Received: 9 June 2022 | Revised: 23 September 2022 | Accepted: 26 September 2022

DOI: 10.1002/nme.7135

RESEARCH ARTICLE

WILEY

Deep learning phase-field model for brittle fractures

Yousef Ghaffari Motlagh[1,2] | **Peter K. Jimack**[1] | **René de Borst**[2]

[1]School of Computing, University of Leeds, Leeds, UK

[2]Department of Civil and Structural Engineering, University of Sheffield, Sheffield, UK

Correspondence
Yousef Ghaffari Motlagh, School of Computing, University of Leeds, Leeds LS2 9JT, UK.
Email: y.ghaffarimotlagh@leeds.ac.uk

Abstract

We present deep learning phase-field models for brittle fracture. A variety of physics-informed neural networks (PINNs) techniques, for example, original PINNs, variational PINNs (VPINNs), and variational energy PINNs (VE-PINNs) are utilized to solve brittle phase-field problems. The performance of the different versions is investigated in detail. Also, different ways of imposing boundary conditions are examined and are compared with a self-adaptive PINNs approach in terms of computational cost. Furthermore, the data-driven discovery of the phase-field length scale is examined. Finally, several numerical experiments are conducted to assess the accuracy and the limitations of the discussed deep learning schemes for crack propagation in two dimensions. We show that results can be highly sensitive to parameter choices within the neural network.

KEYWORDS

brittle fracture, deep learning, finite element method, neural networks, phase-field models, PINNs

1 | INTRODUCTION

Numerical approaches to the modeling of fracture initiation and propagation can be divided into two main categories, namely, discrete crack models and smeared crack models. In discrete crack models, the discontinuities are introduced into the displacement field using interface elements which are inserted in the mesh a priori,[1-4] by means of remeshing,[5-7] or by enriching the basis by inserting discontinuities.[8,9] These methods have been investigated widely and successful applications have been reported. However, robust extensions to complex three-dimensional problems are nontrivial.

Smeared approaches are an alternative, in which the sharp discontinuity is distributed over a small, but finite width.[10] Early smeared approaches appeared to be deficient in the sense that they caused a loss of well-posedness of the boundary value problem at, or close to structural failure. The concomitant grid sensitivity then prevents physically meaningful answers. A host of solutions have been proposed as a remedy, but gradient-enhanced damage models appear to be particularly effective and powerful to model fracture in quasi-brittle and ductile materials.[11]

More recently, the variational approach has become popular as an elegant and mathematically well-founded approach to brittle fracture.[12] In it, the solution to the fracture problem is found as the minimizer of a global energy functional. A phase-field implementation of this model has been proposed by Bourdin et al.[13] and has been cast in a damage-like, engineering format by Miehe et al.[14,15] Indeed, the phase-field approach to brittle fracture can be classified as a smeared approach, and bears much similarity to gradient-enhanced damage models.[16]

Although mathematical and practical data-assimilation endeavors have been growing vastly, the spatiotemporal heterogeneity of available data, along with the lack of universally reliable models, underscores the need for a transformative approach.[17] Machine learning (ML) can be used to explore massive design spaces, identify multi-dimensional correlations

and manage ill-posed problems.[17] Although solving ill-posed inverse problems with conventional solvers can be challenging, PINNs can be easily employed to solve these problems accurately and efficiently. Deep learning approaches can provide tools for naturally extracting features from massive amounts of multi-fidelity observational data that are currently available and characterized by unprecedented spatial and temporal coverage.[18]

Deep learning allows overparameterized neural networks with multiple layers to successively extract higher-level features from the raw input. These networks are well known to deal with supervised learning tasks, which require a large amount of labeled training data. To avoid data collection, which is normally expensive in engineering applications, it is critical to use the method with less data dependency and train deep learning models using primarily constraints (physical laws) rather than data. Physics-informed neural networks (PINNs)[17,19] can be a potential solution. PINNs are capable of leveraging the underlying laws of physics to extract patterns from high-dimensional data generated from experiments. Nevertheless, Krishnapriyan et al.[20] have reported possible issues with PINNs and show that PINNs face some challenges before they can compete with traditional numerical methods in terms of accuracy and computational cost. Publications on physics-informed ML have increased substantially across different disciplines, for example, conservative physics-informed neural networks (cPINNs),[21] fractional physics-informed neural networks (fPINNs),[22] hp variational physics-informed neural networks (hp-VPINNs),[23,24] and extended physics-informed neural networks (XPINNs).[25]

Recently, variational energy-based PINNs (VE-PINNs) methods have been used for solving phase-field problems.[26-29] In this article, we formulate a brittle phase-field model based on PINNs, VPINNs, and VE-PINNs and demonstrate the performance of each PINNs version. Also, we investigate different approaches to impose boundary conditions (BC) such as weakly and strongly applied BC. Additionally, self-adaptive PINNs (SA-PINNs)[30] are studied and compared to PINNs with soft and hard BC impositions. Finally, we investigate the speed of crack propagation for two benchmark problems of fracture.

The remainder of this article is composed as follows. In Section 2, we succinctly summarize the phase-field model for brittle fracture. Formulations of PINNs, VPINNs, and VE-PINNs for brittle phase-field models are explained in Section 3. In Section 4, we discuss and analyze the different PINNs versions for a 1D problem and demonstrate the performance of SA-PINNs against PINNs approaches, and present a data-driven discovery of phase-field length scale. In Section 5, based upon conclusions reached from our extensive 1D studies, a more focused set of 2D numerical experiments are conducted to study the crack propagation path and the propagation velocity.

2 | PHASE-FIELD MODEL FOR BRITTLE FRACTURE

We consider a volume Ω with an internal discontinuity boundary Γ. The position of a material point is determined by the coordinate $\boldsymbol{x}$ in a Cartesian reference frame. Displacement and traction components are prescribed along disjoint parts of the external boundary of the domain, $\partial\Omega_{g_i}$ and $\partial\Omega_{h_i}$, respectively.

2.1 | Variational form of fracture

As the starting point for the derivation of the phase-field approximation to brittle fracture, we consider the total potential energy:[12]

$$\Psi_{\text{pot}} = \int_{\Omega} \psi_e(\boldsymbol{\varepsilon}) \mathrm{d}V + \int_{\Gamma} \mathcal{G}_c \mathrm{d}A. \tag{1}$$

We assume small displacement gradients, and define the infinitesimal strain tensor, $\boldsymbol{\varepsilon}$, with components

$$\varepsilon_{ij} = u_{(i,j)} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \tag{2}$$

as the deformation measure. The displacement components are denoted by u_i. We assume isotropic linear elasticity, such that the elastic energy density is given by

$$\psi_e = \frac{1}{2}\lambda\varepsilon_{ii}\varepsilon_{jj} + \mu\varepsilon_{ij}\varepsilon_{ij} \tag{3}$$

with λ and μ the Lamé constants, and using the Einstein convention. In Equation (1), the fracture energy is denoted by $\mathcal{G}_c$. An irreversibility condition is included which enforces that cracks can only nucleate and propagate, and not heal.[13]

2.2 | Phase-field formulation

The variational approach to brittle fracture[12] determines the nucleation, propagation, and interaction of cracks by finding a global minimizer of the total potential energy. Solving this variational problem numerically for discrete cracks can be difficult because the crack path, Γ, evolves with time. In order to overcome this difficulty, a volumetric approximation to the surface integral has been proposed:[13]

$$\int_{\Gamma} \mathcal{G}_c \mathrm{d}A \approx \int_{\Omega} \mathcal{G}_c \gamma_c \mathrm{d}V. \tag{4}$$

The phase-field approximation introduces a crack density, γ_c, which depends on a length-scale parameter ℓ_0 and the continuous scalar-valued phase-field, $c \in [0, 1]$, to represent the crack, with $c = 0$ away from the crack and $c = 1$ at the crack:[13-15]

$$\gamma_c = \frac{1}{4\ell_0}\left[c^2 + 4\ell_0^2 |\nabla c|^2\right]. \tag{5}$$

Minimizing the above functional under the constraints $c(0) = 1$ and $c(\boldsymbol{x}) \to 0$ as $|\boldsymbol{x}| \to \infty$ leads to the Euler equation:

$$c - 4\ell_0^2 \Delta c = 0. \tag{6}$$

The solution to this equation, c, is the solution to the minimization problem:

$$\mathrm{argmin}(I(c)), \quad I(c) = \int \gamma_c \mathrm{d}V. \tag{7}$$

In 1D, the solution to (6) reads

$$c(x) = e^{-|x|/2\ell_0}. \tag{8}$$

The definition (5) is well-posed variationally for all $c \in H^1(\Omega)$. Note that H^1 is the Sobolev space of functions with square-integrable derivatives.

To model the loss of material stiffness in the failure zone, we follow Miehe et al.[14,15] and define the elastic energy as

$$\psi_e(\boldsymbol{\varepsilon}) = \left[(1-c)^2 + \kappa\right]\psi_e^+(\boldsymbol{\varepsilon}) + \psi_e^-(\boldsymbol{\varepsilon}), \tag{9}$$

where κ is a model parameter which in our simulations we have set $\kappa = 0$. ψ_e^+ and ψ_e^- are the strain energies computed from the positive and negative components of the strain tensor, respectively, which can be defined via a spectral decomposition of the strain tensor,

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^+ + \boldsymbol{\varepsilon}^-, \tag{10}$$

where the former describes the tensile mode and the latter the compressive mode contained in $\boldsymbol{\varepsilon}$. The split is defined based on the spectral decomposition

$$\boldsymbol{\varepsilon}^+ = \sum_{i=1}^{d} \langle \varepsilon_i \rangle^+ \boldsymbol{n}_i \otimes \boldsymbol{n}_i, \tag{11}$$

$$\boldsymbol{\varepsilon}^- = \sum_{i=1}^{d} \langle \varepsilon_i \rangle^- \boldsymbol{n}_i \otimes \boldsymbol{n}_i, \tag{12}$$

where ε_i (with $i = 1, \ldots, d$) are the principal strains and $\boldsymbol{n}_i$ are the corresponding principal directions of the strain tensor and

$$\langle x \rangle^+ = \begin{cases} 0, & x < 0, \\ x, & x \geq 0, \end{cases} \tag{13}$$

$$\langle x \rangle^- = \begin{cases} x, & x < 0, \\ 0, & x \geq 0. \end{cases} \tag{14}$$

Now, ψ_e^+ and ψ_e^- read

$$\psi_e^+(\boldsymbol{\varepsilon}) = \frac{1}{2}\lambda\langle \mathrm{tr}\boldsymbol{\varepsilon}\rangle^2 + \mu \mathrm{tr}[(\boldsymbol{\varepsilon}^+)^2] \tag{15}$$

and

$$\psi_e^-(\boldsymbol{\varepsilon}) = \frac{1}{2}\lambda\langle \mathrm{tr}\boldsymbol{\varepsilon}\rangle^2 + \mu \mathrm{tr}[(\boldsymbol{\varepsilon}^-)^2]. \tag{16}$$

The Lagrange energy functional using (1), (4), (5), and (9) becomes:

$$\mathcal{L}(\boldsymbol{u}, c) = \int_\Omega \left\{(1-c)^2\psi_e^+(\nabla^s \boldsymbol{u}) + \psi_e^-(\nabla^s \boldsymbol{u})\right\} \mathrm{d}V + \int_\Omega \frac{\mathcal{G}_c}{4\ell_0}\left[c^2 + 4\ell_0^2|\nabla c|^2\right] \mathrm{d}V, \tag{17}$$

where the symmetric gradient operator is defined, $\nabla^s : \boldsymbol{u} \rightarrow \boldsymbol{\varepsilon}$, as a mapping from the displacement field to the strain field. In order to obtain the strong form of the governing equations, the Euler–Lagrange equations are utilized

$$(S)\begin{cases} \frac{\partial \sigma_{ij}}{\partial x_j} = 0, & \boldsymbol{x} \in \Omega, \\ \left(\frac{4\ell_0\psi_e^+}{\mathcal{G}_c} + 1\right) c - 4\ell_0^2 \frac{\partial^2 c}{\partial x_i^2} = \frac{4\ell_0\psi_e^+}{\mathcal{G}_c}, & \boldsymbol{x} \in \Omega, \end{cases} \tag{18}$$

where σ_{ij} is the Cauchy stress tensor and is defined as

$$\sigma_{ij} = (1-c)^2 \frac{\partial \psi_e^+}{\partial \varepsilon_{ij}} + \frac{\partial \psi_e^-}{\partial \varepsilon_{ij}}. \tag{19}$$

There is nothing in the formulation so far to prevent cracks from healing if loads are removed. For a detailed discussion on how irreversibility can be enforced, see Miehe et al.[14,15] The idea is to replace the strain energy in the phase-field equation, for example in $(18)_2$, by a strain-energy history, $\mathcal{H}$, which acts as a threshold and which satisfies the Karush–Kuhn–Tucker loading/unloading conditions:

$$\psi_e^+ - \mathcal{H} \leq 0, \quad \dot{\mathcal{H}} \geq 0, \quad \dot{\mathcal{H}}(\psi_e^+ - \mathcal{H}) = 0. \tag{20}$$

Now, by substituting $\mathcal{H}$ for ψ_e^+ in $(18)_2$ the final version of the strong form equations read

$$(S)\begin{cases} \frac{\partial \sigma_{ij}}{\partial x_j} = 0, & \boldsymbol{x} \in \Omega, \\ \left(\frac{4\ell_0\mathcal{H}}{\mathcal{G}_c} + 1\right) c - 4\ell_0^2 \frac{\partial^2 c}{\partial x_i^2} = \frac{4\ell_0\mathcal{H}}{\mathcal{G}_c}, & \boldsymbol{x} \in \Omega. \end{cases} \tag{21}$$

The strong form of equations is complemented by the following boundary conditions

$$(S : BC)\begin{cases} u = g_i, & \boldsymbol{x} \in \partial\Omega_{g_i}, \\ \sigma_{ij} n_j = h_i, & \boldsymbol{x} \in \partial\Omega_{h_i}, \\ \frac{\partial c}{\partial x_i} n_i = 0, & \boldsymbol{x} \in \partial\Omega, \end{cases} \tag{22}$$

with $g_i(x)$ and $h_i(x)$ being prescribed on $\partial\Omega_{g_i}$ and $\partial\Omega_{h_i}$, respectively and with $\boldsymbol{n}(\boldsymbol{x})$ being the outward-pointing normal vector of the boundary. The initial crack is modeled as an induced crack in the phase-field.[31,32] An initial strain-history field, $\mathcal{H}_0$, is utilized to defined an initial crack in the phase-field. The initial strain-history field can be defined as

$$\mathcal{H}_0 = \mathcal{B}\begin{cases} \frac{\mathcal{G}_c}{4\ell_0}\left(1 - \frac{d(\boldsymbol{x},l)}{\ell_0}\right), & d(\boldsymbol{x}, l) \leq \ell_0, \\ 0, & d(\boldsymbol{x}, l) > \ell_0, \end{cases} \tag{23}$$

where $d(\boldsymbol{x}, l)$ is the closest distance from $\boldsymbol{x}$ to the line l that represents the discrete crack. Also, $\mathcal{B}$ is a constant and we use $\mathcal{B} = 1000$ in our work following Borden et al.[31]

For the variational form of momentum and phase-field equations, we construct the trial solution, $\mathcal{S}_u$, for the displacements and $\mathcal{S}_c$, for the phase-field as

$$\boldsymbol{\mathcal{S}}_{\boldsymbol{u}} = \left\{ \boldsymbol{u} \in \left(H^1(\Omega)\right)^d | u_i = g_i \quad \text{on} \quad \partial\Omega_{g_i} \right\}, \tag{24}$$

$$\mathcal{S}_c = \left\{ c \in H^1(\Omega) \right\}. \tag{25}$$

Likewise, the weighting (test) function spaces are defined as

$$\boldsymbol{\mathcal{V}}_{\boldsymbol{u}} = \left\{ \boldsymbol{v} \in \left(H^1(\Omega)\right)^d | v_i = 0 \quad \text{on} \quad \partial\Omega_{g_i} \right\}, \tag{26}$$

$$\mathcal{V}_c = \left\{ q \in H^1(\Omega) \right\}. \tag{27}$$

The weak formulation can be obtained by multiplying equations (21) by the appropriate weighting functions and performing integration by parts, as follows:

$$(W) \begin{cases} \text{Given } \boldsymbol{g} \text{ and } \boldsymbol{h} \text{ find } \boldsymbol{u} \in \boldsymbol{\mathcal{S}}_{\boldsymbol{u}} \text{ and } c \in \mathcal{S}_c, \text{ such that for all} \\ \boldsymbol{v} \in \boldsymbol{\mathcal{V}}_{\boldsymbol{u}} \text{ and for all } q \in \mathcal{V}_c, \\ (\boldsymbol{\sigma}, \nabla \boldsymbol{v})_\Omega = (\boldsymbol{h}, \boldsymbol{v})_{\partial\Omega_h}, \\ \left(\left(\frac{4\ell_0 \mathcal{H}}{\mathcal{G}_c} + 1 \right) c, q \right)_\Omega + \left(4\ell_0^2 \nabla c, \nabla q \right)_\Omega = \left(\frac{4\ell_0 \mathcal{H}}{\mathcal{G}_c} \right)_\Omega, \end{cases} \tag{28}$$

where $(.,.)_\Omega$ is the L^2 inner product on Ω.

3 | PHYSICS INFORMED NEURAL NETWORK FOR PHASE-FIELD MODELS

In this section, we present three versions of the PINNs formulation for the phase-field model for the brittle fracture problem, which are: standard PINNs,[19] variational PINNs (VPINNs),[23,24] and variational energy based PINNs (VE-PINNs).[26,28] Here, we consider a monolithic scheme to solve the phase-field problem.

We consider the strong form of equations (21) and assume that $\mathcal{F}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b})$ is a neural network (NN) approximation of the displacements, $\boldsymbol{u}$, and the phase-field, c in (18) and (22). Particularly, a NN is comprised of ℓ hidden layers with $\mathcal{N}_i$ neurons in each layer and activation functions σ:

$$\mathcal{F}_{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}) = \mathcal{K} \circ T^{(\ell)} \circ T^{(\ell-1)} \circ \cdots \circ T^{(1)}(\boldsymbol{x}), \tag{29}$$

where $\mathcal{K} : \mathbb{R}^{\mathcal{N}_\ell \times d} \to \mathbb{R}^d$ is the linear mapping in the output and d is the input dimension; $\boldsymbol{T}^i(\cdot) = \sigma\left(\boldsymbol{W}^i \times \cdot + \boldsymbol{b}^i\right)$ is the nonlinear mapping in each hidden layer $i = 1, 2, 3, \ldots, \ell$. Note that $\boldsymbol{W}^i$, $\boldsymbol{b}^i$ are the weights and biases. The strong-form residuals $r_k(\boldsymbol{u})$ and $r(c)$, for the momentum and phase-field equations, respectively, and the boundary residual r_u^b can be defined as

$$r^k(\hat{\boldsymbol{u}}) = \frac{\partial \sigma_{kj}}{\partial x_j}, \quad k = 1, \ldots, d, \tag{30}$$

$$r(\hat{c}) = \left(\frac{4\ell \mathcal{H}}{\mathcal{G}_c} + 1 \right) \hat{c} - 4\ell^2 \frac{\partial^2 \hat{c}}{\partial x_i^2} - \frac{4\ell \mathcal{H}}{\mathcal{G}_c}, \tag{31}$$

$$r_b^k(\hat{\boldsymbol{u}}) = \hat{u}_k - g_k, \tag{32}$$

where $\hat{\boldsymbol{u}}$ and $\hat{c}$ are neural network approximations of the displacements and the phase-field, respectively. Now, in order to construct the variational forms of the problem, the weighted integral of the residuals can be defined by mapping them

onto properly chosen spaces of test (weighting) functions $\mathcal{V}$ and $\tilde{\mathcal{V}}$; and then set them to zero. We choose the test functions $v_j^k \in \mathcal{V}$ and $q_j \in \tilde{\mathcal{V}}$ such that

$$\mathcal{R}_j^k(\hat{\boldsymbol{u}}) = \int_\Omega r^k(\hat{\boldsymbol{u}}) v_j^k \mathrm{dV} = 0, \tag{33}$$

$$\mathcal{R}_{bj}^k(\hat{\boldsymbol{u}}) = \int_{\partial\Omega} r_b^k(\hat{\boldsymbol{u}}) v_j^k \mathrm{dA} = 0, \tag{34}$$

$$\tilde{\mathcal{R}}_j(\hat{c}) = \int_\Omega r(\hat{c}) q_j \mathrm{dV} = 0. \tag{35}$$

The following minimization problem can be formulated in place of solving the nonlinear systems resulting from the above equations:

$$\min_{\boldsymbol{W},\boldsymbol{b}} \mathcal{J}(\hat{c}, q, \hat{\boldsymbol{u}}, \boldsymbol{v}), \tag{36}$$

$$\mathcal{J}(\hat{c}, q, \hat{\boldsymbol{u}}, \boldsymbol{v}) = \sum_{j=1}^{N_{rc}} \left(\tilde{\mathcal{R}}_j(\hat{c})\right)^2 + \sum_{k=1}^{d}\sum_{j=1}^{N_{ru}} \left(\mathcal{R}_j^k(\hat{\boldsymbol{u}})\right)^2 + \tau_b \sum_{k=1}^{d}\sum_{j=1}^{N_{bu}} \left(\mathcal{R}_{bj}^k(\hat{\boldsymbol{u}})\right)^2. \tag{37}$$

In (37), the first and second terms represent the weighted integral of the phase-field and momentum residuals, respectively and the last term indicates the weighted integral of the displacement boundary condition. N_{rc} and N_{ru} are the number of test functions corresponding to phase-field and momentum residuals, respectively. N_{bu} is the number of collocation points corresponding the displacement essential boundary conditions. Furthermore, τ_b is a penalty parameter that represents the weight coefficient in the loss function and may be user-specified or tuned manually or automatically,[23] for example, based on the numerical experiment in each problem. Its optimal bound, however, is still an open problem in the literature.[33] It is worthwhile to mention that the *weak* BC enforcement methods, as mentioned above, have several major drawbacks: (1) there is no quantitative guarantee on the accuracy of the BC being imposed and thus the solution could be unsatisfactory; (2) the optimization performance can depend on the relative importance of each term, but how to assign a weight for each term can be difficult. Alternatively, we can impose the BC in a *strong* form, where a particular solution that solely satisfies the boundary condition is added.[34] To do this we can modify the deep neural network state variables, $\hat{\boldsymbol{u}}$. The essential displacement conditions can be imposed by constructing the $\tilde{\boldsymbol{u}}$ as,

$$\tilde{\boldsymbol{u}} = \boldsymbol{g} + G(\boldsymbol{x})\mathcal{F}_{NN}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{b}), \tag{38}$$

where $G(\boldsymbol{x}) = \boldsymbol{0}$ on the Dirchlet boundary. The cost function, $\mathcal{J}$, in (37) for *strong* BC enforcement reads

$$\mathcal{J}(\hat{c}, q, \tilde{\boldsymbol{u}}, \boldsymbol{v}) = \sum_{j=1}^{N_{rc}} \left(\tilde{\mathcal{R}}_j(\hat{c})\right)^2 + \sum_{k=1}^{d}\sum_{j=1}^{N_{ru}} \left(\mathcal{R}_j^k(\tilde{\boldsymbol{u}})\right)^2. \tag{39}$$

3.1 | PINNs formulation

In this subsection, we derive standard PINNs[19] by starting from the variational form, (36) and (37). To this end, we assume each of the test functions to be a Dirac delta function, $v^k(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x}_r)$ and $q(\boldsymbol{x}) = \delta(\boldsymbol{x} - \boldsymbol{x}_{\tilde{r}})$, so that $\boldsymbol{x}_r$ and $\boldsymbol{x}_{\tilde{r}}$ are the collocation points for momentum and phase-field, respectively. Basically, by means of these test functions we can project the residuals onto a finite set of collocation points and enforce the equation to be satisfied at these points. The loss function for the PINNs reads

$$\mathcal{L}_{\text{PINNs}} = \sum_{k=1}^{d} \left[\frac{1}{N_r} \sum_{i=1}^{N_r} \left| r^k(\boldsymbol{x}_r^i) \right|^2 + \tau_b \frac{1}{N_b} \sum_{i=1}^{N_b} \left| r_b^k(\boldsymbol{x}_b^i) \right|^2 \right] + \frac{1}{N_{\tilde{r}}} \sum_{i=1}^{N_{\tilde{r}}} \left| \tilde{r}(\boldsymbol{x}_{\tilde{r}}^i) \right|^2 + \tau_{\tilde{b}} \frac{1}{N_{\tilde{b}}} \sum_{i=1}^{N_{\tilde{b}}} \left| \tilde{r}_{\tilde{b}}(\boldsymbol{x}_{\tilde{b}}^i) \right|^2, \tag{40}$$

where $\tilde{r}_{\tilde{b}} = \frac{\partial \tilde{c}}{\partial x_i}$; and $\{\boldsymbol{x}_r^i\}_{i=1}^{N_r}$, $\{\boldsymbol{x}_{\tilde{r}}^i\}_{i=1}^{N_{\tilde{r}}}$, $\{\boldsymbol{x}_b^i\}_{i=1}^{N_b}$, and $\{\boldsymbol{x}_{\tilde{b}}^i\}_{i=1}^{N_{\tilde{b}}}$ are collocation points in their domains. Note that N_r and $N_{\tilde{r}}$ are the number of collocation points for momentum and phase-field, respectively. In addition, N_b and $N_{\tilde{b}}$ are the number of collocation points corresponding the displacement essential boundary conditions and phase-field natural boundary conditions, respectively. The last term in (40) is the Neumann boundary condition presented in (22).

3.2 | VPINNs and hp-VPINNs formulations

In order to use the weak form of the governing equations, we use the VPINNs formulation introduced in Reference 24. We utilize Legendre polynomials as test functions, that is, $v_j^k(x) = P_{j+1}(x) - P_{j-1}(x),\ j = 1, 2, \dots, K_u,\ q_j(x) = P_{j+1}(x) - P_{j-1}(x),\ j = 1, 2, \dots, K_c$, where $P_j(x)$ are Legendre polynomials; K_u and K_c are the number of test functions corresponding to momentum and phase-field equations, respectively. Gauss quadrature with N_{gauss} quadrature points is performed to compute the integrals. The loss function for VPINNs can be defined as

$$\mathcal{L}_{\text{VPINNs}} = \sum_{k=1}^{d}\left[\frac{1}{K_u}\sum_{j=1}^{K_u}\left|\mathcal{R}_j^k\right|^2 + \tau_b\frac{1}{N_b}\sum_{i=1}^{N_b}\left|r_b^k(\boldsymbol{x}_b^i)\right|^2\right] + \frac{1}{K_c}\sum_{j=1}^{K_c}\left|\tilde{\mathcal{R}}_j\right|^2. \tag{41}$$

Another version of VPINNs is hp-VPINNS presented in Reference 23. In VPINNs, the trial space and test space are both defined globally over the entire computational domain whereas in hp-VPINNs the test space contains piecewise polynomials defined locally. The loss function for hp-VPINNs is given as

$$\mathcal{L}_{\text{hp-VPINNs}} = \sum_{k=1}^{d}\left[\sum_{e=1}^{N_{el}}\left(\frac{1}{K_u^{(e)}}\sum_{j=1}^{K_u^{(e)}}\left|\mathcal{R}_j^{(e)\ k}\right|^2\right) + \tau_b\frac{1}{N_b}\sum_{i=1}^{N_b}\left|r_b^k(\boldsymbol{x}_b^i)\right|^2\right] + \sum_{e=1}^{N_{el}}\left(\frac{1}{K_c^{(e)}}\sum_{j=1}^{K_c^{(e)}}\left|\tilde{\mathcal{R}}_j^{(e)}\right|^2\right), \tag{42}$$

where $K_u^{(e)}$ and $K_c^{(e)}$ are the total number of test functions in element e.

Note that hp-VPINNs are based on sub-domain Petrov–Galerkin methods to allow for hp-refinement via domain decomposition as h-refinement and projection onto high order polynomials as p-refinement.[23] Although in the current hp-VPINNs formulation the domain is decomposed into several sub-domains, a single deep neural network is used to approximate the solution over the entire domain. The optimal choice of test functions is an important open question.

3.3 | VE-PINNs formulation

Here, we recall (17) and rewrite it in the following form;

$$\mathcal{L}(\boldsymbol{u}, c) = \int_\Omega \left[(1-c)^2\psi_e^+(\nabla^s\boldsymbol{u}) + \psi_e^-(\nabla^s\boldsymbol{u})\right]]\mathrm{d}V + \int_\Omega\left\{\frac{\mathcal{G}_c}{4\ell_0}\left[c^2 + 4\ell_0^2|\nabla c|^2\right] + (1-c)^2\mathcal{H}\right\}\mathrm{d}V. \tag{43}$$

The last term in (43) enforces the irreversibility to prevent cracks from healing if loads are removed. Next, we minimize (43) in the NN. To do this, we can define the loss function as follows:

$$\mathcal{L}_{\text{VE-PINNs}} = L(\hat{c}, \hat{\boldsymbol{u}}) + \sum_{k=1}^{d}\left[\tau_b\frac{1}{N_b}\sum_{i=1}^{N_b}\left|r_b^k(\boldsymbol{x}_b^i)\right|^2\right], \tag{44}$$

where

$$L(\hat{c}, \hat{\boldsymbol{u}}) = \int_\Omega \left[(1-\hat{c})^2\psi_e^+(\nabla^s\hat{\boldsymbol{u}}) + \psi_e^-(\nabla^s\hat{\boldsymbol{u}})\right]\mathrm{d}V + \int_\Omega\left\{\frac{\mathcal{G}_c}{4\ell_0}\left[\hat{c}^2 + 4\ell_0^2|\nabla\hat{c}|^2\right] + \mathcal{H}(1-\hat{c})^2\right\}\mathrm{d}V. \tag{45}$$

The loss function for VE-PINNs with strong BC can be written as:

$$\mathcal{L}_{\text{VE-PINNs}} = L(\hat{c}, \hat{\boldsymbol{u}}). \tag{46}$$

Note that an alternative to (43) is the hybrid formulation.[32] The idea in devising the hybrid model is to decrease the computational cost. However, the evolution of phase-field, c, must be driven by the tensile elastic energy ψ_e^+ alone to avoid cracking from occurring in the compressed regions. The hybrid version of (43) is given as

$$\mathcal{L}(\boldsymbol{u}, c) = \int_\Omega \left[(1-c)^2\psi_e(\nabla^s\boldsymbol{u})\right]\mathrm{d}V + \int_\Omega\left\{\frac{\mathcal{G}_c}{4\ell_0}\left[c^2 + 4\ell_0^2|\nabla c|^2\right] + (1-c)^2\mathcal{H}\right\}\mathrm{d}V. \tag{47}$$

4 | COMPARATIVE ANALYSIS AND DISCUSSION

In this section, we investigate the performance and relative accuracy of the different versions of PINNs for a one-dimensional phase-field problem, with the goal of better understanding the relative merits of each variant. For all simulations, we initialize the weights of the network randomly from a Gaussian distribution using Xavier initialization approach.[35] We consider the one-dimensional model problem[36] of a unit cross-sectional area with a modulus of elasticity $E = 1$ depicted in Figure 1. It consists of a bar with fixed ends which is loaded along its axis by a sinusoidal load. We consider a crack at the center of the bar and the discontinuous solution fields in the fully fractured case are given by

$$u_{\text{exact}} = \begin{cases} \frac{1}{\pi^2}\sin(\pi x) - \frac{1+x}{\pi}, & x < 0, \\ \frac{1}{\pi^2}\sin(\pi x) + \frac{1-x}{\pi}, & x \geq 0, \end{cases} \tag{48}$$

$$\sigma_{\text{exact}} = \frac{1}{\pi}\cos(\pi x) - \frac{1}{\pi}. \tag{49}$$

4.1 | PINNs versus collocation method

For the 1D problem of Figure 1 with a crack at the center of bar an initial history field is given by

$$\mathcal{H}_0 = \begin{cases} 1000.0, & d(\boldsymbol{x}, l) \leq \ell_0, \\ 0, & d(\boldsymbol{x}, l) > \ell_0. \end{cases} \tag{50}$$

We consider a fully connected neural network with 4 hidden layers and 50 neurons in each hidden layer. We use a hyperbolic tangent as the activation function while a linear activation function has been implemented for the last hidden layer. We have subdivided the domain, $[-1, 1]$, into three subdomains which are $[-1.0, -2\ell_0]$, $[-2\ell_0, 2\ell_0]$, and $[2\ell_0, 1]$. 112 collocation points have been generated in each subdomain by using the Gauss–Legendre rule. We have chosen the collocation points to make a fair comparison with previous works[26] as well as with the VPINNs and VE-PINNs solutions in subsequent subsections. The network architecture of the neural network, the number of collocation points and the length-scale of phase-field are shown in Table 1. Here, we used the weak BC approach to consider the boundary conditions in the cost function. Figure 2A,B depicts the comparison of the displacement and the stress obtained with PINNs, the exact solution and the isogeometric phase-field collocation method on a mesh of 256 Bézier elements (note that we have selected this as a basis for comparison simply because the isogeometric collocation approach is one of the most accurate numerical schemes currently available).[36] PINNs can accurately resolve the crack, yielding good results that do not exhibit any oscillations unlike isogeometric collocation. Thus, it seems that PINNs are able to provide accurate results while it has been reported previously that only VE-PINNs can do this.[26] Note that PINNs requires a large number of training iterations, epochs, to converge in comparison to VE-PINNs.[28]

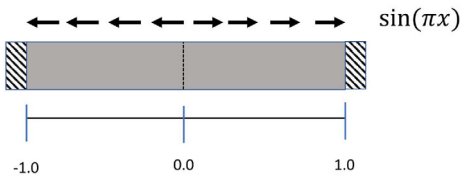

FIGURE 1 One-dimensional model of a bar with a center crack under a sinusoidal load

TABLE 1 Neural network setting for 1D PINNs versus collocation method

Method	Network architecture	No. collocation points	ℓ_0
PINNs	[1, 50, 50, 50, 50, 2]	336	0.015

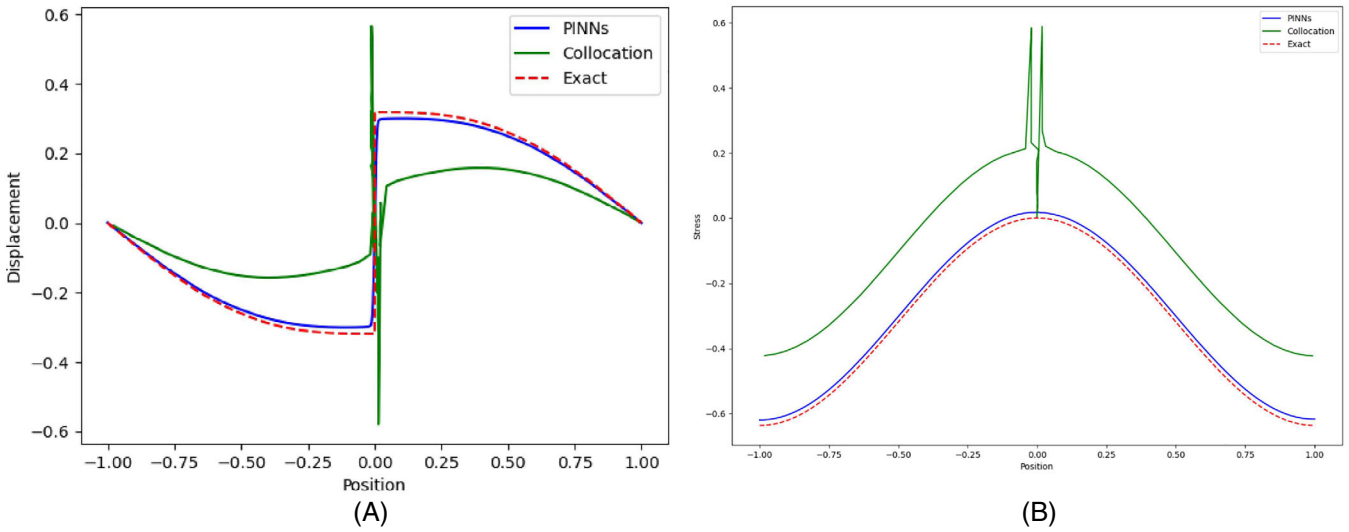

FIGURE 2 Comparison of displacements (A) and stresses (B) for solutions obtained with PINNs, isogeometric phase-field collocation method, and exact solution

4.2 | PINNs, VPINNs, and VE-PINNs

We now investigate the accuracy of PINNs, VPINNs, and VE-PINNs and recall the 1D problem discussed before. The same neural network architecture of the previous subsection is utilized. We use the collocation points of Section 4.1 as the integration points to compute the integral in VPINNs and VE-PINNs. For VPINNs, we have adopted $K_u^{(e)} = K_c^{(e)} = 60$ test functions and used three elements, with each element consisting of 112 quadrature points to compute the integrals. Furthermore, the displacement Dirichlet boundary conditions have been imposed weakly as well as strongly. Note that the penalty parameter has been assumed $\tau_b = 1$. Figures 3A and 4A show the solutions for the displacement of PINNs, VPINNs, and VE-PINNs for weakly and strongly imposed boundary conditions, respectively. Also, Figures 3B and 4B plot the stresses of PINNs, VPINNs, and VE-PINNs for weakly and strongly imposed BC, respectively. Figure 5A,B shows the convergence histories of PINNs, VPINNs, and VE-PINNs for weakly and strongly imposed BC, respectively. VEPINNs converge with using much less iterations compared to VPINNs and PINNs. In order to compare the accuracy of the different methods we assess the quality of the solutions by the L^2 norm and the error in the stress which can be considered as an extension of the error in H^1 semi-norm of the elasticity part to the phase-field.

$$L^2 = \left(\int_\Omega (\hat{u} - u_{\text{exact}})^2 dx \right)^{\frac{1}{2}}, \tag{51}$$

$$e = \left(\int_\Omega \left((1-\hat{c})^2 \frac{d\hat{u}}{dx} - \frac{du_{\text{exact}}}{dx} \right)^2 dx \right)^{\frac{1}{2}}. \tag{52}$$

The errors in the L^2 norm and the H^1 semi-norm for PINNs, VPINNs, and VE-PINNs by imposing the displacement boundary conditions weakly are shown in Table 2. VE-PINNs provide the most accurate solutions in comparison with the other versions of PINNs. Table 3 shows the errors when the Dirchlet boundary conditions are applied strongly. Although these errors are smaller for PINNs there is a concern about the unphysical oscillations in the stress field for the variational approaches. Table 4 shows the accuracy of the VE-PINNs with weakly imposed BC for the different length scales. The error in the stress is the smallest for $\ell_0 = 0.02$, whereas the most accurate solution in the L^2 norm (displacements) obtained for $\ell_0 = 0.005$.

Overall, VE-PINNs result in the smallest errors in terms of the displacement and the stresses compared with PINNs and VPINNs both for weakly and for strongly imposed boundary conditions. In contrast, PINNs show the largest errors in terms of the displacements and the stresses. Although VPINNs result in a slightly better performance than PINNs with soft boundary conditions, it results in lower accuracy with hard imposed boundary conditions. Considering the complexity of

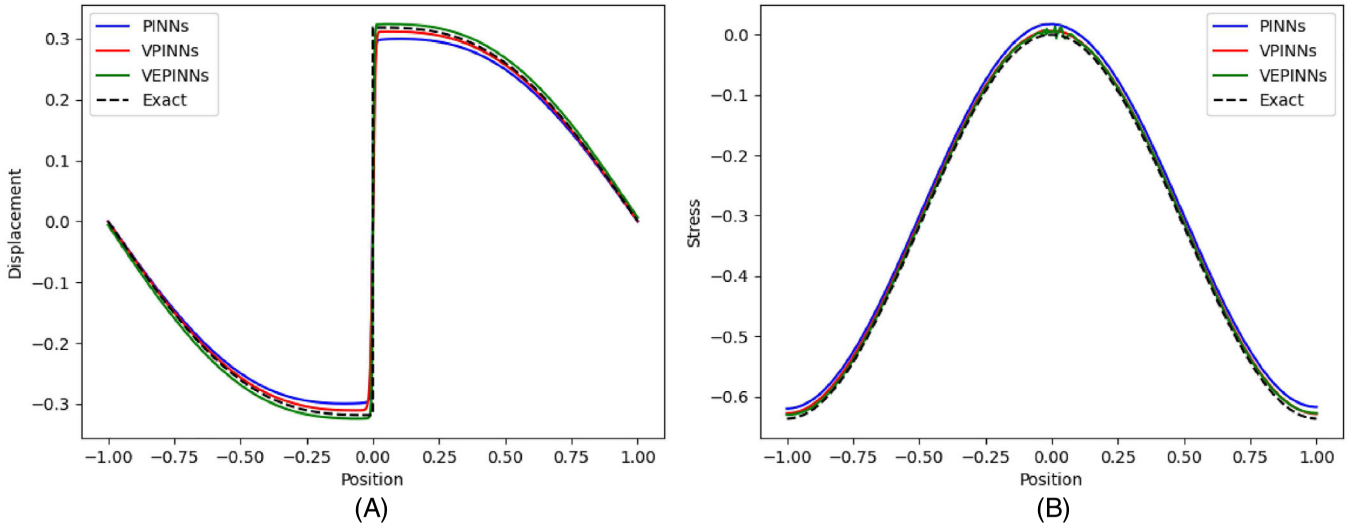

FIGURE 3 (A) Displacements and (B) stresses with a weakly imposed Dirchlet boundary condition

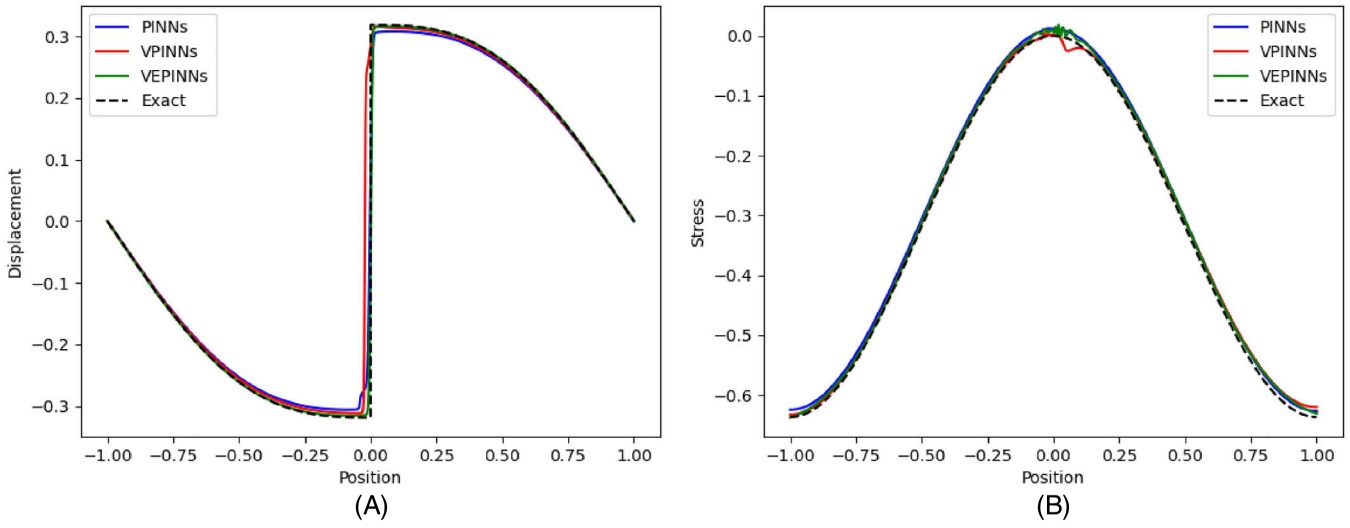

FIGURE 4 (A) Displacements and (B) stresses with a strongly imposed Dirchlet boundary condition

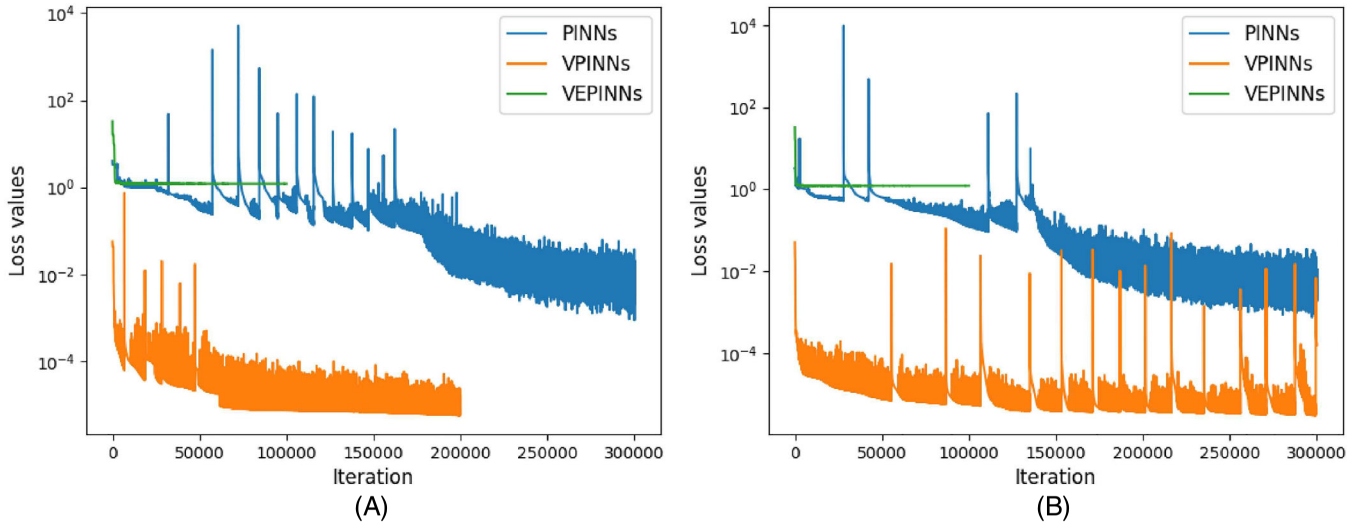

FIGURE 5 Convergence of the loss functions for (A) a weakly and (B) a strongly imposed Dirchlet boundary condition

TABLE 2 Errors in the L^2 norm and the H^1 semi-norm for PINNs, VPINNs, and VE-PINNs with weakly applied boundary conditions

Method	L^2 norm	H^1 semi-norm
PINNs	0.02963	0.02490
VPINNs	0.02806	0.01104
VE-PINNs	0.01961	0.00803

TABLE 3 Errors in the L^2 norm and the H^1 semi-norm for PINNs, VPINNs, and VE-PINNs with strongly applied boundary conditions

Method	L^2 norm	H^1 semi-norm
PINNs	0.03084	0.01612
VPINNs	0.08343	0.01159
VE-PINNs	0.02245	0.01198

TABLE 4 L^2 and semi-H^1 errors for VE-PINNs with different phase-field length scale, ℓ_0

ℓ_0	L^2	Semi-H^1
0.001	0.25145	0.46525
0.005	0.00810	0.01299
0.010	0.01274	0.00813
0.015	0.01961	0.00803
0.020	0.02542	0.00503

implementation and adjustable parameters, based upon these tests, we conclude that VE-PINNs is the best option among the various PINNs versions for phase-field problems.

4.3 | Data-driven discovery of phase-field length scale

Data-driven discovery of partial differential equations using PINNs has been successfully used for a variety of problems in the engineering and scientific domains, for example, the Navier–Stokes equations.[19] We present here a simple example that demonstrates data-driven discovery with the phase-field problem for crack propagation. Using the correct length scale, ℓ_0, is crucial to obtain accurate solutions, since ℓ_0 is a material parameter.[37,38] To identify the correct value of ℓ_0 we first solve a forward problem to find the displacements and phase-field state variables. Subsequently, we can solve an inverse problem to find ℓ_0. First, we recall Equations (44) and (45) and assume ℓ_0 to be a variable. Next, we rewrite the equation as follows and minimize the $\mathcal{L}_{\text{VE-PINNs}}$ to find ℓ_0:

$$\mathcal{L}_{\text{VE-PINNs-data-driven}} = L(\hat{c}, \hat{\boldsymbol{u}}) + \sum_{k=1}^{d} \left[\frac{1}{N_u} \sum_{i=1}^{N_u} \left| \hat{u}^k(\boldsymbol{x}_u^i) - \bar{u}_i^k \right|^2 \right] + \frac{1}{N_c} \sum_{i=1}^{N_c} \left| \hat{c}(\boldsymbol{x}_c^i) - \bar{c}_i \right|^2, \tag{53}$$

where $L(\hat{c}, \hat{\boldsymbol{u}})$ is given in (45) and $\{x_u^i\}, \{x_c^i\}$ denote the training data on $\boldsymbol{u}(\boldsymbol{x})$, the displacements, and $c(\boldsymbol{x})$, the phase-field, respectively. $\bar{\boldsymbol{u}}$ and $\bar{c}$ are the solutions obtained from the forward problem. Note that we are minimizing (53) like (46) with the same network settings. To illustrate the performance of this approach, we consider the 1D problem used in previous subsections. We have created a training dataset by using $N_u = N_c = 500$ integration points. To generate a high-resolution dataset for this problem we use an isogeometric finite element method with quadratic NURBS basis functions. In the forward problem the length scale, ℓ_0, has been assumed 0.015. After minimizing (53) the length-scale is estimated with less than 2% error, $\ell_0^{\text{pred}} = 0.01529$. Figure 6 plots the convergence history.

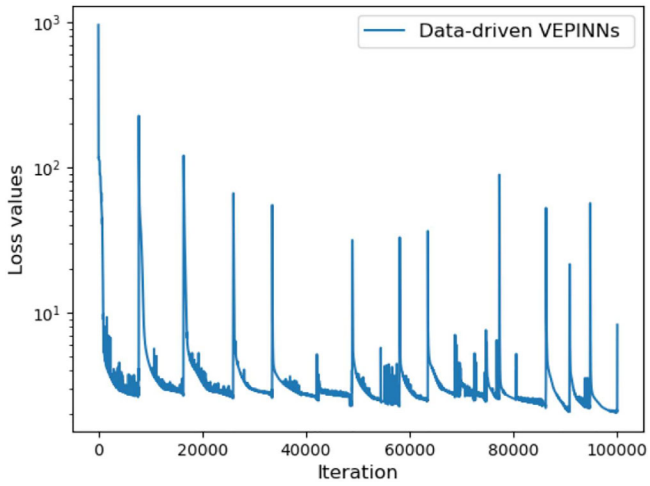

FIGURE 6 Convergence of the loss function data-driven VE-PINNs

4.4 | Self-adaptive PINNS

Now, we apply a self-adaptive approach method to train PINNs and VE-PINNs, which utilizes trainable weights as a soft multiplicative mask reminiscent of the attention mechanism applied in computer vision.[39,40] This method was first introduced in Reference 30 where the adaptation weights in the loss function were updated by backpropagation together with the network weights. In order to define the SA-PINNs loss function, we recall (40) and modify it as follows

$$\mathcal{L}_{\text{PINNs}} = \sum_{k=1}^{d}\left[\frac{1}{N_r}\sum_{i=1}^{N_r}\left[{}^{k}\lambda\; r^{k}(\boldsymbol{x}_r^i)\right]^2 + \frac{1}{N_b}\sum_{i=1}^{N_b}\left[{}^{k}\lambda_b\; r_b^k(\boldsymbol{x}_b^i)\right]^2\right] + \frac{1}{N_r}\sum_{i=1}^{N_r}\left[\tilde{\lambda}\;\tilde{r}(\boldsymbol{x}_r^i)\right]^2 + \frac{1}{N_{\tilde{b}}}\sum_{i=1}^{N_{\tilde{b}}}\left[\tilde{\lambda}_{\tilde{b}}\;\tilde{r}_{\tilde{b}}(\boldsymbol{x}_{\tilde{b}}^i)\right]^2, \tag{54}$$

where ${}^{k}\boldsymbol{\lambda} = ({}^{k}\lambda^1, \ldots, {}^{k}\lambda^{N_r})$, ${}^{k}\boldsymbol{\lambda}_b = ({}^{k}\lambda_b^1, \ldots, {}^{k}\lambda_b^{N_b})$, $\tilde{\boldsymbol{\lambda}} = (\tilde{\lambda}^1, \ldots, \tilde{\lambda}^{N_r})$, and $\tilde{\boldsymbol{\lambda}}_{\tilde{b}} = (\tilde{\lambda}_{\tilde{b}}^1, \ldots, \tilde{\lambda}_{\tilde{b}}^{N_b})$ are trainable self-adaption weights for momentum collocation points, momentum boundary, phase-field collocation points and phase-field boundary, respectively. These weights force the network to satisfy as much as possible the boundary or residual points. The main feature of SA-PINNs is that the loss $\mathcal{L}_{\text{PINNs}}$ is not only minimized with respect to the network weights, $\boldsymbol{W}$, but also maximized with respect to the self-adaptation weights, ${}^{k}\boldsymbol{\lambda}, {}^{k}\boldsymbol{\lambda}_b, \tilde{\boldsymbol{\lambda}}$, and $\tilde{\boldsymbol{\lambda}}_{\tilde{b}}$. This means the training seeks a saddle point

$$\min_{\boldsymbol{W}}\max_{{}^{k}\boldsymbol{\lambda},{}^{k}\boldsymbol{\lambda}_b,\tilde{\boldsymbol{\lambda}},\tilde{\boldsymbol{\lambda}}_{\tilde{b}}} \mathcal{L}(\boldsymbol{W}, {}^{k}\boldsymbol{\lambda}, {}^{k}\boldsymbol{\lambda}_b, \tilde{\boldsymbol{\lambda}}, \tilde{\boldsymbol{\lambda}}_{\tilde{b}}). \tag{55}$$

The self-adaptive VE-PINNs loss function can be defined as follows

$$\mathcal{L}_{\text{VE-PINNs}} = L(\hat{c}, \hat{\boldsymbol{u}}) + \sum_{k=1}^{d}\left[\frac{1}{N_b}\sum_{i=1}^{N_b}\left[{}^{k}\lambda_b\; r_b^k(\boldsymbol{x}_b^i)\right]^2\right], \tag{56}$$

where $L(\hat{c}, \hat{\boldsymbol{u}})$ is given by (45).

Table 5 shows errors for the L^2 norm and the H^1 semi-norm for PINNs and SA-PINNs where 336 and 240 collocation points, are used for PINNs and SA-PINNs, respectively. Self-adaptive VE-PINNs provides more accurate results compared to VE-PINNs while using less collocation points.

Figure 7A,B shows errors for the L^2-error and the H^1 semi-norm for soft, strong, and self-adaptive VE-PINNs. Note that for soft VE-PINNs the Dirchlet boundary condition has been imposed weakly. Self-adaptive VE-PINNs provides the same accuracy of soft and strong VE-PINNs while requiring less collocation points. The convergence histories of soft and strong VE-PINNs as well as self-adaption VE-PINNs are depicted in Figure 8.

TABLE 5 L^2 and semi-H^1 errors for PINNs, SA-PINNs, VE-PINNs, and SA-VE-PINNs

Method	L^2	Semi-H^1
PINNs	0.02963	0.02490
SA-PINNs	0.02951	0.02562
VE-PINNs	0.01961	0.00803
SA-VE-PINNs	0.01627	0.00760

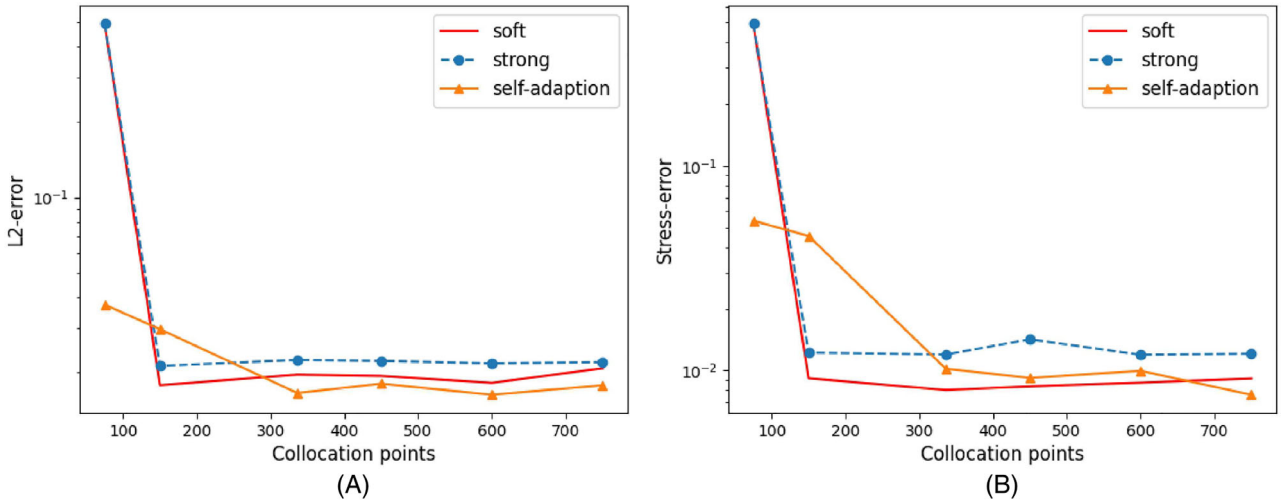

FIGURE 7 L^2 and semi-H^1 errors for soft, strong, and self-adaption VE-PINNs

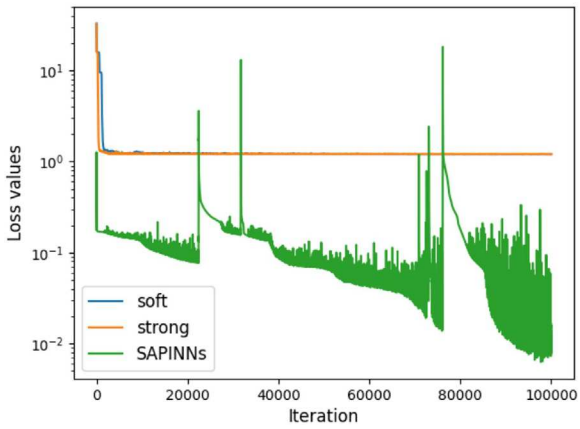

FIGURE 8 Convergence of the loss functions for soft, strong, and self-adaption VE-PINNs

5 | TWO-DIMENSIONAL CASE STUDIES

In this section, we consider two-dimensional examples. Here, we explain how the spectral decomposition is applied to compute ψ_e^+ and ψ_e^- in the neural network when we use VE-PINNs which we have selected based upon the conclusions drawn in the previous section. First, we calculate the displacement gradients, $\nabla \boldsymbol{u}$, and then we compute the eigenvalues of the strain, ($\alpha_1$and α_2) as well as the eigenvectors ($v_1$and v_2). Now, we can compute ψ_e^+ and ψ_e^- as follows

$$\psi_e^+ = \frac{\lambda}{8}(\alpha_t + |\alpha_t|)^2 + \frac{\mu}{4}\sum_{i=1}^{2}(\alpha_i + |\alpha_i|)^2 \quad \text{and} \quad \psi_e^- = \frac{\lambda}{8}(\alpha_t - |\alpha_t|)^2 + \frac{\mu}{4}\sum_{i=1}^{2}(\alpha_i - |\alpha_i|)^2, \tag{57}$$

where $\alpha_t = \sum_{i=1}^{2}\alpha_i$.

5.1 | Single-edge-notched tension test

We consider a square plate with a horizontal notch placed at mid-height from the left outer surface to the center of the specimen. The geometric properties and boundary conditions are depicted in Figure 9A. In order to capture the crack pattern properly, the collocation/integration points are more densely spaced in the area where the crack is expected to propagate, that is, in the center strip of the specimen, Figure 9B. A vertical upward displacement is imposed at the top edge. The material parameters are $E = 210.0$ GPa, $\nu = 0.3$, $G_c = 2.7 \times 10^{-3}$ kN/mm, and $\ell_0 = 0.01$ mm. Displacement control is used with increments $\delta u = 5 \times 10^{-4}$. In order to maintain the irreversibility of the crack propagation the strain-history field, $\mathcal{H}$, is update after each load-step in the same approach used in the standard phase-field models.[15,32] A finite element method with 29,896 linear triangle elements is utilized to validate the deep learning results. In terms of the deep learning setup, VE-PINNs with 4 hidden layers of 50 neurons each are used to carry out the simulations. First, we set *tanh* up as the activation function and consider 58,404 collocation points to perform the computation. Furthermore, the Adam optimizer[41] with a learning rate of 5×10^{-4} is employed. Regarding the boundary conditions, Dirichlet boundary conditions are imposed strongly. To do this, the solutions, u and v, are altered as follows:

$$u = (x^2 - 0.25)\hat{u}, \quad v = (y^2 - 0.25)\hat{v} + (y + 0.5)\Delta v, \tag{58}$$

where $\hat{u}$ and $\hat{v}$ are provided by the network and Δv is the displacement increments.

It is clear from Figure 10A for VE-PINNs, the crack propagation is much faster than in the finite element simulations. As is shown in Figure 10B for a displacement of 5×10^{-3}, crack growth is completed by the deep learning approach

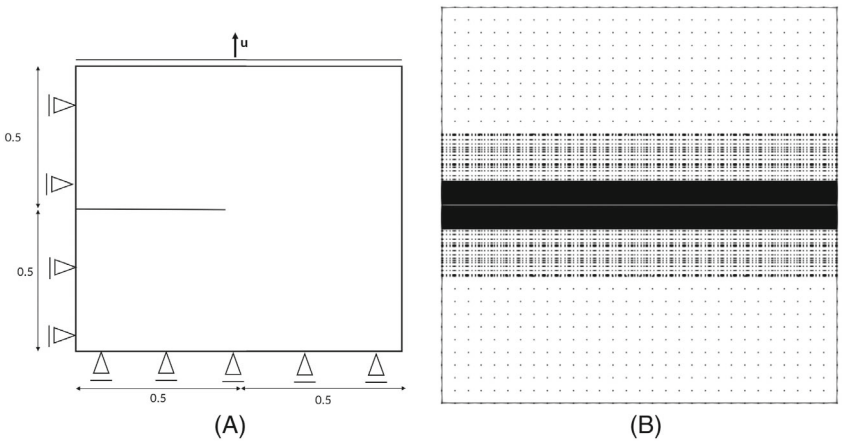

FIGURE 9 (A) Geometry and boundary conditions. (B) Collocation/integration points distribution

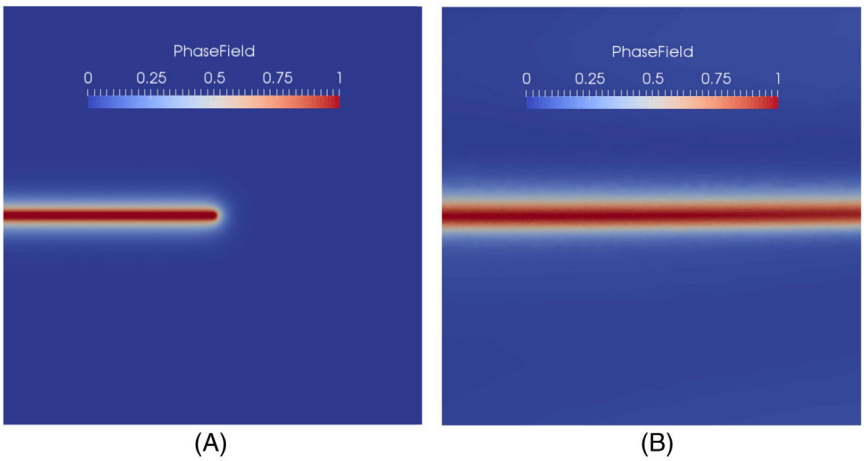

FIGURE 10 (A) Finite element ground truth. (B) Deep learning prediction with *tanh* activation function; for the prescribed displacement of 5×10^{-3} mm

whereas the finite element solution of phase-field is still at the early stage of the propagation, Figure 10A. This suggests that the VE-PINNs approach is not able to predict the correct crack propagation speed. This drawback of VE-PINNs methods (with *tanh* activation function) was not reported in the previous research for modeling phase-field with using VE-PINNs.[26-28] This problem can be resolved by using different activation functions, namely, $ReLU(x) = \max(0, x)$ and $softplus(x) = \ln(1 + e^x)$ by means of a suitable loadstep, 10^{-6} mm. As it is listed in Table 6, *ReLU* is the best prediction in terms of crack propagation speed and also, *sofplus* provides a slightly high crack speed and *tanh* has very high crack growth speed. Figure 11 shows the propagated crack for the different activation functions as well as the finite element solution for the prescribed displacement.

TABLE 6 Speed of crack propagation for the different activation functions for the single-edge-notched tension test with loadstep of 10^{-6} mm

Activation function	Crack propagation speed
ReLU	Normal
softplus	Slightly high
tanh	High

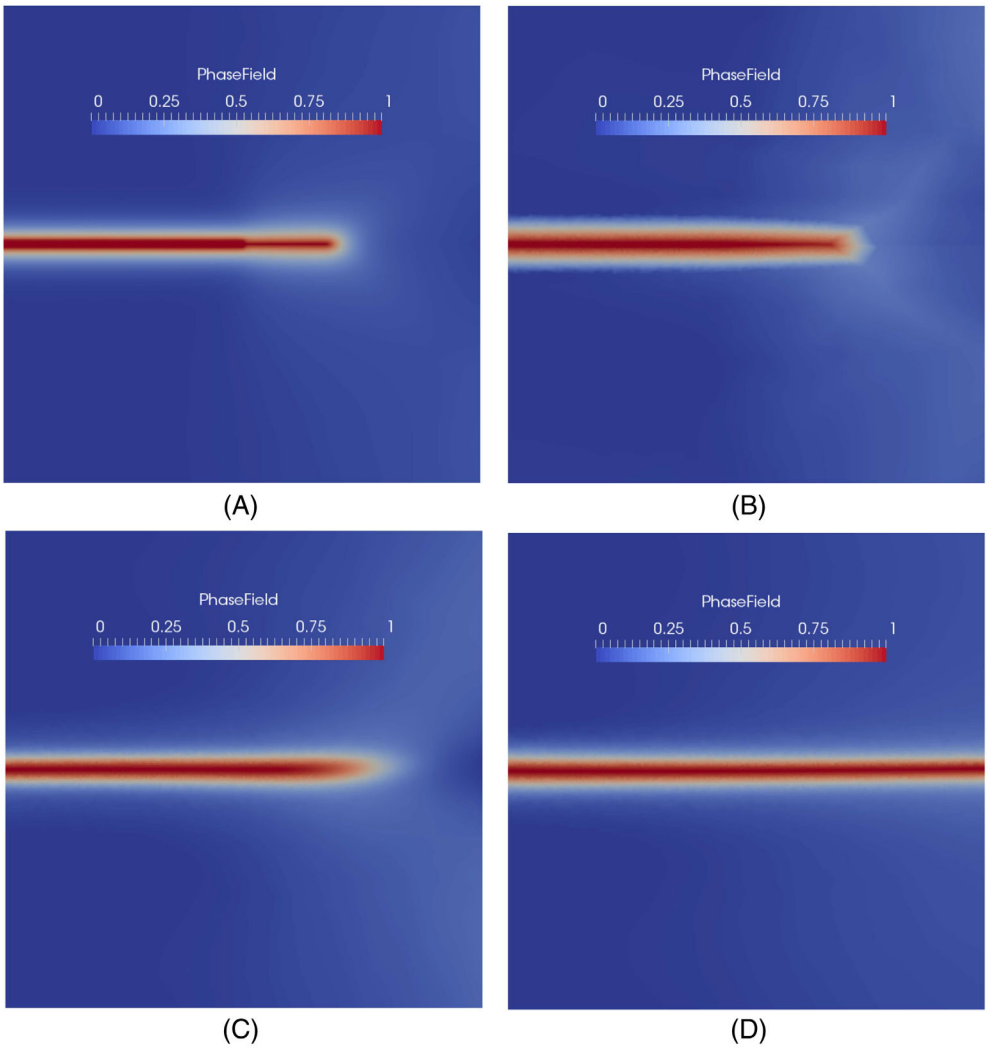

FIGURE 11 (A) Finite element ground truth. (B) Deep learning prediction with *ReLU* activation function. (C) Deep learning prediction with *softplus* activation function. (D) Deep learning prediction with *tanh* activation function; for the prescribed displacement of 6×10^{-3} mm

As linear finite element basis functions are represented by *ReLU*,[42] this results in the promising crack growth speed in comparison to *tanh*. Furthermore, *sofplus* can produce acceptable crack speed as it has the same tendency of *ReLU* as depicted in Figure 12. Note that *sofplus* has continuous derivatives as well.

5.2 | Single-edge-notched shear test

As a second benchmark, we consider the same specimen, but now subjected to a pure shear loading. The corresponding boundary conditions are illustrated in Figure 13A. A horizontal displacement increment, $\Delta u = 1 \times 10^{-6}$ mm, is utilized throughout the loading history. The material parameters are considered as $E = 210.0$ GPa, $\nu = 0.3$, $G_c = 2.7 \times 10^{-3}$ kN/mm, and $\ell_0 = 0.01$ mm. A finite element method with 60,572 linear triangle elements is used to compare with the deep learning predictions. For the deep learning part, the same setup as the previous subsection is implemented. We use *ReLU*, *softplus*, and *tanh* as the activation functions and consider 95,686 collocation points to conduct the computational experiment. Regarding boundary conditions, we utilize the same network settings used for the previous 2D example. The strongly imposed boundary conditions read:

$$v = (x^2 - 0.25)\hat{v}, \quad u = (y^2 - 0.25)\hat{u} + (y + 0.5)\Delta u. \tag{59}$$

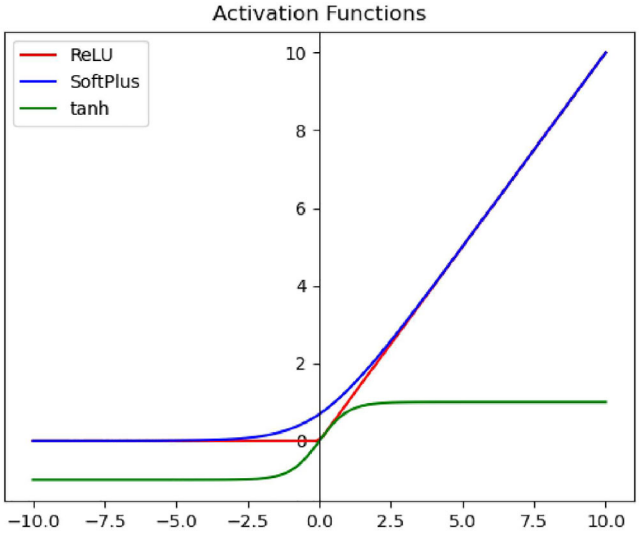

FIGURE 12 *ReLU*, *softplus*, and *tanh* activation functions

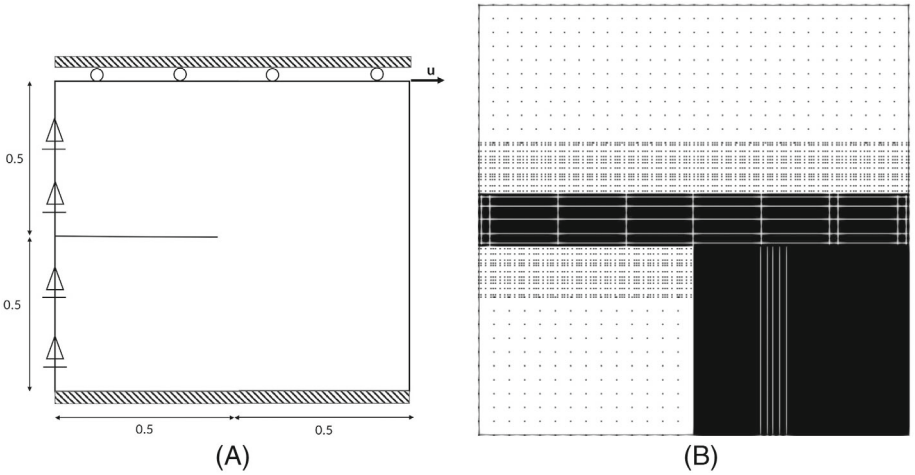

FIGURE 13 (A) Geometry and boundary conditions. (B) Collocation/integration points distribution

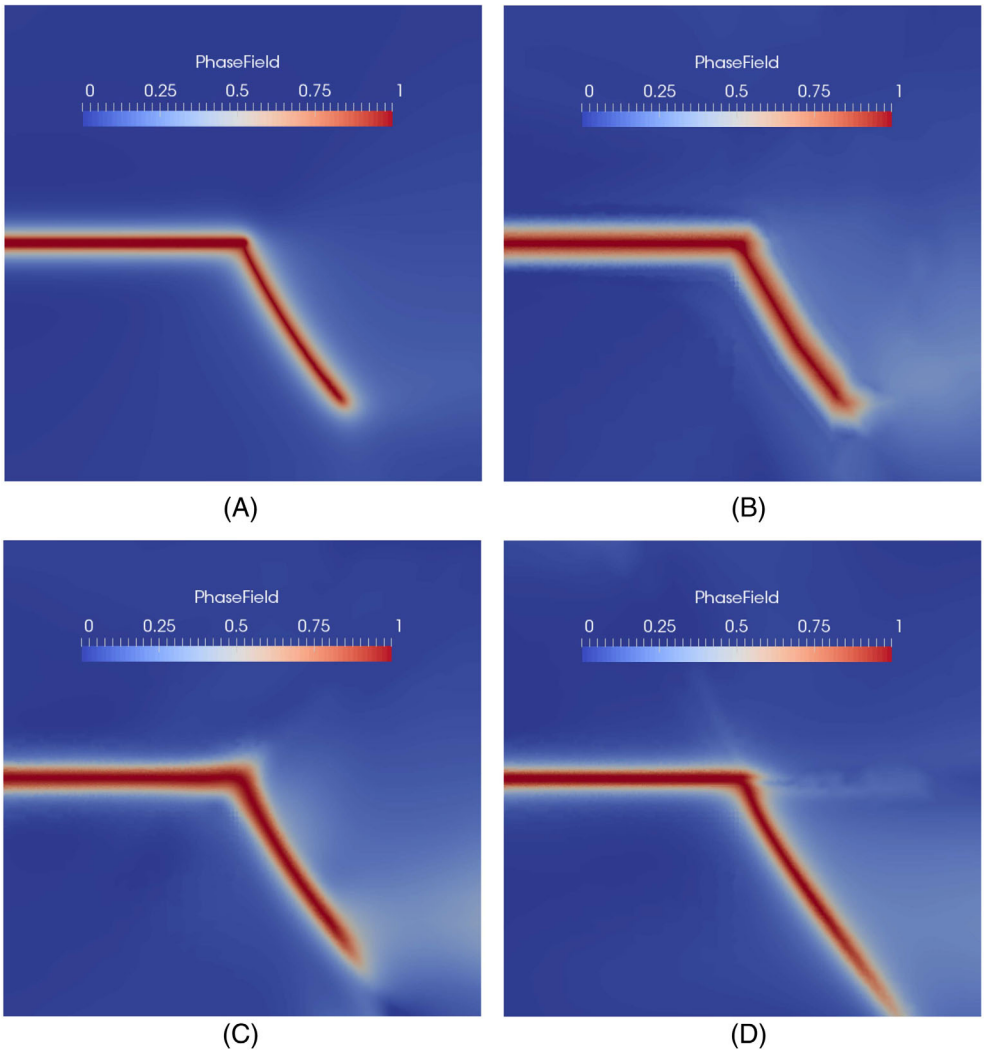

FIGURE 14 (A) Finite element ground truth. (B) Deep learning prediction with *ReLU* activation function. (C) Deep learning prediction with *softplus* activation function. (D) Deep learning prediction with *tanh* activation function; for the prescribed displacement of 1.2×10^{-3} mm

As illustrated in Figure 14D, the VE-PINNs can show a faster crack growth than finite element solution. Nevertheless, by selecting different activation functions such as *ReLU* and *softplus* the crack rate propagation can be kept under control, as shown in Figure 14B,C.

6 | CONCLUDING REMARKS

We have examined a variety of PINNs formulations, namely, standard PINNs, VPINNs, VE-PINNs, and SA-PINNs to solve a 1D phase-field problem. The standard PINNs or collocation PINNs require extra treatment in terms of boundary conditions, as they inherit the features of the collocation methods. This leads to the inclusion of the phase-field Neumann boundary condition in the cost function whereas this boundary condition for VPINNs and VE-PINNs can be treated as a natural boundary condition. Apart from this issue, we can obtain more accurate solutions by means of VPINNs and VE-PINNs in comparison with PINNs. Although VPINNs have almost the same features of finite element methods such as test functions, defining the suitable kind of test functions as well as the enough number of weighting functions is not so straightforward. Moreover, implementation of VPINNs is not as easy as PINNs or VE-PINNs. We found that VE-PINNs are quite simple to implement and can produce more accurate results in comparison to VPINNs and PINNs. VE-PINNs are capable of generating precise results, but are limited to problems which can be formulated in the energy form or in a

functional format. In addition, we have shown that SA-PINNs can produce the results with the same accuracy as the other versions of PINNs by means of a much smaller number of collocation points. Finally, for the 1D phase-field problem, we have investigated data-driven discovery to find the correct phase-field length scale.

Regarding two-dimensional numerical examples, we have assessed two benchmark problems, namely, single-edge-notched tension and shear tests. We have observed that VE-PINNs do not seem to yield a robust predictive approach. While the shape of the crack path is predicted properly, which is especially critical in the shear test, the speed of crack propagation is dependent on the chosen activation function and on the load step. A judicious choice of the activation function is mandatory in order to obtain a correct speed of the crack propagation, with the ReLU activation function being the only one among the activation functions applied which correctly reproduced the finite element results. Finally, we have found that the number of the hidden layers and the number of neurons in each layer does not affect the speed of crack propagation. Nevertheless, it is clear from the sensitivity of the two-dimensional results that further research will need to be undertaken by the community before methods from the PINNs family, including variational or self-adaptive approaches inter alia, can be used reliably for this class of problem in 2D or higher.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to our article as no new data were created or analyzed in this study.

ORCID

Yousef Ghaffari Motlagh https://orcid.org/0000-0002-2626-8510
Peter K. Jimack https://orcid.org/0000-0001-9463-7595
René de Borst https://orcid.org/0000-0002-3457-3574

REFERENCES

1. Rots JG. Smeared and discrete representations of localized fracture. *Int J Fract*. 1991;51:45-59.
2. Allix O, Ladevèze P. Interlaminar interface modelling for the prediction of delamination. *Compos Struct*. 1992;22:235-242.
3. Schellekens JCJ, de Borst R. Free edge delamination in carbon-epoxy laminates: a novel numerical/experimental approach. *Compos Struct*. 1994;28:357-373.
4. Xu XP, Needleman A. Numerical simulations of fast crack growth in brittle solids. *J Mech Phys Solids*. 1994;42:1397-1434.
5. Ingraffea AR, Saouma V. Numerical modelling of discrete crack propagation in reinforced and plain concrete. *Fracture Mechanics of Concrete*. Martinus Nijhoff Publishers; 1985:171-225.
6. Camacho GT, Ortiz M. Computational modelling of impact damage in brittle materials. *Int J Solids Struct*. 1996;33:2899-2938.
7. Secchi S, Simoni L, Schrefler BA. Mesh adaptation and transfer schemes for discrete fracture propagation in porous materials. *Int J Numer Anal Methods Geomech*. 2007;31:331-345.
8. Moës N, Dolbow J, Belytschko T. A finite element method for crack growth without remeshing. *Int J Numer Methods Eng*. 1999;46:131-150.
9. Remmers JJC, de Borst R, Needleman A. A cohesive segments method for the simulation of crack growth. *Comput Mech*. 2003;31:69-77.
10. de Borst R, Remmers JJC, Needleman A, Abellan MA. Discrete vs smeared crack models for concrete fracture: bridging the gap. *Int J Numer Anal Methods Geomech*. 2004;28:583-607.
11. Peerlings RHJ, de Borst R, Brekelmans WAM, de Vree JHP. Gradient enhanced damage for quasi-brittle materials. *Int J Numer Methods Eng*. 1996;39:3391-3403.
12. Francfort GA, Marigo JJ. Revisiting brittle fracture as an energy minimization problem. *J Mech Phys Solids*. 1998;46:1319-1342.
13. Bourdin B, Francfort GA, Marigo JJ. The variational approach to fracture. *J Elast*. 2008;91:5-148.
14. Miehe C, Hofacker M, Welschinger F. A phase field model for rate-independent crack propagation: robust algorithmic implementation based on operator splits. *Comput Methods Appl Mech Eng*. 2010;199:2765-2778.
15. Miehe C, Welschinger F, Hofacker M. Thermodynamically consistent phase-field models of fracture: variational principles and multi-field FE implementations. *Int J Numer Methods Eng*. 2010;83:1273-1311.
16. de Borst R, Verhoosel CV. Gradient damage *vs* phase-field approaches for fracture: similarities and differences. *Comput Methods Appl Mech Eng*. 2016;312:78-94.
17. Karniadakis GE, Kevrekidis IG, Lu L, Perdikaris P, Wang S, Yang L. Physics-informed machine learning. *Nature Rev Phys*. 2021;3(6):422-440.
18. Reichstein M, Camps-Valls G, Stevens B, et al. Deep learning and process understanding for data-driven Earth system science. *Nature*. 2019;566(7743):195-204.
19. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys*. 2019;378:686-707.
20. Krishnapriyan A, Gholami A, Zhe S, Kirby R, Mahoney MW. Characterizing possible failure modes in physics-informed neural networks. *Adv Neural Inf Proces Syst*. 2021;34.
21. Jagtap AD, Kharazmi E, Karniadakis GE. Conservative physics-informed neural networks on discrete domains for conservation laws: applications to forward and inverse problems. *Comput Methods Appl Mech Eng*. 2020;365:113028.

22. Pang G, Lu L, Karniadakis GE. fPINNs: fractional physics-informed neural networks. *SIAM J Sci Comput*. 2019;41(4):A2603-A2626.
23. Kharazmi E, Zhang Z, Karniadakis GE. hp-VPINNs: variational physics-informed neural networks with domain decomposition. *Comput Methods Appl Mech Eng*. 2021;374:113547.
24. Kharazmi E, Zhang Z, Karniadakis GE. Variational physics-informed neural networks for solving partial differential equations. arXiv preprint arXiv:1912.00873, 2019.
25. Jagtap AD, Karniadakis GE. Extended physics-informed neural networks (xpinns): a generalized space-time domain decomposition based deep learning framework for nonlinear partial differential equations. *Commun Comput Phys*. 2020;28(5):2002-2041.
26. Goswami S, Anitescu C, Chakraborty S, Rabczuk T. Transfer learning enhanced physics informed neural network for phase-field modeling of fracture. *Theor Appl Fract Mech*. 2020;106:102447.
27. Goswami S, Anitescu C, Rabczuk T. Adaptive fourth-order phase field analysis using deep energy minimization. *Theor Appl Fract Mech*. 2020;107:102527.
28. Samaniego E, Anitescu C, Goswami S, 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 Eng*. 2020;362:112790.
29. Goswami S, Yin M, Yu Y, Karniadakis G. A physics-informed variational DeepONet for predicting the crack path in brittle materials. arXiv preprint arXiv:2108.06905, 2021.
30. McClenny L, Braga-Neto U. Self-adaptive physics-informed neural networks using a soft attention mechanism. arXiv preprint arXiv:2009.04544, 2020.
31. Borden MJ, Verhoosel CV, Scott MA, Hughes TJ, Landis CM. A phase-field description of dynamic brittle fracture. *Comput Methods Appl Mech Eng*. 2012;217:77-95.
32. Ambati M, Gerasimov T, De Lorenzis L. A review on phase-field models of brittle fracture and a new fast hybrid formulation. *Comput Mech*. 2015;55(2):383-405.
33. Wang S, Teng Y, Perdikaris P. Understanding and mitigating gradient pathologies in physics-informed neural networks. arXiv preprint arXiv:2001.04536, 2020.
34. Sun L, Gao H, Pan S, Wang JX. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Comput Methods Appl Mech Eng*. 2020;361:112732.
35. Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks. Proceedings of the 13th International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings; 2010:249-256.
36. Schillinger D, Borden MJ, Stolarski HK. Isogeometric collocation for phase-field fracture models. *Comput Methods Appl Mech Eng*. 2015;284:583-610.
37. Amor H, Marigo JJ, Maurini C. Regularized formulation of the variational brittle fracture with unilateral contact: numerical experiments. *J Mech Phys Solids*. 2009;57(8):1209-1229.
38. Vignollet J, May S, de Borst R, Verhoosel CV. Phase-field models for brittle and cohesive fracture. *Meccanica*. 2014;49:2587-2601.
39. Wang F, Jiang M, Qian C, et al. Residual attention network for image classification. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017:3156-3164.
40. Pang Y, Xie J, Khan MH, Anwer RM, Khan FS, Shao L. Mask-guided attention network for occluded pedestrian detection. Proceedings of the IEEE/CVF International Conference on Computer Vision; 2019:4967-4975.
41. Gulli A, Pal S. *Deep Learning with Keras*. Packt Publishing Ltd; 2017.
42. He J, Li L, Xu J, Zheng C. ReLU deep neural networks and linear finite elements. arXiv preprint arXiv:1807.03973, 2018.

How to cite this article: Ghaffari Motlagh Y, Jimack PK, de Borst R. Deep learning phase-field model for brittle fractures. *Int J Numer Methods Eng*. 2022;1-19. doi: 10.1002/nme.7135