Role of Fibonacci Sequence Based Activation Function in Physics Informed Neural Networks (PINNs)

Report submitted in fulfillment of the requirements
for the thesis project of

Final Year Integrated Dual Degree
in
Mathematical Sciences

by

Manish Singh 19124021

Under the guidance of

Dr. Rajeev

Department of Mathematical Sciences
INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India
May 2024

Dedicated to

My parents, teachers, family and friends

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and under the general supervision of my supervisor.
2. The work has not been submitted for any project.
3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.
4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Manish Singh

Department of Mathematical Sciences,

Indian Institute of Technology (BHU) Varanasi,

INDIA 221005.

Certificate

*This is to certify that the work contained in this report entitled "**Role of Fibonacci Sequence Based Activation Function in Physics Informed Neural Networks (PINNs)**" being submitted by **Manish** (**Roll No. 19124021**), carried out in the Department of Mathematical Sciences, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Dr. Rajeev
Supervisor of Thesis
Department of Mathematical Sciences,
Indian Institute of Technology (BHU)
Varanasi, INDIA 221005

Dr. SK Pandey
Head of Department
Department of Mathematical Sciences,
Indian Institute of Technology (BHU)
Varanasi, INDIA 221005

Abstract

This thesis investigates the efficacy of a novel activation function based on the Fibonacci sequence in Physics-Informed Neural Networks (PINNs). PINNs, a class of deep learning models, integrate differential equations directly into their loss function, offering a potent framework for solving complex physical and engineering problems. Traditional activation functions, such as the hyperbolic tangent (tanh), while widely utilized, often limit the model's ability to approximate complex behaviors inherent in physical systems. This research introduces a new Fibonacci sequence-based activation function, positing that its unique properties can better handle the non-linearities of physical phenomena. We conducted extensive experiments to assess this hypothesis, focusing on solving the non-linear 1D Burgers' equation, the 1D Poisson equation, and the 2D Helmholtz equation. Our results demonstrate that PINNs equipped with the Fibonacci-based activation function outperform those using the classical activation functions like tanh and sigmoid in terms of accuracy and error reduction. The findings suggest that the intrinsic properties of the Fibonacci sequence may offer significant advantages in the context of PINNs, paving the way for more effective solutions to a broad range of complex differential equations in applied physics and engineering.

Keywords: Physics-Informed Neural Networks, Fibonacci Sequence, Activation Function, Deep Learning, Non-linear Differential Equations, Burgers', Poisson, Elastostatic, Heat and Helmholtz Equation

Contents

Chapter 1

Introduction

1.1 Overview

Physics-Informed Neural Networks (PINNs) have become a significant tool for solving differential equations by incorporating physical laws into their loss functions, thus revolutionizing solutions for partial differential equations and beyond [1]. Traditional activation functions such as tanh and sigmoid are commonly employed in PINNs but may not always provide optimal performance in complex scenarios involving non-linear phenomena, as highlighted in recent advances [2, 3]. This research introduces an innovative Fibonacci sequence-based activation function within the standard PINN architecture to explore its potential advantages.

1.2 Motivation for the Research Work

Traditional activation functions in neural networks, chosen for their mathematical properties like non-linearity and differentiability, may not adequately capture the complexity required by PINNs. The Fibonacci sequence, known for its intrinsic natural and mathematical properties, offers a novel foundation for developing an activation function potentially improving the network's learning and predictive accuracy. This study explores such unconventional approaches to enhance the performance and applicability of PINNs, especially for solving complex equations such as the 1D Burgers' equation, the 1D Poisson equation, time dependent 1D Heat equation, 2D Helmholtz equation and the 2D Elastostatic equation where novel approaches have shown promise in recent studies [4, 5, 6, 7, 8].

1.3 Contribution

This thesis contributes to the field by:

- Developing a new activation function based on the Fibonacci sequence, designed to be compatible with PINN architectures.
- Implementing and integrating this activation function within the standard PINN architecture, as evidenced by related ad-

vancements in PINN architecture customizations.

- Conducting comprehensive testing and evaluation using synthetic data for the 1D Burgers' equation, the 1D Poisson equation, time dependent 1D Heat equation, 2D Helmholtz equation and the 2D Elastostatic equation, comparing its performance against traditional activation functions.

1.4 Related Work

Activation functions are crucial in neural networks, introducing non-linear properties essential for learning complex patterns. In PINNs, the choice of activation function significantly affects model accuracy and convergence. Prior research has focused on conventional functions like tanh and Sigmoid; however, tailoring activation functions to specific differential equations could enhance results. This thesis builds on these insights by exploring a Fibonacci sequence-based activation function, aiming to utilize its unique properties for improved performance in physical equations [9, 10].

1.5 Organization of the Report

This thesis is organized into the following chapters:

1. **Chapter 1: Introduction** - Provides an overview, states

the motivation, outlines contributions, and reviews related works.

2. **Chapter 2: Background and Theory** - Discusses the theoretical underpinnings of PINNs, activation functions, and the Fibonacci sequence.

3. **Chapter 3: Methodology** - Details the design and implementation of the Fibonacci-based activation function within PINNs.

4. **Chapter 4: Results and Discussion** - Describes experimental setups, datasets, and presents findings.

5. **Chapter 5: Conclusion and Future Work** - Summarizes the thesis and outlines future research directions.

Chapter 2

Background and Theory

2.1 Physics-Informed Neural Networks

Physics-Informed Neural Networks (PINNs) integrate deep learning with physical modeling, using neural networks to solve supervised learning tasks while adhering to predefined physical laws, often described by differential equations[1].

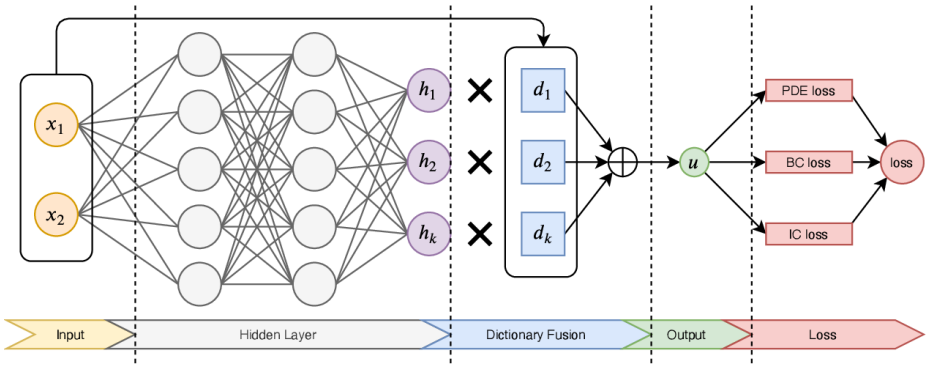

Figure 2.1: Architecture of a Physics-Informed Neural Network (PINN)

Figure 2.1 illustrates the architecture of a Physics-Informed Neural Network (PINN). This network comprises several key components. On the left, the input layer receives variables

$x_1, x_2, \ldots$, which are fed into a series of fully connected hidden layers. Each hidden layer is designed to capture intricate patterns and relationships within the data. The output from these hidden layers is then used in the dictionary fusion step, represented by components $d_1, d_2, \ldots, d_k$, which merge the learned features. The result of this fusion is an output u, which is further evaluated against physical laws encoded in the loss function components. These components include PDE (Partial Differential Equation) loss, BC (Boundary Condition) loss, and IC (Initial Condition) loss, ensuring that the network's predictions are consistent with underlying physical principles. This architecture highlights the integration of data-driven learning with physical model constraints, enabling PINNs to solve complex scientific problems effectively.

2.1.1 Key Components

PINNs integrate the following components:

1. **Neural Network Architecture** - A fully connected deep neural network that approximates functions of interest.

2. **Loss Function** - The loss function in PINNs is given by:

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \lambda \mathcal{L}_{physics}(\theta),$$

   where $\mathcal{L}_{data}$ is the loss due to data discrepancy, $\mathcal{L}_{physics}$ rep-

resents the physical law constraints, and λ is a weighting factor.

3. **Training Algorithm** - Typically involves gradient descent techniques to minimize the loss, ensuring compliance with both data and physical laws.

2.1.2 Application Areas

PINNs are applied in fluid dynamics, quantum mechanics, and materials science, effectively solving complex physical problems.

2.2 Activation Functions

Activation functions introduce necessary non-linearities into the network's architecture, crucial for learning complex patterns[11].

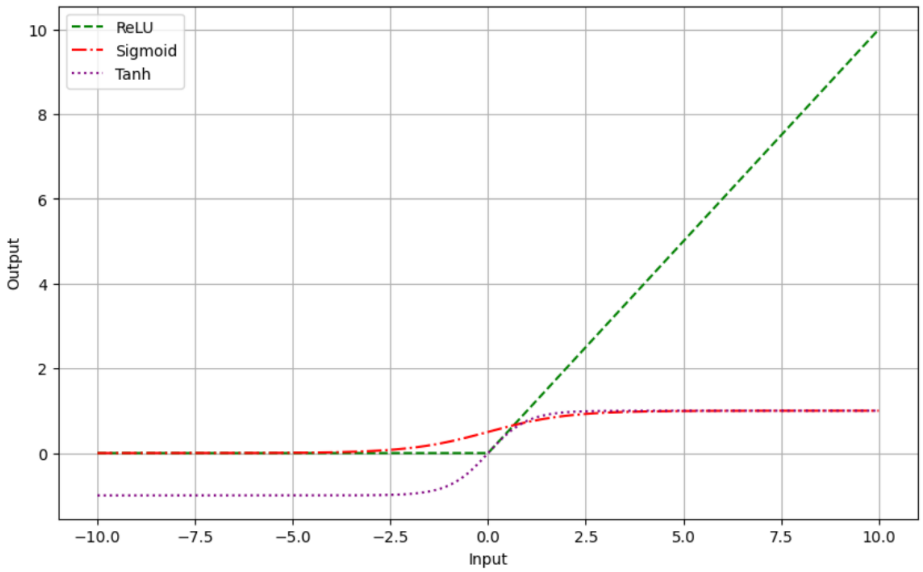

Figure 2.2: Common Activation Functions

2.2.1 Common Activation Functions

Commonly used activation functions include [12], [13]:

- **Tanh** (Hyperbolic Tangent):

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.1}$$

- **Sigmoid**:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

- **ReLU** (Rectified Linear Unit):

$$f(x) = \max(0, x) \tag{2.3}$$

2.2.2 Challenges and Innovations

Traditional activation functions, while effective in many scenarios, may not always suit the specific demands of PINNs. Innovations in activation function design aim to optimize both physical consistency and computational efficiency.

2.3 The Fibonacci Sequence

The Fibonacci sequence is defined by the recurrence relation [14],[15]:

$$F_n = F_{n-1} + F_{n-2}, \quad F_0 = 0, \quad F_1 = 1. \tag{2.4}$$

2.3.1 Fibonacci Polynomials

Fibonacci polynomials extend the concept of the Fibonacci sequence. They are defined by the recurrence relation:

$$F_n(x) = xF_{n-1}(x) + F_{n-2}(x), \quad F_0(x) = 0, \quad F_1(x) = 1. \tag{2.5}$$

2.3.2 Dynamic Programming Optimization

Dynamic programming is a method for solving problems by breaking them down into simpler subproblems and storing the solutions to these subproblems to avoid redundant computations. We apply this technique to compute Fibonacci polynomials efficiently.

Base Cases

The base cases for Fibonacci polynomials are straightforward:

$$F_0(x) = 0 \tag{2.6}$$

$$F_1(x) = x \tag{2.7}$$

Recurrence Relation

For $n \geq 2$, the Fibonacci polynomial can be computed using the following recurrence relation:

$$F_n(x) = x \cdot F_{n-1}(x) + F_{n-2}(x) \tag{2.8}$$

2.3.3 Algorithm Implementation

The dynamic programming approach involves storing the previously computed values of the Fibonacci polynomials to avoid redundant calculations. The algorithm can be implemented as follows:

```
def fibonacci_polynomial(x_value, degree):
    # Handle base cases
    if degree == 0:
        return 0
    elif degree == 1:
        return x_value

    # Initialize variables for dynamic programming
    f_2 = 0
    f_1 = x_value

    # Compute the Fibonacci polynomials iteratively
    for i in range(2, degree + 1):
        f_n = x_value * f_1 + f_2
        f_2 = f_1
        f_1 = f_n

    # Return the required polynomial value
    return f_n
```

Chapter 3

Methodology

3.1 Design of Custom PINNs with Fibonacci Activation Function

The Physics-Informed Neural Networks (PINNs) developed in this study integrate a custom activation function derived from the Fibonacci sequence. The architecture is a fully connected deep neural network with input, multiple hidden, and output layers. The hidden layers utilize the Fibonacci-based activation function to process the nonlinear characteristics of the differential equations being modeled.

3.1.1 Network Architecture

Input Layer

Assume the PINNs have d input neurons, denoted as $\mathbf{x} = [x_1, x_2, \ldots, x_d]$.

Hidden Layers

The network has n hidden layers, each with m nodes. Let $\mathbf{h}^l = [h_1^l, h_2^l, \ldots, h_m^l]$ be the activations of the l-th hidden layer. The activation of the j-th node in the i-th layer, h_j^i, is computed as:

$$h_j^i = \phi_{i+j} \left(\sum_{k=1}^{m} w_{jk}^{i-1} h_k^{i-1} + b_j^i \right) \tag{3.1}$$

where w_{jk}^{i-1} is the weight from the k-th node in the $(i-1)$-th layer to the j-th node in the i-th layer, b_j^i is the bias for the j-th node in the i-th layer, and ϕ_{i+j} is the Fibonacci polynomial of degree $i+j$.

Fibonacci activation function is implemented in tf as follows:

```
class FibonacciActivation(tf.keras.layers.Layer):
    def __init__(self, degree=2, **kwargs):
        super(FibonacciActivation, self).__init__(**kwargs)
        self.degree = degree

    def call(self, inputs):
        return fibonacci_polynomial(inputs, self.degree)
```

Output Layer

The output layer consists of a single neuron that aggregates the information from the final hidden layer:

$$y = \sum_{j=1}^{m} w_j^n h_j^n + b \tag{3.2}$$

where w_j^n is the weight from the j-th node in the n-th layer to the output neuron, and b is the output bias.

3.1.2 Calculation of Total Loss

Loss in PINNs is a weighted sum of data fidelity and physical law compliance, uniquely combined to train the network.

The total loss $\mathcal{L}$ for the network is defined as:

$$\mathcal{L}(\theta) = \mathcal{L}_{data}(\theta) + \lambda \mathcal{L}_{physics}(\theta), \tag{3.3}$$

where $\mathcal{L}_{data}$ represents the mean squared error (MSE) between the predicted ($\hat{y}$) and true values (y):

$$\mathcal{L}_{data}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2, \tag{3.4}$$

where N is the number of data points.

For the 1D Poisson equation:

$$\mathcal{L}_{physics}(x) = \left(u_{xx} - f(x)\right)^2, \tag{3.5}$$

where $u(x)$ is the unknown function to be solved, $f(x)$ is the source term, and u_{xx} is the second spatial derivative of $u(x)$ with respect to x.

For the 1D Burgers' equation:

$$\mathcal{L}_{physics}(x,t) = \left(u_t + uu_x - \nu u_{xx}\right)^2, \tag{3.6}$$

where u is the velocity field, ν is the kinematic viscosity, u_t is the time derivative, and u_x, u_{xx} are the first and second spatial derivatives, respectively.

For the 1D Heat equation:

$$\mathcal{L}_{\text{physics}}(x,t) = \left(\frac{\partial u}{\partial t} - \alpha\frac{\partial^2 u}{\partial x^2} - f(x,t)\right)^2, \tag{3.7}$$

where $u(x,t)$ is the temperature distribution, α is the thermal diffusivity, and $f(x,t)$ is the heat source term.

For the 2D Helmholtz equation:

$$\mathcal{L}_{physics}(x,y,t) = \left(\Delta u + k^2 u - f\right)^2, \tag{3.8}$$

where Δ denotes the Laplacian operator, k is the wave number, u is the wave function, and f represents source terms.

For the 2D Elastostatic equation:

$$\mathcal{L}_{\text{physics}}(\mathbf{u}) = \left(\nabla \cdot \sigma + \mathbf{f}\right)^2, \tag{3.9}$$

where $\mathbf{u}$ is the displacement field, σ is the stress tensor, $\nabla\cdot$ represents the divergence operator, and $\mathbf{f}$ denotes external body forces.

3.1.3 Gradient Computation using Backpropagation

Gradients of the loss function with respect to network parameters θ are crucial for training. In PyTorch, gradient computation is facilitated by the autograd library, which automatically computes derivatives. This process involves:

1. Forward pass: Compute loss $\mathcal{L}(\theta)$ by passing input data through the network.

2. Backward pass: Automatically calculate the gradients by propagating the error back through the network layers:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \text{autograd}(\mathcal{L}, \theta_i). \tag{3.10}$$

3. Using the chain rule, the partial derivatives of the loss with respect to each parameter are calculated as follows:

$$\frac{\partial \mathcal{L}}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \theta_i}, \tag{3.11}$$

   where $\frac{\partial \mathcal{L}}{\partial \hat{y}}$ depends on the derivative of the loss function used, and $\frac{\partial \hat{y}}{\partial \theta_i}$ is derived from the network's output with respect to its weights and biases.

   The gradients of the loss function with respect to the weights and biases are computed using the chain rule.

For a weight w_{jk}^i, the gradient is:

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^i} = \frac{\partial \mathcal{L}}{\partial h_j^i} \frac{\partial h_j^i}{\partial w_{jk}^i}, \tag{3.12}$$

where

$$\frac{\partial h_j^i}{\partial w_{jk}^i} = \phi'_{i+j} \left(\sum_{k=1}^{m} w_{jk}^{i-1} h_k^{i-1} + b_j^i \right) h_k^{i-1}, \tag{3.13}$$

and $\phi'_{i+j}(x)$ is the derivative of the Fibonacci polynomial of degree $i + j$.

For a bias b_j^i, the gradient is:

$$\frac{\partial \mathcal{L}}{\partial b_j^i} = \frac{\partial \mathcal{L}}{\partial h_j^i} \frac{\partial h_j^i}{\partial b_j^i}, \tag{3.14}$$

where

$$\frac{\partial h_j^i}{\partial b_j^i} = \phi'_{i+j} \left(\sum_{k=1}^{m} w_{jk}^{i-1} h_k^{i-1} + b_j^i \right). \tag{3.15}$$

This method efficiently handles the complex derivatives involved in PINNs, especially given the non-standard activation function and the physics-informed components of the loss.

3.2 Data Preparation

This section outlines the data preparation process for solving the 1D Burgers' equation, the 1D Poisson equation, time dependent 1D Heat equation, 2D Helmholtz equation and the 2D

Elastostatic equation using Physics-Informed Neural Networks (PINNs) with the Fibonacci sequence-based activation function. The process involves selecting collocation points, utilizing Latin Hypercube Sampling (LHS).

3.2.1 Sampling Points in the Domain

Collocation Points

Collocation points are critical for discretizing the domains to solve the equations effectively. These points must capture the essential features of the solution across the domains. For our implementation, we employ uniformly spaced collocation points for each equation.

For the 1D domains, we use N uniformly spaced collocation points:

$$x_i = -1 + \frac{2i}{N-1}, \quad i = 0, 1, 2, \ldots, N-1. \tag{3.16}$$

For the 2D domain, we use a grid of $N \times N$ collocation points:

$$(x_i, y_j) = \left(-1 + \frac{2i}{N-1}, -1 + \frac{2j}{N-1}\right), \quad i, j = 0, 1, 2, \ldots, N-1. \tag{3.17}$$

Latin Hypercube Sampling

Latin Hypercube Sampling (LHS) is used to generate well-distributed sets of sample points across the input domains [16]. This sampling technique ensures that the entire range of each variable is explored efficiently, which is essential for capturing the variability in the solutions. We use LHS to sample points for validating the solutions:
for 1D problems

$$x_{\text{LHS}} = \{x_1, x_2, \ldots, x_M\}. \tag{3.18}$$

for 2D problems

$$(x_{\text{LHS}}, y_{\text{LHS}}) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}. \tag{3.19}$$

3.2.2 Data Preprocessing Steps

Before training the PINNs, the input features are normalized to a standard range to ensure uniformity in training dynamics. This step is crucial for maintaining stability and improving the convergence rate of the neural network. Boundary conditions are also encoded into the dataset to preserve the essential physical constraints during network training.

Chapter 4

Results and Analysis

This research paper explores the efficacy of Fibonacci sequence-based activation functions in Physics-Informed Neural Networks (PINNs) and compares it against traditional PINNs with tanh and sigmoid activations. Each model studied—ranging from fluid dynamics to wave propagation to electrostatics—has been solved using these three approaches to highlight the unique advantages of the Fibonacci activation in complex physical simulations.

4.1 Example 1: Poisson equation in 1D with Dirichlet boundary condition

$$u_{xx} = f(x) \tag{4.1}$$

This equation models the potential field generated by a given charge or mass distribution in a unidirectional scenario. The Poisson equation captures the relationship between the second

spatial derivative of the potential function $u(x)$ and the source term $f(x)$.

In our simulations, we apply three different approaches: a PINN with a sigmoid activation, a PINN with a tanh activation function, and a PINN enhanced with a Fibonacci sequence-based activation. Each method aims to address the challenges posed by the unidirectional aspects of the equation effectively. Then each PINN prediction is tested against the exact solution [17].

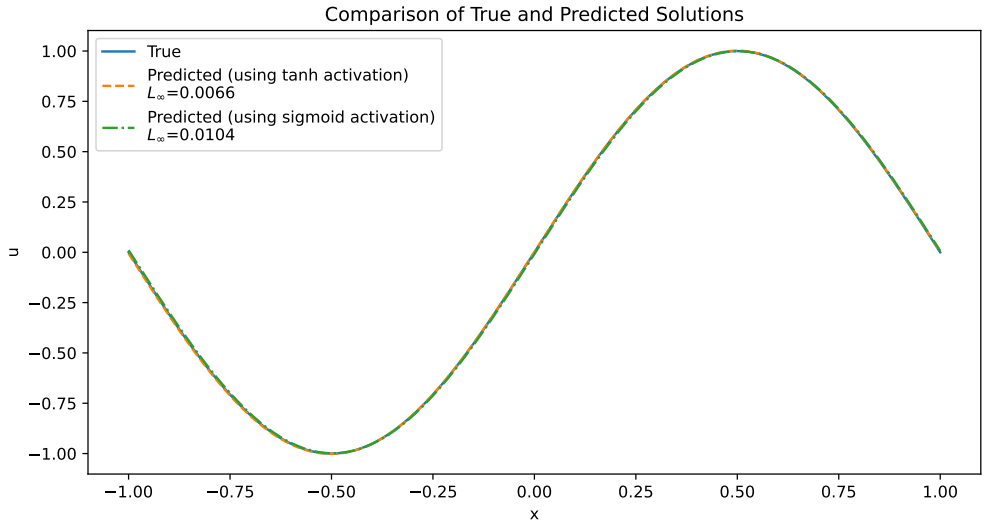

Figure 4.1: Poisson Equation with Classical Activation

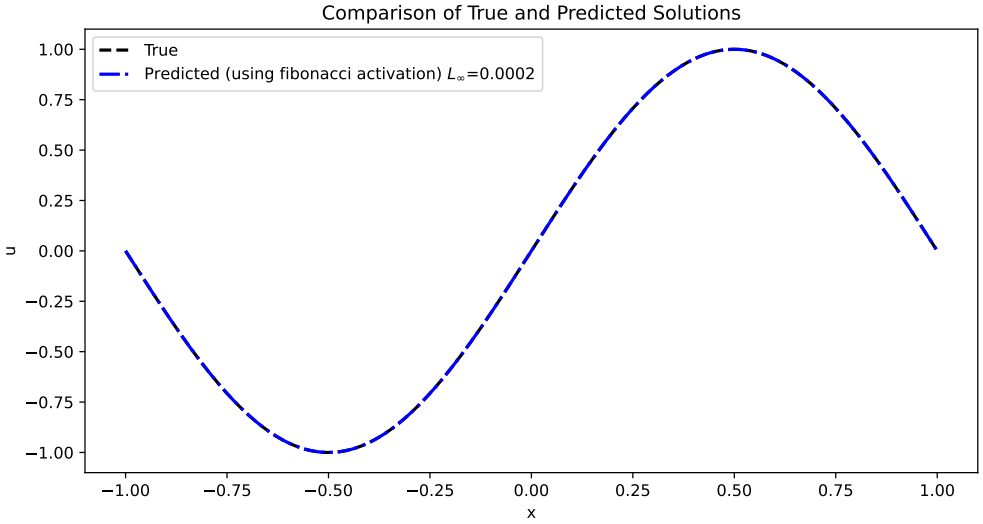

Figure 4.2: Poisson Equation with Fibonacci Activation

As seen in Figure 4.1, the tanh activation function performs better than the sigmoid activation function, with approximately 10 times lower error rates. Furthermore, as depicted in Figure 4.2, the Fibonacci activation function outperforms the tanh activation function by approximately 30 times, showcasing its superiority in accurately approximating solutions for the Poisson equation.

4.2 Example 2: Burgers' equation in 1D with Dirichlet boundary condition

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{4.2}$$

This Burger's equation models the behavior of a viscous fluid and incorporates both non-linear advection and diffusion terms.

The non-linear term $(u\frac{\partial u}{\partial x})$ represents internal convection, while the diffusion term $(\nu\frac{\partial^2 u}{\partial x^2})$ accounts for viscosity in the fluid.

The model is implemented using a PINN with sigmoid activation, a PINN with hyperbolic tangent (tanh) activation, and a PINN with a novel Fibonacci sequence-based activation. Each approach is iteratively trained and evaluated to understand its effectiveness in accurately simulating the dynamic behavior of the fluid and then tested against the approximate solution by VIM [18].

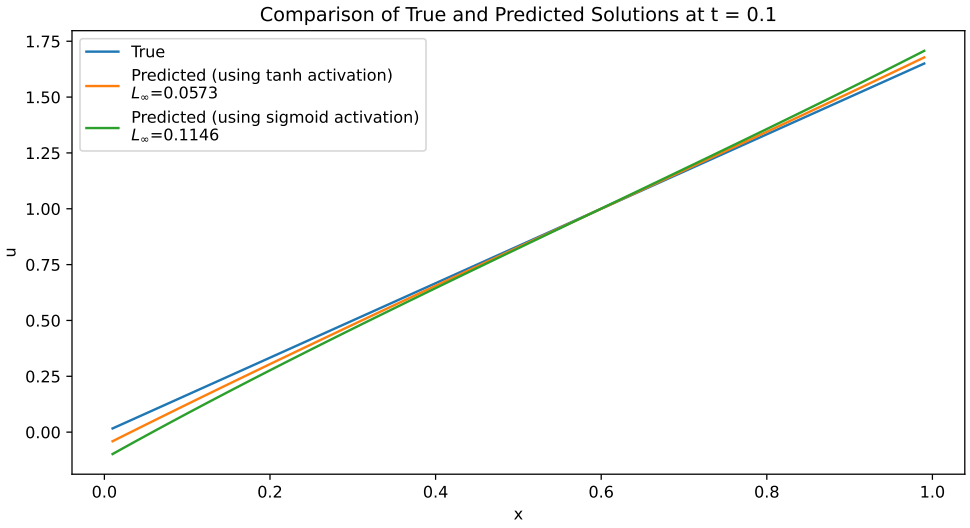

Figure 4.3: Burger's Equation at $t = 0.1$ with Classical Activations

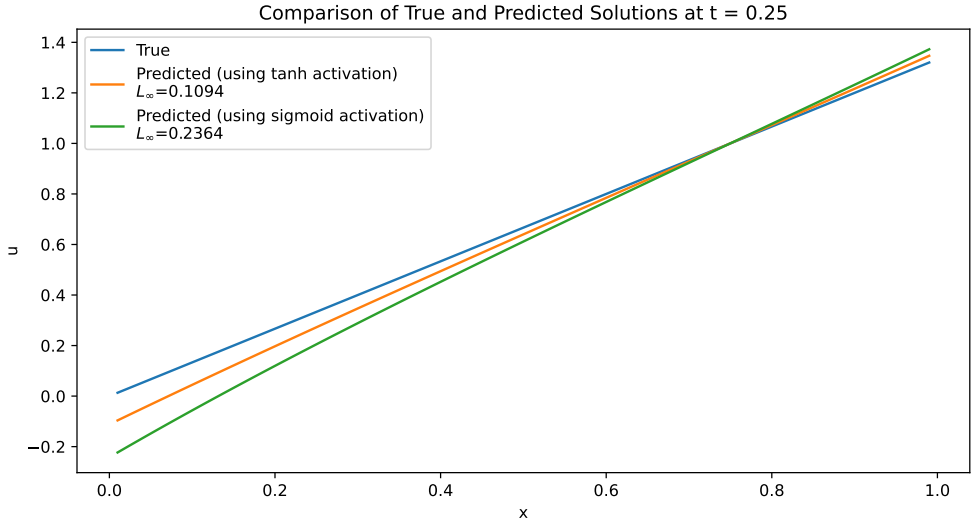

Figure 4.4: Burger's Equation at $t = 0.25$ with Classical Activations

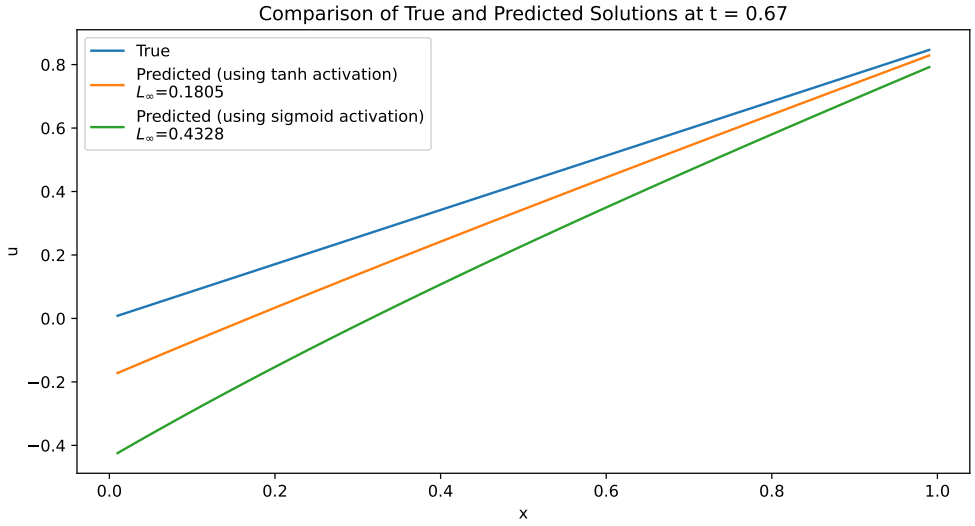

Figure 4.5: Burger's Equation at $t = 0.67$ with Classical Activations

As seen in the Figure 4.3, Figure 4.4 and Figure 4.5 the predictions made by the tanh activation function exhibit error rates that are consistently 5 to 10 times lower compared to sigmoid

activation.

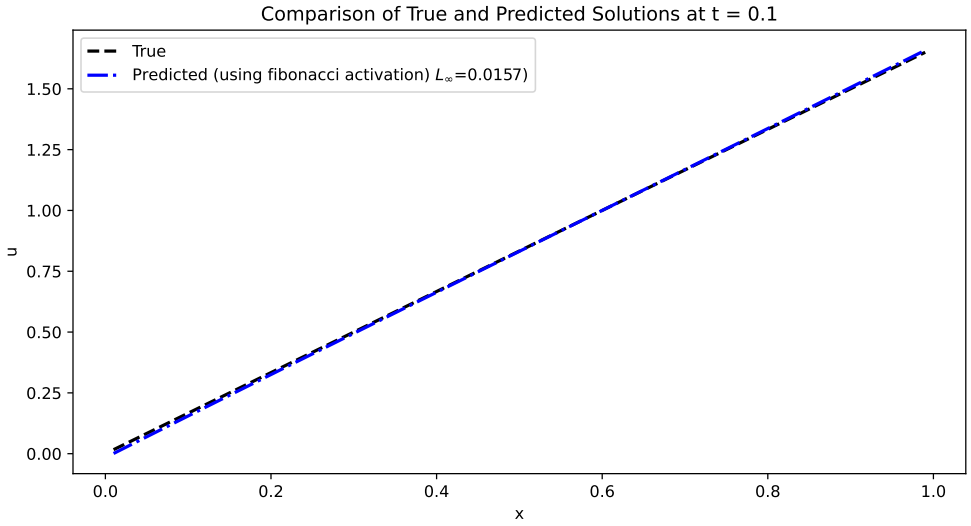

Figure 4.6: Burger's Equation at $t = 0.1$ with Fibonacci Activation

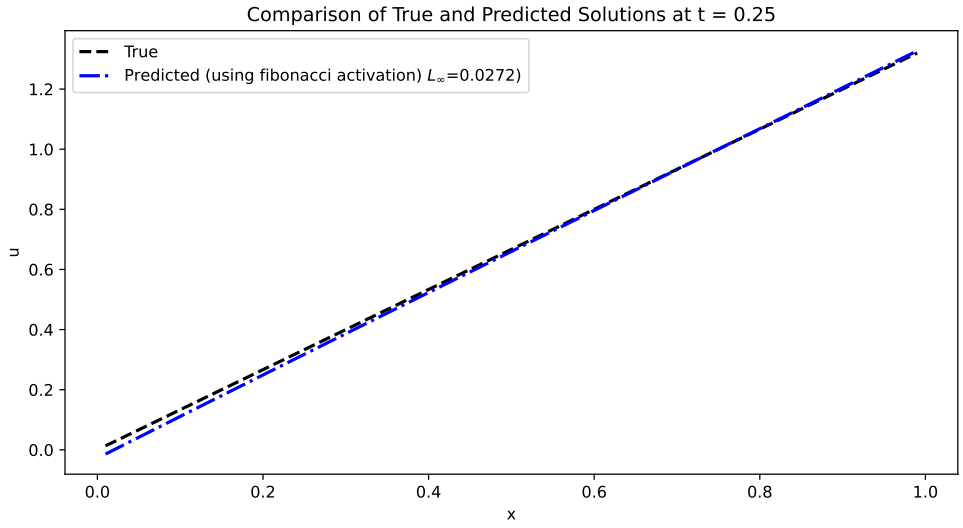

Figure 4.7: Burger's Equation at $t = 0.25$ with Fibonacci Activation

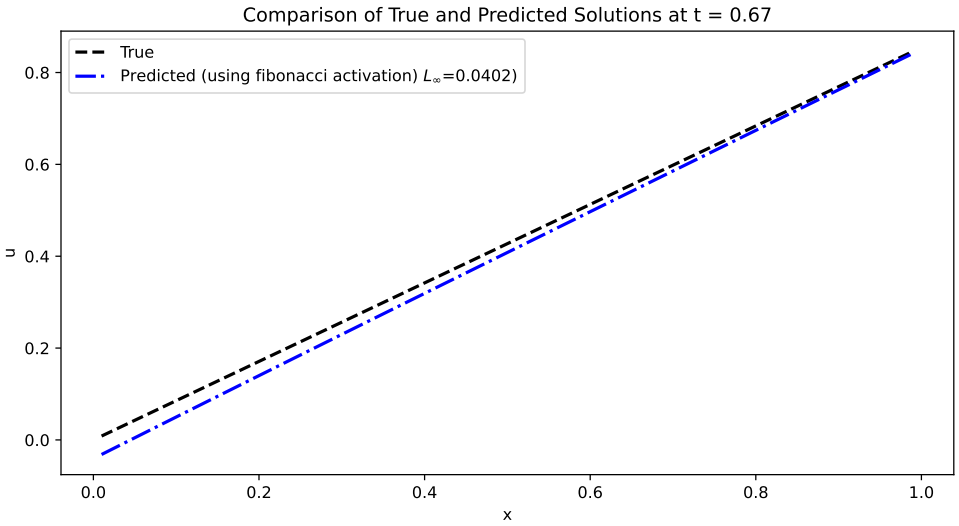

Figure 4.8: Burger's Equation at $t = 0.67$ with Fibonacci Activation

On the other hand, the Fibonacci activation plot for Burger's equation, depicted in Figure 4.6, Figure 4.7 and Figure 4.8 showcases the performance of the Fibonacci activation function at the same time instances (t = 0.1, 0.25, 0.67). As observed in the plots ,the predictions obtained using the Fibonacci activation function demonstrate error rates that are consistently more than 10 times lower than those achieved with the tanh activation function, highlighting the superior performance of Fibonacci activation in accurately predicting solutions for Burger's equation.

4.3 Example 3: Heat equation in 1D with variable time

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2} \tag{4.3}$$

The heat equation models the distribution of heat (or vari-

ation in temperature) in a given region over time. Here, α is the thermal diffusivity constant. This equation is fundamental in thermal analysis and can be used to predict temperature distribution in materials.

We employ three different approaches to solve this problem: a PINN with sigmoid activation, a PINN with tanh activation, and a PINN with Fibonacci sequence-based activation. Each method is tested against the exact solution [19] to evaluate their performance in simulating the temperature distribution over time.

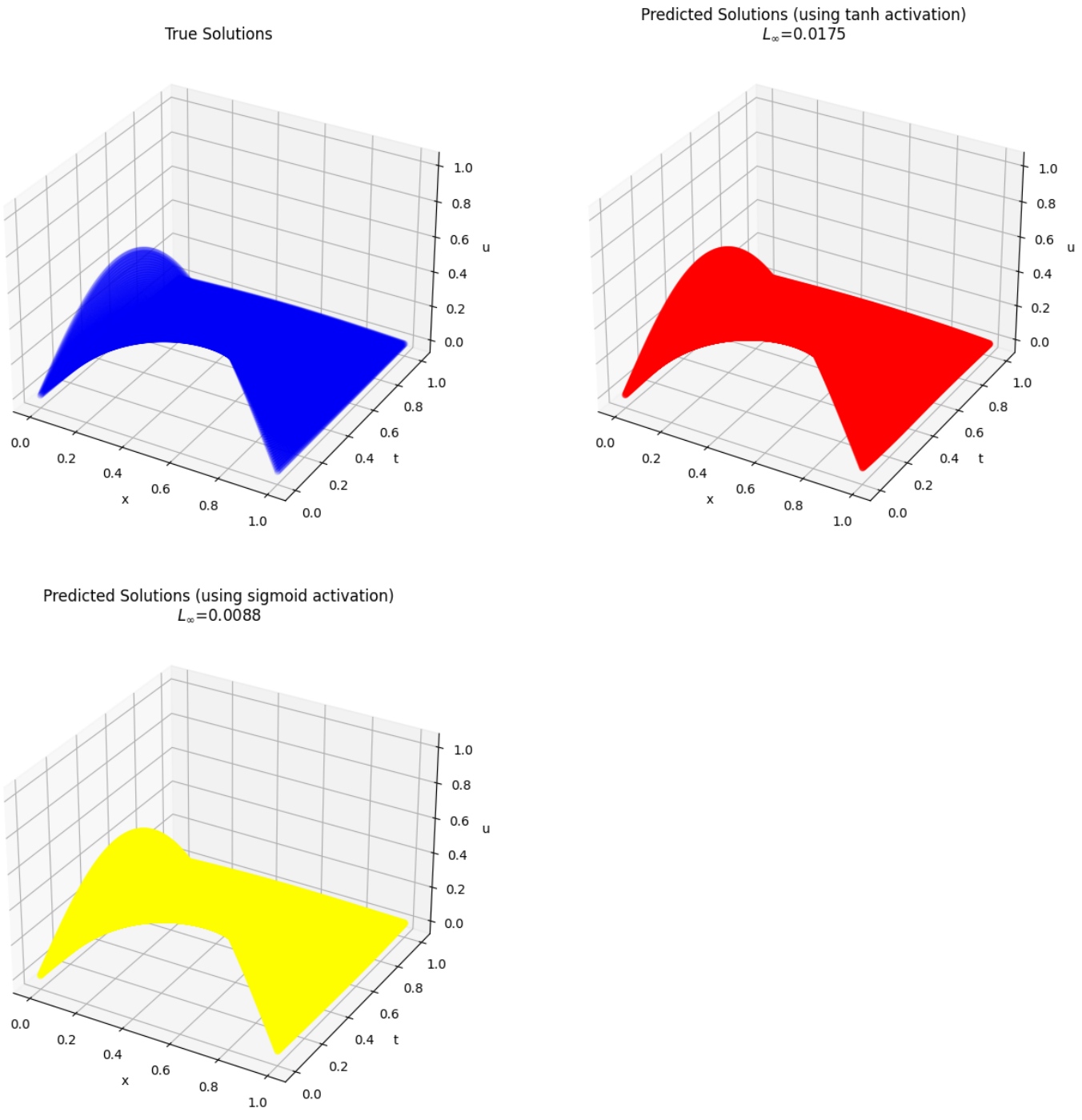

Figure 4.9: Heat Equation with Classical Activations

As observed in Figure 4.9, both tanh and sigmoid activations exhibit similar error rates, with sigmoid activation slightly outperforming tanh. This suggests that there may not be significant differences in performance between these two activation functions for the heat equation.

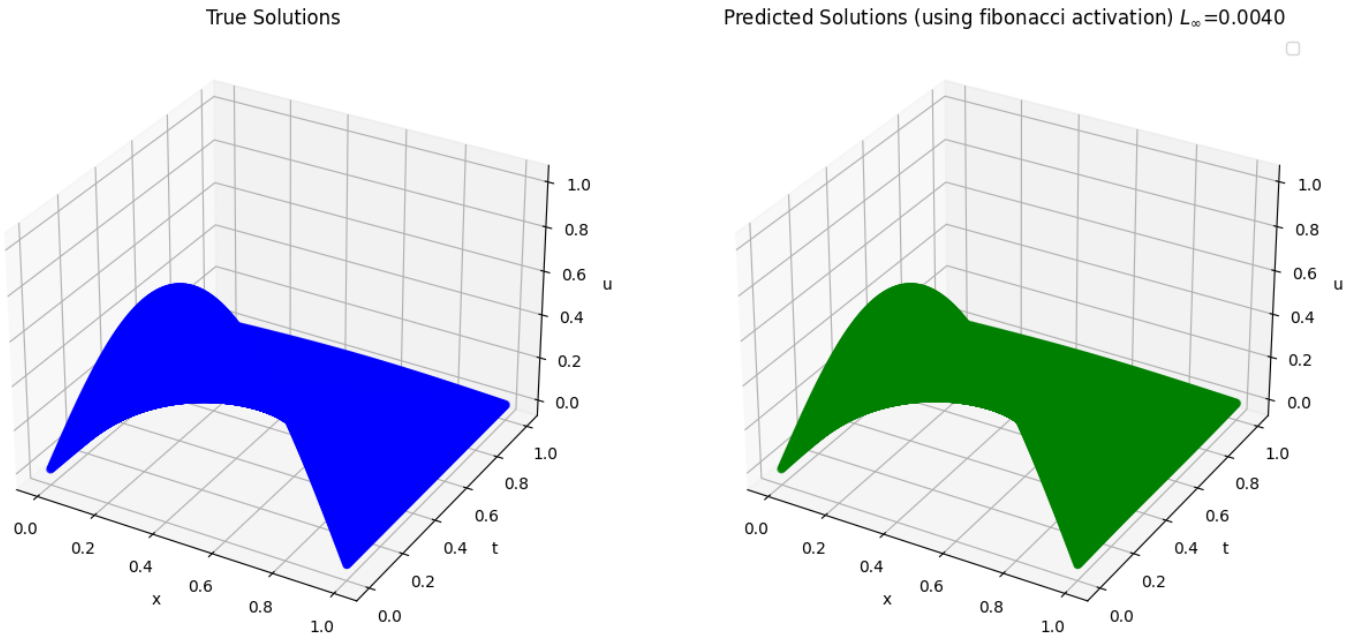

Figure 4.10: Heat Equation with Fibonacci Activation

On the other hand, Figure 4.10 showcases the predictions obtained using the Fibonacci activation function demonstrate error rates that are consistently 2 to 3 times lower than those achieved with tanh and sigmoid activations. However, it is worth noting that further study and experimentation are required to fully understand the potential of Fibonacci activation in improving the accuracy of PINNs for the heat equation.

4.4 Example 4: Helmholtz equation over a 2D square domain

$$\nabla^2 u + k^2 u = 0 \tag{4.4}$$

The Helmholtz equation arises in problems involving wave propagation, such as electromagnetic waves or sound waves, with k representing the wave number. This equation relates the Laplacian of the wave function u to its source function, modeling how waves propagate in a homogeneous medium.

To address this complex wave propagation scenario, we employ three distinct approaches: a Physics-Informed Neural Network (PINN) with sigmoid activation, a Physics-Informed Neural Network (PINN) with tanh activation, and a PINN utilizing a Fibonacci sequence-based activation. Each method is meticulously analyzed against the exact solution [20] to ascertain its effectiveness in simulating wave behavior accurately and efficiently.

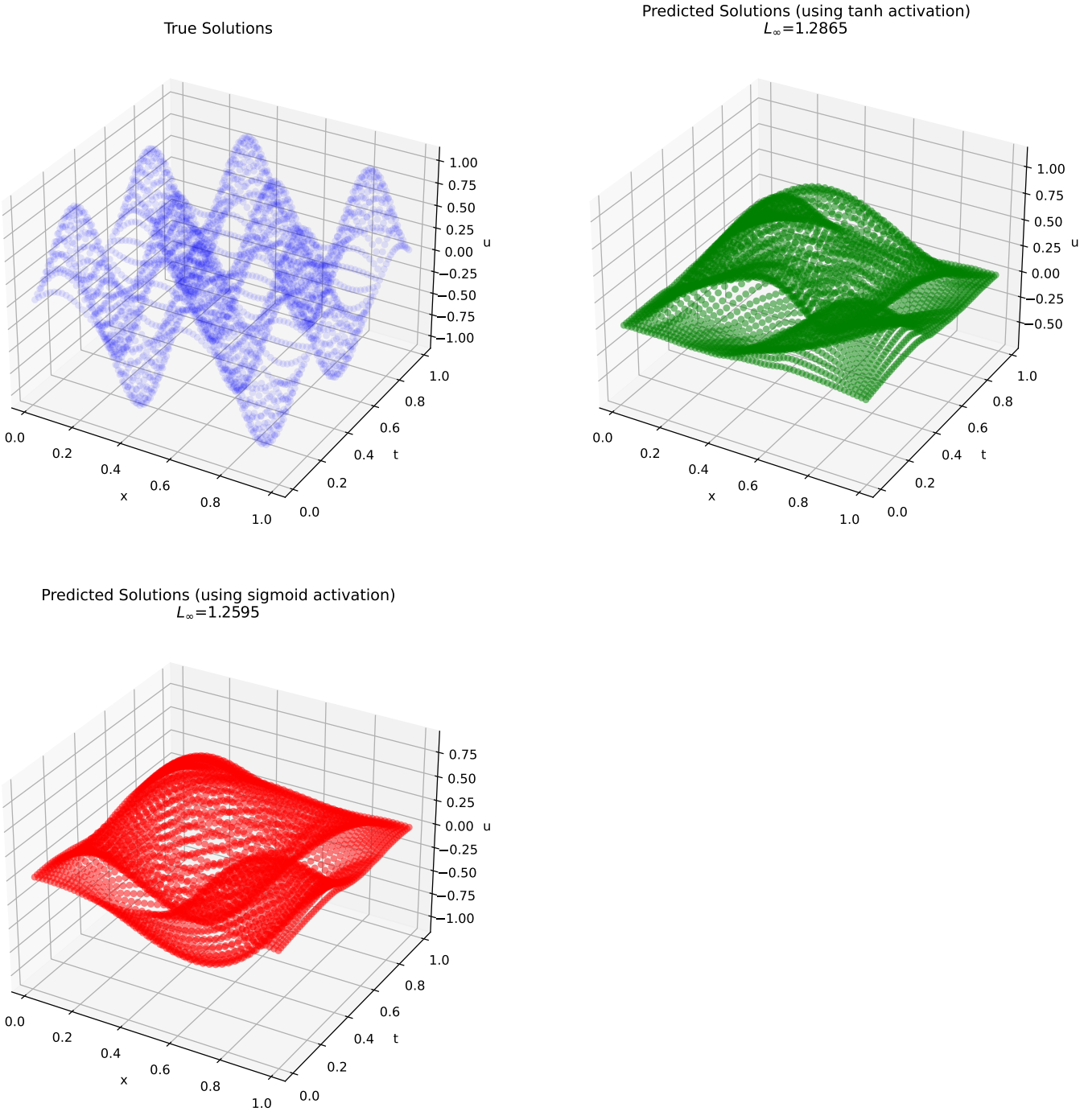

Figure 4.11: Helmholtz Equation with Classical Activations

The results for the heat equation highlight the limitations of classical activation functions such as tanh and sigmoid in accurately modeling complex equations like the 2D Helmholtz equation (see Figure 4.11). As observed in the Helmhotz plot, both tanh and sigmoid activations exhibit similar and very high error rates, indicating their inability to capture the intricacies of the equation.

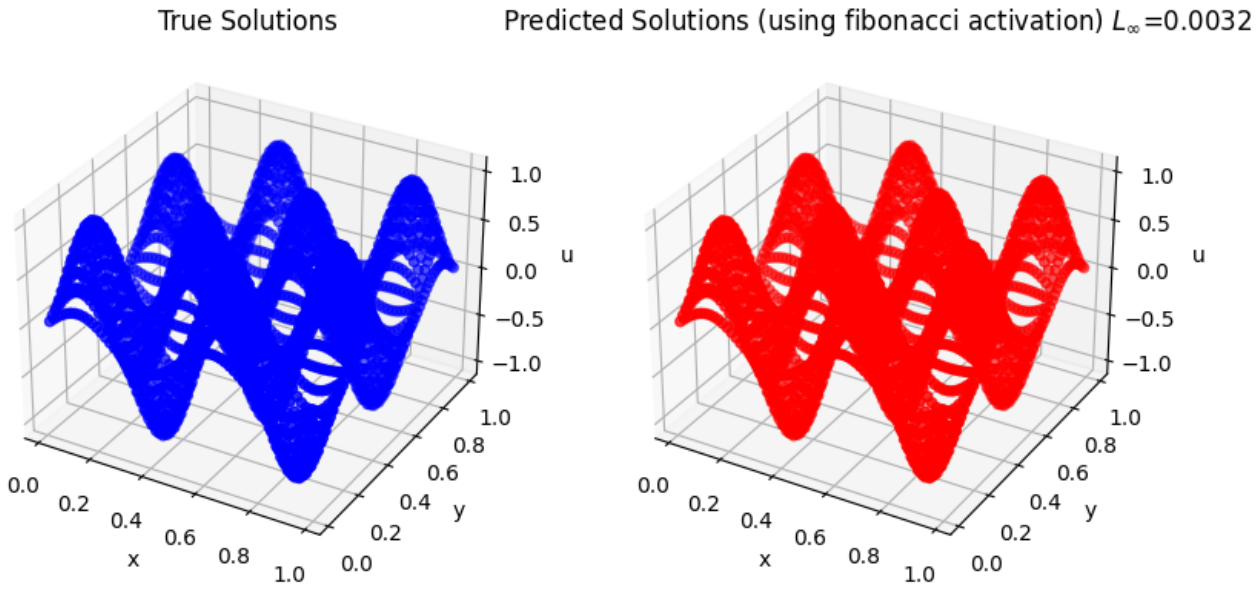

Figure 4.12: Helmholtz Equation with Fibonacci Activation

On the other hand, the Fibonacci activation function show-

cases remarkable performance (see Figure 4.12). The predictions obtained using Fibonacci activation demonstrate error rates that are approximately 100 times lower than those achieved with tanh and sigmoid activations.

This remarkable reduction in error rates suggests that Fibonacci activation can significantly enhance the accuracy of simulations involving wave propagation, providing critical improvements in fields such as acoustics and electromagnetic wave studies.

4.5 Example 5: Linear electrostatic equation over a 2D square domain

$$\nabla^2 \phi = -\rho \tag{4.5}$$

The linear electrostatic equation models the electric potential ϕ in a region with a given charge density ρ. This equation is fundamental in electrostatics and is used to describe how electric fields are generated by static charges.

We apply three different approaches to solve this problem: a PINN with sigmoid activation, a PINN with tanh activation, and a PINN with Fibonacci sequence-based activation. Each method is tested against the exact solution [21] to evaluate their performance in predicting the electric potential distribution.

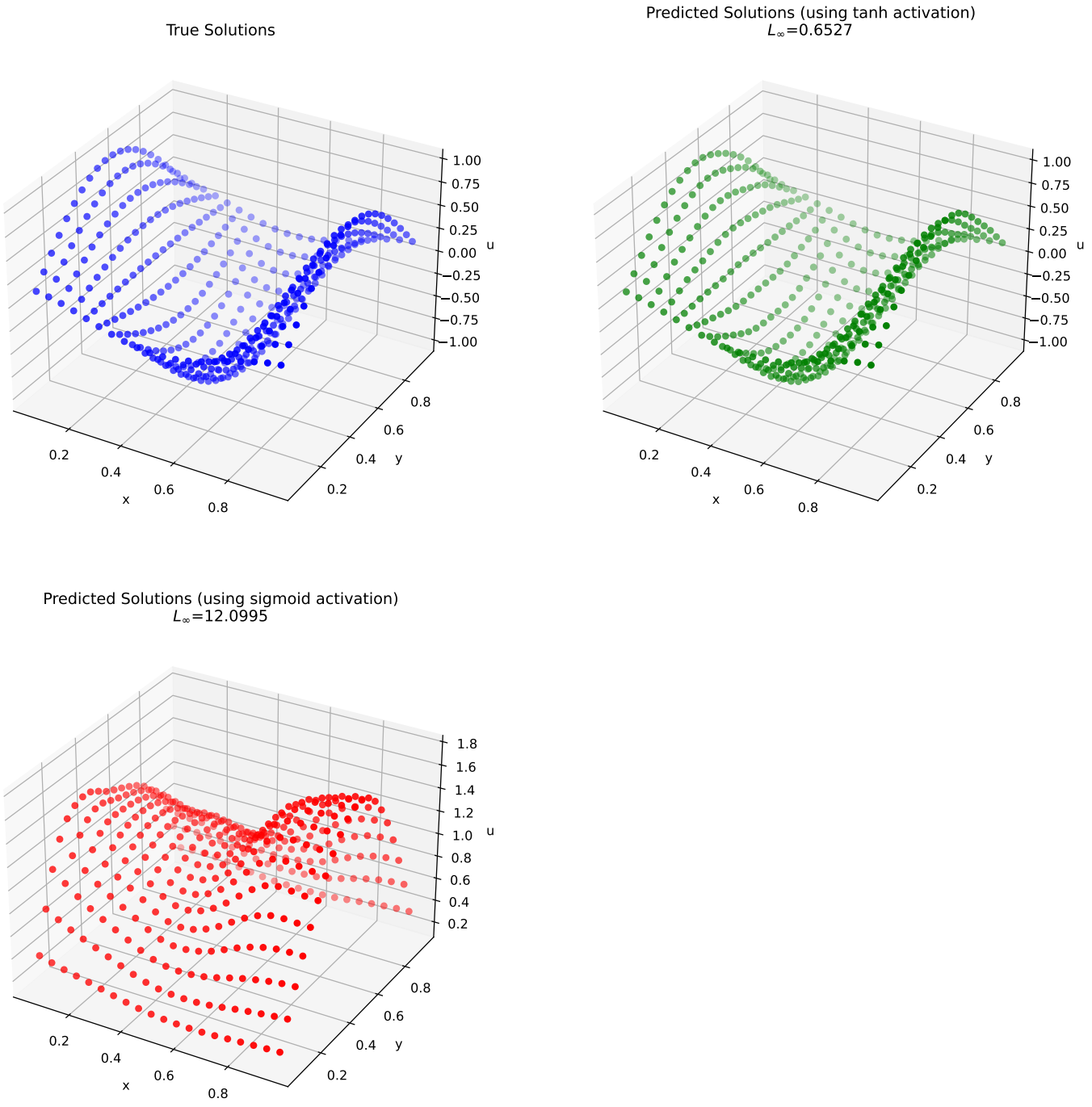

Figure 4.13: Elastostatic Equation with Classical Activation

As seen in the Elastostatic plot (Figure 4.13), the sigmoid activation function struggles to model the elastostatic equation, resulting in high error rates. Meanwhile, the tanh activation shows a significant improvement, with error rates approximately 20 times lower compared to sigmoid activation.

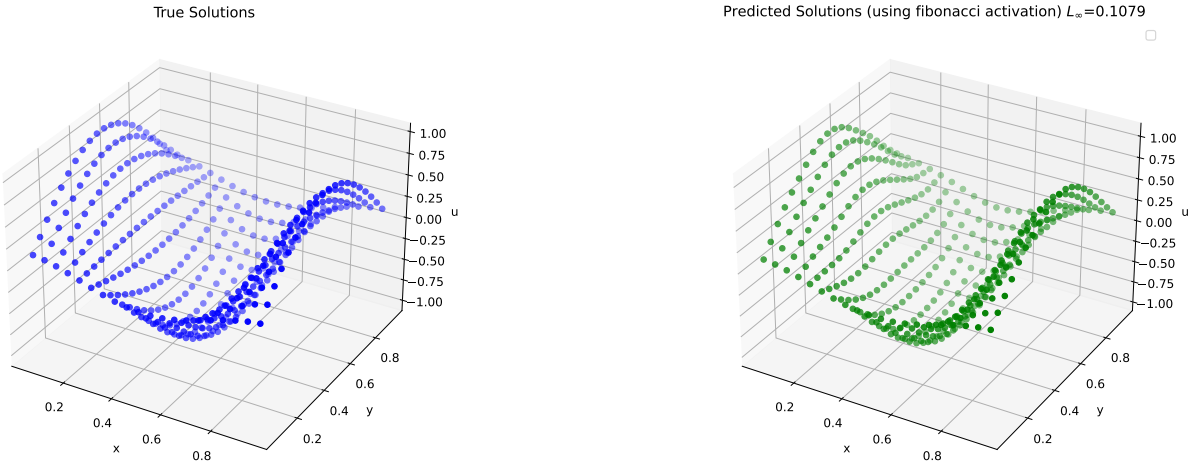

Figure 4.14: Elastostatic Equation with Fibonacci Activation

On the other hand, in the Elastostatic plot (Figure 4.14), the predictions obtained using the Fibonacci activation function demonstrate even lower error rates compared to both tanh and sigmoid activations, further highlighting its superiority in

accurately modeling the elastostatic equation. The Fibonacci activation show approximately 5-6 times lower error rates wrt. tanh activation.

This significant reduction in error rates with Fibonacci activation underscores its potential to enhance the accuracy of electrostatic simulations, making it a promising approach for applications in electric field modeling and design.

Chapter 5

Conclusions and Discussion

In this research, we investigated the efficacy of Fibonacci-based activation functions in Physics-Informed Neural Networks (PINNs) across various physical problems, including the 1D Poisson equation, 1D Burgers' equation, time-dependent 1D heat equation, 2D Helmholtz equation, and 2D elastostatic equation. Our analysis provides valuable insights into the performance of Fibonacci activation compared to the commonly used tanh and sigmoid activations in PINNs.

5.1 Discussion of Results

Our findings reveal interesting trends across the different equations studied. In general, tanh activation consistently outperforms sigmoid activation, except in the case of the heat equation where sigmoid show better results that tanh. However, the introduction of Fibonacci activation brings about significant im-

provements over the current preferred activation, tanh.

For the 2D Helmholtz equation, Fibonacci activation demonstrates a remarkable improvement, with error rates approximately 100 times lower than tanh activation. Similarly, for the 1D Poisson equation Fibonacci activation showcases substantial enhancements, exhibiting error reductions of approximately 30 times compared to tanh activation. Even for the 1D Burgers' equation and 2D elastostatic equation, Fibonacci activation exhibits mild yet notable improvements, with error reductions ranging from 5 to 10 times over tanh activation. And in the case of time-dependent 1D heat equation Fibonacci activation shows only 2-3 times lower error rates than tanh activation which shows that Fibonacci activation has potential to improve accuracy of PINNs but it need further research for some physical equations.

5.2 Implications

The significant improvements observed with Fibonacci activation underscore its potential to revolutionize the learning of physical laws in PINNs for a wide range of problems, including ordinary differential equations (ODEs), partial differential equations (PDEs), and functional differential equations (FDEs). By reducing error rates and enhancing accuracy, Fibonacci ac-

tivation can facilitate more reliable predictions and simulations, thereby advancing our understanding and application of physics-based machine learning models.

5.3 Future Directions

Moving forward, future research endeavors could delve deeper into understanding the underlying mechanisms behind the superior performance of Fibonacci activation in PINNs. Additionally, exploring the application of hybrid activation functions that combine Fibonacci activation with other activation functions could provide further insights and improvements. Furthermore, conducting large-scale experiments on real-world datasets and applications will be crucial to validate the scalability and generalizability of Fibonacci activation in practical settings.

5.4 Conclusion

In conclusion, our research demonstrates the transformative potential of Fibonacci-based activation functions in enhancing the accuracy and effectiveness of Physics-Informed Neural Networks. By significantly reducing error rates and improving learning capabilities across various physical problems, Fibonacci activation emerges as a promising avenue for advancing physics-based machine learning models and unlocking new opportuni-

ties for scientific discovery and innovation.

Appendix A

Analytical Solutions

A.1 Variational Iteration Method (VIM)

A.1.1 Solving 1D Burgers' Equation

The one-dimensional Burgers' equation given is:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}, \tag{A.1}$$

with the initial condition $u(x, 0) = 2x$.

The correction functional for VIM is applied as:

$$u_{n+1}(x,t) = u_n(x,t) - \int_0^t \left(\frac{\partial u_n}{\partial \tau} + u_n\frac{\partial u_n}{\partial x} - \nu\frac{\partial^2 u_n}{\partial x^2}\right)(x,\tau)d\tau. \tag{A.2}$$

The function converges to:

$$u(x,t) = \frac{2x}{1+2t}, \tag{A.3}$$

providing an exact solution for the specified initial condition.

A.2 Variable Separation Method

This classical method solves partial differential equations by assuming that each variable can be separated and solved independently [20].

A.2.1 Solving the time dependent Heat Equation with Dirichlet Boundary Condition

To solve the heat equation with Dirichlet boundary conditions and a sinusoidal initial condition, you can follow these steps[19]:

1. **Formulate the Problem**: The heat equation is given by:
$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$
with boundary conditions $u(0,t) = 0$ and $u(L,t) = 0$, and an initial condition $u(x,0) = \sin\left(\frac{n\pi x}{L}\right)$.

2. **Apply Separation of Variables**: Assume a solution of the form $u(x,t) = X(x)T(t)$. Substitute into the heat equation:
$$X(x)T'(t) = \alpha X''(x)T(t)$$
Divide both sides by $\alpha X(x)T(t)$ to separate the variables:
$$\frac{T'(t)}{\alpha T(t)} = \frac{X''(x)}{X(x)} = -\lambda$$

This gives two ordinary differential equations (ODEs):

$$T'(t) + \alpha\lambda T(t) = 0$$

$$X''(x) + \lambda X(x) = 0$$

3. **Solve the Spatial ODE**: The spatial ODE $X''(x) + \lambda X(x) = 0$ with boundary conditions $X(0) = 0$ and $X(L) = 0$ has non-trivial solutions for specific values of λ:

$$X_n(x) = \sin\left(\frac{n\pi x}{L}\right)$$

where $\lambda = \left(\frac{n\pi}{L}\right)^2$.

4. **Solve the Temporal ODE**: The temporal ODE $T'(t) + \alpha\lambda T(t) = 0$ has the solution:

$$T_n(t) = e^{-\alpha\left(\frac{n\pi}{L}\right)^2 t}$$

5. **Combine Solutions**: The general solution is a sum of solutions for different n:

$$u(x,t) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) e^{-\alpha\left(\frac{n\pi}{L}\right)^2 t}$$

6. **Apply Initial Conditions**: Given $u(x,0) = \sin\left(\frac{\pi x}{L}\right)$, we

find B_n such that:

$$u(x,0) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) = \sin\left(\frac{\pi x}{L}\right)$$

From this, we get $B_1 = 1$ and $B_n = 0$ for $n \neq 1$.

Thus, the exact solution is:

$$u(x,t) = \sin\left(\frac{\pi x}{L}\right) e^{-\alpha\left(\frac{\pi}{L}\right)^2 t}$$

A.2.2 Solving 1D Poisson Equation with Dirichlet Boundary Condition

We will solve a Poisson equation [17]:

$$-\Delta u = \pi^2 \sin(\pi x), \quad x \in [-1,1], \tag{A.4}$$

with the Dirichlet boundary conditions

$$u(-1) = 0, \quad u(1) = 0. \tag{A.5}$$

To solve this equation, we use the method of variable separation.

General Solution to the Differential Equation

The general solution to the inhomogeneous differential equation can be written as the sum of the particular solution and the

complementary (homogeneous) solution:

$$u(x) = u_p(x) + u_c(x). \tag{A.6}$$

Particular Solution ($u_p(x)$)

To find a particular solution, we assume a form similar to the non-homogeneous term:

$$u_p(x) = A\sin(\pi x). \tag{A.7}$$

Substituting $u_p(x)$ into the Poisson equation:

$$-u_p''(x) = -A\pi^2\sin(\pi x) = \pi^2\sin(\pi x). \tag{A.8}$$

This implies:

$$-A\pi^2 = \pi^2 \implies A = -1. \tag{A.9}$$

Thus, the particular solution is:

$$u_p(x) = -\sin(\pi x). \tag{A.10}$$

Complementary Solution ($u_c(x)$)

The complementary solution solves the homogeneous equation:

$$u_c''(x) = 0. \tag{A.11}$$

The general solution to this is:

$$u_c(x) = C_1 x + C_2. \tag{A.12}$$

General Solution

Combining the particular and complementary solutions:

$$u(x) = -\sin(\pi x) + C_1 x + C_2. \tag{A.13}$$

Applying Boundary Conditions

Using the boundary conditions to find C_1 and C_2:

$$u(-1) = 0 \implies -\sin(-\pi)+C_1(-1)+C_2 = 0 \implies 0 = C_1(-1)+C_2 \implies \tag{A.14}$$

$$u(1) = 0 \implies -\sin(\pi)+C_1(1)+C_2 = 0 \implies 0 = C_1(1)+C_2 \implies C_2 = \tag{A.15}$$

From the above equations:

$$C_1 = -C_1 \implies C_1 = 0. \tag{A.16}$$

Therefore, C_2 must also be zero:

$$C_2 = 0. \tag{A.17}$$

Final Solution

Substituting $C_1 = 0$ and $C_2 = 0$ back into the general solution:

$$u(x) = -\sin(\pi x). \tag{A.18}$$

Since the boundary conditions are satisfied, the exact solution is:

$$u(x) = \sin(\pi x). \tag{A.19}$$

Therefore, the exact solution to the 1D Poisson equation $-\Delta u = \pi^2 \sin(\pi x)$ with the given Dirichlet boundary conditions $u(-1) = 0$ and $u(1) = 0$ is:

$$u(x) = \sin(\pi x). \tag{A.20}$$

A.2.3 Solving 2D Helmholtz Equation over 2D Square Domain

The 2D Helmholtz equation is given by:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + k^2 u = 0. \tag{A.21}$$

Applying the method of separation of variables:

Assume $u(x, y) = X(x)Y(y)$ and substitute into the Helmholtz equation:

$$X''(x)Y(y) + X(x)Y''(y) + k^2 X(x)Y(y) = 0. \tag{A.22}$$

This equation separates into:

$$\frac{X''(x)}{X(x)} + \frac{Y''(y)}{Y(y)} = -k^2. \tag{A.23}$$

Solving the resulting ordinary differential equations with boundary conditions:

$$X(x) = A\sin(\sqrt{\lambda}x), \quad Y(y) = B\sin(\sqrt{\mu}y),$$

where $\lambda + \mu = k^2$.

The exact solutions are formulated as:

$$u(x,y) = \sum_{n,m} C_{n,m}\sin(n\pi x)\sin(m\pi y). \tag{A.24}$$

A.3 Exact Solution of Linear Elastostatic Equation over 2D Square Domain

For solving the linear elastostatic equation over a 2D square domain. The governing equations under the linear elastic constitutive model are given by[21]:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y} + f_x = 0, \tag{A.25}$$

$$\frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y} + f_y = 0, \tag{A.26}$$

where the domain is defined as $x \in [0,1]$ and $y \in [0,1]$. The

stress components σ_{xx}, σ_{yy}, and σ_{xy} are related to the strains ϵ_{xx}, ϵ_{yy}, and ϵ_{xy} through the relations:

$$\sigma_{xx} = (\lambda + 2\mu)\epsilon_{xx} + \lambda\epsilon_{yy}, \tag{A.27}$$

$$\sigma_{yy} = (\lambda + 2\mu)\epsilon_{yy} + \lambda\epsilon_{xx}, \tag{A.28}$$

$$\sigma_{xy} = 2\mu\epsilon_{xy}, \tag{A.29}$$

with the kinematic relations:

$$\epsilon_{xx} = \frac{\partial u_x}{\partial x}, \tag{A.30}$$

$$\epsilon_{yy} = \frac{\partial u_y}{\partial y}, \tag{A.31}$$

$$\epsilon_{xy} = \frac{1}{2}\left(\frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x}\right). \tag{A.32}$$

The body forces f_x and f_y are given by:

$$f_x = \lambda\left[4\pi^2\cos(2\pi x)\sin(\pi y) - \pi\cos(\pi x)Qy^3\right] \tag{A.33}$$

$$+ \mu\left[9\pi^2\cos(2\pi x)\sin(\pi y) - \pi\cos(\pi x)Qy^3\right], \tag{A.34}$$

$$f_y = \lambda\left[-3\sin(\pi x)Qy^2 + 2\pi^2\sin(2\pi x)\cos(\pi y)\right] \tag{A.35}$$

$$+ \mu\left[-6\sin(\pi x)Qy^2 + 2\pi^2\sin(2\pi x)\cos(\pi y) + \frac{\pi^2}{4}\sin(\pi x)Qy^4\right]. \tag{A.36}$$

The boundary conditions are displacement constraints:

$$u_x(x,0) = u_x(x,1) = 0, \tag{A.37}$$

$$u_y(0,y) = u_y(1,y) = u_y(x,0) = 0, \tag{A.38}$$

and traction boundary conditions:

$$\sigma_{xx}(0,y) = 0, \tag{A.39}$$

$$\sigma_{xx}(1,y) = 0, \tag{A.40}$$

$$\sigma_{yy}(x,1) = (\lambda + 2\mu)Q\sin(\pi x). \tag{A.41}$$

The material parameters are set as $\lambda = 1$, $\mu = 0.5$, and $Q = 4$. The exact solution to this problem is:

$$u_x(x,y) = \cos(2\pi x)\sin(\pi y), \tag{A.42}$$

$$u_y(x,y) = \sin(\pi x)\frac{Qy^4}{4}. \tag{A.43}$$

Appendix B

Experimental Setup

B.1 Environment and Libraries

Experiments were conducted using Google Colab, which provides access to high-performance GPUs, specifically the Tesla K80 GPU. The software environment included:

- **Python** version 3.9
- **TensorFlow** version 2.12.0 for building and training neural networks
- **DeepXDE** version 1.6.0 for implementing Physics-Informed Neural Networks (PINNs)
- **Matplotlib** version 3.6.2 for data visualization

B.2 Code Availability

The complete source code for the experiments, including data preprocessing, model implementation, training, and evaluation

scripts, is available at the following GitHub repository: `https://github.com/astromanish/fibonacci-PINNs`. The experiments were executed in a Jupyter notebook environment, which is accessible on Google Colab, providing an interactive platform for running and modifying the code.

B.3 Dependencies and Installation

To replicate the experiments, the following steps should be followed to set up the environment:

```
# Clone the repository locally
git clone https://github.com/astromanish/fibonacci-PINNs
cd fibonacci-PINNs

# Install the required libraries
pip install -r requirements.txt
```

The 'requirements.txt' file includes all necessary dependencies with their respective versions to ensure compatibility.

B.4 Hyperparameters

The model was configured with the following hyperparameters:

- **Optimizer**: Adam optimizer with a learning rate of 0.001
- **Training Epochs**: 10,000 epochs

- **Batch Size**: 32

B.5 Reproducibility Checklist

To ensure the reproducibility of the results, the following measures were taken:

- **Random Seed**: Random seeds were set to 42 for both NumPy and TensorFlow to ensure consistent results across different runs.
- **Runtime**: The training process took approximately 30 minutes on average on Google Colab, leveraging the computational power of the Tesla K80 GPU.

Bibliography

[1] 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, 2019.

[2] G. Pang, L. Lu, and G. Karniadakis, "fpinns: Fractional physics-informed neural networks," *SIAM Journal on Scientific Computing*, vol. 41, pp. A2603–A2626, 2018.

[3] F. S. Costabal, S. Pezzuto, and P. Perdikaris, "delta-pinns: physics-informed neural networks on complex geometries," 2022.

[4] Z. K. Lawal, H. Yassin, D. Lai, and A. Che-Idris, "Physics-informed neural network (pinn) evolution and beyond: A systematic literature review and bibliometric analysis," *Big Data and Cognitive Computing*, vol. 6, p. 140, 2022.

[5] A. Kovacs, L. Exl, A. Kornell, J. Fischbacher, M. Hovorka, M. Gusenbauer, L. Breth, H. Oezelt, M. Yano, N. Sakuma,

A. Kinoshita, T. Shoji, A. Kato, and T. Schrefl, "Conditional physics informed neural networks," 2021.

[6] B. Shan, Y. Li, and S.-J. Huang, "Vi-pinns: Variance-involved physics-informed neural networks for fast and accurate prediction of partial differential equations," 2022.

[7] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks for heat transfer problems," *Journal of Heat Transfer*, 2021.

[8] B. Huang and J. Wang, "Applications of physics-informed neural networks in power systems - a review," *IEEE Transactions on Power Systems*, vol. 38, pp. 572–588, 2023.

[9] L. S. d. Oliveira, L. Kunstmann, D. Pina, D. d. Oliveira, and M. Mattoso, "Pinnprov: Provenance for physics-informed neural networks," in *2023 International Symposium on Computer Architecture and High Performance Computing Workshops (SBAC-PADW)*, 2023, pp. 16–23.

[10] S. Markidis, "On physics-informed neural networks for quantum computers," *Frontiers in Applied Mathematics and Statistics*, vol. 8, 2022.

[11] V. Kunc and J. Kléma, "Three decades of activations: A comprehensive survey of 400 activation functions for neural networks," *arXiv preprint arXiv:2402.09092*, 2024.

[12] S. R. Dubey, S. K. Singh, and B. B. Chaudhuri, "Activation functions in deep learning: A comprehensive survey and benchmark," *arXiv preprint arXiv:2109.14545*, 2021.

[13] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," *arXiv preprint arXiv:1811.03378*, 2018.

[14] Encyclopaedia Britannica, "Fibonacci sequence," https://www.britannica.com/science/Fibonacci-number, 2024.

[15] J. L. Herrera, J. J. Bravo, and C. A. Gómez, "Curious generalized fibonacci numbers," *Mathematics*, vol. 9, no. 20, p. 2588, 2021.

[16] W.-L. Loh, "On latin hypercube sampling," *Annals of Statistics*, vol. 24, no. 5, pp. 2058–2080, 1996.

[17] W. Zhu, Z. Huang, J. Chen, and Y.-W. Chang, "Analytical solution of poisson's equation with application to vlsi global placement," *arXiv preprint arXiv:2307.12041*, 2023. [Online]. Available: https://arxiv.org/pdf/2307.12041

[18] M. A. Abdou and A. A. Soliman, "Variational iteration method for solving burger's and coupled burger's equations," *Journal of Computational and Applied Mathematics*, vol. 181, pp. 245–251, 2005.

[19] J. Feldman, "Solution of the heat equation by separation of variables," *University of British Columbia*, 2023, accessed: 2024-05-29.

[20] R.-E. Plessix and W. Mulder, "Separation-of-variables as a preconditioner for an iterative helmholtz solver," *Applied Numerical Mathematics*, vol. 44, pp. 385–400, 2003.

[21] D. D. Team, "Linear elastostatic equation over a 2d square domain," https://deepxde.readthedocs.io/en/latest/demos/pinn_forward/elasticity.plate.html, 2024, accessed: 2024-05-29.