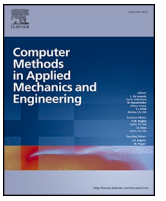
Computer Methods in Applied Mechanics and Engineering 428 (2024) 117070

Contents lists available at ScienceDirect

Computer Methods in Applied Mechanics and Engineering

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



N-adaptive ritz method: A neural network enriched partition of unity for boundary value problems

Jonghyuk Baek [a,b], Yanran Wang [a], Jiun-Shyan Chen [a,*]

[a] Department of Structural Engineering, University of California, San Diego, La Jolla, CA 92093, USA
[b] Coreform, LLC, Orem, UT 84097, USA

ARTICLE INFO

Keywords:
Ritz method
Partition of unity
Neural network enrichment
Energy minimization
Transfer-learning
Adaptivity

ABSTRACT

Conventional finite element methods are known to be tedious in adaptive refinements due to their conformal regularity requirements. Further, the enrichment functions for adaptive refinements are often not readily available in general applications. This work introduces a novel neural network-enriched Partition of Unity (NN-PU) approach for solving boundary value problems via artificial neural networks with a potential energy-based loss function minimization. The flexibility and adaptivity of the NN function space are utilized to capture complex solution patterns that the conventional Galerkin methods fail to capture. The NN enrichment is constructed by combining pre-trained feature-encoded NN blocks with an additional untrained NN block. The pre-trained NN blocks learn specific local features during the offline stage, enabling efficient enrichment of the approximation space during the online stage through the Ritz-type energy minimization. The NN enrichment is introduced under the Partition of Unity (PU) framework, ensuring convergence of the proposed method. The proposed NN-PU approximation and feature-encoded transfer learning form an adaptive approximation framework, termed the neural-refinement (n-refinement), for solving boundary value problems. Demonstrated by solving various elasticity problems, the proposed method offers accurate solutions while notably reducing the computational cost compared to the conventional adaptive refinement in the mesh-based methods.

1. Introduction

The adaptability and robust representation capabilities of neural networks (NNs) [1,2] have propelled advancements in computational mechanics, contributing notably to various aspects such as data-driven constitutive modeling [3-8], data-driven computational mechanics [9-15], multiscale modeling [16-20], and damage and fracture modeling [21,22]. Leveraging their inherent adaptivity, NNs serve as effective function approximators for solving general partial differential equations (PDEs). Their ability to construct flexible function spaces makes NNs an appealing alternative to the mesh-based method, such as the generalized finite element method [23] and the global-local enrichment [24], particularly for problems that involve localized features.

As data accessibility grows and platforms like Theano [25], TensorFlow [26], MXNet [27], and Keras [28] offer features like high-performance computing and automatic differentiation [29], neural networks emerge as a new approach to solve mathematical models. Jacobs et al. [30] proposed the mixture of experts models for solving complex regression and nonlinear classification problems

* Corresponding author.
E-mail address: jsc137@ucsd.edu (J.-S. Chen).

https://doi.org/10.1016/j.cma.2024.117070
Received 16 January 2024; Received in revised form 17 April 2024; Accepted 16 May 2024
Available online 14 June 2024

with nonstationary or piecewise continuous data. Construction of Partition of Unity functions by neural networks for solving PDEs has also been proposed [31,32]. In these approaches, polynomial spaces with adjustable coefficients are linked to each partition, resulting in an approximation similar to hp-elements. Raissi et al. [33] introduced physics-informed neural networks (PINNs), using prior PDE knowledge to estimate physical systems with sparse data via residual-based minimization based on the collocation framework. Drawing from the concept of PINNs, Haghighat and Juanes [34] created SciANN, a Python wrapper tailored for scientific computing, successfully capturing localization behaviors in perfectly plastic materials. Sirignano and Spiliopoulos [35] combined Galerkin methods with deep learning, leveraging deep neural networks for function approximation and minimizing residual-based objective functions at random collocation points. However, a major challenge of using NNs for PDE solved via residual-based loss functions with strong form collocation is the extensive computational cost, requiring a large number of sampling points across the problem domain for adequate solution resolution. An alternative method is formulated with variational problems, where the corresponding functional acts as an energy descriptor for mechanical systems, serving as a loss function to optimize NNs. Weinan and Yu [36] introduced the Deep Ritz method, employing deep neural networks as trial functions for solving PDEs by minimizing energy, adaptable to higher dimensions. However, the resulting variational problems can be non-convex, leading to challenges like local minima and saddle points, and imposing essential boundary conditions presents non-trivial complexities. Samaniego et al. [37] introduced NNs as function approximation machines and use the energy of mechanical system as a natural loss function, demonstrating that energy-based loss functions can produce superior results with significantly fewer unknowns compared to the collocation type methods relying on the residual-based loss functions. Nguyeh-Thanh et al. [38] extended previous work in [37] and applied it to solve nonlinear finite deformation hyperelasticity problems. Additionally, Nguyen-Thanh et al. [39] enhanced convergence by training models in a parametric space, avoiding classical geometry discretization. Using Gauss quadrature in the parametric domain is natural, but mapping between parametric and physical domains poses challenges due to the employment of NURBS basis functions in this approach. Saha et al. [40] introduced Hierarchical Deep Learning Neural Network (HiDeNN), which integrates inputs from physical and parametric spaces, together with physics-informed or experimentally driven deep neural networks, for solving problems in solid mechanics, multiscale analysis, and parameter extractions. Baek et al. [21] proposed a neural network-enhanced reproducing kernel particle method (NN-RKPM) for modeling localization. The RK approximation on a coarse and uniform discretization is employed to approximate the smooth part of the solutions, while the neural networks, determined by the optimization of an energy-based loss function, control parameters to automatically capture the location and oritentaion of the localized solutions. Later on, Baek and Chen [22] propsed an improved version of NN-RKPM in [21] for modeling brittle fracture, in which the NN approximation and background RK approximation are patched together with Partion of Unity to ensure convergence.

In this work, we propose a neural network-enriched Partition of Unity (NN-PU) approximation method for solving PDEs for general elasticity problems, which is a generalization of NN-RKPM previously proposed for modeling localization and fractures [21,22]. Employing a coarse background discretization, the method utilizes extrinsic NN-based enrichment functions to enhance the background approximation within the Partition of Unity framework [41,42], in a similar spirit to the hp-Clouds [43,44], the generalized finite element method [23], and the global-local enrichment [24]. To improve the efficiency and adaptivity of the NN-PU approach, a block-level NN approximation is introduced, where each NN block is designed to target a specific underlying local feature. During offline training, multiple NN enrichment basis sets are pre-trained against different "parent" problems with different local features to embed various underlying solution features into the basis functions. These feature-encoded NN basis sets are then utilized in online simulation stages through transfer learning to capture localized solution efficiently. In this approach, the NN approximation control parameters automatically construct enrichment functions driven by minimizing an energy-based loss function. The proposed NN-PU approximation and feature-encoded transfer learning form an adaptive approximation framework, termed the neural-refinement (n-refinement), departing from traditional polynomial (p-based) or mesh (h-based) adaptivity complicated by the need to impose conforming constraints.

The remainder of the paper is organized as follows. Section 2 states the problem of interest and outlines the neural network-enriched approximation within the Partition of Unity framework for solving general elasticity problems. Section 3 details NN architectures composed of block-level NN approximation, transferable NN block structure construction, and sub-block architecture forming the NN block. Error analysis of the coupled NN approximation and global NN-PU approximation is also provided in Section 3. Section 4 presents numerical implementations, encompassing offline training of feature-encoded NN enrichment basis sets and the online calculations using trained NN bases via transfer learning. The selection of NN enrichment region based on an energy error indicator is also introduced in Section 4. In Section 5, several numerical examples are presented to assess the solution accuracy, convergence, and effectiveness of the proposed NN-PU approximations. The paper concludes with a discussion and summary in Section 6.

2. Neural network-enriched partition of unity approximation

2.1. Model problem

In this study, we consider an elastostatic boundary value problem for easy demonstration of the basic concept. Here a d-dimensional elastic body defined in domain $\Omega \in \mathbb{R}^d$ is subjected to boundary conditions on the Dirichlet boundary $\partial\Omega^g$ and Neumann boundary $\partial\Omega^t$ as follows:

$$\nabla \cdot \boldsymbol{\sigma} = \boldsymbol{b} \text{ in } \Omega$$

$$\boldsymbol{u} = \boldsymbol{g} \text{ on } \partial\Omega^g$$

$$\boldsymbol{\sigma}\cdot\boldsymbol{n}=\boldsymbol{t}\ \text{on}\ \partial\Omega^t, \tag{1}$$

where $\boldsymbol{\sigma}$, $\boldsymbol{b}$, $\boldsymbol{g}$, and $\boldsymbol{t}$ denote the Cauchy stress, body force, prescribed displacements, and tractions, respectively. The Cauchy stress for a linear elastic material is defined as:

$$\boldsymbol{\sigma}=2\mu\boldsymbol{\varepsilon}+\lambda tr(\boldsymbol{\varepsilon})\boldsymbol{I}, \tag{2}$$

with the strain tensor $\boldsymbol{\varepsilon}=\frac{1}{2}(\nabla\otimes\boldsymbol{u}+\boldsymbol{u}\otimes\nabla)$, identity tensor $\boldsymbol{I}$, and Lamé's first and second parameters λ and μ. The minimization problem corresponds to Eq. (1) is: for $\boldsymbol{u}\in H^1$, $\boldsymbol{u}=\boldsymbol{g}$ on $\partial\Omega^g$,

$$\min_{\boldsymbol{u}}\Pi(\boldsymbol{u})=\min_{\boldsymbol{u}}\left[\frac{1}{2}\int_\Omega\boldsymbol{\varepsilon}:\boldsymbol{\sigma}\,d\Omega-\int_\Omega\boldsymbol{u}\cdot\boldsymbol{b}\,d\Omega-\int_{\partial\Omega^t}\boldsymbol{u}\cdot\boldsymbol{t}\,d\Gamma\right], \tag{3}$$

where $\Pi(\boldsymbol{u})$ is the potential energy. In this work, $\Pi(\boldsymbol{u})$ will later be used as the loss function to obtain the neural network-enriched solution via loss function minimization.

2.2. Partition of unity approximation with neural network enrichment

Let the domain Ω be discretized by a set of discrete points $\mathscr{S}\equiv\{1,2,\cdots,NP\}$, with NP the number of points in the domain discretization, and $V^h=\text{span}\,\{\Psi_I(\boldsymbol{x})\}_{I\in\mathscr{S}}$ be the finite dimensional approximation space with $\{\Psi_I(\boldsymbol{x})\}_{I\in\mathscr{S}}$ forming a partition of unity (PU), that is, $\sum_{I\in\mathscr{S}}\Psi_I(\boldsymbol{x})=1$. Examples of partition of unity functions are the finite element (FE) shape functions, the non-uniform rational basis spline (NURBS) functions in isogeometric analysis (IGA) [45], the moving least-squares (MLS) approximation functions in Element Free Galerkin (EFG) method [46], the reproducing kernel (RK) approximation functions in the reproducing kernel particle method (RKPM) [47,48], among others. The PU property in the approximation assures first order L_2 convergence in solving 2nd order PDEs. Higher order convergence can be achieved by enriching the PU via, for example, FE with higher order polynomials, MLS or RK with higher order basis functions as the intrinsic enrichment [49,50], or both intrinsic and extrinsic enrichments introduced under the frameworks such as the partition of unity method (PUM) [41,42] and the hp-Clouds [43,44].

We start with employing the PU functions $\{\Psi_I(\boldsymbol{x})\}_{I\in\mathscr{S}}$ with intrinsic polynomial bases as the "background" approximation, and the neural network (NN) enrichment functions $\{\mathscr{T}_I(\boldsymbol{x})\}_{I\in\mathscr{S}}$ as the extrinsic bases to the background approximation under the PUM [41,42] framework as follows:

$$u(\boldsymbol{x})\approx u^h(\boldsymbol{x})=\sum_{I\in\mathscr{S}}\Psi_I(\boldsymbol{x})(\overline{d}_I+\mathscr{T}_I(\boldsymbol{x})), \tag{4}$$

where $\overline{d}_I$ is a coefficient associated with I-th background PU shape function $\Psi_I(\boldsymbol{x})$. We consider a general neural network approximation represented by:

$$\mathscr{T}_I(\boldsymbol{x})=\sum_{K=1}^{n_I^\zeta}\zeta_{IK}(\boldsymbol{x})w_{IK}+\beta_I. \tag{5}$$

where $\zeta_{IK}(\boldsymbol{x})$, n_I^ζ, β_I are the K-th NN basis function, the number of NN basis functions, and the NN bias associated with node I, and w_{IK} is the coefficient associated with the NN basis $\zeta_{IK}(\boldsymbol{x})$. Defining a background approximation coefficient $d_I\equiv\overline{d}_I+\beta_I$ to avoid the linear dependency, we express the Neural Network-enriched Partition of Unity (NN-PU) approximation as:

$$u^h(\boldsymbol{x})=\sum_{I\in\mathscr{S}}\Psi_I(\boldsymbol{x})\left(d_I+\sum_{K=1}^{n_I^\zeta}\zeta_{IK}(\boldsymbol{x})w_{IK}\right). \tag{6}$$

In this work, we call $\zeta_{IK}(\boldsymbol{x})$ the *NN enrichment basis*, and w_{IK} the *NN enrichment coefficient*.

The approximation $u^h(\boldsymbol{x})$ can also be viewed as the superposition of a background approximation $u^{BG}(\boldsymbol{x})$ and an NN approximation $u^{NN}(\boldsymbol{x})$, and $\Psi_I(\boldsymbol{x})$ with the partition of unity property serves as a patching function for the NN enrichment bases $\{\zeta_{IK}(\boldsymbol{x})\}_{K=1}^{n_I^\zeta}$. Rewrite Eq. (6) as:

$$u^h(\boldsymbol{x})=u^{BG}+u^{NN}, \tag{7}$$

with

$$u^{BG}(\boldsymbol{x})=\sum_{I\in\mathscr{S}}\Psi_I(\boldsymbol{x})d_I,$$

$$u^{NN}(\boldsymbol{x}) = \sum_{I\in\mathscr{S}^\zeta} \Psi_I(\boldsymbol{x}) \sum_{K=1}^{n_I^\zeta} \zeta_{IK}(\boldsymbol{x}) w_{IK}, \tag{8}$$

where $\mathscr{S}^\zeta$ is the set of nodes with NN enrichment, $\mathscr{S}^\zeta \subset \mathscr{S}$, and $w_{IK} = 0 \ \forall \ I \in \mathscr{S} \setminus \mathscr{S}^\zeta$, to allow for enrichment only at a selective set of points, which will be discussed in Section 4.1.3. Let's now consider global NN-enrichment bases that are not associated with the background discretization, i.e., $\zeta_K(\boldsymbol{x}) \equiv \zeta_{IK}(\boldsymbol{x})$ and $n^\zeta \equiv n_I^\zeta$, independent of the discretization index I. Then, $u^{NN}(\boldsymbol{x})$ defined in Eq. (8) can be written as:

$$u^{NN}(\boldsymbol{x}) = \sum_{I\in\mathscr{S}^\zeta} \sum_{K=1}^{n^\zeta} \zeta_K(\boldsymbol{x}) \Psi_I(\boldsymbol{x}) w_{IK}, \tag{9}$$

where n^ζ is the number of NN enrichment bases and let $\mathbf{W}_K = \{w_{IK}\}_{I\in\mathscr{S}}$ be an NN enrichment coefficient set.

Note that the proposed approximation takes a similar form to the standard Partition of Unity and *hp*-clouds methods [41-44]. A key difference is its flexibility in dynamically adapting enrichment functions constructed via a neural network loss function minimization, without pre-defined enrichment functions, to be discussed in the next section. We term this type of refinement a neural-refinement (*n*-refinement). The proposed approach is also fundamentally different from solving PDEs by the Galerkin approximation at a stationary, which instead is solved by a Ritz-type approach through a neural-network-based potential function minimization to be discussed in the next section.

Remark 2.1. While the background shape functions $\Psi_I(\boldsymbol{x})$ can be of any type that possesses the partition of unity, this work adopts the reproducing kernel (RK) functions [47,48] for the background approximations. RK approximation offers independent choice over the order of continuity (smoothness) and the order of completeness, enabling high-order continuity with lower-order bases, particularly beneficial for approximating smooth solutions while relying on the NN approximations to capture localized behavior such as corner singularities, see [47,48,51,52] for details.

Remark 2.2. The computational architecture of the neural network approximation in Eq. (9) can be decomposed into multiple NN blocks as those designed in [21,22]. On this concept of block-level NN approximation, we can rewrite Eq. (9) with n^B blocks as follows:

$$u^{NN}(\boldsymbol{x}) = \sum_{J=1}^{n^B} u_J^B(\boldsymbol{x}) = \sum_{J=1}^{n^B} \left(\sum_{I\in\mathscr{S}_J^\zeta} \sum_{L=1}^{n_J^\zeta} \zeta_{JL}(\boldsymbol{x}) \Psi_I(\boldsymbol{x}) w_{IJL} \right), \tag{10}$$

where $u_J^B(\boldsymbol{x})$, n_J^ζ, ζ_{JL}, and $\mathscr{S}_J^\zeta$ denote the block-level NN approximation, the number of NN bases, L-th NN basis, and the NN enriched nodeset of NN block J, respectively.

Remark 2.3. Two types of enrichment to the background approximation can be considered. An "intrinsic" enrichment adds enrichment basis to the background RK shape functions, while an "extrinsic" enrichment introduces the enrichment functions outside the background RK shape functions, then are "patched" together by the RK shape functions under the PUM [41,42] framework, which yields Eq. (6). The issue of using the intrinsic enrichment is the induced discontinuities when switching basis from point to point. The intrinsic enrichment also requires solving a larger linear system in constructing the enriched RK shape functions, and it could lead to ill-conditioning in solving the local linear system if the enrichment functions are too similar to the original polynomial basis functions. The extrinsic enrichment introduced in Eq. (6), on the other hand, allows the enrichment functions to vary from point to point without discontinuity issues if they are patched together with the background RK shape functions. However, the extrinsic enrichment approach does increase the total degrees of freedom in solving PDEs, thus requiring additional computational resources, but it still offers a better efficiency than the conventional model refinement approach.

3. Transferable neural network architecture for NN approximation

3.1. Block-level NN approximation

One goal of this work is to develop feature-encoded and computationally efficient NN enrichment bases based on the neural network block architectures. The block-level NN approximation, as shown in Fig. 1(b), possesses high sparsity in the network structure compared to the densely connected counterpart shown in Fig. 1(a). Meanwhile, a block neural network has the flexibility to incorporate multiple feature-encoded NN bases targeting different localized features in the solution. This is a unique feature for constructing local enrichment for enhanced solution accuracy without the conventional mesh adaptive refinement.

3.2. Transferable neural network block architecture

Concerning the network architecture of each neural network block, it is not feasible to optimize the entire block structure during the online computation due to the associated high computational cost. Conversely, utilizing fully trained (fixed) neural network block

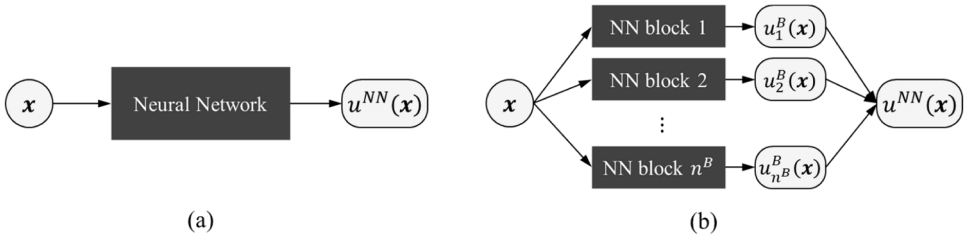


Fig. 1. Neural network architectures for NN approximation: (a) a single-block densely connected deep neural network and (b) multi-block sparsely connected neural network.

is also undesirable, as they may lack the flexibility in adjusting enrichment functions to capture unseen solution behaviors in the online computation. We aim to construct partially trained NN block structures. Herein, partial training entails training enrichment functions in an offline stage and employing them for online computations. This transfer learning-type approach aims to achieve a balance between computational accuracy and efficiency.

To this end, we define a block-level NN approximation as the composition of offline-trained NN sub-block $\mathcal{N}^{\zeta}$ and online-optimized NN sub-blocks $\mathcal{N}^{P}$ and $\mathcal{N}^{C}$ as follows (omitting the subscript J associated with the J-th NN block for brevity):

$$u^B(\boldsymbol{x}) = \mathcal{N}^C(\cdot;\mathbf{W}^C) \circ \mathcal{N}^{\zeta}(\cdot;\mathbf{W}^{\zeta}) \circ \mathcal{N}^P(\boldsymbol{x};\mathbf{W}^P), \tag{11}$$

where $\circ$ denotes the function composition operator, $\mathcal{N}^P$, $\mathcal{N}^{\zeta}$, and $\mathcal{N}^C$ denote the *Parametric sub-block, NN Basis sub-block*, and *Coefficient sub-block*, as shown in Fig. 2, and $\mathbf{W}^P$, $\mathbf{W}^{\zeta}$, and $\mathbf{W}^C$ are the weight sets associated with $\mathcal{N}^P$, $\mathcal{N}^{\zeta}$, and $\mathcal{N}^C$, respectively.

The Parametric sub-block, $\mathcal{N}^P(\cdot,\mathbf{W}^P) : \boldsymbol{x}\rightarrow\boldsymbol{y}$, is designed to generate parametric coordinates $\boldsymbol{y}$ to facilitate the utilization of pre-trained NN bases in a parametric coordinate $\boldsymbol{y}$ to represent functions with local features. Nonlinear mapping $\boldsymbol{x}\rightarrow\boldsymbol{y}$ also allows a representation of complex high-dimensional features by NN bases in a lower-dimensional manifold. The NN Basis sub-block, $\mathcal{N}^{\zeta}(\cdot;\mathbf{W}^{\zeta}) : \boldsymbol{y}\rightarrow\mathbf{Z}$, takes the parametric coordinates $\boldsymbol{y} = [y_1,\cdots,y_d]$ and generates a NN basis vector $\mathbf{Z} = [\zeta_1,\cdots,\zeta_{n^{\zeta}}]$. The weight set $\mathbf{W}^{\zeta}$ is trained in the offline stage and can be further optimized during the online calculation. This sub-block is designed to embed the trained NN bases with underlying local features that are not captured by the background approximation functions. The linear combination of the NN basis by the coefficient set $\mathbf{W}^C$ is also computed during the online stage. With the computed $\mathbf{W}^C$, the Coefficient sub-block, $\mathcal{N}^C(\cdot,\mathbf{W}^C) : \mathbf{Z}\rightarrow u^B$, adds the contribution of each NN enrichment basis to the total solution of the problem.

3.3. Offline training of a single feature by a reduced NN block architecture

In order to capture various types of solution features, an approximation may be constructed upon multiple NN blocks during online computation, each of which is partially trained against a specific solution feature in the offline training stage. In this work, a "parent" problem that presents a distinct solution feature to embed is used to train the NN Basis sub-block $\mathcal{N}^{\zeta}$ with a single NN block during the offline training stage, and the physical domain of the "parent" problem is utilized as the parametric coordinates. During the offline training, the Parametric sub-block is set to be the identity map, i.e., $\mathcal{N}^P : \boldsymbol{x}\rightarrow\boldsymbol{x}$. Then, the block-level approximation in Eq. (11) can be written as:

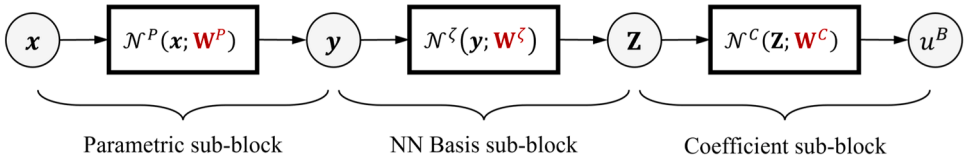


Fig. 2. Global architecture of a transferable neural network adopted for a single NN block.

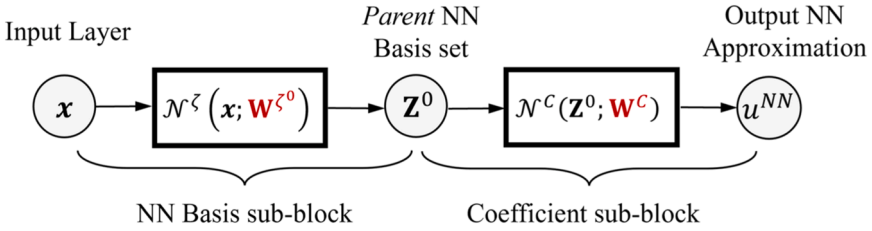


Fig. 3. Reduced NN architecture for one offline training.

$$u^{NN}(\boldsymbol{x}) = u^{B}(\boldsymbol{x}) = \mathscr{N}^{C}\left(\cdot;\mathbf{W}^{C}\right) \circ \mathscr{N}^{\zeta}\left(\boldsymbol{x};\mathbf{W}^{\zeta^0}\right) \equiv \mathscr{N}^{C}\left(\mathbf{Z}^{0};\mathbf{W}^{C}\right), \tag{12}$$

and the *parent* NN basis set $\mathbf{Z}^0 = \left[\zeta_1^0, \cdots, \zeta_{n^\zeta}^0\right]$ is defined as:

$$\mathbf{Z}^{\mathbf{0}} = \mathscr{N}^{\zeta}\left(\boldsymbol{x};\mathbf{W}^{\zeta^0}\right), \tag{13}$$

where the NN Basis sub-block $\mathscr{N}^{\zeta}$ directly takes the physical coordinates of the "parent" problem with single feature as its input and $\mathbf{W}^{\zeta^0}$ is the converged weight sets of NN Basis sub-block during the offline training. Fig. 3 depicts the reduced NN architecture for an offline training with single feature. Note that setting $\mathscr{N}^{P}$ to the identity map eliminates the operations associated with $\mathscr{N}^{P}$, which reduces the computational cost during the offline stage. Meanwhile, although the Coefficient sub-block $\mathscr{N}^{C}$ is involved, and its weight set $\mathbf{W}^{C}$ is optimized together with $\mathbf{W}^{\zeta}$ in the offline training, $\mathbf{W}^{C}$ obtained in the offline training is discarded and re-evaluated in the online simulation as the online computation could involve multiple features, and the weights for the linear combination of NN bases need to be recomputed.

3.4. Structures of NN sub-block architectures

3.4.1. Architecture of $\mathscr{N}^{\zeta}$

The NN basis sub-block is constructed with the standard deep neural network (DNN) architecture. Fig. 4 shows a general DNN architecture, represented by $\mathscr{N}^{DNN}(\mathscr{I};\boldsymbol{\omega})$ with input vector $\mathscr{I} = (i_1, \cdots, i_{n_i})$, output vector $\boldsymbol{O} = (o_1, \cdots, o_{n_o})$, and weight set $\boldsymbol{\omega}$. The weight set $\boldsymbol{\omega}$ is defined as the collection of the weights and biases of all the layers in DNN, i.e., $\boldsymbol{\omega} = \{\boldsymbol{\theta}_\ell, \beta_\ell\}_{\ell=1}^{NL}$, where $\boldsymbol{\theta}_\ell$, β_ℓ, and NL are the weight matrix of layer ℓ, the bias of layer ℓ, and the total number of hidden and output layers, respectively. Note that layer 0 is the input layer, layer NL is the output layer, and the intermediate layers are hidden layers of DNN. The intermediate output of layer ℓ, denoted as $\boldsymbol{h}_\ell$, is written as:

$$\boldsymbol{h}_\ell = \alpha_\ell(\mathscr{L}(\boldsymbol{h}_{\ell-1};\{\boldsymbol{\theta}_\ell, \beta_\ell\})), \ \ell = 1, \cdots, NL-1, \tag{14}$$

where the linear combination operator $\mathscr{L}(\boldsymbol{h};\{\boldsymbol{\theta},\beta\})$ is defined as:

$$\mathscr{L}(\boldsymbol{h};\{\boldsymbol{\theta},\beta\}) = \boldsymbol{\theta}\boldsymbol{h} + \beta, \tag{15}$$

and α_ℓ denotes the activation function applied to layer ℓ, with the exponential linear unit (ELU) activation function [53] considered in this work. Note that we have $\mathscr{I} = \boldsymbol{h}_0$ and $\boldsymbol{O} = \boldsymbol{h}_{NL}$. For the NN Basis sub-block $\mathscr{N}^{\zeta}$, a DNN is adopted with input $\mathscr{I} = \boldsymbol{y}$, with $\boldsymbol{y}$ the parametric coordinates, output $\boldsymbol{O} = \mathbf{Z}$, with $\mathbf{Z}$ the NN basis set, and weight set $\boldsymbol{\omega} = \mathbf{W}^{\zeta}$.

3.4.2. Architecture of $\mathscr{N}^{P}$

As discussed in Section 3.3, the Parametric sub-block $\mathscr{N}^{P}$ is set to be the identity map for the offline training stage. In this work, $\mathscr{N}^{P}$ is inspired by the residual neural network (ResNet) [54]:

$$\mathscr{N}^{P}(\boldsymbol{x};\mathbf{W}^{P}) = \mathscr{N}^{DNN}(\boldsymbol{x};\ \mathbf{W}^{P}) + \boldsymbol{x}, \tag{16}$$

and the network architecture is shown in Fig. 5. Note that by setting $\mathbf{W}^{P} = 0$, the identity map used for the offline training stage is recovered. Besides its identity reproducibility, ResNet is proven to avoid the gradient vanishing issues commonly encountered in deep neural networks, thereby enhancing computational efficiency [54].

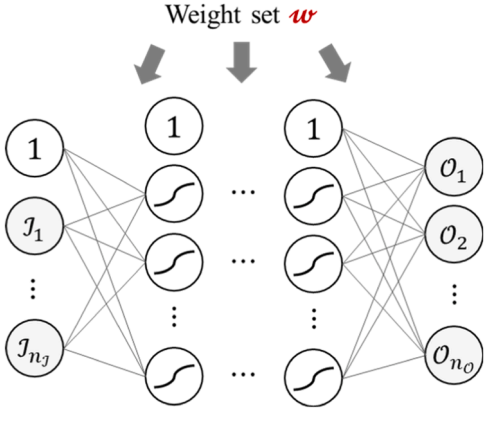


Fig. 4. A NN basis sub-block $\mathscr{N}^{\zeta}$ constructed by a densely connected neural network.

3.4.3. Architecture of $\mathscr{N}^{\mathbf{C}}$

The Coefficient sub-block $\mathscr{N}^{C}$ is defined as follows:

$$\mathscr{N}^{C}(\mathbf{Z};\mathbf{W}^{C}) = \sum_{I\in\mathscr{S}^{\zeta}} \Psi_I(\boldsymbol{x})\mathscr{F}_I(\boldsymbol{x};\mathbf{W}^{C}), \tag{17}$$

with $\mathbf{Z} = [\zeta_1, \cdots, \zeta_{n^{\zeta}}]$ and $\mathscr{F}_I(\boldsymbol{x};\mathbf{W}^{C}) = \sum_{L=1}^{n^{\zeta}} \zeta_L(\boldsymbol{x}) w_{IL}$. The sub-block architecture is shown in Fig. 6.

3.5. Convergence

Melenk and Babuška (1996) [41] provided the following global error bound of a Partition of Unity approximation u^{PU} as follows: with a generic constant C_G,

$$\| u^{PU} - u\|_{0,\Omega} \le C_G \left(\sum_I \epsilon_I^2 \right)^{1/2}, \tag{18}$$

where ϵ_I is a local error limit and bounds the error on $\Omega_I \equiv \text{supp}(\Psi_I(\boldsymbol{x}))$:

$$\| w_I - u\|_{0,\Omega_I} \le \epsilon_I, \tag{19}$$

where $w_I = w_I(\boldsymbol{x})$ is the local approximation function of a general Partition of Unity approximation, i.e., $u^{PU}(\boldsymbol{x}) = \sum_I \Psi_I(\boldsymbol{x}) w_I(\boldsymbol{x})$. Let p^{Ψ} and p^{w} be the order of complete polynomial reproduced by $\text{span}\{\Psi_I\}_{I\in\mathscr{S}}$ and $w_I(\boldsymbol{x})$, respectively. For the case that p^{w} is the leading order, i.e., $p^{w} \ge p^{\Psi}$, Duarte and Oden (1996) [44] proved the following global error estimate for smooth u:

$$\| u^{PU} - u\|_{0,\Omega} \le \mathscr{O}\left(h^{p^{w}+1}\right), \tag{20}$$

where $h = \max_I h_I$ is the maximum nodal spacing. For the case that p^{Ψ} is the leading order and $w_I(\boldsymbol{x}) = d_I$ is constant, e.g., the Reproducing Kernel approximation, Han et al. (2002) [55] and Chen et al. (2003) [56] showed the following global error estimate for smooth u:

$$\| u^{PU} - u\|_{0,\Omega} \le \mathscr{O}\left(h^{p^{\Psi}+1}\right). \tag{21}$$

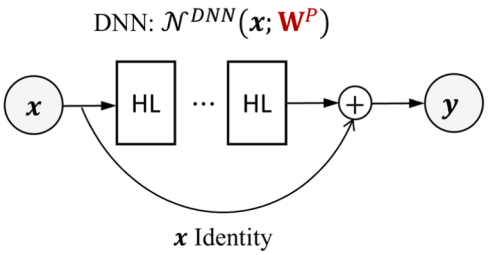


Fig. 5. Modified architecture for parametrization network (HL stands for hidden layer).

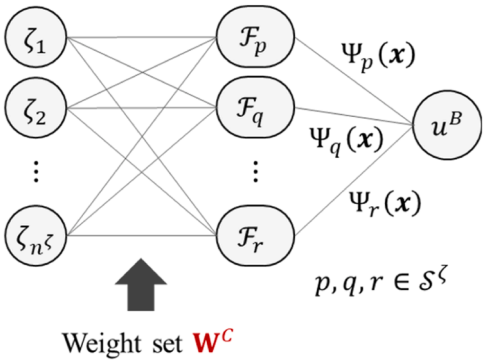


Fig. 6. Network architecture of coefficient sub-block.

3.5.1. Convergence of pure neural network approximation in partition of unity

Based on the universal approximation theorem [2] and a more recent study in [57], the local error of a neural network enrichment function $\mathscr{T}_I = \mathscr{T}(\boldsymbol{x}; \mathbf{W}_I)$ can be represented as follows: for $u \in L_2$,

$$\| \mathscr{T}_I - u\|_{0,\Omega_I} \leq C_{\Omega_I} \widehat{n}^{1/2}, \tag{22}$$

where $\widehat{n}$ is a size measure of the neural network and C_{Ω_I} is a parameter defined as

$$C_{\Omega_I} = \left(\int_{\Omega_I} B|\widehat{u}(\omega)| \, d\Omega(\omega) \right)^{1/2}, \tag{23}$$

where $\widehat{u}$ is the frequency-domain solution, ω is the frequency, and B is a constant independent of the network size measure $\widehat{n}$ and the nodal spacing of I-th background node h_I. To remove the dependency of C_{Ω_I} on the support size of Ψ_I, the physical coordinate can be normalized by defining a parametric coordinate $\boldsymbol{\xi} \equiv (\boldsymbol{x} - \boldsymbol{x}_I)/h_I$ with parametric domain $\square$, which yields:

$$C_{\Omega_I} = h_I^{d/2} \left(\int_{\square} B|\widehat{u}(\omega)| \, d\xi(\omega) \right)^{1/2} \equiv h_I^{d/2} \widehat{C}, \tag{24}$$

where d is the space dimension. With Eqs. (18), (22), and (24), we have the following approximation error of the NN approximation u^{NN}:

$$\| u^{NN} - u\|_{0,\Omega} \leq C_G \widehat{C} \left(\sum_I h_I^d \right)^{1/2} \widehat{n}^{1/2} \leq C_G C \Big((NP) h^d \Big)^{1/2} \widehat{n}^{1/2} \tag{25}$$

Note that u^{NN} defined in Eq. (8) itself is a Partition of Unity approximation. Therefore, Eq. (18) is directly applied in obtaining the global approximation error of u^{NN} in Eq. (25). With $NP \sim \mathscr{O}\left(h^{-d}\right)$, Eq. (25) becomes:

$$\| u^{NN} - u\|_{0,\Omega} \leq \mathscr{O}\left(\widehat{n}^{1/2}\right). \tag{26}$$

3.5.2. Convergence of NN-PU approximation

Consider an NN-PU approximation with u^{BG} of degree p and u^{NN} of size measure $\widehat{n}$. In order to estimate the local error bound of NN-PU, a local interpolation error from u^{BG} is first estimated, followed by the estimation of the error from u^{NN}. Note that in this work $u^{BG} = u^{RK}$. Let $L^p(x; \mathbf{C})$ be a degree-p polynomial function of x with a set of constants $\mathbf{C}$. For u^{BG} that can locally reproduce a complete polynomial of degree p, and considering the Taylor expansion of u, we have the local approximation:

$$\begin{aligned}
&\| u^{BG} + u^{NN} - u\|_{0,\Omega_I} \\
&= \| L^p(\boldsymbol{x}; \mathbf{C}_I) + \mathscr{T}(\boldsymbol{x}; \mathbf{W}_I) - u(\boldsymbol{x})\|_{0,\Omega_I} \\
&= \| \mathscr{T}(\boldsymbol{x}; \mathbf{W}_I) - \sum_{|\alpha|=p+1} \frac{(\boldsymbol{x} - \overline{\boldsymbol{x}})^\alpha}{\alpha!} u^{(\alpha)}(\overline{\boldsymbol{x}}) - \sum_{|\alpha|=p+2} \mathscr{O}((\boldsymbol{x} - \overline{\boldsymbol{x}})^\alpha)\|_{0,\Omega_I}, \text{ with } \alpha! = \alpha_1! \cdots \alpha_d! \text{ and } u^{(\alpha)} = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}},
\end{aligned} \tag{27}$$

where α is the multi-index notation defined in Appendix. Further introducing a parametric coordinate $\boldsymbol{\xi} \equiv (\boldsymbol{x} - \overline{\boldsymbol{x}})/h_I$ with parametric domain $\square$, we have:

$$\begin{aligned}
&\| \mathscr{T}(\boldsymbol{x}; \mathbf{W}_I) - \sum_{|\alpha|=p+1} \frac{(\boldsymbol{x} - \overline{\boldsymbol{x}})^\alpha}{\alpha!} u^{(\alpha)}(\overline{\boldsymbol{x}}) - \sum_{|\alpha|=p+2} \mathscr{O}((\boldsymbol{x} - \overline{\boldsymbol{x}})^\alpha)\|_{0,\Omega_I} \\
&= h_I^{d/2} \| \mathscr{T}(\boldsymbol{\xi}; \overline{\mathbf{W}}_I) - h_I^{p+1} \sum_{|\alpha|=p+1} \frac{\boldsymbol{\xi}^\alpha}{\alpha!} u^{(\alpha)}(\overline{\boldsymbol{x}}) - \sum_{|\alpha|=p+2} \mathscr{O}\left(h_I^{p+2} \boldsymbol{\xi}^\alpha\right)\|_{0,\square} \\
&= h_I^{p+1+d/2} \| \mathscr{T}(\boldsymbol{\xi}; \overline{\overline{\mathbf{W}}}_I) - \sum_{|\alpha|=p+1} \frac{\boldsymbol{\xi}^\alpha}{\alpha!} u^{(\alpha)}(\overline{\boldsymbol{x}}) - \sum_{|\alpha|=p+2} \mathscr{O}(h_I \boldsymbol{\xi}^\alpha)\|_{0,\square}.
\end{aligned} \tag{28}$$

In Eq. (28), with $\mathbf{W}_I \equiv \{\boldsymbol{\theta}_{I\ell}, \beta_{I\ell}\}_{\ell}^{NL}$ following the notations in Section 3.4.1 and with the parametrization, the modified weight set $\overline{\mathbf{W}}_I \equiv \{\overline{\boldsymbol{\theta}}_{I\ell}, \overline{\beta}_{I\ell}\}_{\ell=1}^{NL}$ is defined as $\overline{\boldsymbol{\theta}}_{I\ell} = h_I \boldsymbol{\theta}_{I\ell}$, $\overline{\beta}_{I\ell} = \beta_{I\ell} + \overline{\boldsymbol{x}} \sum_k \theta_{I\ell k}$ for $\ell = 1$ and $\{\overline{\theta}_{I\ell}, \overline{\beta}_{I\ell}\}_{\ell=2}^{NL} = \{\boldsymbol{\theta}_{I\ell}, \beta_{I\ell}\}_{\ell=2}^{NL}$. Similarly, the modified weight set $\overline{\overline{\mathbf{W}}}_I \equiv \{\overline{\overline{\boldsymbol{\theta}}}_{I\ell}, \overline{\overline{\beta}}_{I\ell}\}_{\ell=1}^{NL}$ is defined as $\overline{\overline{\boldsymbol{\theta}}}_{I\ell} = \overline{\boldsymbol{\theta}}_{I\ell}/h_I^{p+1}$, $\overline{\overline{\beta}}_{I\ell} = \overline{\beta}_{I\ell}/h_I^{p+1}$ for $\ell = NL$ and $\{\overline{\overline{\boldsymbol{\theta}}}_{I\ell}, \overline{\overline{\beta}}_{I\ell}\}_{\ell=1}^{NL-1} = \{\overline{\boldsymbol{\theta}}_{I\ell}, \overline{\beta}_{I\ell}\}_{\ell=1}^{NL-1}$.

Utilizing Eq. (22), we obtain the following local error estimate when $h_I \to 0$: for $u^{(\alpha)} \in L_2,\ \forall |\alpha| = p+1$,

$$\| u^{BG} + u^{NN} - u \|_{0,\Omega_I} \leq h_I^{p+1+d/2} \| \mathscr{T}(\boldsymbol{\xi}; \overline{\overline{\mathbf{W}}}_I) - \sum_{|\alpha|=p+1} \frac{\boldsymbol{\xi}^\alpha}{\alpha!} u^{(\alpha)}(\overline{\boldsymbol{x}}) \|_{0,\square}$$

$$\leq \widehat{C}_I h_I^{p+1+d/2} \widehat{n}^{1/2} \equiv \epsilon_I^{NNPU}. \tag{29}$$

With Eq. (18), Eq. (29), and $NP \sim \mathscr{O}\left(h^{-d}\right)$, we have the following global NN-PU approximation error with background RK approximation:

$$\| u^h - u \|_{0,\Omega} \leq \mathscr{O}\left(h_I^{p+1} \widehat{n}^{1/2}\right). \tag{30}$$

Remark 3.1. The optimal convergence property in the universal approximation theorem [2] used in Eq. (22) could be affected by the optimization algorithm, such as the gradient descent in minimizing the loss function. In the present work, the potential function in Eq. (3) is used as the loss function. The convexity in the loss function together with the NN enrichment of the background RK solution yields the optimal convergence in the NN part of solution as demonstrated in Fig. 11 in Section 5.1.1. For methods with different types of loss functions or with different solution strategies, such as those using the residual-based loss function and using NN basis functions for the total solution, the convergence property in the universal approximation theorem [2] should be used with caution.

4. Numerical implementation

The minimization problem in Eq. (3) can be expressed as:

$$\min_{\mathbf{d},\mathbb{W}} \left[\Pi\left(\boldsymbol{u}^h(\mathbf{d},\mathbb{W})\right)\right], \tag{31}$$

where $\boldsymbol{u}^h(\mathbf{d},\ \mathbb{W}) = \boldsymbol{u}^{RK}(\mathbf{d}) + \boldsymbol{u}^{NN}(\mathbb{W})$ is the NN-PU approximation with background RK approximation $\boldsymbol{u}^{RK}$ and $\mathbb{W} = \left\{\mathbb{W}^P, \mathbb{W}^\zeta,\ \mathbb{W}^C\right\}$ denoting the neural network weight sets of Parametric, NN Basis, and Coefficient sub-blocks, respectively, with $\mathbb{W}^P = \left\{\mathbf{W}_J^P\right\}_{J=1}^{n^B}$, $\mathbb{W}^\zeta = \left\{\mathbf{W}_J^\zeta\right\}_{J=1}^{n^B}$, and $\mathbb{W}^C = \left\{\mathbf{W}_J^C\right\}_{J=1}^{n^B}$. This work follows a staggered approach to obtain the NN-PU approximations, where the background RK solutions are first computed without NN enrichments, followed by the optimization of NN-PU approximations with fixed pre-computed background solutions. Fig. 7 outlines the solution procedures for obtaining the NN-PU approximation.

4.1. NN-PU approximation by loss function minimization

4.1.1. Input variables

The required inputs for the NN-PU optimization are given in the gray block in Fig. 7. The background RK coefficient vector $\mathbf{d} \equiv \{d_I\}_{I \in \mathscr{I}}$ is obtained by the standard Galerkin approximation with RK approximation functions [47,48,58]. Hence, the physical coordinates of domain integration evaluation points $\{x_e\}_{e=1}^{Ne}$ and a set of RK shape functions evaluated at those evaluation points $\left\{\{\Psi_I(\boldsymbol{x}_e)\}_{I=1}^{NP}\right\}_{e=1}^{Ne}$ are given as inputs. The background RK solutions are also pre-evaluated at those evaluation points and inputted for the computation of NN-PU approximation. Background nodal coordinates and RK nodal shape functions, evaluated with normalized support sizes of a and $2a$, are also pre-determined for error estimation analysis and updating NN-enriched nodesets as detailed in Section 4.1.3.

4.1.2. Loss function minimization

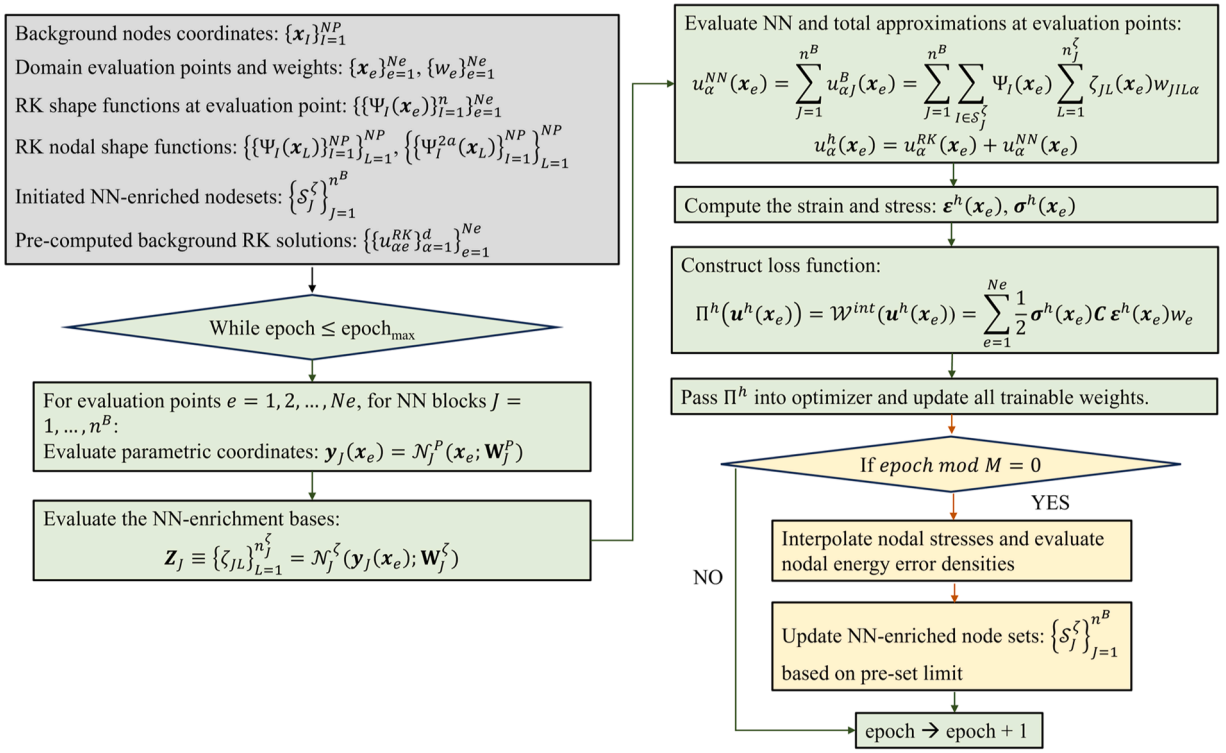
The NN loss function minimization procedures are outlined in green blocks in Fig. 7. For the *offline training* of the "parent" NN enrichment basis functions given background RK solution $\boldsymbol{u}^{RK}$, the weights and biases of the NN Basis and Coefficient sub-blocks are obtained by:

$$\mathbb{W}^{\zeta^0},\ \mathbb{W}^C = \underset{\mathbb{W}^{\zeta^0},\ \mathbb{W}^C}{\operatorname{argmin}} \left[\Pi\left(\boldsymbol{u}^{RK}(\mathbf{d}) + \boldsymbol{u}^{NN}\left(\left\{\mathbb{W}^{\zeta^0},\ \mathbb{W}^C\right\}\right)\right)\right]$$

subjected to

$$\boldsymbol{u}^{RK}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}),\quad \boldsymbol{u}^{NN}(\boldsymbol{x}) = 0,\quad \forall \boldsymbol{x} \in \partial\Omega^g. \tag{32}$$

Recall that the Parametric sub-block is designated as an identity map during offline training so that $\mathbb{W}^P = 0$. The weight set $\mathbb{W}^{\zeta^0}$ of the NN Basis sub-block and the resultant "parent" NN enrichment basis sets $\mathbb{Z}^0 = \left\{\mathscr{N}_J^\zeta\left(\boldsymbol{x}; \mathbf{W}_J^{\zeta^0}\right)\right\}_{J=1}^{n^B}$ are preserved for subsequent online computations for problems with similar local features. As indicated in Eq. (32), the essential boundary conditions are satisfied by the background RK solutions. The homogenous essential boundary condition is imposed on NN approximations by setting $w_{IJL} = 0$ for all I



* For brevity, this flowchart does not include the contributions of the body force and external tractions for the loss function computation

Fig. 7. Flowchart of the NN-PU solution procedures (gray block for inputs, green blocks for loss function minimization procedures, yellow blocks for updating NN-enriched nodesets). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

on essential boundaries.

The *online computation* of problems involving multiple features employs transfer-learning with feature-encoded "parent" NN enrichment basis sets, each targeting a single feature, as follows:

Given $\boldsymbol{u}^{RK}(\boldsymbol{d})$ and $\mathbb{Z}^0 = \left\{\mathcal{N}_J^{\zeta}\left(\boldsymbol{x}; \boldsymbol{W}_J^{\zeta^0}\right)\right\}_{J=1}^{n^B}$, find $\{\mathbb{W}^P,\ \mathbb{W}^C\}$ such that the loss function Π is minimized as:

$$\mathbb{W}^P,\ \mathbb{W}^C = \underset{\mathbb{W}^P,\ \mathbb{W}^C}{\operatorname{argmin}}\left[\Pi\left(\boldsymbol{u}^{RK}(\mathbf{d}) + \boldsymbol{u}^{NN}\left(\left\{\mathbb{W}^P, \mathbb{W}^{\zeta^0},\ \mathbb{W}^C\right\}\right)\right)\right]$$

subjected to

$$\boldsymbol{u}^{RK}(\boldsymbol{x}) = \boldsymbol{g}(\boldsymbol{x}),\quad \boldsymbol{u}^{NN}(\boldsymbol{x}) = 0,\quad \forall \boldsymbol{x} \in \partial\Omega^g. \tag{33}$$

During online solution computations, weight sets of NN Basis sub-blocks are known and pre-trained in a parametric space during offline computation, while the to-be-optimized weight set $\mathbb{W}^P$ finds an optimal mapping between $\boldsymbol{x}$ and $\boldsymbol{y}_J$ given the NN enrichment basis sets obtained from "parent" problems, and $\mathbb{W}^C$ is responsible for the coefficients of the NN basis functions.

4.1.3. Error indicator based on reproducing kernel background approximation

In the online computation, the NN enrichment nodes are initially informed by the error indicator for RK background solution. It has been investigated in [59,60], the discrete convolution of two reproducing kernels with *n*th-order bases and support sizes *a* results in a filtered kernel of the same order of completeness. This property makes reproducing kernel discrete convolution suitable for indicating errors and identifying high gradients in the RK background approximation needing solution enrichments. The filtered RK approximated stress, or recovered stress [60], denoted as $\boldsymbol{\sigma}^*(\boldsymbol{x})$, is obtained as follows:

$$\boldsymbol{\sigma}^*(\boldsymbol{x}) = \sum_{I\in\mathscr{I}} \Psi_I^{2a}(\boldsymbol{x})\boldsymbol{\sigma}^a(\boldsymbol{x}_I), \tag{34}$$

where $\boldsymbol{\sigma}^a(\boldsymbol{x}_I)$ is the nodal RK stress solution obtained by RK kernel function with support "a", and $\left\{\Psi_I^{2a}\right\}_{I\in\mathscr{I}}$ is the set of RK shape functions constructed with twice the support size used for the RK solution. The error indication of the stress solution is defined as:

$$\boldsymbol{e}_{\boldsymbol{\sigma}}^{*} = \boldsymbol{\sigma}^{*} - \boldsymbol{\sigma}^{a}, \tag{35}$$

and the nodal energy error density ρ_I^* can be computed using $\boldsymbol{e}_{\boldsymbol{\sigma}}^{*}$ as follows:

$$\rho_I^* = \left\{ \frac{1}{2} \boldsymbol{e}_{\boldsymbol{\sigma}}^{*}(\boldsymbol{x}_I)^T \boldsymbol{C}^{-1} \boldsymbol{e}_{\boldsymbol{\sigma}}^{*}(\boldsymbol{x}_I) \right\}^{1/2}, \ \forall I \in \mathscr{S}, \tag{36}$$

where $\boldsymbol{C}$ is the elasticity tensor. The NN enriched nodeset $\mathscr{S}^{\zeta}$ for a NN block is defined based on the nodal energy error densities:

$$\mathscr{S}^{\zeta} = \{ I | \rho_I^* \geq \tau \cdot \rho_{max}^* \}, \tag{37}$$

where $\rho_{\max}^* \equiv \max_{I \in \mathscr{S}} \rho_I^*$, and $\tau \in (0, 1]$ is a scaling parameter. Note that nodes situated on the Dirichlet boundaries are excluded in the node set $\mathscr{S}^{\zeta}$ for NN enrichment.

As per Eq. (9), the size of the NN enrichment coefficient set is closely related to the size of the NN enriched nodeset. Further, the error estimation analysis outlined in Eqs. (34)-(36) is performed periodically throughout the loss function minimization process, outlined in the yellow block in Fig. 7. The NN-enriched nodesets $\mathscr{S}^{\zeta}$ determined by Eq. (37) are updated in the gradient decent iteration until they stabilize.

5. Numerical examples

In this section, we present the offline construction of NN enrichment functions within the "parent" problems and illustrate their utilization in online solution computations through transfer learning. Subsequent subsections showcase the numerical procedures and explore solution convergence when employing the suggested NN enrichment for problems involving localized features:

- Section 5.1: Offline training of NN enrichment functions for "parent" problems with local features
- Section 5.2: Online solution of problems with similar local features by transfer leaning
- Section 5.3: Online solution of problems with multiple local features by transfer leaning

Unless specified otherwise, the background RK shape functions are constructed using the linear basis and cubic B-spline kernel function with a normalized support size (normalized with nodal distance) $a = 2.0$. Stabilized Conforming Nodal Integration [58] is employed for domain integration of the background solution and numerical integration of the loss function in Eq. (3). The RK shape function scaling approach [61] is introduced for strong imposition of essential boundary conditions on the pre-computed background RK approximations. Zero NN coefficients are enforced on essential boundaries to ensure that the NN solutions satisfy the homogenous essential boundary condition. Material properties $E = 10^4$ and $\nu = 0.3$ are chosen for all examples in this section, where E and ν are the Young's modulus and Poisson's ratio, respectively. In the NN Basis sub-block, the exponential linear unit (ELU) activation function [53] is employed for non-linear transformations, effectively addressing the vanishing gradient issue commonly associated with ReLU activation functions. ELU's distinctive inflection point in its first derivative, as shown in Fig. 8, makes it suitable for modeling stress singularity or concentration. The hyperbolic tangent activation function is selected for all hidden layers in the Parametric sub-block except the last one, where a linear activation function is applied. NN optimization utilizes the first-order Adam (adaptive momentum) optimizer [62], provided in the open-source machine learning library TensorFlow [26].

In the following demonstration problems, the solutions obtained from NN-PU with NN-enrichment will be compared with finite element (FE) solutions with various levels of h- and p- refinements. *Here, the FEM solutions are obtained by Galerkin approximation of static equilibrium using the* 3×3 *Gauss integration, that is the stationary of potential energy, while the NN-PU solution is obtained via minimization of potential energy through gradient decent iterative procedures as stated in Section 4.1*. In addition, according to the property that the Galerkin approximation (obtained from the stationary state of potential energy) overestimates the potential energy of the continuum system in equilibrium [63], the potential energy in the NN-PU loss function will be compared with the FE potential energy as an accuracy measure. The NN-PU CPU reported in the numerical examples includes both background RK solution computations and online NN optimizations.

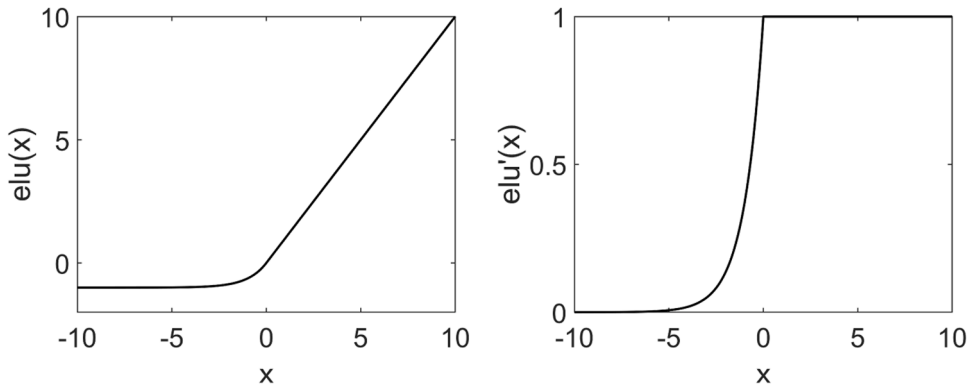


Fig. 8. Plot of the ELU activation function (left) and its first derivative (right).

5.1. Offline training of parent NN enrichment functions

This subsection demonstrates the offline training of NN enrichment functions in two "parent" problems, and examines the efficiency, accuracy, and convergence of the proposed NN-PU computational framework. These problems target local features and employ a single NN block with the Parametric sub-block set to be an identity map in constructing the NN enrichment functions. Neural network iteration convergence is met either when the loss function reduction is less than 10 percent of the learning rate in 5 epochs or when the total epochs surpass 30,000.

5.1.1. Parent problem 1: Plate with a hole

The first "parent" problem concerns a plate with a circular hole subjected to a far-field uniaxial tension of 10. Due to the axisymmetry, only the upper right quadrant of the domain is modeled, as shown in Fig. 9. Symmetric boundary conditions are imposed on the left vertical and bottom horizontal edges, while the exact traction field, calculated using the analytical solutions presented in [64], is prescribed on the remaining edges. This problem focuses on analyzing the convergence rates of NN-PU solutions and how the selection of the enrichment region influences the potential's minimizing performance.

Three levels of background discretization are employed to facilitate the convergence studies, as shown in Fig. 10, and a single hidden layer with 5 NN enrichment basis functions is used for constructing the NN basis functions. The solution accuracy is investigated in terms of the normalized displacement error norms (e_{L2}) with high-order Gauss integration as follows:

$$e_{L2} = \sqrt{\frac{\int_\Omega (\boldsymbol{u}^{\text{exact}}(\boldsymbol{x}) - \boldsymbol{u}^h(\boldsymbol{x}))^2 d\Omega}{\int_\Omega (\boldsymbol{u}^{\text{exact}}(\boldsymbol{x}))^2 d\Omega}}. \tag{38}$$

The convergence curves for varying background RK nodal spacing (h) with different fixed hidden layer widths (n_{NR}) are demonstrated in Fig. 11(a), and Fig. 11(b) presents the convergence curve for varying hidden layer widths for each background discretization. The NN enrichment is introduced to all discrete nodes, except for the case represented by the red curve in Fig. 11(a), where only

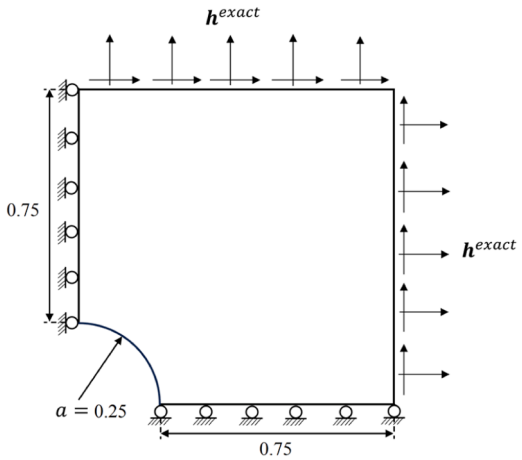


Fig. 9. Plate with a hole unit problem setting.

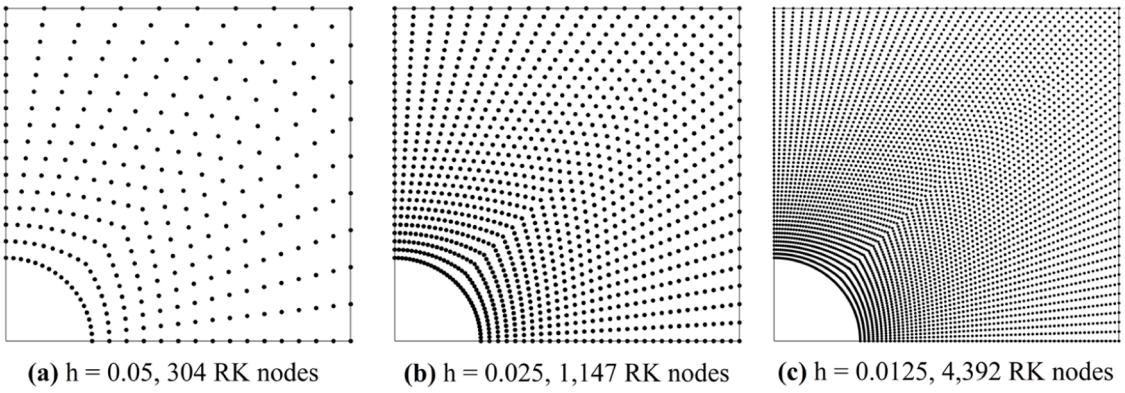


Fig. 10. Background RK discretization: (a) 304 nodes with $h = 0.05$ (b) 1147 nodes with $h = 0.025$ and (c) 4392 nodes with $h = 0.0125$ (h: averaged nodal spacing).

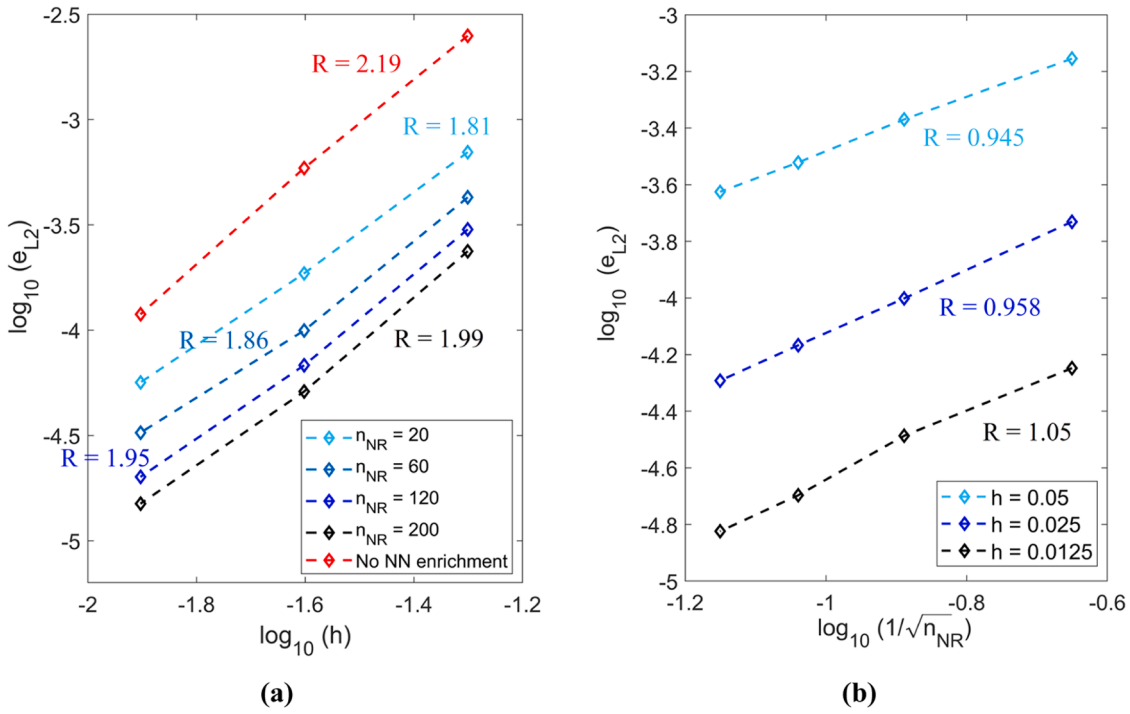


Fig. 11. Normalized L_2 convergence rates: (a) for varying background h with different fixed widths of hidden layer and (b) for varying n_{NR} with different level of background discretization (R: average rate of convergence).

background RK approximation is considered, i.e., $u^h(\boldsymbol{x}) = u^{RK}(\boldsymbol{x})$. As indicated in Fig. 11(a), the background RK solutions achieve the optimal convergence rate of 2, consistent with the analytical convergence in Eq. (30) for the linear basis. Additionally, with a fixed background RK approximation at each domain discretization level, the NN-PU approximation yields convergence rate of 1 with respect to $\hat{n}^{-1/2}$, where $\hat{n}$ is the number of neurons, again verify the error analysis result in Eq. (30).

Next, the adaptive NN enrichment is investigated. The coarse domain discretization with $h = 0.05$, as shown in Fig. 10(a) is utilized. Four cases are examined, using an energy error density ((34)-(37)) thresholds of $0.1\rho^*_{max}$, $0.2\rho^*_{max}$, $0.5\rho^*_{max}$ and $0.9\rho^*_{max}$, as shown in Fig. 12. We also consider enriching all nodes and adaptively selecting NN enrichment regions during the loss function minimization, as shown in Fig. 13. The results of total potential energy (loss function) minimization for all cases are demonstrated in Fig. 14, where the reference energy is calculated according to the analytical solutions. Enriching all nodes within the domain results in the potential energy converging closest to the analytical reference energy level among all cases. Alternatively, adaptive NN enrichment during optimization achieves comparable accuracy to that achieved by enriching all nodes.

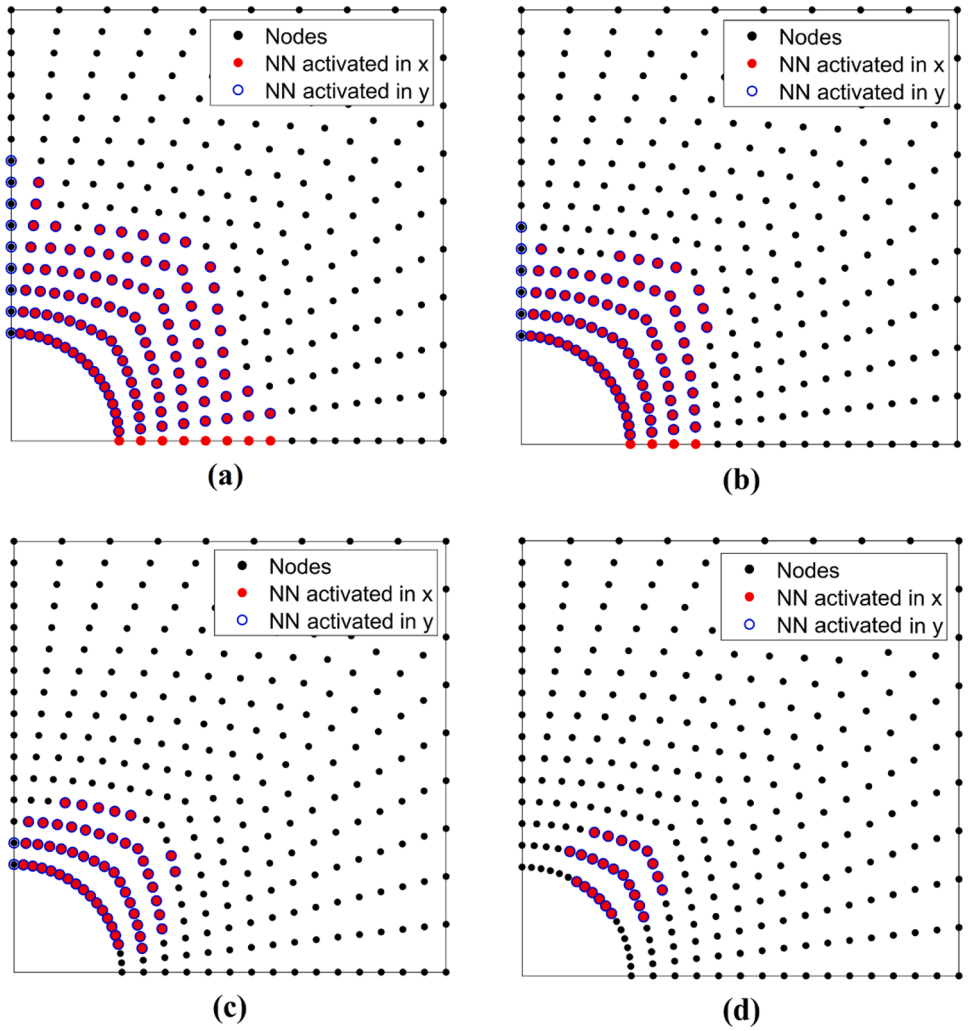


Fig. 12. NN enrichment region selections: (a) 113 and 114 nodes are enriched in x- and y- directions based on $0.1\rho^*_{\max}$ (b) 80 and 82 nodes are enriched in x- and y-directions based on $0.2\rho^*_{\max}$ (c) 50 and 52 nodes are enriched in x- and y-directions based on $0.5\rho^*_{\max}$ and (d) 24 nodes are enriched in both directions based on $0.9\rho^*_{\max}$.

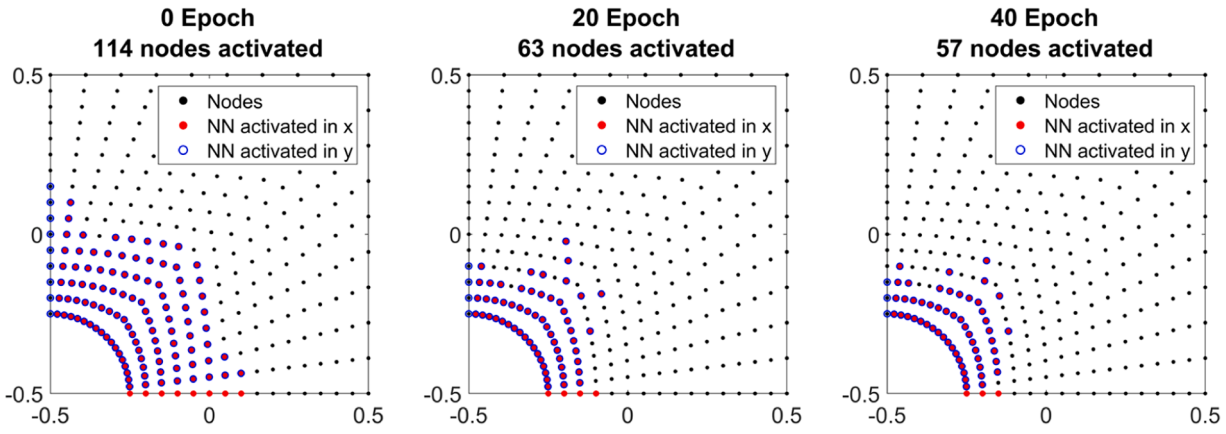


Fig. 13. Adaptive NN enrichment during loss function minimization.

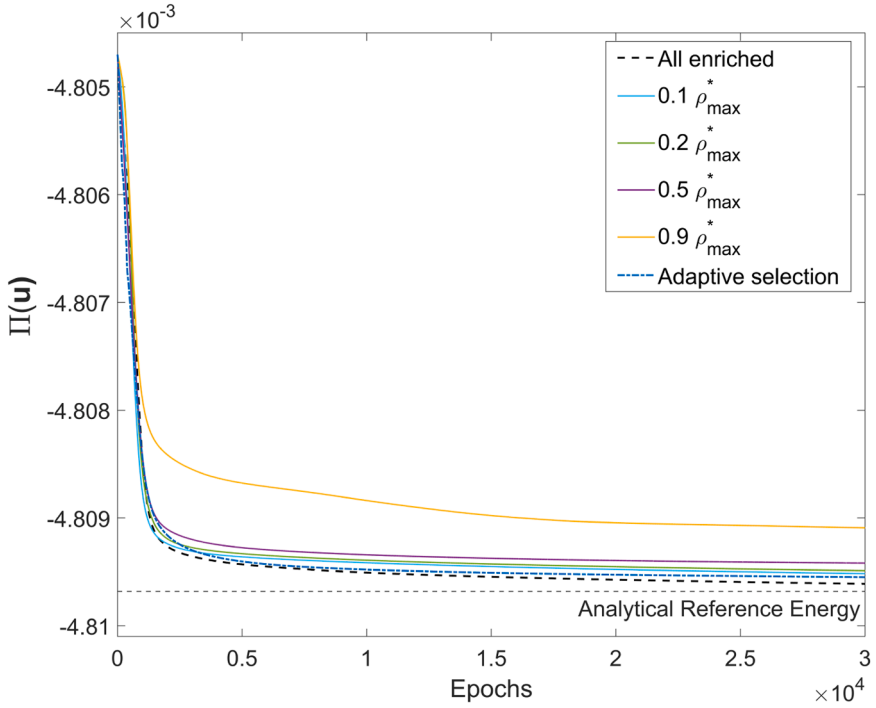


Fig. 14. Loss function minimization for NN enrichment regions using different energy error density thresholds and adaptive enrichment.

5.1.2. Parent problem 2: L-shaped panel

Fig. 15 presents an L-shaped panel "parent" problem with a surface angle $\theta = 90^{\circ}$. As a reference of the NN-PU background discretization, finite element discretization with various *h*- and *p*- refinements shown in Fig. 16 are considered, where Q4 and Q8 represent the 4-node quadrilateral element and the 8-node serendipity element, respectively. The FEM stress solutions shown in Fig. 17 reveal that only the Q8 FEM solution with 578,291 nodes precisely captures the stress singularity around the concave corner.

The NN-PU background discretization is based on the nodal locations of the coarsest FE discretion with 341 nodes in Fig. 16(a). Three parametric studies are performed to evaluate the effects of NN Basis sub-block's architecture on its convergence behaviors in minimizing the total potential energy functional (loss function). The NN-enriched nodeset $\mathscr{S}^{\zeta}$ obtained based on the algorithms in Eqs. (34)-(37) with a threshold of $0.9\rho^{*}_{max}$ contains 13 nodes around the concave corner, as shown in Fig. 18, and they are used as the enrichment nodes for all parametric studies. A constant learning rate of 10^{-4} is chosen for optimization of NN enrichment basis functions.

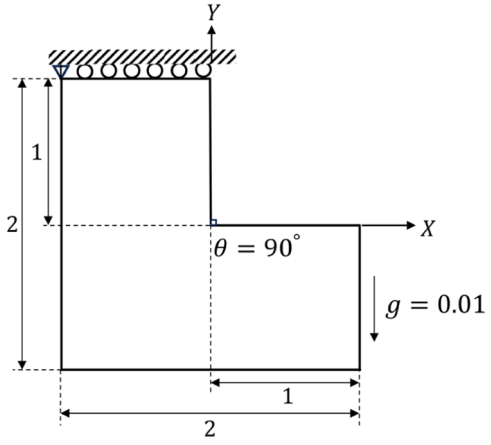


Fig. 15. L-shaped panel problem setting.

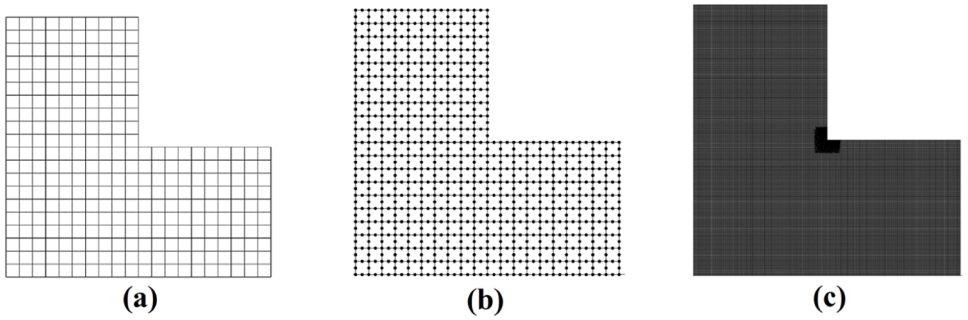


Fig. 16. Domain discretization used for obtaining FEM solutions: (a) Q4 FEM with 341 nodes (b) Q8 FEM with 981 nodes (c) Q8 FEM with 578,291 nodes.

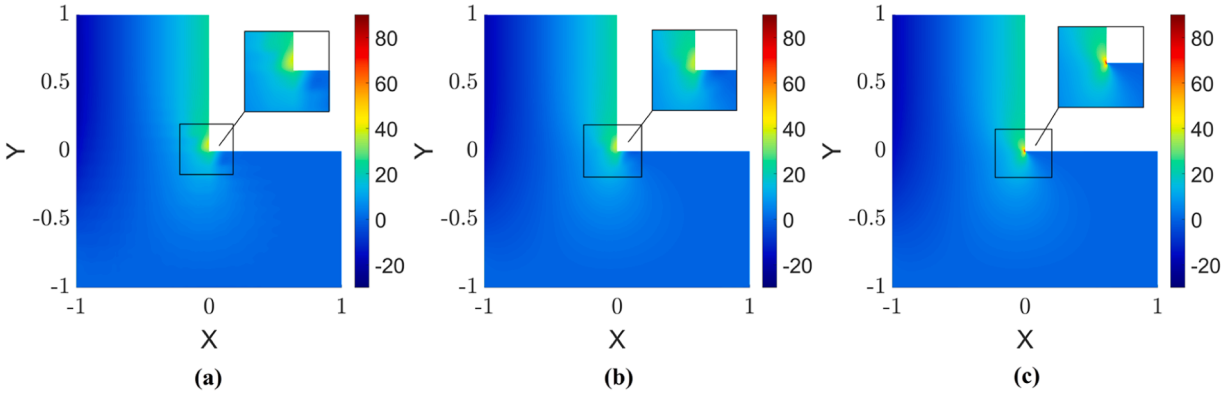


Fig. 17. Comparison of the FEM σ_{22} results: (a) Q4 FEM with 341 nodes (b) Q8 FEM with 981 nodes (c) Q8 FEM with 578,291 nodes.

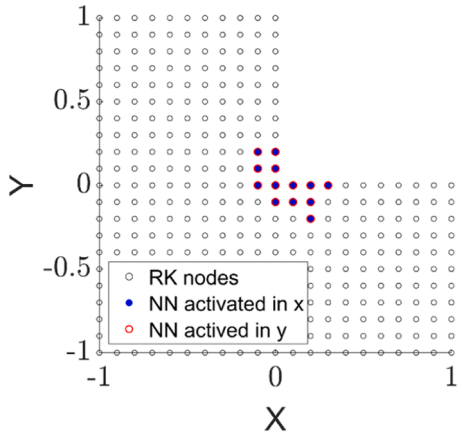


Fig. 18. Nodes activated for NN enrichment.

(1) The effects of the number of NN-enrichment bases n^{ζ}: The number of NN-enrichment bases is determined by the number of neurons in the last hidden layer of NN Basis sub-block. A four-layered deep neural network is adopted for this study, and the first 3 hidden layers contain 20 neurons in each hidden layer.

It is expected that the numerical solution with lower potential energy corresponds to a higher solution accuracy. As shown in Fig. 19, FE model with finer discretization and higher order of approximation generates lower potential energy. Fig. 19 depicts the optimization results of loss functions (potential energies) for NN architectures featuring 2 to 9 NN-enrichment bases. Zoom-in plots display where the loss functions intersect with reference potential energy levels and the final converged potential energy levels. Table 1 summarizes the required epochs and CPU for different NN architectures to yield potential energy right below those obtained from the

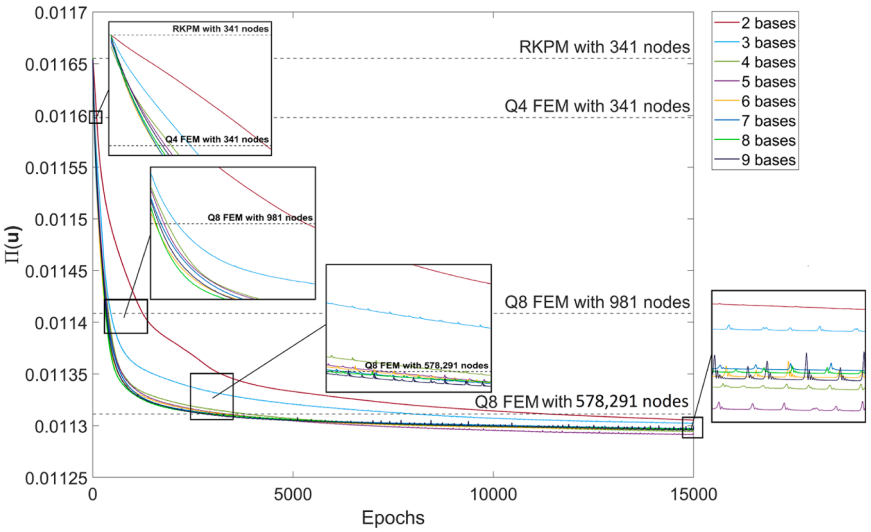


Fig. 19. Loss function (potential energy) minimization for NN architectures with different numbers of NN-enrichment bases and the comparison with FEM potential energies.

Table 1
Summary of the potential energies (PE) and CPU time in seconds for NN architectures with different numbers of NN-enrichment bases.

Number of Bases	PE less than Q4 FEM with 341 nodes (PE = 1.1598E-02, CPU = 1.4 s)		PE less than Q8 FEM with 981 nodes (PE = 1.1409E-02, CPU = 2.1 s)		PE less than Q8 FEM with 578,291 nodes (PE = 1.1311E-02, CPU = 40 s)		Converged PE
	Epoch	CPU (s)	Epoch	CPU (s)	Epoch	CPU (s)	
2	97	1.6	1248	20.0	11,342	181.5	1.130139E-02
3	51	0.83	451	7.4	7989	130.7	1.129684E-02
4	39	0.65	393	6.6	3827	64.0	1.128858E-02
5	35	0.60	372	6.3	3325	56.5	1.128733E-02
6	23	0.40	300	5.2	4066	70.9	1.129156E-02
7	30	0.53	318	5.7	3222	57.3	1.129242E-02
8	31	0.56	323	5.9	2908	52.8	1.129219E-02
9	34	0.63	345	6.4	3018	55.8	1.129262E-02

FEM solutions. Increasing the number of learned NN-enrichment bases from 2 to 5 results in a noteworthy reduction in the required epochs and CPU to achieve the specific potential energy level of the much-refined FEM solution, as observed in Table 1. However, further increasing the number of NN bases beyond 5 does not lead to a further reduction in the converged potential energy. Therefore, n^{ζ} is selected to be 5 for the remaining parametric studies in this example.

(2) The effects of the number of hidden layers: In this study, each hidden layer, except for the last one, contains 20 neurons, and 5 neurons are used in the last hidden layer, which corresponds to 5 NN bases.

The loss functions during the minimization process for NN architectures with 2 to 8 hidden layers are plotted in Fig. 20. Table 2 summarizes the required numbers of epochs and CPU for the loss functions of different architectures to become lower than specific reference FE potential energy levels, together with the converged potential energies. For NN Basis sub-block comprising of 2 or 3 hidden layers, the optimization process is computationally demanding. However, it's evident that deeper NN architectures lead to a significant reduction in the potential energy, as well as reduced epochs and total CPU. Employing an eight-layered NN architecture with 5 bases achieves the lowest reference potential energy level of the finest FEM solution in just 6.3 s, far outperforming the 40-second CPU time in FEM.

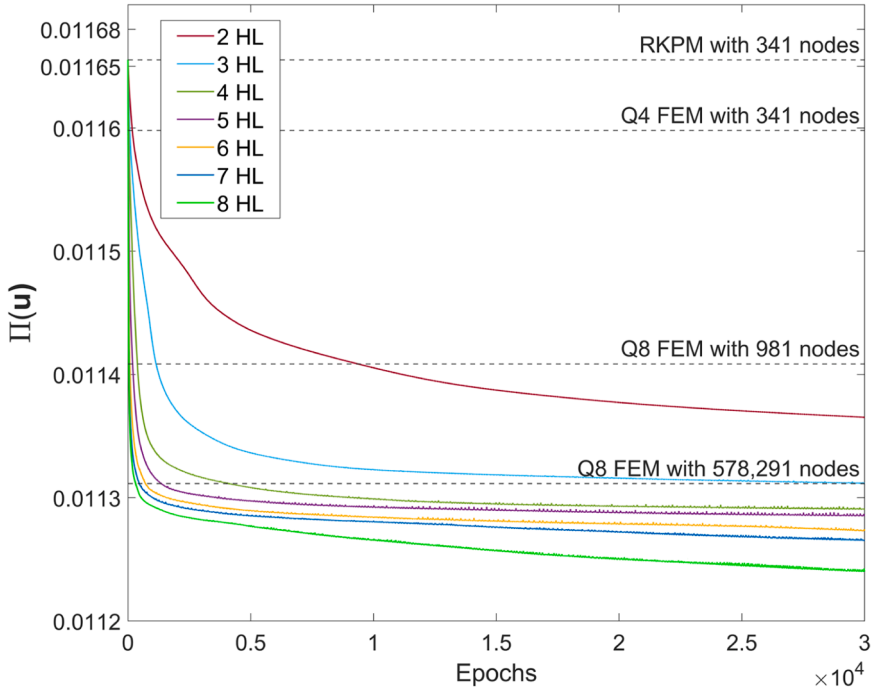


Fig. 20. Loss function minimization for NN architectures containing different numbers of hidden layers and the comparison with FEM potential energies.

Table 2
Summary of the potential energies (PE) and CPU time in seconds for NN architectures containing different number of hidden layers.

Number of hidden layers	PE less than Q4 FEM with 341 nodes (PE = 1.1598E-02, CPU = 1.4 s)		PE less than Q8 FEM with 981 nodes (PE = 1.1409E-02, CPU = 2.1 s)		PE less than Q8 FEM with 578,291 nodes (PE = 1.1311E-02, CPU = 40 s)		Converged PE
	Epochs	CPU (s)	Epochs	CPU (s)	Epochs	CPU (s)	
2	178	1.8	9360	93.6	> 30,000	–	1.135332E-02
3	73	0.88	1146	17.2	> 30,000	–	1.131179E-02
4	35	0.60	372	6.3	3325	56.5	1.128733E-02
5	21	0.38	190	3.4	1356	24.4	1.128538E-02
6	11	0.21	78	1.5	716	13.6	1.127323E-02
7	7	0.14	48	0.96	463	9.3	1.126534E-02
8	4	0.084	37	0.78	313	6.3	1.125375E-02

(3) The effects of the number of neurons within each hidden layer: The NN Basis sub-block is constructed using 8 hidden layers and 5 bases, and the first 7 hidden layers contain an identical number of neurons.

Fig. 21 demonstrates the potential energies for NN architectures featuring 10 to 80 neurons per hidden layer. Table 3 documents the epochs and CPU required for various NN architectures to generate potential energy below the reference FEM potential energies and their respective converged energy levels. The results clearly indicate that increasing the width of the hidden layer from 10 neurons to 40 neurons yields a notable acceleration in the optimization speed of the neural network, as the NN-enrichment bases contain more trainable parameters to capture the local features. However, as the number of neurons per hidden layer further increases, the improvement becomes marginal. Overall, when employing 8 hidden layers and 5 bases for constructing the NN-enrichment, the offline learning is shown to be more efficient than the Q8 FEM solution with 579,012 nodes (FEM CPU = 40 s), especially when the hidden layer has a width of 20 neurons or more.

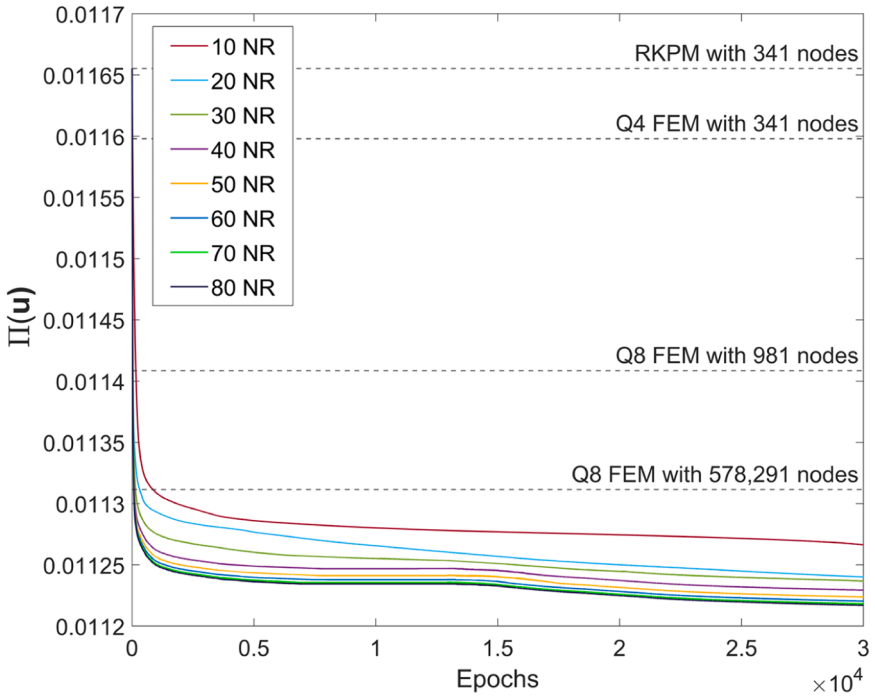


Fig. 21. Loss function minimization for NN architectures containing different numbers of neurons per hidden layer and the comparison with FEM potential energies.

Table 3
Summary of the potential energies (PE) and CPU time in seconds for NN architectures containing different number of neurons per hidden layer.

No. of Neurons	PE less than Q4 FEM with 341 nodes (PE = 1.1598E-02, CPU = 1.4 s)		PE less than Q8 FEM with 981 nodes (PE = 1.1409E-02, CPU = 2.1 s)		PE less than Q8 FEM with 578,291 nodes (PE = 1.1311E-02, CPU = 40 s)		Converged PE
	Epochs	CPU (s)	Epochs	CPU (s)	Epochs	CPU (s)	
10	22	0.46	155	3.26	860	18.1	1.126635E-02
20	4	0.08	37	0.78	313	6.3	1.125375E-02
30	5	0.11	32	0.70	140	3.1	1.123675E-02
40	5	0.12	29	0.67	86	2.0	1.122940E-02
50	5	0.13	29	0.73	77	1.9	1.122378E-02
60	6	0.16	31	0.81	72	1.9	1.122042E-02
70	7	0.20	31	0.90	70	2.0	1.121817E-02
80	6	0.19	32	0.99	71	2.2	1.121705E-02

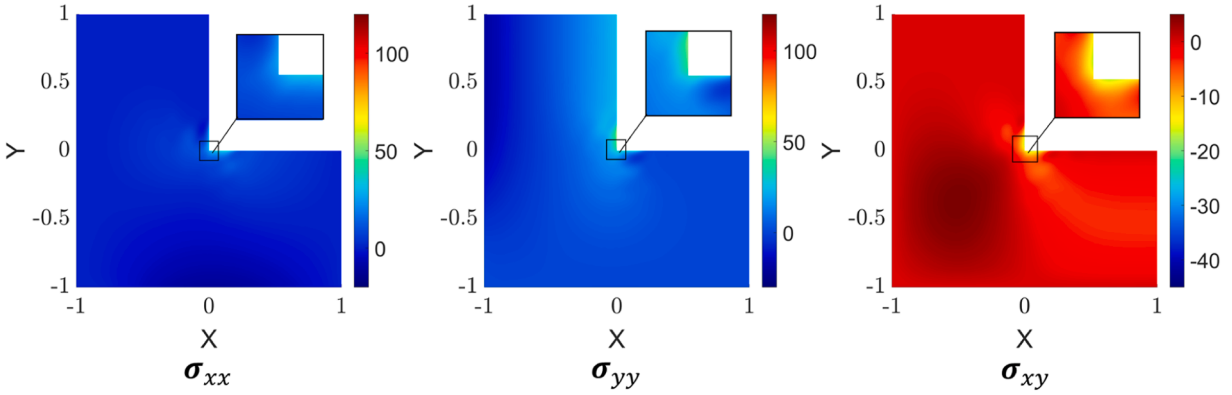
Fig. 22 demonstrates a comparison between the stress distributions obtained using the RKPM with 341 nodes, NN-PU approximation with 341 nodes and 2815 NN hyperparameters, and Q8 FEM with 578,291 nodes (over a million degrees of freedom). The NN Basis sub-block comprises 8 hidden layers with 20 neurons each, and 5 bases, totaling 2685 trainable parameters. The NN enrichment is applied to 13 nodes, each with 2 degrees of freedom, yielding 130 NN enrichment coefficients. Consequently, the total number of NN unknowns is 2815. The normalized energy error norm (e_{H1}) of the NN-PU approximations is computed with high-order Gauss integration using Eq. (39) to quantitatively measure the relative error of the NN-PU approximations compared to the finest Q8 FEM solutions, and $e_{H1} = 2.23$ % is obtained with the selected NN enrichment bases.

$$e_{H1} = \sqrt{\frac{\int_\Omega (\boldsymbol{\varepsilon}^{\mathrm{FEM}}(\boldsymbol{x}) - \boldsymbol{\varepsilon}^h(\boldsymbol{x})) \cdot (\boldsymbol{\sigma}^{\mathrm{FEM}}(\boldsymbol{x}) - \boldsymbol{\sigma}^h(\boldsymbol{x})) d\Omega}{\int_\Omega \boldsymbol{\varepsilon}^{\mathrm{FEM}}(\boldsymbol{x}) \cdot \boldsymbol{\sigma}^{\mathrm{FEM}}(\boldsymbol{x}) d\Omega}} \tag{39}$$

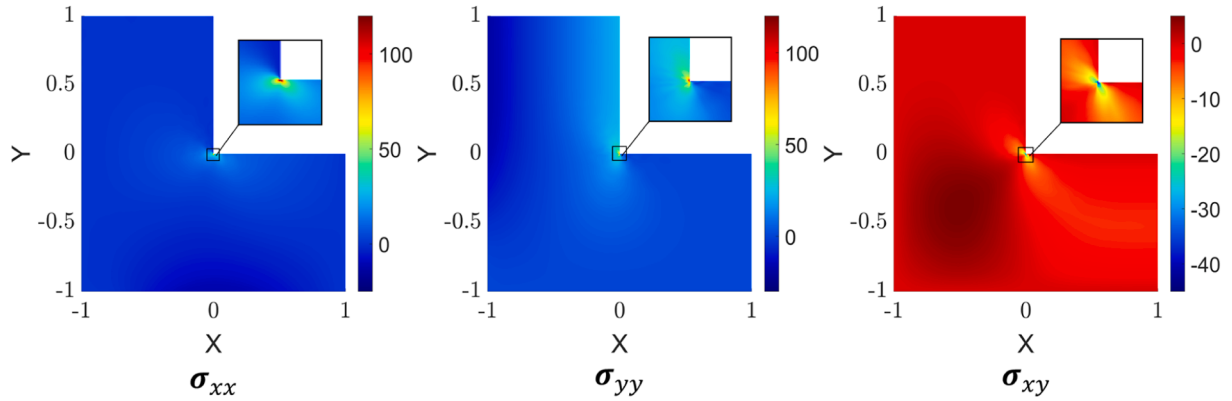
As shown in Fig. 22, the NN-PU approximation is able to obtain comparable solutions with the ones obtained using the Q8 FEM with a much finer discretization with just 0.3 % DOFs and 16.6 % CPU time of the FEM computation.

5.2. Online analysis with transfer learning

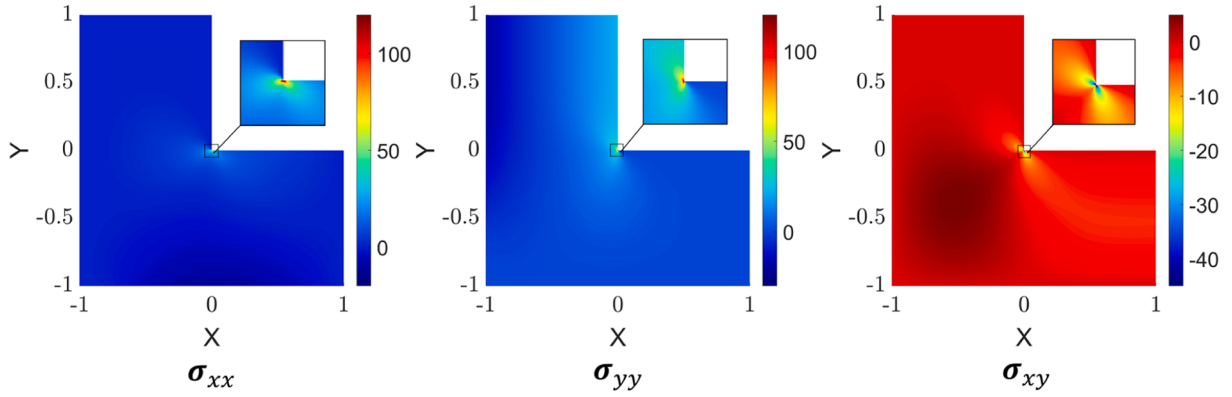
This subsection demonstrates how the NN basis functions constructed for local features in the "parent" problems introduced in



(a) RKPM with 341 nodes



(b) NN-PU with 341 RK nodes and 2,815 NN unknowns



(c) Q8 FEM with 578,291 nodes

Fig. 22. Comparison of the stress solutions: (a) RKPM solutions with 341 nodes (b) NN-PU solutions with 341 RK nodes and 2815 NN unknowns and (c) Q8 FEM solutions with 578,291 nodes.

Section 5.1 can be used as the NN enrichments for problems with the same or similar local features via a transfer learning strategy as introduced in Section 4.1.2.

5.2.1. Circular-shaped panel with a 90-degree corner: transfer learned from the L-shaped parent problem

Fig. 23 presents a variation of the "parent" L-shaped panel discussed in Section 5.1.2 to a circular-shaped panel with a 90-degree corner. Despite these domain geometry changes, the problem maintains its local 90-degree corner as in the L-shaped "parent" problem.

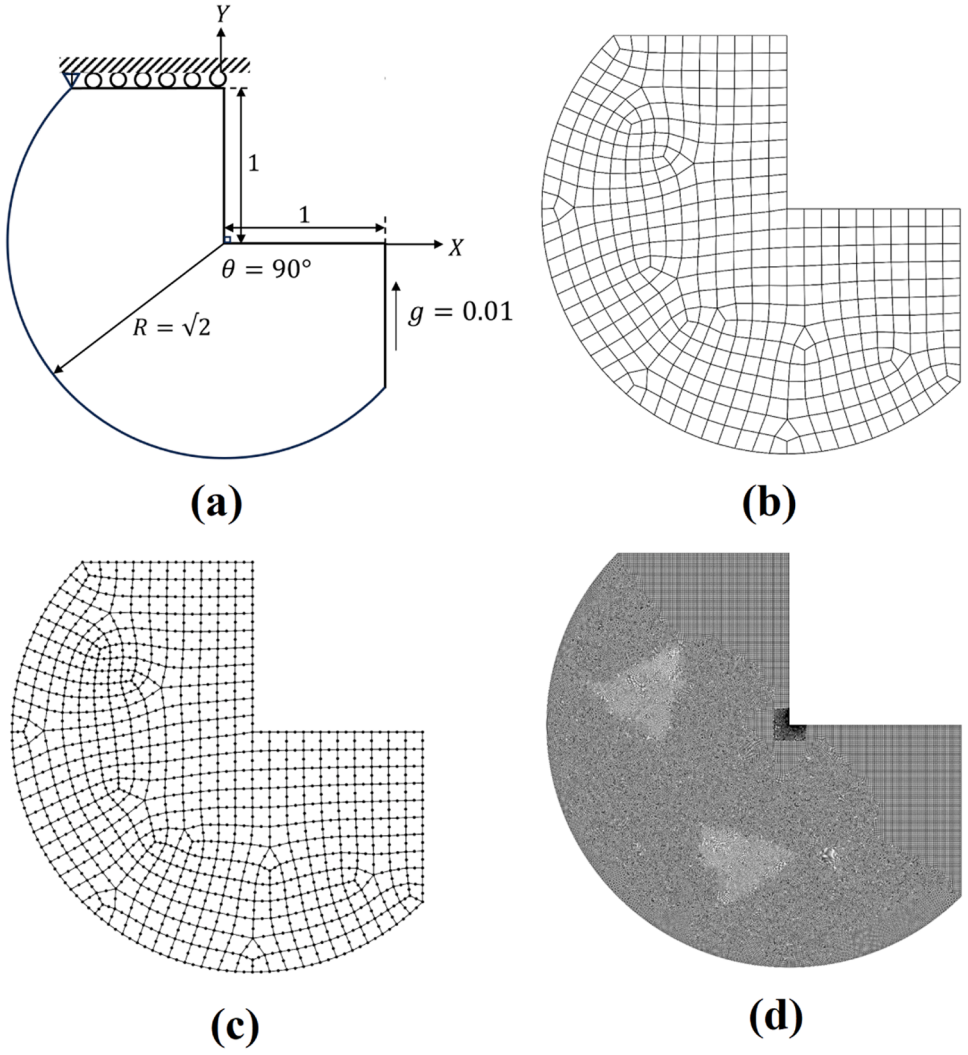


Fig. 23. (a) L-shape panel problem setting with a circular domain and 3 levels of FE discretization: (b) Q4 FEM with 480 nodes (c) Q8 FEM with 1396 nodes (d) Q8 FEM with 776,438 nodes.

Utilizing a "parent" NN-enrichment basis set obtained from the L-shaped problem, transfer learning is applied to the online solution stage of this circular domain problem. We incorporate a Parametric sub-block $\mathcal{N}^P$ comprising 2 hidden layers with 5 neurons in the first hidden layer, 2 neurons in the second hidden layer, totaling of 27 trainable parameters, to facilitate the utilization of the pre-trained NN enrichment basis sets during online computations. The same set of NN basis functions in the L-shaped problem are employed without additional optimization in their spatial distributions, and only the weight sets of the Parametric sub-block ($\mathbf{W}^P$) and NN basis function coefficients ($\mathbf{W}^C$) are iterated in the loss function minimization.

The chosen NN basis sub-block architecture from the "parent" L-shaped problem is composed of 8 hidden layers, with the first 7 containing 20 neurons each, and it considers 5 NN-enrichment bases as studied in Section 5.1.2. The offline training time of the selected NN enrichment bases is 6.3 s. During the online solution stage, the NN enriched nodeset is updated every 20 epochs with a threshold of $0.1\rho^*_{max}$ until the number of enriched nodes remains unchanged. Fig. 24 demonstrates the adaptive enrichment process of the NN-enriched nodes during the optimization, where the neural network efficiently identifies the areas where enrichment is most needed within just 40 epochs, consolidating the number of enriched nodes from 180 to 29.

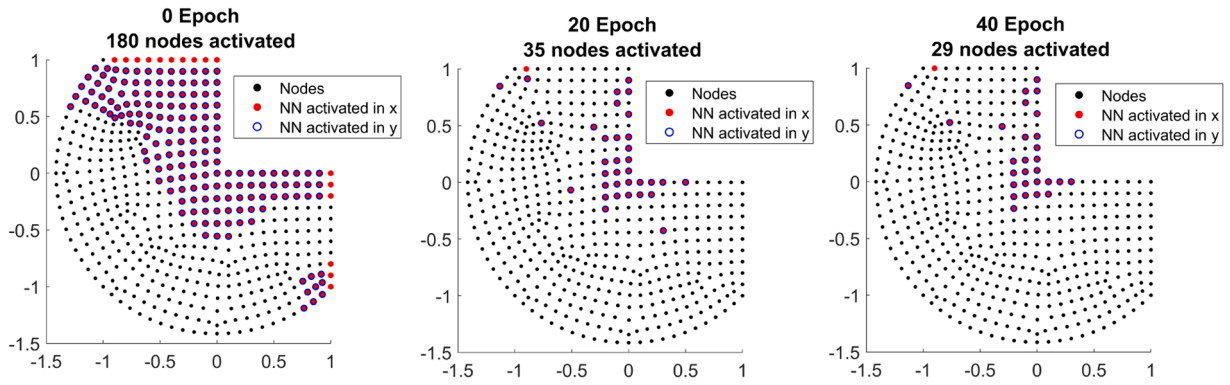


Fig. 24. Adaptive selection of the nodes for NN enrichment.

Fig. 25 demonstrates the NN-PU loss function minimization in the online solution stage in comparison with various reference potential energy levels computed using FEM solutions with varying spatial discretization levels shown in Fig. 23. Table 4 provides a detailed comparison of the number of unknowns, the discrete total potential energy levels, and the computation time required for NN-PU and FEM with different domain discretization. In addition, it also reports the number of required epochs and CPU for the NN-PU online calculation with the learned "parent" NN-enrichment basis set to reach each FEM potential energy level. When compared to the high-order highly refined FEM model (with 1551,873 DOFs), the NN-PU approximation achieves similar accuracy with just 317 NN variables (1254 DOFs total) as shown in Fig. 26, and requires only 20.4 % of computation time in the online computation as shown in Table 4. The final converged potential energy for the online solution is 1.6518E-02, 3.74 % lower (more accurate) than that from Q8 FEM with 776,438 nodes (1551,873 DOFs).

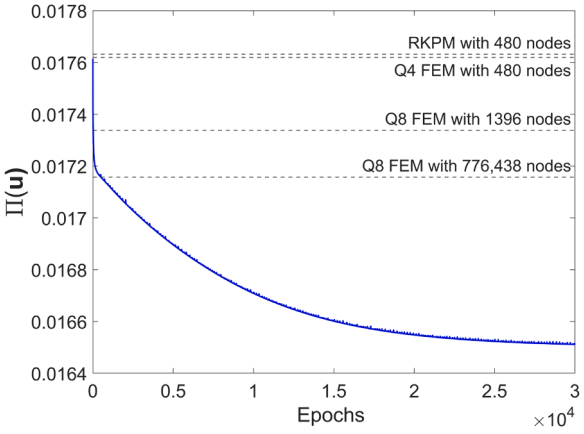
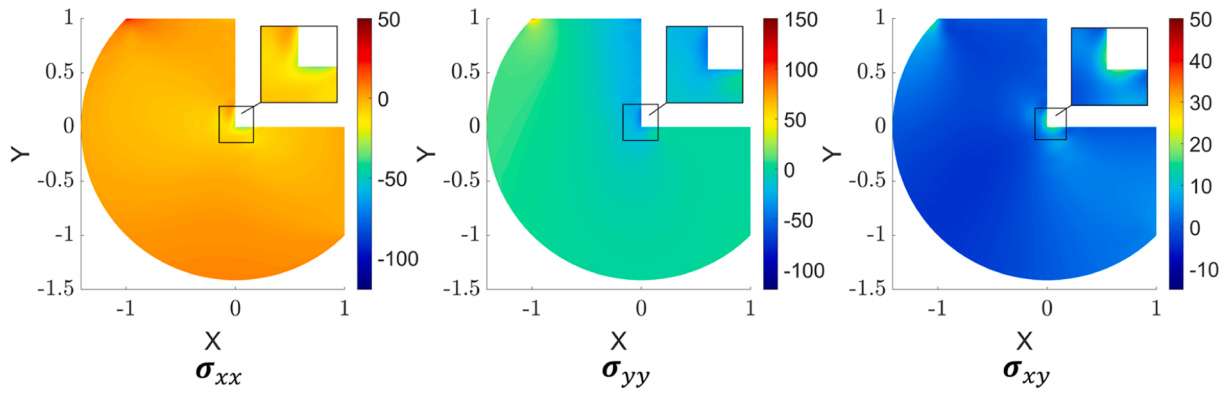


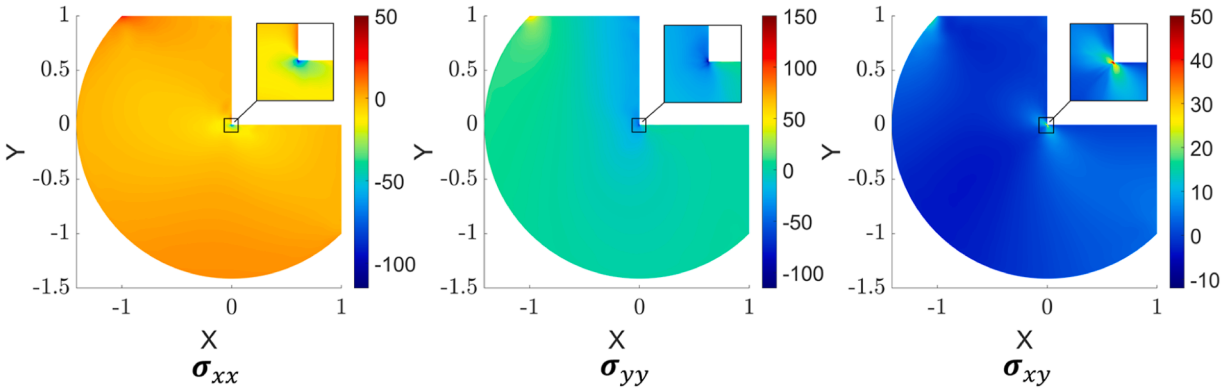
Fig. 25. Loss function minimization of NN-PU and the comparison with FEM potential energies (PEs).

Table 4
Comparison of the required numbers of epochs and runtimes for NN-PU solution to reach the potential energy (PE) of FEM with different discretization for the circular L-shaped panel problem.

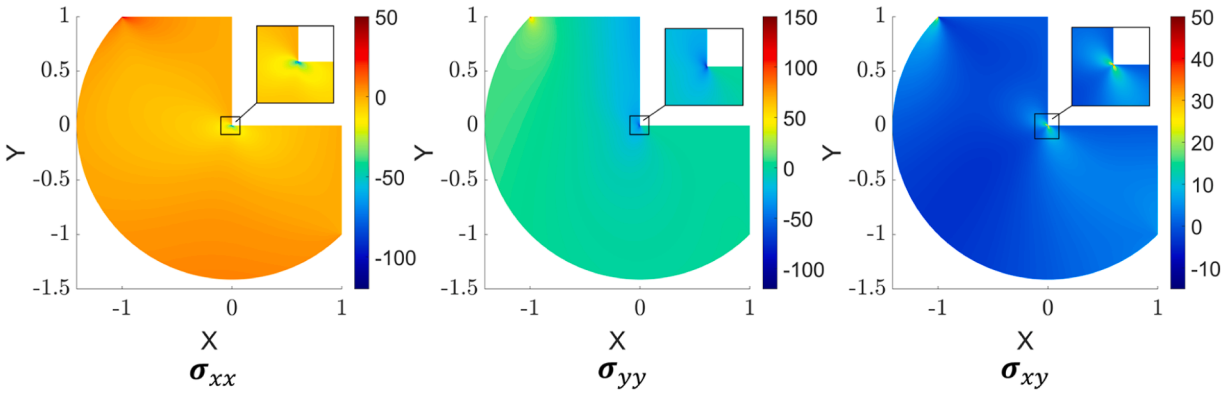
NN-PU with potential energy (PE) lower than FEM PE	Q4 FEM with 480 nodes (PE = 1.762E-02, CPU = 1.9 s)	Q8 FEM with 1396 nodes (PE = 1.734E-02, CPU = 2.9 s)	Q8 FEM with 776,438 nodes (PE = 1.716E-02, CPU = 56.3 s)
Number of Epochs NN-PU with 480 nodes used to reach each FEM potential energy levels	1	16	468
CPU for NN-PU with 480 nodes to reach each FEM potential energy level (s)	2.2	2.4	11.5



(a) RKPM with 480 nodes



(b) NN-PU with 480 RK nodes and 317 NN unknowns



(c) Q8 FEM with 776,438 nodes

Fig. 26. Comparison of the stress solutions: (a) RKPM solutions with 480 nodes (b) NN-PU solutions with 480 RK nodes and 317 NN unknowns and (c) Q8 FEM solutions with 776,438 nodes.

5.2.2. L-shape panel with 60° *corner: transfer learned from the L-shaped parent problem*

Here we consider a L-shaped panel problem featuring a 60° interior corner as shown in Fig. 27. Different from the "parent" L-shaped panel with 90° corner, the degree of singularity around the concave corner has changed, and the essential boundary condition is now

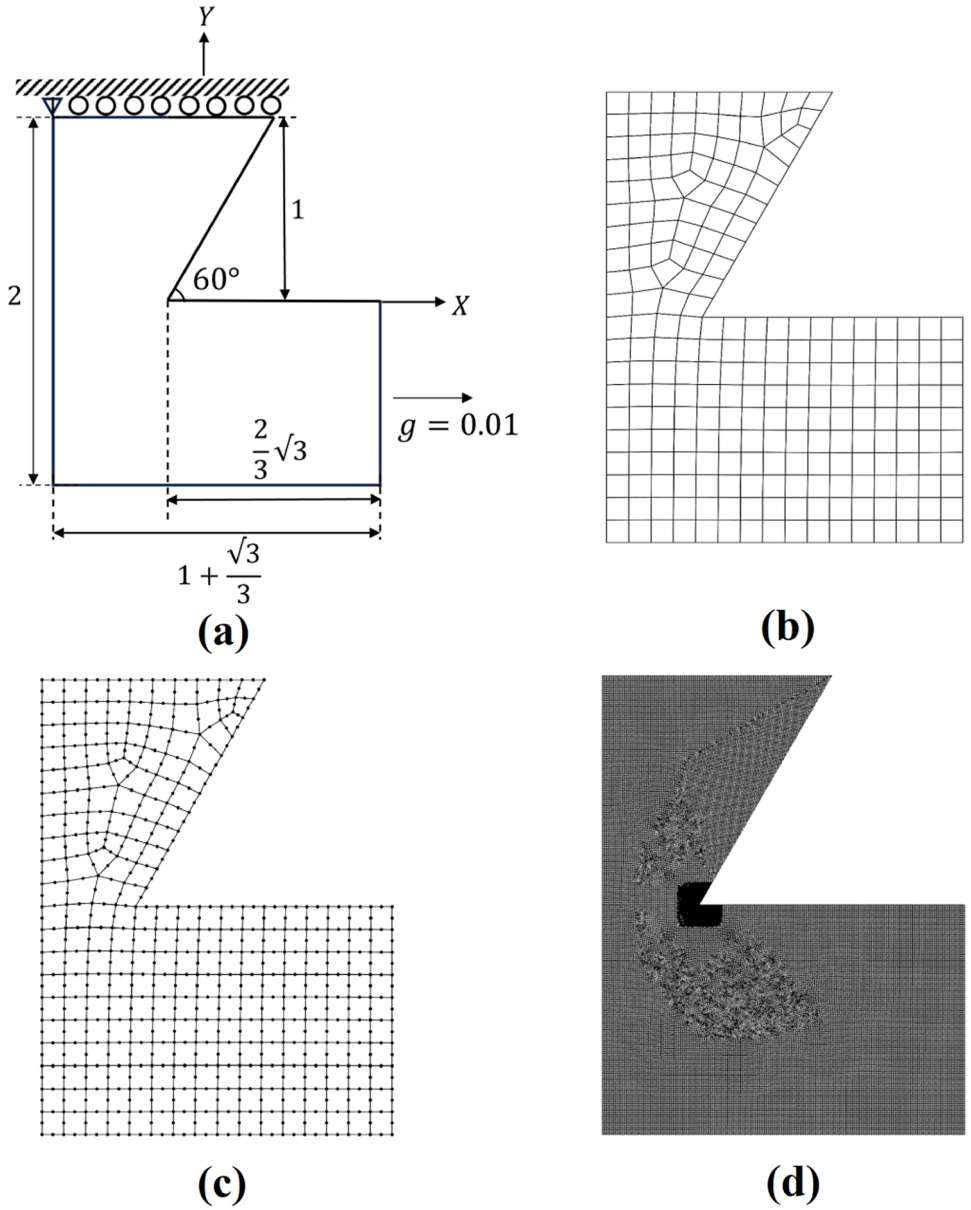


Fig. 27. (a) L-shape panel with 60° corner problem setting and 3 levels of FE discretization: (b) Q4 FEM with 273 nodes (c) Q8 FEM with 777 nodes (d) Q8 FEM with 447,602 nodes.

applied along the positive x-axis. Due to the change of local feature, we use a larger Parametric sub-block $\mathcal{N}^{\mathcal{P}}$ comprising 3 hidden layers with 20 neurons in the first two hidden layers and 2 neurons in the last hidden layer, totaling of 522 trainable parameters. The NN Basis sub-block is transfer learned from the 90° L-shaped problem in Section 5.1.2 again containing 8 hidden layers, with 20 neurons in the first 7 layers, and 5 NN-enrichment bases in the last hidden layer, and the offline training time of the selected NN enrichment bases is 6.3 s. Like the previous problems, the selection of NN-enriched nodes is dynamically adjusted during the loss function minimization process with an activation threshold of $0.1\rho^{*}_{max}$. As shown in Fig. 28, the initial set of 148 nodes chosen for NN-enrichment is subsequently condensed to just 50 nodes situated around the corner.

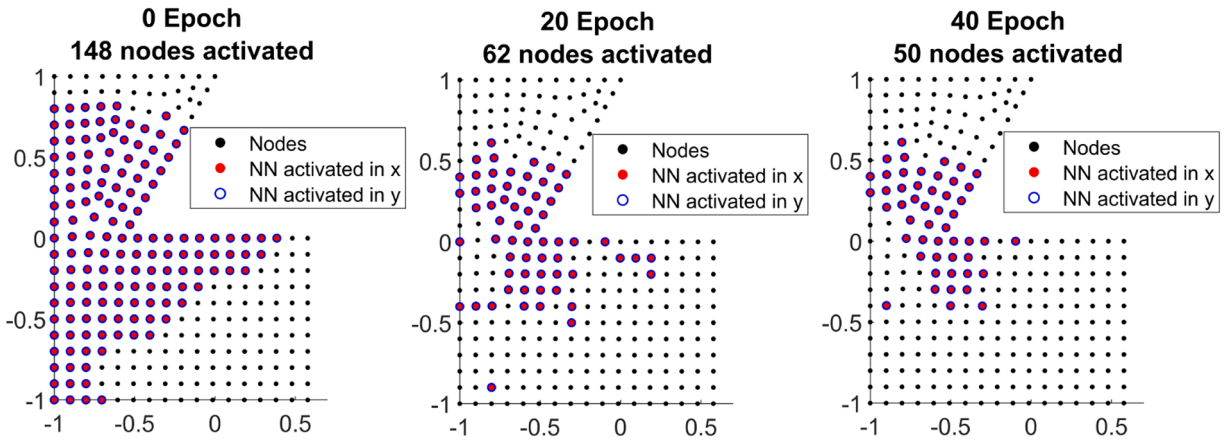


Fig. 28. Adaptive selection of the nodes for NN enrichment.

Fig. 29 demonstrates the loss function reduction during the optimization process with and without a transfer learning from pre-trained "parent" NN enrichment basis set. Table 5 compares the epochs and CPU time required for NN-PU solution computation with and without transfer learning to yield the potential energy levels of FEM approximations with different domain discretization. The results show that utilizing transfer learned NN Basis sub-block weight set notably accelerates the optimization process compared to the case without transfer learning where random number initialization of NN Basis sub-block weights and biases were considered. Fig. 30 presents the stress solutions obtained using three different approaches: RKPM with 273 nodes, NN-PU approximation with the transfer-learned NN enrichment bases, and Q8 FEM with 447,602 nodes. While RKPM without NN enrichment exhibits oscillations near the corner due to the employment of SCNI [58] without stabilization, NN-PU effectively mitigates these oscillations without any

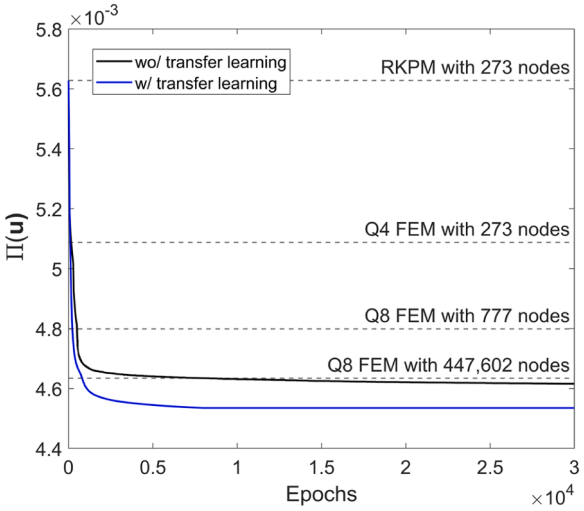
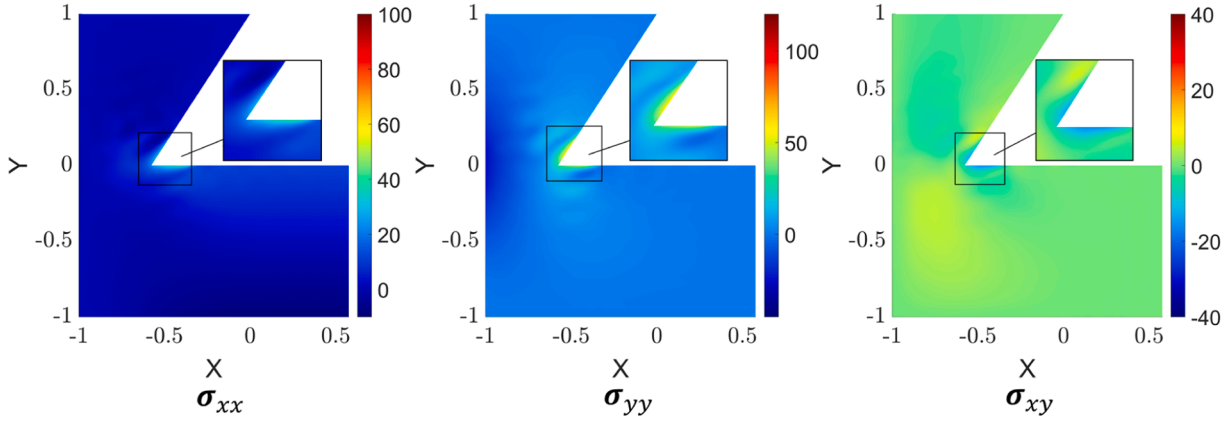


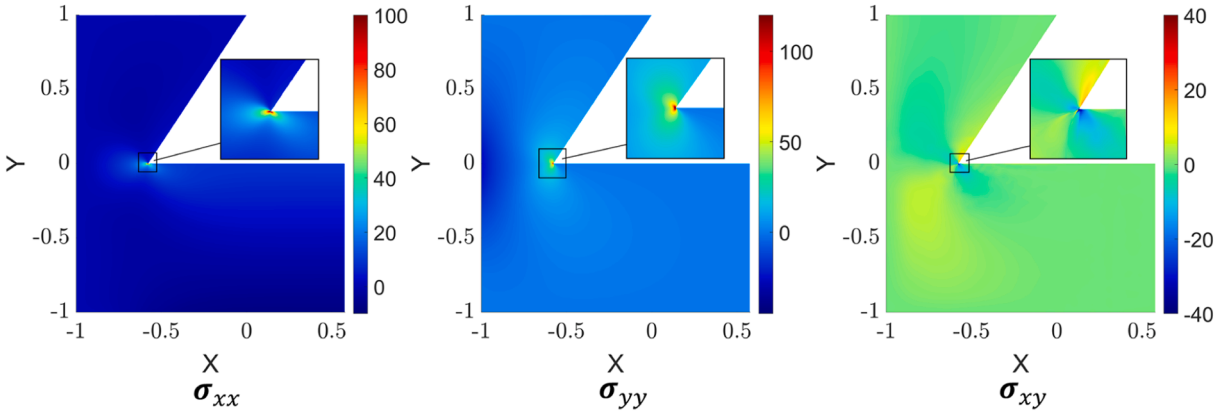
Fig. 29. Loss function minimization for NN-PU with and without transfer learning and the comparison with FEM potential energies.

Table 5
Comparison of the required numbers of epochs and runtimes for NN-PU solution to reach the potential energy (PE) of FEM solutions with different discretization with and without weights transfer learned from the parent NN basis set for the 60° corner problem.

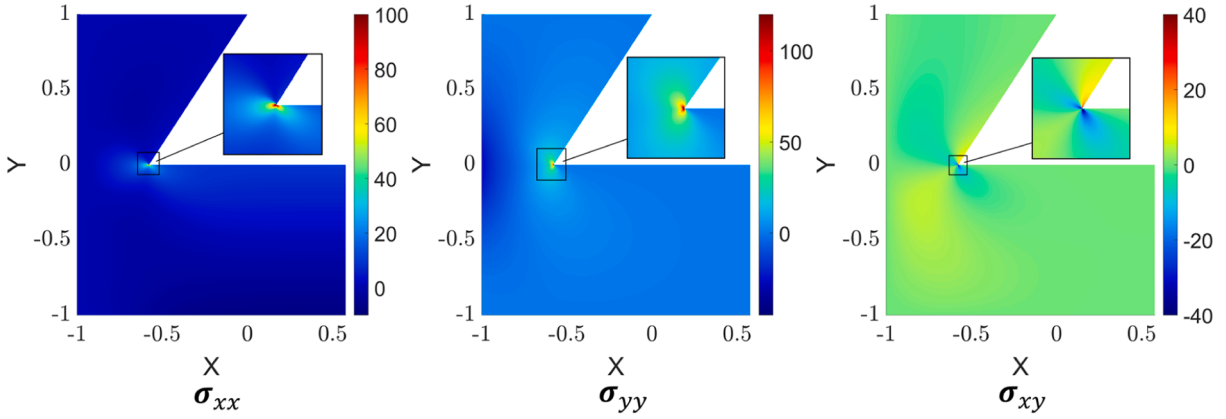
NN-PU with potential energy (PE) lower than FEM PE		Q4 FEM with 273 nodes (PE = 5.088E-03, CPU = 1.3 s)	Q8 FEM with 777 nodes (PE = 4.799E-03, CPU = 1.9 s)	Q8 FEM with 447,602 nodes (PE = 4.634E-03, CPU = 32.9 s)
Number of Epochs to reach the accuracy	w/ transfer learning	88	201	802
	wo/ transfer learning	160	501	8018
CPU for NN-PU (s)	w/ transfer learning	4.5	6.9	20.2
	wo/ transfer learning	6.0	13.5	178.9



(a) RKPM with 273 nodes



(b) NN-PU with 273 RK nodes and 1,022 NN unknowns



(c) Q8 FEM with 447,602 nodes

Fig. 30. Comparison of the stress solutions: (a) RKPM solutions with 273 nodes (b) NN-PU solutions with 273 RK nodes and 1022 NN unknowns and (c) Q8 FEM solutions with 447,602 nodes.

stabilization. Compared to the high-order much refined FEM solution with nearly a million degrees of freedom, NN-PU with the transfer-learned NN-enrichment basis achieves a similar accuracy with 61.4 % of computation time. The final converged potential energy for the NN-PU online solution is 4.5350E-03, leading to a 2.14 % potential energy reduction (better accuracy) compared to the

one from the Q8 FEM solution with 447,602 nodes. The results demonstrate that the proposed NN-PU with transfer learning from parent problems with different features becomes more effective than FEM when higher solution accuracy is in demand.

5.3. Transfer learning with multiple local features

5.3.1. Plate with two circular holes: transfer learned from the plate with a hole parent problem

In this example, we consider a plate with two identical circular holes subjected to an x-directional uniaxial tension, as illustrated in Fig. 31 and Fig. 32, where the only difference between the two lies in the radius of the holes. Given such a problem setting, the same localized stress pattern is anticipated to be present around both circular holes. Two NN blocks are utilized for this problem for both plate geometries, and two Parametric sub-blocks are employed, denoted as $\mathcal{N}_1^P$ and $\mathcal{N}_2^P$. Both Parametric sub-blocks use the same architecture as mentioned in Section 5.2.1. The NN Basis sub-blocks $\mathcal{N}_1^\zeta$ and $\mathcal{N}_2^\zeta$ incorporate the same learned NN-enrichment basis functions from the plate-with-a-hole "parent" problem discussed in Section 5.1.1. An NN basis sub-block consisting of 4 hidden layers, with the first 3 layers comprising 50 neurons each and incorporating 5 NN-enrichment bases in the last hidden layer learned from the "parent" problem in Section 5.1.1 is adapted, and the offline training time is 1.5 s for the selected NN bases architecture. Note that all weights and biases in $\mathcal{N}_1^\zeta$ and $\mathcal{N}_2^\zeta$ are directly transferred from those in the "parent" problem, and hence only the weight sets of Parametric sub-blocks ($\mathbb{W}^P$) and the coefficients of the NN basis functions ($\mathbb{W}^C$) are iterated during the loss function minimization.

Note that every neural network block is associated with its unique NN-enriched nodeset, referred as $\mathcal{S}_1^\zeta$ and $\mathcal{S}_2^\zeta$, respectively, which are initialized through an error estimator as follows:

$$\mathcal{S}_1^\zeta = \{I | x_{1I} < 0,\ \rho_I^* > 0.1\rho_{\max}^*\},$$

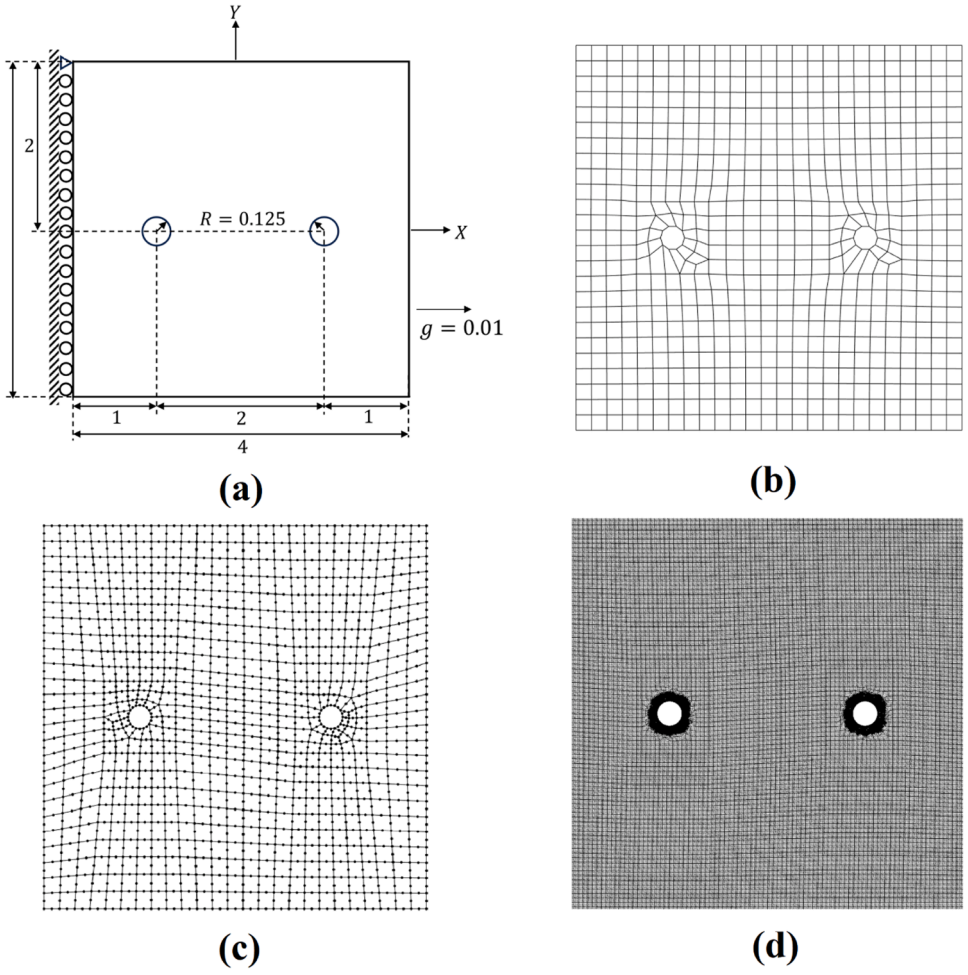


Fig. 31. (a) Problem settings of plates with two small circular holes and 3 levels of FE discretization: (b) Q4 FEM with 688 nodes (c) Q8 FEM with 2015 nodes (d) Q8 FEM with 499,843 nodes.

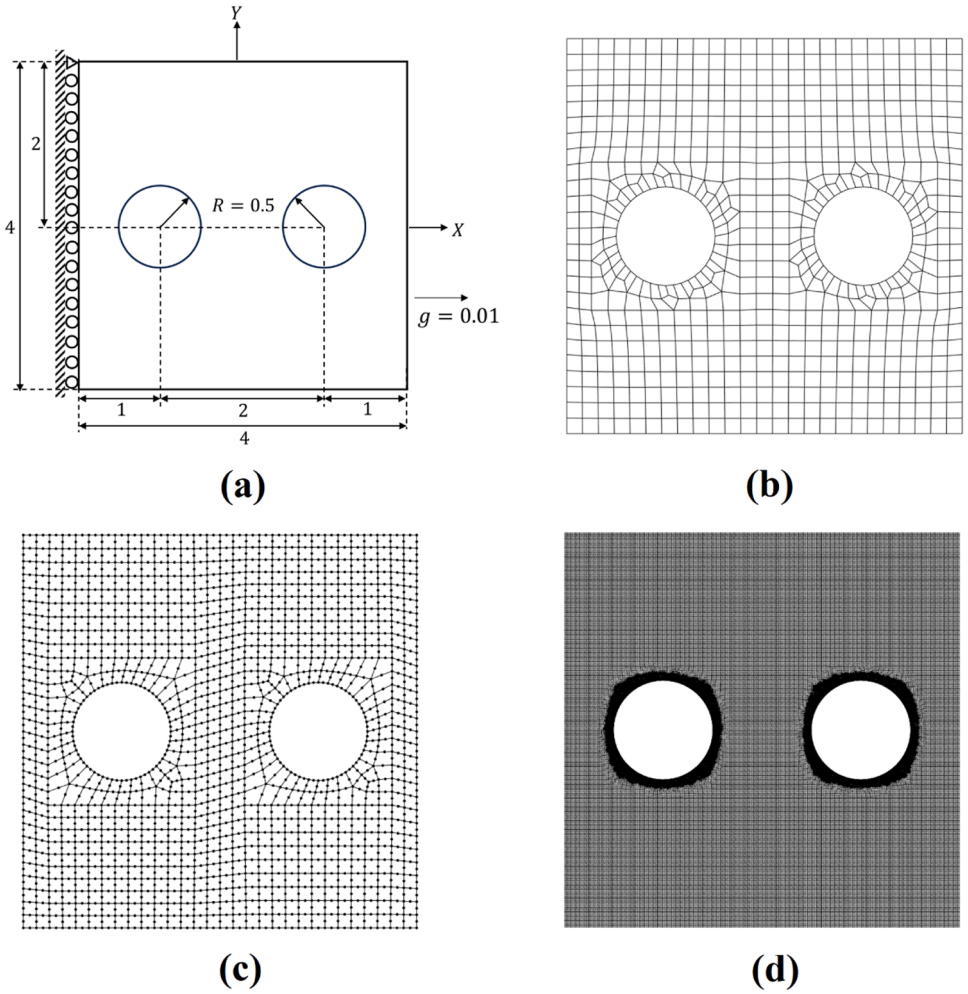


Fig. 32. (a) Problem settings of plates with two large circular holes and 3 levels of FE discretization: (b) Q4 FEM with 728 nodes (c) Q8 FEM with 2625 nodes (d) Q8 FEM with 504,452 nodes.

$$\mathscr{S}_2^{\zeta} = \{I | x_{1I} \geq 0,\ \rho_I^* > 0.1\rho_{max}^*\}, \tag{40}$$

where x_{1I} is the x directional coordinate of a node I. Similar to previous problems, the nodal energy error densities are re-evaluated every 20 epochs, and both $\mathscr{S}_1^{\zeta}$ and $\mathscr{S}_2^{\zeta}$ are updated at the online computation. Fig. 33 and Fig. 34 demonstrate the evolutions of $\mathscr{S}_1^{\zeta}$ and $\mathscr{S}_2^{\zeta}$ for both cases, and the numbers of NN-enriched nodes are settled after three updates.

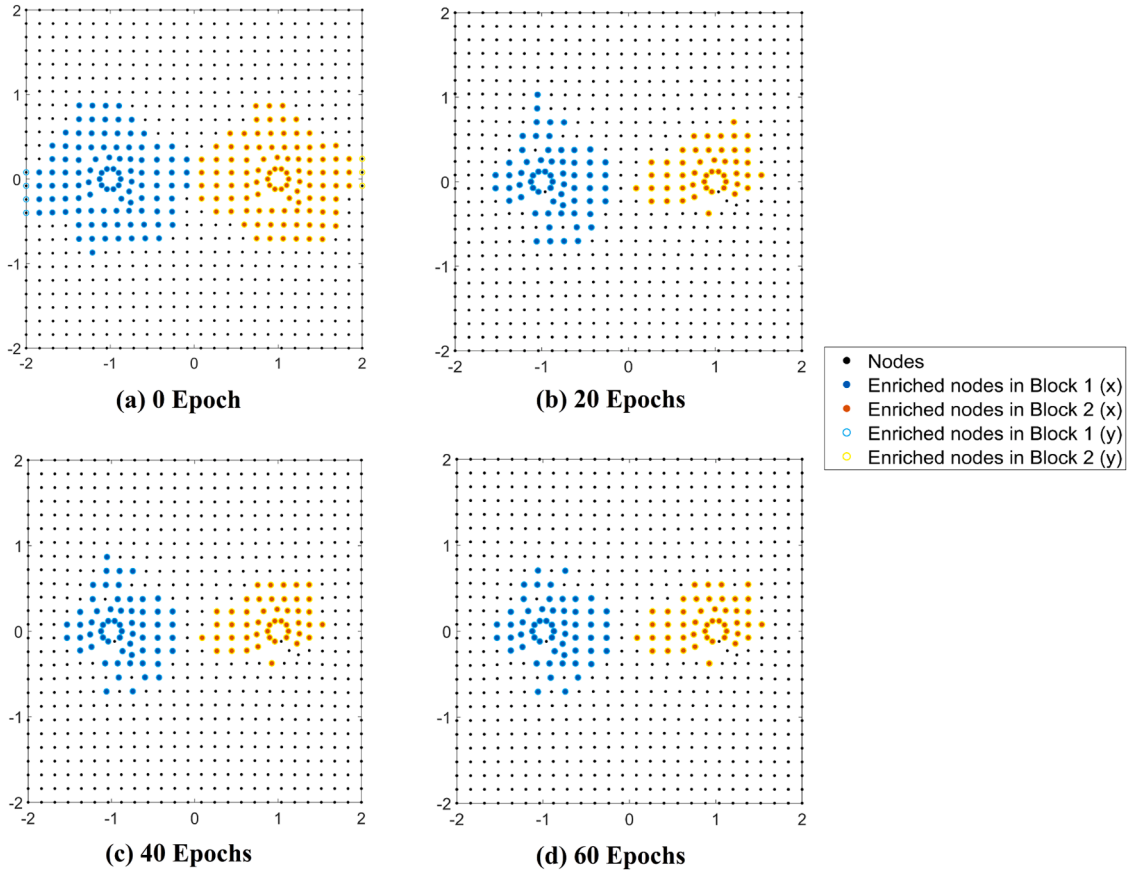


Fig. 33. Adaptive selection of the NN-enriched nodes for the problem with small circular holes: (a) total 206 and 213 enriched nodes in x- and y-directions (b) total 133 enriched nodes in both directions (c) total 109 enriched nodes in both directions and (d) total 105 enriched nodes in both directions.

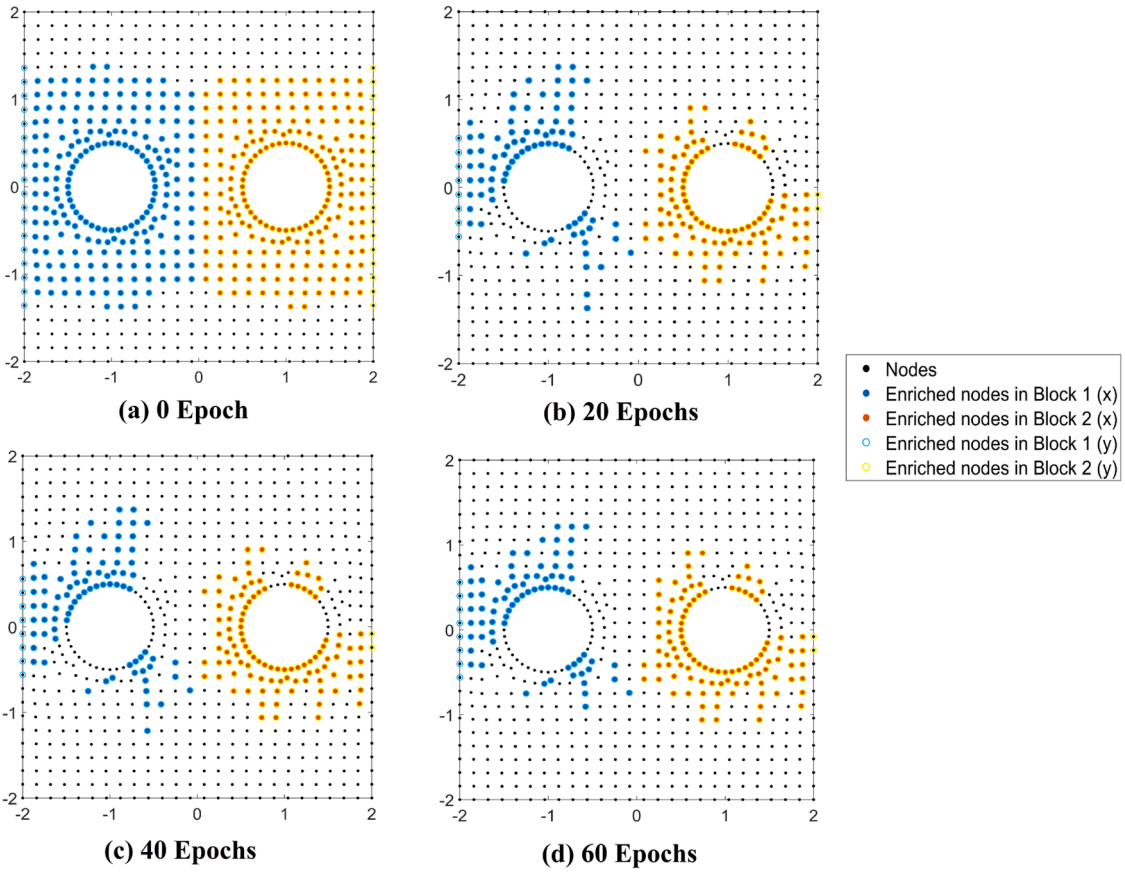


Fig. 34. Adaptive selection of the NN-enriched nodes for the problem with large circular holes: (a) total 436 and 472 enriched nodes in x- and y-directions (b) total 188 and 198 enriched nodes in x- and y-directions (c) total 187 and 197 enriched nodes in x- and y-directions and (d) total 180 and 190 enriched nodes in x- and y-directions.

Fig. 35 illustrates the process of loss function minimization for both plate geometries. When examining a plate with larger holes, as seen Fig. 35(b), it becomes evident that the optimization process leads to faster convergence due to smaller interactions between the local solutions near two holes. Table 6 reports the results from NN-PU and FEM with different levels of domain discretization and

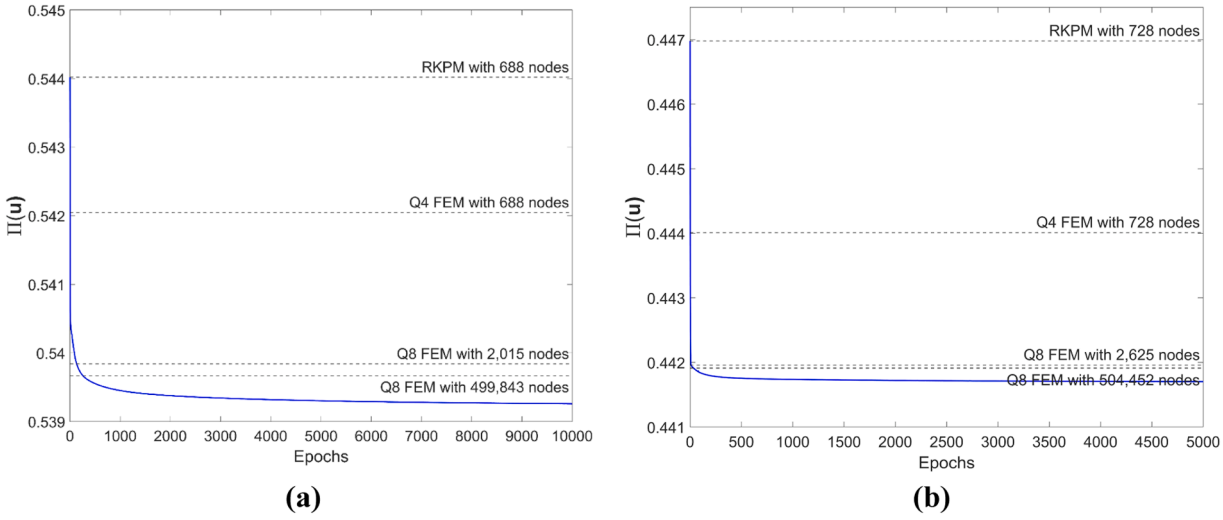


Fig. 35. Loss function minimization and the comparison with FEM potential energies (a): plate with two small circular holes (b) plate with two large holes.

Table 6
Comparison between the NN-PU solution obtained with known "parent" basis set and FEM solutions with different spatial discretization for the plate with two holes problems.

Two small holes with $R = 0.125$			
NN-PU with potential energy (PE) lower than FEM PE	Q4 FEM with 688 nodes (PE = 5.421E-01, CPU=1.8 s)	Q8 FEM with 2015 nodes (PE = 5.398E-01, CPU= 2.1 s)	Q8 FEM with 499,843 nodes (PE = 5.397E-01, CPU=30.3 s)
Number of Epochs NN-PU used to reach the FEM PE levels	3	135	257
CPU for NN-PU to reach FEM PE levels (s)	2.7	6.6	10.3
Two large holes with $R = 0.5$			
NN-PU with potential energy (PE) lower than FEM PE	Q4 FEM with 728 nodes (PE = 4.440E-01, CPU=1.9 s)	Q8 FEM with 2625 nodes (PE = 4.420E-01, CPU=2.2 s)	Q8 FEM with 504,452 nodes (PE = 4.419E-01, CPU=31.9 s)
Number of Epochs NN-PU used to reach the FEM PE levels	2	11	35
CPU for NN-PU to reach the FEM PE levels (s)	3.2	3.6	4.4

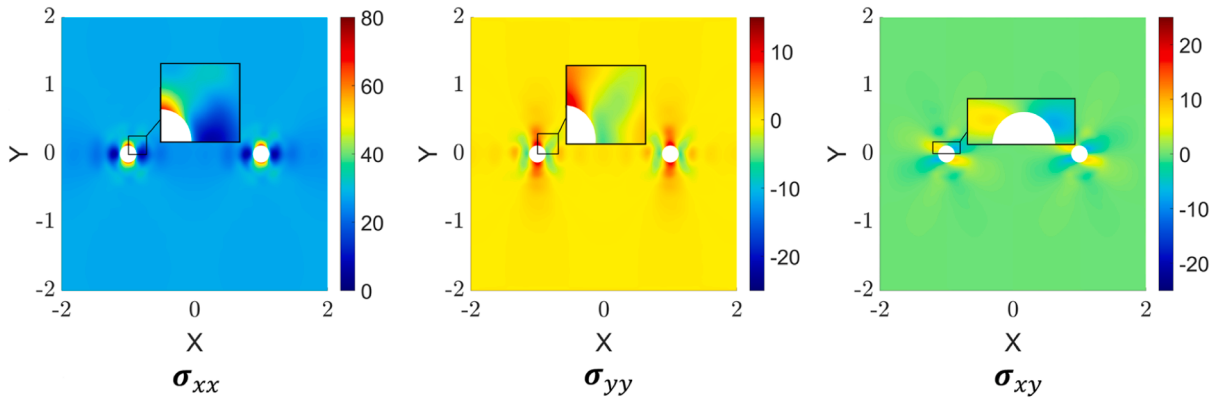
Fig. 36 and Fig. 37 compare the stress results for plates with small holes and larger holes problems, respectively, obtained using coarse RKPM, NN-PU approximation, and high-order highly refined FEM with body-fitted meshes. The results indicate that when employing the "parent" NN-enriched enrichment basis set, NN-PU with less than 1 % of the degrees of freedom can achieve similar accuracy to FEM with substantial CPU reduction in the online calculation, as the computed normalized energy error norms (Eq. (39)) compared to the fine Q8 FEM solutions for the small holes and larger holes problems are 0.82 % and 0.51 %, respectively. This efficiency is particularly evident in the case of larger holes, where the runtime is 1:7.25 compared to that of the finest FEM model. The final converged total potential energy is 5.3921E-01 for the plate with smaller holes and 4.4167E-01 for the plate with larger holes, leading to potential energy reductions of 0.08 % and 0.05 %, respectively, compared to the corresponding values computed using Q8 FEM with much refined meshes. The corresponding CPU times are 34.0 % and 13.8 % of the corresponding FEM computational time, respectively.

5.3.2. Plate with multiple holes: transfer learned from the parent problems with single local features

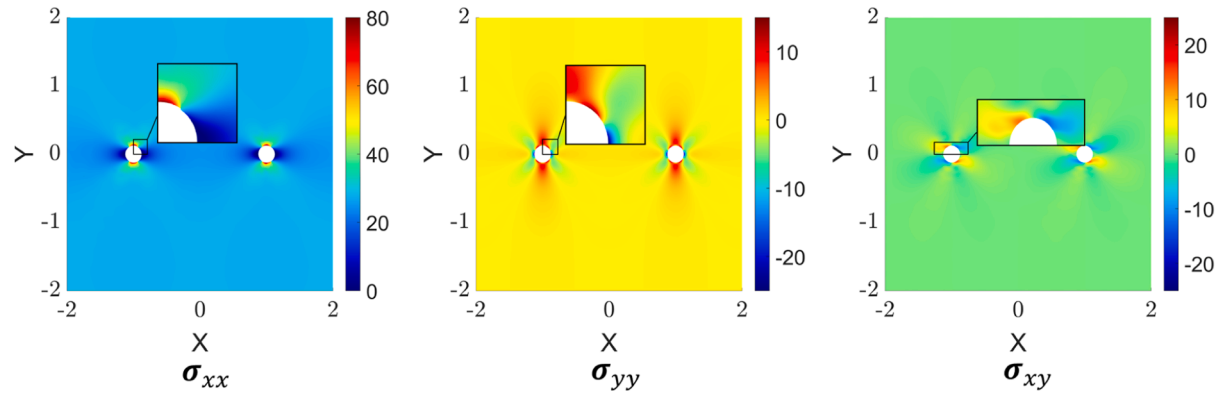
The last numerical example deals with a plate with multiple holes undergoing uniaxial tensile loading in the x-direction, as depicted in Fig. 38. In this case, 4 NN blocks are employed, each dedicated to capturing the localized characteristics around each hole. For these 4 NN blocks, 4 corresponding Parametric sub-blocks $\mathscr{N}^{\mathcal{P}}_1, \mathscr{N}^{\mathcal{P}}_2, \mathscr{N}^{\mathcal{P}}_3$, and $\mathscr{N}^{\mathcal{P}}_4$ are employed. All Parametric sub-blocks except the first one associated with the triangular hole utilize the same 2-layered architecture as that used in Section 5.2.1, and $\mathscr{N}^{\mathcal{P}}_1$ adopts the same architecture as the one in Section 5.2.2. The NN-enrichment basis sets for each hole are taken from their corresponding "parent" problems in Section 5.1 with transfer learning. The NN Basis sub-block $\mathscr{N}^{\zeta}_1$, corresponding to the first NN block and associated with the triangular hole, are transfer-learned from the 90° "parent" L-shaped panel problem discussed in Section 5.1.2. Similarly, the NN Basis sub-blocks $\mathscr{N}^{\zeta}_2$ and $\mathscr{N}^{\zeta}_3$, corresponding to the two circular holes, are transfer-learned from the plate-with-a-hole "parent" problem in Section 5.1.1. The NN Basis sub-block $\mathscr{N}^{\zeta}_4$ corresponding to the square hole employs an eight-layered architecture with 20 neurons in the first 7 hidden layers and 5 NN-enrichment bases, which are learned from the 90° L-shaped panel "parent" problem discussed in Section 5.1.2. During the online calculation, all weights and biases in $\mathscr{N}^{\zeta}_1, \mathscr{N}^{\zeta}_2, \mathscr{N}^{\zeta}_3$, and $\mathscr{N}^{\zeta}_4$ are directly transferred from parent NN-enrichment basis sets, learned from their respective "parent" problems. The offline training duration amounts to 6.3 s for the parent NN basis sets utilized in $\mathscr{N}^{\zeta}_1$ and $\mathscr{N}^{\zeta}_4$, and 1.5 s for the one adopted in $\mathscr{N}^{\zeta}_2$ and $\mathscr{N}^{\zeta}_3$. Consequently, the loss function minimization process only recomputes the optimal parametric coordinates and NN-enrichment coefficients within each NN block.

The NN-enriched nodesets for the 4 NN blocks are initiated as follows:

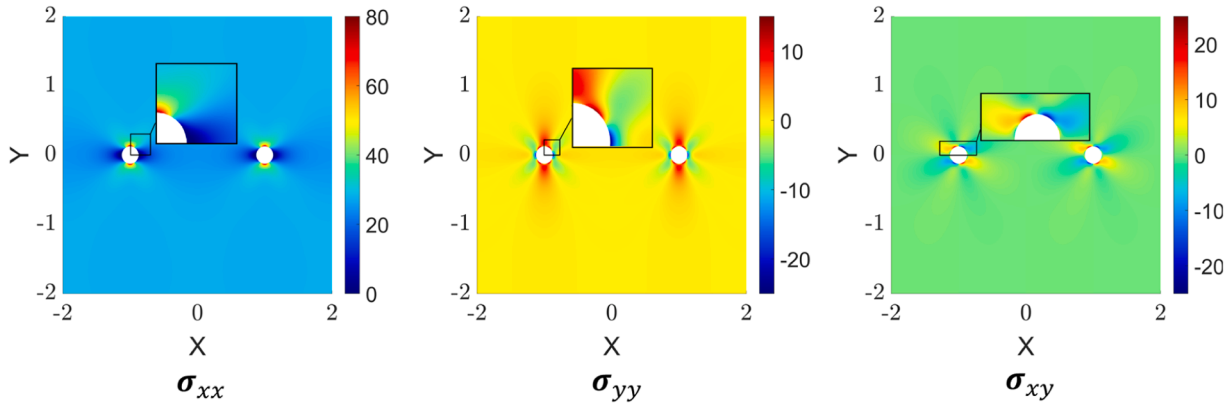
$$\begin{aligned}
\mathscr{S}^{\zeta}_1 &= \left\{I \middle| x_{1I} < 0,\ x_{2I} > 0,\ \rho^*_I > 0.1\rho^*_{\max}\right\},\\
\mathscr{S}^{\zeta}_2 &= \left\{I \middle| x_{1I} \geq 0,\ x_{2I} > 0, \rho^*_I > 0.1\rho^*_{max}\right\},\\
\mathscr{S}^{\zeta}_3 &= \left\{I \middle| x_{1I} < 0,\ x_{2I} \leq 0,\ \rho^*_I > 0.1\rho^*_{\max}\right\},\\
\mathscr{S}^{\zeta}_4 &= \left\{I \middle| x_{1I} \geq 0,\ x_{2I} \leq 0, \rho^*_I > 0.1\rho^*_{max}\right\},
\end{aligned} \tag{41}$$



(a) RKPM with 688 nodes

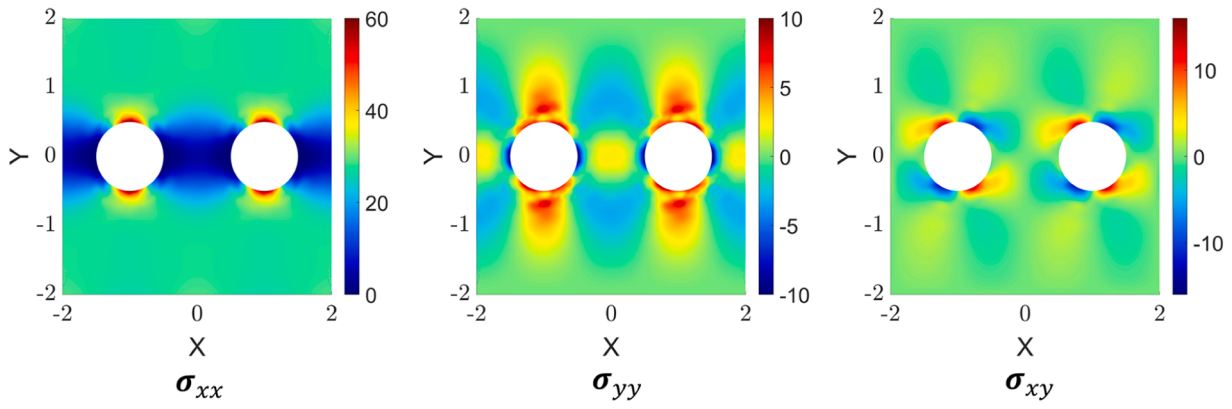


(b) NN-PU with 688 RK nodes and 1,104 NN unknowns

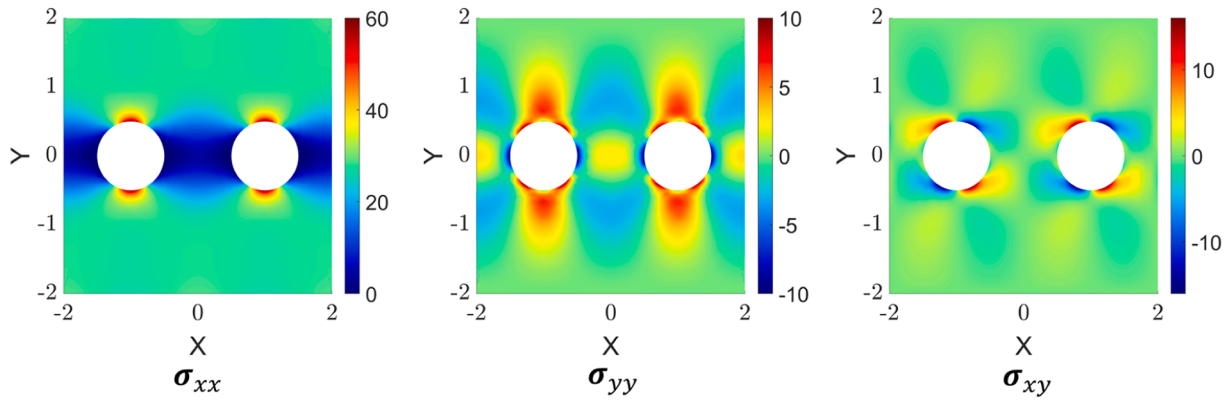


(c) Q8 FEM with 499,843 nodes

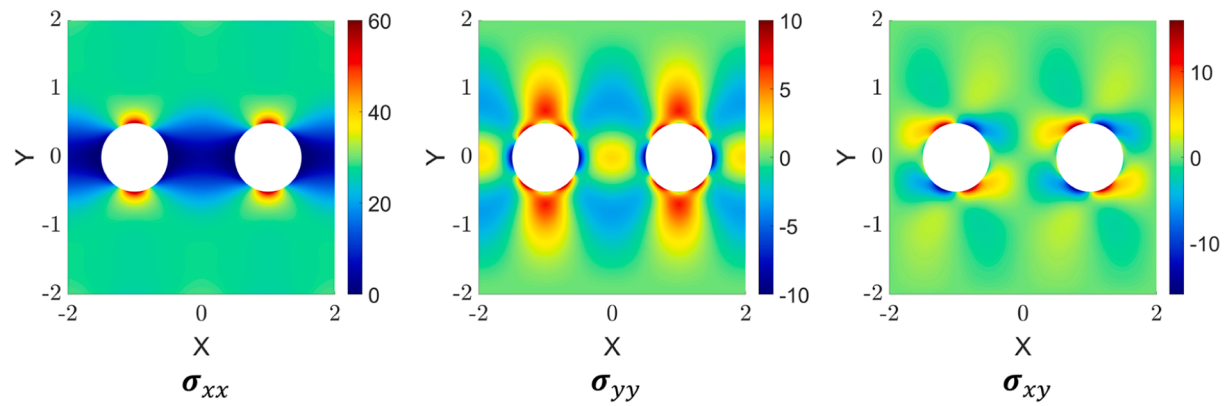
Fig. 36. Comparison of the stress solutions for the plate with two small holes: (a) RKPM solutions with 688 nodes (b) NN-PU solutions with 688 RK nodes and 1104 NN unknowns and (c) Q8 FEM solutions with 499,843 nodes.



(a) RKPM with 728 nodes



(b) NN-PU with 728 RK nodes and 1,854 NN unknowns



(c) Q8 FEM with 504,452 nodes

Fig. 37. Comparison of the stress solutions for the plate with two big holes: (a) RKPM solutions with 728 nodes (b) NN-PU solutions with 728 RK nodes and 1854 NN unknowns and (c) Q8 FEM solutions with 504,452 nodes.

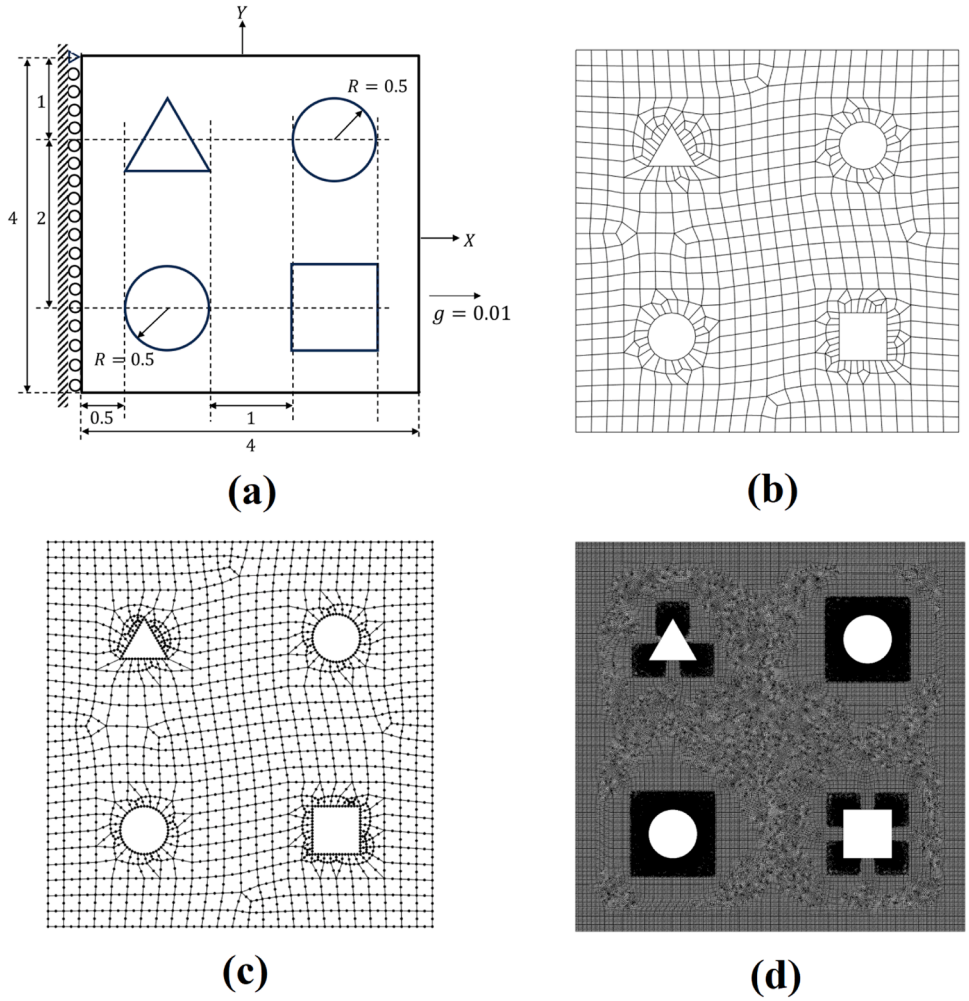


Fig. 38. (a) Problem setting for the plate with multiple holes and 3 levels of FE discretization: (b) Q4 FEM with 768 nodes (c) Q8 FEM with 2208 nodes (d) Q8 FEM with 647,174 nodes.

where x_{1I} and x_{2I} denote the x and y directional coordinates of a node I, and the subscripts in the NN-enriched nodesets are associated with the indices of the NN blocks. The assessment of nodal energy error densities takes place every 20 epochs, and the 4 enrichment node subsets are updated independently, as demonstrated in Fig. 39, resulting in a total of 2990 trainable NN-enrichment coefficients.

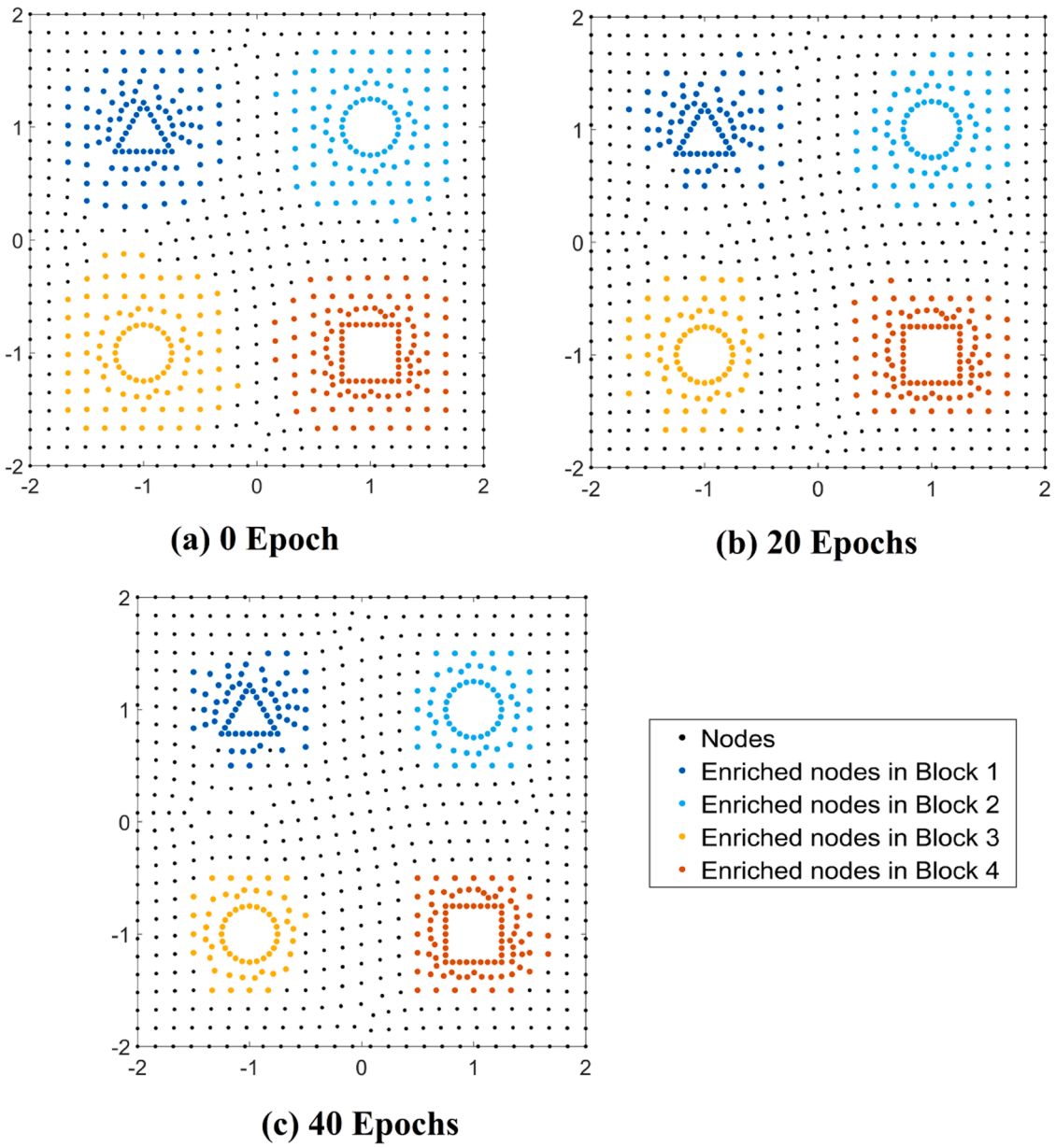


Fig. 39. Adaptive selections of the NN-enriched nodes for 4 NN blocks: (a) total 442 nodes are enriched (b) total 351 nodes are enriched and (c) total 299 nodes are enriched (numbers of enriched nodes in x- and y- directions are the same).

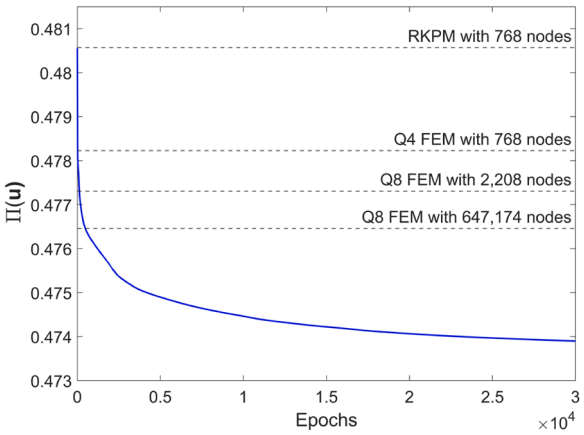


Fig. 40. Loss function minimization for NN-PU and the comparison with FEM potential energy.

The loss function minimization in the online calculation is illustrated in Fig. 40. Table 7 reports a performance between NN-PU and FEM methods with varying degrees of domain discretization.

The stress results comparison between RKPM, NN-PU, and the high-order highly refined FEM with over a million degrees of freedom is demonstrated in Fig. 41. The NN-PU approximation with transfer-learned NN-enrichment basis sets is able to achieve solutions with a normalized energy error norm (Eq. (39)) of 1.67 % compared to the much-refined higher-order FEM solutions with 50.4 % CPU reduction. The online calculation of NN-PU yields a converged potential energy of 4.7391E-01, resulting in a 0.5 % potential energy reduction compared to that computed from Q8 FEM solutions with 647,174 nodes. Note that other than utilizing the NN enrichment basis sets pre-trained from "parent" problems in Section 5.1, $\mathscr{N}_1^{\zeta}$ can be transfer learned from the converged NN bases in the 60° L-shaped panel problem in Section 5.2.2. In this way, $\mathscr{N}_1^{P}$ can take the same 2-layered architecture as the other NN blocks, totaling of 3098 NN unknowns and leading to a larger CPU reduction (61.5 %) compared to the much-refined FEM. This study demonstrates that the NN-PU approach with transfer learning from parent problems can be more effective than the FEM approach, especially when repetitive local features are involved in the problems.

Table 7
Comparison between the NN-PU solution obtained with known NN enrichment basis sets and FEM solutions with different spatial discretization for the plate with four holes problem.

NN-PU with potential energy (PE) lower than FEM PE	Q4 FEM with 768 nodes (PE = 4.782E-01, CPU=1.9 s)	Q8 FEM with 2208 nodes (PE = 4.773E-01, CPU=2.2 s)	Q8 FEM with 647,174 nodes (PE = 4.765E-01, CPU=45.2 s)
Number of Epochs NN-PU used to reach the FEM PE levels	17	120	480
CPU (s) for NN-PU to reach the FEM PE levels	3.9	8.1	22.4

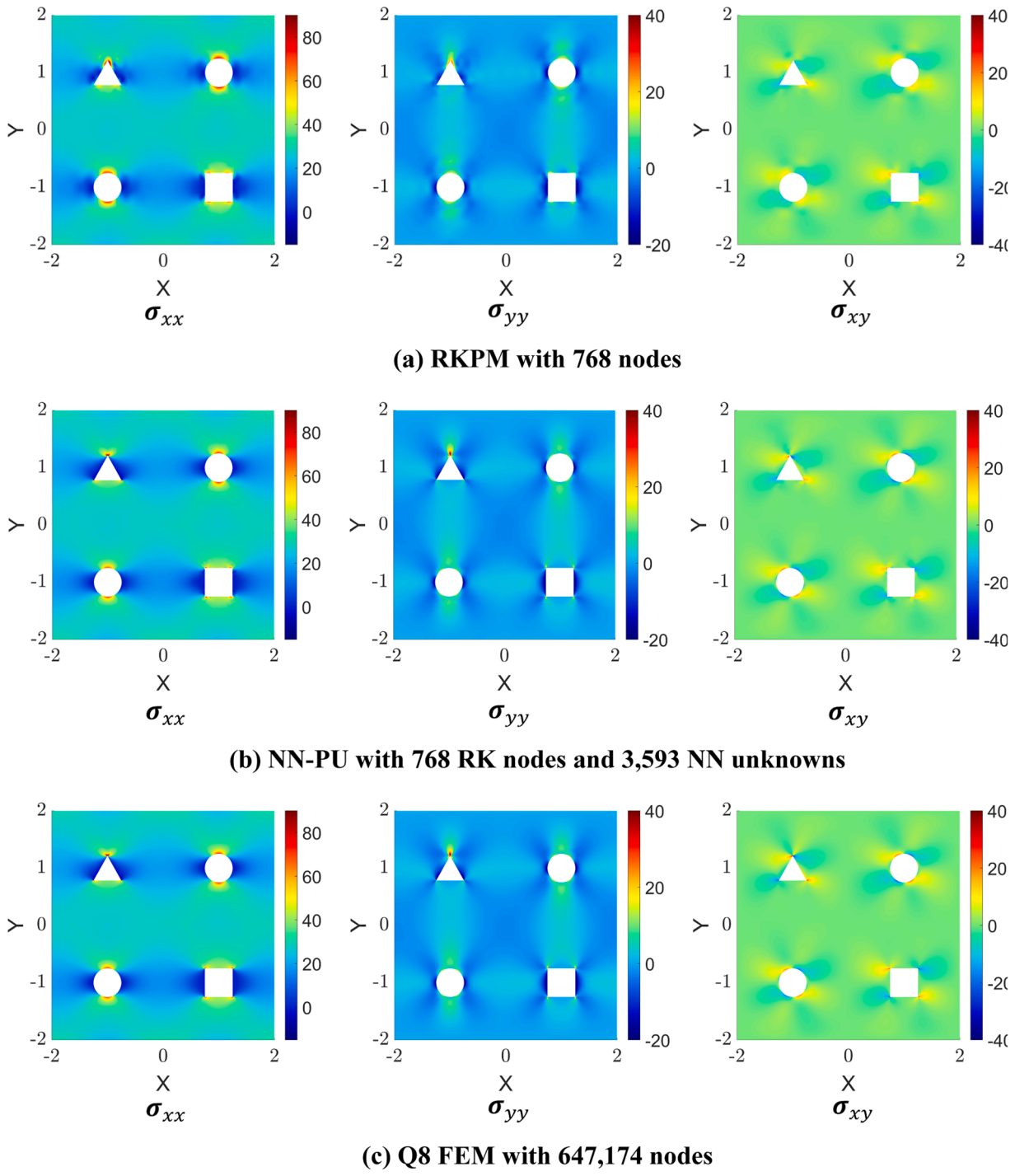


Fig. 41. Comparison of the stress solutions: (a) RKPM solutions with 768 nodes (b) NN-PU solutions with 768 RK nodes and 3593 NN unknowns and (c) Q8 FEM solutions with 647,174 nodes.

6. Conclusion

A neural network-enriched Partition of Unity (NN-PU) method has been proposed for solving boundary value problems. In this approach, the domain geometry, boundary conditions, and loading conditions are discretized by the background approximation that possesses partition of unity properties. The background approximation with a fixed coarse discretization is enriched adaptively by feature-encoded local NN basis functions trained at the offline stage via artificial neural networks with the minimization of a potential

energy-based loss function. The transferable block-level NN architecture used for the NN approximation keeps high sparsity of the network structure and offers flexibility for integrating the pre-trained feature-encoded NN enrichment basis sets with the background PU approximation, facilitating a more efficient online solution strategy. This block neural network has the flexibility to incorporate multiple feature-encoded NN bases targeting different localized features in the solution. The proposed NN-PU method, coupled with feature-encoded transfer learning, provides an effective adaptivity framework for solving PDEs. Error analysis for the proposed NN-PU with reproducing kernel as the background PU approximation shows how the convergence is determined by both background PU approximation as well as the NN enrichment.

Employing a coarse background discretization, the method utilizes extrinsic NN-based enrichment functions to enhance the background approximation within the Partition of Unity framework. To improve the efficiency and adaptivity of the NN-PU approach, a block-level NN approximation is introduced, where each NN block is designed to target a specific underlying feature in the problem's physics. The architecture of the block-level NN approximation is designed with three sub-blocks: (1) the Parametric sub-block to facilitate the utilization of pre-trained NN bases in a parametric coordinate to represent functions with local features, (2) the NN Basis sub-block taking the parametric coordinates to generate NN enrichment basis functions and to embed the trained NN bases to the background approximation functions, (3) the Coefficient sub-block to determine coefficients for linear combination of NN enrichment bases by optimizing an energy-based loss function and add the contribution of each NN enrichment basis to the total solution of the problem. During offline training, multiple NN enrichment basis sets are pre-trained for different "parent" problems to embed various underlying solution features into the NN basis functions. These feature-encoded NN basis sets are then utilized in online computation stages through transfer learning to capture complex solution patterns efficiently.

Parametric studies on the effects of different NN architectures in minimizing NN loss functions show that employing deeper and wider neural networks (up to 50 neurons in hidden layers) and increasing the number of NN bases (up to 5) significantly improve the speed of offline training of NN enrichment basis sets. An error indicator is introduced to identify the initial enrichment nodes, and the enrichment nodes are continuously updated during the loss function minimization iteration to facilitate the adaptive NN enrichment. The effectiveness of the NN-PU method in solving elasticity problems with local features has been substantiated through numerical examples. Comparison with FEM solutions obtained via Galerkin approximation has been made to examine the effectiveness of the proposed method. Numerical error analysis of a "parent" plate-with-a-hole problem shows good agreement on the analytically derived convergence rates. Assessments conducted during online solution calculations for problems featuring single or multiple local features indicate that the NN-PU method, incorporating feature-encoded transfer learning with pre-trained NN-enrichment basis sets, achieves comparable solution accuracy to highly refined FEM with considerably reduced unknowns and CPU time. This NN-PU framework can be extended to inelastic problems by a proper choice of energy function.

While the present work offers an alternative approach for constructing NN-based enrichment functions as a local enrichment of Galerkin solution, mathematically sound strategies, such as those proposed for optimal refinement in FEM that automatically select optimal h-refinements and distribution of order of approximation [65,66], have not been fully investigated and thus not yet robust in the present work for the optimal selection of number of layers, neurons, activation functions, etc. This is a motivation for future study as an extension of the present work.

CRediT authorship contribution statement

Jonghyuk Baek: Writing – review & editing, Writing – original draft, Software, Resources, Methodology, Investigation, Formal analysis, Conceptualization. **Yanran Wang:** Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis. **Jiun-Shyan Chen:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

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

No data was used for the research described in the article.

Acknowledgements

The support of this work by the Sandia National Laboratories to UC San Diego under Contract Agreement 1655264 is greatly acknowledged. Sandia National Laboratories is a multi-mission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. The support from National Science Foundation under Award Number CMMI-1826221 to University of California, San Diego is greatly appreciated.

Appendix. Construction of the reproducing kernel (RK) shape function

Let a closed domain $\overline{\Omega} = \Omega \cup \partial\Omega \subset \mathbb{R}^d$ be discretized by NP nodes with a node set $\mathscr{S} = \{1,\ 2,\ 3, \ldots, NP\}$ with nodal coordinates $\{\boldsymbol{x}_I | \boldsymbol{x}_I \in \overline{\Omega}\}_{I \in \mathscr{S}}$, as shown in Fig. 42. The RK shape function is constructed by the product of a kernel function $\phi_a(\boldsymbol{x} - \boldsymbol{x}_I)$ with a compact support a and a correction function $C(\boldsymbol{x}; \boldsymbol{x} - \boldsymbol{x}_I)$ as:

$$\Psi_I(\boldsymbol{x}) = C(\boldsymbol{x}; \boldsymbol{x} - \boldsymbol{x}_I)\phi_a(\boldsymbol{x} - \boldsymbol{x}_I). \tag{42}$$

The correction function is composed of a linear combination of n^{th} order monomial basis functions:

$$C(\boldsymbol{x}; \boldsymbol{x} - \boldsymbol{x}_I) \equiv \sum_{|\alpha| \le n} (\boldsymbol{x} - \boldsymbol{x}_I)^{\alpha} b_{\alpha}(\boldsymbol{x}) \equiv \boldsymbol{H}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_I)\boldsymbol{b}(\boldsymbol{x}),$$

$$\boldsymbol{H}^{T}(\boldsymbol{x} - \boldsymbol{x}_I) = [1,\ x_1 - x_{1I},\ x_2 - x_{2I},\ \ldots,\ (x_3 - x_{3I})^n], \tag{43}$$

where α is a multi-index notation such that $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ with a length defined as $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_d$, and $\boldsymbol{x}^{\alpha} \equiv \boldsymbol{x}_1^{\alpha_1} \cdot \boldsymbol{x}_2^{\alpha_2}, \ldots, \boldsymbol{x}_d^{\alpha_d}$, $b_{\alpha} = b_{\alpha_1 \alpha_2 \cdots \alpha_d}$. The terms $\{(\boldsymbol{x} - \boldsymbol{x}_I)^{\alpha}\}_{|\alpha| \le n}$ form a set of basis functions, and $\boldsymbol{H}^{T}(\boldsymbol{x} - \boldsymbol{x}_I)$ is the corresponding vector of basis functions to the order n. The correction function is designed to introduce completeness to the RK approximation, and the coefficient vector $\boldsymbol{b}(\boldsymbol{x})$ is solved by enforcing the following discrete reproducing conditions [67]:

$$\sum_{I \in S} \Psi_I(\boldsymbol{x})(\boldsymbol{x} - \boldsymbol{x}_I)^{\alpha} = \delta_{0\alpha}, \quad |\alpha| \le n$$

or

$$\sum_{I \in S} \Psi_I(\boldsymbol{x})\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_I) = \boldsymbol{H}(\boldsymbol{0}), \tag{44}$$

where $\boldsymbol{H}(\boldsymbol{0}) = [1,\ 0,\ \cdots,\ 0]^{T}$. The final RK shape function takes the following form:

$$\Psi_I(\boldsymbol{x}) = \boldsymbol{H}^{\mathrm{T}}(\boldsymbol{0})\boldsymbol{M}^{-1}(\boldsymbol{x})\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_I)\phi_a(\boldsymbol{x} - \boldsymbol{x}_I), \tag{45}$$

where $\boldsymbol{M}(\boldsymbol{x})$ is the moment matrix and is formulated as:

$$\boldsymbol{M}(\boldsymbol{x}) = \sum_{I \in S} \boldsymbol{H}(\boldsymbol{x} - \boldsymbol{x}_I)\boldsymbol{H}^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{x}_I)\phi_a(\boldsymbol{x} - \boldsymbol{x}_I). \tag{46}$$

Note that the reproducing conditions in Eq. (44) are met provided $\boldsymbol{M}(\boldsymbol{x})$ is invertible, which requires at least $(n+d)!/(n!d!)$ non-planar nodes that cover $\boldsymbol{x}$ [68].

RK approximation has the ability to introduce high-order continuity into the approximation space, independent of the basis order, as the smoothness of the approximation functions is directly inherited from the smoothness of the kernel functions.

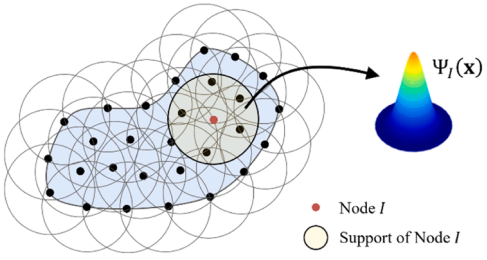


Fig. 42. Illustration of RK discretization and shape function.

References

[1] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Netw. 4 (2) (1991) 251–257, https://doi.org/10.1016/0893-6080(91)90009-T.

[2] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inf. Theory 39 (3) (1993) 930–945, https://doi.org/10.1109/18.256500.

[3] N.N. Vlassis, R. Ma, W.C. Sun, Geometric deep learning for computational mechanics Part I: anisotropic hyperelasticity, Comput. Methods Appl. Mech. Eng. 371 (2020) 113299, https://doi.org/10.1016/J.CMA.2020.113299.

[4] N.N. Vlassis, W.C. Sun, Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening, Comput. Methods Appl. Mech. Eng. 377 (2021) 113695, https://doi.org/10.1016/J.CMA.2021.113695.

[5] K. Xu, D.Z. Huang, E. Darve, Learning constitutive relations using symmetric positive definite neural networks, J. Comput. Phys. 428 (2021) 110072, https://doi.org/10.1016/J.JCP.2020.110072.
[6] X. He, J.S. Chen, Thermodynamically consistent machine-learned internal state variable approach for data-driven modeling of path-dependent materials, Comput. Methods Appl. Mech. Eng. 402 (2022) 115348, https://doi.org/10.1016/J.CMA.2022.115348.
[7] Y. He, S.J. Semnani, Machine learning based modeling of path-dependent materials for finite element analysis, Comput. Geotech. 156 (2023) 105254, https://doi.org/10.1016/J.COMPGEO.2023.105254.
[8] Z. Xiong, M. Xiao, N. Vlassis, W.C. Sun, A neural kernel method for capturing multiscale high-dimensional micromorphic plasticity of materials with internal structures, Comput. Methods Appl. Mech. Eng. 416 (2023) 116317, https://doi.org/10.1016/J.CMA.2023.116317.
[9] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Eng. 304 (2016) 81–101, https://doi.org/10.1016/J.CMA.2016.02.001.
[10] R. Ibañez, E. Abisset-Chavanne, J.V. Aguado, D. Gonzalez, E. Cueto, F. Chinesta, A manifold learning approach to data-driven computational elasticity and inelasticity, Arch. Comput. Methods Eng. 25 (1) (2018) 47–57, https://doi.org/10.1007/s11831-016-9197-9.
[11] R. Eggersmann, T. Kirchdoerfer, S. Reese, L. Stainier, M. Ortiz, Model-free data-driven inelasticity, Comput. Methods Appl. Mech. Eng. 350 (2019) 81–99, https://doi.org/10.1016/J.CMA.2019.02.016.
[12] Q. He, J.S. Chen, A physics-constrained data-driven approach based on locally convex reconstruction for noisy database, Comput. Methods Appl. Mech. Eng. 363 (2020) 112791, https://doi.org/10.1016/J.CMA.2019.112791.
[13] X. He, Q. He, J.S. Chen, U. Sinha, S. Sinha, Physics-constrained local convexity data-driven modeling of anisotropic nonlinear elastic solids, Data Centr. Eng 1 (5) (2020) e19, https://doi.org/10.1017/DCE.2020.20.
[14] X. He, Q. He, J.S. Chen, Deep autoencoders for physics-constrained data-driven nonlinear materials modeling, Comput. Methods Appl. Mech. Eng. 385 (2021) 114034, https://doi.org/10.1016/J.CMA.2021.114034.
[15] B. Bahmani, W.C. Sun, Distance-preserving manifold denoising for data-driven mechanics, Comput. Methods Appl. Mech. Eng. 405 (2023) 115857, https://doi.org/10.1016/J.CMA.2022.115857.
[16] Z. Liu, M.A. Bessa, W.K. Liu, Self-consistent clustering analysis: an efficient multi-scale scheme for inelastic heterogeneous materials, Comput. Methods Appl. Mech. Eng. 306 (2016) 319–341, https://doi.org/10.1016/J.CMA.2016.04.004.
[17] K. Wang, W.C. Sun, A multiscale multi-permeability poroplasticity model linked by recursive homogenizations and deep learning, Comput. Methods Appl. Mech. Eng. 334 (2018) 337–380, https://doi.org/10.1016/J.CMA.2018.01.036.
[18] X. He, K. Taneja, J.S. Chen, C.H. Lee, J. Hodgson, V. Malis, U. Sinha, S. Sinha, Multiscale modeling of passive material influences on deformation and force output of skeletal muscles, Int. J. Numer. Method. Biomed. Eng. 38 (4) (2022) e3571, https://doi.org/10.1002/CNM.3571.
[19] D. Bishara, Y. Xie, W.K. Liu, S. Li, A state-of-the-art review on machine learning-based multiscale modeling, simulation, homogenization and design of materials, Arch. Comput. Methods Eng. 30 (1) (2023) 191–222, https://doi.org/10.1007/S11831-022-09795-8.
[20] H. Wei, C.T. Wu, W. Hu, T.H. Su, H. Oura, M. Nishi, T. Naito, S. Chung, L. Shen, LS-DYNA machine learning–based multiscale method for nonlinear modeling of short fiber–reinforced composites, J. Eng. Mech. 149 (3) (2023) 04023003, https://doi.org/10.1061/JENMDT.EMENG-694.
[21] J. Baek, J.S. Chen, K. Susuki, A neural network-enhanced reproducing kernel particle method for modeling strain localization, Int. J. Numer. Methods Eng. 123 (18) (2022) 4422–4454, https://doi.org/10.1002/nme.7040.
[22] J. Baek, J.S. Chen, A neural network-based enrichment of reproducing kernel approximation for modeling brittle fracture, Comput. Methods Appl. Mech. Eng. 419 (2024) 116590, https://doi.org/10.1016/J.CMA.2023.116590.
[23] C.A. Duarte, I. Babuška, J.T. Oden, Generalized finite element methods for three-dimensional structural mechanics problems, Comput. Struct. 77 (2) (2000) 215–232, https://doi.org/10.1016/S0045-7949(99)00211-4.
[24] C.A. Duarte, D.J. Kim, Analysis and applications of a generalized finite element method with global–local enrichment functions, Comput. Methods Appl. Mech. Eng. 197 (6–8) (2008) 487–504, https://doi.org/10.1016/J.CMA.2007.08.017.
[25] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, Theano: a CPU and GPU math expression compiler, Proc. Python Sci. Comput. Conf. 4 (3) (2010) 1–7.
[26] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, TensorFlow: a system for large-scale machine learning, in: Proc. 12th USENIX Symp. Oper. Syst. Des. Implementation, OSDI 2016, 2016, pp. 265–283. Available: https://arxiv.org/abs/1605.08695v2.
[27] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, "MXNet: a flexible and efficient machine learning library for heterogeneous distributed systems," 2015, Available: https://arxiv.org/abs/1512.01274v1.
[28] F. Chollet, Deep Learning With Python, Simon and Schuster, 2021.
[29] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018) 1–43.
[30] R.A. Jacobs, M.I. Jordan, S.J. Nowlan, G.E. Hinton, Adaptive mixtures of local experts, Neural Comput 3 (1) (1991) 79–87.
[31] K. Lee, N.A. Trask, R.G. Patel, M.A. Gulian, and E.C. Cyr, "Partition of unity networks: deep hp-approximation," arXiv preprint arXiv:2101.11256, 2021.
[32] N. Trask, A. Henriksen, C. Martinez, E. Cyr, Hierarchical partition of unity networks: fast multilevel training, Proc. Mach. Learn. Res. 145 (2022) 1–20.
[33] 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, https://doi.org/10.1016/J.JCP.2018.10.045.
[34] E. Haghighat, R. Juanes, SciANN: a Keras/TensorFlow wrapper for scientific computations and physics-informed deep learning using artificial neural networks, Comput. Methods Appl. Mech. Eng. 373 (2021) 113552, https://doi.org/10.1016/J.CMA.2020.113552.
[35] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364, https://doi.org/10.1016/J.JCP.2018.08.029.
[36] E. Weinan, B. Yu, The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12, https://doi.org/10.1007/s40304-018-0127-z.
[37] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: concepts, implementation and applications, Comput. Methods Appl. Mech. Eng. 362 (2020) 112790, https://doi.org/10.1016/j.cma.2019.112790.
[38] V.M. Nguyen-Thanh, X. Zhuang, T. Rabczuk, A deep energy method for finite deformation hyperelasticity, Eur. J. Mech. A/Solids 80 (2020) 103874, https://doi.org/10.1016/J.EUROMECHSOL.2019.103874.
[39] V.M. Nguyen-Thanh, C. Anitescu, N. Alajlan, T. Rabczuk, X. Zhuang, Parametric deep energy approach for elasticity accounting for strain gradient effects, Comput. Methods Appl. Mech. Eng. 386 (2021) 114096, https://doi.org/10.1016/J.CMA.2021.114096.
[40] S. Saha, Z. Gan, L. Cheng, J. Gao, O.L. Kafka, X. Xie, H. Li, M. Tajdari, H.A. Kim, W.K. Liu, Hierarchical deep learning Neural Network (HiDeNN): an artificial intelligence (AI) framework for computational science and engineering, Comput. Methods Appl. Mech. Eng. 373 (2021) 113452, https://doi.org/10.1016/j.cma.2020.113452.
[41] J.M. Melenk, I. Babuška, The partition of unity finite element method: basic theory and applications, Comput. Methods Appl. Mech. Eng. 139 (1–4) (1996) 289–314, https://doi.org/10.1016/S0045-7825(96)01087-0.
[42] I.K. Babuška, J.M. Melenk, The partition of unity method, Int. J. Numer. Methods Eng. 40 (1997) 727–758, https://doi.org/10.1002/(SICI)1097-0207(19970228)40:4.
[43] C.A. Duarte, J.T. Oden, An h-p adaptive method using clouds, Comput. Methods Appl. Mech. Eng. 139 (1–4) (1996) 237–262, https://doi.org/10.1016/S0045-7825(96)01085-7.
[44] A. Duarte, J. Tinsley Odent, H-p clouds-An h-p meshless method, Numer. Methods Partial Differ. Equations An Int. J. 12 (1996) 673–705, https://doi.org/10.1002/(SICI)1098-2426(199611)12:6.
[45] T.J.R. Hughes, J.A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Comput. Methods Appl. Mech. Eng. 194 (39–41) (2005) 4135–4195, https://doi.org/10.1016/J.CMA.2004.10.008. Oct.

[46] T. Belytschko, Y.Y. Lu, L. Gu, Element-free Galerkin methods, Int. J. Numer. Methods Eng. 37 (2) (1994) 229–256, https://doi.org/10.1002/nme.1620370205. Jan.
[47] W.K. Liu, S. Jun, Y.F. Zhang, Reproducing kernel particle methods, Int. J. Numer. Methods Fluids 20 (8–9) (1995) 1081–1106, https://doi.org/10.1002/FLD.1650200824.
[48] J.S. Chen, C. Pan, C.T. Wu, W.K. Liu, Reproducing kernel particle methods for large deformation analysis of non-linear structures, Comput. Methods Appl. Mech. Eng. 139 (1–4) (1996) 195–227, https://doi.org/10.1016/S0045-7825(96)01083-3. Dec.
[49] M. Fleming, Y.A. Chur, T. Belytschko, Enriched element-free Galerkin methods for crack tip fields, Int. J. Numer. Methods Eng. 29 (1996) 1483–1504, https://doi.org/10.1002/(SICI)1097-0207(19970430)40:8.
[50] T. Belytschko, M. Fleming, Smoothing, enrichment and contact in the element-free Galerkin method, Comput. Struct. 71 (2) (1999) 173–195, https://doi.org/10.1016/S0045-7949(98)00205-3.
[51] J.S. Chen, M. Hillman, W. Chi, Meshfree methods: progress made after 20 years, Am. Soc. Civ. Eng. 143 (4) (2017), https://doi.org/10.1061/(ASCE)EM.1943-7889.0001176.
[52] T. Belytschko, J.S. Chen, M.C. Hillman, Meshfree and Particle methods: Fundamentals and Applications, John Wiley \& Sons, 2024.
[53] D.A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (ELUs), in: 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc, 2015. Available: https://arxiv.org/abs/1511.07289v5.
[54] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. Available: http://image-net.org/challenges/LSVRC/2015/.
[55] W. Han, W.K. Liu, G.J. Wagner, Convergence analysis of a hierarchical enrichment of Dirichlet boundary conditions in a mesh-free method, Int. J. Numer. Methods Eng. 53 (6) (2002) 1323–1336, https://doi.org/10.1002/NME.336. Feb.
[56] J.S. Chen, W. Han, Y. You, X. Meng, A reproducing kernel method with nodal interpolation property, Int. J. Numer. Methods Eng. 56 (2003) 935–960, https://doi.org/10.1002/nme.592.
[57] E. Weinan, Q. Wang, Exponential convergence of the deep neural network approximation for analytic functions, Sci. China Math. 61 (10) (2018) 1733–1740, https://doi.org/10.1007/s11425-018-9387-x.
[58] J.S. Chen, C.T. Wu, S. Yoon, Y. You, Stabilized conforming nodal integration for Galerkin mesh-free methods, Int. J. Numer. Methods Eng. 50 (2) (2001) 435–466, https://doi.org/10.1002/1097-0207(20010120)50:2<435::AID-NME32>3.0.CO;2-A.
[59] Y. You, J.S. Chen, H. Lu, Filters, reproducing kernel, and adaptive meshfree method, Comput. Mech. 31 (3) (2003) 316–326, https://doi.org/10.1007/s00466-003-0434-3.
[60] H. Lu, J.S. Chen, Adaptive Galerkin Particle Method. Meshfree Methods for Partial Differential Equations, Springer, Berlin Heidelberg, 2003, pp. 251–265.
[61] Y. Wang, J. Baek, Y. Tang, J. Du, M. Hillman, J.S. Chen, Support vector machine guided reproducing kernel particle method for image-based modeling of microstructures, Comput. Mech. 2023 (Oct. 2023) 1–36, https://doi.org/10.1007/S00466-023-02394-9.
[62] D.P. Kingma and J.L. Ba, "Adam: a method for stochastic optimization," 2015, Available: https://arxiv.org/abs/1412.6980v9.
[63] T.J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.
[64] E.G. Kirsch, Die Theorie der Elastizit t und die bed Rfnisse der Festigkeitslehre, Zeitshrift des Vereines Dtsch. Ingenieure 42 (1898) 797–807.
[65] J. Kurtz, L. Demkowicz, A fully automatic hp-adaptivity for elliptic PDEs in three dimensions, Comput. Methods Appl. Mech. Eng. 196 (37–40) (2007) 3534–3545, https://doi.org/10.1016/J.CMA.2006.10.053. Aug.
[66] W. Rachowicz, J.T. Oden, L. Demkowicz, Toward a universal h-p adaptive finite element strategy part 3. design of h-p meshes, Comput. Methods Appl. Mech. Eng. 77 (1–2) (1989) 181–212, https://doi.org/10.1016/0045-7825(89)90131-X.
[67] J.S. Chen, C. Pan, C.M.O.L. Roque, H.P. Wang, A Lagrangian reproducing kernel particle method for metal forming analysis, Comput. Mech. 22 (3) (1998) 289–307, https://doi.org/10.1007/S004660050361.
[68] H.Y. Hu, J.S. Chen, W. Hu, Error analysis of collocation method based on reproducing kernel approximation, Numer. Methods Partial Differ. Equ. 27 (3) (2011) 554–580, https://doi.org/10.1002/num.20539.