arXiv:2607.25608v1 [cs.LG] 28 Jul 2026

Physics-Informed Broad Learning System: An Efficient Backpropagation-Free Framework for Solving Partial Differential Equations

Pinki Khatun[†‡] M. Sajid[§] Abhinav Jha[†] M. Tanveer[§]

Abstract

Physics-informed neural networks (PINNs) have emerged as a powerful paradigm for solving partial differential equations (PDEs) by embedding governing physical laws into deep neural networks. However, their reliance on computationally expensive gradient-based optimization and deep architectures often results in slow training, high computational cost, and limited scalability. In this work, we propose a novel physics-informed broad learning system (PI-BLS), the first physics-informed learning framework based on broad RdNNs. The proposed formulation embeds the governing differential operator and the associated initial and boundary constraints directly into a linear output-layer optimization problem, thereby replacing nonlinear gradient-based training with a deterministic least-squares solution obtained via the pseudoinverse. Consequently, the entire learning process is reduced to a single linear optimization stage while preserving the underlying physical constraints. As a result, PI-BLS offers an efficient learning paradigm for a physics-informed learning framework for solving PDEs that eliminates iterative backpropagation while preserving the underlying physical constraints. Experimental results on representative forward PDE benchmarks demonstrate that PI-BLS achieves competitive and often superior performance with reduced training time and model parameters compared with conventional PINNs.

Keywords: Physics-Informed Neural Networks, Broad Learning System, Randomized Neural Networks, Least-Squares Optimization, Partial Differential Equations.

MSC codes: 35A35, 65F20, 68T05, 68T07.

[†]Department of Mathematics, Indian Institute of Technology Gandhinagar, Palaj, Gandhinagar, 382055, Gujarat, India (abhinav.jha@iitgn.ac.in).

[‡]Department of Industrial Engineering, Università degli Studi di Firenze, Viale Morgagni 40/44, 50134 Firenze, Italy (pinki.khatun@unifi.it).

[§]Department of Mathematics, Indian Institute of Technology Indore, Indore 453552, India (phd2101241003@iiti.ac.in, mtanveer@iiti.ac.in).

1 Introduction

Leveraging existing data for accurate modeling and prediction of partial differential equations (PDEs)-governed systems continues to be a challenging task, especially when dealing with ill-posed problems involving unknown parameters or boundary conditions, the presence of noise, and high computational demands. The idea of employing neural networks as universal function approximators for solving PDEs dates back to the 1990s [8]. Building upon this early concept, physics-informed machine learning has emerged over the past few decades as a powerful framework for addressing PDEs by integrating prior physical knowledge with available data [16]. A prominent realization of this paradigm is represented by physics-informed neural networks (PINNs), originally proposed in [8, 18], which incorporate the governing physical laws directly into the loss function of neural networks. PINNs extend the original idea by leveraging modern deep neural architectures, whose expressive power enables the representation of highly complex solution structures. Leveraging automatic differentiation [1, 27], the authors in [28] introduced a practical framework for approximating solutions of PDEs in both forward and inverse settings. This development has been made possible by rapid advances in computational hardware, optimization algorithms, and large-scale training strategies, together with the widespread availability of automatic differentiation tools.

Being mesh-free and easy to implement, the PINN algorithm can be effectively applied to various classes of PDEs, including fractional PDEs [25], integro-differential equations [22], and stochastic PDEs [43, 42]. Consequently, PINNs have attracted significant attention and have been successfully applied to a wide range of scientific and engineering problems, such as fluid dynamics [24], heat transfer [4], biomedicine [29], systems biology [40], and electromagnetism [32]. By embedding the underlying PDEs as soft constraints within the learning process, PINNs enable neural networks to produce solutions that remain consistent with both observational data and fundamental physical principles, even in regimes where data are sparse or noisy. Since the development of PINNs, numerous extensions have been proposed to improve their accuracy, stability, and computational efficiency, such as gradient-enhanced PINNs [37, 39, 41], extented PINN [14, 19, 44], variational PINNs (vPINNs) [3, 10, 21] and hp-vPINNs [11], which enhance solution stability by incorporating variational test spaces and domain decomposition-based schemes [14, 15, 38]. See [12, 13, 23] for the recent developments and comprehensive performance analysis.

Despite their remarkable success, PINNs still face several fundamental challenges that limit their scalability and practical applicability. First, conventional PINNs rely on deep neural network architectures with a large number of trainable parameters, often resulting in complex optimization landscapes and slow convergence [17, 35]. Second, training is performed using gradient-based iterative optimization algorithms, which are computationally expensive and frequently suffer from issues such as gradient pathologies and convergence to poor local minima, particularly for high-dimensional PDEs [17, 35, 36]. Third, the repeated computation of higher-order derivatives through automatic differentiation substantially increases both memory consumption and computational cost, making the training process highly dependent on modern GPU hardware and limiting its efficiency

for large-scale scientific computing applications [7, 28]. Furthermore, the performance of PINNs is often sensitive to the selection of network architectures, loss-function weighting strategies, collocation-point sampling, and hyperparameter tuning, requiring considerable empirical effort to obtain reliable solutions [7]. The aforementioned challenges motivate the search for efficient and scalable physics-informed learning frameworks that overcome the optimization and computational limitations of conventional PINNs while maintaining adherence to the underlying physical laws.

2 Motivation and Contribution

Randomized neural networks (RdNNs) constitute a class of feed-forward architectures [26, 31] developed to alleviate the computational burden and optimization difficulties commonly encountered in gradient-descent–based neural networks. The central idea behind these models is to introduce randomness into the network structure by fixing a subset of parameters, typically those associated with the hidden layers, while only a smaller set of parameters, usually in the output layer, is learned during training. As a result, the training process is significantly simplified, since the remaining parameters can often be obtained through efficient linear optimization procedures or even explicit closed-form solutions. By reducing the number of trainable parameters and avoiding expensive iterative backpropagation, RdNNs can achieve competitive performance with substantially lower computational cost and training time, while requiring far fewer computational resources.

Motivated by the need for computationally efficient learning algorithms, several alternative neural network paradigms have been proposed to improve training efficiency and scalability [5, 26, 34]. Among these, the broad learning system (BLS) [5] has emerged as a promising alternative to conventional deep neural networks. Built upon the principles of RdNNs, BLS replaces deep hierarchical architectures with a broad, flat network that expands horizontally rather than vertically. Specifically, it constructs expressive representations by integrating mapped feature nodes with nonlinear enhancement nodes, thereby enriching the feature space without increasing network depth [6]. Furthermore, the output weights are computed through efficient linear least-squares or pseudoinverse solutions, eliminating the need for computationally expensive iterative backpropagation. Consequently, BLS offers significantly faster training, lower computational complexity, and excellent scalability, making it an attractive candidate for scientific computing applications involving PDEs, where computational efficiency and rapid model adaptation are of paramount importance [30, 33]. BLS expands the network in width, thereby avoiding many optimization challenges associated with deep models while maintaining strong approximation capability [6]. These characteristics naturally position BLS as a promising backbone for developing efficient physics-informed learning algorithms.

Despite these compelling advantages, existing BLS frameworks have been developed primarily for data-driven learning and do not explicitly incorporate the governing physical laws represented by PDEs. Consequently, they cannot be directly employed to solve forward PDE problems, where the predicted solution must simultaneously satisfy the gov-

erning equations and the prescribed initial and boundary conditions. This limitation reveals an important gap between efficient randomized learning models and physics-informed scientific machine learning.

Building upon the computational advantages of broad RdNNs, this paper proposes a novel *physics-informed broad learning system (PI-BLS)* for solving forward PDEs. To the best of our knowledge, PI-BLS is the first physics-informed learning framework based on broad RdNNs for this class of problems. In the proposed framework, the PDE solution is approximated using a BLS architecture, where the feature and enhancement nodes are randomly generated and remain fixed throughout training. A physics-informed objective function is constructed by enforcing the governing PDE together with the prescribed boundary and initial conditions at collocation points. Consequently, the resulting learning problem is reformulated as a linear least-squares optimization problem with respect to the output-layer parameters, replacing the computationally expensive gradient-based optimization employed by conventional PINNs with an efficient closed-form learning strategy that completely eliminates backpropagation. This formulation yields a computationally efficient, scalable, and physically consistent framework for solving forward PDEs.

The main contributions of this work are summarized as follows:

- We propose a novel PI-BLS, which, to the best of our knowledge, is the first physics-informed learning framework based on broad RdNNs for solving forward PDEs.
- We develop a new physics-informed learning formulation that embeds the governing PDE together with the associated boundary and initial conditions directly into the BLS, enabling the approximation of PDE solutions within a randomized broad neural architecture.
- We reformulate the resulting learning problem as a linear least-squares optimization with respect to the output-layer parameters, allowing the model to be trained without iterative gradient-based backpropagation while preserving the underlying physical constraints.
- We present a theoretical analysis of PI-BLS by deriving its stability property, residual error bound, sensitivity to perturbations, and physics consistency, providing rigorous guarantees for the robustness and physical fidelity of the proposed framework.
- We evaluate the proposed PI-BLS framework on a diverse set of forward PDE benchmark problems, including one-dimensional steady-state, two-dimensional Poisson, and diffusion–reaction equations, and compare its performance against existing baselines in terms of solution accuracy, training losses, computational time, and model complexity.

The remainder of this paper is organized as follows. Section 3 reviews the existing literature on BLS and PINNs. Section 4 presents the proposed PI-BLS framework and its mathematical formulation. Section 5 gives details of the theoretical properties and

observations of the PI-BLS. Section 6 reports numerical experiments and discusses the performance of proposed PI-BLS and comparison with the existing ones for several case studies. Section 7 provides the concluding remarks of the paper.

3 Brief Overview of BLS and PINN

This section briefly introduces the BLS and the PINN architectures. BLS is a wide neural network model that enables efficient learning through a least-squares-based training strategy, while PINNs incorporate governing physical laws into the loss function to solve PDEs. These two architectures form the basis of the proposed approach. Before proceeding, we introduce the following useful notation.

Notation: We denote by $\mathbb{R}^{m\times n}$ the set of all $m \times n$ real matrices and by $\mathbb{R}^n$ the set of n-dimensional real vectors. The superscript $(\cdot)^T$ indicates the transpose of a matrix or vector. By $\begin{bmatrix} A & B \end{bmatrix} \in \mathbb{R}^{m\times(n+p)}$, we denote the concatenation of the matrices $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{m\times p}$. The symbol I represents the identity matrix of appropriate dimension.

3.1 Broad Learning System

In this subsection, we present the architecture of the BLS. Let $\{(X, Y)|X \in \mathbb{R}^{N\times m}, Y \in \mathbb{R}^N\}$ be the training dataset, where $X \in \mathbb{R}^{N\times m}$ is the input matrix and $Y \in \mathbb{R}^N$ is the output matrix.

The BLS is composed of three key components: (1) mapping feature nodes, (2) enhancement feature nodes, and (3) the output matrix. The training process of the BLS involves two main stages. First, the weights for both the mapping and enhancement nodes are randomly generated, eliminating the need for iterative tuning in these layers. Second, the weights connecting the hidden layer to the output layer are computed analytically, typically using a least-squares solution. In essence, only the final output weights are learned during training, while the rest of the network remains fixed. This significantly reduces computational cost and training time.

The architecture of the BLS [5] is shown in Figure 1. Let there be N_c feature layers, each consisting of p nodes. For the i^{th} feature layer, let $W_i^f \in \mathbb{R}^{m\times p}$ and $b_i^f \in \mathbb{R}^{1\times p}$ denote the randomly generated weight matrix and bias vector, respectively. The output of the i^{th} feature layer is then defined as:

$$F^i = \psi_i(XW_i^f + \mathbf{1}^T \otimes b_i^f) \in \mathbb{R}^{N\times p}, \quad i = 1, 2, \ldots, N_c, \tag{3.1}$$

where $\mathbf{1} \in \mathbb{R}^N$ is the vector of all ones and ψ_i is the activation function (feature mapping) associated with the i^{th} feature group, applied elementwise to the matrix $\mathbf{F}W_i^f + b_i^f$. The symbol $\otimes$ denotes the Kronecker product. The following augmented matrix gives the output of the feature group:

$$\mathbf{F} = \begin{bmatrix} F^1 & F^2 & \ldots & F^{N_c} \end{bmatrix} \in \mathbb{R}^{N\times pN_c}. \tag{3.2}$$

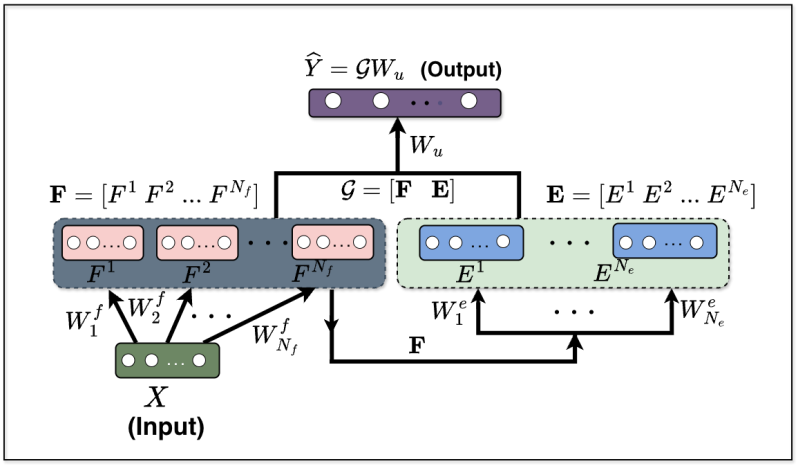


Figure 1: Architecture of the BLS

In the next step, the augmented feature matrix is projected into the enhancement space through random transformations, followed by the application of activation functions. Let N_e denote the number of enhancement layers, with each group consisting of q nodes. Further, let $W_i^e \in \mathbb{R}^{pN_c \times q}$ be the randomly generated weight matrix connecting the feature matrix $\mathbf{F}$ to the i^{th} enhancement group, and let $b_i^e \in \mathbb{R}^{1\times q}$ denote the corresponding bias vector for the i^{th} enhancement group. Then

$$E^i = \phi_i(\mathbf{F}W_i^e + \mathbf{1} \otimes b_i^e) \in \mathbb{R}^{N\times q}, \quad i = 1, 2, \ldots, N_e, \tag{3.3}$$

where ϕ_i is the activation function used to generate nonlinear features, applied elementwise to the matrix $\mathbf{F}W_j^e + b_j^e$. The output of the N_e enhancement feature groups is

$$\mathbf{E} = \begin{bmatrix} E^1 & E^2 & \ldots & E^{N_e} \end{bmatrix} \in \mathbb{R}^{N\times qN_e}. \tag{3.4}$$

Finally, the resulting enhancement matrix $\mathbf{E}$, together with the augmented feature matrix $\mathbf{F}$, is passed to the output layer, where the resulting output of the network is computed as follows:

$$\widehat{Y} = \mathcal{G}W_u, \tag{3.5}$$

where $\mathcal{G} = \begin{bmatrix} \mathbf{F} & \mathbf{E} \end{bmatrix} \in \mathbb{R}^{N\times(pN_c+qN_e)}$ denotes the concatenated matrix of the augmented feature layer and the resultant enhancement layer, and $W_u \in \mathbb{R}^{(pN_c+qN_e)\times 1}$ is the randomly generated weight matrix that connects this combined representation to the output layer. Finally, the training of the BLS is completed by estimating the output weight matrix through the following least-squares optimization problem:

$$\min_{W_u} \|\mathcal{G}W_u - Y\|_2^2. \tag{3.6}$$

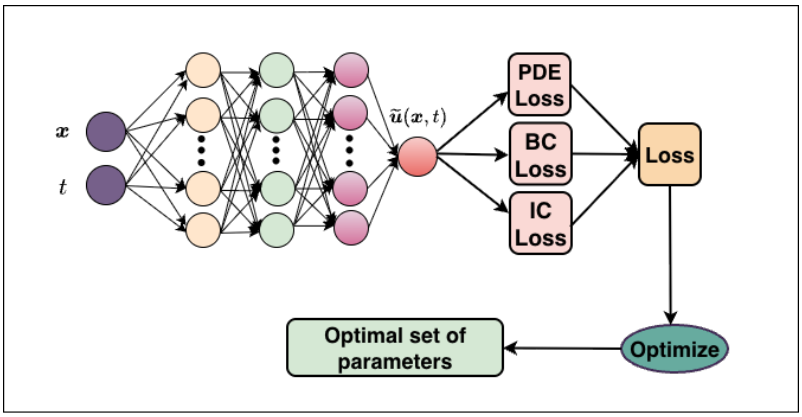


Figure 2: Framework of PINN

3.2 Physics-Informed Neural Network

Physics-informed neural networks (PINNs) are a class of machine learning methods that incorporate domain-specific knowledge, typically represented by PDEs—into the neural network training process. The central idea of PINNs is to embed the governing physical laws as constraints, ensuring that the neural network's predictions adhere to the underlying system dynamics during training. We consider the following form of a general PDE:

$$\frac{\partial}{\partial t}\boldsymbol{u}(\boldsymbol{x},t)+\mathcal{N}_{\boldsymbol{x}}[\boldsymbol{u}(\boldsymbol{x},t)]=R(\boldsymbol{x},t),\quad \boldsymbol{x}\in\Omega,\ \text{and}\ t\in[0,T] \tag{3.7}$$

$$\mathcal{B}[\boldsymbol{u}(\boldsymbol{x},t)]=B(\boldsymbol{x},t),\quad \boldsymbol{x}\in\partial\Omega,\ \text{and}\ t\in[0,\,T], \tag{3.8}$$

$$\boldsymbol{u}(\boldsymbol{x},0)=\mathcal{I}(\boldsymbol{x}),\ \ \boldsymbol{x}\in\Omega, \tag{3.9}$$

where $\boldsymbol{x}\in\mathbb{R}^d$, $t\in[0,\,T]$, and $\mathcal{N}_{\boldsymbol{x}}[\boldsymbol{u}]$ and $\mathcal{B}[\boldsymbol{u}]$ are the potentially nonlinear differential operator and the boundary condition operator, respectively, and $\partial\Omega$ is the boundary of the computation domain Ω.

The framework for PINN is given in Figure 2. The PINN is employed to approximate the solution $\boldsymbol{u}(x,t)$ by $\widetilde{\boldsymbol{u}}(x,t)$, where the network takes $(\boldsymbol{x},t)$ as inputs. Let the collocation points at the interior of the domain be denoted by $\{\boldsymbol{x}_{i,u},t_{i,u}\}_{i=1}^{N_c}$, the boundary points by $\{\boldsymbol{x}_{i,b},t_{i,b}\}_{i=1}^{N_b}$, and the initial points $\{\boldsymbol{x}_{i,I},0\}_{i=1}^{N_I}$, where N_u, N_b and N_I denote the number of training collocation points, boundary points, and initial condition points, respectively. The residuals associated with the PDE, boundary conditions, and initial conditions are denoted by $\mathcal{J}_u$, $\mathcal{J}_b$, and $\mathcal{J}_I$, respectively, and given as follows:

$$\mathcal{J}_u(\boldsymbol{x},t)=\frac{\partial\widetilde{\boldsymbol{u}}}{\partial t}(\boldsymbol{x},t)+\mathcal{N}_{\boldsymbol{x}}[\widetilde{\boldsymbol{u}}](\boldsymbol{x},t)-R(\boldsymbol{x},t)\ \ \text{on}\ \ \{\boldsymbol{x}_{i,u},t_{i,u}\}_{i=1}^{N_c}, \tag{3.10}$$

$$\mathcal{J}_b(\boldsymbol{x},t)=\mathcal{B}[\widetilde{\boldsymbol{u}}](\boldsymbol{x},t)-B(\boldsymbol{x},t)\ \ \text{on}\ \ \{\boldsymbol{x}_{i,b},t_{i,b}\}_{i=1}^{N_b}, \tag{3.11}$$

$$\mathcal{J}_I(\boldsymbol{x},0)=\widetilde{\boldsymbol{u}}(\boldsymbol{x},0)-\mathcal{I}(\boldsymbol{x})\ \ \text{on}\ \ \{\boldsymbol{x}_{i,I},0\}_{i}^{N_I}. \tag{3.12}$$

Therefore, the loss terms due to the PDE, boundary conditions, and initial conditions, respectively, are given by

$$\mathcal{J}_u = \sum_{i=1}^{N_c} \left| \frac{\partial \widetilde{\boldsymbol{u}}}{\partial t}(\boldsymbol{x}_{i,u}, t_{i,u}) + \mathcal{N}_{\boldsymbol{x}}[\widetilde{\boldsymbol{u}}(\boldsymbol{x}_{i,u}, t_{i,u})] - R(\boldsymbol{x}_{i,u}, t_{i,u}) \right|^2, \tag{3.13}$$

$$\mathcal{J}_b = \sum_{i=1}^{N_b} \left| \mathcal{B}[\widetilde{\boldsymbol{u}}](\boldsymbol{x}_{i,b}, t_{i,b}) - B(\boldsymbol{x}_{i,b}, t_{i,b}) \right|^2, \tag{3.14}$$

$$\mathcal{J}_I = \sum_{i=1}^{N_I} \left| \widetilde{\boldsymbol{u}}(\boldsymbol{x}_{i,I}, 0) - \mathcal{I}(\boldsymbol{x}_{i,I}) \right|^2. \tag{3.15}$$

Accordingly, the total loss to be minimized for a PINN is given by:

$$\mathcal{J} := \frac{1}{2N_u}\mathcal{J}_u + \frac{1}{2N_b}\mathcal{J}_b + \frac{1}{2N_I}\mathcal{J}_b. \tag{3.16}$$

The partial derivatives involved in the loss function can be efficiently computed using automatic differentiation [1, 27]. By applying a gradient-based optimization algorithm, we can minimize the loss function $\mathcal{J}$ to determine the optimal weights and biases of the neural network.

4 Proposed Physics-Informed Broad Learning System

In this section, we introduce the physics-informed broad learning system (PI-BLS), which integrates the principles of PINNs with BLS to solve linear PDEs. The core idea is to incorporate physical laws directly into the BLS framework as part of the cost function. This transforms linear PDE problems into a regularized linear least-squares problem, allowing the solution to simultaneously satisfy the governing equations and boundary/initial conditions in a computationally efficient manner. In the following, we discuss the mathematical architecture of PI-BLS.

Consider the general PDE problem (3.7)-(3.9). The architecture of the PI-BLS is shown in Figure 3. Define $X = [\boldsymbol{x}^T, t] \in \mathbb{R}^{1\times(d+1)}$, where $\boldsymbol{x} \in \mathbb{R}^d$ and $t \in \mathbb{R}$. Let the network have N_f feature layers with p feature nodes in each layer. The feature layers are denoted by F^i, $i = 1, 2, \ldots, N_c$. Further, let $W_i^f \in \mathbb{R}^{(d+1)\times p}$ be the weight matrix and $b_i^f \in \mathbb{R}^{1\times p}$ be the biases in the i^{th} feature layer. Then the output of the feature layer segment is given by

$$\mathbf{F} = \begin{bmatrix} F^1 & F^2 & \ldots & F^{N_c} \end{bmatrix} \in \mathbb{R}^{1\times pN_c}, \tag{4.1}$$

where $F^i(X) = \psi_i(XW_i^f + b_i^f) \in \mathbb{R}^{1\times p}$ is the output in the i^{th} physics-informed feature layer, and $\psi_i : \mathbb{R} \to \mathbb{R}$ are the feature maps or nonlinear activation functions.

Let the network consist of N_e enhancement layers, each containing q enhancement nodes. These enhancement layers are denoted by E^j, for $j = 1, 2, \ldots, N_e$. Let $W_j^e \in \mathbb{R}^{pN_c\times q}$

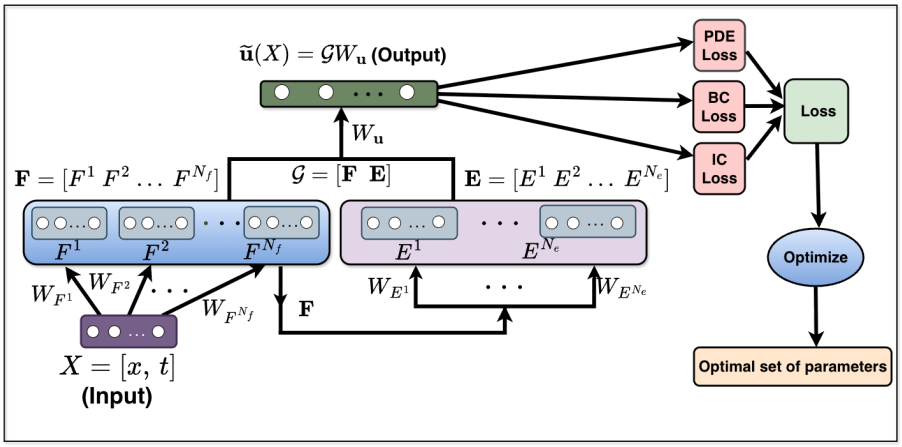


Figure 3: Framework of the proposed PI-BLS

be the weight matrix and $b_j^e \in \mathbb{R}^{1\times q}$ be the bias vector in the j^{th} enhancement layer. Then, the output of the enhancement layer is given by:

$$\mathbf{E} = \begin{bmatrix} E^1 & E^2 & \dots & E^{N_e} \end{bmatrix} \in \mathbb{R}^{1\times qN_e}, \tag{4.2}$$

where

$$E^j(X) = \phi_j(\mathbf{F}W_j^e + b_j^e) \in \mathbb{R}^{1\times q},$$

with $\phi_j : \mathbb{R} \to \mathbb{R}$ being nonlinear activation functions. The final output of the network denoted by $\widetilde{\boldsymbol{u}}$ and is given by

$$\widetilde{\boldsymbol{u}}(X; W_u) = \mathcal{G}W_{\boldsymbol{u}}, \tag{4.3}$$

where $\mathcal{G} = [\boldsymbol{F}\ \ \boldsymbol{E}] \in \mathbb{R}^{1\times(pN_c+qN_e)}$ and $W_{\boldsymbol{u}} \in \mathbb{R}^{(pN_c+qN_e)\times 1}$. Hence, $\widetilde{\boldsymbol{u}}(X; W_u)$ is the approximate solution of the PDE (3.7)–(3.9) obtained using the network.

Let N_u, N_b, and N_I denote the numbers of collocation points, boundary points, and initial conditions, respectively, due to training. The error functions in approximating the PDE, boundary conditions, and initial conditions are the same as in (3.10), (3.11), and (3.12). Hence, the loss functions to be minimized for a PI-BLS are given by:

$$\mathcal{J}_u(\boldsymbol{x}, t) = \frac{\partial(\mathcal{G}W_u)}{\partial t}(\boldsymbol{x}, t) + \mathcal{N}_{\boldsymbol{x}}[\mathcal{G}W_u](\boldsymbol{x}, t) - R(\boldsymbol{x}, t) \ \text{ on } \ \{\boldsymbol{x}_{i,u}, t_{i,u}\}_{i=1}^{N_c}, \tag{4.4}$$

$$\mathcal{J}_b(\boldsymbol{x}, t) = \mathcal{B}[\mathcal{G}W_u](\boldsymbol{x}, t) - B(\boldsymbol{x}, t) \ \text{ on } \ \{\boldsymbol{x}_{i,b}, t_{i,b}\}_{i=1}^{N_b}, \tag{4.5}$$

$$\mathcal{J}_I(\boldsymbol{x}) = \mathcal{G}W_u(\boldsymbol{x}, 0) - \mathcal{I}(\boldsymbol{x}) \ \text{ on } \ \{\boldsymbol{x}_{i,I}, 0\}_i^{N_I}. \tag{4.6}$$

For convenience, we define the corresponding data matrices as:

$$X_{\text{col}} = [(\boldsymbol{x}_{1,u}, t_{1,u}), (\boldsymbol{x}_{2,u}, t_{2,u}), \dots, (\boldsymbol{x}_{N_u,u}, t_{N_u,u})]^T,$$

$$X_b = [(\boldsymbol{x}_{1,b}, t_{1,b}), (\boldsymbol{x}_{2,b}, t_{2,b}), \dots, (\boldsymbol{x}_{N_b,b}, t_{N_b,b})]^T,$$

and

$$X_I = [(\boldsymbol{x}_{1,I}, 0), (\boldsymbol{x}_{2,I}, 0), \ldots, (\boldsymbol{x}_{N_I,I}, 0)]^T.$$

Algorithm 1 Training process of proposed PI-BLS for PDE problems

Input: Dimension of the the input X, number of feature groups N_c, number of feature nodes p, number of enhancement groups N_e, number of enhancement nodes q, hyperparameters λ_b, $\lambda_I, \lambda_{\text{reg}} > 0$

Output: Output weight matrix W_u^*/approximate solution $\widetilde{\boldsymbol{u}}(X; W_u^*)$.

Generate collocation points X_{col}, boundary points X_b and initial points X_I.

Feature Layer Construction:

for $i = 1$ to N_c **do**

    Generate random weight matrix $W_i^f \in \mathbb{R}^{d\times p}$ and bias vector $b_i^f \in \mathbb{R}^{1\times p}$.

    Compute feature mapping: $F^i = \psi_i(XW_i^f + b_i^f)$.

end for

Concatenate feature outputs $\mathbf{F} = \begin{bmatrix} F^1 & F^2 & \cdots & F^{N_c} \end{bmatrix}$.

Enhancement Layer Construction:

for $i = 1$ to N_e **do**

    Generate random weight matrix $W_i^e \in \mathbb{R}^{(pN_c)\times q}$ and bias vector $b_i^e \in \mathbb{R}^{1\times q}$.

    Compute enhancement mapping: $\mathbf{E}^i = \phi_i(\mathbf{F}W_i^e + b_i^e)$.

end for

Concatenate enhancement outputs $\mathbf{E} = \begin{bmatrix} E^1 & E^2 & \cdots & E^{N_e} \end{bmatrix}$.

Compute $\mathcal{G} = \begin{bmatrix} \boldsymbol{F} & \boldsymbol{E} \end{bmatrix}$

Compute

$$H_{\text{col}} = \frac{\partial \mathcal{G}}{\partial t}(X_{\text{col}}) + \mathcal{N}_{\boldsymbol{x}}[\mathcal{G}](X_{\text{col}}), \quad H_b = \mathcal{B}[\mathcal{G}](X_b), \quad H_I = \mathcal{G}(X_I).$$

Construct the matrices

$$\mathcal{H} = \begin{bmatrix} H_{\text{col}} \\ \sqrt{\lambda_b} H_b \\ \sqrt{\lambda_I} H_I \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} R_{\text{col}} \\ \sqrt{\lambda_b} R_b \\ \sqrt{\lambda_I} R_I \end{bmatrix}.$$

Compute output layer weight matrix W_u using (4.13).

When $\mathcal{N}_{\boldsymbol{x}}[\,\cdot\,]$ and $\mathcal{B}[\,\cdot\,]$ are linear differential operators, then the proposed minimization problem is expressed as:

$$\arg\min_{W_u} \Big(\|H_{\text{col}} W_u - R(X_{\text{col}})\|_2^2 + \lambda_b \|H_b W_u - B(X_b)\|_2^2 \tag{4.7}$$

$$+ \lambda_I \|H_I W_u - \mathcal{I}(X_I)\|_2^2 + \lambda_{\text{reg}} \|W_u\|_2^2 \Big), \tag{4.8}$$

Table 1: Computational complexity comparison of different physics-informed learning frameworks.

Method	**Training Strategy**	**Computational Complexity**
PINN	Gradient-based optimization	$\mathcal{O}(TN_uL_hN_h^2)$
PI-BLS	Regularized least-squares	$\mathcal{O}(N_uM^2 + M^3)$
PI-ELM	Least-squares	$\mathcal{O}(N_uN_h^2 + N_h^3)$
B-PI-ELM	Bayesian least-squares	$\mathcal{O}(N_uN_h^2 + N_h^3)$

where

$$H_{\text{col}} = \frac{\partial \mathcal{G}}{\partial t}(X_{\text{col}}) + \mathcal{N}_{\boldsymbol{x}}[\mathcal{G}](X_{\text{col}}) \in \mathbb{R}^{N_u \times (pN_c + qN_e)}, \tag{4.9}$$

$$H_b = \mathcal{B}[\mathcal{G}](X_b) \in \mathbb{R}^{N_b \times (pN_c + qN_e)}, \tag{4.10}$$

$$H_I = \mathcal{G}(X_I) \in \mathbb{R}^{N_I \times (pN_c + qN_e)}. \tag{4.11}$$

The hyperparameters λ_b, $\lambda_I, \lambda_{\text{reg}} > 0$ are scalar weights used to balance the loss function. The above optimization problem can be compactly expressed as:

$$W_u^* = \arg\min_{W_u} \|\mathcal{H}W_u - Z\|_2^2 + \lambda_{\text{reg}} \|W_u\|_2^2, \tag{4.12}$$

where

$$\mathcal{H} = \begin{bmatrix} H_{\text{col}} \\ \sqrt{\lambda_b} H_b \\ \sqrt{\lambda_I} H_I \end{bmatrix} \in \mathbb{R}^{N_u \times (pN_c + qN_e)} \text{ and } Z = \begin{bmatrix} R_{\text{col}} \\ \sqrt{\lambda_b} R_b \\ \sqrt{\lambda_I} R_I \end{bmatrix} \in \mathbb{R}^{N_u},$$

and $N_u = (N_c + N_b + N_I)$. The matrices $\mathcal{H}$ and Z depend upon the physics law of PDE, boundary conditions, and initial conditions. For this optimal weight matrix in the output layer, PI-BLS approximate solution is defined as $\widetilde{\boldsymbol{u}}(X; W_u^*)$.

The weight matrix W_u at the output layer can be calculated by the least squares method and is given by

$$W_u^* = \begin{cases} (\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\mathcal{H}^T Z, & N \leq (pN_c + qN_q), \\ \mathcal{H}^T(\mathcal{H}\mathcal{H}^T + \lambda_{\text{reg}} I)^{-1}\mathcal{H}^T Z, & N > (pN_c + qN_q). \end{cases} \tag{4.13}$$

Algorithm 1 gives the step-by-step implementation of the training process for a general linear PDE. We use automatic differentiation [1, 27] to find the derivative of the approximate function.

4.1 Computational Complexity Analysis

The computational complexity of the proposed PI-BLS framework is compared with those of PINN, and RdNN-based physics-informed extreme learning machine (PI-ELM) [9], and Bayesian PI-ELM (B-PI-ELM) [20]. Let N_u denote the total number of collocation points, including interior, boundary, and initial points. For PINN, let L_h and N_h denote the

number of hidden layers and hidden nodes per layer, respectively, and let T denote the total number of optimization iterations. For PI-BLS, let $M = pN_c + q$, where $N_e = 1$, $N_c \times p$ represents the total number of mapped feature nodes, and q denotes the number of enhancement nodes. For a conventional PINN, the dominant computational cost arises from repeated forward propagation and backpropagation over all trainable parameters during each optimization iteration. Assuming a fully connected network, the overall training complexity can be approximated as $\mathcal{O}_{\text{PINN}} = \mathcal{O}(TN_uL_hN_h^2)$. In contrast, the hidden-layer parameters of PI-BLS remain fixed after random initialization, and only the output-layer weights are estimated through a regularized least-squares problem. The computational cost consists of constructing the hidden-layer output matrix, forming the normal equations, and solving the resulting linear system, yielding $\mathcal{O}_{\text{PI-BLS}} = \mathcal{O}(N_uM^2 + M^3)$. Similarly, both PI-ELM and B-PI-ELM employ randomized hidden layers and determine the output weights through linear least-squares optimization. Their dominant computational complexity is therefore given by $\mathcal{O}_{\text{PI-ELM}} = \mathcal{O}_{\text{B-PI-ELM}} = \mathcal{O}(N_uN_h^2 + N_h^3)$.

The computational complexities of all competing methods are summarized in Table 1. It can be observed that the complexity of PINN scales linearly with the number of optimization iterations, making the training process considerably more expensive for large-scale problems. In contrast, PI-BLS, PI-ELM, and B-PI-ELM require only a single least-squares optimization, eliminating iterative backpropagation altogether. Consequently, PI-BLS achieves computational complexity of the same order as PI-ELM and B-PI-ELM while maintaining a richer broad learning architecture.

5 Theoretical Properties of PI-BLS

This section establishes the theoretical foundations of the proposed PI-BLS framework. Specifically, we analyze its numerical stability, error bound, sensitivity to perturbations, and consistency with the underlying governing physics. Together, these theoretical results demonstrate the soundness of the proposed formulation and provide rigorous guarantees for its application to forward PDE problems.

Theorem 5.1 (**Stability of the PI-BLS Solution**). *Let $\mathcal{H} \in \mathbb{R}^{N_u \times M}$ denote the hidden-layer output matrix of the proposed PI-BLS. Let $W_u \in \mathbb{R}^M$ denote the output-weight vector obtained by minimizing the regularized physics-informed objective*

$$\mathcal{J}(W_u) := \|\mathcal{H}W_u - Z\|_2^2 + \lambda_{\text{reg}}\|W_u\|_2^2, \tag{5.1}$$

where $Z \in \mathbb{R}^{N_u}$ is the target vector containing the PDE residuals together with the prescribed boundary and initial conditions, and $\lambda_{\text{reg}} > 0$ is the regularization parameter.

Then the PI-BLS solution is stable in the sense that

$$\|W_u\|_2 \leq \frac{\|\mathcal{H}\|_2}{\lambda_{\text{reg}}} \|Z\|_2. \tag{5.2}$$

Hence, every bounded physics-informed target vector produces a bounded output-weight vector.

Proof. The objective function is given by

$$\mathcal{J}(W_u) = \|\mathcal{H}W_u - Z\|_2^2 + \lambda_{\text{reg}}\|W_u\|_2^2. \tag{5.3}$$

Since $\mathcal{J}(W_u)$ is continuously differentiable, its minimizer satisfies the first-order optimality condition

$$\nabla\mathcal{J}(W_u) = 2\mathcal{H}^T(\mathcal{H}W_u - Z) + 2\lambda_{\text{reg}}W_u = 0. \tag{5.4}$$

Rearranging the above equation gives

$$(\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}}I)W_u = \mathcal{H}^T Z, \tag{5.5}$$

and therefore

$$W_u = (\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}}I)^{-1}\mathcal{H}^T Z. \tag{5.6}$$

Taking the Euclidean norm on both sides yields

$$\|W_u\|_2 \leq \|(\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}}I)^{-1}\|_2 \, \|\mathcal{H}^T\|_2 \, \|Z\|_2. \tag{5.7}$$

Since $\lambda_{\min}(\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}}I) \geq \lambda_{\text{reg}}$, we have

$$\|(\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}}I)^{-1}\|_2 \leq \frac{1}{\lambda_{\text{reg}}}. \tag{5.8}$$

Moreover, $\|\mathcal{H}^T\|_2 = \|\mathcal{H}\|_2$. Hence,

$$\|W_u\|_2 \leq \frac{\|\mathcal{H}\|_2}{\lambda_{\text{reg}}} \|Z\|_2, \tag{5.9}$$

which proves the desired result. ■

Theorem 5.2 (**Residual Error Bound**). *Let* $W_u^* = \underset{W_u}{arg\,min}\mathcal{J}(W_u)$*, where*

$$\mathcal{J}(W_u) = \|\mathcal{H}W_u - Z\|_2^2 + \lambda_{\text{reg}}\|W_u\|_2^2, \tag{5.10}$$

with $\lambda_{\text{reg}} \geq 0$*. Then, for every admissible output-weight vector* W_u*, the optimal PI-BLS solution satisfies*

$$\mathcal{J}(W_u^*) \leq \mathcal{J}(W_u). \tag{5.11}$$

Furthermore, if there exists an exact output-weight vector W_u^e *such that* $\mathcal{H}W_u^e = Z$*, then the approximation error satisfies*

$$\|\mathcal{H}(W_u^* - W_u^e)\|_2 \leq \sqrt{\mathcal{J}(W_u^*)}. \tag{5.12}$$

Proof. Since W_u^* is the minimizer of the objective function, it follows immediately that

$$\mathcal{J}(W_u^*) \leq \mathcal{J}(W_u)$$

for every admissible W_u. Suppose there exists an exact solution W_u^e satisfying $\mathcal{H}W_u^e = Z$. Then $\mathcal{H}(W_u^* - W_u^e) = \mathcal{H}W_u^* - Z$. Therefore,

$$\|\mathcal{H}(W_u^* - W_u^e)\|_2^2 = \|\mathcal{H}W_u^* - Z\|_2^2 \leq \mathcal{J}(W_u^*), \tag{5.13}$$

since the regularization term $\lambda_{\text{reg}}\|W_u^*\|_2^2$ is non-negative. Taking square roots on both sides completes the proof. ■

Theorem 5.3 (**Sensitivity of the PI-BLS Solution**). *Assume that the output-weight vector is computed by solving*

$$W_u = (\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\mathcal{H}^T Z, \tag{5.14}$$

where $\lambda_{\text{reg}} > 0$. *Let the physics-informed target vector be perturbed as* $\widetilde{Z} = Z + \Delta Z$, *and let* $\widetilde{W}_u$ *denote the corresponding PI-BLS solution. Then the perturbation in the learned output weights satisfies*

$$\|\widetilde{W}_u - W_u\|_2 \leq \frac{\|\mathcal{H}\|_2}{\lambda_{\text{reg}}} \|\Delta Z\|_2. \tag{5.15}$$

Hence, sufficiently small perturbations in the physics-informed target vector produce proportionally small variations in the learned PI-BLS solution.

Proof. The perturbed solution is given by

$$\widetilde{W}_u = (\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\mathcal{H}^T \widetilde{Z}. \tag{5.16}$$

Subtracting the original solution yields

$$\widetilde{W}_u - W_u = (\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\mathcal{H}^T(\widetilde{Z} - Z) = (\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\mathcal{H}^T \Delta Z. \tag{5.17}$$

Taking the Euclidean norm gives

$$\|\widetilde{W}_u - W_u\|_2 \leq \|(\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\|_2 \, \|\mathcal{H}^T\|_2 \, \|\Delta Z\|_2. \tag{5.18}$$

Since

$$\|(\mathcal{H}^T\mathcal{H} + \lambda_{\text{reg}} I)^{-1}\|_2 \leq \frac{1}{\lambda_{\text{reg}}} \quad \text{and} \quad \|\mathcal{H}^T\|_2 = \|\mathcal{H}\|_2,$$

it follows that

$$\|\widetilde{W}_u - W_u\|_2 \leq \frac{\|\mathcal{H}\|_2}{\lambda_{\text{reg}}} \|\Delta Z\|_2,$$

which completes the proof. ■

Theorem 5.4 (**Physics Consistency of PI-BLS**). *Let* $\widetilde{\boldsymbol{u}}(X; W_u)$ *denote the PI-BLS approximation of the solution* $\boldsymbol{u}(X)$, *where* W_u *represents the trainable output weights. Suppose that* W_u^* *minimizes the physics-informed objective function*

$$\mathcal{J} = \mathcal{J}_u + \lambda_b \mathcal{J}_b + \lambda_I \mathcal{J}_I + \lambda_{\text{reg}} \|W_u\|_2^2, \tag{5.19}$$

with $\lambda_b, \lambda_I, \lambda_{\text{reg}} \geq 0$. *If the regularization parameter satisfies* $\lambda_{\text{reg}} = 0$ *and the minimum value of the objective function is zero, i.e.,* $\mathcal{J}(W_u^*) = 0$, *then the PI-BLS approximation exactly satisfies the governing PDE together with all prescribed boundary and initial conditions at every collocation point.*

Proof. The objective function is given by

$$\mathcal{J} = \mathcal{J}_u + \lambda_b \mathcal{J}_b + \lambda_I \mathcal{J}_I + \lambda_{\text{reg}} \|W_u\|_2^2,$$

where each loss term is non-negative because it is defined as the mean squared residual of the corresponding physical constraint. Hence, $\mathcal{J}_u \geq 0,\ \mathcal{J}_b \geq 0,$ and $\mathcal{J}_I \geq 0$.

Assume that $\lambda_{\text{reg}} = 0$ and that the global minimum satisfies $\mathcal{J}(W_u^*) = 0$. Since the objective is a sum of non-negative terms, each term must vanish individually, namely, $\mathcal{J}_u = 0,\ \mathcal{J}_b = 0,$ and $\mathcal{J}_I = 0$. The condition $\mathcal{J}_u = 0$ implies that the residual of the governing differential equation is zero at every interior collocation point, i.e., $\frac{\partial \widetilde{\boldsymbol{u}}(X;W_u^*)}{\partial t} + \mathcal{N}_{\boldsymbol{x}}(\widetilde{\boldsymbol{u}}(X;W_u^*)) = R(X)$. Similarly, $\mathcal{J}_b = 0$ implies that all prescribed boundary conditions are satisfied exactly on the boundary collocation points, while $\mathcal{J}_I = 0$ ensures that the initial conditions are satisfied exactly at the temporal collocation points whenever the problem is time-dependent.

Therefore, the PI-BLS approximation satisfies the governing PDE together with all prescribed boundary and initial conditions at every collocation point. Hence, the obtained solution is fully consistent with the imposed physical constraints. ■

6 Experiments and Discussions

To evaluate the performance of the proposed PI-BLS, we consider several linear PDEs. First, we provide details of the implementation used in the empirical study, followed by discussions of the test cases.

6.1 Implementation Details

All experiments are conducted under a unified computational environment to ensure a fair comparison among the competing methods. The implementation details, test problems, baseline methods, hyperparameter configuration, and metrics are discussed below.

6.1.1 Setup

All experiments were implemented in `Python 3.14.3` using the `PyTorch 2.11.0` framework with CUDA 12.6 and cuDNN 9.10 acceleration. The experiments were conducted on a workstation equipped with an Intel® Xeon® Platinum 8260 CPU (24 cores, 48 logical processors, 2.30 GHz), 256 GB RAM, and an NVIDIA RTX A4500 GPU running the Windows 10 operating system. For all learning models, hyperparameters were selected using a 5-fold cross-validation strategy combined with grid search. Specifically, the training data were partitioned into five folds, and for each candidate hyperparameter configuration, the validation RMSE was computed independently on each fold. The average validation RMSE over the five folds was used as the model selection criterion, and the hyperparameter configuration yielding the lowest average validation RMSE was selected. The final model was

subsequently retrained using the selected hyperparameters, and its performance was evaluated on an independent test set using RMSE, MSE, MAE and their corresponding relative error metrics. For a fair comparison, all competing methods are trained and evaluated using identical training and testing points for each benchmark problem.

6.1.2 Test Problems

The test cases of PDEs are taken from [2, 41]. In the first example, we provide a detailed construction of the matrices and the associated loss function to help the reader gain a better understanding of the method. Let $\boldsymbol{u}(X)$ and $\widetilde{\boldsymbol{u}}(X; W_u^*)$ be the exact solution of the PDE and PI-BLS solution, respectively.

For the one-dimensional steady-state advection equation, the number of interior collocation points was varied as $N_{\text{train}} \in \{5, 10, 15, 20, 25, 30\}$, while the test set consisted of $N_{\text{test}} = 100$ evaluation points. The interior collocation points were generated by randomly sampling points from the computational domain $(0, 1)$.

For the two-dimensional Poisson equation, the number of interior collocation points was varied as $N_{\text{train}} \in \{50, 60, 70, 80, 90, 100\}$, and the solution accuracy was evaluated using $N_{\text{test}} = 10,000$ test points. The interior collocation points were randomly sampled within the irregular computational domain using a polar-coordinate parameterization, thereby ensuring an approximately uniform spatial distribution while accurately preserving the domain geometry.

For the diffusion–reaction equation, the number of interior collocation points was also varied as $N_{\text{train}} \in \{50, 60, 70, 80, 90, 100\}$, with $N_{\text{test}} = 10,000$ evaluation points. The interior collocation points were generated by uniformly sampling the spatial variable over $[-\pi, \pi]$ and the temporal variable over $[0, 1]$, producing space–time samples at which the governing PDE residual is enforced.

6.1.3 Baseline Methods

The proposed PI-BLS framework is evaluated through comprehensive comparisons with the conventional PINN [28] with two hidden layers. We also take two representative randomized physics-informed learning models, namely PI-ELM [9] and B-PI-ELM [20], on all three benchmark problems. All competing methods are trained and evaluated under identical experimental settings to ensure a fair and unbiased comparison.

6.1.4 Hyperparameter Configuration

For each benchmark problem, the hyperparameters of all competing methods were selected using the 5-fold cross-validation and grid-search procedure described previously. Since different PDEs impose different physical constraints, the search spaces were adjusted accordingly. In particular, the initial-condition penalty parameter (λ_I) was introduced only for the time-dependent diffusion–reaction equation, whereas the regularization parameter (λ_{reg}) was considered only when applicable. The hyperparameter search spaces for each benchmark problem are summarized in Tables 2, 3, and 4.

Table 2: Hyperparameter search space for the one-dimensional steady-state advection equation problem.

Method	Search Space
PI-BLS	$N_c \in \{5, 13, 21\}$, $p \in \{10, 30, 50\}$, $q \in \{25, 65, 105\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
PINN	$N_h \in \{3, 23, \ldots, 203\}$, $L_h = 2$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
PI-ELM	$N_h \in \{3, 23, \ldots, 203\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
B-PI-ELM	$N_h \in \{3, 23, \ldots, 203\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$

Table 3: Hyperparameter search space for the two-dimensional Poisson equation.

Method	Search Space
PI-BLS	$N_c \in \{5, 13, 21\}$, $p \in \{10, 30, 50\}$, $q \in \{25, 65, 105\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\lambda_{\text{reg}} \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
PINN	$N_h \in \{3, 23, \ldots, 203\}$, $L_h = 2$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
PI-ELM	$N_h \in \{3, 23, \ldots, 203\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
B-PI-ELM	$N_h \in \{3, 23, \ldots, 203\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$

Table 4: Hyperparameter search space for the diffusion–reaction equation.

Method	Search Space
PI-BLS	$N_c \in \{5, 13, 21\}$, $p \in \{10, 30, 50\}$, $q \in \{25, 65, 105\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\lambda_I \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
PINN	$N_h \in \{3, 23, \ldots, 203\}$, $L_h = 2$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\lambda_I \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
PI-ELM	$N_h \in \{3, 23, \ldots, 203\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\lambda_I \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$
B-PI-ELM	$N_h \in \{3, 23, \ldots, 203\}$, $\lambda_b \in \{10^{-4}, 10^{-2}, 1\}$, $\lambda_I \in \{10^{-4}, 10^{-2}, 1\}$, $\sigma \in \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}\}$

6.1.5 Metrics

To assess the accuracy of the predicted solutions, we evaluate the methods using the metrics tabulated in Table 5.

6.2 Discussion: PI-BLS for Steady Advection Equation

The one-dimensional advection equation is expressed as follows:

$$\boldsymbol{u}_x = R(x),\ 0 < x \leq 1. \tag{6.1}$$

The expressions for R and the Dirichlet boundary conditions are derived based on the following exact solution:

$$\boldsymbol{u}(x) = \sin(2\pi x)\cos(4\pi x) + 1. \tag{6.2}$$

The differential operator $\mathcal{N}_x[\,\cdot\,]$ for these problems is $\frac{\partial}{\partial x}$. The boundary operator is the identity for all the problems.

The expressions for the error function due to PDE residual is given by:

$$\mathcal{J}_u = \widetilde{\boldsymbol{u}}_x(x_i) - R(x_i) \ \text{ on } \ \{x_{i,u}\}_{i=1}^{N_c}. \tag{6.3}$$

The error function for the boundary condition is given by

$$\mathcal{J}_b = \widetilde{\boldsymbol{u}}(x) - B(x) \ \text{ on } \ \{x_{i,b}\}_{i=1}^{N_b}.$$

Table 5: Performance evaluation metrics. Here, N is the total number of test samples, $\boldsymbol{u}(X)$ is the exact solution, $\widetilde{\boldsymbol{u}}(X)$ is the predicted solution, and $\bar{\boldsymbol{u}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{u}(X_i)$.

Metric	**Expression**
MSE	$\dfrac{1}{N}\sum_{i=1}^{N}\left(\widetilde{\boldsymbol{u}}(X_i) - \boldsymbol{u}(X_i)\right)^2$
RMSE	$\sqrt{\dfrac{1}{N}\sum_{i=1}^{N}\left(\widetilde{\boldsymbol{u}}(X_i) - \boldsymbol{u}(X_i)\right)^2}$
MAE	$\dfrac{1}{N}\sum_{i=1}^{N}\lvert\widetilde{\boldsymbol{u}}(X_i) - \boldsymbol{u}(X_i)\rvert$
Relative MSE	$\dfrac{\sum_{i=1}^{N}(\widetilde{\boldsymbol{u}} - \boldsymbol{u})^2}{\sum_{i=1}^{N}\boldsymbol{u}^2}$
Relative RMSE (L^2 Error)	$\dfrac{\sqrt{\sum_{i=1}^{N}(\widetilde{\boldsymbol{u}} - \boldsymbol{u})^2}}{\sqrt{\sum_{i=1}^{N}\boldsymbol{u}^2}}$
Relative MAE	$\dfrac{\sum_{i=1}^{N}\lvert\widetilde{\boldsymbol{u}} - \boldsymbol{u}\rvert}{\sum_{i=1}^{N}\lvert\boldsymbol{u}\rvert}$
L_∞ Error	$\max\limits_{1\le i\le N}\lvert\widetilde{\boldsymbol{u}}(X_i) - \boldsymbol{u}(X_i)\rvert$
Relative L_∞ Error	$\dfrac{\max_{1\le i\le N}\lvert\widetilde{\boldsymbol{u}} - \boldsymbol{u}\rvert}{\max_{1\le i\le N}\lvert\boldsymbol{u}\rvert}$
R^2 Score	$1 - \dfrac{\sum_{i=1}^{N}(\widetilde{\boldsymbol{u}} - \boldsymbol{u})^2}{\sum_{i=1}^{N}(\boldsymbol{u} - \bar{\boldsymbol{u}})^2}$

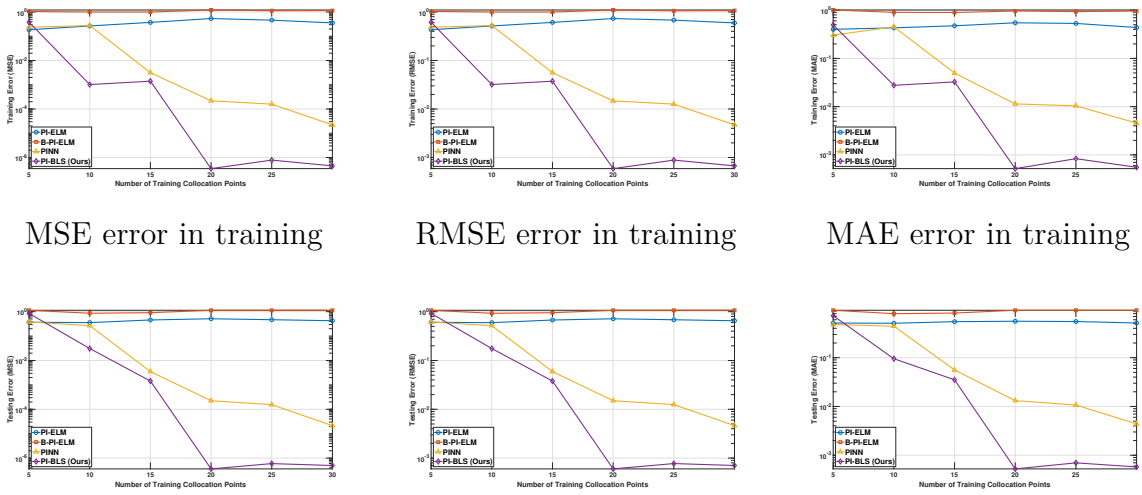
Table 6: Comparison of performance metrics for PI-BLS, PINN, PI-ELM, and B-PI-ELM methods for the advection equation with 30 collocation points.

	PI-BLS		PINN		PI-ELM		B-PI-ELM	
Parameter	Train	Test	Train	Test	Train	Test	Train	Test
MSE	4.5149e-07	4.9807e-07	2.2248e-05	2.1427e-05	3.5523e-01	4.2637e-01	1.1356e+00	1.1407e+0
RMSE	6.7193e-04	7.0573e-04	4.7167e-03	4.6290e-03	5.9601e-01	6.5297e-01	1.0656e+00	1.0680e+00
MAE	5.5396e-04	5.7593e-04	4.5665e-03	4.4032e-03	4.3809e-01	5.1484e-01	9.5790e-01	9.3827e-01
Relative MSE	3.9758e-07	4.3665e-07	1.9591e-05	1.8785e-05	3.1281e-01	3.7379e-01	1.0000e+00	1.0000e+00
Relative RMSE	6.3054e-04	6.6079e-04	4.4262e-03	4.3342e-03	5.5930e-01	6.1138e-01	1.0000e+00	1.0000e+00
Relative MAE	5.7830e-04	6.1382e-04	4.7672e-03	4.6928e-03	4.5734e-01	5.4870e-01	1.0000e+00	1.0000e+00

In this example, we directly calculated the derivatives of the approximate functions for the listed PDEs. The coefficient matrices are given by:

$$H_{\text{col}} = \frac{\partial \mathcal{G}}{\partial x}(X_{\text{col}}) = \begin{bmatrix}\mathbf{F}_x(X_{\text{col}}) & \mathbf{E}_x(X_{\text{col}})\end{bmatrix} \quad \text{and} \quad H_b = \begin{bmatrix}E(X_b) & F(X_b)\end{bmatrix}.$$

Table 6 presents a comparative performance analysis of the proposed PI-BLS against



MSE error in training | RMSE error in training | MAE error in training

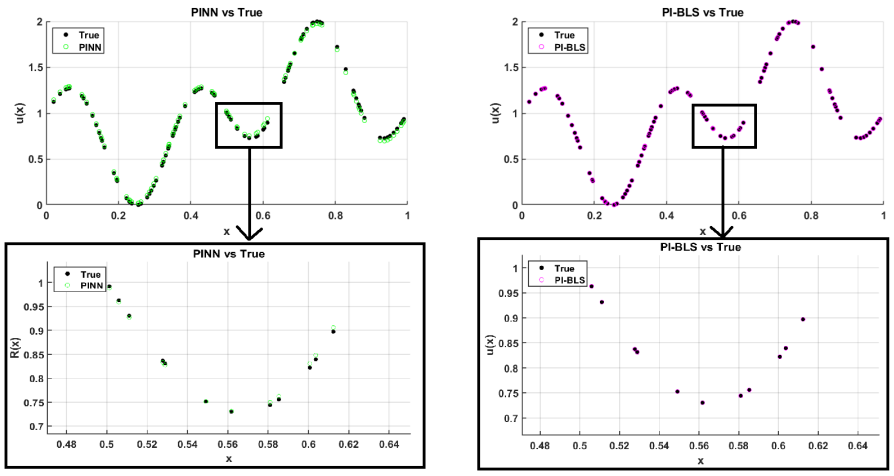
MSE error in testing | RMSE error in testing | MAE error in testing

Figure 4: Comparison of various error metrics for the proposed PI-BLS, PINN, PI-ELM, and B-PI-ELM methods across different numbers of training points for the advection equation.

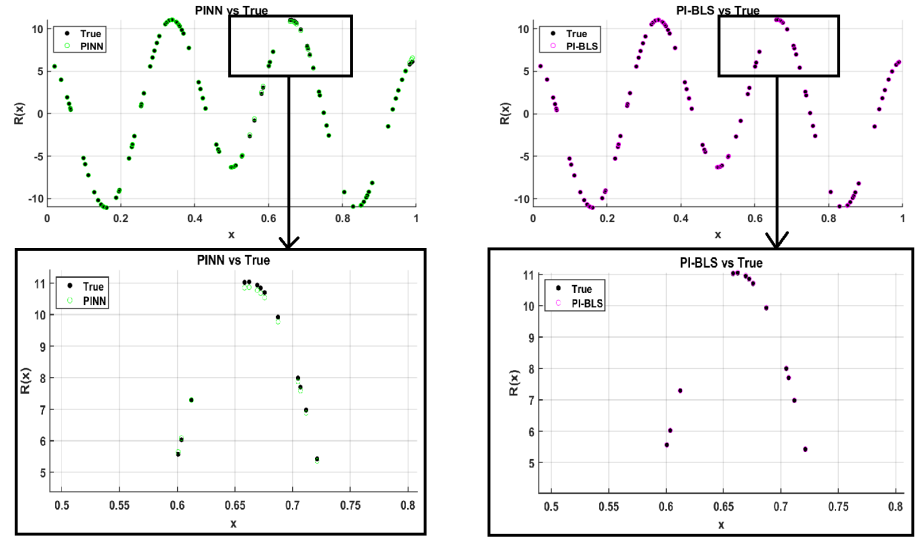
PINN, PI-ELM, and B-PI-ELM for solving the advection equation using 30 training collocation points. As observed, PI-BLS consistently outperforms the competing methods across all evaluation metrics. It achieves the lowest training and testing values of MSE, RMSE, MAE, and their corresponding relative error measures, demonstrating its superior approximation accuracy and generalization capability. On the testing dataset, PI-BLS attains an MSE of 4.9807×10^{-7}, which is approximately 43 times smaller than that of PINN and several orders of magnitude lower than those of PI-ELM and B-PI-ELM. Similar improvements are observed for RMSE, MAE, and the corresponding relative error metrics. Moreover, the relative MSE, relative RMSE, and relative MAE obtained by PI-BLS remain in the range of 10^{-7}–10^{-4}, whereas PI-ELM and B-PI-ELM exhibit significantly larger normalized errors, with B-PI-ELM approaching unity. These results demonstrate the superior numerical accuracy, stability, and robustness of the proposed PI-BLS framework.

Figure 4 illustrates the variation of the training and testing MSE, RMSE, and MAE with different numbers of training collocation points. The proposed PI-BLS consistently yields the smallest errors for all metrics, with both training and testing errors decreasing rapidly as the number of collocation points increases and eventually converging to very small values. In contrast, PINN, PI-ELM, and B-PI-ELM exhibit comparatively larger errors throughout the experiments, indicating slower convergence and reduced prediction accuracy. These results further confirm the effectiveness and scalability of PI-BLS for physics-informed learning.

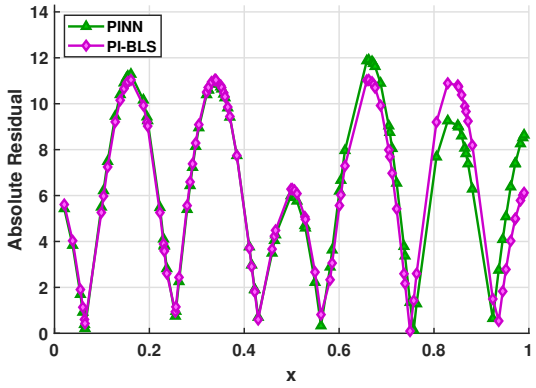
Figure 5 provides a qualitative comparison of the approximate solutions obtained by PI-BLS and PINN with the corresponding exact solution. As shown in Fig. 5(a), both methods successfully capture the overall solution profile; however, the PI-BLS solution



(a) Approximate vs true solution



(b) Approximate vs exact $R(x)$



(c) Absolute residual varying x

Figure 5: Comparison of the approximate solution vs true solution for the advection equation.

exhibits a closer agreement with the exact solution across the entire computational domain. The comparison of the computed and exact right-hand side $R(x)$ in Fig. 5(b) further demonstrates that PI-BLS more accurately reproduces the governing physical relationship than PINN. This observation is corroborated by the absolute residual distributions shown in Fig. 5(c), where PI-BLS consistently produces smaller residuals, indicating a more accurate satisfaction of the governing equation and enhanced numerical reliability.

6.3 Discussion: PI-BLS for Two-Dimensional Steady Poisson Equation

We consider a 2D Poisson problem [41]

$$-(\boldsymbol{u}_{xx} + \boldsymbol{u}_{yy}) = R(x, y), \quad (x, y) \in [0, 1]^2,$$

where R is obtained from the exact solution

$$\boldsymbol{u}(x, y) = 2^{4a} x^a (1 - x)^a y^a (1 - y^a).$$

We consider the zero Dirichlet boundary conditions on the boundary.

The PI-BLS loss functions are given as follows:

$$\text{PDE loss:} \quad \mathcal{J}_u = \widetilde{\boldsymbol{u}}_{xx}(x, y) + \widetilde{\boldsymbol{u}}_{yy}(x, y) - R(x, y) \ \text{ on } \ \{x_{i,u}, y_{i,u}\}_{i=1}^{N_c}. \tag{6.4}$$

$$\text{Boundary loss:} \quad \mathcal{J}_b = \widetilde{\boldsymbol{u}}(x, y) - B(x, y) \ \text{ on } \ \{x_{i,b}, y_{i,b}\}_{i=1}^{N_b}, \tag{6.5}$$

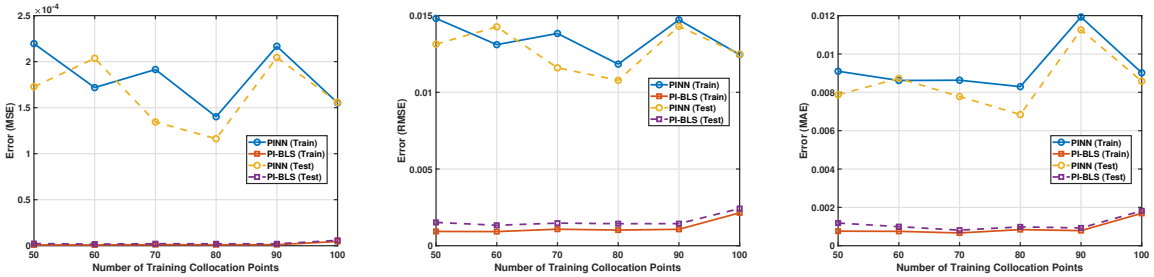
where the input $X = [x,\, y]$. The formulation for the coefficient matrices is given by:

$$\begin{aligned} H_{\text{col}} &= \frac{\partial^2 \mathcal{G}}{\partial x^2}(X_{\text{col}}) + \frac{\partial^2 \mathcal{G}}{\partial y^2}(X_{\text{col}}) \\ &= \begin{bmatrix} -(\mathbf{F}_{xx,\text{col}} + \mathbf{F}_{yy,\text{col}}) & -(\mathbf{E}_{xx,\text{col}} + \mathbf{E}_{yy,\text{col}}) \end{bmatrix}, \\ H_b &= \begin{bmatrix} E(X_b) & F(X_b) \end{bmatrix}. \end{aligned}$$

Table 7 compares the performance of the proposed PI-BLS and the conventional PINN for the two-dimensional Poisson equation. In the experiment, the number of training collocation points is set to $N_{\text{train}} = 100$, while the testing dataset consists of $N_{\text{test}} = 10,000$ collocation points. Further, in this example, we consider the error matrices ℓ_∞ Error, Relative ℓ_∞ Error, and R^2 Score. The results show that PI-BLS consistently outperforms PINN across all evaluation metrics. Specifically, PI-BLS achieves substantially lower MSE, RMSE, MAE, and ℓ^∞ errors than PINN, indicating a more accurate approximation of the exact solution. On the testing dataset, the MSE is reduced from 1.5591×10^{-4} to 5.8687×10^{-6}, representing more than one order of magnitude improvement. Likewise, the RMSE, MAE, and ℓ^∞ errors are all approximately five times smaller than those obtained by PINN. Similar improvements are observed for the relative error metrics, while the R^2 score of PI-BLS (0.99994) remains closer to the ideal value of one than that of PINN (0.99836).

Table 7: Comparison of performance metrics for PI-BLS and PINN for the Poisson equation in 2D problem with 100 training collocation points.

	PI-BLS		PINN	
Parameter	**Train**	**Test**	**Train**	**Test**
MSE	4.5950e-06	5.8687e-06	1.5526e-04	1.5591e-04
RMSE	2.1436e-03	2.4225e-03	1.2460e-02	1.2486e-02
MAE	1.6879e-03	1.8196e-03	9.0178e-03	8.5803e-03
ℓ_∞	5.7685e-03	1.6726e-02	4.5706e-02	8.3972e-02
Relative MSE	3.3228e-06	3.9407e-06	1.1228e-04	1.0469e-04
Relative RMSE	1.8229e-03	1.9851e-03	1.0596e-02	1.0232e-02
Relative MAE	1.4842e-03	1.5411e-03	7.9297e-03	7.2672e-03
Relative ℓ_∞	3.8460e-03	1.1151e-02	3.0473e-02	5.5982e-02
R^2 Score	9.9995e-01	9.9994e-01	9.9827e-01	9.9836e-01



MSE RMSE MAE

Figure 6: Comparison of various error metrics for the proposed PI-BLS and PINN methods across different numbers of training points for the Poisson 2D equation.

These results demonstrate that PI-BLS provides superior approximation accuracy and generalization capability, even when trained using only 100 collocation points.

Figure 6 further illustrates the variation of the MSE, RMSE, and MAE as the number of training collocation points increases from 50 to 100. Across all training sizes, PI-BLS consistently maintains significantly lower training and testing errors than PINN. Moreover, the training and testing curves of PI-BLS remain close to each other with relatively small fluctuations, indicating stable convergence, strong generalization, and robustness with respect to the number of training collocation points. These results confirm that the proposed PI-BLS is a more accurate and reliable physics-informed solver than the conventional PINN for the two-dimensional Poisson equation.

Table 8: Comparison of performance metrics for PI-BLS, PINN, PI-ELM, and B-PI-ELM for diffusion-reaction for 100 training collocation points.

	PI-BLS		PINN		PI-ELM		B-PI-ELM	
Parameter	Train	Test	Train	Test	Train	Test	Train	Test
MSE	8.2170e-02	8.7424e-02	2.3443e-01	3.1215e-01	1.8193e-01	2.2393e-01	1.8029e-01	2.5049e-01
RMSE	2.8665e-01	2.9568e-01	4.8418e-01	5.5870e-01	4.2653e-01	4.7321e-01	4.2460e-01	5.0049e-01
MAE	2.4843e-01	2.5334e-01	4.0089e-01	4.5368e-01	3.3576e-01	3.6853e-01	3.4144e-01	3.9102e-01
Relative MSE	3.7369e-01	2.8537e-01	1.0661e+00	1.0189e+00	7.1826e-01	7.3093e-01	8.1991e-01	8.1763e-01
Relative RMSE	6.1130e-01	5.3420e-01	1.0325e+00	1.0094e+00	8.4750e-01	8.5494e-01	9.0549e-01	9.0423e-01
Relative MAE	6.3950e-01	5.5976e-01	1.0320e+00	1.0024e+00	8.1043e-01	8.1427e-01	8.7894e-01	8.6396e-01

6.4 Discussion: Unsteady Case :- Diffusion–Reaction Equation

We consider the following one-dimensional unsteady diffusion–reaction problem described by the PDE taken as in [41]:

$$\frac{\partial \boldsymbol{u}}{\partial t} = \mu \frac{\partial^2 \boldsymbol{u}}{\partial x^2} + R(x,t), \ (x,t) \in [-\pi, \pi] \times [0,1], \tag{6.6}$$

where $\mu = 1$ and

$$R(x,t) = e^{-t}\left(\frac{3}{2}\sin(2x) + \frac{8}{3}\sin(3x)\frac{15}{4}\sin(4x) + \frac{63}{8}\sin(8x)\right).$$

The initial and boundary conditions are given by

$$\begin{aligned} \boldsymbol{u}(x,0) &= \sum_{i=1}^{4} \frac{\sin(ix)}{i} + \frac{\sin(8x)}{8} \\ \boldsymbol{u}(-\pi,t) &= \boldsymbol{u}(\pi,t) = 0. \end{aligned}$$

The exact solution of the PDE is given by

$$\boldsymbol{u}(x,t) = e^{-t}\left(\sum_{i=1}^{4} \frac{\sin(ix)}{i} + \frac{\sin(8x)}{8}\right).$$

The PI-BLS equations for these functions are given as follows:

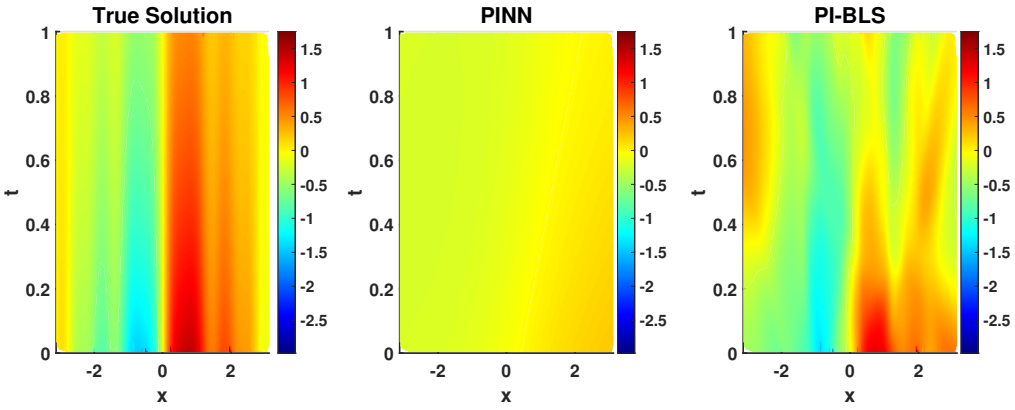
$$\text{PDE loss:} \quad \mathcal{J}_u = \widetilde{\boldsymbol{u}}_t(x,t) - \mu\, \widetilde{\boldsymbol{u}}_x(x,t) - R(x,t) \ \text{ on } \ \{x_{i,u}, t_{i,u}\}_{i=1}^{N_c}. \tag{6.7}$$

$$\text{Boundary loss:} \quad \mathcal{J}_b = \widetilde{\boldsymbol{u}}(x,t) - B(x,t) \ \text{ on } \ \{x_{i,b}, t_{i,b}\}_{i=1}^{N_b}. \tag{6.8}$$

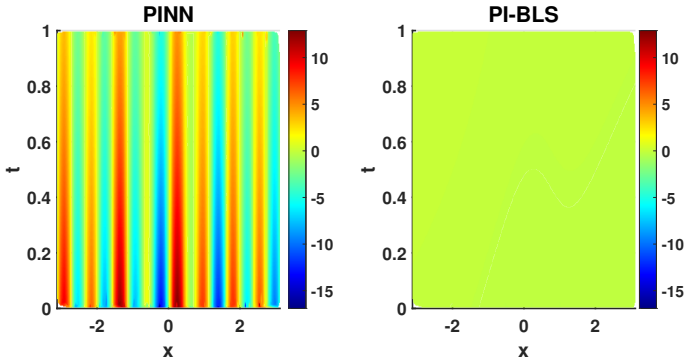
$$\text{Initial condition loss:} \quad \mathcal{J}_I = \widetilde{\boldsymbol{u}}(x,0) - \mathcal{I}(t) \ \text{ on } \ \{x_{i,I}\}_{i=1}^{N_I}. \tag{6.9}$$

The formula for the coefficient matrix is given by:

$$H_{\text{col}} = \frac{\partial \mathcal{G}}{\partial t}(X_{\text{col}}) - \frac{\partial \mathcal{G}}{\partial x}(X_{\text{col}}) = \begin{bmatrix} \mathbf{F}_{t,\text{col}} - \mu \mathbf{F}_{x,\text{col}} & \mathbf{E}_{t,\text{col}} - \mu \mathbf{E}_{x,\text{col}} \end{bmatrix}.$$

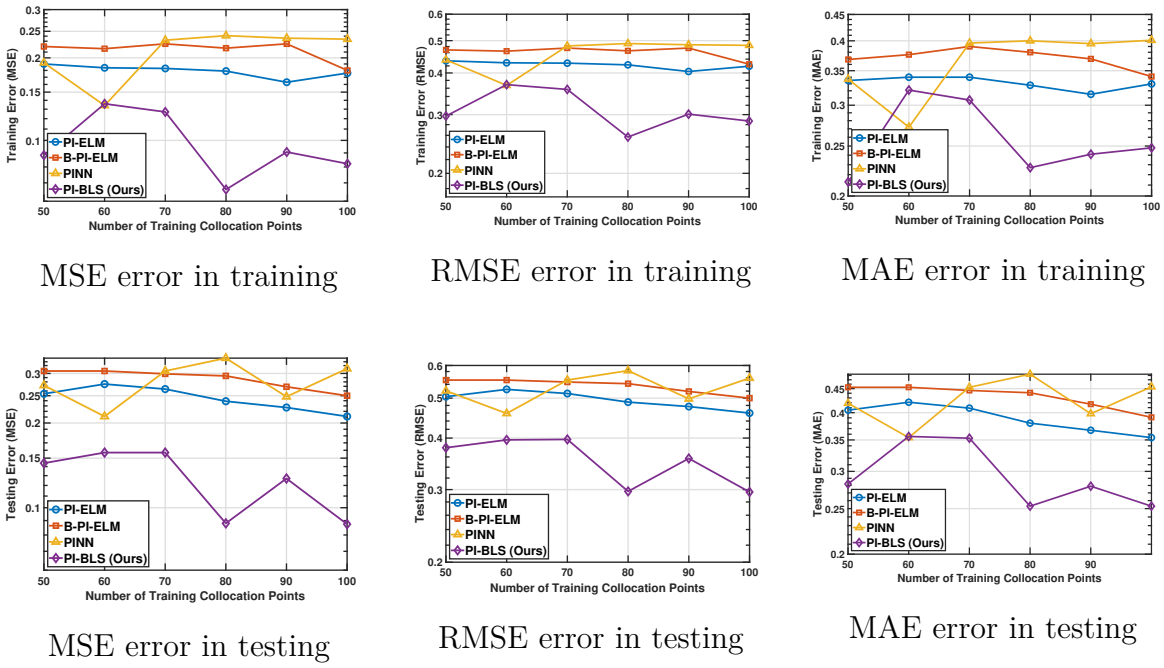


(a) Approximate vs true solution



(b) Absolute residual varying x

Figure 7: Comparison of the approximate solution vs true solution for the diffusion-reaction equation for the number of training points equal 50.



MSE error in training | RMSE error in training | MAE error in training

MSE error in testing | RMSE error in testing | MAE error in testing

Figure 8: Comparison of various error metrics for the proposed PI-BLS, PINN, PI-ELM, and B-PI-ELM methods across different numbers of training points for the diffusion reaction.

Table 8 compares the performance of PI-BLS, PINN, PI-ELM, and B-PI-ELM for the diffusion–reaction equation using 100 training collocation points. Among all methods, PI-BLS consistently achieves the lowest training and testing errors across all evaluation metrics, including MSE, RMSE, MAE, Relative MSE, Relative RMSE, and Relative MAE. The close agreement between the training and testing errors also indicates good generalization capability. In contrast, PINN, PI-ELM, and B-PI-ELM exhibit comparatively larger errors, demonstrating that PI-BLS provides the most accurate and robust approximation for this problem.

Figure 7 compares the true solution, approximate solutions, and absolute residuals obtained by PINN and the proposed PI-BLS method for the diffusion–reaction equation using 50 training collocation points. As shown in Fig. 7(a), the PI-BLS solution provides a much more accurate approximation to the true solution, successfully capturing the spatial and temporal variations of the exact profile. In comparison, the PINN approximation exhibits noticeable deviations from the true solution in several regions of the computational domain. Further, the absolute residual distributions presented in Fig. 7(b) highlight the superior performance of PI-BLS, which maintains consistently lower residuals across the computational domain. These results demonstrate that PI-BLS provides a more accurate and stable approximation than PINN, even with a relatively small number of training collocation points.

Figure 8 compares the training and testing performance of PI-BLS with PI-ELM, BPI-

ELM, and PINN using MSE, RMSE, and MAE across different numbers of training collocation points. The results consistently show that PI-BLS achieves the lowest error values for all three metrics in both the training and testing phases, demonstrating its superior approximation accuracy and better performance in solving the diffusion-reaction equation. In contrast, PI-ELM, BPI-ELM, and PINN exhibit comparatively higher error trends as the number of collocation points increases. These observations confirm that the proposed PI-BLS framework provides more accurate and robust solutions for the diffusion-reaction equation over a wide range of testing data sizes.

6.5 CPU Time Comparison for Proposed PI-BLS vs PINN

Table 9: Comparison of training time and time per collocation point for PINN and PI-BLS across different test cases, averaged over 10 runs. The experiments use varying test data sizes, with 30 training collocation points for the advection equation and 500 for both Poisson 2D and diffusion-reaction.

	Advection equation		Poisson equation		Diffusion-reaction equation	
	PINN	PI-BLS	PINN	PI-BLS	PINN	PI-BLS
Training Time (s)	5.0670	3.8257	7.7663	7.2153	6.4458	4.9763
Time per Collocation Point (s)	1.69×10^{-1}	1.28×10^{-1}	1.55×10^{-2}	1.44×10^{-2}	1.29×10^{-2}	9.95×10^{-3}

Table 9 compares the computational efficiency of PINN and the proposed PI-BLS for all three test problems in terms of total training time in seconds (s) and time per collocation point. The results show that PI-BLS consistently outperforms PINN, reducing the total training time from 6.4458 s to 4.9763 s, corresponding to an improvement of approximately 22.8%. Likewise, the average computation time per collocation point for the diffusion-reaction equation decreases from 1.29×10^{-2} s for PINN to 9.95×10^{-3} s for PI-BLS, representing a reduction of about 22.9%. These findings demonstrate that PI-BLS provides a more computationally efficient framework while preserving solution accuracy, making it a promising alternative to PINNs for solving PDEs.

6.6 Comparison of Trainable Parameters for Proposed PI-BLS vs PINN

Table 10 presents a comparison of the number of trainable parameters between the conventional PINN and the proposed PI-BLS under their respective optimal hyperparameter configurations. It is evident that PI-BLS consistently requires substantially fewer trainable parameters than PINN across all three benchmark problems. Specifically, PI-BLS employs only 155–755 trainable parameters, whereas the corresponding PINN models require between 21,165 and 42,225 trainable parameters.

For the one-dimensional advection equation, PI-BLS reduces the number of trainable parameters by approximately 98.3%–99.4%, requiring only 235–495 parameters compared

Table 10: Comparison of the number of trainable parameters between PINN and the proposed PI-BLS under the optimal hyperparameter configuration. The reduction factor (Red.) is computed as PINN/PI-BLS. The "×" column represents the parameter reduction factor ($\frac{\text{PINN}}{\text{PI-BLS}}$), i.e., the number of times fewer trainable parameters are required by PI-BLS than by PINN.

Advection Equation					**Poisson equation**					**Diffusion–reaction equation**				
$\mathbf{N}_{\text{train}}$	PINN	PI-BLS	Red. (%)	×	$\mathbf{N}_{\text{train}}$	PINN	PI-BLS	Red. (%)	×	$\mathbf{N}_{\text{train}}$	PINN	PI-BLS	Red. (%)	×
5	34405	455	98.68	75.62	50	42225	155	99.63	272.42	50	27385	755	97.24	36.27
10	21165	355	98.32	59.62	60	42225	155	99.63	272.42	60	27385	235	99.14	116.53
15	34405	495	98.56	69.51	70	42225	155	99.63	272.42	70	34405	235	99.32	146.40
20	34405	235	99.32	146.40	80	21165	155	99.27	136.55	80	27385	735	97.32	37.26
25	34405	235	99.32	146.40	90	21165	155	99.27	136.55	90	21165	735	96.53	28.80
30	42022	235	99.44	178.82	100	21165	155	99.27	136.55	100	27385	235	99.14	116.53

with 21,165–42,022 parameters for PINN. This corresponds to a reduction factor ranging from approximately 60× to 179×. For the two-dimensional Poisson equation, PI-BLS consistently utilizes only 155 trainable parameters for all training configurations, achieving a parameter reduction of approximately 99.3%–99.6%, or equivalently, requiring about 137× to 272× fewer trainable parameters than PINN. Similarly, for the diffusion–reaction equation, PI-BLS requires only 235–755 trainable parameters, yielding a reduction of 96.5% to 99.3% compared to the corresponding PINNs, which translates to approximately 29× to 146× fewer trainable parameters.

The remarkable reduction in trainable parameters originates from the underlying learning mechanism of PI-BLS. Unlike PINNs, where all network weights and biases are optimized through iterative backpropagation, PI-BLS fixes the randomly generated feature and enhancement nodes and optimizes only the output-layer parameters through a linear least-squares formulation. Consequently, the optimization problem is significantly smaller, resulting in a drastic reduction in the number of unknown variables. This compact parameterization not only lowers the optimization complexity but also provides a explanation for the superior computational efficiency and reduced training time of PI-BLS demonstrated in the subsequent experiments.

7 Conclusion and Future Directions

This paper presented a novel PI-BLS for solving forward PDEs by combining the computational efficiency of broad RdNNs with the physics-informed learning paradigm. Unlike conventional PINNs, the proposed framework reformulates the physics-informed learning problem as a linear least-squares optimization task, thereby eliminating the need for iterative gradient-based backpropagation while preserving the governing physical laws through the differential equation and the associated boundary and initial conditions. Furthermore, we established the theoretical foundations of the proposed framework by analyzing its stability, residual error bound, sensitivity to perturbations, and physics consistency, providing rigorous guarantees for the robustness and reliability of PI-BLS.

The effectiveness of the proposed framework was extensively evaluated on three representative benchmark problems, namely the one-dimensional advection equation, the two-dimensional Poisson equation, and the diffusion–reaction equation. Experimental comparisons with PINN, PI-ELM, and B-PI-ELM consistently demonstrated the superiority of PI-BLS in terms of both predictive accuracy and computational efficiency. Across different benchmark problems, PI-BLS achieved lower training and testing errors while requiring substantially fewer trainable parameters than conventional PINNs. In particular, PI-BLS reduced the number of trainable parameters by approximately 96.5%–99.6%, corresponding to up to 272× fewer optimization variables. This compact formulation also translated into significant computational gains, where the proposed framework reduced the average training time by approximately 24.6%, 46.3%, and 24.0% for the advection, Poisson, and diffusion–reaction equations, respectively. Moreover, PI-BLS consistently produced lower approximation errors across all benchmark problems; for example, in the two-dimensional Poisson problem, it reduced the testing RMSE from 1.2486×10^{-2} to 2.4225×10^{-3}, representing an improvement of approximately 80.6% over the conventional PINN. Overall, the proposed PI-BLS establishes an efficient and scalable alternative to conventional gradient-based physics-informed learning.

Limitations and Future Work: The proposed PI-BLS framework has some limitations that motivate future research:

- The current formulation is designed for forward PDEs with known governing equations. Extending PI-BLS to inverse PDE problems and parameter identification constitutes an important future research direction.

- The present framework is primarily validated on linear PDEs. Extending PI-BLS to nonlinear PDEs will require solving nonlinear least-squares optimization problems, for which efficient iterative solvers such as the Gauss–Newton or Levenberg–Marquardt methods may be incorporated.

- The randomized feature generation and dense least-squares formulation may become computationally demanding for large-scale or high-dimensional problems. Future work will investigate adaptive broad architectures, sparse representations, and parallel implementations to further improve scalability and efficiency.

Acknowledgment

Pinki Khatun acknowledges the financial support provided by the Indian Institute of Technology Gandhinagar (IIT Gandhinagar) provided through the project grant No. IP/52012. This work was initiated while Pinki Khatun was a Postdoctoral Fellow at the IIT Gandhinagar, India. The author gratefully acknowledges the support and research environment provided by the institute during that period. M. Sajid acknowledges the Indian Institute of Technology Indore for providing financial support through the Translational

Research Fellowship (TRF). Abhinav Jha acknowledges partial support by the IIT Gandhinagar Internal Project: IP/52016 and INSPIRE Faculty Fellowship Research Grant: DST/INSPIRE/04/2024/000202.

References

[1] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18 (153):1–43, 2018.

[2] J. Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[3] S. Berrone, C. Canuto, and M. Pintore. Variational physics informed neural networks: the role of quadratures and test functions. *Journal of Scientific Computing*, 92(3): 100, 2022.

[4] S. Cai, Z. Wang, S. Wang, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

[5] C. P. Chen and Z. Liu. Broad learning system: An effective and efficient incremental learning system without the need for deep architecture. *IEEE Transactions on Neural Networks and Learning Systems*, 29(1):10–24, 2017.

[6] C. P. Chen, Z. Liu, and S. Feng. Universal approximation capability of broad learning system and its structural variations. *IEEE Transactions on Neural Networks and Learning Systems*, 30(4):1191–1204, 2018.

[7] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

[8] M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994.

[9] V. Dwivedi and B. Srinivasan. Physics informed extreme learning machine (pielm)–a rapid method for the numerical solution of partial differential equations. *Neurocomputing*, 391:96–118, 2020.

[10] M. S. Eshaghi, C. Anitescu, M. Thombre, Y. Wang, X. Zhuang, and T. Rabczuk. Variational physics-informed neural operator (vino) for solving partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 437:117785, 2025.

[11] L. Gu, S. Qin, L. Xu, and R. Chen. Physics-informed neural networks with domain decomposition for the incompressible navier–stokes equations. *Physics of Fluids*, 36 (2), 2024.

[12] J. Guo, H. Zhu, Y. Yang, and C. Guo. Advances in physics-informed neural networks for solving complex partial differential equations and their engineering applications: A systematic review. *Engineering Applications of Artificial Intelligence*, 161:112044, 2025.

[13] Z. Hao, J. Yao, C. Su, H. Su, Z. Wang, F. Lu, Z. Xia, Y. Zhang, S. Liu, L. Lu, et al. Pinnacle: A comprehensive benchmark of physics-informed neural networks for solving pdes. *Advances in Neural Information Processing Systems*, 37:76721–76774, 2024.

[14] A. D. Jagtap and G. E. Karniadakis. Extended physics-informed neural networks (xpinns): A generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Communications in Computational Physics*, 28(5), 2020.

[15] A. D. Jagtap, E. Kharazmi, and G. E. Karniadakis. Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365: 113028, 2020.

[16] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[17] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560, 2021.

[18] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks and Learning Systems*, 9(5):987–1000, 1998.

[19] J. Lee, S. Shin, H. Choi, A. Lee, B. Park, and S. Lee. Extended multiphysics-informed neural network for conjugate heat transfer problems. *International Journal of Heat and Mass Transfer*, 246:127098, 2025.

[20] X. Liu, W. Yao, W. Peng, and W. Zhou. Bayesian physics-informed extreme learning machine for forward and inverse pde problems with noisy data. *Neurocomputing*, 549: 126425, 2023.

[21] M. Loś, T. Służalec, P. Maczuga, A. Vilkha, C. Uriarte, and M. Paszyński. Collocation-based robust variational physics-informed neural networks (crvpinns). *Computers & Structures*, 316:107839, 2025.

[22] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM Review*, 63(1):208–228, 2021.

[23] K. Luo, J. Zhao, Y. Wang, J. Li, J. Wen, J. Liang, H. Soekmadji, and S. Liao. Physics-informed neural networks for pde problems: A comprehensive review. *Artificial Intelligence Review*, 58(10):323, 2025.

[24] Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360: 112789, 2020.

[25] G. Pang, L. Lu, and G. E. Karniadakis. fpinns: Fractional physics-informed neural networks. *SIAM Journal on Scientific Computing*, 41(4):A2603–A2626, 2019.

[26] Y.-H. Pao, G.-H. Park, and D. J. Sobajic. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2):163–180, 1994.

[27] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop on Autodiff*, 2017.

[28] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

[29] F. Sahli Costabal, Y. Yang, P. Perdikaris, D. E. Hurtado, and E. Kuhl. Physics-informed neural networks for cardiac activation mapping. *Frontiers in Physics*, 8:42, 2020.

[30] M. Sajid, A. K. Malik, and M. Tanveer. Intuitionistic fuzzy broad learning system: Enhancing robustness against noise and outliers. *IEEE Transactions on Fuzzy Systems*, 32(8):4460–4469, 2024.

[31] W. F. Schmidt, M. A. Kraaijveld, R. P. Duin, et al. Feed forward neural networks with random weights. In *International conference on pattern recognition*, pages 1–1. IEEE Computer Society Press, 1992.

[32] S. Son, H. Lee, D. Jeong, K.-Y. Oh, and K. H. Sun. A novel physics-informed neural network for modeling electromagnetism of a permanent magnet synchronous motor. *Advanced Engineering Informatics*, 57:102035, 2023.

[33] M. Tanveer, A. Mishra, M. Sajid, A. Quadir, A. D. N. Initiative, et al. Bls-cil: Class imbalance broad learning system via dual weighting and layer trimming. *Pattern Recognition*, page 112940, 2025.

[34] D. Wang and M. Li. Stochastic configuration networks: Fundamentals and algorithms. *IEEE Transactions on Cybernetics*, 47(10):3466–3479, 2017.

[35] S. Wang, Y. Teng, and P. Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[36] S. Wang, X. Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

[37] S. Wang, B. Li, P. Perdikaris, et al. Gradient alignment in physics-informed neural networks: A second-order optimization perspective. *Advances in Neural Information Processing Systems*, 38:168482–168532, 2026.

[38] Y. Wang, Y. Peng, Z. Hu, and Y. Li. Exploring prototype-guided strategy for domain decomposition in physics-informed neural network. *Nonlinear Dynamics*, 113(11): 12487–12514, 2025.

[39] X. Yang, M. Ran, and H. Pu. A novel adaptive gradient-enhanced physics-informed neural networks for solving partial differential equations. *Computers & Mathematics with Applications*, 213:242–256, 2026.

[40] A. Yazdani, L. Lu, M. Raissi, and G. E. Karniadakis. Systems biology informed deep learning for inferring parameters and hidden dynamics. *PLoS computational biology*, 16(11):e1007575, 2020.

[41] J. Yu, L. Lu, X. Meng, and G. E. Karniadakis. Gradient-enhanced physics-informed neural networks for forward and inverse pde problems. *Computer Methods in Applied Mechanics and Engineering*, 393:114823, 2022.

[42] D. Zhang, L. Lu, L. Guo, and G. E. Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[43] D. Zhang, L. Guo, and G. E. Karniadakis. Learning in modal space: Solving time-dependent stochastic pdes using physics-informed neural networks. *SIAM Journal on Scientific Computing*, 42(2):A639–A665, 2020.

[44] B. Zhu, H. Li, and Q. Zhang. Extended physics-informed extreme learning machine for linear elastic fracture mechanics. *Computer Methods in Applied Mechanics and Engineering*, 435:117655, 2025.