RESEARCH IN STATISTICS
2025, VOL. 3, NO. 1, 2542577
https://doi.org/10.1080/27684520.2025.2542577

Taylor & Francis
Taylor & Francis Group

Solving Steady-State Elliptic Problems in Irregular Domains Using Physics-Informed Neural Networks and Fictitious Domain Methods

José A. Rodrigues

CIMA and Department of Mathematics of ISEL, Polytechnic University of Lisbon, Portugal.

ABSTRACT

This paper introduces an innovative methodology for solving steady-state elliptic partial differential equations defined over irregular domains, by coupling the capabilities of Physics-Informed Neural Networks with the Fictitious Domain Method. The primary emphasis is placed on applications involving the heat equation, a fundamental model in thermal analysis where the complexity of non-standard geometries often poses significant challenges for traditional numerical methods. The proposed approach exploits the inherent strength of Physics-Informed Neural Networks in embedding the underlying physical laws directly into the learning process, enabling the model to approximate solutions without relying on mesh-based discretization. Simultaneously, the Fictitious Domain Method facilitates the treatment of irregular computational domains by embedding them within a larger, regular domain, thereby simplifying the application of boundary conditions and numerical operations. The synergy between these two techniques results in a flexible, efficient, and accurate computational framework that is well-suited for addressing heat transfer problems in complex geometrical configurations.

ARTICLE HISTORY
Received 18 September 2024
Accepted 25 July 2025

KEYWORDS
Physics-Informed Neural Networks, Fictitious Domain Method, Irregular Domains, Steady-State Heat Conduction, NURBS Curves.

1. Introduction

Elliptic partial differential equations (PDEs), particularly the steady-state heat equation, form the foundation for many physical and engineering applications, such as heat conduction, electrostatics, and fluid flow. The steady-state heat equation, which describes how heat diffuses in a material until a stable temperature distribution is reached, is crucial in fields like thermodynamics, building energy simulations, and material science. In these applications, the governing equation is elliptic in nature and generally takes the form:

$$-\nabla \cdot (k\nabla u) = f \quad \text{in} \quad \Omega \tag{1}$$

where u represents the temperature distribution, k is the thermal conductivity of the material, and f is an external heat source within a given domain Ω.

While there are well-established numerical methods for solving these elliptic equations, such as the Finite Element Method (FEM) and Finite Difference Method, they are often designed to work efficiently on regular, structured domains. For domains with complex, irregular geometries—such as those encountered in realistic engineering designs or biological structures—traditional methods pose significant challenges. Meshing these irregular domains becomes computationally expensive, and often, such meshing techniques may introduce numerical errors due to boundary approximation, mesh distortions, or the need for higher mesh resolution near complex boundaries. This not only increases the computational cost but also complicates the implementation of boundary conditions, especially when non-standard boundary conditions are required.

Moreover, these traditional methods struggle to handle geometric complexities efficiently without resorting to sophisticated meshing strategies, which may not scale well for high-dimensional or multi-scale problems. The demand for innovative numerical methods that can bypass the challenges of meshing and can flexibly handle irregular domains has led to the exploration of alternative approaches, such as Physics-Informed Neural Networks and the Fictitious Domain Method.

This paper aims to integrate Physics-Informed Neural Networks (PINNs) with the Fictitious Domain Method (FDM) to solve the steady-state heat equation in irregular domains more effectively. The goal is to develop a framework that leverages the strengths of both techniques: the ability of PINNs to encode the physics of the problem into a neural network-based solution and the utility of the FDM to embed irregular domains into simpler, regular geometries.

PINNs, a relatively recent development in the realm of machine learning and numerical PDE solving, use deep neural networks to approximate solutions of differential equations. The physics of the system is enforced as part of the loss function during the neural network's training process, allowing the neural network to learn the solution while respecting the underlying governing equation and boundary conditions. This method has shown promise in solving a variety of PDEs, offering the advantage of working on unstructured data and eliminating the need for grid-based meshing altogether. Moreover, PINNs are

CONTACT José A. Rodrigues jose.rodrigues@isel.pt CIMA and Department of Mathematics of ISEL, Polytechnic University of Lisbon, Portugal.

well-suited for high-dimensional problems and can integrate data-driven elements, making them attractive for applications where the system's physics may not be entirely known or where measurement data is available.

On the other hand, the Fictitious Domain Method provides a way to simplify complex geometries by embedding the irregular domain within a larger, more regular domain (often a rectangular or circular domain). The original boundary conditions of the problem are imposed indirectly, either by introducing additional penalty terms or by using Lagrange multipliers to enforce the boundary conditions over the irregular domain's boundaries. This transformation allows the problem to be solved more efficiently on the regular domain while ensuring that the solution remains valid for the original geometry.

By combining these two approaches, the paper seeks to offer a robust and flexible method for solving elliptic problems, such as the steady-state heat equation, in complex geometries. PINNs will provide a powerful means of approximating the solution without requiring domain discretization, while the Fictitious Domain Method simplifies boundary enforcement and reduces the computational burden associated with handling irregular geometries. This hybrid approach has the potential to outperform traditional methods in terms of both accuracy and computational efficiency, especially for problems where traditional meshing is infeasible or inefficient.

The remainder of the paper is organized as follows:

Section 2 provides a comprehensive review of the theoretical background and existing numerical techniques for solving steady-state elliptic PDEs, with a particular emphasis on challenges posed by irregular domains. Additionally, we explore recent advancements in PINNs, highlighting their growing utility in solving PDEs through data-driven, mesh-free approaches. The section also presents a historical overview of the FDM and its evolution as a tool for managing complex geometrical constraints. Finally, we discuss emerging hybrid strategies that integrate neural networks with classical numerical schemes to enhance flexibility and accuracy.

Section 3 conducts a critical comparison between the proposed PINN-FDM approach and other state-of-the-art numerical methods, including isogeometric analysis (IGA), meshless methods, and multi-domain PINNs. The evaluation focuses on their respective strengths, limitations, and implementation complexities, particularly when applied to PDEs in non-standard geometries. Special attention is given to the ability of each method to handle boundary conditions and irregular domains efficiently, providing context for the advantages of the integrated PINN-FDM framework.

Section 4 introduces the mathematical formulation of the steady-state heat equation, commonly used to model elliptic problems. We discuss the unique challenges associated with solving such equations in irregular domains and present the theoretical foundation for combining PINNs with the Fictitious Domain Method. The derivation of governing equations and boundary conditions is provided, along with a detailed explanation of how PINNs can be adapted to incorporate the FDM. The synergy between the two approaches forms a cohesive framework for addressing elliptic PDEs in complex geometrical settings.

Section 5 describes the computational structure of the proposed PINN-FDM method. The process begins by embedding the irregular domain within a regular, fictitious domain to simplify the application of boundary conditions. We then detail the construction and training of the PINN, which is designed to approximate conservative variables by embedding physical laws into the loss function. The methodology section also outlines the architecture of the neural network, the optimization procedure, and the mechanisms used to enforce PDE residuals and boundary constraints throughout the training process.

Section 6 presents a series of numerical experiments demonstrating the effectiveness of the PINN-FDM framework across various problem settings. We examine domains with complex shapes and evaluate the framework under different boundary conditions, including Dirichlet and Neumann types. The performance of the model is assessed in terms of solution accuracy, convergence behavior, and computational efficiency. Additionally, we analyze the sensitivity of the results to changes in neural network architecture and training parameters.

Section 7 introduces methods for estimating and analyzing errors in the neural network solution. We discuss techniques for quantifying the residuals of the PDE, evaluating boundary condition satisfaction, and computing relative and absolute errors across the domain. Strategies for improving model accuracy through adaptive refinement and loss function rebalancing are also explored. This analysis provides deeper insights into the reliability and precision of the PINN-FDM approach.

Finally, Section 8 summarizes the main contributions of this work, emphasizing the effectiveness of the PINN-FDM framework in solving steady-state elliptic PDEs in irregular domains. We highlight the advantages of the method in terms of flexibility, accuracy, and ease of implementation. Limitations related to training stability and computational cost are acknowledged, and future directions are proposed, including extensions to time-dependent and nonlinear problems, integration with adaptive mesh strategies, and enhancements in neural network architecture and training techniques.

2. Steady-State Elliptic PDEs and Numerical Methods

Steady-state elliptic partial differential equations are a fundamental class of problems in mathematical physics and engineering, often modelling processes such as heat conduction, electrostatics, and fluid flow. Common numerical methods for solving these PDEs include the FEM and the FDM. Both of these methods have been extensively studied and applied to a wide variety of problems due to their versatility and computational efficiency.

The FEM is widely regarded as one of the most powerful tools for solving elliptic PDEs. FEM works by dividing a domain into smaller subdomains or "elements", usually triangles or quadrilaterals in two dimensions, over which approximate solutions to the governing equations are constructed using basis functions. The FEM is particularly advantageous because of its ability to handle complex geometries, as well as its adaptability to irregular meshes. However, FEM can become computationally expensive when dealing with highly irregular domains, as it requires mesh generation that accurately conforms to complex boundaries. For

irregular geometries, mesh refinement near boundaries often leads to high computational costs and potential errors due to mesh distortion.

The FDM takes a different approach by embedding complex geometries within a larger, regular computational domain. This allows the governing PDEs to be discretized over a structured grid, regardless of the irregularity of the actual physical boundaries. FDM is computationally efficient and simplifies mesh generation, as it avoids the need to conform the mesh to complex geometries. However, accurately enforcing boundary conditions within the fictitious domain requires specialized techniques, such as penalty methods, Lagrange multipliers, or immersed boundary formulations. These additions can increase implementation complexity and may affect the accuracy or stability of the method.

In general, both FEM and FDM face challenges when dealing with irregular geometries. While FEM requires body-fitted meshes, which complicate mesh generation and adaptation, FDM must manage accurate boundary condition enforcement within a non-body-fitted grid. In both cases, achieving high resolution and minimizing discretization errors near complex boundaries often necessitates adaptive strategies or advanced numerical treatments. This motivates the ongoing search for alternative methods that combine geometric flexibility with computational efficiency.

2.1. Physics-Informed Neural Networks

Physics-Informed Neural Networks are a relatively new class of machine learning algorithms designed to solve differential equations by embedding the governing physical laws into the structure of the neural network. First introduced by Raissi, Perdikaris, and Karniadakis in 2019 Raissi, Perdikaris, and Karniadakis (2019), PINNs have quickly gained attention for their ability to tackle a wide range of problems governed by PDEs, including steady-state elliptic problems like the heat equation.

Unlike traditional numerical methods that require discretization of the spatial domain, PINNs approximate the solution to PDEs by training a neural network to minimize a loss function that represents the residuals of the governing PDE and the boundary conditions. The solution is represented by a neural network with parameters that are adjusted through optimization to minimize the error in satisfying the PDE and the boundary conditions over a set of training points. One key advantage of PINNs is their ability to directly handle irregular domains without the need for meshing or grid generation, as the neural network learns the solution across the entire domain continuously.

Additionally, PINNs can integrate data-driven components, making them highly suitable for problems where physical laws are partially known or noisy experimental data is available. For example, PINNs can be trained using partial datasets or observations of the system while enforcing the underlying physics as constraints during training. This data-driven capability makes PINNs flexible and valuable in real-world applications where precise models may be incomplete or difficult to obtain.

Moreover, PINNs can efficiently handle complex boundary conditions, which are often challenging for traditional methods. By including boundary conditions as part of the loss function, PINNs ensure that these constraints are satisfied without the need for special boundary discretization or meshing techniques. This flexibility in handling boundary conditions and irregular domains makes PINNs an attractive alternative to classical numerical methods, especially in cases where traditional methods struggle due to geometric or boundary complexities Karniadakis et al. (2021).

Recent research has demonstrated their versatility across a broad range of applications, particularly in scenarios where traditional numerical methods face challenges due to complex boundary conditions, discontinuities, or irregular geometries. For instance, Wu et al. (2023) extended the PINN framework to fractional calculus by solving Hausdorff derivative Poisson equations, illustrating the method's adaptability to nonlocal and anomalous diffusion problems. Moreover, multi-domain PINNs have gained attention for their ability to handle problems involving heterogeneous or layered materials. Zhang et al. (2022) proposed a multi-domain PINN approach for steady-state heat conduction in multilayer media, while Zhang, et al. (2023) further extended this framework to transient heat conduction problems, demonstrating improved accuracy and computational efficiency in capturing interfacial continuity and temporal dynamics. These advancements underscore the growing interest in enhancing PINN architectures to better accommodate domain decomposition, material heterogeneity, and geometric complexity-motivating the current study's focus on comparing PINNs with other numerical and hybrid techniques for solving PDEs in complex geometries.

2.2. Fictitious Domain Methods

The Fictitious Domain Method is a powerful approach designed to simplify the solution of partial differential equations in irregular domains by transforming the problem into a more convenient setting Glowinski et al. (2001). This method embeds the irregular domain into a larger, simpler regular domain, often rectangular or circular, and extends the governing PDE to this extended domain. Boundary conditions for the original irregular domain are imposed indirectly, either through penalty methods Babuška (1973) or using Lagrange multipliers Glowinski, Lions, and Tremolieres (1984). This enables the problem to be solved more easily in the regular domain while ensuring that the boundary conditions of the original domain are satisfied.

One of the primary advantages of FDM is its ability to transform complex, irregular geometries into regular shapes that are easier to handle using standard numerical techniques such as finite difference methods or finite element methods Lions and Mercier (1990). The key idea is that by embedding the irregular domain into a regular one, the computational burden associated with meshing complex geometries is significantly reduced. Instead of constructing a conforming mesh that fits the boundary of the irregular domain, a simple grid can be used over the larger regular domain. The complexity of the geometry is managed by imposing the boundary conditions appropriately, often through the introduction of auxiliary terms that enforce the solution to satisfy the boundary conditions on the original irregular domain Ramière, Angot, and Belliard (2007).

FDM has proven particularly useful in problems involving complex geometries, such as fluid flow around obstacles Peskin (2002) or heat transfer in irregularly shaped objects Burman and Hansbo (2012). Its ability to simplify domain discretization while maintaining accuracy has made it a valuable tool in computational mechanics. However, the indirect enforcement of boundary conditions can sometimes introduce additional computational overhead, as auxiliary constraints must be solved alongside the primary PDE Glowinski, He, and Le Tallec (2003). Despite this limitation, FDM remains a popular choice for solving PDEs in complex domains due to its flexibility and robustness.

2.3. Hybrid Approaches

In recent years, there has been increasing interest in hybrid approaches that combine the strengths of traditional numerical methods with machine learning techniques, particularly PINNs. These hybrid methods aim to address the shortcomings of classical methods in handling irregular domains, while leveraging the data-driven capabilities and flexibility of neural networks.

One promising approach is to use PINNs in conjunction with the Fictitious Domain Method to solve elliptic problems in irregular geometries. In such a framework, the FDM simplifies the domain by embedding it in a regular shape, while PINNs are employed to approximate the solution across the entire extended domain. This approach combines the geometric simplifications offered by FDM with the flexibility and continuous approximation power of neural networks. By training the PINN on the extended domain and enforcing boundary conditions via the FDM, the hybrid method can efficiently solve problems in complex geometries without the need for intricate meshing techniques.

Another hybrid approach involves combining FEM or FDM with PINNs, where traditional numerical methods are used to discretize the domain, and a PINN is applied to handle difficult boundary conditions or regions where classical methods perform poorly. In such cases, the neural network can learn corrections or adjustments to the solution in regions with high errors or complex boundaries. This combination of traditional numerical techniques and machine learning provides a powerful tool for solving PDEs in challenging settings, blending the best aspects of both worlds.

The integration of neural networks into classical numerical frameworks opens up new possibilities for solving high-dimensional and multi-physics problems, where traditional methods face significant computational and implementation challenges. These hybrid methods represent a promising direction for future research and offer new strategies for efficiently solving steady-state elliptic PDEs in irregular domains Zhang (2020).

3. Comparison of PINN-FDM with Other Numerical Methods

Traditional numerical methods such as the FEM and Finite Volume Method (FVM) have been the cornerstone of solving PDEs in complex geometries. FEM is well-regarded for its flexibility in handling arbitrary geometries and mesh refinement strategies, while FVM is preferred in computational fluid dynamics due to its conservative properties. However, both require mesh generation, which becomes computationally intensive for high-dimensional or evolving domains.

As above mentioned PINNs have emerged as a powerful tool for solving problems with partial differential equations, particularly in complex geometries where traditional numerical methods face significant challenges. Among these methods, isogeometric analysis, meshless methods, and multi-domain PINNs offer alternative approaches to tackle PDEs. This section presents several comparisons between PINNs and traditional or hybrid methods focusing on their respective strengths, limitations, and implementation challenges in dealing with complex geometries.

3.1. Isogeometric Analysis

Isogeometric analysis integrates computational geometry and numerical simulation by using the same basis functions for both geometry representation and solution approximation, typically Non-Uniform Rational B-Splines (NURBS). This approach provides higher-order continuity and accurate geometric modeling, which is advantageous for problems requiring smooth solutions or involving complex geometries Hughes (2005). However, implementing IGA poses several challenges. The construction of NURBS-based meshes requires precise control over knot vectors and degree elevation, which can be cumbersome for irregular domains Cottrell (2009) and Rodrigues (2020). Additionally, while IGA excels in structural mechanics problems, it may struggle with high-dimensional or highly nonlinear PDEs due to the computational cost associated with assembling and solving large systems of equations Bazilevs et al. (2007).

In contrast, PINNs with fictitious domain formulations do not require explicit mesh generation. Instead, they rely on collocation points that can be randomly sampled within the domain, including its boundaries. This eliminates the need for intricate meshing procedures and allows for straightforward handling of complex geometries Raissi, Perdikaris, and Karniadakis (2019). Furthermore, PINNs naturally handle high-dimensional problems, making them suitable for applications such as uncertainty quantification and inverse problems, where IGA might become computationally prohibitive Karniadakis et al. (2021).

3.2. Meshless Methods

Meshless methods, such as the Element-Free Galerkin method and Smoothed Particle Hydrodynamics (SPH), avoid the use of a predefined mesh by employing shape functions defined over scattered nodes. These methods are particularly effective for problems involving large deformations, moving boundaries, and crack propagation Liu (2003). Nevertheless, meshless methods have their own set of limitations. One major issue is the ill-conditioning of the system matrices, which can arise from the lack of structured nodal arrangements Belytschko, Lu, and Gu (1996). Moreover, enforcing essential boundary conditions in meshless methods often requires additional techniques, such as Lagrange multipliers or penalty methods, which can complicate the implementation process Atluri and Zhu (1998).

PINNs, on the other hand, enforce boundary conditions implicitly through the loss function, avoiding the need for specialized techniques. The fictitious domain approach further simplifies this process by extending the computational domain beyond the physical boundaries, allowing for seamless imposition of boundary constraints Jagtap, Kawaguchi, and Karniadakis (2020). While meshless methods may offer advantages in specific applications, their sensitivity to node distribution and computational overhead for large-scale problems make them less versatile compared to PINNs Li and Liu (2002).

3.3. Deep Galerkin Methods

Deep Galerkin Methods (DGM), introduced by Sirignano and Spiliopoulos (2018), represent a mesh-free, neural network-based approach for solving high-dimensional PDEs. Unlike classical numerical schemes that rely on discretizing space and time domains, DGM employs deep feedforward neural networks to approximate the solution of the PDE directly. The network is trained to minimize the residual of the PDE, as well as its boundary and initial conditions, by sampling points in the domain-typically using random Monte Carlo techniques. A distinguishing feature of DGM is its connection to the Galerkin method. Instead of discretizing the PDE on a mesh, DGM projects the PDE residual onto a function space spanned by the neural network architecture. This projection effectively minimizes the PDE error in an average sense across the domain, making the method particularly attractive for high-dimensional problems like those in quantitative finance and statistical physics. **Strengths:** DGM exhibits excellent scalability to high dimensions. It is well-suited for solving high-dimensional PDEs where traditional mesh-based methods (such as FEM or FDM) become infeasible due to the curse of dimensionality. Additionally, the method is inherently mesh-free, requiring no structured grid, which simplifies its application to irregular domains. **Limitations:** Despite its strengths, DGM can struggle to capture fine-grained or sharp boundary features, particularly in complex geometries. The training process can be unstable, as the method is sensitive to neural network architecture, optimization strategies, and point sampling. Furthermore, the learned solution is embedded in the neural network weights, making interpretability and generalization to different domains challenging.

3.4. Convolutional and Graph Neural Networks

Another emerging approach to PDE solving leverages convolutional neural networks (CNNs) and graph neural networks (GNNs) for surrogate modeling and approximate solvers. Zhu et al. (2019) proposed a physics-constrained deep learning framework that integrates data-driven learning with known physical laws. This framework enables models to make predictions without labeled data (i.e., without known PDE solutions) by enforcing the governing physics directly in the loss function. CNNs are typically applied to grid-based domains, where spatial correlations can be effectively captured through convolutional filters. They are particularly effective at learning local patterns and have seen extensive use in approximating PDE solutions in domains like fluid dynamics and materials science. However, their reliance on structured grid data limits their application in arbitrarily shaped or irregular geometries. GNNs extend neural networks to graph-structured data and are better suited for unstructured meshes or point cloud representations of physical domains. This makes GNNs particularly powerful for modeling PDEs over irregular surfaces, triangulated domains, and FEM meshes, where connectivity does not follow a regular pattern. GNNs can effectively propagate information across these irregular domains, offering enhanced flexibility.

Once trained, CNN and GNN models provide extremely fast inference, making them ideal for real-time simulation and control. GNNs are notably flexible, as they can handle complex and irregular geometries without requiring explicit meshing. Furthermore, these architectures offer potential for transfer learning across similar PDE tasks.

A key limitation of CNN and GNN models is their dependence on large, labeled datasets or strong physical priors for training. Without embedding physics-based constraints, their predictions may lack physical consistency and robustness. Purely data-driven models can also struggle to generalize to unseen boundary or initial conditions. Additionally, the performance of these models is highly sensitive to architecture design, including depth, connectivity, and resolution.

3.5. Multi-Domain Physics-Informed Neural Networks

Multi-domain PINNs extend the standard PINN framework by dividing the computational domain into subdomains, each solved independently but coupled through interface conditions Jagtap, Kawaguchi, and Karniadakis (2020). This approach offers several benefits, including improved scalability for large-scale problems and better handling of discontinuities or sharp gradients within the solution. By decomposing the problem into smaller subproblems, multi-domain PINNs can reduce the overall training time and memory requirements, making them suitable for parallel computing environments Zhang, Guo, and Karniadakis (2021).

However, the fictitious domain approach within single-domain PINNs provides a simpler and more unified framework for solving PDEs. It avoids the need for explicit domain decomposition and interface coupling, reducing the complexity of the implementation. Moreover, the fictitious domain method inherently accounts for irregular boundaries and non-smooth solutions without requiring additional modifications Khoo, Lu, and Ying (2021). While multi-domain PINNs may outperform single-domain PINNs in certain scenarios, the added complexity of domain decomposition must be carefully weighed against the potential gains in performance.

The fictitious domain approach within PINNs offers a flexible and robust framework for solving PDEs, particularly in complex geometries. Compared to isogeometric analysis, it eliminates the need for intricate meshing procedures and scales well to high-dimensional problems. Unlike meshless methods, PINNs enforce boundary conditions naturally and avoid issues related to ill-conditioned system matrices. Furthermore, while multi-domain PINNs provide advantages in scalability and handling discontinuities, the fictitious domain method maintains simplicity and generality, making it an attractive choice for a wide range of applications. As research continues to advance, PINNs and their variants are likely to play an increasingly important role in computational science and engineering.

Table 1. Comparison of numerical methods in terms of Geometry Handling, Accuracy, Computational Cost, Mesh Requirement, and Implementation Ease

Method	Geometry Handling	Accuracy	Computational Cost	Mesh Requirement	Implementation Ease
FEM	Good	High	Medium-High	Yes	Medium
FVM	Moderate	High (Conservation)	Medium	Yes	Medium
IGA	Excellent	Very High	High	Yes (CAD-based)	Complex
Meshless	Excellent	Medium-High	Medium-High	No	Moderate
PINN	Good	Medium-High	High	No	High (Training time)
PINN-FDM (Proposed)	**Excellent**	**High**	**Lower (than PINN alone)**	**No**	**Moderate**
DGM	Good	Medium	Medium	No	High
CNN/GNN	Moderate	Medium (Data-dependent)	Low	No	Moderate

3.6. Comparative Analysis of PINN-FDM Against Established Numerical and Learning-Based Techniques

When comparing PINN-FDM to traditional and machine learning-based methods, several key factors emerge:

Accuracy

PINN-FDM achieves high accuracy by combining data-driven learning with physical constraints and leveraging the stable discretization of FDM. Compared to FEM and FVM, it offers comparable accuracy without requiring complex meshing. It also outperforms standard PINNs in geometries where grid-based discretization improves gradient estimation.

Computational Efficiency

FEM and FVM often involve high computational overhead in mesh generation and sparse matrix solvers. PINN-FDM eliminates mesh generation and leverages GPU-accelerated training, making it scalable for moderate problem sizes. Unlike CNN-based methods, PINN-FDM does not require precomputed datasets or fine-resolution data for training. DGMs and GNNs, while mesh-free, often require longer training times and hyperparameter tuning.

Ease of Implementation

FEM and FVM require domain-specific expertise and software (e.g., COMSOL, OpenFOAM), whereas PINN-FDM can be implemented in general-purpose ML frameworks (TensorFlow, PyTorch). Compared to CNNs and GNNs, PINN-FDM has fewer architectural constraints and leverages physics directly, reducing the need for labeled data.

To provide a comprehensive comparison between the proposed PINN-FDM approach and existing numerical and machine learning-based methods for solving PDEs, Table 1 summarizes key characteristics of various techniques. These include traditional methods such as the FEM and FVM, hybrid strategies like IGA, and advanced learning-based models including PINNs, DGM, CNNs, and GNNs. The criteria for comparison include geometry handling capability, solution accuracy, computational cost, mesh requirements, and ease of implementation. This tabular overview highlights the balance achieved by the PINN-FDM method in terms of computational efficiency and flexibility, especially when dealing with complex or irregular geometries.

4. PINN-FDM for Solving Elliptic Problems

The first step in the hybrid PINN-FDM algorithm is to embed the irregular domain into a larger, regular domain using the Fictitious Domain Method. In traditional numerical approaches, the irregular geometry would require complex mesh generation tailored to the boundary, which could lead to computational inefficiency. By contrast, FDM extends the problem into a simpler geometry, typically a rectangle or a circle, where solving the PDE becomes more straightforward. The boundary of the irregular domain is implicitly enforced through auxiliary terms that apply the original boundary conditions, while the regular domain facilitates the use of efficient numerical techniques and regular grid structures.

The FDM allows the original elliptic PDE to be posed in this extended regular domain, effectively converting a difficult problem in an irregular domain into one that is computationally easier to handle. The complex boundary conditions imposed by the irregular domain are handled indirectly using penalty methods or Lagrange multipliers, which ensures that the solution respects the original domain's boundary.

4.1. Loss Function Definition

The loss function is the cornerstone of the PINN-FDM algorithm, as it drives the training of the neural network and enforces the constraints of the problem. In this hybrid approach, he total loss function consists of three main components:

1 **1** PDE Residual Loss: The PDE residual refers to how well the predicted solution from the neural network satisfies the governing elliptic PDE within the entire extended domain. Let $\mathcal{D}_{\text{reg}}$ represent the extended regular domain, and $\mathcal{D}_{\text{irr}}$ represent the original irregular domain. The residual of the elliptic PDE (e.g., the heat equation $\Delta u = 0$) is given by the difference between the left-hand side and the right-hand side of the equation at each point in the domain. The PDE residual is computed at a set of collocation points, which are points sampled throughout the extended domain:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_{\text{coll}}} \sum_{i=1}^{N_{\text{coll}}} \left(\Delta u(x_i) - f(x_i)\right)^2, \tag{2}$$

where N_{coll} is the number of collocation points in the domain, $u(x_i)$ is the network-predicted solution at point x_i, and $f(x_i)$ represents any source terms or forcing functions within the PDE.

2 **2** Boundary Condition Loss: Enforcing boundary conditions from the irregular domain into the extended domain is essential. The FDM allows us to incorporate these boundary conditions effectively. Let $\partial\mathcal{D}_{\text{irr}}$ represent the boundary of the original irregular domain. Different types of boundary conditions contribute to the boundary loss:

Dirichlet Boundary Condition: When the solution u is prescribed on the boundary:

$$\mathcal{L}_{\mathrm{D}} = \frac{1}{N_{\mathrm{bc}}} \sum_{j=1}^{N_{\mathrm{bc}}} \left(u(x_j) - g(x_j)\right)^2. \tag{3}$$

Neumann Boundary Condition: When the normal derivative of the solution is specified on the boundary:

$$\mathcal{L}_{\mathrm{N}} = \frac{1}{N_{\mathrm{bc}}} \sum_{j=1}^{N_{\mathrm{bc}}} \left(\frac{\partial u}{\partial n}(x_j) - h(x_j)\right)^2. \tag{4}$$

Robin Boundary Condition: A combination of Dirichlet and Neumann conditions:

$$\mathcal{L}_{\mathrm{R}} = \frac{1}{N_{\mathrm{bc}}} \sum_{j=1}^{N_{\mathrm{bc}}} \left(\alpha u(x_j) + \beta \frac{\partial u}{\partial n}(x_j) - r(x_j)\right)^2. \tag{5}$$

The total boundary loss is given by:

$$\mathcal{L}_{\mathrm{BC}} = \lambda_D \mathcal{L}_{\mathrm{D}} + \lambda_N \mathcal{L}_{\mathrm{N}} + \lambda_R \mathcal{L}_{\mathrm{R}}, \tag{6}$$

where λ_D, λ_N, and λ_R are weighting factors for Dirichlet, Neumann, and Robin boundary conditions, respectively.

3 **3** Data fidelity: The data fidelity term quantifies how well the neural network's predicted solution aligns with available observations or target data within the irregular domain $\mathcal{D}_{\mathrm{irr}}$. This component is essential when supervised data (e.g., sensor measurements or high-fidelity simulation outputs) are available, and ensures that the model does not merely satisfy the governing equations, but also adheres to real-world or expected outcomes.

For $\{(x_j, u_j^{\mathrm{data}})\}_{j=1}^{N_{\mathrm{data}}}$ a set of observation points and corresponding data values. The data fidelity loss is defined as:

$$\mathcal{L}_{\mathrm{data}} = \frac{1}{N_{\mathrm{data}}} \sum_{j=1}^{N_{\mathrm{data}}} \left(u(x_j) - u_j^{\mathrm{data}}\right)^2, \tag{7}$$

where $u(x_j)$ is the predicted solution at point x_j. This term encourages the neural network to produce solutions that not only satisfy the PDE but are also consistent with empirical data, improving the model's accuracy and reliability.

The total loss function is then expressed as:

$$\mathcal{L} = \mathcal{L}_{\mathrm{PDE}} + \mathcal{L}_{\mathrm{BC}} + \mathcal{L}_{\mathrm{data}}, \tag{8}$$

In PINNs, the primary objective is to learn solutions that satisfy the underlying governing physical laws, typically formulated as PDEs, without the need for supervised data. Consequently, a data fidelity term is not included in the loss function. Instead, PINNs enforce the physical constraints by minimizing the PDE residual at a set of collocation points. This physics-driven approach allows PINNs to approximate solutions even in the absence of labeled data, relying solely on the structure of the PDE and associated boundary and initial conditions. For this reason, the loss function in PINNs typically takes the form:

$$\mathcal{L} = \mathcal{L}_{\mathrm{PDE}} + \mathcal{L}_{\mathrm{BC}}, \tag{9}$$

where the boundary loss ensures adherence to the imposed constraints.

For time-dependent PDEs, such as the heat or wave equation, the solution must satisfy known conditions at the initial time $t = 0$. In the PINN framework, these initial conditions are enforced by introducing a loss term that penalizes discrepancies between the predicted solution and the prescribed initial values.

Let $\{(x_k, u_k^{\mathrm{init}})\}_{k=1}^{N_{\mathrm{init}}}$ be a set of initial data points, where u_k^{init} denotes the known value of the solution at spatial location x_k and initial time $t = 0$. The initial condition loss is defined as:

$$\mathcal{L}_{\mathrm{init}} = \frac{1}{N_{\mathrm{init}}} \sum_{k=1}^{N_{\mathrm{init}}} \left(u(x_k, 0) - u_k^{\mathrm{init}}\right)^2, \tag{10}$$

where $u(x_k, 0)$ is the neural network's prediction at the initial time. This term ensures that the learned solution aligns with the known physical state of the system at $t = 0$, which is critical for the well-posedness of the PDE and for accurately capturing the system's temporal evolution.

In cases where the PDE also specifies the initial time derivative (as in second-order systems like the wave equation), an additional initial condition loss can be defined for $\partial_t u(x, 0)$ in a similar fashion.

For the evolutive problems, this initial condition loss $\mathcal{L}_{\mathrm{init}}$ is incorporated into the total loss function, which typically includes the PDE residual loss and boundary condition loss:

$$\mathcal{L}_{\mathrm{total}} = \mathcal{L}_{\mathrm{PDE}} + \mathcal{L}_{\mathrm{BC}} + \mathcal{L}_{\mathrm{init}}. \tag{11}$$

This ensures that the neural network solution satisfies not only the governing PDE and spatial boundary conditions, but also the initial condition, resulting in a physically consistent model over both space and time.

4.2. Neural Network Architecture

To approximate the solution of the elliptic PDE, we employ a fully connected feed-forward neural network. The architecture comprises an input layer, several hidden layers, and an output layer. The input layer accepts the spatial coordinates x and y corresponding to points in the domain, while the output layer provides the predicted solution $u(x, y)$.

The hidden layers utilize nonlinear activation functions, such as the hyperbolic tangent (tanh) or Rectified Linear Unit (ReLU), to introduce the nonlinearity required for approximating complex functions. The choice of activation functions in a PINN plays a crucial role in determining the network's ability to approximate solutions to partial differential equations effectively. In this study, the hyperbolic tangent activation function was selected due to its smoothness, nonlinearity, and ability to capture complex function mappings required for solving the heat equation. The tanh function is particularly advantageous because it provides a range of outputs between -1 and 1, which helps maintain stable gradient flow during backpropagation, reducing the risk of vanishing gradients compared to the sigmoid function. Furthermore, its second derivative remains continuous, making it well-suited for PDEs requiring higher-order derivatives, as in the Poisson equation with Neumann boundary conditions.

Despite these advantages, the tanh activation function is not without limitations. One primary concern is its susceptibility to saturation, where large inputs cause gradients to diminish, slowing down training. This phenomenon is particularly significant in deep networks, where multiple layers can accumulate small gradients, leading to training inefficiencies. Additionally, while the tanh function ensures smooth transitions in solution approximations, it might struggle to capture sharp discontinuities or steep gradients in the solution space, which can arise in problems with complex boundary conditions or highly localized heat sources.

Given these limitations, alternative activation functions can be explored to enhance the network's performance. Rectified Linear Unit and its variants (Leaky ReLU, Parametric ReLU) are popular choices in deep learning due to their non-saturating properties and computational efficiency. However, standard ReLU lacks smoothness, which can lead to issues when computing second-order derivatives, making it less suitable for PDEs involving Laplacians. Leaky ReLU, on the other hand, mitigates the issue of zero gradients in the negative domain, though it still lacks the smoothness required for accurate differential approximations. Gaussian activation functions and sinusoidal functions (such as sine or Softplus) offer smooth alternatives that preserve differentiability while avoiding saturation. Sinusoidal activations, in particular, have been successfully employed in Fourier Neural Operators and SIREN (Sinusoidal Representation Networks), where they enable improved approximation of high-frequency components.

A potential hybrid approach involves using adaptive activation functions, where the choice of activation varies depending on the training stage or location within the domain. For example, using tanh in early training stages for its stability and then transitioning to a more oscillatory function like sine could improve accuracy in capturing fine-scale variations. Similarly, learnable activation functions, such as Swish or adaptive parametric activations, could dynamically adjust to the nature of the solution, offering a balance between smoothness and flexibility.

The architecture is characterized by the number of hidden layers and the number of neurons in each layer, which can be tuned as hyperparameters. Typically, architectures with 3 to 6 hidden layers are used, where each layer contains 50 to 100 neurons, depending on the complexity of the problem.

The choice of the optimizer also plays a crucial role in training the neural network. Common optimizers include Adam, SGD (Stochastic Gradient Descent), and RMSprop, with Adam being a popular choice due to its adaptive learning rate and efficiency in converging to a solution.

4.3. NURBS Boundary Definition

A NURBS curve of degree d is defined by the parametric equation:

$$C(u) = \sum_{i=0}^{n} R_{i,d}(u) P_i, \quad u \in [0,1], \tag{12}$$

where P_i are the control points and $R_{i,d}(u)$ are the rational basis functions given by:

$$R_{i,d}(u) = \frac{N_{i,d}(u) w_i}{\sum_{j=0}^{n} N_{j,d}(u) w_j}. \tag{13}$$

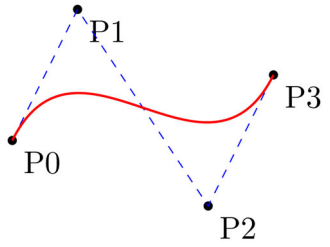


Figure 1. Illustration of a NURBS-like curve with control points and control polygon. Black dots represent control points P_0 to P_3, the dashed blue lines form the control polygon, and the red curve approximates the resulting NURBS geometry.

Here, $N_{i,d}(u)$ are the B-spline basis functions and w_i are the corresponding weights associated with each control point.

Figure 1 illustrates the geometric construction of a NURBS-like curve, which plays a central role in the boundary definition of complex domains within this study. The diagram shows a set of control points labeled P_0 to P_3, represented by black dots, which govern the shape of the curve. These points are connected by dashed blue lines to form the control polygon, providing a visual guide to how the curve interpolates and is influenced by the surrounding geometry. The resulting smooth red curve, drawn using a Bézier approximation for illustrative purposes, demonstrates the essential characteristics of a NURBS representation-namely, its ability to produce smooth and flexible curves that conform to the desired geometry while maintaining mathematical continuity and compactness. This visual aid helps convey how NURBS are employed in the PINN framework to model domain boundaries with high fidelity and minimal meshing effort.

4.4. NURBS in connection to PINN Domain Definition

The use of NURBS is crucial in this study for defining the boundary of a geometrically complex domain, specifically one with a flower-like shape, which poses significant challenges for traditional structured meshing techniques. In our implementation, a NURBS curve is constructed using cubic B-splines together with a closed set of control points to accurately describe the boundary. This parametric representation allows for precise control over the domain's geometry while preserving smoothness and continuity, which is particularly beneficial for high-fidelity modeling.

Within the PINN framework, the points sampled along this NURBS-defined boundary serve two essential purposes. First, they are used for enforcing Dirichlet boundary conditions by incorporating penalty terms in the loss function that quantify deviations of the network's output from the prescribed boundary values. This method allows the network to satisfy boundary constraints naturally without needing an explicitly meshed boundary. Second, the NURBS boundary aids in characterizing the interior of the domain. To identify which training points fall inside the physical domain, a geometric point-in-polygon algorithm-such as the ray casting method-is applied, effectively distinguishing interior points from those outside the domain. This approach enables flexible and efficient sampling of training data, even in domains with intricate contours.

By integrating NURBS into the PINN framework, we achieve a smooth and adaptive representation of complex geometries

while maintaining compatibility with mesh-free training strategies. This integration eliminates the need for complex remeshing or coordinate transformations and establishes a seamless connection between precise geometric modeling and physics-informed neural computation. Such a capability is especially advantageous in engineering and computational mechanics applications, where domains often exhibit irregular and curved boundaries that are not easily captured by traditional meshing schemes.

5. Computational Framework and Methodology

5.1. Computational Framework

The PINN approach was developed using the `TensorFlow 2.x` framework, which enables the construction of custom deep learning models with efficient support for automatic differentiation—a critical requirement for computing PDE residuals and gradients with respect to input spatial coordinates. TensorFlow's `tf.GradientTape` API was used to compute both first- and second-order derivatives, essential for enforcing the physics of the underlying partial differential equation.

All experiments were executed on a local workstation equipped with an NVIDIA RTX-series GPU, which offered substantial acceleration over CPU-based training by parallelizing matrix operations and backpropagation steps. This GPU-based setup was especially beneficial due to the high computational cost of evaluating PDE loss terms involving second-order derivatives at a large number of spatial points. Nevertheless, the framework remains compatible with CPU execution, although at significantly slower runtimes.

The PINN architecture consisted of a fully connected feed-forward neural network with three hidden layers, each comprising 50 neurons. The activation function used across all hidden layers was the hyperbolic tangent, which is commonly favored in PINNs due to its smoothness, nonlinearity, and bounded nature. These characteristics help ensure that both the function and its derivatives are well-behaved, making `tanh` particularly suitable for learning the continuous and differentiable solutions expected from PDEs.

5.2. Optimizer Justification

Optimization of the network weights was carried out using the Adam optimizer. Adam is a first-order gradient-based optimization algorithm that computes adaptive learning rates for each parameter. It combines the advantages of two widely used optimizers: AdaGrad and RMSProp. Specifically, it maintains an exponentially decaying average of past gradients (momentum) and past squared gradients (adaptive learning rate scaling), making it highly suitable for noisy gradients and sparse data.

Adam was chosen for this work primarily due to its efficiency, ease of tuning, and its ability to handle the complex loss landscape that arises in PINNs. In this problem, the domain features a nontrivial geometry defined by a NURBS boundary, introducing geometric complexity that influences the loss surface. Additionally, the PDE includes a localized heat source, which creates steep gradients in the solution field. Adam's adaptivity helped maintain stable and consistent convergence under these challenges.

Although second-order optimizers like L-BFGS have shown superior convergence properties in some PINN applications—especially where higher precision is required or when training data is limited—they were not used here. TensorFlow lacks robust native support for L-BFGS, and custom implementations typically incur higher memory overhead and batch-size constraints. Furthermore, for large-scale or long-duration training, first-order methods like Adam are generally more scalable.

5.3. Hyperparameter Tuning

The selection of hyperparameters was guided by a combination of prior literature, heuristic rules, and empirical tuning. A fixed learning rate of 0.001 was adopted after preliminary experiments, as it provided a good trade-off between training speed and convergence stability. The choice of using three hidden layers with 50 neurons each was based on the observed balance between model complexity and training time. Larger architectures led to diminishing returns in accuracy, while significantly smaller models underfit the data.

Hyperparameter tuning was conducted manually using a grid-search-like approach. Configurations were evaluated based on the total training loss, convergence rate, and visual inspection of the final temperature field predictions. Loss curves were monitored throughout training, and configurations leading to divergence or stagnation were discarded early. While effective in this case, such manual tuning is time-consuming and may not generalize across domains. Future extensions may incorporate more systematic approaches, such as Bayesian optimization or automated machine learning (AutoML) techniques.

The dataset for training the PINN consisted of three point sets:

Interior points: 1000 randomly sampled points from a uniform distribution over the unit square domain. These were used to enforce the PDE residual condition.
Boundary points: 200 points sampled from a NURBS curve representing a flower-like domain boundary. These points imposed Dirichlet boundary conditions.
Fictitious domain points: 200 additional points sampled outside the boundary, used to enforce solution decay and avoid unphysical behavior in the extended domain.

These data quantities were empirically chosen to balance computational efficiency and solution fidelity. More refined sampling strategies, such as adaptive point placement or importance sampling based on loss gradients, are left for future work.

5.4. Physics-Informed Neural Network

The neural network $\mathcal{N}_\theta$ is used to approximate the temperature field:

$$T_\theta(x, y) \approx T(x, y), \tag{14}$$

where $T(x, y)$ represents the true temperature distribution and $T_\theta(x, y)$ is the approximation parameterized by the weights and biases θ.

The forward propagation through the network can be described mathematically as follows:

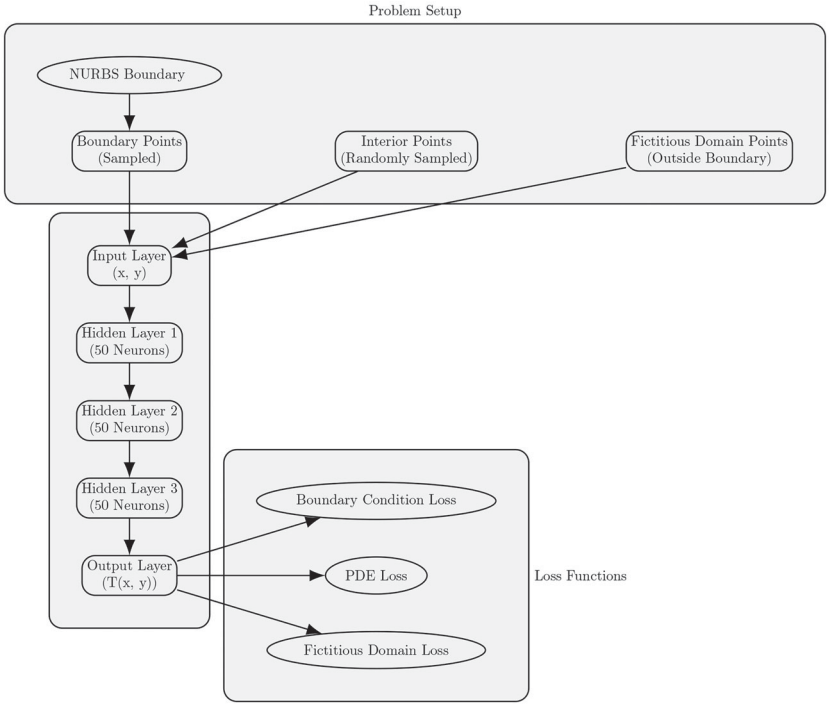


Figure 2. Workflow of the Physics-Informed Neural Network (PINN) for Solving the Heat Equation in a Complex Domain.

Let $z^{(l)}$ represent the activations at layer l, where $l = 0, 1, 2, 3$ corresponds to the input layer, first hidden layer, second hidden layer, and third hidden layer, respectively. The output layer corresponds to $l = 4$. The input to the network is denoted as $z^{(0)} = [x, y]^\top$, which consists of the spatial coordinates x and y. The transformations between consecutive layers are defined as:

$$z^{(l+1)} = \tanh\left(W^{(l)} z^{(l)} + b^{(l)}\right), \quad l = 0, 1, 2, \tag{15}$$

where $W^{(l)}$ and $b^{(l)}$ are the trainable weight matrix and bias vector for layer l, respectively.

After the final hidden layer ($l = 3$), the output layer computes the predicted temperature field without applying an activation function:

$$T_\theta(x, y) = W^{(3)} z^{(3)} + b^{(3)}. \tag{16}$$

Thus, the neural network maps the input spatial coordinates (x, y) to the approximated temperature field $T_\theta(x, y)$ through a series of nonlinear transformations governed by the weights and biases $\theta = \{W^{(l)}, b^{(l)}\}_{l=0}^{3}$.

This architecture allows the network to learn complex relationships between the spatial coordinates and the temperature field while being informed by physical constraints, as detailed in subsequent sections.

The diagram at Figure 2 illustrates the key components of a PINN used to solve a PDE with a complex domain defined by a NURBS boundary. The problem involves solving a heat equation with a heat source inside a irregular domain. The PINN architecture is trained using three types of loss functions: PDE loss, boundary condition loss, and fictitious domain loss. The diagram outlines the workflow from defining the domain and sampling points to training the neural network and computing the losses.

5.5. Boundary Condition Enforcement

One of the primary advantages of combining PINNs with FDM is the flexibility in handling boundary conditions. In traditional numerical methods, enforcing boundary conditions typically requires careful mesh generation along the domain boundaries, which can become especially cumbersome for irregular geometries. In contrast, the PINN-FDM framework embeds boundary conditions directly into the training process by incorporating them into the overall loss function, thus eliminating the need for explicit meshing or domain discretization near the boundary.

For Dirichlet boundary conditions, where the value of the solution is prescribed along the boundary, the network is penalized (3) in the loss function whenever it deviates from these prescribed values at the sampled boundary points. This encourages the learned solution to adhere to the boundary values throughout training, effectively enforcing the Dirichlet condition without architectural modifications.

For Neumann boundary conditions, where the normal derivative of the solution is specified along the boundary, an additional term is introduced into the loss function (4). This term computes the directional derivative of the network output at the boundary points using automatic differentiation—usually by projecting the gradient onto the boundary normal vector—and penalizes the squared error between this computed derivative and the prescribed Neumann value. This formulation ensures that the network not only learns the correct function values but also satisfies gradient constraints at the boundary.

By integrating both Dirichlet and Neumann conditions into the loss function, the PINN-FDM method can naturally accommodate complex and nonstandard boundary geometries. This flexibility allows the neural network to enforce physical laws at the boundary without requiring specialized treatment in the

network architecture or training scheme, thus simplifying implementation while maintaining accuracy.

6. Numerical results

6.1. Test Case Setup

In this section, we present the application of the PINN-FDM framework to several test cases involving irregular domains. The test cases include geometries such as ellipses, L-shaped regions, and domains with holes. For each test case, the specific parameters of the heat equation are detailed, including thermal conductivity, heat source terms, and boundary conditions. These cases are selected to demonstrate the versatility of the PINN-FDM approach in handling complex geometries that are challenging for traditional numerical methods.

One consider the steady-state heat equation with a heat source is given by:

$$\Delta u(\mathbf{x}) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega, \tag{17}$$

where: $u(\mathbf{x})$ is the temperature distribution at point $\mathbf{x} = (x, y)$, Δ is the Laplacian operator and $f(\mathbf{x})$ is the heat source term.

The heat source $f(\mathbf{x})$ is defined as:

$$f(\mathbf{x}) = \begin{cases} 100, & \text{if} \|\mathbf{x} - \mathbf{x}_c\| < r, \\ 0, & \text{otherwise}, \end{cases} \tag{18}$$

in this case we use $\mathbf{x}_c = (0.5, 0.5)$ is the center of the heat source and $r = 0.1$ is the radius of the heat source.

The domain Ω is a closed region with blue collocation points, bounded by Γ a NURBS curve represented by the dashed line in Figure 3. The boundary condition is:

$$u(\mathbf{x}) = u_D = 0, \quad \mathbf{x} \in \Gamma = \Gamma_D, \tag{19}$$

indicating that the temperature is zero on the boundary of the domain.

In the Fictitious Domain Method the real domain Ω is embedded into a larger regular domain $\tilde{\Omega} = [0, 1]^2$ (unit square). The problem is solved over the extended domain $\tilde{\Omega}$ by the solution $\tilde{u}$, and the boundary conditions are imposed using a penalty method.

The loss function in PINNs consists of several components:

PDE residual:

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(-\nabla \cdot (k \nabla \tilde{u}(\mathbf{x}_i)) - \tilde{f}(\mathbf{x}_i) \right)^2 \tag{20}$$

Dirichlet boundary condition:

$$\mathcal{L}_{\Gamma_D} = \frac{1}{N_D} \sum_{j=1}^{N_D} \left(\tilde{u}(\mathbf{x}_j) - u_D(\mathbf{x}_j) \right)^2 \tag{21}$$

Neumann boundary condition (if applicable):

$$\mathcal{L}_{\Gamma_N} = \frac{1}{N_N} \sum_{k=1}^{N_N} \left(k \nabla \tilde{u}(\mathbf{x}_k) \cdot \mathbf{n} - u_N(\mathbf{x}_k) \right)^2 \tag{22}$$

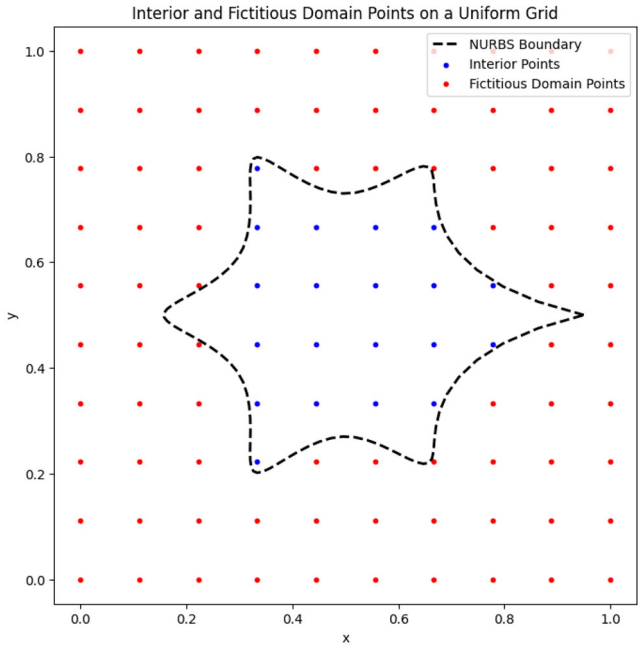


Figure 3. In this context, the collocation points employed in the PINNs framework are distributed to enhance identification.

The total loss function:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{PDE}} \mathcal{L}_{\text{PDE}} + \lambda_{\Gamma_D} \mathcal{L}_{\Gamma_D} + \lambda_{\Gamma_N} \mathcal{L}_{\Gamma_N} \tag{23}$$

Here we assume that all regulation parameters are unitary:

$$\lambda_{\text{PDE}} = \lambda_{\Gamma_D} = \lambda_{\Gamma_N} = 1 \tag{24}$$

In other words, each loss term contributes equally to the total loss. There is no prioritization or scaling of one type of constraint over another.

In practice, choosing appropriate values for the λ's can be crucial for training stability and accuracy. Sometimes, certain terms may dominate the loss due to differences in magnitude, so adjusting the λ's helps balance them. When all regulation parameters are unitary, we are assuming that all loss components are already on comparable scales, and we intentionally want to treat all constraints as equally important.

In the Figure 3 we illustrate the collocation points used in the Physics-Informed Neural Network framework. These points are strategically chosen throughout the domain to ensure accurate training and evaluation of the neural network. Additionally, the Figure 4 displays the results obtained by minimizing the loss function (23), with $\lambda_{\Gamma_N} = 0$ (no Neumann boundary conditions are imposed on the domain's boundary), which represents the discrepancy between the predicted solution and the actual solution. The minimization of this loss function is crucial for refining the network's accuracy and achieving a reliable approximation of the solution within the domain. This comprehensive visualization highlights the effectiveness of the PINN approach in capturing the solution's behavior and demonstrates how well the network has learned to satisfy the underlying physical constraints of the problem.

The results obtained using the PINN method, displayed in the Figure 4, demonstrate the capability of deep learning techniques to accurately approximate the solution of the heat equation

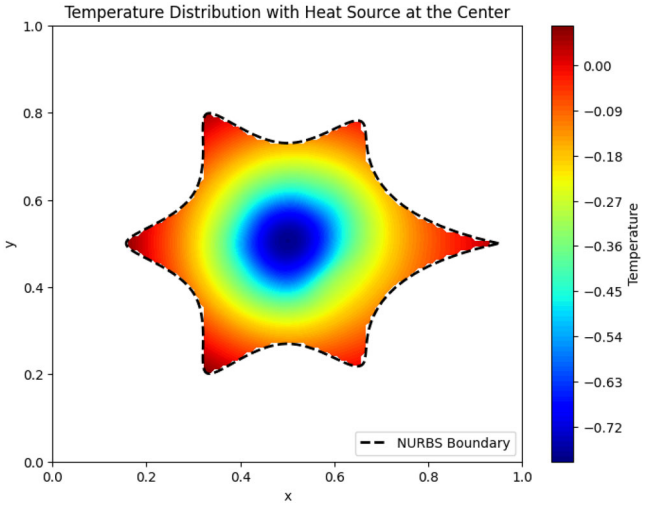


Figure 4. The results were obtained by minimizing the loss function, including terms (16) and (17).

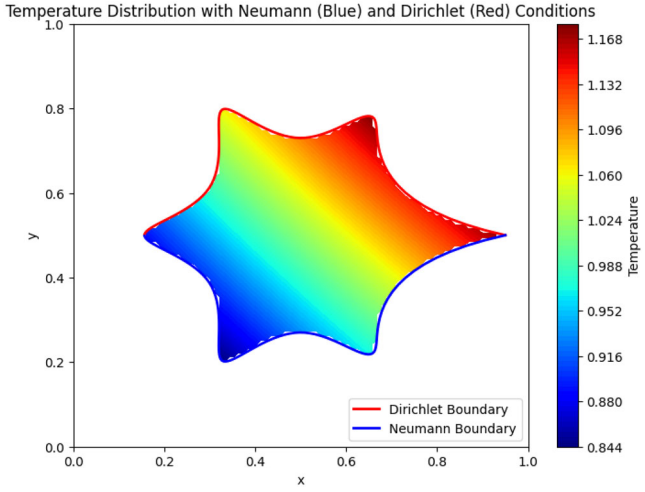


Figure 5. Contour plot of the temperature distribution within the NURBS-defined domain. The boundary of the domain, incorporates both Dirichlet (the red one) and Neumann (the blue one) boundaries.

with complex domain boundaries. The loss function showed consistent convergence over the training epochs, confirming the network's ability to satisfy the governing PDE and imposed boundary conditions.

The contour plots of the predicted temperature distribution align well with the numerical reference solution, further validating the accuracy of the PINN approach. Notably, the PINN method provides a smooth and continuous approximation, unlike traditional grid-based methods, which may introduce numerical artifacts due to discretization.

One limitation of the PINN method observed in this study is the computational cost associated with training the neural network, which is significantly higher than solving the system using FDM. However, the PINN approach remains advantageous for problems with complex geometries where traditional meshing techniques may be challenging to implement.

Overall, the PINN method successfully captured the temperature distribution with high accuracy, demonstrating its potential as a viable alternative to conventional numerical methods for solving PDEs in irregular domains.

6.2. Test Case Setup with Neumann Boundary Conditions

In the previous example, we explored the solution of a PDE using a PINN with Dirichlet boundary conditions. Building upon this foundation, we now extend the problem to include Neumann boundary conditions on a portion of the boundary Γ_N ($\Gamma_N = \Gamma \setminus \Gamma_D$). The Neumann condition specifies the normal derivative of the temperature field along Γ_N, given by:

$$\frac{\partial u}{\partial n}(x, y) = u_N(x, y), \quad (x, y) \in \Gamma_N, \tag{25}$$

In this example, we assume $u_N(x, y) = 0.5$ and $u_D(x, y) = 1.0$.

By incorporating both Dirichlet and Neumann boundary conditions, this extended formulation allows us to model more complex physical scenarios where different types of boundary constraints coexist. The PINN approach remains well-suited for solving such problems, as it seamlessly integrates the PDE residual, Dirichlet boundary conditions, and Neumann boundary conditions into a unified loss function (23).

The example illustrated in Figure 5 highlights the flexibility and robustness of PINNs in effectively handling a variety of boundary conditions. Despite the complexity and diversity of these conditions, the model consistently maintains high levels of accuracy and computational efficiency, demonstrating its strong potential for solving complex partial differential equations in practical scenarios.

Using the above PINN architecture, the training process minimizes the sum of three losses: the PDE residual loss, the Dirichlet boundary condition loss, and the Neumann boundary condition loss.

After training for 8000 epochs, the model predicts the temperature distribution within the domain. The results show that the PINN successfully learns the solution, as evidenced by the smooth temperature contours inside the NURBS boundary. The contour plot highlights the influence of the heat source at the center of the domain and the specified boundary conditions. Additionally, the loss curve demonstrates a steady decrease in the total loss over epochs, indicating effective convergence.

6.3. Computational Performance and Comparison with FEM

To assess the computational performance of the proposed PINN-FDM framework, the total training time was recorded for a typical simulation involving 1000 interior points, 200 NURBS-generated boundary points, and 200 fictitious domain points. The training was carried out on a standard desktop computer equipped with an Intel Core i7 CPU (3.2 GHz, 8 cores) and 32 GB of RAM, without the use of GPU acceleration.

The training required approximately 12 minutes for 8000 epochs using the Adam optimizer with a fixed learning rate of 0.001. The loss curve, shown in Figure 6, exhibited steady convergence, with a final residual loss on the order of 10^{-4}, indicating a sufficiently accurate solution for the steady-state heat equation with a localized source.

To benchmark the quality of the PINN solution, we compared it against a traditional FEM solution implemented in FEniCS Scroggs et al. (2022). Specifically, we consider the domain $\Omega =$

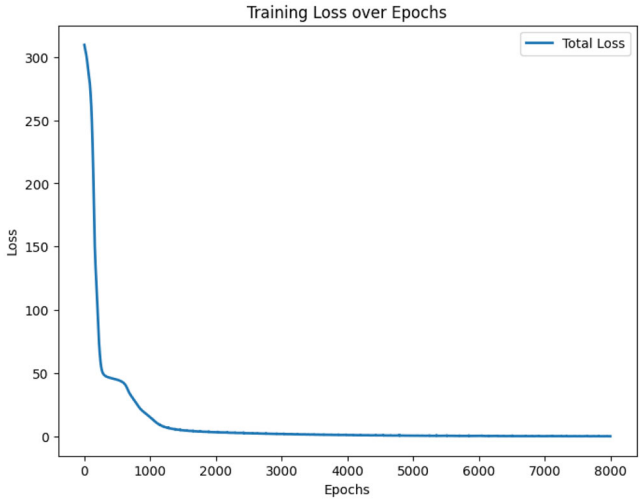


Figure 6. Convergence behavior of the training process. The residual loss decreases steadily, reaching a value on the order of 10^{-4}, which indicates the model's successful approximation of the steady-state heat equation with a localized source.

$\tilde{\Omega}$, where $\tilde{\Omega}$ represents the fictitious domain used in the PINN formulation. The problem under consideration is detailed in Subsection 6.1. The FEM solution served as a ground-truth reference for evaluating the relative Mean Error of the PINN prediction.

Figure 7 presents a side-by-side comparison between the temperature distribution obtained using the traditional FEM, shown in Figure 7(a), and the solution produced by the PINN, shown in Figure 7(b). Both solutions are evaluated over the entire computational domain, including the fictitious region outside the physical boundary defined by the NURBS curve.

The FEM solution serves as a reference, leveraging a finely discretized unstructured mesh that conforms to the complex geometry of the domain. In contrast, the PINN solution is obtained without any mesh generation, relying solely on scattered sampling points and the incorporation of boundary and PDE constraints into the training loss.

Visual inspection reveals a high degree of similarity between the two temperature fields, particularly in the region surrounding the central heat source and along the NURBS-defined boundary. The contours, gradients, and overall spatial structure of the temperature distribution match closely, demonstrating that the PINN framework is capable of capturing the essential physics of the problem with comparable fidelity to the FEM benchmark.

This strong agreement not only validates the accuracy of the PINN-FDM method but also underscores its potential as a mesh-free alternative for solving partial differential equations in geometrically complex domains. Moreover, the flexibility of the PINN approach in handling such irregular boundaries without requiring domain-conforming meshes highlights its advantage in scenarios where remeshing or mesh adaptation would otherwise be necessary.

Figure 8 represents the pointwise error distribution between the two obtained solutions, displayed as a color map. The largest discrepancies are observed near the boundaries, likely due to differences in approximation techniques between PINN and FEM. This provides context for interpreting the figure and highlights areas where the error is most pronounced.

Quantitatively, the relative Mean Error between the PINN prediction and FEM solution over the domain was approximately 3%, as shown in Figure 8, which is competitive given the smoothness of the PINN approximation and the lack of explicit meshing or remeshing. Qualitative comparisons via contour plots of the temperature field showed excellent agreement in both the central region (containing the heat source) and near the complex boundary.

While FEM generally achieves higher accuracy per degree of freedom, the PINN-FDM framework offers notable advantages in terms of geometric flexibility and mesh-independence. In particular, the use of NURBS for boundary representation allows for smooth, high-fidelity domain definitions without the overhead of mesh generation, which is particularly beneficial in applications involving evolving or parameterized geometries.

These results suggest that for problems where meshing is difficult or dynamic boundaries are involved, PINNs augmented with fictitious domain techniques provide a viable and efficient alternative to traditional numerical solvers.

7. Error Estimation Using Neural Network

Considering a partial differential equation defined over a domain $\Omega \subset \mathbb{R}^d$ with a true solution $u \in \mathcal{H}$, where $\mathcal{H}$ is a suitable function space. For instance, if u is sufficiently smooth, we might choose $\mathcal{H}$ to be a Sobolev space $H^k(\Omega)$ or a space of continuous functions $C(\Omega)$. In this demonstration, we use the Sobolev space $H^k(\Omega)$, consisting of functions whose derivatives up to order k are square-integrable.

The approximation capability of neural networks can be analyzed using results from approximation theory. Specifically, for neural networks with ReLU activation functions, the following result is pertinent:

Theorem 7.1. ***Universal Approximation Theorem for ReLU Networks***

Let $\mathcal{F}$ be the class of functions representable by a neural network with one hidden layer of N neurons and ReLU activation functions. Then, for any continuous function $f \in C(\Omega)$ and any $\epsilon > 0$, there exists a function $\hat{f}_\theta \in \mathcal{F}$ such that:

$$\|f - \hat{f}_\theta\|_{L^2(\Omega)} \leq \epsilon$$

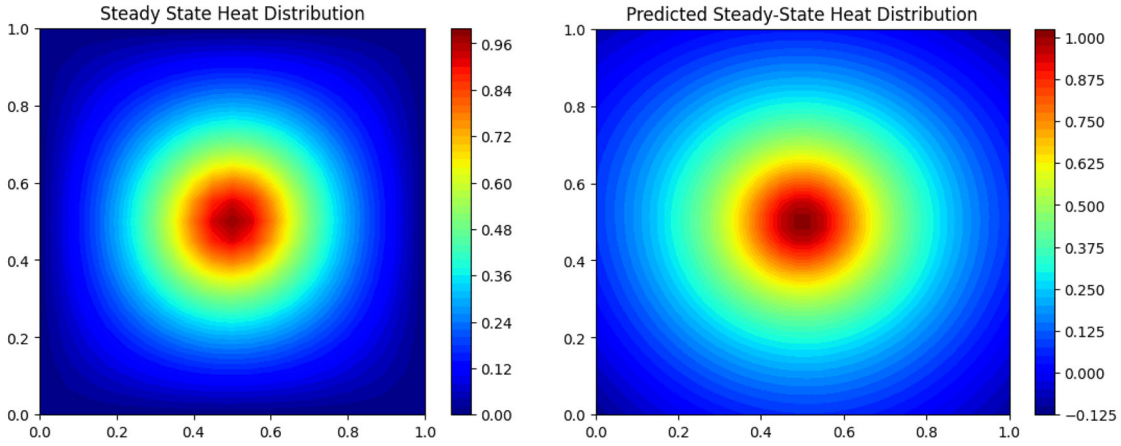
provided that N is sufficiently large.

This result is well-established in the literature Cybenko (1989), Hornik (1991).

For neural networks with multiple hidden layers and neurons, the approximation capability improves. Specifically, for a neural network with L hidden layers, each with N neurons, the approximation error can be bounded as follows:

Theorem 7.2. ***Approximation Error Bound for Deep Networks***

Let $u \in C^k(\Omega)$ be the true solution and $\hat{u}_\theta$ be the neural network approximation with L hidden layers and N neurons per



(a) A numerical FEM solution. (b) A numerical PINNs solution.

Figure 7. Comparison of the numerical solutions obtained over the fictitious domain.

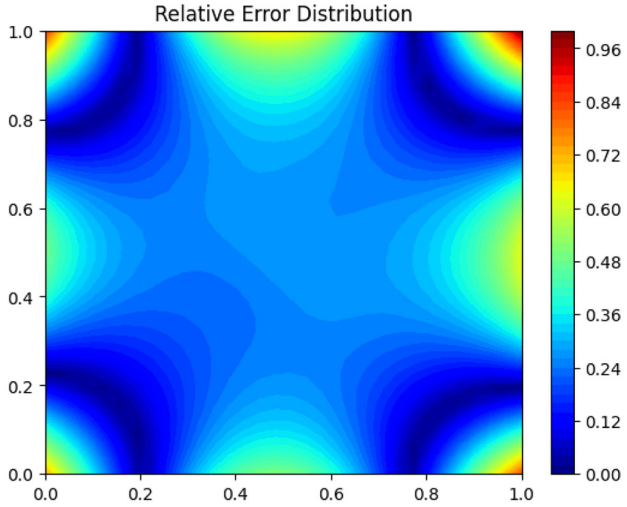


Figure 8. Mean error between FEM and PINN: 0.02927.

layer. The approximation error ϵ_{approx} can be bounded by:

$$\epsilon_{approx} = \|u - \hat{u}_\theta\|_{L^2(\Omega)} \leq C \cdot \left(\frac{1}{N}\right)^{\frac{k}{d}} \cdot (L+1)^d$$

where C is a constant dependent on u, d is the dimension of the input space, and L is the number of hidden layers.

This bound is derived from results in approximation theory and neural network capacity analysis Barron (1993), Devroye (1996).

Here we provide a sketch of the proof for the approximation bound using neural networks with ReLU activations:

Proof. 1. Network Architecture and Capacity: Consider a neural network with L hidden layers, each containing N neurons. The network function can be expressed as:

$$\hat{u}_\theta(x) = \sum_{i=1}^{N} w_i \phi\left(\sum_{j=1}^{L} V_{ij} \cdot x + b_j\right) + b_0$$

where ϕ is the ReLU activation function, w_i and V_{ij} are weights, and b_j and b_0 are biases.

2. Approximation Capability: By the universal approximation theorem, a neural network with one hidden layer can approximate any continuous function to arbitrary accuracy if the number of neurons N is sufficiently large.

3. Approximation Error with Multiple Layers: With multiple hidden layers, the network's ability to represent complex functions improves. The approximation error bound for a function $u \in C^k(\Omega)$ is given by:

$$\epsilon_{approx} = \|u - \hat{u}_\theta\|_{L^2(\Omega)} \leq C \cdot \left(\frac{1}{N}\right)^{\frac{k}{d}} \cdot (L+1)^d$$

where the term $\left(\frac{1}{N}\right)^{\frac{k}{d}}$ reflects the error reduction with increasing N, and $(L+1)^d$ reflects the increased capacity with deeper networks.

For a single hidden layer network, the bound on the approximation error is:

$$\epsilon_{approx} \leq C \cdot N^{-\frac{k}{d}}$$

where C is a constant depending on the function u and the domain Ω.

For deep networks with L layers and N neurons per layer, the bound becomes:

$$\epsilon_{approx} \leq C \cdot \left(\frac{1}{N}\right)^{\frac{k}{d}} \cdot (L+1)^d$$

Here, C incorporates constants related to the specific network architecture and the true solution u. □

8. Conclusion and Future Work

8.1. Summary of Key Findings

This approach combining Physics-Informed Neural Networks with the Fictitious Domain Method demonstrates substantial promise in solving the steady-state heat equation within irregular domains. This method capitalizes on the unique strengths

of both PINNs and FDM to address challenges associated with complex geometries. Specifically, PINNs offer a flexible framework for incorporating the physics of the problem directly into the neural network's training process, while FDM simplifies the domain by embedding irregular geometries into a larger, regular domain.

Our findings indicate that this integration leads to highly accurate solutions with a notable reduction in the complexity of the meshing process. The PINN-FDM approach effectively manages boundary conditions that would otherwise complicate traditional numerical methods, thereby offering a robust solution for problems involving intricate domain shapes. Furthermore, the method's ability to generalize across different types of elliptic PDEs demonstrates its versatility and broad applicability.

8.2. Advantages and Limitations

The proposed PINN-FDM method boasts several advantages:

- Handling Complex Boundaries: One of the most significant benefits is its ability to address complex boundary conditions without the need for traditional meshing. This feature simplifies the problem setup and reduces the computational overhead associated with generating and managing meshes in irregular domains.
- Flexibility and General Applicability: The method's flexibility extends beyond the heat equation, making it applicable to a wide range of elliptic PDEs and potentially other types of PDEs. Its adaptability to various boundary conditions and domain shapes highlights its potential for solving diverse physical problems.

However, there are also some limitations:

- Training Time: One of the notable challenges is the longer training times associated with PINNs compared to classical solvers. The iterative nature of training neural networks, particularly with large or complex architectures, can result in extended computational times, which may impact the overall efficiency of the method.
- Network Architecture Sensitivity: The accuracy and performance of the PINN-FDM approach can be sensitive to the choice of network architecture and hyperparameters. Optimizing these aspects requires careful tuning and may involve a trade-off between accuracy and computational cost.

8.3. Future Directions

One major direction for future development is the extension of the PINN-FDM framework to time-dependent problems. At present, the methodology is focused on steady-state partial differential equations. However, incorporating time-dependent behavior would significantly broaden its scope and applicability, enabling it to tackle a wider class of physical phenomena such as transient heat transfer and time-evolving systems. This extension would require integrating temporal variables into the neural network architecture and modifying the loss function to accurately capture the solution's evolution over time. A promising pathway for this advancement lies in applying the techniques developed for time-dependent problems in Rodrigues (2024a), where Rodrigues (2024) successfully demonstrated the use of PINNs in modeling tumor cell growth. Adapting these temporal modeling strategies to the PINN-FDM context could provide a robust foundation for solving dynamic PDEs within complex geometries.

Another promising avenue is the improvement of network architectures. Enhancing the design of neural networks used within the PINN-FDM framework could lead to faster convergence and increased accuracy. Research could focus on novel neural network architectures, such as deep learning models with advanced features or hybrid networks that leverage different types of neural units. Additionally, exploring advanced training techniques, such as improved optimization algorithms or adaptive learning strategies, could address current challenges related to longer training times and network performance. When mentioning future work on enhancing network architectures, cite current trends or recent breakthroughs in neural network design that could be applicable. Examples might include the integration of convolutional layers to better handle spatial hierarchies or attention mechanisms to improve the focus on critical features of the PDE solutions.

An interesting direction for future research involves exploring the use of advanced activation functions within the PINN framework to enhance training efficiency and convergence stability. In particular, the recently proposed Scaled Polynomial Constant Unit (SPOCU) activation function has shown promising results in improving gradient flow and convergence speed in classification tasks by maintaining self-normalizing properties throughout the network layers Kiseľák, Lu, and Švihra (2021). Integrating SPOCU into selected layers of the PINN architecture may help mitigate issues related to vanishing gradients and slow convergence, which are common challenges in deep learning-based PDE solvers. Preliminary investigations could evaluate the performance impact of replacing traditional tanh activations with SPOCU in terms of training time and solution accuracy. This line of research could pave the way for more robust and efficient PINN-based solvers in complex geometries and high-dimensional problems.

The application to multi-physics problems represents a further area of exploration. Multi-physics problems involve simultaneous interactions between different physical processes, such as coupled heat transfer and fluid dynamics. Applying the PINN-FDM method to these complex scenarios could provide valuable insights and solutions where traditional methods might struggle. This involves developing strategies to handle multiple interacting PDEs within the same framework and ensuring that the neural network can accurately model and solve these interconnected phenomena.

Finally, focusing on the optimization of computational resources is crucial for enhancing the practical utility of the PINN-FDM approach. Future research could aim at reducing the computational demands associated with training and implementing the PINN-FDM method. Techniques such as parallel computing, which distributes computational tasks across multiple processors, and the development of more efficient algorithms could help mitigate the resource-intensive nature of neural network training. These advancements would make the PINN-FDM method more accessible and feasible for large-scale or real-time applications.

In conclusion, the PINN-FDM method represents a significant advancement in solving elliptic PDEs in irregular domains, offering a flexible and efficient alternative to traditional numerical methods. By addressing current limitations and exploring future directions, this approach has the potential to make substantial contributions to the field of numerical analysis and its applications across various scientific and engineering disciplines.

Acknowledgements

This research was partially sponsored with national funds through the Fundação Nacional para a Ciência e Tecnologia, Portugal-FCT, under projects UIDB/04674/2020 (CIMA).
DOI: https://doi.org/10.54499/UIDB/04674/2020

ORCID

José A. Rodrigues http://orcid.org/0000-0001-5630-7149

References

Atluri S N, Zhu T. 1998. A new Meshless Local Petrov-Galerkin (MLPG) approach in computational mechanics. Comput. Mech. 22:117–127. doi: 10.1007/s004660050346.

Babuška Ivo. 1973. The finite element method with Lagrangian multipliers. Numer Math. 20:179–192. doi: 10.1007/BF01436561.

Barron A R. 1993. Universal approximation bounds for superpositions of a sigmoidal function. Neural Netw. 4:213–215.

Bazilevs Y, Calo VM, Cottrell JA, Hughes TJR, Reali A, Scovazzi G. 2007. Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows. Computer Methods in Applied Mechanics and Engineering. 197:173–201. doi: 10.1016/j.cma.2007.07.016.

Belytschko T, Lu Y Y, Gu L. 1994. Element-free Galerkin methods. Numerical Meth Engineering. 37:229–256. doi: 10.1002/nme.1620370205.

Burman E, Hansbo P. 2012. Fictitious domain finite element methods using cut elements: II. A stabilized Nitsche method. Appl. Numer. Math. 62:328–341. doi: 10.1016/j.apnum.2011.01.008.

Cottrell, J A, TJR, Hughes, Y, Bazilevs, 2009. *Isogeometric Analysis: Toward Integration of CAD and FEA*: Wiley.

Cybenko G. 1989. Approximation by superpositions of a sigmoidal function. Math Control Signal Systems. 2:303–314. doi: 10.1007/BF02551274.

Devroye, L, L, Gyorfi, G, Lugosi, 1996. *A Probabilistic Theory of Pattern Recognition*: Springer.

Glowinski, R, J-L, Lions, R, Tremolieres, 1984. *Numerical Analysis of Variational Inequalities*, North-Holland,

Glowinski R, Pan T W, Periaux J, Thomasset F. 2001. Fictitious domain methods for incompressible viscous flow around moving rigid bodies. Computational Fluid Dynamics Journal. 9:111–130.

Glowinski R, He J, Le Tallec P. 2003. On an iterative method for the solution of saddle point problems. J. Comput. Appl. Math. 152:219–233.

Hornik K. 1991. Approximation capabilities of multilayer feedforward networks. Neural Netw. 4:251–257. doi: 10.1016/0893-6080(91)90009-T.

Hughes TJR, Cottrell JA, Bazilevs Y. 2005. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering. 194:4135–4195. doi: 10.1016/j.cma.2004.10.008.

Jagtap AD, Kawaguchi K, Karniadakis GE. 2020. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Comput. Phys. 404:109136 doi: 10.1016/j.jcp.2019.109136.

Jagtap A D, Kawaguchi K, Karniadakis G E. 2019. Locally adaptive activation functions with slope recovery for deep and physics-informed neural networks. Proc. R. Soc. 2020:476.

Karniadakis GE, Kevrekidis IG, Lu Lu, Perdikaris P, Wang S, Yang Liu. 2021. Physics-informed machine learning. Nat Rev Phys. 3:422–440. doi: 10.1038/s42254-021-00314-5.

Khoo Y, Lu J, Ying L. 2021. Solving parametric PDE problems with artificial neural networks. Eur. J. Appl. Math. 32:72–93.

Kiseľák J, Lu Y, Švihra Ján, Szépe P, Stehlík M. 2021. SPOCU": scaled polynomial constant unit activation function. Neural Comput & Applic. 33:3385–3401. doi: 10.1007/s00521-020-05182-1.

Lions P L, Mercier B. 1979. Splitting Algorithms for the Sum of Two Nonlinear Operators. SIAM J Numer Anal. 16:964–979. doi: 10.1137/0716071.

Li S, Liu WK. 2002. Meshfree and particle methods and their applications. Applied Mechanics Reviews. 55:1–34. doi: 10.1115/1.1431547.

Liu, G R, 2003. *Mesh Free Methods: Moving Beyond the Finite Element Method*: CRC Press.

Peskin CS. 2002. The immersed boundary method. Acta Numer. 11:479–517. doi: 10.1017/S0962492902000077.

Raissi M, Perdikaris P, Karniadakis GE. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. Comput. Phys. 378:686–707. doi: 10.1016/j.jcp.2018.10.045.

Ramière I, Angot P, Belliard M. 2007. A general fictitious domain method with immersed jumps and multilevel nested structured meshes. Comput. Phys. 225:1347–1387. doi: 10.1016/j.jcp.2007.01.026.

Rodrigues JA. 2020. Isogeometric Analysis for Fluid Shear Stress in Cancer Cells. MCA. 25:19 doi: 10.3390/mca25020019.

Rodrigues JA. 2024. Using Physics-Informed Neural Networks (PINNs) for Tumor Cell Growth Modeling. Mathematics. 12:1195 doi: 10.3390/math12081195.

Rodrigues J A. 2024. Exploring Linear Elasticity: Unveiling the Power of Physics-Informed Neural Networks (PINNs). In: International Workshop on Mathematics and Physical Sciences (Matphys); 2024 Jul 11–12. Évora (Portugal): CIMA – Center for Research in Mathematics and Applications, University of Évora. Oral communication.

Scroggs M W, Dokken J S, Richardson C N, Wells G N. Construction of arbitrary order finite element degree-of-freedom maps on polygonal and polyhedral cell meshes. ACM Trans. Math. Softw. 2022:23.

Sirignano J, Spiliopoulos K. 2018. DGM: A deep learning algorithm for solving partial differential equations. Comput. Phys. 375:1339–1364. doi: 10.1016/j.jcp.2018.08.029.

Wu G, Wang Fajie, Qiu Lin. 2023. Physics-Informed Neural Network for Solving Hausdorff Derivative Poisson Equations. Fractals. 2340103.

Zhang D, Guo L. 2020. Hybrid Physics-Informed Neural Network and Finite Element Method for High-Dimensional Inverse Problems. Comput. Phys. 409:109–124.

Zhang D, Guo L, Karniadakis G E. 2021. Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks. SIAM J. Sci. Comput. 42:243–271.

Zhang Benrong Wu Guozheng Gu Yan Wang X, Wang F. 2022. Multi-domain physics-informed neural network for solving forward and inverse problems of steady-state heat conduction in multilayer media. Physics of Fluids. 34:116116.

Zhang Benrong Wang Fajie, Qiu Lin. 2023. Multi-domain physics-informed neural networks for solving transient heat conduction problems in multilayer materials. J. Appl. Phys. 133:245103.

Zhu Y, Zabaras N, Koutsourelakis P-S, Perdikaris P. 2019. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. Comput. Phys. 394:56–81. doi: 10.1016/j.jcp.2019.05.024.