Journal of Computational and Applied Mathematics 467 (2025) 116596

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Dual Neural Network (DuNN) method for elliptic partial differential equations and systems

Min Liu [a,*], Zhiqiang Cai [b], Karthik Ramani [a]

[a] School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette IN 47907-2088, United States of America

[b] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067, United States of America

ARTICLE INFO

Keywords:
Elliptic PDE
Linear elasticity
Deep neural network
Dual formulation

ABSTRACT

This paper presents the Dual Neural Network (DuNN) method, a physics-driven numerical method designed to solve elliptic partial differential equations and systems using deep neural network functions and a dual formulation. The underlying elliptic problem is formulated as an optimization of the complementary energy functional in terms of the dual variable, where the Dirichlet boundary condition is weakly enforced in the formulation. To accurately evaluate the complementary energy functional, we employ a novel discrete divergence operator. This discrete operator preserves the underlying physics and naturally enforces the Neumann boundary condition without penalization. For problems without reaction term, we propose an outer-inner iterative procedure that gradually enforces the equilibrium equation through a pseudo-time approach.

1. Introduction

Neural networks (NNs) have demonstrated remarkable performance in computer vision, natural language processing, and various other artificial intelligence tasks. Recently, their application to solving partial differential equations (PDEs) has gained significant traction [1–8]. As a new class of approximating functions, NNs exhibit exceptional approximation capabilities, surpassing those of continuous and discontinuous piecewise polynomials on *fixed* meshes (see, e.g., [9–11]). In particular, a NN function can automatically adapt to a target function through a "moving mesh" behavior, making it one of the most promising candidates among all known functional classes for addressing various challenging problems in scientific computing.

Since NN functions are nonlinear with respect to their parameters, the discretization of a PDE using NN can be formulated as an optimization problem through either natural minimization or manufactured least-squares (LS) principles. Consequently, existing NN-based numerical methods for solving PDEs fall into two main categories: (1) energy-based methods [1,8,12,13], which utilize the principle of natural energy minimization, and (2) deep LS methods employing various types of manufactured least squares [2,3,5,7,14]. Most elliptic problems adhere to the basic minimization principle in the form of an energy functional. Therefore, when using NN as approximating functions, it is natural to discretize the underlying problem based on the energy formulation.

For applications in continuum mechanics, the dual variable, such as stress in elasticity or flux in porous media flow, often stands as the primary physical quantity of interest. While it can be derived from methods based on the primal variable, such as displacement or pressure, through differentiation, this approach comes at the cost of degrading the order of the approximation for the dual variables. In this paper, we propose dual neural network (DuNN), a numerical method that solves elliptic partial differential equations and systems using NNs as the approximating functions for the dual variable, and the complementary energy functional as the loss function. Compared to existing physics-driven NN-based approaches, DuNN offers the following advantages:

* Corresponding author.
E-mail addresses: liu66@purdue.edu (M. Liu), caiz@purdue.edu (Z. Cai), ramani@purdue.edu (K. Ramani).

https://doi.org/10.1016/j.cam.2025.116596
Received 5 July 2024; Received in revised form 12 February 2025
Available online 5 March 2025

(1) In many continuum mechanics problems, accurately computing stress/flux is often more important than displacement/pressure. DuNN achieves this directly without differentiation, as stress/flux is the sole independent variable in the complementary energy functional.
(2) DuNN is applicable to a wider range of problems, including those with or without discontinuities or singularities. Additionally, DuNN is suitable for incompressible materials, which are not adequately addressed by standard energy-based methods.
(3) DuNN enforces both Dirichlet and Neumann boundary conditions naturally, eliminating the need for any penalty term in the loss functional. This results in fewer hyperparameters to adjust.

The remainder of the paper is structured as follows. Section 2 reformulates an elliptic PDE into a minimization problem using a dual formulation. Section 3 presents the DuNN method in detail, and we show our numerical studies in Section 4 and conclude the paper in Section 5.

2. Dual formulation of elliptic partial differential equations

Let Ω be a bounded, open, connected subset of $\mathbb{R}^d$ ($d = 2$ or 3) with a Lipschitz continuous boundary $\partial\Omega$. Let $\boldsymbol{n} = (n_1, \dots, n_d)$ be the outward unit vector normal to the boundary. Partition the boundary $\partial\Omega$ of the domain Ω into two open subsets Γ_D and Γ_N such that $\partial\Omega = \Gamma_D \cup \Gamma_N$ and $\Gamma_D \cap \Gamma_N = \emptyset$. For simplicity, we assume that Γ_D is not empty (i.e., mes$(\Gamma_D) \neq 0$). Otherwise, solutions of partial differential equations considered in this paper are unique up to an additive constant or rigid motions.

We will use the standard notation and definitions for the Sobolev space $\boldsymbol{H}^s(\Omega)^d$ and $\boldsymbol{H}^s(\Gamma)$ for a subset Γ of the boundary of the domain $\Omega \in \mathbb{R}^d$. The standard associated inner product and norms are denoted by $(\cdot,\cdot)_{s,\Omega,d}$ and $(\cdot,\cdot)_{s,\Gamma,d}$ and by $\|\cdot\|_{s,\Omega,d}$ and $\|\cdot\|_{s,\Gamma,d}$, respectively. When there is no ambiguity, the subscript Ω and d in the designation of norms will be suppressed. When $s = 0$, $\boldsymbol{H}^0(\Omega)^d$ coincides with $\boldsymbol{L}^2(\Omega)^d$. In this case, the inner product and norm will be denoted by $(\cdot,\cdot)$ and $\|\cdot\|$, respectively.

2.1. Second-order elliptic problems

Consider the following self-adjoint second-order scalar elliptic partial differential equation:

$$\left\{ \begin{array}{rcll} -\text{div}\,(A\nabla\, u) + c\, u & = & f, & \text{in } \Omega, \\ u & = & g_D, & \text{on } \Gamma_D, \\ \mathbf{n}\cdot A\nabla\, u & = & g_N, & \text{on } \Gamma_N, \end{array} \right. \tag{1}$$

where div is the divergence operator; $f \in L^2(\Omega)$, $c \in L^\infty(\Omega)$, $g_D \in H^{1/2}(\Gamma_D)$, $g_N \in H^{-1/2}(\Gamma_N)$; $A(\boldsymbol{x})$ is a $d\times d$ symmetric matrix-valued function in $L^2(\Omega)^{d\times d}$; and $\mathbf{n}$ is the outward unit vector normal to the boundary. We assume that A is uniformly positive definite and that $c(\boldsymbol{x}) \geq 0$ for almost all $\boldsymbol{x} \in \Omega$.

Introducing the dual (flux) variable $\boldsymbol{\sigma} = -A\nabla u$, then the dual problem is to maximize the complementary energy functional (see, e.g., [15,16]). Specifically, denote the collection of square-integrable vector fields whose divergence are also square-integrable by

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

which is a Hilbert space equipped with norm

$$\|\boldsymbol{\tau}\|_{\text{div},\Omega} = \left(\|\boldsymbol{\tau}\|^2_{0,\Omega} + \|\text{div}\,\boldsymbol{\tau}\|^2_{0,\Omega}\right)^{1/2}.$$

Denote the subset of $H(\text{div};\Omega)$ satisfying the Neumann boundary condition by

$$H_{g,N}(\text{div};\Omega) = H(\text{div};\Omega) \cap \{\mathbf{n}\cdot\boldsymbol{\sigma}|_{\Gamma_N} = g_N\}$$

and the negative complementary functional by

$$J_1(\boldsymbol{\tau};\gamma) = \frac{1}{2}\left\{\left\|A^{-1/2}\boldsymbol{\tau}\right\|^2_{0,\Omega} + \left\|\gamma^{1/2}\,(\text{div}\,\boldsymbol{\tau} - f)\right\|^2_{0,\Omega}\right\} + (g_D, \boldsymbol{\tau}\cdot\mathbf{n})_{0,\Gamma_D}, \tag{2}$$

where γ is given by

$$\gamma = \left\{ \begin{array}{ll} c^{-1}(\mathbf{x}), & \text{if } c > 0, \\ 0, & \text{if } c = 0. \end{array} \right. \tag{3}$$

Then the dual problem is to find $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_g$ such that

$$J_1(\boldsymbol{\sigma};\gamma) = \min_{\boldsymbol{\tau}\in\boldsymbol{\Sigma}_g} J_1(\boldsymbol{\tau};\gamma), \tag{4}$$

where $\boldsymbol{\Sigma}_g$ is given by

$$\boldsymbol{\Sigma}_g = \left\{ \begin{array}{ll} H_{g,N}(\text{div};\Omega), & \text{if } c > 0, \\ \{\boldsymbol{\tau} \in H_{g,N}(\text{div};\Omega) \,:\, \text{div}\,\boldsymbol{\tau} = f\}, & \text{if } c = 0. \end{array} \right. \tag{5}$$

The following proposition is well-known (see, e.g., [17]).

Proposition 1. *Problem (4) has a unique solution $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_g$. Moreover, the solution $\boldsymbol{\sigma}$ satisfies the following a priori estimate:*

$$\|\boldsymbol{\sigma}\|_{div,\Omega} \leq C\left(\|f\|_{0,\Omega} + \|g_D\|_{1/2,\Gamma_D} + \|g_N\|_{-1/2,\Gamma_N}\right).$$

2.2. Linear elasticity and Stokes equations

In linear elasticity problems, it is often more useful to compute accurately stress rather than displacement. This can be achieved by using the dual formulation that maximizes the complementary energy functional for the stress (dual) variable $\boldsymbol{\sigma}$. This section describes the dual formulation for both linear elasticity and Stokes equations.

To this end, denote by $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ the displacement/velocity field and the stress tensor, respectively. Then the stress–displacement/velocity formulation (see, e.g., [17–19]) has the form

$$\left\{\begin{array}{rcll} -\mathbf{div}\,\boldsymbol{\sigma} + c\,\boldsymbol{u} &=& \boldsymbol{f}, & \text{in } \Omega, \\ \mathcal{A}_\lambda\,\boldsymbol{\sigma} - \epsilon(\boldsymbol{u}) &=& \mathbf{0}, & \text{in } \Omega \end{array}\right. \tag{6}$$

with boundary conditions

$$\boldsymbol{u}\Big|_{\Gamma_D} = \boldsymbol{g}_D \quad \text{and} \quad (\boldsymbol{\sigma}\boldsymbol{n})\Big|_{\Gamma_N} = \boldsymbol{g}_N,$$

where $\mathbf{div}$ is the divergence operator; $c \in L^\infty(\Omega)$ is a given scalar-valued function; $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega)^d$, $\boldsymbol{g}_D \in \boldsymbol{H}^{1/2}(\Gamma_D)^d$, and $\boldsymbol{g}_N \in \boldsymbol{H}^{-1/2}(\Gamma_N)^d$ are given vector-valued functions defined on Ω, Γ_D, and Γ_N, representing body force, boundary displacement/velocity, and boundary traction force, respectively; $\epsilon(\boldsymbol{u}) = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\right)$ is the strain tensor; and $\mathcal{A}_\lambda$ is the compliance tensor of fourth order

$$\mathcal{A}_\lambda\,\boldsymbol{\tau} = \frac{1}{2\mu}\left(\boldsymbol{\tau} - \frac{\lambda}{2\mu + d\lambda}(\mathrm{tr}\boldsymbol{\tau})\,\boldsymbol{\delta}_{d\times d}\right) \quad \text{with} \quad \mathrm{tr}\boldsymbol{\tau} = \sum_{i=1}^{d} \tau_{ii}.$$

Here, $\boldsymbol{\delta}_{d\times d}$ is the d-dimensional identity tensor; μ and λ are the material Lamé constants. The material is said to be nearly incompressible if $\lambda \gg 1$ or incompressible if $\lambda = \infty$. It is easy to see that

$$\mathcal{A}_\infty\,\boldsymbol{\tau} = \frac{1}{2\mu}\left(\boldsymbol{\tau} - \frac{1}{d}(\mathrm{tr}\boldsymbol{\tau})\,\boldsymbol{\delta}_{d\times d}\right).$$

Hence the formulation in (6) is valid for both compressible and incompressible materials.

Denote the collection of all symmetric stress whose divergence is square integrable by

$$\boldsymbol{H}^s(\mathrm{div};\Omega) = \left\{\boldsymbol{\tau} \in \boldsymbol{L}^2(\Omega)^{d\times d} \,:\, \boldsymbol{\tau}^t = \boldsymbol{\tau},\ \mathbf{div}\,\boldsymbol{\tau} \in \boldsymbol{L}^2(\Omega)^d\right\}$$

and its subset satisfying the Neumann boundary condition by

$$\boldsymbol{H}^s_{g,N}(\mathrm{div};\Omega) = \left\{\boldsymbol{\tau} \in \boldsymbol{H}^s(\mathrm{div};\Omega) \,:\, \boldsymbol{\tau}\boldsymbol{n}\Big|_{\Gamma_N} = \boldsymbol{g}_N\right\}.$$

The negative complementary energy functional is given by

$$J_2(\boldsymbol{\tau};\gamma) = \frac{1}{2}\left\{\left(\mathcal{A}_\lambda\,\boldsymbol{\tau},\boldsymbol{\tau}\right)_{0,\Omega} + \|\gamma\,(\mathbf{div}\,\boldsymbol{\tau} - \boldsymbol{f})\|^2_{0,\Omega}\right\} - \int_{\Gamma_D} \boldsymbol{g}_D \cdot (\boldsymbol{\tau}\mathbf{n})\,ds, \tag{7}$$

where the γ is given in (3) and

$$\left(\mathcal{A}_\lambda\,\boldsymbol{\tau},\boldsymbol{\tau}\right)_{0,\Omega} = \frac{1}{2\mu}\int_\Omega \left|\mathcal{A}_\infty\boldsymbol{\tau}\right|^2\,d\mathbf{x} + \frac{1}{d(2\mu + d\lambda)}\int_\Omega |\mathrm{tr}\boldsymbol{\tau}|^2\,d\mathbf{x}.$$

Then the dual formulation of problem (6) is to seek $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_g$ such that

$$J_2(\boldsymbol{\sigma};\gamma) = \min_{\boldsymbol{\tau}\in\boldsymbol{\Sigma}_g} J_2(\boldsymbol{\tau};\gamma), \tag{8}$$

where $\boldsymbol{\Sigma}_g$ is given by

$$\boldsymbol{\Sigma}_g = \left\{\begin{array}{ll} \boldsymbol{H}^s_{g,N}(\mathrm{div};\Omega), & \text{if } c > 0, \\ \{\boldsymbol{\tau} \in \boldsymbol{H}^s_{g,N}(\mathrm{div};\Omega) \,:\, \mathbf{div}\,\boldsymbol{\tau} + \boldsymbol{f} = \mathbf{0}\}, & \text{if } c = 0. \end{array}\right. \tag{9}$$

The following existence, uniqueness, and stability are also well-known [17].

Proposition 2. *Problem (8) has a unique solution $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_g$. Moreover, there exists a positive constant such that*

$$\|\boldsymbol{\sigma}\|_{div,\Omega} \le C\left(\|\boldsymbol{f}\|_{0,\Omega} + \|\boldsymbol{g}_D\|_{1/2,\Gamma_D} + \|\boldsymbol{g}_N\|_{-1/2,\Gamma_N}\right).$$

2.3. Abstract setting

For convenience, this section uses an abstract setting to unify the dual formulations in (4) and (8). To this end, for any $\boldsymbol{\sigma},\ \boldsymbol{\tau} \in \boldsymbol{\Sigma}_g$, introduce the following bilinear and linear forms

$$a(\boldsymbol{\sigma},\boldsymbol{\tau};\gamma) = \left\{\begin{array}{ll} (A^{-1}\boldsymbol{\sigma},\boldsymbol{\tau})_{0,\Omega} + (\gamma\,\mathrm{div}\,\boldsymbol{\sigma},\mathrm{div}\,\boldsymbol{\tau})_{0,\Omega}, & \text{problem (1)}, \\ (\mathcal{A}_\lambda\,\boldsymbol{\sigma},\boldsymbol{\tau})_{0,\Omega} + (\gamma\,\mathbf{div}\,\boldsymbol{\sigma},\mathbf{div}\,\boldsymbol{\tau})_{0,\Omega}, & \text{problem (6)} \end{array}\right.$$

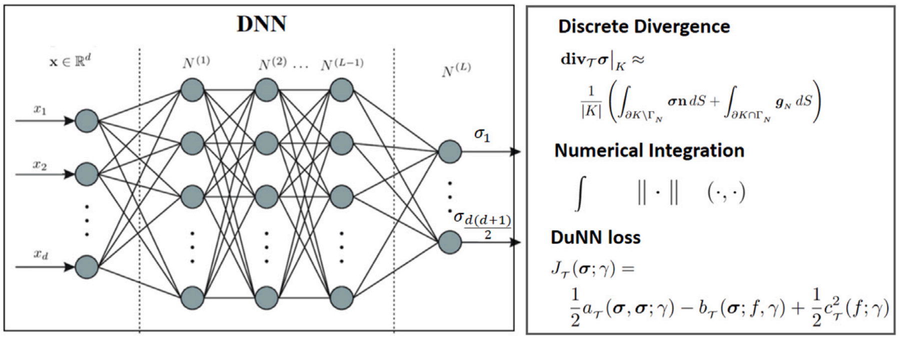

DNN
$\mathbf{x} \in \mathbb{R}^d$
$N^{(1)}$
$N^{(2)} \dots N^{(L-1)}$
$N^{(L)}$
x_1
x_2
x_d
σ_1
$\sigma_{\frac{d(d+1)}{2}}$
Discrete Divergence
$\mathbf{div}_{\mathcal{T}}\boldsymbol{\sigma}\big|_K \approx$
$\frac{1}{|K|}\left(\int_{\partial K\setminus\Gamma_N} \boldsymbol{\sigma}\mathbf{n}\,dS + \int_{\partial K\cap\Gamma_N} \boldsymbol{g}_N\,dS\right)$
Numerical Integration
$\int \quad \|\cdot\| \quad (\cdot,\cdot)$
DuNN loss
$J_{\mathcal{T}}(\boldsymbol{\sigma};\gamma) =$
$\frac{1}{2}a_{\mathcal{T}}(\boldsymbol{\sigma},\boldsymbol{\sigma};\gamma) - b_{\mathcal{T}}(\boldsymbol{\sigma};f,\gamma) + \frac{1}{2}c_{\mathcal{T}}^2(f;\gamma)$

Fig. 1. DuNN architecture. A fully connected L-layer network is employed to generate the map from an arbitrary spatial point $\boldsymbol{x}$ in Ω to its flux $\boldsymbol{\sigma}(\boldsymbol{x})$, quadrature based numerical integration and discrete divergence operator are used to approximate the discrete complementary energy functional $J_{\mathcal{T}}(\boldsymbol{\sigma};\gamma)$ as the DuNN loss.

and

$$b(\boldsymbol{\tau};f,\gamma) = \begin{cases} (\gamma\, f, \operatorname{div}\boldsymbol{\tau})_{0,\Omega} - \int_{\Gamma_D} g_D\, \boldsymbol{\tau}\cdot\mathbf{n}\, ds, & \text{problem (1)},\\ (\gamma\, \boldsymbol{f}, \mathbf{div}\,\boldsymbol{\tau})_{0,\Omega} + \int_{\Gamma_D} \boldsymbol{g}_D \cdot (\boldsymbol{\tau}\mathbf{n})\, ds, & \text{problem (6)} \end{cases}$$

respectively, where $\boldsymbol{\Sigma}_g$ is a subset of $H(\text{div};\Omega)^d$ satisfying constraints like essential boundary condition, symmetry, and/or the equilibrium equation (see (5) and (9)). Define the negative complementary functional by

$$J(\boldsymbol{\tau};\gamma) = \frac{1}{2}\, a(\boldsymbol{\tau},\boldsymbol{\tau};\gamma) - b(\boldsymbol{\tau};f,\gamma) + \frac{1}{2}\, c^2(f;\gamma), \tag{10}$$

where $c(f;\gamma) = \|\gamma^{1/2} f\|_{0,\Omega}$ or $c(f;\gamma) = \|\gamma^{1/2}\boldsymbol{f}\|_{0,\Omega}$ for problems (1) or (6), respectively, is a constant. Then the dual formulation is to seek $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_g$ such that

$$J(\boldsymbol{\sigma};\gamma) = \min_{\boldsymbol{\tau}\in\boldsymbol{\Sigma}_g} J(\boldsymbol{\tau};\gamma). \tag{11}$$

Assume that there exists a positive $\gamma_0 > 0$ such that $\gamma(\boldsymbol{x}) \geq \gamma_0$. Then the solution $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_g$ of (11) satisfies

$$a(\boldsymbol{\sigma},\boldsymbol{\tau};\gamma) = b(\boldsymbol{\tau};f,\gamma), \quad \forall\ \boldsymbol{\tau} \in \boldsymbol{\Sigma}_0. \tag{12}$$

3. Dual neural network (DuNN) method

In this section, we describe the dual neural network (DuNN) method. Simply, the DuNN method is a discretization method for solving a partial differential equation or system based on the dual formulation of the underlying problem. DuNN includes a standard fully connected DNN as the class of approximating functions and the negative complementary energy functional $J_{\mathcal{T}}(\boldsymbol{\sigma};\gamma)$ as the loss functional estimated by numerical integration and differentiation (discrete divergence operator). The general structure of the DuNN is illustrated in Fig. 1.

3.1. Deep neural network

For $j = 1,\dots,l-1$, let $N^{(j)} : \mathbb{R}^{n_{j-1}} \to \mathbb{R}^{n_j}$ be the vector-valued ridge function of the form

$$N^{(j)}(\mathbf{x}^{(j-1)}) = \zeta(\boldsymbol{\omega}^{(j)}\mathbf{x}^{(j-1)} - \mathbf{b}^{(j)}) \quad \text{for} \quad \mathbf{x}^{(j-1)} \in \mathbb{R}^{n_{j-1}}, \tag{13}$$

where $\boldsymbol{\omega}^{(j)} \in \mathbb{R}^{n_j\times n_{j-1}}$ and $\mathbf{b}^{(j)} \in \mathbb{R}^{n_j}$ are the respective weights and biases to be determined; $\mathbf{x}^{(0)} = \mathbf{x}$; and $\zeta(t)$ is the activation function and its application to a vector is defined component-wise. There are many choices of activation functions such as ReLU, logistic, Gaussian, hyperbolic tangent, and sigmoids (see, e.g. [20]).

Let $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{d_o\times n_{l-1}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^{d_o}$ be the output weights and bias, respectively, where $d_o = d$ for problem (1) and $d_o = 3(d-1)$ for problem (6). Then a l-layer neural network generates the following set of vector-valued functions in $\mathbb{R}^{d_o}$

$$\begin{aligned} \mathcal{M}_M &= \mathcal{M}_M(\zeta) = \mathcal{M}_M(\zeta,l) \\ &= \left\{\boldsymbol{\omega}^l\left(N^{(l-1)}\circ\cdots\circ N^{(1)}(\mathbf{x})\right) - \boldsymbol{b}^l :\ \boldsymbol{\omega}^{(j)} \in \mathbb{R}^{n_j\times n_{j-1}}, \mathbf{b}^{(j)} \in \mathbb{R}^{n_j} \text{ for all } j\right\}, \end{aligned} \tag{14}$$

where the symbol $\circ$ denotes the composition of functions.

This class of functions is rich enough to accurately approximate any continuous function defined on a compact set $\Omega \in \mathbb{R}^d$ (see [21,22] for the universal approximation property). However, this is not the main reason why NNs are so effective in practice. One way to understand its approximation power is from the point view of polynomial spline functions with free knots [23]. The set $\mathcal{M}_M(\zeta,2)$ may be regarded as a beautiful extension of free knot splines from one dimensional scalar-valued function to multi-dimensional vector-valued function. It has been shown that the approximation of functions by splines can generally be dramatically improved if the knots are free.

3.2. DuNN method

The DuNN method is a discretization method for approximating the solution of partial differential equations or systems based on the dual formulation and using neural networks as approximating functions. The resulting discrete, non-convex minimization problem of the DuNN method is sophisticated and computationally intensive and can be numerically solved using existing iterative methods such as ADAM, BFGS, etc.

Notice that $\mathcal{M}_M$ is a subset of $\boldsymbol{C}^0(\Omega)^{d_o}$ due to the continuity of the activation function $\zeta(t)$. Hence, $\mathcal{M}_M(\zeta)\cap\boldsymbol{\Sigma}_g$ is the set of admissible functions for the minimization problem in (11). The DuNN method is then to seek an approximation by minimizing the negative complimentary functional in the set of neural network functions $\mathcal{M}_M(\zeta)\cap\boldsymbol{\Sigma}_g$. To design a viable DuNN method, we need to address the following three numerical issues: (1) numerical integration, (2) discrete divergence operator, and (3) the constraints (Neumann boundary conditions and symmetry of the stress for the PDE system) on $\boldsymbol{\Sigma}_g$.

First, unlike finite element methods, numerical integration for NN-based methods is a nontrivial matter. The difficulty stems from the fact that the NN approximation function is unknown, and hence so is its physical partition [14,24]. To overcome this obstacle, we recently introduced an adaptive quadrature method in [8] to achieve the prescribe accuracy with fewer integration points. In this paper, we consider only the composite midpoint quadrature rule on a fixed partition for simplicity of presentation and refer readers to [8] for accurate and efficient numerical integration. To this end, partition the domain Ω by a collection of subdomains

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

such that

$$\bar{\Omega}=\cup_{K\in\mathcal{T}}\bar{K}\quad\text{and}\quad K\cap T=\emptyset,\quad\forall\ K,T\in\mathcal{T}.$$

That is, the union of all subdomains of $\mathcal{T}$ equals to the whole domain Ω, and any two distinct subdomains of $\mathcal{T}$ have no intersection. The resulting partitions of the boundary Γ_D and Γ_N are

$$\mathcal{E}_D=\{E=\partial K\cap\Gamma_D\ :\ K\in\mathcal{T}\}\quad\text{and}\quad\mathcal{E}_N=\{E=\partial K\cap\Gamma_N\ :\ K\in\mathcal{T}\},$$

respectively. Denote by $\boldsymbol{x}_K$ and $|K|$ the respective centroid and volume of element $K\in\mathcal{T}$, and by $\mathbf{x}_E$ and $|E|$ the respective centroid and area of boundary element $E\in\mathcal{E}_S$ for $S=D$ and N. Then

$$\int_\Omega v(\boldsymbol{x})\,d\boldsymbol{x}\approx\sum_{K\in\mathcal{T}}v(\boldsymbol{x}_K)|K|\quad\text{and}\quad\int_{\Gamma_S}v(\boldsymbol{x})\,ds\approx\sum_{E\in\mathcal{E}_S}v(\boldsymbol{x}_E)|E|.\tag{15}$$

Second, numerical differentiation becomes a critical component for a viable DuNN method. This difficulty stems from the fact that the admissible solution set $\boldsymbol{\Sigma}_g$ whose functions may not be continuous in tangential directions across some interfaces. Hence, the divergence differential operator cannot be approximated by standard finite difference scheme along coordinate directions or auto-differentiation. To circumvent this obstacle, we use a newly developed discrete divergence operator introduced in [25] to approximate the divergence operator. Below let us briefly define the discrete divergence operator denoted by $\mathbf{div}_\mathcal{T}\boldsymbol{\tau}$ for any $\boldsymbol{\tau}\in\boldsymbol{\Sigma}_g=\boldsymbol{H}^s_{g,N}(\text{div};\Omega)$, that may be defined for any $\boldsymbol{\tau}\in\boldsymbol{\Sigma}_g=H_{g,N}(\text{div};\Omega)$ in a similar fashion. The $\mathbf{div}_\mathcal{T}\boldsymbol{\tau}$ is a piece-wise constant vector field and its restriction on each $K\in\mathcal{T}$ is an approximation to the average of $\mathbf{div}\,\boldsymbol{\tau}$, i.e.,

$$\mathbf{div}_\mathcal{T}\boldsymbol{\tau}\Big|_K\approx\text{avg}_K\mathbf{div}\,\boldsymbol{\tau}=\frac{1}{|K|}\left(\int_{\partial K\setminus\Gamma_N}\boldsymbol{\tau}\mathbf{n}\,dS+\int_{\partial K\cap\Gamma_N}\boldsymbol{g}_N\,dS\right),\tag{16}$$

where $\mathbf{n}$ is the outward unit vector normal to ∂K, the boundary of K. Surface integrals in (16) may be approximated by either proper standard or adaptive numerical integration.

Third, the symmetry of $\boldsymbol{\Sigma}_g=\boldsymbol{H}^s_{g,N}(\text{div};\Omega)$ for problem (6) can be easily enforced strongly by setting $\sigma_{ij}=\sigma_{ji}$ so that the stress has only $d_o=3(d-1)$ variables. The Neumann boundary condition in $\boldsymbol{\Sigma}_g$ for both problems becomes an essential boundary condition in the dual formulation (11). One may penalize the complementary functional in (10) by adding either the $H^{-1/2}$ or a weighted L^2 norm of the residual of the Neumann boundary condition. This type of treatments has been discussed for the deep Ritz method (see, e.g., [8]). An attractive feature of the discrete divergence operator defined in (16) is that the Neumann boundary condition is already weakly enforced. Therefore, it is not necessary to enforce it by adding penalization terms. Adjusting the penalization coefficient is, in general, nontrieval, and therefore, using the discrete divergence operator simplifies the training process.

Now, for the simple composite midpoint quadrature rule, we are ready to define the discrete negative complementary functional as

$$J_\mathcal{T}(\boldsymbol{\tau};\gamma)=\frac{1}{2}a_\mathcal{T}(\boldsymbol{\tau},\boldsymbol{\tau};\gamma)-b_\mathcal{T}(\boldsymbol{\tau};f,\gamma)+\frac{1}{2}c^2_\mathcal{T}(f;\gamma),\tag{17}$$

where $c_\mathcal{T}(f;\gamma)=\sum_{K\in\mathcal{T}}|K|\big(\gamma f^2\big)(\mathbf{x}_K)$ for problem (1) and $c_\mathcal{T}(f;\gamma)=\sum_{K\in\mathcal{T}}|K|\big(\gamma\,|\boldsymbol{f}|^2\big)(\mathbf{x}_K)$ for problem (6), and the discrete quadratic and linear forms are given by

$$a_\mathcal{T}(\boldsymbol{\tau},\boldsymbol{\tau};\gamma)=\begin{cases}\displaystyle\sum_{K\in\mathcal{T}}|K|\left\{\boldsymbol{\tau}^TA^{-1}\boldsymbol{\tau}+\gamma\left(\text{div}_\mathcal{T}\boldsymbol{\tau}\right)^2\right\}(\mathbf{x}_K), & \text{problem (1)},\\ \displaystyle\sum_{K\in\mathcal{T}}|K|\left\{\frac{1}{2\mu}\left|\boldsymbol{\tau}^D\right|^2+\frac{1}{d(2\mu+d\lambda)}|\text{tr}\boldsymbol{\tau}|^2+\gamma\left|\mathbf{div}_\mathcal{T}\boldsymbol{\tau}\right|^2\right\}(\mathbf{x}_K), & \text{problem (6)},\end{cases}$$

and

$$b_{\mathcal{T}}(\boldsymbol{\tau};f,\gamma)=\begin{cases}\displaystyle\sum_{K\in\mathcal{T}}|K|\big(\gamma\, f\,\mathrm{div}\boldsymbol{\tau}\big)(\mathbf{x}_K)-\sum_{E\in\mathcal{E}_D}|E|\big(g_D\boldsymbol{\tau}\cdot\mathbf{n}\big)(\mathbf{x}_E), & \text{problem (1)},\\ \displaystyle\sum_{K\in\mathcal{T}}|K|\big(\gamma\,\boldsymbol{f}\cdot\mathbf{div}\,\boldsymbol{\tau}\big)(\mathbf{x}_K)+\sum_{E\in\mathcal{E}_D}|E|\big(\boldsymbol{g}_D\cdot(\boldsymbol{\tau}\mathbf{n})\big)(\mathbf{x}_E), & \text{problem (6)},\end{cases}$$

respectively. Then, the dual neural network (DuNN) method is to find $\boldsymbol{\sigma}_{\mathcal{T}}\in\mathcal{M}_M\cap\boldsymbol{\Sigma}$ such that

$$J_{\mathcal{T}}(\boldsymbol{\sigma}_{\mathcal{T}};\gamma)=\min_{\boldsymbol{\tau}\in\mathcal{M}_M\cap\boldsymbol{\Sigma}}J_{\mathcal{T}}(\boldsymbol{\tau};\gamma), \tag{18}$$

where $\boldsymbol{\Sigma}=H(\mathrm{div};\Omega)$ for problem (1) and $\boldsymbol{\Sigma}=\boldsymbol{H}^s(\mathrm{div};\Omega)$ for problem (6).

To understand the effect of numerical integration and differentiation, we extend the first Strang lemma for the Galerkin approximation over a subspace (see, e.g, [26]) to the DuNN approximation over a subset.

Theorem 1. *Assume that there exists a positive constant α independent of $\mathcal{M}_{2M}\cap\boldsymbol{\Sigma}$ such that*

$$\alpha\,\|\boldsymbol{\tau}\|_a^2\le a_{\mathcal{T}}(\boldsymbol{\tau},\boldsymbol{\tau}),\quad\forall\,\boldsymbol{\tau}\in\mathcal{M}_{2M}\cap\boldsymbol{\Sigma}. \tag{19}$$

Let $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}_{\mathcal{T}}\in\mathcal{M}_M$ be the solutions of (11) and (18), respectively. Then there exists a positive constant C such that

$$\|\boldsymbol{\sigma}-\boldsymbol{\sigma}_{\mathcal{T}}\|_a\le C\Big(\inf_{\boldsymbol{\tau}\in\mathcal{M}_{2M}\cap\boldsymbol{\Sigma}}\boldsymbol{E}(\boldsymbol{\tau})+\sup_{\boldsymbol{\tau}\in\mathcal{M}_{2M}\cap\boldsymbol{\Sigma}}|f(\boldsymbol{\tau})-f_{\mathcal{T}}(\boldsymbol{\tau})|/\|\boldsymbol{\tau}\|_a\Big), \tag{20}$$

where $\boldsymbol{E}(\boldsymbol{\tau})=\|\boldsymbol{\sigma}-\boldsymbol{\tau}\|_a+\sup_{\boldsymbol{v}\in\mathcal{M}_{2M}\cap\boldsymbol{\Sigma}}|a(\boldsymbol{\tau},\boldsymbol{v})-a_{\mathcal{T}}(\boldsymbol{\tau},\boldsymbol{v})|/\|\boldsymbol{v}\|_a$.

Proof. For any $\boldsymbol{\tau}\in\mathcal{M}_M\cap\boldsymbol{\Sigma}$, let $\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})=\boldsymbol{\sigma}^{\epsilon}_{\mathcal{T}}-\boldsymbol{\tau}$. It is easy to see that

$$J_{\mathcal{T}}(\boldsymbol{\sigma}^{\epsilon}_{\mathcal{T}};\gamma_\epsilon)\le J_{\mathcal{T}}(\boldsymbol{\tau};\gamma_\epsilon)\quad\text{and}\quad a(\boldsymbol{\sigma}^{\epsilon},\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau}))=f(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau}))+g(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})),$$

where $g(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau}))=$. This, together with the assumption in (19), implies

$$\begin{aligned}\frac{\alpha}{2}\left\|\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right\|_a^2&\le\frac{1}{2}a_{\mathcal{T}}\left(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau}),\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right)\le f_{\mathcal{T}}\left(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right)-a_{\mathcal{T}}\left(\boldsymbol{\tau},\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right)\\&=\Big(f_{\mathcal{T}}\left(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right)-f\left(\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right)\Big)+\Big(a\big(\boldsymbol{\tau},\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\big)-a_{\mathcal{T}}\left(\boldsymbol{\tau},\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right)\Big)+a\big(\boldsymbol{\sigma}-\boldsymbol{\tau},\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\big).\end{aligned}$$

Since $\left|f(\boldsymbol{\tau})-f_{\mathcal{T}}(\boldsymbol{\tau})\right|\le\|\boldsymbol{\tau}\|_a\sup_{\boldsymbol{w}\in\mathcal{M}_{2M}}|f(\boldsymbol{w})-f_{\mathcal{T}}(\boldsymbol{w})|/\|\boldsymbol{w}\|_a$, by the Cauchy–Schwarz inequality and the fact that $\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\in\mathcal{M}_{2M}$, we have

$$\left\|\boldsymbol{e}_{\mathcal{T}}(\boldsymbol{\tau})\right\|_a^2\le C\left(\boldsymbol{E}(\boldsymbol{\tau})+\sup_{\boldsymbol{\tau}\in\mathcal{M}_{2M}}|f(\boldsymbol{\tau})-f_{\mathcal{T}}(\boldsymbol{\tau})|/\|\boldsymbol{\tau}\|_a\right).$$

Now, the validity of (20) follows from using the triangle inequality and taking infimum over all $\boldsymbol{\tau}\in\mathcal{M}_{2M}\cap\boldsymbol{\Sigma}$. This completes the proof of the theorem. □

Theorem 1 indicates that the total error in the energy norm is bounded by the approximation error of the set of neural network functions plus the numerical integration and differentiation error.

3.3. Constrained minimization

In the case that $\gamma=0$, i.e., $c=0$ in (1) or (6), (11) is a constrained minimization problem. One may use the method of Lagrange multiplier or penalty. The former leads to a saddle point problem and the latter has difficulty to choose a proper penalization parameter that is good in both accuracy and efficiency. On one hand, a standard perturbation theory [17] suggests that the penalization parameter (still denoted by γ) should be $\gamma=\epsilon^{-1}$ with $0<\epsilon\ll 1$ for accuracy. On the other hand, this choice leads to an ill-conditioned algebraic problem.

This section introduces an iterative procedure to gradually enforce the equilibrium equation. For simplicity of presentation, we describe the procedure at the continuous level. Let δ_{k-1} be the previous time step size and $u^{(k)}$ and $\boldsymbol{u}^{(k)}$ are the previous approximation to the solution of problem (1) and problem (6), respectively. Set

$$f^{(k)}=\begin{cases}f+\delta_k^{-1}u^{(k)}, & \text{problem (1)},\\ \boldsymbol{f}+\delta_k^{-1}\boldsymbol{u}^{(k)}, & \text{problem (6)}.\end{cases}$$

Given the previous approximation $\boldsymbol{\sigma}^{(k)}$ to the solution of (11), define the following negative complementary functional at the kth step by

$$J^{(k)}(\boldsymbol{\tau})=\frac{1}{2}\,a(\boldsymbol{\tau},\boldsymbol{\tau};\delta_k)-b(\boldsymbol{\tau};f^{(k)},\delta_k)+\frac{1}{2}\,c^2(f^{(k)};\delta_k). \tag{21}$$

Then the iterative procedure is to find $\boldsymbol{\sigma}^{(k+1)}\in\boldsymbol{\Sigma}_g$ such that

$$J^{(k)}\left(\boldsymbol{\sigma}^{(k+1)}\right)=\min_{\boldsymbol{\tau}\in\boldsymbol{\Sigma}_g}J^{(k)}(\boldsymbol{\tau}) \tag{22}$$

Table 1
Relative L^2 errors for test example I using three DNN structures (2-128-128-d_o| 2-64-64-64-d_o| 2-96-96-96- d_o, where $d_o = 2$ for DuNN, and $d_o = 1$ for PINN and Deep Ritz).

	Method	$\frac{\|u - u^N\|}{\|u\|}$	$\frac{\|\sigma - \sigma^N\|}{\|\sigma\|}$
$\varepsilon = 0.05$	PINN	13.19% | 12.63% | 12.38%	48.04% | 42.63% | 37.97%
	Deep Ritz	**0.984%** | **0.910%** | **0.904%**	16.73% | 12.00% | 12.07%
	DuNN	4.248% | 2.682% | 2.227%	**4.957%** | **2.826%** | **2.190%**
$\varepsilon = 0.005$	PINN	8.522% | 5.814% | 2.727%	73.01% | 57.81% | 34.33%
	Deep Ritz	**2.382%** | **1.056%** | **0.997%**	32.18% | 28.67% | 28.49%
	DuNN	3.019% | 1.751% | 1.524%	**24.04%** | **12.57%** | **9.385%**

***training details:**
1. Activation function: ReLU;
2. numerical integration: 400 × 360 uniformly distributed quadrature points;
3. Adam optimization: 80,000 iterations; learning rate starts with 0.004 and decays 50% per 10,000 iterations;
4. penalization coefficient in loss function: for PINN, $\gamma_D = 100$, and for Deep Ritz, $\gamma_D = 1$.

and set

$$\begin{cases} u^{(k+1)} = \delta_k \left(f - \operatorname{div} \boldsymbol{\sigma}^{(k+1)} \right) + u^{(k)}, & \text{problem (1)}, \\ \boldsymbol{u}^{(k+1)} = \delta_k \left(\boldsymbol{f} - \operatorname{\mathbf{div}} \boldsymbol{\sigma}^{(k+1)} \right) + \boldsymbol{u}^{(k)}, & \text{problem (6)} \end{cases} \quad \text{and} \quad f^{(k+1)} = \begin{cases} f + \delta_k^{-1} u^{(k+1)}, & \text{problem (1)}, \\ \boldsymbol{f} + \delta_k^{-1} \boldsymbol{u}^{(k+1)}, & \text{problem (6)}. \end{cases}$$

4. Numerical studies

In this section, we present our numerical studies on several second-order elliptic PDEs. Existing NN-based methods include the deep Ritz [1] and PINN [5], which are based on primal and primitive LS formulations, respectively. Essential boundary condition(s) (Dirichlet for the primal and both Dirichlet and Neumann for the primitive LS) are enforced by penalizing them in the loss functional. The deep Ritz has recently been extended to linear elasticity in [8,27]. We will compare the proposed DuNN with the aforementioned NN-based methods.

In all experiments, the structure of the DNN used is expressed as d-n_1-$n_2 \cdots n_{l-1}$-d_o for a l-layer network with n_1, n_2 and n_{l-1} neurons in the respective first, second, and $(l-1)$th layers. The d and d_o represent the input and output dimensions of the network. For DuNN, $d_o = 3(d-1)$, and for deep Ritz and PINN, $d_o = d$. The minimization of the loss functionals in all experiments is solved using the Adam optimization algorithm [28].

4.1. *Test example I: a two-dimensional singularly perturbed reaction–diffusion problem*

Consider the following 2D scalar reaction–diffusion problem:

$$-\varepsilon^2 \Delta u + u = f \quad \text{in} \quad \Omega, \quad u = 0 \quad \text{on} \quad \partial\Omega,$$

with the true solution $u = tanh(\frac{1}{\varepsilon}(x^2 + y^2 - \frac{1}{4})) - tanh(\frac{3}{4\varepsilon})$ defined in the unit disk $\Omega = \{(x, y) \in \mathbb{R}^2 : x^2 + y^2 < 1\}$. Consider the problem in two cases: $\varepsilon = 0.05$ and $\varepsilon = 0.005$, and note that there is a sharp interior transition layer at $r = \sqrt{x^2 + y^2} = 1/2$ with a width of order ε in the solution. When ε is small, there is a numerical difficulty in solving these types of problem.

Set the flux $\boldsymbol{\sigma} = -\varepsilon^2 \nabla u$, and with the vanish boundary condition, the corresponding DuNN loss functional using the complementary energy (2) is reduced to

$$J^*(\boldsymbol{\tau}) = \frac{1}{2} \left\{ \left\| \varepsilon^{-1} \boldsymbol{\tau} \right\|_{0,\Omega}^2 + \| (\operatorname{div} \boldsymbol{\tau} - f) \|_{0,\Omega}^2 \right\}. \tag{23}$$

To compare, we tested the deep Ritz and PINN methods as well. Both deep Ritz and PINN use DNNs to approximate the primary variable u. Deep Ritz employs the following energy-based loss functional,

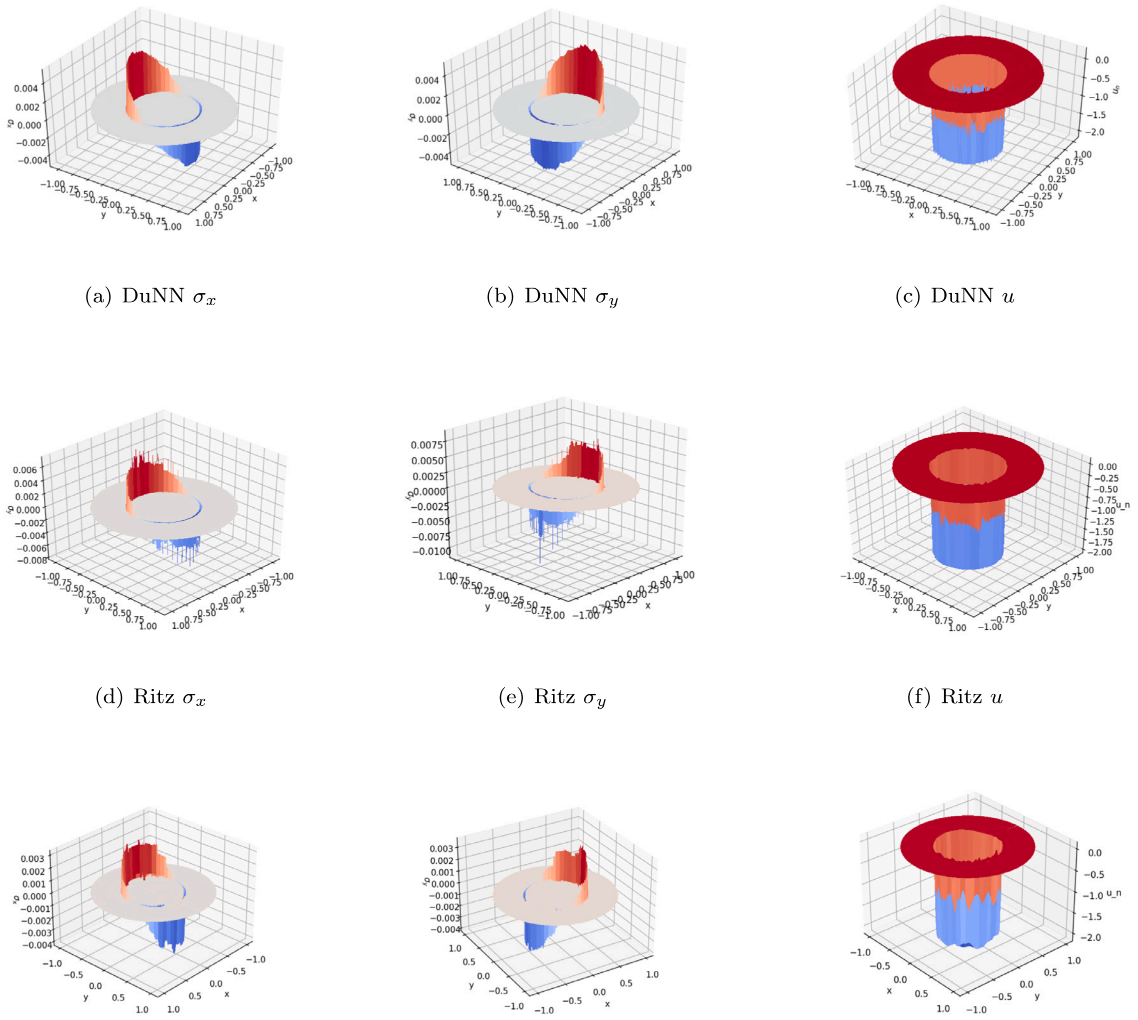
$$J(v) = \frac{1}{2} \left\{ \|\varepsilon \nabla v\|_{0,\Omega}^2 + \|v\|_{0,\Omega}^2 + \gamma_D \|v\|_{1/2,\partial\Omega}^2 \right\} - (v, f), \tag{24}$$

while PINN uses a direct least square loss functional,

$$L(v) = \| - \varepsilon^2 \Delta v + v - f \|_{0,\Omega}^2 + \gamma_D \|v\|_{0,\partial\Omega}^2, \tag{25}$$

where γ_D is the penalization coefficient.

Table 1 reports the results of the three methods. As shown in the table, for both material cases and the three different DNN structures(a three-layer DNN with 128 neurons in the hidden layer, and two four-layer DNNs with 64 and 96 neurons in the hidden layer, respectively), DuNN achieves better accuracy in approximating the flux $\boldsymbol{\sigma}$, and Deep Ritz performs better in approximating the primary variable u. As illustrated in Fig. 2, the DuNN method yields a direct approximation of the flux $\boldsymbol{\sigma}$, which results in fewer numerical oscillations; see Figs. 2(a) and 2(b). The other two methods calculate the flux $\boldsymbol{\sigma}$ indirectly using $\boldsymbol{\sigma} = -\varepsilon^2 \nabla u$, which



(a) DuNN σ_x (b) DuNN σ_y (c) DuNN u

(d) Ritz σ_x (e) Ritz σ_y (f) Ritz u

(g) PINN σ_x (h) PINN σ_y (i) PINN u

Fig. 2. Numerical Results of test example I ($\varepsilon = 0.005$) using three NN-based methods.

involves a differential operation[1] on the DNN output function u. As shown in Figs. 2(d) 2(e) and 2(g) 2(h)), this leads to some numerical oscillations in flux simulation.

4.2. Test example II: two-dimensional Poisson equation

The second test problem is a two-dimensional Poisson equation defined on a square unit $\Omega = (0,1)\times(0,1)$. The exact solution for the primary variable $u = sin(\frac{\pi}{2}x)sin(\pi y) + x^2y^2$. And the dual variable $\boldsymbol{\sigma}$ has the analytic form,

$$\boldsymbol{\sigma} = -\nabla u = \begin{pmatrix} -\frac{\pi}{2}cos(\frac{\pi}{2}x)sin(\pi y) - 2xy^2 \\ -\pi sin(\frac{\pi}{2}x)cos(\pi y) - 2x^2y \end{pmatrix}.$$

With the right-hand side $f = \operatorname{div}\boldsymbol{\sigma} = \frac{5\pi^2}{4}sin(\frac{\pi}{2}x)sin(\pi y) - 2(x^2+y^2)$, and the Dirichlet boundary condition defined in $x = 0$ and $y = 0$, the Neumann boundary prescribed in $x = 1$ and $y = 1$, we tested the performance of DuNN and compared it with the deep Ritz and PINN. Specifically, for DuNN, since the primary variable term vanishes in the Poisson equation ($c = 0$), we tested two approaches to

[1] In our experiments, numerical differentiation was used to obtain the results.

Table 2
Relative L^2 errors for test example II using penalization method (DNN structure: 2-50-50-d_o).

DuNN		Deep Ritz		PINN	
$\gamma = 20$		$\gamma_D = 1000$		$\gamma_D = 10000, \gamma_N = 10000$	
$\frac{\|\boldsymbol{u} - \boldsymbol{u}^N\|}{\|\boldsymbol{u}\|}$	$\frac{\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^N\|}{\|\boldsymbol{\sigma}\|}$	$\frac{\|\boldsymbol{u} - \boldsymbol{u}^N\|}{\|\boldsymbol{u}\|}$	$\frac{\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^N\|}{\|\boldsymbol{\sigma}\|}$	$\frac{\|\boldsymbol{u} - \boldsymbol{u}^N\|}{\|\boldsymbol{u}\|}$	$\frac{\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^N\|}{\|\boldsymbol{\sigma}\|}$
0.740%	1.307%	0.949%	3.730%	0.594%	1.3224%

***training details:**
1. activation function: Sigmoid;
2. numerical integration: 100 × 100 uniform distributed quadrature points (h = 0.02).
3. Adam optimization: 200,000 iterations;
learning rate starts with 0.01 and decays 90% per 20,000 iterations until reaches 1e-5.

Table 3
Relative L^2 errors for test example II using pseudo-time method (DNN structure: 2-50-50-2).

Time step size δ	0.1	0.05	0.01	0.005	0.001
Inner iteration per time step	5,000	2,500	500	250	50
Outer iteration number	20	40	200	400	2000
$\frac{\|\boldsymbol{\sigma} - \boldsymbol{\sigma}^N\|}{\|\boldsymbol{\sigma}\|}$	0.273%	0.221%	0.182%	0.134%	0.173%

***training details:**
1. Total number of iterations: 100,000;
2. learning rate is $1e-3$ for the first 50,000 iterations and $1e-4$ for the rest.

solve the corresponding constrained minimization problem. The first is the penalization method that uses the added penalty term $\gamma\ \|(\operatorname{div}\tau - f)\|^2_{0,\Omega}$, where γ is a penalization coefficient that needs to be adjusted. And the second method is the outer-inner iterative procedure using pseudo-time described in Section 3.3.

Using the penalization method, DuNN needs to tune one parameter γ for the force balance term, deep Ritz needs one parameter γ_D for the Dirichlet boundary condition term, and PINN needs two parameters, γ_D and γ_N for both the Dirichlet and Neumann boundary condition terms. Table 2 reports the results of the comparison. In all three methods, we adjusted the penalization coefficients in their respective loss functions and reported the best results. Note that in the DuNN method, the primary variable u is reconstructed using another DNN of the same structure (2-50-50-1), and the loss function for reconstructing u is $L(v) = \|\nabla v + \boldsymbol{\sigma}_N\|^2_{0,\Omega} + \gamma_D\|v - g_D\|^2_{0,\partial\Omega}$, where $\boldsymbol{\sigma}_N$ is the obtained numerical flux from DuNN. The tuned penalization parameters are shown in the second row of the Table 2. From the error measures shown in the last row of the table, we can see that for smooth problems like the one in this test, all three methods perform well if the hyper parameters are tuned into the appropriate scales.

We then tested the outer-inner pseudo-time method for the constrained minimization problem. The same DNN structure (2-50-50-2) and activation function (Sigmoid) were used as in the penalization method previously. In the experiment, the pseudo-time step size remained constant throughout the outer-inner iterations. Table 3 records the numerical results of using different pseudo-time step sizes δ. It is found that the pseudo-time iterative method, compared to the penalization method, is less sensitive to the parameter (time-step size) and converges to a better solution in fewer iterations (a total of 100,000 iterations). We also observed that, in general, a larger step size requires more inner iterative steps, as reported in the second row of the table. Figs. 3(a) and 3(b) plot the numerical results for the approximate flux $\boldsymbol{\sigma}$ using the time step size $\delta = 0.005$.

Another benefit of the pseudo-time method is that the added term u_t converges to u_N, becoming a byproduct of the iterative process. Fig. 3(c) illustrates the resulting u_N. Alternatively, one may also reconstruct u_N using another DNN, as previously used. Recovering u_N using another DNN requires additional time and resources to form the approximated primary variable u, but produces a smoother result in this case, as shown in Fig. 3(d). The effectiveness of the pseudo-time method is further demonstrated in Fig. 3(e). For each time step, the force balance term $\|(\operatorname{div}\tau - f)\|^2_{0,\Omega}$ continuously decays to nearly zero, and the negative complementary energy converges to the true value of this problem, which is 1.571.

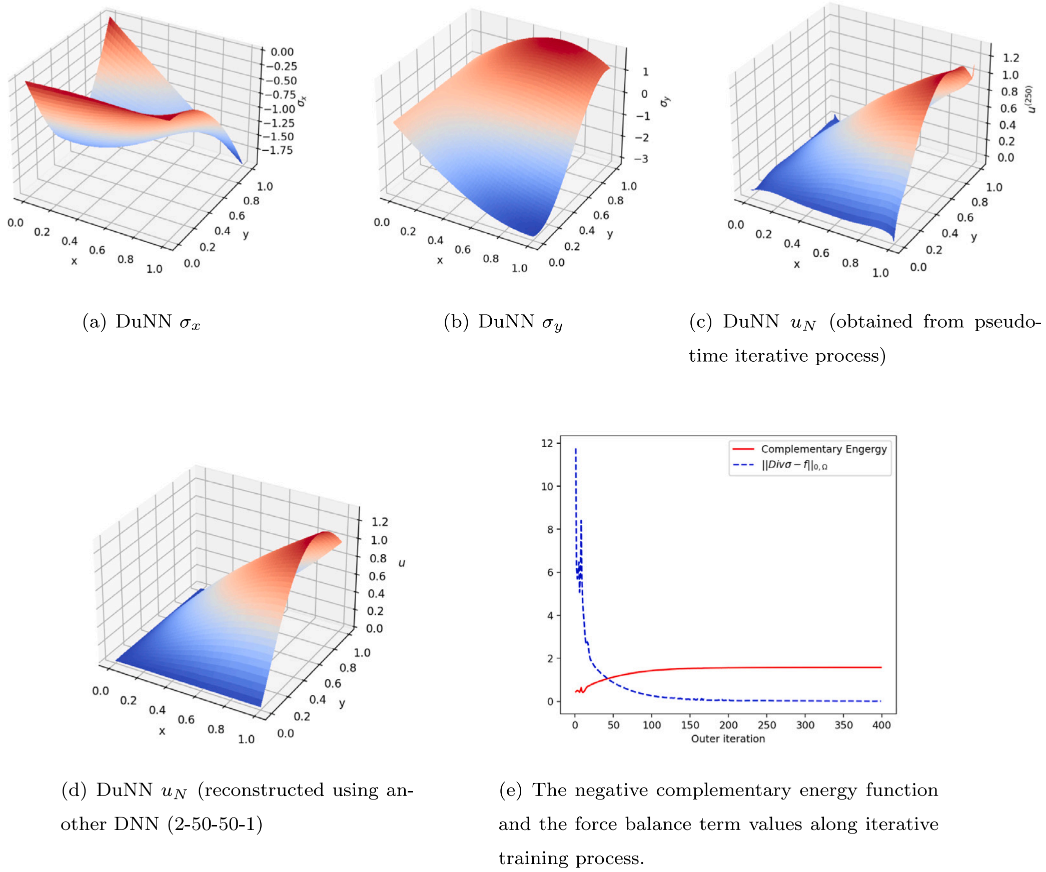
4.3. Test example III: L-shaped linear elastic plate under stress

The last test example is a common benchmark problem for linear elasticity Eq. (6) featuring a re-entrant corner and a resulting point singularity [29]. This problem is posed on a L−shaped domain $\Omega = (-1,1)^2 \setminus ([0,1]\times[-1,0])$ with a body force $\boldsymbol{f} = \boldsymbol{0}$. The analytical solution for displacement $\boldsymbol{u}$ is,

$$\boldsymbol{u} = [A\cos\theta - B\sin\theta, A\sin\theta - B\cos\theta]^T,$$

where A and B are defined in polar coordinates:

$$\begin{cases} A = \frac{r^\alpha}{2\mu}\Big(-(1+\alpha)\cos\big((1+\alpha)\theta\big) + C_1(C_2 - 1 - \alpha)\cos\big((1-\alpha)\theta\big)\Big), \\ B = \frac{r^\alpha}{2\mu}\Big((1+\alpha)\sin\big((1+\alpha)\theta\big) - C_1(C_2 - 1 + \alpha)\sin\big((1-\alpha)\theta\big)\Big). \end{cases}$$



(a) DuNN σ_x

(b) DuNN σ_y

(c) DuNN u_N (obtained from pseudo-time iterative process)

(d) DuNN u_N (reconstructed using another DNN (2-50-50-1)

(e) The negative complementary energy function and the force balance term values along iterative training process.

Fig. 3. Numerical Results of Poisson equation using pseudo-time outer-inner iterative method ($\delta = 0.005$).

Here $\alpha \approx 0.544483737$ is the *critical* exponent and the definition of C_1, C_2 together with the exact form of stress $\boldsymbol{\sigma}$ are referenced in [29]. We tested two materials with Young's modulus $E = 100000$ and Poisson's ratio $\nu = 0.3$ for a compressible material and $\nu = 0.49999$ for a nearly incompressible material. The Lamé constants are given by $\mu = \frac{E}{2(1+\nu)}$ and $\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}$.

The method of PINN does not apply here due to the existence of a stress singularity at the origin point $(0, 0)$. Therefore, we compare only the numerical results of the two energy-based methods: Deep Ritz [8] and DuNN.

Material case I ($\nu = 0.3$): we used only the penalization method in this case. For penalization coefficients, we tested various values and finally adjusted them to $\gamma = 1e-4$ for DuNN and $\gamma_D = 10$ for deep Ritz. Uniform quadrature methods with the midpoint quadrature rule and two set of the integration mesh sizes were tested and the corresponding results are reported in Table 4. The numerical experiments show that both energy-based methods (deep Ritz and DuNN) have the capability of handling reentrant corner singularity, while DuNN performs better in terms of relative L^2 approximation error for the numerical stress, using both integration mesh sizes.

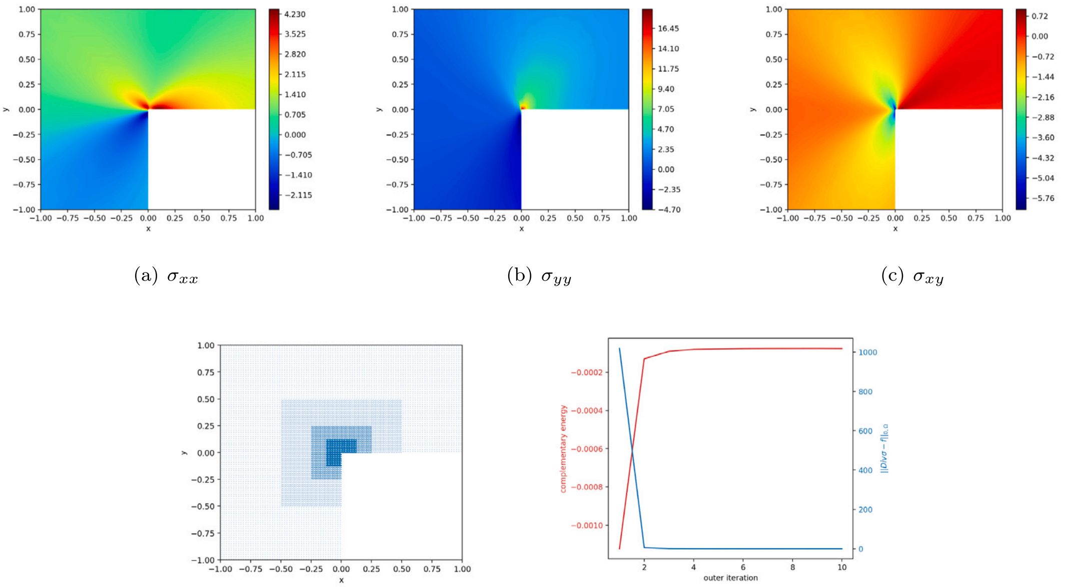
Material case II ($\nu = 0.49999$): since the deep Ritz method does not accurately characterize the stress under the near-incompressible condition (locking phenomenon), we tested DuNN alone and compared the penalization method with the pseudo-time method for the constrained minimization problem. Both the uniform and non-uniform quadrature methods were tested in this material case. For the non-uniform quadrature method, a manual integration mesh was constructed using progressive refinement near the singular point (see the corresponding non-uniform quadrature points in Fig. 3(d)). Note that this non-uniform integration mesh can be constructed adaptively using the adaptive quadrature refinement (AQR) method proposed in [8]. Since numerical integration is not a main focus of this work, we used this manually generated set of quadrature points to investigate the effect of numerical integration.

From the result shown in Table 4, we can see that both the penalization method and the pseudo-time method can handle incompressibility and simulate the stress distribution with point singularity reasonably well. The non-uniform quadrature method produced slightly better result with fewer number of quadrature points. During the training process, we also observed that the pseudo-time-based minimization, although also having a time step δ_t parameter to be determined, converges faster due to the gradual enforcement of the equilibrium equation. In this test, the penalization method required 200,000 iterations, while the pseudo-time method required less than 100,000 iterations to converge. The corresponding numerical results are plotted in Fig. 4, where subfigures

Table 4
Relative L^2 errors of numerical stress for the L-shaped problem(Network: 2-48-48-48-d_o, Activation: sigmoid).

	Method	Quadrature	$\frac{\|\sigma - \sigma_{_N}\|}{\|\sigma\|}$
$\nu = 0.3$	Deep Ritz (penalization)	uniform $h = 0.02$	38.59%
		uniform $h = 0.01$	31.62%
	DuNN (penalization)	uniform $h = 0.02$	10.81%
		uniform $h = 0.01$	10.08%
$\nu = 0.49999$	DuNN (penalization)	uniform: $h = 0.01$	12.44%
		Non-uniform	12.39%
	DuNN (pseudo-time)	uniform $h = 0.01$	10.96%
		Non-uniform	10.30%

***training details:**
1. penalization Method: DuNN: $\gamma = 1e - 4$, Deep Ritz: $\gamma_D = 10$
200,000 iterations and learning rate starts from 0.01 and decays 50% every 50,000 iterations.
2. pseudo-time method: $\delta = 1e - 6$
inner iteration: 10,000; number of time-step:10; total iteration: 100,000.



(a) σ_{xx} (b) σ_{yy} (c) σ_{xy}

(d) 30465 non-uniform quadrature points (from $h = 1/50$, refined 3 times to 24,543 points)

(e) The complementary energy and the force balance term along training process

Fig. 4. Numerical results using DuNN for the L-shaped elastic plate problem, case II ($\nu = 0.49999$) (structure: 2-48-48-48-3, activate function: sigmoid, non-uniform quadrature and pseudo-time outer-inner iterative method).

(a)–(c) represent the numerical stress distributions of DuNN. The results show that both the point singularity and incompressibility of the material are well handled using DuNN. Fig. 4(e) illustrates the pseudo-time-based negative complementary function minimization process. During the iterative process, the force balance term quickly decreases to near zero, while the negative complementary energy converges to its theoretic value.

5. Conclusion

In this paper, we established a physics-driven deep neural network-based computational framework to solve elliptic partial differential equations and systems. The problem is formulated as an optimization of the complementary energy functional with the benefit of using the sole dual variable. Combined with the physics-preserved discrete divergence operator, all boundary conditions can be enforced naturally without using any penalization term. For problems without the primary variable term, a pseudo-time-based iterative method was developed to gradually enforce the equilibrium equation.

Numerical studies demonstrate that DuNN accurately approximates dual variables for elliptic problems. Compared to existing neural network-based methods, DuNN offers superior flux prediction accuracy and is applicable to a broader range of problems, including those with discontinuities or singularities. It is also effective for problems involving both compressible and incompressible materials.

Acknowledgments

This work was supported in part by the National Science Foundation, United States under grant DMS-2110571 and we thank the support of DARPA project on symbiotic design and Stanford Research International (SRI) for partial support of the project.

Data availability

Data will be made available on request.

References

[1] W. E., B. Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[2] J. Berg, K. Nystrom, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[3] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1139–1364.
[4] M. Raissi, G.E. Karniadakis, Deep hidden physics models: Deep learning of nonlinear partial differential equations, J. Mach. Learn. Res. 19 (1) (2018) 932–955.
[5] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[6] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. USA 116 (31) (2019) 15344–15349.
[7] Z. Cai, J. Chen, M. Liu, X. Liu, Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs, J. Comput. Phys. 420 (2020) 109707.
[8] M. Liu, Z. Cai, K. Ramani, Deep Ritz method with adaptive quadrature for linear elasticity, Comput. Methods Appl. Mech. Engrg. 415 (2023) 116229.
[9] I. Daubechies, R. DeVore, S. Foucart, B. Hanin, G. Petrova, Nonlinear approximation and (Deep) ReLU networks, 2019, ArXiv Preprint ArXiv:.
[10] R. DeVore, B. Hanin, G. Petrova, Neural network approximation, Acta Numer. 30 (2021) 327–444, http://dx.doi.org/10.1017/S0962492921000052.
[11] D. Yarotsky, Error bounds for approximations with deep ReLU networks, Neural Netw. 94 (2017) 103–114.
[12] J. Xu, The finite neuron method and convergence analysis, Commun. Comput. Phys. 28 (2020) 1707–1745.
[13] M. Liu, Z. Cai, Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs, Comput. Math. Appl. 113 (2022) 103–116.
[14] Z. Cai, J. Chen, M. Liu, Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation, J. Comput. Phys. 443 (2021) 110514.
[15] I. Ekeland, R. Témam, Convex Analysis and Variational Problems, Society for Industrial and Mathematics, Philadelphia, 1999.
[16] S. Zhang, Primal-dual reduced basis methods for convex minimization variational problems: robust true solution a posteriori error certification and adaptive greedy algorithms, SIAM J. Sci. Comput. 42 (1) (2020) A3638–A3676.
[17] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Springer-Verlag, New York, 1991.
[18] Z. Cai, G. Starke, Least-squares methods for linear elasticity, SIAM J. Numer. Anal. 42 (2) (2004) 826–842.
[19] Z. Cai, B. Lee, P. Wang, Least-squares methods for incompressible Newtonian fluid flow: linear stationary problems, SIAM J. Numer. Anal. 42 (2) (2004) 843–859.
[20] A. Pinkus, Approximation theory of the MLP model in nueral networks, Acta Numer. 15 (1999) 143–195.
[21] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989) 303–314.
[22] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[23] L. Schumaker, Spline Functions: Basic Theory, John Wiley, New York, 1981.
[24] M. Liu, Z. Cai, J. Chen, Adaptive two-layer ReLU neural network: I. best least-squares approximation, Comput. Math. Appl. 113 (2022) 34–44.
[25] Z. Cai, J. Chen, 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.
[26] P.G. Ciarlet, The finite element method for elliptic problems, Society for Industrial and Applied Mathematics, 1978.
[27] W. Li, M.Z. Bazant, J. Zhu, A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches, Comput. Methods Appl. Mech. Engrg. 383 (2021) 113933.
[28] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Representation Learning, San Diego, 2015.
[29] G. Harper, J. Liu, S. Tavener, B. Zheng, Lowest-order weak Galerkin finite element methods for linear elasticity on rectangular and brick meshes, J. Sci. Comput. 78 (3) (2019) 1917–1941.