Neural Networks

An efficient wavelet-based physics-informed neural network for multiscale problems

--Manuscript Draft--

Manuscript Number:	NEUNET-D-25-04760
Article Type:	Article
Section/Category:	Learning Systems
Keywords:	Physics-informed neural networks; Wavelets; Singularly perturbed problems; Multiscale problems
Abstract:	Physics-informed neural networks (PINNs) are a class of deep learning models that utilize physics in the form of differential equations to address complex problems, including those that may involve limited data availability. However, solving differential equations with rapid oscillations, steep gradients, or singular behavior becomes a challenge for PINNs. To address this, we propose an efficient wavelet-based PINN (W-PINN) that learns the solutions in wavelet space. Here, we represent the solution in wavelet space using a family of localized wavelets. This framework represents the solution of a differential equation with significantly fewer degrees of freedom while retaining the dynamics of complex physical phenomena. The proposed architecture enables the training process to search for solutions within the wavelet domain, where the multiscale characteristics are less pronounced compared to the physical domain. This facilitates a more efficient training for this class of problems. Furthermore, the proposed model does not rely on automatic differentiation (AD) for derivatives involved in the loss function and does not require any prior information regarding the behavior of the solution, such as the location of abrupt features. The removal of the AD requirement significantly reduces training time while maintaining accuracy. Thus, through a strategic fusion of wavelets with PINNs, W-PINNs excel at capturing localized non-linear information, making them well-suited for problems showing abrupt behavior in certain regions, such as singularly perturbed and other multiscale problems. We further analyze the convergence behavior of W-PINN through a comparative study using the Neural Tangent Kernel (NTK) theory. The efficiency and accuracy of the proposed neural network model are demonstrated in various problems, i.e., the FitzHugh-Nagumo (FHN) model, the Helmholtz equation, the Maxwell equation, the Allen-Cahn equation, and lid-driven cavity flow, along with other highly singularly perturbed non-linear differential equations.

Dr. Ratikanta Behera
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Bangalore
Karnataka, India. 560012.
Email: ratikanta@iisc.ac.in

To,
Prof. DeLiang Wang,
Editor-in-Chief,
Neural Networks

September 17, 2025

Cover Letter

Dear Prof. DeLiang Wang,

I am pleased to submit the original manuscript entitled "***An efficient wavelet-based physics-informed neural network for multiscale problems***" for consideration for publication in "*Neural Networks*".

The key contribution of the work can be summarised as follows:

- A Wavelet-based Physics-informed Neural Network (W-PINN) architecture is developed for multiscale problems.
- W-PINN eliminates the need for automatic differentiation in computing derivatives in the loss function, making the training process significantly faster.
- W-PINN addresses the loss balancing issue of PINN for singular or multiscale problems.
- The proposed method is validated through the FitzHugh-Nagumo (FHN) model, Helmholtz equation, Allen-Cahn equation, Maxwell's equation, Lid-driven cavity flow, and various other singularly perturbed problems.
- The performance of W-PINN is evaluated against several state-of-the-art methods. Furthermore, a comparative analysis based on the neural tangent kernel (NTK) theory is presented to demonstrate the accelerated convergence of the proposed approach.

A preprint of this paper is available on arxiv with Ref. No. arXiv:2409.11847

We sincerely thank you for your time and consideration, and we look forward to the opportunity to contribute to the "Neural Networks".

Sincerely,

Dr. Ratikanta Behera
Assistant Professor
Dept. of Computational and Data Sciences
Indian Institute of Science
Bangalore- 560012, India

Place: IISc Bangalore

Date: September 17, 2025

Highlights of the Manuscript:

- A Wavelet-based Physics-informed Neural Network (W-PINN) architecture is developed for multiscale problems.
- W-PINN eliminates the need for automatic differentiation in computing derivatives in the loss function, making the training process significantly faster.
- W-PINN addresses the loss balancing issue of PINN for singular or multiscale problems.
- The proposed method is validated through the FitzHugh-Nagumo (FHN) model, Helmholtz equation, Allen-Cahn equation, Maxwell's equation, Lid-driven cavity flow, and various other singularly perturbed problems.
- The performance of W-PINN is evaluated against several state-of-the-art methods. Furthermore, a comparative analysis based on neural tangent kernel (NTK) theory is presented to demonstrate the accelerated convergence of the proposed approach.

An efficient wavelet-based physics-informed neural network for multiscale problems

Himanshu Pandey[1], **Anshima Singh**[2] and **Ratikanta Behera**[1*]

[1] *Department of Computational and Data Sciences, Indian Institute of Science, Bangalore, 560012, India*
[2] *Department of Mathematics, University of Manchester, Oxford Road, Manchester M13 9PL, UK*

Abstract. Physics-informed neural networks (PINNs) are a class of deep learning models that utilize physics in the form of differential equations to address complex problems, including those that may involve limited data availability. However, solving differential equations with rapid oscillations, steep gradients, or singular behavior becomes a challenge for PINNs. To address this, we propose an efficient wavelet-based PINN (W-PINN) that learns the solutions in wavelet space. Here, we represent the solution in wavelet space using a family of localized wavelets. This framework represents the solution of a differential equation with significantly fewer degrees of freedom while retaining the dynamics of complex physical phenomena. The proposed architecture enables the training process to search for solutions within the wavelet domain, where the multiscale characteristics are less pronounced compared to the physical domain. This facilitates a more efficient training for this class of problems. Furthermore, the proposed model does not rely on automatic differentiation (AD) for derivatives involved in the loss function and does not require any prior information regarding the behavior of the solution, such as the location of abrupt features. The removal of the AD requirement significantly reduces training time while maintaining accuracy. Thus, through a strategic fusion of wavelets with PINNs, W-PINNs excel at capturing localized non-linear information, making them well-suited for problems showing abrupt behavior in certain regions, such as singularly perturbed and other multiscale problems. We further analyze the convergence behavior of W-PINN through a comparative study using the Neural Tangent Kernel (NTK) theory. The efficiency and accuracy of the proposed neural network model are demonstrated in various problems, i.e., the FitzHugh-Nagumo (FHN) model, the Helmholtz equation, the Maxwell equation, the Allen-Cahn equation, and lid-driven cavity flow, along with other highly singularly perturbed non-linear differential equations.

AMS subject classifications: 65L11, 68T07, 65T60

Key words: Physics-informed neural networks, Wavelets, Singularly perturbed problems, Multiscale problems

1. Introduction

The domain of computational science has been profoundly reshaped in recent years by the emergence of artificial intelligence. Among these developments, physics-informed neural networks (PINNs) [1, 2] have gained substantial recognition as a powerful alternative to traditional numerical methods for solving partial differential equations (PDEs). Traditional numerical methods such as finite difference, finite element, and finite volume approaches have long served as the workhorses of computational science. However, these techniques often struggle with complex geometries that require intricate mesh generation and face significant challenges when scaling to higher-dimensional problems. PINNs address these limitations by offering a mesh-free approach that eliminates the tedious process of mesh construction and the associated numerical artifacts that arise from poor mesh quality [3]. What makes PINNs particularly compelling is their ability to generalize continuous functional representations across the entire computational domain rather than discrete solution points.

*Corresponding author. *Email addresses:* phimanshu@iisc.ac.in (H. Pandey), anshima.singh@manchester.ac.uk (A. Singh), ratikanta@iisc.ac.in (R. Behera)

Different types of PINN have been developed, such as variational physics-informed neural networks (hp-VPINN) [4], gradient-enhanced physics-informed neural networks (GPINN) [5], and extended physics-informed neural networks (XPINN) [6], and many others to enhance the performance of conventional PINN. For more details, see a comprehensive review of PINNs [7, 8] and references therein.

Despite their many significant advantages, the performance of PINNs significantly deteriorates when dealing with problems that exhibit high gradients, rapid oscillations, or singular behavior [8]. Wang et al. demonstrate through extensive numerical experiments that conventional PINN methods exhibit significant limitations when applied to multiscale problems [9, 10]. Singularly perturbed differential equations represent another challenging class, as their solutions exhibit thin transition layers, often adjacent to the domain boundaries. These solutions or their derivatives undergo rapid fluctuations within specific areas while maintaining smooth behavior elsewhere [11], which presents particular difficulties for standard PINN approaches. A key reason for these challenges is that various components in the loss function have significantly different magnitudes and converge at different rates during training, complicating the optimization process [12]. Furthermore, when tackling multiscale problems characterized by multi-frequency or high-frequency features, conventional PINN method faces dual challenges: not only does it encounter these loss-term magnitude imbalances, but it also struggles to accurately capture high-frequency functions due to spectral bias [13]. These combined limitations ultimately result in compromised prediction accuracy. These limitations are also theoretically verified by Neural Tangent Kernel (NTK) theory for neural network learning, which was originally formulated by Jacot et al. [14]. Subsequently, Cao et al. [15] provided a detailed spectral analysis of NTK, and later Wang et al. [16] provided a similar analysis for PINNs.

The recent literature has proposed several approaches to address these limitations. McClenny et. al. introduced Self-adaptive PINNs (SA-PINNs)[17] that employ trainable weights to dynamically balance loss terms, while Arzani et al. [18] proposed a BL-PINN approach that splits the domain into inner and outer regions to effectively handle boundary layer problems with singular perturbation. Moreover, Wang et al. developed a practical PINN framework for multiscale problems with multi-magnitude loss terms (MMPINN)[19], incorporating specialized architectures and regularization strategies. Although these methods demonstrate improved performance, they still face computational overhead from automatic differentiation and require careful tuning of multiple hyperparameters. Traditional PINNs typically use automatic differentiation (AD) to compute derivative terms. Although AD provides machine-precision derivatives, it comes at the cost of maintaining computation-intensive computational graphs that grow increasingly complex with network depth and order of derivatives involved. The automatic differentiation (AD) mechanism in traditional PINNs remains a computational bottleneck that the aforementioned variants fail to address fully.

In addition, researchers have proposed alternative methods to address the computational overhead inherent in AD in PINNs. Monte Carlo approximation techniques for second-order PDE derivatives were explored by Sgouralis and Spiliopoulos [20]. Navaneeth et al. developed a gradient-free learning methodology utilizing random projections, successfully modeling complex fourth-order phase field fracture scenarios [21]. Other developments include using meshless radial basis functions by Xiao et al. [22] to compute spatial derivatives, which theoretically accommodates random collocation point arrangements. One more recent development in this direction is DT-PINN [23], which achieved two to four times speedups over AD-based PINNs by eliminating AD through meshless radial basis function-finite differences (RBF-FD). These methods help mitigate computational inefficiencies and improve accuracy. However, derivative-based schemes can encounter difficulties when applied to irregular geometries. Certain methods fail to generalise to high-order PDEs, while others require the use of additional collocation points outside the physical domain. Designing a method that is both efficient and robust remains a crucial point toward resolving these persistent challenges.

Building upon these foundations, the motivation for our present work is two-fold. First, we seek to overcome the challenge of multi-magnitude residuals, which limit the effectiveness of conventional PINNs in capturing sharp transitions and high-frequency features in multiscale problems. Second, we aim to reduce the significant computational overhead associated with AD techniques that dominate

existing implementations. By developing a novel wavelet-based physics-informed neural network PINN) approach that simultaneously addresses both limitations, we propose an efficient alternative eliminates the memory-intensive computational graphs of automatic differentiation while dynamically balancing loss terms across varying scales.

The localization properties of wavelets make them beneficial n analyzing problems with localized features, such as initial or boundary layers, in singularly perturb ifferential equations [24, 25]-as well as sharp transitions and high-frequency components in multiscale problems. Mallat's work [26] and Daubechies' contributions [27] lay down the foundational principles of wavelet analysis. Wavelets excel at capturing non-linear information within localized regions of the solution [28, 29]. In contrast, neural networks are well known for learning and modeling global non-linear relationships. By incorporating wavelets with neural networks, we can potentially leverage the ability of wavelets to extract these localized non-linearities. This can complement the neural network's non-linear learning capability and potentially enhance the overall ability to capture intricate non-linear patterns, possibly improving the accuracy of the network. Incorporation of wavelets into PINNs has been attempted in [30]. However, the authors of [30] focused on wavelets solely as activation functions, with the overall structure of PINNs remaining unchanged from its original introduction. Our findings show that this method also fails to satisfactorily approximate the problems under consideration.

In this study, we use the wavelet family to represent an approximate solution to the differential equation, and the weight of each family member is optimized using the W-PINN. The proposed neural architecture does not rely on the automatic differentiation of the derivatives involved in the loss function, significantly reducing training time. In addition, this method does not require prior information about the nature of the solution, making it practical and easy to implement. We used Gaussian and Mexican hat wavelets for implementation, and their performance was compared with conventional PINN and state-of-the-art methods to validate the efficacy. Our approach demonstrates high accuracy across a range of differential equations that exhibit steep gradients, rapid oscillations, singularities, and multiscale behavior, establishing it as a robust method for these class problems. The key contributions of this work can be summarized as follows.

- We introduce a W-PINN framework that eliminates automatic differentiation in loss function derivative computation, significantly accelerating training while maintaining or improving solution accuracy compared to recent methods in the literature.

- W-PINN effectively addresses the loss balancing challenges inherent in conventional PINNs, particularly for problems that exhibit multiscale phenomena, singular behavior, or rapid oscillations in their solutions.

- The proposed method is validated through NTK theory and various examples, including the FHN model, Helmholtz equation, Allen-Cahn equation, Maxwell equation, Lid-driven cavity flow, and various other singularly perturbed problems.

The organization of the remaining paper is summarized as follows: Section 2 initiates with the introduction of standard PINNs, a summary of NTK theory for PINNs, followed by a thorough discussion on the design and operational mechanism of W-PINNs. Section 3 presents numerical results along with a comparison with other well-known methods in the literature for various differential equations. In this section, we also demonstrate ch faster convergence of W-PINN using NTK theory. Finally, Section 4 serves as the concluding se t, offering a concise summary of the key findings, contributions, and future work along with limitations.

2. Methodology

2.1. Physics-informed neural networks (PINNs)

A neural network is a mathematical model consisting of layers of neurons interconnected via non-linear operations. These layers include an input layer for initial data, multiple hidden layers for complex computations, and an output layer for predictions. It is widely acknowledged that with enough hidden layers and neurons per layer, a neural network can approximate any function [31]. The values of neurons in each layer depend on the previous layer in the following manner:

$$\begin{aligned} &\text{Input layer:} && \boldsymbol{z}^0 = \mathbf{x} \in \mathbb{R}^{n_0}, \\ &\text{Hidden layers:} && \boldsymbol{z}^k = \sigma(\boldsymbol{\omega}^k \boldsymbol{z}^{k-1} + \boldsymbol{b}^k),\ 1 \leq k \leq L-1, \\ &\text{Output layer:} && \boldsymbol{z}^L = \boldsymbol{\omega}^L \boldsymbol{z}^{L-1} + \boldsymbol{b}^L \in \mathbb{R}^{n_L}, \end{aligned} \tag{1}$$

where $\boldsymbol{z}^k \in \mathbb{R}^{n_k}$ is the outcome of k-th layer, $\boldsymbol{\omega}^k \in \mathbb{R}^{n_k \times n_{k-1}}$, $\boldsymbol{b}^k \in \mathbb{R}^{n_k}$, and n_k is the number of neurons in k^{th} layer. Here, $\boldsymbol{\omega}$ and $\boldsymbol{b}$ denote the weight and bias, the model parameters to be optimized. Moreover, σ is a non-linear mapping known as an activation function. This process is widely known as feed-forward propagation. We define a loss function ($\mathcal{L}$) for network output, which quantifies the disparity between the network's predictions and the desired outcomes. The neural network is trained using backpropagation [32], a technique wherein optimization algorithms, such as gradient descent, are used to minimize the loss function and fine-tune the model parameters $(\boldsymbol{\omega}, \boldsymbol{b})$. Trained parameters capture the underlying structure of the problem under consideration. A well-known class of neural networks is physics-informed neural networks (PINNs), which incorporate the underlying physics of the system by using a governing set of equations along with initial and boundary conditions into the loss function. It measures how much the network's predicted solution violates the differential equation at several collocation points.

Consider a PINN model, $\hat{u}(\boldsymbol{x}, t; \boldsymbol{\theta})$ that predicts the solution of a differential equation in the spatiotemporal domain, $\boldsymbol{x} \in \Omega$ and $t \in (0, T]$. Here $\boldsymbol{\theta}$ are trainable parameters of the network, which describe the non-linear relationship $(\boldsymbol{x}, t) \rightarrow \hat{u}$ according to 1. A general form of a partial differential equation (PDE) can be ~~considered as~~

$$\begin{cases} \mathscr{L}_{\boldsymbol{x},t}[u(\boldsymbol{x},t)] = f(\boldsymbol{x},t), & \boldsymbol{x} \in \Omega,\ t \in (0,T], \\ \mathscr{B}[u(\boldsymbol{x},t)] = g(\boldsymbol{x},t), & \boldsymbol{x} \in \partial\Omega,\ t \in (0,T], \\ \mathscr{I}[u(\boldsymbol{x},0)] = h(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \end{cases} \tag{2}$$

where $(\boldsymbol{x}, t)$ denotes spatiotemporal ~~points~~, $\mathscr{L}_{\boldsymbol{x},t}[\cdot]$ represents differential operator, $\mathscr{B}[\cdot]$ and $\mathscr{I}[\cdot]$ correspond to boundary and initial conditions respectively. The loss function for PINN can be defined as $\mathcal{L} = \mathcal{L}_{\text{res}} + \mathcal{L}_{\text{ic}} + \mathcal{L}_{\text{bc}}$. Here,

$$\begin{aligned} \mathcal{L}_{\text{res}} &= \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\mathscr{L}[\hat{u}_i(\boldsymbol{x}, t; \boldsymbol{\theta})] - f_i(\boldsymbol{x}, t) \right)^2, \quad \boldsymbol{x} \in \Omega,\ t \in (0, T], \\ \mathcal{L}_{\text{bc}} &= \frac{1}{N_{bc}} \sum_{i=1}^{N_{bc}} \left(\mathscr{B}[\hat{u}_i(\boldsymbol{x}, t; \boldsymbol{\theta})] - g_i(\boldsymbol{x}, t) \right)^2, \quad \partial\boldsymbol{x} \in \Omega,\ t \in (0, T], \\ \mathcal{L}_{\text{ic}} &= \frac{1}{N_{ic}} \sum_{i=1}^{N_{ic}} \left(\mathscr{I}[\hat{u}_i(\boldsymbol{x}, 0; \boldsymbol{\theta})] - h_i(\boldsymbol{x}) \right)^2, \quad \boldsymbol{x} \in \Omega, \end{aligned} \tag{3}$$

where $\mathcal{L}_{\text{res}}$ denotes the residual mean squared error, $\mathcal{L}_{\text{bc}}$ and $\mathcal{L}_{\text{ic}}$ denote mean squared errors for boundary and initial conditions, respectively. During optimization, the derivative of the loss function with respect to the model parameters and derivatives of the network's output with respect to input parameters, exhibited in the loss function, are required. These derivatives are effectively computed using "automatic differentiation" [33]. Thus, the objective of PINNs is to identify the optimal parameters that minimize the specified loss function, which encapsulates all physics-related information.

2.2. Neural tangent kernel for PINNs

The Neural Tangent Kernel (NTK) theory offers a rigorous ematical formulation for analyzing the dynamics of neural network functions, $f(\boldsymbol{x};\boldsymbol{\theta})$, where $\boldsymbol{x}$ is ; and $\boldsymbol{\theta}$ are the network's parameters. NTK was first proposed by Jacot et al. [14]. They introduced a kernel regression framework that allows the study of neural network training in the function space rather than in the parameter space. Since the functional cost (e.g., mean square loss) is convex in contrast to the parameter cost, this makes analysis more tractable in function space. Jacot et al. demonstrated that, under training with an infinitesimally small learning rate, the network function $f(\boldsymbol{x};\boldsymbol{\theta})$ evolves according to the kernel gradient descent with respect to the NTK. Moreover, in the infinite-width limit, the NTK converges to a deterministic limiting kernel that remains constant throughout training.

Subsequent work by Cao et al. [15] provided a spectral analysis of the NTK, showing that smaller NTK eigenvalues lead to slower convergence of high-frequency components in the target function. This phenomenon, known as the "spectral bias" of neural networks, states that neural networks tend to learn low-frequency features more efficiently than high-frequency ones. Building on this foundation, Wang et al. [16] extended the NTK theory to the convergence rate of training error in PINNs. ir analysis can be summarized as follows. Consider a PINN $\hat{u}(\boldsymbol{x};\boldsymbol{\theta})$ that is an approximate solution e following PDE.

$$\begin{cases} \mathscr{L}[u(\boldsymbol{x})] = f(\boldsymbol{x}), & \boldsymbol{x} \in \Omega, \\ \mathscr{B}[u(\boldsymbol{x})] = g(\boldsymbol{x}), & \boldsymbol{x} \in \partial\Omega. \end{cases}$$

Here, $\mathscr{L}$ and $\mathscr{B}$ represent differential operators and boundary conditions, respectively. For given data points $\{\boldsymbol{x}_r^i, f(\boldsymbol{x}_r^i)\}_{i=1}^{N_r}$ and $\{\boldsymbol{x}_b^i, g(\boldsymbol{x}_b^i)\}_{i=1}^{N_b}$ lynamics of $\hat{u}$ and $\mathscr{L}[\hat{u}]$ follows.

$$\begin{aligned} \begin{bmatrix} \frac{d\hat{u}(\boldsymbol{x}_b;\boldsymbol{\theta}(t))}{dt} \\ \frac{d\mathscr{L}[\hat{u}](\boldsymbol{x}_r;\boldsymbol{\theta}(t))}{dt} \end{bmatrix} &= -\begin{bmatrix} \boldsymbol{K}_{uu}(t) & \\ \boldsymbol{K}_{ru}(t) & (t) \end{bmatrix} \begin{bmatrix} \hat{u}(\boldsymbol{x}_b;\boldsymbol{\theta}(t)) - g(\boldsymbol{x}_b) \\ \mathscr{L}[\hat{u}](\boldsymbol{x}_r;\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix} \\ &= -\boldsymbol{K}(t) \begin{bmatrix} \hat{u}(\boldsymbol{x}_b;\boldsymbol{\theta}(t)) - g(\mathbf{x}_b) \\ \mathscr{L}[\hat{u}](\boldsymbol{x}_r;\boldsymbol{\theta}(t)) - f(\boldsymbol{x}_r) \end{bmatrix}, \end{aligned} \tag{4}$$

where t is training step and $K(t)$ is NTK for PINN, whose $\{i,j\}$-th element is obtained by:

$$\begin{aligned} (\boldsymbol{K}_{uu})_{ij}(t) &= \left\langle \frac{\partial \hat{u}(\boldsymbol{x}_b^i;\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial \hat{u}(\boldsymbol{x}_b^j;\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}} \right\rangle, \\ (\boldsymbol{K}_{ur})_{ij}(t) &= \left\langle \frac{\partial \hat{u}(\boldsymbol{x}_b^i;\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial \mathscr{L}[\hat{u}](\boldsymbol{x}_r^j;\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}} \right\rangle, \\ (\boldsymbol{K}_{rr})_{ij}(t) &= \left\langle \frac{\partial \mathscr{L}[\hat{u}](\boldsymbol{x}_r^i;\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}}, \frac{\partial \mathscr{L}[\hat{u}](\boldsymbol{x}_r^j;\boldsymbol{\theta}(t))}{\partial \boldsymbol{\theta}} \right\rangle, \end{aligned}$$

where $\langle\cdot,\cdot\rangle$ represents the inner product in all parameters. Similarly to NTK theory for neural networks, Wang et al. [16] establish that under certain assumptions, NTK for a PINN also remains constant during training; therefore, we approximate $\boldsymbol{K}(t) \approx \boldsymbol{K}(0)$. This assumption leads to a solution of 4 as

$$\begin{aligned} &\begin{bmatrix} \hat{u}(\boldsymbol{x}_b;\boldsymbol{\theta}(t)) \\ \mathscr{L}[\hat{u}](\boldsymbol{x}_r;\boldsymbol{\theta}(t)) \end{bmatrix} = \left(\boldsymbol{I} - e^{-\boldsymbol{K}(0)t}\right) \begin{bmatrix} g(\boldsymbol{x}_b) \\ \\ f(\boldsymbol{x}_r) \end{bmatrix}, \\ \text{equivalently,} \quad &\begin{bmatrix} \hat{u}(\boldsymbol{x}_b;\boldsymbol{\theta}(t)) \\ \mathscr{L}[\hat{u}](\boldsymbol{x}_r;\boldsymbol{\theta}(t)) \end{bmatrix} - \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix} = -e^{-\boldsymbol{K}(0)t} \begin{bmatrix} g(\boldsymbol{x}_b) \\ f(\boldsymbol{x}_r) \end{bmatrix}. \end{aligned} \tag{5}$$

This is how the training error evolves with iterations. It can be further simplified using the semi-positive definiteness of NTK. We decompose $\boldsymbol{K}(0)$ as $\boldsymbol{K}(0) = \boldsymbol{Q}^T \boldsymbol{\Lambda} \boldsymbol{Q}$, where $\boldsymbol{Q}$ is an orthogonal matrix, and $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues $\lambda_i \geq 0$ as diagonal elements. With this, we rewrite 5 as

$$\begin{bmatrix} \hat{u}(x_b; \boldsymbol{\theta}(t)) \\ \mathscr{L}[\hat{u}](x_r; \boldsymbol{\theta}(t)) \end{bmatrix} - \begin{bmatrix} g(x_b) \\ f(x_r) \end{bmatrix} = -\boldsymbol{Q}^T e^{-\boldsymbol{\Lambda} t} \boldsymbol{Q} \begin{bmatrix} g(x_b) \\ f(x_r) \end{bmatrix}.$$

This provides an estimate of the convergence of a PINN, which is governed by the magnitude of the eigenvalues of the NTK. Specifically, the larger the eigenvalue λ_i, the faster the i-th error component decays. Detailed mathematics of the above analysis can be found in [16]. Later in the results section, we use these findings to establish a much faster convergence of WPINN over PINN for stiff problems.

2.3. Wavelet-based physics-informed neural networks (W-PINNs)

The W-PINN formulation begins by defining a family of wavelet basis functions, constructed as scaled and translated versions of a mother wavelet, ~~mathematically,~~ which can be written as

$$\Psi_{j,k}(x) = \sqrt{2^j}\psi(2^j x - k), \quad j, k \in \mathbb{Z}.$$

Here, j denotes the scale (or resolution level) and k is the translation parameter. For a compact domain $[a, b]$, the number of wavelet basis functions is determined by a predefined set of resolution levels $J = \{J_1, J_2, \ldots, J_N\}$. For a fixed $j \in J$, the translation indices k span the interval $\left[\lfloor (a-\gamma)\cdot 2^j \rfloor, \lceil (b+\gamma)\cdot 2^j \rceil\right]$, where γ is a translation hyperparameter (of order of domain length), $\lfloor\cdot\rfloor$ and $\lceil\cdot\rceil$ denote floor and ceiling functions, respectively. This ensures sufficient coverage of the domain at each resolution level.

After constructing a family of wavelets, we approximate the solution of the differential equation as follows:

$$\hat{u}(x) = \sum_{j=J_1}^{J_N} \sum_{k=\lfloor (a-\gamma)\cdot 2^j \rfloor}^{\lceil (b+\gamma)\cdot 2^j \rceil} c_{j,k} \Psi_{j,k}(x) + \mathcal{B}, \tag{6}$$

where $c_{j,k}$ are trainable coefficients and $\mathcal{B}$ is a trainable bias to allow the representation of nonzero-mean solutions, given that the chosen wavelets are symmetric and inherently zero-mean. Extension of this formulation for higher-dimensional problems is straightforward; for instance, here is a formulation for a 2D problem:

$$\begin{aligned} \Psi_{j_1,j_2,k_1,k_2}(x,y) &= \sqrt{2^{j_1}\cdot 2^{j_2}}\psi_X(2^{j_1}x - k_1)\,\psi_Y(2^{j_2}y - k_2), \\ \hat{u}(x,y) &= \sum_{j_1=J_{11}}^{J_{1N_1}} \sum_{j_2=J_{21}}^{J_{2N_2}} \sum_{k_1=\lfloor (a_1-\gamma)\cdot 2^{j_1} \rfloor}^{\lceil (b_1+\gamma)\cdot 2^{j_1} \rceil} \sum_{k_2=\lfloor (a_2-\gamma)\cdot 2^{j_2} \rfloor}^{\lceil (b_2+\gamma)\cdot 2^{j_2} \rceil} c_{j_1,j_2,k_1,k_2} \Psi_{j_1,j_2,k_1,k_2}(x,y) + \mathcal{B}. \end{aligned} \tag{7}$$

This hierarchical construction of wavelet families at different resolutions and representing the solution in wavelet space enables capturing features at multiple scales, from broad, global behavior to fine, local details. In this study, we consider two mother wavelets, namely, the Gaussian and the Mexican Hat, which are single and double derivatives of the Gaussian function, respectively. The mathematical expressions for the Mexican hat ($\psi^M(x)$), Gaussian ($\psi^G(x)$) can be written as

$$\psi^M(x) = (1 - x^2)e^{-\frac{x^2}{2}}, \quad \psi^G(x) = -xe^{-\frac{x^2}{2}}.$$

The W-PINN architecture comprises three parts:

1. **Pointwise feature mapping**: A shallow fully-connected network that maps spatial-temporal coordinates to a scalar latent feature.

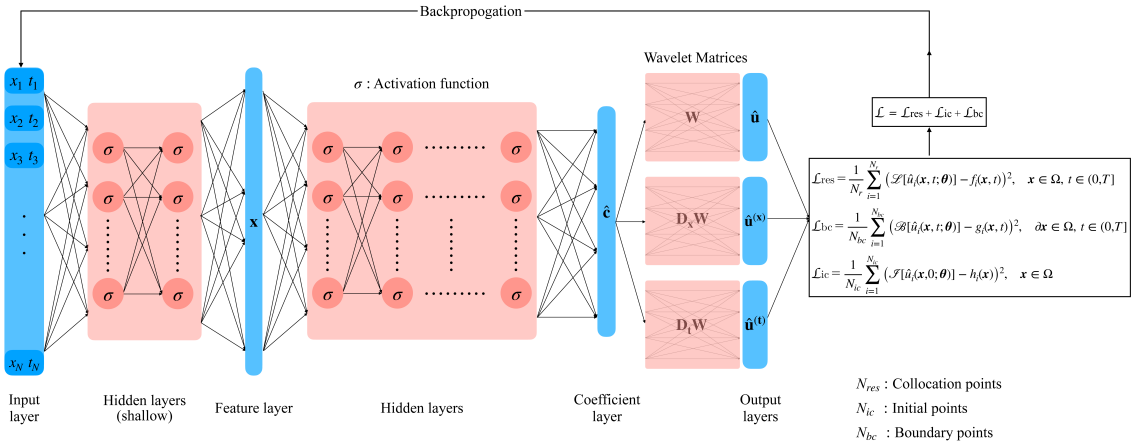


Figure 1: A W-PINN tecture.

2. **Global coefficient predictor**: A deep fully-connected network that predicts wavelet coefficients $c_{j,k}$, that is learning solution in wavelet space.

3. **Non-trainable inverse mapping**: A fixed layer that transforms the solution back to physical space using predicted coefficients and precomputed wavelet basis matrices.

The architecture of W-PINN for a two-dimensional problem is illustrated in Figure 1. In general, for a d-dimensional problem, the network architecture comprises an input layer with d neurons corresponding to the spatial-temporal dimensions. These inputs are processed through a shallow network that maps each collocation point to a corresponding entry in the feature layer. For one-dimensional problems, this mapping can be simplified by directly using the input layer as the feature layer. Now, the feature layer passes through a fully connected network to get coefficients. To further enhance model performance, these coefficients can be separately fine-tuned through hyperparameter optimization. The final solution and its derivatives are then computed by combining these optimized coefficients with the pre-calculated wavelet matrices. A trainable bias is also added to the network to incorporate $\mathcal{B}$.

For a given set of collocation points $\mathbf{x} = (x_0, x_1, \cdots, x_N)$ and resolution set $J = [J_1, J_2, \cdots, J_n]$, using the domain $[0, 1]$ and $\gamma = 0$, the wavel trices are constructed as follows:

$$\mathbf{W} = \begin{pmatrix} \psi_{J_1,0}(x_0) & \cdots & \psi_{J_1,2^{J_1}}(x_0) & \cdots & \psi_{J_n,2^{J_n}}(x_0) \\ \psi_{J_1,0}(x_1) & \cdots & \psi_{J_1,2^{J_1}}(x_1) & \cdots & \psi_{J_n,2^{J_n}}(x_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \psi_{J_1,0}(x_N) & \cdots & \psi_{J_1,2^{J_1}}(x_N) & \cdots & \psi_{J_n,2^{J_n}}(x_N) \end{pmatrix}, \mathbf{D_1W} = \begin{pmatrix} \psi'_{J_1,0}(x_0) & \cdots & \psi'_{J_1,2^{J_1}}(x_0) & \cdots & \psi'_{J_n,2^{J_n}}(x_0) \\ \psi'_{J_1,0}(x_1) & \cdots & \psi'_{J_1,2^{J_1}}(x_1) & \cdots & \psi'_{J_n,2^{J_n}}(x_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \psi'_{J_1,0}(x_N) & \cdots & \psi'_{J_1,2^{J_1}}(x_N) & \cdots & \psi'_{J_n,2^{J_n}}(x_N) \end{pmatrix},$$

where $\psi'(x)$ is single derivate of $\psi(x)$ and similarly $\mathbf{D_2W}$ is constructed by double derivative of $\psi(x)$. Once the approximate solution and its derivatives are obtained, the loss function is computed as 3.

This wavelet-based formulation eliminates the need for automatic differentiation (AD) in computing PDE residuals, which not only makes training faster by reducing the complexity of the computational graph but also avoids potential numerical instabilities that commonly arise in AD during training, particularly in problems exhibiting high gradients or singularities. The core principle behind W-PINN's efficacy is that abrupt and multiscale features in the physical space become smoother in the wavelet domain, thus significantly simplifying the learning task for the neural network.

3. Results and discussions

In this section, we present the results obtained using the proposed method over a handful of examples and compare them with the performance of conventional PINNs and state-of-the-art methods. For a fair comparison, all network parameters are kept the same throughout [redacted] are tabulated along with the examples. We utilize the Adam optimizer [34], which combines two stoch[redacted] gradient descent algorithms: gradient descent with momentum and root mean squared propagation (RMSProp). RMSprop, also known as adaptive gradient descent, adapts the learning rate with iterations. The Adam optimizer is memory efficient and is well-suited for problems with high-dimensional parameter spaces.

The trainable parameters of networks are initialized using Xavier's technique or Glorot initialization [35], which, in contrast to random initialization, initializes parameters in a suitable range to achieve faster convergence during training. During backpropagation, it maintains gradients within a reasonable range (not too low or high), thereby preventing the algorithm from getting stuck in poor local minima. Besides this, it can also reduce the need for extensive hyperparameter tuning, particularly the learning rate. We sampled the training set by Sobol sequencing throughout the domain, as we need efficient exploration of the entire input space for a single training set. For evaluation purposes, we employ the relative L_2 metric over uniform samples across the domain.

$$\text{Relative } L_2 \text{ error} = \sqrt{\frac{\sum_{i=1}^{M}(u_i - \hat{u}_i)^2}{\sum_{i=1}^{M} u_i^2}}, \tag{8}$$

where M represents the total number of testing samples, u_i is the exact solution and $\hat{u}_i$ is the predicted one for the i^{th} sample. ~~In the case of problems with unknown solutions~~, we consider the exact solution as ~~a~~ numerical solution obtained from ~~the~~ in-house solver.

For implementation purposes, PyTorch 2.4.1 with CUDA 12.1 is used. Training is done on the NVIDIA RTX A6000, which has 48GB of GPU memory. As the performance of a neural network is highly sensitive to the initialization of its parameters, we perform each experiment five times and report the relative L_2-error and average training time based on these five random runs. The source code of the proposed W-PINN method in this work is available on GitHub at `https://github.com/himanshup21/W-PINN.git`.

Example 1. *Consider the following one-dimensional linear advection-diffusion equations [36]:*

$$\begin{cases} \epsilon \dfrac{d^2u}{dx^2} + (1+\epsilon)\dfrac{du}{dx} + u = 0, \ \ x \in (0,1), \\ u(0) = 0, \ \ u(1) = 1. \end{cases} \tag{9}$$

Table 1: Parameters used for solving Example 1.

Parameters	Value
Translation hyperparameter γ	1.0
Set of resolutions (J_M/J_G)	[0,8]/[0,10]
Number of hidden layers	6
Neurons per layer	100
Number of collocation points	10^3
Maximum number of iterations	5×10^4

Table 2: Average training time (in seconds) of PINN and W-PINN with network depth for Example 1.

Depth	PINN	W-PINN	Speed-up
2	20.6	7.3	2.8x
4	27.2	6.2	4.4x
8	47.6	8.3	5.7x
16	82.6	12.4	6.6x
32	153.5	22.8	6.7x
64	288.4	38.7	7.4x

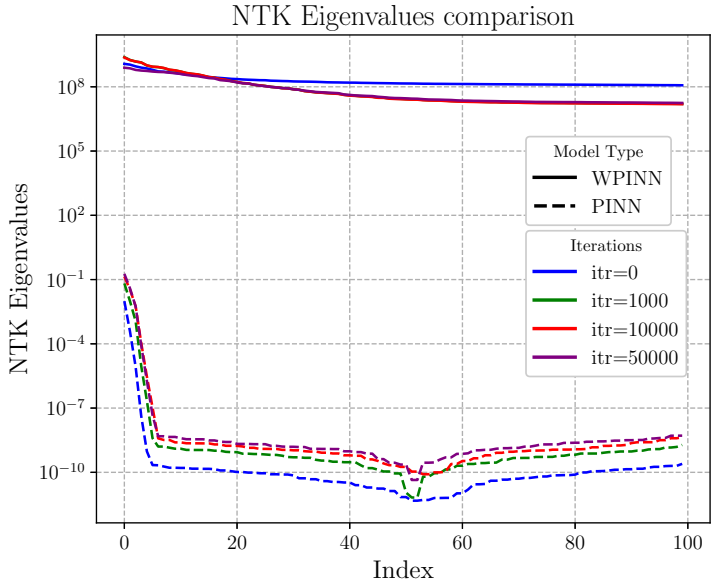


Figure 2: Comparison of NTK eigenvalues of PINN(dashed line) and W-PINN (solid line) for Example 1 with $\epsilon = 2^{-7}$ at various iterations. Eigenvalues are sorted in descending order (y-axis (log scale)) and plotted against the respective index (x-axis).

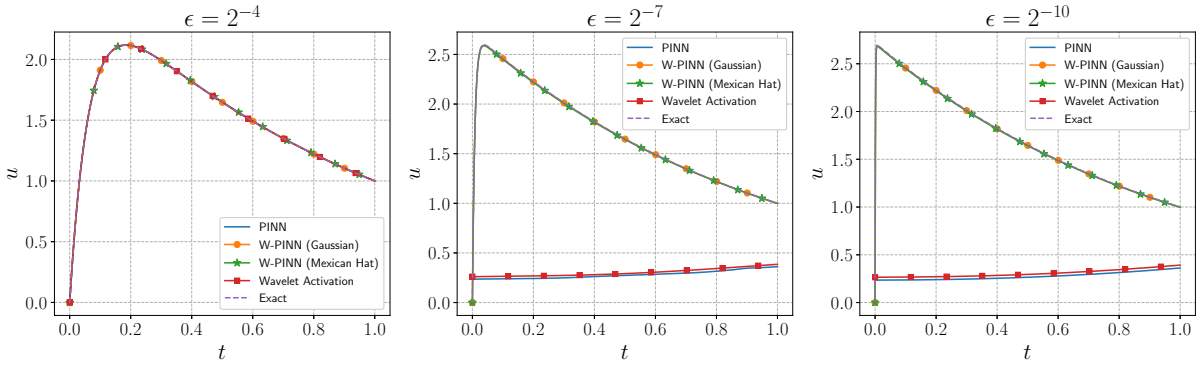


Figure 3: Comparison of solutions obtained by PINN, with wavelet activation, and W-PINN methods for Example 1. From left to right: with $\epsilon = 2^{-4}$, $\epsilon = 2^{-7}$, and $\epsilon = 2^{-10}$.

Here, $0 < \epsilon \ll 1$. *The exact solution to the considered problem is*

$$u(x) = \frac{\exp(-x) - \exp(-\frac{x}{\epsilon})}{\exp(-1) - \exp(-\frac{1}{\epsilon})}.$$

Such models, called Friedrich's boundary layer models, are often used to show how difficult it is to model viscous flow boundary layers [18]. It is impossible for PINNs to capture the singularity that happens when $\epsilon \to 0$.

The parameters of the W-PINN for the Example 1 are listed in Table 1. Table 2 compares the training time of automatic differentiation based PINN and W-PINN. The experiments were conducted with $\epsilon = 2^{-4}$ and each test was executed for 10^4 iterations. The results demonstrate that conventional PINN exhibits a steeper growth in training time with an increase in network depth, requiring up to 7.4 times longer than W-PINN at depth 64. This substantial difference in computational overhead can be

Table 3: Performance comparison of various methods for solving mple 1.

ϵ	Method	L_2-error	Average Training Time
$\epsilon = 2^{-4}$	PINN	7.9e-05	1m 58s
	Wavelet Activation [30]	3.9e-05	52s
	W-PINN (Gaussian)	2.5e-05	23s
	W-PINN (Mexican hat)	8.1e-05	48s
$\epsilon = 2^{-7}$	PINN	0.83	-
	Wavelet Activation [30]	0.85	-
	W-PINN (Gaussian)	5.3e-04	18s
	W-PINN (Mexican hat)	9.2e-04	23s
$\epsilon = 2^{-10}$	PINN	0.84	-
	Wavelet Activation [30]	0.85	-
	W-PINN (Gaussian)	3.1e-03	38s
	W-PINN (Mexican hat)	4.2e-03	42s

attributed to W-PINN's elimination of automatic differentiation calculations, which become increasingly expensive in conventional PINNs as network depth increases. Moreover, Figure 2 presents a comparison of the NTK eigenvalue spectra for W-PINN and standard PINN across multiple training iterations for $\epsilon = 2^{-7}$. Here, the NTK eigenvalues are sorted in descending order and plotted against their respective indices. The observed differences in eigenvalues span several orders of magnitude, highlighting the superior conditioning of the W-PINN NTK. This suggests that W-PINNs can efficiently converge to accurate solutions for such challenging problems, including singularly perturbed, multiscale, and stiff systems. In contrast, the extremely small eigenvalues associated with the NTK of standard PINNs indicate poor convergence behavior, making them ineffective for such a class of problems. The comparison of various methods has been presented in Table 3, which provides the relative L_2-error and average training time of the W-PINN, conventional PINN and PINN with gaussian-wavelet activation function [30]. This table shows that the W-PINN approximates the solution with a much smaller L_2-error for lower values of ϵ. Similar observations are drawn from Figure 3, which demonstrates that when $\epsilon = 2^{-4}$, there is no such sharp singularity near the origin, and all methods approximate the solution well. However, as ϵ decreases, the exact solution of the problem has a strong singularity near the origin. Then, both PINN and PINN with a wavelet activation function are unable to capture the behavior of the solution accurately. In contrast, W-PINN effectively resolves the singularity, with the predicted solution closely following the exact solution.

Example 2. *Consider the following highly singularly perturbed non-linear problem [37]:*

$$\begin{cases} \epsilon \dfrac{d^2u}{dt^2} + (3+t)\dfrac{du}{dt} + u^2 - \sin(u) = f, \ t \in (0,1], \\ u(0) = 1, \ u'(0) = 1/\epsilon, \end{cases} \tag{10}$$

where f is chosen such that $u(t) = 2 - \exp(-t/\epsilon) + t^2$ is the exact solution.

This example corresponds to a highly singularly perturbed non-linear equation with a known solution. For Example 2, the network parameters are given in Table 4. We compare the performance of W-PINN with the traditional PINN, and the PINN proposed in [30] (using a Gaussian wavelet as an activation function) for various ϵ. The relative L_2-error and average training time for these are tabulated in Table

5, which illustrates that the W-PINN outperforms other methods as it takes much less training time and provides better approximation, especially for lower values of ϵ. Figure 4 demonstrates similar observations, for $\epsilon = 2^{-10}$, both conventional PINN and PINN with the Gaussian wavelet as activation function [30] are unable to detect the singularity satisfactorily, whereas W-PINN effectively handles the singularity.

In addition, we investigate the relationship between the hyperparameters of W-PINN and its performance for this example. Figure 5 illustrates the relationship between the resolution level ($[0, J]$) and the relative L_2-error. Here, we employ a network with 8 hidden layers, each containing 200 neurons, and fix $10,000$ collocation points. It shows that W-PINN gives optimal results for a specific resolution range (not too high nor too low). Lower resolution is incapable of capturing sharp singularities, and a large J makes the loss function too complex to be optimized. The optimal range of the resolution is also dependent on the wavelet family. In this specific case, the W-PINN with Gaussian performed well for J, which is between 8 and 14, while this range is narrower for the Mexican hat. However, W-PINN performance also depends on the kind of wavelet used for a particular problem.

Figure 6 presents the effect of the number of collocation points on performance. We consider the identical network as previously used, with the Gaussian resolutions ($J_G = [0, 12]$) and the Mexican hat resolutions ($J_M = [0, 11]$). It is evident that the relative L_2-error decreases as the number of collocation points increases. However, the error decrease is not significant beyond a certain threshold. After a certain

Table 4: Parameters used for solving Example 2.

Parameters	Value
Translation hyperparameter γ	1.0
Set of resolutions (J_M/J_G)	[0,9]/[0,10]
Number of hidden layers	6
Neurons per layer	100
Number of collocation points	10^4
Maximum number of iterations	10^5

Table 5: Performance comparison of various methods solving Example 2.

ϵ	Method	L_2-error	Average Training Time
$\epsilon = 2^{-4}$	PINN	1.2e-04	31s
	Wavelet Activation [30]	2.1e-04	58s
	W-PINN (Gaussian)	2.3e-04	48s
	W-PINN (Mexican hat)	1.8e-04	51s
$\epsilon = 2^{-7}$	PINN	8.5e-02	3m 33s
	Wavelet Activation [30]	2.2e-02	3m 14s
	W-PINN (Gaussian)	8.3e-05	2m 21s
	W-PINN (Mexican hat)	3.7e-04	2m 24s
$\epsilon = 2^{-10}$	PINN	1.4	-
	Wavelet Activation [30]	0.34	-
	W-PINN (Gaussian)	6.6e-05	2m 27s
	W-PINN (Mexican hat)	4.5e-04	2m 48s

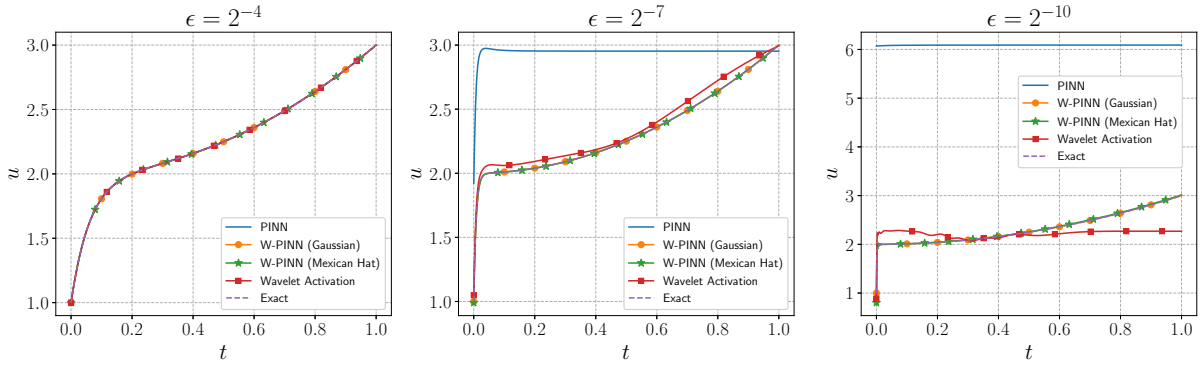


Figure 4: Comparison of solutions obtained by PINN, PINN with wavelet activation, and W-PINN methods for Example 2. From left to right: with $\epsilon = 2^{-4}$, $\epsilon = 2^{-7}$, and $\epsilon = 2^{-10}$.

number of collocation points, additional points do not come with any new information about the behavior of the problem under consideration.

Figure 7 presents experiments on the trainable part of W-PINN. We employ $10,000$ collocation points for both Gaussian and Mexican hat, with resolutions of $[0, 12]$ and $[0, 11]$, respectively. ~~It implies~~ that a shallow network is incapable of learning the behaviors of the problem, regardless of the number of neurons per layer. However, a network with a significant number of layers and sufficient neurons per layer performs adequately. ~~It is~~ redundant to employ a large number of layers and neurons, which increases computational cost rather than accuracy.

Example 3. *Consider the following singularly perturbed non-linear problem with Neumann boundary conditions*

$$\begin{cases} -\epsilon \dfrac{d^2u}{dt^2} + u^5 + 3u - 1 = 0, \ \ t \in [0, 1], \\ u'(0) = \sin(0.5), \ \ u'(1) = \exp(-0.7). \end{cases} \tag{11}$$

This example corresponds to a singularly perturbed non-linear problem with Neumann boundary conditions. This problem possesses a boundary layer singularity at both ends. The exact solution to this problem is unknown, so we obtained a numerical solution using Scipy solver and treated it as an exact solution.

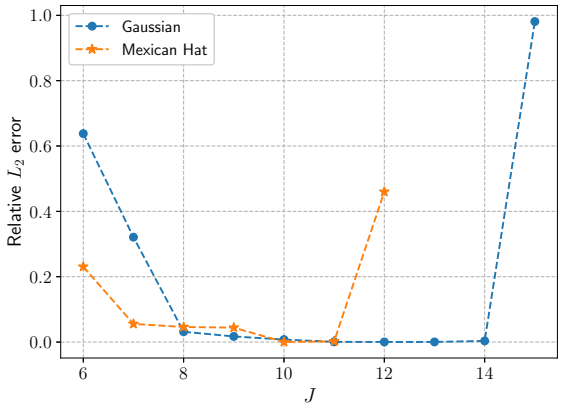


Figure 5: A plot for relative L_2-error with different wavelet resolution levels ($[0, J]$) for Gaussian (blue) and Mexican-Hat (orange) wavelets.

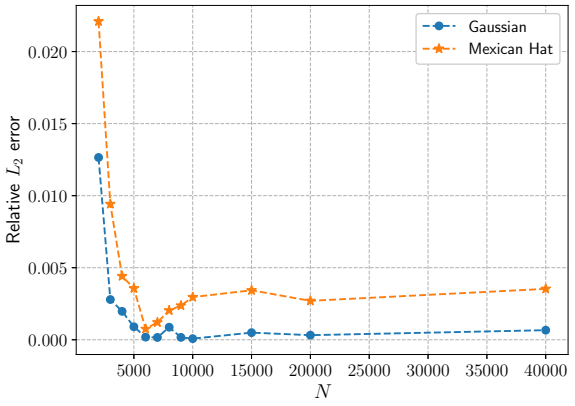
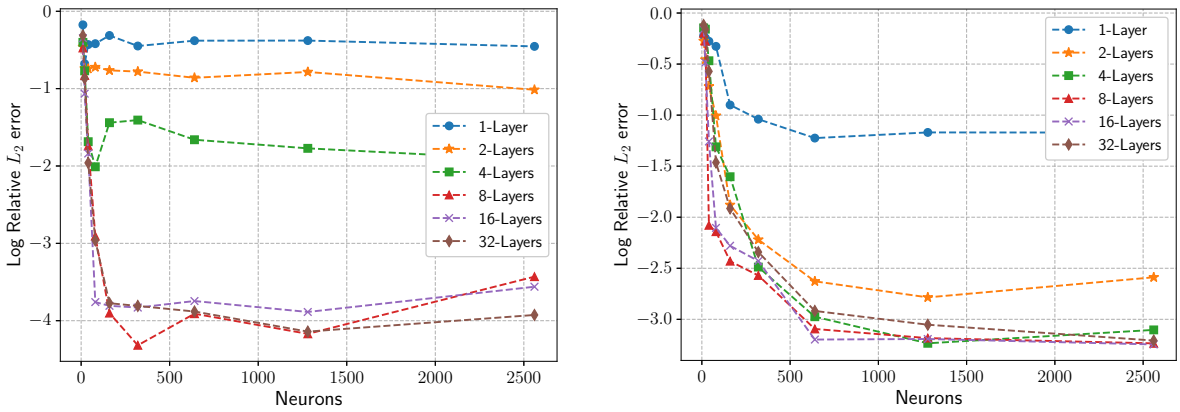


Figure 6: A plot for relative L_2-error variation with number of collocation points (N) for Gaussian (blue) and Mexican-Hat (orange) wavelets.



(a) Gaussian wavelet (b) Mexican wavelet

Figure 7: Impact of number of hidden layers and number of neurons per layer on performance of W-PINN using Gaussian wavelet(left) and Mexican-Hat wavelet(right).

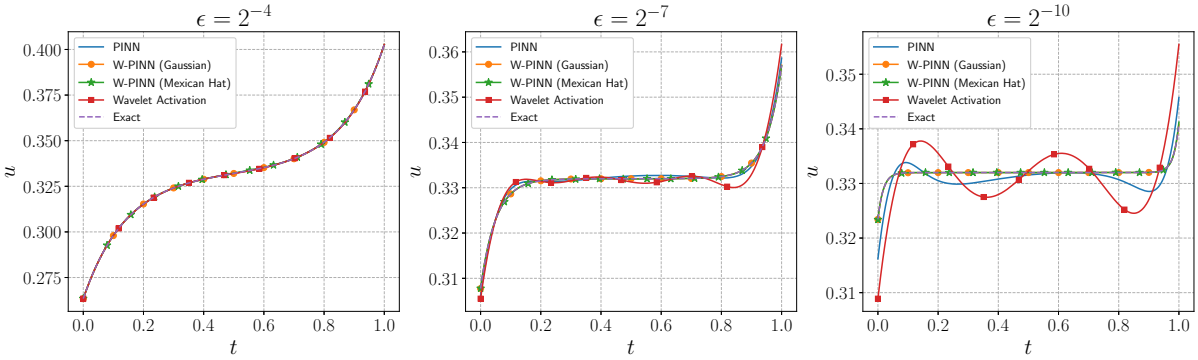


Figure 8: Comparison of solutions obtained by PINN, PINN with wavelet activation, and W-PINN methods for Example 3. From left to right: with $\epsilon = 2^{-4}$, $\epsilon = 2^{-7}$, and $\epsilon = 2^{-10}$.

Table 6 ~~represents~~ the parameters of the network for the Example 3. The results have been displayed in Table 7 and Figure 8. Specifically, the Table 7 compares the performance of W-PINN, traditional PINN, and the PINN with a Gaussian wavelet as activation function [30] for various ϵ in terms of relative L_2-error and average training time. This table shows that W-PINN takes much less training time and approximates the solution to a better extent. Figure 8 demonstrates that as the singularity becomes

Table 6: Parameters used for solving Example 3.

Parameters	Value
Translation parameter γ	1.0
Set of resolutions (J_M/J_G)	[0,8]/[0,9]
Number of hidden layers	8
Neurons per layer	200
Number of collocation points	10^4
Maximum number of iterations	10^4

Table 7: Performance comparison of various methods for solving Example 3

ϵ	Method	L_2-error	Average Training Time
$\epsilon = 2^{-4}$	PINN	8.6e-05	2m 26s
	Wavelet Activation [30]	3.2e-04	4m 49s
	W-PINN (Gaussian)	4.7e-05	28s
	W-PINN (Mexican hat)	1.4e-04	34s
$\epsilon = 2^{-7}$	PINN	1.8e-03	4m 41s
	Wavelet Activation [30]	4.1e-03	6m 13s
	W-PINN (Gaussian)	8.9e-06	14s
	W-PINN (Mexican hat)	8.3e-05	21s
$\epsilon = 2^{-10}$	PINN	5.3e-03	7m 14s
	Wavelet Activation [30]	8.4e-03	12m 28s
	W-PINN (Gaussian)	6.5e-06	13s
	W-PINN (Mexican hat)	9.9e-05	28s

more prominent, W-PINN effectively addresses the singularity to a greater degree at both ends. In contrast, other PINN methods couldn't resolve the singularity and introduced unnatural oscillations over the domain.

Example 4. *FitzHugh-Nagumo (FHN) model [38]:*

The FitzHugh-Nagumo (FHN) model is a simplified version of the Hodgkin-Huxley model, which simulates the dynamics of spiking neurons. It captures how a neuron generates electrical signals (spikes) in response to stimulation (an external current). The mathematical form of the FHN dynamical model is defined as follows:

$$\begin{cases} \dfrac{dv}{dt} - v + \dfrac{v^3}{3} + w - RI = 0, \\ \tau \dfrac{dw}{dt} - v + bw + a = 0, \end{cases} \tag{12}$$

where v denotes the neuron's membrane voltage, w is the recovery variable, reflecting the activation state of ion channels, I represents the external current applied to the neuron, and R is the resistance across the neuron. In addition, a and b are scaling parameters, and τ denotes the time constant for the recovery variable (w), which acts as a singularly perturbed parameter for this model. In particular, for $a = b = 0$, the FHN model describes the Van der Pol oscillator, which describes self-sustaining oscillations in many systems, such as heartbeats, economies, and electronic circuits.

For the numerical computations, we set: $a = 1,\ b = 1,\ I = 0.1,\ R = 1$, and $\tau = 2^{-10}$, $v(0) = 0.5$ and $w(0) = 0.1$. The dynamics of the system are computed till $t = 1$. Table 8 provides the parameters for this Example 4. From Table 9, it is evident that W-PINN gets a more accurate prediction and takes significantly less training time. Figure 9 compares the W-PINN approximations of v and w with PINN and PINN with wavelet activation. The figure illustrates that a strong initial layer is present in the solution profile of w, which both PINN and W-PINN fail to capture and generate spurious oscillations near the singularity, whereas W-PINN effectively resolves the singularity.

Example 5. *A heat conduction problem with large gradients [39]:*

The heat conduction problem investigates temperature flow when an intense heat source suddenly appears.

Table 8: Parameters used for solving Example 4.

Parameters	Value
Translation parameter γ	1.0
Set of resolutions (J_M/J_G)	[0,10]/[0,11]
Number of hidden layers	10
Neurons per layer	200
Number of collocation points	10^4
Number of iterations	2×10^4

Table 9: Performance comparison of various methods for solving Example 4 with $\tau = 0.15$.

Methods	L_2-error (v/w)	Average Training Time
PINN	$5.9e-04$ $8.1e-02$	6m 38s
Wavelet Activation	$2.1e-03$ 0.11	8m 22s
W-PINN (Gaussian)	$9.2e-05$ $5.7e-04$	1m 06s
W-PINN (Mexican hat)	$4.4e-03$ $8.2e-03$	1m 36s

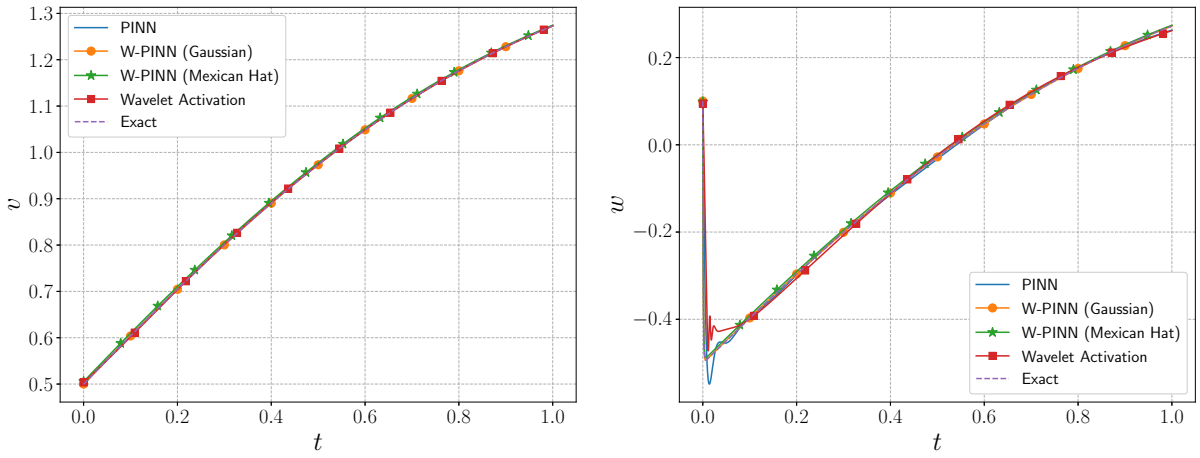


Figure 9: Comparison of solution of FHN model 4 obtained by PINN, PINN with wavelet activation, and W-PINN methods with $\tau = 2^{-10}$.

It represents a practical challenge often seen in fusion applications, where researchers need to understand and model rapid heat transfer.

The following mathematical model includes a small positive constant ϵ that creates steep temperature

Table 10: Parameters used for solving Example 5.

Parameters	Value
Translation hyperparameter γ	0.2
Set of resolutions (J_x, J_t)	[-6,5], [-6,5]
Number of hidden layers	6
Neurons per layer	50
Number of collocation points	10^4
Number of boundary points	10^3
Number of initial points	500

gradients, making it a useful test case for our developed method:

$$\begin{cases} \dfrac{du}{dt} = \dfrac{d^2u}{dx^2} + f(x,t), \ x \in (-1,1), \ t \in [0,1], \\ u(x,0) = (1-x^2)\exp\left(\dfrac{1}{1+\epsilon}\right), \ x \in (-1,1), \\ u(-1,t) = 0, \ u(1,t) = 0, \ t \in (0,1], \end{cases} \tag{13}$$

where f is chosen such that $u(x,t) = (1-x^2)\exp\left(\dfrac{1}{(2t-1)^2+\epsilon}\right)$ is the exact solution.

The behavior of this model exhibits interesting characteristics depending on the value of the parameter ϵ. However, when ϵ becomes very small, the solution shows dramatic changes near $t = 0.5$, exhibiting distinct multiscale characteristics. An important observation is the relationship between the supervised loss term and the residual term - while the boundary conditions ensure the supervised loss term remains minimal (as the boundary value is 0 and the initial value stays below ϵ), the residual term grows considerably as ϵ decreases. This is quantitatively demonstrated as, when $\epsilon = 0.15$ the ratio of $\mathcal{L}_{\text{bc}} : \mathcal{L}_{\text{ic}} : \mathcal{L}_{\text{res}} = 1 : 10 : 10^7$. Traditional PINN methods generally perform well for smooth problems, but face challenges when dealing with such large disparities between supervised and residual terms. Our proposed W-PINN can effectively handle such loss imbalances.

Table 10 represents the parameters of the network for this Example 5. Further, Table 11 presents a comparative analysis of different methods for solving Example 5 with $\epsilon = 0.15$, evaluating them based on L_2-error and average training time. The conventional PINN fails to get an accurate prediction. The SA-PINN [17] shows significant improvement in accuracy but requires substantial computational resources, as evident from its lengthy training time. The MMPINN-DNN [19] achieves better accuracy with reduced computational cost. However, our proposed W-PINN approach demonstrates the most promising balance between accuracy and efficiency. It achieves comparable or better accuracy than other advanced methods while reducing the computational overhead, making it particularly attractive

Table 11: Performance comparison of various methods for solving Example 5 with $\epsilon = 0.15$.

Methods	L_2-error	Average Training Time
Conventional PINN	$1.07 \pm 0.11 \times 10^{0}$	-
SA-PINN [17]	1.76×10^{-3}	66.75 min
MMPINN-DNN [19]	$5.01 \pm 1.52 \times 10^{-4}$	11.02 min
W-PINN (proposed method)	$2.56 \pm 1.3 \times 10^{-4}$	4.5 min

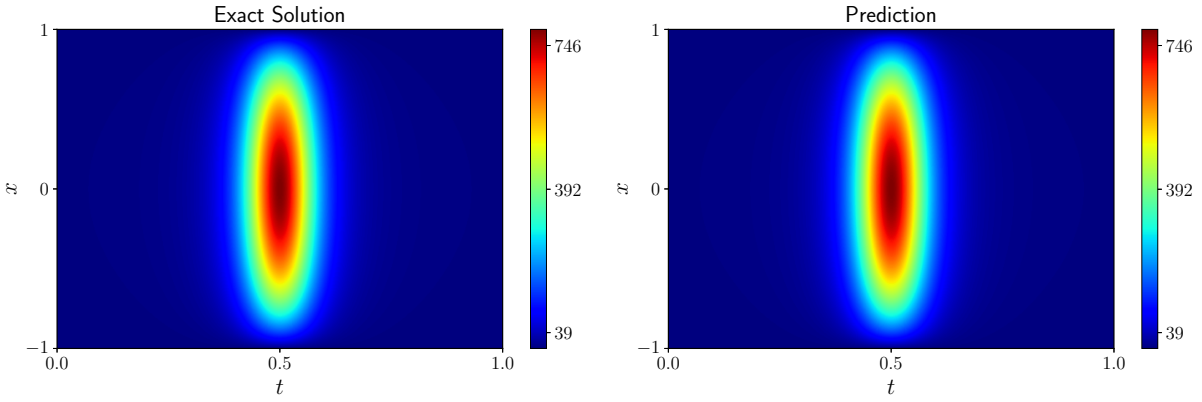


Figure 10: The exact solution (left) and the prediction of W-PINN (right) with $\epsilon = 0.15$ for high gradient heat conduction problem 5 .

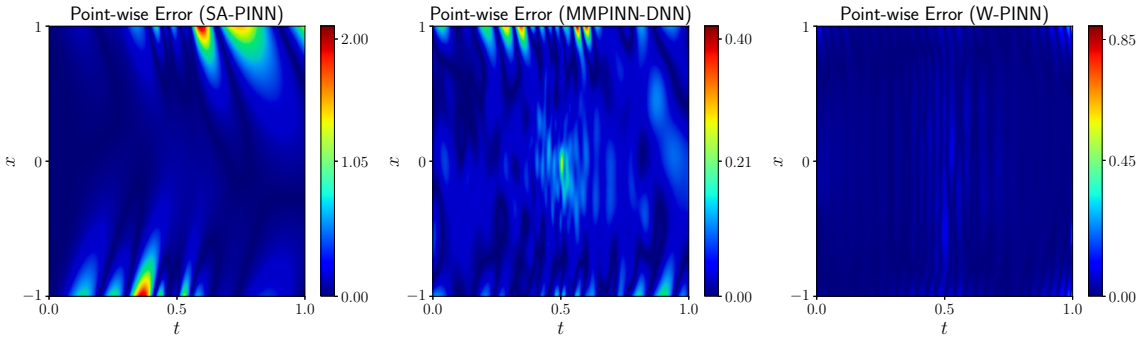


Figure 11: Point-wise absolute [redacted] for Example 5. From left to right: SA-PINN, MMPINN-DNN, and W-PINN.

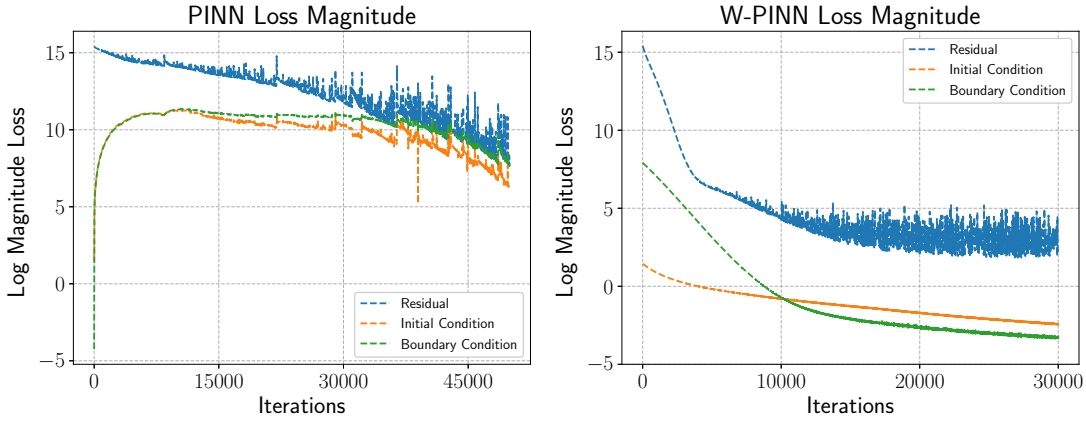


Figure 12: Loss curve of PINN (left) and W-PINN (right) for Example 5 with $\epsilon = 0.15$.

for practical applications. This improvement in both accuracy and computational efficiency highlights the effectiveness of our proposed W-PINN in handling multiscale characteristics.

Figure 10 demonstrates the similarity between the exact solution and the W-PINN's prediction for Example 5. Figure 11 shows point-wise absolute error comparisons across three approaches: SA-PINN,

Table 12: Parameters used for solving Example 6.

Parameters	Value
Translation hyperparameter γ	0.2
Set of resolutions (J_x, J_y)	[-4,5], [-4,5]
Number of hidden layers	6
Neurons per layer	50
Number of collocation points	10^4
Number of boundary points	10^3

Table 13: Performance comparison of various methods for solving Example 6.

Methods	L_2-error	Average Training Time
Conventional PINN	$4.93 \pm 1.56 \times 10^{-2}$	23.18 min
SA-PINN [17]	$1.27 \pm 1.11 \times 10^{-2}$	152.88 min
MMPINN-MFF[19]	$6.56 \pm 4.17 \times 10^{-4}$	22.75 min
MMPINN-INN[19]	$2.13 \pm 0.43 \times 10^{-4}$	64.29 min
W-PINN (proposed method)	$3.12 \pm 0.71 \times 10^{-4}$	6.3 min

MMPINN-DNN, and W-PINN. This error plot shows that errors for W-PINN throughout the domain remain consistently low except for extremely small regions near corners. Further, Figure 12 demonstrates the training progress of conventional PINN and W-PINN by showing their loss components over iterations. PINN shows unstable behavior with the boundary and initial condition losses initially increasing, suggesting potential training difficulties. In contrast, W-PINN exhibits a much more stable and efficient training pattern, where all three loss components (residual, initial condition, and boundary condition) consistently decrease from the start, reaching lower magnitudes in fewer iterations. This indicates that W-PINN achieves better optimization performance than the standard PINN approach.

Example 6. *Helmholtz equation with high-frequency:*

The Helmholtz equation is a crucial elliptic partial differential equation that models electromagnetic wave behavior. It appears in wide applications in physics and engineering. The equation combines the Laplacian operator (Δu) with a wave number term (c^2), and is typically studied on bounded domains with appropriate boundary conditions. In the high-frequency regime, this equation becomes particularly challenging to solve numerically due to the oscillatory nature of its solutions.

$$\begin{cases} \Delta u(x,y) + c^2 u(x,y) = f(x,y), & (x,y) \in \Omega = (-1,1) \times (-1,1), \\ u(x,y) \quad x,y), & (x,y) \in \partial\Omega. \end{cases} \tag{14}$$

f and p are ~~obtained~~ in such a way $u(x,y) = \sin(b_1 \pi x) \sin(b_2 \pi y)$ is the exact solution.

Due to the significant initial imbalance between residual and supervised components, conventional PINN fails to provide accurate predictions as the optimization process becomes heavily skewed towards residual minimization [10].

To test the performance of W-PINN, we set the model parameters as $c = 1, b_1 = 1$, and $b_2 = 8$. e 12 represents the parameters of the network for this Example 6. The comparative analysis pres d in Table 13 demonstrates that W-PINN achieves comparable or superior accuracy in terms of L_2-error relative to state-of-the-art methods in recent literature. Along with its significantly reduced training

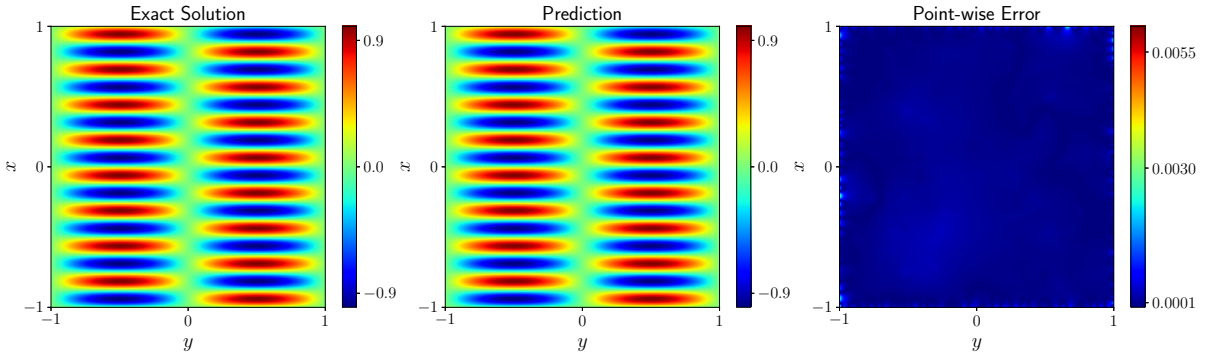


Figure 13: From left to right: The exact solution, the predicted solution, and the point-wise absolute error using W-PINN for the Helmholtz equation 6.

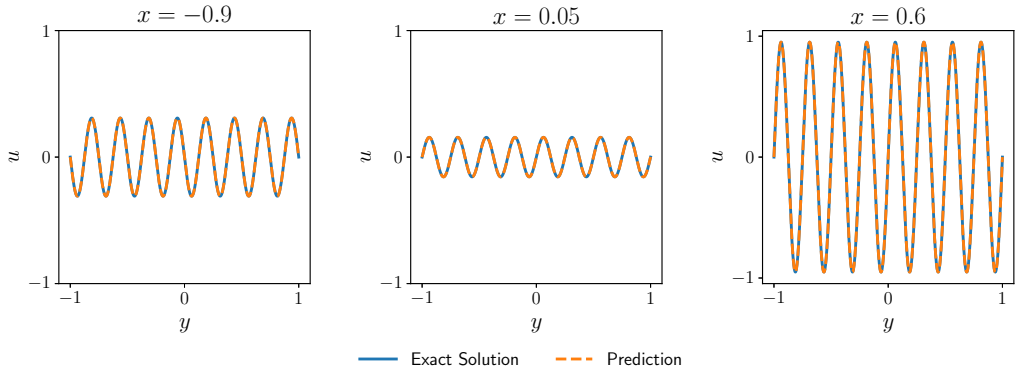


Figure 14: Comparison of the exact solution (solid blue line) and the prediction (dashed orange line) using W-PINN at different spatial stamps for Example 6.

time, W-PINN is established as a particularly attractive framework. Figure 13 illustrates the remarkable agreement between the analytical solution and W-PINN predictions across the entire domain. In Figure 14, cross-sectional comparisons at various x-coordinates further validate the method's accuracy, with predicted solutions indistinguishable from the exact solutions.

Example 7. *Allen-Cahn reaction-diffusion equation:*

The Allen-Cahn reaction-diffusion PDE is a widely used model in materials science, particularly for simulating phase separation processes in metallic alloys [40, 41]. The equation describes the evolution of an order parameter, u, which represents the phase state of the material. The PDE combines a diffusion term with a non-linear reaction term that drives the phase separation. The Allen-Cahn equation used in this study is defined as:

$$\begin{cases} u_t - \epsilon u_{xx} + 5u^3 - 5u = 0, \quad x \in [-1, 1], \quad t \in [0, 1], \\ u(t, -1) = u(t, 1), \\ u_x(t, -1) = u_x(t, 1), \\ u(x, 0) = x^2 \cos(\pi x), \end{cases} \tag{15}$$

here ϵ denotes a singularly perturbed parameter. For numerical computations, we choose $\epsilon = 10^{-4}$. Moreover, the exact solution to this problem is unknown, so we obtained a numerical solution using Scipy solver and treated it as an [redacted] solution. The Allen-Cahn equation is particularly challenging due to its rapid changes across sp[redacted]nd time, making it difficult to model accurately. Additionally,

Table 14: Parameters used for solving Example 7.

Parameters	Value
Translation parameter γ	0.4
Resolutions (J_x, J_t)	[-5,6], [-5,5]
Number of hidden layers	6
Neurons per layer	100
Number of collocation points	2×10^4
Number of boundary points	2×10^3
Number of initial points	10^3

Table 15: Performance comparison of various methods for solving Example 7 with $\epsilon = 10^{-4}$.

Methods	L_2-error
Conventional PINN	0.96 ± 0.06
Time-adaptive approach [42]	$8.0 \times 10^{-2} \pm 0.56 \times 10^{-2}$
SA-PINN [17]	$2.1 \times 10^{-2} \pm 1.21 \times 10^{-2}$
W-PINN (proposed method)	$4.8 \times 10^{-2} \pm 0.6 \times 10^{-2}$

its periodic boundary conditions provide an extra layer of complexity, making it an ideal benchmark for testing the model's capabilities.

Table 15 presents a comparative analysis of different PINN-based methods. The conventional PINN completely fails to capture the complex dynamics of the problem. While the time-adaptive method [42] shows some improvement, its accuracy remains unsatisfactory. In contrast, our proposed W-PINN achieves accuracy comparable to SA-PINN [17] while demonstrating computational efficiency with approximately twice the speed (30 ms/iteration) of SA-PINN. Figure 15 illustrates the predicted dynamics using the W-PINN method, where the predominance of dark blue regions in the error distribution plot indicates consistently low prediction errors across the solution domain. Further, Figure 16 compares W-PINN prediction with the exact solution at three different time stamps, validating the model's ability to capture the temporal evolution of the system dynamics accurately.

Example 8. *Maxwell's Equation:*

Maxwell's equations represent the fundamental principles governing electromagnetic theory, describing the relationships between electric and magnetic fields and their interaction with matter. Their applications span numerous fields of engineering and physics. In telecommunications, these equations govern the propagation of electromagnetic waves through optical fibers and wireless channels. The medical industry relies heavily on Maxwell's equations for diagnostic imaging technologies, particularly in magnetic resonance imaging (MRI) [43]. In the aerospace sector, these equations are essential for radar system design and satellite communication systems. In their differential form, Maxwell's equations are expressed as:

$$\begin{cases} \nabla \times \boldsymbol{E}(t, \boldsymbol{x}) \ = -\mu(\boldsymbol{x}) \dfrac{\partial \boldsymbol{H}(t, \boldsymbol{x})}{\partial t}, \\ \nabla \times \boldsymbol{H}(t, \boldsymbol{x}) = \varepsilon(\boldsymbol{x}) \dfrac{\partial \boldsymbol{E}(t, \boldsymbol{x})}{\partial t}, \end{cases} \tag{16}$$

where $\boldsymbol{E}$ is the electric field vector, $\boldsymbol{H}$ is the magnetic field vector. The material properties are characterized by μ, the magnetic permeability, which quantifies the medium's response to magnetic

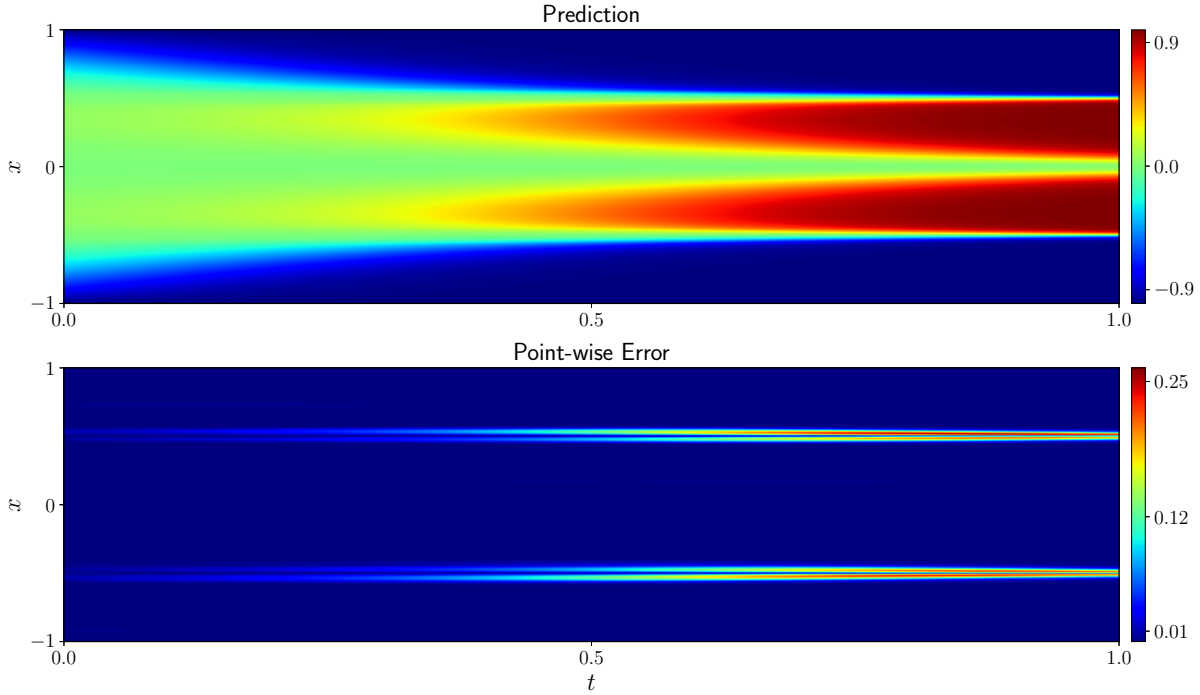


Figure 15: Top: Predicted solution of Allen-Cahn equation 7 via W-PINN. Bottom: point-wise absolute error distribution.

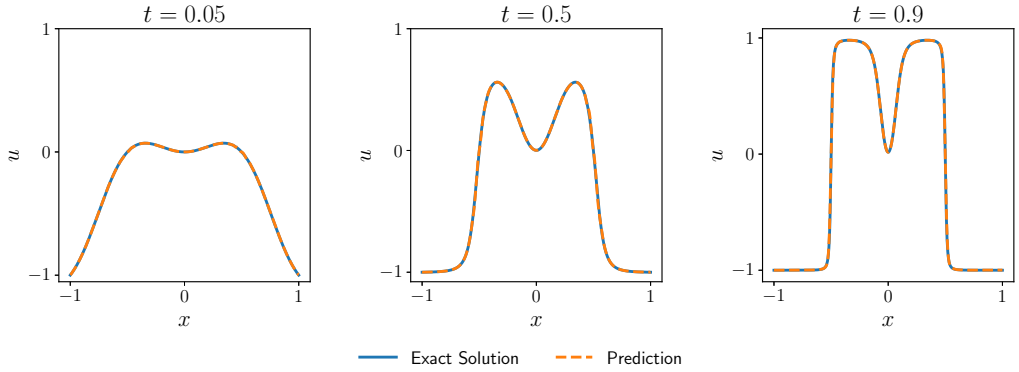


Figure 16: Comparison of exact solution (solid blue line) and predicted solution (dashed orange line) using W-PINN at different time instances for Example 7.

fields, and ϵ, the electric permittivity, which describes the medium's capacity to store electrical energy. The operators ∇ and $\times$ represent the del operator and cross product, respectively.

A significant challenge in solving these equations arises from their inherent rapid oscillations in both space and time. This characteristic poses particular difficulties for traditional PINNs, especially when dealing with heterogeneous media where μ and ϵ exhibit discontinuous behavior at media interfaces. Here, we present a solution of Maxwell's equation in both a homogeneous and a heterogeneous medium via W-PINN and compare them with the traditional PINN.

A. Homogeneous media

We first investigate the performance of our proposed method by solving Maxwell's equations in a

Table 16: Parameters used for solving Example 8.

Parameters	Value
Translation hyperparameter γ	0.5
Resolutions (J_x, J_t)	[-5,5], [-5,5]
Number of hidden layers	8
Neurons per layer	50
Number of collocation points	10^4
Number of boundary points	10^3
Number of initial points	500

one-dimensional cavity model with homogeneous media. The governing equations are:

$$\frac{\partial E_y}{\partial t} = -\frac{1}{\epsilon}\frac{\partial H_z}{\partial x}, \quad \frac{\partial H_z}{\partial t} = -\frac{1}{\mu}\frac{\partial E_y}{\partial x}, \quad x \in [0,1], \quad t \in [0,1]. \tag{17}$$

The system is subject to perfectly electric conductor (PEC) boundary conditions:

$$\begin{cases} E_y(0,t) = E_y(1,t) = 0, \\ \dfrac{\partial H_z(x,t)}{\partial x}\Big|_{x=0,1} = 0. \end{cases} \tag{18}$$

The ground truth for the example is given by:

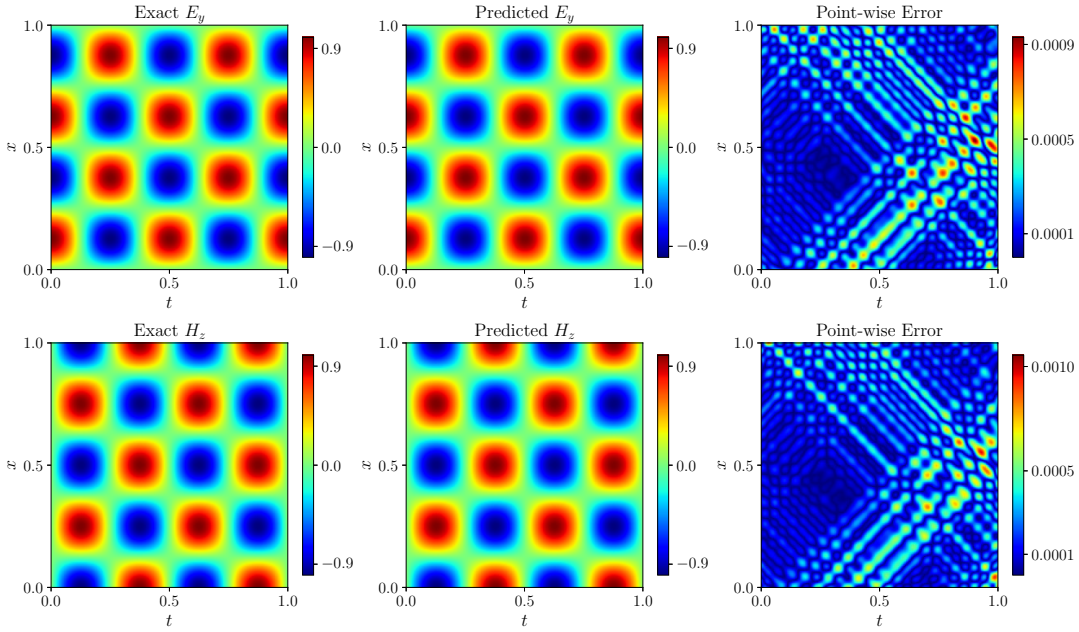


Figure 17: Solution of Maxwell's equation 17 in homogeneous medium using W-PINN. Top: E_y Exact, predicted, and corresponding point-wise absolute error. Bottom: H_z Exact, predicted, and corresponding point-wise absolute error.

$$\begin{cases} E_y = \sin(n\pi x)\cos(\omega t), \\ H_z = -\cos(n\pi x)\sin(\omega t), \end{cases} \tag{19}$$

where $n = 4$ is the order of cavity mode, cavity frequency $\omega = n\pi/l$ and l is the length of medium.

Over five random runs, W-PINN achieved average relative L2-errors of $5.47 \pm 1.12 \times 10^{-4}$ and $5.51 \pm 2.01 \times 10^{-4}$ for E_y and H_z, respectively. In contrast, traditional PINNs exhibited notably higher errors of $3.08 \pm 1.18 \times 10^{-3}$ and $5.18 \pm 1.93 \times 10^{-3}$. The computational efficiency of W-PINN is evident in its training time, requiring an average of 6.2 minutes with 18 ms per iteration on Nvidia A6000 GPU. The network architecture parameters are tabulated in Table 16. Figure 17 presents the W-PINN predictions for both electric and magnetic fields, alongside their corresponding point-wise errors within the cavity. The remarkably low point-wise error distribution validates the high accuracy of our approach in capturing the electromagnetic field behavior.

B. Heterogeneous media

For this case, we chose Maxwell's equation 17 in a heterogeneous medium with the following analytical solution:

$$E_y = \begin{cases} \cos(2t - 2x + 1) \\ +0.5\cos(2t + 2x - 1), & \text{if } x \in \Omega_1, \\ \\ 1.5\cos(2t - 3x + 1.5), & \text{if } x \in \Omega_2, \end{cases} \tag{20}$$

$$H_z = \begin{cases} \cos(2t - 2x + 1) \\ -0.5\cos(2t + 2x - 1), & \text{if } x \in \Omega_1, \\ \\ 0.5\cos(2t - 3x + 1.5), & \text{if } x \in \Omega_2, \end{cases} \tag{21}$$

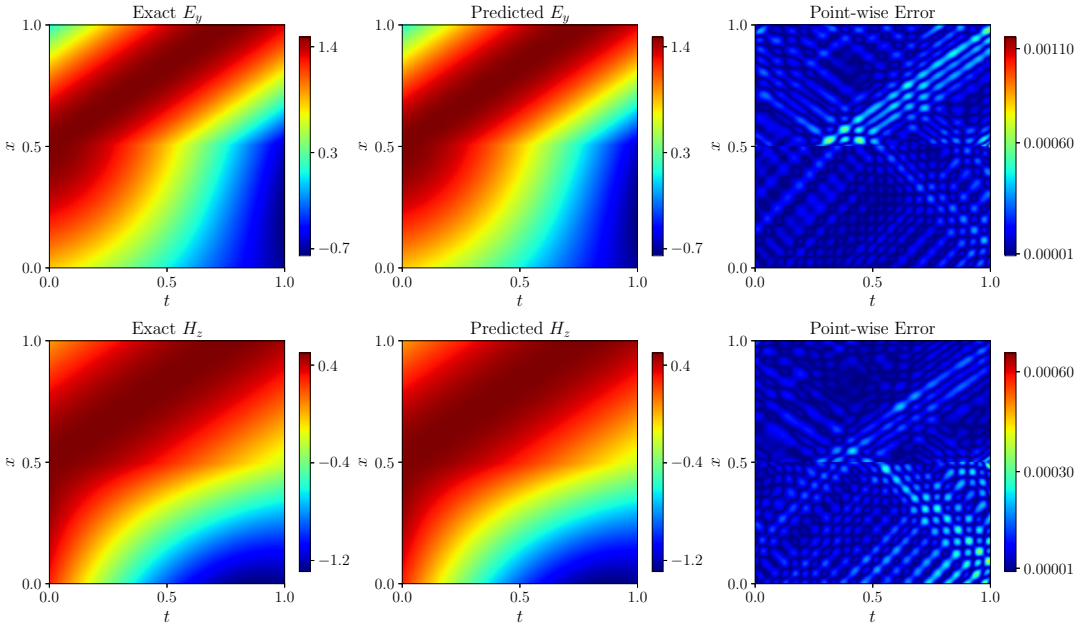


Figure 18: Solution of Maxwell's equation 17 in heterogeneous medium using W-PINN. Top: E_y Exact, predicted, and corresponding point-wise absolute error. Bottom: H_z Exact, predicted, and corresponding point-wise absolute error.

Table 17: Parameters used for solving Example 9.

Parameters	Value
Translation hyperparameter γ	0.5
Resolutions (J_x, J_y)	[-15,5], [-15,5]
Number of hidden layers	12
Neurons per layer	100
Number of collocation points	4×10^4
Number of boundary points	2000

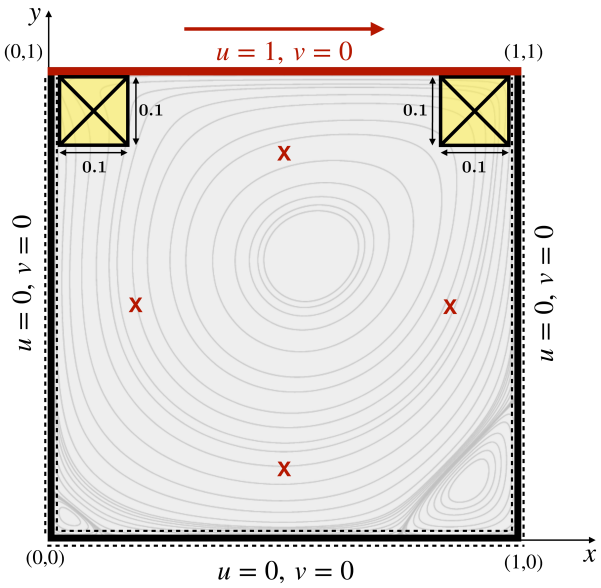


Figure 19: A lid-driven unit square cavity. For Re = 400: additional collocation points are sampled from the yellow crossed region, and red crosses indicate reference points.

where $\Omega_1 = [0, 0.5]$ and $\Omega_2 = [0.5, 1]$. Here $\mu = 1, \epsilon = 1$ in Ω_1 and $\mu = 4.5, \epsilon = 0.5$ in Ω_2. According to the electromagnetic interface condition:

$$\begin{cases} E_y(0.5,t)\big|_{x\in\Omega 1} = E_y(0.5,t)\big|_{x\in\Omega 2}, \\ H_z(0.5,t)\big|_{x\in\Omega 1} = H_z(0.5,t)\big|_{x\in\Omega 2}. \end{cases} \tag{22}$$

In this challenging context of heterogeneous media, W-PINN demonstrates remarkable accuracy, achieving average relative L2-errors of $2.41 \pm 1.22 \times 10^{-4}$ and $2.81 \pm 1.51 \times 10^{-4}$ for electric (E_y) and magnetic (H_z) fields, respectively. These results represent a two-order-of-magnitude improvement over traditional PINNs, which exhibit substantially higher errors of $1.37 \pm 1.18 \times 10^{-2}$ and $2.07 \pm 0.51 \times 10^{-2}$. W-PINN took an average of 16.1 minutes with around 30 ms/iteration [redacted] Nvidia A6000 GPU. The network parameters are the same as in Table 16. Figure 18 illustrat[redacted] W-PINN predictions for both electric and magnetic fields, alongside their corresponding point-wise absolute errors. While the media interface presents the greatest computational challenge, W-PINN maintains impressive accuracy throughout the domain, successfully capturing the Electromagnetic field behavior across the media.

Example 9. *Lid-Driven Cavity flow:*

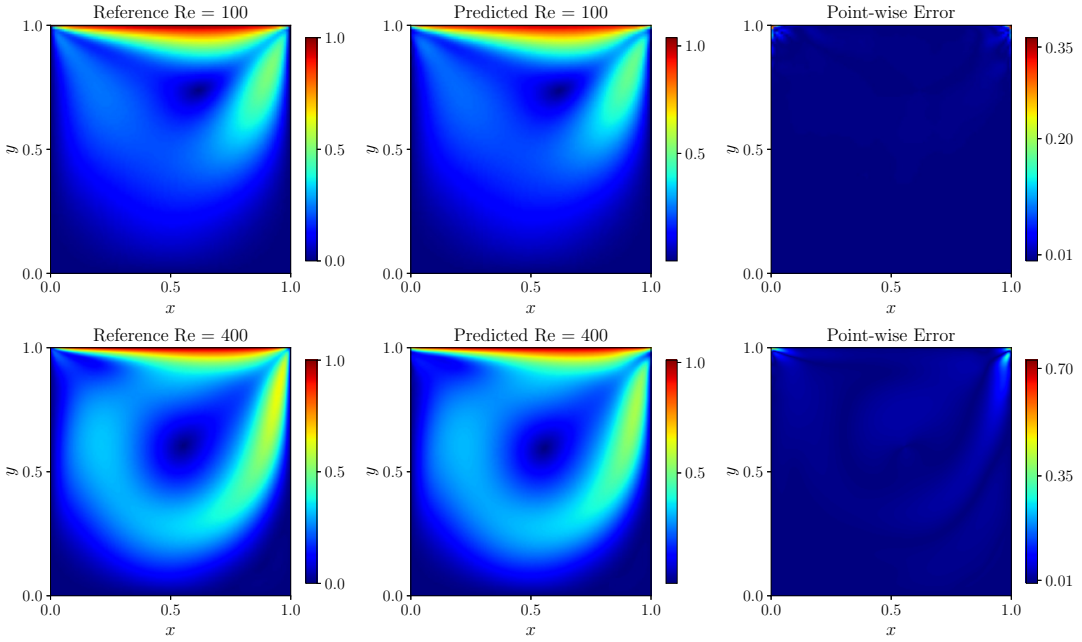


Figure 20: From left to right: The reference fluid flow, the W-PINN predicted fluid speed, and the point-wise absolute error for the steady-state lid-driven cavity flow 23 for Re=100 (top) and Re=400 (bottom).

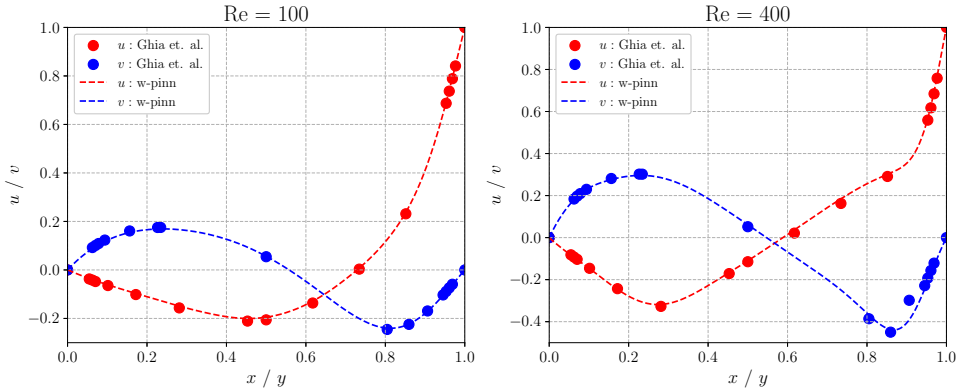


Figure 21: Comparison of W-PINN solution (dashed line) with Ghia's benchmark (scattered circles) for the steady-state lid-driven cavity flow 23 for Re=100 (left) and Re=400 (right). These data points lie on the horizontal and vertical lines through the geometric center of the cavity, for red color: $x = 0.5$ and blue color: $y = 0.5$

The steady state two-dimensional incompressible lid-driven cavity flow is a classic benchmark problem in computational fluid dynamics. The flow is governed by the incompressible Navier-Stokes equations, which present significant challenges due to their non-linear nature and the presence of pressure-velocity coupling. As illustrated in Figure 19, the problem consists of a unit square cavity where the top wall moves with a uniform velocity of 1 unit while maintaining no-slip conditions on all other walls. The governing Navier-Stokes equations for this system are:

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} = 0, \\ u\dfrac{\partial u}{\partial x} + v\dfrac{\partial u}{\partial y} + \dfrac{\partial p}{\partial x} - \dfrac{1}{\mathrm{Re}}\left(\dfrac{\partial^2 u}{\partial x^2} + \dfrac{\partial^2 u}{\partial y^2}\right) = 0, \\ u\dfrac{\partial v}{\partial x} + v\dfrac{\partial v}{\partial y} + \dfrac{\partial p}{\partial y} - \dfrac{1}{\mathrm{Re}}\left(\dfrac{\partial^2 v}{\partial x^2} + \dfrac{\partial^2 v}{\partial y^2}\right) = 0, \end{cases} \tag{23}$$

where u and v are the fluid velocities in x and y directions respectively, and p is the fluid pressure. Re is the Reynolds number, which represents the ratio of inertial forces to viscous forces. We evaluate our W-PINN approach for two distinct flow regimes: Re = 100 and Re = 400, and compare the results with the solution obtained through COMSOL and also validate against the widely accepted benchmark solutions of Ghia et al.[44].

For this example, the network's parameters are tabulated in 17. Figure 20 presents the predicted fluid flow speed distribution ($|V| = \sqrt{u^2 + v^2}$), along with a comparison with the COMSOL reference solution, demonstrating successful capture of the characteristic vortex structure. The relative L2-error for Re = 100 and Re = 400 are $2.75 \pm 1.02 \times 10^{-2}$ and $6.17 \pm 1.05 \times 10^{-2}$ respectively. W-PINN took around 20 minutes of training with roughly 6 it/s on Nvidia A6000. At the higher Reynolds number of 400, where flow complexity increases, we uniformly sampled an additional 10^4 collocation points from the upper left and upper right corners of the cavity (crossed region in Figure 19). In addition, only four strategically chosen reference points (marked by red crosses in Figure 19) were added. Interestingly, these minimal additions reduced the relative L2-error roughly by half. A validation against Ghia's benchmark data is illustrated in Figure 21. These results demonstrate W-PINN's robust capability to handle complex fluid dynamics problems to a good extent.

4. Conclusion

~~This study~~ successfully demonstrated the advantages of W-PINNs over other PINN-based methods in addressing multi-scale problems characterized by steep gradients, rapid oscillations, or singular behavior. The wavelet theory served as the foundation for the development of W-PINN, which combined the learning efficiency of PINN with the localization properties of wavelets (both in scale and space) to capture non-linear information within localized regions. A key advantage of the proposed framework is ~~its elimination of~~ the need for automatic differentiation (autograd) in computing residual derivatives, which substantially accelerates training compared to methods that rely on autograd. To support faster convergence behavior, we provide a comparative analysis of W-PINN using NTK theory. A limitation of the proposed method is its dependence on a careful selection of the wavelet basis for optimal performance. Furthermore, W-PINNs show strong potential for application to other classes of problems that exhibit inherent singular behavior, such as fractional differential equations. These models often present non-locality and memory effects, where the localized resolution capabilities of wavelets can offer a significant advantage.

CRediT authorship contribution statement

Himanshu Pandey: Conceptualization, Methodology, Programming, Formal analysis, Writing - original draft. **Anshima Singh:** Conceptualization, Investigation, Methodology, Formal analysis, Writing - original draft. **Ratikanta Behera:** Funding acquisition, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that there are no conflicts of interest.

Data Availability Statement

No data were used for the research described in the article. However, the source codes of problems addressed in this study are available at `https://github.com/himanshup21/W-PINN.git`

Acknowledgments

Ratikanta Behera is supported by the Anusandhan National Research Foundation (ANRF), Government of India, under Grant No. EEQ/2022/001065. We want to express our gratitude to the editor for taking the time to handle the manuscript and to the anonymous referees for their constructive comments.

Preprint

A preprint of this paper is available on arxiv with Ref. No. arXiv:2409.11847

References

[1] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000. doi:10.1109/72.712178.

[2] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.

[3] Z. Hu, K. Shukla, G. E. Karniadakis, K. Kawaguchi, Tackling the curse of dimensionality with physics-informed neural networks, Neural Netw. 176 (2024) 106369. doi:10.1016/j.neunet.2024.106369.

[4] E. Kharazmi, Z. Zhang, G. E. Karniadakis, hp-vpinns: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Engrg. 374 (2021) 113547. doi:https://doi.org/10.1016/j.cma.2020.113547.

[5] J. Yu, L. Lu, X. Meng, G. E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Engrg. 393 (2022) Paper No. 114823, 22. doi:10.1016/j.cma.2022.114823.

[6] A. D. Jagtap, G. E. Karniadakis, Extended physics-informed neural networks (XPINNs): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations, Commun. Comput. Phys. 28 (5) (2020) 2002–2041. doi:10.4208/cicp.oa-2020-0164.

[7] S. Cuomo, V. Schiano Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: where we are and what's next, J. Sci. Comput. 92 (3) (2022) Paper No. 88, 62. doi:10.1007/s10915-022-01939-z.

[8] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440. doi:10.1038/s42254-021-00314-5.

[9] S. Wang, H. Wang, P. Perdikaris, On the eigenvector bias of Fourier feature networks: from regression to solving multi-scale PDEs with physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 384 (2021) Paper No. 113938, 28. doi:10.1016/j.cma.2021.113938.

[10] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081. doi:10.1137/20M1318043.

[11] H.-G. Roos, M. Stynes, L. Tobiska, Robust numerical methods for singularly perturbed differential equations, 2nd Edition, Vol. 24 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2008, convection-diffusion-reaction and flow problems. doi:10.1007/978-3-540-34467-4.

[12] G. Farhani, N. H. Dashtbayaz, A. Kazachek, B. Wang, A simple remedy for failure modes in physics informed neural networks, Neural Netw. 183 (2025) 106963. doi:10.1016/j.neunet.2024.106963.

[13] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: International conference on machine learning, PMLR, 2019, pp. 5301–5310.

[14] A. Jacot, F. Gabriel, C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 31, Curran Associates, Inc., 2018.
URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf

[15] Y. Cao, Z. Fang, Y. Wu, D.-X. Zhou, Q. Gu, Towards understanding the spectral bias of deep learning, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 2205–2211, main Track. doi:10.24963/ijcai.2021/304.

[16] S. Wang, X. Yu, P. Perdikaris, When and why PINNs fail to train: a neural tangent kernel perspective, J. Comput. Phys. 449 (2022) Paper No. 110768, 28. doi:10.1016/j.jcp.2021.110768.

[17] L. D. McClenny, U. M. Braga-Neto, Self-adaptive physics-informed neural networks, J. Comput. Phys. 474 (2023) Paper No. 111722, 23. doi:10.1016/j.jcp.2022.111722.

[18] A. Arzani, K. W. Cassel, R. M. D'Souza, Theory-guided physics-informed neural networks for boundary layer problems with singular perturbation, J. Comput. Phys. 473 (2023) Paper No. 111768, 15. doi:10.1016/j.jcp.2022.111768.

[19] Y. Wang, Y. Yao, J. Guo, Z. Gao, A practical PINN framework for multi-scale problems with multi-magnitude loss terms, J. Comput. Phys. 510 (2024) Paper No. 113112, 19. doi:10.1016/j.jcp.2024.113112.

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

[21] N. N., S. Chakraborty, Stochastic projection based approach for gradient free physics informed learning, Comput. Methods Appl. Mech. Engrg. 406 (2023) Paper No. 115842, 21. doi:10.1016/j.cma.2022.115842.

[22] Y. Xiao, L. Yang, Y. Du, Y. Song, C. Shu, Radial basis function-differential quadrature-based physics-informed neural network for steady incompressible flows, Physics of Fluids 35 (7) (2023). doi:10.1063/5.0159224.

[23] R. Sharma, V. Shankar, Accelerated training of physics-informed neural networks (pinns) using meshless discretizations, NeurIPS 35 (2022) 1034–1046.
URL https://proceedings.neurips.cc/paper_files/paper/2022/file/0764db1151b936aca59249e2c1386101-Paper-Con
pdf

[24] V. Kumar, M. Mehra, Wavelet optimized finite difference method using interpolating wavelets for self-adjoint singularly perturbed problems, J. Comput. Appl. Math. 230 (2) (2009) 803–812. doi:10.1016/j.cam.2009.01.017.

[25] N. Kumar, V. Kumar, M. Mehra, Legendre wavelet collocation method for singularly-perturbed problems using operational matrix and Laplace transformation, ZAMM Z. Angew. Math. Mech. 105 (2) (2025) Paper No. e202400200. doi:10.1002/zamm.202400200.

[26] S. G. Mallat, A theory for multiresolution signal decomposition: the wavelet representation, IEEE Trans. Pattern Anal. MAch. Intell. 11 (7) (1989) 674–693. doi:10.1137/1.9781611970104.

[27] I. Daubechies, Ten lectures on wavelets, Vol. 61 of CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992. doi:10.1137/1.9781611970104.

[28] R. Behera, M. Mehra, Integration of barotropic vorticity equation over spherical geodesic grid using multilevel adaptive wavelet collocation method, Appl. Math. Model. 37 (7) (2013) 5215–5226. doi:10.1016/j.apm.2012.10.027.

[29] K. Goyal, M. Mehra, An adaptive meshfree spectral graph wavelet method for partial differential equations, Appl. Numer. Math. 113 (2017) 168–185. doi:10.1016/j.apnum.2016.11.011.

[30] Z. Uddin, S. Ganga, R. Asthana, W. Ibrahim, Wavelets based physics informed neural networks to solve non-linear differential equations, Sci Rep 13 (1) (2023) 2882. doi:10.1038/s41598-023-29806-3.

[31] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (4) (1989) 303–314. doi:10.1007/BF02551274.

[32] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, nature 323 (6088) (1986) 533–536. doi:10.1038/323533a0.

[33] A. l. m. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2017) Paper No. 153, 43.
URL http://jmlr.org/papers/v18/17-468.html

[34] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

doi:10.48550/arXiv.1412.6980.

[35] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Y. W. Teh, M. Titterington (Eds.), Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Vol. 9 of Proceedings of Machine Learning Research, PMLR, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.
URL https://proceedings.mlr.press/v9/glorot10a.html

[36] C. M. Bender, S. A. Orszag, Advanced mathematical methods for scientists and engineers. I, Springer-Verlag, New York, 1999, asymptotic methods and perturbation theory, Reprint of the 1978 original. doi:10.1007/978-1-4757-3069-2.

[37] Z. Cen, F. Erdogan, A. Xu, An almost second order uniformly convergent scheme for a singularly perturbed initial value problem, Numer. Algorithms 67 (2) (2014) 457–476. doi:10.1007/s11075-013-9801-0.

[38] J. Su, Delayed oscillation phenomena in the FitzHugh Nagumo equation, J. Differential Equations 105 (1) (1993) 180–215. doi:10.1006/jdeq.1993.1087.

[39] K. Lan, Dream fusion in octahedral spherical hohlraum, Matter Radiat. Extrem. 7 (5) (2022). doi:https://doi.org/10.1063/5.0103362.

[40] N. Moelans, B. Blanpain, P. Wollants, An introduction to phase-field modeling of microstructure evolution, Calphad 32 (2) (2008) 268–294. doi:https://doi.org/10.1016/j.calphad.2007.11.003.

[41] C. Kunselman, V. Attari, L. McClenny, U. Braga-Neto, R. Arroyave, Semi-supervised learning approaches to class assignment in ambiguous microstructures, Acta Mater. 188 (2020) 49–62. doi:https://doi.org/10.1016/j.actamat.2020.01.046.

[42] C. L. Wight, J. Zhao, Solving Allen-Cahn and Cahn-Hilliard equations using the adaptive physics informed neural networks, Commun. Comput. Phys. 29 (3) (2021) 930–954. doi:10.4208/cicp.oa-2020-0086.

[43] R. W. Brown, Y.-C. N. Cheng, E. M. Haacke, M. R. Thompson, R. Venkatesan, Magnetic resonance imaging: physical principles and sequence design, John Wiley & Sons, 2014. doi:10.1002/9781118633953.

[44] U. Ghia, K. Ghia, C. Shin, High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method, J. Comput. Phys. 48 (3) (1982) 387–411. doi:https://doi.org/10.1016/0021-9991(82)90058-4.