UNIVERSIDAD DE INVESTIGACIÓN DE TECNOLOGÍA EXPERIMENTAL YACHAY

Escuela de Ciencias Matemáticas y Computacionales

TÍTULO: Level Set Methods Enhancement Based on Artificial Intelligence

Trabajo de integración curricular presentado como requisito para la obtención del título de Matemático

Autor/a:

Chachalo Perugachi Roberth Adrián

Tutor/a:

PhD.Manzanilla Morillo Raúl

Urcuquí, Diciembre 2023

Autoría

Yo, **Roberth Adrián Chachalo Perugachi**, con cédula de identidad 1004529028, declaro que las ideas, juicios, valoraciones, interpretaciones, consultas bibliográficas, definiciones y conceptualizaciones expuestas en el presente trabajo; así cómo, los procedimientos y herramientas utilizadas en la investigación, son de absoluta responsabilidad de el/la autor/a del trabajo de integración curricular. Así mismo, me acojo a los reglamentos internos de la Universidad de Investigación de Tecnología Experimental Yachay.

Urcuquí, Diciembre, 2023.

Firmado electrónicamente por:
ROBERTH ADRIAN
CHACHALO PERUGACHI

Roberth Adrián Chachalo Perugachi
CI: 1004529028

Autorización de publicación

Yo, **Roberth Adrián Chachalo Perugachi**, con cédula de identidad 1004529028, cedo a la Universidad de Investigación de Tecnología Experimental Yachay, los derechos de publicación de la presente obra, sin que deba haber un reconocimiento económico por este concepto. Declaro además que el texto del presente trabajo de titulación no podrá ser cedido a ninguna empresa editorial para su publicación u otros fines, sin contar previamente con la autorización escrita de la Universidad.

Asimismo, autorizo a la Universidad que realice la digitalización y publicación de este trabajo de integración curricular en el repositorio virtual, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación

Urcuquí, Diciembre, 2023.

Firmado electrónicamente por:
ROBERTH ADRIAN
CHACHALO PERUGACHI

Roberth Adrián Chachalo Perugachi
CI: 1004529028

Dedication

I want to thank my mother for supporting me at every stage of my life, for giving me the strength I needed in the most difficult moments, and for helping me reach this goal. I want to thank my grandmother, who is not with us today, but left me a lesson in life, always fighting for my dreams in the face of all difficulties. I love my aunts and their political uncle for supporting me and giving me the tools to achieve this great goal. And I want to thank my friends for sharing great moments and teachings on this path.

Roberth Adrián Chachalo Perugachi

Acknowledgment

First, I want to thank Professor Israel Pineda for teaching me all the knowledge described in this project and helping me grow academically and personally. In the same way, I want to thank Professor Raúl Manzanilla for contributing valuable knowledge to finish this project and my friends, especially Cristina Godoy and Jaime Astudillo, for their wise advice and unwavering support.

Roberth Adrián Chachalo Perugachi

Resumen

Este proyecto se enfocó en desarrollar modelos continuos y discretos de redes neuronales basados en principios físicos para resolver un problema de ecuación diferencial parcial conocido como ecuación de conjunto de nivel; en el contexto del método de conjunto de nivel , sin condiciones de borde. La red neuronal se entrena utilizando datos de soluciones numéricas obtenidas a partir de métodos de diferencias finitas, incluyendo el método de difusión ascendente de primer orden y los métodos de orden superior, como el esencialmente no oscilatorio y el esencialmente no oscilatorio ponderado para la discretización espacial, así como el método Runge-Kutta de tercer orden con disminución de variación total para la discretización temporal. Además, se emplea el método de características para obtener la solución analítica del problema, que se utiliza para comparar con las soluciones numéricas y las predicciones de la redes neuronales desarrolladas. Las soluciones inferidas por los modelos son relativamente buenas desde el punto de vista de los errores $L1$ y $L2$; pero la calidad de los datos en el modelo discreto no fue significativa para mejorar la inferencia, a diferencia del modelo continuo el cual presentó una mejora adecuada cuando la red neuronal se entrenó con datos ENO. Sin embargo, estos resultados no tiene una apropiada aproximación de la solución exacta; y esto se debe a que el error cuadrático medio de las condiciones de borde no fueron consideradas en la formulación de la función de costo. Las condiciones de borde conocidas como salida y entrada se implementarán en trabajos futuros para mejorar la inferencia de soluciones para la ecuación de nivel de conjunto.

Palabras Clave:

Método de conjunto de nivel , Ecuación de conjunto de nivel, Redes neuronales informadas por la física, Difusión ascendente, Método esencialmente no oscilatorio, Método esencialmente no oscilatorio ponderado, Runge Kutta de disminución de variación total.

Abstract

This project focused on developing continuous and discrete time models of neural networks based on the physics-informed neural network (PINNS) to solve a one-dimensional partial differential equation problem known as the level set equation; in context of level set method, without boundary conditions. The neural network is trained using data from numerical solutions obtained from finite difference methods, including the the first-order accuracy upwind method and high-order Essentially Non-oscillatory (ENO) and Weighted Essentially Non-oscillatory (WENO) methods for discretizing space, as well as third-order accuracy TVD-Runge-Kutta method for the discretizing time. The method of characteristics is also employed to obtain the analytical solution to the problem, which is used for comparison with the numerical solutions and the predictions of the developed neural network. The inferred solutions by models are relatively good from the point of view of $L1$ and $L2$ errors; but the quality of data for the discrete-time model was not significant to improve the inference, in contrast, the continuous model presented a suitable improvement when the neural network was training by ENO data. However, these results in the context of the accuracy of approximation for exact solution are poor; and this is because the mean square error of boundary conditions in the loss function was not considered. Inflow and outflow boundary conditions will be implemented in future works to improve the inference of solutions for the level-set equation.

Keywords:
Level Set Method, Level Set Equation, Physics Informed Neuronal Networks, Upwind , Essentially Non-oscillatory, Weighted Essentially Non-oscillatory, Total Variation Diminishing Runge Kutta.

Contents

Dedication **v**

Acknowledgment **vii**

Resumen **ix**

Abstract **xi**

Contents **xiii**

List of Tables **xvii**

List of Figures **xix**

1 Introduction **1**

1.1 Background 1

1.2 Problem Statement 3

1.3 Objectives 4

1.3.1 General Objective 4

1.3.2 Specific Objectives 4

2 Theoretical Framework **7**

2.1 Hamilton Jacobi Equations 7

2.2 Level Set Method 8

2.3 Signed Distance Function 9

2.4 Finite Differences Methods 12

2.4.1 Upwind Method 13

2.4.2 Essentially Non-oscillatory 14

2.4.3 Weighted Essentially Non-Oscillatory 14
2.4.4 Total Variation Diminishing Runge Kutta 15
2.5 Neural Networks 17
2.5.1 Physics Informed Neural Networks 18

3 State of the Art 23
3.1 Studies on High-order Finite Differences Methods 23
3.2 Extensions on Level Set Method 24
3.3 Physics Informed Neuronal Networks on Fluid Dynamics 25
3.4 Preliminaries Studies on Tracking Problems 27

4 Methodology 29
4.1 Description Problem 29
4.2 Analytic Solution of 1D Level Set Equation 35
4.3 Finite Difference Method 37
4.3.1 Upwind Schemes 37
4.3.2 Essentially Non-oscillatory Schemes 39
4.3.3 Weighted Essentially Non-oscillatory Schemes 42
4.3.4 TVD-Runge Kutta Scheme 45
4.4 Physics Informed Neural Networks for Level Set Method 46
4.4.1 Discrete Time Model 46
4.4.2 Continuous Time Model 48
4.5 Data 49

5 Results and Discussion 51
5.1 Numerical Solutions of Upwind Schemes 52
5.2 Numerical Solutions of ENO Schemes 57
5.3 Numerical Solutions of WENO Schemes 61
5.4 Numerical Solutions of Level Set Method-Physics Informed Neural Networks 66
5.4.1 Discrete Model 66
5.4.2 Continuous Model 74
5.5 Discussion 78

6 Conclusions **81**

6.1 Recomendations 82

6.2 Future Works 82

6.3 Limitations 82

Bibliography **85**

List of Tables

4.1 Coefficients for $\phi_{x,i}^{-,s}$ and $\phi_{x,i}^{+,s}$ for $s = 0, 1, 2$ 40
4.2 Data Structure 50

5.1 Results of numerical experiments for the first-order upwind scheme. 54
5.2 Results of numerical experiments for third order accuracy of TVD/Runge Kutta coupled with ENO3. 57
5.3 Results of numerical experiments for third order accuracy of TVD/Runge Kutta coupled with WENO5. 61
5.4 Results from training neural network with different data. 66
5.5 Numerical experiments based on increasing layers, hidden layers, and neurons in each hidden layer. 67
5.6 Results of training with different data for continuous time model. 74

List of Figures

2.1 Motion under curvature of the front. Source [1] 8

2.2 a. Representation of interface by zero level of signed distance function, b. The interface of (a). Source [2, 3] 9

2.3 On the left side is a representation of a signed distance function, and on the right side is the interface of SDF. Source [4] 10

2.4 Representation of level set function 11

2.5 Uniform partition of interval $[0, 1]$. Source [5] 12

2.6 Representation of a simple neuronal network 18

4.1 Tracking the zero level of explicit function $\phi(x, y) = x^2 + y^2 - 1$ using Level Set Method 30

4.2 Spurious oscillation on the numerical solution. Source [6] 32

4.3 a. Initial interface. b. One revolution of the circle using the forward Euler method and upwind differences. Source [7] 33

4.4 Single vortex test to show the efficiency of WENO5. Souce [8] 33

4.5 A characteristic curve. 35

4.6 Grid for the discretization of Level Set Equation 37

4.7 Left based stencil and s^{th} stencils for construct $\phi^-_{x,i}$. Source [9] 40

4.8 Right based stencil and s^{th} stencils for construct $\phi^+_{x,i}$. source [9] 41

4.9 Neuronal Network Structure of Level Set Method for One-Dimensional . . 48

5.1 A initial condition given by a signed distance function $\phi_0(x)$. 52

5.2 Derivative of the signed distance function $\phi_0(x)$ respect to x. 52

5.3 The exact solution of (4.1) at time 50. 53

5.4 (a) Numerical and exact solution at $T = 50$ for $N = 20$ (b) The L_1 error for each point between the exact and numerical solution $T = 50$. 54

5.5 (a) Numerical and exact solution at $T = 50$ for $N = 40$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 54

5.6 (a) Numerical and exact solution at $T = 50$ for $N = 80$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 55

5.7 (a) Numerical and exact solution at $T = 50$ for $N = 160$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 55

5.8 (a) Numerical and exact solution at $T = 50$ for $N = 320$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 56

5.9 L_1 error for each solution at $T = 50$ that depends on size of grid; $N = 20$, $N = 40$, $N = 80$, $N = 160$, $N = 320$ using Upwind Schemes. 56

5.10 (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 20$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 58

5.11 (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 40$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 58

5.12 (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 80$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 59

5.13 (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 160$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 59

5.14 (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 320$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 60

5.15 L_1 error for each solution at $T = 50$ that depends on the size of the grid; $N = 20$, $N = 40$, $N = 80$, $N = 160$, $N = 320$ using the third order TVD-Runge Kutta and Essentially Non-oscillatory schemes. 60

5.16 (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 20$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 62
5.17 (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 40$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 62
5.18 (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 80$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 63
5.19 (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 160$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 63
5.20 (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 220$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$. 64
5.21 L_1 error for each solution at $T = 50$ that depends on the size of the grid; $N = 20$, $N = 40$, $N = 80$, $N = 160$, $N = 320$ using the third order Total Variation Diminishing (TVD)-Runge Kutta (RK) and the fifth order Weighted Essentially Non-oscillatory (WENO). 64
5.22 Solution at $T = 50$ for each numerical method: Upwind, TVD/RK3-ENO3 and TVD/RK3-WENO5. 65
5.23 L_1 error for each solution at $T = 50$ generated by Upwind, the TVD/RK3 coupled with ENO3 and WENO5 using a size of grid $N = 320$. 65
5.24 Exact, numerical and inferred solution using Upwind data. 68
5.25 Exact, numerical and inferred solution using TVD/RK3-ENO3 data. . . . 68
5.26 Exact, numerical and inferred solution using TVD/RK3-WENO5 69
5.27 L_1 errors computed from exact solution and inferred solutions generated by discrete time model 69
5.28 Experiments of layers: (a), (b), (c) are graphs of inferred solution generated by discrete time model when the NN is feeding with data simulated by the first-order upwind scheme. 70

5.29 Experiments of layers: (a), (b), (c) are graphs of inferred solution generated by discrete time model when the NN is feeding with data simulated by TVD/RK3-ENO3. 71

5.30 Experiments of layers: (a), (b), (c) are graphs of the inferred solution generated by the discrete time model when the NN is feeding with data simulated by TVD/RK3-WENO5. 72

5.31 Numerical solutions generated by traditional methods; Upwind, Essentially non-oscillatory, and weighed essentially non-oscillatory, compared inferred solutions from discrete model LSM-PINNs at $T = 50$. 73

5.32 Inferred solution by the continuous model with training data generated by Upwind. 75

5.33 Inferred solution by the continuous model with training data generated by TVD/RK3-ENO3. 75

5.34 Inferred solution by the continuous model with training data generated by TVD/RK3-WENO5 76

5.35 L_1 errors computed from exact solution and inferred solution generated by continuous model; and L_1 error of solution from TVD/RK3-WENO5 at $T = 50$. 76

5.36 Numerical solutions generated by traditional methods; Upwind, Essentially non-oscillatory, and weighed essentially non-oscillatory, compared inferred solutions from continuous model LSM-PINNs at $T = 50$. 77

Chapter 1

Introduction

1.1 Background

The growth of numerical methods within the field of mathematics has transformed the way to solve complex problems. Along with the development of computer programs and sophisticated algorithms, numerical methods play a crucial role within modern-day mathematical modeling [10]. One of the significant parts of numerical methods is the partial differential equation (PDE), which entails partial derivatives of an unknown function of two o more free variables. A typical example of a PDE is the wave equation, which governs various physical phenomena, such as aerodynamics, electrodynamics, acoustics, and elasticity [11]. The finite element, finite volume, and finite difference methods are some of the traditional numerical methods that have emerged to tackle an approximate solution of particular PDEs. These methods are widely used in different fields, including physics, engineering, and mathematics, to solve complex real-world problems [12]. As technology evolves, numerical methods will also advance, founding more accurate and efficient solutions to complex mathematical problems.

The research of partial differential equations is necessary for understanding the behavior of complex physical phenomena. There are three principal categories of PDEs depending on the behavior of their solutions; parabolic, hyperbolic, and elliptic. Hyperbolic equations are often linked with wave propagation phenomena, including sound, electromagnetic, seismic, etc. Also, their solutions can show wave-like behavior and finite propagation speed [13]. On the other hand, elliptic equations govern steady phenomena, such as the deformation of elastic material under equilibrium conditions and the static temperature distribution, among others [14, 15]. Finally, parabolic equations govern time-dependent diffusion or heat transfer process, and their solutions are smooth every time [14]. Classifying PDEs based on their behavior is crucial to choosing suitable numerical problem-solving methods.

The Hamilton Jacobi equations are hyperbolic partial differential equations involving some applications of optics, mechanics, and semi-classical quantum theory. Mainly, this type of equation allows us to model the motion of fronts propagation on fluid dynamics [1]. For example, Kun Lao, Changxiao Shao et el. [16] simulated the impact of a droplet on a hot plate to track the behavior of the droplet was represented as an interface. The tacking of the motion of interface through Hamilton Jacobi formulation is most used in computational fluid dynamics [17].

Two significant cases of Hamilton Jacobi formulation are studied in [1, 18]. First, when the motion of the interface depends on its local curvature, Osher and Sethian [1] developed numerical experiments to track and capture, at different times, the evolution of the interface. Second, when the motion of the interface depends on an external velocity field, Israel Pineda and Oubong Gwun [19] simulated the growth process of leaves where a vector field leads the evolution of the leaf growth. In any of the two contexts, the term Level set method is described as a technique used to capture the interface; then LSM is a Hamilton Jacobi formulation.

The level set method is a powerful tool for tracking and capturing the interface in simulations involving local curvature and external velocity fields. The advantages of this method are based on defining the interface as a level set of an initial high-dimensional function, which typically is a signed distance function; this formulation allows captures the evolution of an interface if it suffers severe changes in topology, for example, two interfaces split or merge under a vector field [1, 18, 7]. The level set method is suitable to simulate two-phase flow [20]; mainly, the simulation of cysts can be described as dual-fluid entities delineated by a bi-layer membrane. Moreover, Vincent Doyeux, Yann Guyot, et al. in [20] used Finite Elements Method to solve LSM, which is more flexible.

To solve LSM, the level set equation must be resolved because this PDE is responsible for evolving the interface. Three main methods for approximating the spatial derivatives are studied; finite differences method (FDM), finite elements method (FEM), and finite volume method (FVM). Jiang and Shu [21] developed high-order schemes based on the finite differences method, which are called ENO and WENO, to achieve third-order and fifth-order accuracy. However, the traditional upwind procedures are suitable for solving simple simulations. In the context of FEM, Vorgelegt Von [22] developed the variational formulation (VF) of the level set method for capturing interfaces with applications in two-phase flow problems; to find solutions of VF, the author used the Standard Galerkin method and Discontinuous Galerkin method which are classical finite elements methods. On the other hand, Néstor Balcázar, Lluís Jofre, Oriol Lehmkuhl, et al. [23] used a conservative finite-volume approximation to discretize the traditional level set equation. Moreover, LSE involves derivatives concerning time; Shu and Osher [24] developed a total variation diminishing Runge Kutta (TVD- RK) method that, together with FDM, solves the LSE, in the same way, the numerical experiments proposed in [22] using the combination Crank-Nicolson method and Standard Galerkin method to make simulations of two-phase flow.

On the other hand, a new method was introduced by [25] to infer solutions of partial differential equations based on neuronal networks. Dissanayake and Phan-Thien [26] first introduced neural-network-based approximations to solve partial differential equations; and in recent years, with the development of new techniques of machine learning, this topic began to emerge with intensity [27]. Raissi et al. [25] developed a new method that allows the conservation of the laws in physics imposed by PDEs. PINNs are deep neural networks that are trained with few data generated by traditional numerical methods; the key idea is to define the loss function based on mean squared errors (MSE) of initial data, boundary condition, and residual of PDEs; it allows to minimize the loss function to obtain suitable inferences for solutions of PDEs [25]. In the context of tracking problems, several research has been developed in [28, 29, 30, 31, 32] using PINNs.

1.2 Problem Statement

The level set method allows us to study the motion of curves or surfaces represented by a zero-level set of an implicit high-dimensional function. This is achieved by tracking the evolution of this function using a partial differential equation (PDE) called the level set equation. Sethian [33] introduced this approach to solve problems involving complex geometries in various areas such as fluid dynamics, control theory, and microchip manufacturing. In this context, finite differences and finite element methods are the most commonly used numerical methods for tracking problems.

Various numerical methods involve finite differences to solve the spatial derivatives in the level set equation. High-order schemes are predominantly used to study the evolution of complex interfaces subjected to severe topology changes, for example, in the simulation of two-phase flows. However, these methods may present potential challenges related to computational costs, accuracy, efficiency, and stability [34, 35]. Therefore, research in this field is of significant importance in finding new variations of existing numerical methods to improve simulations of real-world problems [36].

This project proposes two feed-forward neural network models to infer one-dimensional solutions of the level set equation based on physics-informed neural networks (PINNs). The project also aims to establish foundations for studying two- and three-dimensional cases of the level set method. Furthermore, the data to train the neural network will be obtained by solving the level set equation using finite differences methods such as upwinding, high-order ENO, and WENO. The proposed models will be evaluated using the exact solution of the level set equation as well as numerical solutions generated by finite differences methods and total variation diminishing (TVD) Runge-Kutta methods for space and time discretization, respectively. This will allow us to assess the effectiveness of the machine learning technique compared to traditional methods.

1.3 Objectives

1.3.1 General Objective

Infer solutions of the level set equation with free boundary conditions using a deep neural network that learns from the level set equation and data generated by finite differences schemes, including upwinding, ENO, and WENO methods.

1.3.2 Specific Objectives

1) Implement the finite differences, upwind, ENO, and WENO schemes coupled with the third-order TVD-Runge Kutta schemes to discretize space and time. Hence,

generating numerical solutions of LSE to feed the neural network.

2) Study the accuracy of inferred solutions by neural networks by comparing them with the numerical solutions from the finite differences methods.

3) Compare the results of proposed models with the exact and numerical solutions using $L1$ and $L2$ errors.

Chapter 2

Theoretical Framework

This chapter will explain in detail the theory needed to understand this project.

2.1 Hamilton Jacobi Equations

Hamilton-Jacobi equations are a family of hyperbolic partial differential equations (PDEs) in classical mechanisms to describe the evolution of systems under an action. The duality between trajectories and wavefronts is a crucial property of HJ in many areas of physics such as optics, quantum mechanism, and fluid dynamics; mainly, it helps to understand the behavior of waves and particles in physical systems [37]. In tracking problems, Osher and Sethian [1] studied the propagation of the front with curvature-dependent speed using HJ formulation to describe the motion of the front (Fig.2.1). The general HJ equation is given by [9]

$$\phi_t + H(\nabla\phi) = 0, \quad \phi(x, 0) = \phi_0(x), \tag{2.1}$$

where $x \in \mathcal{R}^n$, $t > 0$. The solutions of (2.1) are not continuously differentiable even when the initial function $\phi_0(x)$ is smooth; the analytic study of uniqueness, stability, and existence of the viscosity solution for (2.1) is proved under suitable assumptions on the Hamiltonian H [9].

The HJ approach has significant advantages in applications. One of the most important advantages is its capacity to tackle severe topological changes of the front, which makes it appropriate for propagating the interface. Another advantage in control problems is to allow generating of optimal trajectories that minimize a given cost function. Making it useful for applications such as path planning, where the objective is to find the optimal path between two points while avoiding obstacles and minimizing energy consumption [1, 38].

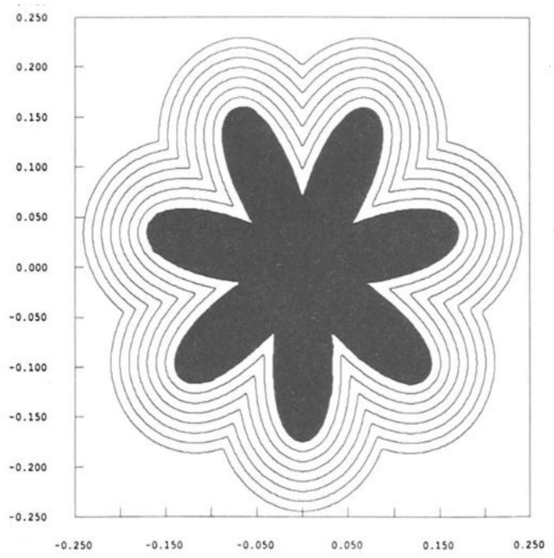


Figure 2.1: Motion under curvature of the front. Source [1]

2.2 Level Set Method

The level set method is a numerical technique to track and simulate the motion of interface or fronts between two regions of a physical system. This method was first devised by Osher and Sethian in 1988 [1] and has been used in diverse areas such as image processing, fluid dynamics, and computational geometry. One of the most essential ideas is to represent the interface as a level set of a higher-dimensional function. For example, if considering the zero level set of a three-dimensional function, the interface is represented in two dimensions (Fig.2.2). Consequently, this representation allows evolving the interface as a higher-dimensional function using the level set equation [18].

In order to evolve the interface, Osher and Fedkiw [18] considered a specific case of the Hamilton Jacobi Equation, taking

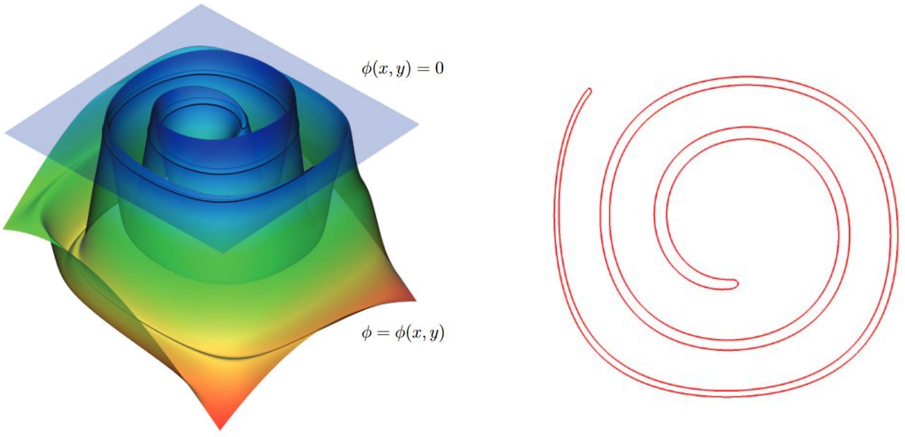
$$H(\nabla \phi) = V \cdot \nabla \phi, \tag{2.2}$$

Replacing (2.2) in (2.1), they obtained the simple advection equation (or the level set equation) given by

$$\phi_t + V \cdot \nabla\phi = 0, \quad \phi(x, 0) = \phi_0(x), \tag{2.3}$$

with $x \in \mathcal{R}^n$, $t > 0$. Moreover, V is the velocity field that is defined in all domain $\mathcal{R}^n$ and the initial interface $\phi_0(x) = 0$ is usually represented as the zero level set of the signed distance function. This is known as the Eulerian formulation for describing the interface evolution since capturing the interface is led to high-dimensional function.

The main advantage of the level set method is capturing complex topological changes in the interface, such as splitting, merging, and re-connection, without meshing methods. The method also allows for the accurate tracking of the interface even in the presence of strong gradients and shocks, and it can handle arbitrary geometries. In this sense, the equation (2.3) is solved using numerical techniques such as finite differences, finite elements, or spectral methods. Additionally, the method also requires various numerical techniques to handle the interface, such as reinitialization, advection, and curvature computation [18].



(a) (b)

Figure 2.2: a. Representation of interface by zero level of signed distance function, b. The interface of (a). Source [2, 3]

2.3 Signed Distance Function

Signed-distance functions (SDFs) are a subset of implicit functions that emerge from adding an extra property, $|\nabla\phi| = 1$, to the level set function by defining the distance function

(Fig.2.3). Mainly, a signed distance function gives the distance from an arbitrary point in the domain to the closest point on the interface [18]. This type of function is used in computer vision, mostly in real-time computer graphics, to describe the object geometry.

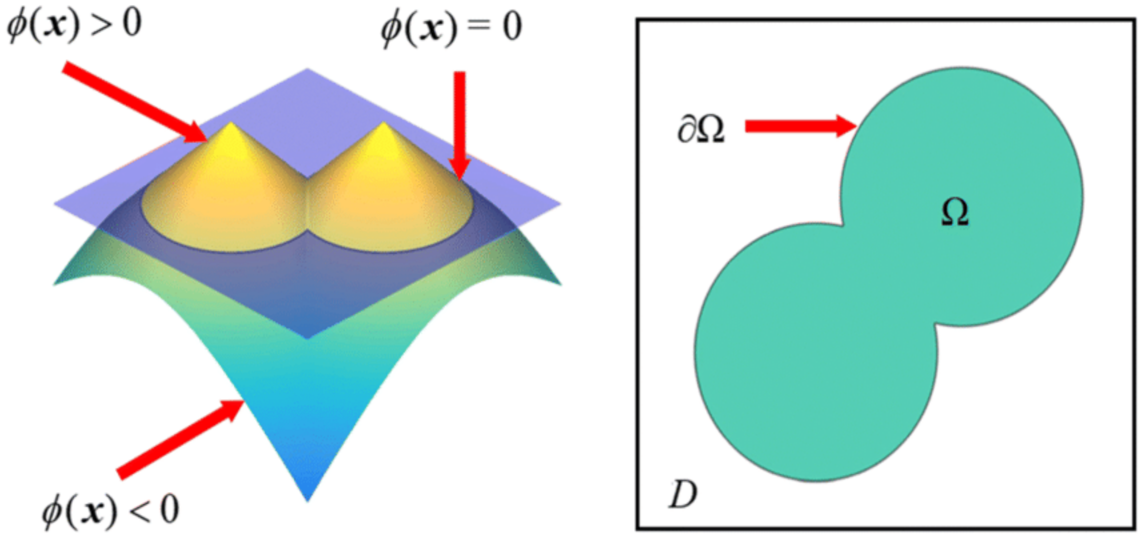


Figure 2.3: On the left side is a representation of a signed distance function, and on the right side is the interface of SDF. Source [4]

Let's consider a continuous scalar function ϕ with a free surface Γ defined by zero level set of ϕ, i.e,

$$\Gamma = \{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}.$$

Then, the level set function is defined as [39] (Fig. 2.4)

$$\phi(\mathbf{x}, t) = \begin{cases} > 0, & \text{in } \mathbf{x} \in \Omega^+ \\ = 0, & \text{in } \mathbf{x} \in \Gamma \\ < 0. & \text{in } \mathbf{x} \in \Omega^- \end{cases} \tag{2.4}$$

Now, define the distance function as the minimum distance from an arbitrary point $\mathbf{x} \in \Omega$ to the closest point $\mathbf{x}_\Gamma \in \Gamma$, i.e.,

$$d(\mathbf{x}) = \min(|\mathbf{x} - \mathbf{x}_\Gamma|). \tag{2.5}$$

Notice that if the point $\mathbf{x}$ is on the interface, $d(\mathbf{x}) = 0$. Therefore, using (2.5) can define a

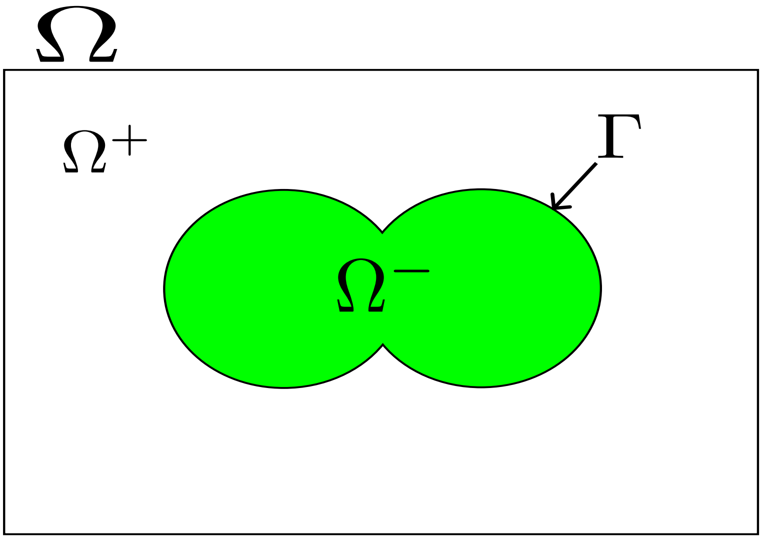


Figure 2.4: Representation of level set function

signed distance function of (2.6), as follows;

$$\phi(\mathbf{x},t) = \begin{cases} d(\mathbf{x}), & \text{in } \Omega^+ \\ -d(\mathbf{x}). & \text{in } \Omega^- \end{cases} \tag{2.6}$$

As a result of definition (2.6), signed distance functions have some interesting properties like [18];

(1) $|\nabla\phi| = 1$

(2) Let ϕ_1 and ϕ_2 be SDFs, $\min(\phi_1, \phi_2)$ is the union of two interior regions.

(3) Let ϕ_1 and ϕ_2 be SDFs, $\max(\phi_1, \phi_2)$ is the intersection of two interior regions.

(4) Let ϕ be SDF, define $-\phi$ as complement of ϕ

Example 1 *Let $\phi(x,y) = x^2 + y^2 - 1$ be an implicit function of a circle centered at the origin with unit radius, then the signed distance function for ϕ is defined as;*

$$\phi(x,y) = \sqrt{x^2+y^2} - 1.$$

2.4 Finite Differences Methods

Finite difference methods are numerical techniques that help to solve partial differential equations. The idea of FDM is to compute the finite differences between values of a function at discrete points for approximating its derivatives at a point [5]. The discrete points $x_i, i = 0, ..., M$ represent a partition of a interval $[a, b]$ with $M + 1$ points (Fig. 2.5), and a finite difference between points is defined by (2.7)

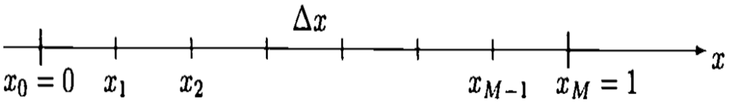


Figure 2.5: Uniform partition of interval $[0, 1]$. Source [5]

$$\Delta^+ f_i = \frac{f_{i+1} - f_i}{\Delta x}, \tag{2.7}$$

with i represents point in partition, f_i represents the value of function at point i and Δx is the spacing between two discrete points. The formula (2.7) is a discrete expression for computing the following derivative;

$$\frac{d}{dx} f \approx \frac{f(x + h) - f(x)}{h}.$$

This is a brief description to understand finite differences methods. In literature, the first-order numerical methods for solving hyperbolic PDEs [40]:

(a) Upwind scheme;

(b) Lax-Friedrichs method;

(c) Lax-Wendroff method;

and high-order methods;

(a) Second-order Lax-Wendroff

(b) Third-order essentially non-oscillatory (ENO)

(c) Fifth-order weighted essentially non-oscillatory (WENO)

In the following sections, upwind, ENO, WENO, and TVD-Runge Kutta schemes are described in a general way in order to solve the level set method.

2.4.1 Upwind Method

Let's define the motion of the interface by PDE [18];

$$\phi_t + \overrightarrow{V} \cdot \nabla\phi = 0. \tag{2.8}$$

In order to find the solution at time t^{n+1}, a first-order accurate method is used to discretize the time in equation (2.8), which is the forward Euler method given by

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \overrightarrow{V}^n \cdot \nabla\phi^n = 0, \tag{2.9}$$

where $\overrightarrow{V}^n$ is the externally generated velocity field at time t^n, and $\nabla\phi^n$ is the gradient operator applies to ϕ at time t^n. Then equation (2.9) can be written as [18];

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + u^n\phi^n_x + v^n\phi^n_y + w^n\phi^n_z = 0.$$

In order to approximate ϕ^n_x at the point x_i. The method of characteristics is described as

(a) If $u_i > 0$, then ϕ_x is approximated by

$$\frac{\partial}{\partial x}\phi \approx \frac{\phi_i - \phi_{i-1}}{\Delta x}.$$

This is called a first-order accurate backward difference.

(b) If $u_i < 0$, then ϕ_x is approximated by

$$\frac{\partial}{\partial x}\phi \approx \frac{\phi_{i+1} - \phi_i}{\Delta x}.$$

This is called a first-order accurate forward difference.

Similarly, the terms ϕ_y and ϕ_z are independently approximated with forward and backward differences. The above description is the idea of the upwind method.

2.4.2 Essentially Non-oscillatory

The Essentially Non-oscillatory method (ENO) was first introduced by Harten et al. [41] as an improvement for forward and backward differences. This high-order numerical approach is used to solve hyperbolic partial differential equations in various applications such as image processing, computational fluid dynamics, etc.

The ENO idea is to define a partition of an interval $[a, b]$ and consider a subset around a point in the partition, mainly; for the third-order accurate ENO, to approximate a derivative at point x_i by left, the left-based stencil is given by

$$S = \{x_{i-3}, x_{i-2}, x_{i-1}, x_i, x_{i+1}, x_{i+2}\},$$

using this stencil, Harten et al. [1] defined sub-stencils as

$$\begin{aligned} S_0 &= \{x_{i-3}, x_{i-2}, x_{i-1}, x_i\}, \\ S_1 &= \{x_{i-2}, x_{i-1}, x_i, x_{i+1}\}, \\ S_2 &= \{x_{i-1}, x_i, x_{i+1}, x_{i+2}\}. \end{aligned}$$

Then the strategy is to construct an interpolation polynomial using each sub-stencil; papers [1, 18, 42] built this procedure based on Newton divided differences. Now, the aim is to choose the suitable interpolation polynomial based on the local smoothness of the function to be an approximation of derivative [9]. For example;

(a) If the discontinuity is in the point x_{i-2}, then the interpolation polynomial built by S_2 is suitable.

This shows the capacity of the ENO method to precisely represent sharp changes and abrupt shifts in the solution while minimizing any false oscillations [42].

2.4.3 Weighted Essentially Non-Oscillatory

The Weighted Essentially Non-Oscillatory (WENO) was first introduced by Liu, Osher, and Chan [43] as an improvement for the ENO method. This high-order method is the most used in various applications such as computational fluid dynamics, astrophysics, and computational electromagnetic [42].

The WENO idea developed by [43] involved combining three ENO approximations in a convex manner. In cases where any of the approximations go across a discontinuity, it is assigned a small weight to minimize its impact and errors. All three approximations are given a substantial role in smooth flow areas, thereby enhancing local accuracy from third order to fourth order. Later, Jiang and Shu [21] developed optimal weights for the convex combination of ENO approximations which caused a fifth-order accuracy in smooth regions. Below is a description of WENO's idea by [18];

Let ϕ_x^0, ϕ_x^1 and ϕ_x^2 be ENO approximations, the convex combination to approximate a derivative is given by

$$\phi_x = \omega_0\phi_x^0 + \omega_1\phi_x^1 + \omega_2\phi_x^2,$$

where the weights must satisfy the following

$$\omega_0 + \omega_1 + \omega_2 = 1.$$

Jiang and Shu [21], proposed the following weights; $\omega_0 = 0.1, \omega_1 = 0.6$ and $\omega_2 = 0.3$ to achieve the fifth-order accuracy in smooth regions. While in regions with discontinuities, the weights are chosen to be $\omega_k = 0$ or $\omega_k = 1$, this procedure gives a single approximation to ϕ_x,i.e., one of the ENO approximations [18].

2.4.4 Total Variation Diminishing Runge Kutta

Gottlieb and Shu [44] described TVD/RK as a method to solve a system of ODEs;

$$\phi_t = L(u).$$

Coupled with suitable initial conditions emerges from a method of lines approximation to hyperbolic conservation law:

$$\phi_t = -f(\phi)_x,$$

where the spatial derivative $f(\phi)_x$ is approximated by finite differences schemes and is denoted by -$L(\phi)$. Then, the firs-order TVD/RK is the Euler forward stepping;

$$\phi^{n+1} = \phi^n + \Delta t L(\phi^n),$$

under suitable restriction;

$$\Delta t \leq \Delta t_1.$$

Then the aim of high-order TVD Runge Kutta time discretization is to keep the TVD property:

$$TV(\phi^{n+1}) \leq TV(\phi^n),$$

where TV is the total variation of the numerical solution defined as;

$$TV(\phi) = \sum_j |\phi_{j+1} - \phi_j|.$$

In order to achieve higher order accuracy in time, the restriction with a different time step is given by

$$\Delta t \leq c\Delta t_1, \tag{2.10}$$

where c is the CFL coefficient for the high-order time discretization. Then the following Lemma describes the necessary condition to be TVD [44].

Lemma 1 *The general Runge-Kutta method describe in [44] is TVD under the CFL coefficient (2.10):*

$$c = \min_{i,k} \frac{\alpha_{ik}}{\beta_{ik}}, \tag{2.11}$$

provided that $\alpha_{ik} \geq 0$, $\beta_{ik} \geq 0$.

Under this condition, Gottielb and Shu [44] described the third-order TVD Runge Kutta;

Proposition 1 *If we require* $\alpha_{ik} \geq 0$, *and* $\beta_{ik} \geq 0$, *then the optimal third-order TVD Runge Kutta method [44] is given by*

$$\begin{aligned}
\phi^{(1)} &= \phi^n + \Delta t L(\phi^n), \\
\phi^{(2)} &= \frac{3}{4}\phi^n + \frac{1}{4}\phi^{(1)} + \frac{1}{4}\Delta t L(\phi^{(1)}), \\
\phi^{n+1} &= \frac{1}{3}\phi^n + \frac{2}{3}\phi^{(2)} + \frac{2}{3}\Delta t L(\phi^{(2)}),
\end{aligned}$$

with a CFL coefficient $c = 1$.

2.5 Neural Networks

Neural Network is a model that simulates the function of the human brain, which contains layers where each one has interconnected neurons to each other. Each neuron contains parameters called weights and biases that help process the input information. Moreover, the out-of-neuron is controlled by nonlinear functions called activation functions [45]. For example, Figure 2.6 represents that the neuronal network contains two layers with two neurons, each one. Kunihiko Fukushima and Sei Miyake [46] developed an important neuronal network called Neocognitron. It was used for visual pattern recognition, mainly classifying and recognizing patterns on objects according to their shapes. The successor of this neural network is the famous Convolutional Neural Network (CNN) used today in the segmentation and recognition of images. In this context, a complex neural network with more than two layers is called a deep neural network.

The traditional architectures in deep learning are fully connected feed-forward networks (FFN), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). However, a recent work [47] described that there are new deep neural networks based on unsupervised learning and Bayesian probabilistic. In the same way, the activation function plays a crucial role in DNN training performance, and the most used are ReLU, Sigmoid, and Tanh [48]. For example, the suitable activation functions in physics-informed neuronal networks (PINNs) are smooth activation functions such as the sigmoid and hyperbolic tangent [27]. Therefore, all these concepts can be represented by a mathematical framework.

Caterini and Chang [45] proposed a mathematical model of neural network with n layers as the composition of n functions; i.e,

$$\phi_\theta(x) = f_n(f_{n-1}(...(f_1(x)))).$$

Each function $f_i(x_i, \theta_i)$ is defined as

$$f_i : E_i \times H_i \to E_{i+1},$$

where E_i, H_i and E_{i+1} are inner products spaces for all $i = 1, ..., n$. Moreover, $x_i \in E_i$ are

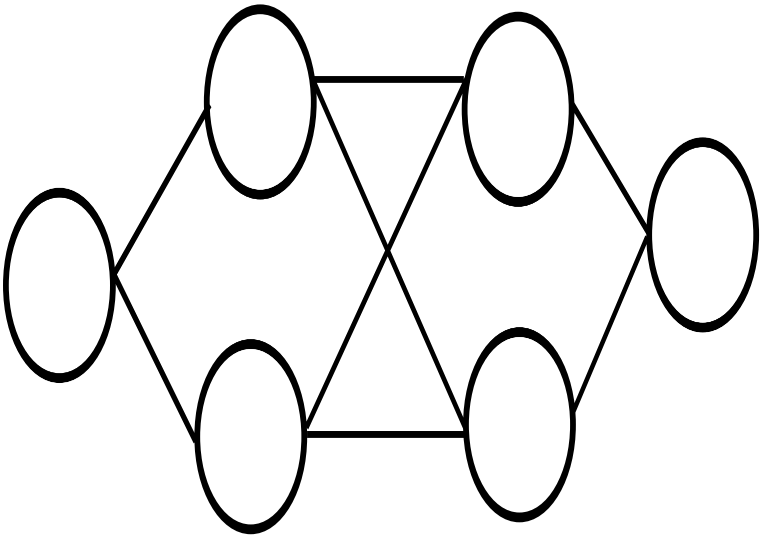


Figure 2.6: Representation of a simple neuronal network

state variables, and $\theta_i \in H_i$ is the set of parameters for i-th layer.

2.5.1 Physics Informed Neural Networks

In recent years, physics-informed neural networks (PINNs) are feed-forward neural networks that have experienced significant growth in applications involving partial differential equations such as fluid dynamics, material science, solid mechanism, etc. PINNs infer PDE solutions based on minimizing a loss function when the neural network is trained. The training can address problems that contain little data or noisy data; in this context, the main advantage of PINNs is that they are neural networks that handle supervised learning problems since they can utilize available data while following specific physical laws imposed by non-linear or linear partial differential equations [27, 49].

Cuomo et al. 2022 [27] described the most general form of differential equations that PINNs can solve;

$$\begin{aligned} F(f(\mathbf{x});\gamma) =& u(\mathbf{x}), \quad \mathbf{x} \in \Omega \\ B(f(\mathbf{x})) =& g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega \end{aligned}$$

defined on the domain $\Omega \subset \mathcal{R}^n$. Where the vector $\mathbf{x}$ represents the variables in space and time, f is the unknown solution, γ are the parameters related to physics such as pressure (ρ), viscosity constant (ν), velocity field and so on. Moreover, F represents a linear or non-linear operator depending on the problem involving partial derivatives, and B are the boundary conditions defined as Dirichlet, Neumann, or periodic boundary conditions [27, 49].

In order to solve the above problem, the general idea of PINNs is to computationally approximate a function $f(\mathbf{x})$ by a NN, which is parameterized by a set of parameters θ; then the approximation is defined by

$$\hat{f}_\theta(\mathbf{x}) \approx f(\mathbf{x}),$$

where $\hat{f}_\theta$ is a NN approximation of f under parameters θ. These parameters are known as weights ($\mathbf{W}$) and biases ($\mathbf{b}$). In the context of FF-NN, the NN approximation is defined as a matrix form;

$$\hat{f}(\mathbf{x};\theta) = \hat{f}(\mathbf{x};\mathbf{W},\mathbf{b}) = \alpha \cdot (\mathbf{W} \cdot \mathbf{x} + \mathbf{b}),$$

where α is the activation function which can be ReLU, sigmoid, etc.

On the other hand, in order to find the suitable parameters θ of the $\hat{f}_\theta$, θ is obtained by the process of minimization of loss function $L(\theta)$ [27];

$$\theta = \arg\min_\theta L(\theta),$$

where the loss function is defined as

$$L(\theta) = \omega_F L_F(\theta) + \omega_B L_B(\theta) + \omega_d L_{data}(\theta).$$

In papers developed by He et al., [50] and Stiller et al. [51], described that L_F is the loss produced by a mismatch with the governing differential equations F, i,e, this terms imposed the physical constraints by automatic differentiating applied on differential operator F using chain rule; mainly, this helps to compute the derivative of NN ($\hat{f}_\theta$)[52]. Kollmannsberger at al. [53] defined the L_θ based on mean square error as;

$$L_F(\theta) = MSE_F = \frac{1}{N_c}\sum_{i=1}^{N_c} \left\| F(\hat{f}_\theta(x_i)) - u(x_i) \right\|^2,$$

where N_c are randomly selected collocation points inside the domain. In the same way, he defined the boundary and initial conditions as;

$$L_B(\theta) = MSE_B = \frac{1}{N_b}\sum_{i=1}^{N_b}\left\|B(\hat{f}_\theta(\mathbf{x}_i)) - g(x_i)\right\|^2,$$

$$L_{data}(\theta) = MSE_{data} = \frac{1}{N_d}\sum_{i=1}^{N_d}\left\|\hat{f}_\theta(\mathbf{x}_i) - f_i\right\|^2,$$

where f_i is a known data point, this helps to compute the error of the approximation f. Notice that N_c, N_b, N_d are chosen randomly. Moreover, L_F penalizes the discrepancy between the predicted left-hand side of a partial differential equation (PDE) and the actual right-hand side of the PDE [53].

Various optimization methods are used to reduce the computational cost of minimizing the loss function in physics-informed neural networks (PINNs). The most commonly used optimization methods in PINNs are minibatch sampling with Adam and the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm, which is a type of quasi-Newton optimization algorithm. Mathews et al. [52] pointed out that this method is optimum for increasing the sample size of data training. However, there is another method called stochastic gradient descent (SGD) to increase converge speed [27].

Raissi et al. [25] studied particular PDEs that have the form;

$$u_t + \mathcal{N}[u;\lambda] = 0, \quad x \in \Omega, \quad t \in [0,T], \tag{2.12}$$

where the authors defined the loss function $L(\theta)$ for training as;

$$L(\theta) = \frac{1}{N_c}\sum_{i=1}^{N_c}\left\|\frac{\partial}{\partial t}\hat{u}_\theta(x,t) + F_x[\hat{u}_\theta(x,t)] - r_i\right\|^2 + \frac{1}{N_d}\sum_{i=1}^{N_d}\left\|\hat{u}_\theta(x_i,t_i) - u_i\right\|^2.$$

Example 2 *For example, in one-dimensional Burger's equation with Dirichlet boundary conditions given by [25]:*

$$u_t + uu_x - (0.01/\pi)u_{xx} =, \quad x \in [-1,1], \quad t \in [0,1]$$

and

$$u(0,x) = -\sin(\pi x), \quad u(t,-1) = u(t,1) = 0.$$

For this case, the physics-informed neural network is defined as;

$$f(t, x) = u_t + uu_x - (0.01/\pi)u_{xx}.$$

Moreover, the implementation of $f(t, x)$ in Example 2 using programming language Python and library TensorFlow;

```
def u(t, x):
    u = neural_net(tf.concat[t, x], 1), weights,  biases
    return you

def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x= tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

Convergence Aspects

De Ryck et al. [54] studied the convergence of PINNs based on when a sequence of predictive solutions $\hat{f}_\theta$ converges to the solution of the physical problem, i,e.

$$(\hat{f}_\theta)_n \rightarrow, \quad n \rightarrow \infty.$$

The researchers demonstrated that as the width of a pre-defined neural network (NN) with the activation function tanh increases to infinity, the difference between the estimated function $\hat{f}_\theta$ and the actual function f will approach zero. Therefore, the width corresponds to the number of parameters or weights the neural network learns during the training phase. It should be selected based on the problem's complexity, the dataset's size, and the computational resources available to guarantee convergence [54].

Chapter 3

State of the Art

3.1 Studies on High-order Finite Differences Methods

The level set method is a field where various authors have developed studies to improve the numerical methods advised in [1] to solve tracking problems using Hamilton- Jacobi equations involving formulation of fronts propagating with local curvature speed. Osher and Sethian [1] introduced the concept Level Set Method by defining the acts using a level set of high-order dimensional functions; it helped to track the propagation of fronts easily; because, from this point of view, the severe topological changes subject to the shows as they evolve were successfully tracked.

For hyperbolic partial differential equations, Harten, Osher, et al. [41] developed numerical approximations to weak solutions of the hyperbolic initial value problem (2.1) to approximate the derivatives concerning spatial variables based on high-order differences method; Essentially non-oscillatory schemes use an adaptive stencil to avoid interpolations through discontinuities, i.e., piece-wise polynomial reconstruction of the solution from its cell averages. Moreover, the discretization of time developed in [41] was the total variation diminishing schemes that have at most first-order accuracy in truncation error. Still, it can cause smoothing of certain areas using this technique could result in pollution and potentially create nonlinear instability, ultimately causing the schemes to fail [24]. This approach prevents spurious oscillations near discontinuities called the Gibbs phenomenon, in contrast to traditional high-order accurate finite difference, finite volume, finite element, or spectral schemes [42]. Therefore, the studies [41, 1, 24] have been improving to solve

Hyperbolic PDEs, which positively affects the level set method to obtain suitable capturing interfaces.

The challenge remains which numerical method gives the best approximation to solutions of hyperbolic PDEs. In the context of LSM, this is important because allowing to capture the interface preserving the geometry; Pineda et al. [7] developed various testing examples to analyze the accuracy of upwind and ENO schemes when the interface is passively advected in a flow field. The results showed that the ENO schemes keep the geometry interface, in contrast to upwind produces deformation of the interface; the grid size is significant to obtain good results; consequently, the computational cost grows. However, the ENO approach is third-order accurate, then Liu, Osher, and Chan [43] developed a new type of ENO scheme called weighted ENO (WENO) based on a convex combination of interpolating polynomials of ENO for smooths regions and keeping the third order of ENO on discontinuities regions. It produces an arbitrarily high order of improvement in accuracy; mainly, the fifth order WENO (WENO5), is more used in flow simulation applications. Luo et al. [8] developed a comparison between WENO5 and Finite Volume WENO5 when the simulation involved smooth vortex propagation to the intense shock interaction; in context, Navier-Stokes equations, the results showed similar solutions; however, the number of points on mesh played a significant role because the reduction of mesh points involving problems with accurate of WENO5. In the same way, Gu et al. [55] developed a comparison between the traditional WENO5 and a new version called dispersion-relation-preserving which is compact-reconstruction weighted essentially non-oscillatory (DRP-CRWENO4) to preserve the interface that captures level set method in simulation of dam-break flows; the results showed greater precision with DRP-CRWENO4. Therefore, ENO and WENO schemes are suitable to solve problems for capturing and preserving the interface; however, the two and three dimensions of the level set method involve extra computational cost [55, 34].

3.2 Extensions on Level Set Method

The efficiency of the level set method is based on numerical methods to preserve the interface with low computational cost. ENO and WENO schemes effectively solve the problem involving the external velocity field; however, other methods or versions show the best properties to preserve the interface. Enright et al. [56] developed a numerical method based on Lagrangian market particles to conserve the properties of the level set method; it allows the reconstruction of the interface in regions that are underresolved. This procedure improves the interface-preserving in two and three dimensions, unlike WENO5; however, constructing particles involves an extra computational cost. On the other hand, Sussman et al. [57] developed a reinitialization level set method that maintains the signed distance property; in other words, the level set function is typically initialized into a signed distance function, which the zero level set of this function represents the interface; it satisfies as a unique viscosity solution of the Eikonal equation; however, after being set up as a signed distance function, the level set function typically loses this property as it evolves according to Eikonal equation, and therefore requires regular reinitialization. This procedure allows to improve the conservation of properties of LSM, and these previous methods still entail the WENO5 schemes to solve the evolution of interfaces [58]. From another point of view, Vorgelegt von [22], in his doctoral work, developed the theory of finite elements method to solve the level set equation; in this context, Dag lindbo [59] designed numerical experiments based on the variational formulation for level set equation, this method is more suitable than finite differences in terms of stability and converge; moreover, Long and Hyoung [60] mentioned that the computational cost is low with Finite element compared with reinitialization LSM even if refinement meshes are used. Therefore, various studies showed improvements in solving the initial value LSM, aiming to preserve the interface and have low-cost computational.

3.3 Physics Informed Neuronal Networks on Fluid Dynamics

In recent years, neural networks have played an essential role in inferring solutions of ordinary or partial differential equations. Kossacka et al. [28] used deep learning techniques to improve the fifth-order WENO scheme; they trained a neural network to modify the smoothness indicators and achieve suitable numerical results at discontinuities; however, there are other methods based on training a deep neural network using a few data and the information of PDEs to inference solutions; Raissi et al. [25] developed a new idea based on neural networks can be trained to perform supervised learning tasks while respecting laws of physics that are described by nonlinear partial differential equations. These are called Physics informed neural networks (PINNs). The authors showed the proposed framework's efficacy by solving various traditional problems in fluid dynamics, quantum mechanics, reaction-diffusion systems, and the transmission of nonlinear shallow-water waves. Erik Laurin Strelow et al. [29] used PINNs to study gas transport problems to avoid redundant computations, is fast for similar simulations, and can maintain high accuracy. According to their experiments, achieving exact approximations requires solving the optimization problem during the training phase with exceptional accuracy. However, conventional machine-learning tasks typically avoid solving the training problem with excessive accuracy to prevent overfitting.

On the other hand, Qiu [30] led a complex flow modeling based on PINNs to tackle the challenge by solving high-order derivate terms and capturing the interface adaptively; they developed a different test to show the interface-capturing ability of PINNs; the outcomes indicated that PINNs utilize automatic differentiation while retaining the exceptional accuracy of the phase-field method, it causes that under the influence of the mobility, the capturing accuracy is significant. In addition, Jiaewi Li, Wei Wu, et al. [61] improved PINNs to solve two-dimensional Stefan problems; through examples of unstable regions and free boundaries, this paper illustrated that the proposed method yielded accurate and effective predictions. A leading property of PINNs is the mesh-free approach for solving partial differential equations (PDEs), in contrast to classical numerical techniques. Shibo Li, Michael Penwarden, et al. [62] mentioned this property could mitigate the complexity of the interface, reduce the computational cost and allow parallelization; in addition, Aaron B. Buhendwa, Stefan Adami, et al. [32] applied physics-informed neural networks to tackle problems involving incompressible two-phase flows. Specifically, they focused on the forward problem [63], which entails solving governing equations based on given initial and boundary conditions; here, the level set method is used to capture the interface generated by a numerical simulation for training the neuronal network. However, in the literature, studies about PINNs focus on track problems, such as the level set method.

3.4 Preliminaries Studies on Tracking Problems

To study the level set method, there are works focused on finding solutions to advection equations. In section 2, the level set equation is called the advection equation when the velocity field is constant. From this point of view, in the paper of Shashank Reddy Vadyala and Sai Nethra Betgeri [64], various finite-difference approximations and physics-informed neural networks (PINNs) are utilized to compute numerical solutions of the advection equation under circumstances that permit an analytical solution. In the same way, Qi Zhi He and Alexandre M Tartako; theyciteHe2021 suggested a technique for solving the coupled advection-dispersion equation (ADE) and Darcy flow equation with hydraulic conductivity that changes in space involves utilizing the physics-informed neural network (PINN) approach, which does not require discretization. Vincent Liu and Hongkyu Yoon [65] employed physics-informed neural networks to forecast fluid flow in a limited space and compared their outcomes with analytical and numerical solutions acquired from fluid dynamics simulations. They evaluated their models by analyzing diverse flow and transport scenarios in 2D domains utilizing the Navier-Stokes and advection-diffusion differential equations. For example, Chulin Wang, Eloisa Bentivegna et al. [66] studied a model of atmospheric pollution plumes that incorporates advection-diffusion; they investigated a super-resolution (SR) technique for reconstructing high-resolution images (4x) from low-resolution ones based on PINNs; the results compared to conventional super-resolution techniques, physics-informed neural networks (NNs) are more successful at rebuilding degraded images and producing superior outcomes. All the studies developed so far do not show foundations to solve the level-set method with PINNs.

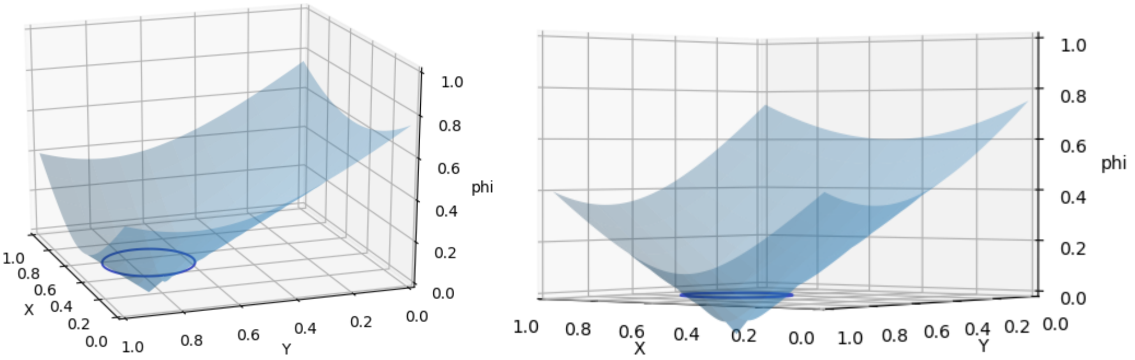
Chapter 4

Methodology

4.1 Description Problem

In 1988, Osher and Sethian [1] introduced new propagation of surfaces under curvature algorithms (PSC) to track interfaces propagating, such as flame propagating and crystal growth, in which the speed depends on the local curvature of fronts. They defined the motion of the interface by the initial-value Hamilton-Jacobi equation and the surface as a level set of a signed distance function. This formulation allowed to track of severe topological changes in the front, in contrast to a technique based on parametrizing the moving interface and a set of marker points to discretize the parametrization produced an accurate tracking of the front when the moving involves small perturbations. Still, with the large complex motion the result of precision decreases. Therefore, the level set method defines the tracking of the interface based on a level set of signed distance functions.

In this context, a level set method is a type of Hamilton Jacobi formulation for tracking an interface. The motion is defined by a level set equation and the speed depends on an externally generated velocity field (Fig.4.1). Thus, the numerical schemes must be chosen appropriately to avoid spurious oscillations in numerical solutions leading to poor accuracy and the non-convergence to solution. Following this, the main papers focused on traditional numerical methods to approximate the partial derivatives with respect to space variables; [9, 67, 21, 24] in which the authors developed schemes based on finite differences for hyperbolic systems of conservation laws. Osher and Salomon [67] proposed upwind differences schemes to discretize space on hyperbolic equations, and they pointed out that these schemes accomplish the properties of conservation form; no spurious oscillations, low computational cost, but these methods are limited to first-order; it causes problems when the front contains many discontinuities. In order to solve it, Osher and Sethian [1] proposed a non-oscillatory numerical method based on polynomial interpolation; this feature allowed the construction of an arbitrary high-order method called ENO. This type of approach accomplishes conservation laws, and above all, if the initial function is discontinued, it produces an accurate solution. In this same sense, Jiang and Shu [21] improved WENO schemes developed in [43]; this method is formulated to work with fifth-order in the smooth region and third-order accuracy of ENO schemes at discontinuities. This kind of approach has significant features such as the accuracy and the convergence of solutions, and computing time is faster than the ENO approach and so on. Therefore, each scheme has advantages and disadvantages that can result in better or worse depending on the analyzed situation.



(a) (b)

Figure 4.1: Tracking the zero level of explicit function $\phi(x, y) = x^2 + y^2 - 1$ using Level Set Method

Another important part of solving the level set method is to identify suitable numerical methods to approximate the derivative of a function with respect to time; it is crucial to obtain linear stability coupled with upwind, ENO, and WENO schemes. Wang and Spiteri [68] argued that the forward Euler method joined with WENO leads to linear unstable; consequently, it is not convergent. For this reason, Shu and Osher [1] proposed a total variation diminishing (TVD) Runge-Kutta(RK); in this fact, the three-order TVD-Runge Kutta coupled with WENO is linearly stable. Additionally, this technique enhances the precision of the method of lines; which is used to find the solution of the level set equation at $t + \Delta t$; this is particularly important when solving the level set method that depends on the external velocity field. This method assumes that the temporal discretization can be separated from the spatial discretization in a semi-discrete manner, allowing the temporal discretization of the PDE to be treated independently as an ODE. Therefore, the accuracy of solutions depends on suitable numerical methods.

Due to the nature of numerical methods, the research in this field is based on studies for obtaining an efficient numerical technique to apply a specific real-world problem, such as simulations of fluids, image processing, and so on. TVD-RK-ENO and TVD-RK-WENO developed in [1, 43] showed appropriate properties to achieve the conservation laws of hyperbolic equations; however, later studies [69, 21, 3] proposed significant changes for ENO and WENO schemes to improve the computational cost. For example, Henrick et al. [70] developed a new idea for maintaining the fifth-order of WENO near critical points. Still, this extra process involves extra computational cost, unlike the method, improved in [21]. In fact, some proposed schemes of WENO or ENO that exist in literature [70, 61] involve a high precision and high cost computational using uniform meshes; consequently, two and three dimensions of level set equation is increased by the number of points in grids. Nevertheless, there are techniques based on adaptive meshes [69] and triangular meshes [71] to decrease computational cost; moreover, using another numerical approach to solve this issue, Teng et al. [72] compared finite volume WENO schemes with uniform WENO schemes obtaining low cost for finite volume schemes. Therefore, the challenge remains to find an efficient method to address the tracking problem using the level set equation.

On the other hand, the sensitivity of parameters of these numerical methods is significant for study. Mainly, the smoothness indicator ϵ proposed by Lian, Osher, and Chan [43] produces slight oscillations near shock waves; it propagates to smooth regions, causing the increment of truncation error. However, Zhang and Shu [6] modified the smooth indicator near shock regions to improve the fifth-order WENO schemes. In fact, this variation helps to reduce the slight oscillation near discontinuities, also the convergence to the optimal solution (Fig.4.2). Therefore, this parameter is crucial to improve the geometry of the interface.

All previous studies in aspects such as linear stability and convergence, types of meshes, the sensitivity of parameters, and computational cost are crucial to obtain meaningful

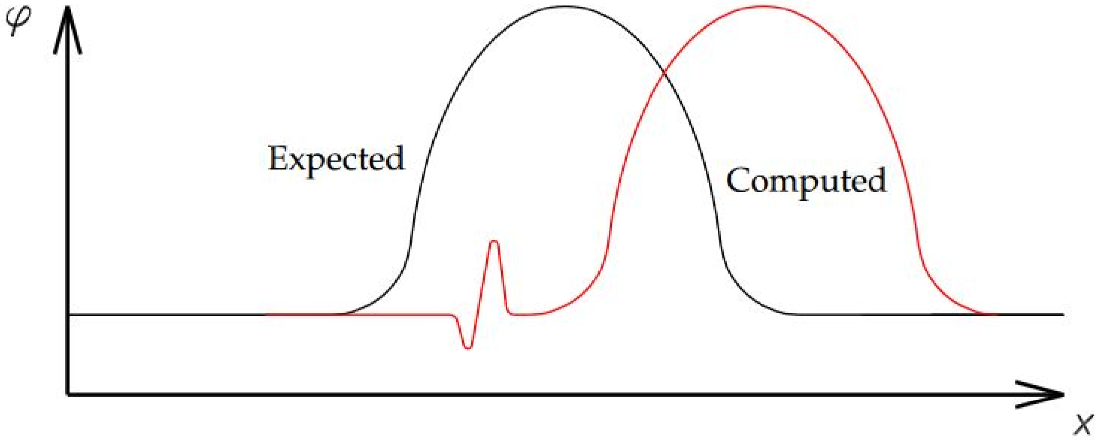
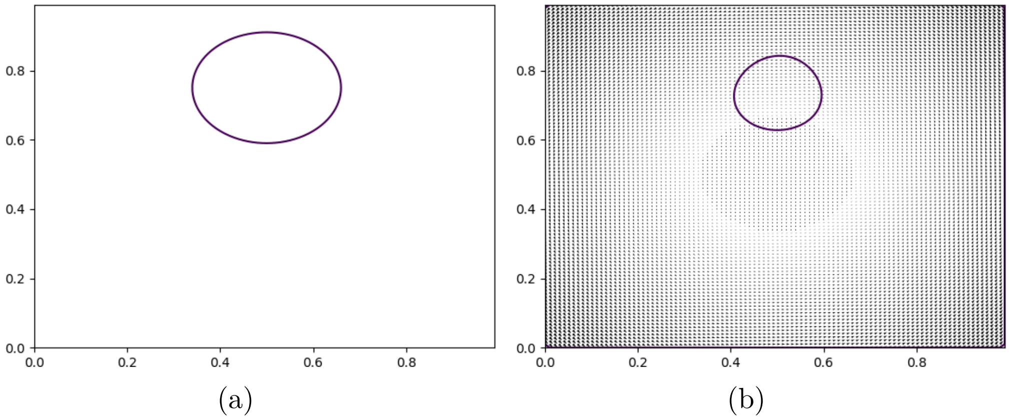


Figure 4.2: Spurious oscillation on the numerical solution. Source [6]

numerical simulations for free surface flow using Level Set Method. Studies [3, 73, 2, 7, 74, 75] developed a numerical simulation using the two and three dimensions of the level Set Equation with external vector field. L_1 and L_∞ are used to measure the error produced by a numerical method in simulations. In fact, Pineda et al. [7] developed various examples of interfaces to show the efficiency of upwind and ENO schemes proposed in [18, 9]; Figure 4.3 represents the one evolution of circle through circular vector field using a mesh resolution of 50^2, the results showed a deformation of shape. Furthermore, a complex numerical test, Henri et. at. [3], developed an interface shape for the single vortex test case using a resolution mesh of 1024^2 to compare the original fifth-order WENO [43] with High-Order Upstream Central (HOUC5) scheme; figure (4.5) shows high preservation of the interface and they emphasized that the combination of WENO5 and HOUC5 reduce a 50% of the computational cost of WENO5. Therefore, analyzing spatial and temporal numerical methods is critical to lead a real simulation.

Although significant advancements have been made in simulating physics phenomena through the discretization of PDEs, the process of mesh generation remains complex, and solving some high-dimensional problems cannot be tackled because of stability, computational cost, and so on. Deep neural networks have been used recently to tackle classical mathematical problems involving partial differential equations (PDEs) based on techniques of machine learning and artificial intelligence [27]. Raissi et al. [25] proposed a new numerical method called Physics Informed Neural Networks (PINNs); it is an architecture of



(a) (b)

Figure 4.3: a. Initial interface. b. One revolution of the circle using the forward Euler method and upwind differences. Source [7]

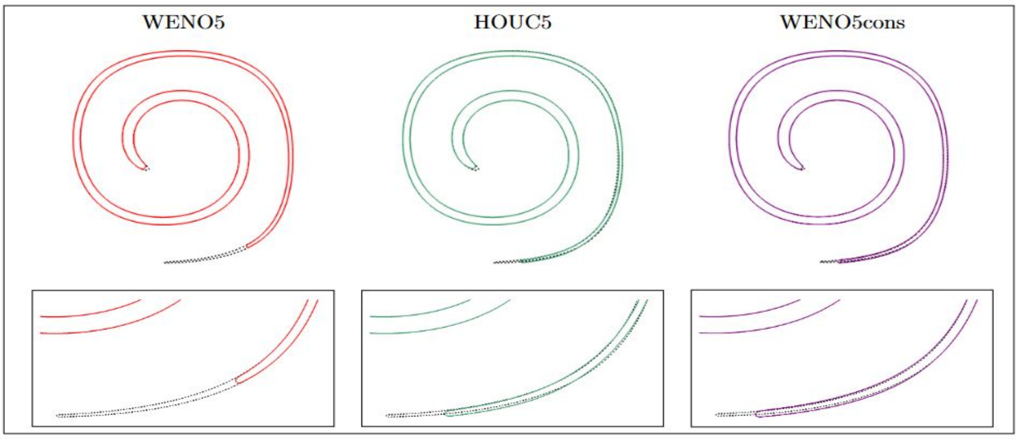


Figure 4.4: Single vortex test to show the efficiency of WENO5. Souce [8]

DNN to infer a solution of PDE by minimizing a loss function that depends on physical information of PDE, initial and boundary conditions. In this sense, defining a suitable loss function for a specific PDE is crucial; it allows it to converge to a desirable solution [27]. Therefore, this project developed two models of DNN to infer solutions in a one-dimensional level Set equation based on an architecture developed by Raissi in [25], discrete and continuous models are considered. The discrete model defines the physics-informed neural network based on the second term of the equation as a linear differential operator, i.e., it only depends on spatial derivative and velocity field. In order to compute the predictive solution, it uses n steps of Runge Kutta methods, and the loss function for training is defined only mean square error (MSE) of data ($L_{data}(\theta)$) and MSE of the differential operator. On the other hand, the continuous model is today the most used where

PINNs are defined by the left side of the equation. It contributes to the physical constraint of PDE and the loss function; for this case, it is defined by MSE of PDEs ($L_F(\theta)$) and MSE of data ($L_{data}(\theta)$). For either of the two cases, the mean square error of the boundary condition is not considered. This allows one to evaluate the proposed models based on their effectiveness if one considers a problem with free boundary conditions versus numerical methods. The training data are numerical solutions generated by upwind, ENO3, and WENO5. However, the problem has an analytic solution described in [76], allowing us to compare the analytic solution, traditional numerical methods (upwind, ENO, WENO schemes), and the proposed model. Finally, this is the first study to help build a significant foundation for tracking of the interface in two and three dimensions; mainly to investigate which parameters can be considered to infer solutions of LSE both in the training and in the formulation of the loss function.

Let's consider the following problem (4.1); one-dimensional level set equation (advection equation) with the initial condition given by signed distance function and constant velocity field $\mathbf{u}(t,x) = 0.01$,

$$\begin{cases} \frac{\partial}{\partial t}\phi(x,t) + \mathbf{u}(x,t) \cdot \frac{\partial}{\partial x}\phi(x,t) = 0, & \text{in } \mathbb{R} \times [0,T] \\ \phi(x,0) = \phi_0(x), & \text{in } \mathbb{R} \end{cases} \tag{4.1}$$

where

$$\phi_0(x) = |x| - 1. \quad \forall x \in \mathbb{R}$$

Note that no boundary conditions are needed in this case as the PDE holds in the whole domain $\mathbb{R}$.

4.2 Analytic Solution of 1D Level Set Equation

Let's consider the following one-dimensional level set equation, which is called the advection equation; the following description to find the analytic solution is based on [76]:

In order to solve (4.1) with initial condition $\phi_0(x)$, we will use the method of characteristic; that is, let us reduce this problem to an ordinary differential equation case.

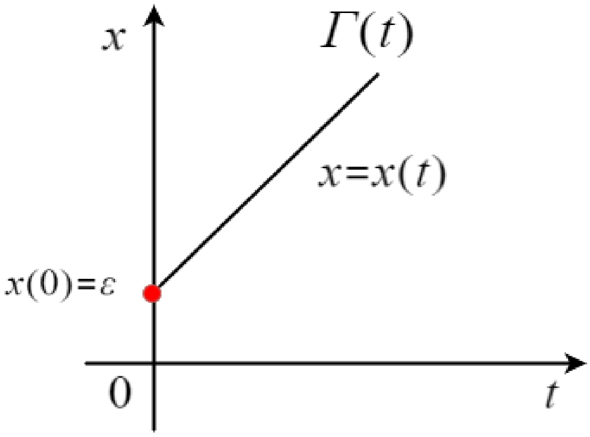


Figure 4.5: A characteristic curve.

Let $x(t) \in \Gamma(t)$ be a curve with $x(0) = \epsilon$ (Figure 4.5) where Γ is a surface and the slope of $x(t)$ is given by:

$$\frac{d}{dt}x(t) = c, \tag{4.2}$$

and let's define the curve $x(t)$ such that

$$\frac{d}{dt}\phi(t, x(t)) = \frac{\partial}{\partial t}\phi(t, x) + \mathbf{u} \cdot \frac{\partial}{\partial x}\phi(t, x) = 0. \tag{4.3}$$

Applying the chain rule of differentiation on the left side of (4.3), we get

$$\frac{d}{dt}\phi(t, x(t)) = \frac{\partial \phi}{\partial t}\frac{dt}{dt} + \frac{\partial \phi}{\partial x}\frac{dx}{dt} = \frac{\partial \phi}{\partial t} + \frac{\partial \phi}{\partial x}\frac{dx}{dt}. \tag{4.4}$$

By Lagrangian formulation, assume that velocity field $\mathbf{u}(x, t)$ is given for each $x(t) \in \Gamma(t)$, then the movement of Γ is described by solving the EDO

$$\frac{d}{dt}x(t) = \mathbf{u}(x, t). \tag{4.5}$$

If we consider a constant velocity field, the slope of $x(t)$ described by (4.5) matches with (4.2); that is,

$$\frac{d}{dt}x(t) = \mathbf{u}(x,t) = c, \tag{4.6}$$

then the equation (4.4) is equivalent to (4.3), in other words, the ODE on the left-hand side of (4.3) is equal to PDE. By (4.4) and (4.6), we have that

$$\frac{d}{dt}\phi(t, x(t)) = \frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x}, \tag{4.7}$$

It follows that the solution of the level set equation can be obtained from the ODE;

$$\frac{d}{dt}\phi(t, x(t)) = 0, \tag{4.8}$$

along any associated curves to $x(t)$, which are the solution curves of equation given by

$$\frac{d}{dt}x(t) = c. \tag{4.9}$$

A characteristic curve of LSE is a curve given by $x = x(t)$, where $x(t)$ is a solution of the differential equation (4.9). Clearly, the value of ϕ stays constant along such curves. Thus the solution of (4.1) is reduced to find the solution from the system of EDOs given by

$$\frac{d\phi}{dt} = 0, \tag{4.10}$$

$$\frac{dx}{dt} = 0. \tag{4.11}$$

Integrating (4.11), we get

$$\int \frac{d}{dt}x(t)dt = \int cdt = ct + \epsilon,$$

where ϵ is the x-intercept of the curve. It shows that the characteristic curves are straight lines (Fig. 4.5) with slope given by (4.2). Moreover, we know that ϕ is a constant along a given characteristic curve its value can be defined from the initial condition, that is;

$$\phi(t, x) = \phi(0, \epsilon) = \phi_0(\epsilon).$$

Since $\epsilon = x - ct$, the solution of PDE in (4.1) is given by

$$\phi(t, x) = \phi_0(x - \mathbf{u}t). \tag{4.12}$$

4.3 Finite Difference Method

4.3.1 Upwind Schemes

Let us define a grid of points in the $(x, t)-$plane. Let Δx and Δt be positive numbers; then the points on the grid is defined by

$$(x_i, t_n) = (i\Delta x, n\Delta t),$$

with uniform spacing Δx, some time step Δt, and arbitrary integer number i and n (Fig. 4.6). Moreover, a function ϕ is defined on the grid as

$$\phi_i^n = \phi(x_i, t_n).$$

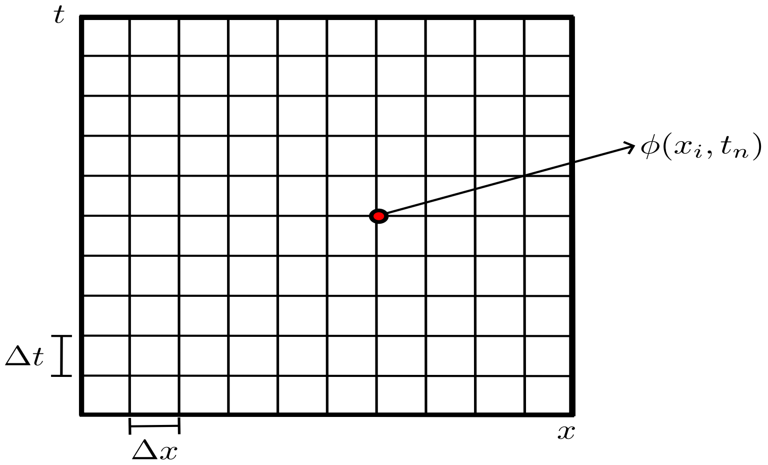


Figure 4.6: Grid for the discretization of Level Set Equation

Using the method of characteristic [18], the discretization of (4.1) is based on two cases; $u > 0$ and $u > 0$, since u is constant.

Case: $u > 0$. First, let us consider a first-order accurate method to approximate the time derivative $\phi_t = \partial\phi/\partial t$; it is called the forward Euler method; i.e,

$$\frac{\partial \phi}{\partial t} \approx \frac{\phi_i^{n+1} - \phi_i^n}{\Delta t}. \tag{4.13}$$

Since the solution to the level set equation depends on u, we have that

$$\phi(x_i, t^{n+1}) = \phi(x_i - u\Delta t, t^n).$$

Since $u > 0$, it follows

$$x_i - u\Delta t < x_i.$$

In this sense, ϕ_i^{n-1} is necessary to find the value ϕ_i^n. Therefore, Backward schemes is used to approximate $\partial\phi/\partial x$, i.e.,

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi_i^n - \phi_{i-1}^n}{\Delta x}, \tag{4.14}$$

so, the discretization of (4.1) defined by (4.19) and (4.20);

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + u\frac{\phi_i^n - \phi_{i-1}^n}{\Delta x} = 0, \tag{4.15}$$

with $\phi_i^0 = \phi(x_i, t^0)$ that is the initial data. By (4.22), in order to find the numerical solution ϕ_i^{n+1} at $t^n + \Delta t$ is given by

$$\phi_i^{n+1} = \phi_i^n + u\frac{\Delta t}{\Delta x}(\phi_i^n - \phi_{i-1}^n).$$

Case: $u < 0$. In the same way, if $u < 0$, then

$$\phi(x_i, t^{n+1}) = \phi(x_i + u\Delta t, t^n),$$

it follows that

$$x_i + u\Delta t > x_i.$$

The information of ϕ_i^n is necessary to find the value of ϕ_i^{n+1}, here Forward scheme is used to approximate $\partial\phi/\partial x$, i.e.,

$$\frac{\partial \phi}{\partial x} \approx \frac{\phi_{i+1}^n - \phi_i^n}{\Delta x}. \tag{4.16}$$

By (4.19) and (4.23), the discretization of (4.1) when $u < 0$ is given by

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + u\frac{\phi_{i+1}^n - \phi_i^n}{\Delta x} = 0,$$

then, the numerical solution ϕ_i^{n+1} is given by,

$$\phi_i^{n+1} = \phi_i^n + u\frac{\Delta t}{\Delta x}(\phi_{i+1}^n - \phi_i^n).$$

Summary about numerical solutions of Level set equation based on upwind schemes;

(a) If $u > 0$, then

$$\phi_i^{n+1} = \phi_i^n + u\frac{\Delta t}{\Delta x}(\phi_i^n - \phi_{i-1}^n). \tag{4.17}$$

(b) If $u < 0$, then

$$\phi_i^{n+1} = \phi_i^n + u\frac{\Delta t}{\Delta x}(\phi_{i+1}^n - \phi_i^n). \tag{4.18}$$

4.3.2 Essentially Non-oscillatory Schemes

One dimensional scheme

We describe ENO method basen on [21, 9];

Let us consider a partition of interval $[a, b]$ defined as;

$$a = x_0 < x_1 < \ldots < x_{i-1} < x_i < x_{i+1} < \ldots < x_n = b,$$

with uniform spacing Δx. Let's consider the discretize function ϕ as

$$\phi_n = \phi(x_n),$$

and define the following differences based on forward and backward differences;

$$\Delta^+\phi_n = \phi_{n+1} - \phi_n, \tag{4.19}$$

$$\Delta^-\phi_n = \phi_n - \phi_{n-1}. \tag{4.20}$$

In order to approximate ϕ_x with ϕ_x^-, let's consider a left-based stencil (Fig.4.21) given by the subset of partition;

$$\{x_n, n = i-3, i-2, i-1, i, i+1, i+2\}. \tag{4.21}$$

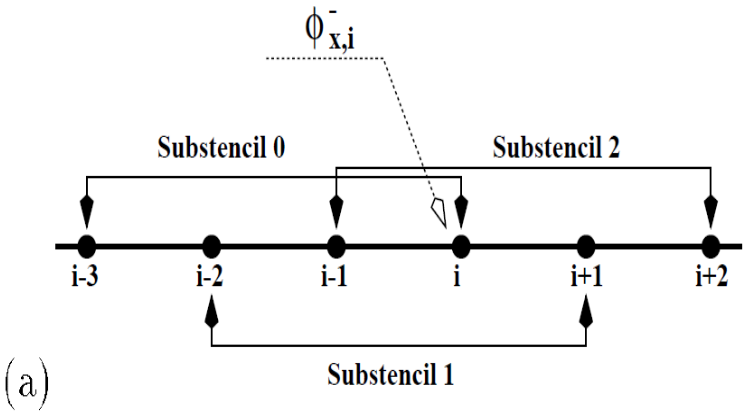


Figure 4.7: Left based stencil and s^{th} stencils for construct $\phi^-_{x,i}$. Source [9]

Then, Jian [9] defined suitable three order approximations for $\phi^-_{x,i}$ as follows

$$\phi^{-,0}_{x,i} = a_0\frac{\Delta^+\phi_{i-3}}{\Delta x} + b_0\frac{\Delta^+\phi_{i-2}}{\Delta x} + c_0\frac{\Delta^+\phi_{i-1}}{\Delta x}, \tag{4.22}$$

$$\phi^{-,1}_{x,i} = a_1\frac{\Delta^+\phi_{i-2}}{\Delta x} + b_1\frac{\Delta^+\phi_{i-1}}{\Delta x} + c_1\frac{\Delta^+\phi_{i}}{\Delta x}, \tag{4.23}$$

$$\phi^{-,2}_{x,i} = a_2\frac{\Delta^+\phi_{i-1}}{\Delta x} + b_2\frac{\Delta^+\phi_{i}}{\Delta x} + c_3\frac{\Delta^+\phi_{i+1}}{\Delta x}. \tag{4.24}$$

The appropriate coefficients $a_s, b_s, c_s, s = 0, 1, 2$ developed by Jiang [21] (Table. 4.1). In general, $\phi^{-,s}_{x,i}$, gives the three order accurate ENO method to approximate $\phi_{x,i}$ based on the s^{th} sub-stencil of (4.21);

$$\{x_n, n = i+s-3, i+s-2, i+s-1, i+s\}.$$

Now, in order to choose which between (4.22), (4.23) and (4.24) are suitable, it is based on the relative "smoothness" of ϕ on the sub-stencils. In this sense, Jiang [9] described the

s	a_s	b_s	c_s
0	1/3	−7/6	11/6
1	−1/6	5/6	1/3
2	1/3	5/6	−1/6

Table 4.1: Coefficients for $\phi^{-,s}_{x,i}$ and $\phi^{+,s}_{x,i}$ for $s = 0, 1, 2$

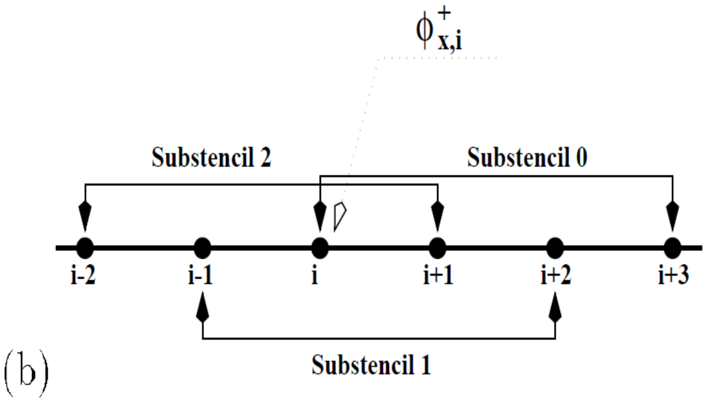


Figure 4.8: Right based stencil and s^{th} stencils for construct $\phi^+_{x,i}$. source [9]

following smooth condition as follows

$$\begin{aligned}
\alpha &= |\Delta^-\Delta^+\phi_{i-1}| < |\Delta^-\Delta^+\phi_i|, \\
\beta &= |\Delta^-\Delta^-\Delta^+\phi_{i-1}| < |\Delta^+\Delta^-\Delta^+\phi_{i-1}|, \\
\gamma &= |\Delta^-\Delta^+\phi_{i-1}| > |\Delta^-\Delta^+\phi_i|, \\
\nu &= |\Delta^-\Delta^-\Delta^+\phi_i| > |\Delta^+\Delta^-\Delta^+\phi_i|.
\end{aligned}$$

By these conditions and $\phi^{-,s}_{x,i}, s = 0, 1, 2$. The three order ENO approximation for $\phi_{x,i}$ with $\phi^-_{x,i}$ is given by

$$\phi^-_{x,i} = \begin{cases} \phi^{-,0}_{x,i}, & \text{if } \alpha \text{ and } \beta \text{ are satisfied} \\ \phi^{-,2}_{x,i}, & \text{if } \gamma \text{ and } \nu \text{ are satisfied} \\ \phi^{-,1}_{x,i}, & \text{otherwise} \end{cases} \tag{4.25}$$

In the same way, let us describe the approximation of $\phi_{x,i}$ with $\phi^+_{x,i}$. Now, let us consider a right-biased stencil (Fig.4.8) given by

$$\{x_n, n = i-2, i-1, i, i+1, i+2, i+3\}.$$

So, the third order approximations to $\phi^{+}_{x,i}$ as follows

$$\begin{aligned}
\phi^{+,0}_{x,i} &= a_0\frac{\Delta^-\phi_{i+3}}{\Delta x} + b_0\frac{\Delta^-\phi_{i+2}}{\Delta x} + c_0\frac{\Delta^-\phi_{i+1}}{\Delta x}, && (4.26)\\
\phi^{+,1}_{x,i} &= a_1\frac{\Delta^-\phi_{i+2}}{\Delta x} + b_1\frac{\Delta^-\phi_{i+1}}{\Delta x} + c_1\frac{\Delta^-\phi_{i}}{\Delta x}, && (4.27)\\
\phi^{+,2}_{x,i} &= a_2\frac{\Delta^-\phi_{i+1}}{\Delta x} + b_2\frac{\Delta^-\phi_{i}}{\Delta x} + c_3\frac{\Delta^-\phi_{i-1}}{\Delta x}. && (4.28)
\end{aligned}$$

The coefficients are equal to $\phi^{-,2}_{x,i}, s = 0, 1, 2$. By (4.26), (4.27), (4.28), and same conditions, one can approximate $\phi_{x,i}$ by $\phi^{+}_{x,i}$ as follows;

$$\phi^{+}_{x,i} = \begin{cases} \phi^{+,0}_{x,i}, & \text{if } \alpha \text{ and } \beta \text{ are satisfied} \\ \phi^{+,2}_{x,i}, & \text{if } \gamma \text{ and } \nu \text{ are satisfied} \\ \phi^{+,1}_{x,i}, & \text{otherwise} \end{cases} \tag{4.29}$$

4.3.3 Weighted Essentially Non-oscillatory Schemes

One dimensional scheme

We describe the WENO approach proposed by [21, 9]. Let's define WENO approximation of $\phi_{x,i}$ as convex combination of $\phi^{-,s}_{x,i}, s = 0, 1, 2$, i,e.,

$$\phi^{-}_{x,i} = \omega_0\phi^{-,0}_{x,i} + \omega_1\phi^{-,1}_{x,i} + \omega_2\phi^{-,2}_{x,i}, \tag{4.30}$$

with $\omega_s \geq 0$ which are the weights associated with s^{th} sub-stencil and they satisfy the following condition

$$\omega_0 + \omega_1 + \omega_2 = 1. \tag{4.31}$$

Jiang [21] mentioned that the formula (4.30) is the fifth order approximation to $\phi_{x,i}$ and it produces the smallest truncation error on such a six-point stencil.

In order to find optimal weights that satisfy the ENO properties and the fifth-order accuracy, the authors proposed two fundamental features on weights to build the WENO schemes;

(1) If ϕ remains smooth on the entire stencil, then the weights must satisfy $\omega_s = C_s + O(\Delta x^2)$, in this case, the WENO approximation 4.30 is uniformly fifth order accurate.

(2) If ϕ contains a discontinuity on the stencil, the weights approach to ENO with 1 or 0 to avoid oscillations.

So, the WENO schemes are described as follows;

From (4.31), we get

$$\omega_1 = 1 - \omega_0 - \omega_2. \tag{4.32}$$

Replacing (4.32) into 4.30, we have that

$$\begin{aligned}
\phi_{x,i}^{-} &= \omega_0\phi_{x,i}^{-,0} + (1-\omega_0-\omega_2)\phi_{x,i}^{-,1} + \omega_2\phi_{x,i}^{-,2} \\
&= \omega_0\phi_{x,i}^{-,0} + \phi_{x,i}^{-,1} - \omega_0\phi_{x,i}^{-,1} - \omega_2\phi_{x,i}^{-,1} + \omega_2\phi_{x,i}^{-,2} \\
&= \omega_0\phi_{x,i}^{-,0} + \frac{1}{2}\phi_{x,i}^{-,1} + \frac{1}{2}\phi_{x,i}^{-,1} - \omega_0\phi_{x,i}^{-,1} - \omega_2\phi_{x,i}^{-,1} + \omega_2\phi_{x,i}^{-,2} + \frac{1}{2}\phi_{x,i}^{-,2} - \frac{1}{2}\phi_{x,i}^{-,2} \\
&= \frac{1}{2}(\phi_{x,i}^{-,1} + \phi_{x,i}^{-,2}) + \omega_0(\phi_{x,i}^{-,0} - \phi_{x,i}^{-,1}) + \left(\omega_2 - \frac{1}{2}\right)(\phi_{x,i}^{-,2} - \phi_{x,i}^{-,1}). \qquad (4.33)
\end{aligned}$$

Note that $\frac{1}{2}(\phi_{x,i}^{-,1} + \phi_{x,i}^{-,2})$ does not depend on ω_s. Replacing $\phi_{x,i}^{-,s}, s = 0,1,2$ into (4.33); we have that

$$\begin{aligned}
\phi_{x,i}^{-} = & \frac{1}{12}\left(-\frac{\Delta^+\phi_{i-2}}{\Delta x} + 7\frac{\Delta^+\phi_{i-1}}{\Delta x} + 7\frac{\Delta^+\phi_{i}}{\Delta x} - \frac{\Delta^+\phi_{i+1}}{\Delta x}\right) \\
& +\omega_0\left(\frac{1}{3}\frac{\Delta^+\phi_{i-3}}{\Delta x} - \frac{\Delta^+\phi_{i-2}}{\Delta x} + \frac{\Delta^+\phi_{i-1}}{\Delta x} - \frac{1}{3}\frac{\Delta^+\phi_{i}}{\Delta x}\right) \\
& + \left(\omega_2 - \frac{1}{2}\right)\left(\frac{1}{6}\frac{\Delta^+\phi_{i-2}}{\Delta x} - \frac{1}{2}\frac{\Delta^+\phi_{i-1}}{\Delta x} + \frac{1}{2}\frac{\Delta^+\phi_{i}}{\Delta x} - \frac{1}{6}\frac{\Delta^+\phi_{i+1}}{\Delta x}\right).
\end{aligned}$$

Using (4.19) and (4.20), the last terms can be rewritten as function called ϕ^{WENO} given by

$$\phi^{WENO}(a,b,c,d) = \frac{1}{3}\omega_0(a - 2b + c) + \frac{1}{6}\left(\omega_2 - \frac{1}{2}\right), \tag{4.34}$$

where

$$\begin{aligned} a &= \frac{\Delta^-\Delta^+\phi_{i-2}}{\Delta x}, \\ b &= \frac{\Delta^-\Delta^+\phi_{i-1}}{\Delta x}, \\ c &= \frac{\Delta^-\Delta^+\phi_{i}}{\Delta x}, \\ d &= \frac{\Delta^-\Delta^+\phi_{i+1}}{\Delta x}. \end{aligned}$$

Therefore, the approximation of $\phi_{x,i}$ by $\phi^-_{x,i}$ as follows;

$$\phi^-_{x,i} = \frac{1}{12}\left(-\frac{\Delta^+\phi_{i-2}}{\Delta x} + 7\frac{\Delta^+\phi_{i-1}}{\Delta x} + 7\frac{\Delta^+\phi_{i}}{\Delta x} - \frac{\Delta^+\phi_{i+1}}{\Delta x}\right) - \phi^{WENO}(a,b,c,d). \quad (4.35)$$

Moreover, Jiang [9] proposed the general computed weights for (4.35) as;

$$\begin{aligned} \omega_0 &= \frac{\alpha_0}{\alpha_0+\alpha_1+\alpha_2}, \\ \omega_2 &= \frac{\alpha_2}{\alpha_0+\alpha_1+\alpha_2} \end{aligned}$$

where

$$\begin{aligned} \alpha_0 &= \frac{1}{(\epsilon + IS_0)^2}, \\ \alpha_1 &= \frac{6}{(\epsilon + IS_1)^2}, \\ \alpha_2 &= \frac{3}{(\epsilon + IS_2)^2} \end{aligned}$$

and

$$\begin{aligned} IS_0 &= 13(a-b)^2 + 3(a-3b)^2, \\ IS_1 &= 13(b-c)^2 + 3(b+c)^2, \\ IS_2 &= 13(c-d)^2 + 3(3c-d)^2. \end{aligned}$$

Notice that ϵ is used to prevent denominators are zero and the optimal $\epsilon = 10^{-6}$.

In the same way, the approximation of $\phi_{x,i}$ by $\phi^+_{x,i}$ as follows

$$\phi^+_{x,i} = \frac{1}{12}\left(-\frac{\Delta^-\phi_{i-2}}{\Delta x} + 7\frac{\Delta^-\phi_{i-1}}{\Delta x} + 7\frac{\Delta^-\phi_{i}}{\Delta x} - \frac{\Delta^-\phi_{i+1}}{\Delta x}\right) - \phi^{WENO}(a,b,c,d). \quad (4.36)$$

4.3.4 TVD-Runge Kutta Scheme

One dimensional scheme

We describe the third order accurate TVD-Runge Kutta scheme proposed in [24, 18].

Let us consider an Euler step to find the solution at time $t^n + \Delta t$;

$$\frac{\phi^{n+1} - \phi^n}{\Delta x} + u \cdot \phi^n_x = 0, \quad (4.37)$$

followed by a second Euler step to find the solution at time $t^n + 2\Delta t$ as follows

$$\frac{\phi^{n+2} - \phi^{n+1}}{\Delta t} + u \cdot \phi^{n+1}_x = 0. \quad (4.38)$$

Taking the average of (4.38) and ϕ^n at time t^n, we have that

$$\phi^{n+\frac{1}{2}} = \frac{3}{4}\phi^n + \frac{1}{4}\phi^{n+2}, \quad (4.39)$$

where $\phi^{n+\frac{1}{2}}$ is an approximation of solution at time $t^n + \frac{1}{2}\Delta t$. So, using another Euler step to find the solution to time $t^n + \frac{3}{2}\Delta t$, it is the second step of the Runge Kutta method,

$$\frac{\phi^{n+\frac{3}{2}} - \phi^{n+\frac{1}{2}}}{\Delta t} + u \cdot \phi^{n+\frac{1}{2}}_x = 0. \quad (4.40)$$

Again, taking the average of (4.40) and ϕ^n at time t^n; we get the solution at $t^n + \Delta t$ given by

$$\phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\phi^{n+\frac{3}{2}}. \quad (4.41)$$

4.4 Physics Informed Neural Networks for Level Set Method

In this section, we described two models to infer solutions to the problem (4.1) using the syntaxis of models developed by Raissi et al., in [25]. Mainly, we defined the physics-informed neural networks for two cases and loss functions; for the level set equation of the problem (4.1), which is given by;

$$\phi_t + u \cdot \phi_x = 0, \quad t \in [0, T], \quad x \in [-2, 2] \tag{4.42}$$

with constant velocity field u and $\phi_0(x)$ a signed distance function.

4.4.1 Discrete Time Model

First, let us assume that $\phi(x,t)$ is a signed distance function, which is approximated by a neural network $\hat{\phi}(x,t;\theta)$, i.e.,

$$\phi(x,t) \approx \hat{\phi}(x,t;\theta) = \hat{\phi}_\theta(x,t), \tag{4.43}$$

and it satisfies the Eikonal equation

$$\left|\nabla \hat{\phi}_\theta(x,t)\right| = 1.$$

From (4.43), we can rewritten the equation (4.42) as;

$$\frac{\partial}{\partial t}\hat{\phi}_\theta(x,t) + u \cdot \frac{\partial}{\partial x}\hat{\phi}_\theta(x,t) = 0. \tag{4.44}$$

Now, let us define a linear differential operator F as;

$$F(\hat{\phi}_\theta(x,t), u) = u \cdot \frac{\partial}{\partial x}\hat{\phi}_\theta(x,t). \tag{4.45}$$

Replacing (4.45) onto (4.44), we get

$$\frac{\partial}{\partial t}\hat{\phi}_\theta(x,t) + F(\hat{\phi}_\theta(x,t), u) = 0. \tag{4.46}$$

Applying the general form of Runge-Kutta methods with q stages [77] to equation (4.46), we obtain

$$\begin{aligned} \hat{\phi}_{\theta}^{n+c_j} &= \hat{\phi}_{\theta}^{n} - \Delta t \sum_{j=i}^{q} a_{ij} F(\hat{\phi}_{\theta}^{n+c_j}, u), \quad i = 1, ..., q \qquad (4.47) \\ \hat{\phi}_{\theta}^{n+1} &= \hat{\phi}_{\theta}^{n} - \Delta t \sum_{j=1}^{q} b_j F(\hat{\phi}_{\theta}^{n+c_j}, u), \qquad (4.48) \end{aligned}$$

where $\hat{\phi}_{\theta}^{n+c_j}(x,t) = \hat{\phi}_{\theta}(x, t^n + c_j \Delta t)$ for $j = 1, ..., q$. This general form encapsulates both implicit and explicit time-stepping schemes, depending on the choice of the parameters $\left\{a_{ij}, b_j, c_j\right\}$. The formulas (4.47) and (4.48) can be equivalently expressed as

$$\begin{aligned} \hat{\phi}_{\theta}^{n} &= \hat{\phi}_{\theta,i}^{n}, \quad i = 1, ...q, \\ \hat{\phi}_{\theta}^{n} &= \hat{\phi}_{\theta,q+1}^{n}, \end{aligned}$$

where

$$\begin{aligned} \hat{\phi}_{\theta,i}^{n} &:= \phi_{\theta}^{n+c_i} + \Delta t \sum_{j=1}^{q} a_{ij} F(\hat{\phi}_{\theta}^{n+c_j}, u), \quad i = 1, ..., q. \qquad (4.49) \\ \hat{\phi}_{\theta,q+1}^{n} &:= \phi_{\theta}^{n+1} + \Delta t \sum_{j=1}^{q} b_j F(\hat{\phi}_{\theta}^{n+c_j}, u). \qquad (4.50) \end{aligned}$$

We proceed by placing a multi-output neural network prior on

$$\left[\hat{\phi}_{\theta}^{n+c_1}(x), \ldots, \hat{\phi}_{\theta}^{n+c_q}(x), \hat{\phi}_{\theta}^{n+1}(x)\right]. \qquad (4.51)$$

This prior assumption coupled with formulas (4.49) and (4.50) result in a **physics informed neural network** that takes x as an input and outputs

$$\left[\hat{\phi}_{\theta,1}^{n}(x), \ldots, \hat{\phi}_{\theta,q}^{n}(x), \hat{\phi}_{\theta,q+1}^{n}(x)\right]. \qquad (4.52)$$

Moreover, the adjustment of shared parameters between the neural networks (4.51) and (4.52) by minimizing a loss function $L(\theta)$, i.e.,

$$\theta^* = \arg\min_{\theta} L(\theta),$$

where $L(\theta) = SSE_{Fn} + SSE_n$; SSE_n is defined by

$$SSE_n = \sum_{j}^{q+1} \sum_{i}^{N_n} \left\| \hat{\phi}^n_{\theta,j}(x^{n,i}) - \phi^{n,i} \right\|^2 ,$$

where the finite set $\left\{ x^{n,i}, \phi^{n,i} \right\}_{i=1}^{N_n}$ is the data at time-step t^n.

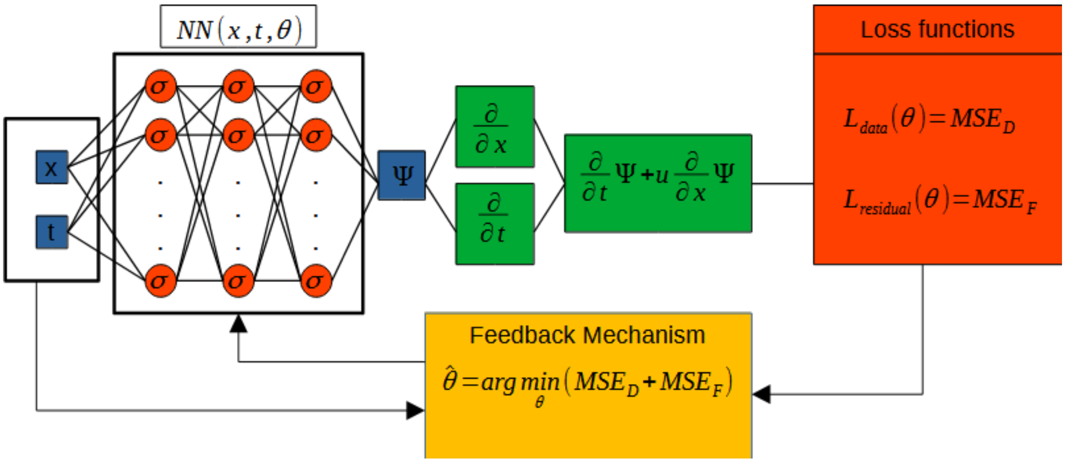


Figure 4.9: Neuronal Network Structure of Level Set Method for One-Dimensional

4.4.2 Continuous Time Model

In the same way, let us assume that $\phi(x,t)$ is a signed distance function, which is approximated by a neural network $\hat{\phi}(x,t;\theta)$, i.e.,

$$\phi(x,t) \approx \hat{\phi}(x,t;\theta) = \hat{\phi}_\theta(x,t), \tag{4.53}$$

and it satisfies the Eikonal equation. θ are weights and biases of the neural network (Fig. 4.9). Now, let us define the linear operator differential $F(\hat{\phi}(x,t;\theta),u)$ by the left-hand-side of equation (4.42) as;

$$F(\hat{\phi}(x,t;\theta),u) = \frac{\partial}{\partial t}\hat{\phi}_\theta(x,t) + u \cdot \frac{\partial}{\partial x}\hat{\phi}_\theta(x,t). \tag{4.54}$$

The assumption (4.53) coupled with definition (4.54) result in a **physics informed neural network** $F(\hat{\phi}(x,t;\theta),u)$. Then, the equation (4.42) can be rewritten as;

$$F(\hat{\phi}(x,t;\theta),u) = \frac{\partial}{\partial t}\hat{\phi}_\theta(x,t) + u \cdot \frac{\partial}{\partial x}\hat{\phi}_\theta(x,t) = 0. \tag{4.55}$$

In order to compute the partial derivatives of neural network $\hat{\phi}(x,t;\theta)$, automatic differentiation is used. Moreover, the adjustment of shared parameters between the neural network $\hat{\phi}(x,t;\theta)$ and $F(\hat{\phi}(x,t;\theta),u)$ by minimizing a loss function $L(\theta)$, i.e.,

$$\theta^* = \arg\min_{\theta} L(\theta),$$

where $L(\theta) = MSE_\phi + MSE_F$, and the terms MSE_ϕ, MSE_F are defined by;

$$\begin{aligned} MSE_F =& \frac{1}{N_c}\sum_{i=1}^{N_c} \left\| F(\hat{\phi}_\theta(x_F^i, t_F^i), u) - 0 \right\| \\ =& \frac{1}{N_c}\sum_{i=1}^{N_c} \left\| F(\hat{\phi}_\theta(x_F^i, t_F^i), u) \right\| \end{aligned}$$

and

$$MSE_\phi = \frac{1}{N_\phi}\sum_{i=1}^{N_\phi} \left\| \hat{\phi}_\theta(x_\phi^i, t_\phi^i) - \phi^i \right\|$$

where the finite set $\left\{x_\theta^i, t_\phi^i, \phi^i\right\}_{i=1}^{N_\phi}$ denotes the initial training data generated by numerical methods. Moreover, the finite set $\left\{x_F^i, t_F^i\right\}_{i=1}^{N_c}$ are the collocations points for F.The term MSE_ϕ represents the loss function associated with the initial data and measures how well the model fits the training set. On the other hand, MSE_F corresponds to a loss function that enforces the structure imposed by equation (4.42) at a limited number of collocation points. This helps ensure that the model's predictions exhibit the desired behavior specified by the equation and fit the training data [25].

4.5 Data

In this section, we described the data structure of solutions generated by Upwind schemes for training neural networks, similarly for TVD/RK3 with ENO3 and WENO5.

In case 1D, the level set method is

$$\begin{aligned} \frac{\phi^{n+1} - \phi^n}{\Delta t} + u^n \phi_x^n = 0 \\ \frac{\phi^{n+1} - \phi^n}{\Delta t} = -u^n \phi_x^n \\ \phi^{n+1} = \phi^n - \Delta t u^n \phi_x^n \end{aligned}$$

We generate solutions from 0 to $n+1$ time:

$$\begin{aligned}
\phi^0 &= \text{Initial function}\\
\phi^1 &= \phi^0 - \Delta t u^0 \phi_x^0\\
\phi^2 &= \phi^1 - \Delta t u^1 \phi_x^1\\
\cdot &= \cdot\\
\cdot &= \cdot\\
\cdot &= \cdot\\
\phi^{n+1} &= \phi^n - \Delta t u^n \phi_x^n
\end{aligned}$$

Table 4.2 shows the data structure to train the Neural Networks has matrix form where the first column ϕ^0 represents the values of initial functions, and $\phi^1, \ldots, \phi^{n+1}$ are the columns with solutions that have been calculated by numerical methods mentioned in section 4.

ϕ^0	ϕ^1	ϕ^2	ϕ^3	$\cdots$	ϕ^{n+1}

Table 4.2: Data Structure

Chapter 5

Results and Discussion

In this section, we will present numerical solutions to the problem (4.1) at time $T = 50$; moreover, to compute the solution, we consider a computational domain on $[-2, 2]$ and a constant velocity field $\mathbf{u}(x, t) = 0.01$, then (4.1) is redefined as:

$$\begin{cases} \frac{\partial}{\partial t}\phi(x,t) + 0.01 \cdot \frac{\partial}{\partial x}\phi(x,t) = 0, & \text{in } [-2,2] \times [0,50] \\ \phi(x,0) = \phi_0(x) & \text{in } [-2,2] \end{cases} \tag{5.1}$$

and the initial condition is defined as

$$\phi_0(x) = |x| - 1, \quad \forall x \in [-2, 2].$$

Figure 5.1 shows the initial condition at $T = 0$ and the red points represents the interface $\Gamma = \{-1, 1\}$ given by zero level set $\phi_0(x) = 0$. Moreover, Figure 5.2 represents the property of the signed distance function, i.e., it satisfies the Eikonal equation;

$$|\nabla \phi| = 1.$$

Figure 5.3 shows the exact solution at time 50 that is given by

$$\phi(x, 50) = \phi_0(x - 0.5).$$

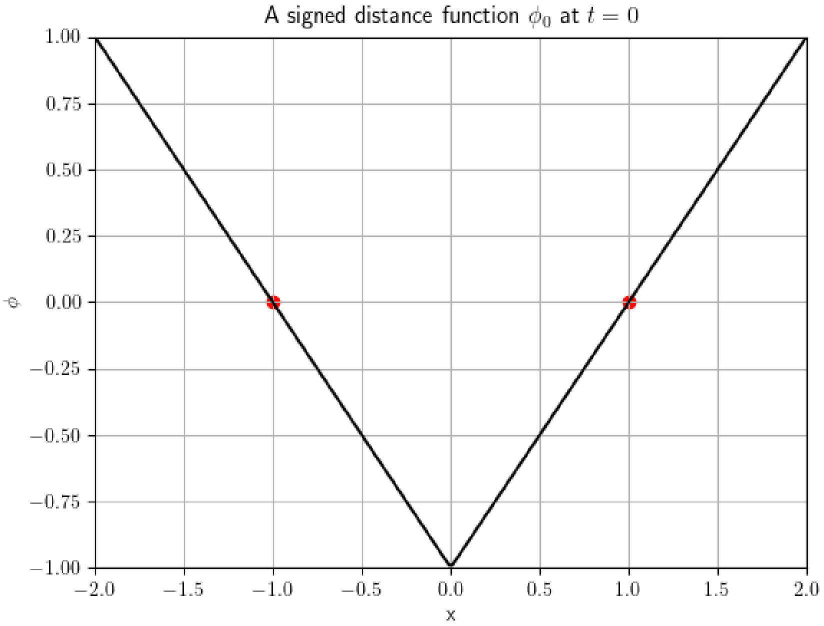


Figure 5.1: A initial condition given by a signed distance function $\phi_0(x)$.

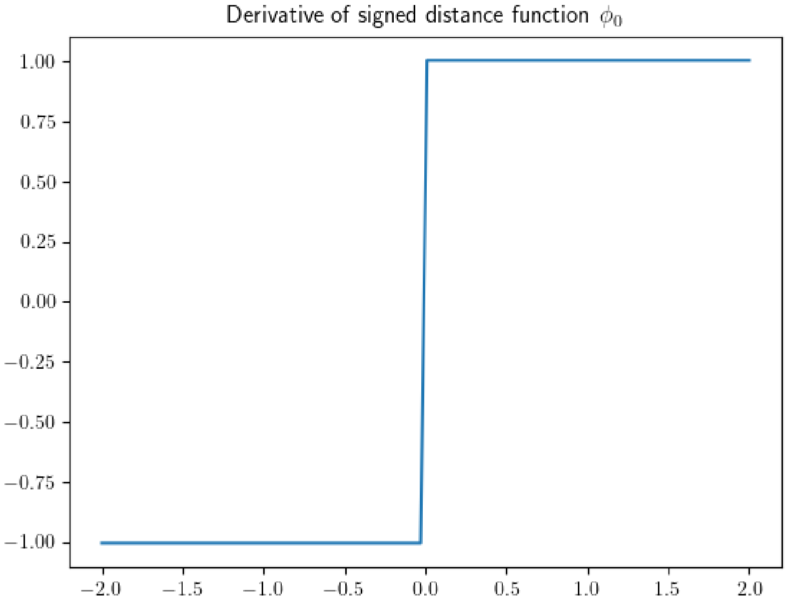


Figure 5.2: Derivative of the signed distance function $\phi_0(x)$ respect to x.

5.1 Numerical Solutions of Upwind Schemes

In this section, we presented the results obtained by upwind schemes to approximate the solution at $T = 50$ of the problem (5.1). Table 5.1 shows the values obtained by computing

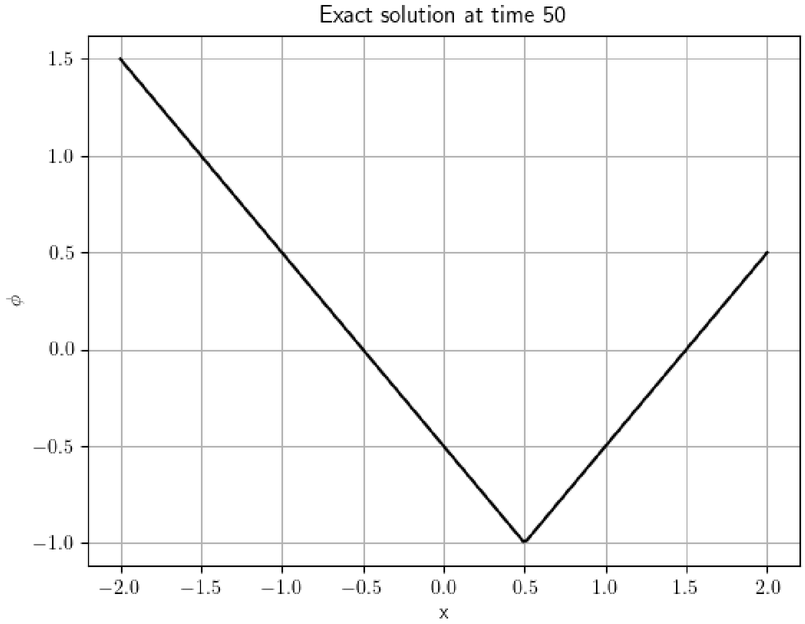


Figure 5.3: The exact solution of (4.1) at time 50.

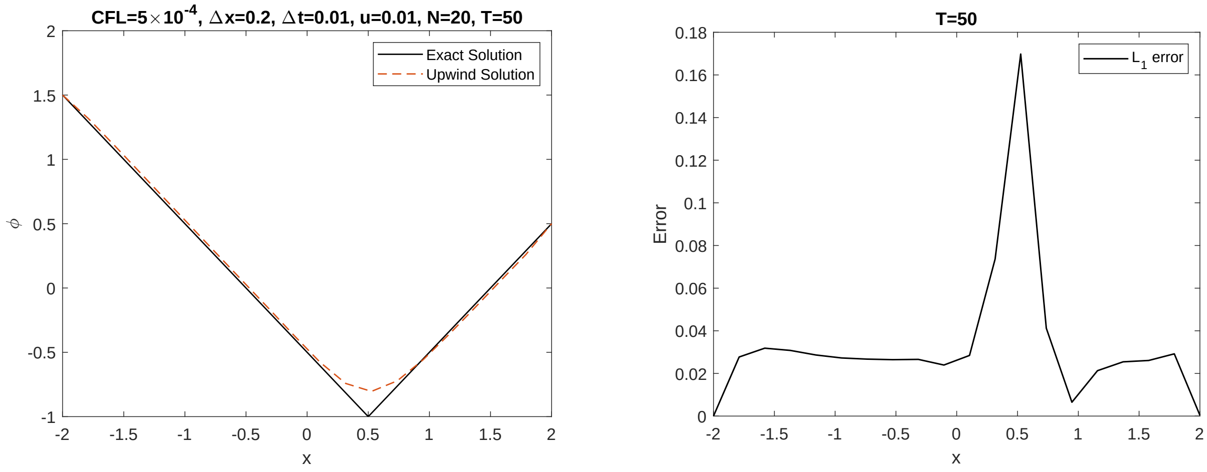
the L_1, L_2, and L_∞ errors between the exact and approximated solutions; the experiments were carried out by increasing the grid size, and the same way the execution time was determined to for each N. The order of convergence L_∞ and L_1 to measure how fast the algorithm converges to solution as the grid size increases.

Figure 5.4a shows the exact solution represented with a black line and the approximated solution with a red dashed line at $T = 50$ using $N = 20$ the grid size; moreover, under the CFL restriction 5×10^{-4}, and Figure 5.4b shows the representation of error L_1 computed for each point between exact and numerical solution. In the same way, Figures 5.5a, 5.6a, 5.7a, and 5.8a shows the approximation of solution at $T = 50$ for $N = 40$, $N = 80$, $N = 160$, and $N = 320$, respectively. Moreover, Figures 5.5b, 5.6b, 5.7b, and 5.8b represent their L_1 error in each point of exact solution.

Finally, Figure 5.9 shows the evolution of L_1 error as the grid size increases $2 \times N$ for upwind schemes.

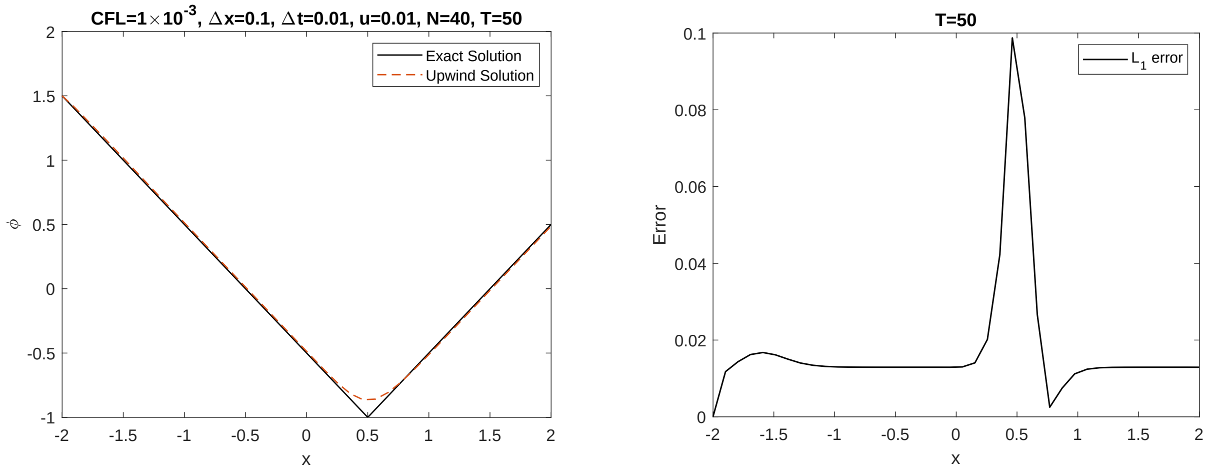
Method	N	L_∞ error	L_∞ order	L_1 error	L_1 order	L_2 error	Exec. time
Upwind	20	0.169	-	0.134	-	0.067	3.29 s
	40	0.099	0.77	0.073	0.88	0.035	6.45 s
	80	0.074	0.42	0.036	1.02	0.021	11.83 s
	160	0.055	0.43	0.018	1	0.012	23.84 s
	320	0.040	0.46	9×10^{-3}	1	0.007	45.96 s

Table 5.1: Results of numerical experiments for the first-order upwind scheme.



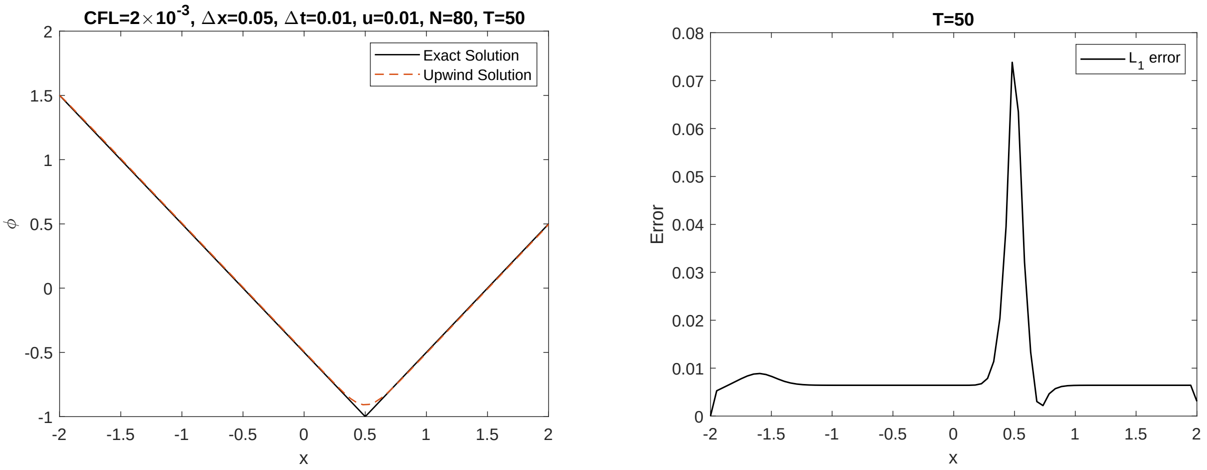
(a) Approx. vs Exact (b) L_1 error

Figure 5.4: (a) Numerical and exact solution at $T = 50$ for $N = 20$ (b) The L_1 error for each point between the exact and numerical solution $T = 50$.



(a) Approx. vs Exact (b) L_1 error

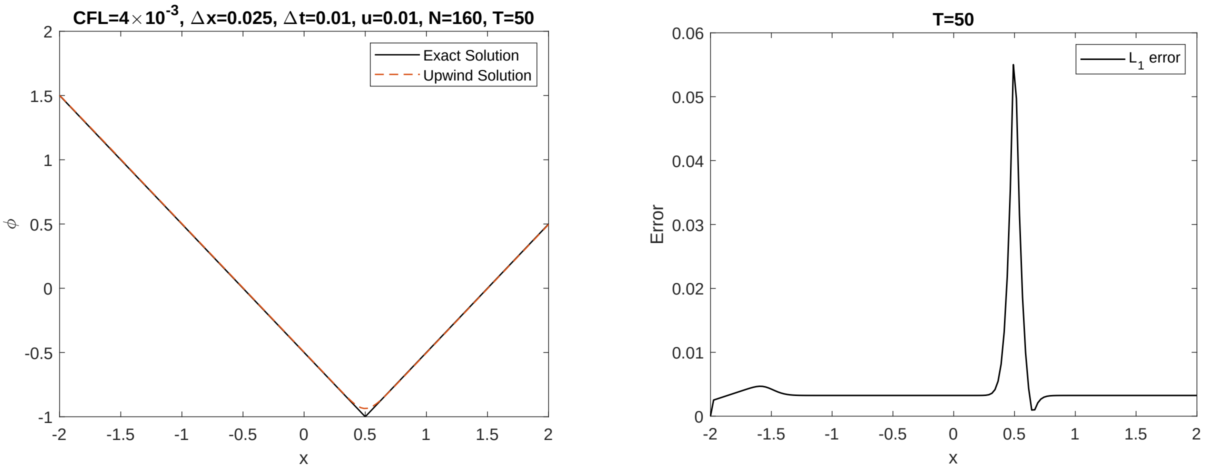
Figure 5.5: (a) Numerical and exact solution at $T = 50$ for $N = 40$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact

(b) L_1 error

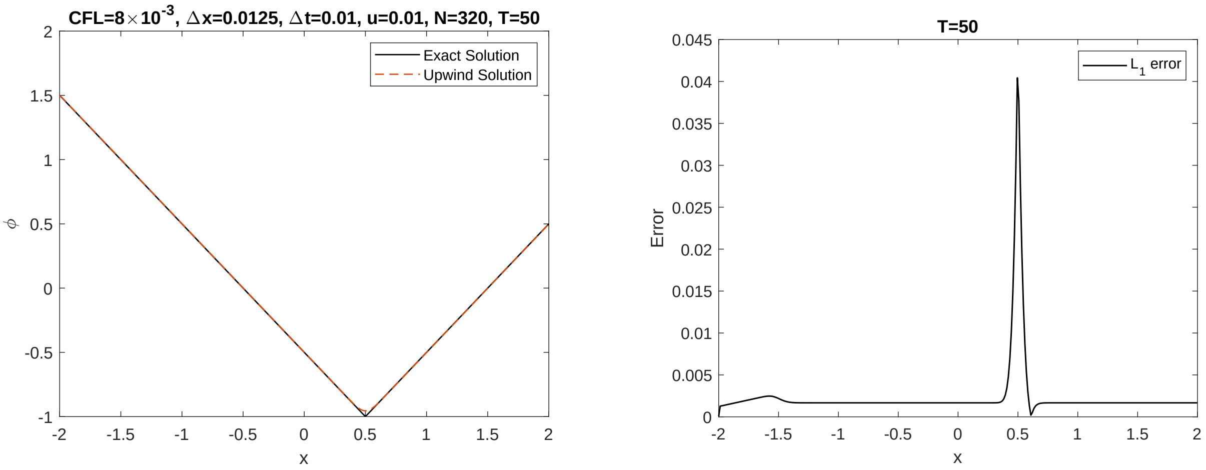
Figure 5.6: (a) Numerical and exact solution at $T = 50$ for $N = 80$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact

(b) L_1 error

Figure 5.7: (a) Numerical and exact solution at $T = 50$ for $N = 160$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact

(b) L_1 error

Figure 5.8: (a) Numerical and exact solution at $T = 50$ for $N = 320$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.

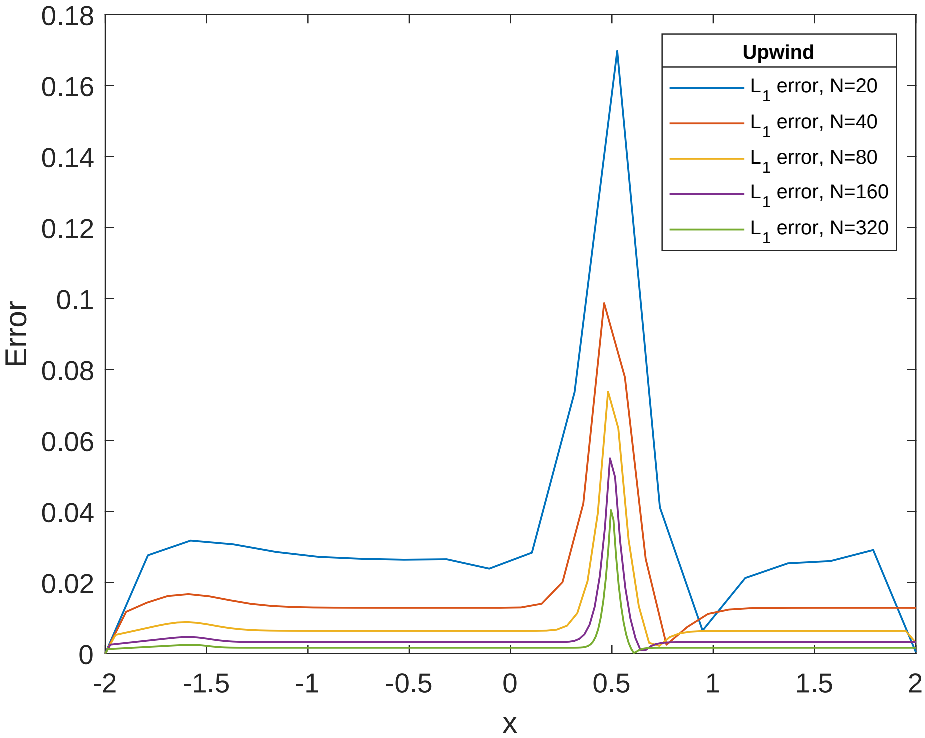


Figure 5.9: L_1 error for each solution at $T = 50$ that depends on size of grid; $N = 20$, $N = 40$, $N = 80$, $N = 160$, $N = 320$ using Upwind Schemes.

5.2 Numerical Solutions of ENO Schemes

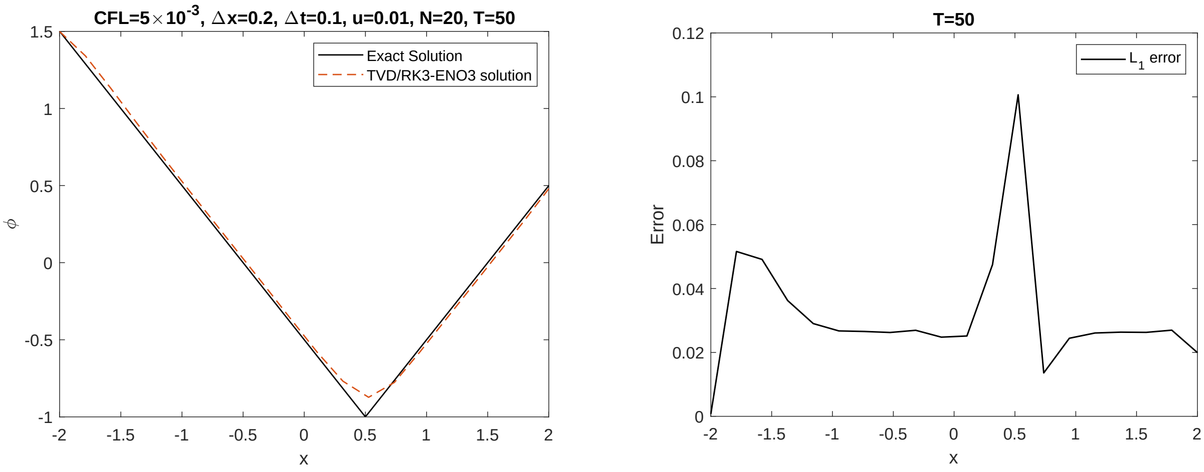
In a similar way, Table 5.5 described the results obtained by the third-order TVD/Runge Kutta coupled the third-order essentially non-oscillatory schemes to approximate the solution at $T = 50$ of the problem (5.1). Moreover, it shows the values obtained by computing the L_1, L_2, and L_∞ errors between the exact and approximated solutions; in order to analyze the approximated improvements on high-order methods, the experiments were carried out by increasing the grid size, and the same way the execution time was determined to for each N. The order of convergence L_∞ and L_1 to measure how fast the algorithm converges to solution as the grid size increases.

Figure 5.10a shows the exact solution represented with a black line and the approximated solution with a red dashed line at $T = 50$ using $N = 20$ the grid size; moreover, under the CFL restriction 5×10^{-3}, and Figure 5.10b shows the representation of error L_1 computed for each point between exact and numerical solution. In the same way, Figures 5.11a, 5.12a, 5.13a, and 5.14a shows the approximation of solution at $T = 50$ for $N = 40$, $N = 80$, $N = 160$, and $N = 320$, respectively. Moreover, Figures 5.11b, 5.12b, 5.13b, and 5.14b represent the graphs of L_1 error for each grid size.

Finally, Figure 5.15 shows the evolution of L_1 error as the grid size increases $2 \times N$ for TVD/RK3 coupled with ENO3.

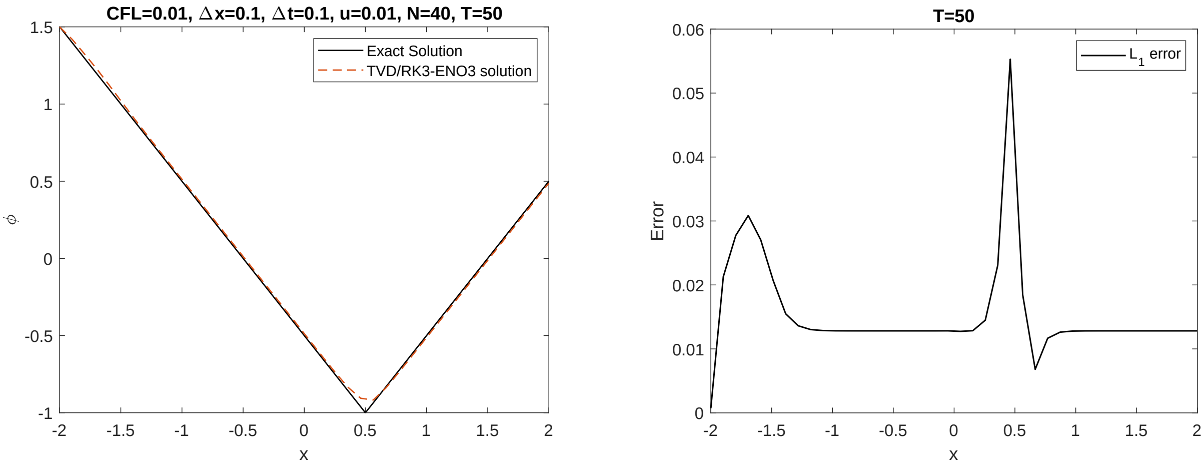
Method	N	L_∞ error	L_∞ order	L_1 error	L_1 order	L_2 error	Exec. time
TVD/RK3-ENO3	20	0.101	-	0.127	-	0.052	0.91 s
	40	0.055	0.88	0.062	1.03	0.025	1.63 s
	80	0.036	0.61	0.030	1.04	0.013	3.06 s
	160	0.023	0.64	0.015	1	0.006	6.35 s
	320	0.014	0.72	7×10^{-3}	1	0.003	12.64 s

Table 5.2: Results of numerical experiments for third order accuracy of TVD/Runge Kutta coupled with ENO3.



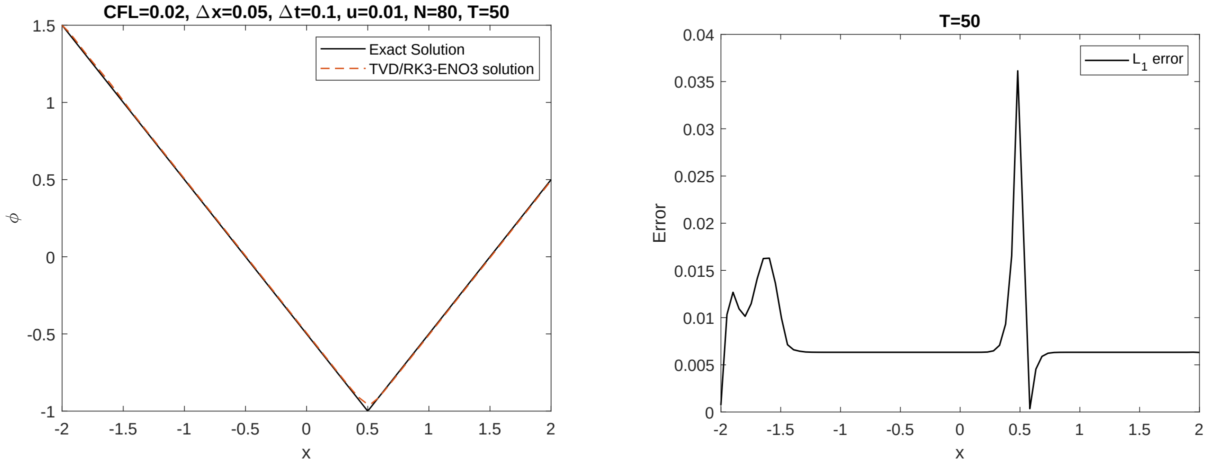
(a) Approx. vs Exact (b) L_1 error

Figure 5.10: (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 20$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



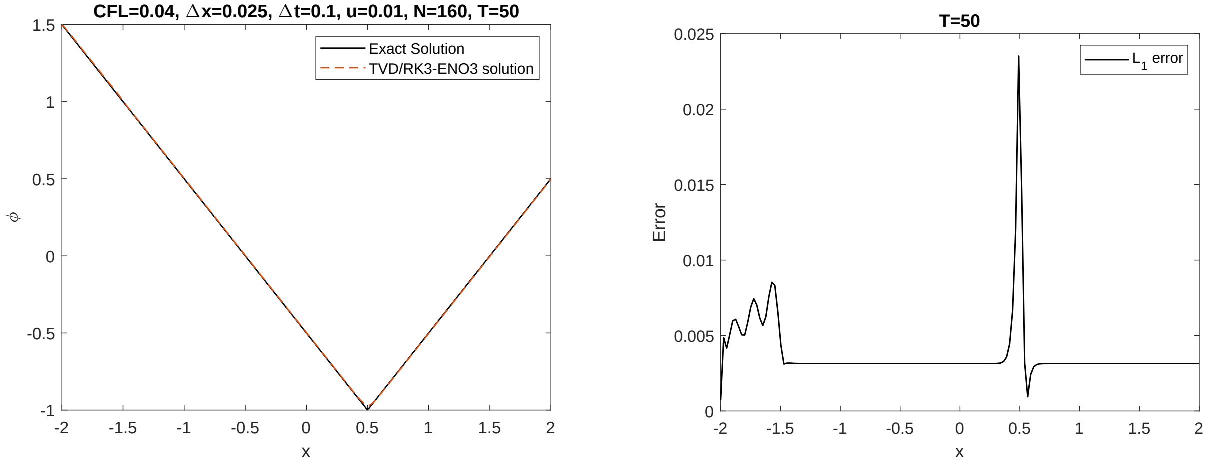
(a) Approx. vs Exact (b) L_1 error

Figure 5.11: (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 40$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



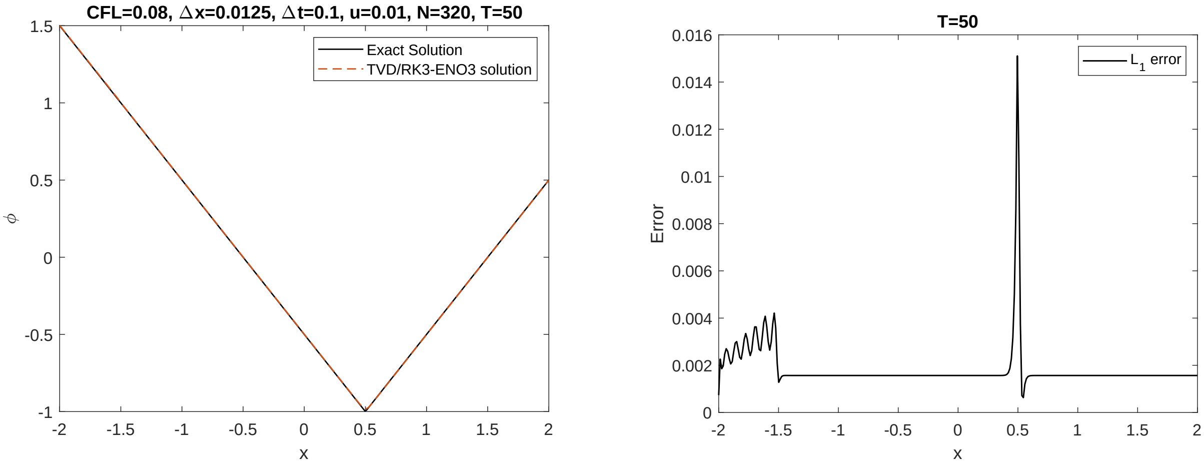
(a) Approx. vs Exact (b) L_1 error

Figure 5.12: (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 80$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact (b) L_1 error

Figure 5.13: (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 160$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact (b) L_1 error

Figure 5.14: (a) Numerical and exact solution at $T = 50$ obtained with third order TVD-Runge Kutta and ENO3 for $N = 320$, (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.

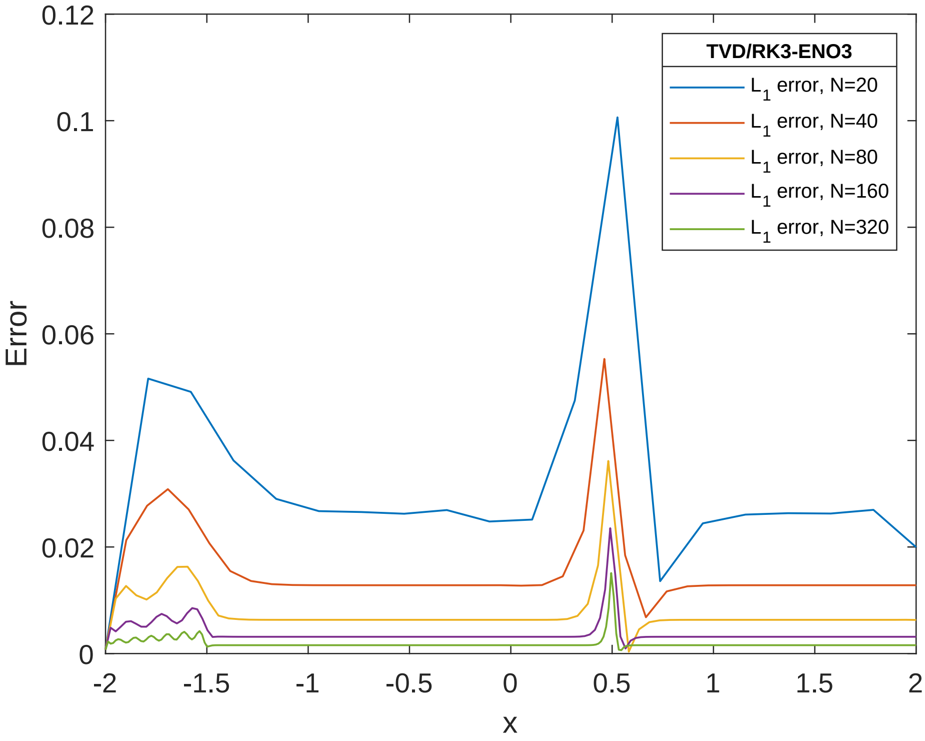


Figure 5.15: L_1 error for each solution at $T = 50$ that depends on the size of the grid; $N = 20$, $N = 40$, $N = 80$, $N = 160$, $N = 320$ using the third order TVD-Runge Kutta and Essentially Non-oscillatory schemes.

5.3 Numerical Solutions of WENO Schemes

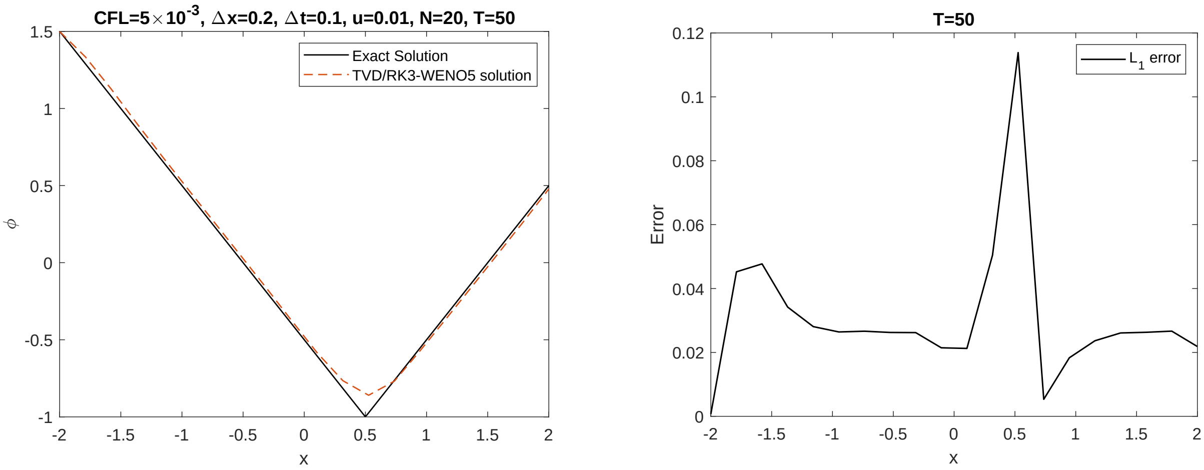
In the same way, we presented Table 5.3 that describes the results obtained by the third-order TVD/Runge Kutta coupled with the fifth-order weighted essentially non-oscillatory schemes (WENO5) to approximate the solution at $T = 50$ of the problem (5.1). Moreover, it shows the values obtained by computing the L_1, L_2, and L_∞ errors between the exact and approximated solutions; in order to analyze the approximated improvements on high-order methods, the experiments were carried out by increasing the grid size, and the same way the execution time was determined to for each N. The order of convergence L_∞ and L_1 to measure how fast the algorithm converges to solution as the grid size increases.

Figure 5.16a shows the exact solution represented with a black line and the approximated solution with a red dashed line at $T = 50$ using $N = 20$ the grid size; moreover, under the CFL restriction 5×10^{-3}, and Figure 5.16b shows the representation of error L_1 as a function which is computed for each point between exact and numerical solution. In the same way, Figures 5.17a, 5.18a, 5.19a, and 5.20a shows the approximation of solution at $T = 50$ for $N = 40$, $N = 80$, $N = 160$, and $N = 320$, respectively. Moreover, Figures 5.17b, 5.18b, 5.19b, and 5.20b represent the graphs of L_1 error for each grid size.

Finally, Figure 5.15 shows the evolution of L_1 error as the grid size increases $2 \times N$ for TVD/RK3 coupled with WENO5. As well as Figure 5.22 shows the solutions for each numerical method with grid size $N = 320$, and Figure 5.23 shows the comparison between L_1 error of each method to approximate the exact solution at $T = 50$.

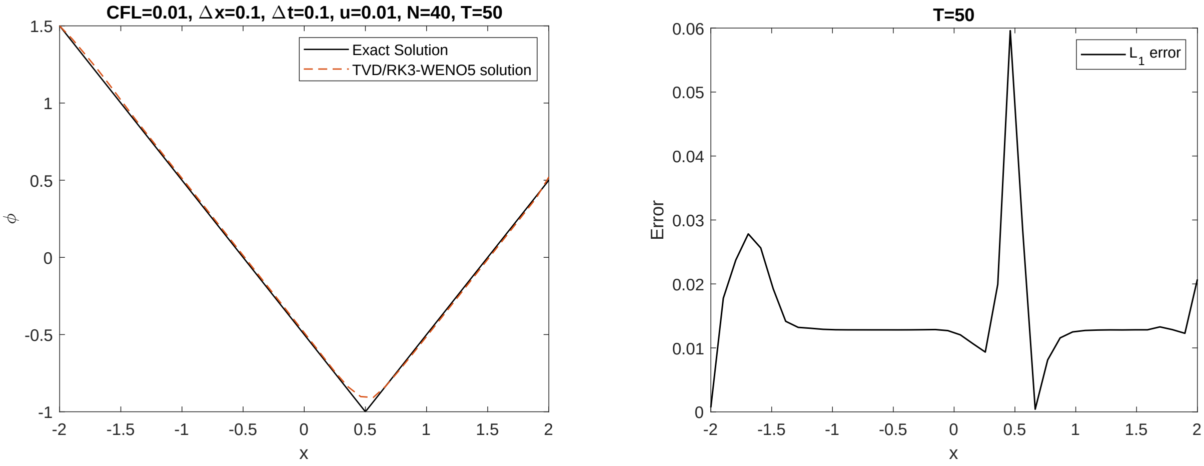
Method	N	L_∞ error	L_∞ order	L_1 error	L_1 order	L_2 error	Exec.time
TVD/RK3-WENO5	20	0.113	-	0.123	-	0.053	0.89 s
	40	0.059	0.93	0.060	1.03	0.025	1.65 s
	80	0.039	0.60	0.030	1	0.012	2.96 s
	160	0.025	0.64	0.014	1.09	0.006	5.90 s
	320	0.015	0.73	6.9×10^{-3}	1.02	0.003	11.78 s

Table 5.3: Results of numerical experiments for third order accuracy of TVD/Runge Kutta coupled with WENO5.



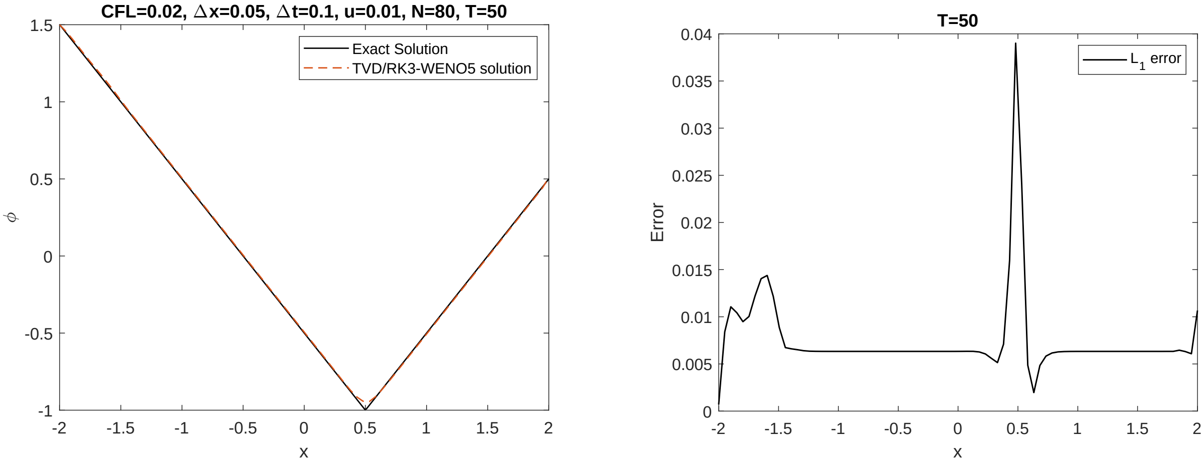
(a) Approx. vs Exact (b) L_1 error

Figure 5.16: (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 20$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



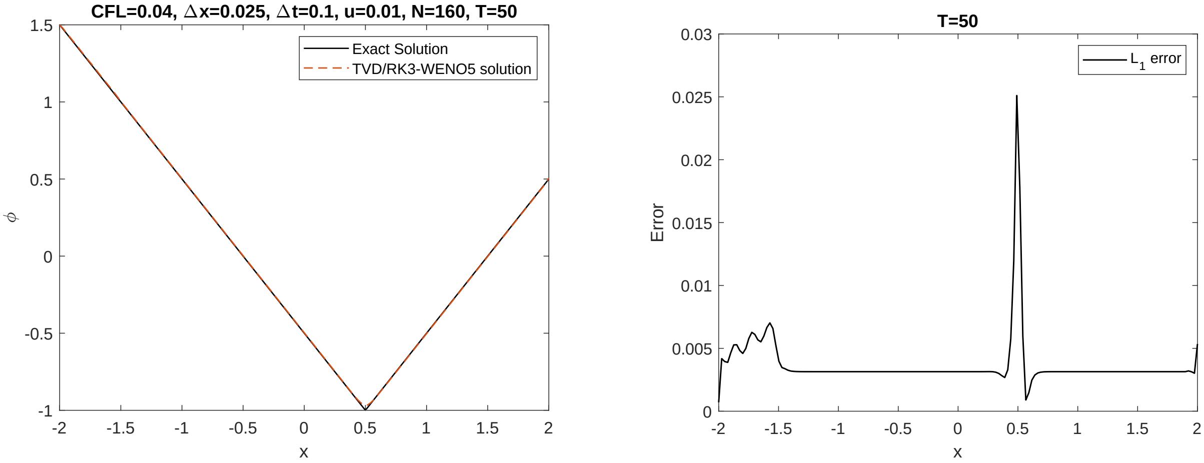
(a) Approx. vs Exact (b) L_1 error

Figure 5.17: (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 40$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



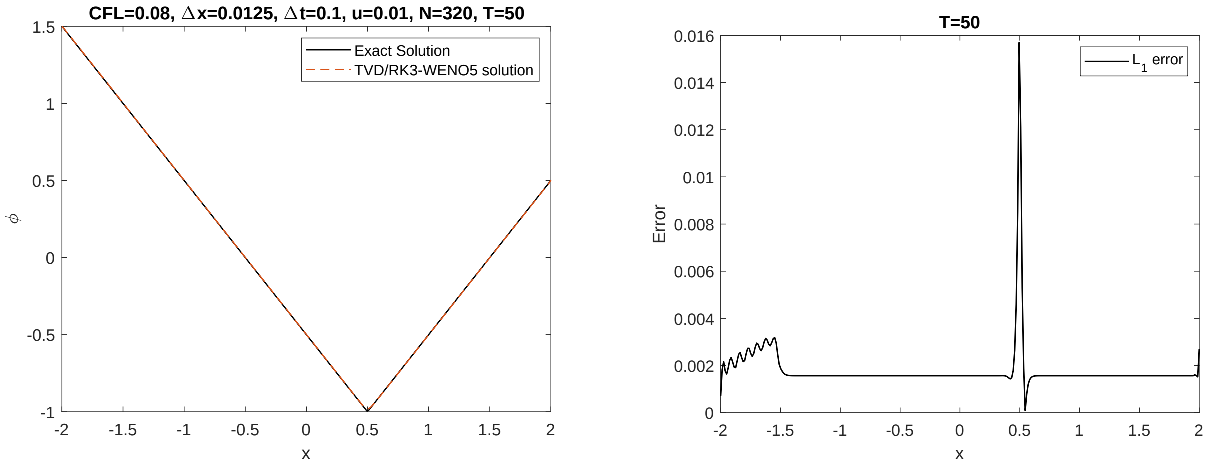
(a) Approx. vs Exact (b) L_1 error

Figure 5.18: (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 80$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact (b) L_1 error

Figure 5.19: (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 160$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.



(a) Approx. vs Exact (b) L_1 error

Figure 5.20: (a) Numerical and exact solution of problem (5.1) at $T = 50$ obtained with third order TVD-Runge Kutta and WENO5 for $N = 220$ (b) The L_1 error for each point between the exact and numerical solution at $T = 50$.

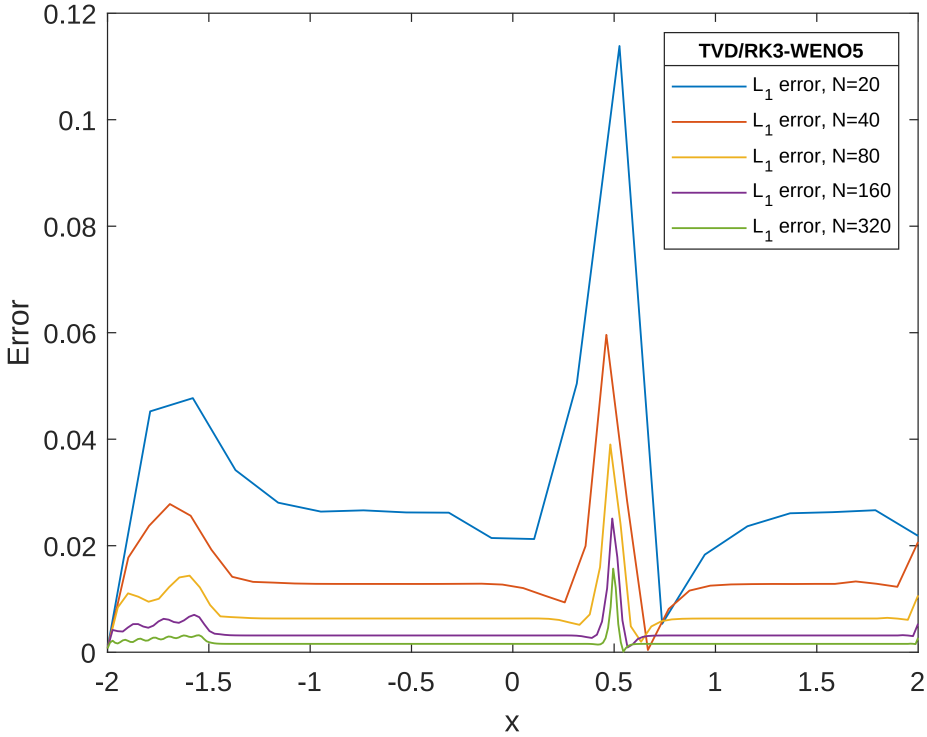


Figure 5.21: L_1 error for each solution at $T = 50$ that depends on the size of the grid; $N = 20$, $N = 40$, $N = 80$, $N = 160$, $N = 320$ using the third order Total Variation Diminishing (TVD)-Runge Kutta (RK) and the fifth order Weighted Essentially Non-oscillatory (WENO).

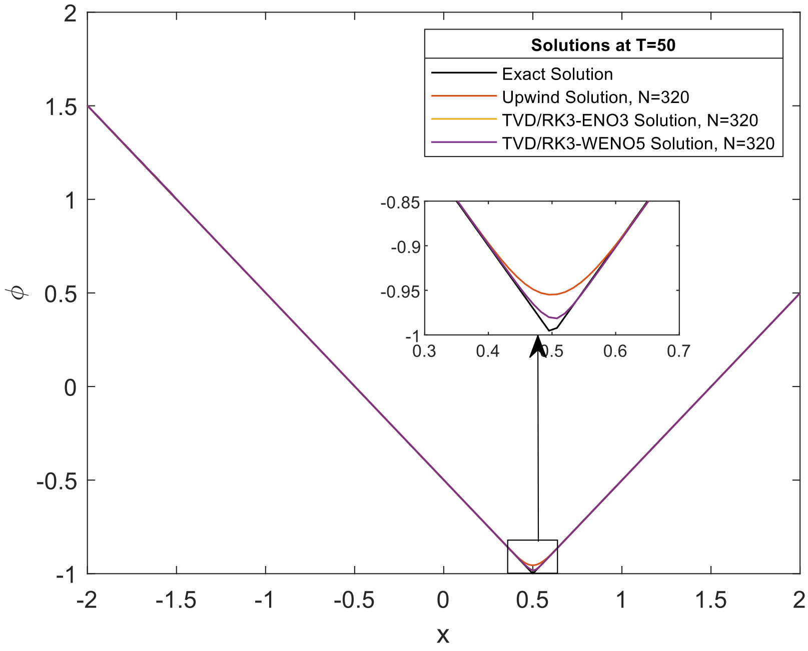


Figure 5.22: Solution at $T = 50$ for each numerical method: Upwind, TVD/RK3-ENO3 and TVD/RK3-WENO5.

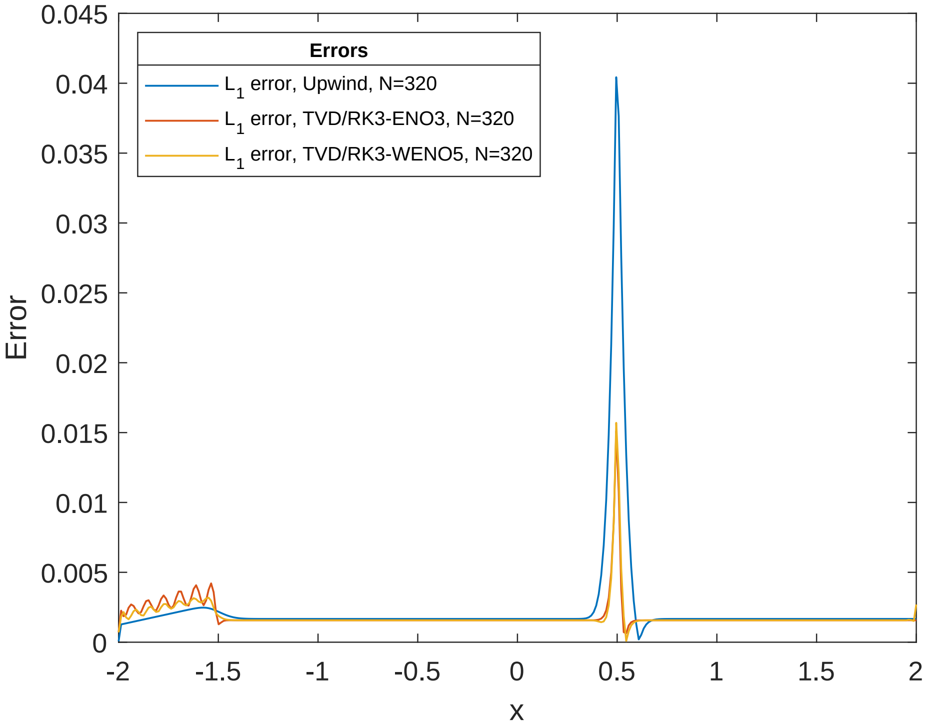


Figure 5.23: L_1 error for each solution at $T = 50$ generated by Upwind, the TVD/RK3 coupled with ENO3 and WENO5 using a size of grid $N = 320$.

5.4 Numerical Solutions of Level Set Method-Physics Informed Neural Networks

In this section, we present some numerical solutions of LSM-PINNs based on two models, the discrete-time model and continuous-time model described in chapter 4, in order to infer a solution at $T = 50$ of the problem (4.1). The training data are numerical solutions obtained from Upwind, Essentially Non-Oscillatory (ENO), and Weighted Essentially Non-Oscillatory (WENO) under the following parameters; $CFL = 0.08$, $\Delta x = 0.0125$, $\Delta t = 0.01$, $u = 0.01$, $N = 320$. Moreover, the optimization method used in all training for loss function is minibatch sampling with Adam and the limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm.

5.4.1 Discrete Model

Table 5.4 shows L_2 errors for measuring the accuracy of the model to approximate the exact solution; three scenarios were presented; first, we considered a neural network with 6-layers and 4 hidden layers, 50 neurons per hidden layer; to predict the solution at $T = 50$ using Upwind data at $T = 45$, as well as the same architecture for TVD/RK3 ENO3 and TVD/RK3 WENO5 data at $T = 45$. Also, the time of training was considered for each case.

Figure 5.24 shows the exact solution, numerical solution computed by upwind schemes, and inferred solution by the model when NN was trained with upwind data. Similarly, Figures 5.25 and 5.26 shows the prediction of solution at $T = 50$ using TVD/RK3 ENO3 and TVD/RK3 WENO5 data, respectively.

Data	Layers	Hidden L.	$N°$ N. per H-L	N_ϕ	L_2 error	Time T.
Upwind	6	4	50	200	5.98×10^{-2}	178.93 s
TVD/RK3-ENO3	6	4	50	200	5.87×10^{-2}	208.54 s
TVD/RK3-WENO5	6	4	50	200	5.86×10^{-2}	248.61 s

Table 5.4: Results from training neural network with different data.

Table 5.5 shows some experiments for increasing the layers, hidden layers, number of neurons per layer, and the number of data to minimize the loss function. Figures 5.28, 5.29, and 5.30 show these results, in the same way, exact solution, numerical solution, and inferred solution by model when we uses a different configuration of layers and neurons.

All training used the aspects; Latin Hypercube Sampling approach was used to generate all randomly sampled point locations, ensuring a space-filling distribution. Moreover, epochs = 10000, learning rate 0.001, and a hyperbolic tangent activation function. Finally, Figure 5.31 shows all solutions, both numerical and inferred by discrete time model; to compare the solutions at point $(0.5, -1)$, and Figure 5.27 shows the L_1 errors for each prediction of solution in Table 5.4, the error L_1 for the approximated solution from TVD7RK3-WENO5 was considered to see the difference of errors.

Data	Layers	Hidden Layers	N° neurons per H-L	N_ϕ	L_2 error
Upwind	6	4	50	80	0.217
	7	5	100	100	0.274
	8	6	150	120	0.425
TVD/RK3-ENO3	6	4	50	80	0.195
	7	5	100	100	0.249
	8	6	150	120	0.383
TVD/RK3-WENO5	6	4	50	80	0.193
	7	5	100	100	0.249
	8	6	150	120	0.378

Table 5.5: Numerical experiments based on increasing layers, hidden layers, and neurons in each hidden layer.

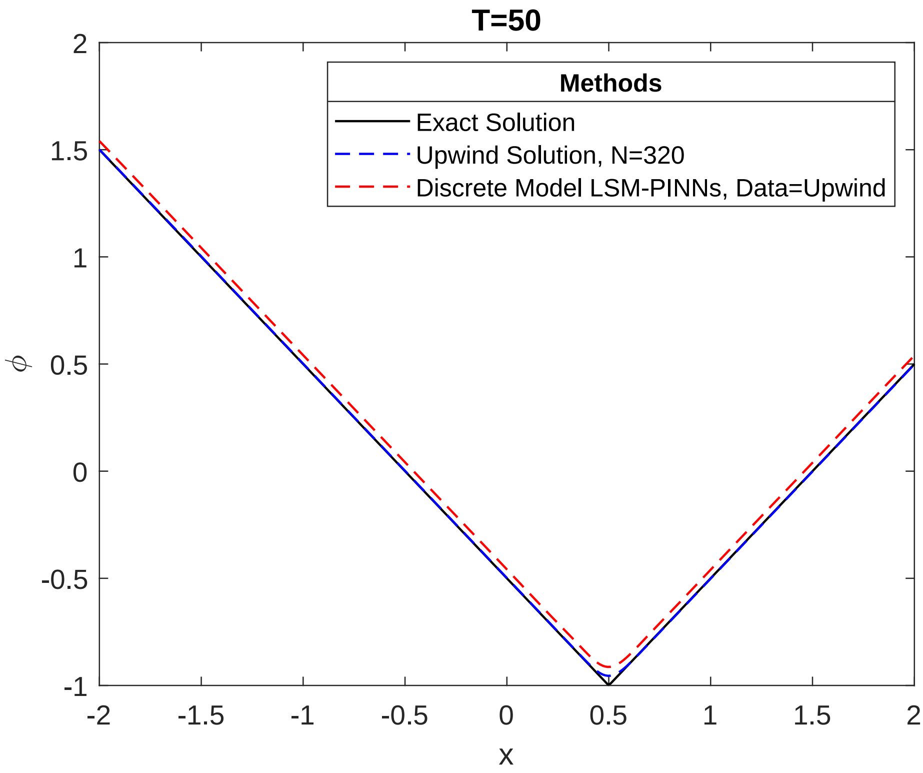


Figure 5.24: Exact, numerical and inferred solution using Upwind data.

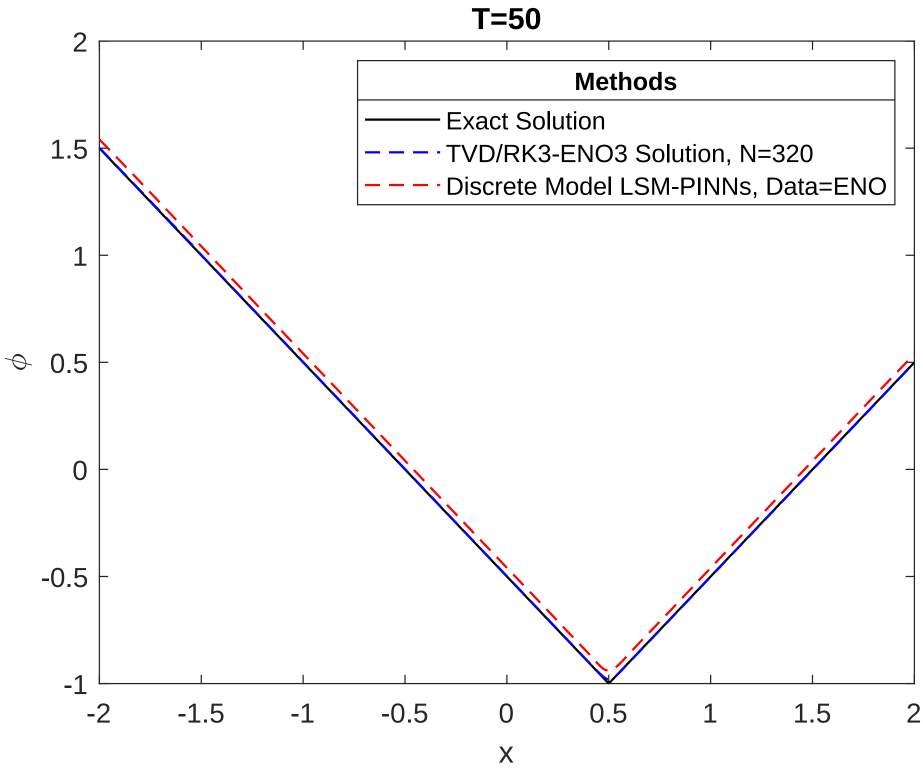


Figure 5.25: Exact, numerical and inferred solution using TVD/RK3-ENO3 data.

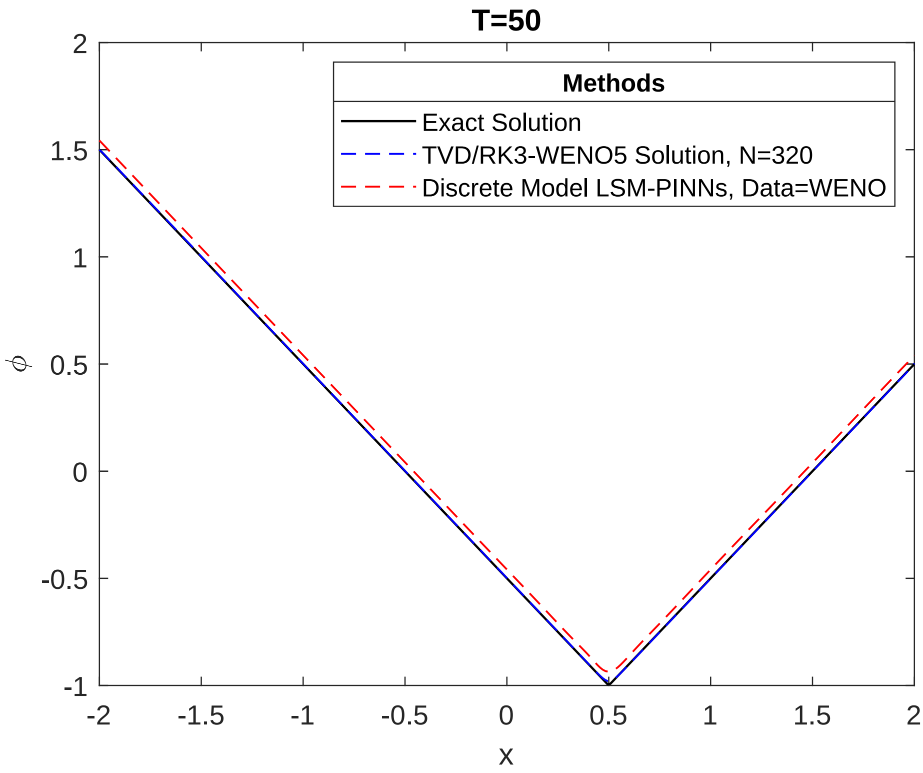


Figure 5.26: Exact, numerical and inferred solution using TVD/RK3-WENO5

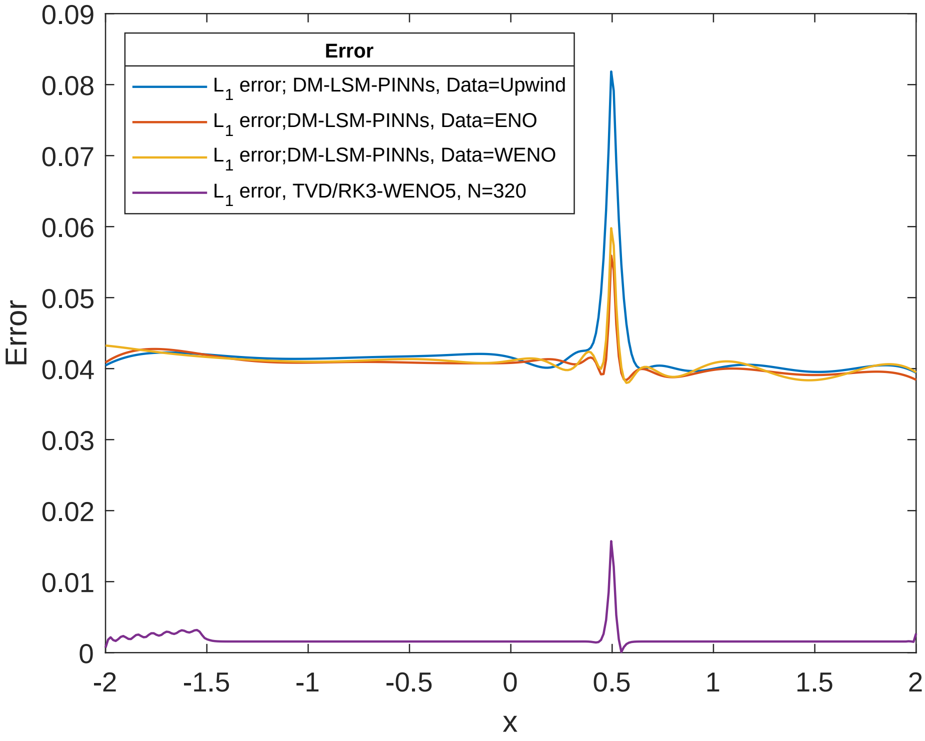
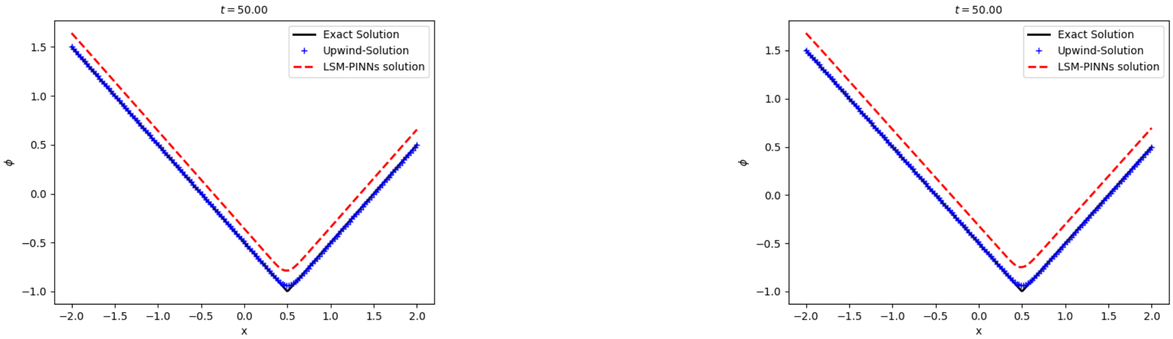
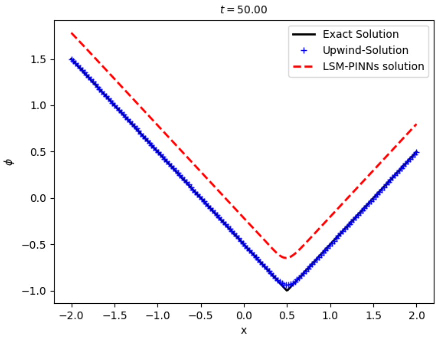


Figure 5.27: L_1 errors computed from exact solution and inferred solutions generated by discrete time model

.

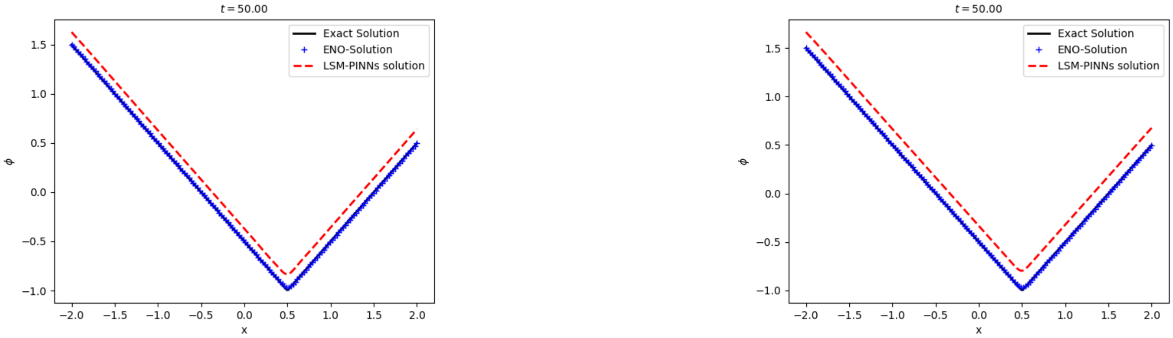


(a) 6-L, 4-HL, 50 neurons (b) 7-L, 5-HL, 100 neurons



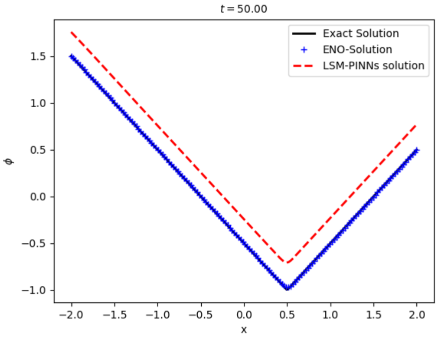
(c) 8-L, 6-HL, 150 neurons

Figure 5.28: Experiments of layers: (a), (b), (c) are graphs of inferred solution generated by discrete time model when the NN is feeding with data simulated by the first-order upwind scheme.



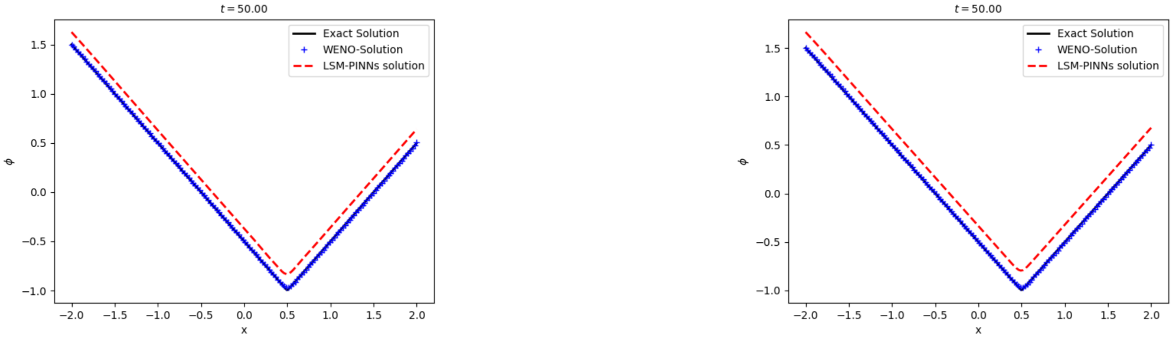
(a) 6-L, 4-HL, 50 neurons

(b) 7-L, 5-HL, 100 neurons



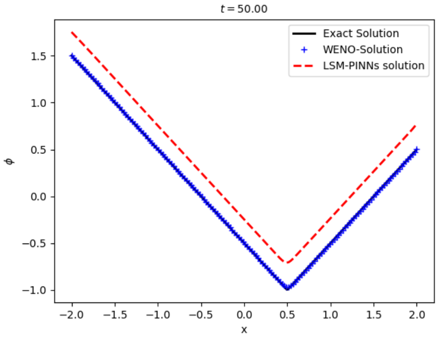
(c) 8-L, 6-HL, 150 neurons

Figure 5.29: Experiments of layers: (a), (b), (c) are graphs of inferred solution generated by discrete time model when the NN is feeding with data simulated by TVD/RK3-ENO3.



(a) 6-L, 4-HL, 50 neurons

(b) 7-L, 5-HL, 100 neurons



(c) 8-L, 6-HL, 150 neurons

Figure 5.30: Experiments of layers: (a), (b), (c) are graphs of the inferred solution generated by the discrete time model when the NN is feeding with data simulated by TVD/RK3-WENO5.

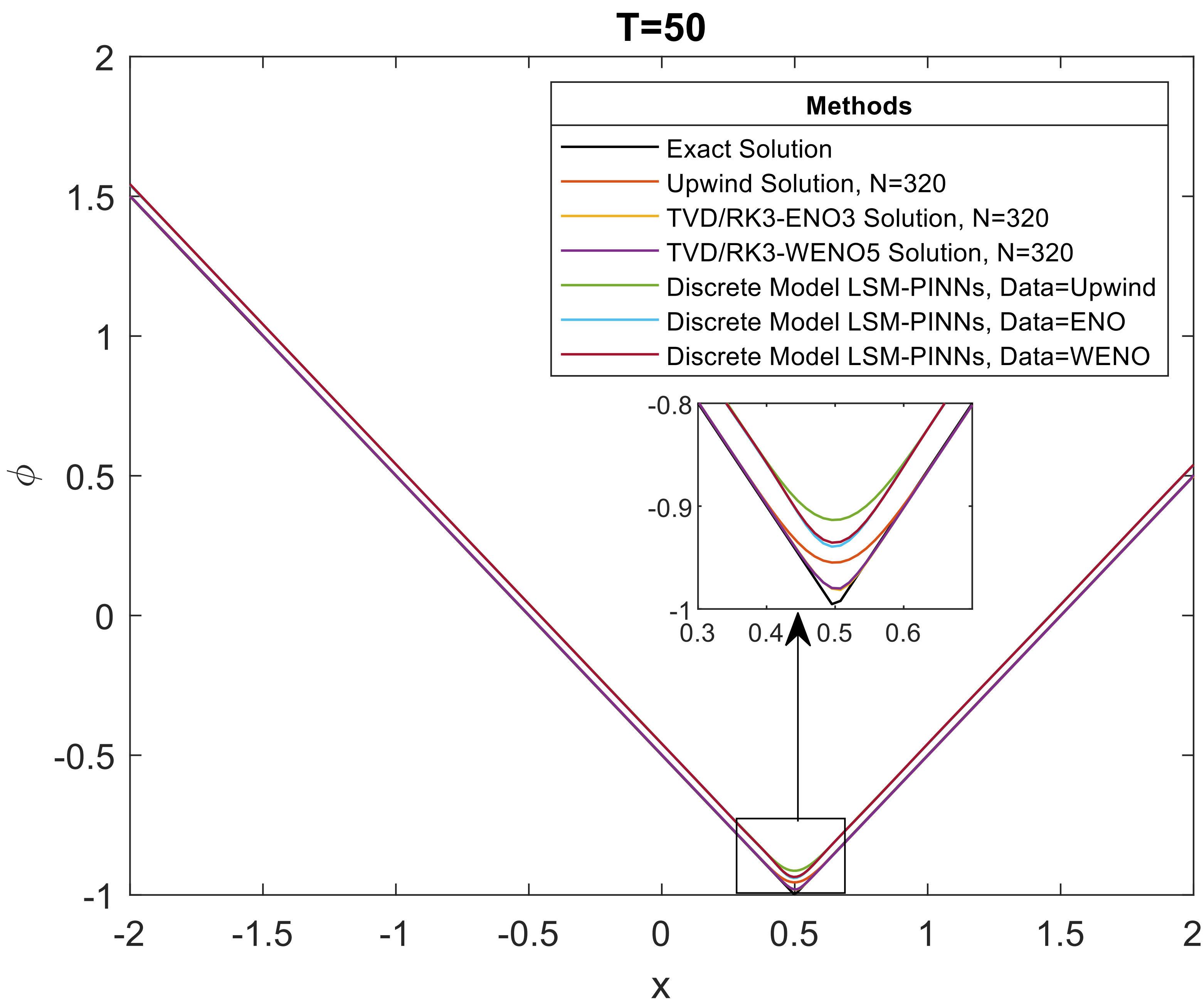


Figure 5.31: Numerical solutions generated by traditional methods; Upwind, Essentially non-oscillatory, and weighed essentially non-oscillatory, compared inferred solutions from discrete model LSM-PINNs at $T = 50$.

5.4.2 Continuous Model

Table 5.6 shows L_2 errors for measuring the accuracy of the continuous time model to approximate the exact solution; three scenarios were presented; first, we considered a neural network with 6-layers and 4 hidden layers, 100 neurons per hidden layer; to predict the solution at $T = 50$ using Upwind data, as well as the same setting to find a solution of the problem (5.1) when the NN is trained with TVD/RK3 ENO3 and TVD/RK3 WENO5 data. Also, the time of training was considered for each case.

Data	Layers	Hidden Layers	$N°$ N-L	N_ϕ	L_2 error	Time T.
Upwind	6	4	100	100	0.489	26.18 s
TVD/RK3-ENO3	6	4	100	100	3.03×10^{-2}	54.08 s
TVD/RK3-WENO5	6	4	100	100	6.67×10^{-2}	78.06 s

Table 5.6: Results of training with different data for continuous time model.

Figure 5.32 shows the exact solution, numerical solution computed by upwind schemes, and inferred solution by the continuous time model when NN was trained with upwind data. Similarly, Figures 5.33 and 5.34 shows the prediction of solution at $T = 50$ using TVD/RK3 ENO3 and TVD/RK3 WENO5 data, respectively.

Finally, Figure 5.36 illustrates all solutions, both numerical and inferred by continuous time model for comparing the solutions at point $(0.5, -1)$, and Figure 5.35 shows the L_1 errors for each prediction of solution in Table 5.6, the error L_1 for the approximated solution from TVD7RK3-WENO5 was considered to see the difference of errors.

All these training had the following parameters; the training set consists of a total of $N_d = 100$ data as well as $N_c = 10000$ randomly sampled collocation points used to enforce level set equation inside the solution domain; Latin Hypercube Sampling approach was used to generate all randomly sampled point locations, ensuring a space-filling distribution. Moreover, epochs = 10000, learning rate 0.001, and a 4-layer deep neural network with 100 neurons per layer and a hyperbolic tangent activation function.

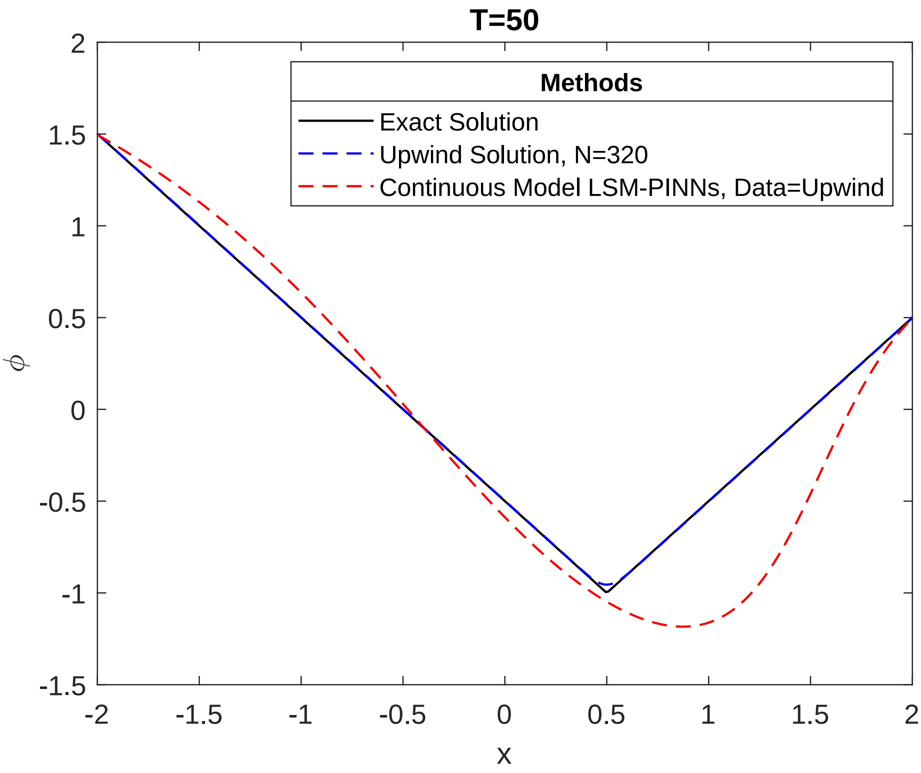


Figure 5.32: Inferred solution by the continuous model with training data generated by Upwind.

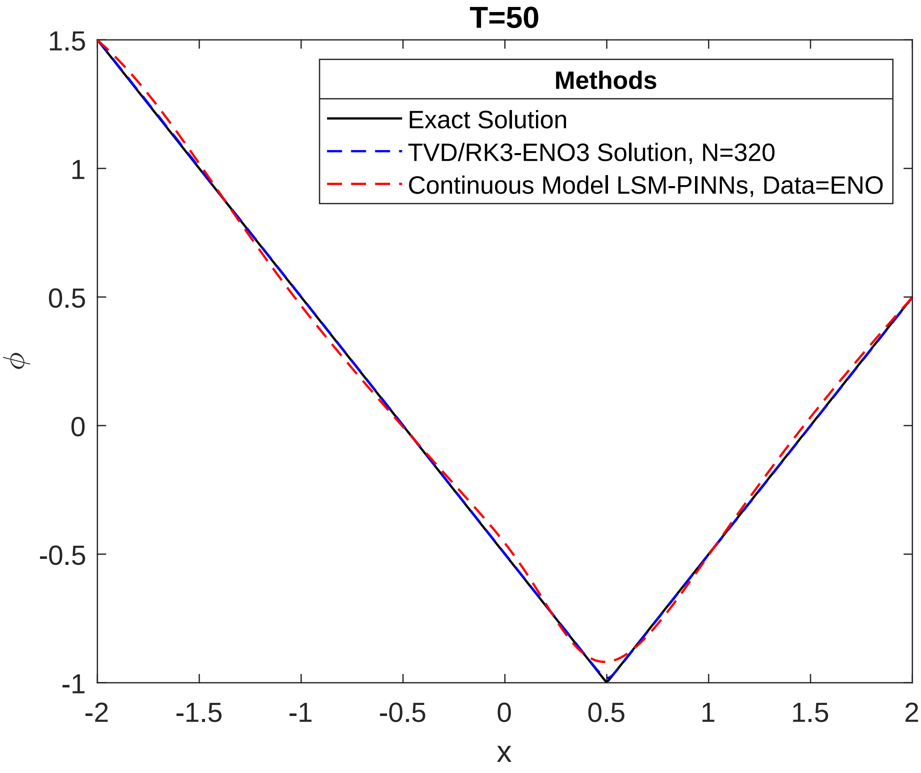


Figure 5.33: Inferred solution by the continuous model with training data generated by TVD/RK3-ENO3.

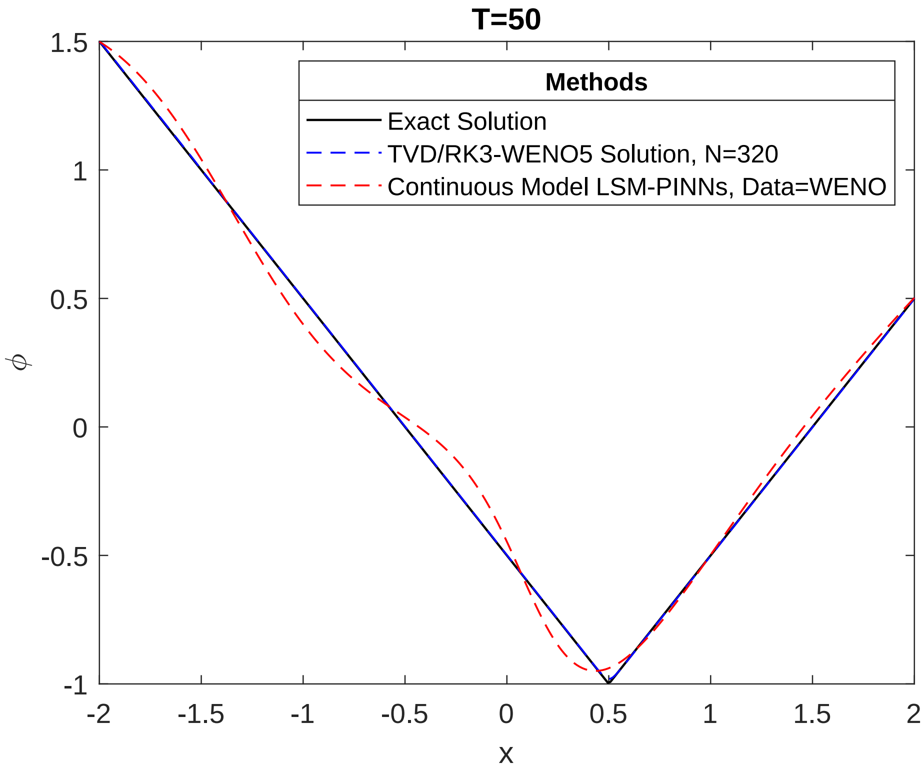


Figure 5.34: Inferred solution by the continuous model with training data generated by TVD/RK3-WENO5

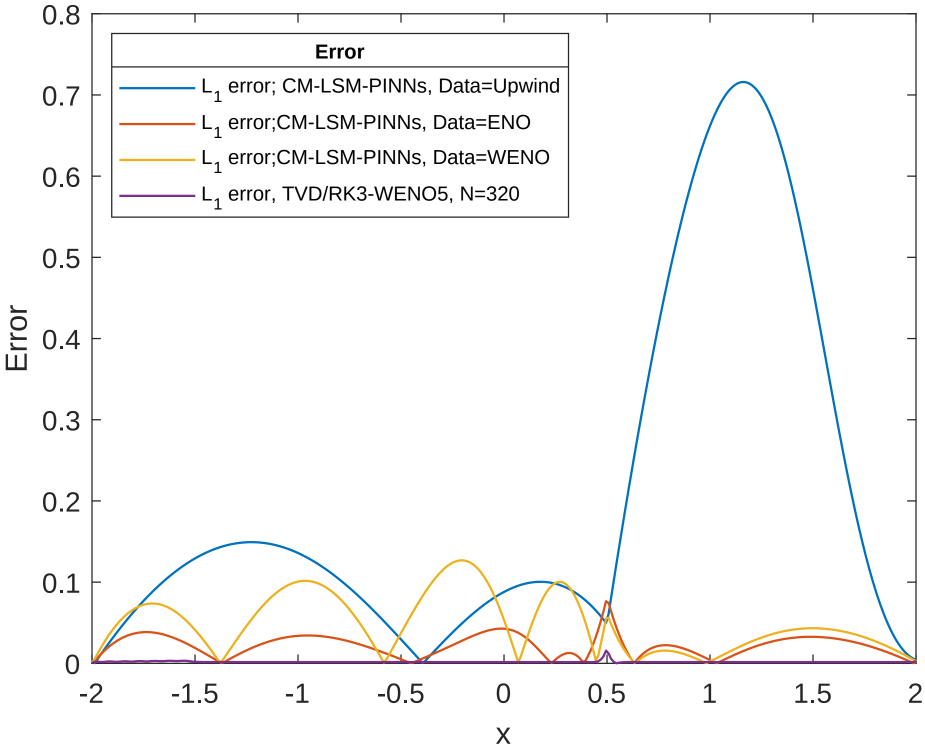


Figure 5.35: L_1 errors computed from exact solution and inferred solution generated by continuous model; and L_1 error of solution from TVD/RK3-WENO5 at $T = 50$.

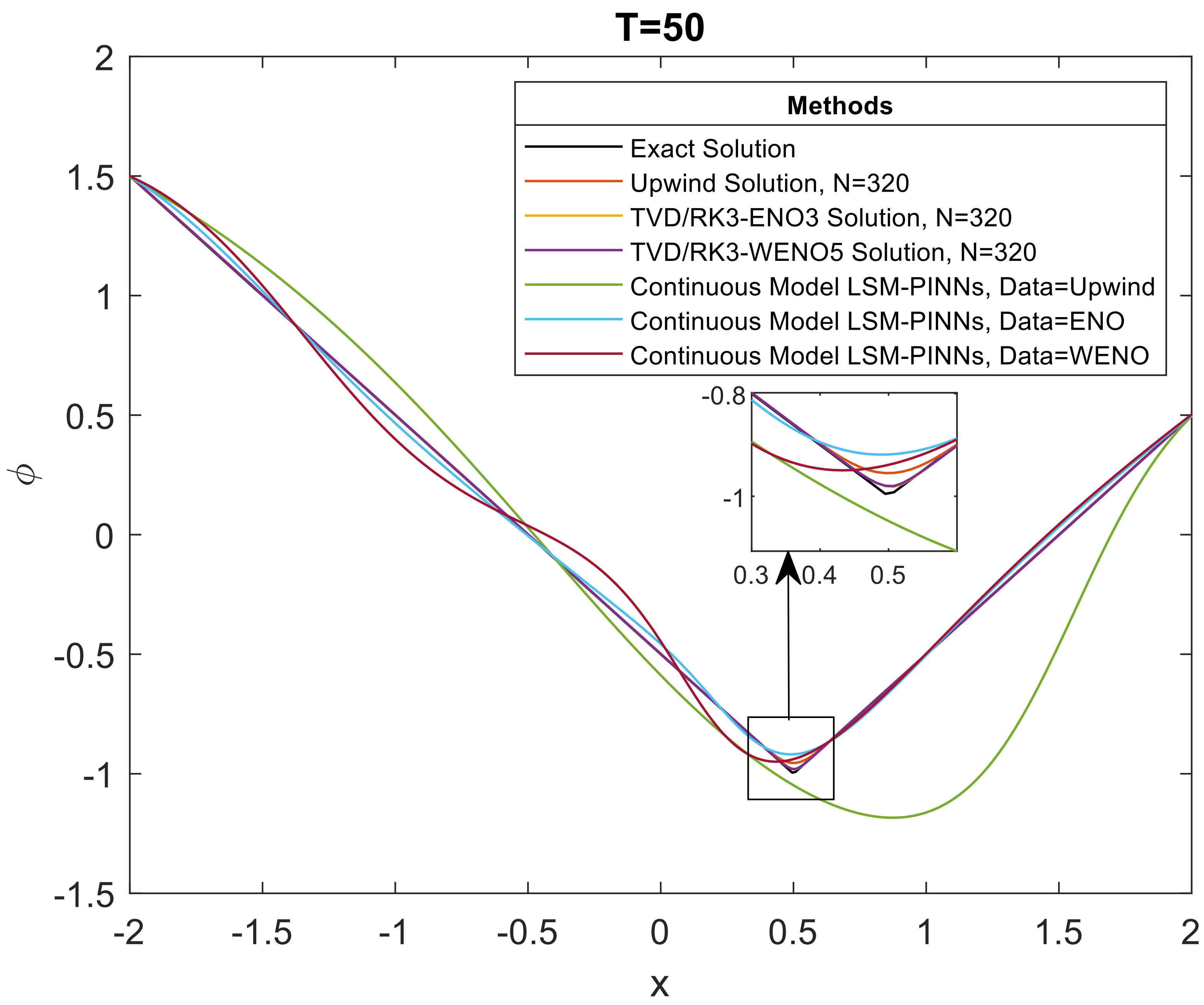


Figure 5.36: Numerical solutions generated by traditional methods; Upwind, Essentially non-oscillatory, and weighed essentially non-oscillatory, compared inferred solutions from continuous model LSM-PINNs at $T = 50$.

5.5 Discussion

In recent years, physics-informed neural networks (PINNs) are becoming a topic of exhaustive study by several researchers in the field of numerical methods. PINNs emerged as a new paradigm to solve partial differential equations that model physical phenomena; for example, the Navier Stokes equations describe the motion of viscous fluid [27]. The level set method is a formulation that captures interfaces or fronts by representing it as a zero level of high dimensional function; typically, it is a signed distance function. This involves solving a Hamilton-Jacobi formulation, mainly a PDE called level set equation or advection equation that describes the motion of the interface in an externally generated vector field. In this context, two and three dimensions, the main issues are the preserving geometry of the interface; it is related to a directly signed distance function, as it evolves over time and computational cost [7, 8]. One dimensional case is studied to understand the behavior of numerical solutions generated by both finite different methods and discrete/continuous models of PINNs without boundary condition defined in loss function (Fig.**??**, 5.36) under certain parameters on CFL conditions and training, respectively. For the measurement of the quality to approximate a signed distance function, we used L_1, L_2 and L_∞ errors to compare with the exact solution at $T = 50$.

The results of the study of the numerical solutions from finite difference methods to generate relevant data to train the LSM-Physics informed neural networks present the following situations; Tables 5.1, 5.5, and 5.3 showed that one of the aspects for obtaining a good approximation depends on the size of the grid; in fact, the error of approximation for an exact solution using the upwind method with $N = 320$ is $L_1 = 9 \times 10^{-3}$ compared to TVD/RK3-ENO3 and TVD/RK3-WENO5 that produced $L_1 = 7 \times 10^{-3}$ and $L_1 = 6.9 \times 10^{-3}$, respectively (Fig.5.23). In the same way, we can see that the order of numerical methods is significantly notable if the size of the grid is small, particularly the $L_1 = 0.134$ error for the upwind method with $N = 20$ reduced to $L_1 = 0.127$ and $L_1 = 0.123$ for ENO3 and WENO3, respectively. Therefore, Figure (5.22) illustrates the solutions at $T = 50$ go to near exact solution as the order of the methods increases. However, Figures 5.14 and 5.20 emerged small spurious oscillations identified by L_1 error; it is caused by the sensitivity of parameters in ENO and WENO schemes. In the case of upwind schemes, the solution at $T = 50$ with $N = 320$ does not satisfy the Eikonal equation; then it is a not signed distance function; in contrast, the ENO and WENO schemes have a significant approximation to be SDF.

To train the discrete model of LSM-PINNs, we considered relevant data generated by each numerical method; Upwind, ENO, and WENO schemes with the size of grid $N = 320$; which had good approximations to exact solution; in this context, the training of model was led by standard numbers of hidden layer proposed in [78] for a non-complex problem. This main observation on results is the quality of data does not present significant changes for improving the inference of solutions. Table 5.4 shows L_2 errors, which represents a deviation in the prediction of solution at $T = 50$, regardless of whether the data comes from one of the three methods used (Fig. 5.25, 5.25, 5.25). In a recent study, Shashank et al. [64] obtained good results in the same problem by setting a smooth initial condition and Dirichlet boundary conditions. Furthermore, we showed some predicted solutions based on increasing the layers and hidden layers; the results show that L_2 error also increases. $L_2 = 0.217$ increases to $L_2 = 0.425$ with Upwind data; the same patterns happen for other cases (Table. 5.30 and Fig. 5.28, 5.29, 5.30).

On the other hand, the continuous model of LSM-PINNs demonstrated a significant difference between feeding to a neuronal network with Upwind data and TVD/RK3-ENO3 data. Table 5.6shows that $L_2 = 0.489$ error corresponds to the inferred solution using some upwind data; in contrast, $L_2 = 3.03 \times 10^{-2}$ error obtained in feeding the NN with ENO3 data. Consequently, the quality of data affects in a certain way improves the inferred solution by the continuous model. However, the L_2 error of the inferred solution by training NN with some WENO data is relatively large compared to the error of continuous model training with ENO data. These results are expected due to the lack of information on the boundary conditions. Figure 5.36 shows how efficiency is continuous model versus the numerical solutions; these results show that finite difference methods are more suitable to compute the solution at $T = 50$ (Fig.5.27, 5.35).

Computational cost in numerical methods is directly proportional to the size of the grid; in the same way, the time of training increases due to the data quality. However, this project does not show if PINNs reduce the computational cost to find an approximated solution to the problem (4.1), but we point out that once the training parameters are appropriate for the situation (4.1), we can save these parameters to have a predefined neural network, and thus find the approximate solution at any instant of time without having to train the neural network. Shashank et al. [64] mentioned that PINNs reduce computational costs.

Chapter 6

Conclusions

The level set method is a powerful tool to capture interfaces by using signed distance functions; numerical methods, such as finite differences, help find solutions for the level set equation. In this context, the challenge remains to find a method that generates good approximations and low computational cost; consequently, this will determine a good capturing interface to solve the preserving problem in LSM. The implementations of upwind, essentially non-oscillatory, and weighted essentially non-oscillatory schemes coupled with total variation diminishing Runge Kutta schemes under the CFL conditions to keep simulation stability were successful to solve the one dimensional problem (4.1). Tables 5.1, 5.5 and 5.3 showed that as the order of the differences is increased the L_∞ and L_1 decrease even if the grid size is small. Therefore the results showed good approximations for exact solution; consequently it represents a suitable data to train the neural networks.

On the other hand, the inferred solutions by continuous time and discrete time models had a tendency to approximate the exact solution at $T = 50$, however Table 5.4 and 5.6 showed that L_2 error compared with L_2 error of numerical solution of finite different methods are significantly large. For the discrete time model the shape of initial function was kept at $T = 50$ (Fig. 5.26), in contrast, the continuous model showed oscillations (Fig.5.34) and it produced the large error. Moreover, the quality of data for training did not directly affect the accuracy of the solutions. Therefore, the inaccuracy of prediction was due to define the loss function without the mean square error in boundary conditions.

6.1 Recomendations

Consider other measurements mentioned in [27] to evaluate the quality of solutions generated by models. Moreover, one can study convergence for the models to create suitable training parameters and improve results. However, the necessary information developed in this project allows us to identify possible approaches to improve the neural network and apply it in two-dimensional cases. The following are possible improvements:

(a) Externally generated velocity field is defined in all domain; then, we can define the loss function with inflow and outflow boundary conditions. Therefore, it allows leading experiments for the two-dimensional case.

(b) To maintain a signed distance function as it evolves over time, we can add an extra term in the loss function defined by the Eikonal equation. This allows for preserving the interface's geometry involving severe interface testing.

6.2 Future Works

These preliminaries results allowed to find possible solutions, the first step, we will implement these features en loss functions, and the second step is to create a functional neural network for the two-dimensional case of the level set method for developing numerical testing to evaluate PINNs and find applications in computational fluid dynamics.

6.3 Limitations

This study is limited to a one-dimensional level set method with a constant velocity field, but what happens when the velocity field depends on time? It is possible to study with physics-informed neural networks the setting of a suitable loss function will play an essential role in investigating this case.

All these algorithms are robust where we used an old version of TensorFlow 1.14; new versions help to reformulate these approaches. Nowadays, there is a simple way to code PINNs using PyTorch [64] and solve the same problem. Moreover, new optimization methods to minimize the loss function, such as stochastic gradient descent, Wasserstein GANs with gradient penalty, and so on [27], must be considered.

On the other hand, we used standard layers described in [78] for feed-forward neural networks. However, the approaches described above need a deep study to find several layers in which each hidden layer contains an optimal number of neurons for this problem. Moreover, this project helps to understand the processing and structure of data on the training of PINNs; it is a crucial aspect that will allow developing ideas for setting data on two dimensions to use the neural network built for the one-dimensional case without the need to code another PINNs for the two-dimensional case of LSM.

Bibliography

[1] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations," *Journal of Computational Physics*, vol. 79, pp. 12–49, 11 1988.

[2] V. F. Maschinenbau, "Level set method for simulating the dynamics of the fluid-fluid interfaces: Application of a discontinuous galerkin method." [Online]. Available: http://tuprints.ulb.tu-darmstadt.de/3872/

[3] F. Henri, M. Coquerelle, and P. Lubin, "An efficient hybrid advection scheme in a level set framework coupling weno5 and houc5 schemes based on kink detection." [Online]. Available: https://hal.science/hal-03280902

[4] Y. Song, Q. Ma, Y. He, M. Zhou, and M. Y. Wang, "Stress-based shape and topology optimization with cellular level set in b-splines," *Structural and Multidisciplinary Optimization*, vol. 62, pp. 2391–2407, 11 2020. [Online]. Available: https://www.researchgate.net/publication/342990705_Stress-based_shape_and_topology_optimization_with_cellular_level_set_in_B-splines

[5] J. W. Thomas, "Numerical partial differential equations: Finite difference methods," vol. 22, 1995. [Online]. Available: http://link.springer.com/10.1007/978-1-4899-7278-1

[6] S. Zhang and C. W. Shu, "A new smoothness indicator for the weno schemes and its effect on the convergence to steady state solutions," *Journal of Scientific Computing*, vol. 31, pp. 273–305, 6 2007. [Online]. Available: https://link.springer.com/article/10.1007/s10915-006-9111-y

[7] I. Pineda, D. Arellano, and R. Chachalo, “Analysis of essentially non-oscillatory numerical techniques for the computation of the level set method,” *Communications in Computer and Information Science*, vol. 1193 CCIS, pp. 395–408, 2020. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-030-42517-3_30

[8] J. Luo, L. Xuan, and K. Xu, “Comparison of fifth-order weno scheme and finite volume weno-gas-kinetic scheme for inviscid and viscous flow simulation,” *Commun. Comput. Phys*, vol. 14, pp. 599–620, 2013. [Online]. Available: http://www.global-sci.com/599

[9] G. S. Jiang and D. Peng, “Weighted eno schemes for hamilton–jacobi equations,” *https://doi.org/10.1137/S106482759732455X*, vol. 21, pp. 2126–2143, 7 2006. [Online]. Available: https://epubs.siam.org/doi/10.1137/S106482759732455X

[10] B. H. Hahn and D. T. Valentine, “Introduction to numerical methods,” *Essential Matlab for Engineers and Scientists*, pp. 301–328, 2013.

[11] B. J. Lewis, E. N. Onder, and A. A. Prudil, “Partial differential equations,” *Advanced Mathematics for Engineering Students*, pp. 131–164, 2022. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780128236819000137

[12] J. Kirkwood, “Three important equations,” *Mathematical Physics with Partial Differential Equations*, pp. 217–236, 2018.

[13] S. Alinhac, “Hyperbolic partial differential equations,” *Hyperbolic Partial Differential Equations*, 2009.

[14] P. Knabner and L. Angermann, “Numerical methods for elliptic and parabolic partial differential equations,” vol. 44, 2021. [Online]. Available: https://link.springer.com/10.1007/978-3-030-79385-2

[15] J. Trangenstein, “Numerical solution of hyperbolic conservation laws,” p. 620, 2009.

[16] K. Luo, C. Shao, M. Chai, and J. Fan, “Level set method for atomization and evaporation simulations,” *Progress in Energy and Combustion Science*, vol. 73, pp. 65–94, 7 2019.

[17] J. Tu, G. H. Yeoh, and C. Liu, *Computational fluid dynamics: A practical approach*. Elsevier, 1 2018. [Online]. Available: http://www.sciencedirect.com:5070/book/9780081011270/computational-fluid-dynamics

[18] S. Osher and R. Fedkiw, "Level set methods and dynamic implicit surfaces," vol. 153, 2003. [Online]. Available: http://link.springer.com/10.1007/b98879

[19] I. Pineda and O. Gwun, "Leaf modeling and growth process simulation using the level set method," *IEEE Access*, vol. 5, pp. 15 948–15 959, 8 2017.

[20] V. Doyeux, Y. Guyot, V. Chabannes, C. Prud'homme, C. Prud, V. Doyeux, Y. Guyot, V. Chabannes, C. Prud'homme, and M. Ismail, "Simulation of two phase flow using a level set method: Application to bubbles and vesicle dynamics simulation of two phase flow using a level set method application to bubbles and vesicle dynamics," pp. 14–17, 2011. [Online]. Available: https://hal.science/hal-00657152

[21] G. S. Jiang and C. W. Shu, "Efficient implementation of weighted eno schemes," *Journal of Computational Physics*, vol. 126, pp. 202–228, 6 1996.

[22] vorgelegt von, "The level set method for capturing interfaces with applications in two-phase flow problems."

[23] N. Balcázar, L. Jofre, O. Lehmkuhl, J. Castro, and J. Rigola, "A finite-volume/level-set method for simulating two-phase flows on unstructured grids," *International Journal of Multiphase Flow*, vol. 64, pp. 55–72, 9 2014.

[24] C. W. Shu and S. Osher, "Efficient implementation of essentially non-oscillatory shock-capturing schemes," *Journal of Computational Physics*, vol. 77, pp. 439–471, 8 1988.

[25] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2 2019.

[26] M. W. Dissanayake and N. Phan-Thien, "Neural-network-based approximations for solving partial differential equations," *Communications in*

Numerical Methods in Engineering, vol. 10, pp. 195–201, 3 1994. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1002/cnm.1640100303https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303https://onlinelibrary.wiley.com/doi/10.1002/cnm.1640100303

[27] S. Cuomo, V. S. D. Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli, "Scientific machine learning through physics-informed neural networks: Where we are and what's next," *Journal of Scientific Computing*, vol. 92, 1 2022. [Online]. Available: https://arxiv.org/abs/2201.05624v4

[28] T. Kossaczká, M. Ehrhardt, and M. Günther, "Enhanced fifth order weno shock-capturing schemes with deep learning," *Results in Applied Mathematics*, vol. 12, p. 100201, 11 2021.

[29] E. L. Strelow, A. Gerisch, J. Lang, and M. E. Pfetsch, "Physics informed neural networks: A case study for gas transport problems," *Journal of Computational Physics*, vol. 481, p. 112041, 5 2023.

[30] R. Qiu, R. Huang, Y. Xiao, J. Wang, Z. Zhang, J. Yue, Z. Zeng, and Y. Wang, "Physics-informed neural networks for phase-field method in two-phase flow," *Physics of Fluids*, vol. 34, p. 52109, 5 2022. [Online]. Available: https://pubs.aip.org/aip/pof/article/34/5/052109/2846695/Physics-informed-neural-networks-for-phase-field

[31] R. Li, E. Lee, and T. Luo, "Physics-informed neural networks for solving multiscale mode-resolved phonon boltzmann transport equation," *Materials Today Physics*, vol. 19, p. 100429, 7 2021.

[32] A. B. Buhendwa, S. Adami, and N. A. Adams, "Inferring incompressible two-phase flow fields from the interface motion using physics-informed neural networks," *Machine Learning with Applications*, vol. 4, p. 100029, 6 2021.

[33] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts." *Proceedings of the National Academy of Sciences*, vol. 93, pp. 1591–1595, 2 1996. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.93.4.1591

[34] A. Nouri-Borujerdi and A. Kebriaee, "Upwind compact implicit and explicit high-order finite difference schemes for level set technique," *International Journal for Computational Methods in Engineering Science and Mechanics*, vol. 13, pp. 308–318, 08 2012.

[35] C. W. Shu, "High-order finite difference and finite volume weno schemes and discontinuous galerkin methods for cfd," *International Journal of Computational Fluid Dynamics*, vol. 17, pp. 107–118, 4 2003.

[36] R. Borges, M. Carmona, B. Costa, and W. S. Don, "An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws," *Journal of Computational Physics*, vol. 227, no. 6, pp. 3191–3211, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021999107005232

[37] B. Houchmandzadeh, "The hamilton-jacobi equation : an intuitive approach."

[38] P. R. Wolenski, "Hamilton–jacobi theory for hereditary control problems," *Nonlinear Analysis: Theory, Methods and Applications*, vol. 22, pp. 875–894, 4 1994.

[39] N. D. Katopodes, "Level set method," *Free-Surface Flow*, pp. 804–828, 1 2019.

[40] S. Dong, "Finite difference methods for the hyperbolic wave partial differential equations." [Online]. Available: http://mit.edu/dongs/Public/18.086/Project1.

[41] A. Harten, S. Osher, B. Engquist, and S. R. Chakravarthy, "Some results on uniformly high-order accurate essentially nonoscillatory schemes," *Applied Numerical Mathematics*, vol. 2, pp. 347–377, 10 1986.

[42] C.-W. Shu, "Essentially non-oscillatory and weighted essentially non-oscillatory schemes," *Acta Numerica*, vol. 29, p. 701–762, 2020.

[43] X. D. Liu, "Weighted essentially non-oscillatory schemes," *Journal of Computational Physics*, vol. 115, pp. 200–212, 11 1994.

[44] S. Gottlieb and C.-W. Shu, "Total variation diminishing runge-kutta schemes," *MATHEMATICS OF COMPUTATION*, vol. 67, pp. 913–915, 1998.

[45] A. L. Caterini and D. E. Chang, "Generic representation of neural networks," *SpringerBriefs in Computer Science*, vol. 0, pp. 23–34, 2018.

[46] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," pp. 267–285, 1982. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-46466-9_18

[47] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. V. Essen, A. A. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics (Switzerland)*, vol. 8, 3 2019.

[48] L. Sun, H. Gao, S. Pan, and J. X. Wang, "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data," *Computer Methods in Applied Mechanics and Engineering*, vol. 361, p. 112732, 4 2020.

[49] S. Goswami, C. Anitescu, S. Chakraborty, and T. Rabczuk, "Transfer learning enhanced physics informed neural network for phase-field modeling of fracture," *Theoretical and Applied Fracture Mechanics*, vol. 106, p. 102447, 4 2020.

[50] Q. He and A. M. Tartakovsky, "Physics-informed neural network method for forward and backward advection-dispersion equations," *Water Resources Research*, vol. 57, no. 7, p. e2020WR029479, 2021, e2020WR029479 2020WR029479. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2020WR029479

[51] P. Stiller, F. Bethke, M. Böhme, R. Pausch, S. Torge, A. Debus, J. Vorberger, M. Bussmann, and N. Hoffmann, "Large-scale neural solvers for partial differential equations," *Communications in Computer and Information Science*, vol. 1315 CCIS, pp. 20–34, 9 2020. [Online]. Available: https://arxiv.org/abs/2009.03730v1

[52] A. Mathews, M. Francisquez, J. W. Hughes, D. R. Hatch, B. Zhu, and B. N. Rogers, "Uncovering turbulent plasma dynamics via deep learning from partial observations," *Physical review. E*, vol. 104, 8 2021. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/34525532/

[53] S. Kollmannsberger, D. D'Angella, M. Jokeit, and L. Herrmann, "Physics-informed neural networks," *Deep Learning in Computational Mechanics*, vol. 977, pp. 55–84, 2021.

[54] T. D. Ryck, S. Lanthaler, and S. Mishra, "On the approximation of functions by tanh neural networks," *Neural Networks*, vol. 143, pp. 732–750, 11 2021.

[55] Z. H. Gu, H. L. Wen, C. H. Yu, and T. W. H. Sheu, "Interface-preserving level set method for simulating dam-break flows," *Journal of Computational Physics*, vol. 374, pp. 249–280, 2018. [Online]. Available: www.elsevier.com/locate/jcp

[56] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell, "A hybrid particle level set method for improved interface capturing," *Journal of Computational Physics*, vol. 183, pp. 83–116, 11 2002.

[57] M. Sussman, P. Smereka, and S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *Journal of Computational Physics; (United States)*, vol. 114:1, pp. 146–159, 9 1994.

[58] D. Hartmann, M. Meinke, and W. Schröder, "The constrained reinitialization equation for level set methods," 2009.

[59] D. lindbo, "Finite element computations for a conservative level set method applied to two-phase stokes flow," 2006. [Online]. Available: www.csc.kth.se

[60] L. C. Ngo and H. G. Choi, "A multi-level adaptive mesh refinement for an integrated finite element/level set formulation to simulate multiphase flows with surface tension," *Computers and Mathematics with Applications*, vol. 79, no. 4, pp. 908–933, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0898122119304043

[61] R. Li and W. Zhong, "A robust and efficient component-wise weno scheme for euler equations," *Applied Mathematics and Computation*, vol. 438, p. 127583, 2 2023.

[62] S. Li, M. Penwarden, R. M. Kirby, and S. Zhe, "Meta learning of interface conditions for multi-domain physics-informed neural networks."

[63] Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 360, p. 112789, 3 2020.

[64] S. R. Vadyala, S. N. Betgeri, and N. P. Betgeri, "Physics-informed neural network method for solving one-dimensional advection equation using pytorch," *Array*, vol. 13, p. 100110, 3 2022.

[65] V. Liu and H. Yoon, "Prediction of advection and diffusion transport using physics informed neural networks." 11 2020.

[66] C. Wang, E. Bentivegna, W. Zhou, L. J. Klein, and B. Elmegreen, "Physics-informed neural network super resolution for advection-diffusion models," *CoRR*, vol. abs/2011.02519, 2020. [Online]. Available: https://arxiv.org/abs/2011.02519

[67] S. Osher and F. Solomon, "Upwind difference schemes for hyperbolic systems of conservation laws," vol. 38, pp. 339–374, 1982.

[68] R. Wang and R. J. Spiteri, "Linear instability of the fifth-order weno method *," *ANAL. c 0000 Society for Industrial and Applied Mathematics*, vol. 0, pp. 0–000. [Online]. Available: http://www.siam.org/journals/sinum/x-x/63786.html

[69] M. P. Martín, E. M. Taylor, M. Wu, and V. G. Weirs, "A bandwidth-optimized weno scheme for the effective direct numerical simulation of compressible turbulence," 2006. [Online]. Available: www.elsevier.com/locate/jcp

[70] A. K. Henrick, T. D. Aslam, and J. M. Powers, "Mapped weighted essentially non-oscillatory schemes: Achieving optimal order near critical points," *Journal of Computational Physics*, vol. 207, pp. 542–567, 8 2005.

[71] C. Hu and C. W. Shu, "Weighted essentially non-oscillatory schemes on triangular meshes," *Journal of Computational Physics*, vol. 150, pp. 97–127, 3 1999.

[72] F. Teng, L. Yuan, and T. Tang, "A speed-up strategy for finite volume weno schemes for hyperbolic conservation laws," *Journal of Scientific Computing 2010 46:3*, vol. 46,

pp. 359–378, 7 2010. [Online]. Available: https://link.springer.com/article/10.1007/s10915-010-9407-9

[73] S. Kurioka and D. R. Dowling, "Numerical simulation of free surface flows with the level set method using an extremely high-order accuracy weno advection scheme," *https://doi.org/10.1080/10618560902776786*, vol. 23, pp. 233–243, 4 2009. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/10618560902776786

[74] C. Bahbah, M. Khalloufi, A. Larcher, Y. Mesri, T. Coupez, R. Valette, and E. Hachem, "Conservative and adaptive level-set method for the simulation of two-fluid flows." [Online]. Available: https://hal.science/hal-02154945

[75] F. Gibou, R. Fedkiw, and S. Osher, "A review of level-set methods and some recent applications," 2017. [Online]. Available: http://www.elsevier.com/open-access/userlicense/1.0/

[76] R. J. LeVeque, "Finite volume methods for hyperbolic problems," *Finite Volume Methods for Hyperbolic Problems*, 8 2002. [Online]. Available: https://www.cambridge.org/core/books/finite-volume-methods-for-hyperbolic-problems/97D5D1ACB1926DA1D4D52EAD6909E2B9

[77] A. Iserles, "A first course in the numerical analysis of differential equations, second edition," *A First Course in the Numerical Analysis of Differential Equations, Second Edition*, pp. 1–459, 1 2008. [Online]. Available: https://www.cambridge.org/core/books/first-course-in-the-numerical-analysis-of-differential-equations/2B4E05F5CFC58CFDC7BBBC6D1150661B

[78] A. M. Tartakovsky, C. O. Marrero, P. Perdikaris, G. D. Tartakovsky, and D. Barajas-Solano, "Physics-informed deep neural networks for learning parameters and constitutive relationships in subsurface flow problems," *Water Resources Research*, vol. 56, p. e2019WR026731, 5 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/full/10.1029/2019WR026731https://onlinelibrary.wiley.com/doi/abs/10.1029/2019WR026731https://agupubs.onlinelibrary.wiley.com/doi/10.1029/2019WR026731