

UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale

Corso di Laurea Magistrale in Aerospace Engineering

**Development of a computational
framework for the identification of
peridynamic constitutive laws via neural
networks**

Relatore

Prof. Mirco Zaccariotto

Laureando

Alessandro Riggio

Correlatori

Prof. Ugo Galvanetto

Dott. Francesco Scabbia

Matricola: 2155381

Anno Accademico 2025/2026

Abstract

The structural assessment of damaged components requires accurately predicting crack propagation. However, standard numerical approaches like the Finite Element Method (FEM) struggle with this task because Classical Continuum Mechanics (CCM) relies on differential equations that assume a continuous domain. Peridynamics (PD) addresses this fundamental limitation by employing an integral formulation, allowing discontinuities to be handled naturally within the governing equations. Reproducing the response of a given material within PD, though, requires selecting a constitutive formulation among the several models that are typically available, together with the parameters governing the resulting response. The PD correspondence formulation already addresses this task in principle, by directly relating the PD force state to the classical Piola stress tensor defined within CCM, but due to its considerable computational cost, a manual calibration of the underlying PD variables against CCM parameters is generally preferred in practice for lighter models such as Bond-Based PD. To address these constraints, this thesis proposes an alternative data-driven approach, in which a convolutional neural network (CNN) is trained to identify, directly from an observed stress-strain curve, both the most suitable PD constitutive law and its governing parameters, using quasi-brittle materials as a case study. A displacement-controlled arc-length solver is developed to handle the post-peak instabilities and reconstruct the softening branch deriving from quasi-static tensile tests, and is used to generate a synthetic dataset spanning three reference constitutive laws. The internal consistency of the framework is validated before training and separately testing the individual components of the network: a classifier, identifying the constitutive law, and three law-specific regressors, estimating the governing parameters. Finally, the complete identification pipeline is validated end-to-end against a benchmark stress-strain curve, by predicting the underlying constitutive law and parameters, and comparing the re-simulated response to the original one. The results obtained show good agreement between the reconstructed and reference curves, highlighting the capability of the proposed approach and suggesting its

extension, in future work, to a broader range of material behaviors.

Abstract in lingua italiana

La verifica strutturale di componenti danneggiati richiede di prevedere accuratamente la propagazione delle cricche. Gli approcci numerici standard come il metodo degli elementi finiti (FEM) faticano tuttavia in questo compito, poiché la Meccanica del Continuo Classica (CCM) si basa su equazioni differenziali che assumono un dominio continuo. La teoria peridinamica (PD) affronta questa limitazione impiegando una formulazione integrale, che permette di gestire le discontinuità in modo naturale. Riprodurre la risposta di un dato materiale all'interno della PD richiede però di scegliere una formulazione costitutiva tra i diversi modelli generalmente disponibili, oltre ai parametri che governano la risposta risultante. Il PD correspondence model affronta già in linea di principio questo compito, collegando direttamente lo stato di forza PD al tensore degli sforzi di Piola classico definito nell'ambito della CCM, ma a causa del suo costo computazionale considerevole, nella pratica si preferisce generalmente una calibrazione manuale delle quantità sottostanti PD rispetto ai parametri CCM applicata a modelli più leggeri come la PD Bond-Based. Per superare questi limiti, la presente tesi propone un approccio alternativo data-driven, in cui una rete neurale convoluzionale (CNN) viene allenata per identificare, direttamente da una curva sforzo-deformazione osservata, sia la legge PD costitutiva più adatta sia i suoi parametri governanti, utilizzando i materiali con comportamento quasi-fragili come caso studio. A tal fine, è stato sviluppato un solutore arc-length a controllo di spostamento per gestire le instabilità post-picco e ricostruire il tratto di softening generato da prove di trazione quasi-statiche, che è poi stato utilizzato per generare un dataset sintetico che copre tre leggi costitutive di riferimento. La coerenza interna del framework è stata validata prima di allenare e testare separatamente le singole componenti della rete: un classificatore, che identifica la legge costitutiva, e tre regressori specifici per legge, che stimano i parametri governanti. Infine, l'intera pipeline di identificazione è stata validata end-to-end su una curva sforzo-deformazione di riferimento, prevedendone la legge costitutiva e i parametri sottostanti, e confrontando la risposta ri-simulata con quella origi-

nale. I risultati ottenuti mostrano un buon accordo tra le curve ricostruita e di riferimento, evidenziando così le capacità dell'approccio proposto e suggerendone l'estensione, in lavori futuri, a una gamma più ampia di comportamenti di materiale.

Contents

Abstract	i
Abstract in lingua italiana	iii
Nomenclature	xviii
Acronyms	xix
1 Introduction	1
2 Peridynamics theory	3
2.1 General theory foundation	3
2.2 Bond Based formulation	6
2.3 State Based formulation	9
2.4 Constitutive law for quasi-brittle material	15
2.5 Implementation strategies	19
2.5.1 Volume Correction	21
2.5.2 Surface Effect Correction	23
2.5.3 Peridynamics force flux	24
2.5.4 Peridynamic convergence analysis type	26
3 Neural Networks	29
3.1 Overview of Machine Learning	29
3.1.1 Training, validation and test sets	32
3.1.2 Generalization and overfitting	32
3.2 Artificial Neural Networks	34
3.3 Training of Neural Networks	40
3.3.1 Backpropagation	40
3.3.2 Gradient-based optimization	42
3.3.3 Adam optimizer	43
3.3.4 Training hyperparameters	44

3.4	Convolutional Neural Networks	45
3.4.1	One-dimensional convolution	45
3.4.2	Convolutional layers	48
3.4.3	Pooling	50
3.4.4	CNN structure	51
4	Numerical solver	53
4.1	Newton-Raphson iterative schemes	53
4.2	Snap-back/snap-through instabilities	55
4.3	Displacement controlled arc-length method	57
4.3.1	Arc-length constraint and corrector algorithm	60
4.3.2	Predictor step	61
4.3.3	Update and convergence criterion	63
4.3.4	Construction of the tangent stiffness matrix	64
4.3.5	Stiffness matrix index allocation	67
4.4	Implementation strategies	69
4.4.1	Adaptive arc-length step size	70
5	Dataset Generation	75
5.1	Sampling strategy for material parameters	75
5.2	Simulation parameters and dataset organization	78
5.3	Simulation results	83
5.4	Dataset results	84
5.5	Data pre-processing for CNN	88
5.6	Dataset intrinsic limitations	91
6	Adopted CNN Architecture	93
6.1	Classifier branch	94
6.2	Regressor branches	96
6.3	Training setup	99
7	Results discussion	101
7.1	Classifier performance	101
7.2	Regressors performance	105
7.3	End-to-end validation of the identification pipeline	110

8 Conclusions	115
Bibliography	117

List of Figures

2.1	Kinematics of the peridynamic bond between a material point $\mathbf{x}$ and a neighbor $\mathbf{x}'$ within its horizon family $\mathcal{H}_{\mathbf{x}}$ [5].	4
2.2	Relative position vector $\boldsymbol{\xi}$ and relative displacement vector $\boldsymbol{\eta}$ [5].	5
2.3	Representation of the interaction force density for different PD formulations [8].	6
2.4	Pairwise force density as a function of the bond stretch for the PMB material model [9].	9
2.5	Reference position state field $\underline{\mathbf{X}}$ and deformation state field $\underline{\mathbf{Y}}$ [8].	10
2.6	PD constitutive laws adopted.	15
2.7	Schematic representation of the meshfree spatial discretization of a two-dimensional domain with uniform grid spacing Δx . . .	20
2.8	Categorization of family nodes according to their relative cell position with respect to the horizon boundary: fully contained internal nodes (stars), partially intersected boundary nodes (triangles and squares), and external nodes (circles) [16].	21
2.9	Peridynamic surface effect: points close to the boundary of the body, such as <i>Point 2</i> , have an incomplete neighborhood [18]. .	23
2.10	Peridynamic force flux.	25
2.11	Possible types of PD convergence analysis [5].	26
3.1	Taxonomy of machine learning paradigms.	30
3.2	Graphical representation of underfitting, overfitting, and an optimal generalization, for both classification and regression. . .	33
3.3	Illustration of the early stopping mechanism. The vertical dashed line indicates the optimal training epoch corresponding to the minimum validation loss [22].	33
3.4	Schematic representation of an artificial neuron [21]. Symbol $\mathbf{f}$ here indicates what in this thesis is defined as Φ	35
3.5	Schematic representation of a feed-forward (fully connected) neural network with two hidden layers.	36

3.6	Illustrative example of a NN used as reference to explain the backpropagation method.	40
3.7	Example of the convolution (cross-correlation) operation [33]. . .	46
3.8	Same padding, unit stride: the input (bottom, dashed cells denote locations where zeros are added) is padded just enough so that the output (top) has the same length as the input [34]. . .	47
3.9	Full padding, unit stride: enough zero padding is added (dashed cells) for every input sample to be visited K times by the kernel, producing an output longer than the input [34].	48
3.10	No padding (valid convolution), unit stride: the kernel visits every position fully contained within the input, and the output is shorter than the input by $K - 1$ [34].	48
3.11	No padding, stride greater than one: the kernel skips positions as it slides across the input, further reducing the length of the output beyond what valid padding alone would cause [34]. . . .	49
3.12	Example of max pooling with a $(2, 2)$ filter.	50
4.1	Schematic representation of a uniaxial tensile test on a quasi-brittle specimen with its corresponding stress–displacement response [35].	55
4.2	Non linear material behavior characterized both by snap-through and snap-back instabilities [14].	56
4.3	Index allocation of the tangent stiffness contribution associated with a bond connecting nodes i and j into the global tangent stiffness matrix.	68
4.4	Schematic representation of the arc-length adaptation routine. .	72
5.1	Normalized stress–crack-opening curves for (a) normalweight concrete (NC) and (b) lightweight concrete (LC), showing experimental data points together with two possible law fits, from which the shape parameters c_1 and c_2 are obtained [15].	76
5.2	Distribution of the sampled material parameters (E , f_t , G_0), with the 3σ confidence ellipsoid computed from the sample covariance.	77

5.3	Realized marginal distributions of the sampled material parameters E , f_t and G_0 , overlaid with the theoretical target normal PDF of Eq. (114).	78
5.4	Dog-bone specimen geometry used for the PD tensile tests, with fixed (left) and prescribed-displacement (right) boundary conditions, together with a detail of the bond connectivity in correspondence of the two pre-crack.	80
5.5	Stress-strain response imposing bilinear constitutive law for three different values of fracture energy G_0 , with $f_t = 1$ MPa and $E = 20$ GPa held fixed, together with four snapshots of the nodal damage field φ (points a–d) at increasing levels of deformation for fixed $G_0 = 30.0$ J/m ²	84
5.6	Number of valid simulations obtained per constitutive law. . . .	85
5.7	Distribution of run time per simulation (on the left) and cumulative wall-clock time of the campaign as a function of simulation index (on the right).	85
5.8	Overlay of all valid stress-strain curves obtained from the simulation campaign.	86
5.9	Stress-strain response of the trilinear constitutive law for three values of the elastic modulus E , with f_t and G_0 held fixed. . . .	87
5.10	Stress-strain response of the trilinear constitutive law for three values of the tensile strength f_t , with E and G_0 held fixed. . . .	87
5.11	Stress-strain response of the trilinear constitutive law for three values of the fracture energy G_0 , with E and f_t held fixed. . . .	88
5.12	Raw stress–strain response, with an enlarged view of the curve in the critical region where the sampled points are closely spaced in terms of strain.	89
5.13	Same curve as in Figure 5.12, resampled onto a uniform strain grid consisting of N points using linear interpolation. Enlarged view shows that the resampled data points closely reproduce the original raw curve, including in regions corresponding to very small differences in relative strain.	90

5.14	Resampled curve of Figure 5.13 after per-sample normalization, mapping both strain and stress onto the $[0, 1]$ range regardless of the simulation's absolute scale.	91
6.1	High-level overview of the hierarchical CNN architecture.	93
6.2	Layer-by-layer architecture of the classifier branch (here N_{samp} is shortened in N_s).	95
6.3	Layer-by-layer architecture of the law-specific regressor branches, each producing the bond-level parameters s_0, s_c (here N_{samp} is shortened in N_s).	98
7.1	Classifier training progress.	101
7.2	Confusion matrix of the classifier branch on the test set.	102
7.3	Cumulative explained variance of the CNN feature space as a function of the number of principal components.	103
7.4	Test-set samples projected onto the first two principal components of the CNN feature space, with cluster centroids and the misclassified sample highlighted.	104
7.5	Misclassified trilinear sample compared against a typical, correctly classified curve of both the trilinear and the exponential law.	104
7.6	5-fold cross-validation distribution of the MAPE for s_0 and s_c , trilinear regressor.	106
7.7	Parity plots for s_0 and s_c , trilinear regressor test subset.	106
7.8	5-fold cross-validation distribution of the MAPE for s_0 and s_c , bilinear regressor.	107
7.9	Parity plots for s_0 and s_c , bilinear regressor test subset.	107
7.10	5-fold cross-validation distribution of the MAPE for s_0 and s_c , exponential regressor.	108
7.11	Parity plots for s_0 and s_c , exponential regressor test subset.	108
7.12	End-to-end validation procedure.	110
7.13	Digitized experimental stress-strain curve of a notched concrete specimen under direct tension, extracted from Figure 7(a) of [53].	112

7.14	Predicted constitutive law and associated probability (on the left), together with the PD bond-force law reconstructed from the estimated parameters (on the right).	112
7.15	Comparison between the reference curve and the curve reconstructed from the predicted bond-level parameters (on the left), together with the point-wise error along the strain axis (on the right).	113

Nomenclature

Δl	Arc length
$\Delta l_{\max}$	Arc-length upper bound
μ	Bond scalar state function
s	Bond scalar stretch
k	Bulk modulus
$\mathbf{P}$	Correlation matrix
s_0	Critical elastic stretch
s_c	Critical ultimate stretch
A	Cross-sectional area
$\mathbf{J}$	Deformation gradient tensor
$\mathbf{Y}$	Deformation state
$\mathbf{M}$	Deformed bond versor
$\mathbf{y}$	Deformed position vector
I_d	Desired number of corrector iterations
$\underline{e}^d$	Deviatoric extension state
$\mathbf{D}$	Dirichlet nodal displacement
$\delta \mathbf{u}$	Displacement incremental correction
$\mathbf{u}$	Displacement vector field
h	Dog-bone specimen thickness
$\delta^{[l]}$	Error signal at layer l

$\hat{\mathbf{p}}$	Estimated classifier probability output
c_1, c_2	Exponential model constant
$\underline{e}$	Extension state
ΔV	Finite nodal volume
$\underline{\mathbf{T}}$	Force state
G_0	Fracture energy
$\mathbf{G}$	Geometric matrix
Δx	Grid spacing
m	Horizon factor
λ	Incremental scalar load factor
$\underline{\omega}$	Influence function
Δl_0	Initial arc length
$\mathbf{F}$	Internal force vector
K	Kernel filter size
w	Kernel filter
s_k	Kink point stretch
η	Learning rate
ρ	Local density
$\mathcal{L}$	Loss function
s_{max}	Maximum achieved history stretch
$n_{\max}$	Maximum number of increments for arc-length solver
c	Micromodulus

N_{sampl}	Minimum number of sampling points for stress-strain pre-processing
$\mathcal{H}_{\mathbf{x}}$	Neighborhood of material point x
$\mathbf{N}$	Neumann nodal displacement
Φ	Neural activation function
a	Neural activation
b	Neural bias
z	Neural pre-activation
$\mathbf{w}$	Neural weight
$\mathcal{D}$	NN Dataset
$\boldsymbol{\theta}$	NN trainable parameters
φ	Nodal damage factor
ϑ	Nodal dilation factor
C_{in}	Number of CNN 1D input channel
F	Number of kernels applied in parallel
$\mathbf{f}$	Pairwise force density function
δ	PD horizon
σ	Piola stress tensor
ν	Poisson's ratio
$\mathbf{b}$	Prescribed body force
E_{cacl}	Reconstructed Young modulus
q	Reference nodal action vector
$\underline{\mathbf{X}}$	Reference position state

$\mathbf{d}$	Reference vector of imposed nodal displacements
L_2	Regularization factor
$\boldsymbol{\eta}$	relative displacement vector
$\boldsymbol{\xi}$	Relative position vector
I_p	Required number of corrector iterations
$\bar{\epsilon}$	Residual convergence tolerance for arc-length solver
R	Residual vector
$\boldsymbol{\mu}$	Sample parameters mean vector
$\boldsymbol{\sigma}$	Sample parameters standard deviation vector
k_s	Secant axial stiffness
$\bar{\mathbf{K}}$	Secant stiffness matrix
$\mathbf{S}$	Shape tensor
G	Shear modulus
S	Stride
λ_b	Surface effect correction factor
c_T	Tangent micromodulus
$\mathbf{K}$	Tangent stiffness matrix
f_t	Tensile strength
t	Thickness
γ, β	Trilinear model constant
β_{ij}	Volume correction coefficient
$\hat{m}$	Weighted volume
E	Young's modulus

Acronyms

PD Peridynamics

CCM Classical Continuum Mechanics

BB Bond-Based

OSB Ordinary State-Based

NOSB Non-Ordinary State-Based

PMB Prototype Microelastic Brittle

LPF Linear Peridynamic Fluid

LPS Linear Peridynamic Solid

ML Machine Learning

NN Neural Network

ReLU Rectified Linear Unit

Adam Adaptive Moment Estimation

CNN Convolutional Neural Network

SGD Stochastic Gradient Descent

PCA Principal Component Analysis

LCE Logarithmic Cross-Entropy

MSE Mean Squared Error

MAE Mean Absolute Error

MAPE Mean Absolute Percentage Error

RMSE Root Mean Square Error

NRMSE Normalized Root Mean Square Error

CSC Compressed Sparse Column

Chapter 1

Introduction

Numerical simulations play a considerable role in the engineering design field, since it allows to obtain solutions to structural problems that would otherwise require a much greater consumption in terms of time and resources if addressed analytically or, worse, experimentally. A topic of particular interest within this wider context is the prediction of crack propagation through simulation. Numerous strategies have been proposed to simulate crack propagation numerically. Among these, cohesive zone modeling [1] introduces cohesive interfaces between adjacent finite elements, whose stiffness degrades once a certain relative displacement is exceeded; in this approach, however, the crack is constrained to propagate only along the directions in which such interface elements were introduced in the model. Another widely adopted strategy is the extended finite element method [2], in which the crack propagates through the splitting of finite elements; this method, however, does not handle crack branching and interaction particularly well. Another alternative is the phase-field [3], in which the crack is represented as a diffuse, continuously-varying scalar damage field rather than an explicit geometric discontinuity, at the cost of requiring a finer mesh resolution within the regularized crack zone.

Among the approaches developed to describe crack propagation, Peridynamics (PD) [4] has emerged as a particularly effective framework in which discontinuities such as cracks are naturally represented. In particular, crack propagation, branching, and interaction arise directly from its nonlocal formulation without requiring an explicit crack-tracking procedure or additional propagation criteria. In this context, numerous contributions have addressed the simulation of quasi-brittle materials through the use of PD theory, and a wide range of constitutive laws has been proposed aiming to reproduce the softening response observed experimentally in materials such as concrete. Regardless of

the specific softening law adopted, two main steps are required: the selection of the law that best represents the material response and the identification of its corresponding parameters. Both tasks typically rely on calibrating the PD model against macroscopic properties derived from experimental tests, which can be a process that is often time-consuming and iterative. This raises the question of whether these two tasks can be approached in a more systematic, data-driven manner, capable of directly mapping macroscopic stress–strain curves to both the underlying constitutive law and its PD parameters.

This thesis aims to answer this question by developing a computational framework, based on convolutional neural networks (CNNs), for the identification of PD constitutive laws directly from a stress–strain curve. The framework is organized hierarchically: a classifier first identifies which of the candidate softening laws most plausibly generated the observed response, after which one of the law-specific regressor branches estimates the corresponding law parameters. Training such a framework requires a large, labeled dataset of stress–strain curves spanning a representative range of quasi-brittle material behavior, which must be entirely generated through synthetic PD simulations.

The remainder of this thesis is organized as follows: chapters 2 and 3 introduce the theoretical background of Peridynamics and neural networks, respectively. Chapter 4 presents the displacement-controlled arc-length solver developed to trace the full equilibrium path of the quasi-static tensile tests used to generate the training data. Chapter 5 describes the design and execution of the synthetic simulation campaign, the resulting dataset, and its pre-processing for use as CNN training data. Chapter 6 details the architecture of the classifier and regressor branches and the rationale behind their design choices. Chapter 7 presents and discusses the performance of the trained framework, from the isolated evaluation of the classifier and regressors, to the end-to-end validation of the complete identification pipeline. Finally, Chapter 8 summarizes the main findings of this work, discusses its limitations, and outlines directions for future development.

Chapter 2

Peridynamics theory

2.1 General theory foundation

The term *Peridynamics* (*PD*) was first formally introduced by Dr. Stewart A. Silling in 2000 [4]. It is etymologically derived from the Greek roots *peri* and *dynamics*, which can be translated as “around” and “force,” respectively. This nomenclature encapsulates the notion that forces are transmitted within a finite neighborhood surrounding a material point, thereby reflecting a key feature of the theory: the interactions between material points are intrinsically *nonlocal*. In the original peridynamic formulation, known as *Bond Based (BB)* theory, which will be better addressed in the following paragraph, the equation of motion for a generic material point $\mathbf{x}$, belonging to a body $\mathcal{B}$, is given by:

$$\rho(\mathbf{x}) \ddot{\mathbf{u}}(\mathbf{x}, t) = \int_{\mathcal{H}_{\mathbf{x}}} \mathbf{f}(\mathbf{u}(\mathbf{x}', t) - \mathbf{u}(\mathbf{x}, t), \mathbf{x}' - \mathbf{x}) dV_{\mathbf{x}'} + \mathbf{b}(\mathbf{x}, t), \quad (1)$$

where ρ is the local density, $\mathbf{u}$ is the displacement vector field, $\mathcal{H}_{\mathbf{x}}$ is a *neighborhood* of $\mathbf{x}$ (often also called *family* of $\mathbf{x}$), $dV_{\mathbf{x}'}$ is the infinitesimal volume occupied by $\mathbf{x}'$ within $\mathcal{H}_{\mathbf{x}}$, $\mathbf{f}$ is a *pairwise force density function*, that is the force vector (per unit volume squared) that the particle $\mathbf{x}'$ exerts on the particle $\mathbf{x}$, and $\mathbf{b}$ is a prescribed body force (per unit volume). The neighborhood $\mathcal{H}_{\mathbf{x}}$, mathematically defined as $\{\mathbf{x}' \in \mathcal{B} : |\mathbf{x}' - \mathbf{x}| \leq \delta\}$ represents the set of points with $\mathbf{x}$ interacts within a defined positive radius δ called *horizon*. The quantities appearing in Eq. (1) are schematically represented in Figure 2.1, which illustrates the horizon of a material point in the reference and deformed configurations and the pairwise force interactions exchanged between two particles, which arise subsequent to the deformation.

The force per unit volume acting at a material point $\mathbf{x}$, obtained by evaluating the integral on the right-hand side of Eq. (1), depends on interactions with

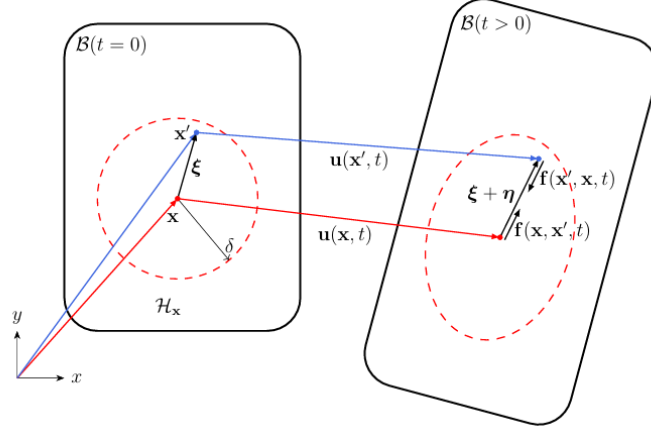


Figure 2.1: Kinematics of the peridynamic bond between a material point $\mathbf{x}$ and a neighbor $\mathbf{x}'$ within its horizon family $\mathcal{H}_{\mathbf{x}}$ [5].

other material points located at a finite distance from $\mathbf{x}$; this dependence is what characterizes the theory as nonlocal, and directly relates it to the concept of a horizon. Indeed, according to the magnitude of horizon, the nonlocal theory admits, as a limiting case for $\delta \rightarrow 0$, a collapse to the framework of *classical continuum mechanics (CCM)* [6]. In this context, the CCM, which relies on spatial derivatives of the displacement field, is reformulated by an integral formulation that depends solely on the displacement field itself. This feature is of fundamental importance because the peridynamic formulation of the equations of motion remains valid even in the presence of discontinuities in the displacement field. Consequently, Peridynamics is particularly well suited for modeling crack initiation and propagation without introducing mathematical inconsistencies.

Hereafter, the relative position between two material points in the reference configuration will be identified by *bond* $\boldsymbol{\xi}$ and their relative displacement by $\boldsymbol{\eta}$, (as illustrated in Figure 2.2):

$$\boldsymbol{\xi} = \mathbf{x}' - \mathbf{x}, \quad \boldsymbol{\eta} = \mathbf{u}(\mathbf{x}', t) - \mathbf{u}(\mathbf{x}, t). \quad (2)$$

It is possible to distinguish among three main formulations of the PD theory: *Bond-Based (BB-PD)*, *Ordinary State-Based (OSB-PD)*, and *Non-Ordinary*

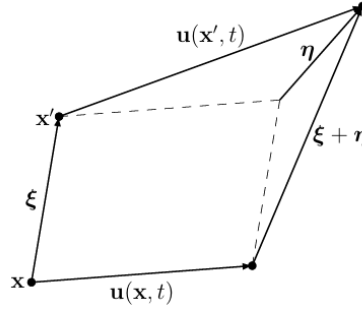


Figure 2.2: Relative position vector ξ and relative displacement vector η [5].

State-Based (NOSB-PD). In BB-PD, the interaction between each pair of material points depends exclusively on the deformation of the bond connecting them. The reciprocal forces exchanged are equal in magnitude and lie along the direction of the bond which connect the two material point. Since in this first formulation any pair of particles interacts only through a central potential that is totally independent of all other local conditions, there are some limitations on the Poisson's ratio ν ($1/4$ in 3D and 2D plane strain models, $1/3$ in 2D plane stress models) that arise from the constraints imposed by the Cauchy's relations [7].

A generalization of the BB-PD, which by the fact resolves the Poisson's ratio limitation, is represented by the OSB-PD formulation, in which the interaction between points no longer depends solely on the single pair of points, but on the deformation state of the entire horizon. In this case, the forces exchanged between material points acts along the vector connecting the points in the deformed configuration, but they do not necessarily have to be equal in magnitude. The NOSB-PD theory, and in particular the *correspondence model*, offers the possibility to connect directly the constitutive laws defined in the CCM framework to the PD framework, without needed to calibrate PD properties to the classical one. However, the formulation suffers from intrinsic problems related to zero-energy modes, making it necessary to adopt stabilization strategies, resulting in a significant increase in computational cost. A summary of the main difference between the three formulation for what concern force density exchanged between pair of material points is provided in

Figure 2.3.

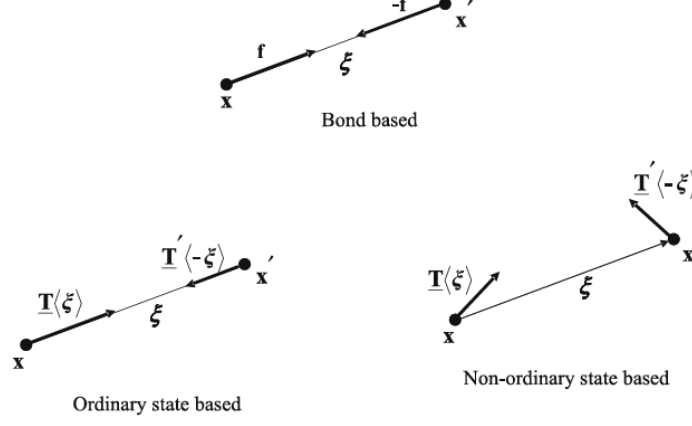


Figure 2.3: Representation of the interaction force density for different PD formulations [8].

2.2 Bond Based formulation

In the BB formulation, the pairwise force density exchanged between two material points $\mathbf{x}$ and $\mathbf{x}'$ is modeled as a spring-like interaction, whose magnitude depends solely on the deformation of the bond connecting the two points, and whose direction coincides with the deformed bond direction itself. By Newton's third law, in order to satisfy the conservation of linear momentum the forces exchanged by the two points are equal and opposite:

$$\mathbf{f}(\mathbf{x}, \mathbf{x}') = -\mathbf{f}(\mathbf{x}', \mathbf{x}). \quad (3)$$

Moreover, conservation of angular momentum requires these forces to be aligned with the line connecting the two points, i.e. with the deformed bond direction along the versor $\mathbf{M}$:

$$\mathbf{f}(\boldsymbol{\xi}, \boldsymbol{\eta}) = f(\boldsymbol{\xi}, \boldsymbol{\eta}) \mathbf{M}, \quad \mathbf{M} = \frac{\boldsymbol{\xi} + \boldsymbol{\eta}}{\|\boldsymbol{\xi} + \boldsymbol{\eta}\|}. \quad (4)$$

The simplest developed BB constitutive model is the *prototype microelastic brittle material (PMB)* [9] that allows to naturally incorporate fracture in the formulation. Bonds can be thought of as linear springs that can break if they exceed their maximum strain. The bond scalar stretch s , equivalent to the strain of the bond, is defined as:

$$s(\boldsymbol{\xi}, \boldsymbol{\eta}) = \frac{\|\boldsymbol{\xi} + \boldsymbol{\eta}\| - \|\boldsymbol{\xi}\|}{\|\boldsymbol{\xi}\|}. \quad (5)$$

For the PMB model, the pairwise force is defined as:

$$\mathbf{f}(\boldsymbol{\xi}, \boldsymbol{\eta}) = cs(\boldsymbol{\xi}, \boldsymbol{\eta})\mu(\boldsymbol{\xi}, \boldsymbol{\eta})\mathbf{M}, \quad (6)$$

where c is the constant *micromodulus*, i.e. is the stiffness of the bond, and μ is a history dependent scalar-valued function illustrated in Eq. (7) used to introduce the concept of *irreversible broken bonds* once time the stretch overcome the critical one defined as s_0 .

$$\mu(\boldsymbol{\xi}, \boldsymbol{\eta}(t)) = \begin{cases} 1 & \text{if } s_{\max}(\boldsymbol{\xi}, \boldsymbol{\eta}(t)) < s_0, \\ 0 & \text{otherwise,} \end{cases} \quad s_{\max}(\boldsymbol{\xi}, \boldsymbol{\eta}(t)) = \max_{0 \leq t' \leq t} s(\boldsymbol{\xi}, \boldsymbol{\eta}(t')). \quad (7)$$

Once the deformed configuration is known, both the actual bond stretch and the direction along which the force is applied can be computed directly. However, determining the boolean variable used to identify broken bonds, as well as the micromodulus, requires prior knowledge of s_0 and c , respectively. These parameters can be obtained by calibration with respect to classical quantities derived from the CCM framework. It can be shown, by imposing a prescribed homogeneous isotropic deformation and equating the resulting PD strain energy density to its CCM counterpart, that the micromodulus c is given by the following expression:

$$c = \begin{cases} \frac{12E}{\pi\delta^4} & \text{in 3D,} \\ \frac{48E}{5\pi t\delta^3} & \text{in 2D plane strain,} \\ \frac{9E}{\pi t\delta^3} & \text{in 2D plane stress,} \\ \frac{2E}{A\delta^2} & \text{in 1D,} \end{cases} \quad (8)$$

where E is the Young's modulus, t is the thickness (in the 2D cases), and A is the cross-sectional area (in the 1D case). The critical bond stretch s_0 (whose definition is illustrated in Figure 2.4) is calibrated by requiring that the work needed to break all the bonds crossing a unit fracture surface equals the material macroscopic fracture energy G_0 [9]. This energy-based criterion leads to the following analytical expressions:

$$s_0 = \begin{cases} \sqrt{\frac{5G_0}{6E\delta}}, & \text{3D,} \\ \sqrt{\frac{5G_0}{12E\delta}}, & \text{2D plane strain,} \\ \sqrt{\frac{4G_0}{9E\delta}}, & \text{2D plane stress,} \\ \sqrt{\frac{3G_0}{E\delta}}, & \text{1D.} \end{cases} \quad (9)$$

Once bond breakage occurs, its macro-effects on the material can be quantified by introducing a local damage metric. The local damage factor φ represents the ratio of broken bonds to the total number of initial bonds within the family horizon $\mathcal{H}_{\mathbf{x}}$ of a given material point $\mathbf{x}$. It is defined as:

$$\varphi(\mathbf{x}, t) = 1 - \frac{\int_{\mathcal{H}_{\mathbf{x}}} \mu(\mathbf{x}, t, \boldsymbol{\xi}) dV_{\boldsymbol{\xi}}}{\int_{\mathcal{H}_{\mathbf{x}}} dV_{\boldsymbol{\xi}}}. \quad (10)$$

The value of the damage factor spans from $\varphi = 0$, indicating an undamaged material state where all bonds are fully functional, to $\varphi = 1$, representing total material degradation where the point $\mathbf{x}$ is completely disconnected from its surrounding neighborhood and unable to transmit internal forces.

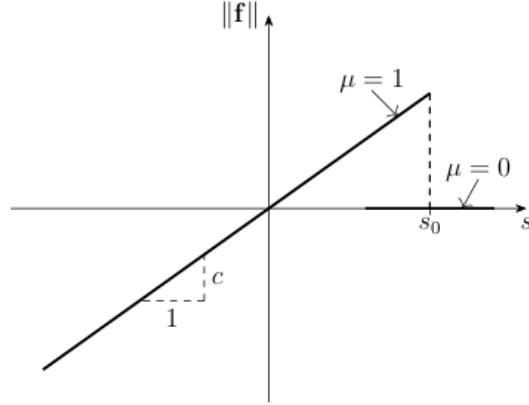


Figure 2.4: Pairwise force density as a function of the bond stretch for the PMB material model [9].

2.3 State Based formulation

In order to obtain a more general material response, it is necessary to introduce inter dependencies between bonds beyond pair interactions. This generalization is provided using the idea of *states*, a mathematical object defined as a function mapping each bond in the family of a point onto a tensor quantity.

Definition 2.3.1 (State). *A state of order m is a function $\underline{\mathbf{A}}\langle\cdot\rangle : \mathcal{H} \rightarrow \mathcal{L}_m$, where $\mathcal{H}$ denotes the family (neighborhood) of a material point and $\mathcal{L}_m$ is the set of tensors of order m . The image of a bond $\boldsymbol{\xi} \in \mathcal{H}$ under the state $\underline{\mathbf{A}}$ is written $\underline{\mathbf{A}}\langle\boldsymbol{\xi}\rangle$, with angle brackets denoting the bond on which the state operates. A state of order 1 is called a vector state (denoted with an underscored uppercase bold symbol, e.g. $\underline{\mathbf{A}}$), while a state of order 0 is called a scalar state (denoted with an underscored lowercase symbol, e.g. $\underline{a}$) [8].*

As example, an important state-quantity in Peridynamics is the *deformation state* $\underline{\mathbf{Y}}$. Given a material point $\mathbf{x}$ and a bond $\boldsymbol{\xi} = \mathbf{x}' - \mathbf{x}$ associated with it, the deformation state associates to this specific bond a vector quantity: the deformed relative position of the bond,

$$\underline{\mathbf{Y}}[\mathbf{x}]\langle\mathbf{x}' - \mathbf{x}\rangle = \mathbf{y}(\mathbf{x}') - \mathbf{y}(\mathbf{x}), \quad (11)$$

that is, the Cartesian components of the relative deformation of the bond itself (note that $\mathbf{y}$ is defined as the vectorial sum between $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$, so that it identifies directly the deformed position of the generic material point $\mathbf{x}$). In this way, $\underline{\mathbf{Y}}$ maps each bond onto its deformed counterpart (Figure 2.5).

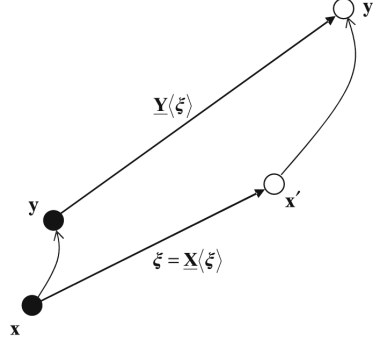


Figure 2.5: Reference position state field $\underline{\mathbf{X}}$ and deformation state field $\underline{\mathbf{Y}}$ [8].

It should be noted that, although numerically the state $\underline{\mathbf{Y}}[\mathbf{x}]$ can be represented as a discrete structure indexed by node and by bond (an array of the form $\mathbf{Y}[\text{node}]\langle\text{bond}\rangle[\text{component}]$), conceptually it remains a continuous functional object $\underline{\mathbf{Y}}[\mathbf{x}]\langle\cdot\rangle : \mathcal{H} \rightarrow \mathbb{R}^3$.

Analogously, the force state $\underline{\mathbf{T}}[\mathbf{x}]$ associates at each bond a force density vector, so that the pairwise force densities exchanged between $\mathbf{x}$ and $\mathbf{x}'$ can be written as

$$\mathbf{t}(\mathbf{x}', \mathbf{x}) = \underline{\mathbf{T}}[\mathbf{x}]\langle\mathbf{x}' - \mathbf{x}\rangle, \quad \mathbf{t}(\mathbf{x}, \mathbf{x}') = \underline{\mathbf{T}}[\mathbf{x}']\langle\mathbf{x} - \mathbf{x}'\rangle. \quad (12)$$

The equation of motion defined for the BB formulation Eq. (1) can therefore be recast in the form of Eq. (13) for the SB formulation. This change in notation arises from the fact that the forces exchanged between two material points are no longer required to be equal in magnitude and opposite in direction. Consequently, the resultant force density exchanged between $\mathbf{x}$ and $\mathbf{x}'$ can no longer be determined solely from the information associated with the bond $\mathbf{x}' - \mathbf{x}$. Instead, both contributions, $\underline{\mathbf{T}}[\mathbf{x}]\langle\mathbf{x}' - \mathbf{x}\rangle$ and $\underline{\mathbf{T}}[\mathbf{x}']\langle\mathbf{x} - \mathbf{x}'\rangle$, depend on the deformation of the entire family of the two nodes involved through aggregate (nonlocal) quantities, rather than exclusively on the deformation of the bond shared.

$$\rho(\mathbf{x}) \ddot{\mathbf{u}}(\mathbf{x}, t) = \int_{\mathcal{H}_x} \mathbf{t}(\mathbf{x}', \mathbf{x}) - \mathbf{t}(\mathbf{x}, \mathbf{x}') dV_{\mathbf{x}'} + \mathbf{b}(\mathbf{x}, t). \quad (13)$$

A material model owing to the SB framework is therefore a state-valued function in the form,

$$\underline{\mathbf{T}} = \hat{\underline{\mathbf{T}}}(\underline{\mathbf{Y}}). \quad (14)$$

Because the force state is no longer required to be aligned with the deformed bond, the balance of angular momentum is not automatically satisfied and must instead be imposed as a constraint on the material model, referred to as *non-polarity*:

$$\int_{\mathcal{H}} \underline{\mathbf{T}}(\underline{\boldsymbol{\xi}}) \times \underline{\mathbf{Y}}(\underline{\boldsymbol{\xi}}) dV_{\boldsymbol{\xi}} = \mathbf{0}. \quad (15)$$

A material is called *ordinary* if the force state is always directed parallel to the deformed bond, i.e. if there exists a scalar-valued state $\underline{g}$ such that $\underline{\mathbf{T}} = \underline{g} \underline{\mathbf{Y}}$; in this case non-polarity is automatically satisfied. It is straightforward to show that the Bond-Based theory is recovered as a special case of an ordinary State-Based material.

Linear Peridynamic Solid

A convenient class of ordinary elastic materials is obtained by assuming that the strain energy density depends on a weighted average of the bond extensions. Defining the extension state $\underline{e}(\underline{\boldsymbol{\xi}}) = |\underline{\mathbf{Y}}(\underline{\boldsymbol{\xi}})| - |\underline{\boldsymbol{\xi}}|$, a nonlocal dilatation ϑ can be introduced as

$$\vartheta = \frac{c_\theta}{\hat{m}} \int_{\mathcal{H}} \underline{\omega}(\underline{\boldsymbol{\xi}}) |\underline{\boldsymbol{\xi}}| \underline{e}(\underline{\boldsymbol{\xi}}) dV_{\boldsymbol{\xi}}, \quad \hat{m} = \int_{\mathcal{H}} \underline{\omega}(\underline{\boldsymbol{\xi}}) |\underline{\boldsymbol{\xi}}|^2 dV_{\boldsymbol{\xi}}, \quad (16)$$

where $\underline{\omega}$ is a radially symmetric *influence function* and $\hat{m}$ is a normalization constant known as the weighted volume (here, the notation $\hat{\cdot}$ is used to distinguish the weighted volume $\hat{m}$ from the standard variable m introduced later in the chapter). The factor c_θ is derived so that, for a small isotropic expansion, the nonlocal dilatation agrees with the classical local dilatation, which imposed for the 3D case that $c_\theta = 3$. If the strain energy density is assumed to depend on the deformation state only through ϑ , the corresponding

force state is found to be

$$\underline{\mathbf{T}}\langle\boldsymbol{\xi}\rangle = \frac{-3p\omega\langle\boldsymbol{\xi}\rangle|\boldsymbol{\xi}|}{\hat{m}}\underline{\mathbf{M}}, \quad p = -\frac{\partial W}{\partial \vartheta}, \quad (17)$$

with $\underline{\mathbf{M}} = \underline{\mathbf{Y}}/|\underline{\mathbf{Y}}|$ the deformed bond direction. Choosing the simple linear pressure law $p = -k\vartheta$, with k the classical bulk modulus, defines the *linear peridynamic fluid (LPF)* [8]: since the nonlocal pressure is mapped isotropically onto the bond forces, this model can only reproduce a purely volumetric (fluid-like) response. To extend the model to solid material behavior, the deviatoric part of the deformation is isolated by subtracting the dilatational contribution from the extension state,

$$\underline{e}^d\langle\boldsymbol{\xi}\rangle = \underline{e}\langle\boldsymbol{\xi}\rangle - \frac{\vartheta|\boldsymbol{\xi}|}{3}, \quad (18)$$

which vanishes identically under a purely isotropic expansion. Adding a term proportional to $\underline{e}^d$ to the fluid response yields the *linear peridynamic solid (LPS)* material model:

$$\underline{\mathbf{T}}\langle\boldsymbol{\xi}\rangle = \left(\frac{3k\vartheta\omega\langle\boldsymbol{\xi}\rangle|\boldsymbol{\xi}|}{\hat{m}} + \alpha\omega\langle\boldsymbol{\xi}\rangle\underline{e}^d\langle\boldsymbol{\xi}\rangle \right) \underline{\mathbf{M}}, \quad (19)$$

where α is calibrated against the shear modulus G of the material equating the strain energy density in an infinite body modeled with OSB-PD with the one obtained from CCM under homogeneous deformation. Given the assumption of infinite body, the calibration maintain its accuracy sufficiently far from any boundary or interface, providing the simple closed-form result

$$\alpha = \frac{15G}{\hat{m}}. \quad (20)$$

The choice of the influence function ω , although largely arbitrary, also affects the accuracy and smoothness of the discretized model, and has been the subject of dedicated numerical studies [10].

Correspondence Materials

Before going deeper into the PD constitutive laws investigated to model the material softening behavior, it is useful to briefly discuss a further class of State-Based material models, known as *correspondence materials*. This model is not employed in the present work; it is introduced here purely as a term of comparison, as it is already capable of performing the task investigated in this thesis, i.e., directly connecting CCM theory to PD without relying on calibration procedures, although at a significantly higher computational cost than the OSB formulation, and even more so than the BB formulation adopted throughout this thesis.

The underlying idea of a correspondence material is to adapt a classical (local) constitutive model directly into the PD framework. Given a classical elastic material described by a *Piola stress tensor* $\sigma(\mathbf{J})$, where $\mathbf{J} = \partial \mathbf{y} / \partial \mathbf{x}$ is the deformation gradient tensor, an approximate deformation gradient $\bar{\mathbf{J}}$ is reconstructed at each material point from the deformation of its family:

$$\bar{\mathbf{J}} = \left(\int_{\mathcal{H}} \underline{\omega}(\boldsymbol{\xi}) \underline{\mathbf{Y}}(\boldsymbol{\xi}) \otimes \boldsymbol{\xi} dV_{\xi} \right) \mathbf{S}^{-1}, \quad \mathbf{S} = \int_{\mathcal{H}} \underline{\omega}(\boldsymbol{\xi}) \boldsymbol{\xi} \otimes \boldsymbol{\xi} dV_{\xi}, \quad (21)$$

where $\mathbf{S}$ is the (always symmetric, and, provided $\underline{\omega}$ is positive on $\mathcal{H}$, positive definite) shape tensor. Once $\bar{\mathbf{J}}$ is known, the classical stress tensor $\sigma(\bar{\mathbf{J}})$ is evaluated and mapped back onto the bond forces:

$$\underline{\mathbf{T}}(\boldsymbol{\xi}) = \underline{\omega}(\boldsymbol{\xi}) \sigma(\bar{\mathbf{J}}) \mathbf{S}^{-1} \boldsymbol{\xi}. \quad (22)$$

Based on this formulation, the correspondence model in principle allows essentially *any* classical constitutive model to be reused directly within a PD setting, without the need to reformulate it in terms of Bond-Based or State-Based quantities. However, this generality comes at a significant computational cost, for several reasons:

- **Per-point tensor inversion.** Correspondence approach requires the computation and inversion of the shape tensor $\mathbf{S}$ at *every* material point of the discretized body. While $\mathbf{S}$ depends only on the reference configuration and can therefore be precomputed (and also its inverse $\mathbf{S}^{-1}$) once

for a fixed discretization, the approximate deformation gradient $\bar{\mathbf{J}}$ must be reconstructed at every time step through a full tensor (dyadic) product integral over the family, followed by a matrix-vector product with $\mathbf{S}^{-1}$, providing a considerably heavy operation.

- **Evaluation of a full classical constitutive model.** Once $\bar{\mathbf{J}}$ is available, the classical stress $\sigma(\bar{\mathbf{J}})$ must be evaluated pointwise, which, for any non-trivial material law (e.g. hyperelastic or elastic-plastic models), can itself be computationally expensive, and must be repeated identically for every point in every family, effectively nesting a classical constitutive evaluation inside the outer peridynamic integral.
- **Zero-energy modes and their suppression.** Perhaps the most consequential practical difficulty is that the mapping from bond deformations to $\bar{\mathbf{J}}$ is not invertible: multiple distinct deformations of a family can produce the same $\bar{\mathbf{J}}$, and therefore the same force state. This non-uniqueness manifests numerically as spurious zero-energy modes [11] (uncontrolled oscillatory deformation patterns that do not affect the computed forces), which must be actively suppressed to obtain a physically meaningful solution. The simplest remedy described in [12], the inclusion of *background bonds*, requires blending the correspondence force state with an auxiliary Bond-Based (spring-like) contribution,

$$\underline{\mathbf{T}}\langle\boldsymbol{\xi}\rangle = (1 - \gamma_z) \underline{\omega}\langle\boldsymbol{\xi}\rangle \sigma(\bar{\mathbf{J}}) \mathbf{S}^{-1} \boldsymbol{\xi} + \gamma_z \frac{c \underline{e}\langle\boldsymbol{\xi}\rangle}{|\boldsymbol{\xi}|}, \quad (23)$$

where γ_z is a small stabilization parameter and c is the Bond-Based micromodulus previously defined.

Taken together, these aspects make correspondence materials considerably more demanding, in terms of both computational cost and implementation complexity, than the OSB and BB model discussed in the previous sections. In light of these critical issues, this thesis investigates an alternative approach to the correspondence model, aimed at connecting classical theory to Peridynamics at a much lower computational overhead.

2.4 Constitutive law for quasi-brittle material

The constitutive model employed in this study is specifically formulated to replicate the fracture response characteristic of quasi-brittle materials [13]. Unlike perfectly brittle materials, where bond failure occurs instantaneously once the critical stretch s_0 is reached, quasi-brittle materials exhibit a progressive degradation of their load-carrying capacity after the elastic limit. This macroscopic behavior is reproduced by introducing softening PD constitutive laws.

In all the constitutive models considered, the bond response is history-dependent and is governed by the maximum stretch attained during the loading history, denoted by $s_{\max}$. During loading, the bond follows a predefined softening backbone after the elastic threshold s_0 is exceeded, while during unloading and subsequent reloading it follows a linear path connecting the origin to the current point on the backbone. Consequently, the bond stiffness progressively decreases as damage accumulates, while complete failure occurs when the ultimate stretch s_c is reached. Different softening laws are considered in this work, namely a bilinear, an exponential and a trilinear model, as illustrated in Figure 2.6.

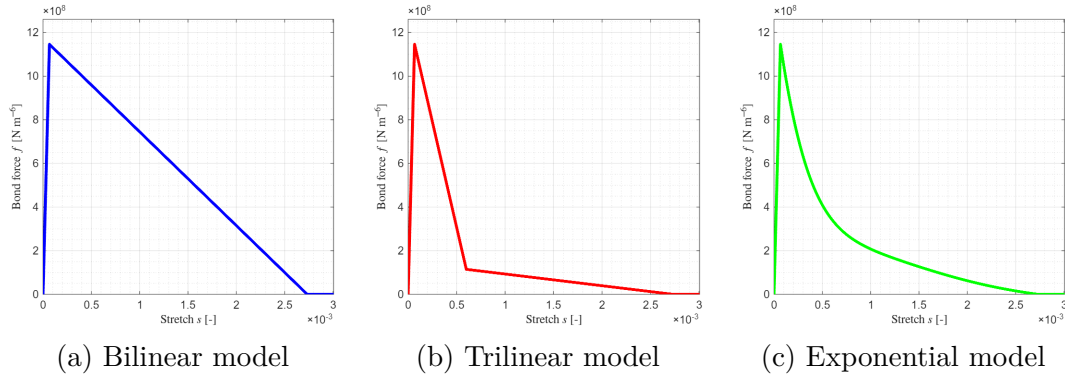


Figure 2.6: PD constitutive laws adopted.

Bilinear model The bilinear constitutive law (Figure 2.6a) is the simplest softening model considered. The bond exhibits a linear elastic response up

to the critical elastic stretch s_0 , after which the transmitted force decreases linearly until it vanishes at the ultimate stretch s_c .

$$f(\boldsymbol{\eta}, \boldsymbol{\xi}) = \begin{cases} cs, & s \leq s_{\max} \leq s_0, \\ c \left(1 - \frac{s - s_0}{s} \frac{s_c}{s_c - s_0} \right) s, & s_0 < s_{\max} \leq s_c, s_{\max} = s, \\ c \left(1 - \frac{s_{\max} - s_0}{s_{\max}} \frac{s_c}{s_c - s_0} \right) s, & s_0 < s_{\max} \leq s_c, s < s_{\max}, \\ 0, & s_{\max} > s_c, \end{cases} \quad (24)$$

where $s_{\max}$ is the maximum stretch attained by the bond throughout the loading history. The first branch describes the linear elastic response, the second and third branches describe, respectively, the loading and the unloading/reloading paths within the softening regime, and the last branch enforces complete bond failure once s_c is exceeded.

Trilinear model The trilinear constitutive law (Figure 2.6b) introduces an additional branch within the softening regime, allowing greater flexibility in shaping the post-peak response. After the initial elastic branch, the force decreases with a first softening slope until an intermediate stretch, denoted as *kink point stretch* s_k is reached,

$$s_k = s_0 + \frac{s_c - s_0}{\gamma}, \quad (25)$$

in correspondence of which the force is defined by $f_k = cs_0\beta$, followed by a second softening branch with a different degradation rate before complete bond failure at s_c . The two additional degrees of freedom introduced at the kink point enables a more accurate representation of materials exhibiting multiple damage mechanisms or a pronounced change in stiffness during crack propagation. The parameters γ and β are intrinsic material properties that can be determined through experimental calibration procedures. Well-known standard values are $\gamma = 1/4$ and $\beta = 20/3$, as further discussed in Section 5.1. In its general form, the trilinear law can be written as

$$f(\boldsymbol{\eta}, \boldsymbol{\xi}) = \begin{cases} cS, & s \leq s_{\max} \leq s_0, \\ cS_0 \left[1 - \frac{\gamma(\beta-1)}{s_c-s_0}(s-s_0) \right], & s_0 < s_{\max} \leq s_k, \ s_{\max} = s, \\ cS_0 \gamma \beta \frac{s_c-s}{(s_c-s_0)(\gamma-1)}, & s_k < s_{\max} \leq s_c, \ s_{\max} = s, \\ \frac{f|_{s_{\max}}}{s_{\max}} s, & s_0 < s_{\max} \leq s_c, \ s < s_{\max}, \\ 0, & s_{\max} > s_c. \end{cases} \quad (26)$$

Exponential model The exponential constitutive law provides a smooth degradation of the bond force after the elastic limit for which the post-peak response follows an exponential decay calibrated to reproduce the desired fracture energy. This continuous evolution avoids the discontinuity in stiffness associated with piecewise-linear models and offers a more realistic description of the gradual damage accumulation observed in many quasi-brittle materials. Following the formulation adopted in [14] the pairwise force function reads

$$f(\boldsymbol{\eta}, \boldsymbol{\xi}) = \begin{cases} cS, & s \leq s_{\max} \leq s_0, \\ cS_0 \gamma_0, & s_0 < s_{\max} \leq s_c, \ s_{\max} = s, \\ cS_0 \frac{s}{s_{\max}} \gamma_1, & s_0 < s_{\max} \leq s_c, \ s < s_{\max}, \\ 0, & s_{\max} > s_c, \end{cases} \quad (27)$$

where the softening functions γ_0 and γ_1 that are given by

$$\gamma_0 = \left(1 + (c_1 \hat{s}_0)^3 \right) \exp(-c_2 \hat{s}_0) - \left(1 + c_1^3 \right) \hat{s}_0 \exp(-c_2), \quad \hat{s}_0 = \frac{s-s_0}{s_c-s_0}; \quad (28)$$

$$\gamma_1 = \left(1 + (c_1 \hat{s}_1)^3 \right) \exp(-c_2 \hat{s}_1) - \left(1 + c_1^3 \right) \hat{s}_1 \exp(-c_2), \quad \hat{s}_1 = \frac{s_{\max}-s_0}{s_c-s_0}. \quad (29)$$

c_1, c_2 are material-dependent softening parameters, set respectively to 3.00 and 6.93 for plain concrete as recommended in [14, 15].

Calibration of s_0 and s_c For all degrading models, the critical stretch s_0 , which differs from the previously defined expression in Eq. 9 for brittle materials, is determined by establishing a relationship between the bond stretch s and the tensile strength of the material f_t . Following [14], the normal stress component on a given surface, evaluated at the limit of linear elasticity, is set equal to f_t . Under this condition, all bonds intersecting that surface are assumed to experience a uniform stretch, such that $s = s_0$.

$$s_0 = \begin{cases} \frac{f_t}{2E}, & 3\text{D}, \\ \frac{5f_t}{9E}, & 2\text{D plane stress}, \\ \frac{5f_t}{12E}, & 2\text{D plane strain}, \\ \frac{f_t}{E}, & 1\text{D}. \end{cases} \quad (30)$$

The failure stretch s_c is obtained by equating G_0 to the total energy required to break the bond $0 \leq s \leq s_c$ which provides different analytical outcome for each of the three models:

$$\text{Bilinear Softening: } s_c = \begin{cases} s_0 + \frac{5G_0}{6E\delta s_0}, & 3\text{D}, \\ s_0 + \frac{4G_0}{9E\delta s_0}, & 2\text{D plane stress}, \\ s_0 + \frac{5G_0}{12E\delta s_0}, & 2\text{D plane strain}, \\ s_0 + \frac{3G_0}{E\delta s_0}, & 1\text{D}. \end{cases} \quad (31a)$$

$$\text{Exponential Softening: } s_c = \begin{cases} s_0 + 2.140 \frac{G_0}{E\delta s_0}, & 3\text{D}, \\ s_0 + 3.586 \frac{G_0}{E\delta s_0}, & 2\text{D plane stress}, \\ s_0 + 3.361 \frac{G_0}{E\delta s_0}, & 2\text{D plane strain}, \\ s_0 + 7.704 \frac{G_0}{E\delta s_0}, & 1\text{D}. \end{cases} \quad (31b)$$

$$\text{Trilinear Softening : } s_c = \begin{cases} s_0 + \frac{G_0}{4E\delta s_0}, & 3\text{D}, \\ s_0 + \frac{2G_0}{15E\delta s_0}, & 2\text{D plane stress}, \\ s_0 + \frac{G_0}{8E\delta s_0}, & 2\text{D plane strain}, \\ s_0 + \frac{9G_0}{20E\delta s_0}, & 1\text{D}. \end{cases} \quad (31c)$$

Although the three constitutive laws differ in the evolution of the post-peak response, they are calibrated to share the same Young's modulus E , tensile strength f_t and fracture energy G_0 . Therefore, the influence of the shape of the softening branch on the structural response can be investigated independently of the elastic properties and the total energy dissipated during fracture.

2.5 Implementation strategies

To numerically solve the PD equation of motion, Eq. (1), while mitigating potential numerical errors, a series of dedicated computational strategies, described below, are commonly adopted. The first step required is the discretization of the integro-differential equation. Among the numerical strategies available in the literature for this purpose, the *meshfree method* is the most widely adopted, and is the one employed throughout this work. In the meshfree method, a generic body $\mathcal{B}$ is discretized into a finite collection of material points $\mathbf{x}_i$, arranged on a uniform grid with spacing Δx in each direction, as

shown in Figure 2.7 for a two-dimensional case. Each node $\mathbf{x}_i$ is associated with a finite volume ΔV , such that the union of all nodal volumes reconstructs the volume of the entire body. For the two-dimensional plane-stress models considered in this work, this volume is given by $\Delta V = t \Delta x^2$, where t denotes the thickness of the body. The horizon δ is expressed in terms of the grid spacing through the dimensionless parameter $m = \delta/\Delta x$, commonly referred to as the *horizon factor* or *m-ratio*, which governs the number of neighboring nodes $\mathbf{x}_j$ interacting with each source node $\mathbf{x}_i$.

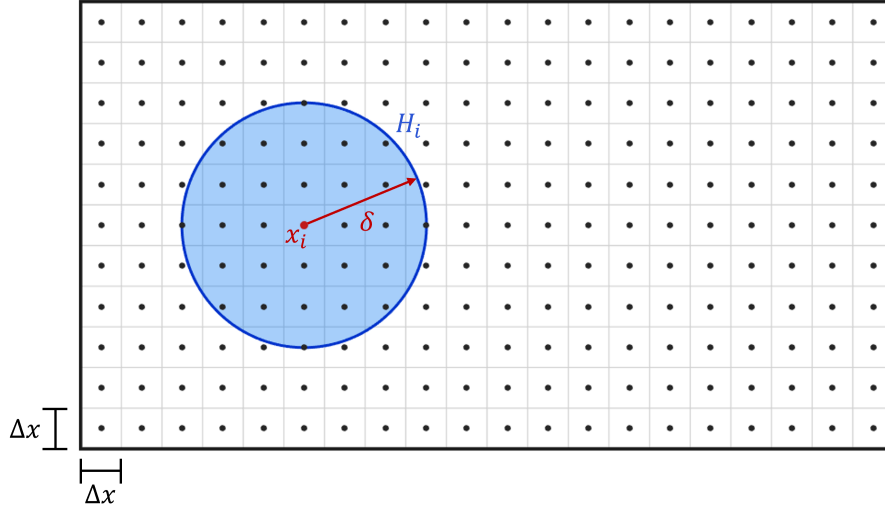


Figure 2.7: Schematic representation of the meshfree spatial discretization of a two-dimensional domain with uniform grid spacing Δx .

The spatial integration over the neighborhood $\mathcal{H}_i$ is then approximated by a finite summation over all family nodes $\mathbf{x}_j \in \mathcal{H}_i$, using a midpoint quadrature rule, where cell centers are used as quadrature points and volumes are used as weights. Applying this discretization to the PD equation of motion, the discretized form at node $\mathbf{x}_i$ and time step n reads

$$\rho(\mathbf{x}_i) \ddot{\mathbf{u}}_i^n = \sum_{j \in \mathcal{H}_i} \mathbf{f}(\mathbf{u}_j^n - \mathbf{u}_i^n, \mathbf{x}_j - \mathbf{x}_i) \Delta V + \mathbf{b}_i^n. \quad (32)$$

As a consequence of the discretization, a finite network of bonds is formed, connecting each node to every family node with which it interacts. This network is established once, at the beginning of the analysis, and its connectivity

remains fixed throughout the deformation, i.e. although individual bonds may stretch, rotate and break, no further bonds are created as the body deforms.

2.5.1 Volume Correction

In the discretized formulation of Eq. (32), each family node $\mathbf{x}_j \in \mathcal{H}_i$ is assigned its nodal volume ΔV_j in full, implicitly assuming that the entire cell associated with $\mathbf{x}_j$ lies within the horizon δ of the source node $\mathbf{x}_i$. This assumption, however, only holds for cells sufficiently far from the boundary of the neighborhood: since $\mathcal{H}_i$ is a circular (or spherical, in 3D) region, its boundary does not coincide with the edges of the underlying grid, so that cells located near the edge of the horizon are only partially contained within it (Figure 2.8). Assigning the full cell volume to these boundary nodes would systematically overestimate the effective interaction volume, while entirely excluding them would just as systematically underestimate it, and in both cases a spurious dependence of the numerical solution on the grid resolution is introduced.

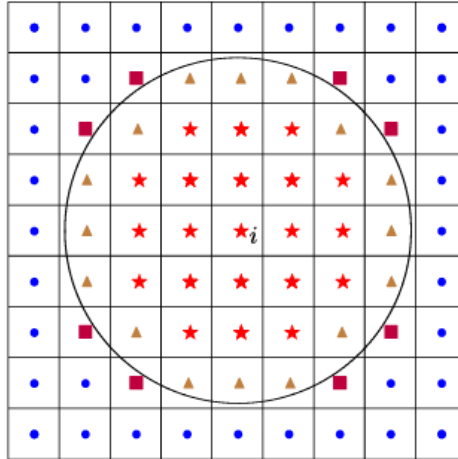


Figure 2.8: Categorization of family nodes according to their relative cell position with respect to the horizon boundary: fully contained internal nodes (stars), partially intersected boundary nodes (triangles and squares), and external nodes (circles) [16].

To correct for this effect, a *volume correction coefficient* $\beta_{ij} \in [0, 1]$ is introduced, representing the fraction of the cell volume associated with node $\mathbf{x}_j$ that actually lies within $\mathcal{H}_i$:

$$\rho(\mathbf{x}_i) \ddot{\mathbf{u}}_i^n = \sum_j \left[\mathbf{f}(\mathbf{u}_j^n - \mathbf{u}_i^n, \mathbf{x}_j - \mathbf{x}_i) \cdot \beta_{ij} \Delta V_j \right] + \mathbf{b}_i^n. \quad (33)$$

The simplest adopted approximation for β_{ij} is a piecewise-linear function of the bond length $|\boldsymbol{\xi}|$:

$$\beta_{ij} = \begin{cases} 1, & |\boldsymbol{\xi}| \leq \delta - 0.5 \Delta x, \\ \frac{\delta + 0.5 \Delta x - |\boldsymbol{\xi}|}{\Delta x}, & \delta - 0.5 \Delta x < |\boldsymbol{\xi}| \leq \delta + 0.5 \Delta x, \\ 0, & |\boldsymbol{\xi}| > \delta + 0.5 \Delta x. \end{cases} \quad (34)$$

This formulation assumes $\beta_{ij} = 1$ for family nodes whose cell lies entirely within the neighborhood, $\beta_{ij} = 0$ for nodes lying entirely outside it, and a linear interpolation in between, based solely on the distance of $\mathbf{x}_j$ from $\mathbf{x}_i$ rather than on the actual geometry of the cell-horizon intersection. While computationally inexpensive, this approximation does not account for the true shape of the intersected area, which in general depends not only on the distance $|\boldsymbol{\xi}|$ but also on the specific position of the cell relative to the circular boundary of $\mathcal{H}_i$.

A more accurate alternative consists in computing the *exact* fraction of intersected area (or volume, in 3D) for every family node, defined as

$$\beta_{ip} = \begin{cases} \frac{A_{\text{intersection}}}{\Delta x^2}, & |\boldsymbol{\xi}| \leq \delta + 0.5 \Delta x, \\ 0, & |\boldsymbol{\xi}| > \delta + 0.5 \Delta x, \end{cases} \quad (35)$$

where $A_{\text{intersection}}$ is the area of the region obtained by intersecting the square cell of node $\mathbf{x}_j$ with the circle of radius δ centered at $\mathbf{x}_i$. This exact evaluation requires decomposing the cell-horizon intersection into elementary geometric shapes, whose closed-form expressions depend on the number of cell vertices lying inside the horizon – ranging from cells entirely inside or outside $\mathcal{H}_i$, to cells with one, two, or three vertices inside the circular boundary, each case requiring a dedicated geometric construction [16]. Compared to the linear approximation of Eq. (34), this approach yields the true value of β_{ij} for every

family node, at the cost of a considerably more involved pre-processing stage, which however can be done one single time and then the results can be used through a *lock-up table* approach.

2.5.2 Surface Effect Correction

As discussed in the previous section, the expression for the elastic constant c is derived under the assumption that the material point $\mathbf{x}$ lies in the interior of a homogeneous body, sufficiently far from any boundary, that its horizon $\mathcal{H}_{\mathbf{x}}$ is entirely contained within the body domain. When this condition is not satisfied, i.e. when the point lies close to a free surface, its family of bonds is incomplete, since interactions with points outside the body are missing, as shown in Figure 2.9. This results in a local reduction of the strain energy density with respect to the case of a complete *horizon*, and consequently in a decrease in stiffness within the surface region: a phenomenon known in the literature as the *surface effect*, or *surface softening* [17]. In Bond-Based Peridynamics, this effect acts on an external layer of thickness δ , whereas in the OSB formulation the affected layer has thickness 2δ , since in this case the interaction between two points also depends on their respective families.

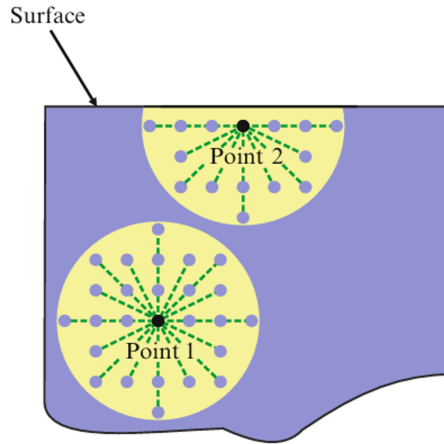


Figure 2.9: Peridynamic surface effect: points close to the boundary of the body, such as *Point 2*, have an incomplete neighborhood [18].

Several strategies have been proposed in literature to mitigate the surface effect. A first approach consists in modifying the mechanical properties of the bonds close to the boundary, so as to compensate for the reduction in strain energy caused by the incomplete *horizon*. A second, alternative approach relies instead on the introduction of a *fictitious layer* of thickness δ (or 2δ in the OSB case) surrounding the body, whose displacements are extrapolated from those of the actual points of the body, so as to artificially complete the neighborhoods of the points close to the boundary without altering the constitutive properties of the bonds. In the specific case of BB *Prototype Microelastic PM* and PMB materials, a practical and widely adopted correction [12, 17] consists in directly modifying the constant c through a correction factor λ_b , defined as the ratio between the nominal volume of the horizon sphere $V_0 = \frac{4}{3}\pi\delta^3$ and the average of the volumes of material actually contained within the families $\mathcal{H}_{\mathbf{x}}$ and $\mathcal{H}_{\mathbf{q}}$ of the two interacting points:

$$c^* = c\lambda_b, \quad \lambda_b = \frac{2V_0}{V(\mathbf{x}) + V(\mathbf{q})}, \quad V_0 = \frac{4\pi\delta^3}{3}, \quad (36)$$

where $V(\mathbf{x})$ and $V(\mathbf{q})$ denote the volumes of material contained in $\mathcal{H}_{\mathbf{x}}$ and $\mathcal{H}_{\mathbf{q}}$, respectively, while c^* here is defined as the bulk micromodulus c corrected by the factor λ_b . This correction ensures that, under isotropic expansion, the strain energy density obtained by summing the individual bond contributions matches the value predicted by classical theory, regardless of whether the family fully occupies the horizon sphere or not. The practical effect of the correction is a stiffening of the bonds close to the boundary, proportional to the volumetric deficit of the family with respect to the case of a complete horizon.

2.5.3 Peridynamics force flux

In contrast to the CCM framework, where the Cauchy stress tensor is defined locally as the force per unit area transmitted across an infinitesimal surface element, Peridynamics is inherently nonlocal and does not admit a stress tensor. In order to recover a quantity directly comparable with the classical stress, the concept of *peridynamic flux* is introduced. A peridynamic flux represents

the resultant interaction force transmitted through a virtual surface by all the bonds interactions crossing it.

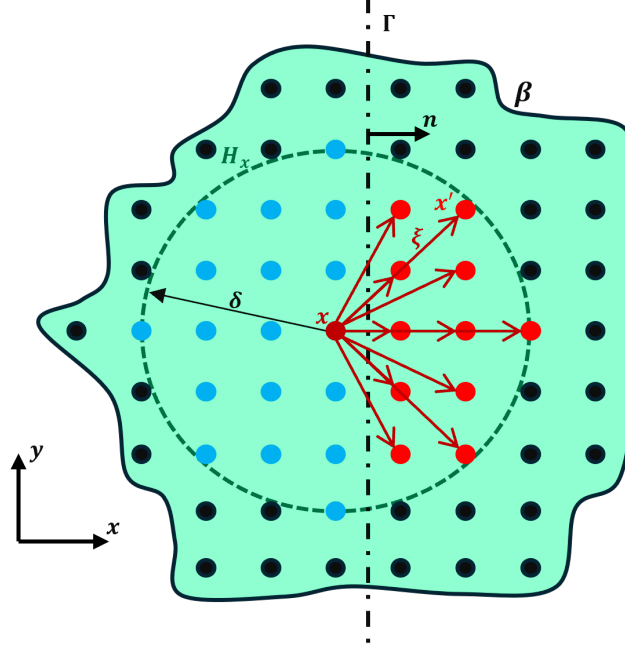


Figure 2.10: Peridynamic force flux.

Let's consider a virtual cross-section Γ , as the one illustrated in Figure 2.10. Let x be an arbitrary material point whose horizon $\mathcal{H}_x$ intersects the section. Only the bonds connecting x with material points located on the opposite side of Γ contribute to the force transmitted across the section. By repeating this procedure for every material point whose horizon intersects the section, and summing the contributions of all active bonds crossing it, the total internal force F_Γ transmitted across Γ is obtained:

$$F_\Gamma = \sum_{i \in \Omega^-} \sum_{j \in \Omega^+} \mu_{ij} (\mathbf{f}_{ij} \cdot \mathbf{n}) \beta_{ij} \Delta V^2, \quad (37)$$

where Ω^- and Ω^+ denote the material points located on the two sides of the section and $\mathbf{n}$ is the unit normal to the section, such that $\mathbf{f}_{ij} \cdot \mathbf{n}$ corresponds to the components of the pairwise force normal to the section Γ . Finally, the average normal stress is evaluated by dividing the resultant force F_Γ by the resistant cross-sectional area A .

2.5.4 Peridynamic convergence analysis type

The nonlocality of the implemented model depends on horizon size, which in turn can either be calibrated physically against experimental data or chosen purely as a numerical parameter to reduce computational overhead. Generally horizon is selected as a multiple of the grid spacing Δx , i.e. m is found to be an integer. In Peridynamics, three possible types of convergence analysis (graphically explained in Figure 2.11) can be defined [19]:

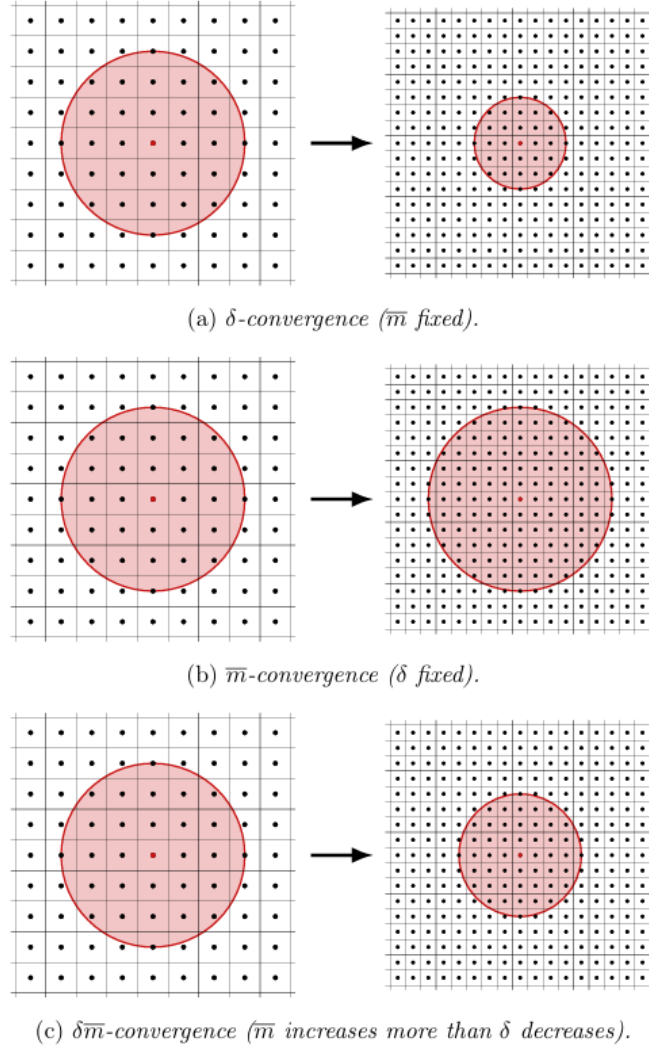


Figure 2.11: Possible types of PD convergence analysis [5].

- **δ -convergence:** the horizon size δ decreases while the m -ratio is kept fixed. As shown in Figure 2.11(a), the grid spacing Δx decreases while the number of nodes within the neighborhood remains constant. In this case, the numerical PD solution converges to the solution obtained with CCM.
- **m -convergence:** the m -ratio increases while δ is kept fixed. As shown in Figure 2.11(b), the number of nodes within the neighborhood increases as the grid spacing Δx decreases. In this case, the numerical solution converges to the exact nonlocal continuous PD solution for the given δ .
- **δm -convergence:** the horizon size δ decreases while m increases faster than δ decreases, so that the ratio m/δ increases, as shown in Figure 2.11(c). In this case, the numerical solution converges uniformly to both the exact PD solution and the CCM solution.

The third type of convergence analysis is less frequently employed in practice, since the effects due to the decrease of δ and the increase of m are intertwined and therefore difficult to distinguish from one another. In standard applications, m -convergence is preferred for verifying spatial discretization accuracy for a given physical horizon, whereas δ -convergence is adopted to assess convergence toward the CCM solution. In this work, δ -convergence was initially performed to verify the correct behavior of the numerical solver described in Chapter 4 and ensure its convergence toward the classical continuum limit. Subsequently, an m -convergence study was conducted to determine the optimal grid discretization and horizon factor to be implemented in the actual numerical simulations, balancing solution accuracy and computational efficiency.

Chapter 3

Neural Networks

This chapter presents the theoretical foundations of Neural Networks (NN) required to comprehend both the computational framework developed and the specific architectural choices adopted in this thesis for the identification of PD constitutive laws.

3.1 Overview of Machine Learning

Machine learning (ML) is the branch of artificial intelligence concerned with the design of algorithms that improve their performance on a given task through exposure to data, rather than through explicitly programmed rules. Following the widely adopted definition given by Tom M. Mitchell [20],

a computer program is said to learn from experience E with respect to a task T and a performance measure P , if its performance at T , as measured by P , improves with E .

In the context of this thesis, the task T is the identification of the PD constitutive law (and its parameters) underlying an observed stress–strain response, the experience E is a large dataset of simulated stress–strain curves with known ground-truth constitutive laws, and the performance measure P is the accuracy with which the trained model recovers the correct law and parameters on unseen data. Machine learning algorithms are commonly grouped into paradigms according to the nature of the data available during training and the feedback signal used to guide learning. The three principal paradigms (summarized in Figure 3.1) are:

Supervised learning. The learning system is supplied with input variables $\mathbf{x}$, referred to as *features or characteristics*, together with their corresponding

target outputs $\mathbf{y}$, referred to as *labels*, which collectively define the desired functional mapping from inputs to outputs. Given a labeled dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ constituted by N samples, the goal of supervised learning is to find a function f_{θ} , parameterized by a set of trainable parameters θ and a set of fixed hyperparameter, that approximates the underlying relationship between $\mathbf{x}$ and $\mathbf{y}$ as closely as possible, according to a chosen loss function $\mathcal{L}$. Depending on the nature of the target $\mathbf{y}$, supervised learning problems are further divided into *classification* and *regression*:

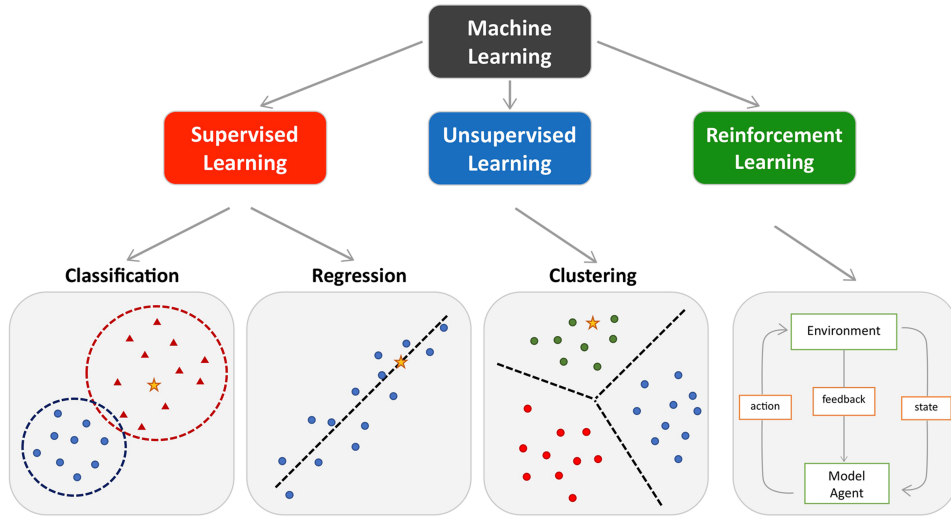


Figure 3.1: Taxonomy of machine learning paradigms.

- **Classification.** In a classification problem, the target variable $\mathbf{y}$ takes values in a finite set of discrete classes. Given an input $\mathbf{x}$, the model learns a mapping that assigns the input to one of the possible classes C . In practice, the network outputs a vector of class probabilities:

$$\hat{\mathbf{p}} = f_{\theta}(\mathbf{x}), \quad (38)$$

with

$$\hat{p}_c \in [0, 1], \quad \sum_{c=1}^C \hat{p}_c = 1.$$

Each component $\hat{p}_c$ represents the estimated probability that the input belongs to class c . The predicted class is then selected as the one asso-

ciated with the highest probability:

$$\hat{y} = \arg \max_c \hat{p}_c. \quad (39)$$

Classification is therefore appropriate when the possible outcomes are distinct categories rather than continuous quantities. In this thesis, it is used to identify which of the candidate PD softening laws (Section 2.4) is associated with a given stress–strain curve.

- **Regression.** In a regression problem, the target variable is continuous rather than discrete. The objective is to learn a functional relationship between an input $\mathbf{x}$ and one or more continuous target variables. The model therefore provides a numerical prediction:

$$\hat{\mathbf{y}} = f_{\boldsymbol{\theta}}(\mathbf{x}), \quad \hat{\mathbf{y}} \in \mathbb{R}^m, \quad (40)$$

where m denotes the number of continuous quantities predicted by the model. Unlike classification, where the output represents a discrete class, regression provides a continuous estimate whose accuracy can be evaluated by measuring the discrepancy between the predicted and target values.

Regression can therefore be used to infer physical or geometrical parameters directly from observed data. In this thesis, it is employed to estimate the continuous material parameters of the identified PD constitutive law from the corresponding stress–strain curve.

Unsupervised learning. In the absence of *labels*, unsupervised algorithms aim to discover the intrinsic structure of the data distribution $p(\mathbf{x})$ itself. Typical tasks include clustering (grouping similar samples together, e.g. *k*-means), dimensionality reduction (finding compact latent representations, e.g. *principal component analysis (PCA)* or autoencoders), and density estimation. The underlying goal is to discover implicit relationships within the available data so as to group it appropriately, revealing distinctive properties along which the data separates naturally, without any external supervision indicating which

grouping or separation is the correct one.

Reinforcement learning. Unlike the previous paradigms, reinforcement learning does not rely on a static dataset. Instead, an *agent* interacts sequentially with an *environment*: at each time step it observes a *state*, selects an *action* according to a policy, and receives a *reward* together with a new state. The objective is to learn the policy that maximizes the expected cumulative (discounted) reward. RL is particularly suited to sequential decision-making problems (e.g. control, game playing).

3.1.1 Training, validation and test sets

To obtain a reliable estimate of how a trained model will perform on new, unseen data, the available dataset $\mathcal{D}$ is customarily partitioned into three disjoint subsets:

- the **training set**, used to iteratively update the model parameters θ by minimizing the loss function $\mathcal{L}$;
- the **validation set**, used during training to monitor generalization performance on data not used for parameter updates, guiding decisions such as hyperparameter selection, early stopping (Section 3.1.2), and model architecture choices;
- the **test set**, which is kept completely separate until training and model selection have finished, and is used a single time to provide a final, unbiased estimate of generalization performance.

A common split ratio is 70–80% of the data for training [21], with the remainder divided between validation and test; the exact proportions used for the datasets generated in this work are detailed in Section 5.5.

3.1.2 Generalization and overfitting

The ultimate goal of supervised learning is not to minimize the loss on the training set, but to achieve good *generalization*. A model that fits the training

data increasingly well while its performance on the validation set stagnates or degrades is said to be *overfitting*: rather than learning the underlying signal, the model has begun to fit noise or artifacts specific to the training examples (left column images of Figure 3.2).

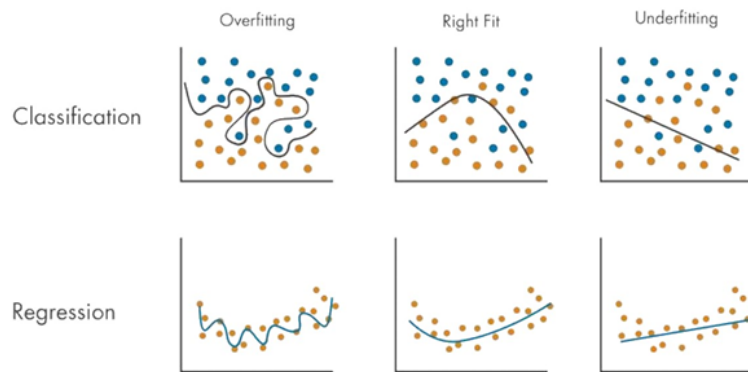


Figure 3.2: Graphical representation of underfitting, overfitting, and an optimal generalization, for both classification and regression.

Conversely, a model too simple to capture the underlying relationship, which generates results not sufficiently accurate or completely wrong, performing poorly on both training and validation data, is said to *underfit* (right column images of Figure 3.2).

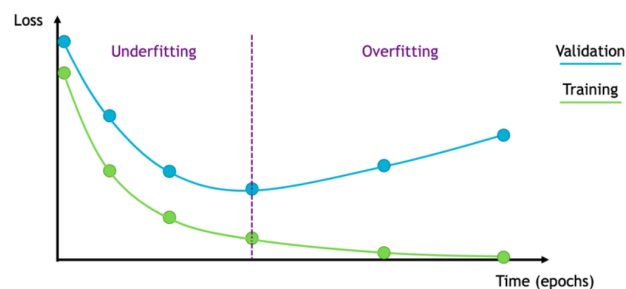


Figure 3.3: Illustration of the early stopping mechanism. The vertical dashed line indicates the optimal training epoch corresponding to the minimum validation loss [22].

As illustrated in Figure 3.3, overfitting typically manifests as a growing divergence between training and validation loss as training progresses. Common

strategies to mitigate overfitting include:

- **Regularization:** techniques that constrain the effective complexity or capacity of the model during training to prevent it from fitting noise in the training data. Standard approaches include:
 - *Weight decay (L_2 regularization):* adds a penalty proportional to the magnitude of the model parameters to the loss function, discouraging the network from relying on excessively large weights.
 - *Dropout:* at each training iteration, a random fraction of neurons is temporarily deactivated, in order to prevent single neurons from co-adapting and forces the network to learn more robust, redundant representations.
- **Data augmentation:** artificially expands the training dataset by applying domain-specific random transformations (such as rotations, cropping, flipping, or additive noise). This exposes the model to a broader variety of patterns, enhancing its invariance to task-irrelevant variations, and increasing specifically dataset dimensions.
- **Early stopping:** training is halted once performance on the validation set stops to improve, rather than continuing until the training loss is fully minimized. The model parameters corresponding to the best validation score are retained, implicitly limiting how far the network can drift toward fitting training-set-specific noise.
- **Dataset size and diversity:** a larger and more varied training set reduces the relative impact of noise and individual anomalies, making it harder for the model to memorize specific examples rather than learning the underlying target function.

3.2 Artificial Neural Networks

The elementary building block of a NN is the *artificial neuron* (Figure 3.4), inspired by the biological neuron and first formalized in the pioneering *Per-*

ceptron: given an input vector $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$, a neuron computes a weighted sum of its inputs, adds a bias term b , and passes the result through a (generally nonlinear) activation function Φ :

$$z = \sum_{i=1}^n w_i x_i + b = \mathbf{w}^T \mathbf{x} + b, \quad a = \Phi(z), \quad (41)$$

where $\mathbf{w} \in \mathbb{R}^n$ is the vector of weights, $b \in \mathbb{R}$ is the bias, z is referred to as the neuron's pre-activation, and a is its output (activation). The weights and bias constitute the trainable parameters of the neuron; the activation function Φ (Section 3.2) is instead fixed a priori as part of the architecture design.

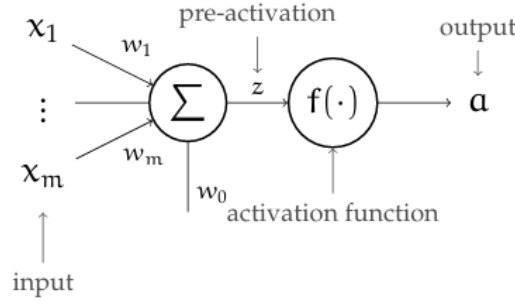


Figure 3.4: Schematic representation of an artificial neuron [21]. Symbol f here indicates what in this thesis is defined as Φ .

By organizing artificial neurons into successive *layers* (as shown in Figure 3.5), a *feed-forward neural network*, or multilayer perceptron, is obtained. In this network, the output of each layer serves as the input to the next one, with information flowing strictly in one direction (from input to output), without cycles or feedback connections. A network with L layers computes, for $l = 1, \dots, L$:

$$\mathbf{z}^{[l]} = \mathbf{W}^{[l]} \mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}, \quad \mathbf{a}^{[l]} = \Phi^{[l]}(\mathbf{z}^{[l]}), \quad (42)$$

with $\mathbf{a}^{[0]} = \mathbf{x}$ the network input, $\mathbf{W}^{[l]} \in \mathbb{R}^{n_l \times n_{l-1}}$ and $\mathbf{b}^{[l]} \in \mathbb{R}^{n_l}$ the weight matrix and bias vector of layer l (with n_l neurons), and $\mathbf{a}^{[L]}$ the final network output. Layers between the input and the output are referred to as *hidden layers*; a network is called *deep* when it is formed by multiple hidden layers. The *universal approximation theorem* [23] establishes that a feed-forward

network with a single hidden layer of sufficient width, and a suitable nonlinear activation function, can approximate any continuous function to arbitrary accuracy.

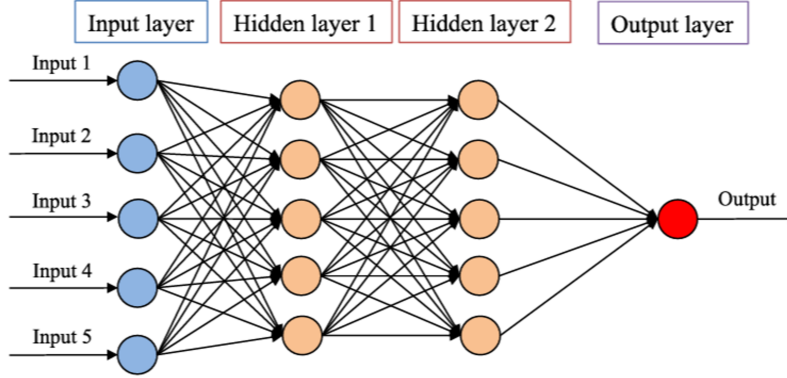


Figure 3.5: Schematic representation of a feed-forward (fully connected) neural network with two hidden layers.

Activation functions

The activation function Φ introduces nonlinearity into the network: without it, the composition of successive linear layers would collapse into a single linear transformation, regardless of network depth, severely limiting the class of functions representable. Nonlinearity is what allows the network to bend and fold its decision surface to fit data that otherwise no single hyperplane could separate or approximate. The most commonly used activation functions are described below:

- **Sigmoid:** mapping $\mathbb{R} \rightarrow (0, 1)$; widely used, but prone to the *vanishing gradient* problem [24] for large $|z|$, where $\Phi'(z) \rightarrow 0$.

$$\Phi(z) = \frac{1}{1 + e^{-z}}, \quad \Phi'(z) = \Phi(z)(1 - \Phi(z)). \quad (43)$$

- **Hyperbolic tangent:** mapping $\mathbb{R} \rightarrow (-1, 1)$; zero-centered, but still susceptible to vanishing gradients at saturation.

$$\Phi(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}, \quad \Phi'(z) = 1 - \Phi(z)^2. \quad (44)$$

- **Rectified Linear Unit (ReLU)**: computationally inexpensive and free of vanishing-gradient issues for $z > 0$, and the most widely adopted activation in modern deep networks; its main drawback is the so-called *dying ReLU* problem, whereby neurons with persistently negative pre-activations stop contributing gradient information entirely [25].

$$\Phi(z) = \max(0, z), \quad \Phi'(z) = \begin{cases} 0, & z \leq 0, \\ 1, & z > 0. \end{cases} \quad (45)$$

- **Leaky ReLU**: the introduction of a small $\alpha > 0$ mitigates the dying ReLU problem by allowing a small, non-zero gradient for $z < 0$.

$$\Phi(z) = \begin{cases} \alpha z, & z \leq 0, \\ z, & z > 0. \end{cases} \quad \Phi'(z) = \begin{cases} \alpha, & z \leq 0, \\ 1, & z > 0. \end{cases} \quad (46)$$

- **Softmax**: given a classification task, each neuron in the output layer of this NN is directly associated with one of the specific classes C . Softmax is used to convert the vector of raw pre-activation scores (logits) $\mathbf{z} = [z_1, \dots, z_C]^\top \in \mathbb{R}^C$ into a valid discrete probability distribution Φ over these C mutually exclusive classes.

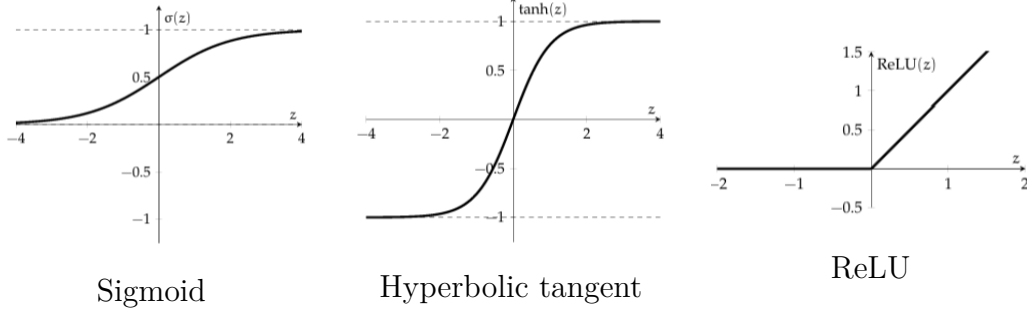
$$\Phi_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_{k=1}^C e^{z_k}}, \quad i = 1, \dots, C, \quad (47)$$

where $\Phi_i(\mathbf{z})$ represents the predicted probability that the input belongs to class i and ensures that, both $\sum_{k=1}^C \Phi_k(\mathbf{z}) = 1$ and $\Phi_i(\mathbf{z}) > 0$ are satisfied. Its derivative, being a vector-valued function of a vector input, takes the form of a Jacobian matrix,

$$\frac{\partial \Phi_i(\mathbf{z})}{\partial z_j} = \Phi_i(\mathbf{z}) (\delta_{ij} - \Phi_j(\mathbf{z})), \quad i, j = 1, \dots, C, \quad (48)$$

with δ_{ij} the Kronecker delta [26] ($\delta_{ij} = 1$ if $i = j$, 0 otherwise).

Generally, ReLU activations are standard for internal hidden layers. For out-



put layers, sigmoid functions are typically used in binary classification tasks, whereas softmax is preferred for multi-class classification [21].

Loss functions

Given an input vector $\mathbf{x}$ associated with a known ground-truth target y (here considered as a scalar, but it can also be a vector in the case of multi-output), the loss function $\mathcal{L}(\hat{y}, y)$ measures the discrepancy between the network's prediction $\hat{y}$ and the true label y for a single sample. During training, the objective is to minimize the empirical risk (or average loss) evaluated across a dataset or mini-batch of N independent samples, $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^N$:

$$J(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\hat{y}^{(i)}, y^{(i)}). \quad (49)$$

The choice of the loss function depends directly on the task formulation:

- **Regression Tasks:** The network outputs a continuous scalar or vector $\hat{y}_i \in \mathbb{R}$ aimed at approximating the real target $y_i \in \mathbb{R}$ for sample i .

– *Mean Squared Error (MSE):*

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2, \quad (50)$$

which penalizes larger errors quadratically, assuming normally distributed target noise [27].

- *Mean Absolute Error (MAE)*:

$$\mathcal{L}_{\text{MAE}} = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i|, \quad (51)$$

which provides a linear penalty and is less sensitive to extreme outliers.

- **Classification Task:** The network outputs a probability distribution vector $\hat{\mathbf{p}} = [\hat{p}_1, \dots, \hat{p}_C]^\top$ via Softmax, where $\hat{p}_k = P(y_k = 1 \mid \mathbf{x})$.

- The *Categorical Cross-Entropy* [26] loss evaluates the quality of these predicted probabilities against the one-hot target vector $\mathbf{y}$. For a single sample, it is defined as:

$$\mathcal{L}_{\text{CE}}(\hat{\mathbf{p}}, \mathbf{y}) = - \sum_{k=1}^C y_k \log \hat{p}_k. \quad (52)$$

In practice, because y_k acts as an indicator variable (filtering out all incorrect classes where $y_k = 0$), the summation in Eq. (52) collapses into a single non-zero term:

$$\mathcal{L}_{\text{CE}}(\hat{\mathbf{p}}, \mathbf{y}) = - \log \hat{p}_{c^*}. \quad (53)$$

Thus, the loss directly penalizes the negative logarithm of the probability assigned by the network to the true class c^* . If the network predicts $\hat{p}_{c^*} \rightarrow 1$, the loss approaches 0; conversely, as $\hat{p}_{c^*} \rightarrow 0$, the loss grows towards $+\infty$. When training over a mini-batch of N independent samples, the overall loss is calculated as the average single-sample loss across the batch:

$$\mathcal{L}_{\text{CE_batch}} = - \frac{1}{N} \sum_{i=1}^N \sum_{k=1}^C y_{i,k} \log \hat{p}_{i,k}, \quad (54)$$

where $y_{i,k}$ and $\hat{p}_{i,k}$ denote the target indicator (0 or 1) and the predicted probability for class k of the i -th sample, respectively.

3.3 Training of Neural Networks

This section describes how the trainable parameters $\theta = \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]}\}_{l=1}^L$ of a neural network are determined from data, by minimizing the chosen loss function $\mathcal{L}(\theta)$ over the training set.

3.3.1 Backpropagation

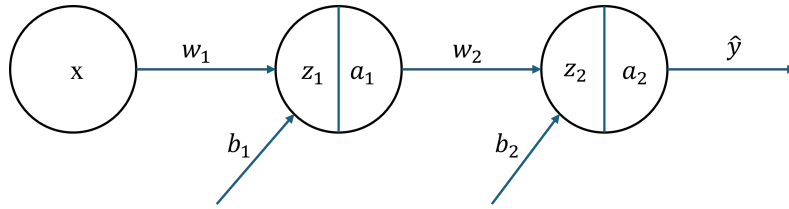


Figure 3.6: Illustrative example of a NN used as reference to explain the backpropagation method.

Backpropagation [28, 21] is the algorithm used to efficiently compute the gradient of the loss function with respect to every trainable parameter of the network, $\nabla_{\theta}\mathcal{L}$, by repeated application of the chain rule of differentiation, propagating the error signal backward through the network, from the output layer to the input layer. Let us consider the simplified NN with a single hidden layer represented in Figure 3.6: given the scalar input x , the NN generates a prediction $\hat{y}$ based on the current set of parameters θ , which generates an error $e = y - \hat{y}$ with respect to the correct value y corresponding to x . In order to minimize this error, the parameters must be updated following the *gradient descent* direction, i.e. each parameter is updated using the negative of the derivative of the loss function with respect to that parameter, propagated back to its level through the network. For the example under consideration, the following quantities must be computed:

$$\frac{\partial \mathcal{L}}{\partial w_2}, \quad \frac{\partial \mathcal{L}}{\partial b_2}, \quad \frac{\partial \mathcal{L}}{\partial w_1}, \quad \frac{\partial \mathcal{L}}{\partial b_1}. \quad (55)$$

The first derivative is computed here in a very intuitive way as a reference, to better understand the concept behind backpropagation algorithm. By follow-

ing the chain rule:

$$\frac{\partial \mathcal{L}}{\partial w_2} = \frac{\partial \mathcal{L}}{\partial e} \frac{\partial e}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_2} \frac{\partial z_2}{\partial w_2}. \quad (56)$$

Each partial derivative on the right-hand side is straightforward to compute once the activation and loss functions have been chosen (in this case, the sigmoid σ and the squared error e^2 , respectively):

$$L = e^2 \Rightarrow \frac{\partial L}{\partial e} = 2e$$

$$e = y - \hat{y} \Rightarrow \frac{\partial e}{\partial \hat{y}} = -1$$

$$\hat{y} = \sigma(z_2) \Rightarrow \frac{\partial \hat{y}}{\partial z_2} = \sigma(z_2)(1 - \sigma(z_2)) = \hat{y}(1 - \hat{y}) \text{ from Eq.(43)}$$

$$z_2 = w_2 a_1 + b_2 \Rightarrow \frac{\partial z_2}{\partial w_2} = a_1$$

Therefore:

$$\frac{\partial L}{\partial w_2} = 2e(-1)(\hat{y}(1 - \hat{y}))(a_1)$$

where every term on the right-hand side has already been computed during the forward pass, so that it is sufficient to keep these quantities in memory and simply perform a multiplication.

$$w_2 \leftarrow w_2 - \eta \frac{\partial L}{\partial w_2}. \quad (57)$$

Eq. (57) shows the so-called *delta rule* [28] for the parameter update, where η is a small multiplier, called the *learning rate*, which prevents excessively large jumps across the loss surface in parameter space $\boldsymbol{\theta}$. In this basic version, η is fixed; most modern algorithms, however, employ an adaptive learning rate that accounts for the history of previous updates (Section 3.3.3).

The example above illustrates, for a single parameter of a minimal two-neuron network, the mechanism underlying backpropagation. In a general feed-forward network of L layers (as the one introduced in Section 3.2), the same chain-rule argument is applied systematically, layer by layer, by introducing the error

signal $\boldsymbol{\delta}^{[l]} := \partial \mathcal{L} / \partial \mathbf{z}^{[l]}$. At the output layer,

$$\boldsymbol{\delta}^{[L]} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^{[L]}} \odot f'^{[L]}(\mathbf{z}^{[L]}), \quad (58)$$

with $\odot$ the Hadamard (element-wise) product [29]; the error signal is then propagated backward through the recursion

$$\boldsymbol{\delta}^{[l]} = \left((\mathbf{W}^{[l+1]})^\top \boldsymbol{\delta}^{[l+1]} \right) \odot f'^{[l]}(\mathbf{z}^{[l]}), \quad l = L - 1, \dots, 1, \quad (59)$$

from which the gradients with respect to every weight matrix and bias vector in the network follow directly:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \boldsymbol{\delta}^{[l]} (\mathbf{a}^{[l-1]})^\top, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \boldsymbol{\delta}^{[l]}. \quad (60)$$

This recursive formulation is precisely what the scalar example above illustrated concretely for a single weight: it allows the gradient with respect to *every* parameter in the network to be computed in a single backward pass, at a computational cost comparable to that of the forward pass itself, making the training of deep networks computationally tractable. In practice, this procedure is carried out automatically by modern deep learning frameworks through *automatic differentiation*, without requiring the gradient expressions to be derived by hand for each specific architecture.

3.3.2 Gradient-based optimization

Once the gradient $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ has been computed via backpropagation, the network parameters are updated iteratively so as to decrease the loss, following the direction of steepest descent. The simplest such scheme is *gradient descent* [30],

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}), \quad (61)$$

where $\eta > 0$ is the *learning rate*, controlling the step size of each update. Computing $\nabla_{\boldsymbol{\theta}} \mathcal{L}$ exactly requires evaluating the loss over the entire training set at every iteration, which is computationally prohibitive for large datasets.

Stochastic gradient descent (SGD) instead approximates the gradient using a single randomly sampled training example (or, more commonly in practice, a small random *minibatch* of m examples),

$$\nabla_{\theta} \mathcal{L} \approx \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \mathcal{L}_i(\theta).$$

One full pass over the entire training set is referred to as an *epoch*; training typically proceeds over many epochs, with the data reshuffled at the start of each.

3.3.3 Adam optimizer

Whereas standard gradient descent-based optimization algorithms update all parameters using the same, fixed learning rate, adaptive optimization methods adjust the effective step size individually for each parameter, based on the history of its past gradients. *Adam* (Adaptive Moment Estimation) [31, 30], the optimizer employed throughout the neural network training performed in this thesis, combines two adaptive mechanisms, both essential for different reason: *momentum* ($\mathbf{m}_t$) and *RMSProp-style scaling* ($\mathbf{v}_t$). This combination offers a dual advantage over traditional SGD. While *Momentum* stabilizes the search trajectory by dampening high-frequency oscillations across steep, ravine-like loss surfaces, *RMSProp-style scaling* dynamically rescales the learning rate per parameter. This per-parameter adaptation ensures optimal learning rates across all neurons, preventing parameters with large gradients from destabilizing training while allowing those with small gradients to make sufficient progress.

At each iteration t , given the current gradient $\mathbf{g}_t = \nabla_{\theta} \mathcal{L}(\theta_{t-1})$, Adam keeps track of two running averages:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1) \mathbf{g}_t, \quad \mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2) \mathbf{g}_t^2, \quad (62)$$

namely a running average of the gradient itself, $\mathbf{m}_t$, and a running average of its square, $\mathbf{v}_t$, with typical decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Since both

are initialized at zero, they start out biased toward zero, especially in the first few iterations; this is corrected as

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1}, \quad \hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2}. \quad (63)$$

The parameters are then updated as

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\mathbf{m}}_t}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon}, \quad (64)$$

where ϵ is a small constant (e.g. 10^{-8}) added only to avoid dividing by zero. In practice, dividing by $\sqrt{\hat{\mathbf{v}}_t}$ is what makes the step size adaptive: parameters with a history of large gradients get a smaller effective step, while parameters with small or noisy gradients get a comparatively larger one. Adam is widely adopted in practice owing to its robustness to the choice of learning rate and its fast convergence across a broad range of architectures and tasks.

3.3.4 Training hyperparameters

Beyond the trainable parameters $\boldsymbol{\theta}$ learned via backpropagation and gradient-based optimization, the training procedure is governed by a number of *hyperparameters*, fixed prior to training (coming from a combination of experience, literature guidelines, and validation-set performance) rather than learned from data. The principal hyperparameters relevant to this work are:

- **Learning rate η :** controls the magnitude of each parameter update by Eq. (64); too large a value can cause training to diverge or oscillate, while too small a value results in slow convergence. A *learning rate schedule*, which reduces η over the course of training (e.g. by a fixed factor every fixed number of epochs), is often employed to combine fast initial progress with fine-grained convergence in later epochs.
- **Number of epochs:** the number of full passes over the full training set; typically determined in conjunction with early stopping (Section 3.1.2), monitoring validation loss rather than fixing this value a priori.

- **Network architecture hyperparameters:** the number of layers, the number of neurons (or filters, for convolutional layers 3.4) per layer, and the choice of activation function, all of which determine the capacity of the model and are tuned to balance underfitting and overfitting for the specific task at hand.

The specific hyperparameter values adopted in this thesis are reported and justified in Chapter 6.

3.4 Convolutional Neural Networks

While the fully connected feed-forward architecture introduced in Section 3.2 is, in principle, a universal function approximator, it is poorly suited to inputs with an inherent spatial or sequential structure, such as images or one-dimensional signals with specific features. Applied directly to such inputs, a fully connected layer would require a separate weight for every input–output neuron pair, resulting in a very large number of trainable parameters and a model prone to overfitting. More fundamentally, it has no built-in notion that a given local pattern carries the same meaning regardless of *where* it occurs in the input. Convolutional neural networks (CNNs) address this limitation by replacing dense connections with *convolutional layers*, in which a small set of learnable filters is applied across the input to detect local patterns. Because the same filter is applied at every position, a pattern learned at one location is automatically recognized when it appears, also if it is shifted in the input. This CNN property is known as *translation equivariance* [32]. Stacking multiple convolutional layers allows the network to build increasingly abstract representations, with early layers typically capturing simple, low-level features and deeper layers combining them into more complex, higher-level patterns. CNNs constitute the architectural basis of the classification and regression models developed in this work.

3.4.1 One-dimensional convolution

Given a one-dimensional input signal $x \in \mathbb{R}^n$, where n denotes the number of components present in the signal (in this work, $n = N_{\text{points}}$, the number of

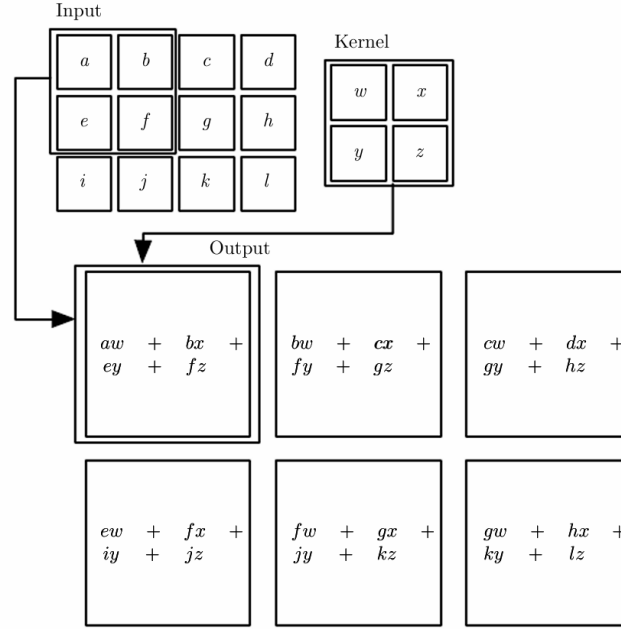


Figure 3.7: Example of the convolution (cross-correlation) operation [33].

points at which the stress–strain curve is sampled), and a kernel w of size K , the convolution operation produces an output signal

$$y[n] = \sum_{k=0}^{K-1} w[k] x[n + k - \lfloor K/2 \rfloor] + b, \quad (65)$$

where b is a scalar bias term, shared across all positions n . It is worth noting that, despite the name, the operation implemented in practically all deep learning frameworks is technically a *cross-correlation* (graphically illustrated in Figure 3.7 for a 2D input matrix) rather than a true convolution [33], since the kernel is not flipped before being applied; the two operations coincide up to a reflection of the kernel indices, and since the kernel weights are learned rather than fixed, this distinction has no practical consequence and the term “convolution” is retained by convention.

Compared to what was shown for the fully connected NN, two additional hyperparameters govern the behavior of the convolution: the *stride* S , which controls the step size with which the kernel slides across the input, and the *padding*, which determines how the input is extended at its boundaries.

Three padding regimes are commonly distinguished:

- *valid* padding, in which no padding is added and the kernel is only allowed to visit positions fully contained within the input, causing the output length to shrink by $K - 1$ at every layer;
- *full* padding, in which enough zeros are added at extreme of the input signal so that every input sample is visited K times, producing an output longer than the input;
- *same* padding, in which the input is padded with zeros so that the output signal has the same length as the input, for unit stride.

Same padding is generally preferred in deep architectures because it allows an arbitrary number of convolutional layers to be stacked without the representation collapsing in length, at the cost of a minor underrepresentation of the boundary samples relative to the central ones [34]. Figure 3.8 and 3.9 illustrates the same padding and full padding concepts respectively.

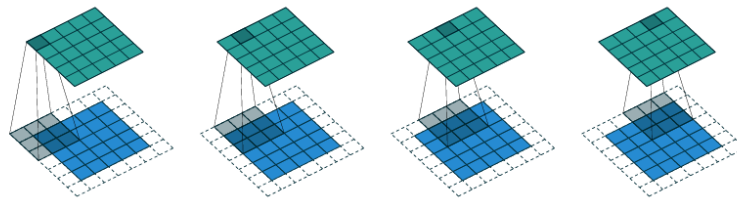


Figure 3.8: Same padding, unit stride: the input (bottom, dashed cells denote locations where zeros are added) is padded just enough so that the output (top) has the same length as the input [34].

While the padding regime controls the extent of the input, the stride controls how densely the kernel samples it. Figures 3.10 and 3.11 illustrate this effect for the case of no padding (valid convolution): with unit stride the kernel visits every possible position, whereas with a stride greater than one reduces the length of the output signal, effectively performing *downsampling*.

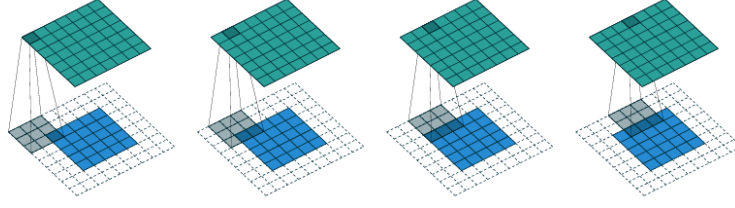


Figure 3.9: Full padding, unit stride: enough zero padding is added (dashed cells) for every input sample to be visited K times by the kernel, producing an output longer than the input [34].

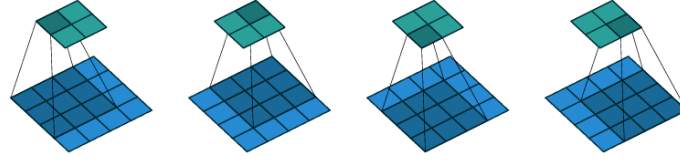


Figure 3.10: No padding (valid convolution), unit stride: the kernel visits every position fully contained within the input, and the output is shorter than the input by $K - 1$ [34].

3.4.2 Convolutional layers

In Section 3.4.1, the input to a convolutional layer has been treated as a single one-dimensional signal. In the general case, however, the input can consist of C_{in} distinct channels, stacked together as $x \in \mathbb{R}^{C_{\text{in}} \times n}$. Contextually, a convolutional layer typically applies not just a single kernel, but F distinct kernels in parallel to this multi-channel input, each producing a separate output channel, commonly referred to as a *feature map*. During training, each feature map becomes specialized in detecting a particular class of local patterns in the input. Crucially, each of the F kernels spans *all* C_{in} input channels at once, rather than acting on a single channel independently: for a given output position n , the f -th kernel combines information from all C_{in} channels within its receptive field into a single scalar output, so that the f -th output feature map is computed as

$$y_f[n] = \Phi \left(\sum_{c=1}^{C_{\text{in}}} \sum_{k=0}^{K-1} w_{f,c}[k] x_c[n + k - \lfloor K/2 \rfloor] + b_f \right), \quad f = 1, \dots, F, \quad (66)$$

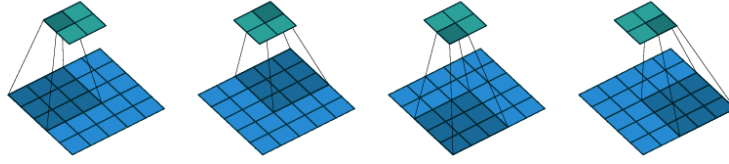


Figure 3.11: No padding, stride greater than one: the kernel skips positions as it slides across the input, further reducing the length of the output beyond what valid padding alone would cause [34].

with Φ a nonlinear activation function (Section 3.2), applied element-wise, exactly as in a fully connected layer. Note that each of the F kernels has its own set of weights $w_{f,c}[k]$ for $c = 1, \dots, C_{\text{in}}$ and its own bias b_f , so that the total number of trainable parameters in the layer is $F \times C_{\text{in}} \times K + F$, exactly as it is a bi-dimensional filter. Additional specific features of convolutional layers, which make them more suitable than fully connected layers, are [33]:

- **Parameter sharing.** The same kernel weights $w_{f,c}[k]$ are reused at every position n of the input, rather than a separate set of weights being learned for every input–output pair. This is also referred to as *tied weights*, and it is what allows a pattern learned at one location to be recognized wherever it occurs elsewhere in the signal.
- **Equivariance to translation.** As a direct consequence of parameter sharing, convolution is *equivariant* to translations of the input: if a feature in the input signal is shifted by n_0 samples, its representation in the output feature map is shifted by the same amount, rather than changing shape or disappearing.

Successive convolutional layers can be stacked, with the feature maps produced by one layer serving as the input to the next. Because each output value depends only on a local window of K samples of its input, a single convolutional layer has a limited *receptive field*, equal to K samples of the original input. However, when a second layer with the same kernel size is stacked on top of the first, each of its output values depends on K consecutive outputs of the first layer, each of which in turn depends on K samples of the original input: the receptive field of the second layer is therefore not K , but $2(K - 1) + 1$ samples

of the original signal. After L stacked layers with unit stride and kernel size K , the receptive field grows linearly with depth as

$$R_L = 1 + L(K - 1). \quad (67)$$

Eq. (67) assumes that all L layers share the same kernel size K . More generally, if the l -th layer uses a kernel of size K_l (with $l = 1, \dots, L$), the receptive field after L stacked layers with unit stride becomes

$$R_L = 1 + \sum_{l=1}^L (K_l - 1). \quad (68)$$

Based on Eq. (68), it is clear that stacked layers allows deeper layers to capture increasingly global patterns in the signal.

3.4.3 Pooling

Pooling layers perform a fixed, local downsampling operation on their input, most commonly *max pooling* (retaining, within each local window, only the maximum activation) or *average pooling* (retaining the average activation within each window):

$$y_{\max}[n] = \max_{k=0, \dots, P-1} x[nP + k], \quad y_{\text{avg}}[n] = \frac{1}{P} \sum_{k=0}^{P-1} x[nP + k], \quad (69)$$

for a pooling window of size P (typically applied with stride equal to P , so that pooling windows do not overlap and the sequence length is reduced by a factor of P).

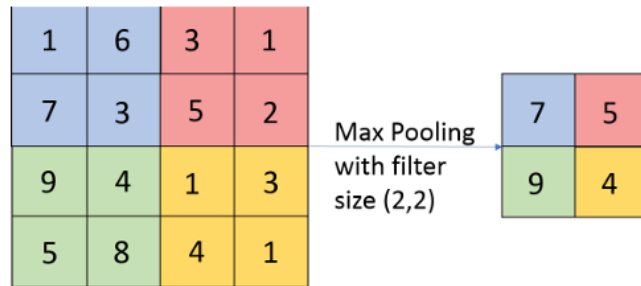


Figure 3.12: Example of max pooling with a (2, 2) filter.

Pooling serves two main purposes. First, it reduces the dimensionality of the feature maps as depth increases, reducing both the computational cost and the number of parameters in any subsequent fully connected layers. Second, it introduces a degree of local *invariance* to translation: because pooling replaces the exact value of a feature with a summary statistic over a small neighborhood, small shifts in the position of a detected pattern leave most of the pooled output unchanged [33].

3.4.4 CNN structure

A typical convolutional layer can be decomposed into three conceptual stages, applied in sequence: a *convolution stage*, in which the layer computes several convolutions in parallel following Eq. (66), producing a set of linear activations; a *detector stage*, in which each linear activation is passed through the nonlinear activation function Φ ; and, optionally, a *pooling stage* (Section 3.4.3), in which a pooling function further downsamples the output of the detector stage. A full CNN architecture is then obtained by stacking several such layers, progressively transforming the raw input signal into increasingly abstract, feature representations, with the number of feature maps typically increasing and the sequence length decreasing, due to pooling and/or strided convolution. The resulting feature representation is finally fed into one or more fully connected layers (Section 3.2) that combine the extracted features and produce the final network output.

Chapter 4

Numerical solver

Given the objective of characterizing the quasi-static tensile response of specimens exhibiting quasi-brittle behavior, the discretized PD equation of motion (32) can be reformulated as a system of algebraic equations in the form

$$\mathbf{R}(\mathbf{u}) = \mathbf{F}(\mathbf{u}) + \boldsymbol{\lambda} \mathbf{q} = \mathbf{0}, \quad (70)$$

where $\mathbf{u} \in \mathbb{R}^{2N}$ (or $\mathbb{R}^{3N}$ in 3D) collects the nodal displacement components of the N discretization nodes, $\mathbf{F}(\mathbf{u})$ is the vector of internal (peridynamic) nodal forces, $\mathbf{q}$ is a constant vector of reference nodal actions (forces or displacements, depending on the type of control adopted), $\boldsymbol{\lambda}$ is a prescribed incremental scalar load (or displacement) factor of $\mathbf{q}$ and $\mathbf{R}(\mathbf{u})$ is the residual vector, i.e. the imbalance between the internal forces $\mathbf{F}(\mathbf{u})$ and the external actions $\boldsymbol{\lambda} \mathbf{q}$.

The numerical problem thus consists, at each incremental step n , in finding the displacement vector $\mathbf{u}$ that satisfies the equilibrium condition $\mathbf{R}(\mathbf{u}) = \mathbf{0}$. Owing to the softening constitutive laws introduced in Section 2.4, the internal force vector $\mathbf{F}(\mathbf{u})$ is a nonlinear, history-dependent function of the displacement field: each bond degrades according to a softening law once s_0 is overcome, and the branch governing its current stiffness is dictated by the maximum stretch $s_{\max}$ achieved during the loading history. Consequently, the residual vector $\mathbf{R}$ in Eq. (70) cannot be obtained in closed form, and an incremental-iterative numerical solution procedure is therefore required.

4.1 Newton-Raphson iterative schemes

The classical strategy for solving Eq. (70) consists in imposing the prescribed boundary condition incrementally and, at each increment $n \rightarrow n+1$, correcting

an initial estimate of the displacement field through a sequence of linearized problems, until the out-of-balance force vector vanishes within a prescribed tolerance. Expanding $\mathbf{F}$ in a first-order Taylor series about the current configuration ${}^k\mathbf{u}^{n+1}$ yields

$${}^{k+1}\mathbf{F}^{n+1} = {}^k\mathbf{F}^{n+1} + \left. \frac{\partial \mathbf{F}}{\partial \mathbf{u}} \right|_{{}^k\mathbf{u}^{n+1}} \delta \mathbf{u} = {}^k\mathbf{F}^{n+1} + {}^k\mathbf{K}^{n+1} \delta \mathbf{u}, \quad (71)$$

where ${}^k\mathbf{K}^{n+1} = \partial \mathbf{F} / \partial \mathbf{u}|_{{}^k\mathbf{u}^{n+1}}$. Substituting Eq. (71) by ${}^{k+1}\mathbf{F}^{n+1}$ into Eq. (70) evaluated at time step $n+1$ and iteration $k+1$ gives

$${}^{k+1}\mathbf{R}^{n+1} = {}^k\mathbf{F}^{n+1} + {}^k\mathbf{K}^{n+1} \delta \mathbf{u} + {}^{k+1}\boldsymbol{\lambda}^{n+1} \mathbf{q}. \quad (72)$$

In the classical Newton-Raphson scheme the load (or displacement) factor is prescribed once and for all at the beginning of the increment, and is therefore *not* updated by the corrector iteration:

$${}^{k+1}\boldsymbol{\lambda}^{n+1} = {}^k\boldsymbol{\lambda}^{n+1} = \boldsymbol{\lambda}^{n+1}. \quad (73)$$

Substituting ${}^k\mathbf{F}^{n+1} = -\boldsymbol{\lambda}^{n+1} \mathbf{q} + {}^k\mathbf{R}^{n+1}$ into Eq. (72) and applying the equivalence in Eq. (73), yields

$${}^{k+1}\mathbf{R}^{n+1} = {}^k\mathbf{R}^{n+1} + {}^k\mathbf{K}^{n+1} \delta \mathbf{u}. \quad (74)$$

Imposing equilibrium at the corrected iterate, ${}^{k+1}\mathbf{R}^{n+1} = \mathbf{0}$, Eq. (74) finally gives the linearized system solved for the displacement correction $\delta \mathbf{u}$:

$${}^k\mathbf{K}^{n+1} \delta \mathbf{u} + {}^k\mathbf{R}^{n+1} = \mathbf{0}, \quad (75)$$

where ${}^k\mathbf{K}^{n+1}$ denotes the tangent stiffness matrix evaluated at the k -th iteration, and $\delta \mathbf{u}$ is the resulting displacement correction. Two classical variants of this scheme are commonly employed:

- **Force-controlled Newton–Raphson:** the load factor λ^{n+1} is prescribed a priori and increased monotonically at every step, while $\delta \mathbf{u}$ is obtained by solving Eq. (75) directly.

- **Displacement-controlled Newton–Raphson:** the nodal displacement vector is partitioned into a prescribed (Dirichlet) $\mathbf{D}$ part, incremented monotonically to impose the boundary condition, and a free (Neumann) $\mathbf{N}$ part, obtained at each iteration by solving the linearized equilibrium equation restricted to the free degrees of freedom, ${}^k\mathbf{K}_{ff}^{n+1} \delta\mathbf{N} + {}^k\mathbf{R}_N^{n+1} = \mathbf{0}$, with $\mathbf{K}_{ff}$ corresponding to the reduced part of the tangent stiffness matrix which relates the free-degree-of-freedom forces to the free-degree-of-freedom displacements. In this case, the reaction forces on the prescribed degrees of freedom are then recovered a posteriori, since they are originally unknown.

Both variants share the same fundamental weakness: they treat λ^{n+1} (or, equivalently, the imposed displacement) as a monotonically increasing, externally prescribed quantity, rather than as an additional unknown of the problem.

4.2 Snap-back/snap-through instabilities

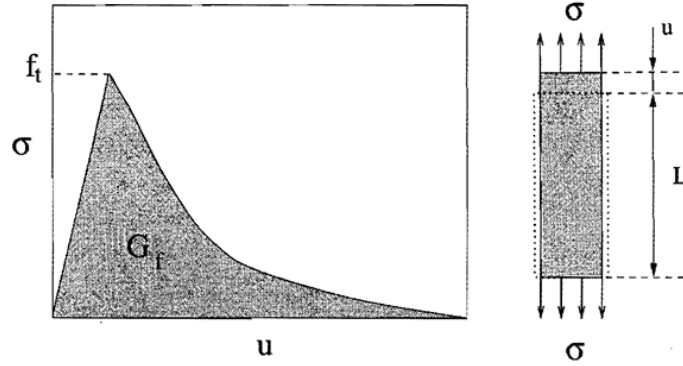


Figure 4.1: Schematic representation of a uniaxial tensile test on a quasi-brittle specimen with its corresponding stress–displacement response [35].

In quasi-brittle materials undergoing softening (Figure 4.1), the equilibrium path in the load–displacement plane may exhibit *limit points*, beyond which further application of a monotonically increasing load (or displacement) is no longer compatible with a stable equilibrium configuration. In particular, as is clearly illustrated in Figure 4.2:

- a *snap-through* instability occurs when, for a given load level, multiple equilibrium displacement values coexist, and the tangent stiffness matrix becomes singular;
- a *snap-back* instability occurs when the equilibrium path folds back on itself in the load-displacement plane, so that even the imposed displacement is no longer a monotonically increasing function.

Near such limit points, the tangent stiffness matrix ${}^k\mathbf{K}^{n+1}$ becomes poorly conditioned or singular, so that Eq. (75) does not admit a physically meaningful solution. As a consequence, force-controlled schemes are unable to trace the equilibrium path beyond a snap-through point, while displacement-controlled schemes, although capable of overcoming snap-through instabilities, fail in the presence of snap-back behavior, since the prescribed displacement itself is not a monotonic path parameter in that case. Snap-through instabilities are particularly critical for the response of quasi-brittle materials, since they systematically occur at the peak load, where the tangent stiffness matrix vanishes once time the material transitions from the elastic branch to the softening regime is reached; snap-back instabilities, although less systematic, can also arise within the softening branch itself, depending on the specimen geometry and the boundary conditions.

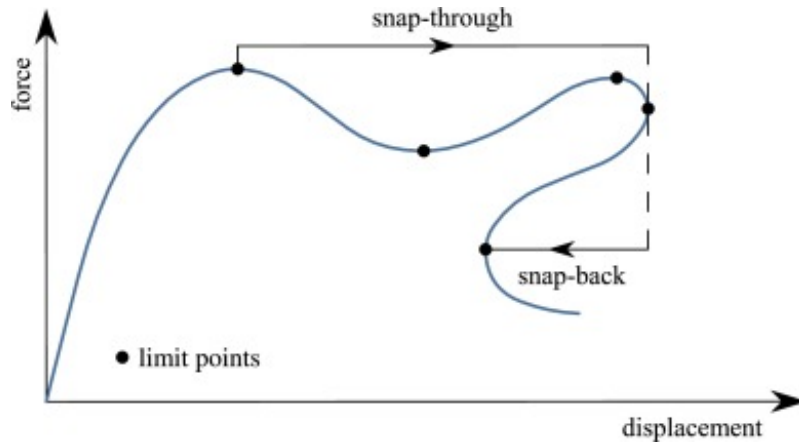


Figure 4.2: Non linear material behavior characterized both by snap-through and snap-back instabilities [14].

To overcome both limitations, *path-following* or *arc-length* methods have been proposed in the computational mechanics literature [36, 37, 38]. These methods treat the load (or displacement) factor λ as an additional unknown of the nonlinear problem, together with the nodal displacements, and introduce an auxiliary constraint equation (typically expressed in terms of the Euclidean norm of the incremental displacement and load-factor variation) so that the augmented system of equations remains well-posed even in the proximity of limit points. Among the various formulations available, the *cylindrical arc-length method*, originally proposed by Crisfield [37, 39], as an extension of the Riks method [36], is adopted in the present work.

4.3 Displacement controlled arc-length method

The classical (force-controlled) global arc-length method is formulated under the assumption of a fixed reference load pattern $\mathbf{q}$, scaled incrementally by the load factor λ , while the Dirichlet boundary conditions are homogeneous (i.e. the prescribed displacements are held at zero). Consequently, the method is, by construction, inapplicable to problems in which the loading is instead applied through a non-zero, incrementally increasing prescribed displacement, since no reference force vector $\mathbf{q}$ is available to be scaled by λ in that case. Because the numerical campaign carried out in this work reproduces displacement-controlled uniaxial tensile tests, a *displacement-controlled* variant of the cylindrical arc-length method, following [14], is employed.

Let the nodal displacement vector $\mathbf{u}^n$, at increment n , be partitioned into the vector of prescribed (Dirichlet) displacements $\mathbf{D}^n$ and the vector of free (Neumann) displacements $\mathbf{N}^n$ as follow:

$$\mathbf{u}^n = \begin{pmatrix} \mathbf{D}^n \\ \mathbf{N}^n \end{pmatrix}, \quad \mathbf{D}^n = \lambda^n \mathbf{d}, \quad (76)$$

where $\mathbf{d}$ is a constant reference vector of nodal displacements and λ^n is the scalar displacement factor at step n . Since the problem is entirely displacement-controlled, the reference force vector $\mathbf{q}$ in Eq. (70) is identically zero, and the residual reduces to the internal force vector alone. Similarly to

Eq. (76), both $\mathbf{R}$ and $\mathbf{F}$ are partitioned following the same notation,

$$\mathbf{R}(\mathbf{u}) = \begin{pmatrix} \mathbf{R}_D(\mathbf{u}) \\ \mathbf{R}_N(\mathbf{u}) \end{pmatrix} = \begin{pmatrix} \mathbf{F}_D(\mathbf{u}) \\ \mathbf{F}_N(\mathbf{u}) \end{pmatrix}, \quad (77)$$

and the tangent stiffness matrix is decomposed as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fp} \\ \mathbf{K}_{pf} & \mathbf{K}_{pp} \end{bmatrix}, \quad (78)$$

where the subscripts f and p denote, respectively, the free ($\mathbf{N}$) and prescribed ($\mathbf{D}$) degrees of freedom, so that $\mathbf{R} = \mathbf{K}\mathbf{u}$ reads,

$$\begin{pmatrix} \mathbf{R}_D \\ \mathbf{R}_N \end{pmatrix} = \begin{bmatrix} \mathbf{K}_{pp} & \mathbf{K}_{pf} \\ \mathbf{K}_{fp} & \mathbf{K}_{ff} \end{bmatrix} \begin{pmatrix} \mathbf{D} \\ \mathbf{N} \end{pmatrix}. \quad (79)$$

In this formulation, the vector of prescribed displacements $\mathbf{D}$ is specified at each load increment via Eq. (76), whereas the vector $\mathbf{N}$ remains the only unknown. The first block row of the partitioned system in Eq. (79) contains two unknown quantities, $\mathbf{R}_D$ and $\mathbf{N}$, and therefore cannot be solved independently. It can, however, be employed a posteriori, once $\mathbf{N}$ has been computed, to evaluate $\mathbf{R}_D$, i.e., the reaction forces associated with the prescribed degrees of freedom.

In contrast, the second block row involves only the unknown vector $\mathbf{N}$ and thus represents the actual equilibrium condition to be enforced. Accordingly, equilibrium is imposed exclusively on the Neumann subset of degrees of freedom by requiring their residual to vanish:

$$\mathbf{R}_N^{n+1} = \mathbf{K}_{fp} \mathbf{D}^{n+1} + \mathbf{K}_{ff} \mathbf{N}^{n+1} = \mathbf{0}. \quad (80)$$

This equation is the one that is effectively solved for the free displacement vector $\mathbf{N}$, within the incremental-iterative solution procedure described below. Within each increment, the trial state $(\mathbf{D}^{n+1}, \mathbf{N}^{n+1})$ is updated iteratively until equilibrium, Eq. (80), is satisfied. Let's consider again the first-order Taylor expansion of $\mathbf{F}_N^{n+1}$ at iteration $k+1$ about the current configuration ${}^k\mathbf{u}^{n+1}$, as

in Eq. (71): since, in the displacement-controlled case, $\mathbf{R}_N = \mathbf{F}_N$, the left-hand side of Eq. (71) is the residual at iteration $k + 1$, which is required to vanish upon convergence, while the first term on the right-hand side is the residual at the current iteration, ${}^k\mathbf{F}_N^{n+1} = {}^k\mathbf{R}_N^{n+1}$.

$$\left. \frac{\partial \mathbf{F}_N}{\partial \mathbf{u}} \right|_{k_{\mathbf{u}^{n+1}}} \delta \mathbf{u} = -{}^k\mathbf{R}_N^{n+1}. \quad (81)$$

The derivative on the left-hand side is expanded using the partition of $\delta \mathbf{u}$ introduced in Eq. (76) and the tangent stiffness blocks defined in Eq. (78):

$$\left. \frac{\partial \mathbf{F}_N}{\partial \mathbf{u}} \right|_{k_{\mathbf{u}^{n+1}}} \delta \mathbf{u} = \left. \frac{\partial \mathbf{F}_N}{\partial \mathbf{N}} \right|_{k_{\mathbf{N}^{n+1}}} \delta \mathbf{N} + \left. \frac{\partial \mathbf{F}_N}{\partial \mathbf{D}} \right|_{k_{\mathbf{D}^{n+1}}} \delta \mathbf{D} = {}^k\mathbf{K}_{ff}^{n+1} \delta \mathbf{N} + {}^k\mathbf{K}_{fp}^{n+1} \delta \mathbf{D}. \quad (82)$$

Substituting Eq. (82), together with $\delta \mathbf{D} = \delta \lambda \mathbf{d}$, into Eq. (81) finally yields

$${}^k\mathbf{K}_{ff}^{n+1} \delta \mathbf{N} + {}^k\mathbf{K}_{fp}^{n+1} \delta \lambda \mathbf{d} + {}^k\mathbf{R}_N^{n+1} = \mathbf{0}. \quad (83)$$

Eq. (83) is a linear system in the two unknowns $\delta \mathbf{N}$ and $\delta \lambda$, with $\delta \lambda$ appearing only as a coefficient multiplying the known vector ${}^k\mathbf{K}_{fp}^{n+1} \mathbf{d}$. By the superposition principle for linear systems, the solution $\delta \mathbf{N}$ can therefore be split into the sum of two independent contributions: the solution obtained with $\delta \lambda = 0$, and the solution obtained by setting the residual term to zero and retaining only the term proportional to $\delta \lambda$. Accordingly, $\delta \mathbf{N}$ is decomposed as

$$\delta \mathbf{N} = \delta \bar{\mathbf{u}} + \delta \lambda \delta \mathbf{u}_t, \quad (84)$$

where $\delta \bar{\mathbf{u}}$ is the correction that would eliminate the current out-of-balance residual ${}^k\mathbf{R}_N^{n+1}$ at fixed displacement factor ($\delta \lambda = 0$), while $\delta \mathbf{u}_t$ is the additional correction of the free degrees of freedom associated with a unit increment of the displacement factor, i.e., it acts as a tangential correction along the direction defined by the prescribed reference vector. The tangential contribution $\delta \mathbf{u}_t$ is essential since it introduces the explicit dependence of $\delta \mathbf{N}$ on $\delta \lambda$ required to solve the arc-length constraint (discussed hereafter) for $\delta \lambda \neq 0$.

Substituting Eq. (84) into Eq. (83), and separately equating the terms inde-

pendent of $\delta\lambda$ and those proportional to it, yields the two associated linear systems, from which $\delta\bar{\mathbf{u}}$ and $\delta\mathbf{u}_t$ are obtained:

$$\delta\bar{\mathbf{u}} = -\left({}^k\mathbf{K}_{ff}^{n+1}\right)^{-1} {}^k\mathbf{R}_N^{n+1}, \quad \delta\mathbf{u}_t = -\left({}^k\mathbf{K}_{ff}^{n+1}\right)^{-1} {}^k\mathbf{K}_{fp}^{n+1} \mathbf{d}. \quad (85)$$

In the numerical implementation, $\delta\bar{\mathbf{u}}$ and $\delta\mathbf{u}_t$ are computed directly by a single linear solve with a two-column right-hand side,

$$\begin{bmatrix} \delta\bar{\mathbf{u}} & \delta\mathbf{u}_t \end{bmatrix} = \left({}^k\mathbf{K}_{ff}^{n+1}\right)^{-1} \begin{bmatrix} -{}^k\mathbf{R}_N^{n+1} & -{}^k\mathbf{K}_{fp}^{n+1} \mathbf{d} \end{bmatrix}, \quad (86)$$

factorizing ${}^k\mathbf{K}_{ff}^{n+1}$ only once per iteration. The scalar $\delta\lambda$ remaining in Eq. (84) is not yet determined at this stage: it is determined, at every iteration, by enforcing the arc-length constraint, discussed next.

4.3.1 Arc-length constraint and corrector algorithm

To close the system, an auxiliary constraint is imposed on the cumulative incremental displacement of the free degrees of freedom within the current step:

$$\left({}^{k+1}\Delta\mathbf{N}^{n+1}\right)^\top \left({}^{k+1}\Delta\mathbf{N}^{n+1}\right) = \Delta l^2, \quad (87)$$

where ${}^{k+1}\Delta\mathbf{N}^{n+1} = {}^k\Delta\mathbf{N}^{n+1} + \delta\mathbf{N}$, and Δl denotes the prescribed arc length of the current increment. Eq. (87) corresponds to the cylindrical arc-length formulation, while the original spherical formulation, by contrast, augments the constraint with an additional term proportional to $\Delta\lambda^2 (\mathbf{q}^\top \mathbf{q})$. In the displacement-controlled formulation adopted in this work, since the reference load vector $\mathbf{q}$ is identically zero, the additional term of the spherical constraint therefore vanishes, and the two formulations coincide exactly. The cylindrical formulation is nonetheless retained as the reference form, since it leads directly to a simpler quadratic equation in $\delta\lambda$ (Eqs. (88)–(89) below), without requiring the additional bookkeeping of the spherical term. Substituting Eq. (84) into Eq. (87) yields a quadratic equation in the unknown load-factor correction $\delta\lambda$:

$$\alpha_1 \delta\lambda^2 + \alpha_2 \delta\lambda + \alpha_3 = 0, \quad (88)$$

with

$$\begin{aligned}
\alpha_1 &= \delta \mathbf{u}_t^\top \delta \mathbf{u}_t, \\
\alpha_2 &= 2 \left({}^k \Delta \mathbf{N}^{n+1} + \delta \bar{\mathbf{u}} \right)^\top \delta \mathbf{u}_t, \\
\alpha_3 &= \left({}^k \Delta \mathbf{N}^{n+1} + \delta \bar{\mathbf{u}} \right)^\top \left({}^k \Delta \mathbf{N}^{n+1} + \delta \bar{\mathbf{u}} \right) - \Delta l^2.
\end{aligned} \tag{89}$$

Eq. (88) admits, in general, two real roots, $\delta \lambda_1$ and $\delta \lambda_2$ (geometrically, the two intersections between the cylindrical constraint surface and the tangent line to the equilibrium path in the augmented $(n_f + 1)$ -dimensional space of the free displacements $\mathbf{N}$ and the displacement factor λ), provided that the discriminant $\alpha_2^2 - 4\alpha_1\alpha_3$ is non-negative.

To select the physically meaningful root and avoid retracing the equilibrium path already covered within the increment, the criterion proposed by de Souza Neto and Feng [40] is adopted: the correct root is the one for which the resulting incremental displacement vector ${}^{k+1} \Delta \mathbf{N}^{n+1}$ forms an acute angle with the incremental displacement of the previous iteration, ${}^k \Delta \mathbf{N}^{n+1}$:

$$\bar{\theta}_m = \left({}^k \Delta \mathbf{N}^{n+1} + \delta \mathbf{N}_m \right)^\top \left({}^k \Delta \mathbf{N}^{n+1} \right), \quad m = 1, 2, \tag{90}$$

the correct solution corresponding to $\bar{\theta}_m > 0$. The full decision logic, including the degenerate cases in which both angles are acute or the discriminant is negative, is summarized in Algorithm 1.

4.3.2 Predictor step

The predictor provides the initial estimate for a new load increment and is constructed starting from the last converged equilibrium state $(\mathbf{D}^n, \mathbf{N}^n)$. Since this state satisfies the equilibrium equations, the residual at the beginning of the new increment vanishes,

$${}^0 \mathbf{R}_N^{n+1} = \mathbf{R}_N^n = \mathbf{0}.$$

Consequently, the residual-driven contribution in Eq. (85) is identically zero,

$$\delta \bar{\mathbf{u}} = \mathbf{0},$$

Algorithm 1: Corrector algorithm (root-selection procedure) [14, 40].

```

1  Compute  $\alpha_1, \alpha_2, \alpha_3$  from Eq. (89) and the discriminant
    $\Delta_{\text{disc}} = \alpha_2^2 - 4\alpha_1\alpha_3$  of the quadratic Eq. (88)
2  if  $\Delta_{\text{disc}} \geq 0$  then
3      Compute  $\delta\lambda_1, \delta\lambda_2$  from Eq. (88), and the corresponding  $\delta\mathbf{N}_1, \delta\mathbf{N}_2$ 
       via Eq. (84) and Eq. (85)
4      Compute  $\bar{\theta}_1, \bar{\theta}_2$  from Eq. (90)
5      if  $\bar{\theta}_1 > 0$  and  $\bar{\theta}_2 \leq 0$  then
6          select  $\delta\lambda \leftarrow \delta\lambda_1$ 
7      else
8          if  $\bar{\theta}_1 \leq 0$  and  $\bar{\theta}_2 > 0$  then
9              select  $\delta\lambda \leftarrow \delta\lambda_2$ 
10         else
11             if  $\bar{\theta}_1 \geq 0$  and  $\bar{\theta}_2 \geq 0$  then
12                 select the root closest to the linearized solution  $-\alpha_3/\alpha_2$ 
13             else
14                 both roots non-physical: reduce  $\Delta l$  and restart the
                  increment
15             end
16         end
17     end
18 else
19     no real intersection ( $\Delta_{\text{disc}} < 0$ ): reduce  $\Delta l$  and restart the
       increment
20 end

```

and the first incremental correction contains only the tangential component,

$$\delta \mathbf{N} = \delta \lambda \delta \mathbf{u}_t.$$

The predictor therefore corresponds geometrically to advancing from the last converged equilibrium point along the tangent to the equilibrium path. The magnitude of this tangential step is determined by the arc-length constraint. In fact, at the beginning of the new increment no previous incremental displacement is available,

$${}^0\Delta \mathbf{N}^{n+1} = \mathbf{0}.$$

Therefore, the angle criterion of Eq. (90), which requires a previous incremental direction, cannot be applied at this stage. Instead, the arc-length constraint of Eq. (87) is applied directly to the tangential predictor,

$$\delta \lambda = \text{sgn}(\det(\mathbf{K}_{ff})) \frac{\Delta l}{\sqrt{\delta \mathbf{u}_t^T \delta \mathbf{u}_t}}, \quad (91)$$

which determines the magnitude of $\delta \lambda$. The sign of the tangent stiffness is introduced to ensure that, when a limit point is reached, the direction of the predictor changes consistently with the transition to the subsequent equilibrium branch. The resulting state provides the initial estimate from which the corrector iterations of Algorithm 1 are subsequently performed.

4.3.3 Update and convergence criterion

Once $\delta \lambda$ has been selected (via Algorithm 1 for $k \geq 1$, or via Eq. (91) for the predictor step) the load factor and the free displacements are updated as

$${}^{k+1}\Delta \lambda^{n+1} = {}^k\Delta \lambda^{n+1} + \delta \lambda, \quad {}^{k+1}\Delta \mathbf{N}^{n+1} = {}^k\Delta \mathbf{N}^{n+1} + \delta \bar{\mathbf{u}} + \delta \lambda \delta \mathbf{u}_t, \quad (92)$$

and the trial state is reconstructed as $\mathbf{N}^{n+1} = \mathbf{N}^n + {}^{k+1}\Delta \mathbf{N}^{n+1}$, $\mathbf{D}^{n+1} = (\lambda^n + {}^{k+1}\Delta \lambda^{n+1}) \mathbf{d}$. The iterative loop is repeated until the residual norm satisfies the convergence criterion

$$\|{}^k\mathbf{R}_N^{n+1}\| \leq \bar{\epsilon}, \quad (93)$$

with $\bar{\epsilon}$ a prescribed absolute tolerance on the out-of-balance force. A relative criterion, $\|{}^k\mathbf{R}_N^{n+1}\|/\|{}^0\mathbf{R}_N^{n+1}\| \leq \bar{\epsilon}$, is checked in parallel to ensure robust convergence detection also when the absolute residual level varies significantly across increments (e.g. close to the peak load).

4.3.4 Construction of the tangent stiffness matrix

As shown in Section 2.2, $\mathbf{f}$ is collinear with $\mathbf{M}$ and its magnitude f depends on the deformation of the bond through the bond elongation but each bond can be treated, for the purpose of assembling the residual and its derivative, as a two-node nonlinear bar element aligned with the *reference* bond vector [14]. For a bar element connecting nodes i and j , the secant axial stiffness matrix is defined by

$$\bar{\mathbf{K}}_{(i,j)}^{n,local} = \frac{f(e_{ij}^n)\beta_{ij}V_iV_j}{e_{ij}^n} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (94)$$

and mapping it into the global coordinate system via the direction cosines $\mathbf{c} = \boldsymbol{\xi}/\|\boldsymbol{\xi}\|$, specifically by employing the tensor (outer) product $\otimes$, results in the $2n \times 2n$ secant stiffness matrix of the bond (with $n = 2$ and $n = 3$ corresponding to the two- and three-dimensional cases, respectively):

$$\bar{\mathbf{K}}_{(i,j)}^{n,global} = \frac{f(e_{ij}^n)}{e_{ij}^n} \underbrace{\frac{\beta_{ij}V_iV_j}{\|\boldsymbol{\xi}\|^2} \begin{bmatrix} \boldsymbol{\xi} \otimes \boldsymbol{\xi} & -\boldsymbol{\xi} \otimes \boldsymbol{\xi} \\ -\boldsymbol{\xi} \otimes \boldsymbol{\xi} & \boldsymbol{\xi} \otimes \boldsymbol{\xi} \end{bmatrix}}_{\mathbf{G}_{ij}} = k_s(e_{ij}^n) \mathbf{G}_{ij}, \quad (95)$$

where $\mathbf{G}_{ij}$ is a purely *geometric* matrix, built from the reference bond vector $\boldsymbol{\xi}$ alone and therefore independent of the current displacement i.e. consistently with the small-rotation truss idealization, in which the bond's geometric orientation is fixed at its reference configuration.

Denoting the bond's contribution to the out-of-balance residual as $\mathbf{R}(\mathbf{u}) = \bar{\mathbf{K}}(\mathbf{u})\mathbf{u}$ (recalling that in the case under analysis $\mathbf{R} = \mathbf{F}$ since no external

load are prescribed), its derivative with respect to the nodal displacements — the *tangent* stiffness matrix — is obtained, exactly as for a general nonlinear stiffness formulation, by applying the product rule to the secant relation, since $\bar{\mathbf{K}}$ itself depends on $\mathbf{u}$:

$$\widetilde{\mathbf{K}}_{(i,j)}^n = \frac{d\mathbf{R}}{d\mathbf{u}} = \bar{\mathbf{K}}(\mathbf{u}) + \frac{d\bar{\mathbf{K}}(\mathbf{u})}{d\mathbf{u}} \mathbf{u}. \quad (96)$$

Since $\bar{\mathbf{K}}$ from Eq. (95) depends on $\mathbf{u}$ only through the scalar elongation $e(\mathbf{u})$, while $\mathbf{G}_{ij}$ is constant, the chain rule gives

$$\frac{d\bar{\mathbf{K}}(\mathbf{u})}{d\mathbf{u}} \mathbf{u} = \frac{dk_s}{de} \left(\frac{de}{d\mathbf{u}} \cdot \mathbf{u} \right) \mathbf{G}_{ij}. \quad (97)$$

Consistently with the same geometrically linear treatment already adopted for $\mathbf{G}_{ij}$, i.e. evaluating the sensitivity of the elongation along the bond's fixed reference direction, rather than the instantaneously deformed one, the elongation $e(\mathbf{u})$ is effectively homogeneous in $\mathbf{u}$, so that $de/d\mathbf{u} \cdot \mathbf{u} = e$. Substituting Eq. (97) and Eq. (95) into Eq. (96),

$$\widetilde{\mathbf{K}}_{(i,j)}^n = \left[k_s + \frac{dk_s}{de} e \right] \mathbf{G}_{ij} = \left[\frac{f}{e} + \left(\frac{f' e - f}{e^2} \right) e \right] \mathbf{G}_{ij} = f' \mathbf{G}_{ij}, \quad (98)$$

where the secant terms cancel exactly, leaving the tangent stiffness matrix governed solely by the local slope of the pairwise force–elongation law, $f' = df/de$, in place of the secant ratio f/e . Since $e = s\|\boldsymbol{\xi}\|$, the chain rule further gives

$$f' = \frac{df}{ds} \frac{ds}{de} = \frac{df}{ds} \frac{1}{\|\boldsymbol{\xi}\|} = \frac{c_T(t, \boldsymbol{\xi})}{\|\boldsymbol{\xi}\|}, \quad (99)$$

motivating the definition of the *tangent micromodulus* i.e. simply the local slope of the pairwise force–stretch law $f(s)$ along the branch currently occupied by the bond, evaluated at the current stretch s_{ij}^n . With this definition, Eq. (98) reads

$$\widetilde{\mathbf{K}}_{(i,j)}^n = \frac{c_T}{\|\boldsymbol{\xi}\|} \mathbf{G}_{ij} = c_T \frac{\beta_{ij} V_i V_j}{\|\boldsymbol{\xi}\|^3} \begin{bmatrix} \boldsymbol{\xi} \otimes \boldsymbol{\xi} & -\boldsymbol{\xi} \otimes \boldsymbol{\xi} \\ -\boldsymbol{\xi} \otimes \boldsymbol{\xi} & \boldsymbol{\xi} \otimes \boldsymbol{\xi} \end{bmatrix}. \quad (100)$$

As a worked example, let's consider the bilinear model of Eq. (24) on its softening loading branch ($s_0 < s_{\max} \leq s_c$, $s_{\max} = s$). Simplifying Eq. (24) brings to:

$$f(s) = c \left(1 - \frac{s - s_0}{s} \frac{s_c}{s_c - s_0} \right) s = \frac{c s_0 (s_c - s)}{s_c - s_0}. \quad (101)$$

Differentiating $f(s)$ with respect to s as shown in Eq. (99) then gives directly

$$c_T = \frac{\partial f}{\partial s} = -\frac{c s_0}{s_c - s_0} = \frac{c s_0}{s_0 - s_c}, \quad (102)$$

which reproduces exactly the second branch of Eq. (103). On the unloading/reloading branch ($s_0 < s_{\max} \leq s_c$, $s < s_{\max}$), the bond response is linear, so that the tangent and secant moduli coincide identically, consistent with the third branch of Eq. (103). Finally, beyond the ultimate stretch ($s_{\max} > s_c$) the bond is broken ($f \equiv 0$), so $c_T = 0$:

$$c_T(t, \xi) = \begin{cases} c, & s \leq s_{\max} \leq s_0, \\ \frac{c s_0}{s_0 - s_c}, & s_0 < s_{\max} \leq s_c, \quad s_{\max} = s, \\ \frac{f|_{s_{\max}}}{s_{\max}}, & s < s_{\max}, \\ 0, & s_{\max} > s_c. \end{cases} \quad (103)$$

By applying the same procedure, it can be demonstrated that the tangent micromoduli associated with the exponential model given in Eq. (27) are expressed as follows,

$$c_T(t, \xi) = \begin{cases} c, & s \leq s_{\max} \leq s_0, \\ c s_0 \hat{\gamma} \exp\left(c_2 \frac{-2s_0 + s_c + s}{s_0 - s_c}\right), & s_0 < s_{\max} \leq s_c, \quad s_{\max} = s, \\ \frac{f|_{s_{\max}}}{s_{\max}}, & s < s_{\max}, \\ 0, & s_{\max} > s_c, \end{cases} \quad (104)$$

with coefficient $\hat{\gamma}$ which simplifies to Eq. (105) and $\bar{s} = \frac{s_0 - s}{s_0 - s_c}$.

$$\hat{\gamma} = \frac{(1 + c_1^3) \exp(c_2 \bar{s}) + c_2 \exp(c_2) [1 + (c_1 \bar{s})^3] - 3c_1 \exp(c_2) (c_1 \bar{s})^2}{s_0 - s_c}. \quad (105)$$

Similarly, the trilinear model presented in Eq. (26) becomes

$$c_T(t, \boldsymbol{\xi}) = \begin{cases} c, & s \leq s_{\max} \leq s_0, \\ \frac{c \gamma s_0 (\beta - 1)}{s_c - s_0}, & s_0 < s_{\max} \leq s_\gamma, \quad s_{\max} = s, \\ c s_0 \beta \frac{\gamma}{(s_c - s_0)(1 - \gamma)}, & s_\gamma < s_{\max} \leq s_c, \quad s_{\max} = s, \\ \frac{f|_{s_{\max}}}{s_{\max}}, & s_0 < s_{\max} \leq s_c, \quad s < s_{\max}, \\ 0, & s_{\max} > s_c. \end{cases} \quad (106)$$

Note that, in the unloading/reloading branch ($s < s_{\max}$), c_T reduces in all three models to the ratio $f|_{s_{\max}}/s_{\max}$, consistent with the linear unloading path assumed by the constitutive model of Section 2.4.

4.3.5 Stiffness matrix index allocation

The global tangent stiffness matrix $\mathbf{K}$ is assembled directly in *sparse triplet format* (`row_idx`, `col_idx`, `val`). Each bond connecting nodes i and j contributes only to the *four* 2×2 (in 2D problem) sub-blocks associated with the degrees of freedom of the two endpoints. The bond contribution to the global tangent stiffness matrix in global reference frame is defined as

$$\mathbf{K}_{ij}^{\text{glo}} = \frac{1}{2} c_T \beta_{ij} \frac{V_i V_j}{\|\boldsymbol{\xi}\|^3} (\boldsymbol{\xi} \otimes \boldsymbol{\xi}), \quad (107)$$

where c_T is the tangent micromodulus of the active constitutive branch (Section 4.3.4). The factor $\frac{1}{2}$ compensates for the fact that the neighbor list is traversed once per node: since the outer loop runs over every source node i and, for each i , over every family node $j \in \mathcal{H}_i$, each bond (i, j) is encountered twice, so that halving the contribution at each encounter yields the correct total accumulation into the global matrix. In two dimensions, the degrees of

freedom associated with nodes i and j are

$$\begin{aligned}\text{dof_i} &= [2i - 1, 2i], \\ \text{dof_j} &= [2j - 1, 2j].\end{aligned}\tag{108}$$

The resulting contribution has the block structure illustrated in Figure 4.3, with positive and negative signs corresponding to the diagonal and off-diagonal sub-blocks, respectively.

$$\begin{array}{c} \begin{array}{cc} 2i-1 & 2i \end{array} \\ \begin{array}{cc} 2i-1 & 2i \end{array} \\ \begin{array}{cc} 2j-1 & 2j \end{array} \\ \begin{array}{cc} 2j-1 & 2j \end{array} \end{array} \begin{bmatrix} \begin{bmatrix} +[K]^{glo} \\ \\ \\ \end{bmatrix} & \begin{bmatrix} -[K]^{glo} \\ \\ +[K]^{glo} \\ \end{bmatrix} \\ \begin{bmatrix} \\ +[K]^{glo} \\ -[K]^{glo} \\ \end{bmatrix} & \begin{bmatrix} \\ \\ +[K]^{glo} \\ \end{bmatrix} \end{bmatrix}$$

Figure 4.3: Index allocation of the tangent stiffness contribution associated with a bond connecting nodes i and j into the global tangent stiffness matrix.

For example, the (i, i) block is inserted into the triplet arrays as

$$\begin{aligned}\text{row_idx} &= [\text{dof_i}(1), \text{dof_i}(1), \text{dof_i}(2), \text{dof_i}(2)], \\ \text{col_idx} &= [\text{dof_i}(1), \text{dof_i}(2), \text{dof_i}(1), \text{dof_i}(2)], \\ \text{val} &= [K_{ij}^{\text{glo}}(1, 1), K_{ij}^{\text{glo}}(1, 2), K_{ij}^{\text{glo}}(2, 1), K_{ij}^{\text{glo}}(2, 2)].\end{aligned}\tag{109}$$

The remaining (i, j) , (j, i) , and (j, j) sub-blocks are assembled analogously, using the corresponding degree-of-freedom indices and the signs shown in Figure 4.3. Once all bond contributions have been appended, the global matrix is constructed as

$$\mathbf{K} = \text{sparse}(\text{row_idx}, \text{col_idx}, \text{val}, 2N, 2N).\tag{110}$$

MATLAB automatically sums entries having identical row-column indices, thereby accumulating the contributions of all bonds acting on the same global degrees of freedom. The resulting matrix is stored directly in *compressed sparse*

column (*CSC*) format. Finally, the free-free and free-prescribed blocks required to compute Eq. (86) are extracted as

$$\mathbf{K}_{ff} = \mathbf{K}(\text{dof_N}, \text{dof_N}), \quad \mathbf{K}_{fp} = \mathbf{K}(\text{dof_N}, \text{dof_D}), \quad (111)$$

where `dof_N` and `dof_D` contain the global indices of the free and prescribed degrees of freedom, respectively.

4.4 Implementation strategies

Beyond the core algorithm presented in Section 4.3, a number of implementation choices were adopted in the MATLAB solver developed for this work, aimed at improving robustness and computational efficiency, particularly in the context of large batches of automated simulations used to generate training data for the neural network described in Chapter 6.

- **Restart on non-physical roots.** Consistently with Algorithm 1, whenever the discriminant of Eq. (88) is negative, or both roots yield a non-acute angle $\bar{\theta}_m$, the current increment is discarded, the arc length is reduced ($\Delta l \leftarrow 0.5 \Delta l$), and the increment is restarted from the last converged configuration, without incrementing the step counter n . This strategy of reducing the arc length upon encountering a non-physical root is motivated by preliminary numerical observations: increasing Δl in this situation was found to always restore convergence of the corrector iterations, but at the cost of large, non-physical jumps along the equilibrium path, associated with the simultaneous failure of numerous bonds and an abrupt transition to a markedly different stable configuration. Allowing multiple bonds to fail simultaneously prevents the necessary redistribution of forces among adjacent bonds, thereby altering the residual stiffness of the structure and introducing severe numerical inaccuracies in the predicted crack propagation path and load-displacement response [41]. Reducing Δl , by contrast, forces the solver to resolve the softening response with a finer resolution.

- **Bond status update and irreversibility.** Only at the end of each converged increment, the maximum stretch $s_{\max}$ attained by every bond is updated, and the Boolean state μ_{ij} of Eq. (7) is set to zero for all bonds whose stretch has exceeded s_c , enforcing the irreversibility of bond failure.
- **Early termination criteria.** Two complementary criteria are used to truncate a simulation once it no longer contributes information to the response curve: a *plateau* criterion, which stops the run once the relative variation of the reaction force over a sliding window of increments, evaluated sufficiently far past the peak load, falls below a prescribed tolerance; and a *complete fracture* criterion, which stops the run once the reaction force drops below a small fraction of its peak value. Both criteria are only activated past the peak load in order to ensure optimal data storage.
- **Computational budget management.** Given the scale of the dataset generation campaign, a *per-run timeout* and a *global wall-clock budget* are enforced, so that a single out-layer run cannot compromise the entire batch. Results are saved to disk immediately upon completion of each run, rather than being accumulated in memory, to avoid data loss in the event of an unexpected interruption.

4.4.1 Adaptive arc-length step size

At the end of each converged increment n , the arc-length parameter is updated for the next increment according to the classical heuristic of Crisfield [37],

$$\Delta l_{n+1} = \Delta l_n \sqrt{\frac{I_d}{I_p^n}}, \quad (112)$$

where I_p^n is the number of corrector iterations required to converge increment n , and I_d is a desired iteration count. Eq. (112) increases Δl when convergence is achieved faster than desired, and decreases it otherwise, so as to keep the number of corrector iterations approximately constant across the

simulation.

Rather than applying Eq. (112) unconditionally, the resulting Δl_{n+1} is combined with two additional bounds before being accepted,

$$\Delta l_{n+1} \leftarrow \min(\Delta l_{n+1}, \Delta l_{\max}^n, 2 \Delta l_n), \quad (113)$$

namely an upper bound $\Delta l_{\max}^n$ and a growth cap limiting the increase of Δl to at most a factor of two per increment, which prevents the heuristic of Eq. (112) from prescribing an excessively large jump after a series of fast-converging increments. The upper bound $\Delta l_{\max}^n$ is chosen adaptively, as can be deduced from Figure 4.4, depending on the current state of the response, since the numerical difficulty of the problem changes markedly across the loading history:

- **Elastic regime (1° region).** The arc length follows the general heuristic of Eq. (112), capped at $\Delta l_{\max}^n = 10^{-5}$ and subject to the growth cap of Eq. (113): a slight bias is thus introduced, allowing Δl to grow gently whenever the corrector converges in fewer than the desired number of iterations ($I_d = 5$), so as to avoid spending an excessive number of small increments traversing the initial, predetermined elastic stretch of the response. In this regime the tangential direction provided by the predictor alone is often already sufficient to satisfy equilibrium, so that convergence is frequently achieved in a single corrector iteration ($I_p^n = 1$), giving a growth factor $\sqrt{I_d/I_p^n} = \sqrt{5}$ in Eq. (112); in this case, it is the growth cap of Eq. (113) that limits the increase of Δl to at most a factor of two per increment. When two iterations are instead required ($I_p^n = 2$), the growth factor reduces to $\sqrt{5/2} = 1.58$, which is instead smaller than the cap and therefore governs the update directly.
- **Onset of softening and Near-peak region (2° region)** ($E_{\text{calc}} < 0.95 E$). Since the location of the peak reaction is not known a priori, the transition out of the purely elastic regime is not detected from the displacement factor λ directly, but inferred by monitoring the secant Young's modulus computed from the current state: once E_{calc} falls below

95% of the initial elastic modulus E , the onset of bond damage is inferred and the general heuristic of Eq. (112) is overridden. The arc length is instead forced to a small, fixed value, $\Delta l = 10^{-6}$, so as to resolve this critical region with high resolution regardless of how quickly the corrector iterations happen to converge. This regime is retained until the peak has been detected and the current displacement factor remains within a prescribed multiple of λ_{peak} ($\lambda < \kappa_{\text{np}} \lambda_{\text{peak}}$, with $\kappa_{\text{np}} = 2$ in this work, chosen heuristically based on what observed across the simulation campaign), so as to adequately resolve the region immediately following the transition to softening, where snap-through and, potentially, snap-back instabilities are most likely to occur (Section 4.2).

- **Deep into the softening tail (3° region)** ($E_{\text{calc}} < 0.95 E$, and $\lambda \geq \kappa_{\text{np}} \lambda_{\text{peak}}$). Once the displacement factor has moved sufficiently far past the one at which the peak reaction was recorded, the response is assumed to have entered the plateau-like tail of the softening branch, where the risk of snap-back has receded. In this regime the arc length is grown progressively at every increment, $\Delta l \leftarrow 1.1 \Delta l$ allowing the solver to traverse the tail with fewer, larger increments.

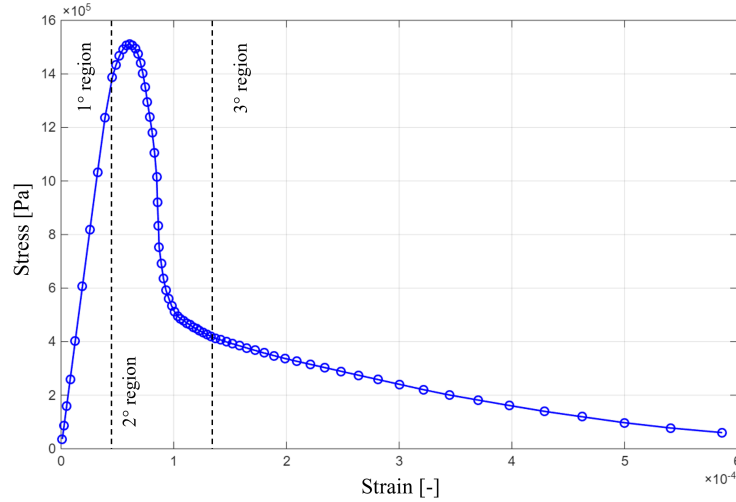


Figure 4.4: Schematic representation of the arc-length adaptation routine.

In addition to this *arc-adaptation* routine, the arc length is dynamically adjusted during the corrector iterations, as outlined in Section 4.4, whenever

a non-physical root is encountered. In such cases, Δl is halved and the increment is restarted, provided it remains above a minimum admissible value $\Delta l_{\min}$, below which execution is terminated.

Chapter 5

Dataset Generation

The methodology developed in this thesis for the identification of the PD laws relies entirely on supervised learning (Chapter 3), which in turn requires a large collection of stress–strain curves labeled with their known, ground-truth constitutive law and material parameters. No such experimental dataset exists at the scale required to train a deep neural network, nor could it, since the constitutive law underlying an experimentally measured curve is itself the unknown required to be identified. The training data are therefore generated entirely synthetically, by running the displacement-controlled arc-length solver described in Chapter 4 a large number of times, each time with a different, randomly sampled combination of material parameters and with each of the three candidate softening laws introduced in Section 2.4 equally represented. This chapter describes how this simulation campaign was designed and executed, how the resulting curves were organized and stored, and how they were subsequently prepared for use as training data, closing with a discussion of the resulting dataset composition and of the limitations inherent to a purely synthetic training set.

5.1 Sampling strategy for material parameters

In order to obtain a model able to generalize to materials that were not explicitly represented in the training set, the synthetic dataset must provide a representative coverage of the space of physically plausible material parameters for the class of quasi–brittle materials. Each simulation is therefore assigned an independently sampled combination of the three material parameters entering the constitutive laws of Section 2.4: the elastic modulus E , the tensile strength f_t , and the fracture energy G_0 . Regardless the generation of such parameters, in the exponential softening law two additional shape parame-

ters, c_1 and c_2 characterize the constitutive relation. These are not treated as independently sampled quantities in the dataset generation and their values ($c_1 = 3.00$, $c_2 = 6.93$ for normalweight concrete or $c_1 = 1.00$, $c_2 = 5.64$ for lightweight concrete) are instead fixed a priori, taken directly from [15], where they were obtained by least-squares fitting a proposed model (reported below) for the stress-crack opening curve to the experimental data obtained by a deformation-controlled uniaxial tensile tests on concrete:

$$\frac{\sigma}{f_t} = \left\{ 1 + \left(c_1 \frac{\delta}{\delta_0} \right)^3 \right\} \exp \left(-c_2 \frac{\delta}{\delta_0} \right) - \frac{\delta}{\delta_0} \left(1 + c_1^3 \right) \exp(-c_2).$$

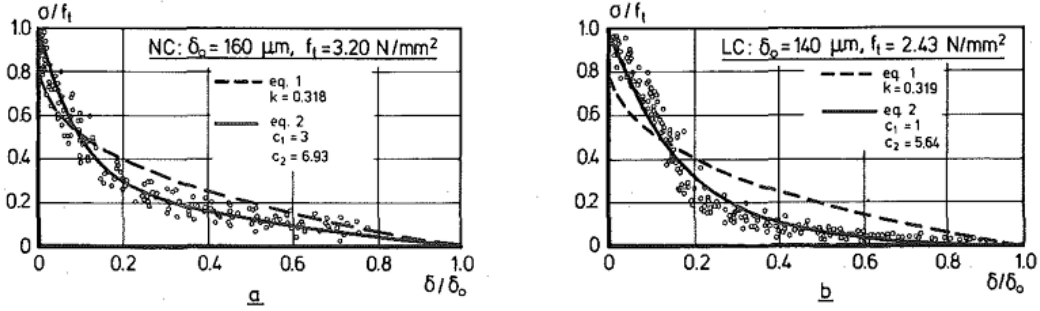


Figure 5.1: Normalized stress–crack-opening curves for (a) normalweight concrete (NC) and (b) lightweight concrete (LC), showing experimental data points together with two possible law fits, from which the shape parameters c_1 and c_2 are obtained [15].

For what concerns the trilinear constitutive law, γ and β , i.e. the kick point identification multiplier, are also fixed a priori based on calibration done on experimental data, but with different approaches that in some cases include finding those parameter which minimize discrepancy between the measured and simulated load–displacement curves. A well accredited set of value proposed by Bažant [42] corresponds to $\gamma = 1/4$ and $\beta = 20/3$ for concrete, resumed also by other successive works [43, 14].

Concerning three main parameters, they are drawn from a multivariate normal

distribution,

$$\begin{pmatrix} E \\ f_t \\ G_0 \end{pmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \boldsymbol{\mu} = \begin{pmatrix} \mu_E \\ \mu_{f_t} \\ \mu_{G_0} \end{pmatrix}, \quad \boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}) \mathbf{P} \text{diag}(\boldsymbol{\sigma}), \quad (114)$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ collect the mean and standard deviation of each parameter, and $\mathbf{P}$ is a correlation matrix which collapses to the identity matrix, $\mathbf{P} = \mathbf{I}$, in the absence of any prescribed correlation between parameters, as in the case assumed in the present work for simplicity. The adopted mean values and standard deviations are summarized in Table 5.1. These values are selected as representative reference values for the range of mechanical properties encountered in the literature regarding concrete-based quasi-brittle materials [44, 45, 46, 47] and through information coming from material dataset like [48].

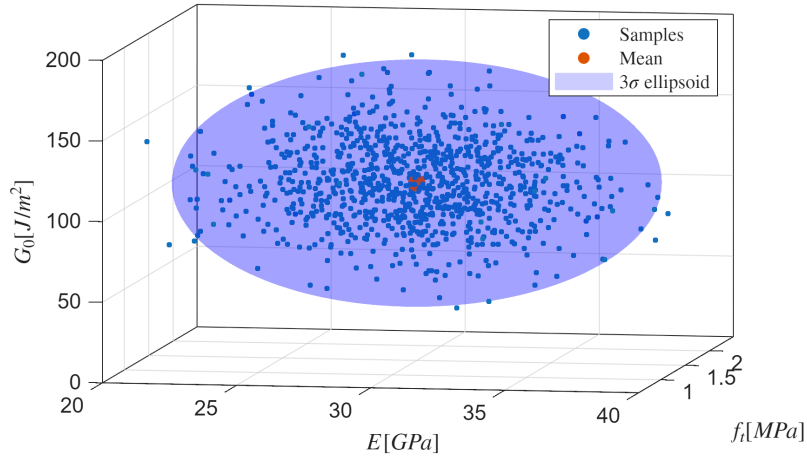


Figure 5.2: Distribution of the sampled material parameters (E, f_t, G_0) , with the 3σ confidence ellipsoid computed from the sample covariance.

Sampling from Eq. (114) is performed via the *Cholesky factorization* of $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^\top$, so that a sample is obtained as $\boldsymbol{\mu} + \mathbf{L}\mathbf{z}$, with $\mathbf{z}$ a vector of independent standard normal random variables; this formulation allows correlated sampling to be introduced in future extensions of the dataset simply by populating the off-diagonal entries of $\mathbf{P}$, without otherwise changing the sampling procedure.

Property	Mean Value	Standard Deviation
E [GPa]	30	3
f_t [MPa]	1.5	0.3
G_0 [J/m ²]	110	25

Table 5.1: Representative reference values and standard deviations of the mechanical properties for quasi-brittle materials.

To validate the correct implementation of the sampling procedure defined in Eq. (114), Figure 5.3 reports the marginal distributions of the realized E , f_t , and G_0 values in the dataset, overlaid with the corresponding theoretical normal probability density function (PDF). The close agreement between the sampled histograms and the target curves confirms that the three parameters are drawn with the prescribed mean and standard deviation, and that no bias is introduced by the Cholesky-based sampling scheme.

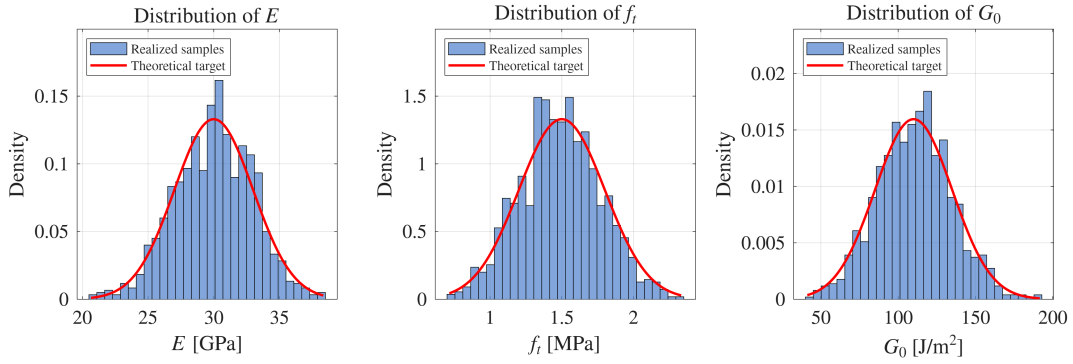


Figure 5.3: Realized marginal distributions of the sampled material parameters E , f_t and G_0 , overlaid with the theoretical target normal PDF of Eq. (114).

5.2 Simulation parameters and dataset organization

Beyond the material parameters, which are sampled differently at each simulation, the dataset generation campaign requires a number of further specifications that remain invariant across all runs. These include the definition of the specimen geometry and its peridynamic discretization, type and magnitude

of the boundary conditions, solver parameters governing the arc-length algorithm, and the choice of data format used to store the resulting stress-strain responses for subsequent analysis.

Geometry and PD discretization

Based on the experimental set-up used in direct tensile test of concrete [45, 46], a flat dog-bone specimen geometry is adopted in this work. The geometrical dimension selected are briefly sketched in Figure 5.4 as reference, with the thickness fixed to $h = 25 \text{ mm}$. The gradual reduction of the cross-section characterizing the central portion, known as *gauge section*, promotes failure in the central region far away from the boundary layers where the displacement is prescribed, avoiding risk of an unrepresentative crack initiation, at the load-application ends.

To further control the location of crack initialization and to facilitate convergence of the arc-length solver, a pre-crack of length equal to one grid spacing Δx is introduced at the mid-height of the gauge section, with the crack surface perpendicular to the direction of the applied displacement. The pre-crack is implemented by deactivating, prior to the start of the simulation, all bonds whose two endpoints lie on opposite sides of the crack passing within the notch region: the corresponding entry of the bond-state matrix μ is set to zero, so that no force is transmitted across the pre-crack from the very first load step. The pre-crack length is deliberately kept to the minimum resolvable by the discretization, i.e. a single Δx , as illustrated in the detailed views of Figure 5.4, since the dog-bone geometry already localizes the maximum stress in the gauge section on its own, so that an extended notch is not required to achieve this localization. The sole purpose of the pre-crack here is to deterministically fix the crack initiation point and break the otherwise perfect symmetry of the domain, which could lead to an ill-defined or numerically unstable initiation point under the arc-length solver.

For what concern the discretization adopted, convergence criteria discussed in Section 2.5.4 are carried out prior to the campaign, bringing to select $m = 4$

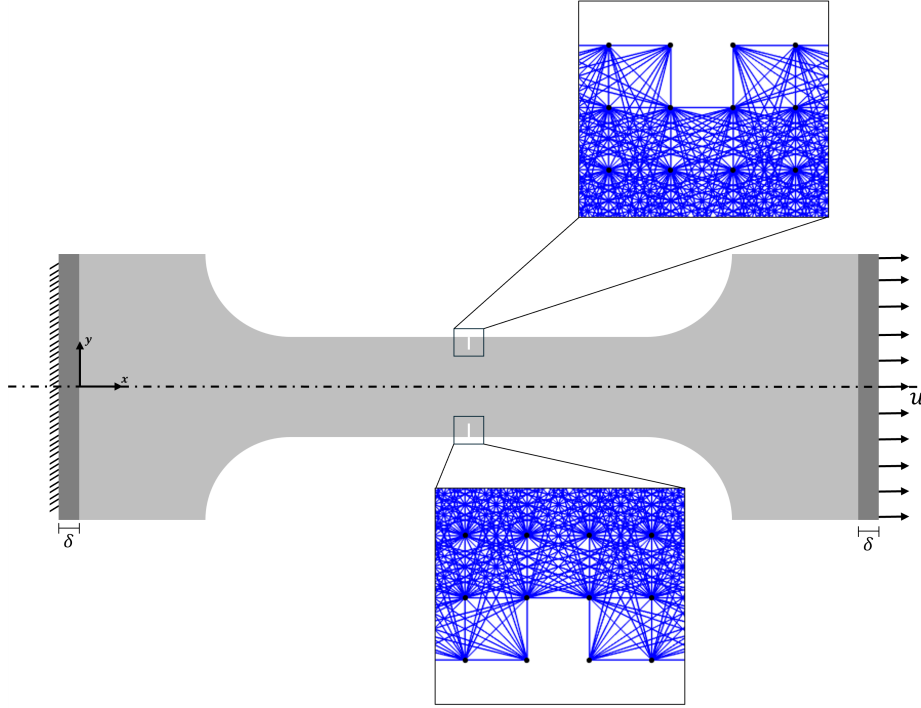


Figure 5.4: Dog-bone specimen geometry used for the PD tensile tests, with fixed (left) and prescribed-displacement (right) boundary conditions, together with a detail of the bond connectivity in correspondence of the two pre-crack.

and $\Delta x = 2$ mm and resulting in a total amount of nodes equal to 10 055 (including the one of the boundary layers discussed next). This combination was selected as a compromise between numerical accuracy and computational cost: $m = 4$ is adopted to keep limited the grid-orientation effects associated with lower m ratios [49], while remaining below the range where computational cost become excessive, and the corresponding Δx keeps the total number of nodes, and hence the size of the bond connectivity and stiffness matrices assembled at every load step, within a range compatible with running the full batch campaign.

Boundary condition

The displacement boundary conditions are not applied on a single layer of surface nodes, but on two auxiliary strips of nodes of thickness equal to one

horizon δ , added at both ends of the specimen. This fictitious boundary layer is a common strategy in PD to account for the non-local nature of the boundary conditions and to complete the neighbourhoods family of non-fictitious nodes located near the physical boundary [50, 18, 51]. Let's consider the reference frame illustrated in Figure 5.4, with the origin lying along the longitudinal axis of symmetry and the x -axis aligned with the loading direction, the two boundary layers are identified as

$$\text{left_node} = \{i : x_i < 0\}, \quad \text{right_node} = \{i : x_i > l_x\}, \quad (115)$$

and denoted collectively as `Dirichlet_nodes`, with the remaining nodes defined as `Neumann_nodes`, which are left unconstrained (l_x refers to the specimens length). Instead to apply on `Dirichlet_nodes` a simple clamped-prescribed pair, a more consistent with a uniform uniaxial strain state displacement field is prescribed:

$$u_x = \varepsilon x, \quad u_y = -\nu \varepsilon y, \quad \varepsilon = d_{\text{ref}}/l_x, \quad (116)$$

where d_{ref} is the target end displacement, identified by the arc-length solver as **d**. This choice arises from two reasons:

1. the field automatically removes all rigid-body modes (horizontal translation, since $u_x \rightarrow 0$ as $x \rightarrow 0$; vertical translation, since $u_y \rightarrow 0$ as $y \rightarrow 0$; and rotation, since no residual rotational term is present), without requiring an additional constraint on a single node;
2. the field is consistent with the transverse (Poisson) contraction expected, so that the boundary layers do not act as a rigid clamp that would suppress this contraction locally and introduce a spurious, artificially triaxial stress state near the `left_nodes` region.

Solver parameters

The arc-length solver relies on a set of parameters governing convergence, step-size adaptation, and the various regimes described in Section 4.4.1; these are kept constant throughout all simulations conducted in this campaign, and are summarized in Table 5.2.

Parameter	Value
Residual convergence tolerance $\bar{\epsilon}$	10^{-4}
Desired corrector iteration count I_d (elastic regime)	5
Maximum corrector iterations	30
Maximum number of increments $n_{\max}$	400
Initial arc length Δl_0	$10^{-4} s_0$
Arc-length upper bound $\Delta l_{\max}^n$, elastic regime	10^{-5}
Maximum arc-length growth per increment	$\times 2$
Arc-length value, near-peak regime	10^{-6}
Softening-onset threshold, E_{calc}/E	0.95
Arc-length growth factor, softening tail	1.1
Arc-length reduction factor (non-physical root)	$\times 0.5$

Table 5.2: Arc-length solver parameters, fixed across all simulations of the campaign.

Unlike the other parameters in Table 5.2, the initial arc length Δl_0 is not fixed to a single absolute value shared by all simulations, but is scaled on the elastic limit stretch s_0 of the sampled material. Since s_0 depends on the sampled material parameters f_t and E , and therefore varies from one simulation to another, an initial arc length fixed in absolute terms would represent a very different fraction of the elastic stretch range depending on the sampled material: for a material with a small s_0 , a fixed absolute Δl_0 could overshoot the elastic branch in too coarse increment, while for a material with a large s_0 the same value would result in an unnecessarily large number of increments before reaching the region of interest.

Dataset storage

For every completed simulation, the resulting stress-strain curve is stored together with a unique simulation index `sim_id`, the material parameters E, f_t, G_0 and the type of constitutive law used to generate it, the corresponding micromodulus c and critical stretches s_0, s_c , the run time, and a flag indicating whether the run was truncated early (Section 4.4).

Each simulation is saved twice immediately upon completion: an individual file named with the simulation index (`stress_strain_%03d.mat`) stores the record for the specific simulation alone, while the same record is also appended to a single cumulative file collecting the entire campaign, read back, updated, and written to disk before the next simulation begins. Since batch campaigns can run unattended for several hours and are subject to the per-run timeouts and early terminations discussed in Section 4.4, this incremental scheme ensures that a failure or interruption at any point only risks the simulation currently in progress, rather than the results accumulated up to that point. When a new batch of simulation is started, the existing cumulative file is read, and the simulation index is restarted from one above the highest `sim_id` already stored, so that simulations from different campaigns can be accumulated into the same dataset without index collisions or the need to track progress manually between runs.

5.3 Simulation results

Figure 5.5 presents the numerical solution for a representative simulation, illustrating how the solver strategies, together with the imposed constraint-based numerical setup, govern the traced equilibrium path and the associated evolution of the damage field.

Point (a) marks the peak stress. At this stage, the damage field shows that the accumulated damage is not symmetrically split between the two edges of the ligament, but is more advanced on one side; more generally, across the simulations of the campaign the propagation of the fracture from the two pre-defects can develop either symmetrically, with both sides advancing together toward the center, or, as in the case shown here, predominantly from one side. This asymmetry is consistent with the competitive nature of the two advancing crack tips: the perfectly symmetric configuration, in which both tips advance at the same rate, is extremely sensitive to minor perturbations, so that even a small imbalance between the two — whether introduced by the discretization or simply inherent to the numerical solution — is enough to break the symmetry. What can also be observed at point (a) is that a layer

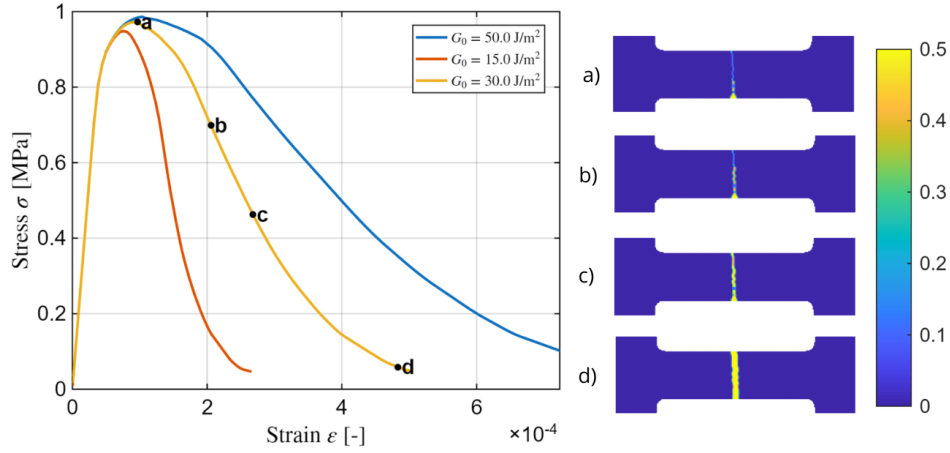


Figure 5.5: Stress-strain response imposing bilinear constitutive law for three different values of fracture energy G_0 , with $f_t = 1$ MPa and $E = 20$ GPa held fixed, together with four snapshots of the nodal damage field φ (points a–d) at increasing levels of deformation for fixed $G_0 = 30.0 \text{ J/m}^2$.

of nodes already spans the entire width of the cross-section with a fraction of their bonds broken ($\varphi \neq 0$), consistent with the reduction in secant stiffness that has already occurred by the time the peak is reached. Moving from (b) to (d), the damage at this layer keeps increasing while the band itself widens: since full separation of the cross-section requires every bond crossing the crack plane to be broken, and such bonds connect nodes lying within one horizon δ on either side of the plane, a band roughly 2δ wide should reach the saturation value of φ before that the reaction force can drop to zero, which is exactly the trend visible between (b) and (d). Overall, the sequence provides a qualitative validation of the solver’s ability to trace the full equilibrium path of a quasi-brittle response, from the elastic branch through the near-peak instability to the softening tail, showing also a coherent crack propagation pattern.

5.4 Dataset results

The full simulation campaign yielded a dataset of valid stress-strain constituted respectively by 539 completed runs for the exponential model, 518 for the trilinear model, and 523 for the bilinear model (Figure 5.6), for a total of over 1500 simulations.

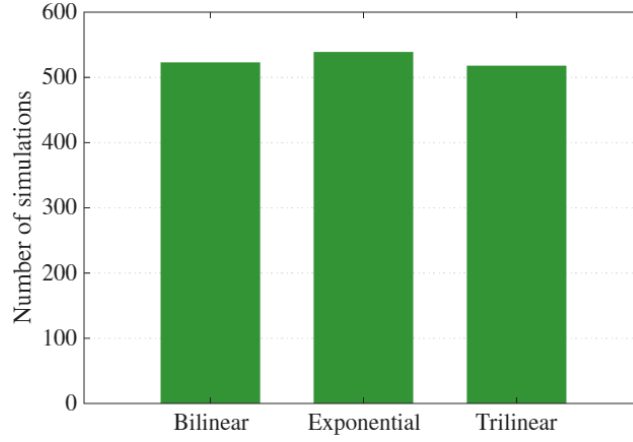


Figure 5.6: Number of valid simulations obtained per constitutive law.

Given the incremental-iterative scheme of the arc-length solution and the fine step-size control required to accurately capture the post-peak structural response, the computational campaign was characterized by a numerical cost, which can be quantified as follows: individual simulations exhibited average wall-clock times in the range of approximately 1650–2100 s, depending on the adopted constitutive model, yielding a cumulative wall-clock time of 702.02 h for the generation of the complete dataset (Figure 5.7). Simulations were run in MATLAB using a parallel pool (`parpool`) distributed across 12 cores to reduce the overall campaign duration.

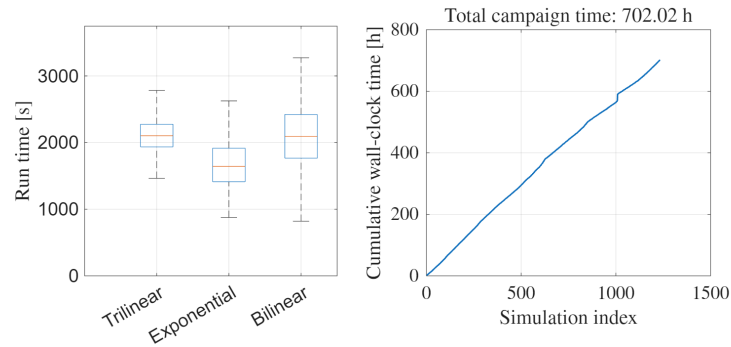


Figure 5.7: Distribution of run time per simulation (on the left) and cumulative wall-clock time of the campaign as a function of simulation index (on the right).

The complete set of stress-strain responses obtained is shown in Figure 5.8, where the characteristic elastic loading branch, peak stress, and post-peak softening tail are consistently reproduced across the sampled parameter space for all three constitutive models.

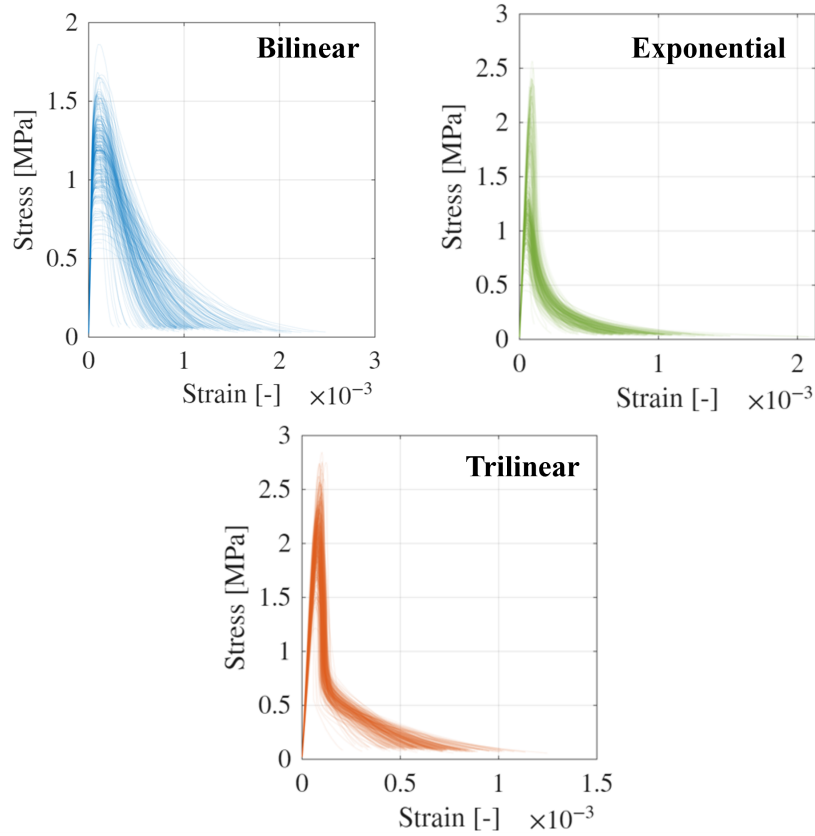


Figure 5.8: Overlay of all valid stress-strain curves obtained from the simulation campaign.

To verify that the framework responds correctly from a physical standpoint to variations in the input material parameters, the models were further examined by fixing two of the three parameters and varying the third: increasing E makes the elastic branch steeper and shifts the peak to a lower strain value while leaving the peak stress essentially unaffected (so that area below the curve can remain the same), as expected since E governs only the initial stiffness (Figure 5.9).

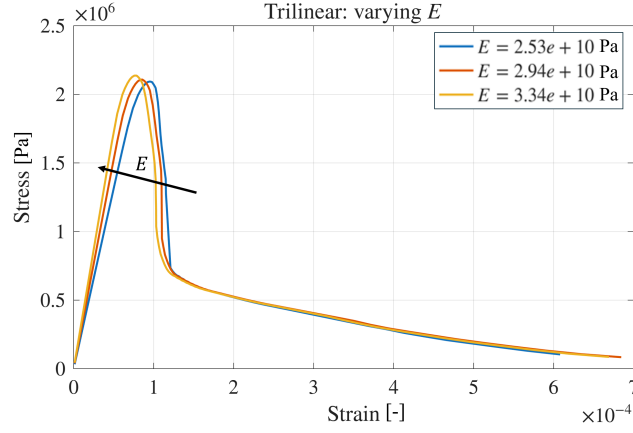


Figure 5.9: Stress-strain response of the trilinear constitutive law for three values of the elastic modulus E , with f_t and G_0 held fixed.

Increasing f_t proportionally raises the peak stress while shifting the softening tail to the left, thereby shortening the curve end to preserve the area behind it and maintain a constant fracture energy G_0 as imposed (Figure 5.10).

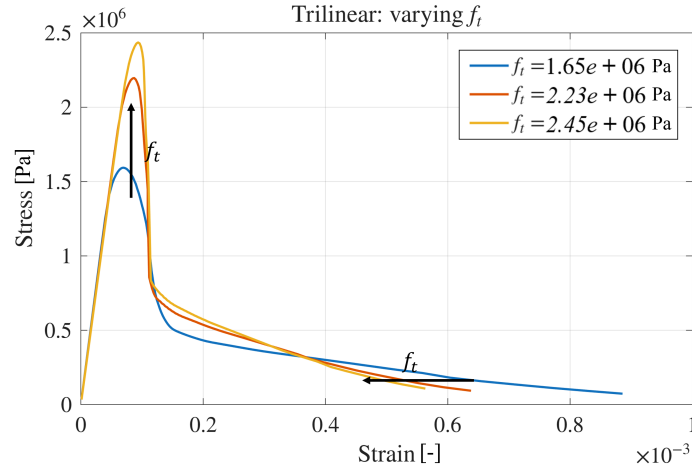


Figure 5.10: Stress-strain response of the trilinear constitutive law for three values of the tensile strength f_t , with E and G_0 held fixed.

Increasing G_0 widens the post-peak softening branch, increasing the area under the curve without altering the peak stress and the slope of the elastic branch,

in agreement with its physical interpretation as the fracture energy governing the energy dissipated during softening (Figure 5.11).

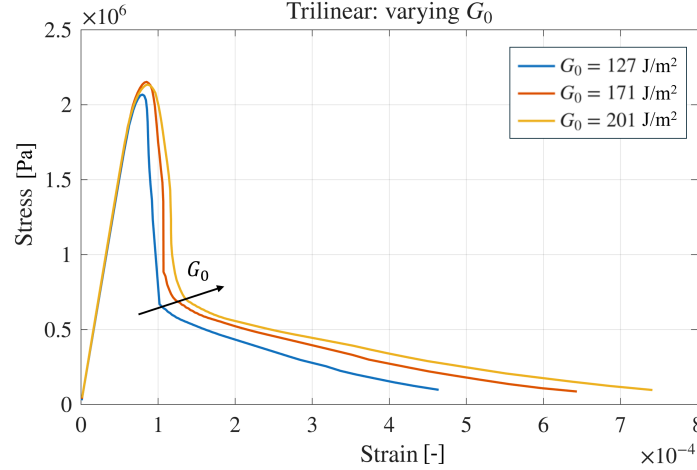


Figure 5.11: Stress-strain response of the trilinear constitutive law for three values of the fracture energy G_0 , with E and f_t held fixed.

In all three cases the simulated response varies in agreement with the selected input parameter, confirming the correct implementation of the framework. Values were deliberately chosen outside the sampling boundaries when needed to accentuate variations in the macroscopic response.

5.5 Data pre-processing for CNN

The raw stress-strain curves obtained from simulation cannot be fed directly to the CNN framework developed and discussed in Chapter 6, for two main reasons:

1. curves obtained from different simulations exhibit variable vector lengths, because the number of increments required to achieve convergence depends on the adaptive arc-length step size (Section 4.4.1) which can vary depending on the specific equilibrium path followed. By contrast, CNN architecture is characterized by a fixed number of inputs for all samples that are fed into the network;

2. absolute scale of the curves varies substantially across the dataset, since it occupies a wide portion of the parameter space (Section 5.1). Unscaled input features with varying magnitudes can dominate the gradient-based optimization process, leading to ill-conditioned loss surfaces, and unstable gradient updates [33]. Furthermore, large scale discrepancies make more difficult for the convolutional filters to extract efficiently scale-invariant shape features, such as the post-peak softening behavior.

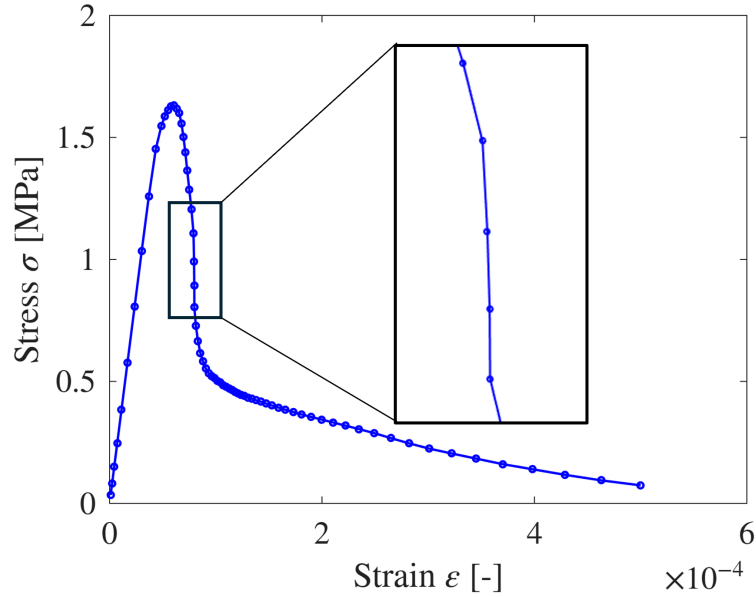


Figure 5.12: Raw stress-strain response, with an enlarged view of the curve in the critical region where the sampled points are closely spaced in terms of strain.

Both issues are addressed in a *pre-processing* stage applied to every simulation before the training. Each curve is resampled onto a uniform grid with a fixed number of points via linear interpolation over its strain domain, ensuring a constant vector length across all network inputs regardless of the solver increments. To avoid loss of information and accurately resolve sharp localized features, the minimum number of sampling points is determined based on the smallest strain increment observed across the adaptive arc-length steps, thereby guaranteeing that even the finest step size is adequately captured during resampling. The required global minimum number of sampling points

N_{smp} is formalized as:

$$N_{smp} = \text{round} \left(\frac{\max_j \left(\varepsilon_{\max}^{(j)} \right)}{\min_j \left(\min_i \left(\varepsilon_{i+1}^{(j)} - \varepsilon_i^{(j)} \right) \right)} \right) + 1, \quad (117)$$

where j indexes the simulations in the dataset and i denotes the discrete steps within each simulation before the re-sampling. N_{smp} represents the ratio between the maximum strain range spanning across all simulations in the dataset, identifying the *worst-case* curve in terms of total deformation domain, and the smallest strain increment computed over all adaptive arc-length solver steps and all simulations, establishing the upper bound for the allowable grid spacing. The obtained value is rounded to the closer integer number and unity is summed, providing a number of sample point equal to 2341.

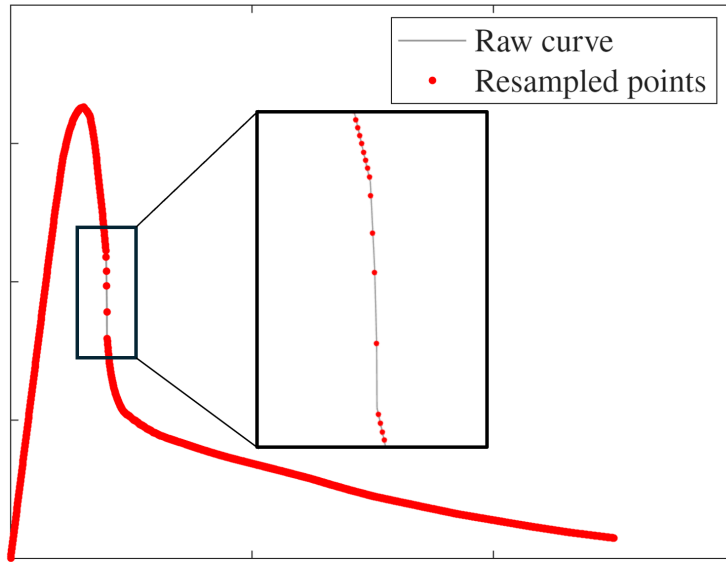


Figure 5.13: Same curve as in Figure 5.12, resampled onto a uniform strain grid consisting of N points using linear interpolation. Enlarged view shows that the resampled data points closely reproduce the original raw curve, including in regions corresponding to very small differences in relative strain.

The resampled curve is then normalized on a per-sample basis, i.e. the strain and stress values of each simulation are divided by the peak strain and peak stress of that same simulation, rather than by the largest peak observed across

the whole dataset. Normalizing by a single maximum value would compress curves with a smaller peak into a narrower band relative to the dominant simulation, reducing the resolution available to represent their post-peak softening and biasing the loss towards the simulations with the largest absolute values. Per-sample normalization instead maps every curve onto the same $[0, 1]$ range regardless of its absolute scale, ensuring comparable resolution for the softening shape across the whole dataset.

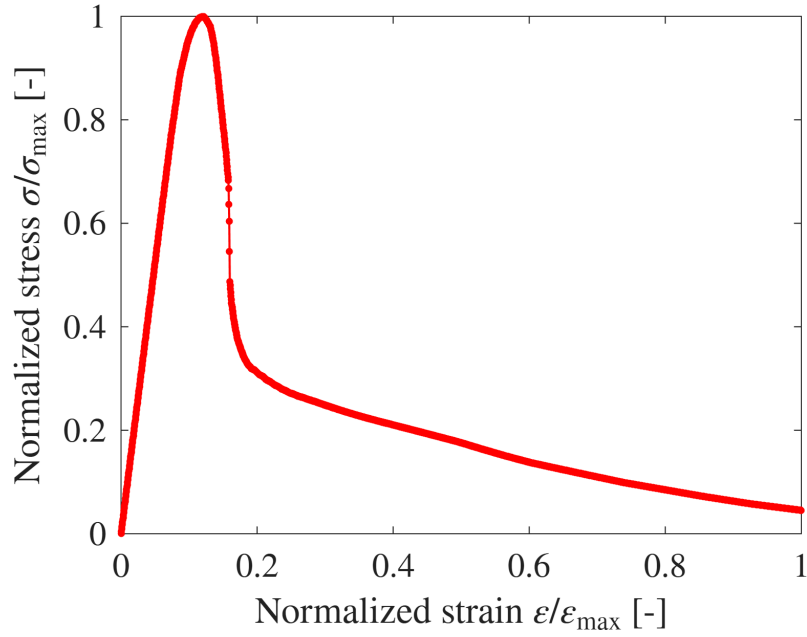


Figure 5.14: Resampled curve of Figure 5.13 after per-sample normalization, mapping both strain and stress onto the $[0, 1]$ range regardless of the simulation’s absolute scale.

Finally, the dataset is split into training, validation, and test sets (70%, 15%, and 15%, respectively), stratified by constitutive law to ensure a proportional class distribution across all subsets (Section 3.1.1).

5.6 Dataset intrinsic limitations

Being entirely synthetic, the dataset generated for this work is subject to a number of limitations that should be kept in mind when interpreting the

results of the classification and regression models trained on it, and reported in Chapter 7:

1. since both the training data and the adopted constitutive laws originate from the same peridynamic model and the same solver, the dataset is exposed to what is commonly referred to as an *inverse crime*: a model can appear to perform well simply because it has implicitly learned the internal consistency of the generating simulator, rather than a genuinely transferable relationship between stress–strain response and constitutive behavior;
2. the sampling distributions adopted in Section 5.1 necessarily bound the range of material behavior represented in the dataset; a material whose parameters lie far outside the sampled range, or whose constitutive response departs qualitatively from all three candidate laws, cannot be expected to be correctly identified by a model trained exclusively on this dataset.

Chapter 6

Adopted CNN Architecture

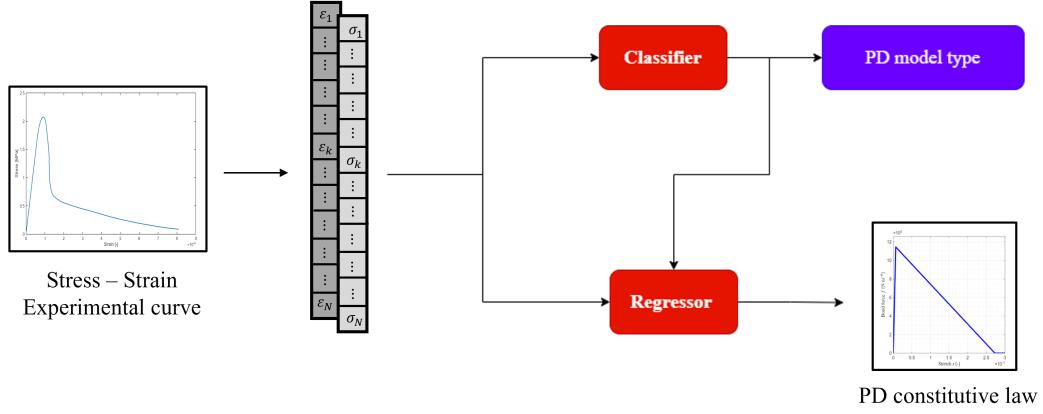


Figure 6.1: High-level overview of the hierarchical CNN architecture.

The identification task addressed in this work is decomposed into two sub-problems of a different nature: determining which of the candidate softening laws *most probably* generated a given stress-strain response and, once the law is known, estimating the corresponding PD parameters. The first is a classification problem over a small, discrete set of classes, while the second is a continuous regression problem aimed at estimating the parameters characterizing the selected model from the classifier.

Rather than training a single network to regress all parameters directly from the curve, regardless of the underlying constitutive law, a *hierarchical* architecture based on the aforementioned separation, shown in Figure 6.1, is employed. This decomposition obviates the need for a single shared output layer to represent parameters whose physical interpretation and admissible ranges differ across constitutive laws. Each regressor is thus free to specialize over the parameter domain specific to its associated law, rather than internally disambiguating among three qualitatively distinct softening behaviors, while the

classifier alone is responsible for selecting the optimal constitutive PD law. This separation of concerns makes both sub-networks more efficient.

As sketched in Figure 6.1, the pre-processed curve is resampled to a fixed length N_{samp} and normalized on a per-sample basis, constituting the input fed to both the classifier and the regressor branches in the form of two separate channels of a one-dimensional sequence.

6.1 Classifier branch

The classifier branch is composed of three convolutional blocks of decreasing kernel size and increasing filter count (kernel sizes K 7, 5, 3; filters F 16, 32, 64 respectively), each consisting of a one-dimensional convolution followed by batch normalization, a ReLU activation, and max pooling (with the exception of the third block) with stride S equal to two, progressively halving the sequence length while extracting features at increasingly coarse resolution. The three convolutional blocks are followed by a global average pooling layer, which collapses the entire sequence into a single value per filter: since the task at this stage is to identify which of three qualitatively distinct curve shapes generated the response (a judgment based on the overall shape of the curve rather than on any one specific location along it) averaging over the full sequence length discards positional detail that is not required for this decision, while yielding a compact, fixed-size representation. The resulting 64-dimensional feature vector is passed through a fully-connected layer of 64 units with ReLU activation and dropout set to a rate of 0.3 (meaning 30% of the neurons were randomly deactivated during each training step to prevent overfitting, as better explained in Section 3.1.2), followed by a final fully-connected layer with a softmax activation producing the predicted probability over the three constitutive law classes. The classifier is trained with a categorical cross-entropy loss.

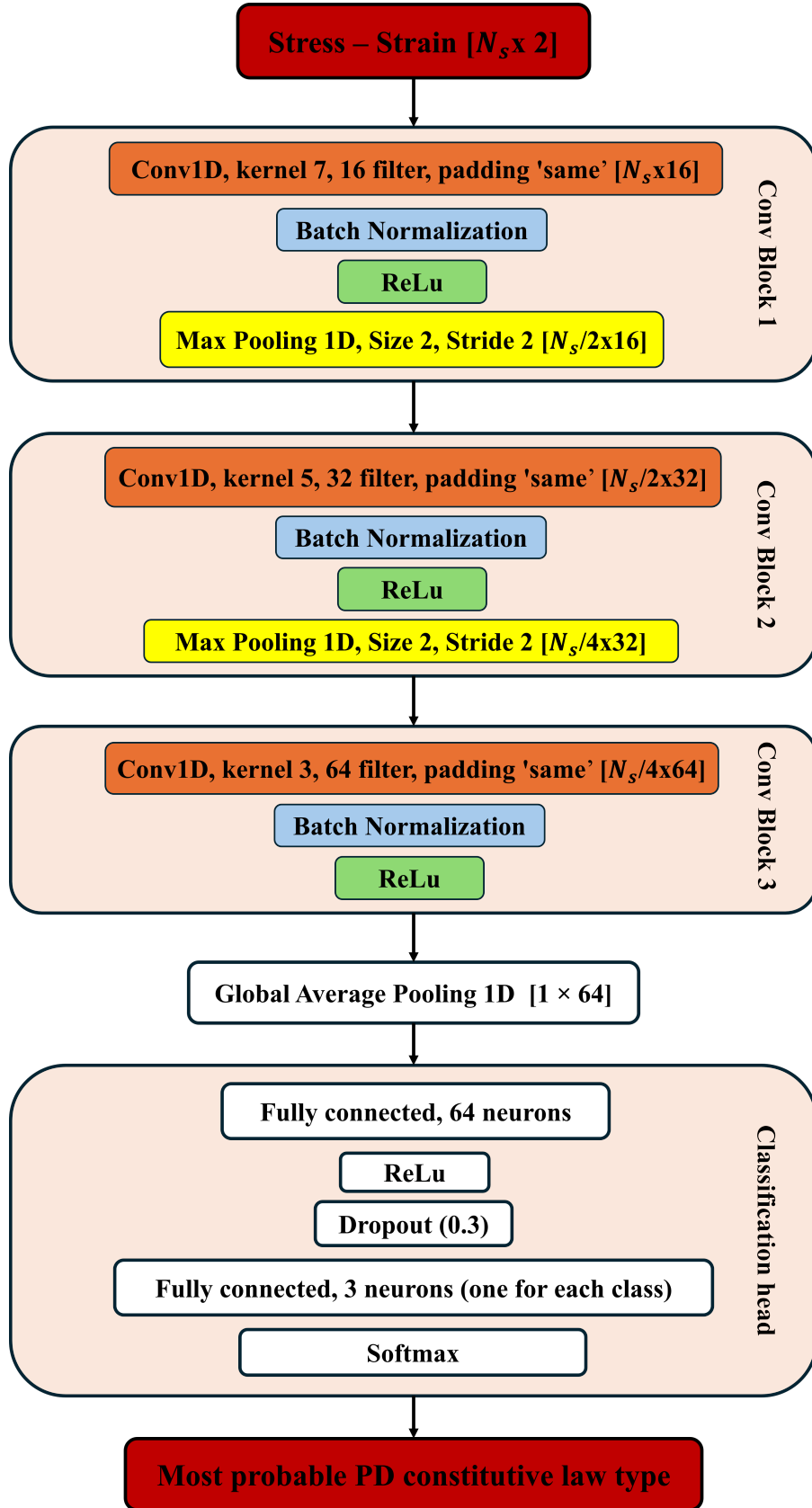


Figure 6.2: Layer-by-layer architecture of the classifier branch (here N_{samp} is shortened in N_s).

6.2 Regressor branches

Before to enter into detail of the specific architecture adopted for the regressor, it is important to highlight how the per-sample normalization adopted eliminates the absolute scale of the input curve. Nevertheless, the network maintains the capacity to infer s_0 and s_c , since the normalized curve shape still retains statistical correlations with these parameters, enabling the network to infer their values through relative geometric features, that can be encoded, e.g. in the relative position of the peak with respect to the beginning and tail of the curve, or in the characteristic normalized slope of the softening branch. In this sense, global average pooling adopted for the classifier, which collapses the sequence to a single value per channel to capture the overall curve shape, is not adapted for the regressor branches, in which the spatial distribution of feature maps along the strain axis must be preserved. Because s_0 and s_c correspond to specific transition points, positional details are critical and cannot be discarded. Consequently, the global average pooling layer used in the classifier is replaced in the regressor architecture by a flattening layer, which group together all the 1D dimensional vector features sparse among different filters, in a unique one containing a series of compressed features information. To avoid memory and overfitting issues that would arise from flattening full-length feature maps into an excessively wide fully-connected input, the spatial reduction strategy is also modified with respect to the classifier. The regressor branches employ a pooling scheme with increasing stride values ($S = 4$ in the first two blocks, $S = 8$ in the third). This progressive downsampling significantly compresses the sequence length prior to flattening, keeping the number of trainable parameters in the regression head manageable for the limited size of each law-specific dataset.

To handle the larger input size resulting from flattening, the downstream regression head is designed with a deeper fully-connected architecture that progressively compresses the spatial information toward the output. Specifically, the flattened vector is processed through three dense layers of decreasing capacity ($N/4$, $N/16$, and $N/32$ units) each followed by a ReLU activation and with dropout applied to the first two to mitigate overfitting. This structure is

followed by a final fully-connected layer with two linear units predicting s_0 and s_c . Since s_0 and s_c differ in physical scale, the regression targets are normalized on a per-parameter basis (zero mean, unit standard deviation, computed strictly on each training fold) prior to training,

$$y_{\text{norm}} = \frac{y - \mu_y}{\sigma_y}, \quad (118)$$

where μ_y and σ_y are the sample mean and standard deviation of the parameter, respectively, computed strictly on the training fold. This normalization prevents the loss from being dominated by the parameter with the larger absolute magnitude i.e. s_c . The regression loss adopted is the mean squared error, computed jointly over both normalized outputs.

Each of the three regressor branches is implemented as a separate, stand-alone network and trained independently on the subset of data corresponding to its specific constitutive law, rather than sharing a unified convolutional backbone with the classifier or with one of the other two regressor. While a multi-task framework, with a shared feature extractor and specialized fully-connected heads, is a common Deep Learning design [52], it was deliberately avoided in this context for the following reason: the geometric interpretation of stress-strain curves fundamentally changes depending on the underlying damage model: the bilinear law exhibits a sharp derivative discontinuity at peak stress followed by a linear decay, the trilinear law introduces an intermediate slope change, while the exponential model exhibits a non-linear decay. Forcing a single shared convolutional backbone across all three regimes would force the initial 1D kernels to adopt a compromised representation, causing one law to degrade feature quality for another during optimization. Decoupling the regressor into independent branches avoids this interference, allowing each network’s convolutional kernels to fully specialize in capturing the distinct spatial signatures of its specific softening behavior.

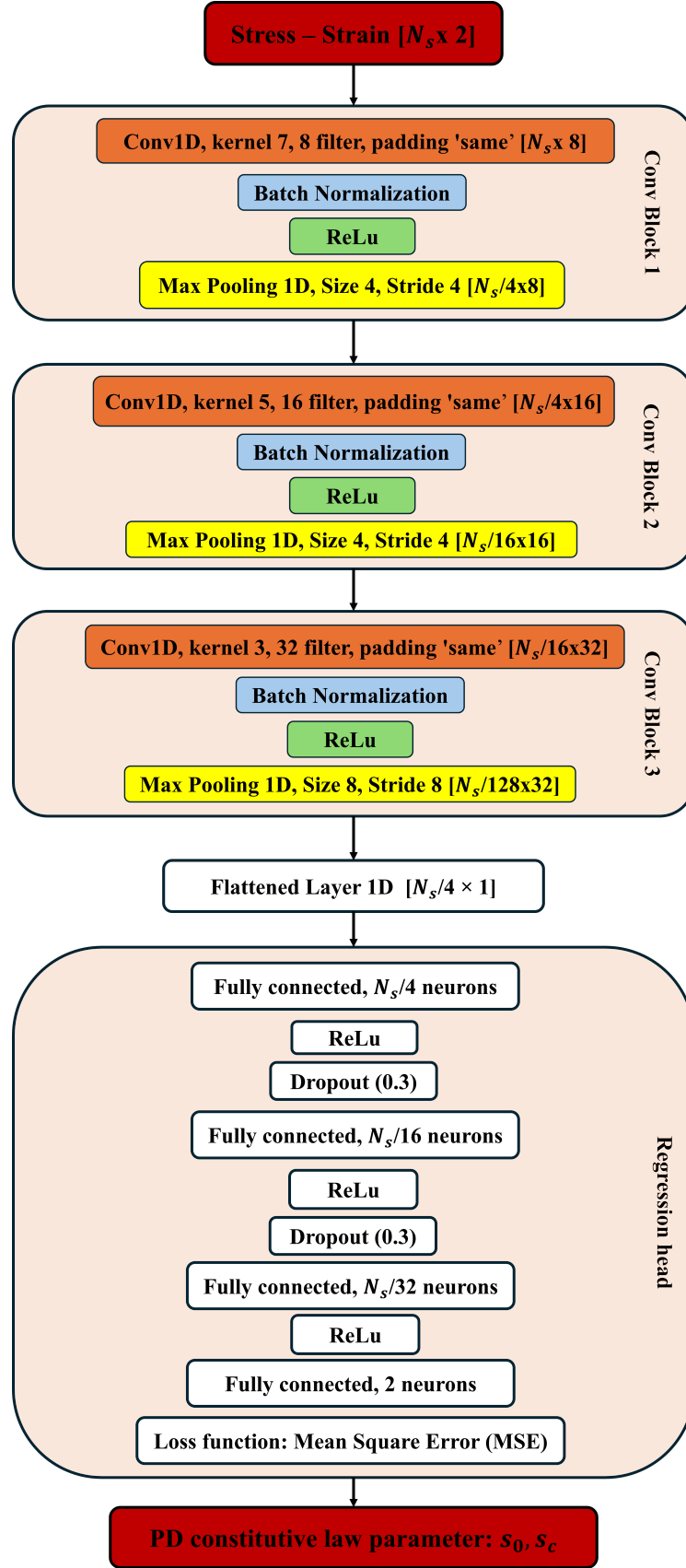


Figure 6.3: Layer-by-layer architecture of the law-specific regressor branches, each producing the bond-level parameters s_0, s_c (here $N_{s_{\text{samp}}}$ is shortened in N_s).

6.3 Training setup

Hyperparameter	Classifier	Regressors
Optimizer	Adam	Adam
Initial learning rate	1×10^{-3}	3×10^{-4}
Learning rate schedule	Piecewise	Piecewise
Learning rate drop factor	0.5	0.5
Learning rate drop period	10 epochs	10 epochs
Mini-batch size	32	16
Maximum epochs	100	200
Validation frequency	30 iterations	15 iterations
Early stopping patience	10 checks	30 checks
Data shuffling	Every epoch	Every epoch
Dropout	0.3	0.3
L_2 weight decay	/	1×10^{-4}

Table 6.1: Training hyperparameters adopted for the classifier and regression branches.

The classifier and the three regressor branches are trained independently of one another, each using the training, validation, and test split described in Section 5.5, applied to the full dataset for the classifier and, independently, to each law-specific subset for the corresponding regressor. All branches are optimized with *Adam*, using a piecewise learning rate schedule that halves the learning rate at fixed intervals, and are regularized through dropout (and also L_2 weight decay for regressor). Training is stopped early once the validation loss fails to improve for a fixed number of consecutive validation checks, rather than running for a fixed number of epochs, since preliminary training runs showed both the classifier and the regressors to converge well before the maximum epoch budget was reached, making early stopping an effective way to avoid unnecessary training time without affecting the final performance. Table 6.1 summarizes the training hyperparameters adopted, kept constant across all branches.

Chapter 7

Results discussion

7.1 Classifier performance

The classifier branch described in Section 6.1 was trained on the full dataset, following the stratified 70/15/15 train/validation/test split. Consistent with the rapid convergence anticipated in Section 6.3, training stopped well before the maximum epoch budget (25 over 100 epochs) was reached, with early stopping activated once the validation loss failed to improve for the prescribed number of consecutive checks (10 for the classifier). Despite this early termination, validation accuracy had already reached a value approaching 100% within the first few initial epochs as illustrated in Figure 7.1, with the remaining training time contributing no further measurable performance gains.

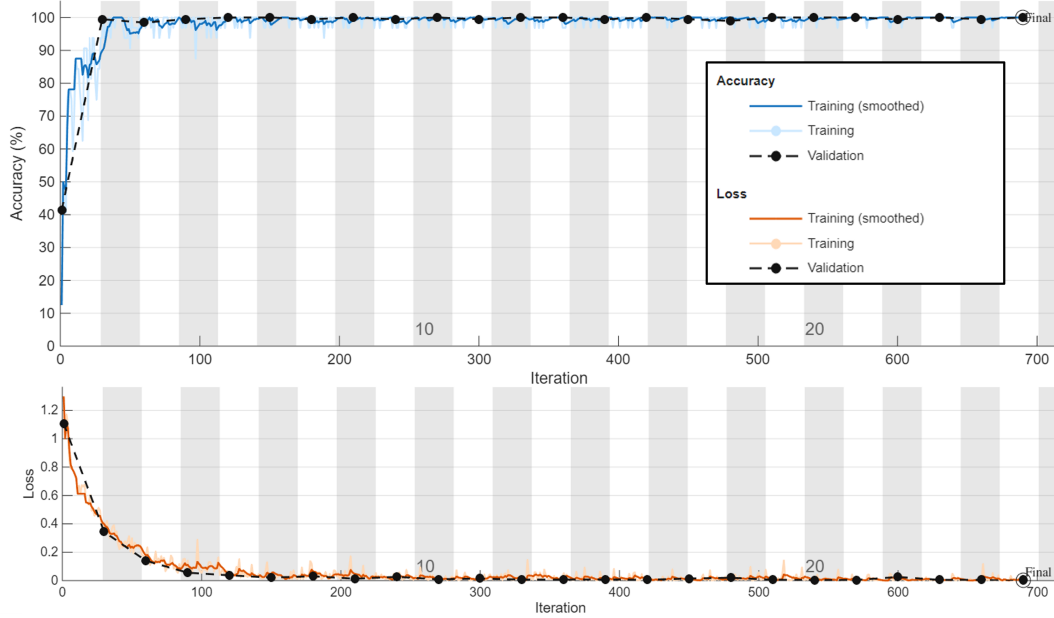


Figure 7.1: Classifier training progress.

Test-set accuracy. Figure 7.2 reports the confusion matrix obtained on the test set (236 samples, distributed across the three classes). A single sample is misclassified, corresponding to a test accuracy of 99.56%: the bilinear and exponential classes are both perfectly classified, while one trilinear sample is predicted as exponential. No sample belonging to the bilinear class is ever confused with either of the other two laws, consistent with its markedly different softening branch; the single error occurs between the exponential and trilinear laws, whose post-peak responses are considerably closer in shape.

True Class	bilinear	78		
	exponential		81	
	trilinear		1	76
		bilinear	exponential	trilinear
		Predicted Class		

Figure 7.2: Confusion matrix of the classifier branch on the test set.

Feature-space structure. As described in Section 6, the convolutional branch transforms the input stress–strain curve into a compact 64-dimensional representation through the sequence of convolutional blocks and the final global average pooling layer. To visualize this learned representation, PCA is applied to the 64-dimensional feature vectors extracted from the test set in order to further characterize the learned representation and better understand if a clear separation between the three model in terms of their shape is found.

Results from PCA shows that cumulative explained variance grows extremely rapidly with the number of components: the first component alone accounts for 75% of the total variance, and the first two together for 96.8%, with the curve

essentially saturating beyond roughly four to five components (Figure 7.3). This indicates that, although nominally 64-dimensional, the representation learned by the network is effectively very low-dimensional, consistent with the fact that only three qualitatively distinct classes need to be separated.

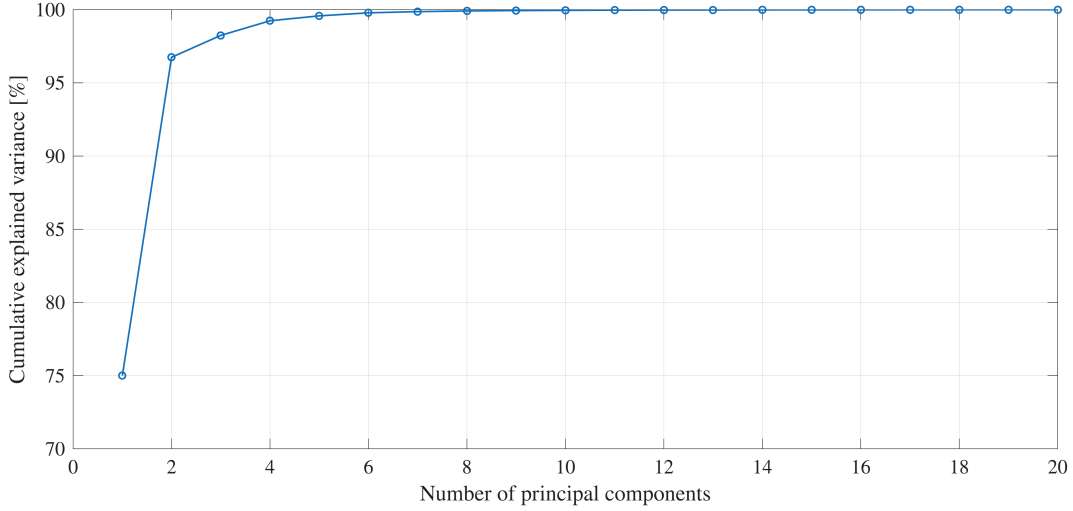


Figure 7.3: Cumulative explained variance of the CNN feature space as a function of the number of principal components.

Projecting the test-set feature vectors onto the first two principal components (Figure 7.4) reveals three well-separated, compact clusters, each with its own centroid and no overlap among them. In particular, it is possible to highlight how the bilinear model is strongly separated from the other two based on the first principal component, while the second one allows to distinguish between the trilinear and exponential model. The single misclassified sample highlighted in the confusion matrix lies almost exactly midway between the exponential and trilinear centroids, confirming, at the feature-space level, that the classifier’s single error is not attributable to a poorly learned representation, but corresponds to an ambivalent input whose extracted features straddle the boundary between two classes.

Figure 7.5 compares the misclassified curve directly against representative, correctly classified samples from both constitutive laws, for sets of close material parameters. As observed in the plot, the misclassified sample closely follows

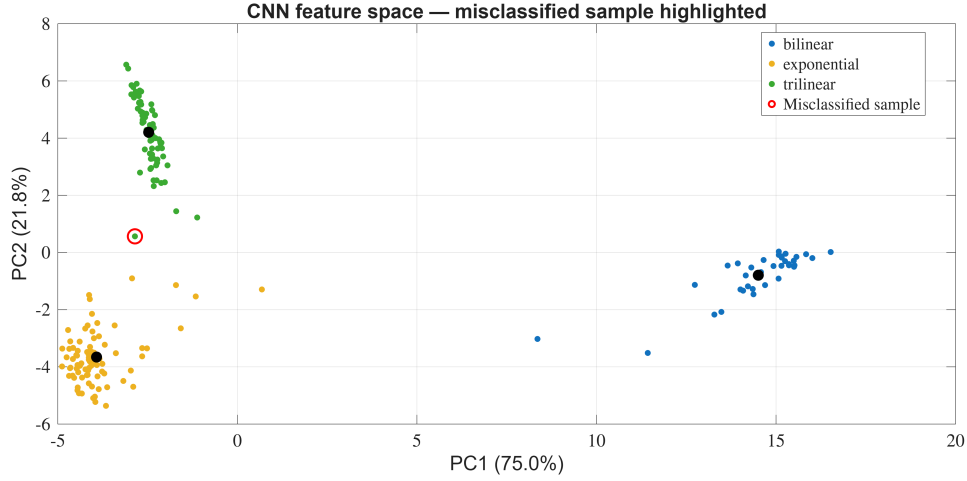


Figure 7.4: Test-set samples projected onto the first two principal components of the CNN feature space, with cluster centroids and the misclassified sample highlighted.

the standard trilinear behavior up to the peak stress and along the initial post-peak drop. However, instead of exhibiting the expected single knee point, the post-peak softening region exhibits two distinct slope discontinuities. This anomalous stiffness loss and multi-slope decay can be attributed to local instabilities of the equilibrium solution, which occur at the first knee point and have altered the overall material response.

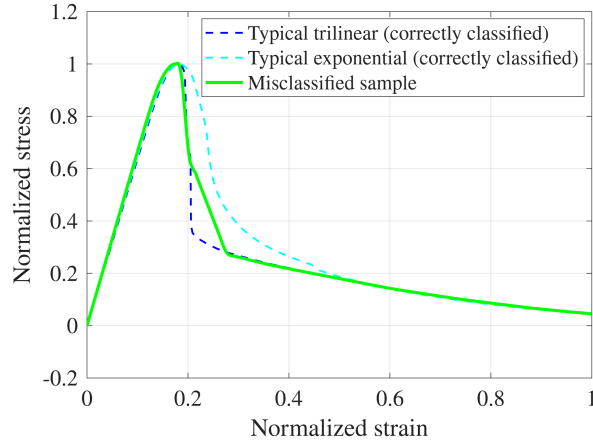


Figure 7.5: Misclassified trilinear sample compared against a typical, correctly classified curve of both the trilinear and the exponential law.

The combination of the smoothed transition points and the altered softening rate causes the 1D convolutional network to assign a predicted probability of 51.51% to the exponential model versus 44.81% to the true trilinear class (with a residual 3.69% assigned to the bilinear model). Rather than a misclassification based on a wrong learned pattern, these predicted probabilities demonstrate that the network operated directly on the decision boundary, capturing a case of genuine ambiguity between the two classes.

It should nonetheless be kept in mind, when interpreting this near-perfect accuracy, that both the training data and the three candidate constitutive laws originate from the same peridynamic model and solver: part of the classifier's performance may therefore reflect the internal consistency of the generating simulator rather than a fully general capability to distinguish this softening behavior when an experimental curve is provided. This point is revisited in Section 7.3.

7.2 Regressors performance

Each of the three regressor branches was trained independently on the specific subset associated to its law, following the same train/validation/test split already enunciated. Performances are reported separately for each branch and for each of the predicted parameters, s_0 and s_c .

On the trilinear test subset, the regressor achieves a mean absolute percentage error (MAPE) of 8.65% for s_0 and 8.83% for s_c . Since the size of each law-specific subset is limited, the sensitivity of this estimate to the particular train/test split was further assessed via 5-fold cross-validation: the dataset is divided into five subsets (folds) and the model is trained five times, each time using four folds for training and the remaining fold for validation, so that every sample is used once for validation. The resulting MAPE remains stable across folds, ranging between approximately 8% and 10% for s_0 and between 8.5% and 10.5% for s_c (Figure 7.6), with neither parameter showing a fold with anomalously large error. This indicates that the single-split estimate above is representative of the regressor's actual performance rather than an artifact of

a favorable test partition.

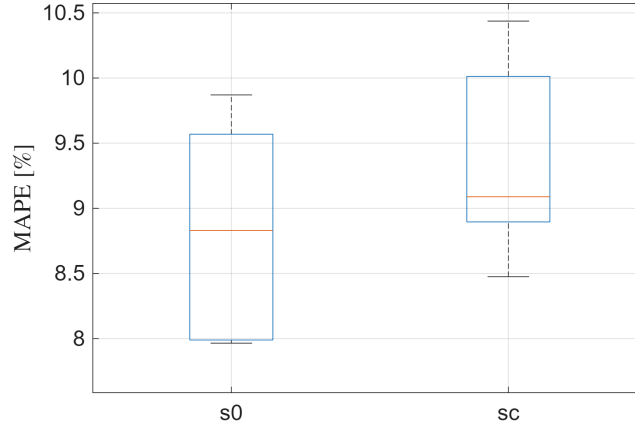


Figure 7.6: 5-fold cross-validation distribution of the MAPE for s_0 and s_c , trilinear regressor.

Figure 7.7 shows the corresponding *parity plots* (predicted vs. true) for s_0 and s_c on the test subset: predictions are scattered symmetrically around the *perfectly identity line* across the full sampled range, without showing any specific bias.

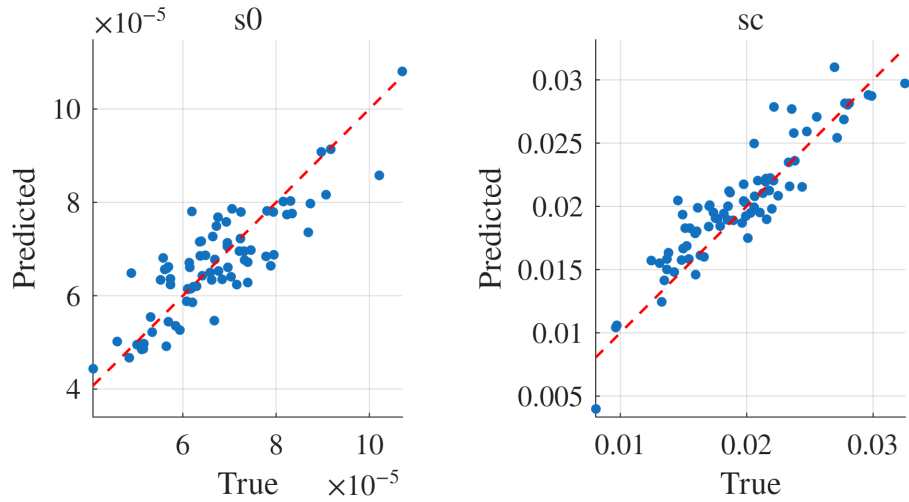


Figure 7.7: Parity plots for s_0 and s_c , trilinear regressor test subset.

The same procedure — single-split evaluation followed by 5-fold cross-validation — is applied to the bilinear and exponential regressors, trained independently on their respective law-specific subsets. Results are reported in Figures 7.8–7.11 and summarized in Table 7.1. Both regressors exhibit the same qualitative behavior already observed for the trilinear law: a MAPE in the same order of magnitude for s_0 and s_c , stable across cross-validation folds, and parity plots scattered symmetrically around the identity line with no evident bias toward either end of the sampled range.

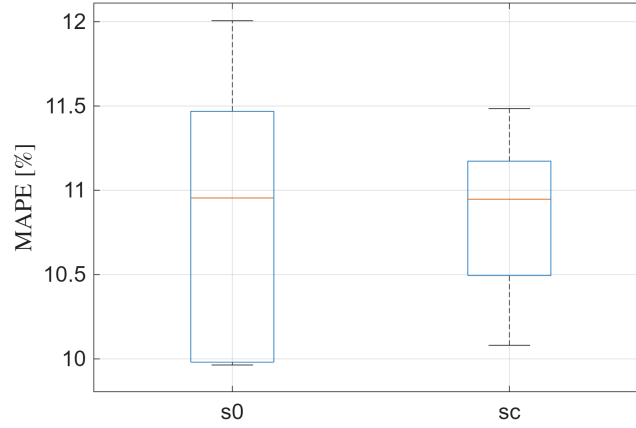


Figure 7.8: 5-fold cross-validation distribution of the MAPE for s_0 and s_c , bilinear regressor.

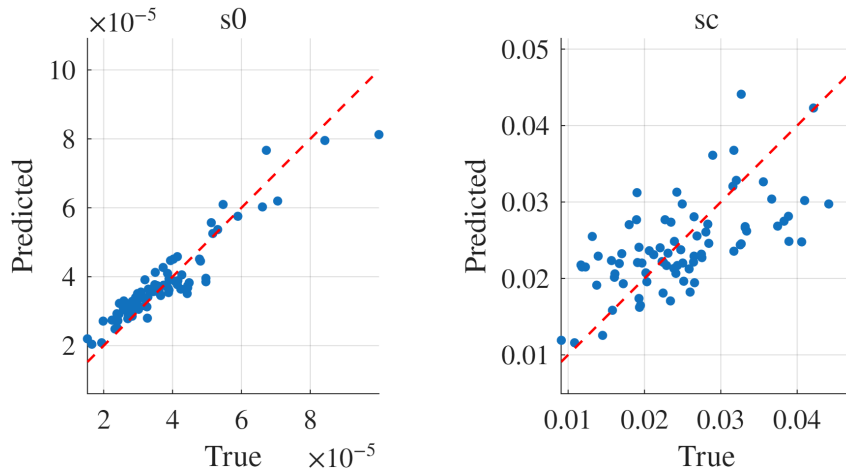


Figure 7.9: Parity plots for s_0 and s_c , bilinear regressor test subset.

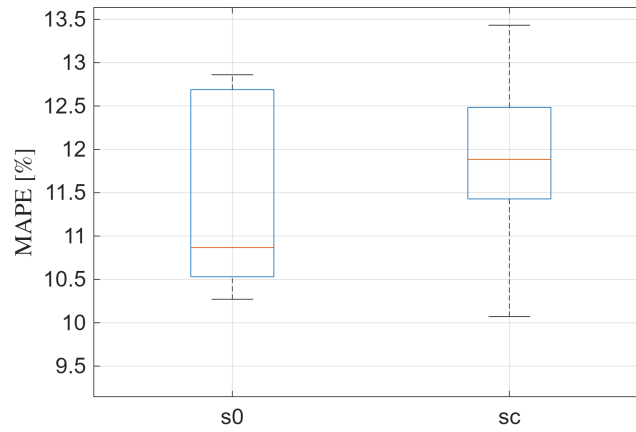


Figure 7.10: 5-fold cross-validation distribution of the MAPE for s_0 and s_c , exponential regressor.

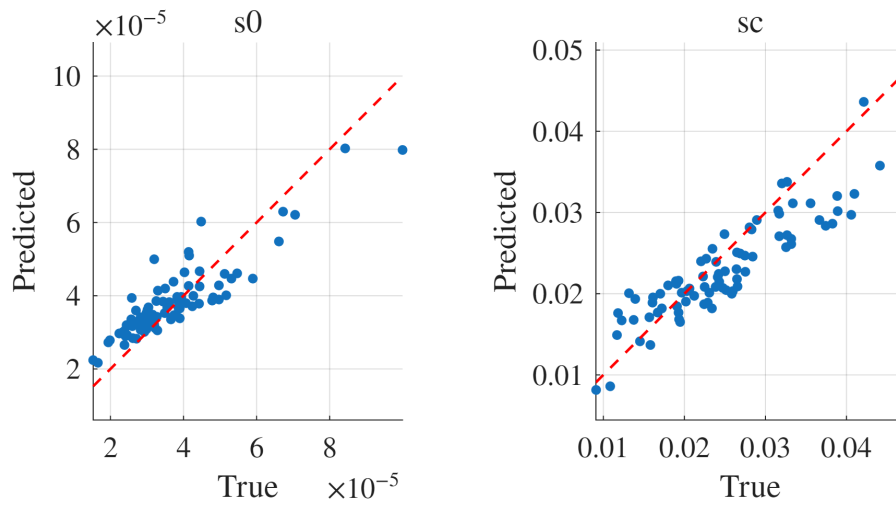


Figure 7.11: Parity plots for s_0 and s_c , exponential regressor test subset.

Law	Parameter	MAPE (single split)	MAPE (5-fold mean)	MAPE (5-fold range)
Bilinear	s_0	11.76 %	10.9 %	[10.0, 12.0] %
Bilinear	s_c	11.20 %	10.9 %	[10.1, 11.5] %
Exponential	s_0	10.91 %	10.8 %	[10.3, 12.8] %
Exponential	s_c	11.24 %	11.9 %	[10.1, 13.4] %
Trilinear	s_0	8.65%	8.8 %	[8.0, 9.8] %
Trilinear	s_c	8.83%	9.1 %	[8.5, 10.4] %

Table 7.1: Summary of regressor performance (MAPE) for s_0 and s_c , across the three constitutive laws.

7.3 End-to-end validation of the identification pipeline

While Sections 7.1 and 7.2 assess the classifier and the regressors performances separately against ground-truth labels and parameters, which are synthetic as already highlighted, neither metric directly verifies that the complete identification pipeline reproduces the *physical response* of the material. A curve can, in principle, be assigned to the most appropriate class and yield a small parameter error in a normalized sense, while still producing a reconstructed stress–strain response that deviates non-negligibly from the original curve. To close this gap, the full pipeline was validated end-to-end.

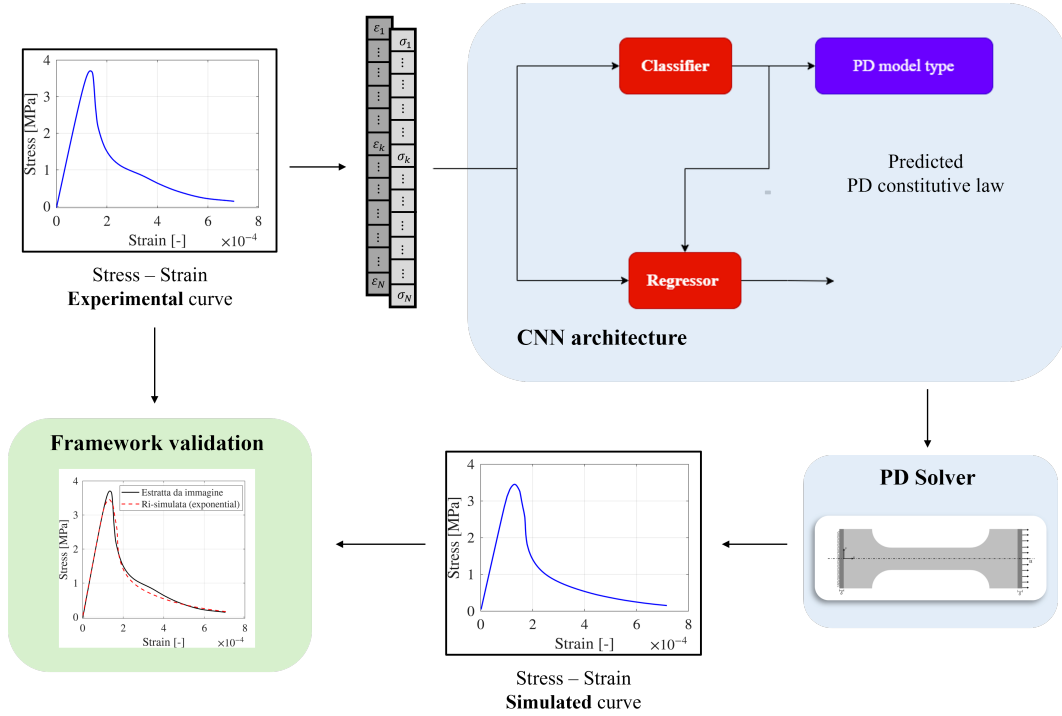


Figure 7.12: End-to-end validation procedure.

Procedure. Given a stress–strain curve, the following steps are performed as illustrated in Figure 7.12:

- (i) the extracted curve is resampled and normalized following the same pre-processing pipeline applied during training (Section 5.5);
- (ii) the pre-processed curve is passed to the trained classifier, which selects the most probable constitutive law;
- (iii) the curve is passed to the regressor branch corresponding to the predicted law, which outputs the estimated bond-level parameters $\hat{s}_0$, $\hat{s}_c$, with the corresponding micromodulus $\hat{c}$ obtained from Eq. (8);
- (iv) the estimated parameters are used to run a new PD simulation, using the same arc-length solver and specimen configuration described in Chapter 4 and Section 5.2, under the predicted PD constitutive law;
- (v) the resulting *reconstructed curve* is compared against the *reference curve* used as input and comparison metrics are extracted.

Comparison metrics. Both curves are interpolated onto a common strain grid, restricted to the range shared by the two, and compared using the root-mean-square error (RMSE), its normalized counterpart (NRMSE, obtained by scaling the RMSE with the tensile strength f_t of the reference curve) and the coefficient of determination (R^2) between the reconstructed and original stress values.

Case study. The stress-strain curve used for this end-to-end test, shown in Figure 7.13, is a digitized version of the response of a notched concrete specimen under direct tension reported by Gopalaratnam and Shah [53], whose experimental campaign characterized the complete, post-peak softening behavior of plain concrete in direct tension. The curve was digitized from the printed figure via a dedicated pixel-color extraction routine, using the axis calibration marks of the original plot to convert pixel coordinates into physical strain and stress pairs of values.

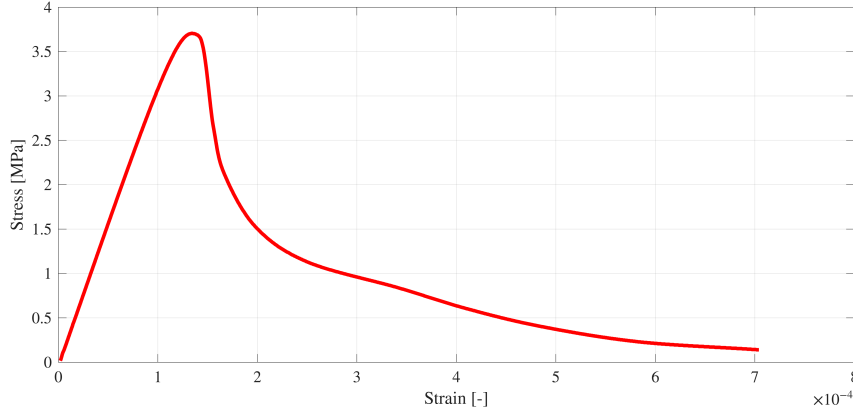


Figure 7.13: Digitized experimental stress-strain curve of a notched concrete specimen under direct tension, extracted from Figure 7(a) of [53].

This curve, after resampling and normalization, is passed through the classifier and regressor branches of the identification pipeline (Figure 7.14). The exponential law is predicted with 100% confidence, and the corresponding regressor outputs the estimated critical stretches $\hat{s}_0 = 6.7612 \times 10^{-5}$ and $\hat{s}_c = 1.3195 \times 10^{-2}$, both dimensionless. The micromodulus $\hat{c}$ is instead obtained directly from the elastic modulus estimated by averaging the secant stiffness of the reference curve over its initial linear region, yielding $\hat{c} = 2.1042 \times 10^{21} \text{ N/m}^6$.

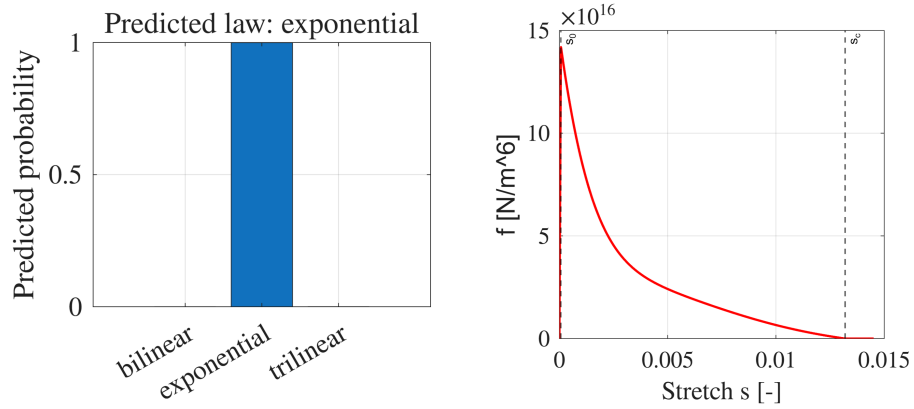


Figure 7.14: Predicted constitutive law and associated probability (on the left), together with the PD bond-force law reconstructed from the estimated parameters (on the right).

Figure 7.15 shows the outcome of the resulting re-simulation. Overall, the reconstructed response tracks the original curve closely over both the elastic branch and the softening tail, capturing the material behavior with high fidelity as reflected by a coefficient of determination of $R^2 = 0.981$ and an overall RMSE = 0.133 MPa (corresponding to an NRMSE = 3.78%).

The residual error varies along the curve in a non-uniform way: far from the peak, the reconstructed and the reference profiles are in near-perfect agreement, while the residual deviation is heavily concentrated in the immediate post-peak region. This higher localized discrepancy can be justified by the fact that this region represents the most challenging phase to model numerically, as it coincides with localized propagation of the initialized micro-cracks, rapid load-bearing capacity degradation, and continuous updates of the equilibrium configuration. The localized error peak in this transition branch reflects the extreme sensitivity of the post-peak softening behavior to minor variations in parameter identification, rather than a qualitative failure to reconstruct the global curve shape.

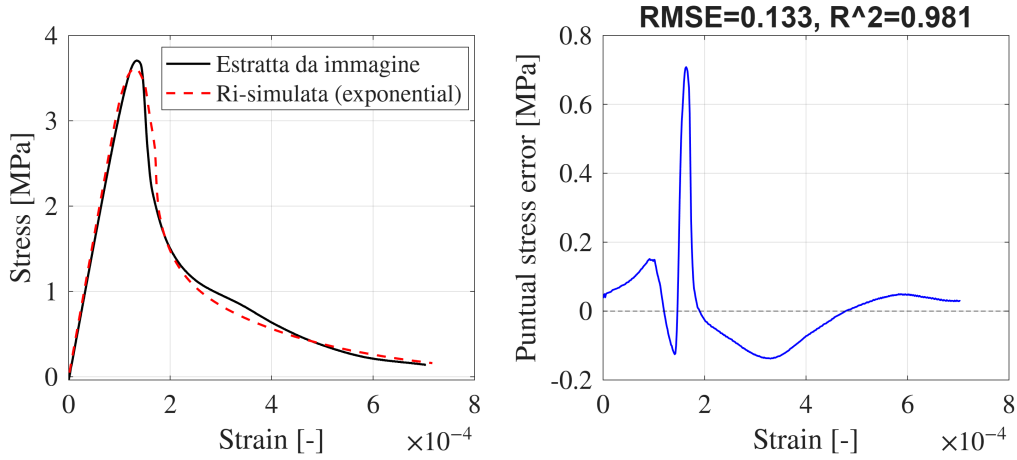


Figure 7.15: Comparison between the reference curve and the curve reconstructed from the predicted bond-level parameters (on the left), together with the point-wise error along the strain axis (on the right).

Chapter 8

Conclusions

The aim of this work was to develop a data-driven pipeline for the identification of peridynamic constitutive laws from macroscopic stress–strain responses. To this aim, a displacement-controlled arc-length solver was fully developed on MATLAB to trace the complete equilibrium path of quasi-static tensile tests, including the post-peak regimes, and was successively used to generate a synthetic dataset of 1580 stress–strain curves under three reference peridynamic softening laws, namely bilinear, exponential, and trilinear. This dataset was used to train a hierarchical convolutional neural network, consisting of a classification stage followed by three law-specific regression branches. The classifier achieved a test accuracy of 99.56% over 229 samples, indicating that the adopted design choices yield effective performance. This finding is further supported by the interpretation provided by the principal component analysis of the CNN-learned feature space, which revealed a clear separation among the three constitutive laws. The regression branches estimated the critical stretches s_0 and s_c with mean MAPE values, evaluated through 5-fold cross-validation, of respectively 11.76–11.20% for the bilinear law, 10.91–11.24% for the exponential law, and 8.65–8.83% for the trilinear law. While these errors are not negligible, they are consistent with the limited size of each law-specific subset, and a larger, more representative dataset could further improve regression performance.

The complete identification pipeline was finally validated on experimental stress–strain curve drawn from the literature. The reconstructed response showed good agreement with the reference curve, with $R^2 = 0.981$ and an NRMSE of 3.78%, providing an end-to-end validation of the pipeline: from an input stress–strain response, through the identification of the constitutive law and its parameters, to their use in a new peridynamic simulation reproducing

the original curve.

Beyond the specific numerical results, the main contribution of this thesis lies in the development of a modular identification pipeline rather than in the predictive accuracy for the material class considered. Quasi-brittle materials and the three softening laws were adopted as a case study, while the underlying peridynamic solver is independent of the constitutive law. Replacing the constitutive law allows the same framework to generate the corresponding macroscopic responses and training data for different material classes without modifying the solver or overall identification architecture. The proposed methodology is therefore not restricted to quasi-brittle materials, but is extensible to any material class for which a suitable peridynamic constitutive law can be formulated.

On the other hand, the framework developed in this thesis is subject to some limitations. First, the methodology is entirely based on the Bond-Based formulation of Peridynamics, which constrains the Poisson's ratio to a fixed value and therefore limits the range of materials that can be simulated. Second, although the identification pipeline was validated against an experimental curve, the model was trained and cross-validated exclusively on synthetic data generated by the same peridynamic solver. Future developments could therefore focus on extending the framework to a State-Based peridynamic formulation and testing the identified constitutive laws on problems with setups beyond the simple tensile test, enabling systematic validation against experimental and numerical benchmarks.

As a further future development, an approach inspired by autoencoder architectures could be explored, in which the network directly maps the input stress-strain response to the corresponding one-dimensional peridynamic constitutive curve. In this formulation, the network would no longer select among predefined laws, but would directly provide the constitutive relation required by the peridynamic solver. This would provide a natural extension of the present framework towards a more general, fully data-driven characterization of material behavior.

Bibliography

- [1] Arne Hillerborg, Mats Mod  er, and P-E Petersson. “Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements”. In: *Cement and concrete research* 6.6 (1976), pp. 773–781.
- [2] Ted Belytschko and Tom Black. “Elastic crack growth in finite elements with minimal remeshing”. In: *International journal for numerical methods in engineering* 45.5 (1999), pp. 601–620.
- [3] Blaise Bourdin, Gilles A Francfort, and Jean-Jacques Marigo. “Numerical experiments in revisited brittle fracture”. In: *Journal of the Mechanics and Physics of Solids* 48.4 (2000), pp. 797–826.
- [4] Stewart A Silling. “Reformulation of elasticity theory for discontinuities and long-range forces”. In: *Journal of the Mechanics and Physics of Solids* 48.1 (2000), pp. 175–209.
- [5] Francesco Scabbia. “An improved peridynamic framework for the accurate solution of multi-physics problems”. PhD thesis. University of Padova, 2024.
- [6] Stewart A Silling and Richard B Lehoucq. “Convergence of peridynamics to classical elasticity theory”. In: *Journal of Elasticity* 93.1 (2008), pp. 13–37.
- [7] Jeremy Trageser and Pablo Seleson. “Bond-based peridynamics: a tale of two Poisson’s ratios”. In: *Journal of Peridynamics and Nonlocal Modeling* 2.3 (2020), pp. 278–288.
- [8] Stewart A Silling et al. “Peridynamic states and constitutive modeling”. In: *Journal of elasticity* 88.2 (2007), pp. 151–184.
- [9] Stewart A Silling and Ebrahim Askari. “A meshfree method based on the peridynamic model of solid mechanics”. In: *Computers & structures* 83.17-18 (2005), pp. 1526–1535.

- [10] Pablo Seleson and Michael Parks. “On the role of the influence function in the peridynamic theory”. In: *International Journal for Multiscale Computational Engineering* 9.6 (2011).
- [11] Stewart A Silling. “Stability of peridynamic correspondence material models and their particle discretizations”. In: *Computer Methods in Applied Mechanics and Engineering* 322 (2017), pp. 42–57.
- [12] Florin Bobaru et al. *Handbook of peridynamic modeling*. CRC press, 2016.
- [13] Sina Niazi, Ziguang Chen, and Florin Bobaru. “Crack nucleation in brittle and quasi-brittle materials: A peridynamic analysis”. In: *Theoretical and Applied Fracture Mechanics* 112 (2021), p. 102855.
- [14] Pouria Sheikhbahaei, Farshid Mossaiby, and Arman Shojaei. “An efficient peridynamic framework based on the arc-length method for fracture modeling of brittle and quasi-brittle problems with snapping instabilities”. In: *Computers & Mathematics with Applications* 136 (2023), pp. 165–190.
- [15] Hans W Reinhardt, Hans AW Cornelissen, and Dirk A Hordijk. “Tensile tests and failure analysis of concrete”. In: *Journal of structural engineering* 112.11 (1986), pp. 2462–2477.
- [16] Pablo Seleson. “Improved one-point quadrature algorithms for two-dimensional peridynamic models based on analytical calculations”. In: *Computer Methods in Applied Mechanics and Engineering* 282 (2014), pp. 184–217.
- [17] QV Le and Florin Bobaru. “Surface corrections for peridynamic models in elasticity and fracture”. In: *Computational Mechanics* 61.4 (2018), pp. 499–518.
- [18] Erdogan Madenci and Erkan Oterkus. “Peridynamic theory”. In: *Peridynamic theory and its applications*. Springer, 2013, pp. 19–43.
- [19] Florin Bobaru et al. “Convergence, adaptive refinement, and scaling in 1D peridynamics”. In: *International Journal for Numerical Methods in Engineering* 77.6 (2009), pp. 852–877.

- [20] Ryszard S Michalski, Jaime Guillermo Carbonell, and Tom M Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.
- [21] *Lecture Notes – MIT 6.390 Spring 2023*. Course lecture notes. 2023.
- [22] Maria Vakalopoulou et al. “Deep learning: basics and convolutional neural networks (CNNs)”. In: *Machine learning for brain disorders (2023)*, pp. 77–115.
- [23] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Networks* 2.5 (1989), pp. 359–366.
- [24] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. “Learning long-term dependencies with gradient descent is difficult”. In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.
- [25] Lu Lu et al. “Dying relu and initialization: Theory and numerical examples”. In: *arXiv preprint arXiv:1903.06733* (2019).
- [26] John S Bridle. “Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition”. In: *Neurocomputing: Algorithms, architectures and applications*. Springer, 1990, pp. 227–236.
- [27] Wei Xu, Wen Chen, and Yingjie Liang. “Feasibility study on the least square method for fitting non-Gaussian noise data”. In: *Physica A: Statistical Mechanics and its Applications* 492 (2018), pp. 1917–1930.
- [28] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.
- [29] Michael A Nielsen. “Neural networks and deep learning”. In: (2015).
- [30] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).
- [31] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

- [32] Taco Cohen and Max Welling. “Group equivariant convolutional networks”. In: *International conference on machine learning*. PMLR. 2016, pp. 2990–2999.
- [33] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA: MIT Press, 2016.
- [34] Vincent Dumoulin and Francesco Visin. “A guide to convolution arithmetic for deep learning”. In: *arXiv preprint arXiv:1603.07285* (2016).
- [35] M. A. J. van Gils. “Quasi-brittle fracture of ceramics”. Chemical Engineering and Chemistry. PhD thesis. Eindhoven, The Netherlands: Technische Universiteit Eindhoven, 1997.
- [36] Eduard Riks. “An incremental approach to the solution of snapping and buckling problems”. In: *International journal of solids and structures* 15.7 (1979), pp. 529–551.
- [37] Michael A Crisfield. “A fast incremental/iterative solution procedure that handles “snap-through””. In: *Computational methods in nonlinear structural and solid mechanics*. Elsevier, 1981, pp. 55–62.
- [38] Jean-Louis Batoz and Gouri Dhatt. “Incremental displacement algorithms for nonlinear problems”. In: *International Journal for Numerical Methods in Engineering* 14.8 (1979), pp. 1262–1267.
- [39] Mario Fafard and Bruno Massicotte. “Geometrical interpretation of the arc-length method”. In: *Computers & structures* 46.4 (1993), pp. 603–615.
- [40] EA de Souza Neto and YT Feng. “On the determination of the path direction for arc-length methods in the presence of bifurcations and snap-backs”. In: *Computer methods in applied mechanics and engineering* 179.1-2 (1999), pp. 81–89.
- [41] Tao Ni et al. “Static solution of crack propagation problems in Peridynamics”. In: *Computer Methods in Applied Mechanics and Engineering* 346 (2019), pp. 126–151.
- [42] Zdeněk P Bažant. “Concrete fracture models: testing and practice”. In: *Engineering fracture mechanics* 69.2 (2002), pp. 165–205.

- [43] Dong Yang et al. “An improved ordinary state-based peridynamic model for cohesive crack growth in quasi-brittle materials”. In: *International Journal of Mechanical Sciences* 153 (2019), pp. 402–415.
- [44] Dirk Arend Hordijk. “Local approach to fatigue of concrete.” In: (1993).
- [45] Duy H Nguyen, Vinh TN Dao, and Pietro Lura. “Tensile properties of concrete at very early ages”. In: *Construction and Building Materials* 134 (2017), pp. 563–573.
- [46] Dongya Ren and Lambert Houben. “Determination of fracture energy of early age concrete through a uniaxial tensile test on an un-notched specimen”. In: *Materials* 13.3 (2020), p. 496.
- [47] Zongjin Li, SM Kulkarni, and SP Shah. “New test method for obtaining softening response of unnotched concrete specimen under uniaxial tension”. In: *Experimental Mechanics* 33.3 (1993), pp. 181–188.
- [48] MatWeb. *MatWeb: Material Property Data*. <https://www.matweb.com/>. Online material property database. 2026.
- [49] Daniele Dipasquale et al. “Dependence of crack paths on the orientation of regular 2D peridynamic grids”. In: *Engineering Fracture Mechanics* 160 (2016), pp. 248–263.
- [50] Pablo Seleson and David J Littlewood. “Convergence studies in meshfree peridynamic simulations”. In: *Computers & Mathematics with Applications* 71.11 (2016), pp. 2432–2448.
- [51] Francesco Scabbia, Mirco Zaccariotto, and Ugo Galvanetto. “A novel and effective way to impose boundary conditions and to mitigate the surface effect in state-based Peridynamics”. In: *International Journal for Numerical Methods in Engineering* 122.20 (2021), pp. 5773–5811.
- [52] Rich Caruana. “Multitask learning”. In: *Machine learning* 28.1 (1997), pp. 41–75.
- [53] VS Gopalaratnam and Surendra P Shah. “Softening response of plain concrete in direct tension”. In: *ACI journal* 82.3 (1985), pp. 310–323.

