POLITECNICO DI TORINO

Master's Degree in Mechanical Engineering

Master Degree Thesis

Variational Machine Learning Method for Simulation of Fracture Mechanics

Supervisor

Prof. Aurelio Somà

Co-supervisors

Prof. Dr.-Ing. Sandra Klinge

Francesca Pistorio

Stefan Hildebrand

Candidate

Martina Toncelli

Academic Year 2024-2025

Acknowledgements

I would like to express my special gratitude to my Italian supervisors, Prof. Aurelio Somà and Francesca Pistorio, for their support and guidance throughout the development of my thesis over these past months.
I am also truly grateful to my German supervisors, Prof. Dr.-Ing. Sandra Klinge and Stefan Hildebrand, for hosting me at the Technical University of Berlin and for helping me tackle new topics and challenges.

A heartfelt thanks goes to my friends, who have supported me in reaching this important goal and have stood by my side during every moment of difficulty throughout this journey. I would also like to thank them in advance for the help they will continue to offer me in the future.

Finally, a big thank you goes to my entire family, who has always encouraged me to give my best and believed in me, supporting me in every decision. I am truly grateful for all the opportunities and experiences you have given me.

Abstract

In the last few decades, Machine Learning has experienced rapid growth thanks to the development of hardware technologies, and the improvement of optimization algorithms, and computational capabilities. It found space in a variety of businesses, such as in the engineering field, where several neural network architectures can be employed, enabling the processing of a huge amount of data and solving complex problems, such as solving partial differential equations.
That is noteworthy since the partial differential equations constitute the governing laws to model the physics behind most engineering problems. Their solution is crucial for understanding the behavior of the systems, but usually is complex and requires advanced computational techniques.
In this context is located fracture mechanics, the field of study that examines and predicts how cracks propagate through the structures, to understand how and when failure occurs.
The current work aims to simulate fracture in two bi-dimensional plates by exploiting the potential of neural networks. The study is carried out using Variational Physics-Informed Neural Networks. This type of structure includes the governing equations directly into its architecture, guaranteeing their fulfillment and proving their suitability to investigate the phenomenon.
To study the fracture behavior, the phase-field model is adopted, which is based on a global energy approach. Therefore, the Deep Energy Method is employed to approximate the solution of the differential equations characterizing the phase field model. In this innovative approach the loss function that get minimized is the overall energy of the system.
The results so obtained are compared with the FEM simulation ones, conducted using COMSOL Multiphysics software and adopting the phase-field method. Consequently, the capabilities and the criticalities of the neural network model are highlighted.
The first study investigated a mode I fracture and revealed a good level of accuracy in the results predicted by the neural network. Indeed, both the damage and displacement fields are well approximated once the model parameters have been tuned. However, this level of accuracy significantly affects computational times. It was therefore shown that it is possible to obtain reasonable results in a shorter computational time by sacrificing some accuracy.
The second study highlighted the capability of the neural network to track a mixed mode I-II fracture.

The crack path is outlined, but some differences come out in the field distribution. However, the computational time is extremely short, making this approach convenient.

This study shows that neural networks have great potential in fracture mechanics, as they can provide a fairly accurate estimate of fracture propagation. The results of this project are promises for further developments in the future.

Contents

List of Figures

List of Tables

Acronyms

DEM	Deep Energy Method
FEM	Finite Element Method
HRR	Hutchinson, Rice and Rosengren
KANs	Kolmogorov-Arnold Neural Networks
LEFM	Linear Elastic Fracture Mechanics
L-BFGS	Limited-memory Broyden Fletcher Goldfarb Shanno
NN	Neural Network
PINNs	Physics-Informed Neural Networks
PDEs	Partial Differential Equations
ReLU	Rectified Linear Unit
VPINNs	Variational Physics-Informed Neural Network

Introduction

Materials' behavior and structure failure over time play a primary role in mechanical design. Without their understanding, sudden collapses may occur causing safety problems. Moreover, lacking proper knowledge, technical requirements and safety standards cannot be satisfied.
It is clear that understanding crack nucleation, propagation, and their effect on the resistance of the material is essential. Indeed, cracks are responsible for major structural collapses, which can happen without a warning.
Experimental tests are often impractical and expensive and, consequently, they cannot be performed in every situation. Furthermore, the majority of physical phenomena are typically described by partial differential equations (PDEs). Their analytical solution is not always available, and, due to their complexity, a numerical resolution is the only alternative.
All these needs, over the years, led to the development of several numerical approaches that allowed the resolution of the PDEs and made the material response available in different scenarios without conducting physical tests.
Nowadays, many methods are available for simulating cracking and, within these, the phase-field method is particularly suitable since the crack geometry doesn't need to be explicitly tracked. Instead, it is described by a continuous phase variable, enabling the description of both nucleation and damage propagation.
At the same time, thanks to the constant evolution of machine learning, innovative methods are emerging. In particular, adopting neural networks to model fracture behavior has recently become widespread.
Physics-Informed Neural Networks (PINNs) and Variational Physics-Informed Neural Networks (VPINNs) are capable of solving the PDEs, supplying a numerical solution. In light of that, they appear as a promising alternative to more traditional methods.
In this work, the study of bi-dimensional plates is conducted, under the assumptions of linear elastic fracture mechanics, using the phase-field method in COMSOL Multiphysics and then through neural networks.
The main purpose is to evaluate the ability of neural networks to predict fracture propagation, to assess if they can be a suitable approach.

In the first chapter, an overview of the main numerical methods used to study fracture propagation is presented, including the phase-field method,

which is the one adopted in this work.
Next, the topic of artificial neural networks is introduced describing their architecture, the training process, and the most common optimization algorithms that enable it.

The second chapter describes two classes of neural networks: PINNs and VPINNs. They are especially suitable to be applied in the engineering field, such as in fracture mechanics, since they allow the fulfillment of the PDEs by integrating them into their structure. Next, a description of the Deep Energy Method (DEM) is provided. This approach optimizes network parameters minimizing the global energy of a system.
The Kolmogorov-Arnold Neural Network (KAN) architecture is then briefly introduced as an alternative to classical PINNs.

In the third chapter, it is explained how DEM and the phase-field equations are integrated to study fracture propagation and solve the damage and displacement fields. Then, an in-depth analysis of the types of discretization and the integration methods used to solve the variational problem is presented.

In the fourth chapter, the implementation of the phase-field method within the software COMSOL Multiphysics is shown.
Initially, it describes how to use the built-in model already present within the software. In the second part, it is deeply explained the procedure to manually implement the governing equations characterizing this method.

Finally, two bi-dimensional plates are analyzed in the fifth chapter. Here, crack nucleation and propagation are investigated through the two explored approaches: phase-field in COMSOL Multiphysics and neural network-based predictions.
The presented results are analyzed and a comparison is made to validate the accuracy of the neural network approximation.

In the last chapter, a conclusive summary of the work is conducted.

Chapter 1

Theoretical Foundations

During their life, structures are subjected to several stresses that can lead to crack nucleation and propagation, causing their structural collapse.
The topic of fracture plays a fundamental role in the engineering field, as failure prevention is one of the primary constraints in mechanical design.
Since carrying out experimental tests at each design stage is impossible, different numerical models and techniques have been developed to model fracture phenomena [1].
The first part of this chapter is dedicated to the description of some numerical models used to predict fracture evolution. Following this order, it includes the Linear Elastic Fracture Mechanics, the J-Integral, and the Phase-field Model. This last method is deeply discussed as it is the one adopted in the present work.
The second part presents the basic architecture and functioning of neural networks: there are described the forward and the backward propagation and the most common activation functions, with a final digression on regularization techniques.

1.1 Fracture Mechanics

1.1.1 Linear Elastic Fracture Mechanics

Linear Elastic Fracture Mechanics (LEFM) describes fracture in materials under the hypothesis of linear elasticity, through a discrete approach.
It allows the rating of the extension and the direction of crack growth before the component fails, as well as the determination of the minimum load required to initiate crack propagation, which can occur in two modes: stable or unstable. In stable propagation, an increasing external load is required to allow further crack growth. Conversely, in unstable propagation, the crack continues to grow without an increasing external load [2].
This theory is valid only if the plastic zone at the crack tip is negligible compared to the total crack dimensions. In other words, the material's behavior

must be linear elastic up to fracture and the plastic deformation must remain confined to a small zone around the crack tip [2].
The stress state at the crack tip can be described using either a global energetic approach or an approach based on the determination of the stress intensity factor [3].
The energetic approach, initially developed by Griffith for ideally brittle materials, is based on a total energy balance of the entire system.
Considering a cracked elastic plate and its loading system [4], the corresponding total energy U is expressed as

$$U = U_o + U_a + U_\gamma - F \tag{1.1}$$

where

- U_o is the elastic potential energy of the plate before the introduction of the crack;
- U_a takes into account the change in elastic potential energy resulting from the introduction of the crack;
- U_γ is the surface energy contribution due to the creation of new surfaces;
- F represents the work done by external forces.

Since U is a function of the half crack length a, the equilibrium condition concerning the crack extension can be derived by imposing

$$\frac{dU}{da} = 0 \tag{1.2}$$

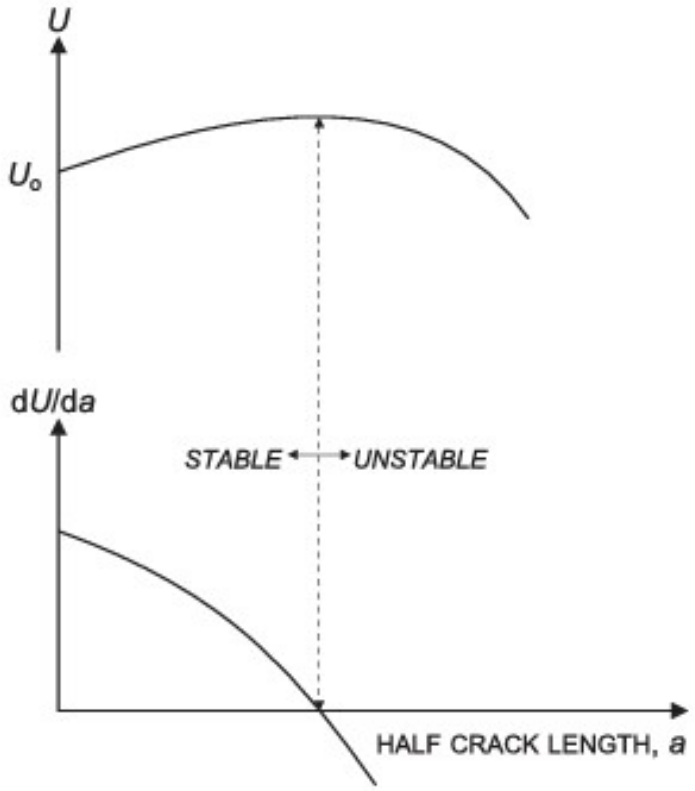


Figure 1.1: Variation of U as a function of a [4].

As illustrated in Fig.1.1, the equilibrium position marks the transition from stable to unstable crack growth. The unstable regime occurs when U decreases with the crack length, which happens after U reaches its maximum value. Mathematically, this can be expressed as

$$\frac{dU}{da} < 0 \tag{1.3}$$

or,since U_0 is independent of a

$$\frac{d(U_a + U_\gamma - F)}{da} < 0 \tag{1.4}$$

Rearranging Eq.(1.4)

$$\frac{d(F - U_a)}{da} > \frac{dU_\gamma}{da} \tag{1.5}$$

$$-\frac{dU_p}{da} > \frac{dU_\gamma}{da} \tag{1.6}$$

The extension of the crack by a quantity equal to da causes a decrease in potential energy and, at the same time, an increase in surface energy.
According to Eq.(1.6), for crack growth to occur, the energy available for crack

extension must be greater than the energy required.
The energy available for an increment of crack extension $d(2a)$ was defined by Irwin as the energy release rate G, while the energy required for that increment is defined as the crack resistance R, which is a material property [4].
In light of this distinction, Eq.(1.6) can be expressed in a simpler form as

$$G = \frac{d(F - U_a)}{d(2a)} > R = \frac{dU_\gamma}{d(2a)} \tag{1.7}$$

If the crack size is small relative to the dimensions of the plate and the load conditions are fixed - meaning no external work is performed - the energy U_a of a finite plate approximates that of an infinite plate

$$G = \frac{d(-U_a)}{d(2a)} \approx \frac{d}{d(2a)}(\frac{\pi\sigma^2 a^2}{E}) = \frac{\pi\sigma^2 a}{E} \tag{1.8}$$

The surface energy U_γ is given by the product of the material's surface tension and the crack's surface area, which consists of two surfaces, each having a length of $2a$.

$$U_\gamma = 2(2a\gamma_e) \tag{1.9}$$

Finally, for a small central crack in a large plate loaded under fixed grip conditions, Eq.(1.7) becomes

$$G = \frac{\pi\sigma^2 a}{E} > G_c = R = 2\gamma_e \tag{1.10}$$

This criterion is effective for brittle materials, where plastic deformation can be entirely neglected.
Later, a modification was proposed by Irwin and Orowan to extend Griffith's criterion to both brittle materials and metals that exhibit plastic deformation [4]. The condition for crack propagation is thus modified as:

$$G = \frac{\pi\sigma^2 a}{E} > G_c = 2(\gamma_e + \gamma_p) = R \tag{1.11}$$

where γ_p is the plastic strain work.
The second approach is based on the evaluation of the stress intensity factor K, which depends on the applied nominal stress σ, the length of the crack a, the geometry of the specimen, and the crack location through the geometric factor Y.
It is generally expressed in the form

$$K = Y\sigma\sqrt{a} \tag{1.12}$$

According to Irwin's theory [2], the stress field at the crack tip is given by the equation

$$\sigma_{ij} = \frac{K}{\sqrt{2\pi r_c}} f_{ij}(\theta) \tag{1.13}$$

where

- K is the stress intensity factor;
- σ_{ij} represents the components of the stress tensor;
- r_c and θ are the polar coordinates with the origin at the crack tip;
- $f_{ij}(\theta)$ is a dimensionless shape function.

K and $f_{ij}(\theta)$ vary depending on the type of crack propagation.
There are three cracking modes [5]:

- Mode I or opening mode: the load is perpendicular to the crack plane and tends to open the crack;
- Mode II or sliding mode: the shear load is parallel to the crack's plane and tends to slide one crack face with respect to another;
- Mode III or tearing mode: the fracture is identified by the lateral shear movement of the two crack surfaces relative to each other. The shear load is perpendicular to the crack direction and it generates an out-of-plane deformation.

The stress fields due to each mode are expressed by the following set of equations [2]

$$\begin{cases} \sigma_{xx} = \frac{K_I}{\sqrt{2\pi r_c}} cos(\frac{\theta}{2})[1 - sin(\frac{\theta}{2})sin(\frac{3\theta}{2})] \\ \sigma_{yy} = \frac{K_I}{\sqrt{2\pi r_c}} cos(\frac{\theta}{2})[1 + sin(\frac{\theta}{2})sin(\frac{3\theta}{2})] \\ \tau_{xy} = \frac{K_I}{\sqrt{2\pi r_c}} sin(\frac{\theta}{2})cos(\frac{\theta}{2})cos(\frac{3\theta}{2})] \\ \sigma_{zz} = \nu(\sigma_{xx} + \sigma_{yy}) \\ \tau_{xz} = \tau_{yz} = 0 \end{cases} \tag{1.14}$$

$$\begin{cases} \sigma_{xx} = \frac{K_{II}}{\sqrt{2\pi r_c}} sin(\frac{\theta}{2})[2 + cos(\frac{\theta}{2})cos(\frac{3\theta}{2})] \\ \sigma_{yy} = \frac{K_{II}}{\sqrt{2\pi r_c}} sin(\frac{\theta}{2})cos(\frac{\theta}{2})cos(\frac{3\theta}{2})] \\ \tau_{xy} = \frac{K_{II}}{\sqrt{2\pi r_c}} cos(\frac{\theta}{2})[1 - sin(\frac{\theta}{2})sin(\frac{3\theta}{2})] \\ \sigma_{zz} = \nu(\sigma_{xx} + \sigma_{yy}) \\ \tau_{xz} = \tau_{yz} = 0 \end{cases} \tag{1.15}$$

$$\begin{cases} \sigma_{xx} = \sigma_{yy} = \sigma_{zz} = \tau_{xy} = 0 \\ \tau_{xz} = \frac{K_{III}}{\sqrt{2\pi r_c}} \\ \tau_{yz} = \frac{K_{III}}{\sqrt{2\pi r_c}} cos(\frac{\theta}{2}) \end{cases} \tag{1.16}$$

The mode I takes place most frequently; however, depending on the type of load, a mixed mode can also occur.
As can be seen from the set of equations above, the stress tends to infinity as r_c approaches zero, indicating that the stress at the crack tip is infinite [4].
This result describes a stress singularity which, from a physical point of view, is an inconsistency. In reality, there is a small plastic region around the crack tip that limits the stress value.
However, if the plastic region has negligible dimensions and does not alter the stress distribution in the vicinity of the crack tip, the hypothesis of linear elasticity is valid, and LEFM can be applied.
Due to the use of K to characterize the stress field, a threshold parameter can be identified to formulate a fracture criterion.
If the component is subjected to a single cracking mode, this limit is identified as $K_{i,cr}$ and it is called fracture toughness [2].
Unstable crack propagation occurs when the material is unable to withstand the stresses at the crack tip. This can be expressed through the following formulation

$$K_i \geq K_{i,cr} \tag{1.17}$$

The mixed-mode case requires a more general formulation of the type [1]

$$f(K_I, K_{II}, K_{III}) = 0 \tag{1.18}$$

There is a direct relationship between the stress intensity factor K and the energy release rate G. For a given mode i, the expressions are as follows [2, 4]

$$G_I = \begin{cases} \frac{K_I^2}{E}, & \text{for plane stress} \\ \frac{(1-\nu^2)K_I^2}{E}, & \text{for plane strain} \end{cases} \tag{1.19}$$

$$G_{II} = \frac{(1-\nu^2)K_{II}^2}{E} \tag{1.20}$$

$$G_{III} = \frac{K_{III}^2}{2\mu_L} \tag{1.21}$$

where E is the Young's modulus, ν is the Poisson ratio and μ_L is the shear modulus.

Crack propagation can also occur due to fatigue under the application of a cyclic loading characterized by a stress range $\Delta\sigma = \sigma_{max} - \sigma_{min}$.
The stress at the crack tip can be evaluated as the difference between the maximum and the minimum intensity factors associated with the loading cycle, as expressed by Eq.(1.22) [4].

$$\Delta K = K_{I,max} - K_{I,min} = Y(\sigma_{max} - \sigma_{min})\sqrt{a} = Y\Delta\sigma\sqrt{a} \tag{1.22}$$

Thanks to experimental studies, it was possible to correlate ΔK with the fatigue crack propagation rate, which is the differential ratio between the crack

extension da and the number of cycles dN.
Plotting this relation on a logarithmic scale, it is possible to observe that the process of crack growth can be divided into three different stages, according to Fig.1.2 [2, 4].

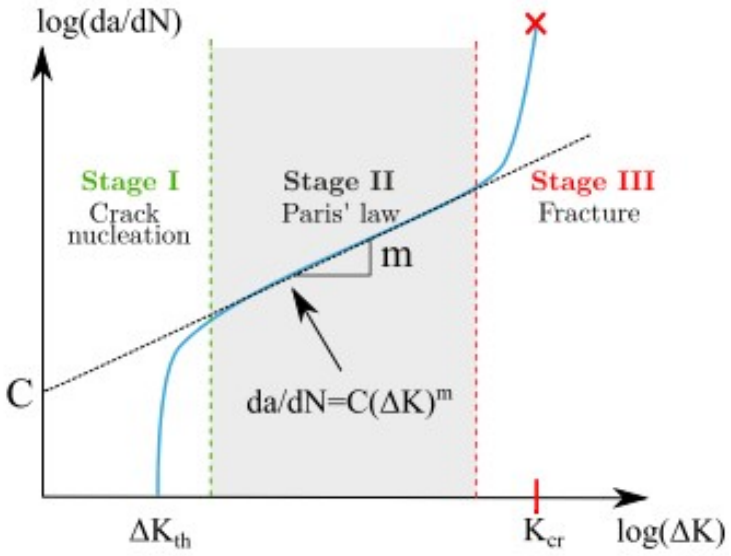


Figure 1.2: Crack growth rate curve (da/dN) as a function of the stress intensity factor range (ΔK) [6].

- Stage I: nucleation of small cracks. It is characterized by a threshold parameter ΔK_{th} below which there isn't an appreciable crack growth, so until the following relation is valid

$$\Delta K < \Delta K_{th} \tag{1.23}$$

- Stage II: stable propagation of cracks
- Stage III: the crack growth rate rises rapidly, causing unstable propagation. K_{max} reaches the critical stress intensity factor leading to component failure.

The stable propagation is usually described by Paris' law, which establishes a linear relationship between $\log \frac{da}{dN}$ and $\log \Delta K$ and is defined as

$$\frac{da}{dN} = C(\Delta K)^m \tag{1.24}$$

where C and m are materials constants determined experimentally [2].
While this law is applicable for mode I loading [7], many failures occur due to cracks subjected to mixed-mode loading. It is possible to extend Paris' law by evaluating ΔK_{eq}, which is a combination of K_I and K_{II}, and substituting it

in Eq.(1.24). In Table 1.1 are reported the most common expressions used for ΔK_{eq}.

Source	ΔK_{eq} **models**
Tanaka_1	$\Delta K_{eq} = (\Delta K_I^2 + 2\Delta K_{II}^2)^{1/2}$
Tanaka_2	$\Delta K_{eq} = (\Delta K_I^4 + 8\Delta K_{II}^4)^{1/4}$
Irwin	$\Delta K_{eq} = \sqrt{\Delta K_I^2 + \Delta K_{II}^2}$
Yan, et al	$\Delta K_{eq} = \frac{1}{2}\cos\frac{\theta}{2}\left[\Delta K_I(1+\cos\theta) - 3\Delta K_{II}\sin\theta\right]$
Hussain, et al	$\Delta K_{eq} = \sqrt{\frac{4}{(3+\cos^2\theta)^2}\left(\frac{1-\theta/\pi}{1+\theta/\pi}\right)^{\theta/\pi}\left[(1+3\cos^2\theta)\Delta K_I^2 + 4\sin 2\theta\,\Delta K_I\Delta K_{II} + (9-5\cos^2\theta)\Delta K_{II}^2\right]}$
Richard, et al	$\Delta K_{eq} = \frac{\Delta K_I}{2} + \frac{1}{2}\sqrt{\Delta K_I^2 + 4(1.155\Delta K_{II})^2}$

Table 1.1: Models for ΔK_{eq} from [7].

By integrating Eq.(1.24), the increment in the number of cycles ΔN_i associated with a specific crack growth can be determined. In this manner, it becomes possible to determine the total number of cycles that the component can stand before experiencing failure. Indeed, the fatigue life cycle is determined by summing ΔN_i for each step [6], following the equation

$$N_t = \sum_{i=1}^{n} \Delta N_i \tag{1.25}$$

It should be noted that the main drawback of LEFM theory is its inability to model the crack nucleation stage: fracture can be predicted only in bodies with pre-existing cracks, so a priori presence of a crack is assumed [2].

1.1.2 J integral

A method widely employed to characterize fracture behavior at the crack tip in elasto-plastic materials is the J contour integral, a path-independent parameter introduced by Rice in 1968 [3].
The main assumption behind this theory is idealizing elastic-plastic deformation as nonlinear elastic. In this latter case, even if the material's behavior is nonlinear, there is a unique relationship between stress and strain. The two models are essentially equivalent as long as stresses increase monotonically.
Thus, this approximation is valid until unloading or cyclic loading occurs, since elastic-plastic materials exhibit irreversible plasticity and hysteresis.

The J integral [3] for a two-dimensional fracture problem, considering any counterclockwise path Γ surrounding the crack tip, is defined as

$$J = \oint_{\Gamma} \left(w\, dy - T_i \frac{du_i}{dx}\, ds \right) \tag{1.26}$$

where

- w is the strain energy density;
- T_i are the components of the traction vector acting on the path, defined as $T_i = \sigma_{ij} n_j$, where n_j represents the components of the unit vector normal to Γ;
- u_i are the displacement vector components;
- d_s is a length increment along the contour Γ.

Rice demonstrated two key properties of the J integral: its value is independent of the integration path and it is equivalent to the energy release rate when LEFM is valid [2].
This implies that, for the linear elastic case, the following relationship holds [8]

$$G = J = \frac{1}{\bar{E}}(K_I^2 + K_{II}^2) + \frac{1+\nu}{E} K_{III}^2 \tag{1.27}$$

where

$$\bar{E} = \begin{cases} \frac{E}{1-\nu^2} & \text{for plane strain,} \\ E & \text{for plane stress.} \end{cases} \tag{1.28}$$

The J integral [3] can be regarded not only as an energetic parameter but also as a stress intensity factor, as described by the HRR theory, from Hutchinson, Rice and Rosengren.
Assuming a power-law hardening material of the form

$$\frac{\epsilon}{\epsilon_0} = \frac{\sigma}{\sigma_0} + \alpha(\frac{\sigma}{\sigma_0})^n \tag{1.29}$$

the stresses and strains ahead of the crack tip can be expressed using the following formulation

$$\sigma_{ij} = k_1(\frac{J}{r})^{\frac{1}{n+1}} \tag{1.30}$$

$$\epsilon_{ij} = k_2(\frac{J}{r})^{\frac{n}{n+1}} \tag{1.31}$$

where σ_0 is a reference stress value usually equal to the yield strength, $\epsilon_0 = \frac{\sigma_0}{E}$, α is a dimensionless constant, k_1 and k_2 are proportionality constants and n is the strain hardening exponent.
A similarity with the LEFM theory can be observed: when r approaches

zero, the stress approaches infinite values, showing a singularity called HRR singularity [3]. In reality, plastic deformations cause the blunting of the crack tip and the reduction of the stress intensities. This region is called large deformation area, and the HRR theory is not valid due to significant plasticity. As initially described, the approximation underlying this theory assumes the behavior of an elastic-plastic material to be nonlinear.
This approach has some limitations in the fracture studies of ductile materials, where some unloading processes occur due to crack propagation. In these cases, the J integral is no longer path-independent and the use of this approximation is not adequate. Instead, a flow theory of plasticity is required to accurately describe the material's behavior [8].
Moreover, the J integral method can evaluate the stability of pre-existing crack-like but cannot predict crack initiation and propagation from stress raisers or notches and it is not applicable to arbitrarily complex geometries [9].

1.1.3 Phase-field model

Phase-field modeling provides a viable method for representing fracture phenomena. It can effectively simulate complex processes, such as crack initiation, propagation, merging, and branching assuming that the discontinuities introduced by the crack are not sharp, but can be approximated as diffuse damage using a scalar field parameter, known as phase-field variable (d). This parameter d varies continuously across the space assuming values that go from 0 to 1, distinguishes the intact from the cracked material, in accordance with Eq.(1.32)

$$d(\mathbf{x}) = \begin{cases} 0, & \text{unbroken material} \\ 1, & \text{completed broken material.} \end{cases} \tag{1.32}$$

The evolution of d due to external loading conditions models the fracture process and the propagation of cracks is automatically tracked.
As a result, the representation of fractures is no longer dependent on geometry or mesh [1, 2, 9].
The phase-field model is derived from the pioneering work of Francfort and Marigo [10], who developed a variational theory of fracture grounded in energy minimization principles.
Later, Bourdin et al.[11] introduced a regularized formulation to facilitate efficient numerical implementation by introducing a length scale parameter (l_0), which governs the width of the diffusive crack zone and determines the region over which the variable d evolves. The phase-field model converges to Griffith's theory for brittle fracture as l_0 tends to zero [2].
The determination of the correct value for the length scale parameter is still challenging, as it significantly affects the accuracy of the results and must account for both the material properties and the numerical discretization. A poorly chosen l_0 can lead to inaccurate simulations, as it influences the width of the damage transition zone, which, in turn, impacts fracture propagation.

Additionally, it needs to be compatible with the mesh resolution, as an improper scale may either require an excessively fine mesh or lead to numerical instabilities.
Borden et al. in [12] defined an empirical formulation for an initial estimation of l_0, which is directly linked to material properties as it is expressed by the following formulation

$$l_0 = \frac{27EG_c}{256\ \sigma_{cr}^2} \tag{1.33}$$

where G_c is the critical release rate, E is the Young modulus, σ_{cr} is the critical stress and they can be determined through experimental tests [13].
The rate-independent dynamic phase-field model [9, 14] was introduced by Borden et al. as an extension of the quasi-static model of Bourdin et al..
Consider an arbitrarily shaped domain $\Omega \subset \mathbb{R}^d$, with d as the spatial dimension, having boundary $\partial\Omega$, as shown in Fig.1.3. The body contains a crack denoted by Γ. Let $u(x,t) \in \mathbb{R}^d$ be the displacement of a point $x \in \Omega$ at time t which satisfies the Dirichlet and Neumann boundary conditions on $\partial\Omega_D$ and $\partial\Omega_N$, with $\partial\Omega_D \cup \partial\Omega_N = \partial\Omega$ and $\partial\Omega_N \cap \partial\Omega_D = \emptyset$.

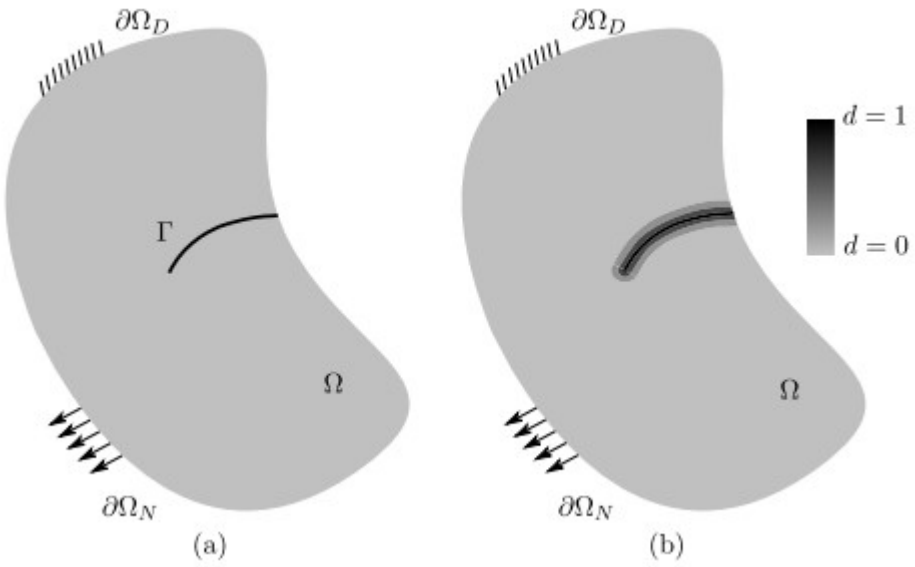


Figure 1.3: Schematic of a solid with (a) a sharp and (b) a regularized diffuse crack [15].

According to Griffith's theory, when cracks propagate, a part of the strain energy density is dissipated and stored as fracture energy. The governing equation of the problem [2, 15] can be derived from the total potential energy

of the cracked body, given by

$$\begin{aligned}\Psi_{pot}(u,\Gamma) &= \Psi_e + \Psi_{frac} + \Psi_{ext} \\ &= \int_\Omega \Psi_e(\varepsilon(u))\, d\Omega + \int_\Gamma G_c\, d\Gamma \\ &\quad - \int_\Omega b \cdot u\, d\Omega - \int_{\partial\Omega_N} T \cdot u\, d\Gamma \end{aligned} \tag{1.34}$$

where

- Ψ_e is the elastic strain energy density function;
- $\varepsilon(x,t)$ is the second-order strain tensor defined as the symmetric part of the displacement gradient;
- G_c is the fracture toughness of the material;
- T is the external traction vector;
- b represents the body force.

The kinetic energy of the body, instead, is defined as

$$\Psi_{kin}(\dot{u}) = \frac{1}{2}\int_\Omega \rho\, \dot{u} \cdot \dot{u}\, dx \tag{1.35}$$

where

- ρ is the mass density of the material;
- $\dot{u} = \frac{\partial u}{\partial t}$ is the velocity.

The Lagrangian for the discrete fracture problem is defined as

$$\begin{aligned} L(u,\dot{u},\Gamma) &= \Psi_{kin}(\dot{u}) - \Psi_{pot}(u,\Gamma) \\ &= \int_\Omega \left[\frac{1}{2}\rho\, \dot{u} \cdot \dot{u} - \Psi_e(\varepsilon(u))\right] d\Omega - \int_\Gamma G_c\, d\Gamma \\ &\quad + \int_\Omega b \cdot u\, d\Omega + \int_{\partial\Omega_N} T \cdot u\, d\Gamma \end{aligned} \tag{1.36}$$

The fracture surface integral [2] can be approximated, introducing the regularized formulation to enable a numerically efficient implementation, with a volume integral defined over the entire domain Ω, according to Eq. (1.37)

$$\Psi_{frac} = \int_\Gamma G_c\, \mathrm{d}\Gamma \approx \int_\Omega G_c\, \gamma(d,\nabla d)\, \mathrm{d}\Omega = \int_\Omega G_c \left(\frac{1}{2\ell} d^2 + \frac{\ell}{2}|\nabla d|^2\right) \mathrm{d}\Omega \tag{1.37}$$

where $\gamma(d,\nabla d)$ is the crack surface density function.
From this equation, it is clear that a crack is represented by regions where d approaches one.
Crack propagation results in a gradual degradation of the material's stiffness,

which is accounted for by coupling the elastic strain energy density with the phase-field variable d through the degradation function $g(d)$.
Different expressions of the degradation function can be found in the literature, as shown in Fig.1.4. One of the most commonly used formulations is the one proposed by Bourdin et al.[2].

$g(\phi)$	authors
$(1-\phi)^2$	Bourdin et al. [2000]
$3(1-\phi)^2 - 2(1-\phi)^3$	Karma et al. [2001]
$(3-s)(1-\phi)^2 - (2-s)(1-\phi)^3$	Borden et al. [2016]
$4(1-\phi)^3 - 3(1-\phi)^4$	Kuhn et al. [2015]
$\frac{(1-\phi)^2}{(1-\phi)^2+Q(\phi)}$, $Q(\phi) = a_1\phi + a_1 p\phi^2$	Lorentz et al. [2012, 2011], Lorentz [2017]
$\frac{(1-\phi)^2}{1+(k-1)Q(\phi)}$, $Q(\phi) = 1-(1-\phi)^2$	Alessi et al. [2015]
$\frac{(1-\phi)^p}{(1-\phi)^p+Q(\phi)}$, $Q(\phi) = a_1\phi + a_1a_2\phi^2 + a_1a_2a_3\phi^3$	Wu [2017]

Figure 1.4: Frequently used degradation functions [16].

The function $g(d)$ monotonically decreases as d increases and satisfies the two conditions $g(d=0)=1$ and $g(d=1)=0$.
To prevent crack growth under compression, which would lead to non-physically crack evolution paths, and to distinguish between tension and compression fracture behavior, only the tensile part of the elastic strain energy is affected by the degradation function, while the compressive part remains undegraded. This formulation is expressed in accordance with Eq.(1.38)

$$\Psi_e(\epsilon, d) = g(d)\Psi_e^+(\epsilon) + \Psi_e^-(\epsilon) \tag{1.38}$$

where $\Psi_e^+(\epsilon)$ and $\Psi_e^-(\epsilon)$ are respectively the tensile and compressive contributions of the elastic strain energy density [2, 9].
There are two main approaches in the literature for decomposing the elastic energy density [15].
According to the first formulation, the elastic energy is divided into volumetric and deviatoric contributions, as shown in the following expressions:

$$\Psi_e^+(\epsilon) = \frac{1}{2}K_n \langle \mathrm{tr}(\epsilon)\rangle_+^2 + \mu(\epsilon_{\mathrm{dev}} : \epsilon_{\mathrm{dev}}) \tag{1.39}$$

$$\Psi_e^-(\epsilon) = \frac{1}{2}K_n \langle \mathrm{tr}(\epsilon)\rangle_-^2 \tag{1.40}$$

$$K_n = \lambda + \frac{2\mu}{3}, \quad \langle a\rangle_\pm = \frac{1}{2}\Big(a \pm |a|\Big), \quad \epsilon_{\mathrm{dev}} = \epsilon - \frac{1}{3}\mathrm{tr}(\epsilon)I \tag{1.41}$$

where λ and μ are the Lamè coefficients.
The second formulation, instead, provides a spectral decomposition of the strain tensor $\varepsilon = \sum_{I=1}^{3} \langle \varepsilon_I \rangle \, \mathbf{n}_I \otimes \mathbf{n}_I$, where $\{\varepsilon_I\}_{I=1}^{3}$ and $\{\mathbf{n}_I\}_{I=1}^{3}$ are the principal strains and principal strain directions, respectively.
Thus, considering that $\varepsilon_{\pm} = \sum_{I=1}^{3} \varepsilon_I \pm \mathbf{n}_I \otimes \mathbf{n}_I$, the positive and the negative parts of the elastic strain energy density are given by

$$\Psi_e^{\pm}(\varepsilon) = \frac{1}{2}\lambda \langle \mathrm{tr}(\varepsilon) \rangle_{\pm}^2 + \mu \, \mathrm{tr}(\varepsilon_{\pm}^2). \tag{1.42}$$

By substituting what is stated in Eq.(1.36), it is obtained that

$$\begin{aligned} L(u, \dot{u}, d) = &\int_{\Omega} \left[\frac{1}{2}\rho \dot{u} \cdot \dot{u} - g(d)\Psi_e^{+}(\boldsymbol{\epsilon}) - \Psi_e^{-}(\boldsymbol{\epsilon}) \right] d\Omega \\ &- \int_{\Omega} G_c \left(\frac{1}{2\ell} d^2 + \frac{\ell}{2} |\nabla d|^2 \right) d\Omega \\ &+ \int_{\Omega} b \cdot u \, d\Omega + \int_{\partial\Omega_N} T \cdot u \, d\Gamma \end{aligned} \tag{1.43}$$

Then, assuming a quasi-static condition, the Euler-Lagrange equations of displacement u and phase-field variable d are obtained by finding a stationary point of the Lagrangian imposing $\partial L = 0$, corresponding to its minimization [2]. The strong form of the problem and the boundary conditions are reported below [15].

$$\begin{cases} \nabla \sigma + b = 0 & \text{in } \Omega, \\ G_c \left(\frac{d}{l_0} - l_0 \Delta d \right) - 2(1-d)\Psi_e^{+}(\epsilon) = 0 & \text{in } \Omega. \end{cases} \tag{1.44}$$

$$\begin{cases} u = u_D & \text{on } \partial\Omega_D, \\ \sigma \cdot n - T = 0 & \text{on } \partial\Omega_N, \\ \nabla d \cdot n = 0 & \text{on } \partial\Omega. \end{cases} \tag{1.45}$$

where σ is the Cauchy stress tensor and it is expressed by

$$\sigma = \frac{\partial \Psi_e}{\partial \epsilon} = g(d) \frac{\partial \Psi_e^{+}}{\partial \epsilon} + \frac{\partial \Psi_e^{-}}{\partial \epsilon}$$

A local history field variable H is introduced to ensure the irreversibility of the phase-field evolution, preventing the reduction of the crack length as the tensile energy decreases [2].
H is defined as the maximum tensile contribution of the elastic energy density function, as outlined in Eq.(1.46), and must comply with the Kuhn-Tucker conditions [12] for both loading and unloading, as expressed in Eq.(1.47).

$$H(x,t) = \max_{d \in [0,t]} \Psi_e^{+} \tag{1.46}$$

$$\Psi_e^{+} - H \leq 0, \quad \dot{H} \geq 0, \quad \dot{H}(\Psi_e^{+} - H) = 0 \tag{1.47}$$

Replacing Ψ_e^+ by $H(x,t)$ in Eq.(1.44), the strong form is obtained by

$$\begin{cases} \nabla\sigma + b = 0 & \text{in } \Omega, \\ G_c\left(\frac{d}{l_0} - l_0\Delta d\right) - 2(1-d)H = 0 & \text{in } \Omega. \end{cases} \tag{1.48}$$

In addition, the equations of motion are supplemented with initial conditions

$$\begin{cases} u(x,0) = u_0(x) \\ \dot{u}(x,0) = v_0(x) \\ d(x,0) = d_0(x) \end{cases} \tag{1.49}$$

where the initial phase-field d_0 can be used to model pre-existing cracks by setting its value equal to 1[2, 12].
Summarizing, the major advantages of this method are [16]:

- It is based on the energy minimization principle and doesn't require the modeling of pre-existing cracks, so crack nucleation, growth, and coalescence can be automatically determined;
- Crack interfaces are described using a continuous variable, allowing for a smooth transition between different areas;
- It can deal with merging and branching of multiple cracks without a greater effort;
- It can deal with multi-physics problems.

However, the main drawback is the high computational cost, as a sufficiently refined mesh in the damaged zone is required to accurately study fracture behavior.

1.2 Artificial Neural Networks

Artificial neural networks are computational models, that have their roots in the structure and operation of the human brain, capable of identifying complex relationships between input and output data [17, 18].
Their suitability for a wide range of applications is linked to their ability to make predictions, as outlined by the universal approximation theorem formulated by Hornik [19, 20].
A deep neural network [21, 22] is a computational system comprising multiple layers, each having several processing units, called neurons. Its main goal is to approximate a continuous function mapping inputs to corresponding outputs. The basic architecture of a network, illustrated in Fig.1.5, is organized as follows:

- Input layer: the first layer, also called the 0-th layer, where the initial data are fed;

- Hidden layers: one or more intermediate layers, that process data;
- Output layer: the last layer, also called the L-th layer (where L is the total number of layers), which provides the network's outputs.

Fig.1.5 schematically shows the structure of a deep neural network, highlighting the different layers and the connections between neurons.

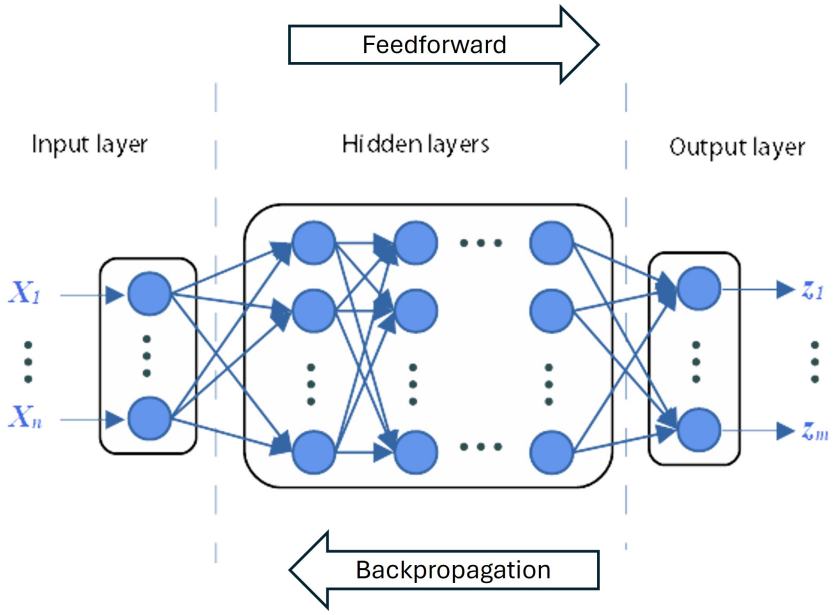


Figure 1.5: Structure of a deep neural network with layers and connections (adapted from [20]).

One of the most common types of architectures is the feed-forward fully connected neural network.
In this network, all neurons in one layer are connected to every other neuron in the following layer and the information flows through the network in one direction only: from the input layer, through the hidden layers, to the output layer. This unidirectional flow gives the name "feed-forward" to the network [18].
The input of the i-th neuron in the l-th layer is expressed as [22]:

$$z^l = \sigma_{l-1}\left(\sum_{j=1}^{m_l-1} W_{i,j}^l z_j^{l-1} + b_i^l\right) \tag{1.50}$$

where

- b_i^l and $W_{i,j}^l$ are, respectively, the adaptive bias vector and the weight matrix, determined through a training process. Weight stands for the contribution of the previous neurons to the current one, while the bias shifts the activation function to better fit data;

- z_j^{l-1} is the output from the $(l-1)$-th layer;
- $m_l - 1$ denotes the number of neurons in the $(l-1)$-th layer;
- $\sigma_{l-1}(\cdot)$ stands for the activation function of the $(l-1)$-th layer.

This process is repeated sequentially layer by layer: the outcome of a previous layer becomes the input for the next one until the output layer produces the final results.
The activation function [22] performs a nonlinear transformation on the input data. Without an activation function, a network would not be able to approximate complex relationships or capture nonlinearities in data.
Typically, the same activation function is applied to all neurons in a layer. In most cases, the output layer is a linear function of the last hidden layer so that the set of outcomes is unrestricted. This means that no activation function is applied to the output layer. The most common activation functions are analyzed in section 1.2.1.
The number of hidden layers determines the depth of the network while the number of neurons in each layer determines its width.
To set up a deep neural network, the following parameters must be designed [22]:

- The number of hidden layers;
- The number of neurons per layer;
- The activation function;
- The optimizer algorithm, including the learning rate and the number of epochs. The learning rate determines the step size at each iteration moving toward a minimum of the loss function. The number of epochs, instead, corresponds to the number of complete training cycles on the entire dataset.

All these parameters are called hyperparameters and can be user-defined or chosen by an outer optimization strategy [20].
To ensure that the outputs of the network give the closest approximation to the solution currently under investigation, the optimal parameters of the network, represented by weights and biases of all layers, should be determined [21].
Most of the optimization algorithms, during training, solve an optimization problem that minimizes a cost function, called loss function ($\mathcal{L}(\theta)$), which represents the quality of the approximated solution.

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta) \tag{1.51}$$

The above minimization requires the computation of the gradient of the loss function with respect to the weights and biases. A well-known technique to accomplish this is backpropagation [21], which is better described in section 1.2.2.

1.2.1 Activation functions

A non-linear activation function allows the representation of any non-linearity in the data [18]. The most common activation functions are the sigmoid function, the hyperbolic tangent, and the rectified linear unit (ReLU)[18, 23].
The sigmoid function is defined as

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{1.52}$$

in which $z \in (-\infty, +\infty)$ and $\sigma(z) \in (0,1)$.
The hyperbolic tangent function is defined as the ratio between sine and cosine functions:

$$\sigma(z) = tanh(z) = \frac{sinh(z)}{cosh(z)} = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{1.53}$$

This function is symmetric with respect to the origin and its output ranges between -1 and 1. Due to its properties, neural networks that use this activation function tend to converge faster than those that use the sigmoid function.
However, as the weights increase, these two functions tend to saturate. This means that their derivatives will reduce to zero in the saturation regions, which can hinder proper learning and training of the network. This situation is called vanishing gradient.
The ReLU function is defined as

$$\sigma(z) = max(0, z) = \begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases} \tag{1.54}$$

Since the function grows linearly for positive inputs, it helps reduce saturation and the vanishing gradient problem. Notice that it is not differentiable at $x = 0$, but is usually set to zero.
However, this activation function presents a main disadvantage: it is left hard saturating, meaning its derivative is equal to 0 when $x < 0$, leading to a possible deactivation of some neurons. This can negatively affect convergence.
A graphical representation of the mentioned activation functions is shown in Fig.1.6.

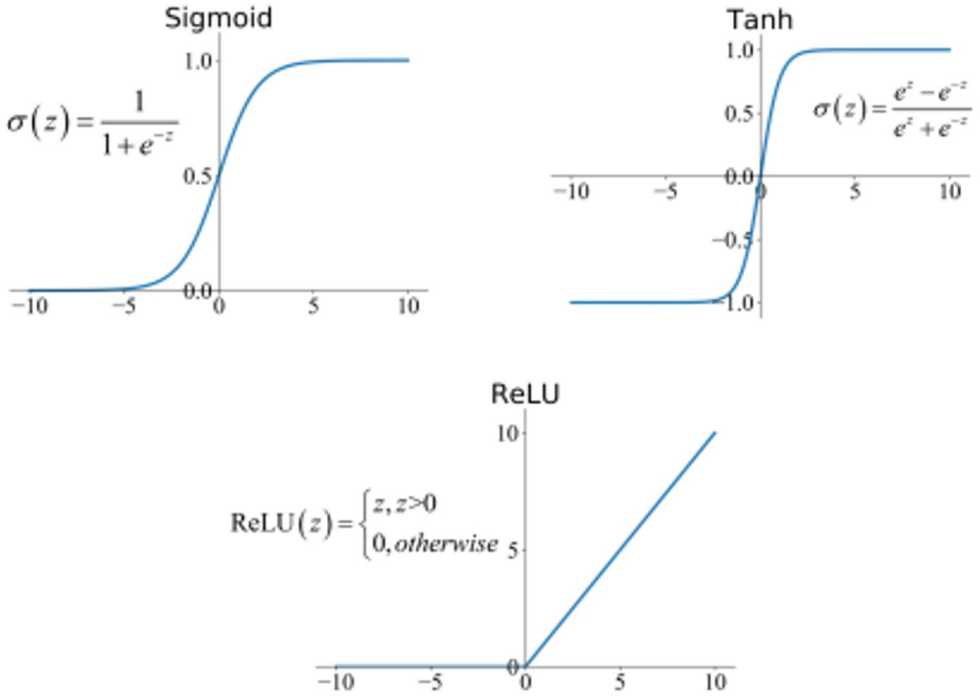


Figure 1.6: Graphical representation of common activation functions [24].

A primary characteristic of all the mentioned activation functions is differentiability, as it is essential for ensuring backpropagation works correctly. Indeed, if this property is not satisfied, the gradients cannot be properly evaluated [22].

1.2.2 Backpropagation

Backpropagation is an efficient technique, part of automatic differentiation, that enables the training of neural networks [18]. It allows the calculation of the gradient of the loss function with respect to network parameters, permitting the minimization of the loss function and the reduction of the error.
The process starts from the output layer and proceeds backward through the network to the input layers, computing at each step the loss function gradient. These gradients are then used by an optimization algorithm to modify the weights and biases, according to an update rule [18, 21].
The name 'backpropagation' refers to the fact that the calculations and updates are done in reverse order, from output to input layer.
While backpropagation is the way to provide gradients, optimization algorithms are responsible for minimizing the loss function and the way learning proceeds. They iteratively generate a series of approximate solutions until a stopping criterion is verified. In each iteration, the algorithm finds a direction and moves from the current point to a new iterate with a lower loss function value [25].
The step size between two consecutive iterations can be a constant value or evaluated based on a line search method.
Although several options are available, two of the most popular gradient-based

optimization algorithms are Adam and L-BFGS [22].
Adam [26], whose name derives from adaptive moment estimation, is a method for efficient stochastic optimization that evaluates individual adaptive learning rates for different parameters.
It is based on the estimation and updating of the first and second moments of the gradients, respectively corresponding to the mean and the uncentered variance.
The first moment estimate is given by (m_t), which corresponds to the exponential moving averages of the gradients, while the second moment is expressed by (v_t), which corresponds to the exponential moving averages of the squared gradients.
At the beginning of training, m_0 and v_0 are initialized as vectors of zero.
At any given step of iteration, moments are updated according to the following rules

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) g_t \tag{1.55}$$

$$v_{t+1} = \beta_2 v_t + (1 - \beta_2) g_t^2 \tag{1.56}$$

where

- t is the time step;
- g is the vector of partial derivatives of the loss function with respect to the model parameters (θ_t);
- β_1 and β_2 control the exponential decay rate of the moments and are within the range [0,1). By default, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Then, the estimates are bias-corrected to take into account the previous initialization to zero.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{1.57}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{1.58}$$

At last, the parameters θ_t are updated

$$\theta_{t+1} = \theta_t - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{1.59}$$

where α is the learning rate and ϵ is a constant for numerical stabilization to prevent division by zero, by default set to 10^{-8}.
This algorithm [26] is straightforward to implement and computationally efficient. It is suitable for problems with a large amount of data or parameters, thanks to the adaptive learning rate. The hyper-parameters usually need a little tuning, since they generally perform well with the default values.
However, although Adam converges quickly in the early stages of training and gets a rough estimate of the optimal value, sometimes it is unable to reach the local optimum accurately.

The Limited-memory Broyden Fletcher Goldfarb Shanno algorithm [25], commonly referred to as L-BFGS, is a second-order quasi-Newton method.
Instead of inverting the Hessian matrix of the loss function at each iteration, as Newton's methods do, it iteratively refines an approximation of the inverse Hessian matrix to determine the direction of descent.
Additionally, instead of storing a fully dense n x n approximated matrix, it only saves a limited number of m vector pairs, with m«n. These vectors collect the differences in parameters ($s_k = \theta_{k+1} - \theta_k$) and in gradients ($y_k = g_{k+1} - g_k$) over the last m iterations. This information is used to update the inverse Hessian approximation.
When m iterations have already been saved, the oldest vector pair is deleted so that there is space to store information about the newest iteration.
The key mechanism of this algorithm is the two-loop recursion, in which the direction of descent is evaluated using only the stored vector pairs, without explicitly computing the inverse Hessian matrix.
The model parameter update is defined as

$$\theta_{t+1} = \theta_t + \alpha_t H_t g_t \tag{1.60}$$

where

- H_t is the approximation of the inverse Hessian matrix.
- g_t is the gradient of the loss function;
- α_t is the step size, determined automatically using a line search.

This optimizer is particularly suitable for large problems, since it avoids the computational costs of inverting the Hessian matrix, which can be expensive for high-dimensional data. Moreover, it requires a limited amount of memory, unlike Newton's methods, since only a few pairs of vectors are stored [25].

1.2.3 Regularization

The capacity of a machine learning model is its ability to fit different functions. Indeed, one primary property of a neural network is the ability to generalize on unseen data [18].
However, the model can face two problems during training: under-fitting and over-fitting. Under-fitting occurs when the network is too simple, for instance, too few neurons, that it is unable to detect signals in a complicated data set. This corresponds to a model with a very low capacity. Over-fitting, instead, is linked to a loss in the generalization ability. The model fits training data perfectly, but it is not able to generate good predictions for unseen data [18, 22].
To mitigate that issue and increase the generalization capability of neural networks on unseen data, a possible approach is to adopt a control mechanism. Regularization [18] includes all strategies aiming to diminish the test error

without increasing the training error.
The most common regularization techniques include:

- Early stopping: it interrupts the training process as soon as the model risks losing its generalization capability. This can be done by monitoring the training and validation errors. During the training, if the training error steadily decreases while the test error starts to increase, this indicates that the model is entering the over-fitting regime.
  The process terminates when the validation error does not improve for a predefined number of consecutive iterations;

- Dropout: at each training iteration half of the hidden neurons are randomly and temporarily dropped, forcing the network to learn parameters with only a subset of its neuron units.
  Deactivating some units is similar to training different neural networks, since at each iteration a different network configuration is used.

- L^1 and L^2 regularization: belonging to the parameter norm penalty class, they prevent over-fitting by penalizing network parameters thanks to a penalty term Ω added to the loss function $\mathcal{L}$.

  $$\tilde{\mathcal{L}} = \mathcal{L} + \lambda\Omega \tag{1.61}$$

  where λ is a hyperparameter that determines the weight of the penalty term. Even if its value can be different for each layer, it is usually set equally for the entire network for simplicity.
  Regularization L^1 adds a penalty term proportional to the sum of the absolute values of the weights ($\lambda \sum |w|$) emphasizing the selection of few high-importance connections and forcing other weights toward zero.
  L^2 adds a penalty term proportional to the sum of the square of weights ($\lambda \sum w^2$). In this second case, instead, the magnitude of weights is reduced proportionally to w, so for small weights it is of less entity compared to L^1 regularization. This can help to stabilize the model, since larger weights modify drastically the outputs even for a small input variation;

- Dataset augmentation: it is a widely used technique in image classification and involves the creation of "fake" data to train the model on a larger dataset. In this context, some variations to the input images are performed, such as translation, rotation, or inclusion of small amounts of random noise.

Chapter 2

State of Science

The current chapter provides a description of some machine learning methods that can be used for the resolution of the governing equations of the engineering problems.
Initially, Physics-Informed Neural Networks and Variational Physics-Informed Neural Networks are described, as they minimize the residual losses of PDEs. Then, the Deep Energy Method is introduced, which considers the total energy of the system as a loss function.
Finally, a brief description of the KAN network is provided as a promising alternative to traditional neural networks.

2.1 Physics-Informed Neural Networks

Physics-informed neural networks are a class of neural networks designed to solve problems involving partial differential equations.
Physical laws governing the problem are directly integrated into the network's structure and loss function so that the response of the system satisfies the physics of the context. Essentially, solutions are obtained by transforming the problem of directly solving the governing equations into an optimization problem [27].
A key advantage behind PINNs is that they can be employed in the absence of labeled data, such as previous simulations or experiments.
Additionally, they can be adopted for both forward and inverse problems. In the former case, the solution is computed based on PDE parameters, initial and boundary conditions; in the latter, the coefficients of the PDEs must be identified from observed data [18, 27].
PINNs are a family of collocation methods, which can optionally be mesh-free. The primary inputs to the network are the coordinates of the sample points in the reference configuration, while the output is trained to be the solution field of the PDEs evaluated at the collocation points [18].
As described in Section 1.2, the network is trained to find the optimal network parameters by minimizing a loss function. In this context, the loss function

is based on the PDE residuals, which quantify the error between the approximated solution and the exact solution that satisfies the governing equations. Specifically, the loss function is built by summing the residuals of the PDEs evaluated at the collocation points and the objective is to minimize the error point by point, locally approximating the exact solution [18, 27].
Derivatives required in the loss function are commonly calculated using the autograd functionality. This technique enables precise and fast calculation of gradients, even in more complex cases.

2.1.1 Treatment of boundary conditions

Dealing with a forward problem, there are two different methods to enforce boundary and initial conditions: a "soft" enforcement and a "strong" one [18]. The weak enforcement [28] requires the addition of penalty terms to the loss function, which has the role of pointing up deviations from the prescribed boundary and initial conditions.
The loss function to be minimized, in the most general case, is made up of three terms, as outlined by the following expression:

$$\mathcal{L} = \mathcal{L}_g + \lambda_1 \mathcal{L}_{BC} + \lambda_2 \mathcal{L}_{IC} \tag{2.1}$$

where

- $\mathcal{L}_g$ represents the residual loss coming from the resolution of the governing equations;
- $\mathcal{L}_{BC}$ represents the boundary value losses. They are defined as the sum of the mean square error of the deviation on the Dirichlet and Neumann boundaries, as outlined by

$$\mathcal{L}_{BC} = \|u - B_D\|^2_{\partial\Omega_D \times [0,T]} + \|n \cdot \sigma - B_N\|^2_{\partial\Omega_N \times [0,T]} \tag{2.2}$$

  where u and σ come from the neural network's approximated solution;
- $\mathcal{L}_{IC}$ represents the initial value losses, which are defined as

$$\mathcal{L}_{IC} = \|u - I_0\|^2_{\Omega \times \{t=0\}} + \|u_t - I_1\|^2_{\Omega \times \{t=0\}} \tag{2.3}$$

- λ_1, λ_2 are positive weighting coefficients. Usually, they are determined following a trial-and-error procedure to guarantee correct enforcement, which is not always so accurate due to pathology issues of the gradient. From here, the attribute "soft".

In Fig.2.1 a schematic representation of soft enforcement is provided.

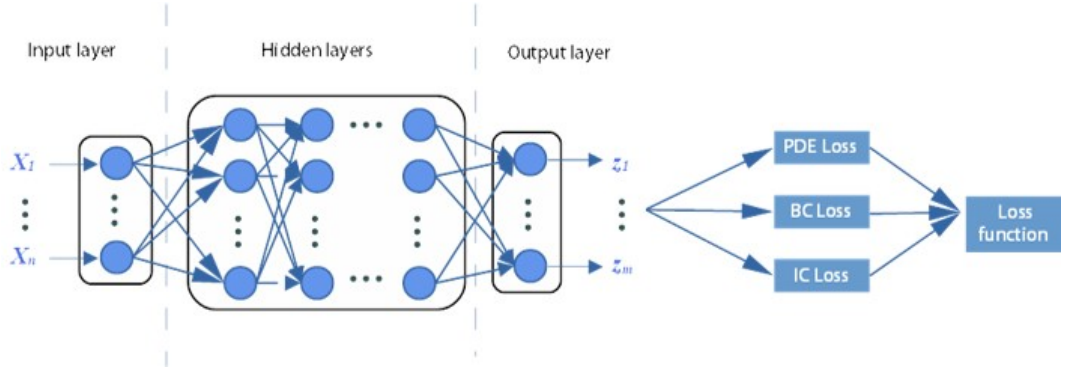


Figure 2.1: Schematic representation of PINNs soft enforcement.

The strong enforcement method [18] requires the imposition of the conditions in a "strong" form, which provides the predicted solution automatically satisfies the requirements thanks to an adaptation of the network's outputs.
In this last case, specific functions are designed and used to modify the network's outputs, allowing compliance with the imposed conditions.
In Fig.2.2 a schematic representation of strong enforcement is provided.

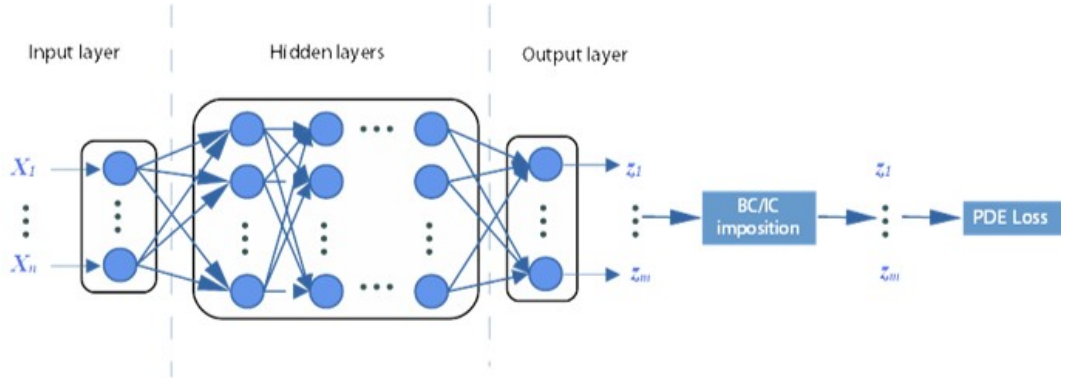


Figure 2.2: Schematic representation of PINNs strong enforcement.

As already described, the loss function is given by the strong form of the PDE residual, but in some problems, PINNs fail to predict the correct solution. In these cases, such as problems with low regularity of data, the solution has sense only in a variational form [29].

2.2 Variational Physics-Informed Neural Networks

Variational Physics-Informed Neural Net- works extend the concept of classical PINNs by combining machine learning techniques with the variational approach [30].
Unlike PINNs, where PDEs are included in the architecture of neural networks

in a strong form, this version incorporates the variational formulation of the PDEs in the loss function.
To obtain the variational form, known as the weak form, terms of the governing equations are multiplied by properly selected test functions and then are integrated over the whole domain.
The test functions are represented by a set of known functions and quadrature points are used to compute the integrals involved [31]. To evaluate them, some numerical integration techniques can be employed, such as the Gauss quadrature, as described in Section 3.3.
Instead of computing the residual loss at specific points of the domain given as input to the network, the loss function is defined by the integral of the residual over the entire domain.
This difference provides some advantages [30, 31]:

- The domain can be decomposed into many sub-domains, each of them characterized by a separate number of test functions, leading to a more elemental flexible learning approach;
- The integration involved in the weak form requires less regularity of the approximate solution;
- The number of quadrature points is smaller than the number of collocation points required by PINNs.

The loss function takes into account both the weak residual and terms for the boundary/initial conditions. During training, the loss function is minimized to satisfy the variational formulation, reducing the global error and allowing the model parameters to be updated.

2.3 Deep Energy Method

The deep energy method is an innovative approach to approximate the solution of partial differential equations [18].
It can be seen as an extension of classical PINNs, since instead of shutting down the residual losses coming from the resolution of PDEs, this method minimizes the potential energy of the system, which is taken as a loss function. Notice that this approach, due to its intrinsic definition, is only applicable to physical systems that fulfill the principle of minimum energy, such as in mechanics problems and other physical applications where potential energy is a natural quantity to be minimized.
The boundary conditions can be imposed both by adding penalty terms to the loss functions or through strong enforcement [18]. However, the second approach leads to an unconstrained optimization problem, since the researched solution already satisfies the boundary conditions and no other restrictions are needed [20]. This implies that solely the potential energy has to be minimized

and, consequently, it's easier to estimate the solution.
The process of DEM is depicted in Fig.2.3.

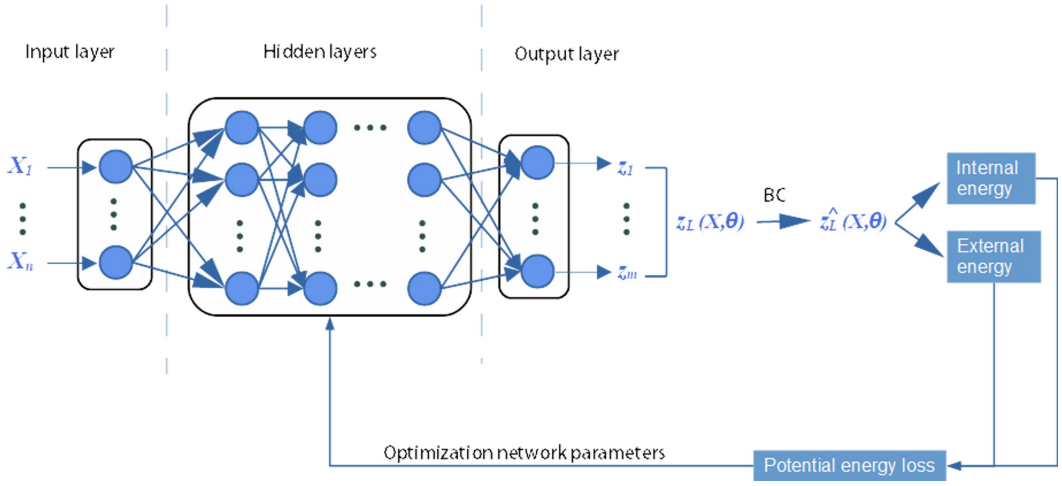


Figure 2.3: Overall process of Deep Energy Method (adapted from [20]).

The initial points [18, 21] are given as input to a neural network, usually a feed-forward fully connected neural network, whose number of neurons and layers depends on the specific application.
After data is processed by all the layers, an approximate solution $z_L(X;\theta)$ is obtained.
At this stage, the Dirichlet boundary conditions are imposed in a strong way, modifying the output $z_L(X;\theta)$ so that

$$\hat{z}_L(X;\theta) = A(X) + B(X) \cdot z_L(X;\theta) \tag{2.4}$$

where $A(x)$ and $B(X)$ are, respectively, a particular solution and a distance function.
The distance function $B(X)$ is designed to be 0 on the Dirichlet boundary, whereas the particular solution $A(X)$ satisfies the Dirichlet boundary conditions.
Notice that Neumann boundary conditions do not have to be directly enforced, as they are already accounted for in the potential energy through the external work.
The total potential energy of the system is expressed, indeed, as the sum of the internal and external energy [18]

$$\Pi = \Pi_i + \Pi_e \tag{2.5}$$

and it is used as the loss function to be minimized

$$\mathcal{L} = \Pi \tag{2.6}$$

Calculation of both energy components requires solving an integral, which depends on the approximate solution $\hat{z}_L(X;\theta)$. Since the latter is known only

at the points passed as input, a numerical integration scheme has to be applied. This will be better analyzed in section 3.3.
Finally, the unconstrained optimization problem [21] is defined, in accordance with Eq.(1.51), as

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta) \tag{2.7}$$

The optimization process leads to identifying the optimal network parameters θ and it is performed using numerical techniques such as the optimizers Adam or L-BFGS.
When this occurs, the process reaches convergence and the neural network can provide the approximate solution to the problem governed by the PDEs.

2.4 Kolmogorov-Arnold Neural Networks

The Kolmogorov-Arnold Neural Network (KAN) is a promising alternative to the conventional neural network, first proposed by Liu et al. in [32].
It is inspired by the Kolmogorov-Arnold representation theorem, which establishes that any multivariate continuous function $f(x)$ can be decomposed into sums of univariate continuous functions $\phi_{p,q}$ and ϕ_p.
Formally, it is expressed as

$$f(x) = f(x_1, x_2, \ldots, x_n) = \sum_{q=0}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{p,q}(x_p) \right) \tag{2.8}$$

Like fully connected feed-forward neural networks, also KANs have a fully connected structure, but a key difference exists between these two classes: instead of having fixed activation functions on neurons, learnable activations functions are applied on edges [32].
This difference provides greater flexibility to the network, as activation functions are not defined in advance as hyperparameters.
Thanks to that, predictions can closely align with exact solutions, capturing better the behavior described by governing equations.
Unlike traditional deep neural networks, nodes simply sum incoming signals without introducing any non-linear transformations. Instead, edges incorporate learnable activation functions, which are a combination of a basis function and a B-spline [33].

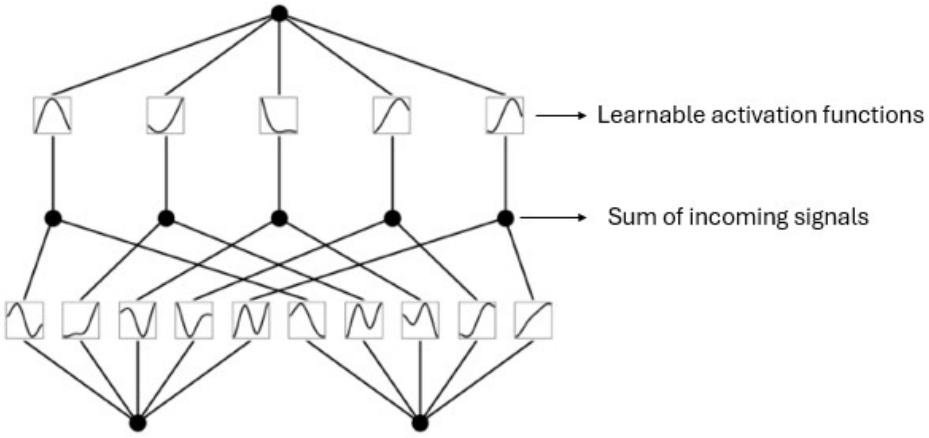


Figure 2.4: Schematic representation of the KAN architecture (adapted from [32]).

During the training process, these spline-based activation functions are optimized to match the target solutions .
Thanks to its structure, fewer parameters and a smaller network size are sufficient to get acceptable results.
It is possible to apply this architecture to solve partial differential equations similarly to PINNs/V-PINNs, just replacing the traditional artificial neural network with a KAN architecture [33]. By making the physics loss as small as possible, the neural network solution adheres to the physical laws represented by the governing equations.

Chapter 3

Methodology

The first part of this chapter describes the methodology for coupling the phase-field model with the deep energy method, as both methods are linked to the total energy balance of the system. For this reason, this approach seems to be well-suited for performing fracture studies using neural networks. Then, two possible typologies of domain discretization are discussed: the first option consists of using collocation points, while the second one adopts elements and nodes. Finally, since solving the governing equations requires the computation of some integrals, the main integration techniques are introduced, including the Newton-Cotes methods and Gauss quadrature.

3.1 DEM for Phase-Field Simulations

As described in Section 1.1.3, in the phase-field method, crack evolution is described by a set of coupled governing equations involving the damage variable and the displacement field.
These equations are based on the energy balance approach that considers elastic energy, fracture energy, and the external work[13].
Since DEM is a physics-informed approach that assumes the potential energy associated with a system as a loss function, it can be applied to solve the phase-field model by integrating the governing equation into the network architecture and minimizing the total energy of the system, as done by Goswami et al. in [22].
To describe crack propagation, it is necessary to incrementally apply an imposed displacement (or load) for n time steps until failure occurs.
At each step, the neural network is trained for a predefined number of epochs to approximate the displacement and damage fields. Treating each time step as a quasi-static condition, the network parameters are optimized using an optimization algorithm. Then, the crack evolution is determined by updating the loading condition, and the training for the new loading condition is performed.
Resuming the equations of the phase-field approach previously described, the

problem statement can be written as

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\theta) \tag{3.1}$$

where

$$\begin{aligned} \mathcal{L}(\theta) &= \Psi_e + \Psi_{frac} + \Psi_{ext}, \\ \Psi_{frac} &= \int_\Omega \left[\frac{G_c}{2} \left(\frac{d^2}{\ell} + \ell |\nabla d|^2 \right) + g(d) H(x,t) \right] d\Omega, \\ \Psi_e &= \int_\Omega \left[g(d) \Psi_e^+(\boldsymbol{\epsilon}) + \Psi_e^-(\boldsymbol{\epsilon}) \right] d\Omega, \\ \Psi_{ext} &= -\int_\Omega b \cdot u \, d\Omega - \int_{\partial\Omega_N} T \cdot u \, dS \end{aligned} \tag{3.2}$$

The strain history variable $H(x,t)$ can be used to model preexisting cracks in the system or to initialize crack propagation. In particular, considering l as a line representing the discrete crack, the initial value $H(x,0)$ could be defined as a function of the closest distance from any point x to the line l [12].
This is expressed by

$$H(x,0) = \begin{cases} \frac{BG_c}{2l_0} \left(1 - \frac{2\,\mathrm{dist}(x,l)}{l_0} \right), & \text{if } \mathrm{dist}(x,l) \leq \frac{l_0}{2}, \\ 0, & \text{if } \mathrm{dist}(x,l) > \frac{l_0}{2}. \end{cases} \tag{3.3}$$

where B [22] is a scalar parameter that controls the magnitude of the scalar history field and is defined as

$$B = \frac{1}{1-d}, \quad \text{for} \quad d < 1 \tag{3.4}$$

By tuning it, it is possible to control the intensity of the initial damage field. Through this initialization, initial cracks can be located anywhere in the domain. After the generation of the training points, the neural network architecture is set up as follows:

- The number of input neurons is equal to the dimension of the spatial coordinates of the training points;
- The number of hidden layers and the corresponding number of neurons is tuned according to the problem;
- The number of output neurons coincides with the components of the displacement field, plus one to take into account the damage variable d.

Giving the training point as input, the network provides an approximate solution for the displacement and damage fields. In this work, the neural network outputs are modified to correctly impose the Dirichlet boundary conditions, employing strong enforcement.
To obtain the energy density terms to compute the correct network parameters,

as stated in Eq.(3.2), the gradients of the displacement and the damage field are evaluated.
The strains are computed as follows

$$\varepsilon_{xx} = \frac{\partial u}{\partial x} \tag{3.5}$$

$$\varepsilon_{yy} = \frac{\partial v}{\partial y} \tag{3.6}$$

$$\varepsilon_{xy} = \frac{1}{2}\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right) \tag{3.7}$$

To get the most general form, the elastic strain energy density is decomposed into tensile and compressive parts, through the eigenvalues of the strain tensor [22].

$$\Psi_e^+ = \frac{\lambda}{8}(\lambda_s + |\lambda_s|)^2 + \frac{\mu}{4}\sum_{i=1}^{n}(\lambda_i + |\lambda_i|)^2 \tag{3.8}$$

$$\Psi_e^- = \frac{\lambda}{8}(\lambda_s - |\lambda_s|)^2 + \frac{\mu}{4}\sum_{i=1}^{n}(\lambda_i - |\lambda_i|)^2 \tag{3.9}$$

where $\lambda_s = \sum_{i=1}^{d} \lambda_i$ and n corresponds to the problem's dimension. λ and μ are respectively the Lamé constants, defined as

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \tag{3.10}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{3.11}$$

At the same time, $H(x,i)$ is obtained as the maximum value between its value at the previous step and the current value of Ψ_e^+ just calculated

$$H(x,i) = max[\Psi_e^+, H(x,i-1)] \tag{3.12}$$

To ensure crack irreversibility, $H(x,i)$ is updated at each step and used as the initial value for training the neural network at the next load step [22].
Then, finally, the fracture energy density is computed.
Since the values calculated so far refer to the input points and not to the overall system, the integration over the whole domain has to be carried out.
Several options exist to approximate the integrals, depending on the type of inputs and the problem's purposes.
The most common numerical integration techniques are analyzed in Section 3.3.
Once the global energy is known, the optimizer minimizes the loss function and performs the training through backpropagation.

3.2 Discretization

Depending on the complexity of the geometry of the problem being treated and the study's purpose, either mesh-free discretization or the generation of elements consisting of nodes and edges can be adopted.
In the context of mesh-free methods, DEM uses collocation points as input for the neural network [18].
Collocation points are a set of spatial and/or temporal coordinates where, in the context of PINNs and VPINNs, it is imposed that the governing equations characterizing the physics of the problem must be satisfied [28].
Since in the present study, the solution must describe fracture propagation according to a quasi-stationary approach, only spatial discretization is necessary.
In general, it is possible to divide the collocation points into two categories:

- Interior points: they are distributed within the domain where the solution is to be evaluated;
- Boundary points: they are located along the edges of the domain and allow the imposition and compliance with Dirichlet and Neumann boundary conditions.

Collocation points can be randomly placed or uniformly distributed in space, following a regular and equispaced pattern.
An example of discretization with uniform collocation points and boundary conditions is given in Fig.3.1.

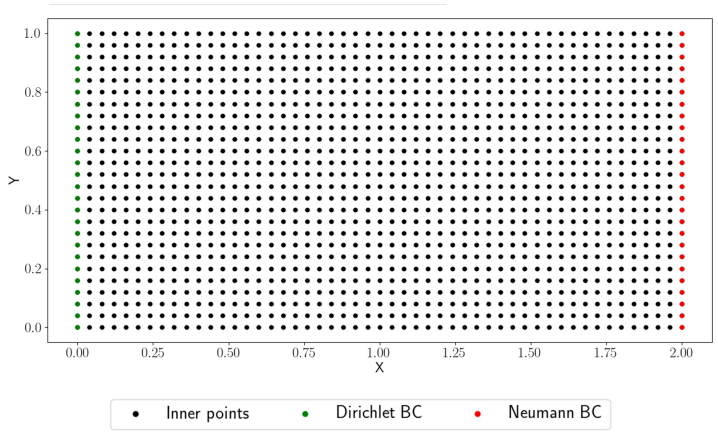


Figure 3.1: Discretization with uniformly distributed collocation points and boundary conditions.

A second possibility is to discretize the model by generating a mesh consisting of elements that are as regular as possible, as is done in the Finite Element Method (FEM).
The neural network receives as input the nodes where the solution is calculated.

Then, through shape functions, the displacement and damage fields are evaluated at Gauss points. Gauss points are specific nodes internal to the element that allow easier integration of energy quantities over the entire domain [34], as clarified in the Section 3.3.
The shape functions are polynomials that allow the interpolation of the solution at points inside the element, as expressed by the formula

$$\mathbf{u}_g^\theta(\mathbf{x}) = \sum_{i=1}^{n} N_i(\mathbf{x})\mathbf{u}^\theta, \tag{3.13}$$

$$d_g^\theta(\mathbf{x}) = \sum_{i=1}^{n} N_i(\mathbf{x})d^\theta. \tag{3.14}$$

The coefficients of the shape functions are expressed in natural coordinates, which simplifies the integration [35].
Depending on the complexity and the degree of accuracy to be achieved, either first- or second-order elements can be used. First-order elements are characterized by linear shape functions (first-degree polynomials); second-order elements are characterized by quadratic shape functions. Considering bi-dimensional quadrilateral elements, which are the ones used in one of the studies, the element's complexity can gradually increase as follows [36]:

- First-order elements: characterized by 4 nodes placed at the vertices of the square;
- Second-order elements of Serendipity type: characterized by 8 nodes. The first 4 nodes correspond to the vertices of the square, the other 4 are placed in the middle of each side
- Second-order elements of Lagrange type: characterized by 9 nodes. The first 8 are arranged like those in Serendipity type elements, the ninth is at the center of the square.

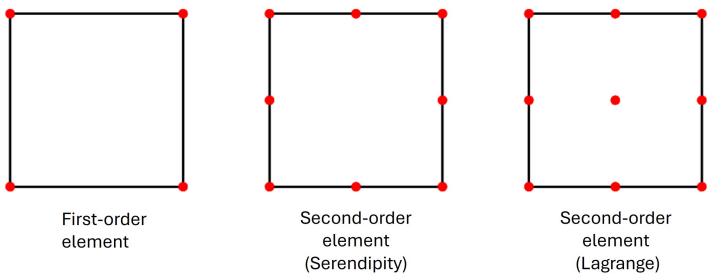


Figure 3.2: Bi-dimensional quadrilateral element types.

Although first-order elements are easy to implement and require less computational cost, they provide less flexibility and have a restricted capacity to

represent nonlinear variations.
Second-order elements allow for more accurate results with fewer elements. However, the computational times are longer due to the higher number of degrees of freedom associated with elements and the increased complexity of calculations involving shape functions [35].

3.3 Numerical Integration

The described energy-potential-based method requires the computation of integrals to estimate the overall energy of the system, as the solution is known only at the training points.
Due to its complexity, an analytical solution is not possible. To overcome this issue, numerical integration can be used allowing the resolution of PDEs.
However, since it provides an approximation of the exact values, errors are introduced and their magnitude depends on the chosen numerical method and its accuracy, as well as the number of integration points [34].
The most commonly used integration schemes are: the midpoint rule, trapezoidal rule, Simpson's rule, and Gaussian quadrature. The first three methods are part of the most generic category of the Newton-Cotes method, which can be applied to equally spaced sampling points.
Each technique is described below.

Midpoint rule The midpoint rule [18] is the simplest integration scheme and involves dividing the domain into rectangles.
Given a continuous function $f(x)$ on the interval $[a, b]$, the integral can be approximated as

$$\int_a^b f(x)\,dx \approx (b-a) f\left(\frac{b+a}{2}\right) \tag{3.15}$$

in the case of a single-point approximation, or as

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} f\left(\frac{x_i + x_{i-1}}{2}\right)\Delta x \tag{3.16}$$

when subdividing the domain into small subintervals, where $\Delta x = \frac{b-a}{n}$ is the distance between two consecutive integration points $x_i = a + i\Delta x$. This technique implies the evaluation of the function only at the midpoints, excluding the boundaries. For this reason, it is part of the Newton-Cotes open rules.

Trapezoidal rule The Trapezoidal rule [18, 34] splits the domain into trapezoids, approximating the area below the integral as

$$\int_a^b f(x)dx \approx \frac{b-a}{2}(f(a) + f(b)) \tag{3.17}$$

when using two integration points, or as

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right) \tag{3.18}$$

in the most generic case, where Δx and x_i are calculated as before. Unlike the midpoint rule, this method evaluates the function also at the endpoints of the interval, making it a closed Newton-Cotes formula.

Simpson's rule Simpson's rule [18, 34], as the trapezoidal rule, falls within Newton-Cotes closed formulas for numerical integration. The integral is approximated as

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) \tag{3.19}$$

adopting three integration points. In the more general case, it can be written as

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}\left(f(x_0) + 4\sum_{i=1,\text{ odd}}^{n-1} f(x_i) + 2\sum_{i=2,\text{ even}}^{n-2} f(x_i) + f(x_n)\right) \tag{3.20}$$

where Δx and x_i are computed as before. This numerical method produces more accurate results since the approximation is made with quadratic polynomials, but it adds complexity.

All three formulas can be applied in two dimensions by performing the integration sequentially: first in one dimension and then in the other, applying the same formula for both steps [34].

Gauss - Legendre quadrature This numerical procedure approximates the integral by a weighted sum of function values evaluated at specific interior points, combined with their corresponding weights [34]. The integration points, called Gauss points, are optimally chosen so that the greatest accuracy is achieved. In particular, they correspond to the roots of Legendre polynomials, which are equal to zero at the optimal points of the integration interval [35]. The choice of weights depends on the roots x_i to ensure the best possible approximation.
The generic formulation is expressed as

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{N} w_i f(x_i) \tag{3.21}$$

where N is the number of Gauss points and $[-1,1]$ is the standard integration interval.

If the limits of integration differ from [−1,1], a linear transformation can be used to map them to the standard integral [34].

$$x = \frac{a+b}{2} + \frac{b-a}{2}\xi \tag{3.22}$$

where ξ is a point in the standard interval. Gauss quadrature is particularly efficient since it can exactly integrate a function $f(x)$ of degree up to $2N-1$. Gauss quadrature can be used to compute integrals over elements that discretize the domain.
Considering the same quadrilateral elements from Section 3.2, given the number of Gauss points N, their positions and their corresponding weights are reported in Table 3.1 [35].

N	**Gauss points natural coordinates** (ξ, η)	**Weights** w_i
1	(0,0)	2
2	$(-\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})$	1
	$(\frac{1}{\sqrt{3}}, -\frac{1}{\sqrt{3}})$	1
	$(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$	1
	$(-\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$	1
3	$(-\sqrt{\frac{3}{5}}, -\sqrt{\frac{3}{5}})$	$\frac{5}{9} \times \frac{5}{9}$
	$(\sqrt{\frac{3}{5}}, -\sqrt{\frac{3}{5}})$	$\frac{5}{9} \times \frac{5}{9}$
	$(\sqrt{\frac{3}{5}}, \sqrt{\frac{3}{5}})$	$\frac{5}{9} \times \frac{5}{9}$
	$(-\sqrt{\frac{3}{5}}, \sqrt{\frac{3}{5}})$	$\frac{5}{9} \times \frac{5}{9}$
	$(0, -\sqrt{\frac{3}{5}})$	$\frac{8}{9} \times \frac{5}{9}$
	$(\sqrt{\frac{3}{5}}, 0)$	$\frac{5}{9} \times \frac{8}{9}$
	$(0, \sqrt{\frac{3}{5}})$	$\frac{8}{9} \times \frac{5}{9}$
	$(-\sqrt{\frac{3}{5}}, 0)$	$\frac{5}{9} \times \frac{8}{9}$
	(0, 0)	$\frac{8}{9} \times \frac{8}{9}$

Table 3.1: Gauss points and weights for bi-dimensional quadrilateral elements.

Since the integration takes place in the natural domain, a transformation of the variables must be made to map it to the physical domain or vice versa. To convert areas to the physical domain, the determinant of the Jacobian is used, as it represents the scaling factor.

In the $2D$ domain, the integral over an element can be approximated as

$$\int_{\Omega_e} f(x,y)\, d\Omega = \int_{-1}^{1}\int_{-1}^{1} f(x(\xi,\eta), y(\xi,\eta))\, |J(\xi,\eta)|\, d\xi\, d\eta \tag{3.23}$$

$$\approx \sum_{i=1}^{N}\sum_{j=1}^{N} w_i w_j f(x(\xi_i,\eta_j), y(\xi_i,\eta_j)) \cdot |J(\xi_i,\eta_j)| \tag{3.24}$$

where $|J(\xi_i,\eta_j)|$ is the Jacobian determinant [35].

Chapter 4

Phase-Field Modeling in COMSOL Multiphysics

The following chapter explains the two approaches adopted to obtain the FEM results for the subsequent validation of the neural network solutions.
First, the existing setup in COMSOL Multiphysics is described, which allows the representation of the crack through a phase-field damage attribute. Then, the methodology to manually implement the governing equations of the multi-field problem is discussed, with particular attention to the three necessary modules.

4.1 Built-in Phase-Field Setup

In COMSOL Multiphysics, it is possible to simulate fracture through a damage model that regularizes the sharp geometry of cracks by the phase-field approximation.
The Solid Mechanics interface makes it possible to analyze the response of materials, simulating their mechanical behavior under different types of loading. Through the solution of this module, the equations of motion together with a constitutive material model are solved. Thanks to this, it is possible to compute deformations, stresses, and displacements.
General structural analysis can be performed in $1D$, $2D$, and $3D$ domains, but in this discussion, the focus will be on $2D$ studies.
In the interface, for a $2D$ model, it must be defined:

- the type of $2D$ approximation: plane stress, plane strain, or out-of-plane stress;
- the thickness of the component;
- the structural transient behavior: quasi-static behavior or including inertial terms. The former is used to describe slow or steady loading conditions, while the latter considers time-dependent or dynamic loading.

There are several material model options available, such as linear elasticity, nonlinear elasticity, superelasticity, and others.
For this study, the linear elastic material is selected, assuming small deformations and isotropic behavior.
The characteristic material properties required for this model include Young's modulus, Poisson's ratio, and density. To simulate crack behavior, a damage attribute can be assigned to the linear elastic material model by selecting the damage option.
This attribute governs and describes the transition of the material from the undamaged to the broken state.
Three options are available: Scalar damage, Mazars damage for concrete, and Phase-field damage. By choosing the last alternative, several additional parameters must be defined:

- Crack driving force: the two possible alternatives are strain energy density and principal stress criterion. The first one requires the critical energy release rate and the strain energy threshold, while the second one requires the critical fracture stress and the post-peak slope parameter;
- The length scale parameter l_0;
- Damage evolution function;
- Strain energy split: the first distinction is whether to split the elastic energy between tensile and compressive parts or not. If splitting is chosen, which allows for a more faithful modeling of reality, further options include volumetric-deviatoric split, stress spectral decomposition, or strain spectral decomposition.

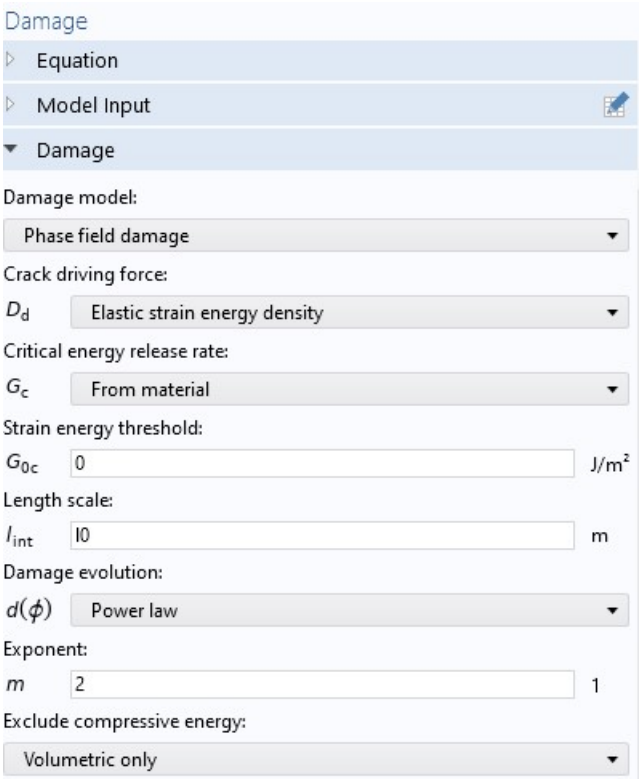


Figure 4.1: Phase-field damage attribute: required parameters.

Inside the Solid Mechanics interface, it is also required to define the boundary conditions, specifying the applied loads and constraints through appropriate attributes.
The accuracy of the results and the fracture path depends strongly on the discretization and the characteristic length l_0.
For these reasons, the mesh has to be refined in the area of assumed propagation to allow better approximation of the results.
In the areas of least interest, on the other hand, a larger mesh size can be adopted to decrease the computational costs associated with the calculation.
To solve the coupled system of equations, a segregated approach is selected, which splits the solution process into substeps.
The explanation of its functioning is better described in Section 4.2.2.

4.2 Phase-Field Model: Custom Approach

As previously described, the phase-field fracture model is a multi-field coupling problem.
To implement the governing equations in COMSOL Multiphysics, three main modules need to be defined [13]: the Solid Mechanics Module, the History Strain Module, and the Phase-Field Module. These modules allow for solving the corresponding fields, namely displacement (u), history strain variable (H),

and damage fields (d).
Then, the problem requires the solution of the coupled system of equations. This can be performed using a segregated solution approach or a fully coupled approach.
In this work, a segregated approach is adopted to obtain the solution and the quantities of interest.

4.2.1 Modules configuration and implementation

Each of the previously mentioned modules characterizes a key aspect of material behavior during the fracture process: the displacement field illustrates the mechanical response, the history strain variable guarantees the crack irreversibility condition by preventing any possible length reduction as elastic strain energy decreases, and the phase-field variable captures the degradation of the material. Details of their characterization and implementation are provided below.

Solid Mechanics As described in Section 4.1, this module enables the solution of stress and strain fields.
Since fracture problems of quasi-brittle materials are considered, the linear elastic material model is selected for this study [37].
As in the previous case, Young's modulus, Poisson's ratio, and density have to be defined, as well as the boundary conditions and whether there is a dependence or not on inertial forces.

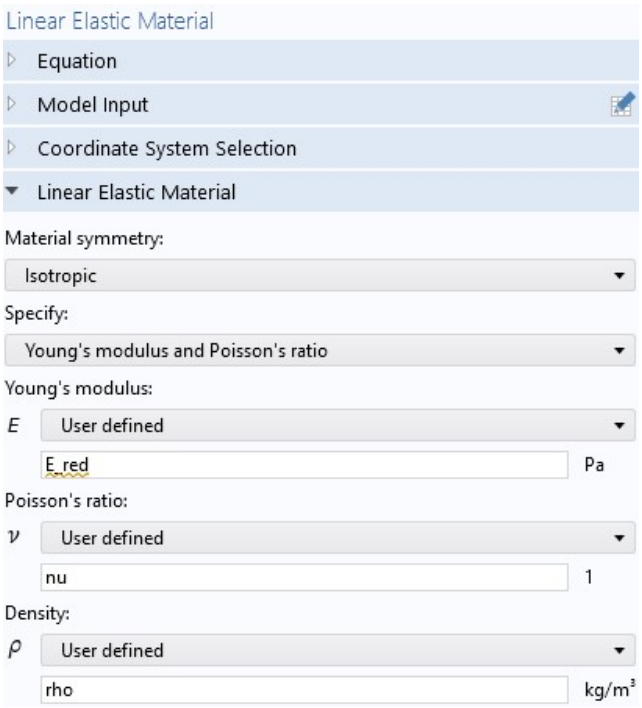


Figure 4.2: Setup linear elastic material model.

When crack propagation occurs, areas around the crack are affected by increasing damage and the material does not have anymore the same capacity to withstand external loading as an intact material.
This apparent reduction in the elastic modulus can be modeled by multiplying it by a degradation function $g(d)$, where d is the damage variable (equal to 0 for an undamaged material and 1 for a completely broken material), as reported in Eq.(4.1)

$$E_{red} = E * g(d) \tag{4.1}$$

The function $g(d)$ must be specified as a local variable within the appropriate space, since it varies at each iteration step with the progression of the damage. In this way, at each iteration step, the stiffness of the material is updated based on the evolution of the damage field.
This reflects the corresponding reduction in stored elastic energy in the damaged material, as described in Section 1.1.3.

Phase-field Module The damage field [37] can be represented by a coefficient type PDE, which has the form

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c\nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + au = f \tag{4.2}$$

where u is the dependent variable and e_a, d_a, a, c, α, β, γ are custom coefficients. These coefficients have to be defined based on the physics of the problem.
The phase-field governing equation is expressed as

$$[\frac{2l_0(1-\kappa)H}{G_c} + 1]d - l_0^2 \frac{\partial^2 d}{\partial x_i^2} = \frac{2l_0(1-\kappa)H}{G_c} \tag{4.3}$$

To achieve a correct representation of the damage field, Eq.(4.3) should be expressed in the same form as Eq.(4.2). This requires the following correspondences:

$$\begin{cases} e_a = 0,\ d_a = 0,\ \alpha = 0,\ \beta = 0,\ \gamma = 0, \\ u = d, \\ c = l_0^2, \\ a = 1 + \frac{2l_0(1-\kappa)H}{G_c}, \\ f = \frac{2l_0(1-\kappa)H}{G_c}. \end{cases} \tag{4.4}$$

where k is a numerical constant necessary for a proper conditioning of the problem. In COMSOL, this can be implemented using the Helmholtz equation

$$\nabla \cdot (-c\nabla d) + ad = f \tag{4.5}$$

where the coefficients c, a, f are the ones defined in Eq.(4.4).

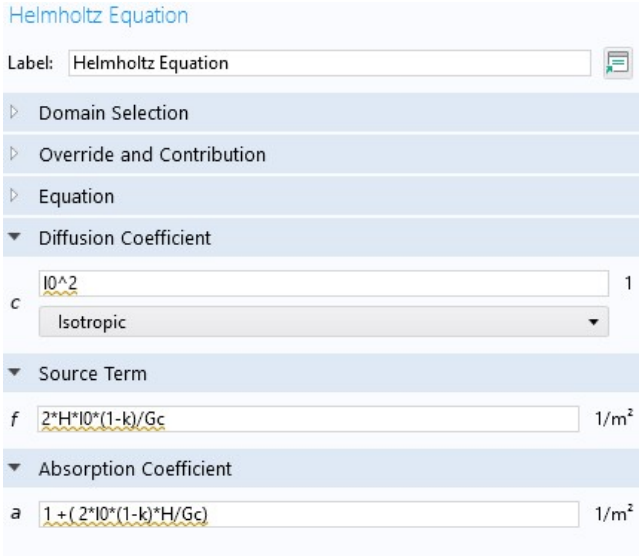


Figure 4.3: Phase-field module: governing equations and implementation.

History Strain Module The history strain variable H, as described in Section 1.1.3, is defined as the maximum tensile contribution of the elastic energy density function $H(x,t) = \max_{d\in[0,t]} \Psi_e^+$.
As outlined by its expression, its value is not constant but changes at each iteration and must be updated to reflect the evolving state of the system. The model needed to solve the history state variable is represented by the domain Ordinary Differential Equation and Differential-Algebraic Equation Module [37].
The governing equation is given by

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} = f \tag{4.6}$$

where u is the dependent variable and e_a, d_a, f are custom coefficients.
As in the previous case, these coefficients have to be defined as follows

$$\begin{cases} e_a = 0,\ d_a = 0, \\ u = H, \\ f = \begin{cases} H_{n+1} - \psi^+_{n+1}(\epsilon), & \text{if } \psi^+_{n+1}(\epsilon) > H_n, \\ H_{n+1} - H_n, & \text{if } \psi^+_{n+1}(\epsilon) \leq H_n. \end{cases} \end{cases} \tag{4.7}$$

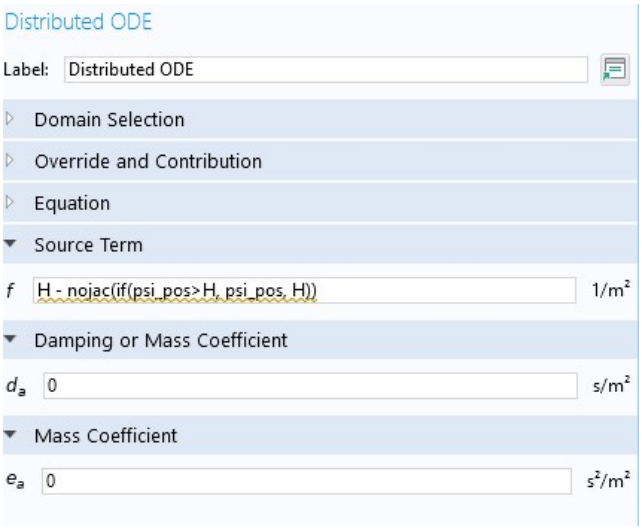


Figure 4.4: History strain module: governing equations and implementation.

The positive strain energy density has to be predefined as a local variable in the proper space, as illustrated in Fig.4.5.
Notice that its calculation requires the knowledge of the strain tensor since it is obtained as $\psi_\epsilon^+(\epsilon) = \frac{\lambda}{2}(\mathrm{tr}(\epsilon))^2 + \mu\,\mathrm{tr}(\epsilon_+^2)$. This implies that it is evaluated after the solution of the displacement field.

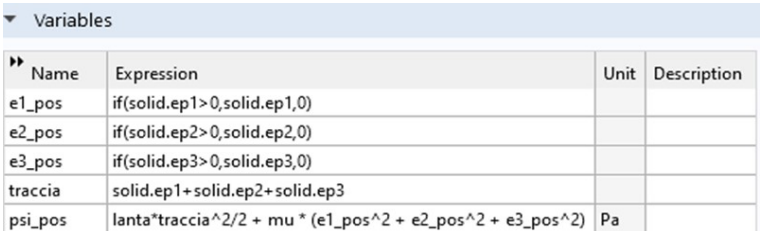


Figure 4.5: Definition of the positive strain energy density as a local variable.

The condition applied to f provides a comparison of the current value of the positive elastic strain energy with the historical maximum to guarantee the irreversibility condition for the propagation of the crack and ensure that the historical variable H will never decrease.
The nojac function is used to exclude its argument from the Jacobian matrix computation. More precisely, its argument will not be involved in the partial derivatives evaluation of the governing equations and variables [36].
In this way, the size of the computational model is reduced.
Merging the three modules, it is possible to represent and solve the coupled problem.

4.2.2 Solution scheme

The numerical solution of the model is obtained by solving the nonlinear system of equations previously described. This can be performed using two alternative approaches: a direct solution scheme and a staggered solution scheme [36].
The direct solution scheme, also known as the fully coupled approach in COMSOL Multiphysics, considers all the equations, such as the governing equation for the displacement field and the phase-field damage, as part of the same global system.
As a consequence, it solves simultaneously all the unknowns of the coupled problem within a single iteration.
Although the coupling between variables is rigorously solved, the nonlinearity of the governing equation can lead to convergence problems, and a large amount of memory is needed for each iteration.
The staggered solution scheme, instead, allows the interaction between fields without solving a global coupled system. It decouples the displacement field and the phase-field variable, subdividing the problem into different segregated steps. The equations are then solved as independent systems, using the updated solution of one field as input for the others. The solution process, considering a generic step t_i, can be summarized as follows [13]:

1. Initialize the variables u_i^j, ϕ_i^j, and H_i^j, in order to satisfy the initial conditions, where j denotes the current iteration.

2. Using the results from the previous iteration u_i^j, ϕ_i^j, and H_i^j, solve the displacement field evaluating u_i^{j+1}.

3. Substitute u_i^{j+1} into the governing equation for the history strain to evaluate H_i^{j+1}.

4. Finally, compute ϕ_i^{j+1} based on the updated values u_i^{j+1} and H_i^{j+1}.

5. Evaluate the relative error between the j and $j+1$ solutions. If it is less than an imposed tolerance, the calculation for the i step is concluded, otherwise, another iteration is performed until convergence is achieved.

6. When convergence is reached, the system moves to the next time step $i+1$.

7. The updated solutions u_i, ϕ_i, and H_i are used as initial values for iteration $i+1$.

The process just described is graphically represented in Fig.4.6.

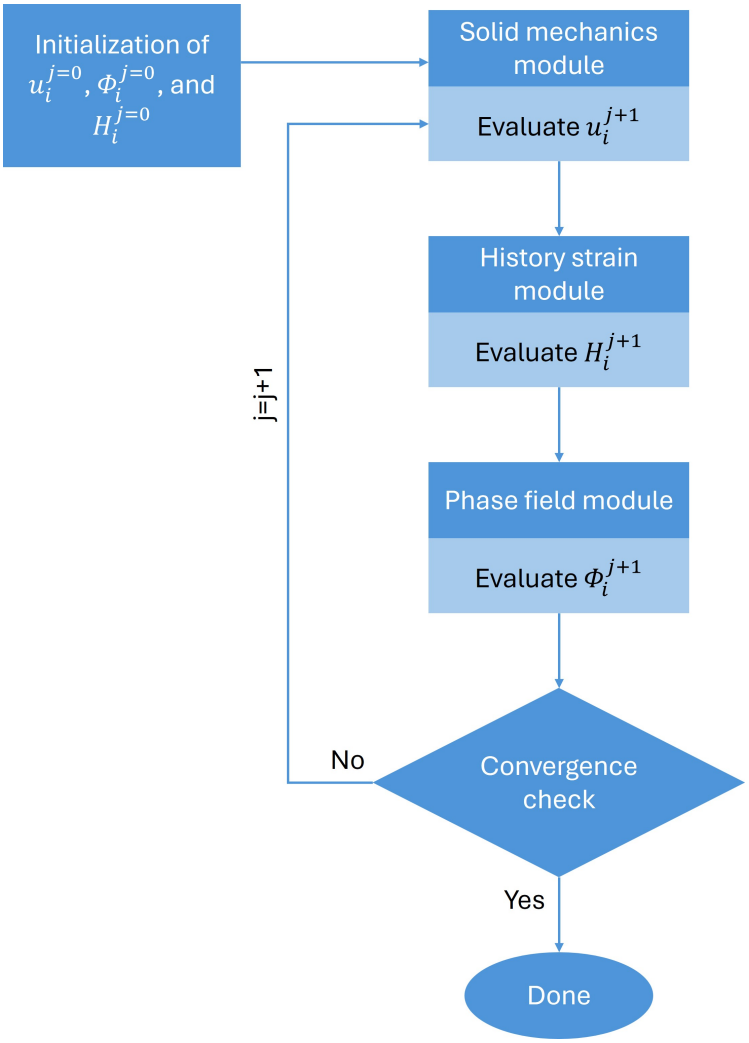


Figure 4.6: Flowchart of the segregated solution process.

As described, this approach follows an iterative scheme in which each segregated step solves only one field at a time.
The iterative convergence is slower and requires more iterations to be achieved; however, since the equations are solved separately, each iteration requires less memory [13, 36].
In this work, this latter approach is chosen as a resolution scheme.

Chapter 5

Case Studies

5.1 Single-Edge Notched Plate

The first case study consists of a single-edge notched rectangular plate subjected to tension. The initial notch extends in a vertical direction from the midpoint of the top side to the center of the plate.
The left side is clamped in both directions, while on the right side of the plate, a constant incremental displacement is applied in the positive x-direction to study crack propagation.
The final displacement is equal to $u = 0.01$ mm.
The characteristic geometric dimensions are summarized in Table 5.1, while a schematic representation of the problem is depicted in Fig.5.1.

Variable	**Symbol**	**Value**
Width	L_x	2 mm
Height	L_y	1 mm
Thickness	s	1 mm
Initial crack length	a_0	0.5 mm
Length scale parameter	l_0	0.01 mm

Table 5.1: Geometric parameters of the single-edge notched plate.

Figure 5.1: Schematic representation of the single-edge notched plate problem.

A plane strain condition is assumed since the thickness has the same order of magnitude as the other characteristic dimensions.
Moreover, LEFM assumptions are adopted, so plastic deformation at the tip of the defect is neglected.
Material properties, instead, are summarized in Table 5.2 [22].

Property	**Symbol**	**Value**
Young's modulus	E	210000 MPa
Poisson's ratio	ν	0.3
Critical release energy rate	G_c	2.7 N/mm

Table 5.2: Material properties of the single-edge notched plate.

5.1.1 FEM validation setup

For this case study, the phase-field model already implemented in COMSOL Multiphysics is adopted. A general reference for this setup can be found in Section 4.1.
The following assumptions are considered upon selecting the Solid Mechanics module for 2D models.

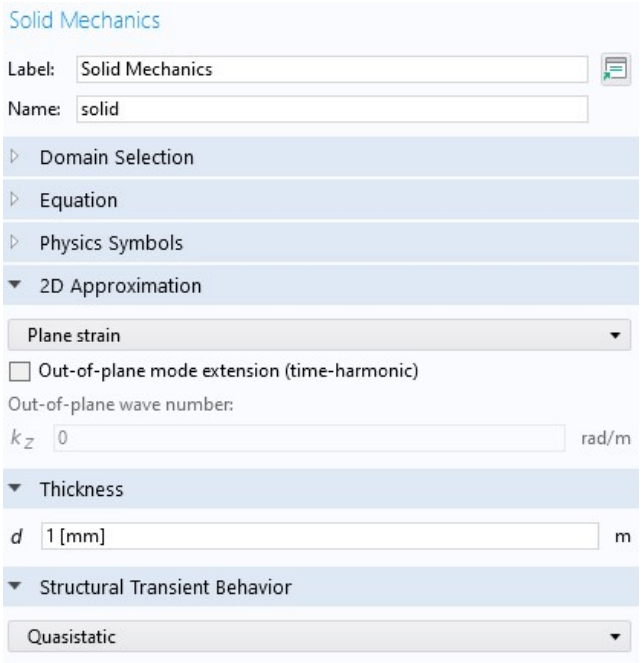


Figure 5.2: Solid Mechanics module assumptions for the single-edge notched plate.

Next, after selecting the linear elastic material model, Young's modulus and Poisson's ratio given in Table 5.2, are assigned and the phase-field damage attribute is introduced.
For this specific case, the strain energy density is used as the crack driving force.
The chosen degradation function $g(d)$ follows the formulation proposed by Bourdin et al., leading to the selection of a damage evolution power law. The needed exponent is the standard one, equal to 2, resulting in

$$f(d) = 1 - (1 - d)^m = 1 - (1 - d)^2 \tag{5.1}$$

where $g(d) = (1 - d)^2$ is the degradation function formulated by Bourdin et al. [11].

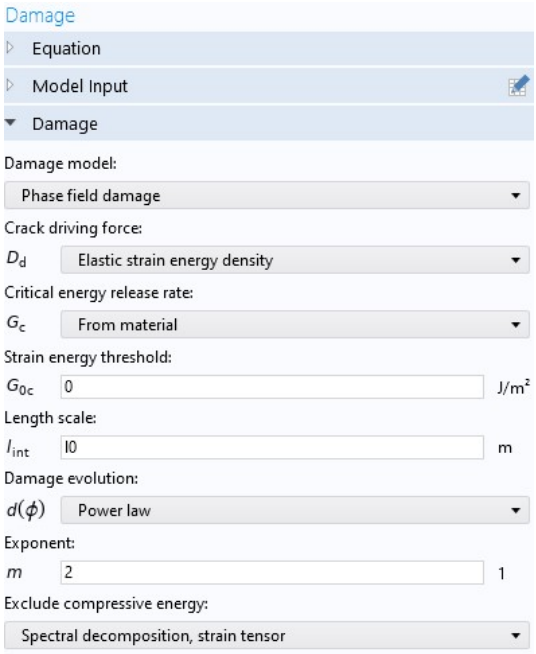


Figure 5.3: Damage attribute applied to the single-edge notched plate.

To be in line with the method used in the neural network setup for the decomposition of the elastic strain energy density, spectral decomposition of the strain tensor is applied.
Once the geometry is modeled, the boundary conditions are applied using

- Prescribed displacement: imposed on the right side of the plate. In the y direction, the displacement is set equal to zero, while in the x direction, it is applied as a function of the variable p. The displacement is updated at each iterative step through the auxiliary sweep, which allows for progressive loading by varying p from an initial value of 0 to a final value of 0.01 mm with a series of n intermediate steps.

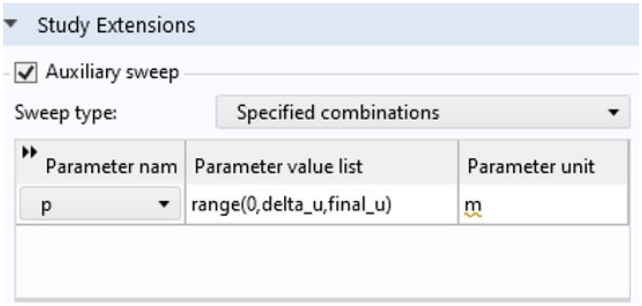


Figure 5.4: Auxiliary sweep.

- Fixed constraint: applied at the left side of the plate, whose displacement is prevented in both directions.

To properly resolve the phase-field and achieve a stable material behavior, a high mesh density is required in the proximity of the propagating crack. Since the expected crack trajectory is known in advance, the mesh is locally refined.

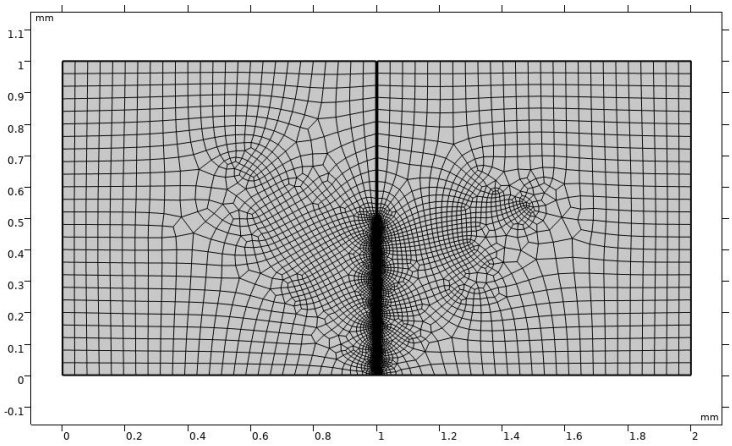


Figure 5.5: Domain discretization for the single-edge notched plate.

To achieve this, the area of interest is delimited and a mapped mesh is created to discretize it with quadrilaterals in an ordered structure. The size of these elements is chosen equal to $\frac{l_0}{4}$, and is progressively increased up to $4l_0$ in the farthest regions, which are not influenced by the fracture process.
To solve the coupled problem, the segregated strategy is adopted, splitting the evolution of the crack phase-field and the displacement field into two groups.

5.1.2 NN setup

To model the plate's geometry, a grid of collocation points is generated by dividing the domain into three vertical regions to employ different sampling densities. This approach is useful since more collocation points in the fracture area help track the crack path more accurately.
The subdivision is performed as follows:

- Left region: it is located on the left side of the pre-cracked zone and is defined in the range $[0, 0.5 \cdot L_x - \text{offset}]$;
- Central region: it is the area where the crack is expected to propagate, in the interval $[0.5 \cdot L_x - \text{offset}, 0.5 \cdot L_x + \text{offset}]$;
- Right region: it is the region located on the right side of the pre-cracked zone and is defined in the range $[0.5 \cdot L_x + \text{offset}, L_x]$

where $\text{offset} = 2 \cdot l_0$ is a distance parameter acting as a threshold for the refinement region.
Each area is initially discretized using 80×45 collocation points.

The existing crack is simulated by imposing the initial value of the strain history variable $H(x,t)$, following Eq.(3.3), using $B = 100$.
The initial setup is illustrated in Fig.5.6.

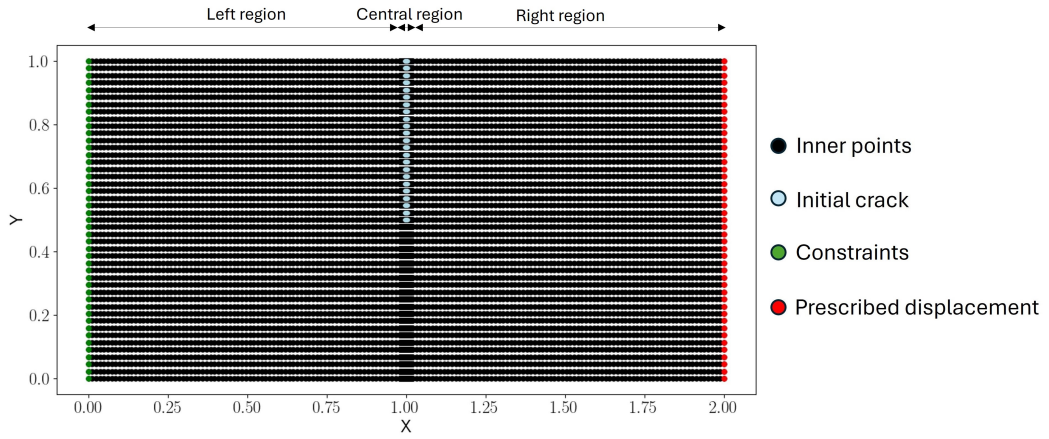


Figure 5.6: Initial grid points for discretizing the plate's geometry: three regions with different sampling densities are created.

To obtain the failure path, a fully connected neural network is employed. Its characteristics and the optimizers adopted are described below.

- 7 hidden layers;
- 50 neurons in each hidden layer;
- *Tanh* is used as an activation function for all the layers, except for the last one, which employs a linear activation;
- The Adam optimizer is adopted for the training of the first 6000 epochs, followed by L-BGFS for additional 500 epochs.

The boundary conditions that must be satisfied include

$$\begin{cases} u(x=0) = 0, \\ v(x=0) = 0, \\ u(x=L_x) = \Delta u. \end{cases} \tag{5.2}$$

To ensure these conditions, strong enforcement is preferred. The outputs of the neural network are modified as follows:

$$u = u_{NN} \cdot \left(\frac{x}{L_x}\right) \cdot \left(\frac{x}{L_x} - 1\right) + \Delta u \cdot \left(\frac{x}{L_x}\right) \tag{5.3}$$

$$v = v_{NN} \cdot \left(\frac{x}{L_x}\right) \tag{5.4}$$

where u_{NN} and v_{NN} are obtained from the neural network, x is the first nodal coordinate of each collocation point and Δu is the applied displacement.
The evaluation of fracture and elastic strain energy densities is performed by applying the formulation reported in Section 3.1.
To compute the integrals of the energy densities, the numerical trapezoidal rule is used, since it provides a good approximation without increasing computational cost as much as Simpson's rule does.
An incremental displacement is applied to simulate and describe crack propagation, dividing the final value into N steps.
For each step, a training process is performed to minimize the loss function and optimize the network parameters. At the end of each step, the displacement and damage fields are solved and the solution for that load condition is determined. The variable $H(x,t)$ is updated, as it is used as the initial value for the next step. The prescribed displacement is updated following the rule

$$\Delta u_{i+1} = n_{i+1} \cdot \delta u \tag{5.5}$$

where n_{i+1} is the current iteration steps and $\delta u = \frac{u_{final}}{N}$, and a new training process starts.
A schematic representation of the process described is depicted in Fig.5.7.

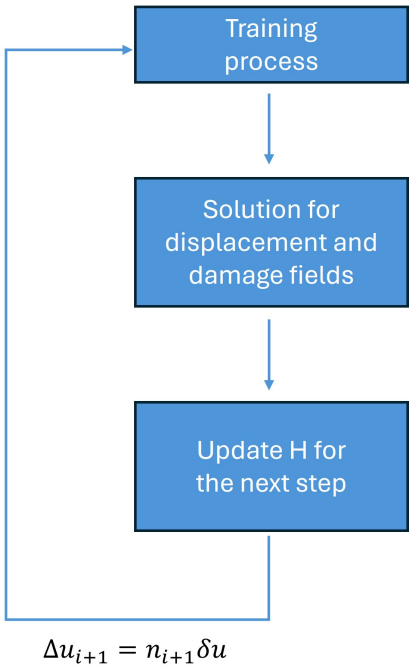


Figure 5.7: Flow chart of the computational procedure for crack propagation.

5.1.3 Results and comparisons

The accuracy and the reliability of the neural network outputs are validated through a comparison with the FEM simulation results, which served as a

reference.
Regarding crack development, the neural network successfully predicts its propagation, perfectly tracing the fracture path, as highlighted by the comparison in Fig.5.8.

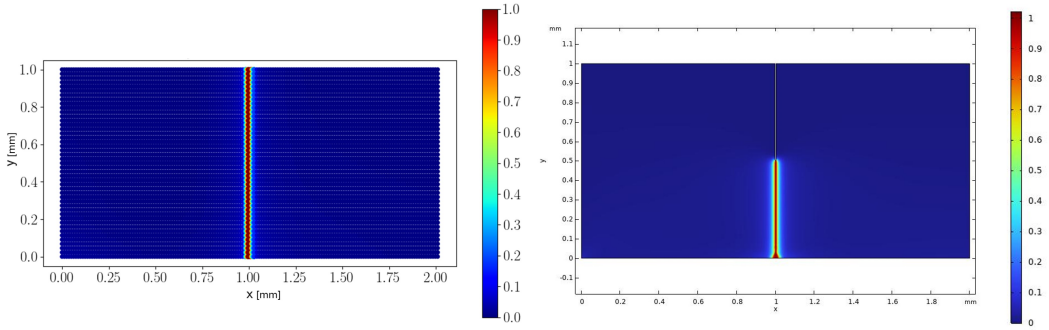
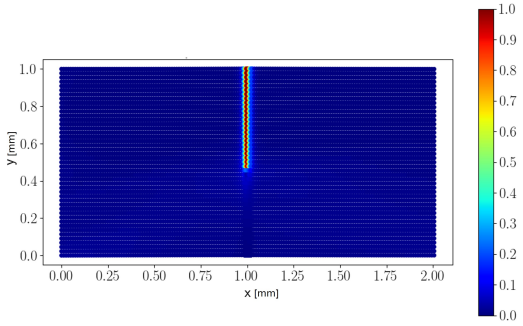
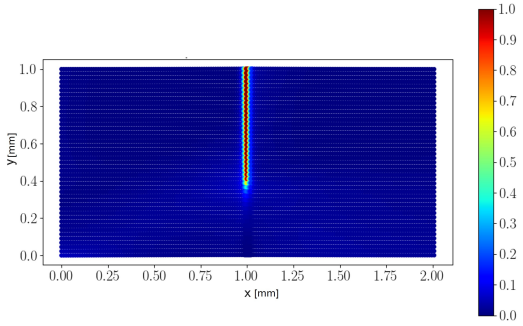


Figure 5.8: Crack path comparison - NN vs FEM results.

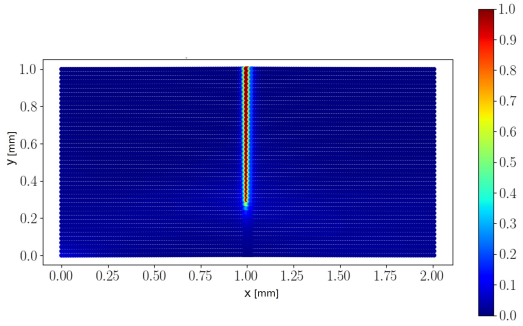
From the figure, it can be observed that the crack propagates perfectly in the vertical direction, from the center to the lower side of the plate.
Figures at various instants of load Δu are provided below to highlight how the neural network can effectively predict crack evolution.



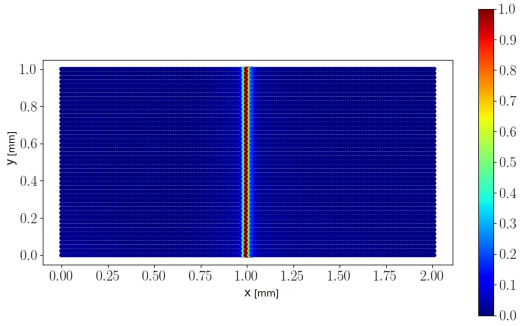
(a) $d(\Delta u = 0.006 \text{ mm})$



(b) $d(\Delta u = 0.0075 \text{ mm})$



(c) $d(\Delta u = 0.008 \text{ mm})$



(d) $d(\Delta u = 0.01 \text{ mm})$

Figure 5.9: Evolution of crack propagation in the single-edge notched plate.

The results relating to the displacement field also show good accuracy compared to those obtained with the FEM method. In particular, the shape of the displacement fields, both in the x and y direction, is consistent with that predicted by the FEM, confirming the validity of the adopted model.

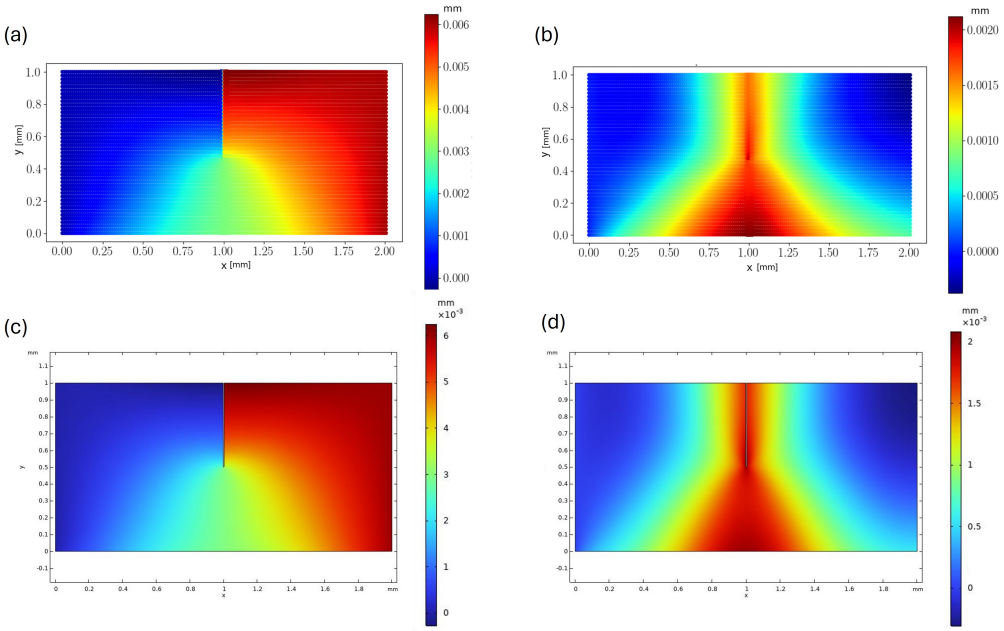


Figure 5.10: Comparison of u and v between NN and FEM for ($\Delta u = 0.006$ mm): a) u(NN). b) v(NN). c) u(FEM) d) v(FEM).

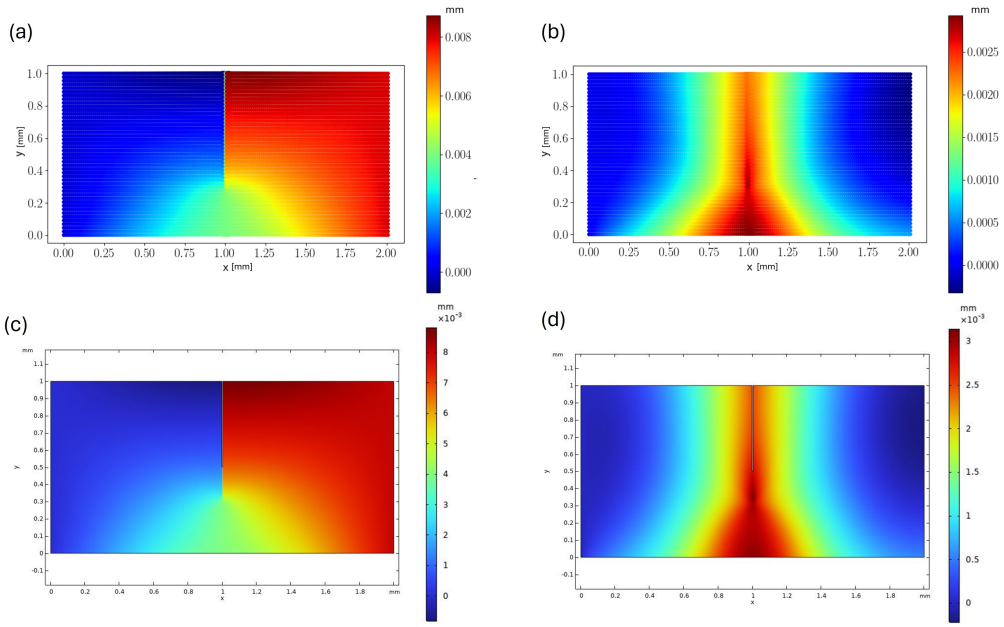


Figure 5.11: Comparison of u and v between NN and FEM for ($\Delta u = 0.008$ mm): a) u(NN). b) v(NN). c) u(FEM) d) v(FEM).

However, it is possible to highlight some discrepancies between the two models. The crack propagation predicted by the neural network occurs at a slower rate, leading to complete failure at a higher value of Δu compared to the FEM case. In particular, the complete fracture, instead of occurring at $\Delta u = 8.52 \cdot 10^{-3}$ mm, happens at $\Delta u = 9 \cdot 10^{-3}$ mm. This difference can be pointed out by analyzing, for example, the elastic strain energy as a function of the prescribed displacement in both simulations, as shown in Fig.5.12.

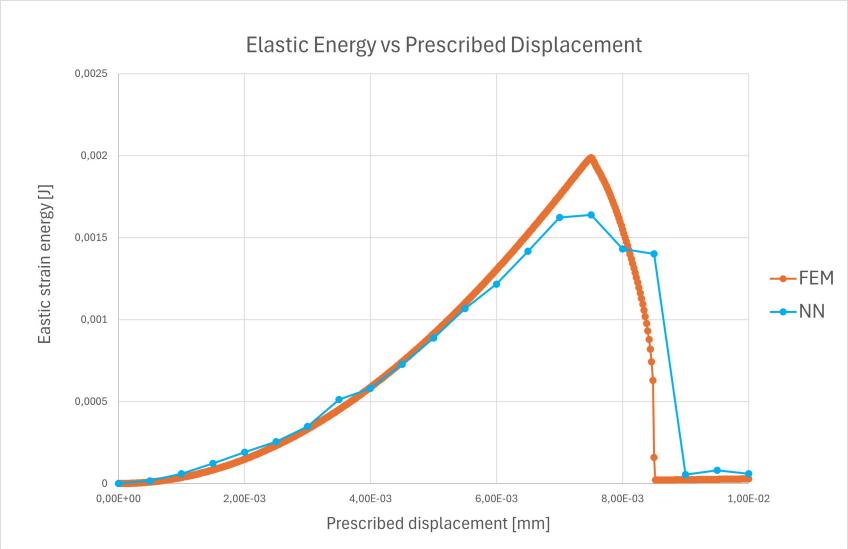


Figure 5.12: Elastic energy vs. prescribed displacement for the single-edge notched plate.

As can be seen from the graph, at the beginning the two curves match perfectly. The point of maximum elastic energy coincides with the beginning of fracture propagation. After this point, the stored elastic energy is released and, consequently, by the graph, the curve exhibits a negative slope.
Although there is a slight difference in the value assumed by the elastic energy at the moment of fracture, it can be observed that the abscissa, corresponding to the applied displacement, is the same in both simulations. Indeed, the maximum value of the elastic energy is at $\Delta u = 0.0075$ mm in both curves.
After $\Delta u = 0.008$ mm, the greatest discrepancy between the curves occurs and it can be observed that the prescribed displacement corresponding to the breaking point obtained using NN is higher than that obtained with FEM simulation.
Trying to better fit the FEM curve, a larger number of steps δu is used from $\Delta u = 0.007$ mm (which corresponds to the applied displacement value before fracture propagation in the NN model) up to $\Delta u = 0.01$ mm. In particular, instead of using $\delta u = 5 \cdot 10^{-4}$ mm, a refined step size of $\delta u = 2.5 \cdot 10^{-4}$ mm is employed in that range.
Figure 5.13 shows the elastic energy as a function of displacement for the three mentioned cases.
The FEM and NN1 curves are the ones previously illustrated, while the NN2 curve is generated by maintaining the same setup as NN1 but increasing the iteration steps as described.

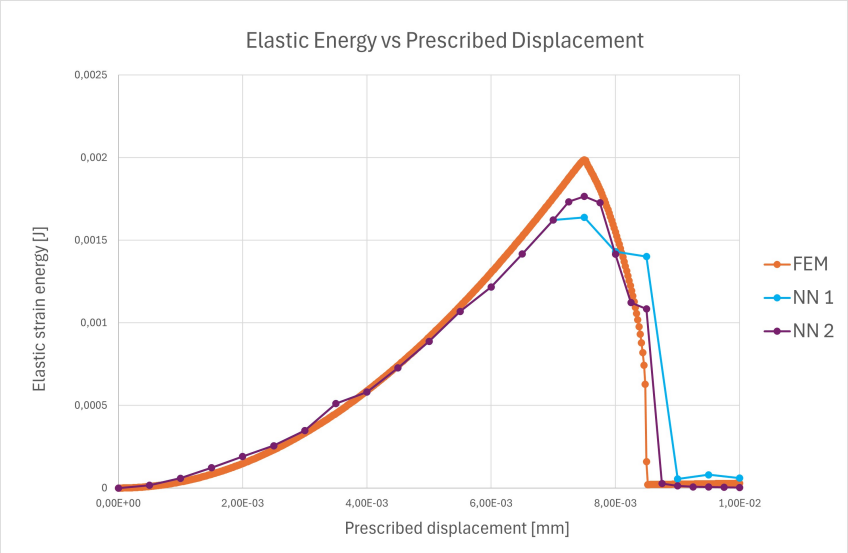


Figure 5.13: Effect of additional iteration steps on elastic energy vs. prescribed displacement in the single-edge notched plate.

As shown in the graph, the NN2 curve better approximates the FEM behavior, both in terms of the maximum value of elastic energy and after the onset of propagation.
Even the moment of complete failure of the material is closer to the FEM case, although not perfectly coincident. Indeed, in this case, the complete fracture occurs at $\Delta u = 8.75 \cdot 10^{-3}$ mm, reducing the error from 5.64% to 2.46%.
The starting point remains unchanged, but the maximum value of the elastic energy passes from $1.64 \cdot 10^{-3}$ J to $1.77 \cdot 10^{-3}$ J.
Furthermore, post-fracture behavior is also better modeled, since the horizontal line is more accurately represented.
All the comparisons are summarized in Table 5.3.

	FEM	NN1	NN2
Starting fracture point [mm]	$7.5 \cdot 10^{-3}$	$7.5 \cdot 10^{-3}$	$7.5 \cdot 10^{-3}$
Ending fracture point [mm]	$8.52 \cdot 10^{-3}$	$9.00 \cdot 10^{-3}$	$8.75 \cdot 10^{-3}$
Maximum elastic energy [J]	$1.99 \cdot 10^{-3}$	$1.64 \cdot 10^{-3}$	$1.77 \cdot 10^{-3}$

Table 5.3: Comparison of the fracture points and elastic energy values between the FEM and neural network models.

However, it is important to underline that to achieve sufficiently accurate results with the neural network, the required computational times are significantly higher than those needed for FEM simulation.
For that reason, the focus is placed on obtaining a solution that is qualitatively comparable rather than precisely accurate in terms of numerical values.

Thus, the number of collocation points was reduced to 40×35 in the lateral areas and to 60×35 in the central area, increasing the parameter offset to $3 \cdot l_0$. Additionally, the number of hidden layers was reduced from 7 to 5 and training epochs were decreased to 1200 for Adams (except for the first iteration which uses 10000 epochs) and 100 for L-BFGS.
In this way, computational times were exponentially reduced, going from 2.21 hours to 10 minutes. In this case, the characteristic shape of the curve is still captured, even if some differences are present. The starting breaking point occurs earlier than expected, as shown in Fig.5.14, as it happens at $\Delta u = 7 \cdot 10^{-3}$ mm. The peak value of the elastic energy is $1.67 \cdot 10^{-3}$ J.

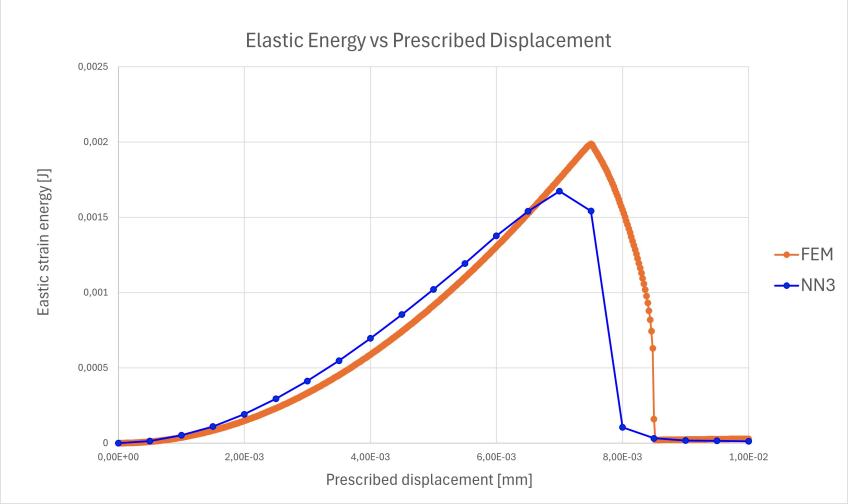


Figure 5.14: Elastic energy vs. prescribed displacement with a coarser grid.

Nonetheless, the fracture path is well identified and the shape of the displacement fields remains consistent, as shown in Fig.5.15 and Fig.5.16.

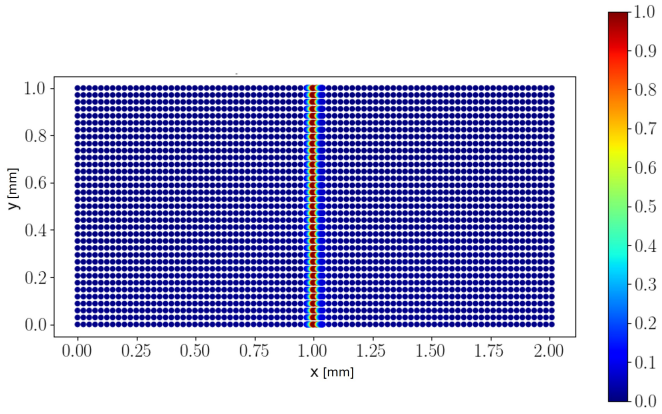


Figure 5.15: Crack path evolution using a coarser grid.

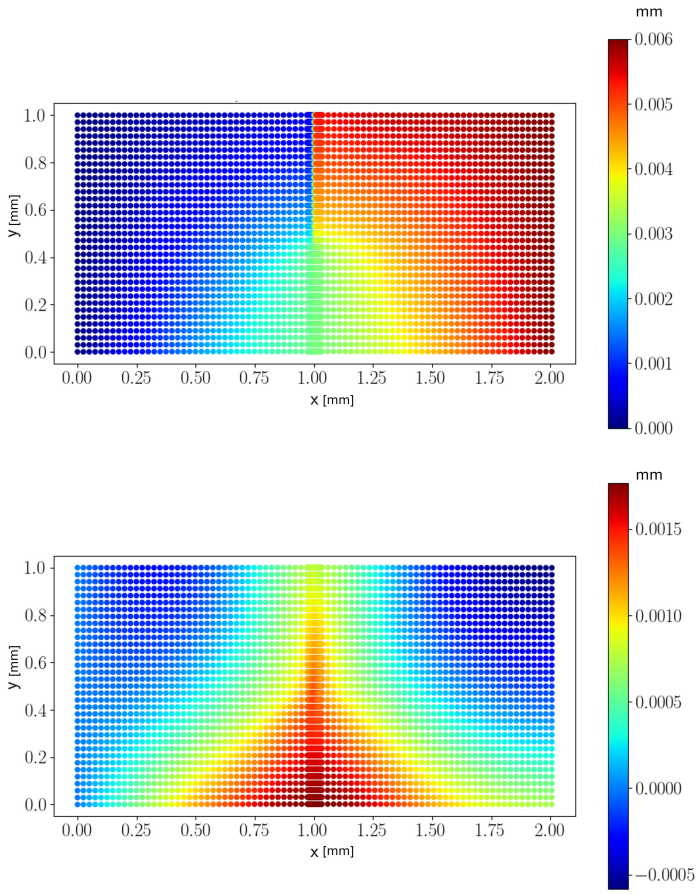


Figure 5.16: Displacement components u and v for $\Delta u = 0.006$ mm using a coarser grid.

5.2 Mixed Mode I-II Fracture in a Plate with Holes

The second case study considers a bi-dimensional plate characterized by three holes: two of the same size and placed symmetrically on the left side at the top and bottom; the third one is smaller and more centrally located, which is responsible for the propagation of fracture according to the mixed mode I-II. The lower hole is constrained in both directions, while a prescribed displacement is applied in the positive y-direction at the upper hole, as shown in Fig. 5.17.

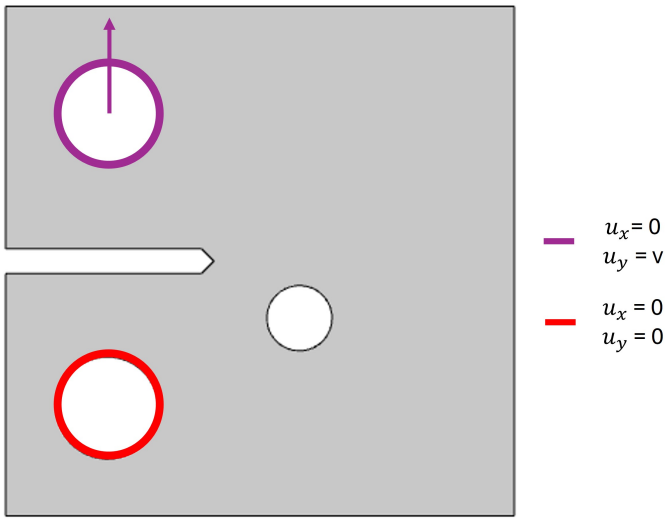


Figure 5.17: Schematic representation of the plate with holes.

The geometric dimensions are shown in Table 5.4, while material properties are reported in Table 5.5 [38].

Variable	Symbol	Value
Width	L_x	62.5 mm
Height	L_y	62.5 mm
Thickness	s	4 mm
Diameter of big holes	d_1	12.5 mm
Diameter of small hole	d_2	8 mm

Table 5.4: Geometric parameters of the plate with holes.

Since the thickness is smaller than the other characteristic dimensions, a plane stress approximation can be adopted.

Property	Symbol	Value
Young's modulus	E	72000 MPa
Poisson's ratio	ν	0.3
Yield strength	σ_s	469 MPa
Fracture toughness	K_{IC}	29 MPa $\sqrt{m}$

Table 5.5: Material properties of Aluminum 7075-T6.

As shown in Table 5.5 not all the quantities required for the study through the phase-field method are known among the material properties. Therefore, it is necessary to determine them analytically as follows

- Critical energy release rate G_c: assuming plane stress condition and LEFM hypothesis, it can be determined using the expression reported in Eq.(1.19)

$$G_c = \frac{K_{IC}^2}{E} = 15000 \frac{J}{m^2}$$

- Length scale parameter l_0: an initial estimation of its value can be made using the Eq.(1.33)

$$l_0 = \frac{27 E G_c}{256 \sigma_c^2} = 5.18 \cdot 10^{-4} \text{m}$$

  This value, by affecting the width of the transition band from broken to intact material, can lead to unrealistic fracture paths if not properly chosen. For this reason, having the experimental test and knowing the correct fracture path, it is chosen equal to $l_0 = 0.15$ mm.

5.2.1 FEM validation setup

For this case study, the phase-field governing equations are implemented manually as described in 4.2.1.
Selecting the Solid Mechanics module, the following approximations are adopted.

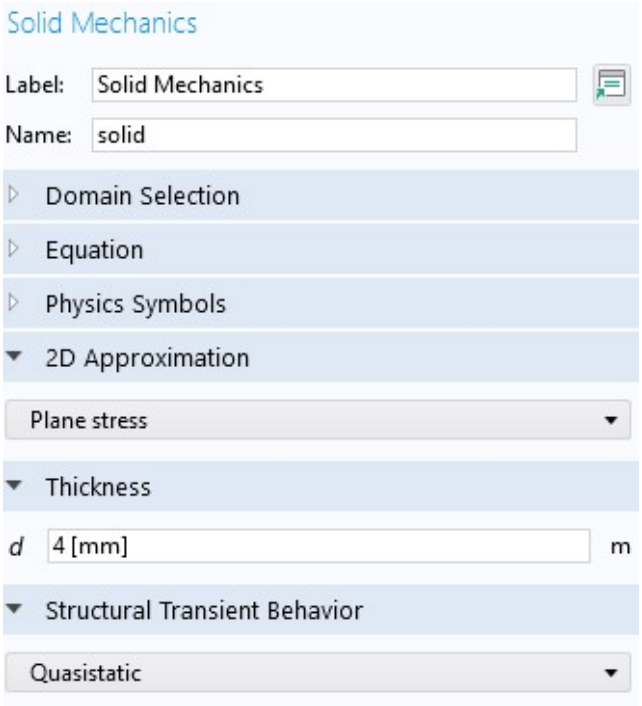


Figure 5.18: Solid Mechanics module assumptions for the plate with holes.

The degradation function is set equal to

$$g(d) = (1-d)^2 \cdot (1-k)$$

and it is defined as a local variable. k is chosen equal to 10^{-9} to avoid numerical problems.
The boundary conditions are applied using

- Prescribed displacement: imposed to the lower hole with $u_x = u_y = 0$
- Prescribed displacement: applied to the upper hole imposing $u_x = 0$ and a vertical displacement u_y variable at each iterative step with the parameter p, which varies between 0 and the final displacement through the auxiliary sweep.

For each of the three implemented modules, the initial values of the variables should be given.
Since no preexisting cracks are present, both the damage field and the history strain variable are initialized to zero, as well as the initial displacement.

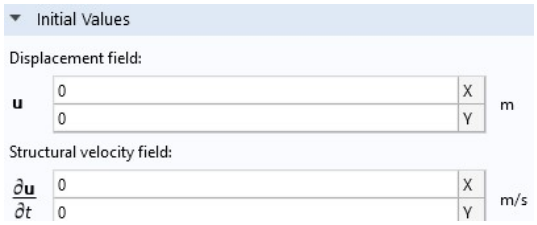


Figure 5.19: Initial displacement condition.

Figure 5.20: Initial phase-field condition.

Figure 5.21: Initial history field condition.

In the area where the crack is expected to propagate, a more regular and smaller mesh should be adopted to ensure more accurate results. For that reason, as in the previous case study, some areas have been demarcated as depicted in Fig.5.22.
Inside the highlighted areas, a quadrilateral mapped mesh is created with a maximum size of l_0, while outside a triangular mesh is chosen.

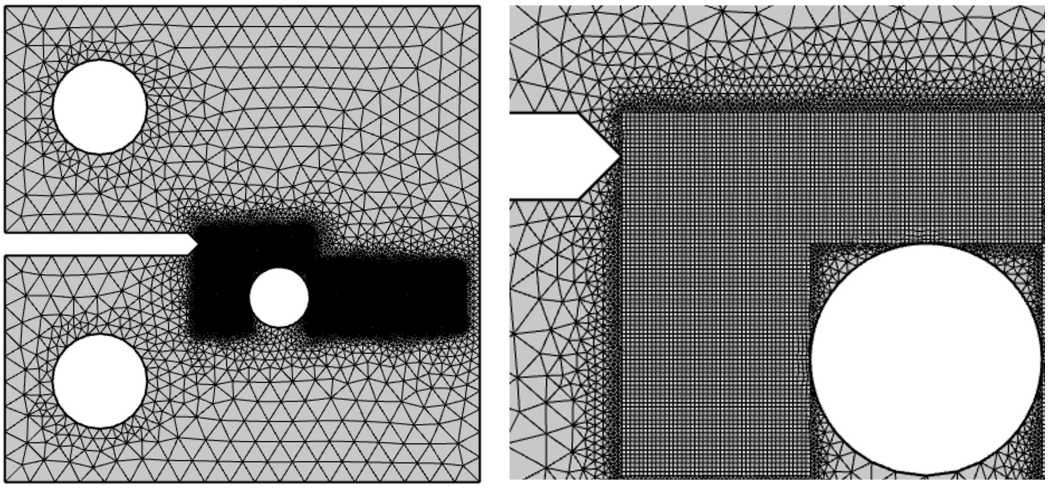


Figure 5.22: Domain discretization for the plate with holes for FEM simulation.

In FEM analyses, triangular elements do not represent the best choice as their shape functions are linear polynomials, and consequently, the deformations are constant within each element.
However, due to the presence of holes and the need to delimit the areas of potential fracture, a quadrangular mesh would be significantly distorted near the central hole.
For this reason, since the main focus is on crack propagation, a triangular mesh is adopted in the areas around the central hole. Moreover, a triangular mesh is also used in the regions of least interest, as depicted in Fig.5.22.
The staggered solution scheme described in Section 4.2.2 is adopted, solving the three modules as independent systems and using the updated solution of one field as input for the others.

5.2.2 NN setup

Due to the complexity of the geometry and to track the shape of the holes, a mesh is used to discretize the domain. In particular, first-order quadrilateral elements are chosen.
To correctly study crack propagation, a finer and more uniform mesh is generated in the region where the fracture is expected to appear. In this area, the element size is set to l_0, while in the farthest regions, it is progressively incremented to limit computational costs.
A representation of the discretized domain is given in Fig.5.23.

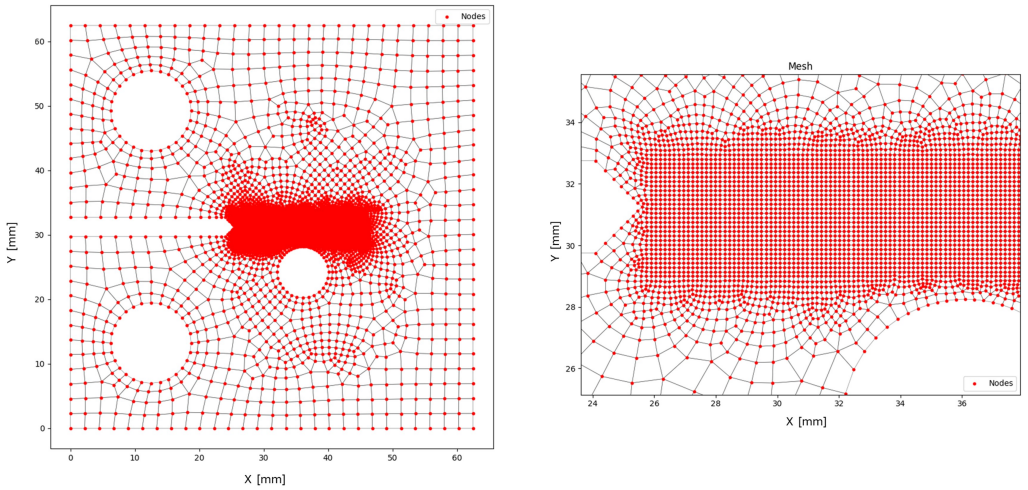


Figure 5.23: Discretization of the plate with holes using quadrilateral elements for NN simulation.

To obtain the failure path, a fully connected neural network is employed. Its characteristics and the optimizers adopted are described below.

- 7 hidden layers;
- 40 neurons in each hidden layer;
- Tanh is used as an activation function for all the layers, except for the last one, which employs a linear activation;
- The Adam optimizer is adopted with 10000 epochs for the first step and 2000 epochs for all subsequent steps.

The boundary conditions that must be satisfied include

$$\begin{cases} u(d = r_{lh}) = 0, \\ v(d = r_{lh}) = 0, \\ u(d = r_{uh}) = \Delta u \\ v(d = r_{uh}) = 0. \end{cases} \tag{5.6}$$

where d is the Euclidean distance and r_{lh} and r_{uh} refer respectively to the lower hole radius and the upper hole radius. To enforce these conditions, strong enforcement is adopted. The outputs of the neural network are modified as follows:

$$u = u_{NN} \cdot f_1 \cdot f_2, \tag{5.7}$$

$$v = v_{NN} \cdot f_1 \cdot f_2 + \Delta v \cdot z. \tag{5.8}$$

where

$$f_1 = \left(\frac{dist_{lh} - r_{lh}}{dist_{lh} - r_{lh} + \gamma}\right)^2 \tag{5.9}$$

$$f_2 = (\frac{dist_{uh} - r_{uh}}{dist_{uh} - r_{uh} + \gamma})^2 \tag{5.10}$$

$$z = \frac{f_1}{f_1 + f_2} \tag{5.11}$$

$dist_{lh}$ is the Euclidean distance from the lower hole center, $dist_{uh}$ is the Euclidean distance from the upper hole center, and γ is a parameter introduced to avoid division by zero and to control the size of the transition zone.
The nodal coordinates are passed as input to the neural network but, due to geometry dimensions which are much larger than the normalized interval [0,1], a pre-scaling of the inputs is performed.
Then, through the shape functions, the neural output values are interpolated at the Gauss points to evaluate the energy and perform the integration of the energetic quantities, using the Gauss quadrature described in Section 3.3
An incremental displacement is applied to simulate and describe crack propagation, as in the previous case, and the updating rule for the displacement is expressed by

$$\Delta v_{i+1} = n_{i+1} \cdot \delta v \tag{5.12}$$

The iteration process remains unchanged.
To test the potentiality of a KAN architecture, the same problem is analyzed by employing the fast-KAN network. The procedure remains unchanged, just the setup of the neural network changes as follows:

- 3 hidden layers;
- 18 neurons for each hidden layer;
- The Adam optimizer is employed using 10000 epochs for the first step and 2000 epochs for all subsequent steps.

5.2.3 Results and comparisons

The presence of the centered hole influenced the crack propagation, inducing a mixed mode I-II fracture.
The crack path is represented in Fig.5.24.

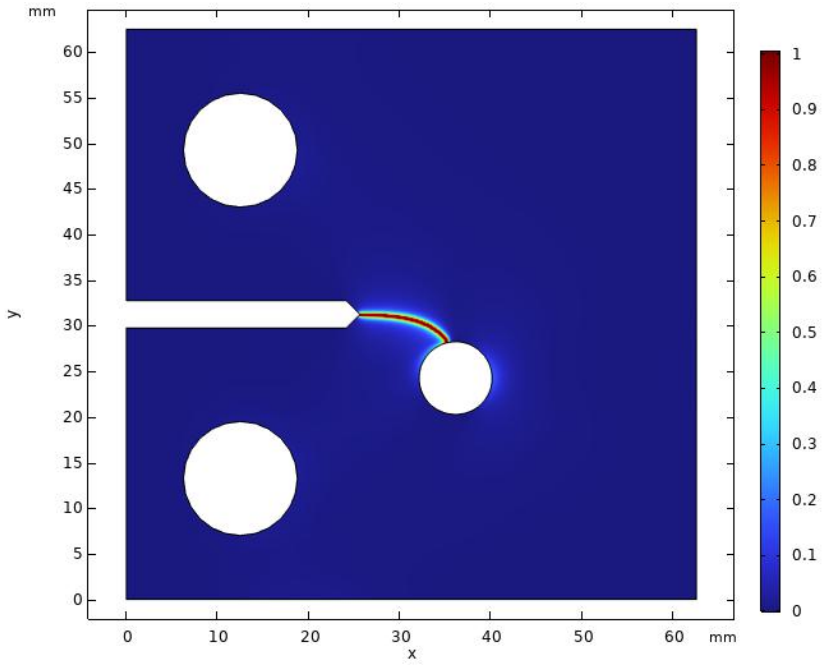
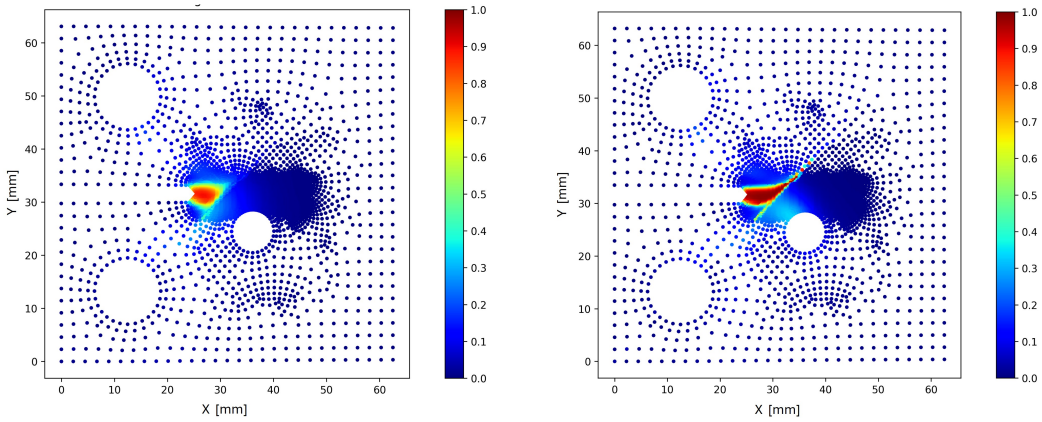


Figure 5.24: Crack path in the plate with holes from FEM results.

Despite the potential of the fast-KAN architecture, which has learnable activation functions, it is not able to accurately track the crack path in this study. As outlined in Fig.5.25a, the fracture initially starts following a horizontal propagation. However, instead of deviating towards the hole, the fracture proceeds as described in Fig.5.25b.



(a) Initial crack propagation. **(b)** Deviation in crack propagation.

Figure 5.25: Crack propagation using fast-KAN architecture.

From the distribution, a singularity in correspondence of $x = y$ is visible. Moreover, instead of exhibiting a continuous propagation of the crack, the area surrounding the hole is damaged first, as described in Fig.5.26.

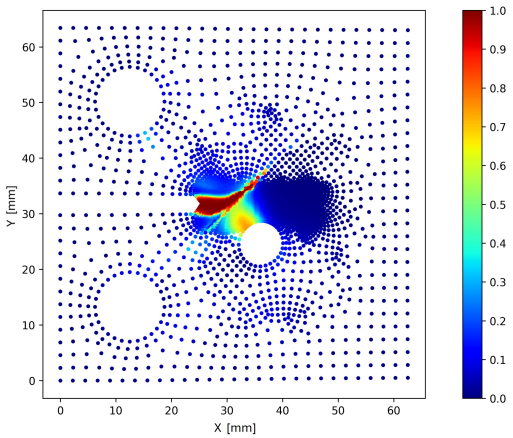


Figure 5.26: Damage distribution with an highlighting the singularity and the initiation of damage around the centered hole.

This architecture has not proven to be suitable for investigating this type of problem.
The classical fully connected neural network, instead, has shown promising results. Different tests have been carried out by varying the length scale parameter l_0 since it influences the fracture prediction. Using $l_0 = 1.5 \cdot 10^{-4}$mm, as in the FEM simulation, the crack path appears as in Fig.5.27.

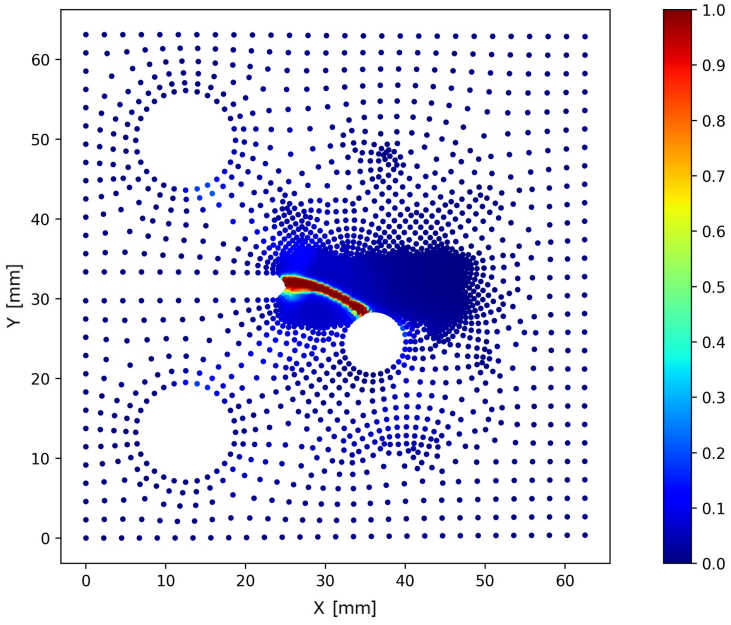


Figure 5.27: Crack path in the plate with hole using $l_0 = 0.15$ mm.

The fracture propagation is correctly tracked and the starting and final points are consistent with what is shown in Fig.5.24, even if the trajectory appears

straighter. On the other hand, choosing $l_0 = 0.1$ mm, the curvature of the fracture is better captured, as depicted in Fig.5.28.

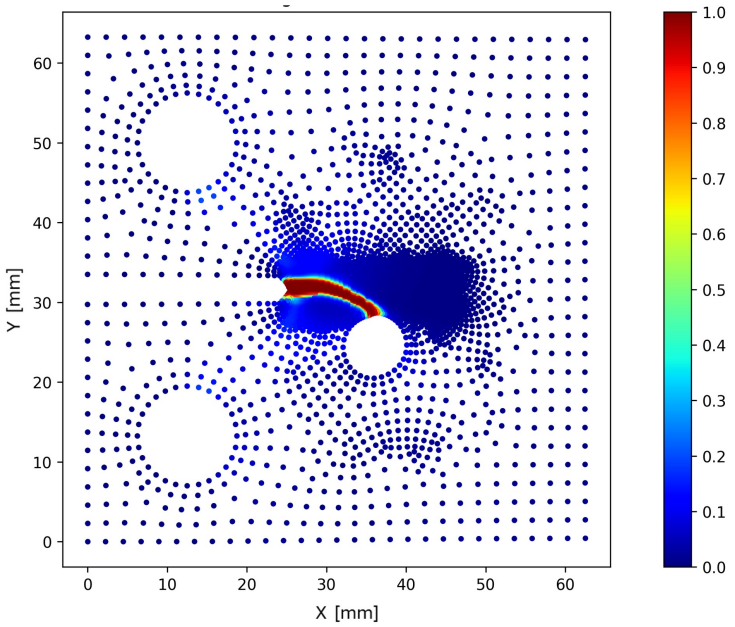


Figure 5.28: Crack path in the plate with hole using $l_0 = 0.1$ mm.

From here, it can be outlined that the phase-field method is strongly influenced by the choice of l_0: the accuracy of the solution of the resulting fracture and the shape strongly depend on that. For this reason, the selected value for the FEM simulation may not be the best one for the network's prediction.
However, the choice of l_0 doesn't only influence the shape of the crack trajectory, but also the moment in which the fracture starts.
To outline this, it is possible to check elastic energy again.

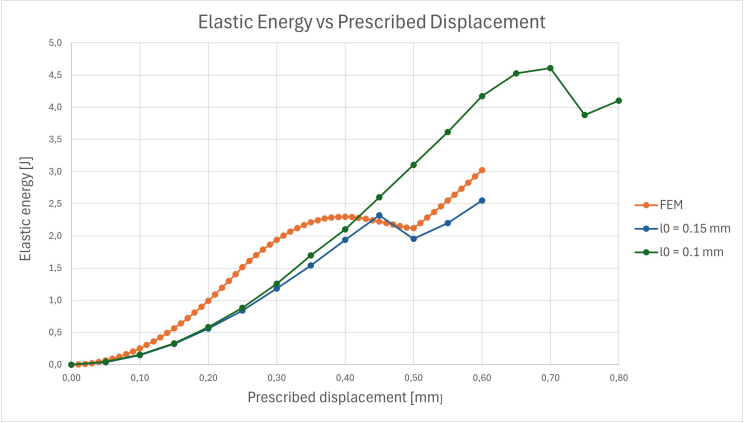
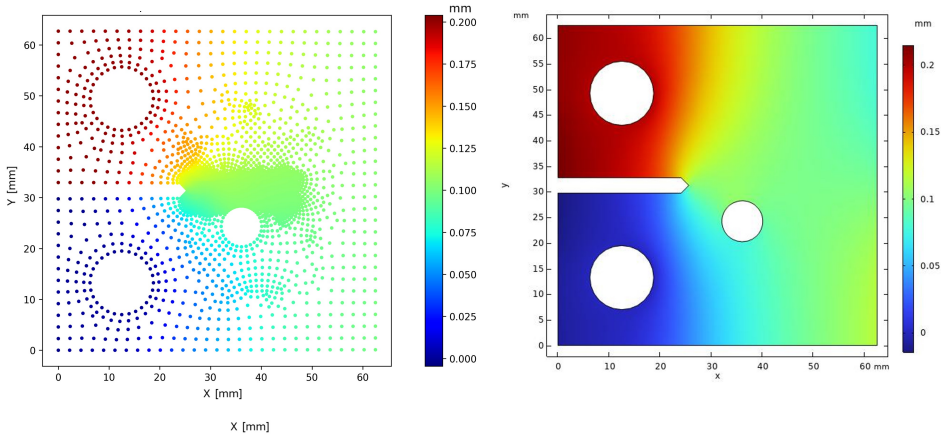
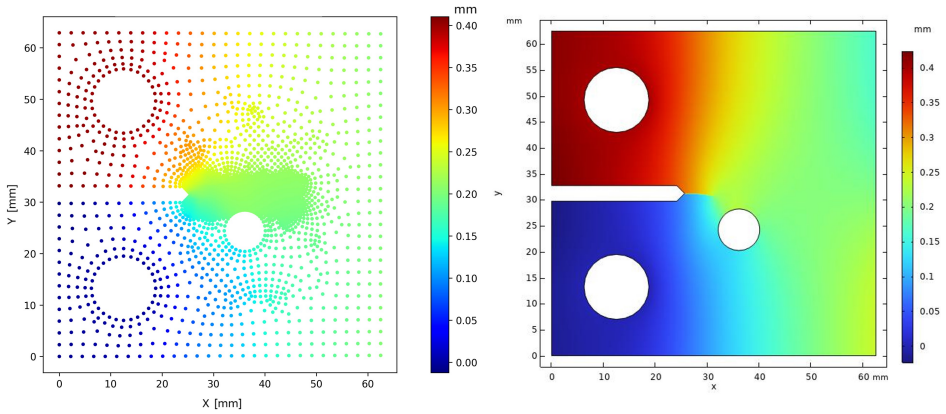


Figure 5.29: Elastic energy vs. prescribed displacement for the plate with holes.

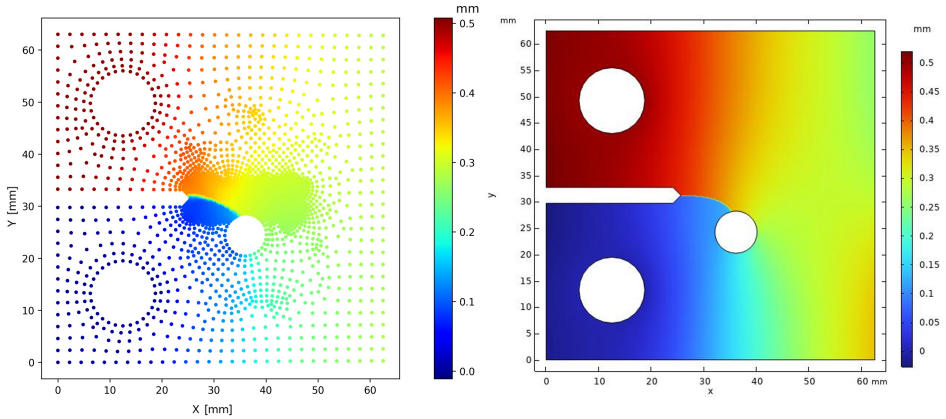
As shown, the curves are quite different.
Using $l_0 = 0.15$ mm, the final point coincides with the FEM one, which is at $v = 0.50$ mm, while the beginning of the propagation is at $v = 0.45$ mm instead of $v = 0.40$ mm.
Instead, crack propagation is shifted to the interval $v = [0.7, 0.75]$ mm when adopting $l_0 = 0.1$ mm.
If the purpose is obtaining the best path approximation, this second case matches the FEM simulation better, since it captures more precisely the characteristic curvature of the I-II mixed mode.
On the other hand, to analyze the numerical values and the displacement distribution, the first case is more suitable, since it appears more similar. For this reason, only the solution with $l_0 = 0.15$ mm is analyzed from now on.
The elastic energy presents the characteristic shape: the curve starts from zero and rises with a positive slope until the starting fracture point, then it decreases with a negative slope until during the fracture propagation. When the crack reaches the central hole, the elastic energy starts to increase again as a new crack will propagate from the right side of the hole.
However, numerically some differences are evident and it is clear from the initial shape that the energy solution doesn't converge. This can be explained by the displacement fields: the calculation of the elastic energy is based on strain decomposition, which in turn depends on displacement, as the strains are obtained by calculating the gradients.
The vertical component of the displacement is more faithful to the FEM results, as shown in Fig.5.30, where a comparison of the two simulations at various Δv values is provided.
As depicted, there is an analogy in the distribution of the vertical displacement and also in the numerical values.
As described before, the fracture starts later in the NN simulation, and for this reason, in Fig.5.30b it is not yet present. Instead, the behavior at $\Delta v = 0.5$ mm is well captured, as shown in Fig.5.30c.



(a) $\Delta v = 0.2$ mm



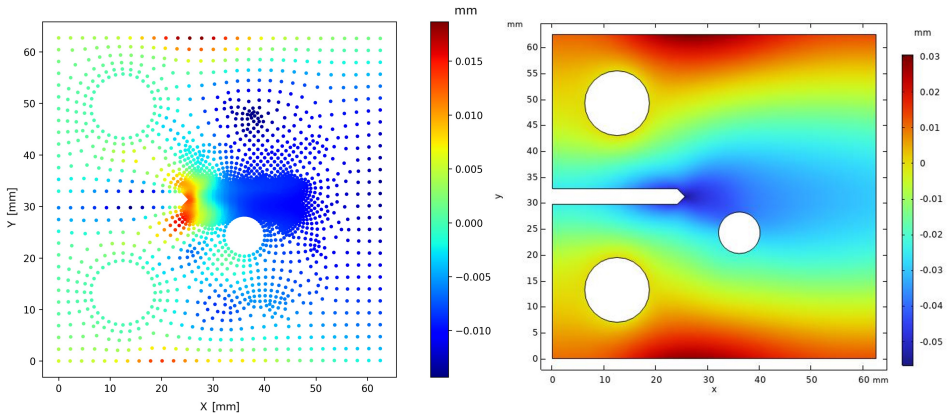
(b) $\Delta v = 0.4$ mm



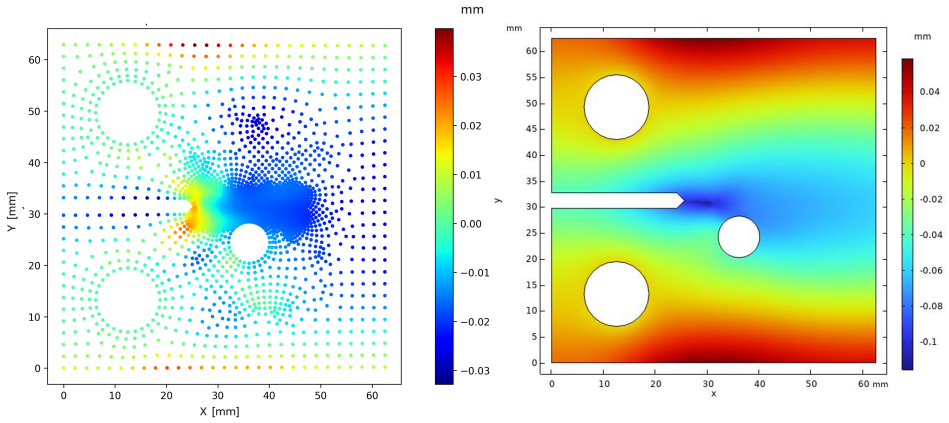
(c) $\Delta v = 0.5$ mm

Figure 5.30: Comparison of v displacement at various Δv values between NN and FEM.

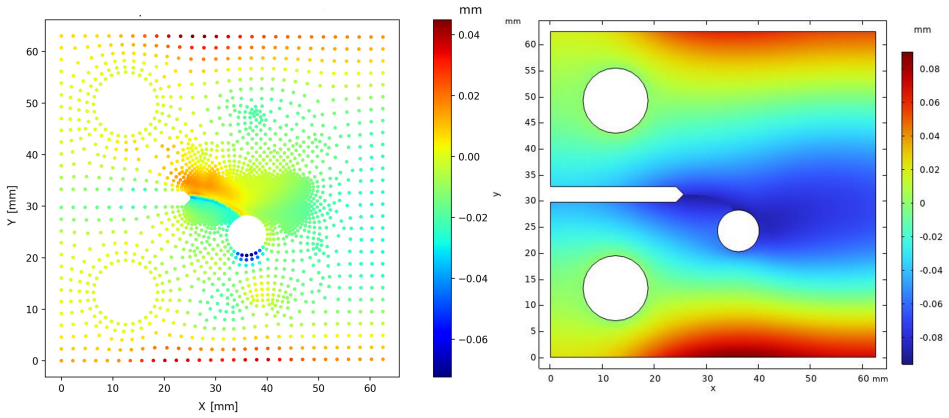
Instead, a higher gap is present for what concerns the horizontal component of the displacement, as depicted in Fig.5.31.



(a) $\Delta v = 0.2$ mm



(b) $\Delta v = 0.4$ mm



(c) $\Delta v = 0.5$ mm

Figure 5.31: Comparison of u displacement at various Δv values between NN and FEM.

In particular, the area subject to positive displacement is restricted to a small central region instead of the entire upper and lower edges of the plate.
The areas surrounding the holes are correctly subjected to zero displacement

through the enforcement of the boundary conditions. At the same time, at the notch, a completely different behavior is observed compared to that predicted by FEM simulation. Indeed, this area turns out to be subjected to a positive displacement instead of a negative one.
The central area, on the other hand, exhibits higher values of compression.
Such differences are particularly evident before the fracture.
In Fig.5.31c, after fracture propagation, there is greater affinity along the boundary edges, while the region surrounding the notch still exhibits the criticality explained previously.
One possible influencing factor could be the choice of enforcement. Indeed, to impose the boundary conditions, analytical functions have been chosen, which affect the distribution of the displacement based on the Euclidean distance.
Another possibility could be the activation function: *tanh* has been adopted for this study, as it is a "smooth" function. However, this choice has a significant impact on the network results and could lead to inaccurate solutions or even to a failure of the training process.
Regardless of the highlighted differences, it must be considered that the simulation runs just in 4 minutes, which is a sufficiently short time to balance the presence of the discrepancies.

5.2.4 Mesh size influence

One of the main drawbacks of the phase-field method is that, to accurately study fracture behavior, a sufficiently refined mesh in the damaged zone is required. Moreover, the element size should be of a dimension comparable to or smaller than the scale parameter l_0, usually less than $0.5 \cdot l_0$ to obtain a precise crack topology [16, 13]. This impacts computational costs.
Below, some figures are provided, representing the discretization and the fracture path predicted by the neural network as the element size in the expected crack region varies. In particular, in Fig.5.32 the adopted element size is l_0; in Fig.5.33 it is equal to $2 \cdot l_0$; while in Fig.5.34 an element size equal to $5 \cdot l_0$ is used.

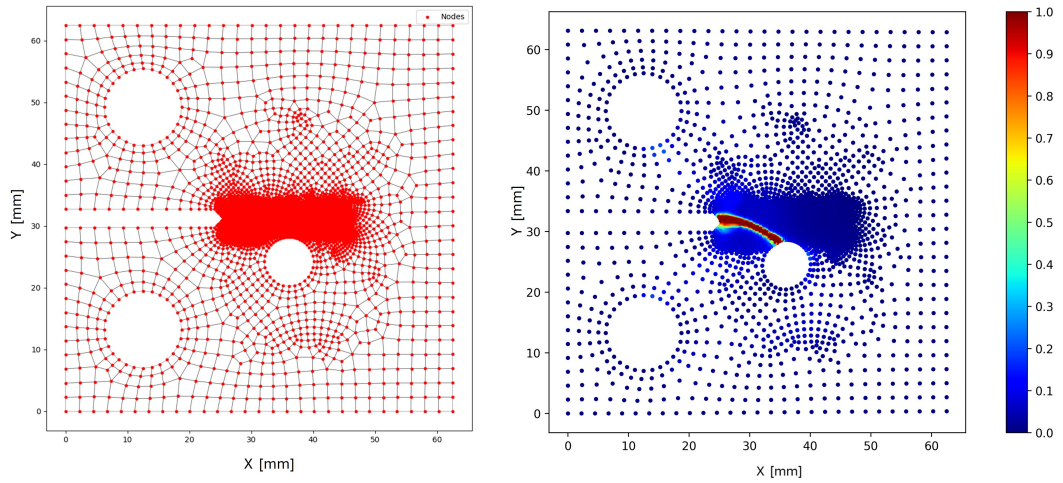


Figure 5.32: Crack propagation with element size l_0 in the finer mesh region.

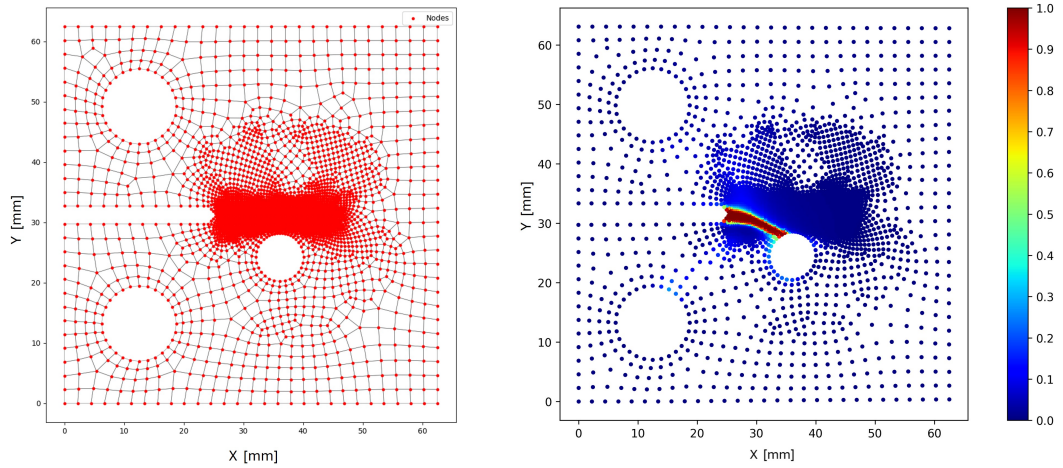


Figure 5.33: Crack propagation with element size $2 \cdot l_0$ in the finer mesh region.

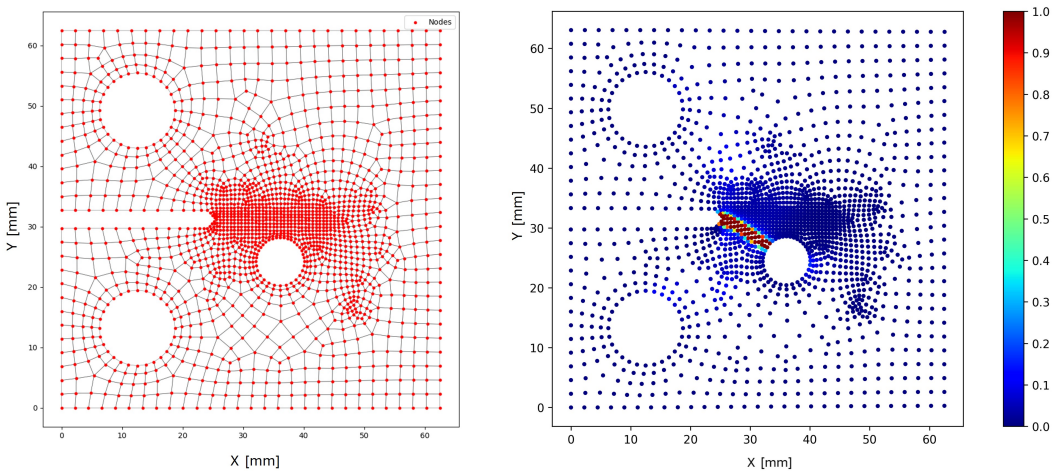


Figure 5.34: Crack propagation with element size $5 \cdot l_0$ in the finer mesh region.

As it can be seen, as the element size increases, the fracture path undergoes a slight deviation. In particular, the characteristic curvature of the I-II mixed mode is lost and the final fracture point moves from the upper point of the hole towards the left.
However, the discretization in Fig.5.34 is much less refined and the fracture path, although less precise, provides a good approximation of the real propagation.
Despite the fracture path, some problems exist concerning the energy associated with crack propagation. The elastic energy curves associated with the different element sizes are shown in Fig.5.35. It can be noticed that the energy doesn't converge to FEM results.

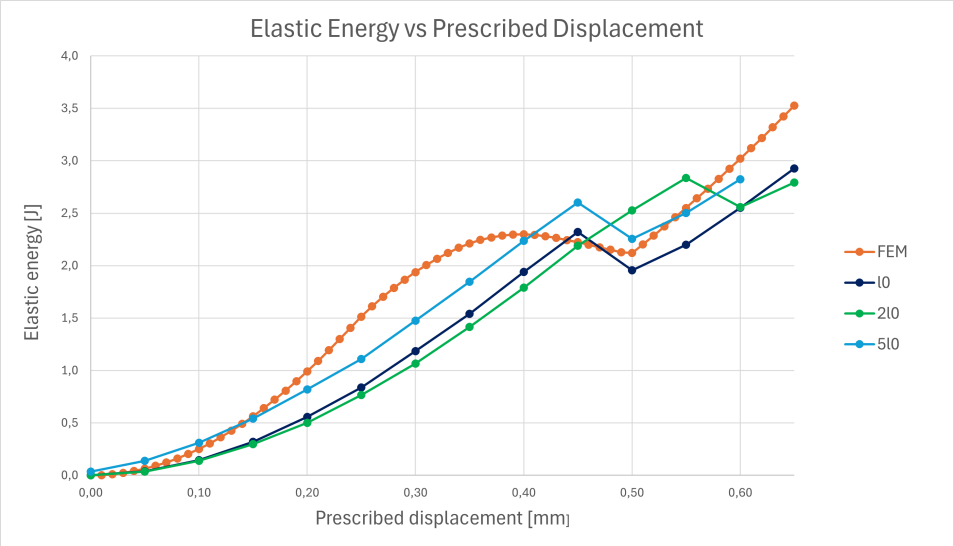


Figure 5.35: Elastic energy vs. prescribed displacement varying the element size in the refined region.

Chapter 6

Conclusions

Fracture studies are a fundamental aspect of mechanical design and require numerical methods to solve partial differential equations.
In this work, the phase-field method is implemented together with the deep energy method to simulate crack propagation in two bi-dimensional plates using deep neural networks. Their application in this kind of analysis has shown promising results, demonstrating the capability of neural networks in fracture mechanics simulations.

The crack path is correctly identified in both cases, which was the primary goal.
The results demonstrate that the neural network is suitable for predicting I-mode fracture and the more complex I-II mixed-mode fracture.
The displacement field also shows good accuracy, even if there are some differences in the u component for the plate with holes.

A key factor is the computation time. In the first example, it has been proven that it is possible to reduce significantly the computational time by decreasing the number of the sampling points and conducting hyperparameters tuning.
The second plate, instead, shows excellent computation time compared to the first one. This could be associated with the change in the integration method. Indeed, while in the first case, a trapezoidal integration is performed, in this second example Gauss integration is employed. Thus, despite the more complex geometry, this allows for a proportional decrease in the number of nodes, even if the second domain is more than ten times larger. In addition, the intrinsic parallelization capabilities of numerical methods can be used to distribute the calculation among multiple units, speeding up the simulation.

Another aspect to focus attention on is the second plate's size: unlike many case studies that can be found in the literature on this topic, its dimensions are not in the range of the normalized interval [0,1] but are one order of magnitude

higher. This is particularly promising since it proves the possibility of extending this approach to larger-scale and real engineering problems.

Finally, a key advantage lies in DEM, which allows a unified multiphysics modeling based on the energy principle. Incorporating the energy approach in its definition, it can be applied to model and integrate different physical phenomena.

Despite the positive results, some limitations are still present. Indeed, a trade-off between the computational effort and the accuracy is still necessary: depending on the type of problem, if a high accuracy is required, computational times could be prohibitive. Moreover, a systematic approach for calibrating and tuning the neural network still doesn't exist: usually, a trial-and-error strategy is adopted for each problem. Indeed, the same setup can be used, but optimization and adaptation are needed. Additionally, strong enforcement requires the definition of analytical boundary functions, which aren't always easy to determine.

The results of this project are promising for further developments in the future. A more precise hyperparameter tuning, different network architectures, together with alternative activation functions and boundary conditions enforcement, could reduce the current discrepancies leading to more accurate results. Moreover, the current framework can be extended to include more material behaviors, such as plasticity, hyperelasticity, and load cases, such as fatigue. A wide range of applications could enhance the robustness of this method, allowing a growing space of neural networks in fracture mechanics.

Bibliography

[1] Charlotte Kuhn. «Numerical and Analytical Investigation of a Phase Field Model for Fracture». doctoralthesis. Technische Universität Kaiserslautern, 2013, pp. X, 143. URL: https://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-35257.

[2] Francesca Pistorio, Davide Clerici, Francesco Mocera, and Aurelio Somà. «Review on the numerical modeling of fracture in active materials for lithium ion batteries». In: *Journal of Power Sources* 566 (2023), p. 232875. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2023.232875. URL: https://www.sciencedirect.com/science/article/pii/S0378775323002501.

[3] T.L. Anderson. *Fracture Mechanics: Fundamentals and Applications*. 2017. DOI: 10.1201/9781315370293.

[4] Michael Janssen, J. Zuidema, and Russell Wanhill. *Fracture Mechanics*. Jan. 2004.

[5] Mehdi Farshad. «3 - Fracture of plastic pipes». In: *Plastic Pipe Systems*. Ed. by Mehdi Farshad. Oxford: Elsevier Science, 2006, pp. 53–100. ISBN: 978-1-85617-496-1. DOI: https://doi.org/10.1016/B978-185617496-1/50004-5. URL: https://www.sciencedirect.com/science/article/pii/B9781856174961500045.

[6] Francesca Pistorio and Aurelio Soma. «Fatigue fracture mechanics in gold-based MEMS notched specimens: experimental and numerical study». In: *Journal of Micromechanics and Microengineering* 33 (June 2023). DOI: 10.1088/1361-6439/acddf3.

[7] S. Sajith, K. S. R. Krishna Murthy, and P. S. Robi. «Prediction of Accurate Mixed Mode Fatigue Crack Growth Curves using the Paris' Law». In: *Journal of the Institution of Engineers (India) Series C* 100.1 (2019), pp. 165–174. ISSN: 2250-0545.

[8] Lallit Anand and Sanjay Govindjee. *Continuum Mechanics of Solids*. Oxford University Press, July 2020. ISBN: 9780198864721. DOI: 10.1093/oso/9780198864721.001.0001. URL: https://doi.org/10.1093/oso/9780198864721.001.0001.

[9] M. Ambati, T. Gerasimov, and L. De Lorenzis. «Phase-field modeling of ductile fracture». In: *Computational Mechanics* 55.5 (2015), pp. 1017–1040. ISSN: 1432-0924. DOI: 10.1007/s00466-015-1151-4. URL: https://doi.org/10.1007/s00466-015-1151-4.

[10] G.A. Francfort and J.-J. Marigo. «Revisiting brittle fracture as an energy minimization problem». In: *Journal of the Mechanics and Physics of Solids* 46.8 (1998), pp. 1319–1342. ISSN: 0022-5096. DOI: https://doi.org/10.1016/S0022-5096(98)00034-9. URL: https://www.sciencedirect.com/science/article/pii/S0022509698000349.

[11] B. Bourdin, G.A. Francfort, and J-J. Marigo. «Numerical experiments in revisited brittle fracture». In: *Journal of the Mechanics and Physics of Solids* 48.4 (2000), pp. 797–826. ISSN: 0022-5096. DOI: https://doi.org/10.1016/S0022-5096(99)00028-9. URL: https://www.sciencedirect.com/science/article/pii/S0022509699000289.

[12] Michael J. Borden, Clemens V. Verhoosel, Michael A. Scott, Thomas J.R. Hughes, and Chad M. Landis. «A phase-field description of dynamic brittle fracture». In: *Computer Methods in Applied Mechanics and Engineering* 217-220 (2012), pp. 77–95. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2012.01.008. URL: https://www.sciencedirect.com/science/article/pii/S0045782512000199.

[13] Shuwei Zhou, Timon Rabczuk, and Xiaoying Zhuang. «Phase field modeling of quasi-static and dynamic crack propagation: COMSOL implementation and case studies». In: *Advances in Engineering Software* 122 (2018), pp. 31–49. ISSN: 0965-9978. DOI: https://doi.org/10.1016/j.advengsoft.2018.03.012. URL: https://www.sciencedirect.com/science/article/pii/S0965997818300061.

[14] Abhinav Gupta, Duc Tien Nguyen, Hirshikesh, and Ravindra Duddu. «Damage mechanics challenge: Predictions from an adaptive finite element implementation of the stress-based phase-field fracture model». In: *Engineering Fracture Mechanics* 306 (2024), p. 110252. ISSN: 0013-7944. DOI: https://doi.org/10.1016/j.engfracmech.2024.110252. URL: https://www.sciencedirect.com/science/article/pii/S0013794424004156.

[15] Tianju Xue, Sigrid Adriaenssens, and Sheng Mao. «Mapped phase field method for brittle fracture». In: *Computer Methods in Applied Mechanics and Engineering* 385 (2021), p. 114046. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2021.114046. URL: https://www.sciencedirect.com/science/article/pii/S0045782521003777.

[16] Jian-Ying Wu, Vinh Phu Nguyen, Chi Thanh Nguyen, Danas Sutula, Sina Sinaie, and Stéphane P.A. Bordas. «Phase-field modeling of fracture». In: ed. by Stéphane P.A. Bordas and Daniel S. Balint. Vol. 53. Advances in Applied Mechanics. Elsevier, 2020, pp. 1–183. DOI: https://doi.org/

10.1016/bs.aams.2019.08.001. URL: https://www.sciencedirect.com/science/article/pii/S0065215619300134.

[17] Anders Krogh. «What are artificial neural networks?» In: *Nature Biotechnology* 26.2 (2008), pp. 195–197. ISSN: 1546-1696. DOI: 10.1038/nbt1386. URL: https://doi.org/10.1038/nbt1386.

[18] Stefan Kollmannsberger, Davide D'Angella, Moritz Jokeit, and Leon Herrmann. *Deep Learning in Computational Mechanics: An Introductory Course*. 1st ed. Studies in Computational Intelligence. Springer Cham, Aug. 2021, pp. VI, 104. ISBN: 978-3-030-76587-3. DOI: 10.1007/978-3-030-76587-3. URL: https://doi.org/10.1007/978-3-030-76587-3.

[19] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. «Multilayer feedforward networks are universal approximators». In: *Neural Networks* 2.5 (1989), pp. 359–366. ISSN: 0893-6080. DOI: https://doi.org/10.1016/0893-6080(89)90020-8. URL: https://www.sciencedirect.com/science/article/pii/0893608089900208.

[20] S. Hildebrand and S. Klinge. «Comparison of neural FEM and neural operator methods for applications in solid mechanics». In: *Neural Computing and Applications* 36 (2024), pp. 16657–16682. DOI: https://doi.org/10.1007/s00521-024-10132-2.

[21] Vien Minh Nguyen-Thanh, Xiaoying Zhuang, and Timon Rabczuk. «A deep energy method for finite deformation hyperelasticity». In: *European Journal of Mechanics - A/Solids* 80 (2020), p. 103874. ISSN: 0997-7538. DOI: https://doi.org/10.1016/j.euromechsol.2019.103874. URL: https://www.sciencedirect.com/science/article/pii/S0997753819305352.

[22] Somdatta Goswami, Cosmin Anitescu, and Timon Rabczuk. «Adaptive fourth-order phase field analysis using deep energy minimization». In: *Theoretical and Applied Fracture Mechanics* 107 (2020), p. 102527. ISSN: 0167-8442. DOI: https://doi.org/10.1016/j.tafmec.2020.102527. URL: https://www.sciencedirect.com/science/article/pii/S0167844219306858.

[23] Bin Ding, Huimin Qian, and Jun Zhou. «Activation functions and their characteristics in deep neural networks». In: *2018 Chinese Control And Decision Conference (CCDC)*. 2018, pp. 1836–1841. DOI: 10.1109/CCDC.2018.8407425.

[24] Junxi Feng, Xiaohai He, Qizhi Teng, Chao Ren, Honggang Chen, and Yang Li. «Reconstruction of porous media from extremely limited information using conditional generative adversarial networks». In: *Physical Review E* 100 (Sept. 2019). DOI: 10.1103/PhysRevE.100.033308.

[25] Maryam M. Najafabadi, Taghi M. Khoshgoftaar, Flavio Villanustre, and John Holt. «Large-scale distributed L-BFGS». eng. In: *Journal of big data* 4.1 (2017), pp. 1–17. ISSN: 2196-1115.

[26] Diederik P Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: (2014). URL: https://arxiv.org/abs/1412.6980.

[27] Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. «Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next». In: *Journal of Scientific Computing* 92 (July 2022). DOI: 10.1007/s10915-022-01939-z.

[28] Chengping Rao, Hao Sun, and Yang Liu. «Physics-Informed Deep Learning for Computational Elastodynamics without Labeled Data». In: *Journal of Engineering Mechanics* 147 (Aug. 2021), p. 04021043. DOI: 10.1061/(ASCE)EM.1943-7889.0001947.

[29] Sergio Rojas, Paweł Maczuga, Judit Muñoz-Matute, David Pardo, and Maciej Paszyński. «Robust Variational Physics-Informed Neural Networks». In: *Computer Methods in Applied Mechanics and Engineering* 425 (2024), p. 116904. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2024.116904. URL: https://www.sciencedirect.com/science/article/pii/S0045782524001609.

[30] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. *Variational Physics-Informed Neural Networks For Solving Partial Differential Equations*. 2019. arXiv: 1912.00873 [cs.NE]. URL: https://arxiv.org/abs/1912.00873.

[31] Stefano Berrone, Claudio Canuto, and Moreno Pintore. «Variational Physics Informed Neural Networks: the Role of Quadratures and Test Functions». In: *Journal of Scientific Computing* 92.3 (Aug. 2022). ISSN: 1573-7691. DOI: 10.1007/s10915-022-01950-4. URL: http://dx.doi.org/10.1007/s10915-022-01950-4.

[32] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. *KAN: Kolmogorov-Arnold Networks*. 2024. arXiv: 2404.19756 [cs.LG]. URL: https://arxiv.org/abs/2404.19756.

[33] Subhajit Patra, Sonali Panda, Bikram Keshari Parida, Mahima Arya, Kurt Jacobs, Denys I. Bondar, and Abhijit Sen. *Physics Informed Kolmogorov-Arnold Neural Networks for Dynamical Analysis via Efficent-KAN and WAV-KAN*. 2024. arXiv: 2407.18373 [cs.LG]. URL: https://arxiv.org/abs/2407.18373.

[34] S.P. Venkateshan and Prasanna Swaminathan. «Chapter 9 - Numerical Integration». In: *Computational Methods in Engineering*. Ed. by S.P. Venkateshan and Prasanna Swaminathan. Boston: Academic Press, 2014, pp. 317–373. ISBN: 978-0-12-416702-5. DOI: https://doi.org/10.1016/B978-0-12-416702-5.50009-0. URL: https://www.sciencedirect.com/science/article/pii/B9780124167025500090.

[35] A. Gugliotta, N. Zampieri, and A. Somà. *Elementi finiti*. Quine, 2022. ISBN: 9788831284066. URL: https://books.google.de/books?id=NoNFzwEACAAJ.

[36] *COMSOL Multiphysics Reference Manual*.

[37] Wenbing Zhang, Zhen-zhong Shen, Jie Ren, Lei Gan, Liqun Xu, and Yiqing Sun. «Phase-field simulation of crack propagation in quasi-brittle materials: COMSOL implementation and parameter sensitivity analysis». In: *Modelling and Simulation in Materials Science and Engineering* 29 (June 2021), p. 055020. DOI: 10.1088/1361-651X/ac03a4.

[38] Yahya Fageehi. «Two- and Three-Dimensional Numerical Investigation of the Influence of Holes on the Fatigue Crack Growth Path». In: *Applied Sciences* 11 (Aug. 2021), p. 7480. DOI: 10.3390/app11167480.