

Integrated Finite Element Neural Network (I-FENN) for non-local continuum damage mechanics

Panos Pantidis*, Mostafa E. Mobasher

Civil and Urban Engineering Department, New York University Abu Dhabi, Abu Dhabi, P.O. Box 129188, UAE

Abstract

We present a new Integrated Finite Element Neural Network framework (I-FENN), with the objective to accelerate the numerical solution of nonlinear computational mechanics problems. We leverage the swift predictive capability of neural networks (NNs) and we embed them inside the finite element stiffness function, to compute element-level state variables and their derivatives within a nonlinear, iterative numerical solution. This process is conducted jointly with conventional finite element methods that involve shape functions: the NN receives input data that resembles the material point deformation and its output is used to construct element-level field variables such as the element Jacobian matrix and residual vector. Here we introduce I-FENN to the continuum damage analysis of quasi-brittle materials, and we establish a new non-local gradient-based damage framework which operates at the cost of a local damage approach. First, we develop a physics informed neural network (PINN) to resemble the non-local gradient model and then we train the neural network offline. The network learns to predict the non-local equivalent strain at each material point, as well as its derivative with respect to the local strain. Then, the PINN is integrated in the element stiffness definition and conducts the local to non-local strain transformation, whereas the two PINN outputs are used to construct the element Jacobian matrix and residual vector. This process is carried out within the nonlinear solver, until numerical convergence is achieved. The resulting method bears the computational cost of the conventional local damage approach, but ensures mesh-independent results and a diffused non-local strain and damage profile. As a result, the proposed method tackles the vital drawbacks of both the local and non-local gradient method, respectively being the mesh-dependence and additional computational cost. We showcase through a series of numerical examples the computational efficiency and generalization capability of I-FENN in the context of non-local continuum damage, and we discuss the future outlook. The PINN training code and associated training data files have been made publicly available online, to enable reproducibility of our results.

Keywords: I-FENN, finite element method, neural networks, continuum damage mechanics, non-local gradient

*Corresponding author. *E-mail address:* pp2624@nyu.edu (Panos Pantidis)

**Corresponding author. *E-mail address:* mostafa.mobasher@nyu.edu (Mostafa Mobasher)

1. Introduction

1.1. Literature review: machine-learning in computational mechanics

Maturing over the course of several decades, the Finite Element Method (FEM) has played a fundamental role in the understanding and prediction of complex mechanical phenomena and processes, constituting arguably one of the most robust, reliable and versatile numerical approaches which are currently available. The FEM's capability to model systems of arbitrary geometry and loading/boundary conditions, as well as the capability of handling non-linear problems, have established the role of FEM in several fields of computational mechanics [1–3]. This has led to the development and use of FEM to the solution of many problems of immense practical importance such as brittle and ductile fracture [4–6], time-dependent response of materials [7, 8], non-linear multi-physics problems [9, 10], multi-scale [11] and multiple length-scale [12] problems, as well as other applications [13, 14]. Despite its proven strengths, the computational cost of non-linear FEM models presents a challenge as the level of detail in a model increases, since it requires the iterative solution of significantly larger non-linear systems of equations. Therefore, continuous research efforts aim at reducing the cost of non-linear FEM models [15–20].

Machine-learning (ML) strategies have recently emerged as promising candidates to overcome these shortcomings, by improving the accuracy [21] and lowering the computational expense of computational mechanics simulations [22]. One potential pathway is the development of data-driven surrogate models, which can be implemented in multi-scale methods in order to bypass the computationally exhaustive representative volume element (RVE) simulations. These pre-trained constitutive models represent the homogenized stress-strain response at the micro-scale, thus allowing for more numerically efficient multi-scale analyses [11, 23]. A few representative examples of this approach include micromechanical modeling of polycrystalline materials [24, 25], path-dependent plasticity in aluminum-rubber composite materials [26] and history-dependent hyperelasticity models [27]. Alternatively, purely data-driven approaches attempt to avoid simulation biases and offer an alternative approach by calculating the material point equilibrium states based purely on experimentally available data. The work of Kirchdoerfer and Ortiz [28] introduced the principles of this framework, and extensions of the original elasticity-based work include applications in computational plasticity [29] and fracture [30]. However, data-driven frameworks have two major limitations. The first is the lack of physical bounds to the solution, which can lead to difficulties of the physical interpretation of their results [31]. The second is that the data-driven nature limits the extensibility of these approaches towards a more generic solution method [32–34]. Consequently data-driven methods still suffer fundamentally from their dependency on the training datasets [35], whether these originate from expensive nonlinear RVE-level simulations or from the available experimental data points.

To overcome these issues, several researchers have implemented more physics-consistent efforts, such as the incorporation of the thermodynamics relationships in the learning process [36]. Several variational methods were also introduced to enhance data-driven approaches by relating energy-based formulations to the problem setup [33, 37–39]. Additionally, Samaniego et al [40] developed the Deep Energy Method (DEM) which utilized the neural networks relationships as the discretization functions for the potential energy minimization

statement, and more recently DEM was extended to modeling finite strain hyperelasticity [41]. However, the non-convex nature of the optimization problems still poses significant challenges [40] and the presence of multiple local optima often hinders the efficiency of these methods, necessitating further work along this frontier [37, 38].

Other approaches have also attempted to enrich the finite element method using neural networks in a conceptually different way than the aforementioned. The Finite Element Network Analysis framework [42, 43] mimics the conventional finite element assembly process by utilizing pre-trained neural networks as surrogate building-blocks to formulate the physical system. Despite the generalization potential of this method to more complicated cases, this approach has been thus far implemented only to linear elastic 1D bars and 2D thin plates. More recently, Mitusch et al [44] demonstrated the ability of their hybrid FEM-NN network to recover unknown PDE terms by augmenting the differential equations and subsequently discretize them in space using FEM.

A conceptually different approach to minimize the cost of computational mechanics models is centered around the capability of machine-learning methods to predict the solution of the governing partial differential equations (PDEs), which were otherwise numerically solved using FEM and other numerical approaches. Along this pathway, physics-informed neural networks (PINNs) have gained significant attention over the last years. PINNs are tailored to the discovery of PDEs or the prediction of their solution. The idea behind them is to convert the objective of the network learning process from the minimization of the error between data points into a minimization process of the PDE residual error. This goal is achieved during the neural network training stage, by equating the expression of the network cost function to the PDE residual while also accounting for the imposed boundary conditions. The origins of PINNs can be traced back to almost three decades ago [45, 46], but the major developments were proposed more recently in the seminal work of Raissi et al [47, 48]. Since then, interest in this field has emerged and PINNs have found applications in several fields, such as quantum physics [49], flow problems [50], uncertainty models [51], optics and electromagnetics [52], and more. However, they have found only limited applications in the field of computational solid mechanics, particularly with respect to nonlinear problems [31, 53, 54].

1.2. Overview of proposed methodology and implementation

In this paper, we develop a methodology with the overarching goal of reducing the computational expense of nonlinear mechanics problems by leveraging the swift predictive capability of pre-trained physics-informed neural networks. The pivotal point from other methods in the literature is that we utilize the pre-trained Neural Network (NN) to compute both state variables and their derivatives on an element-level basis, jointly with conventional finite element procedures that involve shape functions and their derivatives. These two methods act in a synergistic fashion: the NN receives input data that resembles the material point deformation, and then its output data is used to construct element-level variables such as the element Jacobian matrix and the element residual vector. One of the key characteristics of the proposed framework is that the NN is directly embedded inside the element stiffness function, and therefore its utilization within a nonlinear, iterative procedure becomes straightforward once the network has been trained. More importantly this approach

preserves the framework of the solution of non-linear problems within FEM, and assures that the state variables and their derivatives which are derived as the NN output are re-computed in each iteration. The continuous update of the NN input and output within the iterative procedure is the key feature which allows for the convergence of the nonlinear finite element solution. A descriptive flowchart of our approach is shown in Figure 1.

I-FENN: Integrated Finite Element Neural Network

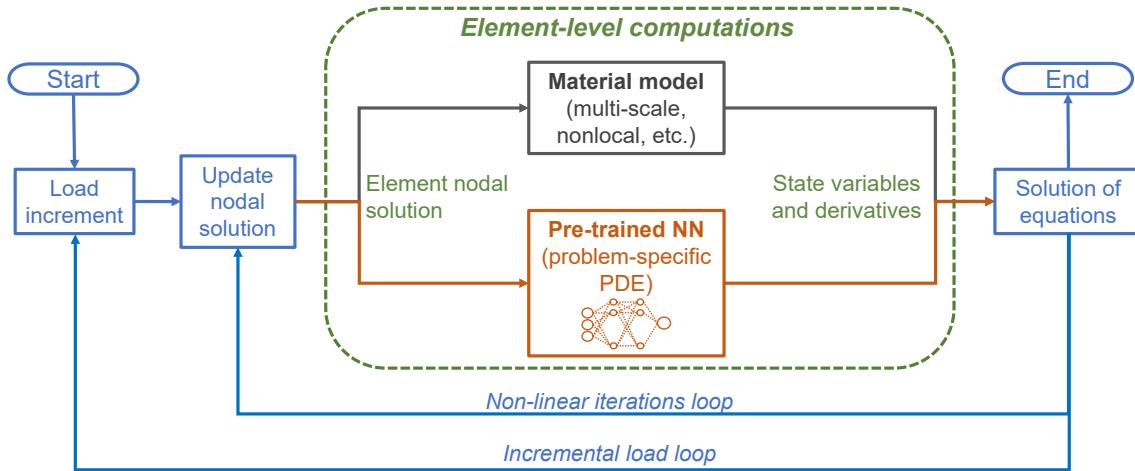


Figure 1: Flowchart of the proposed Integrated Finite Element Neural Network (I-FENN) methodology. A pre-trained physics-informed neural network is embedded inside the finite element stiffness function, and it is used to compute element-level state variables and their derivatives jointly with finite element shape functions. The entire scheme is encapsulated within a nonlinear, iterative numerical solver.

In this work we demonstrate the overall strategy of the proposed framework and we develop a specific implementation which targets the continuum damage mechanics (CDM) analysis of quasi-brittle materials. Specifically, we develop a CDM framework which has a non-local nature and utilizes PINNs as the key computational tool to predict the non-local strain field based on the gradient non-local PDE [55]. Generally, the main motivation behind using a non-local damage is to regularize the mesh dependence and the lack of solution uniqueness associated with local damage models [56–58]. However, the use of non-local damage models is often associated with additional computational cost and implementation complexities [55, 59]. The specific objective of our framework is to decrease the computational expense by developing a non-local damage model that resembles the gradient non-local relationship, while reducing the size of the system of equations. This is achieved by predicting the solution of the non-local gradient equation using a trained network, and hence eliminating the need for its numerical solution which includes solving for additional degrees of freedom and the need to implement a non-symmetric Jacobian matrix [55]. In other words, we eliminate the need for doing any geometric search as in the non-local integral approach [56], and we eliminate the need for the expanded system of equations exhibited by the non-local gradient model [55]. Thus, the output framework presents a non-local damage model that works at the computational cost of a local damage model.

The road-map of the implementation is briefly presented below, and a more extensive discussion is pro-

vided in the later sections. First, a PINN is developed and trained to predict the local to non-local strain transformation based on the gradient non-local strain equation [55]. Adopting a mathematical approach which is similar in spirit to the conventional local damage framework, we perform the following steps at the element-level basis: a) using the nodal displacements, we compute the local equivalent strain at each integration point, b) using the pre-trained PINN, we transform the latter to a non-local equivalent strain and extract their differential relationship, c) using the constitutive damage law, we compute a non-local damage variable based on the non-local strain PINN output, and d) we utilize the non-local damage from step (c), and the partial derivative of non-local to local equivalent strain from step (b) to construct the element Jacobian matrix and residual vector which are used for the non-linear FEM solution. Ultimately, non-locality is embedded in the FEM solution via the pre-trained PINN, whereas the size of the equations system matrix is in the order of the local damage method. As a result, we tackle the drawbacks of both the local and non-local gradient method, respectively being the mesh-dependence and computational cost. The PINN training code and associated training data files which are used in this study have been made publicly available online and can be found at: <https://drive.google.com/drive/folders/1mM35pXk7gzyx27NbIF3flqQxehr1tebh?usp=sharing>.

This study presents several novel contributions. From the integration of NNs in non-linear finite element point of view, the paper a) establishes the theoretical principles of the I-FENN framework, b) presents the details of a problem-specific implementation, which is targeted in the continuum damage analysis of quasi-brittle materials, and c) showcases through a series of numerical examples the feasibility and potential of the proposed methodology. In addition, from a damage modeling point of view, this paper presents a new non-local damage model in which the non-local damage variable is computed from the output of a neural network without the need for spatial averaging or gradient non-local PDE solution.

The paper is structured as follows: Section 2 presents a theoretical overview and a brief mathematical description of the local and non-local gradient theory. Section 3 details the principles and background of the proposed framework. Section 4 discusses the constitutive models which are adopted in this study, while Section 5 presents the numerical examples and elaborates on the observations and limitations of this work. Finally, Section 6 presents the overall conclusions of this paper and discusses the future outlook.

2. Theoretical Overview: local and non-local damage methods

2.1. Continuum damage mechanics methods

Material damage is the physical process during which degradation or complete loss of the mechanical properties occurs [60, 61]. In the case of quasi-brittle materials, such as ceramics and cementitious ones, damage evolves through crack initiation and subsequent propagation. From a representation standpoint, one can use either *discrete* and *smeared* numerical methods to describe the crack formulation. The first family of methods is based on the concepts of Linear Elastic Fracture Mechanics [62], where the crack is explicitly represented as a material discontinuity. The second approach adopts the fundamentals of Continuum Damage Mechanics (CDM) [60, 61], where the failure process is described through the use of strain-softening constitutive

relationships. In this case, damage is mathematically treated as a gradual stiffness loss which is calculated at each material point.

The definition of the damage variable at a local material point leads to loss of ellipticity of the PDE, which yields an ill-posed mathematical problem and consequently causes lack of uniqueness of the numerical solution [63]. From a numerical perspective, during the material degradation phase, this leads to the phenomenon of strain localization over an ever-decreasing area upon refinement of the mesh resolution. As a result, numerical methods suffer from mesh dependency, not only with regards to the element size but also to the mesh orientation. To overcome this deficiency, non-local constitutive models have been developed, aiming to model damage as a diffused property over a material region. In CDM, the most common frameworks are the a) *non-local integral* [56] and b) *non-local gradient* method [55]. Several notable attempts have been made towards eliminating the need for mathematically imposing the non-local field by leveraging Thick Level Set and Lip-Field approaches [64–67]. Phase-field approaches have also been developed and used to describe diffused material damage [68], and several studies indicate their analogy to the non-local gradient methods [69].

In the non-local integral approach, the non-local damage variable at a material point is computed as a weighted average of the local damage values at points within a characteristic length scale. This integral-type definition results to mesh-independence, but it is associated with several numerical challenges: a) the need to perform an algorithmic search for all integration points that lie within the neighbourhood of each material point, b) performance issues at the domain boundary, where natural discontinuities such as external boundaries and holes exist, and c) the difficulty of the derivation of the Jacobian matrix [70, 71]. Alternatively, the non-local gradient approach replaces the integral definition of the macroscopic deformation with a gradient-based formulation. This method introduces another governing PDE in the problem setup, coupling the local and non-local deformation measures. The major advantage of this method is that it retains a non-local character in an implicit manner, without the integral requirements of search and averaging continuously throughout the finite element analysis. Nevertheless, the inclusion of an additional PDE increases the computational cost. This will become apparent in Section 3, where we explicitly report the size of the Jacobian matrix for the local and the non-local gradient method. In addition, the resulting system of equations is not symmetric, which further increases the cost of the solution of the system of equations [55].

2.2. Mathematical formulation of the local and non-local gradient damage theories

Consider the elastic domain Ω shown in Figure 2, where Γ denotes its boundary. Displacements $\bar{\mathbf{u}}$ are prescribed in the Dirichlet boundary part Γ_u and traction forces \bar{t}_j are applied in the Neumann boundary part Γ_t ($\Gamma = \Gamma_u \cup \Gamma_t$). The analysis is governed by the equilibrium equation (1) and appropriate boundary conditions (2) and (3):

$$\sigma_{ij,j} = 0 \quad \text{in } \Omega \quad (1)$$

$$u_i = \bar{u}_i \quad \text{in } \Gamma_u \quad (2)$$

$$\sigma_{ij} \cdot n_i = \bar{t}_j \quad \text{in } \Gamma_t \quad (3)$$

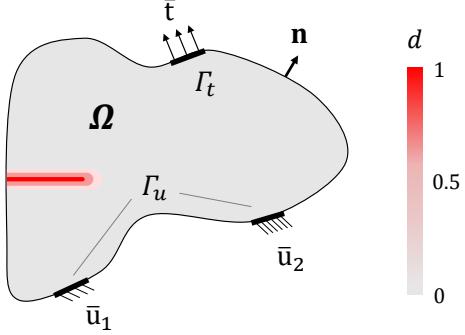


Figure 2: Schematic representation of a body with prescribed displacement and traction boundary conditions

where σ_{ij} is the Cauchy stress tensor and n_i denotes the outward unit normal vector on Γ . Assuming isotropic behavior, a scalar variable d is used to represent the stiffness loss magnitude. This state variable takes values between 0 and 1, with $d = 0$ indicating the intact material and $d = 1$ denoting the fully damaged state. The total stress σ_{ij} is expressed as follows [61]:

$$\sigma_{ij} = (1 - d) C_{ijkl} \varepsilon_{kl} \quad (4)$$

where C_{ijkl} is the fourth-order elasticity tensor and ε_{kl} is the strain tensor. The elasticity tensor is given by:

$$C_{ijkl} = \left[K \delta_{ij} \delta_{kl} + \mu \left[\delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk} - \frac{2}{3} \delta_{ij} \delta_{kl} \right] \right] \quad (5)$$

where K is the bulk modulus, μ is the shear modulus, and δ_{ij} is the Kronecker delta. Assuming small deformations, the strain-displacement relationship reads:

$$\varepsilon_{ij} = \frac{1}{2} [u_{i,j} + u_{j,i}] \quad (6)$$

A scalar invariant measure of deformation, termed *local equivalent strain*, can be defined as a function of the strain tensor: $\varepsilon_{eq} = \varepsilon_{eq}(\varepsilon_{ij})$. The local damage variable d is a function of the local equivalent strain, $d = d(\varepsilon_{eq})$, and their relationship is given by the appropriate damage evolution law.

In the non-local gradient damage model [55] the damage variable d evolves as a diffusive property over a finite material zone which is defined by a characteristic length scale measure l_c [56]. This is achieved by evolving the damage variable as a function of a non-local equivalent strain $d = d(\bar{\varepsilon}_{eq})$. The non-local equivalent strain $\bar{\varepsilon}_{eq}$ is calculated based on the following gradient-based PDE:

$$\bar{\varepsilon}_{eq} - g \nabla^2 \bar{\varepsilon}_{eq} = \varepsilon_{eq} \quad (7)$$

In the expression above, ∇^2 is the Laplacian operator and g is defined as $g = l_c^2/2$. The PDE definition in Equation (7) is usually complimented by the following boundary condition:

$$\nabla \bar{\varepsilon}_{eq} \cdot n_i = 0 \quad \text{in } \Gamma \quad (8)$$

Here we adopt equation (8) for the boundary conditions expression, a choice which is in consistence with previous works [55, 58]. This condition does not interfere with the physical interpretation of the non-local gradient model and allows for consistent damage diffusion in areas close to the boundaries.

3. Methodology

This section illustrates the implementation of the proposed Integrated Finite Element Neural Network I-FENN framework for the solution of the non-local damage model based on the gradient representation [55]. This process is carried out in two steps and it is graphically represented in Figure 3. First, we do an offline training of the neural network on the governing PDE with the objective to learn the local to non-local strain transformation. Subsection 3.1 discusses the formulation and underlying assumptions of the proposed PINN, and presents its design and training principles. The second step involves the integration of the PINN inside the element-level stiffness function, and for this purpose we devise the following strategy:

1. Using the appropriate nodal displacements $\hat{\mathbf{u}}$ and shape function derivatives \mathbf{B} , we calculate the local equivalent strain ε_{eq} at each integration point (IP).
2. Using the pre-trained PINN, we predict the non-local equivalent strain $\bar{\varepsilon}_{eq}^{NN}$ and its derivative with respect to the local equivalent strain $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$ at each IP.
3. Using the governing damage law, we compute the damage state variable d as a function of the non-local equivalent strain, $d = d(\bar{\varepsilon}_{eq}^{NN})$.
4. Using the conventional finite element procedures and utilizing d (step 3) and $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$ (step 2), we construct the element Jacobian matrix \mathbf{J}_{elem} and residual vector \mathbf{R}_{elem} .
5. Using the finite element connectivity matrix, we assemble the system-level Jacobian matrix \mathbf{J} and residual vector \mathbf{R} ; which is followed by solution of the equations.
6. Steps 1-5 are repeated until \mathbf{R} converges within the nonlinear iterative solver.

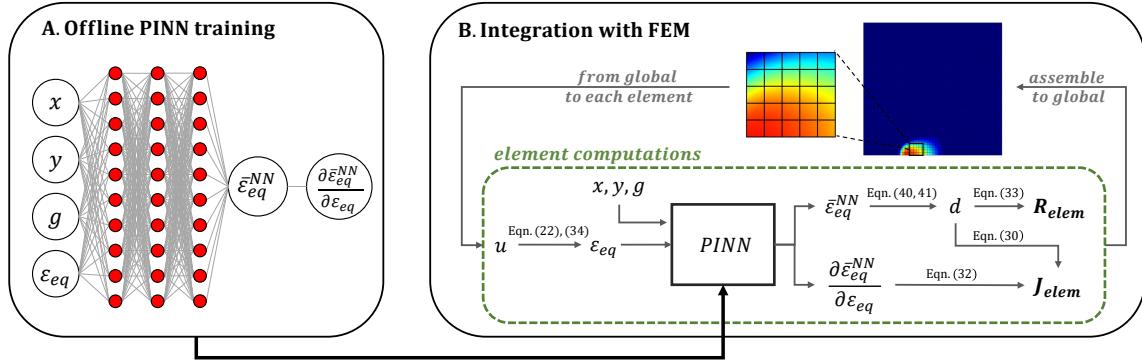


Figure 3: A schematic of the overall methodology of integrating the PINN with FEM towards and I-FENN aimed at the efficient solution of non-local damage. Box **A**. Schematic formulation of the adopted PINN. The input variables are the material point coordinates x and y , the local equivalent strain ε_{eq} and the internal length scale measure g . The output variables which will be used inside FEM are the non-local equivalent strain $\bar{\varepsilon}_{eq}$ and its partial derivative with respect to the local strain, $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$. Box **B**. Integration of the pre-trained PINN within the element-level computations. The PINN input is computed using the nodal displacements and the shape function derivative matrix. The PINN output is utilized in the calculation of the element Jacobian matrix \mathbf{J}_{elem} and residual force vector \mathbf{R}_{elem} .

Subsection 3.2 presents the theoretical derivation of the new non-local damage model, and provides more details on the strategy of trained PINN integration into the framework of a nonlinear finite element analysis as a complementary component on the element-level computations.

3.1. Physics-informed Neural Network formulation for the non-local gradient model

In the next paragraphs we detail the generic setup of Physics-Informed Neural Networks, while commenting on our choice of all the network hyperparameters. For a broader discussion on PINNs, the interested reader is directed to [47, 48, 72]. PINNs are a scientific machine-learning technique which is used to approximate the solution of partial differential equations (PDEs). Compared to the traditional multi-layer perceptrons, they are different in the sense that they a) compute partial derivatives of the output variable with respect to the input variables, and b) utilize these derivatives in the definition of the objective cost function. PINNs construct approximate numerical solutions to PDEs by minimizing the PDE residual error at selected interior points of the domain, termed *collocation points*, as well as enforcing the initial/boundary conditions along the spatio-temporal domain boundary. In the case of supervised learning several labeled value pairs may also be available, and therefore some of the predictions can be directly compared to the corresponding labeled values.

A generic mathematic formulation of the differential expressions solved by PINNs is given below [72]:

$$\mathbf{F}(\mathbf{h}(\mathbf{z})); \gamma = \mathbf{f}(\mathbf{z}) \quad \mathbf{z} \text{ on } \Omega \quad (9)$$

$$\mathbf{I}(\mathbf{h}(\mathbf{z})) = \mathbf{b}(\mathbf{z}) \quad \mathbf{z} \text{ on } \Gamma \quad (10)$$

where \mathbf{z} contains the spatio-temporal input variables, \mathbf{h} is the unknown quantity we aim to solve for, γ are physics-related parameters, \mathbf{F} is the differential expression, \mathbf{f} is a function relevant to the problem data, \mathbf{I} denotes initial or boundary conditions which constrain the solution space, \mathbf{b} is a boundary-data related function, and Ω is the domain with Γ indicating its boundary. PINNs approximate \mathbf{h} with a surrogate solution \mathbf{h}_θ , where θ indicates the network parameters. The training process begins by initialization of these parameters using any of the existing methods which exist in the literature with regards to this aspect [73, 74]. In our case, we utilize the Xavier initialization [73], which is a common practice in the PINN literature [31, 75, 76]. The network learns the parameter vector θ by minimizing a cost function which is defined as the weighted sum of the PDE residual error J_{PDE} , the initial/boundary conditions error J_{BCs} and the labeled data mismatch J_{Data} . Each term in this expression is multiplied with a corresponding weight factor, namely w_{PDE} , w_{BCs} and w_{Data} respectively, and therefore the cost function J reads:

$$J(\theta) = w_{PDE} J_{PDE}(\theta) + w_{BCs} J_{BCs}(\theta) + w_{Data} J_{Data}(\theta) \quad (11)$$

The optimum parameter vector θ^* is computed as:

$$\theta^* = \arg \min_{\theta} (J) \quad (12)$$

by the solution of the optimization problem involving the minimization of the cost function J . Minimization of the cost function J is carried out by performing sequential forward and backward propagation passes across the layers. This process continuously updates the network parameters, typically with gradient-based algorithms [77] such as the stochastic gradient descent or its more advanced successors (AdaGrad [78], Adam [79]), until a predefined convergence criterion is satisfied. The calculation of the partial derivatives between the input and output quantities in the PDE and BCs residuals is enabled via a powerful technique which is available on

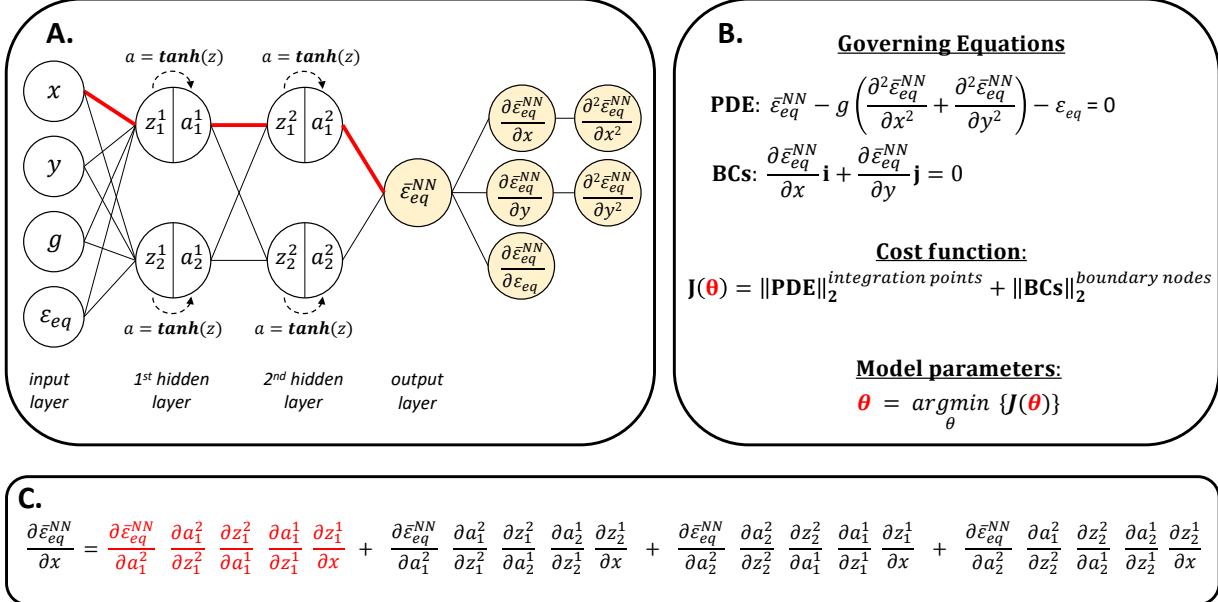


Figure 4: An overview of the developed PINN to represent the non-local gradient model. Box **A**. Detailed representation of our PINN shown with only two hidden layers and two neurons for clarity. Box **B**. The mathematical backbone of our framework: the partial differential equations which govern the non-local gradient formulation, the cost function definition and the expression for the PINN parameters θ . Box **C**. The analytical expression of the first-order partial derivative of $\bar{\varepsilon}_{eq}^{NN}$ with respect to the x coordinate. The red-highlighted term corresponds to the path shown with the red lines in (A).

many platforms platforms, namely the *automatic differentiation* [80, 81]. In this study, we make additional use of the automatic differentiation to compute the derivative of the network output with respect to an input variable, which allows us to compute the Jacobian matrix and residual for the FEM computations as discussed in the next section.

The PINN we develop in this study follows the layout and principles displayed in Figure 4. The first sub-plot of this figure (Box A) shows a schematic representation of the network architecture, with just two hidden layers with two neurons each for the sake of clarity. The proposed PINN has a fully-connected deep neural network structure. The input data in our case are the x and y coordinate of the material points, the internal length scale measure g , and the local equivalent strain ε_{eq} . The PINN output variables are the:

- non-local equivalent strain $\bar{\varepsilon}_{eq}^{NN}$
- second order derivatives of $\bar{\varepsilon}_{eq}^{NN}$ with respect to the x and y coordinates, $\frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial x^2}$ and $\frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial y^2}$, computed at the integration points
- first order derivatives of $\bar{\varepsilon}_{eq}^{NN}$ with respect to the x and y coordinates, $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial x}$ and $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial y}$, computed at the boundary nodes
- first order derivatives of $\bar{\varepsilon}_{eq}^{NN}$ with respect to the local equivalent strain, $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$, computed at the integration points

The first four partial derivatives $\left(\frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial x^2}, \frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial y^2}, \frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial x}, \frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial y} \right)$ are used in the cost function definition, and

more details are given in the next paragraphs. The partial derivative $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$ is incorporated in the nonlinear FEM solver and it is used for the Jacobian matrix calculation; the details of this implementation are given in Section 3.2.

Regarding the cost function, we first clarify that in the proposed model we assume a completely unsupervised learning context, meaning that there are no labeled pairs for the output predictions $\bar{\varepsilon}_{eq}^{NN}$. Therefore, $w_{data} = 0$ and the cost function degrades to the weighted sum of the governing differential equation and the boundary condition cost terms. Assuming a value of unity for each of the other weight terms ($w_{PDE} = 1, w_{BCs} = 1$), the proposed cost function reads:

$$J(\theta) = J_{PDE}(\theta) + J_{BCs}(\theta) \quad (13)$$

It is important to clarify the material point locations where each of the cost function terms is evaluated. In FEM, the unknown displacements are computed at the nodes, whereas using the shape functions derivatives the strains are calculated at the Integration Points (IPs). Since strains lie at the core of the problem-specific PDE, we treat the IPs as the PINN collocation points and we minimize the PDE residual in these locations. As a result, the input dataset used for the evaluation of the governing PDE is tied to the IPs, and so do the output variables $\bar{\varepsilon}_{eq}^{NN}$, $\frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial x^2}$, and $\frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial y^2}$. The output partial derivative $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$ is also calculated at the IPs, and it will be used to calculate the Jacobian matrix based on the I-FENN approach postulated in section 3.2. On the other hand, the boundary conditions are applied on the domain edges. Therefore, the input dataset used for the evaluation of the boundary condition differential expression is related to the boundary nodes, and so do the output variables $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial x}$ and $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial y}$. Ultimately, both J_{PDE} and J_{BCs} are vectors of dimension equal to the total number of IPs and boundary nodes respectively, and each entry contains the residual value at the relevant IP or boundary node. The error metric used for both vectors is the L2-norm, and its formula reads:

$$\|.\|_2 = \sqrt{\sum (.)^2} \quad (14)$$

The reasoning behind adopting the L2-norm, a metric which has been used in other studies as well [82], is the following. The appropriateness and accuracy of the selected error metric depends on the output range values [42]. For the specific type of quasi-brittle materials under consideration, strain values are in the order of 10^{-3} and below. Error metrics such as the mean squared error and its relatives tend to yield lower values for this range than the L2-norm, and therefore the latter is more critical and it is used as our cost function metric.

To summarize, the expressions for the governing differential equations which dictate the PINN training, as well as the cost function equation, are given below:

$$\mathbf{PDE} : \bar{\varepsilon}_{eq}^{NN} - g\left(\frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial x^2} + \frac{\partial^2 \bar{\varepsilon}_{eq}^{NN}}{\partial y^2}\right) - \varepsilon_{eq} = 0 \quad (15)$$

$$\mathbf{BCs} : \frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial x} \mathbf{i} + \frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial y} \mathbf{j} = 0 \quad (16)$$

$$J = \|\mathbf{PDE}\|_2^{integration\ points} + \|\mathbf{BCs}\|_2^{boundary\ nodes} \quad (17)$$

where \mathbf{i} and \mathbf{j} are the outward unit normal vectors in the x and y directions, respectively.

The presence of second-order derivatives in the PDE residual dictates the use of activation functions which are at least twice differentiable; otherwise, these derivatives will trivially degrade to zero. Therefore we select the *hyperbolic tangent (tanh())* function, which is always an admissible solution for PINNs [83], and we denote that piece-wise linear activation functions such as ReLU or leaky-ReLU are not appropriate choices for this specific problem. The search for the optimum hyperparameters, such as the number of neurons, layers, epochs and learning rate value, still lies as an open question of active research in many areas [84], and this is true for the subject problem as well. The network performance is a priori sensitive to these choices, and the later still relies heavily on trial-and-error methods, empirical observations and literature guidance. Each of the numerical examples which are demonstrated in Section 5 utilizes a different value for these hyperparameters, which are obtained through trial-and-error and they are reported in Table 2. As for the chosen optimizers, at the beginning of the training we use the Adam [79] algorithm for a predefined number of epochs, followed by the L-BFGS algorithm until a specified convergence threshold is satisfied. This approach has become a common training procedure which is adopted in several studies [75]. We also note that in its current formulation, the input data can potentially span different numerical scales. For example, the x and y coordinates as well as the length scale parameter g depend on the dimension units of the problem, whereas the local strain is dimensionless, and typically exhibits values that are orders of magnitude below unity. Though not obligatory, this contrast between scales in many engineering applications requires some sort of input feature scaling in order to increase the network ability to learn, which is a common practice in the field of data science [85–87]. In the numerical examples presented in the next section, we normalize the local equivalent strain by a scaling factor in the form of 10^c ($c = 1$ or 2), and subsequently unnormalize the output values by the same factor, as an initial attempt to increase the network learning speed.

It is useful to note here that the greater goal of this research effort is to design a single universal NN that can represent the local to non-local strain transformation regardless of problem-specific details, such as geometry, mesh discretization, boundary conditions, applied load, damage growth, etc. However, the development and training of such a network requires the abstraction of the spatial and temporal characteristics of non-local gradient damage evolution [88]. To the best of the author’s knowledge, current PINN applications are always limited by a certain set of geometries, boundary conditions, and other model-specific details. This is because PINNs are tied to the PDE they aim to approximate its solution, and the latter is sensitive to different configurations. Consequently, the state-of-the-art implementation of PINNs lacks the desired generalizability. The achievement of this network requires further research on the design of the network as well as the pre-processing of the data that gets passed to the network, which is beyond the immediate objectives of this study and requires some advancements in the design of neural networks by the active research community in this field [31, 75, 83]. Therefore, we bound the exploration space in most of the aforementioned parameters and mainly focus on the feasibility of integrating PINNs within a nonlinear finite element analysis, training a single PINN for a single combination of meshed geometry and applied load.

3.2. FEM implementation

This section presents the mathematical formulation of the proposed non-local gradient-based damage model and details its implementation within a nonlinear finite element solver. We remind the reader that the goal of the proposed implementation is twofold: a) maintain the size of equations system as in the local damage framework, in other words avoid solving numerically for the non-local equivalent strain, and b) embed a non-local nature in the damage field variable, and calculate the element Jacobian and residual vector consistently with the non-local damage definition. At the end of this section we include Table 1 which presents a comprehensive comparison between the local, non-local gradient and proposed methodology. This allows for a straightforward observation of the similarities and differences between the three methods, and confirms the advantages of the proposed model. For a complete derivation of the classical local [61] and non-local gradient [55] damage methods, the reader is referred to [Appendix A](#) and [Appendix B](#) respectively.

We begin by formulating the strong form of the governing PDE, which is the equilibrium condition. Substituting Equation (4) in (1) leads to:

$$[C_{ijkl}(d)\varepsilon_{kl}]_{,j} = 0 \quad \text{in } \Omega \quad (18)$$

where:

$$C_{ijkl}(d) = (1 - d)C_{ijkl} \quad (19)$$

Proceeding with the weak form of Equation (18):

$$\mathbf{R}(u) = \int_{\Omega} w^u \left[[C_{ijkl}(d)\varepsilon_{kl}]_{,j} \right] d\Omega \quad (20)$$

where w^u are the displacement field weight functions. Integration by parts of Equation (20) yields:

$$\mathbf{R}(u) = - \int_{\Gamma} [w^u t_i] d\Gamma + \int_{\Omega} w^u_{,j} [C_{ijkl}(d)\varepsilon_{kl}] d\Omega \quad (21)$$

where t_i corresponds to the external stress vector applied on the domain boundary. Denoting with \mathbf{N} and \mathbf{B} the displacement shape function matrix and its spatial derivatives respectively, the displacements and strains at the integration points are calculated as follows:

$$\mathbf{u} = \mathbf{N}\hat{\mathbf{u}}; \quad \varepsilon_{ij} = \mathbf{B}\hat{\mathbf{u}} \quad (22)$$

$$\mathbf{w}^u = \mathbf{N}\hat{\mathbf{w}}^u; \quad \mathbf{w}^u_{,j} = \mathbf{B}\hat{\mathbf{w}}^u \quad (23)$$

where the (\cdot) symbol is used to indicate nodal values. Substitution of Equations (22) and (23) into Equation (21) results to the following expression of the residual vector:

$$\mathbf{R}(u) = - \int_{\Gamma} [\mathbf{N}\hat{t}_i] d\Gamma + \int_{\Omega} \mathbf{B}^T (1 - d) C_{ijkl} \mathbf{B} \hat{\mathbf{u}} d\Omega \quad (24)$$

In a quasi-static analysis, equilibrium at each load increment is achieved once the residual vector is sufficiently minimized. The problem non-linearity necessitates an incremental and iterative procedure in order to

obtain the new equilibrium state, and appropriate nonlinear iterative numerical solvers such as the Newton-Raphson method or the arc-length method need to be implemented. In this study we adopt the Newton-Raphson approach, and a generic algorithm is provided in [Appendix C](#). In this scheme, denoting with i the iteration counter, the new displacement vector $\hat{\mathbf{u}}_i$ can be computed as follows:

$$\hat{\mathbf{u}}_i = \hat{\mathbf{u}}_{i-1} + \delta\hat{\mathbf{u}}_{i-1} \quad (25)$$

where $\hat{\mathbf{u}}_{i-1}$ is the displacement solution vector at the previous iteration, and $\delta\hat{\mathbf{u}}_{i-1}$ is the incremental change of the unknown degrees of freedom vector which is calculated as:

$$\mathbf{J}\delta\hat{\mathbf{u}} = -\mathbf{R}(u) \quad (26)$$

In Equation (26), the term \mathbf{J} is the Jacobian matrix and it is computed upon differentiation of Equation (24) with respect to the nodal displacements:

$$\mathbf{J} = \frac{\partial \mathbf{R}(u)}{\partial \hat{\mathbf{u}}} \quad (27)$$

Numerical convergence within each increment is achieved once the following equation is satisfied:

$$\frac{\|\delta\hat{\mathbf{u}}_{i=last}\|_2}{\|\delta\hat{\mathbf{u}}_{i=1}\|_2} \leq tol \quad (28)$$

where $\delta\hat{\mathbf{u}}_{i=last}$ and $\delta\hat{\mathbf{u}}_{i=1}$ are the L2-norms of the displacement solution vector incremental changes at the last and first iteration respectively, and tol is a numerical tolerance variable.

We now focus on the derivation of the \mathbf{J} and \mathbf{R} terms, which constitute the novelty of the proposed methodology. Plugging Equation (24) into Equation (27), the expression for the global Jacobian matrix \mathbf{J} reads:

$$\mathbf{J} = \mathbf{K}(u) + \frac{\partial \mathbf{K}(u)}{\partial \hat{\mathbf{u}}} \hat{\mathbf{u}} \quad (29)$$

where:

$$\mathbf{K}(u) = \int_{\Omega} \mathbf{B}^T (1-d) C_{ijkl} \mathbf{B} d\Omega \quad (30)$$

$$\frac{\partial \mathbf{K}(u)}{\partial \hat{\mathbf{u}}} = \int_{\Omega} \mathbf{B}^T C_{ijkl} \left(-\frac{\partial d}{\partial \hat{\mathbf{u}}} \right) \mathbf{B} d\Omega \quad (31)$$

Following standard FEM discretization procedures, the integral formulation of Equations (30) and (31) is replaced by a summation of these quantities computed on an element-level basis. Proceeding with element-level calculations, the damage variable d in Equation (30) is a function of the non-local strain PINN output, $d = d(\bar{\varepsilon}_{eq}^{NN})$, and it is calculated by substituting $\bar{\varepsilon}_{eq}^{NN}$ into the damage evolution law in [Section 4](#). Therefore, once the neural network is trained and it has learned the local to non-local strain transformation as explained in [Section 3.1](#), calculation of d is straightforward. The term $\frac{\partial d}{\partial \hat{\mathbf{u}}}$ in Equation (31) reflects the dependency of d to the nodal displacements $\hat{\mathbf{u}}$, which can be calculated as follows:

$$\frac{\partial d}{\partial \hat{u}_k} = \frac{\partial d}{\partial \bar{\varepsilon}_{eq}^{NN}} \frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}} \frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}} \frac{\partial \varepsilon_{ij}}{\partial \hat{u}_k} \quad (32)$$

[Equation \(32\)](#) represents the complete dependence of d to \hat{u}_k , accounting for all the intermediate transformation steps. Taking a closer look at each term in Equation (32):

- the term $\frac{\partial d}{\partial \bar{\varepsilon}_{eq}^{NN}}$ reflects the dependence of damage to the non-local strain PINN output, and it is computed based on the selected damage evolution relationship. The damage evolution laws used in this study are described in Section 4.
- the term $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$ represents the differential relationship between the PINN input ε_{eq} and the PINN output $\bar{\varepsilon}_{eq}^{NN}$. This partial derivative is one of the two output variables of the neural network, and it is computed based on automatic differentiation. Its incorporation inside Equation (32) is the key step that affirms the consistent Jacobian matrix definition and ensures the sound performance of the Newton-Raphson algorithm.
- the term $\frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}}$ depends on the definition of the local equivalent strain ε_{eq} upon the tensorial strain components ε_{ij} . The definition of the equivalent strain used in this study is provided in Section 4.
- the term $\frac{\partial \varepsilon_{ij}}{\partial \hat{u}_k}$ is the **B** matrix components, representing the displacement-strain relationship at the integration points.

This procedure computes the element-level Jacobian matrix \mathbf{J}_{elem} , and the global **J** is constructed upon the assembly of all \mathbf{J}_{elem} . As for the residual vector **R**, the implementation of the shape function derivative matrix **B** in Equation (20) yields:

$$\mathbf{R}(u) = \int_{\Omega} \mathbf{B}^T (1 - d) C_{ijkl} \varepsilon_{kl} d\Omega \quad (33)$$

The integral expression is evaluated at the element-level residuals \mathbf{R}_{elem} , which are then assembled to lead to the global **R** vector. Ultimately, substituting Equations (29) and (33) into Equation (26) allows for the calculation of $\delta\hat{\mathbf{u}}$ at each iteration. The nonlinear iterative process continues until Equation (28) is satisfied and numerical convergence has been achieved.

The workflow of the methodology is shown in Algorithm 1, which demonstrates how the trained neural network is integrated in the element-level computations and it is being utilized within a nonlinear, iterative procedure. The displacement and local strain fields are re-computed in each iteration, and the latter is used to update the non-local strain profile, jacobian matrix and residual vector. Therefore the PINN input and output are continuously updated within the Newton-Raphson procedure, which is the key element that allows for the residuals convergence.

Table 1 provides a comparison across the local, non-local gradient and proposed framework with regards to the main computed variables. We highlight that the local damage method and the proposed framework share the same number of degrees of freedom, whereas the non-local gradient [55] treats the non-local equivalent strain as an additional DOF. Specifically for a 2D finite element mesh with first order shape functions, the total number of DOFs for the local damage and the proposed method is $2n$ and for the gradient model it is $3n$. The residual expressions are provided in their weak form to allow for a more straightforward comparison across the three approaches, and we denote with $w^{\bar{\varepsilon}}$ the non-local strain weighting functions for the non-local gradient method. The mathematical derivation of the conventional local and non-local gradient methods can be found in Appendix A and Appendix B respectively.

Algorithm 1: Algorithm of the I-FENN framework in the case of non-local gradient damage

```

1 while Equation (28) is not satisfied do
2   for each integration point and each boundary node do
3     Compute the shape functions N and their derivatives B
4     Compute and extract coordinates,  $g$  and  $\varepsilon_{eq}$ 
5   end
6   Use the pre-trained PINN to predict  $\bar{\varepsilon}_{eq}^{NN}$  and  $\frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}}$  for all IPs
7   for each finite element do
8     for each integration point do
9       Compute the shape functions N and their derivatives B
10      Calculate  $\frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}}$ , based on the local equivalent strain definition
11      Calculate  $d$  and  $\frac{\partial d}{\partial \bar{\varepsilon}_{eq}^{NN}}$ , based on the governing damage law
12      Compute the IP contribution to the element jacobian matrix and residual vector
13   end
14 end
15 Assemble global Jacobian matrix J, solve for the displacement vector u, check convergence of
  residuals R
16 end

```

Table 1: Comparison between local, non-local gradient and proposed framework.

	Local	Non-local gradient	Proposed
#Nodes	n	n	n
#DOFs (2D)	$2n$	$3n$	$2n$
R (Residual)	$R = \int_{\Omega} w^u \left[[C_{ijkl}(d)\varepsilon_{kl}]_{,j} \right] d\Omega$	$R^u = \int_{\Omega} w^u \left[[C_{ijkl}(d)\varepsilon_{kl}]_{,j} \right] d\Omega$ $R^{\bar{\varepsilon}} = \int_{\Omega} w^{\bar{\varepsilon}} [\bar{\varepsilon}_{eq} - g\bar{\varepsilon}_{eq,ii} - \varepsilon_{eq}] d\Omega$	$R = \int_{\Omega} w^u \left[[C_{ijkl}(d)\varepsilon_{kl}]_{,j} \right] d\Omega$
J (Jacobian)	$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{u}} \end{bmatrix}$	$\begin{bmatrix} \frac{\partial \mathbf{R}^u}{\partial \hat{\mathbf{u}}} & \frac{\partial \mathbf{R}^u}{\partial \hat{\varepsilon}} \\ \frac{\partial \mathbf{R}^{\bar{\varepsilon}}}{\partial \hat{\mathbf{u}}} & \frac{\partial \mathbf{R}^{\bar{\varepsilon}}}{\partial \hat{\varepsilon}} \end{bmatrix}$	$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \hat{\mathbf{u}}} \end{bmatrix}$
Dependencies	$\frac{\partial d}{\partial \hat{u}_k} = \frac{\partial d}{\partial \varepsilon_{eq}} \frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}} \frac{\partial \varepsilon_{ij}}{\partial \hat{u}_k}$	$\frac{\partial d}{\partial \hat{\varepsilon}_{eq}}, \frac{\partial \varepsilon_{eq}}{\partial \hat{u}_k} = \frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}} \frac{\partial \varepsilon_{ij}}{\partial \hat{u}_k}$	$\frac{\partial d}{\partial \hat{u}_k} = \frac{\partial d}{\partial \bar{\varepsilon}_{eq}^{NN}} \frac{\partial \bar{\varepsilon}_{eq}^{NN}}{\partial \varepsilon_{eq}} \frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}} \frac{\partial \varepsilon_{ij}}{\partial \hat{u}_k}$

4. Constitutive modeling

This section presents the relationships to compute the local equivalent strain and the associated damage variable. As for the scalar invariant measure of the deformation, ε_{eq} , we adopt two definitions. The first one follows Lemaitre's approach [60]:

$$\varepsilon_{eq} = \sqrt{\langle \varepsilon_1 \rangle^2 + \langle \varepsilon_2 \rangle^2 + \langle \varepsilon_3 \rangle^2} \quad (34)$$

where $\varepsilon_i, i = 1, 2, 3$ are the principal strains and $\langle \cdot \rangle$ are the Macaulay brackets: $\langle \cdot \rangle = \frac{1}{2}(\cdot + |\cdot|)$. Following

this expression, ε_{eq} depends evidently only in the positive principal strains. This is a common assumption in quasi-brittle materials, which exhibit significantly lower tensile strength than the compressive strength [89]. The second definition of ε_{eq} follows the modified von Mises strain definition and its expression is provided below [57]:

$$\varepsilon_{eq} = \frac{k-1}{2k(1-2\nu)} + \frac{1}{2k} \sqrt{\frac{(k-1)^2}{(1-2\nu)^2} I_1^2 + \frac{2k}{(1+\nu)^2} J_2} \quad (35)$$

where I_1 and J_2 are the strain invariants given by the following formulas (ε is the strain tensor):

$$I_1 = \text{tr}(\varepsilon) \quad (36)$$

$$J_2 = 3\text{tr}(\varepsilon \cdot \varepsilon) - \text{tr}^2(\varepsilon) \quad (37)$$

The damage evolution laws relate the value of the damage variable d to the equivalent strain. For the sake of notation clarity, we note that $d = d(\varepsilon_{eq}^*)$ where ε_{eq}^* indicates any of the equivalent strains:

$$\varepsilon_{eq}^* = \begin{cases} \varepsilon_{eq} & (\text{local}) \\ \bar{\varepsilon}_{eq} & (\text{non-local gradient}) \\ \bar{\varepsilon}_{eq}^{NN} & (\text{I-FENN}) \end{cases} \quad (38)$$

In this study, we adopt two versions of an exponential-based damage model which are extensively used in the literature. The first version is the original model proposed by Mazars [90], in which damage depends on the equivalent strain based on the following relationship:

$$d(\varepsilon_{eq}^*) = \begin{cases} 0 & \text{if } \varepsilon_{eq}^* < \varepsilon_D \\ 1 - \frac{\varepsilon_D(1-\alpha)}{\varepsilon_{eq}^*} - \frac{\alpha}{\exp(\beta(\varepsilon_{eq}^* - \varepsilon_D))} & \text{if } \varepsilon_{eq}^* \geq \varepsilon_D \end{cases} \quad (39)$$

In the expression above, the variable ε_D represents a strain threshold value which signals damage initiation, and it is a material property. The second damage model is a slightly modified version of the original formulation, in which damage is calculated as follows [91]:

$$d(\varepsilon_{eq}^*) = \begin{cases} 0 & \text{if } \varepsilon_{eq}^* < \varepsilon_D \\ 1 - \frac{\varepsilon_D}{\varepsilon_{eq}^*} \{(1-\alpha) + \alpha \exp(\beta(\varepsilon_D - \varepsilon_{eq}^*))\} & \text{if } \varepsilon_{eq}^* \geq \varepsilon_D \end{cases} \quad (40)$$

In the definition of d provided in Equations (39) and (40), the parameter α is related to the residual strength in the material when the tensile strain approaches infinity, and the parameter β is associated with the softening branch of the curve with higher values indicating a steeper post-peak slope and a more brittle behavior [89]. A graphical illustration of the two versions is shown in Figure 5, along with the associated stress-strain curve in uniaxial tension for each model. Finally, we mention that maintaining thermodynamic consistency enforces damage irreversibility, and therefore the Karush-Kuhn-Tucker relationships need to be satisfied [92].

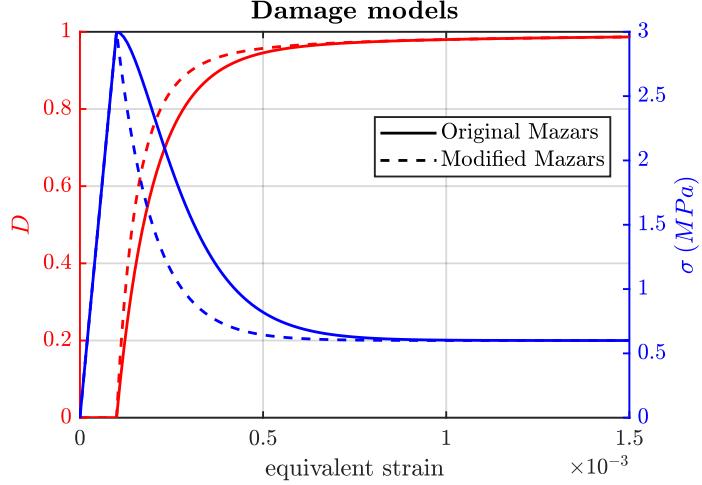


Figure 5: Graphical illustration of the damage evolution (red) and the associated stress-strain (blue) curves for uniaxial tension. The original Mazars model [90] is shown with continuous lines, and the modified version is depicted with dashed lines. In both cases $\alpha = 0.8$, $\beta = 10000$, $\varepsilon_D = 0.0001$, and the modulus of elasticity is $E = 30000\text{ MPa}$.

5. Numerical Examples

This section presents the numerical implementation of the proposed methodology. Three distinct cases are analyzed, in order to examine the applicability and potential of I-FENN. First, in section 5.1 we apply the method to a domain with a structured finite element mesh and a single crack. Then, in section 5.2 we extend our investigation to a geometry with a structured mesh and two cracks, to examine whether the neural network can identify the presence of multiple strain localization regions. Finally, in section 5.3 we implement I-FENN to a geometry with a single crack and an unstructured mesh, to verify that the choice of mesh discretization does not impact the learning ability of the neural network. In all cases we compare our results with the conventional non-local gradient method [55]. The load application is displacement-controlled. For the remainder of this section the applied displacement/load is shown by means of a relative loadfactor value and it is denoted with the symbol lf . This variable depicts the applied displacement at the load increment of interest \mathbf{u}^{app} normalized by the value of the total applied displacement $\bar{\mathbf{u}}$, therefore $lf = \frac{\mathbf{u}^{app}}{\bar{\mathbf{u}}}$. Table 2 contains the values for the hyperparameters which are used for the training of the neural network of each test case. Specifically, we mention the number of layers, neurons per layer, epochs and learning rate.

Table 2: List of hyperparameters used for the training of each test case.

	Single notch			Double notch		L – shaped		
	$lf = 0.42$	$lf = 0.70$	$lf = 0.82$	$lf = 0.45$	$lf = 0.70$	$lf = 0.25$	$lf = 0.50$	$lf = 0.70$
#layers	10	10	10	10	10	10	10	10
#neurons/layer	50	50	50	140	120	100	100	140
#epochs	20000	20000	20000	30000	100000	30000	50000	100000
learning rate	0.001	0.001	0.001	0.001	0.001	0.0005	0.0005	0.0005

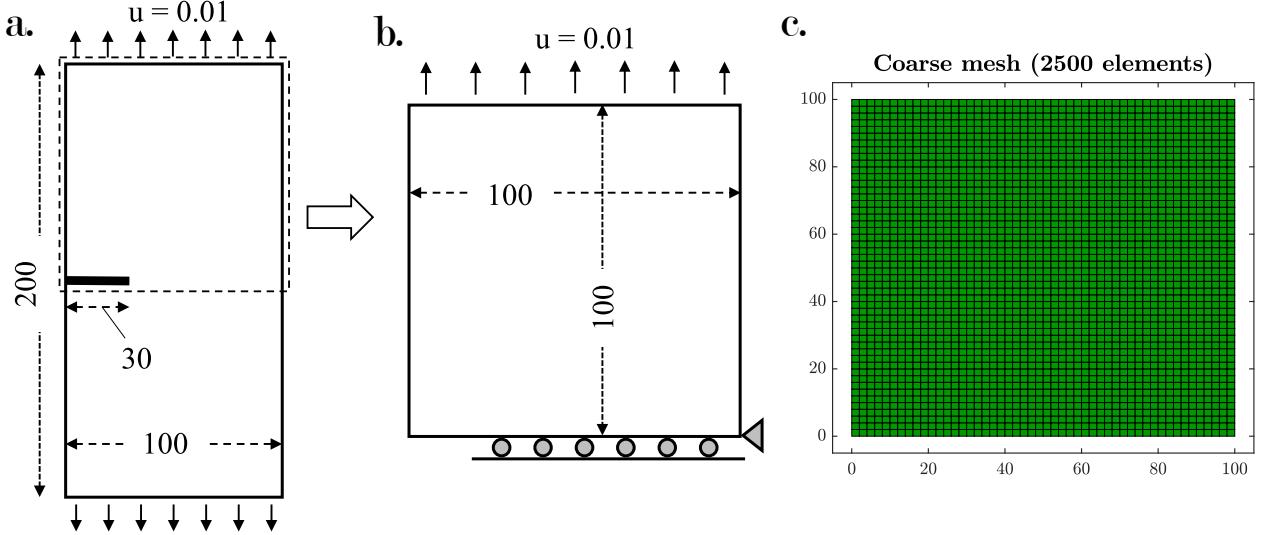


Figure 6: **a.** Schematic illustration of the geometry, boundary conditions and loading for the single-notch case. **b.** Due to symmetry, the upper half of the initial geometry is used in all numerical analyses. Domain continuity across the middle centerline is restored numerically through rollers. **c.** The *Coarse* finite element model of the domain, which is developed using a structured mesh of 2500 elements. All length units are in mm.

5.1. Single-notched specimen

The first numerical example is a mode-I loading case of a single-notch specimen. The geometry under consideration is shown in Figure 6. The symmetric nature of the problem allows the isolation and analysis of only the upper half of the domain, as shown in Figure 6b. A series of rollers is applied at the horizontal centerline of the intact domain, constraining the y-displacement of these nodes. The x-displacement of the bottom right node is also constrained. The shear modulus is taken as $G = 125000$ KPa and Poisson's ratio is $\nu = 0.2$. Plane strain conditions are considered, and the convergence tolerance variable is set to $tol = 10^{-6}$.

In order to generate the training data, the FE model is first solved using the non-local gradient model described in Appendix B. The domain is discretized using three structured meshes of varying element size. In the rest of this subsection, the three models are referred to as the *Coarse*, *Intermediate* and *Fine*, and they each have 2500, 6400 and 10000 elements in total, respectively. Each side is discretized with 50, 80 and 100 elements and the finite element lengths for each case are $l_{elem} = 2\text{mm}$, $l_{elem} = 1.25\text{mm}$ and $l_{elem} = 1\text{mm}$, respectively. In all cases the characteristic length is taken as $l_c = 4\text{mm}$, thus ensuring that l_c is at least twice the maximum element length. In this example we use the local strain definition of Equation (34) and the damage law given in Equation (39), with $\alpha = 0.70$ and $\beta = 10000$.

First, we verify the mesh-independent character of the numerical solution. Figure 7 shows the reaction-loadfactor curves for the three cases. It can be observed that the resulting curves essentially coincide, which is the first indication of the solver mesh-independence. Additionally, Figure 8 shows a comparison of the non-local strain propagation across the three different meshes, with snapshots at the loadfactors marked on Figure 7. These are $lf = 0.42, 0.72, 0.82$. The first row of graphs corresponds to $lf = 0.42$, where the domains are still in the elastic regime. The second row corresponds to $lf = 0.70$, where damage has already

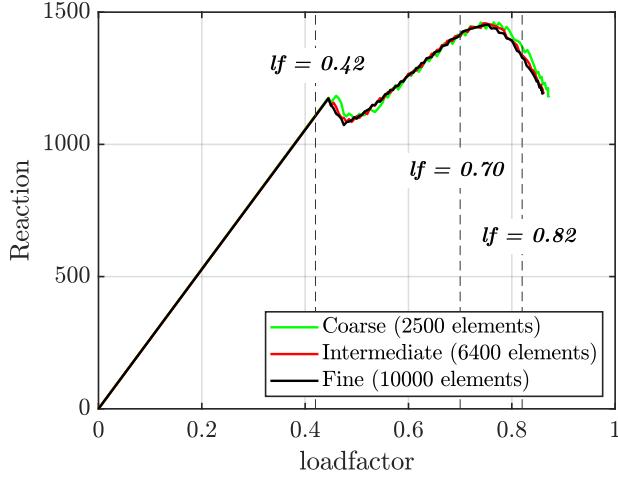


Figure 7: Reaction force-loadfactor curves for the single-notch case using the various mesh configurations.

initiated but the model is still in the hardening zone. The third row corresponds to $lf = 0.82$, which falls in the softening regime of the response. Following the same notation, Figure 9 demonstrates how damage evolves at these load increments. For the sake of a transparent cross-mesh comparison, only the crack tip neighbourhood ($20 \leq x \leq 60, 0 \leq y \leq 40$) is shown, and the black grid in all subplots indicates the finite element edges. It is evident that both the non-local strain $\bar{\varepsilon}_{eq}$ and the damage variable d share an almost identical profile across the three idealizations. This consistency further verifies that the non-local gradient method yields mesh-objective results throughout the entire load history.

Figure 10 shows the comparison between FEM and I-FENN for the Coarse case at $lf = 0.42$. The domain is still in the elastic region and only the non-local strain profiles are shown. The first column of plots shows the *FEM - true* solutions for the non-local strain (top) and converging residuals (bottom). Within the residuals subplot, the red line corresponds to the L2-norm of the internal stress residual vector \mathbf{R} calculated based on Equation (33). The blue line refers to the L2-norm of the displacement vector incremental change $\delta\hat{\mathbf{u}}$, and we remind the reader of the convergence criterion in Equation (28). The second column of subplots depicts the *I-FENN - predicted* solutions for the non-local strain (top) and convergence behavior (bottom). We observe a remarkable similarity between the predicted and targeted non-local strains. This similarity is both qualitative, captured by the overall contour shape, but also quantitative. The latter is verified by the top right subplot of Figure 10 which shows the Relative Squared Error (RSE) between the two. The formula for this error metric is shown in Equation (41), and it is used to measure the relative error for both $\bar{\varepsilon}_{eq}$ and d :

$$\bar{\varepsilon}_{eq,RSE} = \frac{(\bar{\varepsilon}_{eq,FEM} - \bar{\varepsilon}_{eq,I-FENN})^2}{(\bar{\varepsilon}_{eq,FEM})^2} \quad (41a)$$

$$d_{RSE} = \frac{(d_{FEM} - d_{I-FENN})^2}{(d_{FEM})^2} \quad (41b)$$

We underline that the plots in the first row of Figure 10 do not depict a continuous contour throughout the domain. Instead, each filled circle in these plots represents the value of the variable of interest in the

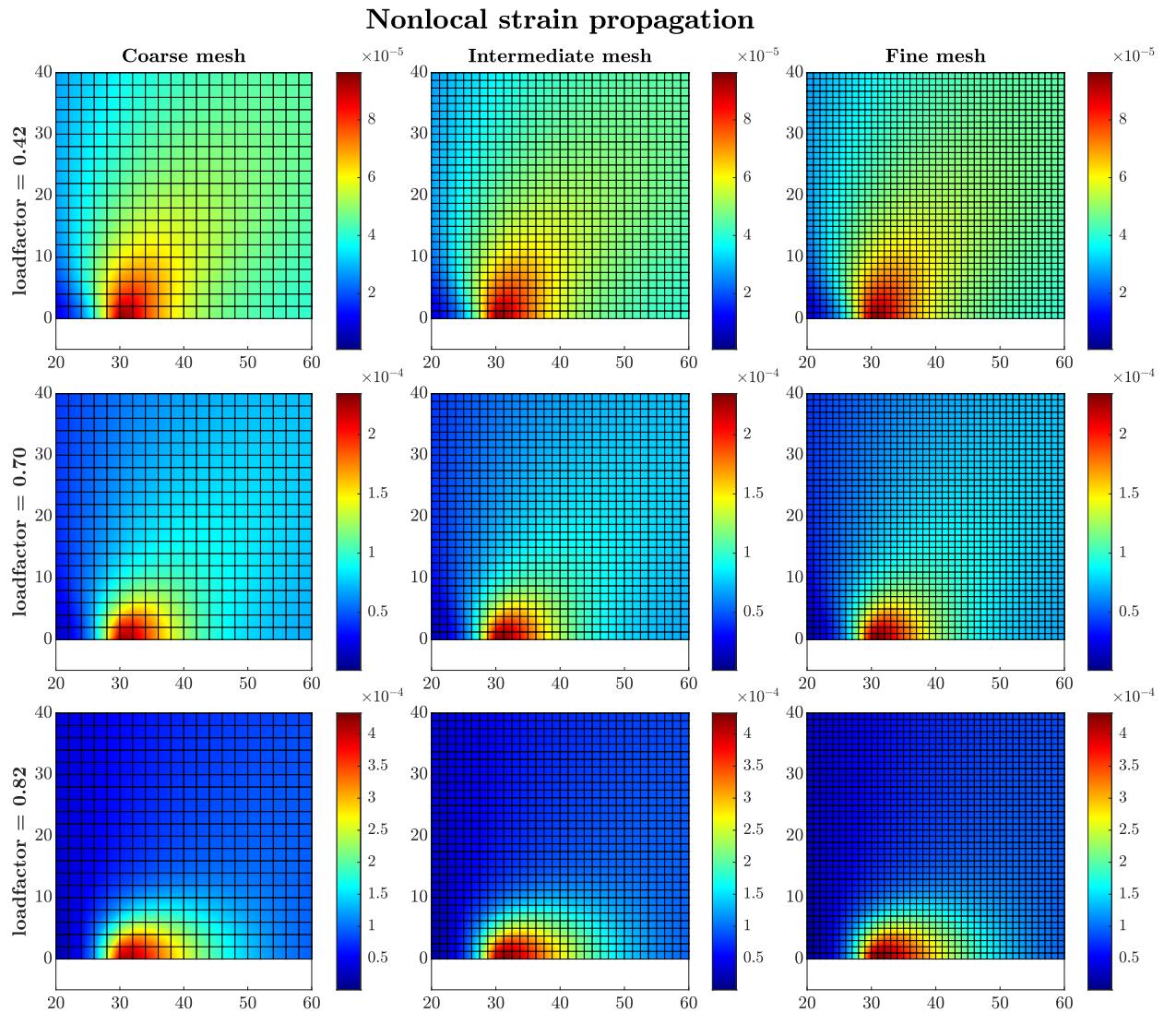


Figure 8: Snapshots of non-local equivalent strain $\bar{\varepsilon}_{eq}$ propagation across the Coarse, Intermediate and Fine meshes at loadfactor values $lf = 0.42, 0.70$ and 0.82 . For the sake of clarity, only a zoomed-in area around the crack tip is shown. The resulting contours demonstrate the mesh-independent nature of the non-local gradient solver.

Nonlocal damage propagation

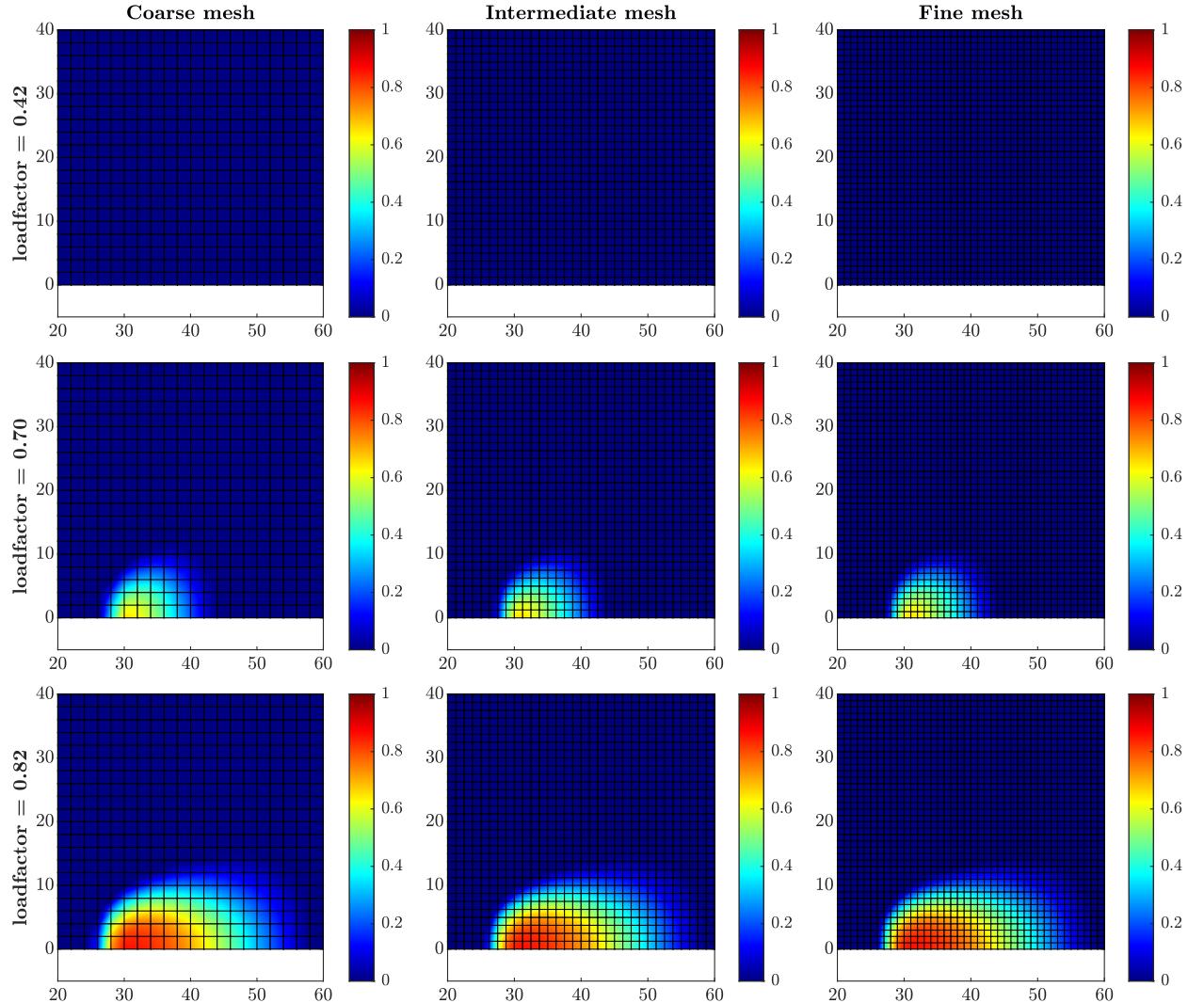


Figure 9: Snapshots of non-local damage d propagation across the three different mesh idealizations at loadfactor values $lf = 0.42, 0.70$ and 0.82 . The contours are shown only for the region around the crack tip and they illustrate the mesh-objective character of the non-local gradient formulation.

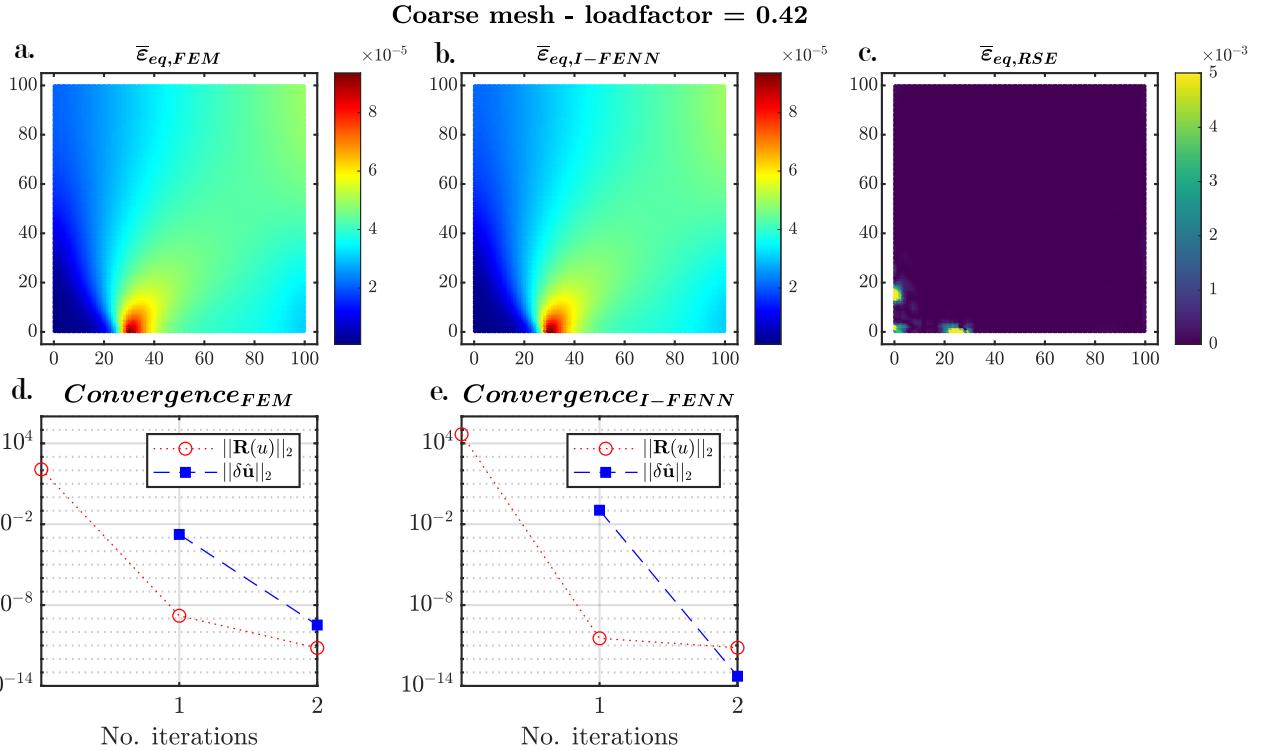


Figure 10: Comparison between FEM (left column) and I-FENN (middle column) for the Coarse model of the single-notch case, at a loadfactor value $lf = 0.42$. **a-b.** The non-local equivalent strain $\bar{\varepsilon}_{eq}$ maps for the two methods. Each filled circle in these plots indicates the value of $\bar{\varepsilon}_{eq}$ in the corresponding integration point. **c.** The relative squared error of $\bar{\varepsilon}_{eq}$ calculated at each integration point using Equation (41). With regards to the entire non-local strain vectors, the L2-norm of their differences is $\|\bar{\varepsilon}_{true} - \bar{\varepsilon}_{pred}\|_2 = 1.88 \times 10^{-5}$. **d-e.** Convergence performance of the two methods. The red line indicates the L2-norm of the residual vector \mathbf{R} , and the blue line shows the L2-norm of the displacement vector incremental change $\delta\hat{\mathbf{u}}$.

corresponding integration point. This allows for a point-by-point comparison between the two methods, and ensures that the relative squared error is not smeared out due to plotting interpolation. A close observation of Figure 10e reveals that the $\bar{\varepsilon}_{RSE}$ lies even below 0.1% throughout the vast majority of the domain. This is an exceptional approximation of the target values, while there are only a few IPs where relative squared error gets higher. These points are located in the bottom left corner of the domain and specifically behind the crack tip. The elevated $\bar{\varepsilon}_{eq,RSE}$ at these points is attributed to the numerical issue of the range of $\bar{\varepsilon}_{eq}$ values. The non-local strain values in this area range between 10^{-6} and 10^{-5} , whereas in the rest of the domain they range between 10^{-5} to 10^{-4} . This difference in the order of magnitude yields more pronounced differences in the strain relative error, even if the absolute error is smaller in these locations as compared to the rest of the domain. This numerical feature remains present throughout the rest of the examined cases and it requires additional efforts on adapting a generalized PINN structure and training strategy to be alleviated, as discussed in Section 3.1. For example, scalable approaches with individual subdomain normalization of the input data such as the FBPINNs framework [87] could be a promising candidate to resolve this numerical issue. Nevertheless, we note that the area where the elevated values of $\bar{\varepsilon}_{eq,RSE}$ occur is sufficiently distanced from the damage process zone, and since the strain values are substantially low they do not affect the accuracy of damage predictions. This is shown more clearly in the next figures, where damage is also present.

Another important observation from Figure 10 is that the L2-norms of \mathbf{R} and $\delta\hat{\mathbf{u}}$ which are computed with I-FENN are continuously decreasing, until they have converged. This is the most crucial component of a successful non-linear finite element analysis, and proves the applicability of our proposed methodology. It is also important to note that our method converges with just two iterations, similar to the FEM solution, as we would expect for a load increment in the elastic regime. Based on these observations, we can conclude that Figure 10 provides evidence of both a) the successful training of the physics-informed neural network, and b) the feasibility of integrating it within the element stiffness function.

Coarse mesh - loadfactor = 0.70

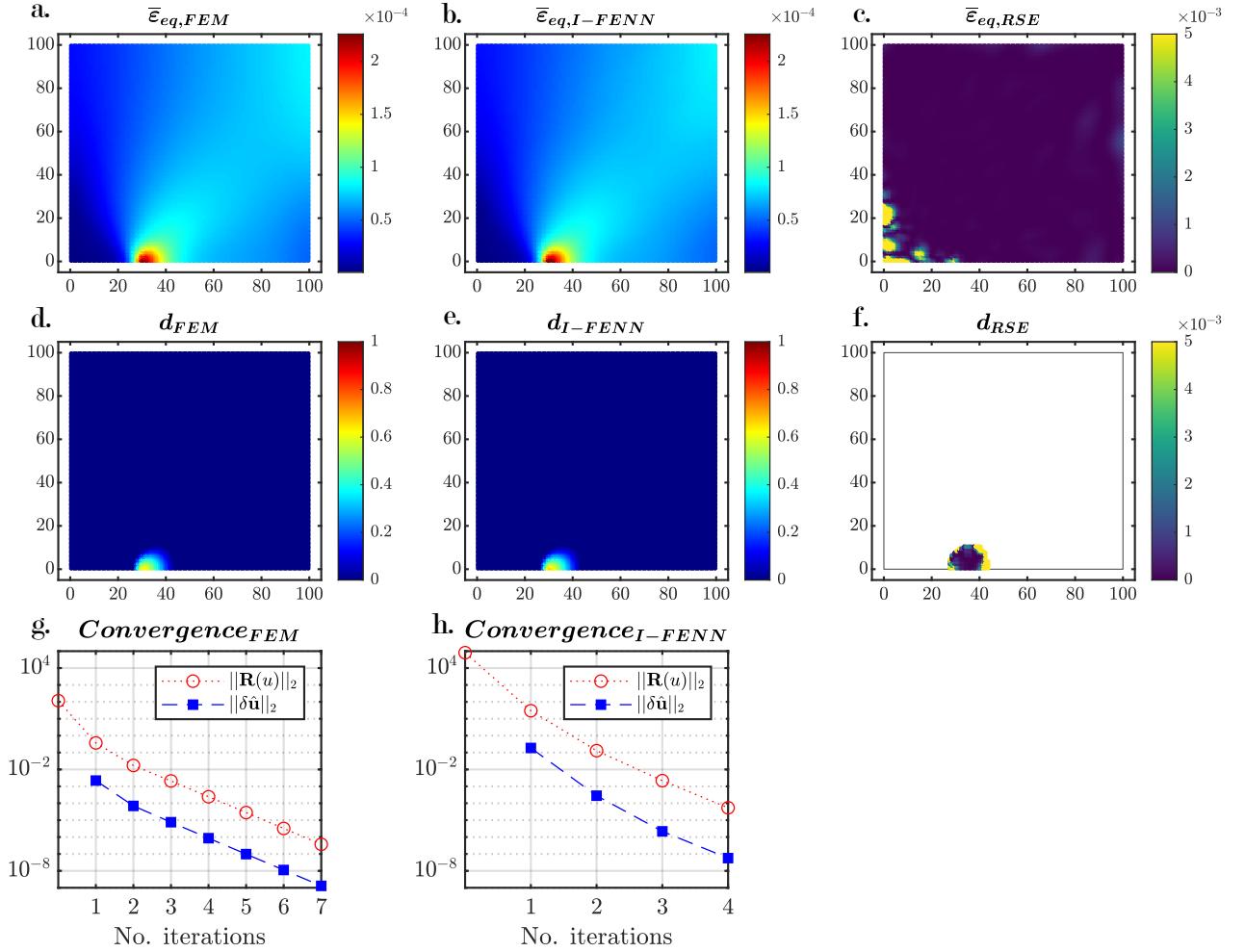


Figure 11: Comparison between FEM (left column) and I-FENN (middle column) for the Coarse model of the single-notch case, at loadfactor value $lf = 0.70$. The non-local equivalent strain and damage values at each integration point are shown in subplots **a-b** and **d-e** respectively. The relative squared error for the two variables is shown in subplots **c** and **f**, calculated using Equation (41). The white space in subplot **c** is due to the FEM value of $d = 0$, and therefore d_{RSE} is not defined. The L2-norm of the strain vector is $\|\bar{\varepsilon}_{FEM} - \bar{\varepsilon}_{I-FENN}\|_2 = 5.167 \times 10^{-5}$. **g-h.** Convergence of the two methods. The red line indicates the L2-norm of the residual vector \mathbf{R} , and the blue line shows the L2-norm of the displacement vector incremental change $\delta\hat{\mathbf{u}}$.

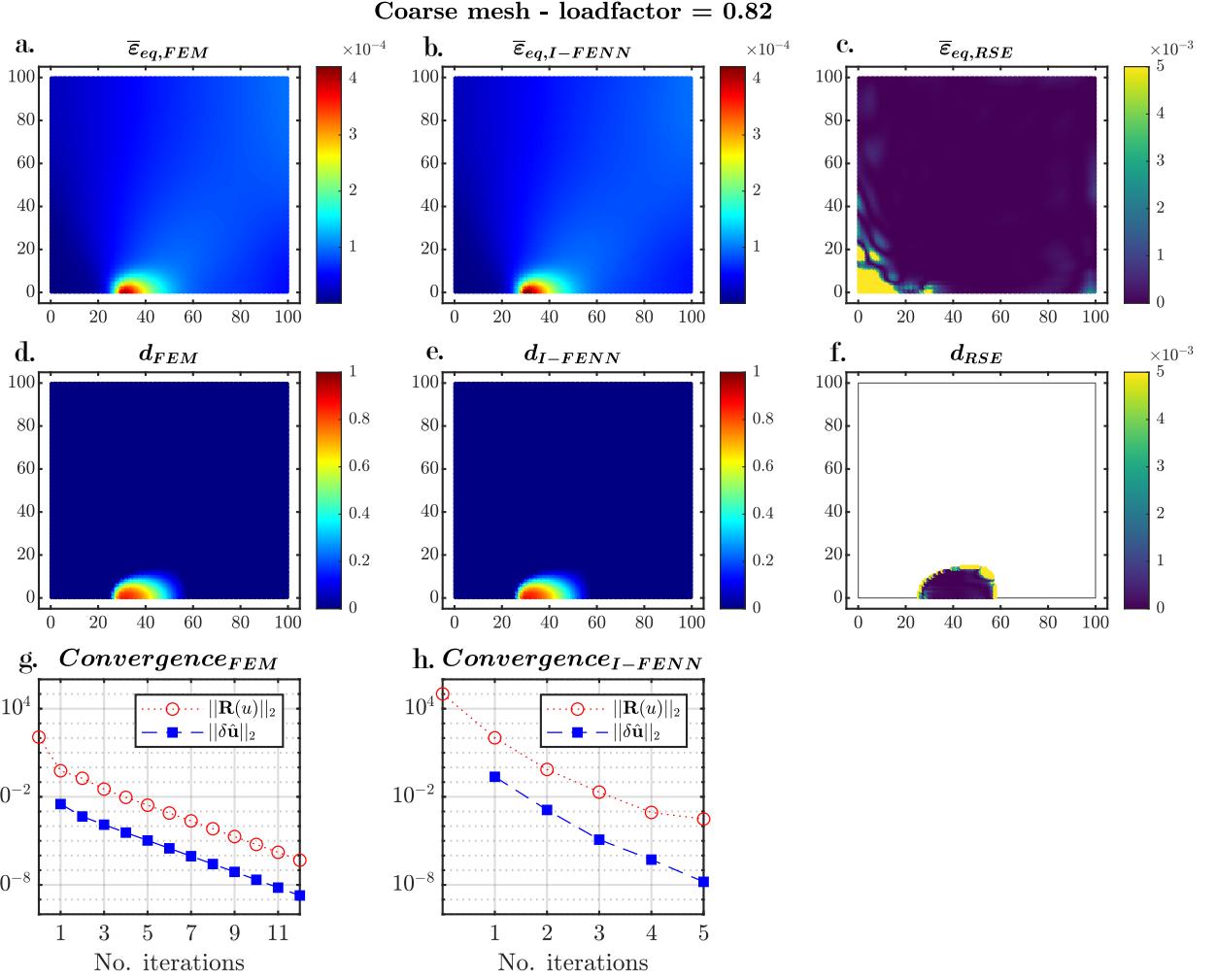


Figure 12: Comparison between FEM (left column) and I-FENN (middle column) for the Coarse model of the single-notch case at loadfactor value $lf = 0.82$. The non-local strain and damage values at each integration point, along with the relative squared error, are shown in the top and middle row respectively. The L2-norm of the $\bar{\varepsilon}_{eq}$ is $\|\bar{\varepsilon}_{FEM} - \bar{\varepsilon}_{I-FENN}\|_2 = 1.296 \times 10^{-4}$. The bottom row of subplots illustrates the convergence performance of the two methods. The red line indicates the L2-norm of the residual vector \mathbf{R} , and the blue line shows the L2-norm of the displacement vector incremental change $\delta\hat{\mathbf{u}}$.

Figures 11 and 12 show a similar comparison for the Coarse case, at $lf = 0.70$ and $lf = 0.82$ respectively. The subplot layout is similar to Figure 10, with the addition of the damage contours in the middle row of the figures. Even in the substantially more challenging damage-dominated zone, the evident conclusion is that I-FENN is capable of capturing with very good accuracy both the non-local strain and damage profiles. As far as the non-local strains are concerned, this observation is again subtly violated only at the area behind the crack tip, which is not critical for the crack propagation. With respect to the damage landscape, I-FENN also captures with very good accuracy its spatial extent and magnitude. We note that the white space in Figures 11h and 12h is due to the FEM $d = 0$, and therefore d_{RSE} is mathematically undefined. Finally, we emphasize the continuously decreasing L2-norms of \mathbf{R} and $\delta\hat{\mathbf{u}}$ in Figures 11g and 12h, which further verify the feasibility of our method in the presence of damage.

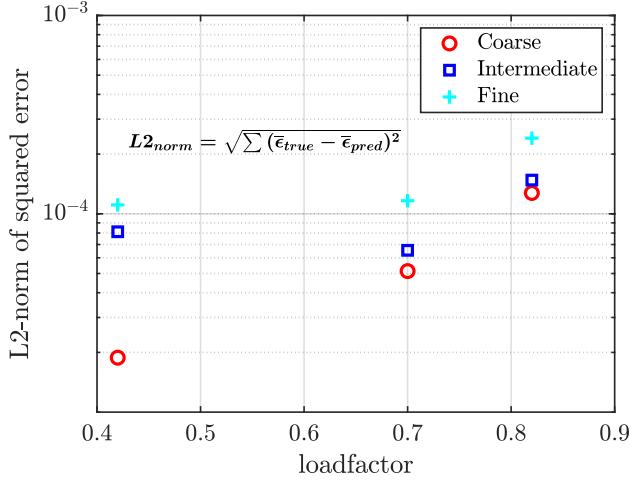


Figure 13: The L2-norms of the non-local equivalent strain errors when the PINN is trained and tested in the same mesh resolution. The formula for the L2-norm of the strain vector absolute error reads: $\| \cdot \|_2 = \sqrt{\sum (\bar{\varepsilon}_{eq,FEM} - \bar{\varepsilon}_{eq,I-FENN})^2}$

Additionally, we provide in Figure 13 a concise way to monitor the error metrics. Figure 13 shows the L2-norms of the non-local strain vector absolute error between FEM and I-FENN. In this graph the PINNs are trained and tested against the same mesh resolution. Overall we observe values of the L2-norm in the order of 10^{-5} to 10^{-4} , while there are two general trends: a) the error value increases monotonically as the mesh resolution is refined, and b) the error value tends to increase as damage grows and the domain enters the softening stage. Both trends are rather anticipated, the first due to already discussed presence of more integration points contributing to the error accumulation, and the second due to the higher absolute values in the non-local strain profile as damage increases.

Next, we examine a critical aspect of our methodology: the ability of the neural network to generalize across different mesh densities. Evidently, the gain from training a PINN on a coarsely discretized geometry and utilizing it for predictions in substantially finer meshes is a crucial feature from a computational efficiency standpoint, opening another pathway through which I-FENN could decrease the numerical cost of numerical simulations. Here we touch upon this path as well, by testing the three PINNs which are trained on the Coarse mesh against the data from the other two finer mesh discretizations at the same loadfactor values. From a computational execution standpoint, this is feasible because we maintain the same number for the input matrix (4 variables: x, y, g, and ε_{eq}) and output vector (1 variable: $\bar{\varepsilon}_{eq}$) variables, regardless the mesh density. This ensures the consistency between the inner dimensions of the weight matrices multiplication inside the network, and therefore only the outer dimensions of the input matrix and output vector change, which correspond to the number of collocation points. We also underline that in this comparative study, we evaluate the trained PINN itself, without integrating it inside the nonlinear solver. In other words, we feed the input from the Intermediate/Fine mesh to the Coarse-trained PINN, and implement only one forward propagation pass to get the output predictions without further refining the predictions using the Newton-Raphson solver.

The results of this study are shown in Figures 14 and 15, for the Intermediate and Fine models respectively, and each row of plots corresponds to a different loadfactor value. Across the entire domain, we observe a

Trained on Coarse – Tested on Intermediate

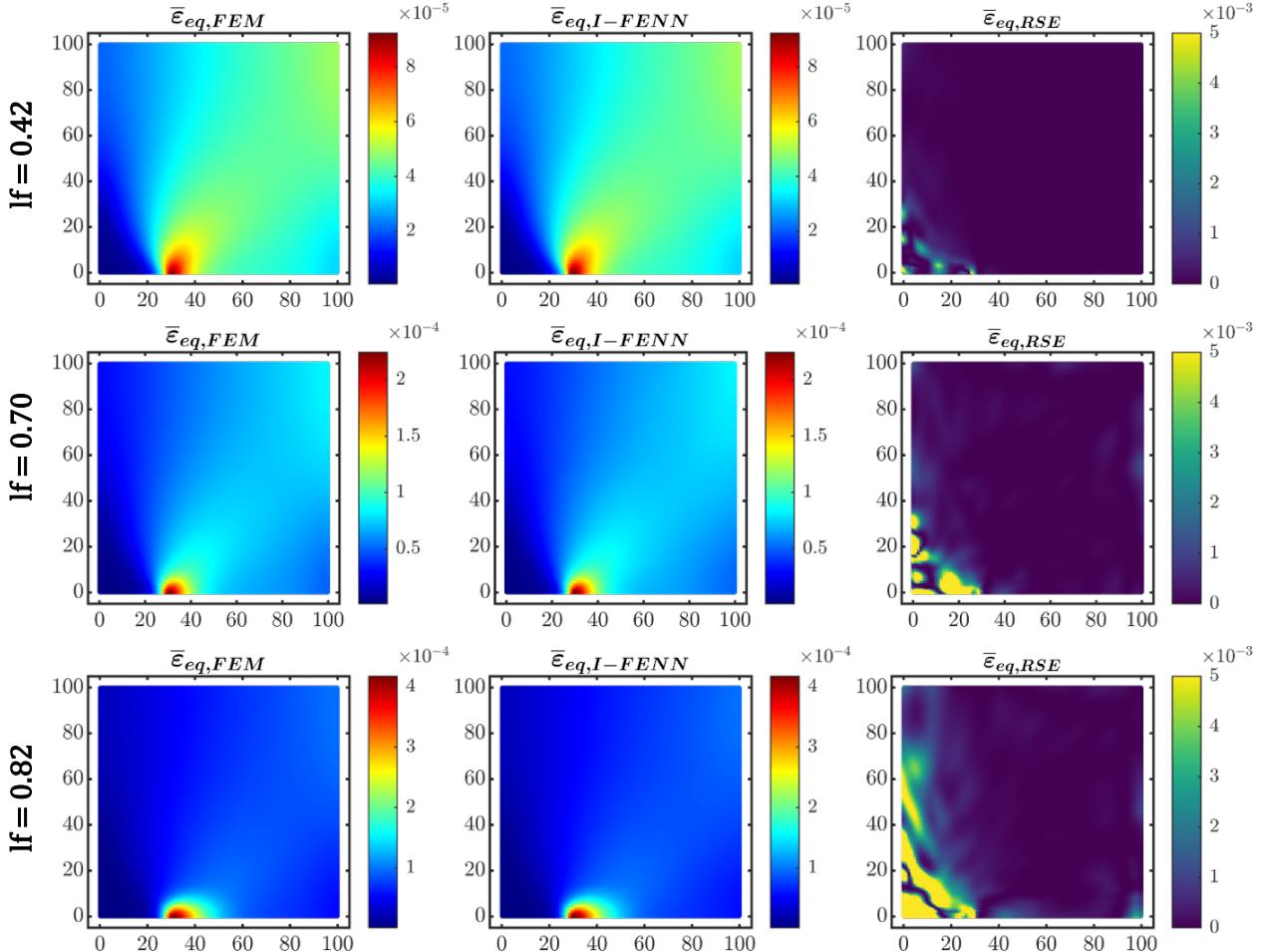


Figure 14: Cross-mesh generalization: The PINN is trained on the Coarse mesh data of the single-notch case and it is used to predict the non-local strain $\bar{\varepsilon}_{eq}$ profile in the Intermediate mesh model. The left column shows the FEM results of $\bar{\varepsilon}_{eq}$ in the Intermediate mesh, the middle column illustrates the I-FENN predictions, and the right column depicting the relative squared error. Each row of subplots corresponds to a different loadfactor value.

Trained on Coarse – Tested on Fine

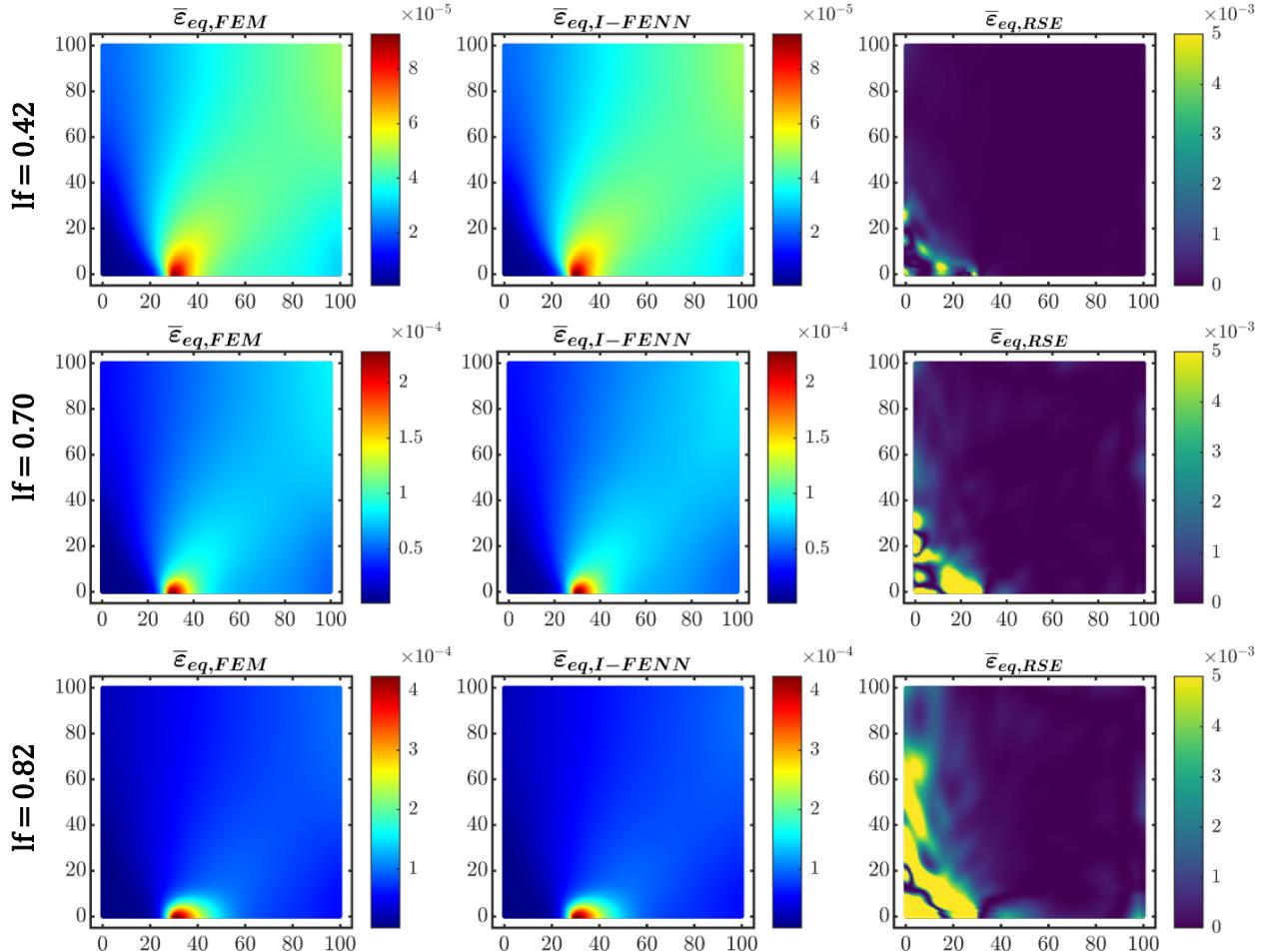


Figure 15: Cross-mesh generalization: The PINN which is trained on the Coarse mesh data of the single-notch case is used to predict the non-local strain $\bar{\varepsilon}_{eq}$ profile in the Fine mesh model. The left column shows the FEM results of $\bar{\varepsilon}_{eq}$ in the Intermediate mesh, the middle column illustrates the I-FENN predictions, and the right column depicting the relative squared error. Each row of subplots corresponds to a different loadfactor value.

Trained on Fine - Tested on Coarse

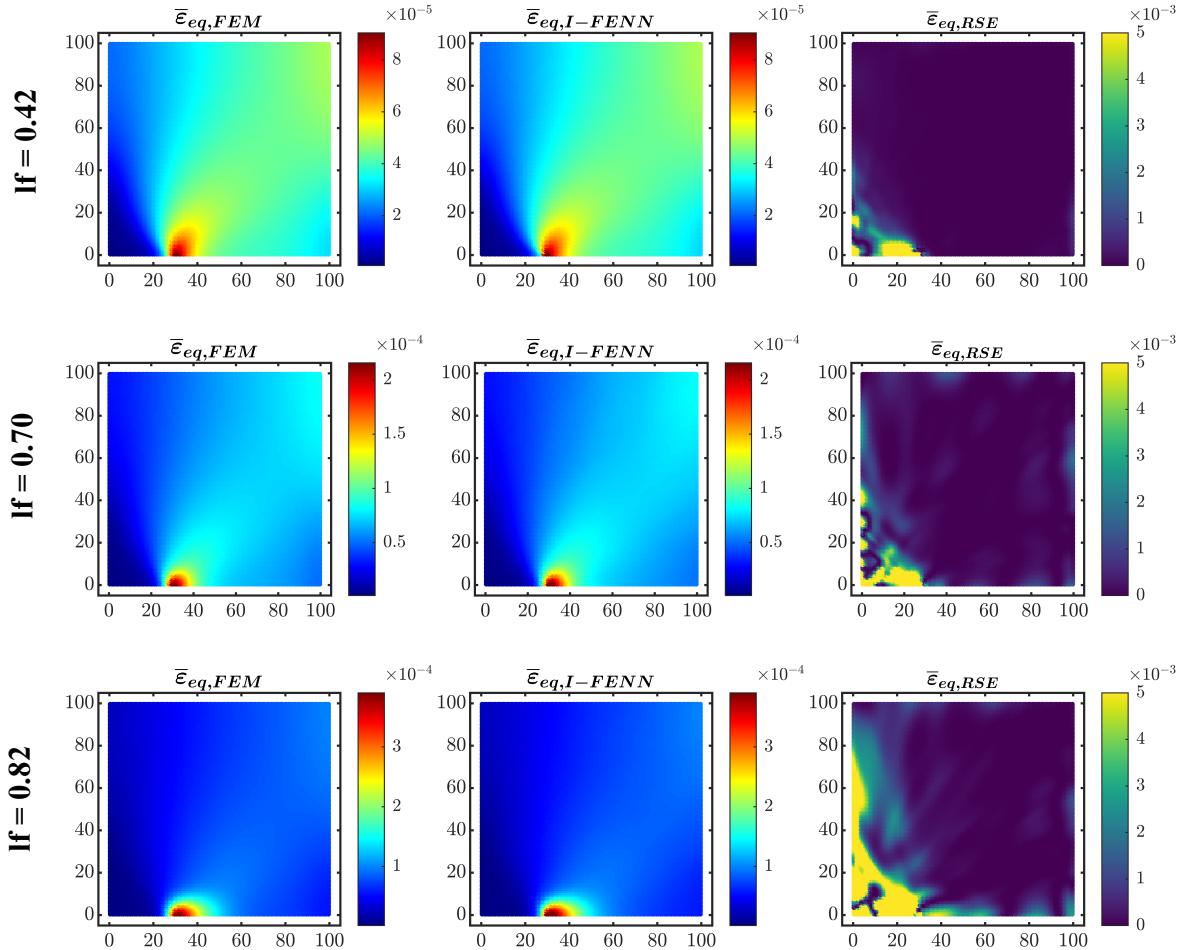


Figure 16: Cross-mesh generalization: The PINN which is trained on the Fine mesh data of the single-notch case is used to predict the non-local strain $\bar{\varepsilon}_{eq}$ profile in the Coarse mesh model. The left column shows the FEM results of $\bar{\varepsilon}_{eq}$ in the Intermediate mesh, the middle column illustrates the I-FENN predictions, and the right column depicting the relative squared error. Each row of subplots corresponds to a different loadfactor value.

remarkable resemblance of the non-local strain profile between the FEM and I-FENN calculations for all three loadfactor values. The relative squared error is particularly low when the comparison is conducted in the elastic regime, which corresponds to the first row of plots in Figures 14 and 15. It can be observed that the error metric performance gradually deteriorates as we move deeper into the presence of damage; however, this is an artifact of the growing difference between the maximum and minimum strain value range. The spread in the $\bar{\varepsilon}_{eq,RSE}$ is observed to increase in the regions where the strains are much lower than the damage initiation threshold strain ε_D and do not contribute to damage evolution. Additionally, a closer observation in the RSE plots between Figures 14 and 15 reveals that the Coarse-trained network performs slightly better on the Intermediate mesh as compared to the Fine mesh. This is anticipated, as the larger number of integration points in the Fine mesh is expected to dilate the accumulated error; nevertheless the non-local strain profile is still captured with very good accuracy as shown in the second column of plots in Figure 15.

The converse response is also investigated and the results are shown in Figure 14. A PINN which is trained in the Fine mesh discretization is tested against the data generated with the Coarse mesh, at the loadfactor values of interest. The network is capable of predicting the nonlocal strain profile with good accuracy. The most accurate results are obtained when training and testing occurs at identical mesh refinement levels, but overall, this parametric study shows promising results on the feasibility of training and testing a PINN at different mesh discretizations. This is expected to have a significant impact in accelerating the numerical analysis of densely meshed geometries in multi-scale and multi-physics simulations.

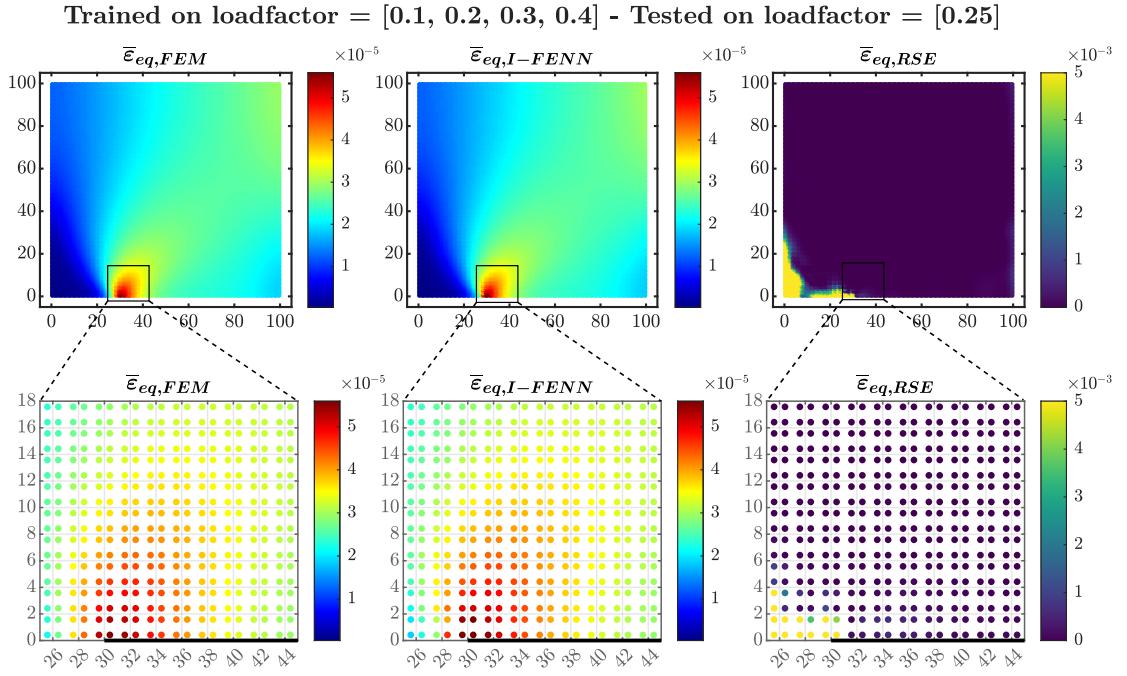


Figure 17: Cross-load history generalization: The PINN is trained on input data from four elastic load increments with $lf = 0.1, 0.2, 0.3$ and 0.4 , and then tested against the data corresponding to $lf = 0.25$. The top row of subplots shows the FEM (left) and I-FENN (middle) results for the non-local strain $\bar{\varepsilon}_{eq}$, and the relative squared error is shown in the right. The bottom row of graphs show a point-by-point comparison in a zoomed-in region around the crack tip, further demonstrating the predicting accuracy of I-FENN.

Thus far, we have illustrated how a neural network which is trained and tested at the same load increment is capable of predicting the non-local strain profile. Next we examine how this approach can be generalized across multiple load increments. First, we confine our interest in the elastic regime and train a PINN with input data from four load increments, namely at $lf = 0.1, 0.2, 0.3$ and 0.4 . We then test the ability of the PINN against unseen input data generated at loadfactor $lf = 0.25$. We also note that for this numerical study, where the PINN is trained and tested at different loadfactor values, these values are used as an additional input variable into the network. The results of this parametric study are shown in Figure 17. The top series of graphs shows a comparison between FEM and I-FENN non-local strain values throughout the entire domain. The bottom series of plots zooms on the crack tip neighbourhood, allowing for an even more clear point-by-point comparison in the region of high strains concentration. We observe a very good correlation between the

target and predicted values, both in the vicinity of the crack tip and also across the full domain.

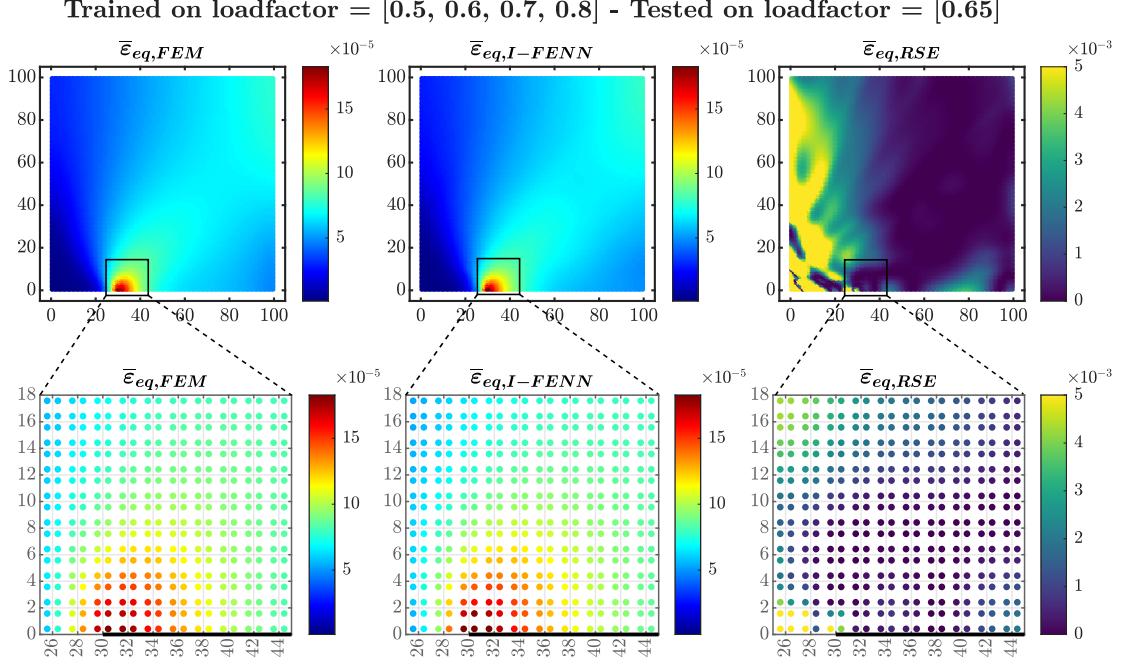


Figure 18: Cross-load history generalization: The PINN is trained on input data from four inelastic load increments with $lf = 0.5, 0.6, 0.7$ and 0.8 , and it is used to make predictions at $lf = 0.65$. The top row of subplots shows the FEM (left) and I-FENN (middle) results for the non-local strain $\bar{\varepsilon}_{eq}$, and the relative squared error is shown in the right. I-FENN is capable of capturing with sufficient accuracy the strain landscape, particularly around the critical crack-tip area, as it is shown in the zoomed-in subplots in the bottom row of the figure.

We then follow a similar approach for the inelastic zone, training the PINN with data from $lf = 0.5, 0.6, 0.7$ and 0.8 and testing against $lf = 0.65$. The results of this parametric study are shown in Figure 18. As shown in the top row of graphs, even though the network shows a slightly deteriorated performance when compared to the study conducted in the elastic regime, it is still capable to generalize with adequate accuracy both overall and particularly in the zone where damage is expected to propagate. This is more clearly demonstrated in the bottom series of zoomed-in graphs, where the clarity of these plots enables a straightforward comparison of the critical integration points behavior.

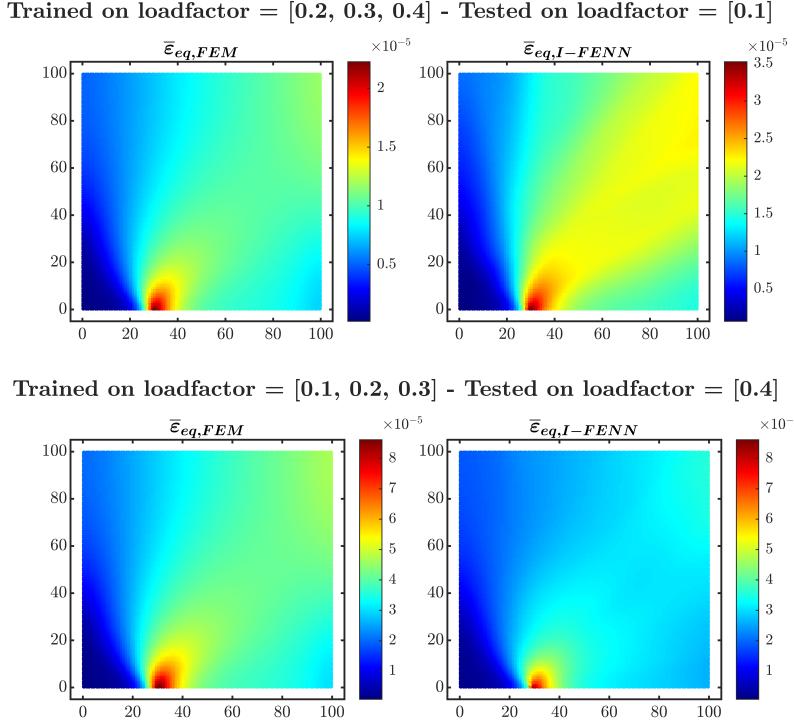


Figure 19: Cross-load history generalization: The PINN is trained on input data from three load increments and tested against an unseen load value which lies outside the range of the training dataset. The top row of subplots shows the case where the training dataset corresponds to $lf = 0.1, lf = 0.2, lf = 0.3$ and the test dataset to $lf = 0.4$ (extrapolation to a higher loadfactor value). The bottom row of subplots shows the case where the training dataset corresponds to $lf = 0.2, lf = 0.3, lf = 0.4$ and the test dataset to $lf = 0.1$ (extrapolation to a lower loadfactor value). The left column shows the FEM results and the right the I-FENN results for the non-local strain $\bar{\epsilon}_{eq}$. The PINN approximates qualitatively the non-local strain profile in both cases, but fails to capture accurately its magnitude.

Until this point, generalization of the PINN predictive capability has been attempted in an *interpolating* sense, since the unseen test dataset corresponds to a loadfactor which lies in between the training ones. We also attempt to examine the *extrapolating* capability of the trained PINN, by conducting two additional numerical studies. We have trained a PINN on loadfactor values $lf = 0.1, 0.2, 0.3$ and tested on $lf = 0.4$, as well as training on $lf = 0.2, 0.3, 0.4$ and testing on $lf = 0.1$. These two parametric studies aim to shed light on how the PINN performs when the unseen test dataset lies either above or below the training dataset, and the results of this investigation are shown in Figure 19. The PINN approximates qualitatively the non-local strain profile in both cases, but fails to capture correctly its magnitude. This is rather anticipated, since PINNs in their standard form are well-known for their poor extrapolation performance, which is also confirmed in this case.

Overall, the results presented at Figures 17, 18 and 19 shed some initial yet promising light towards the feasibility of the PINN to generalize beyond the input data it has been trained upon. Even in its current vanilla formulation, the PINN demonstrates a good interpolating capability. This is a crucial dimension of I-FENN, since it potentially allows to carefully select few training datasets that correspond to the loadfactor range boundaries, as well as a few in-between steps, and based on the interpolation capability of the PINN

make predictions in the intermediate range. Yet, we underline that the purpose of Figures 17 - 19 is only to provide some light on the generalization aspect of the PINN across the load history, and not to establish rigorous guidance on how this generalization should be executed. For this purpose, more sophisticated network architectures such as GRUs or LSTMs may prove more suitable candidates.

To summarize, in this section we have implemented the new non-local damage model to a geometry with mode-I loading conditions and a structured finite element mesh. Three different discretizations of varying mesh density were used. After demonstrating the non-local character of the non-local gradient solver across the three models, we illustrated the following key points:

- The pre-trained PINN is capable of learning accurately the local to non-local strain transformation at the offline training stage. This is shown in the first row of subplots in Figures 10 - 12, where the non-local strain profile from I-FENN shows very good resemblance to the FEM solution.
- The PINN is integrated within the element-level stiffness function and it is repeatedly executed within the iterative Newton-Raphson method. Its input and output variables are continuously updated as it acts jointly with the finite element interpolation functions. As a result, a damage profile with a non-local character is generated, which matches with remarkable accuracy the damage profile from FEM. This is illustrated in the second row of subplots in Figures 10 - 12
- The robust convergence of the Newton-Raphson method in I-FENN is demonstrated in the bottom row of subplots in Figures 10 - 12. The L2-norms of the residual force vector and incremental displacement change are continuously decreased, until the convergence criterion of Equation 25 is satisfied.
- We tested the neural network which is trained on the Coarse mesh against the other two mesh models, across all three loadfactor values. The results are shown in Figures 14 - 15. We observe a very good match between the FEM and I-FENN predictions, which shows evidence of a cross-mesh generalization capability of the proposed methodology.
- We trained a neural network on four elastic load increments and tested against the unseen data from another load increment, and we repeated this study in the inelastic zone. The results of this study are shown in Figures 17 - 18. In both cases we observe very good predictions of I-FENN, a trend which is even more profound in the elastic load case.

This section has a) illustrated the feasibility of I-FENN and b) it has shed some initial light across potential generalization pathways. In the next two subsections we are mainly focusing on the first aspect, and we demonstrate the applicability of I-FENN on more challenging geometries by extending our investigation to: a) a geometry with two cracks and a structured mesh, and b) a geometry with one crack and unstructured mesh.

5.2. Double-notched specimen

The next numerical example is a direct tension test on a double-notched geometry that has been previously modeled in [89], and the purpose of this implementation is to examine the ability of the network to recognize

the presence of more than one crack locations. The geometric details of the domain are shown in Figure 20a. The specimen is fixed on the bottom edge and a vertical upward displacement load is applied at the top nodes. The shear modulus is $G = 125000$ KPa and Poisson's ratio is $\nu = 0.20$. In this case we adopt the Equation (35) for the local strain definition and the damage law from Equation (40), with $\alpha = 0.99$ and $\beta = 400$. The numerical tolerance for convergence is $tol = 10^{-4}$.

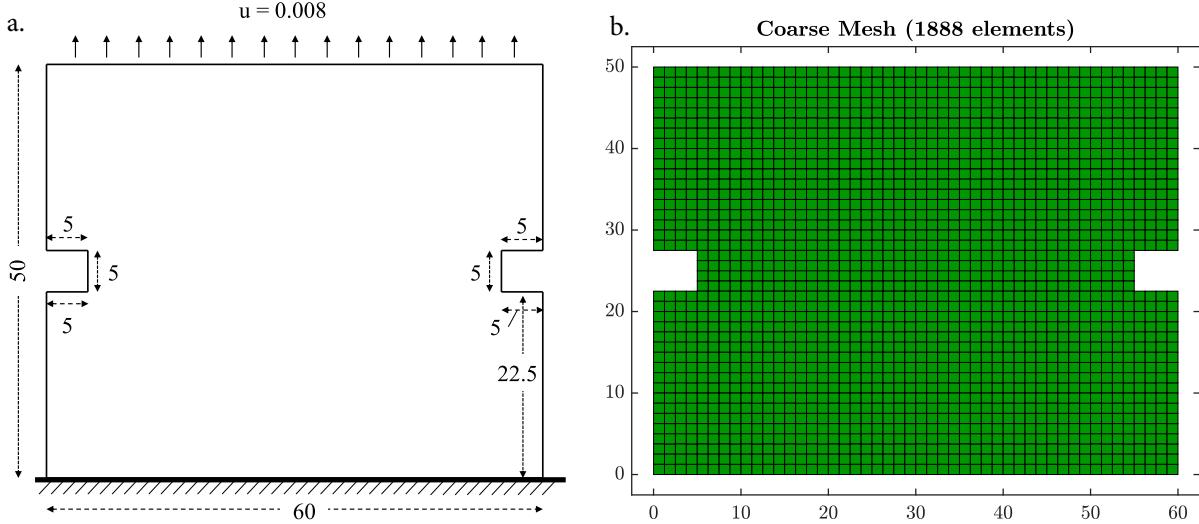


Figure 20: Geometry, loading, boundary conditions and the structured coarse mesh discretization of the double-notch domain.

For this geometry we apply a structured finite element mesh discretization with two mesh densities. The two resulting models are termed *Coarse* and *Fine*, with element sizes of 1.25mm and 0.625mm respectively. The characteristic element length is $l_c = 2$ mm. Figure 20b shows the Coarse model, which employs 1888 elements. Similar to the single-notch problem, we first execute the numerical simulation using the conventional non-local gradient method, and we demonstrate once again the mesh-independence of the solver by comparing the overlapping reaction-loadfactor curves of the two models as shown in Figure 21.

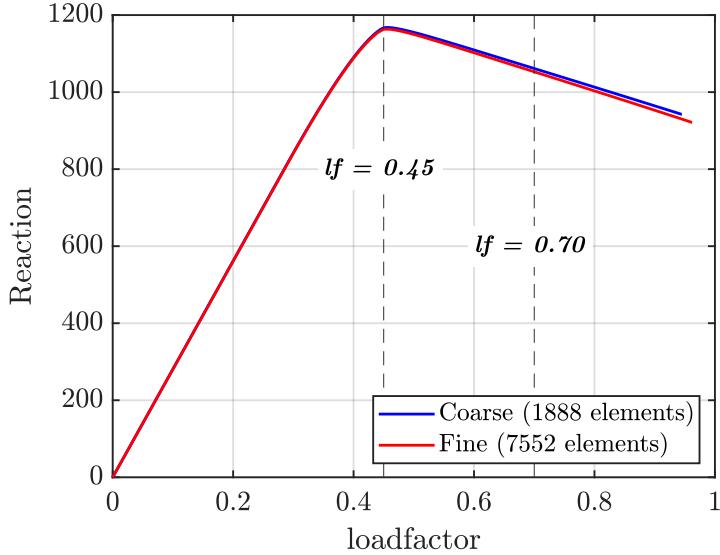


Figure 21: Reaction - loadfactor curves for the two mesh idealizations of the double-notch specimen.

We compare the results of IFENN against the FEM simulations at two load increments. The first case is at $lf = 0.45$, where the reaction curve has reached its peak value and damage has already initiated around the crack tips. The results of this comparison are shown in Figure 22. The first column of plots are the target values generated with FEM, and the second column stems from the I-FENN implementation. The first row of plots depicts the non-local equivalent strains $\bar{\varepsilon}_{eq}$, the second row displays the corresponding damage profile d , and the third row of plots shows the convergence performance of the two methods. Similar to the previous case, the red line in Figures 22e and 22f corresponds to the L2-norm of the residual stress vector \mathbf{R} , and the blue line indicates the L2-norm values of the displacement vector incremental change $\delta\hat{\mathbf{u}}$. Comparison of the non-local strain profiles in 22a and 22b shows the apparent capability of the trained network to identify the presence of more than one region where strain is localized. This is a very important observation which shows the flexibility of the training algorithm, and provides preliminary evidence of the network's capability to adapt to more complex cases than a single crack. The network's ability to identify both regions results to very similar damage d profiles between the two methods, as shown in 22c and 22d. Finally, we emphasize the excellent convergence response of I-FENN shown in 22f, where the norms of both \mathbf{R} and $\delta\hat{\mathbf{u}}$ decrease at an almost constant rate until the convergence criterion of Equation (25) is satisfied.

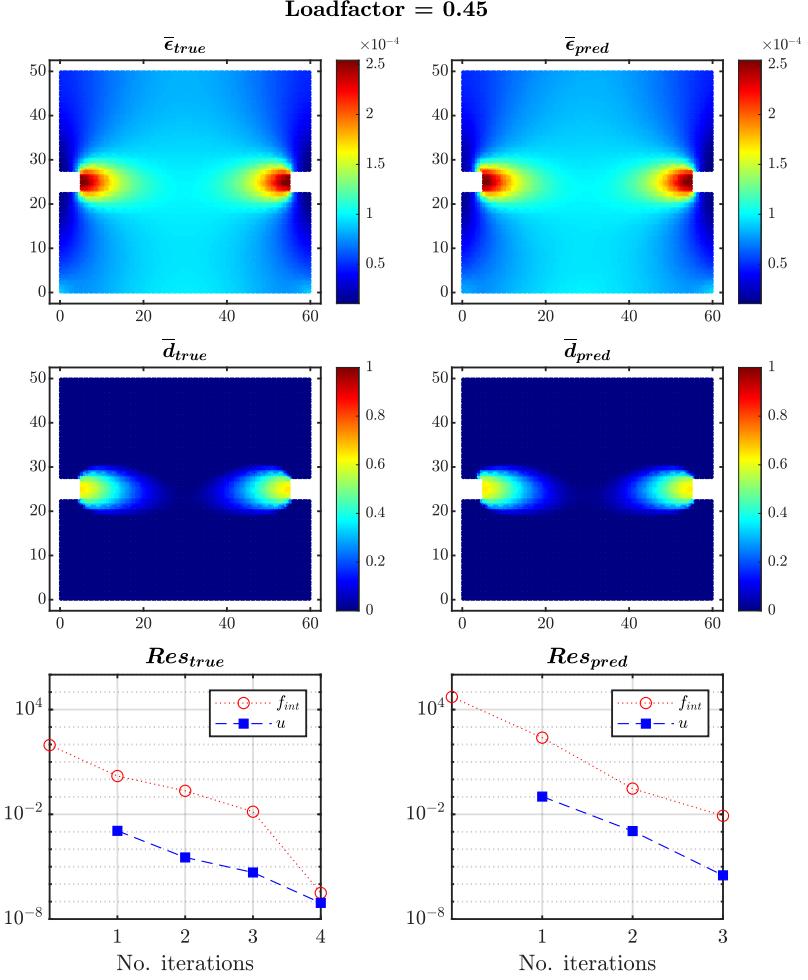


Figure 22: Comparison between FEM (left column) and I-FENN (right column) for the Coarse model of the double-notch case, at a loadfactor $lf = 0.45$. The first row of plots shows the non-local equivalent strain $\bar{\epsilon}_{eq}$ profile, the second row depicts damage d , and the third row reports the calculated residuals. The results demonstrate the capability of I-FENN to identify the presence of more than one strain localization regions, and therefore to detect the presence of multiple crack locations in the domain. The L2-norm of the strain mismatch vector is $\|\bar{\epsilon}_{FEM} - \bar{\epsilon}_{I-FENN}\|_2 = 2.2778 \times 10^{-4}$.

We examine one additional case for the double-notch specimen, at $lf = 0.70$. At this load increment the domain has entered the softening regime and damage is widespread across a much wider zone. Figure 23 depicts the comparative study between FEM and I-FENN, where the subplots follow a similar layout as in Figure 22. These graphs clearly indicate the excellent performance of the proposed methodology, since both the predicted non-local strain and damage profiles resemble with sufficient accuracy the target values. Additionally, the numerical solution converges within a few iterations and the residuals norms follow a monotonically decreased trajectory. Overall, the results of this example establish further confidence in the feasibility of the proposed methodology, and demonstrate its successful implementation in geometries with more than one strain localization regions.

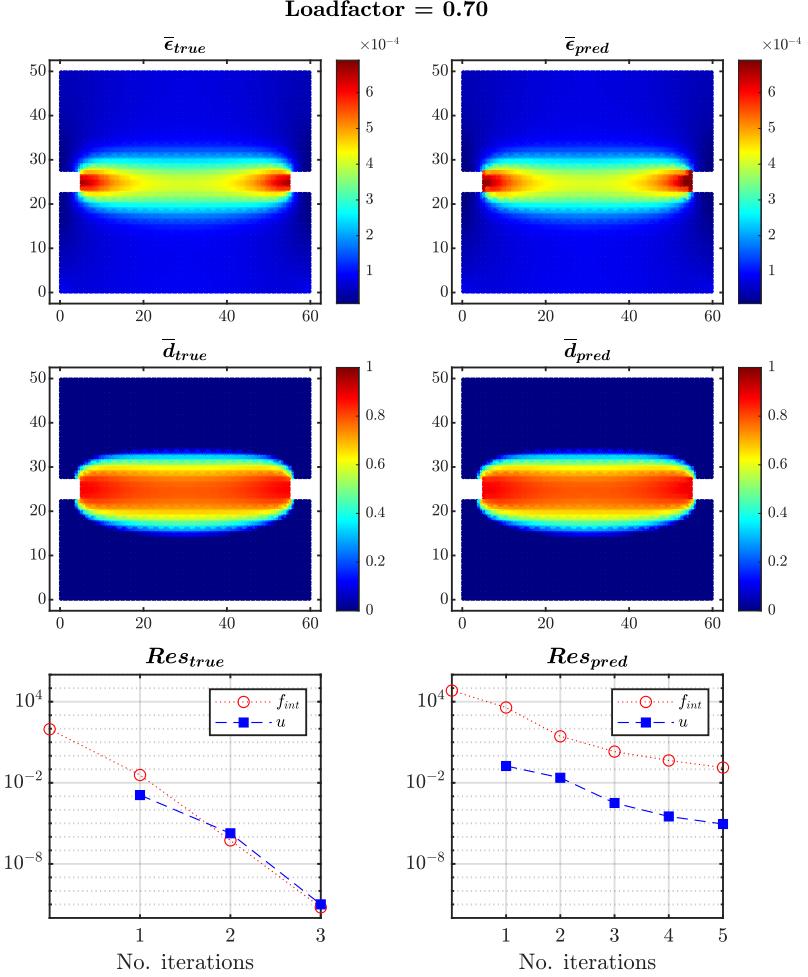


Figure 23: Comparison between FEM (left column) and I-FENN (right column) for the Coarse model of the double-notch case, at a loadfactor $lf = 0.70$. The first row of plots shows the non-local equivalent strain $\bar{\epsilon}_{eq}$ profile, the second row depicts damage d , and the third row reports the calculated residuals. The trained PINN captures with very good accuracy the non-local strain landscape and subsequently I-FENN calculates a damage profile which is very similar to the target one. The L2-norm of the strain differences is $\|\bar{\epsilon}_{FEM} - \bar{\epsilon}_{I-FENN}\|_2 = 8.418 \times 10^{-4}$.

5.3. L-shaped specimen

Thus far, we have validated the proposed methodology in geometries which are discretized using a structured finite element mesh. In this example, which is an L-shaped geometry, we utilize an unstructured mesh and investigate whether this modeling choice has an impact in the solution performance. The geometric and loading/boundary conditions of this example are shown in Figure 24a. The L-shaped specimen is fixed along its left side, and a downward displacement-controlled load is applied across the right edge. The shear modulus is $G = 125000$ KPa and Poisson's ratio is $\nu = 0.20$. For this example we use Equation (35) for the local strain definition and the damage law from Equation (40), with $\alpha = 0.99$ and $\beta = 350$. The numerical tolerance for convergence is set to $tol = 10^{-4}$. Figure 24b shows the resulting finite element discretization, using a mesh of 4100 elements in total. In the fine-mesh zone, the element length is $l_{elem} \approx 2.5\text{mm}$. The characteristic element length is $l_c = 5\text{mm}$. We anticipate damage initiation at the inner corner of the domain, and therefore

we refine the mesh in that region as well as along the expected damage path.

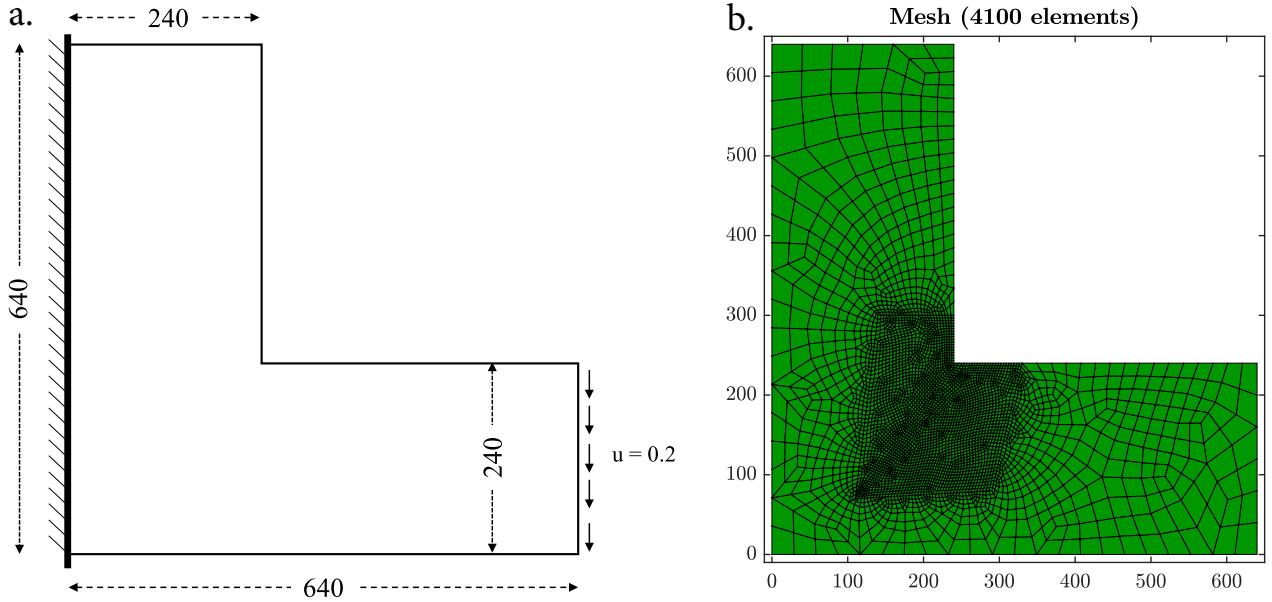


Figure 24: **a.** Geometric and loading details for the L-shaped specimen. **b.** Finite element discretization of the domain using an unstructured mesh of 4100 elements in total.

Figures 25 - 27 present the results of the method implementation in the L-shaped geometry, for loadfactor values $lf = 0.25, 0.50$ and 0.75 . All three cases belong to the inelastic zone, but they represent distinctively different levels of damage spread. The subplots of each figure follow a similar layout to the previous examples, where the FEM results of strain, damage and convergence behavior are shown across the first column of the subplots, and the corresponding results using I-FENN are illustrated across the second column. Overall, similar trends as in the other two numerical examples are observed, and I-FENN is capable of producing results which are almost indistinguishable from FEM. The results alignment is clear across all three load increments and holds true with respect to both variables of interest, namely the non-local strain and damage. Additionally, as shown in the bottom row of plots in these figures, the algorithm converges within a reasonable number of iterations while showing a constantly decreasing trend. The results of this example indicate that the developed framework is insensitive to the choice of mesh orientation, structure and refinement and provide further evidence of the generalizability of I-FENN.

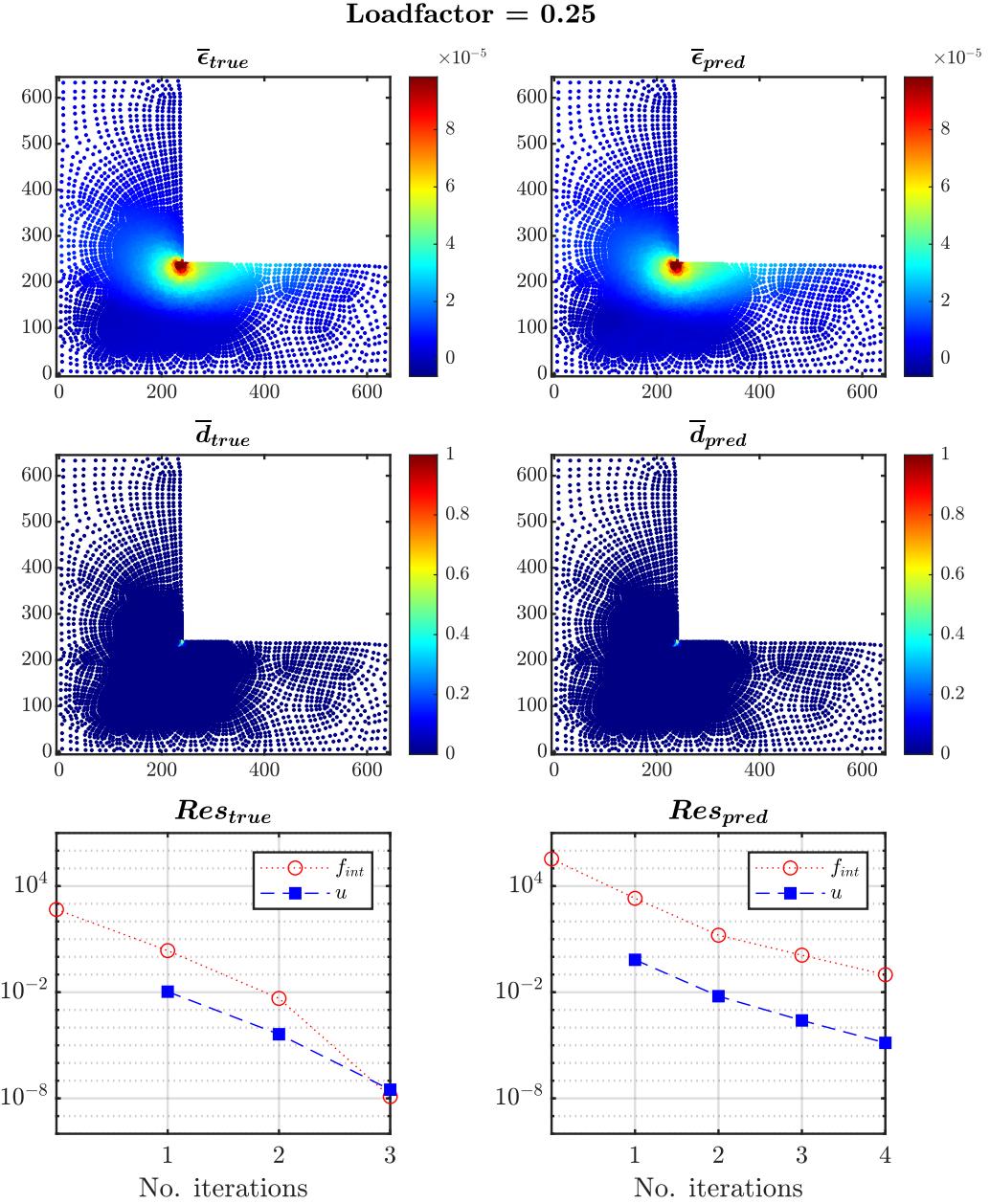


Figure 25: Comparison between FEM (left column) and I-FENN (right column) for the L-shaped specimen at loadfactor $lf = 0.25$. The first row shows the non-local equivalent strain $\bar{\epsilon}_{eq}$, the second row depicts the damage d profile, and the third reports the residuals during this load increment. The L2-norm of the strain differences vector is $\|\bar{\epsilon}_{FEM} - \bar{\epsilon}_{I-FENN}\|_2 = 6.834 \times 10^{-5}$.

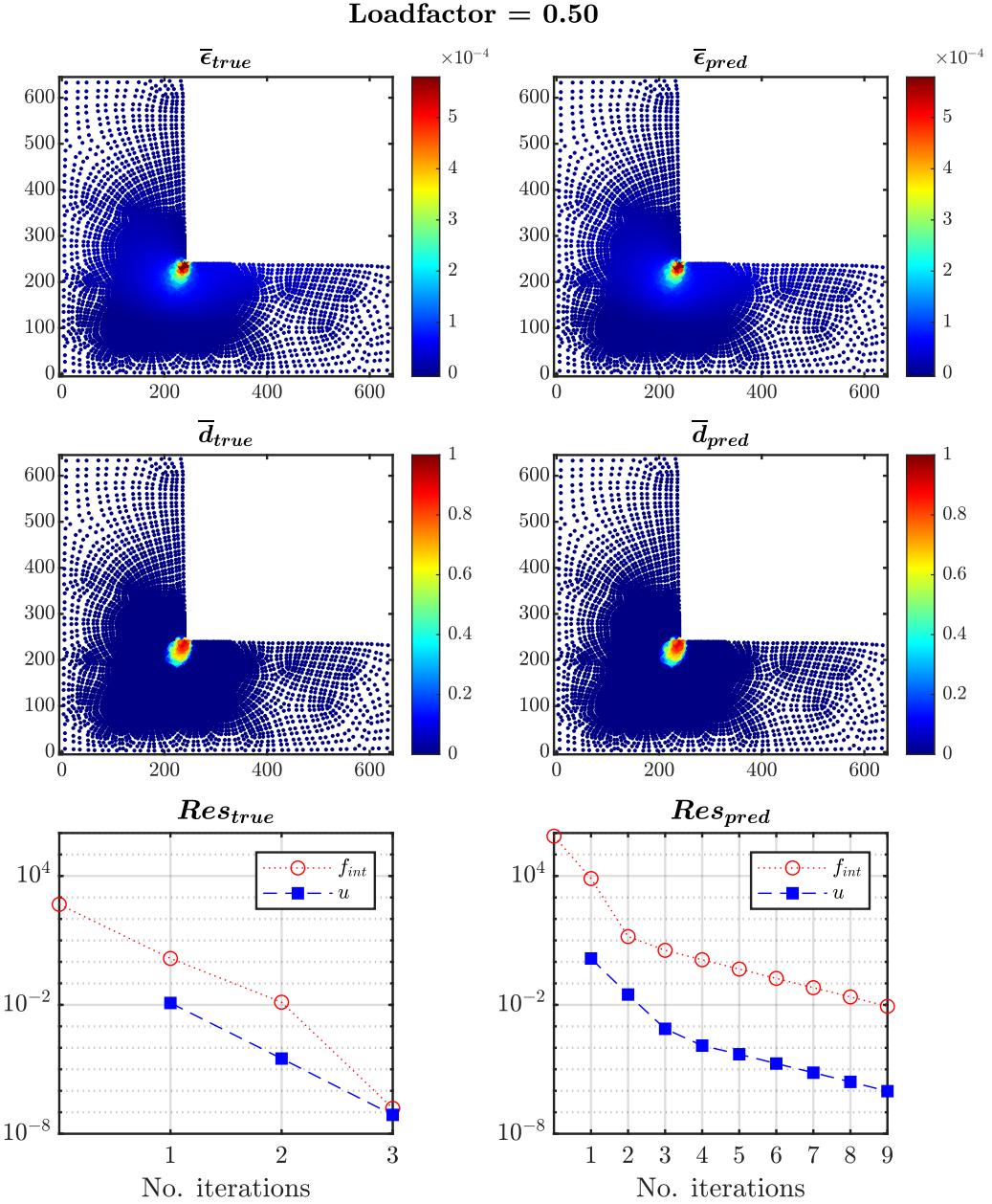


Figure 26: Comparison between FEM (left column) and I-FENN (right column) for the L-shaped specimen at loadfactor $lf = 0.50$. The first row shows the non-local equivalent strain $\bar{\epsilon}_{eq}$, the second row depicts the damage d profile, and the third reports the residuals during this load increment. The L2-norm of the strain mismatch vector is $\|\bar{\epsilon}_{FEM} - \bar{\epsilon}_{I-FENN}\|_2 = 6.797 \times 10^{-4}$.

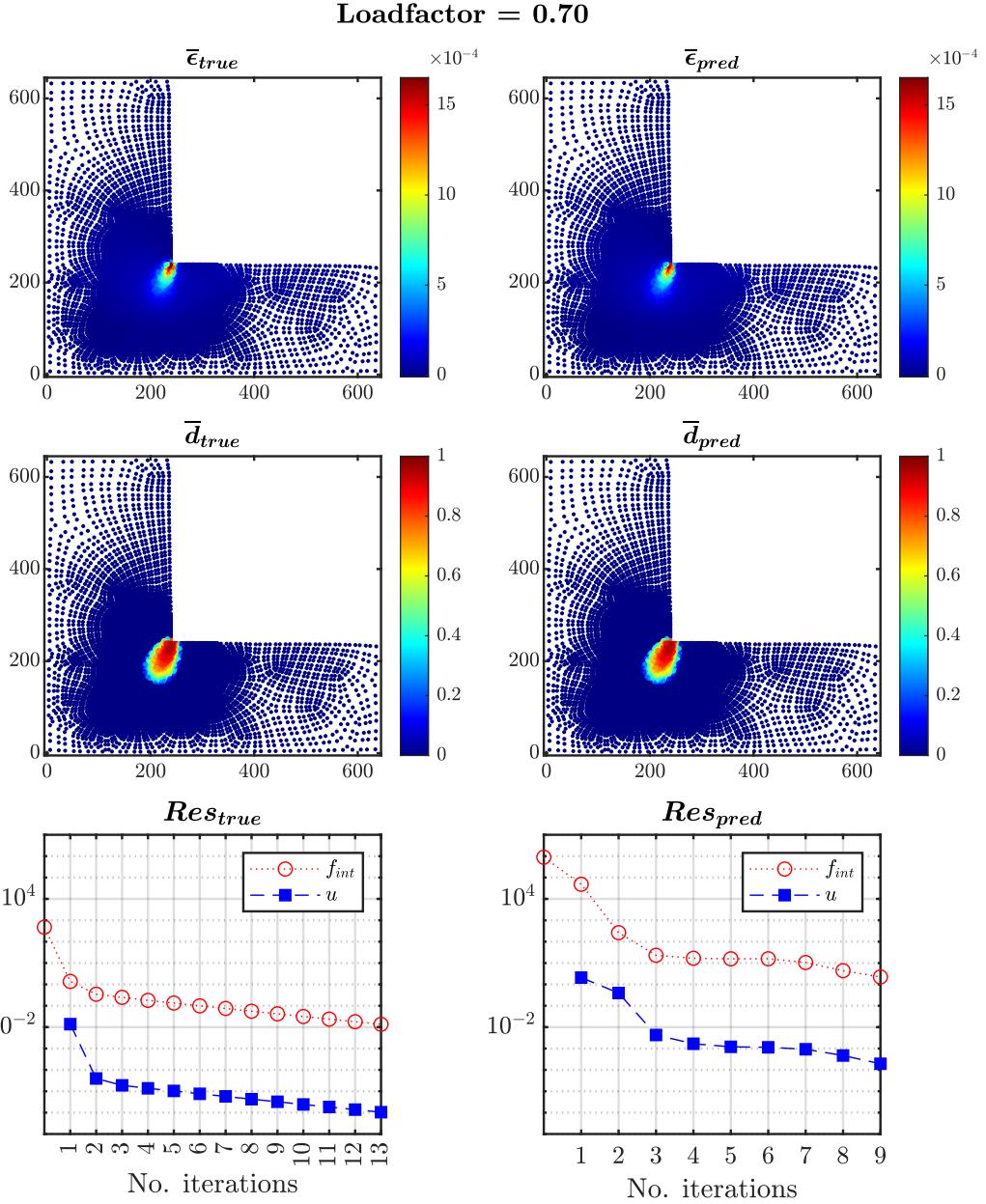


Figure 27: Comparison between FEM (left column) and I-FENN (right column) for the L-shaped specimen at loadfactor $lf = 0.75$. The first row shows the non-local equivalent strain $\bar{\epsilon}_{eq}$, the second row depicts the damage d profile, and the third reports the residuals during this load increment. The L2-norm of the strain error vector is $\|\bar{\epsilon}_{FEM} - \bar{\epsilon}_{I-FENN}\|_2 = 4.441 \times 10^{-3}$.

5.4. Discussion on computational efficiency

To this end, we evaluate the computational efficiency of I-FENN by comparing its simulation runtime vs the conventional FEM solver. We perform this comparison across all the investigated geometries and at the load increments of interest, and we report these values in Table 3. We also note that we utilize a monolithic and not a staggered solver for the FEM, since despite their advantages the latter are well-known for the increased cost and difficulty of capturing the softening regime [93]. We observe that I-FENN outperforms the conventional FEM implementation of the non-local gradient damage model in all but one cases. This case, which corresponds

to the L-shaped geometry at $lf = 0.50$ is dominated by the fact that I-FENN needs 9 iterations to converge, whereas the conventional solver takes 3. In all other cases the performance is significantly improved, with a runtime acceleration ratio between 2-5 times. Since this comparison is conducted only at specific load increments, Table 3 provides just an initial insight on the computational benefit of I-FENN, and we note that the full potential of the proposed framework should be evaluated once the entire load history is analyzed. We also note that in the computational time of I-FENN we do not account for the offline PINN training time. This is because once the PINN is trained, it acts as a standalone nonlinear function which can be used for an arbitrary number of FEM simulations. Therefore, a fair comparison between the two methods should account for just the numerical simulation part, which is a common practice in the literature [26, 94]

Table 3: Comparison of simulation runtimes between I-FENN and FEM. All numbers correspond to seconds.

Single notch				Double notch				L - shaped							
If = 0.42		If = 0.70		If = 0.82		If = 0.45		If = 0.70		If = 0.25		If = 0.50		If = 0.70	
<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM	<i>I - FENN</i>	FEM
4.727	6.821	5.941	20.914	7.523	35.413	2.796	7.786	4.078	5.317	16.784	31.645	37.923	32.766	35.591	131.264

6. Summary and Concluding remarks

In this paper we developed a conceptually new approach of utilizing machine-learning techniques in order to reduce the computational expense of nonlinear computational solid mechanics problems. We introduced an Integrated Finite Element Neural Network framework (I-FENN), where pre-trained neural networks are directly deployed into the finite element stiffness function and act cooperatively with element interpolation functions to compute element-level variables. The trained network is used to compute the state variables, and their derivatives; and hence, it contributes to the calculation of the Jacobian matrix and residual vector utilized in the non-linear FEM analysis. The nonlinear and iterative nature of the solution algorithm is preserved, which allows for the continuous update of the trained network input and output variables until numerical convergence is achieved.

We presented the implementation of the I-FENN framework in the development of a new continuum damage mechanics model. The proposed non-local damage model operates at the lower computational cost of the local damage method, while providing a fully consistent non-local damage description. The key point is the transformation of the local strain to a non-local strain at each material point, a transformation which is operated by the trained neural network. A PINN is developed and trained on the gradient non-local model that is commonly used in non-local damage modeling. The two PINN outputs, i.e. the non-local strain and its derivative with respect to the local, are directly used in the computation of the element Jacobian matrix and residual vector. We showcased through a series of examples the feasibility of the proposed non-local damage model in three distinctly different cases: a) structured mesh with a single crack, b) structured mesh with two cracks, and c) unstructured mesh with a single crack. Through these comparative studies we provided sufficient evidence of the results accuracy and the robustness of numerical convergence. We also illustrated potential generalization pathways of the method, by either a) training the PINN on coarsely meshed geometries and

using it for predictions in finer discretizations, or b) by training the PINN on few selected load increments and applying it against data from unseen time steps.

To the authors' best knowledge, this is the first effort towards a full integration of trained neural network within the framework of the iterative non-linear finite element modeling. The sound implementation and numerical examples show the potential of the I-FENN approach to aid the efforts of improving the computational efficiency of the solution of non-linear problems in mechanics. The deployment of the I-FENN approach within additional mechanics problems may lead to a paradigm shift in the way that machine learning tools are utilized and developed for computational mechanics modeling. The concepts which are proposed in this new benchmark implementation can be tailored and further utilized in many other cases where the need for numerical solution of additional PDEs increases the computational cost to prohibitive levels, such as multi-physics, multi-scale and/or multiple length-scale problems.

As we illustrate the potential of the proposed I-FENN framework, we also mention the outstanding challenges that persist. PINNs are inherently tied to the underlying target problem: the partial differential equation whose solution they approximate. The cost of solving a PDE is notoriously amplified when different scenarios are investigated, such as varying initial and boundary conditions, different domain geometries, different inputs, etc [76]. Consequently, a well-known drawback of PINNs is that once they are trained, they lack adaptiveness to these variations. Our current setup is based on PINNs, and hence it is bounded by its inherent limitations. To this end, machine learning models with a more versatile nature could be even better candidates to approximate the solution of parametric PDEs. This extension could aid to address variations in the geometry, loading setup and boundary conditions, and alleviate the need for repetitive PINN training at each configuration. Additionally, incorporation of the time-dependent behavior of the nonlinear problem requires a more in-depth study of the network design and architecture.

7. Data Availability

The neural network training code, as well as several excel files with training datasets that correspond to the numerical examples, have been made publicly available on Google Drive at:

<https://drive.google.com/drive/u/0/folders/1mM35pXk7gzyx27NbIF3flqQxehr1tebh>

Appendix A. FEM discretization for the local damage method

This appendix presents the mathematical derivation of the numerical solution of the local damage method. Indicial notation is mostly used throughout this section.

Elasticity tensor:

$$C_{ijkl} = \left[K\delta_{ij}\delta_{kl} + \mu \left[\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk} - \frac{2}{3}\delta_{ij}\delta_{kl} \right] \right] \quad (\text{A.1})$$

where K is the bulk modulus, μ is the shear modulus, and δ is the Kronecker delta.

Compatibility equation:

Assuming small deformations, the compatibility equation reads:

$$\varepsilon_{ij} = \frac{1}{2} [u_{i,j} + u_{j,i}] \quad (\text{A.2})$$

Strong form of governing equations:

The equilibrium equation reads:

$$\sigma_{ij,j} = 0 \quad (\text{A.3})$$

Weak form:

The weak form of Equation A.3 reads:

$$\mathbf{R}(u) = \int_{\Omega} w^u \left[[C_{ijkl}(d)\varepsilon_{kl}]_{,j} \right] d\Omega \quad (\text{A.4})$$

where w^u are the displacement field weight functions and $C_{ijkl}(d) = (1-d)C_{ijkl}$. Integration by parts yields:

$$\mathbf{R}(u) = - \int_{\Gamma} [w^u t_i] d\Gamma + \int_{\Omega} w_{,j}^u [C_{ijkl}(d)\varepsilon_{kl}] d\Omega \quad (\text{A.5})$$

where t_i corresponds to the external stress vector applied on the domain boundary.

FEM discretization:

The displacements and strains at the integration points are calculated as follows:

$$u = \mathbf{N}\hat{u}; \quad \varepsilon_{ij} = \mathbf{B}\hat{u} \quad (\text{A.6})$$

$$w^u = \mathbf{N}\hat{w}^u; \quad w_{,j}^u = \mathbf{B}\hat{w}^u \quad (\text{A.7})$$

where \mathbf{N} and \mathbf{B} are the displacement shape function matrix and its derivatives respectively. Nodal values are denoted with the (\cdot) symbol. Substituting equations A.6 and A.7 into A.5 results to:

$$\mathbf{R}(u) = - \int_{\Gamma} [\mathbf{N}\hat{t}_i] d\Gamma + \int_{\Omega} \mathbf{B}^T (1-d) C_{ijkl} \mathbf{B} \hat{u} d\Omega \quad (\text{A.8})$$

In a nonlinear iterative solver, the system of equations to be solved is:

$$\mathbf{J}\delta\hat{u} = -\mathbf{R}(u) \quad (\text{A.9})$$

where \mathbf{J} is the Jacobian matrix and it is calculated as follows:

$$\mathbf{J} = -\frac{\partial \mathbf{R}(u)}{\partial \hat{u}} = \mathbf{K}(u)\mathbf{I} + \frac{\partial \mathbf{K}(u)}{\partial \hat{u}} \hat{u} \quad (\text{A.10})$$

where $\mathbf{I} = \frac{\partial \hat{u}}{\partial \hat{u}}$ is the identity matrix and the expressions for the stiffness matrix \mathbf{K} and its derivative with respect to the nodal displacements are provided below (matrix notation is used for the elasticity tensor):

$$\mathbf{K}(u) = \int_{\Omega} \mathbf{B}^T (1 - \mathbf{d}) \mathbf{C} \mathbf{B}^u d\Omega \quad (\text{A.11})$$

$$\frac{\partial \mathbf{K}(u)}{\partial \hat{u}} = - \int_{\Omega} \mathbf{B} \mathbf{C} \mathbf{B} \frac{\partial \mathbf{d}}{\partial \hat{u}} d\Omega \quad (\text{A.12})$$

The derivative of damage with respect to the nodal displacements at each Gauss point is calculated using the chain rule:

$$\frac{\partial \mathbf{d}}{\partial \hat{u}_k} = \frac{\partial \mathbf{d}}{\partial \varepsilon_{eq}} \frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}} \frac{\partial \varepsilon_{ij}}{\partial \hat{u}_k} \quad (\text{A.13})$$

where the first term is given by the governing damage law, the second term depends on the relationship between the local equivalent strain and the tensor strain, and the third term is the shape function derivatives (Equation A.6).

Appendix B. FEM discretization for the non-local gradient damage method

This appendix presents the mathematical derivation of the numerical solution of the non-local gradient damage method. Indicial notation is mostly used throughout this section.

Elasticity tensor:

$$C_{ijkl} = \left[K\delta_{ij}\delta_{kl} + \mu \left[\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk} - \frac{2}{3}\delta_{ij}\delta_{kl} \right] \right] \quad (\text{B.1})$$

where K is the bulk modulus, μ is the shear modulus, and δ is the Kronecker delta.

Compatibility equation:

Assuming small deformations, the compatibility equation reads:

$$\varepsilon_{ij} = \frac{1}{2} [u_{i,j} + u_{j,i}] \quad (\text{B.2})$$

Strong form of governing equations:

The strong form of the equilibrium equation and the non-local equivalent strain PDE is provided below:

$$\sigma_{ij,j} = 0 \quad (\text{B.3})$$

$$\bar{\varepsilon}_{eq} - g\bar{\varepsilon}_{eq,ii} = \varepsilon_{eq} \quad (\text{B.4})$$

where $\varepsilon_{eq} = \varepsilon_{eq}(\varepsilon_{ij})$ is the local equivalent strain and $\bar{\varepsilon}_{eq}$ is the non-local equivalent strain.

Weak form:

The weak form of Equation B.3 reads:

$$R_i^u = \int_{\Omega} w^u \left[[C_{ijkl}(d)\varepsilon_{kl}]_{,j} \right] d\Omega \quad (\text{B.5})$$

where w^u are the displacement field weight functions and $C_{ijkl}(d) = (1-d)C_{ijkl}$. Upon integration by parts, Equation B.5 yields:

$$R_i^u = - \int_{\Gamma} [w^u t_i] d\Gamma + \int_{\Omega} w_{,j}^u [C_{ijkl}(d)\varepsilon_{kl}] d\Omega \quad (\text{B.6})$$

where t_i corresponds to the external stress vector applied on the domain boundary. The weak form of the gradient equation B.4 is:

$$R^{\varepsilon} = \int_{\Omega} w^{\varepsilon} [\bar{\varepsilon}_{eq} - g\bar{\varepsilon}_{eq,ii} - \varepsilon_{eq}] d\Omega \quad (\text{B.7})$$

where w^{ε} are the non-local strain field weight functions. Upon integration by parts, Equation B.7 yields:

$$R^{\varepsilon} = \int_{\Gamma} [w^{\varepsilon} \bar{\varepsilon}_{eq,i} n_i] d\Gamma - \int_{\Omega} w_{,i}^{\varepsilon} [-g\bar{\varepsilon}_{eq,i}] d\Omega + \int_{\Omega} w^{\varepsilon} [\bar{\varepsilon}_{eq} - \varepsilon_{eq}] d\Omega \quad (\text{B.8})$$

The first term corresponds to the boundary condition expression as given in Equation 8, and therefore vanishes to zero. Upon rearrangement, Equation B.8 reads:

$$R^{\varepsilon} = \int_{\Omega} w_{,i}^{\varepsilon} [g\bar{\varepsilon}_{eq,i}] + w^{\varepsilon} [\bar{\varepsilon}_{eq} - \varepsilon_{eq}] d\Omega \quad (\text{B.9})$$

FEM discretization:

The displacement and non-local strain values at the integration points are calculated based on the following expressions:

$$u = \mathbf{N}^u \hat{u}; \quad \bar{\varepsilon} = \mathbf{N}^\varepsilon \hat{\bar{\varepsilon}} \quad (\text{B.10})$$

$$w^u = \mathbf{N}^u \hat{w}^u; \quad w^\varepsilon = \mathbf{N}^\varepsilon \hat{w}^\varepsilon \quad (\text{B.11})$$

where \mathbf{N}^u and \mathbf{N}^ε are the interpolation functions for displacements and strains respectively, and (\cdot) denotes the corresponding nodal values. As for the local strains and the derivatives of the non-local strains at the integration points, the following equations are used:

$$\varepsilon_{ij} = \mathbf{B}^u \hat{u}; \quad \bar{\varepsilon}_{,j} = \mathbf{B}^\varepsilon \hat{\bar{\varepsilon}} \quad (\text{B.12})$$

$$w_{,j}^u = \mathbf{B}^u \hat{w}^u; \quad w_{,j}^\varepsilon = \mathbf{B}^\varepsilon \hat{w}^\varepsilon \quad (\text{B.13})$$

where \mathbf{B}^u and \mathbf{B}^ε are the shape function derivatives for the displacements and non-local strains respectively.

Using Equations B.11 and B.13, we rewrite Equations B.6 and B.8 as follows:

$$R_i^u = - \int_\Gamma \mathbf{N}^{uT} t_i \, d\Gamma + \int_\Omega \mathbf{B}^{uT} C_{ijkl}(d) \varepsilon_{kl} \, d\Omega \quad (\text{B.14})$$

$$R_i^\varepsilon = \int_\Omega \mathbf{B}^{\varepsilon T} g \bar{\varepsilon}_{eq,i} + \mathbf{N}^{\varepsilon T} \bar{\varepsilon}_{eq} - \mathbf{N}^{\varepsilon T} \varepsilon_{eq} \, d\Omega \quad (\text{B.15})$$

Differentiating the residuals with respect to the displacement and non-local strain fields, results to the following terms for the Jacobian matrix components (matrix notation is used for the elasticity tensor):

$$\mathbf{J}^{uu} = \frac{\partial R^u}{\partial u} = \int_\Omega \mathbf{B}^{uT} (1-d) \mathbf{C} \mathbf{B}^u \, d\Omega \quad (\text{B.16})$$

$$\mathbf{J}^{u\varepsilon} = \frac{\partial R^u}{\partial \bar{\varepsilon}} = - \int_\Omega \mathbf{B}^{uT} \mathbf{C} \frac{\partial \mathbf{d}}{\partial \bar{\varepsilon}} \varepsilon_{kl} \mathbf{N}^\varepsilon \, d\Omega \quad (\text{B.17})$$

$$\mathbf{J}^{\varepsilon u} = \frac{\partial R^\varepsilon}{\partial u} = - \int_\Omega \mathbf{N}^{\varepsilon T} \frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}} \mathbf{B}^u \, d\Omega \quad (\text{B.18})$$

$$\mathbf{J}^{\varepsilon\varepsilon} = \frac{\partial R^\varepsilon}{\partial \bar{\varepsilon}} = \int_\Omega (\mathbf{N}^{\varepsilon T} \mathbf{N}^\varepsilon + \mathbf{B}^{\varepsilon T} g \mathbf{B}^\varepsilon) \, d\Omega \quad (\text{B.19})$$

The discretization of the nodal internal and external forces reads:

$$\hat{\mathbf{f}}_{int}^u = \int_\Omega \mathbf{B}^{uT} \hat{\sigma} \, d\Omega \quad (\text{B.20})$$

$$\hat{\mathbf{f}}_{ext}^u = \int_\Gamma \mathbf{N}^{uT} \hat{t} \, d\Omega \quad (\text{B.21})$$

$$\hat{\mathbf{f}}_{int}^{\bar{\varepsilon}} = \mathbf{J}^{\bar{\varepsilon}\bar{\varepsilon}} \hat{\bar{\varepsilon}}_{eq} \quad (\text{B.22})$$

$$\hat{\mathbf{f}}_{ext}^{\bar{\varepsilon}} = \int_\Omega \mathbf{N}^{\bar{\varepsilon}T} \varepsilon_{eq} \, d\Omega \quad (\text{B.23})$$

where \hat{t} are the nodal boundary forces. Ultimately, the matrix equation system for the non-local gradient damage method reads:

$$\begin{bmatrix} \mathbf{J}^{uu} & \mathbf{J}^{u\bar{\varepsilon}} \\ \mathbf{J}^{\bar{\varepsilon}u} & \mathbf{J}^{\bar{\varepsilon}\bar{\varepsilon}} \end{bmatrix} \begin{bmatrix} \delta \hat{\mathbf{u}} \\ \delta \hat{\bar{\varepsilon}}_{eq} \end{bmatrix} = - \begin{bmatrix} \hat{\mathbf{f}}_{int}^u \\ \hat{\mathbf{f}}_{int}^{\bar{\varepsilon}} \end{bmatrix} + \begin{bmatrix} \hat{\mathbf{f}}_{ext}^u \\ \hat{\mathbf{f}}_{ext}^{\bar{\varepsilon}} \end{bmatrix} \quad (\text{B.24})$$

Appendix C. Appendix C: Newton-Raphson algorithm

The algorithm below presents the implementation details of the nonlinear iterative Newton-Raphson solver. We note that $\delta\hat{\mathbf{u}}_F$ denotes the incremental change of the nodal displacements at the free (Neumann) boundary. Also, the *tolerance* is a sufficiently small number dictating numerical convergence and it is specified in the order of 10^{-4} to 10^{-6} .

Algorithm 2: Newton-Raphson algorithm

```

1 Initialize all variables, set flag as true
2 while loadfactor < 1 do
3   iter = 1
4   Evaluate global Jacobian matrix  $\mathbf{J}$ , residual forces vector  $\mathbf{R}$  and history variable matrix
5   while flag is true do
6     Solve  $\mathbf{J}\delta\hat{\mathbf{u}} = -\mathbf{R}$ ; update  $\hat{\mathbf{u}}$ 
7     Evaluate global Jacobian  $\mathbf{J}$ , residual forces vector  $\mathbf{R}$  and history variable matrix
8     if  $\frac{\|\delta\hat{\mathbf{u}}_{F,iter}\|_2}{\|\delta\hat{\mathbf{u}}_{F,1}\|_2} < tolerance$  and iter < max_iter then
9       Update history variable; calculate reactions; increase loadfactor; set flag as false
10      else
11        Discard the updated history variable matrix; iter = iter + 1
12        if iter = max_iter then
13          Set flag as false; decrease the applied load increment
14      end
15    end
16 end

```

Appendix D. Appendix D: Element Jacobian formulation for the local and non-local gradient damage methods

The following algorithms describe the element Jacobian formulation \mathbf{J}_{elem} for the conventional local and non-local gradient methods:

Algorithm 3: Element Jacobian formulation for the local damage method

```

1 Initialize variables
2 for each finite element do
3   Compute number, positions and weights of integration points
4   for each integration point do
5     Compute the shape functions  $\mathbf{N}$  and their derivatives  $\mathbf{B}$ 
6     Calculate the local strain vector  $\boldsymbol{\epsilon} = \mathbf{B} \times \mathbf{d}$ 
7     Compute the local equivalent strain  $\varepsilon_{eq}$  and its derivative w.r.t. the strain vector  $\frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}}$ 
8     Calculate the local damage variable  $D$  and keep  $\max(D, D_{stored})$ 
9     Evaluate the stress vector  $\boldsymbol{\sigma} = (1 - D) \times \mathbf{C} \times \boldsymbol{\epsilon}$ 
10    Compute the point contribution to the  $\mathbf{R}_{elem} = \mathbf{R}_{elem} + \mathbf{B} \times \boldsymbol{\sigma}$ 
11    Compute the point contribution to the  $\mathbf{J}_{elem}$  based on Equations A.11, A.12 and A.13.
12  end
13 end

```

Algorithm 4: Element Jacobian formulation for the non-local gradient damage method

```

1 Initialize variables
2 for each finite element do
3   Identify the number, positions and weights of integration points
4   for each integration point do
5     Compute the shape functions  $\mathbf{N}$ ,  $\bar{\mathbf{N}}$  and their derivatives  $\mathbf{B}$ ,  $\bar{\mathbf{B}}$  for  $\mathbf{d}$  and  $\bar{\boldsymbol{\epsilon}}$  respectively
6     Calculate the local strain vector  $\boldsymbol{\epsilon} = \mathbf{B} \times \mathbf{d}$ 
7     Calculate the local equivalent strain  $\varepsilon_{eq}$  and its derivative w.r.t. the strain vector  $\frac{\partial \varepsilon_{eq}}{\partial \varepsilon_{ij}}$ 
8     Compute the non-local equivalent strain  $\bar{\varepsilon} = \bar{\mathbf{N}} \times \hat{\boldsymbol{\epsilon}}$  and history variable
9     Evaluate the stress vector  $\boldsymbol{\sigma} = (1 - \bar{D}) \times \mathbf{C} \times \boldsymbol{\epsilon}$ 
10    Compute the point contribution to the  $\mathbf{R}_{elem} = \mathbf{R}_{elem} + \mathbf{B} \times \boldsymbol{\sigma}$ 
11    Compute the point contribution to  $\mathbf{J}_{elem}$  based on Equations B.16, B.17, B.18 and B.19.
12  end
13 end

```

References

- [1] J. N. Reddy, An Introduction to Nonlinear Finite Element Analysis Second Edition: with applications to heat transfer, fluid mechanics, and solid mechanics, OUP Oxford, 2014.
- [2] T. Belytschko, W. K. Liu, B. Moran, K. Elkhodary, Nonlinear finite elements for continua and structures, John wiley & sons, 2014.
- [3] P. Wriggers, Nonlinear finite element methods, Springer Science & Business Media, 2008.
- [4] C. McAuliffe, H. Waisman, A coupled phase field shear band model for ductile–brittle transition in notched plate impacts, Computer Methods in Applied Mechanics and Engineering 305 (2016) 173–195.
- [5] F. P. Duda, A. Ciarbonetti, P. J. Sánchez, A. E. Huespe, A phase-field/gradient damage model for brittle fracture in elastic–plastic solids, International Journal of Plasticity 65 (2015) 269–296.
- [6] C. A. Duarte, O. Hamzeh, T. Liszka, W. Tworzydlo, A generalized finite element method for the simulation of three-dimensional dynamic crack propagation, Computer methods in applied mechanics and engineering 190 (15-17) (2001) 2227–2262.
- [7] F. Barlat, D. J. Lege, J. C. Brem, A six-component yield function for anisotropic materials, International Journal of Plasticity 7 (7) (1991) 693–712.
- [8] F. Barlat, J. J. Gracio, M.-G. Lee, E. F. Rauch, G. Vincze, An alternative to kinematic hardening in classical plasticity, International Journal of Plasticity 27 (9) (2011) 1309–1327.
- [9] M. E. Mobasher, L. Berger-Vergiat, H. Waisman, Non-local formulation for transport and damage in porous media, Computer Methods in Applied Mechanics and Engineering 324 (2017) 654–688.
- [10] M. E. Mobasher, H. Waisman, L. Berger-Vergiat, Thermodynamic framework for non-local transport-damage modeling of fluid driven fracture in porous media, International Journal of Rock Mechanics and Mining Sciences 111 (2018) 64–83.
- [11] D. Ozturk, S. Kotha, S. Ghosh, An uncertainty quantification framework for multiscale parametrically homogenized constitutive models (phcms) of polycrystalline ti alloys, Journal of the Mechanics and Physics of Solids 148 (2021) 104294.
- [12] M. E. Mobasher, H. Waisman, Dual length scale non-local model to represent damage and transport in porous media, Computer Methods in Applied Mechanics and Engineering 387 (2021) 114154.
- [13] P. Pantidis, S. Gerasimidis, Progressive collapse of 3d steel composite buildings under interior gravity column loss, Journal of Constructional Steel Research 150 (2018) 60–75.
- [14] Y. Zhang, C. Bajaj, Finite element meshing for cardiac analysis (08 2004).

- [15] C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, D. Rixen, Feti-dp: a dual–primal unified feti method—part i: A faster alternative to the two-level feti method, *International journal for numerical methods in engineering* 50 (7) (2001) 1523–1544.
- [16] M. E. Mobasher, H. Waisman, Adaptive modeling of damage growth using a coupled fem/bem approach, *International Journal for Numerical Methods in Engineering* 105 (8) (2016) 599–619.
- [17] J. A. White, N. Castelletto, S. Klevtsov, Q. M. Bui, D. Osei-Kuffuor, H. A. Tchelepi, A two-stage preconditioner for multiphase poromechanics in reservoir simulation, *Computer Methods in Applied Mechanics and Engineering* 357 (2019) 112575.
- [18] N. Castelletto, J. A. White, H. Tchelepi, Accuracy and convergence properties of the fixed-stress iterative solution of two-way coupled poromechanics, *International Journal for Numerical and Analytical Methods in Geomechanics* 39 (14) (2015) 1593–1618.
- [19] H. Waisman, L. Berger-Vergiat, An adaptive domain decomposition preconditioner for crack propagation problems modeled by xfem, *International Journal for Multiscale Computational Engineering* 11 (6) (2013).
- [20] J. Fish, T. Jiang, Z. Yuan, A staggered nonlocal multiscale model for a heterogeneous medium, *International journal for numerical methods in engineering* 91 (2) (2012) 142–157.
- [21] T. Tancogne-Dejean, M. B. Gorji, J. Zhu, D. Mohr, Recurrent neural network modeling of the large deformation of lithium-ion battery cells, *International Journal of Plasticity* 146 (2021) 103072.
- [22] H. J. Logarzo, G. Capuano, J. J. Rimoli, Smart constitutive laws: Inelastic homogenization through machine learning, *Computer Methods in Applied Mechanics and Engineering* 373 (2021) 113482.
- [23] A. Fascetti, C. Oskay, Multiscale modeling of backward erosion piping in flood protection system infrastructure, *Computer-Aided Civil and Infrastructure Engineering* 34 (12) (2019) 1071–1086.
- [24] D. Reimann, K. Nidadavolu, H. ul Hassan, N. Vajragupta, T. Glasmachers, P. Junker, A. Hartmaier, Modeling macroscopic material behavior with machine learning algorithms trained by micromechanical simulations, *Frontiers in Materials* 6 (2019).
- [25] N. N. Vlassis, R. Ma, W. Sun, Geometric deep learning for computational mechanics part i: anisotropic hyperelasticity, *Computer Methods in Applied Mechanics and Engineering* 371 (2020) 113299.
- [26] M. Mozaffar, R. Bostanabad, W. Chen, K. Ehmann, J. Cao, M. A. Bessa, Deep learning predicts path-dependent plasticity, *Proceedings of the National Academy of Sciences* 116 (52) (2019) 26414–26420.
- [27] D. Huang, J. N. Fuhr, C. Weißenfels, P. Wriggers, A machine learning based plasticity model using proper orthogonal decomposition, *Computer Methods in Applied Mechanics and Engineering* 365 (2020) 113008.
- [28] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, *Computer Methods in Applied Mechanics and Engineering* 304 (2016) 81–101.

- [29] F. Chinesta, P. Ladeveze, R. Ibanez, J. V. Aguado, E. Abisset-Chavanne, E. Cueto, Data-driven computational plasticity, *Procedia Engineering* 207 (2017) 209–214, international Conference on the Technology of Plasticity, ICTP 2017, 17-22 September 2017, Cambridge, United Kingdom.
- [30] P. Carrara, L. De Lorenzis, L. Stainier, M. Ortiz, Data-driven fracture mechanics, *Computer Methods in Applied Mechanics and Engineering* 372 (2020) 113390.
- [31] C. Rao, H. Sun, Y. Liu, Physics informed deep learning for computational elastodynamics without labeled data, *arXiv preprint arXiv:2006.08472* (2020).
- [32] S. Kumar, D. Kochmann, What Machine Learning Can Do for Computational Solid Mechanics, 2022, pp. 275–285.
- [33] W. Li, M. Bazant, J. Zhu, A physics-guided neural network framework for elastic plates: Comparison of governing equations-based and energy-based approaches, *Computer Methods in Applied Mechanics and Engineering* 383 (2021) 113933.
- [34] M. Abdulla, R. L. Sousa, I. Arzuaga, O. AlDajani, B. G. da Silva, H. H. Einstein, Fracprop: stochastic fracture propagation model, *Rock Mechanics and Rock Engineering* 54 (5) (2021) 2513–2531.
- [35] C.-T. Chen, G. X. Gu, Learning hidden elasticity with deep neural networks, *Proceedings of the National Academy of Sciences* 118 (31) (2021) e2102721118.
- [36] F. Masi, I. Stefanou, Multiscale modeling of inelastic materials with thermodynamics-based artificial neural networks (tann), *Computer Methods in Applied Mechanics and Engineering* 398 (2022) 115190.
- [37] B. Yu, et al., The deep ritz method: a deep learning-based numerical algorithm for solving variational problems, *Communications in Mathematics and Statistics* 6 (1) (2018) 1–12.
- [38] D. W. Abueidda, S. Koric, R. A. Al-Rub, C. M. Parrott, K. A. James, N. A. Sobh, A deep learning energy method for hyperelasticity and viscoelasticity, *European Journal of Mechanics, A/Solids* 95 (2022).
- [39] E. Kharazmi, Z. Zhang, G. E. Karniadakis, hp-vpinns: Variational physics-informed neural networks with domain decomposition, *Computer Methods in Applied Mechanics and Engineering* 374 (2021) 113547.
- [40] E. Samaniego, C. Anitescu, S. Goswami, V. M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, *Computer Methods in Applied Mechanics and Engineering* 362 (2020) 112790.
- [41] J. N. Fuhr, N. Bouklas, The mixed deep energy method for resolving concentration features in finite strain hyperelasticity, *Journal of Computational Physics* 451 (2022).
- [42] M. Jokar, F. Semperlotti, Finite element network analysis: A machine learning based computational framework for the simulation of physical systems, *Computers & Structures* 247 (2021) 106484.

- [43] M. Jokar, F. Semperlotti, Two-dimensional finite element network analysis: Formulation and static analysis of structural assemblies, *Computers & Structures* 266 (2022) 106784.
- [44] S. K. Mitusch, S. W. Funke, M. Kuchta, Hybrid fem-nn models: Combining artificial neural networks with the finite element method, *Journal of Computational Physics* 446 (2021) 110651.
- [45] D. C. Psichogios, L. H. Ungar, A hybrid neural network-first principles approach to process modeling, *AIChE Journal* 38 (10) (1992) 1499–1511.
- [46] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, *IEEE transactions on neural networks* 9 (5) (1998) 987–1000.
- [47] M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, *arXiv preprint arXiv:1711.10561* (2017).
- [48] 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, *Journal of Computational physics* 378 (2019) 686–707.
- [49] P. Stiller, F. Bethke, M. Böhme, R. Pausch, S. Torge, A. Debus, J. Vorberger, M. Bussmann, N. Hoffmann, Large-scale neural solvers for partial differential equations, in: *Smoky Mountains Computational Sciences and Engineering Conference*, Springer, 2020, pp. 20–34.
- [50] A. Mathews, M. Francisquez, J. W. Hughes, D. R. Hatch, B. Zhu, B. N. Rogers, Uncovering turbulent plasma dynamics via deep learning from partial observations, *Physical Review E* 104 (2) (2021) 025205.
- [51] L. Yang, D. Zhang, G. E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, *SIAM Journal on Scientific Computing* 42 (1) (2020) A292–A317.
- [52] Z. Fang, A high-efficient hybrid physics-informed neural networks based on convolutional neural network, *IEEE Transactions on Neural Networks and Learning Systems* (2021).
- [53] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, *Computer Methods in Applied Mechanics and Engineering* 379 (2021) 113741.
- [54] A. Henkes, H. Wessels, R. Mahnken, Physics informed neural networks for continuum micromechanics, *Computer Methods in Applied Mechanics and Engineering* 393 (2022) 114790.
- [55] R. H. Peerlings, R. de Borst, W. M. Brekelmans, J. de Vree, Gradient enhanced damage for quasi-brittle materials, *International Journal for numerical methods in engineering* 39 (19) (1996) 3391–3403.
- [56] G. Pijaudier-Cabot, Z. P. Bažant, Nonlocal damage theory, *Journal of engineering mechanics* 113 (10) (1987) 1512–1533.

- [57] J. De Vree, W. Brekelmans, M. van Gils, Comparison of nonlocal approaches in continuum damage mechanics, *Computers & Structures* 55 (4) (1995) 581–588.
- [58] R. Peerlings, M. Geers, R. De Borst, W. Brekelmans, A critical comparison of nonlocal and gradient-enhanced softening continua, *International Journal of Solids and Structures* 38 (44-45) (2001) 7723–7746.
- [59] B. Kiefer, T. Waffenschmidt, L. Sprave, A. Menzel, A gradient-enhanced damage model coupled to plasticity—multi-surface formulation and algorithmic concepts, *International Journal of Damage Mechanics* 27 (2) (2018) 253–295.
- [60] J. Lemaitre, R. Desmorat, *Engineering Damage Mechanics. Ductile, Creep, Fatigue and Brittle Failure*, 2005.
- [61] L. Kachanov, *Introduction to continuum damage mechanics*, Mechanics of Elastic Stability, Springer Netherlands, 2013.
- [62] A. A. Griffith, G. I. Taylor, The phenomena of rupture and flow in solids, *Philosophical Transactions of the Royal Society of London* 221 (1920) 163–198.
- [63] M. Geers, R. de Borst, W. Brekelmans, R. Peerlings, Strain-based transient-gradient damage model for failure analyses, *Computer Methods in Applied Mechanics and Engineering* 160 (1) (1998) 133–153.
- [64] N. Moës, C. Stolz, P.-E. Bernard, N. Chevaugeon, A level set based model for damage growth: the thick level set approach, *International Journal for Numerical Methods in Engineering* 86 (3) (2011) 358–380.
- [65] N. Chevaugeon, N. Moës, Lipschitz regularization for fracture: the lip-field approach, CoRR abs/2111.04771 (2021). [arXiv:2111.04771](https://arxiv.org/abs/2111.04771).
- [66] A. Parrilla Gómez, C. Stolz, N. Moës, D. Grégoire, G. Pijaudier-Cabot, On the capability of the thick level set (tls) damage model to fit experimental data of size and shape effects, *Engineering Fracture Mechanics* 184 (2017) 75–87.
- [67] N. Moës, N. Chevaugeon, Lipschitz regularization for softening material models: the Lip-field approach, *Comptes Rendus. Mécanique* 349 (2) (2021) 415–434.
- [68] J.-Y. Wu, V. P. Nguyen, C. T. Nguyen, D. Sutula, S. Sinaie, S. P. Bordas, Chapter one - phase-field modeling of fracture, Vol. 53 of *Advances in Applied Mechanics*, Elsevier, 2020, pp. 1–183.
- [69] R. de Borst, C. V. Verhoosel, Gradient damage vs phase-field approaches for fracture: Similarities and differences, *Computer Methods in Applied Mechanics and Engineering* 312 (2016) 78–94, phase Field Approaches to Fracture.
- [70] M. Jirásek, B. Patzák, Consistent tangent stiffness for nonlocal damage models, *Computers & structures* 80 (14-15) (2002) 1279–1293.

- [71] Y. Chen, M. E. Mobasher, H. Waisman, Dynamic soil consolidation model using a nonlocal continuum poroelastic damage approach, *International Journal for Numerical and Analytical Methods in Geomechanics* 46 (3) (2022) 486–528.
- [72] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, F. Piccialli, Scientific machine learning through physics-informed neural networks: Where we are and what's next, arXiv preprint arXiv:2201.05624 (2022).
- [73] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: *Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings*, 2010, pp. 249–256.
- [74] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [75] S. Markidis, The old and the new: Can physics-informed deep-learning replace traditional linear solvers?, *Frontiers in big Data* (2021) 92.
- [76] S. Wang, H. Wang, P. Perdikaris, Learning the solution operator of parametric partial differential equations with physics-informed deepnets, *Science advances* 7 (40) (2021) eabi8605.
- [77] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, *nature* 521 (7553) (2015) 436–444.
- [78] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization., *Journal of machine learning research* 12 (7) (2011).
- [79] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).
- [80] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning representations by back-propagating errors, *nature* 323 (6088) (1986) 533–536.
- [81] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: a survey, *Journal of Machine Learning Research* 18 (2018) 1–43.
- [82] H. Gao, L. Sun, J.-X. Wang, Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain, *Journal of Computational Physics* 428 (2021) 110079.
- [83] S. Mishra, R. Molinaro, Estimates on the generalization error of physics informed neural networks (pinns) for approximating pdes., Tech. Rep. 2020-45, Seminar for Applied Mathematics, ETH Zürich, Switzerland (2020).
- [84] K. O. Lye, S. Mishra, D. Ray, Deep learning observables in computational fluid dynamics, *Journal of Computational Physics* 410 (2020) 109339.

- [85] M. Yin, E. Zhang, Y. Yu, G. E. Karniadakis, Interfacing finite elements with deep neural operators for fast multiscale modeling of mechanics problems, Computer Methods in Applied Mechanics and Engineering (2022) 115027.
- [86] S. Bhanja, A. Das, Impact of data normalization on deep neural network for time series forecasting, CoRR abs/1812.05519 (2018).
- [87] B. Moseley, A. Markham, T. Nissen-Meyer, Finite basis physics-informed neural networks (fbpinns): a scalable domain decomposition approach for solving differential equations, arXiv preprint arXiv:2107.07871 (2021).
- [88] Y. Wang, D. Oyen, W. G. Guo, A. Mehta, C. B. Scott, N. Panda, M. G. Fernández-Godino, G. Srinivasan, X. Yue, Stressnet-deep learning to predict stress with fracture propagation in brittle materials, npj Materials Degradation 5 (1) (2021) 1–10.
- [89] R. H. Peerlings, R. de Borst, W. Brekelmans, M. G. Geers, Gradient-enhanced damage modelling of concrete fracture, Mechanics of Cohesive-frictional Materials: An International Journal on Experiments, Modelling and Computation of Materials and Structures 3 (4) (1998) 323–342.
- [90] J. Mazars, A description of micro-and macroscale damage of concrete structures, Engineering Fracture Mechanics 25 (5-6) (1986) 729–737.
- [91] R. de Borst, C. V. Verhoosel, Gradient damage vs phase-field approaches for fracture: Similarities and differences, Computer Methods in Applied Mechanics and Engineering 312 (2016) 78–94.
- [92] N. Triantafyllidis, E. Aifantis, A gradient approach to localization of deformation. i. hyperelastic materials, Journal of Elasticity 16 (1986).
- [93] R. Bharali, S. Goswami, C. Anitescu, T. Rabczuk, A robust monolithic solver for phase-field fracture integrated with fracture energy based arc-length method and under-relaxation, Computer Methods in Applied Mechanics and Engineering 394 (2022) 114927.
- [94] L. Liang, M. Liu, C. Martin, W. Sun, A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis, Journal of The Royal Society Interface 15 (138) (2018) 20170844.