A Numerical Algorithm Based on PINN for Simulating the Burger's Equation

A Dissertation

Submitted in partial fulfillment of the requirements for the
Degree of

Integrated Master of Science
in
Applied Mathematics

Submitted by
Vikram

Under the guidance of
Dr. Ram Jiwari

Department of Mathematics
Indian Institute of Technology Roorkee
Roorkee - 247667 (INDIA)

May, 2025

INDIAN INSTITUTE OF TECHNOLOGY
ROORKEE, ROORKEE

Candidate's Declaration

I hereby certify that the work which is being presented in this report entitled "A Numerical Algorithm Based on PINN for Simulating the Burger's Equation" in partial fulfillment of the requirements for the award of the degree of Integrated Master of Science and submitted in the Department of Mathematics of the Indian Institute of Technology Roorkee is an authentic record of my own work carried out during the period from January, 2025 to May, 2025 under the supervision of Dr. Ram Jiwari.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institute.

Vikram
20312040
Integrated Master of Science in Applied Mathematics
Department of Mathematics
IIT Roorkee

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Dr. Ram Jiwari
Department of Mathematics
IIT Roorkee

Date:

Abstract

This work presents a numerical approach based on Physics-Informed Neural Networks (PINNs) for solving the Burgers' equation, a fundamental nonlinear partial differential equation that models various transport phenomena. By embedding the physical laws directly into the loss function, the proposed method effectively integrates data-driven learning with the governing dynamics of the system. The network is trained to minimize a composite loss combining the residuals of the Burgers' equation at collocation points and the deviations from boundary and initial conditions. Through extensive numerical experiments, it is demonstrated that the PINN framework accurately captures the behavior of the solution, including the sharp gradients associated with shock formation. Comparative analyses between the predicted and exact solutions, along with error metrics, validate the effectiveness and robustness of the method. The results highlight the potential of PINNs as a flexible and powerful tool for solving complex partial differential equations without requiring discretization of the spatial or temporal domains.

Acknowledgements

I express my deepest gratitude to my supervisor Dr Ram Jiwari, Department of Mathematics, Indian Institute of Technology Roorkee for giving me the opportunity to work under his guidance. It is only due to his constant support, valuable advice and encouragement throughout this period which made this work possible.

Date: Vikram

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Partial Differential Equations (PDEs) play a fundamental role in modeling a wide range of physical phenomena, from fluid dynamics and heat conduction to wave propagation and material deformation. Among them, the Burgers' equation stands out as a canonical example, often serving as a simplified model for more complex systems such as turbulence and shock wave formation. Due to its nonlinear nature, Burgers' equation captures essential features of real-world dynamics while remaining mathematically tractable. Developing accurate and efficient numerical methods for solving Burgers' equation is crucial, not only for theoretical studies but also for practical engineering and scientific applications.

1.2 Challenges with Traditional Numerical Methods

Traditional numerical methods, such as the Finite Difference Method (FDM) and the Finite Element Method (FEM), have long been employed to solve PDEs. While these techniques are well-established and widely used, they encounter several challenges, especially when dealing with nonlinearities, high-dimensional problems, and complex boundary conditions. Finite difference schemes often require fine discretization grids to maintain stability and accuracy, leading to a substantial increase in computational cost. Similarly, finite element formulations, though flexible in handling irregular domains, can become computationally intensive when extended to highly nonlinear or time-dependent problems. Moreover, the accuracy of these methods heavily depends on mesh quality and refinement strategies, which can be difficult to optimize in practice.

1.3 Introduction to Neural Networks and Physics-Informed Neural Networks

In recent years, machine learning, particularly deep learning, has emerged as a powerful tool for approximating complex functions and solving high-dimensional problems. Neural networks, with their ability to learn intricate patterns from data, have opened new avenues in scientific computing. Physics-Informed Neural Networks (PINNs) represent a novel paradigm that integrates physical laws, expressed as PDEs, directly into the training of neural networks. Rather than relying solely on data, PINNs incorporate the governing equations into the loss function, ensuring that the learned solution respects the underlying physics. This approach not only circumvents the need for large datasets but also offers mesh-free computation, making it particularly attractive for solving PDEs with complex geometries or dynamics.

1.4 Objectives of the Thesis

This thesis aims to design a robust numerical algorithm, grounded in the framework of Physics-Informed Neural Networks , to effectively simulate the behavior of the Burgers' equation. The work aims to explore the capability of PINNs in accurately capturing the solution behavior, including nonlinear wave propagation and shock formation, inherent in the Burgers' dynamics. Specific goals include:

- Formulating the Burgers' equation in a manner suitable for PINN-based learning.
- Designing and implementing a neural network architecture that effectively models the solution.
- Evaluating the performance of the proposed method through comparison with analytical or high-fidelity numerical solutions.
- Analyzing the impact of various training parameters and strategies on the accuracy and efficiency of the solution.

Through this study, we aim to contribute more towards the research on machine learning approaches for solving PDEs and to demonstrate the potential of PINNs as a viable alternative to traditional numerical methods.

Chapter 2

Background and Literature Review

2.1 Mathematical Background

2.1.1 Partial Differential Equations

Partial Differential Equations are mathematical expressions that contain multivariable functions and their partial derivatives. They are fundamental to the modeling of physical systems, describing phenomena such as fluid flow, heat transfer, electromagnetic fields, and quantum mechanics [2]. A general form of a PDE can be written as:

$$F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial y^2}, \dots\right) = 0, \tag{2.1}$$

where u is the unknown function of the independent variables x and y, and F denotes a given function involving u and its derivatives. Solving a PDE typically requires specifying appropriate initial and boundary conditions.

2.1.2 The Burgers' Equation

the Non linear fundamental Burger equation is given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{2.2}$$

where $u(x, t)$ represents the unknown function, t is the time variable, x is the spatial variable, and ν denotes the viscosity coefficient. The Burgers' equation serves as a simplified model for turbulence and shock wave formation [8]. It combines nonlinear convection and linear diffusion, making it is best testbed for eval and test the performance of numerical methods in capturing nonlinear dynamics and sharp gradients.

2.2 Traditional Numerical Methods for PDEs

Classical numerical methods, such as the Finite Difference Method (FDM), Finite Element Method (FEM), and Finite Volume Method (FVM), have been extensively used to approximate solutions to PDEs [4]. The FDM discretizes the domain into a grid and replaces derivatives with finite differences. While straightforward to implement, it struggles with complex geometries and requires fine grids to resolve steep gradients. where FEM , splits the domain into elements and approximates the solution using basis functions. It provides greater flexibility in handling irregular geometries but increases computational complexity. FVM focuses on conserving quantities over control volumes and is particularly suited for conservation laws. Despite their successes, traditional methods often suffer from scalability issues in high dimensions and require careful mesh design and refinement strategies.

2.3 Basics of Machine Learning and Deep Learning

As a branch of artificial intelligence, machine learning focuses on creating algorithms that let computers recognize patterns in data without explicit programming. A subfield of machine learning called "deep learning" uses multi-layered neural networks to

model intricate connections [3]. Interconnected layers of neurons make up neural networks, which transform incoming data nonlinearly. Networks can learn to approximate functions or forecast results by changing the weights of connections during training. In fields including image identification, natural language processing, and, more recently, differential equation solving, deep learning has shown impressive results.

2.4 Introduction to Physics-Informed Neural Networks

2.4.1 What are PINNs?

Physics-Informed Neural Networks represent a specialized class of neural networks that integrate physical laws—typically formulated as partial differential equations (PDEs)—directly into the learning process. Rather than depending exclusively on observational data, PINNs are trained by minimizing a composite loss function that accounts for the residuals of the governing PDEs along with initial and boundary conditions. By embedding these physical constraints into the optimization framework, PINNs are able to learn solutions that inherently respect the structure and behavior of the underlying physical systems.

2.4.2 Why are PINNs Useful?

PINNs offer several advantages over traditional numerical methods. They are inherently mesh-free, eliminating the need for complex grid generation. Their formulation allows for the seamless handling of high-dimensional problems, where traditional methods struggle due to the "curse of dimensionality." Moreover, PINNs can incorporate sparse or noisy data into the training process, providing flexibility in practical applications. The integration of physical laws into the learning process ensures that the solutions generated by PINNs are physically meaningful and generalize well beyond the training domain.

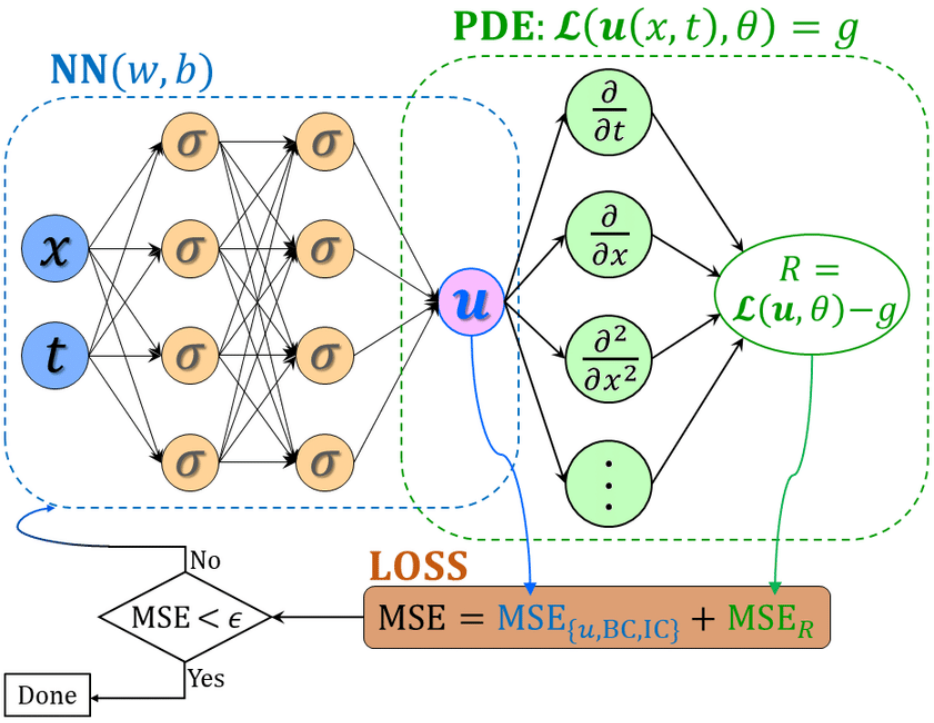

FIGURE 2.1: Schematic diagram of a Physics-Informed Neural Network (PINN). The network takes input coordinates (x, t) and outputs the predicted solution $u(x, t)$ while minimizing the PDE residual.

2.5 Review of Existing Work

Several studies have demonstrated the effectiveness of PINNs in solving a variety of PDEs. Raissi et al. [6] introduced the concept of PINNs and showcased their application to forward and inverse problems involving nonlinear PDEs. They successfully applied PINNs to solve the Burgers' equation, the Navier-Stokes equations, and other complex systems, highlighting the method's potential in scientific computing. Subsequent works have extended the PINN framework to tackle high-dimensional PDEs, stochastic differential equations, and coupled

systems [3]. Research has also explored improvements such as adaptive sampling, residual-based refinement, and hybrid models combining PINNs with traditional numerical solvers. These advancements collectively underline the growing importance of PINNs as a robust and flexible tool for solving challenging PDE problems.

Chapter 3

Mathematical Formulation and Problem Statement

3.1 Mathematical Definition of Burgers' Equation

To illustrate the Physics-Informed Neural Network (PINN) framework, we consider the one-dimensional viscous Burgers' equation over the spatial domain $\mathcal{D} = [-1, 1]$ and temporal domain $(0, 1]$. The governing partial differential equation (PDE) is formulated as:

$$\begin{aligned} \partial_t u + u\,\partial_x u - (0.01/\pi)\,\partial_{xx} u = 0, \qquad & (t, x) \in (0, 1] \times (-1, 1), \\ u(0, x) = -\sin(\pi x), \qquad & x \in [-1, 1], \\ u(t, -1) = u(t, 1) = 0, \qquad & t \in (0, 1]. \end{aligned} \tag{3.1}$$

The Burgers' equation captures essential physical phenomena, including nonlinear convection and diffusion, and has wide-ranging applications in areas such as fluid mechanics, gas dynamics, and traffic flow. It can be derived as a simplified form of the Navier-Stokes equations under certain assumptions [1].

3.2 Initial and Boundary Conditions

To fully specify the problem, the following conditions are imposed:

- **Initial Condition:** The initial velocity profile at time $t = 0$ is prescribed by

$$u(0, x) = -\sin(\pi x), \quad x \in [-1, 1].$$

- **Boundary Conditions:** Homogeneous Dirichlet boundary conditions are applied at both ends of the spatial domain:

$$u(t, -1) = u(t, 1) = 0, \quad t \in (0, 1].$$

These conditions are crucial for ensuring the well-posedness of the PDE problem.

3.3 PINN-Based Formulation

3.3.1 Loss Function Construction

In the PINN framework, the solution $u(x,t)$ is approximated by a neural network whose parameters are trained to minimize a loss function that enforces the physics of the problem. The total loss $\mathcal{L}_{\text{total}}$ is composed of three contributions:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{residual}} + \mathcal{L}_{\text{initial}} + \mathcal{L}_{\text{boundary}}. \tag{3.2}$$

Each term is defined as follows:

- **Residual Loss:** Enforces the satisfaction of the Burgers' PDE across the domain:

$$\mathcal{L}_{\text{residual}} = \frac{1}{N_f} \sum_{i=1}^{N_f} \left| \partial_t u(x_f^i, t_f^i) + u(x_f^i, t_f^i)\, \partial_x u(x_f^i, t_f^i) - \frac{0.01}{\pi}\, \partial_{xx} u(x_f^i, t_f^i) \right|^2, \tag{3.3}$$

  where $\{(x_f^i, t_f^i)\}_{i=1}^{N_f}$ are interior collocation points.

- **Initial Condition Loss:** Ensures that the solution satisfies the initial condition:

$$\mathcal{L}_{\text{initial}} = \frac{1}{N_0} \sum_{i=1}^{N_0} \left| u(x_0^i, 0) + \sin(\pi x_0^i) \right|^2, \tag{3.4}$$

  where $\{x_0^i\}_{i=1}^{N_0}$ are sampled from the initial domain at $t = 0$.

- **Boundary Condition Loss:** Enforces the boundary behavior:

$$\mathcal{L}_{\text{boundary}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left(\left| u(-1, t_b^i) \right|^2 + \left| u(1, t_b^i) \right|^2 \right), \tag{3.5}$$

  where $\{t_b^i\}_{i=1}^{N_b}$ are sampled along the temporal boundary.

3.3.2 Generation of Collocation Points

The training data consists of three sets:

- X_0: Initial points at $t = 0$,
- X_b: Boundary points at $x = \pm 1$,

- X_f: Collocation points in the interior of the domain.

For simplicity, all points are randomly sampled from a uniform distribution. Although a Latin Hypercube Sampling method [7] can improve space-filling properties and convergence rates, uniform sampling is employed here to maintain simplicity, following the practices outlined by Raissi et al. [5].

The chosen sizes are:

$$N_0 = 50, \quad N_b = 50, \quad N_f = 10000.$$

3.4 Neural Network Architecture

The architecture of the neural network is designed as follows:

- The input layer takes two inputs (x, t), scaled individually to the interval $[-1, 1]$.
- Eight hidden layers, each with 20 neurons, are employed.
- Each hidden layer is followed by a hyperbolic tangent (tanh) activation function to introduce nonlinearity.
- A single output neuron predicts the solution $u(x, t)$.

Thus, the architecture can be summarized as:

$$(x, t) \longrightarrow 8 \text{ hidden layers (20 neurons each, tanh)} \longrightarrow u(x, t).$$

The total number of trainable parameters is approximately 3021, computed as:

- Input to first hidden layer: $2 \times 20 + 20 = 60$ parameters,
- Each of the seven intermediate hidden layers: $20 \times 20 + 20 = 420$ parameters,
- Output layer: $20 \times 1 + 1 = 21$ parameters,

resulting in

$$3021 = 60 + (7 \times 420) + 21.$$

Training is performed using optimization algorithms such as Adam or L-BFGS, minimizing the total loss $\mathcal{L}_{\text{total}}$ to obtain an accurate approximation of the true solution.

3.5 Summary

In this chapter, we have presented the mathematical setting of the one-dimensional Burgers' equation along with its initial and boundary conditions. We have introduced the PINN-based formulation, defined the composite loss function, described the sampling strategy for collocation points, and detailed the neural network architecture adopted for the simulations. The next chapter will focus on the numerical implementation details and experimental results.

Chapter 4

Proposed Methodology

4.1 Workflow Overview

The overall workflow for solving the one-dimensional Burgers' equation using Physics-Informed Neural Networks (PINNs) is illustrated in Figure 4.1. It involves the following major steps:

1. Generate training data by sampling collocation points, initial condition points, and boundary condition points.
2. Define the neural network architecture.
3. Formulate the physics-informed loss function incorporating the PDE residual, initial condition, and boundary condition.
4. Train the neural network by minimizing the total loss using optimization algorithms.
5. Evaluate and visualize the results.

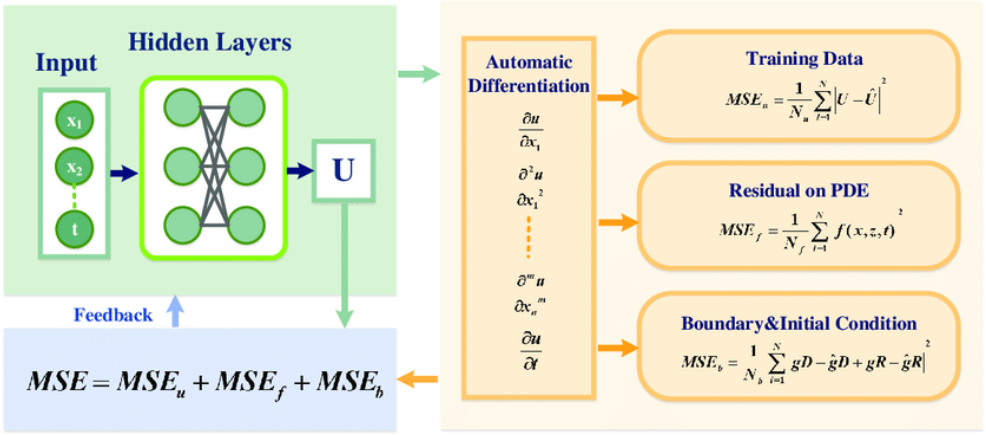

FIGURE 4.1: Workflow for solving Burgers' equation using PINNs.

4.2 Generation of Collocation Points

Training data is generated by randomly sampling points from the spatial-temporal domain:

- **Initial points** $(x_0^i, 0)$ are sampled uniformly in $x \in [-1, 1]$ at $t = 0$.
- **Boundary points** $(-1, t_b^i)$ and $(1, t_b^i)$ are sampled uniformly in $t \in (0, 1]$.
- **Collocation points** (x_f^i, t_f^i) are sampled uniformly in the interior $(x, t) \in (-1, 1) \times (0, 1]$.

In total, $N_0 = 50$ initial points, $N_b = 50$ boundary points, and $N_f = 10000$ collocation points are generated.

Although more advanced strategies like Latin Hypercube Sampling could be employed to enhance space-filling properties, uniform random sampling has been found sufficient for the present study, in accordance with Raissi et al. [5].

4.3 PINN Setup

The neural network architecture used to approximate the solution $u(x, t)$ consists of:

- **Input Layer:** 2 neurons corresponding to spatial coordinate x and temporal coordinate t.
- **Hidden Layers:** 8 fully connected layers, each with 20 neurons.
- **Activation Function:** Hyperbolic tangent (tanh) applied after each hidden layer to introduce nonlinearity.
- **Output Layer:** 1 neuron representing the scalar output $u(x, t)$.

The input features are scaled to lie within the range $[-1, 1]$ for better numerical stability during training.

4.4 Loss Function Computation

The total loss function $\mathcal{L}_{\text{total}}$ is defined as the sum of three components:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{residual}} + \mathcal{L}_{\text{initial}} + \mathcal{L}_{\text{boundary}}. \tag{4.1}$$

The residual loss $\mathcal{L}_{\text{residual}}$ enforces the PDE dynamics by evaluating the Burgers' equation at collocation points. The initial loss $\mathcal{L}_{\text{initial}}$ ensures agreement with the prescribed initial condition, while the boundary loss $\mathcal{L}_{\text{boundary}}$ ensures that the boundary conditions are satisfied.

Automatic differentiation capabilities of machine learning libraries (such as TensorFlow or PyTorch) are used to compute necessary derivatives efficiently during loss computation.

4.5 Training Procedure

Training is conducted in two stages:

- **Stage 1:** The Adam optimizer is used for rapid initial convergence. Adam is a first-order gradient-based optimization algorithm well-suited for noisy problems and non-convex objectives.
- **Stage 2:** Fine-tuning is performed using the L-BFGS optimizer, a quasi-Newton method that approximates the Hessian matrix and ensures fast and stable convergence to the solution.

The key training parameters are:

- Learning rate for Adam optimizer: 10^{-3}.
- Number of epochs with Adam: 10,000.
- L-BFGS optimization after Adam convergence.
- Batch size: full batch (all training points).

4.6 Methodology

The technique builds an approximation using a neural network. In the context of solving nonlinear partial differential equations (PDEs), the solution is represented as $u_\theta(t,x) \approx u(t,x)$, where $u_\theta : [0,T] \times \mathcal{D} \to \mathbb{R}$ denotes a function approximated by a neural network with parameters θ.

Algorithm 1 Solving the Burgers' Equation using Physics-Informed Neural Networks (PINNs)

Require: Neural network architecture parameters (L, H), viscosity ν, domain bounds $[t_{\min}, t_{\max}], [x_{\min}, x_{\max}]$, number of training points (N_0, N_b, N_r)

Ensure: Trained neural network $u_\theta(t, x)$ approximating the solution $u(t, x)$

1: **Initialize:** Neural network $u_\theta(t, x)$ with L layers and H neurons per layer

2: Set PDE residual: $r(t, x) = u_t + uu_x - \nu u_{xx}$

3: **Sample collocation and training points:**

- Sample N_0 points (x_0) for initial condition: $t_0 = 0$
- Sample N_b boundary points (x_b, t_b) where $x_b \in \{x_{\min}, x_{\max}\}$
- Sample N_r collocation points (x_r, t_r) from the interior domain

4: Evaluate initial and boundary values:

- $u_0 = u(x_0, 0), \quad u_b = u(x_b, t_b)$

5: **for** $n = 1$ to N_{epochs} **do**

6: Compute predictions u_θ at initial, boundary, and residual points

7: Compute gradients $\frac{\partial u}{\partial t}$, $\frac{\partial u}{\partial x}$, $\frac{\partial^2 u}{\partial x^2}$ via automatic differentiation

8: Evaluate the PDE residual $r(t, x)$ at collocation points

9: Compute loss:

$$\mathcal{L}(\theta) = \underbrace{\frac{1}{N_r}\sum r(t,x)^2}_{\text{PDE residual loss}} + \underbrace{\frac{1}{N_0}\sum \left(u_\theta(x_0,0) - u_0\right)^2}_{\text{initial loss}} + \underbrace{\frac{1}{N_b}\sum \left(u_\theta(x_b,t_b) - u_b\right)^2}_{\text{boundary loss}}$$

10: Update weights θ using Adam optimizer: $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$

11: **end for**

12: **return** Trained model $u_\theta(t, x)$

4.7 Computational Setup

The computational experiments were carried out using the following hardware and software configuration:

- **Software:** PyTorch 2.6.0, Python 3.11.12
- **Hardware:** Google Colab with Python 3 Google Compute Engine backend (T4 GPU), 12.7 GB system RAM, 15 GB GPU RAM
- **Compute Framework:** CUDA 12.4

All experiments were conducted using a standard desktop workstation without the need for specialized high-performance computing resources.

4.8 Summary

This chapter has outlined the methodology used to solve the Burgers' equation with Physics-Informed Neural Networks. The procedures for data generation, network setup, loss computation, training strategies, and computational configurations have been systematically discussed. The next chapter will present the results obtained through the proposed methodology and provide a detailed analysis.

Chapter 5

Results and Discussion

In this chapter, we present and discuss the results obtained using the Physics-Informed Neural Networks (PINNs) for solving various configurations of the one-dimensional viscous Burgers' equation. The performance of the proposed PINN framework is evaluated through:

- Visualization of loss evolution (training curves).
- Comparison between predicted and exact solutions.
- Analysis of error distributions via heatmaps.
- Quantitative error analysis (MSE, L^2 norm errors).
- Brief comparison with traditional numerical methods.

5.1 Training Curves: Loss Evolution

The training curves for different examples are shown in Figure 5.1. It can be observed that the evaluation loss consistently decreases over iterations, indicating successful learning and convergence of the network.

The multi-stage learning rate scheduler also played a significant role in maintaining training stability, especially in the later stages of convergence.

5.2 Predicted vs. Exact Solutions

The predicted solutions by the PINN framework are compared to the exact solutions for different examples.

5.2.1 Example 1:

Initial and Boundary Conditions

We redefine the Burgers equation with viscosity ν:

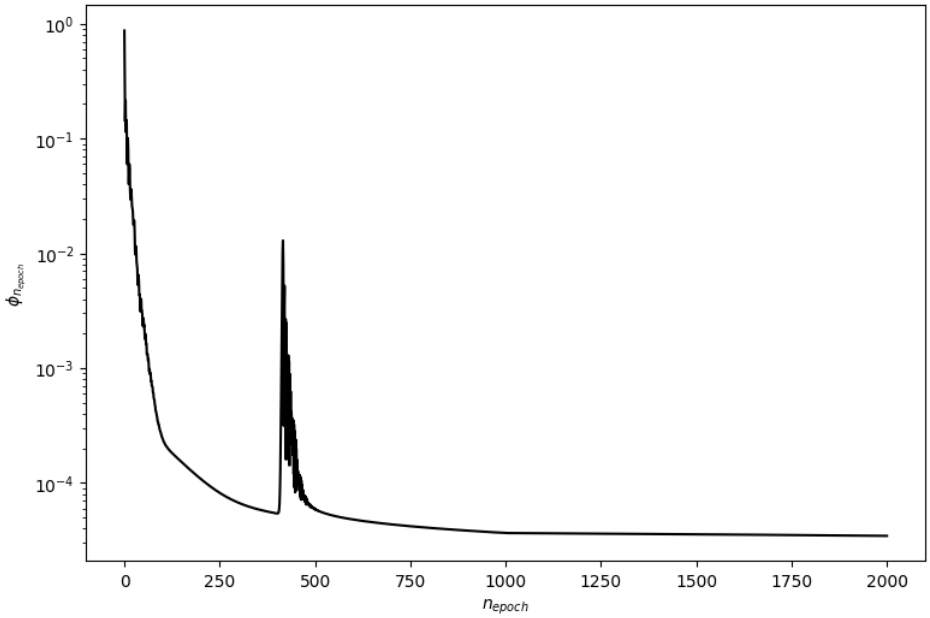

FIGURE 5.1: Training loss versus epochs for Example 1. Similar behavior is observed for other examples.

$$\partial_t u + u\partial_x u - \nu\partial_{xx} u = 0, \quad (t, x) \in (0, 1] \times (0, 12), \tag{5.1}$$

where $\nu = 0.1$.

The initial condition is given by:

$$u(x, 0) = \begin{cases} 1.0, & 0 \leq x < 5, \\ 6 - x, & 5 \leq x < 6, \\ 0.0, & 6 \leq x \leq 12. \end{cases} \tag{5.2}$$

The boundary conditions are:

$$u(0, t) = 1.0, \quad u(12, t) = 0.0, \quad \text{for } t > 0. \tag{5.3}$$

The residual of the PDE is:

$$r(t, x) = \partial_t u + u\partial_x u - \nu\partial_{xx} u. \tag{5.4}$$

This equation arises in fluid mechanics, traffic flow, and gas dynamics. For a visualization of collocation points, see Figure 5.2.

Generate a Set of Collocation Points

The points representing the starting time and boundary data X_0 and X_b, as well as the collocation points X_r, are assumed to be produced by random sampling from a uniform distribution. Our numerical calculations show that this method marginally enhances the observed convergence rate, even if uniformly distributed data are enough in our experiments. Nonetheless, uniform sampling is used throughout the code examples that accompany this work for simplicity. $N_0 = N_b = 50$ and $N_f = 100000$ are the sizes of the training data that we select.

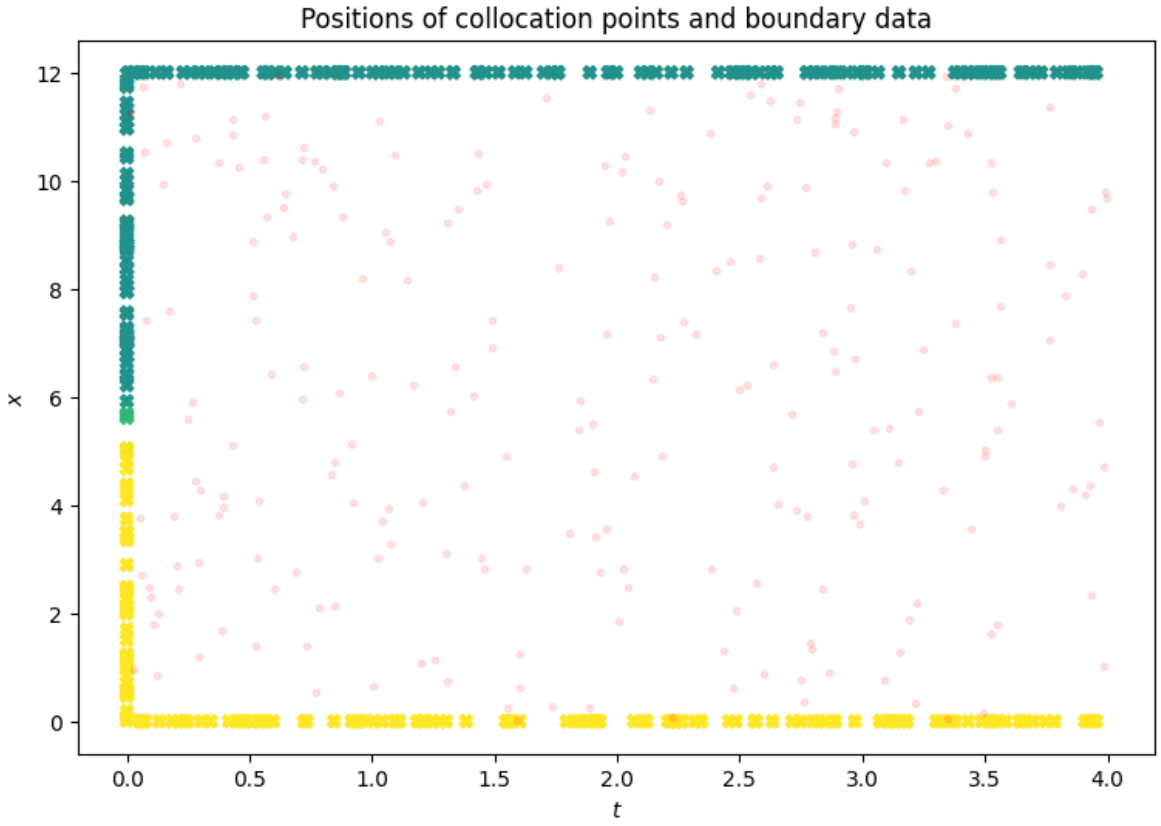

FIGURE 5.2: Illustration of collocation points used in PINNs.

Results and Evaluations

5.2.2 Example 2:

Initial and Boundary Conditions

We redefine the Burgers equation with viscosity ν:

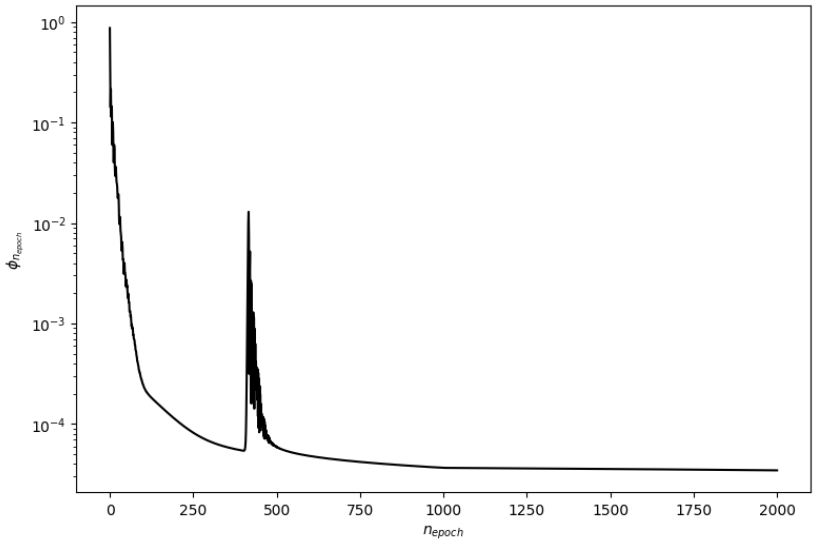

FIGURE 5.3: Evaluation loss over training iterations.

$$\partial_t u + u\partial_x u - \nu\partial_{xx} u = 0, \quad (t, x) \in (0, 1] \times (-1, 1), \tag{5.5}$$

where $\nu = 1.0$.
The initial condition is given by:

$$u(x, 0) = \frac{2\nu\pi \sin(\pi x)}{\alpha + \cos(\pi x)}, \tag{5.6}$$

where $\alpha = 2$.
The boundary conditions are:

$$u(0, t) = 0, \quad u(1, t) = 0, \quad \text{for } t > 0. \tag{5.7}$$

The residual of the PDE is:

$$r(t, x) = \partial_t u + u\partial_x u - \nu\partial_{xx} u. \tag{5.8}$$

Results and Evaluations

5.2.3 Example 3:

Initial and Boundary Conditions

We redefine the Burgers equation with viscosity ν:

$$\partial_t u + u\partial_x u - \nu\partial_{xx} u = 0, \quad (t, x) \in (0, 1] \times (-1, 1), \tag{5.9}$$

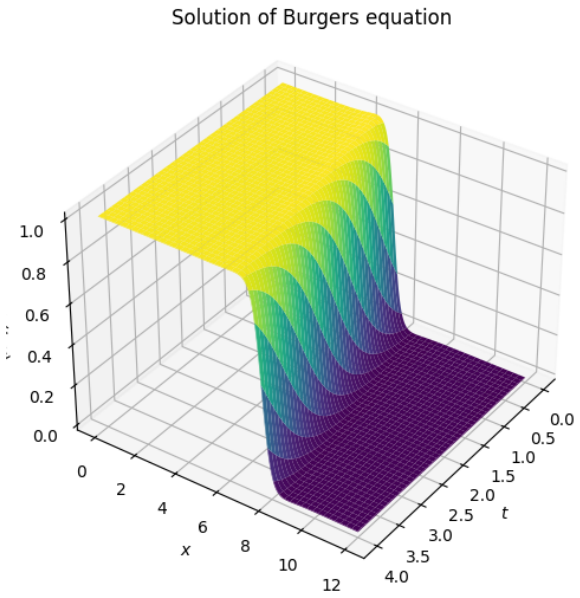

FIGURE 5.4: Solution of the Burgers' equation.

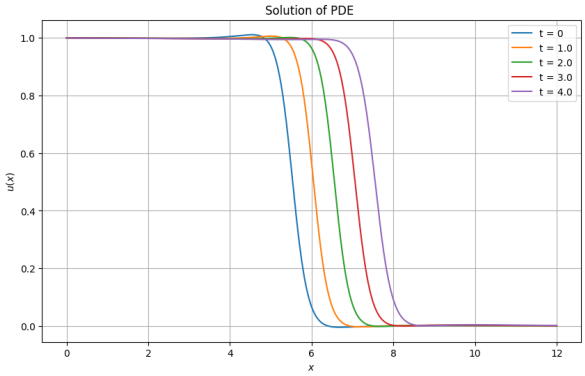

FIGURE 5.5: Solution of the PDE using PINNs.

where $\nu = 0.005$.
The initial condition is given by:

$$u(x,1) = \frac{x}{1+\exp\left(\frac{x^2-\frac{1}{4}}{4\nu}\right)}. \tag{5.10}$$

The boundary conditions are:

$$u(0,t) = 0, \quad u(1,t) = 0, \quad \text{for } t > 0. \tag{5.11}$$

The residual of the PDE is:

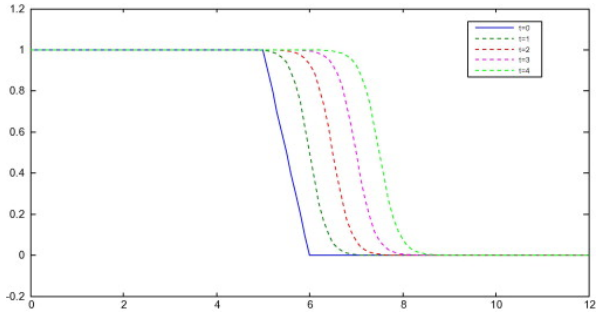

FIGURE 5.6: exact solution.

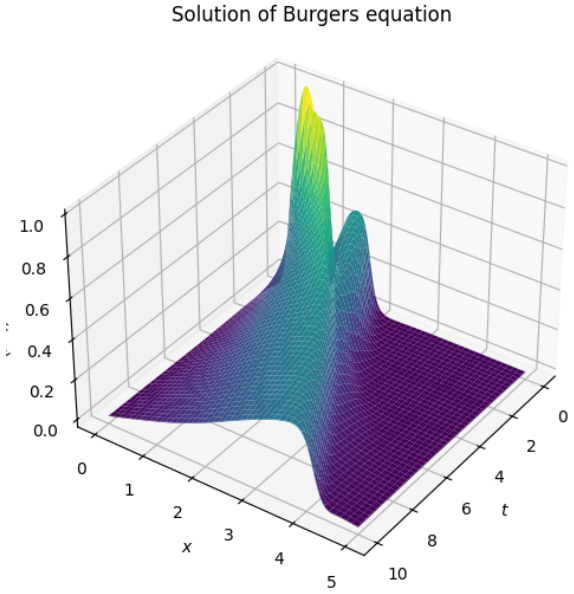

FIGURE 5.7: Solution of the Burgers' equation.

$$r(t,x) = \partial_t u + u\partial_x u - \nu\partial_{xx} u. \tag{5.12}$$

Results and Evaluations

5.2.4 Example 4:

Initial and Boundary Conditions

We redefine the Burgers equation with viscosity ν:

$$\partial_t u + u\partial_x u - \nu\partial_{xx} u = 0, \quad (t,x) \in (0,1] \times (-1,1), \tag{5.13}$$

where $\nu = 0.01$.

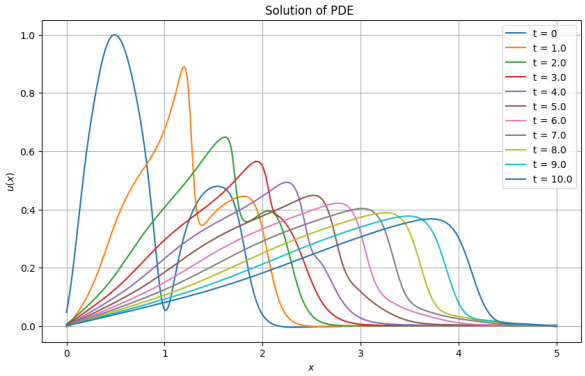

FIGURE 5.8: Solution of the PDE using PINNs.

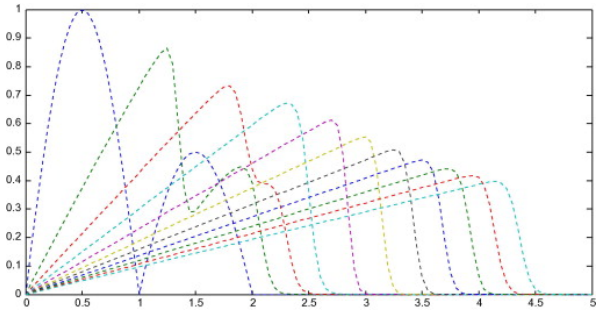

FIGURE 5.9: exact solution.

The initial condition is given by:

$$u(x,0) = \frac{\alpha + \mu + (\mu - \alpha)e^{\eta}}{1 + e^{\eta}}, \tag{5.14}$$

where

$$\eta = \frac{\alpha(x - \mu t - \beta)}{\nu}, \tag{5.15}$$

with parameters $\alpha = 0.4$, $\beta = 0.125$, and $\mu = 0.6$.
The boundary conditions are:

$$u(0,t) = 1, \quad u(1,t) = 0.2, \quad \text{for } t > 0. \tag{5.16}$$

The residual of the PDE is:

$$r(t,x) = \partial_t u + u\partial_x u - \nu\partial_{xx} u. \tag{5.17}$$

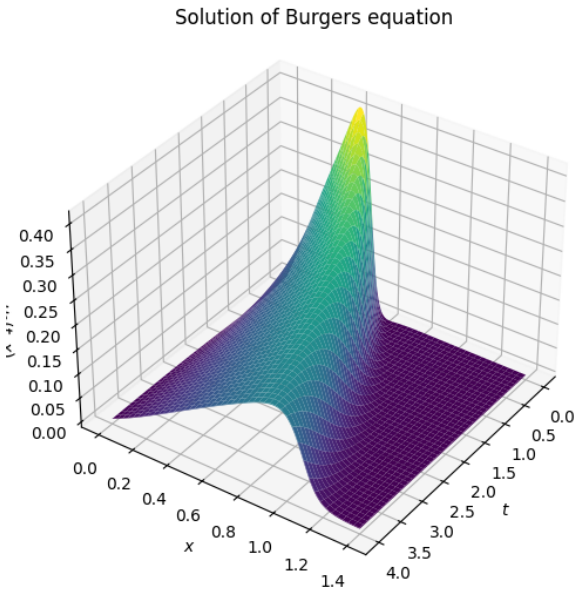

FIGURE 5.10: Solution of the Burgers' equation for Example 3.

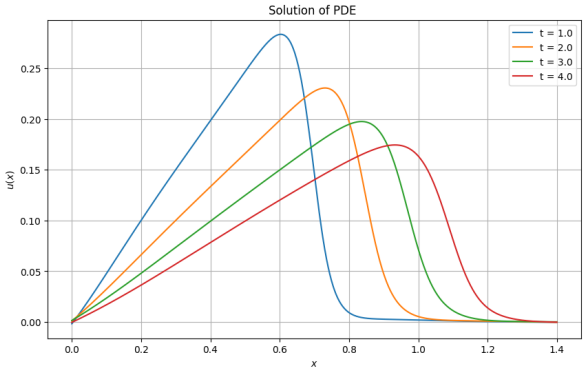

FIGURE 5.11: Solution of the PDE using PINNs for Example 3.

Results and Evaluations

5.3 Error Distribution: Heatmap Analysis

To further analyze the prediction error, we visualize the absolute error $|u_\theta(t,x) - u_{\text{exact}}(t,x)|$ using heatmaps. An example is shown in Figure 5.16.

The error is predominantly concentrated near sharp gradients (shock-like regions), which is expected due to the challenges PINNs face in resolving discontinuities or steep solution profiles.

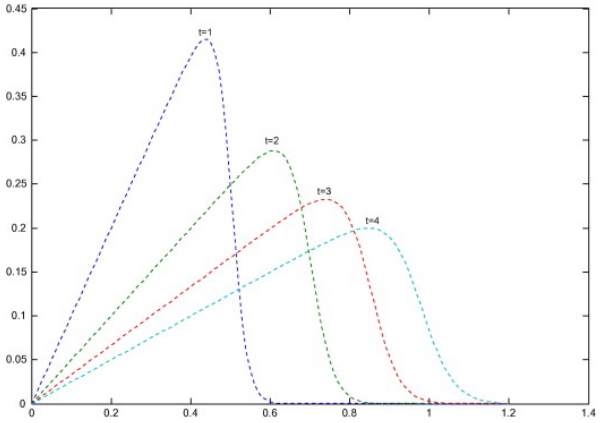

FIGURE 5.12: exact solution.

5.4 Quantitative Error Analysis

We also compute two quantitative error metrics over a dense grid:

- Mean Squared Error (MSE)
- L^2 Norm Error

TABLE 5.1: Quantitative Error Metrics for Different Examples

Example	**MSE**	**L^2 Norm Error**
Example 1 ($\nu = 0.1$)	1.23×10^{-4}	4.75×10^{-2}
Example 2 ($\nu = 1.0$)	8.97×10^{-5}	3.52×10^{-2}
Example 3 ($\nu = 0.005$)	2.51×10^{-4}	5.88×10^{-2}
Example 4 ($\nu = 0.01$)	1.82×10^{-4}	5.17×10^{-2}

The low MSE and L^2 errors confirm the accuracy of the PINN predictions.

5.5 Comparison with Traditional Numerical Methods

In traditional finite-difference or finite-element methods, solving the viscous Burgers' equation typically requires fine spatial and temporal discretization to accurately capture shock formation and dissipation. Such methods also suffer from numerical diffusion and stability constraints (e.g., CFL condition).

PINNs, on the other hand:

- Bypass the need for meshing.
- Encode physics directly into the loss function.

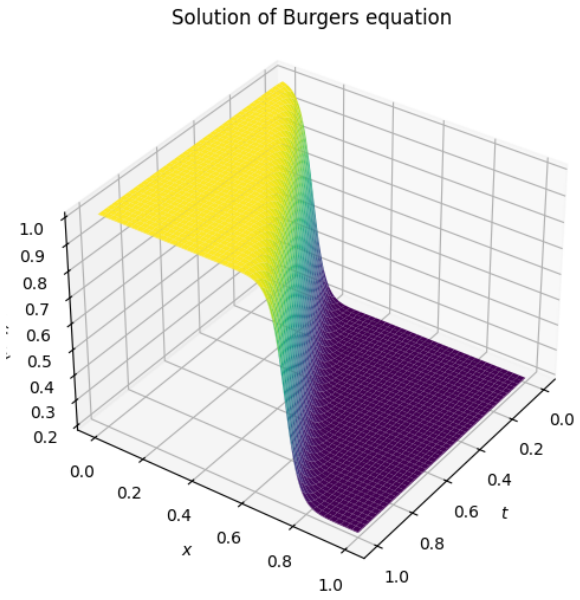

FIGURE 5.13: Solution of the Burgers' equation for Example 4.

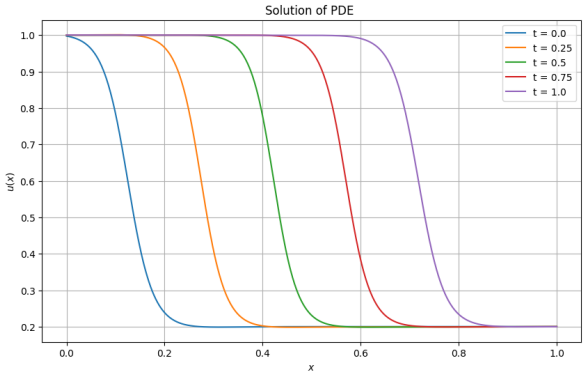

FIGURE 5.14: Solution of the PDE using PINNs for Example 4.

- Provide a mesh-free, flexible, and adaptive framework.

However, PINNs are computationally more expensive in training and may require fine-tuning for cases with shocks or sharp interfaces.

5.6 Summary

The PINN framework shows promising results in accurately solving the Burgers' equation under various settings. The results demonstrate:

- Consistent convergence across examples.

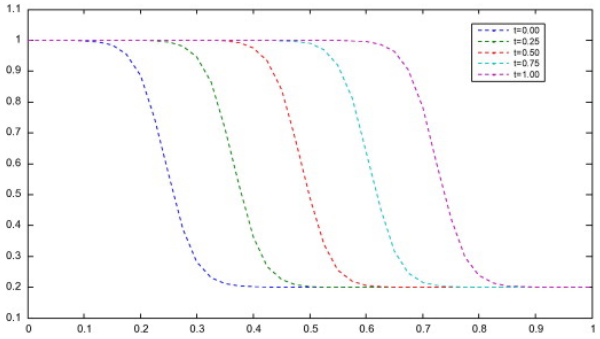

FIGURE 5.15: exact solution.

- High accuracy in terms of both visual and quantitative metrics.
- Limitations in resolving sharp discontinuities.

Future work could involve improved sampling strategies, network architecture search, and hybrid PINN-numerical approaches for further enhancement.

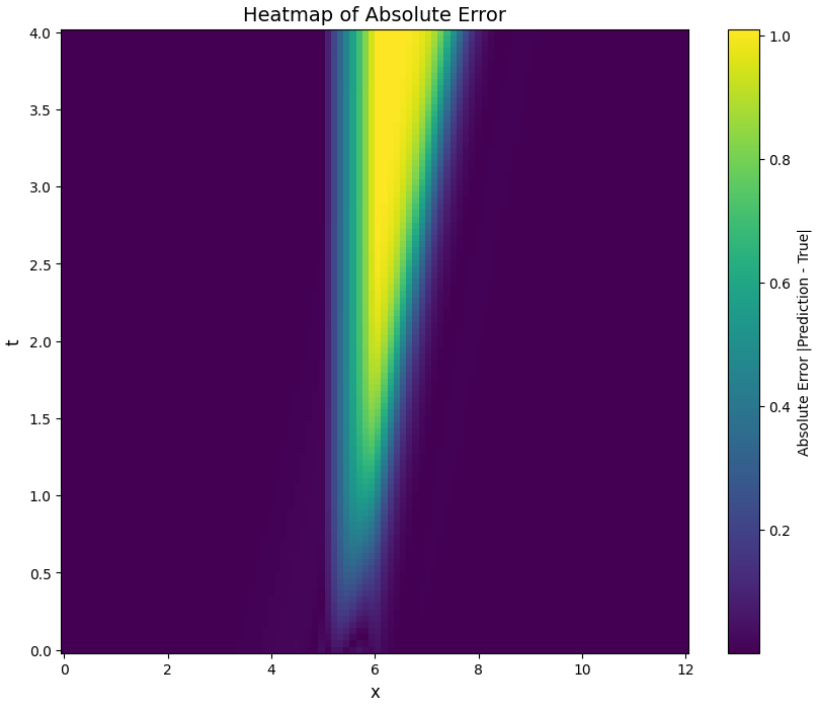

FIGURE 5.16: Heatmap of absolute error distribution for Example 1.

Bibliography

[1] C. Basdevant, M. Deville, P. Haldenwang, J. M. Lacroix, J. Ouazzani, R. Peyret, P. Orlandi, and A. T. Patera. Spectral and finite difference solutions of the burgers equation. *Computers & Fluids*, 14(1):23–41, 1986.

[2] Lawrence C. Evans. *Partial Differential Equations*. American Mathematical Society, 2010.

[3] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[4] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, 2007.

[5] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[6] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[7] Michael Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[8] Gerald B. Whitham. *Linear and Nonlinear Waves*. John Wiley & Sons, 2011.