Theme Article (Controversies on the Usage of AI/ML for Science)

Physics-Preserving Neural Network (P^2NN) Methods for Elliptic Partial Differential Equations

Min Liu, *Purdue University, West Lafayette, IN, 47906, USA*

Zhiqiang Cai, *Purdue University, West Lafayette, IN, 47906, USA*

Abstract—Motivated by the ongoing debate over whether physics-informed neural networks (PINNs) can serve as universal solvers for partial differential equations (PDEs), this paper uses elliptic PDEs as a concrete setting to describe a family of physics-preserving neural network (P^2NN) methods. Beyond being physics-informed, P^2NN methods are designed to preserve the underlying physical principles which are often violated by PINNs. Specifically, P^2NN methods employ neural networks as approximating functions, an equivalent formulation of the governing PDEs, and physics-preserving numerical differentiation operators. For diffusion-reaction equations, the Ritz and the dual formulations, associated with the primal and dual variables respectively, provide natural optimization principles and hence preserve physics. For convection-diffusion-reaction equations, we examine the Bramble-Schatz and first-order system least-squares (LS) formulations and assess their applicability. Finally, we emphasize that a viable NN-based method also hinges on accurate numerical integration and physics-preserving numerical differentiation. These components are essential, nontrivial, and are discussed within the proposed P^2NN framework.

Neural networks (NNs) have demonstrated remarkable performance in computer vision, natural language processing, and other core areas of artificial intelligence. Recently, there has been a rapid surge of interest in leveraging NNs to solve partial differential equations (PDEs). Yet, despite the rapid proliferation of articles in recent years, NN-based numerical methods for solving PDEs in the context of science and engineering remain at an early stage. Before these approaches can become dependable tools for computational science and engineering, several foundational controversies must be confronted, many of which stem from a mismatch between data-driven learning pipelines and numerical discretization principles. In our view, the most consequential open questions are:

(i) For which applications do NNs offer clear advantages over finite elements for approximation?
(ii) How can we develop NN discretization methods that are not only physics-informed, but more importantly physics-preserving?
(iii) How can we develop reliable and efficient "training" algorithms for NN discretizations, given the underlying nonconvex optimization?
(iv) For a given task, how can we design a nearly optimal NN architecture that achieves a prescribed accuracy?

Questions (i), (iii), and (iv) have been partially addressed in [6], [10], and [17], respectively (see also references therein). We will return to Question (iii) later in this paper. The main focus of this paper is Question (ii): we use elliptic partial differential equations as a concrete setting to describe a family of physics-preserving neural network (P^2NN) methods.

Existing NN-based numerical methods for solving PDEs broadly fall into two main categories: (1) **Energy-based methods**, such as deep Ritz method [15], [25], [17], [19], which employ the Ritz formulation for the primary variable [16], and the dual neural network (DuNN) [20], which targets a complementary energy optimization for the dual variable [5], [16]. (2) **Least-squares**

XXXX-XXX
Digital Object Identifier 10.1109/XXX.0000.0000000

(LS)/residual-minimization methods, which formulate PDE residuals into manufactured least squares using various norm choices and problem forms [14], [1], [21], [24], [8], [7].

When a genuine minimization principle exists (typically self-adjoint and positive definite problems that commonly arise in continuum mechanics), energy-based methods are naturally aligned with classical variational discretizations and are, in that sense, *physics-preserving*. However, many scientific and engineering problems are non-self-adjoint and/or not positive definite, and thus lack a natural optimization principle. In such cases, one can then apply the LS principle to create a manufactured one (see, e.g., [4], [11], [3], [12], [2] for scalar elliptic PDEs). The LS framework is highly flexible: any set of consistent equations and data can, in principle, be incorporated into a single LS functional. In practice, however, balancing the various terms is delicate; poorly scaled LS functionals can degrade accuracy and slow down training. The minimal requirement for a viable LS formulation is its equivalence to the original PDE; otherwise, the physical fidelity of the resulting model is no longer guaranteed. For convection-diffusion-reaction equations, we examine the classic Bramble-Schatz and first-order system least-squares (FOSLS) formulations for their applicability.

Beyond problem formulation itself, the numerical discretization of PDEs also entails choosing an approximation function class, and designing numerical integration and differentiation operators. With the physics-preserving formulation perspective in place, we, in this paper, turn from general neural-network approximations to the specific structure of ReLU neural networks and study the numerical operators required to make the ReLU-NN-based methods both computable and physics-preserving. Although ReLU NN functions are continuous piecewise linear, the induced *physical partition* [17] of a desired NN approximation is irregular and unknown; moreover, this partition evolves throughout training. Consequently, numerical integration and differentiation are both essential and nontrivial components of NN-based discretization.

Another controversy concerns the *source and meaning of "data"* in scientific machine learning. In NN-based discretization for PDEs, "data" are typically not labeled input–output pairs from a standard supervised pipeline. Instead, they are integration (quadrature) points at which the solution is unknown, while the governing physical constraints are known. Consequently, both the selection of integration points and the associated quadrature rule must be designed with care: they largely determine whether the essential characteristics of the (unknown) solution can be captured. In addition, when the PDE solution lacks sufficient smoothness, designing physics-preserving numerical differentiation operators becomes crucial for the viability of NN-based methods. These issues, along with robust integration over a moving, implicit partition and differentiation compatible with non-smooth solutions, will be discussed in this paper.

ReLU Neural Networks

This section describes l-hidden-layer ReLU NNs as a set of continuous piecewise linear functions. ReLU refers to the rectified linear activation function defined by $\xi(s) = \max\{0, s\}$ that is a continuous piecewise linear function with one *breaking* point at $s = 0$. For $k = 1, \dots, l$, let n_k denote the number of neurons at the k^{th} hidden layer; denote by

$$\mathbf{b}^{(k)} \in R^{n_k} \text{ and } \boldsymbol{\omega}^{(k)} = \left(\boldsymbol{\omega}_1^{(k)}, \dots, \boldsymbol{\omega}_{n_k}^{(k)}\right)^T \in R^{n_k \times n_{k-1}} \quad (1)$$

the biases and weights of neurons at the k^{th} hidden layer, respectively, where $\boldsymbol{\omega}_i^{(k)} \in \mathcal{S}^{n_{k-1}}$ is the weights of the i^{th} neuron, $\mathcal{S}^{n_{k-1}}$ denotes the unit sphere in $R^{n_{k-1}}$, and $n_0 = d$. Let $\mathbf{x}^{(0)} = \mathbf{x} \in R^d$. For $k = 1, \dots, l$, define vector-valued functions $\mathbf{x}^{(k)} : R^{n_0} \to R^{n_k}$ by

$$\mathbf{x}^{(k)} = \xi\left(\boldsymbol{\omega}^{(k)}\mathbf{x}^{(k-1)} + \mathbf{b}^{(k)}\right). \quad (2)$$

A ReLU neural network with l hidden layers and n_k neurons at the k^{th} hidden layer can be defined as the collection of continuous piecewise linear functions:

$$\mathcal{M}(l) = \left\{ c_0 + \sum_{i=1}^{n_l} c_i \xi(\mathbf{x}_i^{(l)}) : c_i, b_i^{(k)} \in R, \boldsymbol{\omega}_i^{(k)} \in \mathcal{S}^{n_{k-1}} \right\}, \quad (3)$$

where $\mathbf{x}^{(l)}$ is recursively defined in (2). The constraint that the weight vector of each neuron lies on the unit sphere arises from normalization for the ReLU activation function (see, e.g., [17]). This restriction can narrow the set of solutions for a given approximating problem. Any NN function in $\mathcal{M}(l)$ is determined by the linear parameters $\mathbf{c} = (c_0, c_1, \dots, c_{n_l})^T$ and the nonlinear parameters $\mathbf{b}^{(k)}$ and $\boldsymbol{\omega}^{(k)}$ for $k = 1, \dots, l$.

Regardless of the input dimension, network depth, or the number of neurons per layer, a NN function $v(\mathbf{x}) \in \mathcal{M}(l)$ is always a continuous piecewise linear function with respect to a *physical partition* determined by the nonlinear parameters (see, e.g., [17]). The physical partition associated with an optimal NN approximation, whether for a target function or a PDE solution, can adapt to local features of the underlying problem. This adaptive partitioning helps explain the network's strong approximation capability for non-smooth or even discontinuous functions with *unknown* interface locations (see, e.g., [23], [7], [9]).

Elliptic Partial Differential Equation and Its Equivalent Formulations

Let Ω be a bounded domain in R^d ($d = 2$ or 3) with Lipschitz boundary $\partial\Omega = \bar{\Gamma}_D \cup \bar{\Gamma}_N$ and $\Gamma_D \cap \Gamma_N = \varnothing$, and let $\mathbf{n}$ be the outward unit vector normal to the boundary. Consider the following linear second-order scalar elliptic partial differential equation:

$$\begin{cases} -\mathrm{div}\,(A\nabla u) + \boldsymbol{\beta}\cdot\nabla u + c\,u &= f, \quad \text{in } \Omega, \\ u &= g_D, \quad \text{on } \Gamma_D, \\ \mathbf{n}\cdot A\nabla u &= g_N, \quad \text{on } \Gamma_N, \end{cases} \tag{4}$$

where $\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_d}\right)^T$ and div are the gradient and divergence operators, respectively; $A(\mathbf{x})$ is the diffusion coefficient, a $d \times d$ symmetric matrix-valued function in $L^2(\Omega)^{d\times d}$; $\boldsymbol{\beta}(\mathbf{x}) = (\beta_1, \cdots, \beta_d)^T \in L^2(\Omega)^d$ is the advective velocity field; $c(\mathbf{x}) \in L^2(\Omega)$ is the reaction coefficient; and $f \in L^2(\Omega)$, $g_D \in H^{1/2}(\Gamma_D)$, and $g_N \in H^{-1/2}(\Gamma_N)$ are given scalar-valued functions. Here and hereafter, we use the standard notations and definitions for the Sobolev spaces $H^s(\Omega)^d$ and $H^s(\Gamma)^d$ for s and $\Gamma \subset \partial\Omega$. The standard associated inner products are denoted by $(\cdot,\cdot)_{s,\Omega}$ and $(\cdot,\cdot)_{s,\Gamma}$, and their respective norms are denoted by $\|\cdot\|_{s,\Omega}$ and $\|\cdot\|_{s,\Gamma}$.

We assume that $A(\mathbf{x})$ is uniformly positive definite for almost all $\mathbf{x} \in \Omega$. When employing NNs as approximating functions, a crucial step is to identify or develop an *equivalent* minimization formulation of (4) that lends itself well to discretization.

Least-squares formulations

The convection-diffusion-reaction problem in (4) is not self-adjoint, and hence there is no natural minimization principle. In such case, one can then apply the least-squares (LS) principle to create a manufactured one. The LS principle offers great flexibility: all consistent equations and data can be incorporated into the LS functional. Yet, balancing the various terms is challenging; ill-scaled LS functionals can lead to suboptimal accuracy and inefficient training. The minimal requirement for a viable LS formulation is its equivalence to the original PDE; otherwise, the physical fidelity of the model is compromised.

A classical example is the Bramble–Schatz LS (BSLS) formulation [4] (see also [2]) for elliptic PDEs, which applies the LS principle directly to the *strong* form of the PDE with the boundary conditions in (4) using appropriate Sobolev norms. Specifically, the Bramble-Schatz least-squares functional is defined as

$$\begin{aligned} \mathcal{L}(v) &= \|-\mathrm{div}\,(A\nabla v) + \boldsymbol{\beta}\cdot\nabla v + c\,v - f\|^2_{0,\Omega} \\ &\quad + \|v - g_D\|^2_{\frac{3}{2},\Gamma_D} + \|A\nabla v\cdot\mathbf{n} - g_N\|^2_{\frac{1}{2},\Gamma_N} \end{aligned} \tag{5}$$

for all $v \in H^2(\Omega)$, where the $H^{3/2}(\Gamma_D)$ and $H^{1/2}(\Gamma_N)$ are used for the respective Dirichlet and Neumann boundary conditions. Then the Bramble-Schatz LS formulation [4] is to find $u \in H^2(\Omega)$ such that

$$\mathcal{L}(u) = \min_{v\in H^2(\Omega)} \mathcal{L}(v). \tag{6}$$

When the solution of (4) is sufficiently smooth, e.g., belongs to $H^2(\Omega)$ (the collection of functions whose second-order weak derivatives are square-integrable), the BSLS is equivalent to the underlying problem and yields a well-balanced formulation. However, it fails in the presence of singularities, such as those caused by geometric corners or material interfaces. Such a limitation is overcome by the FOSLS formulation (see (8)– (9)), i.e., applying the LS principle to an equivalent first-order system (see, e.g., Cai-Lazarov-Manteuffel-McCormick [11])).

Introducing the dual (flux) variable (an important physical quantity), $\boldsymbol{\sigma} = -A\nabla u$, problem (4) may be rewritten as the following first-order system:

$$\begin{cases} \boldsymbol{\sigma} + A\nabla u &= \mathbf{0}, \quad \text{in } \Omega, \\ \mathrm{div}\,\boldsymbol{\sigma} + \boldsymbol{\beta}\cdot\nabla u + cu &= f, \quad \text{in } \Omega \end{cases} \tag{7}$$

with boundary conditions

$$u = g_D \text{ on } \Gamma_D \quad \text{and} \quad \boldsymbol{\sigma}\cdot\mathbf{n} = -g_N \text{ on } \Gamma_N.$$

Let $H(\mathrm{div};\Omega)$ be the collection of all square integrable vector fields whose divergence is also square integrable:

$$H(\mathrm{div};\Omega) = \{\boldsymbol{\tau} \in L^2(\Omega)^d \,:\, \mathrm{div}\,\boldsymbol{\tau} \in L^2(\Omega)\}.$$

Then, the first-order system least-squares (FOSLS) formulation is to find $(\boldsymbol{\sigma}, u) \in H(\mathrm{div};\Omega)\times H^1(\Omega)$ such that

$$\mathcal{G}(\boldsymbol{\sigma}, u) = \min_{(\boldsymbol{\tau},v)\in H(\mathrm{div};\Omega)\times H^1(\Omega)} \mathcal{G}(\boldsymbol{\tau}, v), \tag{8}$$

where the FOSLS functional is given by

$$\begin{aligned} \mathcal{G}(\boldsymbol{\tau}, v) &= \|\mathrm{div}\,\boldsymbol{\tau} + \boldsymbol{\beta}\cdot\nabla v + cv - f\|^2_{0,\Omega} + \|v - g_D\|^2_{\frac{1}{2},\Gamma_D} \\ &\quad + \|A^{-\frac{1}{2}}\boldsymbol{\tau} + A^{\frac{1}{2}}\nabla v\|^2_{0,\Omega} + \|\boldsymbol{\tau}\cdot\mathbf{n} + g_N\|^2_{-\frac{1}{2},\Gamma_N}, \end{aligned} \tag{9}$$

where the $H^{1/2}(\Gamma_D)$ and $H^{-1/2}(\Gamma_N)$ are used for the respective Dirichlet and Neumann boundary conditions.

Primal and dual formulations

When the convection velocity field vanishes, i.e., $\boldsymbol{\beta} \equiv \mathbf{0}$, (4) becomes the following diffusion-reaction problem

$$\begin{cases} -\mathrm{div}\,(A\nabla u) + c\,u = f \ \text{ in } \Omega \in R^d, \\ u\big|_{\Gamma_D} = g_D, \quad \text{and} \quad (A\nabla u)\cdot\mathbf{n}\big|_{\Gamma_N} = g_N. \end{cases} \tag{10}$$

We assume that $A(\mathbf{x})$ is uniformly positive definite and that $c(\mathbf{x}) \geq 0$ for almost all $\mathbf{x} \in \Omega$. Under these assumptions, (10) is self-adjoint, coercive, and hence has natural optimization principles.

Define the energy functional by

$$J(v) = \frac{1}{2}\left\{\left\|A^{\frac{1}{2}}\nabla v\right\|_{0,\Omega}^2 + \left\|c^{\frac{1}{2}}v\right\|_{0,\Omega}^2 + \|v - g_D\|_{\frac{1}{2},\Gamma_D}^2\right\} - (f, v)_{0,\Omega} - (g_N, v)_{0,\Gamma_N} \quad (11)$$

for all $v \in H^1(\Omega)$, where the $H^{1/2}(\Gamma_D)$ norm is used for the Dirichlet boundary condition. Then the primal formulation (often called the Ritz form) is to find the primal variable $u \in H^1(\Omega)$ such that

$$J(u) = \min_{v \in H^1(\Omega)} J(v). \quad (12)$$

For applications in continuum mechanics, the dual variable, such as flux in porous media flow or stress in elasticity, often stands as the primary physical quantity of interest. While it can be derived from methods based on the primal variable, such as pressure or displacement, through differentiation, this approach comes at the cost of degrading the order of the approximation for the dual variables. To compute the dual variable directly, one often uses the complementary energy functional defined by

$$J^*(\tau) = \frac{1}{2}\left\{\left\|A^{-\frac{1}{2}}\tau\right\|_{0,\Omega}^2 + \left\|c^{-\frac{1}{2}}(\operatorname{div}\tau - f)\right\|_{0,\Omega}^2\right\} + (g_D, \tau\cdot\mathbf{n})_{0,\Gamma_D} \quad (13)$$

for all $\tau \in \Sigma_N = H(\operatorname{div};\Omega) \cap \{\mathbf{n}\cdot\tau|_{\Gamma_N} = g_N\}$, where we assume that $c > 0$. Then the dual formulation is to find the dual variable $\sigma \in \Sigma_N$ such that

$$J^*(\sigma) = \max_{\tau \in \Sigma_N} J^*(\tau). \quad (14)$$

If $c(\mathbf{x}) = 0$ for all $\mathbf{x} \in \Omega$, the complementary energy functional and the solution set are modified as

$$J^*(\tau) = \frac{1}{2}\left\|A^{-\frac{1}{2}}\tau\right\|_{0,\Omega}^2 + (g_D, \tau\cdot\mathbf{n})_{0,\Gamma_D}$$

and $\Sigma_{N,0} = \{\tau \in \Sigma_N : \operatorname{div}\tau = f\}$, respectively. For more details, see [20].

Physics-Preserving Neural Network (P²NN) Methods

For the diffusion-reaction problem, the deep Ritz [15] and the dual neural network (DuNN) [20] were studied recently by seeking an approximated solution in the set of NNs based on the primal and the dual formulation, respectively. These methods do preserve the underlying physics. For the primal formulation in (12), the Dirichlet boundary condition is essential and is enforced through penalization in either $L^2(\Gamma_D)$ norm [15] or $H^{1/2}(\Gamma_D)$ norm [19]. Notice that the well-known Nitsche method, using a perturbed Robin boundary condition, is equivalent to the $L^2(\Gamma_D)$ norm penalization.

In many applications, the dual variable is often the primary quantity of interest. Based on the dual formulation in (14), the DuNN method was introduced in [20]. The key in the design of the DuNN is the introduction of the discrete divergence operator that preserves the underlying physics. Additional benefit of the DuNN is that the essential (Neumann) boundary condition of the dual formulation can be enforced through the discrete divergence operator.

Recently, several NN-based least-squares approaches have been proposed for elliptic PDEs and nonlinear hyperbolic conservation laws. Those methods can be categorized as PINNs [14], [21] and the least-squares neural network (LSNN) methods [8], [7], [6]. PINNs applies the discrete l^2 norm LS principle to the **strong form** of the PDE, precisely a discrete version of the BSLS but using the l^2 norm penalization for both Dirichlet and Neumann boundary conditions. In practice, they often rely on Monte Carlo sampling for numerical integration and on numerical or automatic differentiation along coordinate directions. Since the strong form of a PDE with non-smooth solution is always wrong/incomplete, the resulting PINNs loss may no longer faithfully represent the intended physics, which can lead to inaccurate or even physically inconsistent approximations.

The LSNN methods are built on a correct "**weak**" form of the PDE that preserves the underlying physical principles by design. Moreover, they employ the continuous L^2 norm, admit ReLU activation function, incorporate physics-preserving numerical differentiation, and use adaptive integration strategies to accurately evaluate the loss over the induced partition of the domain.

For the convection-diffusion-reaction problem in (4), the deep FOSLS approach in [8] employs the conventional divergence operator defined along coordinate-direction differentiation. This choice can be adequate in one dimension, where $H(\operatorname{div};\Omega) = H^1(\Omega)$ when $\Omega \subset R$, but it would fail in multi-dimension, e.g., for elliptic interface problem. In this section, we therefore modify the deep FOSLS framework to obtain an LSNN method by replacing the standard divergence with the discrete divergence operator introduced in [6].

Numerical integration

The evaluation of the least-squares functionals $\mathcal{L}(v)$ and $\mathcal{G}(\tau, v)$ defined respectively in (5) and (9), involves

integrations over the computational domain $\Omega \subset R^d$ ($d = 2$ or 3) and the boundary $\partial\Omega$. In practice, these integrals must be approximated using numerical quadrature. To this end, let

$$\mathcal{T} = \{K : K \text{ is an open subdomain of } \Omega\}$$

be a partition of the domain Ω, referred to as the *integration mesh*. Here, the partition means that the union of all subdomains of $\mathcal{T}$ equals the whole domain Ω and that any two distinct subdomains of $\mathcal{T}$ have no intersection. A composite quadrature over $\mathcal{T}$ is then written as

$$\sum_{K\in\mathcal{T}} \mathcal{Q}_K(w) \approx \sum_{K\in\mathcal{T}} \int_K w(\mathbf{x})\, d\mathbf{x} = \int_\Omega w(\mathbf{x})\, d\mathbf{x},$$

where $\mathcal{Q}_K(w) \approx \int_K w(\mathbf{x})\, d\mathbf{x}$ represents a quadrature rule over subdomain K. The specific quadrature rule $\mathcal{Q}_K$ may vary across different elements $K \in \mathcal{T}$, and can be chosen from standard formulas such as Gaussian quadrature or Newton-Cotes formulas, including the midpoint, trapezoidal, or Simpson's rule. For instance, using the midpoint rule for all $K \in \mathcal{T}$, $\mathcal{Q}_K(w) = w(\mathbf{x}_K)|K|$, where $\mathbf{x}_K$ is the centroid of element K and $|K|$ denotes its d-dimensional measure.

The true solution u of a PDE is unknown and may exhibit localized features such as steep gradients, discontinuities, or singularities. How to choose an integration mesh, that represents the solution well, and in turn an accurate quadrature rule is highly non-trivial. When computational cost is not a concern, one may use a uniform partition $\mathcal{T}$ that is fine enough to approximate the unknown solution well using a piecewise polynomial. A better and most efficient way to achieve this goal is the so-called adaptive quadrature, (see, e.g., [22]), which was used and studied in [19] for the deep Ritz method in linear elasticity and in [6] for the LSNN method for scalar hyperbolic PDEs.

Physics-preserving numerical differentiation

The solution u of (4) is always in $H^{1+\alpha}(\Omega)$ for some $\alpha > 0$, but α is less than one for either the domain Ω having re-entrant corner or the diffusion coefficient having discontinuity (elliptic interface problem). In those cases, $\boldsymbol{\sigma} = -A\nabla u$ is not in $H^1(\Omega)^d$ but only in $H(\mathrm{div};\Omega)$. Therefore, the FOSLS formulation in (8) is equivalent to the original problem, while the BSLS formulation in (6) is not since the solution space $H^2(\Omega)$ is too small.

A $H(\mathrm{div};\Omega)$ vector field is known for continuity of its normal component across any surface in the domain Ω, while its tangential components could have discontinuities across some interfaces. This makes conventional numerical or auto-differentiations along coordinate directions inadequate for approximating the divergence operator. To overcome this obstacle, we introduced the discrete divergence operator in [6], [18] based on a weak definition of the divergence operator:

$$\mathrm{div}\,\boldsymbol{\tau}(\mathbf{x}) = \lim_{\epsilon\to 0} \frac{1}{|B_\epsilon(\mathbf{x})|} \int_{\partial B_\epsilon(\mathbf{x})} \boldsymbol{\tau}\cdot\mathbf{n}\, dS, \qquad (15)$$

where $B_\epsilon(\mathbf{x}) \in R^d$ is a ball of radius ϵ centered at $\mathbf{x}$, $\partial B_\epsilon(\mathbf{x})$ is the boundary of $B_\epsilon(\mathbf{x})$, and $\mathbf{n}$ is the outward unit vector normal to $\partial B_\epsilon(\mathbf{x})$.

Now, we define the physics-preserving discrete divergence operator at each integration point $\mathbf{x}$ with a corresponding control volume $K_\mathbf{x}$ that contains the point as

$$\mathrm{div}_\mathcal{T}\,\boldsymbol{\tau}(\mathbf{x}) = \frac{1}{|K_\mathbf{x}|}\mathcal{Q}_{\partial K_\mathbf{x}}(\boldsymbol{\tau}\cdot\mathbf{n}), \qquad (16)$$

where $\mathcal{Q}_{\partial K_\mathbf{x}}(\cdot)$ denotes a *composite quadrature rule* applied over the boundary $\partial K_\mathbf{x}$ of the control volume $K_\mathbf{x}$ and $\mathbf{n}$ is the unit outward vector normal to $\partial K_\mathbf{x}$.

Under the midpoint quadrature rule $\mathcal{Q}_K$, each subdomain $K \in \mathcal{T}$ has a single integration point $\mathbf{x}_K$, typically chosen as the centroid of K. In this case, the control volume is simply K. More generally, suppose the quadrature rule $\mathcal{Q}_K$ for a subdomain $K \in \mathcal{T}$ has J distinct integration points,

$$\mathbf{x}_{K_j} \in K \subset \mathcal{T}, \quad \text{for } j = 1, \dots, J.$$

Let $\mathcal{T}_K = \{K_j\}_{j=1}^J$ be a partition of K such that $\mathbf{x}_{K_j} \in K_j$, where K_j is referred to as the control volume of the integration point $\mathbf{x}_{K_j}$. Let $\mathcal{Q}_{\partial K_j}(\cdot)$ be a composite quadrature rule over the boundary ∂K_j, then the discrete divergence operator $\mathrm{div}_\mathcal{T}$ at the integration point $\mathbf{x}_{K_j}$ can be defined analogously via (16).

A least-squares neural network (LSNN) method

For $S = D$ or N, denote by

$$\mathcal{E}_S = \{E = \partial K \cap \Gamma_S : K \in \mathcal{T}\} \qquad (17)$$

the collections of the Dirichlet or Neumann boundary faces of the integration mesh $\mathcal{T}$. For each face E in $\mathcal{E}_S$, let $\mathcal{Q}_E(w)$ denote a quadrature rule applied to an integrand w defined on E.

First, we discuss how to weakly enforce the Neumann boundary condition through the discrete divergence operator defined in (16). For simplicity, assume that the quadrature rule $\mathcal{Q}_K(\cdot)$ is the midpoint rule. In this case, $\mathbf{x}_K$ is the centroid of K and the sole integration point within each subdomain K. For each

$K \in \mathcal{T}$, if $\partial K \cap \Gamma_N \neq \varnothing$, then we modify the discrete divergence operator at the integration point $\mathbf{x}_K$ by

$$\mathrm{div}_{\mathcal{T}}\,\boldsymbol{\tau}(\mathbf{x}_K) = \frac{\mathcal{Q}_{\partial K\setminus(\partial K\cap\Gamma_N)}(\boldsymbol{\tau}\cdot\mathbf{n}) - \mathcal{Q}_{\partial K\cap\Gamma_N}(g_N)}{|K|}. \quad (18)$$

Now, we are ready to define a discrete FOSLS functional using the $L^2(\Gamma_D)$ penalization as

$$\mathcal{G}_{\mathcal{T}}(\boldsymbol{\tau},v) = \sum_{K\in\mathcal{T}} \mathcal{Q}_K\Big((\mathrm{div}_{\mathcal{T}}\boldsymbol{\tau}+\boldsymbol{\beta}\cdot\nabla v + cv - f)^2 + |\boldsymbol{\tau}+A\nabla v|^2\Big) + \gamma_D \sum_{E\in\mathcal{E}_D} \mathcal{Q}_E\left((v-g_D)^2\right),$$

where γ_D is a penalization constant. Another option is to use the discrete $H^{\frac{1}{2}}(\Gamma_D)$ norm in [19] to obtain a less sensitive γ_D. The LSNN (modified deep FOSLS) method is to find $(\boldsymbol{\sigma}_{N,\mathcal{T}}, u_{N,\mathcal{T}}) \in \mathcal{M}(l)^{d+1}$ such that

$$\mathcal{G}_{\mathcal{T}}(\boldsymbol{\sigma}_{N,\mathcal{T}}, u_{N,\mathcal{T}}) = \min_{(\boldsymbol{\tau},v)\in\mathcal{M}(l)^{d+1}} \mathcal{G}_{\mathcal{T}}(\boldsymbol{\tau},v). \quad (19)$$

Numerical Experiments

Consider a two-dimensional Kellogg interface problem posed on the unit disk $\Omega = \{(r,\theta)|\, r\in[0,1), \theta\in[0,2\pi)\}$ and satisfying the Helmholtz equation in (10) with $c = -I$, $\Gamma_N = \varnothing$, $\Gamma_D = \partial\Omega$, and $A = \alpha(\theta)I$. The diffusion coefficient $\alpha(\theta)$ is discontinuous, equaling to 1 in the second and fourth quadrants and jumps to 161.4476387975881 in the first and third quadrants. The right-hand side is prescribed by $f(r,\theta) = -r^{\beta}\mu(\theta)$, where

$$\mu(\theta) = \begin{cases} c_1\cos((\theta-\pi/2+\rho)\beta), & 0\le\theta\le\pi/2,\\ \cos(\rho\beta)\cos((\theta-\pi+\sigma)\beta), & \pi/2\le\theta\le\pi,\\ \cos(\sigma\beta)\cos((\theta-\pi-\rho)\beta), & \pi\le\theta\le 3\pi/2,\\ c_2\cos((\theta-3\pi/2-\sigma)\beta), & 3\pi/2\le\theta\le 2\pi. \end{cases}$$

Here $\beta = 0.1$, $\rho = \pi/4$, $\sigma = -14.92256510455152$, $c_1 = \cos((\pi/2-\sigma)\beta)$, and $c_2 = \cos((\pi/2-\rho)\beta)$. The exact solution is given by $u = -f$.

The Kellogg problem is a challenging benchmark test for adaptive mesh refinement methods,(see e.g. [13]), and the Poisson equation was previously tested using the Ritz formulation in [17]. In the present example, the reaction coefficient c in this test is negative, so the problem is self-adjoint but not positive definite and therefore does not admit a natural minimization principle. The problem is further complicated by intersecting discontinuous interfaces and by the point singularity of ∇u at the origin. Consequently, $u \in H^{1+\beta}(\Omega)$ but not in $H^2(\Omega)$. As a result, the BSLS (5) and its discrete version, the vanilla PINN [21] formulation are not theoretically well suited for this problem.

To test the LSNN method under P²NN framework, we introduce the flux variable $\boldsymbol{\sigma}_r$ in polar coordinates,

$$\boldsymbol{\sigma}_r = -\alpha\nabla_r u = -\alpha\begin{pmatrix} u_r \\ r^{-1}u_\theta \end{pmatrix} = \begin{pmatrix} \sigma_{r1} \\ \sigma_{r2} \end{pmatrix},$$

and the corresponding FOSLS loss functional is,

$$\mathcal{G}(\boldsymbol{\tau}_r, v) = \|\mathrm{div}\boldsymbol{\tau}_r - v - f\|^2_{0,\Omega} + \|\boldsymbol{\tau}_r + \alpha\nabla_r v\|^2_{0,\Omega} + \gamma_D\|v - g_D\|^2_{0,\Gamma_D}.$$

The numerical accuracy of the proposed P²NN method is influenced by several factors, including the NN approximation error, the numerical integration and differentiation errors, and the optimization error. To investigate these effects, we conducted sensitivity studies with respect to the network architecture, the integration mesh size, and the quadrature rule and subinterval size used in the discrete divergence operator.

Specifically, we employed three two-hidden-layer ReLU NNs of the form d_{in}-n_1-n_2-d_{out} as the approximation set in $\mathcal{M}(2)^3$. Here $d_{in} = 2$ denotes the input dimension. The two inputs are the spatial variable (r,θ). $d_{out} = 3$ is the NN output dimension. The three outputs are the numerical approximations of $u(r,\theta)$, $\sigma_{r1}(r,\theta)$, and $\sigma_{r2}(r,\theta)$. We considered $n_1 = n_1 = 30$, 50, and 80 neurons in each hidden layer, corresponding to 1083, 2803, and 6883 trainable parameters, respectively. The penalty parameter γ_D for the Dirichlet boundary term was chosen empirically to be of order $O(1/h)$, where h denotes the integration mesh size.

Table 1 reports the relative L^2 error, $\frac{\|u-u_{N,\tau}\|_{0,\Omega}}{\|u\|_{0,\Omega}}$, produced by the LSNN method. The results were obtained on an integration mesh $\mathcal{T}$ of size 100×360, aligned with the polar coordinate axes, together with a midpoint quadrature using one interval per segment for the discrete divergence operator estimation. As expected, increasing the network size improves the approximation capability of the NN and leads to smaller numerical errors. In addition, the method appears fairly robust with respect to the choice of γ_D. As shown in Table 1, comparable relative L^2 errors are observed for γ_D ranging from 100 to 1000.

TABLE 1. Relative L^2 errors: sensitivity with respect to γ_D.

Network	γ_D				
	20	100	200	500	1000
2-30-30-3	0.457	0.115	0.066	0.068	0.099
2-50-50-3	0.343	0.060	0.057	0.064	0.066
2-80-80-3	0.291	0.057	0.053	0.056	0.077

Next, we examine the effects of numerical integration and divergence operator estimation using the network structure 2-80-80-3. Two integration mesh/control-volume sizes were tested, together with

two quadrature rules, namely the trapezoidal and midpoint rules, were tested using different number of sub-intervals in the boundary integral of the discrete divergence operator (16). Table 2 shows that the numerical accuracy improves as the integration mesh is refined and as the number of sub-intervals used in the boundary integral increases. This trend is consistent for both quadrature rules, highlighting the importance of accurate numerical integration in the proposed discrete divergence operator. On the coarser mesh, the two quadrature rules yield comparable results, whereas on the finer mesh the midpoint rule gives slightly smaller errors. Overall, these results indicate that the proposed method is stable with respect to the choice of quadrature rule, while benefiting from finer integration meshes and more accurate boundary integration in the discrete divergence operator calculation.

We also compared the proposed numerical integration scheme with Monte Carlo integration based on random sampling. Specifically, 9,000 random points, comparable to the 50×180 integration mesh, were sampled. Around each sample point, a control volume was constructed as a small disk of radius $\sqrt{1/9000}$, and the boundary integral in the discrete divergence operator was evaluated using 4 sub-intervals along the circular boundary, roughly comparable to using 1 sub-interval per segment in the structured integration mesh. The results (see Table 2.) show that Monte Carlo integration, when combined with the proposed divergence operator, still produces physically meaningful solutions, with a relative L^2 error of 0.120. Nevertheless, this is substantially less accurate than the 0.064 error achieved by the specifically designed numerical integration scheme.

TABLE 2. Effect of quadrature rule and integration-point resolution in the discrete divergence operator, combined with different integration mesh/control-volume sizes.

Integration Mesh $\mathcal{T}$	Quadrature rule $\mathcal{Q}_{\partial K_j}(\cdot)$	# of sub-intervals in ∂K_j		
		1	2	4
50×180	Trapezoidal	0.071	0.065	0.053
	Midpoint	(0.064)	0.061	0.059
100×360	Trapezoidal	0.059	0.060	0.055
	Midpoint	0.053	0.049	0.044
Random sample 9000 points	Midpoint	(0.120)		

For $\nabla_r v$, the above tests were carried out using a forward finite-difference approximation, with the step size $d_r = h_r/2$, and $d_\theta = h_\theta/2$, where h_r and h_θ denote the integration mesh size along radial and angular directions, respectively. We further performed a sensitivity study on the numerical differentiation (ND) with varying the step sizes, and compared the results with those obtained using automatic differentiation (AD). Table. 3 shows that ND with different step sizes yields very similar results, indicating that the method is relatively insensitive to the choice of differentiation step size within this range. In contrast, the AD-based approach performs significantly worse and yields a non-physical solution (see. Figure 1(b) for the graphical illustration of the result) . This degradation is likely due to the presence of interfaces and the low regularity of the solution.

TABLE 3. Sensitivity analysis with respect to the step size in numerical differentiation (ND), and comparison with automatic differentiation (AD).

Net: 2-80-80-3 $\mathcal{T}$: 50×180 $\gamma_D = 200$	ND $d_r = h_r/m$, $d_\theta = h_\theta/m$			AD
	m=2	m=4	m=8	
	0.064	0.060	0.064	0.536

We also examined the performance of a vanilla PINN. Since PINNs rely on automatic differentiation and the Helmholtz equation contains a second-order Laplacian term, the ReLU activation function is not suitable because its second derivative vanishes almost everywhere. We therefore replaced ReLU with smoother activation functions, namely sigmoid, hyperbolic tangent, and Swish$_1$, and tested all three using a 2-80-80-1 network with 9,000 randomly sampled collocation points. As expected, the strong-form PINN formulation does not perform well for this low-regularity problem. The resulting PINN solution is non-physical, as illustrated in Figure 1(c). For all three activation functions, the relative L^2 error is around 0.3.

The Adam optimizer was used to solve the discrete minimization problem in all the tests reported above. The optimization was initialized with a learning rate of 10^{-3}, which was reduced by 50% every 20,000 iterations. To ensure convergence to a stable accuracy level and to make comparisons across different parameter choices consistent, we used a total of 150,000 iterations and the same learning-rate decay strategy in all tests. Figure 1(c) shows the training loss curves for the LSNN method combined with ND and AD, together with the loss curve of the vanilla PINN. As shown in the figure, all three loss curves decrease rapidly in the early stage and then continue to decrease more gradually. However, the vanilla PINN loss stagnates at a relatively large residual level and leads to a nonphysical solution. For the LSNN method using AD, the corresponding loss curve decreases faster and reaches a smaller residual. Nevertheless, a nonphysical solution is still obtained, even though the loss is minimized to a relatively small value. In

contrast, only the LSNN method combined with the physics-preserving numerical differentiation operator converges to a physically correct approximation (See Figure 1(a), 1(d) and 1(e) for the three outputs.)

The loss decay behavior also reflects the difficulty of using first-order gradient-based optimization to solve nonlinear and nonconvex NN discretizations for low-regularity interface problems. Consequently, a relatively large number of iterations is typically required. This observation is closely related to Question (iii) raised in the Introduction and further motivates the development of more efficient nonlinear solvers for P^{2}NN discretizations, which is an important direction of our ongoing work.

For the ReLU NNs, the neurons in the first hidden layer were initialized to partition the domain uniformly in the angular direction, while the neurons in the second hidden layer were initialized randomly. A ReLU NN represents a continuous piecewise linear function over a physical partition induced by its hidden-layer neurons, and this partition can adapt to the PDE solution during training. As shown in Figure 2, the training process deforms the initial partition in Figure 2(a) into a more suitable learned partition in Figure 2(b). In this example, the learned partition aligns with the intersecting interfaces and automatically becomes denser near the singular point at the origin. This behavior illustrates the moving-mesh capability of NN-based methods and highlights their potential advantage over fixed-mesh methods, particularly when interface locations are unknown.

Conclusion and Discussion

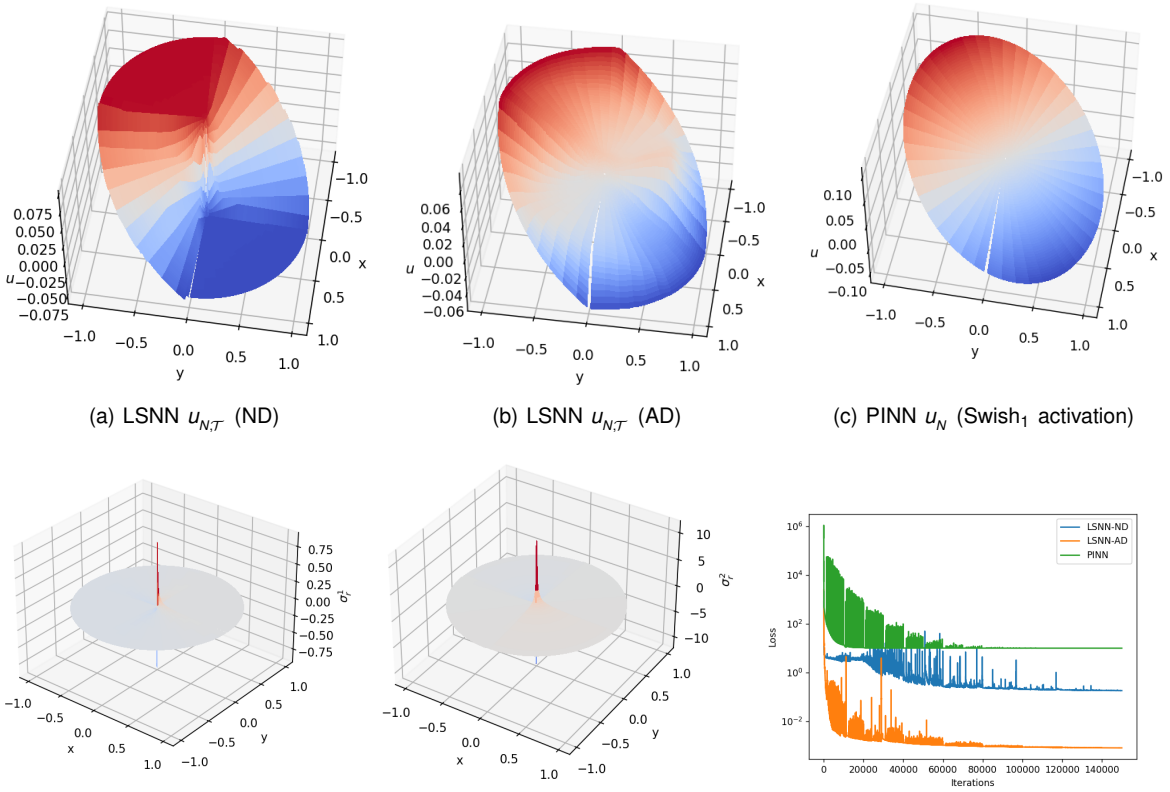
As illustrated in this paper and in [20], [7], [6], [18], designing a physics-preserving neural network (P^{2}NN) method begins with an equivalent optimization formulation of the underlying PDE. A second essential step is to discretize key differential operators, for example, the divergence operator in elliptic PDEs and hyperbolic conservation laws, using physical-preserving numerical differentiation. Numerical integration for NN-based methods is also non-trivial, because the final physical partition is irregular and unknown, accurate quadrature becomes critical, as it directly controls the quality of the "training data".

The numerical results in this paper provide evidence that the proposed P^{2}NN framework can improve upon vanilla PINN-type discretizations for elliptic interface problems by using equivalent formulations, physics-preserving differential operators, and carefully designed numerical integration. These results should not be interpreted as claiming that P^{2}NN methods resolve all difficulties associated with NN-based PDE solvers. Rather, they show that preserving the mathematical and physical structure of the underlying PDE is a necessary step toward reliable NN-based discretizations. In particular, natural optimization principles, when they exist, are generally preferable to manufactured formulations based solely on the LS principle.

Although the degrees of freedom (DoFs) of an optimal NN approximation can be substantially lower than those of finite element methods (FEMs) for problems with discontinuities, thin interior/boundary layers, and interface singularities, the computational cost of determining the nonlinear parameters in NNs is often much higher than solving the corresponding linear systems in FEMs. Two key factors contribute to this gap: (i) NN parameters enter the approximation nonlinearly, and (ii) there remains a lack of optimization algorithms specifically designed to efficiently adapt the implicit "mesh" encoded by the network. Recent progress has been made along these lines; see, for example, [10] and references therein. Nevertheless, existing results are largely limited to shallow networks and/or one-dimensional settings.

REFERENCES

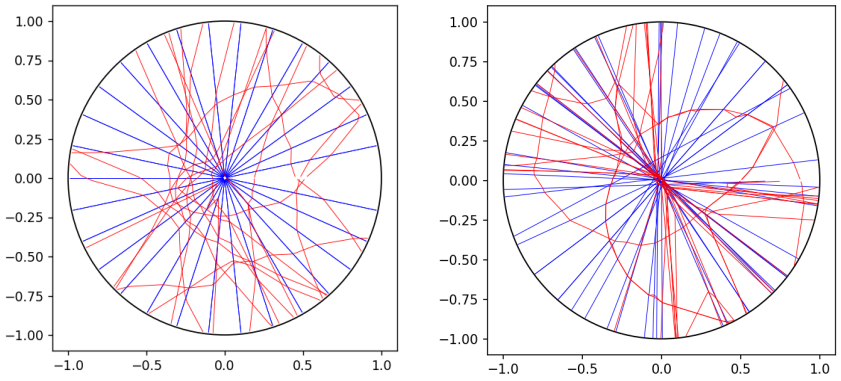
1. J. Berg and K. Nystrom. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.
2. P. B. Bochev and M. D. Gunzburger. *Least-Squares Finite Element Methods*, volume 166. Applied Mathematics Sciences, Springer, 2009.
3. J. Bramble, R. Lazarov, and J. Pasciak. A least-squares approach based on a discrete minus one inner product for first order system. *Math. Comp*, 66:935–955, 1997.
4. J. H. Bramble and A. H. Schatz. Rayleigh-Ritz-Galerkin-methods for Dirichlet's problem using subspaces without boundary conditions. *Comm. Pure Appl. Math*, 23:653–675, 1970.
5. F. Brezzi and M. Fortin. *Mixed and Hybrid Finite Element Methods*. Springer-Verlag, New York, 1991.
6. Z. Cai, J. Chen, and M. Liu. Least-squares ReLU neural network (LSNN) method for scalar nonlinear hyperbolic conservation laws: discrete divergence operator. *J. Comput. Appl. Math.*, 433 (2023) 115298, arXiv:2110.10895v3 [math.NA], 2023.
7. Z. Cai, J. Chen, and M. Liu. Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation. *J. Comput. Phys.*, 443 (2021) 110514.
8. Z. Cai, J. Chen, M. Liu, and X. Liu. Deep least-



(a) LSNN $u_{N;\mathcal{T}}$ (ND) (b) LSNN $u_{N;\mathcal{T}}$ (AD) (c) PINN u_N (Swish$_1$ activation)

(d) LSNN $\sigma_{r_1 N;\mathcal{T}}$ (ND) (e) LSNN $\sigma_{r_2 N;\mathcal{T}}$ (ND) (f) Loss decay curves for the three methods

FIGURE 1. Kellogg interface problem: numerical results of LSNN with a 2-80-80-3 network and the vanilla PINN with a 2-80-80-1 network.



(a) Initial physical partition (b) Trained physical partition

FIGURE 2. Kellogg interface problem: trained physical partition for a 2-30-30-3 network using LSNN. Blue lines denote the 30 first-hidden-layer breaking lines, and red curves denote the 30 second-hidden-layer breaking polylines.

squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. *J. Comput. Phys.*, 420 (2020) 109707.
9. Z. Cai, J. Choi, and M. Liu. ReLU neural network approximation to piecewise constant functions. *arXiv:2410.16506 [math.FA]*, 2024.
10. Z. Cai, A. Doktorova, R. D. Falgout, and C. Herrera. Efficient shallow Ritz method for 1D diffusion-reaction problems. *SIAM J. Sci. Comput.*, page S414–S435, 2025.
11. Z. Cai, R. Lazarov, T. A. Manteuffel, and S. F. McCormick. First-order system least squares for second-order partial differential equations: part I. *SIAM J. Numer. Anal*, 31:1785–1799, 1994.
12. Z. Cai, T. A. Manteuffel, and S. F. McCormick. First-order system least squares for second-order partial differential equations: Part II. *SIAM J. Numer. Anal*, 34(2):425–454, 1997.
13. Z. Cai and S. Zhang. Recovery-based error estimator for interface problems: Conforming linear elements. *SIAM J. Numer. Anal.*, 47(3):2132–2156, 2009.
14. M. 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.
15. W. E and B. Yu. The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1), 3 2018.
16. I. Ekeland and R. Témam. *Convex Analysis and Variational Problems*. North-Holland and American Elsevier, Amsterdam and New York, 1976.
17. M. Liu and Z. Cai. Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs. *Comput. Math. Appl.*, 113:103–116, 2022.
18. M. Liu and Z. Cai. Least-squares neural network (LSNN) method for scalar hyperbolic partial differential equations. *arXiv:2601.20013 [math.NA]*, 2026.
19. M. Liu, Z. Cai, and K. Ramani. Deep Ritz method with adaptive quadrature for linear elasticity. *Comput. Methods Appl. Mech. Engrg.*, 415 (2023) 116229.
20. M. Liu, Z. Cai, and K. Ramani. Dual neural network (DuNN) method for elliptic partial differential equations and systems. *J. Comput. Appl. Math.*, 467:116596, 2025.
21. 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. *J. Comput. Phys.*, 378:686–707, 2019.
22. J. R. Rice. A metalgorithm for adaptive quadrature. *J. ACM*, 22(1):61–82, 1975.
23. L. Schumaker. *Spline Functions: Basic Theory*. Wiley, New York, 1981.
24. J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1139–1364, 2018.
25. J. Xu. The finite neuron method and convergence analysis. *arXiv:2010.01458vl*, 2020.