Ministry of Science and Higher Education of the Republic of Kazakhstan

SDU University

Batyr Sharimbayev

Development and optimization of physics-informed neural networks for solving partial differential equations

THESIS

Presented in Partial Fulfillment for the
Degree of Master of Science in Mathematics
(degree code: 7M05401)
Department of Mathematics and Natural Sciences
Faculty of Engineering and Natural Sciences

Supervisor: **Shirali Kadyrov, PhD**

Kaskelen, June 2025

SDU University
Faculty of Engineering and Natural Sciences
Department of Mathematics and Natural Sciences

Dean of Faculty of Engineering and Natural Sciences

Assistant Professor, Ph.D. Akhmedov Ramis

_____

« 18 » 06 2025

Topic of the thesis:

Development and optimization of physics-informed neural networks for solving partial differential equations.

Thesis submitted as part of the requirements for the award of the MSc in "7M05401-Mathematics"

Head of Department Beklolat Bayro, PhD

Academic Supervisor Kadyrov Shirali, PhD

Master student Sharimbayev Batyr

Kaskelen, 2025

Declaration

I hereby declare that this dissertation is my own original work. Any materials from other sources have been properly acknowledged.

Batyr Sharimbayev

June 2025

Acknowledgements

I'm very thankful to my supervisor, Dr. Shirali Kadyrov, for his guidance, insightful feedback, and continuous encouragement throughout the course of this research. I am equally grateful of Aleksei Kavokin's support for his mentorship and thoughtful advice, which greatly enriched my academic development. Most importantly, I extend my deepest appreciation to my beloved mother, Bakhyt Pirnazarovna, whose unconditional love, unwavering support, and enduring strength have been the cornerstone of all my achievements. Her sacrifices and encouragement have made this journey possible.

Abstract

This thesis talks about using physics-informed neural networks (PINNs) to solve Poisson equations in both one-dimensional and two-dimensional areas. These equations are common in many physical problems, like heat transfer and electrostatics. The results from PINNs are compared to the finite difference method (FDM), which is a classical numerical method often used to solve these kinds of equations. The study shows that PINNs can give results that are close to those from FDM, with the added benefit of being more flexible for different types of problems. Another part of this work focuses on using multi-task learning with PINNs. In this part, the neural network does more than one job. It not only finds the solution of the differential equation, but it also learns unknown values or parameters that are part of the equation. For example, in one test problem, the equation had a source term and a coefficient that changes depending on the position. The PINN was able to learn both of them correctly while still solving the equation with a low training error. The results show that PINNs can work well even when the equation is more complex or has unknown parts. The model showed good performance on new or unseen data and was able to find the correct hidden values in the system. Because of this, PINNs may be very useful in future applications for solving advanced problems in science and engineering, especially where traditional methods might be harder to use.

Аңдатпа

Бұл диссертацияда Пуассон теңдеулерін бір өлшемді және екі өлшемді аймақтарда шешу үшін физикадан хабардар нейрондық желілерді (PINN) пайдалану туралы айтылады. Бұл теңдеулер жылу беру және электростатика сияқты көптеген физикалық мәселелерде жиі кездеседі. PINN нәтижелері осы теңдеу түрлерін шешу үшін жиі қолданылатын классикалық сандық әдіс болып табылатын соңғы айырмашылық әдісімен (FDM) салыстырылады. Зерттеу көрсеткендей, PINN FDM нәтижелеріне жақын нәтижелер бере алады, сонымен қатар проблемалардың әртүрлі түрлеріне икемді болуының қосымша артықшылығы бар. Бұл жұмыстың тағы бір бөлігі PINN көмегімен көп тапсырманы оқытуды пайдалануға бағытталған. Бұл бөлімде нейрондық желі бірнеше жұмысты орындайды. Ол дифференциалдық теңдеудің шешімін тауып қана қоймайды, сонымен қатар белгісіз мәндерді немесе теңдеудің бөлігі болып табылатын параметрлерді үйренеді. Мысалы, бір сынақ есебінде теңдеудің бастапқы мүшесі мен позицияға байланысты өзгеретін коэффициенті болды. Төмен жаттығу қатесі бар теңдеуді шешу кезінде PINN екеуін де дұрыс үйрене алды. Нәтижелер PINN теңдеу күрделірек немесе белгісіз бөліктері бар болса да жақсы жұмыс істей алатынын көрсетеді. Модель жаңа немесе көрінбейтін деректерде жақсы өнімділікті көрсетті және жүйеде дұрыс жасырын мәндерді таба алды. Осы себепті, PINN ғылым мен техникадағы озық мәселелерді шешуге арналған болашақ қолданбаларда өте пайдалы болуы мүмкін, әсіресе дәстүрлі әдістерді пайдалану қиынырақ болуы мүмкін.

Аннотация

В этой работе говорится об использовании нейронных сетей с физической информацией (PINN) для решения уравнений Пуассона как в одномерных, так и в двумерных областях. Эти уравнения распространены во многих физических задачах, таких как теплопередача и электростатика. Результаты PINN сравниваются с методом конечных разностей (FDM), который является классическим численным методом, часто используемым для решения таких уравнений. Исследование показывает, что PINN могут давать результаты, близкие к результатам FDM, с дополнительным преимуществом большей гибкости для различных типов задач. Другая часть этой работы сосредоточена на использовании многозадачного обучения с PINN. В этой части нейронная сеть выполняет более одной работы. Она не только находит решение уравнения, но и изучает неизвестные значения или параметры, которые являются частью уравнения. Например, в одной тестовой задаче уравнение имело исходный член и коэффициент, который изменяется в зависимости от положения. PINN смогла правильно изучить оба из них, при этом по-прежнему решая уравнение с низкой ошибкой обучения. Результаты показывают, что PINN могут хорошо работать даже в случае, когда уравнение более сложное или имеет неизвестные части. Модель показала хорошую производительность на новых и неизвестных данных и смогла найти правильные скрытые значения в системе. Благодаря этому PINN могут быть очень полезны в будущих приложениях для решения сложных задач в науке и технике, особенно там, где традиционные методы могут быть сложнее в использовании.

Table of Contents

1. Introduction

Partial Differential Equations (PDEs) are of significant relevance in a variety of branches in engineering and science. They provide vital insights in complex systems [1]. In fluid dynamics [2], for instance, they are used in modeling and representing fluid behavior in certain conditions as tools for flow and turbulence phenomena analysis. In heat transfer problems [3], they represent diffraction in liquids and solids in time. In financial modeling [4], they enable financial market conditions related computation and implications, financial derivative and option values, and in biomedical engineering [5], they are applied in modeling tissue movement and are a key component in drug delivery systems for improving biological processes comprehension. In addition, in materials science [6], they are as equally vital in predicting how applied force causes a material to deform and alter shape.

The numerical techniques applied for ordinary differential equations (ODE) and PDE solutions include a vast variety of techniques, some of which are the finite difference method (FDM) [7], finite element method (FEM) [8], finite volume method (FVM) [9], spectral method [10], and characteristic method [11]. The techniques are applied for approximating solutions for complex PDEs since solutions may be impractical in some cases when derived using analytical techniques. Each method is suitable for addressing different problems, for example, irregular geometries, conservation laws, or requiring high accuracy. By providing sound frameworks for computational approximations, techniques improve analysis and simulation in a vast variety of phenomena in scientific and engineering fields.

FDM is a computational method for the solution of PDEs based on approximating derivatives in terms of finite differences. Substitute continuous derivatives in place with algebraic formulas involving values at finite locations on a computational grid. FDM can be classified as explicit, implicit, or schemes of higher order depending on the form, and each possesses distinct stability and accuracy characteristics. FDM is typically applied in time-dependent and spatially inhomogeneous problems and is widely utilized in fluid dynamics [12], heat transfer [13], and wave propagation [14]. Its relative simplicity and easy programming and structured grid constraints make it best suited for problems in regular domains. Stability constraints, numerical dispersion, and complexity in dealing with complex

geometries are some of the problems in application in complex cases.

Deep learning techniques have emerged as viable strategies for solving PDEs [15], mitigating some typical problems associated with regular numerical solvers, including computational expenses, meshing problems, and limitations in dealing with high-dimensional or irregular problems. Along this line, techniques like Physics-Informed Neural Networks (PINNs) [16] and Convolutional Neural Networks (CNNs) [17] use neural networks to represent solutions while imposing physical constraints and functions at the same time, thus allowing efficient surrogate modeling, real-time simulation, and solutions to inverse problems. The main advantage lying in these techniques is their capability for scalability, flexibility in dealing with complex geometries, and ability to incorporate experimental or sparse information. However, some limitations come with their use, including high training expenses, need for extensive databases or expert architectures, and possible problems in achieving computational stability and accuracy.

PINNs offer an innovative approach to solving PDEs by integrating the governing equations [16]. Unlike traditional numerical methods, PINNs leverage automatic differentiation to compute gradients. This mesh-free approach allows PINNs to effectively handle complex geometries and multi-scale phenomena while incorporating experimental data to enhance predictions. While they present challenges related to architecture selection, convergence, and computational resources, PINNs provide a powerful tool for solving inverse problems [18].

1.1 Problem statement

The accurate numerical solution of PDEs is essential in various fields of science. Traditional numerical methods have been widely used for solving PDEs. However, these methods can be computationally expensive and may require significant discretization efforts, particularly for high-dimensional problems [19].

By using deep learning to solve PDEs without the necessity for explicit mesh formation, PINNs have become a viable substitute. PINNs exhibit significant versatility and data efficiency by integrating controlling physics equations into the loss function and use neural networks to approximate answers. However, when compared to conventional numerical techniques, PINNs frequently suffer from issues with accuracy, convergence, and generalization, despite their potential.

1.2 Research objectives

The first objective of this research is to rigorously evaluate the performance of PINNs in solving PDEs, with a specific focus on comparing the accuracy, convergence, and computational efficiency of PINNs to traditional numerical

methods, particularly FDM. This objective involves applying PINNs to both 1D and 2D Poisson equations to determine the capability of PINNs in accurately approximating solutions to these problems. A thorough analysis is conducted to assess the relative performance of PINNs in terms of their solution accuracy, error metrics, and computational resources required, with the aim of identifying both the strengths and limitations of PINNs as compared to classical numerical approaches. Furthermore, this objective seeks to examine the robustness of PINNs when solving problems with varying boundary conditions and domain sizes, to better understand their practical utility in real-world scenarios.

Investigating the use of multi-task learning (MTL) in the PINN framework to solve PDEs is the second goal of this study, with the goal of enhancing the model's ability to simultaneously solve for the primary solution while also predicting associated physical quantities, such as source terms and varying coefficients. By incorporating MTL, this objective aims to leverage the shared information between related tasks. In particular, the research seeks to develop a unified PINN model capable of addressing complex multi-dimensional problems by jointly solving the PDE, reconstructing unknown coefficients, and estimating the source term within a single optimization process. This methodology is anticipated to yield profound insights into the capabilities of PINNs for addressing higher-order and more intricate PDEs, as well as for managing scenarios where numerous interrelated values must be deduced concurrently.

1.3 Research questions

This work is to investigate the capabilities, constraints, and prospective upgrades of PINNs in solving PDEs in order to fulfill the above-mentioned research goals. As deep learning-based numerical solvers gain increasing attention, it is crucial to compare their effectiveness with traditional methods, assess their adaptability to more complex PDE tasks, and identify areas for enhancement. The following significant research questions are especially addressed by this study.

1. How do the accuracy and computational efficiency of PINNs compare to the FDM when solving 1D and 2D Poisson equations?
2. Can PINNs effectively solve multiple interrelated PDE tasks within a multi-task learning framework, and how does this approach impact solution accuracy?

This research will advance PINN development through a detailed comparison with FDM. Furthermore, it will examine the application of PINNs in multi-task learning frameworks for addressing coupled PDEs, emphasizing their potential to enhance the accuracy and generalizability of outcomes. The research will also determine the key challenges of PINNs and suggest ways of enhancing their

performance, including architectural modifications and hybrid models. The findings of this research will advance the use of PINNs in the solution of complex PDEs, offering new insights for theoretical and practical application.

2. Literature review

This section presents a thorough survey of the literature on PINNs and their applications. First, we introduce traditional numerical methods, discussing their advantages and limitations. We then transition to modern machine learning-based approaches, highlighting the pioneering work in PINNs and their subsequent advancements. A particular focus is given to their application in solving various types of PDEs, including the Poisson equation. Additionally, we compare PINNs with conventional methods, examining their respective strengths and weaknesses. Finally, we review emerging research on multi-task learning-based approaches for solving coupled PDEs, a rapidly evolving area in machine learning-driven numerical solutions.

2.1 Traditional methods

Ordinary and partial differential equations are now frequently solved numerically. These techniques approximate the answer in cases where the PDEs lack an analytical solution. For instance, using the Caputo fractional derivative, Abdulazeez and Modanli [20] investigate the use of the explicit FDM to solve PDEs. They offer a stability estimate for the suggested approach and create a finite difference technique specifically designed for these PDEs. The entropy-based thermal transport properties of tangent hyperbolic hybrid nanofluid flow over a slippy surface under the influence of an inclined magnetic field, and thermal radiation are examined by Hussain and Jamshed in a different study [21]. The FDM is used to numerically solve the governing equations.

Using three equations and four parameters, Li et al. create a model for rocks in the study [22]. This model effectively captures stress–strain behavior. In order to study engineering problems, such as the stress and displacement calculations for a rock slope, and a pressurized hollow cylinder with thick walls, the model is incorporated into finite element codes. The results demonstrate that at low stress levels, the basic 2D model almost matches the elastic solution, but at higher stress levels. Notably, the basic 2D model is the only one that can accurately depict sidewall convergence, and slope failures in rock formations. Tannhäuser et al.[23] present a three-dimensional second-order collocation FEM to model higher-

order electromechanical coupling with computational efficiency. Flexoelectricity, a characteristic of nano/micro-scale dielectric materials that permits interactions between electric fields and nonuniform mechanical strains, is captured by the technique. The collocation principle is used to formulate a 27-noded mixed finite element, which is then implemented in Fortran. Additionally, a mesh generation script for the 27-noded elements is developed. Validation through boundary value problem simulations demonstrates strong agreement with existing 2D and 3D results.

Berta et al. [24] offer a FVM for multidimensional classical magnetohydrodynamics. Significant improvements are provided by the approach, such as complex limiting strategies with an order reduction algorithm and a discontinuity detector, upwind limited transport averaging, and pointwise-to-pointwise reconstructions rather than conventional one-dimensional finite volume approaches. A novel spectral approach based on semi-discrete Fourier transformations is proposed by Zhou and Zhang [25] to discretize the fractional Laplacian $(-\Delta)^{\frac{\alpha}{2}}$. This distinctive feature sets it apart from existing methods, particularly Fourier pseudospectral approaches, which are typically restricted to periodic boundary conditions. The numerical analysis confirms that the method achieves spectral accuracy. In the context of stochastic processes, Lyons et al. [26] examine the function of the path signature, providing an alternative to the anticipated signature for figuring out the law of random signatures without regularity requirements. The authors use a PDE-based method to determine the characteristic function of the signature at a specified time horizon, with a focus on time-homogeneous Itô diffusion processes. The development of the generalized-signature process, which makes it easier to apply a Feynman-Kac-type theorem using a martingale technique, is a key component of their methodology.

Traditional numerical methods offer well-established, mathematically rigorous approaches for solving PDEs with high accuracy in structured domains [27] [28]. They provide deterministic solutions with error estimates and convergence guarantees. However, these methods often struggle with complex geometries, high-dimensional problems, and adaptive meshing requirements, leading to high computational costs.

2.2 Deep learning-based methods

As neural networks become more popular, scientists are using several deep learning methods to solve equations. For example, the study [29] introduces the Gabor-Filtered Fourier Neural Operator, an enhancement of the Fourier Neural Operator (FNO) for solving PDEs. It integrates a learnable Gabor-Filtered module that improves the frequency transformation matrix, emphasizing key frequencies and reducing redundancies. The results show that GFNO outperforms the original

FNO, achieving up to 28.7% error reduction on various PDEs. Additionally, a smaller version, GFNO-small, achieves similar improvements with significantly fewer parameters, demonstrating the efficiency and effectiveness of this approach. The Factorized F-FNO is presented in the study [30] as an effective surrogate model for modeling the propagation of 3D waves. F-FNO uses neural operators to overcome the difficulties of memory constraints and processing expense in 3D simulations. The publicly available HEMEW-3D collection of 30,000 wavefield simulations in heterogeneous settings is used to train the F-FNO, which predicts 3D surface wavefields that vary in space and time. Four FNO variations are compared in the study, and the effect of hyperparameters such the number of layers and channels on accuracy is examined. The findings indicate that while adding more Fourier modes had no impact on accuracy, richer models did. Strong generalization and transfer learning skills for out-of-distribution data are demonstrated by the F-FNO, which is also sensitive to input heterogeneities and robust to noise.

Enriched-DeepONet (En-DeepONet) is a unique architecture that was introduced in the study [31]. Its purpose is to improve the expressivity of neural operators in order to solve difficult seismological issues, specifically earthquake hypocenter localization. By effectively managing moving-solution operators using a combination of operations and a special "root" network, En-DeepONet overcomes the drawbacks of current models. This method improves accuracy by up to four orders of magnitude without raising training expenses. It is demonstrated that the model greatly improves earthquake localization in situations with varying arrival times and velocities, opening the door for real-time hypocenter localization in applications. The research [32] introduces RiemannONets, a neural operator approach designed to solve Riemann problems. The authors enhance the DeepONet architecture using a two-stage training process, where a basis is first extracted from the network, and then used to train the branch network. This modification significantly improves the accuracy, efficiency, and robustness of the model, yielding accurate solutions to problems compared to the standard DeepONet. Furthermore, this approach allows for physical interpretability, as the learned basis reflects key flow features. The research also compares RiemannONets with a U-Net-based neural operator, which is more accurate for large pressure ratios but computationally expensive. The paper [33] presents a numerical analysis of the deep Ritz method. While deep neural networks have gained popularity in solving PDEs, the theoretical understanding of their effectiveness has lagged behind. The authors use smooth activation functions to construct the first non-asymptotic convergence rate in the H^1 norm for the method. A unique iterative deep Ritz method (IDRM) is presented by Hu, Jin, and Wang [34] to solve a general class of elliptic problems. In order to enhance algorithm performance, the approach uses a convex penalty and the underlying function space geometry, drawing inspiration from the iterative process employed in neural network training. In contrast to conventional techniques, IDRM does not require strict regularity assumptions on

the solution and can be used for elliptic problems involving monotone operators. The method performs better than other neural PDE solvers in terms of accuracy for this class of elliptic problems. The authors evaluate the learning error and determine a convergence rate utilizing methods from monotone operator theory and Banach space geometry.

One of the popular method is Physics-Informed Neural Networks (PINNs) [16] by Raissi. The authors introduce PINNs, a deep learning framework for solving forward and inverse problems involving nonlinear PDEs. PINNs are data-efficient and appropriate for complicated spatiotemporal problems because they use neural networks to approximate solutions while naturally adhering to the governing physical principles. The study explores two main approaches: continuous-time models, which act as efficient function approximators, and discrete-time models, which integrate implicit Runge–Kutta schemes for accurate time stepping. The proposed framework is demonstrated across various classical PDE problems. In the another work [35] by De Ryck and Mishra, the authors conduct a analysis of PINNs and related models within the broader framework of PINN. They present a unified approach to analyzing the different components of error in PINNs with a focus on how these errors vary depending on the PDE type and domain dimension. The study highlights the influence of solution regularity and stability on the overall accuracy of PINNs. Notably, the authors identify training errors as a critical challenge, potentially limiting the effectiveness of PINNs in solving complex PDE problems. Cuomo et al. [18] provide a comprehensive review of PINNs and their advancements in solving PDEs. PINNs operate as a multi-task learning framework, where neural networks approximate solutions while minimizing PDE residuals. The review not only characterizes PINNs and their strengths and limitations but also explores various collocation-based extensions, such as Physics-Constrained Neural Networks (PCNNs). A key focus is on customization strategies, including activation functions, optimization techniques, network architectures, and loss function designs. While PINNs have demonstrated feasibility in some contexts compared to classical methods like FEM, the study highlights ongoing theoretical challenges that must be addressed to enhance their robustness and applicability.

The last but not least, Grossmann et al. [36] systematically compare PINNs with the FEM for solving PDEs, such as Poisson, Allen-Cahn, and semilinear Schrödinger equations. While PINNs have shown promising results in approximating PDEs, the study reveals that, in terms of solution time and accuracy, PINNs have not outperformed FEM. Although in certain experiments, PINNs were faster at evaluating the solved PDEs, FEM remained superior in terms of both computational cost and approximation accuracy. The study provides valuable insights into the strengths and limitations of PINNs when compared to traditional numerical methods like FEM.

In light of the findings from previous works, we conduct a comprehensive

comparison between the well-established FDM and the state-of-the-art PINNs in the context of solving non-complex PDEs. The FDM, known for its robustness and reliability, has long been a standard technique for numerically solving PDEs with relatively simple geometries and boundary conditions. On the other hand, PINNs are a new and interesting way to use deep learning. In our research, we look at how PINNs have grown and changed. We study them by thinking of them as a kind of multi-task learning, or MTL. This means that the PINN can work on more than one task at the same time.

When PINNs learn from many tasks at once, they can share what they learn from one problem to help with the others. This makes learning faster and better. It also helps the PINN solve different kinds of problems. So, using MTL makes PINNs more powerful and useful in many areas.

This study shows how FDM and PINNs work in solving math problems. We test both and see which one gives better answers. We also use PINNs for more than one problem at the same time. This helps them learn faster and work better [37].

3. Mathematical Background

This chapter provides the theoretical foundations and essential mathematical tools utilized throughout the thesis. We begin by introducing the fundamental concepts of PDEs, with particular focus on the Poisson equation, which serves as the central problem studied in this work. This is followed by an overview of neural networks, establishing the groundwork necessary for understanding their integration into scientific computing.

Subsequently, we delve into PINN, a modern approach that combines deep learning with the governing laws of physics to solve PDEs. This section outlines the formulation, architecture, training procedure, and performance metrics relevant to PINNs. Finally, we describe the FDM, a classical numerical technique for solving PDEs, which serves as a benchmark to assess the accuracy and efficiency of the PINN-based solutions. Collectively, these sections offer a comprehensive background for the computational experiments and methodologies discussed in the later chapters.

3.1 Partial differential equation

Definition 3.1.1. *A PDE is an equation that involves an unknown function of multiple variables along with its partial derivatives. Specifically, let $u : U \to \mathbb{R}$ be an unknown function defined over an open subset $U \subset \mathbb{R}^n$, where*

$$x = (x_1, x_2, \ldots, x_n) \in U. \tag{3.1.1}$$

A PDE of order k takes the general form:

$$F(D^k u, D^{k-1} u, \ldots, Du, u, x) = 0, \tag{3.1.2}$$

where

$$F : \mathbb{R}^{n^k} \times \mathbb{R}^{n^{k-1}} \times \cdots \times \mathbb{R}^n \times \mathbb{R} \times U \to \mathbb{R}, \tag{3.1.3}$$

and D represents the partial derivative operator.

Initial conditions specify the state of the function at an initial time t_0, given as:

$$u(x, t_0) = g(x), \quad x \in \Omega, \tag{3.1.4}$$

where $\Omega \subseteq \mathbb{R}^n$ is the spatial domain. Boundary conditions impose constraints on the function at the boundary $\partial\Omega$ of the domain, commonly expressed as:

$$B(u, x) = h(x), \quad x \in \partial\Omega, \tag{3.1.5}$$

where B represents a boundary operator that could enforce dirichlet condition.

Example 3.1.1. *(Simplest First-Order PDE)*

$$u_x = 0. \tag{3.1.6}$$

This equation implies that u is independent of x, meaning u is constant along the x-direction.

Example 3.1.2. *(Laplace's Equation)*

$$\Delta u = u_{xx} + u_{yy} = 0. \tag{3.1.7}$$

Laplace's equation arises in electrostatics, fluid dynamics, and potential theory.

Example 3.1.3. *(Heat Equation)*

$$u_t = k\Delta u. \tag{3.1.8}$$

This equation describes heat conduction and diffusion processes where k is the thermal diffusivity.

Example 3.1.4. *(Schrödinger Equation)*

$$i\hbar\frac{\partial\psi}{\partial t} = \left(-\frac{\hbar^2}{2m}\Delta + V\right)\psi. \tag{3.1.9}$$

This fundamental equation in quantum mechanics describes the evolution of a quantum state.

Example 3.1.5. *(Wave Equation)*

$$u_{tt} - c^2\Delta u = 0. \tag{3.1.10}$$

This equation models wave propagation in acoustics, electromagnetism, and elasticity.

Example 3.1.6. *(Elasticity Equation)*

$$u_{tt} + K\Delta^2 u = 0. \tag{3.1.11}$$

This equation is used in elasticity and describes oscillations in plates and membranes.

Example 3.1.7. *(Maxwell's Equations)*

$$\begin{cases} E_t - c\nabla \times H = 0, \\ H_t + c\nabla \times E = 0, \\ \nabla \cdot E = 0, \\ \nabla \cdot H = 0. \end{cases} \tag{3.1.12}$$

These equations describe the behavior of electric (E) and magnetic (H) fields in vacuum.

Example 3.1.8. *(Navier-Stokes Equations)*

$$\begin{cases} \rho v_t + (v \cdot \nabla)v - \nu \Delta v = -\nabla p, \\ \nabla \cdot v = 0. \end{cases} \tag{3.1.13}$$

These equations model fluid motion and are widely used in aerodynamics and hydrodynamics.

3.1.1 Poisson equation

Definition 3.1.2. *The Poisson equation is a elliptic PDE of the form:*

$$\Delta u(x) = f(x), \quad x \in \Omega, \tag{3.1.14}$$

where:

- $\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$ *is the Laplacian operator,*
- $u : \Omega \to \mathbb{R}$ *is the unknown function,*
- $f : \Omega \to \mathbb{R}$ *is a given source function,*
- $\Omega \subseteq \mathbb{R}^n$ *is an open domain.*

If $f(x) \equiv 0$, the equation reduces to the Laplace equation:

$$\Delta u(x) = 0. \tag{3.1.15}$$

Example 3.1.9. *Consider the 1D Poisson equation on the interval $\Omega = (0, 1)$:*

$$\frac{d^2u}{dx^2} = -\sin(\pi x), \quad x \in (0, 1). \tag{3.1.16}$$

This equation describes a system where the source function is $f(x) = -\sin(\pi x)$.

Example 3.1.10. *In two dimensions, the Poisson equation can be written as:*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = e^x \sin(y), \quad (x, y) \in \Omega. \tag{3.1.17}$$

Here, the source function is $f(x, y) = e^x \sin(y)$*, and* Ω *represents a region in* $\mathbb{R}^2$*.*

Example 3.1.11. *For a three-dimensional case, we take:*

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -\cos(x)y^2 + z, \quad (x, y, z) \in \Omega. \tag{3.1.18}$$

This represents a Poisson equation with a source function $f(x, y, z) = -\cos(x)y^2 + z$ *in a three-dimensional domain.*

3.2 Neural networks

A fundamental model of a node in an artificial neural network (ANN) is illustrated in Figure 3.1. Similar to the strength of synaptic connections in biological neurons, each node gets several inputs from other nodes, each of which is assigned a specific weight. The node activates and sends the signal to other nodes via a transfer function when the weighted total of inputs exceeds a predetermined threshold. Mathematically, this can be represented as:

$$y_{pred} = f\left(\sum_{i=0}^{n} w_i x_i - T\right) \tag{3.2.1}$$

where y represents the node's output, f is the transfer function, w_i is the weight associated with input x_i, and T is the activation threshold. The transfer function can take different forms, with non-linear transfer functions generally being more effective than linear ones, as linear functions can only classify a limited set of problems. A fundamental example of a non-linear transfer function is the step function, shown in Figure 3.2:

$$y_{pred} = \begin{cases} 0, & \text{if } \sum_{i=0}^{n} w_i x_i > T \\ 1, & \text{if } \sum_{i=0}^{n} w_i x_i < T \end{cases} \tag{3.2.2}$$

Additionally, the sigmoid function is commonly used as an activation function due to its smooth, continuous nature, which ensures the continuity of both the function and its derivative (see 3.2.3). The parameter β controls the steepness of the curve, with higher values resulting in a sharper transition between 0 and 1.

$$y = \frac{1}{1 + \exp(-\beta x)} \tag{3.2.3}$$

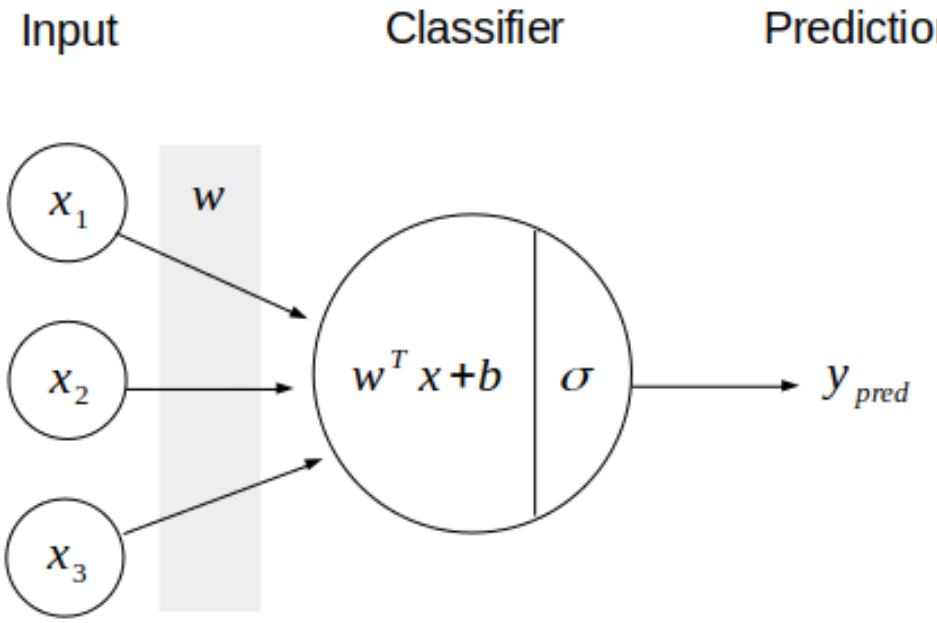

Figure 3.1: A basic model of a single node: x_i = input, w = weight, b = bias, σ = activation function, y_{pred} = output

In scientific computing, Neural networks have emerged as a tool for solving PDEs. Unlike traditional numerical methods, deep learning approaches, particularly PINNs, integrate physical laws into training, enabling efficient solutions in high-dimensional spaces.

3.2.1 Deep Neural Networks

A *deep neural network* (DNN) is defined as a parameterized function $\mathcal{F}_{\boldsymbol{\theta}} : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$, where $\boldsymbol{\theta} \in \mathbb{R}^P$ denotes the collection of all trainable parameters (weights and biases), and the function $\mathcal{F}_{\boldsymbol{\theta}}$ is composed of a finite sequence of affine transformations and nonlinear activation functions.

Formally, let $L \in \mathbb{N}$ be the number of layers. For each $\ell = 1, 2, \ldots, L$, define:

$$\mathbf{z}^{(\ell)} = \mathbf{W}^{(\ell)}\mathbf{a}^{(\ell-1)} + \mathbf{b}^{(\ell)}, \quad \mathbf{a}^{(\ell)} = \sigma\left(\mathbf{z}^{(\ell)}\right)$$

where:

$$\begin{aligned}
\mathbf{a}^{(0)} &= \mathbf{x} \in \mathbb{R}^{d_{\text{in}}} \quad \text{(input vector)} \\
\mathbf{W}^{(\ell)} &\in \mathbb{R}^{d_\ell \times d_{\ell-1}} \quad \text{(weight matrix)} \\
\mathbf{b}^{(\ell)} &\in \mathbb{R}^{d_\ell} \quad \text{(bias vector)} \\
\sigma(\cdot) &: \mathbb{R} \to \mathbb{R} \quad \text{(activation function, e.g., ReLU or sigmoid)}
\end{aligned}$$

The output of the network is then expressed as:

$$\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{a}^{(L)}$$

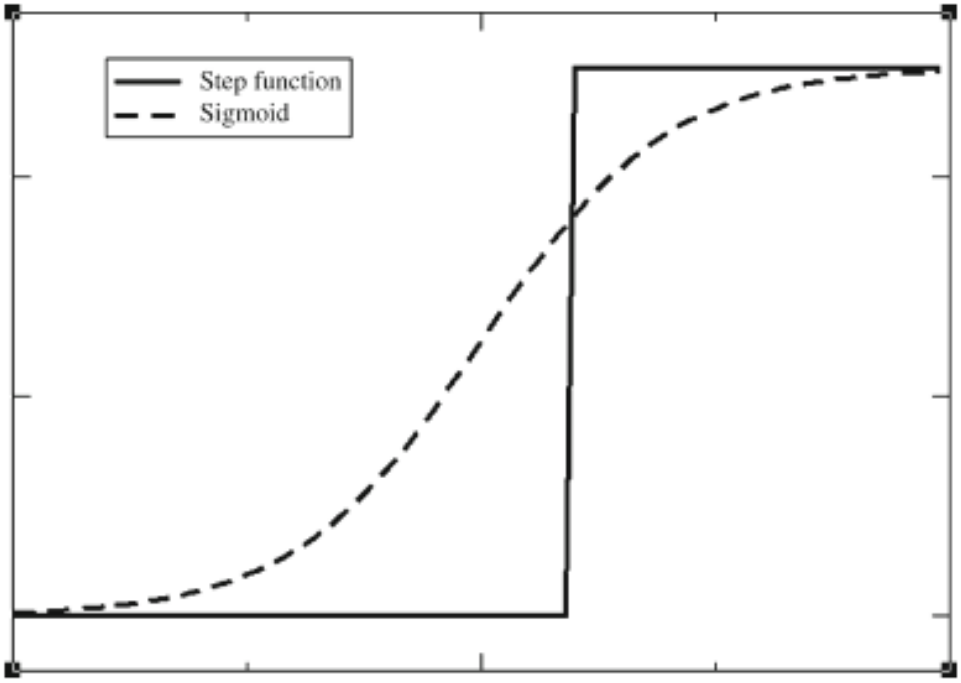

Figure 3.2: Activation function

The trainable parameter set is defined by:

$$\boldsymbol{\theta} = \left\{ \left(\mathbf{W}^{(\ell)}, \mathbf{b}^{(\ell)} \right) \right\}_{\ell=1}^{L}$$

The goal of training is to find $\boldsymbol{\theta}^* \in \mathbb{R}^P$ that minimizes a loss functional $\mathcal{L}$, typically given by:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} \ell \left(\mathcal{F}_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), \mathbf{y}^{(i)} \right)$$

where $\ell : \mathbb{R}^{d_{\text{out}}} \times \mathbb{R}^{d_{\text{out}}} \to \mathbb{R}_+$ is a pointwise loss function such as mean squared error:

$$\ell(\hat{\mathbf{y}}, \mathbf{y}) = \|\hat{\mathbf{y}} - \mathbf{y}\|_2^2$$

Optimization is conducted via gradient-based methods, often stochastic gradient descent (SGD), such that:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(k)}), \quad \eta \in \mathbb{R}_+ \text{ (learning rate)}$$

Under sufficient capacity and regularization, the DNN can approximate any measurable function $f \in L^p(\mathbb{R}^d)$, as per the universal approximation theorem.

In the context of solving PDEs, one defines a neural surrogate $u_{\boldsymbol{\theta}}(\mathbf{x}) \approx u(\mathbf{x})$ and formulates a physics-informed loss that includes both empirical data and differential residuals, enabling mesh-free, high-dimensional approximations of the solution space. The Figure 3.3 shown here illustrates the structure of DNNs, highlighting the layers of interconnected neurons.

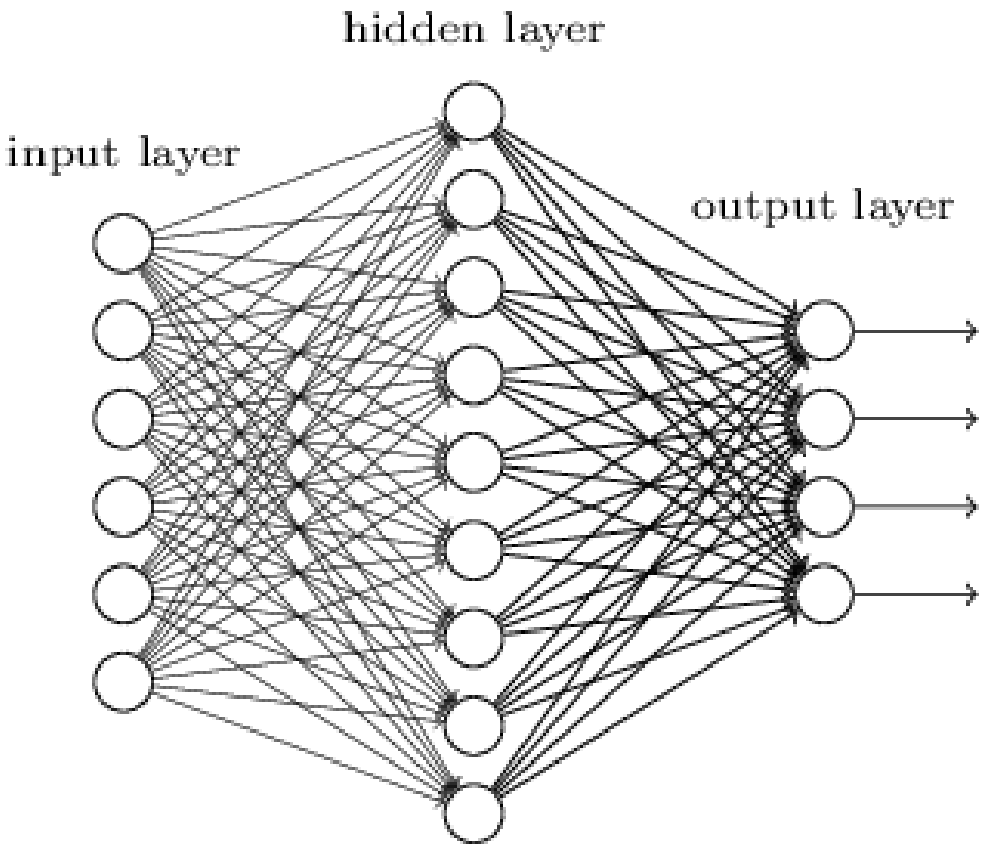

Figure 3.3: Deep Neural Networks

3.3 Physics-Informed Neural Networks

PINN is a class of DL models, typically represented by differential equations. Unlike conventional neural networks that rely solely on large datasets, PINNs incorporate prior knowledge of the underlying governing equations, such as PDEs and ODEs, to enforce physical consistency. Traditional data-driven models require extensive labeled datasets for training, which may be impractical or infeasible in many real-world applications, particularly in scientific and engineering problems where data acquisition is costly or sparse. In contrast, physics-informed learning leverages the fundamental laws governing a system, reducing the dependency on large datasets while ensuring physically consistent solutions. By embedding domain knowledge into neural networks, PINNs enhance generalization, improve interpretability, and mitigate issues such as overfitting and data inefficiency.

3.3.1 Problem statement

Consider a general PDE defined in a domain Ω with appropriate boundary and initial conditions:

$$\mathcal{F}\left(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x}), \nabla^2 u(\mathbf{x}), \dots\right) = 0, \quad \mathbf{x} \in \Omega, \tag{3.3.1}$$

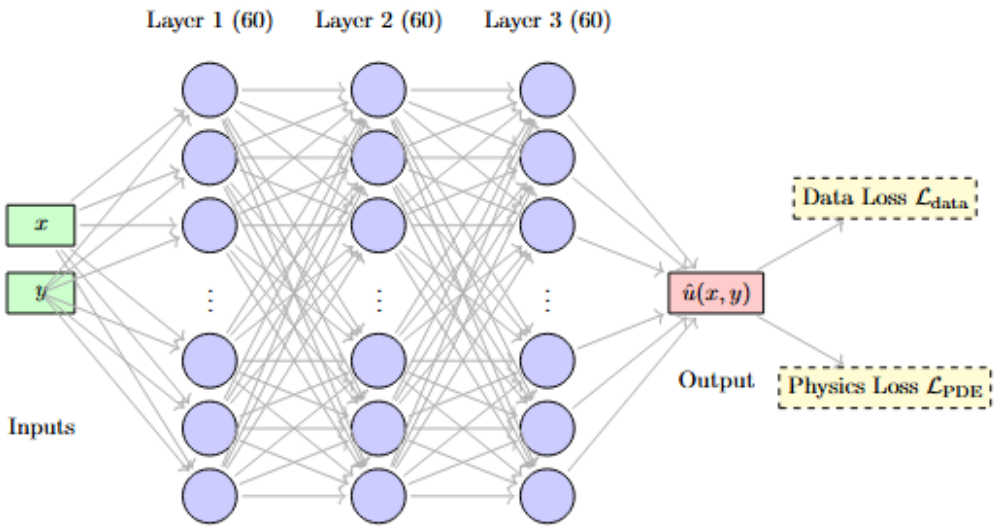

Figure 3.4: Physics-Informed Neural Networks

where $u(\mathbf{x})$ is the solution function, and $\mathcal{F}$ is a differential operator that describes the physical system.

The PDE is subject to boundary conditions:

$$\mathcal{B}\left(u(\mathbf{x})\right) = g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \tag{3.3.2}$$

where $g(\mathbf{x})$ represents a boundary condition.

If applicable, initial conditions are given by:

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{3.3.3}$$

3.3.2 Neural network approximation

In PINNs, the answer to the PDE is guessed using a neural network $u_\theta(\mathbf{x})$ with settings called θ. The network learns to follow the PDE and its rules by making a special number, called a loss, as small as possible. This loss includes mistakes from the main equations and from the starting or edge conditions [38][39].

The neural network $u_\theta(\mathbf{x})$ is parameterized as:

$$u_\theta(\mathbf{x}) \approx u(\mathbf{x}). \tag{3.3.4}$$

The derivatives required in $\mathcal{F}$ are computed using automatic differentiation.

3.3.3 Loss function

The total loss function $\mathcal{L}$ in PINNs consists of multiple components:

$$\mathcal{L} = \mathcal{L}_{\text{PDE}} + \mathcal{L}_{\text{BC}} + \mathcal{L}_{\text{IC}}. \tag{3.3.5}$$

The PDE residual loss enforces the governing equation:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left| \mathcal{F}\left(\mathbf{x}_i, u_\theta, \nabla u_\theta, \nabla^2 u_\theta, \dots\right) \right|^2. \tag{3.3.6}$$

The boundary loss ensures boundary conditions are satisfied:

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_b} \sum_{i=1}^{N_b} \left| \mathcal{B}\left(u_\theta(\mathbf{x}_i)\right) - g(\mathbf{x}_i) \right|^2. \tag{3.3.7}$$

The initial loss (for time-dependent problems) ensures initial conditions are met:

$$\mathcal{L}_{\text{IC}} = \frac{1}{N_i} \sum_{i=1}^{N_i} \left| u_\theta(\mathbf{x}_i, 0) - u_0(\mathbf{x}_i) \right|^2. \tag{3.3.8}$$

The data-based loss (if available) is given by:

$$\mathcal{L}_{\text{Data}} = \frac{1}{N_d} \sum_{i=1}^{N_d} \left| u_\theta(\mathbf{x}_i) - u_{\text{data}}(\mathbf{x}_i) \right|^2. \tag{3.3.9}$$

3.3.4 Training procedure

The training of PINNs involves solving the optimization problem:

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta). \tag{3.3.10}$$

To approximate the solution, we sample points:

- $\{\mathbf{x}_i^r\}_{i=1}^{N_r}$ inside the domain Ω (residual points),
- $\{\mathbf{x}_i^b\}_{i=1}^{N_b}$ on the boundary $\partial\Omega$ (boundary condition points),
- $\{\mathbf{x}_i^i\}_{i=1}^{N_i}$ for the initial condition (if applicable).

The PINN is trained using a Adam optimizer. The parameters θ are updated iteratively using:

$$\theta^{(k+1)} = \theta^{(k)} - \eta \nabla_\theta \mathcal{L}(\theta^{(k)}), \tag{3.3.11}$$

where η is the learning rate.

3.3.5 Adam optimization algorithm

The Adam optimizer is an adaptive gradient-based optimization method used for training PINNs. It updates the parameters θ using first and second moment estimates of the gradients.

Given the loss function $\mathcal{L}(\theta)$, the change in the loss with respect to the settings θ at step k is:

$$g_k = \nabla_\theta \mathcal{L}(\theta_k). \tag{3.3.12}$$

Adam maintains an exponentially decaying average of past gradients:

$$m_k = \beta_1 m_{k-1} + (1 - \beta_1) g_k, \tag{3.3.13}$$

and an exponentially decaying average of past squared gradients:

$$v_k = \beta_2 v_{k-1} + (1 - \beta_2) g_k^2. \tag{3.3.14}$$

To correct the bias in the moment estimates, the bias-corrected values are computed as:

$$\hat{m}_k = \frac{m_k}{1 - \beta_1^k}, \quad \hat{v}_k = \frac{v_k}{1 - \beta_2^k}. \tag{3.3.15}$$

The parameter update step is then given by:

$$\theta_{k+1} = \theta_k - \eta \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}, \tag{3.3.16}$$

where η is the learning rate, $\beta_1, \beta_2 \in [0, 1)$ are decay rates, and ϵ is a small constant for numerical stability.

3.3.6 Evaluate metric

To rigorously assess the accuracy of the solutions obtained via PINNs and the FDM, the L_2 relative error is employed as a standard quantitative metric. This metric provides a normalized measure of the discrepancy between the predicted solution u_θ — typically produced by a neural network or numerical scheme — and the exact or reference solution u^{true}. Mathematically, the L_2 relative error is defined as

$$\mathcal{L}_2 = \frac{\|u_\theta - u^{\text{true}}\|_2}{\|u^{\text{true}}\|_2}, \tag{3.3.17}$$

where $\|\cdot\|_2$ denotes the discrete L_2-norm, typically computed over a finite number of points in the spatial domain. This formulation ensures that the error is not

only expressed in absolute terms but also scaled relative to the magnitude of the exact solution, thereby enabling a fair comparison across different problems and solution scales.

The L_2 relative error serves as a critical benchmark for evaluating the fidelity of the learned solution in replicating the true physical behavior governed by the differential equation. A smaller value of $\mathcal{L}_2$ signifies a closer match between the predicted and true solutions, thereby indicating that the model-whether data-driven or numerical-effectively captures the underlying mathematical and physical structure of the problem. This error metric is especially valuable in contexts where precision is paramount, such as inverse problems, high-resolution simulations, and physics-based modeling in engineering and applied sciences.

3.4 Finite Difference Method

The FDM is a classical numerical approach for approximating the solutions of PDEs by discretizing the continuous computational domain into a finite set of uniformly or non-uniformly spaced grid points. In this framework, the continuous spatial derivatives present in the governing equations are systematically replaced by finite difference approximations, such as forward, backward, or central differences, depending on the desired order of accuracy and stability considerations.

This transformation converts the original differential equation into a corresponding system of algebraic equations that can be solved using conventional linear or nonlinear solvers. The discretization resolution and the particular finite difference techniques used have a significant impact on the FDM's accuracy. As the mesh is refined, the numerical solution is expected to converge toward the exact analytical solution, assuming sufficient smoothness of the underlying function.

Consider a function $u(x)$ defined on a one-dimensional domain $x \in [a, b]$. A uniform discretization of this domain consists of grid points x_i such that

$$x_i = a + ih, \quad i = 0, 1, \ldots, N, \tag{3.4.1}$$

where $h = \frac{b-a}{N}$ is the grid spacing. The derivative of $u(x)$ at a grid point x_i can be approximated using finite differences. The forward difference approximation of the first derivative is given by

$$\left.\frac{du}{dx}\right|_{x_i} \approx \frac{u_{i+1} - u_i}{h} + \mathcal{O}(h), \tag{3.4.2}$$

which has a first-order truncation error. The backward difference approximation is given by

$$\left.\frac{du}{dx}\right|_{x_i} \approx \frac{u_i - u_{i-1}}{h} + \mathcal{O}(h). \tag{3.4.3}$$

A more accurate approximation is the central difference formula, which has a second-order error:

$$\left.\frac{du}{dx}\right|_{x_i} \approx \frac{u_{i+1} - u_{i-1}}{2h} + \mathcal{O}(h^2). \tag{3.4.4}$$

For the second derivative, the central difference approximation is given by

$$\left.\frac{d^2u}{dx^2}\right|_{x_i} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} + \mathcal{O}(h^2). \tag{3.4.5}$$

To illustrate the use of FDM, consider the one-dimensional Poisson equation,

$$\frac{d^2u}{dx^2} = f(x), \quad x \in (a, b), \tag{3.4.6}$$

with Dirichlet boundary conditions $u(a) = u_0$ and $u(b) = u_N$. Discretizing the second derivative using the central difference formula, we obtain the finite difference equation at each interior grid point:

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f_i, \quad i = 1, \ldots, N-1. \tag{3.4.7}$$

Rearranging, this forms a system of linear equations,

$$u_{i-1} - 2u_i + u_{i+1} = h^2 f_i. \tag{3.4.8}$$

These equations, together with the boundary conditions, form a system that can be solved for u_i. This demonstrates how FDM transforms a differential equation into a discrete system, making it suitable for numerical computation.

4. Methodology

The methodology encompasses data sampling, numerical solution via FDM, PINNs implementation, and validation through error analysis. All implementations were developed using Python, with computational experiments conducted using standard machine learning frameworks. The complete code is publicly accessible on GitHub at [40], facilitating replication and further exploration.

4.1 Data generation

The dataset used to obtain the FDM solution for the 1D Poisson equation is generated by discretizing a one-dimensional domain. A uniform grid of 512 points is created in the interval $[0, 1]$, at which the function values are calculated. The dataset used to obtain the FDM solution for the 2D Poisson equation is generated by discretizing a square domain $[0, 1] \times [0, 1]$. A uniform grid of 1000×1000 points is constructed, where function values are evaluated at each grid point. The discretization follows a structured finite difference scheme, leading to a large system of linear equations that approximate the continuous problem. The dataset includes the computed numerical solution, an analytical reference solution, and the corresponding source term values.

The training sets employed by the PINNs to solve the 1D and 2D Poisson equations are acquired through Latin Hypercube Sampling (LHS) [41] to achieve the full coverage of the computational domain. In the 1D case, the domain is approximated by 256 training points sampled from the domain $[0, 1]$ through LHS and the boundary conditions are imposed by introducing more sampled points onto the domain boundaries. To test the prediction accuracy, a 512-point uniform discretization is employed. For two-dimensional Poisson equation, the dataset construction follows a similar course, extending LHS-based sampling to a two-dimensional domain $[0, 1] \times [0, 1]$. The training set is comprised of 2000 randomly distributed collocation points across the domain so that the PINN is exposed to a sufficiently large input range during training. Boundary conditions are imposed by 250 strategically chosen points on the domain's boundary. To test performance, a high-resolution 1000×1000 mesh is generated, allowing fine-resolution comparison between the target solution and the network prediction.

4.2 FDM implementation

In this section, we explain the numerical approach used to solve one and two dimensional Poisson equations using the FDM. The objective is to approximate the exact solutions of the numerical method through error analysis.

For the 1D case, the domain is discretized using 512 points. The central difference scheme is applied to approximate the second-order derivative. The resulting tri-diagonal system is solved to obtain the numerical solution, which is then compared to the exact solution. Error analysis is performed using the relative $L2$-norm error.

For the two-dimensional case, the domain is discretized into a 1000×1000 grid. A five-point stencil is used to construct a sparse system matrix, incorporating boundary conditions. The system is solved using sparse linear solvers, and the numerical solution is compared against an analytical solution. The accuracy is assessed through relative $L2$-norm error computation.

The implementation is carried out in python, utilizing numerical libraries for matrix operations and sparse solvers. Results, including numerical and exact solutions and error metrics, are stored in a structured format for further analysis. The computed relative $L2$ errors provide insight into the performance of the FDM in solving Poisson equations in both one and two dimensions.

4.3 PINN implementation

In this section, we present the methodology employed to approximate the solution of a Poisson equation using PINNs. The framework integrates deep learning with physical constraints to ensure adherence to the governing PDE and boundary conditions. The neural network follows a fully connected architecture implemented using the Flax framework, with multiple hidden layers and hyperbolic tangent activation functions. The output layer produces a scalar-valued approximation of the solution at a given input location.

The training process involves minimizing a loss function that consists of three components: the PDE residual, the Dirichlet boundary condition residual. The residuals are computed using automatic differentiation techniques such as Hessian-vector products to enforce PDE constraints. To ensure an even distribution of training points, LHS is employed for both the domain and boundary conditions.

Optimization is performed in two stages: first, the model is trained using the Adam optimizer for a predefined number of epochs with a fixed learning rate. Subsequently, a fine-tuning step is conducted using the L-BFGS optimizer, which employs a quasi-Newton approach to refine the solution further. The PINN model

is implemented in JAX, allowing for just-in-time compilation and efficient parallel execution on GPU hardware.

After training, the model's performance is assessed by comparing its predictions against an analytical solution. The accuracy is quantified using the relative L_2 error norm, measuring the deviation between the predicted and true solutions. Finally, the computed solutions, evaluation points, and error metrics are stored in a structured JSON format to facilitate further analysis and reproducibility.

4.4 Multi-task learning

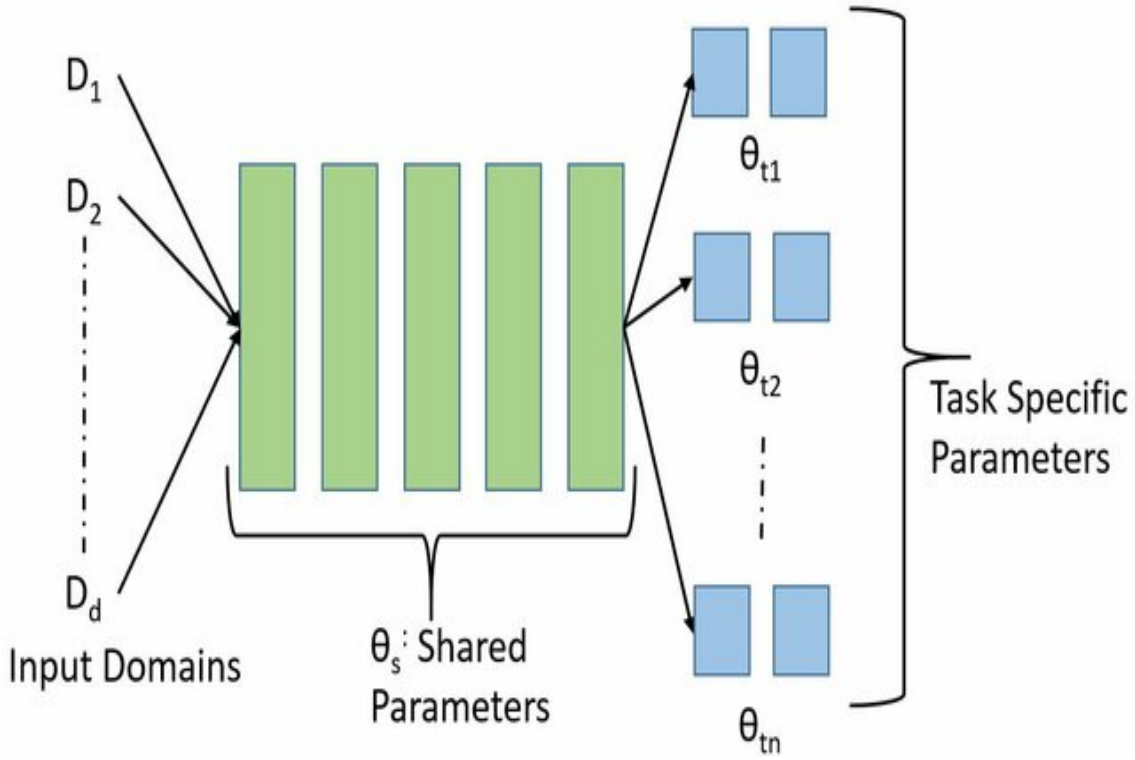

Figure 4.1: A Generalized Multi-Task Learning Framework for Solving Parameterized Computational Problems

In this subsection, we introduce and analyze a hybrid computational framework that integrates the FDM with PINN for solving a second-order partial differential equation characterized by a spatially varying coefficient and an unknown source term. This formulation addresses the inverse problem by treating both the governing dynamics and unknown parameters as learning targets within a unified multi-task learning paradigm. Specifically, the PINN architecture is designed to simultaneously infer the solution $u(x)$, reconstruct the latent source term $Q(x)$, and identify the spatially dependent coefficient $a(x)$.

This joint estimation is achieved by embedding the physics of the system-expressed through the differential operator-directly into the loss function. The model leverages a combination of data-driven supervision from known solution

or boundary values and physics-based constraints imposed by the differential equation. Such an approach allows for effective learning even in scenarios where observational data is sparse or partially missing. By coupling domain knowledge with neural network flexibility, this hybrid framework offers a promising strategy for solving complex inverse problems that may be ill-posed or underdetermined using traditional numerical methods alone.

5. Comparative analysis between FDM and PINN

The studies carried out to assess and contrast the effectiveness of the FDM and PINN in resolving Poisson equations are presented in this chapter. The objective is to assess the accuracy, robustness, and generalization capabilities of these two approaches in different spatial dimensions. The results are organized into two sections: the first addresses the 1D Poisson equation, while the second extends the analysis to the 2D case.

5.1 Experiment 1: 1D Poisson equation

Let us investigate 1D Poisson equation defined as:

$$\frac{d^2u(x)}{dx^2} = 16x^7e^{-x^4} - 20x^3e^{-x^4}, \quad x \in [0,1] \tag{5.1.1}$$

with Dirichlet boundary conditions:

$$u(0) = 0, \quad u(1) = e^{-1}. \tag{5.1.2}$$

The equation has an analytical solution, which can be written as:

$$u(x) = xe^{-x^4}. \tag{5.1.3}$$

As discussed in Section 3.4, the FDM begins by reformulating the partial differential equation (PDE) into its weak form. This transformation facilitates the discretization of the continuous domain. For the Poisson equation, this step has already been completed. The subsequent procedure involves constructing a computational mesh by discretizing the spatial domain $[0, 1]$ into smaller subintervals, or cells. In this experiment, the domain is divided into 512 uniformly spaced cells. Increasing the number of cells leads to a finer grid, which generally improves the accuracy of the numerical solution; however, it also imposes a greater computational cost.

In parallel, the 1D Poisson equation is also solved using PINNs. The design of the PINN model necessitates the specification of three key components prior to training. The first is the loss function, which evaluates how well the neural network solution satisfies the PDE. In this work, we adopt the conventional formulation used in vanilla PINNs, where the total loss is computed as the mean squared error across the collocation points of the PDE.

The neural network architecture, which includes the network type, activation function, and hidden layer and neuron configuration, is the second component. A fully connected neural network with the (`tanh`) activation function is used in this instance. Three hidden layers with 20 neurons each and one output node are represented by the selected architecture, which has the following structure [20, 20, 20, 1].

To assess performance, the approximated solutions obtained from both FDM and PINN are compared to the exact solution over the interval $[0, 1]$, using 512 evaluation points. Figure 5.1 illustrates the exact solution along with the numerical approximations. While most PINN configurations produce results in close agreement with the ground truth, a particular setup with only one hidden layer and a single neuron demonstrates significant deviation and fails to adequately enforce the boundary conditions.

Quantitatively, the relative error for the FDM solution is computed as 7.26×10^{-8}, significantly lower than the corresponding PINN error of 5.63×10^{-6}. These findings indicate that, in the one-dimensional case, the FDM offers a more accurate approximation of the Poisson equation compared to the PINN approach, albeit at the cost of greater dependency on domain discretization.

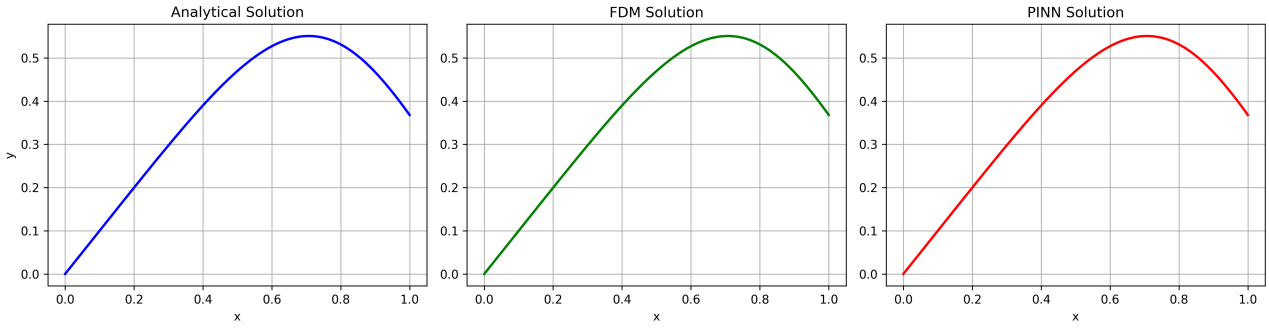

Figure 5.1: Comparison of solutions: 1-Exact, 2-FDM, and 3-PINN

5.2 Experiment 2: 2D Poisson equation

Let us now investigate 2D Poisson equation defined as:

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 2(x^4(3y-2) + x^3(4-6y) + x^2). \tag{5.2.1}$$

The boundary conditions are:

$$u(x,0) = u(x,1) = u(0,y) = u(1,y) = 0. \tag{5.2.2}$$

The analytical solution of the equation is:

$$u(x,y) = (x-1)^2 y (y-1)^2 x^2. \tag{5.2.3}$$

To comprehensively evaluate the capabilities of both numerical and machine learning-based methods in higher-dimensional contexts, we consider the solution of the two-dimensional Poisson equation over the spatial domain $[0,1] \times [0,1]$. This problem setup allows us to investigate the performance of the methods under increased computational and geometric complexity compared to the one-dimensional case. The FDM is implemented on a finely discretized uniform grid of 1000×1000 points, ensuring a high-resolution numerical approximation that serves as a reliable benchmark for comparative analysis.

In parallel, a PINN is trained to approximate the same solution across the identical domain. The neural architecture consists of a fully connected feed-forward network with three hidden layers, each comprising 60 neurons, resulting in a layer configuration of $[60, 60, 60, 1]$. The hyperbolic tangent (`tanh`) activation function is used due to its smooth, differentiable nature, which aids in learning complex spatial features while ensuring compatibility with the gradient-based optimization routines used during training.

Figure 5.2 illustrates the comparative results, showing the analytical solution alongside the approximations produced by the FDM and the PINN. A visual inspection suggests that both methods are capable of capturing the overall qualitative structure of the solution. The PINN, in particular, demonstrates a remarkable ability to replicate the broad features of the solution profile without the need for discretized grids. However, visual agreement alone does not fully capture the accuracy and fidelity of the numerical approximations; therefore, a detailed quantitative evaluation is necessary.

To achieve this, we calculate the L_2 relative errors for both methods using the analytical solution as the ground truth reference. The FDM achieves a very low relative error of 2.21×10^{-4}, highlighting its exceptional precision when used on a sufficiently fine grid. In contrast, the PINN yields a higher relative error of 6.01×10^{-3}, which, although larger, still represents a reasonable approximation given the absence of a mesh and the general-purpose nature of the learning algorithm.

These results corroborate the observations made in the 1D case: traditional numerical methods like FDM continue to offer superior accuracy in well-posed problems with known analytical solutions and structured domains. However,

the PINN remains competitive and demonstrates its potential as a flexible and generalizable method, particularly valuable in scenarios where data are sparse, the domain is irregular, or the governing equations include unknown parameters. This experiment reinforces the idea that while PINNs may not yet surpass classical solvers in raw precision, their adaptability and data-driven nature make them a compelling alternative in broader and more complex problem settings.

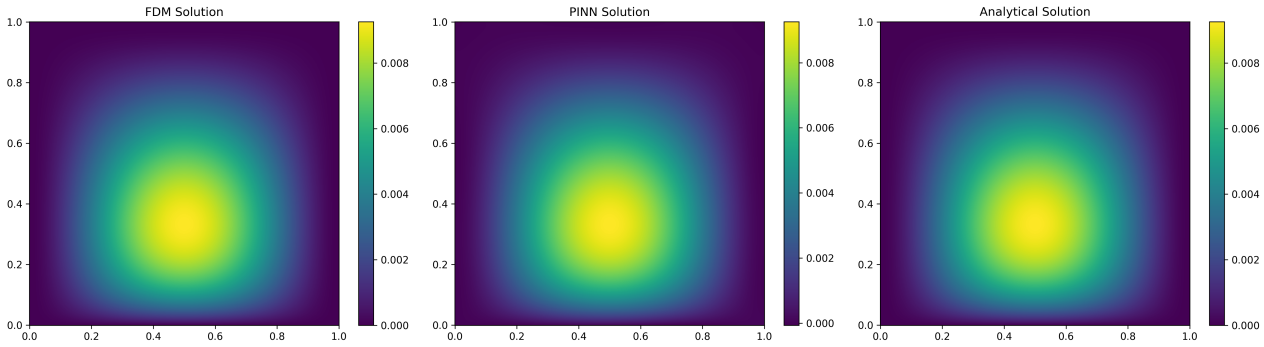

Figure 5.2: Comparison of solutions: 1-PINN, 2-Exact, and 3-FDM

6. A multi-task framework for solving PDEs with PINN

The problem at hand is governed by the following second-order differential equation:

$$\frac{d^2U(x)}{dx^2} - a(x)U(x) = Q(x), \quad 0 < x < L \tag{6.0.1}$$

where $Q(x)$ represents the source term, and $a(x)$ is the varying coefficient. Specifically, the source term is given by:

$$Q(x) = 1 + b_1 \sin(w_1 x) \tag{6.0.2}$$

and the varying coefficient $a(x)$ is defined as:

$$a(x) = b_1 + \frac{x}{1 + x^2} \tag{6.0.3}$$

with boundary conditions $U(0) = 1$ and $U(L) = 3$.

In the context of multi-task learning, we aim to solve the PDE for $U(x)$ while simultaneously predicting and reconstructing the source term $Q(x)$ and the varying coefficient $a(x)$. The model is tasked with learning the relationships between the solution $U(x)$, the source term $Q(x)$, and the varying coefficient $a(x)$, allowing for joint optimization of these interconnected tasks.

6.1 Experiment 1: Joint reconstruction of solution and source term

The first experimental study is devoted to the simultaneous reconstruction of the source term $Q(x)$ and the corresponding solution $U(x)$ in a second-order linear differential equation characterized by a spatially varying coefficient $a(x)$. This problem is emblematic of a class of inverse modeling tasks wherein key components of the system-specifically, the forcing function is not directly observable. To address this, a PINN framework is employed, in which the differential operator

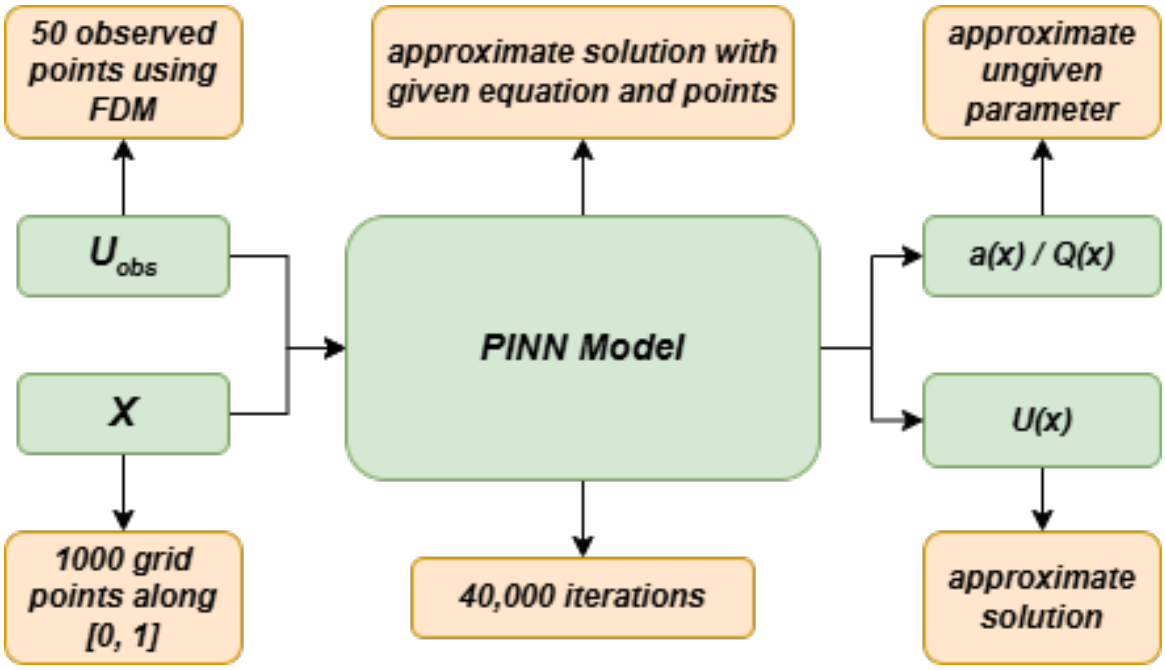

Figure 6.1: Visualization of MTL model

is embedded into the loss function. This enables the model to learn both the observable solution and the hidden source term in a unified, data-efficient manner.

The training process leverages a neural architecture where both $U(x)$ and $Q(x)$ are treated as trainable outputs, subject to the constraints imposed by the governing physical laws. The residuals of the partial differential equation are incorporated into the loss function to guide learning, ensuring that the network produces outputs consistent with the underlying dynamics of the system.

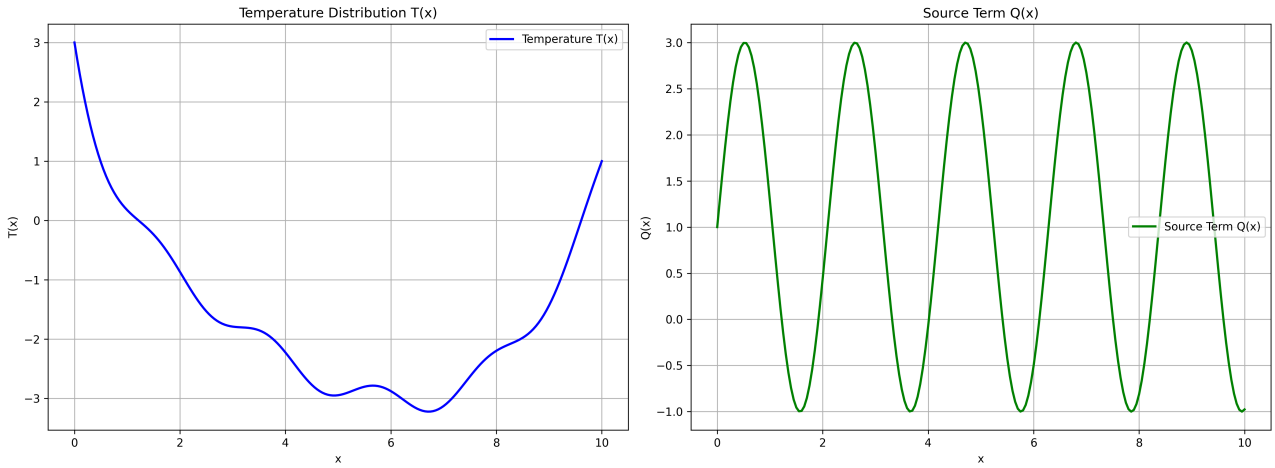

Figure 6.2: Reference solution and source term $Q(x)$ computed using FDM

The model was trained over a total of 40,000 epochs using a gradient-based optimizer. During the early stages of training, the composite loss function exhibited high values, reflecting the network's initial lack of alignment with the physics of the system. However, as training progressed, the loss function steadily decreased,

converging to a final value of 1.87×10^{-2}. This sustained decline in loss is indicative of the network's progressive acquisition of the governing patterns embedded in the differential equation, ultimately resulting in accurate recovery of both the solution and the latent source term.

To assess the performance of the trained PINN, its predicted solution $U(x)$ was compared to the numerical solution obtained using the FDM, as shown in Figure 6.2. The results revealed a high degree of concordance between the two approaches, thereby validating the fidelity of the PINN-based approximation. Notably, the PINN exhibited strong generalization performance, accurately predicting solution values even at spatial locations not explicitly included in the training set, thus demonstrating its robustness in extrapolative inference.

Figure 6.3 presents the temperature field predicted by the PINN, along with the inferred source function $Q(x)$. This visualization highlights the model's capability to capture both the observed and unobserved components of the system, showcasing the effectiveness of physics-informed learning in solving inverse problems where classical methods may fall short due to limited or incomplete data.

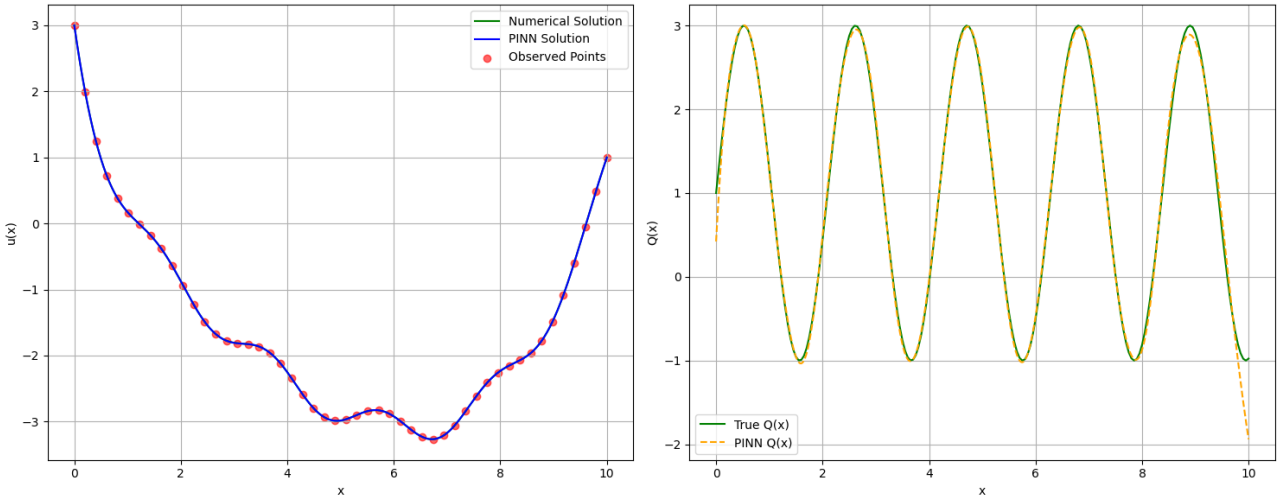

Figure 6.3: PINN-based prediction of solution $U(x)$ and reconstructed source term $Q(x)$

6.2 Experiment 2: Joint estimation of solution and spatially varying coefficient

The second experimental investigation focuses on the simultaneous identification of the spatially varying coefficient $a(x)$ and the corresponding solution $U(x)$ of a governing differential equation. This task represents a class of inverse problems, wherein certain physical parameters are not directly observable and must be inferred from limited data and underlying physical laws. Within this framework, the coefficient $a(x)$ is modeled as an unknown, spatially dependent function, and

is jointly estimated alongside the state variable $U(x)$ through the training process of a PINN.

The implementation of this approach is motivated by the need to address scenarios in which conventional numerical methods struggle to recover hidden system characteristics, particularly in the absence of complete or noise-free measurements. By embedding the differential equation into the loss function, the PINN is capable of learning representations that satisfy both empirical data constraints and the governing physical principles. In this setup, both $U(x)$ and $a(x)$ are treated as outputs of the neural network, and their predictions are optimized with respect to the residual of the differential operator, ensuring that the learned solutions remain consistent with the underlying physics.

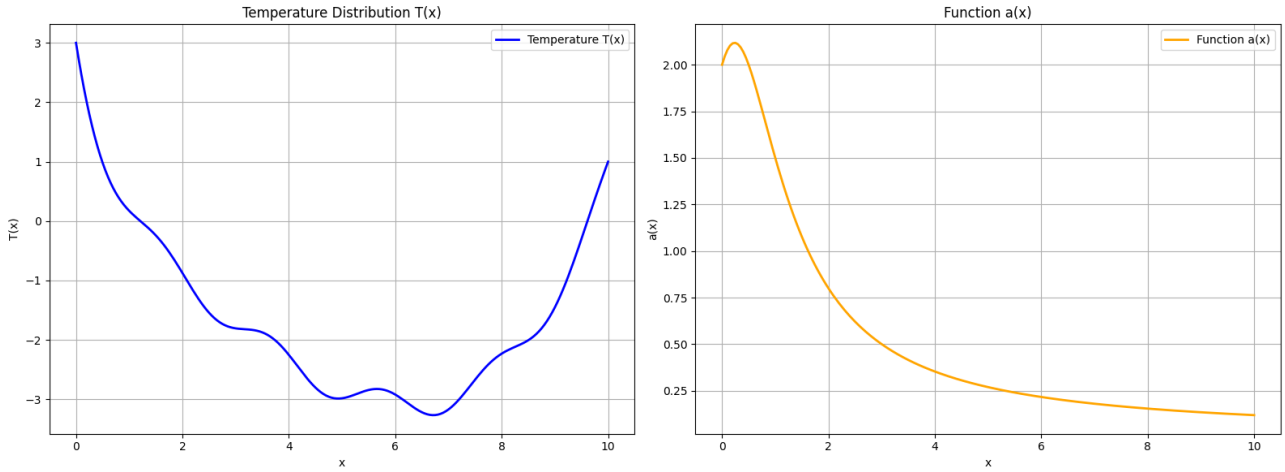

Figure 6.4: Reference solution and varying coefficient $a(x)$ obtained using FDM

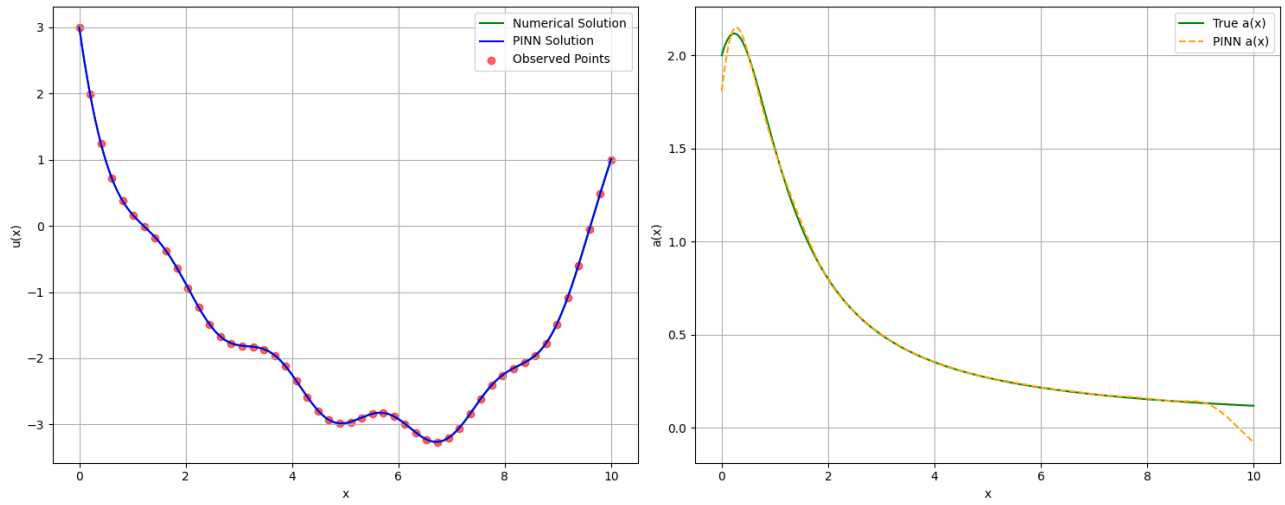

Figure 6.5: Visualisation of PINN Solution and prediction of a(x)

The training process was conducted over 40,000 epochs utilizing a gradient-based optimization algorithm. Initially, the model exhibited elevated loss values, reflective of a limited prior approximation of the system behavior. However, a

progressive decline in the total loss was observed throughout training, culminating in a final value of 3.1563×10^{-2}. This convergence behavior evidences the model's capability to effectively internalize the governing physics and accurately estimate both the primary solution and the latent parameter function.

Figure 6.4 presents the reference solution and the spatially varying coefficient $a(x)$ computed using the FDM, serving as a benchmark for evaluating the accuracy of the PINN approach. The FDM solution, derived through a well-established discretization technique, provides a reliable and highly accurate approximation of the true behavior of the system under investigation.

Building on this benchmark, the outcomes of the PINN-based experiment, as depicted in Figure 6.5, demonstrate the model's capacity to recover both the solution $U(x)$ and the underlying coefficient function $a(x)$ with commendable precision. The close visual and quantitative agreement between the PINN results and the FDM reference validates the effectiveness of the data-driven model in capturing the governing physics.

7. Discussion

The comparative analysis between the FDM and PINN reveals insightful distinctions regarding their capabilities, strengths, and limitations. FDM, as a classical numerical technique, has long been established as a reliable and efficient method for solving PDEs, particularly when the problem domain is well-defined, the boundary conditions are clearly specified, and the physical parameters are known in advance. Its strength lies in its mathematical rigor, high accuracy, and computational speed for structured and discretized domains. The method's foundation in finite difference approximations allows for the precise modeling of spatial and temporal derivatives, making it ideal for deterministic simulations in controlled settings.

However, while FDM excels in such traditional scenarios, it is inherently limited when faced with problems that involve uncertainty, incomplete data, or unstructured and complex geometries. In contrast, PINNs present a fundamentally different approach by leveraging the power of deep learning while embedding the governing physical laws directly into the training process. This hybridization of machine learning and physics allows PINNs to operate effectively even in situations where conventional methods struggle—such as in inverse problems, where some system parameters or boundary conditions are unknown, or in data-driven scenarios where only partial measurements are available.

One of the most significant advantages of PINNs is their flexibility and adaptability. Because they incorporate the PDE constraints as part of the loss function during training, they do not require a predefined mesh or strict discretization, making them well-suited to handle irregular domains or dynamically changing environments. Moreover, their ability to perform *multi-task learning*—simultaneously learning the solution to the PDE and estimating unknown parameters—sets them apart from traditional solvers. This capability was demonstrated clearly in our experiments, where PINNs successfully inferred hidden parameters while also reconstructing the solution field, despite limited access to ground truth data. Such performance showcases their potential as a powerful tool for scientific machine learning and data assimilation tasks.

That said, it is important to acknowledge that PINNs are not without their challenges. In terms of raw numerical precision, especially in low-noise and well-conditioned environments, they may not always outperform conventional solvers

like FDM. Their training process can be computationally expensive and sensitive to hyperparameters, neural network architecture, and the choice of optimization techniques. Additionally, convergence issues and difficulties in capturing sharp gradients or discontinuities can arise, particularly in stiff or highly nonlinear systems. Despite these drawbacks, the overall robustness of PINNs in handling noisy, sparse, or incomplete data makes them a compelling alternative or complement to traditional approaches.

Our findings strongly suggest that PINNs hold significant promise for a range of applications where classical methods are either infeasible or insufficient. These include, but are not limited to, problems involving parameter inference, extrapolation to unobserved regions, and integration of experimental measurements into physical simulations. Their capability to generalize beyond the training data and to embed domain knowledge directly into the learning process opens up new possibilities in modeling real-world phenomena, especially in fields like geophysics, biomedical engineering, and climate science.

8. Conclusion

This thesis has examined the application of PINNs and the FDM for solving the Poisson equation across both one- and two-dimensional spatial domains. Through a systematic comparative analysis, it was observed that while FDM, a well-established numerical technique, consistently yielded higher accuracy in the test cases considered, PINNs demonstrated notable potential due to their inherent flexibility and data-driven nature.

In the one-dimensional case, the FDM approach achieved a remarkably low relative error of 7.26×10^{-8}, compared to 5.63×10^{-6} obtained using the PINN method. Similarly, in the two-dimensional domain, FDM produced a relative error of 2.21×10^{-4}, significantly outperforming the PINN's error of 6.01×10^{-3}. These quantitative findings confirm that FDM offers more precise approximations for Poisson-type problems under the conditions evaluated.

Beyond standard problem-solving tasks, this work also investigated the relevance of PINNs to multi-task learning scenarios. Specifically, the network was trained to simultaneously approximate the solution and infer unknown components of the governing differential equation. The model effectively reconstructed the source term $Q(x)$ and the spatially varying coefficient $a(x)$, achieving loss reductions to 1.87×10^{-2} and 3.1563×10^{-2}, respectively, over the course of 40,000 training epochs. The PINN framework exhibited strong generalization capabilities, providing accurate predictions even at spatial locations not included in the training set.

In summary, while PINNs do not yet surpass traditional numerical methods like FDM in terms of raw accuracy for Poisson equations, their adaptability and suitability for inverse problems and parameter discovery highlight their promise for more complex PDEs where classical methods may be less effective or inapplicable.

Looking ahead, future research in PINNs prioritize enhancing both their accuracy and computational efficiency. This can be achieved by exploring innovative neural network architectures and more effective training strategies tailored to the unique challenges of solving PDEs with data-driven models. A particularly promising direction involves the creation of hybrid frameworks that combine the strengths of PINNs with those of traditional numerical methods

Such hybrid systems hold the potential to significantly improve the reliability

and generalizability of simulations, especially in real-world applications involving inverse problems, parameter estimation, and scenarios that require multi-task learning. Ultimately, this integration could pave the way for next-generation scientific computing tools that blend the rigor of physics-based modeling with the adaptability of machine learning.

Bibliography

[1] E. C. Zachmanoglou and D. W. Thoe, *Introduction to partial differential equations with applications.* Courier Corporation, 1986.

[2] C. L. Fefferman, J. C. Robinson, J. L. R. Diez, and J. L. Rodrigo, *Partial differential equations in fluid mechanics.* Cambridge University Press, 2018, vol. 452.

[3] X. Yang and K. Yao, "Uncertain partial differential equation with application to heat conduction," *Fuzzy Optimization and Decision Making*, vol. 16, pp. 379–403, 2017.

[4] Y. Achdou, O. Bokanowski, and T. Lelièvre, "Partial differential equations in finance," *The Encyclopedia of Financial Models*, vol. 2, 2012.

[5] W. E. Schiesser, *Partial differential equation analysis in biomedical engineering: case studies with MATLAB.* Cambridge University Press, 2012.

[6] L. Debnath and L. Debnath, *Nonlinear partial differential equations for scientists and engineers.* Springer, 2005, vol. 2.

[7] P.-b. Zhou and P.-b. Zhou, "Finite difference method," *Numerical analysis of electromagnetic fields*, pp. 63–94, 1993.

[8] V. Jagota, A. P. S. Sethi, and K. Kumar, "Finite element method: an overview," *Walailak Journal of Science and Technology (WJST)*, vol. 10, no. 1, pp. 1–8, 2013.

[9] Z.-C. Li and S. Wang, "The finite volume method and application in combinations," *Journal of computational and applied mathematics*, vol. 106, no. 1, pp. 21–53, 1999.

[10] B. Costa, "Spectral methods for partial differential equations," *CUBO, A Mathematical Journal*, vol. 6, no. 4, pp. 1–32, 2004.

[11] T. D. Van, M. Tsuji, and N. D. T. Son, *The Characteristic method and its generalizations for first-order nonlinear partial differential equations.* CRC Press, 1999, vol. 101.

[12] S. K. Godunov and I. Bohachevsky, "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics," *Matematičeskij sbornik*, vol. 47, no. 3, pp. 271–306, 1959.

[13] M. N. Özişik, H. R. Orlande, M. J. Colaço, and R. M. Cotta, *Finite difference methods in heat transfer*. CRC press, 2017.

[14] D. W. Zingg, "Comparison of high-accuracy finite-difference methods for linear wave propagation," *SIAM Journal on Scientific Computing*, vol. 22, no. 2, pp. 476–502, 2000.

[15] M. A. Nabian and H. Meidani, "A deep learning solution approach for high-dimensional random differential equations," *Probabilistic Engineering Mechanics*, vol. 57, pp. 14–25, 2019.

[16] 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.

[17] N. Winovich, K. Ramani, and G. Lin, "Convpde-uq: Convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains," *Journal of Computational Physics*, vol. 394, pp. 263–279, 2019.

[18] S. Cuomo, V. S. Di 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, no. 3, p. 88, 2022.

[19] C.-H. Tsai, H.-T. Chen, Y.-N. Wang, C.-H. Lin, and L.-M. Fu, "Capabilities and limitations of 2-dimensional and 3-dimensional numerical methods in modeling the fluid flow in sudden expansion microchannels," *Microfluidics and Nanofluidics*, vol. 3, pp. 13–18, 2007.

[20] S. T. Abdulazeez and M. Modanli, "Solutions of fractional order pseudo-hyperbolic telegraph partial differential equations using finite difference method," *Alexandria Engineering Journal*, vol. 61, no. 12, pp. 12 443–12 451, 2022.

[21] S. M. Hussain and W. Jamshed, "A comparative entropy based analysis of tangent hyperbolic hybrid nanofluid flow: Implementing finite difference method," *International Communications in Heat and Mass Transfer*, vol. 129, p. 105671, 2021.

[22] Z. Li, Y. Fujii, A. Alam, Z. Li, F. Du, and W. Wei, "Implementing a simple 2d constitutive model for rocks into finite element method," *Computers and Geotechnics*, vol. 167, p. 106095, 2024.

[23] K. Tannhäuser, P. H. Serrao, and S. Kozinov, "A three-dimensional collocation finite element method for higher-order electromechanical coupling," *Computers & Structures*, vol. 291, p. 107219, 2024.

[24] V. Berta, A. Mignone, M. Bugli, and G. Mattia, "A 4th-order accurate finite volume method for ideal classical and special relativistic mhd based on pointwise reconstructions," *Journal of Computational Physics*, vol. 499, p. 112701, 2024.

[25] S. Zhou and Y. Zhang, "A novel and simple spectral method for nonlocal pdes with the fractional laplacian," *Computers & Mathematics with Applications*, vol. 168, pp. 133–147, 2024.

[26] T. Lyons, H. Ni, and J. Tao, "A pde approach for solving the characteristic function of the generalised signature process," *arXiv preprint arXiv:2401.02393*, 2024.

[27] S. Markidis, "The old and the new: Can physics-informed deep-learning replace traditional linear solvers?" *Frontiers in big Data*, vol. 4, p. 669097, 2021.

[28] J. Blechschmidt and O. G. Ernst, "Three ways to solve partial differential equations with neural networks—a review," *GAMM-Mitteilungen*, vol. 44, no. 2, p. e202100006, 2021.

[29] K. Qi and J. Sun, "Gabor-filtered fourier neural operator for solving partial differential equations," *Computers & Fluids*, vol. 274, p. 106239, 2024.

[30] F. Lehmann, F. Gatti, M. Bertin, and D. Clouteau, "3d elastic wave propagation with a factorized fourier neural operator (f-fno)," *Computer Methods in Applied Mechanics and Engineering*, vol. 420, p. 116718, 2024.

[31] E. Haghighat, U. bin Waheed, and G. Karniadakis, "En-deeponet: An enrichment approach for enhancing the expressivity of neural operators with applications to seismology," *Computer Methods in Applied Mechanics and Engineering*, vol. 420, p. 116681, 2024.

[32] A. Peyvan, V. Oommen, A. D. Jagtap, and G. E. Karniadakis, "Riemannonets: Interpretable neural operators for riemann problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 426, p. 116996, 2024.

[33] Y. Jiao, Y. Lai, Y. Lo, Y. Wang, and Y. Yang, "Error analysis of deep ritz methods for elliptic equations," *Analysis and Applications*, vol. 22, no. 01, pp. 57–87, 2024.

[34] T. Hu, B. Jin, and F. Wang, "An iterative deep ritz method for monotone elliptic problems," *Journal of Computational Physics*, p. 113791, 2025.

[35] T. De Ryck and S. Mishra, "Numerical analysis of physics-informed neural

networks and related models in physics-informed machine learning," *Acta Numerica*, vol. 33, pp. 633–713, 2024.

[36] T. G. Grossmann, U. J. Komorowska, J. Latz, and C.-B. Schönlieb, "Can physics-informed neural networks beat the finite element method?" *IMA Journal of Applied Mathematics*, vol. 89, no. 1, pp. 143–174, 2024.

[37] B. Sharimbayev, S. Kadyrov, and A. Kavokin, "Development and optimization of physics-informed neural networks for solving partial differential equations," *arXiv preprint arXiv:2502.02599*, 2025.

[38] R. DeVore, B. Hanin, and G. Petrova, "Neural network approximation," *Acta Numerica*, vol. 30, pp. 327–444, 2021.

[39] D. Elbrächter, D. Perekrestenko, P. Grohs, and H. Bölcskei, "Deep neural network approximation theory," *IEEE Transactions on Information Theory*, vol. 67, no. 5, pp. 2581–2623, 2021.

[40] B. Sharimbayev, "Github: https://github.com/hardkazakh/pinn-vs-fdm," 2023.

[41] M. D. Shields and J. Zhang, "The generalization of latin hypercube sampling," *Reliability Engineering & System Safety*, vol. 148, pp. 96–108, 2016.

Article

Development and optimization of physics-informed neural networks for solving partial differential equations

Batyr Sharimbayev* [1], Shirali Kadyrov [2], and Aleksei Kavokin [1]

[1]Department of Mathematics and Natural Sciences, Almaty, Kazakhstan
[2]Department of General Education, New Uzbekistan University, Tashkent, Uzbekistan

DOI: 10.47344/2ekq1942

Abstract

This study investigates the application of physics-informed neural networks (PINNs) for solving Poisson equations in both 1D and 2D domains and compares them with finite difference method. Additionally, the study explores the capability of multi-task learning with PINNs, where the network not only predicts the solution but also estimates unknown parameters. In the case of a second-order differential equation with a varying coefficient, PINNs successfully approximated both the source term and the varying coefficient while achieving low training loss. The model demonstrated excellent generalization capabilities and accurate reconstruction of the underlying system parameters, showing the potential of PINNs in complex physical simulations.

Keywords: numerical analysis, multi-task learning, deep learning, PINNs, FDM

I. Introduction

PDEs form a crucial backbone in the understanding and modeling of various real-world problems. In simple words, PDEs describe how something changes through time and space based on very well-defined mathematical rules. The central issue with PDEs is determining whether these rules suffice to guarantee a unique solution to the problem [1] [2].

In most real-life cases, it is impossible to find the exact solutions for such complex equations involving PDEs. This gave rise to the development of numerical methods for estimating solutions. Some of the popular methods include the finite element method, finite difference method, finite volume method, and spectral element method. Out of these, FEM is the most advanced with strong mathematical support for ensuring accurate results, stability, and error control. FEM solvers often employ efficient techniques such as sparse linear systems or iterative methods, which make them suitable for many practical problems [3].

Most recently, new techniques have emerged as a result of the growth of deep learning which aid in the resolution of PDEs. One of these is Physics-Informed Neural Networks (PINNs). PINNs are neural networks which, by means of their loss function,

Email: batyr.sharimbayev@sdu.edu.kz ORCID: 0009-0006-3323-231X
Email: sh.kadyrov@newuu.uz ORCID: 0000-0002-8352-2597
Email: aleksei.kavokin@sdu.edu.kz ORCID: 0009-0000-1931-4518

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

encode the physical laws described by differential equations to drive the learning process towards solutions that better respect the underlying physics. PINNs can approximate solutions to partial differential equations and ordinary differential equations and solve inverse problems, e.g., estimating model parameters from scarce data. They also happen to be tremendously useful for solving various kinds of PDEs because of their simplicity in implementation and direct incorporation of the underlying physics within the learning itself.

Despite their adoption, deep learning algorithms, including PINNs, still suffer from high computational costs, complex optimization, and weak theoretical foundations. This study aims to achieve two main objectives. First, it aims to rigorously evaluate the performance of PINNs in solving PDEs by comparing their accuracy, convergence, and computational efficiency to traditional numerical methods, particularly the Finite Difference Method (FDM). By applying PINNs to 1D and 2D Poisson equations, the study assesses their accuracy, error metrics, and computational demands while examining their robustness across different boundary conditions and domain sizes. Second, the research explores the integration of multi-task learning (MTL) [5] within the PINN framework to enhance its ability to solve ODE while simultaneously predicting related physical quantities, such as source terms and varying coefficients. By leveraging shared information across tasks, the study aims to improve the generalization capabilities of PINNs and develop a unified model capable of solving complex multi-dimensional problems in a single optimization process. We build upon our prior work [4] and provide additional analysis.

II. Related Works

The increasing use of neural networks has prompted researchers to utilize a variety of techniques in deep learning for solving mathematical equations. Raissi et al. [6] present PINNs as a broad framework in deep learning to solve nonlinear forward and inverse partial differential equations (PDEs). With integration of basic principles from physics, PINNs provide a mathematically strong and data-effective approach to solving complex spatio-temporal problems using continuous-time and discrete-time Runge–Kutta methods. This method has been validated by extensive testing across a variety of applications in PDEs, ranging from fluid dynamics to quantum mechanics.

Ryck et al. [7] present a rigorous numerical analysis of PINNs, systematically classifying errors into approximation, generalization, and training errors. They critically examine how PDE characteristics and domain dimensionality influence accuracy, identifying training error as a significant constraint on PINN performance. Their findings underscore the importance of solution regularity and stability in ensuring the reliability of PINN-based computations. Hu et al. [8] investigate the application of PINNs in computational solid mechanics, addressing limitations associated with sparse, noisy, and high-dimensional data. By integrating prior physical knowledge, PINNs enhance model generalizability, enforce physical consistency, and improve computational efficiency. The study provides a comprehensive review of PINN architectures, algorithmic advancements, and their implementation in constitutive modeling, damage evaluation, and inverse problem-solving. Cuomo et al. [9] provide a thorough review of PINNs and explore their use in solving a wide variety of partial differential equation (PDE) problems, both fractional and stochastic types. The paper also discusses developments like Physics-Constrained Neural Networks (PCNNs) and variational hp-VPINNs and touches upon optimization algorithms, network architectures, and loss function setup. Though PINNs have already shown great promise, the paper identifies open problems to be solved in order to enhance their reliability and usability.

Grossmann et al. [10], past comparisons between PINNs and other numerical methods have shown that each approach has strengths and weaknesses. However, the paper does not clearly explain these differences, making it harder to assess how competitive PINNs are for different types of problems. While the study presents specific cases where PINNs struggle or perform well, it lacks a systematic discussion of key factors such as computational cost, accuracy, stability, and problem structure. Without this detailed comparison, it is difficult to determine in which scenarios PINNs might be advantageous or where traditional methods like FEM remain superior. The findings suggest that FEM remains the more reliable method, particularly for high dimensional PDEs. In light of the findings from previous works, we conduct a comprehensive comparison between the well-established FDM and the state-of-the-art PINNs in the context of solving non-complex PDEs, where FDM has demonstrated superior accuracy and efficiency. The FDM, known for its robustness and reliability, has long been a standard technique for numerically solving PDEs with relatively simple geometries and boundary conditions. On the other hand, PINNs represent an emerging approach in deep learning. Additionally, we extend our comparison by framing the development of the PINN method within the context of multi-task learning (MTL). In this setting, the PINN is treated as an MTL model, enabling it to simultaneously learn from multiple related tasks. This approach not only facilitates a more efficient learning process but also allows for the adaptation of the PINN to a broader range of problems by sharing knowledge between tasks.

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

III. Mathematical Background

In this section, we explain the mathematical background of the methods used in this work.

A. Poisson Equation

The Poisson equation is one of the most basic PDEs, which results in a potential field created by a given source. It provides significant applications to physics and engineering, modeling electrostatics, heat conduction, and fluid dynamics among many others [11] [12].

In one dimension, the Poisson equation is given by:

$$\frac{d^2u(x)}{dx^2} = f(x) \tag{1}$$

where $u(x)$ is the unknown function, $f(x)$ is a source term.

In such cases, the domain is normally an interval $[a, b]$ and the boundary conditions are often given at both ends, say in the following form of Dirichlet boundary conditions:

$$u(a) = u_0, \quad u(b) = u_1$$

In two dimensions, the Poisson equation generalizes to:

$$\nabla^2 u(x, y) = f(x, y) \tag{2}$$

where ∇^2 is the 2D Laplacian operator, $u(x, y)$ represents the unknown function, and $f(x, y)$ is called the source term. In two dimensions, it can normally be a rectangular region or even more complicated geometry. On edges, boundary conditions are imposed. There may also be **Dirichlet** boundary conditions on these. A typical **Dirichlet** boundary condition takes the following form: $u(x, y) = g(x, y)$ on the boundary of the domain.

The accuracy of the solution approximations for the 1D and 2D Poisson equations is evaluated using the L_2 norm and relative error. The L_2 norm of a vector $v = [v_1, v_2, \ldots, v_n]$ is defined as:

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} v_i^2}.$$

This norm gives the size of the magnitude of a vector in Euclidean space. To verify these results for accuracy, we calculate the L_2 relative error between the computed solution $\hat{u}$ and the exact solution u defined as:

$$L_2 = \frac{\|\hat{u} - u\|_2}{\|u\|_2}. \tag{3}$$

B. Physics-Informed Neural Networks

Let $u_\theta(x, y)$ denote the neural network approximation of $u(x, y)$ where θ is used to denote the parameters of the network. The network architecture considered here is a fully connected feed-forward neural network, such that the application of activation functions is performed layer by layer. It can be written as:

$$u_\theta(x, y) = \text{NN}_\theta(x, y)$$

The network layers are defined as follows:

$$x^{(i+1)} = \sigma\left(w^{(i)}x^{(i)} + b^{(i)}\right)$$

for each hidden layer, and the output layer is linear. The residual of the PDE is defined as:

$$R(x, y) = \frac{\partial^2 u_\theta}{\partial x^2} + \frac{\partial^2 u_\theta}{\partial y^2} - f(x, y)$$

where $f(x, y)$ is the source term. The PDE loss is computed as the mean squared error of the residual over a set of domain points (x_i, y_i):

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

$$L_{\text{PDE}} = \frac{1}{N} \sum_{i=1}^{N} R(x_i, y_i)^2 \tag{4}$$

The boundary conditions loss is computed similarly, where the loss for boundary points (x_j, y_j) is given by:

$$L_{\text{BC}} = \frac{1}{M} \sum_{j=1}^{M} \left(u_\theta(x_j, y_j) - u(x_j, y_j)\right)^2 \tag{5}$$

The total loss function combines the PDE residual loss and the boundary conditions loss:

$$L(\theta) = L_{\text{PDE}} + L_{\text{BC}} \tag{6}$$

C. Finite Difference Method (FDM)

The FDM is a numerical method for the solution of PDEs that is based on discretizing their solutions on a lattice of points that discretize the domain. Based on this concept, the FDM relies upon approximating the derivatives of an unknown function in terms of finite differences that replace the PDE by a set of algebraic equations.

First, consider the one-dimensional Poisson equation:

$$\frac{d^2u(x)}{dx^2} = f(x)$$

Using a finite difference scheme, the second derivative can be approximated by the following way:

$$\frac{d^2u(x)}{dx^2} \approx \frac{u(x + \Delta x) - 2u(x) + u(x - \Delta x)}{(\Delta x)^2} \tag{7}$$

where Δx is the step size and $u(x)$ is the unknown function at the grid points. By discretizing the domain $[a, b]$ with a grid of points, we convert the continuous PDE into a system of algebraic equations which can be solved numerically.

In two dimensions, the Poisson equation is given by:

$$\nabla^2 u(x, y) = f(x, y)$$

where ∇^2 is the Laplacian operator:

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

The second derivatives in the x and y directions are approximated by finite differences. The Laplacian operator in 2D is approximated as:

$$\nabla^2 u(x, y) \approx \frac{u(x + \Delta x, y) + u(x - \Delta x, y) - 2u(x, y)}{(\Delta x)^2} + \frac{u(x, y + \Delta y) + u(x, y - \Delta y) - 2u(x, y)}{(\Delta y)^2} \tag{8}$$

where Δx and Δy are increments in steps in directions x and y, respectively.

The FDM discretizes both the domain for solving both the 1D and 2D Poisson equation and then solves for resulting algebraic equations. Iterating through each grid point, $u(x)$ or $u(x, y)$ values would be computed based on the finite difference approximation offered by the PDE. Boundary conditions would be specified along the domain's boundary in terms of either Dirichlet conditions.

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

IV. Methods

This paper focuses on constructing and improving PINNs for solving the 1D and 2D Poisson equations and comparing them with the FDM.

To begin with, FDM is the first approach we had to study to solve the Poisson equation. The domain of the 1D example is split into a lattice grid, and the equation is solved through second-derivative approximation. The same procedure is applied in 2D, in which both spatial dimensions are discretized. The numerical solution is obtained and compared to the calculated exact solutions in order to check accuracy using the L_2 relative error.

At this stage, the PINN is implemented to solve the equations. The neural network is trained by minimizing a loss function that comprises the Poisson equation and the boundary conditions. The training is therefore carried out in two steps: firstly, with the Adam optimizer to fit the parameters of the model, and subsequently with the L-BFGS method for fine-tuning.

Latin Hypercube Sampling, in both 1D and 2D scenarios, provides points for model analyses. The network is trained to provide a set of boundary conditions. The performance of the model is determined by comparing the output of the neural network with a known solution using the relative L_2 norm.

Ultimately, we applied the FDM to a thermal problem involving a second-order ordinary differential equation (ODE). We split the spatial domain into parts, provided intervals, and boundary conditions at both ends. The problem was solved iteratively until the difference between subsequent values was less than some small predefined threshold. The material properties and source term were represented as functions of spatial position. The PINN solution was compared to the one obtained using multivariate interpolation to assess its accuracy. Then, the forward and inverse problem was solved using a PINN structure. The PINN was based on a feedforward neural network with hidden layers that were trained to estimate temperature and the source term via a learned loss function. The model was validated against FDM and observational data. The code was written in Python and is available on GitHub: https://github.com/hardkazakh/pinn-vs-fdm

V. Experimental Results

In this section, we will solve 1D and 2D Poisson equations using FDM and PINNs. We will also use the PINNs approach to multi-task learning.

A. 1D Poisson equation

Let us investigate the 1D Poisson equation defined as:

$$\frac{d^2u(x)}{dx^2} = 16x^7e^{-x^4} - 20x^3e^{-x^4}, \quad x \in [0,1], \tag{9}$$

with Dirichlet boundary conditions:

$$u(0) = 0, \quad u(1) = e^{-1}.$$

In Section III-C, we explained that the first step to solving PDEs with the FDM is to rewrite the equation in a weak form. We already did this for the Poisson equation. The next step is to create a mesh. This is like breaking the interval $[0, 1]$ into small pieces, called cells. The number of cells is 512. More cells mean the grid is finer, which gives a more accurate solution, but it also takes more time and computing power.

For solving the 1D Poisson equation using PINNs, there are three design parameters that we need to specify before training. The first step is choosing a loss function. Following the vanilla PINNs approach, we evaluate the goodness of the solution using the discretised mean squared error over the PDE, boundary, and initial conditions.

The second design parameter is the neural network architecture, that is, the type of neural network, the activation function, and the number of hidden layers and nodes. For the 1D Poisson case, we train feed-forward dense neural networks with $tanh$ as the activation function. We use the result on the architecture of [20, 20, 20, 1].

The approximations of the 1D Poisson equation solution using FDM and PINNs are compared to the exact solution on a $[0, 1]$ interval with 512 points. Fig 1 shows the exact solution and the approximations. One PINN setup, with only one hidden layer and one node, performs poorly and fails to satisfy the boundary conditions. All approximations are very close to the exact solution.

For the 1D Poisson equation, the relative error for the FDM is calculated to be 7.26×10^{-8}, while the relative error for the PINN approach is 5.63×10^{-6}. These results show that FDM provides a more accurate approximation of the solution compared to PINNs in the 1D case.

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

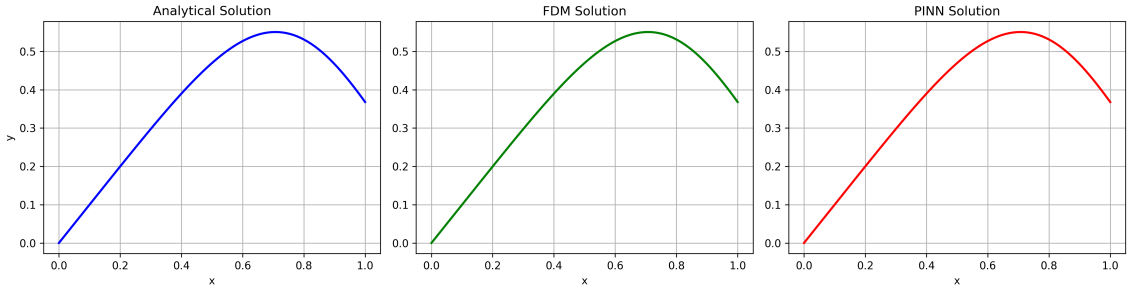

Fig. 1. Comparison of solutions: 1-Exact, 2-FDM, and 3-PINN

B. 2D Poisson equation

Let us now investigate 2D Poisson equation defined as:

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 2\left(x^4(3y-2) + x^3(4-6y) + x^2\right). \tag{10}$$

The boundary conditions are:

$$u(x,0) = u(x,1) = u(0,y) = u(1,y) = 0.$$

The analytical solution of the equation is:

$$u(x,y) = (x-1)^2 y (y-1)^2 x^2.$$

The approximations of the 2D Poisson equation solution using FDM and PINNs are compared to the exact solution on the $[0,1] \times [0,1]$ domain, discretized with a 1000×1000 grid. The neural networks are trained using the $\tanh$ activation function with an architecture of $[60, 60, 60, 1]$.

Fig 2 shows the analytical and approximate solutions of the 2D Poisson equation. For the 2D Poisson equation, the L_2 relative error for the FDM approximation is 2.21×10^{-4}, while for the PINN, it is 6.01×10^{-3}. Again, the FDM method shows a significantly lower error compared to the PINN, indicating that FDM achieves a more precise solution in both 1D and 2D cases.

These error analyses again confirm the accuracy of the FDM approach in solving Poisson equations, especially in comparison with PINNs, which showed higher relative errors in both 1D and 2D problems. However, in higher-order equations, PINN can give better results. We consider this in our future research.

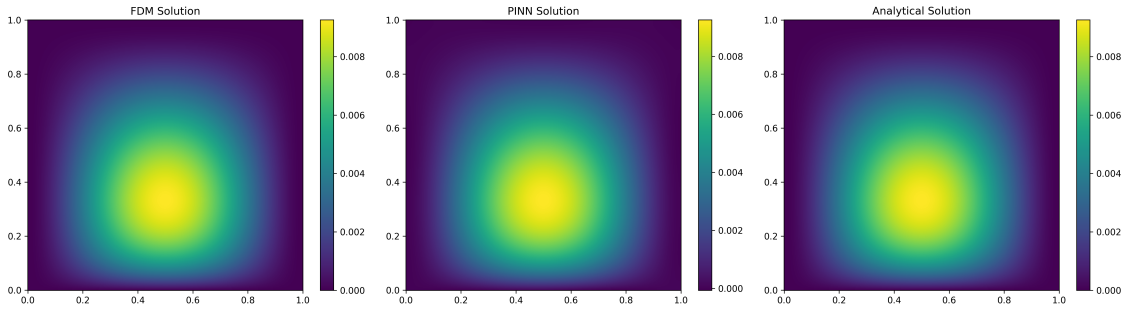

Fig. 2. Comparison of solutions: 1-PINN, 2-Exact, and 3-FDM.

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

C. PINNs approach for multi-task learning (MTL)

PINNs embed both forward and inverse problems under one framework by embedding the data and physical laws in the loss function of the neural network. The network jointly predicts the forward solution of a PDE and estimates unknown parameters or inputs (inverse problem) through the optimization of one combined loss function that considers residual of PDE, boundary conditions, and discrepancies between model predictions and observations. This capability of handling both tasks together makes PINNs highly effective at solving problems that involve solution estimation and identification of parameters.

The problem is defined by the following second-order differential equation:

$$\frac{d^2U(x)}{dx^2} - a(x)U(x) = Q(x), \quad 0 < x < L, \tag{11}$$

where $Q(x)$ represents the source term and $a(x)$ describes the varying coefficient. Specifically, the source term is given by:

$$Q(x) = 1 + b_1 \sin(w_1 x)$$

and the varying coefficient is defined as:

$$a(x) = b_1 + \frac{x}{1 + x^2}.$$

The boundary conditions for the differential equation are given by: $U(0) = 1, U(L) = 3$. The problem now involves the solution of this second-order differential equation along with the boundary conditions shown above. $Q(x)$, the source function, incorporates a sine function that could model some periodic influence in the system. This coefficient, $a(x)$, depends on the position x in the domain; therefore, this makes the equation more complicated, introducing spatial dependence in the solution.

The aim of Experiment 1 is to approximate both the source term $Q(x)$ and the solution $U(x)$ for the given differential equation. PINN simultaneously predicts the solution $U(x)$ while reconstructing the source term $Q(x)$ using the varying coefficient $a(x)$ as part of the system.

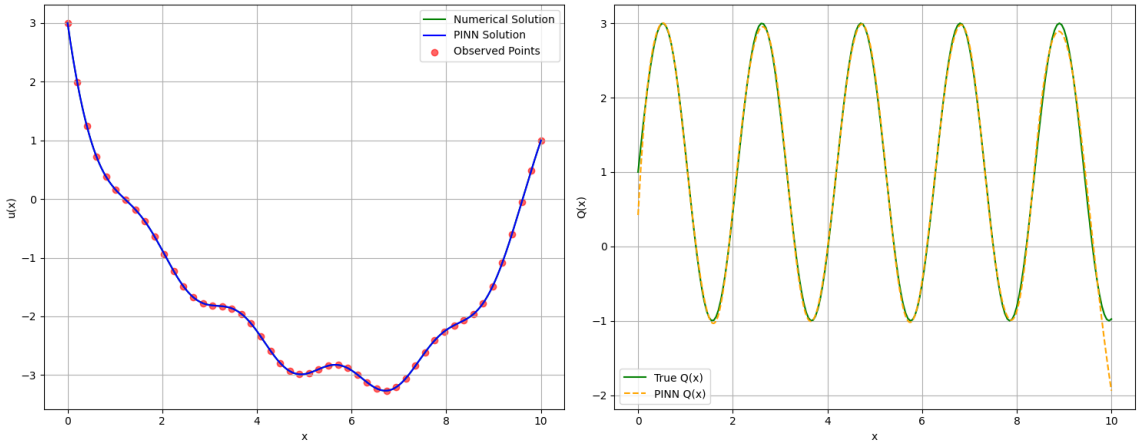

Fig. 3. Visualization of FDM Solution and varying coefficient of a(x)

The PINN model was trained for 40,000 epochs. Initially, the loss function was quite large, but it gradually decreased as the model learned. By the end of the training, the loss had reduced to 1.87×10^{-2}, indicating that the model had learned the underlying physics of the problem. The predictions made by the PINN for the differential equation solution were then analyzed and aligned with the FDM results. These predictions were accurate not only at the observed points but also at unobserved locations, highlighting the PINN's ability to generalize well across the entire domain. Fig 3 presents the visualization of the PINN's predicted temperature distribution and the corresponding predicted source term.

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

The aim of Experiment 2 is to approximate both the varying coefficient $a(x)$ and the solution $U(x)$. By incorporating $a(x)$ as an unknown parameter in the system, the model predicts the solution $U(x)$ while reconstructing $a(x)$ from the given data.

The PINN model was trained for 40,000 epochs. Initially, the loss function exhibited high values, but with training, it gradually decreased as the model captured the intricate relationships within the system. By the end of the training, the loss had reduced to 3.1563×10^{-2}, demonstrating that the model effectively learned both the solution and the coefficient.

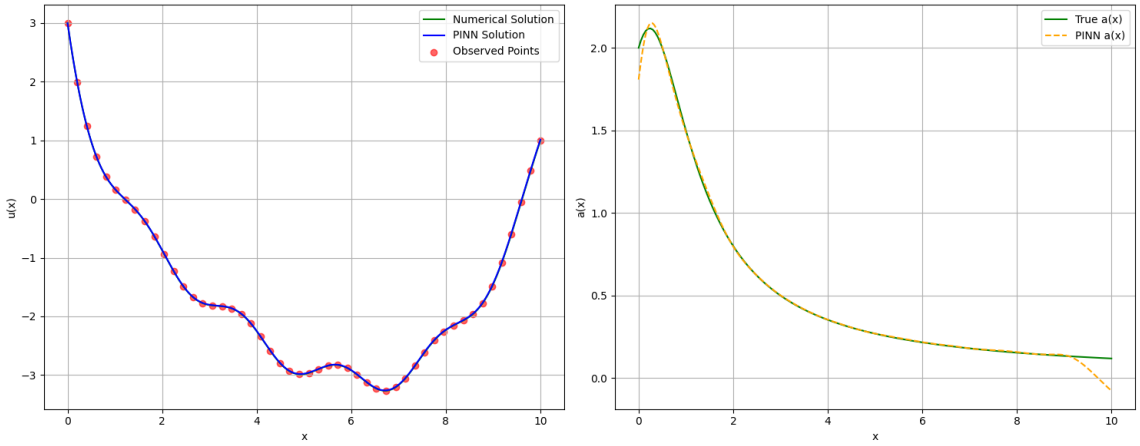

Fig. 4. Visualization of PINN Solution and prediction of a(x)

Fig 4 illustrates the visualization of the PINN's predictions for the temperature distribution and the reconstructed varying coefficient. The predicted solution from the PINN closely aligns with the numerical solution obtained using FDM.

VI. Discussion & Conclusion

In this paper, we have discussed using PINNs and FDM to solve the problem in both 1D and 2D. From our comparison, it was illustrated that even though PINNs offer a flexible and data-driven approach, traditional numerical methods such as FDM offer improved accuracy in such a case.

For the 1D Poisson equation, FDM's relative error was calculated as 7.26×10^{-8} and PINN's relative error was calculated as 5.63×10^{-6}. In a similar manner, for the 2D Poisson equation, FDM's relative error was calculated as 2.21×10^{-4}, which was much lesser than PINN's error value, which was 6.01×10^{-3}. These findings establish that FDM gives a better approximation in problems involving Poisson.

Additionally, we have explained multi-task learning with PINNs. We have witnessed that PINN reconstructed variable $a(x)$ and source term $Q(x)$ with decreasing loss to 1.87×10^{-2} and 3.15×10^{-2}, respectively, in 40,000 epochs. The PINN model had strong generalization capability and predicted with high accuracy even at unseen locations. Regarding the computational complexity, PINN's training to solve the 2D Poisson equation was very time-consuming. This reflects high computational cost in high-dimensional PDEs, whose practicality and efficiency depend greatly upon high-end GPU access. The longer time to train reflects a major disadvantage of PINNs compared to standard numerical methods in problems requiring quick and scalable solutions. We believe that PINNs have more errors than FDM because it has problems in optimizations and it has problems in fine solution feature capture.

Overall, our findings indicate that while PINNs do not presently outperform FDM in terms of accuracy for non-complex equations because they can be flexible and can perform multi-task learning, they provide a reasonable alternative to complex PDEs when conventional numerical methods have limitations.

In the present study, our aim will be to improve PINN's performance and efficiency with more advanced neural network architectures and learning techniques to solve other PDEs. One direction would be to design hybrid algorithms by combining PINNs with traditional numerical methods such as FDM. With FDM's high-precision performance in structured grid regions and PINNs's adaptability in unstructured or sparsity regions, a more effective and robust framework can be established. This hybrid

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.

method can rectify PINN's limitation in high-precision attainment without sacrificing PINN's ability to solve inverse problems and multi-task learning.

VII. Acknowledgment

"This research was funded by the Ministry of Science and Higher Education of the Republic of Kazakhstan within the framework of project AP23487777."

References

[1] S.L. Brunton, "Promising directions of machine learning for partial differential equations," Nature Computational Science, 2024.

[2] J. Blechschmidt, "Three ways to solve partial differential equations with neural networks—A review," GAMM-Mitteilungen, 2021.

[3] A. Björck, "Numerical methods for least squares problems," SIAM, 2024.

[4] B. Sharimbayev, "Development and optimization of physics-informed neural networks for solving partial differential equations," arXiv preprint arXiv:2502.02599, 2025.

[5] S. Ruder, "An overview of multi-task learning in deep neural networks," arXiv preprint arXiv:1706.05098, 2017.

[6] M. Raissi, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational physics, 2019.

[7] T. De Ryck, "Numerical analysis of physics-informed neural networks and related models in physics-informed machine learning," Acta Numerica, 2024.

[8] H. Hu, "Physics-informed Neural Networks (PINN) for computational solid mechanics: Numerical frameworks and applications," Thin-Walled Structures, 2024.

[9] S. Cuomo, "Scientific machine learning through physics–informed neural networks: Where we are and what's next," Journal of Scientific Computing, 2022.

[10] T.G. Grossmann, "Can physics-informed neural networks beat the finite element method?," IMA Journal of Applied Mathematics, 2024.

[11] W. Hackbusch, "Elliptic differential equations: theory and numerical treatment," Springer, 2017.

[12] T. Matsuura, "Numerical solutions of the Poisson equation," Applicable Analysis, 2004.

Received: February 13, 2025. Reviewed: March 20, 2025. Accepted: April 30, 2025.