

Article

PDGM-PINN: Partial Derivative Guided Multi-Branch Physics-Informed Neural Network

Shangpeng Lei ¹, Chenghan Yang ¹, Roberts Grants ², Uldis Grunde ² and Nadezhda Kunicina ^{2,*}

¹ Faculty of Information Technology and Artificial Intelligence, Al-Farabi Kazakh National University, Almaty 050040, Kazakhstan; ley_shanpen@live.kaznu.kz (S.L.)

² Faculty of Computer Science, Information Technology and Energy, Institute of Industrial Electronics, Electrical Engineering and Energy, Riga Technical University, 1048 Riga, Latvia

* Correspondence: nadezda.kunicina@rtu.lv; Tel.: +371-2616-2662

Abstract

With the development of scientific machine learning (SciML), the proposal of physics-informed neural networks (PINNs) has provided a powerful paradigm for solving partial differential equations (PDEs). While PINNs perform well in solving high-dimensional PDEs, they perform worse than traditional numerical methods for low-dimensional problems. This discrepancy arose from potential convergence conflicts induced by distinct physical magnitude of loss terms. To decouple the convergence conflicts, we propose a partial derivative guided multi-branch physics-informed neural network (PDGM-PINN). Inspired by SciML, we treat both the solution and partial derivatives as dependent variables to be predicted. The partial derivatives are directly predicted by sub-branches, while the main branch approximates the PDE solution, and all branches share error backpropagation information. Furthermore, we redesign the loss function. The loss of the governing equation is computed with the solution and partial derivatives predicted by the main and sub-branches. Schwarz's theorem and Kullback–Leibler divergence are incorporated into the loss terms as soft constraints of partial derivatives continuity and residual distributions consistency for the governing equations. We conducted comprehensive experimental evaluations on seven PDEs, and ablation experiments, sensitivity analyses, and complexity analyses were carried out to investigate the rationality of PDGM-PINN. The results demonstrate that PDGM-PINN achieves the best performance among PINN variants with the fewest trainable parameters, effectively avoiding architectural redundancy.

Keywords: physics-informed neural networks; partial differential equations; KL divergence; scientific machine learning

MSC: 68T01



Academic Editor: Leonid Piterbarg

Received: 19 March 2026

Revised: 13 April 2026

Accepted: 13 April 2026

Published: 17 April 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Partial differential equations (PDEs) are widely used to describe phenomena in science and engineering. However, owing to the strong nonlinearity of PDEs, analytical solutions are difficult to derive [1]. With the development of scientific computing, research on approximate solutions of PDEs has been greatly advanced. Traditionally, numerical discretization to approximate solutions has become the classical numerical paradigm for PDEs, achieving remarkable success in both theoretical derivation and engineering computation [2]. Meanwhile, the differential transform method (DTM), which solves PDEs by seeking an analytical solution in the form of a certain functional series, also provides a

universal approach [3,4]. Nevertheless, the performance of numerical methods or DTM relies heavily on the complex preprocess, such as mesh generation and complete parameter information [5], which significantly limits their generalization and applicability as solutions. In recent years, benefiting from the growth of data and computation resources, machine learning (ML) techniques have developed rapidly, and their powerful nonlinear approximation capability has been widely recognized [6,7]. Therefore, embedding prior knowledge as soft constraints into ML models and transforming numerical computations into data-driven approximation problems have attracted increasing attention, rising into the research field of scientific machine learning (SciML) [8].

Due to the powerful nonlinear approximation capability exhibited by neural networks (NNs), physics-informed neural networks (PINNs) have been proposed [9] for solving PDEs. Compared with traditional numerical solvers, PINNs solve PDEs by embedding the mathematical equations into the loss function, where the partial derivatives are calculated by the auto difference mechanism of NNs [6]. Consequently, the introduction of PINNs provides a new solution to alleviate the curse of dimensionality [10]. Meanwhile, PINNs do not rely on grid generation [11], which offers great convenience for solving problems with complex geometries. Currently, PINNs have been successfully applied in a wide range of physical fields such as heat transfer [12], mechanics [13] and acoustics [14].

However, as a novel solver for PDEs, PINNs still face some challenges. PINNs solve PDEs by training the weights and biases of the NNs to approximate the solution, minimizing the residuals of the embedded governing equations and boundary conditions over the sampled points dataset in the computational domain. Thus, the PINNs' solution performance for PDEs is affected by the sampling strategy and network design. According to different guiding features, sampling strategies of PINNs can be categorized into residual [15,16] and importance [17–19] sampling. However, the sampling strategies often require multiple computation training rounds, which causes high memory overhead and weak generalization, limiting the performance of PINNs. In addition, due to the magnitude discrepancy among physical loss terms, PINNs suffer from convergence conflicts during training with sparse sampling data [20], leading to PDE solution accuracy that is often not superior to traditional methods [21]. To alleviate convergence conflict, extensive studies have focused on loss function optimization, such as adaptive weighting [22–24] and dynamic learning strategy [25–27]. With further analysis, researchers have found that only improving the loss function cannot fundamentally resolve the convergence issue of PINNs. Accordingly, neural network architectures adapted to PINNs have attracted increasing attention. SK-PINNs replace automatic differentiation of PINNs by a smoothing kernel function, which embeds random Fourier into multi-layer perceptron (MLP) [28]. Deeper PINNs add element-wise multiplication layers into deep NN layers, enhancing the nonlinear expression capability in initialization and high-loss regions [29]. ML-PINNs [30] are developed by integrating Mamba with long short-term memory (LSTM) networks, enabling shorter input sequences and lower memory consumption. Although numerous valuable PINN variants have been developed, existing architectural improvements mainly focus on the integration and application of advanced algorithms, while insufficient work has been made to alleviate convergence conflicts in terms of neural network architectures.

In the process of solving high-dimensional PDEs, classical numerical methods are constrained by algorithmic design and grid generation, leading to unsatisfactory approximation errors. In contrast, PINNs can be equipped with more diverse optimization alternatives, such as network depth, hidden neurons, loss functions, and refined sampling strategies, which can guarantee accuracy more generally. However, when solving low-dimensional PDEs, classical numerical methods can avoid generalization errors by employing weak formulations and Gaussian quadrature rules, while PINNs often perform

unsatisfactorily [31]. Obviously, due to the computational limitations of traditional numerical methods in high-dimensional PDEs, studies were usually focused on studying the application of PINNs in high-dimensional problems [32,33]. However, low-dimensional problems are also widely used to describe the physical problems in science and engineering, and the performance for solving low-dimensional problems provides a clearer comparison between numerical methods and PINNs. If PINNs were better than numerical methods in solving low-dimensional problems, they are more likely to release greater potential in high-dimensional cases, and this issue has gradually attracted attention [34].

To further optimize the generalization of PINNs in solving low-dimensional problems and decouple PDEs to alleviate the convergence conflict during training, we propose a partial derivative guided multi-branch physics-informed neural network (PDGM-PINNs). First-order partial derivatives serve as an intermediary linking the solution function and high-order derivatives. Therefore, we use branch neural networks to directly predict the first-order partial derivatives. To guarantee the convergence consistency of the first-order partial derivatives, we design a dedicated loss term based on Schwarz's theorem. In addition, to ensure the distribution similarity between the solution of the main branch and the predicted first-order partial derivatives of sub-branches, we set a soft constraint using the Kullback–Leibler (KL) divergence in the loss function. This paper's contributions are summarized as follows:

- We propose a multi-branch PINN. The main branch approximates the solution, and the sub-branches predict the first-order partial derivatives. This design decouples PDEs to alleviate the loss conflict in PINNs during training, without introducing extra redundant structures.
- We reconstruct the loss function. The partial derivatives predicted by the sub-branch are used to compute the residuals of the governing equations. Schwarz's theorem and the KL divergence are incorporated to constrain the continuity of partial derivatives and the consistency of the solution distribution.
- We conduct comprehensive experiments to evaluate the performance of the proposed method, in comparison with numerical methods and various PINN variants. The sensitivity analysis and ablation experiments are conducted to verify the effectiveness of the model inference.

This paper is organized as follows: Section 2 introduces the relevant concepts of PDEs, as well as the standard PINN (SPINN). Section 3 presents the proposed PDGM-PINN. In Section 4, we detail the experiment PDEs and results. Section 5 provides the limitations and directions for future work. In Section 6, we summarize the main conclusions.

2. Related Work

A general PDE form [35] for a function $u(x, y)$ is as follows:

$$\begin{cases} F(x, y, u) = g(x, y), (x, y) \in \Omega, \\ B(x, y) = f(x, y), (x, y) \in \partial\Omega, \end{cases} \quad (1)$$

where x, y are the independent variables, and u is the unknown solution. F and B denote the differential operators. $g(x, y)$ and $f(x, y)$ are known functions.

Physical phenomena, such as temperature and fluid velocity fields, generally exhibit continuous behavior without instantaneous derivative discontinuities; therefore, we assume that the mixed partial derivatives of $u(x, y)$ are continuous. According to Schwarz's theorem [36], the following relation holds true.

$$\frac{\partial^2 u}{\partial x \partial y} = \frac{\partial^2 u}{\partial y \partial x}. \quad (2)$$

Here, Equation (2) serves to guide the update direction of the first-order derivatives within the loss function.

The SPINN mechanism is the same as that of standard neural networks [9], consisting of an input layer, an output layer, and hidden layers, and learning is achieved through error backpropagation. Unlike traditional NNs, where the loss function measures the residual between predictions and label values, the loss function in PINNs incorporate residuals, which are obtained by predicted $\hat{u}$ and automatically differentiated $\hat{u}_x, \hat{u}_y$ of the governing equations and boundary conditions as soft constraints. The loss terms are shown as follows:

$$\mathcal{L}_r = \sum_{k=1}^{N_r} [\mathcal{F}(\hat{u}) - g(x, y)]^2, \quad (3)$$

$$\mathcal{L}_b = \sum_{k=1}^{N_b} [\mathcal{B}(\hat{u}) - f(x, y)]^2, \quad (4)$$

$$\mathcal{L}_t = \mathcal{L}_r + \mathcal{L}_b, \quad (5)$$

where $\mathcal{L}_r$ and $\mathcal{L}_b$ are the PDE loss term and boundary condition loss term, respectively. $\mathcal{L}_t$ is the total loss of PINNs. $\mathcal{F}(\cdot)$ and $\mathcal{B}(\cdot)$ are differential operators composed of automatic differentiation operations of PINNs and fundamental mathematical operations.

Although specific forms of the loss functions are different, the ultimate goal of PINNs is to asymptotically approximate the real solution u with the NN output $\hat{u}$. Meanwhile, both first-order and second-order derivatives of $\hat{u}$ satisfy the governing equations and the boundary conditions. However, the NNs were originally developed to learn network weights and biases through large-scale data training, rather than to solve mathematical or physical equations. Consequently, merely modifying the loss function is insufficient to achieve satisfactory performance in solving PDE tasks. In recent years, researchers have recognized the importance of partial derivatives and proposed structural improvements to PINN variants, most of which focus on gradient optimization. GIB-PINNs [23] incorporate the gradient variation in boundary conditions into the loss function of PINNs, improving the anti-noise ability and accuracy of PINNs. An adaptive gradient descent algorithm [24] is proposed to rebalance the magnitude difference in loss gradients, replacing the traditional Adam optimizer. In particular, the IPD-PINN [37] introduces a structure with multiple subnets that independently predicts partial derivatives, and a dedicated loss term is designed to enforce the residual derivatives to be zero, ensuring undisturbed learning of partial derivatives. However, multiple subnets increase the redundancy of the IPD-PINN, and independent learning fails to fully consider the continuity of the solution. In addition, recent PGINNs [34] use graph neural networks to analyze the issue of PINNs for solving low-dimensional PDEs, where the computational efficiency and accuracy are still lower than classical numerical methods, emphasizing the significance of optimizing PINNs in low-dimensional cases. To decouple the convergence issues in PINNs and analyze the performance disparity between numerical methods and PINNs in low-dimensional cases, we propose PDGM-PINN. Schwarz's theorem and KL divergence are used to guide the consistent convergence between partial derivatives and the solution, and the number of branch neurons can be less than other variant PINNs.

3. Methodology

3.1. Architecture Design

The architecture of the PDGM-PINN comprises three branches. The main branch B_m is used to predict the solution u , and the two sub-branches B_x and B_y are dedicated to predict the u_x and u_y . As shown in Figure 1, main branch and partial derivative guided branches are constructed on shared layers, where the linear layer is used to guide the consistent convergence of all branches through the backpropagation of errors in NNs. The main branch

fuses three branches to output the predicted solution. Generally, approximating first-order derivatives is easier than solution function for NNs because first-order derivatives are directly related to PDEs without differentiation. Meanwhile, as the first-order derivatives are approximated by sub-branch networks, we can obtain second-order derivatives by NN automatic differentiation of first-order in sub-branches, shortening the differentiation path to reduce errors [38]. More accurate first-order partial derivatives serve as feature inputs to the main branch, providing superior reference information for approximating the solution. The main branch is not required to synchronously predict partial derivatives, which guides the main branch to improve its solution performance.

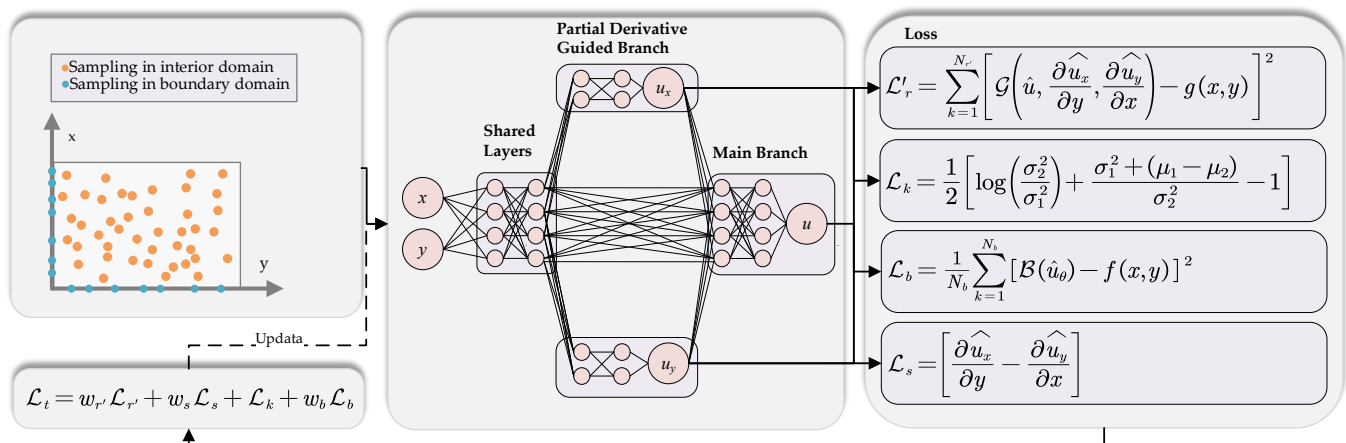


Figure 1. The framework of the PDGM-PINN.

3.2. Loss Function Improvement

To leverage the decoupling advantage of architecture, the loss function of the PDGM-PINN is fully reconstructed. Since the sub-branch is designed to predict partial derivatives, additional loss conflicts could be avoided if continuity of solution is constrained during the branch learning. Subsequently, to enhance the consistency of convergence directions, it is necessary to evaluate the discrepancy between the approximated first-order derivatives and the approximated PDE solutions. Furthermore, all predicted values are reintegrated to compute the residual of the governing equation in the interior domain and output the final approximate solution of the PDEs. Thus, as illustrated in Figure 1, the loss function is reconstructed into four terms, with only the boundary loss term being retained from the SPINN.

To address the learning continuity issue in sub-branches, we use Schwarz's theorem as loss constraints that the mixed partial derivatives of each sub-branch at the same point are equal.

$$\mathcal{L}_s = \frac{\partial \hat{u}_x}{\partial y} - \frac{\partial \hat{u}_y}{\partial x}, \quad (6)$$

where $\partial \hat{u}_x / \partial y$ and $\partial \hat{u}_y / \partial x$ are obtained by the automatic differentiation in sub-branches B_x and B_y .

SPINNs obtain first-order and second-order partial derivatives of u by automation differentiation as shown in Equation (3). In contrast, this approach is unsuitable for the architecture of PDGM-PINNs, as it fails to evaluate the approximation quality of the first-order derivatives and the solution in different branches. Therefore, we design a new PDE loss term, $\mathcal{L}'_{r'}$, to replace Equation (3).

$$\mathcal{L}'_{r'} = \sum_{k=1}^{N_{r'}} \left[\mathcal{G} \left(\hat{u}, \frac{\partial \hat{u}_x}{\partial y}, \frac{\partial \hat{u}_y}{\partial x} \right) - g(x, y) \right]^2, \quad (7)$$

where $\mathcal{G}(\cdot)$ denotes the operator that consists of automatic differentiation operations with respect to u_x and u_y , as well as fundamental mathematical operations. The structure of Equation (7) is consistent with the governing PDEs, while the operations involving the solution function are performed with the main branch output u . This differs from Equation (3) in that all terms are derived from the predicted solution and its partial derivatives computed via automatic differentiation.

Another challenge arises when the governing PDEs do not consist of the solution function. The sub-branches are not effectively constrained as it does not approximate $\hat{u}$, which may lead to additional potential conflicts with the main branch during the approximation of partial derivatives. To establish the connection between the sub-branch and the main branch, we introduce the KL divergence as the soft constraint between the main branch and sub-branches. As a directional metric for data distribution, the KL divergence softly constrains the governing equation residuals of the predicted partial derivatives to be consistent with the main branch, which ensures directional consistency during the approximation process and avoids introducing additional loss convergence conflicts. As shown in Equation (12), $\mathcal{L}_k$ represents the KL divergence computed from the residuals of the governing equations corresponding to the main branch and the sub-branches under discrete condition. We assume that residuals of the governing equation in the main branch and sub-branches follow Gaussian distributions [39] $P \sim \mathcal{N}(\mu_1, \sigma_1)$ and $Q \sim \mathcal{N}(\mu_2, \sigma_2)$. Therefore, their probability density functions (PDFs) can be given as follows:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right), \quad (8)$$

$$q(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right), \quad (9)$$

where μ_1 and μ_2 denote the mean of governing equation residual in the main branch and sub-branches. σ_1 and σ_2 denote the variance of governing equation residual in the main branch and sub-branches. According to the definition of the KL divergence:

$$D_{KL}(P||Q) = \int_{-\infty}^{+\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx, \quad (10)$$

where the integral form is adopted, as the solution function is theoretically continuous. Then, substituting Equations (8) and (9) into Equation (10), we obtain Equation (11).

$$D_{KL}(P||Q) = \frac{1}{2} \left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) + \frac{\sigma_1^2 + (\mu_1 - \mu_2)^2}{\sigma_2^2} - 1 \right], \quad (11)$$

where $D_{KL}(P||Q) = 0$ when $P \sim \mathcal{N}(\mu_1, \sigma_1)$ and $Q \sim \mathcal{N}(\mu_2, \sigma_2)$ share exactly the same distribution, which is consistent with the characteristics of the loss term. Therefore, we define Equation (10) as the loss term $\mathcal{L}_k$.

$$\mathcal{L}_k = D_{KL}(P||Q). \quad (12)$$

Finally, the total loss $\mathcal{L}_t$ is obtained as follows:

$$\mathcal{L}_t = \omega_{r'} \mathcal{L}_{r'} + \omega_s \mathcal{L}_s + \omega_k \mathcal{L}_k + \omega_b \mathcal{L}_b, \quad (13)$$

where $\omega_{r'}$, ω_s , ω_k and ω_b are weights of loss term. Notably, as the physical significance of $\mathcal{L}_s$ and $\mathcal{L}_k$ are distinct from other physical terms, we set $\omega_s = 1$ and $\omega_k = 1$. $\omega_{r'}$ and ω_b are chosen by sensitive experiment.

4. Experiments

Experiments were conducted on seven PDEs to comprehensively evaluate the performance of the PDGM-PINN on typical physical problems. Three methods were used as baselines for comparison: the SPINN represents the general baseline, the IPD-PINN as the partial derivative PINN variant baseline, and the relaxation method serves as the numerical baseline.

4.1. Setup

To ensure a fair comparison among the PINNs, the hyperparameter settings and sampling strategies in the SPINN, IPD-PINN, and PDGM-PINN were kept the same. All PINNs employed three hidden layers with 30 neurons per layer, adopt the hyperbolic tangent (Tanh) as activation function, and set a learning rate of 0.001 with Adam optimizer. It should be noted that the PDGM-PINN used one hidden layer in shared layers, both main branch and sub-branch are constructed by two hidden layers with 10 neurons, and the $\omega_{r'}$ and ω_b were all set to 50,000. The remaining configurations are summarized in Table 1. In addition, the relaxation method, as a numerical baseline, used a relaxation coefficient ω_{rel} to control the convergence process.

Table 1. Model hyperparameter settings for different PDEs. Here, ‘-’ denotes that the method is not applicable in this equation. N_r , N_b and N_t are the number of sampling points on the internal domain, boundary, and test set. The HCD equation denotes homogeneous convection diffusion, and the ICD equation is inhomogeneous convection diffusion.

PDEs	Epochs	Sampling Points			ω_{rel}
		N_r	N_b	N_t	
Laplace equation	8×10^3	2.5×10^3	1×10^2	5×10^3	1.9
Poisson equation	1×10^5	2.5×10^3	1×10^2	5×10^3	1.9
HCD equation	2×10^5	1×10^3	1×10^2	5×10^3	1.9
ICD equation	2×10^5	1×10^3	1×10^2	5×10^3	1.9
Wave equation	1×10^5	2×10^3	2×10^2	5×10^3	1.0
Biharmonic equation	1×10^5	2×10^3	2×10^2	5×10^3	-
High-frequency equation	1×10^5	2×10^3	2×10^2	5×10^3	1.0

In each experiment, the optimal value of ω_{rel} is searched within the interval by a step size of 0.1. Furthermore, the grid size was uniformly set to 80×80 across all experiments. To facilitate comparison, the number of testing points was fixed at 5000. The second-order derivatives were approximated using the central difference scheme, while the first-order derivatives were computed using the upwind scheme. To ensure reproducibility and reliability, the results of all PINN models are averaged over three independent runs. All models were conducted with PyTorch 2.5.1 on an NVIDIA RTX 3060 GPU (NVIDIA Corporation, Santa Clara, CA, USA).

4.2. Evaluation Metrics

The accuracy of the solutions was evaluated using the MSE and L2-Error metrics. Our experiments focus on forward problems. To facilitate observation, we monitor the variation

in evaluation metrics between the approximate and exact solutions during the solution process, and the results are as shown in Table 2.

$$MSE = \frac{1}{N} \sum_{i=1}^N (u_i - \hat{u})^2, \quad (14)$$

$$L_2 - Error = \frac{\|u - \hat{u}\|_2}{\|u\|_2}, \quad (15)$$

where $\|\cdot\|$ denotes the L_2 norm.

Table 2. MSE and L_2 -Error of different equations. Here, ‘-’ denotes that the method is not applicable in this equation. The HCD equation denotes homogeneous convection diffusion, and the ICD equation is inhomogeneous convection diffusion.

PDEs	SPINN		IPD-PINN		PDGM-PINN		Relaxation	
	MSE	L_2 -Error	MSE	L_2 -Error	MSE	L_2 -Error	MSE	L_2 -Error
Laplace equation	8.80×10^{-5}	2.69×10^{-2}	5.10×10^{-5}	2.02×10^{-2}	9.00×10^{-6}	8.10×10^{-3}	1.00×10^{-6}	1.02×10^{-4}
Poisson equation	2.52×10^{-3}	8.22×10^{-2}	2.54×10^{-3}	8.51×10^{-2}	5.00×10^{-6}	2.92×10^{-3}	1.00×10^{-6}	1.73×10^{-4}
HCD equation	4.51×10^{-1}	9.68×10^{-1}	3.86×10^{-1}	9.03×10^{-1}	3.57×10^{-1}	8.70×10^{-1}	2.99×10^{-1}	7.97×10^{-1}
ICD equation	2.35	1.19×10^{-1}	2.23	1.15×10^{-1}	1.99	1.12×10^{-1}	31.81	4.48×10^{-1}
Wave equation	2.00×10^{-6}	2.88×10^{-3}	1.00×10^{-6}	2.10×10^{-3}	1.00×10^{-6}	1.61×10^{-3}	8.34×10^{-2}	5.16×10^{-1}
Biharmonic equation	4.29×10^{-2}	4.23×10^{-1}	2.30×10^{-2}	3.22×10^{-1}	1.27×10^{-2}	2.51×10^{-1}	-	-
High-frequency equation	2.14×10^{-1}	1.14	2.11×10^{-1}	1.01	1.53×10^{-1}	7.63×10^{-1}	2.59×10^{-1}	4.78×10^{-1}

4.3. Numerical Experiments

4.3.1. Laplace Equation

The Laplace equation is a classical equation that characterizes the distribution of potential fields. We consider a source-free physical field, in which one boundary is assigned a sinusoidal distribution while all other boundaries are fixed at zero. The equation is given as follows:

$$\begin{cases} u_{xx} + u_{yy} = 0, x \in [0, 1], y \in [0, 1], \\ u(1, y) = \sin(\pi y), \\ u(0, y) = u(x, 0) = u(x, 1) = 0. \end{cases} \quad (16)$$

Here, the exact solution is as follows:

$$u(x, y) = \frac{e^{\pi x} - e^{-\pi x}}{e^{\pi} - e^{-\pi}} \sin(\pi y), x \in [0, 1], y \in [0, 1]. \quad (17)$$

As illustrated in Figure 2, the color intensity in subplots Figure 2a–c represent the magnitude of the relative error, which is obtained by dividing the absolute errors of the three by the maximum value. The relaxation method exhibits the lowest error and achieves highly accurate results across the entire computational domain. Among the three PINN models, the PDGM-PINN demonstrates the smallest error with a uniformly distributed error field, showing no apparent light-colored regions. According to Figure 2f, the convergence rate of the PDGM-PINN decreases more slowly during the initial training stage. However, after approximately 4000 epochs, its accuracy surpasses that of the SPINN and IPD-PINN. This

indicates that the PDGM-PINN requires an initial phase to balance multiple loss terms, after which it achieves superior solution performance for PDEs.

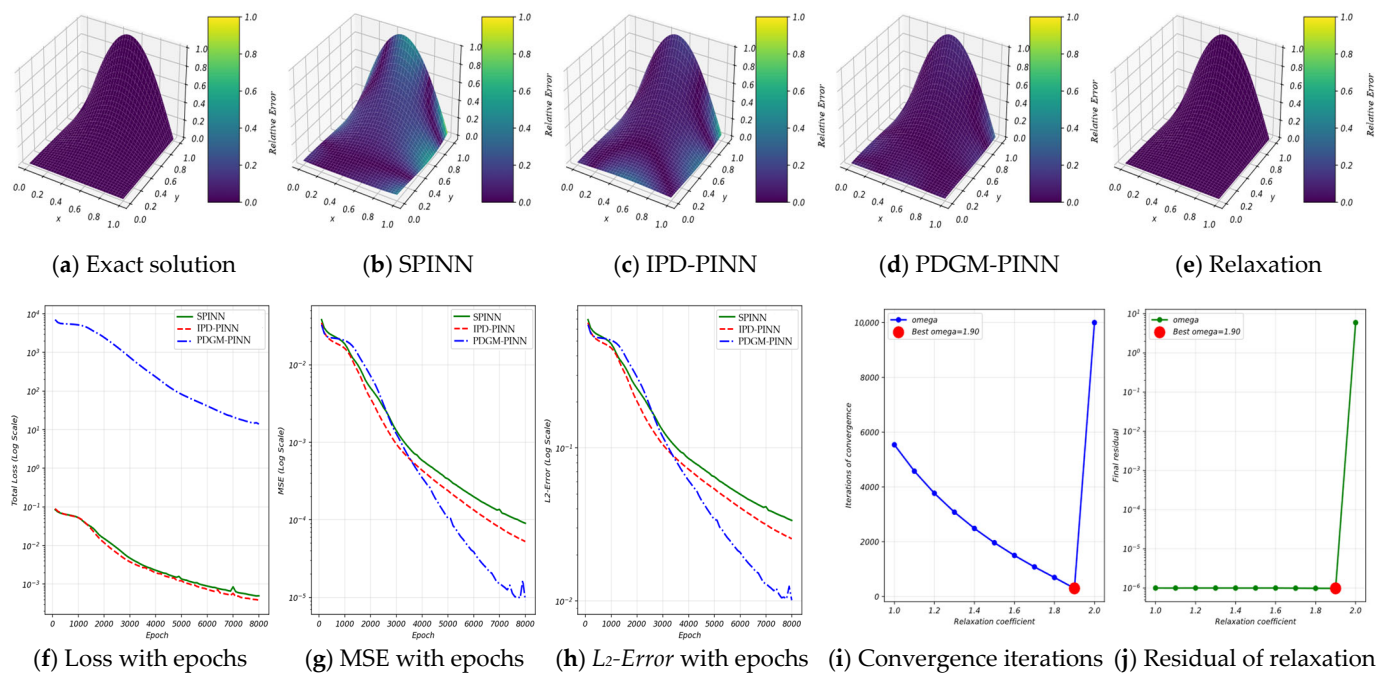


Figure 2. The Laplace equation solutions: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) relaxation solution, (f) the loss with epoch, (g) MSE with epochs, (h) L2-Error with epochs, (i) convergence iterations with relaxation coefficient, and (j) residual with relaxation coefficient.

4.3.2. Poisson Equation

The Poisson equation extends the Laplace equation by introducing a source term. The experimental equation is shown as follows:

$$\begin{cases} u_{xx} + u_{yy} = -2\pi^2 \sin(\pi x) \sin(\pi y), & x \in [0, 1], y \in [0, 1], \\ u(0, y) = u(1, y) = u(x, 0) = u(x, 1) = 0. \end{cases} \quad (18)$$

Here, the exact solution is as follows:

$$u(x, y) = \sin(\pi x) \sin(\pi y), \quad x \in [0, 1], y \in [0, 1]. \quad (19)$$

In Figure 3, the relaxation method also obtained the solution with the lowest error. All methods exhibited an increase in errors in comparison to the Laplace equation. This observation indicates that the introduction of a source term increases the complexity of the PDEs, raising the computational difficulty. It can be observed that the PDGM-PINN still maintains superior performance, particularly in capturing the distinct fitting behavior between the boundary and the central regions, achieving a more accurate and balanced representation of the solution. In addition, as shown in Figure 4g,h, both the MSE and L2-Error of the SPINN and IPD-PINN exhibit a trend of initial increase followed by a decrease in the early training stage, whereas the PDGM-PINN remains relatively stable. As training continues, the PDE residual loss term becomes dominant, resulting in an overall reduction in errors.

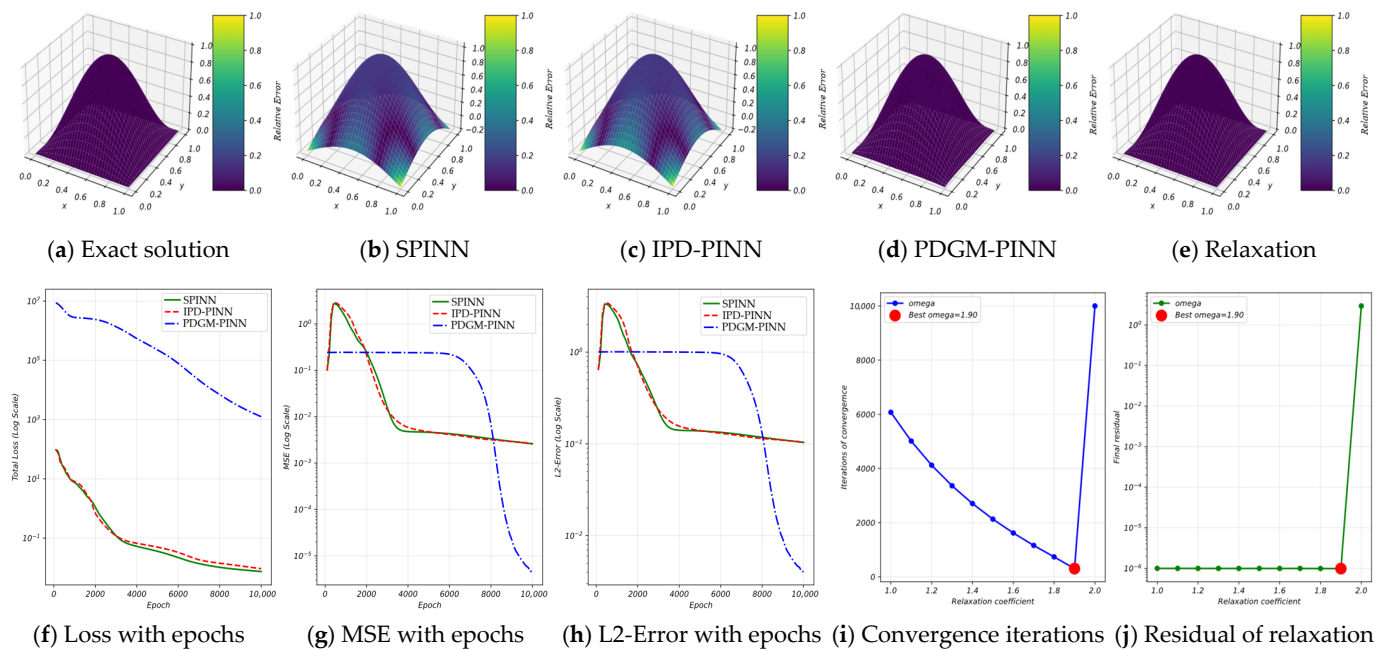


Figure 3. The Poisson equation solutions: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) relaxation solution, (f) the loss with epoch, (g) MSE with epochs, (h) L2-Error with epochs, (i) convergence iterations with relaxation coefficient, and (j) residual with relaxation coefficient.

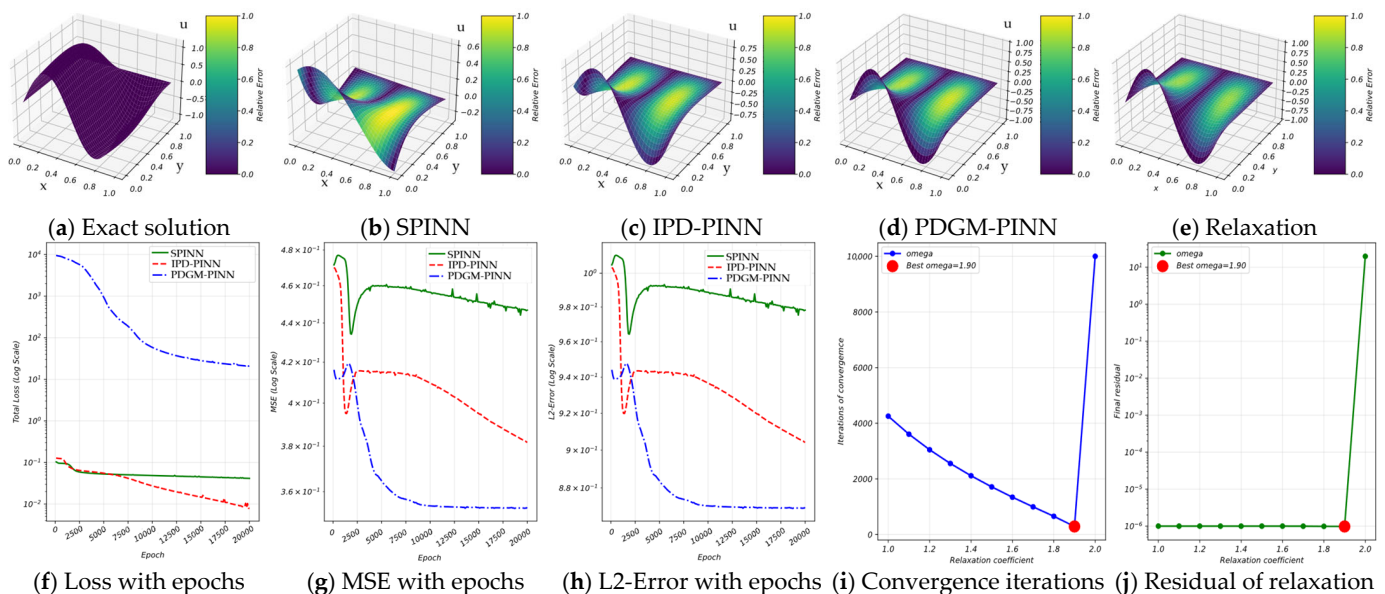


Figure 4. The homogeneous convection diffusion equation solutions: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) relaxation solution, (f) the loss with epoch, (g) MSE with epochs, (h) L2-Error with epochs, (i) convergence iterations with relaxation coefficient, and (j) residual with relaxation coefficient.

4.3.3. Homogeneous Convection Diffusion

The homogeneous convection diffusion equation is used to describe the variation in physical quantities in the flow process. The equation is as follows:

$$\begin{cases} 4u_x = u_{xx} + u_{yy}, x \in [0, 1], y \in [0, 1], \\ u(0, y) = \sin(2\pi y), \\ u(1, y) = u(x, 0) = u(x, 1) = 0. \end{cases} \quad (20)$$

Here, the exact solution is as follows:

$$u(x, y) = \frac{e^{(2-a)(x-1)} - e^{(2+a)(x-1)}}{e^{-(2-a)} - e^{-(2+a)}} \cdot \sin(2\pi y), x \in [0, 1], y \in [0, 1], \quad (21)$$

where $a = 2\sqrt{1 + \pi^2}$.

As shown in Figure 4, the errors of all models were primarily concentrated in the central region, while the results near the boundaries remain relatively accurate. Compared with the solution of the Poisson equation, the overall errors of all models increase significantly, and the relaxation method still exhibited the highest accuracy. The PDGM-PINN continued to demonstrate superior solution performance, exhibiting an enhanced ability to capture the transition trend from the boundaries toward the domain center compared with the standard PINN and IPD-PINN. Meanwhile, as shown in Figure 4g,h, both the SPINN and IPD-PINN exhibit a brief decrease followed by an increase in their MSE and L_2 -Error curves, while their total losses continue to decrease, as illustrated in Figure 4f. In contrast, the PDGM-PINN exhibited less fluctuations in MSE and L_2 -Error, indicating that under mixed-derivative guidance $\mathcal{L}_s$, it can effectively balance the boundary and PDE losses throughout the training process.

4.3.4. Inhomogeneous Convection Diffusion

The inhomogeneous convection diffusion equation has a source term, and this case is as follows:

$$\begin{cases} 2u_x = u_{xx} + u_{yy} + 10e^x(\pi^2 + 1)\sin(\pi y), x \in [0, 1], y \in [0, 1], \\ u(0, y) = 10\sin(\pi y), \\ u(1, y) = 10e\sin(\pi y), \\ u(x, 0) = u(x, 1) = 0. \end{cases} \quad (22)$$

Here, the exact solution is as follows:

$$u(x, y) = 10e^x \sin(\pi y), x \in [0, 1], y \in [0, 1]. \quad (23)$$

In Figure 5, the maximum error occurs in the relaxation method, primarily concentrated around the heat source region. The error distributions of the PINN-based models are more dispersed, among which the PDGM-PINN achieves the best overall performance. In particular, its errors are smaller than those of other PINNs in the transition region from the boundary toward the heat source. Furthermore, it can be observed that along the upper boundary, among all PINN variants, only the PDGM-PINN produces fitted values exceeding 25 at the extrema, which is consistent with the analytical solution, demonstrating superior accuracy.

4.3.5. Wave Equation

To explore the capability of the PDGM-PINN in solving nonlinear transient problems, the wave equation was selected for experimentation. The wave equation primarily describes various wave phenomena occurring in nature, which is as follows:

$$\begin{cases} u_{tt} - u_{xx} = 0, t \in [0, 1], x \in [0, 1], \\ u(0, x) = \sin(\pi x), \\ u(1, x) = -\sin(\pi x), \\ u(t, 0) = \sin(\pi t), \\ u(t, 1) = -\sin(\pi t). \end{cases} \quad (24)$$

Here, the exact solution is as follows:

$$u(t, x) = \sin[\pi(t + x)], t \in [0, 1], x \in [0, 1]. \quad (25)$$

As shown in Figure 6, the relaxation method failed to converge, reaching the maximum iteration limit of 10,000 in each coefficient. According to Table 1, the PDGM-PINN and IPD-PINN exhibit the same MSE, as the minimum precision threshold was set at 1×10^{-6} . Therefore, we compare the L_2 -Error, from which it can be observed that the PDGM-PINN achieves a 31.25% improvement in accuracy over the IPD-PINN. Additionally, based on Figure 6g,h, we can observe that after the epoch reaches 8000, the SPINN and IPD-PINN exhibit significant oscillations. In contrast, the PDGM-PINN still maintains a relatively stable downward trend, with only one oscillation.

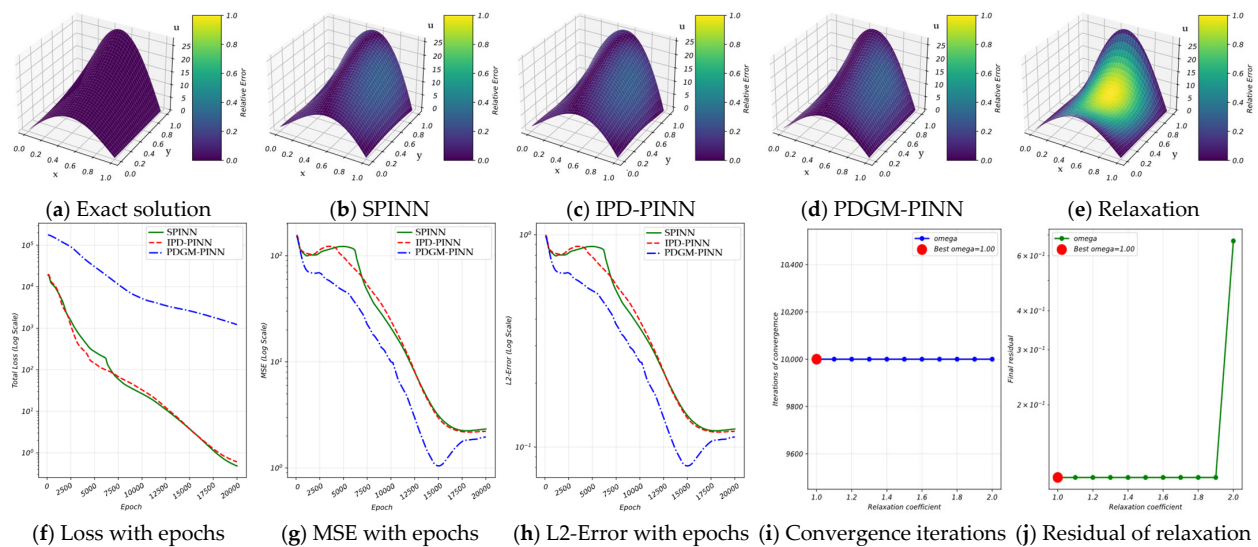


Figure 5. The inhomogeneous equation solutions: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) relaxation solution, (f) the loss with epoch, (g) MSE with epochs, (h) L_2 -Error with epochs, (i) convergence iterations with relaxation coefficient, and (j) residual with relaxation coefficient.

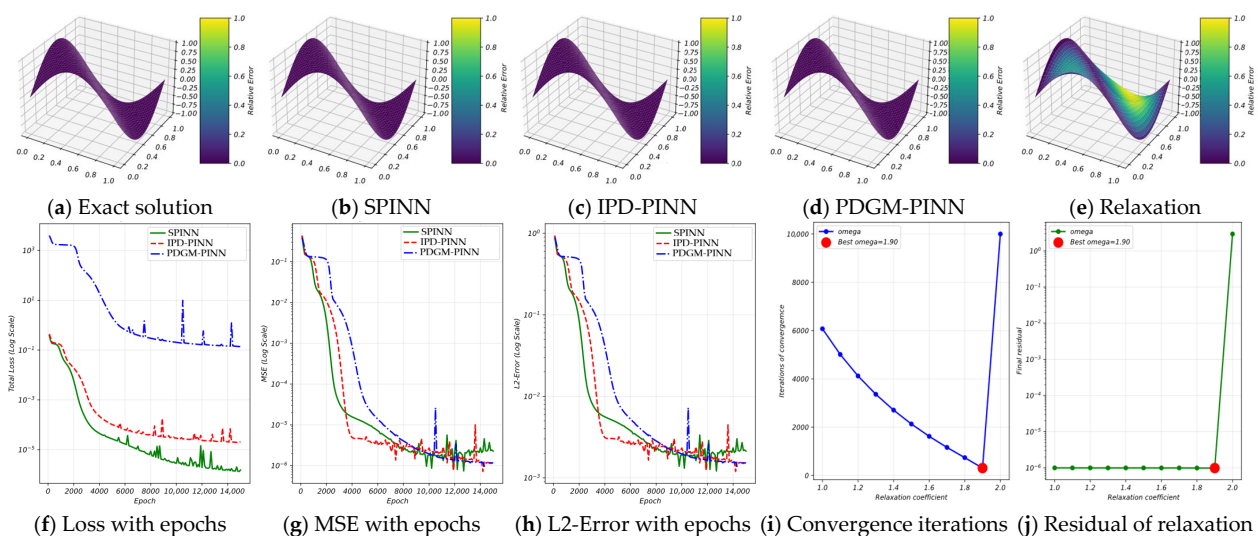


Figure 6. The wave convection diffusion equation solutions: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) relaxation solution, (f) the loss with epoch, (g) MSE with epochs, (h) L_2 -Error with epochs, (i) convergence iterations with relaxation coefficient, and (j) residual with relaxation coefficient.

4.3.6. Biharmonic Equation

To further evaluate the potential of the PDGM-PINN in solving high-order PDEs, an exploratory experiment was conducted on the biharmonic equation, which is a typical fourth-order PDE as follows:

$$\begin{cases} \frac{\partial^4 u}{\partial x^4} + 2\frac{\partial^4 u}{\partial x^2 \partial y^2} + \frac{\partial^4 u}{\partial y^4} = 24(x + \sin(\pi y)), x \in [0, 1], y \in [0, 1], \\ u(1, y) = y - y^2 + \sin(\pi y), \\ u(x, 0) = x^3 - x^4, \\ u(0, y) = u(x, 1) = 0. \end{cases} \quad (26)$$

Here, the exact solution is as follows:

$$u(x, y) = \left[(x^3(1-x) + xy) \cdot (1-y) \right] + x^2 \sin(\pi y), x \in [0, 1], y \in [0, 1]. \quad (27)$$

For higher-order equations, the traditional relaxation method is difficult to converge. Therefore, we compared three PINNs. As shown in Figure 7, the SPINN performs poorly in solving the biharmonic equation, with the largest errors occurring near the source. These central errors further hinder adequate convergence near the boundaries. In contrast, the IPD-PINN achieves better results but still exhibits noticeable deviations both at the source and along the boundaries. The PDGM-PINN produces accurate results across the entire domain, closely matching the analytical solution. Interestingly, during the training process, the MSE and L2-Error of the SPINN and IPD-PINN initially increased and then decreased, whereas those of the PDGM-PINN decreased continuously, as shown in Figure 7f,g. However, the total loss of all three PINN variants exhibited a monotonic decrease. This observation indicates that the partial derivative guided loss term in the PDGM-PINN provides a correct convergence direction during training, improving both the accuracy and efficiency in solving high-order PDEs, which demonstrates the strong potential of the PDGM-PINN in solving high-order PDEs.

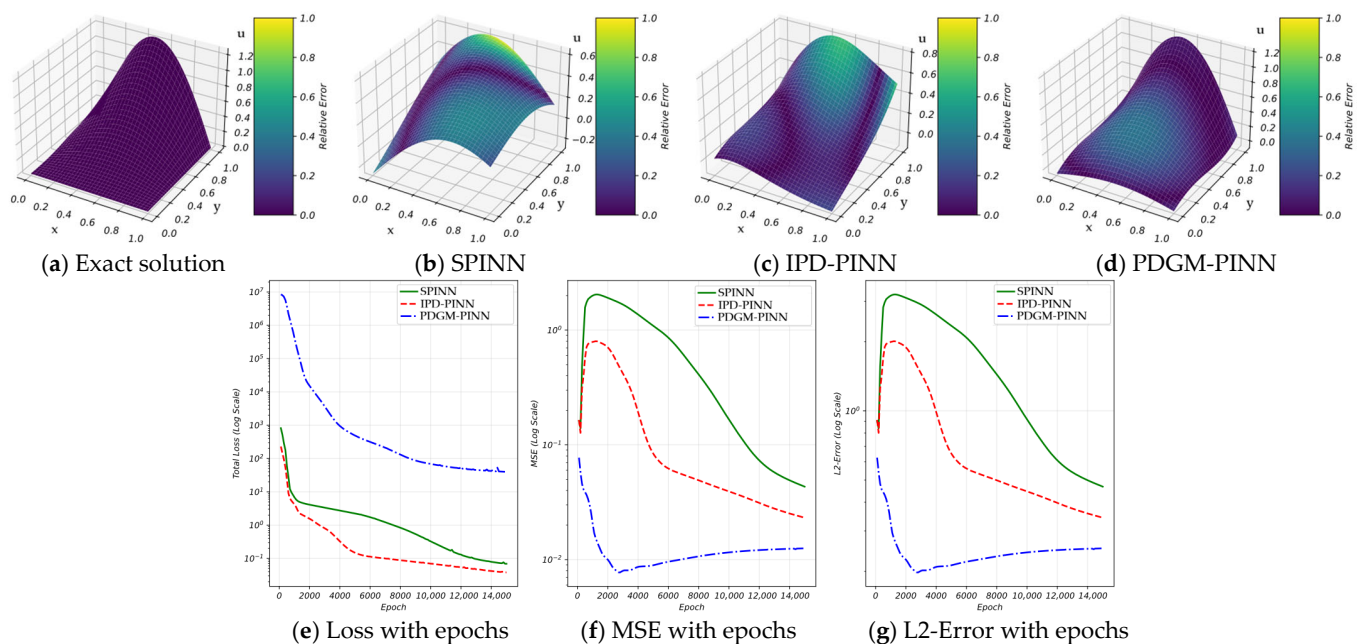


Figure 7. The biharmonic convection diffusion equation solutions: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) the loss with epoch, (f) MSE with epochs, and (g) L2-Error with epochs.

4.3.7. High-Frequency Equation

PINNs generally perform unsatisfactorily on high-frequency PDEs due to spectral bias during NN training. To validate the effectiveness of the PDGM-PINN for high-frequency problems, we performed an experiment on the following spatiotemporally oscillatory equation:

$$\begin{cases} \frac{\partial u}{\partial t} + 0.01 \frac{\partial u}{\partial^2 x} = f(x, t), x \in [0, 1], y \in [0, 1], \\ u(x, 1) = \frac{1}{e} \sin(12\pi x) \cos(8\pi), \\ u(x, 0) = \sin(12\pi x), \\ u(0, t) = u(1, t) = 0. \end{cases} \quad (28)$$

Here, $f(x, t) = e^{-t} \sin(12\pi x) [(1.44\pi^2 - 1)(\cos(8\pi t) - 8\pi \sin(8\pi t))]$. The exact solution is as follows:

$$u(x, t) = e^{-t} \sin(12\pi x) \cos(8\pi x), x \in [0, 1], y \in [0, 1]. \quad (29)$$

As shown in Figure 8b–d, the errors of all the PINNs for PDEs exhibit obvious deviation, which clearly demonstrate the limitations of PINNs in solving high-frequency PDEs. The loss mechanism of PINNs tends to prioritize the global average accuracy across the domain, leading to insufficient attention when capturing the detailed evolutionary trend of the solution. In contrast, the relaxation method achieves good stability and accuracy. During the solution process, all the PINNs show obvious oscillations. As illustrated in Figure 8g,h, the PDGM-PINN exhibits the most stable solution among all the PINNs, demonstrating that the derivative-guided mechanism alleviates loss conflicts during the training process.

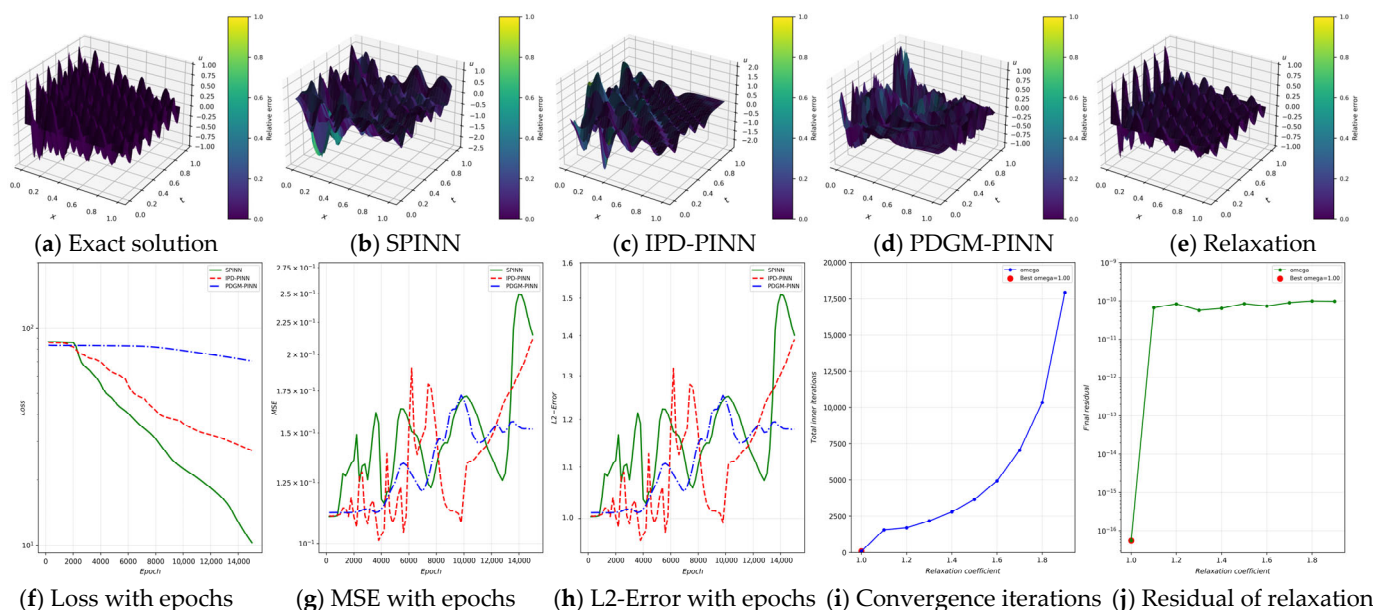


Figure 8. The high-frequency equation: (a) exact solution, (b) SPINN solution, (c) IPD-PINN solution, (d) PDGM-PINN solution, (e) relaxation solution, (f) the loss with epoch, (g) MSE with epochs, (h) L2-Error with epochs, (i) convergence iterations with relaxation coefficient, and (j) residual with relaxation coefficient.

4.4. Ablation Experiments

The performance enhancement of the PDGM-PINN benefits from the adaptive loss function design of the architecture. To further evaluate the influence of partial derivative continuity and KL divergence loss terms, we disabled $\mathcal{L}_s$ in #1 case and $\mathcal{L}_k$ in #2 case, and

conducted ablation experiments on the Poisson equation and homogeneous advection diffusion equation.

$$\mathcal{L}_{t(\#1)} = \omega_{r'} \mathcal{L}_{r'} + \omega_k \mathcal{L}_k + \omega_b \mathcal{L}_b, \quad (30)$$

$$\mathcal{L}_{t(\#2)} = \omega_{r'} \mathcal{L}_{r'} + \omega_s \mathcal{L}_s + \omega_b \mathcal{L}_b. \quad (31)$$

As shown in Table 3, the PDGM-PINN achieves comparable performance to #1 case, but differs from #2 case by several orders of magnitude, which validates the necessity of imposing constraints between the partial derivative branch and the solution distribution.

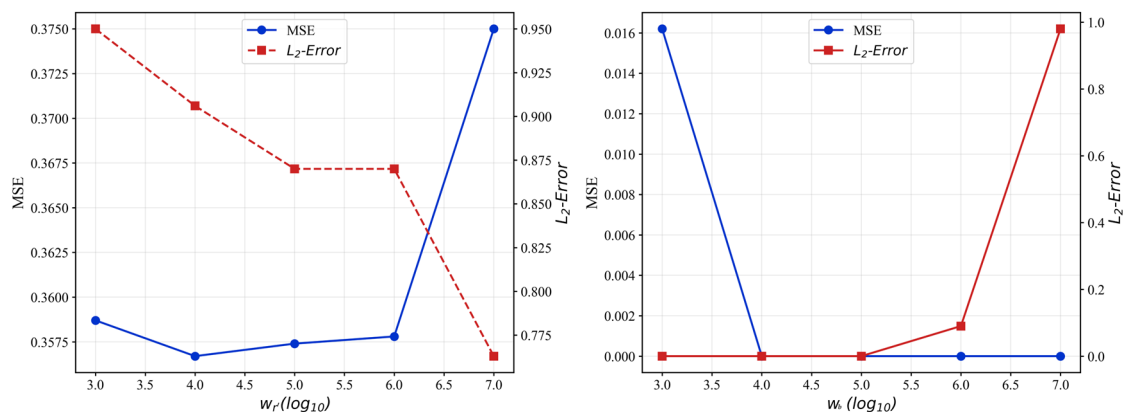
Table 3. The ablation experiment results of the PDGM-PINN.

PDEs	SPINN		PDGM-PINN		#1 Case		#2 Case	
	MSE	L ₂ -Error	MSE	L ₂ -Error	MSE	L ₂ -Error	MSE	L ₂ -Error
Laplace equation	2.52×10^{-3}	8.22×10^{-2}	5.00×10^{-6}	2.92×10^{-3}	6.00×10^{-6}	3.94×10^{-3}	7.01×10^{-2}	7.01×10^{-1}
HCD equation	4.51×10^{-1}	9.68×10^{-1}	3.57×10^{-1}	8.70×10^{-1}	5.87×10^{-1}	1.12	7.63×10^{-1}	1.25

4.5. Parameter Sensitivity Analysis

During the design of the PDGM-PINN, weight coefficients were introduced for loss terms to reweight the losses of different branches. To verify the sensitivity of parameters $\omega_{r'}$ and ω_b , experiments were carried out on the Poisson equation and the homogeneous convection diffusion equation.

As illustrated in Figure 9, the Poisson equation and the homogeneous convection diffusion equation exhibit different variation trends. Despite this discrepancy, a desirable trade-off between stability and accuracy can be obtained at 50,000.



(a) MSE and L2-Error with different weights in Poisson equation **(b)** MSE and L2-Error with different weights in HCD equation

Figure 9. Parameter sensitivity analysis results.

4.6. Complexity Analysis

To evaluate the computational cost of the PINNs, we calculate the number of trainable parameters. The number of trainable parameters of the network is determined by counting the weights and biases, which can be calculated by Equation (30). We adopt the same hidden layers and adjust the neurons in each hidden layer. As the PDGM-PINN has three branches, we set 10 neurons per branch to keep the total neurons the same as the SPINN. As listed in Table 4, the PDGM-PINN reduces the number of trainable parameters by 38.98% compared with the IPD-PINN, and by 9.28% compared with the SPINN. The performance

of the PDGM-PINN is improved due to partial derivative guided architecture proposed with fewer parameters.

$$P = \sum_{i=1}^{H+1} N_i N_{i-1} + N_i, \quad (32)$$

where N_i is the number of i -th layer neurons, and H is the number of layers.

Table 4. The trainable parameters of PINNs. Here, $(m_1 + m_2 + m_k) \times n$ in hidden layers denotes a network with n branches, where the i -th layer contains m_i trainable parameters. “-” indicates that the shared layers were not used in the SPINN; TTPs indicates the total trainable parameters.

PINNs	Input Layer	Hidden Layers	Shared Layers	Output Layer	TTPs
SPINN	90	$(930 + 930 + 930) \times 1$	-	31	2911
IPD-PINN	90	$(930 + 930) \times 2$	930	31	4771
PDGM-PINN	90	$(310 + 110) \times 3$	930 + 330	31	2641

5. Discussion

Overall, the PDGM-PINN provides a more efficient architecture. Compared with the IPD-PINN and SPINN, the PDGM-PINN decouples partial derivatives from the solution and facilitates a stable convergence process. Meanwhile, benefiting from the prediction of partial derivatives, the main branch for the approximate solution achieves higher accuracy with fewer neurons. The introduction of Schwarz’s theorem is necessary, which inspires the exploration of the underlying physical assumptions embedded in PDEs. The KL divergence reflects a data-driven philosophy, where both the solution and its first-order derivatives can be regarded as dependent variables to be predicted. A multi-branch PINN with reasonably designed loss functions can improve the performance in solving PDEs without redundant structures or parameters, which provides a useful reference for the optimal design of PINN variants.

However, we also verify that the performance of PINNs in solving PDEs is inferior to that of traditional numerical methods for low-dimensional problems, even though the PDGM-PINN narrows the performance gap. Meanwhile, results on the wave equation and biharmonic equation reveal the limitations of numerical methods in handling high-order and nonlinear problems. Generally, we remain confident in the advancement of PINNs. The PDGM-PINN demonstrates the potential of neural architecture improvements and offers a novel solution to alleviate convergence conflicts in PINNs.

In future work, we plan to explore adaptive weight multi-branch PINNs to avoid introducing extra hyperparameters. Additionally, the decoupling of governing equations and boundary conditions will be investigated to further boost the performance of PINNs in solving low-dimensional PDEs.

6. Conclusions

In this study, we proposed a novel PDGM-PINN, which improved architecture and redesigned loss function to decouple partial derivatives and solution during approximation convergence. The PDGM-PINN explicitly accounts for the continuity of mixed derivatives, incorporating Schwarz’s theorem into the loss function. Moreover, to alleviate the loss conflict, the governing equation loss term is reconstructed with a predicted partial derivative. To compare the performance of numerical methods and PINNs in solving low-dimensional PDEs, we conducted experiments on seven PDEs, including strong nonlinear, high-order and high-frequency conditions. Furthermore, to comprehensively validate the proposed PDGM-PINN model, we carried out ablation experiments, parameter sensitivity analysis, and complexity analysis. Our conclusions are as follows:

1. The PDGM-PINN achieves the best performance with the fewest number of trainable parameters. Compared with the SPINN and IPD-PINN, the performance of the PDGM-PINN is improved by at least 15.32% and 7.51%, respectively, in all PDE experiments.
2. The relaxation method exhibited the best performance for the Laplace, Poisson, homogeneous convection diffusion, and high-frequency equations. However, as PDE complexity increased, all PINN-based methods significantly outperformed the relaxation method.
3. The multi-branch architecture of the PINN is effective in alleviating conflicts during the convergence process, and a reasonable loss function design can avoid additional architectural redundancy, enabling the decoupling of partial derivatives and solutions in the approximation process.

This study emphasizes that the performance of PINNs in solving low-dimensional PDE problems still needs further optimization compared with traditional numerical methods, though many variant PINNs have been proposed for complex situations. We also propose that the architectural design can effectively improve the performance of PINNs in solving PDEs.

Author Contributions: Conceptualization, S.L. and N.K.; Methodology, S.L.; Validation, U.G.; Formal analysis, C.Y.; Investigation, N.K.; Writing—original draft, S.L.; Writing—review and editing, C.Y., R.G. and U.G.; Visualization, U.G.; Funding acquisition, R.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research and APC are funded by EU RRF within project No. 5.2.1.1.i.0/2/24/I/CFLA/003, academic career doctoral grant, ID 1081.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors would like to thank Balakayeva Gulnar from the Faculty of Artificial Intelligence and Information Technology, Al-Farabi Kazakh National University, for her constructive comments and support during the preparation of this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	Machine learning
PINN	Physics-informed neural network
SciML	Science machine learning
PDE	Partial differential equation
MLP	Multi-layer perceptron
NN	Neural network

References

1. Krishnapriyan, A.; Gholami, A.; Zhe, S.; Kirby, R.; Mahoney, M.W. Characterizing Possible Failure Modes in Physics-Informed Neural Networks. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 26548–26560.
2. Cheng, S.; Bocquet, M.; Ding, W.; Finn, T.S.; Fu, R.; Fu, J.; Guo, Y.; Johnson, E.; Li, S.; Liu, C.; et al. Machine Learning for Modelling Unstructured Grid Data in Computational Physics: A Review. *Inf. Fusion* **2025**, *123*, 103255. [[CrossRef](#)]
3. Coelho, C.; Costa, M.F.P.; Ferrás, L.L. Fractional Calculus Meets Neural Networks for Computer Vision: A Survey. *AI* **2024**, *5*, 1391–1426. [[CrossRef](#)]
4. Brociek, R.; Pleszczyński, M. Differential Transform Method and Neural Network for Solving Variational Calculus Problems. *Mathematics* **2024**, *12*, 2182. [[CrossRef](#)]
5. Hao, Z.; Yao, J.; Su, C.; Su, H.; Wang, Z.; Lu, F.; Xia, Z.; Zhang, Y.; Liu, S.; Lu, L.; et al. Pinnacle: A Comprehensive Benchmark of Physics-Informed Neural Networks for Solving Pdes. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 76721–76774.

6. Karniadakis, G.E.; Kevrekidis, I.G.; Lu, L.; Perdikaris, P.; Wang, S.; Yang, L. Physics-Informed Machine Learning. *Nat. Rev. Phys.* **2021**, *3*, 422–440. [\[CrossRef\]](#)
7. Zhang, G.; Yuan, G.; Cheng, D.; Liu, L.; Li, J.; Zhang, S. Disentangled Contrastive Learning for Fair Graph Representations. *Neural Netw.* **2025**, *181*, 106781. [\[CrossRef\]](#) [\[PubMed\]](#)
8. Fan, W.; Chen, X. Embedding Physics into Machine Learning: A Review of Physics Informed Neural Networks as Partial Differential Equation Forward Solvers. *Tsinghua Sci. Technol.* **2026**, *31*, 1326–1364. [\[CrossRef\]](#)
9. Raissi, M.; Perdikaris, P.; Karniadakis, G.E. Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *J. Comput. Phys.* **2019**, *378*, 686–707. [\[CrossRef\]](#)
10. Hu, Z.; Shukla, K.; Karniadakis, G.E.; Kawaguchi, K. Tackling the Curse of Dimensionality with Physics-Informed Neural Networks. *Neural Netw.* **2024**, *176*, 106369. [\[CrossRef\]](#)
11. Sharma, R.; Shankar, V. Accelerated Training of Physics-Informed Neural Networks (PINNs) Using Meshless Discretizations. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1034–1046.
12. Zhao, Y.; Li, D.; Wang, C.; Xi, H. An Innovative End-to-End PINN-Based Solution for Rapidly Simulating Homogeneous Heat Flow Problems: An Adaptive Universal Physics-Guided Auto-Solver. *Case Stud. Therm. Eng.* **2024**, *56*, 104277. [\[CrossRef\]](#)
13. Hu, H.; Qi, L.; Chao, X. Physics-Informed Neural Networks (PINN) for Computational Solid Mechanics: Numerical Frameworks and Applications. *Thin-Walled Struct.* **2024**, *205*, 112495. [\[CrossRef\]](#)
14. Zhang, L.; Li, K.; Wang, H.; Liu, D.; Ma, Z.; Zhang, Q.; Li, M.; Zhang, W.; Wang, C.; Zhao, X.; et al. MFC-PINN: A Method to Improve the Accuracy and Robustness of Acoustic Emission Source Planar Localization. *Measurement* **2024**, *235*, 114995. [\[CrossRef\]](#)
15. Wu, C.; Zhu, M.; Tan, Q.; Kartha, Y.; Lu, L. A Comprehensive Study of Non-Adaptive and Residual-Based Adaptive Sampling for Physics-Informed Neural Networks. *Comput. Methods Appl. Mech. Eng.* **2023**, *403*, 1–23. [\[CrossRef\]](#)
16. Zhang, H.; He, Y.-L.; Liu, D.; Hang, Q.; Yao, H.-M.; Xiang, D. Residual Resampling-Based Physics-Informed Neural Network for Neutron Diffusion Equations. *Nucl. Sci. Tech.* **2026**, *37*, 28. [\[CrossRef\]](#)
17. Li, H.; Zhang, Y.; Wu, Z.; Wang, Z.; Wu, T. An Importance Sampling Method for Generating Optimal Interpolation Points in Training Physics-Informed Neural Networks. *Mathematics* **2025**, *13*, 150. [\[CrossRef\]](#)
18. Roy, A.; Chatterjee, T.; Adhikari, S. A Physics-Informed Neural Network Enhanced Importance Sampling (PINN-IS) for Data-Free Reliability Analysis. *Probabilistic Eng. Mech.* **2024**, *78*, 103701. [\[CrossRef\]](#)
19. Feng, W.; Hu, M.; Huang, R.; Tang, C.; Huang, S.; Lv, J. DSAIS-PINN: Dynamic Seeds Allocation Importance Sampling for Physics-Informed Neural Networks. *Neurocomputing* **2025**, *648*, 130578. [\[CrossRef\]](#)
20. Monaco, S.; Apiletti, D. Training Physics-Informed Neural Networks: One Learning to Rule Them All? *Results Eng.* **2023**, *18*, 101023. [\[CrossRef\]](#)
21. Fernández de la Mata, F.; Gijón, A.; Molina-Solana, M.; Gómez-Romero, J. Physics-Informed Neural Networks for Data-Driven Simulation: Advantages, Limitations, and Opportunities. *Phys. A Stat. Mech. Its Appl.* **2023**, *610*, 128415. [\[CrossRef\]](#)
22. Kim, J.; Son, H. Causality-Aware Training of Physics-Informed Neural Networks for Solving Inverse Problems. *Mathematics* **2025**, *13*, 1057. [\[CrossRef\]](#)
23. Wang, Z.-P.; Guo, L.-L.; Zhang, L.-S.; Zhang, Z.-Y. The GIB-PINN: A New Gradient-Enhanced Physics-Informed Neural Networks for Solving Forward and Inverse Problems of Partial Differential Equations. *Eur. Phys. J. Plus* **2025**, *140*, 1–23. [\[CrossRef\]](#)
24. Li, X.; Liu, Y.; Liu, Z. Physics-Informed Neural Network Based on a New Adaptive Gradient Descent Algorithm for Solving Partial Differential Equations of Flow Problems. *Phys. Fluids* **2023**, *35*, 063608. [\[CrossRef\]](#)
25. Yu, J.; Lu, L.; Meng, X.; Karniadakis, G.E. Gradient-Enhanced Physics-Informed Neural Networks for Forward and Inverse PDE Problems. *Comput. Methods Appl. Mech. Eng.* **2022**, *393*, 114823. [\[CrossRef\]](#)
26. Wang, Y.; Qiu, X.; Pei, Q.; Wang, J.; Zhang, P.; Bai, X. FAMAW-PINN: A Physics-Informed Neural Network Integrating Adaptive Loss Weighting with Firefly-Inspired Adaptive Point Movement. *J. Comput. Phys.* **2025**, *542*, 114363. [\[CrossRef\]](#)
27. Li, S.; Lai, S. PINN Based on Multi-Scale Strategy for Solving Navier–Stokes Equation. *Results Appl. Math.* **2025**, *25*, 100526. [\[CrossRef\]](#)
28. Pan, C.; Li, C.; Liu, Y.; Zheng, Y.; Ye, H. SK-PINN: Accelerated Physics-Informed Deep Learning by Smoothing Kernel Gradients. *Comput. Methods Appl. Mech. Eng.* **2025**, *440*, 117956. [\[CrossRef\]](#)
29. Jiang, F.; Hou, X.; Ye, J.; Xia, M. Deeper-PINNs: Unlocking the Power of Deep Physics-Informed Neural Networks. *Appl. Soft Comput.* **2025**, *185*, 114048. [\[CrossRef\]](#)
30. Gao, Y.; Wang, B.; Lu, J.; Tian, Z. ML-PINN: A Memory-Efficient Physics-Informed Mamba-LSTM Network for Fast and Accurate PDE Solving. *Neurocomputing* **2025**, *655*, 131446. [\[CrossRef\]](#)
31. Grossmann, T.G.; Komorowska, U.J.; Latz, J.; Schönlieb, C.-B. Can Physics-Informed Neural Networks Beat the Finite Element Method? *IMA J. Appl. Math.* **2024**, *89*, 143–174. [\[CrossRef\]](#) [\[PubMed\]](#)
32. Du, H.; Liu, Z.; Zhu, P. Accurate Solution of High-Dimensional Partial Differential Equations: Research on Data-Physics Collaborative Modeling and Adaptive Sampling. *Appl. Soft Comput.* **2026**, *191*, 114472. [\[CrossRef\]](#)

33. Hu, Z.; Yang, Z.; Wang, Y.; Karniadakis, G.E.; Kawaguchi, K. Bias-Variance Trade-Off in Physics-Informed Neural Networks with Randomized Smoothing for High-Dimensional PDEs. *SIAM J. Sci. Comput.* **2025**, *47*, C846–C872. [[CrossRef](#)]
34. Chen, F.; Meng, Y.; Li, K.; Yang, C.; Dai, J. Physics-Graph-Informed Neural Networks (PGINNs): From Local Point-Wise Constraint to Global Nodal Association. *J. Comput. Phys.* **2026**, *551*, 114685. [[CrossRef](#)]
35. Pinchover, Y.; Rubinstein, J. *An Introduction to Partial Differential Equations*; Cambridge University Press: Cambridge, UK, 2005; Volume 10.
36. Aksoy, A.; Martelli, M. Mixed Partial Derivatives and Fubini's Theorem. *Coll. Math. J.* **2002**, *33*, 126–130. [[CrossRef](#)]
37. Wang, L.; Lin, Y.; Yan, X.; Shao, Y.; Xu, Z.; Yang, C.; Fan, H.; Xie, Y.; Zheng, C. Modeling of Distributed Parameter Systems Based on Independent Partial Derivative-Physics-Informed Neural Network. *Neurocomputing* **2025**, *642*, 130396. [[CrossRef](#)]
38. Gladstone, R.J.; Nabian, M.A.; Sukumar, N.; Srivastava, A.; Meidani, H. FO-PINN: A First-Order Formulation for Physics-Informed Neural Networks. *Eng. Anal. Bound. Elem.* **2025**, *174*, 106161. [[CrossRef](#)]
39. Shan, B.; Li, Y.; Huang, S.-J. VI-PINNs: Variance-Involved Physics-Informed Neural Networks for Fast and Accurate Prediction of Partial Differential Equations. *Neurocomputing* **2025**, *623*, 129360. [[CrossRef](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.