Computer Methods in Applied Mechanics and Engineering 435 (2025) 117655

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

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

Extended physics-informed extreme learning machine for linear elastic fracture mechanics

Bokai Zhu [a], Hengguang Li [b], Qinghui Zhang [a,c,*,1]

[a] School of Science, Harbin Institute of Technology, Shenzhen, PR China
[b] Department of Mathematics, Wayne State University, Detroit, MI 48202, USA
[c] Guangdong Province Key Laboratory of Computational Science, Sun Yat-Sen University, Guangzhou, 510006, PR China

ARTICLE INFO

Keywords:
Machine learning
Extreme learning machine
Crack
Singularity
Accuracy

ABSTRACT

The machine learning (ML) methods have been applied to numerical solutions to partial differential equations (PDEs) in recent years and achieved great success in PDEs with smooth solutions and in high dimensional PDEs. However, it is still challenging to develop high-precision ML solvers for PDEs with non-smooth solutions. The linear elastic fracture mechanics equation is a typical non-smooth problem, where the solution is discontinuous along with the crack face and has the radial singularity around the crack front. The general ML methods for the linear elastic fracture mechanics can achieve a relative error for displacements, about 10^{-3}. To improve the accuracy, we analyze and extract the singular factors from the asymptotic expansions of solutions of the crack problem, such that the solution can be expressed by the singular factor multiplied by other smooth components. Then the general ML methods are enriched (multiplied) by the singular factor and used in a physics-informed neural network formulation. The new method is referred to as the extended physics-informed ML method, which improves the approximation significantly. We consider two typical ML methods, fully connected neural networks and extreme learning machine, where the extended physics-informed ML based on the extreme learning machine (XPIELM) achieves the relative errors about 10^{-12}. We also study the stress intensity factor based on the XPIELM, and significantly improve the approximation of the stress intensity factor. The proposed XPIELM is applied to a two-dimensional Poisson crack problem, a two-dimensional elasticity problem, and a fully three-dimensional edge-crack elasticity problem in the numerical tests that exhibit various features of the method.

1. Introduction

The linear elastic fracture mechanics (LEFM) has been widely used in fracture analysis of brittle materials. It plays a critical role in studying crack propagation, stress concentration, fatigue, impact test, delamination of composites, and so on. The LEFM applies the theory of elasticity to crack-front fields to evaluate features of crack growth. The crack modeling involves two types of non-smooth features — the discontinuity at the crack surface and the singularity at the crack front. Conventional numerical techniques for the crack problem, e.g., the finite element method (FEM), need mesh matching or refinement. This increases computational costs substantially. There have also been many unfitted mesh methods and meshfree methods developed for the crack problem to improve

* Corresponding author at: School of Science, Harbin Institute of Technology, Shenzhen, PR China.
E-mail addresses: 200810224@stu.hit.edu.cn (B. Zhu), li@wayne.edu (H. Li), zhangqh@hit.edu.cn, zhangqh6@mail.sysu.edu.cn (Q. Zhang).

[1] This research was partially supported by the Natural Science Foundation of China under grant 12471370, and Guangdong Provincial Natural Science Foundation of China under grant 2022A1515011187.

https://doi.org/10.1016/j.cma.2024.117655
Received 24 April 2024; Received in revised form 4 December 2024; Accepted 5 December 2024

computational efficiency, such as the Generalized/eXtended FEM [1–8], enriched element-free Galerkin or particle methods [9–13], and isogeometric analysis [14]. Nevertheless, it remains a challenge to develop efficient and accurate numerical simulation techniques for the analysis of the crack problem.

This paper studies applications of machine learning (ML) for the numerical solution to the crack problem, with a focus on neural networks (NNs). The ML techniques [15] have achieved great success in image recognition, speech recognition, and natural language processing [15,16]. In recent years the ML has been extensively applied to the numerical solution to PDEs. The ML solvers enjoy many advantages. For instance, (a) the ML solver is meshfree and dimensionless, (b) it is flexible to approximate nonlinear terms and complex geometries, and (c) it is coding-friendly and computationally efficient because of the availability of automatic differentiation mechanics, optimization solvers, powerful architectures (e.g., TensorFlow and PyTorch [17]). According to different formulation principles (e.g., strong, weak and mixed forms), the typical ML techniques include the physics-informed neural network (PINN), physics informed augmentation of neural networks, and DGM [18–20], energy methods [21,22], the weak adversarial neural network [23], the mixed residual method [24], Nitsche based methods [25], radial basis neural network [26,27], and many others [28–34]. The applications of ML to engineering computations can be found in [22,26,35–39]. The ML has also been used to solve high-dimensional PDEs [23,24,40–44]. Most of the afore-mentioned studies are concentrated on the numerical solution to the PDEs without using measured data. The ML techniques based on measured data were developed in [45,46], where the high-fidelity locally measured data are incorporated into certain physical models to derive global predictions.

Despite the success of ML methods in the smooth and high-dimensional problems, their application to the non-smooth problem is not straightforward, as fewer studies are devoted to the non-smooth problem. A PDE is called non-smooth if its solution involves non-smooth features [7], such as (weak) discontinuities, kinks, singularities, cracks, multi-scale properties, boundary layers, sharp gradients, high waves. In general, the ML methods produce poor approximation for the non-smooth problem in comparison with the smooth problem. This is partly because the commonly-used NN functions are smooth due to the smooth activation functions (except for the ReLU function). Recently, NN methods have been applied to the non-smooth problems including interface problems [47–51], sharp gradients [52], multi-scale problems [27], corner singularities [53], singularly-perturbed problems [54], waves [26,55], and porous media [56]. The crack problem is a typical non-smooth problem, which involves both the discontinuity and the crack front singularity. We refer to [39,57,58] for the applications of NNs to the crack modeling in brittle and quasi-brittle materials. This paper will report the implementation of a fully connected NN (using a piecewise version because of the discontinuity) for a crack problem, where the relative error of displacements is about 10^{-3}, showing the challenge to approximate non-smooth problems.

To improve the precision, we incorporate the singular information of crack solutions into the construction of the NN structure. By analyzing the singular factor based on asymptotic expansions of the crack solution [59–64], we extract the singular factor and express the solution as the singular factor multiplied by other smooth components. Then the general NNs are enriched (multiplied) by the singular factor and used in a PINN scheme [18,20]. The new method is referred to as extended physics-informed ML (XPIML). The idea is motivated by the extended FEM (XFEM) [1–8], where the standard FEM is enriched with the branch functions to improve the precision. The XPIMLs improve the approximation precision significantly. We consider two typical ML methods, the fully connected NN (FCNN) [15] and the extreme learning machine (ELM) [65] in this study. The ELM is a shallow NN. Unlike the conventional NNs, the weights and bias coefficients in hidden layers of the ELM are initialized with random values, which are fixed during training process. The ELM has been applied to the numerical solutions to PDEs. See [50,52,66–68] for instance. We shall show that the XPIML based on the ELM (XPIELM) achieves the high precision. The relative errors in the displacement are about 10^{-12} (compared with about 10^{-3} for the conventional FCNN). The stress intensity factor (SIF) is crucial to quantify the severity of a crack front stress. We also compute the SIF using the $J-$integral method [69,70] based on the XPIELM solution. The high precision in the SIF is achieved thanks to the good approximation by the XPIELM. The proposed XPIELMs are applied to a two dimensional (2D) Poisson crack problem [64,71], a 2D elasticity problem [5,6,72], and a fully 3D edge-crack elasticity problem [63,73,74] in a unified way, where its computational advantages are demonstrated.

The paper is organized as follows. The LEFM model problem is described in Section 2, including a 2D Poisson crack problem, 2D and 3D elasticity crack problems, and the asymptotic expansions of solutions. Various existing ML methods are described in Sections 3, and the piecewise NNs are presented to show the precision of conventional NNs used in the crack problem. The extended physics-informed NNs based on the FCNN and ELM are proposed in Section 4. The numerical experiments and concluding remarks are presented in Sections 5 and 6, respectively.

2. Model problem

Let Ω be a bounded and cracked domain in $\mathbb{R}^d, d = 2,3$ with the crack surface Γ_O, where O is the crack front, see Fig. 1. $\overline{\Omega}$ is the closure of Ω, and $\Gamma = \partial\overline{\Omega}$ is the boundary of $\overline{\Omega}$. We use characters in bold, e.g., $\mathbf{x}$, to represent points in $\mathbb{R}^d$, and denote $\mathbf{x} = (x_1, \dots, x_d)$. For simplicity, we also denote $\mathbf{x} = (x, y)$ and $\mathbf{x} = (x, y, z)$ in 2D and 3D problems, respectively. Likewise, We use characters in bold, e.g., $\mathbf{u}$, to represent vector-valued functions in $H^1(\Omega;\mathbb{R}^d)$, and denote $\mathbf{u} = [u_1, u_2, \dots, u_d]^T$. Except for the elasticity crack problem, we also consider a Poisson crack problem in this paper. In this case the solution is scalar-valued function, and for convenience of presentation, we also use the bold $\mathbf{u}$ to represent the scalar-valued function with an understanding $\mathbf{u} = u \in H^1(\Omega;\mathbb{R})$.

Consider a linear elastic fracture mechanics model in Ω:

$$-\nabla \cdot \sigma(\mathbf{u}) = \mathbf{f}, \quad \text{in } \Omega, \tag{2.1}$$

$$\mathbf{u} = \mathbf{g}, \quad \text{on } \Gamma,$$

$$\sigma(\mathbf{u})\vec{n}_O = \mathbf{0}, \quad \text{on } \Gamma_O, \tag{2.2}$$

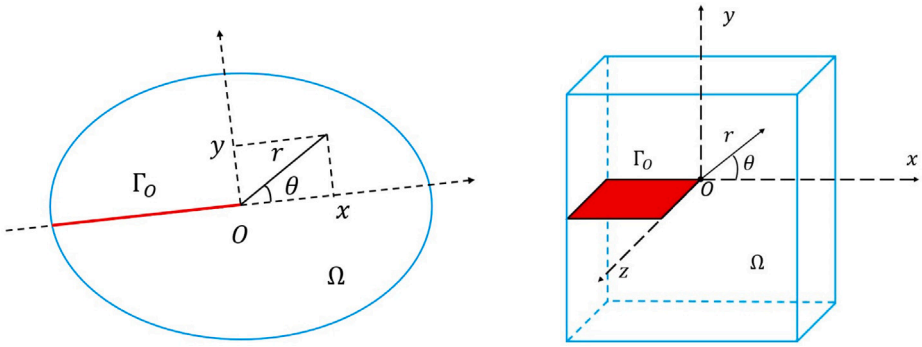

Fig. 1. The 2D (left) and the 3D (right) cracked domain Ω with the crack surface Γ_O and the crack front O. $(x, y), (x, y, z)$ are the Cartesian coordinate systems in 2D and 3D, respectively, and $(r, \theta), (r, \theta, z)$ are the associated polar coordinates and the cylindrical polar coordinates, respectively.

where $\sigma(\mathbf{u})$ is the stress tensor, $\mathbf{f}$ is the body force, and $\mathbf{g}$ represents the essential boundary condition. We assume that $\mathbf{f}$ and $\mathbf{g}$ are smooth to simplify description of the main algorithm. The Eq. (2.2) is called the traction free condition on the crack Γ_O, and $\vec{n}_O$ is the unit outward normal vector to Γ_O. In this paper we study a Poisson crack problem, a 2D elasticity crack problem, and a fully 3D edge-crack elasticity problem. We describe these models below.

- The 2D Poisson crack problem:

$$\sigma(\mathbf{u}) := \nabla \mathbf{u} = \left[\frac{\partial \mathbf{u}}{\partial x}, \frac{\partial \mathbf{u}}{\partial y}\right].$$

- The 2D elasticity crack problem:

$$\sigma(\mathbf{u}) := \begin{bmatrix} \sigma_x & \sigma_{xy} \\ \sigma_{xy} & \sigma_y \end{bmatrix}$$

with the constitutive model given by

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_{xy} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} \text{ or } \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix},$$

where the former is the plane strain, while the latter is the plane stress, E is Young's modulus, and ν is Poisson's ratio, $x-$ and $y-$ directions are shown in Fig. 1 (left), and ϵ_x, ϵ_y, γ_{xy} are the components of strain tensor defined by

$$\begin{bmatrix} \epsilon_x & \frac{1}{2}\gamma_{xy} \\ \frac{1}{2}\gamma_{xy} & \epsilon_y \end{bmatrix} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{1}{2}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) \\ \frac{1}{2}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) & \frac{\partial u_2}{\partial y} \end{bmatrix}.$$

- The fully 3D edge-crack elasticity problem:

$$\sigma(\mathbf{u}) := \begin{bmatrix} \sigma_x & \sigma_{xy} & \sigma_{zx} \\ \sigma_{xy} & \sigma_y & \sigma_{yz} \\ \sigma_{zx} & \sigma_{yz} & \sigma_z \end{bmatrix}$$

with the constitutive model given by

$$\begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \sigma_{xy} \\ \sigma_{yz} \\ \sigma_{zx} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & 1 & \frac{\nu}{1-\nu} & 0 & 0 & 0 \\ \frac{\nu}{1-\nu} & \frac{\nu}{1-\nu} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix},$$

where the components of the strain tensor are defined by

$$\epsilon(\mathbf{u}) := \begin{bmatrix} \epsilon_x & \frac{1}{2}\gamma_{xy} & \frac{1}{2}\gamma_{zx} \\ \frac{1}{2}\gamma_{xy} & \epsilon_y & \frac{1}{2}\gamma_{yz} \\ \frac{1}{2}\gamma_{zx} & \frac{1}{2}\gamma_{yz} & \epsilon_z \end{bmatrix} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{1}{2}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) & \frac{1}{2}\left(\frac{\partial u_3}{\partial x} + \frac{\partial u_1}{\partial z}\right) \\ \frac{1}{2}\left(\frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}\right) & \frac{\partial u_2}{\partial y} & \frac{1}{2}\left(\frac{\partial u_3}{\partial y} + \frac{\partial u_2}{\partial z}\right) \\ \frac{1}{2}\left(\frac{\partial u_3}{\partial x} + \frac{\partial u_1}{\partial z}\right) & \frac{1}{2}\left(\frac{\partial u_3}{\partial y} + \frac{\partial u_2}{\partial z}\right) & \frac{\partial u_3}{\partial z} \end{bmatrix}.$$

For the 2D problem, we assume that Γ_O is a straight line with a tip O, and (r, θ) is the polar coordinate with the crack tip O serving as the pole and the opposite direction of crack line as the polar line (see Fig. 1 (left)). For the 3D elasticity problem, we

assume that Ω is a plate of thickness $2t$ given by $\Omega = \varpi \times [-t, t]$, where ϖ is a region in $\mathbb{R}^2$. The crack front O is a straight line that is parallel to the z-axis. For convenience of presentation, we use the cylindrical polar coordinates (r, θ, z) (see Fig. 1 (right)), where

- z- the direction along the crack front O,
- r- the radial direction out from the crack front O,
- θ- the tangential direction around the z-direction (counterclockwise when viewed from the positive direction of z), and $\theta \in (-\pi, \pi]$.

The representations of solutions to these crack problems have been studied theoretically, and we describe them below.

- The 2D Poisson crack problem [64].

$$u(r,\theta) = \sum_{i=1}^{\infty} c_i r^{\frac{2i-1}{2}} \sin(\frac{2i-1}{2}\theta) + \sum_{i=0}^{\infty} d_i r^i \cos(i\theta) + u_{sm}, \tag{2.3}$$

  where c_i, d_i are the constants, u_{sm} is a smooth part in $\overline{\Omega}$.
- The 2D elasticity crack problem.
  The solution $\mathbf{u}$ of the crack problem (2.1) can be decomposed into [61]

$$\mathbf{u} = \mathbf{S} + \mathbf{u}_{sm}, \ \ \mathbf{S} = [s_1, s_2]^T, \tag{2.4}$$

  where $\mathbf{u}_{sm}$ is a smooth part in $\overline{\Omega}$, and $\mathbf{S}$ corresponds to the singular component given by

$$\begin{aligned} s_1 \ =& \sum_{i=1}^{\infty} a_{i1} \frac{r^{i/2}}{2\mu} \left[(\kappa + \frac{i}{2} + (-1)^i) \cos\frac{i\theta}{2} - \frac{i}{2}\cos\frac{(i-4)\theta}{2} \right] \\ & -a_{i2} \frac{r^{i/2}}{2\mu} \left[(\kappa + \frac{i}{2} - (-1)^i) \sin\frac{i\theta}{2} - \frac{i}{2}\sin\frac{(i-4)\theta}{2} \right], \end{aligned} \tag{2.5}$$

$$\begin{aligned} s_2 \ =& \sum_{i=1}^{\infty} a_{i1} \frac{r^{i/2}}{2\mu} \left[(\kappa - \frac{i}{2} - (-1)^i) \sin\frac{i\theta}{2} + \frac{i}{2}\sin\frac{(i-4)\theta}{2} \right] \\ & +a_{i2} \frac{r^{i/2}}{2\mu} \left[(\kappa - \frac{i}{2} + (-1)^i) \cos\frac{i\theta}{2} + \frac{i}{2}\cos\frac{(i-4)\theta}{2} \right], \end{aligned} \tag{2.6}$$

  where a_{i1}, a_{i2} are the constants, μ is a Lamé constant, κ is the Kolosov constant that is $3-4\nu$ for the plane strain and $\frac{3-\nu}{1+\nu}$ for the plane stress. It is known that the terms in (2.5) and (2.6) with the factor $r^{\frac{1}{2}}$ expressed by

$$\mathbf{u}_I := M_I \sqrt{r} \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} \end{bmatrix} (\kappa - \cos\theta), \ \ \mathbf{u}_{II} := M_{II} \sqrt{r} \begin{bmatrix} \sin\frac{\theta}{2}(\kappa + 2 + \cos\theta) \\ \cos\frac{\theta}{2}(2 - \kappa - \cos\theta) \end{bmatrix}, \tag{2.7}$$

  are referred to as the I and II opening modes (see [1,4,72]), where M_I, M_{II} are the coefficients associated with the stress intensity factors (SIFs).
- The fully 3D edge-crack elasticity problem.
  A solution $\mathbf{u}$ of the 3D edge-crack problem (2.1) that incorporates the singular information was derived in [62,63], which is expressed explicitly in the Cartesian Coordinates as follows:

$$\begin{aligned} \mathbf{u} \ =& \ A(z) r^{\frac{1}{2}} \begin{bmatrix} (Q_1 - 1)\cos\frac{\theta}{2} - \cos\frac{3}{2}\theta \\ (Q_1 + 1)\sin\frac{\theta}{2} - \sin\frac{3}{2}\theta \\ 0 \end{bmatrix} - A'(z) r^{\frac{3}{2}} \begin{bmatrix} 0 \\ 0 \\ 2\cos\frac{\theta}{2} - \frac{2}{3}(Q_1 + 1)\cos\frac{3}{2}\theta \end{bmatrix} \\ & + \frac{A''(z)}{2} r^{\frac{5}{2}} \begin{bmatrix} (Q_2 - Q_3 - \frac{Q_1+1}{6} - Q_4)\cos\frac{\theta}{2} + \cos\frac{3}{2}\theta + (Q_4 - Q_3)\cos\frac{5}{2}\theta \\ (Q_2 + Q_3 - \frac{Q_1+1}{6} + Q_4)\sin\frac{\theta}{2} + \sin\frac{3}{2}\theta + (Q_4 - Q_3)\sin\frac{5}{2}\theta \\ 0 \end{bmatrix}, \end{aligned} \tag{2.8}$$

  where

$$Q_1 = \frac{2\lambda + 6\mu}{\lambda + \mu}, Q_2 = \frac{3\lambda - \mu}{6(\lambda + \mu)}, Q_3 = \frac{45\lambda^2 + 138\lambda\mu + 61\mu^2}{90(\lambda + \mu)^2}, Q_4 = \frac{-15\lambda^2 + 2\lambda\mu + 49\mu^2}{90(\lambda + \mu)^2}, \tag{2.9}$$

  and

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)}, \ \ \mu = \frac{E}{2(1+\nu)}$$

  are the Lamé constants. If $A(z)$ is taken as $1 + z$ or $1 + z + z^2$ for example, we have the fully 3D edge-crack problem (neither plane strain nor plane stress), which involves the factors $r^{\frac{3}{2}}$ and/or $r^{\frac{5}{2}}$. The term with $A(z)r^{\frac{1}{2}}$ in (2.8) is referred to as the major singular function of $\mathbf{u}$. We note that there is an essential difference between the 2D and 3D crack problems; in the 2D major singular function (2.7) is much simpler (without $A(z)$).

Remark 2.1. It is obvious that the aforementioned solutions, $\mathbf{u}$, involve two typical non-smooth features: the singularity characterized by the factor $\sqrt{r}$ and the discontinuity across the crack front Γ_O. This poses a difficulty for the development of

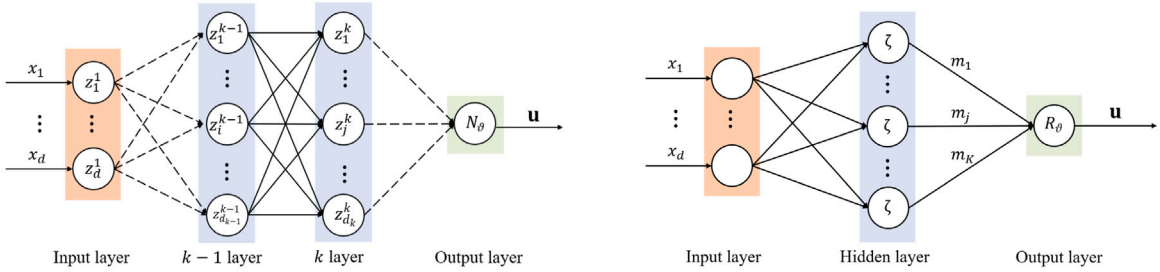

Fig. 2. Basic structure of the FCNN network (left) and the ELM network (right), where the dashed lines indicate the multilayer structure.

high-precision numerical methods. Below, we will propose the extended NN method, which depends on the expansion of solutions, (2.3), (2.4), and (2.8). □

3. Conventional neural networks and loss functions

The ML techniques are powerful in high-dimensional and nonlinear approximations, and have been widely applied to the numerical solutions of PDE. The conventional ML techniques of solving PDE involve the key gradients: (a) neural network structures, (b) loss functions for training process, and (c) optimization algorithms to optimize the loss function. We describe their formulations when applied to the crack problem.

We start with two typical ML methods, the fully connected NN (FCNN) and the extreme learning machine (ELM). Let $\mathbf{z}^k \in \mathbb{R}^{d_k}$ be a vector of dimension d_k, and $\mathbf{z}^1 = (x_1, \dots, x_d)^T$. Let $\mathbf{W}^k \in \mathbb{R}^{d_{k+1} \times d_k}$ be a $d_{k+1} \times d_k$ matrix and $\mathbf{b}^k \in \mathbb{R}^{d_{k+1}}$ be a vector of dimension d_{k+1}, and d_{L+1} is the same as the dimension of $\mathbf{u}$, which is 1, 2, 3 for the Poisson problem, 2D elasticity problem, 3D elasticity problem, respectively. Denote the activation function by ζ, and $\zeta(\mathbf{z}^k)$ is defined by $(\zeta(\mathbf{z}_1^k), \dots, \zeta(\mathbf{z}_{d_k}^k))^T$. A FCNN function of L-layer is defined by

$$N_\vartheta := \mathbf{W}^L N_{L-1} \circ \cdots \circ N_2 \circ N_1(\mathbf{z}^1) + \mathbf{b}^L, \;\; N_k(\mathbf{z}^k) := \zeta(\mathbf{W}^k \mathbf{z}^k + \mathbf{b}^k), \;\; k = 1, 2, \dots, L-1,$$

where $\vartheta = \{\mathbf{W}^k, \mathbf{b}^k\}_{k=1}^L$ are referred to as parameter set. A FCNN is exhibited in Fig. 2 (left). The improved versions of FCNN for the numerical solution to PDE, e.g., ResNet [16,21,42], can be found in the literature. We do not present their results in this study for simplicity, and they can be used in the proposed method directly in the same way as the FCNN. The parameters ϑ are determined by solving certain minimization problems, which are updated using stochastic gradient descent (SGD) methods [15].

The second ML method is the extreme learning machine (ELM) [65]. The ELM is a shallow NN with randomly selected weights and bias. Let $\mathbf{x}^c = (x_1^c, x_2^c, \dots, x_d^c) \in \Omega$ be a relative center of Ω, and r_c be a positive scaled parameter. The functions of ELM are the following:

$$R_\vartheta = \sum_{j=1}^{K} m_j \zeta\left(\frac{\mathbf{x} - \mathbf{x}^c}{r_c} \cdot \mathbf{W}_j^T + b_j\right),$$

where $\mathbf{W}_j$ and $b_j, j = 1, 2, \dots, K$ are random numbers uniformly distributed on $[-Rm, Rm]$; Rm is a hyper parameter. It is noted that the parameters $\mathbf{W}_j, b_j$ are generated in advance and fixed in the training process. See Fig. 2 (right) for an illustration of ELM. The optimization process of ELM can be executed by solving certain least square problems. The ELM has been applied to solve the PDE, see [52,66–68] for instance.

In this paper we use the tanh activation function [15], and thus both the FCNN and ELM functions are C^∞ continuous. We mention that commonly-used activation functions are continuous, such as ReLU, sigmoid, sin/cos, and thus the NN functions are generally continuous. This causes a difficulty in solving the crack problems because the crack solution is discontinuous across the crack surfaces. Therefore, the usual NNs with the continuous activation functions cannot be used directly for the crack problem, and some operations have to be developed, such as, piecewise NNs [48]. We present a piecewise NN below.

The parameters of FCNN or ELM need to be identified by optimizing particular loss functions, whose optimizers are the approximate solutions of equations. A natural loss function is based on energy functional:

$$\mathbb{L}(\mathbf{u}) := \frac{1}{2} B(\mathbf{u}, \mathbf{u}) - F(\mathbf{u}) + \alpha \|\mathbf{u} - \mathbf{g}\|^2_{L^2(\Gamma)}, \tag{3.1}$$

where α is a regularization parameter, and

$$B(\mathbf{u}, \mathbf{w}) = \int_\Omega \sigma(\mathbf{u}) : \epsilon(\mathbf{w})\, dV, \quad F(\mathbf{w}) = \int_\Omega \mathbf{f} \cdot \mathbf{w}\, dV, \;\; \forall \;\; \mathbf{u}, \;\; \mathbf{w} \in H^1(\Omega). \tag{3.2}$$

A NN method based on (3.1) was referred to as the deep Ritz method (DRM) [21,22]. The DRM uses the energy formulation to establish the loss function. However, the energy formulation may not be available in many PDEs. To avoid this difficulty, a so-called weak adversarial network (WAN, [23]) utilizes the weak forms of PDEs to construct the loss function as follows:

$$\mathbb{L}(\mathbf{u}, \mathbf{w}) := \frac{(B(\mathbf{u}, \mathbf{w}) - F(\mathbf{w}))^2}{\|\mathbf{w}\|^2_{L^2(\Omega)}} + \alpha \|\mathbf{u} - \mathbf{g}\|^2_{L^2(\Gamma)}, \tag{3.3}$$

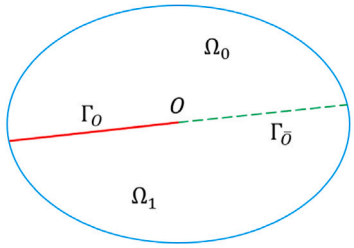

Fig. 3. The domain is divided into two sub-domains, Ω_0 and Ω_1, with the cracked part Γ_O and the non-crack part $\Gamma_{\bar{O}}$.

and the associated optimization problem is

$$\min_{\mathbf{u}} \max_{\mathbf{w}} \ \mathbb{L}(\mathbf{u}, \mathbf{w}).$$

Above, $\mathbf{u}, \mathbf{w}$ are approximated using the primal and adversarial NNs, respectively. (3.3) is based on the weak form, which is easy to attain for general PDEs using standard integration by part (or Green formulae). To optimize the parameters, a gradient descent and ascent approach is employed.

Compared with DRM and WAN, more convenient loss functions are based on the strong form of equation, (2.1), in the following:

$$\mathbb{L}(\mathbf{u}) := \| -\nabla \cdot \sigma(\mathbf{u}) - \mathbf{f}\|^2_{L^2(\Omega)} + \alpha\|\mathbf{u} - \mathbf{g}\|^2_{L^2(\Gamma)} + \beta\|\sigma(\mathbf{u})\vec{n}_O - \mathbf{0}\|^2_{L^2(\Gamma_O)}, \tag{3.4}$$

where α and β are the penalty parameters. The physics-informed neural network [18,20] was based on the strong form of equation. We also refer to (3.4) as PINN. Derivatives of higher orders are needed in the PINN. However, this is not a problem thanks to automatic differentiation mechanism for the NNs.

As mentioned above, the NN functions are continuous, which cannot be applied to the crack problem directly. The piecewise NNs are natural choices to simulate the discontinuous solutions. To this end, we extend the crack surface to the boundary of $\overline{\Omega}$ and divide Ω into two parts, Ω_0 and Ω_1, and the non-cracked part of $\overline{\Omega}_0 \cap \overline{\Omega}_1$ is denoted by $\Gamma_{\bar{O}}$, see Fig. 3. Two NNs are set in Ω_0 and Ω_1 to approximate $\mathbf{u}$, respectively,

$$\mathbf{u}(\mathbf{x}) \approx \mathbf{u}_\vartheta(\mathbf{x}) = \begin{cases} \mathbf{u}_{\vartheta_0}(\mathbf{x}), & \mathbf{x} \in \Omega_0, \\ \mathbf{u}_{\vartheta_1}(\mathbf{x}), & \mathbf{x} \in \Omega_1, \end{cases}$$

where $\vartheta = \vartheta_l$ if $\mathbf{x} \in \Omega_l$, $l = 0, 1$, and ϑ_l are the parameters of FCNN or ELM. This means that $\mathbf{u}_{\vartheta_0}$ and $\mathbf{u}_{\vartheta_1}$ are different NNs, which are set in Ω_0 and Ω_1, respectively. $\mathbf{u}_\vartheta$ is called piecewise NNs. The PINN based on the strong form, (2.1), and using the piecewise NNs, $\mathbf{u}(\mathbf{x}; \vartheta)$, is presented below:

$$\begin{aligned} \mathbb{L}(\mathbf{u}_\vartheta) := & \| -\nabla \cdot \sigma(\mathbf{u}_\vartheta) - \mathbf{f}\|^2_{L^2(\Omega)} + \alpha\|\mathbf{u}_\vartheta - \mathbf{g}\|^2_{L^2(\Gamma)} \\ & + \beta_0\|\sigma(\mathbf{u}_{\vartheta_0})\vec{n}_O - \mathbf{0}\|^2_{L^2(\Gamma_O)} + \beta_1\|\sigma(\mathbf{u}_{\vartheta_1})\vec{n}_O - \mathbf{0}\|^2_{L^2(\Gamma_O)} \\ & + \gamma_0\|\mathbf{u}_{\vartheta_0} - \mathbf{u}_{\vartheta_1}\|^2_{L^2(\Gamma_{\bar{O}})} + \gamma_1\|\nabla\mathbf{u}_{\vartheta_0} \cdot \vec{n}_O - \nabla\mathbf{u}_{\vartheta_1} \cdot \vec{n}_O\|^2_{L^2(\Gamma_{\bar{O}})}. \end{aligned} \tag{3.5}$$

The second line of (3.5) represents the traction free conditions (2.2) for each of piecewise NNs. It is noteworthy that the third line of (3.5) is to enforce continuous conditions along with $\Gamma_{\bar{O}}$ because $\Gamma_{\bar{O}}$ is the non-cracked surface, and the solution $\mathbf{u}$ and its derivatives are continuous at $\Gamma_{\bar{O}}$. The enforcement of continuity of piecewise NNs can be also seen in [68].

In the training process, the loss function (3.5) needs to be discretized by sampling points. Let $\{\mathbf{x}_i^{\Omega_0}\}_{i=1}^{N_0}$, $\{\mathbf{x}_i^{\Omega_1}\}_{i=1}^{N_1}$, $\{\mathbf{x}_j^b\}_{j=1}^{N_b}$, $\{\mathbf{x}_k^{\Gamma_O}\}_{k=1}^{N_{\Gamma_O}}$, $\{\mathbf{x}_m^{\Gamma_{\bar{O}}}\}_{m=1}^{N_{\Gamma_{\bar{O}}}}$ be the sampling points in Ω_0, Ω_1, Γ, Γ_O, $\Gamma_{\bar{O}}$, respectively, and $N_0, N_1, N_b, N_{\Gamma_O}, N_{\Gamma_{\bar{O}}}$ be the numbers of associated sampling points. These points can be sampled randomly or uniformly. Based on these sampling points we discretize (3.5) to get a loss function for training the NNs, $\mathbf{u}_\vartheta$:

$$\begin{aligned} \mathbb{L}(\mathbf{u}_\vartheta) := & \frac{1}{N_0}\sum_{i=1}^{N_0}\left([-\nabla \cdot \sigma(\mathbf{u}_\vartheta)](\mathbf{x}_i^{\Omega_0}) - \mathbf{f}(\mathbf{x}_i^{\Omega_0})\right)^2 + \frac{1}{N_1}\sum_{i=1}^{N_1}\left([-\nabla \cdot \sigma(\mathbf{u}_\vartheta)](\mathbf{x}_i^{\Omega_1}) - \mathbf{f}(\mathbf{x}_i^{\Omega_1})\right)^2 \\ & + \frac{\alpha}{N_b}\sum_{j=1}^{N_b}\left(\mathbf{u}_\vartheta(\mathbf{x}_j^b) - \mathbf{g}(\mathbf{x}_j^b)\right)^2 + \frac{\beta_0}{N_{\Gamma_O}}\sum_{k=1}^{N_{\Gamma_O}}\left(\sigma(\mathbf{u}_{\vartheta_0})(\mathbf{x}_k^{\Gamma_O})\vec{n}_O - \mathbf{0}\right)^2 + \frac{\beta_1}{N_{\Gamma_O}}\sum_{k=1}^{N_{\Gamma_O}}\left(\sigma(\mathbf{u}_{\vartheta_1})(\mathbf{x}_k^{\Gamma_O})\vec{n}_O - \mathbf{0}\right)^2 \\ & + \frac{\gamma_0}{N_{\Gamma_{\bar{O}}}}\sum_{m=1}^{N_{\Gamma_{\bar{O}}}}\left(\mathbf{u}_{\vartheta_0}(\mathbf{x}_m^{\Gamma_{\bar{O}}}) - \mathbf{u}_{\vartheta_1}(\mathbf{x}_m^{\Gamma_{\bar{O}}})\right)^2 + \frac{\gamma_1}{N_{\Gamma_{\bar{O}}}}\sum_{m=1}^{N_{\Gamma_{\bar{O}}}}\left(\nabla\mathbf{u}_{\vartheta_0}(\mathbf{x}_m^{\Gamma_{\bar{O}}}) \cdot \vec{n}_O - \nabla\mathbf{u}_{\vartheta_1}(\mathbf{x}_m^{\Gamma_{\bar{O}}}) \cdot \vec{n}_O\right)^2. \end{aligned} \tag{3.6}$$

The training process is to minimize the following optimization problem:

$$\min_{\vartheta} \ \mathbb{L}(\mathbf{u}_\vartheta) = \min_{\vartheta_0, \vartheta_1} \ \mathbb{L}(\mathbf{u}_\vartheta). \tag{3.7}$$

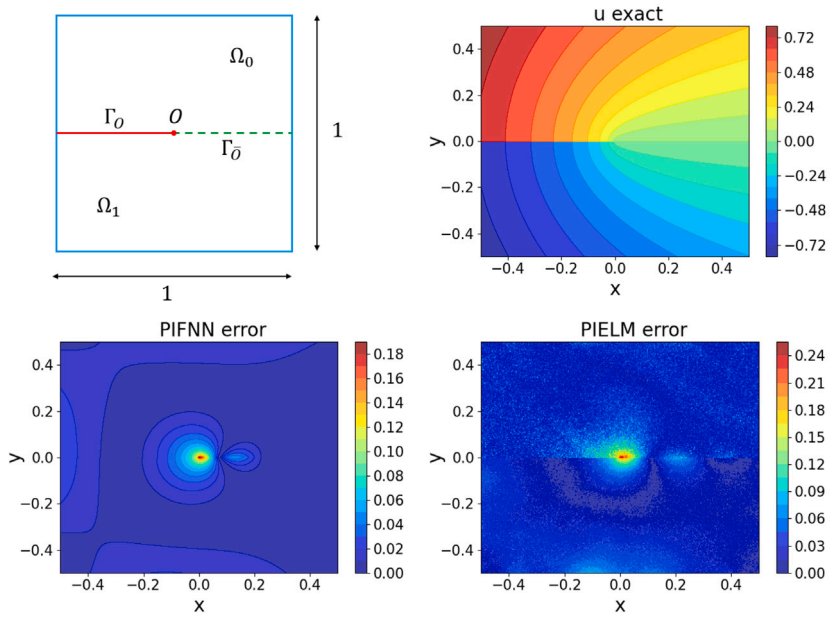

Fig. 4. First row: the crack domain Ω (left) and the exact solution (right) for the tested 2D Poisson crack problem; Second row: the absolute errors of PIFNN method (left) and PIELM method (right).

The minimizers of (3.7) based on the FCNN and ELM are referred to as physics-informed FCNN (PIFNN) and physics-informed ELM (PIELM) for the crack problem, respectively. The idea of piecewise NNs can also be applied to the energy and weak form situations, (3.1) and (3.3), in a similar approach. However, we do not present them because these loss functions do not exhibit accuracy merits over the PINN loss (3.5) in many PDEs including the crack equation investigated in this study.

Finally, we state the optimization algorithms of (3.7). For the PIFNN, the parameters ϑ are updated using stochastic gradient descent (SGD) methods [15]. The optimization process of PIELM can be executed by solving certain least square problems, see [52,66–68] for instance.

As seen in the aforementioned crack models, the solutions involve factors like r^α, i.e., these solutions are singular. It is known that the NNs used to approximate singular functions generally cause poor precision [53]. This is essentially different from the situation of smooth problems. To demonstrate it, we solve a Poisson crack problem using the PIFNN and PIELM.

We consider a 2D Poisson crack problem (2.1)–(2.2) on a domain $\Omega = [-0.5, 0.5]^2$ with the crack line $\Gamma_O = \{-0.5 \le x \le 0, y = 0\}$ and the crack tip O at the origin, see Fig. 4 (left-up). We use a manufactured solution $u(r,\theta) = r^{\frac{1}{2}} \sin(\frac{1}{2}\theta)$, see Fig. 4 (right-up). This crack problem is solved by the PIFNN and PIELM based on (3.7). We define the relative L^2 error of an approximation solution, u_ϑ, as $\frac{\|u-u_\vartheta\|_{L^2(\Omega)}}{\|u\|_{L^2(\Omega)}}$. The relative L^2 errors of the PIFNN and the PIELM methods are 3.078×10^{-2} and 7.177×10^{-2}, respectively. The errors are much bigger than those of NNs for the smooth problem. The Fig. 4 (left-below and right-below) draw the absolute errors of two methods, which show that the two methods produce notable errors around the crack tip.

4. Proposed extended PINNs

The objective of this study is to enhance the approximation precision of NNs for the crack problem. The idea is to incorporate the singular information of crack solutions into the NNs. To this end, we analyze the expansions of solutions and extract the major singularity. Then the solutions can be written as the major singularity multiplied by smooth parts. The NNs enriched with the major singularity are employed to solve the crack equations using the PINN scheme. We start with the 2D Poisson crack problem to illustrate the idea.

We rewrite the solution expansion, (2.3), as

$$\begin{aligned}\mathbf{u} &= r^{\frac{1}{2}} \sin\frac{\theta}{2} \sum_{i=1}^{\infty} c_i r^{i-1} \cos((i-1)\theta) + r^{\frac{1}{2}} \cos\frac{\theta}{2} \sum_{i=1}^{\infty} c_{i+1} r^i \sin(i\theta) + \sum_{i=0}^{\infty} d_i r^i \cos(i\theta) + u_{sm} \\ &=: \eta_1 \mathbf{H}_p(r,\theta) + \eta_2 \mathbf{G}_p(r,\theta) + \eta_0 \mathbf{K}_p(r,\theta), \end{aligned} \tag{4.1}$$

where

$$[\eta_1, \eta_2, \eta_0] = [r^{\frac{1}{2}} \sin\frac{\theta}{2}, r^{\frac{1}{2}} \cos\frac{\theta}{2}, 1], \tag{4.2}$$

and

$$\mathbf{H}_p(r,\theta)=\sum_{i=1}^{\infty} c_i r^{i-1}\cos((i-1)\theta),\ \mathbf{G}_p(r,\theta)=\sum_{i=1}^{\infty} c_{i+1} r^{i}\sin(i\theta),\ \mathbf{K}_p(r,\theta)=\sum_{i=0}^{\infty} d_i r^{i}\cos(i\theta)+u_{sm}.$$

We note that $\mathbf{H}_p, \mathbf{G}_p, \mathbf{K}_p$ are analytic and uniformly convergent. Most importantly, these functions do not have the discontinuity and singularity caused by the crack surface and tip, in other words, they are smooth in $\overline{\Omega}$. Therefore, $\mathbf{H}_p, \mathbf{G}_p, \mathbf{K}_p$ can be approximated by the NNs with high precision. We mention that in (4.1) the bold characters represent the scalar-valued functions for the Poisson problem, as remarked at the beginning of Section 2. We write this to describe the NN algorithms for the Poisson and elasticity crack problems using a unified way. Motivated from (4.1) we propose an extended NN for the Poisson crack problem as follows:

$$\mathbf{U}_{\vartheta}=\mathbf{U}_{\vartheta_1,\vartheta_2,\vartheta_0}=\eta_1\mathbf{u}_{\vartheta_1}+\eta_2\mathbf{u}_{\vartheta_2}+\eta_0\mathbf{u}_{\vartheta_0}, \tag{4.3}$$

where $\mathbf{u}_{\vartheta_l}, l=0,1,2$, are the general NN functions with the parameters, $\vartheta_l, l=0,1,2$, respectively, such as the FCNN, the ELM. It is clear that

$$\mathbf{u}-\mathbf{U}_{\vartheta}=\eta_1(\mathbf{u}_{\vartheta_1}-\mathbf{H}_p)+\eta_2(\mathbf{u}_{\vartheta_2}-\mathbf{G}_p)+\eta_0(\mathbf{u}_{\vartheta_0}-\mathbf{K}_p),$$

which shows that $\mathbf{u}_{\vartheta_l}, l=0,1,2$, are set to approximate the smooth functions $\mathbf{H}_p, \mathbf{G}_p, \mathbf{K}_p$, respectively. Therefore, the high precision is predicted in this way.

We next present the extended NN for the 2D elasticity crack problem. Using the similar arguments, we rewrite the solution expansion, $\mathbf{u}$ in (2.4), (2.5), and (2.6), as

$$\begin{aligned}\mathbf{u}=&\sqrt{r}\sin\frac{\theta}{2}\mathbf{H}_e(r,\theta)+\sqrt{r}\cos\frac{\theta}{2}\mathbf{G}_e(r,\theta)\\&+\sqrt{r}\sin\frac{\theta}{2}\cos\theta\mathbf{K}_e(r,\theta)+\sqrt{r}\cos\frac{\theta}{2}\cos\theta\mathbf{L}_e(r,\theta)+\mathbf{M}_e(r,\theta),\end{aligned}$$

with

$$\begin{aligned}\mathbf{H}_e=&\sum_{l=1}^{\infty} r^{l-1}\left(\begin{bmatrix}c_{11}^l\\d_{11}^l\end{bmatrix}\sin(l-1)\theta+\begin{bmatrix}c_{12}^l\\d_{12}^l\end{bmatrix}\cos(l-1)\theta\right.\\&\left.+\begin{bmatrix}c_{13}^l\\d_{13}^l\end{bmatrix}\sin(l-2)\theta\sin\theta+\begin{bmatrix}c_{14}^l\\d_{14}^l\end{bmatrix}\cos(l-2)\theta\sin\theta\right),\\
\mathbf{G}_e=&\sum_{l=1}^{\infty} r^{l-1}\left(\begin{bmatrix}c_{21}^l\\d_{21}^l\end{bmatrix}\sin(l-1)\theta+\begin{bmatrix}c_{22}^l\\d_{22}^l\end{bmatrix}\cos(l-1)\theta\right.\\&\left.+\begin{bmatrix}c_{23}^l\\d_{23}^l\end{bmatrix}\sin(l-2)\theta\sin\theta+\begin{bmatrix}c_{24}^l\\d_{24}^l\end{bmatrix}\cos(l-2)\theta\sin\theta\right),\\
\mathbf{K}_e=&\sum_{l=1}^{\infty} r^{l-1}\left(\begin{bmatrix}c_{31}^l\\d_{31}^l\end{bmatrix}\sin(l-1)\theta+\begin{bmatrix}c_{32}^l\\d_{32}^l\end{bmatrix}\cos(l-1)\theta\right),\\
\mathbf{L}_e=&\sum_{l=1}^{\infty} r^{l-1}\left(\begin{bmatrix}c_{41}^l\\d_{41}^l\end{bmatrix}\sin(l-1)\theta+\begin{bmatrix}c_{42}^l\\d_{42}^l\end{bmatrix}\cos(l-1)\theta\right),\\
\mathbf{M}_e=&\sum_{l=1}^{\infty} r^{l}\left(\begin{bmatrix}c_{51}^l\\d_{51}^l\end{bmatrix}\sin l\theta+\begin{bmatrix}c_{52}^l\\d_{52}^l\end{bmatrix}\cos l\theta\right)\\&+\sum_{l=1}^{\infty} r^{2}\left(\begin{bmatrix}c_{53}^l\\d_{53}^l\end{bmatrix}r^{l-2}\sin(l-2)\theta+\begin{bmatrix}c_{54}^l\\d_{54}^l\end{bmatrix}r^{l-2}\cos(l-2)\theta\right)+\mathbf{u}_{sm},\end{aligned}$$

where c_{ij}^l, d_{ij}^l are the constants obtained from a_{i1}, a_{i2} in (2.5) and (2.6). Here, $\mathbf{H}_e, \mathbf{G}_e, \mathbf{K}_e, \mathbf{L}_e, \mathbf{M}_e$ are analytic and uniformly convergent, and they are smooth in $\overline{\Omega}$. Let

$$[\eta_1,\eta_2,\eta_3,\eta_4,\eta_0]=[\sqrt{r}\sin\frac{\theta}{2},\sqrt{r}\cos\frac{\theta}{2},\sqrt{r}\sin\frac{\theta}{2}\cos\theta,\sqrt{r}\cos\frac{\theta}{2}\cos\theta,1], \tag{4.4}$$

and the extended NN for 2D elasticity crack problem is proposed as follows:

$$\mathbf{U}_{\vartheta}=\mathbf{U}_{\vartheta_1,\vartheta_2,\vartheta_3,\vartheta_4,\vartheta_0}=\eta_1\mathbf{u}_{\vartheta_1}+\eta_2\mathbf{u}_{\vartheta_2}+\eta_3\mathbf{u}_{\vartheta_3}+\eta_4\mathbf{u}_{\vartheta_4}+\eta_0\mathbf{u}_{\vartheta_0}, \tag{4.5}$$

where $\mathbf{u}_{\vartheta_l}, l=0,1,\ldots,4$, are the general NN functions with the parameters, $\vartheta_l, l=0,1,\ldots,4$, respectively, such as FCNN, ELM. We note that $\eta_l, l=1,2,3,4$ in (4.4) are the standard branch functions used to develop the so-called GFEM/XFEM for the crack problem [1,4,72].

Finally, we present the extended NN for the 3D crack problem. Since we consider a fully 3D fully edge-crack problem, the solution has the expression (2.8) [62,63,74]. It was proven in [74] that a stable GFEM enriched by the branch functions (4.4) for 2D crack problem can achieve the optimal convergence in the fully 3D edge-crack problem. Therefore, we also use (4.4) to construct the extended NN for the 3D planar fully edge-crack elasticity problem, i.e.,

$$\mathbf{U}_{\vartheta}=\mathbf{U}_{\vartheta_1,\vartheta_2,\vartheta_3,\vartheta_4,\vartheta_0}=\eta_1\mathbf{u}_{\vartheta_1}+\eta_2\mathbf{u}_{\vartheta_2}+\eta_3\mathbf{u}_{\vartheta_3}+\eta_4\mathbf{u}_{\vartheta_4}+\eta_0\mathbf{u}_{\vartheta_0}, \tag{4.6}$$

but remember that these functions are three-dimensional.

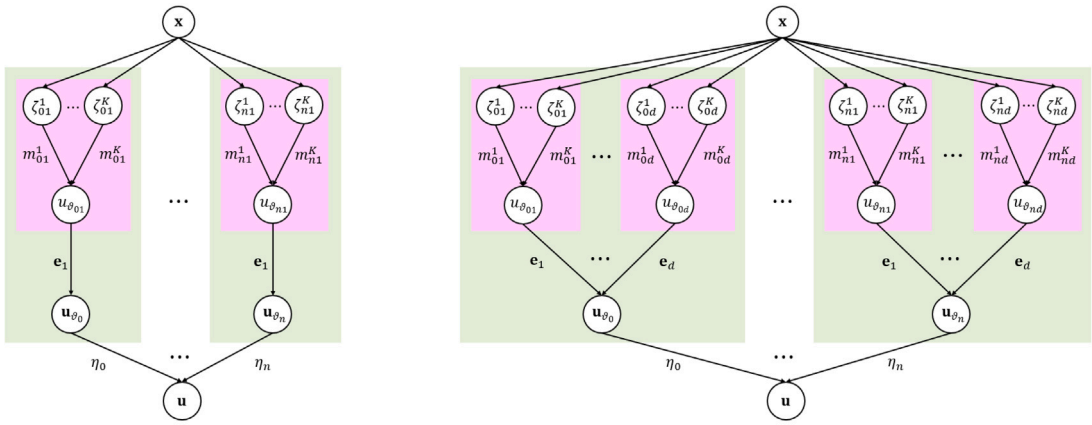

Fig. 5. The structure of the XPIELM model of the Poisson crack problem (left) and the elasticity crack problem (right). $n = 2$ for the Poisson crack problem and $n = 4$ for the elasticity crack problem. Each pink block is a ELM model for the component $u_{\vartheta_{lj}}$, and each green block is for $\mathbf{u}_{\vartheta_l}$ in (4.3) and (4.5). The only parameters to be trained are the output weights of each ELM model $m_{lj}^k, l = 0, \dots, n, j = 0, \dots, d, k = 1, \dots, K$.

Based on the sampling points used (3.6) we establish a loss function for training the extended NNs (a unified way for the Poisson, 2D and 3D crack problems), $\mathbf{U}_\vartheta$:

$$\begin{aligned}\mathbb{L}(\mathbf{U}_\vartheta) := \frac{1}{N_0+N_1}\sum_{i=1}^{N_0+N_1}\Big([-\nabla\cdot\sigma(\mathbf{U}_\vartheta)](\mathbf{x}_i^\Omega)-\mathbf{f}(\mathbf{x}_i^\Omega)\Big)^2+\frac{\alpha}{N_b}\sum_{j=1}^{N_b}\Big(\mathbf{U}_\vartheta(\mathbf{x}_j^b)-\mathbf{g}(\mathbf{x}_j^b)\Big)^2\\ +\frac{\beta_0}{N_{\Gamma_O}}\sum_{k=1}^{N_{\Gamma_O}}\Big(\sigma(\mathbf{U}_\vartheta^+)(\mathbf{x}_k^{\Gamma_O})\vec{n}_O-\mathbf{0}\Big)^2+\frac{\beta_1}{N_{\Gamma_O}}\sum_{k=1}^{N_{\Gamma_O}}\Big(\sigma(\mathbf{U}_\vartheta^-)(\mathbf{x}_k^{\Gamma_O})\vec{n}_O-\mathbf{0}\Big)^2,\end{aligned} \tag{4.7}$$

where $\{\mathbf{x}_i^\Omega\} = \{\mathbf{x}_i^{\Omega_0}\}\cup\{\mathbf{x}_i^{\Omega_1}\}$,

$$\mathbf{U}_\vartheta^+(\mathbf{x}) = \lim_{\mathbf{y}\to\mathbf{x},\mathbf{y}\in\Omega_0}\mathbf{U}_\vartheta(\mathbf{y}),\ \mathbf{U}_\vartheta^-(\mathbf{x}) = \lim_{\mathbf{y}\to\mathbf{x},\mathbf{y}\in\Omega_1}\mathbf{U}_\vartheta(\mathbf{y}),\ \forall\ \mathbf{x}\ \in\Gamma_O.$$

In comparison with (3.6), we use the full NNs instead of the piecewise NNs, and thus the NN functions are continuous, and the terms associated with the sampling on $\Gamma_{\bar{O}}$ (the non-cracked part, see Fig. 3) are not included.

The training process is to minimize the following optimization problem:

$$\min_\vartheta\ \mathbb{L}(\mathbf{U}_\vartheta) = \begin{cases}\min_{\vartheta_1,\vartheta_2,\vartheta_0}\ \mathbb{L}(\mathbf{U}_\vartheta), & \text{for the 2D Poisson crack problem,}\\ \min_{\vartheta_1,\vartheta_2,\vartheta_3,\vartheta_4,\vartheta_0}\mathbb{L}(\mathbf{U}_\vartheta), & \text{for the 2D and 3D elasticity crack problems.}\end{cases} \tag{4.8}$$

The minimizers of (4.8) based on the extended FCNN and ELM, (4.3) and (4.6), are referred to as the extended physics-informed FCNN (XPIFNN) and the extended physics-informed ELM (XPIELM) for the crack problem, respectively. It will be shown below that the precision of XPIFNN and XPIELM is improved essentially compared with their preliminary versions, PIFNN and PIELM (based on (3.6) and (3.7)). In computation, the precision of XPIELM is much higher than that of XPIFNN. Therefore, the XPIELM is mainly suggested in this study. The result of XPIFNN is also presented for comparison with both the PIFNN (based on (3.6) and (3.7)) and the XPIELM.

For the XPIFNN, the parameters ϑ in the minimization problem (4.8) are updated using the stochastic gradient descent (SGD) methods [15]. The optimization process of XPIELM based on (4.8) can be executed by solving a least square problem, see [52,66–68] for instance. We specify the algorithm of XPIELM below.

For simplicity of description, we denote the singular functions in (4.2) and (4.4) by $\eta_0, \dots, \eta_n$, which are (4.2) with $n = 2$ for the 2D Poisson crack problem and (4.4) with $n = 4$ for the 2D and 3D elasticity crack problems, respectively. For the XPIELM, we express each of the $\mathbf{u}_{\vartheta_l}, l = 0, \dots, n$ in (4.5) using the components of output neurons of ELM, they are:

$$\mathbf{u}_{\vartheta_l} = u_{\vartheta_{l1}}\cdot\mathbf{e}_1+\cdots+u_{\vartheta_{ld}}\cdot\mathbf{e}_d,$$

where $u_{\vartheta_{lj}}$ is the output of the ELM as well as the jth component of the $\mathbf{u}_{\vartheta_l}$, and the $\mathbf{e}_j$ is the d-dimensional standard unit column vector, where the jth component is 1 and the others are 0. For the Poisson crack problem, we have $\mathbf{u}_{\vartheta_l} = u_{\vartheta_{l1}}$ and $d = 1$ because it is the scalar-valued problem. Therefore, we create $(n+1)d$ ELM models for the XPIELM method. For each $u_{\vartheta_{lj}}$ ELM model, we have

$$u_{\vartheta_{lj}} = \sum_{k=1}^K m_{lj}^k\zeta_{lj}^k(\mathbf{x}) = \sum_{k=1}^K m_{lj}^k\zeta(\mathbf{x}\cdot(\mathbf{W}_{lj}^k)^T+b_{lj}^k), \tag{4.9}$$

where $m_{lj}^k, k = 1, 2, \dots, K$, are the output weights of the ELM model, ζ is the tanh activation function, $\mathbf{W}_{lj}^k$ and $b_{lj}^k, k = 1, \dots, K$ are random numbers sampled in advance that are not updated in the training process. The structure of the XPIELM is shown in Fig. 5.

Feeding the sampling points into the pipeline of the XPIELM, we have the residual terms (4.7). Denote the trained parameters m_{lj}^k by

$$\mathbf{m} = \left[\mathbf{m}_1, \dots, \mathbf{m}_d\right]^T, \quad \mathbf{m}_j = \left[\mathbf{m}_{0j}, \dots, \mathbf{m}_{nj}\right], \ \mathbf{m}_{lj}(k) = m_{lj}^k, \ \ j = 1, \dots, d, \ \ l = 0, \dots, n, \ \ k = 1, \dots, K.$$

Then the optimization problem (4.8) is equivalent to solving the least square problem in what follows:

$$\min_{\mathbf{m}} \|\mathbf{H}\mathbf{m} - \mathbf{S}\|_2^2, \tag{4.10}$$

where the matrix

$$\mathbf{H} = \left[\mathbf{H}_1, \dots, \mathbf{H}_d\right], \ \mathbf{H}_j = \begin{bmatrix} \mathbf{H}_{0j}^{\Omega} & \cdots & \mathbf{H}_{nj}^{\Omega} \\ \mathbf{H}_{0j}^{b} & \cdots & \mathbf{H}_{nj}^{b} \\ \mathbf{H}_{0j}^{\Gamma_O^+} & \cdots & \mathbf{H}_{nj}^{\Gamma_O^+} \\ \mathbf{H}_{0j}^{\Gamma_O^-} & \cdots & \mathbf{H}_{nj}^{\Gamma_O^-} \end{bmatrix}, \quad j = 1, \dots, d,$$

and the column vector

$$\mathbf{S} = \begin{bmatrix} \mathbf{F} \\ \mathbf{G} \\ \mathbf{0} \end{bmatrix}. \tag{4.11}$$

Here, for $l = 0, \dots, n, j = 1, \dots, d$, we have

$$\mathbf{H}_{lj}^{\Omega} = \sqrt{\frac{1}{N_0 + N_1}} \begin{bmatrix} -\nabla\sigma(\eta_l\zeta_{lj}^1(\mathbf{x}_1^{\Omega})\cdot\mathbf{e}_j) & -\nabla\sigma(\eta_l\zeta_{lj}^2(\mathbf{x}_1^{\Omega})\cdot\mathbf{e}_j) & \cdots & -\nabla\sigma(\eta_l\zeta_{lj}^K(\mathbf{x}_1^{\Omega})\cdot\mathbf{e}_j) \\ -\nabla\sigma(\eta_l\zeta_{lj}^1(\mathbf{x}_2^{\Omega})\cdot\mathbf{e}_j) & -\nabla\sigma(\eta_l\zeta_{lj}^2(\mathbf{x}_2^{\Omega})\cdot\mathbf{e}_j) & \cdots & -\nabla\sigma(\eta_l\zeta_{lj}^K(\mathbf{x}_2^{\Omega})\cdot\mathbf{e}_j) \\ \vdots & \vdots & \vdots & \vdots \\ -\nabla\sigma(\eta_l\zeta_{lj}^1(\mathbf{x}_{N_0+N_1}^{\Omega})\cdot\mathbf{e}_j) & -\nabla\sigma(\eta_l\zeta_{lj}^2(\mathbf{x}_{N_0+N_1}^{\Omega})\cdot\mathbf{e}_j) & \cdots & -\nabla\sigma(\eta_l\zeta_{lj}^K(\mathbf{x}_{N_0+N_1}^{\Omega})\cdot\mathbf{e}_j) \end{bmatrix},$$

$$\mathbf{H}_{lj}^{b} = \sqrt{\frac{\alpha}{N_{\Gamma_b}}} \begin{bmatrix} \eta_l\zeta_{lj}^1(\mathbf{x}_1^{\Gamma})\cdot\mathbf{e}_j & \eta_l\zeta_{lj}^2(\mathbf{x}_1^{\Gamma})\cdot\mathbf{e}_j & \cdots & \eta_l\zeta_{lj}^K(\mathbf{x}_1^{\Gamma})\cdot\mathbf{e}_j \\ \eta_l\zeta_{lj}^1(\mathbf{x}_2^{\Gamma})\cdot\mathbf{e}_j & \eta_l\zeta_{lj}^2(\mathbf{x}_2^{\Gamma})\cdot\mathbf{e}_j & \cdots & \eta_l\zeta_{lj}^K(\mathbf{x}_2^{\Gamma})\cdot\mathbf{e}_j \\ \vdots & \vdots & \vdots & \vdots \\ \eta_l\zeta_{lj}^1(\mathbf{x}_{N_b}^{\Gamma})\cdot\mathbf{e}_j & \eta_l\zeta_{lj}^2(\mathbf{x}_{N_b}^{\Gamma})\cdot\mathbf{e}_j & \cdots & \eta_l\zeta_{lj}^K(\mathbf{x}_{N_b}^{\Gamma})\cdot\mathbf{e}_j \end{bmatrix},$$

$$\mathbf{H}_{lj}^{\Gamma_O^+} = \sqrt{\frac{\beta_0}{N_{\Gamma_O}}} \begin{bmatrix} \sigma(\eta_l^+\zeta_{lj}^1(\mathbf{x}_1^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \sigma(\eta_l^+\zeta_{lj}^2(\mathbf{x}_1^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \cdots & \sigma(\eta_l^+\zeta_{lj}^K(\mathbf{x}_1^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O \\ \sigma(\eta_l^+\zeta_{lj}^1(\mathbf{x}_2^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \sigma(\eta_l^+\zeta_{lj}^2(\mathbf{x}_2^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \cdots & \sigma(\eta_l^+\zeta_{lj}^K(\mathbf{x}_2^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O \\ \vdots & \vdots & \vdots & \vdots \\ \sigma(\eta_l^+\zeta_{lj}^1(\mathbf{x}_{N_{\Gamma_O}}^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \sigma(\eta_l^+\zeta_{lj}^2(\mathbf{x}_{N_{\Gamma_O}}^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \cdots & \sigma(\eta_l^+\zeta_{lj}^K(\mathbf{x}_{N_{\Gamma_O}}^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O \end{bmatrix},$$

$$\mathbf{H}_{lj}^{\Gamma_O^-} = \sqrt{\frac{\beta_1}{N_{\Gamma_O}}} \begin{bmatrix} \sigma(\eta_l^-\zeta_{lj}^1(\mathbf{x}_1^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \sigma(\eta_l^-\zeta_{lj}^2(\mathbf{x}_1^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \cdots & \sigma(\eta_l^-\zeta_{lj}^K(\mathbf{x}_1^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O \\ \sigma(\eta_l^-\zeta_{lj}^1(\mathbf{x}_2^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \sigma(\eta_l^-\zeta_{lj}^2(\mathbf{x}_2^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \cdots & \sigma(\eta_l^-\zeta_{lj}^K(\mathbf{x}_2^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O \\ \vdots & \vdots & \vdots & \vdots \\ \sigma(\eta_l^-\zeta_{lj}^1(\mathbf{x}_{N_{\Gamma_O}}^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \sigma(\eta_l^-\zeta_{lj}^2(\mathbf{x}_{N_{\Gamma_O}}^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O & \cdots & \sigma(\eta_l^-\zeta_{lj}^K(\mathbf{x}_{N_{\Gamma_O}}^{\Gamma_O})\cdot\mathbf{e}_j)\cdot\vec{n}_O \end{bmatrix}.$$

where

$$\eta_l^+(\mathbf{x}) = \lim_{\mathbf{y}\to\mathbf{x},\mathbf{y}\in\Omega_0} \eta_l(\mathbf{y}), \ \ \eta_l^-(\mathbf{x}) = \lim_{\mathbf{y}\to\mathbf{x},\mathbf{y}\in\Omega_1} \eta_l(\mathbf{y}), \ \ \forall\ \mathbf{x} \ \in \Gamma_O.$$

In the column vector, **S**(4.11), we have

$$\mathbf{F} = \sqrt{\frac{1}{N_0 + N_1}} \begin{bmatrix} \mathbf{f}(\mathbf{x}_1^{\Omega}) \\ \mathbf{f}(\mathbf{x}_2^{\Omega}) \\ \vdots \\ \mathbf{f}(\mathbf{x}_{N_0+N_1}^{\Omega}) \end{bmatrix}, \quad \mathbf{G} = \sqrt{\frac{\alpha}{N_{\Gamma_b}}} \begin{bmatrix} \mathbf{g}(\mathbf{x}_1^{\Gamma}) \\ \mathbf{g}(\mathbf{x}_2^{\Gamma}) \\ \vdots \\ \mathbf{g}(\mathbf{x}_{N_b}^{\Gamma}) \end{bmatrix}$$

and the zero column vector $\mathbf{0}$ with $4N_{\Gamma_O}$ dimensions. We note that for the Poisson crack problem, $\mathbf{H} = \mathbf{H}_1$ and $\mathbf{m} = \mathbf{m}_1^T$.

The solution of (4.10) is $\mathbf{m} = \mathbf{H}^{\dagger}\mathbf{S}$, where $\mathbf{H}^{\dagger}$ is the pseudo inverse of $\mathbf{H}$. The least square problem, (4.10), can be solved by available algorithms, such as, *torch.linalg.lstsq* in the PyTorch library [17]. PyTorch is a common machine learning library focusing on both usability and speed, and is consistent with other popular scientific computing libraries. The XPIELM algorithm for the crack problem (2.1)–(2.2) is summarized in Algorithm 1 below.

The stress intensity factor (SIF) is crucial to quantify the severity of a crack tip stress. In our computations, the XPIELM yields very high-precision approximation solution, $\mathbf{u}_{\vartheta}$. Therefore, the SIF computed based on this solution is predicted to produce high-precision approximation to the SIF. This is verified in the numerical experiments below. To compute the SIF, we adopt the equivalent integration method [69,70] to calculate the J-integral J. For the 2D crack problem, let the strain energy density factor

$$\omega = \frac{1}{2}\boldsymbol{\sigma}\boldsymbol{\epsilon}^T, \ \ \boldsymbol{\sigma} = [\sigma_x, \sigma_{xy}, \sigma_y], \ \ \boldsymbol{\epsilon} = [\epsilon_x, \epsilon_{xy}, \epsilon_y],$$

Algorithm 1: XPIELM algorithm for solving the LEFM (2.1)–(2.2).

1: Generating a set of random sampling points in Ω, consisting of $\{\mathbf{x}_i^{\Omega}\}_{i=1}^{N_0+N_1}$ on the interior of Ω, $\{\mathbf{x}_i^{\Gamma}\}_{i=1}^{N_b}$ on the boundary Γ, and $\{\mathbf{x}_i^{\Gamma_{\bar{O}}}\}_{i=1}^{N_{\Gamma_{\bar{O}}}}$ on the crack surface Γ_O.

2: Selecting $(n+1)d$ ELM models for the XPIELM and randomly initializing weights and biases of each ELM hidden layer, which are fixed in the hidden layers.

3: Computing the hidden layers output matrix $\mathbf{H}$ and the vector $\mathbf{S}$ in (4.10).

4: Computing the output weights $\mathbf{m}$ using from (4.10).

and J-integral

$$J = \int_A \left[\left(\sigma_{xx}\frac{\partial \mathbf{u}_{\vartheta,1}}{\partial x} + \tau_{xy}\frac{\partial \mathbf{u}_{\vartheta,2}}{\partial x} - \omega\right)\frac{\partial q}{\partial x} + \left(\tau_{xy}\frac{\partial \mathbf{u}_{\vartheta,1}}{\partial x} + \sigma_{yy}\frac{\partial \mathbf{u}_{\vartheta,2}}{\partial x}\right)\frac{\partial q}{\partial y}\right] dV, \tag{4.12}$$

A is a ring integration domain (see Fig. 11 for example) with inner and outer boundaries denoted by Γ_{in} and Γ_{out}, respectively, q is a reference function that is 1 on Γ_{in}, 0 on Γ_{out}. q is not sensitive for the computation of J [69]. Then the SIF, K_{SIF}, is given by:

$$K_{SIF} = \sqrt{\overline{E}J}, \tag{4.13}$$

where $\overline{E} = E$ for the plan stress, $\overline{E} = \frac{E}{1-\nu^2}$ for the plan strain.

5. Numerical experiments

We consider three LEFM models (2.1)–(2.2) with the cracked domain Ω, including the 2D Poisson crack problem, the 2D elasticity crack problem, and the 3D elasticity fully edge-crack problem. Their domains and manufactured solutions will be prescribed in the following sub-sections. For the elasticity problems, we take the Young's modulus $E = 90$, the Poisson's ratio $\nu = 0.28$, and the relevant Lamé constants $\lambda = 44.7443$ and $\mu = 35.1563$. We test the following methods for comparison:

- PIFNN–the piecewise FCNN based on (3.6) and (3.7),
- PIELM–the piecewise ELM based on (3.6) and (3.7),
- XPIFNN–the extend FCNN based on (4.8),
- XPIELM–the extend ELM based on (4.8) and (4.10).

where the XPIELM that achieves high precision is suggested in this study. For all the crack problems, we compute the approximation solutions (denoted by $\mathbf{u}_\vartheta$) using these methods and compare their relative error in L^2 norm:

$$\frac{\|\mathbf{u}-\mathbf{u}_\vartheta\|_{L^2(\Omega;\mathbb{R}^2)}}{\|\mathbf{u}\|_{L^2(\Omega)}},$$

and the relative gradient error in H^1 seminorm [19]:

$$\frac{\|\nabla(\mathbf{u}-\mathbf{u}_\vartheta)\|_{L^2(\Omega)}}{\|\nabla\mathbf{u}\|_{L^2(\Omega)}}.$$

To compute the relative L^2 error and H^1 seminorm error, we randomly generate $N_{test} = 5000$ points in the Ω, respectively. To visualize the results, we generate a uniform set of $Q\times Q$ grid points ($Q = 501$) for drawing relevant pictures. For 2D elasticity problem, we also compute the SIFs based on the XPIELM solution by means of the J-integral method [69,70] to show the performance of the proposed method.

We employ the deep learning framework PyTorch (version 2.1.1) to develop the code. For all methods, the tanh function serves as the activation function. The NN structures and associated parameters will be described in the following sub-sections. To train the various NN methods, we randomly sample $N_0 = N_1 = 1000$ points on the interior of Ω, $N_b = 400d$ points on the boundary Γ ($d = 2, 3$ in the 2D and 3D problems, respectively), and $N_{\Gamma_O} = 100$ points on the crack surface Γ_O. For the PIFNN and PIELM methods, we generate $N_{\Gamma_{\bar{O}}} = 100$ points on the non-crack part, $\Gamma_{\bar{O}}$ (see (3.6)). For FCNNs, we choose the *Adam* optimizer to train the parameters, set the learning rate as 2×10^{-3}, and the decay rate is 0.95 with 1000 steps decay interval, meaning that every 1000 steps we decay the learning rate 5%. For ELMs, the linear least squares methods to calculate the parameters are based on routines *torch.linalg.lstsq* from the PyTorch library, and $Rm = 1$. We now present our numerical results in the following sub-sections.

5.1. The 2D Poisson crack problem

For the 2D Poisson crack problem, we consider a cracked domain $\Omega = [-0.5, 0.5]^2$, with the crack line Γ_O given by $\Gamma_O = \{-0.5 \le x \le 0, y = 0\}$ and the crack tip O at the origin. We use a manufactured solution u as follows:

$$u(r,\theta) = r^{\frac{1}{2}}\sin(\frac{1}{2}\theta) + r^{\frac{3}{2}}\sin(\frac{3}{2}\theta) + r^{\frac{5}{2}}\sin(\frac{5}{2}\theta) + r^{\frac{7}{2}}\sin(\frac{7}{2}\theta), \tag{5.1}$$

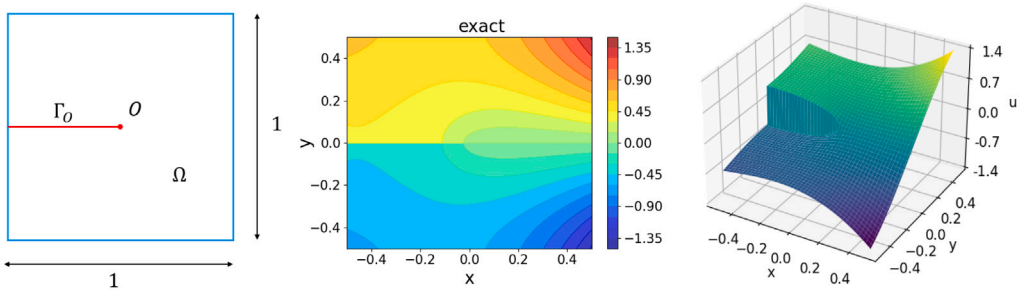

Fig. 6. The cracked domain Ω (left) and the exact solution (5.1) (middle and right) of the 2D Poisson crack problem.

Table 1
Relative errors for the 2D Poisson crack problem.

Methods	L^2 norm	H^1 seminorm
PIFNN	3.823×10^{-2}	1.837×10^{-1}
PIELM	6.089×10^{-2}	2.264×10^{-1}
XPIFNN	3.783×10^{-3}	1.016×10^{-2}
XPIELM	2.922×10^{-14}	1.335×10^{-13}

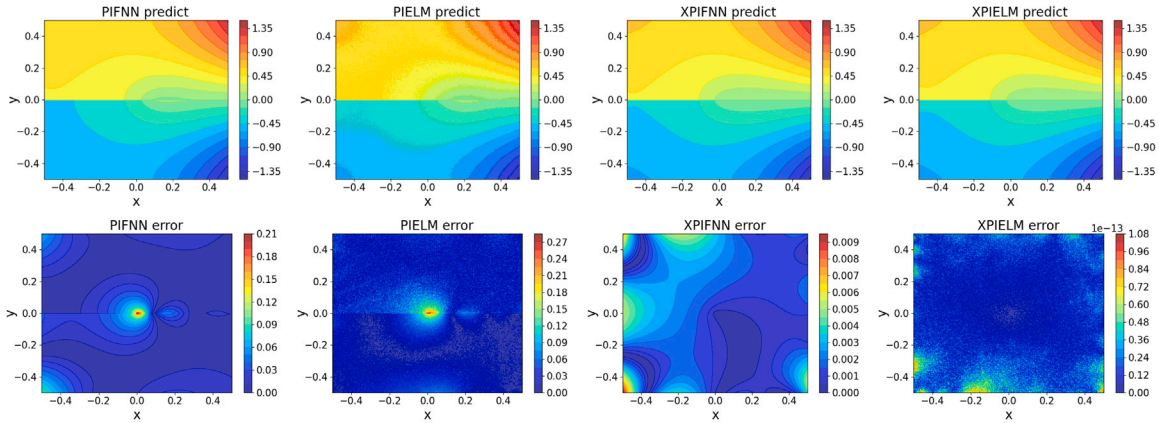

Fig. 7. The point-wise distributions of the four methods' predict solution u_ϑ and the absolute error $|u_\vartheta - u|$ for the 2D Poisson crack problem.

where (r, θ) is the polar coordinate with the pole at the crack tip O and the polar line parallel to the x-axis. f (=0) and the boundary function g in (2.1)–(2.2) are calculated from u and the equation. The domain and the exact solution (5.1) are shown in Fig. 6.

For the PIFNN and the XPIFNN methods, the structure of two piecewise NNs and the extended NNs is chosen as the FCNN which have 4 layers and 30 neurons on each layer. To train the FCNNs, we iterate 20 000 steps using the *Adam* optimizer from PyTorch. In addition, the penalty parameters of the loss functions (see (3.6) and (4.7)) is set as 1. For the PIELM and the XPIELM methods, the number of ELM's hidden layer neurons is $K = 200$.

The relative L_2 errors and H^1 seminorm errors of the four methods are shown in Table 1. It can be seen that the XPIELM and XPIFNN methods perform much better than the PIFNN and PIELM methods. This means that the extended NN methods essentially improve the precision compared with their preliminary versions. The errors of XPIELM reaches the highest precision, about 10^{-14} for the relative L^2 error and about 10^{-13} for the relative H^1 seminorm error. Fig. 7 exhibits point-wise distributions of the predicted solution, u_ϑ, and the absolute error $|u_\vartheta - u|$. It can be observed that the errors of the PIFNN and the PIELM methods are very large around the crack tip O, which means that these methods cannot simulate the singularity around the crack tip O efficiently. The errors of the XPIFNN and the XPIELM methods are much smaller than the PIFNN and PIELM, and the singular solution can be well approximated around the crack tip O.

We test the sensitivity of penalty parameters in (4.7), i.e., α, β_0, β_1. To this end, we take different parameter values in (4.7) to train the approximation solutions and see the error levels. Since β_0, β_1 are all associated with the penalty of the action free condition, we set $\beta_0 = \beta_1 = \beta$. We take $\alpha, \beta = 0.01, 0.1, 1, 10, 100$, and the relative L^2 error of the associated XPIFNN and XPIELM are presented in Fig. 8. For the XPIFNN method, the penalty parameter α is sensitive, and the greater α is, the smaller the error is. The parameter β is not so sensitive. On the contrary, for the proposed XPIELM, both α and β are not sensitive to the relative L^2 errors, which range between 10^{-14} and 10^{-12}.

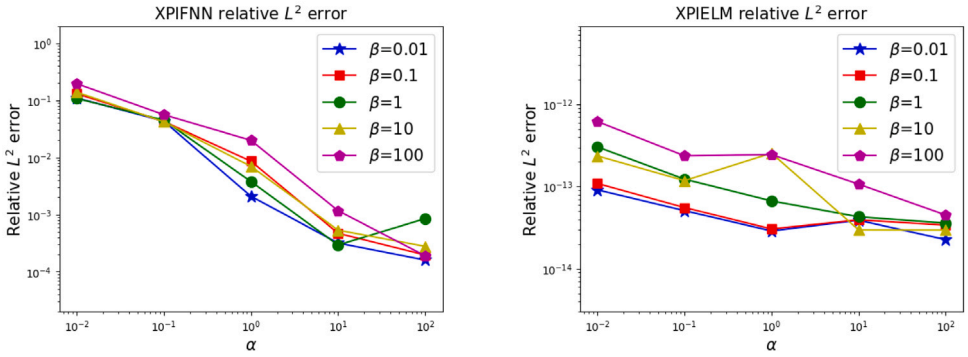

Fig. 8. The relative L^2 errors of the XPIFNN and XPIELM methods with respect to the various penalty parameters in (4.7) for the 2D Poisson crack problem, and $\alpha, \beta = 0.01, 0.1, 1, 10, 100$. For each error line, β is fixed, and α varies from 0.01 to 100. Left: XPIFNN; right: XPIELM.

Table 2
Relative L^2 errors with different number of boundary points N_b for the 2D Poisson crack problem.

Methods	$N_b = 400$	$N_b = 800$	$N_b = 1200$	$N_b = 2000$	$N_b = 4000$
XPIFNN	3.722×10^{-3}	3.783×10^{-3}	3.305×10^{-3}	5.174×10^{-3}	2.941×10^{-3}
XPIELM	7.628×10^{-14}	2.922×10^{-14}	4.255×10^{-14}	9.137×10^{-14}	5.265×10^{-14}

We next investigate effect of the number of boundary sampling points, N_b, on the accuracy. We use the same parameters and interior sampling points as in the above computation, and vary the number of boundary sampling points, N_b, from 400 to 4000. The relative L^2 errors of XPIFNN and XPIELM are listed in Table 2. We find that the number of boundary points N_b seems to have little effect on the relative L^2 errors, and increasing the number of boundary sampling points does not reduce the errors.

5.2. The 2D elasticity crack problem

For the 2D elasticity crack problem, we consider a domain $\Omega = [-1, 1]^2$ with crack line $\Gamma_O = \{-1 \le x \le 0, y = 0\}$ and the crack tip O at the origin. We use a manufactured solution in [72]:

$$\mathbf{u} = \frac{K_I}{E}\sqrt{\frac{r}{2\pi}}\begin{bmatrix}\cos\frac{\theta}{2}\\ \sin\frac{\theta}{2}\end{bmatrix}(a + b\cos\theta) + \frac{4}{3\lambda + 3\mu}r^{\frac{3}{2}}\begin{bmatrix}\cos\frac{\theta}{2}\left((3\lambda + 7\mu)\cos^2\frac{\theta}{2} - 3\lambda - 6\mu\right)\\ \sin\frac{\theta}{2}\left((\lambda + 5\mu)\cos^2\frac{\theta}{2} - \lambda - 2\mu\right)\end{bmatrix}, \tag{5.2}$$

where $a = 2 + \frac{2\mu}{\lambda+2\mu}, b = -2\frac{\lambda+\mu}{\lambda+2\mu}, c = \frac{\lambda+3\mu}{\lambda+\mu}$, the stress intensity factor (SIF) $K_I = 1$. The two components of the exact solution (5.2) are shown in Fig. 9. It is clear in Fig. 9 that the second component u_2 is discontinuous across the crack line Γ_O.

For the NN structure, we use the FCNNs with 4 hidden layers each of which contains 30 neurons and 2 output neurons for the PIFNN and the XPIFNN methods. We use the ELMs with one hidden layer of $K = 120$ neurons and 1 output neuron for the PIELM and the XPIELM. To train the FCNNs, we optimize 10 000 steps for the PIFNN method and 6000 steps for the XPIFNN. We set the penalty parameters as $\alpha = 10^5, \beta_0 = \beta_1 = 4, \gamma_0 = \gamma_1 = 10^4$ for the PIFNN, $\alpha = 10^6, \beta_0 = \beta_1 = 4, \gamma_0 = \gamma_1 = 10^4$ for the PIELM, and $\alpha = 10^5, \beta_0 = \beta_1 = 4$ for the XPIFNN, while $\alpha = 10^5, \beta_0 = \beta_1 = 4$ for the XPIELM.

Fig. 10 shows the point-wise absolute errors of the two components u_1 and u_2 for the 2D elasticity crack problem. Similarly, we can observe that the two components' absolute errors of the PIFNN and the PIELM methods are large around the crack tip O or along the line $\overline{\Omega}_0 \cap \overline{\Omega}_1$, while the XPIFNN and the XPIELM perform much better than the former. Table 3 presents the relative errors of four methods. We see that the XPIELM reaches about the L^2 and H^1 error order of 10^{-15}, while the other methods' errors are between $10^{-4} \sim 10^{-2}$. The error of XPIFNN is smaller than that of the PIFNN.

In this sub-section, we calculate the SIF K_I according to (4.12) and (4.13) using the XPIELM solution to test its accuracy. To this end, we take a ring sub-domain A with width 0.1, $[-0.5, -0.5]^2 \setminus (-0.4, -0.4)^2$, as shown in Fig. 11. To compute the J-integral (4.12), we portion A into a series of square elements with sides length 0.1 and compute the J-integral on each of them. The function q is continuous that is 1 on Γ_{in} and 0 on Γ_{out}. The integrals in square elements are computed by the standard Gaussian integration rule. The approximation value of SIF computed using the XPIELM solution is denoted by $\tilde{K}_I$, and we have $\tilde{K}_I - K_I$ is about 2.20×10^{-10}. This shows the high precision because of the high precision of the XPIELM solution. The SIF errors of other methods are large, and we do not present them here.

5.3. The fully 3D edge-crack elasticity problem

In this sub-section, we consider the fully 3D edge-crack elasticity problem with the crack domain $\Omega = [0, 1] \times [0, 1] \times [0, \frac{1}{3}]$, the crack surface $\Gamma_O = \{0 \le x \le 1, y = 0.5, 0 \le z \le \frac{1}{3}\}$ and the crack front $O = \{x = 0, y = 0, 0 \le z \le \frac{1}{3}\}$.

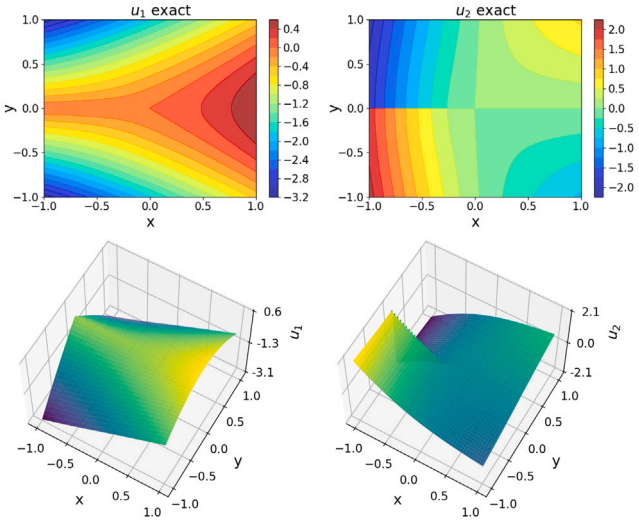

Fig. 9. The exact solution's components u_1 (left column) and u_2 (right column) (5.2) of the 2D elasticity crack problem.

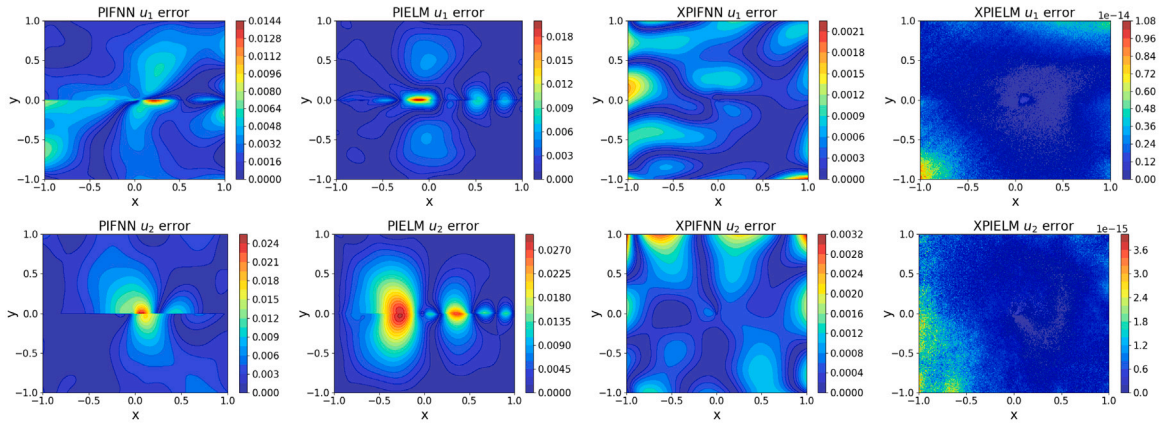

Fig. 10. The absolute errors of the components u_1, u_2 with the proposed methods for the 2D elasticity problem.

Table 3
Relative errors of the components u_1, u_2 for the 2D elasticity crack problem.

Methods	L^2 norm		H^1 seminorm	
	u_1 error	u_2 error	u_1 error	u_2 error
PIFNN	1.792×10^{-3}	4.418×10^{-3}	5.048×10^{-3}	1.226×10^{-2}
PIELM	2.304×10^{-3}	1.006×10^{-3}	1.119×10^{-2}	2.812×10^{-2}
XPIFNN	4.736×10^{-4}	9.936×10^{-4}	1.569×10^{-3}	2.978×10^{-3}
XPIELM	1.430×10^{-15}	7.882×10^{-16}	2.065×10^{-15}	1.676×10^{-15}

We use the singular function (2.8) with function $A(z) = 1 + z + z^2$ in (2.8) as the manufactured solution [63,74]:

$$\mathbf{u} = (1+z+z^2)r^{\frac{1}{2}}\begin{bmatrix}(Q_1-1)\cos\frac{\theta}{2} - \cos\frac{3}{2}\theta\\ (Q_1+1)\sin\frac{\theta}{2} - \sin\frac{3}{2}\theta\\ 0\end{bmatrix} - (1+2z)r^{\frac{3}{2}}\begin{bmatrix}0\\0\\2\cos\frac{\theta}{2} - \frac{2}{3}(Q_1+1)\cos\frac{3}{2}\theta\end{bmatrix}$$
$$+r^{\frac{5}{2}}\begin{bmatrix}(Q_2-Q_3-\frac{Q_1+1}{6}-Q_4)\cos\frac{\theta}{2} + \cos\frac{3}{2}\theta + (Q_4-Q_3)\cos\frac{5}{2}\theta\\ (Q_2+Q_3-\frac{Q_1+1}{6}+Q_4)\sin\frac{\theta}{2} + \sin\frac{3}{2}\theta + (Q_4-Q_3)\sin\frac{5}{2}\theta\\ 0\end{bmatrix}, \tag{5.3}$$

where $Q_1 - Q_4$ are defined in (2.9). The corresponding Cartesian coordinates (x, y, z) in domain Ω are defined by

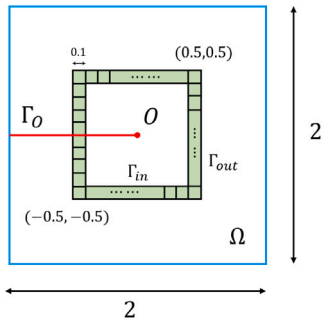

Fig. 11. The ring area A to compute the J-integral (4.12).

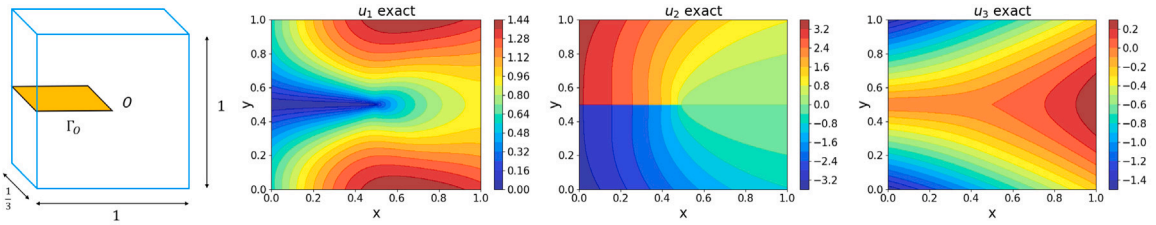

Fig. 12. The crack domain Ω (left one) and the exact solution's components u_1, u_2, u_3 in the $z = 0$ cross section (right three) of the 3D planar elasticity crack problem.

Table 4
Relative errors of the components u_1, u_2 and u_3 for the fully 3D edge-crack elasticity problem.

Methods	L^2 norm			H^1 seminorm		
	u_1 error	u_2 error	u_3 error	u_1 error	u_2 error	u_3 error
PIFNN	5.588×10^{-2}	3.554×10^{-2}	2.135×10^{-2}	3.229×10^{-1}	2.867×10^{-1}	6.714×10^{-2}
PIELM	5.593×10^{-2}	6.222×10^{-2}	3.617×10^{-2}	3.913×10^{-1}	3.981×10^{-1}	1.689×10^{-1}
XPIFNN	2.295×10^{-3}	1.060×10^{-3}	5.324×10^{-3}	1.276×10^{-2}	7.737×10^{-3}	1.429×10^{-2}
XPIELM	6.800×10^{-13}	3.346×10^{-13}	1.485×10^{-12}	1.215×10^{-11}	5.892×10^{-12}	1.100×10^{-12}

$$x = r\cos\theta + 0.5,\ \ y = r\sin\theta + 0.5,\ \ z = z + \frac{1}{6}.$$

The cracked domain and the components of the exact solution are shown in Fig. 12, where the components are shown in the cross section of $z = 0$. From this figure we see that all the components are singular around the crack front O and the u_2 component is discontinuous along with the crack surface Γ_O.

In this test, the NN structure consists of 5 hidden layers each of which contains 60 neurons for the PIFNN and 4 hidden layers each of which contains 30 neurons for the XPIFNN. For the PIELM and the XPIELM 1 hidden layer with 300 neurons is set. To train FCNNs of the PIFNN and the XPIFNN, we optimize 10 000 steps and 4000 steps respectively. The penalty parameters are set as $\alpha = 10^6, \beta_0 = \beta_1 = 4, \gamma_0 = \gamma_1 = 10^4$ for the PIFNN, $\alpha = 10^6, \beta_0 = \beta_1 = 4, \gamma_0 = \gamma_1 = 10^4$ for the PIELM, $\alpha = 10^6, \beta_0 = \beta_1 = 4$ for the XPIFNN, and $\alpha = 10^6, \beta_0 = \beta_1 = 4$ for the XPIELM.

Fig. 13 and Table 4 illustrate the errors of the three components of various methods. Similarly with the sub-sections above, they show that the PIFNN and PIELM cannot approximate the solution efficiently, especially along the line $\overline{\Omega}_0 \cap \overline{\Omega}_1$, and the relative errors are between $10^{-2} \sim 10^{-1}$. Among the four methods, the XPIELM reaches the highest precision, about the order of 10^{-12} for relative L^2 errors and 10^{-11} for relative H^1 seminorm errors.

6. Conclusion and remarks

In this study we proposed the extended neural networks based on two typical machine learning methods, the fully connected neural network and extreme learning machine, for the numerical computation of the crack problem. The main idea was to incorporate the singular information of the crack solution into the neural networks. The precision of the proposed XPIFNN and XPIELM using the physics-informed neural network scheme was improved significantly. Especially, the relative error of the XPIELM reaches about 10^{-12}. This is a significant improvement in comparison with the conventional neural networks (with the relative error about 10^{-3}) when used in the crack problem. The stress intensity factor was also studied based on the XPIELM solution, and the high precision was shown. The proposed XPIELM was applied to the 2D Poisson crack problem, the 2D elasticity problem, and

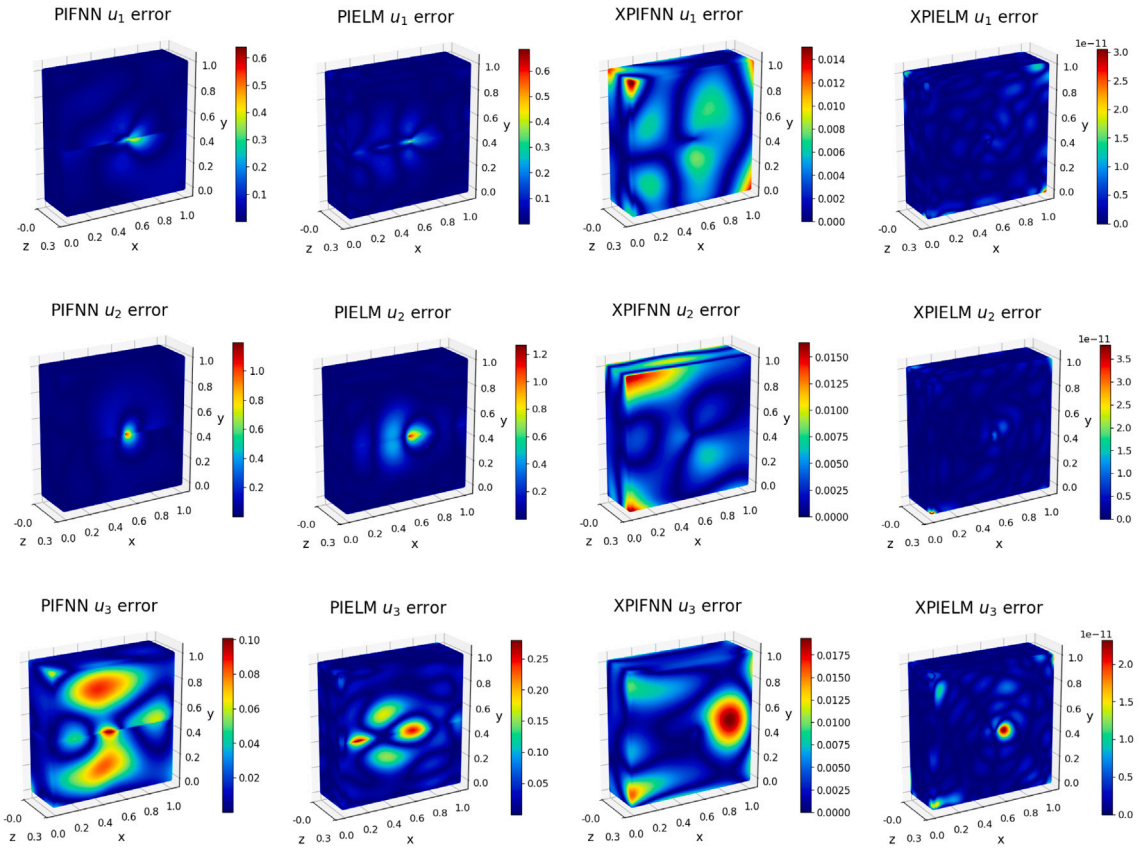

Fig. 13. The absolute error distributions of the components of the four methods for the fully 3D edge-crack elasticity problem. Top row: u_1 component; middle row: u_2 component; bottom row: u_3 component.

the fully 3D edge-crack elasticity problem in a unified way. The extension of the proposed method to crack propagations is under investigation.

CRediT authorship contribution statement

Bokai Zhu: Writing – original draft, Validation, Methodology, Data curation, Conceptualization. **Hengguang Li:** Writing – original draft, Validation, Methodology, Investigation, Formal analysis, Conceptualization. **Qinghui Zhang:** Writing – original draft, 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

Data will be made available on request.

References

[1] T. Belytschko, T. Black, Elastic crack growth in finite elements with minimal remeshing, Internat. J. Numer. Methods Engrg. 45 (1999) 601–620.
[2] C. Daux, N. Moës, J. Dolbow, N. Sukumar, T. Belytschko, Arbitrary branched and intersecting cracks with extended finite element method, Internat. J. Numer. Methods Engrg. 48 (2000) 1741–1760.
[3] J. Oden, C.A. Duarte, Cloud, cracks and FEMs, in: B. Daya Reddy (Ed.), Recent Developments in Computational and Applied Mechanics, CIMNE, Barcelona, Spain, 1997, pp. 302–321.

[4] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack without remeshing, Internat. J. Numer. Methods Engrg. 46 (1999) 131–150.
[5] V. Gupta, C.A. Duarte, I. Babuška, U. Banerjee, A stable and optimally convergent generalized FEM (SGFEM) for linear elastic fracture mechanics, Comput. Methods Appl. Mech. Engrg. 266 (2013) 23–39.
[6] C. Cui, Q. Zhang, Stable generalized finite element methods (SGFEM) for elasticity crack problems, Internat. J. Numer. Methods Engrg. 121 (2020) 3066–3082.
[7] T.P. Fries, T. Belytschko, The extended/generalized finite element method: An overview of the method and its applications, Internat. J. Numer. Methods Engrg. 84 (2010) 253–304.
[8] K. Agathos, S.P.A. Bordas, E. Chatzi, Improving the conditioning of XFEM/GFEM for fracture mechanics problems through enrichment quasi-orthogonalization, Comput. Methods Appl. Mech. Engrg. 346 (2019) 1051–1073.
[9] J.S. Chen, M. ASCE, M. Hillman, S. Chi, Meshfree methods: progress made after 20 years, J. Eng. Mech. 143 (2017) 04017001.
[10] T. Belytschko, Y. Lu, L. Gu, Crack propagation by element-free Galerkin methods, Eng. Fract. Mech. 51 (1995) 295–315.
[11] M. Fleming, Y.A. Chu, B. Moran, T. Belytschko, Enriched element-free Galerkin methods for crack tip fields, Internat. J. Numer. Methods Engrg. 40 (1997) 1483–1504.
[12] T. Belytschko, Y. Lu, L. Gu, M. Tabbara, Element-free galerkin methods for static and dynamic fracture, Int. J. Solids Struct. 32 (1995) 2547–2570.
[13] T. Rabczuk, G. Zi, S.P. Bordas, H. Nguyen-Xuan, A simple and robust three-dimensional cracking-particle method without enrichment, Comput. Methods Appl. Mech. Engrg. 199 (2010) 2437–2455.
[14] S. Ghorashi, N. Valizadeh, S. Mohammadi, Extended isogeometric analysis for simulation of stationary and propagating cracks, Internat. J. Numer. Methods Engrg. 89 (2012) 1069–1101.
[15] I.J. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[16] 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.
[17] A. Paszke, et al., Pytorch: an imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. 32 (2019) 8024–8035.
[18] 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.
[19] S.A. Goraya, N. Sobh, A. Masud, Error estimates and physics informed augmentation of neural networks for thermally coupled incompressible Navier Stokes equations, Comput. Mech. 72 (2023) 267–289.
[20] J. Sirignano, K. Spiliopoulos, DGM: A deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[21] W. E, B. Yu, The deep Ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (2018) 1–12.
[22] E. Samaniego, C. Anitescu, et al., An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Comput. Methods Appl. Mech. Engrg. 362 (2020) 112790.
[23] Y. Zang, G. Bao, X. Ye, H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys. 411 (2020) 109409.
[24] L. Lyu, Z. Zhang, M. Chen, J. Chen, MIM: A deep mixed residual method for solving high-order partial differential equations, J. Comput. Phys. 45 (2022) 110930.
[25] Y. Liao, P. Ming, Deep nitsche method: Deep ritz method with essential boundary conditions, Commun. Comput. Phys. 29 (2021) 1365–1384.
[26] J. Bai, G. Liu, A. Gupta, L. Alzubaidi, X. Feng, Y. Gu, Physics-informed radial basis network (PIRBN): A local approximating neural network for solving nonlinear partial differential equations, Comput. Methods Appl. Mech. Engrg. 415 (2023) 116290.
[27] Z. Wang, M. Chen, J. Chen, Solving multiscale elliptic problems by sparse radial basis function neural networks, J. Comput. Phys. 492 (2023) 112452.
[28] L. Lu, X. Meng, Z. Mao, et al., Deepxde: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228.
[29] J.M. Taylor, P. David, I. Muga, A deep Fourier residual method for solving PDEs using neural networks, Comput. Methods Appl. Mech. Engrg. 405 (2023) 115850.
[30] X. Jin, S. Cai, H. Li, et al., NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations, J. Comput. Phys. 426 (2021) 109951.
[31] H. Sheng, C. Yang, PFNN: A penatly-free neural network method for solving a class of second-order boundary-value problem on complex geometries, J. Comput. Phys. 428 (2021) 110085.
[32] S. Chakraborty, Transfer learning based multi-fidelity physics informed deep neural network, J. Comput. Phys. 426 (2021) 109942.
[33] F. Mostajeran, S.M. Hosseini, Radial basis function neural network (RBFNN) approximation of Cauchy inverse problems of the Laplace equation, Comput. Math. Appl. 141 (2023) 129–144.
[34] D. Zhang, L. Guo, G.E. Karniadakis, Learning in modal space: Solving time-dependent stochastic PDEs using physics-informed neural networks, SIAM J. Sci. Comput. 42 (2) (2020) A639–A665.
[35] N. Sukumar, A. Srivastava, Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks, Comput. Methods Appl. Mech. Engrg. 389 (2022) 114333.
[36] J. Bai, T. Rabczuk, A. Gupta, et al., A physics-informed neural network technique based on a modified loss function for computational 2D and 3D solid mechanics, Comput. Mech. 71 (2023) 543–562.
[37] A.M. Roy, R. Bose, V. Sundararaghavan, R. Arróyave, Deep learning-accelerated computational framework based on physics informed neural network for the solution of linear elasticity, Neural Netw. 162 (2023) 472–489.
[38] D.W. Abueidda, Q. Lu, S. Koric, Meshless physics-informed deep learning method for three-dimensional solid mechanics, Internat. J. Numer. Methods Engrg. 122 (2021) 7182–7201.
[39] J. Baek, J.S. Chen, A neural network-based enrichment of reproducing kernel approximation for modeling brittle fracture, Comput. Methods Appl. Mech. Engrg. 419 (2024) 116590.
[40] Y. Wang, S. Dong, An extreme learning machine-based method for computational PDEs in higher dimensions, Comput. Methods Appl. Mech. Engrg. 418 (2024) 116578.
[41] P.V. Oliva, Y. Wu, C. He, et al., Towards fast weak adversarial training to solve high dimensional parabolic partial differential equations using XNODE-WAN, J. Comput. Phys. 463 (2022) 111233.
[42] S. Zeng, Z. Zhang, Q. Zou, Adaptive deep neural networks methods for high-dimensional partial differential equations, J. Comput. Phys. 463 (2022) 111232.
[43] T. Nakamura-Zimmerer, Q. Gong, W. Kang, Adaptive deep learning for high-dimensional Hamilton–Jacobi-Bellman equations, SIAM J. Sci. Comput. 43 (2) (2021) http://dx.doi.org/10.1137/20M134486X.
[44] S. Zeng, Y. Cai, Q. Zou, Deep neural networks based temporal-difference methods for high-dimensional parabolic partial differential equations, J. Comput. Phys. 468 (2022) 111503.
[45] A. Masud, S.A. Goraya, Variational embedding of measured data in physics-constrained data-driven modeling, J. Appl. Mech. 89 (2022) 111001.
[46] A. Masud, S. Nashar, S.A. Goraya, Physics-constrained data-driven variational method for discrepancy modeling, Comput. Methods Appl. Mech. Engrg. 417 (2023) 116295.
[47] Z. Wang, Z. Zhang, A mesh-free method for interface problems using the deep learning approach, J. Comput. Phys. 400 (2020) 108963.
[48] C. He, X. Hu, L. Mu, A mesh-free method using piecewise deep neural network for elliptic interface problems, J. Comput. Appl. Math. 412 (2022) 114358.

[49] S. Wu, B. Lu, INN: Interfaced neural networks as an accessible meshless approach for solving interface PDE problems, J. Comput. Phys. 470 (2022) 111588.
[50] Y. Liang, Q. Zhang, S. Zeng, A piecewise extreme learning machine for interface problems, Math. Comput. Simulation 227 (2025) 303–321.
[51] Y. Tseng, T. Lin, W. Hu, M. Lai, A cusp-capturing PINN for elliptic interface problems, J. Comput. Phys. 491 (2023) 112359.
[52] F. Calabrò, G. Fabiani, C. Siettos, Extreme learning machine collocation for the numerical solution of elliptic PDEs with sharp gradients, Comput. Methods Appl. Mech. Engrg. 387 (1) (2021) 114–188.
[53] S. Zeng, Y. Liang, Q. Zhang, Adaptive deep neural networks for solving corner singular problems, Eng. Anal. Bound. Elem. 159 (0000) 68–80.
[54] A. Mark, D. Justin, Galerkin neural network approximation of singularly-perturbed elliptic systems, Comput. Methods Appl. Mech. Engrg. 402 (2022) 115169.
[55] Z. Aldirany, R. Cottereau, M. Laforest, S. Prudhomme, Multi-level neural networks for accurate solutions of boundary-value problems, Comput. Methods Appl. Mech. Engrg. 419 (2024) 116666.
[56] J.M. Hanna, J.V. Aguado, S. Comas-Cardona, et al., Residual-based adaptivity for two-phase flow simulation in porous media using physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 396 (2022) 115100.
[57] S. Goswami, M. Yin, Y. Yu, G. Karniadakis, A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials, Comput. Methods Appl. Mech. Engrg. 391 (2022) 114587.
[58] B. Zheng, T. Li, H. Qi, L. Gao, X. Liu, L. Yuan, Physics-informed machine learning model for computational fracture of quasi-brittle materials without labelled data, Int. J. Mech. Sci. 223 (2022) 107282.
[59] M.L. Williams, The bending stress distribution at the base of a stationary crack, J. Appl. Mech. 28 (1961) 78–82.
[60] G.J. Fix, S. Gulati, G.I. Wakoff, On the use of singular functions with finite element approximations, J. Comput. Phys. 13 (1973) 209–228.
[61] Z. Yosibash, Singularities in Elliptic Boundary Value Problems and Elasiticity and their Connection with Failure Initiation, Springer, 2011.
[62] N. Omer, Z. Yosibath, On the path independency of the point-wise J integral in three-dimensions, Int. J. Fract. 136 (2005) 1–36.
[63] V. Gupta, C.A. Duarte, I. Babuška, U. Banerjee, Stable GFEM (SGFEM): Improved conditioning and accuracy of GFEM/XFEM for three-dimensional fracture mechanics, Comput. Methods Appl. Mech. Engrg. 289 (2015) 355–386.
[64] Q. Zhang, I. Babuška, U. Banerjee, Robustness in stable generalized finite element methods (SGFEM) applied to Poisson problems with crack singularities, Comput. Methods Appl. Mech. Engrg. 311 (2016) 476–502.
[65] G. Huang, Q. Zhu, C. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1–3) (2006) 489–501.
[66] V. Dwivedi, B. Srinivasan, Physics informed extreme learning machine (PIELM)–A rapid method for the numerical solution of partial differential equations, Neurocomputing 391 (2020) 96–118.
[67] G. Fabiani, F. Calabrò, L. Russo, et al., Numerical solution and bifurcation analysis of nonlinear partial differential equations with extreme learning machines, J. Sci. Comput. 89 (2021) 44.
[68] S. Dong, J. Yang, Local extreme learning machines and domain decomposition for solving linear and nonlinear partial differential equations, Comput. Methods Appl. Mech. Engrg. 387 (2021) 114129.
[69] B. Moran, C.F. Shih, A general treatment of crack tip contour integrals, Int. J. Fract. 35 (1987) 295–310.
[70] K.N. Shivakumar, I.S. Raju, An equivalent domain integral method for three-dimensional mixed-mode fracture problems, Eng. Fract. Mech. 42 (1992) 935–959.
[71] Q. Zhang, DOF-gathering stable generalized finite element methods (SGFEM) for crack problems, in: accepted in Numerical Methods for Partial Differential Equations, 2020, http://dx.doi.org/10.1002/num.22459.
[72] P. Laborde, J. Pommier, Y. Renard, M. Salaün, High order extended finite element method for cracked domains, Internat. J. Numer. Methods Engrg. 64 (2005) 354–381.
[73] N. Sukumar, N. Moës, B. Moran, T. Belytschko, Extended finite element method for for three dimensional crack modelling, Internat. J. Numer. Methods Engrg. 48 (11) (2000) 1549–1570.
[74] C. Cui, Q. Zhang, U. Banerjee, I. Babuška, Stable generalized finite element method (SGFEM) for three-dimensional crack problems, Num. Math. 152 (2022) 475–509.