most famous wavelet *i.e.*, Daubechies wavelet.

In chapter 2, wavelet optimized upwind conservative method (WOUC) has been developed for solving traffic flow problems. Daubechies wavelet has been used to decide where the grid needs to be refined or coarsened for representing the solution optimally. Daubechies wavelet has been extensively used for solving PDEs, but in the 2nd chapter we explored the use of these wavelets to solve real-life problems *i.e.*, traffic flow problems. These wavelets are the most famous in the class of first generation wavelets. It has been shown that how a dynamic adaptive wavelet technique works in a simple way with local singularities of the solution of the traffic-flow problems.

In chapter 3, a set of Matlab routines for the second generation wavelet transformation and inverse wavelet transformation on the space $\mathcal{L}_2([a,b])$ is presented which are given in the appendix. These wavelet transforms are further used for computing the wavelet and scaling function values ($\psi(x)$ and $\phi(x)$ respectively). The second-generation wavelet so constructed has been used for the adaptive-grid generation. In the collection of the Matlab routines, three Matlab functions namely, `Reconstruction_testing.m`, `AdaptiveGrid_standard_testing.m` and `AdaptiveGrid_modified_testing.m` are provided for generating the adaptive grid. After constructing the second generation wavelet, second generation wavelet optimized finite difference method (SGWOFD) is developed for solving the Burger's equation with distinct boundaries. The viscid Burger's equation is considered with Dirichlet, periodic, Robin and Neumann's boundaries. For the approximations of the differential operators, central finite difference scheme has been used and Crank Nicolson's technique has been used for integrating time. Numerical solutions have been optimized on an adaptive grid which is generated using the second-generation wavelet. The second-generation wavelet has the beauty that its construction is not affected by the boundaries. The method's convergence has been checked for each test problem. The computational time carried out by SGWOFD has been computed for each test problem and has been compared with the computational time carried out by the finite difference technique on a uniform grid. It has been revealed that SGWOFD is highly efficient.

In chapter 4, spectral graph wavelet optimized finite difference method (SPGWOFD) has

been proposed for solving Burger's equation with distinct boundary conditions. Central finite difference approach is utilized for the approximations of the differential operators and the grid on which the numerical solution is obtained is chosen with the help of spectral graph wavelet. Four test problems (with Dirichlet, Periodic, Robin and Neumann's boundary conditions) are considered and the convergence of the technique is checked. For assessing the efficiency of the developed technique, the computational time carried out by the developed technique is compared to that of the finite difference method. It has been observed that developed technique is extremely efficient.

In chapter 5, curvelet optimized finite difference method (COFD) has been developed which is used for solving PDEs. The technique uses finite difference approximations for differential operators involved in the PDEs. After the approximation, curvelet is used for the compressing the differential operators and thus the dyadic powers of the operators required to solve PDEs are calculated quickly and efficiently. Furthermore, compression and reconstruction errors for the curvelet have been tested with respect to different parameters. The developed method has been applied on five test problems of different nature. For each test problem the method's convergence has been verified. Moreover, to measure the efficiency of the proposed technique the computational time carried out by the method is compared with the computational time carried out by finite difference technique. It is observed that the proposed method is computationally very efficient.

In chapter 6, a dynamically adaptive curvelet technique has been developed for solving non-linear Schrödinger equation. Central finite difference technique is used for approximating the one and two dimensional differential operators and radial basis functions (RBFs) are used for approximating the differential operators on the sphere. The grid on which the equation is solved, is obtained using curvelets. For one and two dimensional non-linear Schrödinger equation, the computational time carried out by the proposed technique is compared with the computational time taken by the finite difference technique. Moreover, the problem on the sphere has been considered for which, the computational time carried out by the RBF collocation technique is compared to that of computational time taken by the proposed technique. The developed technique is found to perform better in terms of computational time, for example on sphere computational effort reduces by 4 times using proposed method.

Chapter 7 proposes a dynamically adaptive curvelet technique for solving PDEs on the general manifolds. The closest point form has been employed for approximating the Laplacian-Beltrami (∇^2) operator. Curvelets are used to obtain the grid on which the equation is solved. The computational time carried out by the developed technique is compared with the computational time carried out by the closest point approach and the

proposed technique is found to perform better in terms of computational time. To the best of our insight, ours is the first attempt to exploit the useful features of curvelet for solving PDEs on general manifolds. The developed technique has been applied on 3 test problems namely reaction-diffusion equation on a sphere, Schnakenberg model evolving on the surface of ellipsoid and the well-known Fitzhugh-Nagumo equations. The numerical outcomes reveal that the proposed method can precisely catch the development of the localized patterns on all the scales and the arrangement of nodes are adapted accordingly. The method's convergence has also been checked.

The overall concluding observations of this study and few significant directions for the future scope are given in the end of chapters.

Chapter 2

First generation wavelet based numerical methods for solving PDEs

Mathematical solutions of PDEs modeling many real life phenomenon exhibit singularities [147,148] and these singularities are of physical relevance. For example these singularities represent the concentration of stress in elasticity [149] and boundary layer in viscous flows [150] etc. Therefore it is required that these singularities be resolved accurately by numerical methods. PDE's numerical solution is estimated when solution is approximated at discrete set of points of grid. Higher collection of grid points are needed for detecting all the characteristics of the solution, but it increases the cost of both computing and storage. The set needed to discover all of the solution's features may in some cases surpass the practical restraints. To address this issue, we are working on an adaptive grid arrangement that will continue to change over time according to the evolution of the PDE's numerical solution. Rather than taking a large set of points of grid, extra points of grid are included only in the regions where the PDE's numerical solution has sharp characteristics in the adaptive node framework [151]. The process of adding the points of grid, uses the information gained during a given step of a numerical method.

Adaptive mesh refining (AMR) is the extremely common adaptive technique [152]. In AMR, the coarsest cartesian grid covers the whole computational region. On the basis of some a-posteriori principle, individual grid-cells are chosen to refine when going from one stage of the numerical method to the another stage. For instance, the model might be mass per unit cell, so high density areas are more profoundly resolved. Undoubtedly, these strategies are computationally efficient over their corresponding non adaptive counterparts. However, the mathematical theory for these methods (*e.g.*, the convergence rate of the adaptive technique describing the contrast between the exactness and effectiveness of the method's complexity) is not explained precisely [153].

However, the benefit with wavelet based adaptive methods is that there are sound theoretical results that can answer basic questions such as the adaptive method's convergence rate. In this chapter Daubehies wavelet has been used for adapting the grid, as the adaptive grid clearly has advantages over the use of a static grid. The method presented in

this chapter falls under the category of wavelet optimized methods. Different numerical schemes, namely upwind non-conservative, upwind conservative, Lax-Friedrichs, Lax-Wendroff, MacCormack and Godunov are applied and compared on traffic flow problems. The best scheme namely upwind conservative is used for wavelet optimized approach to solve the traffic flow problem for two distinct situations.

2.1 Daubechies wavelet based adaptive grid

Daubechies wavelet based grid generation is based on the following theorem,

Theorem 2.1.1 *For wavelet ψ_k^j, let $M=\frac{D}{2}$ be the no. of vanishing moments and suppose that $f \in C^M(\mathbb{R})$. At that point the coefficients of wavelet d_k^j degenerate as follows*

$$|d_k^j| \leq C_M 2^{-j(M+\frac{1}{2})} \max_{\xi \in I_{j,k}} |f^{(M)}(\xi)|$$

where C_M is a constant and is not dependent on k, j and $I_{j,k}$=Supp$\{\psi_k^j\} = [k/2^j, (k+D-1)/2^j]$.

From above theorem, it can be interpreted that at any scale and location, the function's oscillations could be captured by the wavelets. Let a function $f(x)$ is given for $x \in Z$, where Z is an interval, then function $f(x)$ is broken down into an arrangement of coefficients of wavelet which relies upon two parameters, scale parameter and region parameter, called as d_k^j, here j is the scale and k is the region parameter. If the wavelet's coefficient magnitude is large, *i.e.*, $|d_k^j| > \epsilon$, here ϵ is a threshold parameter (user's selected parameter), then from the above theorem we deduce that f has non smoothness at this location and hence grid can be refined at this location. Now we demonstrate the grid generation technique. Suppose X^c is the current coarsest grid and $f(x_j)_{j \in X^c}$ is known. We will fix a resolution p such that the initial grid is $a : \frac{1}{2^p} : b = X^F$ (where X^F is called the finest possible grid of the required domain $[a, b]$). Using linear interpolation, we compute expanded $f(x)$ on the finest grid X^F. There are also other techniques available for interpolation *i.e.*, cubic interpolation, spline interpolation etc. But we choose the simplest one *i.e.*, linear interpolation. We then apply fast wavelet transformation (FWT) to this expanded $f(x)$ and calculate the coefficient of the discrete scaling function and the coefficient of the wavelet function. All the points associated to the discrete scaling function coefficients will remain intact. Wavelet function coefficients will be larger where the solution is discontinuous and small in the area of smoothness. The grid-points will remove from the area where $|d_k^j| < \epsilon$ (threshold chosen by user) and the point $|d_k^j| \geq \epsilon$ is kept intact. Therefore when the

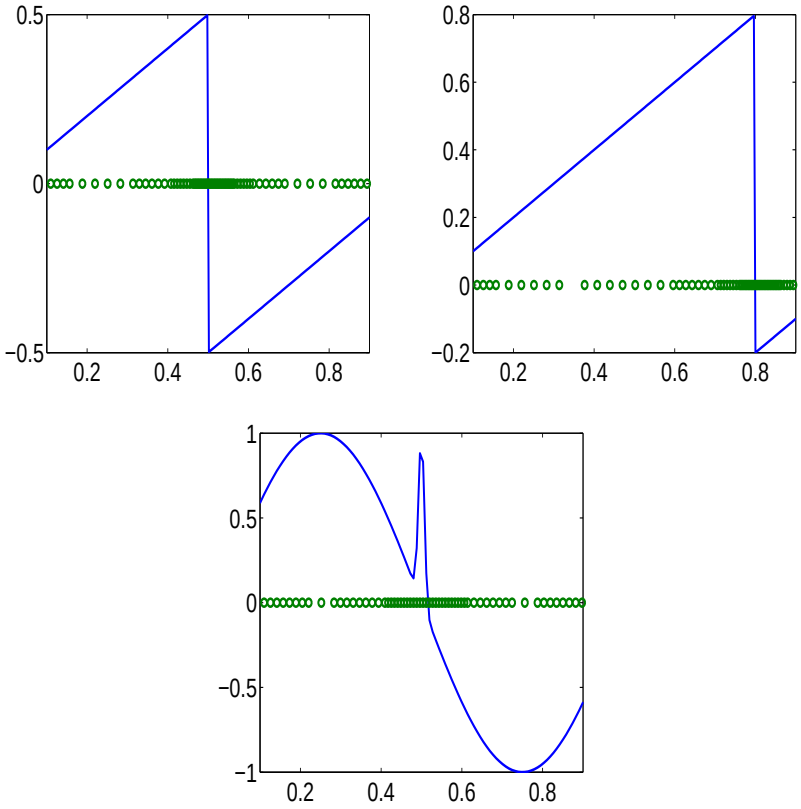


Figure 2.1: The adaptive grid for (a) Sawtooth function having discontinuity at $x = 0.5$ (b) Sawtooth function having discontinuity at $x = 0.8$ (c) $f(x) = \sin(2\pi x) + \exp(-10^4(x - 0.5)^2)$.

$|d_k^j|$ value is large then, we add $1/2^{j+1}$ spacing points about $k/2^j$ position. Furthermore, we found it feasible for distributing points evenly over the whole interval in the light of the Thm. 2.1.1 as the larger gradient could be located anywhere within the associated wavelet's support. Fig. (2.1) displays the adaptive grid generated for distinct functions using the above-mentioned technique. Algorithm of Daubechies wavelet based adaptive grid generation is as follows:

- We will fix a resolution p such that the initial grid is $a : \frac{1}{2^p} : b = X^F$ (where F is called the finest possible grid of [a, b]).
- Compute $\{f(x_j)\}_{j \in X^F}$ from $\{f(x_j)\}_{j \in X^c}$ using interpolation.
- Apply FWT on the expanded $f(x)$ for obtaining the coefficients of scaling and wavelet function.
- Start with X^J.
- Keep intact all the points that corresponds to the scaling functions.

- Keep the points where $|d_k^j| \geq \epsilon$ intact.
- Delete all the other points.

2.2 Traffic flow problem formulation

Enthusiasm in modeling traffic flow has been increased these days due to burgeoning traffic jams. Traffic flow is an analysis of travelers interactions (it includes cyclists, pedestrians, drivers and their automobiles) and infrastructures (which includes signals, highways and devices for traffic control) in order to understand and develop an optimum transport structure with efficient traffic movement and minimize traffic jam issues [154, 155]. Nowadays, traffic flow is one of the main societal and economical problem related to transportation in industrialized countries. Traffic problem include: installation of traffic light or stop signs; cycle timing of traffic lights; where to construct flyovers, whether to change a two way street to a one way street, number of lanes on a road; where to construct overpasses, exits and entrances. In particular, the main aim is to study the traffic phenomena with an objective of eventually making decisions which may palliate congestion, reduce accidents, maximize traffic flow, minimize automobile exhaust pollution etc. Depending upon the communication of a larger no. of vehicles, traffic acts in a non-linear and convoluted way. Ideally, if the vehicle flow behavior can be assumed exactly, then in theory the comprehensive traffic throughput along an area of road could be magnified by flow adjustment in critical areas. This is of the specific interest to high-density regions, due to the accidents, high-volume crest time density or one or more road lanes being closed. In the course of recent decades, many analysts have proposed different theories and models of traffic flows to explain the traffic evolution rules [156–159]. Traffic can be considered as a compressible flow of specific velocity and density, the behavior of which is described by the equation of continuity and the equation of state (density-flow relationship of equilibrium).
Now, we consider a traffic-flow of the cars with just a single lane on a capacious road. Suppose that $\rho(x,t)$ stands for the cars density (in cars per kilometer) and $f(x,t)$ be the traffic flow rate where $x \in \mathbb{R}$ represents the position and $t \geq 0$ denotes the time. At time t, the no. of cars in the interval (x_1, x_2) is $\int_{x_1}^{x_2} \rho(x,t)dx$ and $v(x,t)$ denotes the speed of cars at time t and position x. At time t, the no. of cars that pass through x (in length of unit) is $\rho(x,t)v(x,t)$. By using the conservation law (shown in Fig. (2.2), the following equation has been obtained:

$$\frac{d}{dt}\int_{x_1}^{x_2} \rho(x,t)dx = \rho(x_1,t)v(x_1,t) - \rho(x_2,t)v(x_2,t). \tag{2.2.1}$$

On integrating the Eqn. (2.2.1) w.r.t time and assume that v and ρ are integrable functions, we get

$$\begin{aligned}\int_{t_1}^{t_2}\int_{x_1}^{x_2}\frac{\partial}{\partial t}\rho(x,t)dxdt &= \int_{t_1}^{t_2}\rho(x_1,t)v(x_1,t)-\rho(x_2,t)v(x_2,t)dxdt,\\ &= -\int_{t_1}^{t_2}\int_{x_1}^{x_2}\frac{\partial}{\partial x}\rho(x,t)v(x,t)dxdt. \qquad (2.2.2)\end{aligned}$$

Since $x_1,\ x_2 \in \mathbb{R},\ t_1,\ t_2 > 0$ are random, we achieve

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho v}{\partial x}=0,\ x\in\mathbb{R},\quad t>0. \qquad (2.2.3)$$

It must be accompanied by an initial condition of the form $\rho(x,0)=\rho_0(x),\ x\in\mathbb{R}$. In

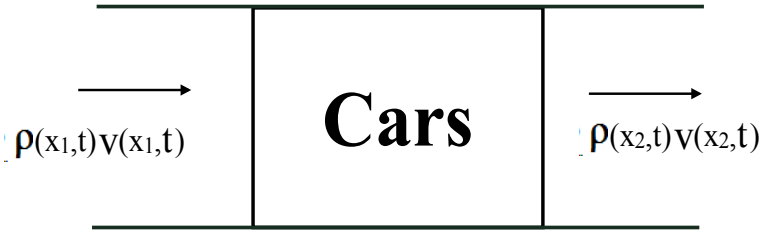


Figure 2.2: Flow of cars.

heavy traffic we will drive at slow speed but on a highway we might ideally want to drive at a speed v_{max} and we will stop ($v=0$) in a fullback where the cars are packed *i.e.*, ($\rho=\rho_{\max}$). Eqn. (2.2.3) contains two unknown functions v and ρ and we assume the velocity v is a function of ρ. The relation between v and ρ is defined by different models which are discussed as below:

- Lighthill-Whitham Richards Model:

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho v(\rho)}{\partial x}=0$$

,

$$v(\rho)=v_{\max}(1-\frac{\rho}{\rho_{\max}}),\qquad 0\le\rho\le\rho_{\max}. \qquad (2.2.4)$$

By using the transformation $u=1-2\frac{\rho}{\rho_{\max}}$, the simplified form of this equation is given as below:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}=0. \qquad (2.2.5)$$

- Greenberg Model:

In the Greenberg model, a logarithmic relation is assumed between speed and density. It is presumed that vehicle's speed for low densities could be very large for this particular model.

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho v(\rho)}{\partial x} &= 0, \\ v(\rho) &= v_{\max} \ln \frac{\rho_{\max}}{\rho}, \quad 0 < \rho \leq \rho_{\max}. \end{aligned}$$

This implies,

$$\frac{\partial \rho}{\partial t} - v_{\max} \frac{\partial \rho \ln \rho}{\partial x} = 0.$$

The model can be analytically derived, that's why the model has become very popular. However, main disadvantage of this model is that density becomes zero when speed reaches infinity. Hence this cannot be used for predicting speeds at lower densities. It shows that the model has an inability for predicting the speeds at lower densities.

- Underwood's Exponential Model: This model has been put forward to resolve the limitations of the model of Greenberg. Underwood proposed an exponential model idea which is given as follows:

$$v(\rho) = v_{\max} \exp^{-\frac{\rho}{\rho_0}},$$

here ρ_0 stands for the optimal density *i.e.*, that density which corresponds to the maximum flow. The major limitation of the model is that velocity becomes zero as density goes to infinity. It demonstrates the model's inability to predict higher density speeds.

- Payne-Whitham Model:
Whitham (1974) and Payne (1971) proposed, one of the foremost non equilibrium traffic-model which we call as Payne-Whitham(PW) model. Two PDEs are used by PW to represent the traffic dynamics.

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho v(\rho)}{\partial x} &= 0, \\ \frac{\partial \rho v}{\partial t} + \frac{\partial}{\partial x}(\rho v^2 + p(\rho)) &= 0. \end{aligned}$$

Traffic "mass" conservation is described by the first PDE and the second attempts to mimic the equation of fluid momentum. The gas particles flow is emulated by

this model. The above equations are actually referred to as the Euler equation of gas dynamics having pressure $p(\rho) = a\rho^{\lambda},\ a > 0,\ \lambda \geq 1$. PW model has a limitation that there might be solution occurs when the speed v is not positive [160].

- Aw Rascle Model:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial \rho v(\rho)}{\partial x} &= 0, \\ \frac{\partial}{\partial t}(\rho v + \rho p(\rho)) + \frac{\partial}{\partial x}(\rho v^2 + \rho v p(\rho)) &= 0. \end{aligned}$$

In an attempt to improve the Payne-Whitham model, Rascle put forward this model. It has been followed from microscopic models [161]. The model is designed to illustrate the anisotropic traffic behaviour.

In this chapter, LWR model has been considered [162,163]. The first model which was used to express the one-dimensional and unidirectional traffic flow on a highway is LWR model. On substituting Eqn. (2.2.4) in Eqn. (2.2.3), we obtain the following equation.

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}\left(\frac{v_{\max}}{\rho_{\max}}\rho\left(\rho_{\max} - \rho\right)\right) = 0,\ x \in \mathbb{R},\ t > 0. \tag{2.2.6}$$

Since the above equation indicates the conservation of number of cars, that's why this equation is called the conservation law. By bringing it in a dimensionless form, Eqn. (2.2.6) can be written in a simplified way. Assume that τ and L stands for the typical time and length, such that, $\frac{L}{\tau} = v_{\max}$. Introduce $x_s = \frac{x}{L},\ t_s = \frac{t}{\tau},\ u = 1 - \frac{2\rho}{\rho_{\max}}$, we have

$$\frac{\partial \rho}{\partial t} = \frac{1}{\tau}\frac{\partial}{\partial t_s}\left[\frac{\rho_{\max}}{2}(1-u)\right] = -\frac{\rho_{\max}}{2\tau}\frac{\partial u}{\partial t_s}, \tag{2.2.7}$$

$$\begin{aligned} \frac{\partial}{\partial x}\left[v_{\max}\rho(1 - \frac{\rho}{\rho_{\max}})\right] &= \frac{1}{L}\frac{\partial}{\partial x_s}\left[v_{\max}\frac{\rho_{\max}}{2}(1-u)\frac{1}{2}(1+u)\right], \\ &= -\frac{\rho_{\max}}{2\tau}\frac{\partial}{\partial x_s}\left(\frac{u^2}{2}\right). \end{aligned} \tag{2.2.8}$$

In the place of (x_s, t_s) we write (x, t), therefore we achieve

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) &= 0,\ x \in \mathbb{R},\ t > 0, \\ u(x,0) &= u_0(x),\ x \in \mathbb{R}, \\ \text{with } u_0 &= 1 - 2\frac{\rho_0}{\rho_{max}}. \end{aligned} \tag{2.2.9}$$

For the tailback ($\rho = \rho_{max}$) we have $u = -1$ and $u = 1$ when the highway is empty. Eqn. (2.2.9) is nothing but inviscid Burger's equation.

The traffic flow models are non-linear, so in this chapter, the study has been conducted to discretize the non-linear equation.

$$\begin{aligned} \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} &= 0, \quad x \in \mathbb{R}, \quad t > 0 && (2.2.10) \\ u(x,0) &= u_0(x), \quad x \in \mathbb{R} && (2.2.11) \end{aligned}$$

Riemann Problem:- The initial value problem (IVP) (2.2.10) having discontinuous initial profile of the form

$$u_0(x) = \begin{cases} u_L & ; \ x \leq 0, \\ u_R & ; \ x > 0. \end{cases} \tag{2.2.12}$$

is known as the Riemann problem. The case when $u_L > u_R$ has been considered as a 1st test problem, *i.e.*, $u_L = 1$ and $u_R = 0$. In the traffic flow interpretation, density of vehicles in $x > 0$ is large as compared to the density when $x \leq 0$. The discontinuous function

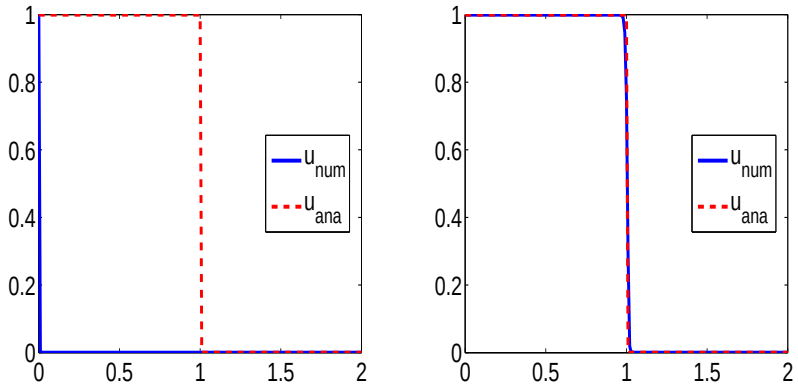
$$u_0(x) = \begin{cases} u_L & ; \ x \leq st, \\ u_R & ; \ x > st. \end{cases} \tag{2.2.13}$$

where $s = \frac{1}{2}(u_L + u_R)$, is a weak solution of the above problem. It could be verified that this is the unique weak solution of the problem. The whole situation alters for the opposite case *i.e.*, $u_L = 0$ and $u_R = 1$. For this case, the solution of the traffic flow problem is:

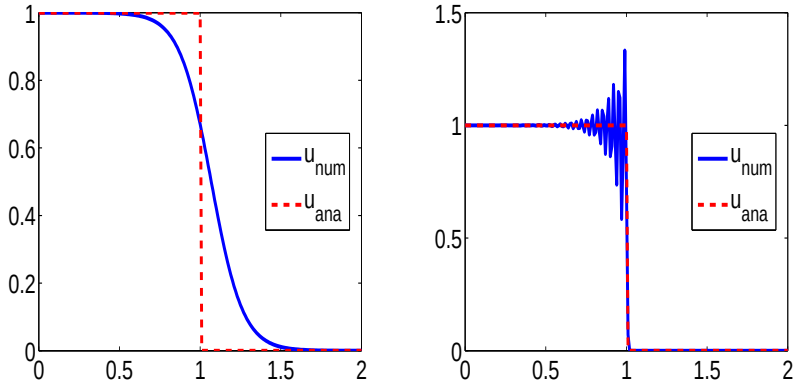
$$u_0(x) = \begin{cases} 0 & ; \ x \leq 0, \\ \frac{x}{t} & ; \ 0 < x \leq t, \\ 1 & ; \ x > t. \end{cases} \tag{2.2.14}$$

2.3 Numerical methods

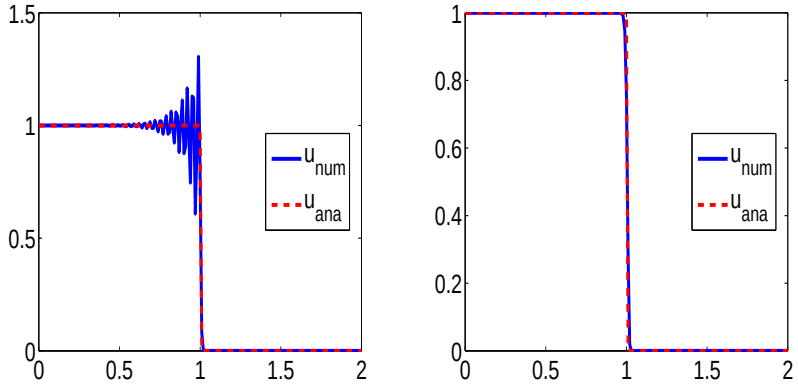
The problem formed in the section (2.2) is solved by using upwind non-conservative, upwind conservative, Lax-Friedrichs, Lax-Wendroff, MacCormack and Godunov scheme for comparison purpose. Out of these schemes, best scheme is used for wavelet optimized method.



(a) Upwind nonconservative (b) Upwind conservative



(c) Lax-Friedrichs (d) Lax-Wendroff



(e) MacCormack (f) Godunov

Figure 2.3: Exact solution and Numerical solution for the traffic flow problem by using the above schemes.

Upwind nonconservative:- This scheme reads as follows

$$u_i^{n+1} = u_i^n - u_i^n(u_i^n - u_i^{n-1})\frac{\Delta t}{\Delta x}, \tag{2.3.1}$$

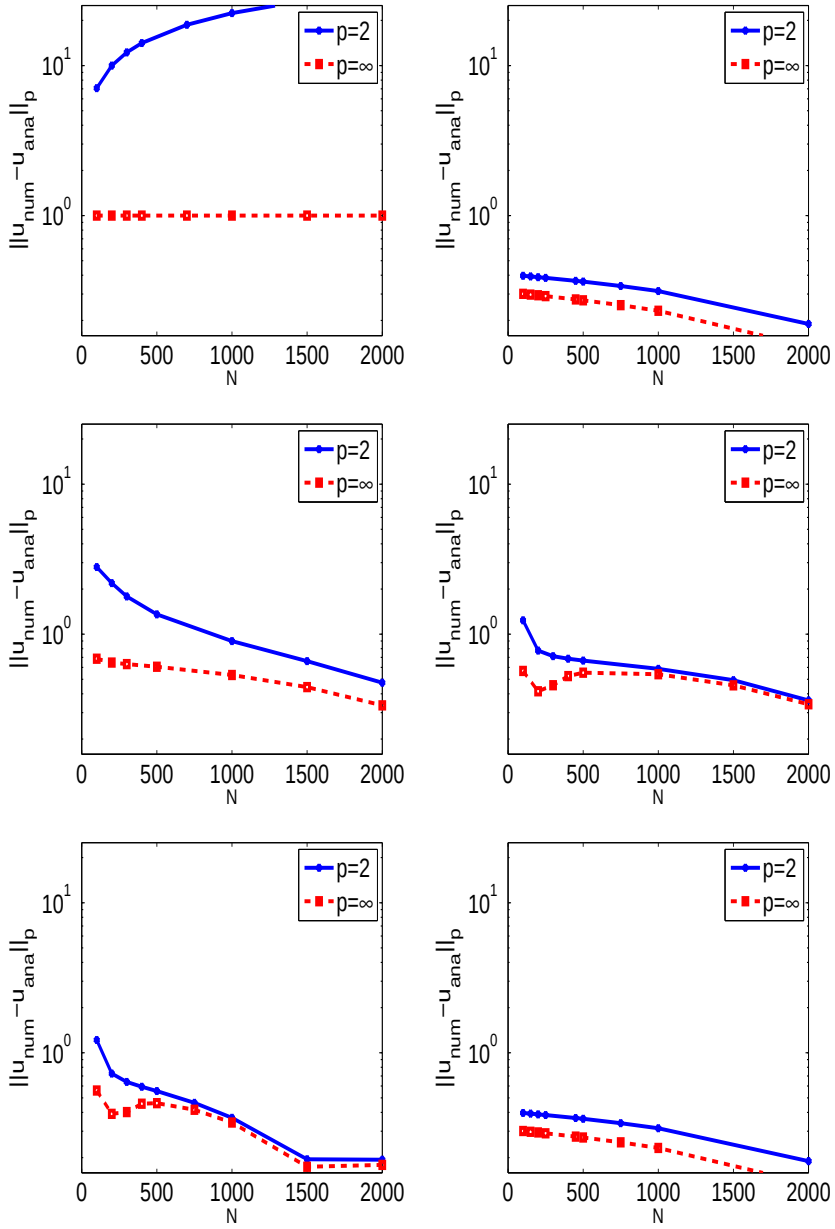


Figure 2.4: Error between numerical and analytical value versus no. of grid points for Upwind non-conservative, Upwind conservative, Lax-Friedrichs, Lax-Wendroff, MacCormack, Gudonov scheme respectively.

here Δt is time spacing and Δx is space spacing and u_i^n is the solution value at ith location in space and at nth time. Eqn. (2.3.1) is stated as the upwind non-conservative technique. The approach is in consistence with Eqn. (2.2.10) and for continuous solutions it is acceptable however in general, as the grid is refined, it does not converge to a weak discontinuous solution.

There is a simple condition to avoid convergence of a method to non-solutions, that we

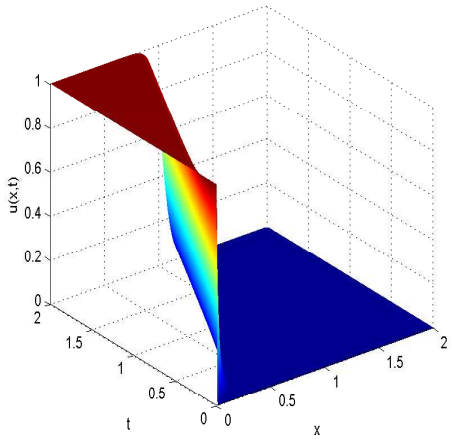
require. For this, the method should be conservative, *i.e.*, in the form

$$u_i^{n+1} = u_i^n - [F(u_{i-p}^n, u_{i-p+1}^n, \cdots, u_{i+q}^n) - F(u_{i-p+1}^n, u_{i-p}^n, \cdots, u_{i+q-1}^n)]\frac{\Delta t}{\Delta x},$$

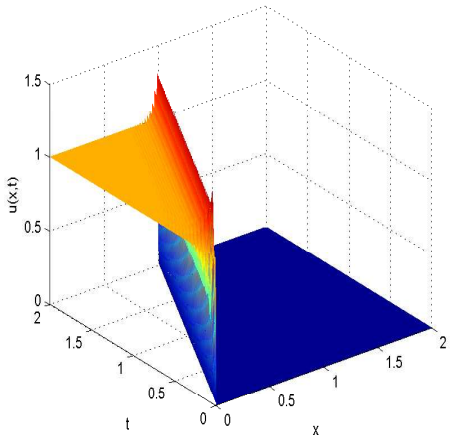
here F is a function of $p+q+1$ arguments which is called the function of numerical-flux. The easiest case is $q = 1$ and $p = 0$, here

$$u_i^{n+1} = u_i^n - [F(u_i^n, u_{i+1}^n) - F(u_{i-1}^n, u_i^n)]\frac{\Delta t}{\Delta x}.$$

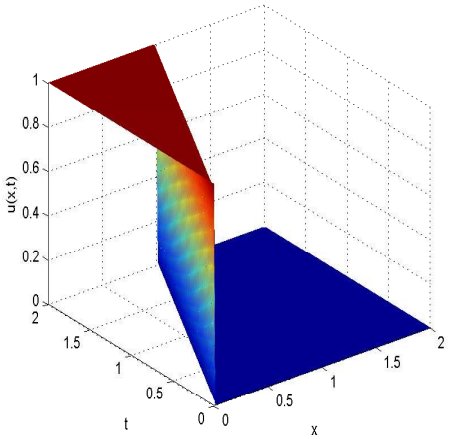
This expression is interpreted as cell average. Conservative methods are the methods that confirm this scheme. This class includes the following schemes.



(a) Lax-Friedrichs



(b) Lax-Wendroff



(c) Godunov

Figure 2.5: Numerical solution of Riemann Problem with $u_L = 1$ and $u_R = 0$ and $dt = 0.001$.

Upwind conservative:- Now we examine a general scalar conservative law

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0,$$

and a conservative method (called up-wind conservative) is obtained by using the standard finite difference discretizations,

$$u_i^{n+1} = u_i^n - [f(u_i^n) - f(u_{i-1}^n)]\frac{\Delta t}{\Delta x}.$$

For the traffic flow problem we have,

$$u_i^{n+1} = u_i^n - \left[\frac{1}{2}(u_i^n)^2 - \frac{1}{2}(u_{i-1}^n)^2\right]\frac{\Delta t}{\Delta x}.$$

Lax-Friedrichs:- For the nonlinear system, the Lax-Friedrichs scheme is:

$$u_i^{n+1} = \frac{1}{2}(u_{i-1}^n + u_{i+1}^n) - [f(u_{i+1}^n) - f(u_{i-1}^n)]\frac{\Delta t}{2\Delta x},$$

The scheme could be written into the conservative form by taking

$$F(u_i^n, u_{i+1}^n) = (u_i - u_{i+1})\frac{\Delta t}{2\Delta x} + \frac{1}{2}[f(u_i) + f(u_{i+1})].$$

For the traffic flow problem we have,

$$u_i^{n+1} = \frac{1}{2}(u_{i-1}^n + u_{i+1}^n) - \left[\frac{1}{2}(u_{i+1}^n)^2 - \frac{1}{2}(u_{i-1}^n)^2\right]\frac{\Delta t}{2\Delta x}.$$

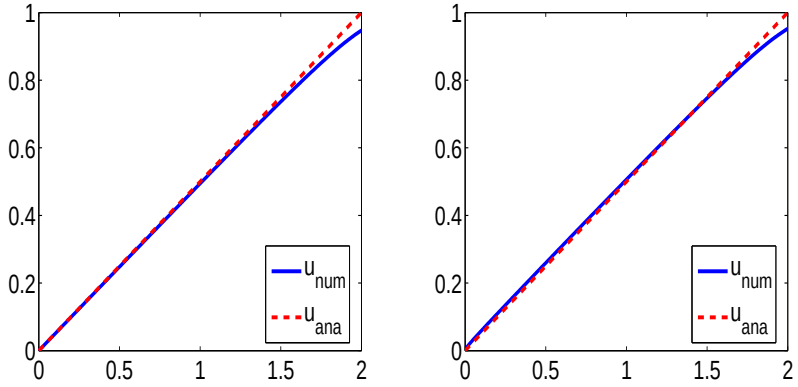
Lax-Wendroff:- For the non-linear conservative forms, the Lax-Wendroff method is a 2nd order scheme and is represented as:

$$\begin{aligned} u_i^{n+1} &= u_{i+1}^n - (f(u_{i+1}^n) - f(u_{i-1}^n))\frac{\Delta t}{2\Delta x} \\ &+ [f'(u_{i+1/2}^n)(f(u_{i+1}^n) - f(u_i^n)) - f'(u_{i-1/2}^n)(f(u_i^n) - f(u_{i-1}^n))]\frac{(\Delta t)^2}{2(\Delta x)^2}, \end{aligned}$$

where $u_{i\pm1/2}^n = \frac{1}{2}(u_i^n + u_{i\pm1/2})$. For the traffic flow problem we have, $f'(u) = u$. So,

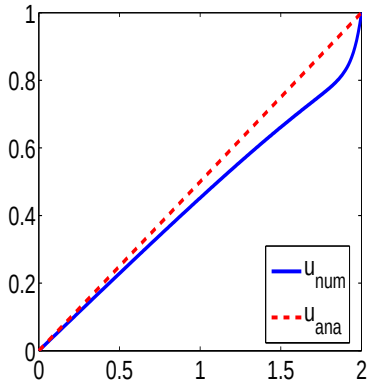
$$\begin{aligned} u_i^{n+1} &= u_{i+1}^n - (\frac{1}{2}(u_{i+1}^n)^2 - \frac{1}{2}(u_{i-1}^n)^2)\frac{\Delta t}{2\Delta x} + \left[(\frac{1}{2}(u_i^n + u_{i+1}^n))\right. \\ (\frac{1}{2}(u_{i+1}^n)^2 &- \left.\frac{1}{2}(u_i^n)^2) - (\frac{1}{2}(u_i^n + u_{i-1}^n))(\frac{1}{2}(u_i^n)^2 - \frac{1}{2}(u_{i-1}^n)^2)\right]\frac{(\Delta t)^2}{2(\Delta x)^2}. \end{aligned}$$

Mac-Cormack:- Another similar scheme is known as the Mac-Cormack scheme. Firstly

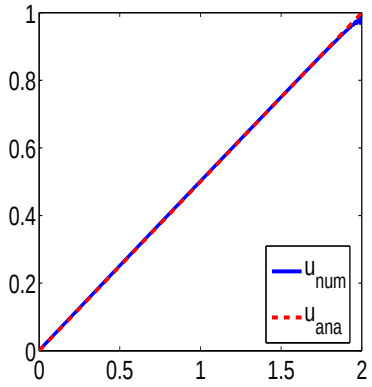


(a) Upwind nonconservative

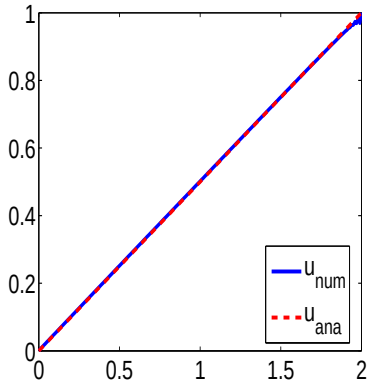
(b) Upwind conservative



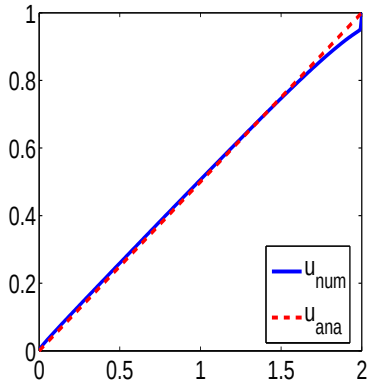
(c) Lax-Friedrichs



(d) Lax-Wendroff



(e) MacCormack



(f) Godunov

Figure 2.6: Analytical solution and Numerical solution for the traffic flow problem by using the above schemes.

forward-difference scheme is applied and after that backward-difference scheme is applied

for achieving the 2nd order accuracy.

$$\begin{aligned} u_i^* &= u_i^n - \frac{\Delta t}{\Delta x}[f(u_{i+1}^n) - f(u_i^n)], \\ u_i^{n+1} &= \frac{1}{2}(u_i^n + u_i^*) - \frac{\Delta t}{2\Delta x}[f(u_i^*) - f(u_{i-1}^*)]. \end{aligned}$$

Godunov:- Another numerical scheme for solving the traffic flow problem that is presented in this chapter comes under the category of finite-volume techniques. The basic concept of the Godunov's scheme is as follows. Assume that on the nth layer, assume that u_i^n is a numerical solution. At $t = t_n$,

$$\widehat{u}^n(x,t) = u_i^n, \quad x_i - \frac{\Delta x}{2} < x < x_i + \frac{\Delta x}{2}, \quad j = 2, \cdots, n-1.$$

On the interval $[t_n, t_{n+1}]$, we denote $\widehat{u}(x,t)$ to be a solution of the Riemann problems collection. On the next layer u_i^{n+1}, the numerical solution is determined by average of $\widehat{u}(x, t_{n+1})$ over the interval $x_i - \frac{\Delta x}{2} < x < x_i + \frac{\Delta x}{2}$. This phenomena is reduced to a simple conservative method as:

$$u_i^{n+1} = u_i^n - [F(u_j^n, u_{j+1}^n) - F(u_{j-1}^n, u_j^n)]\frac{\Delta t}{\Delta x},$$

here F is the numerical flow and is represented as $F(u,v) = \frac{(u^*)^2}{2}$, here u^* is described as following:
When $u \geq v$ then

$$u^* = \begin{cases} u & ; \frac{u+v}{2}, \\ v & ; \text{otherwise.} \end{cases}$$

When $u < v$ then

$$u^* = \begin{cases} u & ; u > 0, \\ v & ; v < 0, \\ 0 & ; u \leq 0 \leq v. \end{cases}$$

Now having explained all the schemes, the numerical results have been presented. Numerical solution of the problem (2.2.10) for the above two cases when $u_L > u_R$ and $u_L < u_R$ by using the different schemes (upwind nonconservative, upwind conservative, Lax-Friedrichs, Lax-Wendroff, MacCormack, Gudonov) are discussed below.
Case 1. Riemann Problem with $u_L = 1$ and $u_R = 0$.

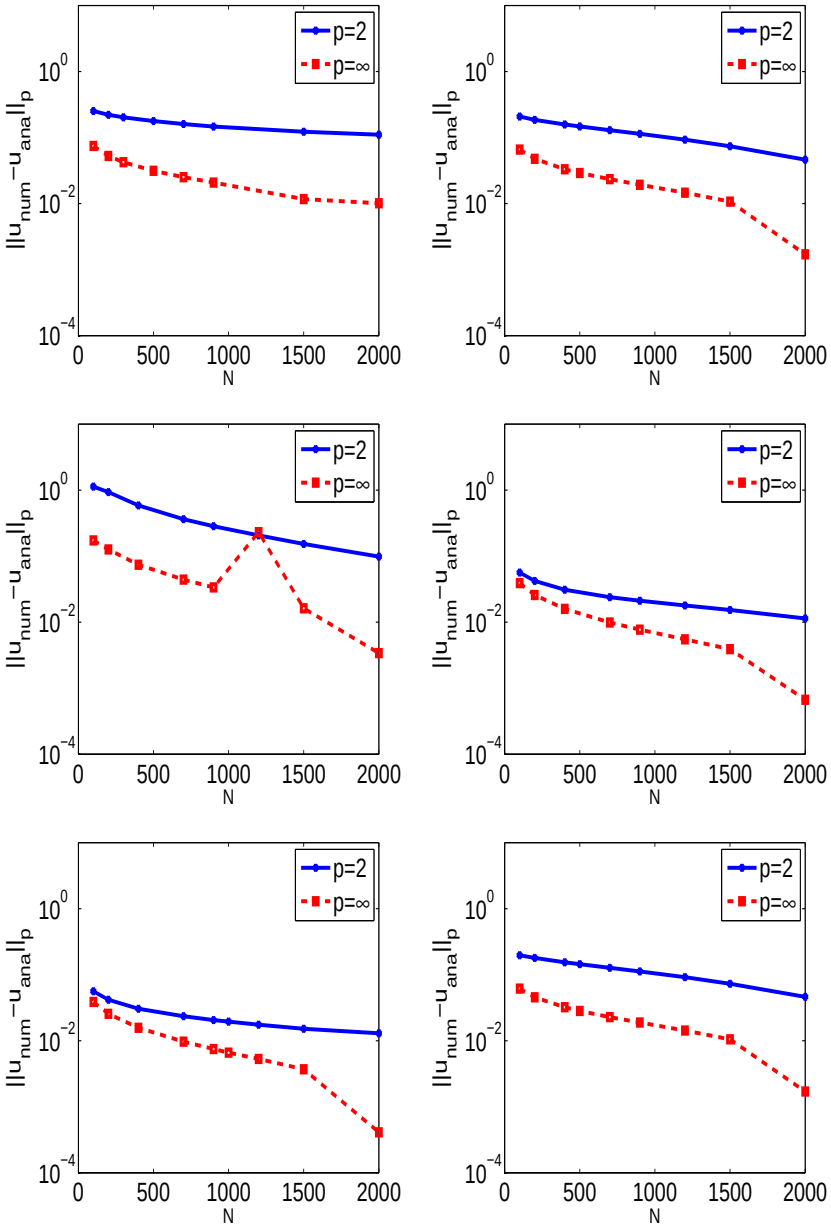


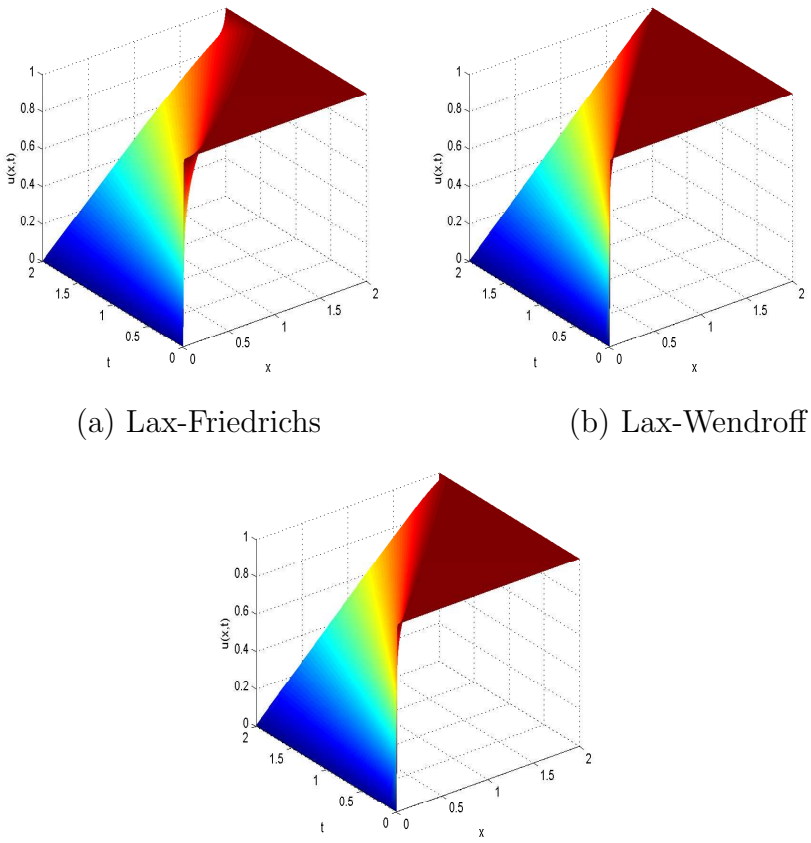
Figure 2.7: Error between numerical and analytical value versus no. of grid points for Upwind non-conservative, Upwind conservative, Lax-Friedrichs, Lax-Wendroff, MacCormack, Gudonov scheme respectively.

- Fig. (2.3) shows the graph of the analytical solution and numerical solution for the traffic flow problem by using different numerical schemes (No. of Grid points are chosen to be 200 and Δt is chosen to be 0.001). Δx and Δt is chosen to be in accordance with Courant-Friedrichs-Lewy (CFL) condition so that $u\frac{\Delta t}{\Delta x} < 1$ is satisfied. The condition of CFL is a mandatory requirement that should be satisfied for the stability of numerical scheme. CFL condition, however, is a condition that is necessary but not a sufficient requirement for numerical scheme stability. It can be observed

from these graphs that the upwind conservative scheme and Godunov scheme gives same results and converges near to exact solution. Lax-Friedrichs method is more diffusive than Upwind method as shown by numerical outcomes.

- Fig. (2.4) displays the graph of the no. of grid points (N) versus error between numerical and analytical value *i.e.*, $\|u_{num} - u_{ana}\|_p$. It is clear from the graphs that error decreases when no. of grid points increases. Numerical results indicate that Godunov type finite volume scheme is stable but can lead to large errors due to its slow convergence. We have also observed that Upwind non-conservative scheme does not provide good results for discontinuous solution.

- Fig. (2.5) displays the numerical solution of the problem by applying Lax-Friedrichs, Lax-Wendroff and Godunov scheme.



(a) Lax-Friedrichs (b) Lax-Wendroff

(c) Godunov

Figure 2.8: Numerical solution of Riemann Problem with $u_L = 0$ and $u_R = 1$ and $dt = 0.001$.

Case 2. Riemann Problem with $u_L = 0$ and $u_R = 1$.

- Fig. (2.6) shows the graph of the analytical solution and numerical solution for the

Riemann problem when $u_L = 0$ and $u_R = 1$ by using different numerical schemes. We have observed from the graphs that, MacCormack provides better results and converges near to exact solution. Lax-Friedrichs method is more diffusive than Upwind method as shown by numerical outcomes.

- Fig. (2.7) shows the relation between no. of grid points (N) and error between numerical and analytical value *i.e.*, $\|u_{num} - u_{ana}\|_p$. It is clear from the graphs that as no. of grid points increases error decreases.
- Fig. (2.8) displays the numerical outcomes of the problem by using Lax-Friedrichs, Lax-Wendroff and Godunov scheme.

Now best scheme has been obtained named as upwind conservative which will be used for wavelet optimized method and this method is called as wavelet optimized upwind conservative (WOUC) scheme.

Table 2.1: Grid modifications while solving case 1

Time (t)	N ($\epsilon = 0$)	N (ϵ)
0	1024	115
0.0625	115	131
0.1250	131	121
0.1875	121	128
0.2500	128	134
0.3125	134	136
0.3750	136	140
0.4375	140	141

Table 2.2: The performance of WOUC while solving case 1.

ϵ	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
10^{-2}	1.912	0.4168	4.587
10^{-4}	1.912	0.6904	3.137
10^{-6}	1.912	0.923	2.07
10^{-8}	1.912	1.04	1.83

2.3.1 WOUC applied to Traffic-flow problem

In this sub-section, WOUC will be applied to Riemann problem for two cases when $u_L > u_R$ and $u_L < u_R$.

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} &= 0, \\ \text{where } f(u) &= \frac{u^2}{2}. \end{aligned}$$

with an initial

$$u(x,0) = u_0(x) = \begin{cases} u_L & ; x \leq 0, \\ u_R & ; x > 0. \end{cases}$$

The aim of this section is to demonstrate the WOUC by applying the D_6 wavelet to create a solution on a non-uniform grid. There is a finer grid size to prevent oscillations from developing at the 'shock'. More precisely, we can say that one has a adequately refined grid to avoid an increase the fluctuations in the solution. In all of the following plots, the spatial and temporal discretization is achieved according to the Upwind conservative scheme. The threshold on WOUC coefficients that decides which grid points to be used depends on the magnitude of the wavelet coefficients. It is noted that when the threshold on WOUC coefficients is set to 0, then finite difference technique is obtained on the uniform finest grid. The size of the sparser grid relies upon the user's selected threshold.

Table 2.3: Modifications of the grid while solving case 2

Time (t)	N ($\epsilon = 0$)	N (ϵ)
0	256	67
0.0625	67	73
0.1250	73	83
0.1875	83	93
0.2500	93	101
0.3125	101	111
0.3750	111	119
0.4375	119	127

Case 1: Traffic flow problem with $u_L = 1$ and $u_R = 0$.
WOUC method is applied for solving this problem. Table (2.1) displays the modifications

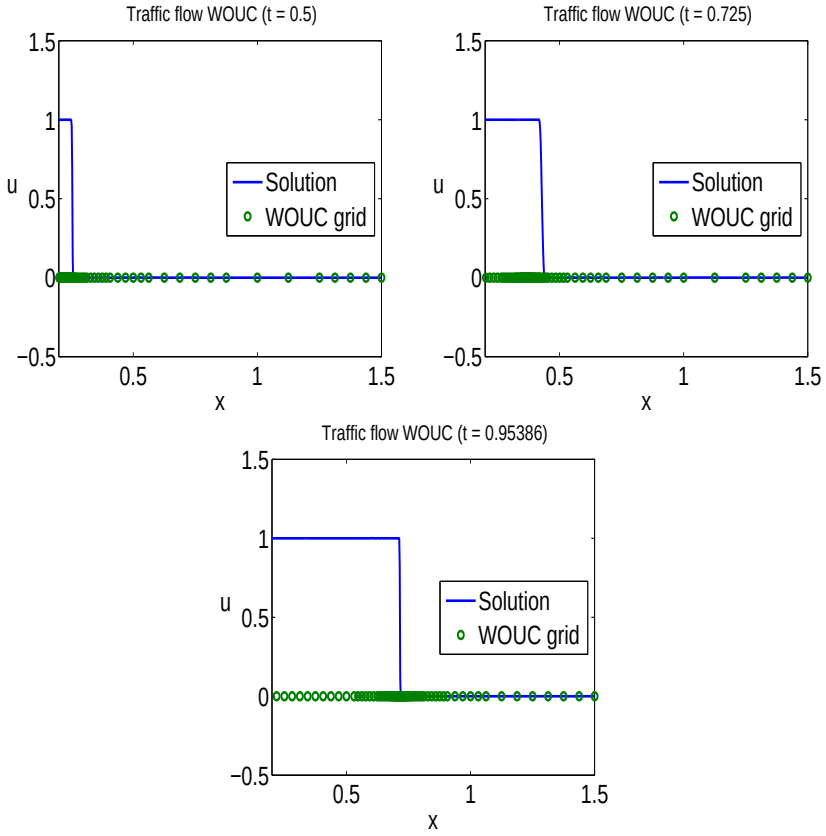


Figure 2.9: Solution and corresponding adaptive grid at time $t = 0.5$, $t = 0.725$, $t = 0.953$ respectively.

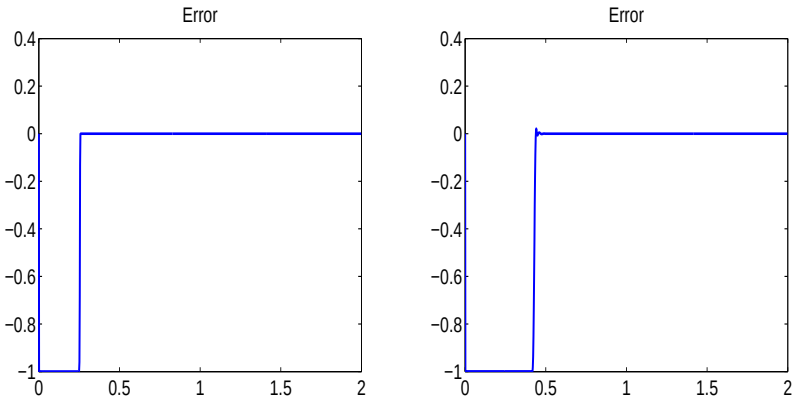


Figure 2.10: Point wise error at time $t = 0.5$, $t = 0.725$.

of the grid at distinct times. The coefficient of time-compression is defined as $\Theta = \frac{CPU(\epsilon=0)}{CPU(\epsilon)}$, where CPU($\epsilon$) denotes the CPU time taken when threshold ϵ is used and CPU($\epsilon = 0$) denotes the time taken where there is no threshold. Table (2.2) shows the variation of ϵ versus CPU(ϵ). It has been found that Θ decreases on decreasing the ϵ value. The higher the Θ value, the adaptive technique is more efficient. Fig. (2.9) represents the solution and the associated WOUC grid at distinct times. Fig. (2.10) displays the pointwise error

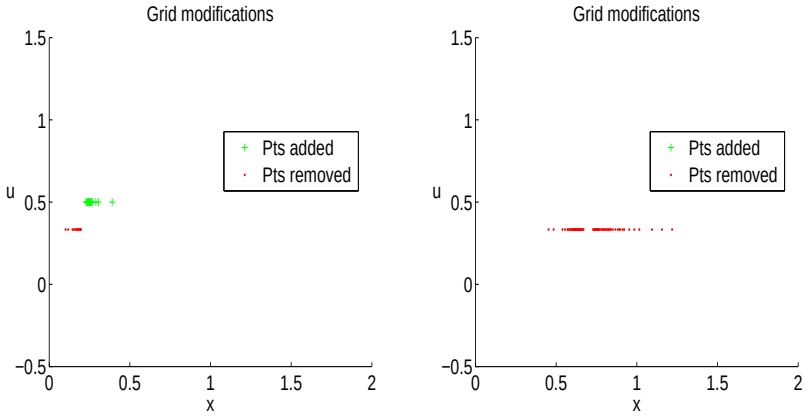


Figure 2.11: Added and removed points at time $t = 0.5$ and $t = 0.95$

at distinct times and it can be seen that where there are variations in the solution, the error value is maximum and therefore more grid points must be included at this region. Fig. (2.11) shows the added and removed points at distinct times. It is found that where the value of gradient is small grid is sparse and where large gradient appears grid is dense.

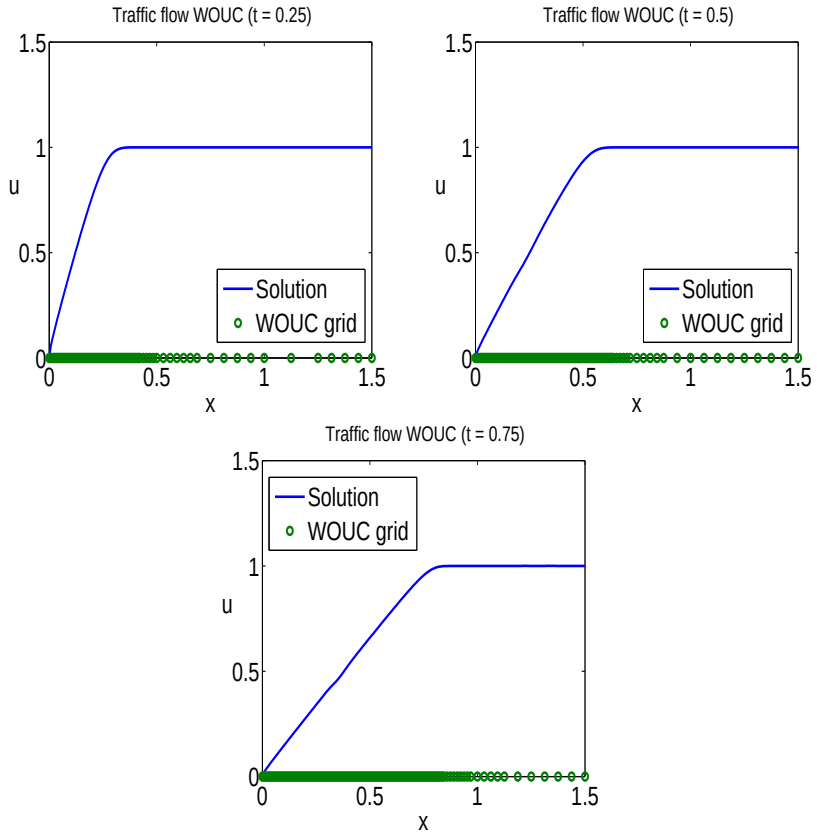


Figure 2.12: Solution and corresponding adaptive grid at time $t = 0.25$, $t = 0.5$, $t = 0.752$ respectively.

Table 2.4: The performance of WOUC while solving case 2.

ϵ	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
10^{-2}	2.3277	0.4219	5.517
10^{-4}	2.3277	0.7469	3.116
10^{-6}	2.3277	0.8265	2.816
10^{-8}	2.3277	1.213	1.918

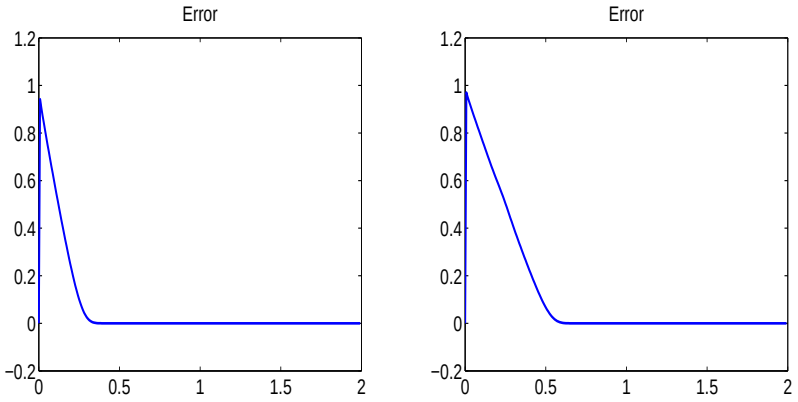


Figure 2.13: Point wise error at time $t = 0.25$, $t = 0.5$.

Case 2: Traffic flow problem with $u_L = 0$ and $u_R = 1$.
Table (2.3) shows the modifications of the grid at distinct times. Table (2.4) gives variation of ϵ versus CPU(ϵ). It has been observed that Θ decreases on decreasing the ϵ value. It shows the efficiency of our adaptive algorithm. Fig. (2.12) displays the solution and its related WOUC grid at distinct times. Fig. (2.13) displays the pointwise error at distinct times.

2.4 Conclusion

Studies of comparison have affirmed that upwind conservative scheme gives the outcomes in great understanding with exact results. It appears to be slightly more accurate than Lax-Friedrichs and Lax-Wendroff scheme. In general, upwind method is less dissipative than the Lax-Friedrichs method and provides more precise results. Having selected the best finite difference scheme namely upwind conservative, we have developed the wavelet optimized conservative (WOUC) method in this chapter. It has been shown that how a

dynamic adaptive wavelet approach works in a smooth way with local singularities of the solution of the Riemann's problem. The numerical outcomes show that the wavelet and computational grid can adapt very efficiently to the solution's local irregularities to resolve sharp transition areas. Comparison of computational time discloses that the proposed technique is extremely effective.

Chapter 3

Second generation wavelet-based numerical methods to solve PDEs

With an aim to develop wavelet based methods to solve PDEs with different boundaries, we shifted from Daubechies to second generation wavelet.

Typical environments where a single function can not be translated and dilated consists of

1. Constructing the wavelets on a bounded domain: It comprises of the wavelets construction at the interval (*i.e.*, for the space $\mathcal{L}_2([a,b])$), or a bounded domain in higher dimensional Euclidean.

2. Constructing the weighted-wavelets: For the weighted inner product wavelets are bi-orthogonal (*i.e.*, for the space $\mathcal{L}_2^w([a,b])$).

To construct wavelets in all the above settings and many more such as irregular samples, arbitrary weight function, the lifting method has been developed. The developed method enables the vast number of discrete bi-orthogonal wavelets to be generated from the first one [164, 165]. Moreover, with the developed method, all the first generation wavelets can be constructed.

In this chapter, a construction of second generation wavelet on an interval has been presented and a collection of Matlab routines for the second generation wavelet transformation and the inverse wavelet transformation on a space $\mathcal{L}_2([a,b])$ has been given in the appendix. Having constructed the second generation wavelet on an interval, the same has been used in generation of adaptive grid. The second-generation wavelet optimized finite-difference (SGWOFD) technique developed in this chapter has been used for solving the Burger's equation with distinct boundaries. Four test problems (with Dirichlet, Periodic, Robin and Neumann's boundaries) are considered and the convergence of the technique is checked. The computational time carried out by the SGWOFD has been computed for each test problem and has been compared to that of the finite difference approach on a grid which is uniform. It has been revealed that SGWOFD is highly efficient.

3.1 Lifting scheme

Multiresolution analysis (MRA): MRA of $\mathcal{L}_2(\mathbb{R})$ is constructed by utilizing the two functions: ψ (a wavelet function) and ϕ (a scaling function). ϕ (scaling function) fulfills the following refinement relationship

$$\phi(x) = 2\sum_k h_k \phi(2x-k). \tag{3.1.1}$$

The set of integer translations of ϕ forms a Riesz basis of the closure of their span. The following wavelet equation gives $\psi \in \mathcal{L}_2(\mathbb{R})$

$$\psi(x) = 2\sum_k g_k \phi(2x-k). \tag{3.1.2}$$

here g_k and h_k are high-pass and low-pass filter-coefficients respectively. The set $\{\psi_j^l(x) = 2^{\frac{j}{2}}\psi(2^j x - l); j, l \in \mathbb{Z}\}$ forms the Riesz bases of the space $\mathcal{L}_2(\mathbb{R})$.

Other than a wavelet function ψ and a scaling function ϕ, there also exist $\tilde{\psi}$ (a dual wavelet function) and $\tilde{\phi}$ (a dual scaling function). The dual wavelet and scaling functions $\tilde{\psi}$ and $\tilde{\phi}$ also generate the MRA. They satisfy the eqns. (3.1.2) and (3.1.1) with coefficients $\{\tilde{g}_k\}$ and $\{\tilde{h}_k\}$ respectively. They are biorthogonal to ψ and ϕ as the following way

$$\langle\tilde{\phi}, \psi(\cdot - l)\rangle = \langle\tilde{\psi}, \phi(\cdot - l)\rangle = 0, \quad \langle\tilde{\phi}, \phi(\cdot - l)\rangle = \langle\tilde{\psi}, \psi(\cdot - l)\rangle = \delta_l \tag{3.1.3}$$

We define the following 2π-periodic functions

$$h(\omega) = \sum_k h_k e^{-ik\omega}, g(\omega) = \sum_k g_k e^{-ik\omega}, \tilde{h}(\omega) = \sum_k \tilde{h}_k e^{-ik\omega}, \tilde{g}(\omega) = \sum_k \tilde{g}_k e^{-ik\omega}.$$

The h is referred as a filter related to the scaling function ϕ where we can consider h either as the 2π periodic function $\{h(\omega)\}$ or a sequence of the coefficients $\{h_k\}$. Similarly g, $\tilde{h}, \tilde{g}$ are the filters associated with wavelet function ψ, dual scaling function $\tilde{\phi}$ and dual wavelet function $\tilde{\psi}$ respectively. Given the filter h, by iterating the refinement Eqn. (3.1.1), we have

$$\hat{\phi}(\omega) = \prod_{j=1}^{\infty} h(2^{-j}\omega), \text{ where Fourier transform of } f \text{ is } \hat{f}(\omega) = \int_{-\infty}^{\infty} f(x)e^{-i\omega x}dx. \tag{3.1.4}$$

When all the four functions we are dealing with (namely $\phi, \tilde{\phi}, \psi, \tilde{\psi}$) are of compactly support *i.e.*, except the coefficients in the relations (3.1.1) and (3.1.2), all others are zero.

In this case we call $h, g, \tilde{g}$ and $\tilde{h}$ as the finite filters. And the bi-orthogonality conditions (3.1.3) could also be composed in terms of the filter coefficients. In this way we have a set $\{h, g, \tilde{g}, \tilde{h}\}$ of finite bi-orthogonal filters. Depending on the functions out of $\phi, \tilde{\phi}, \psi, \tilde{\psi}$ which are being upgraded, the lifting can be classified as either primal lifting or dual lifting explained in the following sections.

3.1.1 Primal lifting scheme

Suppose we have two sets $\{\phi, \psi, \tilde{\phi}, \tilde{\psi}\}$ and $\{h, g, \tilde{h}, \tilde{g}\}$. Eqn. (3.1.4) gives a procedure to go from the set $\{h, \tilde{h}, g, \tilde{g}\}$ to the set $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$ (Note that given a set of bi-orthogonal filters $\{h, \tilde{h}, g, \tilde{g}\}$, the corresponding set of bi-orthogonal functions $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$ does not always exists (see [166] and [78] for more details)).

Now for a moment forget about the scaling and wavelet functions and just think of set $\{h, g, \tilde{h}, \tilde{g}\}$ as a finite bi-orthogonal filters set. We have the following result (see [167] for details)

Theorem 3.1.1 *A new set of finite bi-orthogonal filters $\{h, g, \tilde{h}, \tilde{g}\}$ can be obtained by taking the initial finite bi-orthogonal filters set $\{h, g^0, \tilde{h}^0, \tilde{g}\}$ as follows:*

$$\tilde{h}(\omega) = \tilde{h}^0(\omega) + \tilde{g}(\omega)\overline{s(2\omega)}, \tag{3.1.5}$$

$$g(\omega) = g^0(\omega) - h(\omega)s(2\omega), \tag{3.1.6}$$

here $s(\omega)$ is a polynomial of trigonometric equations and it is chosen such that the bi-orthogonal functions $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$ associated with the new finite bi-orthogonal filters set $\{h, g, \tilde{h}, \tilde{g}\}$ have desirable properties.

The above procedure of constructing the new finite bi-orthogonal filters set $\{h, g, \tilde{h}, \tilde{g}\}$ from an old finite bi-orthogonal filters set $\{h, g^0, \tilde{h}^0, \tilde{g}\}$ is known as primal lifting procedure. The relation between the new bi-orthogonal functions set $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$ and the old bi-orthogonal functions set $\{\phi^0, \tilde{\phi}^0, \psi^0, \tilde{\psi}^0\}$ is represented by the following theorem (see [167] for details).

Theorem 3.1.2 *A new bi-orthogonal set $\{\phi, \psi, \tilde{\phi}, \tilde{\psi}\}$ can be obtained from an initial bi-orthogonal scaling functions and wavelets functions set $\{\phi^0, \psi^0, \tilde{\phi}^0, , \tilde{\psi}^0\}$ as follows:*

$$\phi(x) = \phi^0(x). \tag{3.1.7}$$

$$\psi(x) = \psi^0(x) - \sum_k s_k \phi(x-k). \tag{3.1.8}$$

$$\tilde{\phi}(x) = 2\sum_k \tilde{h}^0_k \tilde{\phi}(2x-k) + \sum_k s_{-k}\tilde{\psi}(x-k). \tag{3.1.9}$$

$$\tilde{\psi}(x) = 2\sum_k \tilde{g}_k \tilde{\phi}(2x-k). \tag{3.1.10}$$

where the coefficients s_k and s_{-k} can be chosen freely.

The crux of the above theorem is that you can start from a very simple and easy to handle wavelet and construct complicated wavelets with desirable properties. This is illustrated in the following examples:

Primal Lifting of the Haar wavelet: The scaling function of Haar [168] is described as

$$\phi(x) = \begin{cases} 1 & \text{if } x \in [0,1) \\ 0 & \text{otherwise} \end{cases}$$

and the corresponding wavelet function is represented as

$$\psi(x) = \begin{cases} 1 & \text{if } x \in [0,0.5) \\ -1 & \text{if } x \in [0.5,1) \\ 0 & \text{otherwise} \end{cases}$$

And since it is an orthogonal wavelet, we have $\tilde{\phi} = \phi$ and $\tilde{\psi} = \psi$. It is easy to see that the following two relations are satisfied

$$\phi(x) = 2\left(\frac{1}{2}\phi(2x) + \frac{1}{2}\phi(2x-1)\right), \tag{3.1.11}$$

$$\psi(x) = 2\left(\frac{1}{2}\phi(2x) - \frac{1}{2}\phi(2x-1)\right). \tag{3.1.12}$$

Comparing the Eqns. (3.1.11) and (3.1.12) with the Eqns. (3.1.1) and (3.1.2) respectively, we get $\{h_k\} = \{h_0 = 1/2, h_1 = 1/2\}$ and $\{g_k\} = \{g_0 = 1/2, g_1 = -1/2\}$. The corresponding 2π-periodic functions are

$$h(\omega) = \tilde{h}^0(\omega) = 1/2 + 1/2e^{-i\omega},$$

$$g^0(\omega) = \tilde{g}(\omega) = 1/2 - 1/2e^{-i\omega}.$$

Haar wavelet has 1 vanishing moment (If a polynomial upto degree p is constructed by the

scaling functions, then the associated wavelet has p vanishing moments). It means that the only scaling functions can be employed for representing the polynomials upto degree p *i.e.*, wavelet coefficients are not used for representing such functions. To represent more complex structures, wavelets should have more vanishing moments. More vanishing moments means that complex structures could be described with a sparse set of wavelet's coefficients. Now assume that we wish to have a wavelet having 2 vanishing moments and we want to apply the primal lifting strategy on Haar wavelet for construction of such wavelet. On lifting $\{h, g^0, \tilde{h}^0, \tilde{g}\}$ to $\{h, g, \tilde{h}, \tilde{g}\}$ we get

$$\tilde{h}(\omega) = \tilde{h}^0(\omega) + \tilde{g}(\omega)\overline{s(2\omega)},$$

$$g(\omega) = g^0(\omega) - h(\omega)s(2\omega).$$

Now the condition that we want 2 vanishing moments will help us to find the function $s(\omega)$ in the above equations.

In order to have 1 vanishing moment we need $g(0) = 0$ which implies $g^0(0) - h(0)s(0) = 0$ which implies $s(0) = 0$ ($g^0(0) = 0$ as Haar wavelet also has 1 vanishing moment, and $h(0) \neq 0$). For 2 vanishing moments we additionally need $g'(0) = 0$ and it implies $s'(0) = -\frac{i}{4}$. Now we have $s(0) = 0, s'(0) = -\frac{i}{4}$, we can choose $s(\omega) = -\frac{i}{4}\sin\omega$ (of course it is not unique!). This choice of $s(\omega)$ will give us

$$\tilde{h}(\omega) = -\frac{1}{16}e^{2i\omega} + \frac{1}{16}e^{i\omega} + \frac{1}{2} + \frac{1}{2}e^{-i\omega} + \frac{1}{16}e^{-2i\omega} - \frac{1}{16}e^{-3i\omega},$$

$$\tilde{g}(\omega) = -\frac{1}{16}e^{2i\omega} - \frac{1}{16}e^{i\omega} + \frac{1}{2} - \frac{1}{2}e^{-i\omega} + \frac{1}{16}e^{-2i\omega} + \frac{1}{16}e^{-3i\omega}.$$

Now from the set of $\{h, \tilde{h}, g, \tilde{g}\}$ filters we could construct the set $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$. It turns out that the set $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$ corresponds to one of the famous Cohen-Daubechies-Feauveau wavelet (known as cdf1.3) shown in Fig. (3.1).

3.1.2 Dual lifting scheme

In the last section we have gone from the set of $\{h, g^0, \tilde{h}^0, \tilde{g}\}$ filters to $\{h, g, \tilde{h}, \tilde{g}\}$ set. The set of filters $\{h, \tilde{h}, g, \tilde{g}\}$ can also be obtained from the initial set of filters $\{h^0, \tilde{h}, g, \tilde{g}^0\}$ (without altering the g and $\tilde{h}$ while the filters $\tilde{g}$ and h be changed). Suppose the associated trigonometric polynomial be represented by $\tilde{s}(\omega)$. This procedure will result in dual lifting. We have the following relations

$$h(\omega) = h^0(\omega) + g(\omega)\overline{\tilde{s}(2\omega)},$$

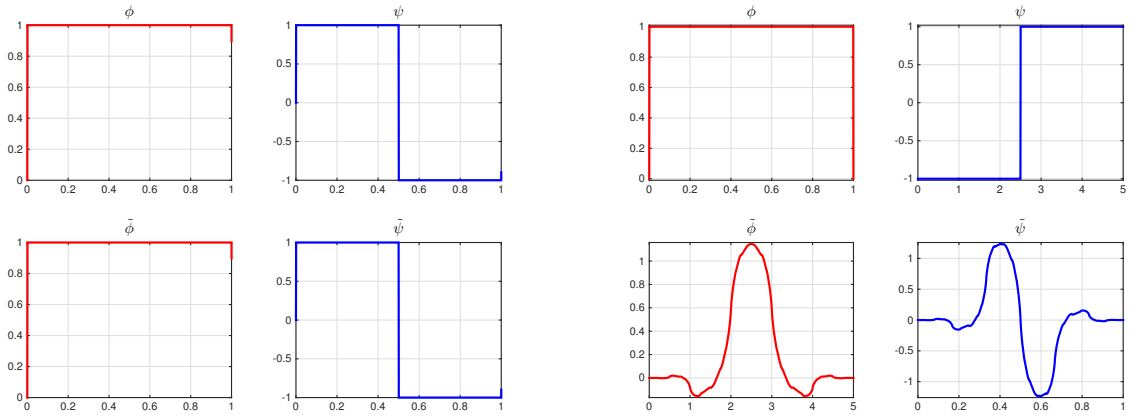


Figure 3.1: Haar wavelet (on left hand side) lifted to the Cohen-Daubechies-Feauveau wavelet (on right hand side).

$$\tilde{g}(\omega) = \tilde{g}^0(\omega) - \tilde{h}(\omega)\tilde{s}(2\omega).$$

The initial bi-orthogonal scaling and wavelet functions set $\{\phi^0, \tilde{\phi}^0, \psi^0, \tilde{\psi}^0\}$ is related to the new $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$ set by the following relations

$$\tilde{\phi}(x) = \tilde{\phi}^0(x),$$

$$\tilde{\psi}(x) = \tilde{\psi}^0(x) - \sum_k \tilde{s}_k \tilde{\phi}(x-k),$$

$$\phi(x) = 2\sum_k h_k^0 \phi(2x-k) + \sum_k \tilde{s}_{-k}\psi(x-k),$$

$$\psi(x) = 2\sum_k g_k \phi(2x-k),$$

where the $\tilde{s}_k$ coefficients could be chosen without any restriction. It is noted that the dual lifting approach is employed for improving the properties of the primal scaling function ϕ or the dual wavelet $\tilde{\psi}$ and primal lifting approach is employed for improving the properties of the dual scaling function $\tilde{\phi}$ or the primal wavelet function ψ.

3.2 The discrete wavelet transformation (DWT) and inverse discrete wavelet transformation (IDWT)

General technique for constructing wavelets with lifting scheme is:

1. Design the DWT without any reference to the wavelet functions $\psi, \tilde{\psi}$ or the scaling

functions $\phi, \tilde{\phi}$.

2. The wavelet functions $\psi, \tilde{\psi}$ and the scaling functions $\phi, \tilde{\phi}$ are then constructed using DWT.

Therefore, the most important step is the construction of DWT and IDWT which is discussed below.

For a function $f \in \mathcal{L}_2(\mathbb{R})$, the coefficients of scaling function λ_j^l and the coefficients of wavelet function γ_j^l are defined as follows

$$\lambda_j^l = \langle f, \tilde{\phi}_j^l \rangle \text{ and } \gamma_j^l = \langle f, \tilde{\psi}_j^l \rangle, \tag{3.2.1}$$

here $\tilde{\phi}_j^l = 2^{j/2}\tilde{\phi}(2^j x - l)$ etc. Let us suppose λ_n^l is given for a fixed n and we want to compute λ_j^l and γ_j^l for $j < n$. It can be done by applying the refinement Eqn. (3.1.1) recursively as follows:

$$\begin{aligned}
\tilde{\phi}_j^l &= 2^{j/2}\tilde{\phi}(2^j x - l), \\
&= 2^{j/2} \cdot 2 \sum_k \tilde{h}_k \tilde{\phi}(2(2^j x - l) - k), \text{ (using (3.1.1))}, \\
&= \sqrt{2} \sum_k \tilde{h}_{k-2l} \tilde{\phi}_{j+1}^k.
\end{aligned}$$

Therefore

$$\begin{aligned}
\lambda_j^l &= \langle f, \tilde{\phi}_j^l \rangle, \\
&= \langle f, \sqrt{2} \sum_k \tilde{h}_{k-2l} \tilde{\phi}_{j+1}^k \rangle, \\
&= \sqrt{2} \sum_k \tilde{h}_{k-2l} \lambda_{j+1}^k.
\end{aligned} \tag{3.2.2}$$

Similarly

$$\gamma_j^l = \sqrt{2} \sum_k \tilde{g}_{k-2l} \lambda_{j+1}^k. \tag{3.2.3}$$

The inverse transformation applies the equation recursively

$$\lambda_{j+1}^k = \sqrt{2} \sum_l h_{k-2l} \lambda_j^l + \sqrt{2} \sum_l g_{k-2l} \gamma_j^l. \tag{3.2.4}$$

The linear method obtained is known as the FWT.

Now with a primal lifting scheme we go from $\{h, \tilde{h}^0, g^0, \tilde{g}\}$ to the set $\{h, \tilde{h}, g, \tilde{g}\}$ (one should note that the function $s(\omega)$ is needed for doing so) and hence from the set $\{\phi^0, \tilde{\phi}^0, \psi^0, \tilde{\psi}^0\}$

to the set $\{\phi, \tilde{\phi}, \psi, \tilde{\psi}\}$. Suppose we wish for computing the scaling function and the wavelet coefficients associated with the later set. Essentially we don't have to design the g and $\tilde{h}$ filters explicitly, however we can deal with the $h, \tilde{h}^0, g^0$ and $\tilde{g}$ instead. The number of operations involved here are less in relative to the standard techniques. The things go like this

$$\begin{aligned}
\tilde{\phi}_j^l &= 2^{j/2}\tilde{\phi}(2^j x - l), \\
&= 2^{j/2}\left(2\sum_k \tilde{h}_k^0 \tilde{\phi}(2(2^j x - l) - k) + \sum_k s_{-k}\tilde{\psi}(2^j x - l - k)\right) \text{(from (3.1.9))}, \\
&= \sqrt{2}\sum_k \tilde{h}_{k-2l}^0 \tilde{\phi}_{j+1}^k + \sum_k s_{l-k}\tilde{\psi}_j^k.
\end{aligned}$$

Therefore

$$\lambda_j^l = \sqrt{2}\sum_k \tilde{h}_{k-2l}^0 \lambda_{j+1}^k + \sum_k s_{l-k}\gamma_j^k. \tag{3.2.5}$$

Because of (3.1.10) we have

$$\gamma_j^l = \sqrt{2}\sum_k \tilde{g}_{k-2l}\lambda_{j+1}^k. \tag{3.2.6}$$

Using (3.1.6) we obtain the inverse transform as

$$\lambda_{j+1}^k = \sqrt{2}\sum_l h_{k-2l}\left(\lambda_j^l - \sum_m s_{l-m}\gamma_j^m\right) + \sqrt{2}\sum_l g_{k-2l}^0 \gamma_j^l. \tag{3.2.7}$$

Therefore the subsequent algorithm presents the fast-lifting wavelet transformation (discrete wavelet transformation DWT) and its inverse (inverse discrete wavelet transformation IDWT).

DWT:

Step I: (Computation of unlifted coefficients)

$$\lambda_j^l = \sqrt{2}\sum_k \tilde{h}_{k-2l}^0 \lambda_{j+1}^k.$$

$$\gamma_j^l = \sqrt{2}\sum_k \tilde{g}_{k-2l}\lambda_{j+1}^k.$$

Step II: (Computation of lifted coefficients)

$$\lambda_j^l = \lambda_j^l + \sum_k s_{l-k}\gamma_j^k.$$

IDWT:

Step I:

$$\lambda_j^l = \lambda_j^l - \sum_k s_{l-k}\gamma_j^k.$$

Step II:

$$\lambda_{j+1}^k = \sqrt{2}\sum_l h_{k-2l}\lambda_j^l + \sqrt{2}\sum_l g_{k-2l}^0\gamma_j^l.$$

We can start with any simple wavelet and then adopt the lifting scheme (primal or dual) for constructing complex wavelets. The easiest possible choice is that one should start with the *lazy wavelet*. The bi-orthogonal filters associated with the lazy wavelet are

$$2\tilde{h}(\omega) = h(\omega) = 1 \text{ and } \tilde{g}(\omega) = 2g(\omega) = e^{-i\omega} \tag{3.2.8}$$

or

$$h = \{h_0 = 1, h_i = 0\ \forall i \neq 0\}, \tilde{h} = \{\tilde{h}_0 = \frac{1}{2}, \tilde{h}_i = 0\ \forall i \neq 0\}. \tag{3.2.9}$$

$$g = \{g_1 = \frac{1}{2}, g_i = 0\ \forall i \neq 1\}, \tilde{g} = \{\tilde{g}_1 = 1, \tilde{g}_i = 0\ \forall i \neq 1\}. \tag{3.2.10}$$

A step of lazy wavelet transformation is nothing more than sub-sampling into odd and even indexed samples. There is no associated set of $\{\psi, \tilde{\psi}, \phi, \tilde{\phi}\}$ in $\mathcal{L}_2$.

Now if we start with the samples $\{\lambda_{j+1}^k\}$ and the wavelet used is lazy wavelet, then the Step I of the DWT gives us the following result

$$\text{for } l = 0 \quad \lambda_j^0 = \sqrt{2}\sum_k \tilde{h}_k^0\lambda_{j+1}^k = \frac{1}{\sqrt{2}}\lambda_{j+1}^0,$$

$$\text{for } l = 1 \quad \lambda_j^1 = \sqrt{2}\sum_k \tilde{h}_{k-2}^0\lambda_{j+1}^k = \frac{1}{\sqrt{2}}\lambda_{j+1}^2,$$

$$\text{for } l = 2 \quad \lambda_j^2 = \sqrt{2}\sum_k \tilde{h}_{k-4}^0\lambda_{j+1}^k = \frac{1}{\sqrt{2}}\lambda_{j+1}^4,$$

and so on.

$$\text{for } l = 0 \quad \gamma_j^0 = \sqrt{2}\sum_k \tilde{g}_k^0\lambda_{j+1}^k = \sqrt{2}\lambda_{j+1}^1,$$

$$\text{for } l = 1 \quad \gamma_j^1 = \sqrt{2}\sum_k \tilde{g}_{k-2}^0\lambda_{j+1}^k = \sqrt{2}\lambda_{j+1}^3,$$

$$\text{for } l = 2 \quad \gamma_j^2 = \sqrt{2}\sum_k \tilde{g}_{k-4}^0\lambda_{j+1}^k = \sqrt{2}\lambda_{j+1}^5,$$

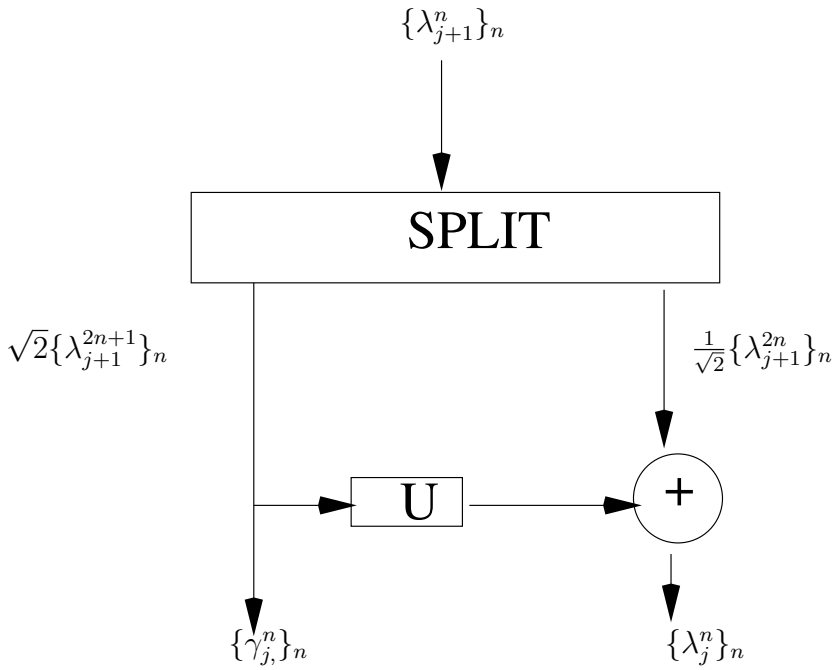


Figure 3.2: Split and Update step of forward transform.

and so on. Therefore the step I of the DWT is nothing but splitting the samples into even and odd indexed samples (and a multiplication with $\sqrt{2}$ of course!). We call it as the SPLIT step. We write it as

$$\{\frac{1}{\sqrt{2}}\lambda_{j+1}^{2n}\}_n, \{\sqrt{2}\lambda_{j+1}^{2n+1}\}_n = \text{SPLIT}(\{\lambda_{j+1}^n\}_n).$$

$$\{\gamma_j^n\}_n = \{\sqrt{2}\lambda_{j+1}^{2n+1}\}_n.$$

The step II of DWT can be composed as

$$\{\lambda_j^n\}_n = \{\frac{1}{\sqrt{2}}\lambda_{j+1}^{2n}\}_n + U(\{\gamma_j^n\}_n),$$

where U stands for the UPDATE operator. Idea of SPLIT and UPDATE step is explained in the Fig. (3.2).

Till now we have considered only the primal lifting scheme into account. If we take the dual lifting scheme into account as well, then the Eqn. (3.2.6) becomes

$$\gamma_j^l = \sqrt{2}\sum_k \tilde{g}_{k-2l}\lambda_{j+1}^k - \sum_k \tilde{s}_{k-l}\lambda_j^k. \tag{3.2.11}$$

With this DWT becomes

Step I: (Computation of unlifted coefficients)

$$\lambda_j^l = \sqrt{2} \sum_k \tilde{h}_{k-2l}^0 \lambda_{j+1}^k.$$

$$\gamma_j^l = \sqrt{2} \sum_k \tilde{g}_{k-2l} \lambda_{j+1}^k.$$

Step II: (Computation of lifted coefficient)

$$\lambda_j^l = \lambda_j^l + \sum_k s_{l-k} \gamma_j^k.$$

$$\gamma_j^l = \gamma_j^l - \sum_k \tilde{s}_{k-l} \lambda_j^k.$$

Starting with the lazy wavelet, the above DWT can be composed as:

$$\{\frac{1}{\sqrt{2}} \lambda_{j+1}^{2n}\}_n, \{\sqrt{2} \lambda_{j+1}^{2n+1}\}_n = \text{SPLIT}(\{\lambda_{j+1}^n\}_n),$$

$$\{\gamma_j^n\}_n = \{\sqrt{2} \lambda_{j+1}^{2n+1}\}_n - P(\{\lambda_{j+1}^{2n}\}_n),$$

$$\{\lambda_j^n\}_n = \{\frac{1}{\sqrt{2}} \lambda_{j+1}^{2n}\}_n + U(\{\gamma_j^n\}_n),$$

where U and P respectively stands for UPDATE and PREDICT operator. The IDWT can be composed as

$$\{\lambda_{j+1}^{2n}\}_n = \sqrt{2} \left(\{\lambda_j^n\}_n - U(\{\gamma_j^n\}_n) \right),$$

$$\{\lambda_{j+1}^{2n+1}\}_n = \frac{1}{\sqrt{2}} \left(\{\gamma_j^n\}_n + P(\{\lambda_{j+1}^{2n}\}_n) \right),$$

$$\{\lambda_{j+1}^n\}_n = \text{MERGE}(\{\lambda_{j+1}^{2n}\}_n, \{\lambda_{j+1}^{2n+1}\}_n).$$

Fig. (3.3) shows DWT and IDWT.

3.2.1 Another point of view of looking at the three steps: SPLIT, PREDICT and UPDATE

Assume that the sampling of a signal having sampling distance is 1, is denoted by $\lambda_0^k = f(k), k \in \mathbb{Z}$. We want to know if the information contained in the signal can be captured with lesser coefficients. This is done with the help of following three steps:

- **SPLIT:** The number of coefficients could be lessened by merely sub-sampling the

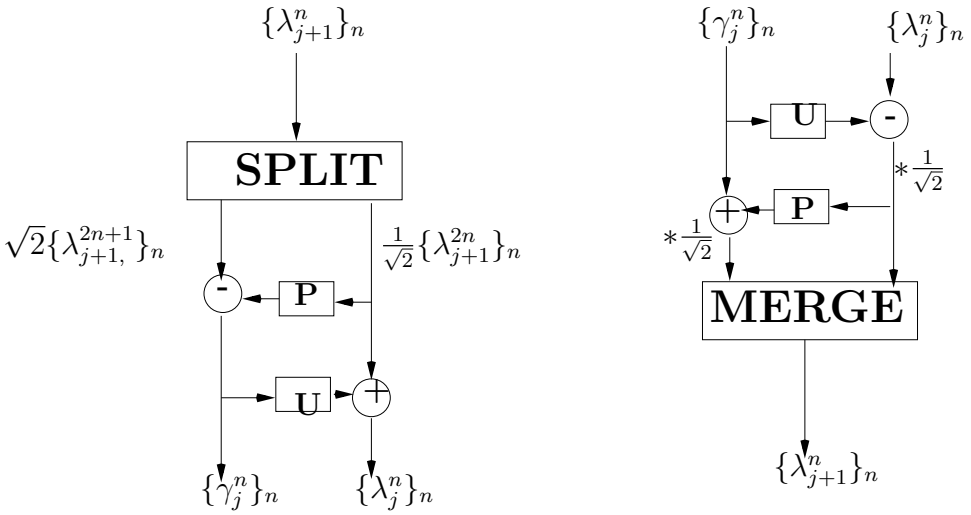


Figure 3.3: Forward and inverse transform.

even samples and achieving a fresh sequence given as

$$\lambda^k_{-1} = \lambda^{2k}_0, k \in \mathbb{Z},$$

we have used the negative indexes because the lower the index, lower is the data set

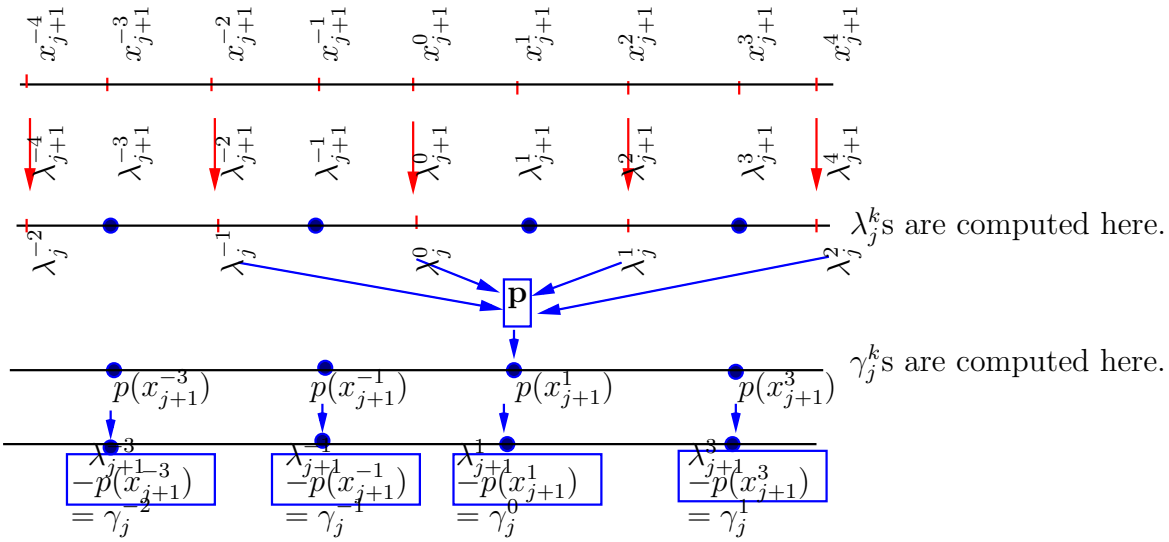


Figure 3.4: Predict step.

Now we would also like to know that how much information is lost, i.e., what is the difference between the set $\{\lambda^k_0\}$ and the set $\{\lambda^k_{-1}\}$. To encrypt this difference, we use the coefficients γ^k_{-1} and these are referred as the wavelet coefficients. The simplest choice is to state that the missing data is contained merely in the the odd coefficients and therefore the coefficients of wavelet are $\gamma^k_{-1} = \lambda^{2k+1}_0$. Therefore all we are doing here is separating the even and odd indexed samples and this step is

called the SPLIT step. The code `Split.m` has been included to perform the above explained algorithm.

- **PREDICT:** As mentioned above, our aim is for obtaining the compact description of $\{\lambda_0^k\}$. In the situation when $\{\gamma_{-1}^k\}$ contains no information (e.g., the signal part is zero), we obviously can replace $\{\lambda_0^k\}$ with a small set $\{\lambda_{-1}^k\}$. But this situation hardly occurs in practice.

  Now if we have a predict operator P which does not depend on the signal such that

  $$\gamma_{-1}^k = P(\lambda_{-1}^k),$$

  then again $\{\lambda_{-1}^k\}$ is replaced by the set $\{\lambda_0^k\}$ (as the missing part is predicted by using the operator P).

  Again, in practice $P(\lambda_{-1}^k)$ may not be exactly equal to γ_{-1}^k. In spite of that $P(\lambda_{-1}^k)$ is probably be near to $\{\gamma_{-1}^k\}$. Therefore we can just save the data $\gamma_{-1}^k - P(\lambda_{-1}^k)$ (which is actually $\lambda_0^{2k+1} - P(\lambda_{-1}^k)$) and call this data as γ_{-1}^k again. Therefore we have

  $$\gamma_{-1}^k = \lambda_0^{2k+1} - P(\lambda_{-1}^k).$$

  If the predict operator P is good, then most of the terms $\lambda_0^{2k+1} - P(\lambda_{-1}^k)$ and hence γ_{-1}^k will be very small and therefore can be discarded. In this way we can have a representation of $\{\lambda_0^k\}$ as $\{\lambda_{-1}^k, \gamma_{-1}^k\}$ which is compact as γ_{-1}^k are small.

  The next task is to decide what this predict operator P should be? Simplest possible way is:

  $$P(\lambda_{-1}^k) = \frac{\lambda_{-1}^k + \lambda_{-1}^{k+1}}{2},$$

  and with this choice wavelet coefficients become

  $$\gamma_{-1}^k = \lambda_0^{2k+1} - \frac{\lambda_{-1}^k + \lambda_{-1}^{k+1}}{2}.$$

  These wavelet coefficients determine the extent to which the original signal is not linear. The prediction operator does not really need to be linear. We might attempt to discover the original signal failed to be cubic or any another highest order. Let us have an idea of how can we try to find the failure of the original signal to be cubic. Rather than define the new values as a linear interpolating the two neighboring values at the mid-point of two old values, the two neighboring values can be used on either side and we describe a polynomial $p(x)$ which is cubic that interpolates those four

values

$$\lambda_j^{k-1} = p(x_j^{k-1}), \lambda_j^k = p(x_j^k), \lambda_j^{k+1} = p(x_j^{k+1}), \lambda_j^{k+2} = p(x_j^{k+2}).$$

The new sampling value then will be evaluated as $p(x_{j+1}^{2k+1})$. In general we can fix an N (N is even and is called the 'no. of dual vanishing moment') and construct a polynomial of order $N-1$ (with the help of $\frac{N}{2}-1$ data points on either side). This polynomial will act as our predict operator. With this, our procedure becomes

$$\lambda_j^k = \lambda_{j+1}^{2k},$$

$$\gamma_j^k = \lambda_{j+1}^{2k+1} - p(x_{j+1}^{2k+1}),$$

as shown in Fig. (3.4).

If we use Lagrange interpolation, then we have

$$p(x_{j+1}^{2k+1}) = p_1\lambda_j^{k-1} + p_2\lambda_j^k + p_3\lambda_j^{k+1} + p_4\lambda_j^{k+2},$$

and our task is then to compute the coefficients p_i. These coefficients are called as **prediction coefficients**.

- **UPDATE:** There is problem in the coefficients λ's which we have computed in our split step. For example imagine the coefficients $1, 0, 1, 0, 1, 0, \cdots, 1$ at level $j+1$. This would result in the sequence $1, 1, 1, \cdots, 1$ at level j. This leads to horrible aliasing effect. To avoid this problem we would like some global properties of the original data to be preserved throughout all the levels. Idea is to find a better set $\{\lambda_j^k\}$ (instead of the set which we have obtained in splitting step) so that a certain scalar quantity $Q()$ is preserved, *i.e.*, $Q(\lambda_{j+1}^k) = Q(\lambda_j^k)$. For the betterment of the set $\{\lambda_j^k\}$ we construct an update operator U and update $\{\lambda_j^k\}$ as

$$\lambda_j^k = \lambda_j^k + U(\gamma_j^k). \tag{3.2.12}$$

Next question is how to construct this update operator U? The answer to the above question is that U should be constructed in such a way that the scalar quantity $Q()$ is preserved.

We can find out this operator U if we know, for updating the λ value how much of the γ coefficient is required. The updated values are called **update coefficients**. To compute the update coefficients we fix a positive even integer $\tilde{N}$ (called the 'no. of the primal vanishing moment') and the quantity which we want preserve is the $\tilde{N}$

vanishing moments of the wavelet, *i.e.*, we want to have

$$\begin{aligned} \int w(x)\psi(x)dx &= 0, \int xw(x)\psi(x)dx = 0, \int x^2 w(x)\psi(x) = 0, \cdots, \\ \int x^{\tilde{N}-1} w(x)\psi(x)dx &= 0, \end{aligned} \tag{3.2.13}$$

here $w(x)$ is the weighted function and note that $w(x) = 1$ if we are constructing wavelets for the space $\mathcal{L}_2([a, b])$. Another way of looking at the above property of preserving vanishing moments is the following:
At the level $j + 1$ we have the coefficients $\{\lambda_{j+1}^k\}$ representing some function f and at the j^{th} level we have the coefficients $\{\lambda_j^k, \gamma_j^k\}$ representing the same function, therefore

$$\sum_k \lambda_{j+1}^k \phi_{j+1}^k = \sum_k \lambda_j^k \phi_j^k + \sum_k \gamma_j^k \psi_j^k. \tag{3.2.14}$$

Multiplying the above equation with the weight function $w(x)$, with $x^p, p = 0, 1, \cdots, \tilde{N}-1$ and on integrating we get,

$$\sum_k \lambda_{j+1}^k \int x^p w(x)\phi_{j+1}^k dx = \sum_k \lambda_j^k \int x^p w(x)\phi_j^k dx + \sum_k \gamma_j^k \int x^p w(x)\psi_j^k dx.$$

Now using the Eqn. (3.2.13), we get

$$\sum_k \lambda_{j+1}^k \int x^p w(x)\phi_{j+1}^k dx = \sum_k \lambda_j^k \int x^p w(x)\phi_j^k dx, \quad p = 0, 1, \cdots, \tilde{N} - 1,$$

or

$$\sum_k \lambda_{j+1}^k M_{j+1,k}^p = \sum_k \lambda_j^k M_{j,k}^p, \quad p = 0, 1, \cdots, \tilde{N} - 1, \tag{3.2.15}$$

where $M_{j,k}^p = \int x^p w(x)\phi_j^k dx$ is called the p^{th} moment of the scaling function ϕ_j^k.

Following are the steps in the computation of the update coefficients are the following:

1. Initialize at first level the moments information for each coefficient. The zeroth initial moment, *i.e.*, $M_{j,k}^0 = \int w(x)\phi_j^k(x)dx$ is set equal to 1 for each k as we know that the area under the scaling function is always 1. For the high order moments the initial values are computed by utilizing the coefficient indexes which are represented in the following table

Initial moments using index k							
$M^0_{j,k}=1$	1	1	1	1	1	1	$\cdots$
$M^1_{j,k}=k$	0	1	2	3	4	5	$\cdots$
$M^2_{j,k}=k^2$	0	1	4	9	16	25	$\cdots$
$M^3_{j,k}=k^3$	0	1	8	27	64	125	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

2. Analyze the λ's that devoted for predicting the γ's and check that how much this contribution has been. (Prediction coefficients that were calculated at the prediction stage, yield these values).

3. At the present level, the moments are updated for each λ by employing the following equation

$$m^p_{j,k} = m^p_{j,k} + p_k \cdot m^p_{j,l},$$

where l is the indices related to the γ coefficient, k is the indices related to the λ coefficient, p_k is its analogous prediction coefficient and p is the updated moment.

4. Now a linear system will be constructed for finding the updated coefficients for each γ by using the property given in the Eqn. (3.2.15). The following are the steps for the construction of this system:

   (i) Fix 1 in a γ coefficient and 0 in the rest of $\gamma's$.

   (ii) To identify how much this γ is contributed to the λ's that updated it, implement an inverse transformation one step up and construct a linear system of $\tilde{N} \times N$ as below:

$$\begin{aligned}
c_1 \cdot M^0_{j,l_1} + c_2 \cdot M^0_{j,l_2} + \cdots + c_N \cdot M^0_{j,l_N} &= M^0_{j,g_j} \\
c_1 \cdot M^1_{j,l_1} + c_2 \cdot M^1_{j,l_2} + \cdots + c_N \cdot M^1_{j,l_N} &= M^1_{j,g_j} \\
c_1 \cdot M^2_{j,l_1} + c_2 \cdot M^2_{j,l_2} + \cdots + c_N \cdot M^2_{j,l_N} &= M^2_{j,g_j} \\
\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots & \\
c_1 \cdot M^{\tilde{N}-1}_{j,l_1} + c_2 \cdot M^{\tilde{N}-1}_{j,l_2} + \cdots + c_N \cdot M^{\tilde{N}-1}_{j,l_N} &= M^{\tilde{N}-1}_{j,g_j}.
\end{aligned}$$

   Here $c_i (1 \leq i \leq N)$ are the update coefficients which are to be computed, $l_i (1 \leq i \leq N)$ are the indices related to contribute the λ coefficients at the current level, $g_j (0 \leq j \leq$ `signal_length`) are the indices related to the γ with the value fix to 1.

   (iii) When $\tilde{N} = N$ (in this chapter we have assumed that $\tilde{N} = N$), the above

system can be solved for finding the update coefficients set for the γ whose value is fixed equal to 1. Note that if $\tilde{N} \neq N$, then there are another techniques available to compute the update coefficients (see [169] for details), however, to have the best of the wavelets it is recommended that N should be taken equal to $\tilde{N}$ [91].

5. For each γ, we have a series of N updated coefficients at each stage. These values are utilized for updating the λ coefficients before proceeding to the next stage.

The prediction and update steps will be discussed in detail for the space $\mathcal{L}_2([a, b])$ in the following sections.

3.3 Wavelets on the intervals

3.3.1 Prediction coefficients

Let us have an illustration of this case with an example. Suppose $N = 4$, then our task is to compute the prediction coefficients $p_i, 1 \leq i \leq 4$. The following cases are possible

1) Near the left boundary: 3 λ's on the right and 1 λ on the left (It should be noted that the splitting step is such that λ always remains on the first position).

2) Middle: 2 λ's on the right and 2 λ's on the left side.

3) Near the right boundary: Here we have the following two possibilities:
(3a) 1 λ on the right and 3 λ's on the left.
(3b) 0 λ on the right and 4 λ's on the left.

Due to symmetry, the prediction coefficients for the case 1 are the same as in the case 3a though in reverse order. Therefore totally we have 3 distinct cases (shown in Fig. (3.5)). In general we have $\frac{N}{2} + 1$ distinct cases ($\frac{N}{2}$ for the symmetrical boundary case and one for the middle case). Let us mark the four λs which are used in this case as $\lambda_0, \lambda_1, \lambda_2$ and λ_3 and assume that they are located at $0, 1, 2, 3$ (It should be noted that as there is an exclusive cubic polynomial interpolation, note that since there is a unique cubic polynomial interpolation, the same cubic interpolating function will be obtained no matter how the samples are separated).

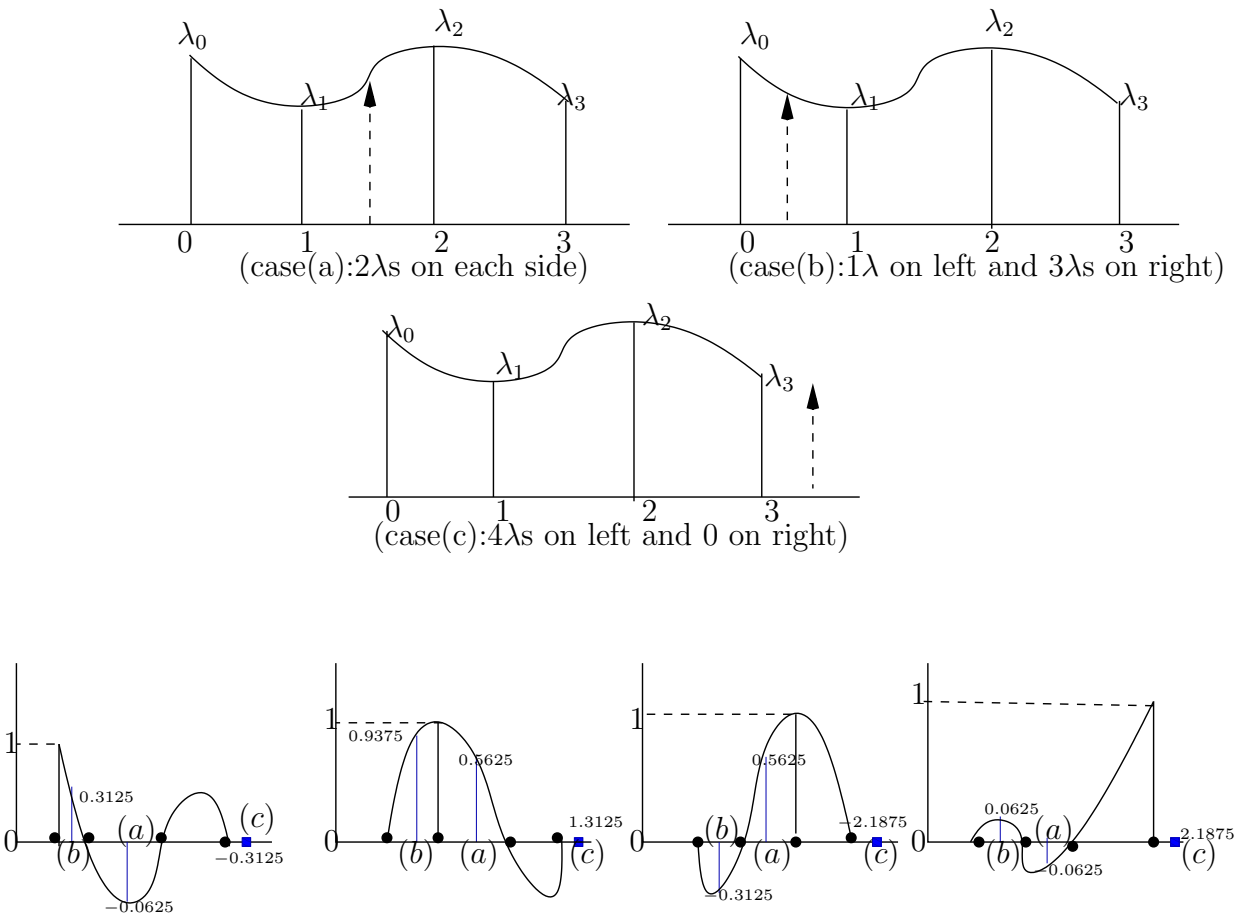


Figure 3.5: Cases for computation of prediction coefficients for $N = 4$.

In case of Lagrange interpolation, the interpolating cubic polynomial (the prediction operator) is

$$\begin{aligned} p(x) &= \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)}\lambda_0 + \frac{(x-0)(x-2)(x-3)}{(1-0)(1-2)(1-3)}\lambda_1 + \frac{(x-0)(x-1)(x-3)}{(2-0)(2-1)(2-3)}\lambda_2 \\ &+ \frac{(x-0)(x-1)(x-2)}{(3-0)(3-1)(3-2)}\lambda_3, \end{aligned} \tag{3.3.1}$$

and the prediction coefficients are such that

$$p(x_{j+1}^{2k+1}) = p_1\lambda_j^{k-1} + p_2\lambda_j^k + p_3\lambda_j^{k+1} + p_4\lambda_j^{k+2}. \tag{3.3.2}$$

Therefore, p_1 is nothing but $\frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)}$ (let us call it $P_1(x)$) computed at the desired point. Note that $P_1(x)$ is a function which is 1 at the node 0 and 0 at all other three nodes. Therefore we find $P_1(x)$ and calculate its value at desired points to compute p_1 for all the three cases (shown in Fig. (3.5)). Neville's algorithm has been used for this purpose. We have included in our package a function named `Prediction_coefficients_interval.m` which computes the prediction coefficients for a given N. This Matlab function implements

the above explained procedure for computation of the prediction coefficients. Tables (3.1, 3.2, 3.3) show the prediction coefficients for $N = 2, 4$ and 6 respectively computed using the Matlab function `Prediction_coefficients_interval.m`.

Cases		Coefficients			
#λs on left	#λs on the right	$k-3$	$k-1$	$k+1$	$k+3$
0	2			1.5	-0.5
1	1		0.5	0.5	
2	0	-0.5	1.5		

Table 3.1: Prediction coefficients for $N = 2$ for second generation wavelet on the interval.

Cases		Coefficients							
#λs on left	#λs on the right	$k-7$	$k-5$	$k-3$	$k-1$	$k+1$	$k+3$	$k+5$	$k+7$
0	4					2.1875	-2.1875	1.3125	-0.3125
1	3				0.3125	0.9375	-0.3125	0.0625	
2	2			-.0625	0.5625	0.5625	-0.0625		
3	1		0.0625	-0.3125	0.9375	0.3125			
4	0	-0.3125	1.3125	-2.1875	2.1875				

Table 3.2: Prediction coefficients for $N = 4$ for second generation wavelet on the interval.

Cases		Coefficients										
#λs on left	#λs on the right	$k-11$	$k-9$	$k-7$	$k-5$	$k-3$	$k-1$	$k+1$	$k+3$	$k+5$	$k+7$	$k+9$	$k+11$
0	6							2.7070	-4.5117	5.4141	-3.8672	1.5039	-0.2461
1	5						0.2461	1.2305	-0.8203	0.4922	-0.1758	0.0273	
2	4					-0.0273	0.4102	0.8203	-0.2734	0.0820	-0.0117		
3	3				0.0117	-0.0977	0.5859	0.5859	-0.0977	0.0117			
4	2			-0.0117	0.0820	-0.2734	0.8203	0.4102	-0.0273				
5	1		0.0273	-0.1758	0.4922	-0.8203	1.2305	0.2461					
6	0	-0.2461	1.5039	-3.8672	5.4141	-4.5117	2.7070						

Table 3.3: Prediction coefficients for $N = 6$ for second generation wavelet on the interval.

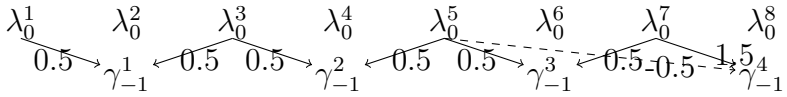
3.3.2 Update coefficients

We give an illustration of how to compute the update coefficients for a 1D signal having $N = 2$, length $L = 8$ and $\tilde{N} = 2$. At the beginning we have eight coefficients $\lambda_0^1, \lambda_0^2, \lambda_0^3, \lambda_0^4, \lambda_0^5, \lambda_0^6, \lambda_0^7, \lambda_0^8$ (assume that for the initial level $j = 0$). Follow the process described in the section (3.2.1), we have to go through the following steps for computing update coefficients:

1. Initialize the zeroth and first moments (because $\tilde{N} = 2$) for every coefficient at the first level.

Initial moments using index k								
$M^0_{0,k} = 1$	1	1	1	1	1	1	1	1
$M^1_{0,k} = k$	1	2	3	4	5	6	7	8

2. Analyze the λ's that devoted for predicting the γ's and check that how much that contribution has been.



- γ^1_{-1} uses λ^1_0 (contribution=0.5) and λ^3_0 (contribution=0.5) for its prediction.
- γ^2_{-1} uses λ^3_0 (contribution=0.5)and λ^5_0 (contribution=0.5) for its prediction.
- γ^3_{-1} uses λ^5_0 (contribution=0.5)and λ^7_0 (contribution=0.5) for its prediction.
- γ^4_{-1} uses λ^5_0 (contribution=-0.5) and λ^7_0 (contribution=1.5) for its prediction.

Therefore we have,

$$\gamma^1_{-1} = \gamma^1_{-1} - (0.5) \star \lambda^1_0 - (0.5) \star \lambda^3_0,$$

$$\gamma^2_{-1} = \gamma^2_{-1} - (0.5) \star \lambda^3_0 - (0.5) \star \lambda^5_0,$$

$$\gamma^3_{-1} = \gamma^3_{-1} - (0.5) \star \lambda^5_0 - (0.5) \star \lambda^7_0,$$

$$\gamma^4_{-1} = \gamma^4_{-1} - (-0.5) \star \lambda^5_0 - (1.5) \star \lambda^7_0,$$

3. Update the moments for every λ at the current level using the following equation

$$M^p_{0,k} = M^p_{0,k} + p_k \cdot M^p_{0,l},$$

where k is the index relative to a λ coefficient, p_k is its corresponding prediction coefficient, p is the moment being updated, and l is the index relative to a γ coefficient. λ^1_0 contributed only to γ^1_{-1} (corresponding to the index l=2) during the prediction step, therefore the corresponding moments get updated as follows

$$\begin{aligned} M^0_{0,1} &= M^0_{0,1} + p_1 \cdot M^0_{0,2} \\ &= 1 + 0.5 \cdot 1 \\ &= 1.5 \end{aligned} \qquad \begin{aligned} M^1_{0,1} &= M^1_{0,1} + p_1 \cdot M^1_{0,2} \\ &= 1 + 0.5 \cdot 2 \\ &= 2 \end{aligned}$$

λ^3_0 contributed to γ^1_{-1} and γ^2_{-1} (corresponding to the index $l = 2$ and $l = 4$ respec-

tively) during the prediction step, therefore the corresponding moments get updated as follows

$$\begin{aligned} M^0_{0,3} &= M^0_{0,3} + p_3 \cdot M^0_{0,2} \\ &= 1 + 0.5 \cdot 1 \\ &= 1.5 \\ M^0_{0,3} &= M^0_{0,3} + p_3 \cdot M^0_{0,4} \\ &= 1.5 + 0.5 \cdot 1 \\ &= 2.0 \end{aligned} \qquad \begin{aligned} M^1_{0,3} &= M^1_{0,3} + p_3 \cdot M^1_{0,2} \\ &= 3 + 0.5 \cdot 2 \\ &= 4 \\ M^1_{0,3} &= M^1_{0,3} + p_3 \cdot M^1_{0,4} \\ &= 4 + 0.5 \cdot 4 \\ &= 6 \end{aligned}$$

λ^5_0 contributed to $\gamma^2_{-1}, \gamma^3_{-1}$ and γ^4_{-1} (corresponding to the index $l = 4, 6$ and $l = 8$ respectively) during the prediction step, therefore the corresponding moments get updated as follows

$$\begin{aligned} M^0_{0,5} &= M^0_{0,5} + p_5 \cdot M^0_{0,4} \\ &= 1 + 0.5 \cdot 1 \\ &= 1.5 \\ M^0_{0,5} &= M^0_{0,5} + p_5 \cdot M^0_{0,6} \\ &= 1.5 + 0.5 \cdot 1 \\ &= 2.0 \\ M^0_{0,5} &= M^0_{0,5} + p_5 \cdot M^0_{0,8} \\ &= 2 + (-0.5) \cdot 1 \\ &= 1.5 \end{aligned} \qquad \begin{aligned} M^1_{0,5} &= M^1_{0,5} + p_5 \cdot M^1_{0,4} \\ &= 5 + 0.5 \cdot 4 \\ &= 7 \\ M^1_{0,5} &= M^1_{0,5} + p_5 \cdot M^1_{0,6} \\ &= 7 + 0.5 \cdot 6 \\ &= 10 \\ M^1_{0,5} &= M^1_{0,5} + p_5 \cdot M^1_{0,8} \\ &= 10 + (-0.5) \cdot 8 \\ &= 6 \end{aligned}$$

λ^7_0 contributed to γ^3_{-1} and γ^4_{-1} (corresponding to the index $l = 6$ and $l = 8$ respectively) during the prediction step, therefore the corresponding moments get updated as follows

$$\begin{aligned} M^0_{0,7} &= M^0_{0,7} + p_7 \cdot M^0_{0,6} \\ &= 1 + 0.5 \cdot 1 \\ &= 1.5 \\ M^0_{0,7} &= M^0_{0,7} + p_7 \cdot M^0_{0,8} \\ &= 1.5 + 1.5 \cdot 1 \\ &= 3.0 \end{aligned} \qquad \begin{aligned} M^1_{0,7} &= M^1_{0,7} + p_7 \cdot M^1_{0,6} \\ &= 7 + 0.5 \cdot 6 \\ &= 10 \\ M^1_{0,7} &= M^1_{0,7} + p_7 \cdot M^1_{0,8} \\ &= 10 + 1.5 \cdot 8 \\ &= 22 \end{aligned}$$

Updated moments				
k	1	3	5	7
$M^0_{0,k}$	$\frac{3}{2}$	2	$\frac{3}{2}$	3
$M^1_{0,k}$	2	6	6	22

4. Now we will formulate a linear system for finding the updated coefficients for each γ. The following are the steps for the construction of this system:

   (i) Put $\gamma^1_{-1} = 1$ coefficient and 0 in all the rest of $\gamma's$.

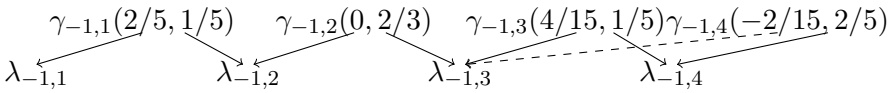
   (ii) To identify how much this γ^1_{-1} is contributed to the λ's that updated it, implement an inverse transformation one step up and construct a linear system of 2×2 as below,

$$\begin{aligned} c_1 \cdot M^0_{0,1} + c_2 \cdot M^0_{0,3} &= M^0_{0,2} \\ c_1 \cdot M^1_{0,1} + c_2 \cdot M^1_{0,3} &= M^1_{0,2}. \end{aligned}$$

   Putting the values of the updated moments computed in last step, we get

$$\begin{aligned} \frac{3}{2}c_1 + 2c_2 &= 1 \\ 2c_1 + 6c_2 &= 2. \end{aligned}$$

   (iii) Solving above system of equation gives us the update coefficients $c_1 = \frac{2}{5}$ and $c_2 = \frac{1}{5}$ corresponding to γ^1_{-1}. Similarly for γ^2_{-1} we get the update coefficients $c_1 = 0, c_2 = \frac{2}{3}$, for γ^3_{-1} we get the update coefficients $c_1 = \frac{4}{5}, c_2 = \frac{1}{5}$, and for γ^4_{-1} we get the update coefficients $c_1 = -\frac{2}{15}, c_2 = \frac{2}{5}$. We have the following situation



5. Now we have a set of 2 update coefficients for every γ at the level -1. These values are used to apply the update operator, U, to the λ^k_{-1} coefficients before proceeding to the next level. λ^1_{-1} uses $\frac{2}{5}$ from γ^1_{-1} for updating, i.e., we have

$$\lambda^1_{-1} = \lambda^1_{-1} + \frac{2}{5} \star \gamma^1_{-1}.$$

   Similarly we have

$$\lambda^2_{-1} = \lambda^2_{-1} + \frac{1}{5} \star \gamma^1_{-1} + 0 \star \gamma^2_{-1},$$

$$\lambda^3_{-1} = \lambda^3_{-1} + \frac{2}{3} \star \gamma^2_{-1} + \frac{4}{15} \star \gamma^3_{-1} - \frac{2}{15} \star \gamma^4_{-1},$$

$$\lambda^4_{-1} = \lambda^4_{-1} + \frac{1}{5} \star \gamma^3_{-1} + \frac{2}{5} \star \gamma^4_{-1},$$

Now we have to repeat the above procedure for the next step. The situation at the next step is the following

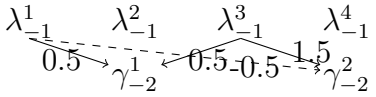
λ^1_{-1} (k =1)	λ^2_{-1} (k=3)	λ^3_{-1} (k=5)	λ^4_{-1} (k=7)
	γ^1_{-2}		γ^2_{-2}

The above procedure goes like this

1. Initial moments at the current level are:

	k=1	k=5
$M^0_{-1,k}$	$\frac{3}{2}$	$\frac{3}{2}$
$M^1_{-1,k}$	2	6

2. Check for the λ's that contributed to predicting every γ and note how much this contribution was.



   - γ^1_{-2} uses λ^1_{-1} (contribution=0.5) and λ^3_{-1} (contribution=0.5) for its prediction.
   - γ^2_{-2} uses λ^1_{-1} (contribution=-0.5)and λ^3_{-1} (contribution=1.5) for its prediction.

   Therefore we have,

$$\gamma^1_{-2} = \gamma^1_{-2} - (0.5) \star \lambda^1_{-1} - (0.5) \star \lambda^3_{-1},$$

$$\gamma^2_{-2} = \gamma^2_{-2} - (0.5) \star \lambda^1_{-1} - (0.5) \star \lambda^3_{-1},$$

3. We have to update the moments of λ^1_{-1} and λ^3_{-1}. λ^1_{-1} contributed to γ^1_{-2} (corresponding to the initial index l=3) and γ^2_{-2} (corresponding to the initial index l=7) during the prediction step, therefore the corresponding moments get updated as follows

$$\begin{aligned} M^0_{-1,1} &= M^0_{-1,1} + p_1 \cdot M^0_{-1,3} \\ &= 1.5 + 0.5 \star 2 \\ &= 2.5 \end{aligned} \qquad \begin{aligned} M^1_{-1,1} &= M^1_{-1,1} + p_1 \cdot M^1_{-1,3} \\ &= 2 + 0.5 \cdot 6 \\ &= 5 \end{aligned}$$

$$\begin{aligned} M^0_{-1,1} &= M^0_{-1,1} + p_1 \cdot M^0_{-1,7} \\ &= 2.5 + (-0.5) \cdot 3 \\ &= 1.0 \end{aligned} \qquad \begin{aligned} M^1_{-1,1} &= M^1_{-1,1} + p_1 \cdot M^1_{-1,7} \\ &= 5 + (-0.5) \cdot 22 \\ &= -6 \end{aligned}$$

λ^3_{-1} contributed to γ^1_{-2} and γ^2_{-2} (corresponding to the index $l = 3$ and $l = 7$ respectively) during the prediction step, therefore the corresponding moments get updated as follows

$$\begin{aligned} M^0_{-1,5} &= M^0_{-1,5} + p_1 \cdot M^0_{-1,3} & M^1_{-1,5} &= M^1_{-1,5} + p_1 \cdot M^1_{-1,3} \\ &= 1.5 + 0.5 \cdot 2 & &= 6 + 0.5 \cdot 6 \\ &= 2.5 & &= 9 \\ M^0_{-1,5} &= M^0_{-1,5} + p_1 \cdot M^0_{-1,7} & M^1_{-1,5} &= M^1_{-1,5} + p_1 \cdot M^1_{-1,7} \\ &= 1.5 + 1.5 \cdot 3 & &= 9 + 1.5 \cdot 22 \\ &= 7.0 & &= 42 \end{aligned}$$

Updated moments		
k	1	5
$M^0_{-1,k}$	1	7
$M^1_{-1,k}$	-6	42

4. Now a linear system will be constructed for finding the updated coefficients for each γ. The following are the steps for the construction of this system:

   (i) Put $\gamma^1_{-2} = 1$ and 0 in the rest of $\gamma's$.

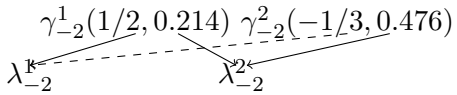
   (ii) To identify how much this γ^1_{-1} is contributed to the λ's that updated it, implement an inverse transformation one step up and construct a linear system of 2×2 as below,

$$\begin{aligned} c_1 \cdot M^0_{-1,1} + c_2 \cdot M^0_{-1,5} &= M^0_{-1,3} \\ c_1 \cdot M^1_{-1,1} + c_2 \cdot M^1_{-1,5} &= M^1_{-1,3}. \end{aligned}$$

   Putting the values of the updated moments computed in last step, we get

$$\begin{aligned} c_1 + 7c_2 &= 2 \\ -6c_1 + 42c_2 &= 6. \end{aligned}$$

   (iii) Solving above system of equation gives us the update coefficients $c_1 = \frac{1}{2}$ and $c_2 = 0.214$ corresponding to γ^1_{-2}. Similary for γ^2_{-2} we get the update coefficients $c_1 = -\frac{1}{3}, c_2 = 0.476$. We have the following situation



5. Now we have a set of 2 update coefficients for every γ at the level -2. These values are used to apply the update operator, U, to the λ^k_{-2} coefficients before proceeding to the next level. λ^1_{-1} uses $\frac{1}{2}$ from γ^1_{-2} for updating, *i.e.*, we have

$$\lambda^1_{-2} = \lambda^1_{-2} + \frac{1}{2} \star \gamma^1_{-2} - \frac{1}{3} \star \gamma^2_{-2}.$$

Similarly we have

$$\lambda^2_{-2} = \lambda^2_{-2} + 0.214 \star \gamma^1_{-2} + 0.476 \star \gamma^2_{-2}.$$

We have included in our package a function named `Update_coefficients_interval.m` which computes the update coefficients for a given $\tilde{N}$ by implementing the above explained procedure.

Having computed the predicted coefficients and update coefficients, we have implemented the above steps for DWT and include in package function named `Wavelet_transform.m`. Also, we have implemented the reverse steps for IDWT and include in our package function named `Wavelet_transform_inverse.m`.

3.3.3 Computing the values of $\phi(x)$ and $\psi(x)$

In the space V_0, the coefficients which are used to represent the basic scaling function ϕ^0_0 are $\{v_0(n)\}_n = \{\cdots, 0, 1, 0 \cdots\} = \delta_n$. Using IDWT (with all detail coefficients $w_i(n) = 0$), the representation of the scaling function in any space $V_i, i > 0$ can be computed. For $i \to \infty$, the coefficients $\{v_i\}_n$ approach the sampling $\{\phi^0_0(n2^{-i})\}_n$.

In the detail space W_0, the coefficients which are used to represent the basic wavelet ψ^0_0 are $\{w_0(n)\}_n = \{\cdots, 0, 1, 0 \cdots\} = \delta_n$. Using the IDWT (with all coefficients $v_0(n) = 0$), the representation of the scaling function in V_1 can be computed. Again by using the cascade algorithm the representation in any space $V_i, i > 1$ can be computed. For $i \to \infty$, the coefficients $\{v_i(n)\}_n$ approach the sampling $\{\psi^0_0(n2^{-i})\}_n$.

3.4 Second generation wavelet optimized finite difference method (SGWOFD)

There are many point of views of looking at the Burger's equation. For example it can be seen as an equation which models turbulence, traffic flows, acoustic transmission in fog etc. It can also be seen as a perturbation of the linear heat equation, which is the angle which has been considered in this chapter. We consider a rod made up of heat conducting material. Let L be the length of a rod and A be the uniform cross sectional area of a rod. It is assumed that the heat can be exchanged only through the ends and the lateral surface of the rod is insulated. Further it is assumed that the thickness of the rod is such that the temperature of the rod is constant throughout any given cross section.

Let us represent the temperature at a time t and position x by $u(t,x)$ and following are the parameters of the material of the rod: k, the thermal conductivity; $\rho(x)$, the density of the material at point x; the specific heat $c(x)$ of a rod at a point x, $f_1(t,x)$, heat production rate per unit time. With these parameters the equation governing the temperature inside the rod is given by

$$\rho(x)c(x)\frac{\partial u(t,x)}{\partial t} = \frac{\partial}{\partial x}\left(k(x)\frac{\partial}{\partial x}u(t,x)\right) + f_1(t,x). \tag{3.4.1}$$

We further suppose that the specific heat, density and the thermal conductivity are independent of the position; with this assumption, the Eqn. (3.4.1) becomes

$$\frac{\partial u(t,x)}{\partial t} = \frac{k}{\rho c}\frac{\partial^2 u(t,x)}{\partial x^2} + \frac{f_1(t,x)}{\rho c}.$$

Letting $\nu = \frac{k}{\rho c}$ and $f(t,x) = \frac{f_1(t,x)}{\rho c}$, the above equation becomes

$$\frac{\partial u(t,x)}{\partial t} = \nu\frac{\partial^2 u(t,x)}{\partial x^2} + f(t,x). \tag{3.4.2}$$

Now we consider a perturbation of Eqn. (3.4.2) as follows

$$\frac{\partial u(t,x)}{\partial t} + u(t,x)\frac{\partial u(t,x)}{\partial x} = \nu\frac{\partial^2 u(t,x)}{\partial x^2} + f(t,x). \tag{3.4.3}$$

Eqn. (3.4.3) is Burger's equation with ν as the viscosity constant. Since this equation is 2nd order equation in space and 1st order equation in time, we need one initial $u(0,x) = u_0(x)$ and two boundaries for solving the same. There are four common types of boundary conditions

1. Periodic:
$$u(x,t) = u(x + (b-a), t).$$

2. Dirichlet:
$$u(a,t) = u_L(t), \quad u(b,t) = u_R(t).$$

3. Neumann:
$$\frac{\partial u(a,t)}{\partial x} = u_L(t), \quad \frac{\partial u(b,t)}{\partial x} = u_R(t).$$

4. Robin's:
$$c_1 u(a,t) + c_2 \frac{\partial u(a,t)}{\partial x} = u_L(t), \quad d_1 u(0,t) + d_2 \frac{\partial u(L,t)}{\partial x} = u_R(t).$$

3.4.1 Finite difference matrices for different boundary conditions

Let the points discretizing the domain $[a,b]$ of the problem be $a = x_1 < x_2 < x_3 < \cdots < x_N = b$ (this discretization need not be uniform). Using the central differences, the 1st and the 2nd derivatives at different points of the domain are defined as

$$\begin{aligned}
\frac{du(x_1)}{dx} &= \frac{u(x_2) - u(x_0)}{x_2 - x_0}, \\
\frac{du(x_2)}{dx} &= \frac{u(x_3) - u(x_1)}{x_3 - x_1}, \\
\vdots &= \vdots \\
\frac{du(x_{N-1})}{dx} &= \frac{u(x_N) - u(x_{N-2})}{x_N - x_{N-2}}, \\
\frac{du(x_N)}{dx} &= \frac{u(x_{N+1}) - u(x_{N-1})}{x_{N+1} - x_{N-1}}.
\end{aligned} \tag{3.4.4}$$

$$\begin{aligned}
\frac{d^2u(x_1)}{dx^2} &= \frac{2u(x_0)}{(x_0-x_1)(x_0-x_2)} + \frac{2u(x_1)}{(x_1-x_0)(x_1-x_2)} + \frac{2u(x_2)}{(x_2-x_0)(x_2-x_1)}, \\
\frac{d^2u(x_2)}{dx^2} &= \frac{2u(x_1)}{(x_1-x_2)(x_1-x_3)} + \frac{2u(x_2)}{(x_2-x_1)(x_2-x_3)} + \frac{2u(x_3)}{(x_3-x_1)(x_3-x_2)}, \\
\vdots &= \vdots \\
\frac{d^2u(x_{N-1})}{dx^2} &= \frac{2u(x_{N-2})}{(x_{N-2}-x_{N-1})(x_{N-2}-x_N)} + \frac{2u(x_{N-1})}{(x_{N-1}-x_{N-2})(x_{N-1}-x_N)} + \frac{2u(x_N)}{(x_N-x_{N-1})(x_N-x_{N-2})}, \\
\frac{d^2u(x_N)}{dx^2} &= \frac{2u(x_{N-1})}{(x_{N-1}-x_N)(x_{N-1}-x_{N+1})} + \frac{2u(x_N)}{(x_N-x_{N-1})(x_N-x_{N+1})} + \frac{2u(x_{N+1})}{(x_{N+1}-x_{N-1})(x_{N+1}-x_N)}.
\end{aligned} \tag{3.4.5}$$

Note that x_0 and x_{N+1} are the points which are not included in our discretization. Depending on the types of the boundary conditions involved we have different finite difference matrices as explained below:

1. Periodic boundary conditions: In this case, all the indices are supposed to be calculated modulo $N-1$ (as $u(a) = u(b)$). Therefore, we treat x_0 as x_{N-1} and x_{N+1} as x_2. Further, we assume that $x_1 - x_0 = x_2 - x_1$ and $x_{N+1} - x_N = x_N - x_{N-1}$. Therefore in this case, the first and second order finite difference matrices $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$ such that $[u'(x_1), u'(x_2), \cdots, u'(x_N)]' = \mathcal{D}^{(1)}[u(x_1), u(x_2), \cdots, u(x_N)]'$ and $[u''(x_1), u''(x_2), \cdots, u''(x_N)]' = \mathcal{D}^{(2)}[u(x_1), u(x_2), \cdots, u(x_N)]'$ are given by

$$\begin{aligned}
\mathcal{D}^{(1)}(1,2) = -\mathcal{D}^{(1)}(1,N-1) &= \frac{1}{2(x_2 - x_1)}, \\
\mathcal{D}^{(1)}(N,2) = -\mathcal{D}^{(1)}(N,N-1) &= \frac{1}{2(x_N - x_{N-1})}, \\
\mathcal{D}^{(1)}(i,i-1) = \mathcal{D}^{(1)}(i,i+1) &= -\frac{1}{x_{i+1} - x_{i-1}}, \\
\mathcal{D}^{(1)}(i,j) &= 0 \quad \forall \text{ other } i \text{ and } j.
\end{aligned}$$

$$\begin{aligned}
-\frac{1}{2}\mathcal{D}^{(2)}(1,1) = \mathcal{D}^{(2)}(1,2) = \mathcal{D}^{(2)}(1,N-1) &= \frac{1}{(x_2 - x_1)^2}, \\
\mathcal{D}^{(2)}(N,2) = \mathcal{D}^{(2)}(N,N-1) = -\frac{1}{2}\mathcal{D}^{(2)}(N,N) &= \frac{1}{(x_N - x_{N-1})^2}, \\
\mathcal{D}^{(2)}(i,i-1) &= \frac{2}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})}, \quad 2 \le i \le N-1, \\
\mathcal{D}^{(2)}(i,i) &= \frac{2}{(x_i - x_{i-1})(x_i - x_{i+1})}, \quad 2 \le i \le N-1, \\
\mathcal{D}^{(2)}(i,i+1) &= \frac{2}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}, \quad 2 \le i \le N-1, \\
\mathcal{D}^{(2)}(i,j) &= 0, \quad \forall \text{ other } i \text{ and } j.
\end{aligned} \tag{3.4.6}$$

2. Dirichlet boundary conditions: In the condition of Dirichlet boundaries, the function is specified at the end points, *i.e.*, $u(a) = u_L$ and $u(b) = u_R$ which imply $u(x_1) = u_L$ and $u(x_N) = u_R$. As the solution is known at the end points, we only have $N-2$ unknowns namely $u(x_2), u(x_3), \cdots, u(x_{N-1})$. Therefore, we consider all but first and last equations of (3.4.4) and (3.4.6) and write $u(x_1) = u_L, u(x_N) = u_R$. We obtain

the following

$$\begin{pmatrix} u'(x_2) \\ u'(x_3) \\ \vdots \\ u'(x_{N-2}) \\ u'(x_{N-1}) \end{pmatrix} = \mathcal{D}^{(1)} \begin{pmatrix} u(x_2) \\ u(x_3) \\ \vdots \\ u(x_{N-2}) \\ u(x_{N-1}) \end{pmatrix} + \begin{pmatrix} -\frac{u_L}{x_3-x_1} \\ 0 \\ \vdots \\ 0 \\ \frac{u_R}{x_N-x_{N-2}} \end{pmatrix}, \tag{3.4.7}$$

$$\begin{pmatrix} u''(x_2) \\ u''(x_3) \\ \vdots \\ u''(x_{N-2}) \\ u''(x_{N-1}) \end{pmatrix} = \mathcal{D}^{(2)} \begin{pmatrix} u(x_2) \\ u(x_3) \\ \vdots \\ u(x_{N-2}) \\ u(x_{N-1}) \end{pmatrix} + \begin{pmatrix} \frac{2u_L}{(x_1-x_2)(x_1-x_3)} \\ 0 \\ \vdots \\ 0 \\ \frac{2u_R}{(x_N-x_{N-1})(x_N-x_{N-2})} \end{pmatrix}, \tag{3.4.8}$$

here $\mathcal{D}^{(1)}$ and $\mathcal{D}^{(2)}$ are the matrices of $(N-2)\times(N-2)$ size with following entries

$$\begin{aligned} \mathcal{D}^{(1)}(1,2) &= \frac{1}{(x_3-x_1)}, & \mathcal{D}^{(1)}(N-2,N-3) &= -\frac{1}{(x_N-x_{N-2})}, \\ \mathcal{D}^{(1)}(i,i-1) &= -\frac{1}{x_{i+2}-x_i}, & \mathcal{D}^{(1)}(i,i+1) &= \frac{1}{x_{i+2}-x_i}, 2 \le i \le N-3, \\ \mathcal{D}^{(1)}(i,j) &= 0 \quad \forall \text{ other } i \text{ and } j. \end{aligned}$$

$$\begin{aligned} \mathcal{D}^{(2)}(1,1) &= \frac{2}{(x_2-x_1)(x_2-x_3)}, \\ \mathcal{D}^{(2)}(1,2) &= \frac{2}{(x_3-x_1)(x_3-x_2)}, \\ \mathcal{D}^{(2)}(N-2,N-3) &= \frac{2}{(x_{N-2}-x_{N-1})(x_{N-2}-x_N)}, \\ \mathcal{D}^{(2)}(N-2,N-2) &= \frac{2}{(x_{N-1}-x_{N-2})(x_{N-1}-x_N)}, \\ \mathcal{D}^{(2)}(i,i-1) &= \frac{2}{(x_i-x_{i+1})(x_i-x_{i+2})}, \quad 2 \le i \le N-3, \\ \mathcal{D}^{(2)}(i,i) &= \frac{2}{(x_{i+1}-x_i)(x_{i+1}-x_{i+2})}, \quad 2 \le i \le N-3, \\ \mathcal{D}^{(2)}(i,i+1) &= \frac{2}{(x_{i+2}-x_i)(x_{i+2}-x_{i+1})}, \quad 2 \le i \le N-3, \\ \mathcal{D}^{(2)}(i,j) &= 0, \quad \forall \text{ other } i \text{ and } j. \end{aligned} \tag{3.4.9}$$

3. Neumann boundary conditions: In the situation of Neumann boundaries, the func-

tion's first derivative at the end points is specified, i.e., $u'(a) = u_L$ and $u'(b) = u_R$. Therefore, we already have the values of $u'(x_1) = u'(a) = u_L$ and $u'(x_N) = u'(b) = u_R$. Furthermore, we use the fact that $u'(x_1) = \frac{u(x_2)-u(x_1)}{x_2-x_1} = u_L \implies u(x_1) = u(x_2) - u_L(x_2 - x_1)$ and similarly $u(x_N) = u(x_{N-1}) + u_R(x_N - x_{N-1})$. Furthermore, we use the fact that $x_1 - x_0 = x_2 - x_1$ and $x_{N+1} - x_N = x_N - x_{N-1}$. Therefore, we have the following

$$\begin{pmatrix} u'(x_2) \\ u'(x_3) \\ \vdots \\ u'(x_{N-2}) \\ u'(x_{N-1}) \end{pmatrix} = \mathcal{D}^{(1)} \begin{pmatrix} u(x_2) \\ u(x_3) \\ \vdots \\ u(x_{N-2}) \\ u(x_{N-1}) \end{pmatrix} + \begin{pmatrix} \frac{u_L(x_2-x_1)}{x_3-x_1} \\ 0 \\ \vdots \\ 0 \\ \frac{u_R(x_N-x_{N-1})}{x_N-x_{N-2}} \end{pmatrix}, \tag{3.4.10}$$

$$\begin{pmatrix} u''(x_1) \\ u''(x_2) \\ \vdots \\ u''(x_{N-1}) \\ u''(x_N) \end{pmatrix} = \mathcal{D}^{(2)} \begin{pmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} + \begin{pmatrix} \frac{-2u_L}{(x_2-x_1)} \\ 0 \\ \vdots \\ 0 \\ \frac{2u_R}{(x_N-x_{N-1})} \end{pmatrix}, \tag{3.4.11}$$

where $\mathcal{D}^{(1)}$, $\mathcal{D}^{(2)}$ are the matrices of size $(N-2) \times (N-2)$ and $N \times N$ with the following entries respectively

$$\begin{aligned} \mathcal{D}^{(1)}(1,1) &= -\frac{1}{(x_3 - x_1)}, & \mathcal{D}^{(1)}(1,2) &= \frac{1}{(x_3 - x_1)}, \\ \mathcal{D}^{(1)}(N-2, N-3) &= \frac{1}{(x_N - x_{N-2})}, & \mathcal{D}^{(1)}(N-2, N-2) &= -\frac{1}{(x_N - x_{N-2})}, \\ \mathcal{D}^{(1)}(i, i-1) &= -\frac{1}{x_{i+2} - x_i}, & \mathcal{D}^{(1)}(i, i+1) &= \frac{1}{x_{i+2} - x_i}, 2 \le i \le N-3, \\ \mathcal{D}^{(1)}(i,j) &= 0 \quad \forall \text{ other } i \text{ and } j. \end{aligned}$$

$$\begin{aligned} \mathcal{D}^{(2)}(1,1) &= \frac{-2}{(x_2 - x_1)^2}, \\ \mathcal{D}^{(2)}(1,2) &= \frac{2}{(x_2 - x_1)^2}, \\ \mathcal{D}^{(2)}(N, N-1) &= \frac{2}{(x_N - x_{N-1})^2}, \\ \mathcal{D}^{(2)}(N,N) &= \frac{-2}{(x_N - x_{N-1})^2}, \end{aligned}$$

$$\mathcal{D}^{(2)}(i,i-1) = \frac{2}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}, \quad 2 \le i \le N-1, \tag{3.4.12}$$
$$\mathcal{D}^{(2)}(i,i) = \frac{2}{(x_i-x_{i-1})(x_i-x_{i+1})}, \quad 2 \le i \le N-1,$$
$$\mathcal{D}^{(2)}(i,i+1) = \frac{2}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}, \quad 2 \le i \le N-1,$$
$$\mathcal{D}^{(2)}(i,j) = 0, \quad \forall \text{ other } i \text{ and } j.$$

4. Robin's boundary conditions: In case of Robin's boundary conditions we have $c_1u(x_1)+c_2u'(x_1) = u_L$ and $d_1u(x_N)+d_2u'(x_N) = u_R$. Therefore, we have $u'(x_1) = -\frac{c_1}{c_2}u(x_1)+\frac{u_L}{c_2}$ and $u'(x_N) = -\frac{d_1}{d_2}u(x_N)+\frac{u_R}{d_2}$. The first derivative at all the other nodes is same as given in Eqn. (3.4.4). Therefore, we get the following

$$\begin{pmatrix} u'(x_1) \\ u'(x_2) \\ \vdots \\ u'(x_{N-1}) \\ u'(x_N) \end{pmatrix} = \mathcal{D}^{(1)} \begin{pmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} + \begin{pmatrix} \frac{u_L}{c_2} \\ 0 \\ \vdots \\ 0 \\ \frac{u_R}{d_2} \end{pmatrix}, \tag{3.4.13}$$

here $\mathcal{D}^{(1)}$ is a matrix of $N \times N$ size with the following values

$$\mathcal{D}^{(1)}(1,1) = -\frac{c_1}{c_2}, \qquad \mathcal{D}^{(1)}(N,N) = -\frac{d_1}{d_2}$$
$$\mathcal{D}^{(1)}(i,i-1) = -\frac{1}{x_{i+1}-x_{i-1}}, \qquad \mathcal{D}^{(1)}(i,i+1) = \frac{1}{x_{i+1}-x_{i-1}}, 2 \le i \le N-1,$$
$$\mathcal{D}^{(1)}(i,j) = 0 \quad \forall \text{ other } i \text{ and } j.$$

For computing the second derivative, we write $c_1u(x_1) + c_2u'(x_1) = u_L$ as $c_1u(x_1) + c_2\frac{(u(x_2)-u(x_0))}{(x_2-x_0)} = u_L$ which implies $u(x_0) = u(x_2) - \frac{(u_L-c_1u(x_1))(x_2-x_0)}{c_2}$. Similarly we obtain $u(x_{N+1}) = u(x_{N-1}) + \frac{(u_R-d_1u(x_N))(x_{N+1}-x_{N-1})}{d_2}$. Furthermore, we again use $x_1 - x_0 = x_2 - x_1$ and $x_{N+1} - x_N = x_N - x_{N-1}$. We get the following

$$\begin{pmatrix} u''(x_1) \\ u''(x_2) \\ \vdots \\ u''(x_{N-1}) \\ u''(x_{N-2}) \end{pmatrix} = \mathcal{D}^{(2)} \begin{pmatrix} u(x_1) \\ u(x_2) \\ \vdots \\ u(x_{N-1}) \\ u(x_N) \end{pmatrix} + \begin{pmatrix} \frac{-2u_L}{c_2(x_2-x_1)} \\ 0 \\ \vdots \\ 0 \\ \frac{2u_R}{d_2(x_N-x_{N-1})} \end{pmatrix}, \tag{3.4.14}$$

here $\mathcal{D}^{(2)}$ is a matrix of $N \times N$ size with the following values

$$\begin{aligned}
\mathcal{D}^{(2)}(1,1) &= \frac{2}{(x_2 - x_1)}\left(\frac{c_1}{c_2} + \frac{1}{x_1 - x_2}\right), \\
\mathcal{D}^{(2)}(1,2) &= \frac{2}{(x_2 - x_1)^2}, \\
\mathcal{D}^{(2)}(N, N-1) &= \frac{2}{(x_N - x_{N-1})^2}, \\
\mathcal{D}^{(2)}(N,N) &= \frac{-2}{(x_N - x_{N-1})^2}\left(\frac{d_1}{d_2} + \frac{1}{x_N - x_{N-1}}\right), \\
\mathcal{D}^{(2)}(i, i-1) &= \frac{2}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})}, \quad 2 \le i \le N-1, \\
\mathcal{D}^{(2)}(i,i) &= \frac{2}{(x_i - x_{i-1})(x_i - x_{i+1})}, \quad 2 \le i \le N-1, \\
\mathcal{D}^{(2)}(i, i+1) &= \frac{2}{(x_{i+1} - x_{i-1})(x_{i+1} - x_i)}, \quad 2 \le i \le N-1, \\
\mathcal{D}^{(2)}(i,j) &= 0, \quad \forall \text{ other } i \text{ and } j.
\end{aligned} \tag{3.4.15}$$

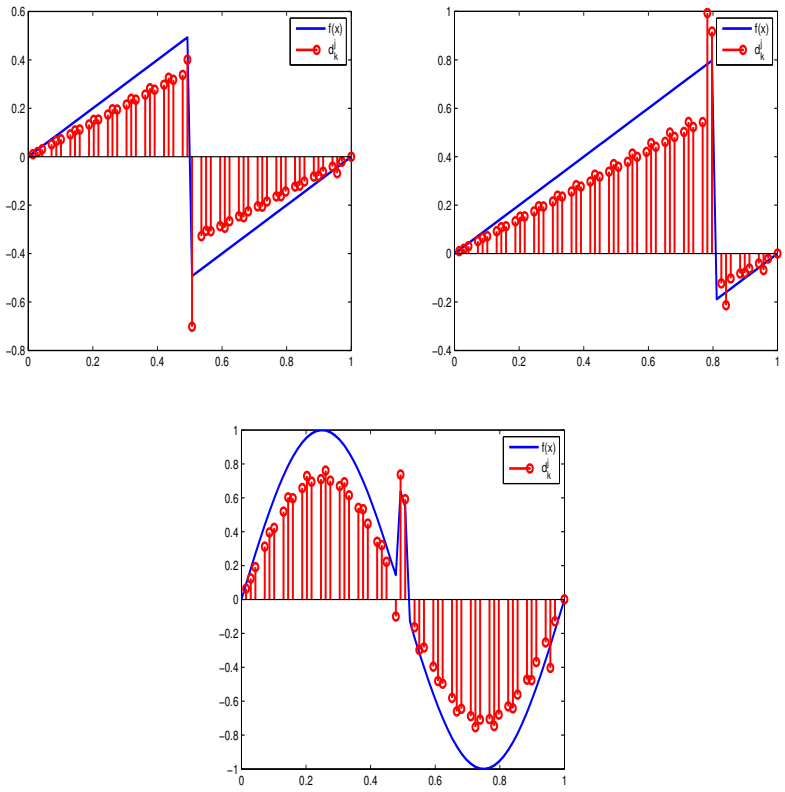


Figure 3.6: d_k^j for different functions (a) Sawtooth function with discontinuity at $x = 0.5$ (b) Sawtooth function with discontinuity at $x = 0.8$ (c) $f(x) = \sin(2\pi x) + \exp(-10^4(x - 0.5)^2)$.

3.4.2 Generation of adaptive grid based on second generation wavelet

To best of our knowledge, we don't have derivatives for second generation wavelets (it can be a future direction). Because of lack of derivatives, use of second generation wavelets in collocation and Galerkin sense is not possible (Galerkin approach solves the equations in wavelet coefficient domain while the collocation approach solves the equations in physical domain on the computational adaptive grid. The estimation of the non-linear quantities in collocation techniques are carried out in the physical space, similar to the Galerkin techniques). Hence the use of second generation wavelets for PDEs can be seen only in wavelet optimized sense. Wavelet optimized approach involve algorithms that relies upon the traditional discretizations (*e.g.*, finite volume, finite difference or finite element) while uses the wavelet analysis [170] for defining the computational grid. In this case computational overhead is decreased since one have to deal with point values in the physical description.

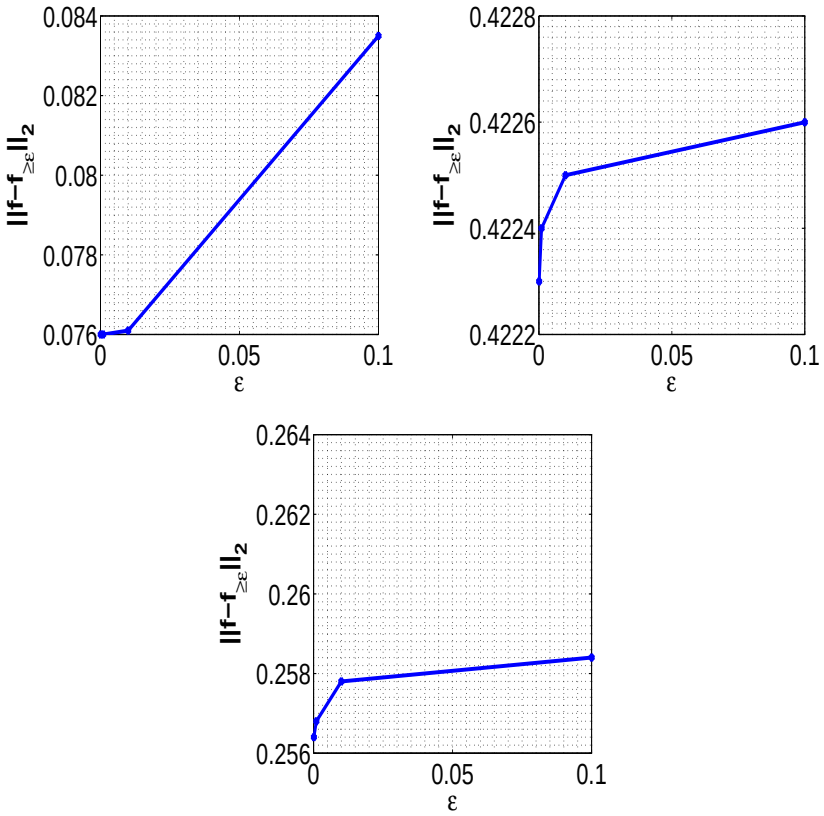


Figure 3.7: $\|f - f_{\geq\epsilon}\|_2$ versus ϵ for $f(x) = x$ for $0 \leq x < 0.5$ and $= x - 1$, otherwise; $f(x) = x$ for $0 \leq x < 0.8$ and $= x - 1$, otherwise; $f(x) = \sin(2\pi x) + \exp(-10^4(x - 0.5)^2)$.

As already discussed, one of the useful features of wavelet is that the coefficients of wavelet

i.e., d_k^j decline quickly where exercises are smooth and for a function which has discontinuities in one of its derivatives then its coefficients of wavelet decline moderately near the discontinuity point and keep a rapid decline far from it. This feature of wavelets makes it suitable for detecting where the shocks are in the PDE's numerical solution. Therefore, the construction of the adaptive grid depends on the wavelet coefficients magnitude.

For the case of second generation wavelet, if $\{t_j\}_{j=1}^J$ is the set of scales, then d_k^j's follow the above mentioned fact rigorously as j goes close to J. Fig. (3.6) shows d_k^j for different functions. Hence, d_k^j's has been considered (coefficients of wavelet at scale t_j) for adapting the grid.

Before using any wavelet for the generation of adaptive grid, it is recommended that the same is tested for the reconstruction error. Given any function f, $f_{reconstructed} = IDWT(DWT(f))$, then the error $\|f - f_{reconstructed}\|$ is termed as reconstruction error. The Matlab code `Reconstruction.m` has been included in our suite to test the second generation wavelet for reconstruction. The reconstruction error obtained is of the order 10^{-15} which is quite acceptable.

There are two wavelet based techniques for the construction of the adaptive grid namely standard adaptation approach and the modified adaptation approach. In standard adaptation approach, one has to start with the finest resolution level and has to remove the grid points depending on the magnitude of the coefficients of wavelet. In the case of the modified adaptation technique, one must start from the lowest resolution level and add the grid points depending on the magnitude of the coefficients of wavelet. Although the standard and the modified approaches differ substantially, the resulting adaptive node arrangements can be shown numerically to be nearly the same. We have included the functions in our package, named `AdaptiveGrid_SGW_modified.m` and `AdaptiveGrid_SGW_standard.m` which constructs the adaptive grid by using modified adaptation technique and standard adaptation technique respectively.

In this chapter, the modified adaptation technique has been used. We begin with the lowest level, j_0 and call the corresponding set of vertices V as $X^{initial}$ (stands for the initial grid). Suppose $f(x_i)_{x_i \in X^{initial}}$ is the set of the values in $X^{initial}$, perform second generation wavelet transformation on this set for obtaining the wavelet and scaling function coefficients. The finer grid X^{new} will be constructed from $X^{initial}$. Since each $x_k \in X^{initial}$ is associated with a scaling function ϕ_k, all the points in $X^{initial}$ will be kept in X^{new}. Moreover, every point in $X^{initial}$ is uniquely associated with the function of second-generation wavelet at the scale t_{J_0}. Analyze $d_k^{J_0}$ coefficients, if $|d_k^{J_0}| \geq \epsilon$ (where ϵ is the pre-decided threshold parameter), then the corresponding grid point $x_k \in X^{initial}$ is termed as active grid point. Corresponding to each point of active grid, x_k, a safety zone [150] will be added in X^{new}.

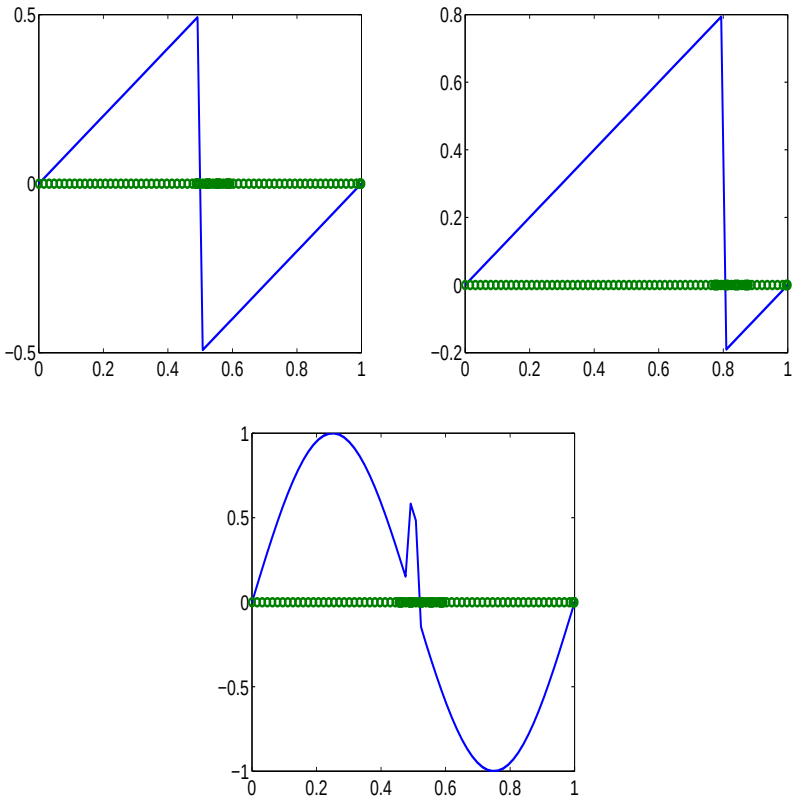


Figure 3.8: The adaptive grid for (a) Sawtooth function with discontinuity at $x = 0.5$ (b) Sawtooth function with discontinuity at $x = 0.8$ (c) $f(x) = \sin(2\pi x) + \exp(-10^4(x - 0.5)^2)$.

Safety zone not only makes the grid finer near the points of singularities but also takes care of the coefficients of wavelet which may become significant during the time when we are not refining the grid.

For adding the safety zone, a parameter M called the safety zone constant is prefixed and in the region $[x_k - 1, x_k + 1]$ (where x_k is the active grid point), $2M$ grid points are uniformly added. Note that precautions are taken when the active grid point is a boundary point, in that case the region $[x_k, x_k + 1]$ (in this case x_k is the left boundary point) or $[x_k - 1, x_k]$ (in this case x_k is the right boundary point) are considered as safety zones.

Having obtained the X^{new}, in the above manner, the refinement of the grid is continued treating X^{new} as $X^{initial}$ (the set $f(xi)_{xi} \in X^{new}$ is obtained by interpolation). The process of refinement is stopped when the function $f(x)$ (which in our case is the solution of the differential equation) becomes grid independent (By grid independent solution, we mean a solution that does not vary significantly even when the grid is refined). We call the final obtained grid as X^{final}. Note that if the parameter M is chosen very small, then the number of refinements required to obtain X^{final} from $X^{initial}$ and hence the compu-

tational cost increases. If, on the other hand, M is selected large, there are chances that the unnecessary points are added to our grid. Therefore, this parameter M should be chosen wisely depending on the problem. Fig. (3.7) shows the variation of the compression error $\|f - f_{\geq\epsilon}\|_2$ versus ϵ for three test functions. It is observed that error increases on increasing the value of ϵ as expected. For $\epsilon = 10^{-4}$, Fig. (3.8) represents the adaptive grid constructed by using the modified adaptation strategy for various functions. Algorithm for second generation wavelet based adaptive grid generation is as follows:

- Compute $\{f(x_j)\}_{j\in X^J}$ from $\{f(x_j)\}_{j\in X^c}$ using interpolation.
- Apply second generation wavelet transform on the expanded $f(x)$ for obtaining the wavelet and scaling coefficients.
- Start with X^J.
- Keep intact all the points that corresponds to the scaling functions.
- Keep the points where $|d_k^j| \geq \epsilon$ intact.
- Keep the points corresponding to the adjacent zone of each active point intact.
- Delete all the other points.

3.4.3 Numerical algorithm for solving PDEs

Having obtained all the necessary ingridients for SGWOFD, we explain the algorithm for SGWOFD:

1. Discretize the domain of the PDE.
2. Use finite difference matrices given in section (3.4.1) to discretize the operators involved in the PDE at hand.
3. Suppose the times $t_1, t_2, \cdots$ are decided for the grid refinements (Note that the choice of these times is problem dependent. If the problem is such that its solution is rapidly changing, then t_is should be chosen closer). Starting with the initial condition, with the use of a time integration scheme (Crank-Nicolson in our case), we obtain a solution at the time t_1, i.e., $u(t_1)$.
4. We use $u(t_1)$ and the corresponding grid X^{t_1} to obtain $X^{t_1+\Delta t}$ using the process explained in the last section.
5. Calculate the PDE differential operators on $X^{t_1+\Delta t}$.

6. Integrate with Crank-Nicolson the ordinary differential equations system obtained, in time to attain the solution at $t = t_1 + \Delta t$.

7. Repeat the step 6 until $u(t_2)$ is obtained. Having obtained $u(t_2)$ go to step 4.

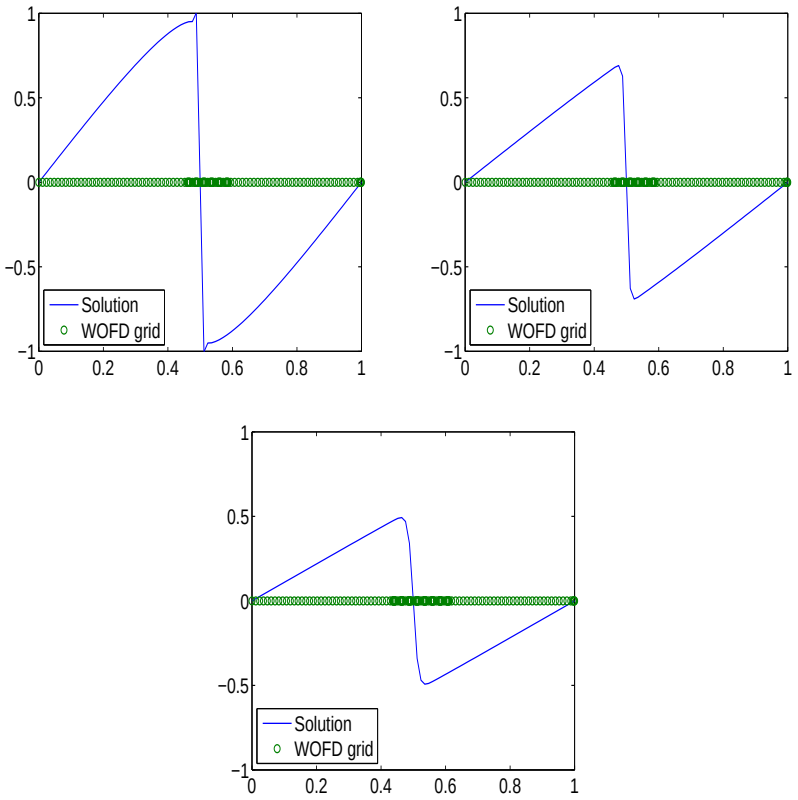


Figure 3.9: Solution and the corresponding adaptive grid at $t = 0.25$, 0.5, 0.75.

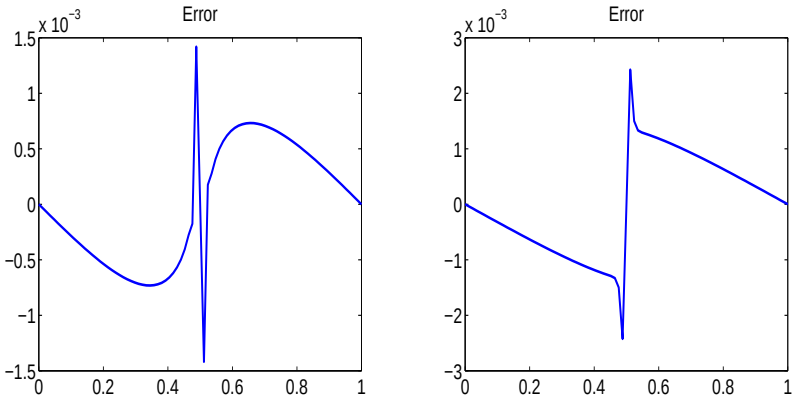
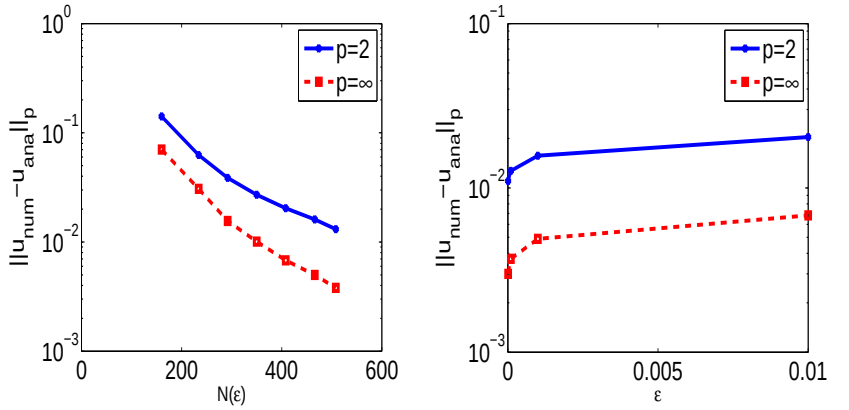
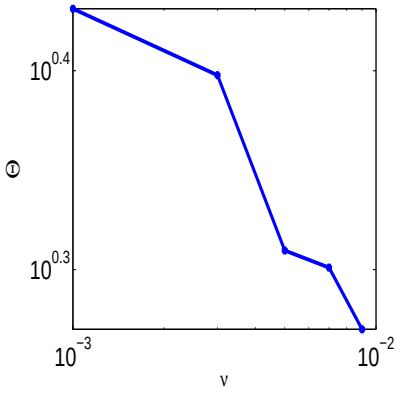


Figure 3.10: Pointwise error at $t = 0.25$ and $t = 0.5$.



(a) $\|u_{num} - u_{ana}\|_2$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_2$ vs. ϵ



(c)

Figure 3.11: Results for the test problem I.

3.5 Numerical results and discussions

To test the proposed SGWOFD method, Burger's equation has been considered with different boundary cases. The equation which we consider is given as follows:

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = \nu \frac{\partial^2 u(x,t)}{\partial x^2}, \tag{3.5.1}$$

with the initial profile given by

$$u(x,0) = g(x).$$

and with appropriate boundaries. The equation is solved by using SGWOFD as described in section (3.4). Depending upon the different types of boundary conditions used, following four problems are considered.

Test problem I

In the 1st test problem, the Burger's equation on the interval $[0, 1]$ has been considered with the initial $u(x, 0) = u_0(x) = \sin(2\pi x)$ and boundaries are periodic, *i.e.*, $u(x, t) = u(x+1, t)$. The problem's exact solution is [171]:

$$u(x,t) = \frac{\int_{-\infty}^{\infty} \frac{x-\xi}{t} \exp(\frac{-(x-\xi)^2}{4\nu t}) \exp(\frac{\cos(2\pi\xi)}{4\pi\nu}) d\xi}{\int_{-\infty}^{\infty} \exp(\frac{-(x-\xi)^2}{4\nu t}) \exp(\frac{cos(2\pi\xi)}{4\pi\nu}) d\xi}.$$

This problem's analytical solution is a static wave which shows a sharp gradient at $x = 0.5$. For $\epsilon = 10^{-4}$ and $\nu = 0.005$, Fig. (3.9) represents its solution and related adapted grid at distinct time. Fig. (3.10) displays the point-wise error at distinct times and it can be seen that where there are variations in the solution, the error value is maximum and therefore more grid points must be included at this region. Fig. (3.11)(a), (3.11)(b) justifies the method's convergence in regards to $N(\epsilon)$ and ϵ respectively. The graphs show that the convergence order which we look for has been achieved. We have found that the error decreases as the ϵ value decreases, however the computational cost increases. ϵ is therefore to be sensibly selected that balances the error with the computational cost. The time compression coefficient is defined as $\Theta = \frac{CPU(\epsilon=0)}{CPU(\epsilon)}$. Fig. (3.11)(c) displays the change in value Θ with respect to ν. It is found that as ν value decreases, the Θ value increases, that means the approach achieves better results for small ν values. Table (3.4) shows the change in the CPU(ϵ) versus ϵ for value of $\nu = 0.002$. It is checked that as ϵ decreases, Θ decreases. The higher the Θ value, more efficient the adaptive algorithm is.

Table 3.4: The efficiency of SGWOFD for the test problem I.

ϵ	10^{-2}	10^{-3}	10^{-4}	10^{-5}
CPU(ϵ)	0.0625	0.0781	0.1250	0.1275
Θ	3.2496	2.6005	1.6248	1.592

Table 3.5: The efficiency of SGWOFD for the test problem II.

ϵ	10^{-2}	10^{-3}	10^{-4}	10^{-5}
CPU(ϵ)	0.0469	0.0625	0.0729	0.0785
Θ	4.997	3.7504	3.2153	2.985

Table 3.6: Solution of test problem II by explicit finite difference method, exact-explicit finite difference method and SGWOFD method.

x	explicit finite difference	exact-explicit finite difference	SGWOFD	exact solution
0.1	0.10863	0.11048	0.1100	0.1095
0.2	0.20805	0.21159	0.2106	0.2098
0.3	0.28946	0.29435	0.2930	0.2919
0.4	0.34501	0.35080	0.3490	0.3479
0.5	0.36845	0.37458	0.3725	0.3716
0.6	0.35601	0.36189	0.3594	0.3590
0.7	0.30728	0.31231	0.3095	0.3099
0.8	0.22588	0.22955	0.2264	0.2278
0.9	0.11966	0.12160	0.1185	0.1207
	0.0070	0.0067	0.0034	

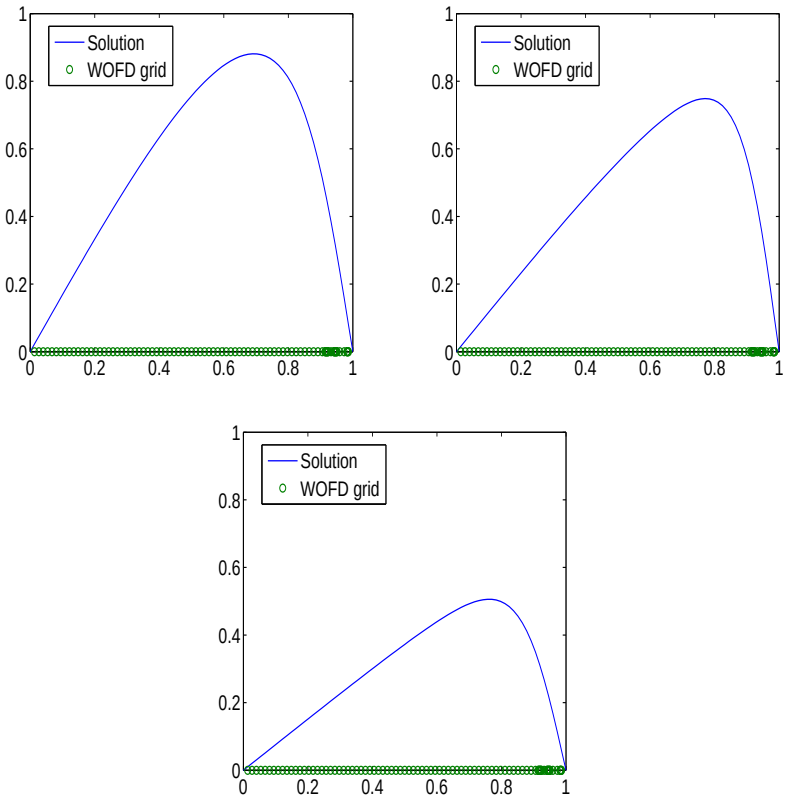


Figure 3.12: Solution and the corresponding adaptive grid at $t = 0.25, 0.5, 0.95$.

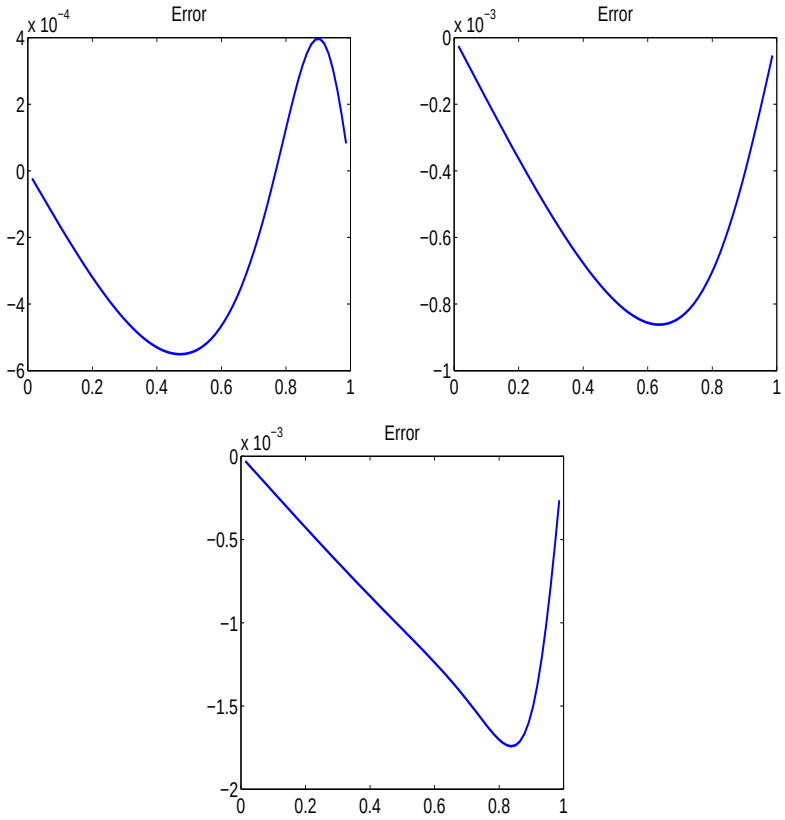


Figure 3.13: Pointwise error at $t = 0.25$, $t = 0.5$ and $t = 0.95$.

Test problem II

In the 2nd test problem, Burger's equation (3.5.1) has been considered on an interval $[0, 1]$ with an initial given as $u(x, 0) = u_0(x) = \sin(\pi x)$ and Dirichlet boundaries, *i.e.*, $u(1, t) = u(0, t) = 0,\ t > 0$.
The problem's exact solution is [172]:

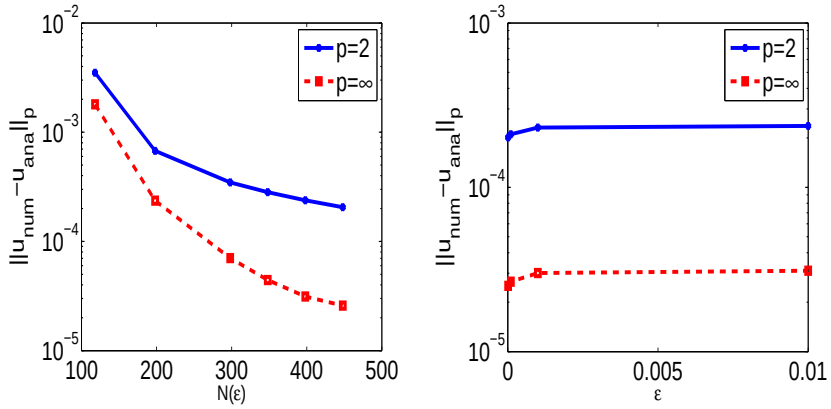
$$u(x,t) = 2\pi\nu \frac{\sum_{n=1}^{\infty} a_n \exp(-n^2\pi^2\nu t) n \sin(n\pi x)}{a_0 + \sum_{n=1}^{\infty} a_n \exp(-n^2\pi^2\nu t)\cos(n\pi x)}.$$

For $\nu = 0.05$ and $\epsilon = 10^{-4}$, Fig. (3.12) displays its solution and its related adaptive grid at distinct times. Fig. (3.13) displays the point-wise error at distinct times. Fig. (3.14)(a), (3.14)(b) checks the method's convergence in regards to $N(\epsilon)$ and ϵ. It can be seen from these graphs that the convergence order which we looked for has been obtained. We have observed that error between numerical and analytical value of the problem decreases as $N(\epsilon)$ increases and decreases on decreasing the value of ϵ. Table (3.5) represents the variation of Θ for different values of ϵ. It is found that Θ increases on increasing the ϵ which shows the efficiency of our adaptive algorithm. Table (3.6) displays explicit, exact-explicit

finite difference method and SGWOFD solutions for $\nu = 1$, $\delta t = 0.00001$, final time=0.1 and $N = 10$. We have noticed that numerical simulations are well in agreement with analytical solution. The numbers in the bottom row of these tables are the error norms $\|u_e - u_{ana}\|$, $\|u_{e_1} - u_{ana}\|$ and $\|u_{num} - u_{ana}\|$ (where u_e stands for numerical solution obtained by explicit finite difference approach, u_{e_1} stands for numerical solution obtained by exact-explicit finite difference approach, u_{num} represents the numerical solution calculated by using SGWOFD and u_{ana} represents the analytical solution of the problem). In all of the three cases, we have observed that SGWOFD method is more accurate.

Table 3.7: The efficiency of SGWOFD for the test problem III.

ϵ	10^{-2}	10^{-3}	10^{-4}	10^{-5}
CPU(ϵ)	5.3438	5.3750	6.2412	8.6719
Θ	3.3186	3.2994	2.8415	2.045



(a) $\|u_{num} - u_{ana}\|_p$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_p$ vs. ϵ

Figure 3.14: Results for the test problem II.

Test problem III

For 3rd test problem, non-homogeneous Burger's equation has been considered

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = \nu \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad x \in [0,1], \tag{3.5.2}$$

where

$$f(x,t) = -\frac{1}{4}e^{-\nu t}\cos(\pi x)(\nu + \frac{\pi}{4}e^{-\nu t}\sin(\pi x) - \nu\pi^2),$$

Table 3.8: Solution of test problem III by Galerkin method, Galerkin/Conservative and SGWOFD method (Re=60).

x	Galerkin	Galerkin/Conservative	SGWOFD
0	0.2505	0.2504	0.2460
0.0588	0.2457	0.2456	0.2414
0.1176	0.2326	0.2325	0.2297
0.1765	0.2118	0.2116	0.2100
0.2353	0.1839	0.1836	0.1828
0.2941	0.1499	0.1496	0.1491
0.3529	0.1108	0.1106	0.1103
0.4118	0.068	0.0679	0.0677
0.4706	0.0229	0.0229	0.0228
0.5294	−0.0229	−0.0229	−0.228
0.5882	−0.068	−0.0679	−0.0677
0.6471	−0.1108	−0.1106	−0.1103
0.7059	−0.1499	−0.1496	−0.1491
0.7647	−0.1839	−0.1836	−0.1828
0.8235	−0.2118	−0.2116	−0.2100
0.8824	−0.2326	−0.2325	−0.2297
0.9412	−0.2457	−0.2456	−0.2414
1	−0.2505	−0.2504	−0.2460
	0.0053	0.0049	0.0045

with an initial $u(x,0) = \frac{1}{4}\cos(\pi x)$ and Neumann boundaries $u_x(0,t) = u_x(1,t) = 0$. The problem's analytical solution is

$$u(x,t) = \frac{1}{4}\exp^{-\nu t}\cos(\pi x).$$

For $\nu = 1$ and $\epsilon = 10^{-4}$, Fig. (3.15) displays the solution of the problem and the associated grid of adaptation at distinct times. Fig. (3.16)(a) displays the error ($i.e.$, $\|u_{num} - u_{ana}\|_2$) at time $t = 0.5$ versus $N(\epsilon)$. It can be observed from this graph that error decreases as $N(\epsilon)$ increases. Fig. (3.16)(b) shows the graph between $\|u_{num} - u_{ana}\|_2$ and ϵ. It shows that error increases on increasing the ϵ. Fig. (3.16)(c) shows the comparison between numerical and analytical value of the problem. Table (3.7) gives the variation of CPU(ϵ) with ϵ. We have observed that as ϵ decreases, Θ decreases. The higher the Θ value, more effective the adaptive algorithm is. Table (3.8), Table (3.9) and Table (3.10) compare the

Table 3.9: Solution of test problem III by Galerkin method, Galerkin/Conservative and SGWOFD method (Re=120).

x	Galerkin	Galerkin/Conservative	SGWOFD
0	0.2529	0.2528	0.2471
0.0588	0.2474	0.2474	0.2417
0.1176	0.2338	0.2337	0.2305
0.1765	0.2127	0.2125	0.2110
0.2353	0.1846	0.1843	0.1836
0.2941	0.1504	0.1502	0.1498
0.3529	0.1113	0.111	0.1108
0.4118	0.0683	0.0681	0.0680
0.4706	0.023	0.023	0.0229
0.5294	−0.023	−0.023	−0.0229
0.5882	−0.0683	−0.0681	−0.0680
0.6471	−0.1113	−0.111	−0.1108
0.7059	−0.1504	−0.1502	−0.1498
0.7647	−0.1846	−0.1843	−0.1836
0.8235	−0.2127	−0.2125	−0.2110
0.8824	−0.2338	−0.2337	−0.2305
0.9412	−0.2474	−0.2474	−0.2417
1	−0.2529	−0.2528	−0.2471
	0.0073	0.0071	0.0049

numerical solution of the problem computed by Galerkin, Galerkin/Conservative [173] and SGWOFD method at final time $t = 1/2$. The finite dimensional models were set up using $N = 16$ elements and the equations were solved with Re=60, 120 and 240 (Re stands for Reynold's number and is equal to inverse of ν). The numbers in the bottom row of these tables are the error norms $\|u_G - u_{ana}\|$, $\|u_{G/C} - u_{ana}\|$ and $\|u_{num} - u_{ana}\|$ (where u_G stands for numerical solution computed by Galerkin method and $u_{G/C}$ stands for numerical solution computed by Galerkin/Conservative method). As Re increases these error norms become larger, indicating reduction in accuracy. This is to be expected because $\nu = \frac{1}{Re} \to 0$ corresponds to the decreasing viscosity. In all of three cases we have observed that SGWOFD method is more accurate.

Table 3.10: Solution of test problem III by Galerkin method, Galerkin/Conservative and SGWOFD method (Re=240).

x	Galerkin	Galerkin/Conservative	SGWOFD
0	0.2556	0.2556	0.2480
0.0588	0.249	0.2489	0.2414
0.1176	0.2345	0.2343	0.2310
0.1765	0.213	0.2128	0.2116
0.2353	0.1849	0.1846	0.1841
0.2941	0.1508	0.1504	0.1501
0.3529	0.1115	0.1112	0.1110
0.4118	0.0685	0.0683	0.0681
0.4706	0.0231	0.023	0.0230
0.5294	−0.0231	−0.023	−0.0230
0.5882	−0.0685	−0.0683	−0.0681
0.6471	−0.1115	−0.1112	−0.1110
0.7059	−0.1508	−0.1504	−0.1501
0.7647	−0.1849	−0.1846	−0.1841
0.8235	−0.213	−0.2128	−0.2116
0.8824	−0.2343	−0.2344	−0.2310
0.9412	−0.249	−0.2489	−0.2414
1	−0.2556	−0.2556	−0.2480
	0.0107	0.0105	0.005

Table 3.11: The efficiency of SGWOFD for the test problem IV.

ϵ	10^{-2}	10^{-3}	10^{-4}	10^{-5}
CPU(ϵ)	0.6719	0.8125	0.8281	0.8750
Θ	2.11	1.746	1.713	1.621

Test problem IV

Consider non-homogeneous Burger's equation

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = \nu \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad x \in [0,100], \tag{3.5.3}$$

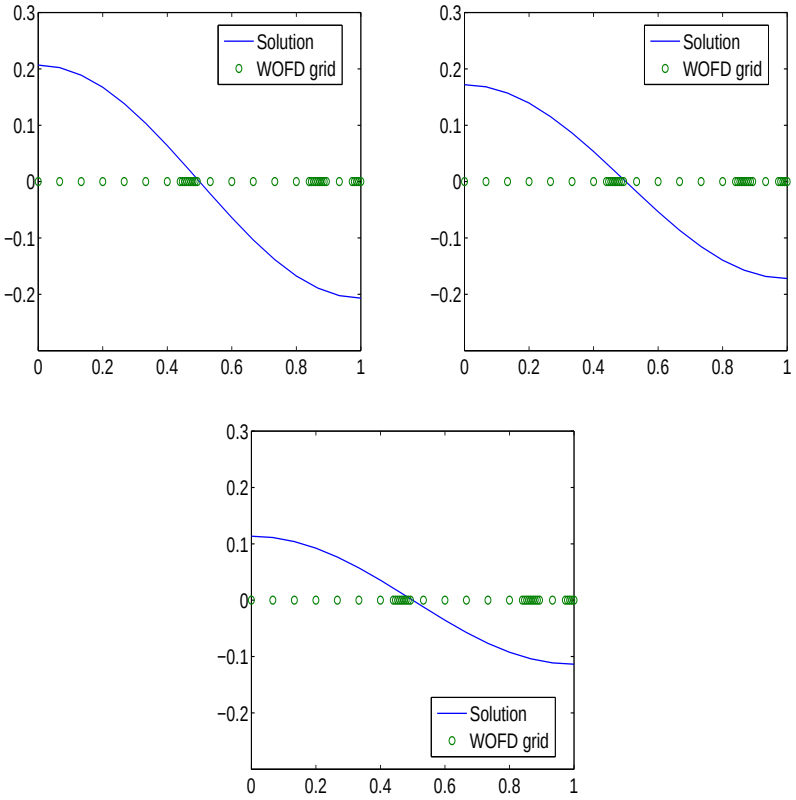


Figure 3.15: Solution and the corresponding adaptive grid at $t = 0.25,\ 0.5,\ 0.95$.

with the initial $u(x,0) = \dfrac{x^2}{10^4}$ and non-homogeneous Robin's boundaries

$$\frac{\kappa_1}{L_1}u(0,t) - \kappa\frac{\partial u(0,t)}{\partial x} = \frac{\kappa_1}{L_1}u_1(t), \tag{3.5.4}$$

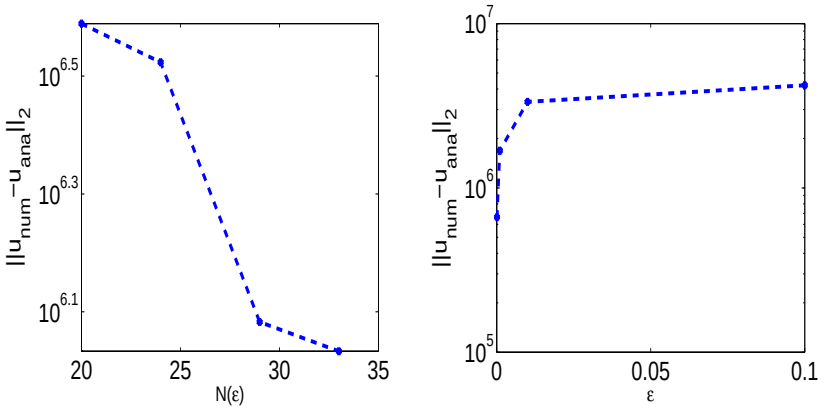
$$\frac{\kappa_2}{L_2}u(L,t) + \kappa\frac{\partial u(L,t)}{\partial x} = \frac{\kappa_2}{L_2}u_2(t), \tag{3.5.5}$$

where

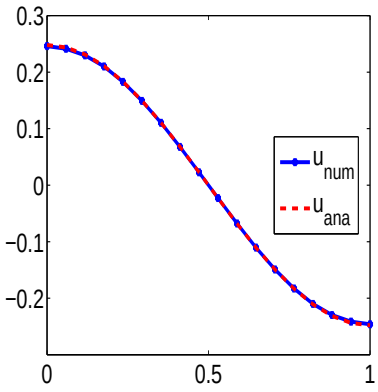
$$\begin{aligned} f(x,t) = {} & \alpha\frac{(100-x)^2}{10^4}\cos(\alpha t) - \beta\frac{x^2}{10^4}\sin(\beta t) - 2\frac{(100-x)^3}{10^8}\sin^2(\alpha t) + 2\frac{x^3}{10^8}\cos^2(\beta t) \\ & + \frac{2}{10^8}(2x^3 - 300x^2 + 10^4 x)\sin(\alpha t)\cos(\beta t) - \frac{2\nu}{10^4}(\sin(\alpha t) + \cos(\beta t)), \end{aligned}$$

$$u_1(t) = (1 + \frac{2\kappa L_1}{100\kappa_1})\sin(\alpha t), \tag{3.5.6}$$

$$u_2(t) = (1 + \frac{2\kappa L_2}{100\kappa_2})\cos(\beta t), \tag{3.5.7}$$



(a) $\|u_{num} - u_{ana}\|_2$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_2$ vs. ϵ



(c)

Figure 3.16: Results for the test problem III.

with $L = 100, L_1 = L_2 = 10, \kappa = 0.93, \kappa_1 = \kappa_2 = 0.55, \nu = 1.14, \alpha = 1.5\pi, \beta = 0.5\pi$. The problem's exact solution is [174]

$$u(x,t) = \frac{(100-x)^2}{10^4}\sin(\alpha t) + \frac{x^2}{10^4}\cos(\beta t).$$

For $\nu = 1.14$ and $\epsilon = 10^{-4}$, Fig. (3.17) displays the solution of the problem and the associated grid of adaptation at distinct times. Fig. (3.18)(a) displays the error (*i.e.*, $\|u_{num} - u_{ana}\|_\infty$) at time $t = 0.5$ versus $N(\epsilon)$. It is found that error decreases as $N(\epsilon)$ increases. Fig. (3.18)(b) displays the relation between $\|u_{num} - u_{ana}\|_\infty$ and ϵ. It shows that error increases on increasing the ϵ. Fig. (3.18)(c) displays the comparison between numerical and analytical value of the problem. Fig. (3.18)(d) compares the computational time carried out by the different numerical schemes *i.e.*, Galerkin, Galerkin/Conservative and SGWOFD for solving test problem 4. It is found that CPU time taken by SGWOFD is less as compared to that of the finite difference scheme. Table (3.11) gives the variation of CPU(ϵ) with ϵ. We have observed that as ϵ decreases, Θ decreases. The higher the Θ

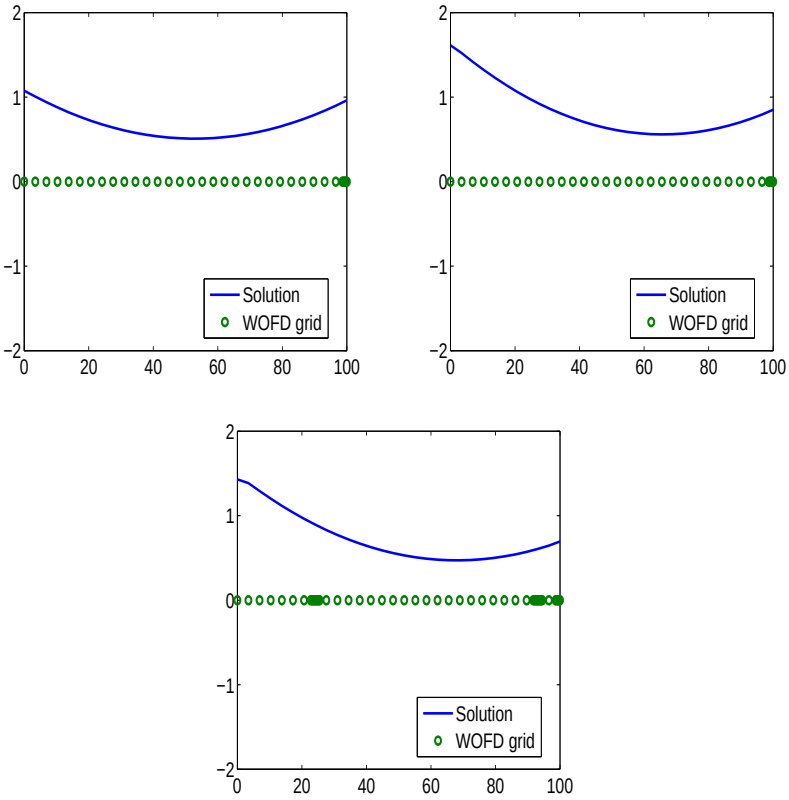
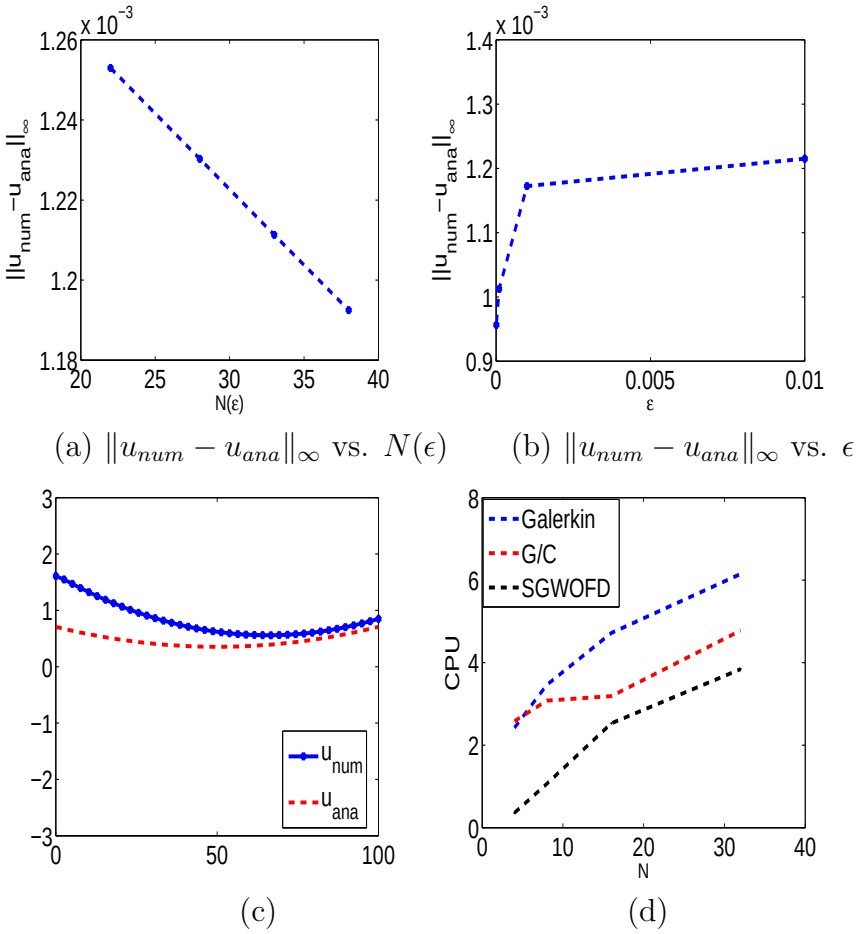


Figure 3.17: Solution and the corresponding adaptive grid at $t = 0.25, 0.5, 0.725$.

value, more effective the adaptive algorithm is.

3.6 Conclusion and Future directions

This chapter presents the formation of second-generation wavelet relying on the lifting concept and the corresponding matlab toolbox. The second-generation wavelet so constructed is utilized for generating the adaptive grid. Furthermore, second generation wavelet based wavelet optimized finite difference method (SGWOFD) has been developed. Central finite difference matrices are used for approximations of differential operators involved in a given PDE. Finally time integration Crank-Nicolson technique has been used. The SGWOFD is tested on Burger's equation with Dirichlet, periodic, Robin and Neumann's boundaries. The convergence and efficiency of SGWOFD are checked for all the four problems considered. Comparison of CPU time reveals that the developed technique is highly efficient. Having checked the efficiency of the developed technique for Burger's equation, we propose the use of SGWOFD for turbulence modeling and for solving problems of fluid dynamics. The development of Matlab suite for second generation wavelets on weighted $\mathcal{L}_p$ spaces is



(a) $\|u_{num} - u_{ana}\|_\infty$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_\infty$ vs. ϵ

(c) (d)

Figure 3.18: Results for the test problem IV.

also one of the future direction.

Chapter 4

Third generation wavelet based numerical method for solving PDEs

Another types of wavelets which can be utilized to solve PDEs with all kinds of boundary conditions are recently developed wavelets such as diffusion wavelet [175] and spectral graph wavelet (SPGW) [176]. We term these wavelets as third generation wavelets. These wavelets are constructed directly on the domain where one has to solve his/her problems, therefore their construction becomes boundary conditions independent. There is some sort of similarity in diffusion and spectral graph wavelet, for example, both require a diffusion operator for their construction. The largest difference between the two is that the diffusion-wavelet is constructed to be orthogonal, whereas the spectral graph wavelet is non-orthogonal. The orthogonalization in the case of diffusion wavelet is attained by applying a confined Gram-Schmidt at every scale after employing dyadic powers of T for yielding the next estimation spaces. Then wavelets are constructed by orthogonalising the vectors that span the change in the estimation spaces. This orthogonalising technique complicates the construction of the diffusion-wavelet and scaling-function transforms. On the other hand, the approach used for the SPGW is much simpler [176]. For details characterized on the vertex of the weighted-graph, SPGW is constructed to define wavelet transformation. The construction technique utilizes just the network data encrypted in the edge weight and does not depend on some other characteristics of the vertices. Accordingly, the transformation can be characterized on any manifold where the fundamental relations between information areas can be spoken to by a weighted diagram. Since the weighted diagrams give an incredibly adaptable model to approximate the information spaces of a vast category of problems, the spectral graph wavelet has potential for applications in many fields.

Weighted graphs generalize consistent grid domains flexibly. Here, we can identify the points of nodes with vertices and connect neighboring points of nodes with corners having weights inversely proportional to the neighborhood distance square. This particular weighted graph model motivated us to try spectral graph wavelet [176] for numerically solving the PDEs.

The diffusion-wavelet was used to solve PDEs in the sense of wavelet optimized finite difference (WOFD) method in the research papers [53, 97, 177]. This chapter proposes an adaptive spectral-graph wavelet optimized finite difference technique (SPGWOFD) for solving the Burger's equation with different boundary conditions. The method uses SPGW for adapting the grid. The finite difference approach is utilized to approximate the derivatives engaged in the Burger's equation. Four test problems (with Dirichlet, Periodic, Robin and Neumann's boundary conditions) are considered and the convergence of the technique is checked. The numerical outcomes show that the wavelet and computational grid can adapt very efficiently to the solution's local irregularities to resolve sharp transition areas. For assessing the efficiency of the developed technique, the computational time carried out by the developed technique is compared to that of finite difference method. It has been observed that developed technique is extremely efficient.

4.1 A short explanation of SPGW

Here, we provide a concise explanation of the SPGW developed in [176]. One can refer to [54] for more details. SPGW is developed on a random graph with finite weight. A graph $G = \{V, w, E\}$ which is weighted comprises of a collection of edges E, collection of vertices V, a weight function $w : E \to \mathbb{R}^+$ that assigns to every edge a non-negative weight. Assume that the no. of vertices in the graph is finite ($i.e.$, $< \infty$). An adjacency matrix $A = \{a_{m,n}\}$ is the $N \times N$ matrix for a G graph, where the graph is weighted

$$a_{m,n} = \begin{cases} w(e) & \text{if } e \in E \text{ connects vertices } m \text{ and } n \\ 0 & \text{otherwise.} \end{cases}$$

The graph is supposed to be undirected (which means that the matrix A is a symmetrical matrix). It ought to be noticed that the graphs on which we work on are connected and finite when dealing with PDEs.

The degree of every vertex m for a weighted graph, composed as $d(m)$, is characterized as the weighted sum of all the edges that occur there, $i.e.$, $d(m) = \sum_n a_{m,n}$. All the diagonal entries of a diagonal matrix $\mathcal{D}$ are characterized as $d(m)$. For the graph a non normalized Laplacian is characterized as $\mathcal{L} = \mathcal{D} - \mathcal{A}$. The Laplacian graph is related to the continuous Laplacian-Beltrami operator's standard stencil estimation with a sign change for a graph resulting from a regular mesh. For example, the adjacenct matrix $\mathcal{A}$ and the matrix $\mathcal{D}$ are

characterized as

$$\mathcal{A} = \begin{bmatrix} 0 & \frac{1}{\delta x^2} & 0 & 0 & 0 \\ \frac{1}{\delta x^2} & 0 & \frac{1}{\delta x^2} & 0 & 0 \\ 0 & \frac{1}{\delta x^2} & 0 & \frac{1}{\delta x^2} & 0 \\ 0 & 0 & \frac{1}{\delta x^2} & 0 & \frac{1}{\delta x^2} \\ 0 & 0 & 0 & \frac{1}{\delta x^2} & 0 \end{bmatrix} \quad , \quad \mathcal{D} = \begin{bmatrix} \frac{1}{\delta x^2} & 0 & 0 & 0 & 0 \\ 0 & \frac{2}{\delta x^2} & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{\delta x^2} & 0 & 0 \\ 0 & 0 & 0 & \frac{2}{\delta x^2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\delta x^2} \end{bmatrix}$$

Hence

$$\mathcal{L} = \frac{1}{\delta x^2} \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

which shows the approximation of central finite difference approach of $-\nabla^2$ with neumann boundaries. The graph Fourier transformation (GFT) $\hat{f}$ of the function $f \in \mathbb{R}^N$ characterized on the vertex of the graph G, is defined as

$$\hat{f}(l) = \langle \chi_l, f \rangle = \sum_{n=1}^{N} \chi_l^*(n) f(n),$$

here $\{\chi_l, \quad l = 0, 1, \cdots, N-1\}$ are eigen-vectors compared to its eigen-values $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \cdots \leq \lambda_{N-1}$ of the $\mathcal{L}$th matrix. It is noticed that, if $\mathcal{L}$th graph Laplacian matrix is real symmetrical then $\mathcal{L}$ has a full set of ortho-normal eigen-vectors. The inverse of GFT is

$$f(n) = \sum_{l=0}^{N-1} \hat{f}(l) \chi_l(n).$$

4.1.1 Spectral graph wavelet transformation (SPGWT)

The kernel-function $g : \mathbb{R}^+ \to \mathbb{R}^+$ is selected to satisfy $\lim_{x \to \infty} g(x) = 0$ and $g(0) = 0$ (g is referred to as the kernel of the SPGW). The wavelet's operator $T_g = g(\mathcal{L})$ follows up on the function f by regulating every Fourier condition for a kernel function g as

$$\widehat{T_g f}(l) = g(\lambda_l) \hat{f}(l),$$

This implies

$$(T_g f)(m) = \sum_{l=0}^{N-1} g(\lambda_l) \hat{f}(l) \chi_l(m).$$

At scale t, the wavelet operator is then characterized as $T_g^t = g(t\mathcal{L})$. Note that although the graph's spatial domain is discrete, the wavelet kernel g has continuous domain and therefore the scaling for the non-negative real number t can be defined. The SPGW is described as

$$\psi_n^t = T_g^t \delta_n,$$

which implies

$$\psi_n^t(m) = T_g^t \delta_n(m) = \sum_{l=0}^{N-1} g(t\lambda_l)\hat{\delta}_n(l)\chi_l(m) = \sum_{l=0}^{N-1} g(t\lambda_l)\chi_l^*(n)\chi_l(m).$$

where δ_n stands for the Kronecker delta that returns the one value at the n^{th} vertices and zero else. The coefficients of wavelet for the function f is obtained by using these wavelets as the inner-product of the function f, as

$$W_f(t,n) = \langle \psi_n^t, f \rangle \text{ (SPGWT)}.$$

The coefficients of wavelet could be directly obtained from the operators of wavelet by utilizing the orthonormality of the $\{\chi_l\}$,

$$W_f(t,n) = (T_g^t)(n) = \sum_{l=0}^{N-1} g(t\lambda_l)\hat{f}(l)\chi_l(n). \tag{4.1.1}$$

4.1.2 Spectral graph scaling function transformation (SPGST)

At a beginning, a single real-valued function defines the spectral-graph scaling function $h : \mathbb{R}^+ \to \mathbb{R}$ that satisfies $\lim_{x\to\infty} h(x) = 0$ and $h(0) > 0$ (h is the scaling function kernel). The spectral-graph scaling function is determined as

$$\phi_n = T_h \delta_n = h(\mathcal{L})\delta_n,$$

and the coefficients of scaling function are determined as

$$S_f(n) = \langle \phi_n, f \rangle \text{ (SPGST)}. \tag{4.1.2}$$

It should be noted that the scaling functions are thus smoothly defined to represent the low-frequency content of a function. The wavelets are not generated by the 2-scale relationship as with orthogonal traditional wavelets.

4.1.3 Fast SPGWT and SPGST

A natural procedure of calculating SPGWT and SPGST, by precisely utilizing Eqn. (4.1.1) and Eqn. (4.1.2) respectively, demands the explicit calculation of the whole set of Laplacian operator's eigen values and eigen functions. For computing large graphs, this technique is ineffective. To make SPGWT and SPGST a helpful device for feasible computational problems, a rapid transformation that averts the demand to compute the entire spectrum of L is needed. To acquire this, low-order polynomials approximate the scaling function h and wavelet kernel g.

The h and g kernels are estimated in Chebyshev by their polynomial extensions. A steady recurrence relationship could construct the Chebyshev polynomial $T_k(x)$ as $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$, with $T_1(x) = 1$, $T_0(x) = x$. They fulfill the trigonometrical interpretation $T_k(x) = \cos(k\cos^{-1}(x))$ for $x \in [-1, 1]$. An orthogonal basis is formed for the Chebyshev polynomial for $L^2([-1, 1], \frac{dx}{\sqrt{1-x^2}})$. Each $f \in L^2([-1, 1], \frac{dx}{\sqrt{1-x^2}})$ has a converging sequence of Chebyshev

$$f(x) = \frac{1}{2}e_0 + \sum_{k=1}^{\infty} e_k T_k(x),$$

having the Chebyshev's coefficients are

$$e_k = \frac{2}{\pi}\int_{-1}^{1} \frac{T_k(x)f(x)}{\sqrt{1-x^2}}dx = \frac{2}{\pi}\int_{0}^{\pi} \cos(k\theta)f(\cos\theta)d\theta.$$

Approximation of $g(t_j x)$ for the fixed wavelet level t_j and for $x \in [0, \lambda_{max}]$ (where λ_{max} is the operator's largest eigenvector $\mathcal{L}$) could be obtained by altering the domain with $y = a(x + 1)$ where $a = \frac{\lambda_{max}}{2}$. The shifted polynomials of Chebyshev are denoted by $\overline{T}_k(x) = T_k\left(\frac{x-a}{a}\right)$, therefore it can be written as

$$g(t_j x) = \frac{1}{2}e_0^j + \sum_{k=1}^{\infty} e_k^j \overline{T}_k(x), \tag{4.1.3}$$

which holds for $x \in [0, \lambda_{max}]$, having

$$e_k^j = \frac{2}{\pi}\int_{0}^{\pi} \cos(k\theta)g(t_j(a\cos\theta + 1))d\theta.$$

For every t_j level, the estimating polynomial p_j can be obtained by restricting the Chebyshev's extension to M_j terms which is given by the Eqn. (4.1.3). For approximating the scaling function kernel by the polynomial p_0, the same strategy can be employed.

The selection of M_j values can be considered a design problem, which pose a balance

between computational cost and exactness. The approximate values of the coefficients of the wavelet and scaling function are determined as

$$\widetilde{W}_f(t_j, n) = \frac{1}{2} e_0^j f(n) + \sum_{k=1}^{M_j} e_k^j \overline{T}_k(\mathcal{L}) f(n),$$

$$\widetilde{S}_f(n) = \frac{1}{2} e_0^0 f(n) + \sum_{k=1}^{M_j} e_k^0 \overline{T}_k(\mathcal{L}) f(n).$$

4.1.4 SPGW frames

The SPGW relies on a continuous scale variable t. T should be modeled as a limited no. of scales for any constructive calculation. Selecting scales J $\{t_j\}_{j=1}^J$ results a collection of JN wavelets $\psi_n^{t_j} = \psi_n^j$, in addition to scaling function N. The size of this set of vectors is given by the following theorem to represent the functions in a graph.

Theorem: Assume that a collection of scales $\{t_j\}_{j=1}^J$ are given, then $\{\phi_n\}_{n=1}^N \cup \{\psi_n^j\}_{j=1,n=1}^{J\quad N}$ sets represents a frame having bounds A, B are determined as

$$A = \min_{\lambda \in [0, \lambda_{N-1}]} G(\lambda),$$

$$B = \max_{\lambda \in [0, \lambda_{N-1}]} G(\lambda),$$

here $G(\lambda) = h^2(\lambda) + \sum_j (g(t_j \lambda))^2$. Hence the function f represented on a graph is given by

$$f(n) = \sum_{k=1}^{N} c_k \phi_k(n) + \sum_{j=1}^{J} \sum_{k=1}^{N} d_k^j \psi_k^j(n), \tag{4.1.4}$$

where $\{d_k^j\}_{k=1}^N$ are the coefficients of wavelet function and $\{c_k\}_{k=1}^N$ are the coefficients of scaling functions at the level j.

4.2 Grid adaptation using spectral graph wavelet

As described in the section (3.4.2), the generation of the adaptive grid depends upon the wavelet coefficients magnitude. For a function $f : V \to \mathbb{R}$ characterized a given threshold ϵ and on the vertices of the graph, we can write Eqn. (4.1.4) as $f(n) = f_{\geq \epsilon}(n) + f_{< \epsilon}$,

where

$$f_{\geq\epsilon}(n) = \sum_{k=1}^{N} c_k \phi_k(n) + \sum_{j=1}^{J} \sum_{|d_k^j| \geq \epsilon} d_k^j \psi_k^j(n) \text{ and } f_{<\epsilon}(n) = \sum_{j=1}^{J} \sum_{|d_k^j| < \epsilon} d_k^j \psi_k^j(n).$$

In [178], it is proven that for adequately smooth functions f

$$\|f - f_{\geq\epsilon}\|_\infty < C\epsilon, \tag{4.2.1}$$

here C is the constant which is independent of f. Although the above result is proved for the interpolating wavelets in [178], numerically we have verified that the same is true for SPGW (analytic proof of the same can be one of the future directions!). Fig. (4.1) displays the variation of the compression error $\|f - f_{\geq\epsilon}\|_p$, N_d (which denotes the number of the discarded wavelet coefficients) versus ϵ for three different functions. From these graphs, we conclude that the value of the slope of log-log plot of error versus ϵ is approximately 0.85. It is also shown that the value of N_d increases with increase in the ϵ value as expected.

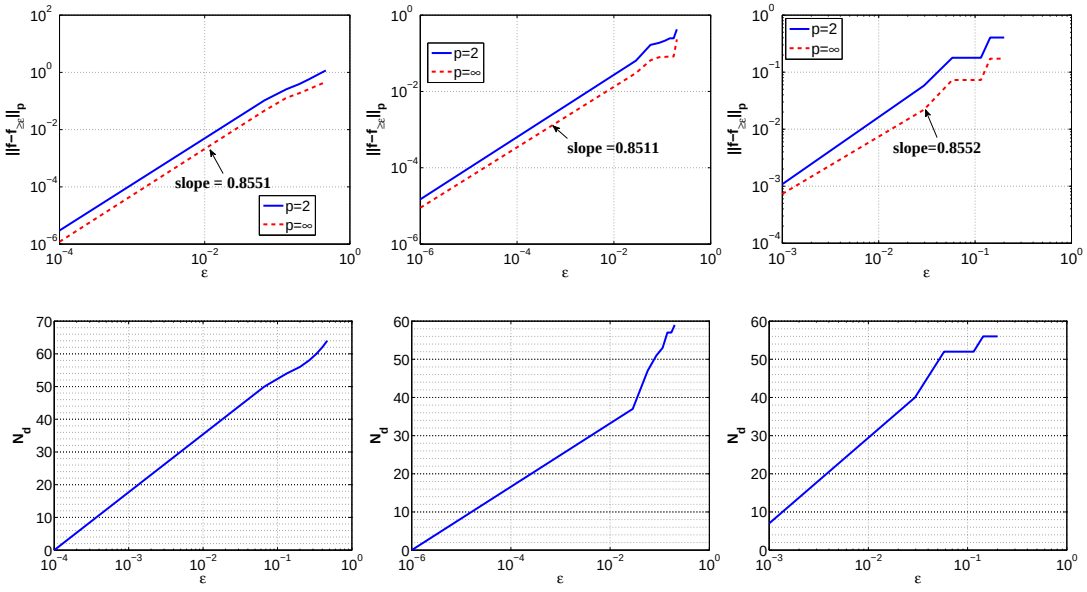


Figure 4.1: $\|f - f_{\geq\epsilon}\|_p$ versus ϵ and N_d versus ϵ for different functions for $f(x) = x$ for $0 \leq x < 0.5$ and $= x - 1$ otherwise, $0 \leq x \leq 1$, $f(x) = \tanh\left(\frac{x+x_0}{2\nu}\right) + e^{-64^2(x-x_0)^2}, -1 \leq x \leq 1, \nu = 10^{-3}, x_0 = \frac{1}{3}$, and $f(x) = x$ for $0 \leq x < 0.5$ and $= x - 1$ otherwise, $-1 \leq x \leq 1$ respectively.

From the Eqn. (4.2.1), it is evident that the grid points associated with the wavelet coefficients d_k^j with magnitude less than ϵ can be removed from the grid provided we are ready to tolerate an error of the $O(\epsilon)$.

We use the modified adaptation technique described in section (3.4.2) for adaptive node arrangement. The function $f(x)$ has been considered, which is represented on a set of points of nodes $\mathcal{G}^{old}$ for illustrating the node adapting algorithm and a pre-selected threshold ϵ parameter. Execute SPGST and SPGWT for obtaining the coefficients set $\{c_k\}_{k=1}^{|\mathcal{G}^{old}|} \cup \{d_k^j\}_{j=1,k=1}^{J \quad |\mathcal{G}^{old}|}$. We will construct $\mathcal{G}^{new}$ (fine node arrangement) from $\mathcal{G}^{old}$ (coarse node arrangement). Since each $x_k \in \mathcal{G}^{old}$ is related to a spectral scaling function ϕ_k, all the points of $\mathcal{G}^{old}$ are kept in $\mathcal{G}^{new}$. We know that at the level t_J each spectral wavelet function *i.e.*, ψ_k^J, is particularly related to an $x_k \in \mathcal{G}^{old}$. Analyse wavelet coefficients d_k^J. If $|d_k^J| \geq \epsilon$, then the corresponding node point $x_k \in \mathcal{G}^{old}$ is called the active nodal point. An adjacent are must be included in $\mathcal{G}^{new}$, for every active nodal point x_k. The adjacent area serves two goals:

1. In the region of keen fluctuations, it makes $\mathcal{G}^{new}$ finer.
2. Additional norm for adapting the nodes must be included to solve evolutionary PDEs. Specifically, the computational nodal points set must comprises of that nodal points corresponding to the coefficients of wavelets that are or may become significant over the period when the nodal points set remains unaltered [27].

For adding the safety zone, a parameter M called the safety zone constant is prefixed and in the region $[x_{k-1}, x_{k+1}]$ (where x_k is the active grid point), $2M$ grid points are uniformly added. Note that the precautions are taken when the active grid point is a boundary point, in that case the region $[x_k, x_{k+1}]$ (in case x_k is the left boundary point) or $[x_{k-1}, x_k]$ (in case x_k is the right boundary point) are considered as safety zones.

Having obtained the $\mathcal{G}^{new}$, in the above manner, the refinement of the grid is continued treating $\mathcal{G}^{new}$ as $\mathcal{G}^{old}$ (the set $\{f(x_i)\},\ x_i \in \mathcal{G}^{new}$ is obtained by interpolation). The process of refinement is stopped when the function $f(x)$ (which in our case is the solution of the differential equation) becomes grid independent (By grid independent solution, we mean a solution that does not vary considerably even if the grid is refined). We call the final obtained grid as $\mathcal{G}^{final}$. Note that if the parameter M is chosen very small, then the number of refinements required to obtain $\mathcal{G}^{final}$ from $\mathcal{G}^{old}$ becomes large and hence the computational cost increases. On the other hand, if M is chosen large, there are chances that the unnecessary points are added to our grid. Therefore, this parameter M should be chosen wisely depending on the problem at the hand. Fig. (4.2) represents the adaptive grid constructed by using the modified adaptation strategy for various functions for $M = 2$. We have discussed below the procedure of generating the adaptive grid framework based on SPGW.

$\mathcal{G}^{new}$= **AdaptiveNodeArrangement**(f, $\mathcal{G}^{old}$)

- Select a non-negative adjacent-zone parameter M and a threshold ϵ.
- $s = 0$
- $\mathcal{G}^s = \mathcal{G}^{old}$
- $f^s = f$
- **do while** $s = 0$ or $\mathcal{G}^s \neq \mathcal{G}^{s-1}$
  - Execute SPGWT on f^s for obtaining the $\{d_k^J\}_{k=1}^{|\mathcal{G}^s|}$.
  - $\mathcal{G}^{s+1} = \mathcal{G}^s$.
  - Analyze the coefficients of SPGW $\{d_k^J\}_{k=1}^{|\mathcal{G}^s|}$ and assemble the active nodal points.
  - Add an adjacent region in $\mathcal{G}^{s+1}$ to each active nodal point
  - Interpolate f^m onto new node arrangement $\mathcal{G}^{s+1}$ using interpolation call it f^{s+1}.
  - $m = m + 1$
- $\mathcal{G}^{new} = \mathcal{G}^m$

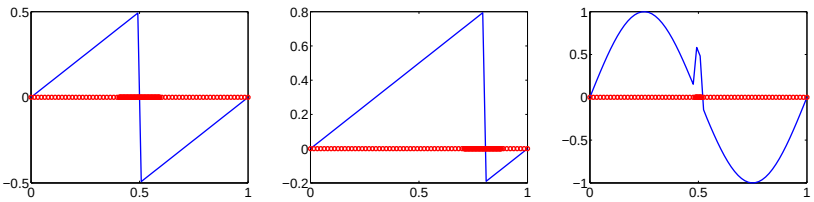


Figure 4.2: The adaptive grid for (a) Sawtooth function having discontinuity at $x = 0.5$ (b) Sawtooth function having discontinuity at $x = 0.8$ (c) $f(x) = \sin(2\pi x) + \exp(-10^4(x - 0.5)^2)$.

4.2.1 Algorithm for solving PDEs

Having obtained all the essential fundamentals for SPGWOFD, we explain the numerical algorithm for SPGWOFD as follows:

1. Having the solution $\mathbf{u}_{num}(t)$ on the nodes $\mathcal{G}^t$, formulate $\mathcal{G}^{t+\Delta t}$ using $\mathcal{G}^{t+\Delta t} = \textbf{Adaptive NodeArrangement}(\mathbf{u}_{num}(t), \mathcal{G}^t)$.
2. We will proceed directly to the next step, when the node arrangements $\mathcal{G}^t$ and $\mathcal{G}^{t+\Delta t}$ are not changed. Else the solution will be interpolated onto the fresh nodes framework $\mathcal{G}^{t+\Delta t}$.
3. Calculate the PDE's differential operator on $\mathcal{G}^{t+\Delta t}$.

4. Integrate with Crank-Nicolson the ordinary differential equations system obtained, in time to attain the solution $\mathbf{u}_{num}(t + \Delta t)$ at time $t + \Delta t$.

5. On the same nodal framework, execute various steps of time.

4.3 Numerical results and discussions

To test the proposed SPGWOFD, Burger's equation (3.5.1) has been considered with different boundary cases. Based on the different types of boundary conditions used, following four problems are considered.

Test problem I

In the first test problem, the Burger's equation on an interval $[0, 1]$ has been considered with the initial $u(x, 0) = u_0(x) = \sin(2\pi x)$ and boundaries are periodic, *i.e.*, $u(x, t) = u(x+1, t)$. For $\epsilon = 10^{-4}$ and $\nu = 0.002$, Fig. (4.3) represents its solution and related adapted grid at distinct times. Fig. (4.4)(a), (4.4)(b) justifies the method's convergence in regards to $N(\epsilon)$ and ϵ respectively. The graphs show that the convergence order which we look for has been achieved. We have found that the error decreases as the ϵ value decreases, however the computational cost increases. ϵ is therefore to be sensibly selected that balances the error with the computational cost. Fig. (4.4)(c) displays the change in value Θ with respect to ν. It is found that as ν value decreases, the Θ value increases, that means the approach achieves better results for small values of ν. Table (4.1) shows the change in the CPU(ϵ) versus ϵ for value of $\nu = 0.002$. It is checked that as ϵ decreases, Θ decreases. The higher the Θ value, more efficient the adaptive algorithm is.

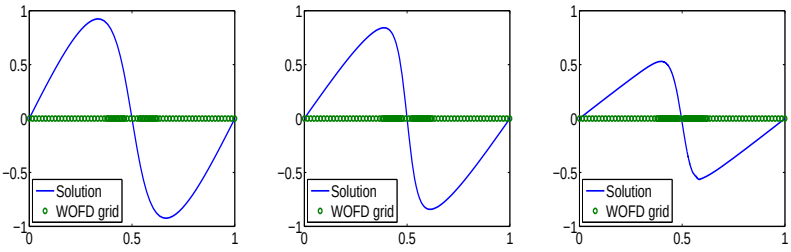
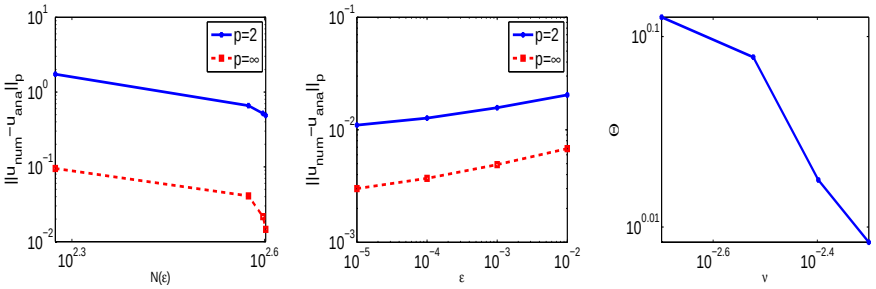


Figure 4.3: Solution and its corresponding adapted grid at $t = 0.1, 0.25, 0.5$.



(a) $\|u_{num} - u_{ana}\|_p$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_p$ vs. ϵ (c) Θ vs. ν

Figure 4.4: Results for the test problem I.

Table 4.1: The efficiency of SPGWOFD for the test problem I.

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}
CPU(ϵ)	0.0469	0.1406	0.1500	0.2188
Θ	5.997	2.0007	1.8753	1.2856

Test problem II

In the 2nd test problem, Burger's equation has been considered on an interval $[0, 1]$ with the initial given as $u(x, 0) = u_0(x) = \sin(\pi x)$ and Dirichlet boundaries, *i.e.*, $u(1, t) = u(0, t) = 0,\ t > 0$.

For $\nu = 0.002$ and $\epsilon = 10^{-4}$, Fig. (4.5) displays its solution and its related adaptive grid at distinct times. Fig. (4.6)(a), (4.6)(b) checks the method's convergence in regards to $N(\epsilon)$ and ϵ. It can be seen from these graphs that the convergence order which we looked for has been obtained. We have observed that error between numerical and analytical value of the problem decreases as $N(\epsilon)$ increases and decreases on decreasing the value of ϵ. Table 4.2 represents the variation of Θ for different values of ϵ. It is found that Θ increases on increasing the ϵ which shows the efficiency of our adaptive algorithm. Table (4.3) display explicit finite difference method, exact-explicit finite difference method and SPGWOFD solutions for $\nu = 1$, $\delta t = 0.00001$, final time $= 0.1$ and $N = 10$. We have observed that numerical predictions are well in agreement with analytical solution. $\|u_e - u_{ana}\|$, $\|u_{e_1} - u_{ana}\|$ and $\|u_{num} - u_{ana}\|$ (where u_e stands for numerically solution achieved by explicit finite difference technique, u_{e_1} stands for numerically solution achieved by exact-explicit finite difference technique, u_{num} represents the numerically solution calculated by using SPGWOFD and u_{ana} represents the problem's analytical solution). In all of the three cases, we have observed that SPGWOFD method is more accurate.

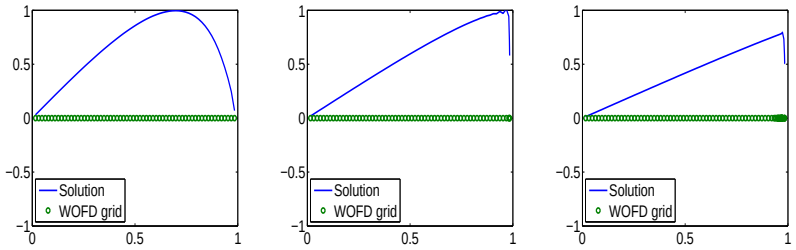
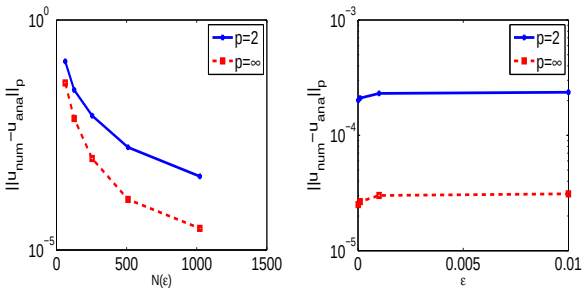


Figure 4.5: Solution and its corresponding adapted grid at $t = 0.25,\ 0.5,\ 0.95$.



(a) $\|u_{num} - u_{ana}\|_p$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_p$ vs. ϵ

Figure 4.6: Results for the test problem II.

Table 4.2: The efficiency of SPGWOFD for the test problem II.

ϵ	10^{-1}	10^{2}	10^{-3}	10^{-4}
CPU(ϵ)	0.0313	0.0469	0.0938	0.2031
Θ	7.488	4.997	2.4989	1.1541

Test Problem III

In the 3rd test problem, non-homogeneous Burger's equation has been considered

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = \nu \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad x \in [0,1], \tag{4.3.1}$$

where

$$f(x,t) = -\frac{1}{4}e^{-\nu t}\cos(\pi x)(\nu + \frac{\pi}{4}e^{-\nu t}\sin(\pi x) - \nu\pi^2),$$

with the initial $u(x,0) = \frac{1}{4}\cos(\pi x)$ and Neumann boundaries $u_x(0,t) = u_x(1,t) = 0$. For $\nu = 1$ and $\epsilon = 10^{-4}$, Fig. (4.7) displays the solution of this problem and its related grid at distinct times. Fig. (4.8)(a) represents the error (*i.e.*, $\|u_{num} - u_{ana}\|_p$) at time $t = 0.5$ versus $N(\epsilon)$. It can be observed from this graph that error decreases as $N(\epsilon)$ increases. Fig. (4.8)(b) represents the graph of $\|u_{num} - u_{ana}\|_p$ versus ϵ. It shows that error increases on increasing the ϵ. Table (4.4) gives the deviation of the CPU(ϵ) with

Table 4.3: Solution of test problem II by Explicit finite difference method, Exact-Explicit finite difference method and SPGWOFD method.

x	explicit finite-difference	exact-explicit finite-difference	SPGWOFD	exact solution
0.1	0.10863	0.11048	0.1101	0.1095
0.2	0.20805	0.21159	0.2108	0.2098
0.3	0.28946	0.29435	0.2932	0.2919
0.4	0.34501	0.35080	0.3493	0.3479
0.5	0.36845	0.37458	0.3727	0.3716
0.6	0.35601	0.36189	0.3596	0.3590
0.7	0.30728	0.31231	0.3097	0.3099
0.8	0.22588	0.22955	0.2266	0.2278
0.9	0.11966	0.12160	0.1185	0.1207
	0.0070	0.0067	0.0036	

ϵ. We have observed that as ϵ decreases, Θ decreases. The higher the Θ value, more effective the adaptive algorithm is. Table (4.5), (4.6) and (4.7) compare the problem's numerical soln. computed by Galerkin, Galerkin/Conservative and SPGWOFD method at final time $t = 1/2$. The finite dimensional models were set up using $N = 16$ elements and the equations were solved with Re=60, 120 and 240 (Re stands for Reynold's number and is equal to inverse of ν). The numbers in the bottom row of these tables are the error norms $\|u_G - u_{ana}\|$, $\|u_{G/C} - u_{ana}\|$ and $\|u_{num} - u_{ana}\|$ (where u_G stands for numerical solution calculated by Galerkin technique and $u_{G/C}$ stands for numerical solution calculated by Galerkin/Conservative technique). As Re increases these error norms become larger, indicating reduction in accuracy. This is to be expected since $\nu = \frac{1}{Re} \to 0$ corresponds to decreasing viscosity. In all of three cases we have observed that SPGWOFD method is more accurate.

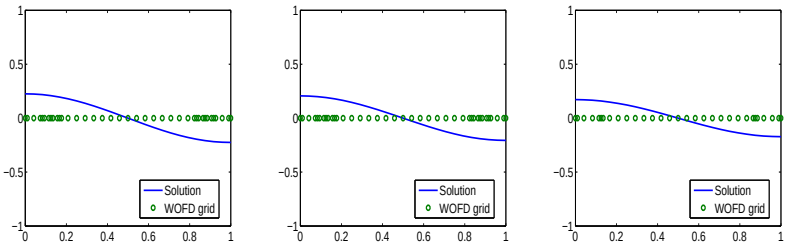
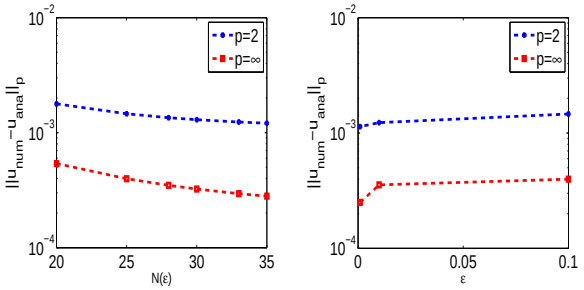


Figure 4.7: Solution and its corresponding adapted grid at $t = 0.125, 0.25, 0.5$.



(a) $\|u_{num} - u_{ana}\|_p$ vs. $N(\epsilon)$ (b) $\|u_{num} - u_{ana}\|_p$ vs. ϵ

Figure 4.8: Results for the test problem III

Table 4.4: The efficiency of SPGWOFD for the test problem III.

ϵ	10^{-1}	10^{-2}	10^{-3}	10^{-4}
CPU(ϵ)	3.5000	3.5313	3.5625	4.5000
Θ	1.558	1.544	1.5306	1.218

Test Problem IV

Consider non-homogeneous Burger's equation

$$\frac{\partial u(x,t)}{\partial t} + u(x,t)\frac{\partial u(x,t)}{\partial x} = \nu \frac{\partial^2 u(x,t)}{\partial x^2} + f(x,t), \quad x \in [0, 100], \tag{4.3.2}$$

with the initial $u(x,0) = \dfrac{x^2}{10^4}$ and non-homogeneous Robin's boundaries

$$\frac{\kappa_1}{L_1}u(0,t) - \kappa\frac{\partial u(0,t)}{\partial x} = \frac{\kappa_1}{L_1}u_1(t), \tag{4.3.3}$$

$$\frac{\kappa_2}{L_2}u(L,t) + \kappa\frac{\partial u(L,t)}{\partial x} = \frac{\kappa_2}{L_2}u_2(t), \tag{4.3.4}$$

Table 4.5: Solution of test problem III by Galerkin method, Galerkin/Conservative and SPGWOFD method (Re=60).

x	Galerkin	Galerkin/Conservative	SPGWOFD
0	0.2505	0.2504	0.24608
0.0588	0.2457	0.2456	0.24139
0.1176	0.2326	0.2325	0.22973
0.1765	0.2118	0.2116	0.21008
0.2353	0.1839	0.1836	0.18285
0.2941	0.1499	0.1496	0.1491780
0.3529	0.1108	0.1106	0.110345
0.4118	0.068	0.0679	0.06774
0.4706	0.0229	0.0229	0.022839
0.5294	−0.0229	−0.0229	−0.22839
0.5882	−0.068	−0.0679	−0.06774
0.6471	−0.1108	−0.1106	−0.110345
0.7059	−0.1499	−0.1496	−0.1491780
0.7647	−0.1839	−0.1836	−0.18285
0.8235	−0.2118	−0.2116	−0.21008
0.8824	−0.2326	−0.2325	−0.22973
0.9412	−0.2457	−0.2456	−0.24139
1	−0.2505	−0.2504	−0.24608
	0.0053	0.0049	0.0048

where

$$f(x,t) = \alpha\frac{(100-x)^2}{10^4}\cos(\alpha t) - \beta\frac{x^2}{10^4}\sin(\beta t) - 2\frac{(100-x)^3}{10^8}\sin^2(\alpha t) + 2\frac{x^3}{10^8}\cos^2(\beta t) + \frac{2}{10^8}(2x^3 - 300x^2 + 10^4 x)\sin(\alpha t)\cos(\beta t) - \frac{2\nu}{10^4}(\sin(\alpha t) + \cos(\beta t)),$$

$$u_1(t) = (1 + \frac{2\kappa L_1}{100\kappa_1})\sin(\alpha t), \tag{4.3.5}$$

$$u_2(t) = (1 + \frac{2\kappa L_2}{100\kappa_2})\cos(\beta t), \tag{4.3.6}$$

Table 4.6: Solution of test problem III by Galerkin method, Galerkin/Conservative and SPGWOFD method (Re=120).

x	Galerkin	Galerkin/Conservative	SPGWOFD
0	0.2529	0.2528	0.2471
0.0588	0.2474	0.2474	0.2417
0.1176	0.2338	0.2337	0.2305
0.1765	0.2127	0.2125	0.2110
0.2353	0.1846	0.1843	0.1836
0.2941	0.1504	0.1502	0.1498
0.3529	0.1113	0.111	0.1108
0.4118	0.0683	0.0681	0.0680
0.4706	0.023	0.023	0.0229
0.5294	−0.023	−0.023	−0.0229
0.5882	−0.0683	−0.0681	−0.0680
0.6471	−0.1113	−0.111	−0.1108
0.7059	−0.1504	−0.1502	−0.1498
0.7647	−0.1846	−0.1843	−0.1836
0.8235	−0.2127	−0.2125	−0.2110
0.8824	−0.2338	−0.2337	−0.2305
0.9412	−0.2474	−0.2474	−0.2417
1	−0.2529	−0.2528	−0.2471
	0.0073	0.0071	0.0056

with $L = 100, L_1 = L_2 = 10, \kappa = 0.93, \kappa_1 = \kappa_2 = 0.55, \nu = 1.14, \alpha = 1.5\pi, \beta = 0.5\pi$. The problem's exact solution is

$$u(x,t) = \frac{(100-x)^2}{10^4}\sin(\alpha t) + \frac{x^2}{10^4}\cos(\beta t).$$

For $\nu = 1.14$ and $\epsilon = 10^{-4}$, Fig. (4.9) displays the solution of the problem and its related adaptive grid at distinct times. As the solution is smooth, therefore grid is not refined as expected. Fig. (4.10)(a) shows the error (*i.e.*, $\|u_{num} - u_{ana}\|_\infty$) at time $t = 0.5$ versus $N(\epsilon)$. It can be seen from the graph that error decreases as $N(\epsilon)$ increases. Fig. (4.10)(b) compares the analytical solution of the problem with its SPGWOFD solution.

Table 4.7: Solution of test problem III by Galerkin method, Galerkin/Conservative and SPGWOFD method (Re=240).

x	Galerkin	Galerkin/Conservative	SPGWOFD
0	0.2556	0.2556	0.2480
0.0588	0.249	0.2489	0.2414
0.1176	0.2345	0.2343	0.2310
0.1765	0.213	0.2128	0.2116
0.2353	0.1849	0.1846	0.1841
0.2941	0.1508	0.1504	0.1501
0.3529	0.1115	0.1112	0.1110
0.4118	0.0685	0.0683	0.0681
0.4706	0.0231	0.023	0.0230
0.5294	−0.0231	−0.023	−0.0230
0.5882	−0.0685	−0.0683	−0.0681
0.6471	−0.1115	−0.1112	−0.1110
0.7059	−0.1508	−0.1504	−0.1501
0.7647	−0.1849	−0.1846	−0.1841
0.8235	−0.213	−0.2128	−0.2116
0.8824	−0.2343	−0.2344	−0.2310
0.9412	−0.249	−0.2489	−0.2414
1	−0.2556	−0.2556	−0.2480
	0.0107	0.0105	0.006

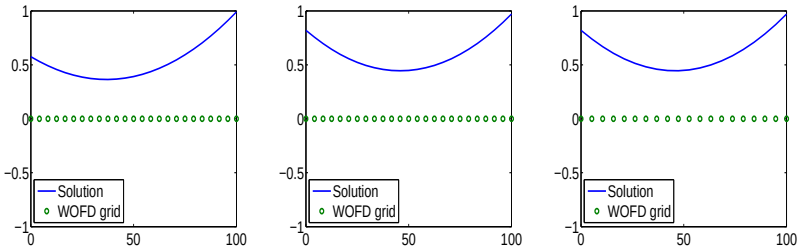
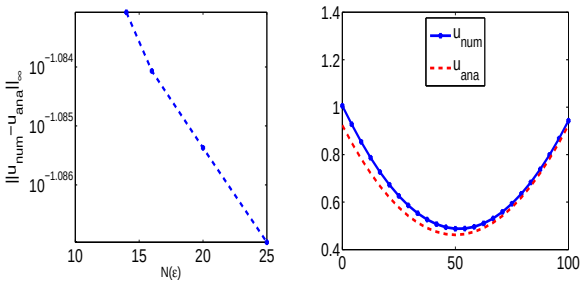


Figure 4.9: Solution and its corresponding adapted grid at $t = 0.0625, 0.125, 0.25$.

4.4 Conclusion and Future directions

In this chapter, spectral graph wavelet based wavelet optimized finite difference technique is proposed to solve the Burger's equation which is called SPGWOFD. Central finite dif-



(a) $\|u_{num} - u_{ana}\|_\infty$ vs. $N(\epsilon)$ (b) Comparison between numerical and analytical value.

Figure 4.10: Results for the Test Problem IV.

ference matrices are used for approximations of differential operators involved in a given PDE. Adaptive grid on which the solution is obtained is generated using SPGW. Finally time integration Crank-Nicolson technique is used. The SPGWOFD is tested on Burger's equation with all boundary conditions (periodic, Dirichlet, Neumann and Robin). The convergence and efficiency of SPGWOFD are checked for all the four problems of testing. Comparison of CPU time reveals that the developed method is highly efficient. Having checked the efficiency of the developed method for Burger's equation, we propose the use of SPGWOFD for turbulence modeling and for solving problems of fluid dynamics.

Chapter 5

Curvelet optimized finite difference method for PDEs

Despite the many advantages of wavelets, they have some limitations such as poor orientation sensitivity. To mitigate these limitations, curvelets were introduced by Demanet and Candes [179]. Curvelets are attractive, because they effectively describe essential problems in which wavelet ideas are not upto mark.

1. Because of the bad orientation selectivity of wavelets, they do not present higher-dimensional irregularities efficiently. This makes curvelets interesting as they can yield an ideally adapted numerical construction to represent functions that display smooth punctuated curve.

2. Wavelets are not sufficient to detect, classify or present an intermediate dimensional arrangement with a compressed description. Curvelet is the better product because it produces better-adapted choices by consolidating concepts from geometry with ideas from classical multiscale analysis.

4. Wavelets are optimal for solving elliptical PDEs [180], but not for hyperbolic PDEs. For hyperbolic PDEs, curvelets provide better results. The hyperbolic solution operator's curvelet representation is efficient as well as ideally sparse [181].

Curvelets take the class of basis elements which are extremely sensitive to direction and extremely anisotropic. Using curvelets to solve PDEs, is one of the recommended forthcoming directions in [75] which is the locus of this chapter. In this chapter, curvelet optimized finite-difference approach (COFD) is proposed for solving PDEs. The method uses finite difference approximations for differential operators involved in the PDEs. After the approximation, curvelet is used for the compression of the finite difference matrices and subsequently for computing the dyadic powers of these matrices required for solving the PDE in a fast and efficient manner, not to produce an adaptive grid. We have taken the advantage of curvelets in order to generate the adaptive grid in the next chapter. In addition, compression and reconstruction errors for the curvelet were tested for various parameters. The proposed technique has been applied on five test problems of different nature. The method's convergence is verified for each problem. Moreover, to measure the

efficiency of the developed technique the computational time carried out by the developed technique is compared to that of finite difference method. It is observed that the developed technique is computationally very efficient.

5.1 A short explanation of curvelet

5.1.1 Drawbacks of Classical Multiscale Approaches

Our aim in this work is to rose curvelets for solving PDEs. In that area, we endeavor a sparse presentation of the grid. Over the past two decades, multiscale approaches such as multi-grid, fast multi-polar techniques in wavelets, finite element techniques with or without adaptive refining have been widespread. All these descriptions mean references to approximately isotropic elements in all positions as well as in scales; the template is re-scaled and handled essentially in the same way in all areas. Isotropic scaling might be useful if the function under examination has no particular highlights along the orientations chosen. Tools inherited from multi-scale analysis such as wavelets are insufficient to detect, establish or present a compact presentation of the intermediate dimensional arrangement. Curvelet is the better product because it produces better- adapted choices by consolidating concepts from geometry with ideas from classical multiscale analysis. Curvelets can produce a geometrical framework that is ideally suited to describe functions that demonstrate punctuated smoothness of the curve.

5.1.2 Introduction for Curvelets

Curvelets are waveforms that have anisotropic behavior in fine layers, with an efficient support that accepts the parabolic postulate, width^2 $\approx$ length [182]. There is a discrete as well as continuous transformation of the curvelet as with the wavelets. A curvelet is classified by three quantities particularly, a parameter of orientation, say, $\theta, \theta \in [-\frac{\pi}{2}, \frac{\pi}{2})$; a scaling parameter $j, 0 < j < 1$ and a position parameter k where $k \in R^2$. At level a, the class of curvelet is produced by dilate, translate, revolution of a fundamental component φ_j

$$\varphi_{j,k,\theta}(x) = \varphi_j(R_\theta(x-k)).$$

here, $\varphi_j(x)$ is any type of spatial wavelet with spatial breadth $\sim j$ and spatial height $\sim \sqrt{j}$, (the symbol $\sim$ stands for proportional) among the minor axis facing in the horizontal

position, which is described as

$$\varphi_j(x) \approx \varphi(D_j x),$$

here $D_j = \begin{pmatrix} 1/j & 0 \\ 0 & 1/\sqrt{j} \end{pmatrix}$ is the parabolic-scaling matrix and $R_\theta = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$ is a 2×2 rotation matrix with θ angle.

A significant feature of curvelets is that it obeys the fundamental law of harmonic analysis, which states that evaluating and restoring an arbitrarily function $f(x_1, x_2)$ as the superpositions of that models is reasonable. It is acceptable to create stiff curvelet-frames and an arbitrarily function $f(x_1, x_2)$ can certainly be extended as an orthonormal set of curvelets. There is a discretisation of angle/location/scale proceeding at a straightforward level of analysis, that approximately goes like $j_a = 2^{-a}, a = 0, 1, ..., \theta_{a,l} = 2\pi l.2^{-\lfloor a/2 \rfloor}, l = 0, 1, ..., 2^{\lfloor a/2 \rfloor} - 1$, and $b_k^{(a,l)} = R_{\theta_{a,l}}(k_1 2^{-j}, k_2 2^{-a/2})$, $k_1, k_2 \in Z$, the set φ_μ is a tight frame, here μ indexing the triplets $(j_a, \theta_{a,l}, k_b^{(a,l)})$. It implies

$$f = \sum_\mu < f, \varphi_\mu > \varphi_\mu, \qquad \|f\|_2^2 = \sum_\mu | < f, \varphi_\mu > |^2. \tag{5.1.1}$$

When a scale is given, curvelets φ_μ are achieved by implementing translations and orientations of the "mother" curvelet $\varphi_{a,0,0}$. In the frequency region,

$$\hat{\varphi}_{a,0,0}(\xi) = 2^{-3a/4} W(2^{-a}|\xi|) V(2^{\lfloor a/2 \rfloor}\theta).$$

In this case V and W are of compactly supported with continuous windows at interval $[1, 2]$ and $[-1/2, 1/2]$ respectively. Curvelets exists in the frequency region adjacent to an orientated rectangle R of width 2^{-a} and length $2^{-a/2}$ while into the spatial region, curvelets are positioned into a parabolic drive of length 2^a and width $2^{a/2}$ with an orthogonal orientation to R. The combined localization in frequency as well as in space permits us to study curves such as the use of the "Heisenberg cell" into the phase space and parabolic scales in these two realms.

5.1.3 Importance of Curvelets over Wavelets

The curvelets give optimum sparseness for "curve-punctuated continuous" function, wherever the function is continuous excluding discontinuities together with C^2 curves [183,184]. The degree of decline in the m-term estimate (reconstructing the function by employing m no. of coefficients) of the data is estimated to be sparse. A sparse description, along with improved compression possibilities and additional sparseness for the remodeling of

the denoising performance, increases the number of smooth functional areas. The curvelet transform is designed in such a fashion that maximum energy of the function is limited in only a few coefficients. This is measurable. There is clearly no basis for a function's coefficients with an uncertain curve of singularity to decline more quickly than in a frame of curvelet. This decline estimate is fast as compared to any other known system, included wavelets. The improved decline in the coefficient provides an optimally sparse representation, which is useful for solving the equation on general manifolds.

5.1.4 Continuous-time Curvelet transform

We begin with the set of $V(t)$ and $W(r)$ windows, which is referred as the "radial window" and the "angular window". The two windows are positive, real valued and continuous with W getting non-negative real argument along with carried out on $r \in [1/2, 2]$ and V using real argument along with carried out on $t \in [-1, 1]$. These both windows will meet the requirements for adequacy, *i.e.*,

$$\sum_{j=-\infty}^{j=\infty} W^2(2^j r) = 1, \ r > 0; \sum_{l=-\infty}^{l=\infty} V^2(t-l) = 1, \ t \in R.$$

For every $j \geq j_0$, we start with a U_j window of frequency, described in the Fourier region as

$$U_j(r, \theta) = 2^{-3j/4} W(2^{-j} r) V\left(\frac{2^{\lfloor j/2 \rfloor} \theta}{2\pi}\right), \tag{5.1.2}$$

here $\lfloor \frac{j}{2} \rfloor$ is the integer portion of $\frac{j}{2}$. Therefore, the U_j support is a polar "wedge" described as the W and V support, the angular and the radial-window, used in all directions with scale-dependent window widths.

With the aid of its Fourier transformation, set the waveform $\varphi_j(x)$ as $\hat{\varphi}_j(w) = U_j(w)$. We can consider φ_j as the "mother" curvelet on 2^{-j} scale means that, on this scale, all curvelets are achieved by translating and rotating the φ_j. Define,

- the equi-spaced series of rotated angles $\theta_l = 2\pi.2^{-\lfloor j/2 \rfloor}.l$ with $l = 0, 1, ...$ means that $0 \leq \theta_l < 2\pi$.
- the sequence of the $k = (k_1, k_2)$ translation parameters of the Z^2.

We define curvelets with these notations

$$\varphi_{j,l,k} = \varphi_j(R_{-\theta_{j,l}}(x - b_k^{(j,l)})). \tag{5.1.3}$$

here R_θ^{-1}, the inverse of R_θ is described as,

$$R_\theta^{-1} = R_\theta^T = R_{-\theta}.$$

The coefficient of a curvelet is merely the inner product of the curvelet $\varphi_{j,l,k}$ and a function $f \in L^2(R^2)$,

$$c(j,l,k) :=< f, \varphi_{j,l,k} >= \int_{R^2} f(x)\overline{\varphi_{j,l,k}}(x)dx. \tag{5.1.4}$$

The curvelet's coefficients, by using the theorem of Plancherel, can be represented as

$$c(j,l,k) = \frac{1}{(2\pi)^2}\int \hat{f}(w)\overline{\hat{\varphi}_{j,l,k}}(w)dw = \frac{1}{(2\pi)^2}\int \hat{f}(w)U_j(R_{\theta_l}w)e^{i<x_k^{(j,l)},w>}dw.$$

We have coarse-scale elements too, as in wavelet theory. The low-pass window W_0 introduced as a function that follows the following equation

$$|W_0(r)|^2 + \sum_{j\geq 0}|W(2^{-j}r)|^2 = 1,$$

and for $k_1, k_2 \in Z$, coarsest scale curvelets are then described by

$$\Phi_{j_0,k}(x) = \Phi_{j_0}(x - 2^{-j_0}k), \quad \hat{\Phi}_{j_0}(\xi) = 2^{-j_0}W_0(2^{-j_0}|\xi|). \tag{5.1.5}$$

The complete curvelet transformation consist of the directional finest scale elements $(\varphi_{j,l,k})_{j\geq j_0,l,k}$ along with an isotropic coarsest level father wavelets $(\Phi_{j_0,k})$.

5.1.5 Fast discrete curvelet transform

In 2006, Demanet et al. developed two fast-discrete curvelet transformations (FDCT). One relies on the unevenly distributed fast-Fourier transformations (USFFT) [182] and the another one depends on the binding of specifically chosen Fourier-samplings (FDCT WRAPPING) [182]. We have used FDCT wrapping in this work, as it is the rapid curvelet transformation. After curvelet transformation, various groups of coefficients of curvelets at distinct angles and scales are constructed. The coefficients of curvelets at angle l and at scale j are expressed as a matrix $c_{j,l}$ where j is from finer level to coarser level and l begins at the upper-left edge and raises clockwise.

Assume that $f(t_1,t_2), 1 \leq t_1 \leq N_1,\ 1 \leq t_2 \leq N_2$ refers to the original function and $\hat{f}[n_1,n_2]$ defines $2D$ discrete Fourier transformation.
The FDCT wrapping implementation is as below:

Step 1. Two dimensional fast Fourier transformation (FFT) is implemented on $f(t_1, t_2)$ to achieve $\hat{f}[n_1, n_2]$ Fourier coefficients.
Step 2. The new sampling function is provided by re-sampling $\hat{f}[n_1, n_2]$ at every combination of direction and scale l, j into the frequency-region as:

$$\hat{f}[n_1, n_2 - n_1 \tan\theta_l], \quad (n_1, n_2) \in P_j, \tag{5.1.6}$$

here $P_j = \{(n_1, n_2), n_{1,0} \leq n_1 < n_{1,0} + L_{1,j}, n_{2,0} \leq n_2 < n_{2,0} + L_{2,j}\}$ and $n_{2,0}, n_{1,0}$ both are the original locations of window function $\widetilde{U}_{j,l}[n_1, n_2]$. $L_{2,j}$, $L_{1,j}$ respectively are significant constants of $2^{j/2}$ and 2^j and are constituents of the width and length of the support-interval of the window function.
Step 3. Multiply the new sampling function $\hat{f}[n_1, n_2 - n_1 \tan_{\theta_l}]$ with a window function $\widetilde{U}_{j,l}[n_1, n_2]$

$$\widetilde{f}_{j,l}[n_1, n_2] = \hat{f}[n_1, n_2 - n_1 \tan_{\theta_l}]\widetilde{U}_{j,l}[n_1, n_2], \tag{5.1.7}$$

where

$$\widetilde{U}_{j,l}[n_1, n_2] = W_j(w_1, w_2)V_j\big(S_{\theta_l}.\frac{2^{\lfloor j/2 \rfloor}w_2}{w_1}\big),$$

$$W_j(w_1, w_2) = \sqrt{\Phi^2_{j+1}(w) - \Phi^2_j(w)},$$

$$\Phi_j(w_1, w_2) = \phi(2^{-j}w_1)\phi(2^{-j}w_2),$$

$$S_{\theta_l} = \begin{pmatrix} 1 & 0 \\ -\tan\theta_l & 1 \end{pmatrix},$$

$$\tan\theta_l = l \times 2^{\lfloor -j/2 \rfloor}, \quad l = -2^{\lfloor -j/2 \rfloor}, \cdots, 2^{\lfloor -j/2 \rfloor} - 1.$$

Step 4. To each $\widetilde{f}_{j,l}$, apply the inverse two dimensional fast Fourier-transformation (FFT) and therefore the discrete coefficients $c_{j,l}$ are obtained.
The FDCT inverse Wrapping algorithm is as follows:

- For each array of the curvelet coefficients.
  a. Take the FFT of the array.
  b. Uncover the rectangular support to the initial orientation state.
  c. Translate to the original location.
- Add all the translated curvelet arrays.
- To reconstruct the function, take the inverse FFT.

5.2 Compression and reconstruction error

For a given function $f(x)$ and the given threshold ϵ, curvelet expansion of function f can be described as $f(x) = f_{\geq\epsilon}(x) + f_{<\epsilon}(x)$, where

$$f_{\geq\epsilon} = \sum_{|<f,\varphi_n>|\geq\epsilon} < f, \varphi_n > \varphi_n \text{ and } f_{<\epsilon} = \sum_{|<f,\varphi_n>|<\epsilon} < f, \varphi_n > \varphi_n.$$

$\|f - f_{\geq\epsilon}\|_p$ is known as the compression-error and reconstruction error stands for no compression, *i.e.*, $\epsilon = 0$.

5.2.1 Algorithm for Compression using Curvelet transform

1. Calculate the curvelet coefficients as

$$c(j, l, k) = \int_{R^2} f(x)\overline{\varphi}_{j,l,k}(x)dx, \tag{5.2.1}$$

   here R represents the real line.

2. Arrange the curvelet's coefficients in descending order.

3. Threshold for coefficients of curvelets is shown as below:

$$M = CPR * size\ of\ function, \quad cut - off = CL * (M). \tag{5.2.2}$$

   where CPR is compression ratio and CL is an array of the coefficients of the curvelet arranging in non-ascending order.

4. Delete the coefficients whose magnitude is less than cutoff

$$C1 = C > cutoff.$$

   The curvelet coefficients whose magnitude is below than cutoff are referred to as insignificant curvelet coefficients while remaining are referred to as significant curvelet coefficients.

5. To get compressed function, execute inverse curvelet-transformation of $C1$.

Now, we will test the behaviour of compression error versus no. of grid points and threshold ϵ (chosen by the user) for two test functions.

Test function I:- $f(x, y) = \exp(-1000(x^2 + y^2))$ on the domain $[-1, 1] \times [1, 1]$ is chosen as our first test function and it is shown in Fig. (5.1)(a).

- Error in reconstructing the function is of the order 10^{-16}.

- Fig. (5.1)(b) represents the graph of the compression error *i.e.*, $\|f - f_{\geq\epsilon}\|_p$ versus no. of grid points. The graph displays the good compression.

- Fig. (5.1)(c) displays the relationship between threshold and compression-error *i.e.*, $\|f - f_{\geq\epsilon}\|_p$. It is observed that compression-error increases on increasing the threshold. The reason for this trend is the fact that with increase in the value of threshold, coefficients of curvelets will be discarded.

- Fig. (5.1)(d) and (5.1)(e) compares the compression error for curvelet and Daubechies wavelet. It can be observed that curvelets are more efficient than wavelets. The wavelet named 'db1' is used in both the cases.

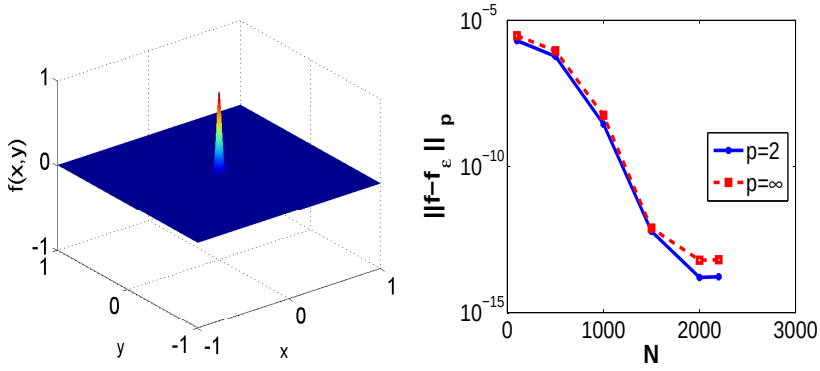
- Fig. (5.1)(f) shows log of curvelet coefficients. In this figure white part shows the significant coefficients of curvelet. The coefficients of the coarsest level (low-frequency) are saved in the middle of the array. The cartesian concentrical coronae demonstrate the coefficients at various levels such as the external coronae relate to high frequencies.

Test function II:- For the function $f(y, x) = \exp(-50(y + 1)^2 + \exp(-50(x + 1)^2))$ on the domain $[-1, 1] \times [1, 1]$ which is shown in Fig. (5.2)(a), we got the results as explained below:

- Error in reconstructing the function is of the order 10^{-15}.

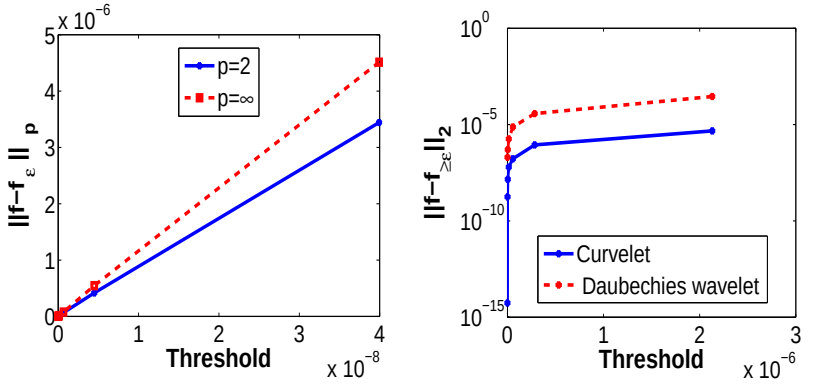
- Fig. (5.2)(b) displays the graph of compression error with respect to no. of grid points. The graph displays the good compression and Fig. (5.2)(c) represents the relationship between the threshold and compression error *i.e.*, $\|f - f_{\geq\epsilon}\|_p$. We have found from the graph that compression error increases on increasing the threshold value.

- Fig. (5.2)(d) and (5.2)(e) represents the comparison of the compression error versus threshold for the case of curvelet and Daubechies wavelet. It is shown that curvelets are more efficient than wavelets.



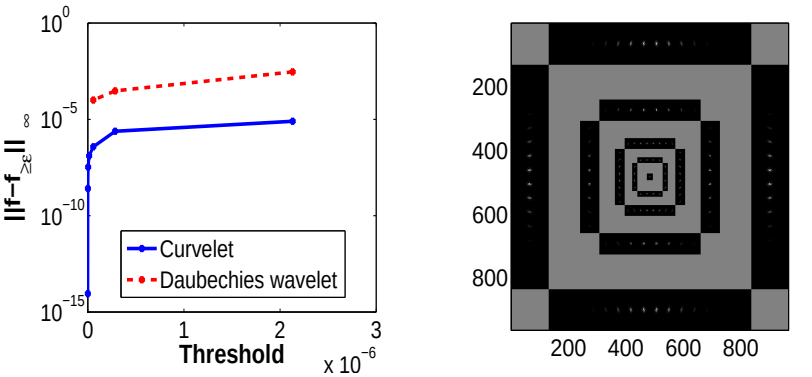
(a) The test function 1

(b) Compression error versus N



(c) Compression error versus threshold

(d) Comparison with Daubechies wavelet(in l_2 norm)



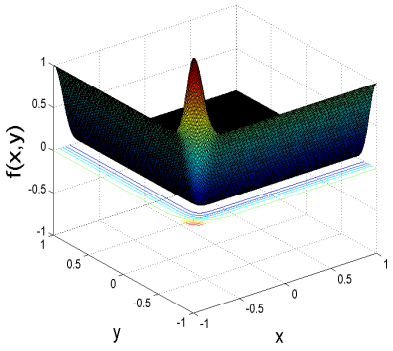
(e) Comparison with Daubechies wavelet(in l_∞ norm)

(f) log of curvelet coefficients

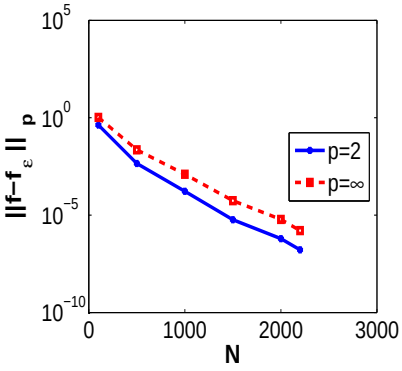
Figure 5.1: Results for the test function I.

5.3 Solving the PDEs using curvelets

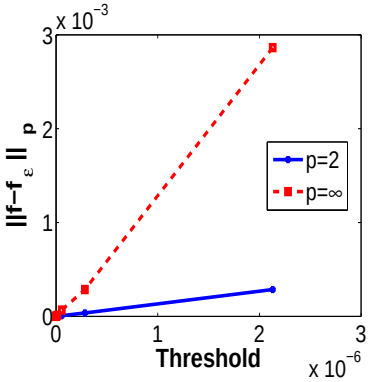
In the following text the fast curvelet based finite difference method (FCFD) is used to solve PDEs is explained. The factor which motivates the development of curvelet based numerical techniques for solving PDEs is that curvelet description of the finite difference



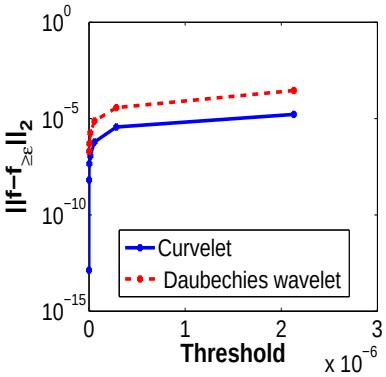
(a) The test function 2



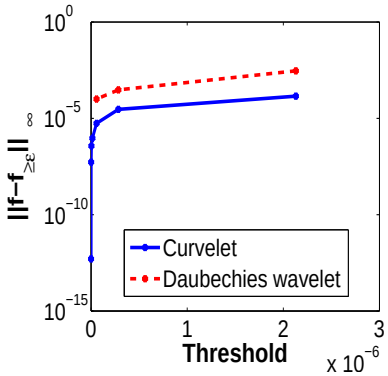
(b) Compression error versus N



(c) Compression error versus threshold



(d) Comparison with Daubechies wavelet(in l_2 norm)



(e) Comparison with Daubechies wavelet(in l_∞ norm)

Figure 5.2: Results for the test function II.

matrices associated with PDEs is well organized and sparse [182]. The main characteristic of the proposed method is to represent the differential operator in a curvelet basis φ_μ of $L^2(R^m)$ or in other words apply curvelet transform on the finite difference matrices approximating the differential operators. Having applied the curvelet transform, one can discard all the curvelet coefficients which are having magnitude less than a pre decided threshold

ϵ. After that, all the computations are performed using the sparse matrices and at the end the inverse FDCT is applied to get the solution at the final time. In [185], curvelet based finite-difference approach for seismic-wave equation has been considered, one can see it for more details.

To explain the method in detail, a 2-D diffusion equation given as follows is considered

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + f(x,y); 0 \leq x, y \leq 1. \tag{5.3.1}$$

with an initial profile $u(x, y, t = 0) = u_0(x, y)$ and with appropriate boundaries such as Periodic, Neumann, Dirichlet or Robin's. On applying time discretization scheme, the Eq.(5.3.1) takes the following form

$$\mathcal{A}u^n = \mathcal{C}u^{n-1} + \Delta t f, \quad u^0 = u_0,$$

where u^n approximated u at time $t = n\Delta t$. $\mathcal{A}$ and $\mathcal{C}$ are differential operators associated with the time discretization method. For example $\mathcal{A} = I, \quad \mathcal{C} = \left(I + \Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right)$ for the explicit forward Euler's scheme and $\mathcal{A} = \left(I - \Delta t\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\right), \quad \mathcal{C} = I$ for the implicit backward Euler's scheme.

After spatial discretization, we get

$$Au^n = Cu^{n-1} + \Delta t f, \quad u^0 = u_0, \tag{5.3.2}$$

where u^n is the vector of all unknowns $u^n_{i,j}, i = 1, \cdots, N; j = 1, \cdots, M$ at time $t = n\Delta t$. Eq.(5.3.2) can be written as

$$u^n = A^{-1}(Cu^{n-1} + \Delta t f). \tag{5.3.3}$$

By using the Eq.(5.3.3) recursively, we obtain

$$u^n = (A^{-1})^n C^n u^0 + \sum_{k=0}^{n-1} (A^{-1})^k C^k \Delta t A^{-1} f. \tag{5.3.4}$$

When A=I, the above equation takes the following form

$$u^n = C^n u^0 + \sum_{k=0}^{n-1} C^k \Delta t f. \tag{5.3.5}$$

Now if we choose, $n = 2^m$ for some integer m which actually means that we are computing the numerical solution at times $2\Delta t, 4\Delta t, 8\Delta t, \cdots$, then

$$\begin{aligned}
\sum_{k=0}^{n-1} C^k \Delta f &= \sum_{k=0}^{2^m-1} C^k \Delta t f, \\
&= (I + C + C^2 + \cdots + C^{2^m-1})\Delta t f, \\
&= \left((I+C) + C^2(I+C) + C^4(I+C) + C^6(I+C) + \cdots C^{2^m-2}(I+C)\right)\Delta t f, \\
&= (I+C)(I + C^2 + C^4 + C^6 + \cdots C^{2^m-2})\Delta t f, \\
&= (I+C)(I+C^2)(I + C^4 + C^8 + \cdots C^{2^m-4})\Delta t f, \\
&\vdots \\
&= \prod_{k=0}^{m-1} (I + (C)^{2^k})\Delta t f. \qquad (5.3.6)
\end{aligned}$$

Using Eq.(5.3.6), Eq.(5.3.5) can be written as

$$u^{2^m} = C^{2^m} u^0 + \prod_{k=0}^{m-1} (I + C^{2^k})\Delta t f. \qquad (5.3.7)$$

It is clear from the Eq. (5.3.7), that the computation of the solution at time $t = 2^m \Delta t$ involves only the repeated squaring of the matrix C which leads to the following algorithm for computing the solution by compressing the dyadic powers of C using curvelets:

Algorithm for FCFD

- **Step 1:** Apply curvelet transform (FDCT) on the matrix C. It gives us a matrix of curvelet coefficients of C. Call this matrix as D.
- **Step 2:** Let $E = I$ (Identity matrix).
  Iterate the following two steps Step 3 and Step 4 m times
- **Step 3:** Replace E with the new matrix which is created by applying threshold ϵ on $E + DE$ (it means discard all the elements which are having magnitude less than ϵ).
- **Step 4:** Replace D with a new matrix which is created by applying threshold ϵ on $D * D$.
  Note that after **Step 4**. we get the curvelet coefficients of C^{2^m} and $\prod_{k=0}^{m-1} (I + (C)^{2^k})$ respectively in the form of D and E.
- **Step 5:** Apply inverse FDCT on D and E and call the new matrices as D_1 and E_1.
- **Step 6:** Solution at time $2^m \Delta t$ is then given by $u^{2^m} = D_1 u^0 + E_1(\Delta t) f$.

It should be noted that, if we do not choose $n = 2^m$, then an algorithm similar to the above can not be designed. Moreover, given a final time T at which we have to compute the solution of the PDE, we can always find an integer m such that $2^m \Delta t$ is approximately equal to T.

5.4 Numerical Outcomes and Discussions

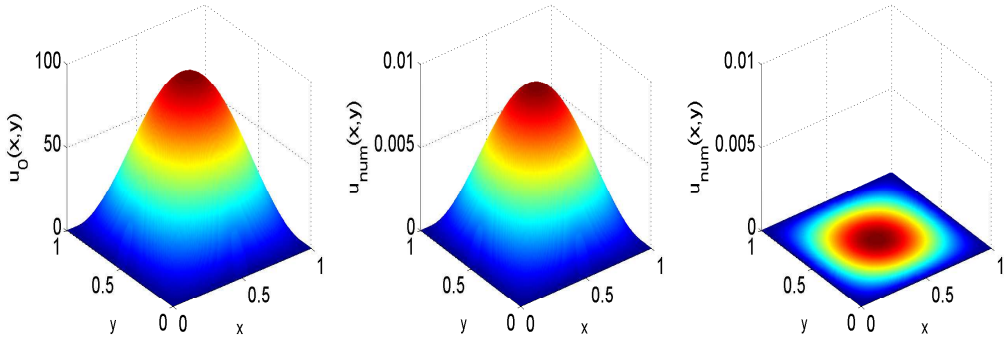
In this section, we examine the outcomes attained numerically with five test problems by using FCFD.

Test problem I

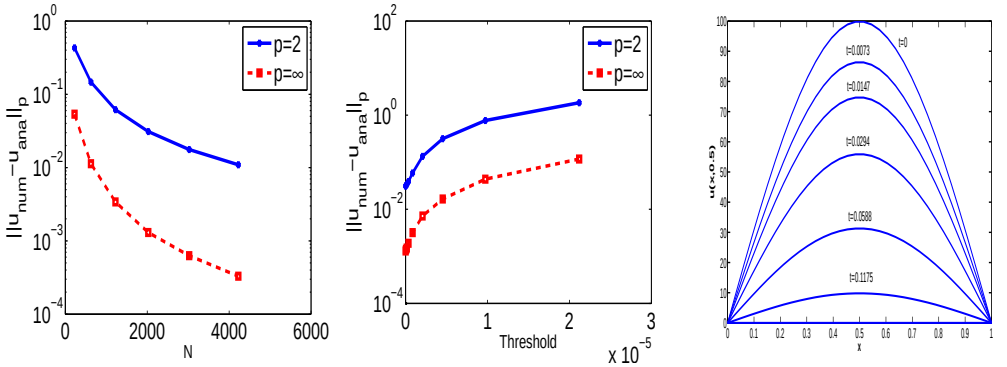
Take $f(x, y) = 0$ in Eqn. (5.3.1) with an initial $u(x, y, t = 0) = u_0(x, y) = 100 \sin(\pi x) \sin(\pi y)$ and boundaries are $u(x = 1, y, t) = u(x = 0, y, t) = u(x, y = 1, t) = u(x, y = 0, t) = 0$. Fig. (5.3)(a), (5.3)(b), (5.3)(c) displays the problem's solution at distinct times. We can see that with time the solution diffuses as expected. Fig. (5.3)(d) shows the error ($i.e., \|u_{num} - u_{ana}\|_p$) versus no. of grid points at time $t = 0.4848$. It can be observed from the graph that as no. of grid points increases, error decreases. Fig. (5.3)(e) demonstrates the graph between threshold and $\|u_{num} - u_{ana}\|_p$. Fig. (5.3)(f) displays the problem's solution at $y = 0.5$ at distinct times and Fig. (5.3)(g) compares the computational time carried out by FCFD with that of finite difference technique for evaluating the soln. at $t = 2^m \Delta t$ time. It is shown that computational time taken by FCFD is much less. Fig. (5.3)(h) shows log of curvelet coefficients of the problem's numerical solution. Table (5.1) gives the variation of CPU time taken (threshold) with threshold. It is found that as Θ decreases, threshold decreases. The higher the Θ value, more efficient the adaptive algorithm is.

Table 5.1: The performance of FCFD for the test problem I.

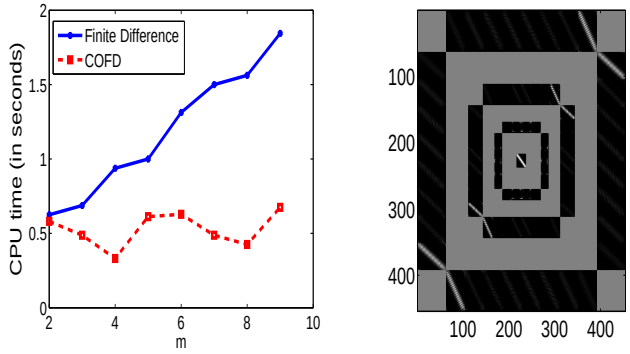
Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.2657	0.4469	0.5469
Θ	2.234	1.32	1.085



(a) The solution at time $t = 0$ (b) The solution at time $t = 0.4848$ (c) The solution at time $t = 3.8784$



(d) $\|u_{num} - u_{ana}\|_p$ versus N (e) $\|u_{num} - u_{ana}\|_p$ versus threshold (f) $u(x, 0.5)$



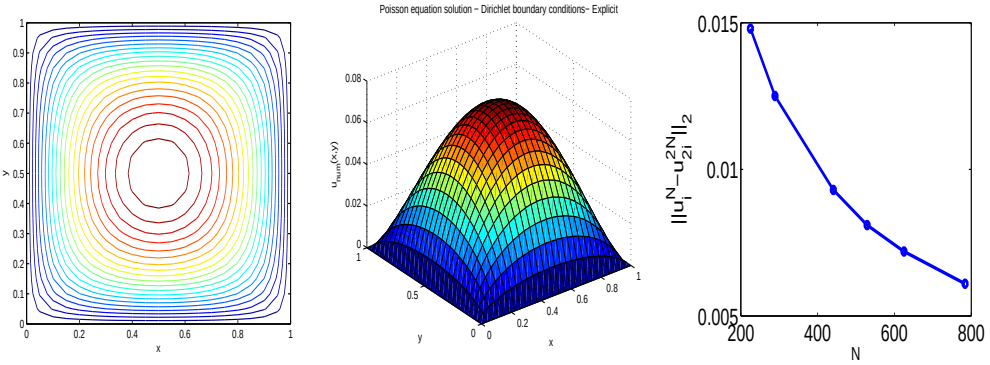
(g) CPU time for computing the solution at the time $t = 2^m \Delta t$ (h) log of curvelet coefficients

Figure 5.3: Results for the test problem I.

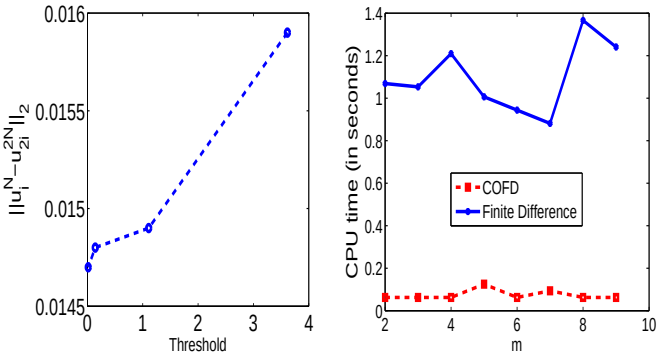
Test Problem II

Now we consider two-dimensional Poisson equation

$$-\Delta u = 1 \quad in \quad \Omega = [0, 1]^2,$$



(a) Contour plot of solution (b) Poisson equation solution (c) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N



(d) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold

(e) CPU time for computing the solution at the time $t = 2^m \Delta t$

Figure 5.4: Results for the test problem II.

Table 5.2: The performance of FCFD for the test problem II.

Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.0103	0.0198	0.0254
Θ	3.038	1.581	1.232

with boundaries

$$u(x = 1, y) = u(x = 0, y) = u(x, y = 1) = u(x, y = 0) = 0.$$

Pick a step $h = \frac{1}{M}$, here M is non-negative number. The discretization of our function is a sequence of elements $u_{i,j}$ with $0 \leq i, j \leq M$. The boundary conditions translate to $u_{n,j} = u_{0,j} = u_{i,n} = u_{i,0} = 0$ for $0 \leq i, j \leq M$. Now on the points situated in the interior

of (0, 1), we have the relation

$$\frac{1}{h^2}(4u_{i,j} - u_{i,j-1} - u_{i,j+1} - u_{i+1,j} - u_{i-1,j}) = 1, i = 1, \cdots, M-1.$$

and these translate to system of equations $Au = f$, where A is $(M-2)^2 \times (M-2)^2$.

After that, we represent A in a curvelet domain. Curvelet representation of A is well-organized as well as ideally sparse. On discretizing a domain with 625×625 points, Fig. (5.4)(a) represent the contour plot of the solution. The closeness of contour lines indicates steepness. We have observed that, the contour lines which are equally distributed and closed together demonstrates a steep, uniform slope. The slope is more steep where the contour lines are close to one another. Fig. (5.4)(b) represents the problem's numerical solution. Fig. (5.4)(c) represents a graph between the error *i.e.*, $\|(u_i)^M - (u_{2i})^{2M}\|_2$ where $i = 1, 2, \cdots, M$ and number of grid points. Here error is calculated according to double mesh principle. The procedure of double mesh principle is to double the no. of mesh points and then error is calculated as above, where $(u_{2i})^{2M}$ is the solution obtained on a mesh containing the same mesh points which are used in the previous mesh. It is observed that error decreases on increasing the no. of grid points. Fig. (5.4)(d) represent the graph between $\|(u_i)^M - (u_{2i})^{2M}\|_2$ and threshold (chosen by the user). It shows that error increases as we increase threshold. Fig. (5.4)(e) compares the computational time carried out by the finite difference approach and FCFD method for estimating the solution at $t = 2^m \Delta t$ time. It is shown that CPU time taken by FCFD method is less than finite difference method. Table (5.2) shows the Θ values for distinct values of threshold. It shows the efficiency of our algorithm.

Test problem III

Now, we consider 2-D wave equation

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \ (x, y) \in (0, 1) \times (0, 1), \ t > 0, \tag{5.4.1}$$

with an initial

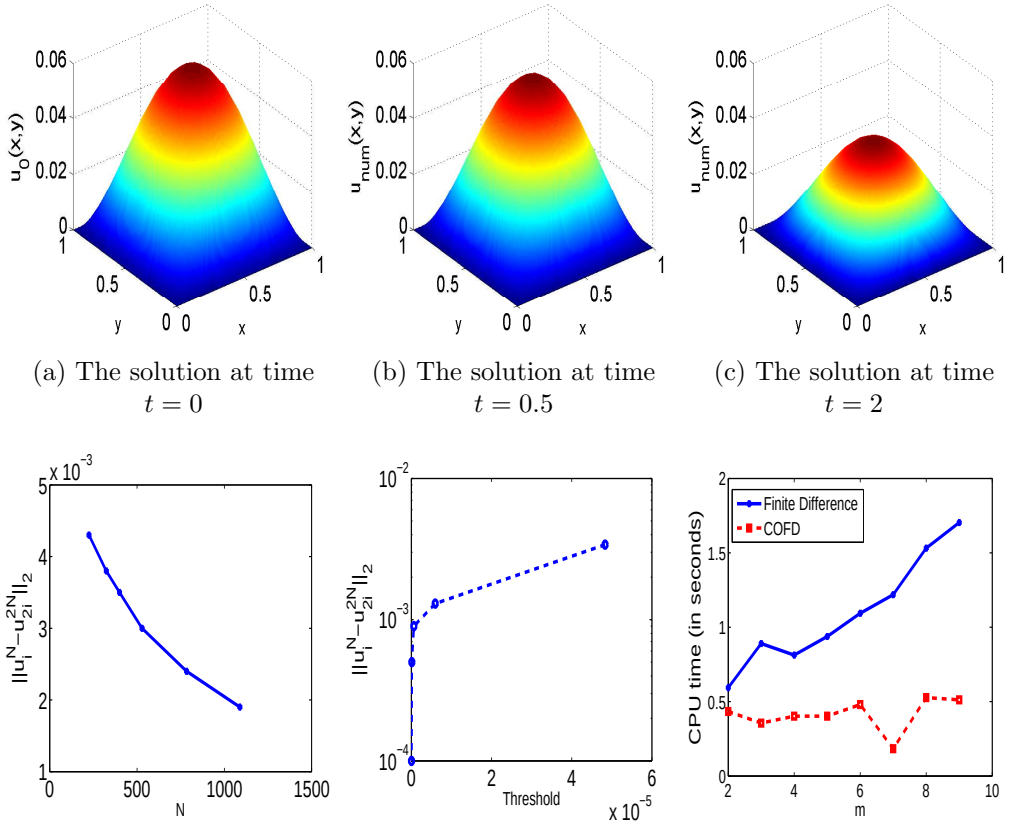
$$u(x, y, 0) = u_0(x, y) = x(x-1)y(y-1),$$

and

$$\frac{\partial w}{\partial t}(x, y, 0) = 0,$$

and boundaries $u(1, y, t) = u(0, y, t) = u(x, 1, t) = u(x, 0, t) = 0$. Fig. (5.5)(a), (5.5)(b), (5.5)(c) displays the problem's solution at distinct times. Fig. (5.5)(d) shows the error

$\|u_i^M - u_{2i}^{2M}\|_2$, where $i = 1, 2, \cdots, M$ versus no. of grid points at time $t = 0.99$. We have observed that error decreases on increasing the no. of grid points. Fig. (5.5)(e) represent a graph between $\|(u_i)^M - (u_{2i})^{2M}\|_2$ and threshold (chosen by the user). It shows that error increases on increasing threshold. Fig. (5.5)(f) compares the computational time carried out by the finite difference approach and FCFD method for estimating the solution at $t = 2^m \Delta t$ time. It shows that CPU time taken by FCFD is less than finite difference approach. The efficiency of this proposed method is shown by the table (5.3) which displays that Θ decreases on decreasing the threshold. The higher the Θ value, more efficient the adaptive algorithm is.



(a) The solution at time $t = 0$

(b) The solution at time $t = 0.5$

(c) The solution at time $t = 2$

(d) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N

(e) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold

(f) CPU time for computing the solution at the time $t = 2^m \Delta t$

Figure 5.5: Results for the test problem III.

Table 5.3: The performance of FCFD for the test problem III.

Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.4853	0.5478	0.5573
Θ	1.197	1.061	1.052

Test problem IV

For the 4th test problem, we consider the 3-D diffusion eqn. given as

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}; 0 \leq x, y, z \leq 1.$$

with an initial

$$u(x, y, z, t = 0) = u_0(x, y, z) = 100 \sin(\pi x) \sin(\pi y) \sin(\pi z),$$

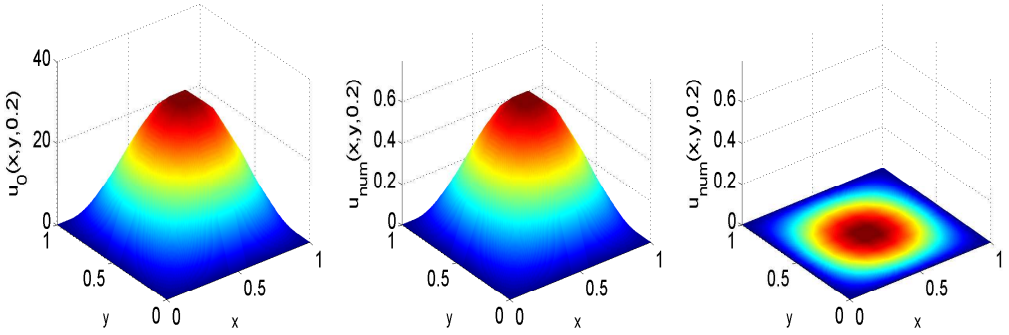
and boundaries

$$u(x = 1, 0, y, z, t) = u(x, y = 1, 0, z, t) = u(x, y, z = 1, 0, t) = 0.$$

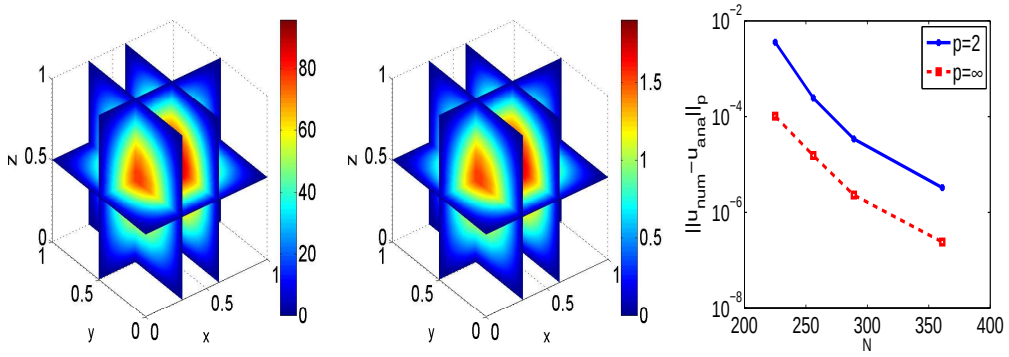
Fig. (5.6)(a), (5.6)(b), (5.6)(c) displays the problem's numerical solution at $z = 0.2$ for distinct times. Fig. (5.6)(d), (5.6)(e) displays the problem's numerical solution for the segments built at $x = \{0.3, 0.6\}$, $z = 0.5$ and $y = 0.5$ at distinct times. The solution has been evaluated at time $t = 4.213$ and the solution diffuses as anticipated. Fig. (5.6)(f) displays the error ($i.e., \|u_{num} - u_{ana}\|_p$) versus no. of grid points. Fig. (5.6)(g) displays the graph of threshold versus $\|u_{num} - u_{ana}\|_p$. We have seen that error decreases on increasing the no. of grid points and increases on increasing the threshold. Fig. (5.6)(h) compares the computational time carried out by the finite-difference approach and FCFD method for estimating the solution at $t = 2^m \Delta t$ time. It shows that CPU time taken by FCFD is less than finite-difference technique.

Table 5.4: The performance of FCFD for the test problem IV.

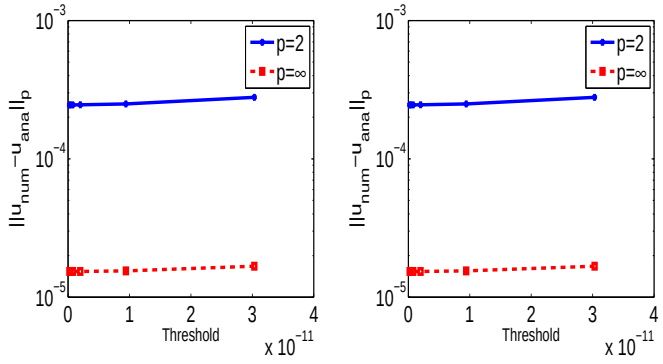
Threshold	10^{-3}	10^{-5}	10^{-6}
CPU time taken (in seconds)	39.9063	41.0938	41.5313
Θ	1.048	1.018	1.007



(a) The solution $u(x, y, 0.2)$ at time $t = 0$ (b) The solution $u(x, y, 0.2)$ at time $t = 0.1316$ (c) The solution $u(x, y, 0.2)$ at time $t = 4.213$



(d) The solution $u(x, y, z)$ at time $t = 0$ (e) The solution $u(x, y, z)$ at time $t = 0.1316$ (f) $\|u_{num} - u_{ana}\|_p$ versus N



(g) $\|u_{num} - u_{ana}\|_p$ versus threshold (h) CPU time for computing the solution at the time $t = 2^m \Delta t$

Figure 5.6: Results for the test problem IV.

Test problem V

Now we examine the prey-predator model of Lotka-Volterra in 2-D

$$u_t = c_{11} u_{xx} + c_{12_{yy}} + a_1 u - r_1 uv,$$

$$v_t = c_{21}v_{xx} + c_{22_{yy}} + a_2 v - r_2 uv,$$

here $v(x, y, t)$ symbolizes the predator-population and $u(x, y, t)$ symbolizes the prey-population density at position (x, y) and time t where $-1 \leq x, y \leq 1$ (denoted S). The $c_{k,j}$'s values are taken as $c_{12} = c_{11} = 0.1$ and $c_{22} = c_{21} = 0.01$ that indicates that prey diffuses faster across the domain than predator. The values of a_1, r_1, a_2, r_2 will be taken achieved from the ODE model for a Hare-Lynx Hudson-Bay model: $a_1 = 0.47$, $r_1 = 0.024$, $a_2 = 0.76$, $r_2 = 0.023$.

The following values have been assumed for the boundaries

$$\overrightarrow{n}.\nabla u = 0, \ \forall x \in \partial S,$$

$$\overrightarrow{n}.\nabla v = 0, \ \forall x \in \partial S.$$

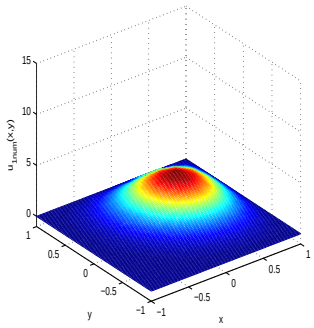
$$u(0, x, y) = \begin{cases} 10 & , \ (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \leq 16, \\ 0 & , \ otherwise. \end{cases} \tag{5.4.2}$$

$$v(0, x, y) = \begin{cases} 10 & , \ (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 \geq 4, \\ 0 & , \ otherwise. \end{cases} \tag{5.4.3}$$

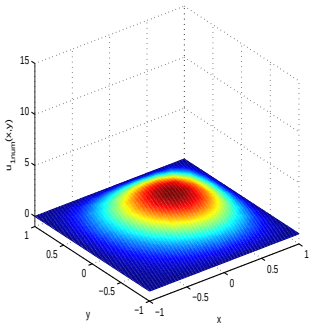
Fig. (5.7)(a), (5.7)(b) shows the numerical solution of prey population density at time $t = 1$, $t = 2$ respectively. Fig. (5.7)(c), (5.7)(d) shows the numerical solution of predator population density at time $t = 1$, $t = 2$ respectively. Fig. (5.7)(e) shows the error ($i.e., \|u_i^M - u_{2i}^{2M}\|_2$) versus no. of grid points. Fig. (5.7)(f) displays the graph of threshold versus $\|u_i^M - u_{2i}^{2M}\|_2$. It is shown that error decreases on increasing the no. of grid points and increases on increasing the threshold. Fig. (5.7)(g) compares the computational time carried out by the finite difference technique and FCFD method for estimating the solution at $t = 2^m \Delta t$ time. It shows that CPU time taken by FCFD is less than finite difference method. Table (5.5) gives the variation of CPU time taken with Θ. It is shown that as threshold increases, Θ increases. It demonstrates that the proposed technique is efficient.

Table 5.5: The performance of FCFD for the test problem V.

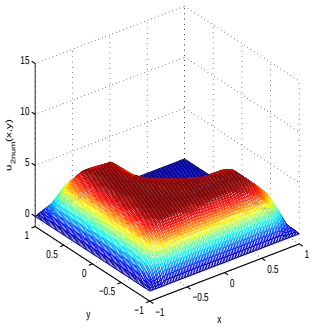
Threshold	10^{-2}	10^{-3}	10^{-4}
CPU time taken (in seconds)	0.3667	0.5379	0.6379
Θ	1.894	1.301	1.089



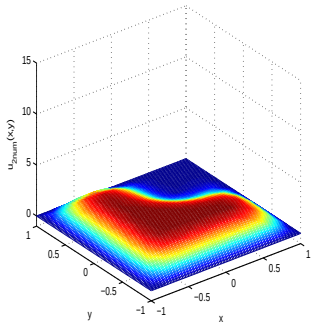
(a) The solution $u(x, y, t)$ at time $t = 1$



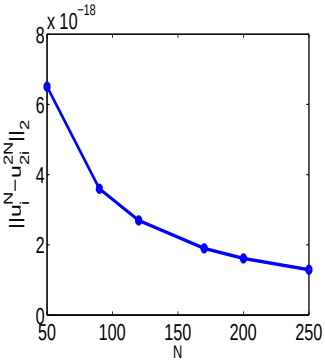
(b) The solution $u(x, y, t)$ at time $t = 2$



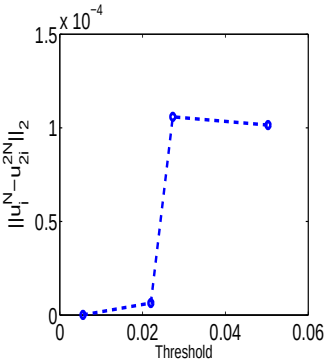
(c) The solution $v(x, y, t)$ at time $t = 1$



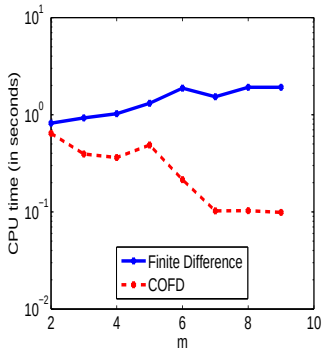
(d) The solution $v(x, y, t)$ at time $t = 2$



(e) $\|u_i^M - u_{2i}^{2M}\|_2$ versus N



(f) $\|u_i^M - u_{2i}^{2M}\|_2$ versus threshold



(g) CPU time for computing the solution at the time $t = 2^m \Delta t$

Figure 5.7: Results for the test problem V.

5.5 Conclusion and Future directions

A fast curvelet based FDM has been devised for finding the numerical solutions of PDEs. The differential operators are approximated using finite difference matrices and curvelets

are used for compressing these matrices. The method is applied on five test problems and it is found that the developed method is computationally very efficient. In future, the proposed method can be applied for solving PDEs on complex manifolds.

Chapter 6

Fast adaptive curvelet method

The numerical experimentation we performed reveal that the curvelet coefficients *i.e.*, c(j,l,k) too satisfy the property that they have high values near discontinuities like wavelets. This motivated us to use curvelets for making an adaptive grid.

This chapter proposes a dynamically adaptive curvelet technique for solving non-linear Schrödinger equation. Central finite difference approach is used for approximating the one and two dimensional differential operators and radial-basis functions (RBFs) are utilized for approximating the differential operators on the sphere. The grid on which the equation is solved, is obtained using curvelets. For $1d$ & $2d$ problems considered in this chapter, the computational time carried out by the proposed technique is compared with the computational time carried out by the finite difference technique. Moreover, the problem on the sphere has been considered for which, the computational time carried out by the RBF collocation technique is compared with the computational time carried out by the proposed technique. It is found that the developed technique performs better in terms of computational time, for example on sphere computational effort reduces by 4 times using proposed method.

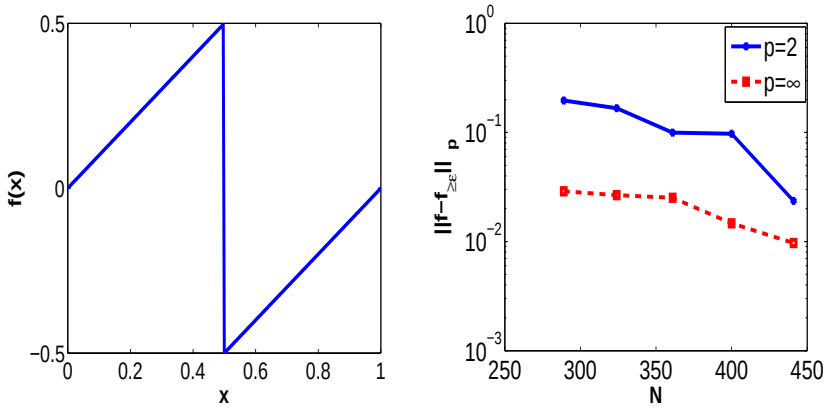


Figure 6.1: Compression error for Sawtooth-function which is discontinuous at $x = 0.5$.

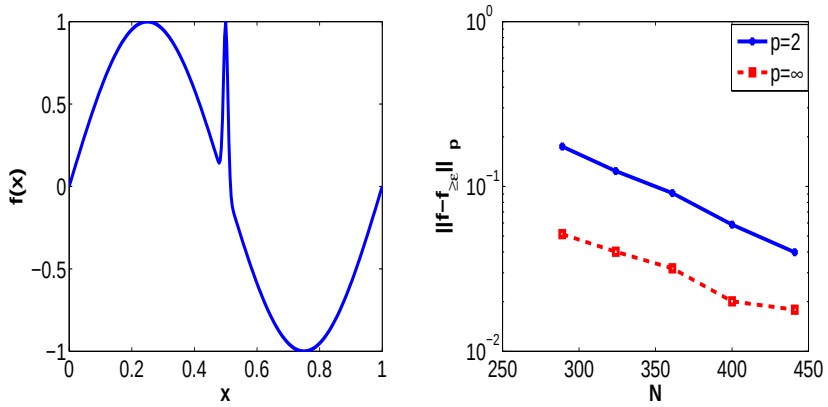


Figure 6.2: Compression error for $F(x) = \sin(2\pi x) + \exp(-10^4(x-0.5)^2)$.

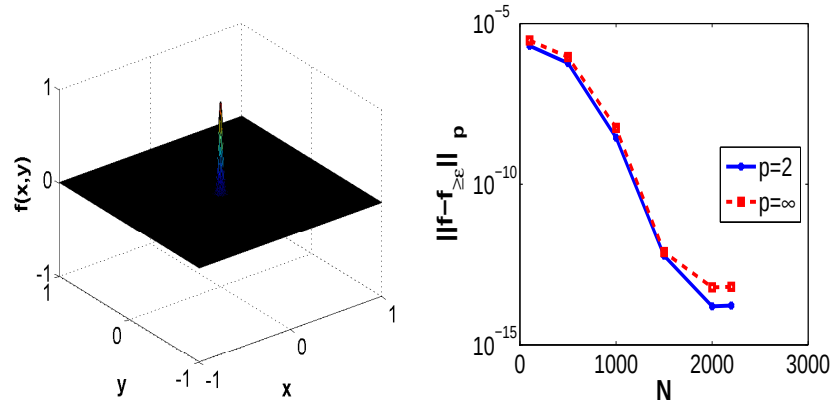


Figure 6.3: Compression error for $f(x,y) = \exp(-1000(x^2+y^2))$.

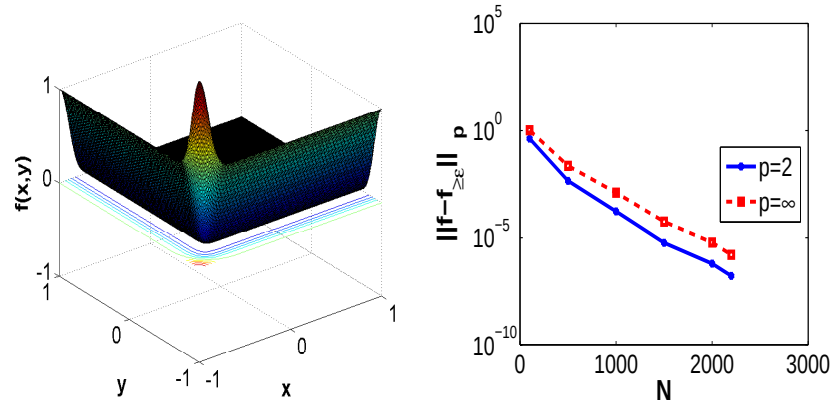


Figure 6.4: Compression error for $f(x,y) = \exp(-50(x+1)^2) + exp(-50(y+1)^2)$.

6.1 Fast adaptive curvelet method (FACM)

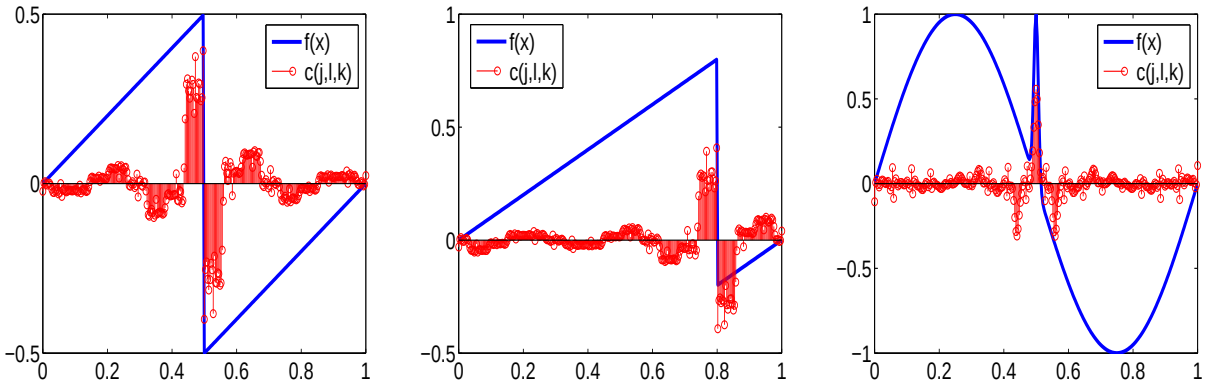
6.1.1 Curvelet based adaptivity

In [178] Donoho stated that for adequately continuous f, we have

$$\|f - f_{\geq\epsilon}\|_\infty < C\epsilon, \tag{6.1.1}$$

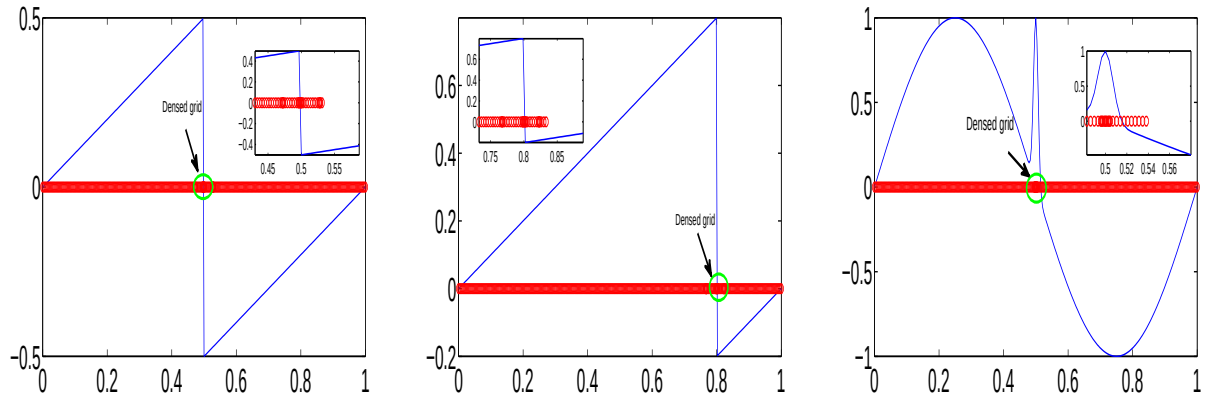
where C is a constant. Fig. (6.1) and (6.2) displays the variation of compression error verses N for two different one dimensional functions. Fig. (6.3) and (6.4) displays the variation of compression error verses N for two dimensional functions. Good compression from these graphs can be observed. Next, we explain the grid formulation process.
Suppose X^c is the current coarse grid, M is the length of X^c and $f(x_j)_{j\in X^c}$ is known which is a vector of length $M \times 1$. After that, decompose $f(x_j)_{j\in X^c}$ into matrix $g(x_j)_{j\in X^c}$ of length $[\sqrt{M}, \sqrt{M}]$. Apply FDCT on this function $g(x)$ to obtain the curvelet coefficients. Extract the curvelet coefficients matrix of finest scale. All points associated with $c(j, l, k)$ curvelet coefficients with a magnitude greater than ϵ are kept intact during the procedure of generation of the new finer grid from the grid. In case of two dimensional, apply FDCT on the given function which is a matrix of length $M \times M$ and repeat the same procedure. In this chapter, modified adaptation technique has been used for adapting the grid.
It is important to note that while solving a PDE, we will not adapt the node arrangement at each time step. In order to assure accuracy, the points of the grid corresponds to the curvelets that may become significant during the time when the grid does not change should also be added. Therefore, if x_l is the active node point in the coarser grid, then x_l as well as x_m such that $|m-l| \leq L$ (for any fixed non-negative number L) are added to make the finer grid. While solving a PDE on the sphere, the geodes $(x_m, x_l) \leq R$ (for any fixed positive integer R, the geodesic distance between x_l and x_m points) are incorporated in X^c, which is the new coarsest grid. Fig. (6.5) gives curvelet coefficients $c(j, l, k)$ for three different functions. For $\epsilon = 0.08$, Fig. (6.6) demonstrates the grid of adaptation generated by using the modified adaptation approach for several one dimensional functions. Function and the corresponding adaptive grid, when x varies from 0.48 to 0.58 has been shown as a subpart of the Fig. (6.6)(a). Similarly, we have done this for Fig. (6.6)(b) and Fig. (6.6)(c), when x varies from 0.43 to 0.59 and 0.73 to 0.89 respectively. Fig. (6.7) and Fig. (6.8) shows the modified adaptive grid for different two dimensional functions. After that we have considered three dimensional test functions one is $f(x) = \frac{4}{5\nu}\exp^{-((a-a_0)^2+(b-b_0)^2+(c-c_0)^2)/5\nu}$ where $a_0 = 1, b_0 = 0, c_0 = 0$ and $\nu = \frac{1}{2\pi^2}$ and second is $f(x) = 1 + \exp^{\frac{a+b+c+1}{2\xi}}$ where $\xi = 0.01$. These functions, their curvelet coefficients of finest scale and their corresponding

reconstructed functions for $\epsilon = 10^{-4}$ are respectively plotted in Fig. (6.9) and Fig. (6.10). Fig. (6.11) displays the two distinct functions on the sphere and their related adaptive grid.



(a) (b) (c)

Figure 6.5: $c(j,l,k)$ of finest scale for different functions (a) Sawtooth function which is discontinuous at $x = 0.5$ (b) Sawtooth function which is discontinuous at $x = 0.8$ (c) $F(x) = \sin(2\pi x) + \exp(-10^4(x-0.5)^2)$.



(a) (b) (c)

Figure 6.6: The modified curvelet adaptive grid for (a) Sawtooth function which is discontinuous at $x = 0.5$ (b) Sawtooth function which is discontinuous at $x = 0.8$ (c) $F(x) = \sin(2\pi x) + \exp(-10^4(x-0.5)^2)$.

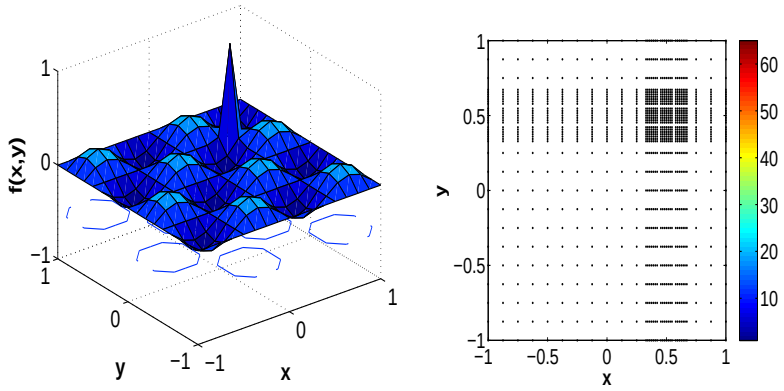


Figure 6.7: Adaptive grid for $f(x,y) = \exp(-200((x-0.5)^2+(y-0.5)^2)) - 0.2\sin(2\pi x)\sin(2\pi y)$.

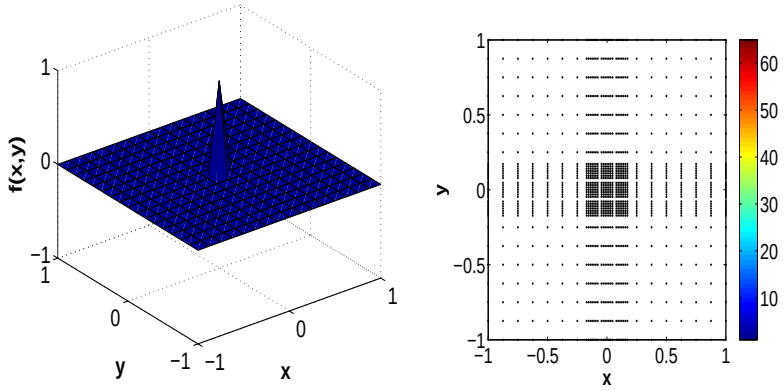


Figure 6.8: Adaptive grid for $f(x,y) = \exp(-1000(x^2+y^2))$.

6.1.2 FACM for non-linear PDEs

In this section, we explain FACM for solving non linear PDEs. For explaining the method, we have considered the one-dimensional NLS equation.

$$\frac{\partial u}{\partial t} = \mathfrak{L}u + i\gamma|u|^2 u, \quad t > 0, x \in R, \tag{6.1.2}$$

where $\mathfrak{L} = \frac{-i}{2}\beta_2\frac{\partial^2}{\partial x^2} - \frac{\alpha}{2}$ with an initial $u(x,0) = u_0(x)$ along with relevant boundary equations.

Algebraic polynomial interpolation is the most familiar and popular way of generating differentiating coefficients. One directly applies the polynomial to the data, accompanied by the polynomial differentiation and at last one estimate the polynomial at the enthusiasm point. As, we are working with the points of grid from the curvelet, derivatives on a homogeneous or non-homogeneous grid are evaluated by means of Lagrangian polynomial

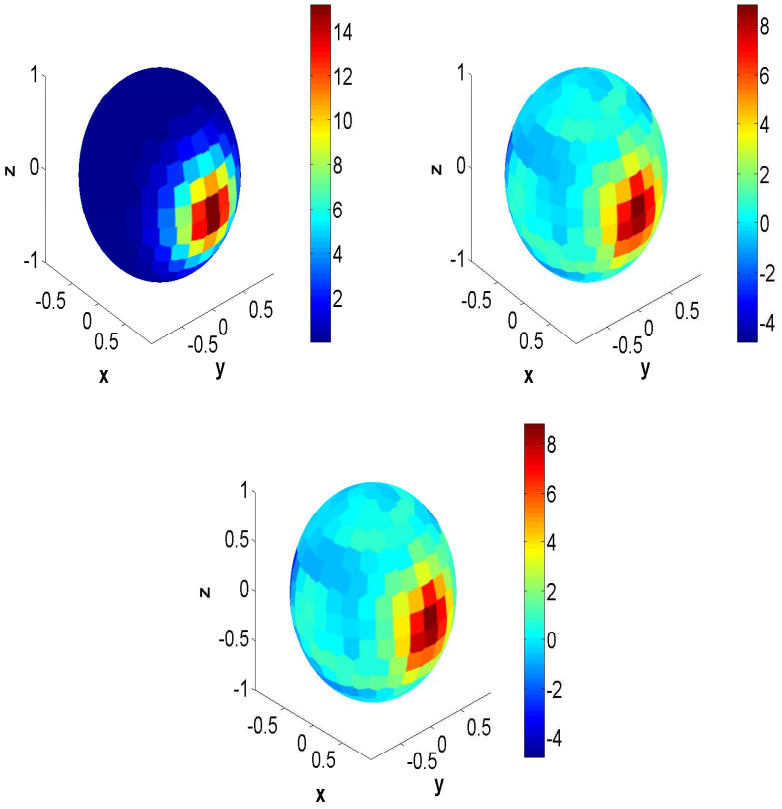


Figure 6.9: Function (a) $f(x) = \frac{4}{5\nu} e^{-((a-a_0)^2+(b-b_0)^2+(c-c_0)^2)/5\nu}$ (b) $c(j,l,k)$ of finest scale (c) $f_{\geq\epsilon}(x)$ for $\epsilon = 10^{-4}$.

interpolation through p points [186]. Let $w = \frac{p-1}{2}$ and describe,

$$u_I(x) = \sum_{k=i-w}^{i+w} u(x_k) \frac{P_{w,i,k}(x)}{P_{w,i,k}(x_k)}, \tag{6.1.3}$$

where $P_{w,i,k}(x) = \prod_{\substack{l=i-w \\ l\neq k}}^{i+w} (x - x_l)$. Eqn. (6.1.3) shows that u is interpolated by the u_I at the points of the grid, *i.e.*, $u_I(x) = u(x_i)$ for $i = 0, 1, 2, \cdots, N_{j-1}, N_j$. Differentiate $u_I(x)$ d times, it gives

$$u_I^{(d)}(x) = \sum_{k=i-w}^{i+w} u(x_k) \frac{P_{w,i,k}^{(d)}(x)}{P_{w,i,k}(x_k)}.$$

The derivatives $u_I^{(d)}(x)$ are evaluated at all points of the grid by $u_I^{(d)}(x) = \mathsf{D}_p^{(d)}(u)$, with differentiated matrix $[\mathsf{D}_p^{(d)}]_{i,k} = \frac{P_{w,i,k}^{(d)}(x_i)}{P_{w,i,k}(x_k)}$. With space discretization of Eqn. (6.1.2), we get $\frac{d}{dt}u(t) = Lu(t) + N(u(t))$ where $L = -\frac{i}{2}\beta_2 \mathsf{D}_p^2 - \frac{\alpha}{2} I$ and $N(u(t)) = i\gamma|u(x,t)|^2$. Now by

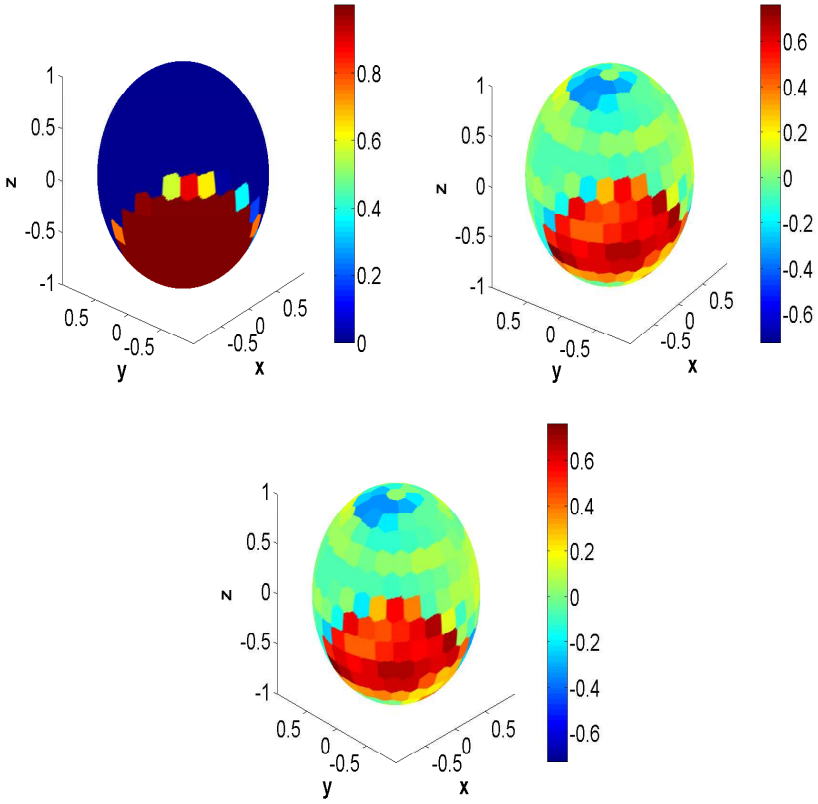


Figure 6.10: Function (a) $f(x) = 1 + e^{\frac{a+b+c+1}{2\xi}}$ (b) $c(j,l,k)$ of finest scale $f_{\geq\epsilon}(x)$ for $\epsilon = 10^{-4}$.

employing Crank-Nicolson approach for discretizing the time, we obtain

$$u^{n+1} = A^{-1}\left(Bu^n + N\Delta t\left(\frac{u^{n+1}+u^n}{2}\right)\frac{u^{n+1}+u^n}{2}\right). \tag{6.1.4}$$

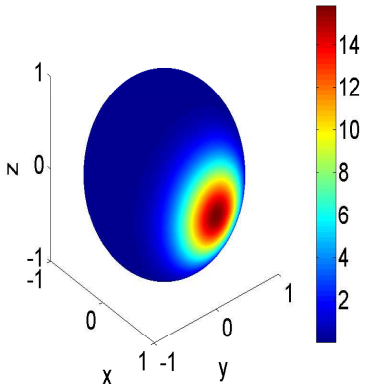
where $A = I - \frac{\Delta t}{2}L$, $B = I + \frac{\Delta t}{2}L$ and $u^{n+1} = u(n\Delta t)$.

6.1.3 Schrödinger equation on the Sphere

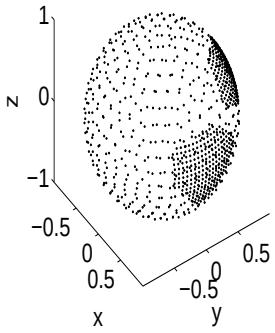
Here, we will talk about the spherical Schrödinger eqn. over the adapted node arrangement using the curvelet. The eqn. which is considered is as follows:

$$i\frac{\partial u}{\partial t}(p,t) = -\nabla^2 u(p,t) - C_1|u(p,t)|^2 u(p,t), \tag{6.1.5}$$

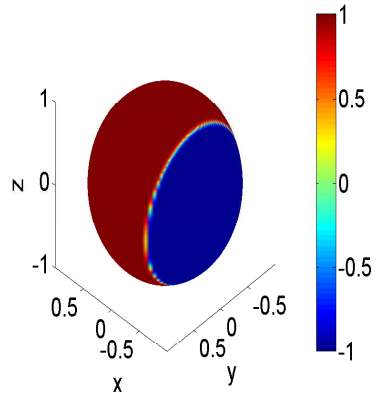
$p = \{(x,y,z) \in R^3 : x,y,z \in R \text{ and } \|p\| = a\}$, where a is the sphere's radius. With space discretization of Eqn. (6.1.5), we get $\frac{d}{dt}u(p,t) = L_1 u(p,t) + N_1 u(p,t)$ where $L_1 = i\nabla^2$ and $N_1 = iC_1|u(p,t)|^2$. Here ∇^2 is Laplacian operator. Now by using Crank-Nicolson technique



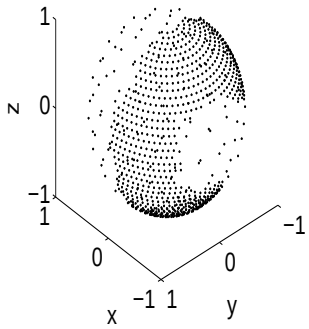
(a) $f(x) = \frac{4}{5\nu} e^{-((a-a_0)^2+(b-b_0)^2+(c-c_0)^2)/5\nu}$



(b) Adaptive Grid



(c) $f(x) = tanh\left(\frac{a+sin(\frac{\pi}{6})}{2\tau}\right),\ \tau = 10^{-2}$



(d) Adaptive Grid

Figure 6.11: Functions and their corresponding adaptive grid on the sphere for $R = 0.1$.

for discretizing the time, we obtain

$$u^{n+1} = A_1^{-1}\left(B_1 u^n + N_1 \Delta t\left(\frac{u^{n+1}+u^n}{2}\right)\left(\frac{u^{n+1}+u^n}{2}\right)\right).$$

where $A_1 = I - \frac{\Delta t}{2}L_1$, $B_1 = I + \frac{\Delta t}{2}L_1$ and $u^{n+1} = u(n\Delta t)$. Next we will explain the estimation of Laplace Beltrami operator by employing RBFs [53, 54].

6.1.4 Approximating the Laplace Beltrami operator by using RBFs

Suppose X be a manifold and $X^N = \{x_1, x_2, \cdots, x_N\}$ be discretized with N points. The RBFs are constructed from the bizonal kernel $\Phi : X \times X \to R$ of the type $\Phi(x, y) =$

$\varphi(x.y), x, y \in X$. Here φ is an univariate function represented on $[-1, 1]$ and $x.y$ is Euclidean dot product of the position vector of the points $x, y \in X$.

The value of $\Phi(x, y)$ relies solely on the geodesic distance from x to y, for a steady x value. Therefore the function $(x, .)$ is radially-symmetric function in regards to the point x and is known as RBF [187]. For every point $x_j \in X^N$, RBF is stated as

$$\Phi_j(x) = \Phi(x, x_j) = \varphi(x.x_j) = \Psi(|x - x_j|). \tag{6.1.6}$$

The Wendland's RBFs have been used in this work, which relies upon two parameters k and d, where $2k$ is the function's smoothness and d is the space dimension. The matrix A is positive definite for these RBFs and therefore reversible for each X^N. Wendland's RBFs are compactly supported and therefore the interpolated matrices are sparse and just few terms have to be examined for the computation of interpolants. This results in effective algorithm for the calculation and computation of the interpolants. The Wendland's basis functions are distinctive to a constant parameter and the degree of polynomial is minimum for the dimensions d and smoothness $2k$. As per as, continuity is concerned, we can choose k according to the differential system we want to solve. Moreover, Wendland's RBFs have advantages over other compactly supported RBFs. For example, the degree of polynomial of Wendland's function is lesser than that of Wu's function for prescribe smoothness. If Wu's functions as well as Wendland's functions are C^4 continuous, radial in R^3 and strictly positive definite then the degree of polynomial for Wendland's function is 8, however for Wu's function is 11 [50].
Given a function f which is smooth on X, then a sequence of numbers $\{\widetilde{f}^j\}_{j=1}^n$ exists which is unique, so that the function

$$I_{X^N} f(x) = \sum_{j=1}^{N} \widetilde{f}^j \Phi_j(X), \tag{6.1.7}$$

satisfies the interpolating condition

$$I_{X^N} f(x_k) = f(x_k),\ 1 \leq k \leq N.$$

Put $x = x_1, x_2, \cdots, x_N$ in Eqn. (6.1.7), we obtain

$$\begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_N) \end{bmatrix} = \begin{bmatrix} \Phi_1(x_1) & \Phi_2(x_1) & \cdots & \Phi_N(x_1) \\ \Phi_1(x_2) & \Phi_2(x_2) & \cdots & \Phi_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_1(x_N) & \Phi_2(x_N) & \cdots & \Phi_N(x_N) \end{bmatrix} \begin{bmatrix} \widetilde{f}^1 \\ \widetilde{f}^2 \\ \vdots \\ \widetilde{f}^N \end{bmatrix}$$

In matrix form, it can be composed as

$$f = A\widetilde{f}.$$

Since A is invertible, therefore

$$\widetilde{f} = A^{-1}f,$$

where $f = [f(x_1), f(x_2), \cdots, f(x_N)]'$ etc. Hence,

$$f(x) \approx I_{X^N} f(x) = \sum_{j=1}^{N} \widetilde{f^j}\Phi_j(X). \tag{6.1.8}$$

From Eqn. (6.1.8),

$$\nabla^2 f(x) = \sum_{j=1}^{N} \widetilde{f^j}\nabla^2\Phi_j(X) = \mathrm{B}\widetilde{f} = \mathrm{B}A^{-1}f, \mathrm{B} = [\nabla^2\Phi_j(x_i)]_{i,j=1}^{N}.$$

The Laplace operator ∇^2 approximates $\mathrm{B}A^{-1}$ on X.

6.1.5 Interpolating using RBFs

Assume, we take node points of two set X^f & X^c (X^c is the coarsest set and X^f is the finest set). Assume $\{f(x_j)\}_{x_j \in X_c}$ is known. By utilizing RBFs, we can evaluate $\{f(x_j)\}_{x_j \in X_f}$ as follows

$$\begin{aligned}
\{f(x_j)\}_{x_j \in X^c} &= [\Phi(x_i, x_j)]_{x_i, x_j \in X^c} \{\widetilde{f^j}\}^{j=1,2,\cdots,\sharp X^c}, \\
\{f(x_j)\}_{x_j \in X^f} &= [\Phi(x_i, x_j)]_{x_i \in X^f, x_j \in X^c} \{\widetilde{f^j}\}^{j=1,2,\cdots,\sharp X^c}, \\
\{f(x_j)\}_{x_j \in X^f} &= [\Phi(x_i, x_j)]_{x_i \in X^f, x_j \in X^c} [\Phi(x_i, x_j)]^{-1}_{x_i, x_j \in X^c} \{f(x_j)\}_{x_j \in X^c}.
\end{aligned}$$

6.1.6 Numerical Algorithm for solving NLS equation

The algorithm for solving Schrödinger equation is as per the following:

1. Discretize the Schrödinger equation's domain.
2. By using finite difference matrices, discretize the operators engaged with the Schrödinger equation (in case of one-dimensional and two-dimensional) at hand. Wendland RBFs

are used for discretization of operators involved in the Schrödinger equation on sphere.

3. Assume the times $t_1, t_2, \cdots$ are chosen to refine the grid (Noted that the selection of these times depends on the problem. If the solution of the problem changes rapidly, then t_is should be selected more closely). Initiate with an initial condition and by using the time- integration method (we have chosen Crank-Nicolson scheme), we get the solution at the time t_1, *i.e.*, $u(t_1)$.

4. Using the procedure explained in the last section, we use $u(t_1)$ and the related grid X^{t_1} to achieve $X^{t_1+\Delta t}$.

5. Compute the Schrödinger's equation differential operators on $X^{t_1+\Delta t}$.

6. Integrate with Crank-Nicolson, the obtained system of ordinary differential eqns. in time to find the solution at $t = t_1 + \Delta t$.

7. Again follow the step 6 until $u(t_2)$ is acquired. Go to step 4, after obtained $u(t_2)$.

For easy understanding flow chart is also given in Fig. (6.12).

Table 6.1: The performance of FACM for the test problem I.

Threshold (ϵ)	N($\epsilon = 0$)=289 N(ϵ) (When we take ϵ into account)	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
0.3	361	0.4705	0.1875	2.5096
0.03	1089	6.608	3.0781	2.147
0.003	2304	9.4022	5.1881	1.8123
0.0003	2500	11.878	8.9432	1.3282

6.2 Numerical outcomes and discussions

Here we examine the numerical outcomes attained with three test problems by using FACM.

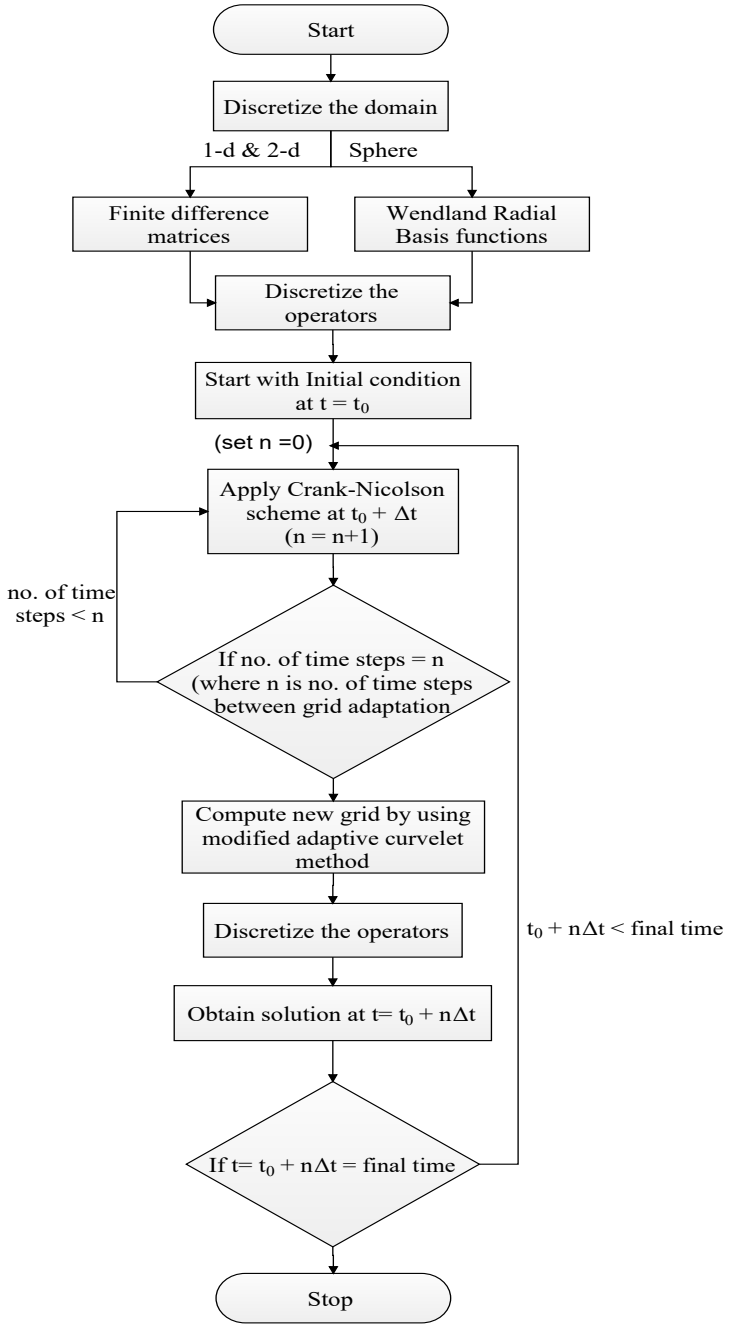


Figure 6.12: Flow chart for solving NLS equation.

Test problem I

For solving the 1-D Schrödinger equation Eqn. (6.1.2), an initial condition $u(x,0) = 2sech(x)$, $u \in [\frac{-L}{2}, \frac{L}{2}]$ where $L = 64$ and boundaries are periodic has been considered. The parameters of Eqn. (6.1.2) are $\beta_2 = -2$, $\gamma = 2$ and $\alpha = 0$. FACM has been used to solve this PDE. Fig. (6.13) displays the solution as well as its related adaptive grid at distinct times. In Fig. (6.14) the point wise error (PWE) is shown at distinct time and can be seen

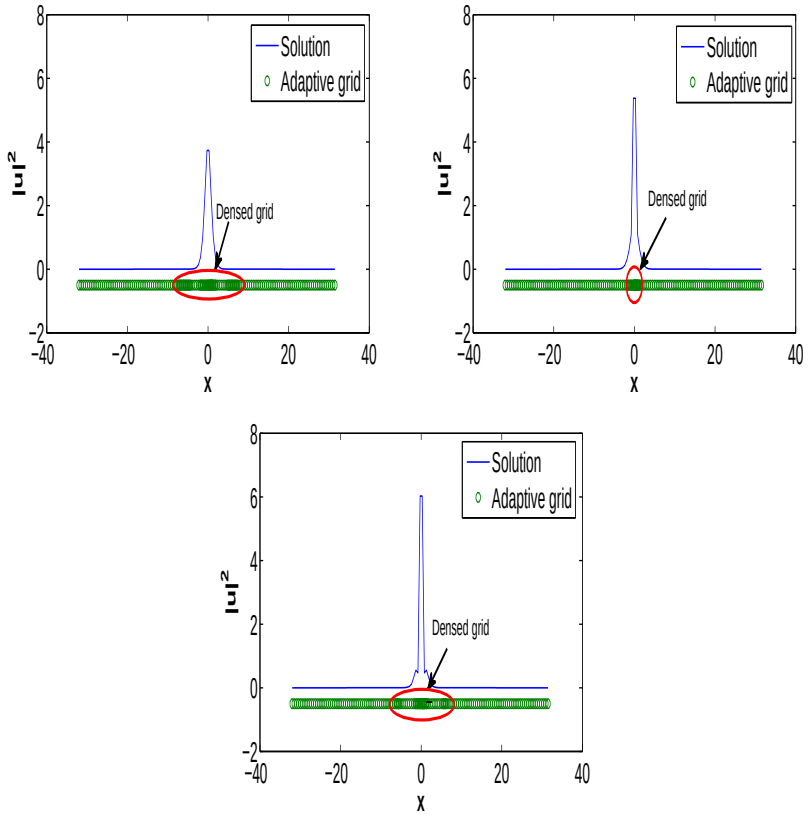


Figure 6.13: Solution and its correspondingly adapted grid at $t = 0, 0.25, 0.5$.

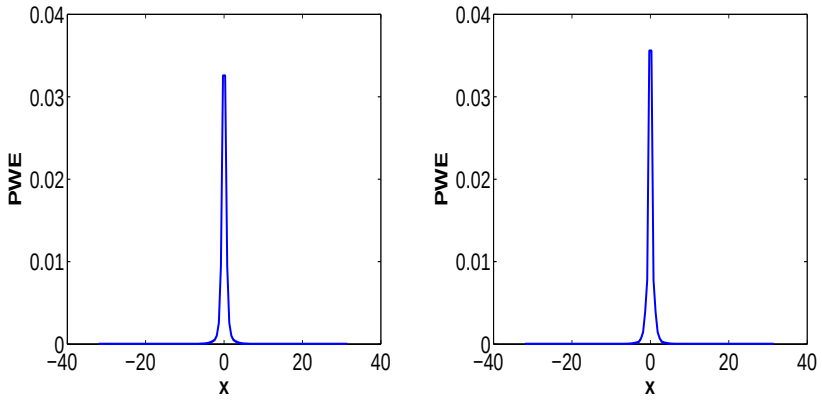
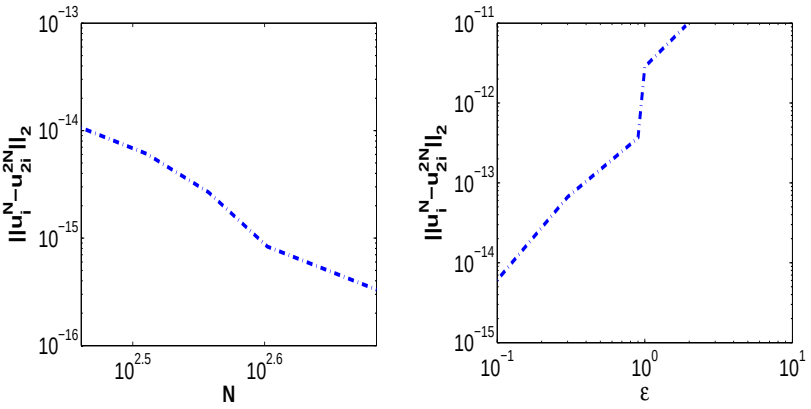


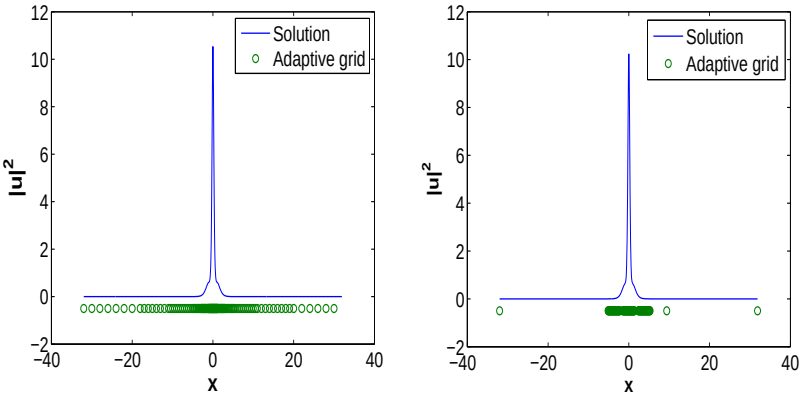
Figure 6.14: Point wise error (PWE) at $t = 0.25, 0.5$.

that the error value is highest close to the solution variation as anticipated. Subsequently extra points will be included in this region and automatic adaptive grid generation will be achieved. Fig. (6.15)(a) shows the error $\|u_i^N - u_{2i}^{2N}\|_2$, where $i = 1, 2, \cdots, N$ versus no. of grid points. Here error is calculated according to double mesh principle. The procedure of double mesh principle is to double the no. of mesh points and then error is calculated as above, where $(u_{2i})^{2N}$ is the obtained solution on a mesh containing the points of the same mesh which are used in the previous mesh. We have observed that



(a) (b)

Figure 6.15: Variation of error $\|u_i^N - u_{2i}^{2N}\|_2$ versus (a) N (b) ϵ.



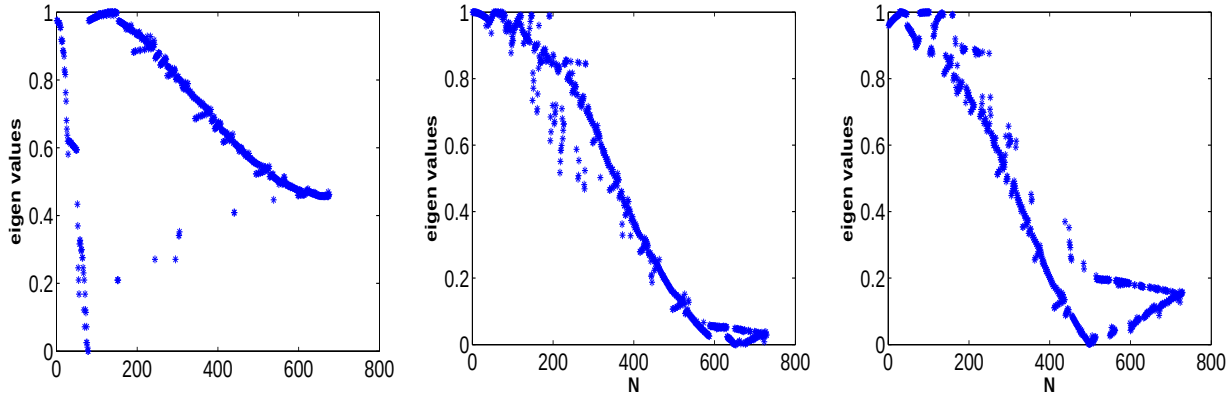
(a) (b)

Figure 6.16: Solution and its corresponding standard adaptation grid (a) Nielsen method result (b) Proposed method result.

Table 6.2: Comparison of CPU time taken between Nielsen method and proposed method.

	CPU time taken by Nielsen method	CPU time taken by proposed method
When $\epsilon = 0$	3.5975	3.5975
When we take ϵ into account	2.309	1.623

error decreases on increasing the no. of grid points. Fig. (6.15)(b) represent the graph between $\|(u_i)^N - (u_{2i})^{2N}\|_2$ and threshold (chosen by the user). It is seen that as threshold increases, error increases. Table (6.1) shows the variation of CPU with threshold. Now,



(a) (b) (c)

Figure 6.17: Absolute eigen values of the differentiation matrix at (a) time t=0.5 (b) time t=1.5 (c) time t=2.

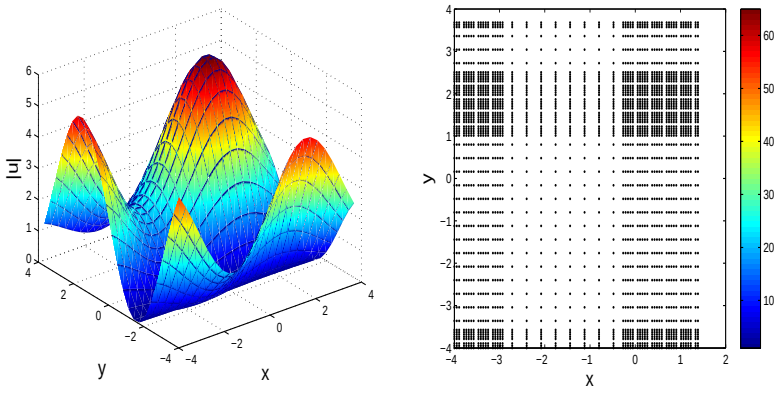
we describe the time compression coefficient as $\Theta = \frac{CPU(threshold=0)}{CPU(threshold)}$. It has been noticed that Θ decreases on decreasing the threshold value. The higher the Θ value, more efficient the adapted method is.

Validation of our results:- In order to validate our technique, our outcomes have been compared with Nielsen's results [10]. Nielsen has solved this problem using Daubechies wavelet and we have obtained the results using curvelet. For validation, grid is obtained with the help of standard adaptation technique. Our outcomes are seen to be in good agreement with Nielsen's outcomes as far as solution is concerned which is shown in Fig. (6.16). But the grid obtained with the help of curvelet is better than wavelet as compression in this case is less which was expected. Because curvelet is designed for this purpose. Moreover, CPU time taken by the Nielsen's approach (using the matlab code `nlswofd.m`) has been compared with the proposed method. It is found that the developed approach takes lesser CPU time than the Nielsen method which is shown in the table (6.2).

Numerical stability of the proposed technique:- The proposed technique can produce essentially a completely arbitrary grid and hence the differentiation matrix can assume an unlimited number of forms. For this reason, an analytical approach to check numerical stability is not possible. Therefore, the numerical stability of the method is ensured in terms of the eigen values of the differentiation matrix, for details [90] can be referred. If all the eigen values of the matrix have magnitude less than 1, then the convergence is achieved [188]. For the proposed method, the matrix whose eigen values are to be checked is

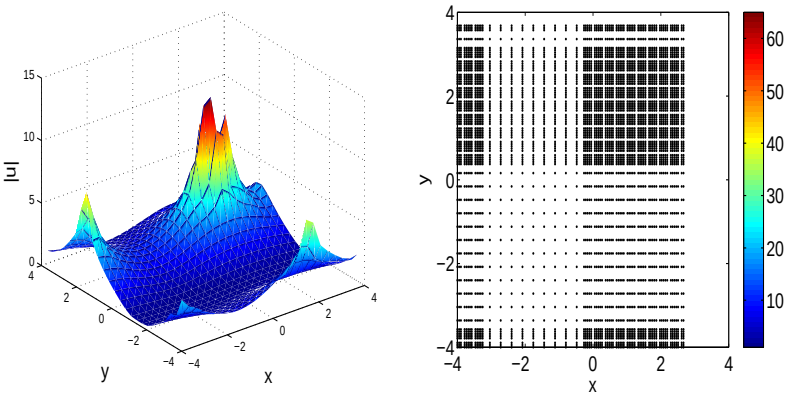
$$[I - \frac{\mathsf{D}^{(2)}}{2}\Delta t]^{-1}[I + \frac{\mathsf{D}^{(2)}}{2}\Delta t], \tag{6.2.1}$$

which corresponds to FACM applied to the linear equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$ with Crank-Nicolson time discretization. The grid on which the computations are done is the one obtained from FACM applied to the non-linear Schrödinger equation. We have checked the magnitudes of the eigen values of the differentiation matrices at time $t = 0.5$, $t = 1.5$ and $t = 2$ and these values are shown in the Fig. (6.17). It is seen that the magnitude of eigen values do not exceed 1 which ensures the numerical stability of our method.



(a) (b)

Figure 6.18: (a) $u(x, t = 0)$ for 2D Schrödinger equation and (b) its adaptive grid.



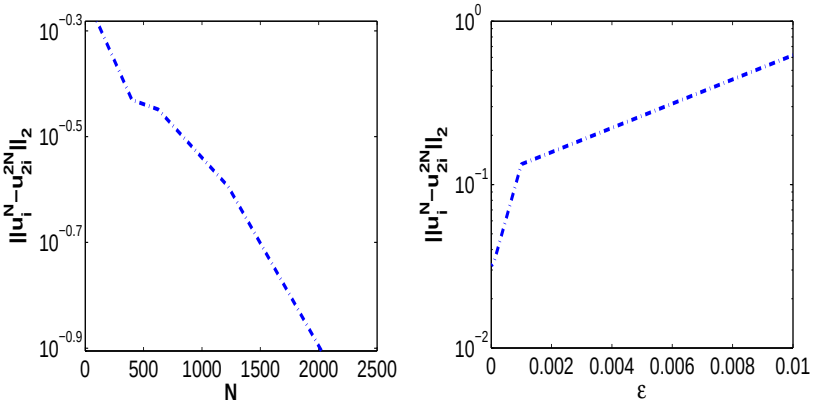
(a) (b)

Figure 6.19: (a) $u(x, y, t = 0.11)$ for test problem II and (b) its adaptive grid.

Test problem II

For the 2nd test problem, the 2-D Schrödinger [189] equation has been considered which is defined as

$$i\frac{\partial u(x,y)}{\partial t} + \nabla^2 u(x,y,t) + C_1|u(x,y,t)|^2 u(x,y,t) = 0, \tag{6.2.2}$$



(a) (b)

Figure 6.20: Variation of error E_2 with respect to (a) N (b) ϵ.

Table 6.3: The performance of FACM for the test problem II.

Threshold (ϵ)	N($\epsilon = 0$)=40 N(ϵ) (When we take ϵ into account)	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
0.9	128	7.989	1.987	4.02
0.2	164	8.905	2.63	3.385
0.09	196	11.68	3.92	2.98
0.009	228	13.03	5.975	2.182

here ∇^2 is the Laplacian operator. To solve this Eqn. (6.2.2), we choose an initial condition $u(x, y, t = 0) = (1 + sin(y))(1 + sin(x))$ and $C_1 = 2$ on a square domain $\left[-\frac{L}{2}, \frac{L}{2}\right] \times \left[-\frac{L}{2}, \frac{L}{2}\right]$ where $L = 8$. In Fig. (6.18), an initial condition and its correspondingly adapted grid have been plotted. The equation is resolved until time $t = 0.11$ and the solution and its related adapted grid are shown in Fig. (6.19). We have found from Fig. (6.18) and Fig. (6.19), that near the singularity, the grid becomes denser. Fig. (6.20)(a) represent the graph between the error *i.e.*, $\|(u_{i,j})^{N,M} - (u_{2i,2j})^{2N,2M}\|_2$ here $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, M$. It can be observed from the graph that as the no. of grid points increases, error decreases. Fig. (6.20)(b) represent the graph between E_2 (we call $\|(u_{i,j})^{N,M} - (u_{2i,2j})^{2N,2M}\|_2$ as E_2) and threshold (chosen by the user). It shows that error increases as we increase threshold. Table (6.3) displays the values of Θ for distinct values of threshold. It has been seen that as Θ decreases on decreasing the value of ϵ.

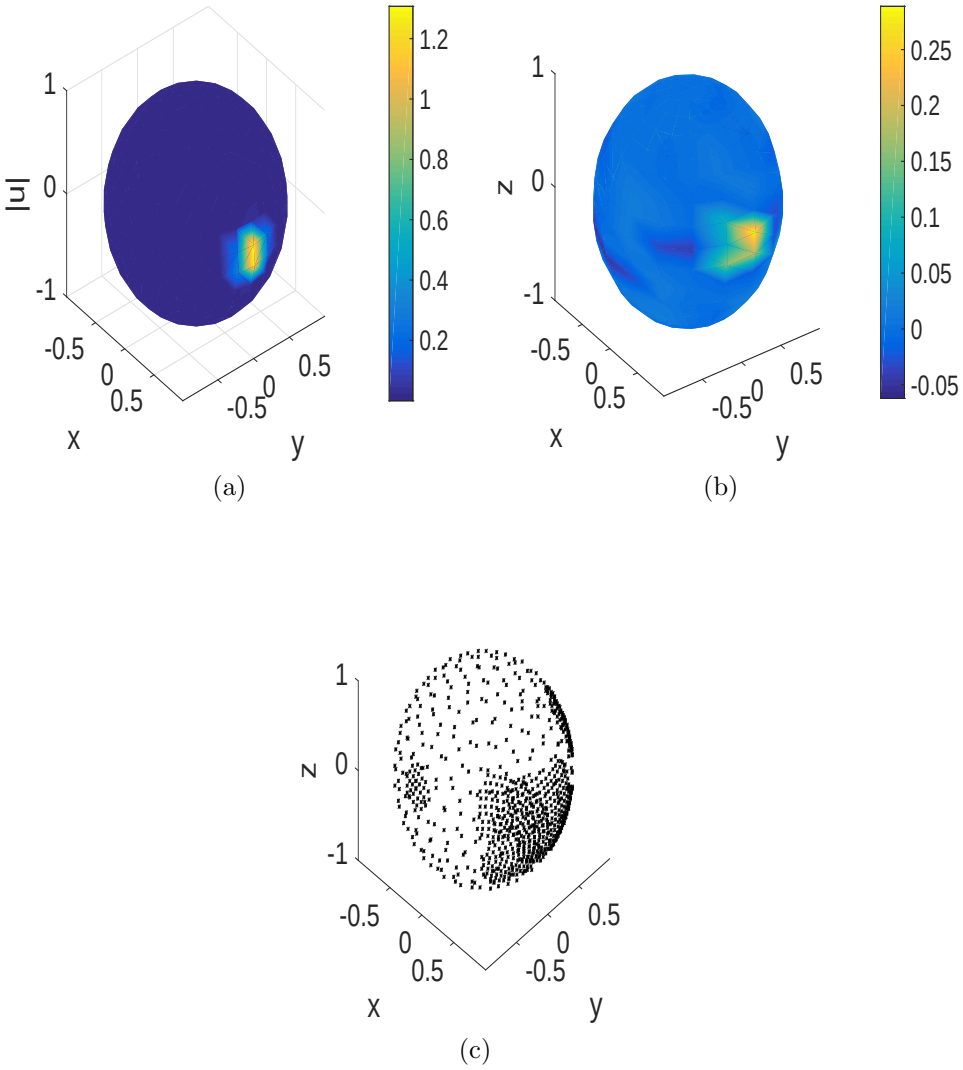


Figure 6.21: (a) The initial test function Eqn. (6.2.3) with $\theta_0 = 0$, (b) $\phi_0 = 0$ and $L = 1/2\pi$, coefficients plot at finest scale and (c) its adaptive grid.

Test problem III

We take the initial condition for testing the NLS equation on the sphere, which is an elementary Gaussian complex function on the sphere represented as:

$$u(\theta, \phi, t = 0) = 2 \exp\left(- i \frac{(\theta - \theta_0) + (\phi - \phi_0)}{L^2 * (t + 1)} \right) \exp\left(- \frac{(\theta - \theta_0)^2 + (\phi - \phi_0)^2}{L^2 * (t + 1)} \right), \quad (6.2.3)$$

where $\phi(-\frac{\pi}{2} \leq \phi \leq \frac{\pi}{2})$ and $\theta(-\pi \leq \theta \leq \pi)$ are the latitude and longitude respectively and the parameters involved in the Eqn. (6.2.3) are $\theta_0 = 0$, $\phi_0 = 0$, $L = 1/2\pi$ and $C_1 = 1$. In Fig. (6.21)(a), we have plotted initial condition, Fig. (6.21)(b) its curvelet coefficients

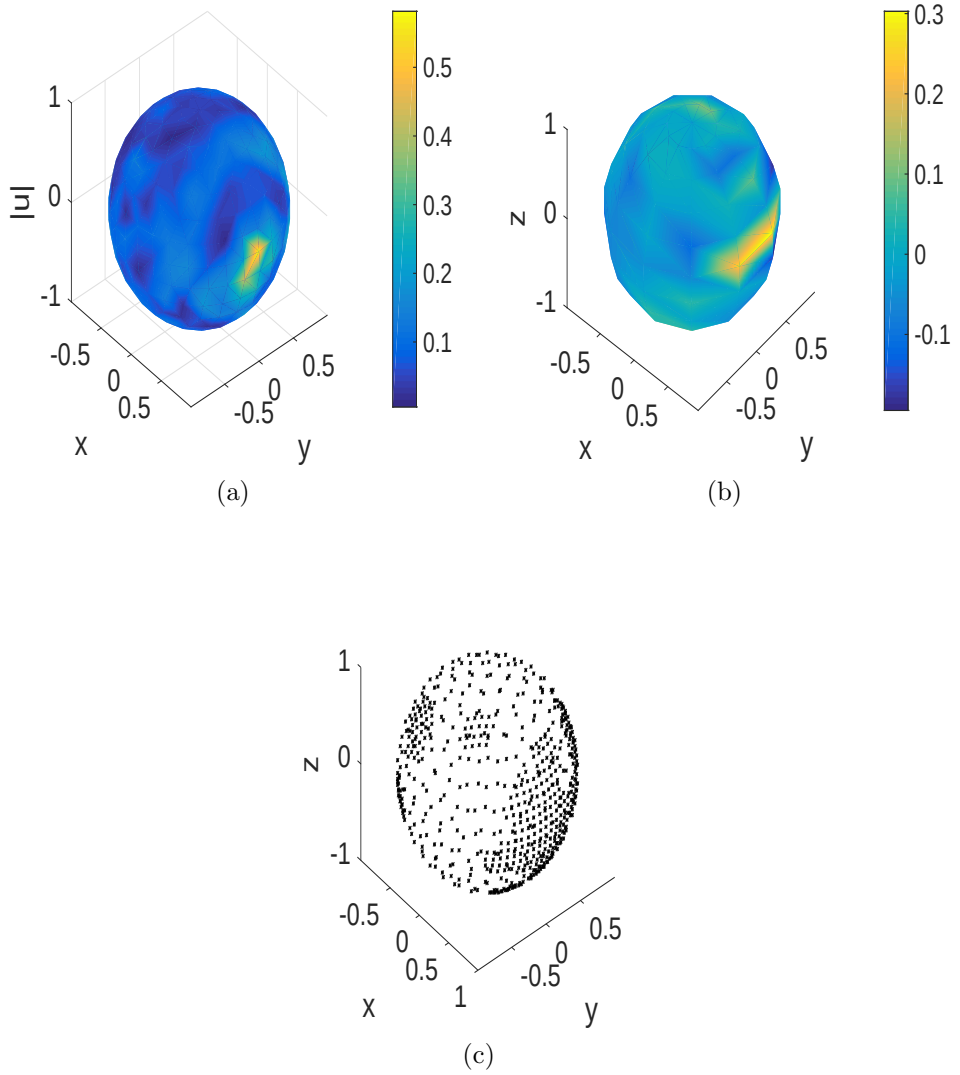


Figure 6.22: (a) The solution of Eqn. (6.1.5), at $t = 0.15$ (b) $\phi_0 = 0$ and $L = 1/2\pi$, coefficients plot at finest scale and (c) its adaptive grid.

at finest scale and Fig. (6.21)(c) its corresponding adaptive grid. In Fig. (6.22)(a), we have plotted its solution at time $t = 0.15$, Fig. (6.22)(b) curvelet coefficients at finest scale and Fig. (6.22)(c) its corresponding adaptive grid. Fig. (6.23) represent the graph between the error *i.e.*, $\|u(p) - u_{\geq}(p)\|_2$ where $i = 1, 2, \cdots, N$ and no. of grid points as we have taken RBF interpolation. We have observed from the graph that as the no. of grid points increases, error decreases. Table (6.4) displays the Θ values for distinct values of threshold. It has been viewed that Θ decreases on decreasing the value of ϵ. The higher the Θ value, more efficient the adaptive method is. There are a variety of radial basis functions available in the literature and Gaussian RBF ($e^{-E^2\|x-x_i\|^2}$) with E as the shape parameter is one of the widely used RBF but it is globally supported [190]. To check that

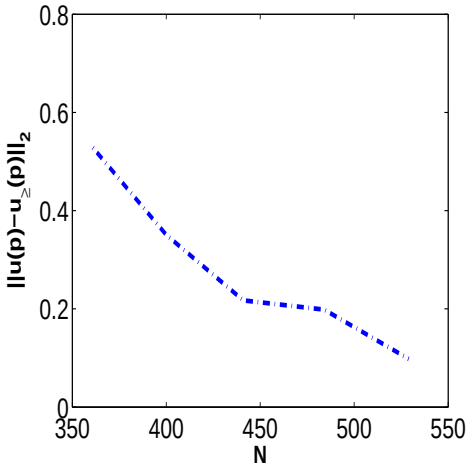


Figure 6.23: Variation of error $\|u(p) - u_{\geq}(p)\|_2$ versus N .

Table 6.4: The performance of FACM for the test problem III.

Threshold (ϵ)	N($\epsilon = 0$)=289 N(ϵ) (When we take ϵ into account)	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
0.07	576	1.622	0.4375	3.7077
0.03	784	2.873	1.2968	2.216
0.009	841	3.276	1.6496	1.986
0.0009	900	3.837	2.9844	1.286

Table 6.5: CPU time (in seconds) comparison for solving NLS equation on sphere.

	Approximation of Laplace operator is done by Wendland's RBFs ($\epsilon = 0$)	Approximation of Laplace operator is done by Gaussian RBFs ($\epsilon = 0$)
When $t = 0.15$	2.297624	3.320603
When $t = 0.5$	3.299502	5.252680

the Wendland's RBFs give us computationally efficient scheme over Gaussian RBFs, we have compared and presented the CPU time results in table (6.5). From the table it can be observed that the CPU time carried out by Wendland's RBFs is lesser in comparison to the Gaussian RBFs.

6.3 Conclusion and Future directions

In this chapter, it is shown that how a dynamic adaptive curvelet approach works very smoothly with the local singularity of the Schrödinger equation's solution. The differential operators are approximated using finite difference in case of one and two dimension and RBFs are used for the approximating the differential operators on sphere. Curvelet is used for generating the grid. Crank-Nicolson approach is finally utilized for time integration. The outcomes show that the curvelet and computational grid can very proficiently adapt to the solution's local irregularities so as to resolve sharp transition regions. Comparison of CPU time shows that the developed technique is extremely efficient. To our best knowledge, the significant properties of the curvelet are used to solve PDEs on an adaptive grid for the first time. Having checked the performance of the proposed technique for Schrödinger equation, we proffer the use of adaptive curvelet approach to model turbulence and solve fluid dynamics problems.

Chapter 7

Fast adaptive curvelet method for solving PDEs on general manifolds

Numerous applications of the applied and natural sciences demands the PDE's solutions on general manifolds. For example the problem of the distribution of surfactants along the fluid moving interface, in image processing [191–193], mathematical physics [194], biological systems [195, 196], fluid dynamics [197–199], computer graphics for texture synthesis [193], vector field visualization [200] and weather prediction [192] etc.

The numerical solution of these equations and treatment of surface differential operators is an area of active research. To solve PDEs on general manifolds, we have different approaches available in literature. For the manifolds which can be easily parameterized the differential operators are expressed in the parameter space and the emerging equations are then discretized. For methods of parametrization, [201] is referred. As it is well known that parametrization of very complex manifolds is hard to obtain, people have shifted to work on manifolds which can be easily triangulated but again for the manifolds with dimension three or more obtaining triangulation is a difficult task [202, 203]. To get rid of the problems of parametrization and triangulation, methods based on implicit representations are being used. In these methods the manifold along with the PDEs characterized on it are embedded in the ambient space and the extensive PDEs are then discretized using cartesian grid [203–205]. Instead of using the implicit representations, closest point representations can also be used [206]. General manifolds are embedded in a higher dimensional Cartesian grid using a closest point function method. The technique is to discretize the PDEs by utilizing a static cartesian grid into the embedding space which is analogous to another embedding approaches. This results in an effective method to solve PDEs on surfaces with great simplicity and additional desirable features. In particular, it is apparent that the surface PDEs are the analogue of the embedding PDEs and the standard cartesian differential operators are involved. Furthermore, the technique deals with the arbitrary manifolds and is not restricted to one co-dimensional surfaces. It additionally enables the calculation to be carried out on a grid described in a confined band close to the object with no deteriorating the efficiency and without enforcing artificial boundaries. The method is advantageous as it can handle manifold of any dimensions or co-dimensions.

The surface normal and curvature are fundamental geometric properties of the surface, and are of interest for a variety of purposes. In the context of solving surface PDEs, our main concern is that such quantities could occur explicitly within the PDE, for example, in the form of a curvature-dependent reaction rate in a reaction diffusion equation. If these geometric quantities are available as a given function on the surface, our general formalism would immediately apply. However, it is more likely that they will actually need to be computed from the surface itself, prior to any extension. Thus we need to look for some another convenient way to deal with such situations. Another disadvantage of this method is that the method only considered stationary surface. To solve PDEs on moving surface, this method cannot be applicable. Moreover, the method is nonadaptive in nature. This will, therefore, increase the cost of both computing and storage as it requires a larger set of points of nodes to capture all of the solution's characteristics. We are working on an adaptive node arrangement to deal with these issues.

Wavelets have been used for more than 10 years to effectively address PDEs that illustrate multi-scale solutions [27,42,207,208] on the surfaces having zero curvature. The main point is to resolve PDE on arbitrary surfaces. Numerous approaches for constructing wavelets on arbitrary surfaces have been proposed such as

1. Wavelet basis are designed on a certain kind of manifolds that could be presented as a disjoint union of a basic cube's smooth parametric images in [44, 145]. The construction depends merely on the unit cube's smooth parametrization, which has various practical disadvantages.

2. The above issue is solved in [46], by constructing wavelet basis functions from a basic finite element discretization.

3. An orthogonal MRA on the sphere is constructed for Spherical Shannon wavelets [47].

4. Sweldens and his co-workers constructed the second generation wavelet in [209] and the major benefit of this wavelet is that it can be constructed on any arbitrary manifold.

The wavelet theory for numerically solving PDEs on arbitrary manifolds is quiet in its emerging phase, although there is wide literature available on wavelets methods for general manifolds. The second-generation wavelet prompted wavelet-based solution of PDEs at first in [91], which deal with laminar diffusion frame equations, 1-D Burger's problem and modified Burger's problems. This work has been a big climacteric for the researchers who use wavelets for numerically solving PDEs, because of the many practical benefits of second-generation wavelet over the existent wavelets. In [48], second-generation wavelet was later utilized for resolving PDEs on a sphere, here the spherical second-generation

wavelet is utilized for constructing a dynamically adaptive numerical technique. The major complication is that an initial mesh structure is needed for second-generation wavelet for approximating the manifold (e.g., an icosahedron mesh is required for approximating the sphere). On the other hand, for an arbitrary manifold, it is difficult to generate an initial mesh. This problem is resolved in [53, 54]. In these papers diffusion wavelet and spectral graph wavelet is used for solving PDEs on sphere. Our main focus is to solve PDEs on arbitrary manifolds.

This chapter proposes a curvelet optimized method for the numerically solving PDEs on general manifolds. The most useful property of the curvelet is that it can be designed on arbitrary manifolds. It can therefore be employed for solving PDEs on arbitrary manifolds. In this chapter, an embedding approach has been utilized to approximate the Laplacian operator which is named as closest point method (CPM) and curvelet has been used to determine whether the grid needs to be coarsened or refined in order to optimally describe the solution. The computational time carried out by the developed technique is compared to that of computational time carried out by the closest point approach and it is found that the developed technique performs better in terms of computational time, for example on sphere computational effort reduces by 4 times using developed technique. To the best of our insight, this is the first attempt to exploit the valuable properties of curvelet for solving PDEs on general manifolds. The proposed technique is tested on 3 test problems, such as, reaction-diffusion equation on a sphere, Schnakenberg model evolving on the surface of ellipsoid and the well-known Fitzhugh-Nagumo equations. These numerical outcomes reveal that the proposed methods can precisely catch the development of the localised patterns on all the scales and the arrangement of nodes are adapted accordingly. The convergence of the method has also been verified.

7.1 A short explanation of CPM

In this section, the CPM has been discussed in detail [206] and some of its key features. It is fundamental to represent the underlying surface for any numerical technique for solving PDEs on arbitrary manifolds. The CPM relies upon representing the closest point form of the underlying manifold.

7.1.1 Closest Point Function (CPF)

CPF: When the surface `S` is given, $cp(x)$ specifies a point belongs to `S` (possibly non-unique) which is nearest to x. The CPF which is characterized in the neighborhood of the surface, provides the surface representation. The representation of closest point form takes into consideration arbitrary surfaces with boundary conditions and does not need the surface to have an outside/inside. The surface can be of mixed codimension [206], or even of any codimension [210].

The objective of the CPM is that a surface PDE should be replaced by an associated PDE in the embedding space that could be solved by utilizing finite element, finite difference or another standard techniques. This extension will be selected accordingly that the embedded PDE ought to be the logical expansion of an initial, *i.e.*, the embedded PDE is formed when intrinsic surface Laplacian is replaced by the standard Laplacian on R^3. Certainly, both developments can not be agreed for long periods of time, but if we choose an appropriate extension at the initial stage, then the development of the embedded PDE would be precise, that would be adequate for updating the solution in time. Therefore, the operator E will be chosen which extends the function characterized on the surface to the function characterized on all of R^3 such that the natural expansion of the surface PDE to the entire space produces the desired change rate. The space equation, which is the natural extension of the surface equation intrinsically, results in a particular class of extension operator, specifically, those who extend values of a function to be constant along normal surface direction.

The method is generally defined by choosing an extension operator that generates a constant normal extension. On the other hand, the means by which the constant normal extension is actually constructed is an element that is critical to the easiness of the development approach. An especially straightforward, precise and effective approach for constant normal extension is to utilize "closest point" presentation of the surface. For any point x in R^3, let $CP(x)$ denote the closest point to x on the surface S. If S is smooth, this function is well defined and smooth near the surface. (It will generally have discontinuities away from S, at points in space that are equidistant from multiple points on S. Such points do not interfere with the embedding PDE method here, which ultimately relies only on points near S, but they may place an upper limit on grid spacing in R^3, in practice). The CPF gives a particular appropriate surface representation, as the normal constant extension operator is simply be represented as composition of function, accordingly as

$$E[u_s](x) = u_s(CP(x)),$$

for a u_s specified on S. In this way, what might be a construction procedure is diminished to standard development, that enormously streamlines both the hypothetical analysis and practicable implementation of the technique. CP gives values lying in S which is just directly a map from R^3 to R^3.

It should be note that using the CPF to represent the underlying surface provides advantages that extends beyond having a fast, simple and accurate extension procedure. We enjoy the flexibility with the closest point representation to represent both closed and open surfaces as well as surfaces having no orientation, such as a Klein bottle or a Mobius strip. Furthermore, filaments or "curves"- objects of co-dimension two or larger than two, are usually included in this design, such as composite objects like collections of surfaces, points and curves. Therefore, in particular, the formalism does not place any restraints on the geometry, topology or dimensionality of the object where the PDE is posed.

If the closest point function is not given to us as an element of the given problem, then there are various possible methods to determine it. It is easy to analytically express the closest point function for simple surfaces such as the sphere or torus, in practice. Standard numerical optimization methods can be employed for computing the closest-point function for parameterized surfaces such as a Mobius strip or an ellipsoid. Subsequently, we can either find out the closest point at the nodes of the grid by employing various kinds of sophisticated algorithms for instance, tree based algorithms of Strain [211] and the public domain closest point transform of Mauch [212] or we can use direct techniques, when the surfaces are in a triangulated form. We note that for a specific shape, the closest-point representation must only be calculated once because the interpolation can be applied to map a well resolved closest-point representation to a desirable computational grid. This means that even inefficient methods are often sufficient to compute the closest-point, for example carrying out convenient local optimizations and search through a table of triangles which describe a triangulated surface. For any situation, the closest-point function will eventually be stored on the underlying computational grid which is used for discretizing the PDE of interest, that will be a cartesian, uniform grid represented in a band over the surface.

7.1.2 Closest Point Method (CPM)

The CPM for developing PDEs on general manifolds is presented in this section. For initializing the method we perform various steps:

- The surface closest-point representation $CP(x)$ will be constructed in accordance with section 7.1 if it is not given earlier.

- The computational field, Ω_c, is selected. The computational field will customarily consist of a band over the surface.
- In the usual cartesian co-ordinates of R^3, the embedded surface PDE is obtained when surface Laplacian is replaced by the standard Laplacian in R^3. As there is no projection matrix associated, therefore it forms a simple PDE than former embedding techniques [202, 203].
- When the initial data of the surface is extended onto the computing space then the solution variable is initialized by using the CPF.

The CPM then continues by alter the subsequent 2 steps:

1. By using the CPF, the surface solution is extended to the computational space *i.e.*, u is replaced by $u(CP)$ for every node of grid onto the computing space.
2. Standard finite differences are used to calculate the solution of the embedding PDE on a cartesian mesh for 1 time step (The closest-point extension must be performed after every step when a Crank-Nicolson technique is chosen for evaluation of time, so that all quantities are assessed at their closest neighbors).

The embedded PDE solution at the surface approximates the surface PDE at any time-step.

7.1.3 Banding

The CPM is different in many aspects within the category of embedded techniques. For instance, it uses CPF to represent the surface. Instead of using the projection operators, the CPM technique uses the familiar and obvious cartesian analogue of the underlying surface PDEs, which makes the CPM method different. Another difference between the CPM and other embedding techniques identifies with how the embedding PDEs are employed and how this affects the evolution of narrow banding-based useful algorithms. Now we give the detail explanation of this last point.

Any embedded technique must consider the embedding PDE on a confined band enclosing the surface for obtaining the efficient algorithms

$$\Omega_c = \{x : \|x - CP(x)\|_2 \leq \lambda\},$$

here λ is a bandwidth. The former embedding techniques deal with the underlying embedding PDE, that is represented throughout the space and its solution, if confined to the

surface, deals with the initial surface PDE for all times t. Restricting the calculation to a confined band obscures the solution significantly for various causes:

- When the embedded PDE is solved on a band, it necessitates the artificial boundaries to be imposed at the boundaries of the computational band as the embedded PDE is described all over time and space. When the band-width changes according to the spacing of mesh, then the choice of these boundaries are not clearly understood and it can prompt the deterioration in the order of efficiency.
- The selection of a bandwidth λ is not clear and the analytical arguments do not favor it.
- The technique should be developed for propagating the surface quantities onto the computational band for improving the consistency and bound the impacts of the imposed boundary conditions.

For the CPM, the evolution approach is fundamentally distinctive. The embedded PDE coincide with the underlying surface PDE, when the surface values coincide with a constant normal surface data extension. Evidently, the need for such unique data suggests that for all periods of time t the embedded PDE can not provide a solution to the underlying surface PDE. Furthermore, a constant normal extension initializes the embedding PDE and this means that the surface and the embedding PDE initially concede on a surface. A constant normal extension of the data of the surface is provided by the consequent closest-point extension step which is convenient for the subsequent phase of the technique. This is all necessary for consistency with explicit time-stepping. The reasonable extension all over the space and detachment of the evolution at the surface considerably explains the confined band as it does not include imposed boundary conditions (while reinforcing the solution's consistency). When the band-width is adequately wide, then at the surface the CPM provides the similar results as an all space computation, when the computation is performed on a band with explicit time stepping. We now present description of the bandwidth λ selection.

Assume that we are dealing with dimensions d and that interpolation is done by using polynomials of order p according to standard Newton divided differences. Another interpolation methods *i.e.*, WENO or ENO may also be utilized and these techniques will commence the large computational bands. We considered the arbitrarily point x on a surface for obtaining the bound on the band-width. Every Newton divided difference interpolation demands that the $(p+1)^d$ node values appearing in the interpolatory stencil have been advanced precisely. According to the differencing stencil, every nodal value rely upon its neighbors which is utilized in the evolution of PDE stage. Assembling the arrangement of

all such neighbors N, it is apparent that the maximal Euclidean distance from a grid node to x which belongs to N gives the bound on the band-width.

An example illustrates this calculation most easily. Assume that the third order interpolation polynomials ($p = 3$) is used and we work with the standard five point Laplacian in 2-D ($d = 2$). Any grid node into the stencil can not exceeds a distance $\sqrt{2^2 + 2^2}\Delta x$ from x. The relevant band should also contain that points because this node point value will rely upon its 4 nearest neighbors. It leads to $\sqrt{2^2 + 3^2}\Delta x$ band-width. In general, performing this computation in dimensions d for the 2nd order centered difference Laplacian operator examined in this work prompts the value of band-width

$$\lambda = \sqrt{(d-1)\left(\frac{p+1}{2}\right)^2 + \left(1 + \frac{p+1}{2}\right)^2}\,\Delta x.$$

Similarly, sharp bounds on the band-width may be determined for other interpolation or differencing stencils.

7.2 Curvelet optimized method for solving PDEs on surfaces (COMS)

Now, the differential equation on the general manifolds has been considered which is given as:

$$\begin{aligned} \frac{\partial u_s}{\partial t} &= \nabla^2 u_s + f(x, u_s, t), \\ u(x, 0) &= u_0(x). \end{aligned} \tag{7.2.1}$$

where ∇^2 is the Laplace Beltrami operator or surface Laplacian. Suppose that time is discretized with forward Euler, the corresponding surface evolution problem reads

$$u_s^{n+1} = u_s^n + \Delta t \nabla^2 u_s^n + \Delta t f(x, u_s^n, t).$$

Instead of treating this equation of the surface, we develop the equation into the embedding space,

$$u^{n+1} = u^n(CP) + \Delta t \nabla^2 u^n + \Delta t f(CP, u^n(CP), t),$$

∇^2 is to be discretized on a grid in $\mathbb{R}^3$ (∇^2 is the standard Cartesian derivative operators in $\mathbb{R}^3$). We just merely update the analogous 3-D problem for obtaining the update of surface function by using the complete standard cartesian operators on the grid for evaluating

the right-hand-side. On the grid nodes, the function of the surface which is updated is displayed. An interpolation step is performed on the basis of these nodal values for obtaining the values of u at the requisite points of surface.

7.2.1 Numerical algorithm for solving PDEs

We presently have all the constituents that are required for constructing an adaptive curvelet technique to solve PDEs on arbitrary manifolds. The algorithm for solving PDEs on general manifolds as per the following

1. Discretize the PDE's domain.
2. By using closest point method, discretize the Laplacian operator engaged with the PDEs on general manifolds
3. Assume the times $t_1, t_2, \cdots$ are chosen to refine the grid (Noted that the selection of these times depends on the problem. If the solution of the problem changes rapidly, then t_is should be selected more closely). Initiate with an initial condition, and by using the time-integration method, we get the solution at the time t_1, i.e., $u(t_1)$.
4. Using the procedure explained in the previous section, we use $u(t_1)$ and the related grid X^{t_1} to achieve $X^{t_1+\Delta t}$.
5. Compute the Laplacian-Beltrami operator by using closest point method on $X^{t_1+\Delta t}$.
6. Integrate with time-integration scheme, the obtained system of ordinary differential eqns. in time to find the solution at $t = t_1 + \Delta t$.
7. Again follow the step 6 until $u(t_2)$ is acquired. Go to step 4, after obtained $u(t_2)$.

7.3 Numerical results and discussions

Test problem I

Firstly, we examine the reaction-diffusion on the sphere

$$\begin{aligned} \frac{\partial u}{\partial t} &= f(u,v) + \nu\nabla^2 u, \\ \frac{\partial v}{\partial t} &= g(u,v) + \mu\nabla^2 v, \end{aligned} \tag{7.3.1}$$

where $f(u,v) = c(1-u) - uv^2$ and $g(u,v) = -(c+d)v + uv^2$. For solving the reaction-diffusion system on a sphere, the initial conditions u_0=1-pert and v_0= 0.5pert have been considered where pert= $0.5\exp(-10(z-1)^2 + 0.5x)$. The parameters of Eqn. (7.3.1) are $c = 0.054$, $d = 0.063$, $\nu = \frac{1}{(3/dx)^2}$ and $\mu = \frac{\nu}{3}$. In Fig. (7.1)(a) we have plotted the solution (u) of Eqn. (7.3.1) at time $t = 100$, Fig. (7.1)(b) displays its curvelet coefficients at finest scale and Fig. (7.1)(c) shows its related adaptive grid. Fig. (7.2) shows the solution (u) of Eqn. (7.3.1), its curvelet coefficients at finest scale and the related adaptive node arrangement at time $t = 500$. Fig. (7.3) and Fig. (7.4) represents the solution (v) of Eqn. (7.3.1), its curvelet coefficients at finest scale and its related adaptive node arrangements at distinct times $t = 100$ and $t = 1000$ respectively. It is clear that COMS is capable of tracking the development of banded structures across the whole sphere. This shows that the developed approach is capable of capturing effectively and precisely the evolution of multiscale localized structures that represents the nonlinear PDE solution. Table (7.1) displays the CPU time variation with threshold. It has been noticed that Θ decreases on decreasing the threshold value. The higher the Θ value, more efficient the adapted algorithm is.

Table 7.1: The performance of COMS for reaction-diffusion equation on a sphere.

Threshold (ϵ)	N($\epsilon = 0$)=289 N(ϵ) (When we take ϵ into account)	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
0.07	676	1.899	0.4375	4.3405
0.03	841	2.976	1.345	2.212
0.009	961	3.142	1.846	1.7020
0.0009	1089	3.885	2.923	1.329

Test problem II

In 1977, Schnakenberg [213] settled the basis for explaining the biological models developed in the chemical or thermo-dynamic reaction system by dragging its state adequately far from the thermo-dynamic equilibrium. The Schnakenberg phenomena is one of the numerous different reaction models which display Turing patterns and is a hypothetical model in some sense. Due to its simplicity it stands out from the others as a good prototype

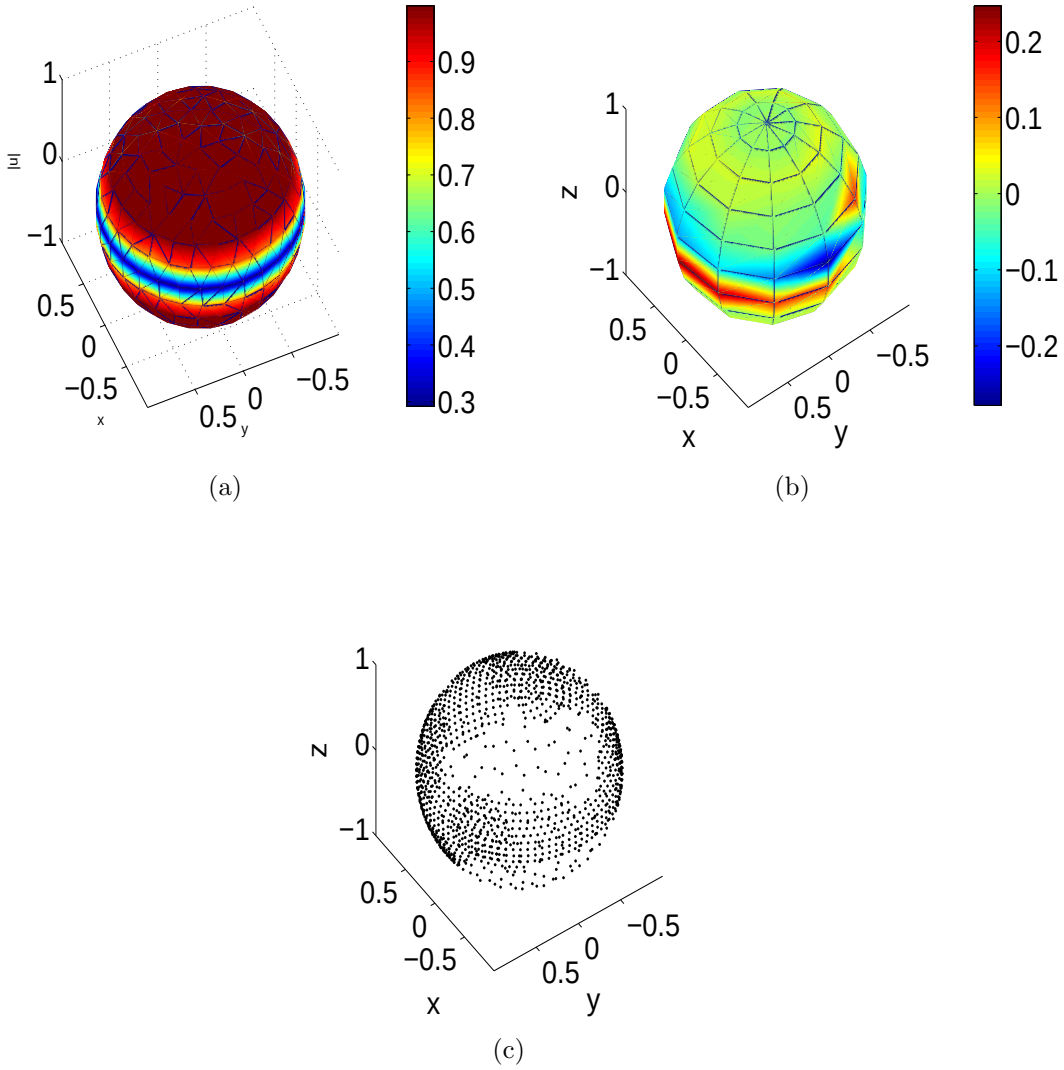


Figure 7.1: (a) The solution (u) for reaction-diffusion equation at time t=100 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

model. It has now been shown that this phenomenon occurs in chemistry and biology. In this example, the simulated system is as:

$$\frac{\partial u}{\partial t} = \gamma(a - u + u^2 v) + \nabla^2 u, \tag{7.3.2}$$

$$\frac{\partial v}{\partial t} = \gamma(b - u^2 v) + \nu \nabla^2 v, \tag{7.3.3}$$

where v is the inhibitor and u is the activator. For perturbing the system afar from the equilibrium, the initials at every point (x, y, z) on a surface are set as

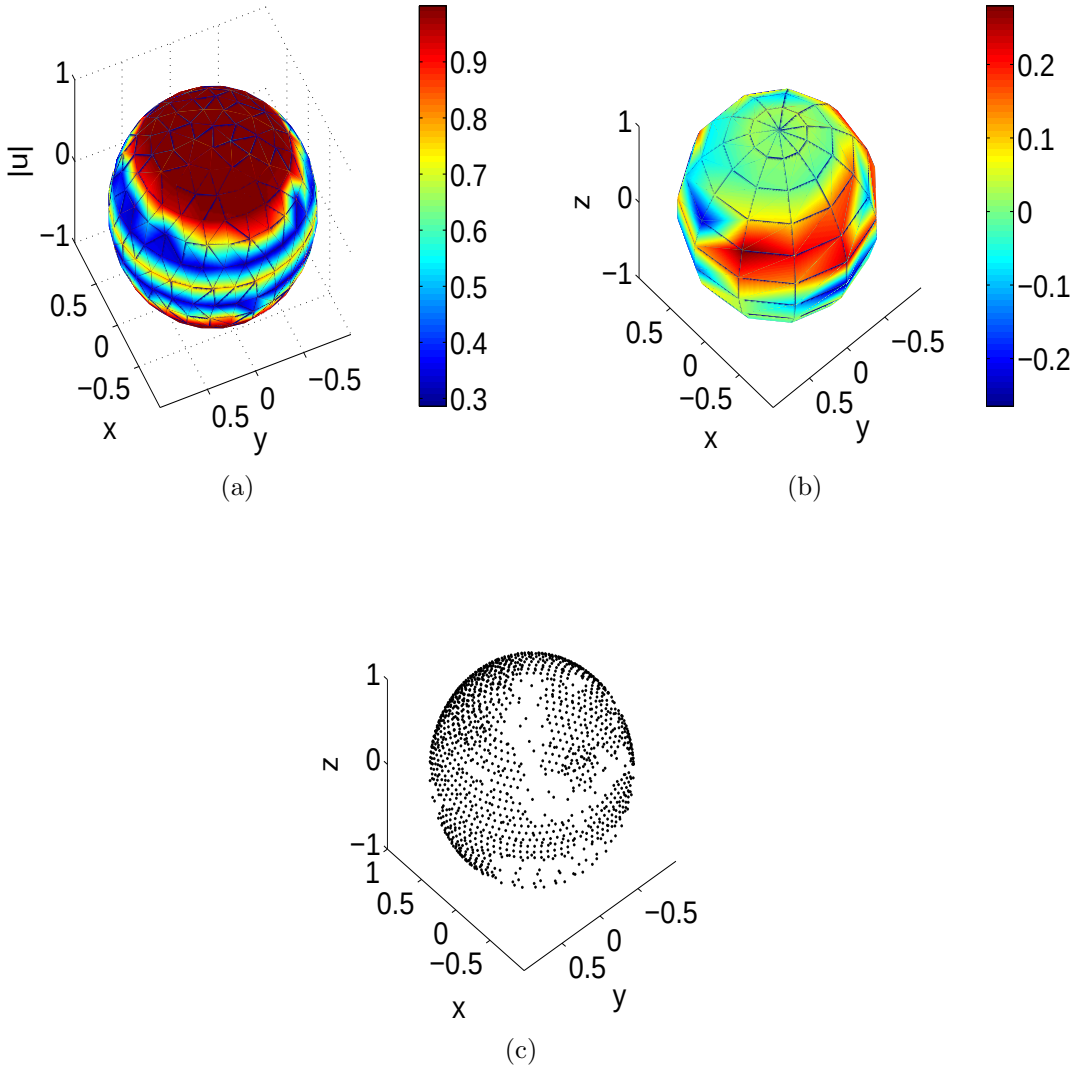


Figure 7.2: (a) The solution (u) for reaction-diffusion equation at time t=500 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

$$u(x,y,z) = b + a + \sum_{i=1}^{5} \frac{1}{20i} \sin(2\pi ix)\sin(2\pi iy)\sin(2\pi iz)$$

$$v(x,y,z) = \frac{a}{(a+b)^2} + \sum_{i=1}^{5} \frac{1}{20i} \cos(2\pi ix)\cos(2\pi iy)\cos(2\pi iz)$$

In this calculation, the parameters are set freely accordingly the values used in [214]: $\gamma = 500$, $a = -0.048113$, $\nu = 120$ and $b = 1.202813$. In Fig. (7.5)(a) we have plotted the solution (u) of Schnakenberg system at time $t = 0.01$, Fig. (7.5)(b) displays its curvelet coefficients at finest scale and Fig. (7.5)(c) shows its corresponding adaptive grid. Fig.

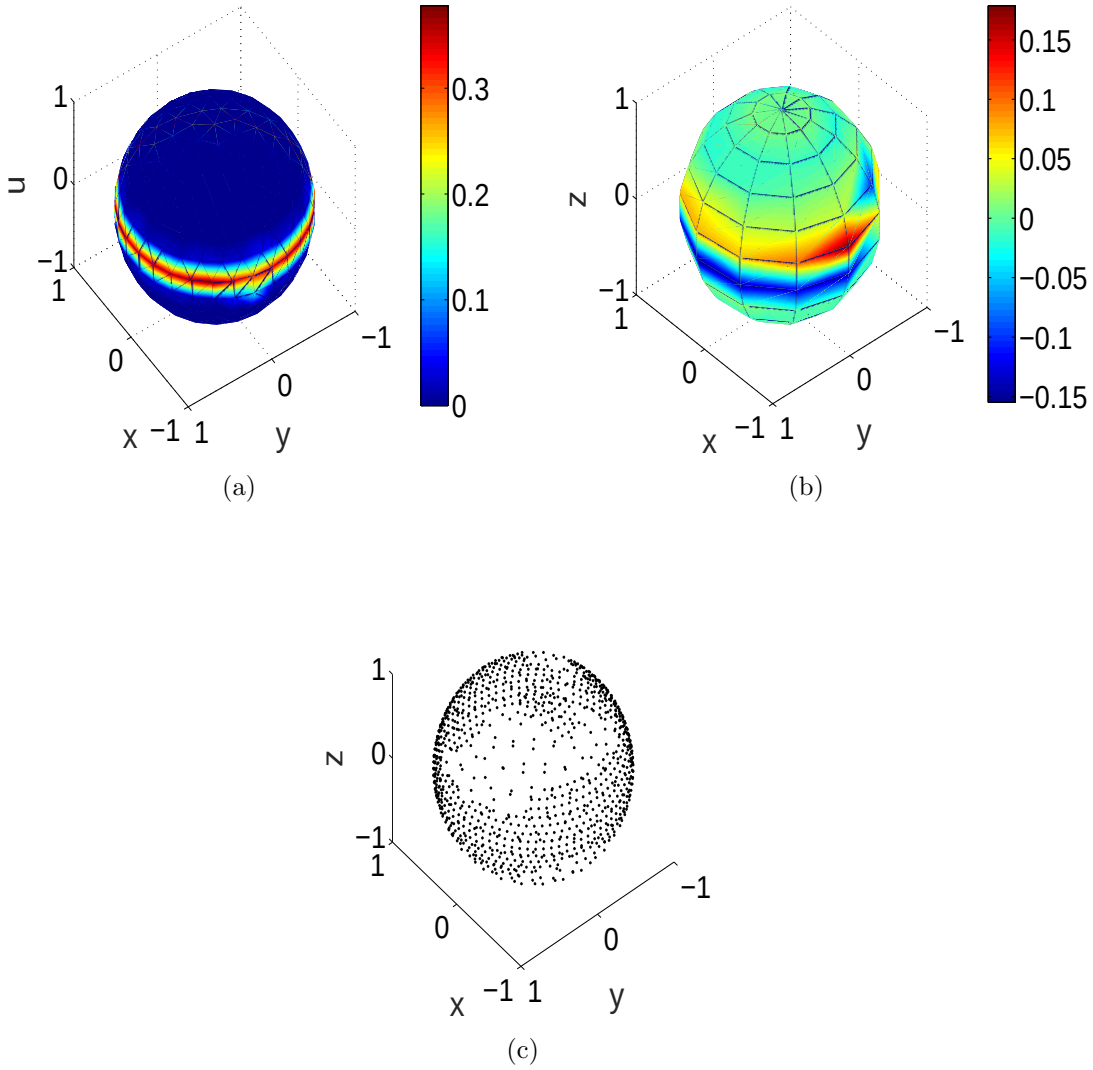


Figure 7.3: (a) The solution (v) for reaction-diffusion equation at time t=100 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

(7.6) represents the solution (u) of Eqn. (7.3.3), its curvelet coefficients at finest scale and its related adaptive node arrangement at time $t = 0.05$. Fig. (7.7) and Fig. (7.8) represents the solution (v) of Eqn. (7.3.3), its curvelet coefficients at finest scale and its related adaptive node arrangements at distinct times $t = 0.01$ and $t = 0.07$ respectively. From these figures, we have found that near the singularity, the grid becomes denser. Table (7.2) displays the values of Θ for distinct values of threshold. It has been seen that as Θ decreases on decreasing the value of ϵ.

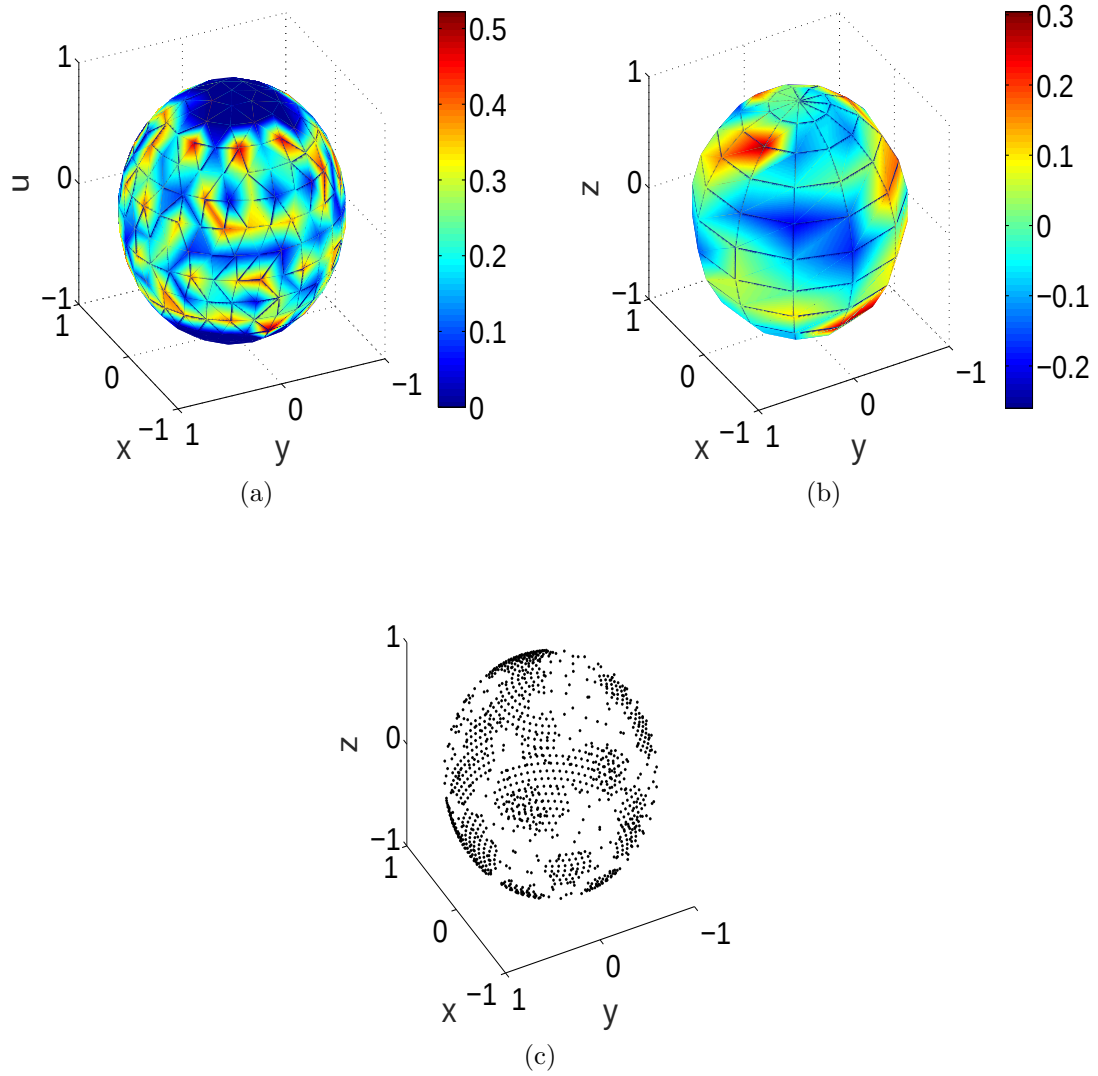


Figure 7.4: (a) The solution (v) for reaction-diffusion equation at time t=1000 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

Test problem III

The FitzHugh-Nagumo model [215] is an excitable media generic model that can be applicable for various systems. FitzHugh called the Bon Hoeffer-Van-der Pol as his simplified model and conclude it as a simplified version of the Hodgkin-Huxley equations in the 1960s. It has been widely used because of its generality and quiet two variable shape. The model can produce numerous subjective features of electrical impulses along heart and nerve fibers, for example absolute and relative refractory periods, the occurence of an excitation threshold and the formation of pulse trains by outer currents. The following

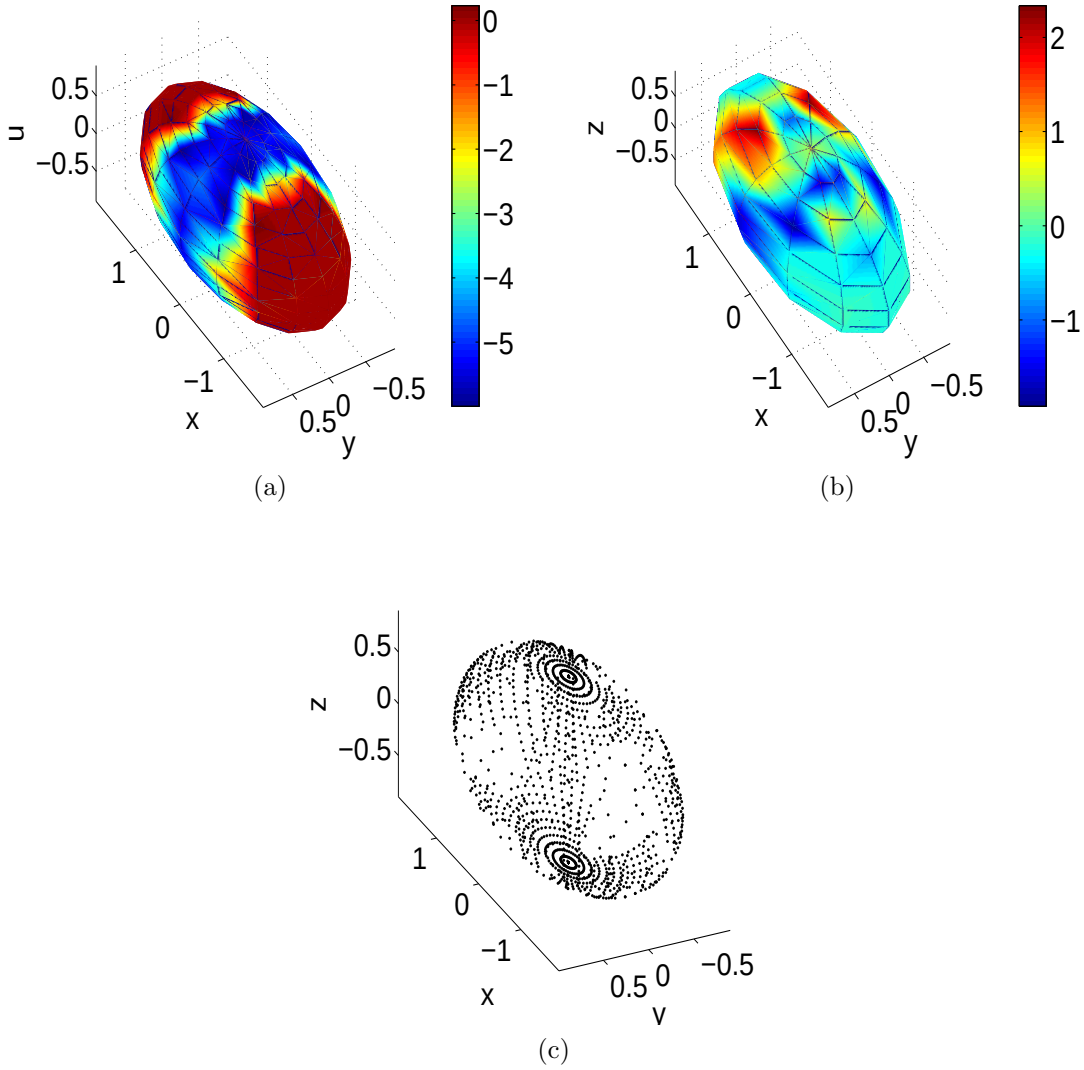


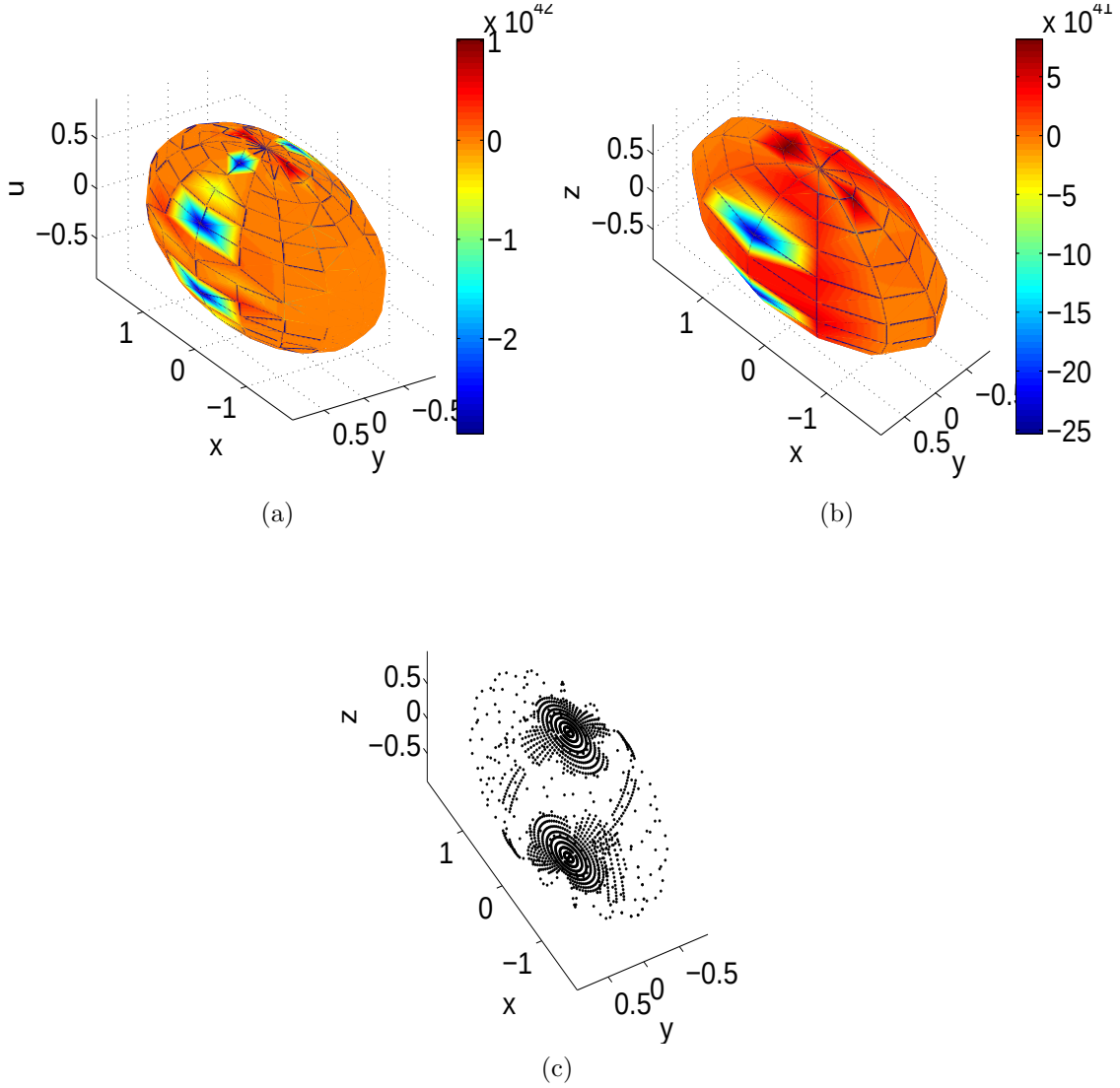
Figure 7.5: (a) The solution (u) for Schnakenberg equation at time t=0.01 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

equations describe the model

$$\frac{\partial u}{\partial t} = (m-u)(u-1)u - v + \nu\nabla^2 u, \tag{7.3.4}$$

$$\frac{\partial v}{\partial t} = n(\beta u - v). \tag{7.3.5}$$

In this model, m serves as the excitation threshold, n means the excitability and u is the excitation variable. The parameters considered for this model are $n = 0.01$, $m = 0.1$,



(a) (b) (c)

Figure 7.6: (a) The solution (u) for Schnakenberg equation at time t=0.05 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

$\beta = 0.5$ and $\nu = 0.0001$. The initials for this model are considered as

$$(u, v) = \begin{cases} (1, 0) & ; x > 0, y > 0, z > 0, \\ (0, 1) & ; x < 0, y > 0, z > 0, \\ (0, 0) & ; otherwise. \end{cases}$$

Fig. (7.9) displays the solution u of Fitzhugh-Nagumo model on cylinder, its curvelet coefficients plot at finest scale and its correspondingly adapted grid at time $t = 100$. We have also plotted the solution of the same model, its curvelet coefficients plot at finest level and related adaptive node arrangement which is represented in the figure (7.10) at

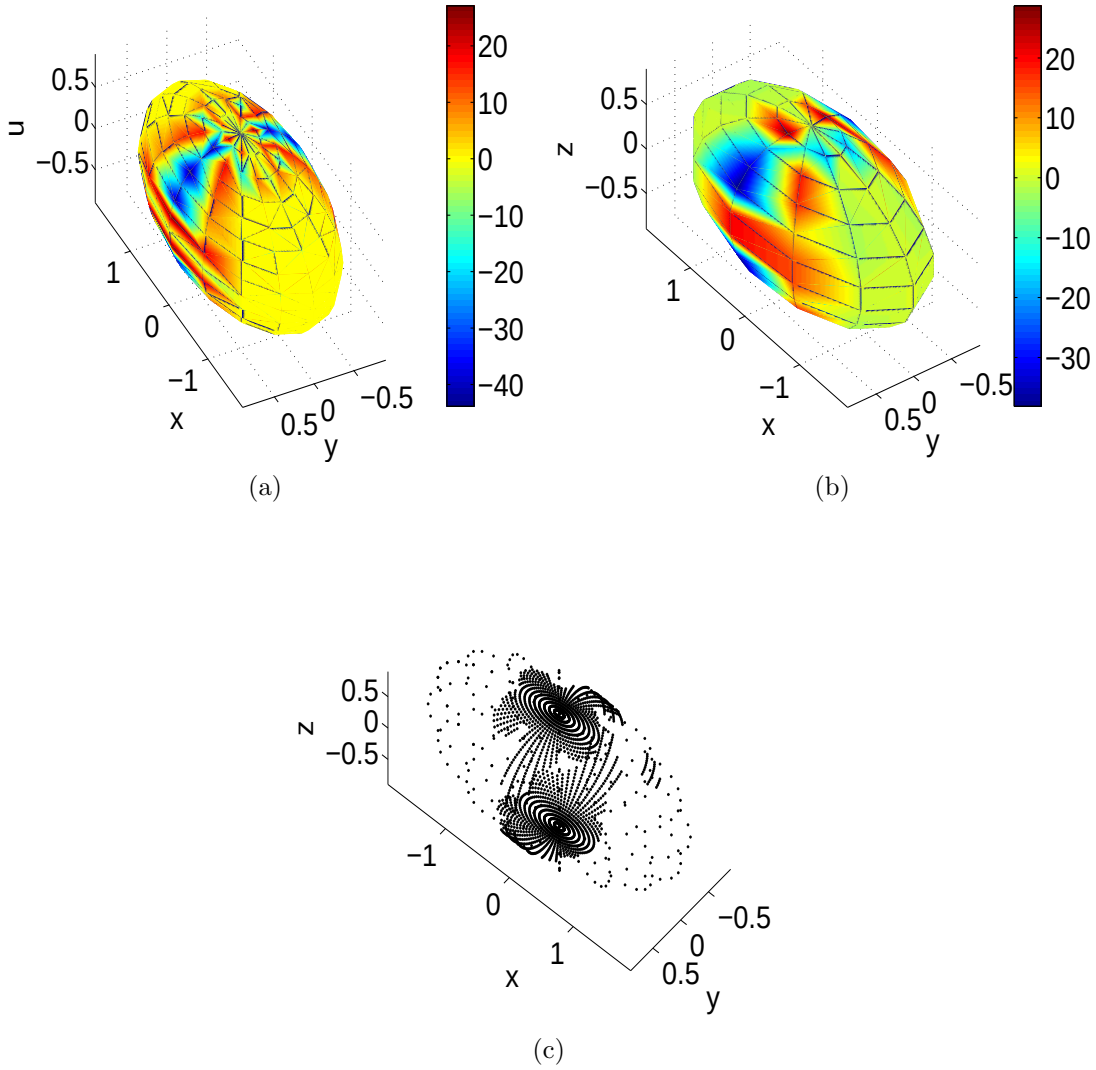


Figure 7.7: (a) The solution (v) for Schnakenberg equation at time t=0.01 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

time $t = 150$. Table (7.3) displays the Θ values for distinct values of threshold. It has been viewed that Θ decreases on decreasing the value of ϵ. The higher the Θ value, more efficient the adaptive method is.

7.4 Conclusion and Future scope

To solve PDEs on the general manifolds, we propose an adaptive curvelet technique. The closest point method, the embedding approach, is utilized for tackling PDEs on general manifolds. The method focuses on choosing the surface's closest point representation.

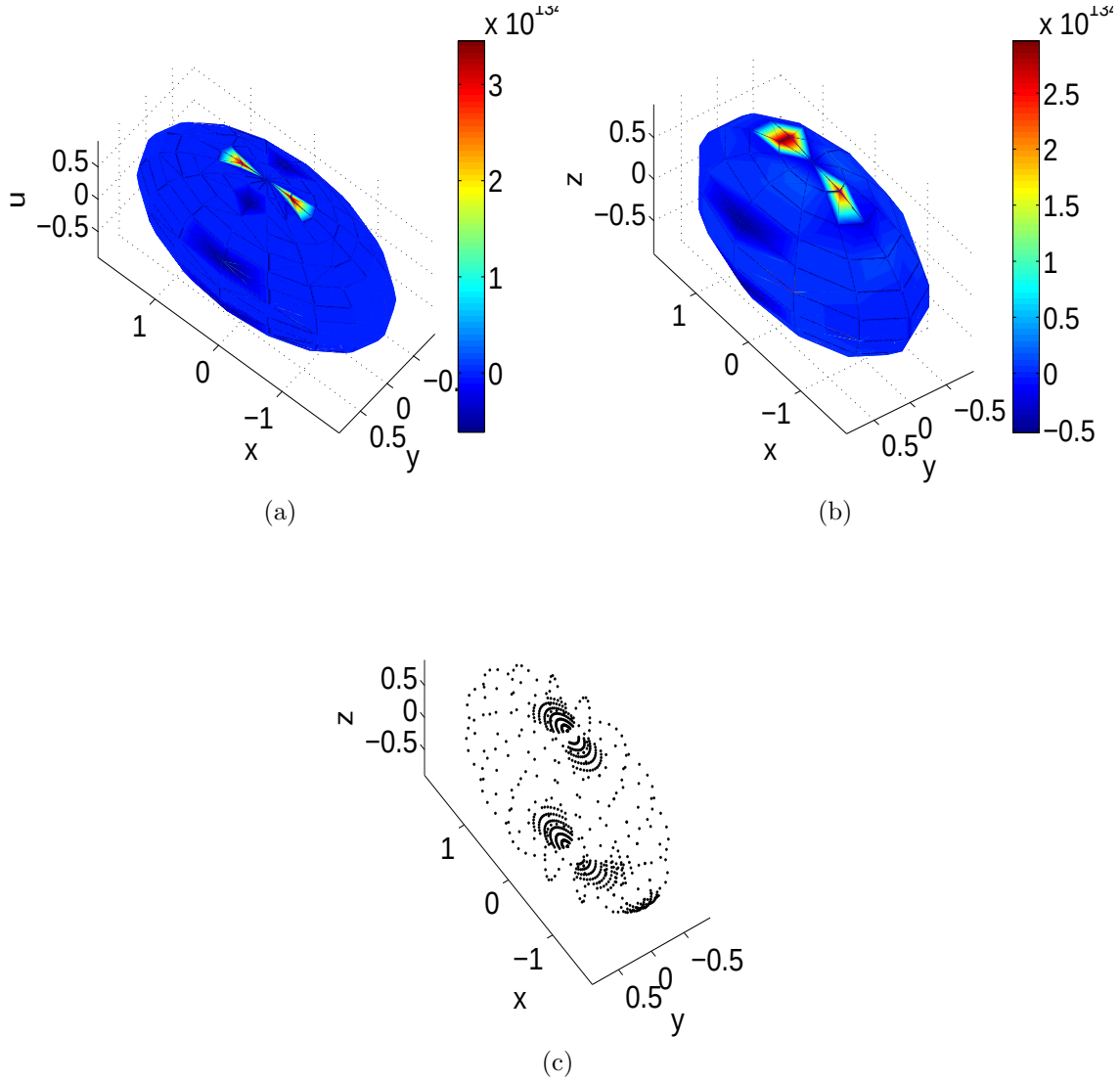


Figure 7.8: (a) The solution (v) for Schnakenberg equation at time t=0.07 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

The representation provides adaptability to treat non-orientated surfaces, open surfaces, two or higher co-dimensional objects and set of objects with different co-dimension. In addition, for adapting the node arrangement curvelet has been used. The convergence and efficiency of the developed technique has been checked. To best of our insight, this is the first time to exploit the advantageous features of curvelet for solving PDEs on an adaptive node framework on the general manifolds. The outcomes show that the curvelet and computational grid can very proficiently adapt to the solution's local irregularities for resolving the sharp transition areas.

In the future, we will adopt our method for solving industrial and academic problems that can not be easily deal with using traditional mesh-based techniques. For instance,

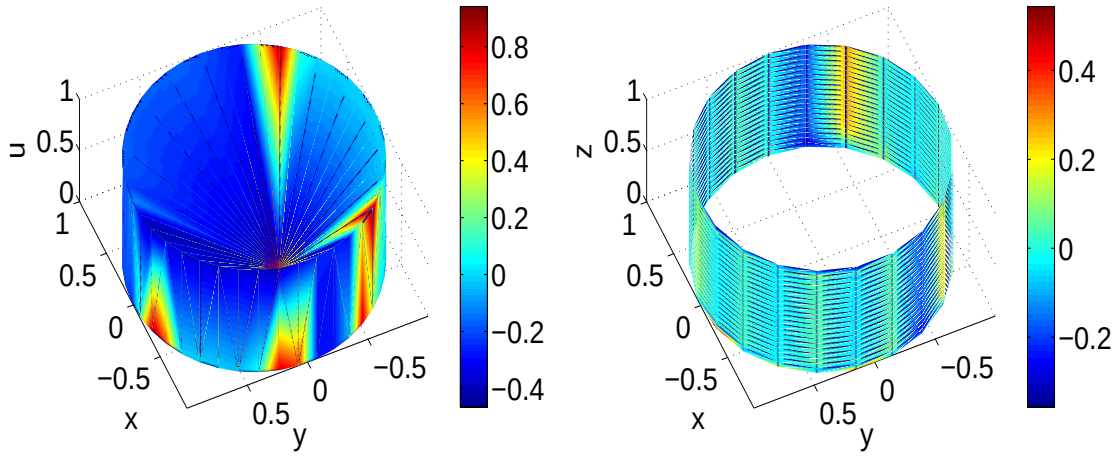
Table 7.2: The performance of COMS for Schnakenberg system on an ellipsoid.

Threshold (ϵ)	N($\epsilon = 0$)=289 N(ϵ) (When we take ϵ into account)	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
0.5	625	4.2562	0.812	5.241
0.1	729	5.483	1.762	3.1118
0.01	1024	8.567	2.998	2.857
0.009	1296	10.234	4.9878	2.0510

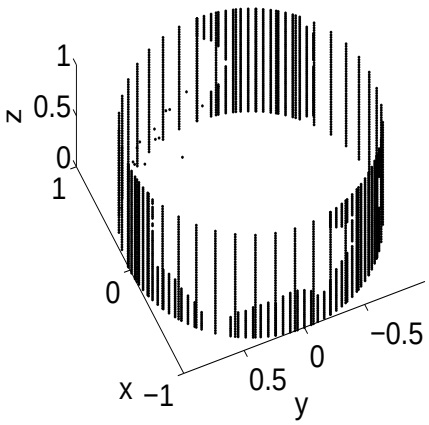
Table 7.3: The performance of COMS for Fitzhugh-Nagumo equations on a cylinder.

Threshold (ϵ)	N($\epsilon = 0$)=1764 N(ϵ) (When we take ϵ into account)	CPU time taken (in seconds) ($\epsilon = 0$)	CPU time taken (in seconds) (When we take ϵ into account)	Θ
0.9	2116	16.423	2.895	5.673
0.1	2601	18.212	3.854	4.725
0.01	3025	20.68	4.987	4.146
0.05	3249	25.824	6.998	3.6901

simulating production processes such as extrusion and modeling, where extremely large mesh deformations need to be addressed. We will further use this technique to model turbulence. It is also interesting to study flows on more generic objects or more general flows.

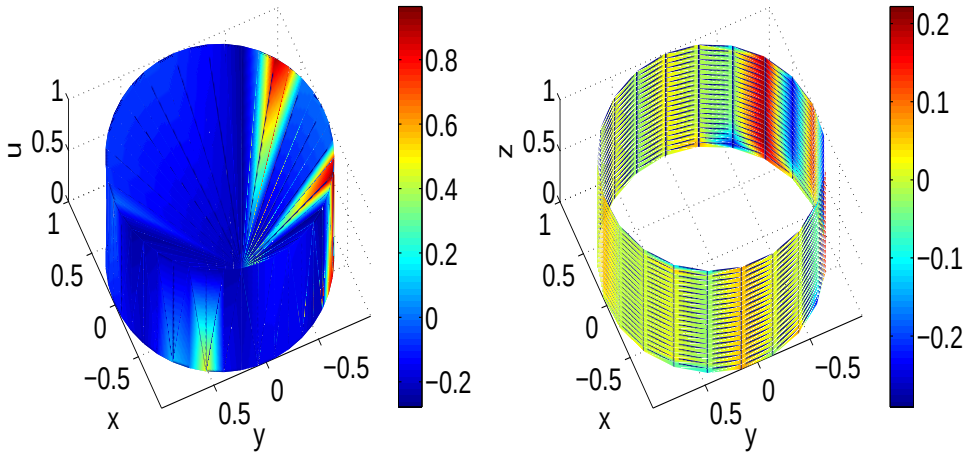


(a) (b)

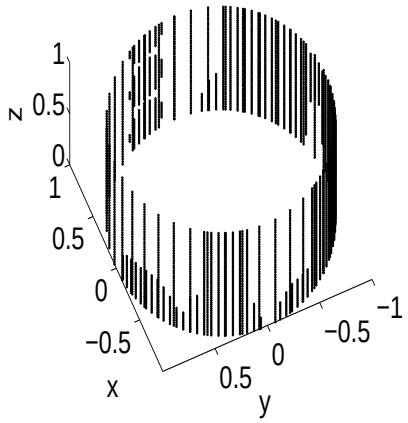


(c)

Figure 7.9: (a) The solution (u) for Fitzhugh-Nagumo equation at time t=100 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.



(a) (b)



(c)

Figure 7.10: (a) The solution (u) for Fitzhugh-Nagumo equation at time t=150 (b) Coefficients plot at finest scale and (c) its corresponding adaptive grid.

Conclusion and Future Scope

Conclusion

In this thesis, wavelet and its variants based fast adaptive methods have been developed for numerically solving PDEs. Daubechies wavelet, second generation wavelet, spectral graph wavelet (SPGW) and curvelet are used for this purpose. Curvelet is already used in various areas of engineering, but to best of our knowledge it has very thin visibility in the field of PDEs. We have made an attempt to exploit useful properties of curvelets for numerical solutions of PDEs.

We started with Daubechies and second-generation wavelet. Daubechies wavelet have been widely used for numerically solving PDEs, but we have used Daubechies wavelet for solving real life problems *i.e.*, traffic flow problems. Furthermore, we developed a Matlab toolbox which contains the routines for the second generation wavelet transform and inverse wavelet transform on the space $\mathcal{L}_2([a, b])$. These wavelet transforms are further used for computing the wavelet and scaling function values ($\psi(x)$ and $\phi(x)$ respectively). After this we have used this developed toolbox to numerically solve the Burger's equation with distinct boundaries. Furthermore, SPGW has also been used for numerically computing the solution of the Burger's equation with different boundaries as our aim was to tackle PDEs with different boundaries.

To be able to solve PDEs with solution having orientation features, we shifted to curvelet. We developed fast, adaptive methods based on curvelet for numerical solutions of PDEs. These methods use finite difference (on the Euclidean domains) or Radial basis functions (on the sphere) or closest point method (on general manifolds) for space discretization and some suitable method (*e.g.*, Crank-Nicolson) for time discretization of a given PDE. The curvelet is used for the compression of the differential operators involved in the PDE's numerical solution and for adaptivity. Following are the characteristics of the above developed methods:

- For compression of the differential operators and hence for rapid computing the powers of the matrices associated with the PDE's numerical solution.
- Methods are used for adaptivity and hence for refining the grid.

The proposed methods are tested on a large number of test problems. Numerical results show that the developed methods can precisely capture the development of the localized

patterns on all the scales and the arrangement of nodes are adapted accordingly. The CPU time analysis of the methods reveals that the methods are efficient as compared to the traditional methods.

Future directions

- We have used second-generation wavelet, SPGW and curvelet in the sense of adaptivity. Curvelet has also been used for compression of the operators. The use of these wavelets in collocation and Galerkin sense [13, 216] for solving PDEs can be an interesting and useful contribution to the field of wavelet-based numerical techniques for solving PDEs. For this, one has to design an algorithm for computing the values of the derivatives of scaling and wavelet functions as exists in the case of Daubechies wavelet.

- One of the important fields in computational biology is to represent and analyze signals given on a biological surface. For example the cortical thickness which is the distance between the outer and the inner cortical surfaces, is very important indicator for the development of the brain and the disorders in the brain. Comparison of these signals from the body of a sick person with the signals from the body of a healthy person can help in early detection of the disease. Fourier transform and Gabor transforms are used to represent and analyse these signals [217, 218].

  Wavelets are also widely used for this purpose. For example second generation wavelets are used for cortical surface shape analysis in [219] and Haar and Daubechies wavelets are used for active shape models in [220]. Recently Nain and Haker used spherical wavelets to analyse biological shapes in [221–223]. The common technique is to map the original shape onto the unit sphere using some conformal mapping and then use the spherical wavelet transform on the unit sphere for the representation and analysis of signals. Since both the SPGW and curvelet can be constructed on arbitrary shaped manifolds, the cost of mapping the original shape onto the unit sphere can be saved.

- Our results can be used to solve academic and industrial problems that can not be treated effectively by using traditional mesh based techniques. For instance, simulation of production procedures, for example, molding and extrusion, where immensely large deformation of the mesh need to be addressed [224].

- Wavelet techniques are a comparatively new field of research in computational fluid

dynamics (CFD). MRA-based methods have been studied intensively within the framework of compressible Euler equation [83, 84] whereas the use of the wavelets for simulating and modeling the turbulent flows is comparatively a new field of study. In the review article [150], a summary of wavelet methods is well summarized. The use of curvelet and SPGW for turbulence modeling and simulations is a challenging and useful area.

- Space-time wavelet methods where the wavelet is used both for the spatial adaptation and time-stepping adaptation is ideal for the problems which are concurrently intermittent in both time and space. Many space time wavelet methods have been developed [138–140]. We propose the use of curvelet and SPGW for time-step adaptivity as a future direction.

Bibliography

[1] A. Grossmann and J. Morlet, Decomposition of Hardy functions into square integrable wavelets of constant shape, SIAM J. Math. Anal. 15, 723-736, 1984.

[2] Y. Y. Tang, Wavelet theory and its applications to pattern recognition, World Scientific, 2000.

[3] Y. Y. Tang, J. Liu, H. Ma and B. F. Li, Wavelet orthonormal decompositions for extracting features in pattern recognition, Intern. J. Pattern Recognit. Artif. Intell. 13, 803-831, 1999.

[4] Y. Y. Tang, L. Yang and J. Liu, Characterization of Dirac-structure edges with wavelet transform, IEEE Trans. Syst. Man Cybern B Cybern. 30, 93-109, 2000.

[5] S. Mallat, A wavelet tour of signal processing - The sparse way. Academic Press, 2009.

[6] E. Avci and Z. H. Akpolat, Speech recognition using a wavelet packet adaptive network based fuzzy inference system, Expert Syst. Appl. 31, 495-503, 2006.

[7] E. J. Stollnitz, T. D. DeRose and D. H. Salesin, Wavelets for Computer Graphics: Theory and Applications. Morgan Kaufmann, San Francisco, 1996.

[8] M. Farge, Wavelet transforms and their application to turbulence, Annu. Rev. Fluid Mech. 24, 395-457, 1992.

[9] C. Meneveau, Analysis of turbulence in the orthogonal wavelet representation, J. Fluid Mech. 232, 469-520, 1991.

[10] O. M. Nilesen, Wavelets in scientific computing, Ph.D. thesis, Technical University of Denmark, Lyngby, 1998.

[11] A. Cohen and R. Masson, Wavelet methods for second order elliptic problems-preconditioning and adaptivity, SIAM Journal on Scientific Computing, 21, 1006-1026, 1999.

[12] Y. Maday, V. Perrier and J. C. Ravel, Dynamical adaptivity using wavelets basis for the approximation of partial differential equations, Comptes rendus de l'Acadmie des sciences 312, 405-410, 1991.

[13] A. Latto, H. Resnikoff and E. Tenenbaum, The evaluation of connection coefficients of compactly supported wavelets, in: Proceedings of the French-USA workshop on wavelets and turbulence, NY: Springer-Verlag, 76-89, 1991.

[14] R. A. Gopinath, W. M. Lawton and C. S. Burrus, Wavelet-Galerkin approximation of linear translation invariant operators, ICASSP91: International Conference on Acoustics, Speech and Signal Processing, 3, 2021-2023, 1991.

[15] G. Beylkin, R. Coifman and V. Rokhlin, Fast wavelet transforms and numerical algorithms i, Commun. Pure Appl. 44, 141-183, 1991.

[16] G. Beylkin, On the representation of operators in bases of compactly supported wavelets, SIAM J. Numer. Anal. 6, 1716-1740, 1992.

[17] S. Jaffard and P. Laurencot, Orthonormal wavelets, analysis of operators and applications to numerical analysis, chui c. wavelets: A tutorial in theory and applications, Academic Press, NewYork, 543-601, 1992.

[18] S. Jaffard and P. Laurencot, Wavelet methods for fast resolution of elliptic problems, SIAM J. Numer. Anal. 29, 965-986, 1992.

[19] G. Zweig, Wavelet transforms as solutions of partial differential equations, los alamos national lab, NM (United States) LA-UR-97-3514, 1997.

[20] W. Dahmen and C. Micchelli, Using the refinement equation for evaluating integrals of wavelets, SIAM J. Numer. Anal. 30, 507-537, 1993.

[21] M. Chen and R. Temam, Non linear Galerkin method in the finite difference case and wavelet-like incremental unknowns, Numer. Math. 64, 271-294, 1993.

[22] M. Mehra and B. V. R. Kumar, Time-accurate solution of advection diffusion problems by wavelet-taylor-Galerkin method, Commun. Numer. Methods Eng. 21, 226-313, 2005.

[23] W. Dahmen, Wavelet and multiscale methods for operator equations, Act. Num. 6, 55-228, 1997.

[24] J. Frohlich and K. Schneider, An adaptive wavelet galerkin algorithm for one dimensional and two dimensional flame compuatations, Eur. J. Mech. B/Fluids, 13, 439-471, 1994.

[25] J. Frohlich and K. Schneider, An adaptive wavelet-vaguelette algorithm for the solution of PDEs, J. Comput. Phys. 130, 90-174, 1997.

[26] V. Kumar and M. Mehra, Cubic spline adaptive wavelet scheme to solve singularly perturbed reaction diffusion problems, Int. J. Wavelets Multiresolution Inf. Process. 5, 317-331, 2007.

[27] J. Liandrat and P. Tchamitchian, Resolution of the 1d regularized burgers equation using a spatial wavelet approximation, NASA contactor report 187480, ICASE Report no. 90-83, 1990.

[28] B. V. R. Kumar and M. Mehra, Wavelet taylor Galerkin method for Burgers equation, BIT Numer. Math. 45, 543-560, 2005.

[29] L. Gagnon and J. M. Lina, Symmetrical daubechies wavelets and numerical solution of nls equations, J. Phys. A: Math. Gen. 27, 8207-8230, 1994.

[30] W. Dahmen, A. Kunoth and K. Urban, A wavelet Galerkin method for stokes equations, Computing, 56, 259-301, 1996.

[31] W. Dahmen, K. Urban, and J. Vorloeper, Adaptive wavelet methods: basic concepts and applications to the stokes problem, Singapore: World Scientific, 39-80, 2002.

[32] O. V. Vasilyev, S. Paolucci and M. Sen, A multilevel wavelet collocation method for solving partial differential equations in a finite domain, J. Comput. Phys. 120, 33-47, 1995.

[33] Y. Maday, V. Perrier and J. C. Ravel, Adaptivite dynamique sur bases dondelettes pour lapproximation dequations aux de rivees partielles, C. R. Acad. Sci. Paris 312, 405-410, 1991.

[34] Y. Maday and J. C. Ravel, Adaptivite par ondelettes: conditions aux limites et dimensions superieures, C. R. Acad. Sci. Paris 315, 85-90, 1992.

[35] S. Bertoluzza, Y. Maday and J. Ravel, A dynamically adaptive wavelet method for solving partial differential equations, Comput. Meth. Appl. Mech. Eng. 116, 293-299, 1994.

[36] E. Bacry, S. Mallat and G. Papanicolau, Wavelet based space-time adaptive numerical method for partial differential equations, Math. Model. Num. Anal. 26, 793-834, 1992.

[37] O. V. Vasilyev and S. Paolucci, A dynamically adaptive multilevel wavelet collocation method for solving partial differential equations in a finite domain, J. Comput. Phys. 125, 498-512, 1996.

[38] O. V. Vasilyev and S. Paolucci, A fast adaptive wavelet collocation algorithm for multidimensional pdes, J. Comput. Phys. 138, 16-56, 1997.

[39] P. Auscher, Compactly supported wavelets and boundary conditions, J. Funct. Anal. 111, 29-43, 1993.

[40] L. Dianfeng, O. Tadashi and Z. Lin, Treatment of boundary conditions in the application of wavelet-Galerkin method to an shwave problem, Int. J. of the Soc. of Mat. Eng. 5, 15-25, 1997.

[41] M. Bergdorf and P. Koumoutsakos, A lagrangian particle-wavelet method, Multiscale Model. Simul. 5, 980-995, 2006.

[42] A. Cohen, W. Dahmen and R. Devore, Adaptive wavelet methods for elliptic operator equations, Math. Comput. 70, 27-75, 2001.

[43] S. S. Ray and A. K. Gupta, Numerical solution of fractional partial differential equation of parabolic type with Dirichlet boundary conditions using two-dimensional Legendre wavelets method, J. Comput. Nonlinear Dynam. 11, 011012-1:9, 2016.

[44] C. Canuto, A. Tabacco and K. Urban, The wavelet element method. Part I: construction and analysis, Appl. Comput. Harmon. Anal. 6, 1-52, 1999.

[45] I. Singh and S. Kumar, Wavelet methods for solving three-dimensional partial differential equations, Math. Sci. 11, 145-154, 2017.

[46] W. Dahmen and R. Stevenson, Element-by-element construction of wavelets satisfying stability and moment conditions, SIAM J. Numer. Anal. 37, 319-352, 1999.

[47] W. Freeden and M. Schreiner, Orthogonal and non-orthogonal multiresolution analysis, scale discrete and exact fully discrete wavelet transform on the sphere, Constr. Approx. 14, 493-515, 1997.

[48] M. Mehra and N. K. R. Kevlahan, An adaptive wavelet collocation method for the solution of partial differential equations on the sphere, J. Comput. Phys. 227, 5610-5632, 2008.

[49] M. Mehra and N. K. R. Kevlahan, An adaptive multilevel wavelet solver for elliptic equations on an optimal spherical geodesic grid, SIAM J. Sci. Comput. 30, 3073-3086, 2008.

[50] G. E. Fasshauer, Meshfree Approximation Methods with MATLAB, Interdiscip. Math. Sci. 6, World Scientific, Hackensack, NJ, 2007.

[51] A. Ferreira, E. J. Kansa, G. Fasshuer and V. M. A. Leitao, Progress on meshless methods, Computational Methods in Applied Sciences (2009).

[52] B. Fornberg and E. Lehto, Stabilization of rbf-generated finite difference methods for convective pdes, J. Comput. Phys. 230, 2270-2285, 2011.

[53] K. Goyal and M. Mehra, An adaptive meshfree diffusion wavelet method for partial differential equations on the sphere, J. Comput. Phys. 272, 747-771, 2014.

[54] K. Goyal and M. Mehra, An adaptive mesh free spectral graph wavelet method for partial differential equations, Appl. Numer. Math. 113, 168-185, 2016.

[55] S. Bertoluzza, A posteriori error estimates for the wavelet Galerkin method, Appl. Math. Lett. 8, 1-6, 1995.

[56] S. Bertoluzza, G. Naldi, and J. C. Ravel, Wavelet methods for the numerical solution of boundary value problems on the interval, in Wavelets: Theory, Algorithms and Applications, C. Chui, L. Montefusco, and L. Puccio, eds., Academic Press, New York, 425-448, 1994.

[57] S. Bertoluzza, Interior estimates for the wavelet Galerkin method, Numer. Math. 78, 1-20, 1997.

[58] S. Bertoluzza and M. Verani, Convergence of a nonlinear wavelet algorithm for the solution of pdes, Appl. Math. Lett. 16, 113-118, 2003.

[59] T. Dubos and N. K. R. Kevlahan, A conservative adaptive wavelet method for the shallow-water equations on staggered grids, Q. J. R. Meteorol. Soc. 139, 1997-2020, 2013.

[60] M. Aechtner, N. K. R. Kevlahan and T. Dubos, A conservative adaptive wavelet method for the shallow-water equations on the sphere, Q. J. R. Meteorol. Soc. 141, 1712-1726, 2015.

[61] N. K. R. Kevlahan, T. Dubos and M. Aechtner, Adaptive wavelet simulation of global ocean dynamics using a new Brinkman volume penalization, Geosci. Model Dev. 8, 3891-3909, 2015.

[62] A. Cohen, N. Dyn and S. M. Kaber, Multiresolution schemes on triangles for scalar conservation laws, J. Comput. Phys. 161, 264-286, 2000.

[63] A. Cohen, S. M. Kaber and M. Postel, Adaptive multiresolution for finite volume solutions of gas dynamics, Comput. Fluids 32, 31-38, 2003.

[64] S. Berrone and L. Dmmel, Towards a realization of a wavelet Galerkin method on non-trivial domains, J. Sci. Comput. 17, 307-317, 2002.

[65] N. Radha, Molecular simulation of liquid crystal polymer flow: A wavelet-finite element analysis, Ph.D. Thesis, Massachusetts Institute of Technology, 1998.

[66] J. R. Williams and K. Amaratunga, Introduction to wavelets in engineering, Int. J. Numer. Methods Eng. 37, 2365-2388, 1994.

[67] Z. Chen and C. A. Micchelli, Discrete wavelet petrov-Galerkin methods, Adv. Comput. Math. 16, 1-28, 2002.

[68] W. Sweldens and R. Piessens, Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions, SIAM J. Numer. Anal. 31, 1240-1264, 1994.

[69] X. F. Chen, B. Li, Y. He and Z. J. He, Second generation wavelet finite element and rotor cracks quantitative identification method, Chin. J. Mech. Eng. 23, 195-199, 2010.

[70] B. Li, X. F. Chen and Z. J. He, A wavelet-based error estimator and adaptive scheme for plate bending problems, Int. J. Comput. Methods 7, 241-259, 2010.

[71] B. Li, X. F. Chen and Z. J. He, Detection of crack location and size instructures using wavelet finite element methods, J. Sound Vib. 285, 767-782, 2005.

[72] X. F. Chen, Z. J. He, J. W. Xiang and B. Li, A dynamic multiscale lifting computation method using Daubechies wavelet, J. Comput. Appl. Math. 188, 228-245, 2006.

[73] W. Dahmen and A. Kunoth, Multilevel preconditioning, Numer. Math. 63, 315-344, 1992.

[74] E. J. Candes and D. L. Donoho, Curvelets - a surprisingly effective non-adaptive representation for objects with edges, Curves and Surfaces, Vanderbilt University Press, Nashville TN, 105-120, 2000.

[75] E. J. Candes, L. Demanet, D. L. Donoho and L. Ying, Fast discrete curvelet transforms, Multiscale Model. Simul. 5, 861-899, 2006.

[76] P. Kittipoom, G. Kutyniok and W. Q. Lim, Construction of compactly supported shearlet frames, Constr. Approx. 35, 21-72, 2012.

[77] C. Lessig, P. Petersen and M. Schafer, Bendlets: A second-order shearlet transform with bent elements, Appl. Comput. Harmon. Anal. 5, 861-899, 2017.

[78] I. Daubechies, Ten Lectures on Wavelets, SIAM. Philadelphia, 1992.

[79] S. G. Mallat, Multiresolution approximations and wavelet orthonormal bases of $L_2(R)$. Transactions of the AMS, 315, 69-87, 1989.

[80] G. Strang and T. Nguyen, Wavelets and Filter Banks. Wellesley-Cambridge press, 1996.

[81] B. V. R. Kumar and M. Mehra, Time accurate fast wavelet-Taylor Galerkin method for partial differential equations, Numer Methods Partial Differ Equ. 22, 274-295, 2005.

[82] B. V. R. Kumar and M. Mehra, A time-accurate pseudo-wavelet scheme for parabolic and hyperbolic PDE's, Nonlinear Anal. 63, 345-356, 2005.

[83] A. Cohen, Wavelet methods in numerical analysis, Handb. Numer. Anal. 7, 417-711, 2000.

[84] S. Muller, Adaptive multiscale schemes for conservation laws, Lect. Notes Comput. Sci. Eng. 2003.

[85] B. V. R. Kumar and M. Mehra, A time-accurate pseudo-wavelet scheme for two-dimensional turbulence, Int. J. Wavelets Multiresolution Inf. Process. 3, 587-599, 2005.

[86] M. Mehra and K. Goyal, Algorithm 929: A suite on wavelet differentiation algorithms, ACM Trans. Math. Softw. 39, 1-28, 2013.

[87] P. K. Sahu and S. S. Ray, Numerical solutions for the system of Fredholm integral equations of second kind by a new approach involving semiorthogonal B-spline wavelet collocation method, Appl. Math. Comput. 234, 368-379, 2014.

[88] S. Leonard, M. Terracol and P. Sagaut, A wavelet-based adaptive mesh refinement criterion for large-eddy simulation, J. Turbul. 7, 1-25, 2006.

[89] T. Liu and P. Schwarz, Multiscale modelling of bubbly systems using wavelet-based mesh adaptation, Book Ser. Lect. Notes Comput. Sci. 3516, 112-119, 2005.

[90] L. Jameson, On the wavelet-optimized finite difference method, Technical Report NASA CR-191601: ICASE Report No. 94-9, 1994.

[91] O. V. Vasilyev and C. Bowman, Second-generation wavelet collocation method for the solution of partial differential equations, J. Comput. Phys. 165, 660-693, 2000.

[92] V. Kumar and M. Mehra, Wavelet optimized finite difference method using interpolating wavelets for solving singularly perturbed problems, Journal of wavelet theory and applications, 1, 83-96, 2007.

[93] V. Kumar, Solving singularly perturbed reaction diffusion problems using wavelet optimized finite difference and cubic spline adaptive wavelet scheme, Int. J. Numer. Anal. Model, 5, 270-285, 2008.

[94] Y. Tao, E. C. Lam and Y. Y. Tang, Feature extraction using wavelet and fractal, Pattern Recognit Lett. 22, 271-287, 2001.

[95] L. Jameson, T. Adachi, O. Ukai and A. Yuasa, Wavelet-based numerical methods, Int. J. Comput. Fluid Dyn. 10, 267-280, 1998.

[96] R. B. Bauer, A hybrid adaptive eno scheme, J. Comput. Phys. 136, 180-196, 1997.

[97] K. Goyal and M. Mehra, A fast adaptive diffusion wavelet method for Burger's equation, Comp. Math. Appl. 68, 568-577, 2014.

[98] T. Belytschko, Y. Y. Lu and L. Gu, Element-free Galerkin methods, Int. J. Numer. Methods Eng. 37, 229-256, 1994.

[99] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming and P. Krys, Meshless methods: an overview and recent development, Comput. Methods Appl. Mech. Eng. 139, 3-47, 1996.

[100] B. Nayroles, G. Touzot and P. Villon, Generalizing the finite element method diffuse approximation and diffuse elements, Comput. Mech. 10, 307-381, 1992.

[101] S. N. Atluri and T. Zhu, A new mesh less local petrov Galerkin (mlpg) approach in computational mechanics, Comput. Mech. 22, 117-127, 1998.

[102] G. R. Liu and Y. T. Gu, A point interpolation method for two-dimensional solids, Int. J. Numer. Methods Eng. 50, 937-951, 2001.

[103] D. F. Wu, K. He and X. Ye, Meshless method based on wavelet function, Recent Advances in Computer Science and Information Engineering, Lect. Notes in Electrical Eng. 125, 755-760, 2012.

[104] Y. N. Liu, Y. H. Liu and Z. Z. Cen, Daubechies wavelet meshless method for 2-d elastic problems, Tsinghua Sci. Technol. 13, 605-608, 2008.

[105] Y. N. Liu, F. Qin, Y. H. Liu and Z. Z. Cen, A daubechies wavelet based method for elastic problems, Eng. Anal. Bound. Elem. 34, 114-121, 2010.

[106] C. A. Brebbia, J. C. R. Telles and L. C. Wrobel, Boundary element techniques: Theory and applications in engineering, Springer-Verlag, New York, 1984.

[107] M. Spasojevic, R. Schneider and P. L. Levin, On the creation of sparse boundary element matrices for two dimensional electrostatics problems using the orthogonal haar wavelet, IEEE Trans. Dielectr. Electr. Insul. 4, 249-258, 1997.

[108] P. L. Levin, M. Spasojevic and R. Schneider, Creation of sparse boundary element matrices for 2-dandaxis-symmetric electrostatics problems using the bi-orthogonal haar wavelet, IEEE Trans. Dielectr. Electr. Insul. 5, 469-484, 1998.

[109] Z. Y. Zhao and X. Wang, Error estimation and h adaptive boundary elements, Eng. Anal. Bound. Elem. 23, 793-803, 1999.

[110] K. Koro and K. Abe, Non-orthogonal spline wavelets for boundary element analysis, Eng. Anal. Bound. Elem. 25, 149-164, 2001.

[111] K. Abe, K. Koro and K. Itami, A h-hierarchical Galerkin bem using haar wavelets, Eng. Anal. Bound. Elem. 25, 581-591, 2001.

[112] H. Harbrecht, F. Paiva, C. Perez and R. Schneider, Biorthogonal wavelet approximation for the coupling of fem-bem, Numer. Math. 92, 35-356, 2002.

[113] P. Gonzalez, J. C. Cabaleiro and T. F. Pena, Parallel iterative solvers involving fast wavelet transforms for the solution of bem systems, Adv. Eng. Softw. 33, 417-426, 2002.

[114] J. Tausch, Sparse bem for potential theory and stokes flow using variable order wavelets, Comput. Mech. 32, 312-318, 2003.

[115] K. Koro and K. Abe, A practical determination strategy of optimal threshold parameter for matrix compression in wavelet bem, Int. J. Numer. Methods Eng. 57, 169-191, 2003.

[116] F. Bucher, L. C. Wrobel, W. J. Mansur and C. Magluta, Fast solution of problems with multiple load cases by using wavelet-compressed boundary element matrices, Commun. Numer. Methods Eng. 19, 387-399, 2003.

[117] K. Eppler and H. Harbrecht, Numerical solution of elliptic shape optimization problems using wavelet-based bem, Optim. Methods Softw. 18, 105-123, 2003.

[118] H. F. Buchera, L. C. Wrobelb, W. J. Mansurc and C. Magluta, On the block wavelet transform applied to the boundary element method, Eng. Anal. Bound. Elem. 28, 571-581, 2004.

[119] J. Ravnik, L. Skerget and M. Hribersek, The wavelet transform for bem computational fluid dynamics, Eng. Anal. Bound. Elem. 28, 1303-1314, 2004.

[120] J. Ravnik, L. Skerget and M. Hribersek, Two-dimensional velocity-vorticity based les for the solution of natural convection in a differentially heated enclosure by wavelet transform based bem and fem, Eng. Anal. Bound. Elem. 30, 671-686, 2006.

[121] J. Y. Xiao, L. H. Wen and D. Zhang, A wavelet-integration-free periodic wavelet Galerkin bem for 2d potential problems, Eng. Comput. 24, 306-318, 2007.

[122] S. Barmada, Improving the performance of the boundary element method with time-dependent fundamental solutions by the use of a wavelet expansion in the time domain, Int. J. Numer. Methods Eng. 71, 363-378, 2007.

[123] J. Ravnik, L. Skerget, M. Hribersek and Z. Zunie, Numerical simulation of dilute particle laden flows by wavelet bem-fem, Comput. Methods Appl. Mech. Eng. 197, 789-805, 2008.

[124] K. Eppler and H. Harbrecht, Wavelet-based boundary element methods for exterior electromagnetic shaping, Eng. Anal. Bound. Elem. 32, 645-657, 2008.

[125] L. Ebrahimnejad and R. Attarnejad, A novel way of using fast wavelet transforms to solve dense linear systems arising from boundary element methods, Eng. Comput. 26, 483-499, 2009.

[126] J. Y. Xiao, J. Tausch and Y. C. Hu, A-posteriori compression of wavelet-bem matrices, Comput. Mech. 44, 705-715, 2009.

[127] J. Y. Xiao, L. H. Wen and J. Tausch, On fast matrix-vector multiplication in wavelet Galerkin bem, Eng. Anal. Bound. Elem. 33, 159-167, 2009.

[128] J. Y. Xiao and W. J. Ye, Wavelet bem for large-scale stokes flows based on the direct integral formulation, Int. J. Numer. Methods Eng. 88, 693-714, 2011.

[129] J. Y. Xiao and J. Tausch, A fast wavelet-multiple method for direct bem, Eng. Anal. Bound. Elem. 34, 673-679, 2010.

[130] J. Y. Xiao, J. Tausch, Y. H. Cao and L. H. Wen, Combined equivalent charge formulations and fast wavelet Galerkin bem for 3-d electrostatic analysis, Int. J. Numer. Methods Eng. 79, 753-772, 2009.

[131] L. Ebrahimnejad, R. Attarnejad and H. Ebrahimnejad, Applying wavelets to improve the boundary element method for elasticity problems, Eng. Anal. Bound. Elem. 34, 810-818, 2010.

[132] C. Scheiblich, R. Banucu, V. Reinauer, J. Albert and W. M. Rucker, Parallel hierarchical block wavelet compression for an optimal compression of 3d bem problems, IEEE Trans. Magn. 47, 1386-1389, 2011.

[133] H. Harbrecht and M. Randrianarivony, Wavelet bem on molecular surfaces solvent excluded surfaces, Computing 92, 335-364, 2011.

[134] C. Basdevant, M. Holschneider and V. Perrier, Traveling wavelets method, Comptes Rendus De L Acad. Des Sci. Ser. I-Math. 310, 647-652, 1990.

[135] O. V. Vasilyev, Solving multi-dimensional evolution problems with localized structures using second generation wavelets, Int. J. Comp. Fluid Dyn. 17, 151-168, 2003.

[136] P. Koumoutsakos, Multiscale flow simulations using particles, Annu. Rev. Fluid Mech. 37, 457-487, 2005.

[137] M. Gutnic, A. Haefele, I. Paun and E. Sonnendrucker, Vlasov simulations on an adaptive phase-space grid, Comput. Phys. Commun. 164, 214-219, 2004.

[138] E. Bacry, S. Mallat and G. Papanicolaou, A wavelet based space-time adaptive numerical method for partial differential equation, Math. Model. Num. Anal. 26, 793-834, 1992.

[139] M. O. Domingues, S. M. Gomes, O. Roussel and K. Schneider, An adaptive multiresolution scheme with local time stepping for evolutionary pdes, J. Comput. Phys. 227, 3758-3780, 2008.

[140] M. O. Domingues, S. M. Gomes, O. Roussel and K. Schneider, Space-time adaptive multiresolution methods for hyperbolic conservation laws: applications to compressible euler equations, Appl. Numer. Math. 59, 2303-2321, 2009.

[141] S. Muller and Y. Stiriba, Fully adaptive multiscale schemes for conservation laws employing locally varying time stepping, J. Sci. Comput. 30, 493-531, 2007.

[142] J. Alam, N. K. R. Kevlahan and O. V. Vasilyev, Simultaneous space-time adaptive wavelet solution of nonlinear partial differential equations, J. Comp. Phys. 214, 829-857, 2006.

[143] N. K. R. Kevlahan, J. M. Alam and O. V. Vasilyev, Scaling of space-time modes with reynolds number in two dimensional turbulence, J. Fluid Mech. 570, 217-226, 2007.

[144] N. K. R. Kevlahan and O. V. Vasilyev, An adaptive wavelet collocation method for fluid-structure interaction at high Reynolds number, 26, 1894-1915, 2005

[145] W. Dahmen and R. Schneider, Wavelets on manifolds i: construction and domain decomposition, SIAM J. Math. Anal. 31, 184-230, 1999.

[146] W. Sweldens, The lifting scheme: A construction of second generation wavelets, SIAM J. Math. Anal. 29, 511-546, 1998.

[147] V. A. Galaktionov and J. L. Vazquez, The problem of blow-up in nonlinear parabolic equations, Discrete Contin. Dyn. Syst. 8, 399433, 2002.

[148] V. Kumar and B. Srinivasan, An adaptive mesh strategy for singularly perturbed convection diffusion problems, Appl Math. Model. 39, 2081-2091, 2015.

[149] J. R. Rice, Mathematical analysis in the mechanics of fracture, Chapter3 of Fracture: an advanced treatise, 2, 191311, 1968.

[150] K. Schneider and O. V. Vasilyev, Wavelet methods in comutational fluid dynamics, Annu. Rev. Fluid Mech. 42, 473-503, 2010.

[151] V. Kumar and M. Mehra, Wavelet optimized finite difference method using interpolating wavelets for self-adjoint singularly perturbed problems, Journal of Comput. and Appl. Math. 230, 803-812, 2009.

[152] M. J. Berger and J. Oliger, Adaptive mesh refinement for hyperbolic partial differential equations, J. Comput. Phys. 53, 482-512, 1984.

[153] A. Cohen, Adaptive methods for pde's wavelets or mesh refinement? Proceedings of the ICM, Beijing, 1, 607-620, 2002.

[154] D. W. Huang, Wavelet analysis in a traffic model, Physica A. 329, 298-308, 2003.

[155] I. Antoniou, Vi. V. Ivanov, Va. V. Ivanov, P. V. Zrelov, Wavelet filtering of network traffic measurements, Physica A. 324, 733-753, 2003. 1989.

[156] B. S. Kerner, Three-phase traffic theory and highway capacity, Physica A. 333, 379-440, 2004.

[157] M. Treiber, A. Kesting and D.Helbing, Delays, inaccuracies and anticipation in microscopic traffic models, Physica A. 360, 71-88, 2006.

[158] R. Nagai, T. Nagatani and A. Yamada, Phase diagram in multi-phase traffic model, Physica A. 355, 530-550, 2005.

[159] J. J. Wu, H. J. Sun and Z. Y. Gao, Dynamic urban traffic flow behaviour on scale-free networks, Physica A. 387, 653-660, 2008.

[160] C. Daganzo, Requiem for second-order fluid approximation to traffic flow, Transport. Res. B 29, 277-286, 1995.

[161] A. Aw, A. Klar, T. Materne and M. Rascle, Derivation of continuum traffic flow models from microscopic follow-the-leader models, SIAM J. Appl. Math. 63, 259-278, 2002.

[162] M. J. Lighthill and G. B. Whitham, On Kinematic Waves.II. A Theory of Traffic Flow on Long Crowded Roads Proc. Royal Soc. London A Math. Phys. Sci. 229, 317-345, 1955.

[163] P. I. Richards, Shock-waves on the highway, Op. Res. 4, 42, 1956.

[164] S. C. Shiralashetti, L. M. Angadi and A. B. Deshi, Wavelet based numerical solution of linear and nonlinear parabolic partial differential equations using Lifting scheme, J. Inf. Comput. Sci. 13, 22-32, 2018.

[165] S. C. Shiralashetti, L. M. Angadi and A. B. Deshi, Wavelet based Lifting scheme for the numerical solution of some class of non-linear partial differential equations, Int. J. Wavelets Multiresolution Inf. Process. 16, 1850046:1-14, 2018.

[166] A. Cohen, I. Daubechies and J. C. Feauveau, Biorthogonal bases of compactly supported wavelets, Comm. Pure Appl. Math. 45, 485-560, 1992.

[167] W. Sweledens, The lifting scheme: a custom design construction of biorthogonal wavelets, ACHA 3, 485-560, 1996.

[168] S. C. Shiralashetti, Haar wavelet based numerical method for the solution of nonlinear boundary value problems arising in fluid dynamics, Int. J. Eng. Sci. Math. 6, 27-39, 2017.

[169] P. Schroder and W. Sweldens, Wavelets in computer graphics, 1996.

[170] Y. Y. Tang, V. Wickerhauser, P. C. Yuen and C. H. Li, Wavelet analysis and its applications: Second International Conference, Hong Kong, China, December 18-20, 2001.

[171] M. Landajuel, Burgers equation, BCAM Internship Summer, 2011.

[172] V. Ulzibayar, T. Zhanlav and O. C. huluunbaatar, Higher order accurate numerical solution of Burgers equation, Int. J. Math. Sci. 33, 2051-5995, 2013.

[173] S. M. Pugh, Finite element approximations of Burgers equations, Virginia, 1995.

[174] V. Q. Nguyen, A Numerical Study of Burgers Equation With Robin Boundary Conditions, Virginia, 2001.

[175] R. R. Coifman and M. Maggioni, Diffusion wavelets, Appl. Comput. Harmon. Anal. 21, 53-94, 2006.

[176] D. K. Hammond, P. Vandergheynst and R. Gribonval, Wavelets on graphs via spectral graph theory, Appl. Comput. Harmon. Anal. 30, 129-150, 2011.

[177] K. Goyal and M. Mehra, Fast diffusion wavelet method for partial differential equations, Appl. Math. Model. 40, 5000-5025, 2016.

[178] D. L. Donoho, Interpolating wavelet transforms, Tech. Rep. 408, Department of Statistics, Stanford University, 1992.

[179] E. J. Candes and D. L. Donoho, Curvelets - a surprisingly effective nonadaptive representation for objects with edges. Curves and Surfaces, Vanderbilt University Press, Nashville, TN, 105-120, 2000.

[180] S. C. Shiralashetti, M. H. Kantli and A. B. Deshi, A new wavelet multigrid method for the numerical solution of elliptic type differential equations, Alexandria Eng. J. 57, 203-209, 2018.

[181] J. Ma and G. Plonka, A review of curvelets and recent applications, in proceedings, 2009

[182] E. J. Candes and L. Demanet, The Curvelet Representation of Wave Propagators is optimally Sparse, Comm. Pure Appl. Math. 58, 1472-1528, 2005.

[183] E. J. Candes and D. L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise C^2 singularities, Comm. Pure Appl. Math. 57, 219-266, 2004.

[184] E. J. Candes and D. L. Donoho, Continuous curvelet transform: i. resolution of the wavefront set, Appl. Comput. Harmon. Anal. 19, 162-197, 2005.

[185] B. Sun, J. Ma, H. Chauris and H. Yang, Solving the wave equation in the curvelet domain: A multiscale and multidirectional approach, J. Seism. Explor. 18, 385-399, 2009.

[186] L. Jameson, A wavelet-optimized, very high order adaptive grid and numerical method, SIAM J. Sci. Comput. 19, 1980-2013, 1998.

[187] Q. T. L. Gia, Approximation of linear partial differential equations on sphere, Ph.D. thesis, Texax A and M University, 2008.

[188] R. L. Burden and J. D. Faires, Numerical analysis. Pacific Grove, CA: Brooks/Cole Pub. Co.

[189] R. Behera and M. Mehra, A Dynamic Adaptive Wavelet Method for Solution of the Schrödinger Equation, J. Multiscale Model. 6, 1450001-22, 2015.

[190] Z. Majdisova and V. Skala, A Radial Basis Function Approximation for Large Datasets, Proceedings of Sigrad, 2016.

[191] G. Turk, Generating textures on arbitrary surfaces using reaction-diffusion, ACM Siggraph Computer Graphics, 25, 289-298, 1991.

[192] J. Dorsey and P. Hanrahan, Digital materials and virtual weathering, Scientific American, 282-289, 2000.

[193] A. Witkin and M. Kass, Reaction-diffusion textures, ACM Siggraph Computer Graphics, 25, 299-308, 1991.

[194] Y. Fu, J. Zhang and L. Wan, Application of the energy balance method to a nonlinear oscillator arising in the micro electromechanical system (mems), Curr. Appl. phys. 11, 482-485, 2011.

[195] M. Hofer and H. Pottmann, Energy-minimizing splines in manifolds, ACM Trans. Graph. 23, 284-293, 2004.

[196] K. Schittkowski, Parameter identification and model verification in systems of partial differential equations applied to transdermal drug delivery, Math. Comput. Simul. 79, 521-538, 2008.

[197] J. C. Misra, A. Sinha and G. C. Shit, A numerical model for the magnetohydrodynamic flow of blood in a porous channel, J. Mech. Med. Biol. 11, 547-562, 2011.

[198] T. G. Myers and J. P. F. Charpin, A mathematical model for atmospheric ice accretion and water flow on a cold surface, Int. J. Heat Mass Transf. 47, 5483-5500, 2004.

[199] T. G. Myers, J. P. F. Charpin and S. J. Chapman, The flow and solidification of a thin fluid film on an arbitrary three-dimensional surface, Phys. Fluids, 14, 2788-2803, 2002.

[200] U. Diewald, T. Preusser and M. Rumpf, Anisotropic diffusion in vector field visualization on euclidean domains and surfaces, IEEE Trans. Vis. Comput. Graph. 6, 139-149, 2000.

[201] M. S. Floater and K. Hormann, Surface parameterization: a tutorial and survey, 157-186, 2005.

[202] M. Bertalmio, L. T. Cheng, S. Osher and G. Sapiro, Variational problems and partial differential equations on implicit surfaces, J. Comput. Phys. 174, 759-780, 2001.

[203] M. Bertalmio, F. Memoli, L. T. Cheng, G. Sapiro and S. Osher, Variational problems and partial differential equations on implicit surfaces: Bye bye triangulated surfaces? Geometric Level Set Methods in Imaging, Vision, and Graphics, 381-397, 2003.

[204] J. J. Xu and H. K. Zhao, An eulerian formulation for solving partial differential equations along a moving interface, J. Sci. Comput. 19, 573-594, 2003.

[205] J. B. Greer, An improvement of a recent eulerian method for solving pdes on general geometries, J. Sci. Comput. 29, 321-352, 2006.

[206] S. J. Ruuth and B. Merriman, A simple embedding method for solving partial differential equations on surfaces, J. Comput. Phys. 227, 1943-1961, 2008.

[207] M. Holmstrom and J. Walden, Adaptive wavelets methods for hyperbolic PDEs, J. Sci. Comput. 13, 19-49, 1998.

[208] O. Roussel, K. Schneider, A. Tsigulin and H. Bockhorn, A conservative fully adaptive multiresolution algorithm for parabolic PDEs, J. Comput. Phys. 188, 493-523, 2003.

[209] W. Sweldens, The lifting scheme: a construction of second generation wavelets, SIAM J. Math. Anal. 29, 511-546, 1998.

[210] C. B. Macdonald and S. J. Ruuth, The implicit Closest Point Method for the numerical solution of partial differential equations on surfaces, SIAM J. Sci. Comput. 31, 4330-4350, 2009.

[211] J. Strain, Fast tree-based redistancing for level set computations, J. Comput. Phys. 152, 648-666, 1999.

[212] S. Mauch, Efficient algorithms for solving static hamilton jacobi equations. PhD thesis, California Institute of Technology, Pasedena, 2003.

[213] J. Schnakenberg, Simple chemical reaction systems with limit cycle behaviour, J. theor. Biol. 81, 389-400, 1979.

[214] S. J. Ruuth, Implicit-explicit methods for reaction-diffusion problems in pattern formation, J. Math. Biol. 34, 148-176, 1995.

[215] R. Fitzhugh, Fitzhugh-Nagumo simplified cardiac action potential model, J. Biophys. 1, 445-446, 1961.

[216] L. Jameson, On the daubechies-based wavelet differentiation matrix, J. Sci. Comput. 8, 267-305, 1993.

[217] I. W. Hunter and M. J. Korenberg, The identifications of non linear biological systems: Wiener and hammerstein cascade models, Biol Cybern. 55, 135-144, 1986.

[218] M. Porat and Y. Y. Zeevi, The generalized gabor scheme of image representation in biological and machine vision, IEEE Trans. Pattern Anal. Mach. Intell. 10, 452-462, 1988.

[219] P. Yu, P. E. Grant, Y. Qi, X. Han, F. Segonne, R. Pienaar, E. Busa, J. Pacheco, N. Makris, R. L. Buckner, P. Golland and B. Fischl, Cortical surface shape analysis based on spherical wavelets, IEEE Trans. Med. Imaging. 26, 582-596, 2007.

[220] C. Davatzikos, X. Tao, and D. Shen, Hierarchical active shape models, using the wavelet transform, IEEE Trans. Med. Imaging. 22, 414-423, 2003.

[221] D. Nain, S. Haker, A. Bobick and A. Tannenbaum, Multiscale 3-d representation and segmantation using spherical wavelets, Lecture Notes in Computer Science, 3750, 459-467, 2005.

[222] D. Nain, S. Haker, A. Bobick and A. Tannenbaum, Shape driven 3d segmentations using spherical wavelets, Lecture Notes in Computer Science, 4190, 66-74, 2006.

[223] D. Nain, S. Haker, A. Bobick and A. Tannenbaum, Multiscale 3-d shape representation and segmentation using spherical wavelets, IEEE Trans. Med. Imaging. 26, 598-618, 2007.

[224] A. Huerta, T. Belytschko, S. Fernandez-Mendez and T. Rabczuk, Meshfree methods in Encyclopedia of Computational Mechanics, John Wiley and Sons, 2004.

Appendix

List of Matlab Programs for Chapter 2

1. `Split.m` (dependency: none).

   The function `Split.m` performs the lazy wavelet transform and separates the even and odd entries of the vector u with suitable multiplication of constants. The calling command for this function is

   ```
   >>[ lambda, gamma, lambda_index, gamma_index ] = Split( u , index)
   ```

   where `u` is the vector on which the lazy wavelet transform is to be performed, `index` is the corresponding vector of indices, `lambda` and `lambda_index` are the vectors containing the odd entries and the corresponding indices respectively, `gamma` and `gamma_index` are the vectors containing the even entries and the corresponding indices respectively. The implementation goes as follows:

```
length_u= length(u);
if mod(length_u,2)==0
    lambda= zeros(1,length_u/2);
    lambda_index=zeros(1,length_u/2);
    gamma=zeros(1,length_u/2);
    gamma_index=zeros(1,length_u/2);
else
    lambda=zeros(1,(length_u+1)/2);
    lambda_index=zeros(1,(length_u+1)/2);
    gamma=zeros(1,(length_u-1)/2);
    gamma_index=zeros(1,(length_u-1)/2);
end
for i=1:length_u
    if mod(i,2)==0
        gamma(1,i/2)=u(i);
        gamma_index(1,i/2)=index(i);
    else
        lambda(1,(i+1)/2)=u(i);
        lambda_index(1,(i+1)/2)=index(i);
    end
```

```
end
lambda=(1/sqrt(2))*lambda;
gamma=sqrt(2)*gamma;
```

2. `Neville_algo.m` (dependency: none).

   The function `Neville_algo.m` implements the Neville's algorithm for interpolation. The main commands are:

```
[yi] = Neville_algo(x, y, xi)
n = length(x);
for k = 1:length(xi)
   xd = [];
   for i = 1:n
      xd(i) = abs(x(i) - xi(k));
   end

   [xds,i] = sort(xd);

   x = x(i);
   y = y(i);

   P = zeros(n,n);
   P(:,1) = y(:);

   for i = 1:n-1
      for j = 1:(n-i)
         P(j,i+1) = ((xi(k)-x(j))*P(j+1,i) +
         (x(j+i)-xi(k))*P(j,i))/(x(j+i)-x(j));
      end
   end
   yi(k) = P(1,n);

end
```

3. `Prediction_coefficients_interval.m` (dependency: `Neville_algo.m`).

   This function computes the prediction coefficients for the second generation wavelets.

The calling command for this function is

```
>>coeff=Prediction_coefficients_interval(N)
```

where `N` is an even number denoting the number of dual vanishing moments and `coeff` gives the prediction coefficients. The main commands used in this function are the following:

```
m=N+1;
Prediction_coefficients=zeros(m,N);
k1=0:N-1;
k2=-0.5:N-0.5;
for i=1:N
    tmp1=zeros(1,N);
    tmp1(i)=1;
    for  j=1:m
        [yi, ypi, P, D] = Neville_algo(k1, tmp1,k2(j));
        Prediction_coefficients(j,i)=yi;
    end
end
```

4. `Contribution.m` (dependency: `Prediction_coefficients_interval.m`).

   This function computes the matrix containing the information about the indicies of the lambdas which are used to predict gammas and the matrix containing the amount by which the lambdas have predicted gammas. The calling command for this function is

   ```
   >>[Contri_count, Contri_amount]= Contribution(length_lambda_input,N)
   ```

   where `N` is an even number denoting the number of dual vanishing moments, `length_lambda_input` is the length of lambdas at a particular level, `Contri_count` gives the matrix containing the information about the indicies of the lambdas which are used to predict gammas, `Contri_amount` gives the matrix containing the amount by which the lambdas have predicted gammas. The main commands used in this function are the following:

```
Prediction_coefficients_matrix=Prediction_coefficients_interval(N);
    if mod(length_lambda_input,2)==0
```

```
    L_tmp=length_lambda_input;
else
    L_tmp=length_lambda_input-1;
end
Contri_count =zeros(L_tmp/2,N);
Contri_amount=zeros(L_tmp/2,N);
Case_number=zeros(L_tmp/2,1);
for tmp6=2:2:L_tmp
    if ((tmp6-(N-1))>=1 && (tmp6+(N-1))<=length_lambda_input)
        Case_number(tmp6/2,1)=N/2;
    end
end
tmp7=find(Case_number);
tmp8=tmp7(1);
tmp9=tmp7(end);
for tmp10=1:tmp8-1
    Case_number(tmp10,1)=tmp10;
end
for tmp11=tmp9+1:L_tmp/2
    Case_number(tmp11,1)=N/2+tmp11-tmp9;
end
for tmp12=1:N
    for tmp13=2:2:L_tmp
        if(Case_number(tmp13/2,1)==tmp12)
            for tmp14=1:tmp12
                Contri_count(tmp13/2,tmp14)=tmp13-(2*tmp12-1)+
                2*(tmp14-1);
            end
            for tmp15=1:N-tmp12
                Contri_count(tmp13/2,tmp12+tmp15)=tmp13+2*tmp15-1;
            end
        end
    end
end
for tmp16=1:N
    for tmp17=2:2:L_tmp
        if (Case_number(tmp17/2,1)==tmp16)
```

```
                Contri_amount(tmp17/2,:)=Prediction_coefficients
                _matrix(tmp16+1,:);
            end
        end
    end
```

---

5. `Update_coefficients_interval.m` (dependency: `Prediction_coefficients_interval.m`).

   The function `Update_coefficients_interval.m` computes the update coefficients for the second generation wavelets on intervals. The calling command for this function is

   ```
   >>[Coeff_matrix, Update_moments, Contri_count, Contri_amount] =
   Update_coefficients_interval(L, Ini_mom, N, tN)
   ```

   where `L` is the length of the original signal, `N` is number of dual vanishing moments, `tN` is number of primal vanishing moments, `Ini_mom` gives a matrix of initial moments, `Coeff_matrix` gives a matrix containing update coefficients corresponding to gamma, `Update_moments` gives a matrix containing updated moments, `Contri_count` gives the matrix containing the information about the indicies of the lambdas which are used to predict gammas and `Contri_amount` gives the matrix containing the amount by which the lambdas have predicted gammas. The implementation goes as follows:

```
L=input('Enter the length of the signal.');
N=input('Enter the number of dual vanishing moments.');
tN=input('Enter the number of primal vanishing moments.');
depth=input('Enter the depth for wavelet transform.');
Coeff_Assi=zeros(depth+1,L);
Coeff_Assi(1,:)=1;
for tmp1=1:L
    if mod(tmp1,2)==1
        Coeff_Assi(2,tmp1)=1;
    else
        Coeff_Assi(2,tmp1)=2;
    end
end
```

```
Ini_mom=zeros(tN,L);
Ini_mom(1,:)=1;
for tmp2=2:tN
    Ini_mom(tmp2,:)=(1:L).^(tmp2-1);
end

if mod(L,2)==0
    L_tmp=L;
else
    L_tmp=L-1;
end
Contri_count =zeros(L_tmp/2,N);
Contri_amount=zeros(L_tmp/2,N);
    Case_number=zeros(L_tmp/2,1);
    for tmp6=2:2:L_tmp
        if ((tmp6-(N-1))>=1 && (tmp6+(N-1))<=L)
            Case_number(tmp6/2,1)=N/2;
        end
    end
    tmp7=find(Case_number);
    tmp8=tmp7(1);
    tmp9=tmp7(end);
    for tmp10=1:tmp8-1
        Case_number(tmp10,1)=tmp10;
    end
    for tmp11=tmp9+1:L_tmp/2
        Case_number(tmp11,1)=N/2+tmp11-tmp9;
    end
    for tmp12=1:N
        for tmp13=2:2:L_tmp
            if(Case_number(tmp13/2,1)==tmp12)
                for tmp14=1:tmp12
                    Contri_count(tmp13/2,tmp14)=tmp13-(2*tmp12-1)+
                    2*(tmp14-1);
                end
                for tmp15=1:N-tmp12
                    Contri_count(tmp13/2,tmp12+tmp15)=tmp13+2*tmp15-1;
```

```
                end
            end
        end
    end

Prediction_coefficients_matrix=Prediction_coefficients_interval(N);
for tmp16=1:N
    for tmp17=2:2:L_tmp
        if (Case_number(tmp17/2,1)==tmp16)
            Contri_amount(tmp17/2,:)=Prediction_coefficients
            _matrix(tmp16+1,:);
        end
    end
end

if mod(L,2)==0
    L_tmp1=L-1;
else
    L_tmp1=L;
end
for tmp18=1:2:L_tmp1
    [tmp19_row, tmp19_col] =find(Contri_count==tmp18);
    gamma_effected=2*tmp19_row;
    length_gamma_effected=length(gamma_effected);
    for tmp20=1:tN
     for tmp21=1:length_gamma_effected
      Ini_mom(tmp20,tmp18)=Ini_mom(tmp20,tmp18)+
      Contri_amount(tmp19_row(tmp21),
       tmp19_col(tmp21))*Ini_mom(tmp20, gamma_effected(tmp21));
      end
    end
end

Coeff_matrix=zeros(L_tmp/2,N);
for tmp21=2:2:L_tmp
    Left_matrix=zeros(tN,N);
    right_vec=zeros(tN,1);
```

```
    for tmp22=1:tN
     for tmp23=1:N
      Left_matrix(tmp22,tmp23)=Ini_mom(tmp22,Contri_count(tmp21/2,tmp23));
      end
        right_vec(tmp22)=Ini_mom(tmp22,tmp21);
    end
    tmp23=Left_matrix\right_vec;
    Coeff_matrix(tmp21/2,:)=tmp23';
end
```

6. `Wavelet_transform.m` (dependency: `Split.m, Prediction_coefficients_interval.m, Update_coefficients_interval.m`).

   The function `Wavelet_transform.m` performs the second generation wavelet transform on the interval. The calling command for this function is

   ```
   >>[C, L, Wavelet_transform_matrix, Wavelet_transform_index] =
   Wavelet_transform(f, N, tN, depth, index)
   ```

   where `f` is the function on which wavelet transform is performed, `index` is the corresponding vector of indices, `N` is an even number denoting the number of dual vanishing moments, `tN` is an even number denoting the number of primal vanishing moments, `depth` determines the numbers of levels upto which the wavelet transform has to be performed. The main commands used in this function are the following:

```
length_f=length(f);
Prediction_coefficients_matrix=Prediction_coefficients_interval(N);
max_NtN=max(N,tN);
max_depth_possible=floor(log2((length_f-1)/(max_NtN-1)));
if depth>max_depth_possible
    error('Maximum depth possible is %f',max_depth_possible );
end
Wavelet_transform_matrix=zeros(depth+1,length_f);
Wavelet_transform_index=zeros(depth+1,length_f);
Wavelet_transform_matrix(1,:)=f;
Wavelet_transform_index(1,:)=index;
lambda_input=f;
index_input=index;
```

```
Ini_mom=zeros(tN,length_f);
Ini_mom(1,:)=1;
for tmp2=2:tN
    Ini_mom(tmp2,:)=index.^(tmp2-1);
end
for j=1:depth
    [ lambda_tmp, gamma_tmp, lambda_index_tmp, gamma_index_tmp ] =
     Split( lambda_input , index_input);
    length_lambda_input=length(lambda_input);
    if mod(length_lambda_input,2)==0
        L_tmp=length_lambda_input;
    else
        L_tmp=length_lambda_input-1;
    end
    Contri_count =zeros(L_tmp/2,N);
    Contri_amount=zeros(L_tmp/2,N);
    Case_number=zeros(L_tmp/2,1);
    for tmp6=2:2:L_tmp
        if ((tmp6-(N-1))>=1 && (tmp6+(N-1))<=L_tmp)
            Case_number(tmp6/2,1)=N/2;
        end
    end
    tmp7=find(Case_number);
    tmp8=tmp7(1);
    tmp9=tmp7(end);
    for tmp10=1:tmp8-1
        Case_number(tmp10,1)=tmp10;
    end
    for tmp11=tmp9+1:L_tmp/2
        Case_number(tmp11,1)=N/2+tmp11-tmp9;
    end
    for tmp12=1:N
        for tmp13=2:2:L_tmp
            if(Case_number(tmp13/2,1)==tmp12)
                for tmp14=1:tmp12
                    Contri_count(tmp13/2,tmp14)=tmp13-(2*tmp12-1)+
                    2*(tmp14-1);
```

```
                end
                for tmp15=1:N-tmp12
                    Contri_count(tmp13/2,tmp12+tmp15)=tmp13+2*tmp15-1;
                end
            end
        end
    end
    for tmp16=1:N
        for tmp17=2:2:L_tmp
            if (Case_number(tmp17/2,1)==tmp16)
                Contri_amount(tmp17/2,:)=Prediction_coefficients
                _matrix(tmp16+1,:);
            end
        end
    end
    gamma_tmp_length=length(gamma_tmp);
    for tmp18=1:gamma_tmp_length
        for tmp19=1:N
            gamma_tmp(tmp18)=gamma_tmp(tmp18)-Contri_amount(tmp18,tmp19)*
            lambda_tmp((Contri_count(tmp18,tmp19)+1)/2);
        end
    end
    [Coeff_matrix, Update_moments] =
    Update_coefficients_interval1(length_lambda_input, Ini_mom, N,tN);
    lambda_tmp_length=length(lambda_tmp);
    for tmp20=1:lambda_tmp_length
        tmp20_to_original=2*tmp20-1;
        [tmp20_row, tmp20_col]=find(Contri_count==tmp20_to_original);
        for tmp21=1:length(tmp20_row)
                lambda_tmp(tmp20)=lambda_tmp(tmp20)+
                Coeff_matrix(tmp20_row(tmp21),
                tmp20_col(tmp21))*gamma_tmp(tmp20_row(tmp21));
        end
    end
Wavelet_transform_matrix(j+1,1:length(lambda_tmp))=lambda_tmp;
Wavelet_transform_matrix(j+1,length(lambda_tmp)+
1:length(lambda_input))=gamma_tmp;
```

```
 Wavelet_transform_index(j+1,1:length(lambda_tmp))=lambda_index_tmp;
 Wavelet_transform_index(j+1,length(lambda_tmp)+1:length(lambda_input))=
 gamma_index_tmp;
 lambda_input=lambda_tmp;
 index_input=lambda_index_tmp;
 Ini_mom=zeros(tN, length(lambda_tmp));
    for tmp22=1:length(lambda_tmp)
        Ini_mom(:,tmp22)=Update_moments(:,2*tmp22-1);
    end
end
end
```

7. `Wavelet_transform_inverse.m` (dependency: `Contribution.m`, `Update_coefficients_interval.m`).

   The function `Wavelet_transform_inverse.m` performs the inverse second generation wavelet transform on the interval. The calling command for this function is

   ```
   >>[f]=Wavelet_transform_inverse(Wavelet_transform_matrix,Wavelet_transfo
   rm_index, N,tN)
   ```

   where `Wavelet_transform_matrix`, `Wavelet_transform_index` are the outputs of `Wavelet transform.m`, `N` is an even number denoting the number of dual vanishing moments, `tN` is an even number denoting the number of primal vanishing moments and `f` is the reconstructed function. The main commands used in this function are the following:

```
depth=size(Wavelet_transform_matrix,1)-1;
length_f=size(Wavelet_transform_matrix,2);
index=1:length_f;
dk=floor((length(nonzeros(Wavelet_transform_matrix(depth+1,:)))+1)/2);
dk1=length(nonzeros(Wavelet_transform_matrix(depth+1,:)));
lambda_tmp=Wavelet_transform_matrix(depth+1,1:dk);
 for tmp24=dk+1:dk1
     gamma_tmp(tmp24-dk)=Wavelet_transform_matrix(depth+1,tmp24);
 end
[D1, D]= Contribution(length(nonzeros(Wavelet_transform
_index(depth+1,:))),N);
```

```
Moments=cell(depth,1);
Contribution_A=cell(depth,1);
Contribution_C=cell(depth,1);
Ini_mom=zeros(tN,length_f);
Ini_mom(1,:)=1;
for tmp2=2:tN
    Ini_mom(tmp2,:)=index.^(tmp2-1);
end
Moments{1,1}=Ini_mom;
for tmp=2:depth
    tmp1=length(nonzeros(Wavelet_transform_index(tmp,:)));
    [Coeff_matrix, Update_moments,Contri_count, Contri_amount] =
    Update_coefficients_interval(tmp1, Ini_mom, N,tN);
    Ini_mom=zeros(tN, length(nonzeros(Wavelet_transform_index(tmp+1,:))));
    for tmp22=1:length(nonzeros(Wavelet_transform_index(tmp+1,:)))
        Ini_mom(:,tmp22)=Update_moments(:,2*tmp22-1);
    end
    Moments{tmp,1}=Ini_mom;
    Contribution_A{tmp,1}=Contri_amount;
    Contribution_C{tmp,1}=Contri_count;
end
for tmp=depth+1:-1:2
        [Coeff_matrix, Update_moments,Contri_count, Contri_amount]=
            Update_coefficients_interval(length(nonzeros(Wavelet_transfo
            rm_index(tmp,:))),
             Moments{tmp-1,1}, N,tN);
        lambda_tmp_length=length(lambda_tmp);
        for tmp20=1:lambda_tmp_length
            tmp20_to_original=2*tmp20-1;
            [tmp20_row, tmp20_col]=find(Contri_count==tmp20_to_original);
            for tmp21=1:length(tmp20_row)
                lambda_tmp(tmp20)=lambda_tmp(tmp20)-
                Coeff_matrix(tmp20_row(tmp21),tmp20_col(tmp21))*gamma_tmp
                (tmp20_row(tmp21));
            end
        end
         gamma_tmp_length=length(gamma_tmp);
```

```
    for tmp18=1:gamma_tmp_length
        for tmp19=1:N
            gamma_tmp(tmp18)=gamma_tmp(tmp18)+
            D(tmp18,tmp19)*lambda_tmp((D1(tmp18,tmp19)+1)/2);
        end
    end
lambda_tmp=sqrt(2)*lambda_tmp;
gamma_tmp=(1/sqrt(2))*gamma_tmp;
for i=2:2:2*length(gamma_tmp)
    tmpd(i)=gamma_tmp(i/2);
end
for i=1:2:2*length(lambda_tmp)-1
    tmpd(i)=lambda_tmp((i+1)/2);
end
lambda_tmp=tmpd;
dk=floor((length(nonzeros(Wavelet_transform_matrix
(tmp-1,:)))+1)/2);
dk1=length(nonzeros(Wavelet_transform_matrix(tmp-1,:)));
    for tmp24=dk+1:dk1
    gamma_tmp(tmp24-dk)=Wavelet_transform_matrix(tmp-1,tmp24);
    end
[D1, D]= Contribution(length(nonzeros(Wavelet_transform
_index(tmp-1,:))),N);
```

8. `Reconstruction.m` (dependency: `Wavelet_transform.m, Wavelet_transform _inverse.m`).

   The function `Reconstruction.m` gives the error when the function f is reconstructed using second generation wavelet transform. It also plots the original given function f in blue and the reconstructed function in red. The calling command for this function is

   ```
   >>[error]=Reconstruction(x,f,N,tN,depth)
   ```

   where `x` is the grid, `f` is the function value, `N` is an even number denoting the number of dual vanishing moments, `tN` is an even number denoting the number of primal vanishing moments, `depth` determines the levels upto which the wavelet transform

is to be performed and `error` gives the reconstruction error. The main commands used in this function are the following:

```
index=1:length(x);
[C,L,Wavelet_transform_matrix, Wavelet_transform_index] =
Wavelet_transform(f,N,tN,depth,index);
[f_reconstructed]=Wavelet_transform_inverse(Wavelet_transform_matrix,
Wavelet_transform_index,N,tN);
error=norm(f-f_reconstructed);
figure;
plot(x,f,'LineWidth',3);
hold on;
plot(x,f_reconstructed,'ro');
legend('original function','reconstructed function')
grid on;
```

9. `AdaptiveGrid_SGW_modified.m` (dependency: `Wavelet_transform.m`).

   The function `AdaptiveGrid_SGW_modified.m` constructs an adaptive grid in 1D for second generation wavelet by using the modified adaptation technique. The calling command for this function is

   ```
   >>[xc] = AdaptiveGrid_SGW_modified(x,u,n,epsilon,N1,tN,depth)
   ```

   where `u` is the function value, `x` is the initial grid, `n` denotes the number of points to be inserted, `epsilon` is threshold value, `N1` is number of dual vanishing moments, `tN` is number of primal vanishing moments, `depth` gives the levels upto which the wavelet transform is to be performed and `xc` is new grid. The main commands used in this function are the following:

```
N=length(x);
index=1:length(x);
[C,L,Wavelet_transform_matrix, Wavelet_transform_index] =
Wavelet_transform(u,N1,tN,depth,index);
d=zeros(size(x));
lx=length(x);
for tmp1=1:depth
    for i=ceil(lx/2)+1:lx
```

```
        d(Wavelet_transform_index(tmp1+1,i))=Wavelet_transform
        _matrix(tmp1+1,i);
    end
    lx=ceil(lx/2);
end
xc=x;
for i=1:N
    if abs(d(i))>=epsilon;
        len_int= (x(i)-x(i-1))/2;
        dis= len_int/(n+1);
        a= zeros(1,2*n);
        for j=1:n
            a(j)=x(i)-(j-(n+1))*dis;
        end
        for j=n+1:2*n
            a(j)=x(i)-(j-n)*dis;
        end
        a1=setdiff(a,x);
        size(xc)
        size(a1)
        xc=[xc; a1'];
    end
end
    if abs(d(N))>=epsilon;
         len_int= (x(N)-x(N-1))/2;
    dis= len_int/(n+1);
    a= zeros(1,n);
    for j=1:n
        a(j)=x(N)-(j)*dis;
    end
    a1=setdiff(a,x);
    xc=[xc; a1'];
    end
xc=sort(xc);
points = zeros(size(xc));
figure;
plot(x,u,xc,points,'or','linewidth',2);
```

10. `AdaptiveGrid_SGW_standard.m` (dependency: `Wavelet_transform.m`).

The function `AdaptiveGrid_SGW_standard.m` constructs an adaptive grid for a function by using the standard adaptation technique. The calling command for this function is

```
>>[x_new]=AdaptiveGrid_SGW_standard(x,u,epsilon,N1,tN,depth)
```

where `u` is the function value, `x` is the initial grid, `epsilon` is threshold value, `N1` is number of dual vanishing moments, `tN` is number of primal vanishing moments, `depth` gives the levels upto which the wavelet transform is to be performed and `x_new` is new grid. The main commands used in this function are the following:

```
N= length(x);
index=1:length(x);
[C,L,Wavelet_transform_matrix, Wavelet_transform_index] =
Wavelet_transform(u,N1,tN,depth,index);
d=zeros(size(x));
lx=length(x);
for tmp1=1:depth
    for i=ceil(lx/2)+1:lx
        d(Wavelet_transform_index(tmp1+1,i))=Wavelet_transform
        _matrix(tmp1+1,i);
    end
    lx=ceil(lx/2);
end
y= zeros(1,N);
for i=1:N
    if(abs(d(i))>=epsilon)
        y(i)=1;
    end
end
x_new=[x(1)];
for i=2:length(x)-1
    if y(i)==1
        x_new=[x_new;x(i)];
    end
end
```

```
x_new=[x_new;x(end)];
points = zeros(size(x_new));
figure;
plot(x,u,x_new,points,'or','linewidth',2);
```

11. `Reconstruction_testing.m` (dependency: `Wavelet_transform.m, Wavelet_transform_inverse.m`).

    The function `Reconstruction_testing.m` tests the reconstruction error for the second generation wavelet. It gives the error when the function f is reconstructed using second generation wavelet transform. It also plots the original given function f in blue and the reconstructed function in red. The calling command for this function is

    ```
    >>[err]=Reconstruction_testing(num)
    ```

    where `num` takes the values 1, 2 and 3. `num=1` means the testing is done for sawtooth function, `num=2` means the testing is done for sine function with a discontinuity, `num=3` means the testing is done for $f(x) = -tanh((x + x0)/(2 * nu)) + exp(-64^2 * (x - x0).^2); x0 = 1/3; nu = 10^{-3}$ and `err` gives the reconstruction error. The main commands used in this function are the following:

```
N=2;
tN=2;
depth=5;
if num==1
  x=linspace(0,1,100);
  u=zeros(size(x));
  for i=1:length(x)
    if (x(i)<0.8)
        u(i)=x(i);
    else
        u(i)=x(i)-1;
    end
  end
  [err]=Reconstruction(x,u,N,tN,depth);
elseif num==2
     x=linspace(0,1,100);
     u=zeros(size(x));
```

```
     for i=1:length(x)
         u(i)=sin(2*pi*x(i))+exp(-10^4*(x(i)-0.5)^2);
     end
     [err]=Reconstruction(x,u,N,tN,depth);
elseif num==3
    x=linspace(0,1,100);
    u=zeros(size(x));
    x0=1/3;
    nu=10^-3;
    for i=1:length(x)
        u(i)=-tanh((x(i)+x0)/(2*nu))+exp(-64^2*(x(i)-x0).^2);
    end
    [err]=Reconstruction(x,u,N,tN,depth);
else
    error('permissible values of num are 1,2 and 3' );
end
```

12. `AdaptiveGrid_modified_testing.m` (dependency: `Wavelet_transform.m`).

    The function `AdaptiveGrid_modified_testing.m` plots the given function f and the corresponding adaptive grid using modified adaptation technique. The implementation goes as follows:

```
N1=2;
tN=2;
depth=2;
n=2;
if num==1
  x=linspace(0,1,70)';
  u=zeros(size(x));
  epsilon=0.7;
  for i=1:length(x)
    if (x(i)<0.8)
        u(i)=x(i);
    else
        u(i)=x(i)-1;
    end
  end
```

```
  [xc] = AdaptiveGrid_SGW_modified(x,u,n,epsilon,N1,tN,depth);
elseif num==2
     x=linspace(0,1,70)';
     u=zeros(size(x));
     epsilon=0.7;
     for i=1:length(x)
         u(i)=sin(2*pi*x(i))+exp(-10^4*(x(i)-0.5)^2);
     end
     [xc] = AdaptiveGrid_SGW_modified(x,u,n,epsilon,N1,tN,depth);
elseif num==3
    x=linspace(-1,1,70)';
    u=zeros(size(x));
    epsilon=0.9;
    x0=1/3;
    nu=10^-3;
    for i=1:length(x)
        u(i)=-tanh((x(i)+x0)/(2*nu))+exp(-64^2*(x(i)-x0).^2);
    end
    [xc] = AdaptiveGrid_SGW_modified(x,u,n,epsilon,N1,tN,depth);
    ylim([-2,2]);
else
    error('permissible values of num are 1,2 and 3' );
end
```

---

13. `AdaptiveGrid_standard_testing.m` (dependency: `Wavelet_transform.m`).

    The function `AdaptiveGrid_SGW_standard.m` plots the given function f and the corresponding adaptive grid using standard adaptation technique. The implementation goes as follows:

```
N1=2;
tN=2;
depth=2;
if num==1
  x=linspace(0,1,100)';
  u=zeros(size(x));
  epsilon=0.2;
```

```
  for i=1:length(x)
    if (x(i)<0.8)
        u(i)=x(i);
    else
        u(i)=x(i)-1;
    end
  end
  [x_new] = AdaptiveGrid_SGW_standard(x,u,epsilon,N1,tN,depth);
elseif num==2
     x=linspace(0,1,100)';
     u=zeros(size(x));
     epsilon=0.7;
     for i=1:length(x)
         u(i)=sin(2*pi*x(i))+exp(-10^4*(x(i)-0.5)^2);
     end
     [x_new] = AdaptiveGrid_SGW_standard(x,u,epsilon,N1,tN,depth);
elseif num==3
    x=linspace(-1,1,100)';
    u=zeros(size(x));
    epsilon=0.75;
    x0=1/3;
    nu=10^-3;
    for i=1:length(x)
        u(i)=-tanh((x(i)+x0)/(2*nu))+exp(-64^2*(x(i)-x0).^2);
    end
    [x_new] = AdaptiveGrid_SGW_standard(x,u,epsilon,N1,tN,depth);
    ylim([-2,2]);
else
    error('permissible values of num are 1,2 and 3' );
end
```

Bio-Data

Full Name : Deepika Sharma

Education

- Ph.D.
  - Joined in January 2016.
  - Research Topic: Wavelet and its variants based numerical methods for Partial Differential Equations
  - Courses undertaken during course work:
    Wavelets and applications (PMC319)
    Research methodology (DMC007)
  - Passed the Pre-Ph.D. course work with C.G.P.A. of 9.67 on a 10 point scale.
- M.Sc. (Hons. School) - Mathematics
  Passed in July, 2014 with 8.71 C.G.P.A. from Department of Mathematics, Guru Nanak Dev University, Amritsar, India.
- B.Sc. (Non Medical)
  Passed in July, 2012 with 69% marks from S. R. Government College for Women, Amritsar, India.
- Higher Secondary
  Passed in year 2009 with 75% marks from S. D. S. Pheruman Memorial Collegiate Girls Senior Secondary School, Amritsar, India.
- High School
  Passed in year 2007 with 84% marks from Jyoti Model High School, Amritsar, India.

Conferences and Workshops Attended

- 6th International Conference on Soft Computing for Problem Solving SocProS 2016, Thapar Institute of Engineering and Technology, Patiala, December, 2016.
- Workshop on Advanced Computational Techniques for Differential Equations with Matlab (ACTDEM 2018) organized by Department of Mathematics, IIT Roorkee, September, 2018.
- Science Academics Lecture Workshop on Essence of Partial Differential Equation organized by School of Basic Sciences, IIT Mandi, H. P., April, 2018.

- Workshop on Applications of Optimization Techniques organized by Thapar Mathematical Society (TMS) in collaboration with Scimatics, TIET, Patiala, March, 2018.

Teaching Experience

- Teaching Assistant at TIET for the following courses:
  - UMA007 (Numerical Analysis).
  - UMA031 (Optimization Techniques).

Honours

- Qualified GATE with 201 All India Rank and having 98 percentile.
- Awarded SRF from CSIR April-2019.
- Best student award in chemistry during graduation.
- Certificate of appreciation from Khalsa College, Amritsar for organizing a debate/declamation contest.