Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 415 (2023) 116229
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Deep Ritz method with adaptive quadrature for linear elasticity

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

[a] School of Mechanical Engineering, Purdue University, 585 Purdue Mall, West Lafayette, IN 7907-2088, United States of America
[b] Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907-2067, United States of America

Received 21 March 2023; received in revised form 27 June 2023; accepted 29 June 2023
Available online xxxx

Abstract

In this paper, we study the deep Ritz method for solving the linear elasticity equation from a numerical analysis perspective. A modified Ritz formulation using the $H^{1/2}(\Gamma_D)$ norm is introduced and analyzed for linear elasticity equation in order to deal with the (essential) Dirichlet boundary condition. We show that the resulting deep Ritz method provides the best approximation among the set of deep neural network (DNN) functions with respect to the "energy" norm. Furthermore, we demonstrate that the total error of the deep Ritz simulation is bounded by the sum of the network approximation error and the numerical integration error, disregarding the algebraic error. To effectively control the numerical integration error, we propose an adaptive quadrature-based numerical integration technique with a residual-based local error indicator. This approach enables efficient approximation of the modified energy functional. Through numerical experiments involving smooth and singular problems, as well as problems with stress concentration, we validate the effectiveness and efficiency of the proposed deep Ritz method with adaptive quadrature.

Keywords: Deep neural network; PDE; Linear elasticity; Deep Ritz; Adaptive quadrature

1. Introduction

In the past decade, Deep Neural Networks (DNNs) have achieved remarkable success in computer vision, natural language processing, and many other machine learning (ML) applications. More recently, scientific machining learning methods based on DNN have also been applied to modeling and solving complex engineering systems. These methods can be broadly divided into three categories based on how the DNN is used: (i) *purely data-driven approaches*, which use supervised ML to create a surrogate model that regresses a physical model from a given simulation dataset or experimental observations [1–6]; (ii) *physics-enhanced approaches*, which use semi-supervised ML to implement physical laws as a regularizing term to solve a target regression problem with limited observation data [7–9] ; and (iii) *physics-driven approaches*, which impose physics into the loss functional and training process and rely on unsupervised ML to directly solve various types of PDEs [10–17]. Mathematically speaking, purely data-driven approaches can be thought of as methods for finding the best curve fit through the data points, using techniques such as least-squares regression. Physics-driven approaches, on the other hand, typically involve solving PDEs using numerical optimization methods. Numerous studies have demonstrated that DNNs possess highly

* 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.cma.2023.116229
0045-7825/

desirable approximation properties that are not available in commonly used finite element methods. One such advantage is that a DNN can adapt its physical partition to match the underlying function being approximated [18–20]. In other words, a DNN model can dynamically adjust its function representation, akin to a moving mesh method, but without the necessity of a geometric mesh.

Due to the fact that the set of DNN functions does not form a linear space, existing physics-driven methods are typically based on either the energy minimization formulation [13,21,22] or various least-squares formulations [11,12,14,15]. Energy minimization formulations are particularly suitable for problems that possess a natural minimization principle, such as many problems encountered in solid mechanics. On the other hand, the effectiveness of least-squares formulations depends on the specific principle employed. For example, the deep Galerkin method (DGM) [11] and the physics-informed neural networks (PINN) [14] rely on the discrete l^2 norm least-squares principle, which applies to differential equations, and boundary and initial conditions. However, these methods suffer from suboptimal approximation and are limited to problems with $\boldsymbol{H}^2$-smooth solutions, thereby excluding problems with geometric or interface singularities. To overcome these limitations, well-designed least-squares methods for PDEs can be employed, as described in books such as [23] and papers such as [24,25] for linear elasticity. Recently, the DNN-based first-order system least-squares (FOSLS) formulation has been utilized for the second-order elliptic PDEs [15,26].

In this paper, we aim to investigate the deep Ritz method for solving the linear elasticity equation from a numerical analysis perspective, as well as to introduce a deep Ritz method with *adaptive* quadrature. Originally proposed in [13] for scalar elliptic PDEs, the deep Ritz method employs DNNs as the class of approximating functions and is based on the Ritz formulation of the underlying PDE. However, unlike finite element approximations, the deep Ritz method encounters two fundamental challenges arising from the characteristics of DNN functions. The first challenge pertains to enforcing the Dirichlet boundary condition effectively. The second challenge involves devising a numerical integration scheme that plays a crucial role in ensuring the accuracy and robustness of the DNN approximation to the solution of the underlying problem.

Regarding the first issue, there are mainly two approaches. One is the Nitsche method [27,28] that converts the Dirichlet boundary condition into the Robin boundary condition by *penalizing* the Neumann boundary condition, and the penalization constant has to be sufficiently small. This approach is equivalent to penalize the energy functional by the L^2 norm of the residual of the Dirichlet boundary condition [13]. The other one is to enforce the Dirichlet boundary condition exactly through an auxiliary continuous function vanishing on the Dirichlet boundary [29]. In this paper, we explore the penalization method with $H^{1/2}$ norm that guarantees stability of the perturbed problem. Specifically, the standard minimization formulation is modified by adding the $\boldsymbol{H}^{1/2}$ norm of the residual of the Dirichlet boundary condition to the energy functional. By using a fundamental inequality of Korn's type in $\boldsymbol{H}^1(\Omega)$ (see, e.g., [30]), the modified minimization problem is shown to have a unique solution and the solution continuously depends on the data (see Proposition 1). Based on the modified minimization problem, the deep Ritz method is defined by minimizing the modified energy functional over the set of DNN functions and the deep Ritz approximation with the exact integration and differentiation is proved to be the best approximation in the modified energy norm (see Theorem 1).

An evaluation of the modified energy functional includes integration over both the domain and the boundary, as well as differentiation at integration points. Naturally, the integration is approximated by quadrature-based methods, since the dimension of the linear elasticity problem is at most four (including space and time). Under a reasonable assumption on numerical integration, we demonstrate that the total error in the energy norm is bounded by the approximation error of the set of DNNs plus the numerical integration error (see Theorem 2). It is important to note that solving the minimization problems under the deep Ritz formulation using DNNs gives rise to a high-dimensional and non-convex optimization problem. However, the algebraic error introduced during the solving/training process falls beyond the scope of this discussion.

In the finite element setting, it is trivial to control the numerical integration error because the unknown finite element approximation is a piece-wise polynomial on a *fixed* triangulation of the computational domain. However, controlling the numerical integration error for the deep Ritz method is difficult since the unknown DNN approximation is a composition function with several layers. Moreover, the accuracy of the DNN approximation is determined by the quality of numerical integration mesh on which the solution can be approximated well by a selected quadrature rule [31]. To overcome this obstacle, we propose an adaptive quadrature method that refines integration mesh and quadrature points adaptively. A modified residual-based local error indicator is used for

marking subdomains to be refined. The effectiveness and efficiency of the deep Ritz method with adaptive quadrature is studied for several benchmarks.

The rest of the paper is organized as follows. Section 2 introduces the modified Ritz formulation for linear elasticity problems and establishes its well-posedness (existence, uniqueness, and stability). Section 3 describes the discretization of the problem with DNN functions and shows the upper bounds of the approximation error. In Section 4, we propose the adaptive quadrature method and introduce the corresponding local error indicator. The last two sections present the numerical results and conclude the paper.

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 $\|\cdot\|$.

2. Modified Ritz formulation of linear elasticity

Let Ω be a bounded domain in $\mathbb{R}^d$ ($d = 2$ or 3) with Lipschitz boundary $\partial\Omega = \Gamma_D \cup \Gamma_N$, where Γ_D and Γ_N are disjoint. Let $\boldsymbol{n}$ be the outward unit vector normal to the boundary. Denote by $\boldsymbol{u}$ and $\boldsymbol{\sigma}$ the displacement field and the stress tensor. Consider the following linear elasticity problem

$$\begin{cases} -\nabla\cdot\boldsymbol{\sigma} &= \boldsymbol{f}, & \text{in } \Omega,\\ \boldsymbol{\sigma}(\boldsymbol{u}) &= 2\mu\boldsymbol{\epsilon}(\boldsymbol{u}) + \lambda\nabla\cdot\boldsymbol{u}\,\boldsymbol{\delta}_{d\times d} & \text{in } \Omega \end{cases} \tag{1}$$

with boundary conditions $\boldsymbol{u}\big|_{\Gamma_D} = \boldsymbol{g}_D$ and $(\boldsymbol{\sigma}\boldsymbol{n})\big|_{\Gamma_N} = \boldsymbol{g}_N$, where $\nabla\cdot$ is the divergence operator; $\boldsymbol{\epsilon}(\boldsymbol{u}) = \frac{1}{2}\big(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^T\big)$ is the strain tensor; the $\boldsymbol{f}$, $\boldsymbol{g}_D$, and $\boldsymbol{g}_N$ are given vector-valued functions defined on Ω, Γ_D, and Γ_N, representing body force, boundary displacement and boundary traction force condition respectively; $\boldsymbol{\delta}_{d\times d}$ is the d-dimensional identity matrix; μ and λ are the material Lamé constants.

Since it is difficult for directly constraining neural network functions to satisfy boundary conditions (see [13]), as in [15], we enforce the Dirichlet (essential) boundary condition weakly using a half norm through the energy functional. The modified Ritz formulation of problem (1) is to find $\boldsymbol{u} \in \boldsymbol{H}^1(\Omega)^d$ such that

$$J(\boldsymbol{u}) = \min_{\boldsymbol{v}\in\boldsymbol{H}^1(\Omega)^d} J(\boldsymbol{v}), \tag{2}$$

where the modified energy functional is given by

$$J(\boldsymbol{v}) = \frac{1}{2}\left\{\int_\Omega \left(2\mu\,|\boldsymbol{\varepsilon}(\boldsymbol{v})|^2 + \lambda\,|\nabla\cdot\boldsymbol{v}|^2\right)d\boldsymbol{x} + \gamma\|\boldsymbol{v}-\boldsymbol{g}_D\|^2_{1/2,\Gamma_D}\right\} - (\boldsymbol{f},\,\boldsymbol{v}) - (\boldsymbol{g}_N,\boldsymbol{v})_{0,\Gamma_N}. \tag{3}$$

Here, $\gamma = \mu\gamma_D$ is a penalization constant scaled by μ, and $\|\cdot\|_{1/2,\Gamma_D}$ denotes the Sobolev–Slobodeckij norm given by

$$\|\boldsymbol{v}\|_{1/2,\Gamma_D} = \left(\int_{\Gamma_D}|\boldsymbol{v}|^2 dS + \int_{\Gamma_D}\int_{\Gamma_D}\frac{|\boldsymbol{v}(\boldsymbol{x})-\boldsymbol{v}(\boldsymbol{x}')|^2}{d(\boldsymbol{x},\boldsymbol{x}')^d}dS(\boldsymbol{x})dS(\boldsymbol{x}')\right)^{1/2}, \tag{4}$$

where $d(\boldsymbol{x},\boldsymbol{x}')$ is the geodesic distance between $\boldsymbol{x}$ and $\boldsymbol{x}'$ in Γ_D. Let

$$\begin{aligned} a(\boldsymbol{u},\boldsymbol{v}) &= 2\mu(\boldsymbol{\epsilon}(\boldsymbol{u}),\boldsymbol{\epsilon}(\boldsymbol{v})) + \lambda(\nabla\cdot\boldsymbol{u},\nabla\cdot\boldsymbol{v}) + \gamma\,(\boldsymbol{u},\boldsymbol{v})_{\frac{1}{2},\Gamma_D}\\ \text{and}\quad f(\boldsymbol{v}) &= (\boldsymbol{f},\boldsymbol{v}) + (\boldsymbol{g}_N,\boldsymbol{v})_{0,\Gamma_N} + \gamma\,(\boldsymbol{g}_D,\boldsymbol{v})_{\frac{1}{2},\Gamma_D}, \end{aligned}$$

then the variational form of (2) is to finding $\boldsymbol{u} \in \boldsymbol{H}^1(\Omega)^d$ such that

$$a(\boldsymbol{u},\boldsymbol{v}) = f(\boldsymbol{v}), \quad \forall\ \boldsymbol{v} \in \boldsymbol{H}^1(\Omega)^d. \tag{5}$$

Proposition 1. *Problem (2) has a unique solution $\boldsymbol{u} \in \boldsymbol{H}^1(\Omega)^d$. Moreover, the solution $\boldsymbol{u}$ satisfies the following a priori estimate:*

$$\|\boldsymbol{u}\|_{1,\Omega} \le C\left(\|\boldsymbol{f}\|_{-1,\Omega} + \|\boldsymbol{g}_D\|_{1/2,\Gamma_D} + \|\boldsymbol{g}_N\|_{-1/2,\Gamma_N}\right). \tag{6}$$

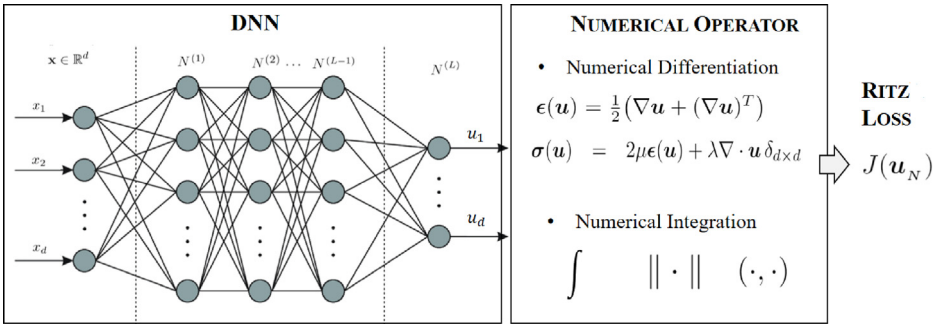


Fig. 1. Deep Ritz NN architecture. A fully connected L-layer network is employed to generate the map from an arbitrary spatial point $\boldsymbol{x}$ in Ω to its displacement $\boldsymbol{u}(\boldsymbol{x})$, numerical operators are used to approximate the gradient, divergence and integral in the discrete energy functional $J(\boldsymbol{u}_N)$ as the Ritz loss.

With the following Korn inequality (see, e.g., [30]),

$$\|\boldsymbol{v}\|_{1,\Omega} \leq C\left(\|\boldsymbol{\varepsilon}(\boldsymbol{v})\|_{0,\Omega} + \|\boldsymbol{v}\|_{1/2,\Gamma_D}\right).$$

the proof of the proposition is standard.

3. Deep Ritz neural network method

In this section, we describe the deep Ritz method which includes a standard fully connected DNN as the class of approximating functions and the discrete energy functional $J_{\mathcal{T}}(\boldsymbol{v})$ as an approximation of the energy functional $J(\boldsymbol{v})$ by numerical integration and differentiation. The structure of the deep Ritz NN is illustrated in Fig. 1.

3.1. Deep neural network

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

$$N^{(j)}(\mathbf{x}^{(j-1)}) = \tau(\boldsymbol{\omega}^{(j)}\mathbf{x}^{(j-1)} - \mathbf{b}^{(j)}) \quad \text{for} \quad \mathbf{x}^{(j-1)} \in \mathbb{R}^{n_{j-1}}, \tag{7}$$

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 bias to be determined; $\mathbf{x}^{(0)} = \mathbf{x}$; and $\tau(t)$ is a non-linear activation function. There are many activation functions such as ReLU, ReLU^p, sigmoids, sinusoidal, and hyperbolic tangent. (see, e.g., [32]).

Let $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^{d \times n_{l-1}}$ and $\boldsymbol{b}^{(l)} \in \mathbb{R}^d$ be the output weights and bias. Then a l-layer neural network generates the following set of vector fields in $\mathbb{R}^d$

$$\mathcal{M}_N(l) = \left\{\boldsymbol{\omega}^l\left(N^{(l-1)} \circ \cdots \circ N^{(1)}(\mathbf{x})\right) - \boldsymbol{b}^l : \boldsymbol{\omega}^{(j)} \in \mathbb{R}^{n_j \times n_{j-1}}, \mathbf{b}^{(j)} \in \mathbb{R}^{n_j} \text{ for all } j\right\}, \tag{8}$$

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

This class of approximating functions is rich enough to accurately approximate any continuous function defined on a compact set $\Omega \in \mathbb{R}^d$ (see [33,34] 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 viewpoint of polynomial spline functions with free knots [35]. The set $\mathcal{M}_N(l)$ 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. Discretization

Note that neural network functions in $\mathcal{M}_N(l)$ are nonlinear with respect to the weights $\{\boldsymbol{\omega}^{(j)}\}_{j=1}^{l-1}$ and the bias $\{\mathbf{b}^{(j)}\}_{j=1}^{l-1}$. This implies that it is difficult to discretize (1) by the conventional approach based on the corresponding

variational formulation (5). Instead, discretization using NNs should be based on an optimization formulation. In this paper, we employ the Ritz formulation (2) that minimizes the energy functional.

To approximate the solution of (1) using a neural network, the deep Ritz method minimizes the energy functional over the set $\mathcal{M}_N(l)$, i.e., finds $\boldsymbol{u}_N \in \mathcal{M}_N(l) \subset \boldsymbol{H}^1(\Omega)^d$ such that

$$J(\boldsymbol{u}_N) = \min_{\boldsymbol{v}\in\mathcal{M}_N(l)} J(\boldsymbol{v}). \tag{9}$$

Since $\mathcal{M}_n^1(l)$ is not a linear space, problem (9) may have many solutions.

Theorem 1. *Let $\boldsymbol{u} \in \boldsymbol{H}^1(\Omega)^d$ be the solution of problem (5), and let $\boldsymbol{u}_N \in \mathcal{M}_N(l)$ be a solution of (9). Then we have*

$$\|\boldsymbol{u} - \boldsymbol{u}_N\|_a = \inf_{\boldsymbol{v}\in\mathcal{M}_N(l)} \|\boldsymbol{u} - \boldsymbol{v}\|_a, \tag{10}$$

where $\|\boldsymbol{v}\|_a := \sqrt{a(\boldsymbol{v},\boldsymbol{v})}$ is the energy norm.

Proof. Since $\boldsymbol{u}_N \in \mathcal{M}_N(l) \subset \boldsymbol{H}^1(\Omega)^d$, (10) is a direct consequence of

$$\|\boldsymbol{u} - \boldsymbol{u}_N\|_a^2 = 2\left(J(\boldsymbol{u}_N) - J(\boldsymbol{u})\right) \le 2\left(J(\boldsymbol{v}) - J(\boldsymbol{u})\right) = \|\boldsymbol{u} - \boldsymbol{v}\|_a^2$$

for any $\boldsymbol{v} \in \mathcal{M}_N(l)$. □

Theorem 1 indicates that $\boldsymbol{u}_N$ is the best approximation with respect to the energy norm $\|\cdot\|_a$, within the neural network functions class $\mathcal{M}_N(l)$.

3.3. Numerical integration

Evaluation of the energy functional requires integration and differentiation which are often computed numerically in practice. This section discusses numerical integration schemes suitable for neural network functions. To this end, let us 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

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

When using a smooth activation function such as sigmoid, ReLU^p etc., neural network functions belong to at least $C^1(\Omega)$, i.e.,

$$\mathcal{M}_N(l) \subset C^1(\Omega). \tag{11}$$

In this case, as in [15] numerical integration may use the composite quadrature rule, such as composite mid-point, trapezoidal, Simpson, Gaussian, etc., defined on an *artificial* partition $\mathcal{T}$ of the domain Ω. Here the artificial partition refers its independence of the underlying geometry of the approximating function and it allows us to partition the domain with few restrictions. With a chosen numerical integration, differentiation is evaluated at quadrature points and can be done by either numerical differentiation with relatively small step size or automatic differentiation.

For simplicity of presentation, we describe the composite mid-point rule for interior and boundary integration in (3) and (4) in two dimensions. Let $\boldsymbol{x}_T$ and $\mathbf{x}_E$ be the centroids of $T \in \mathcal{T}$ and $E \in \mathcal{E}_S$ for $S = D$ and N. For any

integrand $v(\boldsymbol{x})$, the composite "mid-point" quadrature rule over the domain Ω and the boundary Γ_S are given by

$$\int_{\Omega} v(\boldsymbol{x})\, d\mathbf{x} \approx \sum_{T\in\mathcal{T}} v(\boldsymbol{x}_T)\,|T| \quad \text{and} \quad \int_{\Gamma_S} v(\boldsymbol{x})\, ds \approx \sum_{E\in\mathcal{E}_S} v(\boldsymbol{x}_E)\,|E|,$$

where $|T|$ and $|E|$ are the respective volume of element $T \in \mathcal{T}$ and area of boundary element $E \in \mathcal{E}_S$.

Let the Dirichlet boundary Γ_D be the union of several disjoint Γ_D^k for $k = 1, \ldots, I$. On each Γ_D^k, to approximate the Sobolev–Slobodeckij semi-norm in two dimensions, denote by $\boldsymbol{v}[\boldsymbol{x}; \boldsymbol{x}']$ the integrand in the second term of (4) as the divided difference of a vector-valued function $\boldsymbol{v}(\boldsymbol{x})$ along Γ_D^k, where each component of $\boldsymbol{v}[\boldsymbol{x}; \boldsymbol{x}']$ is given by

$$v_i[\boldsymbol{x}; \boldsymbol{x}'] = \begin{cases} \dfrac{dv_i(\boldsymbol{x})}{d\Gamma_D^k}, & \boldsymbol{x}' = \boldsymbol{x}, \\ \dfrac{v_i(\boldsymbol{x}) - v_i(\boldsymbol{x}')}{d(\boldsymbol{x}, \boldsymbol{x}')}, & \boldsymbol{x}' \neq \boldsymbol{x}. \end{cases}$$

Then we have

$$\int_{\Gamma_D^k}\int_{\Gamma_D^k} \frac{|\boldsymbol{v}(\boldsymbol{x}) - \boldsymbol{v}(\boldsymbol{x}')|^2}{d(\boldsymbol{x}, \boldsymbol{x}')^2} ds(\boldsymbol{x}) ds(\boldsymbol{x}') \approx \sum_{E\in\mathcal{E}_D^k}\sum_{E'\in\mathcal{E}_D^k} |\boldsymbol{v}[\boldsymbol{x}_E; \boldsymbol{x}_{E'}]|^2\,|E|\,|E'|. \tag{12}$$

Define the discrete bilinear and linear forms, $a_{\mathcal{T}}(\cdot,\cdot)$ and $f_{\mathcal{T}}(\cdot)$, by

$$\begin{aligned} a_{\mathcal{T}}(\boldsymbol{u},\boldsymbol{v}) = {} & 2\mu \sum_{T\in\mathcal{T}} \big(\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v})\big)(\boldsymbol{x}_T) + \lambda \sum_{T\in\mathcal{T}} \big(\nabla\cdot\boldsymbol{u}\nabla\cdot\boldsymbol{v}\big)(\boldsymbol{x}_T) \\ & + \mu\gamma_D \left\{ \sum_{E\in\mathcal{E}_D} (\boldsymbol{u}\cdot\boldsymbol{v})(\boldsymbol{x}_E) + \sum_{k=1}^{I}\sum_{E\in\mathcal{E}_D^k}\sum_{E'\in\mathcal{E}_D^k} \boldsymbol{u}[\boldsymbol{x}_E; \boldsymbol{x}_{E'}]\cdot\boldsymbol{v}[\boldsymbol{x}_E; \boldsymbol{x}_{E'}]|E\|E'| \right\} \\ f_{\mathcal{T}}(\boldsymbol{v}) = {} & \sum_{T\in\mathcal{T}} (\boldsymbol{f}\cdot\boldsymbol{v})(\boldsymbol{x}_T) + \sum_{E\in\mathcal{E}_N} (\boldsymbol{g}_N\cdot\boldsymbol{v})(\boldsymbol{x}_E) \\ & + \mu\gamma_D \left\{ \sum_{E\in\mathcal{E}_D} (\boldsymbol{g}_D\cdot\boldsymbol{v})(\boldsymbol{x}_E) + \sum_{k=1}^{I}\sum_{E\in\mathcal{E}_D^k}\sum_{E'\in\mathcal{E}_D^k} \boldsymbol{g}_D[\boldsymbol{x}_E; \boldsymbol{x}_{E'}]\cdot\boldsymbol{v}[\boldsymbol{x}_E; \boldsymbol{x}_{E'}]|E\|E'| \right\}. \end{aligned}$$

Define the discrete counterpart of the energy function $J(\cdot)$ by

$$J_{\mathcal{T}}(\boldsymbol{v}) = \frac{1}{2} a_{\mathcal{T}}(\boldsymbol{v}, \boldsymbol{v}) - f_{\mathcal{T}}(\boldsymbol{v}).$$

Then the deep Ritz approximation to the solution of (1) is to seek $\boldsymbol{u}_{\mathcal{T}} \in \mathcal{M}_N(l)$ such that

$$J_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}) = \min_{\boldsymbol{v}\in\mathcal{M}_N(l)} J_{\mathcal{T}}(\boldsymbol{v}). \tag{13}$$

To understand the effect of numerical integration, we extend the first Strang lemma for the Galerkin approximation over a subspace (see, e.g., [36]) to the Ritz approximation over a subset.

Theorem 2. *Assume that there exists a positive constant β independent of $\mathcal{M}_N(l)$ such that*

$$\beta\,\|\boldsymbol{v}\|_a^2 \le a_{\mathcal{T}}(\boldsymbol{v}, \boldsymbol{v}), \quad \forall\ \boldsymbol{v} \in \mathcal{M}_N(l). \tag{14}$$

Let $\boldsymbol{u}$ be the solution of (2) and $\boldsymbol{u}_{\mathcal{T}}$ a solution of (13). Then we have

$$\|\boldsymbol{u} - \boldsymbol{u}_{\mathcal{T}}\|_a \le \frac{2}{\beta} \sup_{\boldsymbol{w}\in\mathcal{M}_{2N}(l)} \frac{|f(\boldsymbol{w}) - f_{\mathcal{T}}(\boldsymbol{w})|}{\|\boldsymbol{w}\|_a} + \frac{2+\beta}{\beta} \inf_{\boldsymbol{v}\in\mathcal{M}_N(l)} \left\{ \|\boldsymbol{u} - \boldsymbol{v}\|_a + \sup_{\boldsymbol{w}\in\mathcal{M}_{2N}(l)} \frac{|a(\boldsymbol{v}, \boldsymbol{w}) - a_{\mathcal{T}}(\boldsymbol{v}, \boldsymbol{w})|}{\|\boldsymbol{w}\|_a} \right\}. \tag{15}$$

Proof. For any $\boldsymbol{v} \in \mathcal{M}_N(l) \subset \boldsymbol{H}^1(\Omega)^d$, we have

$$J_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}) \le J_{\mathcal{T}}(\boldsymbol{v}) \quad \text{and} \quad a(\boldsymbol{u}, \boldsymbol{u}_{\mathcal{T}} - \boldsymbol{v}) = f(\boldsymbol{u}_{\mathcal{T}} - \boldsymbol{v}).$$

It follows from assumption (14) and the definition of $J_{\mathcal{T}}(\cdot)$ that

$$\begin{aligned}
&\frac{\beta}{2}\|\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v}\|_a^2 \le \frac{1}{2}a_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})\\
&= J_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}})-J_{\mathcal{T}}(\boldsymbol{v})+f_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})-a_{\mathcal{T}}(\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v}) \le f_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})-a_{\mathcal{T}}(\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})\\
&= \Big(f_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})-f(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})\Big)+\Big(a(\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})-a_{\mathcal{T}}(\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})\Big)+a(\boldsymbol{u}-\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v}).
\end{aligned}$$

which, together with the triangle and Cauchy–Schwarz inequalities, implies

$$\begin{aligned}
\frac{\beta}{2}\|\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v}\|_a &\le \frac{|f_{\mathcal{T}}(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})-f(\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})|}{\|\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v}\|_a}+\frac{|a(\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})-a_{\mathcal{T}}(\boldsymbol{v},\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v})|}{\|\boldsymbol{u}_{\mathcal{T}}-\boldsymbol{v}\|_a}+\|\boldsymbol{u}-\boldsymbol{v}\|_a\\
&\le \sup_{\boldsymbol{w}\in\mathcal{M}_{2N}(l)}\frac{|f_{\mathcal{T}}(\boldsymbol{w})-f(\boldsymbol{w})|}{\|\boldsymbol{w}\|_a}+\sup_{\boldsymbol{w}\in\mathcal{M}_{2N}(l)}\frac{|a(\boldsymbol{v},\boldsymbol{w})-a_{\mathcal{T}}(\boldsymbol{v},\boldsymbol{w})|}{\|\boldsymbol{w}\|_a}+\|\boldsymbol{u}-\boldsymbol{v}\|_a.
\end{aligned}$$

Combining the above inequality with the triangle inequality

$$\|\boldsymbol{u}-\boldsymbol{u}_{\mathcal{T}}\|_a \le \|\boldsymbol{u}-\boldsymbol{v}\|_a+\|\boldsymbol{v}-\boldsymbol{u}_{\mathcal{T}}\|_a$$

and taking the infimum over all $\boldsymbol{v}\in\mathcal{M}_N(l)$ yield (15). This completes the proof of the theorem. □

This theorem 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 error.

4. Adaptive quadrature method

As indicated in Theorem 2, numerical integration plays an important role in NN-based numerical methods. How to control the numerical integration error for the deep Ritz method is a non-trivial matter because the unknown DNN approximation is a composition function with several layers. To overcome this obstacle, in this section we propose adaptive Ritz method that refines integration mesh adaptively.

Numerical integration defined in the previous section is based on an artificial partition $\mathcal{T}$ of the domain Ω. This partition may not capture well the variation of the underlying solution and hence (13) would possibly lead to an inaccurate approximation.

One may choose a uniform partition with sufficient fine mesh; however, it is cost inefficient. In this section, we describe an adaptive quadrature algorithm on numerical integration introduced in [18] under the assumption that the neural network is large enough to approximate the solution accurately.

A key ingredient for an adaptive quadrature scheme is an efficient local error indicator. In this paper, we use a modified residual-based indicator. To this end, let $\mathcal{T}$ be the current integration mesh and $\boldsymbol{u}_{\mathcal{T}}$ be a solution of (13). For each $T\in\mathcal{T}$, we define the following local error indicator for each $T\in\mathcal{T}$,

$$\eta_T(\boldsymbol{u}_{\mathcal{T}}) = |T|^{1/d}\|\nabla\cdot\boldsymbol{\sigma}_{\mathcal{T}}+\boldsymbol{f}\|_{0,T}, \tag{16}$$

where $\boldsymbol{\sigma}_{\mathcal{T}}$ is the numerical stress given by

$$\boldsymbol{\sigma}_{\mathcal{T}} = 2\mu\boldsymbol{\epsilon}(\boldsymbol{u}_{\mathcal{T}})+\lambda\nabla\cdot\boldsymbol{u}_{\mathcal{T}}\,\boldsymbol{\delta}_{d\times d}.$$

Note that the typical jump terms in finite element vanish due to the fact that $\mathcal{M}_N(l)$ is in $C^1(\Omega)$. The L^2 norm of the residual $\nabla\cdot\boldsymbol{\sigma}_{\mathcal{T}}+\boldsymbol{f}$ on each $T\in\mathcal{T}$ may be approximated as follows,

$$\|\nabla\cdot\boldsymbol{\sigma}_{\mathcal{T}}+\boldsymbol{f}\|_{0,T} \approx |T|^{-\frac{1}{2}}\left|\int_T(\nabla\cdot\boldsymbol{\sigma}_{\mathcal{T}}+\boldsymbol{f})\,d\boldsymbol{x}\right| = |T|^{-\frac{1}{2}}\left|\int_{\partial T}\boldsymbol{\sigma}_{\mathcal{T}}\mathbf{n}\,dS+\int_T\boldsymbol{f}\,d\boldsymbol{x}\right|$$

which implies

$$\eta_T(\boldsymbol{u}_{\mathcal{T}}) \approx |T|^{\frac{2-d}{2d}}\left|\int_{\partial T}\boldsymbol{\sigma}_{\mathcal{T}}\mathbf{n}\,dS+\int_T\boldsymbol{f}\,d\boldsymbol{x}\right|. \tag{17}$$

With this local error indicator, we then define a subset $\hat{\mathcal{T}}$ of $\mathcal{T}$ by using either the following bulk marking strategy: finding a minimal subset $\hat{\mathcal{T}}$ of $\mathcal{T}$ such that

$$\sum_{T\in\hat{\mathcal{T}}}\eta_T^2 \ge \gamma_1\sum_{T\in\mathcal{T}}\eta_T^2 \quad \text{for} \quad \gamma_1\in(0,\,1) \tag{18}$$

or the average marking strategy:

$$\hat{\mathcal{T}} = \left\{ T \in \mathcal{T} : \eta_T \geq \frac{\gamma_2}{\#\mathcal{T}} \sum_{T\in\mathcal{T}} \eta_T \right\}, \quad \text{for } \gamma_2 \in \left(0, \frac{\max\{\eta_T\}}{\#\mathcal{T}} \sum_{T\in\mathcal{T}} \eta_T\right) \tag{19}$$

where $\#\mathcal{T}$ is the number of subdomains of $\mathcal{T}$. For each marked domain $T \in \hat{\mathcal{T}}$, we subdivide T into 2^d subdomains uniformly and denote the refinement partition by $\mathcal{T}'$.

Let $\boldsymbol{u}_{\mathcal{T}}$ be an approximation of (13) based on an initial partition $\mathcal{T}$, which in general, is an uniform partition of the domain, the adaptive quadrature refinement method is summarized as follows,

Algorithm 3.1 Adaptive Quadrature Refinement (AQR) with a fixed NN.

(1) for each $T \in \mathcal{T}$, compute the local error indicator $\eta_{\mathcal{T}}$;
(2) mark $\mathcal{T}$ by the marking strategy and refine marked element to obtain a new partition $\mathcal{T}'$;
(3) numerically solve the minimization problem in (13) on $\mathcal{T}'$;
(4) if $\eta(\boldsymbol{u}_{\mathcal{T}'}) \leq \gamma\eta(\boldsymbol{u}_{\mathcal{T}})$, go to Step (1) with $\mathcal{T} = \mathcal{T}'$; otherwise, output $\mathcal{T}$.

As indicated in [18], the stopping criterion used in Algorithm 3.1 is based on whether or not the quadrature refinement on numerical integration improves approximation accuracy. When the refinement does not improve accuracy much, the adaptive quadrature stops and outputs the current integration mesh.

5. Numerical studies

In this section, we present our numerical results for several 2D problems. In all experiments, the DNN structure is represented as $d_{in} - n_1 - n_2 \cdots n_{l-1} - d_{out}$ for a l-layer network with n_1, n_2 and n_{l-1} neurons in the respective first, second, and $(l-1)$th hidden layers, and $d_{in} = d_{out}$ represent the network input and output dimensions. The minimization of the deep Ritz NN loss function (13) is solved using the Adam version of gradient descent [37]. All differential operators are calculated using numerical differentiation (ND) with the step size $\Delta x = h/4$, where $h = \min\{|T|^{1/d}\}$ is the smallest partition size of an adaptive integration mesh. All experiments use sigmoid ($\sigma(t) = \frac{1}{1+e^{-t}}$) as the activation function. For adaptive quadrature, the average marking strategy is reported due to its computational simplicity.

5.1. Test case I: smooth stress distribution

Consider problem (1) defined on $\Omega = (-1, 1) \times (-1, 1)$ with the body force

$$\boldsymbol{f} = 2\mu\big(3 - x^2 - 2y^2 - 2xy, 3 - 2x^2 - y^2 - 2xy\big)^T + 2\lambda\big(1 - y^2 - 2xy, 1 - x^2 - 2xy\big)^T,$$

and the traction

$$\boldsymbol{g}_N = 2(y^2 - 1)\big(2\mu + \lambda, \mu\big)^T$$

on $\Gamma_N = \{(1, y) : y \in (-1, 1)\}$, with the clamped boundary condition on $\Gamma_D = \partial\Omega \setminus \Gamma_N$. The exact solution of the test problem has the form

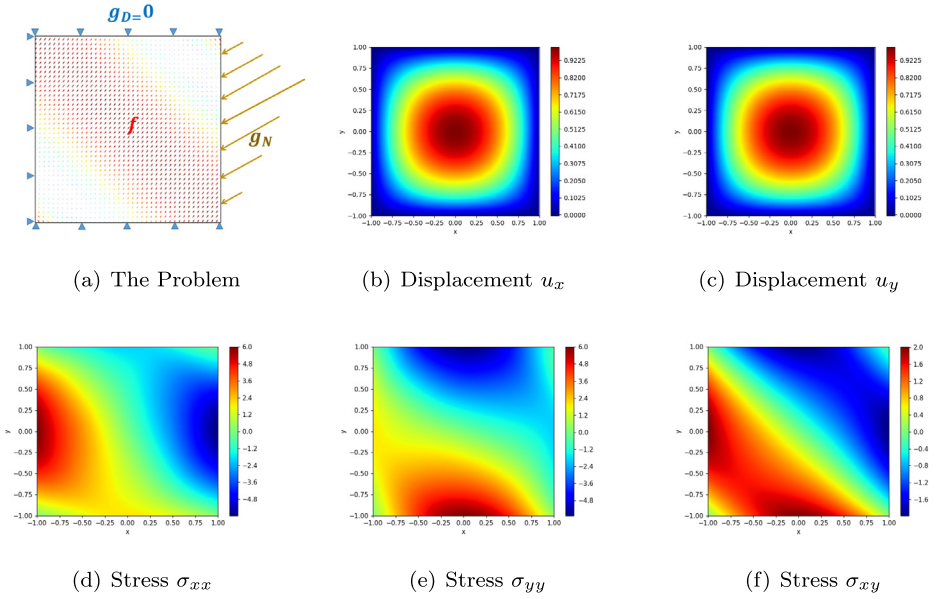
$$\boldsymbol{u}(x, y) = (1 - x^2)(1 - y^2)\big(1, 1\big)^T.$$

Set the material property $\mu = 1$, and $\lambda = 1$, we first test three-layer DNNs of varying number of neurons and different numerical quadrature resolutions. Uniformly distributed quadrature points of size 100×100, 200×200 and 400×400 are used to evaluate the effect of numerical integration combined with three network structures. Table 1 list the numerical results. As shown in the table, With a small DNN of 106 parameters (2-8-8-2) and 100×100 uniformly distributed quadrature points, deep Ritz can approximate the problem at a relative energy norm of 0.1658. Increasing the resolution of quadrature reduces the numerical integration error and therefore improves the approximation accuracy. E.g. by increasing the number of quadrature points to 200×200, and further to 400×400, the accuracy of the numerical solution is continuously improved, as measured by the relative energy norm, or the $L2$

Table 1
Numerical results of deep Ritz method for test case I using fixed quadrature.

DNN (No. para.)	$\#\mathcal{T}$	$\frac{\|\boldsymbol{u}-\boldsymbol{u}_N\|_a}{\|\boldsymbol{u}\|_a}$	$\frac{\|\sigma-\sigma_N\|}{\|\sigma\|}$	$\frac{\|\boldsymbol{u}-\boldsymbol{u}_N\|}{\|\boldsymbol{u}\|}$
2-8-8-2 (106)	100 × 100	16.58%	16.35%	6.42%
	200 × 200	10.81%	10.61%	3.91%
	400 × 400	6.09%	6.00%	2.15%
2-16-16-2 (338)	100 × 100	11.94%	11.77%	4.17%
	200 × 200	8.55%	8.50%	2.90%
	400 × 400	5.94%	5.93%	1.96%
2-32-32-2 (1186)	100 × 100	3.3%	3.67%	1.42%
	200 × 200	2.76%	2.73%	1.13%
	400 × 400	2.19%	2.17%	0.93%

*Training details: $\gamma_D = 100$;
Total number of iterations for each row: 200,000;
Learning rate initial is 0.01 and it decays 90% every 50000 iterations.



(a) The Problem (b) Displacement u_x (c) Displacement u_y

(d) Stress σ_{xx} (e) Stress σ_{yy} (f) Stress σ_{xy}

Fig. 2. Test case I problem ($\mu = 1$, and $\lambda = 1$) and numerical results using deep Ritz (2-32-32-2) with fixed 400 × 400 quadrature points.

norm of $\boldsymbol{u}$ and σ. On the other hand, larger DNNs with more parameters have better expressive power and thus can approximate the solution with better accuracy, this is a property from Theorem 2 and is confirmed experimentally as the results given in Table 1. A three-layer DNN (2-32-32-2) combined with finer quadrature points 400 × 400 has a better accuracy compared with smaller network or coarser quadrature points. The result is also depicted graphically in Fig. 2.

Second, we test the adaptive quadrature refinement (AQR) method using the DNN structure 2-32-32-2. Starting from the initial 100 × 100 uniformly distributed quadrature points, and using the residual-based local error indicator (16) and the average marking strategy (19) with $\gamma_2 = 1$, the AQR process stops at run 4 and reaches a relative energy norm 0.0174. As shown in Table 2, with 45,757 quadrature points after three-run AQR, the adaptive deep Ritz can achieve a similar relative energy norm as the fixed uniform quadrature method using 160,000 quadrature points. Adding more runs of adaptive quadrature process will improve the approximation accuracy, but it converges to a limit when the network approximation error becomes dominant. For example, by increasing the number of quadrature points from 45,757 to 99,262, error measured by the three norms are not improving significantly. To

Table 2
Numerical results of deep Ritz for experiment 1 using AQR.

AQR run	$\#\mathcal{T}$	$\frac{\|\boldsymbol{u}-\boldsymbol{u}_N\|_a}{\|\boldsymbol{u}\|_a}$	$\frac{\|\sigma-\sigma_N\|}{\|\sigma\|}$	$\frac{\|\boldsymbol{u}-\boldsymbol{u}_N\|}{\|\boldsymbol{u}\|}$	$\frac{\sum_{T\in\mathcal{T}}\eta_T}{\#\mathcal{T}}$
1	10,000	3.73%	3.67%	1.42%	0.0004
2	21,145	2.81%	2.78%	1.17%	0.0002
3	45,757	1.98%	1.97%	0.89%	5e−5
4	99,262	1.74%	1.73%	0.82%	2e−5

*Training details: $\gamma_D = 100$;
Run 2–4 are trained using weight transferred from the previous run;
Each trained 100,000 iterations using fixed learning rate 0.001.

further improve the accuracy, one may need to enlarge the DNN size to obtain a better network approximation power.

5.2. Test case II: L-shape plate with corner singularity

The second test is a common benchmark problem with a re-entrant corner forming a typical point singularity [38]. The problem is posed on an $L-$shaped domain $\Omega = (-1,1)^2 \setminus ([0,1]\times[-1,0])$ with a body force $\boldsymbol{f} = \boldsymbol{0}$. The known analytical solution is,

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

where r, θ are the polar coordinates and

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

Here the critical exponent $\alpha \approx 0.544483737$ is the solution of the equation $\alpha\sin(2\omega)+\sin(a\omega\alpha) = 0$ with $\omega = 3\pi/4$ and $C_1 = -(\cos(\alpha+1)\omega)/(\cos(\alpha-1)\omega)$, $C_2 = (2(\lambda+2\mu))/(\lambda+\mu)$ [39]. A bronze material with Young's modulus $E = 100000$ and Poisson's ratio $\nu = 0.3$ is tested with the Neumann BCs prescribed on $\Gamma_N = \{(1,y) : y \in (0,1)\}$ and Dirichlet BCs on $\Gamma_D = \partial\Omega \setminus \Gamma_N$.

Due to a stress singularity at the corner point $(0,0)$, the direct LS physics-driven methods do not apply to this type of problems. Using the deep Ritz method, we test the performance of a four layer DNN structure (2-48-48-48-2, 4898 parameters) with fixed uniform quadrature and adaptive quadrature refinement (AQR). As shown in Table 3, with AQR generated non-uniformly distributed quadrature points, adaptive deep Ritz approximates the solution using less data (quadrature points) compared with the uniform fixed quadrature method. During the iterative process, the marked regions with larger residuals who need refinement are depicted in Figs. 3(g)–3(i) and the generated non-uniform quadrature points are plotted in Fig. 3(f). The stress singularity is well captured by the local error indicators and correspondingly, more quadrature points are distributed over the re-entrant corner region. The final numerical solutions obtained are plotted in Figs. 3(a)–3(d). This experiment shows the validity of the proposed local error estimator for the adaptive quadrature scheme.

5.3. Test case III: A quadratic membrane under tension

The third test problem is given by a quadratic membrane of elastic isotropic material with a circular hole in the center. Traction forces act on the upper and lower edges of the strip, body forces are ignored. Because of the symmetry of the problem, it suffices to compute only a fourth of the total geometry. The computational domain is then given by

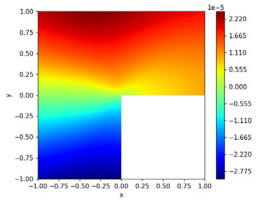
$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^2 : 0 < x < 10, 0 < y < 10, x^2 + y^2 > 1\}.$$

The boundary condition on the top edge of the computation domain ($\Gamma_1 : \{y = 10, 0 < x < 10\}$) are set to $\boldsymbol{\sigma n} = (0, 4.5)^T$, the boundary condition on the bottom ($\Gamma_2 : \{y = 0, 1 < x < 10\}$) are set to $(\sigma_{xx}, \sigma_{xy})\cdot\boldsymbol{n} = 0$, $u_y = 0$

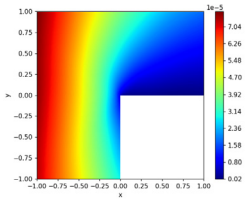
Table 3
Numerical results of adaptive deep Ritz for test case II using a DNN structure 2-48-48-48-2.

Quadrature method	$\#\mathcal{T}$		$\frac{\|\boldsymbol{u}-\boldsymbol{u}_N\|_a}{\|\boldsymbol{u}\|_a}$	$\frac{\|\sigma-\sigma_N\|}{\|\sigma\|}$	$\frac{\|\boldsymbol{u}-\boldsymbol{u}_N\|}{\|\boldsymbol{u}\|}$
uniform fixed	120,000		10.99%	10.20%	1.83%
non-uniform AQR	run 1	30,000	23.38%	21.58%	3.31%
	run 2	42,897	14.44%	13.45%	1.78%
	run 3	60,306	11.32%	10.48%	1.69%
	run 4	90,237	10.71%	9.86%	1.68%

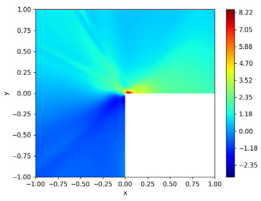
*Training details: $\gamma_D = 1$;
For the uniform quadrature and the AQR run 1: trained with 200,000 iterations with learning rate starts from 0.01 and decays 90% every 50,000 iterations;
For non-uniform AQR run 2–4, trained with 100,000 iterations using fixed learning rate 1e−5.



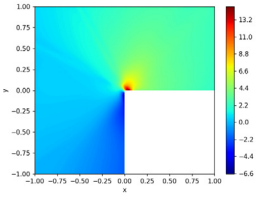
(a) Displacement u_x



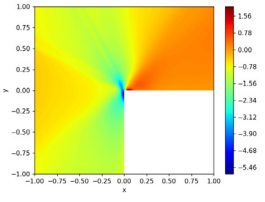
(b) Displacement u_y



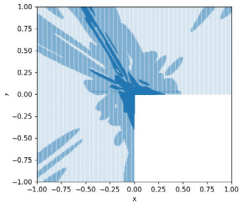
(c) Stress σ_{xx}



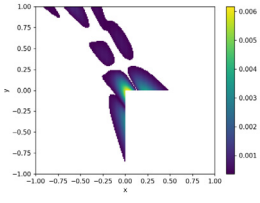
(d) Stress σ_{yy}



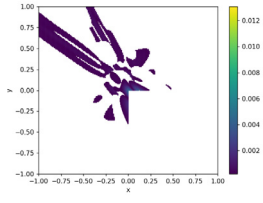
(e) Stress σ_{xy}



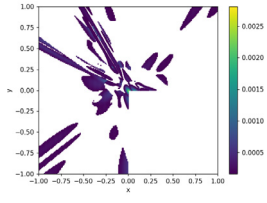
(f) Quadrature points (run 4)



(g) Marked elements at run 2



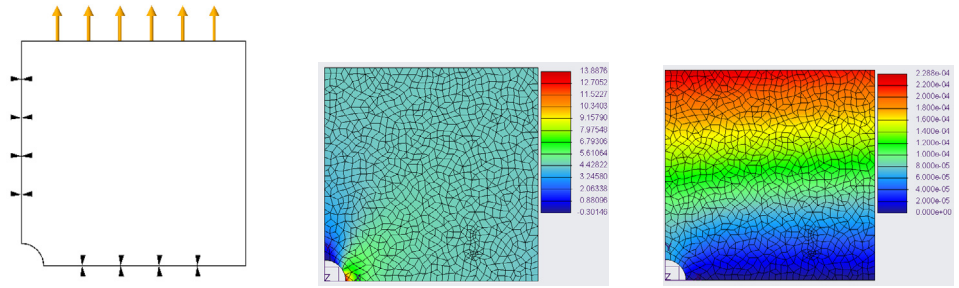
(h) Marked elements at run 3



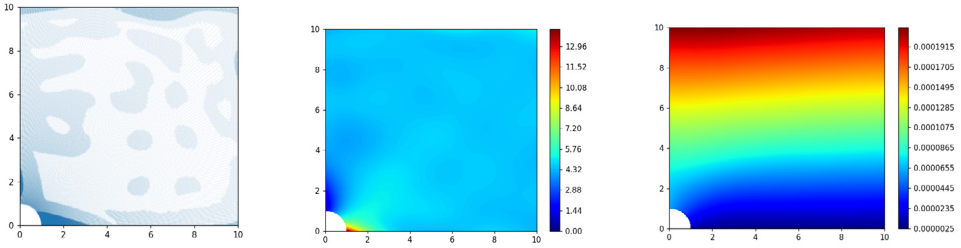
(i) Marked elements at run 4

Fig. 3. Test case II numerical solution using adaptive deep Ritz (2-48-48-48-2). (a–d) component-wise numerical solution $\boldsymbol{u}$ and σ; (f) final quadrature points obtained through AQR; (g–i) marked element during the four-run AQR process.

(symmetry condition), and finally, the boundary condition on the left ($\Gamma_3 : \{x = 0, 1 < y < 10\}$) are given by $(\sigma_{yx}, \sigma_{yy}) \cdot \boldsymbol{n} = 0$, and $u_x = 0$ (symmetry condition). The material parameters are $E = 206900$ for Young's modulus and $\nu = 0.29$ for Poisson's ratio. The challenge of this test is the stress concentration located around point (0,1) due to the presence of the small hole. Since there is no analytic solution, a reference solutions is obtained using finite element analysis (FEA) with an adaptive mesh refinement (adaptive p-element refined with highest



(a) The test problem (b) FEA $\boldsymbol{\sigma}_{yy}$ (c) FEA $\boldsymbol{u}_y$



(d) Last run AQR points (e) Ritz $\boldsymbol{\sigma}_{yy}$ (f) Ritz $\boldsymbol{u}_y$

Fig. 4. Test case III Numerical solution using adaptive FEA p-element and deep Ritz with adaptive quadrature. (Rtiz solution is obtained with a DNN structure: 2-64-64-64-2 of 8578 DoF. FEA solution is obtained using p-element with 14786 DoF).

Table 4
Using Deep Ritz with adaptive quadrature to solve test case III.

Iteration	No. of quad	max $\|\boldsymbol{u}\|$	max $\boldsymbol{\sigma}_{yy}$	$\frac{\sum_{T\in\mathcal{T}}\eta_T}{\#\mathcal{T}}$
1	28,392	2.1327e−4	7.9155	9.5942e−3
2	40,878	2.1566e−4	10.0728	6.7179e−4
3	60,434	2.1623e−4	13.2632	4.4317e−4
4	88,574	2.2672e−4	13.8912	2.9309e−4

*Training details: $\gamma_D = 1$;
Run 1 is trained with 200,000 iterations, with learning rate starts from 0.01 and decays 90% every 50,000 iterations;
Run 2–4: 50,000 iterations each with fixed learning rate 1e−5.

polynomial order of 7). The reference solution is given in Figs. 4(b) and 4(c), where we use two key measures: $\max \boldsymbol{\sigma}_{yy} = 13.8876$, an $\max \|\boldsymbol{u}\| = 2.288\text{e}{-4}$ as references (a similar reference solution is also provided in [24] using adaptive h-elements).

We tested the adaptive deep Ritz with a four-layer structure (2-64-64-64-2, 8578 parameters). Initially, 28,392 uniform quadrature points using polar coordinates are used for getting the first iteration that captures well for the displacement field $\boldsymbol{u}$. However the stress concentration factor is not accurate with uniformly distributed quadrature points. After four iterations of AQR using the average marking strategy ($\gamma_2 = 1.5$), a total of 88,574 quadrature points are generated adaptively as shown in Fig. 4(d) and with this set of non-uniform quadrature points, the method is capable of obtaining a similar stress concentration factor compared with the adaptive FEA solution, see the numerical results depicted in Figs. 2(a)–2(c) and Table 4. As expected, more quadrature points are distributed adaptively near the small hole region during the AQR processes such that the stress concentration can be simulated accurately.

Furthermore, we explored the potential of using transfer learning to solve a family of stress problems. To this end, we conducted a sensitivity study by varying the center hole from $r = 1$ to $r = 5$, taking a step size of 0.5.

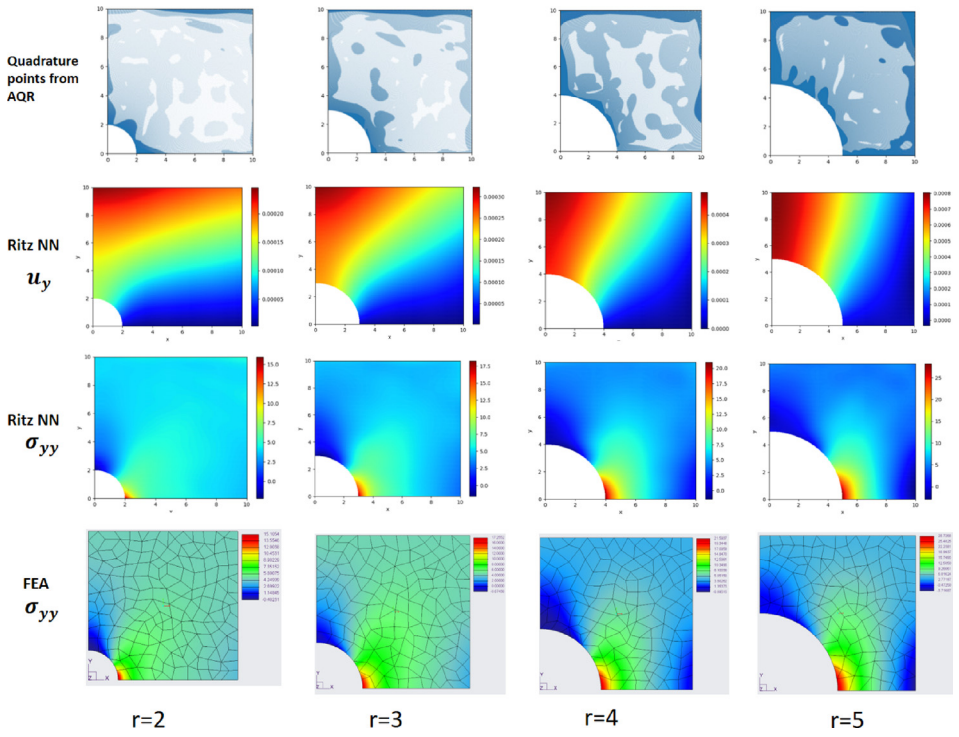


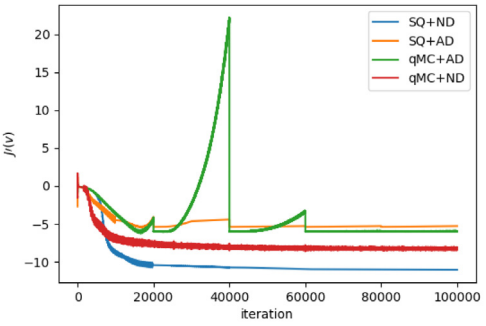
Fig. 5. Sensitivity study of the test case III with varying hole radii. First row shows the generated adaptive quadrature points at the listed hole radius steps; second and third rows show the numerical solution using adaptive deep Ritz, and the last row give the reference adaptive FEA solutions.

Using the trained DNN model in the previous experiment (with an initial hole size of $r = 1$) as the starting point, we stepped through the various hole sizes by training a DNN using weight transferring. Our assumption was that the DNN trained from the previous step forms a good initial for the next parametric step, and therefore the adaptive deep Ritz would converge faster if applied to a family of similar problems. In our test, we verified this assumption and it took significantly fewer number of iterations (in this example, 50,000 iterations and two-run AQR for each hole size step) to converge to the results. This is compared to the total 350,000 iterations used for the result obtained in Table 4, demonstrating that transfer learning saves significant time. Our results at each step also align with the numerical solutions evaluated through FEA adaptive p-refinement method. Fig. 5 lists the displacement component $\boldsymbol{u}_y$ and the associate stress tensor component $\boldsymbol{\sigma}_{yy}$ obtained from adaptive deep Ritz transfer learning and adaptive FEA.

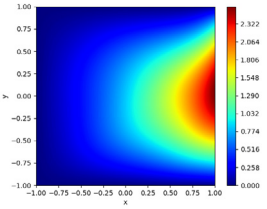
5.4. Discussion

Numerical Differentiation (ND) or Automatic Differentiation (AD). For each quadrature point $\boldsymbol{x}_T$, the evaluations of $\boldsymbol{\varepsilon}\big(\boldsymbol{v}(\boldsymbol{x}_T)\big)$ and $\nabla\cdot\boldsymbol{v}(\boldsymbol{x}_T)$ in the energy functional are based on the first order partial derivatives that may be calculated through either automatic differentiation (AD) or numerical differentiation (ND). AD is a common choice for some physics-driven methods such as the PINN and its variants. When a DNN function has the first order derivatives at all sampling points, AD eliminates numerical error caused by ND. However, when using AD, there are some difficulties in training PINN as noticed in [40]; moreover, a discrete differential operator combining AD and ND was proposed. In our experiments, pathologies in training deep Ritz with AD were also observed. Nevertheless, our experiments show numerically that ND with quadrature-based numerical integration is robust in training/solving the deep Ritz.

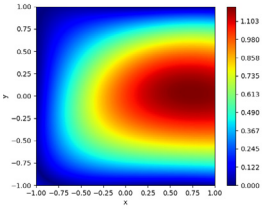
Here, we use the test case I as an example and list the results of using combinations of different integration and differentiation methods: (1) standard quadrature-based integration plus AD (SQ+AD), (2) standard quadrature-based



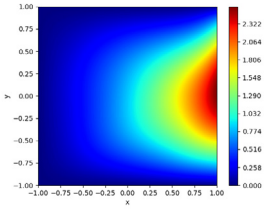
(a) Energy Functional of different methods



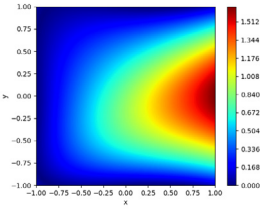
(b) $\|\boldsymbol{u}\|$ (SQ+ND)



(c) $\|\boldsymbol{u}\|$ (SQ+AD)



(d) $\|\boldsymbol{u}\|$ (qMC+ND)



(e) $\|\boldsymbol{u}\|$ (qMC+AD)

Fig. 6. Comparison study for test case I using numerical differentiation v.s. automatic differentiation, and standard quadrature-based integration v.s. quasi-Monte Carlo-based integration. All results are trained using deep Ritz DNN structure (2-32-32-2).

integration plus ND (SQ+ND), (3) quasi-Monte Carlo-based integration plus AD (qMC+AD), and (4) quasi-Monte Carlo-based integration plus ND (qMC+ND). As shown in Fig. 6(a), only SQ+ND converges to the real potential energy of the tested problem $J^{\star}(\boldsymbol{u}) \approx -11.2$ within 100,000 iterations, the other three methods are suspiciously trapped in local minima and the obtained solutions are non-physical as shown in Figs. 6(c)–6(e).

Non-convex optimization The exceptional approximation power of neural networks allows them to effectively represent a wide range of solutions, including those with irregular geometries, discontinuities, or singularities that can pose challenges for traditional finite element methods. However, this type of approximating functions also introduces a computationally demanding optimization problem. NN-based algorithms for solving PDEs typically involve a high-dimensional and non-convex optimization for which the first order stochastic gradient descent-based methods are the most widely employed solvers. Currently, NN-based methods are not competitive with well-established mesh-based methods due to the considerable computational cost of the algebraic solvers, even though NN-based methods may require fewer degrees of freedom than their mesh-based counterparts. Developing fast solvers is an open and challenging problem and requires lots of efforts from numerical analysts.

6. Conclusion

In this paper, linear elasticity problems are formulated under the Ritz framework and are discretized using DNN functions. To enforce the essential boundary condition, the energy functional is modified with an extra penalization term using $H^{1/2}$ norm. It is shown that within the function class, the minimization of the modified energy functional yields the best approximation with respect to the modified energy norm. To calculate the modified energy functional accurately and efficiently, adaptive quadrature refinement equipped with a local residual-based error indicator was proposed and tested, and its effectiveness and efficiency in improving numerical simulation was demonstrated.

There are still numerous unresolved issues that require further research. In this study, we make the assumption that the DNN is sufficiently large to approximate the solution. However, selecting an appropriate network structure for different problems and establishing a suitable initial DNN model are still open questions. While we conducted a basic sensitivity analysis to demonstrate the potential of DNNs for parametric PDEs, exploring efficient methods

to extend the transfer learning strategy to general design space exploration and even solving topology optimization problems requires further investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported in part by the National Science Foundation, United States under grant DMS-2110571 and the Future of Work at the Human Technology Frontier (FW-HTF) 1839971. We also acknowledge the Feddersen Distinguished Professorship Funds. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agency. We thank the support of DARPA project on symbiotic design and Stanford Research International (SRI) for partial support of the project.

References

[1] L. Liang, M. Liu, C. Martin, W. Sun, A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis, J. R. Soc. Interface 15 (138) (2018) 20170844.
[2] W. Gao, X. Lu, Y. Peng, L. Wu, A deep learning approach replacing the finite difference method for in situ stress prediction, IEEE Access 8 (2020) 44063–44074.
[3] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081.
[4] P. Vurtur Badarinath, M. Chierichetti, F. Davoudi Kakhki, A machine learning approach as a surrogate for a finite element analysis: Status of research and application to one dimensional systems, Sensors 21 (5) (2021).
[5] V. Iakovlev, M. Heinonen, H. Lähdesmäki, Learning continuous-time PDEs from sparse data with graph neural networks, in: International Conference on Learning Representations, 2021.
[6] Z. Li, N.B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2021.
[7] M. Raissi, G.E. Karniadakis, Hidden physics models: Machine learning of nonlinear partial differential equations, J. Comput. Phys. (2017).
[8] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019).
[9] Y. Khoo, L. Ying, SwitchNet: A neural network model for forward and inverse scattering problems, SIAM J. Sci. Comput. 41 (5) (2019) A3182–A3201.
[10] W. E., J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) (2017) 349–380.
[11] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1139–1364.
[12] J. Berg, K. Nystrom, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[13] 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.
[14] 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.
[15] 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.
[16] Z. Cai, J. Chen, M. Liu, Least-squares ReLU neural network (LSNN) method for linear advection-reaction equation, J. Comput. Phys. 443 (2021) 110514.
[17] 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.
[18] M. Liu, Z. Cai, J. Chen, Adaptive two-layer ReLU neural network: I. best least-squares approximation, Comput. Math. Appl. 113 (2022) 34–44.
[19] Z. Cai, J. Chen, M. Liu, Least-squares ReLU neural network (LSNN) method for scalar nonlinear hyperbolic conservation law, Appl. Numer. Math. 174 (2022) 163–176.

[20] 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.
[21] J. Xu, The finite neuron method and convergence analysis, Commun. Comput. Phys. 28 (2020) 1707–1745.
[22] M. Liu, Z. Cai, Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic PDEs, Comput. Math. Appl. 113 (2022) 103–116.
[23] P.B. Bochev, M.D. Gunzburger, Least-Squares Finite Element Methods, Vol. 166, Applied Mathematics Sciences, Springer, 2009.
[24] Z. Cai, G. Starke, Least-squares methods for linear elasticity, SIAM J. Numer. Anal. 42 (2) (2004) 826–842.
[25] F. Bertrand, Z. Cai, E.-Y. Park, Least-squares methods for elasticity and Stokes equations with weakly imposed symmetry, Comput. Methods Appl. Math 19 (3) (2019) 415–430.
[26] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741.
[27] J. Nitsche, Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, Abh. Math. Sem. Univ. 36 (1) (1971) 9–15.
[28] Y. Liao, P. Ming, Deep nitsche method: Deep Ritz method with essential boundary conditions, Commun. Comput. Phys. 29 (5) (2021) 1365–1384.
[29] C. Uriarte, D. Pardo, I. Muga, J. Muñoz-Matute, A deep double Ritz method (D2RM) for solving partial differential equations using neural networks, Comput. Methods Appl. Mech. Engrg. 405 (2023) 115892.
[30] S.C. Brenner, Korn's inequalities for piecewise H1 vector fields, Math. Comp. 73 (2003) 1067–1087.
[31] J.A. Rivera, J.M. Taylor, Á.J. Omella, D. Pardo, On quadrature rules for solving partial differential equations using neural networks, Comput. Methods Appl. Mech. Engrg. 393 (2022) 114710.
[32] A. Pinkus, Approximation theory of the MLP model in nueral networks, Acta Numer. 15 (1999) 143–195.
[33] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Systems 2 (1989) 303–314.
[34] K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators, Neural Netw. 2 (1989) 359–366.
[35] L. Schumaker, Spline Functions: Basic Theory, John Wiley, New York, 1981.
[36] P.G. Ciarlet, The Finite Element Method for Elliptic Problems, Society for Industrial and Applied Mathematics, 1978.
[37] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: International Conference on Representation Learning, San Diego, 2015.
[38] 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.
[39] J. Alberty, C. Carstensen, S.A. Funken, R. Klose, Matlab implementation of the finite element method in elasticity, Computing 69 (3) (2002) 239–263.
[40] P.-H. Chiu, J.C. Wong, C. Ooi, M.H. Dao, Y.-S. Ong, CAN-PINN: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method, Comput. Methods Appl. Mech. Engrg. 395 (2022) 114909.