FEDERAL UNIVERSITY OF MINAS GERAIS (UFMG)
SCHOOL OF ENGINEERING
DEPARTMENT OF STRUCTURAL ENGINEERING
GRADUATE PROGRAM IN STRUCTURAL ENGINEERING (PROPEES)

Álefe Freitas Figueiredo

THE CHALLENGES OF CONCRETE CONSTITUTIVE MODELING VIA ARTIFICIAL NEURAL NETWORKS

Belo Horizonte
2023

Álefe Freitas Figueiredo

THE CHALLENGES OF CONCRETE CONSTITUTIVE MODELING VIA ARTIFICIAL NEURAL NETWORKS

Final Version

Master's thesis submitted to the Graduate Program in Structural Engineering (PROPEES) of the School of Engineering at the FEDERAL UNIVERSITY OF MINAS GERAIS (UFMG), in partial fulfillment of the requirements for the master degree in Structural Engineering

Supervisor: Prof. Dr. Roque Luiz da Silva Pitangueira

Co-Supervisor: Msc. Saulo Silvestre de Castro

Belo Horizonte
2023

Figueiredo, Álefe Freitas.
F475c The challenges of concrete constitutive modeling via artificial neural networks [recurso eletrônico] / Álefe Freitas Figueiredo. - 2023.
1 recurso online (215 f. : il., color.) : pdf.

Orientador: Roque Luiz da Silva Pitangueira.
Coorientador: Saulo Silvestre de Castro.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 172-215.

Bibliografia: f. 161-171.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia de estruturas - Teses. 2. Redes neurais (Computação) - Teses. 3. Aprendizado de máquina - Teses. 4. Método dos elementos finitos - Teses. 5. Concreto - Teses. I. Pitangueira, Roque Luiz da Silva. II. Castro, Saulo Silvestre de. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.

CDU: 624(043)

Ficha catalográfica elaborada pela bibliotecária Ângela Cristina Silva CRB/6 2361
Biblioteca Prof. Mário Werneck, Escola de Engenharia da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ESTRUTURAS

PROPEEs UFMG

ATA DA DEFESA DE DISSERTAÇÃO DE MESTRADO EM ENGENHARIA DE ESTRUTURAS Nº: 398 DO ALUNO ÁLEFE FREITAS FIGUEIREDO

Às **14:00** horas do dia **26** do mês de **abril** de **2023**, reuniu-se em ambiente virtual, a Comissão Examinadora indicada pelo Colegiado do Programa em 26 de outubro de 2022, para julgar a defesa da Dissertação de Mestrado intitulada **"The Challenges of Concrete Constitutive Modelling via Artificial Neural Networks"**, cuja aprovação é um dos requisitos para a obtenção do Grau de MESTRE EM ENGENHARIA DE ESTRUTURAS na área de ESTRUTURAS.

Abrindo a sessão, o Presidente da Comissão, **Prof. Dr. Roque Luiz da Silva Pitangueira**, após dar a conhecer aos presentes o teor das Normas Regulamentares passou a palavra ao candidato para apresentação de seu trabalho. Seguiu-se a arguição pelos examinadores, com a respectiva defesa do candidato. Logo após, a Comissão se reuniu, sem a presença do candidato e do público, para julgamento e expedição do resultado final. Foram atribuídas as seguintes indicações:

Prof. Dr. Roque Luiz da Silva Pitangueira - DEES - UFMG (Orientador) - APROVADO
Prof. Dr. Lapo Gori - DEES - UFMG - APROVADO
Prof. Dr. Adriano Alonso Veloso - UFMG - APROVADO
Profa. Dra. Michele Cristina Resende Farage - UFJF - APROVADO

Pelas indicações acima, o candidato foi considerado APROVADO, conforme pareceres em anexo.

O resultado final foi comunicado publicamente ao candidato pelo Presidente da Comissão. Nada mais havendo a tratar, o Presidente encerrou a reunião e lavrou a presente ATA, que será assinada por todos os membros participantes da Comissão Examinadora.

Belo Horizonte, 26 de abril de 2023.

Observações:

1. A aprovação do candidato na defesa da Dissertação de Mestrado não significa que o mesmo tenha cumprido todos os requisitos necessários para obtenção do Grau de Mestre em Engenharia de Estruturas;
2. Este documento não terá validade sem a assinatura do Coordenador do Programa de Pós-Graduação.

Documento assinado eletronicamente por **Roque Luiz da Silva Pitangueira**, **Professor do Magistério Superior**, em 26/04/2023, às 16:51, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Lapo Gori**, **Professor do Magistério Superior**, em 27/04/2023, às 08:20, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Michèle Cristina Resende Farage**, **Usuária Externa**, em 27/04/2023, às 09:15, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Adriano Alonso Veloso**, **Professor do Magistério Superior**, em 04/05/2023, às 10:09, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Felicio Bruzzi Barros**, **Coordenador(a) de curso de pós-graduação**, em 14/06/2023, às 09:42, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2246560** e o código CRC **5278BF2E**.

Referência: Processo nº 23072.223854/2023-19 SEI nº 2246560

We can only see a short distance ahead, but we can see plenty there that needs to be done.
Alan Turing

À vó Deilda e à vô Sabuco (in memoriam)

Agradecimentos

À minha família, em especial ao meu pai, Airton, à minha mãe, Cirléia e ao meu irmão, Áthyla, por todo amor, apoio e incentivo.

À vó Deilda e à vô Sabuco, pelo exemplo de fé, força e persistência (em memória).

Ao professor Roque Pitangueira, pela orientação, confiança e paciência.

Ao Saulo de Castro, pela co-orientação, disponibilidade e suporte.

Aos meus amigos, pelos desabafos e pela presença.

Ao meus colegas de curso, pela colaboração e parceria.

Ao Departamento de Engenharia de Estruturas e à UFMG, pela oportunidade e formação acadêmica.

À CAPES, pelo auxílio financeiro.

Resumo

Este trabalho propõe a representação das relações mecânicas do concreto a partir de uma abordagem da teoria do Aprendizado de Máquina. Este é um estudo inicial na busca da concepção de um modelo constitutivo universal. O concreto, meio parcialmente frágil, apesar de ser um material antigo e amplamente utilizado na construção civil, possui um comportamento mecânico de difícil compreensão e explicação matemática. Tal complexidade, decorrente da heterogeneidade e não-linearidade do meio, ainda motiva diversos estudos e pesquisas nos dias atuais, com a finalidade de desenvolver modelos que sejam capazes de representar, de forma genérica e eficiente, as relações entre tensões e deformações. No contexto das técnicas de aproximação de funções para problemas de regressão, os algoritmos baseados em aprendizado de máquina se destacam. A capacidade de aprender padrões baseados na experiência e de generalizar os conhecimentos adquiridos fazem dos modelos de aprendizagem ferramentas promissoras para aplicação na procura de modelos constitutivos mais representativos e gerais. Nesse estudo, utilizou-se o algoritmo de aprendizagem supervisionada conhecido como Redes Neurais Artificiais. A viabilidade dessa abordagem é verificada através da construção de um modelo constitutivo baseado nas relações de tensão-deformação de dados sintéticos, provenientes de simulações numéricas via elementos finitos planos, por meio da análise de explicabilidade das previsões do modelo e mediante a investigação de sua capacidade de generalização. O modelo constitutivo baseado em redes neurais aqui desenvolvido, se mostrou capaz de capturar o comportamento de tensão-deformação do material e de prever os estados de tensões de estruturas desconhecidas com satisfatória precisão. A prova de generalização do modelo foi realizada através da predição de estados de tensões caracterizados pelo modo misto de falha, comportamento não presente de forma direta nos dados de treinamento. O modelo se mostrou eficaz, também, na previsão de estados de tensões independentemente do tamanho da estrutura e da discretização da malha de elementos finitos usada na geração dos dados de teste.

Palavras-chave: Aprendizado de máquina. Redes neurais artificiais. Modelos constitutivos. Análise não-linear. Método dos elementos finitos. Modelo constitutivo baseado em redes neurais. Concreto.

Abstract

This work proposes the representation of the mechanical relations of concrete from a Machine Learning (ML) theory approach. This is a preliminary study in the search for the conception of a universal constitutive model. Concrete, a quasi-brittle medium, despite being an old and widely used material in civil construction, has a mechanical behavior that is difficult to understand and mathematically explain. Such complexity, resulting from the heterogeneity and nonlinearity of the medium, still motivates several studies and researches nowadays, with the purpose of developing models that are able to represent, in a generic and efficient way, the relations between stresses and strains. In the context of function approximation techniques for regression problems, ML-based algorithms stand out. The ability to learn patterns based on experience and to generalize the acquired knowledge make learning models promising tools for application in the search for more representative and general constitutive models. In this work, the supervised learning algorithm known as Artificial Neural Networks (ANNs) was used. The feasibility of this approach is verified through the construction of a constitutive model based on the stress-strain relationships of synthetic data from numerical simulations via plane finite elements, by analyzing the explainability of the model predictions and by examining its generalization capacity. The Neural Network-based Constitutive Model (NNCM) developed here was shown to be able to capture the material stress-strain behavior and to predict the stress states of unknown structures with satisfactory precision. The proof of generalizability of the model was achieved through the prediction of stress states characterized by the mixed failure mode, a behavior not directly present in the training data. The model also proved to be effective in predicting stress states regardless of the structure size and the finite element mesh discretization used to generate the test data.

Key-words: Machine learning. Artificial neural networks. Constitutive models. Nonlinear analysis. Finite element method. Neural network-based constitutive models. Concrete.

List of tables

List of figures

List of symbols

The list of symbols is subdivided by chapters.

Chapter 2
Constitutive Modeling of Concrete

Smeared Crack Model

E	Young's modulus
G	Transverse elastic modulus
ν	Poisson's ratio
ε_{xx}	Normal strain in the x-direction
ε_{yy}	Normal strain in the y-direction
γ_{xy}	Shear strain in the xy plan
σ_{xx}	Normal stress in the x-direction
σ_{yy}	Normal stress in the y-direction
τ_{xy}	Shear stress in the xy plane
$\{\sigma_l\}$	Local system stress vector
$\{\varepsilon_l\}$	Local system strain vector
$[D_l^s]$	Local secant constitutive matrix
n	Direction normal to the crack
$s; t$	Directions tangential to the crack
ε_{nn}	Normal strain in the direction of the local axis n
ε_{ss}	Normal strain in the direction of the local axis s
γ_{ns}	Shear strain in the ns plane
σ_{nn}	Normal stress in the n-direction
σ_{ss}	Normal stress in the s-direction
τ_{ns}	Shear stress in the ns plane
β_r	Shear retention factor
α_r	Modulus of elasticity reduction factor in the direction normal to the crack
$[T_\sigma(\phi)]$	Stress transformation matrix
$\{\sigma_g\}$	Global system stress vector
$\{\varepsilon_g\}$	Global system strain vector

ϕ	Rotation angle of the coordinate system
$[T_\varepsilon(\phi)]$	Strain transformation matrix
$[D_g^s]$	Global secant constitutive matrix

Damage Laws

E_0	Initial Young's modulus
E_S	Secant Young's modulus
$\sigma(\varepsilon)$	Stress as a function of strain
$w(\varepsilon)$	Damage parameter
$\phi(\varepsilon)$	Scalar damage function
f_c	Compressive strength of concrete
f_t	Tensile strength of concrete
ε_c	Strain corresponding to the concrete compressive strength limit
ε_t	Strain corresponding to the concrete tensile strength limit
ε_{cr}	Critical strain
G_f	Fracture energy
g_f	Area under the softening branch
w	Sum of the microcrack openings in the fracture zone
h	Characteristic length of the material

Chapter 3
Machine Learning

Metrics for Evaluating Results

n	Number of predictions
y_i	i-th expected value
$\overline{y}_i$	i-th predicted value
R^2	Coefficient of determination
R^2_{adj}	Adjusted coefficient of determination
n	Number of independent variables

Artificial Neural Networks

v_j	Induced local field of neuron j
x_i	Input signal i
b_j	Bias of neuron j
$w_{ji}(n)$	Synaptic weight connecting the output of neuron i to the input of neuron j in interaction n
$\varphi(.)$	Activation function
$y(n)$	Target value of the example n

$\bar{y}(n)$	Output signal of the perceptron for example n
η	Learning rate parameter
W	Synaptic weight vector
$\mathscr{E}(\cdot)$	Cost function
∇	Gradient operator
α	Momentum
β_1	Exponential decay rate for the 1_{st} moment estimates
β_2	Exponential decay rate for the 2_{st} moment estimates
ε	Small constant for numerical stability

Chapter 4
Neural Network-based Constitutive Model

Basic Aspects

NN	Neural network as a approximation function of the problem
$\{\Delta\sigma\}$	Stress increment vector
$\{\Delta\varepsilon\}$	Strain increment vector
$\{\sigma\}_n$	Stress vector from iteration n
$\{\varepsilon\}_n$	Strain vector from iteration n

Chapter 5
Training Database

Dataset Generation

f_{ck}	Characteristic compressive strength of concrete
f_t	Characteristic tensile strength of steel
E	Young's modulus
ν	Poisson's ratio
e_c	Concrete specific strain
e_t	Steel specific strain
β_r	Shear retention factor
P	Reference load
σ	Pre-stress

Chapter 6
Neural Network Design

Grid Search Analysis

η	Learning rate parameter
α	Momentum

β_1	Exponential decay rate for the 1_{st} moment estimates
β_2	Exponential decay rate for the 2_{st} moment estimates
ε	Small constant for numerical stability

Chapter 7
Training Process

ML Models Explainability

C	Coalition
V	Coalition value
ϕ_i	Shapley value for feature i
f	Black-box model
x	Input data point
p	Total number of features
S	All possible subsets of feature combinations
$f_x(S)$	Model prediction over the subset S with variable i
$f_x(S \bigcup \{i\})$	Model prediction over the subset S without variable i
f	Black-box model
x	Input data point
p	Total number of features
$E(f(\mathbf{X}))$	Average output value of the model

Local and Global Explanations

σ_{xx}^{n}	Normal stress in the x-direction in the iteration n
σ_{yy}^{n}	Normal stress in the y-direction in the iteration n
τ_{xy}^{n}	Shear stress in the xy plane in the iteration n
ε_{xx}^{n}	Normal strain in the x-direction in the iteration n
ε_{yy}^{n}	Normal strain in the y-direction in the iteration n
γ_{xy}^{n}	Shear strain in the xy plane in the iteration n
σ_{xx}^{n-1}	Normal stress in the x-direction in the iteration $n-1$
σ_{yy}^{n-1}	Normal stress in the y-direction in the iteration $n-1$
τ_{xy}^{n-1}	Shear stress in the xy plane in the iteration $n-1$
ε_{xx}^{n-1}	Normal strain in the x-direction in the iteration $n-1$
ε_{yy}^{n-1}	Normal strain in the y-direction in the iteration $n-1$
γ_{xy}^{n-1}	Shear strain in the xy plane in the iteration $n-1$
σ_{xx}^{n-2}	Normal stress in the x-direction in the iteration $n-2$
σ_{yy}^{n-2}	Normal stress in the y-direction in the iteration $n-2$
τ_{xy}^{n-2}	Shear stress in the xy plane in the iteration $n-2$
ε_{xx}^{n-2}	Normal strain in the x-direction in the iteration $n-2$

ε_{yy}^{n-2}	Normal strain in the y-direction in the iteration $n-2$
γ_{xy}^{n-2}	Shear strain in the xy plane in the iteration $n-2$
$\Delta\sigma_{xx}^{n}$	Normal stress increment in the x-direction in the iteration n
$\Delta\sigma_{yy}^{n}$	Normal stress increment in the y-direction in the iteration n
$\Delta\tau_{xy}^{n}$	Shear stress increment in the xy plane in the iteration n
$\Delta\varepsilon_{xx}^{n}$	Normal strain increment in the x-direction in the iteration n
$\Delta\varepsilon_{yy}^{n}$	Normal strain increment in the y-direction in the iteration n
$\Delta\gamma_{xy}^{n}$	Shear strain increment in the xy plane in the iteration n

Appendix A
Nonlinear Finite Element Analysis

$\{U\}$	Displacement field
λ	Load factor
$[K_t]_{j-1}^{i}$	Tangent stiffness matrix in the iteration $j-1$ from step i
$\{\delta U\}_{j}^{i}$	Incremental displacements vector in the iteration j from step i
$\delta\lambda_{j}^{i}$	Load factor increment in the iteration j from step i
$\{P\}$	Reference load vector
$\{Q\}_{j}^{i}$	Residual forces vector in the iteration j from step i
$\{\delta U\}_{j}^{P}$	Incremental displacements vector in the iteration j referring to the portion of the reference load
$\{\delta U\}_{j}^{Q}$	Incremental displacements vector in the iteration j referring to the portion of the residual force
$\{F\}_{j}$	Internal forces vector in the iteration j
$[L]$	Partial derivative matrix
$[N]$	Approximation functions matrix
$\{\sigma\}$	Stress state at a particular point
$[C_t]_j$	Constitutive matrix in the iteration j

Appendix B
Neural Network-based Constitutive Matrix

$[C_t]$	Tangent stress-strain matrix

List of abbreviations and acronyms

Adam	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient Algorithm
AI	Artificial Intelligence
ANN	Artificial Neural Network
BEM	Boundary Element Method
CANN	Constitutive Artificial Neural Network
FEM	Finite Element Method
FPZ	Fracture Process Zone
GB	Gradient Boosting
INSANE	INteractive Structural ANalysis Environment
IP	Integration Point
LEFM	Linear Elastic Fracture Mechanics
LIME	Local Interpretable Model-agnostic Explanations
LMS	Least Mean Square
MAE	Mean Absolute Error
MFNN	Multilayer Feedforward Neural Network
ML	Machine Learning
MLP	Multilayer Perceptron
MM	Mesh-free Method
MSE	Mean Squared Error
NANN	Nested Adaptive Neural Network
NNCM	Neural Network-based Constitutive Model
ReLu	Rectifier Linear Function
RF	Random Forest
RMSProp	Root mean Square Propagation
SGD	Stochastic Gradient Descent
SHAP	SHapley Additive exPlanations
SVM	Support Vector Machine

Table of contents

Chapter 1

Introduction

Structural engineering is the branch of civil engineering responsible for the structural calculation and design of projects for buildings, bridges, viaducts, tunnels, retaining walls, dams and various other special works of art. In the academic environment, structural engineering research is dedicated to finding mathematical models that support the stress analysis and the subsequent design of structural elements. In the context of structural analysis, these studies seek to create and improve theoretical formulations that explain the physical phenomena that occur in structures, that is, the response of the structural system to external actions, as well as the mechanical behavior of materials used in the construction of these systems.

From the industrialisation of cement production, concrete has become one of the most widely used material in civil construction. Its application in structural engineering, either as plain concrete, reinforced concrete or pre-stressed concrete, it is explained by its resistance to compression, economic feasibility, high durability and flexibility, being able to assume almost any desired geometric shape. Although it is a widely used material, its heterogeneous characteristic makes its mechanical behavior difficult to understand and describe numerically. This complexity explains why the constitutive modeling of concrete is still object of extensive studies. Besides that, in engineering practice, a reliable mechanical analysis means designing increasingly safer and optimized structures.

The growth of computer technology has enable the progress of a series of numerical methods for application in structural analysis, such as the Finite Element Method (FEM), the Boundary Element Method (BEM) and the Mesh-free Method (MM). These approaches have enabled the development of several constitutive models in order to represent the real behavior of *quasi-brittle* materials, such as concrete. However, so far, there is no constitutive model that may be considered of generic use. Each of the developed models has advantages and disadvantages and are suitable for specific situations.

From the methodological perspective of conventional constitutive modeling, it is understood that the proposition of a constitutive model it is closely related to the solution of a feature identification and function approximation, arising from experimental obser-

vations. For a long time, statistics methods were responsible for the main studies and advances on pattern recognition and regression techniques. However, nowadays, the gold standard lies in Artificial Intelligence (AI), through Machine Learning (ML) algorithms. ML techniques gives a system the ability to learn, to store that knowledge and use it to predict and generalize unknown informations. The model captures trends and patterns from the database of interest through the training process. The Artificial Neural Network (ANN) is one of the most used ML technique in regression problems and has been the object of study in the constitutive representation of materials since the late 1980s.

The application of ANNs in modeling materials behavior, that is, the mechanical relation between stress and strain, was driven after Hornik et al. (1989) concluded that feed-forward ANNs was able to approximate virtually any continuous function and therefore, established as a class of universal approximators. Considering that physical characteristics of materials are essentially nonlinear, it was reasonable to think about ANNs abilities to learn and generalize as tools for representing materials constitutive properties.

The literature reveals that ANNs has enabled the behavior representation of complex engineering materials. The basic strategy to develop a neural network-based constitutive model (NNCM) is to train a ANN, using the *backpropagation algorithm*, on the stress-strain results from data that properly correspond to the material information, with stresses, strains and strains increments as input data and stresses increments as output data. These data can be obtained by experimental means or even synthetically, from numerical simulations, since obtaining large amounts of experimental data is notoriously difficult in practice. This computation paradigm was already applied in the modeling of plain concrete under biaxial load and uniaxial cyclic load (Ghaboussi et al., 1991), in composite materials like reinforced concrete and modern fiber-epoxy composites (Ghaboussi et al., 1991), in drained and undrained sand under triaxial tests (Ghaboussi and Sidarta, 1998), in undrained triaxial compression tests of sands with varying grain size distribution and stress history (Ellis et al., 1995), in representing the behavior of viscoplastic materials (Furukawa and Yagawa, 1998), and other applications related to structural analysis.

In addition, NNCM are noteworthy because they can be easily incorporated into finite element softwares as an alternative to the conventional mathematical constitutive models, currently used in structure nonlinear analysis for the solution of boundary value problems, as demonstrated by Javadi et al. (2003), Lefik and Schrefler (2003), Hashash et al. (2004), Liang and Chandrashekhara (2008), Stoffel et al. (2019), Im et al. (2021) and others.

NNCM presents itself as a great alternative to traditional models because has the ability to generalise the problem represented by the data more broadly than classical statistical methods, have the ability to continuously learn as additional material response data becomes available and, when necessary, does not require the calculation of hard-to-obtain material parameters.

Taking into consideration what has been presented, this work proposed the develop-

ment of an NNCM by training an ANN with concrete material stress-strain data. The training database was built from numerical simulations via finite elements with the support of the INSANE[1] (*INteractive Structural ANalysis Environment*) system. The model was developed with the help of the well-known machine learning libraries *scikit-learn*[2] and *keras*[3]. Although ANNs are widely used, they are still targets of much criticism based on the argument that they are not a reliable algorithm due to their lack of clarity in the reason for their predictions. In this sense, this work also presents explicability analyses of the model, in order to demonstrate that the model answers are not random and that there is physical consistency with the type of problem. Finally, the model was tested and validated over synthetic data from structural systems other than those used for training.

1.1 Research Group and Line

This work is linked to the INSANE Research Group, within the area of Numerical and Computational Simulations in Solid and Structural Mechanics and the research line of Intelligent Systems for Structural Engineering. This line of research is dedicated to the study, implementation and application of Machine Learning algorithms and Bio-inspired Methods in the solution of Solid and Structural Mechanics problems. The proposition exposed here is part of the research project titled Intelligent Constitutive Models for Quasi-Brittle Materials. This is a pioneer work in the Research Group.

1.2 Objectives

1.2.1 General Objective

This dissertation aimed to develop, demonstrate the feasibility and explainability of the responses of a Machine Learning-based Constitutive Model, from the perspective of the Artificial Neural Networks algorithm, in order to simulate the behavior of concrete, a quasi-brittle material with strong nonlinear characteristics.

1.2.2 Specific Objectives

This study sought to achieve the following specific objectives:

- To study ANNs in the literature and its application in the constitutive modeling of quasi-brittle materials;

[1]INSANE is an open source software based on the Object-Oriented Programing paradigm and developed by the Structural Engineering Department of the Federal University of Minas Gerais (UFMG) and available at `http://www.insane.dees.ufmg.br`

[2]Scikit-learn is an open source codes for predictive data analysis for the Python programming language.

[3]Keras is an open source software library that provides a Python interface for artificial neural networks

- Create a synthetic database, composed of numerical results from the INSANE system, arising from nonlinear analyses using conventional constitutive models well consolidated in the literature, which enable the characterization of quasi-brittle materials and their applicability in the training and validation of the NNCM;
- Know the main machine learning libraries used in the scientific community, their languages and codes already implemented;
- To study the adjustment of the various hyperparameters and architectures possibilities of ANNs, in order to guide the choice of the best variables for network training;
- Train and interpret the results of the algorithm by analyzing the explainability of the predictions of the model developed with the help of available tools;
- Test and validate the NNCM by comparing the obtained results with that one's from conventional constitutive models.

1.3 Outline

This work is organized into 9 chapters and 3 appendices. After this introduction (Chapter 1), Chapter 2 addresses some aspects related to constitutive modeling of quasi-brittle materials, focusing on the smeared cracking model and its main damage laws. Chapter 3 introduces artificial intelligence and the main concepts related to machine learning algorithms, emphasizing the artificial neural networks technique. Chapter 4, in turn, connects constitutive modeling to artificial neural networks, providing a historical contextualization of the application of this algorithm in structural engineering, focusing on the representation of concrete behavior. The numerically simulated structural systems for generating the synthetic training data are presented in Chapter 5. Then, Chapter 6 presents the definition of the architecture of the model developed in this work, through the optimization study of its hyperparameters. This dissertation also covers the methodology used for training the model and discusses the importance of explainability of ML models in practical engineering, presenting an analysis of explainability of the model developed in this work (Chapter 7). Chapter 8 presents the model validation analysis, by testing the algorithm on data from unknown structural systems. Chapter 9 concludes this work, summarizing the main contributions and future research propositions. Finally, Appendix A provides an overview of the nonlinear analysis process, Appendix B presents the formulation for obtaining the neural network-based constitutive matrix and Appendix C contains the complementary results to those presented in Chapter 8.

Chapter 2

Constitutive Modeling of Concrete

In this chapter, a quick review on constitutive modeling of quasi-brittle materials, focusing on smeared cracking models and its damage laws will be presented.

Concrete, from a macroscale point of view, under low stress states, can be considered a homogeneous, isotropic and linear-elastic material. However, its composition is formed by a mixture of aggregates, binders (cement being the most used) and water. Then, if observed on a meso-level (see Fig. 2.1), concrete is no longer seen as homogeneous and is classified as heterogeneous. Under a state of loading and consequent deformations, a microcrack nucleation process starts, resulting in a nonlinear behavior at the macroscale.

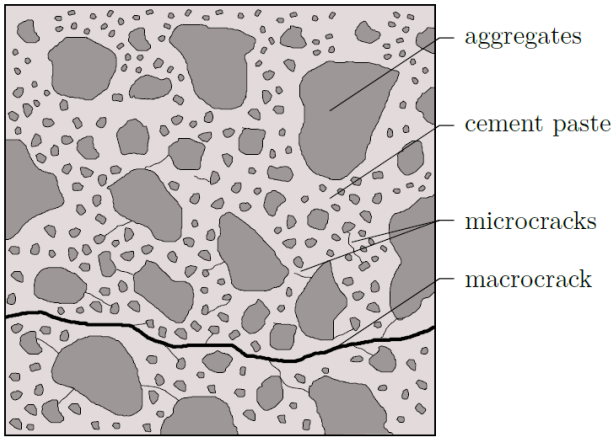
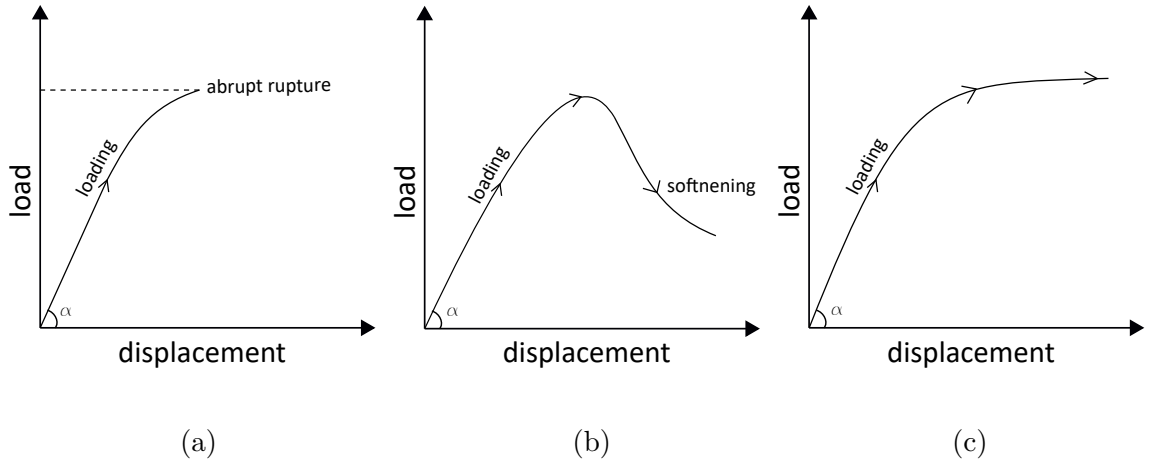


Figure 2.1: Concrete mesostructure (Gori, 2018).

Regarding its mechanical behavior, concrete is characterized as a *quasi-brittle* media. Thus, basing on the behavior of the material under an uniaxial tension stress, from the perspective of the load-displacement relation, while *brittle* materials fails abruptly after reaching a certain level of stress (Fig. 2.2a), *quasi-brittle* materials have a nonlinear behavior after peak stress (Fig. 2.2b) showing a gradual decrease in its stiffness in a process

known as softening. *Ductile* materials, in turn, also exhibit nonlinear post-peak behavior but are characterized by a yield plateau and high strains before rupture (Fig. 2.2c).



(a) (b) (c)

Figure 2.2: Material behavior. (a) Brittle, (b) Quasi-brittle, and (c) Ductile .

The aforementioned characteristics make concrete a material with stress-strain relationships that are difficult to understand and describe analytically. For Gori (2018):

> *"The degradation of a quasi-brittle medium is a complex phenomenom which strongly depends on the heterogeneous character of the medium microstructure."*

Understanding this behavior has been the object of study of several scientific researches over the last decades. The first models developed were based on empirical approximations and simplifying assumptions. After the establishment of the finite element method, several more elaborate constitutive models were studied and developed in order to approximate the behavior of concrete and allow its modeling in nonlinear analyses of structures with greater fidelity.

The models based on linear elastic fracture mechanics (LEFM) had their initial concepts conceived after the studies of Griffith (1920). After experimental tests and observations, Griffith (1920) demonstrated that the existence of fractures, as well as any sort of imperfection in the microstructure of a material under certain external force, results in a significant reduction in its tensile strength. This phenomenon is responsible for the differences observed between theoretical and experimental values of material strength. In the case of concrete, the tensile or compression rupture process is characterized by the nucleation of microcracks, which may eventually agglutinate into a geometric discontinuity or macrocrack. This macrocrack is known as a fissure or crack. Among the models based on fracture mechanics, two are widely applied in concrete constitutive modeling: the Discrete Crack Model and the Smeared Crack Model.

The discrete crack model was first portrayed by Ngo and Scordelis (1967), and presents the crack as a geometric discontinuity. The cracks are inserted into the finite element mesh

in a discrete manner, by changing the mesh in the crack region (Fig. 2.3a). Thus, the discrete model determines that the crack propagates from the ultimate strength of the material. The material's ultimate strength, in turn, can be evaluated by different quantities, such as stresses at the crack tip, stress-intensity factor, fracture energy, among others (Penna, 2011). However, this approach results in a continuous change of node connectivities and restricts crack propagation at the interface between finite elements. Both consequences of the discrete crack process are perceived as drawbacks, since they do not suit the nature of the FEM. Another negative point is that, in reality, the crack formation process is not abrupt, since there are residual stresses that prevent its immediate opening. Several studies have been developed in an attempt to eliminate the disadvantages of this formulation. Among them, the development of automatic mesh redefinition procedures (Ingraffea and Saouma, 1985, Wawrzynek et al., 1987), the application of meshless methods (Belytschko and Black, 1999, Moës et al., 1999, Rabczuk and Belytschko, 2004, 2007), and techniques that allow discrete cracks extending through the finite elements (Blaauwendraad and Grootenboer, 1981, Blaauwendraad, 1985, Wawrzynek et al., 1987), stand out.

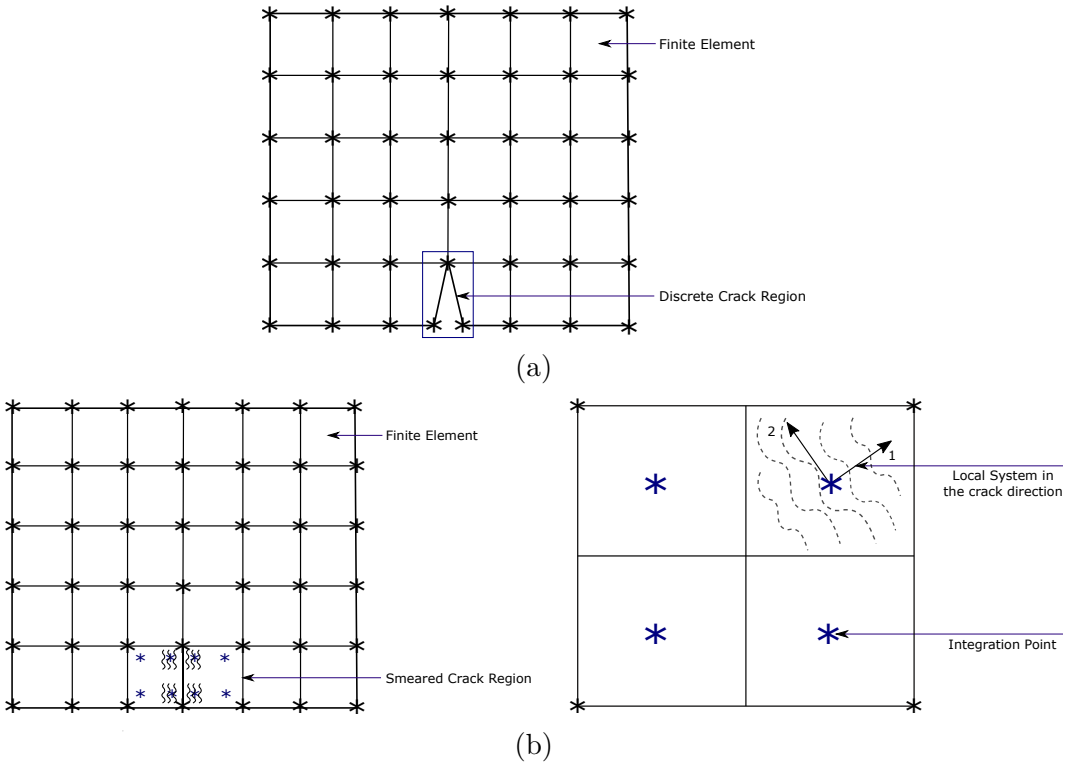


Figure 2.3: Cracking models. (a) Discrete cracks and (b) Smeared cracks. Adapted from Pitangueira (1998).

On the other hand, the smeared crack model idealises the crack as a continuous media, formed by parallel cracks. Rashid (1968) was the first to propose a smeared approach applied to concrete analysis. In this method, cracks are not represented explicitly, but by

imposing parallel cracks at the point of analysis of the stress-strain relations of the finite element (Fig. 2.3b). In other words, cracking is represented by the gradual degradation of the stiffness of the media due to the formation of several microcracks. Subsequent studies were carried out in order to develop this model (Suidan and Schnobrich, 1973, Bažant and Oh, 1983, Rots et al., 1985, de Borst and Nauta, 1985, Rots and De Borst, 1987). The smeared cracking models are based on the material's physical properties degradation monitoring. In this process, the gradual decrease of stresses with increasing strains describe the evolution of cracks or fissures (Penna, 2011). Although this model requires more parameters for modeling residual stresses and is unable to represent cracking geometrically, it outperforms the discrete crack model because it has the advantage of the capacity to represent the material cracks without the need to redefine the finite element mesh, the ability to represent strains after the cracking process, and ease of implementation. In addition, the smeared crack model is able to consider regions with compression damage, since it evaluates the stiffness deterioration from a modification of the local constitutive relation.

The consolidation of the smeared cracking model demanded the development of stress-strain laws to represent the behavior of concrete in degraded regions. On this subject, the works of Carreira and Chu (1985), Carreira and Chu (1986), Boone et al. (1986) and Boone and Ingraffea (1987) are highlighted.

Details of the smeared cracking model formulation (Section 2.1), an overview of the damage laws for concrete (Section 2.2) and a summary of why this model was chosen for this work (Section 2.3) will be presented next .

2.1 Smeared Crack Model

Smeared crack models can be categorized into fixed direction cracking models and variable direction cracking models. For the first case, the crack orientation is considered fixed throughout the damage process. For the second, the crack direction may rotate, according to the variation of the principal strains.

Smeared crack models consider that the material media is, initially, linear, elastic and isotropic, i.e., with constant Young's modulus E and Poisson's ratio ν. For the plane stress state, before the beginning of cracking, there is the constitutive relation described according to Eq. (2.1).

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xy} \end{Bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \dfrac{1}{2(1-\nu)} \end{bmatrix} \begin{Bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \gamma_{xy} \end{Bmatrix} \tag{2.1}$$

After the material point reaches the beginning of cracking, based on concrete strength

criteria and its fracture mechanics parameters, it is assumed that the material behaves orthotropically. Thus, the Young's modulus changes, characterizing the heterogeneous behavior of the material. The crack is distributed in a local reference system, in the principal strain directions. The stress-strain laws are described according to this local axis system n, s and t (Fig. 2.4), where n is the direction normal to the crack (rupture mode I [1]) and s and t are the directions tangential to the crack (rupture mode II [1]).

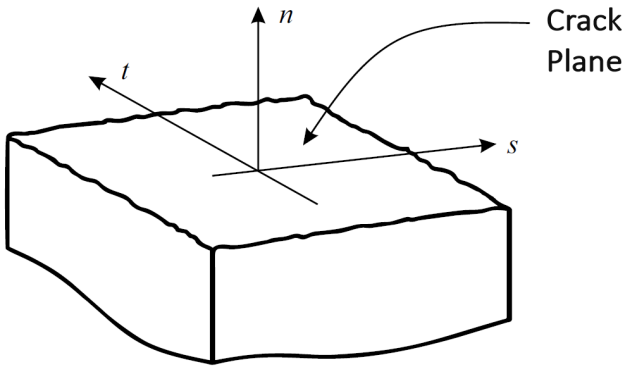


Figure 2.4: Crack coordinate system.

$$\{\sigma_l\} = [D_l^s]\,\{\varepsilon_l\} \tag{2.2}$$

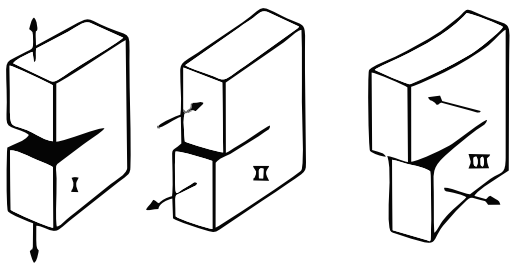
In the Eq. (2.2), $\{\sigma_l\}$ is the local system stress vector; $[D_l^s]$ is the local secant constitutive matrix; e $\{\varepsilon_l\}$ is the strain vector along the local axes.

The first studies (Rashid, 1968, Cervenka, 1970, Valliappan and Doolan, 1972) considered the stiffness null in perpendicular and tangential direction to the crack opening. Thus, the following orthotropic constitutive relation is obtained:

$$\begin{Bmatrix} \sigma_{nn} \\ \sigma_{ss} \\ \tau_{ns} \end{Bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{Bmatrix} \varepsilon_{nn} \\ \varepsilon_{ss} \\ \gamma_{ns} \end{Bmatrix} \tag{2.3}$$

However, it is known that this proposition does not match the experimental reality, because a structural element even under cracking is still able to transmit stresses in normal

[1] The propagation of a fracture can be decomposed according to three distinct components that are called modes I, II and III, whose representations are shown in the figure below:



and tangential directions. In addition, studies have found that the imposition of an abrupt discontinuity can cause numerical instabilities.

The sequence of studies led to the consideration of a residual shear stiffness, lessened by a reduction factor β_r known as the shear retention factor (Suidan and Schnobrich, 1973). This model approximates the numerical behavior to the physical behavior of the material, besides providing a greater numerical stability and can be understood as a way to consider the interlock between the roughness of the crack faces (de Borst, 2002). Then, an evolution of the model also considers a reduction factor of the Young's modulus. To this end, there is

$$\begin{Bmatrix} \sigma_{nn} \\ \sigma_{ss} \\ \tau_{ns} \end{Bmatrix} = \begin{bmatrix} \alpha_r E & 0 & 0 \\ 0 & E & 0 \\ 0 & 0 & \beta_r G \end{bmatrix} \begin{Bmatrix} \varepsilon_{nn} \\ \varepsilon_{ss} \\ \gamma_{ns} \end{Bmatrix} \tag{2.4}$$

where, $0 < \alpha_r \leq 1$ e $0 < \beta_r \leq 1$ and G is the transverse elastic modulus.

Later, Bažant and Oh (1983) introduced to the formulation exposed above, stiffness coefficients outside the main diagonal of the constitutive matrix, in order to consider Poisson's ratio in the material degradation process. The consideration of degradation in the direction tangent to the cracking plane was finally developed by Rots et al. (1985). This formulation relies on the inversion of the flexibility matrix and considers only tensile damage. These contributions to the formulation of the orthotropic constitutive law resulted in Eq. (2.5), below:

$$[D_l^s] = \frac{1}{(1-\nu^2\alpha_r)} \begin{bmatrix} \alpha_r E & \nu\alpha_r E & 0 \\ \nu\alpha_r E & E & 0 \\ 0 & 0 & (1-\nu^2\alpha_r)\,\beta_r \frac{E}{2(1+\nu)} \end{bmatrix} \tag{2.5}$$

As previously stated, the secant constitutive matrix is given in a local reference, according to the directions of principal strains. This tensor must also be defined in a global system. So, being ϕ the angle between the local and global coordinate systems, the strains in the local system are related to the strains in the global system by means of the transformation matrix $[T_\varepsilon(\phi)]$, as presented in Eq. (2.6).

$$\{\varepsilon_l\} = [T_\varepsilon(\phi)]\,\{\varepsilon_g\} \tag{2.6}$$

Similarly, the local and global stresses correlate through the transformation matrix $[T_\sigma(\phi)]$, as shown in Eq. (2.7).

$$\{\sigma_l\} = [T_\sigma(\phi)]\,\{\sigma_g\} \tag{2.7}$$

Therefore, it is possible to establish a correlation between the constitutive matrix for the global and the local system, based on the relationships given by Eq. (2.6) and

Eq. (2.7).

$$\left[D_g^s\right] = \left[T_\sigma^{-1}(\phi)\right] \left[D_l^s\right] \left[T_\varepsilon(\phi)\right] \tag{2.8}$$

However, to sequence a structural analysis iterative incremental process, it is necessary to obtain the tangent constitutive tensor in a global referential. This transformation is demonstrated in details in the works of Pitangueira (1998) and Penna (2011).

Pitangueira (1998) and Penna (2011) also describe, in their works, a generalization of the smeared crack models based on flexibility inversion that enables the damage process of the material both in tension and compression, through the assumption of obtaining the Young's moduli in the local directions independently, in accordance with the current state of strain, being necessary only specific laws for tension and compression. Furthermore, some assumptions for symmetrization of the flexibility matrix of the material are presented. The subsequent subsection briefly discusses concrete damage evolution laws.

2.2 Damage Laws

Concrete constitutive modeling with the smeared crack models requires the existence of a damage function that allows describing the material deterioration process through the degradation of the constitutive matrix coefficients (E_1, E_2 and G_{12}). This process modifies the initial elastic-linear material properties according to the strains in the local crack plane direction.

In mathematical form, one has:

$$E_1 = E_0 \left(1 - w\left(\varepsilon_1\right)\right) \tag{2.9}$$

$$E_2 = E_0 \left(1 - w\left(\varepsilon_2\right)\right) \tag{2.10}$$

where $w(\varepsilon)$ is known as the damage parameter. When it has no degradation, $w(\varepsilon) = 0$ and when the degradation is total, $w(\varepsilon) = 1$. Hence, $0(\varepsilon) \leq 1$.

Writing another way, one has:

$$E_1 = E_0 \left(\phi(\varepsilon_1)\right) \tag{2.11}$$

$$E_2 = E_0 \left(\phi(\varepsilon_2)\right) \tag{2.12}$$

where $\phi(\varepsilon_i) = \left(1 - w\left(\varepsilon_i\right)\right)$ is a scalar damage function.

Then, let any given stress-strain law be expressed by

$$\sigma(\varepsilon) = E_S \varepsilon \tag{2.13}$$

where E_S is the secant Young's modulus.

Thus, the degradation law can be rewritten as

$$E_S = E_0\left(\phi(\varepsilon)\right) \tag{2.14}$$

Then, equating Eq. (2.13) and Eq. (2.14), one has that

$$\phi(\varepsilon) = \frac{\sigma(\varepsilon)}{E_0\varepsilon} \tag{2.15}$$

Therefore, the Eq. (2.15) demonstrates that the degradation function depends only on the strain occurring in that direction, as the stress-strain law is prescribed and can be obtained empirically or experimentally. For a reliable representation of the constitutive behavior of concrete, it is necessary to know precisely the softening branch of this curve. This motivation has driven several propositions of stress-strain curves, both for compression and tension. Here, the focus will be on some of the laws implemented in the INSANE system. Next is the $\phi(\varepsilon)$ representation for modeling material behavior for both compression and tensile laws.

2.2.1 Compression Laws

For a bilinear relation:

The stress-strain diagram for compression damage presents as parameters the initial Young's modulus (E_0), the Young's modulus of the descending branch (E_2) and the compressive strength of concrete (f_c), according to Fig. 2.5.

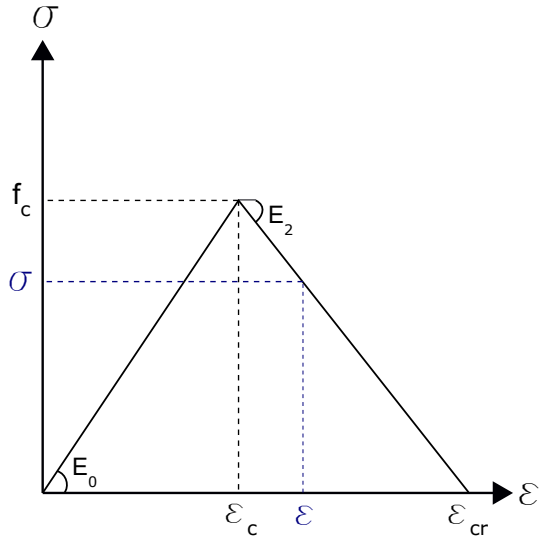


Figure 2.5: Stress-strain curve for bilinear compression law.

For any given point on the stress-strain curve σ and ε, the following relation is given

$$\sigma = f_c + E_2 \left(\varepsilon - \varepsilon_c\right) \tag{2.16}$$

or

$$\sigma = E_0 \varepsilon_c + E_2 \left(\varepsilon - \varepsilon_c\right) \tag{2.17}$$

Rearranging Eq. (2.17), gives

$$\sigma = E_2 \left[\varepsilon - \varepsilon_c \left(1 - \frac{E_0}{E_2}\right)\right] \tag{2.18}$$

where ε_c is the strain corresponding to the concrete compressive strength limit and ε_{cr} the critical strain.

Therefore, the damage function for regions in compression for the bilinear model is

$$\phi(\varepsilon) = \begin{cases} 1, & if \quad \varepsilon < \varepsilon_c \\ E_2 \left[\varepsilon - \varepsilon_c \left(1 - \dfrac{E_0}{E_2}\right)\right], & if \quad \varepsilon_c < \varepsilon < \varepsilon_{cr} \\ 0, & if \quad \varepsilon > \varepsilon_{cr} \end{cases} \tag{2.19}$$

For the exponential law proposed by Carreira and Chu (1985) (Fig. 2.6), the stress function is given by

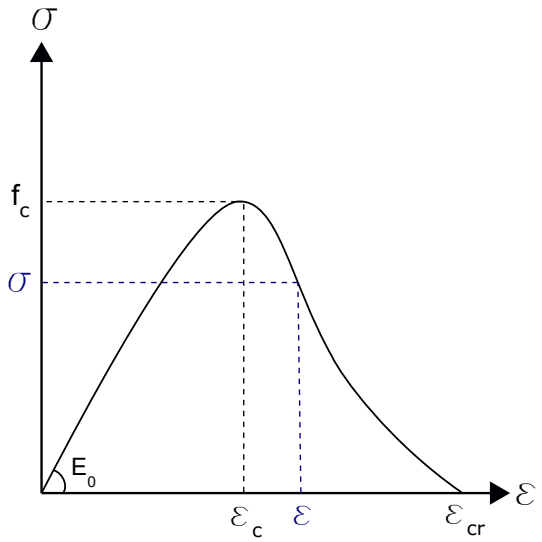


Figure 2.6: Stress-strain curve for Carreira's compression law.

$$\sigma = f_c \frac{k \left(\dfrac{\varepsilon}{\varepsilon_c}\right)}{k - 1 + \left(\dfrac{\varepsilon}{\varepsilon_c}^{k}\right)} \tag{2.20}$$

being

$$k = \frac{1}{1 - \dfrac{f_c}{\varepsilon_c E_0}} \tag{2.21}$$

Then, the nonlinear compressive damage law becomes

$$\phi(\varepsilon) = \frac{1}{1 + \left(\dfrac{\varepsilon}{\varepsilon_c}^{k}\right)\left(\dfrac{E_0 \varepsilon_c}{f_c} - 1\right)} \tag{2.22}$$

2.2.2 Tension Laws

For tensile damage, from the stress-strain diagram, the Young's modulus (E), tensile strength (f_t) and fracture energy (G_f) can be obtained. Fig. 2.7a and Fig. 2.7b show the softening curves for a smeared crack model and a discrete crack model, respectively.

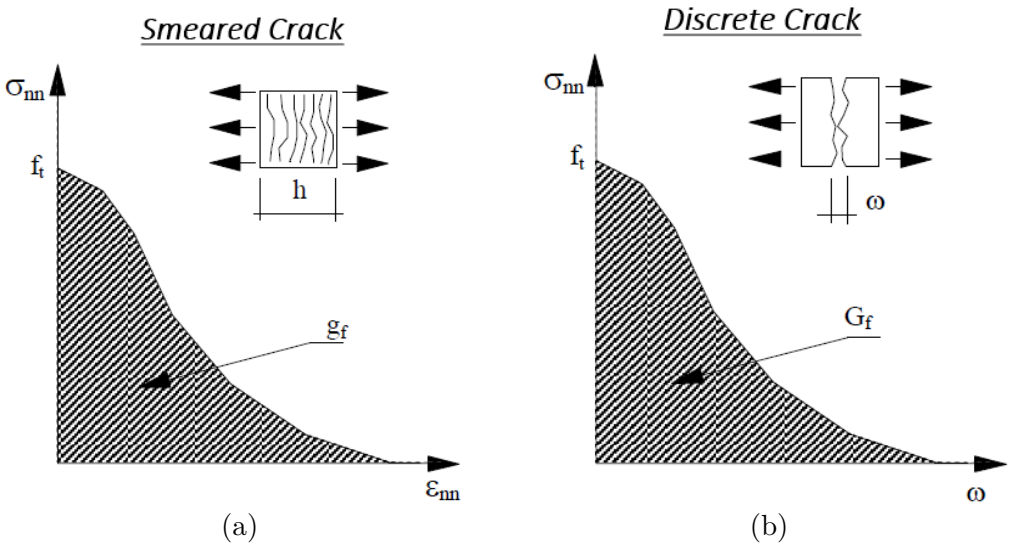


Figure 2.7: Fracture mechanics parameters. (a) Smeared crack, (b) Discrete crack. Adapted from Pitangueira (1998).

The area g_f under the diagram in Fig. 2.7a is represented by

$$g_f = \int \sigma_{nn} d\varepsilon_{nn} \tag{2.23}$$

while the fracture energy (G_f) of Fig. 2.7b, defined by Hillerborg et al. (1976) and Hillerborg (1985) as the amount of energy required to create a fracture with unit area in mode I fracture, can be expressed by

$$G_f = \int \sigma_{nn} dw \tag{2.24}$$

where w represents the sum of the microcrack openings in the fracture zone.

From the perspective of the smeared crack model, w represents an accumulation of strains acting on a range h of the element, defined as the characteristic length of the material, a property of the medium that aims to define the extent of the Fracture Process Zone (FPZ). This quantity is known to have a representative size of the mixture, large enough to treat the medium as continuous and small enough to represent the heterogeneity of the material.

Thus, one has

$$w = \int \varepsilon_{nn} dn \tag{2.25}$$

Assuming that ε_{nn} is uniformly distributed in dimension h, the Eq. (2.25) looks as

$$w = h\varepsilon_{nn} \tag{2.26}$$

Combining Eq. (2.23), Eq. (2.24) and Eq. (2.26) gives the following relation between g_f and G_f

$$G_f = hg_f \tag{2.27}$$

Therefore, for a bilinear function:

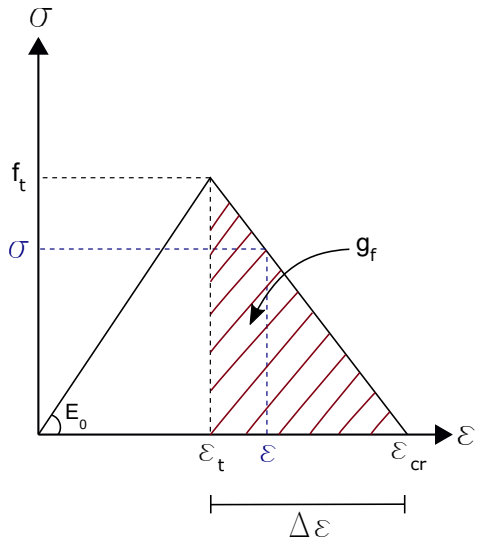


Figure 2.8: Stress-strain curve for bilinear tension law.

The area under the softening curve, represented in Fig. 2.8 is given by

$$g_f = \frac{f_t \Delta\varepsilon}{2} \tag{2.28}$$

where

$$\Delta\varepsilon = \frac{2g_f}{f_t} \tag{2.29}$$

The critical strain is given by

$$\varepsilon_{cr} = \varepsilon_t + \frac{2g_f}{f_t} \tag{2.30}$$

where ε_t is the strain corresponding to the concrete tensile strength limit.

For any given point on the stress-strain curve (σxε), one has the following relation:

$$\sigma = \frac{f_t\left(\varepsilon_{cr} - \varepsilon\right)}{\Delta\varepsilon} \tag{2.31}$$

Rearranging Eq. (2.31), one has

$$\sigma = \frac{f_t\left(\varepsilon_{cr} - \varepsilon\right)}{\left(\varepsilon_{cr} - \varepsilon_t\right)} \tag{2.32}$$

Therefore, the damage function for regions in tension, for the bilinear model is expressed by

$$\phi(\varepsilon) = \begin{cases} 1, & if \quad \varepsilon < \varepsilon_t \\ \dfrac{f_t\left(\varepsilon_{cr} - \varepsilon\right)}{E_0\varepsilon\left(\varepsilon_{cr} - \varepsilon_t\right)}, & if \quad \varepsilon_t < \varepsilon < \varepsilon_{cr} \\ 0, & if \quad \varepsilon > \varepsilon_{cr} \end{cases} \tag{2.33}$$

For the exponential law proposed by Boone and Ingraffea (1987) and represented in Fig. 2.9:

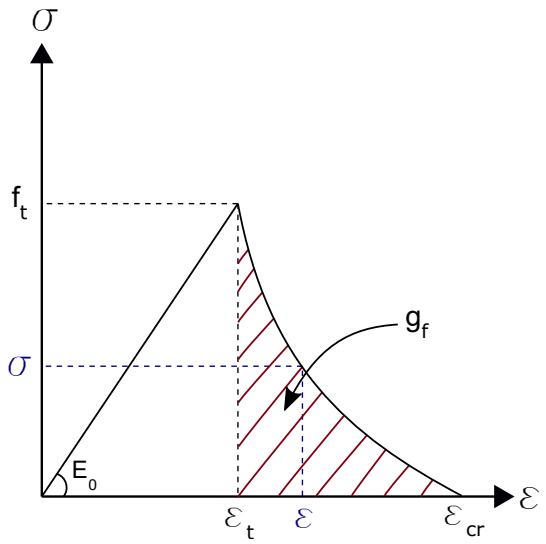


Figure 2.9: Stress-strain curve for exponential tension law.

The stress function is given by

$$\sigma = f_t e^{-k(\varepsilon - \varepsilon_t)} \tag{2.34}$$

where

$$k = \frac{h f_t}{G_f} \tag{2.35}$$

or

$$k = \frac{f_t}{g_f} \tag{2.36}$$

Therefore, the damage function for regions in tension, for the model proposed by Boone and Ingraffea (1987) is given by

$$\phi(\varepsilon) = \begin{cases} 1, & if \quad \varepsilon < \varepsilon_t \\ \dfrac{f_t}{E_0 \varepsilon} e^{-k(\varepsilon - \varepsilon_t)}, & if \quad \varepsilon > \varepsilon_t \end{cases} \tag{2.37}$$

Carreira and Chu (1986) also presented a general equation for tensioned reinforced concrete, similar to that previously defined for compression (Eq. (2.38)).

$$\sigma = f_t \frac{k\left(\dfrac{\varepsilon}{\varepsilon_t}\right)}{k - 1 + \left(\dfrac{\varepsilon}{\varepsilon_t}^{k}\right)} \tag{2.38}$$

as

$$k = \frac{1}{1 - \dfrac{f_t}{\varepsilon_t E_0}} \tag{2.39}$$

Fig. 2.10 illustrates the curve.

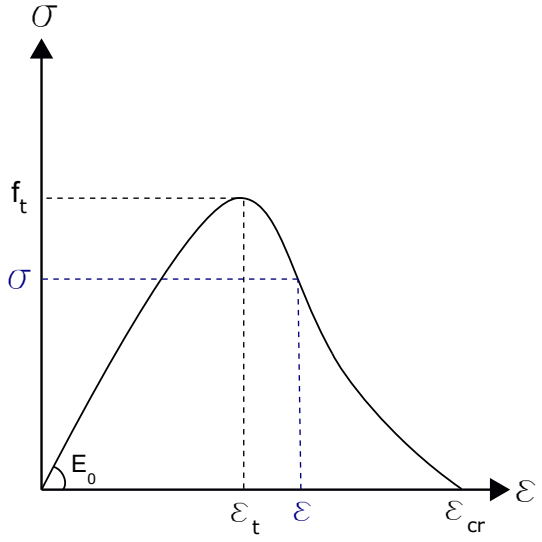


Figure 2.10: Stress-strain curve for Carreira's tension law.

Thus, the nonlinear tensile damage law becomes

$$\phi(\varepsilon) = \frac{1}{1 + \left(\frac{\varepsilon}{\varepsilon_t}^{k}\right)\left(\frac{E_0\varepsilon_t}{f_t} - 1\right)} \tag{2.40}$$

2.3 Summary

The concrete is idealized, initially, as a continuous, homogeneous and isotropic medium. However, it is known that the constituents of the material observed at the mesoscale and the phenomena that permeate the cracking process make it a heterogeneous medium, with nonlinear behavior after limit stresses of tension and compression. Furthermore, it is known that concrete is a quasi-brittle material, with inelastic behavior characterized by a process of stress drop with increasing strains, known as softening.

For constitutive modeling of quasi-brittle media, from the perspective of models based on nonlinear fracture mechanics, the smeared crack model is the most suitable for application via FEM. It does not require mesh redefinition, allows the evaluation of material deterioration due to compression damage, and is easy to implement computationally, being available in the INSANE system.

The use of the smeared crack model has driven several studies aiming to define a constitutive law for concrete. The definition of a general stress-strain law is not trivial, because the parameters obtained experimentally are not intrinsic to the material, since they depend on the characteristics of the concrete type and also on the external test conditions. In addition, the constitutive law defines the nonlinearity of the medium.

In this work, the constitutive model for smeared cracking with fixed direction will be used, with the equations proposed by Carreira and Chu as the compression and tension

constitutive laws (Eq. (2.22) and Eq. (2.40)). This is a consolidated model, which presented the best results in the numerical analyses presented in Penna (2011), with input parameters known and directly obtained from experimental tests, besides being able to represent several concretes with good fidelity.

The difficulty in obtaining constitutive models for structural analysis of quasi-brittle materials highlights how the proposition of constitutive models based on ML can be a good alternative in the search for more generalist models.

Finally, the formulation for the nonlinear analysis process via the FEM implemented in the INSANE system and used for generating the synthetic data with the model and constitutive laws defined above, is presented in the Appendix A of this work.

Chapter 3

Machine Learning

In this chapter, a quick review on the basic concepts concerning machine learning and the artificial neural network technique will be presented.

Artificial Intelligence (AI) is a wide-ranging branch of computer science defined by Kaplan and Haenlein (2019) as the ability of a system to interpret and learn information contained in an external data, and further, to generalize such knowledge to unknown data, in a flexible and adaptive way, in order to solve a given task. An AI system must be able to store knowledge, apply knowledge to solve problems and acquire new knowledge through experience, thus having three fundamental components: representation, reasoning and learning (Sage, 1990). According to Fischler and Firschein (1987):

> *"Knowledge refers to stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world."*

In short, AI aims to transfer to machines the human-like capacity for inferences and decision-making.

Machine Learning (ML) is a branch of AI that systematically applies algorithms with the aim of synthesizing the intrinsic and implicit relationships between data and information (Awad and Khanna, 2015). Mitchell et al. (1997) defines learning broadly, including any computer program that improves your performance on a task through experience. There are several ML algorithms, among which Artificial Neural Networks (ANNs), Support Vector Machines (SVMs), Random Forests (RFs) and Gradient Boosting (GB) stand out. This work will use ANN because it is a robust model that has already been used and studied by other authors in the constitutive representation of materials. However, other techniques will be object of study by the INSANE's machine learning research group, over time.

The use of ML, in science in general, has been spreading and consolidating over the last few decades, due to the development of computational infrastructure (software and hardware) and the increase in the amount of data available for training, thus enabling,

solving increasingly complex problems with increasing precision (Goodfellow et al., 2016). The ability to learn patterns based on experience and to generalize the acquired knowledge make ML models a promising tool for application in the search for more representative and general constitutive models. Therefore, this chapter aims to present the main concepts about ML algorithms, focusing on the technique of ANNs.

3.1 Classification of Learning Algorithms

There are some basic concepts that compose the learning mechanisms of ML models. There are the *learning paradigm* and the *learning task*. The learning paradigm refers to the environment in which the algorithm works, where learning can be carried out with or without the presence of a "teacher" (Haykin, 2009, Goodfellow et al., 2016). They are the *Supervised Learning*, *Unsupervised Learning*, *Semi-supervised Learning* and *Reinforcement Learning* algorithms.

The term supervised learning originates from the need to have a dataset in which each example represents an input signal and a corresponding expected answer as an output signal, used to compare the current result with the desired one and then, calculate an error response. In other words, the algorithm demands that the "teacher" show the target answer in order to perform a data input-output mapping.

In unsupervised learning, the algorithm must learn to interpret the data without a guidance of a "teacher". Unsupervised learning algorithms are designed to discover hidden structures pattern in unlabeled datasets, i.e., where the information regarding the desired output is unknown (Awad and Khanna, 2015). Here, the objective is to find useful and informative properties of the data available for analysis.

Semi-supervised learning algorithms use the combination of both concepts presented above, in a process in which a small number of labeled data and a large number of unlabeled data are needed to generate a model. This class of algorithms can provide a considerable improvement in learning accuracy (Awad and Khanna, 2015).

Finally, reinforcement learning algorithms can be seen as a trial-and-error learning paradigm of the Theory of Control, with rewards and punishments in line with the results of a metric of performance. This methodology involves the exploration of an adaptive sequence of actions or processes, by an "intelligent agent", in a specific situation, in order to maximize a given cumulative reward (Awad and Khanna, 2015). Learning also has an unsupervised character due to the absence of labeled examples of the function to be approximated.

The choice of which learning paradigm to use depends on the proposed learning task. This, in turn, is based on the type of problem to be learned and available data. Using ML algorithms in order to learn the constitutive behavior of concrete means using a function approximation task to solve regression problems. Among the available algorithms, the

one that will be used in this work and detailed in Section 3.7 is the Supervised Learning Algorithm of Artificial Neural Networks.

3.2 Capacity, Overfitting and Underfitting

It is known that during the training process of an ML model, the available data is divided into training data and test data. Details of the training process will be seen in Section 3.5. The power to accurately predict the behavior of a given set of test data based on a model fitted to a different set of training data is what differentiates ML models from simple optimizers. This ability of a model to successfully and precisely perform responses to unknown data is known as generalization (Awad and Khanna, 2015, Goodfellow et al., 2016).

To evaluate the performance of ML models, it is necessary to know and differentiate the concept of two types of errors. The error calculated based on the training data is known as training error or empirical error. The learning process is usually based on trying to reduce the training error. On the other hand, the prediction error of the same model, calculated based on the unknown dataset of examples, is defined as test error or generalization error. A ML algorithm achieves satisfactory performance when it is able to maintain a small difference between training and test errors, while calculating low values for training errors.

Goodfellow et al. (2016) defines the *capacity* of a model as the ability to fit a wide variety of functions. This concept is directly related to the number of parameters to be learned. There are two major challenges concerning ML models that are closely linked to the capacity, training error and generalization error concepts: *underfitting* and *overfitting*. Goodfellow et al. (2016) points out that underfitting occurs when the model is not able to obtain a sufficiently low error value on the training set. This implies that the model has not yet learned the patterns from the training data, and that there is still room for improvement. Underfitting can happen for the following factors: the model may simply not have been trained sufficiently or the model may not be powerful enough, that is, it may not have the necessary capacity to perform the desired approximation. In contrast, overfitting occurs when the gap between the training error and test error is too large (Goodfellow et al., 2016). In this case, the model learns patterns from the training data that will not generalize well the problem. Overfitting can arise from overtraining and/or defining a very complex model (high capacity) with respect to the dataset. Practically speaking, if a model is very simple for the data, the data distribution assumption is also very simple. An example would be a quadratic dataset for a model with linear approximation capability, characterizing underfitting. Conversely, if the model is too complex for the data, the data distribution assumption also becomes too complex, so that it would be the same as trying to fit the same set of quadratic approximation data,

this time to a model with a high polynomial approximation capability, characterizing overfitting.

Studies based on Statistical Learning Theory allow quantification of the capability of learning models. Goodfellow et al. (2016) points out that the main results of these studies (Vapnik and Chervonenkis, 1971, Vapnik, 1982, Blumer et al., 1989, Vapnik, 1995) show that the difference between the training error and the generalization error is limited from the top by an amount that increases as the model capacity increases, but decreases as the number of training examples increases.

The fact is that we should always use the simplest models for a given task, in accordance with the Principle of Parsimony[1], and the training dataset as completely as possible, contemplating all the features of the model that we want to recognize patterns. High-capacity models have the ability to represent highly complex problems as long as there is enough data to train it or the training is done to the right extent. To avoid overfitting, there are also, as a resource, regularization techniques, which restrict the parameters of the algorithm indirectly, in order to simplify the model, thus increasing the possibility of a good generalization. In the context of ANNs, the main regularization methods present in ML libraries are L1 and L2 Regularization, Early Stopping and Dropout. Details about these methods will not be covered in this dissertation, but may be found in the literature (Hoerl and Kennard, 1970, Tibshirani, 1996, Prechelt, 1998, Srivastava et al., 2014).

3.3 Bias/Variance Trade-off

The concepts of bias and variance are closely linked to the concepts of capacity, underfitting and overfitting, presented in Section 3.2. The generalization error, described earlier, is related to the estimator's future predictions and hence to the model's generalization capability. A model that does not make accurate predictions from new data has not achieved its purpose. Therefore, the success of a model is directly related to minimizing its generalization error. The Generalization Error consists of three distinct parts: the Bias Error, the Variance Error, and the Irreducible Error. Mathematically, this error can be described by Eq. (3.1).

$$Error(x) = Bias^2 + Variance + Irreducible\ Error \tag{3.1}$$

The Bias Error can be defined as the deviation between the expected results and predictions of the model relative to the training data. In other words, the bias error is related to a model's ability to fit the training data. The closer a model matches the training data, the smaller its bias. A model with a high bias does not learn from training

[1] widely known as Occam's razor, states that among all the hypotheses that equally explain a given problem, the simplest of all should be adopted

data and normally commits high training and testing errors.

However, the goal is not to have a too low bias such that the model memorizes the training data and does not make good predictions on the test data. In this sense, the Variance Error tells how much a model gets wrong when making predictions on top of the test data. The smaller the error, the smaller the variance.

The irreducible error, on its turn, is a residual error, which can not be reduced, and is a function of the noise in the training data. A portion of this error will always exist, regardless of the effort expended to minimize the generalization error.

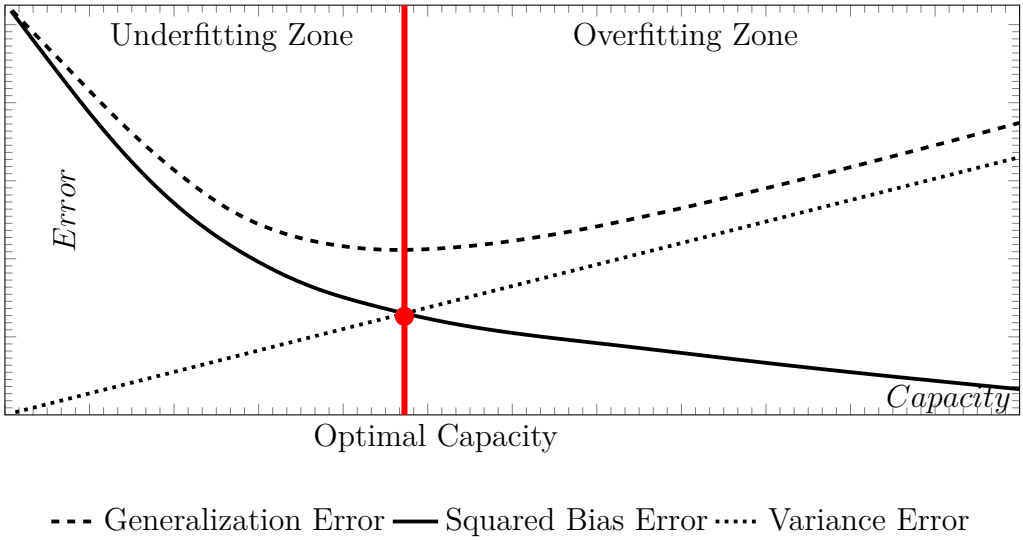


Figure 3.1: Trade-off between bias and variance - Typical relationship between capacity and error.

Fig. 3.1 presents a dilemma, decomposing the Mean Squared Error (MSE) of any ML model: A complex model, with high capacity, has a low bias error, but a high variance error, characterizing an overfitting. The opposite is also true: A shallower, low-capacity model has a higher bias error and a lower variance error, characterizing an underfitting. Therefore, there must be a trade-off between both so that the model neither underfits nor overfits. This balance point also indicates the optimal capacity of the model. Ideally, the model should have a low MSE, meaning that it has a low bias error and a low variance error. Another common way to negotiate this balance is through the process of cross-validation, which will be presented next.

3.4 Parameters and Hyperparameters

Machine learning algorithms have two distinct types of variables, known as parameters and hyperparameters. The parameters of a ML algorithm are the internal variables of the model. These variables can not be set manually and are defined throughout the algorithm's learning process. These variables are "learned" from the dataset used to train

the model, through optimization techniques, and are updated constantly and iteratively until the model be able to map the input and output data within an acceptable margin of error. These parameters are typically defined as random values at the beginning of the learning process or initialized with some special technique. At the end of the process, it is the parameters that characterize the model. In the case of ANNs, the parameters of the model are its synaptic weights[2].

The hyperparameters, in turn, can be divided into two groups: algorithm hyperparameters and model hyperparameters (Lakshmanan et al., 2020). In both, their values must be specified outside the training procedure. The algorithm hyperparameters are settings that control the behavior of the algorithm, directly affecting the speed and quality of training. These hyperparameters do not interfere with model predictions because they are not part of the model. Examples of algorithm hyperparameters are the learning rate, the batch size, the number of epochs, the loss function and the train-test split ratio. The model hyperparameters define the model's architecture, that is, they structure the model's hypothesis and refer to the model selection task. From this perspective, the model can be understood as a hypothesis that describes the problem and its hyperparameters as the adaptation of the hypothesis to a specific set of data. Examples of model hyperparameters are a ANN architecture (the number of hidden layer and the number of neurons in each layer) and the activation function. Usually, both types of hyperparameters are defined manually, before training begins, through previous experiences, trial and error, or even through heuristics. Furthermore, there are hyperparameter optimization techniques, such as GridSearch (which will be presented in Chapter 6), that can be used to choose the best set of hyperparameters for a particular problem. However, there are several possibilities of combinations between hyperparameters, which can make the search for the most suitable set very challenging.

More details about the parameters and hyperparameters of ANNs will be discussed in Section 3.7.

3.5 Validation Process

Training a machine learning model is an iterative process that aims to find an optimal set of model parameters to solve a given problem, through a predefined optimization technique and from a set of correction rules. First, one defines a statistically representative dataset of the problem. Next, one separates this data into training data and testing data. The training data is used in the learning process for model parameter tuning and the testing data to measure model's performance by evaluating the generalization error. It is extremely important that the test data is not used in the model training process. The literature suggests using 80% of the dataset available for training and only 20% for testing.

[2]which concept will be addressed in Section 3.7

However, if the dataset is not large enough, dividing it between fixed train data and fixed test data may become a problem, since a small test set can generate generalization errors whose differences are not statistically significant, thus making it difficult to choose the best model (Goodfellow et al., 2016). Furthermore, evaluating the model only once based on the test dataset is insufficient to conclude that the observed results did not occur solely by chance. In addition, it is known that the larger the amount of training data, the smaller the chances for overfitting to take place. These concerns have led to the creation of a technique known as *cross-validation*.

The cross-validation technique is a statistical tool that consists of randomly dividing the available dataset between training data and test data. The training data is then split between a training set and validation set. The training set is then used to train the model and the validation set to select the model by evaluating its performance. The test data is used at the end of the training process in order to evaluate the generalization error so as to avoid overfitting the validation subset. Separating a portion of the dataset to test data is necessary because of the importance of testing the ability of the model on unknown data, so the test data can be seen as a kind of 'final exam" of the model. This data is also known as hold-out data.

In the cross-validation process, the training data is used to train a few different models and the validation data is used to select the best models. Once the model with the best hyperparameter configuration is chosen, the model is re-trained on all the training data and evaluated on the test data. Ideally, several different data are used to train and validate the model in order to select the best fit. Nevertheless, if the amount of data is short or insufficient, one can use the technique known as k-fold cross validation, which redivides the training and validation data multiple (k) times, imagining that they are different samples. Thus, the model is trained k times with $(k-1)$ dataset, and the remaining dataset is used as the validation set. At each data rescaling, the model and training starts from scratch. At the end of training, the performance is evaluated as the average score between the k evaluations. A drawback of cross-validation is its high computational cost.

Fig. 3.2 illustrates the k-fold cross-validation process for a $k = 5$.

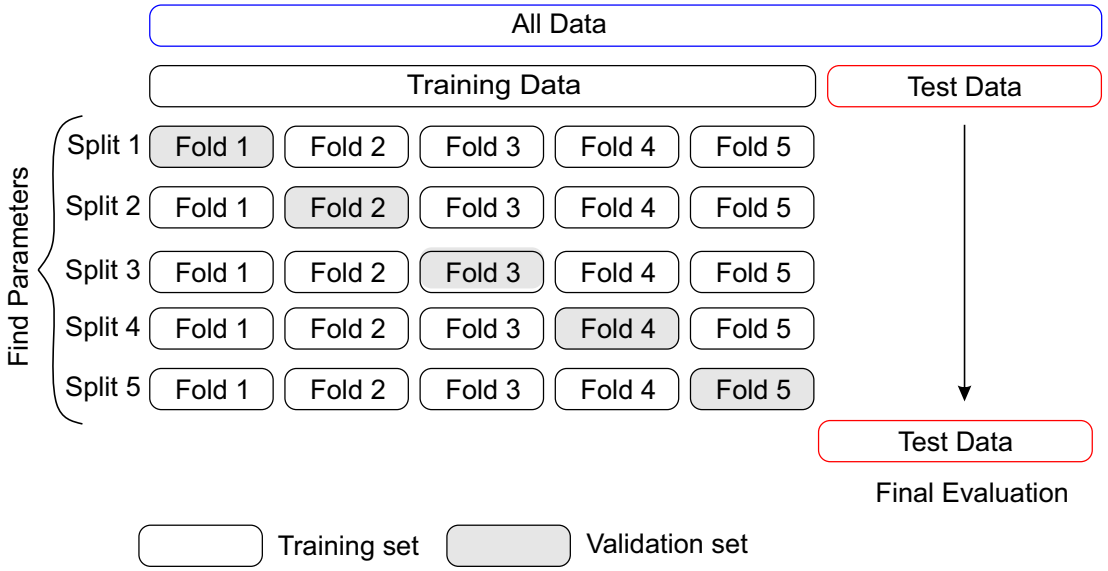


Figure 3.2: *K-Fold* method representation.

3.6 Metrics for Evaluating Results

In the conception phase of a ML model, it is necessary to define a metric for performance evaluation. This decision is usually based on the type of problem that is proposed to be solved and the level of accuracy that is intended to be achieved. This work focuses on solving a regression problem. Predictive regression modeling approximates a nonlinear function by performing an input-output mapping, in which the output variable of the model is a continuous value and dependent of model's features. Regression metrics involve calculating error scores based on the estimator's output variables in order to quantify the quality of the predictions. Then, the desired degree of accuracy is defined. Thus, all subsequent model building actions are taken with the goal of achieving this error rate. From the metric, it is possible to perform comparative evaluations between different models in order to choose the most suitable for the proposed problem. The main metrics for evaluating the results for regression models will be outlined in the following.

3.6.1 Mean Squared Error (MSE)

The Mean Squared Error (MSE) is one of the most widely used error metrics for evaluating regression problems. This metric can be used both for evaluating results and as a cost or loss function in training ML models. However, usually, the cost function is different from the metric for algorithm performance evaluation (Goodfellow et al., 2016).

The MSE calculates the mean or average of the squared differences between predicted and expected values, as per Eq. (3.2),

$$MSE(y, \overline{y}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \overline{y}_i)^2 \tag{3.2}$$

where, n is the number of predictions; y_i is the i-th expected value; and $\overline{y}_i$ is the i-th predicted value.

An important observation regarding the metric being calculated as the square of the error is that the MSE will always be a positive number. A perfect model would calculate an MSE equal to zero, but this does not happen in practice. An advantage of the metric is that it makes it possible to detect outliers in the model, since it gives a larger weight to these errors as a result of the square part of the function. However, it penalizes large errors over smaller errors,"punishing" models more for large errors, when used as a metric or as a cost function. Besides that, evaluating the error with the unit squared can make the results difficult to interpret. One way around this problem is to define a baseline for the metric by evaluating the dataset with trivial prediction models.

Fig. 3.3 shows the variation of the MSE with the increase of the differences between the expected value and the value calculated by the model. For this, it was considered a set of expected values with unit values (1.0) and a set of predicted values ranging from 0.0 to 1.0, being 0.0 a very bad prediction and 1.0 a perfect prediction.

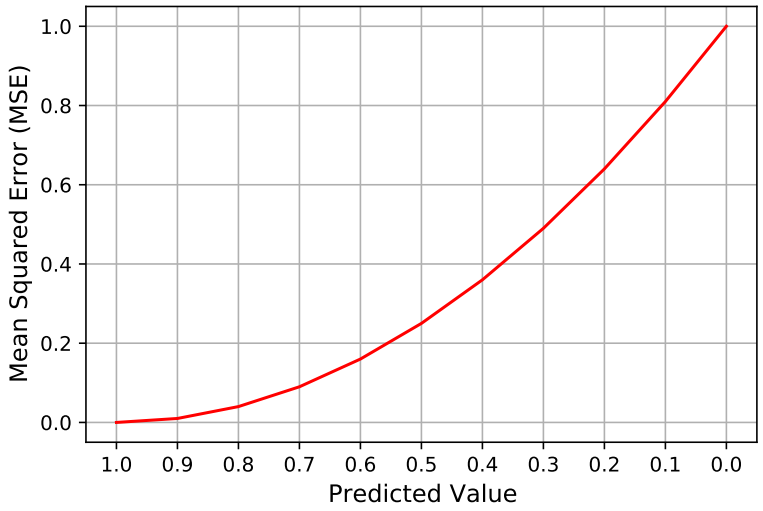


Figure 3.3: MSE behavior with the increase of prediction error.

The MSE is one of the most common cost functions used in ML model projects, due to its simplicity and efficiency, and because practical problems are not so much concerned with outliers as with whether the model is able to deliver good overall results. For this reason, MSE will also be the cost function adopted in this dissertation.

3.6.2 Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is also a popular technique for regression problems. The MAE computes the absolute mean error between the the expected values and the predicted values, as given in Eq. (3.3),

$$MAE(y, \overline{y}) = \frac{1}{n} \sum_{i=1}^{n} (|y_i - \overline{y}_i|) \tag{3.3}$$

where, n is the number of predictions; y_i is the i-th expected value; and $\overline{y}_i$ is the i-th predicted value. Just like the MSE, MAE will never be a negative number.

This metric is a good alternative to MSE because the error unit is the same unit as the target value; it does not penalize small and large errors in different proportions; and its value rises linearly with increasing error (as illustrates Fig. 3.4), which can be simpler to understand and facilitate interpretation of the results.

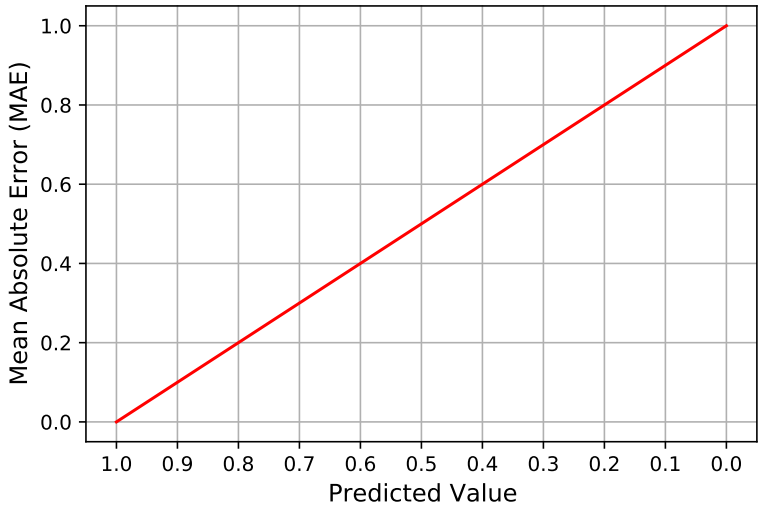


Figure 3.4: MAE behavior with the increase of prediction error.

3.6.3 Coefficient of Determination (R^2)

The Coefficient of Determination (R^2) represents how well the independent variables in the model explain the variability in the dependent variables or response variables. In other words, it gives an indication of the quality of the estimator's predictions, where, n is the number of predictions; y_i is the i-th expected value; $\overline{y}_i$ is the i-th predicted value; and $\widehat{y}$ is determined by Eq. (3.5).

$$R^2(y, \overline{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \overline{y}_i)^2}{\sum_{i=1}^{n} (y_i - \widehat{y})^2} \tag{3.4}$$

$$\widehat{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.5}$$

Since the R^2 coefficient depends on the dataset, it may not be significantly comparable across different datasets. The best value a model can achieve is 1.0, which would indicate perfect predictive ability. An R^2 equal to 0.0 means that the model always predicts constant values regardless of the input values, while negative values can occur and indicate completely inefficient models.

Despite being a widely used metric in the evaluation of machine learning models, the use of R^2 in nonlinear regression models is not a consensus and requires some cautions. First, because it was a metric originally intended for evaluating how well linear regression models perform. If we are dealing with a nonlinear regression problem, using R^2 alone may lead to hasty conclusions (Sapra, 2014). The ideal is always to use other metrics to guide the evaluation of the goodness-of-fit of the model. Second, because as the independent variables in the model are increased, the value of R^2 also increases, regardless of whether the added variable is significant or not, thus passing on a misleading statistic of performance improvement (Sapra, 2014). In other words, a model with more independent variables than another may perform falsely better simply because it has more variables. This may induce the user to add independent variables to the model, producing high R^2 values but low generalizability. In order to get around the second problem described above, the Adjusted R^2 is commonly used, as will be presented next.

3.6.4 Adjusted Coefficient of Determination (Adjusted R^2)

The Adjusted Coefficient of Determination (Adjusted R^2) considers, in the original metric, the influence of the number of model input parameters (features), as indicated by Eq. (3.6).

$$R^2_{adj} = \left\{1 - \left[\frac{(1-R^2)(n-1)}{(n-k-1)}\right]\right] \tag{3.6}$$

where, n is the number of predictions; k is the number of independent variables; and R^2 is the coefficient of determination of the model.

The Adjusted R^2 is considered a less biased metric than R^2 because it increases only if a new term improves the model more than would be expected at random and decreases when the addition of an independent variable improves the model less than expected at random. It is the ideal metric for comparing models with different numbers of independent variables.

3.7 Artificial Neural Networks

Artificial Neural Networks (ANNs) are neuroscience-inspired computer models that are named after similarities with the structure and operation of the central nervous system. Just like the central nervous system, the ANN is a dynamic nonlinear system formed by several interconnected neurons, also known as processing units, which perform a dense parallel processing of information, with the aim of performing a certain task, mapping input-output data in order to learn their relations, store the acquired knowledge, expand it from future experiences, and mainly, generalize it by means of unknown information (Haykin, 2009, Goodfellow et al., 2016, Ghaboussi, 2018). This great computational power characterizes ANNs as excellent tools for performing pattern recognition and function approximation tasks.

Fig. 3.5a illustrates the representation of a biological neuron, in a simplified form, and Fig. 3.5b, the first model of a binary artificial neuron, proposed in the 1940s by McCulloch and Pitts (1943).

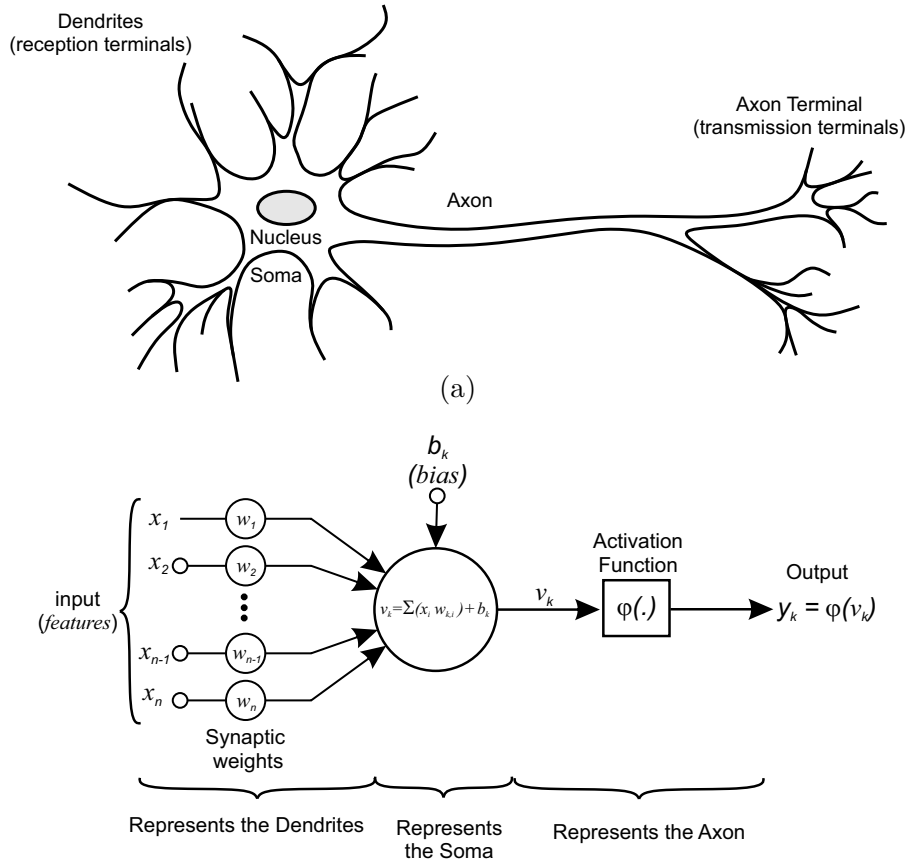


Figure 3.5: Neuron model. (a) A simplified schematic representation of a biological neuron, (b) McCulloch and Pitts (1943) model of a binary artificial neuron labeled j.

The linear combination that happens in any given j neuron can be written according to the following equation:

$$v_j = \sum_{i=1}^{m} w_{ji} x_i + b_j \tag{3.7}$$

where w_{ji} is the synaptic weight, or strength, and the first subscript refers to the neuron that receives the message and the second subscript to the connection from which the signal came; x_i is the input signal applied in the input layer; b_j is the bias and v_j is the induced local field, or activity.

The output signal of each neuron is calculated by applying the activity v_j to the respective activation function $\varphi(.)$. The activation function, in turn, has the role of restricting the output of each neuron and is considered the main source of the algorithm's nonlinearity. In the case of the binary neuron of McCulloch and Pitts (1943), the activation function ranged between 0 or 1 and determined whether the neuron generated a response signal or not. Subsequently, other types of activation functions emerged, which will be discussed in Section 3.7.8.

ANNs are formed by the connection of a set of neurons, whose interaction explains the complexity of the behavior of this system. ANNs are capable of learning, through a learning process or algorithm, and storing the acquired knowledge through their synaptic weights. The initial model of the neuron proposed by McCulloch and Pitts (1943), however, was unable to accumulate knowledge in front of new examples since the synaptic weights were not dynamically adjusted. This limitation motivated the proposition of a new model known as Rosenblatt's Perceptron, which has as its main characteristic the limited architecture of one input and one output layer (single-layer feedforward neural network). This new approach, developed by Rosenblatt (1958), enabled the algorithm to dynamically update its synaptic weights based on errors calculated from target values. Later, other algorithms used for training artificial neurons were developed. Among them is the Least Mean Square (LMS), developed by Widrow and Hoff (1960), the basis of the algorithm that would be used for multilayer perceptrons.

In 1969, Minsky and Papert (1969) proved mathematically that Rosenblatt's perceptron was not capable of performing global generalizations, and so its employment could only be done on a limit set of problems. Minsky and Papert (1969) also proved that the perceptron's shortcomings extended even to its variations, such as multilayer perceptrons (MLP) or multilayer feedforward neural networks (MFNN), which differ from perceptrons due to the addition of hidden layers in the model structure. These conclusions stalled advances in the field until the mid-1980s.

Subsequent advances have circumvented the problems pointed out in Minsky and Papert (1969) propositions. Rumelhart et al. (1986) presented in their publication one of today's most used learning algorithms, known as the *backpropagation error algorithm*.

The backpropagation algorithm together with the studies of Hornik et al. (1989) boosted the MLPs studies and applications, which extend to the present days. The Universal Approximator Theorem (Hornik et al., 1989), established that a neural network with at least one hidden layer and enough neurons, and with a squashing activation function, is enough to approximate any continuous function to any desired degree of accuracy. In this line of reasoning, for whatever the proposed problem is, there is an ANN that is capable of approximating the objective function. Despite the previous statement, some reasons may prevent the network from learning, such as the optimization algorithm not being able to find the optimal model parameters or training algorithm choosing the wrong parameters due to overfitting (Goodfellow et al., 2016).

To proceed with this work, it is necessary to present the main concepts related to Rosenblatt's perceptron, MLPs, the LMS method, and the main learning algorithm used in ANNs, the Backpropagation algorithm. Next, a summary will be presented on the main hyperparameters present in the training and design of an ANN.

3.7.1 Perceptron

The *Perceptron* algorithm is an artificial neural characterized by having an input layer with the features of what is to be learned and an output layer, with the network's responses. Also known as single-layer feedforward neural networks, they are limited to solving linearly separable pattern classification problems. Linearly separable problems represent a hyperplane decision surface in space. The algorithm answers the value of 1 for elements on one side of the hyperplane and answers the value of -1 for elements on the other side of the hyperplane. When the Perceptron has only one neuron in the output layer, it is limited to solving only two-hypothesis problems. However, the addition of neurons in the output layer allows the classification of problems with more than two classes, as long as they are also linearly separable classes. Its operation is based on an error correction learning model. The training process consists in adjusting the synaptic weights, according to Eq. (3.8), until the model performs the proposed classification adequately.

$$w_i(n+1) = w_i(n) + \Delta w_i(n) \tag{3.8}$$

where,

$$\Delta w_i(n) = \eta\,[y(n) - \bar{y}(n)] \tag{3.9}$$

In Eq. (3.9), $y(n)$ refers to the *target value* for example n; $\bar{y}(n)$ is the output signal of the Perceptron for example n; and η is a positive dimensionless constant, defined as the *learning rate parameter*. The learning rate is intended to control the intensity of the adjustments of the synaptic weights at iteration n and will be discussed in more detail in

Section 3.7.7.

The *Learning Convergence Theory* presents evidence for how well the Perceptron algorithm works with respect to linearly separable problems. Inspired by the Perceptron, Widrow and Hoff (1960)) proposed an algorithm known as the The Least Mean Square (LMS) or Delta Rule, the first linear adaptive filter. This rule was used to formulate the ADALINE (Adaptvie Linear Neural Network) model and the MADALINE (Multiple-Adaline) (Widrow, 1962), a neural network with more than one Adaline. While Perceptron uses the class labels to learn the model coefficients (the binary response), Adaline uses continuous predicted values to learn the model coefficients, and is therefore a more powerful technique. The LMS would later become the basis of the backpropagation algorithm (Haykin, 2009). Both concepts will be discussed next.

3.7.2 Least-Mean-Square Method

For the development of the Delta Rule formulation, first an optimization technique is defined and then a cost function $\mathscr{E}(W)$. The cost function has the task of using the error to control the process of adjusting the synaptic weights. The goal of the algorithm is to minimize this cost function $\mathscr{E}(W)$ with respect to the vector of synaptic weights W, in order to obtain the vector W that best represents the relation between the input and output data. To this end, the cost function needs to be differentiable with respect to the vector W.

In this way, an optimal solution that satisfies the following condition is sought:

$$\mathscr{E}(W^*) \leq \mathscr{E}(W) \tag{3.10}$$

where W^* is a vector of synaptic weights distinct from a second vector of synaptic weights W.

To optimality, the following condition need to be satisfied,

$$\nabla\mathscr{E}(W^*) = 0 \tag{3.11}$$

where, ∇ is gradient operator given by

$$\nabla = \left[\frac{\partial}{\partial w_1}, \frac{\partial}{\partial w_2}, ..., \frac{\partial}{\partial w_m}\right]^T \tag{3.12}$$

By interpreting the cost function $\mathscr{E}(W)$ as an error surface, the gradient $\nabla\mathscr{E}(W)$ represents the direction in which this function grows fastest. In this context, the correction of the synaptic weights at iteration n is given by

$$\Delta W(n) = -\eta\nabla\mathscr{E}(W(n-1)) \tag{3.13}$$

or in the form of components

$$\Delta w_i(n) = -\eta \frac{\partial \mathscr{E}(W(n-1))}{\partial w_i(n-1)} \tag{3.14}$$

The adjustments applied to W in the direction opposite to the growth of the cost function $\mathscr{E}(W)$ (negative sign of the Eq. (3.14)), characterize the Gradient Descent optimization technique.

Finally, the Delta Rule is given by

$$w_i(n) = w_i(n-1) - \eta \frac{\partial \mathscr{E}(W(n-1))}{\partial w_i(n-1)} \tag{3.15}$$

3.7.3 Multilayer Perceptron

The *Multilayer Perceptron* (MLP), also known as Multilayer Feedfoward Neural Network (MFNN), are ANNs characterized by having an input layer, one or more hidden layers, and an output layer. In a MLP, each neuron inside a layer connects with those in the next layer, but the neurons in the same layer does not have a connection with each other. Furthermore, each connection processes a synaptic weight, that stores the knowledge of the computational model and works as a filter between correlated neurons. Fig. 3.6 illustrate this model.

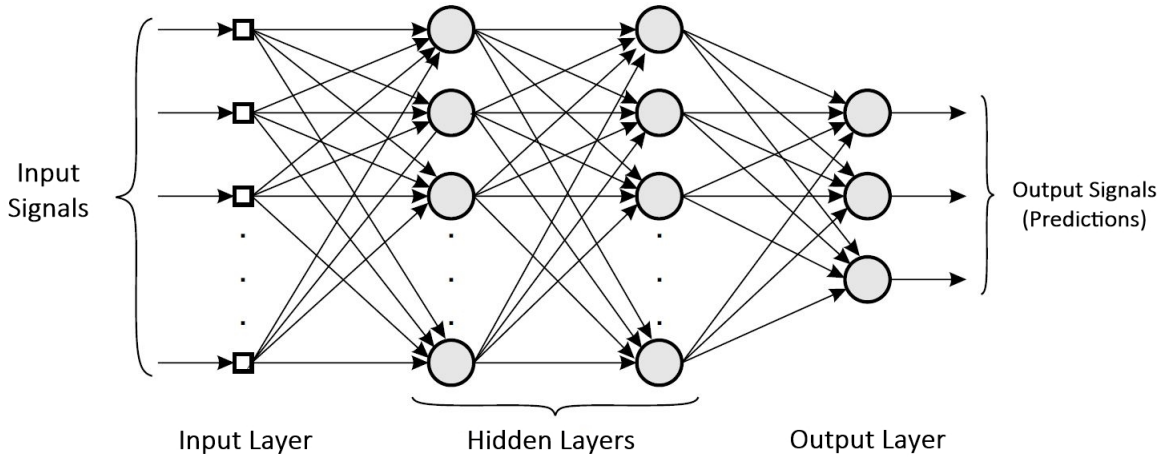


Figure 3.6: MLP representation.

In an MLP, each neuron in the hidden layer or output layer performs two actions: First, it processes an output signal, which is given through a continuous function of its activity. Second, it computes the gradient of the error surface with respect to the synaptic weights connected to the neuron's inputs, for the purposes of the error backpropagation process (Haykin, 2009).

3.7.4 Backpropagation Algorithm

The learning rule applied in a MLP is the *backpropagation algorithm*. This supervised learning process occurs in two steps: the forward pass, where the input signal is sent through the network until its output layer and error signals are calculated; and the backward pass, that is responsible for the modification of the synaptic weights based on the calculation of this error signals. In the backward pass, it is important to point out that each neuron is held accountable in the right proportion of its "guilt" for the calculated error, according to its connection strength.

Just like in the LMS method, the backpropagation algorithm has as main objective to minimize the error (cost function) in order to find the synaptic weight vector W that best approximates a certain unknown function, through an unconstrained optimization technique.

To facilitate mathematical manipulation for the backpropagation process, it is necessary to perform a small adaptation in the model shown in Fig. 3.5b. This adaptation consists in including in the input an additional value equal to 1 so that the associated synaptic weight plays the role of the bias. So, considering that the i, j, k and l indexes refer to distinct neurons in the network so that the signals propagates in the network from left to right, Eq. (3.7) can be rewritten for a neuron k in the middle of the network, in the n_{th} iteration as

$$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n) \tag{3.16}$$

The function signal of neuron k is calculated as

$$y_k(n) = \varphi_k(v_k(n)) \tag{3.17}$$

where φ_k is neuron's k activation function.

Just like in the Delta Rule, based on the Gradient Descent method, and in the derivation of the MSE as the cost function with respect to the synaptic weights, the variation in synaptic weights from one iteration to the next can be evaluated as

$$\Delta w_{kj}(n) = -\eta \frac{\partial \mathscr{E}(n)}{\partial w_{kj}(n)} \tag{3.18}$$

The determination of the partial derivative $\frac{\partial \mathscr{E}(n)}{\partial w_{kj}(n)}$ can be distinguished in two different cases, depending on whether the network neuron k is in the output layer or in the hidden layers. For the former, for a k neuron in the output layer, its cost function is a function of the error. Hence it has

$$\frac{\partial \mathscr{E}(n)}{\partial w_{kj}(n)} = \frac{\partial \mathscr{E}(n)}{\partial v_k} \frac{\partial v_k}{\partial w_{kj}} \tag{3.19}$$

Substituting Eq. (3.16) into Eq. (3.19) gives

$$\frac{\partial \mathscr{E}(n)}{\partial w_{kj}(n)} = \frac{\partial \mathscr{E}(n)}{\partial v_k} y_i \tag{3.20}$$

Applying the chain rule to the derivative $\frac{\partial \mathscr{E}(n)}{\partial v_k}$, one has that

$$\frac{\partial \mathscr{E}(n)}{\partial v_k} = \frac{\partial \mathscr{E}(n)}{\partial y_k}\frac{\partial y_k}{\partial v_k} = \frac{\partial \mathscr{E}(n)}{\partial y_k}\varphi_k'(v_k) = \delta_k \tag{3.21}$$

where the term δ_k represents the local gradient of the k neuron.

By substituting the Eq. (3.21) into Eq. (3.20), one has that

$$\frac{\partial \mathscr{E}(n)}{\partial w_{kj}(n)} = \delta_k y_j \tag{3.22}$$

Finally,

$$\Delta w_{kj}(n) = -\eta \delta_k(n) y_j(n) \tag{3.23}$$

In hidden layers, the inference of $\frac{\partial \mathscr{E}(n)}{\partial w_{kj}(n)}$ is more complex, as it depends on the evaluation of the influence of the k neuron on the error of the output layer. In this case, the algorithm back-propagates the observed error in the hidden layer of the k neuron, as presented in Fig. 3.7.

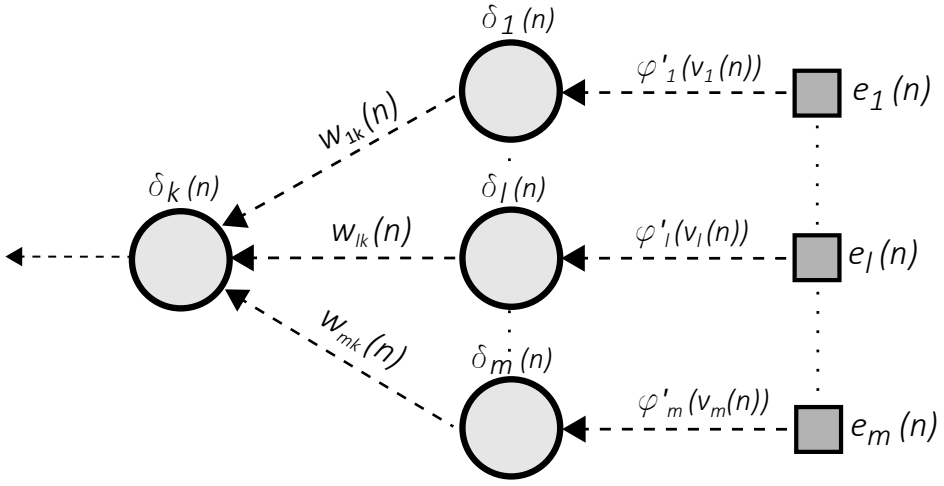


Figure 3.7: Error backpropagation.

Thus, the partial derivative $\frac{\partial \mathscr{E}(n)}{\partial v_k}$ is taken as

$$\frac{\partial \mathscr{E}(n)}{\partial v_k} = \sum_{l=1}^{m} \frac{\partial \mathscr{E}(n)}{\partial v_l}\frac{\partial v_l}{\partial v_k} = \sum_{l=1}^{m} \delta_l \frac{\partial v_l}{\partial v_k} \tag{3.24}$$

Given $v_l = \sum w_{lk}\varphi(v_k)$, it follows that

$$\frac{\partial v_l}{\partial v_k} = \frac{\partial \sum w_{lk}\varphi(v_k)}{\partial v_k} = w_{lk}\varphi'(v_k) \tag{3.25}$$

By replacing Eq. (3.25) with Eq. (3.24), one has that

$$\frac{\partial \mathscr{E}(n)}{\partial v_k} = \varphi'(v_k) \sum_{l=1}^{m} \delta_l w_{lk} = \delta_k \tag{3.26}$$

Finally, it is again

$$\Delta w_{kj}(n) = -\eta \delta_k(n) y_j(n) \tag{3.27}$$

The training of an ANN happens through an iterative process which involves the propagation of the training set and the adjustments of the synaptic weights. Each training set is composed by training examples and a complete presentation of the entire training set through the network is defined as an *epoch*. After an ANN is sufficient trained, or in other words, has learned the patterns of the desired data, the knowledge is stored in the synaptic weights and used in a testing process, that is needed to validate its generalization capability.

3.7.5 Data Scaling Process

The data originally available to train a ML model is usually not properly formatted to perform model training, and may be incomplete, noisy, not in a suitable format, and thus in need of preprocessing before the training phase begins. To improve the stability and performance of ANNs models, it is used *data scaling*, which is a form of data preprocessing that involves the use of techniques such as normalization and standardization to resize the input and output variables.

The input and output variables can have different units and therefore vary in scale. The differences in scales between the input variables can destabilize the learning process, since large input values may result in a model that learns large values of synaptic weights, slowing the convergence of the algorithm (low performance) and increasing the generalization error of the model (Brownlee, 2019). In parallel, since error is used in the learning process, via the backpropagation algorithm, scaleless target variables in regression problems may result in gradient explosion (large error gradient values), leading to synaptic weights changing dramatically and consequently failure of the learning process (Brownlee, 2019).

The scaling of input and output data tends to make the training process more stable, due to numerical optimization conditions. Optimization algorithms based on Gradient Descent are quite sensitive to data scaling (Sarle, 2002). Scaling data in ANNs smooths the gradient descent toward minima, making convergence faster, decreasing the risk of the process getting stuck in local optima, and improving the shape of the error surface. On the other hand, modifying the target data can impact the function one aims to find. Therefore, scaling should be taken with caution as it may discard information that may

or may not be important to the learning task (Sarle, 2002).

Scaling the input data can optimize training time and reduce the chances of the algorithm getting stuck in local minima (Sarle, 2002). The literature emphasises that the main motivations for scaling the input data of ANNs is to avoid saturation and eliminate the problem of scale dependence of the initial synaptic weights. It is known that better initializations will be obtained if the data is centered near zero and if most of the data is distributed in a range of approximately [-1,1] or [-2,2], since it is not good to have the data confined in a very small or very large range (Sarle, 2002).

The scaling of the output data is typically used to aid in the initial stabilization process of the weights. However, if the target value is a vector with more than one variable, and the cost function sensitive to scaling, such as the usual MSE, then the relative variability of each target value can affect the learning process (Sarle, 2002). Variables with a low range may have their importance in the learning process overshadowed by variables with a larger range. Regarding the output variables, one must first make sure that the variation of the expected data is within the domain of the activation function of the output layer. However, it is easier to adapt the output activation function to your data than the other way around (Sarle, 2002). The identity function is usually used for this purpose.

The normalization process is about scaling the original data between predefined limits. This process requires knowing the minimum and maximum values of the variables in the dataset. A pertinent concern, in the case of stress-strain data, is to ensure that test data or data acquired later are within the pre-set limits for training. It is important to look for a way to increase the maximum tensile and compressive stresses and strains by a given value in order to try to cover possible values that may extrapolate the training data in the future.

The standardization process, in turn, consists of rescaling the data value distribution so that the mean is zero and the standard deviation is one. The standardization process assumes that the data has a normal (Gaussian) distribution. Otherwise the data can still be standardized, but the results may not be reliable (Brownlee, 2019).

3.7.6 Optimization Technique

The *optimization techniques* aim to find the optimal parameters that minimize or maximize an objective function. In ANNs, these functions should be continuous, and the values of the input and output data should be real numbers, e.g. floating point values. Optimization algorithms are used to adjust the synaptic weights of neural networks iteratively based on training data, minimizing the network's output error, in order to find the vector of weights that best explains the relationship between the input data and the target data. Gradient Descent based algorithms, also known as First Order algorithms, are used in ANNs. This class of algorithms gets this name because they search for the

optimal solution to the problem in the opposite direction of the first derivative of the function to be minimized (in this case the cost function), i.e., follows the direction of the slope of the surface created by the objective function downhill until it reaches a valley (Ruder, 2016). Put another way, the gradient shows how a small change in the input features of the network impacts its outputs.

There are three variations of the gradient descent method, whose differences lie in the amount of data that is used to calculate the gradient of the objective function. Briefly, the gradient calculation can be performed on every sample (stochastic), over the entire dataset (batch) and over every given number of data (minibatch). Stochastic Gradient Descent (SGD) - first described by Robbins and Monro (1951), is considered a good learning algorithm for training neural networks with large datasets. However, it tends to oscillate in areas where the surface curves much more abruptly in one dimension than in another, a common situation around local optima (Sutton, 1986). In order to overcome this problem, Qian (1999), proposed a method known as Momentum or SGD with Momentum, which adds a "momentum" term α in Eq. (3.27) that speeds up the gradient descent process in the correct directions, leading to faster convergence. This term defines a contribution from previous iterations in updating the weights of the current iteration, according to Eq. (3.28).

$$\Delta w_{kj}(n) = \alpha \Delta w_{kj}(n-1) + \eta \delta_k(n) y_j(n) \tag{3.28}$$

where α assumes values in the range: $0 \leq |\alpha| \leq 1$.

The SGD maintains a single learning rate for all weight updates and the learning rate does not change during training. There are also variants of this method that propose an adaptive variation of the learning rate. Among these methods, the most widely used and often taken as standard in machine learning libraries is the Adaptive Moment Estimation (Adam). Adam is defined by its authors in Kingma and Ba (2015) as *"a method that computes individual adaptive learning rates for different parameters of first and second moment gradient estimates"*. Adam combines the advantages of other SGD variations such as Adaptive Gradient Algorithm (AdaGrad), published by Duchi et al. (2011) and Root Mean Square Propagation (RMSProp), an unpublished optimization algorithm first proposed by Geoff Hinton, both proposals with adaptive learning rate variation, which is computationally efficient, requires little memory, enables faster convergence and is more suitable for problems with a large number of data and parameters.

Mathematically, the proposition is given by:

$$\Delta w_{kj}(n) = -\frac{\eta}{\sqrt{\hat{p} + \epsilon}} \eta \delta_k(n) \hat{m}(n-1) \tag{3.29}$$

where

$$\hat{m}(n-1) = \frac{m(n-1)}{1-\beta_1(n-1)}$$
$$\hat{p}(n-1) = \frac{p(n-1)}{1-\beta_2(n-1)} \tag{3.30}$$

and

$$m(n-1) = \beta_1 m(n-2) + (-\beta - 1)\,\delta_k(n) y_j(n)$$
$$p(n-1) = \beta_1 p(n-2) + (-\beta - 2)\,(\delta_k(n) y_j(n))^2 \tag{3.31}$$

The authors propose default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ, where β_1 is the exponential decay rate for the 1^{st} moment estimates, β_2 is the exponential decay rate for the 2^{st} moment estimates and ε a small constant for numerical stability.

Although Adam is the most widely used method in practice, studies show that SGD with momentum may still be the most suitable for certain datasets, such as that of Desai (2020). The point is that it is important to evaluate which version of the algorithms based on Gradient Descent is the most suitable for the database you want to work with. Details on the other variants of the Gradient Descent based optimization algorithms can be found in Ruder (2016) and Kochenderfer and Wheeler (2019).

3.7.7 Learning Rate and Batch-size

The tuning process of the hyperparameters of a neural network is not trivial and requires a good understanding of the subject, as well as a lot of trial and error. The algorithm hyperparameters of the *learning rate* (η) and the *batch size* have a significant influence on the convergence of the optimization algorithms presented in the previous section.

The learning rate controls the step size of the search for the optimal solution in the possibility space of the problem. The value of the learning rate tells how much the synaptic weights move in the direction of the gradient. For small η values, the w variations are small from one iteration to another, and thus training is more reliable, but slower. For large η values the algorithm converges faster, but large w changes can make the network unstable (oscillatory), since the weight corrections can be so large that the optimizer skips the minimum point of the cost function (Ruder, 2016). Furthermore, as presented in the previous section, the learning rate can be constant or time-varying throughout the training process, depending on the optimization algorithm.

The batch size, in turn, represents the number of training examples used in one iteration of the learning process to correct the algorithm parameters. Training can occur in batches, mini-batches, or sequentially. For the first case, the batch size is equal to the total number of the dataset, and one iteration is equivalent to one epoch. In the second situation, the batch size is greater than 1 and less than the number of the total exam-

ple set. Finally, sequential training is where correction of synaptic weights occurs after each training example is propagated through the network. When updating the synaptic weights of the model after processing all the training data, i.e., at every epoch, learning is slower, and the entire training data often does not fit in the computer's memory. On the other hand, by updating the model parameters after processing each instance, the model updates are too noisy and the process is not computationally efficient.

Therefore, choosing the learning rate and setting the batch size for training ANNs is a trade-off between having faster model learning and memory efficiency, and having accurate updates and computational efficiency. The fact is that it is necessary to find a balance between these two variables. Variations of the gradient descent optimizer are strategies that are mainly aimed at overcoming the difficulty of defining a suitable learning rate.

3.7.8 Activation Functions

Another important hyperparameter in the convergence of the backpropagation algorithm is the *activation function*. The activation function is responsible for how the weighted sum of the input of a node is transformed into its output, that is, for the nonlinear transformation of the process, being accountable for the ability of networks in solving complex problems. The purpose of the activation function is also to limit the amplitude of the output of each neuron in order to prevent small changes in parameters from causing large changes in the network output and to define whether or not a neuron should be activated.

In developing the training algorithm, it was observed that it is necessary to know, for each neuron, not only its activation function, but also its derivative. This means that the function must be continuous and differentiable at all points in its domain. In this context, there are several types of activation functions with this characteristic, commonly used in MLPs. In the following, the main activation functions will be presented.

Logistic Function (Sigmoid)

It is a nonlinear sigmoidal function, which takes on values between the ranges 0.0 and 1.0. For small input values, it results in output values closer to 0.0. For large input values, it results in output values closer to 1.0. Using this function can cause problems with gradient vanishing during network training.

$$\varphi(v) = \frac{1}{1 + e^{-v}} \tag{3.32}$$

$$\varphi(v)' = \varphi(v)(1 - \varphi(v)) \tag{3.33}$$

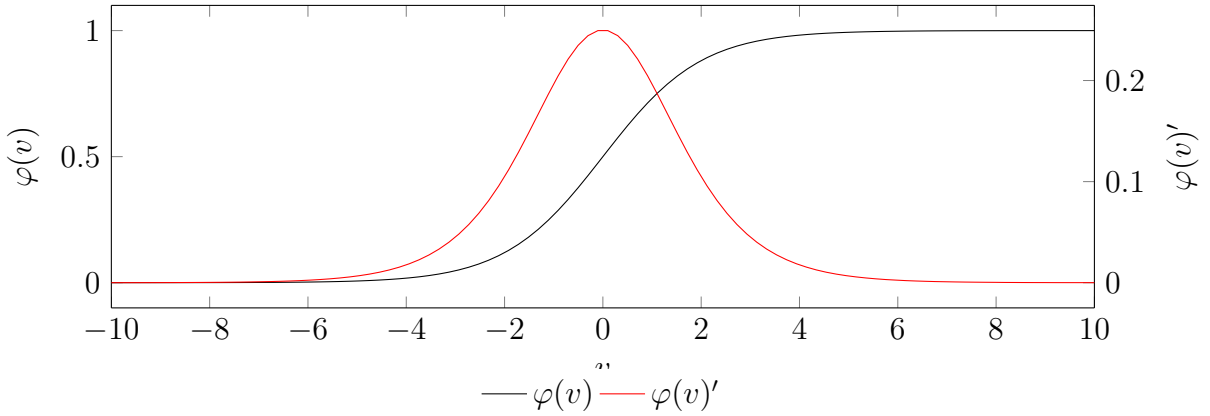


Figure 3.8: Sigmoid function and its derivative.

Hyperbolic Tangent Function (Tanh)

Like the logistic function, it is a nonlinear sigmoidal function, which takes on values between the ranges of -1.0 and 1.0. For small input values, it results in output values closer to -1.0. For large input values, it results in output values closer to 1.0. Although it performs better, it has the same gradient vanishing problem as the logistic function.

$$\varphi(v) = \frac{2}{1 + e^{-2v}} - 1 \tag{3.34}$$

$$\varphi(v)' = 1 - \varphi(v)^2 \tag{3.35}$$

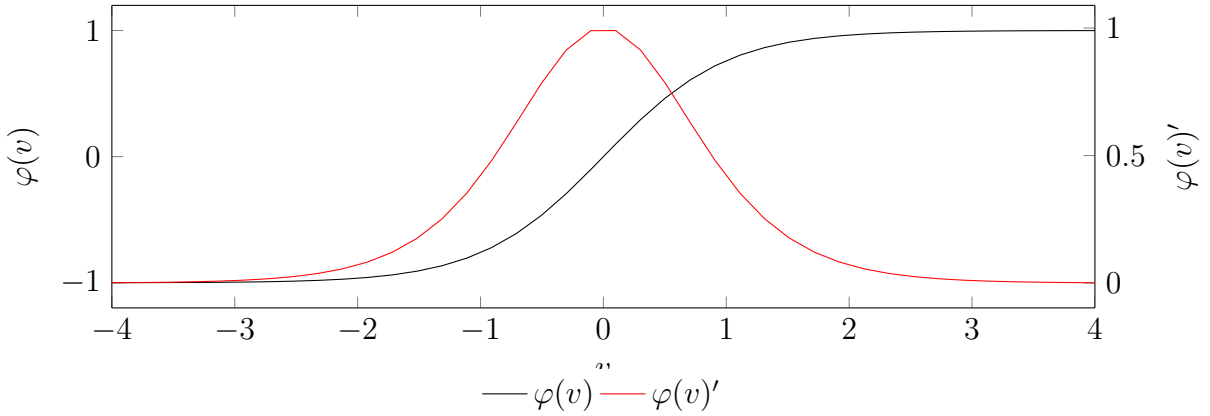


Figure 3.9: Hyperbolic tangent function and its derivative.

Rectifier Linear Function (ReLu)

It is the most popular activation function today. It stands out because, in addition to its simplicity, it is less susceptible to gradient vanishing, in contrast to sigmoidal functions. One point of attention is the possibility of neuron saturation.

$$\varphi(u) = max(0, v) \tag{3.36}$$

$$\varphi(v)' = \begin{cases} 0 & se \quad v < 0 \\ 1 & se \quad v \geq 0 \end{cases} \tag{3.37}$$

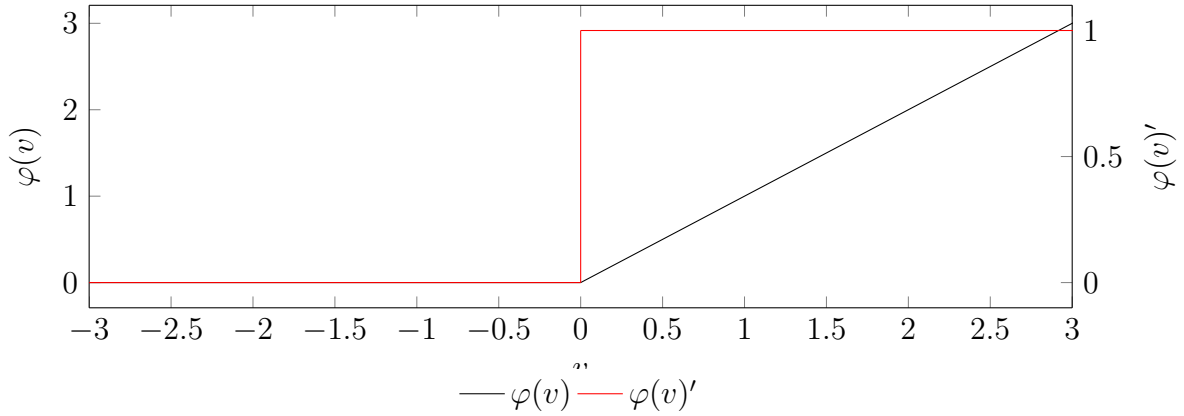


Figure 3.10: ReLu function and its derivative.

Linear Function

The linear activation function is also known as the identity function, because it does not change the activity of the neuron, as seen in the equation below.

$$\varphi(v) = v \tag{3.38}$$

$$\varphi(v)' = 1 \tag{3.39}$$

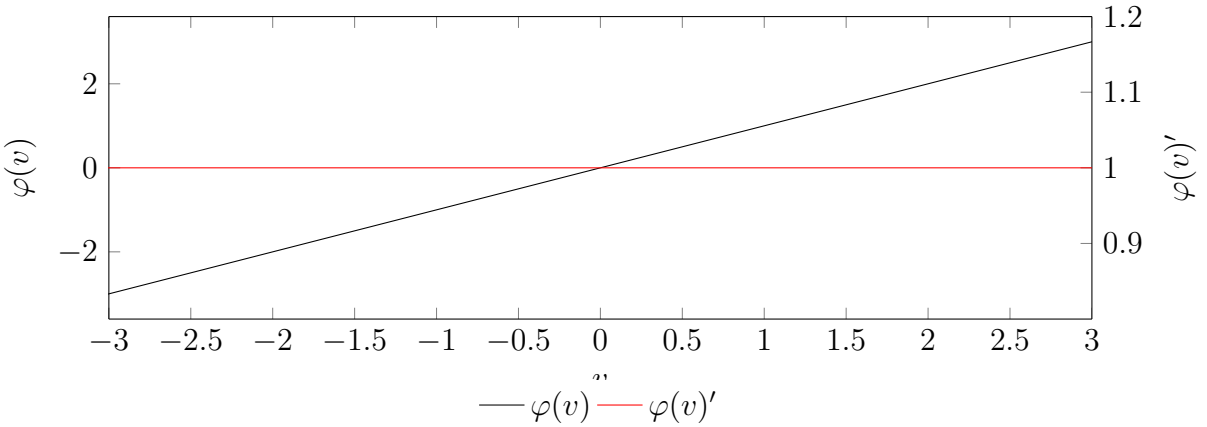


Figure 3.11: Linear function and its derivative

In regression problems, usually Sigmoid, Tanh and ReLu functions are used in the hidden layers of networks, while Linear is used in the output layer. Furthermore, it is usual to use a single function for all hidden layers. For regression problems, which is the case of this work, the default activation function used in the hidden layers is ReLu, and the activation function used in the output layer is Linear. The use of the linear function in the output layer typically uses data scaling techniques, as discussed in Section 3.7.5.

3.7.9 Architecture Design

As seen earlier, defining the capacity of a ML model is a challenging task, since models with low capacity are unable to learn satisfactorily, and models with high capacity tend to decorate the training data and obtain high generalization errors. In ANNs, capacity is related to the number of hidden layers in the model, known as depth; and the number of neurons in each hidden layer, defined as width. Another important aspect to the network's capability is the way neurons in each layer are connected to each other. The depth, width, and how neurons are connected to each other determine the architecture of artificial neural network. Typically all neurons in a neural network are interconnected with each other, and the main challenge when designing the architecture of a ANN is to define the number of neurons and hidden layers.

It is well known that a model with large capacity is capable of learning more complex functions, and may be even more computationally efficient. However, networks with many hidden layers have vanishing gradient problems, which complicate the training process, and are also more prone to overfitting.

In light of the aforementioned, some approaches can be used to help define the architecture of a neural network, since the number of hidden layers and neurons are model hyperparameters that can not be calculated analytically. The first is through experimentation, or trial and error. Performing controlled experiments can help in finding the most suitable architecture for the problem at hand. Also, it can be valuable to find in the literature the definition of architectures for similar problems. Using the configurations from other works as a starting point is one way to begin the search for the ideal model. However, pioneer works may face difficulties in finding initial network architectures that may be used, as happens in ML application in structural engineering. In fact, the same problem applies to all hyperparameters discussed in this dissertation. Usually, the literature does not present justifications for the choice of variables in an ANN, nor the difficulties faced in their definition. A good alternative is to use codes developed to perform systematic tests between models with the most diverse combinations of hyperparameters. These automated and exhaustive search strategies will be the object of study of this work, as a support in choosing the best hyperparameters of an MLP for the constitutive representation of concrete.

Chapter 4

Neural Network-based Constitutive Model

This chapter presents the main concepts related to neural network-based constitutive modeling, the state of the art of neural network application in structural engineering, and a critical analysis of the training data acquisition method used in this work.

4.1 Basic Aspects

The specification of a constitutive model that represents quasi-brittle materials, such as concrete, is an integral part of the structural analysis process using finite elements. To date, several mathematical models have been developed and improved for the most varied structures. However, the constraints arising from the definition of hypotheses, from the development of mathematical expressions and rules, and from the estimation of parameters that in most cases are difficult to obtain in practice, motivate the search for more general approaches to correlate stresses and strains, regardless of the structural context to which the material is subjected. In this scenario, the use of ML models such as Artificial Neural Networks presents itself as a potential alternative for the constitutive modeling of concrete, since they have the ability to learn complex nonlinear relationships between the features and targets due to the presence of activation function in each layer.

In general, the application of neural networks in engineering problems can be represented by Eq. (4.1):

$$\{output\} = \mathbf{NN}(\{input\} : \{architecture\}) \tag{4.1}$$

The left hand side of the equation represents the output vector and the right hand side the input vector and the neural network architecture. The symbol **NN** introduces a neural network as the approximation function of the problem.

In the case of concrete constitutive modeling, the approximation function is the relationship between stresses and strains, and the training data is composed of current stresses and strains, historical stresses and strains, and also stresses and strains increments, evaluated at a given observation point. In this sense, if the input data are strains increments and the output data are stresses increments, the model is known as a *strain-control model*. On the other hand, if the input data are stresses increments and the output data are strains increments, the model is characterized as a *stress-control model*. In addition, the number of historical data to be used in training the model depends on the characteristic of the material and the problem to be represented (Ghaboussi et al., 1991, Wu, 1991). Complex materials such as concrete, which suffers from the softening phenomenon, usually require one or more historical stresses states as input data. A MLP properly trained with such data can be defined as a *Neural Network-based Constitutive Model* (NNCM).

The most suitable scheme for use in finite element modeling and the one that it was used in this work is the strain-control model. A typical mathematical representation that contains the current stresses and strains, denoted by subscript n, and m previous values along the loading path is demonstrated in the Eq. (4.2). Here, based on the current stresses and strains, historical stresses and strains from previous points on the equilibrium path, and strain increments, the neural network predicts the stress increments that added to the current stresses will be the next stress states. The equation is generic and does not represent a specific neural network architecture.

$$\begin{aligned}\{\Delta\sigma\} = \mathbf{NN}(\{\Delta\varepsilon\}, \{\sigma\}_n, \{\varepsilon\}_n, \{\sigma\}_{n-1}, \\ \{\varepsilon\}_{n-1}, ...\{\sigma\}_{n-m}, \{\varepsilon\}_{n-m} :)\end{aligned} \tag{4.2}$$

4.2 State of the Art

The application of ANNs in concrete constitutive modeling was initiated in the 1990s, with the publications of Ghaboussi et al. (1990), Ghaboussi et al. (1991) and Wu (1991). In their papers, the authors modeled, via ANNs, the behavior of plain concrete in a plane stress state under monotonic biaxial loads. Eq. (4.3) represents a model composed of 4 layers, being an input layer, with the current stress-strain vector and the strain increment vector, each one with two components; an output layer with the stress increment vector, also with two components; and 2 hidden layers each with 30 neurons, proposed by Wu (1991). Fig. 4.1 shows schematically the neural network architecture.

$$\{\Delta\sigma\} = \mathbf{NN}\left(\{\Delta\varepsilon\}, \{\sigma\}_n, \{\varepsilon\}_n : 6 \mid 30 \mid 30 \mid 2\ \right) \tag{4.3}$$

In the same work, Wu (1991) trained an ANN to simulate the behavior of plain concrete

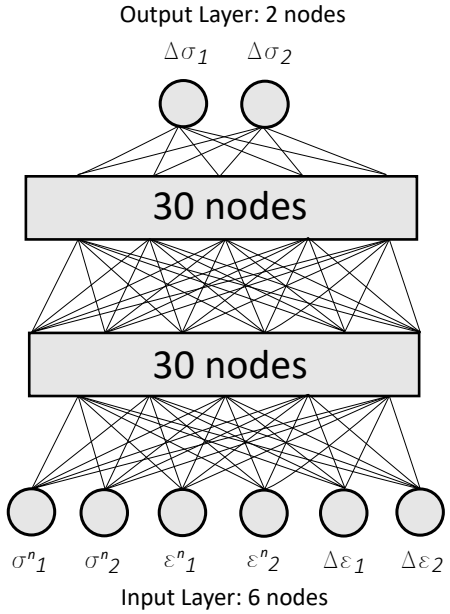


Figure 4.1: Wu's neural network architecture for concrete under monotonic biaxial loads.

under uniaxial cyclic compressive loading. Due to the complex nature of the problem, in order to the network to learn different loading and unloading curves, it was necessary to input two previous points on the stress-strain curve in the input layer. Wu (1991) also presents the application of neural networks in constitutive modeling of reinforced concrete. To represent the stress-strain behavior of composite materials, it is necessary to include as features material parameters or variables that characterize the behavior of the structure. In the case of reinforced concrete, the longitudinal and transverse yield strength of steel and the longitudinal and transverse reinforcement rates were used. In all the cases mentioned above, training data obtained from existing experimental results were employed.

Since the network architecture plays an important role in the performance of a back-propagation network, the need of its predetermination, usually done by trail and error, may be a drawback. Facing this issue, Wu (1991) proposed an adaptive method of architecture determination known as *adaptive architecture*. In a nutshell, in the method of adaptive evolution of network architecture, the training is initiated with a small neurons number in the hidden layers and as the network reaches its capacity, that is, the learning rate decreases at a certain level, new neurons are added with new connections in the hidden layers. Then, the connections of the initial neurons freeze and the training continues so that the new connections acquire knowledge based on the previous training. After that, neurons with frozen connections are released and the training proceeds normally. This process is repeated as long as necessary and at the end of the training, the network architecture is obtained automatically.

Ghaboussi and Sidarta (1998) presented a new model of neural networks, known as

Nested Adaptive Neural Networks (NANN). This new model was thought to deal with the nonlinear behavior of materials. However, its application is not restricted to material data. The concept of nested structures can be characterized by the ability to represent information from previous levels. An example of a nested structure is dimensionality. In structural analyses, the dimensionality refers to the number of components in the stresses and strains vector. A certain boundary value problem can me modelled in 1-D, 2-D plane stress and plane strain, axisymetric and 3-D domains. Strictly speaking, higher dimensional material tests are capable of depict all the lower dimensional cases. This new model was applied to experimental data from sands under drained and undrained triaxial tests.

One of the limitations of NNCMs is the need of a voluminous amount of material test data so that the model gives answers that truly represents the desired features and, beyond that, be able to generalize it for data that the network has not been trained on. From the perspective of material modeling, this means that it is necessary a lot of material tests results and it is known that in practice, this is not easy to get. To overcome this obstacle, Ghaboussi et al. (1998) first introduced a new method, named *Auto-progressive Training*. Based on the knowledge of structural results in the load-displacement form, from a control point, the auto-progressive training consists in simulating, via finite elements, two boundary problems, one in forces and the other in displacements, through the incremental-iterative process of Newton Raphson. The aim is to reproduce the equilibrium path of the experimental analysis point. Thus, a neural network pre-trained under linear results is used as the initial constitutive model, and each step that converges, new training stress-strain data is generated, the previous network is trained with these new data, and it is used to predict the stresses and strains in the new iteration. In this process, the representation capacity of the constitutive model increases progressively. In their publication, Ghaboussi et al. (1998) exemplified the algorithm by using a synthetic curve from a FEM model, which made use of a classical plasticity constitutive model, instead of an experimental curve.

Since then, several other works have been developed in order to model concrete and reinforced concrete. Jung and Ghaboussi (2006) studied an approach that uses stress rates and strain rates as input data in the training of networks for representing the behavior of reinforced concrete. The model has the potential to represent any material with rate-dependent behavior. Furthermore, the tests were performed on synthetic training data from visco-elastic models. Unger and Könke (2009) proposed the use of ANNs in the multiscale analysis of a reinforced concrete structure, through a homogenization technique. For training the proposed model, data from numerical simulations were also used.

The complexity of constitutive behavior is not exclusive to quasi-brittle materials. Geotechnical materials also have behavior that is difficult to equate, limited by assumptions necessary for formulating existing mathematical models and by the need for pa-

rameters that are not easy to obtain and often without physical meaning. Inspired by the work of Ghaboussi et al. (1991), Ellis et al. (1995) proposed the use of ANNs to model the stress-strain relationships of sands with different granulometric distributions and loading histories. The paper presented the use of both direct networks and recurrent[1] networks and showed that recurrent networks were more effective than traditional networks in the constitutive representation of sand. The training database was obtained from experimental tests via undrained triaxial tests on eight different types of sands.

In 1998, Sidarta and Ghaboussi (1998) presented a paper reporting on the training of ANNs for constitutive modeling of sands under triaxial tests with end friction. In contrast to conventional material tests, non-uniform tests generate a wide variety of stress and strain distributions within the sample, thus enabling a wealth of information about the material. In this work, the authors used NANNs (Ghaboussi and Sidarta, 1998) together with the auto-progressive training method (Ghaboussi et al., 1998).

Following this, Penumadu and Zhao (1999) published a paper on the use of recurrent neural networks as predictors of the constitutive behavior and volume variation of different sands and gravels under triaxial compression in drained conditions. In this study, the effects of material mineralogy, grain size and distribution, void ratio, and confining pressure were evaluated. The authors also presented the advantages of non-uniform variation of the increment strains in the input data of the network. The models were tested and validated against experimental data in the literature.

More recently, Stefanos and Gyan (2015) also proposed the use of ANNs in the constitutive modeling of geotechnical structures. In their work, the authors used as input data the length of strain trajectory traced by a material point, also called "intrinsic time". In this work, synthetic training data generated through numerical simulations were used. Two NNCMs were trained, each with a specific traditional constitutive model, and good results were obtained in both cases. The authors highlight, in their text, the computational efficiency as the main advantage of this methodology. Several other papers have presented the use of NNCMs in predicting the behavior of geomaterials, such as Logar and Turk (1997), Wu (1997), Shahin et al. (2008), Smyrniou (2018), and others.

Besides its applicability in quasi-brittle materials, such as concrete, and in geotechnical materials, constitutive modeling via ANNs has also been proposed for other types of materials with complex nonlinear behavior, such as elastomeric foams (Liang and Chandrashekhara, 2008), for high-temperature deformation behavior of Aermet100 steel (Ji et al., 2011), for architectural textiles (Colasante and Gosling, 2016), for biological soft tissues (Liu et al., 2020), for crystal structures (Im et al., 2021), among others, showing promising results and sometimes better than conventional models. Liu et al. (2021)

[1]Recurrent Neural Networks (RNNs) is a class of neural networks where the nodes can have cyclic connections, that is, allow the outputs of some nodes to impact the inputs of the same nodes. This technique is commonly used in data sequences problems such as in natural language processing.

present a review of the application of ANNs in material modeling of composite materials and structures.

The historicity of ANNs application in constitutive modeling of materials ratifies their undeniable promising character. Nevertheless, the non-explicit setting of physical laws in the model configuration concerns some researchers, as can be seen in works such as in Liu et al. (2019). In this regard, Masi et al. (2021) proposed a new class of ANNs, which incorporates the physical laws of thermodynamics into the neural network architecture for modeling elasto-plastic materials. In this work, the authors used data generated through numerical simulations. This new approach presented great assertiveness in the predictions and generalization of the model, placing it as a potential constitutive model for finite element formulations. Still along these lines, Linka et al. (2021) presented a new architecture for material modeling, which the authors named Constitutive Artificial Neural Networks (CANNs). This model incorporates in the network architecture, besides the information of stresses and strains, theoretical knowledge of the theory of materials.

New papers on constitutive modeling have been published, among which are Ling et al. (2016), Liu and Wu (2019), Bock et al. (2019), Huang et al. (2020), Zhou et al. (2021) and Xu et al. (2021).

In a complementary way, Javadi et al. (2003), Lefik and Schrefler (2003), Hashash et al. (2004), Liang and Chandrashekhara (2008), Stoffel et al. (2019), Im et al. (2021), among several other authors, have shown in their works that NNCMs can be easily used as constitutive models in nonlinear analysis via FEM. In their publications, the authors incorporate an NNCM into finite element codes, describe the problems related to this task, and test and validate the implementations. It is important to note that for the NNCM to be used in FEM, the input data of the network must be strains, or their increments, and the output data must be stresses, or their increments.

The proposition of a constitutive model based on machine learning requires the modification of the nonlinear analysis process in two distinct moments: first, in the calculation of the stiffness matrix of the problem, through the assembly of the tangent constitutive matrix of each integration point, and then, through the assembly of the internal forces vector, with the calculation of stresses at each material point. In the case of the internal force vector assembly, having as input data the strain increment and historical stress-strain states if necessary, all calculated at the integration points, the corresponding stress increment is obtained directly through the network. The same does not apply when obtaining the tangent stiffness matrix, because it can not be directly measured from the output of the neural network. Ghaboussi (2018) presents a strategy for obtaining the tangent constitutive matrix. Details of the formulation is described in Appendix B of this dissertation.

The applicability of ANNs, as well as other ML algorithms, is not restricted to material modeling. These techniques have been applied in the identification of material parameters

(Sankarasubramanian and Rajasekaran, 1996, Seibi and Al-Alawi, 1997, Yeh, 1998), in optimization problems (Kortesis and Panagiotopoulos, 1993, Theocaris and Panagiotopoulos, 1993), in multiscale modeling (Haj-Ali et al., 2001, Le et al., 2015), in structural damage identification (Hakim and Razak, 2013, 2014), among others studies. However, materials science and structural engineering is still considered one of the most backward areas in the application of these tools.

Given the importance of quasi-brittle media, especially concrete; the need for generalist constitutive models and the potentiality of using ML techniques, in particular ANNs, as tools in function approximation problems; and also the delay of structural engineering in applying these technologies; this work aims to use ANNs as predictors of concrete constitutive behavior, in order to learn material stress-strain relationships and generalize this knowledge to unknown data.

4.3 Training Data Acquisition

Conventional constitutive models are based on the laws of mechanics and the definition of assumptions and idealizations such as isotropy, elasticity, plasticity, normality rule, hardening, etc. Thus, from the observation of conventional material tests with regions with uniform stresses and strains distribution within the sample, one obtains parameters that feed these mathematical models and enable their use in finite elements. However, data from conventional material tests only have information regarding one material point, and therefore are not sufficient for training ANNs. It is known that a comprehensive training dataset is essential for good model performance, both in terms of learning and generalization capability. Hence, the greater the variety of available stress states, the greater the network's prediction potential. But, in practice, it is impractical to perform different conventional material tests in order to collect different material stress-strain curves. Even the use of non uniform material tests, that has regions with non uniform stresses and strains distribution within the specimen, as performed by Sidarta and Ghaboussi (1998), encounters limitation in the data volume required for training.

The autoprogressive training, proposed by Ghaboussi et al. (1998), is shown as an alternative to the infeasibility of generating a database based on direct experimental tests. Nonetheless, the complexity of the algorithm can be seen as a disadvantage. The code proposed by Ghaboussi et al. (1998) uses a pre-trained neural network with linear stress-strain states in two iterative nonlinear finite element analyses. The goal of this process is to find the target equilibrium path, generating at each iteration of the Newton-Raphson process, new training data; and at each convergence, a new network, trained and improved with this new data. This process would reduce the number of structural tests required to train the network due to the potential wealth of constitutive information contained in the results of these tests. However, this technique is computationally expensive and

susceptible to errors.

Besides the difficulty of obtaining experimental data in representative volume to formulate a comprehensive database, data from experimental tests also present deviations due to external influences and inherent to the material, such as equipment imprecision, specimen geometric inaccuracies, human flaws, among others. In addition, the experimental results, whether load displacement from a control point or stress-strain from an integration point, are represented in the form of a patch, with a spectrum of result combinations, reflecting the variability of the tests and the materials.

In view of what has been exposed, and because this is an initial and pioneer study in the research group, with scarce time and resources, this work uses synthetic data for training the NNCM, coming from numerical simulations with appropriate refinement, based on consolidated conventional constitutive models. This strategy was also observed in many works mentioned above.

In practice, there is no general constitutive model. The literature shows that there are models that give good results, each for a particular situation, when compared to experimental or other numerical results. These models reflect dense studies by various researchers over the past decades. This means that the hypotheses adopted are consistent and valid for the cases analysed. It is believed that these models, which are well accepted for certain situations and structural systems, can be used to generate a database for training a more robust neural network model. The main idea is to combine a range of consolidated traditional constitutive models, each in its best application, and use them to form a comprehensive database. If the model represents the material well and agrees with the experimental curve, it can naturally be used in generating the data so that the network captures the behavioral patterns of the material. It is well-known that the information learned by ANNs is contained in the training data. If the conventional constitutive model is consistent, neural networks can still capture patterns and information that are not clearly identified. This is the reason for the necessity of the availability of comprehensive and reliable data that has the intrinsic information of the medium.

Caution is required, however, because just as neural networks can capture unknown and inherent information from the medium, they can also learn noise or information that does not adequately represent the behavior of the material to be learned. In this sense, one should evaluate the physical consistency of the network responses and, as far as possible, compare the model results with those observed in the literature.

Chapter 5

Training Database

This chapter presents the numerically simulated structural systems for generating the training database for the neural network-based constitutive model and an analyses of the data distribution.

5.1 Dataset Generation

Since this is an initial study, whose main purpose is to demonstrate the feasibility of using ANNs as constitutive models for nonlinear analysis of structures, it was necessary to set several parameters, both in the context of concrete constitutive modeling and machine learning. Besides the definition of a constitutive model, it was necessary to choose a single material to generate the synthetic training database. However, it should be noted that future studies of the research group may contemplate the variation of material properties and the use of different constitutive models. Therefore, a concrete type was defined and the same material was applied in all numerically simulated structural systems. It was sought to select structures under the most varied types of loading and with failure processes characterized by bending, tension, compression, and shear stresses, in order to embrace the largest number of possible stress combinations. Finally, it was assumed that the numerical simulations performed here are reliable of the material behavior.

The structural systems were simulated via finite elements with the smeared crack model with fixed direction and the damage laws of Carreira and Chu, both for compression and tension (Eq. (2.22) and Eq. (2.40)). The motivation for using this constitutive model, as well as its formulation, is described in Chapter 2. The concrete properties adopted in this study were those obtained in the experiment on a L-shaped panel, conducted by Winkler et al. (2004): characteristic compressive strength of concrete $f_{ck} = 31.0$ MPa, characteristic tensile strength of concrete $f_t = 2.7$ MPa, Young's modulus $E = 25850.0$ MPa and Poission's ratio $\nu = 0.18$. For the following parameters, it was considered the calibrated values used by Penna (2011), in his numerical analysis of the same experiment: strain corresponding to the concrete compression strength limit $e_c = 2.2 \times 10^{-3}$ and strain

corresponding to the concrete tensile strength limit $e_t = 1.925 \times 10^{-4}$. The shear retention factor β_r was evaluated and defined individually in each case. All numerical simulations were performed in the INSANE system.

At this point, it is important to say that the generation of training data should be done with a convergence analysis of the mesh of the numerical problem, to find the minimum refinement to have answers with better accuracy. However, due to the short time to produce this research and as the main goal of this work is to develop and demonstrate feasible an NNCM, this analysis was not considered.

By the nature of the nonlinear analysis process, the strain increments are obtained constant from one step to another. To circumvent this problem, it was created a code that interpolates the FEM responses with random values in order to obtain different strain increments for all numerical tests performed. This approach was used so that all regions of the curve have enough points to ensure their representativeness in the database, especially the regions with low strain values and those near the deviator stress (Penumadu and Zhao, 1999), and also because the constancy of the strain increments may impact the predictions and explainability of the network.

5.1.1 Elementary States

As a starting point, concrete plates of unit dimensions were tested, subjected to uniaxial stress states and biaxial stress states. The simulations were performed under the plane stress state. Experimental studies (Kupfer et al., 1969, Nelissen, 1972) have shown that concrete strength can vary significantly depending on the loading stress ratio (σ_1/σ_2). This means that different stress-strain ratios will produce different stress-strain behaviors. As the aim of this chapter is to capture a reasonable amount of different behavioral patterns of concrete, to enrich the information in the training database, the loads were applied in order to obtain pure tension and pure compression states and biaxial tension-tension, tension-compression and compression-compression stress combinations, with variation of the ratio between the principal stresses, from 0 to 1, every 0.1, in both directions and in each one of the tests.

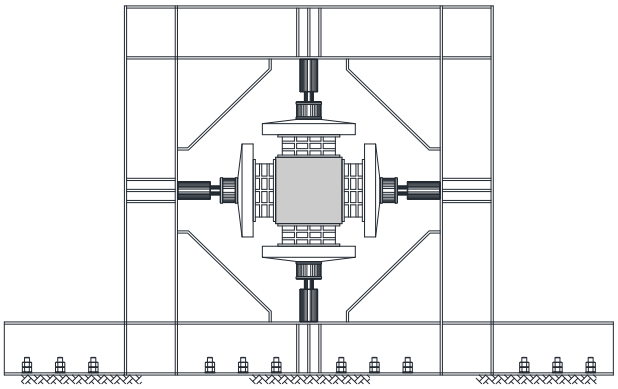


Figure 5.1: Plate on elementary tests.

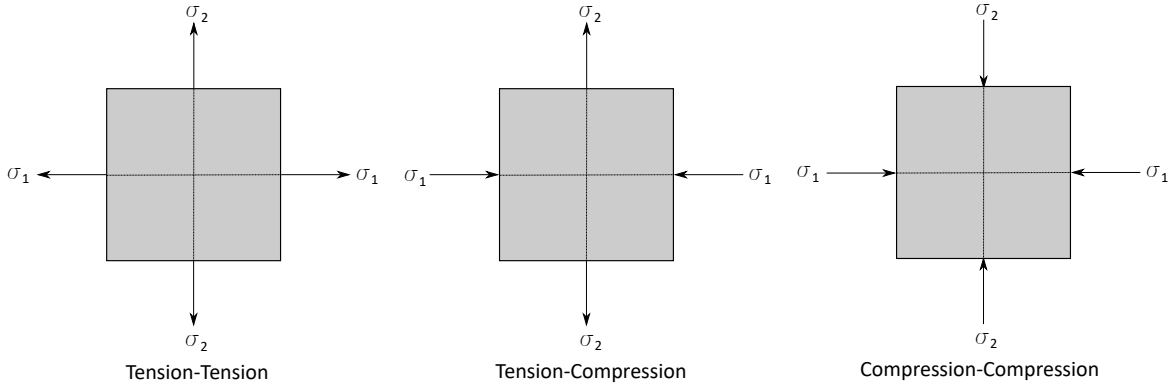


Figure 5.2: Biaxial stress combinations.

Thus, the plate was modeled using four-node quadrilateral finite element mesh. For the solution of the system of nonlinear equations, the Newton-Rhapson process was used, with the direct displacement control method and a tolerance for convergence in displacements of 1×10^{-4}. For each test, a different displacement increment value was set, in order to obtain a dataset with wide variation among the strain increments of each test. The control point was the upper right node.

Finally, to obtain the shear states, the results of the biaxial tests were rotated at angles α from 0° to 90°, every 15°, as illustrates Fig. 5.3.

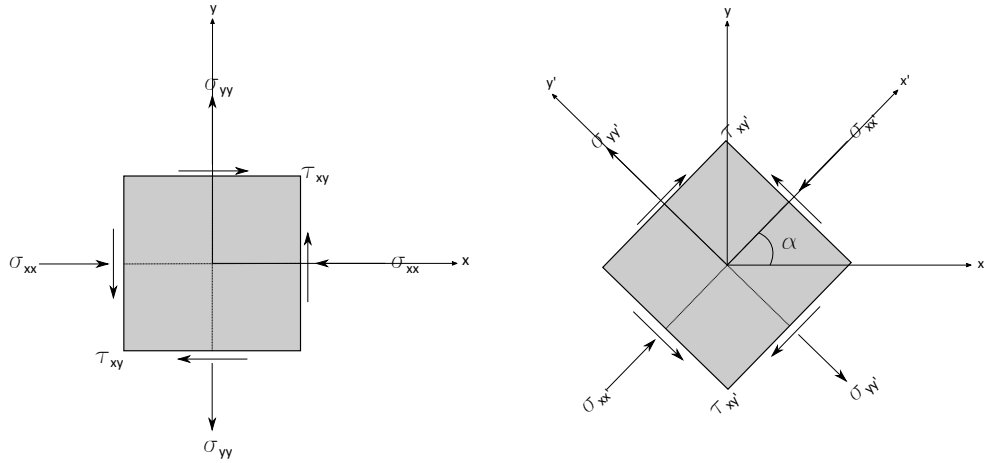


Figure 5.3: Rotated stress states.

As 97 different numerical simulations were performed, and consequently, a very high volume of training data was generated, to compose the final model training dataset it was selected only 23% of this initial database. In this selection, it was sought to guarantee the presence of uniaxial elementary states of tension and compression, in the x and y directions (all tests of this nature), and then it was selected randomly several tension-tension, compression-compression and tension-compression tests, with their respective rotated states. The amount of information regarding elementary tests was reduced in order to avoid an imbalance between the amount of information of this type of test and its stress states and the information of the other tests that will compose the final training database and that will be presented in the sequence of this chapter. Moreover, it is believed that the amount of data used is sufficient and representative of the elementary states for the given problem.

5.1.2 3-Point Bending Test

Next, a 3-point bending concrete beam under the plane stress state was simulated. This beam, experimentally studied by Petersson (1981), was adapted to the material defined in this work. The failure mode of this system is by bending, with tensile stresses. Fig. 5.4 shows the details of the beam as well as the finite element mesh considered.

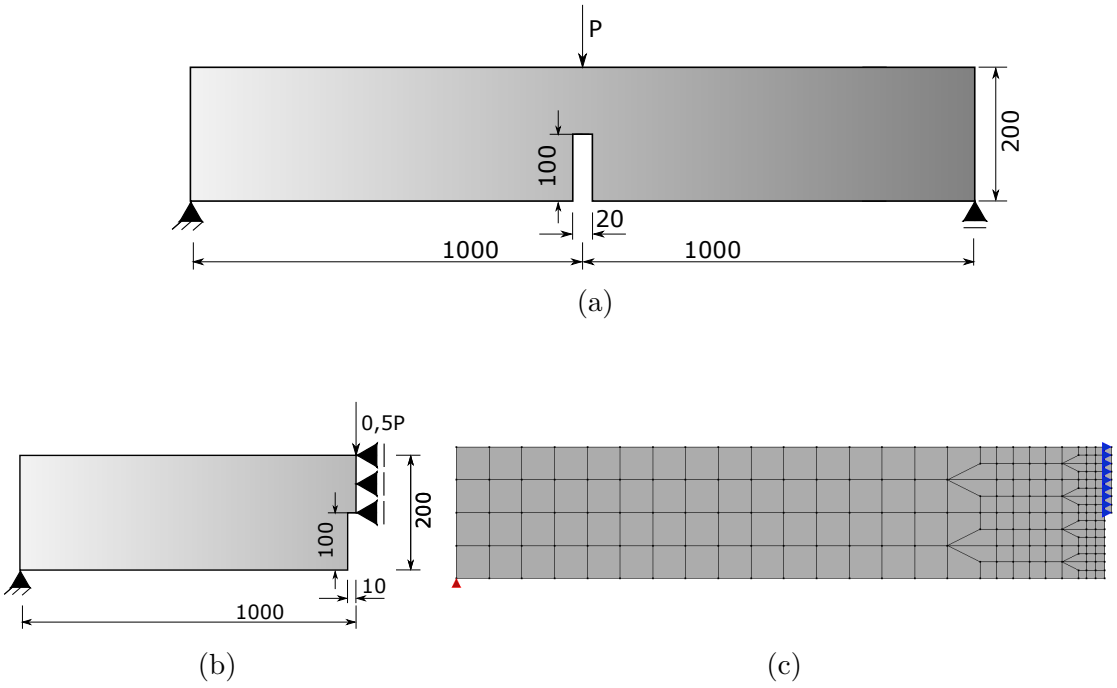


Figure 5.4: 3-point bending test. (a) Problem setting (measures in mm), (b) Modeling, taking advantage from symmetry, (c) Mesh setting.

The beam has a has a size of $2000 \times 200 \times 50$ mm^3, a central crack of 100×20 mm and is subjected to a central load of P = 400.0 N. The equilibrium paths were obtained using the generalized displacement control method, with an initial load factor increment of 0.02 and tolerance for convergence in displacements of 1×10^{-4}. For this case, the shear retention factor was considered to be zero ($\beta_r = 0.0$). The equilibrium path for the horizontal displacement of the left node of the crack tip is shown in Fig. 5.5.

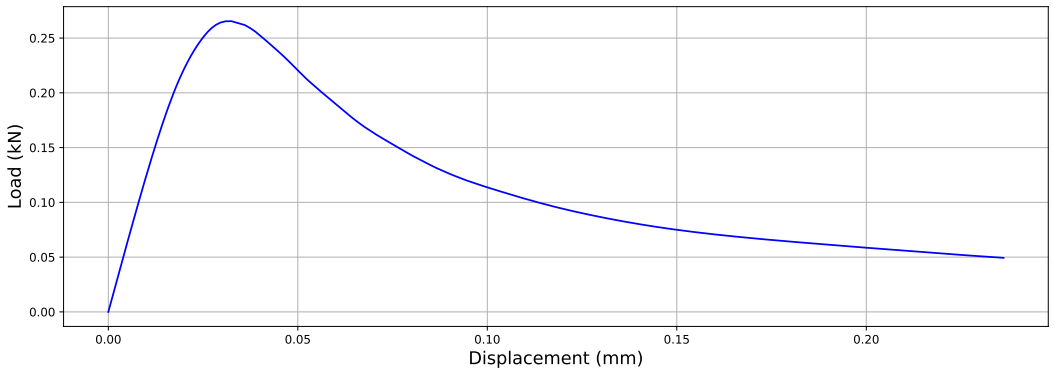
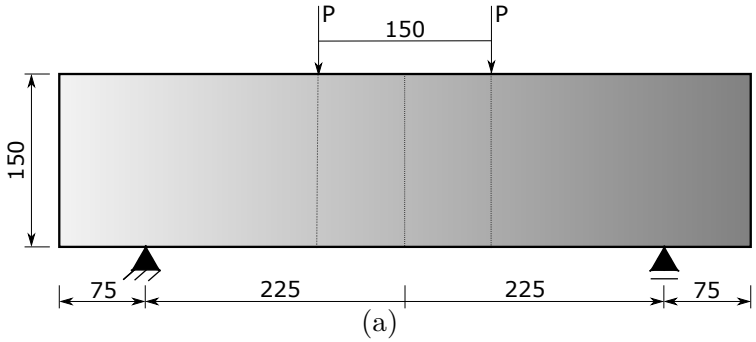


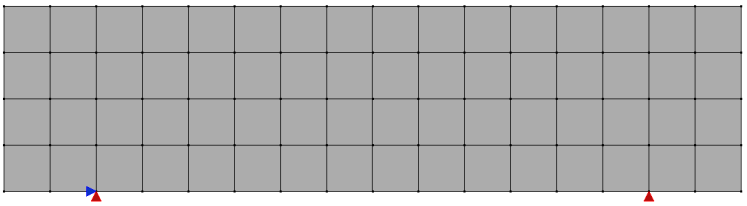
Figure 5.5: Load-displacement curve for the adapted 3-point bending test from Petersson (1981) - Horizontal displacement of the left node of the crack tip.

5.1.3 4-Point Bending Test

A concrete beam under 4-point bending was also simulated to generate training data. It is understood that this test includes in the variability of the training data, stress states referring to pure bending, which occurs in the region of the beam between the applied loads. The beam has a size of $600 \times 150 \times 120$ mm^3 and was simulated under plane stress state with a quadratic finite element mesh, as illustrated in Fig. 5.6. The analysis was performed using the cylindrical arc length control method, with an initial increment of the load factor of 0.05. The convergence was verified in displacements with tolerance of 1×10^{-4}. Fig. 5.7 presents the equilibrium path of the vertical displacement of the point in the lower central region of the beam.



(a)



(b)

Figure 5.6: 4-point bending test - Problem setting (measures in mm), (b) Q4 mesh.

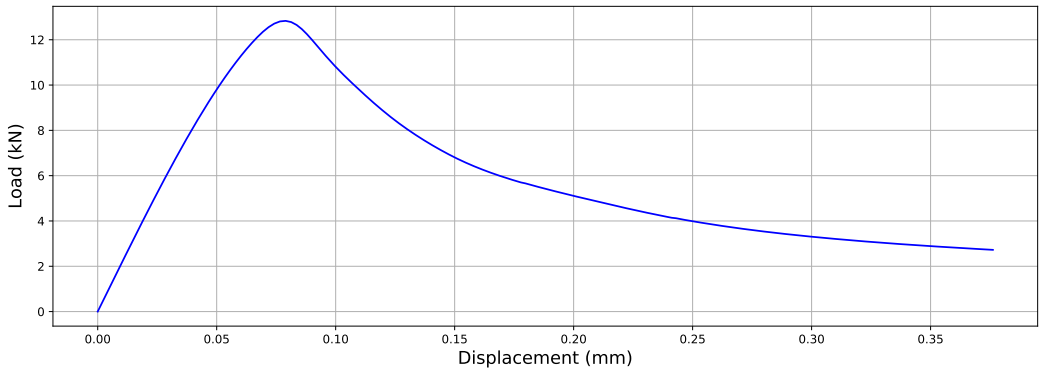


Figure 5.7: Load-displacement curve for the 4-point bending test - Vertical displacement of the bottom central node.

5.1.4 Prestress Test

A four-point concrete bending beam with a size of $500 \times 100 \times 50$ mm^3 was analysed with no pre-stress ($\sigma = 0$), and with a constant 1.0 N/mm^2 compressive pre-stress ($\sigma < 0$) and 1.0 N/mm^2 tensile pre-stress ($\sigma > 0$), under the plane stress state. The boundary conditions of the problem are shown in Fig. 5.8. Regarding the stress states, the model with negative pre-stress stands out, because it has combinations of tensile and compression efforts at the bottom of the beam, presenting a more complex stress state. To obtain the equilibrium paths, the generalized displacement control method was used, with an initial load increment of 0.1, a tolerance for convergence in displacements of 1×10^{-4} and reference loads $P_a = P_b = 1.0$ kN. For this case, the shear retention factor $\beta_r = 0.02$ was considered. Fig. 5.9 presents the results of the load-displacement curve of the bottom center node in the vertical direction.

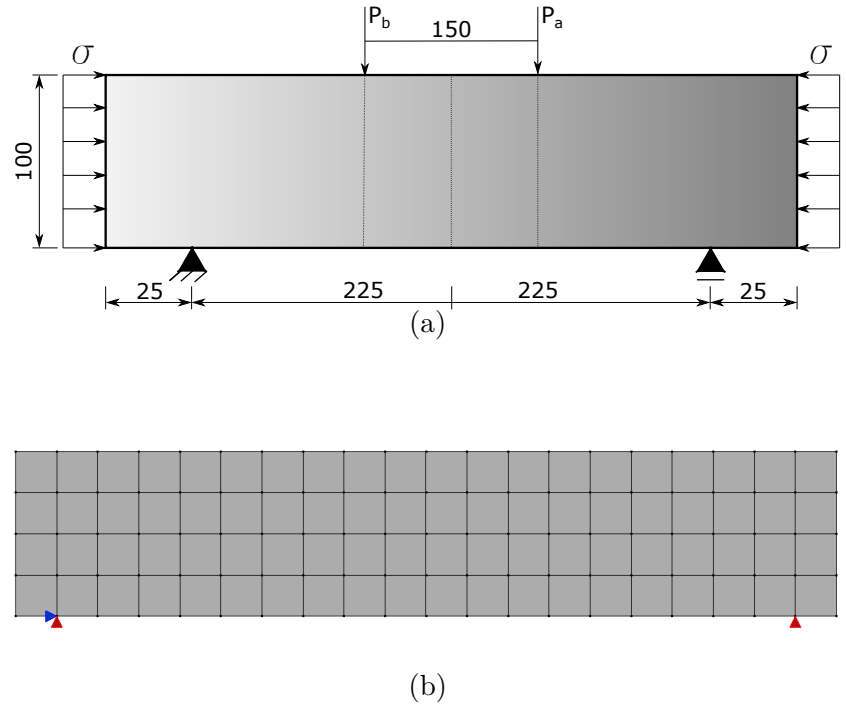


Figure 5.8: Pre-stress test. (a) Problem setting (measures in mm), (b) Q4 mesh.

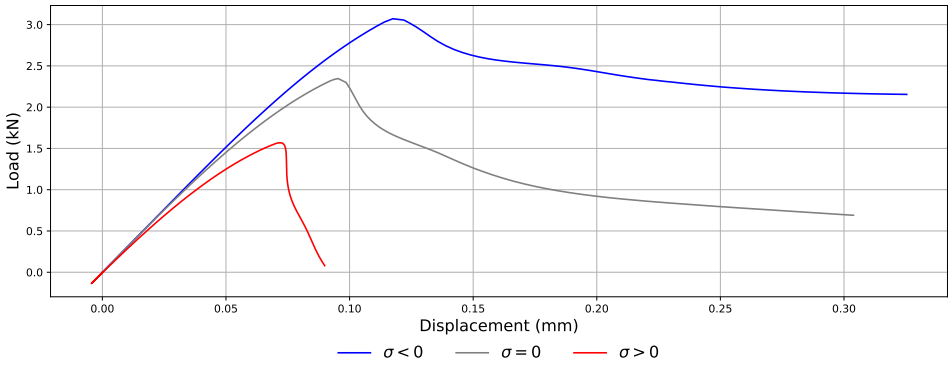


Figure 5.9: Load-displacement curve for the pre-stress test - Vertical displacement of the bottom central node.

5.1.5 Shear Test

Next, a numerical simulation of a shear test was performed, as illustrated in Fig. 5.10. The concrete structure with a size of $1 \times 1 \times 1$ mm^3 and with a initial central horizontal crack of 0.5 mm was modeled with quadrilateral finite elements under the plane stress state. To obtain the equilibrium path, the direct displacement control method was used, with horizontal displacement increment at the top right node and with a tolerance for convergence in displacements of 1×10^{-4}. For this model, it was used shear retention factor $\beta_r = 0.0$. The equilibrium path result of the horizontal displacement at the top

right node is presented in Fig. 5.11.

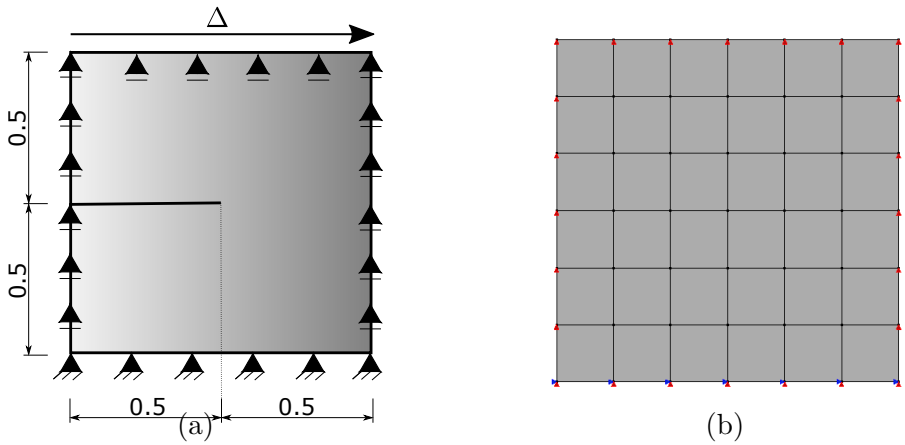


Figure 5.10: Shear test. (a) Problem setting (measures in mm), (b) Q4 mesh.

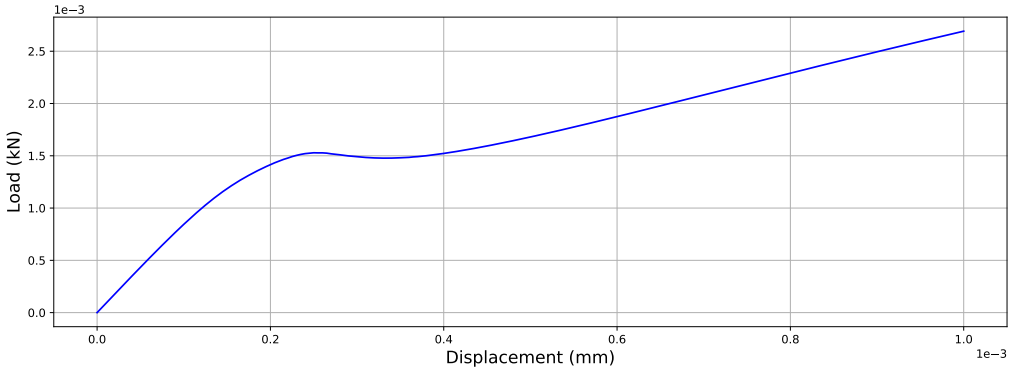


Figure 5.11: Load-displacement curve for the shear test - Horizontal displacement at the top right node.

5.1.6 Brazilian Splitting Test

Finally, to complete the training dataset, the brazilian splitting test (diametrical compression test) described in Fig. 5.12 were also simulated under the plane stress state. This test is commonly used to indirectly obtain the tensile strength of the tested material. The cylinder has a radius of 40 mm and a length of 160 mm. The mesh was generated with quadratic elements of four nodes, considering the double symmetry of the specimen (Fig. 5.13). Seven analyses of the same test were performed, varying the initial crack sizes c in 4, 8, 12, 16, 20, 24 and 28 mm. The rigid block (load application region) was considered to be linear elastic with Young's Modulus $E = 20000.0$ MPa and $\nu = 0.2$. For the generation of the training data, the tests with crack sizes of 4, 12, and 20 mm were used. The diametrical compression simulation results with 8, 16, 24 and 28 mm crack sizes were saved for use as test data.

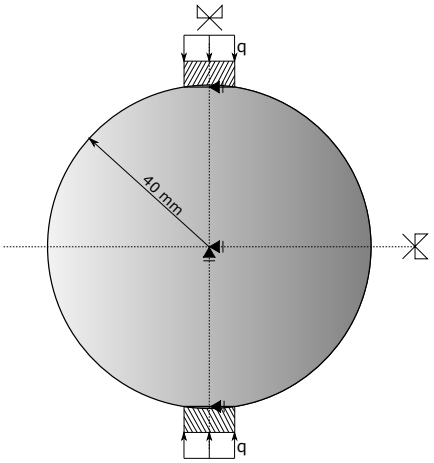


Figure 5.12: Brazilian splitting test (measures in mm).

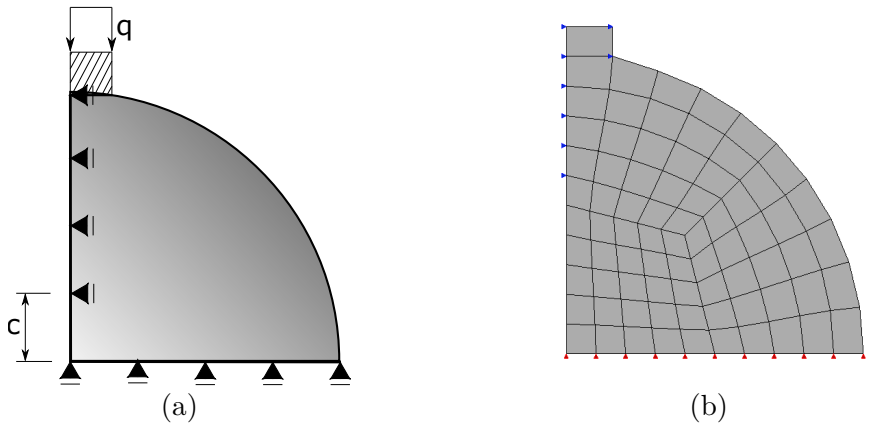


Figure 5.13: Brazilian splitting test setting taking advantage of the symmetry. (a) Problem setting (measures in mm), (b) Q4 mesh.

For the solution of the nonlinear analysis, the direct displacement control method was adopted, with a horizontal displacement increment of the mesh right end node of 7.5×10^{-5}. The load P = 0.5 N was applied and a tolerance for convergence in forces of 1×10^{-4} was set. For this case, the shear retention factor $\beta_r = 0.05$ was considered.

The equilibrium paths for the horizontal displacement of the controlled node and for the vertical displacement of the point of contact of the rigid block with the cylinder are shown in Fig. 5.14 and Fig. 5.15, respectively:

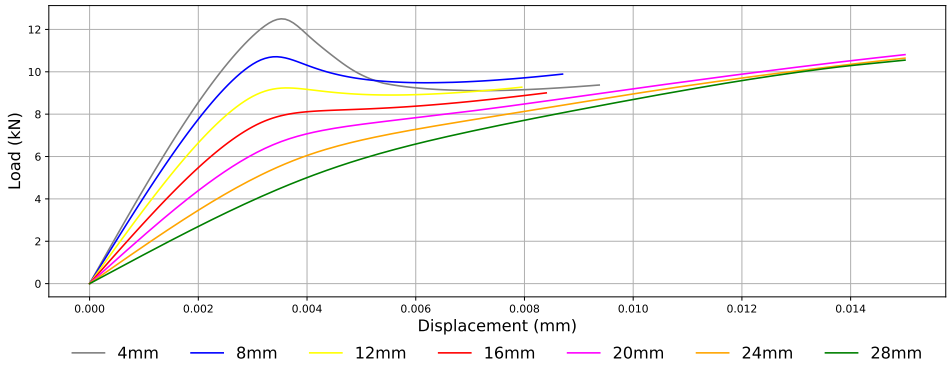


Figure 5.14: Load-displacement curve for the Brazilian splitting test - Horizontal displacement of the controlled node.

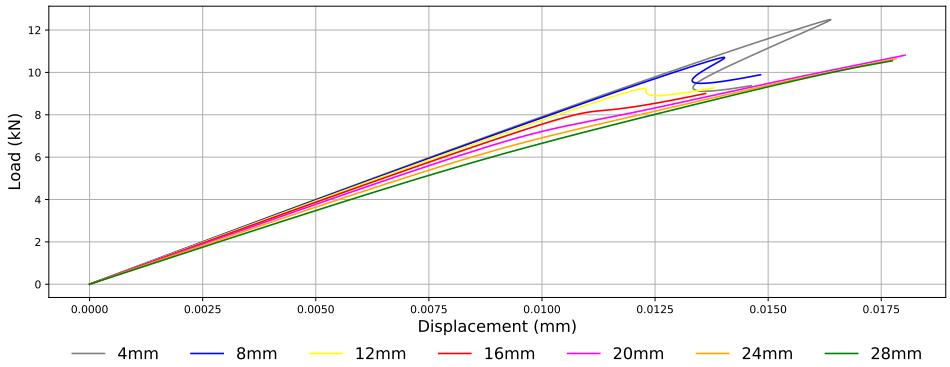


Figure 5.15: Load-displacement curve for the Brazilian splitting test - Vertical displacement of the point of contact of the rigid block with the cylinder.

It can be seen that the behavior of the structure changes as the crack increases, ranging from a softening with "snap back", for the smallest crack (brittle structural behavior), to a hardening, for the largest crack (ductile structural behavior). Furthermore, it is noted that the load capacity of the model is proportional to the effective height of the cylinder. This behavior variation characterizes the scale effect and generates different stress states in the structure, making this test a good example for generating training and test data, thus making it possible to analyze the predictive ability of the network through the change in material behavior with the change in structural behavior.

5.2 Dataset Distribution

To better comprehend the dataset to be used in the training and visualize its distribution, the boxplot graph was used as a resource. The boxplot, also known as box diagram, is a method used for exploratory analysis of quantitative or ordinal variables, which provides statistical information such as the minimum, first quartile (Q1), second quartile (median),

third quartile (Q3), maximum and also possible outliers. This graphical tool allows one to identify the distribution and outliers in the dataset, as well as information such as dispersion and symmetry (see Fig. 5.16).

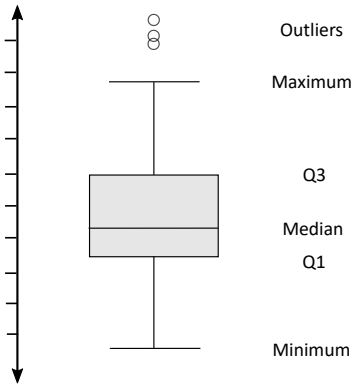


Figure 5.16: Boxplot graph.

Then, 3 boxplot graphs were generated on top of the model variables. The synthetic data generated totals 1.417.656 datasets. To do so, approximately 1% of the dataset was selected randomly. Fig. 5.17 shows the distribution of input variables in terms of the current stresses ($\sigma^n_{xx},\ \sigma^n_{yy}$ and τ^n_{xy}), Fig. 5.18 shows the distribution of input variables in terms of current strains ($\epsilon^n_{xx},\ \epsilon^n_{yy}$ and γ^n_{xy}) and Fig. 5.19 shows the distribution of input variables in terms of strain increments ($\Delta\epsilon^n_{xx},\ \Delta\epsilon^n_{yy}$ and $\Delta\gamma^n_{xy}$). The history stress and strain variables was not considered in theses graphs since they are similar to the current variables and therefore have the same distributions.

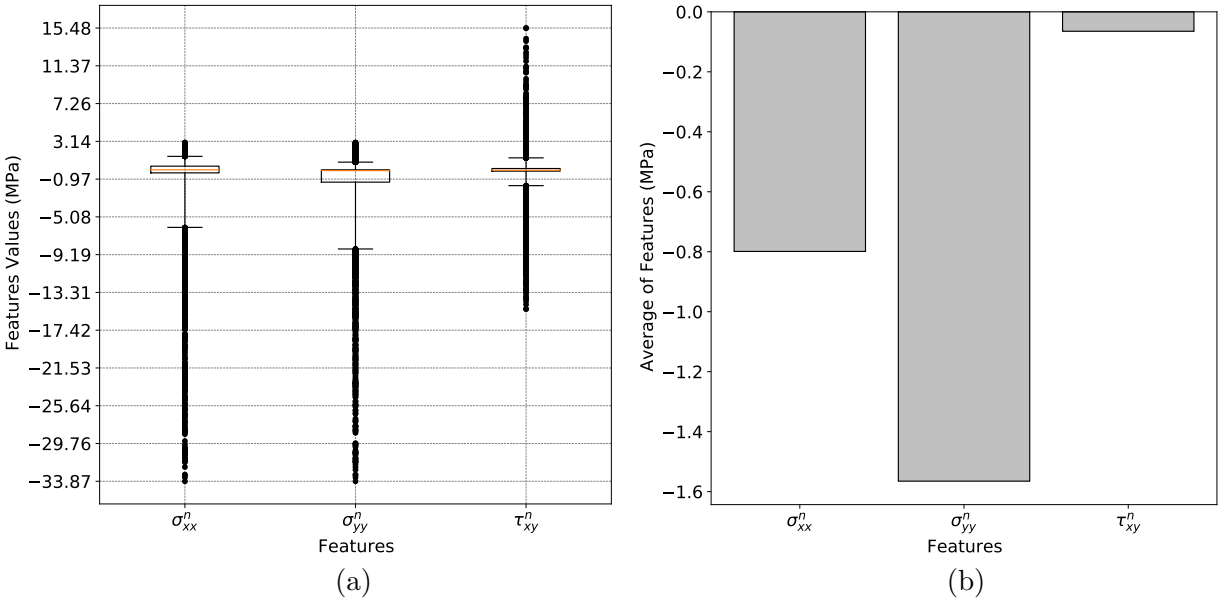


Figure 5.17: Dataset stress input variables distributions. (a) Features distributions, (b) Average of features.

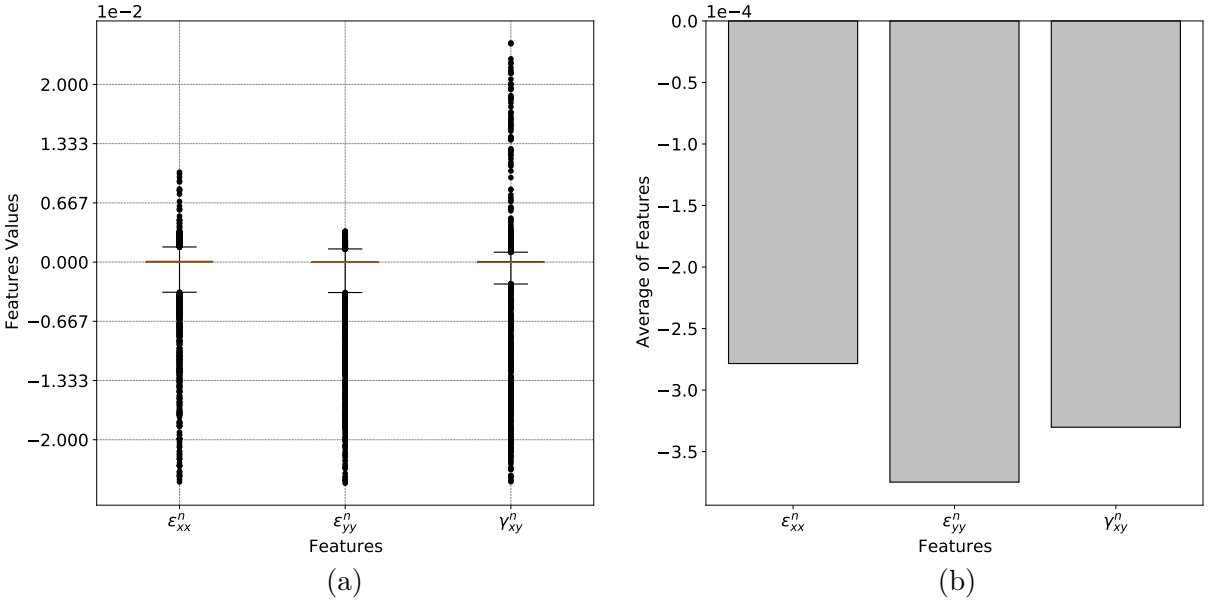


Figure 5.18: Dataset strain input variables distributions. (a) Features distributions, (b) Average of features.

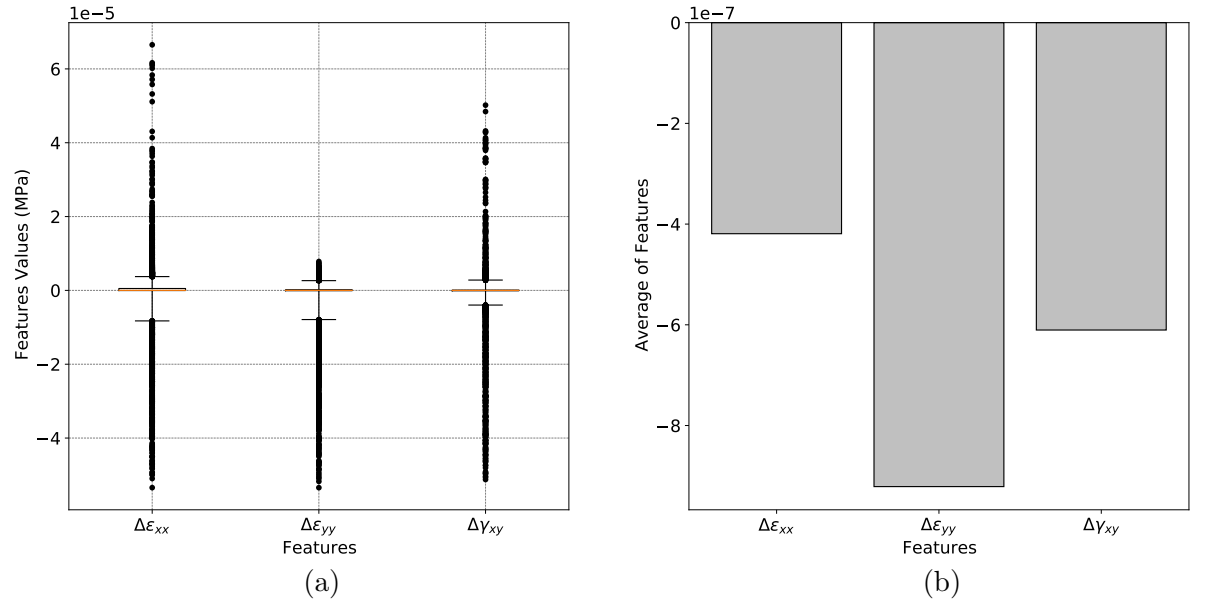


Figure 5.19: Dataset strain increment input variables distributions. (a) Features distributions, (b) Average of features.

Overall, Fig. 5.17 shows that the normal stress variables have a similar distribution, but that the stresses in the y-direction have a larger volume of data in the compression region, which can be seen by evaluating the amplitude of the graphs whiskers. A point of attention is that Fig. 5.18 and Fig. 5.19 indicate that although the negative strains, associated to compression, have similar distributions in both directions, that the positive

strains, related to tensile, reached a higher value in the x-direction. On the other hand, shear stresses and strains appear balanced with respect to their sign in both plots.

It is also observed that the statistical tool characterizes the stresses, strains and strain increments near the tensile and compression limits of the material as outliers. In statistics, an outlier is a point in the sample that is very different from the others, i.e. that deviates from normality. This may be an indication that the database is imbalanced. However, it is understood that this distribution is consistent with the nature of the problem under analysis. In the real world, various applications (such as in finance, meteorology, biology, etc.) present databases with common values that define the "normal" behavior of the system, leading to a scenario of imbalanced data. Despite the fact that data with majority characteristics or standard sizes exist, these problems constantly require the prediction of values that are considered rare and extreme (Krawczyk, 2016). For machine learning theory, this can be an issue, since the model may have difficulty interpreting the behavior of regions with low data volumes. Yet, imbalanced data is not always detrimental to the model, since for some decisions, a sufficient amount of information in the database is enough for the model to be able to learn and generalize. In the present work, it is believed that the database has sufficient information about the behavior of the material at its extreme limits and that the volume of data considered as outliers will decrease when a data scaling process such as normalization or standardization is applied, thus mitigating the impact of data imbalance on model performance.

To represent the behavior of concrete with complex stress states, the literature suggests using at least 2 historical stress states as input data. In this sense, each example will have 21 input variables and 3 output variables. So, to build the database, first, the stresses and strains of all (n) steps and all Gauss points of the finite element analysis were saved. Then, to obtain the historical variables, columns with the stresses and strains referring to steps $(n-1)$ and $(n-2)$ were added. Finally, the strain increments for each step (n) were calculated by taking the difference between the strains of step $(n+1)$ and step (n) and the stress increments by taking the difference between the stress of step $(n+1)$ and step (n). Here, the stress increments are the neural network's expected answers, used to compare with the predictions and calculate the error that will feedback the training process. Table 5.1 shows, in a generic way, the structuring of the data for any finite element k and any gauss point m with the stresses rotated at an angle α with respect to the origin state.

Table 5.1: Database Structure

Step	Current Stress	Current Strain	Historical Stress 1	Historical Strain 1	Historical Stress 2	Historical Strain 2	Strain Increment
n-2	$\{\sigma\}_{n-2}$	$\{\varepsilon\}_{n-2}$	$\{\sigma\}_{n-3}$	$\{\varepsilon\}_{n-3}$	$\{\sigma\}_{n-4}$	$\{\varepsilon\}_{n-4}$	$\{\Delta\varepsilon\}_{n-2}$
n-1	$\{\sigma\}_{n-1}$	$\{\varepsilon\}_{n-1}$	$\{\sigma\}_{n-2}$	$\{\varepsilon\}_{n-2}$	$\{\sigma\}_{n-3}$	$\{\varepsilon\}_{n-3}$	$\{\Delta\varepsilon\}_{n-1}$
n	$\{\sigma\}_{n}$	$\{\varepsilon\}_{n}$	$\{\sigma\}_{n-1}$	$\{\varepsilon\}_{n-1}$	$\{\sigma\}_{n-2}$	$\{\varepsilon\}_{n-2}$	$\{\Delta\varepsilon\}_{n}$
n+1	$\{\sigma\}_{n+1}$	$\{\varepsilon\}_{n+1}$	$\{\sigma\}_{n}$	$\{\varepsilon\}_{n}$	$\{\sigma\}_{n-1}$	$\{\varepsilon\}_{n-1}$	$\{\Delta\varepsilon\}_{n+1}$
n+2	$\{\sigma\}_{n+2}$	$\{\varepsilon\}_{n+2}$	$\{\sigma\}_{n+1}$	$\{\varepsilon\}_{n+1}$	$\{\sigma\}_{n}$	$\{\varepsilon\}_{n}$	$\{\Delta\varepsilon\}_{n+2}$

Chapter 6

Neural Network Design

> Hyperparameter tuning is considered by many users to be one of the main challenges in modeling ML algorithms. ANNs, specifically, have several variables that need to be defined prior to the training process, and that their definitions can significantly impact model performance. Furthermore, the model's performance, and consequently the optimal hyperparameters, depend on the database from which knowledge is to be extracted. In this context, this chapter presents a study on the optimization of these hyperparameters to obtain an optimal NNCM.

The tuning of the hyperparameters of a neural network is critical to the success of a model, as these variables affect the assertiveness of the predictions, the generalizability of the model, and can also impact the memory cost and runtime of the algorithm. The adjustment of these variables can be done manually or automatically. Manual choice requires the user to have prior knowledge about each hyperparameter and how they influence the capability of a model. Automated picking does not require such mastery of the subject matter, but may have a high computational cost. However, using algorithms that optimize the hyperparameters can be a good alternative in the absence of previous studies and applications on the same subject or database (Goodfellow et al., 2016).

The works on the application of ML models in structural engineering, such as those mentioned in Chapter 4, in its great majority, do not present information regarding the hyperparameters used in the training of the proposed network, nor do they discuss the training process and the difficulties encountered by the authors during the process of defining the best model. In view of this, this work studied, by means of optimization techniques, the best hyperparameters to train and define a ANN with a concrete stress-strain database.

In this work, a model hyperparameter tuning technique known as *GridSearchCV* was used. This class evaluates, through the cross-validation technique, each of the possible combinations of hyperparameters, according to the values defined by the user. The best model, and therefore the best set of hyperparameters, is the one with the lowest error

among all the analyses. One of the main disadvantages of *GridSearchCV* is that its computational cost increases as the number of hyperparameters to be adjusted increases. Furthermore, to use this method, it is necessary for the user to fix some hyperparameters. However, some hyperparameters are easier to choose due to heuristics and even good machine learning practices, and can be used as a starting point in the search for other optimal hyperparameters for the given database. In this sense, it was set the activation function of the hidden layers as ReLu and the activation function of the output layer as Identity.

For this study, it was used the ML libraries for Python scikit-learn and keras, available on its website[1].

6.1 Grid Search Analysis

Initially, it was evaluated jointly, which are the scaling technique and the optimization algorithm best suited for the database of this study. To do this, it was necessary to first define a complete neural network architecture. The neural network was defined with 1 hidden layer of 25 neurons, with ReLu activation function in the hidden layer and Identity activation function in the output layer, Random Normal synaptic weights initialization technique, batch size of 32 and the MSE as cost function. The dataset chosen was a 5% random sample of the entire training dataset (56.643 data pairs). The metric used in these analyses was the MAE. To evaluate the performance of the models, the 5-fold cross-validation technique was used over 20 epochs. The network architecture and data volume were designed in order to reduce the computational cost inherent in automated hyperparameter optimization, while respecting the heuristics that state that the number of training data should be larger than the number of free network parameters (synaptic weights). As this work aims to represent concrete under complex stress states, two historical points of the stress-strain curve were used as input data, in the same line as observed in the literature. The structure of the ANN is represented in Eq. (6.1).

$$\begin{aligned}\{\Delta\sigma\} = \mathbf{NN}(&\{\Delta\varepsilon\}, \{\sigma\}_n, \{\varepsilon\}_n, \{\sigma\}_{n-1}, \\ &\{\varepsilon\}_{n-1}, \{\sigma\}_{n-2}, \{\varepsilon\}_{n-2} : 21 \mid 25 \mid 3\ )\end{aligned} \tag{6.1}$$

Regarding the optimization algorithms, 5 different techniques were analysed. The performance of the algorithms was evaluated with the default settings of keras for SGD, Adam, Adagrad and RMSprop, since it is believed that the default settings represent the best combinations of hyperparameters for any given database. However, it is understood that the optimal hyperparameters vary depending on the nature of the database and can

[1]The scikit-learn library is available at: `https://scikit-learn.org/stable/` and the keras library is available at: `https://keras.io/`.

only be defined and validated after a specific and more detailed investigation. In addition, the Momentum algorithm was evaluated, defining the learning rate $\eta = 0.001$ and momentum $\alpha = 0.9$, in order to observe the impacts that the consideration of this variable can bring to the stability of the process. The learning rate was chosen to facilitate the comparison of the Momentum results with the results of the other algorithms. Table 6.1 presents the optimizers and their hyperparameters, most of them introduced and defined in Section 3.7.6 and Section 3.7.7. RMSProp hyperparameter ρ is a discounting factor for the history gradient and is typically close to 0.9.

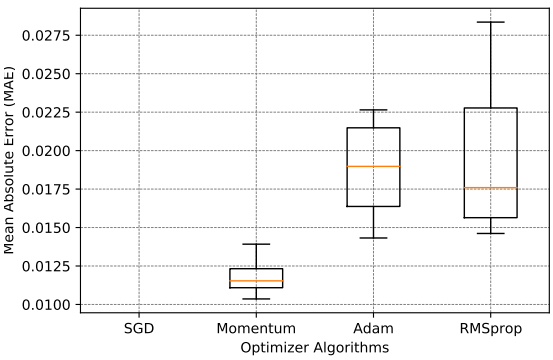
Table 6.1: Optimizers and their hyperparameters

Optimizer	η	α	β_1	β_2	ε	ρ
SGD	0.01	0.0	N/A	N/A	N/A	N/A
Momentum	0.001	0.9	N/A	N/A	N/A	N/A
Adam	0.001	N/A	0.9	0.999	1×10^{-7}	N/A
Adagrad	0.001	N/A	N/A	N/A	1×10^{-7}	N/A
RMSprop	0.001	0.0	N/A	N/A	1×10^{-7}	0.9

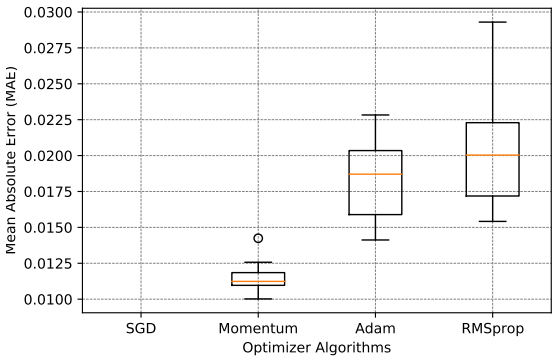
Regarding the data scaling process, 7 different scenarios were evaluated, performing combinations of preprocessing the input and output data of the network without data scaling and with the normalization technique and the standardization technique (both defined in Section 3.7.5), in the following situations:

- `UI-UO`: Unscaled Inputs - Unscaled Outputs;
- `UI-NO`: Unscaled Inputs - Normalized Outputs;
- `UI-SO`: Unscaled Inputs - Standardized Outputs;
- `NI-UO`: Normalized Inputs - Unscaled Outputs;
- `NI-NO`: Normalized Inputs - Normalized Outputs;
- `SI-UO`: Standardized Inputs - Unscaled Outputs;
- `SI-SO`: Standardized Inputs - Standardized Outputs.

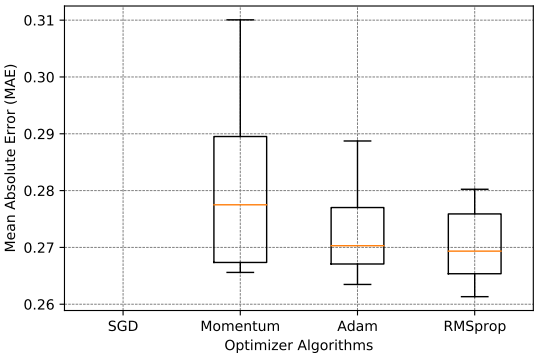
Since the results may vary due to the stochastic nature of the optimization algorithms, the analysis via grid search was repeated 10 consecutive times and the average of the cross-validation process for each of the simulations was saved. Thus, boxplots were plotted with the average results of the MAE metric.



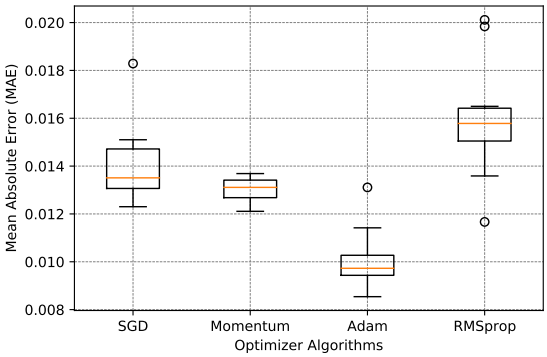
(a)



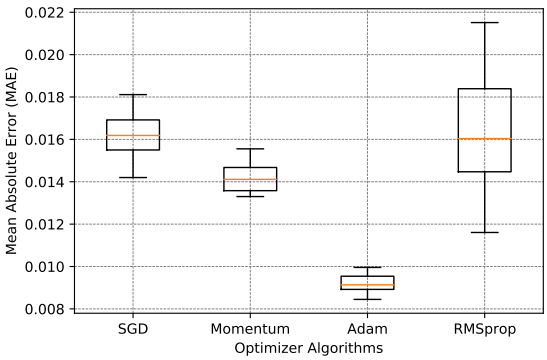
(b)



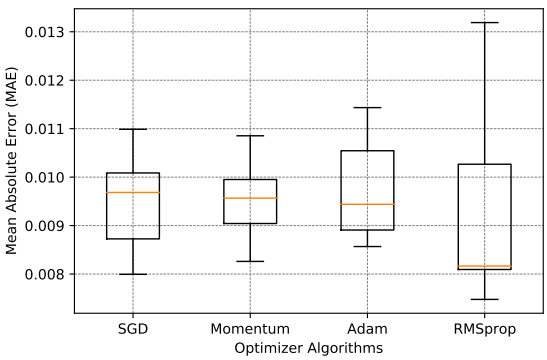
(c)



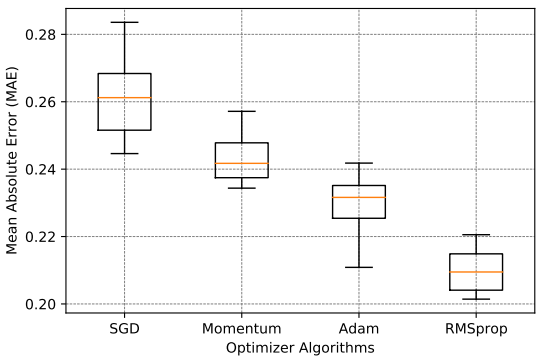
(d)



(e)



(f)



(g)

Figure 6.1: Boxplot of the average performance of the optimizing algorithms in each scenario of the input and output data scaling process. (a) UI-UO, (b) UI-NO, (c) UI-SO, (d) NI-UO, (e) NI-NO, (f) SI-UO, (g) SI-SO.

Table 6.2 summarizes the results of all analyses in terms of the average MAE metric.

Table 6.2: Grid search results on the Optimizer Algorithms and Data Scaling Process

	Optimizer Algorithms				
Data Scaling Process	SGD	Momentum	Adam	Adagrad	RMSprop
UI-UO	N/A	0.0117	0.0189	0.4002	0.0196
UI-NO	N/A	0.0116	0.0184	0.4867	0.0207
UI-SO	N/A	0.2815	0.2727	0.5785	0.2704
NI-UO	0.0141	0.013	0.0101	0.0811	0.016
NI-NO	0.0161	0.0142	**0.0092**	0.0875	0.0164
SI-UO	0.0095	0.0095	0.0097	0.5346	0.0091
SI-SO	0.2606	0.2432	0.2285	0.6437	0.2099

After evaluating the results, it was observed that the Adam and Momentum optimization algorithms showed good performance and stability in all simulations performed. Adam showed the best results for the NI-UO and NI-NO analyses and Momentum for the UI-UO and UI-NO analyses. RMSprop showed a slightly better result than Adam and Momentum only in the SI-UO analysis. Standardization of the output data (UI-SO and SI-SO) showed the worst performances for all combinations carried out. The Adagrad optimization algorithm proved to be the least suitable for the stress-strain database. Therefore, to facilitate the visualization of the results, due to scaling issues, the results of this optimizer were disregarded from the graphical analyses presented in Fig. 6.1. It could also be seen that the SGD algorithm proved unstable for non-scaling of the input data. Of the 10 analyses performed, only $20-30\%$ of the cross-validation tests converged, and therefore, Fig. 6.1.a, Fig. 6.1.b, and Fig. 6.1.c show no results for such an optimizer. The better performance of Momentum over SGD can be explained due to the addition of the momentum variable and also the decrease in the learning rate, from 0.01 to 0.001. It is known that the addition of the momentum factor tends to accelerate the process of the gradient descent in the right direction and lead to faster convergence and smaller learning rates tend to make the process less unstable. Overall, the best performances were for the cases in which input data preprocessing was performed (NI-UO, NI-NO and SI-UO), situations in which the metric showed the lowest average error. However, the results for the non-scaling of the input and output (UI-UO) data indicate that the stress-strain database may not necessarily need preprocessing. Nevertheless, preprocessing has shown that it can bring more stability to the process and help the algorithm to achieve better performance (as observed in the case of SGD). Therefore, to continue the study, the process of scaling the input and output data of the network was fixed with the normalization technique and parallel analysis was performed with the Adam and Momentum optimization algorithms.

The following analysis was performed entirely in the sklearn package. This time, the aim of grid search was to define the best architecture for the training database. This was done by setting the activation functions of the hidden layers as ReLu, the function of the

output layer as Identity, and the batch size as the number of training data divided by 100. The grid search on the optimal hyperparameters was performed according to Table 6.3 for Momentum, with 240 combinations and to Table 6.4 for Adam, with 540 combinations. For the Adam algorithm, the hyperparameter for numerical stability, $\varepsilon = 1 \times 10^{-8}$, was set, as defaulted by the sklearn library. The best ANN was evaluated based on the R^2 metric. For all cases the condition of having a training dataset larger than the number of free parameters was respected. For all analyses, 36% of the training dataset was randomly selected, except for the $(600, 750)$ network architecture, where it was necessary to increase the amount of data to 45% of the total training dataset. The combinations with second hidden layer with 750 neurons were the last one performed, after identifying the trend of the optimal architecture in that region. It is important to remember that the larger the volume of the training database, the higher the computational cost of training the ANNs. Therefore, since grid search is a computationally expensive analysis, the number of the training dataset was increased only when deemed necessary.

Table 6.3: Hyperparameters Used in Combinations for the Momentum Algorithm

1^{st} Hidden Layer	2^{nd} Hidden Layer	η	α
150	150	0.1	0.90
300	300	0.01	0.95
450	450	0.001	0.999
600	600	0.0001	
750			

Table 6.4: Hyperparameters Used in Combinations for the Adam Algorithm

1^{st} Hidden Layer	2^{nd} Hidden Layer	η	β_1	β_2
150	150	0.01	0.90	0.90
300	300	0.001	0.95	0.95
450	450	0.0001	0.999	0.999
600	600			
750				

Fig. 6.2.a and Fig. 6.2.b present, respectively, an isofield and heat graph, according to the R^2 metric, of the best results obtained by each trained architecture, for the Momentum and Adam optimizers.

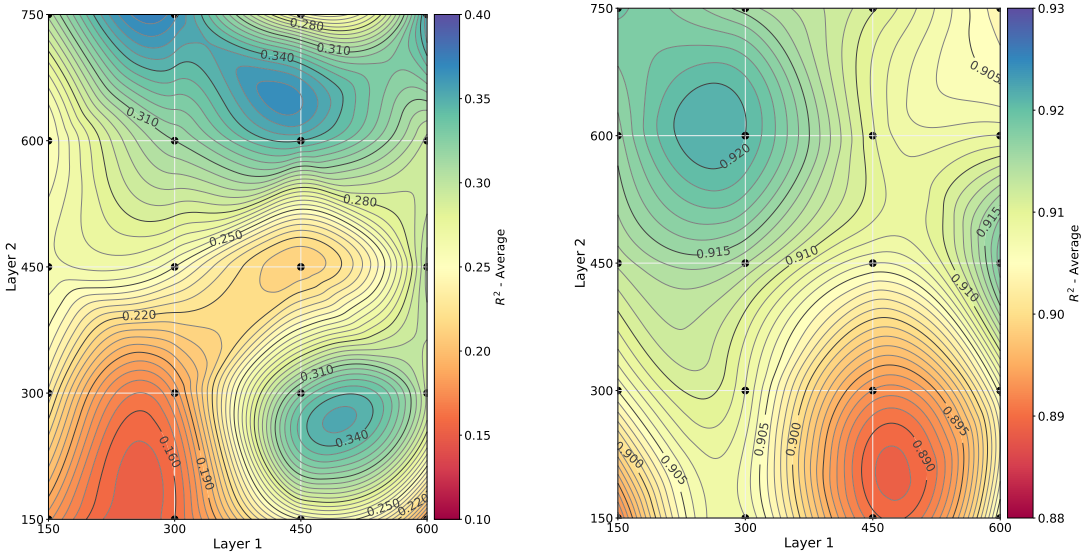
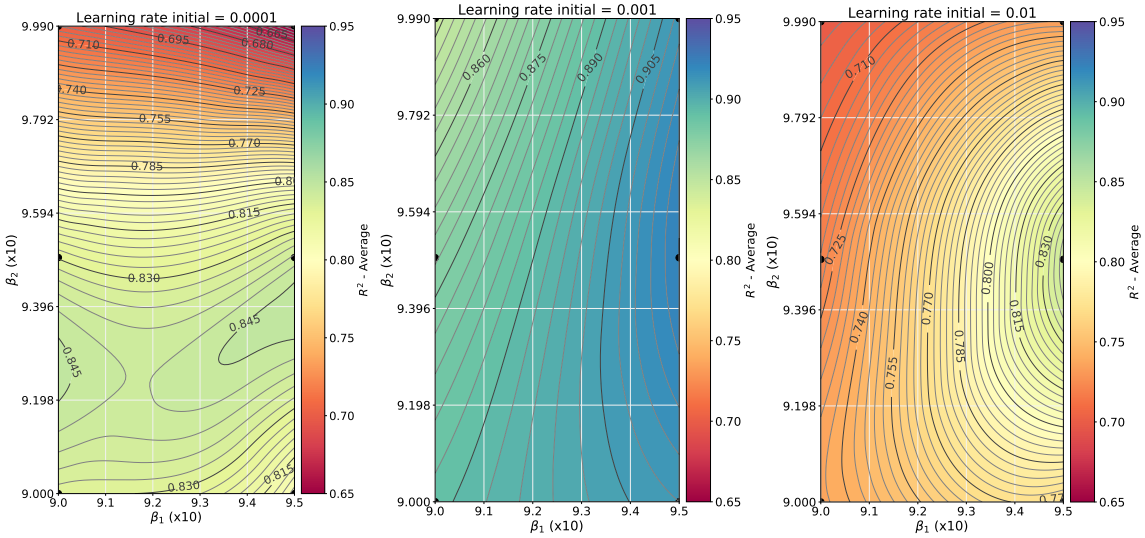


Figure 6.2: Grid search on the ANN architecture. (a) Momentum optimizer algorithm, (b) Adam optimizer algorithm.

By analyzing the Fig. 6.2, it is observed that among both optimization algorithms, Adam shows better results, since it obtained R^2 values closer to 1.0. From Fig. 6.2.b, it can be seen that the network that presented the best performance was the one with architecture of $(300, 600)$, with $R^2 = 0.92$. There is also a tendency for another optimal result to exist on the central right of the graph. However, by the Principle of Parsimony, one should always choose the simplest ANN, and therefore, networks with the number of neurons in the first hidden layer above 600 were not investigated, as they would present a higher number of parameters than the architecture network $(300, 600)$.

Next, the performance of the best architecture network, for each of the 3 learning rates studied, was evaluated based on an iso-band and heat graph, by varying the β_1 and β_2 hyperparameters of the Adam algorithm (Fig. 6.3).



(a) (b) (c)

Figure 6.3: Grid search on the Adam's optimal hyperparameters (a) $\eta = 0.0001$, (b) $\eta = 0.001$, (c) $\eta = 0.01$.

Here, it is important to emphasize that the best network was also tested for the learning rate $\eta = 0.1$, but it presented a negative R^2 and was therefore discarded. As a tendency was observed to have better combinations of β_1 and β_2 between 0.96 and 0.98, for both hyperparameters, new combinations of these hyperparameters were investigated, this time varying the learning rate from the one that presented a better result, as illustrated by Fig. 6.4. Finally, Fig. 6.5 presents, in a more refined way, the behavior of β_1 and β_2 for the learning rates $\eta = 0.001$ and $\eta = 0.0025$.

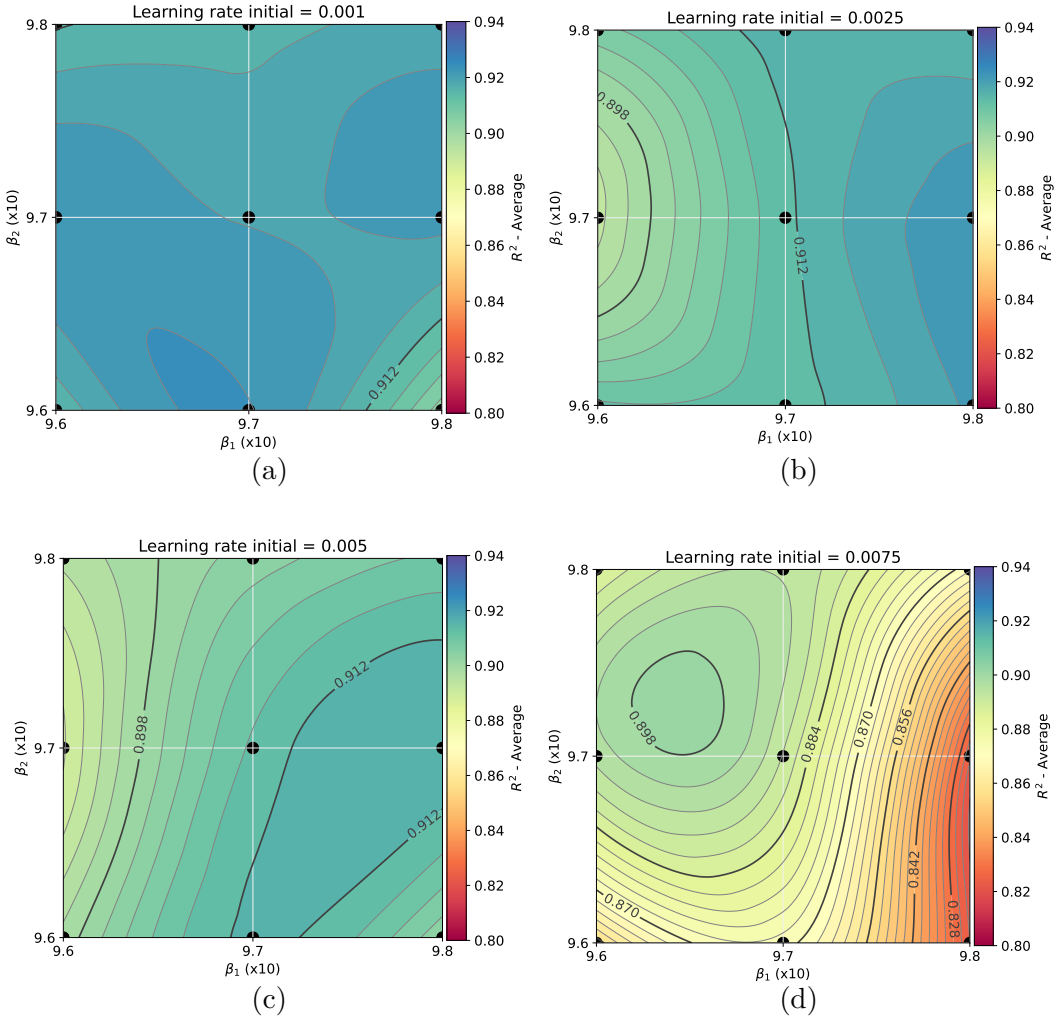


Figure 6.4: Grid search on the Adam's optimal hyperparameters (a) $\eta = 0.001$, (b) $\eta = 0.0025$, (c) $\eta = 0.0050$, (d) $\eta = 0.0075$.

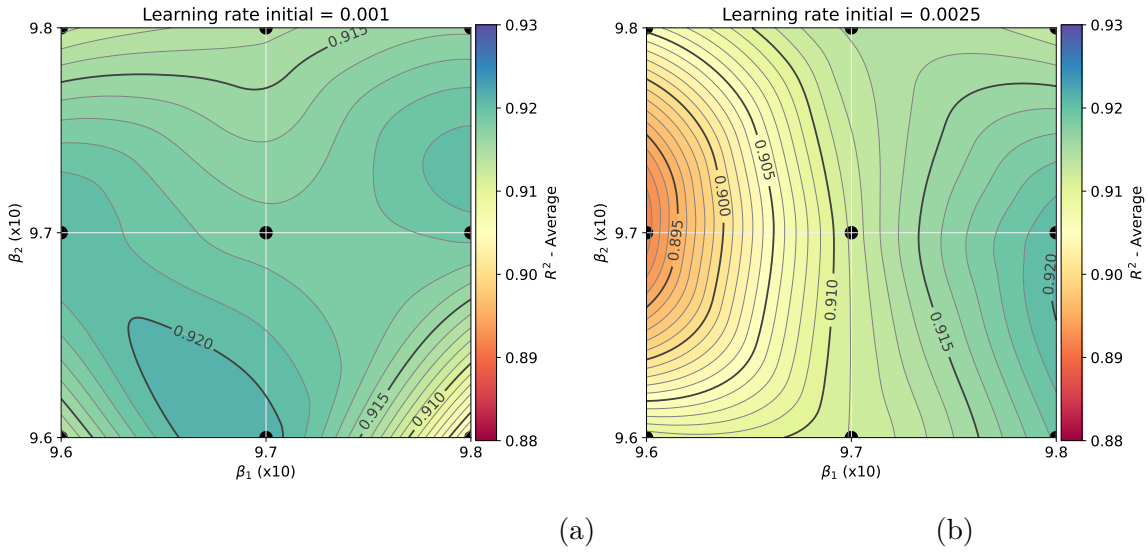


Figure 6.5: Grid search on the Adam's optimal hyperparameters (a) $\eta = 0.001$, (b) $\eta = 0.0025$.

After all the analyses, it is concluded that the best artificial neural network for training with the stress-strain database of the present study is the network with architecture $(300, 600)$, trained with normalized database between $[-1, 1]$, with the Adam optimization algorithm, and hyperparameters $\eta = 0.001$, $\beta_1 = 0.97$ and $\beta_2 = 0.96$.

Fig. 6.6 schematically presents the architecture selected for proceeding with this project.

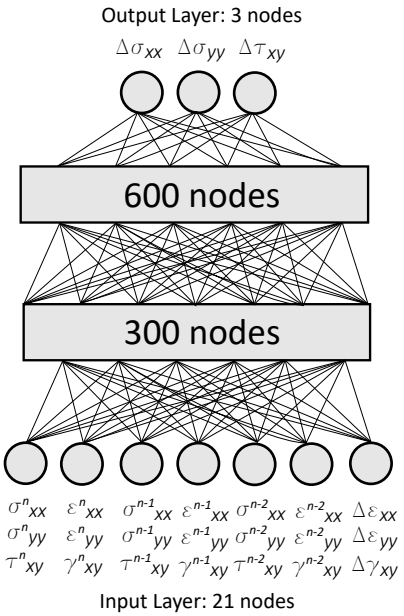


Figure 6.6: Optimized Architecture

Chapter 7

Training Process

This chapter presents the methodology used in training the artificial neural network, the importance of ML model explainability, and an analysis of the proposed model by evaluating the impact of the input variables on the output variables.

7.1 Methodology

After selecting the training dataset and defining the model architecture and hyperparameters, the network training proceeded. A major concern for ML designers is model overfitting. As discussed earlier, overfitting occurs when the network memorises the patterns in the training data and is unable to perform generalisations. In this work, particular attention was given to this issue. In order to avoid overfitting, first, the network with the optimal capacity within the assessed possibilities, obtained through the hyperparameter optimization analysis, was considered (Chapter 6). This analysis was performed via cross-validation technique and with the help of sklearn ML libraries, which implement the L2 regularization technique as standard. Among the best evaluated networks with similar performance, the simplest one was chosen, respecting the principle of parsimony. In addition, a large training database was used, larger than the number of parameters to be learned by the network, as recommended in the literature. Then the final training of the model proceeds.

The input and output data were preprocessed with the data normalization technique to the range of [-1, 1] and the Adam optimization algorithm was used with the parameters $\eta = 0.001$, $\beta_1 = 0.97$, $\beta_2 = 0.96$ and $\varepsilon = 1 \times 10^{-8}$, as defined in Chapter 6. Also, batch size equal to the total number of the dataset divided by 100, activation function ReLu in the hidden layers and Identity in the output layer, and cost function MSE were used. Eq. (7.1) presents the architecture of the MLP.

$$\{\Delta\sigma\} = \mathbf{NN}(\{\Delta\varepsilon\}, \{\sigma\}_n, \{\varepsilon\}_n, \{\sigma\}_{n-1}, \{\varepsilon\}_{n-1}, \{\sigma\}_{n-2}, \{\varepsilon\}_{n-2} : 21 \mid 300 \mid 600 \mid 3) \tag{7.1}$$

For training it was used the checkpoints strategy (Lakshmanan et al., 2020). This methodology consists in saving trained models systematically or not, after a certain number of epochs, storing the synaptic weights of the model. Thus, the model can be used for predictions or to continue in-process training later. This technique is suitable for long trainings, in order to avoid rework or loss of information in the event of a failure, such as a power outage, internet connection drop, or some other unexpected error. Also, saving the model at a particular frequency enables one to make comparisons between the performance of networks trained after different numbers of epochs, and even do different types of analysis by adding more data in a progressive training, changing hyperparameters, etc. Then, the network was trained for a certain number of epochs, the smallest cost function reached in this period was evaluated, and training was continued until the cost function reached the smallest value achieved so far. Thus, the network predictions were evaluated with the metrics MAE and R^2. This process was repeated until no significant improvement in model performance was observed. The fact that the network does not show significant improvement after a certain number of epochs means that it has reached saturation. The network was trained for up to 2.356 epochs. Fig. 7.1 shows the error evolution curve (loss curve) over the training epochs.

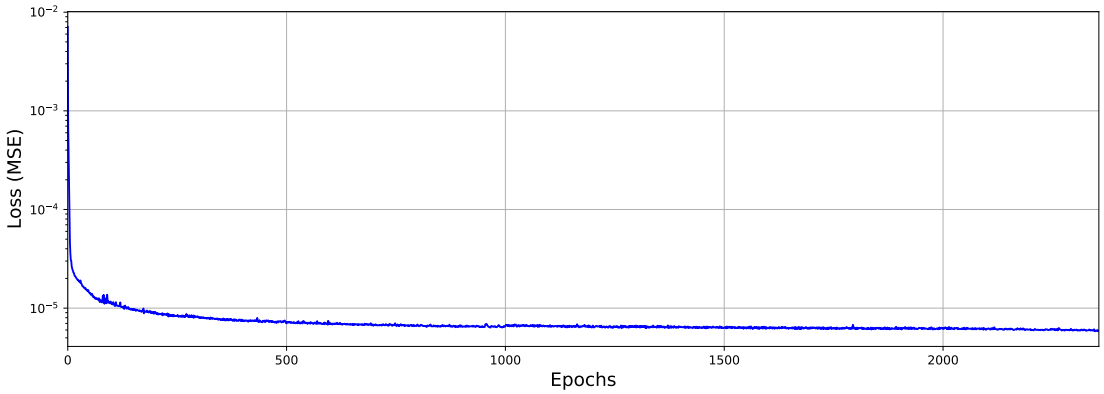


Figure 7.1: Loss Curve

After training, the model's explainability analysis was performed, which will be presented next.

7.2 ML Models Explainability

ANNs, as well as other ML algorithms, are powerful predictive techniques, used in solving complex problems, and with a growing role in decision making processes. Despite being recognized as efficient, the algorithm is considered as a "black box" because has no clear explainability for its answers, and therefore, the target of much criticism. Explainability is the ability of the model to provide clarity in its results, in order to help the user make decisions based on the data in a secure and auditable way. In reality, from a practical and regulatory point of view, not clearly understanding the aspects that lead the model to make a certain decision can make its use unfeasible. This has motivated the emergence of a new and broad area of research that develops studies and methods that make it possible to understand the decisions of these models, by analyzing the influence and impact of each input variable on the output variables (Ras et al., 2022). In addition to enabling the understanding of model decisions, these methods can facilitate the understanding of the complex nonlinear relationships present in the data and bring critical inshights to the model's final applicability.

The methods that have been developed can generally be classified into two distinct groups: intrinsic methods and post hoc methods. Intrinsic methods apply to models with a simple structure, such as Decision Trees, and can therefore be explained from their own arrangement. Post hoc methods, on the other hand, are applied to more complex models after the training stage. In this case, it seeks to explain the model by analyzing its predictions. The methods can be further divided into specific or agnostic, where the specific ones are used for a single learning model and the agnostic ones are applicable to any type of model. The present work uses ANNs as a ML technique. Thus, it is necessary to use a post hoc method to explain the elaborated model.

In the landscape of post hoc models, the agnostic model SHAP, short for SHapley Additive exPlanations, stands out. This tool, elaborated by Lundberg and Lee (2017) and named after Lloyd Shapley - an American mathematician and Nobel laureate, who developed the idea; implements game theory-based processes to evaluate the importance of each input variable on the model's responses. To better understand how the SHAP method works, it is first important to understand the main concepts on which it was based, being the behavior of the Local Interpretable Model-agnostic Explanations (LIME) method and the meaning of Shapley values from coalition game theory.

In front of the challenge of explaining very complex models at once, Ribeiro et al. (2016) sought to explain a single (local) prediction, in order to understand the reasons why a particular model performs a certain prediction. To this end, LIME treats the model as a black box, so that it can be applicable to any kind of model; it performs perturbations (noise) on the features and observes how the model responds to these perturbations. In other words, the technique modifies some input variables randomly and observes how this

impacts the output variables. More details about the LIME method can be found in its source article (Ribeiro et al., 2016).

Shapley values, on the other hand, are based on the mathematical foundation of coalition game theory. This technique was developed from the perspective of explaining, in a fair way, what influence each variable in a given coalition has on the value obtained through that coalition. To facilitate understanding, let C be a coalition that collaborates to produce a coalition value V. To find out how much each member i of the set C contributes to the value V, the Shapley value for each of these members is calculated. This is done by sampling a coalition that contains member i and looking at the value of the coalition formed by removing member i from that group. Then the V values of these pair of coalitions are compared and the difference between them is calculated, which is defined as the marginal contribution of member i to that sample, i.e. how much member i contributes to that particular group. Next, all possible types of coalitions are listed, which are only different from each other based on whether member i is included or not, and the marginal contribution of each of these combinations is evaluated. The Shapley value for member i is the average of its marginal contributions, that is, the average of the marginal contributions of all possible combinations with member i. Mathematically, the process boils down to the following equation:

$$\phi_i(f,x) = \sum_{S \subseteq \{x\}} \frac{|S|!\,(p-|S|-1)!}{p!} \left(f_x\left(S \bigcup \{i\}\right) - f_x\left(S\right) \right) \tag{7.2}$$

where ϕ_i is the shapley value for feature i, f is the black-box model, x the input data point, and p the total number of features. Thus, iterations are made over all possible subsets (S) of feature combinations. In the equation above, $f_x(S)$ is the model prediction over the subset S with variable i and $f_x\left(S \bigcup \{i\}\right)$ the model prediction over the subset S without variable i. This difference tells how much the variable i contributes in predicting the subset S (the marginal value). Each subset combination is weighted according to how many features within the total number of features are within that subset.

The reason that makes Shapley values a fair methodology for assigning importance is that it has the following properties:

Completeness/Efficiency: The contributions of the features should be equal to the sum of the differences between the predictions of the examples and the average of all predictions of the model. Mathematically, it follows that:

$$\sum_{i=1}^{p} \phi_i = f(\mathbf{x}) - E(f(\mathbf{X})) \tag{7.3}$$

Dummy: Features that are not used by the model have zero contribution. Mathematically, it follows that:

if

$$f_x\left(S \cup \{i\}\right) = f_x\left(S\right)$$

for all

$$S \subseteq \{1, ..., p\}$$

then

$$\phi_i = 0$$

Symmetry: The contributions of features i and j must be the same if the two contribute equally to the game in all possible coalitions. Mathematically,

if

$$f_x\left(S \cup \{i\}\right) = f_x\left(S \cup \{j\}\right)$$

for all

$$S \subseteq \{1, ..., p\} \setminus \{i, j\}$$

then

$$\phi_i = \phi_k$$

Monotonicity: If a feature i is consistently contributing more to the "game" (prediction) than feature j, then the Shapley value of feature i should reflect this and be greater than that of feature j;

Linearity/Additivity: If the prediction of a model (reward of a "game") is given by the sum of two intermediate values ($val + val^{+}$), each of which is given from the values of its features, then the contribution of each feature is given by the sum of its contributions in the intermediate predictions ($\phi_j + \phi_j^{+}$).

In a nutshell, the Shapley value problem evaluates how members of a coalition contribute to the value of that coalition. Based on Shapley value, SHAP explains how each of the individual features contribute to the output of the model. SHAP is an additive feature assignment method, defined by Lundberg and Lee (2017) as follows: be an input dataset x and a model $f(x)$, a simplified local input dataset x' and an explanatory model $g(x')$ are defined. The conditions that must be guaranteed are that if x' is approximately equal to x, then $g(x')$ must be approximately equal to $f(x)$,

$$x \approx x'$$

$$f(x) \approx g(x') \tag{7.4}$$

and that $g(x')$ must take the form of Eq. (7.5)

$$g(x') = \phi_0 + \sum_{i=1}^{N} \phi_i x'_i \tag{7.5}$$

where $\phi_0 = E(f(\mathbf{X}))$, that is, the average output value of the model and ϕ_i is the explanatory effect of feature i, that is, how much feature i changes the output of the model. This is called attribution. If these two conditions are met, you have an explanatory model with additive feature attribution.

But how to remove a feature from an ML model? Actually, the models have a fixed input vector size that can not be changed. SHAP solves this problem by replacing the features to be excluded with random values from the training database, because random values have no predictive power. Calculating the shapley values still requires a very high computational cost due to the required permutations being of the order of magnitude of 2^n, where n is the total number of features. The basic idea presented by Lundberg and Lee (2017) is to simply approximate the shapley values instead of calculating all combinations. KernelSHAP (code used in this work), for example, in possession of a sample set of inputs, fits a linear regression model based on those samples. The variables in this linear regression model consider only whether the feature is present or not. After training, the coefficient values of this linear model are interpreted as approximate shapley values, as done in LIME.

The authors also described a set of three desirable properties of such an additive feature method: local accuracy, missingness and consistency.

Local accuracy: It states that if the input and the simplified input are approximately equal, then the real model and the explanatory model should produce approximately the same response.

Missingness: It states that the only thing that can affect the output of the explanatory model is the inclusion of features, not exclusion, and therefore if a feature is excluded from the model, its assignment should be zero.

Consistency: It states that if the original model changes so that the contribution of a particular feature changes, the attribution in the explanatory model can not change in the opposite direction.

The authors argue that SHAP is the only method that satisfy all three properties, while, LIME, for example, only satisfies local accuracy. A major advantage of SHAP is that in addition to presenting individual explanations, it can also be used to construct global explanations, through the combination or average across all local instances. SHAP has implementations of summary graphs that condense the explainability of multiple instances and allow you to understand the behavior of the model from a global perspective. In the following, local and global analyses of the model developed in this work will be presented, on top of the training data, with the help of the SHAP package and its methods.

7.3 Local and Global Explanations

In short, a local method clarifies how a model makes decisions for a single forecast, while a global method, in turn, allows one to understand the overall structure of a model and how it makes its decisions. For the evaluation of an individual prediction, the graphical resource known as Waterfall Plot is used. This tool allows us to see the amplitude and the nature of the impact of a feature and also the order of importance of the features (in descending order, through the y-axis), besides the values taken by each feature for the sample. Here, on the x-axis, the average of the model's predictions and the final predicted value of the model (through the vertical line) are observed. Colors indicate whether the impact is positive (red) or negative (blue) and arrows indicate the direction of this impact. For the global analysis, this work uses the Summary Plot method. This method presents on the y-axis, also in descending order of importance, the features most influential in the model responses. The x-axis presents a scale for the SHAP values, with a center line separating the positive values on the right, which pull the model prediction up, and the negative values on the left, which pull the model prediction down, numerically speaking. The colors of the graph, on the other hand, represent the relative size of the feature values, so that high values are colored red, low values blue, and middle values in shades in between the two extreme colors, as per the color map on the right of the graph. This features enables the user to identify the distribution of SHAP values; which input value most influences the output value and how strong this influence is; whether the input values influence the output value to be low or high values; and how the relative values of each variable influence the model responses. It can be seen that when the variable has no relevant importance in the model decisions, the SHAP values are close and concentrated in the center of the vertical separating line.

7.3.1 Elementary States

The local and global analyses for the elementary states were performed over 4 different tests: a plate with tension-tension stresses with stress ratio $|\sigma_2/\sigma_1| = 0.2$ (test 1), a plate with compression-compression stresses with stress ratio $|\sigma_1/\sigma_2| = 0.9$ (test 2), a plate under tension-compression stresses with stress ratio of $|\sigma_1/\sigma_2| = 0.7$ (test 3) and the same plate under tension-compression stresses but with 45° rotated states (test 4). This last analysis was performed to observe the model behavior in respect to the predictions of the shear stress increments, since in the first 3 tests, the shear stresses are null. It is important to emphasize that none of these tests were used in the MLP training and that this information, although similar to that present in the final dataset composition, is unknown to the MLP. In this case, the FEM results is the same at any of the integration points. Therefore, the local analysis was performed for the integration point of the finite element 1 of each of the tested plates. The stress-strain curves obtained via FEM and

predicted by MLP will be also plotted. Details of obtaining the stress-strain curve via MLP will be discussed in the beginning of Chapter 8.

First, it was evaluated the plate under tension-tension stresses (test 1). In this test, rupture occurs by tension in the x-direction. Therefore, for local explanation, plots were made for points A and B on the ascending and descending branches of the stress-strain curve, for the stress increment component in the x-direction ($\Delta\sigma_{xx}$), according to Fig. 7.2.

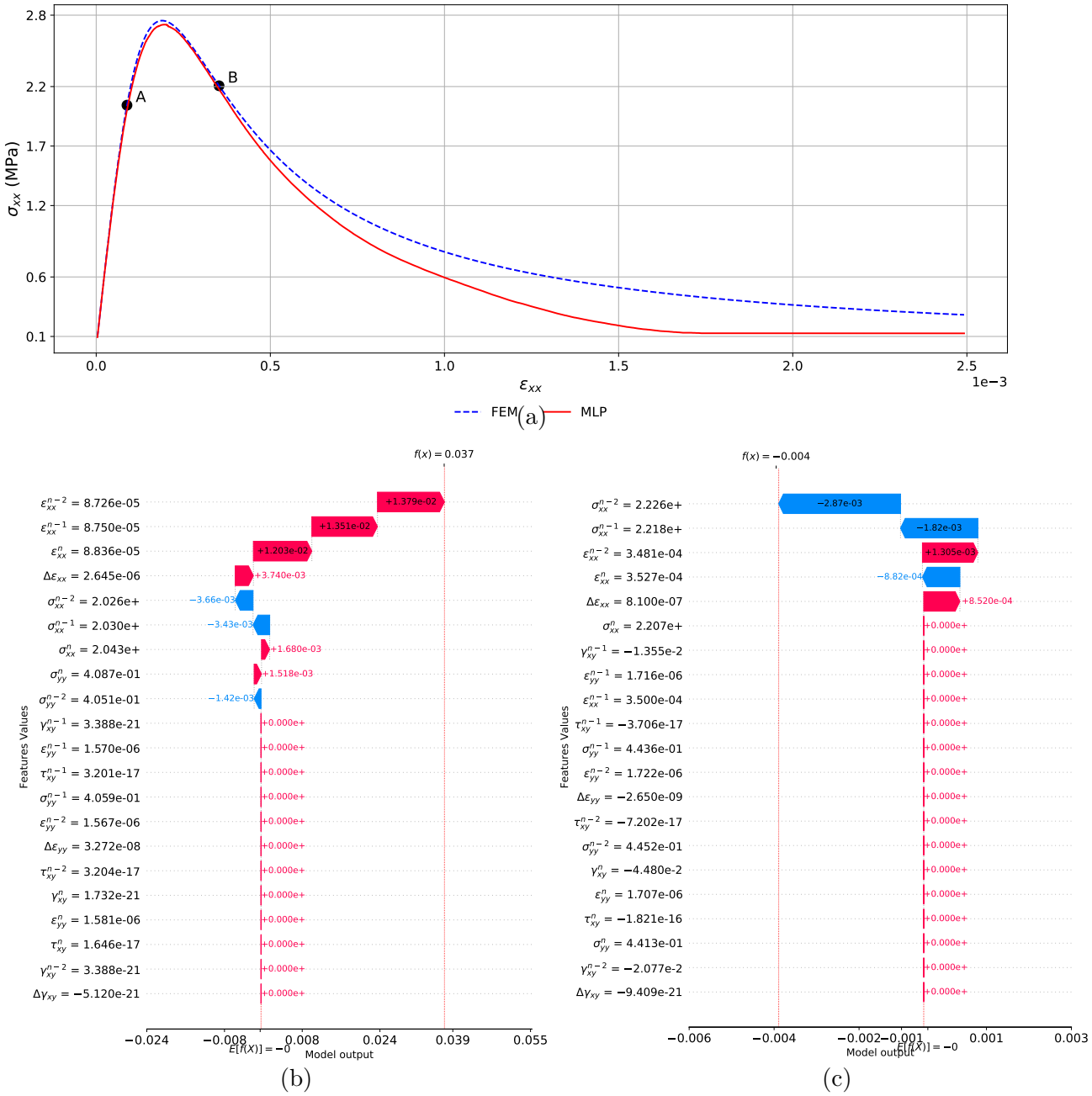


Figure 7.2: Local explanation for the plate under biaxial tension-tension state. (a) Comparison between FEM and MLP results - $\sigma_{xx} \times \varepsilon_{xx}$, (b) Feature importance for output $\Delta\sigma_{xx}$ for point A in the ascending branch, (c) Feature importance for output $\Delta\sigma_{xx}$ for for point B in the descending branch.

For global explanation, there is:

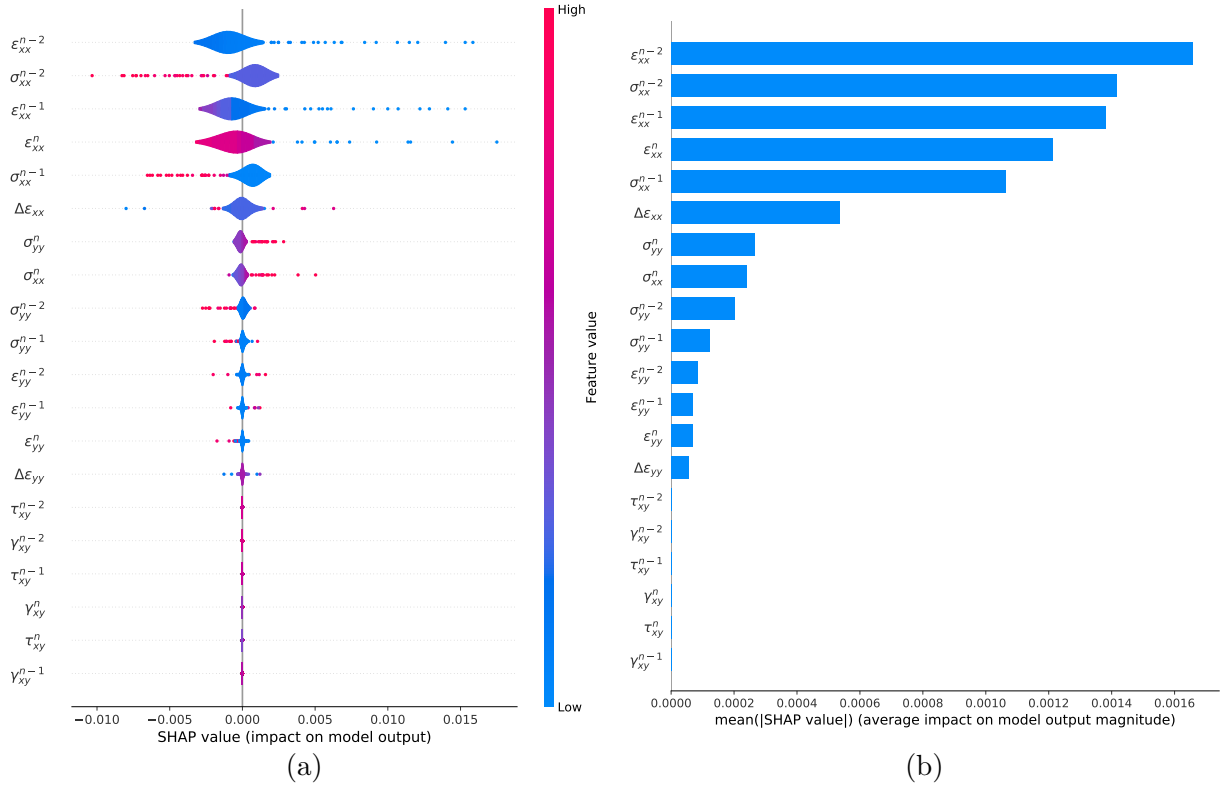


Figure 7.3: Global explanation for the plate under biaxial tension-tension state - Feature importance for output $\Delta\sigma_{xx}$ (a) SHAP value, (b) Mean SHAP value.

Evaluating the shap values, both in the local and global analysis, it is observed that the variables that exert the greatest influence on the prediction of the stress increment at x are the stresses, strains and strain increments in the x-direction. It can also be seen that the shear variables do not impact the network predictions for this dataset, since the shear stresses for this test are zero. Both observations are consistent with the expected.

Next, the plate under compression-compression stresses (test 2) was analysed. In this test, the rupture occurs by compression in the y-direction. Similar to the previous plate, for local explanation of the stress increment component in the y-direction ($\Delta\sigma_{yy}$), one has (Fig. 7.4):

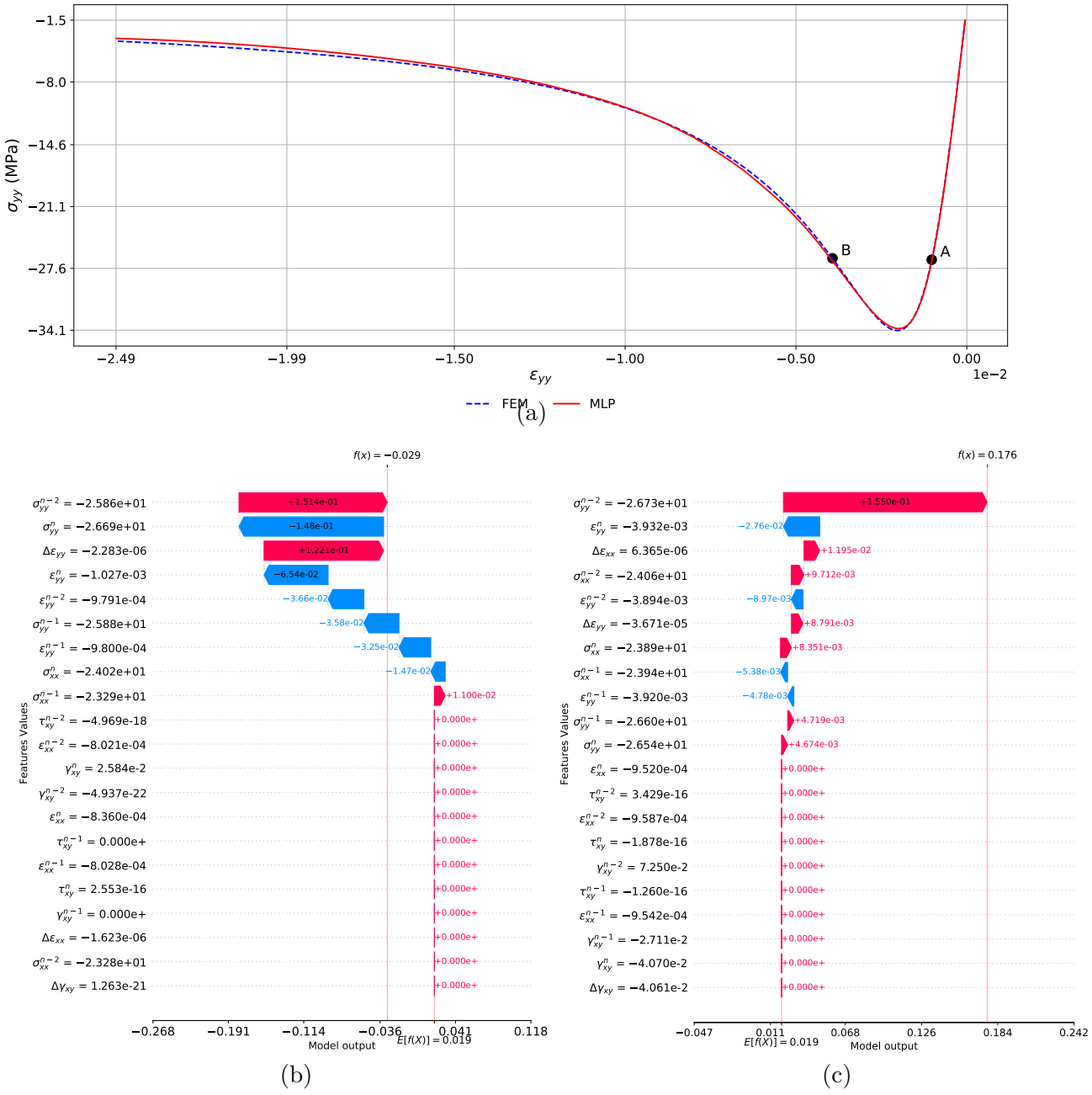


Figure 7.4: Local explanation for the plate under biaxial compression-compression state. (a) Comparison between FEM and MLP results - $\sigma_{yy} \times \varepsilon_{yy}$, (b) Feature importance for output $\Delta\sigma_{yy}$ for point A in the ascending branch, (c) Feature importance for output $\Delta\sigma_{yy}$ for point B in the descending branch.

For global explanation, there is:

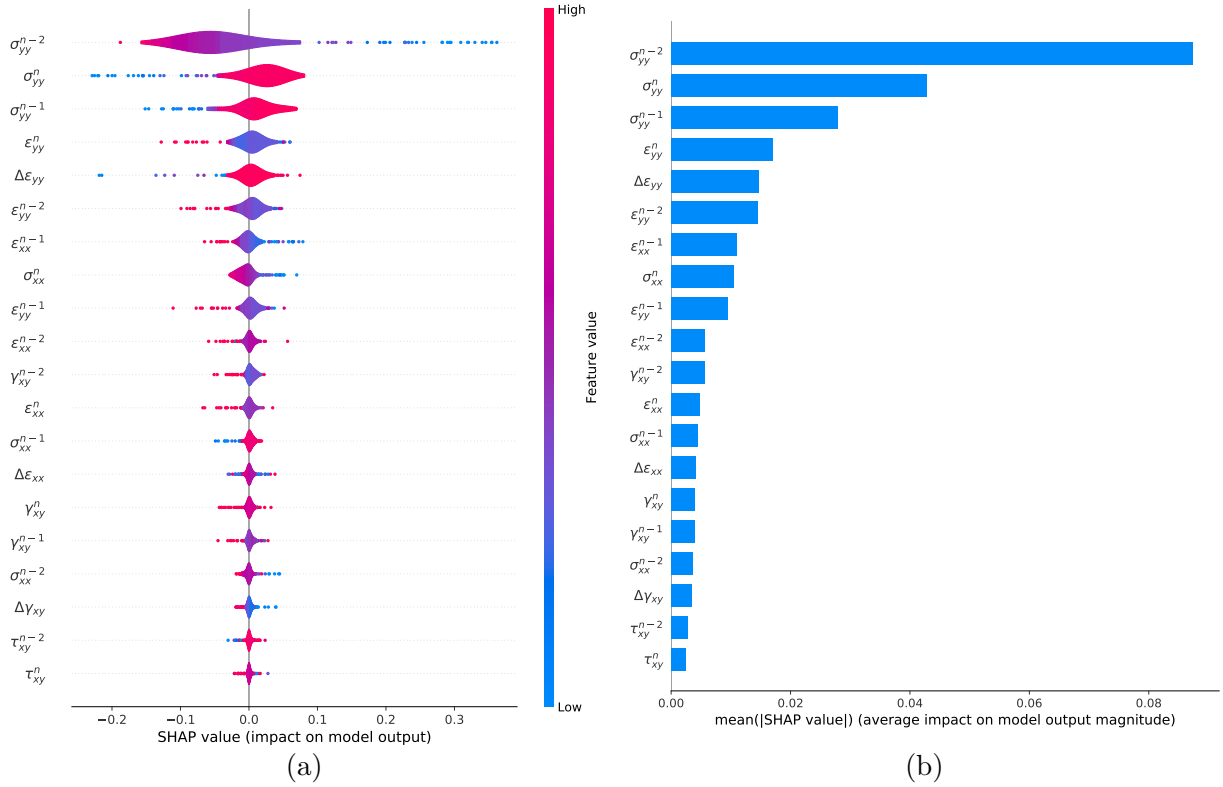


Figure 7.5: Global explanation for the plate under biaxial compression-compression state - Feature importance for output $\Delta\sigma_{yy}$ (a) SHAP value, (b) Mean SHAP value.

Similarly and consistent to the previous plate, it can be seen that the variables that have the greatest influence on predicting the stress increment in y are mostly the stresses, strains, and strain increment in the y-direction, while the shear variables, which are null, remain not significantly influencing the model predictions.

Then the plate under tension-compression stresses was investigated (test 3). In this test, rupture occurs by tension in the x-direction. Thus, for local explanation of the component of stress increment in the x-direction ($\Delta\sigma_{xx}$), one has (Fig. 7.8):

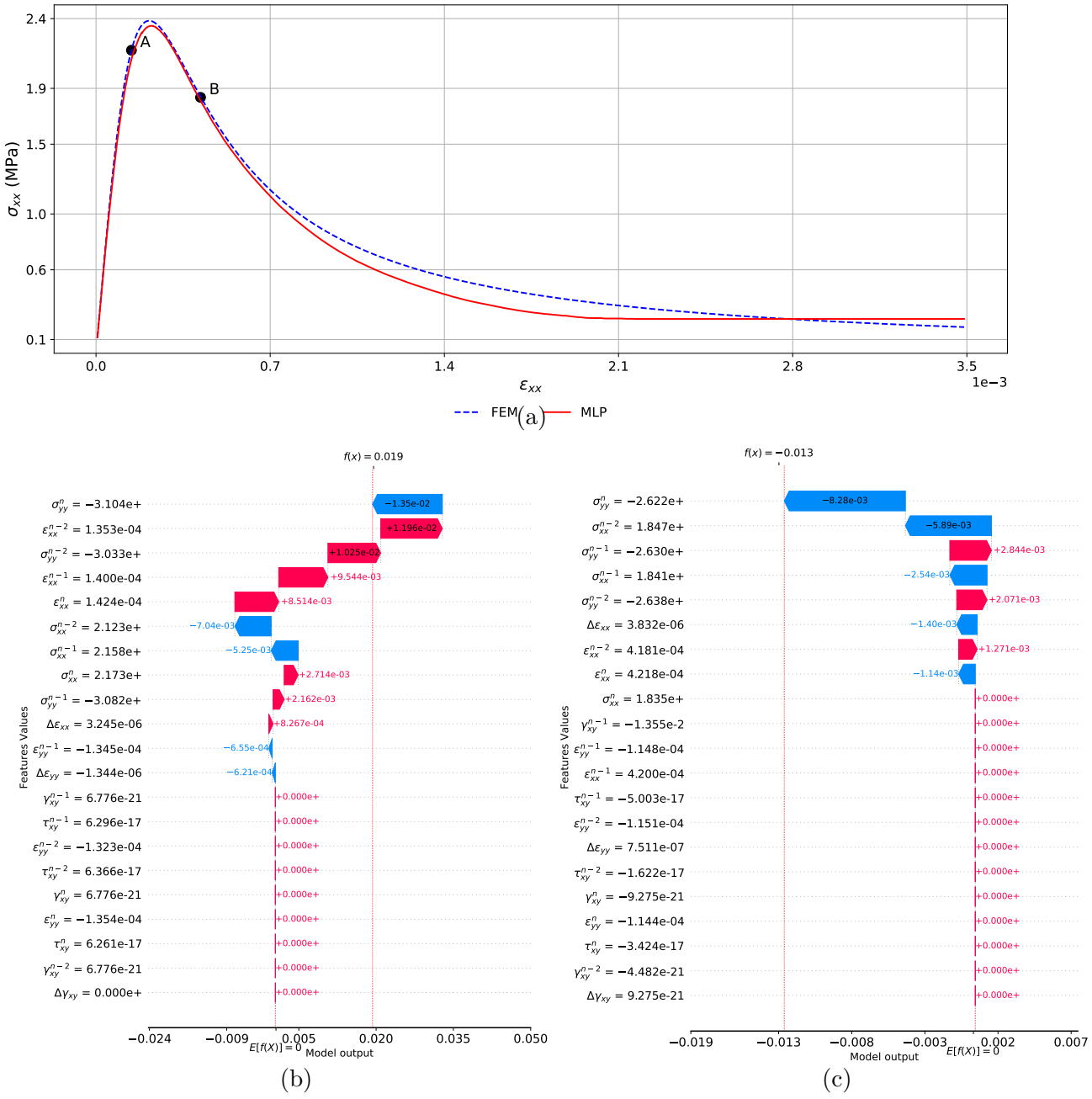


Figure 7.6: Local explanation for the plate under biaxial tension-compression state. (a) Comparison between FEM and MLP results - $\sigma_{xx} \times \varepsilon_{xx}$, (b) Feature importance for output $\Delta\sigma_{xx}$ for point A in the ascending branch, (c) Feature importance for output $\Delta\sigma_{xx}$ for point B in the descending branch.

For global explanation, there is:

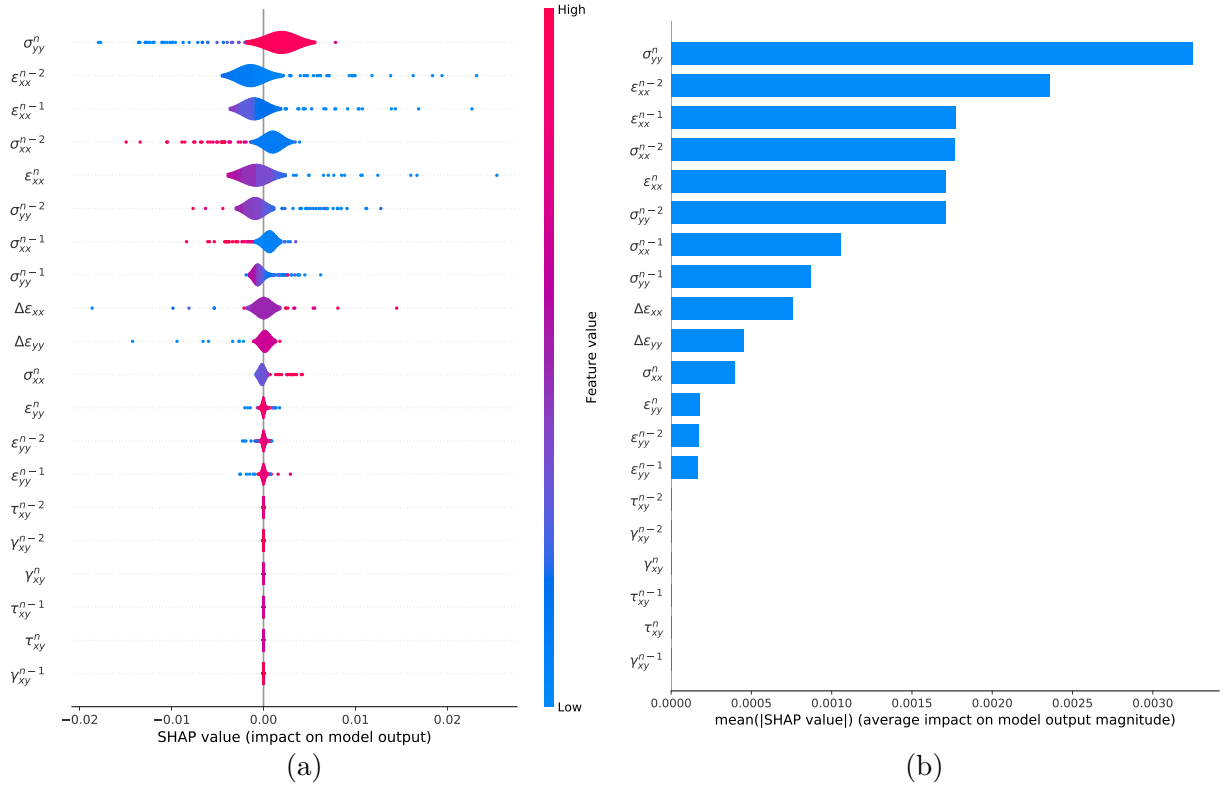


Figure 7.7: Global explanation for the plate under biaxial tension-compression state - Feature importance for output $\Delta\sigma_{xx}$ (a) SHAP value, (b) Mean SHAP value.

This time, in both local and global analyses, it was observed that there is not a clear division of importance between the groups of features in the x and y directions, as noticed in the previous examples. In this case, for instance, the strain variable in y-direction assumed the feature position that most impacts the predictions of the strain increment component in x. The fact is that it is expected that the variables referent to one direction impact the prediction of the stress increment in the other direction, as established by the Poisson effect, and that this relationship becomes more difficult to interpret and understand as the stress states become more complex.

As noted, in all 3 tests studied above there are no shear stresses and the variables related to shear stresses are just numerical noise, i.e., very small values and close to zero, and in all the above cases, it has no or very little influence on the model predictions (as verified through the graphs). To check the behavior of MLP for a case in which there are shear stresses, the plate under tension-compression stress was rotated by 45° (test 4), and the local and global impacts were evaluated for the shear stress increment variable, according to Fig. 7.8 and Fig. 7.9, in this sequence.

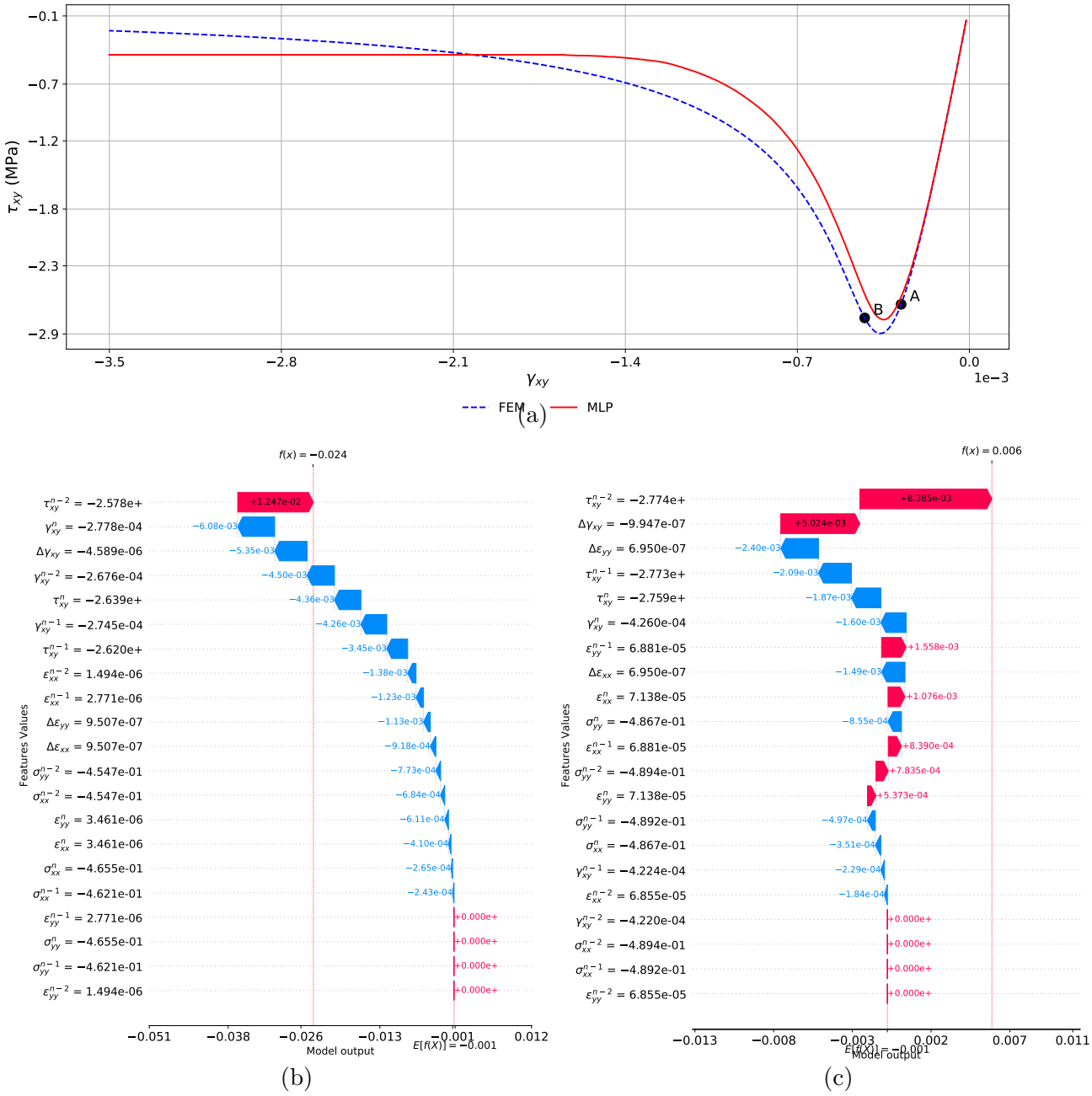


Figure 7.8: Local explanation for the plate under biaxial tension-compression state rotated by 45° - $\sigma_{xx} \times \varepsilon_{xx}$, (b) Feature importance for output $\Delta\sigma_{xx}$ for point A in the ascending branch, (c) Feature importance for point B in output $\Delta\sigma_{xx}$ for the descending branch.

For global explanation, one has:

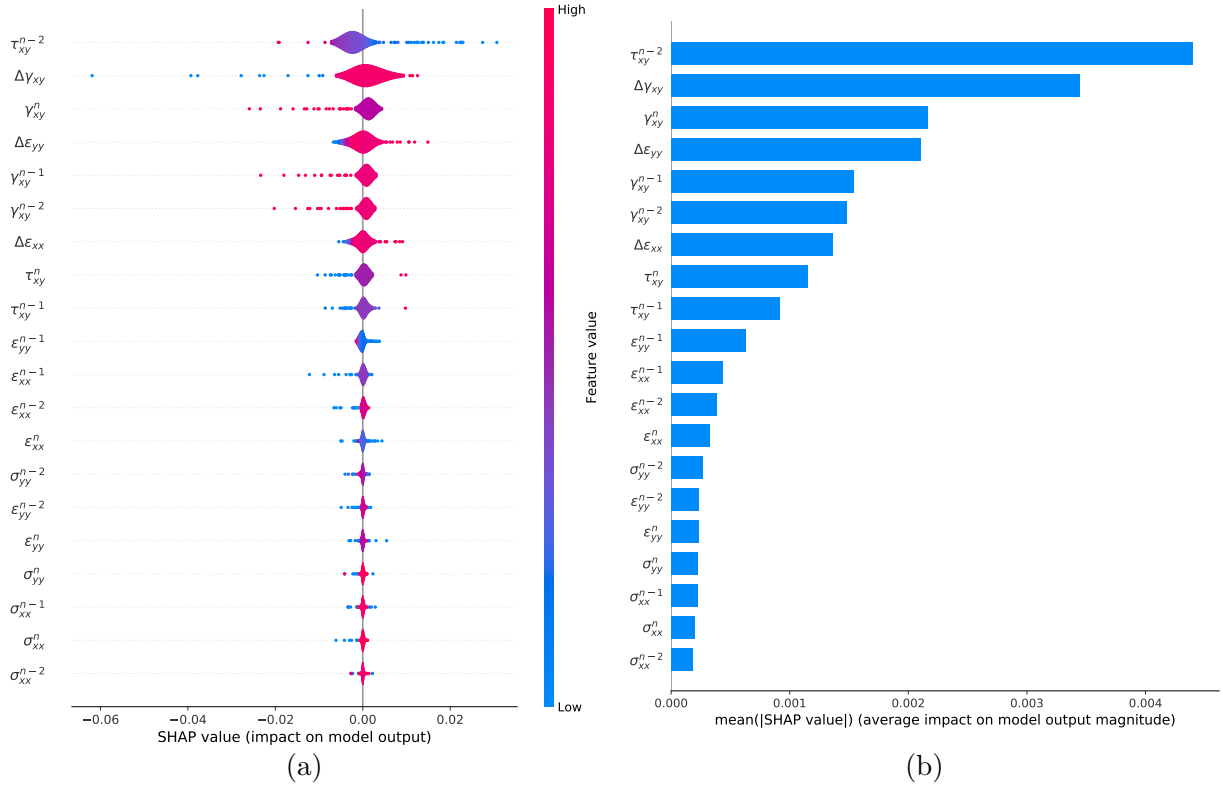


Figure 7.9: Global explanation for the plate under biaxial tension-compression state rotated by 45° - Feature importance for output $\Delta\gamma_{xy}$ (a) SHAP value, (b) Mean SHAP value.

It can be seen that for shear stress increment ($\Delta\tau_{xy}$), the most important variables, according to the shap values, are also the input variables related to shear, appearing at the top of the impact scale in both local and global analysis, as expected.

7.3.2 4-Point Bending Test

This subsection presents the local and global explainability analysis of the beam under 4-point bending data. To facilitate the explainability of the predictions, only the data outlined in red at Fig. 7.10 was selected. This region was chosen in order to verify the behavior of the model for the specific pure bending problem. This means that the average of the model predictions ($E(f(\mathbf{X}))$) will be calculated from the predictions of only the data in this region.

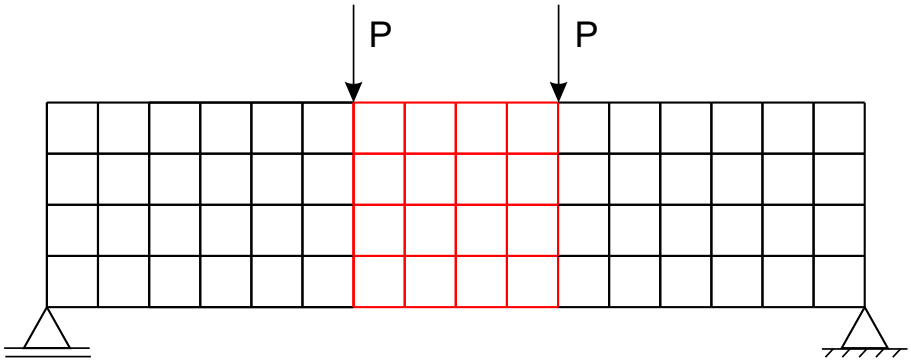


Figure 7.10: Numerical model used to generate data for the consistency study - Demarcation of the data extraction region for SHAP analysis.

Next, the distribution of the data present in the pure bending region was analyzed through the current stresses and strains and the incremental strains.

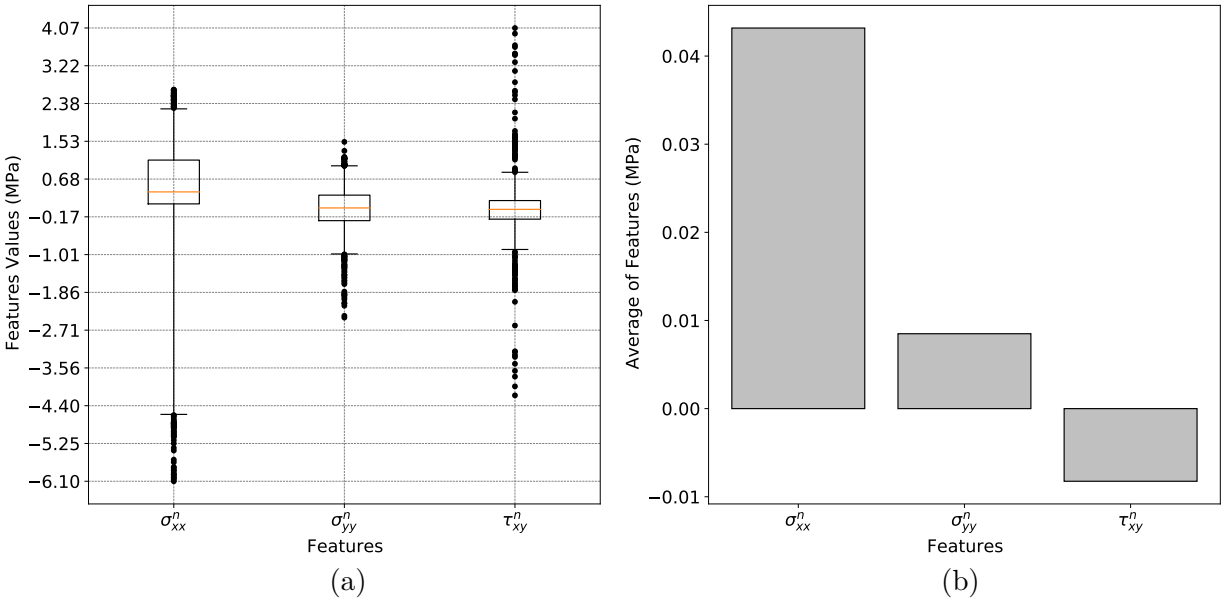


Figure 7.11: Dataset stress input variables distributions. (a) Features distributions, (b) Average of features.

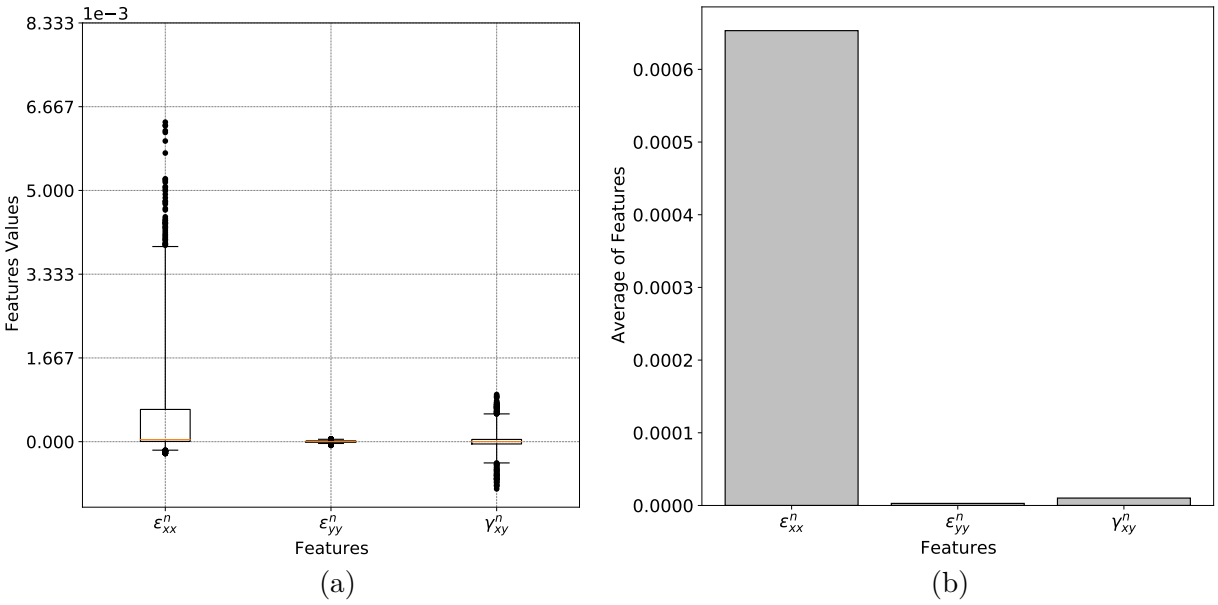


Figure 7.12: Dataset strain input variables distributions. (a) Features distributions, (b) Average of features.

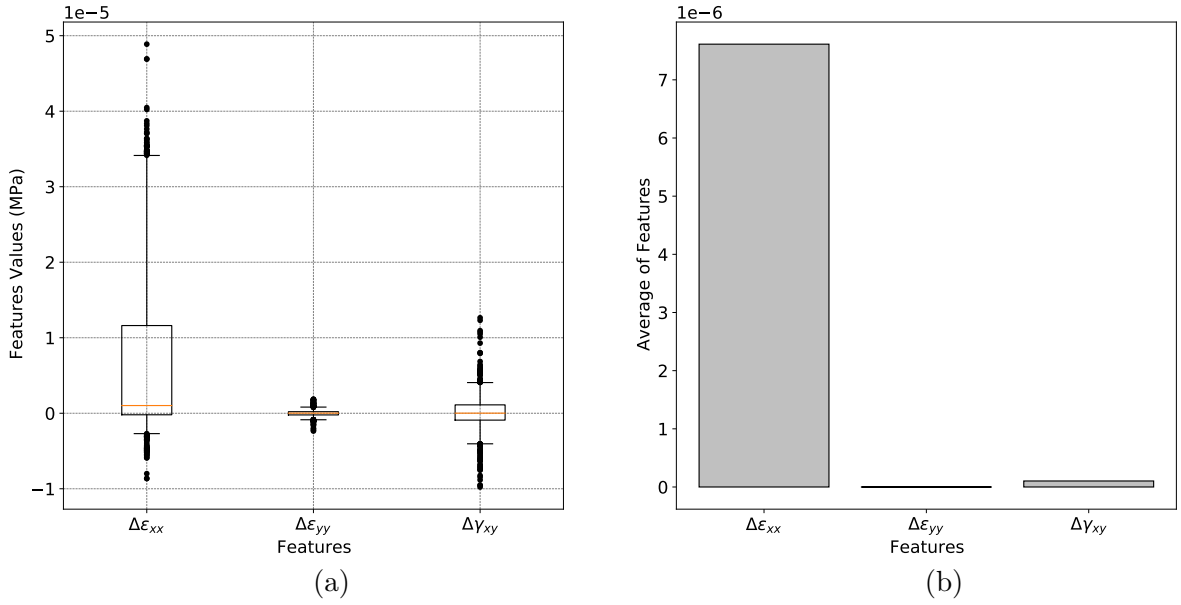


Figure 7.13: Dataset strain increment input variables distributions. (a) Features distributions, (b) Average of features.

By analyzing Fig. 7.11 to Fig. 7.13, it can be seen that the data distribution faithfully represents the pure bending state, with values of shear stresses and shear strains very low and close to zero.

For the local analysis, a point in the lower central region of the beam was chosen, as illustrated by Fig. 7.14. In this example, only the output variable stress increment at x-direction ($\Delta\sigma_{xx}$) was evaluated.

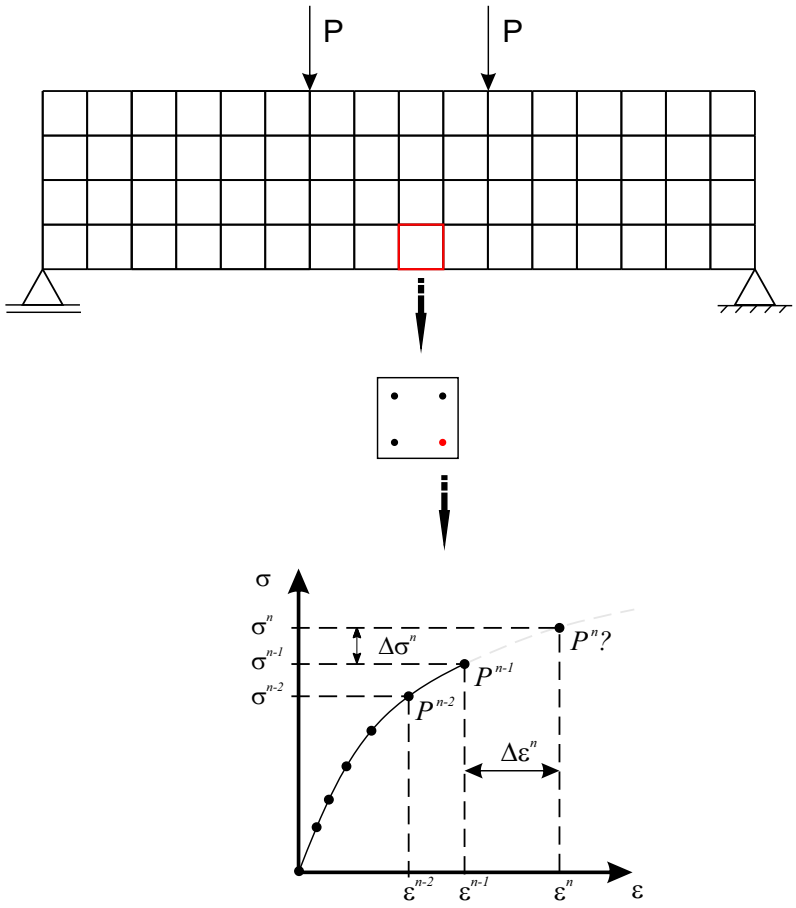


Figure 7.14: Numerical model used to generate data for the consistency study - Demarcation of the selected point for SHAP analysis.

Fig. 7.15 brings the MLP-predicted stress-strain curve and the waterfall plot for the selected point and component.

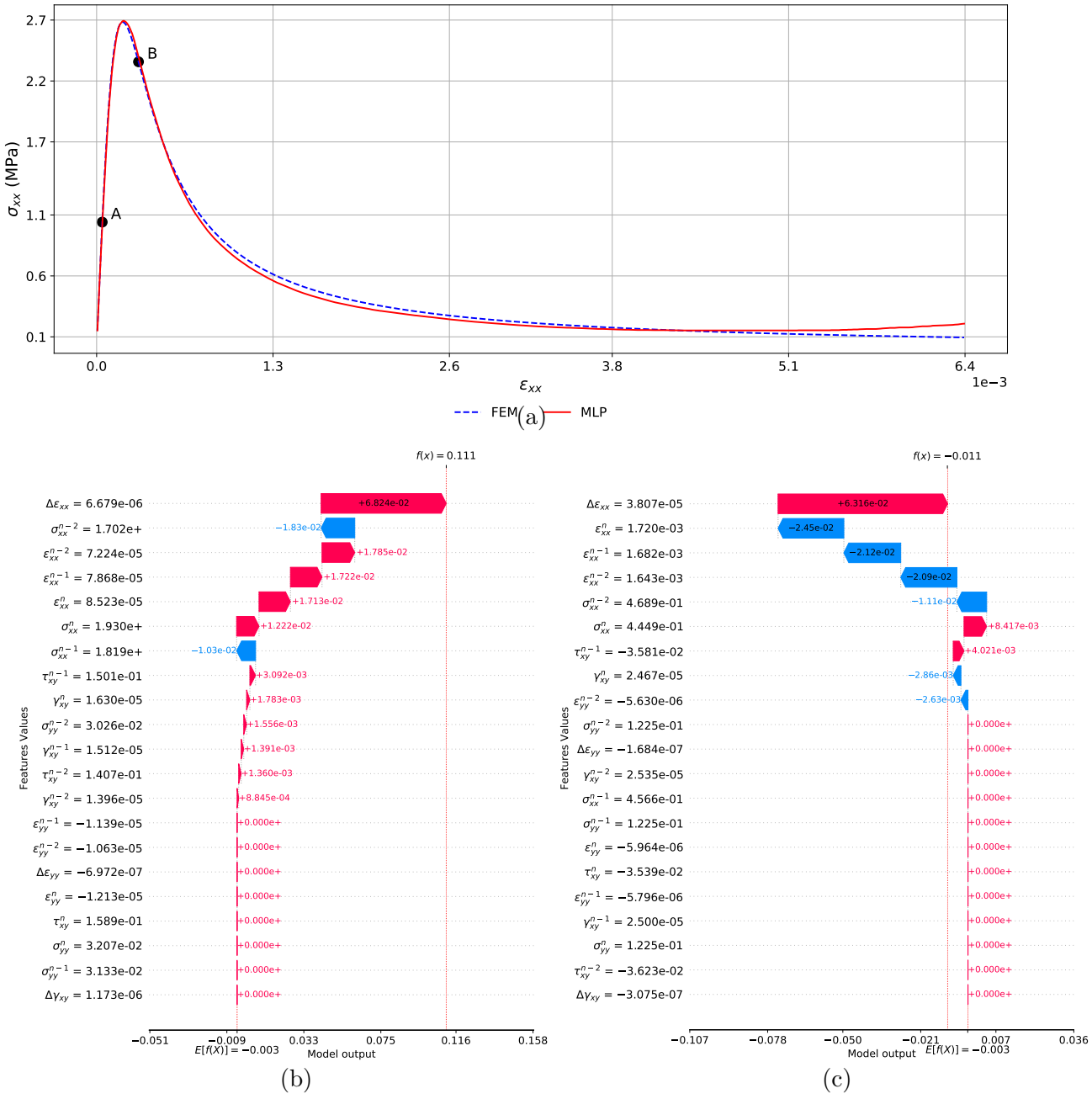


Figure 7.15: Local explanation for the 4-point bending test in the pure bending region - (a) Comparison between FEM and MLP results - $\sigma_{xx} \times \varepsilon_{xx}$, (b) Feature importance for output $\Delta\sigma_{xx}$ for the ascending branch, (c) Feature importance for output $\Delta\sigma_{xx}$ for the descending branch.

It is noticeable that the input variables that have the greatest impact on the output variable analyzed are the stresses, strains and strain increment in the respective direction, for both points evaluated. As this point is mostly under normal stresses in the x-direction, while the normal stresses in y-direction and the shear stresses are very low, the variables related to the last two mentioned stress components should not and did not influence the prediction of stress increment in the x-direction, and therefore, the observed results

presented themselves coherent and consistent with the reality of the problem.

For the global analysis, the influence of the features on the predictions of all curves present in the region under pure bending of the beam under 4-point bending was evaluated. By analyzing Fig. 7.16, it can be seen that the general behavior of the entire dataset in this region is the same as observed in the individual analysis. For the main stress component of this structure, in the region of interest, the features that most influence the MLP prediction are those that have the same direction as the output variable (x-direction).

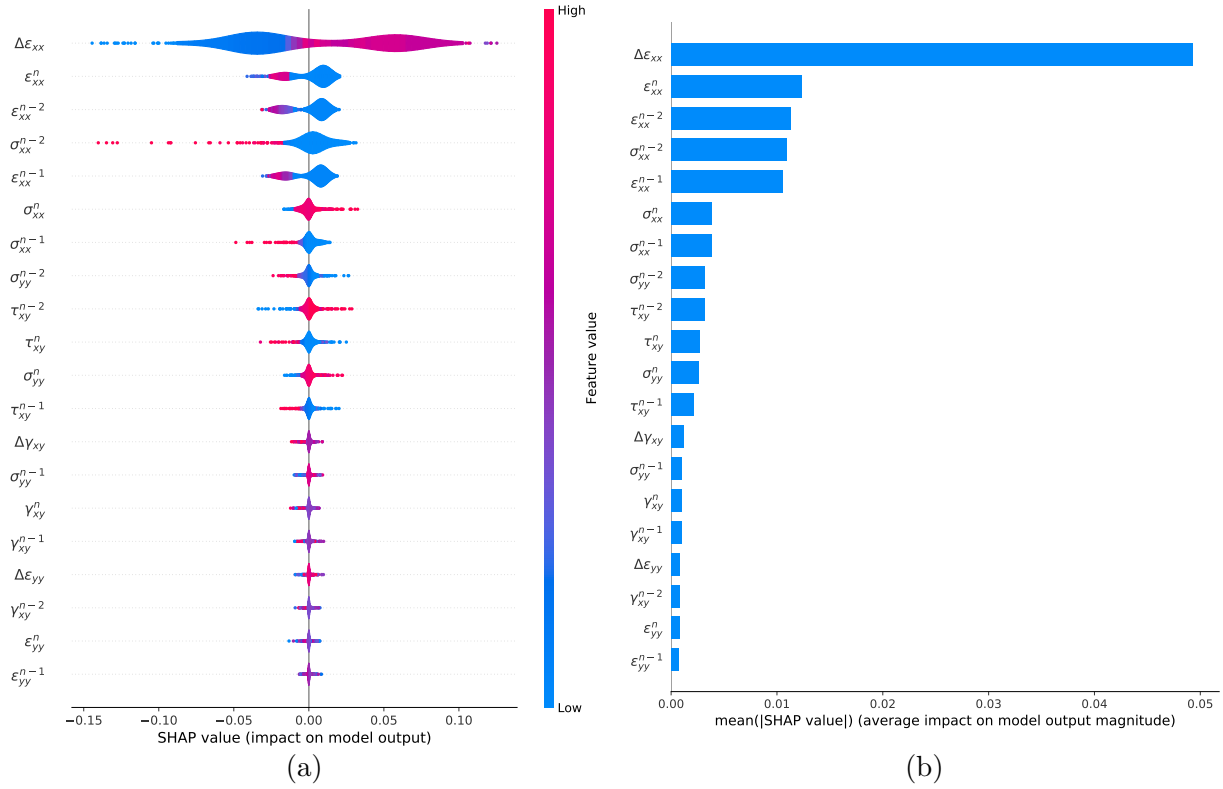


Figure 7.16: Global explanation for the 4-point bending test in the pure bending region - Feature importance for output $\Delta\sigma_{xx}$ (a) SHAP value, (b) Mean SHAP value.

In a straightforward way, from the previous analyses, both for elementary states and pure bending, it can be concluded that the decisions made by the model are not random and that there is physical consistency in the predictions, that is, the variables that most impact the prediction of the cases analyzed are those that somehow are most intrinsically intrarelated.

Chapter 8

Validation

This chapter presents the analysis and validation of the predictions of the neural network-based constitutive model for unknown structural systems.

After defining the hyperparameters (Chapter 6), the neural network was trained with the dataset of numerical results of the structural systems presented in the Chapter 5 and, finally, the physical consistency of the model was verified for the nature of the data of this study (Chapter 7). Next, the model performance analysis (generalization capability) will be presented, based on the following test structural systems and their respective main objectives:

- **4-point shearing test** - To observe the performance of the model for the mixed failure mode. This test is the proof of generalization of the MLP, as the network has only had contact with data of this nature indirectly;
- **Asymmetric traction test** - To observe the network behavior for the tensile failure mode, in which the rupture occurs by routing from one failure to another;
- **L-shaped panel** - To verify the model predictions for the bending failure mode. This failure is well defined in the training dataset, however, a different structural system is evaluated here;
- **Brazilian splitting test** - To observe if the network is able to capture the phenomenon of size change and its impacts on the material response. The MLP will be tested over the diametrical compression test with 4 different cracks from those used in the training;
- **3-point bending test** - To perform a mesh dependency analysis, i.e., whether the model performs well regardless of the level of discretization of the finite element mesh. For this purpose, a beam experimentally tested in the laboratory with the material defined in this study was numerically simulated;

To verify the results, it was developed a method that, to predict the stress-strain curve of any point, simulates the use of the MLP in a nonlinear analysis via FEM. Thus, given a stress state (n), historical stress states $(n-1)$ and $(n-2)$, and strain increments (n),

as presented in Table 5.1, the neural network predicts the respective stress increment (n). In this format, such method uses the stress increments (n) predicted by the network to calculate the current $(n+1)$ stresses, which will be used as input data for the next predictions ($\{\sigma\}_{n+1} = NN(\{\Delta\sigma\}_n) + \{\sigma\}_n$), similarly to how Newton Raphson's process works. Soon, the network predictions for step (n) feed the model as input data for the following predictions $(n+1)$. At each prediction, the current stresses are updated with the stresses increments predicted by the network, and the other variables, historical stresses, strains, historical strains and strain increments, are kept the same as in the original dataset. Finally, the results of the MLP were compared with the results of the finite element analysis at the same Gaussian points. Fig. 8.1 illustrates in a simplified way the operation of the described method.

1 Loop over the m steps to be predicted from the stress-strain curve dataset
2 **for** *n = 1 to m* **do**
3 | Predict the stress increment vector from step n
4 | $\{\Delta\sigma\}_n = \mathbf{NN}(\{\Delta\varepsilon\}_n, \{\sigma\}_n, \{\varepsilon\}_n, \{\sigma\}_{n-1}, \{\varepsilon\}_{n-1}, \{\sigma\}_{n-2}, \{\varepsilon\}_{n-2})$
5 | Calculates the new state of stresses
6 | $\{\sigma\}_{n+1} = \{\sigma\}_n + \{\Delta\sigma\}_n$
7 | Save the calculated stress vector from step n
8 **end**
9 Uses the calculated stresses with the respective original strains to mount the stress-strain curve

Figure 8.1: Assembling the stress-strain curve

To evaluate the performance of the model, the Mean Absolute Error (MAE) was used as a measure. This metric was chosen because it has the same unit as the output variables (MPa), which facilitates the interpretation of the results, and also because it is less sensitive to outliers and numerical values small or close to zero.

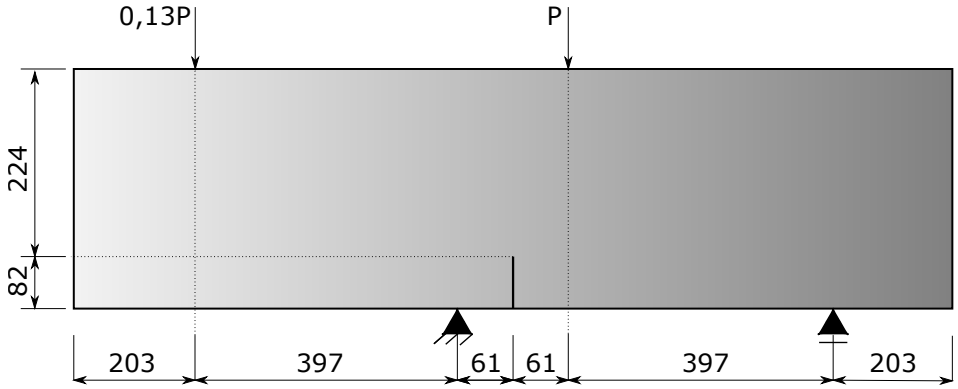
The model performance was evaluated by comparing the stresses obtained via FEM and those predicted by MLP. After that, the mean absolute accumulated error (MAE Accumulated) of each of the current stresses of each step was analyzed, fixing the color scale for the maximum MAE calculated in each structure, in order to observe the evolution of the error along the steps of the equilibrium path. For both graphs, a horizontal line and a vertical line were drawn in the regions of interest for each problem, in order to study the behavior of the quantities evaluated along these lines for a given load level. The results were observed for representative steps in the elastic and inelastic regions and for the maximum load factor, which for practical engineering is the main point of analysis. Next, predictions of the stress-strain curves in the strategic (most stressed) regions for each failure mode were presented. To complement the performance evaluation of each of the curves, graphs of percentage errors versus the structural analysis step were also plotted, with a color scale of the normalized load factor, to facilitate the understanding

of the variation of the accumulated error along the equilibrium path.

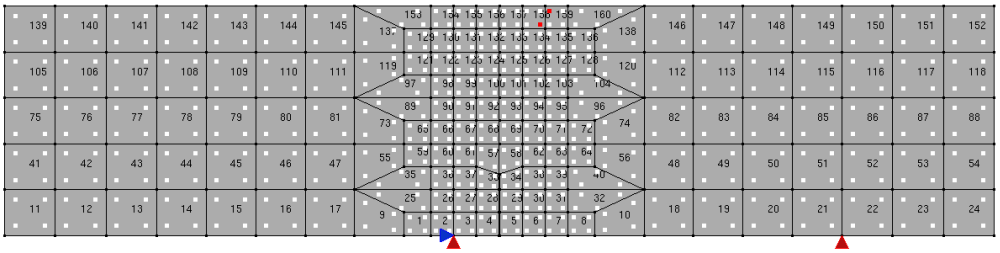
The first validation example shown will be the 4-point shear, since this model is the main proof that the MLP did not overfit the training data. This analysis will look in detail at the results for all the stress components of the system. However, for the remaining test examples, the focus will be on presenting the results of the stress components of interest in the problem, namely the one responsible for the failure mode. Nevertheless, the results of the other components will be presented in the Appendix C of this dissertation.

8.1 Shearing - 4-Point Shearing Test

In order to evaluate the network performance for mixed mode loads, a concrete beam with a size of $1322 \times 310 \times 156$ mm^3 was tested under 4-point shear stress under the plane stress state, simulated by Arrea (1981). Fig. 8.2 shows the geometric details of the test, the boundary conditions (loading and support), as well as the finite element model adopted. A reference load of $P = 130.0$ kN was considered. For the nonlinear analysis, the cylindrical arc length control method was used, with an initial load factor increment of 0.0125 and tolerance for convergence of 1×10^{-4}. For this case, it was used the shear retention factor $\beta_r = 0.02$. Fig. 8.3 and Fig. 8.4 presents the equilibrium paths of the given problem.



(a)



(b)

Figure 8.2: 4-point shearing test. (a) Problem setting from Arrea (1981) (measures in mm), (b) Mesh setting.

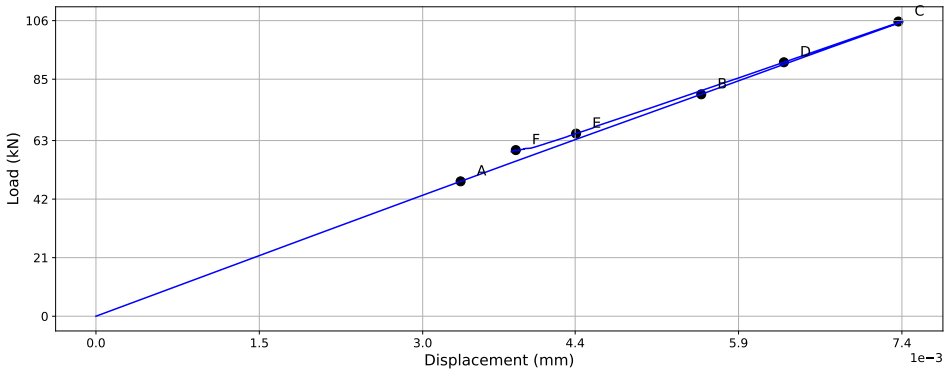


Figure 8.3: Load-displacement curve for the 4-point shearing test - Horizontal displacement of the right node of the crack tip.

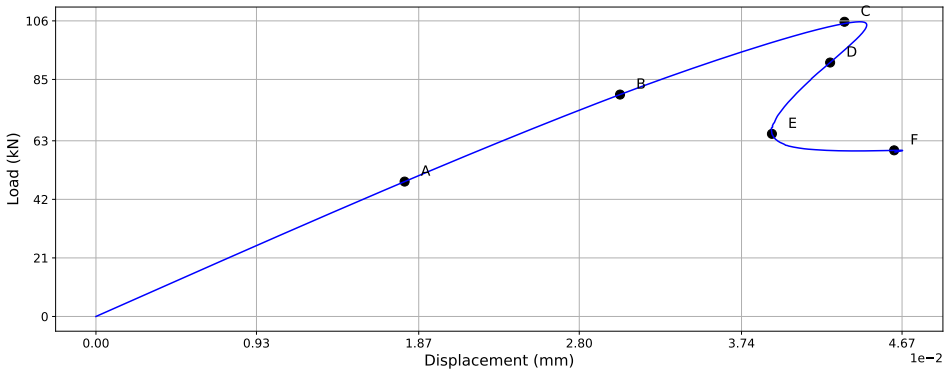


Figure 8.4: Load-displacement curve for the 4-point shearing test - Vertical displacement of the right node of the crack tip.

In this case, the positions of the reference straight lines were defined in the regions of higher shear stresses. The horizontal line was positioned in a central region between the crack tip and the upper face of the structure. The vertical line was setted between the first support and the crack. Then, the stresses obtained via FEM were compared with the stresses predicted by MLP along 6 steps of the equilibrium path, indicated in Fig. 8.3 and Fig. 8.4, so that results were obtained in the elastic and inelastic branches and for the maximum load factor (Fig. 8.5 to Fig. 8.10).

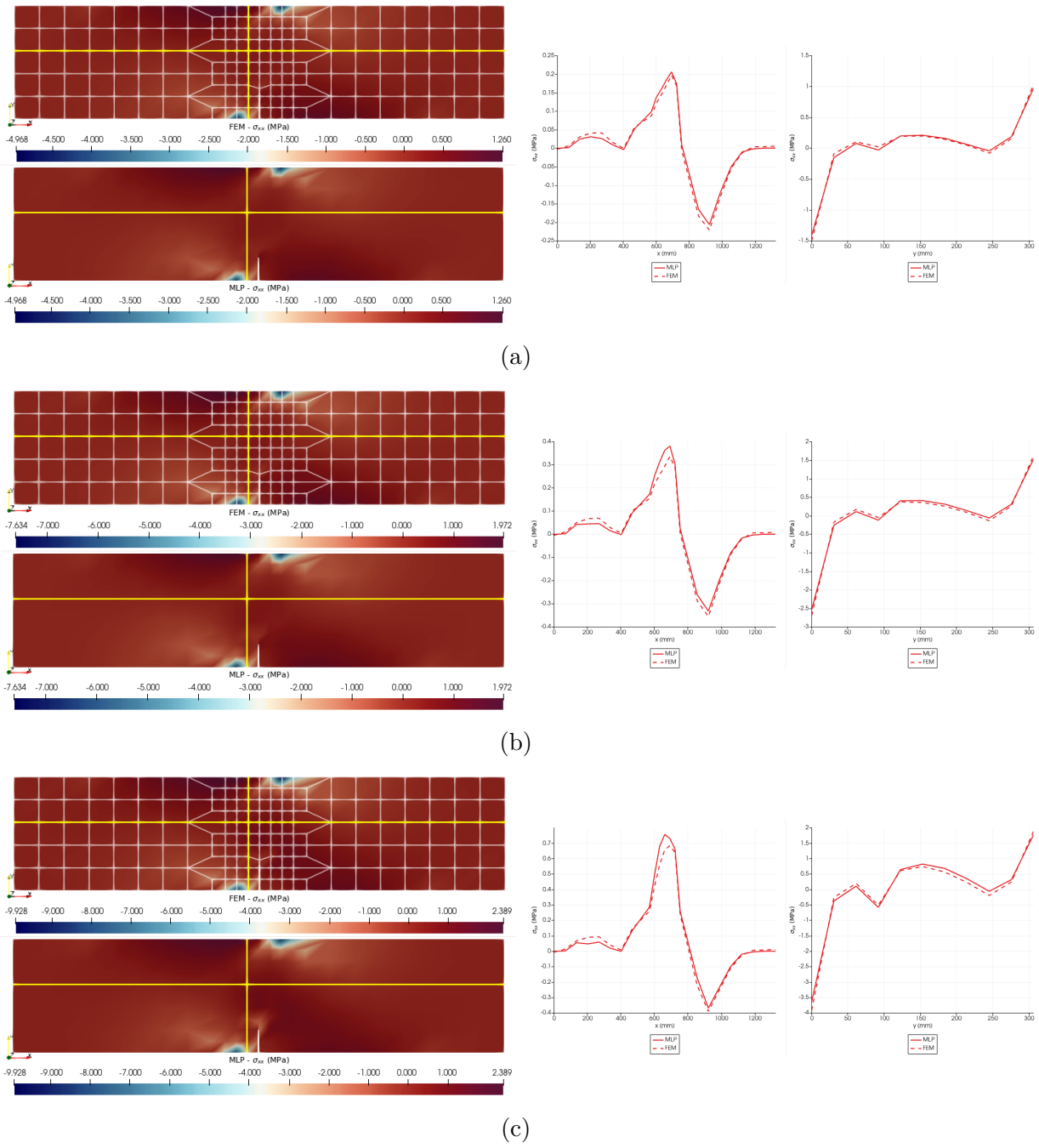
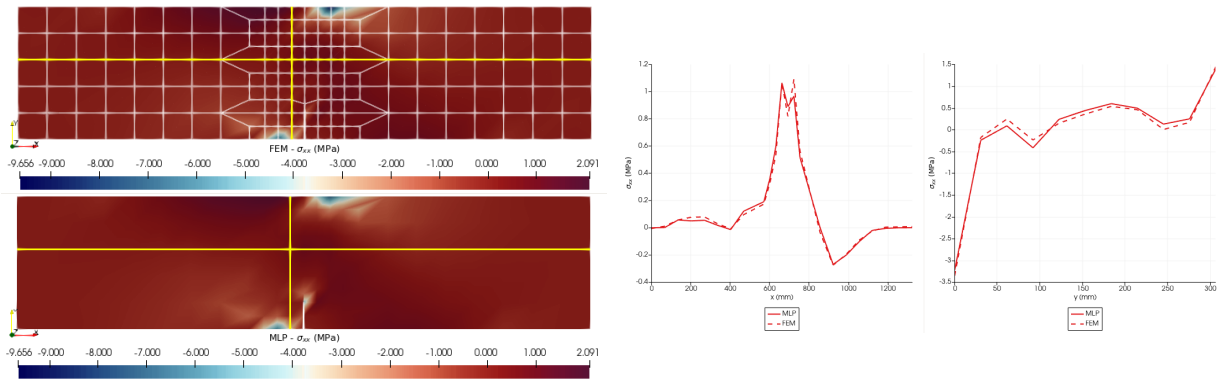
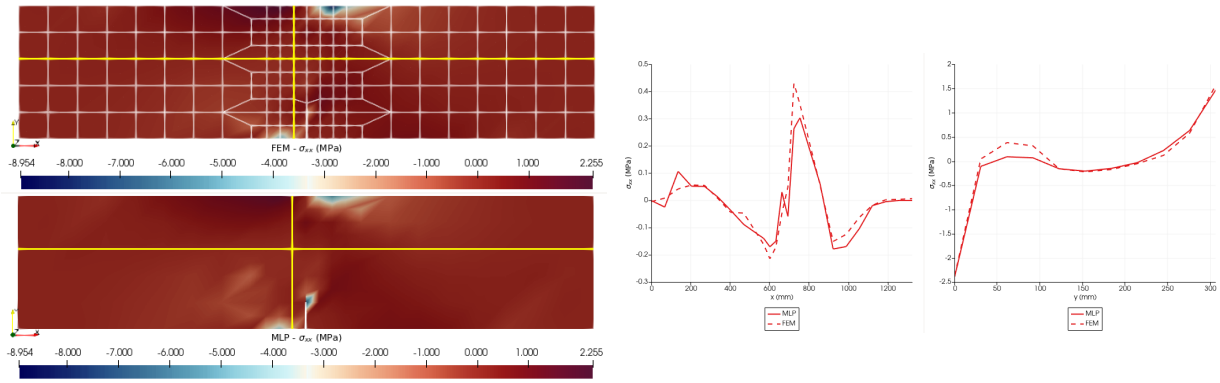


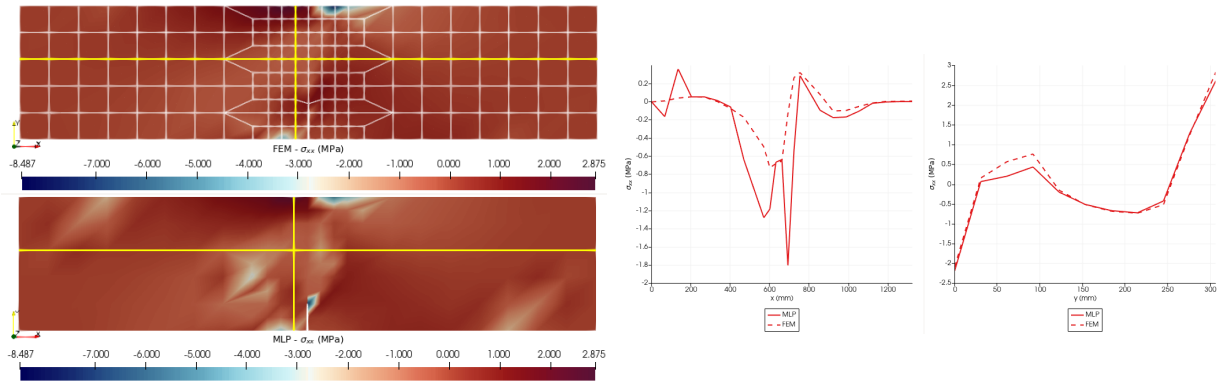
Figure 8.5: 4-point shearing test - Stresses FEM × MLP - σ_{xx}. (a) Point A Step 30, (b) Point B Step 50, (c) Point C Step 72.



(a)



(b)



(c)

Figure 8.6: 4-point shearing test - Stresses FEM × MLP - σ_{xx}. (a) Point D Step 100, (b) Point E Step 150, (c) Point F Step 218.

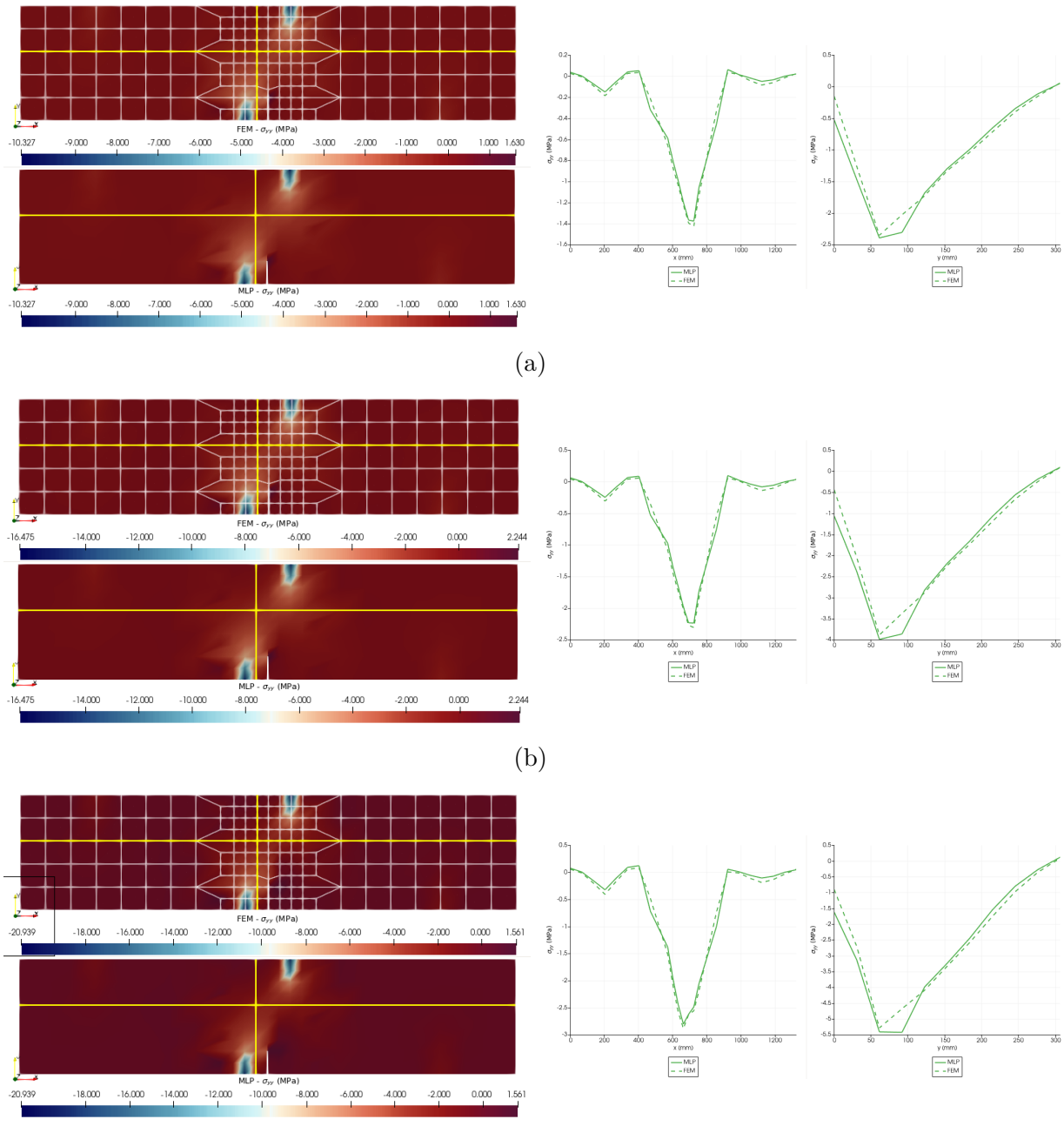


Figure 8.7: 4-point shearing test - Stresses FEM × MLP - σ_{yy}. (a) Point A Step 30, (b) Point B Step 50, (c) Point C Step 72.

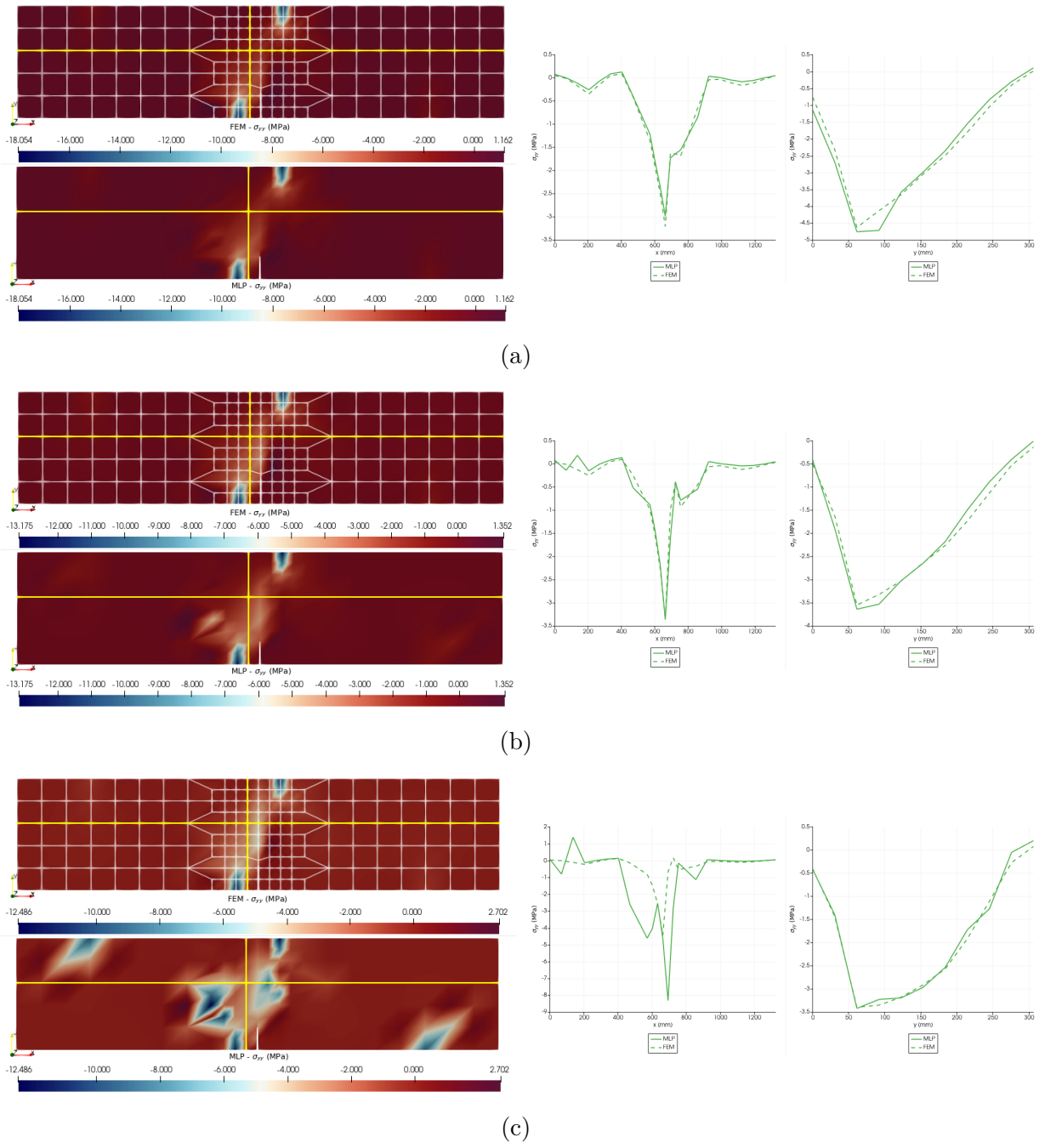


Figure 8.8: 4-point shearing test - Stresses FEM × MLP - σ_{yy}. (a) Point D Step 100, (b) Point E Step 150, (c) Point F Step 218.

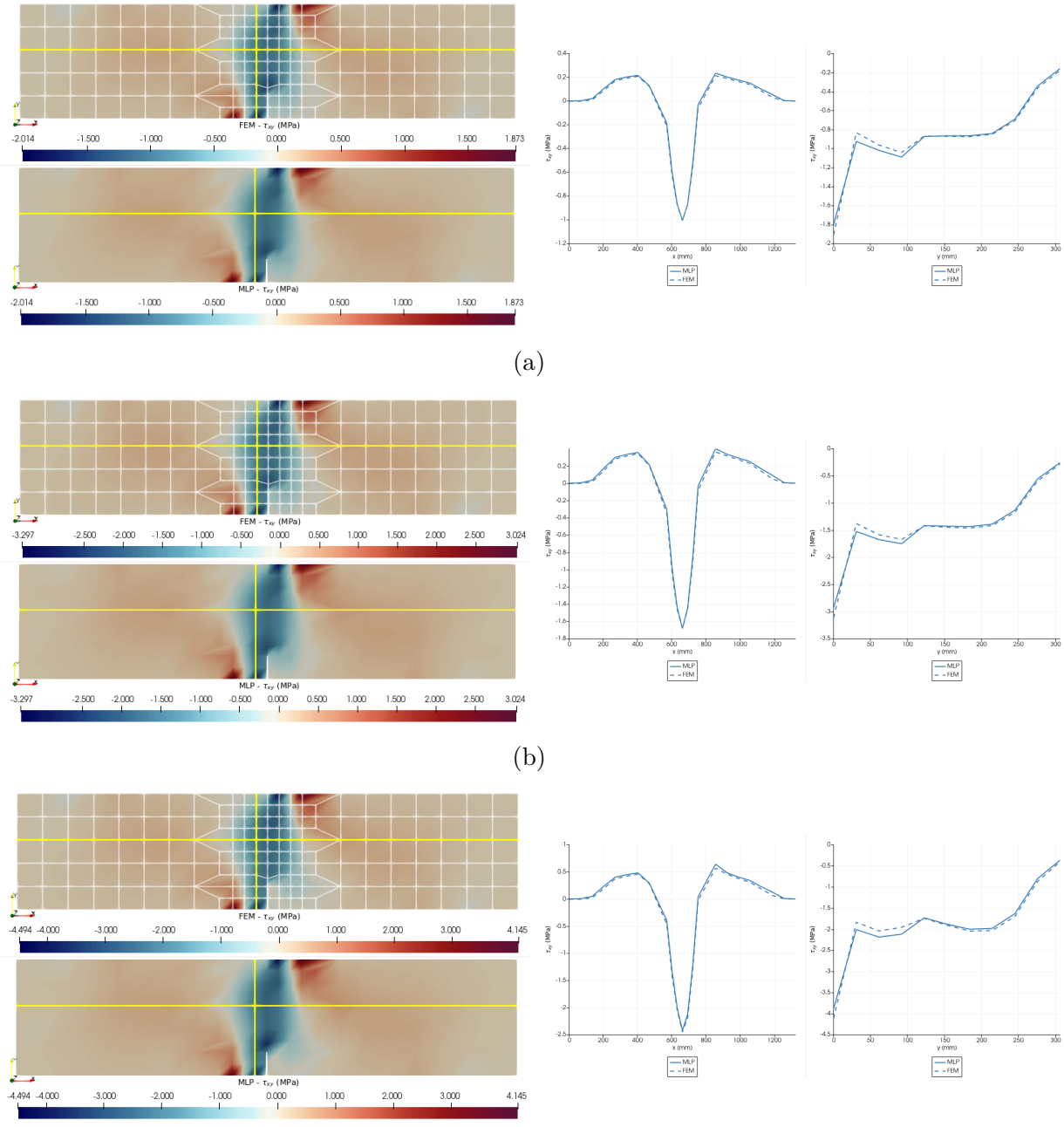


Figure 8.9: 4-point shearing test - Stresses FEM × MLP - τ_{xy}. (a) Point A Step 30, (b) Point B Step 50, (c) Point C Step 72.

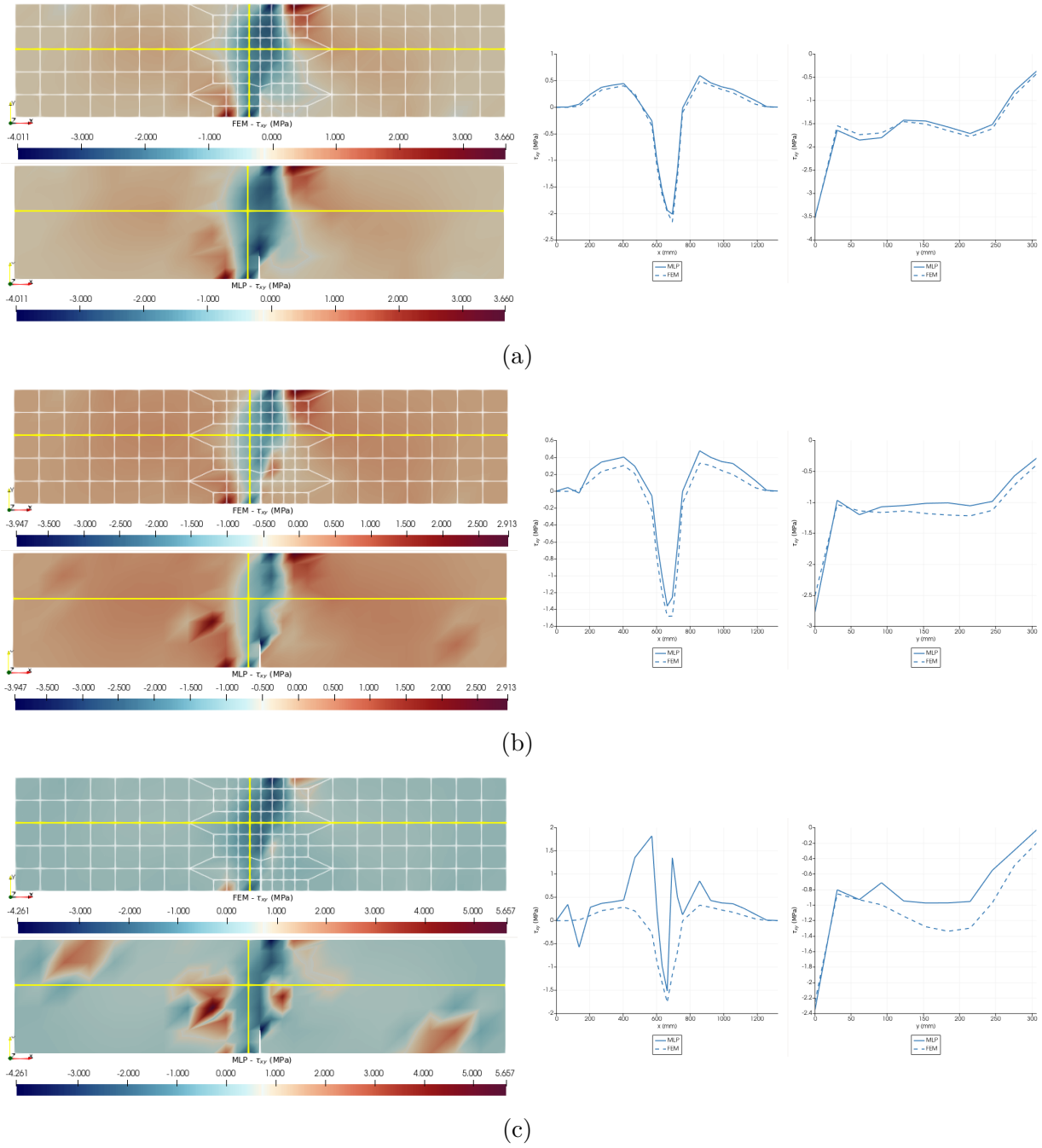


Figure 8.10: 4-point shearing test - Stresses FEM × MLP - τ_{xy}. (a) Point D Step 100, (b) Point E Step 150, (c) Point F Step 218.

Following are plots of the cumulative mean absolute error over the same steps analyzed in the previous plots (Fig. 8.11 to Fig. 8.13).

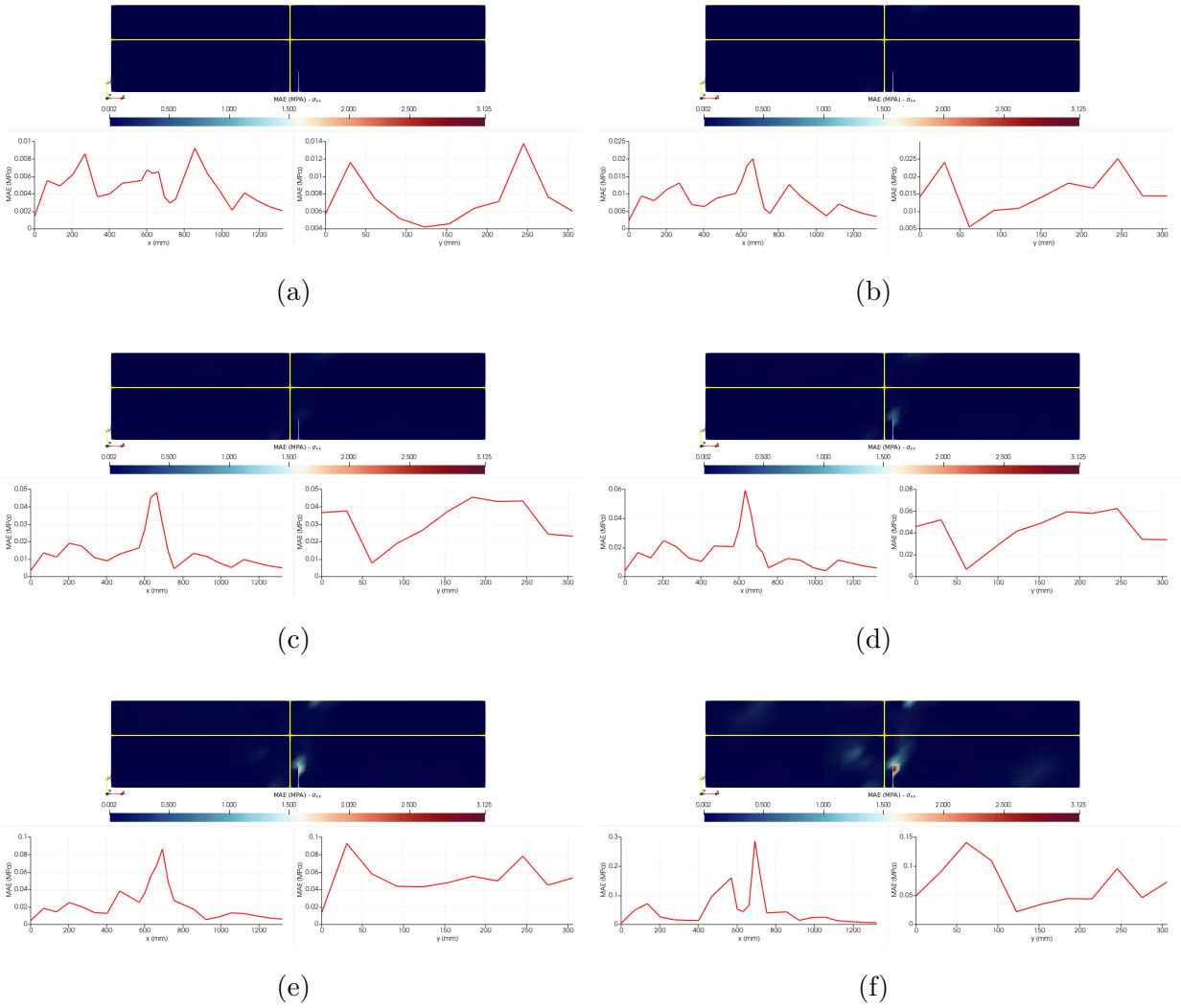


Figure 8.11: 4-point shearing test - MAE Accumulated - σ_{xx}. (a) Point A Step 30, (b) Point B Step 50, (c) Point C Step 72, (d) Point D Step 100, (e) Point E Step 150, (f) Point F Step 218.

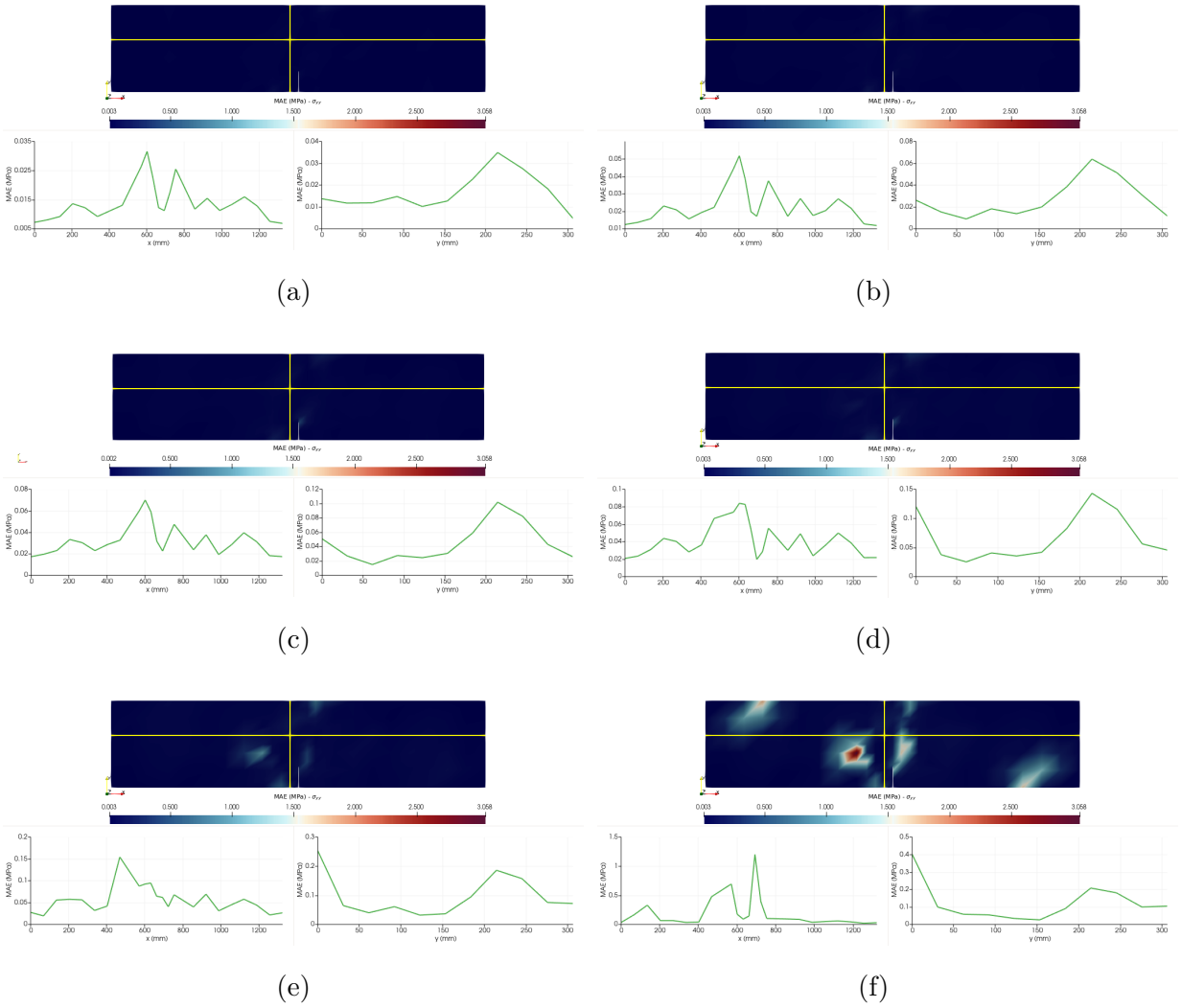


Figure 8.12: 4-point shearing test - MAE accumulated - σ_{yy}. (a) Point A Step 30, (b) Point B Step 50, (c) Point C Step 72, (d) Point D Step 100, (e) Point E Step 150, (f) Point F Step 218.

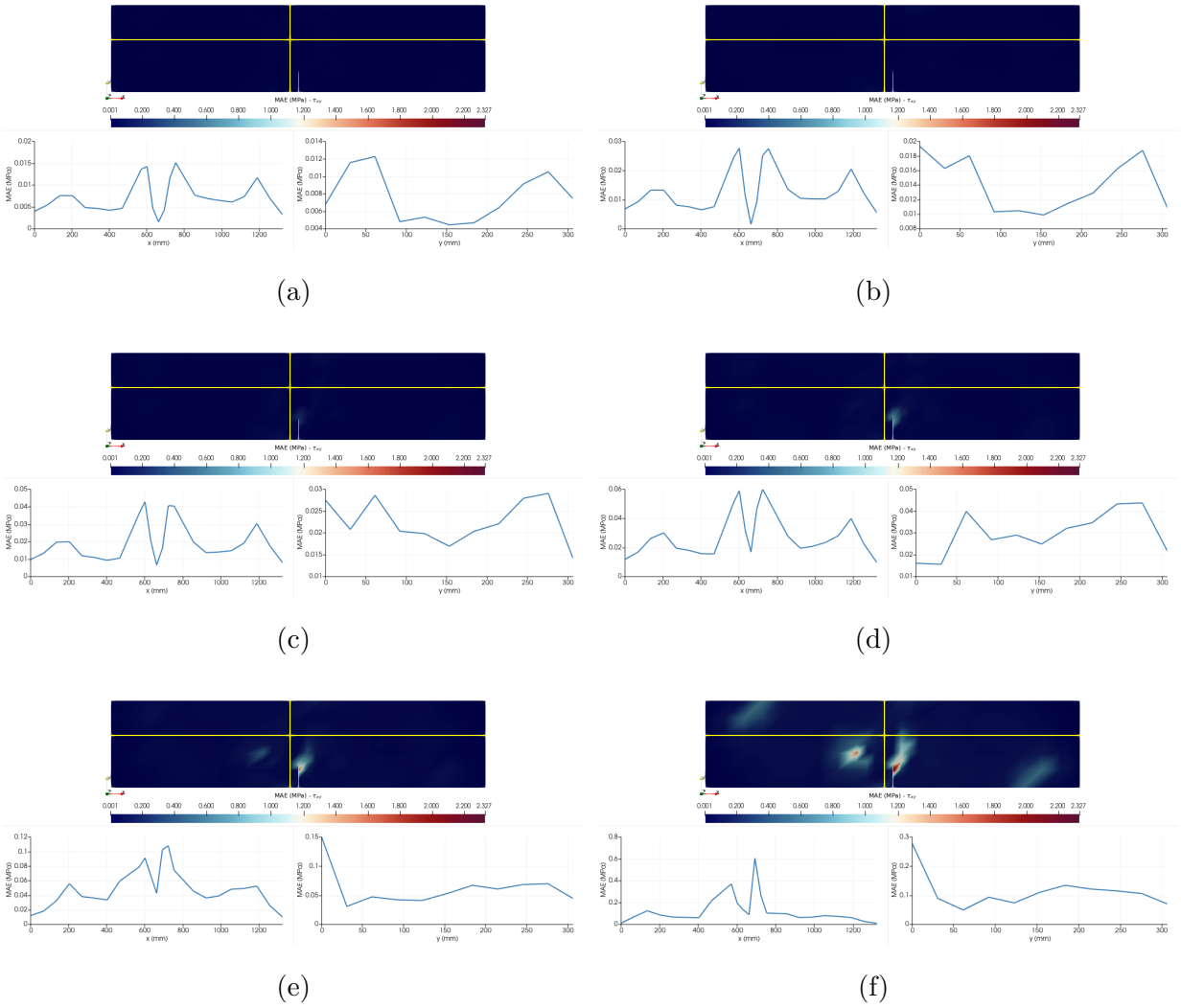
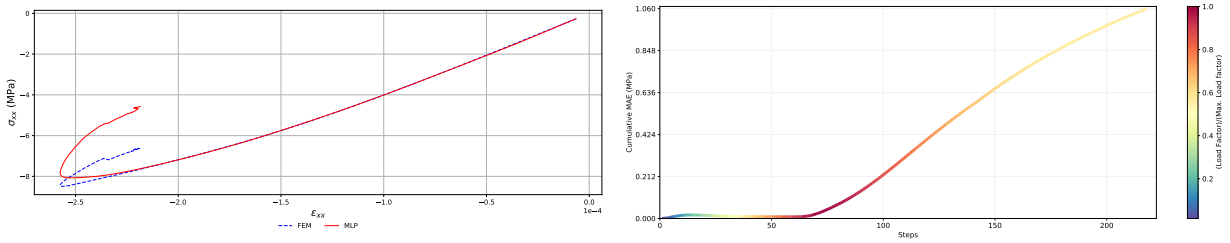
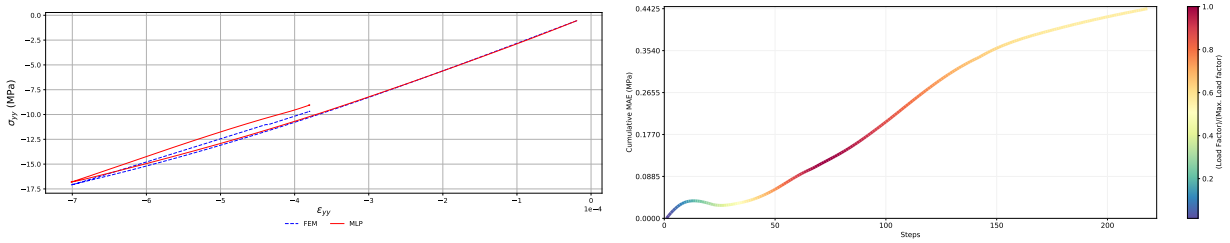


Figure 8.13: 4-point shearing test - MAE accumulated - τ_{xy}. (a) Point A Step 30, (b) Point B Step 50, (c) Point C Step 72, (d) Point D Step 100, (e) Point E Step 150, (f) Point F Step 218.

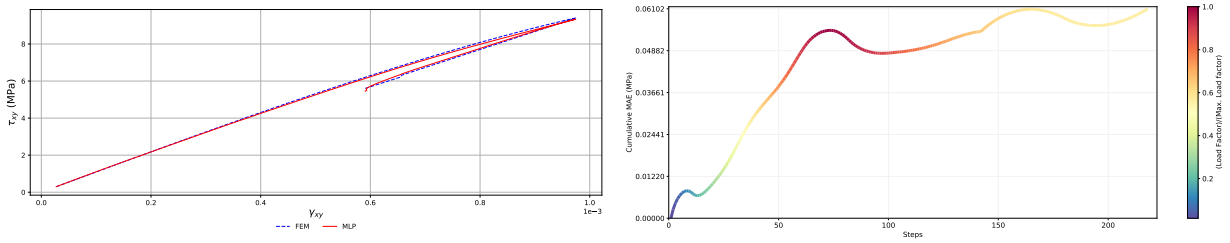
To observe the performance of MLP in predicting the stress histories, two points were selected within the region of interest of analysis, located near the point of application of the load P with coordinate $(722, 306)$. Fig. 8.14 presents the stress-strain curves for the point coincident with the Gauss integration point 3 of element 159 of the finite element mesh generator of the test data and Fig. 8.15 the curves for the point coincident with the Gauss integration point 2 of element 158, both positions indicated in Fig. 8.2b.



(a)

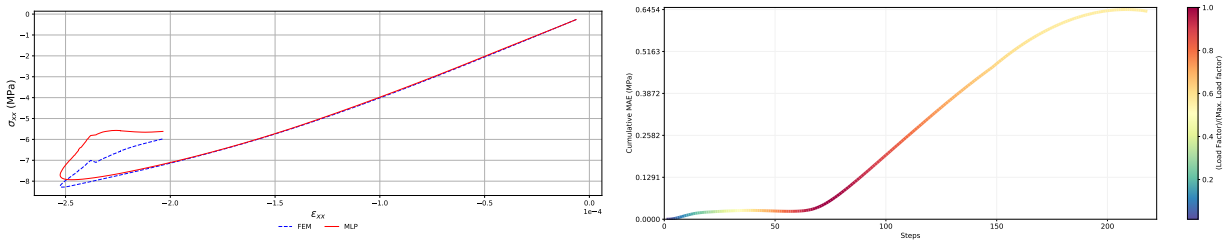


(b)

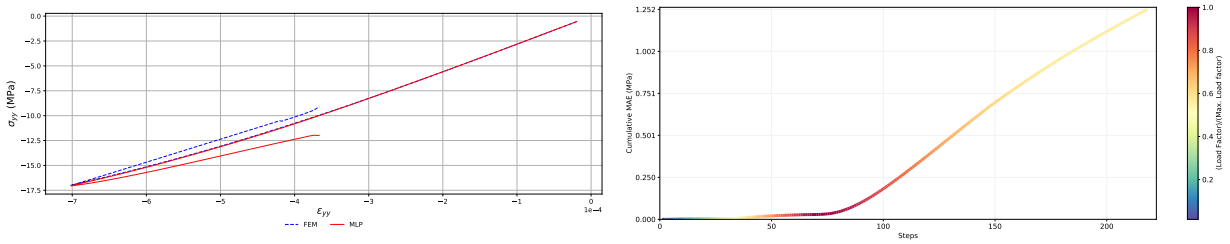


(c)

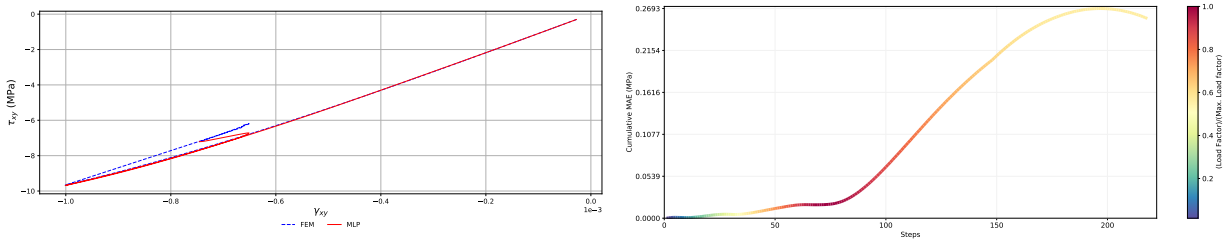
Figure 8.14: 4-point shearing test - Global stress-strain history of E159 - IP3 - Comparison between FEM and MLP results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)



(b)



(c)

Figure 8.15: 4-point shearing test - Global stress-strain history of E158 - IP1 - Comparison between FEM and MLP results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.

Overall, excellent results are observed for all stress components, throughout the structure and in all phases of the equilibrium path. The heat maps of the accumulated mean absolute error show that there is a small variation between the expected and predicted stresses, clear with the increase of the accumulated mean absolute error, but of low values in relation to the respective analysed stresses and only at the end of the equilibrium path, starting close to step 150. Fig. 8.14 and Fig. 8.15 attest that MLP accurately predicted the histories of normal and shear stresses for both analyzed points.

In summary, it was noted that the network was able to learn the behavior of the training data and generalize this knowledge to an unknown structure, where stress states were not presented in the training phase. The observed results are strong evidence that the network did not overfit the training data and that it was able to capture patterns in the behavior of the studied material.

8.2 Traction - Asymmetric Traction Test

The second set of test data was obtained from an asymmetric traction test, adapted from Wu et al. (2020). The problem and mesh are described in Fig. 8.16, in which a plate with size of $1 \times 1 \times 1$ mm^3 was subjected to a constant displacement Δ on the upper face in a plane stress state. In the analysis, the direct displacement control method was considered with a vertical displacement increment of 5×10^{-6} on the upper central node of the plate, with a tolerance in displacements of 1×10^{-4}. Also, a shear retention factor $\beta_r = 0.0$ was used.

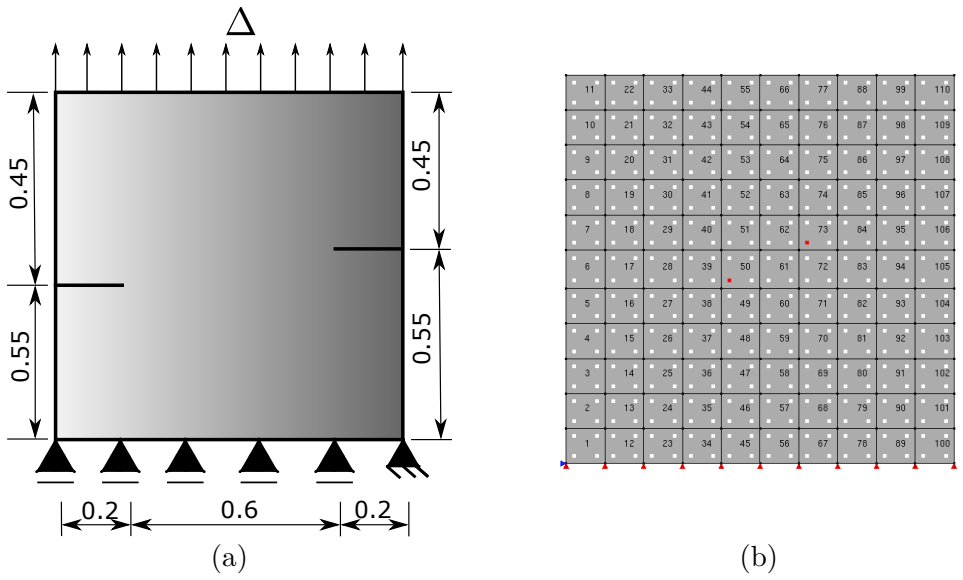


Figure 8.16: Asymmetric traction test. (a) Problem setting by Wu et al. (2020) (measures in mm), (b) Q4 mesh.

The equilibrium path of the node control point is presented in Fig. 8.17.

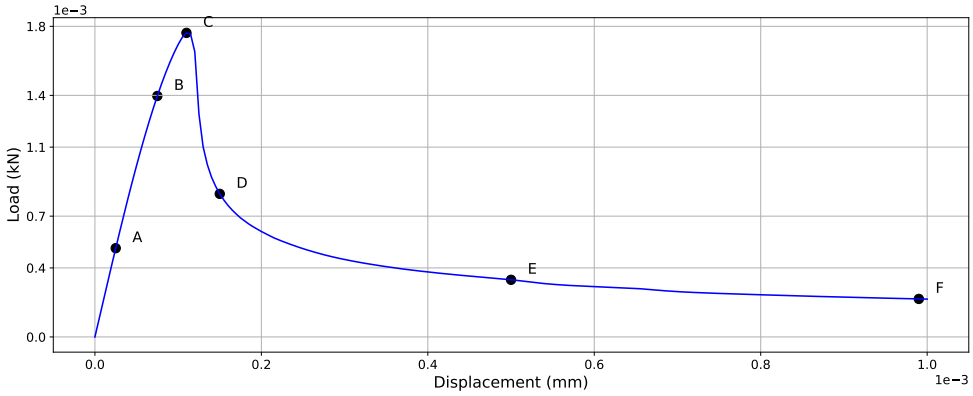
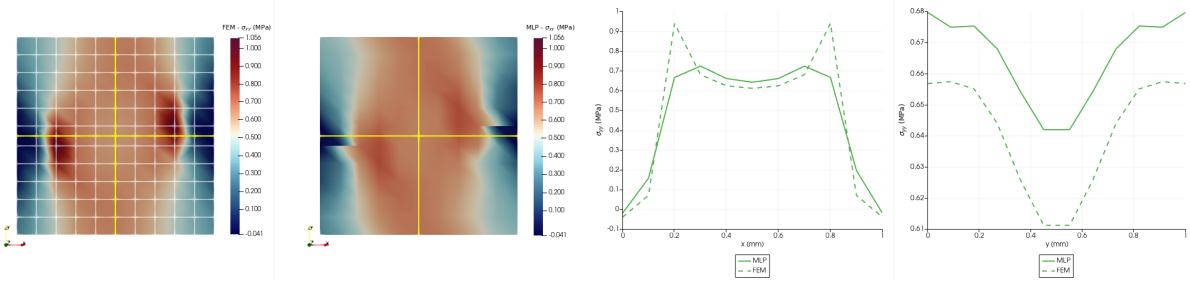
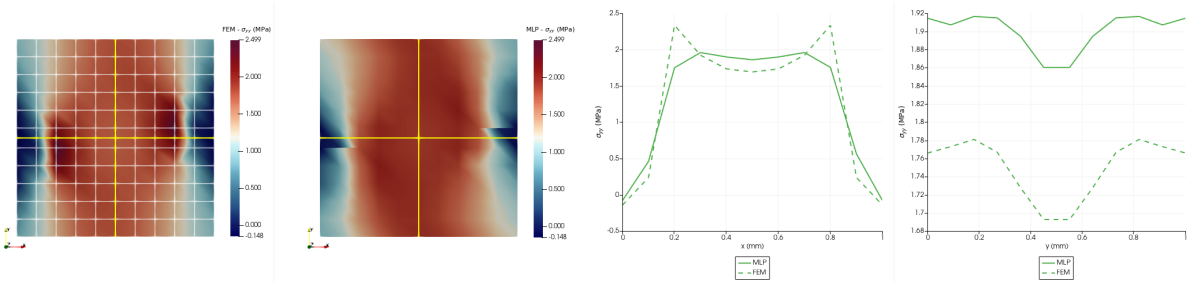


Figure 8.17: Load-displacement curve for the asymmetric traction test - Vertical displacement of the controlled node.

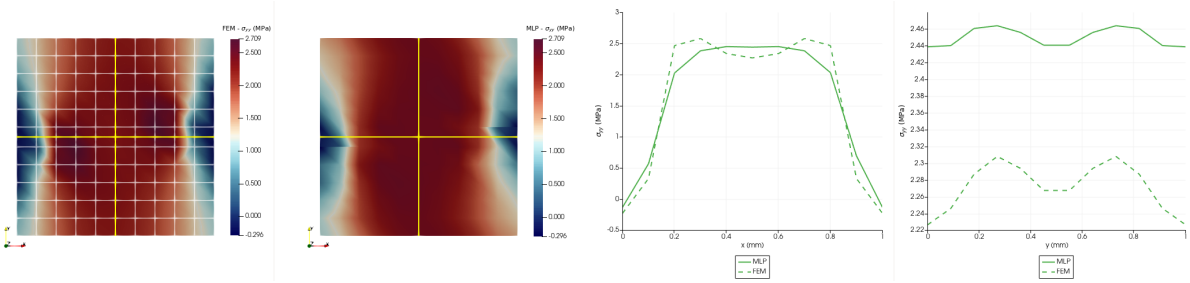
The failure in this type of test occurs due to normal tensile stresses in the load application direction (y-direction), in the central regions, close to the existing cracks. In this structural system, the damage occurs in the central region through the encounter between the initial cracks (see (Leão et al., 2021)). Therefore, the horizontal and vertical analysis straight lines were placed in the central region of the plate. Then, the normal stresses in the y-direction calculated by FEM were compared with those predicted by MLP, along 6 different moments of the equilibrium path of the numerical test (indicated in Fig. 8.17), as illustrated by Fig. 8.18 and Fig. 8.19.



(a)



(b)



(c)

Figure 8.18: Asymmetric traction test - Stresses FEM × MLP - σ_{yy}. (a) Point A Step 5, (b) Point B Step 15, (c) Point C Step 22.

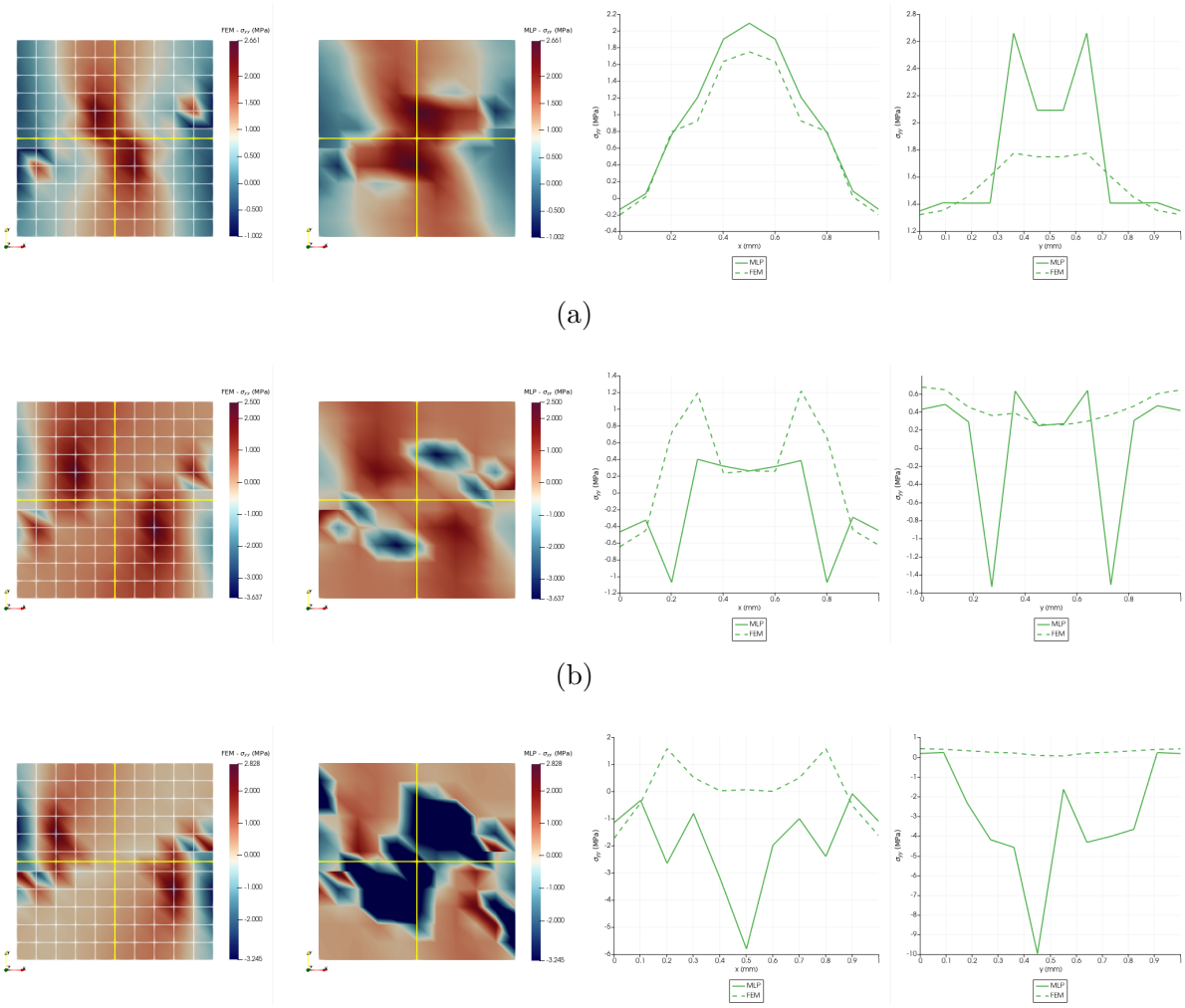


Figure 8.19: Asymmetric traction test - Stresses FEM × MLP - σ_{yy}. (a) Point D Step 30, (b) Point E Step 100, (c) Point F Step 198.

In the sequence, the evolution of the accumulated MAE along the steps previously analyzed was evaluated.

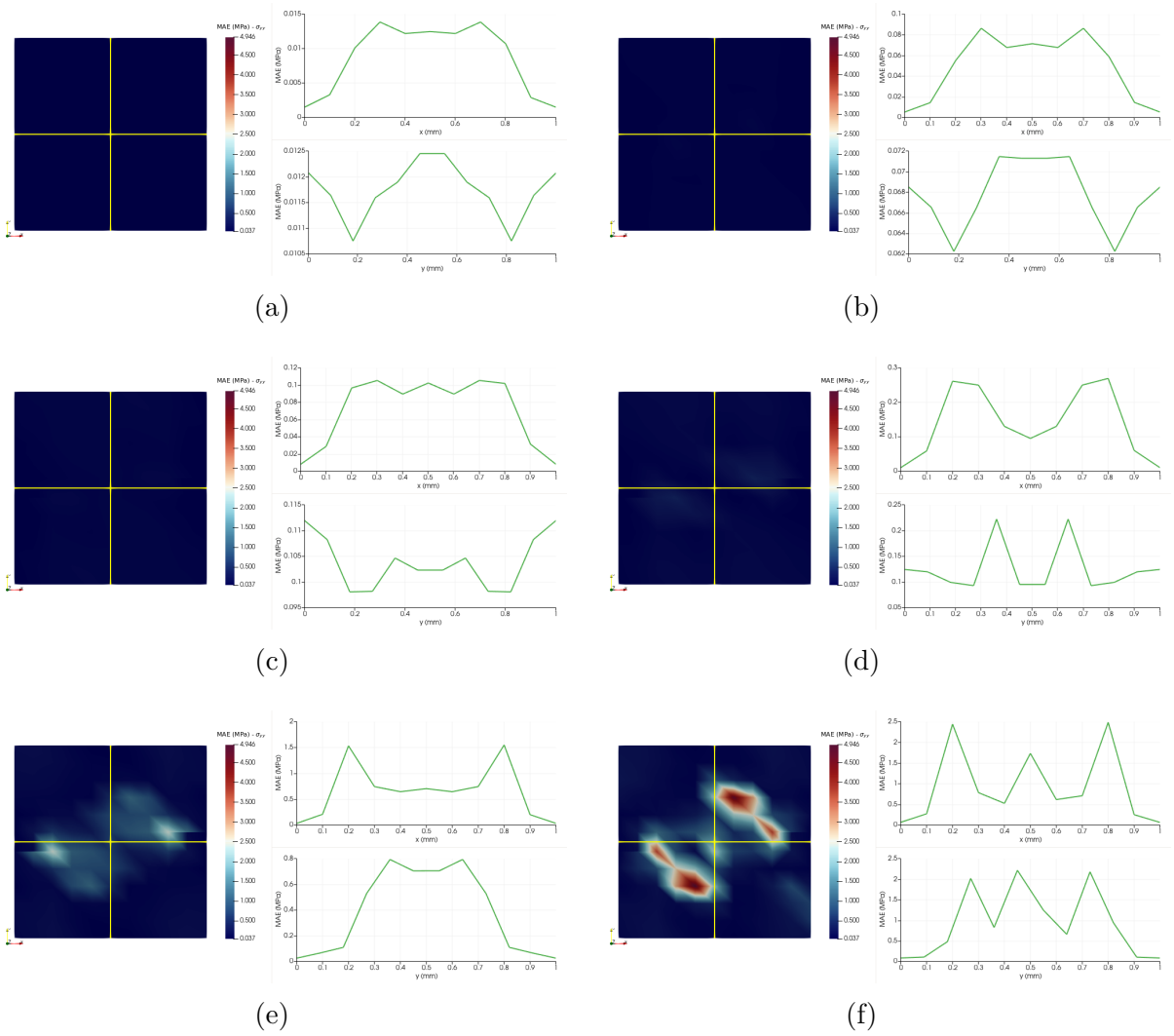
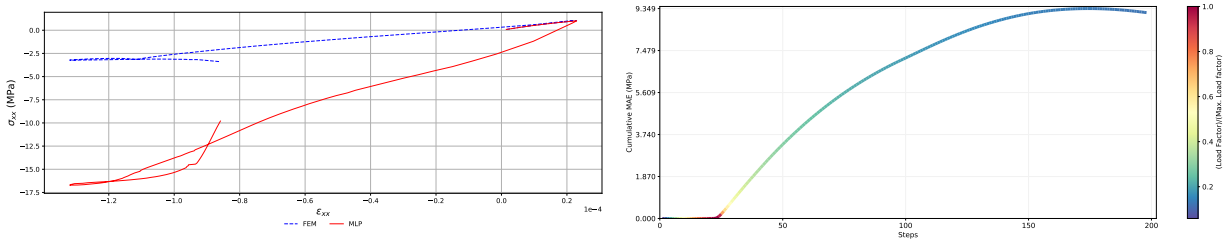
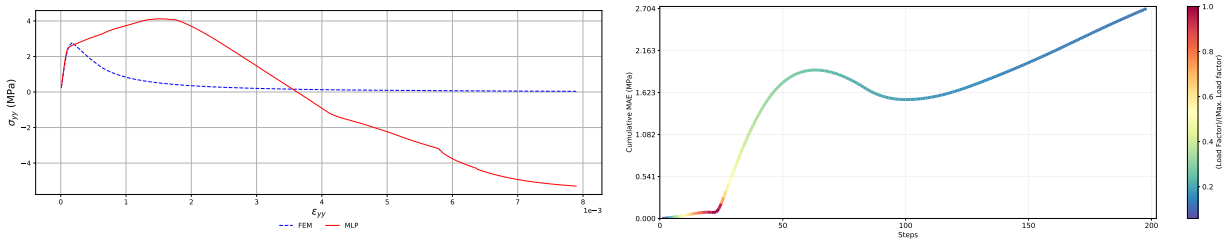


Figure 8.20: Asymmetric traction test - MAE Accumulated - σ_{yy}. (a) Point A Step 5, (b) Point B Step 15, (c) Point C Step 22, (d) Point D Step 30, (e) Point E Step 100, (f) Point F Step 198.

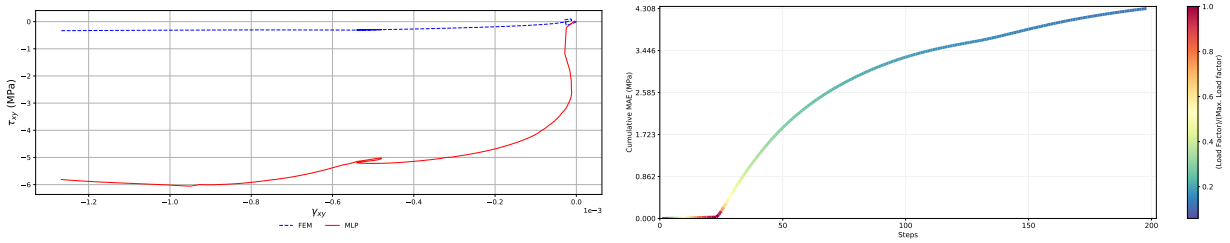
In this case, it was observed that the accumulated MAE concentrated in two regions of the plate, close to the crack, and that it increased from the maximum load step. With this, the stress-strain curves at two different points in the beam were investigated, as indicated in Fig. 8.16b. First, the behavior of a point at which the tensile stresses reach the material limit stress in the y-direction, a point coincident with the Gauss integration point 2 of element 73 (Fig. 8.21) was evaluated. This point also belongs to the region with the highest accumulated MAE throughout the analysis. To better verify the behavior of the point, the same stress graphs were also plotted until step 22, referring to the maximum load applied to the structure (Fig. 8.22). Next, it was evaluated, also in the y-direction, a central point of the plate in which tension was predominant but unloading occurred. This analysis point coincides with the Gauss integration point 2 of element 50 (Fig. 8.23).



(a)

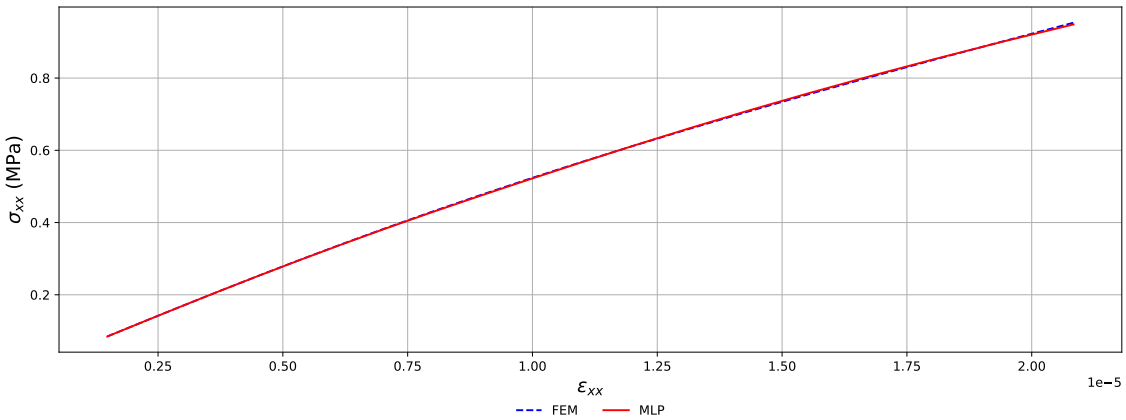


(b)

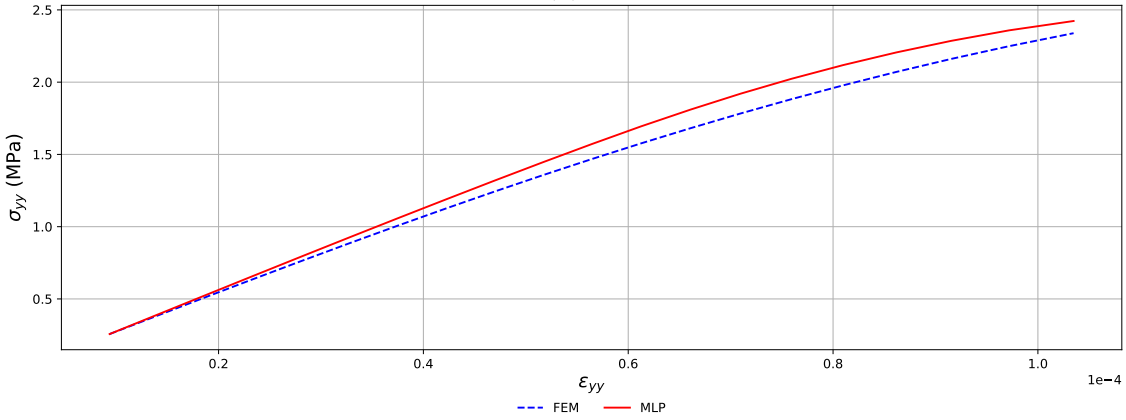


(c)

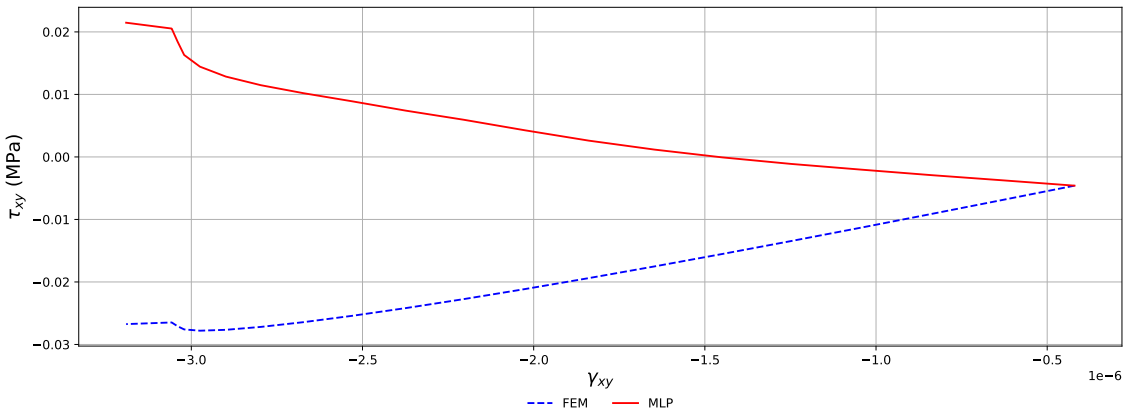
Figure 8.21: Asymmetric traction test - Global stress-strain history of E73 - IP2 - Comparison between FEM and MLP results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)

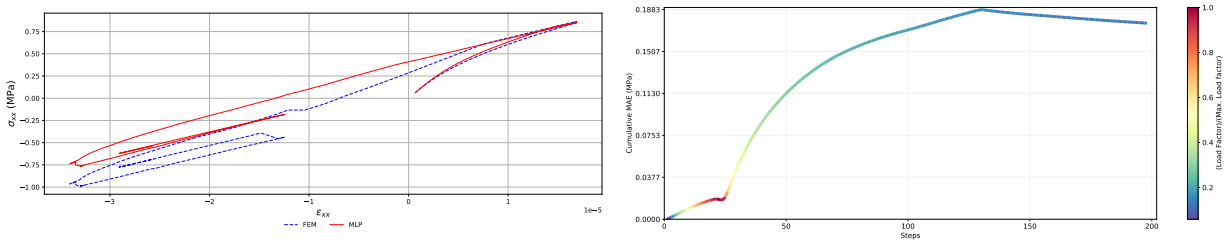


(b)

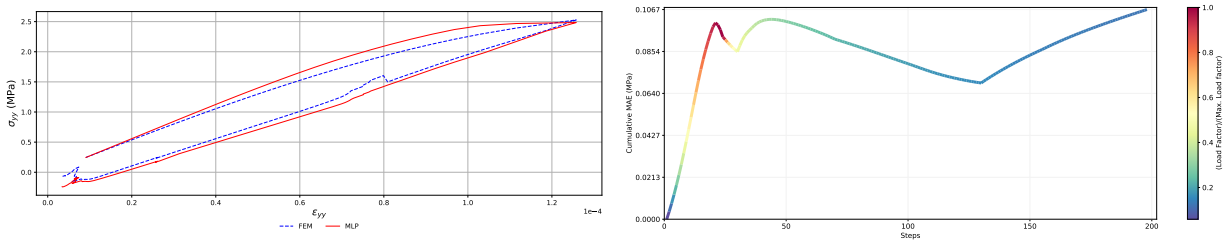


(c)

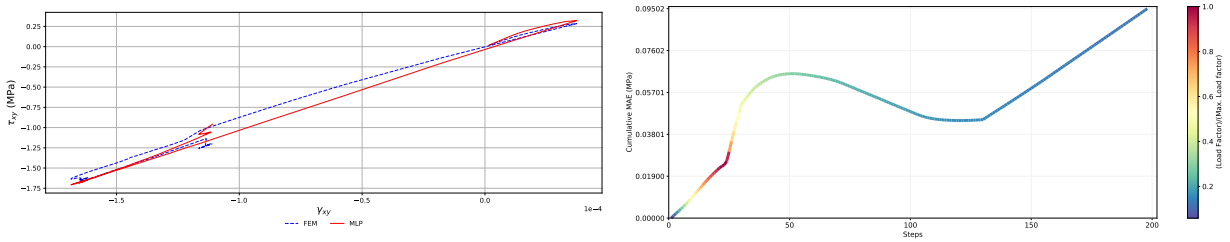
Figure 8.22: Asymmetric traction test - Global stress-strain history of E73 - IP2 until step 22 (Maximum load) - Comparison between FEM and MLP results. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)



(b)



(c)

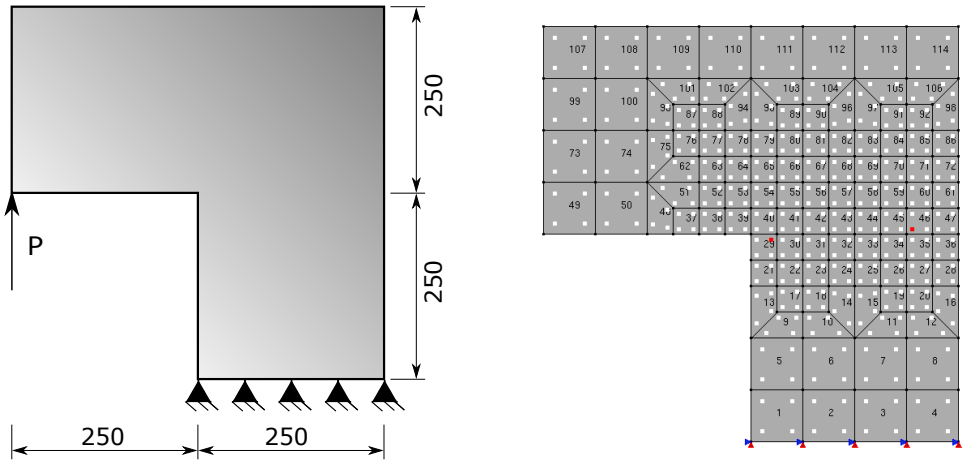
Figure 8.23: Asymmetric traction test - Global stress-strain history of E50 - IP2 - Comparison between FEM and MLP results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.

By analyzing the graphs above, it is possible to observe that for point 2, the MLP was able to predict satisfactorily the behavior of the material. For point 1, the MLP showed good predictive ability until the point of maximum load, not being able, therefore, to predict the behavior of stresses in y in the softening branch of the stress curve. This is a point of attention, since it may be an indication that the training database does not present sufficient information of stress states in which rupture occurs predominantly by tension in the y-direction.

8.3 Bending - L-Shaped Panel

Next, the numerical simulation of the L-shaped panel is presented. This structure was originally tested experimentally by Winkler et al. (2004). The concrete properties, used

in all previous simulations, were extracted from this test, as reported at the beginning of the Chapter 5. The L-shaped panel with thickness of 100 mm and subjected to a vertical force P = 7.0 kN it was simulated under plane stress state, with the boundary conditions and finite element mesh as illustrated in Fig. 8.24. A displacement increment of 0.005 was adopted for the direct displacement control method in the vertical direction at the load application point. A convergence tolerance of 1×10^{-4} and a zero shear retention factor ($\beta_r = 0.0$) were considered.



(a) (b)

Figure 8.24: L-shaped panel. (a) Problem setting Winkler et al. (2004) (measures in mm), (b) Mesh setting.

Fig. 8.25 presents the equilibrium path of the control node and the experimental patch from the test performed by Winkler et al. (2004). It can be seen that the constitutive model used in the numerical simulation presents a good representativeness about the real behavior of the material in the structure.

In this model, the failure mode is by bending with high normal stresses in the x and y directions. The damage starts concentrated in the angular joint of the panel and propagates horizontally throughout most of the concrete element, as can be observed in the works of Penna (2011) and Leão et al. (2021). The horizontal and vertical lines of analysis were positioned in the central region of the L-shaped panel, where it is known that the damage tends to propagate. Then, the normal stresses in the x and y directions calculated via FEM were compared with those predicted by MLP, along 6 different moments of the equilibrium path of the numerical test, as illustrated by Fig. 8.26 to Fig. 8.29.

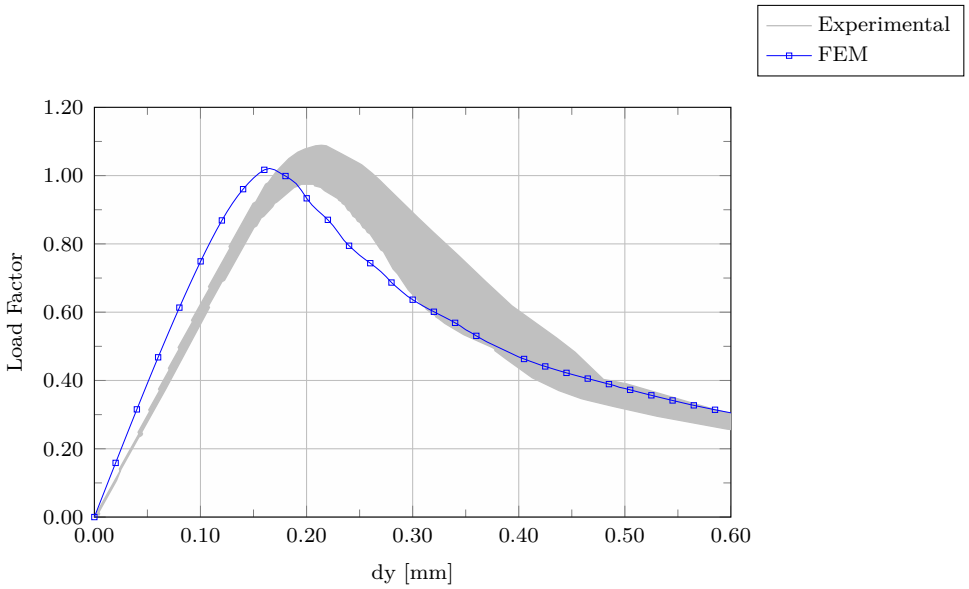
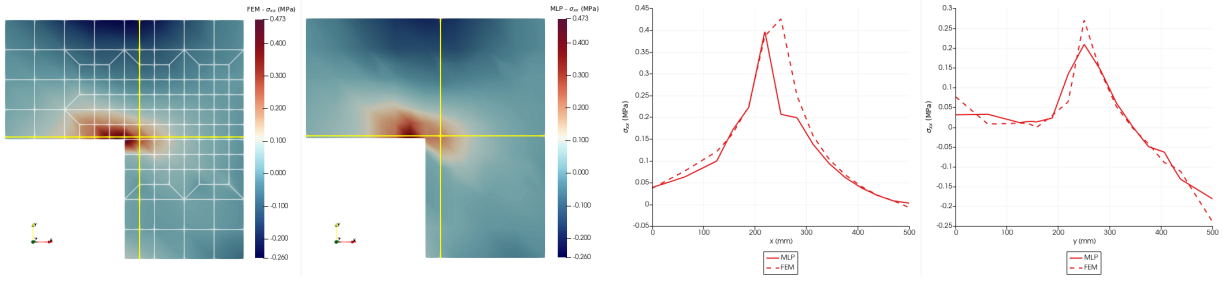
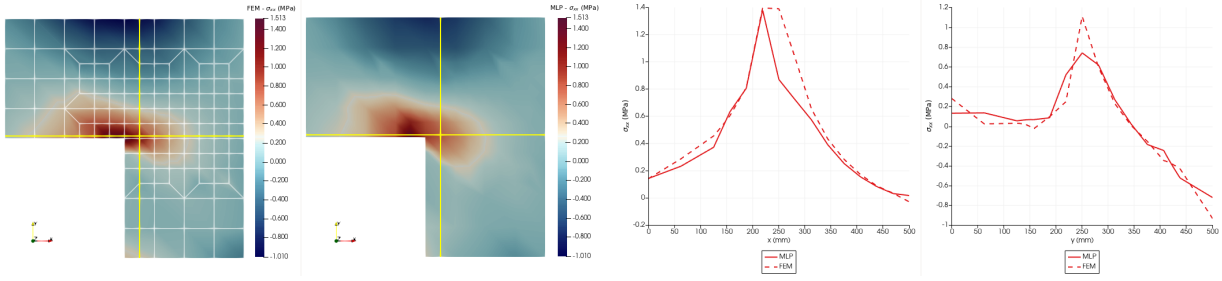


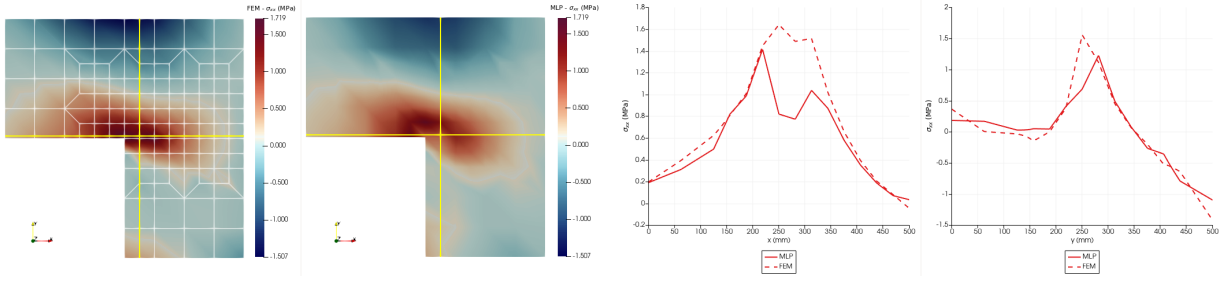
Figure 8.25: Load-displacement curve for the L-shaped panel - Vertical displacement of the load application point and the experimental patch by Winkler et al. (2004).



(a)

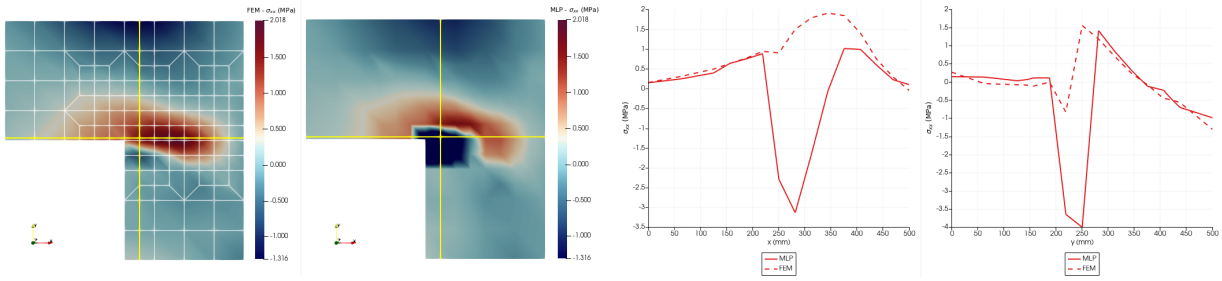


(b)

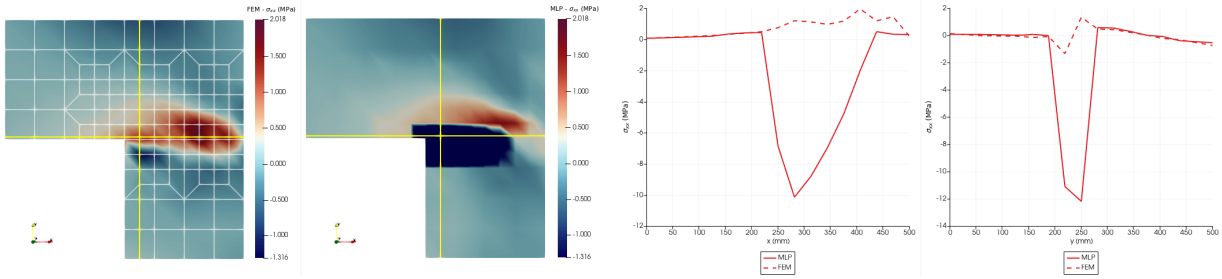


(c)

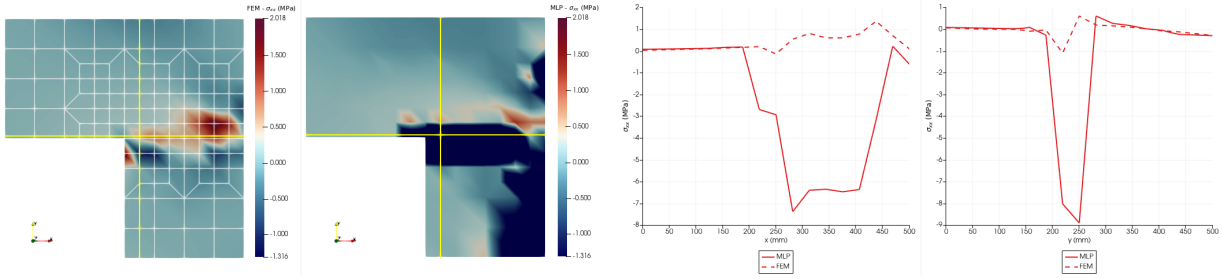
Figure 8.26: L-shaped panel - Stresses FEM × MLP - σ_{xx}. (a) Point A Step 5, (b) Point B Step 20, (c) Point C Step 33.



(a)



(b)



(c)

Figure 8.27: L-shaped panel - Stresses FEM × MLP - σ_{xx}. (a) Point D Step 50, (b) Point E Step 100, (c) Point F Step 250.

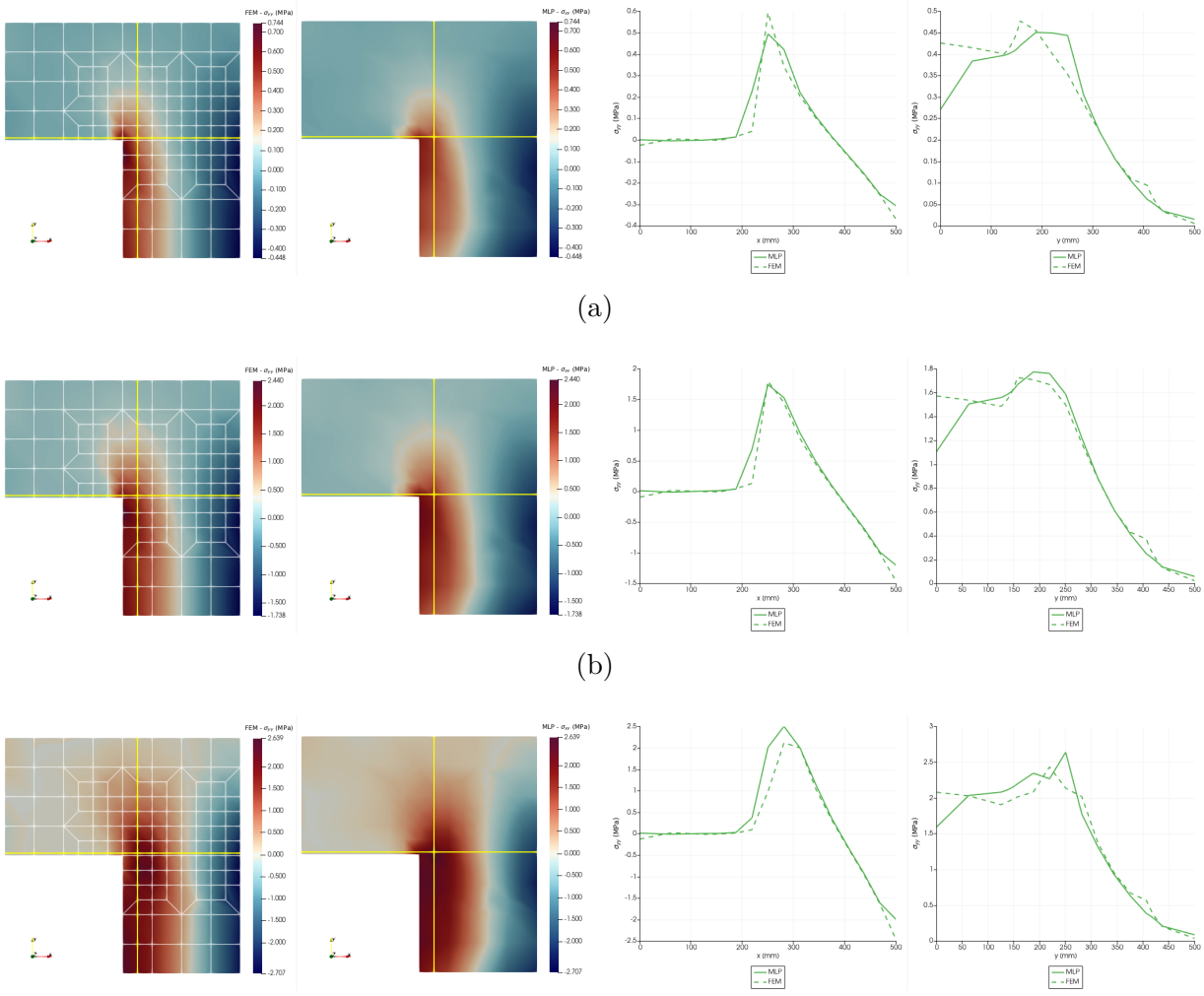


Figure 8.28: L-shaped Panel - Stresses FEM × MLP - σ_{yy}. (a) Point A Step 5, (b) Point B Step 20, (c) Point C Step 33.

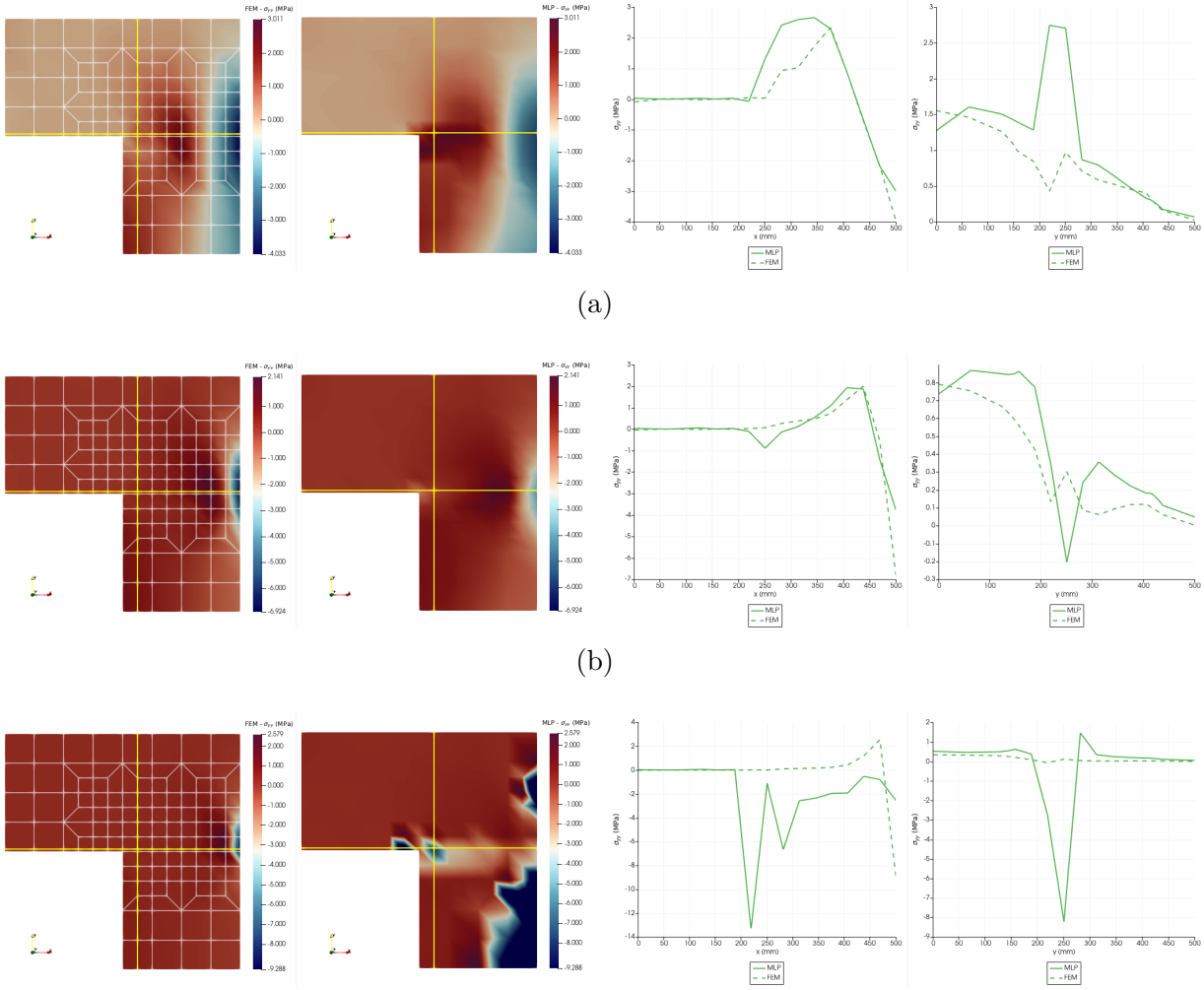


Figure 8.29: L-shaped panel - Stresses FEM × MLP - σ_{yy}. (a) Point D Step 50, (b) Point E Step 100, (c) Point F Step 250.

Then, once again, the behavior of the chosen metric along the equilibrium path of the problem was evaluated.

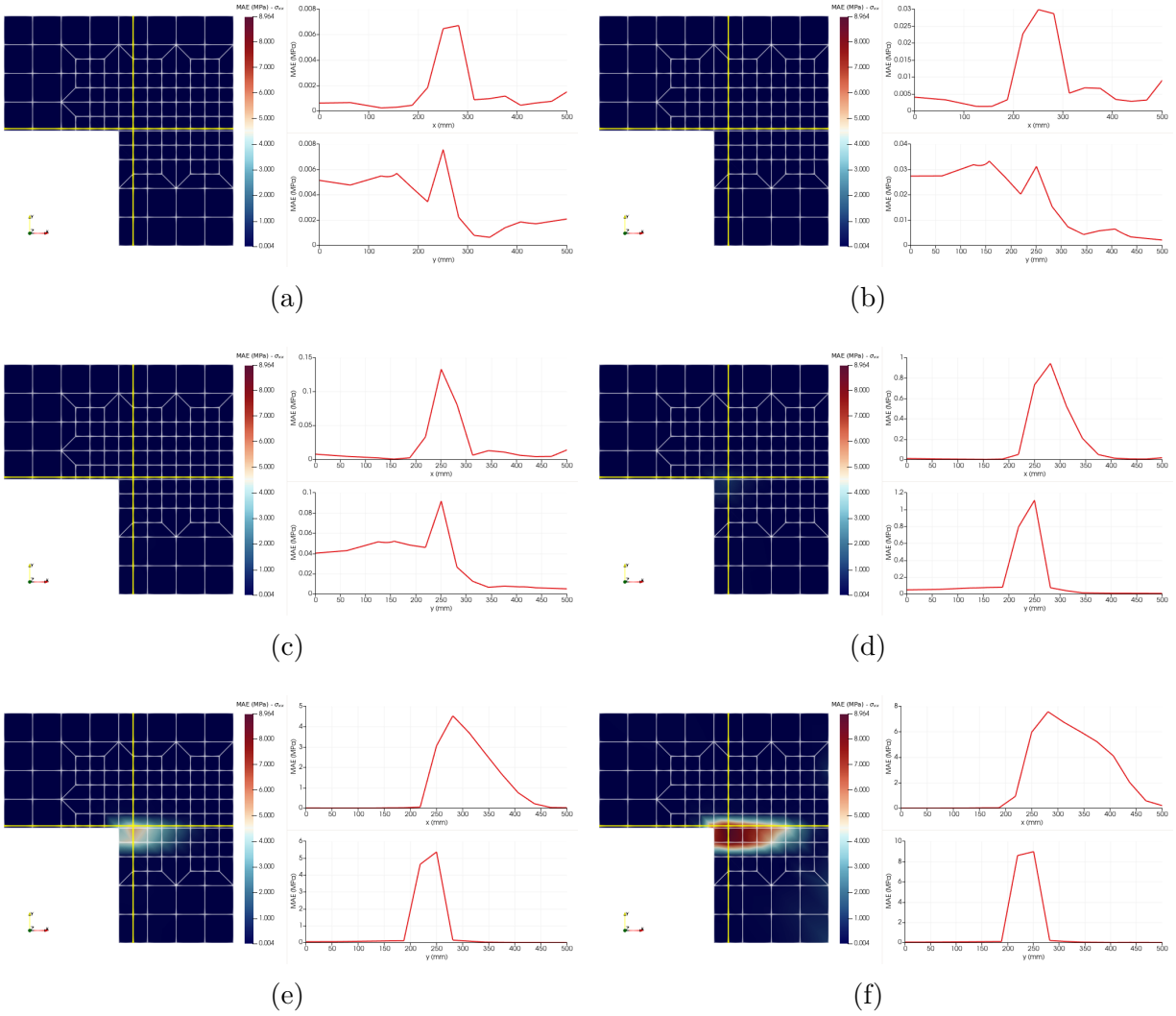


Figure 8.30: L-shaped panel - MAE Accumulated - σ_{xx}. (a) Point A Step 5, (b) Point B Step 20, (c) Point C Step 33, (d) Point D Step 50, (e) Point E Step 100, (f) Point F Step 250.

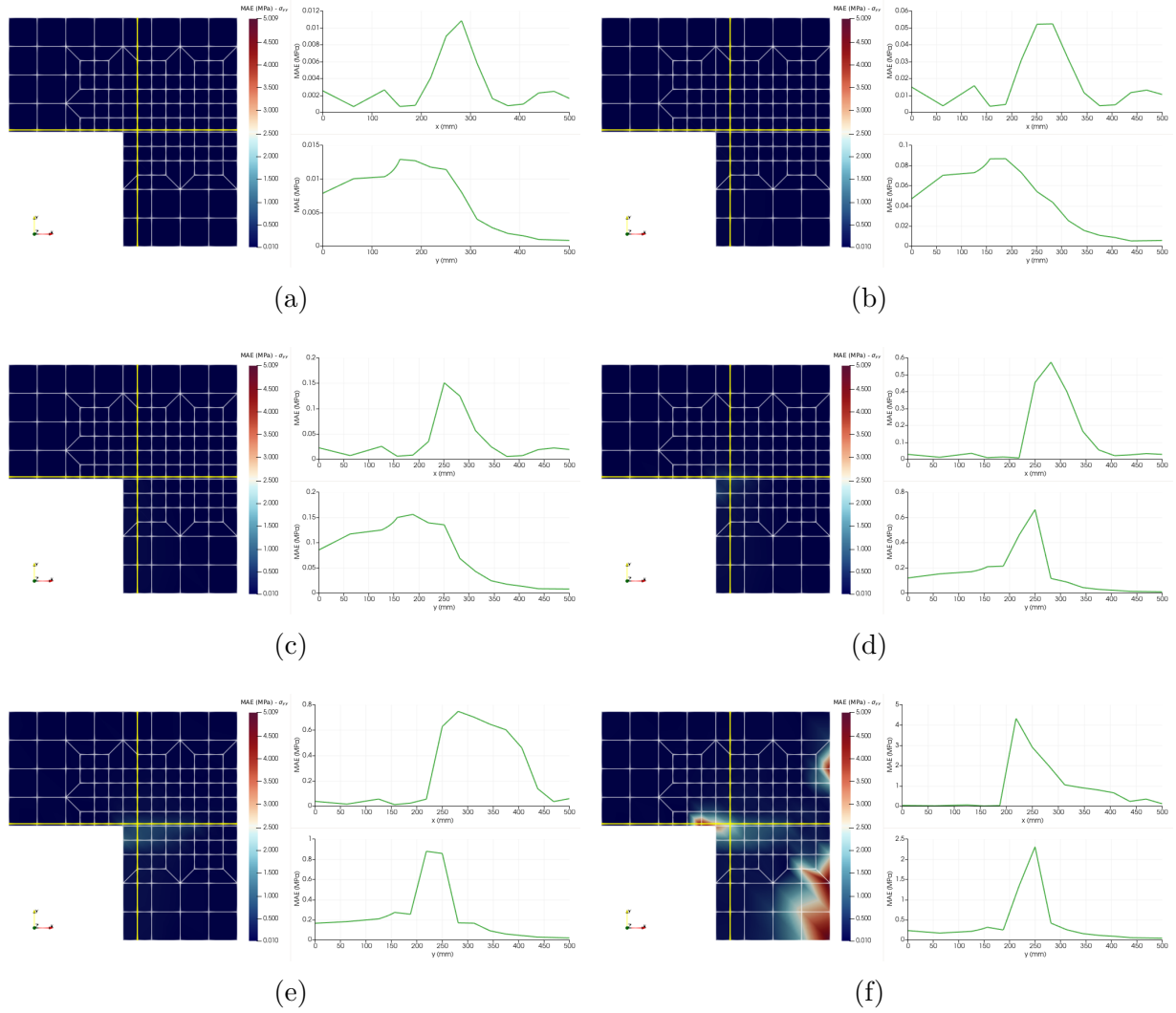
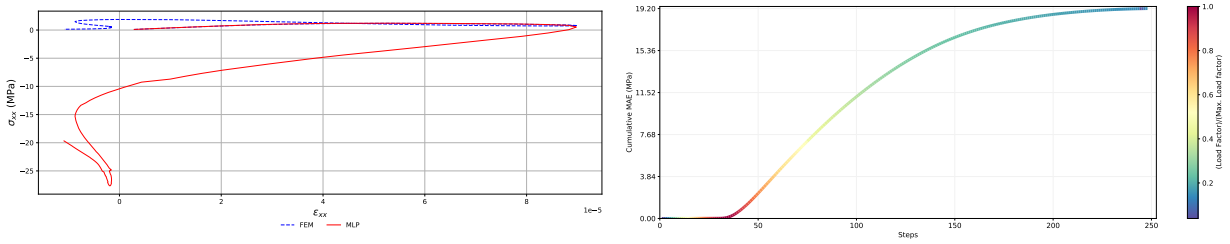
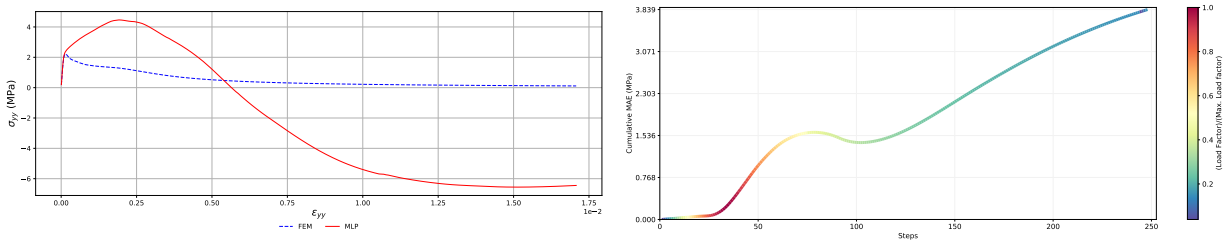


Figure 8.31: L-shaped panel - MAE Accumulated - σ_{yy}. (a) Point A Step 5, (b) Point B Step 20, (c) Point C Step 33, (d) Point D Step 50, (e) Point E Step 100, (f) Point F Step 250.

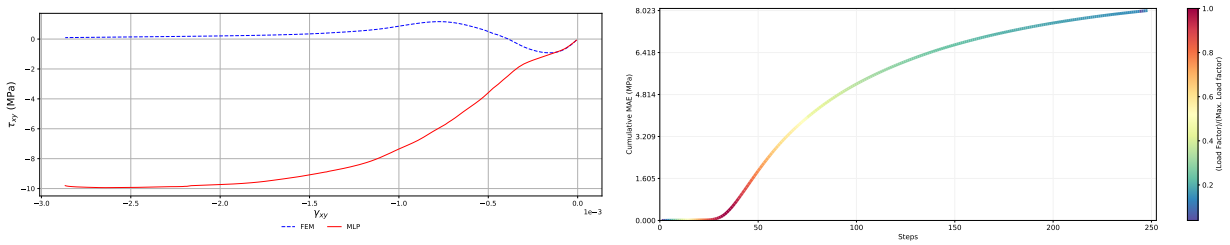
This time around, it was noted that, starting at the maximum load step, the MLP does not predict with reasonable accuracy the stress states in the regions where degradation propagates. In the rest of the structure, the predictions are assertive. To further investigate the behavior of the stress history predictions, the stress-strain curves were evaluated at two different points. The first assessed point was in the region where the main tensile stresses occur in the y-direction, point coincident with the Gauss integration point 2 of element 29 (Fig. 8.32). To better verify the point behavior, the same stress graphs were plotted until step 33, referring to the maximum load applied to the structure (Fig. 8.33). The second point analyzed was in the region where the structure experiences the highest tensile stresses in the x-direction, a point coincident with Gaus integration point 3 of element 46 (Fig. 8.34). Both points are represented in the Fig. 8.24b.



(a)

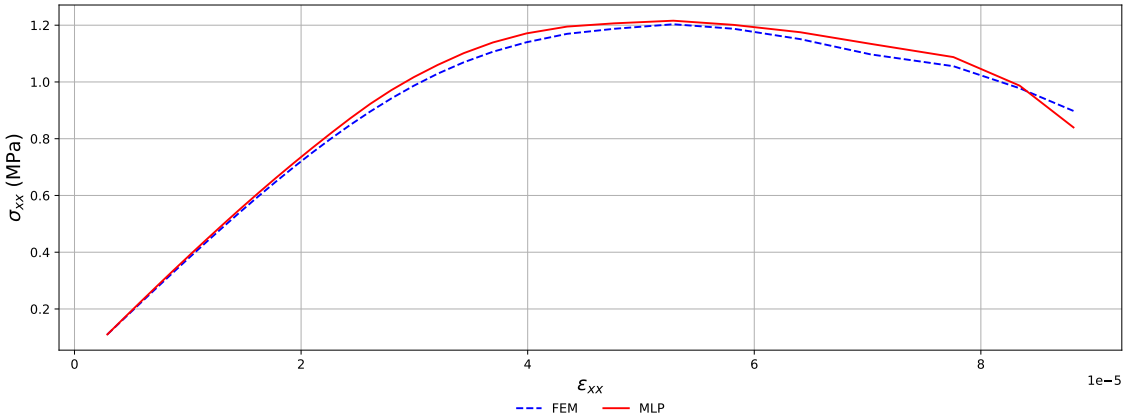


(b)

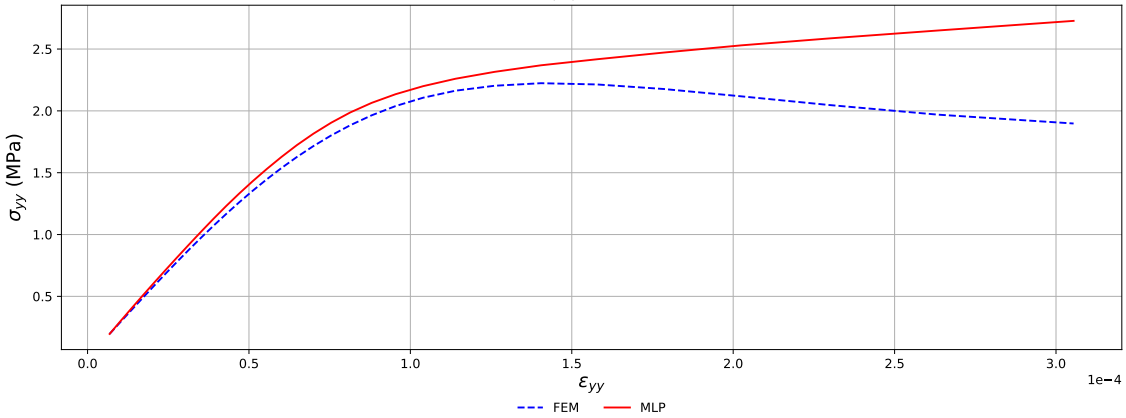


(c)

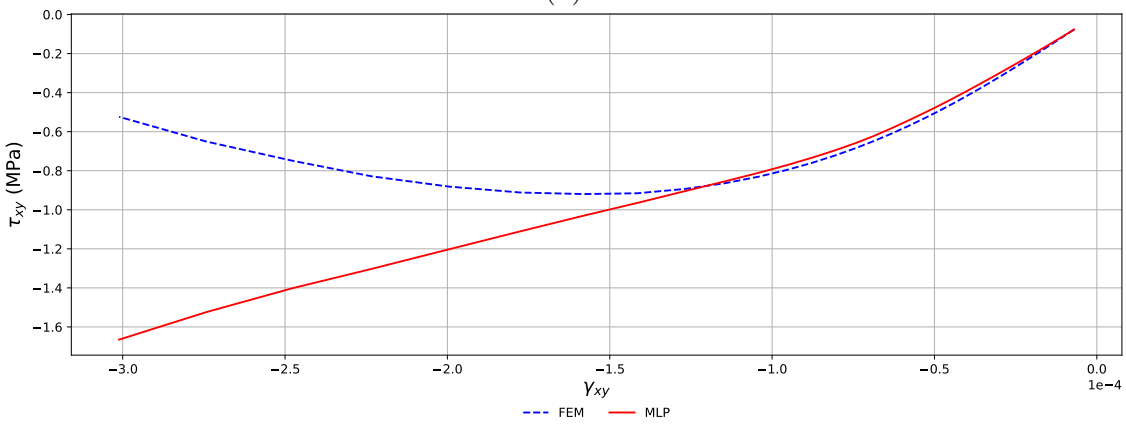
Figure 8.32: L-shaped panel - Global stress-strain history of E29 - IP2 - Comparison between FEM and MLP results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)

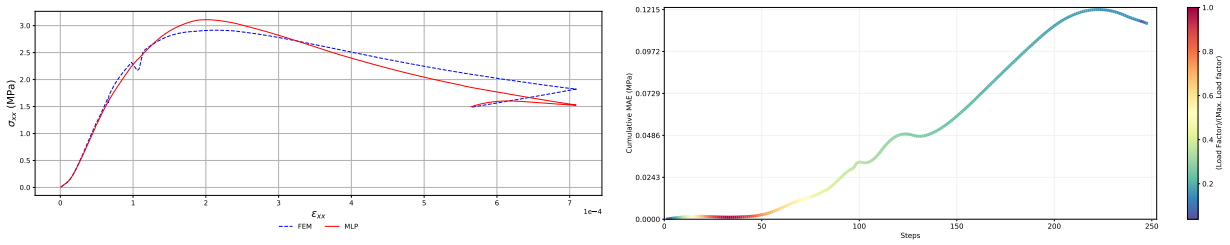


(b)

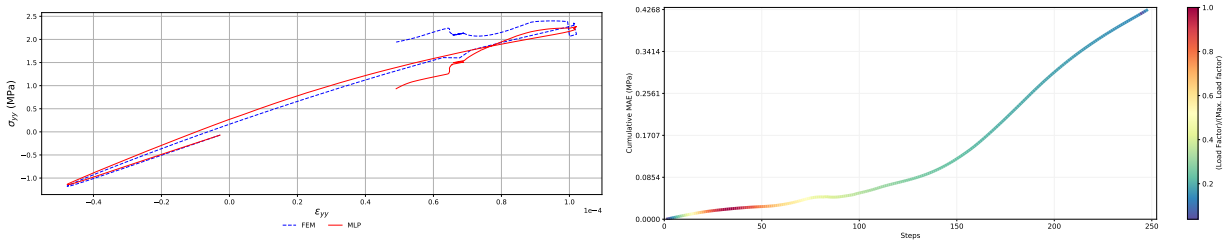


(c)

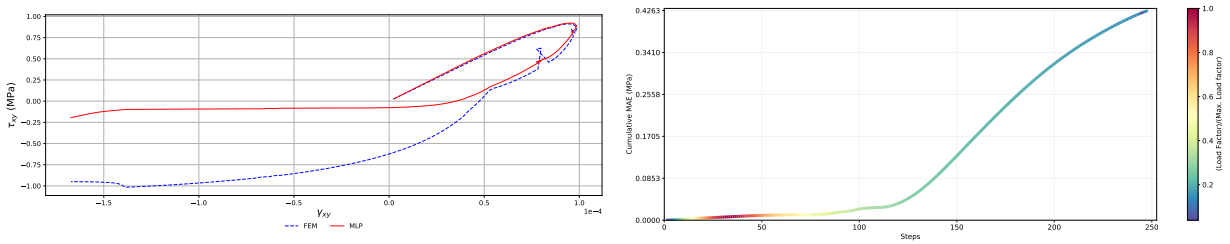
Figure 8.33: L-shaped panel - Global stress-strain history of E29 - IP2 until step 33 (Maximum Load) - Comparison between FEM and MLP results. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)



(b)



(c)

Figure 8.34: L-shaped panel - Global stress-strain history of E46 - IP3 - Comparison between FEM and MLP results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.

The stress-strain curves previously presented demonstrate that MLP was not able to predict the stress states for a point under y-bending stresses as well as it did for a point with x-bending stresses. Similar behavior for y-bending stresses was observed in the analyses of the MLP performance test on top of the asymmetric traction test. This inability, along with an analysis of the distribution of the training data presented in the Chapter 5, Section 5.2, may indicate that the training dataset lacked examples that represented stress states with bending rupture and tensile normal stresses in the y-direction. The graphs show that although the maximum tensile stresses were present in both directions (Fig. 5.17), that the strains in the x-direction reached higher values than the strains in the y-direction (Fig. 5.18). This analysis converges with that observed in Fig. 8.23, where MLP was able to represent well the material behavior in which tensile stresses assume high values, but that unloading occurs and therefore strains decrease. Here, it is important to emphasize that the material behavior and therefore the prediction

of the network is dependent on the combination of the three stress states of the problem, and that therefore this combination of stresses was insufficient (with few values) in the database, and not necessarily only the knowledge of a tensile softening curve in the y-direction. So, a point of improvement would be to study adding to the training data, examples that better represent the stress states of the regions where the network was not able to predict well.

8.4 Size Effect Analysis - Brazilian Splitting Test

The Scale Effect, or Size Effect, can be understood as a change in the overall behavior of any structure when its dimensions are changed without modifying its geometry and the physical properties of the material. The scale effect manifests itself in the behavior of the element in the post-peak and also in relation to the limit load (strength limit). One way to see the infuence of the size effect on the structural response is through the Brazilian splitting test, presented in Section 5.1.6 and illustrated in Fig. 8.35. When analyzing the equilibrium paths (Fig. 5.14 and Fig. 5.15) it was realized that the smaller the crack, the greater the effective height of the cylinder and the more brittle is the structural behavior. The opposite is also true, that is, the larger the crack, the smaller the effective height of the cylinder and the more ductile is the structural response.

This section aims to determine whether the MLP is able to predict the changes in structure behavior inherent to the scale effect in the analyzed structure. For this, the models with cracks of 4, 12 and 20 mm were used as training tests, and the models with cracks of 8, 16, 24 and 28 mm were used for validation.

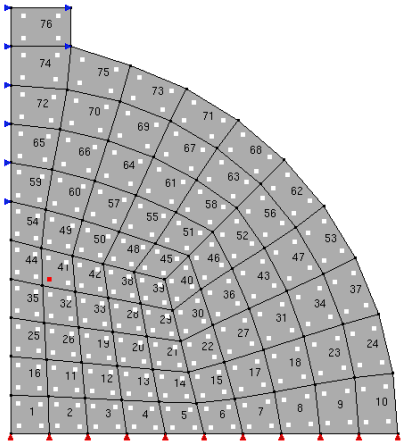


Figure 8.35: Brazilian splitting test setting taking advantage of the symmetry.

The equilibrium paths and analysis points of the validation tests are restated in Fig. 8.36.

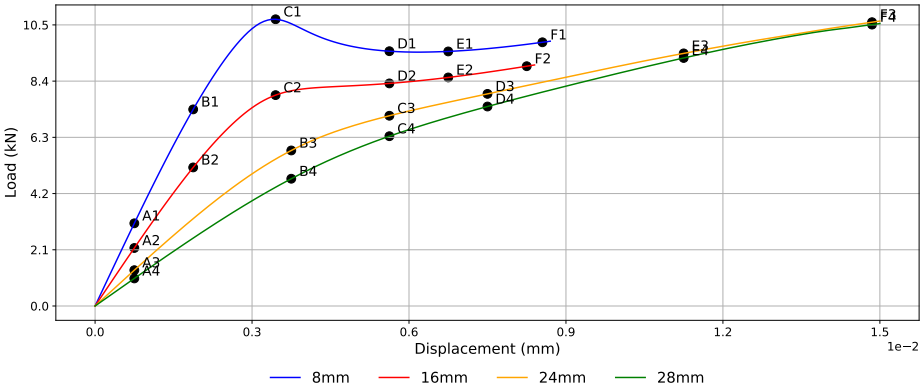


Figure 8.36: Load-displacement curve for the Brazilian Splitting Test - Horizontal displacement of the controlled node.

In this example, it is known that the strength of the specimen is driven by the tensile strength of the concrete. Therefore, only the normal stress component in the x-direction was evaluated. However, the responses of the other components are presented in Appendix C. Thus, once again, the normal stresses in the x-direction obtained via FEM and those predicted by MLP were compared for all 4 test models. To follow the stress responses along the structure, a straight line coincident with the cylinder radius was positioned, which separates the 1/4 of the modeled cylinder into 2 equal parts.

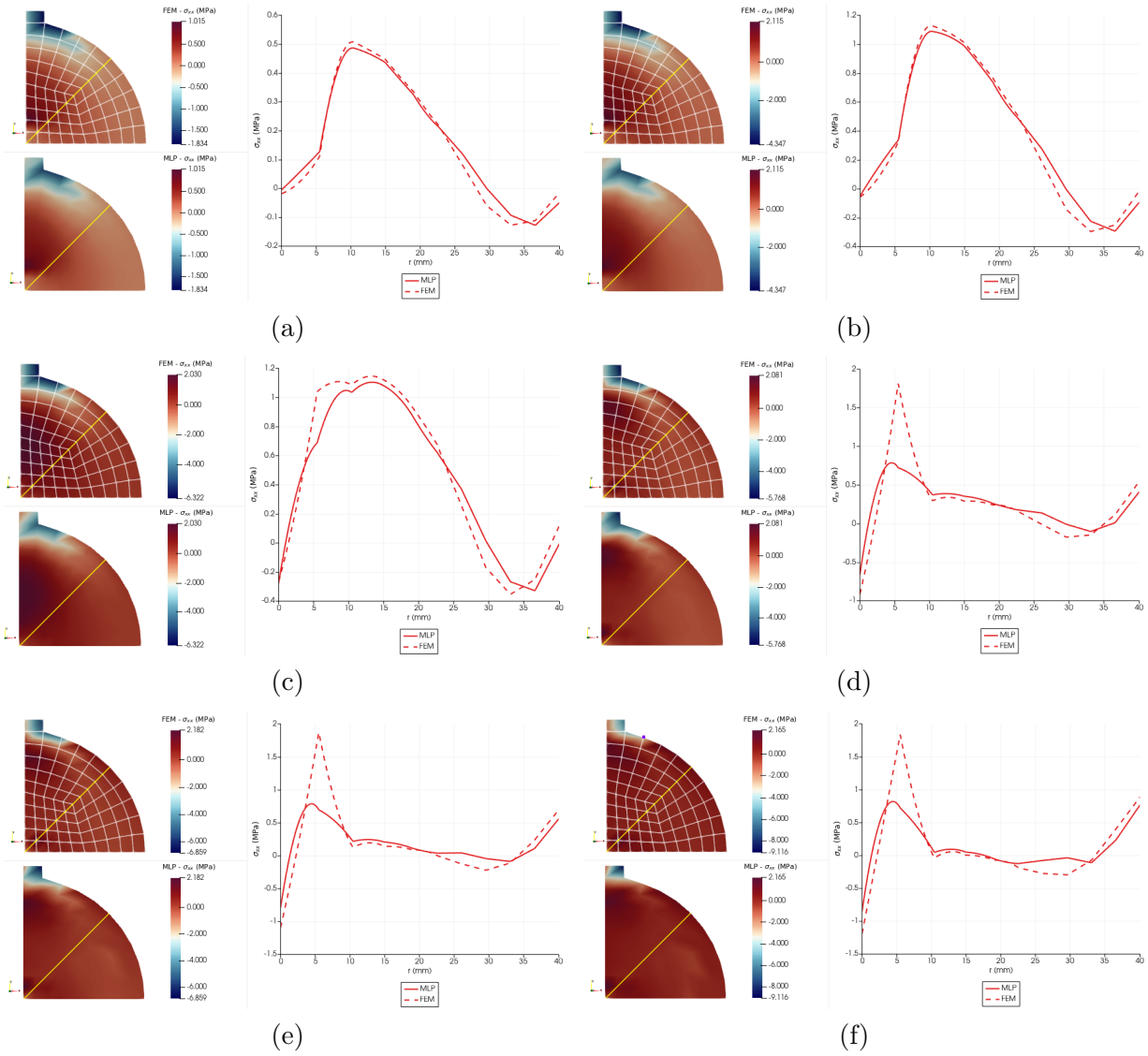


Figure 8.37: Diametrical compression with crack size of 8mm - Stresses FEM × MLP - σ_{xx}. (a) Point A1 Step 10, (b) Point B1 Step 25, (c) Point C1 Step 46, (d) Point D1 Step 75, (e) Point E1 Step 90, (f) Point F1 Step 114.

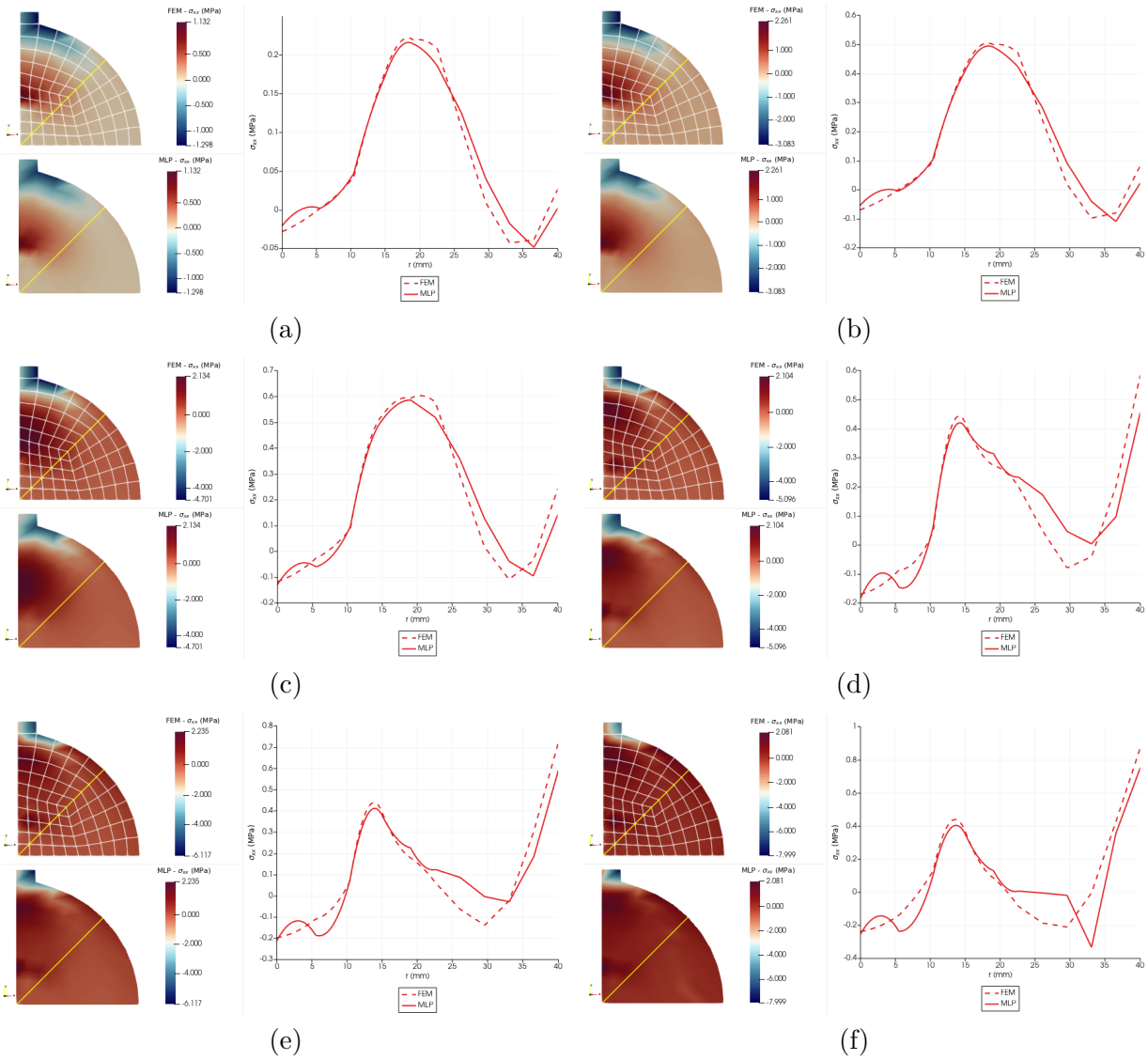


Figure 8.38: Diametrical compression with crack size of 16mm - Stresses FEM × MLP - σ_{xx}. (a) Point A2 Step 10, (b) Point B2 Step 25, (c) Point C2 Step 46, (d) Point D2 Step 75, (e) Point E2 Step 90, (f) Point F2 Step 110.

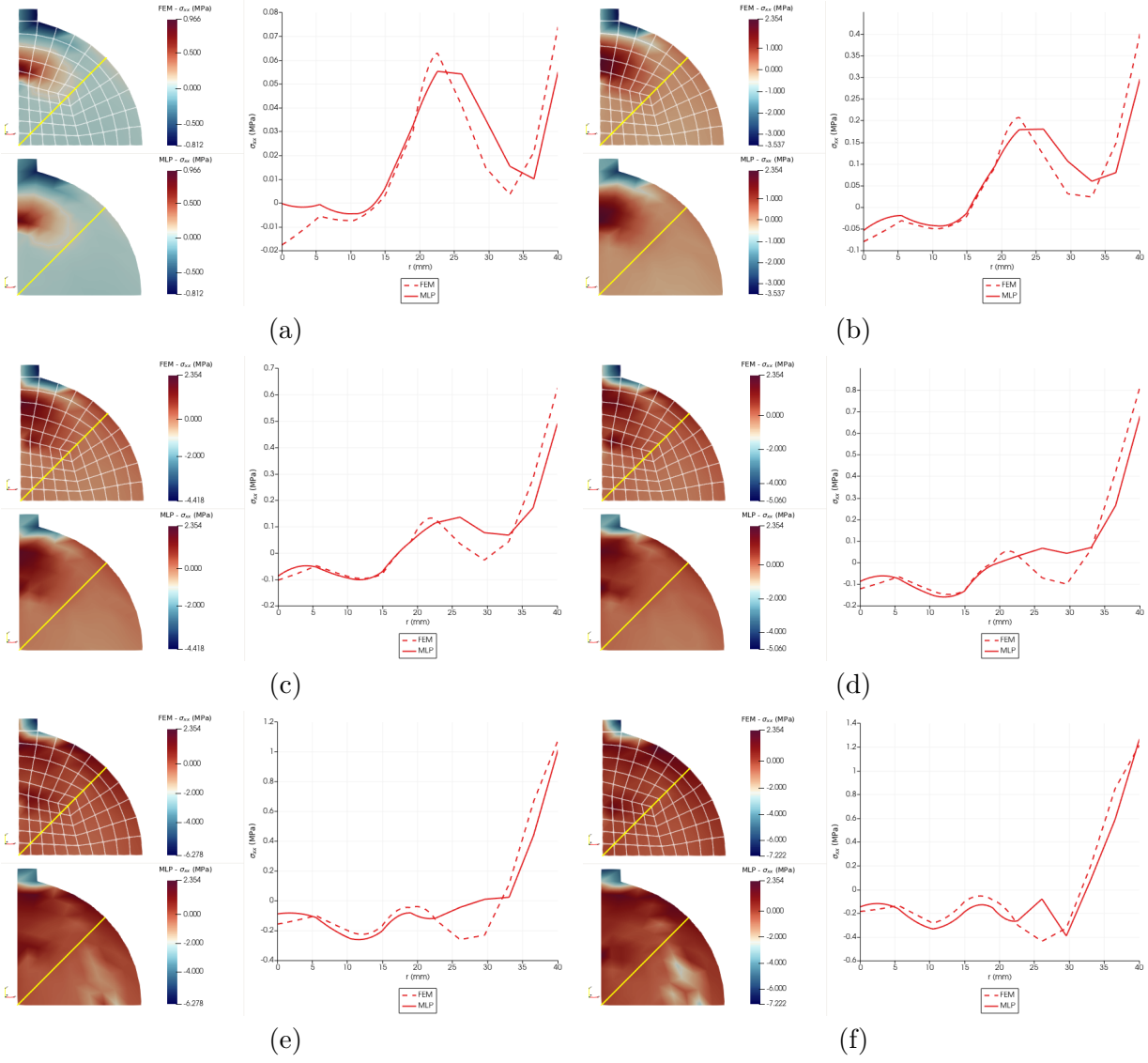


Figure 8.39: Diametrical compression with crack size of 24mm - Stresses FEM × MLP - σ_{xx}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

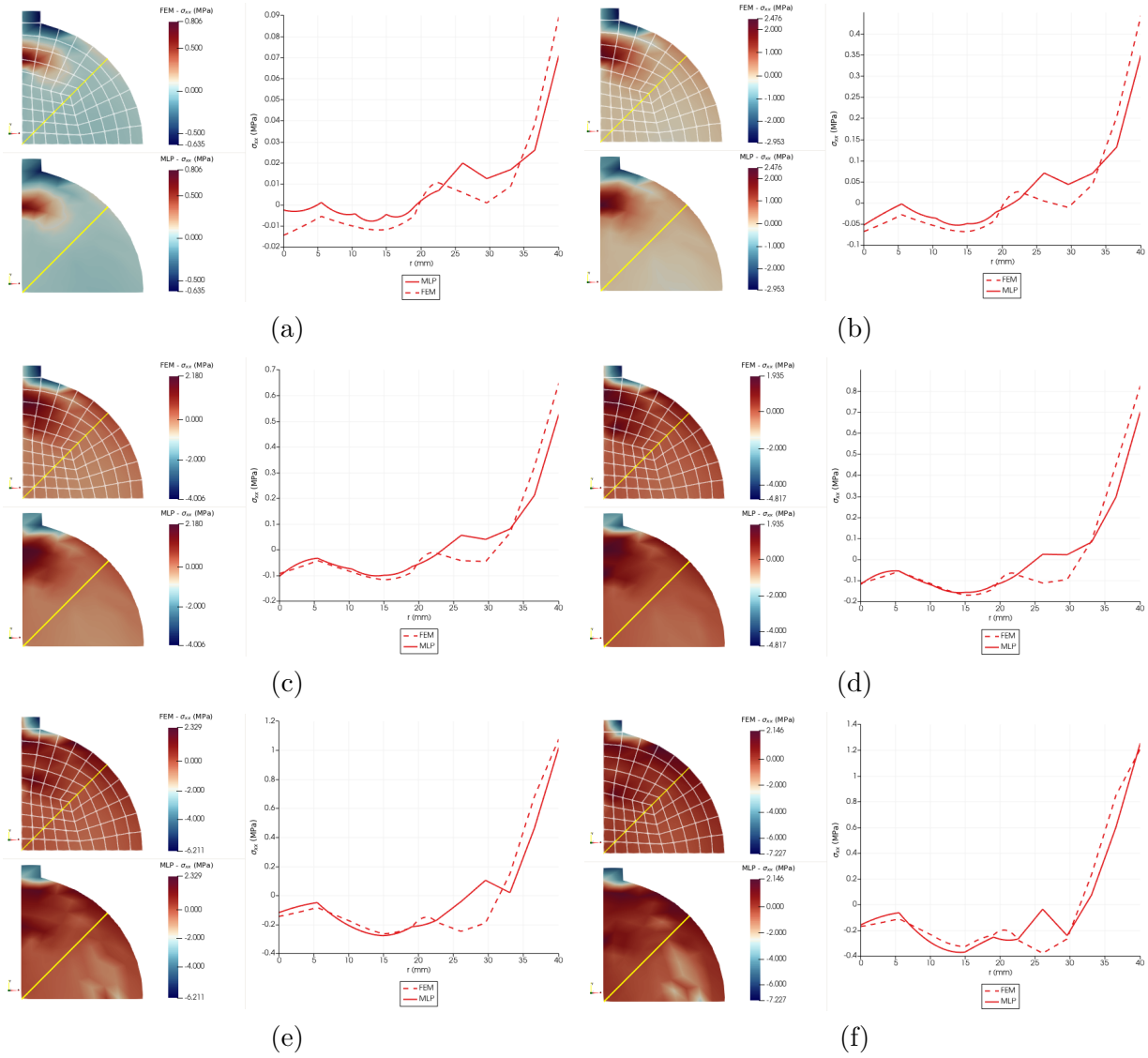


Figure 8.40: Diametrical compression with crack size of 28mm - Stresses FEM × MLP - σ_{xx}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

Next, the behavior of the MAE metric was evaluated, along the equilibrium path of the problem, at the same points previously analyzed.

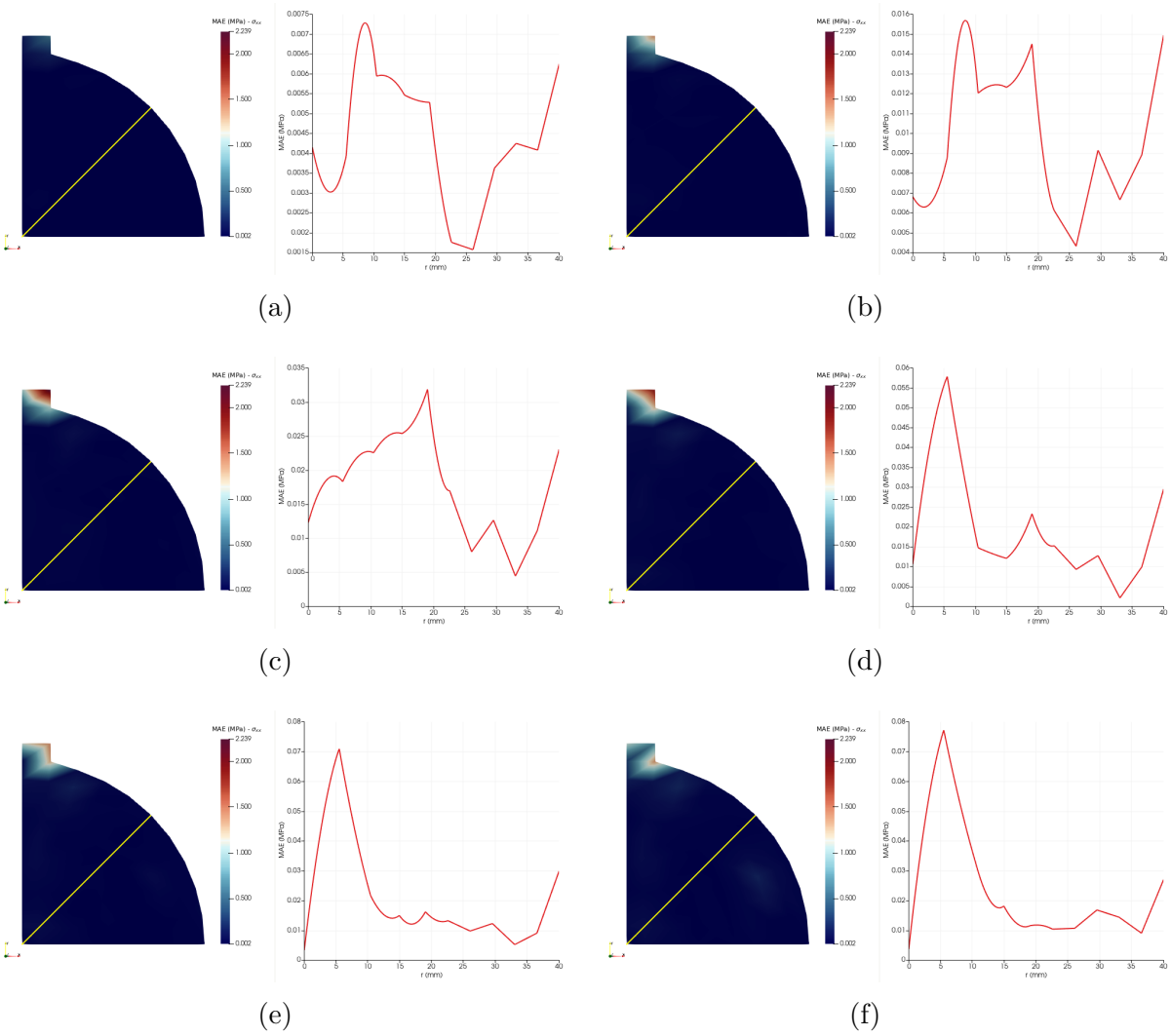


Figure 8.41: Diametrical compression with crack size of 8mm - MAE accumulated - σ_{xx}. (a) Point A1 Step 10, (b) Point B1 Step 25, (c) Point C1 Step 46, (d) Point D1 Step 75, (e) Point E1 Step 90, (f) Point F1 Step 114.

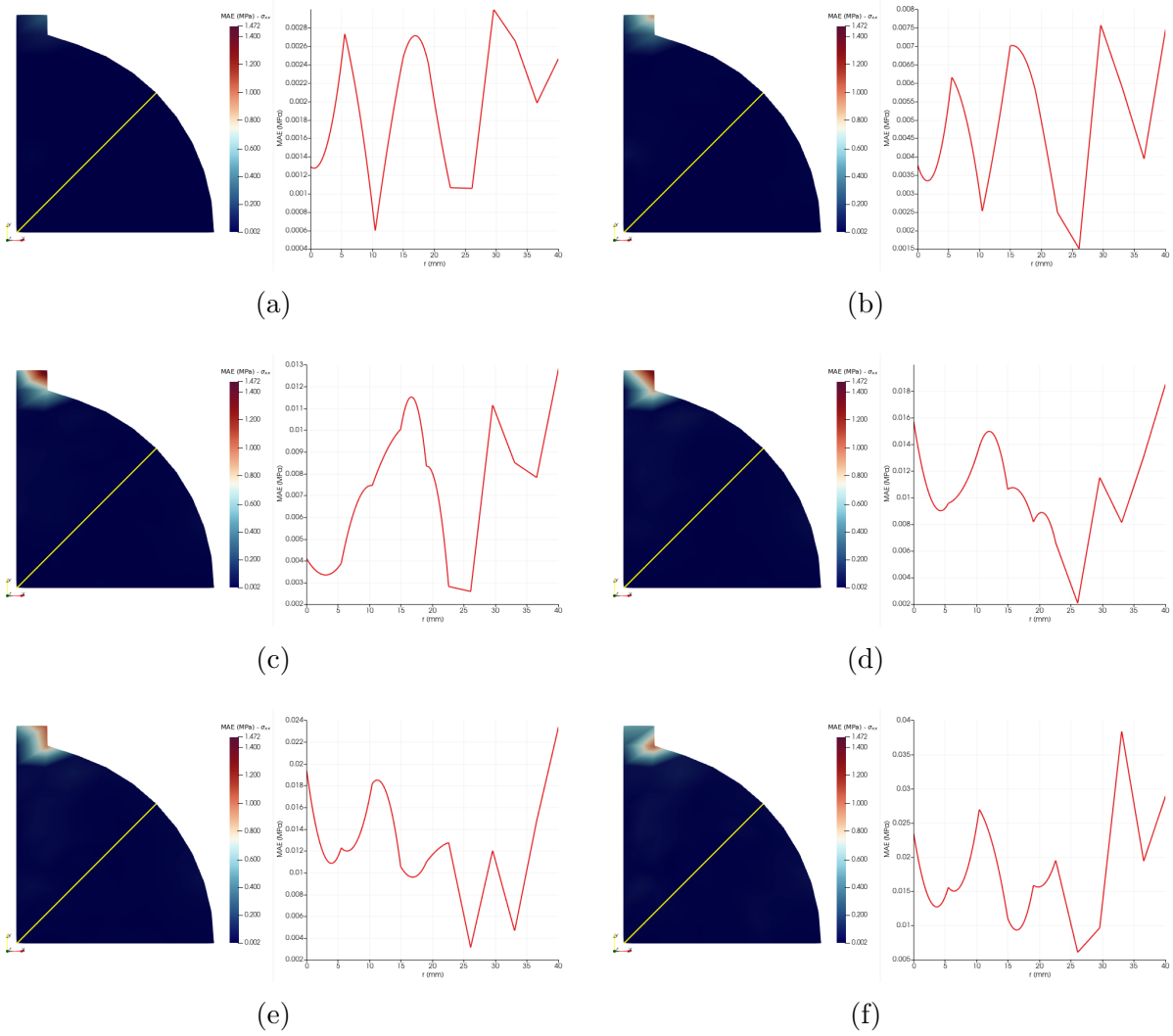


Figure 8.42: Diametrical compression with crack size of 16mm - MAE accumulated - σ_{xx}. (a) Point A2 Step 10, (b) Point B2 Step 25, (c) Point C2 Step 46, (d) Point D2 Step 75, (e) Point E2 Step 90, (f) Point F2 Step 110.

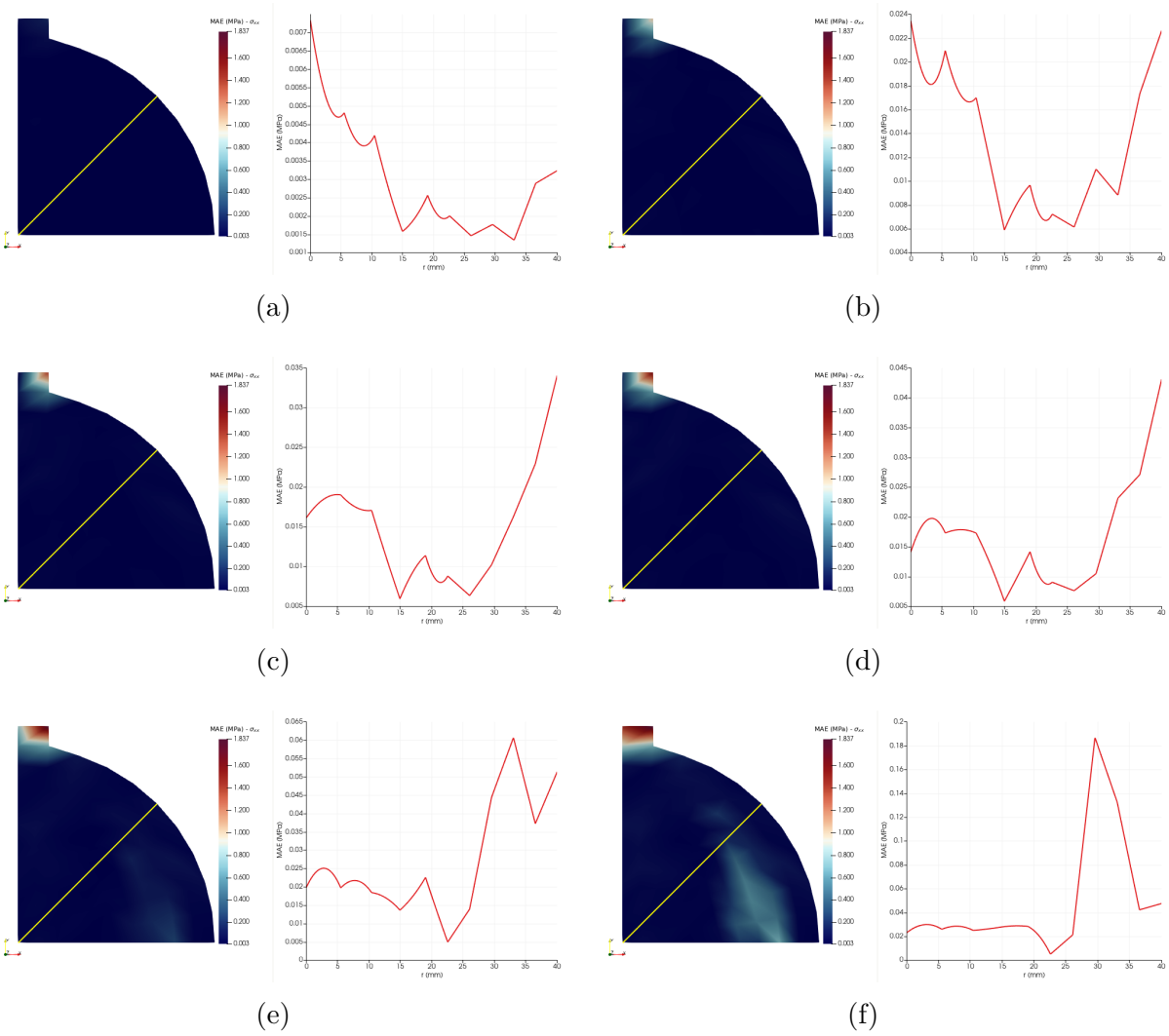


Figure 8.43: Diametrical compression with crack size of 24mm - MAE accumulated - σ_{xx}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

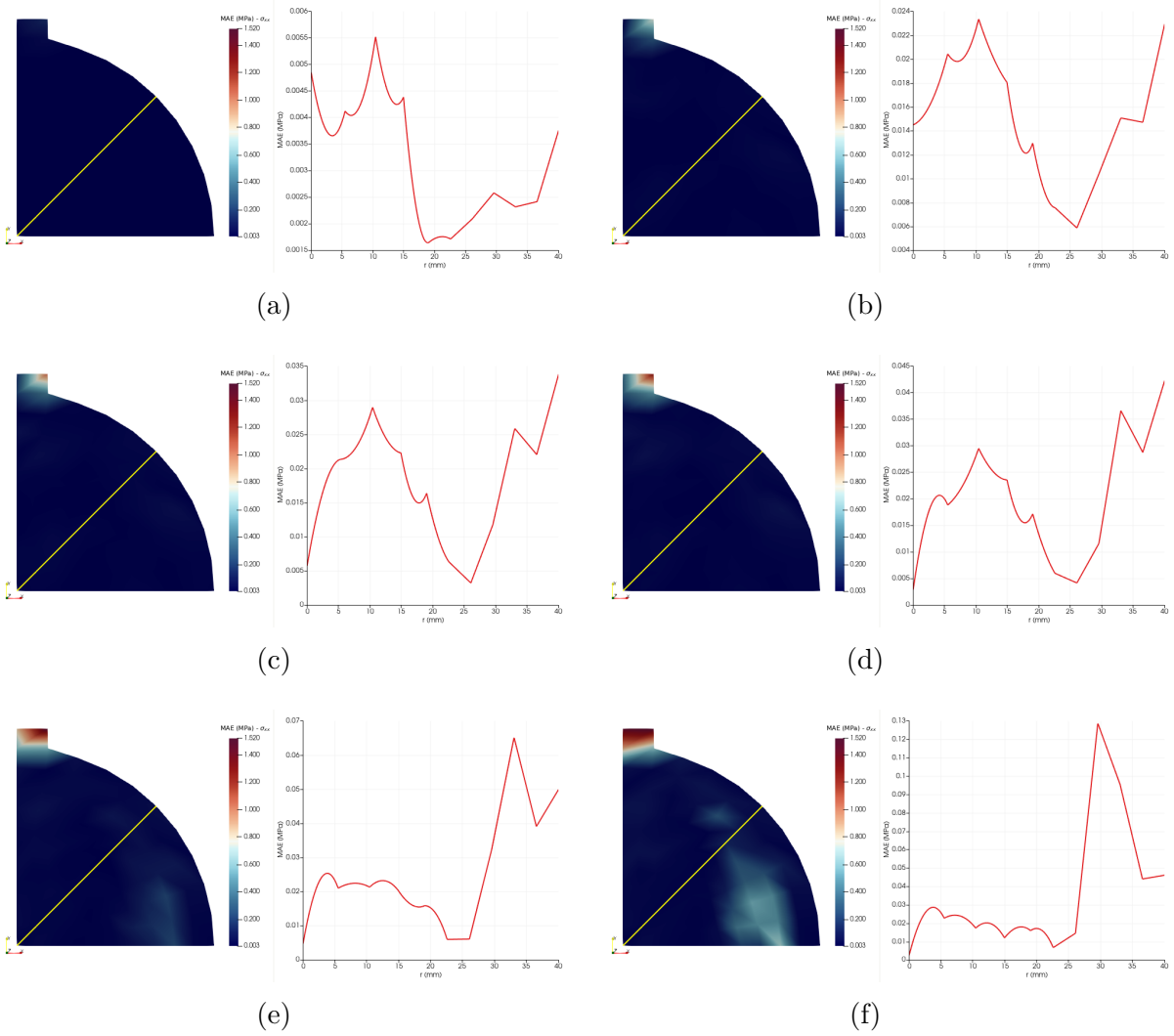
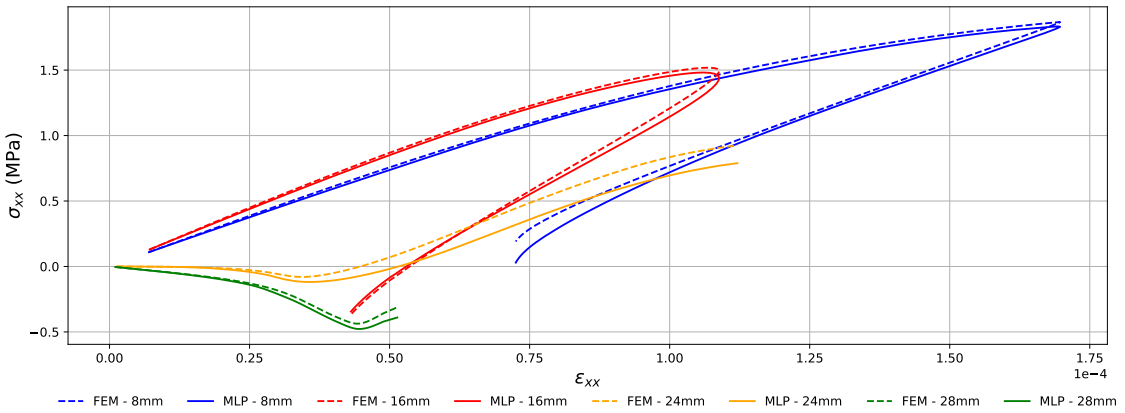


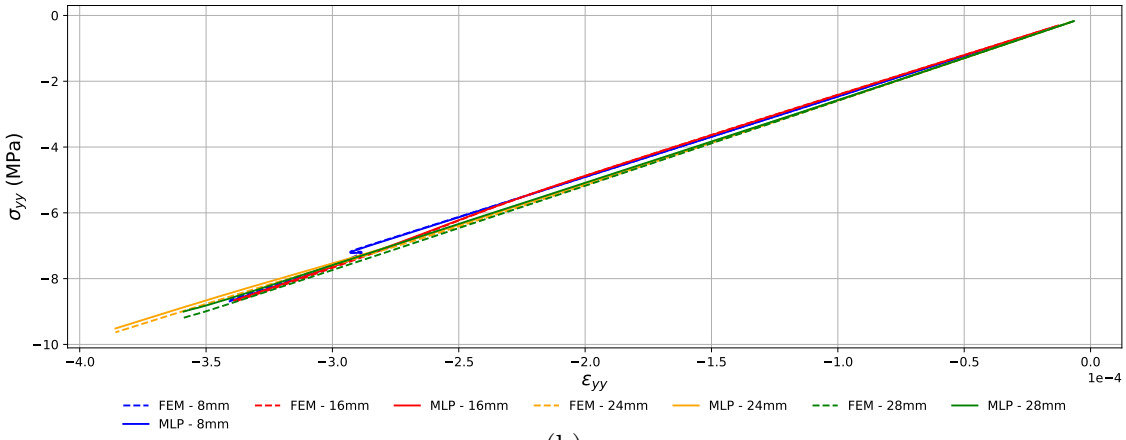
Figure 8.44: Diametrical compression with crack size of 28mm - MAE accumulated - σ_{xx}. (a) Point A4 Step 10, (b) Point B4 Step 50, (c) Point C4 Step 75, (d) Point D4 Step 100, (e) Point E4 Step 150, (f) Point F4 Step 198.

From the graphs above, there is excellent agreement between the stresses obtained via FEM and those predicted by MLP, for all 4 initial crack sizes. The largest MAE occurs in the rigid plate, where the test load was applied.

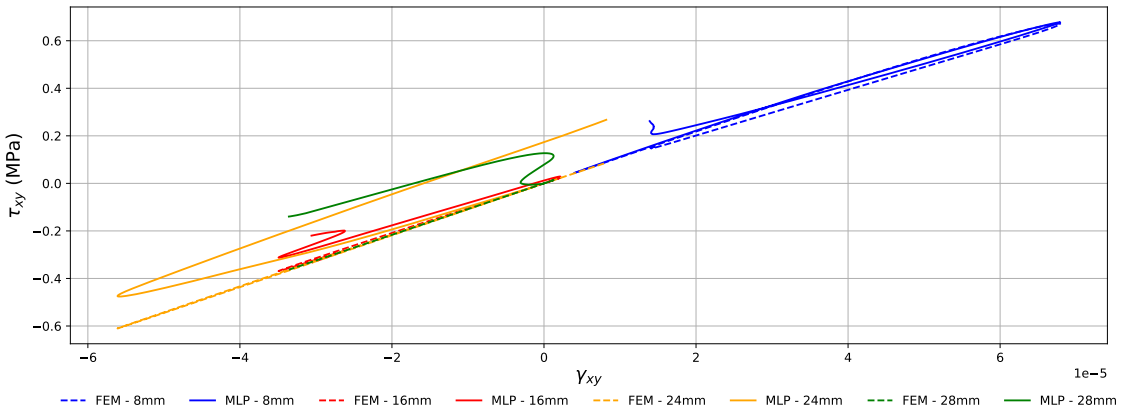
Finally, the MLP was evaluated in predicting the stress histories of a common point (Gauss integration point 4 of element 41) for all 4 models studied (Fig. 8.45) and for the curves of the maximum stress point for each of the tests (Fig. 8.46).



(a)

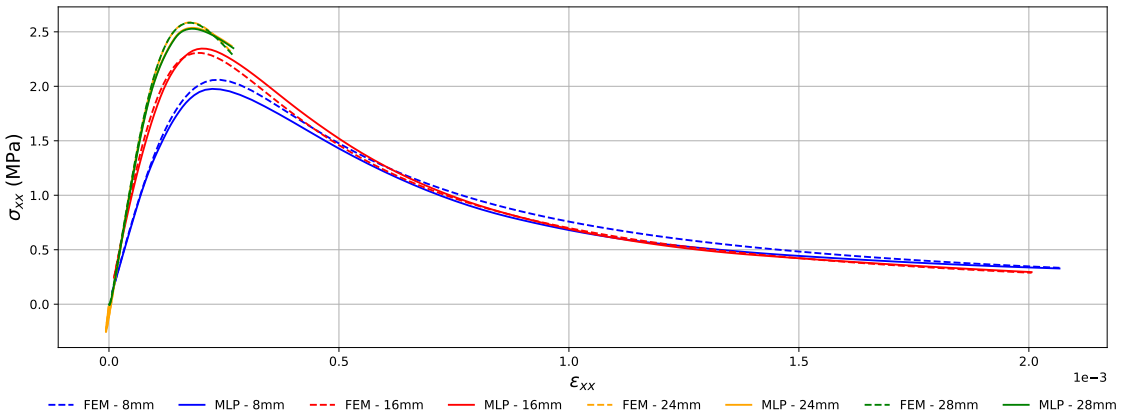


(b)

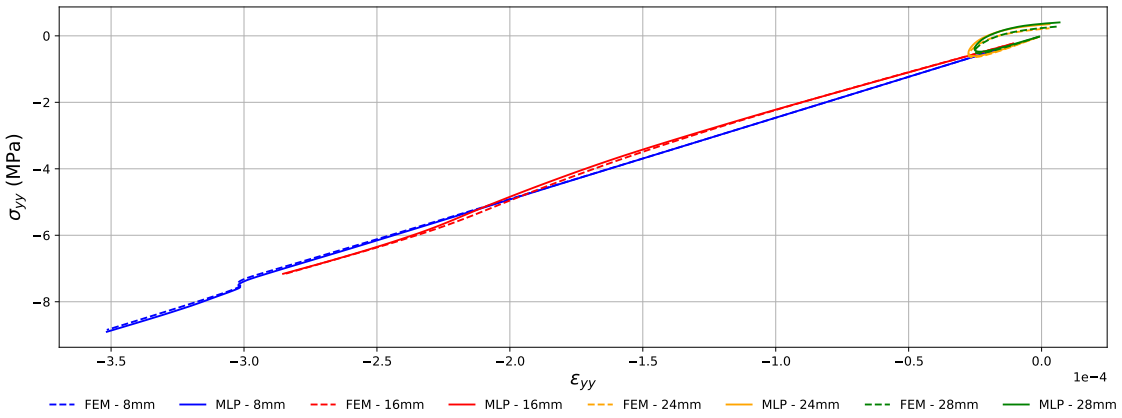


(c)

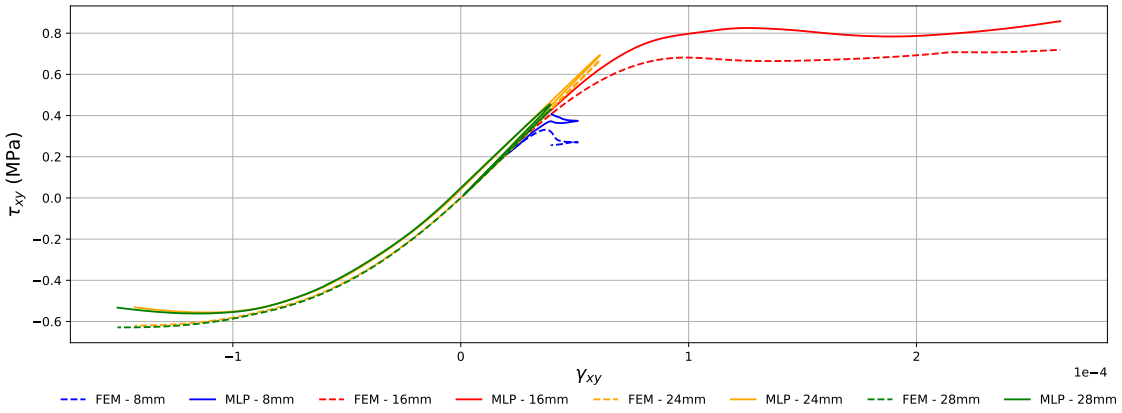
Figure 8.45: Diametrical compression - Global stress-strain history of E41 - IP4 - Comparison between FEM and MLP results. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)



(b)



(c)

Figure 8.46: Diametrical compression - Global stress-strain history of the maximum stress point - Comparison between FEM and MLP results. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.

By evaluating the graphs, it is noticeable that the MLP was able to accurately approximate the stress states of the validation tests. Fig. 8.45 shows that for the same analysis point, the models with the 4 different initial cracks presented distinct behaviors. This means that the model was able to generalize the material behavior, independently of the effective size of the structure.

8.5 Mesh Dependency Analysis - 3-point Bending Test

Finally, it was simulated as a structural system a 3-point bending test adapted from de Andrade e Santos (2015). The concrete beam has a size of $860 \times 200 \times 80$ mm^3 and a initial crack of 80 mm, as can be seen in Fig. 8.47. This system was chosen because a similar model was used in the training phase. The main purpose of using this as test data is to evaluate whether the trained network performs well when compared with FEM models that simulate the same problem, but with different levels of discretization. This is because, for the same structural system, FEM models with different discretization levels present different stress-strain histories.

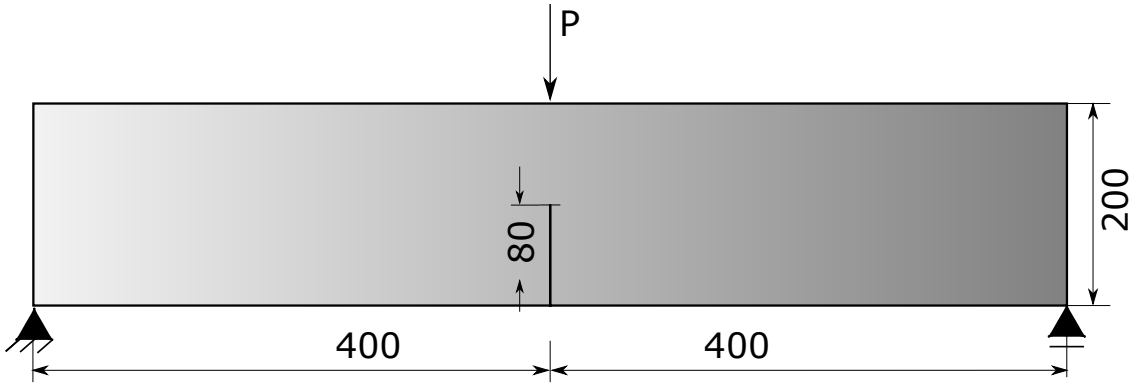
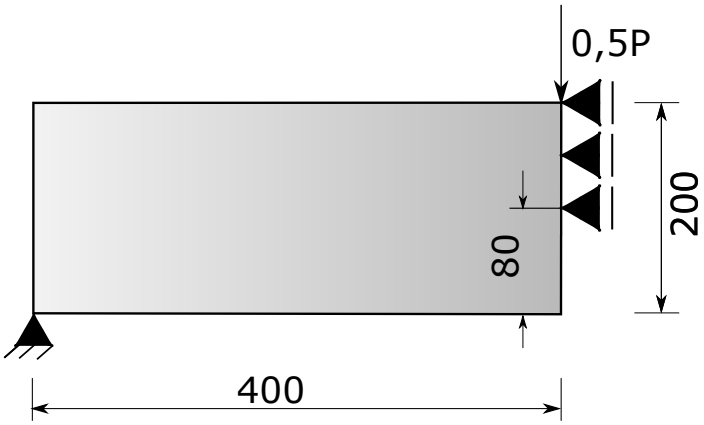
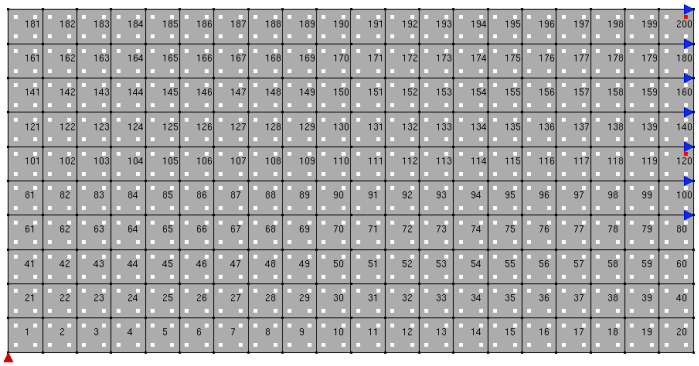


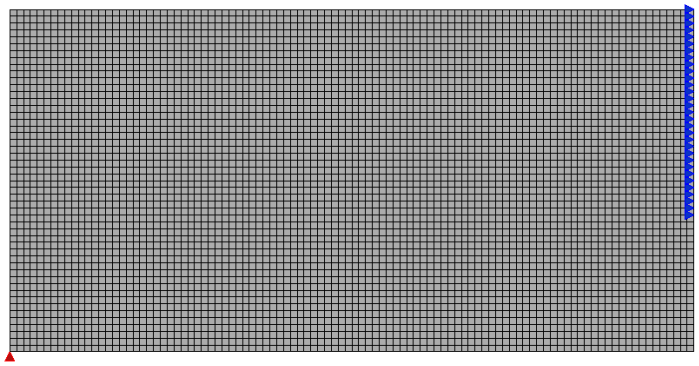
Figure 8.47: 3-point bending test adapted from de Andrade e Santos (2015) - Problem setting (measures in mm).



(a)



(b)



(c)

Figure 8.48: 3-point bending test adapted from de Andrade e Santos (2015). (a) Modeling, taking advantage from symmetry (b) Mesh 1 setting (c) Mesh 2 setting.

The structural system was simulated using the symmetry of the problem, using 2 distinct meshes, as described in Table 8.1 and illustrated in Fig. 8.48. In all models the displacement control method was used with monitoring of the upper central node (same node where the load was applied) with displacement increments equal to -0.001 and reference load $P = 400.0$ N. The equilibrium paths of each of the elaborated meshes and the points of the subsequent analysis are presented in Fig. 8.49.

Table 8.1: Mesh description

ID	No. of elements	Element size
M1	200	$20x20$
M2	5000	$4x4$

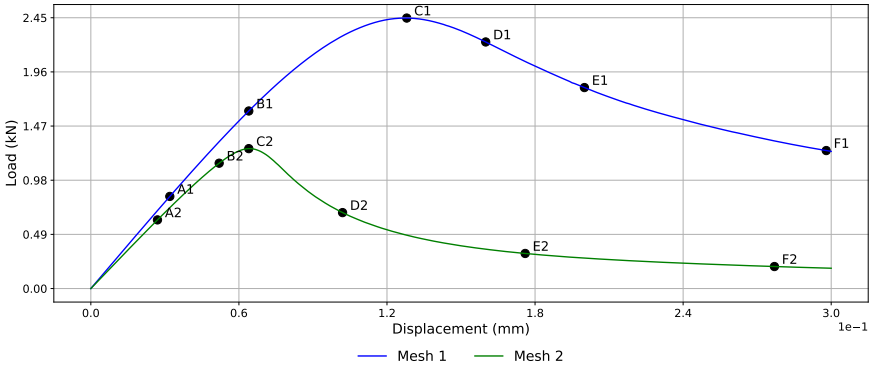
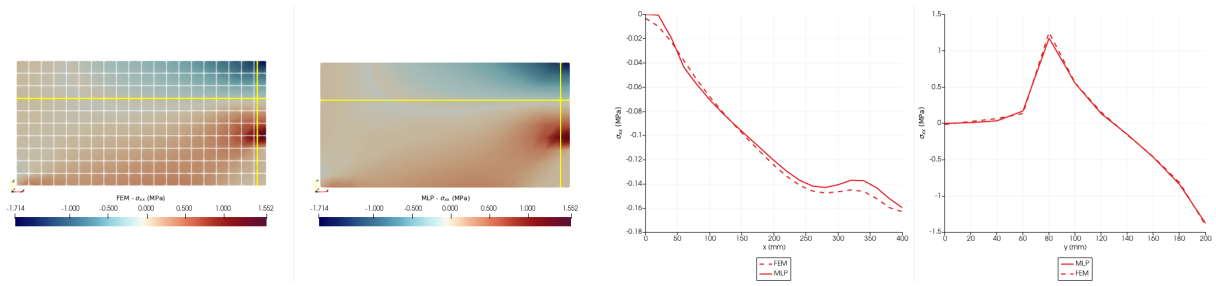
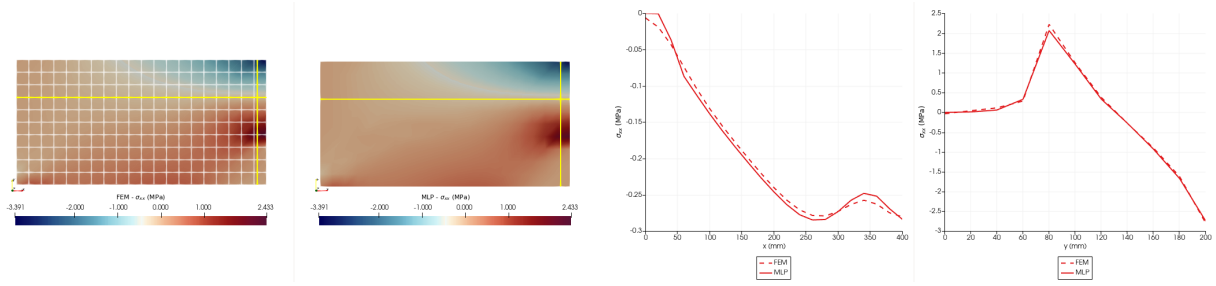


Figure 8.49: Load-displacement curve for 3-point bending test from de Andrade e Santos (2015) - Vertical displacement of the controlled node.

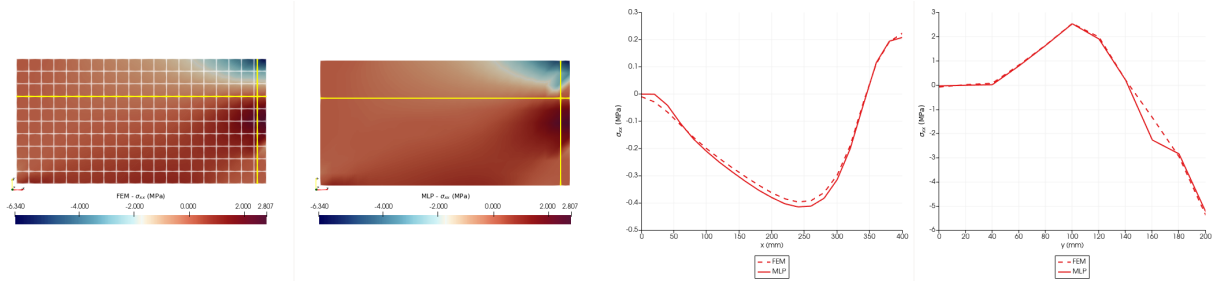
In the 3-point bending beam, the highest normal tensile and compressive stresses occur in the central region and the failure occurs by tension at mid-span, above the pre-existing crack. Thus, the control straight lines were set up in order to observe the behavior of the predictions in these regions. In this example, only the behavior of the normal stresses at x-direction was observed. Fig. 8.50 to Fig. 8.51 present the results for the M1 mesh and Fig. 8.52 to Fig. 8.53 for the M2 mesh.



(a)

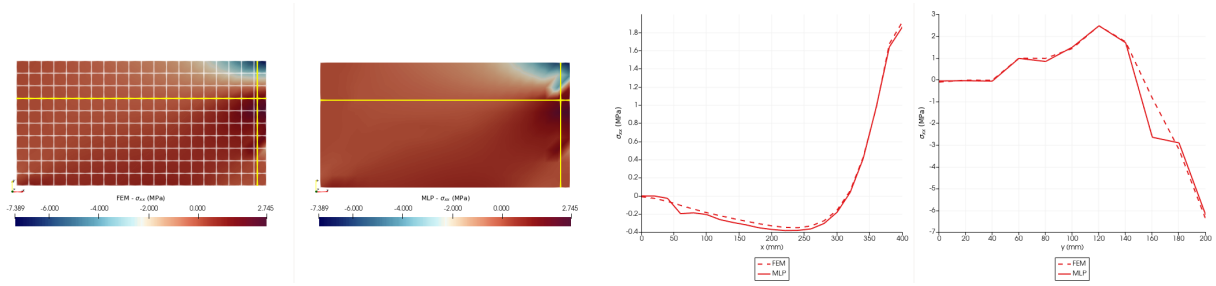


(b)

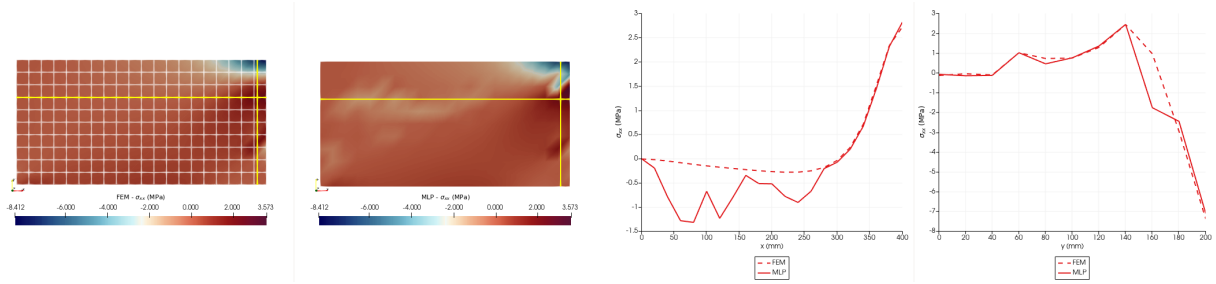


(c)

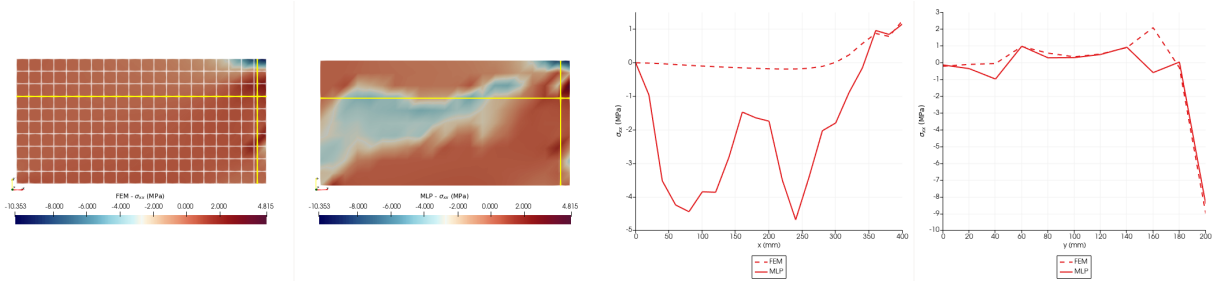
Figure 8.50: 3-point bending test - M1 - Stresses FEM × MLP - σ_{xx}. (a) Point A1 Step 32, (b) Point B1 Step 64, (c) Point C1 Step 128.



(a)

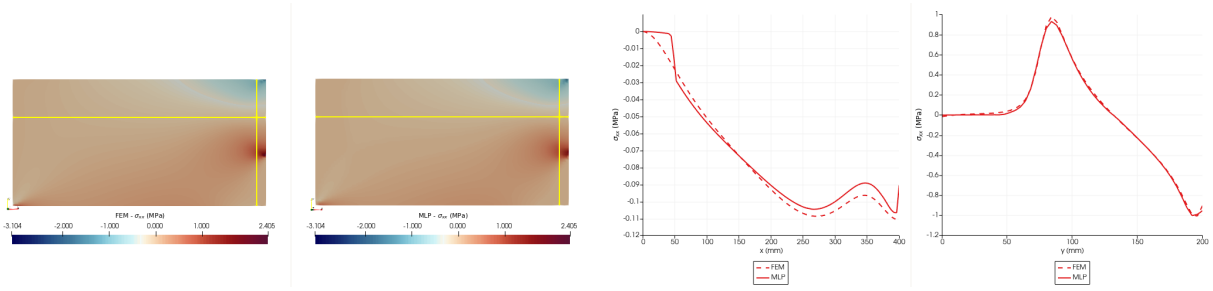


(b)

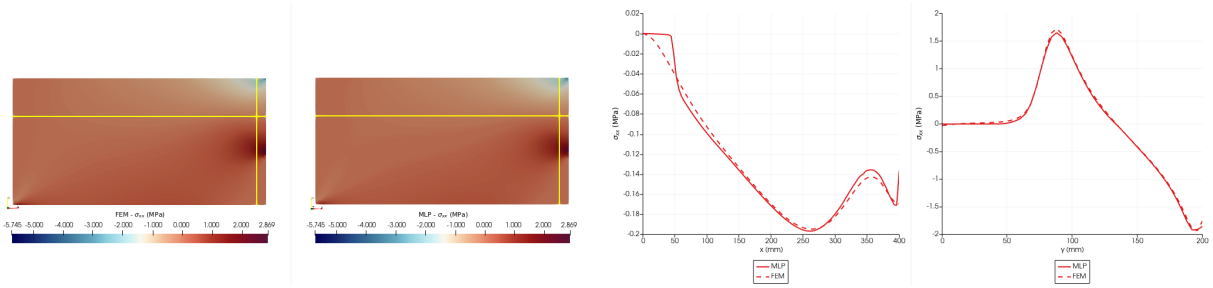


(c)

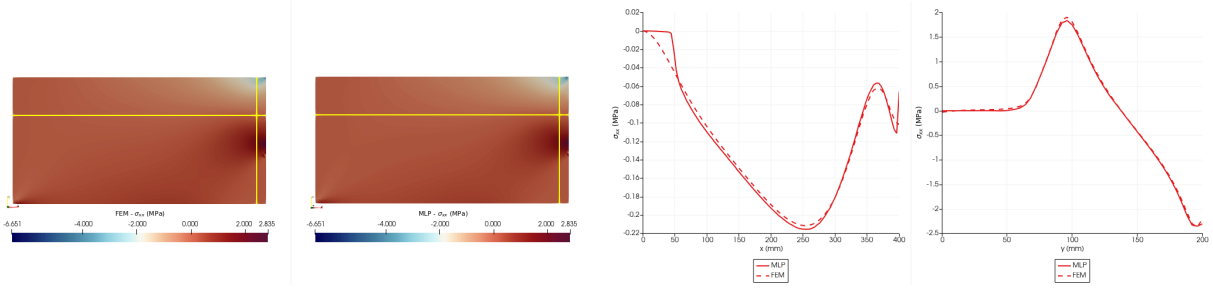
Figure 8.51: 3-point bending test - M1 - Stresses FEM × MLP - σ_{xx}. (a) Point D1 Step 160, (b) Point E1 Step 200, (c) Point F1 Step 298.



(a)



(b)



(c)

Figure 8.52: 3-point bending test - M2 - Stresses FEM × MLP - σ_{yy}. (a) Point A2 Step 27, (b) Point B2 Step 52, (c) Point C2 Step 64.

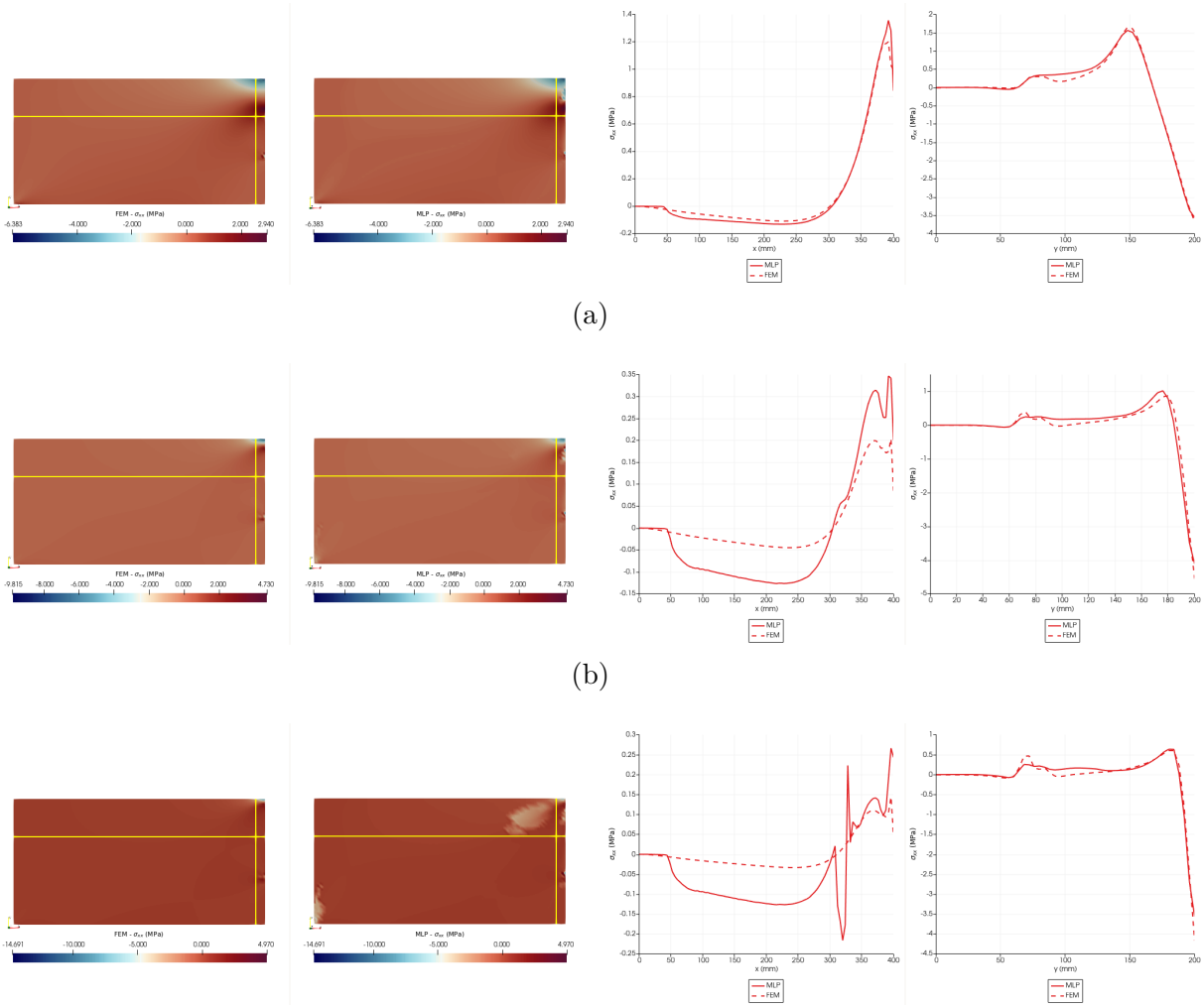


Figure 8.53: 3-point bending test - M2 - Stresses FEM × MLP - σ_{yy}. (a) Point D2 Step 102, (b) Point E2 Step 176, (c) Point F2 Step 277.

Then, once again, the behavior of the chosen metric along the equilibrium path of the problem was evaluated for the M1 and M2 meshes.

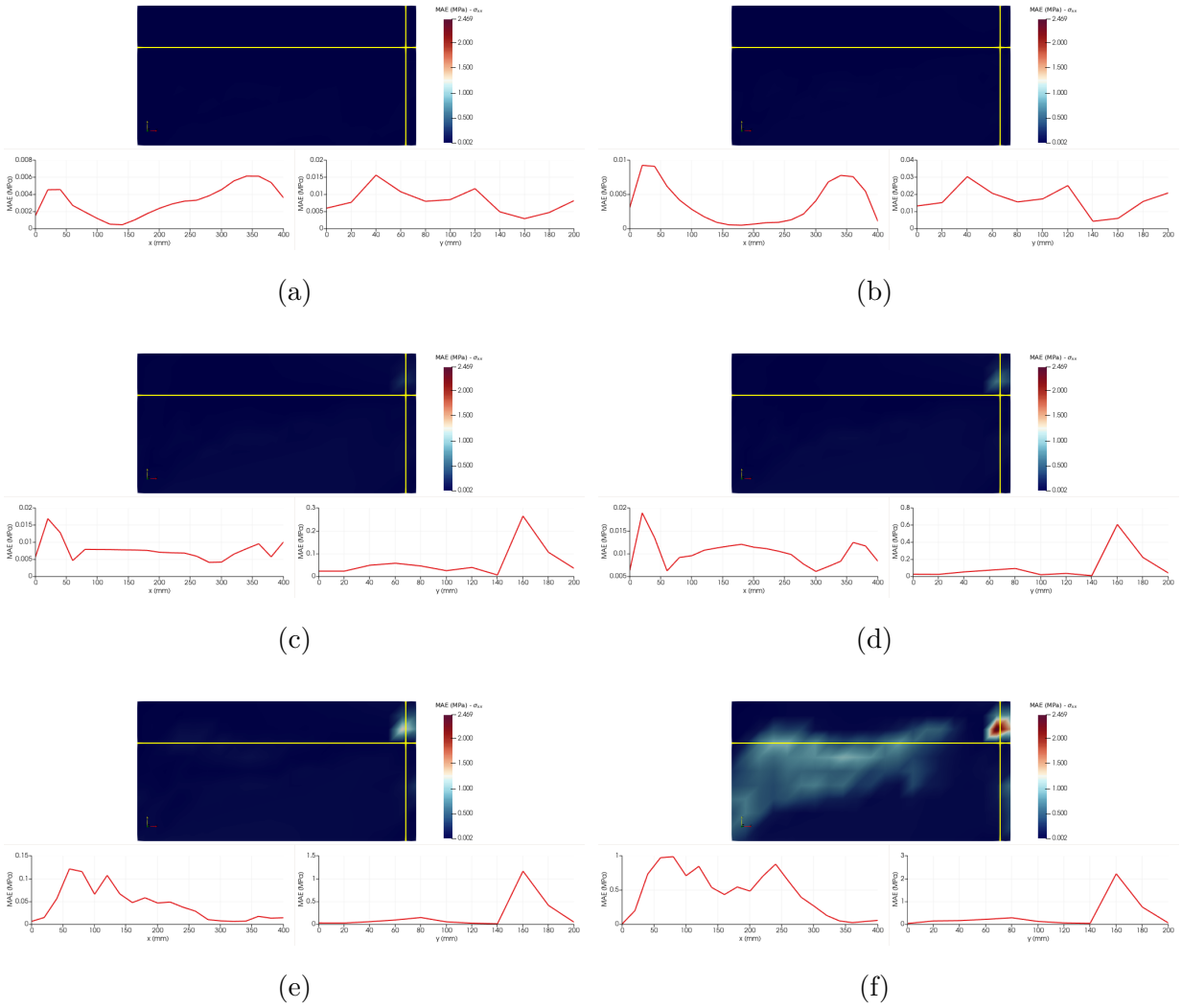


Figure 8.54: 3-point bending test - M1 - MAE accumulated - σ_{xx}. (a) Point A1 Step 32, (b) Point B1 Step 64, (c) Point C1 Step 128, (d) Point D1 Step 160, (e) Point E1 Step 200, (f) Point F1 Step 298.

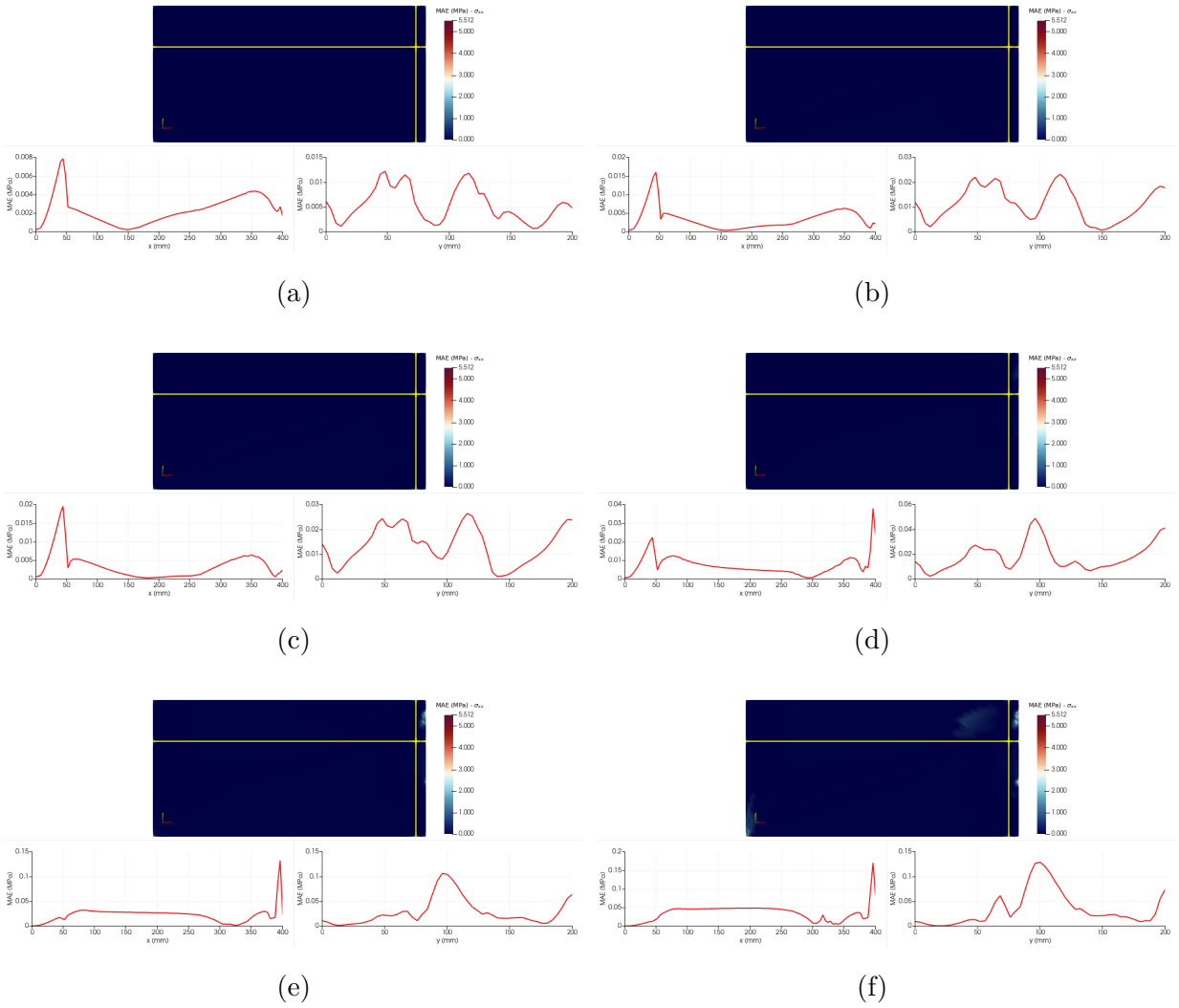
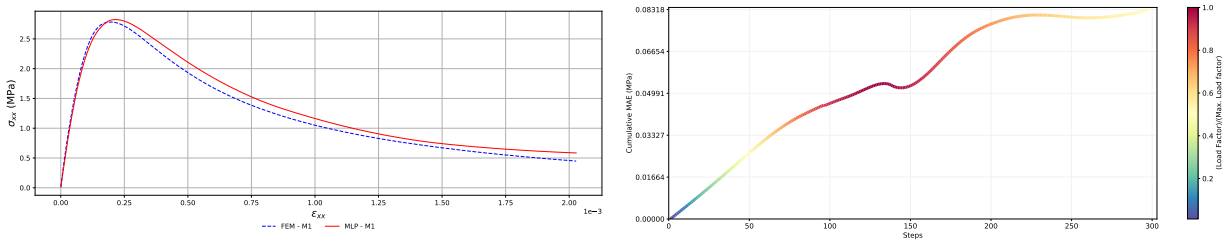
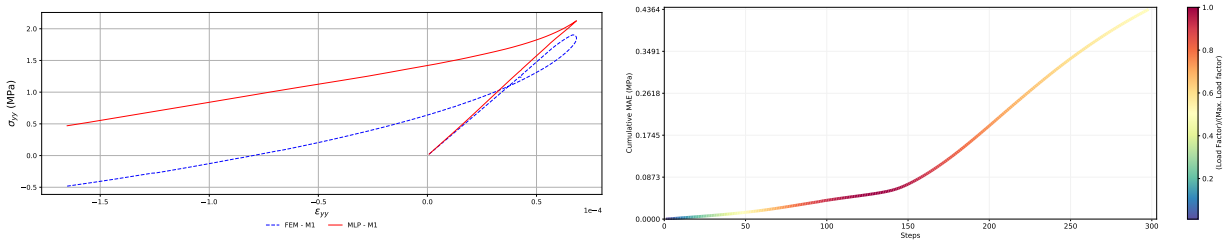


Figure 8.55: 3-point bending test - M2 - MAE accumulated - σ_{xx}. (a) Point A2 Step 27, (b) Point B2 Step 52, (c) Point C2 Step 64, (d) Point D2 Step 102, (e) Point E2 Step 176, (f) Point F2 Step 277.

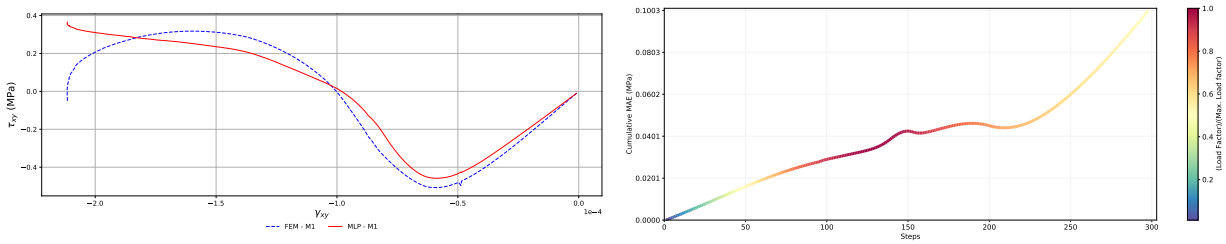
It was also sought to predict the path of the normal stresses at the Gauss integration points closest to the crack tip (Fig. 8.56 and Fig. 8.57) and in the compressed region of the beam (Fig. 8.58 and Fig. 8.59), for M1 and M2 meshes. Again, the analysis points are indicated in Fig. 8.48.



(a)

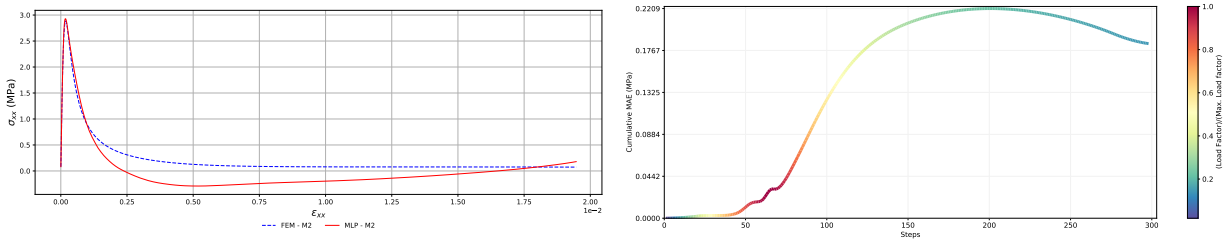


(b)

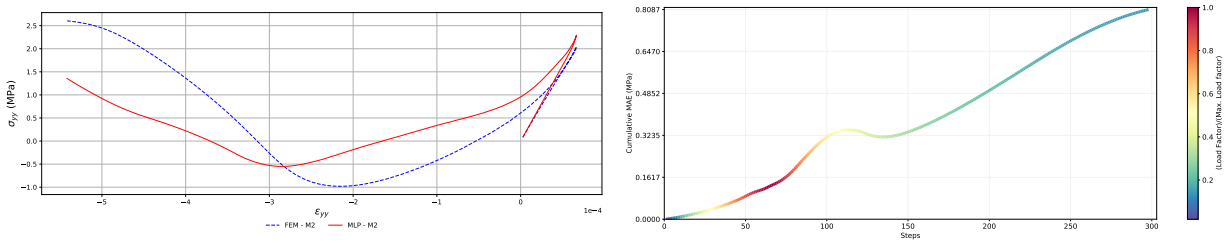


(c)

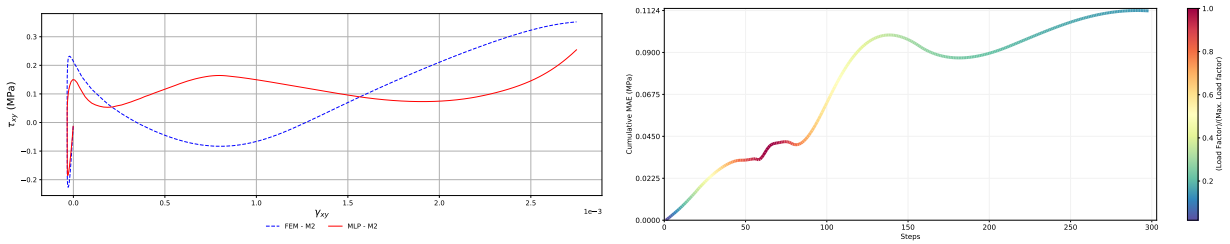
Figure 8.56: 3-point bending test - M1 - Global stress-strain history of E120 - IP1 - Tensile region - Comparison between MEF and MLPs results. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)

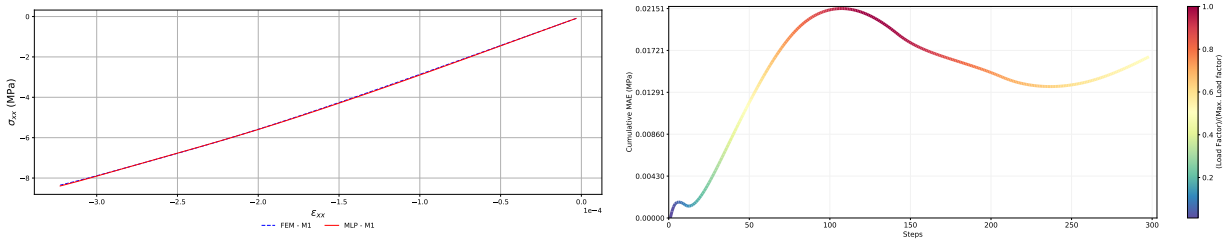


(b)

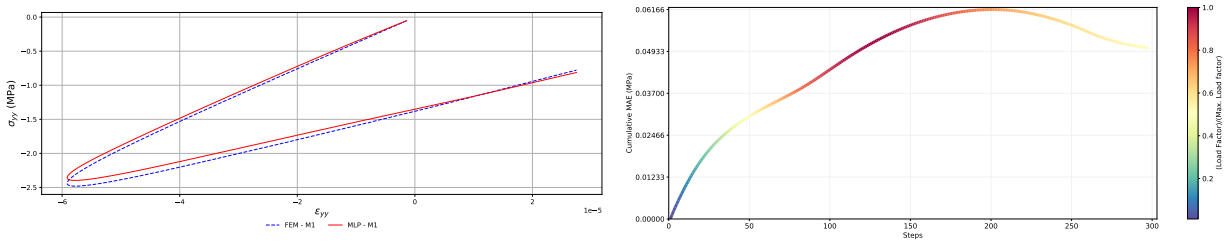


(c)

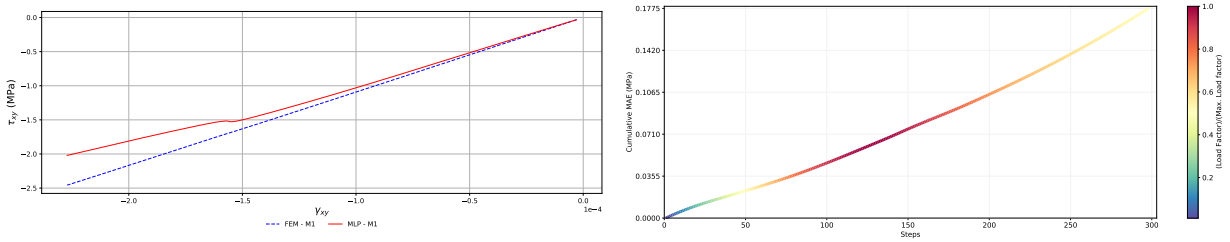
Figure 8.57: 3-point bending test - M2 - Global stress-strain history of E2200 - IP3 - Tensile region - Comparison between MEF and MLPs results. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)

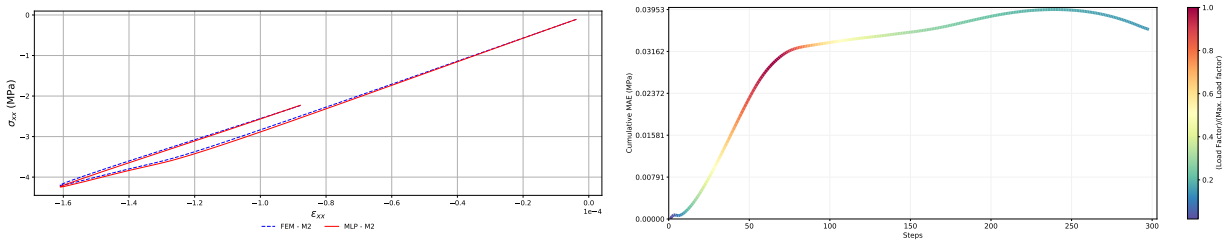


(b)

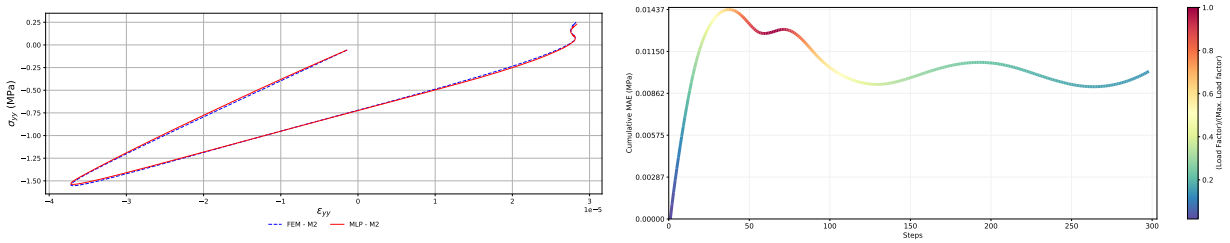


(c)

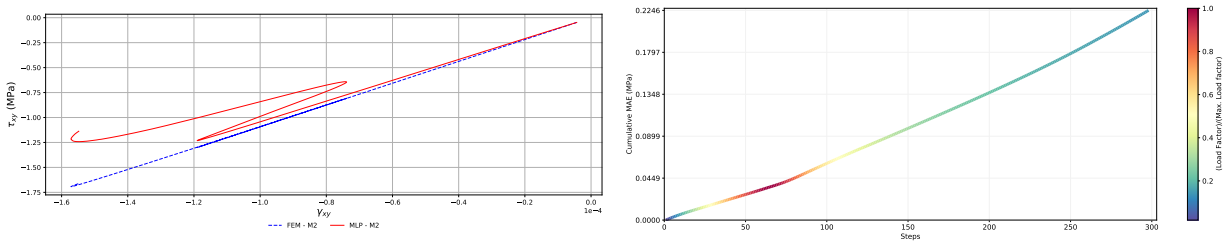
Figure 8.58: 3-point bending test - M1 - Global stress-strain history of E200 - IP1 - Compression region - Comparison between MEF and MLPs results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.



(a)



(b)



(c)

Figure 8.59: 3-point bending test - M2 - Global stress-strain history of E4899 - IP3 - Compression region -Comparison between MEF and MLPs results and Cumulative MAE. (a) $\sigma_{xx} \times \varepsilon_{xx}$, (b) $\sigma_{yy} \times \varepsilon_{yy}$, (c) $\tau_{xy} \times \gamma_{xy}$.

From the results observed earlier, it is understood that the MLP was able to predict with satisfactory accuracy the stress histories of the tested beam, regardless of the mesh discretization level. Although the process used is not able to reproduce the structural behavior (but rather point-to-point), it turns out that the local responses are similar to those obtained in finite element simulations. Here, it is important to emphasize that, despite the MLP was trained with a similar structural system, it was able to predict the stress states of another structure, with different mesh refinements, and consequently, different stress states, as illustrated by the stress-strain curves.

Chapter 9

Conclusions

The main objective of this work was to develop and to demonstrate feasible and reliable the use of a NNCM for constitutive modeling of concrete. This initial study also sought to understand the main concepts, advantages, and limitations of using NNCM as a general model for representing concrete behavior, in order to pave the way for studies by the Intelligent Systems for Structural Engineering research group. ML techniques are considered the gold standard in function approximation and in the representation of complex phenomena such as material behavior. There are several well-established constitutive models, however, none capable of generally representing the complex behavior of quasi-brittle materials. For this purpose, ML techniques present themselves as promising alternatives. Thus, an ANN, of the type of a MLP, was trained with synthetic stress-strain data of the material under analysis.

First, the theoretical basis regarding the constitutive modeling of concrete was studied, focusing on the smeared crack model and its constitutive laws implemented in the INSANE system, in order to use it in the generation of synthetic data from numerical analyses via FEM. Such conventional constitutive model is known in the literature for representing with high fidelity the concrete behavior. As this is an initial study, it was also necessary to fix a single material to be applied in all analyses.

Next, the key concepts related to ML were investigated, focusing on supervised learning algorithms. Special attention was given to the ANNs technique, addressing the structure of the model, its main hyperparameters, and the operation of the error backpropagation algorithm. The consistent knowledge of all these concepts was fundamental for the sequence of this study. In this work, the state of the art of the use of ANNs in structural engineering and in the constitutive modeling of concrete was also addressed.

The structural systems numerically simulated for training were designed in an attempt to constitute a representative database, with structures characterized by the most varied types of loading and failure processes, and consequently, different stress-strain curves. Then, a study was conducted to optimize the network hyperparameters in order to obtain the model with the optimal capacity for the database in question. The choice of the

model, as well as all assumptions used to define the network architecture and conditions used in the training process, aimed to ensure that the model would not overfit the training data, the main fear of a ML model designer.

After training the ANN, explanability analyses were performed. ANNs are historically seen as "black boxes", and their use is sometimes overlooked in practical engineering decision making, due to a lack of clear understanding of how they work. The need to have models that make good predictions and are also consistent and reliable has motivated a series of studies and development of tools that seek to explain the decisions of ML algorithms. In this work, the SHAP package was used. The analyses performed here attested that the predictions of the proposed model are not random and that, for the cases analyzed, the variables that most impact the response variable are those that are most correlated. Such verification shows that the criticism towards the use of ML algorithms is biased and that current studies show the feasibility and physical consistency of using an NNCM.

The test structural systems, on the other hand, were chosen with the purpose of observing the behavior of the developed model for structural systems different from those used in the training, but subjected to similar internal stresses. The main evidence of generalization of the model was verified with the success in predicting the stresses of the system characterized by the mixed failure mode. The model was also able to predict, with satisfactory performance, the material behavior independently of the scale effect and the discretization level of the finite element mesh used to generate the test data. The difficulty of obtaining good answers for regions of structures submitted to bending with rupture by normal tensile stresses in the y-direction of the coordinate system pointed to the existence of few examples of this nature in the training database[1]. Here, one of the main advantages of neural nets reappears, which is the possibility of being able to extend its knowledge from the addition of new information to the training database.

In the end, it can be concluded that the NNCM was able to learn the behavior of the concrete stress-strain relationships of the training structural systems and generalize with reasonable accuracy this knowledge to the test structural systems, proving to be a viable and promising tool for the development of a generalist constitutive model for quasi-brittle media.

9.1 Future Research Topics

This dissertation will support future studies of the Intelligent Systems for Structural Engineering research group. Some possible research topics, which may continue this work, and that somehow were not addressed due to the time limit of this study are listed below:

[1]The training database developed in this work will be made available to the scientific community in the research group's online repository after its completion

- Analysis of different machine learning techniques in the constitutive representation of materials;
- Variation of the material and constitutive model in the generation of the training database, to develop a more generalist model;
- Performance comparison between NNCMs with variation in the number of historical features;
- Evaluation of the impact of partial and progressive training on model performance;
- Implementation of a NNCM in the INSANE system core for nonlinear analysis via FEM.

Bibliography

Arrea, M. (1981), 'Mixed-mode crack propagation in mortar and concrete', *Dept. of Struct. Engrg. Rep.* pp. 81–13.

Awad, M. and Khanna, R. (2015), *Machine Learning*, Apress, Berkeley, CA, pp. 1–18.

Bažant, Z. P. and Oh, B. H. (1983), 'Crack band theory for fracture of concrete', *Matériaux et construction* vol. 16(3), 155–177.

Belytschko, T. and Black, T. (1999), 'Elastic crack growth in finite elements with minimal remeshing', *International journal for numerical methods in engineering* vol. 45(5), 601–620.

Blaauwendraad, J. (1985), Realisations and restrictions: Application of numerical models to concrete structures, *in* 'Finite element analysis of reinforced concrete structures', ASCE, pp. 557–578.

Blaauwendraad, J. and Grootenboer, H. (1981), 'Essentials for discrete crack analysis', *IABSE Reports* vol. 34, 263–272.

Blumer, A., Ehrenfeucht, A., Haussler, D. and Warmuth, M. K. (1989), 'Learnability and the vapnik-chervonenkis dimension', *Journal of the ACM (JACM)* vol. 36(4), 929–965.

Bock, F. E., Aydin, R. C., Cyron, C. J., Huber, N., Kalidindi, S. R. and Klusemann, B. (2019), 'A review of the application of machine learning and data mining approaches in continuum materials mechanics', *Frontiers in Materials* p. 110.

Boone, T. and Ingraffea, A. (1987), Simulation of the fracture process flatrock interfaces, *in* 'Proceedings of the fourth international conference in Numerical Methods in fracture Mechanics', pp. 519–531.

Boone, T., Wawrzynek, P. and Ingraffea, A. (1986), Simulation of the fracture process in rock with application to hydrofracturing, *in* 'International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts', Vol. 23, Elsevier, pp. 255–265.

Brownlee, J. (2019), 'How to use data scaling improve deep learning model stability and performance', *Machine Learning Mastery: Vermont, Australia* .

Carreira, D. J. and Chu, K.-H. (1985), Stress-strain relationship for plain concrete in compression, *in* 'Journal Proceedings', Vol. 82, pp. 797–804.

Carreira, D. J. and Chu, K.-H. (1986), Stress-strain relationship for reinforced concrete in tension, *in* 'Journal Proceedings', Vol. 83, pp. 21–28.

Cervenka, V. (1970), *Inelastic finite element analysis of reinforced concrete panels under in-plane loads*, University of Colorado at Boulder.

Colasante, G. and Gosling, P. (2016), 'Including shear in a neural network constitutive model for architectural textiles', *Procedia engineering* vol. 155, 103–112.

de Andrade e Santos, A. H. (2015), Estudo Experimental do Efeito de Escala em Estruturas de Concreto Utilizando Correlação de Imagem Digital, PhD thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

de Borst, R. (2002), 'Fracture in quasi-brittle materials: a review of continuum damage-based approaches', *Engineering Fracture Mechanics* vol. 69(2), 95 – 112.
URL: *http://www.sciencedirect.com/science/article/pii/S0013794401000820*

de Borst, R. and Nauta, P. (1985), 'Non-orthogonal cracks in a smeared finite element model', *Engineering computations* .

Desai, C. (2020), 'Comparative analysis of optimizers in deep neural networks', *International Journal of Innovative Science and Research Technology* vol. 5(10), 959–962.

Duchi, J., Hazan, E. and Singer, Y. (2011), 'Adaptive subgradient methods for online learning and stochastic optimization.', *Journal of machine learning research* vol. 12(7).

Ellis, G., Yao, C., Zhao, R. and Penumadu, D. (1995), 'Stress-strain modeling of sands using artificial neural networks', *Journal of geotechnical engineering* vol. 121(5), 429–435.

Fischler, M. A. and Firschein, O. (1987), *Intelligence: The eye, the brain, and the computer*, Addison-Wesley Longman Publishing Co., Inc.

Fuina, J. S. (2004), Métodos de controle de deformações para análise não-linear de estruturas, Master's thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brasil.

Furukawa, T. and Yagawa, G. (1998), 'Implicit constitutive modelling for viscoplasticity using neural networks', *International Journal for Numerical Methods in Engineering* vol. 43(2), 195–219.

Ghaboussi, J. (2018), *Soft Computing in Engineering*, CRC Press, Taylor & Francis Group.
URL: *https://books.google.com.br/books?id=rVM3tAEACAAJ*

Ghaboussi, J., Garrett, J. and Wu, X. (1990), Material modeling with neural networks, *in* 'Proceedings of the international conference on numerical methods in engineering: theory and applications, Swansea, UK', pp. 701–717.

Ghaboussi, J., Garrett Jr, J. H. and Wu, X. (1991), 'Knowledge-based modeling of material behavior with neural networks', *Journal of Engineering Mechanics* vol. 117.

Ghaboussi, J., Pecknold, D. A., Zhang, M. and Haj-Ali, R. M. (1998), 'Autoprogressive training of neural network constitutive models', *International Journal for Numerical Methods in Engineering* vol. 42, 105–126.

Ghaboussi, J. and Sidarta, D. E. (1998), 'New nested adaptive neural networks (nann) for constitutive modeling', *Computers and Geotechnics* vol. 22, 29–52.

Goodfellow, I., Bengio, Y. and Courville, A. (2016), *Deep learning*, MIT press.

Gori, L. (2018), Failure analysis of quasi-brittle media using the micropolar continuum theory, elastic-degrading constitutive models, and smoothed point interpolation methods, Phd thesis, Universidade Federal de Minas Gerais (UFMG).

Griffith, A. A. (1920), 'Vi. the phenomena of rupture and flow in solids', *Philosophical transactions of the royal society of london. Series A, containing papers of a mathematical or physical character* vol. 221(582-593), 163–198.

Haj-Ali, R., Pecknold, D. A., Ghaboussi, J. and Voyiadjis, G. Z. (2001), 'Simulated micromechanical models using artificial neural networks', *Journal of Engineering Mechanics* vol. 127(7), 730–738.

Hakim, S. and Razak, H. A. (2013), 'Adaptive neuro fuzzy inference system (anfis) and artificial neural networks (anns) for structural damage identification', *Structural engineering and mechanics: An international journal* vol. 45(6), 779–802.

Hakim, S. and Razak, H. A. (2014), 'Modal parameters based structural damage detection using artificial neural networks-a review', *Smart Structures and Systems* vol. 14(2), 159–189.

Hashash, Y., Jung, S. and Ghaboussi, J. (2004), 'Numerical implementation of a neural network based material model in finite element analysis', vol. 59(7), 989–1005.

Haykin, S. (2009), *Neural networks and learning machines*, Pearson.

Hillerborg, A. (1985), 'The theoretical basis of a method to determine the fracture energyg f of concrete', *Materials and structures* vol. 18(4), 291–296.

Hillerborg, A., Modéer, M. and Petersson, P.-E. (1976), 'Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements', *Cement and Concrete Research* vol. 6(6), 773 – 781.
URL: *http://www.sciencedirect.com/science/article/pii/0008884676900077*

Hoerl, A. E. and Kennard, R. W. (1970), 'Ridge regression: Biased estimation for nonorthogonal problems', *Technometrics* vol. 12(1), 55–67.

Hornik, K., Stinchcomebe, M. and White, H. (1989), 'Multilayer feedforward networks are universal approximators', *Neural networks* vol. 2, 359–366.

Huang, D. Z., Xu, K., Farhat, C. and Darve, E. (2020), 'Learning constitutive relations from indirect observations using deep neural networks', *Journal of Computational Physics* vol. 416, 109491.

Im, S., Kim, H., Kim, W. and Cho, M. (2021), 'Neural network constitutive model for crystal structures', *Computational Mechanics* vol. 67(1), 185–206.

Ingraffea, A. R. and Saouma, V. (1985), Numerical modeling of discrete crack propagation in reinforced and plain concrete, *in* 'Fracture mechanics of concrete: Structural application and numerical calculation', Springer, pp. 171–225.

Javadi, A., Tan, T. and Zhang, M. (2003), 'Neural network for constitutive modelling in finite element analysis', *Computer Assisted Mechanics and Engineering Sciences* vol. 10(4), 523–530.

Ji, G., Li, F., Li, Q., Li, H. and Li, Z. (2011), 'A comparative study on arrhenius-type constitutive model and artificial neural network model to predict high-temperature deformation behaviour in aermet100 steel', *Materials Science and Engineering: A* vol. 528(13-14), 4774–4782.

Jung, S. and Ghaboussi, J. (2006), 'Neural network constitutive model for rate-dependent materials', *Computers & Structures* vol. 84(15-16), 955–963.

Kaplan, A. and Haenlein, M. (2019), 'Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence', *Business Horizons* vol. 62, 15–25.

Kingma, D. P. and Ba, J. L. (2015), Adam: A method for stochastic gradient descent, *in* 'ICLR: International Conference on Learning Representations', pp. 1–15.

Kochenderfer, M. J. and Wheeler, T. A. (2019), *Algorithms for optimization*, Mit Press.

Kortesis, S. and Panagiotopoulos, P. (1993), 'Neural networks for computing in structural analysis: methods and prospects of applications', *International Journal for Numerical Methods in Engineering* vol. 36(13), 2305–2318.

Krawczyk, B. (2016), 'Learning from imbalanced data: open challenges and future directions', *Progress in Artificial Intelligence* vol. 5(4), 221–232.

Kupfer, H., Hilsdorf, H. K. and Rusch, H. (1969), Behavior of concrete under biaxial stresses, *in* 'Journal proceedings', Vol. 66, pp. 656–666.

Lakshmanan, V., Robinson, S. and Munn, M. (2020), *Machine learning design patterns*, O'Reilly Media.

Le, B., Yvonnet, J. and He, Q.-C. (2015), 'Computational homogenization of nonlinear elastic materials using neural networks', *International Journal for Numerical Methods in Engineering* vol. 104(12), 1061–1084.

Leão, H. M. et al. (2021), 'A critical study on phase-field modelling of fracture'.

Lefik, M. and Schrefler, B. A. (2003), 'Artificial neural network as an incremental non-linear constitutive model for a finite element code', *Computer methods in applied mechanics and engineering* vol. 192(28-30), 3265–3283.

Liang, G. and Chandrashekhara, K. (2008), 'Neural network based constitutive model for elastomeric foams', *Engineering structures* vol. 30(7), 2002–2011.

Ling, J., Jones, R. and Templeton, J. (2016), 'Machine learning strategies for systems with invariance properties', *Journal of Computational Physics* vol. 318, 22–35.

Linka, K., Hillgärtner, M., Abdolazizi, K. P., Aydin, R. C., Itskov, M. and Cyron, C. J. (2021), 'Constitutive artificial neural networks: a fast and general approach to predictive data-driven constitutive modeling by deep learning', *Journal of Computational Physics* vol. 429, 110010.

Liu, M., Liang, L. and Sun, W. (2020), 'A generic physics-informed neural network-based constitutive model for soft biological tissues', *Computer methods in applied mechanics and engineering* vol. 372, 113402.

Liu, X., Tian, S., Tao, F. and Yu, W. (2021), 'A review of artificial neural networks in the constitutive modeling of composite materials', *Composites Part B: Engineering* vol. 224, 109152.

Liu, Z. and Wu, C. (2019), 'Exploring the 3d architectures of deep material network in data-driven multiscale mechanics', *Journal of the Mechanics and Physics of Solids* vol. 127, 20–46.

Liu, Z., Wu, C. and Koishi, M. (2019), 'A deep material network for multiscale topology learning and accelerated nonlinear modeling of heterogeneous materials', *Computer Methods in Applied Mechanics and Engineering* vol. 345, 1138–1168.

Logar, J. and Turk, G. (1997), 'Neural network as a constitutive model of soil', *Zeitschrift Fur Angewandte Mathematik Und Mechanik* vol. 77, S195–S196.

Lundberg, S. M. and Lee, S.-I. (2017), 'A unified approach to interpreting model predictions', *Advances in neural information processing systems* vol. 30.

Masi, F., Stefanou, I., Vannucci, P. and Maffi-Berthier, V. (2021), 'Thermodynamics-based artificial neural networks for constitutive modeling', *Journal of the Mechanics and Physics of Solids* vol. 147, 104277.

McCulloch, W. S. and Pitts, W. (1943), 'A logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics* vol. 5, 115–133.

Minsky, M. and Papert, S. (1969), 'Perceptrons cambridge', *MA: MIT Press. zbMATH* .

Mitchell, T. M. et al. (1997), 'Machine learning'.

Moës, N., Dolbow, J. and Belytschko, T. (1999), 'A finite element method for crack growth without remeshing', *International journal for numerical methods in engineering* vol. 46(1), 131–150.

Nelissen, L. (1972), 'Biaxial testing of normal concrete', *HERON, 18 (1), 1972* .

Ngo, D. and Scordelis, A. C. (1967), Finite element analysis of reinforced concrete beams, *in* 'Journal Proceedings', Vol. 64, pp. 152–163.

Penna, S. S. (2011), Formulação Multipotencial para Modelos de Degradação Elástica: Unificação Teórica, Proposta de Novo Modelo, Implementação Computacional e Modelagem de Estruturas de Concreto, phdthesis, Universidade Federal de Minas Gerais.

Penumadu, D. and Zhao, R. (1999), 'Triaxial compression behavior of sand and gravel using artificial neural networks (ann)', *Computers and Geotechnics* vol. 24(3), 207–230.

Petersson, P.-E. (1981), Crack growth and development of fracture zones in plain concrete and similar materials, Technical report, Lund Inst. of Tech.(Sweden). Div. of Building Materials.

Pitangueira, R. L. S. (1998), Mecânica de Estruturas de Concreto com Inclusão de Efeito de Tamanho e Heterogeneidade, PhD thesis, Pontifífica Universidade Católica do Rio de Janeiro.

Prechelt, L. (1998), Early stopping-but when?, *in* 'Neural Networks: Tricks of the trade', Springer, pp. 55–69.

Qian, N. (1999), 'On the momentum term in gradient descent learning algorithms', *Neural networks* vol. 12(1), 145–151.

Rabczuk, T. and Belytschko, T. (2004), 'Cracking particles: a simplified meshfree method for arbitrary evolving cracks', *International journal for numerical methods in engineering* vol. 61(13), 2316–2343.

Rabczuk, T. and Belytschko, T. (2007), 'A three-dimensional large deformation meshfree method for arbitrary evolving cracks', *Computer methods in applied mechanics and engineering* vol. 196(29-30), 2777–2799.

Ras, G., Xie, N., van Gerven, M. and Doran, D. (2022), 'Explainable deep learning: A field guide for the uninitiated', *Journal of Artificial Intelligence Research* vol. 73, 329–397.

Rashid, Y. R. (1968), 'Ultimate strength analysis of prestressed concrete pressure vessels', *Nuclear engineering and design* vol. 7(4), 334–344.

Ribeiro, M. T., Singh, S. and Guestrin, C. (2016), 'Model-agnostic interpretability of machine learning', *arXiv preprint arXiv:1606.05386* .

Robbins, H. and Monro, S. (1951), 'A stochastic approximation method', *The annals of mathematical statistics* pp. 400–407.

Rosenblatt, F. (1958), 'The perceptron: a probabilistic model for information storage and organization in the brain.', *Psychological review* vol. 65(6), 386.

Rots, J. G. and De Borst, R. (1987), 'Analysis of mixed-mode fracture in concrete', *Journal of engineering mechanics* vol. 113(11), 1739–1758.

Rots, J. G., Nauta, P., Kuster, G. and Blaauwendraad, J. (1985), 'Smeared crack approach and fracture localization in concrete', *HERON, 30 (1), 1985* .

Ruder, S. (2016), 'An overview of gradient descent optimization algorithms', *arXiv preprint arXiv:1609.04747* .

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), 'Learning representations by back-propagating errors', *nature* vol. 323(6088), 533–536.

Sage, A. P. (1990), *Concise Encyclopedia of Information Processing in Systems & [and] Organizations*, Pergamon Press New York, NY.

Sankarasubramanian, G. and Rajasekaran, S. (1996), 'Constitutive modeling of concrete using a new failure criterion', *Computers & structures* vol. 58(5), 1003–1014.

Sapra, R. (2014), 'Using r2 with caution', *Current Medicine Research and Practice* vol. 4(3), 130–134.

Sarle, W. (2002), 'Should i normalize/standardize/rescale the data'.

Seibi, A. and Al-Alawi, S. (1997), 'Prediction of fracture toughness using artificial neural networks (anns)', *Engineering Fracture Mechanics* vol. 56(3), 311–319.

Shahin, M. A., Jaksa, M. B. and Maier, H. R. (2008), 'State of the art of artificial neural networks in geotechnical engineering', *Electronic Journal of Geotechnical Engineering* vol. 8(1), 1–26.

Sidarta, D. and Ghaboussi, J. (1998), 'Constitutive modeling of geomaterials from non-uniform material tests', *Computers and Geotechnics* vol. 22(1), 53–71.

Smyrniou, E. (2018), 'Soil constitutive modelling using neural networks'.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), 'Dropout: a simple way to prevent neural networks from overfitting', *The journal of machine learning research* vol. 15(1), 1929–1958.

Stefanos, D. and Gyan, P. (2015), 'On neural network constitutive models for geomaterials', *Journal of Civil Engineering Research* vol. 5(5), 106–113.

Stoffel, M., Bamer, F. and Markert, B. (2019), 'Neural network based constitutive modeling of nonlinear viscoplastic structural response', *Mechanics Research Communications* vol. 95, 85–88.

Suidan, M. and Schnobrich, W. C. (1973), 'Finite element analysis of reinforced concrete', *Journal of the structural division* vol. 99(10), 2109–2122.

Sutton, R. (1986), Two problems with back propagation and other steepest descent learning procedures for networks, *in* 'Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986', pp. 823–832.

Theocaris, P. and Panagiotopoulos, P. (1993), 'Neural networks for computing in fracture mechanics. methods and prospects of applications', *Computer Methods in Applied Mechanics and Engineering* vol. 106(1-2), 213–228.

Tibshirani, R. (1996), 'Regression shrinkage and selection via the lasso', *Journal of the Royal Statistical Society: Series B (Methodological)* vol. 58(1), 267–288.

Unger, J. F. and Könke, C. (2009), 'Neural networks as material models within a multi-scale approach', *Computers & structures* vol. 87(19-20), 1177–1186.

Valliappan, S. and Doolan, T. F. (1972), 'Nonlinear stress analysis of reinforced concrete', *Journal of the Structural Division* vol. 98(4), 885–898.

Vapnik, V. (1982), *Estimation of Dependences Based on Empirical Data*, Springer-Verlag.

Vapnik, V. (1995), *The Nature of Statistical Learning Theory*, Springer, New York.

Vapnik, V. N. and Chervonenkis, A. Y. (1971), 'On the uniform convergence of relative frequencies of events to their probabilities', *Theory of Probability and its Applications* vol. 16(2), 264–280.

Wawrzynek, P., Boone, T. and Ingraffea, A. (1987), Efficient techniques for modeling the fracture process zone in rock and concrete, *in* 'Proc. of the Fourth lnternational Conference on Numerical Methods in Fracture Mechanics', pp. 23–27.

Widrow, B. (1962), 'Generalization and information storage in network of ada-line'neurons", *Self-organizing systems-1962* pp. 435–462.

Widrow, B. and Hoff, M. E. (1960), Adaptive switching circuits, Technical report, Stanford Univ Ca Stanford Electronics Labs.

Winkler, B., Hofstetter, G. and Lehar, H. (2004), 'Application of a constitutive model for concrete to the analysis of a precast segmental tunnel lining', *International Journal for Numerical and Analytical Methods in Geomechanics* vol. 28, 797–819.

Wu, H. (1997), 'Constitutive relation modelling for soil using finite element±neural network hybrid algorithms', *Computer methods and advances in geomechanics. Rotterdam: Balkema* pp. 613–7.

Wu, J.-Y., Nguyen, V. P., Nguyen, C. T., Sutula, D., Sinaie, S. and Bordas, S. P. (2020), 'Phase-field modeling of fracture', *Advances in applied mechanics* vol. 53, 1–183.

Wu, X. (1991), Neural network-based material modeling, PhD thesis, University of Illinois at Urbana-Champaign.

Xu, K., Huang, D. Z. and Darve, E. (2021), 'Learning constitutive relations using symmetric positive definite neural networks', *Journal of Computational Physics* vol. 428, 110072.

Yang, Y. B. and Shieh, M. S. (1990), 'Solution method for nonlinear problems with multiple critical points', *AIAA Journal* vol. 28(12), 2110–2116.

Yeh, I.-C. (1998), 'Modeling of strength of high-performance concrete using artificial neural networks', *Cement and Concrete research* vol. 28(12), 1797–1808.

Zhou, X.-H., Han, J. and Xiao, H. (2021), 'Learning nonlocal constitutive models with neural networks', *Computer Methods in Applied Mechanics and Engineering* vol. 384, 113927.

Appendices

Appendix A

Nonlinear Finite Element Analysis

The nonlinear analysis via finite element method is an important computational method that represent its results in the form of *equilibrium paths* for specific degrees of freedom of the problem through an incremental-iterative process.

Figure A.1 illustrates common equilibrium paths in nonlinear problems.

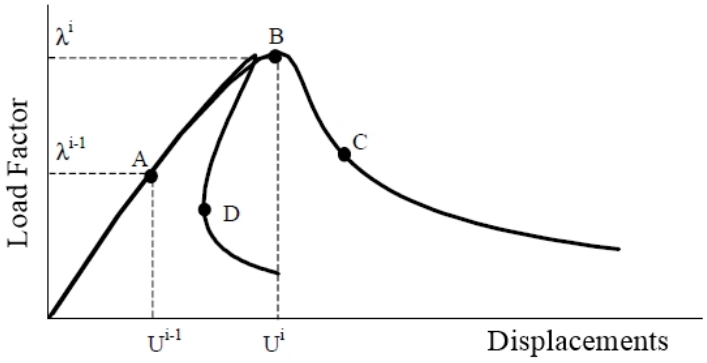


Figure A.1: Typical equilibrium paths in nonlinear problems by Fuina (2004)

It is observed that there are critical points that change the curve's trajectory. The point B is known as limit load point or equilibrium's bifurcation, and the point D is known as limit displacement point or snap-back point. The equilibrium paths shown in Fig. A.1 are typical of quasi-brittle materials, presenting a softening behavior.

Different incremental-iterative methods have been used for structures' nonlinear analysis, among them the methods of load control, direct displacement control, arc length control and generalized displacement control (Fuina, 2004, Pitangueira, 1998). Moreover, the formulation used to obtain the equilibrium paths is the tangent formulation due to its rapid convergence and for better describing the trajectory. This process uses the Newton-Raphson Method to solve a linear system in each iteration, in which the tangent stiffness matrix is updated.

A general formulation of incremental-iterative methods can be written in order to solve the nonlinear system of $N + 1$ unknowns and $N + 1$ equations. The unknowns are

N incremental displacements and an increment in the load factor, and the equations are N equilibrium equations and a constraint equation.

Therefore, for an equilibrium point A, it is desired to find another equilibrium point B based on a displacement field $\{U\}$ and a load factor λ, as shown in Fig. A.1.

Generically, the equation of iteration j from step i can be written as

$$[K_t]^i_{j-1} \cdot \{\delta U\}^i_j = \delta\lambda^i_j \cdot \{P\} + \{Q\}^i_{j-1} \tag{A.1}$$

where

$[K_t]^i_{j-1}$ is the tangent stiffness matrix in the iteration $j-1$ from step i, function of the displacement field $\{U\}^i_{j-1}$;

$\{\delta U\}^i_j$ is the incremental displacements vector in the iteration j from step i;

$\delta\lambda^i_j$ is the load factor increment in the iteration j from step i;

$\{P\}$ is the reference load vector and

$\{Q\}^i_j$ is the residual forces vector in the iteration j from step i.

First, the control method is defined (Detailed information about the control methods may be found on the literature, such as in Fuina (2004) and therefore, the value of the increase in the load factor $\delta\lambda^i_j$. Next, $\{\delta U\}_j$ is calculated by the sum of the portion referring to the reference load $\{\delta U\}^P_j$ and the residual force $\{\delta U\}^Q_j$,

$$\{\delta U\}_j = \delta\lambda_j \cdot \{\delta U\}^P_j + \{\delta U\}^Q_j \tag{A.2}$$

with

$$[K_t]_{j-1} \cdot \{\delta U\}^P_j = \{P\} \tag{A.3}$$

and

$$[K_t]_{j-1} \cdot \{\delta U\}^Q_j = \{Q\}_{j-1} \tag{A.4}$$

Then, there is the update of the problem variables, as the load factor

$$\lambda_j = \lambda_{j-1} + \delta\lambda_j \tag{A.5}$$

the displacement vector

$$\{U\}_j = \{U\}_{j-1} + \{\delta U\}_j \tag{A.6}$$

and the residual force vector

$$\{Q\}_j = \lambda_j \cdot \{P\} - \{F\}_j, \tag{A.7}$$

all of them from iteration j, where $\{F\}_j$ is the internal forces vector or equivalent forces

vector from the internal stresses.

The internal forces vector is given by

$$\{F\}_j = \int_V [B^T] \cdot \{\sigma\}_j \cdot dV \tag{A.8}$$

where the matrix $[B]$ is described as

$$[B] = [L] \cdot [N] \tag{A.9}$$

with $[L]$ being the partial derivative matrix and $[N]$ the approximation functions matrix, both defined by the finite element method; and $\{\sigma\}$ is the stress state at a particular point, calculated according to equation above

$$\{\sigma\}_j = [C]_j \cdot \{\varepsilon\}_j \tag{A.10}$$

The convergence of the process is verified at the end of each iteration. The convergence analysis can be evaluated either by residual forces criterion or by displacements criterion. For the former, the equation is given by

$$\frac{\|\{Q\}\|}{\lambda\{P\}} < tolerance \tag{A.11}$$

For the latter, by

$$\frac{\|\{\delta U\}\|}{\lambda\{U\}} < tolerance \tag{A.12}$$

It is important to notice that at the first iteration of each step, the residual forces vector $\{Q\}_{j-1}$ is null.

If the convergence criterion is not achieved in the current iteration, the tangent stiffness matrix is calculated according to finite element formulation, as follows

$$[K_t]_j = \int_V [B^T] \cdot [C_t]_j \cdot [B] \cdot dV \tag{A.13}$$

where $[C_t]_j$ is the the constitutive matrix, given by the relation

$$\{d\sigma\}_j = [C_t]_j \cdot \{d\varepsilon\}_j \tag{A.14}$$

In short, the incremental-iterative method is a predictor-corrector process, where an incremental displacement vector is predicted due to the definition of the increase of the load factor through a constraint equation and then, the internal forces vector is calculated and its convergence is verified.

Yang and Shieh (1990) proposed a generic algorithm that shows the flow of the incremental-iterative process. Fig. A.2 presents this solution procedure, also adopted

in INSANE.

```
1  begin
2      Assemble the reference load vector {P};
3      foreach iteration j and step i do
4          repeat
5              Mount the global tangent stiffness matrix [K_t]^i_{j-1};
6              Compute the incremental displacement vector
               {δU^P}^i_j and {δU^Q}^i_j;
7              Compute the load factor increment δλ^i_j;
8              Update the displacement vector
               {U}^i_j = {U}^i_{j-1} + δλ^i_j {δU^P}^i_j + {δU^Q}^i_j;
9              Update the load factor λ^i_j = λ^i_{j-1} + δλ^i_j;
10             Mount the internal forces vector {F}^i_j;
11             Update the residual forces vector
               {Q}^i_j = δλ^i_j {P} − {F}^i_j;
12         until convergence
13     end
14 end
```

Figure A.2: Solution Procedure Yang and Shieh (1990)

Appendix B

Neural Network-based Constitutive Matrix

Ghaboussi (2018) presented two different manners to get the tangent stiffness matrix, either by probing the neural network constitutive model, or directly computing it from the connection weights of the neural network. In this present work it will be demonstrated the former.

In this way, in a case of a two-dimensional plane strain or plane stress problem, independent of the number of information in the input data, it is necessary three forward pass (or probes) through the neural network model in order to assemble the columns of the respective tangent stress-strain matrix $[C_t]$. The following equation describes this three process along the strain paths for which each one of them generate a stress vector C_j, i.e., a column of the tangent stress-strain matrix.

$$\begin{cases} C_{ij} = \dfrac{1}{\alpha} \cdot C_j \mathbf{NN}(\{\Delta\varepsilon_i\}_j, \text{historical points} : \ldots) \; j = 1, 2, 3 \\ \Delta\varepsilon_{ij} = \alpha\delta_{ij} \end{cases} \tag{B.1}$$

Eq. (B.1) says that for each strain increment vector from an integration point, $\alpha = \Delta\varepsilon_i$, where δ_{ij} is the Kronecker delta . This lead to the following strain paths in the three forward passes : $[\alpha, 0, 0]^T$, $[0, \alpha, 0]^T$ and $[0, 0, \alpha]^T$. In this particular case, the C_t it would be given by

$$C_t = \begin{bmatrix} \dfrac{\Delta\sigma_1}{\Delta\varepsilon_1} & \dfrac{\Delta\sigma_1}{\Delta\varepsilon_2} & \dfrac{\Delta\sigma_1}{\Delta\varepsilon_3} \\ \dfrac{\Delta\sigma_2}{\Delta\varepsilon_1} & \dfrac{\Delta\sigma_2}{\Delta\varepsilon_2} & \dfrac{\Delta\sigma_2}{\Delta\varepsilon_3} \\ \dfrac{\Delta\sigma_3}{\Delta\varepsilon_1} & \dfrac{\Delta\sigma_3}{\Delta\varepsilon_2} & \dfrac{\Delta\sigma_3}{\Delta\varepsilon_3} \end{bmatrix} \tag{B.2}$$

An attention must be given on the fact that this method does not assure that the matrix $[C_t]$ will be symmetric. To get around this problem, it is possible to impose this condition by doing $[C_t] = \frac{1}{2}([C_t] + [C_t]^T)$. However, the use of the constitutive matrix

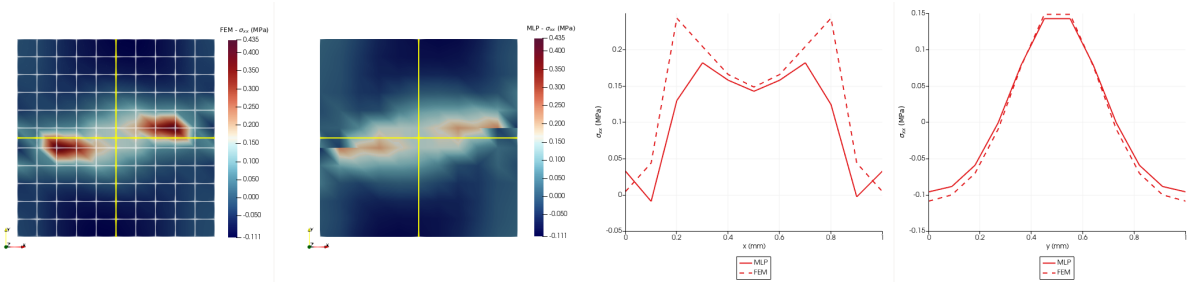
obtained using neural networks in a finite element code, as well as its symmetrization, will be part of the scope of future work of the research group.

Appendix C

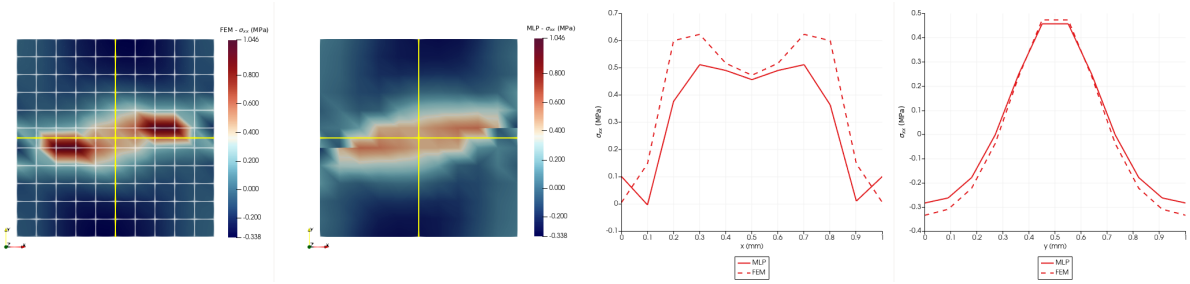
Complementary Results

Chapter 8 discussed the results obtained from the MLP validation analysis in terms of the main stress components of each problem, i.e., those responsible for failure. In this appendix, the analyses of the remaining stress components are presented. First, the comparison plots of the stresses obtained via FEM and those predicted by MLP. Next, the graphs for the analysis of the accumulated MAE.

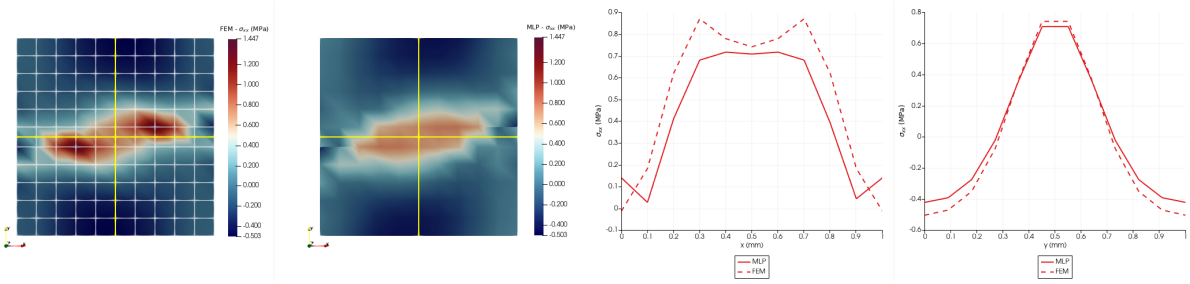
C.1 Traction - Asymmetric Traction Test



(a)

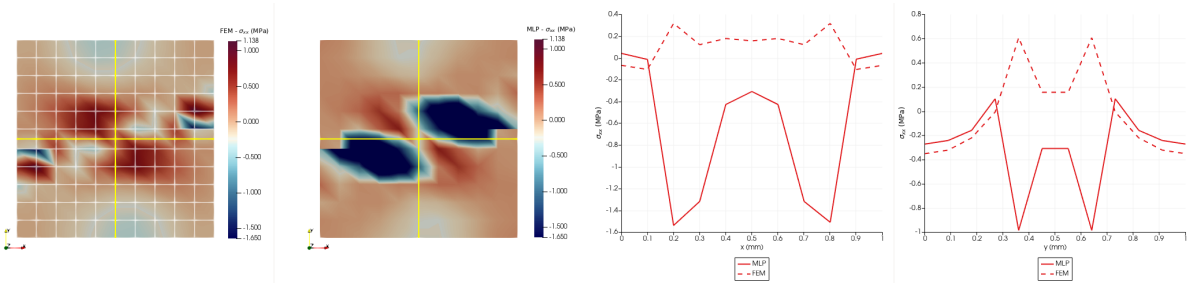


(b)

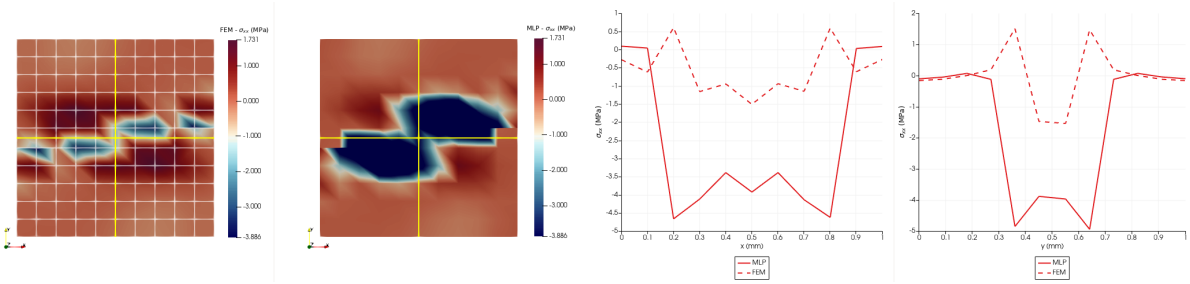


(c)

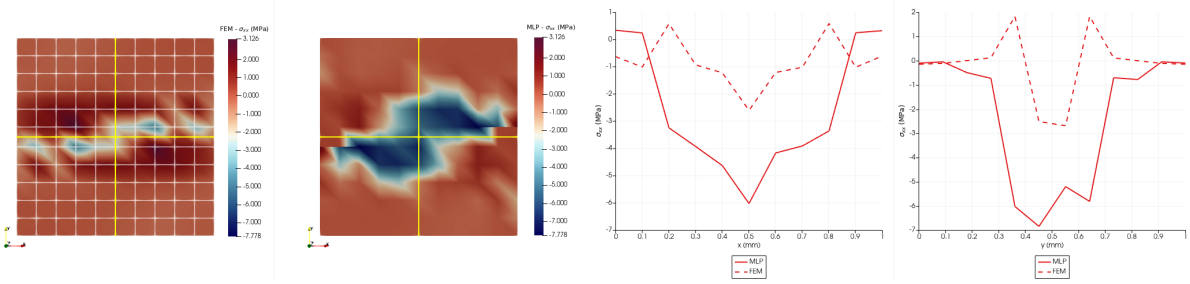
Figure C.1: Asymmetric traction test - Stresses FEM × MLP - σ_{xx}. (a) Point A Step 5, (b) Point B Step 15, (c) Point C Step 22.



(a)

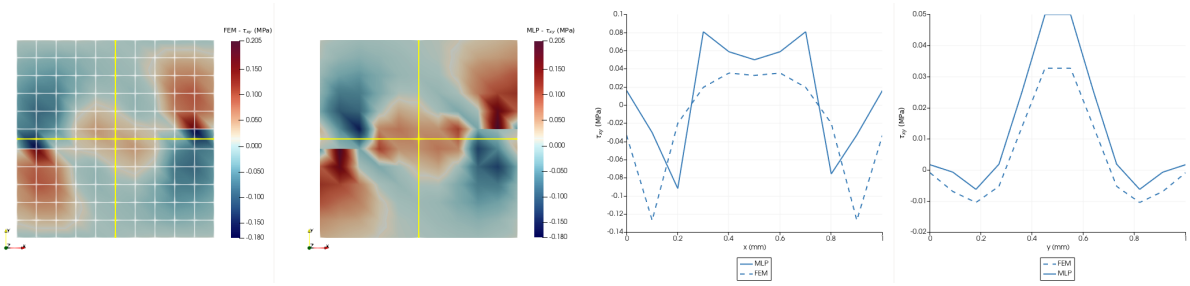


(b)

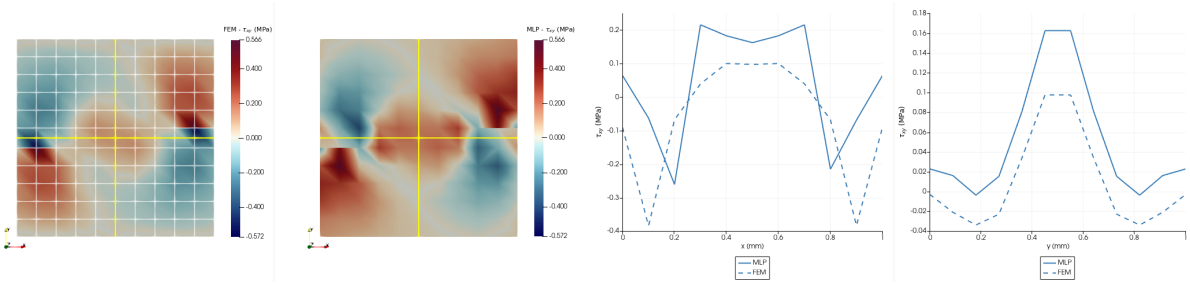


(c)

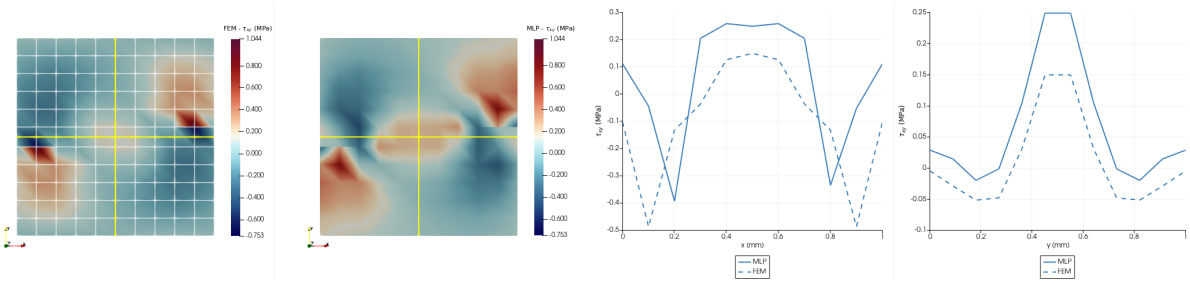
Figure C.2: Asymmetric traction test - Stresses FEM × MLP - σ_{xx}. (a) Point D Step 30, (b) Point E Step 100, (c) Point F Step 198.



(a)



(b)



(c)

Figure C.3: Asymmetric traction test - Stresses FEM × MLP - τ_{xy}. (a) Point A Step 5, (b) Point B Step 15, (c) Point C Step 22.

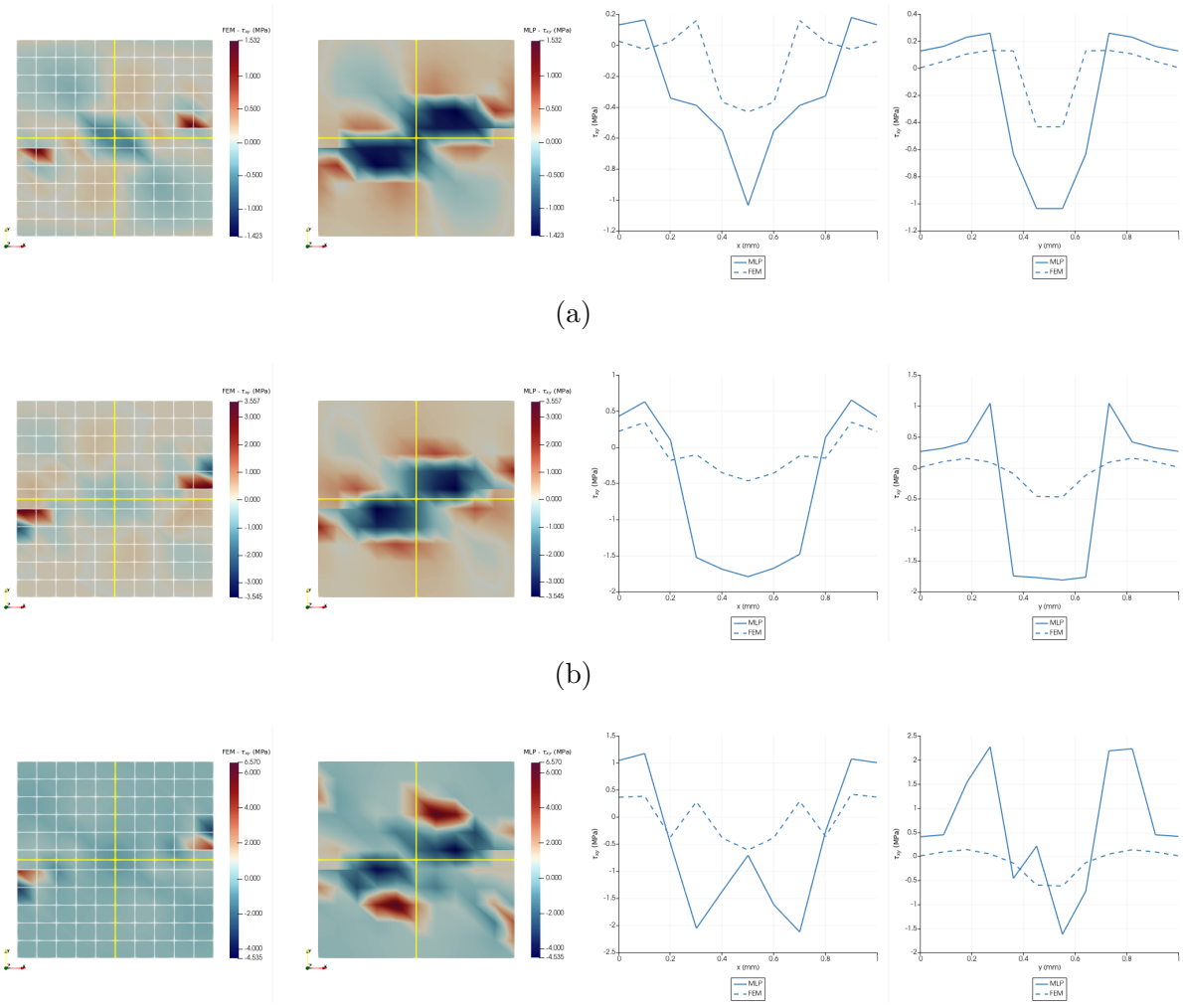


Figure C.4: Asymmetric traction test - Stresses FEM × MLP - τ_{xy}. (a) Point D Step 30, (b) Point E Step 100, (c) Point F Step 198.

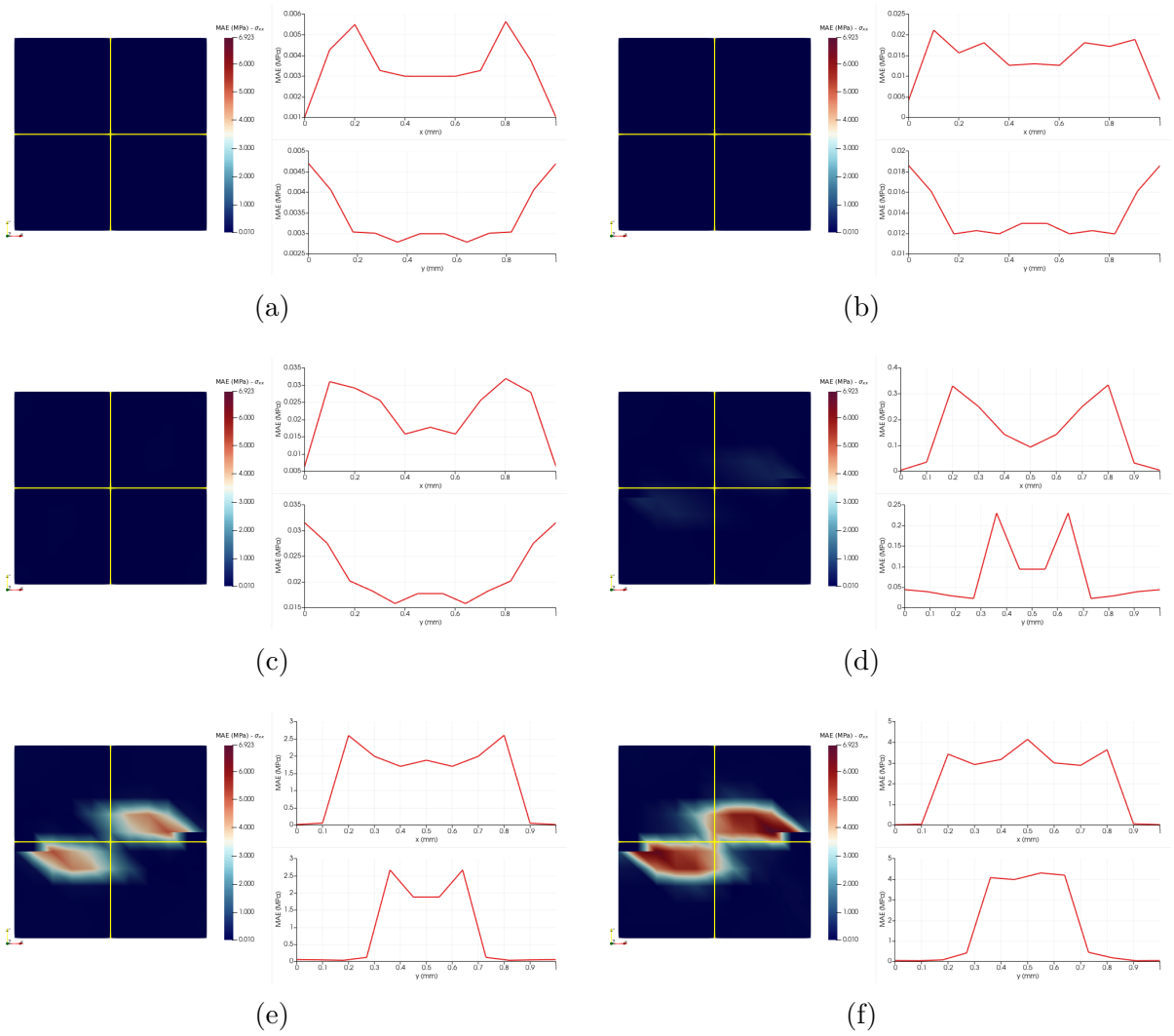


Figure C.5: Asymmetric traction test - MAE accumulated - σ_{xx}. (a) Point A Step 5, (b) Point B Step 15, (c) Point C Step 22, (d) Point D Step 30, (e) Point E Step 100, (f) Point F Step 198.

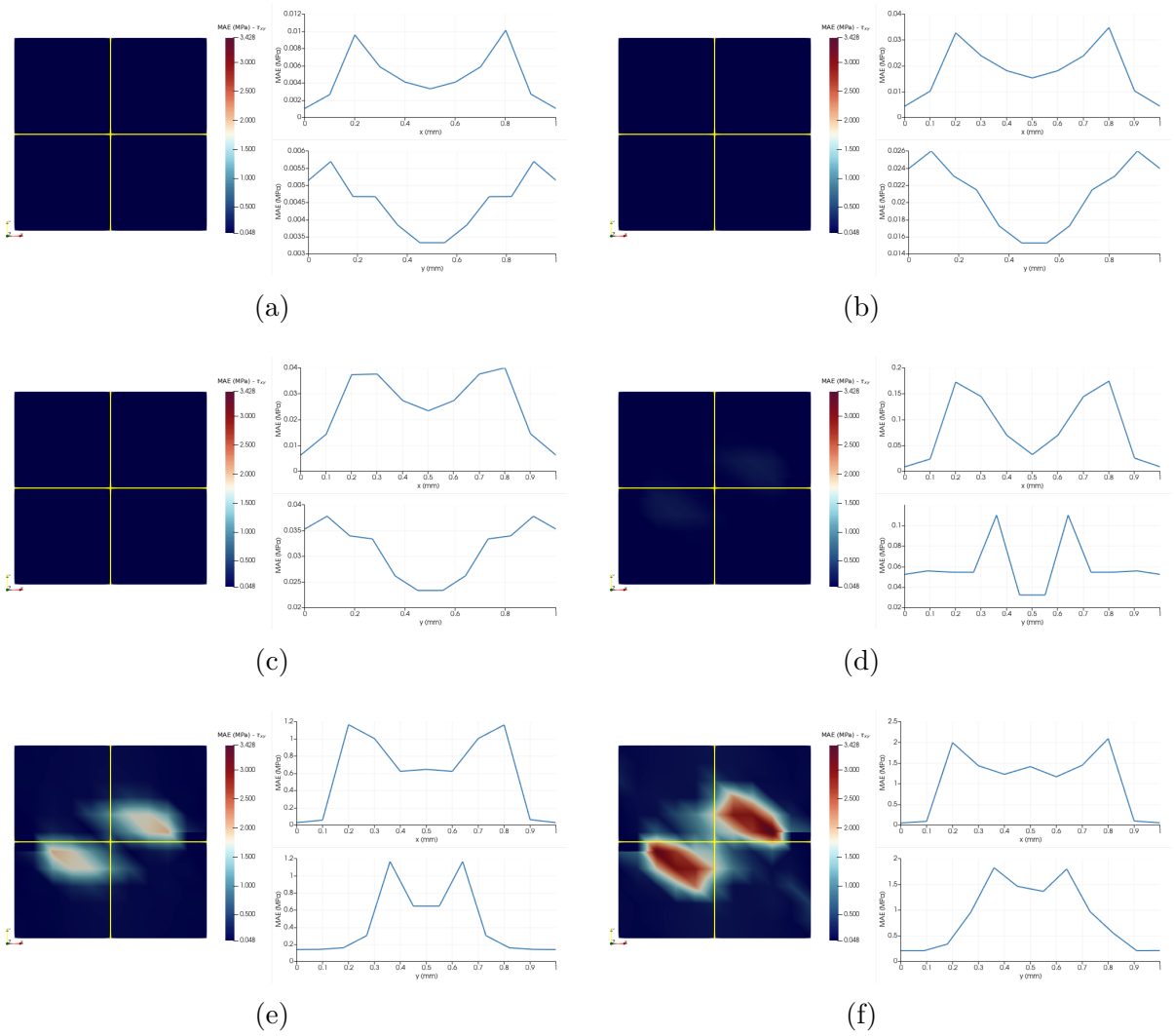
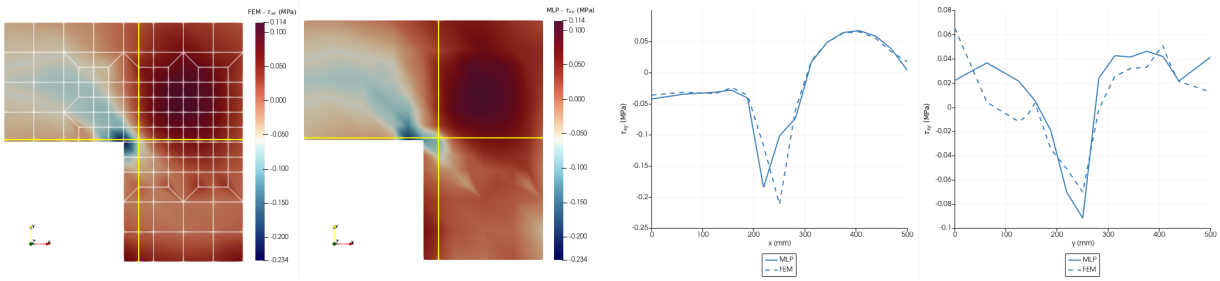
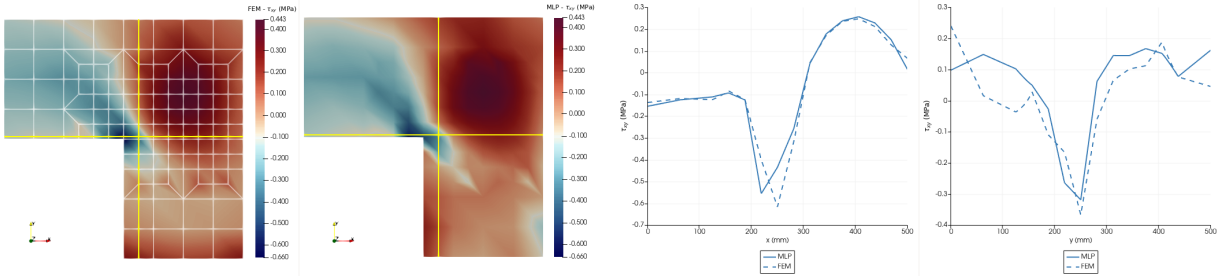


Figure C.6: Asymmetric traction test - MAE accumulated - τ_{xy}. (a) Point A Step 5, (b) Point B Step 15, (c) Point C Step 22, (d) Point D Step 30, (e) Point E Step 100, (f) Point F Step 198.

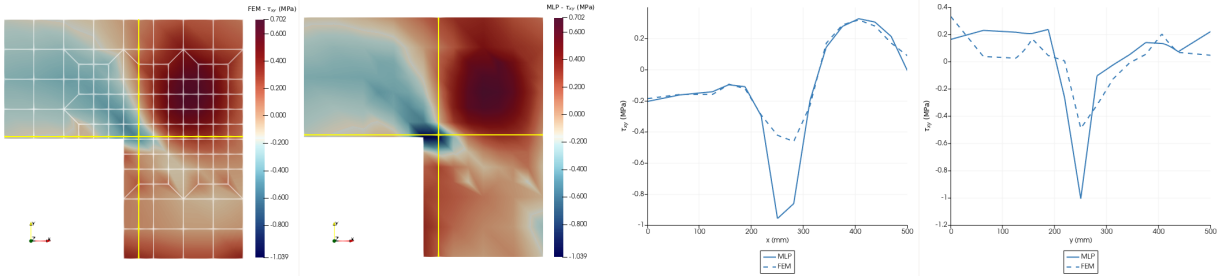
C.2 Bending - L-Shaped Panel



(a)

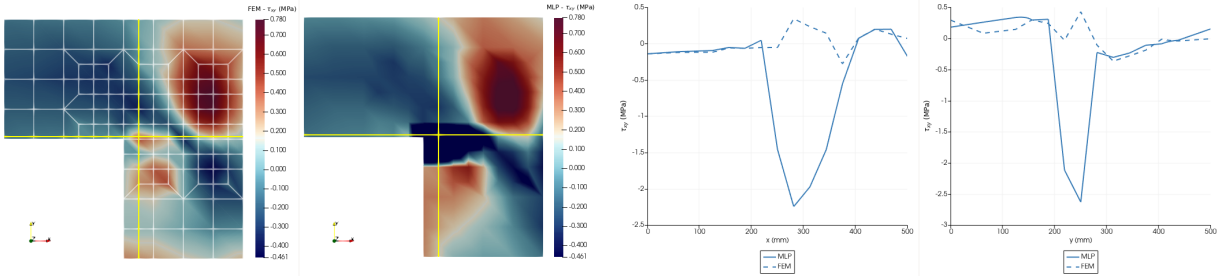


(b)

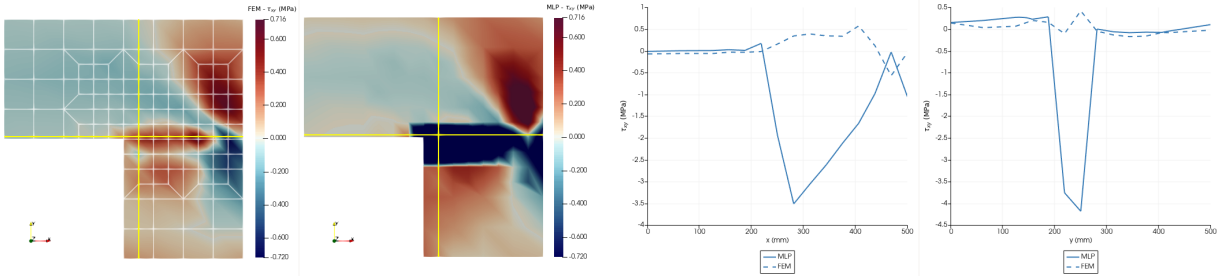


(c)

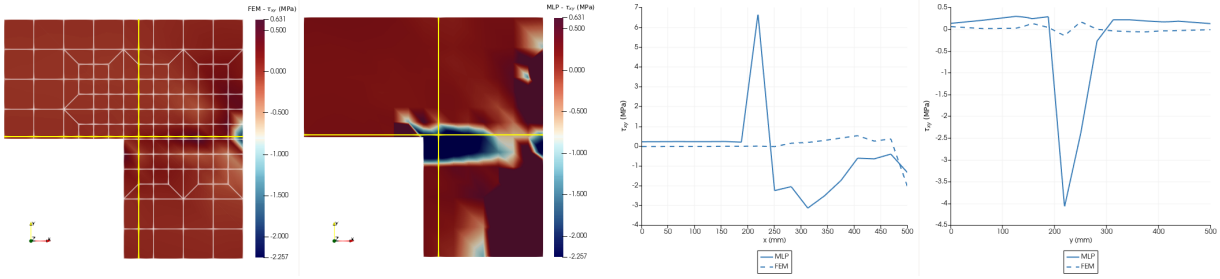
Figure C.7: L-shaped panel - Stresses FEM × MLP - τ_{xy}. (a) Point A Step 5, (b) Point B Step 20, (c) Point C Step 33.



(a)



(b)



(c)

Figure C.8: L-shaped panel - Stresses FEM × MLP - τ_{xy}. (a) Point D Step 50, (b) Point E Step 100, (c) Point F Step 250.

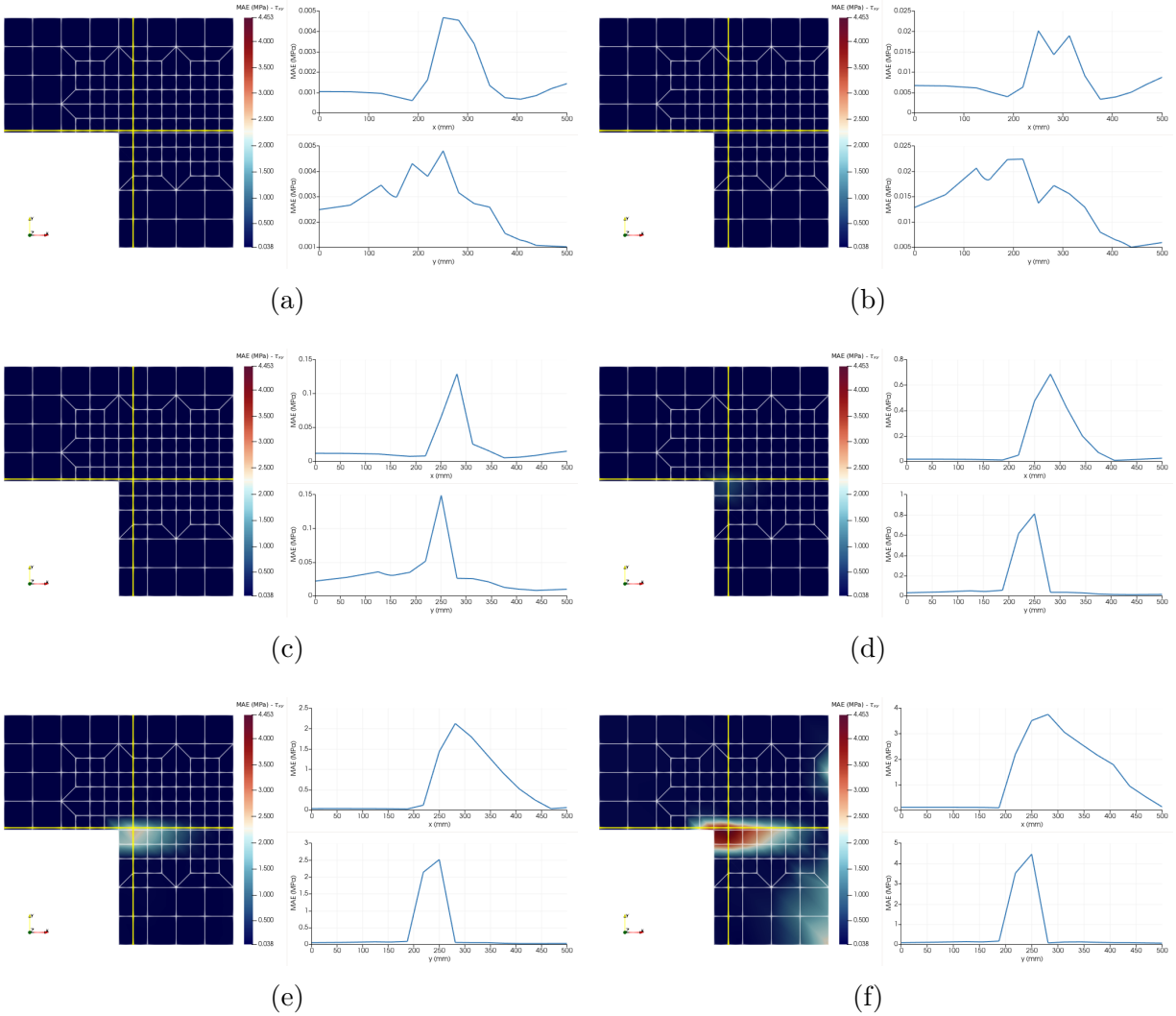


Figure C.9: L-shaped panel - MAE accumulated - τ_{xy}. (a) Point D Step 50, (b) Point E Step 100, (c) Point F Step 250.

C.3 Size Effect Analysis - Brazilian Splitting Test

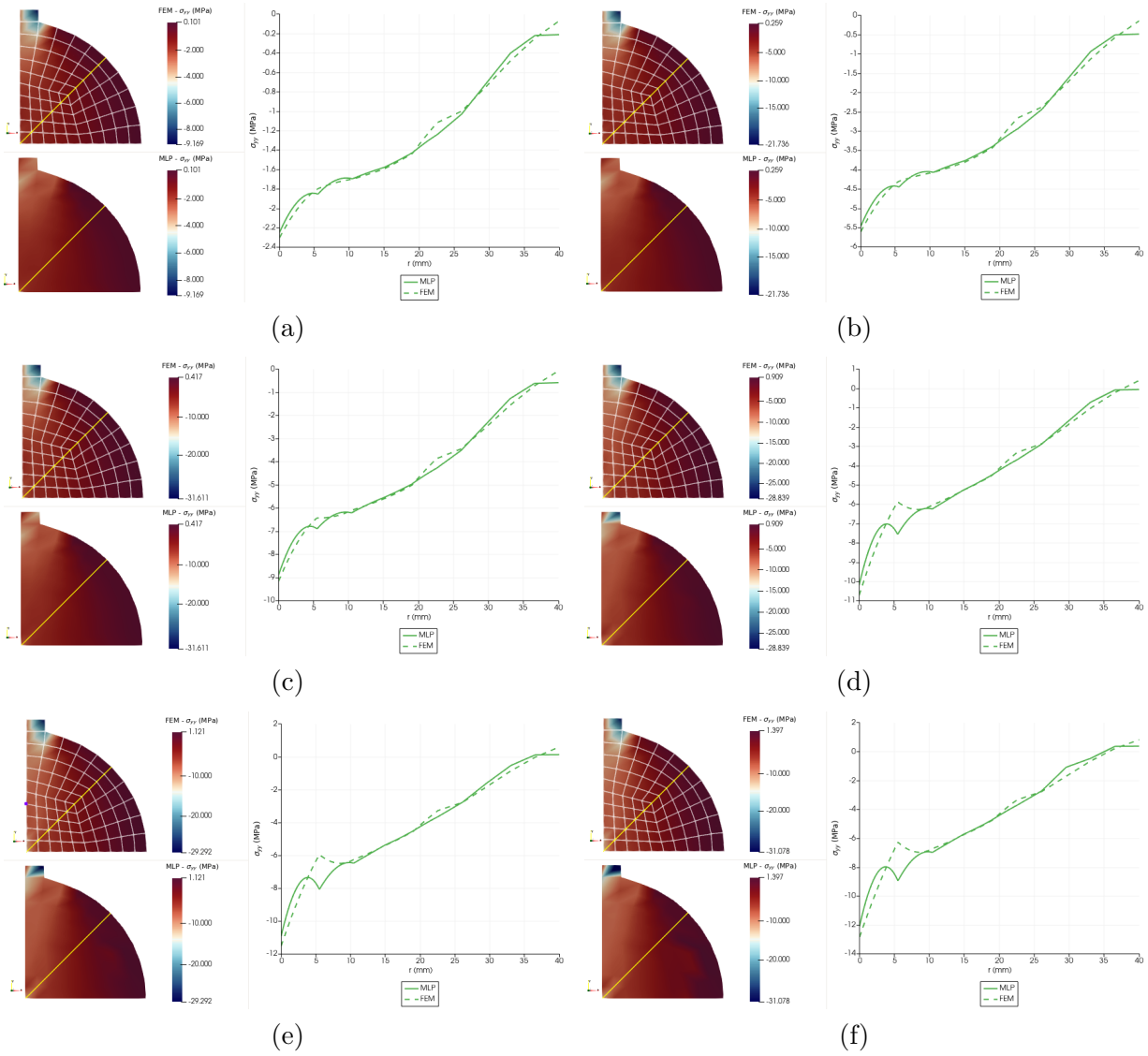


Figure C.10: Diametrical compression with crack size of 8mm - Stresses FEM × MLP - σ_{yy}. (a) Point A1 Step 10, (b) Point B1 Step 25, (c) Point C1 Step 46, (d) Point D1 Step 75, (e) Point E1 Step 90, (f) Point F1 Step 114.

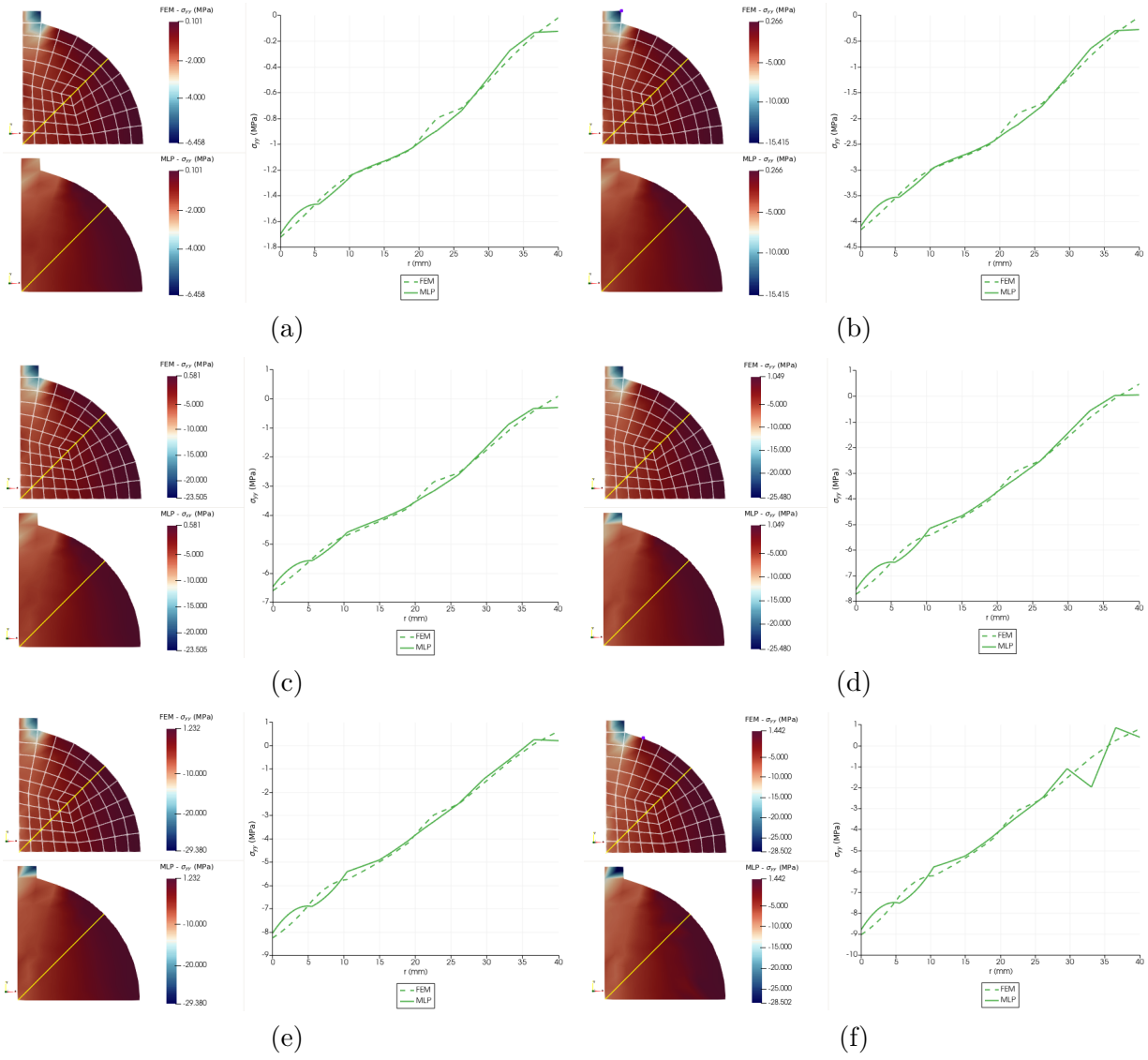


Figure C.11: Diametrical compression with crack size of 16mm - Stresses FEM × MLP - σ_{yy}. (a) Point A2 Step 10, (b) Point B2 Step 25, (c) Point C2 Step 46, (d) Point D2 Step 75, (e) Point E2 Step 90, (f) Point F2 Step 110.

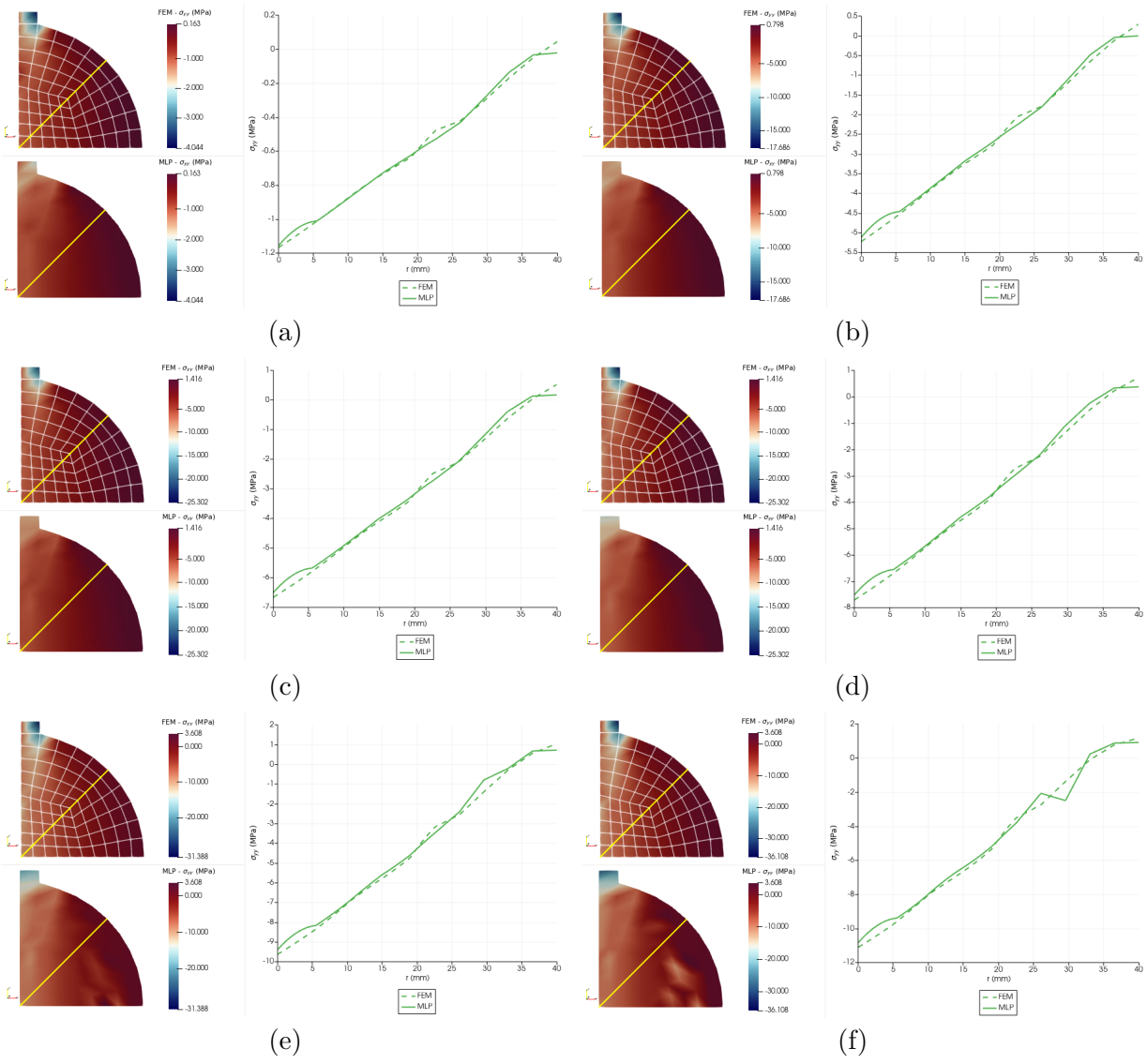


Figure C.12: Diametrical compression with crack size of 24mm - Stresses FEM × MLP - σ_{yy}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

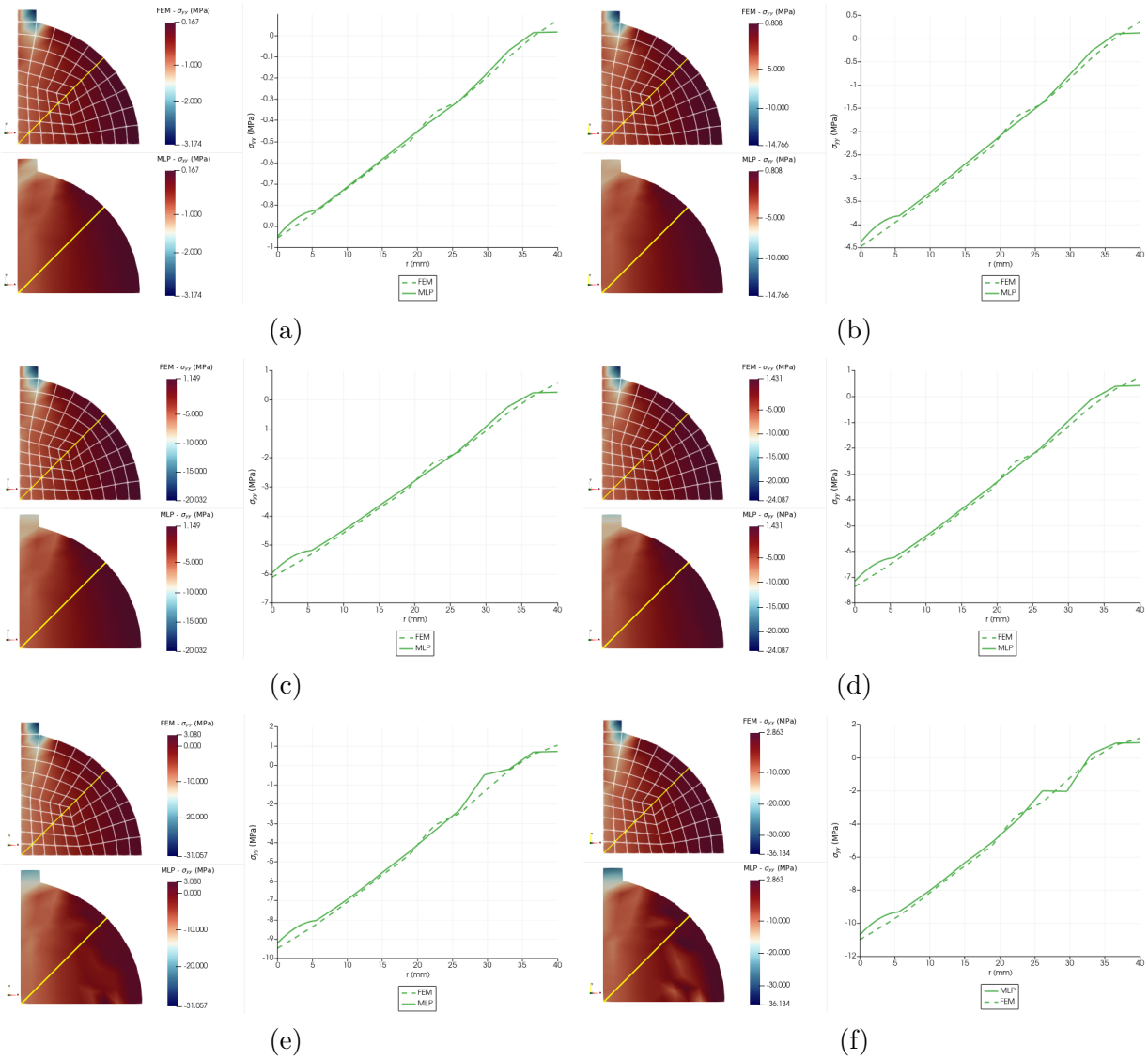


Figure C.13: Diametrical compression with crack size of 28mm - Stresses FEM × MLP - σ_{yy}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

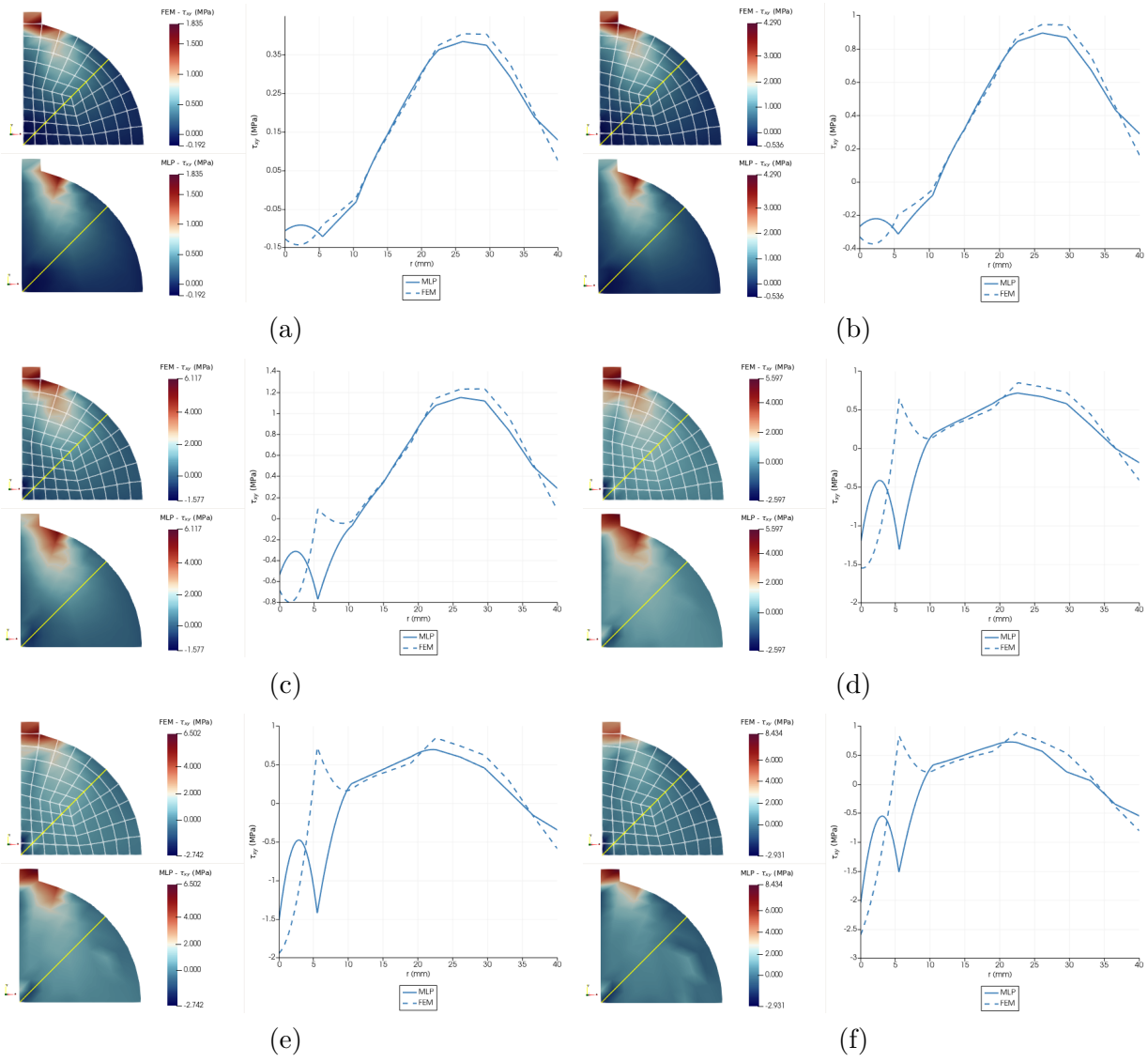


Figure C.14: Diametrical compression with crack size of 8mm - Stresses FEM × MLP - τ_{xy}. (a) Point A1 Step 10, (b) Point B1 Step 25, (c) Point C1 Step 46, (d) Point D1 Step 75, (e) Point E1 Step 90, (f) Point F1 Step 114.

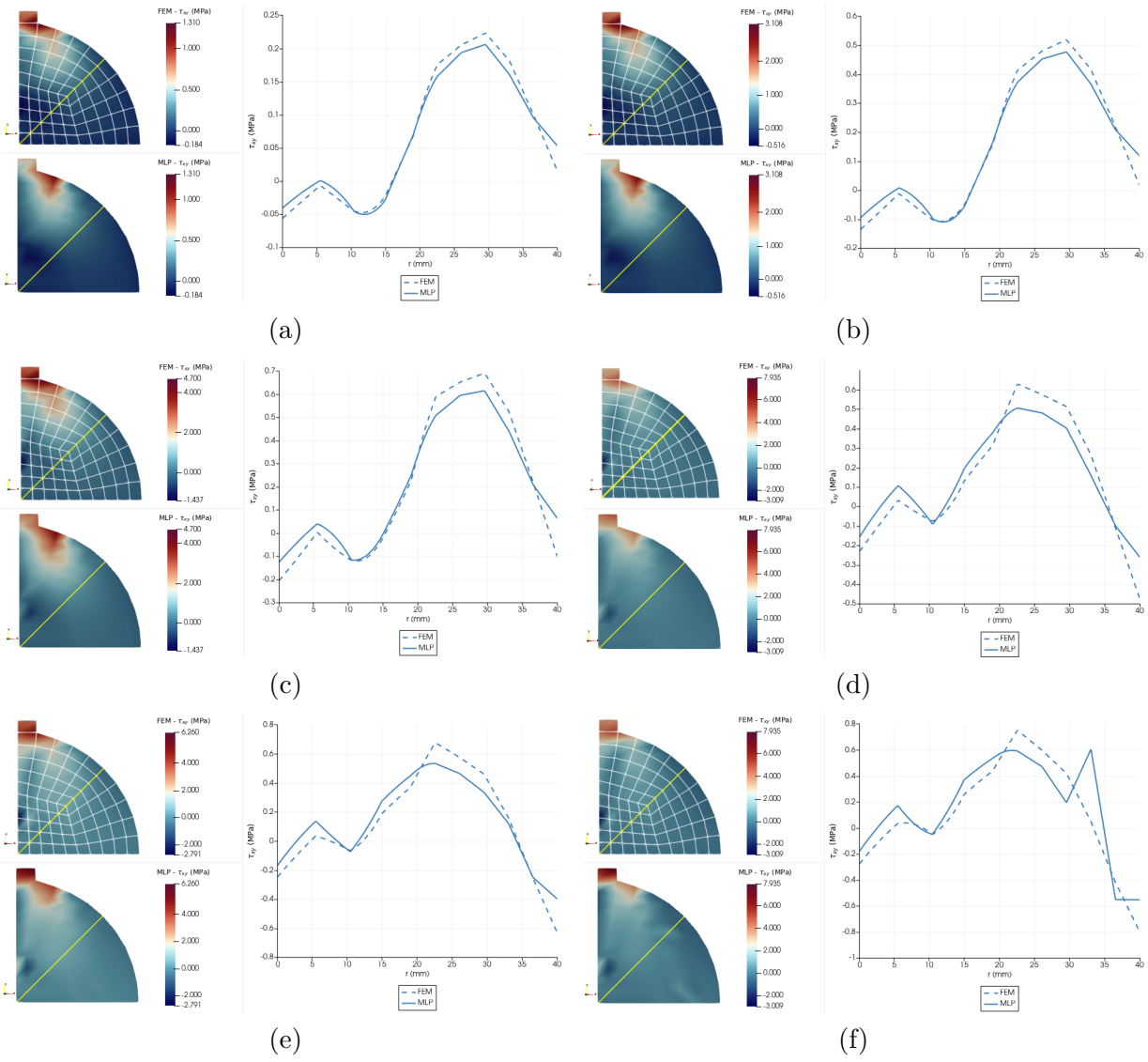


Figure C.15: Diametrical compression with crack size of 16mm - Stresses FEM × MLP - τ_{xy}. (a) Point A2 Step 10, (b) Point B2 Step 25, (c) Point C2 Step 46, (d) Point D2 Step 75, (e) Point E2 Step 90, (f) Point F2 Step 110.

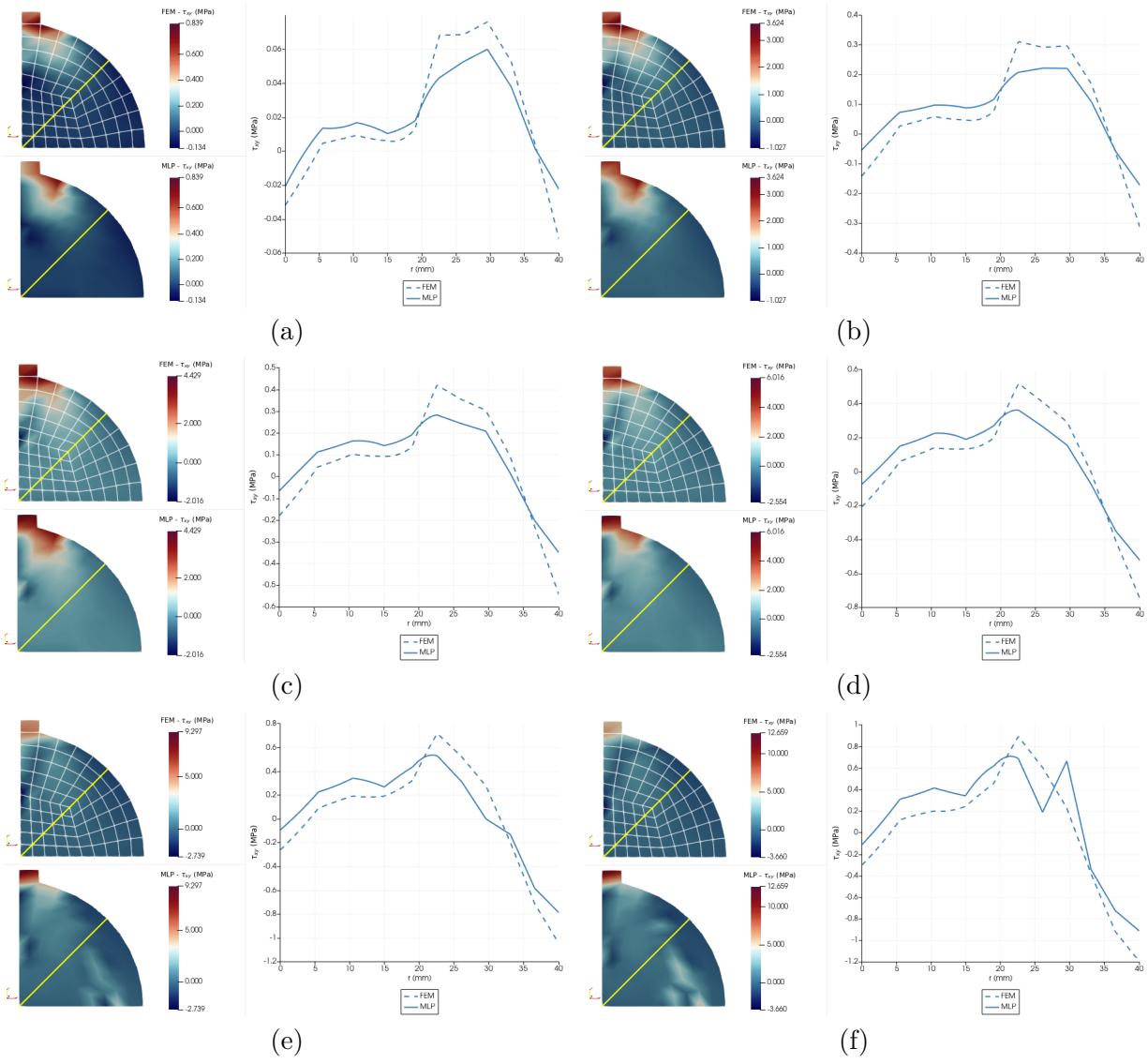


Figure C.16: Diametrical compression with crack size of 24mm - Stresses FEM × MLP - τ_{xy}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

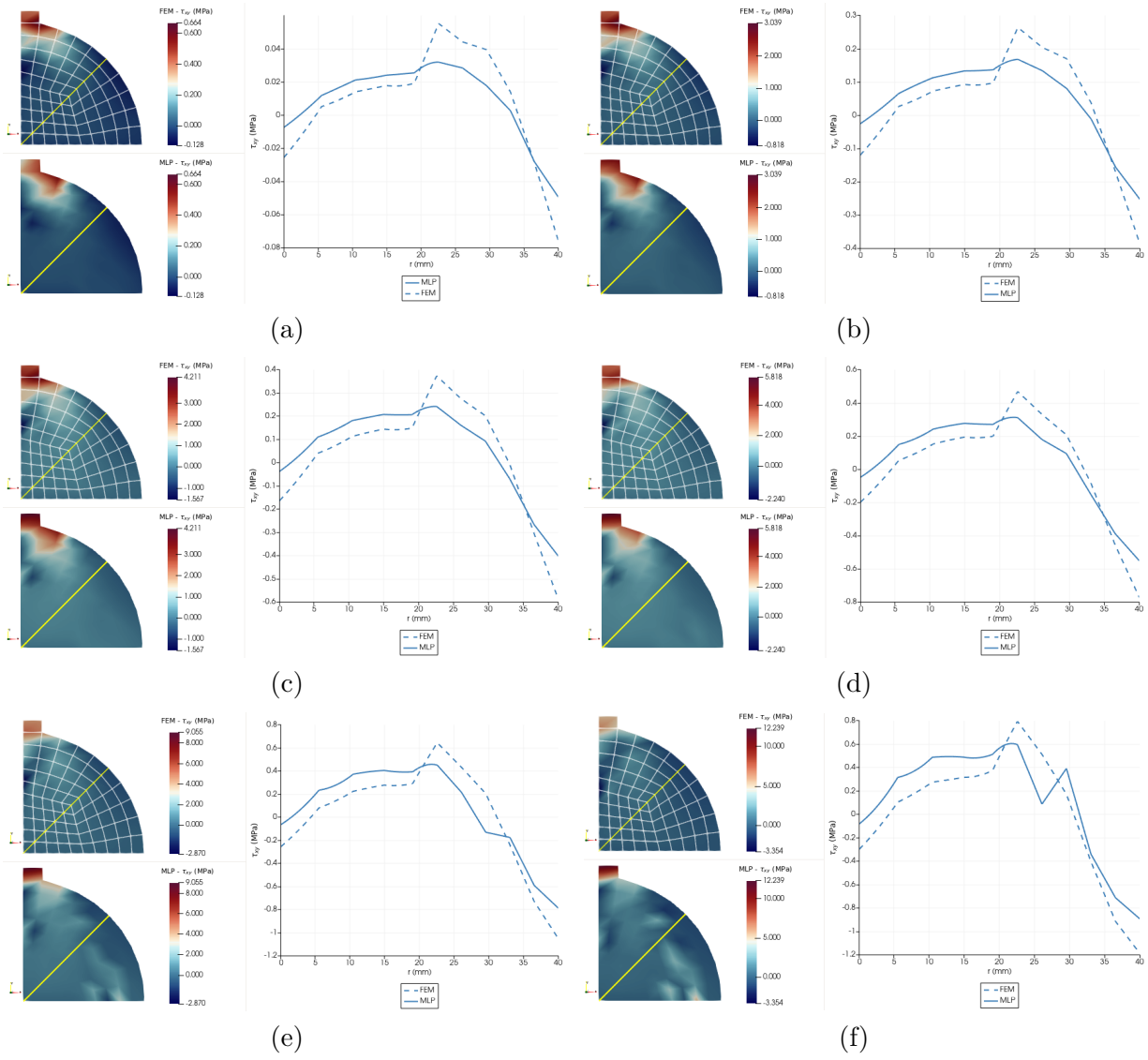


Figure C.17: Diametrical compression with crack size of 28mm - Stresses FEM × MLP - τ_{xy}. (a) Point A4 Step 10, (b) Point B4 Step 50, (c) Point C4 Step 75, (d) Point D4 Step 100, (e) Point E4 Step 150, (f) Point F4 Step 198.

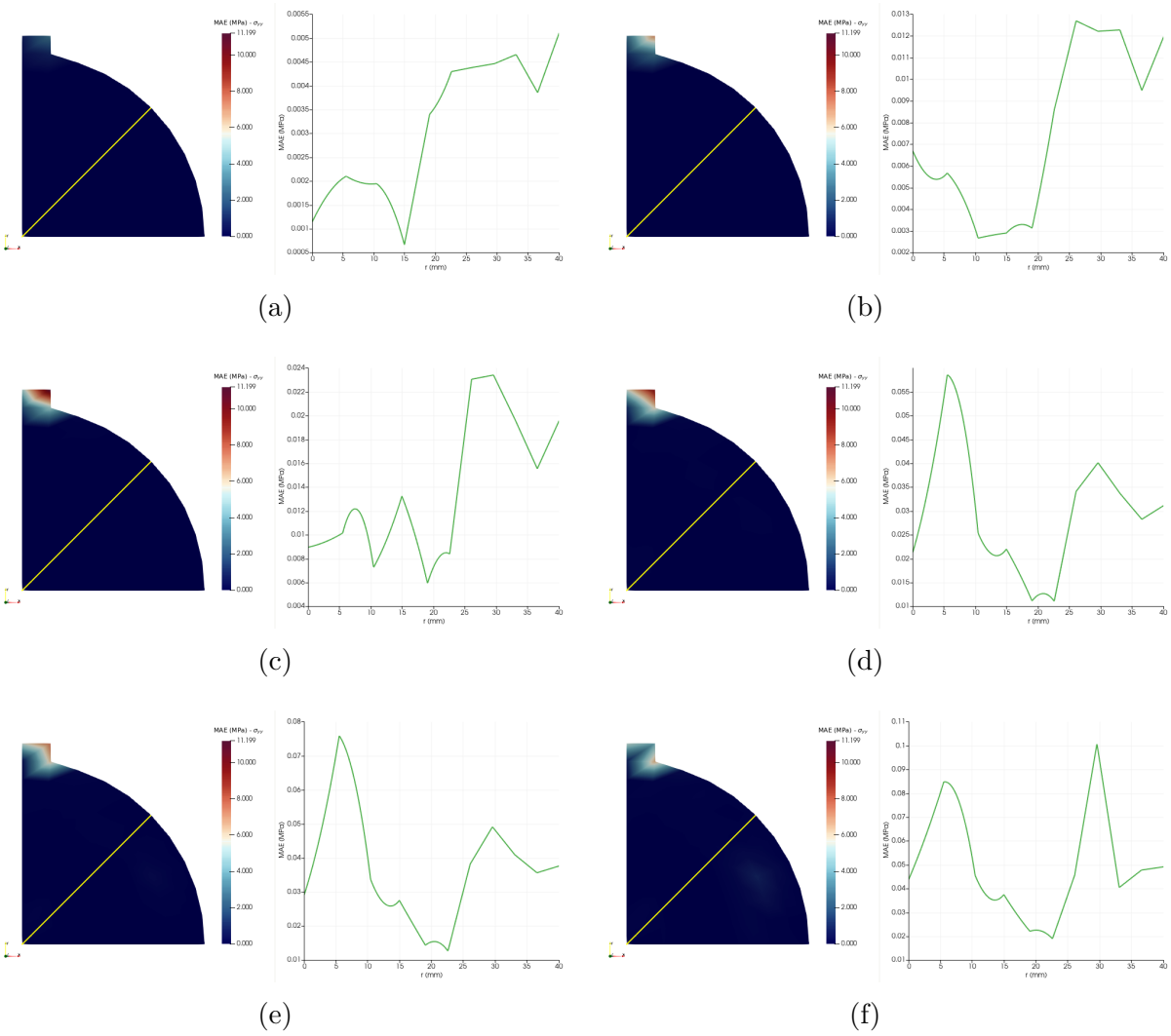


Figure C.18: Diametrical compression with crack size of 8mm - MAE accumulated - σ_{yy}. (a) Point A1 Step 10, (b) Point B1 Step 25, (c) Point C1 Step 46, (d) Point D1 Step 75, (e) Point E1 Step 90, (f) Point F1 Step 114.

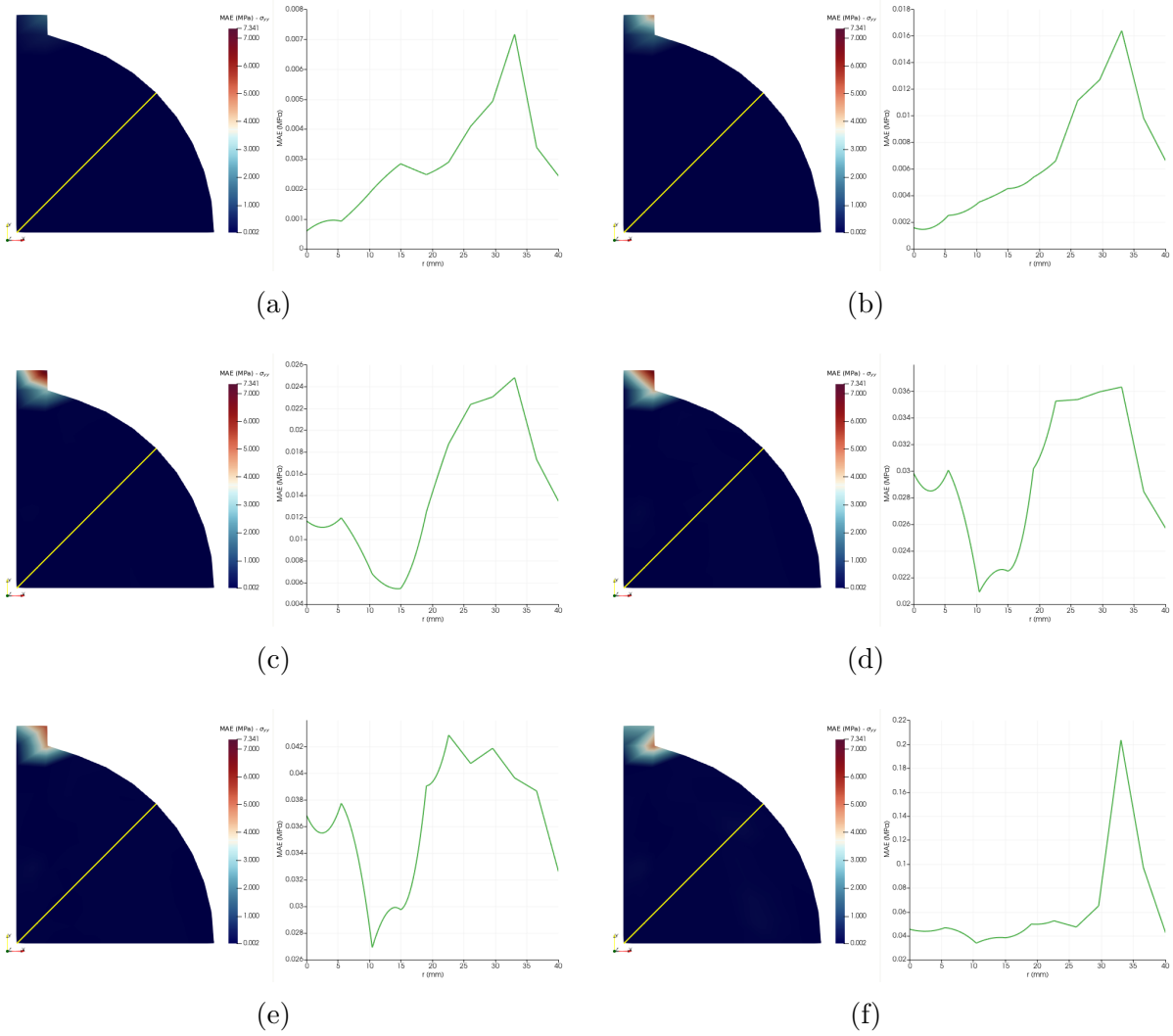


Figure C.19: Diametrical compression with crack size of 16mm - MAE accumulated - σ_{yy}. (a) Point A2 Step 10, (b) Point B2 Step 25, (c) Point C2 Step 46, (d) Point D2 Step 75, (e) Point E2 Step 90, (f) Point F2 Step 110.

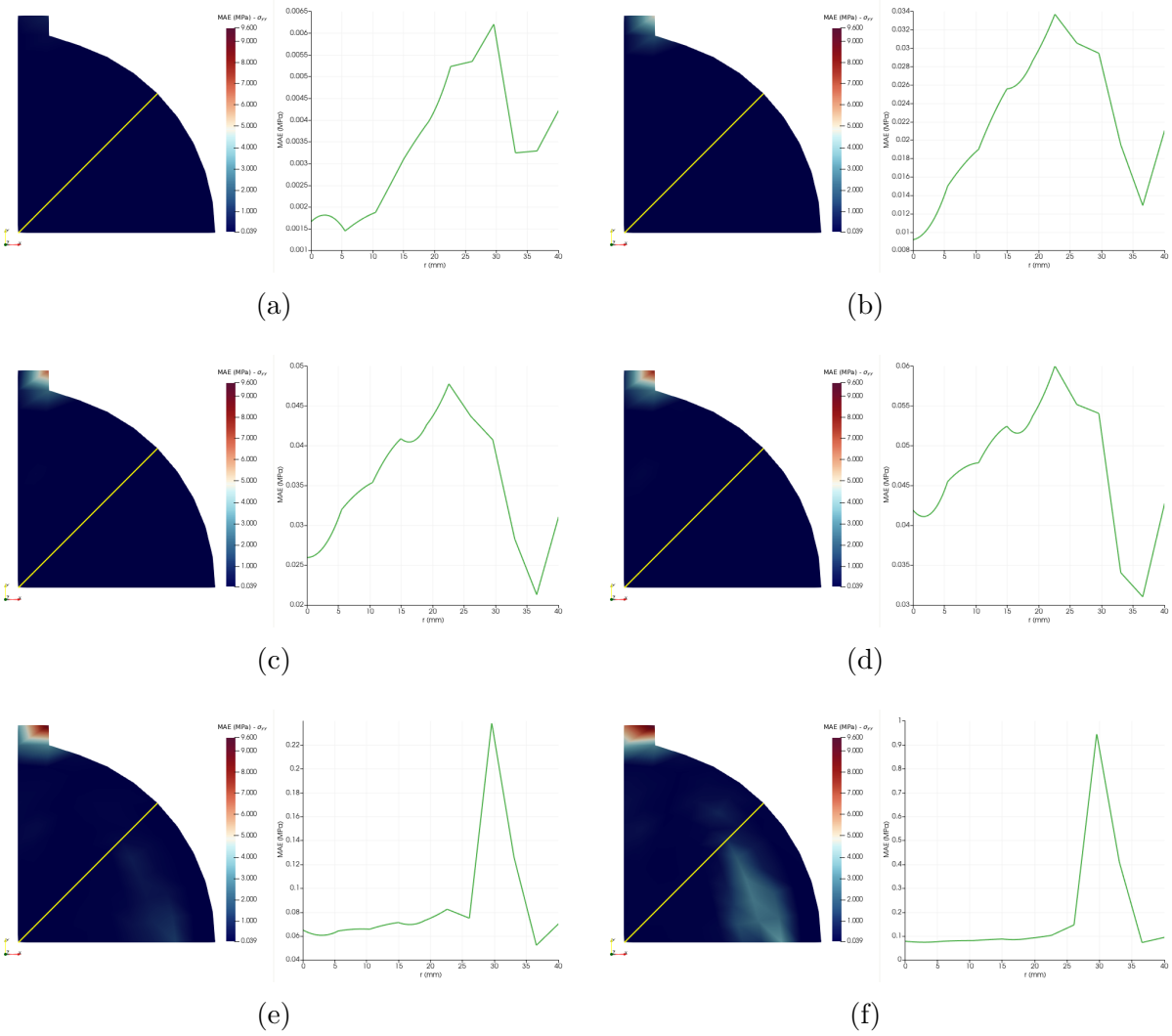


Figure C.20: Diametrical compression with crack size of 24mm - MAE accumulated - σ_{yy}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

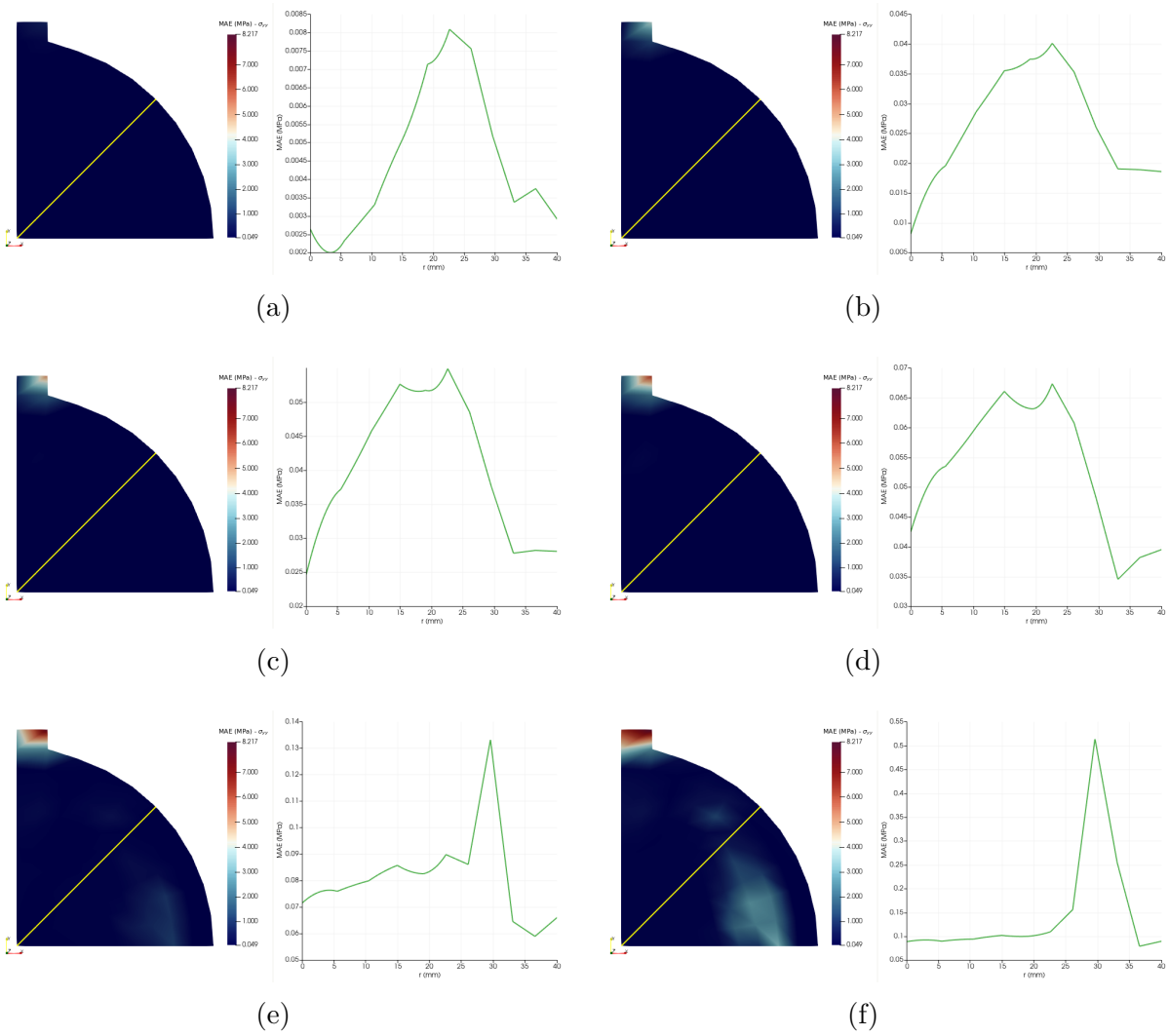


Figure C.21: Diametrical compression with crack size of 28mm - MAE accumulated - σ_{yy}. (a) Point A4 Step 10, (b) Point B4 Step 50, (c) Point C4 Step 75, (d) Point D4 Step 100, (e) Point E4 Step 150, (f) Point F4 Step 198.

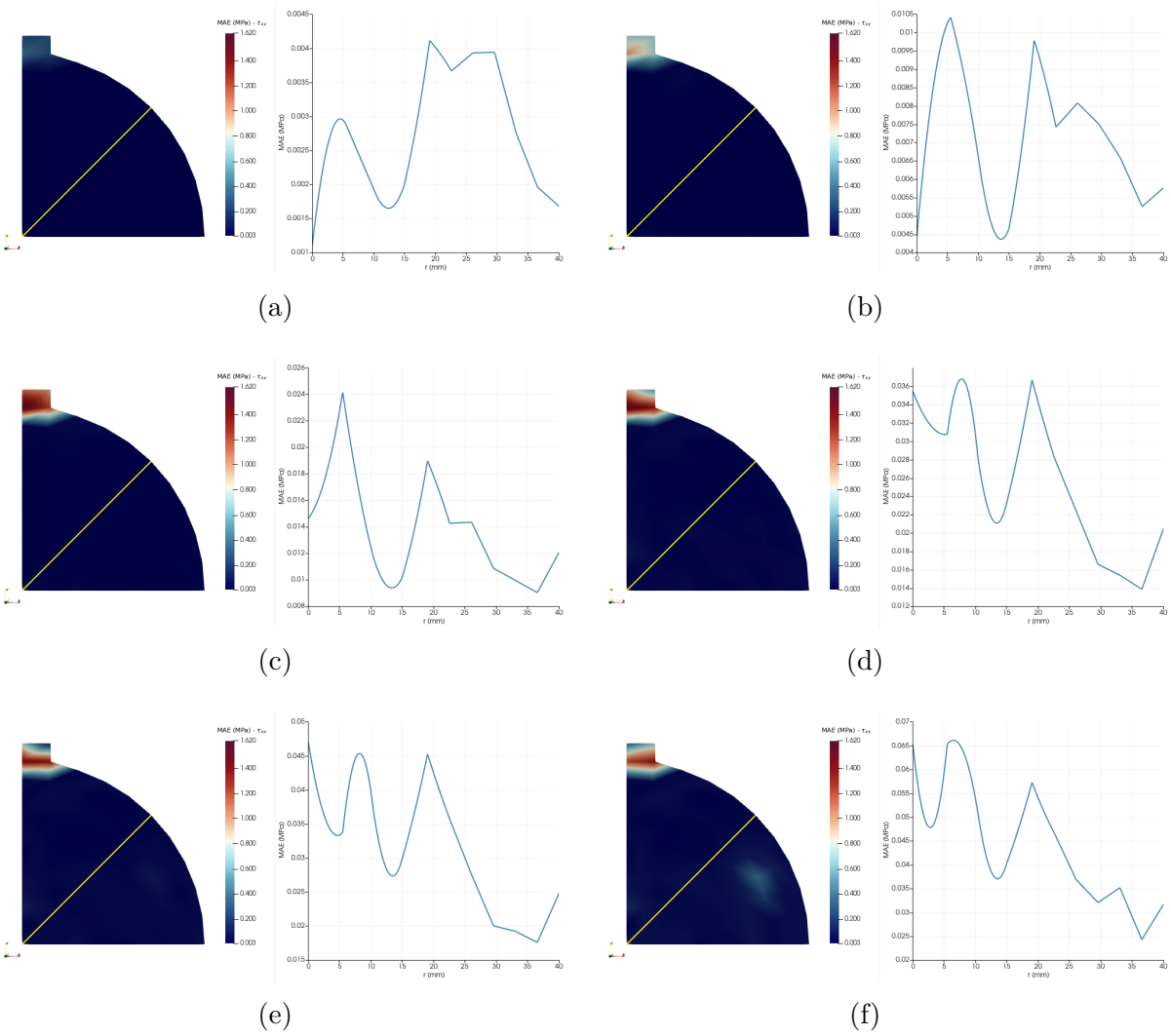


Figure C.22: Diametrical compression with crack size of 8mm - MAE accumulated - τ_{xy}. (a) Point A1 Step 10, (b) Point B1 Step 25, (c) Point C1 Step 46, (d) Point D1 Step 75, (e) Point E1 Step 90, (f) Point F1 Step 114.

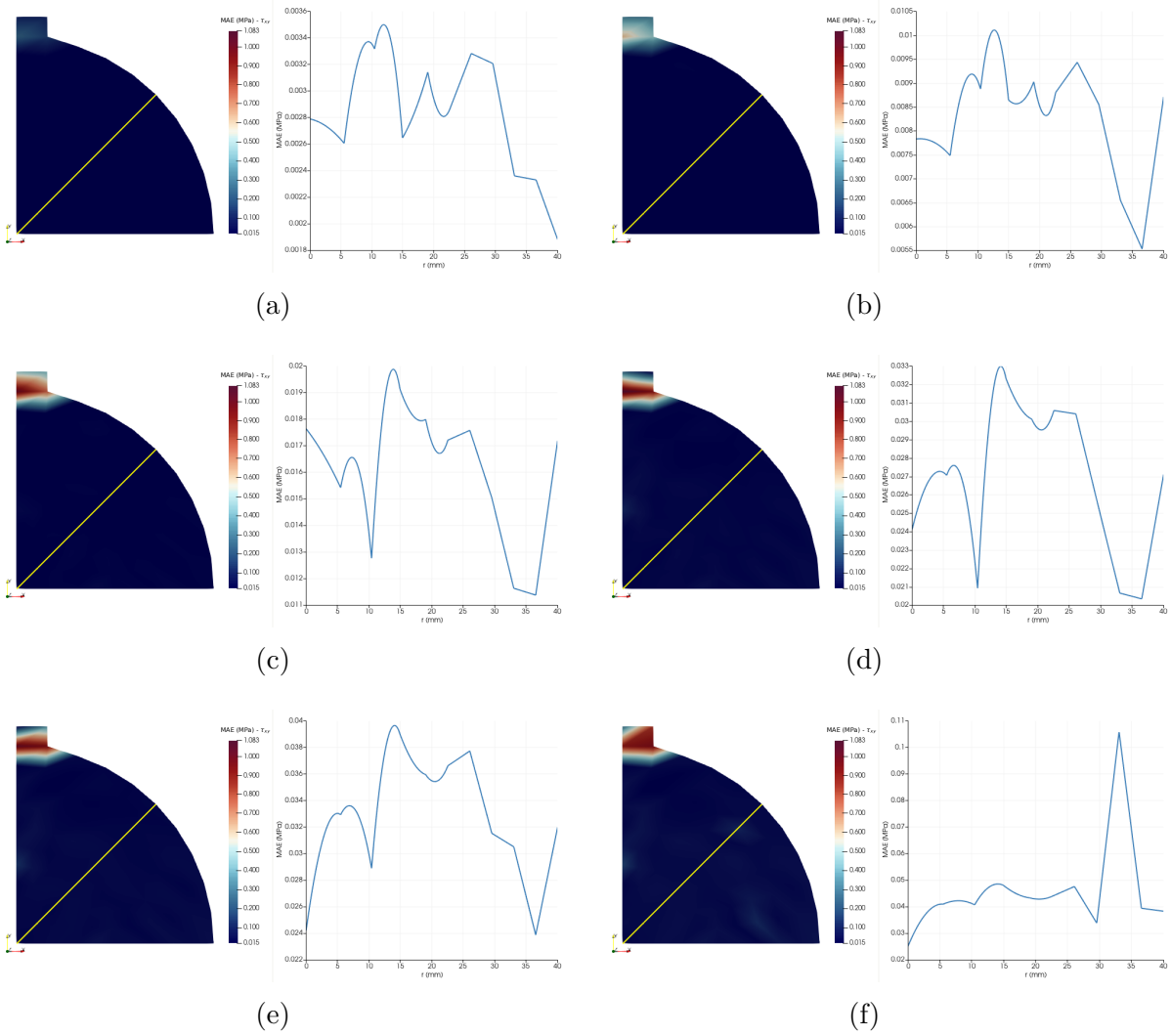


Figure C.23: Diametrical compression with crack size of 16mm - MAE accumulated - τ_{xy}. (a) Point A2 Step 10, (b) Point B2 Step 25, (c) Point C2 Step 46, (d) Point D2 Step 75, (e) Point E2 Step 90, (f) Point F2 Step 110.

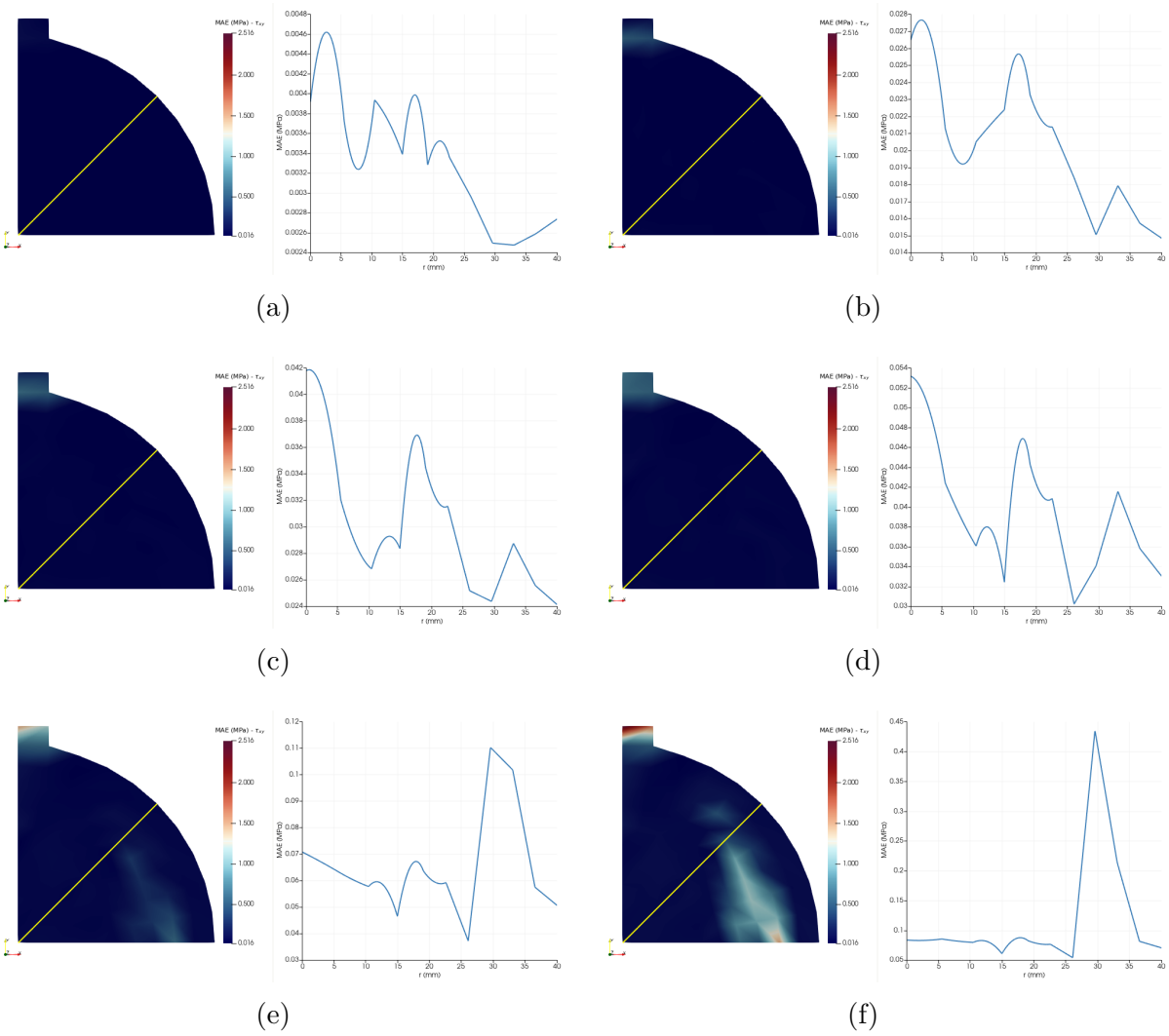


Figure C.24: Diametrical compression with crack size of 24mm - MAE accumulated - τ_{xy}. (a) Point A3 Step 10, (b) Point B3 Step 50, (c) Point C3 Step 75, (d) Point D3 Step 100, (e) Point E3 Step 150, (f) Point F3 Step 198.

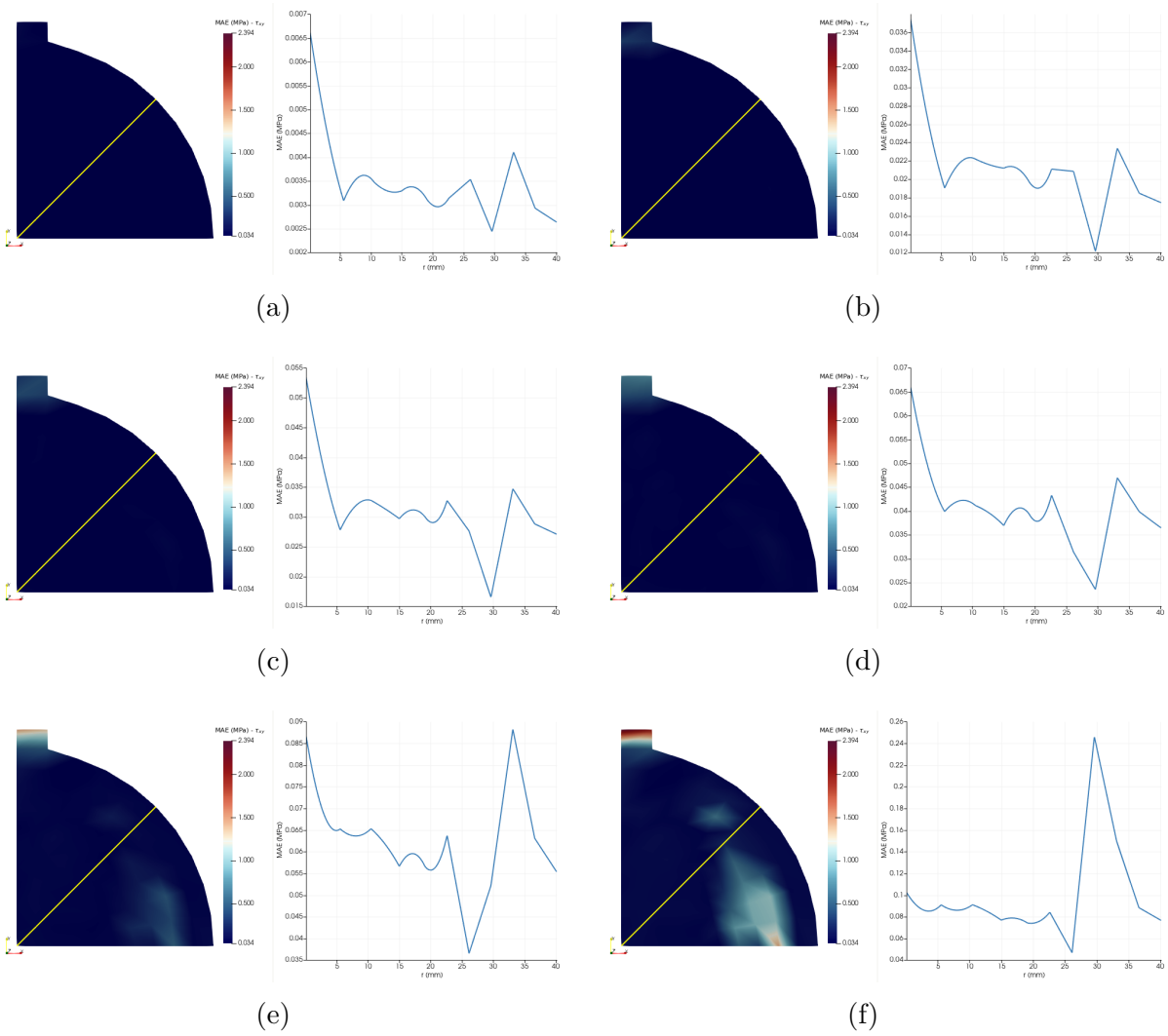
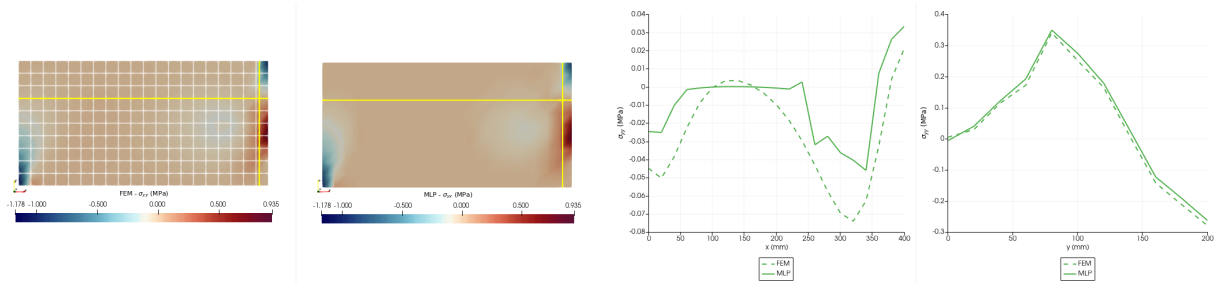
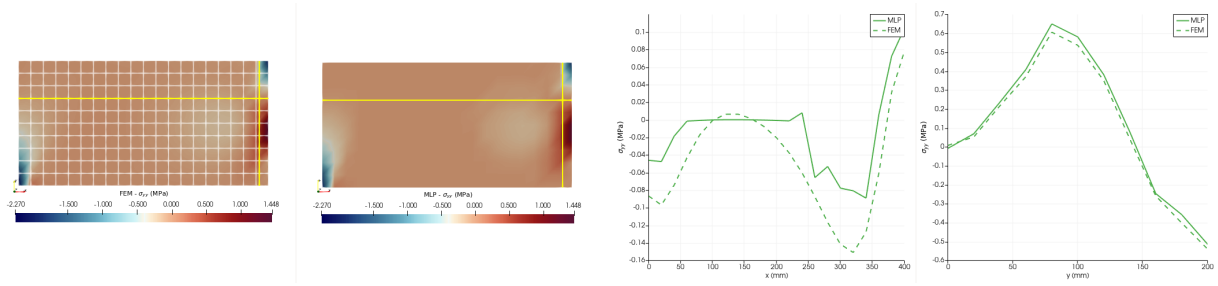


Figure C.25: Diametrical compression with crack size of 28mm - MAE accumulated - τ_{xy}. (a) Point A4 Step 10, (b) Point B4 Step 50, (c) Point C4 Step 75, (d) Point D4 Step 100, (e) Point E4 Step 150, (f) Point F4 Step 198.

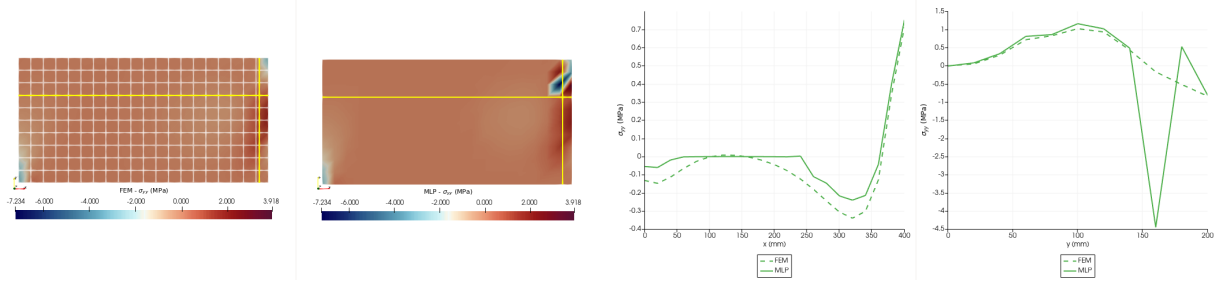
C.4 Mesh Dependency Analysis - 3-point Bending Test



(a)

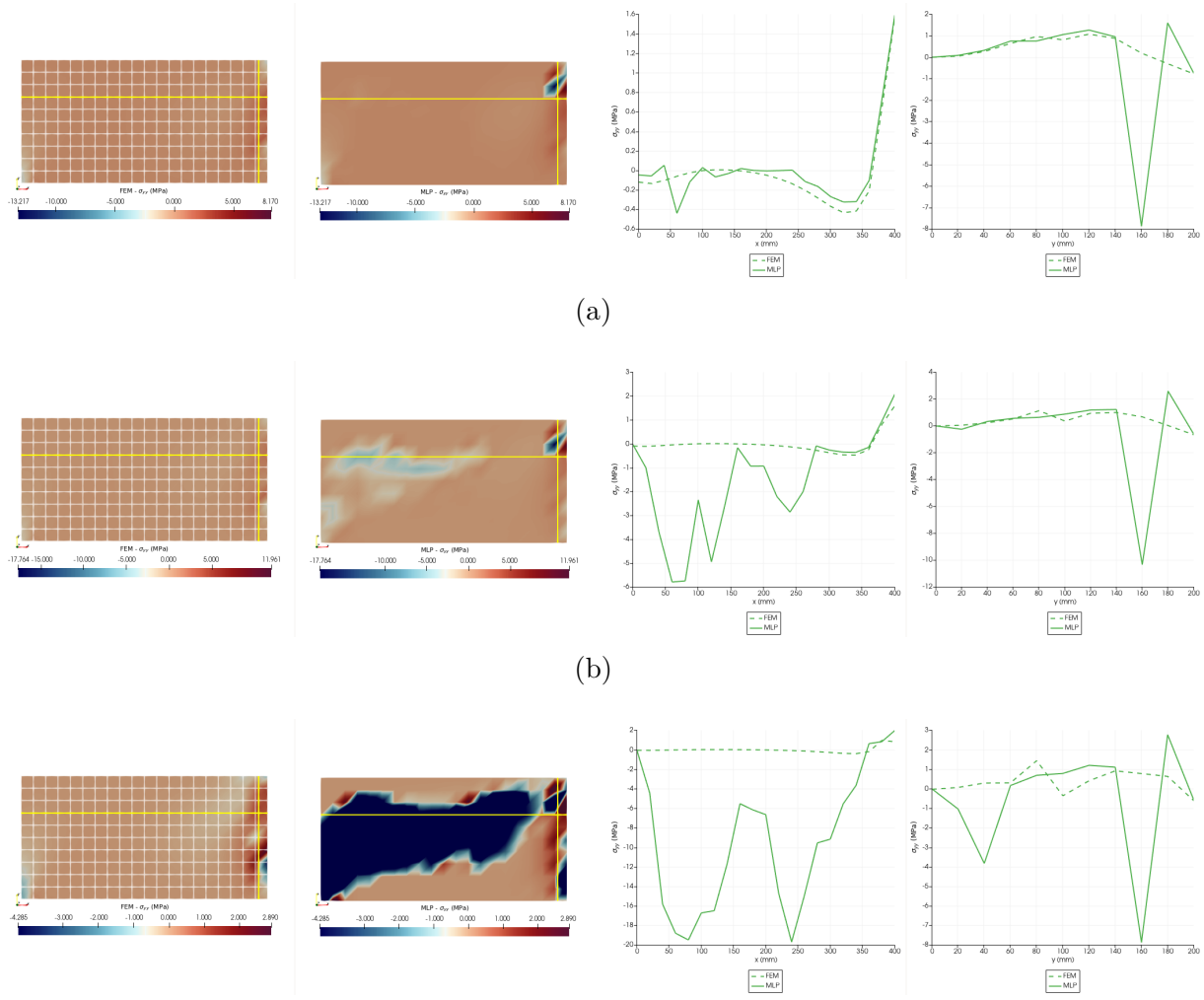


(b)



(c)

Figure C.26: 3-point bending test - M1 - Stresses FEM × MLP - σ_{yy}. (a) Point A1 Step 32, (b) Point B1 Step 64, (c) Point C1 Step 128.

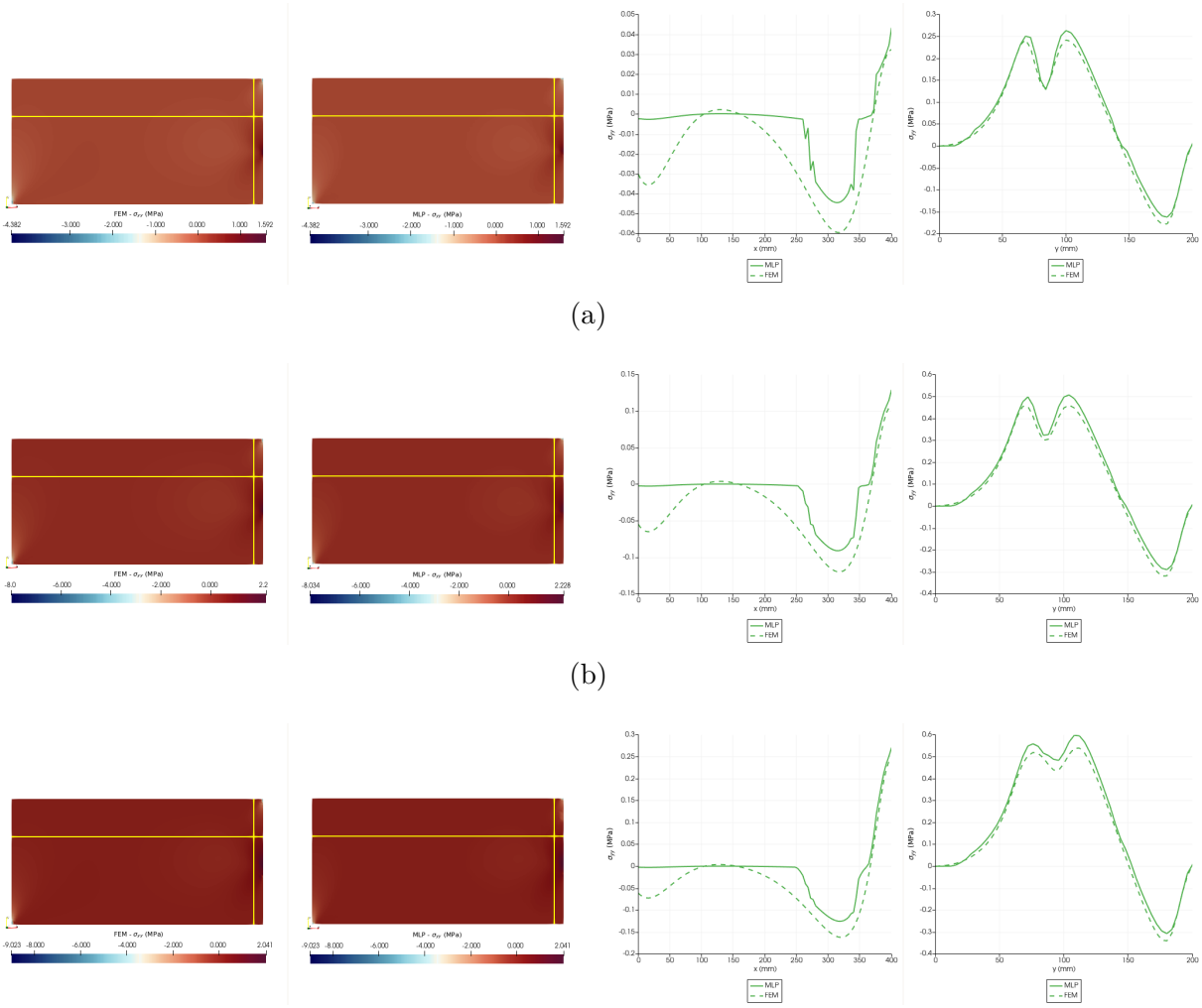


(a)

(b)

(c)

Figure C.27: 3-point bending test - M1 - Stresses FEM × MLP - σ_{yy}. (a) Point D1 Step 160, (b) Point E1 Step 200, (c) Point F1 Step 298.



(a)

(b)

(c)

Figure C.28: 3-point bending test - M2 - Stresses FEM × MLP - σ_{yy}. (a) Point A2 Step 27, (b) Point B2 Step 52, (c) Point C2 Step 64.

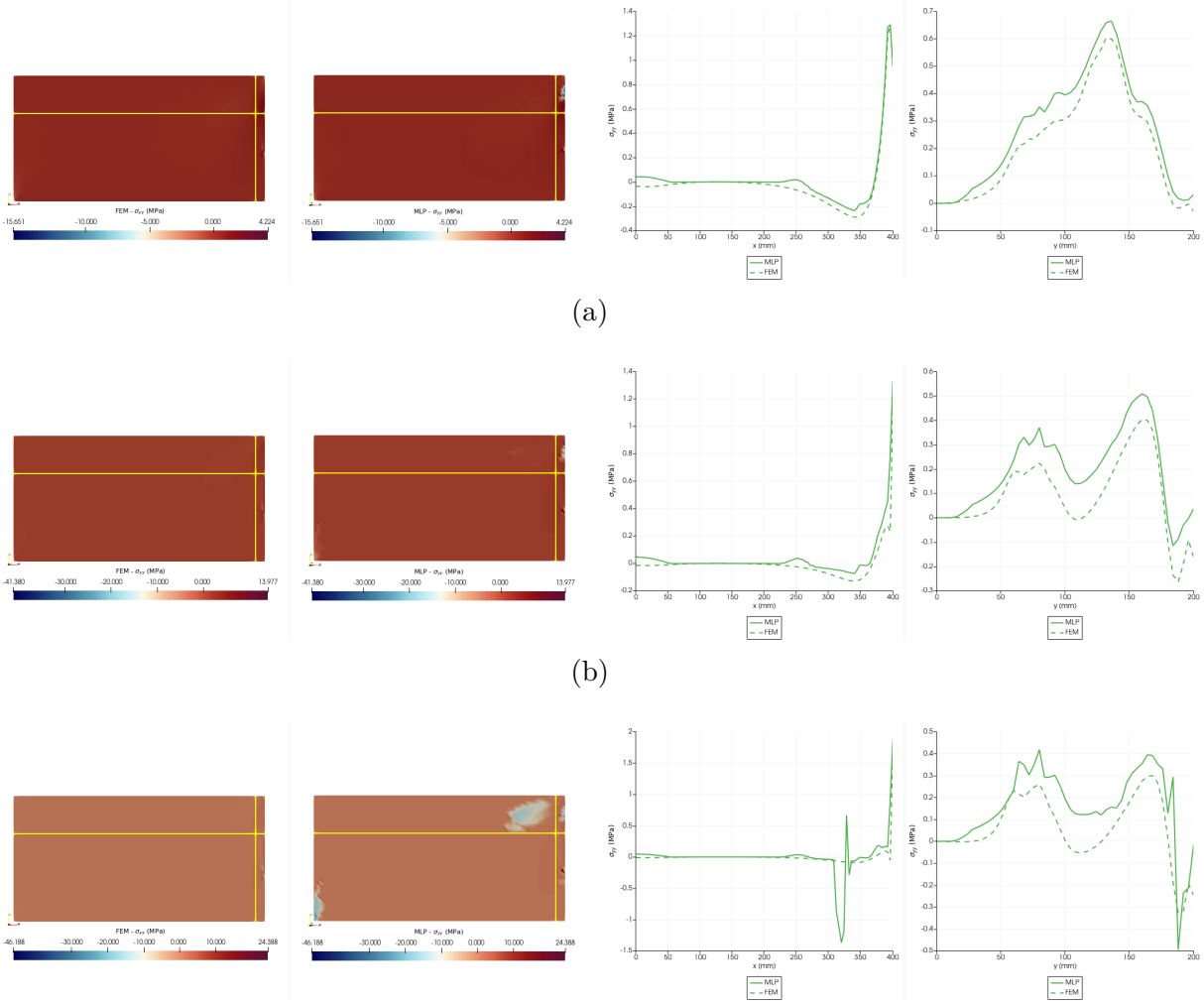


Figure C.29: 3-point bending test - M2 - Stresses FEM × MLP - σ_{yy}. (a) Point D2 Step 102, (b) Point E2 Step 176, (c) Point F2 Step 277.

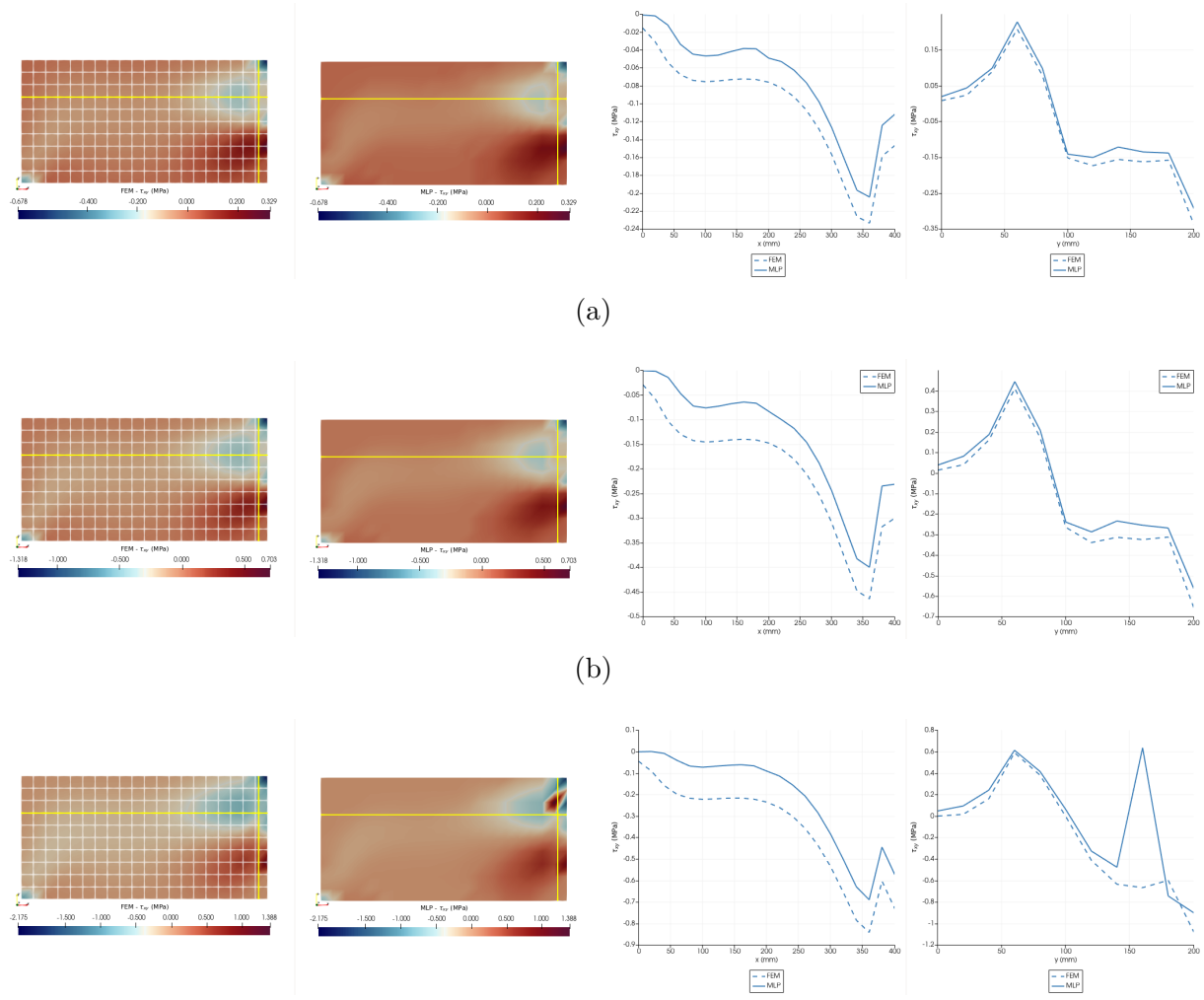


Figure C.30: 3-point bending test - M1 - Stresses FEM × MLP - τ_{xy}. (a) Point A1 Step 32, (b) Point B1 Step 64, (c) Point C1 Step 128.

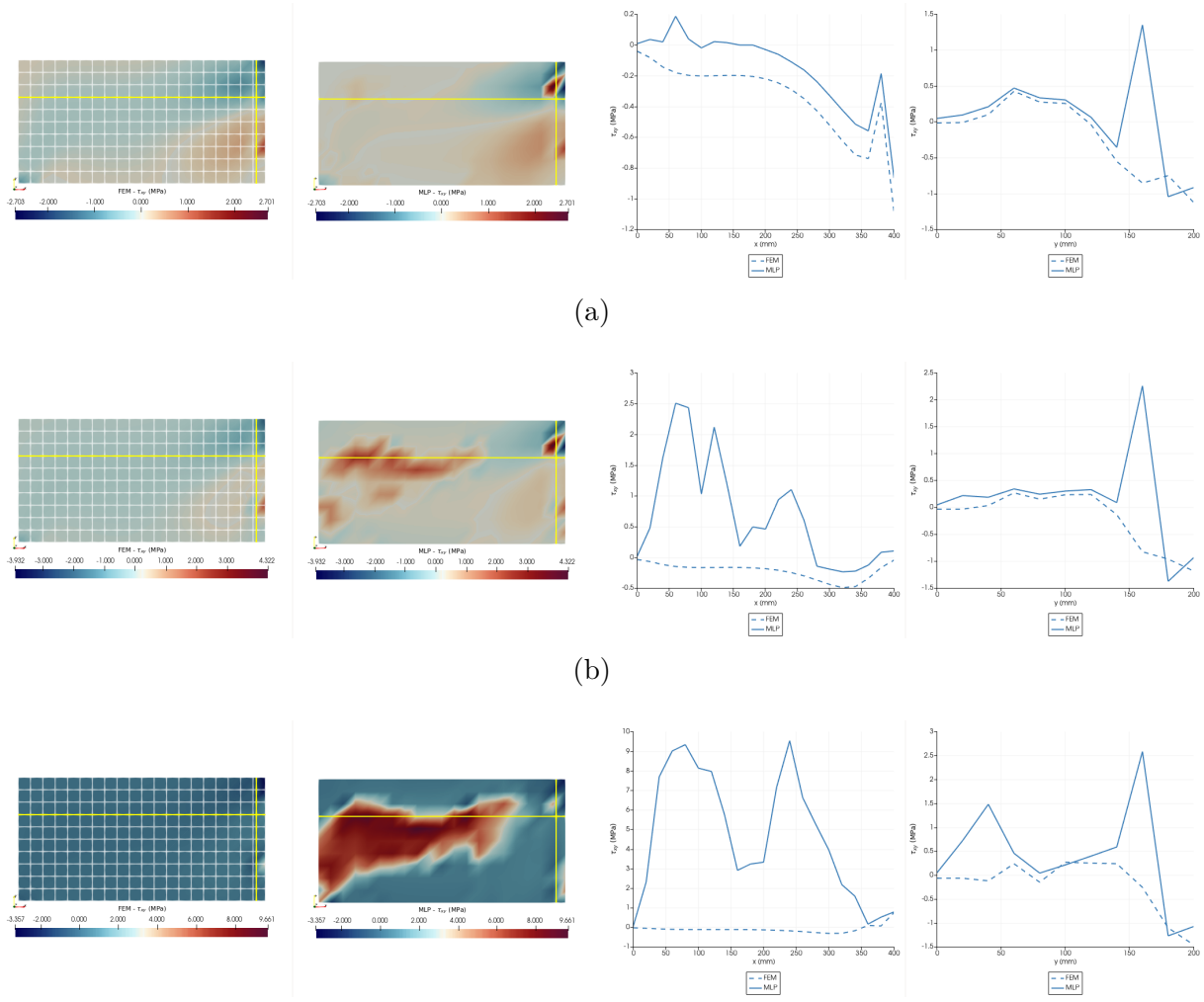
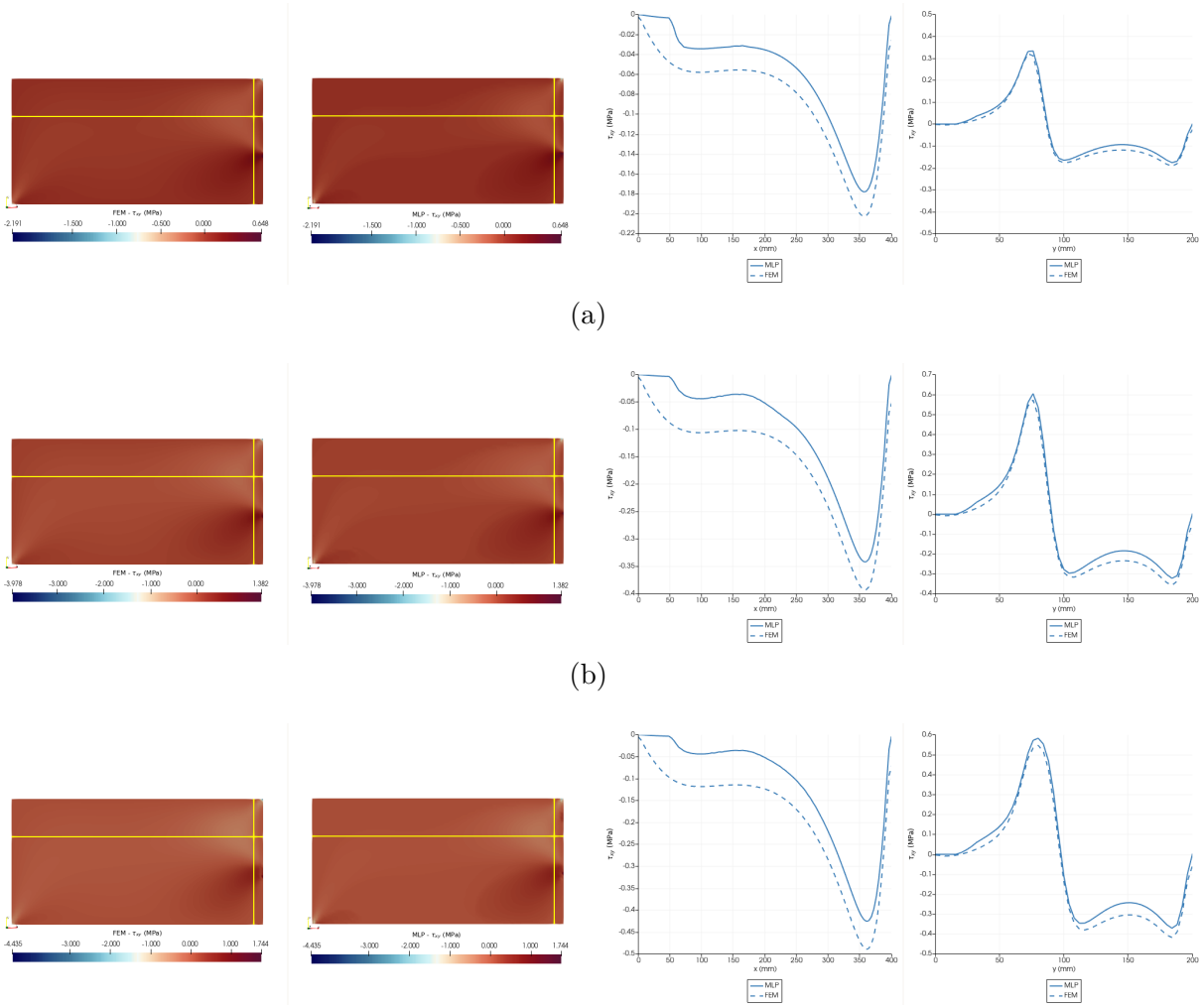


Figure C.31: 3-point bending test - M1 - Stresses FEM × MLP - τ_{xy}. (a) Point D1 Step 160, (b) Point E1 Step 200, (c) Point F1 Step 298.



(a)

(b)

(c)

Figure C.32: 3-point bending test - M2 - Stresses FEM × MLP - τ_{xy}. (a) Point A2 Step 27, (b) Point B2 Step 52, (c) Point C2 Step 64.

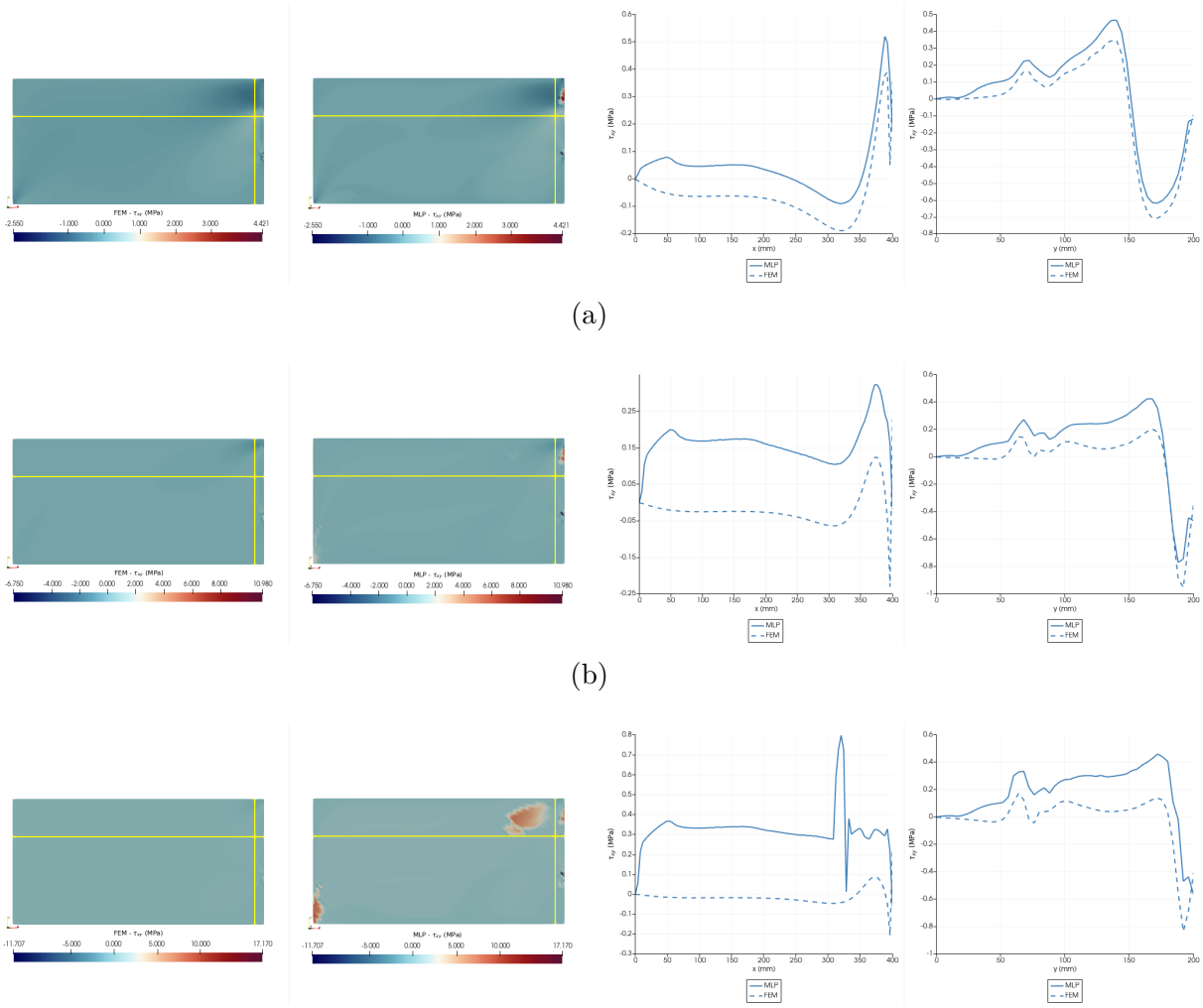


Figure C.33: 3-point bending test - M2 - Stresses FEM × MLP - τ_{xy}. (a) Point D2 Step 102, (b) Point E2 Step 176, (c) Point F2 Step 277.

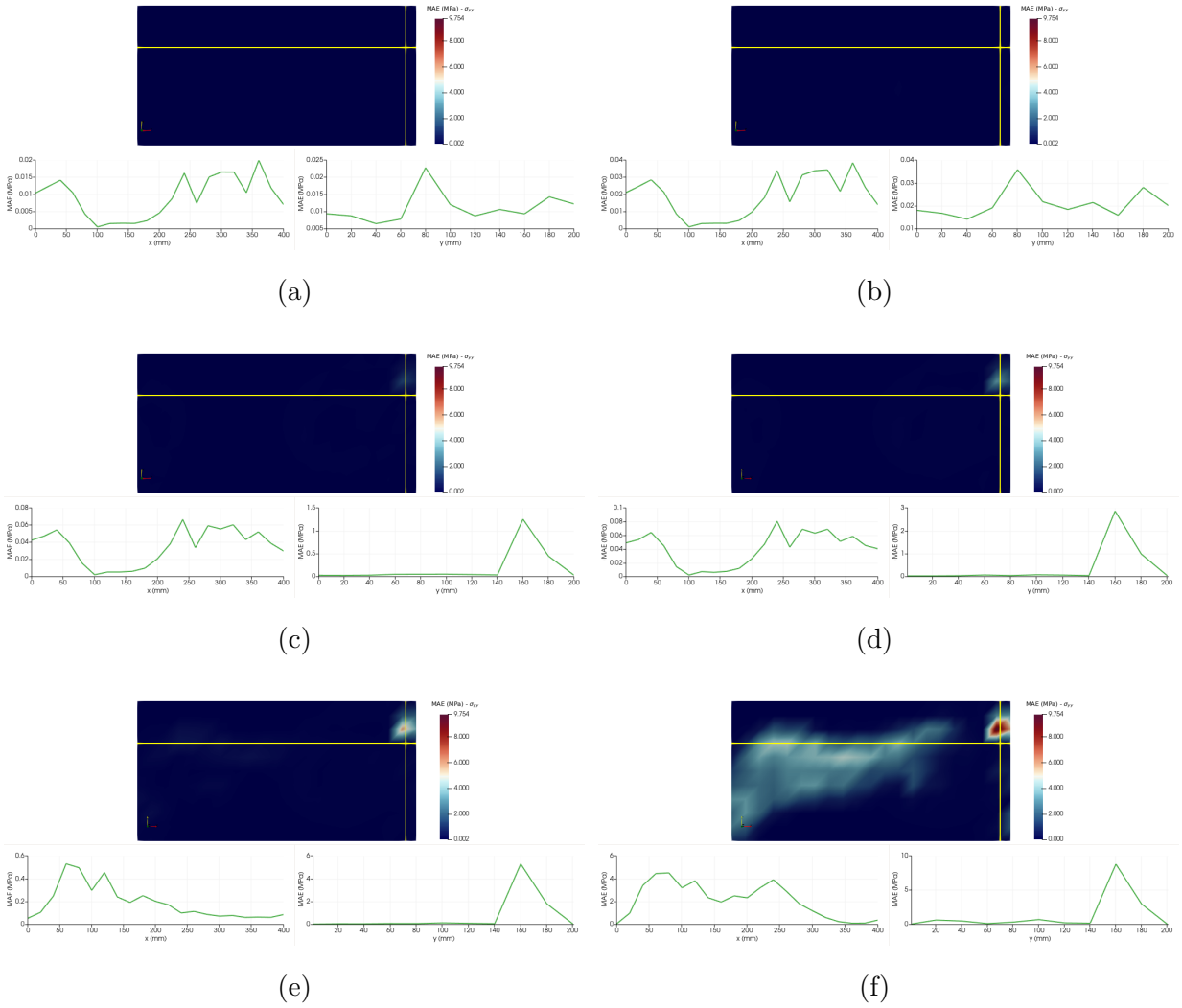


Figure C.34: 3-point bending test - M1 - MAE accumulated - σ_{yy}. (a) Point A1 Step 32, (b) Point B1 Step 64, (c) Point C1 Step 128, (d) Point D1 Step 160, (e) Point E1 Step 200, (f) Point F1 Step 298.

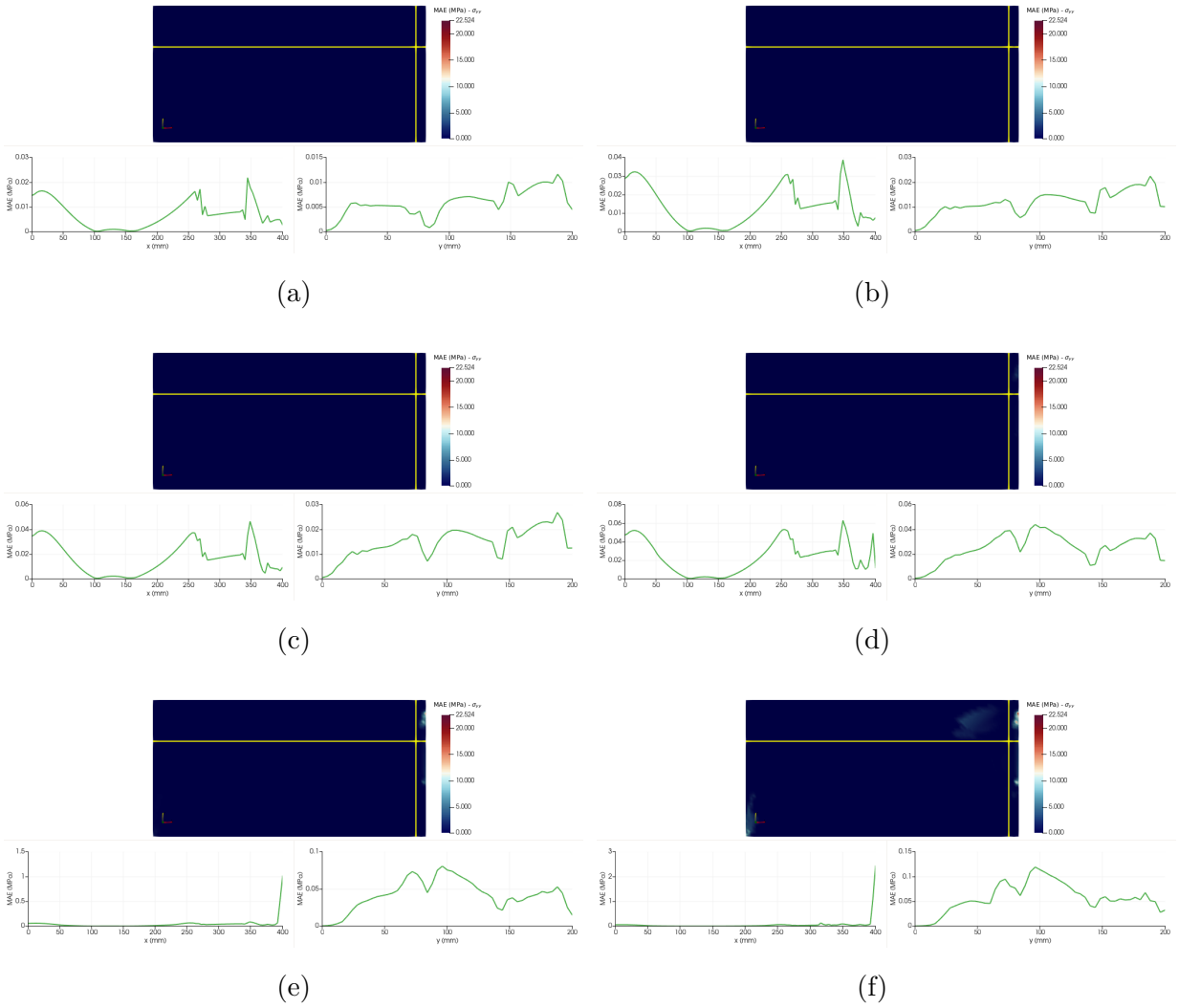


Figure C.35: 3-point bending test - M2 - MAE accumulated - σ_{yy}. (a) Point A2 Step 27, (b) Point B2 Step 52, (c) Point C2 Step 64, (d) Point D2 Step 102, (e) Point E2 Step 176, (f) Point F2 Step 277.

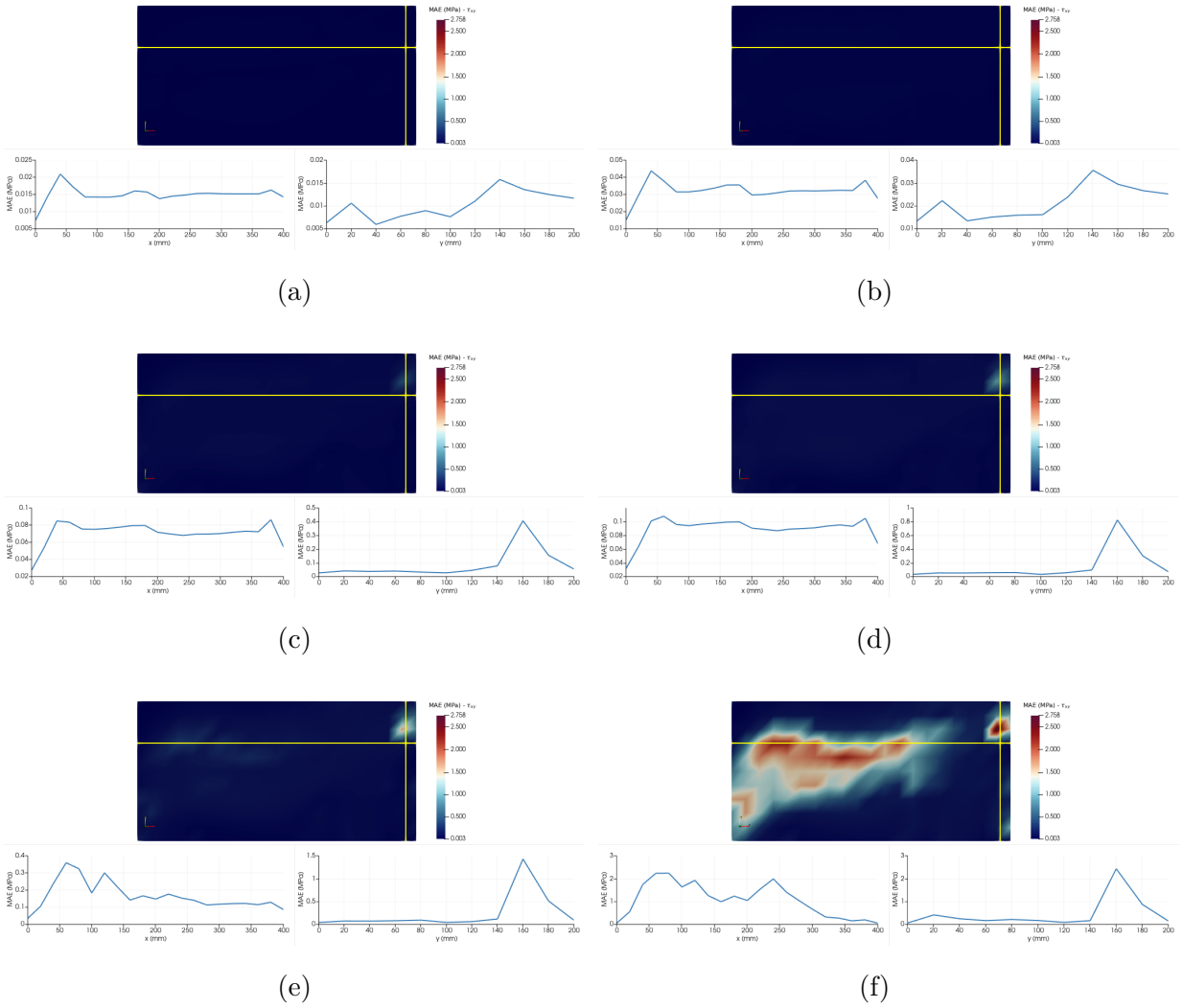


Figure C.36: 3-point bending test - M1 - MAE accumulated - τ_{xy}. (a) Point A1 Step 32, (b) Point B1 Step 64, (c) Point C1 Step 128, (d) Point D1 Step 160, (e) Point E1 Step 200, (f) Point F1 Step 298.

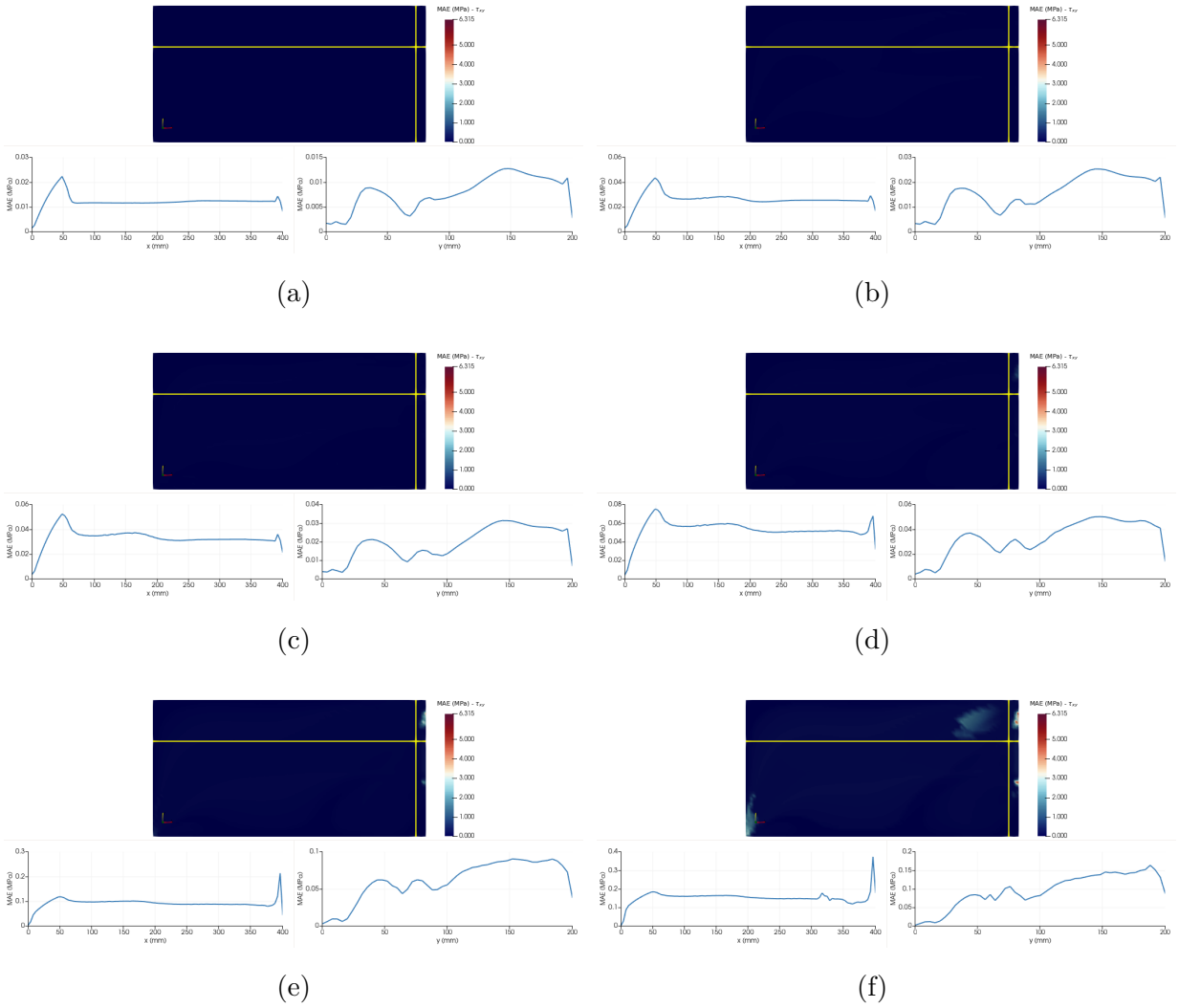


Figure C.37: 3-point bending test - M2 - MAE accumulated - τ_{xy}. (a) Point A2 Step 27, (b) Point B2 Step 52, (c) Point C2 Step 64, (d) Point D2 Step 102, (e) Point E2 Step 176, (f) Point F2 Step 277.