ADAPTIVE MESH REFINEMENT FOR ONE DIMENSIONAL SCALAR CONSERVATION LAWS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KADIR ÇAR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
MATHEMATICS

SEPTEMBER 2024

Approval of the thesis:

ADAPTIVE MESH REFINEMENT FOR ONE DIMENSIONAL SCALAR CONSERVATION LAWS

submitted by **KADIR ÇAR** in partial fulfillment of the requirements for the degree of **Master of Science in Mathematics Department, Middle East Technical University** by,

Date:06.09.2024

ABSTRACT

ADAPTIVE MESH REFINEMENT FOR ONE DIMENSIONAL SCALAR CONSERVATION LAWS

Çar, Kadir
M.S., Department of Mathematics
Supervisor: Assoc. Prof. Dr. Baver Okutmuştur

September 2024, 69 pages

This thesis examines the numerical solutions of one dimensional scalar conservation laws on non-uniform grids by considering an Adaptive Mesh Refinement (AMR) based on an algorithm proposed by Berger and Colella. Finite volume method is used for discretization of the test equations with the CLAWPACK software. Three basic equations in one dimension are of particular interest; linear advection equation, inviscid Burgers equation, and Buckley-Leverett equation. Propagation of shock and rarefaction waves are investigated and compared for both standard uniform mesh and AMR at different time levels with smooth and piecewise smooth initially given functions. The findings demonstrate that results by AMR offer superior efficiency by focusing computational resources where needed, reducing overall costs while preserving accuracy. This comparison underscores the advantages of adaptive strategies in managing sharp gradients and complex wave propagation in one dimensional scalar conservation laws.

Keywords: Scalar Conservation law, linear advection equation, inviscid Burgers equation, Buckley-Leverett equation, Adaptive Mesh Refinement

ÖZ

TEK BOYUTLU SKALER KORUNUM YASALARI İÇİN UYARLANABİLİR ÖRGÜ İYİLEŞTİRME

Çar, Kadir
Yüksek Lisans, Matematik Bölümü
Tez Yöneticisi: Doç. Dr. Baver Okutmuştur

Eylül 2024 , 69 sayfa

Bu tez algoritması Berger ve Colella tarafından tasarlanan Uyarlanabilir Örgü İyileştirme yöntemini dikkate alarak tek boyutlu skaler korunum yasalarının düzensiz ağlar üzerindeki sayısal çözümlerini incelemektedir. Test denklemlerinin ayrıklaştırılması için sonlu hacim yöntemi ile birlikte CLAWPACK yazılımı kullanılmıştır. Doğrusal taşıma, viskozitesiz Burgers ve Buckley-Leverett denklemleri tek boyutlu üç temel denklem olarak ele alınmış ve bu denklemler üzerine odaklanılmıştır. Şok ve seyrelme dalgalarının yayılımını incelemek için yapılan sayısal hesaplamalarda standart düzgün ızgara ile Uyarlanabilir Örgü İyileştirme yöntemleri karşılaştırılmıştır. Bu incelemeler farklı zaman seviyelerinde başlangıç koşulu olarak düzgün ve parçalı düzgün verilen fonksiyonlar için yapılmıştır. Uyarlanabilir Örgü İyileştirme yönteminin hesaplama kaynaklarını gerektiği yerde yoğunlaştırarak maliyetleri düşürdüğü ve doğruluğu koruduğu sonuçlar tarafından gösterilmektedir. Ayrıca, bu bulgular tek boyutlu skaler korunum yasalarındaki keskin gradyanlar ve karmaşık dalga yayılımını yönetmede adaptif stratejilerin sağladığı avantajları da ortaya koymaktadır.

Anahtar Kelimeler: Skaler korunum yasası, doğrusal ilerleme denklemi, viskozitesiz Burgers denklemi, Buckley-Leverett Denklemi, Uyarlanmış Örgü İyileştirme

To my family

ACKNOWLEDGMENTS

First and foremost, I express my deepest gratitude to my supervisor, Baver Okutmuştur, for his guidance and support throughout the thesis period.

I would like to extend my sincere thanks to the committee members, Prof. Dr. Fikriye Nuray Yılmaz and Assoc. Prof. Kostyantyn Zheltukhin, for their helpful comments and patience.

My heartfelt appreciation goes to Şakire Solmaz Acar and Fatma Sevim Aygar for their unwavering support and encouragement which was greatly appreciated during this journey.

I am deeply grateful to my brother and mother for their constant love, support and understanding.

I am also thankful to Prof. Dr. İsmail Uğur Tiryaki for his support, ideas, and direction throughout both my undergraduate and graduate studies.

Additionally, I would like to thank Öznur Özcan for her continuous support and love.

Lastly, I am grateful to TÜBİTAK for awarding me the scholarship 2210-A, which significantly supported my study.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF FIGURES xiii

LIST OF ABBREVIATIONS xv

CHAPTERS

1 INTRODUCTION 1

1.1 Thesis Overview 5

2 FINITE VOLUME METHOD FOR SCALAR CONSERVATION LAWS . . 7

2.1 Conservation Laws 7

2.1.1 Scalar Conservation Laws 8

2.1.2 Examples of 1D Scalar Conservation Laws 9

2.2 Finite Volume Method 10

2.2.1 Finite Volume Method for 1D Scalar Conservation Laws . . . 11

2.3 Riemann Problem 14

2.4 CFL Condition 15

2.5 Some Examples of Numerical Flux Functions 15

3 ADAPTIVE MESH REFINEMENT 17

3.1 Introduction to AMR 17

3.2 Description of the Algorithm of AMR 21

3.3 Description of the Grid 24

3.4 Algorithm of Integration 29

3.5 Estimation of Error 30

3.6 Conditions of Interface 31

3.7 Structures of Data 33

4 NUMERICAL IMPLEMENTATION OF SCALAR CONSERVATION LAWS BY AMR 35

4.1 Numerical Results of Scalar Conservation Laws 35

4.1.1 Test Equation 1: Linear advection equation 36

4.1.2 Test Equation 2: Inviscid Burgers Equation 40

4.1.3 Test Equation 3: Buckley-Leverett equation 55

5 CONCLUSION 63

REFERENCES 65

LIST OF FIGURES

FIGURES

Figure 2.1 Grid Scheme of the Finite Volume 11

Figure 2.2 Finite volume method for updating the cell averages by fluxes in 1D 14

Figure 3.1 Evolution of the step function over time 21

Figure 3.2 Evolution of the solution from a smooth initial condition to an N-wave structure 22

Figure 3.3 Comparison between uniform mesh and adaptive mesh 23

Figure 3.4 Coarse grid with a refined subgrid 25

Figure 3.5 Illusturation of 1D coarse and refined grids 25

Figure 3.6 Levels of refinement in a 1D grid 26

Figure 3.7 Separate identical level refinements 28

Figure 3.8 Merged Identical Level Refinement 28

Figure 3.9 Nested grids example with different levels of refinement 30

Figure 3.10 One dimensional regridding procedure 33

Figure 3.11 Structure of data for 1D 34

Figure 4.1 Numerical solution of linear advection with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.1) and $CFL = 0.9$. 37

Figure 4.2 Comparison of uniform mesh and AMR for linear advection equation. Initial condition (4.1) with $N = 60$ and CFL$= 0.9$. 39

Figure 4.3 Numerical solution of inviscid Burgers equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.2) and CFL$= 0.9$ . . 41

Figure 4.4 Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.2) with $N = 20$ and CFL$= 0.9$. 44

Figure 4.5 Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.2) with $N = 60$ and CFL$= 0.9$. 45

Figure 4.6 Numerical solution of inviscid Burgers equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.3) and CFL$= 0.9$. . . 46

Figure 4.7 Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.3) with $N = 60$ and CFL$= 0.9$. 48

Figure 4.8 Numerical solution of inviscid Burgers equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.4) and CFL$= 0.9$. . . 51

Figure 4.9 Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.4) with $N = 60$ and CFL$= 0.9$. 53

Figure 4.10 Evolution of numerical solutions by AMR for inviscid Burgers equation with $N = 60$. Initial condition (4.4) and CFL$= 0.9$. 54

Figure 4.11 Numerical solution of Buckley-Leverett equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.5) and CFL$= 0.9$. . . 56

Figure 4.12 Comparison of uniform mesh and AMR for Buckley-Leverett equation. Initial condition (4.5) with $N = 60$ and CFL$= 0.9$. 58

Figure 4.13 Evolution of numerical solutions by AMR for Buckley-Leverett equation with $N = 60$. Initial condition (4.5) and CFL$= 0.9$. 59

LIST OF ABBREVIATIONS

1D	1 Dimensional
AMR	Adaptive Mesh Refinement
AMM	Adaptive Mesh Method
MMM	Moving Mesh Method
FDM	Finite Difference Method
FEM	Finite Element Method
FVM	Finite Volume Method
CMAM	Coarse Mesh Approximation Method
PDE	Partial Differential Equation

CHAPTER 1

INTRODUCTION

Partial differential equations (PDEs) are utilized in various parts of daily life contributing to the understanding modelling and optimization of numerous processes and systems. They are applied on many areas as like physics, engineering, mathematics and their applications on the daily life [30, 41]. These equations are vital for describing fluid flow phenomena and are applied in areas like water flow in pipes, aerodynamics of vehicles and aircraft, and simulations of heat transfer processes including heating, cooling, and insulation systems that engineers handle [41, 18].

These equations are classified into three main types: elliptic, parabolic and hyperbolic PDEs. This classification is based on the nature of the characteristics of the equations and the behavior of their solutions [13].

Elliptic types are equations where the characteristic curves are complex, and they do not propagate waves or signals. These equations often describe steady-state or equilibrium situations, where the solution is smooth if the boundary conditions are smooth.

Parabolic PDEs describe processes that evolve over time and tend to a steady state. They have one real characteristic direction and typically model diffusion-like processes where the solution smooths out over time.

Hyperbolic PDEs are characterized by real and distinct characteristic curves, modeling wave propagation and other phenomena where features such as saturation fronts, shock fronts, and discontinuities are transferred along these characteristics. Hyperbolic equations can develop discontinuities such as shock waves even if the initial conditions are smooth. To explore these phenomena in detail, we examine how such discontinuities arise and evolve in different scenarios. In this thesis, we work on three examples of hyperbolic PDEs in 1D:

- Advection equation which describes the transport of a substance or quantity by a fluid flow.
- Inviscid Burgers equation which is a fundamental model for shock waves and turbulence
- Buckley-Leverett equation which models two-phase flow in porous media.

Although all types of PDEs are crucial for modeling various physical and engineering phenomena, solving them can be quite challenging. Their complexity with multiple variables and partial derivatives makes finding exact solutions very difficult. As a result achieving these problems' exact solutions isn't always possible. As the problem gets more complex or involves more dimensions, finding a precise answer becomes even harder. To address these challenges, numerical methods approximate solutions by breaking the problem into smaller, manageable pieces. This process turns the continuous PDE into a set of discrete equations. These equations can then be solved using computational techniques, making it possible to find practical solutions even when exact answers are not feasible. Several numerical methods such a Finite Element Method (FEM), Finite Difference Method (FDM) and Finite Volume Method (FVM) are utilized.

FEM works by dividing the problem domain into smaller pieces called elements. Each element is represented by simple shapes as triangles or rectangles. The method approximates the solution within each element using piecewise functions. This approach is typically applied to solve elliptic types of PDEs.

FDM divides the problem domain into a grid of points and approximates the derivatives using differences between the function values at these points. This method replaces the continuous derivatives with discrete approximations based on the spacing of the grid points. By applying these approximations across the grid, FDM transforms the equation into a system of algebraic equations which are then solved to find the approximate solution.

FVM which is carried out in this thesis, approaches solving partial differential equations by dividing the whole domain of the problem into discrete cells or control volumes. The method ensures that conserved quantities such as energy and mass are preserved in each control volume by integrating the PDE. The conservation laws are applied within each control volume [48, 47]. Integration over the volume ensures that the total conserved quantity remains constant. This integration process converts the equation into a flux balance across the boundaries of each control volume. By approximating these fluxes using values from neighboring control volumes, FVM produces a set of algebraic equations. Solving these equations provides a numerical approximation of the solution [46]. FVM is highly efficient for hyperbolic type of PDEs. The method handles sharp gradients and shock formations well due to its ability to manage these features effectively [48, 47, 26].

Despite these advantages, FEM, FDM, and FVM typically rely on grids to discretize the entire computational domain. For problems involving sudden changes in the solution, finer grids are required to accurately capture the solution features. This necessity increases both computational cost and time. This problem arises with high demand of memory and computational power which we avoid. To address this challenge, AMR is used with these stated methods. AMR adapts the grid resolution dynamically, making it finer where needed and coarser where possible. AMR can be used with the FEM, FDM or FVM to provide more efficient solutions by dynamically adjusting the mesh to capture important features with higher resolution where needed.

In response to this approach, adaptive mesh methods have gained increasing significance in recent decade years. These techniques adjust the mesh structure during computation, strategically concentrating mesh points where they are most needed. These approaches optimize computational resources and improves solution accuracy, which are particularly beneficial for complex problems like phase change, blow-up problems as studied by Russell, Huang and Budd for moving mesh methods [8]. For hyperbolic conservation laws, significant contributions have been made by Marsha Berger [5] as well as Huazhong Tang and Tao Tang [22]. Thus, all these methods including adaptive mesh techniques have proven to be highly efficient, yielding better performance and accuracy in solving PDEs. Adaptive mesh methods have been widely applied in solving these equations, addressing various challenges through specialized mesh methods. These methods are generally categorized into three types [44], each designed to improve the accuracy and efficiency of the numerical solutions.

The first type is known as h-refinement which improves accuracy by subdividing elements to refine the mesh, allowing for the addition or removal of mesh points to adapt to varying solution needs. This method dynamically adjusts based on error estimates by adding points in regions with high variation or error and removing them where the solution is smooth, enhancing spatial resolution. Barrett [3], Sun [42], Li [29], Shengtai Li and Mac Hyman [28], Shen and Qui [39] have successfully applied this method to FEM and FVM. Berger-Oliger [5] AMR technique is one of the pioneering h-method application.

The second one is p-refinement involves adaptively varying the polynomial order based on local error estimates or indicators. This technique improves accuracy by adjusting the polynomial degree in different regions to better capture solution smoothness. Babuška and Suri [2] have applied this approach.

The last one is r-refinement which is also known as the moving mesh method. This technique adjusts grid points within a fixed-node mesh to ensure nodes are concentrated in regions with rapid solution variation. This approach maintains a constant number of nodes while adjusting their positions to improve resolution where needed. Miller [32] developed the moving FEM, and Tao Tang [44] applied r-refinement in computational fluid dynamics.

Additionally, some works combine both r and h, or h and p refinements. For instance Babushka [2] has explored the integration of h-p refinement versions in FEM, providing insights into how these techniques can be used together. Similarly the h-r moving mesh method has been studied by Ong, Russell, and Ruuth for 1D time-dependent PDEs [34]. Additionally Piggott has worked on h-r adaptivity in numerical ocean modeling [36].

In this thesis, we focus on one-dimensional scalar conservation laws. These equations are typically solved using FVM. FVM is particularly well-suited for these type of equations as it conserves quantities at the control volume level and effectively handles discontinuities and sharp gradients through its flux-based approach.

In order to explore the effectiveness of FVM along with AMR, we examine three fundamental one-dimensional hyperbolic equations: advection equation, inviscid Burgers equation, and Buckley-Leverett equation. To solve these equations, we utilize both classical fixed uniform meshes and AMR using the FVM. The comparison of solutions obtained by using fewer and increased numbers of uniform meshes and the results of AMR obtained with the increased number of uniform meshes is performed using CLAWPACK .

Adaptive mesh techniques facilitate a more targeted allocation of computational resources, which is particularly beneficial for regions with rapid changes, such as shock waves. This comparison aims to highlight how adaptive meshing improves the capture of sharp gradients and discontinuities while optimizing overall computational efficiency.

1.1 Thesis Overview

This thesis consists of five chapters, including the introduction. In Chapter 2, we start with an overview of conservation laws and proceed to a detailed discussion of scalar conservation laws in one dimension, addressing both their integral and differential forms. In this chapter, we present 1D scalar conservation laws and focus on three fundamental test equations for these laws: the linear advection equation, the inviscid Burgers equation, and the Buckley-Leverett equation.

In Chapter 2 we present the FVM and outline the necessary conditions for its effective application. Moving to Chapter 3, we provide a comprehensive background on AMR and analyze the applications and detailed descriptions of the AMR algorithm implemented in the CLAWPACK tool, with a focus on the Berger-Oliger method. In Chapter 4, we compare the numerical results of the uniform and AMR for 1D-scalar test equations, using the FVM with the Godunov type numerical flux functions. Finally the Chapter 5 provides a summary of the thesis.

CHAPTER 2

FINITE VOLUME METHOD FOR SCALAR CONSERVATION LAWS

In this chapter, our primary focus is on the FVM as applied to scalar conservation laws. We start by providing a comprehensive introduction to the concept of conservation laws, including their fundamental principles and significance. Following this, we delve into the general form of scalar conservation laws, setting the stage for a deeper understanding. We then proceed to give three specific examples of scalar conservation laws that are especially relevant to our study. Following this, we introduce the general form of the finite volume scheme, detailing its methodology and how it addresses these laws. Finally, we introduce the three types of well known flux functions that is mostly used within FVM.

2.1 Conservation Laws

Conservation laws are foundational principles in physics and engineering that govern the preservation and redistribution of physical quantities within closed systems [13]. These laws assert that certain quantities such as mass, energy and momentum, remain constant over time within isolated systems, reflecting fundamental symmetries and invariances in nature [26, 18]. Conservation laws are typically expressed as PDEs, representing the balance between the rate of change of a conserved quantity and the flux of that quantity through the boundaries of a spatial region.

A general form of a conservation law is given by

$$\frac{\partial u}{\partial t} + \nabla f = 0 \tag{2.1}$$

where the $f : U \longrightarrow \mathbb{R}$ is the known **flux function** which prescribes the rate of flow for each conserved variable and x_i and t are the spatial and time coordinates respectively and U is the open subset of $\mathbb{R}^n$ where $u : R \times R \longrightarrow \mathbb{R}^n$ is the n-dimensional vector of conserved quantities. The mathematical equation of the following form formulating variation of the conserved quantity over time.

$$\nabla f = \frac{\partial f_1}{\partial x_1} + \frac{\partial f_2}{\partial x_2} + \frac{\partial f_3}{\partial x_3} + ... + \frac{\partial f_n}{\partial x_n}$$

In one spatial dimension the flux function simplifies to the form:

$$\nabla f = \frac{\partial f}{\partial x} \tag{2.2}$$

The conservation law in one spatial dimension then becomes

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

2.1.1 Scalar Conservation Laws

Scalar conservation laws are a simplified subset of conservation laws describing the evolution of a single conserved quantity as like $u(x, t)$. These equations are fundamental to understanding wave phenomena, including propagation and shock formation. Their simplified structure compared to systems with multiple conserved variables enables focused analysis and the development of effective numerical solution techniques.

In this thesis we are dealing with 1D hyperbolic PDEs. In 1D space scalar conservation laws are of the form

$$u_t + f(u)_x = 0 \tag{2.3}$$

where $u(x, t)$ is the conserved quantity, t represents time which is called temporal coordinate, x is the spatial coordinate and $f(u)$ is the flux function that represents the rate of displacement of the conserved quantity. Equation 2.3 represents the differential form of the scalar conservation law [14, 48, 33].

In 1D case for any interval (x_1, x_2), the change of the conserved quantity $u(x, t)$ with respect to x equals to the displaced conserved quantity which is the flux. As the integral form arises by

$$\frac{d}{dt}\int_{x_1}^{x_2} u(x, t)dx = \int_{x_1}^{x_2} \frac{d}{dt}u(x, t)dx = \int_{x_1}^{x_2} u_t(x, t)dx, \tag{2.4}$$

from the equation (2.3) it can easily be observed that

$$u_t = -f(u(x,t))_x.$$

Using the above expression of u_t and switching it by the flux in the equation (2.4), it follows that

$$\begin{aligned}
\frac{d}{dt}\int_{x_1}^{x_2} u(x,t)dx &= \int_{x_1}^{x_2} u_t(x,t)\,dx \\
&= \int_{x_1}^{x_2} -f(u(x,t))_x\,dx \\
&= -\int_{x_1}^{x_2} f(u(x,t))_x\,dx \\
&= f(u(x_1,t)) - f(u(x_2,t)) \\
&= [\text{Inflow at the point } x_1] - [\text{Outflow at the point } x_2]
\end{aligned} \tag{2.5}$$

Here $u(x,t)$ represents the quantity that is conserved in the interval, either transformed or balanced but neither produced nor destroyed [25].

2.1.2 Examples of 1D Scalar Conservation Laws

The general form of 1D conservation laws is detailed in the previous subsection. In this subsection, we provide the examples of scalar conservation laws that are analyzed in this thesis. These equations express the conservation of $u(x,t)$ over time t and space in one dimension x. The examples that are considered include :

- Linear advection equation : It is described by

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(cu) = 0 \tag{2.6}$$

  which is a simpler form of the Burgers' equation without the nonlinear term. $u(x,t)$ is the quantity being advected, c is the advection speed, $\frac{\partial u}{\partial t}$ represents the rate of change of u with respect to time, $\frac{\partial}{\partial x}(cu)$ represents the flux of u denoted as $f(u)$ in the x direction [26].

- Inviscid Burgers equation : Introduced by J.M. Burgers [9], the inviscid Burgers equation described by

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0 \tag{2.7}$$

which serves as a fundamental model for understanding the behavior of nonlinear waves, particularly in 1D fluid flows and nonlinear acoustics [33]. $\dfrac{\partial u}{\partial t}$ represents the rate of change of scalar quantity u with respect to time. The term $f(u) = \dfrac{u^2}{2}$ is the flux of the scalar quantity u and the term $\dfrac{\partial}{\partial x}\left(\dfrac{u^2}{2}\right) = u\dfrac{\partial u}{\partial x}$ represents how the flux changes in spatial x direction [26].

- Buckley-Leverett equation : Buckley-Leverett equation is introduced in 1942 by Buckley and Leverett [7] and described by

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{u^2 + a(1-u)^2}\right) = 0, \quad a < 1 \tag{2.8}$$

  This equation serves as a fundamental model for describing one-dimensional immiscible fluid displacement in porous media, particularly in the context of oil recovery processes [35]. It describes the displacement of one fluid by another like water displacing oil in a porous medium where $u(x,t)$ denotes the saturation of one of the fluids, such as the water saturation, which represents the fraction of the pore volume occupied by that fluid. The term

$$f(u) = \left(\frac{u^2}{u^2 + a(1-u)^2}\right)$$

  is the flux function, a nonlinear function of u, which describes the proportion of the total fluid flow that is the particular fluid of interest like water. The term $\dfrac{\partial u}{\partial t}$ represents the rate of change of fluid saturation with respect to time t. The second term, $\dfrac{\partial f(u)}{\partial x}$ represents the flux of the fluid saturation in the spatial direction x. This term describes how the fluid saturation is transported along the spatial dimension due to the flow dynamics within the porous medium. The a is referred as mobility ratio parameter.

2.2 Finite Volume Method

Finite Volume Method (FVM) is a numerical technique for discretizing partial differential equations by partitioning the computational domain into a series of control volumes[26, 46]. It calculates fluxes across the boundaries of these volumes to approximate the integral representation of the equations. Particularly advantageous for hyperbolic PDEs due to its ability to preserve local conservation properties [26], FVM is commonly utilized in scenarios involving scalar hyperbolic PDEs. While this method can be extended to higher dimensions, for this thesis, we specifically restrict its application to 1D scalar PDEs.

2.2.1 Finite Volume Method for 1D Scalar Conservation Laws

In 1D space FVM is constructed by dividing the spatial domain x into smaller segments referred to as finite volumes [26]. The spatial domain is divided into arbitrary grids and the discretized points denoted as $x_i, i \in \mathbb{Z}$ and bounded by some positive integer m such that $x_i : i = 1, 2, \dots, m$ as the spatial domain is divided into m subintervals.

$$\Delta x = \sup_{x_i \in \mathcal{C}_{\Delta x}} |x_i - x_{i-1}|$$

Without loss of generality, we assume that the grids are uniform for ease,
$\Delta x = x_i - x_{i-1}$, then i^{th} grid cell are denoted by

$$C_i = \left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right) = \left(x_i - \frac{x_i - x_{i-1}}{2}, x_i + \frac{x_{i+1} - x_i}{2}\right)$$

and can be seen as displayed in the Figure 2.1

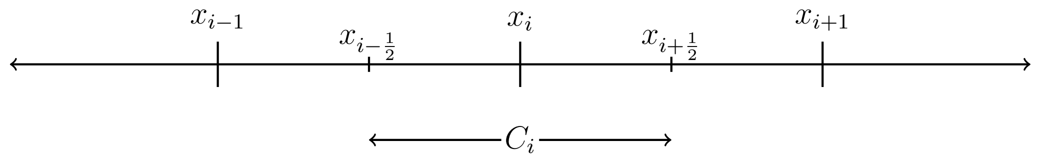


Figure 2.1: Grid Scheme of the Finite Volume

From the equation 2.3, it follows that the integral form of the conservation law on each cell $C_i = (x_{i-1/2}, x_{i+1/2})$ takes the form

$$\frac{1}{dt}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t)\,dx = \frac{1}{dt}\int_{C_i} u(x,t)\,dx \; = f(u(x_{i-1/2},t)) - f(u(x_{i+1/2},t)) \tag{2.9}$$

Integrating (2.9) with respect to time from $t = t^n$ to the upper bound $t = t^{n+1}$, it follows that

$$\begin{aligned}\int_{C_i} u(x,t^{n+1})\,dx - &\int_{C_i} u(x,t^n)\,dx \\ &= \int_{t^n}^{t^{n+1}} f(u(x_{i-\frac{1}{2}},t))\,dt - \int_{t^n}^{t^{n+1}} f(u(x_{i+\frac{1}{2}},t))\,dt\end{aligned} \tag{2.10}$$

Next, dividing each component by $\Delta x = x_i - x_{i-1}$ represents the distances between each consecutive nodes of the spatial domain x, and re-arranging the equation, we achieve average change on the spatial domain x

$$\begin{aligned}\frac{1}{\Delta x}\int_{C_i} u(x,t^{n+1})\,dx = \; &\frac{1}{\Delta x}\int_{C_i} u(x,t^n)\,dx \\ &- \frac{1}{\Delta x}\left(\int_{t^n}^{t^{n+1}} f(u(x_{i+\frac{1}{2}},t))\,dt - \int_{t^n}^{t^{n+1}} f(u(x_{i-\frac{1}{2}},t))\,dt\right)\end{aligned} \tag{2.11}$$

This explains precisely how to update the cell average of $u(x,t)$ in a single time step over the cell [24]. This approach involves maintaining an approximation to the integral of the unknown function u over each of these intervals, ensuring that the conservation laws are satisfied within each volume. To explain more simply, this method breaks the space into small sections and calculates the average value of u in each section. This helps ensure that the overall behavior of u follows the conservation law in each grid cell. Each time step involves updating these values by estimating the flux passing through the endpoints of the intervals. In other words this helps us keep track of how this quantity changes over time in each section of space which is divided by the grid cells. A problem arises when we try to integrate with respect to time on the right side of the equation (2.11). Since $u(x_i \pm \frac{1}{2}, t)$ undergoes changes over time along the cell boundary, we lack an exact solution to precisely calculate its evolution. To address this problem, we use numerical methods. By denoting the approximation of the fluxes with a numerical method at $x = x_{i-\frac{1}{2}}$ and $x = x_{i+\frac{1}{2}}$ on the right side of the equations, and by denoting the integral of the approximate values that include

$u(x,t)$, the numerical value $\mathcal{U}$ serves as an approximation of the average value within the i-th interval at time t^n, since we want to approximate to the value of U_i^{n+1} after a time step $\Delta t = t^{n+1} - t^n$, thus it is given by

$$\mathcal{U}_i^n \approx \frac{1}{\Delta x} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} u(x,t^n)\,dx = \frac{1}{\Delta x} \int_{C_i} u(x,t^n)\,dx.$$

Approximate flux values $\mathcal{F}_{i+\frac{1}{2}}$ and $\mathcal{F}_{i-\frac{1}{2}}$ are introduced respectively by

$$\mathcal{F}_{i+1/2}^n \approx \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} f(u(x_{i+1/2},t))\,dt \quad , \quad \mathcal{F}_{i-1/2}^n \approx \frac{1}{\Delta t} \int_{t^n}^{t^{n+1}} f(u(x_{i-1/2},t))\,dt. \tag{2.12}$$

After plugging the approximate values of the fluxes and the values of the unknown function, we have

$$\mathcal{U}_i^{n+1} = \mathcal{U}_i^n - \frac{\Delta t}{\Delta x}\left(\mathcal{F}_{i+1/2}^n - \mathcal{F}_{i-1/2}^n\right) \tag{2.13}$$

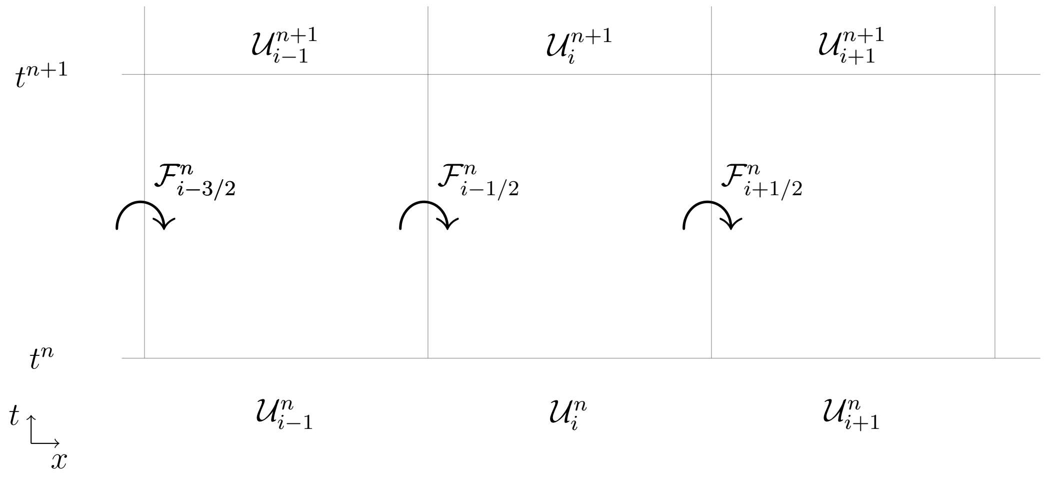
As the values that describe the state of the system in a hyperbolic problem propagate at a finite speed, it makes sense to initially assume that $\mathcal{F}_{i-1/2}^n$ can be achieved only from the cell values of $\mathcal{U}_{i-1}^n$ and $\mathcal{U}_i^n$ on the average values on both sides of the boundary $x_{i-1/2}$. We can use the form

$$\mathcal{F}_{i-1/2}^n = F(\mathcal{U}_{i-1}^n, \mathcal{U}_i^n) \tag{2.14}$$

where F is the numerical flux function. Notable 3 different numerical flux functions are given in detail in the subsequent section. Then, numerical scheme in 2.13 reads

$$\mathcal{U}_i^{n+1} = \mathcal{U}_i^n - \frac{\Delta t}{\Delta x}\left(F(\mathcal{U}_i^n, \mathcal{U}_i^{n+1}) - F(\mathcal{U}_{i-1}^n, \mathcal{U}_i^n)\right) \tag{2.15}$$

In Figure 2.2, we illustrate the Finite Volume Method (FVM) approach for updating the average values $\mathcal{U}_{i-1}^n$, $\mathcal{U}_i^n$, and $\mathcal{U}_{i+1}^n$ within each cell using fluxes at the cell boundaries. This figure, presented in x-t space, shows the transition from time t^n to t^{n+1}. It includes the fluxes $\mathcal{F}_{i-3/2}^n$, $\mathcal{F}_{i-1/2}^n$, and $\mathcal{F}_{i+1/2}^n$ applied at the cell interfaces. The figure illustrates how the flow of conserved quantities is handled across the cell boundaries.



$\mathcal{U}_{i-1}^{n+1}$
$\mathcal{U}_{i}^{n+1}$
$\mathcal{U}_{i+1}^{n+1}$
t^{n+1}
$\mathcal{F}_{i-3/2}^{n}$
$\mathcal{F}_{i-1/2}^{n}$
$\mathcal{F}_{i+1/2}^{n}$
t^{n}
$\mathcal{U}_{i-1}^{n}$
$\mathcal{U}_{i}^{n}$
$\mathcal{U}_{i+1}^{n}$
t
x

Figure 2.2: Finite volume method for updating the cell averages by fluxes in 1D

2.3 Riemann Problem

The Riemann problem is an initial value problem related to conservation laws, where a single discontinuity separates constant values at the boundaries. This problem is typically examined over a short time period surrounding the discontinuity, making it especially useful in multiphase flow studies involving two immiscible fluids with varying densities. The Riemann problem is instrumental in understanding conservation laws as it illustrates the formation of shock waves and rarefaction waves, which are key characteristics in such contexts. In the development of finite volume methods, solving the Riemann problem with defined initial conditions is a crucial technique. These initial conditions are piecewise constant with a single jump discontinuity at a point

$$u_0(x) = \begin{cases} u_L & \text{if } x < 0, \\ u_R & \text{if } x > 0 \end{cases}$$

where $u_L, u_R \in U$ are given constants. In the context of FVM, if U_{i-1} and U_i are the cell averages in two neighboring grid cells, solving the Riemann problem with $u_L = U_{i-1}$ and $u_R = U_i$ provides crucial information for computing numerical fluxes and updating cell averages over time.

In any numerical method as like FVM, convergence is crucial for ensuring that the numerical solution matches the true solution of the differential equation as the grid is refined. Convergence requires satisfying two essential conditions. First the method must be consistent with the differential equation and ensure an accurate local approximation of the true solution. Second the method must be stable which ensures that errors introduced at each time step are controlled and do not increase uncontrollably [26].

2.4 CFL Condition

For an explicit FVM to ensure stability and accuracy it must satisfy the CFL condition which was introduced by Courant, Friedrichs and Lewy [12]. A numerical finite volume scheme of the form (2.13) is called **stable** if the CFL condition described by

$$\max_j \left| f'(\mathcal{U}_j^n) \right| \frac{\Delta t}{\Delta x} \leq 1$$

is satisfied where $\left| f'(\mathcal{U}_j^n) \right|$ is the upper bound on the wave speeds that can occur in the Riemann problem.

2.5 Some Examples of Numerical Flux Functions

Numerical flux functions are essential in the FVM to compute the fluxes at the boundaries of each control volume. These fluxes approximate the flow of conserved quantities across the interfaces between neighboring control volumes. Some of the well-known numerical flux functions are as follows:

- **Godunov Flux**:

$$F_{\text{Godunov}}(u_L, u_R) = \begin{cases} \min\limits_{u_L < u < u_R} f(u), & \text{if } u_L \leq u_R, \\ \max\limits_{u_R < u < u_L} f(u), & \text{if } u_L > u_R. \end{cases}$$

  This flux function ensures conservation and is particularly effective in capturing shock waves and discontinuities [49].

- **Lax-Friedrichs Flux**:

$$F_{\text{Lax-Friedrichs}}(u_L, u_R) = \frac{1}{2} \left(f(u_L) + f(u_R) - \frac{\Delta x}{\Delta t}(u_R - u_L) \right).$$

It is diffusive and helps in maintaining stability in the numerical scheme.

- **Lax-Wendroff Flux**:

$$F_{\text{Lax-Wendroff}}(u_L, u_R) = \frac{f(u_L) + f(u_R)}{2} - \frac{\Delta t}{2\Delta x} \left[f'(\frac{u_R + u_L}{2}) \right] (u_R - u_L).$$

  It incorporates a Taylor series expansion to improve accuracy and stability, especially for problems with smooth solutions.

In order to ensure that the numerical flux is consistent, it should accurately approximate the integral in 2.12. When $u(x,t) \equiv \bar{u}$ is constant in the spatial dimension x, u does not vary over time and the integral simplifies to $f(\bar{u})$. Therefore if $\mathcal{U}_{i-1}^n = \mathcal{U}_i^n = \bar{u}$, the numerical flux function F of 2.14 should be equal to $f(\bar{u})$. This ensures that the numerical flux function F correctly represents the physical flux $f(\bar{u})$ when both $\mathcal{U}_{i-1}^n$ and $\mathcal{U}_i^n$ are $\bar{u}$.

In general, continuity is also required. This means that as $\mathcal{U}_{i-1}^n$ and $\mathcal{U}_i^n$ approach $\bar{u}$, the numerical flux F should approach $f(\bar{u})$. Additionally, Lipschitz continuity is typically necessary. This implies that there exists a constant M such that

$$|F(\mathcal{U}_{i-1}^n, \mathcal{U}_i^n) - f(\bar{u})| \leq M \max\left(|\mathcal{U}_i^n - \bar{u}|, |\mathcal{U}_{i-1}^n - \bar{u}|\right). \tag{2.16}$$

After stability and consistency are addressed, the numerical method is said to be convergent. In FVM, convergence ensures that as the grid is refined and the time step is decreased, the numerical solution approximates the true solution of the PDE with increasing accuracy. This means that the method provides reliable results as long as the conditions of consistency and stability are met. Additionally, convergence is crucial for validating the effectiveness of the FVM in practical applications, ensuring that the numerical results reflect the behavior of the physical system being modeled.

CHAPTER 3

ADAPTIVE MESH REFINEMENT

In this chapter we delve into the intricacies of AMR. The discussion begins with a broad overview of adaptive mesh techniques exploring their history and the various ways they can be classified. In 1D problems we introduce AMR, focusing on the Berger-Oliger type and its implementation in the CLAWPACK. This includes a discussion on the algorithm's key components and how it enhances computational efficiency and accuracy. Finally the grid structures used in this method are broken down detailing how grids are created refined and managed within the AMR framework to tackle problems effectively.

3.1 Introduction to AMR

In numerical solutions of PDEs , selecting the right mesh is crucial for obtaining accurate and efficient results. Finite Element, Finite Difference, and Finite Volume Methods often require extensive mesh adjustments to solve the problem accurately. Although these methods utilize meshes for their simplicity and straightforward implementation, this approach can become computationally intensive, time consuming and hard to store especially in multidimensional problems where the required number of mesh points can be large. Uniform meshes which distribute grid points evenly across the entire domain can be less efficient in terms of computational cost and memory. However adaptive meshes dynamically adjust the placement of grid points based on the complexity of the solution. This adaptability allows for greater resolution and efficiency as it concentrates computational resources where they are needed most to achieve accurate solutions [22].

AMM is particularly beneficial for problems involving sharp changes, discontinuities such as phase change phenomena, blow-up problems, and hyperbolic conservation laws [6, 22]. Berger and Colella implemented a class of AMM called AMR for shock hydrodynamics [6]. Building on this foundational work, Skamarock, Oliger, and Street later extended the application of AMR to numerical weather prediction using the Berger–Oliger technique [40]. This progression underscores the adaptability of AMM in tackling a wide range of complex problems. In recent decade years AMM have proven highly effective for addressing PDEs with substantial solution variations, including those involving shock waves, boundaries, and other complex features [19]. The effectiveness of AMM in its early uses led to more research and development, expanding its application to a wider variety of partial differential equations. This interest shows how adaptable AMM is and how it remains important for improving numerical simulations across various types of problems [37]. In addition to these, AMR is a distinct technique within the wider field of AMM designed to increase both the precision and efficiency of numerical simulations. While AMM covers a range of strategies for modifying the mesh based on the solution's behavior, AMR is particularly focused on hierarchical mesh refinement. Instead of relying on a uniform grid throughout the entire computational domain, AMR adjusts the mesh dynamically. It overlays finer grids on top of coarser ones specifically in areas where the solution shows sharp gradients, discontinuities or complex features. This method ensures that the computational resources are allocated more effectively, improving the accuracy of simulations in critical regions while maintaining overall efficiency. AMR technique has been developed by many researchers over the years. The approach initially developed by Berger, Oliger, and Colella was improved and expanded by others into different forms [38, 50]. The Berger-Oliger AMR was developed for Cartesian grids. After the researcher wanted to implement AMR to work with complex geometries by using finite volume method, they adapted the original approach by developing specialized grid structures to address diverse surface challenges. Chungang Chen, Feng Xiao, and Xingliang Li developed an adaptive global shallow-water model on a cubed-sphere grid by combining the finite volume scheme [10]. All in all in a broad classification, AMM can be categorized into three different types as follows:

h-Method: The h-method, named after the notation $\mathbf{h} = \mathbf{\Delta x}$, is commonly used in AMR. Method typically begins with a uniform mesh and adaptively refines or coarsens the mesh by adding or removing mesh points based on error estimates or indicators. This allows for improved local resolution by dynamically adjusting the spatial mesh. Notable examples include McCormick's work [31] which explores multiLevel adaptive methods, and Berger-Oliger-Corella's works [6] and [5], which detail based work in AMR. These references highlight the method's successful application across various problems and its significant impact on advancing adaptive mesh techniques.

p-method: The p-method involves the adaptive enrichment of the polynomial order. This method, which takes its name from polynomials, also means polynomial improvement. Finite element discretization of PDEs with p-methods is implemented with local polynomials of some degree. The degree of polynomials is usually measured by error estimates, increased or decreased to adapt the method according to the smoothness of the solutions. Numerous studies have implemented the p-method, including notable examples such as [17], [16], and [15], which provide a comprehensive analysis of p-version, h-version, and h-p version finite element methods, focusing on their application to problems with singularities and their effectiveness in improving accuracy through adaptive techniques. Additionally, [1] investigated the application of adaptive refinement techniques in solving parabolic PDEs, focusing on mesh refinement and polynomial degree adjustment to improve accuracy.

r-method: The r-method, also known as the moving mesh method, stands for relocation. It relocates grid points adaptively in a mesh with a fixed number of nodes to concentrate in regions with rapid solution variation. The mesh moves, allowing improvement and coarsening automatically without adding or removing grid points just by changing the position of existing meshes but keeping their number constant. This makes the r-method more economical and less computationally intensive compared to methods that add or remove grid points. Various implementations of the r-method include the work of Huang, Ren and Russell who explored the r-method for moving mesh PDEs using mesh relocation and equidistribution principles, developing efficient moving mesh algorithms for hyperbolic conservation laws and enhancing accuracy and efficiency for problems with shock discontinuities [21]. Additionally, Tang and Tao presented numerical methods for solutions with sharp transitions, focusing on Burgers' equation, demonstrating that nodes can effectively concentrate and move with shocks [43]. Furthermore, Miller also contributed by presenting numerical methods for solutions with sharp transitions, particularly focusing on Burgers equation, and showing that nodes can effectively concentrate and move with shocks [32].

In addition to this broad classification, combination of the mentioned methods as like r-h has been worked on by researchers. The r-h method is an adaptive technique that combines mesh movement with mesh refinement to enhance the accuracy and efficiency of numerical simulations. In this approach, the grid points are relocated to areas requiring higher resolution, while the mesh itself is refined or coarsened to better capture solution features. This dual strategy allows for a more targeted and effective use of computational resources, improving the representation of complex phenomena without unnecessary refinement in regions of lesser importance. Ong, Russell, and Ruuth have studied the r-h moving mesh method for one-dimensional time-dependent partial differential equations [34]. In a different area, Piggott has worked on applying h-r adaptivity within numerical ocean modeling [36]. Similarly, the h-p method combines h-refinement and p-refinement, applying both mesh refinement and increased polynomial order to adaptively improve solution accuracy. This approach is particularly effective in addressing regions with varying solution behavior, balancing computational effort with accuracy. Babushka has explored the integration of h-p refinement techniques within the FEM for various partial differential equations [2]. This work demonstrates how h-p methods can enhance solution accuracy and computational efficiency. In a related area, Paul Houston and Endre Süli have applied h-p adaptivity in Discontinuous Galerkin FEM for first-order hyperbolic problems [20].

In this thesis, we use CLAWPACK software to solve 1D scalar conservation laws numerically. We take three example of test equations. Test equation 1 is linear advection equation (2.6), test equation 2 is inviscid Burgers equation (2.7), and lastly test equation3 is Buckley-Leverett equation (2.8).

The details of the algorithm following by [6] is given in the next section.

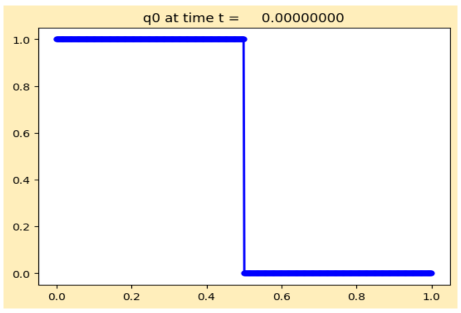
3.2 Description of the Algorithm of AMR

In numerical simulations solving PDEs with sharp transitions or significant changes like discontinuities or oscillations presents a considerable challenge. Classical grid methods often struggle when dealing with solutions that have single jump discontinuities or rapid variations, affecting both accuracy and computational efficiency. In every computational task, there is a goal to design a grid that minimizes discretization errors while enhancing the efficiency of the solution approach. However, in 1D simulations, even the finest grids often fall short in providing adequate resolution for certain features of the solution. This places considerable pressure on computing resources, both in terms of memory and processing time. To overcome this issue, it is crucial to implement a method that intelligently allocates grid points to regions where they are most required.

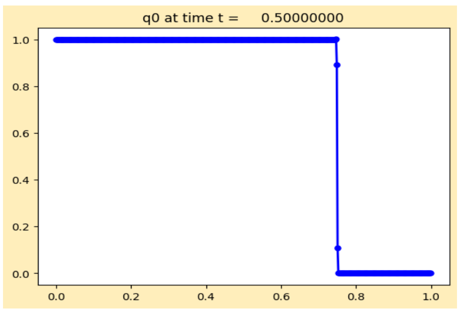
In some problems, such as the step function example for the test equation 2 (2.7), the initial data is considered with the given initial condition,

$$u(x,0) = \begin{cases} 1 & \text{if } x < 0.5, \\ 0 & \text{if } x > 0.5 \end{cases}$$

the solution consists of a single jump, and the location of the discontinuity is known in advance. The discontinuity propagates to the right along the line where $x = 0.5$.



(a) Shockwave when $t = 0$



(b) Shockwave when $t = 0.5$

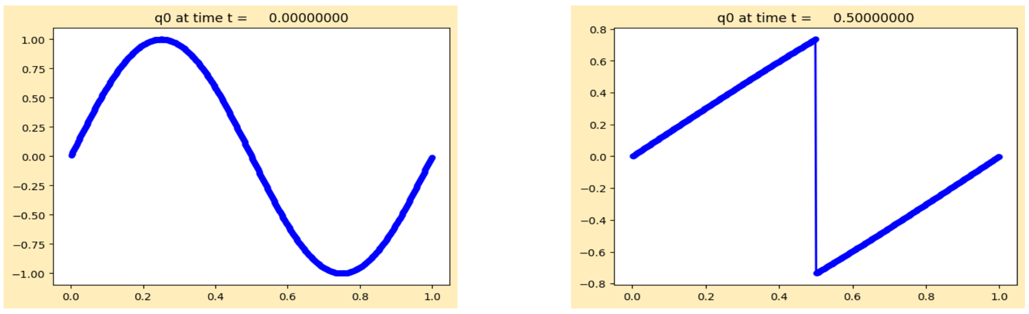
Figure 3.1: Evolution of the step function over time

Apart from the discontinuous point, the sides of the step function remain constant. This predictable behavior makes it relatively easy to handle with uniform grid methods, as can be observed in Figure 3.1.

However, this is not always the case. For example, by considering the same test equation (2.7) with the following initial condition with the initial data and the boundary conditions

$$u(x,0) = \sin(2\pi x), \quad u\left(\frac{1}{4},0\right) = 1 \quad u\left(\frac{3}{4},0\right) = -1.$$

In this case, despite having smooth initial conditions, the solution to the PDE develops into an N-wave structure and eventually reaches a steady state characterized by a single jump discontinuity as depicted in Figure 3.2 below.



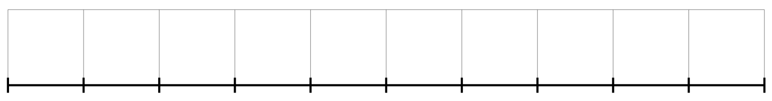
(a) $\sin(2\pi x)$ when $t = 0$ (b) $\sin(2\pi x)$ when $t = 0.5$

Figure 3.2: Evolution of the solution from a smooth initial condition to an N-wave structure

This illustrates how uniform grids may struggle with problems where the discontinuities and critical features are not known in advance, highlighting the need for adaptive techniques like AMR to effectively manage and resolve such dynamic features. We are dealing with 1D hyperbolic PDEs which are characterized by waves that propagate at finite speeds. This allows for frequent adjustments to the grid, ensuring that regions requiring refinement are properly covered. This approach is effective even in steady-state solutions.

In problems where it's predetermined that grid refinement is needed, it's more efficient and cost friendly to incorporate it directly into the uniformly mesh rather than using adaptive techniques. As can be thought guessing the locations of grid refinement is not always possible and challenging, adaptive mesh techniques are preferred at that point. The method is called AMR. In the most general way, algorithm can be defined as follows:

In the 1D AMR algorithm, the computational domain begins with a coarse grid covering the interval $[a, b]$. To identify which intervals require increased resolution, refinement criteria based on error estimation techniques such as 'flag2refine' and Richardson extrapolation are used. These intervals are subdivided into smaller intervals recursively until a desired level of accuracy is achieved. Time steps are adjusted proportionally to grid refinement levels to maintain stability with explicit difference schemes. Boundary conditions are enforced using ghost cells, extending the domain with additional cells beyond $[a, b]$ to handle inflow, outflow, or reflective conditions as needed. Cell averages are updated using flux-differencing algorithms, integrating solutions sequentially by grid level to ensure consistency across refined and coarse grid interfaces. This approach optimizes computational efficiency by focusing grid updates on refined regions while maintaining continuity and accuracy throughout the simulation [4].


(a) Uniform Mesh

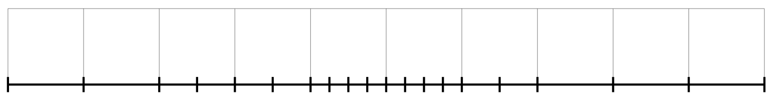

(b) Adaptive Mesh

Figure 3.3: Comparison between uniform mesh and adaptive mesh

3.3 Description of the Grid

In 1D simulations, component grids are an important technique used to handle complex geometries and different physical components. This approach involves dividing the computational domain into smaller, independently defined grids ,subgrids or subintervals, optimizing each grid according to specific characteristics. For example, in order to more accurately model sudden changes in the properties of a medium, such as shock waves, higher resolution and more refined calculations can be performed in these regions [6]. Component grids can also be thought of as subintervals, with each subinterval adjusted to better capture the local behavior of the solution, thus increasing overall accuracy and efficiency. Using separate grids for each component simplifies the grid generation process, but managing the interfaces of these grids presents an additional challenge; smooth transitions and accurate communication must be ensured. Marsha Berger's work provides a framework to understand and manage the computational effort required for using component grids and introduces a specific data structure to effectively handle and manage different components and their interfaces. With this method, 1-dimensional computational simulations can achieve greater accuracy and efficiency, especially in regions with complex geometries or varying physical properties .

In this context, the term *grid* refers to the interval that encompasses the entire set of points defining the grid, rather than the individual points themselves. Specifically, a grid is identified as the smallest interval that includes all the defining points. Overlapping grids imply that there is a non-empty intersection between the intervals of the grids. The computation process starts with an initial base grid, denoted as G_0, which is set up by the user before starting the analysis. This base grid is composed of several component grids, labeled as $G_{0,j}$ where j is a natural number. Each component grid is required to maintain local uniformity, ensuring consistent spacing and organization within its segment of the domain. This consistent arrangement within each component grid is crucial for achieving precise results and effectively managing the analysis. The grids can vary in mesh width, allowing for finer discretization in areas of interest, such as boundary layers, compared to coarser grids in other parts of the domain. Within each grid, the mesh spacing along coordinate x direction does not have to be uniform. However, we assume that G_0 maintains a uniform mesh spacing $h_x = h_0$ across all its component grids $G_{0,j}$. During the computation, adaptive refinement occurs, generating additional subgrids in response to specific features in the transient solution, such as estimated solution errors or the presence of shock fronts.

The Figure 3.4 displays a grid structure comprising two component grids at the coarsest level. Within one of these component grids, there exists a refined subgrid. The shaded area within the subgrid indicates the segment of the coarse grid that requires refinement, as described in [5].

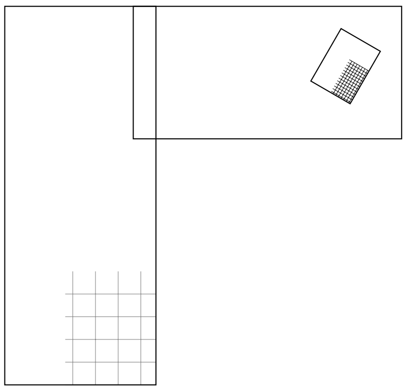


Figure 3.4: Coarse grid with a refined subgrid

In the following example, it illustrates the use of adaptive subintervals in a 1D.

The highlighted region in gray within the coarse grid represents an area requiring more detail around a specific feature in the solution. Around features like a shock front, for instance, a refined subgrid is drawn with higher resolution within the coarse grid, ensuring a more precise representation of that region in the solution.

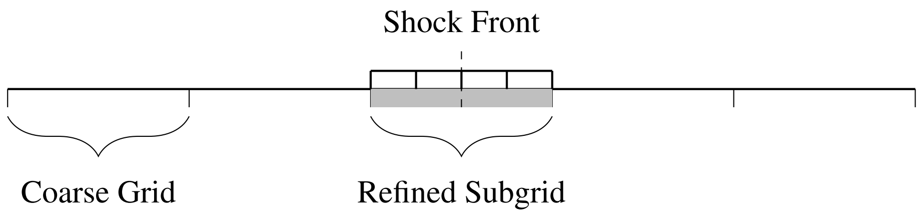


Figure 3.5: Illusturation of 1D coarse and refined grids

It's important to understand that smaller grids, also known as subgrids, are not integrated into the coarse grid. Instead, each grid, whether it is coarse or refined, is defined independently with its own solution vector and storage. This allows each subgrid to be processed nearly independently from the others, which simplifies the use of moving subgrids even if the primary grid remains stationary.

Each grid in the AMR system operates independently, allowing effective domain partitioning that supports multi-processor systems. Subgrids can be nested within larger grids, and points are categorized based on their position relative to these grids either within, on the boundary, or contained by other grids.

The level of a grid is defined by its nesting depth, which indicates the number of coarser grids it resides within. The base grid G_0 represents level 0 in this hierarchy. Subgrids of G_0 are referred to as level 1 refinement, denoted G_1. Further refinement within G_1 leads to level 2 grids, or G_2, and this process continues for additional levels. This hierarchical nesting establishes a sequence of grids with increasingly finer discretizations across both spatial and temporal domains. For a visual example of these refined grids, see Figure 3.9.

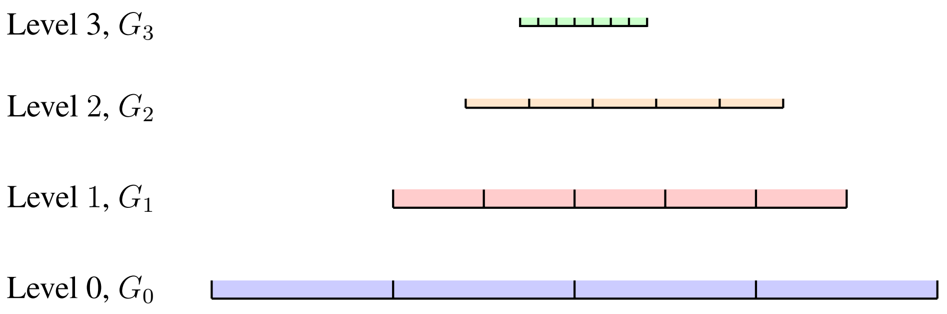


Figure 3.6: Levels of refinement in a 1D grid

In this manner, a hierarchical sequence of grids can be established to progressively refine the discretization within a specific segment of the spatial domain. Each grid, denoted as G_l, represents a level l in this hierarchy and is characterized by a mesh width h_l. Thus, a point within the problem domain may be included in multiple grids. The approximate solution at such a point is determined by interpolating from the finest grid that contains the point.

The set of mesh discretizations $\{h_0, h_1, h_2, \ldots, h_{\max}\}$ is defined in advance, where each mesh width h_l is an integral multiple of h_{l+1}, often using a factor of four. For different regions within the domain, the choice of the optimal refinement ratio depends on specific needs. For instance, if a finer grid is required in a certain area, where $h_{finer} = \frac{h_0}{r}$ and h_0 is the initial mesh width, applying a single refinement level with $h_1 = \frac{h_0}{r}$ is generally more effective than using two levels with a ratio of $\sqrt{r}$. Smaller values of the refinement ratio r are often preferred since not all regions need the same degree of refinement. Specifically, suggested a preference for a refinement ratio of 4 over the more common ratio of 2 found in multi-grid methods, especially for 1D grids [5]

In the context of 1D grids, preference for a refinement ratio of 4 rather than the usual ratio of 2 found in multi-grid methods means that each refined grid segment is divided into four smaller segments rather than just two.

Refinement Ratio of 2: Each interval on the base grid G_0 is subdivided into two smaller intervals. An example can be given as, we start with an interval [a, b], it is divided into

$$\left[a, \frac{a+b}{2}\right], \left[\frac{a+b}{2}, b\right]$$

Refinement Ratio of 4: Each interval on the base grid G_0 is subdivided into four smaller intervals. An example for this can be given as if we start with an interval $[a, b]$, it would be divided into four equal segments:

$$\left[a, a+\tfrac{b-a}{4}\right], \left[a+\tfrac{b-a}{4}, a+\tfrac{b-a}{2}\right], \left[a+\tfrac{b-a}{2}, a+\tfrac{3(b-a)}{4}\right], \left[a+\tfrac{3(b-a)}{4}, b\right]$$

By preferring a refinement ratio of 4, it is aimed to achieve a higher resolution in the refined grids, which helps to capture finer details of the solution. This approach, however, increases the computational workload, as there are more grid points to handle and more computations to perform at each refinement level. In particular scenarios where all regions requiring refinement are expected to need a significant amount of it, higher values of *r* can be implemented efficiently in need. In this thesis with the CLAWPACK environment in some examples we have chosen refinement ratios of 2

and 4 in various examples for computational efficiency and cost considerations based on the specific problem requirements.

In the 1D case addressed in this thesis, each grid represents a simple interval, eliminating the complexities of overlapping and nesting found in higher dimensions. Level nesting in one dimension is identical to straightforward grid nesting, and overlapping of fine grids does not occur. Each finer grid is entirely contained within a coarser grid at the next level of refinement.

Each level of refinement in a 1D grid system divides a line segment, such as from a to b where $a, b \in \mathbb{R}$, into smaller intervals. Each segment represents a different level of detail or refinement. Unlike higher dimensions where grid overlaps can occur, such situations are avoided in one dimension.

If two grids with the same mesh width are at the same refinement level, they either do not intersect at all or merge into a single, larger grid spanning the combined interval.

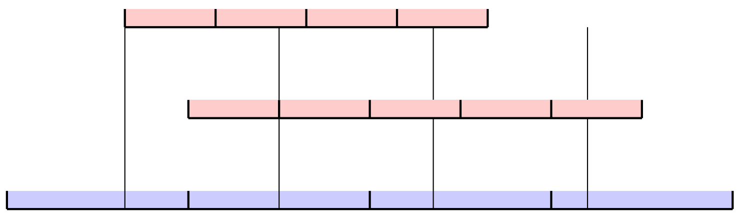


Figure 3.7: Separate identical level refinements

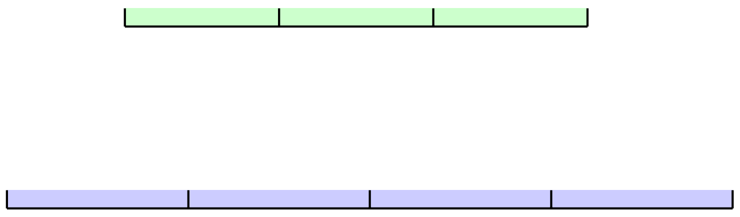


Figure 3.8: Merged Identical Level Refinement

In the first Figure 3.7, two grids of the same width are shown at different refinement levels. The second Figure 3.8 illustrates how these two grids combine to form a coarser grid, maintaining the overall width while integrating the finer details from the individual grids. This process demonstrates the hierarchical nature of grid refinement and coarsening in AMR, where finer grids can be merged into coarser representations to optimize computational efficiency.

Using the concepts of independent intervals and recursive refinement, we create a hierarchy of grids. The entire grid structure is denoted by

$$G = \bigcup_{l} G_l$$

where each G_l represents an interval at refinement level l.

3.4 Algorithm of Integration

In this section, we describe the integration algorithm used to solve a hyperbolic partial differential equation using mesh refinement in one dimension. The algorithm comprises three main components: (i) time integration using finite differences on each grid, (ii) error estimation and subsequent grid generation, and (iii) special grid-to-grid operations necessitated by mesh refinement. Each grid is treated as an independent computational entity with its own solution vector, allowing for independent time integration except for determining boundary values. The integration order is determined by ensuring that mesh widths h_l at level l are related by a refinement factor r relative to h_{l-1}, with time steps $k_l = k_{l-1}/r$ set accordingly to maintain a constant CFL condition $\lambda = c\Delta t/\Delta x$ across all grids that can also be decided by the user in the setrun.py file as desired CFL in CLAWPACK which has a maximum limit of 1. This approach ensures efficient integration across different scales without requiring globally restrictive time steps. The algorithm advances subgrids to the same time level before progressing the parent grid, forming a coarse grid cycle as the primary unit of computation in one dimension. By advancing subgrids to the same time level before their parent grid, the algorithm synchronizes computations on finer grids with those on coarser grids. This method maintains consistency and accuracy in the numerical solution by coordinating the advancement of various grid levels within each computational cycle.

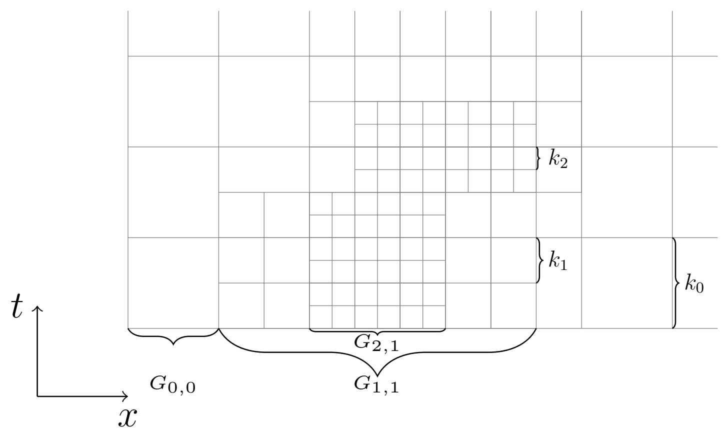


Figure 3.9: Nested grids example with different levels of refinement

In the figure above which is from Berger's thesis [5], The illustration highlights 1D space and time, showcasing a coarse grid labeled $G_{0,1}$, a fine grid at level 1 $G_{1,1}$ compared to $G_{0,1}$, and an even finer grid $G_{2,1}$ at level 2, with a refinement ratio of $r = 2$ at level 1 and $r^2 = 4$ at level 2. This hierarchical refinement structure allows for adaptive resolution across the domain, focusing computational resources where finer detail is needed, such as around features like shocks in the solution. The AMR approach ensures that computational effort is optimized, providing accurate solutions while minimizing computational costs where detailed resolution is less critical.

3.5 Estimation of Error

The CLAWPACK algorithm refines grids by estimating local truncation errors [5]. This involves assessing errors at all grid points to create new finer grids or remove unnecessary ones, optimizing computational resources. For hyperbolic problems, regridding frequency is crucial, often every $3-4$ steps, to account for predictable propagation speeds.

An alternative approach in CLAWPACK is the 'flag2refine' method. This flags cells for refinement based on the maximum undivided difference in the solution with a decided buffer zone of $3-4$ neighboring cells , compared to a specified error tolerance ϵ. This gradient-based approach detects solution discontinuities, differing from truncation error methods that rely on convergence results for hyperbolic systems. Both methods aim to balance computational efficiency and accuracy by refining regions with high error, but 'flag2refine' uses simpler, more direct criteria. We prefer to use this criteria in our numerical results.

3.6 Conditions of Interface

An interface condition in numerical methods and simulations refers to the rules or equations that ensure continuity and consistency across discretized interfaces within a computational domain [27]. In 1D simulations , the Coarse Mesh Approximation Method (CMAM) [11] addresses the interface between a coarse grid and a finer grid. Here, the finer grid, refined by a factor n, features smaller spatial (Δx_{finer}) and temporal (Δt_{finer}) intervals compared to the coarse grid ($\Delta x_{coarser}$ and $\Delta t_{coarser}$). To determine values at the boundary of the finer grid ($u_{0,i}$ for $0 < i \leq n$), the CMAM utilizes values from the coarse grid while applying the same numerical method, but with a reduced time step on the finer grid. This approach ensures stability, particularly when the mesh ratio

$$\lambda = \frac{\Delta t_{coarser}}{\Delta x_{coarser}}$$

of the coarse grid meets specified criteria [5]. Subsequent steps in the mesh refinement algorithm involve updating procedures between coarse and finer grids. In 1D AMR, updating is vital for preserving the accuracy and consistency of solutions across different grid levels. This involves adjusting values on coarser grids by incorporating information from finer grids, either by interpolation or direct injection. Such updating ensures that finer details captured on the finer grids are accurately reflected in the coarser grid solution, preventing accuracy loss. Additionally, this updating process mitigates error propagation: by correcting values on the coarse grid, it prevents inaccuracies from spreading to adjacent regions, including finer grids, thus maintaining the overall integrity and reliability of the solution throughout the simulation.

In CLAWPACK implementation, the mesh refinement algorithm operates recursively to enhance the accuracy of numerical simulations. Initially, the solution is integrated on the coarsest grid level. This process is repeated iteratively, with the number of iterations determined by the ratio of finer to coarser grid spacing. CLAWPACK employs

Richardson extrapolation to estimate errors and compare them against a specified tolerance ϵ. If the estimated errors exceed ϵ, indicating the need for further refinement, new grids with finer mesh widths are created to capture more detailed features of the solution.

Although the grid generation process is efficient, it incurs higher computational costs when integrating finer grids due to their increased area. The goal is to minimize the total area occupied by the refined grids while ensuring that the grid coordinates are closely aligned with the solution, thereby achieving optimal simulation accuracy.

During the regridding process, the algorithm may introduce new grid levels, modify existing ones, or eliminate unnecessary levels. When adjusting a fine grid in a 1D simulation, the process generally involves creating a new grid instance. This new grid is initialized with solution values obtained from the previous refinement before the old grid is removed. This systematic approach ensures that continuity and accuracy are preserved throughout the refinement process, allowing the refined grids to accurately reflect the evolving dynamics of the solution.

The algorithm identifies grid points at level l that need to be refined at a finer level $l + 1$. It then clusters these flagged points and generates new grids based on these clusters. This procedure is essential for adapting the grid structure to capture and refine regions where the solution dynamics require increased resolution.

The algorithm begins by identifying grid points at level l that need refinement into finer grids at level $l + 1$. Subsequently, it clusters these flagged points and generates new grids based on these clusters. In 1D grids, this clustering process is straightforward, as each grid corresponds to an interval. The boundaries of new subgrids are defined by the leftmost and rightmost flagged points on the coarse grid, encompassing all flagged points between them. If there are gaps of unflagged points of sufficient size, multiple separate subgrids may be created instead. The buffer zone around flagged points determines how frequently grids are evaluated and the spacing between them, ensuring efficient refinement. Points closer than twice the buffer zone size are grouped together within the same grid refinement to maintain effective resolution.

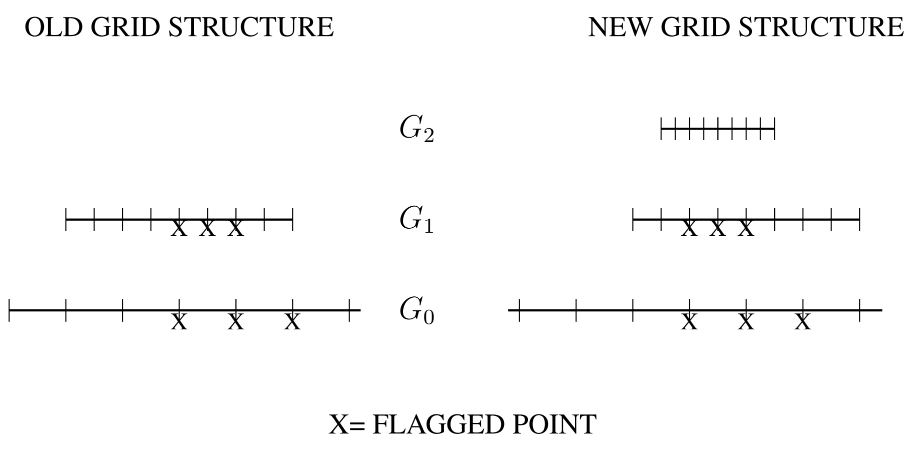


Figure 3.10: One dimensional regridding procedure

The Figure 3.10 [6] illustrates the grid configuration before and after regridding. It emphasizes specific grid points marked with 'X', which represent areas with significant error estimates. Instead of overlaying the grids, each grid is shown separately to provide a clear view of the refined grid structure.

3.7 Structures of Data

The AMR strategy that CLAWPACK uses, applies remarkably direct data structures that are crucial for its feasibility. Especially for 1D problems CLAWPACK utilizes a tree data structure based on [6] where each node represents a grid. The nesting requirement for 1D mesh refinement dictates that each fine grid must be entirely contained within a coarser grid at the next level. This relationship defines the tree structure: a parent node corresponds to a coarser grid, while its descendants represent subgrids. Siblings are subgrids within the same parent grid, and neighbors are fine grids at the same refinement level but with different parents.

An ordered tree data structure is used where each node can have multiple descendants, reflecting that a coarse grid can contain several fine grids. Nodes are ordered using the coordinate value of the left-most grid point in the associated grid. Grid-to-grid operations, such as updating coarse grids and setting internal boundary values for fine grids, follow the tree's path links. An additional non-standard link, indicated by a dashed line, represents the neighbor pointer, facilitating operations across grids at the same refinement level.

Dynamic storage allocation is essential as the tree structure grows or shrinks dynamically. Regridding operations often involve moving only the finest level grids, leaving the coarser level grids fixed and connecting a new bottom half to the existing top half of the tree. Each node in the tree contains a fixed amount of information: grid location, number of grid points, level in the tree, offspring pointer, sibling pointer, parent pointer, a pointer to the next grid at the same level, the time integrated to, and an index into the main storage array for approximate solution values. This structured approach ensures efficient and organized management of grid information and solution storage, supporting the dynamic and hierarchical nature of AMR in 1D contexts. Figure 3.11 illustrates the data structure for 1D for Berger-Oliger AMR.

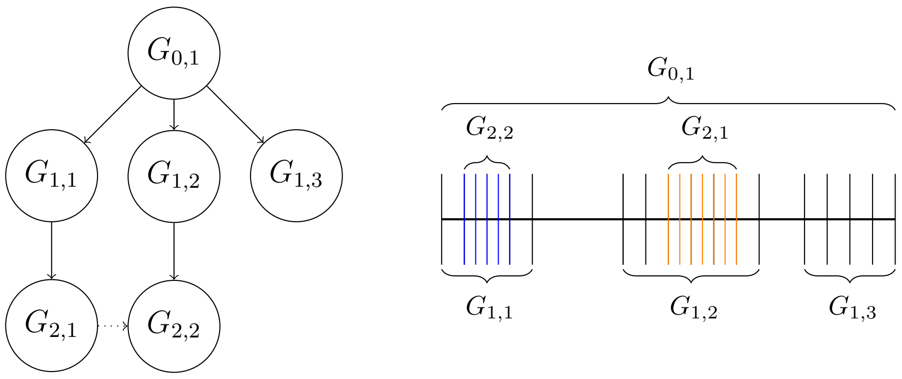


Figure 3.11: Structure of data for 1D

CHAPTER 4

NUMERICAL IMPLEMENTATION OF SCALAR CONSERVATION LAWS BY AMR

In this chapter we focus on approximating the solutions to several 1D scalar conservation laws. Our test equations are linear advection (2.6) , inviscid Burgers (2.7), and Buckley-Leverett (2.8) equations. We apply FVM approximations to these equations and obtain numerical solutions using both uniform and adaptive refined meshes with a predetermined buffer zone of 3 neighbouring cells. Our numerical simulations utilize Godunov type numerical flux which is known for its accuracy in handling shock waves. We pay particular attention to the initial data, as it plays a crucial role in determining how these waves evolve and behave in time.

4.1 Numerical Results of Scalar Conservation Laws

This section presents the numerical investigation of three test equations: Test Equation 1: linear advection equation (2.6), Test Equation 2: inviscid Burgers equation (2.7) and Test Equation 3: Buckley-Leverett equation (2.8). Each equation is analyzed with different initial and boundary conditions. Our investigation includes two groups of figures for each equation. First, the numerical solutions are compared using $N = 20$ and $N = 60$ uniform grids with a valid CFL=0.9. Then, the solution achieved with $N = 60$ uniform grids is compared to the results obtained using AMR for each equation and each initial condition. Firstly we begin with test equation 1 (2.6) with the initial condition (4.1). Then we move on investigating test equation 2 (2.7) with the initial condition 1 (4.2). Following to this we investigate the initial conditions 2 (4.3) and 3 (4.4) for the same test equation. Lastly we investigate 3rd test equation

(2.8) with the initial condition (4.5). The details of the initial conditions are stated in the related subsections for each equation.

4.1.1 Test Equation 1: Linear advection equation

In this part we present the Test Equation 1 (2.6) with initial condition

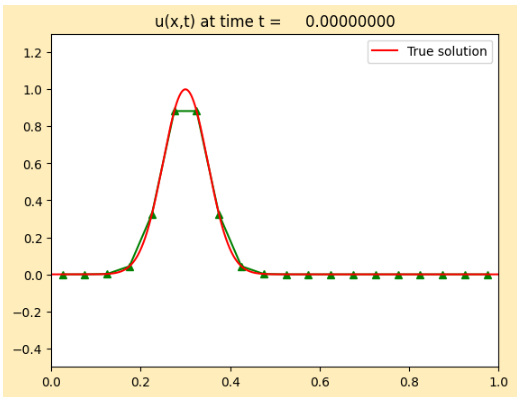
$$u(x,0) = e^{-\beta(x-x_0)^2} = e^{-200(x-0.3)^2} \tag{4.1}$$

where the Gaussian pressure pulse is centered at $x = x_0 = 0.3$ with width determined by $\beta = 200$ and advection velocity $c = 1$.

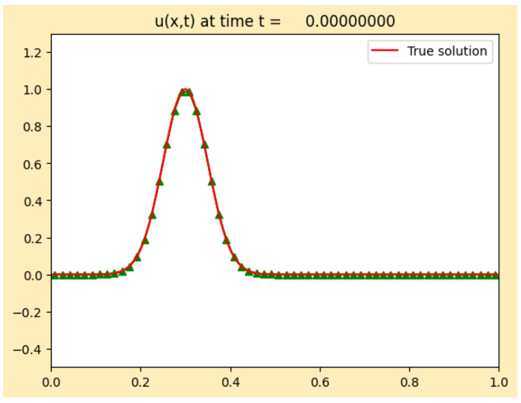
Figure 4.1 shows the solution of (2.6) under periodic boundary conditions. The true solution is represented by the red line, while the numerical solutions are computed using two different grid resolutions. The results are displayed at $t = 0$, $t = 0.5$, and $t = 1.0$ with a fixed $CFL = 0.9$.

For $N = 20$ uniform grids, the grid spacing is $\Delta x \approx 0.0526$ and the time step is $\Delta t \approx 0.0474$. In contrast, with $N = 60$ uniform grids, the grid spacing is reduced to $\Delta x \approx 0.0169$ and the time step to $\Delta t \approx 0.0152$. The results underscore the effect of grid resolution on the accuracy of the numerical solution.

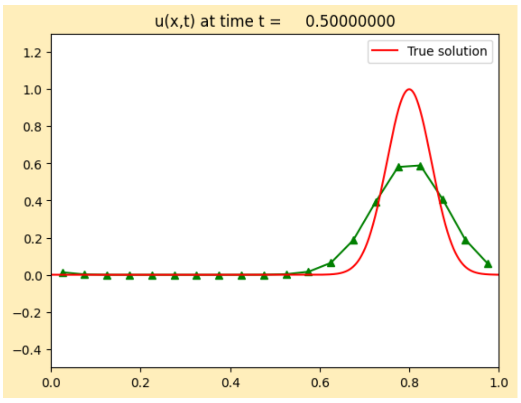
The comparison reveals that with fewer grid points with $N = 20$, the accuracy of the numerical solution relative to the true solution decreases. This is because the lower resolution grid struggles to capture finer details and evolving features of the solution as the simulation progresses. Conversely, as the number of uniform grid points increases when $N = 60$ in the spatial domain $x = [0, 1]$, the resolution improves and the accuracy of the solution becomes significantly better. This demonstrates the critical role of grid resolution in numerical simulations. Higher resolution grids provide a more precise representation of the solution, making them essential for achieving accurate results. The results highlight how increasing the number of grid points can effectively enhance the accuracy of the numerical solution, underscoring the importance of selecting an appropriate grid resolution to accurately model and solve equations like (2.6).



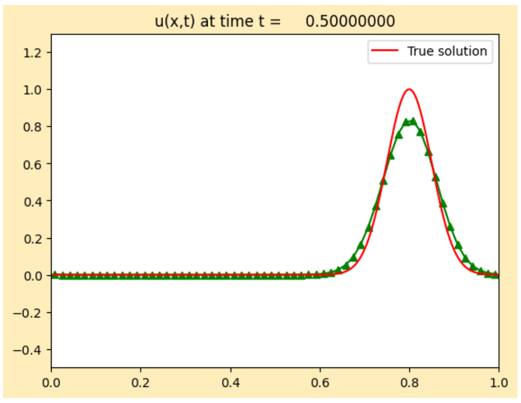
(a) Uniform $N = 20$, $t = 0.0$



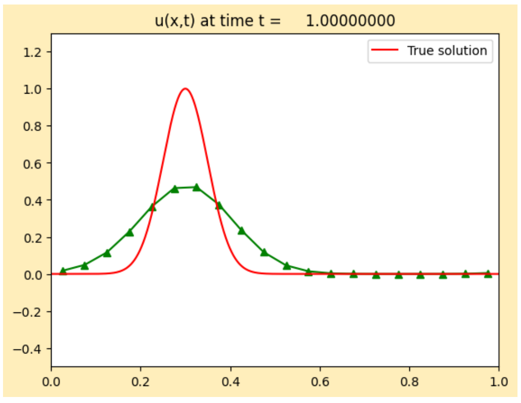
(b) Uniform $N = 60$, $t = 0.0$



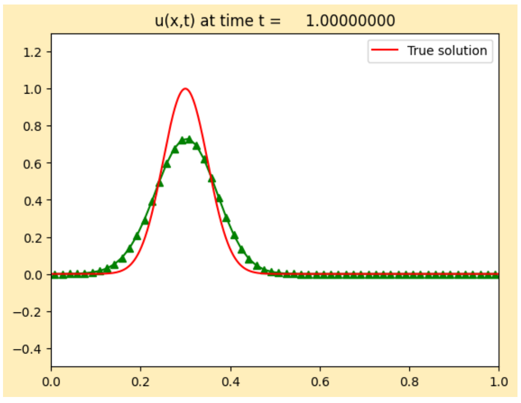
(c) Uniform $N = 20$, $t = 0.5$



(d) Uniform $N = 60$, $t = 0.5$



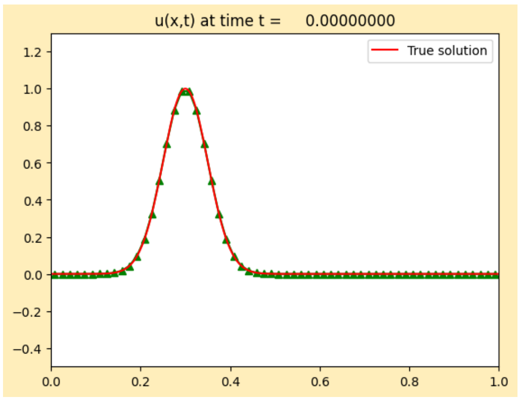
(e) Uniform $N = 20$, $t = 1.0$



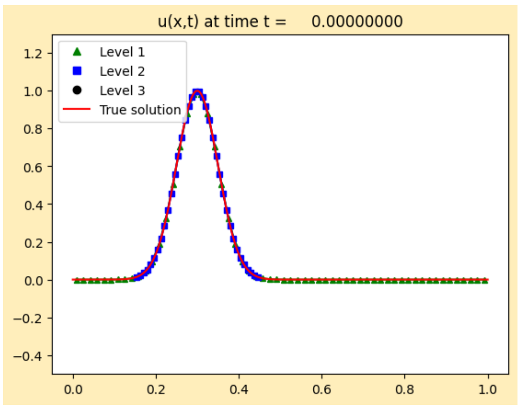
(f) Uniform $N = 60$, $t = 1.0$

Figure 4.1: Numerical solution of linear advection with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.1) and $CFL = 0.9$.

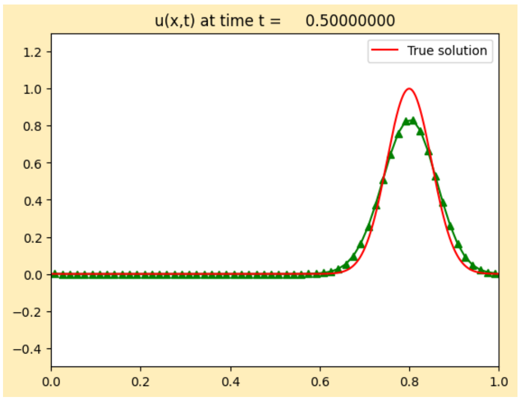
Figure 4.2 offers a comparison of the accuracy between $N = 60$ uniform grids and AMR results at $t = 0$, $t = 0.5$, and $t = 1.0$ of the equation 2.6.The right side of the figure offers the AMR results where the left side of the figure offers the $N = 60$ uniform grid results. In the AMR solution, the error bound is set to $\epsilon = 0.2$ using the 'flag2refine' criteria with a refinement ratio $r = 2$. In the Figure, level 1 corresponds to the grid resolution of 60 uniform grids, represented by the green triangle. Level 2 highlights the areas around the Gaussian pulse that were refined with a refinement ratio of $r = 2$, indicated by the blue squares. Refinement to level 3 was not necessary. Refining only to level 2 provided an efficient solution compared to uniform solution, making further refinement unnecessary as the solution did not form any additional sharp gradients or shocks that would lead to further refinement. It is observed that the grids around Gaussian pulse is refined rather than other grids to increase the effect of the solution. Although using $N = 60$ uniform grids gives us a more accurate solution compared to $N = 20$ uniform grids, the accuracy tends to lessen over time when compared to the true solution. This happens because the uniform grid's constant resolution struggles to keep up with the changing details of the solution feature as the wave propagates. At $t = 0$, the compared solutions look quite similar, but $t = 1.0$, the differences become more noticeable. AMR solution provides a highly efficient solution even with a refinement ratio of $r = 2$. AMR dynamically adjusts the grid resolution around the Gaussian pulse while avoiding unnecessary refinement in the other parts of spatial domain x. By setting a smaller error bound the solution can achieve even higher accuracy, as it refines the grid more precisely in areas with larger errors. Additionally due to the refinement ratio $r = 2$ it can be observed that the AMR solution achieves an accuracy of what would be obtained by using $N = 120$ grids uniformly spread across the entire domain with a more efficient use of grids. By focusing refinement only around the Gaussian pulse and avoiding unnecessary refinement in other areas, the AMR approach achieves higher accuracy with fewer grids. This approach provides a more accurate solution with less computational effort.



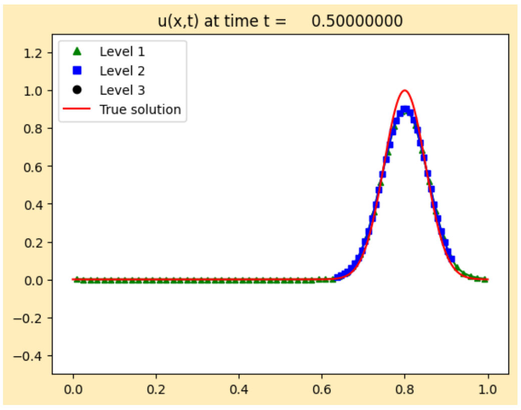
(a) Uniform $N = 60$, $t = 0.0$



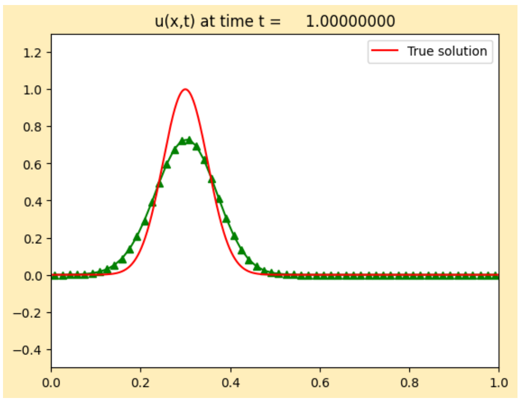
(b) AMR $N = 60$, $t = 0.0$



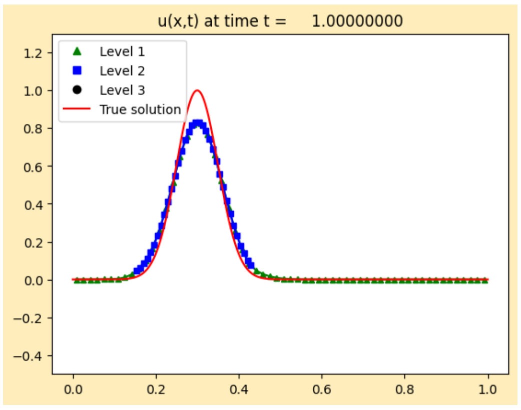
(c) Uniform $N = 60$, $t = 0.0$



(d) AMR $N = 60$, $t = 0.5$



(e) Uniform $N = 60$, $t = 1.0$



(f) AMR $N = 60$, $t = 1.0$

Figure 4.2: Comparison of uniform mesh and AMR for linear advection equation. Initial condition (4.1) with $N = 60$ and CFL$= 0.9$.

4.1.2 Test Equation 2: Inviscid Burgers Equation

In this subsection we present the 2nd test equation (2.7) with three different initial conditions. The first initial condition is

1. The first initial condition is

$$u(x,0) = \begin{cases} 1 & \text{if } x < 0, \\ 0 & \text{if } x > 0. \end{cases} \tag{4.2}$$

   which results shock wave.

2. The second initial condition is

$$u(x,0) = \begin{cases} 0 & \text{if } x < 0, \\ 1 & \text{if } x > 0. \end{cases} \tag{4.3}$$

   results rarefaction wave

3. The third one is a sinusoidal initial condition

$$u(x,0) = \sin(2\pi x) + 0.5 \tag{4.4}$$

gives also shock propagation [23].

We proceed by analyzing the test equation (2.7) with the first initial condition (4.2). In Figure 4.3 reference solution is indicated by the light blue line where the domain is divided by the uniform grids. The figure illustrates the results of the evolution of shock wave at $t = 0,\quad t = 0.5$ and $t = 1.0$ for the number of the uniform grids $N = 20$ and $N = 60$. For both of the uniform solutions $CFL = 0.9$ is fixed and implemented on the solutions.

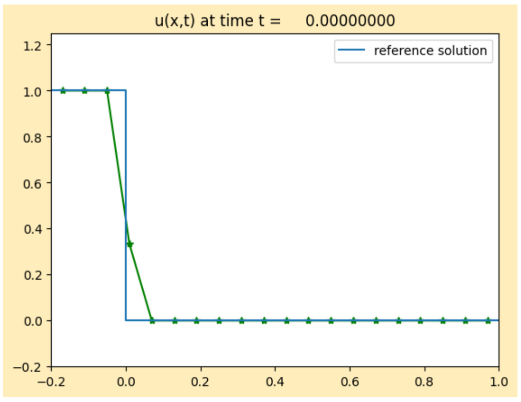
For $N = 20$ grids, the spatial step size is calculated as $\Delta x \approx 0.0632$, and the time step size is $\Delta t \approx 0.1138$, within the spatial domain $x = [-0.2, 1.0]$. In contrast, with $N = 60$ grids, the spatial step size is reduced to $\Delta x \approx 0.0203$, and the time step size becomes $\Delta t \approx 0.0365$. The increased resolution provided by a higher number of grids allows for a more detailed representation of the solution.

Figure 4.3 illustrates the solutions at $t = 0.0$, $t = 0.5$, and $t = 1.0$ for both $N = 20$ and $N = 60$ uniform grids. The differences in the evolution of the solution can be even realized at $t = 0$ unlike with the results observed for the equation (2.6). The initial condition that leads to the formation of a shock wave, highlights the impact of grid resolution on the solution's accuracy.

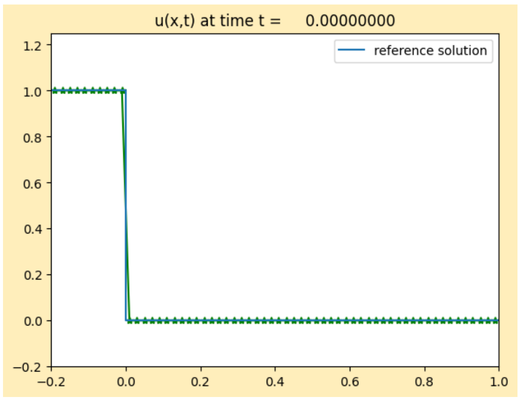
With $N = 60$ grids, the solution captures the shock wave's details and sharp features more effectively compared to $N = 20$. The increased number of uniform grids allow for a more precise representation of the shock wave by closely approaching with the reference solution indicated by the light blue line in the figure.

In contrast, the less uniform grids struggles to accurately capture the evolving shock wave resulting in a significant increase in error compared to the reference solution. The larger spatial and time step sizes lead to a less detailed representation, causing the solution to diverge more from the reference. This differences becomes more feasible over time as the shock wave propagates.

Overall, the comparison shows that fewer grids, with larger step sizes, are less capable of capturing the dynamic features of the shock wave. This results in a larger error between the computed solution and the reference. Increasing the number of grid points provides a finer resolution that more accurately follows the shock wave's evolution and reducing the error and improving the solution's accuracy.

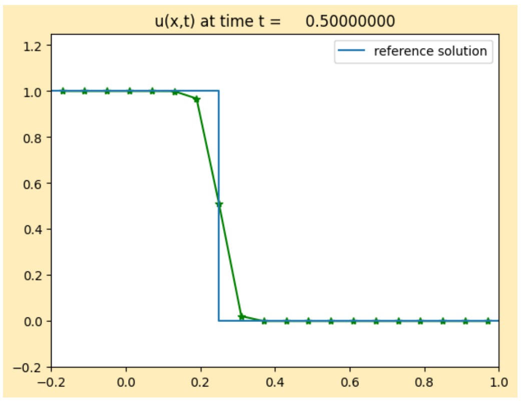


(a) Uniform $N = 20$, $t = 0.0$

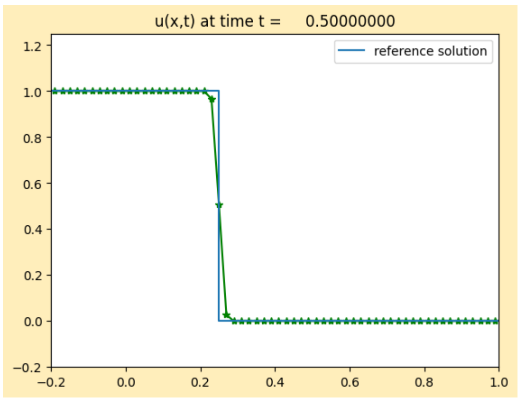


(b) Uniform $N = 60$, $t = 0.0$

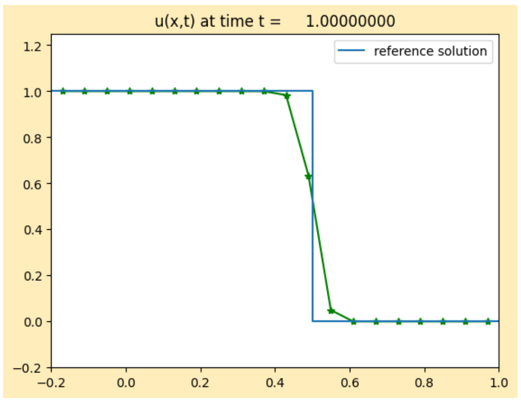
Figure 4.3: Numerical solution of inviscid Burgers equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.2) and CFL= 0.9



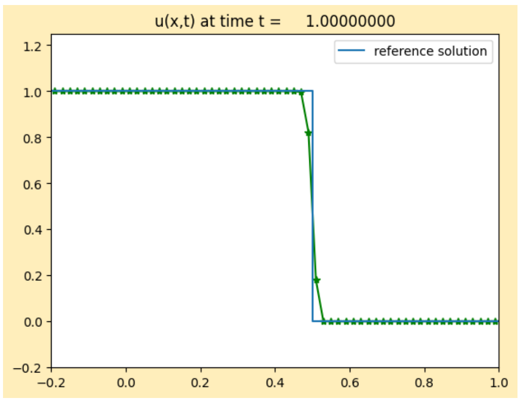
(c) Uniform $N = 20$, $t = 0.5$



(d) Uniform $N = 60$, $t = 0.5$



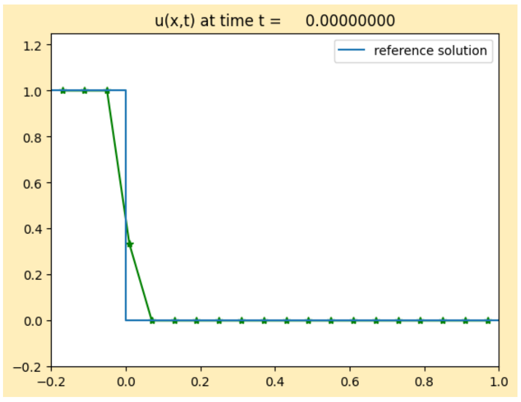
(e) Uniform $N = 20$, $t = 1.0$



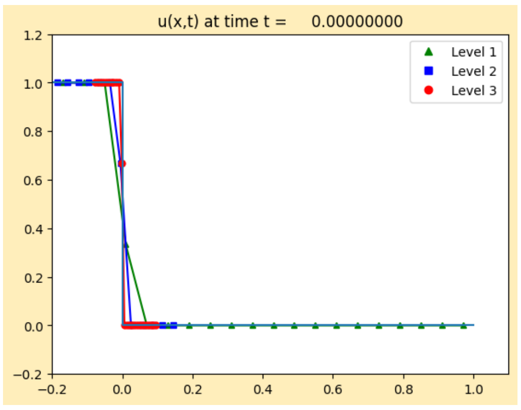
(f) Uniform $N = 60$, $t = 1.0$

Figure 4.3: Continued.

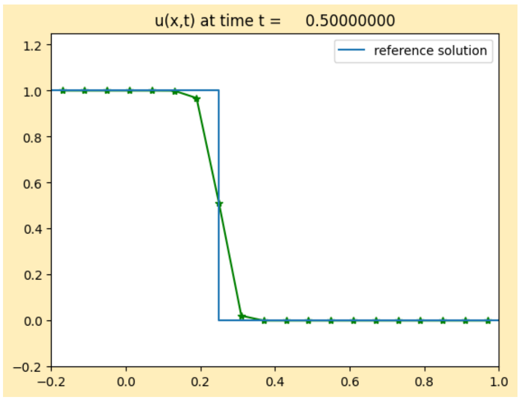
The figures 4.4 and 4.5 illustrate the comparison of $N = 20$ and $N = 60$ uniform grids on the left side of the figure along with the AMR results implemented on $N = 20$ and $N = 60$ results at time $t = 0.0, \quad t = 0.5$ and $t = 1.0$ of the equation (2.7) with shock wave formation. In the combined solution of uniform grids with AMR, the error bound $\epsilon = 0.4$ is set and a buffer zone width is set to be 3 neighbouring cells. Level 1 as illustrated as green triangles represents the $N = 20$ and $N = 60$ uniform grids in the spatial domain $x = [-0.2, 1.0]$ where level 2 is marked as blue squares with level 3 stated with red circles. A refinement ratio of $r = 2$ and $r = 4$ is decided for level 2 and level 3 refinements respectively. Unlike the scenario for the equation (2.6), where higher levels of refinement were not necessary, level 3 refinement is crucial for accurately capturing the shock wave in the (2.7) Equation. This higher level of refinement addresses the sharp discontinuities and evolving features of the shock wave, which are not as effectively captured by lower levels of refinement. As observed in the comparison between $N = 20$ and $N = 60$ uniform grids, increasing the number of grids improves the accuracy of the solution. Similarly with the AMR approach by dynamically adjusting the grid resolution achieves higher accuracy by focusing refinement where it is most needed compared to $N = 20$ and $N = 60$ uniform grids. Even though $N = 60$ uniform grids captures the solution way much better than $N = 20$ uniform grids, by the time steps Figure 4.5 indicates near the shock formation, the solution with $N = 60$ uniform grids struggles to capture the reference solution at $t = 0.5$ and $t = 1.0$. The AMR with level 3 is focused around the sharp discontinuity while level 2 refinement captures broader neighboring grids. As the refinement ratios are set to be 2 and 4, the solution with AMR offers an accuracy of a solution that could be achieved with $N = 480$ uniform grids all together which means instead of increasing the number of the uniform grids all over the spatial domain $x = [-0.2, 1.0]$, the AMR solution refined and achieved the same resolution by refining the grids around the discontinuity in Figure 4.5. With $N = 20$ grids the level of resolution in Figure 4.4 is significantly lower compared to that in Figure 4.5 where a higher number of grids provides a more detailed capture of the shock structure. In Figure 4.4 the effects of AMR refinement are clearly visible as the adaptive approach selectively increases the grid density in regions near the shock enhancing accuracy without unnecessary refinement in smoother areas. The buffer zone and error bound ϵ are the same for both Figures 4.4 and 4.5. Readers can observe the differences between the uniform and AMR solution more clearly in Figure 4.4 because the lower grid resolution makes the adaptive refinement more apparent, highlighting how AMR effectively concentrates computational effort in regions where it's most needed.



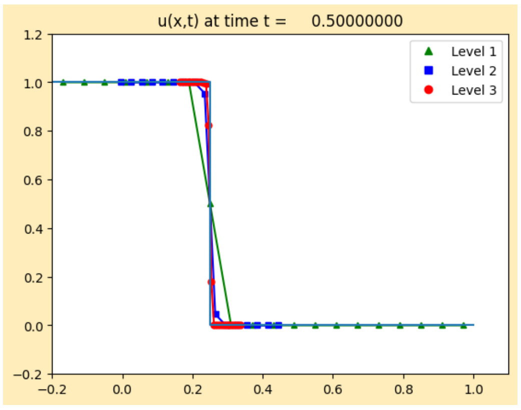
(a) Uniform $N = 20$, $t = 0.0$



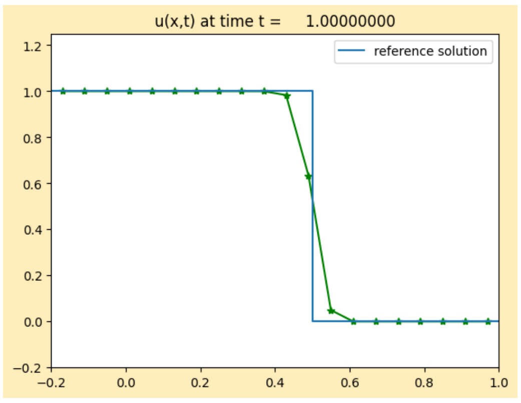
(b) AMR $N = 20$, $t = 0.0$



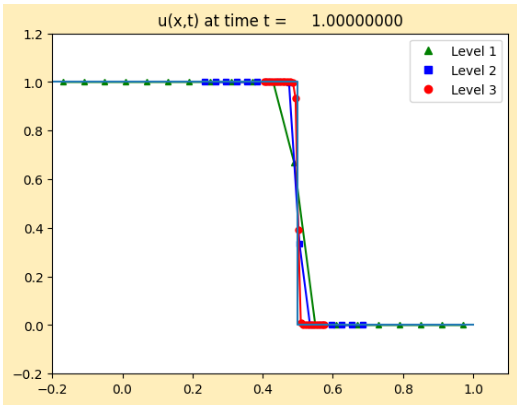
(c) Uniform $N = 20$, $t = 0.5$



(d) AMR $N = 20$, $t = 0.5$

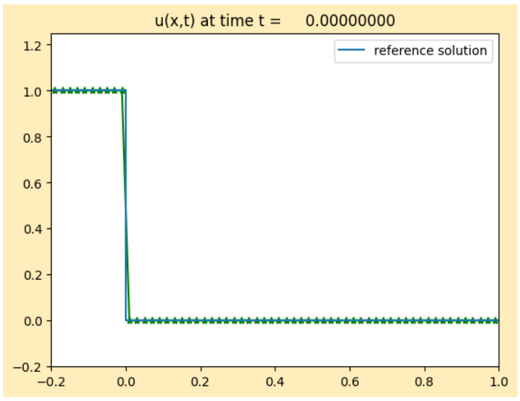


(e) Uniform $N = 20$, $t = 1.0$

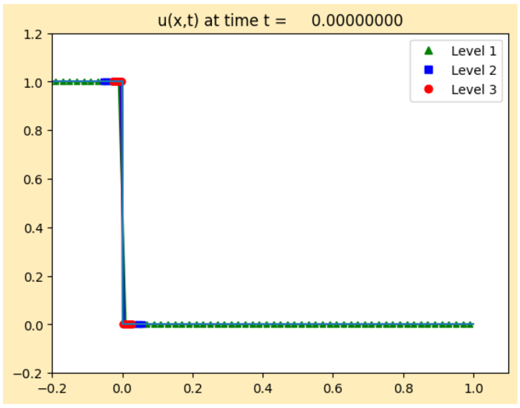


(f) AMR $N = 20$, $t = 1.0$

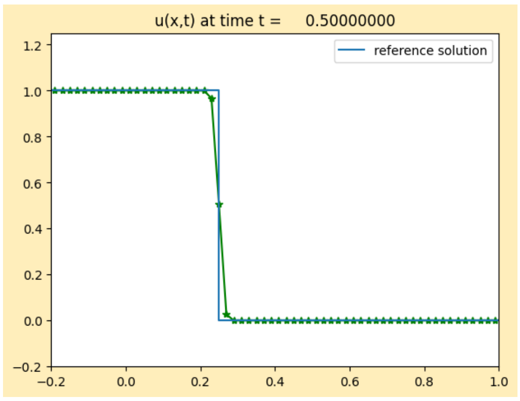
Figure 4.4: Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.2) with $N = 20$ and CFL$= 0.9$.



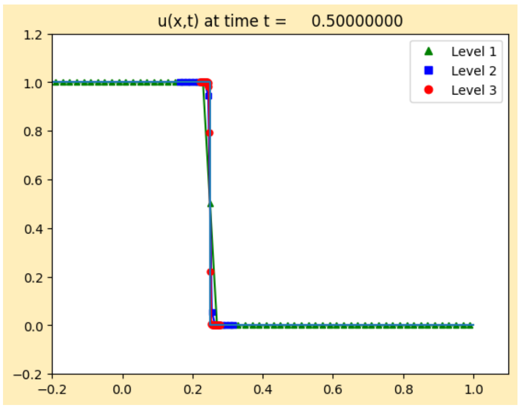
(a) Uniform $N = 60$, $t = 0.0$



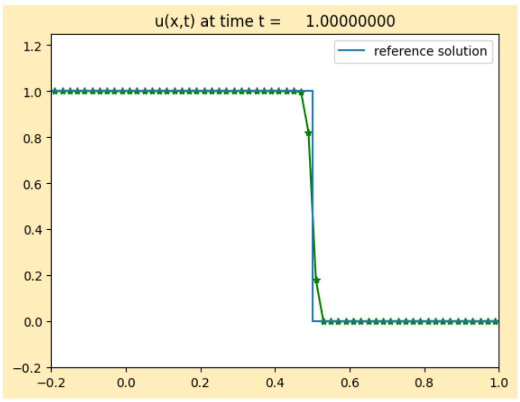
(b) AMR $N = 60$, $t = 0.0$



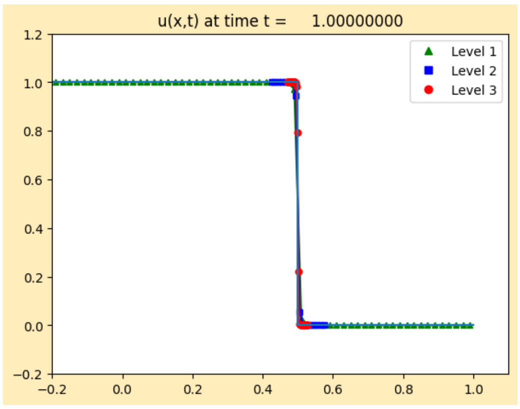
(c) Uniform $N = 60$, $t = 0.5$



(d) AMR $N = 60$, $t = 0.5$



(e) Uniform $N = 60$, $t = 1.0$



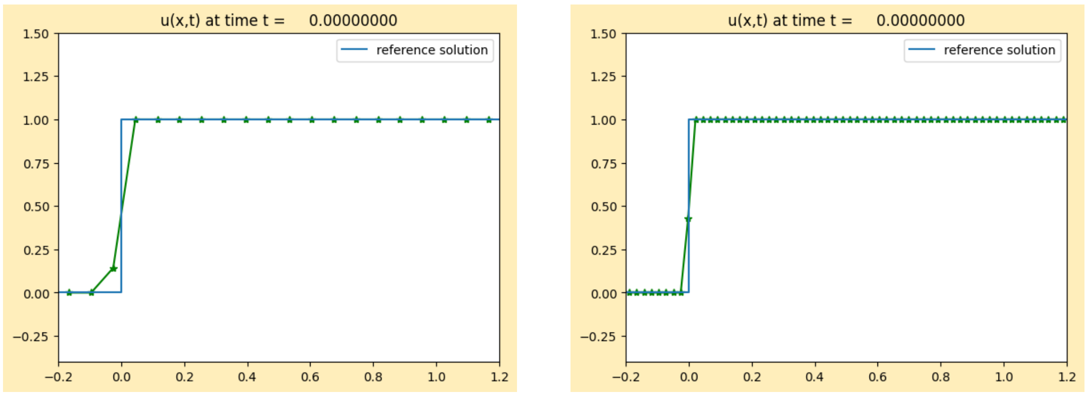
(f) AMR $N = 60$, $t = 1.0$

Figure 4.5: Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.2) with $N = 60$ and CFL$= 0.9$.

The second initial condition (4.3) of the test equation 2 (2.7) is resulting as rarefaction wave of the equation . Initial data is set as $u_L = 0$ and $u_R = 1$. In the Figure (4.6), the simulation results illustrate the propagation of wave using uniform grids with a light blue labelled reference solution. The figure presents the solution of the initial data obtained with $N = 20$ and $N = 60$ uniform grids with a fixed $CFL = 0.9$ in the spatial interval $[-0.2, 1.2]$ at $t = 0, \quad t = 0.5$ and $t = 1.0$.

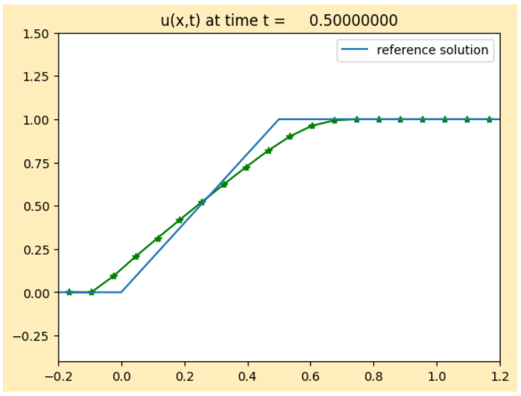
For $N = 20$ uniform grids, the spatial step size $\Delta x \approx 0.0737$ with $\Delta t \approx 0.0663$. Even at the initial time $t = 0$, the resolution is relatively coarse, which results in a less accurate depiction of the wave propagation compared to the reference solution. As the solution evolves over time, it becomes evident that the less uniform grid fails to capture the reference solution accurately compared to $N = 60$ uniform grids.

For $N = 60$ uniform grids, the spatial step size $\Delta x \approx 0.0237$ with $\Delta t \approx 0.0213$. The increased number of uniform grids significantly improves the resolution, capturing the reference solution with greater accuracy. The increased number of uniform grids allows for a more detailed representation of the rarefaction wave, accurately reflecting the values near the initially given values of $u = 1$ and $u = 0$, which $N = 20$ fails to represent. The results demonstrate that the higher grid resolution better approximates the rarefaction wave and its characteristics as it evolves over time.

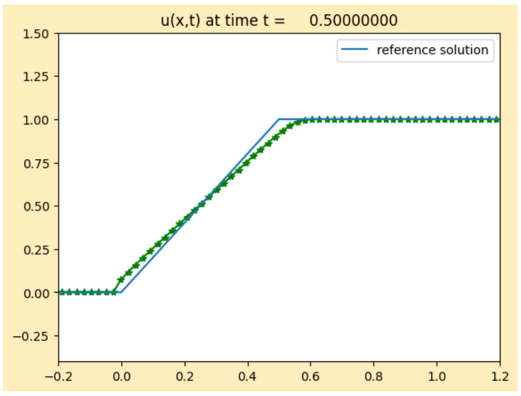


(a) Uniform $N = 20$, $t = 0.0$ (b) Uniform $N = 60$, $t = 0.0$

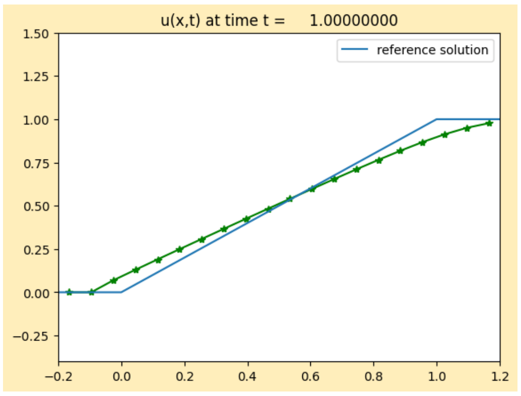
Figure 4.6: Numerical solution of inviscid Burgers equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.3) and CFL= 0.9.



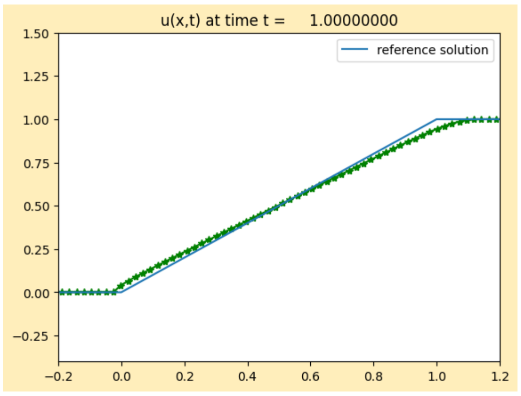
(c) Uniform $N = 20$, $t = 0.5$



(d) Uniform $N = 60$, $t = 0.5$



(e) Uniform $N = 20$, $t = 1.0$

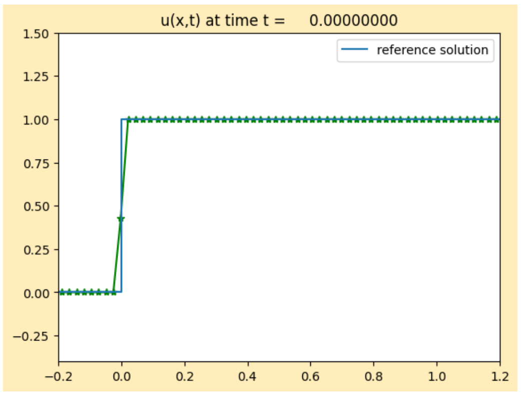


(f) Uniform $N = 60$, $t = 1.0$

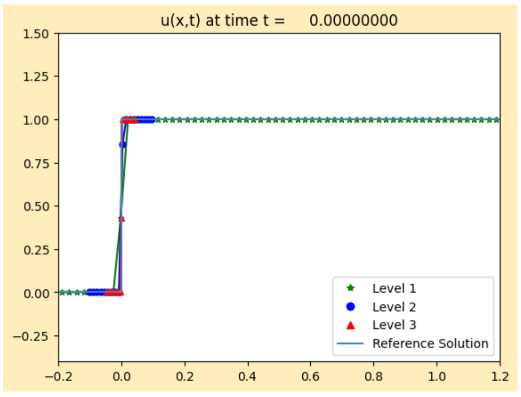
Figure 4.6: Continued.

The Figure 4.7 indicates a comparison of $N = 60$ uniform grid together with the AMR implemented solution with it at time $t = 0.0$, $t = 0.5$ and $t = 1.0$ of the rarefaction wave of the test equation (2.7). An error bound $\epsilon = 0.4$ is implemented for the 'flag2refine' criteria with a buffer zone width of 3 neighboring meshes on the AMR solution. $N = 60$ uniform grids are indicated in the figure with green stars for level 1, blue circles for level 2, and red triangles for level 3. It can be observed that the AMR solution of the rarefaction wave is efficient within the stated error bound for the initial condition. The reason behind this is that at the initial time $t = 0$, there exists a discontinuity. The AMR solution immediately refines the area around the discontinuous point with a refinement ratio $r = 2$ for level 2 and $r = 4$ for level 3. At $t = 0$, the accuracy achieved by the AMR method could be obtained with a uniform grid of 4×60 grids. As the time evolution of the solution progresses, the discontinuity evolves more smoothly. At time $t = 0.5$ and $t = 1.0$, the values of the uniform grids are accurate and get closer to each other, which leads to no need for additional mesh refinement within the predefined error bound ϵ.

Overall, it can be said that with the error bound $\epsilon = 0.4$, the $N = 60$ grids are sufficient to capture the solution accurately, meeting the predefined error bound without requiring further mesh refinement.

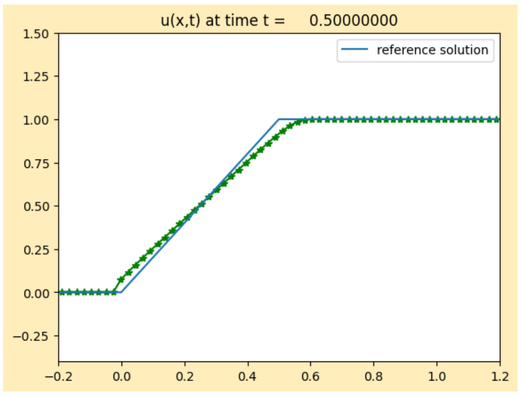


(a) Uniform $N = 60$, $t = 0.0$

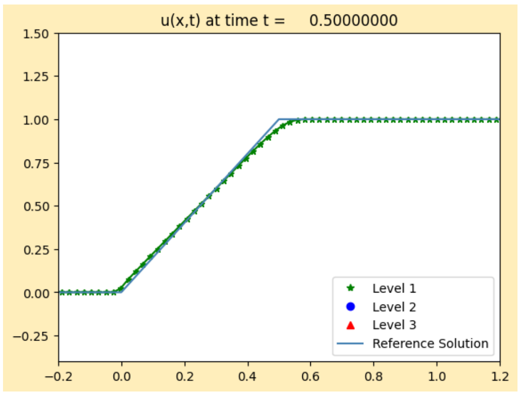


(b) AMR $N = 60$, $t = 0.0$

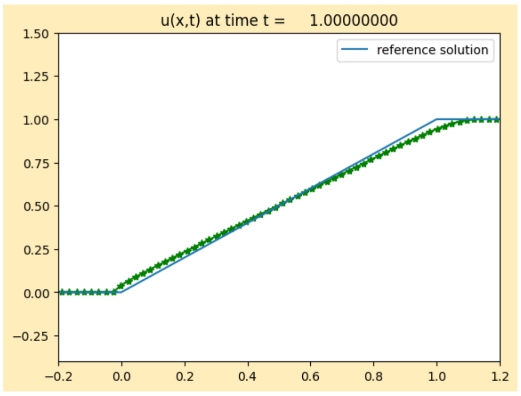
Figure 4.7: Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.3) with $N = 60$ and CFL$= 0.9$.



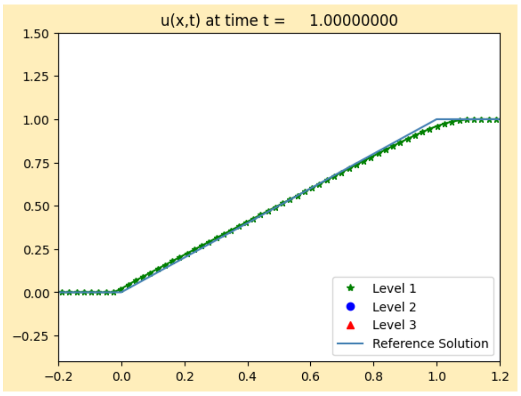
(c) Uniform $N = 60$, $t = 0.5$



(d) AMR $N = 60$, $t = 0.5$



(e) Uniform $N = 60$, $t = 1.0$



(f) AMR $N = 60$, $t = 1.0$

Figure 4.7: Continued.

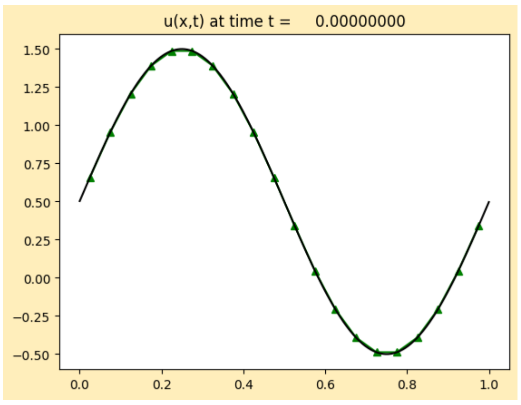
The third initial condition (4.4) for the test equation 2 (2.7) involves a sinusoidal function. Although the initial conditions are smooth, the solution evolves into an N-wave structure over time and eventually reaches a steady state characterized by a single jump discontinuity. The spatial domain $[0, 1]$ is used for this initial condition. To assess and compare the accuracy of the solutions, we use two different uniform grid resolutions: $N = 20$ and $N = 60$.

For $N = 20$ uniform grids, the grid spacing is $\Delta x \approx 0.0526$ and the time step is $\Delta t \approx 0.0316$. With $N = 60$ uniform grids, the grid spacing is reduced to $\Delta x \approx 0.0169$ and the time step to $\Delta t \approx 0.0102$. Simulations are simulated at $t = 0$, $t = 0.25$, and $t = 0.5$ with a valid $CFL = 0.9$ for each both number of uniform grids.

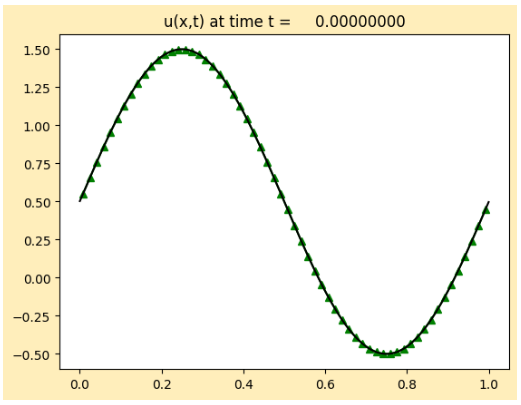
Figure 4.8 highlights the impact of increasing the number of uniform grids from $N = 20$ to $N = 60$ on solution accuracy. The reference solution is shown with a black line for comparison.

With $N = 60$ uniform grids, the solution provides a much clearer results of the sinusoidal wave by capturing finer details and reducing numerical diffusion compared to the $N = 20$ grid solution. This increased number of uniform grids allow for a more accurate representation of the wave's behavior and the sharp transitions resulting from the N-wave. Although the initial condition is captured accurately with $N = 20$ uniform grids, the accuracy of the solution decreases over time. This decrease in accuracy becomes evident at $t = 0.25$ and $t = 0.5$, as the finer details of the solution are not well represented with the lower resolution grid unlike $N = 60$ uniform grids.

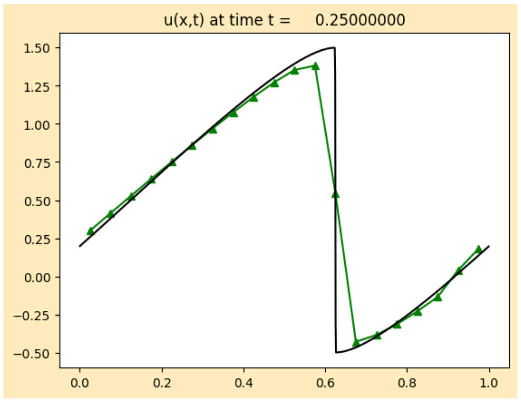
In summary, comparing $N = 20$ and $N = 60$ uniform grids shows how crucial grid resolution is for accuracy. While $N = 20$ captures the initial condition, it struggles with the finer details and sharp transitions of the N-wave as time goes on. On the other hand, $N = 60$ grids provide a much clearer and more accurate picture of the wave's behavior. This underscores the importance of using higher resolution grids to achieve precise and dependable results in simulations.



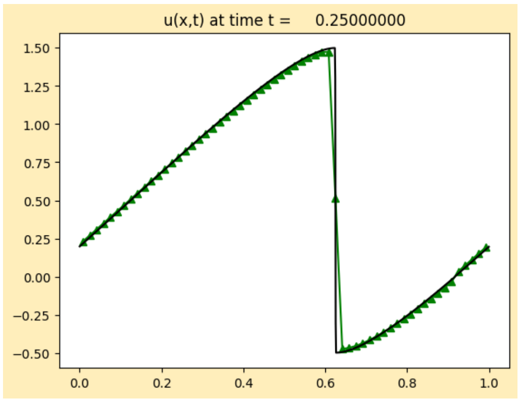
(a) Uniform $N = 20$, $t = 0.0$



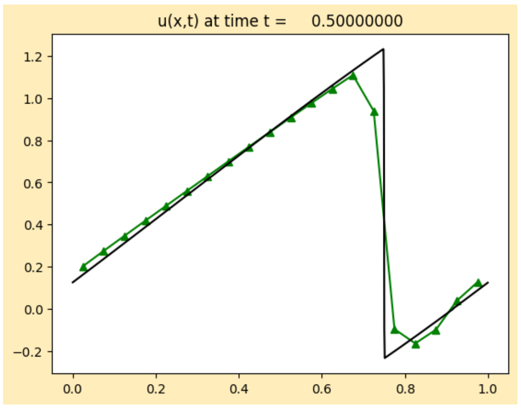
(b) Uniform $N = 60$, $t = 0.0$



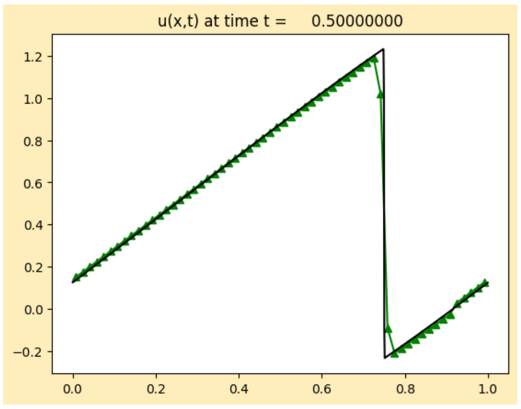
(c) Uniform $N = 20$, $t = 0.25$



(d) Uniform $N = 60$, $t = 0.25$



(e) Uniform $N = 20$, $t = 0.5$



(f) Uniform $N = 60$, $t = 0.5$

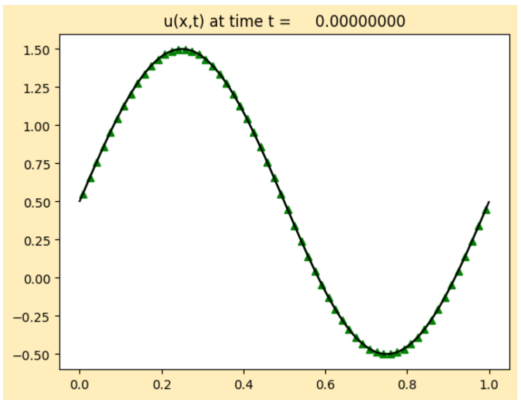
Figure 4.8: Numerical solution of inviscid Burgers equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.4) and CFL= 0.9.

The Figure 4.9 demonstrates the comparison of $N = 60$ uniform grid on the left and adaptively refined version of 60 uniform mesh at time $t = 0.0, \quad t = 0.25$ and $t = 0.5$. An error bound of $\epsilon = 0.4$ is set for the 'flag2refine' criteria with a buffer zone of 3 neighboring grids. On the left side of Figure 4.9, the uniform solution with $N = 60$ grids is shown alongside the reference solution. On the right side, the AMR-implemented solution is displayed.

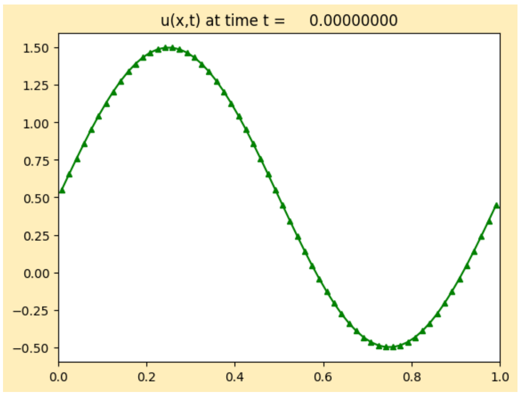
In this figure, level 1 represented by green triangles corresponds to the uniform grid solution with $N = 60$. Level 2 indicated by blue squares with a refinement ratio $r = 2$ and level 3 shown with red circles with a refinement ratio $r = 4$. At the initial condition $t = 0.0$, the error bound ϵ is not exceeded by the divided differences of any of the 3 neighboring uniform grids. Therefore, no refinement is performed at $t = 0$. By $t = 0.25$ and $t = 0.5$, the equation evolves into an N-wave structure with a single jump discontinuity forming a shock. The neighboring uniform grids around the discontinuity have divided differences that exceed the error bound, resulting in refinements at levels 2 and 3.

This adaptive refinement allows for a more detailed representation of the solution's critical features without unnecessarily increasing the computational cost. By selectively refining the grid only around the single jump discontinuity, the AMR significantly improves accuracy while maintaining efficiency.

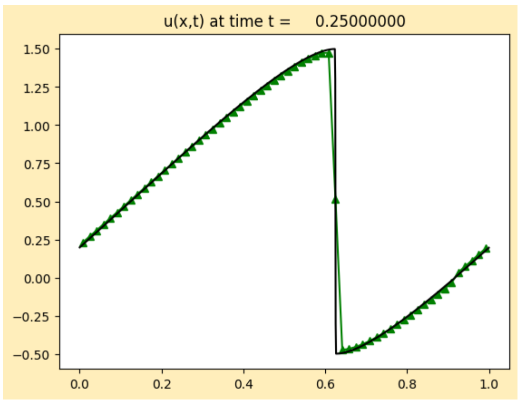
The accuracy achieved by AMR around the jump discontinuity could be matched by using a uniform grid with $N = 480$ cells across the entire spatial domain. In contrast, AMR selectively refines only the areas near the jump discontinuity, offering comparable accuracy while significantly reducing the total number of cells.



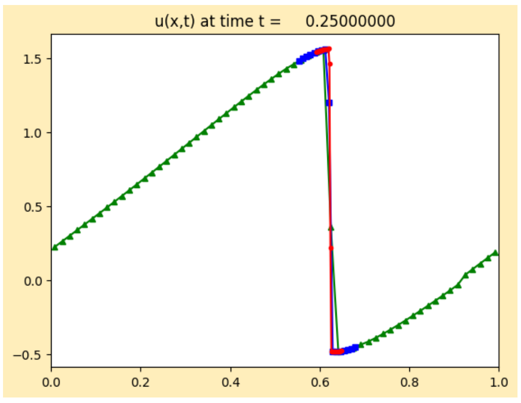
(a) Uniform $N = 60$, $t = 0.0$



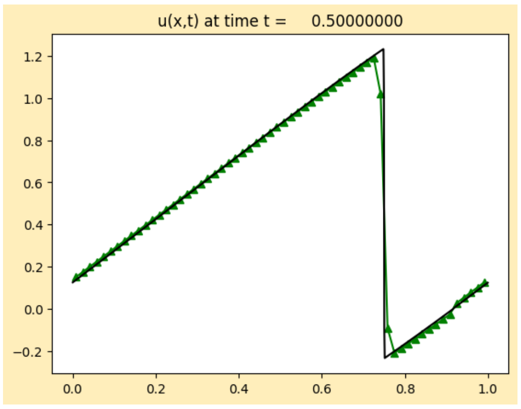
(b) AMR $N = 60$, $t = 0.0$



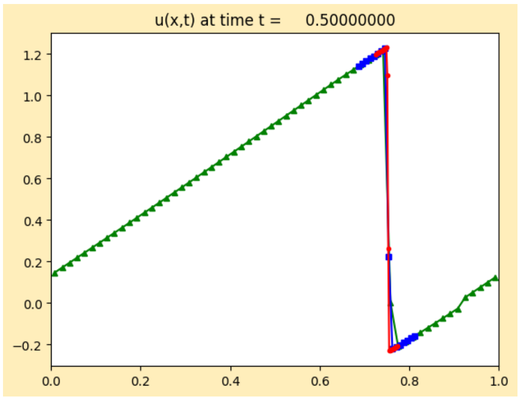
(c) Uniform $N = 60$, $t = 0.25$



(d) AMR $N = 60$, $t = 0.25$

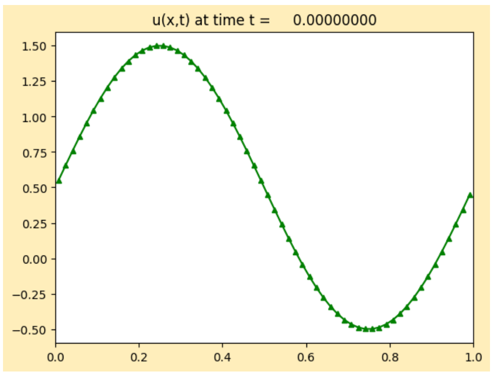


(e) Uniform $N = 60$, $t = 0.5$

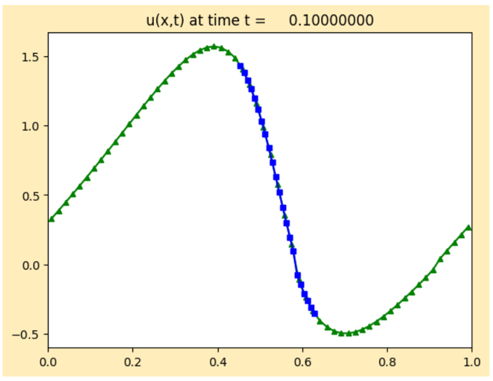


(f) AMR $N = 60$, $t = 0.5$

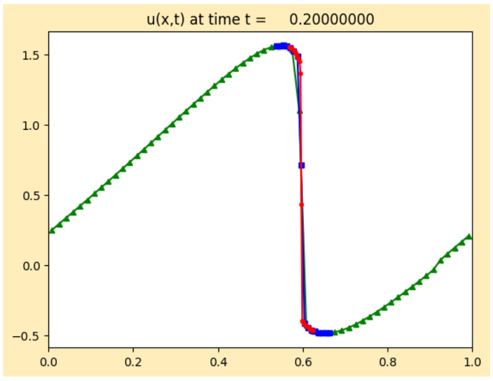
Figure 4.9: Comparison of uniform mesh and AMR for inviscid Burgers equation. Initial condition (4.4) with $N = 60$ and CFL$= 0.9$.



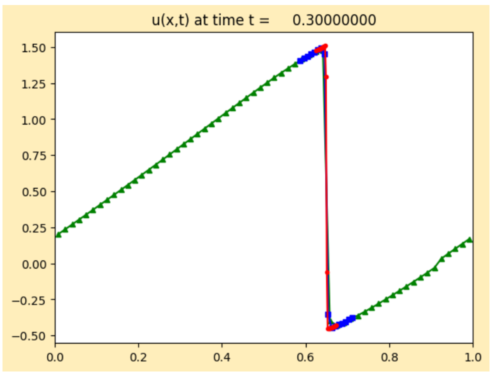
(a) AMR $N = 60$, $t = 0.0$



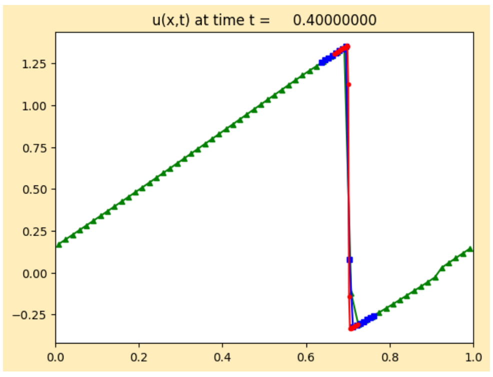
(b) AMR $N = 60$, $t = 0.1$



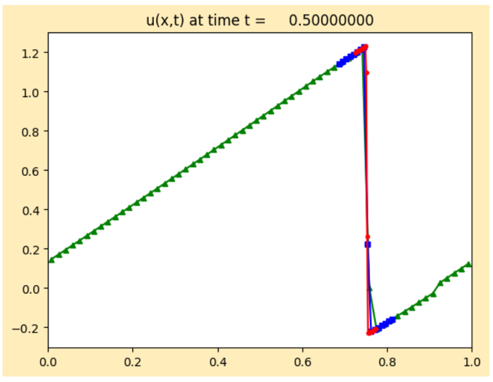
(c) AMR $N = 60$, $t = 0.2$



(d) AMR $N = 60$, $t = 0.3$



(e) AMR $N = 60$, $t = 0.4$



(f) AMR $N = 60$, $t = 0.5$

Figure 4.10: Evolution of numerical solutions by AMR for inviscid Burgers equation with $N = 60$. Initial condition (4.4) and CFL$= 0.9$.

For details on the evolution of the AMR solution for the test equation (2.7) with the sinusoidal initial condition, we present the evolution from $t = 0.0$ to $t = 0.5$ in 6 figures in Figure 4.10. At $t = 0.1$, the solution is refined to level 2 as no further refinement is required and the grids satisfy the error bounds. However, after $t = 0.1$, the formation of an N-wave begins, necessitating level 3 refinement to accurately capture the evolving features. Details beyond this are left for the readers to observe.

4.1.3 Test Equation 3: Buckley-Leverett equation

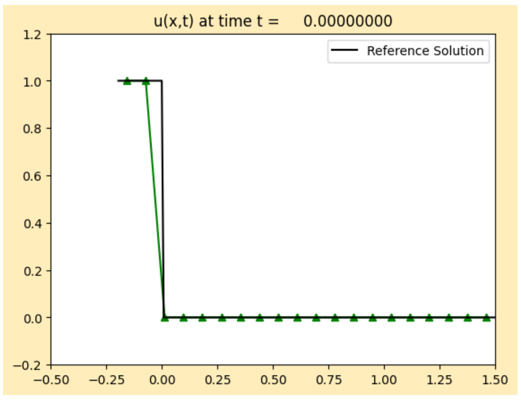
The last test equation is Buckley-Leverett equation (2.8). We consider the following initial condition

$$u(x,0) = \begin{cases} 1 & \text{if } x < 0, \\ 0 & \text{if } x > 0, \end{cases} \tag{4.5}$$

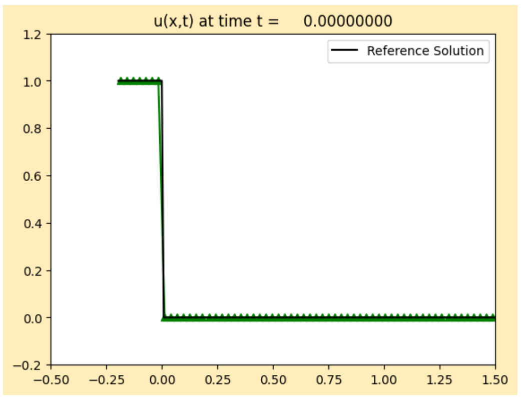
Figure 4.11 compares the evolution of the solution to the test equation 3 (2.8) at times $t = 0.0$, $t = 0.5$, and $t = 1.0$ over the spatial domain $x = [-0.2, 1.5]$ by using $N = 20$ and $N = 60$ uniform grids with a fixed CFL number of 0.9. The reference solution is provided at the same time steps in the same figure.

For $N = 20$ uniform grids, the spatial step size is $\Delta x \approx 0.089$, resulting in a time step size of $\Delta t \approx 0.0387$. In contrast, with $N = 60$ grids, the spatial step size is reduced to $\Delta x \approx 0.0288$, and the time step size becomes $\Delta t \approx 0.0125$. The reference solution is given in Figure 4.11 as black indicated line.

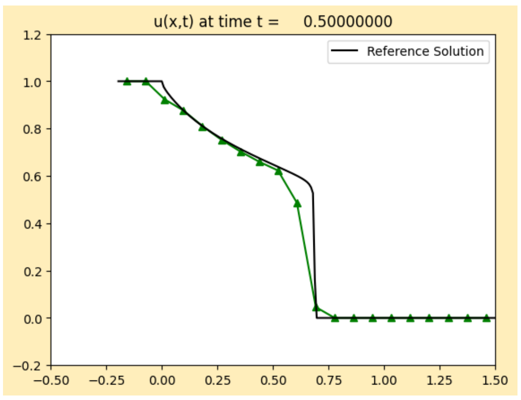
The accuracy of the numerical solution significantly improves with an increase in the number of uniform grids from $N = 20$ to $N = 60$. This improvement in accuracy becomes particularly noticeable with the higher resolution in comparison to the reference solution. While the $N = 60$ grid effectively captures both the initial shape and the evolved structure of the equation, including the formation of shocks, the $N = 20$ grid fails to accurately capture these details. As a result, the $N = 60$ grid solution maintains accuracy that is much closer to the reference solution, demonstrating the benefit of higher grid resolution in representing the dynamic features of the solution more accurately.



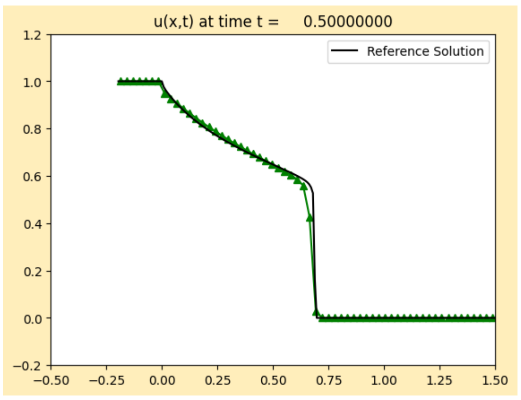
(a) Uniform $N = 20$, $t = 0.0$



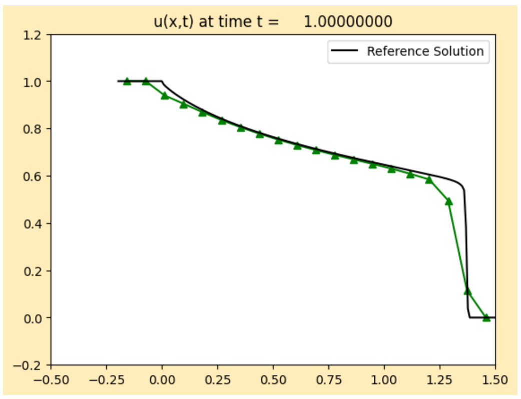
(b) Uniform $N = 60$, $t = 0.0$



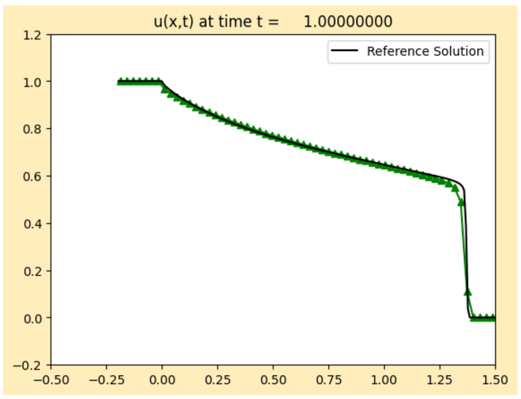
(c) Uniform $N = 20$, $t = 0.5$



(d) Uniform $N = 60$, $t = 0.5$



(e) Uniform $N = 20$, $t = 1.0$



(f) Uniform $N = 60$, $t = 1.0$

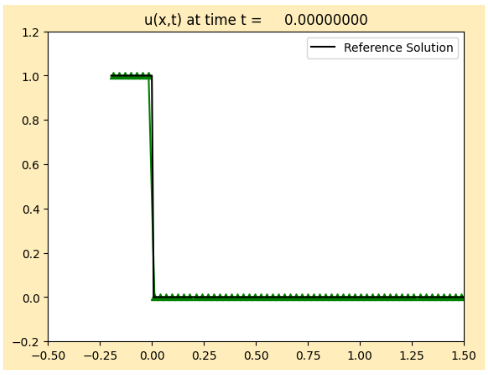
Figure 4.11: Numerical solution of Buckley-Leverett equation with uniform mesh for $N = 20$ and $N = 60$. Initial condition (4.5) and CFL= 0.9.

Figure 4.12 illustrates the application of the AMR method with the 'flag2refine' criteria. An error bound of $\epsilon = 0.4$ is used with a buffer zone containing 3 neighboring grids to determine refinement levels.

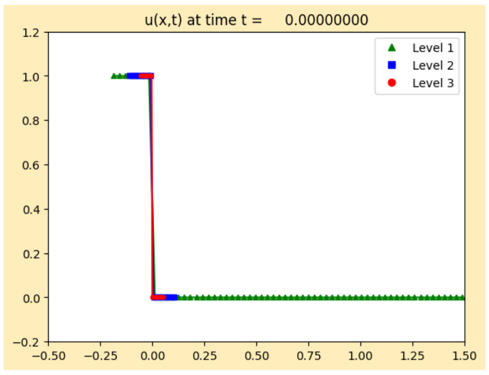
In the figure, level 1 represents the $N = 60$ uniform grids with green triangles, level 2 is indicated by blue squares with a refinement ratio of $r = 2$, and level 3 is shown with red circles with a refinement ratio of $r = 4$.

Although the $N = 60$ uniform grids offer greater efficiency compared to $N = 20$ uniform grids as shown in Figure 4.11, the solution achieved with $N = 60$ uniform grids still falls short in accurately capturing the shock formations during the evolution. Even at $t = 0$, AMR is implemented at $x = 0$ to capture the discontinuity resulting from the initial conditions $u_L = 1.0$ and $u_R = 0.0$.

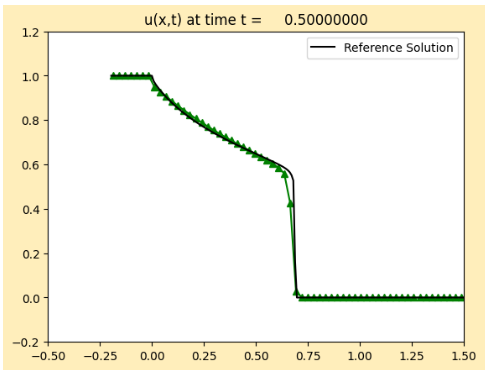
As the solution evolves, the equation (2.8) first develops rarefaction waves where the solution changes gradually. We can see at time step $t = 0.5$, the solution transitions to form sharp shock waves where the solution experiences a sharp gradient. At this stage AMR refines the neighboring grids around the shock . The method dynamically adjusts the grid resolution based on the formation of shocks, ensuring that areas with sharp gradients receive higher resolution. By $t = 1.0$, AMR process has precisely captured the shock wave structure. The AMR provides a resolution comparable to that of $N = 480$ uniform grids but with a more efficient use of computational resources. By focusing refinement around the shock regions, the AMR method improves accuracy while avoiding the high computational cost associated with uniform grids across the entire domain.



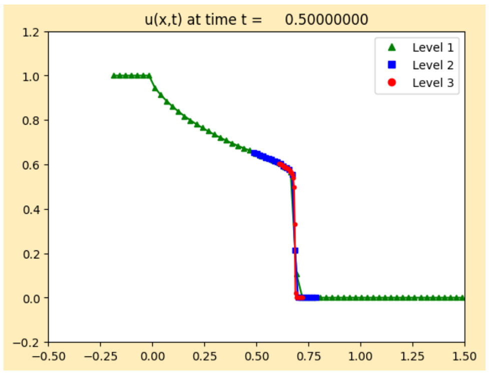
(a) Uniform $N = 60, \quad t = 0.0$



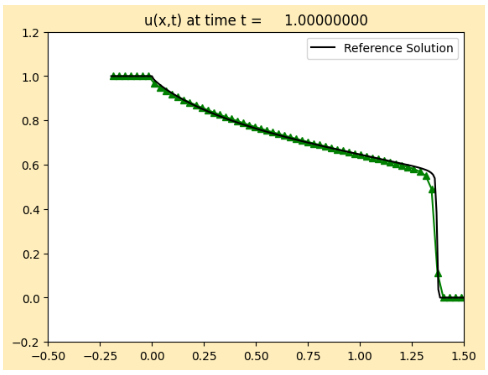
(b) AMR $N = 60$, $t = 0.0$



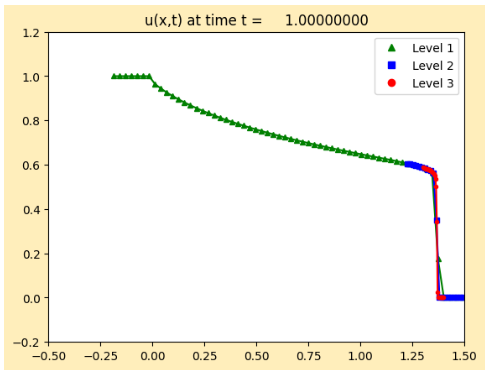
(c) Uniform $N = 60, \quad t = 0.5$



(d) AMR $N = 60$, $t = 0.5$



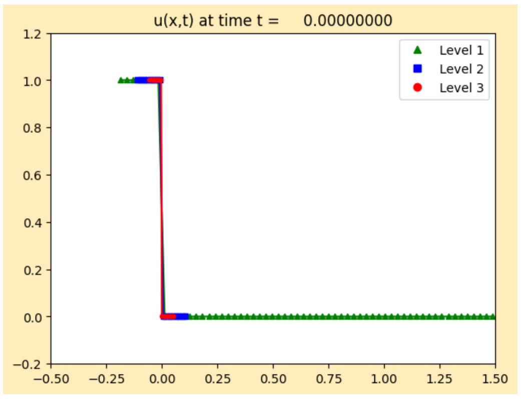
(e) Uniform $N = 60, \quad t = 1.0$



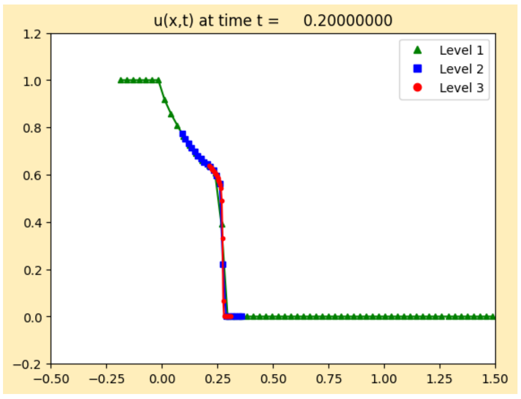
(f) AMR $N = 60$, $t = 1.0$

Figure 4.12: Comparison of uniform mesh and AMR for Buckley-Leverett equation. Initial condition (4.5) with $N = 60$ and CFL$= 0.9$.

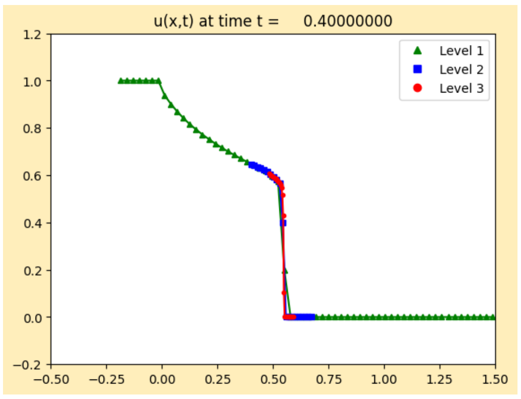
For details on the evolution of the AMR solution for the last test equation (2.8) with the initial condition (4.5) we present the evolution from $t = 0.0$ to $t = 1.0$ in 6 figures in Figure 4.13. Even from the initial condition levels 2 and 3 are used to capture the discontinuity at $x = 0$. As the solution evolves shock waves develop necessitating the continued use of level 3 refinement to accurately capture these features. Details beyond this are left for the readers to observe.



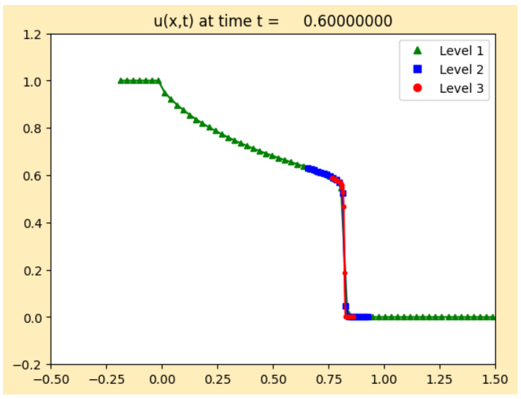
(a) AMR $N = 60$, $t = 0.0$



(b) AMR $N = 60$, $t = 0.2$

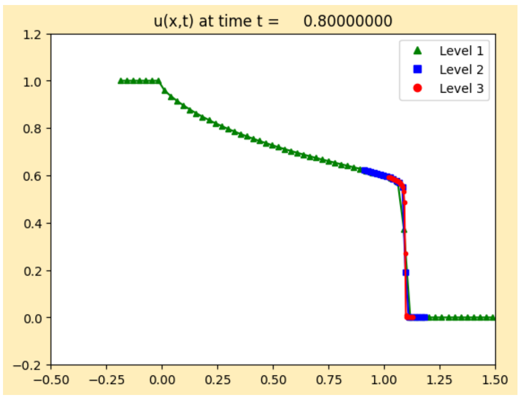


(c) AMR $N = 60$, $t = 0.4$

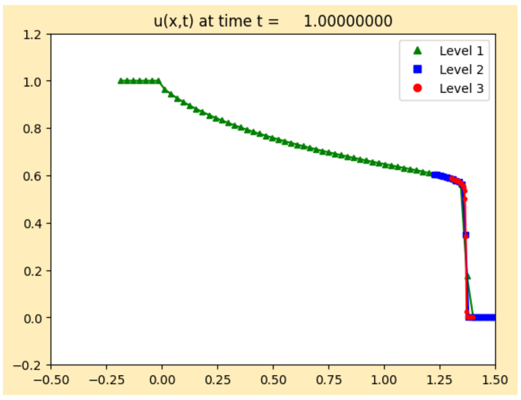


(d) AMR $N = 60$, $t = 0.6$

Figure 4.13: Evolution of numerical solutions by AMR for Buckley-Leverett equation with $N = 60$. Initial condition (4.5) and CFL$= 0.9$.



(e) AMR $N = 60$, $t = 0.8$



(f) AMR $N = 60$, $t = 1.0$

Figure 4.13: Continued.

In summary, the figures presented provide a detailed comparison across the three one dimensional scalar conservation laws: linear advection, inviscid Burgers equation, and Buckley-Leverett equation.

For the test equation 1 (2.6), the comparison between uniform grids with $N = 20$ and $N = 60$ highlights how increasing grid resolution improves the accuracy of capturing the solution. The increased number of uniform grids ($N = 60$) more closely approximates the reference solution, demonstrating an increased ability to resolve the Gaussian pulse and represent the wave with greater accuracy. Additionally, AMR solution achieves a resolution comparable to that of $N = 240$ uniform grids but with significantly fewer total grid points. AMR refines the grids specifically around the pulse, allowing for accurate capture of solution while maintaining computational efficiency.

In the case of inviscid Burgers equation (2.7) the impact of grid resolution on accuracy is particularly notable. With $N = 20$ uniform grids the solution diverges from the reference solution over time. With increased number of uniform grids, ($N = 60$) captures the shock wave and the evolving N-wave pattern with greater accuracy. It matches the reference solution more closely and effectively compared to the solution from $N = 20$ uniform grids.

For each of the three initial conditions increasing the number of uniform grids provides better resolution. AMR is particularly effective in capturing shock wave formations as the solution evolves. However for initial conditions that lead to rarefaction waves, AMR performs well at the beginning but faces difficulties as the solution evolves into a rarefaction wave. Even with $N = 60$ uniform grids the solution can be captured within an acceptable error bound.

Overall except for rarefaction scenarios AMR offers a computationally efficient solution that approximates the resolution achievable with around $N = 480$ uniform grids. AMR achieves this high level of resolution by refining only the necessary areas around sharp gradients balancing accuracy with computational cost.

For the test equation 3 (2.8), the figures reveal how grid resolution affects the depiction of the displacement front. Increased number of uniform grid ($N = 60$) provides a more accurate representation of the front's movement and structure, closely matching the reference solution, compared to the less number of uniform grid ($N = 20$). The equation initially forms rarefaction waves. As the wave speed reaches its maximum at each time step, shock waves develop and discontinuities arise at these points. AMR efficiently captures these discontinuities by focusing refinement specifically on the regions where shocks occur. This allows AMR to achieve accuracy comparable to that of $N = 480$ uniform grids while avoiding the need to refine unnecessary grids throughout the entire spatial domain.

Overall, the comparisons between $N = 20$ and $N = 60$ uniform grids illustrate that increasing the number of grids improves accuracy in numerical simulations. While increased number of uniform grids such as $N = 60$ provide a more precise representation of the solution, the comparison with AMR offers a more efficient approach. AMR selectively refines grids around discontinuities to achieve highly accurate results while optimizing computational cost. By focusing refinement on regions with sharp gradients or discontinuities, AMR balances accuracy and efficiency effectively. AMR achieves a resolution comparable to nearly 8 to 10 times that of $N = 60$ uniform grids without the extensive computational expense.

CHAPTER 5

CONCLUSION

In this study, we explored the effectiveness of AMR for solving 1D scalar conservation laws focusing on the following test equations 1: linear advection (2.6), 2: inviscid Burgers (2.7), and 3: Buckley-Leverett equations (2.8). We compared AMR to standard uniform mesh methods to see how well it performs.

AMR is implemented on uniform grid solutions using the FVM with a Godunov-type numerical flux. Our implementation of AMR with FVM in CLAWPACK, display promising results. This approach significantly increased accuracy in regions with sharp gradients or discontinuities by dynamically adjusting the grid. As a result, we capture detailed solution features with far fewer grid points compared to uniform meshes.

The comparison between AMR and uniform meshes demonstrates that AMR achieves the same level of accuracy as a uniform mesh with an increased number of grid points by refining only the necessary areas according to the refinement ratios related to the levels. As refinement levels increase, accuracy improves in line with these refinement ratios. Matching the accuracy achieved with AMR would require a significantly larger number of uniform grids. Additionally, the numerical results strongly support the theoretical framework, confirming that AMR efficiently avoids unnecessary refinement in smooth regions, focusing resources on areas with significant features such as discontinuities. This targeted approach makes AMR more efficient by directing computational effort where it is most needed.

In summary, this study highlights that AMR is a powerful and efficient tool for solving 1D hyperbolic PDEs. Its adaptivity on mesh refinement provides high accuracy and improved computational efficiency, achieving results comparable to uniform meshes with significantly fewer grid points. This makes AMR a cost-effective and efficient approach for handling complex problems.

The numerical results presented in this thesis were based on FVM techniques as implemented in CLAWPACK, which efficiently handles scalar conservation laws and AMR. For a comprehensive understanding of CLAWPACK and its algorithms, further details can be found in LeVeque's work [45].

REFERENCES

[1] S. Adjerid, J.E. Flaherty, P.K. Moore, and Y.J. Wang. High-order adaptive methods for parabolic systems. *Physica D: Nonlinear Phenomena*, 60(1):94–111, 1992.

[2] I. Babuska and M. Suri. The p and h-p versions of the finite element method, basic principles and properties. *SIAM Review*, 36(4):578–632, 1994.

[3] J. Barrett, J. Blowey, and H. Garcke. Finite element approximation of a fourth order nonlinear degenerate parabolic equation. *Numerische Mathematik*, 80, 08 1997.

[4] M. Berger and R. Leveque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM Journal on Numerical Analysis*, 35, 12 1997.

[5] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.

[6] M.J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989.

[7] S. Buckley and M. Leverett. Mechanism of fluid displacement in sands. *Transactions of the AIME*, 146, 04 2013.

[8] C. J. Budd, W. Huang, and R. D. Russell. Moving mesh methods for problems with blow-up. *SIAM Journal on Scientific Computing*, 17(2):305–327, 1996.

[9] J.M. Burgers. A mathematical model illustrating the theory of turbulence. volume 1 of *Advances in Applied Mechanics*, pages 171–199. Elsevier, 1948.

[10] C. Chen, F. Xiao, and X. Li. An adaptive multimoment global model on a cubed sphere. *Monthly Weather Review*, 139(2):523 – 548, 2011.

[11] M. Ciment. Stable difference schemes with uneven mesh spacings. *Mathematics of Computation*, 25(114):219–227, 1971.

[12] R. Courant, K. Friedrichs, and H. Lewy. Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74, 1928.

[13] L.C. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 2010.

[14] E. Godlewski and P. A. Raviart. Numerical approximation of hyperbolic systems of conservation laws. *Applied Mathematical Sciences*, 1996.

[15] W. Gui and I. Babuška. The h, p and h-p versions of the finite element method in 1 dimension - part ii. the error analysis of the h-and h-p versions. *Numerische Mathematik*, 49(6):613 – 657, 1986. Cited by: 128.

[16] W. Gui and I. Babuška. The h, p and h-p versions of the finite element method in 1 dimension - part iii. the adaptive h-p version. *Numerische Mathematik*, 49(6):659 – 683, 1986.

[17] W. Gui and I. Babuška. The h,p and h-p versions of the finite element method in 1 dimension - part i. the error analysis of the p-version. *Numerische Mathematik*, 49(6):577 – 612, 1986. Cited by: 220.

[18] R. Haberman. *Applied Partial Differential Equations: With Fourier Series and Boundary Value Problems*. Featured Titles for Partial Differential Equations. Pearson, 2013.

[19] D.F. Hawken, J.J. Gottlieb, and J.S. Hansen. Review of some adaptive node-movement techniques in finite-element and finite-difference solutions of partial differential equations. *Journal of Computational Physics*, 95(2):254–302, 1991.

[20] P. Houston and E. Süli. hp-adaptive discontinuous galerkin finite element methods for first-order hyperbolic problems. *SIAM Journal on Scientific Computing*, 23(4):1226–1252, 2001.

[21] W. Huang, Y. Ren, and R. D. Russell. Moving mesh partial differential equations (mmpdes) based on the equidistribution principle. *SIAM Journal on Numerical Analysis*, 31(3):709–730, 1994.

[22] W. Huang and R. D. Russell. *Adaptive Moving Mesh Methods*, volume 174. 01 2011.

[23] A. Kurganov and E. Tadmor. New high-resolution central schemes for nonlinear conservation laws and convection–diffusion equations. *Journal of Computational Physics*, 160:241–282, 05 2000.

[24] R. J. LeVeque. Wave propagation algorithms for multidimensional hyperbolic systems. *Journal of Computational Physics*, 131(2):327–353, 1997.

[25] R. J. Leveque. Nonlinear Conservation Laws and Finite Volume Methods. In *Saas-Fee Advanced Course 27: Computational Methods for Astrophysical Fluid Flow.*, page 1, January 1998.

[26] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 2002.

[27] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.

[28] S. Li and J.M. Hyman. Adaptive mesh refinement for finite difference weno schemes. *Los Alamos Report LA-UR-03*, 89272003, 2003.

[29] Y. Li, D. Jeong, and J. Kim. Adaptive mesh refinement for simulation of thin film flows. *Meccanica*, 49, 01 2014.

[30] J. D. Logan. *Applied partial differential equations*. Springer, 2014.

[31] S. F. McCormick. Multilevel adaptive methods for partial differential equations. In *Frontiers in applied mathematics*, 1989.

[32] K. Miller and R. N. Miller. Moving finite elements. i. *SIAM Journal on Numerical Analysis*, 18(6):1019–1032, 1981.

[33] B. Okutmuştur. Scalar conservation laws. In *Advanced Computational Fluid Dynamics for Emerging Engineering Processes*, chapter 4. IntechOpen, Rijeka, 2019.

[34] B. Ong, R. Russell, S. Ruuth, and Jean-Christophe Weill. An h-r moving mesh method for one-dimensional time-dependent pdes. In *Proceedings of the 21st International Meshing Roundtable*, pages 39–54, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[35] J. R. Philip. Dynamics of fluids in porous media. elsevier, 1972. 764 pp. dfk 94.00. fundamentals of transport phenomena in porous media. elsevier, 1972. 392 pp. dfl 65.00 differential equations of hydraulic transients, dispersion, and ground water flow. by wen-hsiung li. prentice-hall, 1972. 316 pp. *Journal of Fluid Mechanics*, 61(1):206–208, 1973.

[36] M.D. Piggott, C.C. Pain, G.J. Gorman, P.W. Power, and A.J.H. Goddard. h, r, and hr adaptivity with applications in numerical ocean modelling. *Ocean Modelling*, 10(1):95–113, 2005. The Second International Workshop on Unstructured Mesh Numerical Modelling of Coastal, Shelf and Ocean Flows.

[37] T. Plewa, T. Linde, and V.G. Weirs. *Adaptive Mesh Refinement - Theory and Applications: Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3-5, 2003*. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2005.

[38] A. Schwing, I. Nompelis, and G. Candler. Implementation of adaptive mesh refinement in an implicit unstructured finite-volume flow solver. 06 2013.

[39] C. Shen, J.Qiu, and A. Christlieb. Adaptive mesh refinement based on high order finite difference weno scheme for multi-scale simulations. *Journal of Computational Physics*, 230(10):3780–3802, 2011.

[40] W. Skamarock, J. Oliger, and R. L. Street. Adaptive grid refinement for numerical weather prediction. *Journal of Computational Physics*, 80(1):27–60, 1989.

[41] W.A. Strauss. *Partial differential equations: An introduction*. John Wiley & Sons, 2007.

[42] P. Sun, R. D. Russell, and J. Xu. A new adaptive local mesh refinement algorithm and its application on fourth order thin film flow problem. *Journal of Computational Physics*, 224(2):1021–1048, 2007.

[43] H. Tang and T. Tang. Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws. *SIAM Journal on Numerical Analysis*, 41(2):487–515, 2003.

[44] T. Tang. Moving mesh methods for computational fluid dynamics. *Contemp. Math.*, 383, 01 2005.

[45] Clawpack Development Team. Clawpack Software, Version 5.9.2. http://www.clawpack.org, 2024.

[46] E.F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction*. Springer Berlin Heidelberg, 2009.

[47] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Pearson Education Limited, 2007.

[48] E. Vázquez-Cendón. *Solving Hyperbolic Equations with Finite Volume Methods*, volume 90. 01 2015.

[49] L. Wan-Lung and Z. Tan. Moving mesh methods for boussinesq equation. *International Journal for Numerical Methods in Fluids*, 61, 2009.

[50] Y. Zeng, A. Xuan, J. Blaschke, and L. Shen. A parallel cell-centered adaptive level set framework for efficient simulation of two-phase flows with subcycling and non-subcycling. *Journal of Computational Physics*, 448:110740, 2022.