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:

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

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.

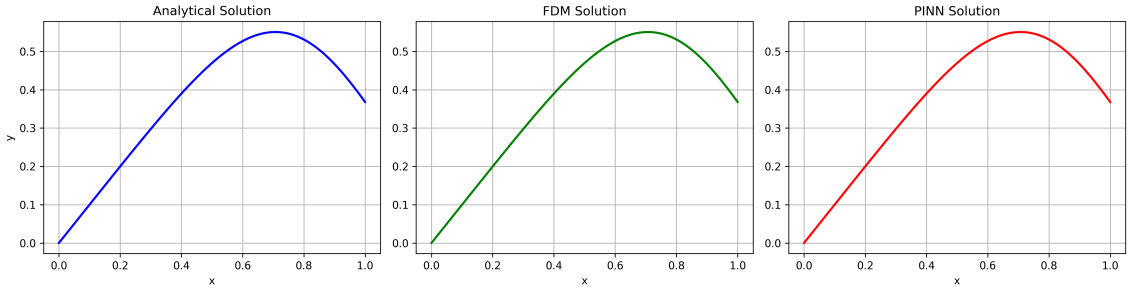

Fig. 1. Comparison of solutions: 1-Exact, 2-FDM, and 3-PINN

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.

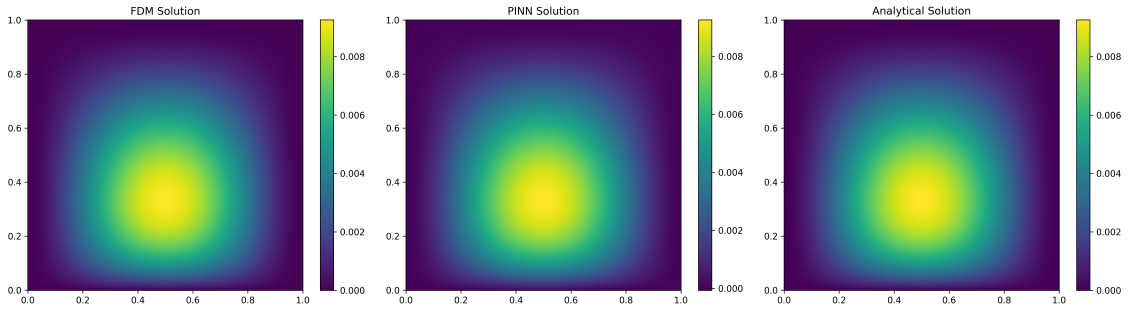

Fig. 2. Comparison of solutions: 1-PINN, 2-Exact, and 3-FDM.

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.

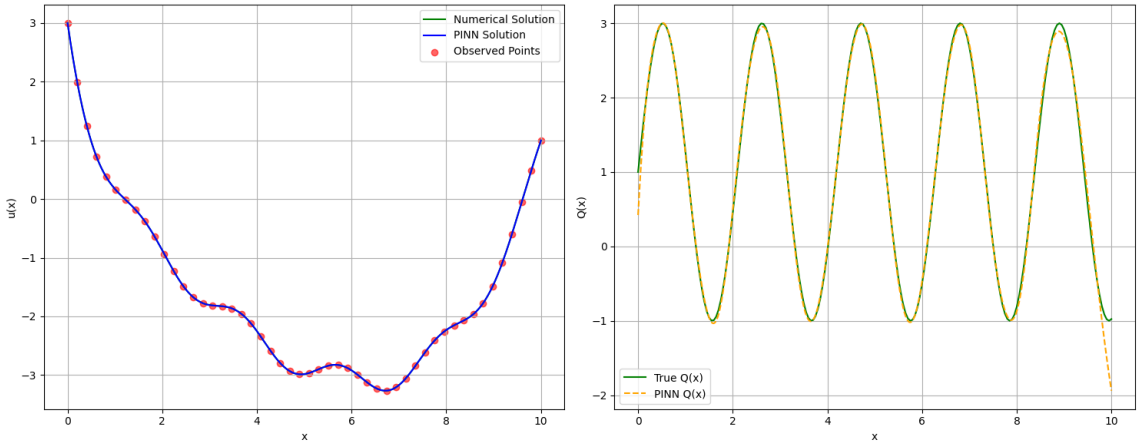

Fig. 3. Visualization of FDM Solution and varying coefficient of a(x)

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.

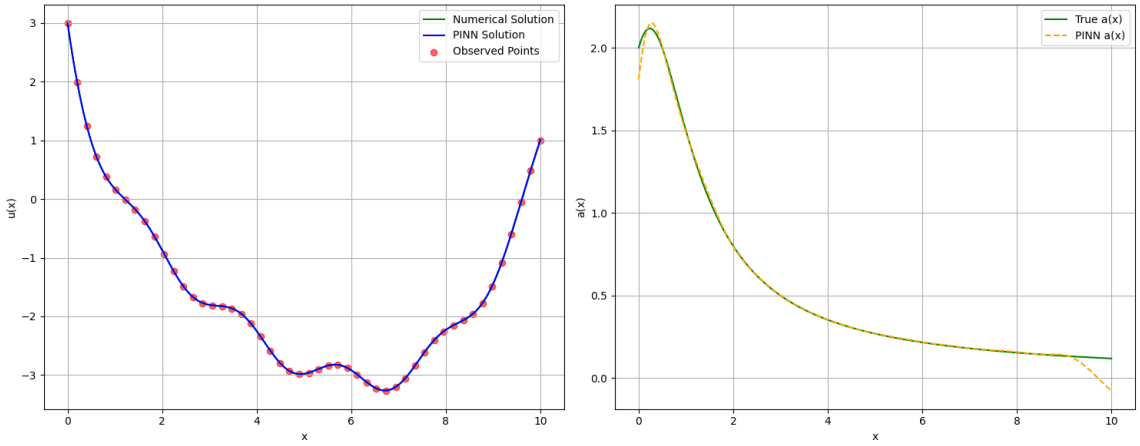

Fig. 4. Visualization of PINN Solution and prediction of a(x)

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.