FEDERAL UNIVERSITY OF SANTA CATARINA
TECHNOLOGICAL CENTER
DEPARTMENT OF MECHANICAL ENGINEERING
UNDERGRADUATE COURSE IN MECHANICAL ENGINEERING

Lucas Manassés Pinheiro de Souza

IMPLEMENTATION OF A DEEP NEURAL NETWORK FOR THE CORRECTION OF PLASTICITY EFFECTS IN THE HOLE DRILLING METHOD FOR MEASURING HIGH LEVELS OF RESIDUAL STRESS IN FLEXIBLE RISERS

Florianópolis
2022

Lucas Manassés Pinheiro de Souza

IMPLEMENTATION OF A DEEP NEURAL NETWORK FOR THE CORRECTION OF PLASTICITY EFFECTS IN THE HOLE DRILLING METHOD FOR MEASURING HIGH LEVELS OF RESIDUAL STRESS IN FLEXIBLE RISERS

Undergraduate thesis submitted to the Undergraduate Program in Mechanical Engineering at Federal University of Santa Catarina as part of the requirements necessary to obtain the bachelor's degree in Mechanical Engineering.
Advisor: Prof. Armando Albertazzi Gonçalves Jr., Dr. Eng.
Co-advisor: Matias Roberto Viotti, Dr. Ing.

Florianópolis
2022

Identification form of the undergraduate thesis elaborated by the author, via Programa de Geração Automática da Biblioteca Universitária da UFSC.

Manassés Pinheiro de Souza, Lucas
Implementation of a deep neural network for the correction of plasticity effects in the hole drilling method for measuring high levels of residual stress in flexible risers / Lucas Manassés Pinheiro de Souza ; orientador, Armando Albertazzi Gonçalves Jr., coorientador, Matias Roberto Viotti, 2022.
105 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Engenharia Mecânica, Florianópolis, 2022.

Inclui referências.

1. Engenharia Mecânica. 2. Residual Stress. 3. Hole Drilling Method. 4. Deep Artificial Neural Network. 5. Plasticity Effect. I. Albertazzi Gonçalves Jr., Armando. II. Viotti, Matias Roberto . III. Universidade Federal de Santa Catarina. Graduação em Engenharia Mecânica. IV. Título.

Lucas Manassés Pinheiro de Souza

IMPLEMENTATION OF A DEEP NEURAL NETWORK FOR THE CORRECTION OF PLASTICITY EFFECTS IN THE HOLE DRILLING METHOD FOR MEASURING HIGH LEVELS OF RESIDUAL STRESS IN FLEXIBLE RISERS

This undergraduate thesis was considered adequate for obtaining the bachelor's degree in Mechanical Engineering and it was approved in its final form by the Examining Committee and the Undergraduate Program in Mechanical Engineering at Federal University of Santa Catarina.

Florianópolis, July 28, 2022.

Documento assinado digitalmente
Carlos Enrique Nino Bohorquez
Data: 19/09/2022 17:10:38-0300
CPF: 564.878.519-49
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Enrique Niño Bohórquez, Dr. Eng.
Coordinator of the Undergraduate Program in Mechanical Engineering

Examination Committee:

Documento assinado digitalmente
THIAGO WILVERT
Data: 14/09/2022 09:47:14-0300
CPF: 056.338.459-00
Verifique as assinaturas em https://v.ufsc.br

Thiago Wilvert, Me. Eng.
Unofficial co-advisor
Federal University of Santa Catarina

Documento assinado digitalmente
Carlos Enrique Nino Bohorquez
Data: 19/09/2022 17:11:05-0300
CPF: 564.878.519-49
Verifique as assinaturas em https://v.ufsc.br

Prof. Carlos Enrique Niño Bohórquez, Dr. Eng.
Responsible Professor for the Manufacturing, Materials and Metrology (3M) fields in the undergraduate course EMC5022 - Trabalho de Curso

Documento assinado digitalmente
HERBERTH BIRCK FROHLICH
Data: 14/09/2022 19:24:00-0300
CPF: 013.326.800-46
Verifique as assinaturas em https://v.ufsc.br

Herberth Birck Fröhlich, Dr. Eng.
Invited Member
Researcher at DELL Technologies

Dedicated to my beloved mother and sister, Vera and Sâmela for their patience and understanding over the years.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God, the Almighty, for blessing me with all my mental capabilities, and for the opportunity to study, to do science, and to finish writing this undergraduate thesis. As well as for the blessed life I have, for my academic achievements and for placing the right persons at the right moment in my path.

And I would like to thank my advisor Prof. Armando Albertazzi Gonçalves Jr. for the opportunity to be his mentee. I also thank him for accepting the challenge of having oriented me in a research topic with such scientific complexity for a bachelor thesis.

I would also like to thank my joint advisor Matias Roberto Viotti, Dr. Eng. for his orientations and technical suggestions regarding the analysis and measurement of residual stresses in flexible risers.

My special thanks goes to my colleague Thiago Wilvert, Ms. Eng. who embraced the idea of implementing an artificial neural network with much enthusiasm and has much contributed for the realization and structuring of this academic work.

I want to express my gratitude towards LABMETRO, FEESC, and PETROBRAS for the financial support and the offered resources for the completion of this undergraduate thesis.

Finally, I would like to thank my family and friends for their support, love, and understanding during the time of my studies. Especially to my mother and sister, who have loved me despite my difficult personality. I also take the opportunity to thank my grandmother who always helped me in prayers.

“If a machine is expected to be infallible, it cannot also be intelligent.”

Alan Turing, 1947

ABSTRACT

In the work of this undergraduate thesis, a Deep Artificial Neural Network (DANN) model was implemented to perform the correction of plasticity effects in the Hole-Drilling Method when measuring high levels of residual stresses in flexible risers, given the limitations of the American Society for Testing and Materials standard E837-13a in the plastic regime. The model was trained on data resulting from several elastoplastic numerical simulations performed via Finite Element Method. Two databases were established for the training and test of the DANN model, one based on uniform stress values, and another based on non-uniform stress values along the hole wall. The pre-processing of the features of the databases was based on similar works. In the validation of the test set of the two databases, both DANN models performed relatively accurately with prediction errors on the order of 4.4×10^{-3} for the database of Test 1 and 3.8×10^{-3} for the database of Test 2. A classical Multivariable Linear Regression (MLR) model was also implemented to compare the performance and errors of the DANN model, it was found that the prediction error from the MLR model was higher than that of the DANN model for the database of Test 1, whereas for the database of Test 2, the MLR model prediction error was lower than that of the DANN model. The result questioned the computational efficiency of implementing a DANN model. The values of the Plasticity Factors f in the testing set were investigated. It was observed that the predictions with the largest errors were simulated with the same value of f (1.30). For exemplification purposes and not model validity, values of four profiles (with different values of f) were taken from the total set of data from each database and used as input for the correction.

Keywords: Residual Stress. Hole-Drilling Method. Deep Artificial Neural Network. Plasticity Effect.

RESUMO

Neste Trabalho de final de Curso, foi implementado um modelo de Redes Neurais Artificias Profunda para realizar a correção dos efeitos de plasticidade no Método do Furo Cego ao medir altos níveis de tensões residuais em risers flexíveis, dadas as limitações no regime plástico da norma E837-13a da Sociedade Americana de Testagem e Materiais. O modelo foi treinado em dados resultantes de várias simulações numéricas elastoplásticas realizadas via Método dos Elementos Finitos. Dois bancos de dados foram estabelecidos para o treinamento e teste dos modelos, um baseado em valores de tensão uniformes e outro baseado em valores de tensão não uniformes ao longo da parede do orifício. O pré-processamento das características dos bancos de dados foi baseado em trabalhos similares. Na validação do conjunto de teste dos dois bancos de dados, ambos os modelos de Redes Neurais Artificias Profunda performaram com relativa precisão com erros de predição na ordem de 4,4 x 10^{-3} para o banco de dados do Teste 1 e 3,8 x 10^{-3} para o banco de dados do Teste 2. Um modelo clássico de Regressão Linear Multivariável também foi implementado para comparar o desempenho e os erros do modelo de Redes Neurais, constatou-se que o erro de previsão do modelo Linear era era maior do que o do modelo de Redes Neurais para o banco de dados do Teste 1, enquanto para o banco de dados do Teste 2, o erro de previsão do modelo Linear era menor do que o do modelo de Redes Neurais. O resultado questionou a eficiência computacional da implementação de um modelo de Redes Neurais Artificias Profunda. Os valores dos Fatores de Plasticidade f no conjunto de testes foram investigados. Observou-se que as previsões com os maiores erros foram simuladas com o mesmo valor de f (1,30). Para fins de exemplificação e não de validade do modelo, valores de quatro perfis (com diferentes valores de f) foram retirados do conjunto total de dados de cada banco de dados e usados como entrada para a correção.

Palavras-chave: Tensão residual. Método do Furo Cego. Rede Neural Artificial Profunda. Efeito de Plasticidade.

LIST OF FIGURES

LIST OF CHARTS

LIST OF TABLES

LIST OF ABBREVIATIONS

ABNT	Associação Brasileira de Normas Técnicas
APDL	Ansys Parametric Design Language
ANN	Artificial Neural Network
ASTM	American Society for Testing and Materials
DANN	Deep Artificial Neural Network
IDE	Integrated Development Environment
FE	Finite Element
FEA	Finite Element Analysis
FEM	Finite Element Method
FEESC	Fundação Stemmer para Pesquisa, Desenvolvimento e Inovação
HDM	Hole-Drilling Method
IM	Integral Method
LABMETRO	Laboratório de Metrologia e Automatização
MLR	Multivariable Linear Regression
MSE	Mean Squared Error
NN	Neural Network
PETROBRAS	Brazilian Petroleum Corporation
TRRiFlex	Residual Stress in Flexible Risers
UFSC	Universidade Federal de Santa Catarina

LIST OF SYMBOLS

Capital Letters

B	Clockwise angle from the x-axis to σ1 (maximum principal stress direction) [°]
E	Young's Modulus [GPa]
K^{ep}	Tangent modulus [GPa]
P, Q, T	Combination stresses [MPa]
C	Cost function [-]
$S^{(l)}$	Summation Function [-]
$W_{(t)}^{(l)}$, $W_{(t-1)}^{(l)}$	Weight in the layer l and iteration t
Y	Label vector, the output values
X	Dataset matrix of input values

Lowercase Letters

$\bar{a}$, b	Calibration constants [-]
c	Tri-diagonal 'second derivative matrix' [-]
f	Dimensionless loading intensity factor [-]
$h^{(l)}$	Activate neuron in the l layer [-]
j	Number of hole depth increments performed [-]
k	Sequence number for the hole depth increment [-]
Δx	Length of the grid [mm]
p, q, t	Combination strains [μm/m]
u_{x_1}, $u_{x_{12}}$	Displacements at the ends of the grid [mm]
y,	Label value, desired value, supervised value
$\hat{y}$	Predicted value
w	Width of the grid [mm]

Greek Letters

α_P	Regularization factor for **P** stresses [-]
α_Q	Regularization factor for **Q** stresses [-]
α_T	Regularization factor for **T** stresses [-]
β	Angle between σ_{max} and the strain gauge ε_1 [°]
γ	Angle between strain gauge 3 and σ_x [°]
ε	Total strain [μm/m]

ε_1, ε_2, ε_3	Strains measured with strain gages [-]
ε_e	Elastic strain [μm/m]
ε_p	Plastic strain [μm/m]
θ	Angle between the strain gauge (or any other point) and the x-axis [º]
σ	Stress [MPa]
σ_0	Stress profile coefficient [MPa]
$\sigma_{10\%}$	True stress at a true plastic strain of 10% [MPa]
σ_{PD}	Predefined stress profile [MPa]
σ_{IM}	Stress profile obtained with the integral method [MPa]
σ_{eqv}	Von mises equivalent stress [MPa]
σ_{max}	Maximum principal stress [MPa]
σ_{min}	Minimum principal stress [MPa]
σ_{yield}	Yield stress of the material [MPa]
$\sigma\ (S^{(l)})$	Activation Function [-]
u_r	Radial displacement [mm]
u_θ	Tangential displacement [mm]
τ	Shear stress [MPa]
v	Poisson's ratio [-]
Φ_1, Φ_2	Local angles [º]
Ω	Stress ratio [-]

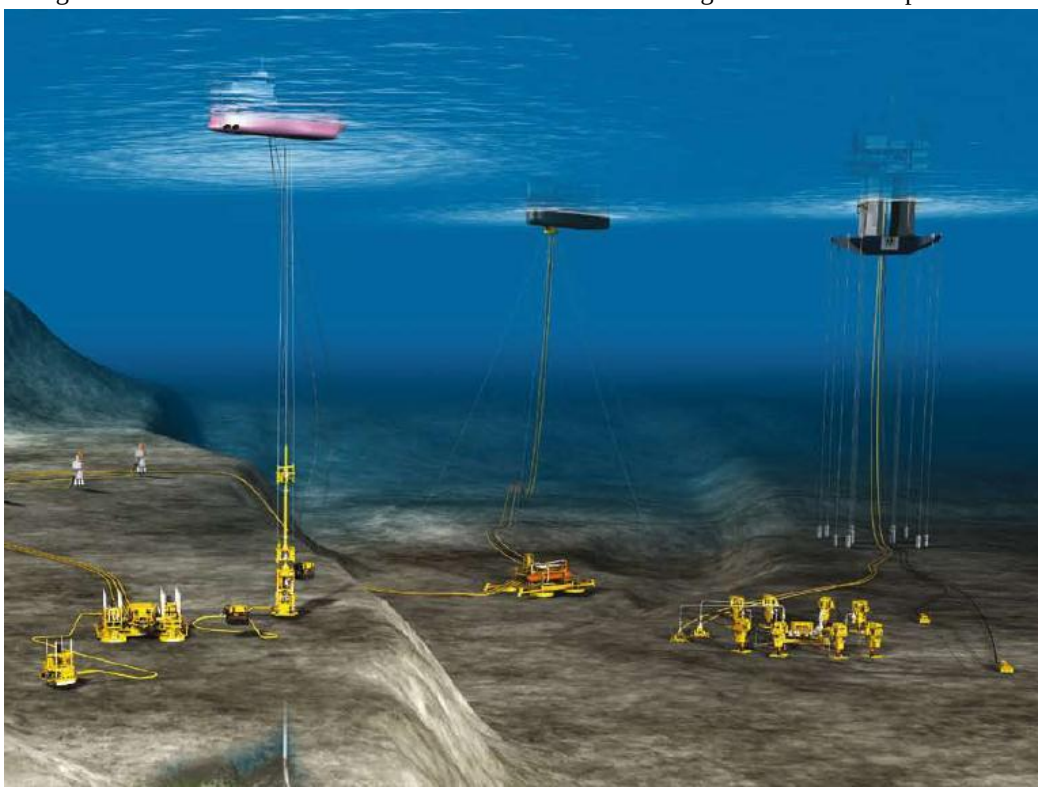
CONTENTS

1 INTRODUCTION

The application of flexible risers in the oil and gas industry has increased substantially in recent years, mainly due to the great industrial demands, such as the constant need for oil extraction from the seabed. Concurrently, environmental and ecological concerns have arisen, requiring the development of riser structures that are technologically more flexible and that could offer high safety levels to prevent accidents from occurring, as argued by Albertazzi and Viotti (2019) in [1].

Fundamentally, flexible risers are special pipelines used in the oil and gas industry to connect ocean floor production units to floating units or platforms as shown in Figure 1.1.

Figure 1.1 - Illustrated installations of flexible risers connecting the seabed to the platforms

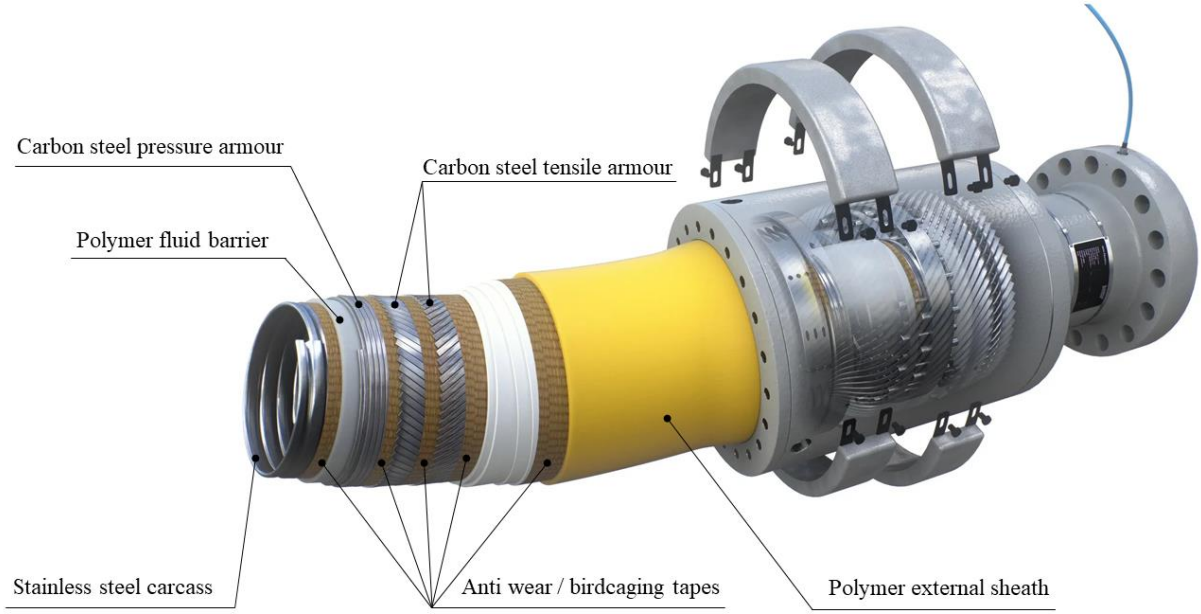


Source: MPCNEWS (2010) [2]

The risers are composed of multiple layers, which all have their own structural and chemical properties as shown in Figure 1.2. The assembled structure of layers in flexible pipelines allows the best properties of each constituent to be combined to provide a pipeline suitable for a particular application. One of the layers is the tensile armour, formed by multiple wire helixes with rectangular cross sections to maintain flexibility. It is suspected that the level

of residual stresses presented in the tensile armour wires has a significant impact on the service life of risers, as suggested by VIVIANI (2020) in [3].

Figure 1.2 - Layers of the flexible riser.



Source: adapted from NOV Inc. (2022) [4]

When in service, the flexible risers are subjected to cyclic loads such as bending, tensile, and internal stress variation. This draws more attention to operational safety in terms of fatigue life. Furthermore, the extreme metal shaping used to make flexible riser armour wires has a negative impact on residual stress levels. That is, the armour wires go through a series of manufacturing procedures, making it difficult to determine the residual stress levels. They are helically wounded around the riser, which adds additional stress to the component's stress levels.

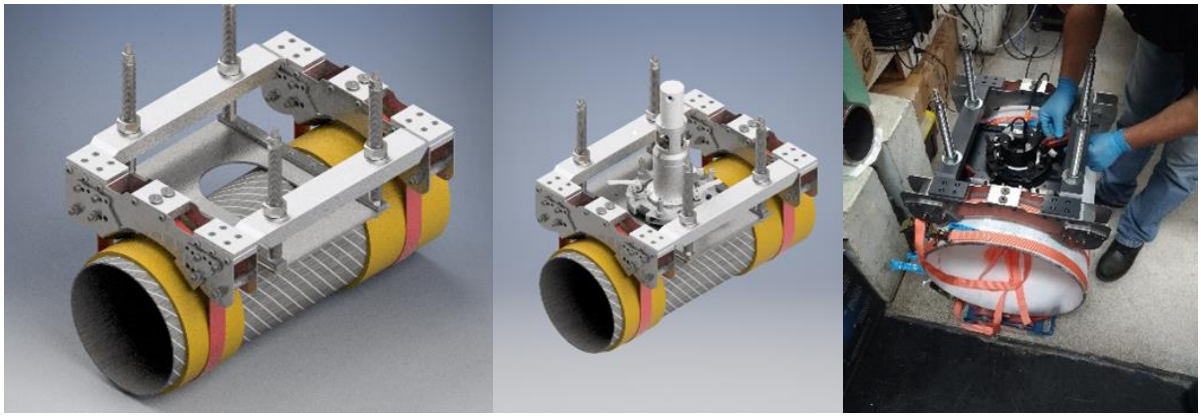
The flexible riser's endurance could be improved by lowering the amounts of undesired residual stresses in the armours. Therefore, there is a demand for reliable techniques for measuring residual stress levels in these components.

In that context, one of the major purposes of the cooperation between The Brazilian Petroleum Corporation (PETROBRAS) and the Metrology and Automatization Laboratory (LABMETRO) of the Federal University of Santa Catarina (UFSC) is to better understand how residual stresses are introduced during the manufacturing process of flexible risers.

The present undergraduate thesis is inserted in a major LABMETRO and PETROBRAS' cooperation project known as TRRiFlex, where the project aims to develop, compare, validate and apply techniques and equipment to measure residual stresses in flexible risers' reinforcement layer in order to characterize the state of stresses at different stages of its manufacturing process.

The midstream businesses focused on the transportation of oil, natural gas, and derivatives depend on reliable techniques for measuring residual stresses. PETROBRAS currently uses the Hole-Drilling Method (HDM) to perform residual stress measurements in pipelines, where this technique is employed to evaluate the stress state of the pipelines [5]. In Figure 1.3 is showed an example of a fastening system employed in the measurement of residual stresses in one of the armour layers of the flexible risers.

Figure 1.3 - Example of a fastening system for measuring residual stresses in flexible risers.



Source: TRRiFlex project (2019) [1]

Basically, the HDM consists of measurements of strains near a relatively small hole drilled by step increments in metallic structures, where the stress relief around the hole results in deformations. The strains around the hole are usually measured with placed strain gauges.

The stresses at the location of the hole are calculated by the Integral Method (IM), which is a well-known procedure for solving the inverse problem of determining the residual stresses from measured surface strain, based on the elastic assumption of the component material [6].

For the proper calculation of the stresses via IM, required calibration data is obtained from finite element simulations, assuming linear elastic material behavior. That assumption gives limitation to the HDM for measuring residual stresses well bellow the yield strength. The

stress concentration effect around a blind hole generates plastic deformations when the component is subjected to a high residual stress level. Exceeding the elastic limit can cause discrepancies from the real stress value, that is, the plastic deformation could mask the true value of the stresses, which is generally overestimated, with stresses values exceeding the yield stress [5]. Such high levels of residual stresses are often introduced in the armour wires of the flexible risers.

There are methodologies available to correct the residual stress measurements considering the plasticity effect in the uniform method. For the nonuniform method, the corrections available are limited to the equibiaxial stress state [5]. As suggested by Chupakhin (2017) in [7], there is a lack of mathematical models for the correction of the plasticity effects in the HDM for measuring high levels of residual stress. Thus, the purpose of this undergraduate thesis in on the development and the implementation of an Artificial Neural Network (ANN) model for correcting plasticity effects that are limited by the hole drilling method, the Standard Test Method for Determining Residual Stresses by the Hole-Drilling Strain-Gage Method (ASTM E837 – 13a) [6].

In this context, given the limitations of the ASTM E837 – 13a [6] in the plastic regime, the developed model seeks to correct the plasticity effects by creating a model based on the data resulting from elastoplastic numerical simulations in order to correct the residual stress values

1.1 OBJECTIVES

In this section, the general and specific objectives of this academic work will be presented in detail.

1.1.1 General objective

The objective of this undergraduate thesis is to develop and implement a Deep Artificial Neural Network (ANN) model, based on numerical simulations data, for the correction of plasticity effects in the HDM, that is, the Standard Test Method for Determining Residual Stresses by the Hole-Drilling Strain-Gage Method, used for measuring high levels of residual stress in flexible risers.

1.1.2 Specific objectives

A few specific objectives were defined to achieve the general objective:

Deepen the study of the ASTM E837 – 13a [6] and its limitations regarding the effects of plasticity through a Literature review, as well as a reproduction of similar models found in the literature.

Develop a Finite Element (FE) model in Ansys Mechanical APDL for the execution of the needed numerical simulations, considering the most appropriate boundary conditions for working with elastoplastic analysis available in the literature.

Execute several numerical simulations with the developed FE model and compute the summarized numerical results values via the IM.

Analyze, filter, and establish different datasets from the summarized results in order to build the Deep ANN model.

Train, validate, test, and implement the Deep ANN model for the correction of plasticity effects in the HDM.

Analyze the model correction performance by means of comparison with a classic multivariate linear regression model and implement the necessary changes to the final model.

2 STATE OF THE ART

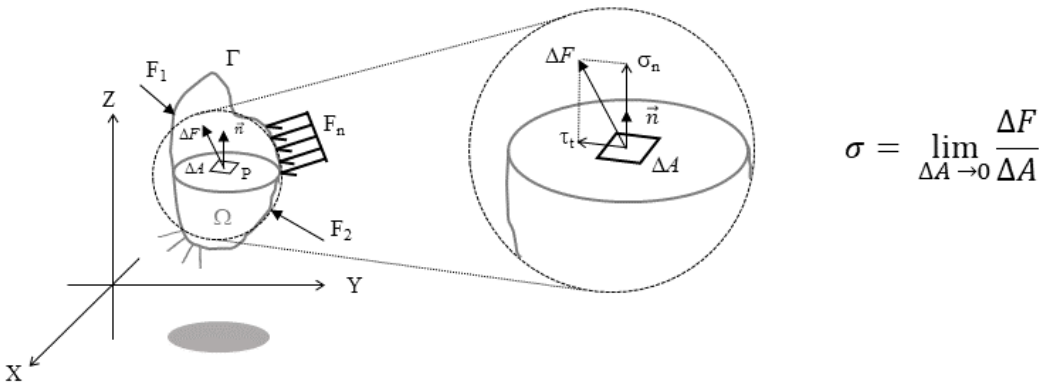
In this chapter, an overview about residual stresses and the fundamentals of the hole drilling method will be presented, together with the limitations regarding the measurement of high levels of residual stresses. The principles of the Finite Elements Method will be reviewed, along with the boundary conditions employed for the casing-load of the flexible risers. Models' strategies already employed by other authors for the correction of plasticity effects in the HDM will be presented and the application of Deep ANN will be also discussed.

2.1 RESIDUAL STRESS

The term stress, represented by the symbol (σ), is used to express the loading of an object in terms of force applied to a specific cross-sectional area. From the standpoint of loading, stress is the applied force or system of forces that tends to deform a body. From the perspective of what is happening within it, stress is the internal distribution of forces within a body that balance and react to the loads given to it. Depending on the nature of the loading scenario, the stress distribution may or may not be uniform.

Many engineering computations and material property determinations use simplifying assumptions to portray stress as a vector quantity. Typically, the term "vector" refers to a quantity with a "magnitude" and a "direction" [8]. In the larger scale detail in the middle of the Figure 2.1, it is possible to see the decomposition of the force vector over the area into normal and tangential components.

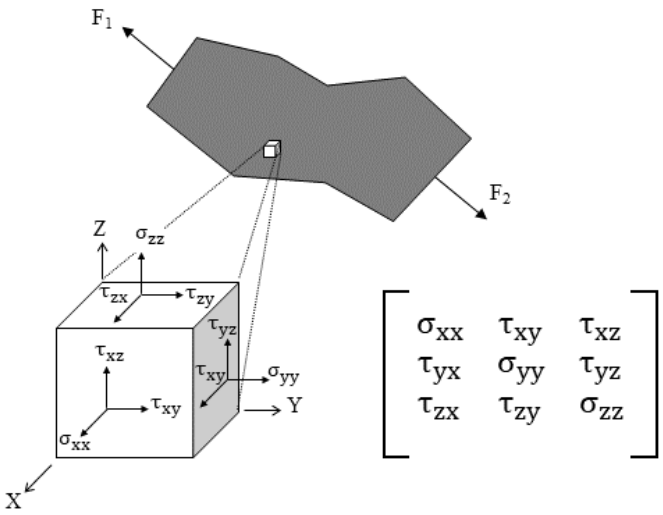
Figure 2.1 - Schematic representation of a solid loaded with external forces and constrained by boundary conditions.



Source: elaborated by the author.

It should be noted that the stresses in most 2-D or 3-D solids are considerably more complex and require a more meticulous definition [8]. Internal forces operating on a tiny area of a plane can be broken down into three components: one normal to the plane and two parallel to the plane. The normal stress (σ) is calculated by dividing the normal force component by the area, while the shear stress (τ) is calculated by dividing the parallel force component by the area. Because the area is finite, these stresses are average stresses, but when the area is permitted to approach zero, the stresses become point stresses.

Figure 2.2 - Stress state in a point of a 3-D solid, represented by a tensor.



Source: elaborated by the author.

Because stresses are defined in reference to the plane that passes through a point, and the number of such planes is infinite, a point appears to have an unlimited set of stresses. Fortunately, the stresses on any plane can be calculated using the stresses on three orthogonal planes passing through the point [8]. Since each plane has three stresses, the stress tensor comprises nine stress components, which entirely characterize the state of stress at any point inside a material, as shown in Figure 2.2. Given the fact that the stress tensor matrix is proven to be symmetric by Cauchy´s equation (due the principle of conservation of angular moment), only six components are considered independent stresses values. The components are represented by (σ_{ij}), where (i) is the direction that the stress is acting and (j) is the face the stress is acting on. Furthermore, the unit of stress in the International System (SI) is the Mega Pascal (MPa).

Strain is a system's response to applied stress. When a material is subjected to a force, it generates stress, which causes the substance to deform. The amount of deformation in the direction of the applied force divided by the starting length of the material is known as engineering strain. As with stress, the strain distribution in a complex structural member may or may not be uniform, depending on the nature of the loading condition [8].

If the stress is mild, the material will only stretch slightly and will revert to its normal size once the force is eliminated. This is known as elastic deformation because, like elastic, it returns to its original state after being stressed. Elastic deformation occurs only when tensions are less than a critical stress known as the yield strength. If a material is loaded above its elastic limit, it will stay distorted after the load is withdrawn. This is known as plastic deformation.

Stresses and strains are linked by material properties. For a linear case, it is possible to define material constants, such as Young's modulus or modulus of elasticity and Poisson's ratio, which are connected to the stiffness of the material and relative deformations in the three directions, x, y, and z. More information on generic situations of completely anisotropic materials, i.e. with distinct Young's Modulus and Poisson's ratio in each direction, can be found in Popov (1990) [9]. The relationships between strain and stress in the case of a linear isotropic material are given by Eq. (2.1)-(2.4).

$$\sigma_{xx} = \left[\frac{E}{(1+\upsilon)(1-2\upsilon)}\right] [(1-\upsilon)\varepsilon_{xx} + \upsilon(\varepsilon_{yy} + \varepsilon_{zz})] \tag{2.1}$$

$$\sigma_{yy} = \left[\frac{E}{(1+\upsilon)(1-2\upsilon)}\right] [(1-\upsilon)\varepsilon_{yy} + \upsilon(\varepsilon_{xx} + \varepsilon_{zz})] \tag{2.2}$$

$$\sigma_{zz} = \left[\frac{E}{(1+\upsilon)(1-2\upsilon)}\right] [(1-\upsilon)\varepsilon_{zz} + \upsilon(\varepsilon_{yy} + \varepsilon_{xx})] \tag{2.3}$$

$$\tau_{xy} = \frac{E}{2(1+\upsilon)}\gamma_{xy} \qquad \tau_{yz} = \frac{E}{2(1+\upsilon)}\gamma_{yz} \qquad \tau_{zx} = \frac{E}{2(1+\upsilon)}\gamma_{zx} \tag{2.4}$$

Residual stress can be defined as "the stress that remains inside a component or structure after all applied forces have been removed". Tensile residual tension pulls the material apart, whereas compressive residual stress pushes it together. Compressive stress is considered mathematically negative, while tensile stress is considered positive.

The residual stresses are formed when a material reaches equilibrium following plastic deformation induced by applied mechanical loads, thermal loads, or phase transitions. Mechanical and thermal treatments performed to a component while in service may potentially change its residual stress state. Examples of processes that can cause residual stress:

1. Plastification of a material during machining - mechanical load.
2. Difference in solidification of the material, for example, in a cooling casting - thermal load.
3. Precipitation/phase transformation resulting in a volume change, for example, austenite to martensite - phase transition.

The total stress experienced by the material at a given location within a component is equal to the residual stress plus the applied stress, as emphasized in Eq. (2.5).

$$\text{TOTAL STRESS} = \text{RESIDUAL STRESS} + \text{APPLIED STRESS} \quad (2.5)$$

To identify the actual loads experienced by a component, it is necessary to understand the residual stress condition. In general, compressive residual stress in a component's surface is advantageous [10]. It has the ability to improve fatigue strength and fatigue life, decrease crack propagation, and increase resistance to environmental assisted cracking such stress corrosion cracking and hydrogen induced cracking [10]. On the other hand, tensile residual stress at the component's surface is generally undesirable because it reduces fatigue strength and fatigue life, promotes crack propagation, and reduces resistance to environmental assisted cracking.

The scale at which residual stresses occur within a material can be used to characterize them. Macro-stresses are stresses that occur over vast distances within a material, whereas micro-stresses are stresses that exist only locally (either between grains or within a grain). The sum of all these three types of stresses at a specific place within a material is considered the total residual stress [10].

The residual stresses of type I are the macro-stresses that occur over long distances involving numerous grains inside a material. Diversely, the residual stresses of type II are the micro-stresses induced by changes in a material's microstructure that occur across distances comparable to the grain size of the material. These can occur in single-phase materials because

of anisotropic behavior of individual grains, or it can happen in multi-phase materials because of the presence of various phases. Additionally, the residual stresses of type III exist within a grain as a result of crystal defects.

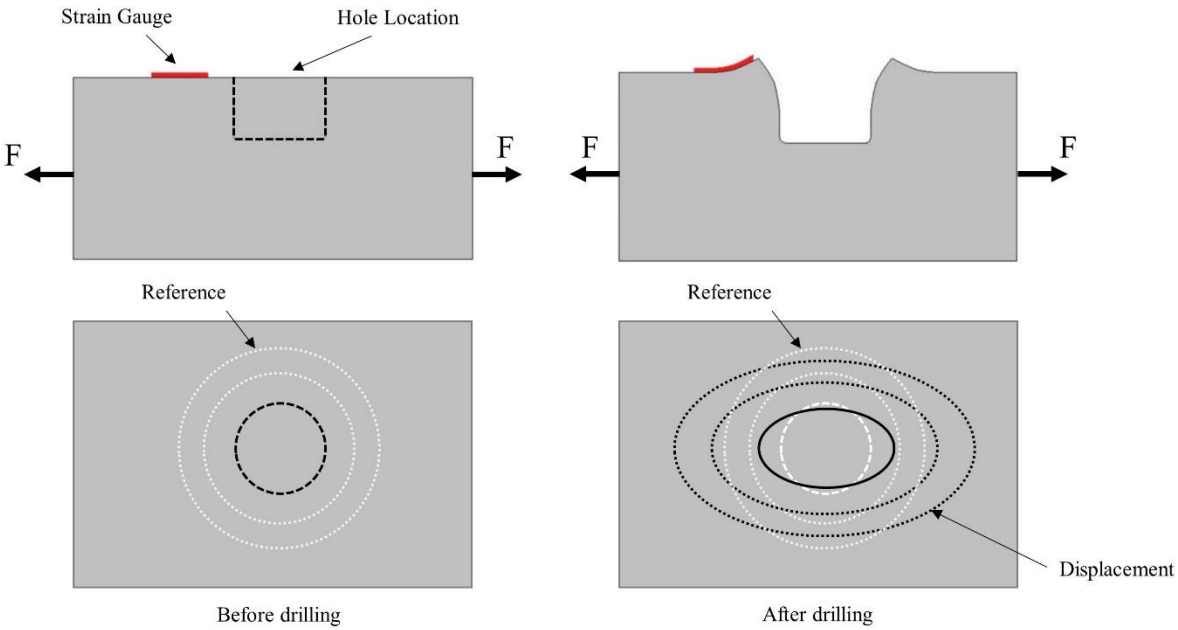
Measuring residual stress has several advantages, such as reducing catastrophic failures by ensuring safety, extend the life of a component or structure by ensuring that sufficient compressive residual stress exists. Tracking residual stress degradation allows for more accurate replacement requirements and increases the likelihood of detecting alternative non-destructive procedures, validating the residual stress distribution derived from FE models and fracture mechanics.

2.2 THE HOLE-DRILLING METHOD

The hole drilling method is a well-known technique for measuring residual stresses in metallic structures that has gained great attention in the last decades. This is due to its ease of use in many applications, minimal damage to the target specimen, general reliability, and acceptable accuracy [7]. The HDM is categorized as a semi-destructive technique with normalized procedures described in the standard ASTM E837 – 13a [6].

Essentially, the experiment procedure involves attaching a strain gauge rosette to the surface of an isotropic linear-elastic material and drilling a blind hole or a through-hole in a predefined number of steps at the geometric center of the rosette, which causes a residual stress redistribution and strain relaxation in the surrounding area of the hole as shown in Figure 2.3.

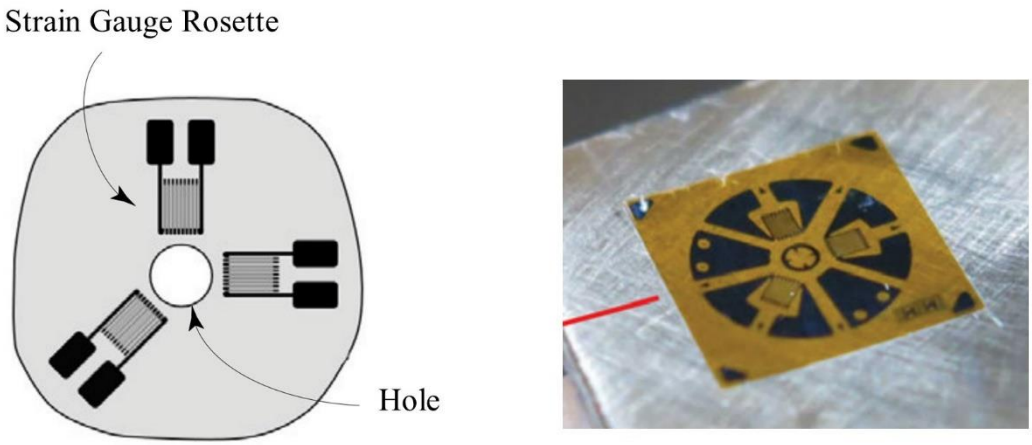
Figure 2.3 – Scheme showing a work piece under uniaxial loading before and after the hole drilling process.



Source: adapted from SCHAJER and WHITEHEAD (2018) [11].

The resulting relieved strains near the surface are measured with a strain gauge rosette as represented in the Figure 2.4. Afterwards, a series of equations is used to compute the residual stresses within the removed material based on the measured strains.

Figure 2.4 – Example of a Strain Gauge rosette placed on the surface of a work piece.

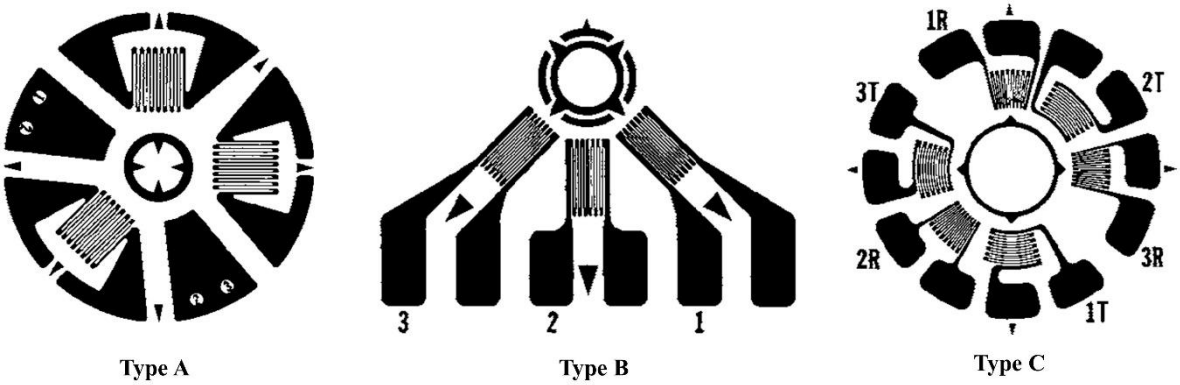


Source: adapted from SCHAJER and WHITEHEAD (2018) [11].

Rosettes of various types can be used to achieve better sensitivity and precision. A few of them, however, are standardized rosettes used in the HDM as displayed in Figure 2.5. The

standard offers calibration coefficients for each type of these rosettes. These calibration coefficients are then employed to compute the residual stress values.

Figure 2.5 - Strain gauge rosettes employed in the HDM.



Source: ASTM E837 13a [6].

Strain gauges are exclusively employed to measure strains in experiments. However, it is possible to acquire strain data from numerical simulation models, accordingly to works of [7, 12, 15, 16]. In these cases, strain gauges are commonly modeled considering individual grids. The strain measured by each grid is simplified by taking the displacements of its two ends into account. Furthermore, the overall strain response of the gauge accounts for the contribution of each grid. As a result, the total strain response is given by Eq. (2.6), as suggested by [11].

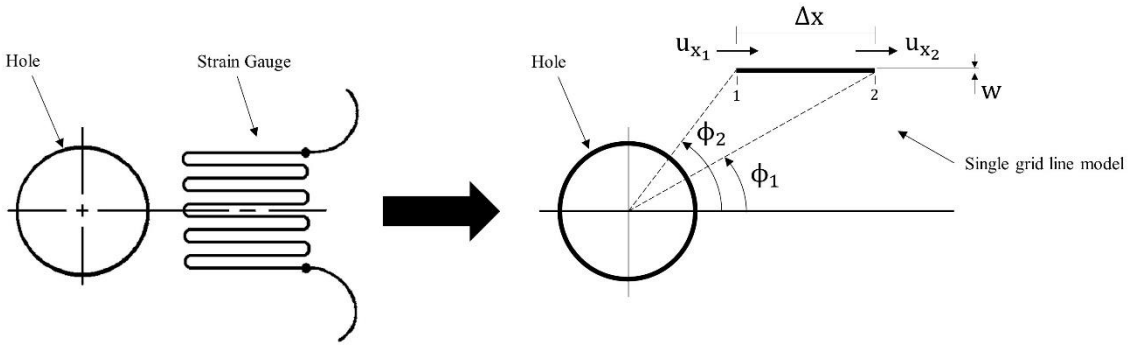
$$\varepsilon = \frac{\sum \frac{(u_{x_2} - u_{x_1})}{w}}{\sum \frac{\Delta x}{w}} \tag{2.6}$$

$$u_{x_1} = u_{r_1}\cos\phi_1 - u_{\theta_1}\sin\phi_1 \tag{2.7}$$

$$u_{x_2} = u_{r_2}\cos\phi_2 - u_{\theta_2}\sin\phi_2 \tag{2.8}$$

Where u_{x_1} in Eq. (2.7) and u_{x_2} in Eq. (2.8) give the displacements at both ends of the grid. The radial and tangential displacements are represented by u_r and u_θ, respectively. Furthermore, x symbolizes the grid's length and w its width. The Figure 2.6 illustrates the local angles ϕ_1 and ϕ_2.

Figure 2.6 - Model of a single grid line.



Source: elaborated by the author.

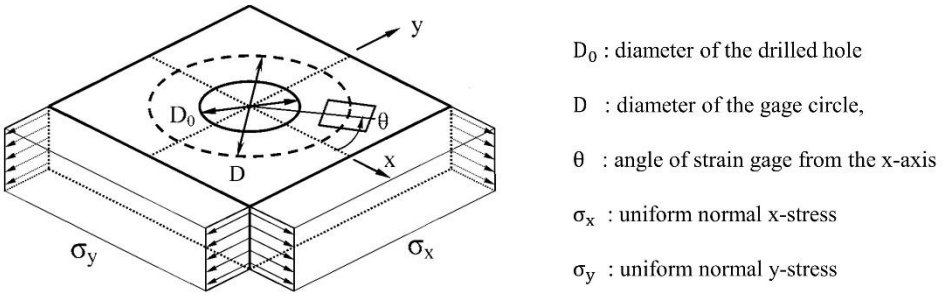
In addition to strain gauges, there are optical methods that are commonly used to capture the strains or the displacements in the HDM.

2.2.1 Stress calculation

As mentioned before, the stresses at the hole location are computed from the experimentally measured strains. This approach takes into account the mathematical relationships based on elasticity theory seen on Section 2.1. Essentially, there are two calculating methods procedures described in the standard ASTM E837 – 13a [4]: the uniform and the non-uniform.

The uniform approach calculates the average of the stresses along the length of the hole, as shown in Figure 2.7.

Figure 2.7 - Hole Geometry and Residual Stresses: Uniform Stresses.



Source: adapted from ASTM E837 13a [6].

This technique requires the use of Eq. (2.9)-(2.11) to calculate the combination strains p, q, and t. The relieved and measured strains by each strain gauge during the experimental technique are represented by ε_1, ε_2, and ε_3. The combination stresses P, Q, and T must then be obtained using Eq. (2.12)-(2.14).

$$p = \frac{(\varepsilon_3 + \varepsilon_1)}{2} \tag{2.9}$$

$$q = \frac{(\varepsilon_3 - \varepsilon_1)}{2} \tag{2.10}$$

$$t = \frac{(\varepsilon_3 + \varepsilon_1 - 2\varepsilon_2)}{2} \tag{2.11}$$

$$P = -\frac{E}{(1+v)} \frac{\sum(\bar{a} \cdot p)}{\sum(\bar{a}^2)} \tag{2.12}$$

$$Q = -E \frac{\sum(\bar{b} \cdot q)}{\sum(\bar{b}^2)} \tag{2.13}$$

$$T = -E \frac{\sum(\bar{b} \cdot t)}{\sum(\bar{b}^2)} \tag{2.14}$$

The P, Q, and T actually represent the uniform equi-biaxial stress, the uniform 45º shear stress, and the uniform x-y shear stress, respectively. The ASTM standard tabulates the calibration coefficients $\bar{a}$ and $\bar{b}$. They represent the strain relief generated by the removal of material at each increment with a unit tension and their values are dependent on the rosette type and the hole diameter.

In order to obtain maximum (σ_{max}) and minimum (σ_{min}) stresses the Eq. (2.15) can be used. The angle between σ_{max} and the strain gauge ε_1 is denoted by β and can be calculated using Eq (2.16). Finally, using Eq. (2.17)-(2.19), the axial and shear stresses can be calculated.

$$\sigma_{max}, \sigma_{min} = P \pm \sqrt{Q^2 + T^2} \tag{2.15}$$

$$\beta = \frac{1}{2} \operatorname{atan}\left(\frac{-T}{-Q}\right) \tag{2.16}$$

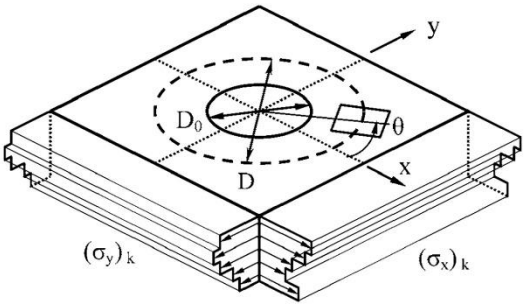
$$\sigma_x = P - Q \tag{2.17}$$

$$\sigma_y = P + Q \tag{2.18}$$

$$\tau_{xy} = T \tag{2.19}$$

The non-uniform technique on the other hand generates a stress profile that has a single stress value for each increment (step k) as illustrated in Figure 2.8. Prior to the discovery of the non-uniform approach, the uniform method was widely used. The non-uniform technique actually originated from an enhancement in the computing procedure [5].

Figure 2.8 - Hole Geometry and Residual Stresses: Non-uniform Stresses



D_0 : diameter of the drilled hole

D : diameter of the gage circle,

θ : angle of strain gage from the x-axis

$(\sigma_x)_k$: normal x-stress within hole depth step k

$(\sigma_y)_k$: normal y-stress within hole depth step k

Source: adapted from ASTM E837 13a [6].

The integral method is used to compute the non-uniform stresses by using strain information for each increment. The combination strain is obtained for each increment with the strain values ε_1, ε_2, and ε_3, in accordance with Eq. (2.20)-(2.22). The subscript j reflects the most recent hole depth increment, whereas the subscript k specifies the hole depth increment sequence number (step k).

$$p_j = \frac{(\varepsilon_3 + \varepsilon_1)_j}{2} \tag{2.20}$$

$$q_j = \frac{(\varepsilon_3 - \varepsilon_1)_j}{2} \tag{2.21}$$

$$t_j = \frac{(\varepsilon_3 + \varepsilon_1 - 2\varepsilon_2)_j}{2} \tag{2.22}$$

$$\bar{a}\,\mathbf{P} = \frac{E}{(1+v)}\mathbf{p} \tag{2.23}$$

$$\bar{b}\,\mathbf{Q} = E\mathbf{q} \tag{2.24}$$

$$\bar{b}\,\mathbf{T} = \mathrm{E}\mathbf{t} \tag{2.25}$$

$$P_k = \frac{(\sigma_y)_k + (\sigma_x)_k}{2} \tag{2.26}$$

$$Q_k = \frac{(\sigma_y)_k - (\sigma_x)_k}{2} \tag{2.27}$$

$$T_k = (\tau_{xy})_k \tag{2.28}$$

$$\bar{a}\,\mathbf{P} = \frac{E}{(1+v)}\mathbf{p} \rightarrow \tag{2.23}$$

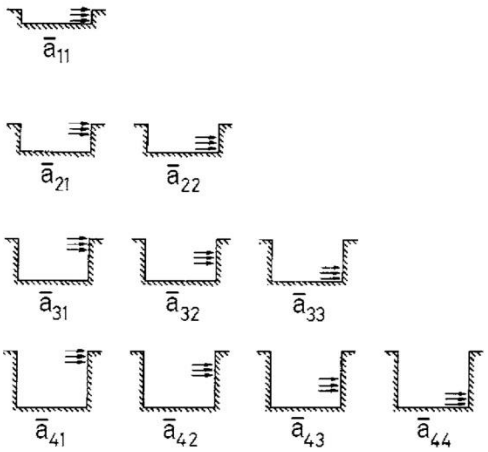
$$\rightarrow \begin{bmatrix} a_{11} & \ddots & \\ \vdots & & \\ a_{1k} & \cdots & a_{jk} \end{bmatrix} \begin{bmatrix} (\boldsymbol{\sigma_y})_1 + (\boldsymbol{\sigma_x})_1 \\ \vdots \\ (\boldsymbol{\sigma_y})_k + (\boldsymbol{\sigma_x})_k \end{bmatrix} \frac{\mathbf{1}}{\mathbf{2}} = \frac{E}{(1+v)} \begin{bmatrix} (\boldsymbol{\varepsilon_3} + \boldsymbol{\varepsilon_1})_1 \\ \vdots \\ (\boldsymbol{\varepsilon_3} + \boldsymbol{\varepsilon_1})_j \end{bmatrix} \frac{\mathbf{1}}{\mathbf{2}} \tag{2.29}$$

Solving three matrix equations in Eq. (2.23)-(2.25) is possible to obtain the combined stresses for each hole depth. The combination stresses **P**, **Q**, and **T** may be expressed as a function of the axial stresses using Eq. (2-26)-(2-28).

In contrast to the uniform method, these equations are defined by vectors and matrices. For a modest number of increments, this approach works effectively. The calibration coefficient matrices become numerically ill-conditioned for a large number of increments (for example, 10 or 20 increments) [6]. Large mistakes in the stress levels originate from small errors in the measured strains. To limit the mistakes caused during the calculating process, the standard suggests using Tikhonov regularization [6]. More details of the Tikhonov regularization procedure can be found in Appendix A.

The cross-section of the hole can be used to explain the physical representation of the calibration coefficients. In Figure 2.9, a hole-drilling operation performed in four increments illustrates the physical meaning of the calibration constants $\bar{a}_{jk}$ and $\bar{b}_{jk}$. These calibration constants are dimensionless and represent the strain relief in a hole with j increments (total depth) and a unity pressure applied at the increment k (step k).

Figure 2.9 - Physical Interpretation of Coefficients $\bar{a}_{jk}$.



Source: ASTM E837 13a [6].

2.3 FINITE ELEMENT METHOD

The Finite Element Method (FEM) or the Finite Element Analysis (FEA) is a numerical method for solving (partial) differential equations and it is widely employed in many fields with the purpose of finding approximate solutions to real continuous problems. Therefore, it is generally applied to problems where there is no simple analytical solution [12]. FEM was first developed to solve problems related to structural analysis and then gained space in other research fields.

In a generic way, the method consists of partitioning the domain of the unknown function of the (partial) differential equations related to the physical problem, and then associating shape functions to the partitions, which by combining them with the unknown variables of the physical problem constitute an approximate solution of the desired physical quantity [12].

A finite element analysis (FEA) can be subdivided into four main steps [12]:

1. The discretization of the continuum by dividing the domain into subdomains or finite elements, which requires the definition of the element shape, number of nodes, and the interpolation function (shape function).

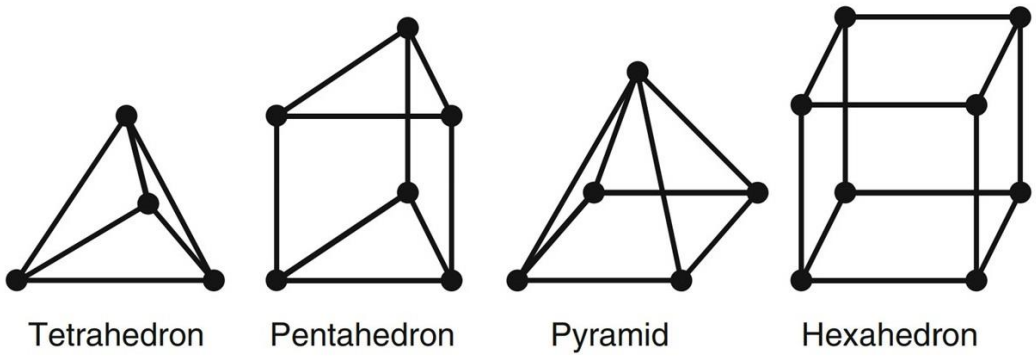
2. The determination of the element equations that describe the relationship between the primary unknowns, such as displacement, and the secondary unknowns, such as stress and strain.

3. The Combination of the element equations into a global matrix, taking into account the boundary and loading conditions.

4. The solution of the system of equations to obtain the numerical simulation results.

There are different types of elements with 1,2 or 3 dimensions that can be chosen according to the needs of the analysis. The Figure 2.10 shows some commonly used 3D elements. In initial simulations, it is reasonable to consider a larger element size to avoid unnecessary expenditure of time and computational resources. However, the element size significantly affects the accuracy of the simulation results and, consequently, more refined meshes lead to better results. To speed up the solution, it is common to implement different mesh densities according to the local stress and displacement gradients [12]. The influence of the element size can be identified with a mesh refinement study while keeping the other parameters constant.

Figure 2.10 - Shape of commonly used 3D elements



Source: adapted from KLOCKE (2011) [12].

Aside from the standard solid volumes, commercial software offers a vast array of aspects. Hole drilling researchers use either bidimensional or tridimensional elements depending on their needs. For full three-dimensional models, applying the loads at the wall requires the use of surface elements in the Ansys Mechanical APDL, a commercial software for structure FE analysis [13]. The ‘surface traction load’ allows the application of a general

load freely chosen which can also be a shear – condition required for the calculation of the calibration coefficients, mentioned in the previous section.

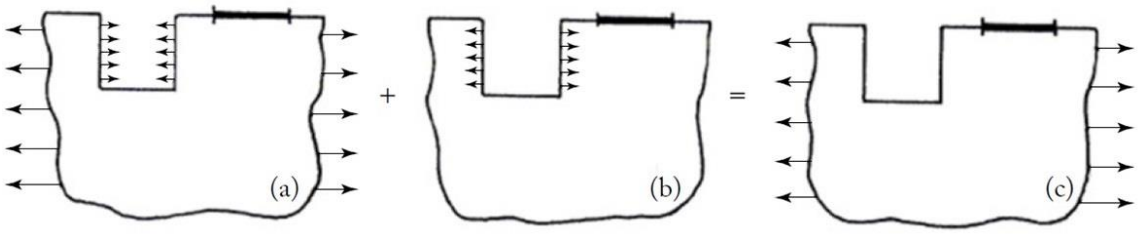
Furthermore, the numerical simulation of a FE model can be considered linear or nonlinear. A linear numerical simulation analysis is related to linear elastic material and small displacements (infinitesimal strain theory), whereas a non-linear numerical simulation analysis is related to large displacements and elastoplastic materials, therefore in the latter case the superposition effect cannot be applied. Real physical problems are always nonlinear, in a way that linear analyses are essentially simplifications of specific cases with reliable results.

There are two modelling approaches for numerically obtaining the strains around the hole available in the literature [5]:

a) applying the stresses at the model's boundaries and subtracting the model with the hole from the model without the hole, or

b) applying the stresses directly at the hole wall using the superposition principle (as shown in Figure 2.11.c. This principle is based on the premise that the material is elastic. Due to the complexity of the non-uniform approach, the loads are applied directly to the hole wall.

Figure 2.11 - Superposition principle employed to evaluate effect of hole-drilling stress relief: a) stress state before the hole drilling, b) stress relaxation after the drilling, and c) stress state after the drilling.



Source: SCHAJER and WHITEHEAD (2018) [11].

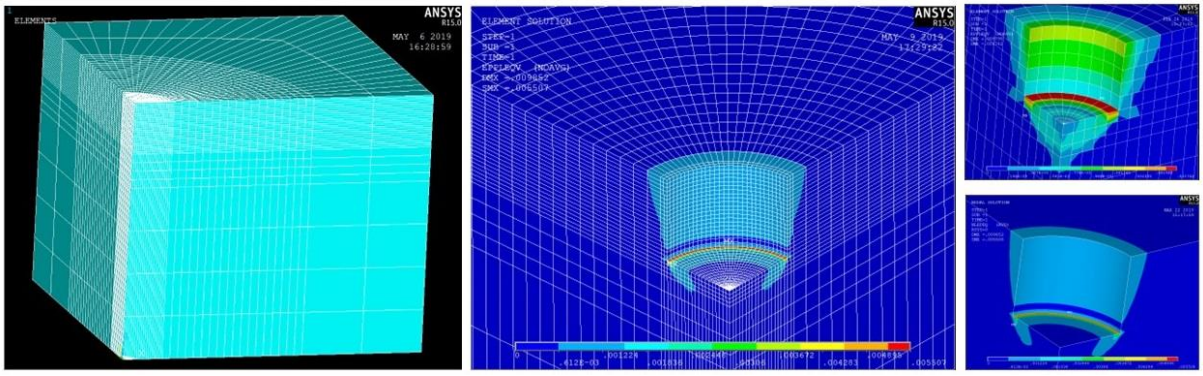
To obtain the coefficient matrices $\bar{a}$ and $\bar{b}$, two load cases must be applied. The calibration matrix for isotropic stresses (illustrated in Figure 2.11.a) is obtained by using an equi-biaxial stress condition. Whereas the calibration matrix for 45° shear stresses (illustrated in Figure 2.11.b) needs the use of a 45° shear case ($\Omega = -1, \sigma_1 = -\sigma_2$). The application of the condition at the model's boundary is simple. The 45° shear case, on the other hand, must be meticulously applied at the hole wall. This stress state is generated by the application of two distinct loads at the hole wall: a radial load varying with cos(2θ) and a tangential load varying

with sin(2θ). The stress state can then be obtained by using the Mohr circle to transform the stresses at the borders [5].

The loads are applied based on their position in the coefficient matrix. Each number (a_{ij} or a_{kj}) in the matrix refers to a specific section of the wall where stress is applied. The increment where the load is applied is represented by the row subscript (k or i), and the depth of the hole is represented by the column subscript (j), as demonstrated by Eq. (2.29) in Section 2.2.1.

For the case of a three-dimensional simulation model, considering an elastoplastic model, the Hole drilling researchers usually develop a three-dimensional mesh. Due to the symmetry conditions and in order to decrease the processing time, only a quarter of the geometry is commonly simulated. The Figure 2.12 shows an example of a meshed 3D model developed by researchers of LABMETRO. It can be seen that the model presents a higher density inside the hole as well as around it in order to improve the calculation of the plastic deformation involved [14].

Figure 2.12 – Example of a meshed model used in a numerical simulation of the HDM for the obtention of the calibration coefficients.



Source: ALBERTAZZI and VIOTTI (2020) [14]

The displacements are recorded on the surface near the hole's rim. The FE software is typically used to obtain the displacements, which are then converted to an approximate strain value measured by each grid line of the strain gauge.

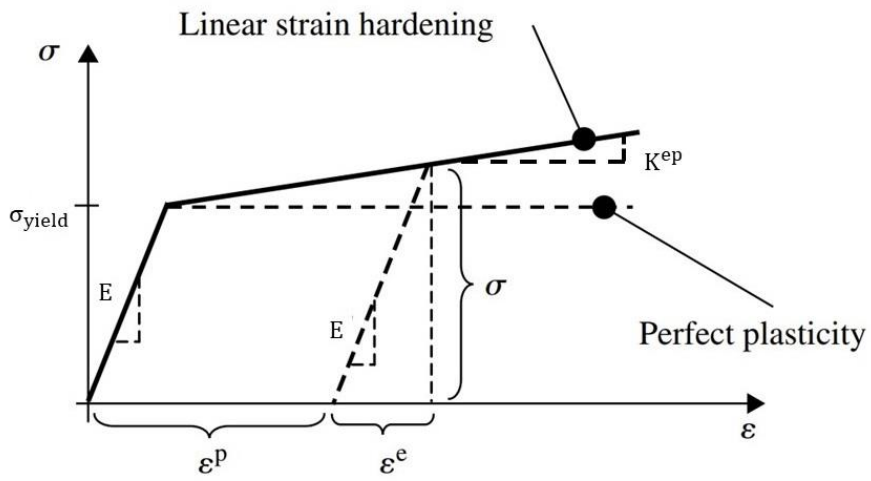
2.3.1 Plasticity effect

The stress σ and the strain ε have a nonlinear relationship in elastoplastic analysis. Since this relationship is dependent on the strain's history of the material, external forces can

be applied incrementally to identify the variation of strains and stresses. This form of numerical analysis is solved iteratively and belongs to the nonlinear category of numerical simulations (time-independent and rate-dependent) [5].

In this context, considering a stress-strain curve generated by a purely uniaxial tensile test shown in Figure 2.13, the material strain hardens after the σ_{yield} (yield stress) under a uniaxial stress. After this point, plasticity effects begin to occur in the material (ε^P). The strains were merely elastic (ε^e) before the plasticity effects began. As indicated in Eq. (2.30), the total strain (ε) can be split into two components. This relationship is known as the classical additive decomposition of strain [15]. Therefore, considering both elastic and plastic strains, the stresses can be obtained according to Eq. (2.31).

Figure 2.13 - Stress-strain curve of a uniaxial tensile test for a bilinear hardening material model



Source: adapted from DUNNE and PETRINIC (2006) [15]

$$\varepsilon = \varepsilon^e + \varepsilon^p \tag{2.30}$$

$$\sigma = E\,\varepsilon^e = E\,(\varepsilon - \varepsilon^p) \tag{2.31}$$

Plastic strain is equal to zero ($\varepsilon^P = 0$) for working conditions within the elastic regime. In processes that cause large strains, such as forging and forming, it is reasonable to assume $\varepsilon^p \approx \varepsilon$ and $\varepsilon^e \approx 0$ [15]. The yield condition criteria for multiaxial stress is based on many approaches. The Tresca and von-Mises criteria are two widely used techniques [5].

Isotropic and kinematic hardening models are commonly employed. Isotropic hardening implies that the yield surface changes size uniformly in all directions such that the yield stress increases in all stress directions as plastic straining occurs [7]. A few isotropic hardening models are available in the literature.

Bilinear isotropic hardening is a technique extensively used by HDM researchers [5]. In Figure 2.13, a typical stress-strain curve for the one-dimensional case of a bilinear isotropic hardening is shown. The strain is governed by the Young's Modulus or elastic modulus (E) until it reaches the yield stress (σ_{yield}). Following this point, the material exhibits elastoplastic behavior regulated by the tangent modulus (K^{ep}). A tangent modulus equivalent to the dotted line would represent completely plastic material behavior (no strain hardening) [5].

Following the execution of the hole, either stresses or displacements are recorded in the traditional measurement. Simultaneously, the geometry of the hole causes a stress concentration surrounding it. The stress concentration is considered lower in blind holes, however, with high loads, the stress near the blind hole surpasses the yield point, causing plastic stresses. The elastoplastic strain could be interpreted as elastic in case of unawareness of the stress level [5]. Under the elastic assumption, strains are converted to stresses. As a result, there are differences between the measured stresses and the real stresses that were previously operating on the workpiece, in a way that the closer the stresses are to the yield point, the bigger the deviations are [5].

Furthermore, the plastic yielding near the hole affects the measurement of high levels of stresses. The elastic assumption is no longer applicable in this scenario. Additionally, the coefficients presented in the standard do not apply to stress magnitudes greater than 80% of the yield stress for blind holes and 50% for through holes [5].

A few papers propose correction parameters for residual stress measurements at the yielding stress. The first paper addressing the plasticity repercussions in HDM computations presented a uniform measurement correction factor. To quantify the plasticity effect on the HDM residual stress results, Beghini et al. [16] proposed a dimensionless plasticity factor, represented by Eq. (2.32).

$$f = \frac{\sigma_{eq} - \sigma_{eq,i}}{\sigma_{yield} - \sigma_{eq,i}} \tag{2.32}$$

The σ_{eq} is the von Mises stress (for plane stress states), which takes biaxiality into account, $\sigma_{eq,i}$ is the equivalent residual stress producing the onset of plasticity in the 2D case, and σ_{yield} is the yield stress of the material. The $\sigma_{eq,i}$ can be expressed as a function of the ratio in Eq. (2.33) and the biaxiality ratio Ω by Eq. (2.34).

$$\sigma_{eq,i} = \sigma_{yield} \frac{\sqrt{1 - \Omega + \Omega^2}}{3 - \Omega} \tag{2.33}$$

$$\Omega = \frac{\sigma_y}{\sigma_x} \tag{2.34}$$

As a result, the plasticity effect can be quantified from $f = 0$, no plasticity, to $f = 1$, full plasticity, i.e., residual stress producing general yielding around the hole. Given the presence of a plane stress condition around the hole, this component theoretically predicts the commencement of yielding.

It is known that, for isotropic materials and biaxial stress states ($\sigma_z = 0$), the stress concentration factor around a through hole attains a minimum value of 2 for equibiaxial stresses ($\sigma_y = \sigma_x$), being equal to 3 for uniaxial stresses ($\sigma_y = 0$) and attains a maximum value of 4 for pure shear stresses ($\sigma_y = -\sigma_x$) [17]. For an equibiaxial stress condition, where $\sigma = \sigma_y = \sigma_x$, $\Omega = 1$, $\sigma_{eq,i} = 0.5\,\sigma_{yield}$ and $\sigma_{eq} = \sigma$, the Eq. (2.32) reduces to Eq. (2.35).

$$f = 2\left(\frac{\sigma}{\sigma_{yield}}\right) - 1 \tag{2.35}$$

2.3.2 Boundary conditions employed in the HDM.

To generate the displacements/strains around the hole, the HDM employs a few distinct layouts. Loads can be applied at the end of the plate (boundaries) or at the hole wall, according to the superposition concept, as mentioned before. The researchers' methodologies are usually influenced by the choice of linear or non-linear material behavior as well as the intended stress profile [5].

To calculate the coefficients under the elastic assumption, axisymmetric and harmonic elements are often used [5]. Some researchers, on the other hand, make use of 3D models as shown in Figure 2.12. These models, however, typically have a larger computational cost. The 3D models are also used when there are obstacles surrounding the hole or when the material behavior is not linear elastic [11]. It is feasible to work with plasticity in 2D axisymmetric models with an equibiaxial stress state, but for any other stress state, axisymmetric models are not employed in the literature when plasticity is considered [5].

On a 2D axisymmetric model, Nobre et al. [17] used the Ansys Mechanical APDL to construct a non-uniform equibiaxial stress state. It was achievable thanks to model constraints and a temperature gradient that varied with depth. Chupakhin et al. [7], on the other hand, applied the stresses directly to the hole wall of a 2D axisymmetric model.

2.4 ARTIFICIAL DEEP NEURAL NETWORKS

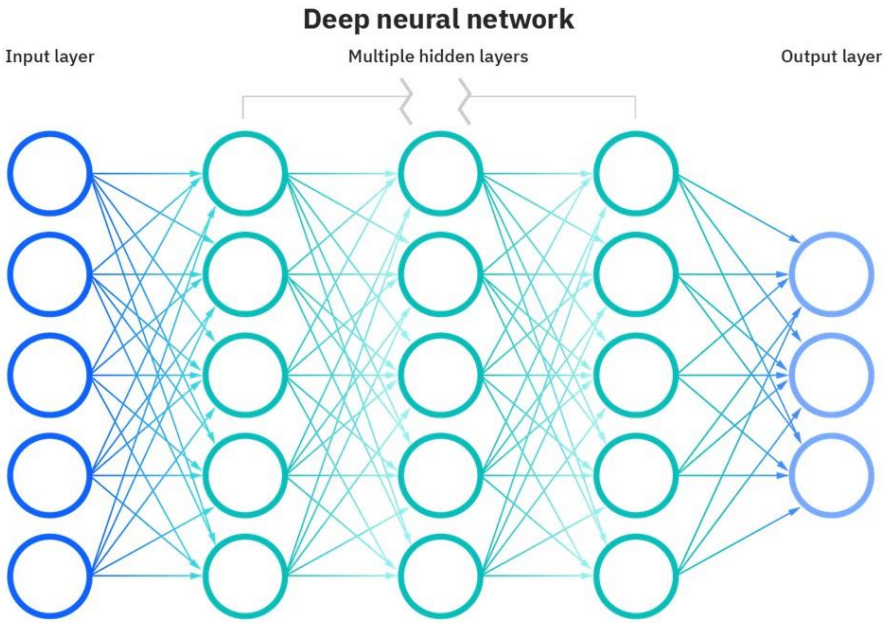
Neural networks (NN), also known as artificial neural networks (ANNs) or deep neural networks (DNNs), are a subset of Machine Learning (ML) and are the foundation of the deep learning algorithms. Machine Learning, as Géron [18] explained, can be defined as the science or even the art of programming computers so that they can learn from data, and it is a subfield of engineering and computer science that evolved from the study of pattern recognition and statistics and is thus a subset of Artificial Intelligence (AI). Therefore, ML is an application of AI that uses data to train and teach machines. The concept or the algorithm of an ANN is one of the techniques employed for performing ML to a set of data.

Essentially, the ANNs are computational models inspired by the human brain' architecture, that can perform ML and pattern recognition [18]. They are a sort of nonlinear processing that is well-suited for solving challenges in which the solution does not exist as a closed-form relationship [7]. Using a training procedure and sample data consisting of patterns or features, the neural network may be trained to simulate any form of nonlinear relationship. ANNs may be used for both supervised learning (classification and regression) and unsupervised learning (pattern recognition and grouping) [19]. In the last decades, the ANNs have been successfully used to solve complex direct problems involving prediction or analysis, as well as inverse problems involving the identification of mechanical properties.

The architecture of ANNs basically consists of layers (and or hidden layers), where each layer has a certain number of neurons, as shown in Figure 2.14. The neurons are

interconnected between layers in such a way that neurons can compute input values, allowing them to send the output values to the neurons of another layer. Each neuron is assigned with a weight and a bias that are adjusted during the learning process. The variations in the weight value change the strength of the neuron's signal [19]. The neurons have activation functions (sums, sigmoids, etc.) for computing the input values [18].

Figure 2.14 - Architecture of an Artificial Deep Neural Network.



Source: IBM (2020) [20].

The first layer of the ANN model has the number of neurons equal to the number of input variables or features when it comes to a simple ANN. The output layer is associated with the target value or category, for example, the value or classification label that the ANN model wants to predict. The second layer, or first hidden layer also has a specified number of neurons.

It is possible to add several hidden layers to the architecture, since the number of layers is related to the level of complexity of the physical problem to be modeled. However, when it comes to more than one hidden layer, the model becomes a deep learning concept [20], that is, a deep artificial neural network (DNN) as shown in Figure 2.14.

The ANN model has the ability to generalize, which is a type of interpolation in which the trained network estimates appropriate output data even for unlearned data. Therefore, this technique relies on training data to learn and improve their accuracy over time. In this context,

it is an important part of evaluating the ML models to separate the data into training and testing sets. When separating most of the data is used for training, and a smaller portion of the data is used for testing [18]. After a model has been trained by using the training set, it is possible to test the model by making predictions against the test set. Because the data in the testing set already contains known values for the attribute or features that is wished to predict, it is easy to determine whether the predictions of the model are correct. In data science a common ration split of the data is 80/20, that is, 80% of the data is reserved for training and 20% is reserved for testing the model [18]. Different ratios can be employed for a study of predictions. But choosing a model upon an optimized split ratio could lead to overfitting and biased results [18].

In general, the training process of a ANN is performed by minimizing the value of an error function descripted by Eq. (2.36). This function is commonly called mean squared error (MSE) or cost function C.

$$C = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.36}$$

In a forward pass, the Eq. (2.37) is used iteratively to calculate each neuron in the next layer. Where, $h^{(l)}$ is the activate neuron in the l layer, W are the weights, b the biases and σ is the activation function. The output $\hat{y}_i$ is measured by the cost function C and the wished result in the output layer.

$$h^{(l)} = \sigma\left(S^{(l-1)}\right) = \sigma\left(W\, h^{(l-1)} + b\right) \tag{2.37}$$

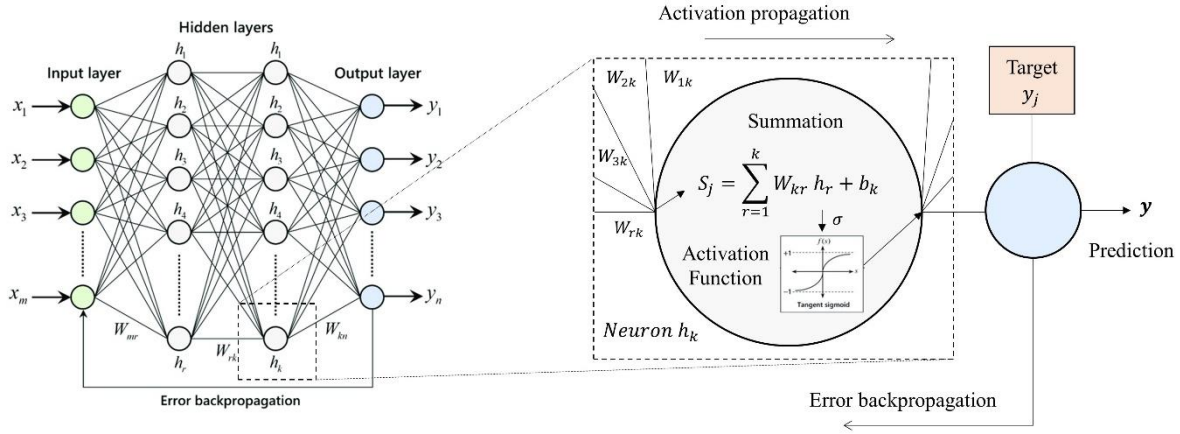
The weights and biases are updated in every iteration (or epoch) in such a way that the difference, or (cost function) between the result values and the target values is minimized. This process is made through a technique called backpropagation. The gradients are calculated in order to update the network, which are small nudges (updates) to individual weights in each layer, as shown in Eq. (2.38). The $S^{(l)}$ is the summation function.

$$\frac{\partial C}{\partial W^{(l)}} = \frac{\partial C}{\partial h^{(l)}}\frac{\partial h^{(l)}}{\partial S^{(l)}}\frac{\partial S^{(l)}}{\partial W^{(l)}} \tag{2.38}$$

In order to update the weights, in each layer, it is subtracted the value of the learning rate, times the cost of a particular weight, from the original value that particular weight had, as demonstrates Eq. (2.39), where l is the layer. In the iteration process t, the ANN attempts to strengthen the connections between neurons that lead to success and diminish those that lead to failure [18].

$$W_{(t)}^{(l)} = W_{(t-1)}^{(l)} - \text{learning rate} \times \frac{\partial C}{\partial W_{(t-1)}^{(l)}} \tag{2.39}$$

Figure 2.15 - Architecture of a simple ANN and the computation process of a neuron.



Source: adapted from FERNÁNDEZ-CABÁN (2018) [21].

ANNs have been employed in engineering for solving problem with complex analytical models or even for problems with a great number of features or input values. Therefore, it is a strong justification for using ANNs to model a solution to the limitation of plasticity effects in the ASTM E837 – 13a [6].

2.4.1 Artificial Neural Networks implemented in the HDM

Publications indicate that ANN may be used to anticipate residual stresses or identify their producing characteristics in a variety of systems [5]. Recently, ANNs models have been applied in the HDM. These applications include the correction of eccentricity errors, the correction of stresses measured on work-pieces having a specified thickness film applied to

them [22], and the prediction of residual stress for an equibiaxial stress state formed by laser peening [7].

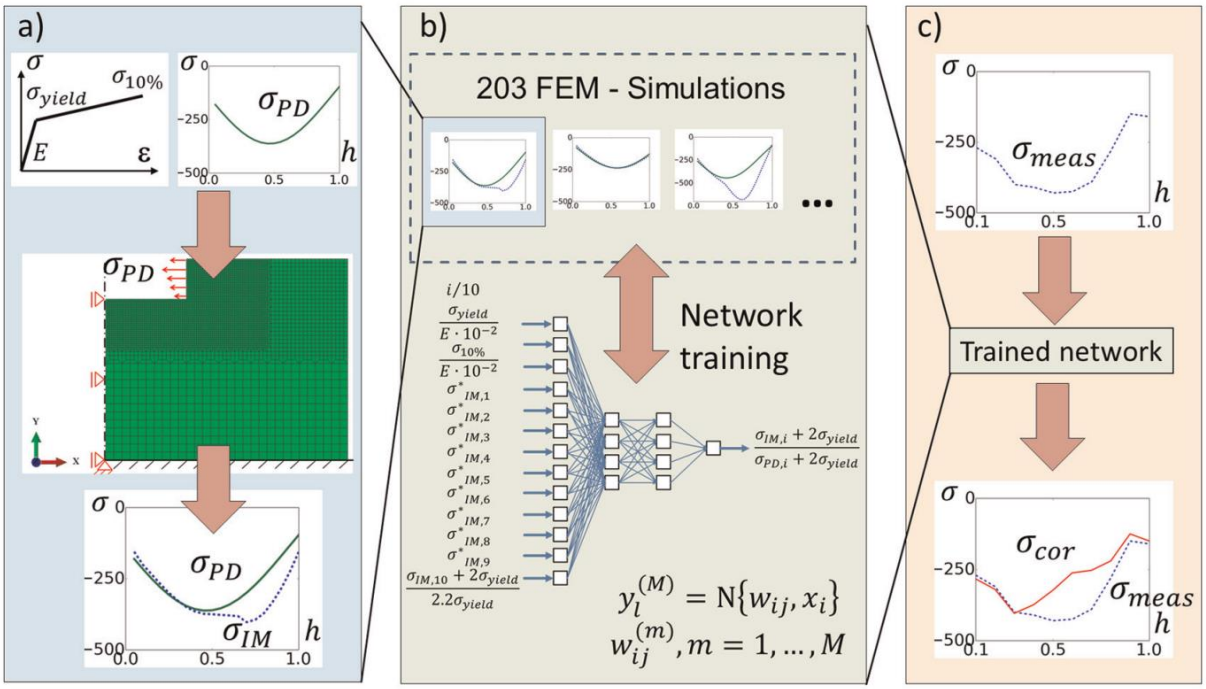
Chupakhin et al. [7] developed an ANN model to calculate a correction factor for residual stress data close to the yield stress. The laser shock peening procedure was assumed to provide an equibiaxial stress profile. Following a thorough established definition for input variables or features as shown by Eq. (2.40), the strategy displayed in Chart 2.1 was used to train and get adjustments with the ANN model in order to counteract plasticity effects.

$$\mathbf{X} := \left\{ \sigma^{*}_{\mathrm{IM},1}, \sigma^{*}_{\mathrm{IM},2}, \dots, \sigma^{*}_{\mathrm{IM},9}, \frac{\sigma_{\mathrm{IM},10} + 2\,\sigma_{\mathrm{yield}}}{2.2\,\sigma_{\mathrm{yield}}}, \frac{\sigma_{\mathrm{yield}}}{\mathrm{E}\cdot 10^{-2}}, \frac{\sigma_{10\%}}{\mathrm{E}\cdot 10^{-2}}, \frac{\mathrm{i}}{10} \right\} \tag{2.40}$$

$$\mathbf{Y} := \left\{ \frac{\sigma_{\mathrm{IM,i}} + 2\,\sigma_{\mathrm{yield}}}{\sigma_{\mathrm{PD,i}} + 2\,\sigma_{\mathrm{yield}}} \right\} \tag{2.41}$$

The **X** in Eq. (2.40) represents the dataset matrix of input values, while the **Y** in Eq. (2.41) is the label vector and it represents the output values. It is important to note that both **X** and **Y** are used for training but for the correction or proper use of the already trained model, only matrix **X**, will be the feeding values, since the vector **Y** contains the variables or values that the model wishes to predict. In the work of Chupakhin et al. [7], the **Y** is said to be the correction factor for residual stress profile, since the author opted for normalizing and to dimensionless the features to reduce the computational cost for the train of the ANN. The variable i represents the depth increment of the HDM applied to the FE model.

Chart 2.1 - Flowchart of an ANN application: (a) pattern generation using FE simulations, (b) training of the ANN and (c) application of the ANN to experimental data.



Source: CHUPAKHIN (2017) [7].

The normalization and scaling procedure applied to the data (dataset or database) before training an ANN model is a much recommended procedure according to the Data Science approach [18]. Chupakhin et al. [7] were able to normalize the input and output data with the actual engineering features involved in the system, as shown in Eq. (2.42) and (2.43).

$$\sigma^{*}_{\mathrm{IM,i}} := \frac{\sigma_{\mathrm{IM,i}} + 2\,\sigma_{\mathrm{yield}}}{\sigma_{\mathrm{IM,10}} + 2\,\sigma_{\mathrm{yield}}} \tag{2.42}$$

$$\sigma^{*}_{\mathrm{PD,i}} := \frac{\sigma_{\mathrm{PD,i}} + 2\,\sigma_{\mathrm{yield}}}{\sigma_{\mathrm{PD,10}} + 2\,\sigma_{\mathrm{yield}}} \tag{2.43}$$

The $\sigma_{\mathrm{IM,i}}$, is the residual stress obtained by applying the IM to simulation data, that is, without correction for plastic deformation effects, at a depth of h(i). According to equation, the PD residual stress profile, which was used as input to the FEM simulation, is translated to the dimensionless quantity $\sigma^{*}_{\mathrm{PD,i}}$, in the same way that $\sigma^{*}_{\mathrm{IM,i}}$ is. The desired stress profile is represented by $\sigma^{*}_{\mathrm{PD,i}}$ and the neural network will correct the $\sigma^{*}_{\mathrm{IM,i}}$.

The ANN was trained using random stress profiles (σ_{PD}) and with the stress profile generated using the integral method (IM), which is impacted by plasticity at high loads. Finally, the trained ANN model (fed with random stress profiles and with the material characteristics as features) was used for solving the inverse problem of stress profile correction.

Within predefined limits, material properties and coefficients vary randomly. The material properties were as follows: σ_{PD} =150-360 MPa, σ_{yield} =300-500 MPa; $\sigma_{10\%}$ = 400-600 MPa; and E = 70-210 GPa.

With the use of a trigonometric function, additional coefficients were used to construct the stress profile. A FE axisymmetric model was used in ABAQUS for simulating the hole drilling process involving plasticity. The IM (used for computing non-uniform stress profiles) was applied to the relaxation strain data for determining the equibiaxial stress field. The depth of the FE model is discretized in the form of $h(i)/h_{ref} = i/10$, where the reference depth is the maximum depth used for the hole drilling simulation, h_{ref} = 1 mm. The values obtained at depths $h(1)$ to $h(10)$ represent the residual stress profile. The residual stress at h_{ref} depth is selected as the reference and is used to normalize the residual stress profile up to this depth.

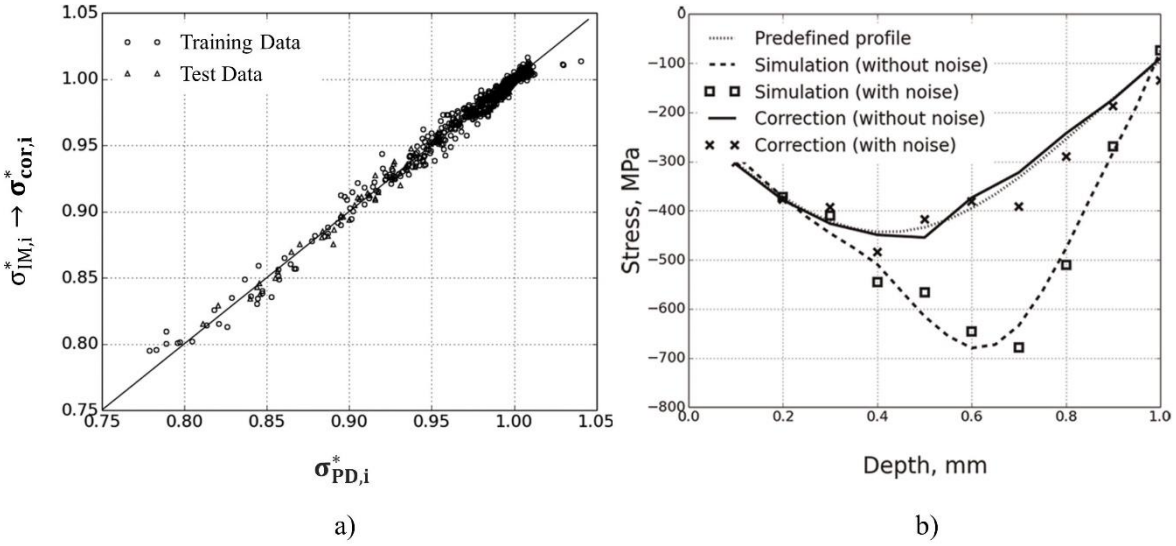
Basically, the ANN trained model approximates the function f descripted by Eq. (2.44).

$$\left(\frac{\sigma_{IM,i} + 2\,\sigma_{yield}}{\sigma_{PD,i} + 2\,\sigma_{yield}}\right) = f\left(\sigma^*_{IM,1}, \sigma^*_{IM,2}, \dots, \sigma^*_{IM,9}, \frac{\sigma_{IM,10} + 2\,\sigma_{yield}}{2.2\,\sigma_{yield}}, \frac{\sigma_{yield}}{E \cdot 10^{-2}}, \frac{\sigma_{10\%}}{E \cdot 10^{-2}}, \frac{i}{10}\right) \qquad (2.44)$$

In Figure 2.16.a) is possible to evaluate the quality of the predictions of the trained ANN both for the training dataset and for the test dataset. In the y-axis is presented the calculated output $\sigma^*_{cor,i}$ and in the x-axis is the desired output $\sigma^*_{PD,i}$.

In terms of the structure of the ANN, Chupakhin et al. [7], defined 13 input neurons, two hidden layers, each consisting of four neurons, and one output neuron. The ANN was trained with 2030 patterns (or row instances) for 5000 epochs (feedforward and backpropagation iteration) without any sign of overlearning. The training patterns (instances) were built with the 203 FEM simulations, where every set of 10 patterns had 12 identical input neurons and only the 13th input, which represents the depth h, was varying within the 10 patterns from values of 0.1 to 1.0.

Figure 2.16 - Evaluation of the ANN model: a) Quality evaluation of the ANN model predictions. b) Example of a residual stress profile correction.



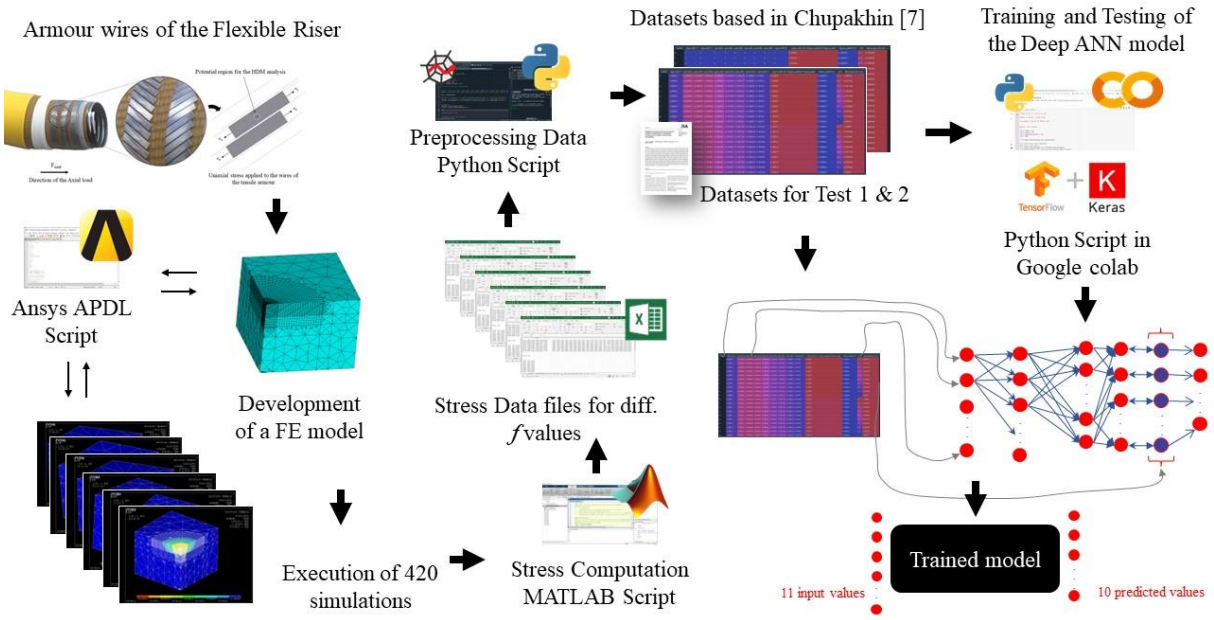
a) b)

Source: CHUPAKHIN (2017) [7].

3 METHODOLOGY

This section describes the methodologies adopted for carrying out the activities established upon the objectives of this undergraduate thesis. A description of the materials and software employed in the project of the thesis will be presented. The Figure 3.1 summarizes the data flow for the development of this thesis with the respective software and materials employed.

Figure 3.1 - Summary of the data flow for the development of the undergraduate thesis.



Source: elaborated by the author.

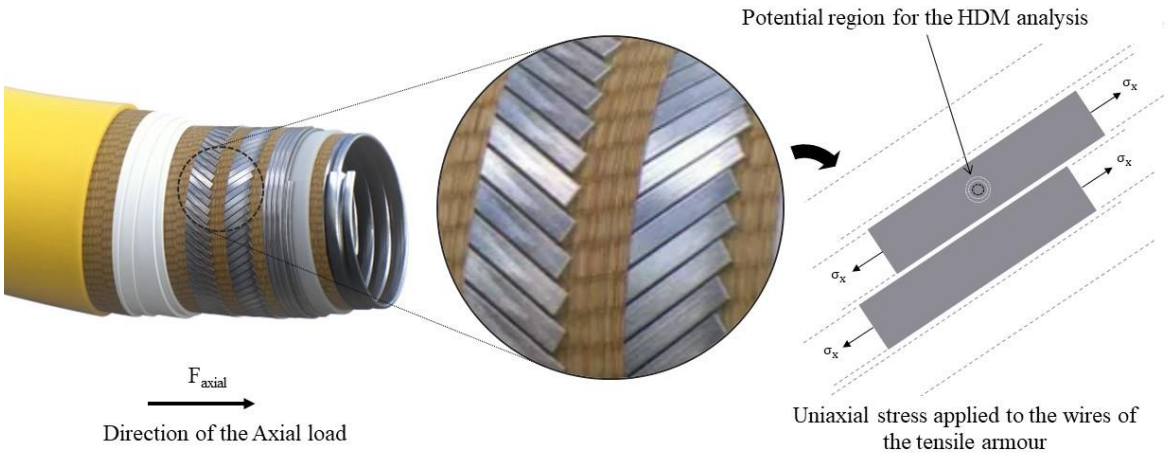
3.1 RESIDUAL STRESS IN FLEXIBLE RISERS

This undergraduate thesis is inserted in the Residual Stresses in Flexible Risers project (TRRiFlex). This abbreviation stands for *Tensões Residuais em Risers Flexíveis*, the project's Portuguese name. Prof. Armando Albertazzi, the advisor of this undergraduate thesis, oversees this project. The techniques developed in this project will contribute to understand how manufacturing processes can affect residual stresses. By minimizing the tensile residual stresses acting in the components of flexible risers, it will be feasible to work with more favourable

safety considerations. The project's major purpose is to develop technology that can evaluate residual stresses in flexible risers as well as individual tensile armour wires [5].

The tensile armour layer of flexible risers in fact presents high levels of residual stresses. Therefore, the proposal of the undergraduate thesis arises, by adopting the hypothesis of the existence of an ANN or Deep Learning model that can correct the residual stress profile taking into account the plastic effects similarly to the methodologies adopted in the papers performed by Chupakhin (2017) [7] and LI (2020) [22], In this context, the implementation of the ANN model would undoubtedly contribute to flexible riser operating safety. It will be beneficial to assess the condition of batches of flexible risers to reduce the likelihood of premature failure. As mentioned, the ANNs have already been used in a few studies linked to the HDM with consistent findings. Considering the assumption that the residual stress profile is substantially uniaxial in the tensile armour layer of flexible risers [1], as schematically shown in the Figure 3.2, the plasticity effects in the residual stresses profile measurements can then be corrected using a similar approach proposed by Chupakhin *et al*. [7]. Where, the precision of ANN model was expected to improve with the inclusion of input data relating to the strain profile around the hole.

Figure 3.2 - Uniaxial load of the tensile armour's wires of flexible risers



Source: elaborated by the author.

Chupakhin *et al*. [7] implemented a method for estimating the residual stress profile in laser peened specimens when the stress ratio is set to $\Omega = 1$ (equibiaxial case). If the strong uniaxial profile can be translated to a fixed value $\Omega = 0$ (uniaxial case), this approach can then

be extended to measure residual stresses in the armour wires of flexible risers. This also permits the non-uniform approach to be used to verify the stress fluctuation with depth.

The material properties corresponding to the armour wires of flexible risers were provided by researcher of the LABMETRO, as shown in Table 3.1. These values were estimated based on tensile test performed during the past years. The values implemented in the numerical simulation are descripted in the following section.

Table 3.1 - Material properties of the armour wires of flexible risers.

Material	**Yield stress** σ_{yield} **(MPa)**	**Young's modulus E (GPa)**	**Poisson's ratio** $\nu_{poisson}$
High-Carbon Steel Alloy*	1056	207.5	0.3

Source: elaborated by the author. Data provided by LABMETRO.

Observe that differently from the approach of Chupakhin *et al*. [7], the numerical simulations performed in this thesis took into account only one type of material, the one related to the armour wires of flexible risers.

3.2 DEVELOPING A FE MODEL FOR NUMERICAL SIMULATIONS

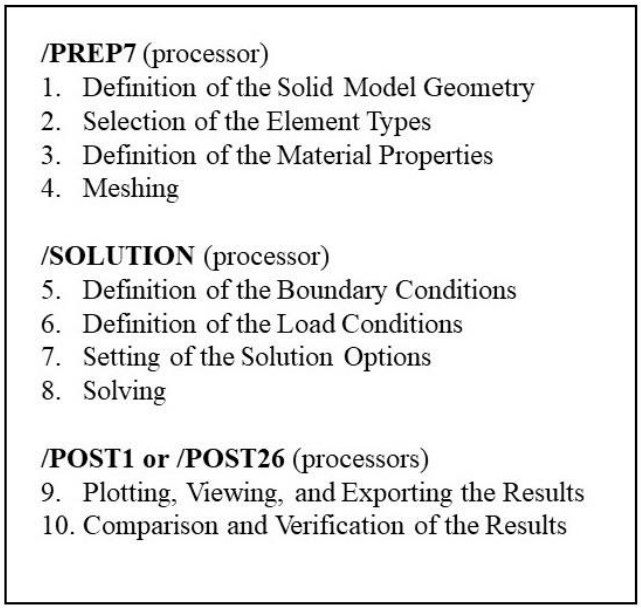
In order to build an ANN model a large amount of data must be available, and this involves performing several numerical simulations with Finite Element (FE) models requiring different loading conditions to meet a variety of stress profiles. The development of the FE 3D model for the thesis was built in Ansys APDL code, and the model is based on [7], [17], and works already developed in LABMETRO's projects. An extract of the code developed in APDL can be seen in Appendix B.

Ansys APDL or Ansys Parametric Design Language is the primary language used to commute with the Mechanical APDL solver. APDL can be used to automate tasks or, in the case of this thesis, create a complete parametric model. Ansys APDL is considered as an old-

* It is possible to disclose in this undergraduate thesis that the material used is a high-carbon steel alloy, for example a ABNT 1070 steel. However, since this is a material that involves company secrecy, the content value of the other elements that compose the armour wires is a confidential information and therefore is not disclosed in this thesis.

fashioned command-driven program that permits the user to input line by line codes and execute them on request. The advantage and what justified the choice of this tool to model and run the numerical simulations is in the level of control over the model. The possibility of writing optimized scripts allows for quick changes in geometry. It also allows more precise control of model mesh parameters. Another important justification is that such a tool is commonly used by many researchers in the field of the hole-drilling method simulations. For the user, the four main steps for the FEA, mentioned in Section 2.3, are performed following the sequence shown in Figure 3.3.

Figure 3.3 - Sequence commonly employed in FE software like APDL.

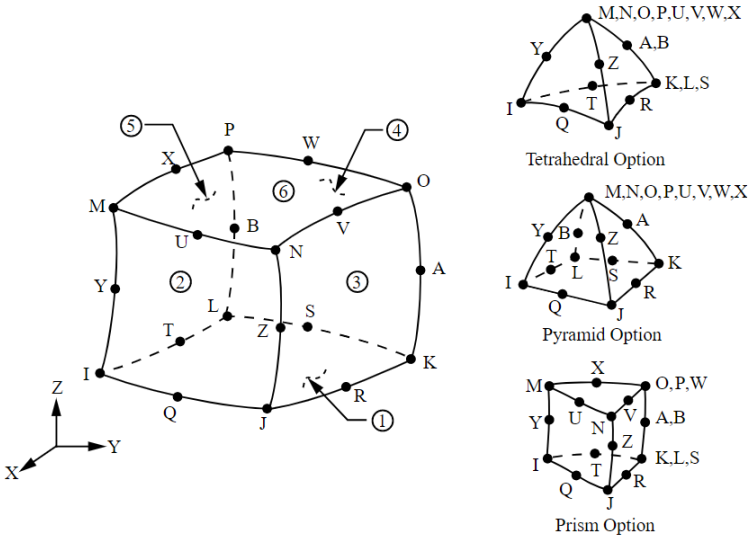


Source: adapted from THOMPSON (2017) [23]

The parametric code developed is intended to be run several times varying the parameter f, the plasticity factor. The variation of this value will affect the stress values applied as boundary and loading conditions of the model to be numerically simulated as explained in Section 2.3.1.

In the context of running numerous simulations, attention must be paid to the type of element chosen in terms of the FEA as well as the element size, which will directly define the computational cost. An available element in Ansys APDL and commonly used in the HDM analysis is the SOLID 186, Homogeneous Structural Solid Element [24], shown in Figure 3.4. This element type was used in the mesh process of FE model as indicated in the script presented in Appendix B.

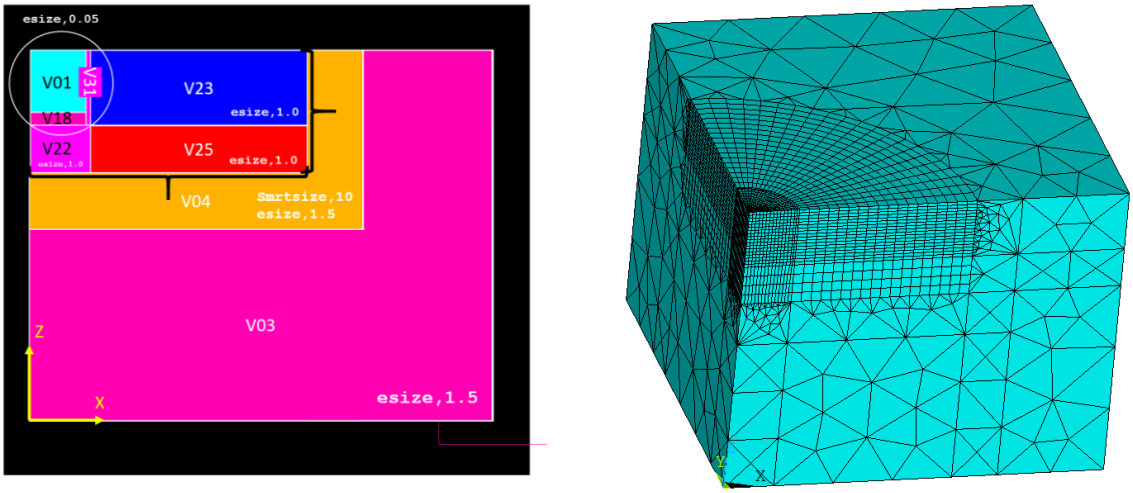
Figure 3.4 - Element SOLID186 Homogeneous Structural Solid Geometry



Source: DEPARTMENT OF APPLIED MECHANICS MM (2021) [24].

In this sense, mesh refinement tests were necessary to perform in the regions closest to the hole of the model. For the 3D model, three different levels of refinement were meticulously constructed as shown in Figure 3.5 and in Appendix B.

Figure 3.5 - Three-level-mesh model. In the left is the schematics for the different regions of mesh refinement. In the right is the 3D mesh model, where the refined meshed regions can be seen. The volumes id were given by order of construction in the internal algorithm of Ansys APDL.



Source: elaborated by the author.

In the Figure 3.5, the different volumes-blocks have different geometries and different values for the element size, which is translated with the *esize* command in APDL. Another aspect that was considered in the model is the structure of the mesh, that is, the distribution of the elements in the 3D model. The volume V03 corresponds to the coarsest mesh, and it is an unstructured mesh, whereas the volumes V01, V32, V23, V25, V18, V22 are structured refined meshes. In this context, "structured" means a mesh comprised of elements that follow a uniform pattern. The intermediate level, represented by the volume V04, it is a mesh that joins the coarse mesh to the refined meshes, and the structure to the unstructured meshes via the APDL command with internal optimization. It can also be seen that only a quarter of the geometry is simulated given its symmetry conditions, a common approach as discussed in Section 2.3.

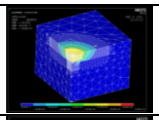
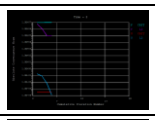
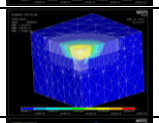
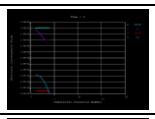
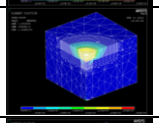
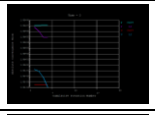
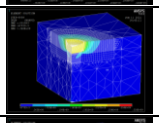
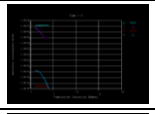
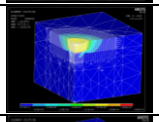
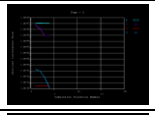
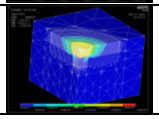
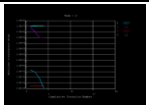
The code in APDL is automated to generate .txt files with data of the node displacements defined by the code. In this sense, the collection of displacements is done meticulously in a region (near the hole) equivalent to a virtual extensometer. This procedure is commonly performed by HDM researchers. A data vector of displacements in three axial directions is described in tabular form. Stress values can also be collected from the same nodes and in different directions, as well as the equivalent Von Mises stress.

Another important aspect to note is the loading conditions. The code was built considering the subtraction of two geometric solutions of the FE model, considering the superposition principle as indicated in Section 2.3. In this way, a model is numerically simulated with a geometry without the hole (part 1), then the volumes corresponding to the region of the hole are suppressed (via the *ekill* command[1]) and the model with the hole is numerically simulated with the same loading conditions (part 2). Then, the model without the hole is subtracted from the model with the hole, obtaining the displacements related to the introduction of the hole as if the load was applied in the hole wall. From the Figure 2.11 it can be understood that to obtain b) = c) - a).

Given the context of the computational cost, that is, the time required to run the simulation, six tests were made varying the element size via parameter change. Since the smaller the element size is, the finer the mesh and the bigger the number of elements is, consequentially bigger is the number of equations for the solver to compute. The summary of the tests can be seen in the Chart 3.1.

[1] When using the ekill command, the finite elements still exist, but the modulus of elasticity becomes very small, as if there was no material. Additionally, it is necessary to keep the same mesh to perform the subtraction of the models.

Chart 3.1 - – Tests of mesh refinement values. Variation of the *esize* command

Test n°	**Solution time (hh:mm:ss)**	**APDL**	**Solver iteration**	**Load (MPa)**	***esize* of V01, V18, V31 (mm)**	**Intermediate Mesh *esize* of V04 (*smartsize*,10) (mm)**	**Coarse Mesh *esize* of V03 (mm)**
00	00:20:30			237.5	0.05	1.5	1.5
01	00:26:30			237.5	0.05	1.5	2.0
02	00:20:20			237.5	0.05	2.0	2.0
03	00:29:10			237.5	0.05	2.0	2.5
04	00:19:00			237.5	0.05	2.5	2.5
05	00:20:50			237.5	0.05	3.0	3.0

Source: elaborated by the author.

The test 00 showed more stability in the numerical processing as well as good accuracy in the results. Therefore, the *esize* values for the several simulations were set according to the parameters of this test.

3.2.1 Execution of the numerical simulations

For the execution of the simulation using the parametric code in APDL language, a range of different values for the plasticity factor (f) was defined based on the works of [14], [16], and [17]. With the increasing range of f values, the uniaxial application stresses $\sigma_x \left(= \sigma_{eq}\right)$ were calculated, according to Eq. (2.32). Its isolated form is represented by Eq. (3.1). Additionally, when $\sigma_y = 0$, the $\sigma_{eq,i} = \frac{1}{3}\,\sigma_{yield} = 352$ MPa.

$$\sigma_x = \sigma_{eq} = f \times \left(\sigma_{yield} - \sigma_{eq,i}\right) + \sigma_{eq,i} \tag{3.1}$$

At total 420 numerical simulations were performed, summing over 100 hours of simulation time. For every 10 simulations, one f value was employed, therefore 42 f values were simulated. Every simulation represents 1 increment of the drilling process. In one drilling process of the standard ASTM E837 – 13a [6], it can be chosen 10 increments. With the predefined 1 mm hole drilling depth, each increment represents a 0,1 mm depth. Therefore, every 10 simulations out of 420, it represents 1 complete hole drilling process. Consequentially, in every 10 simulations, the load condition was the same. The Table 3.2 is an extract of the 42 f values simulated with their respective load conditions. The completed table can be seen in Appendix C.

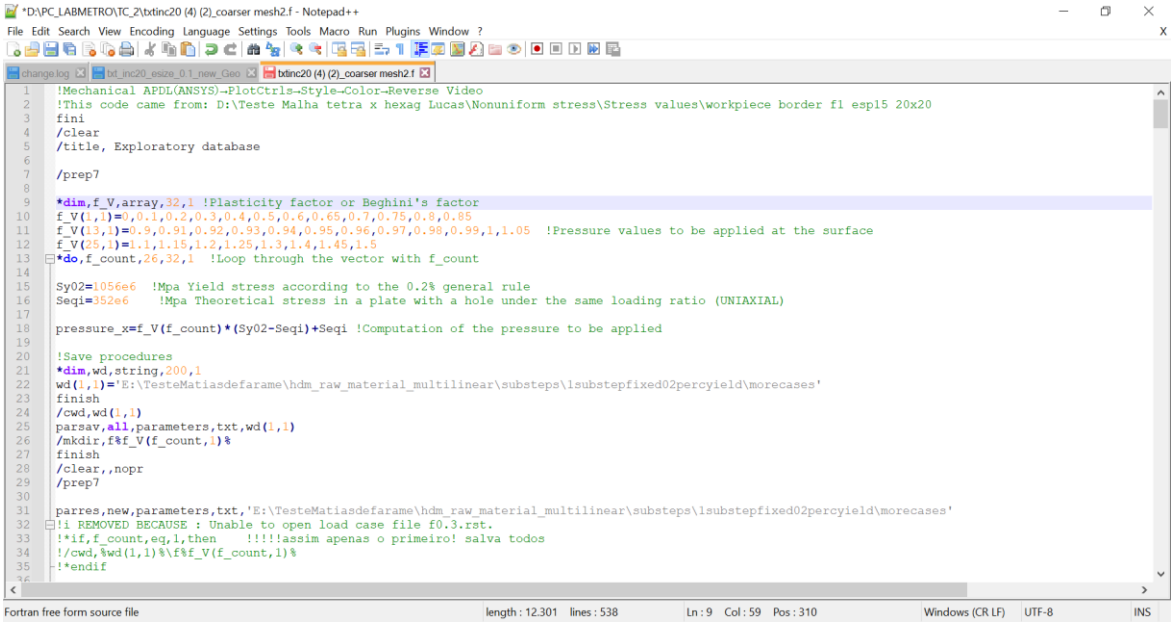
Table 3.2 - Values of Plasticity factor (or Beghini's factor) with the corresponded applied load value.

Plasticity factor	**Applied Load**	
f	$\boldsymbol{\sigma_x\ (=\sigma_{eq})}$ **(MPa)**	$\boldsymbol{\sigma_y}$ **(MPa)**
0	352	0
0,1	422,4	0
0,2	492,8	0
0,3	563,2	0
0,4	633,6	0
0,5	704	0
0,6	774,4	0
0,65	809,6	0
...	...	...
1,1	1126,4	0
1,25	1232	0
1,3	1267,2	0

Source: elaborated by the author.

Thus, the APDL code (shown in Appendix B) was generalized via a loop routine to simulate the 42 values of f, as shown in the screen shot in Figure 3.6.

Figure 3.6 – Generalized script for the execution of the numerical simulations. The f values are set via a range of values.



Source: elaborated by the author.

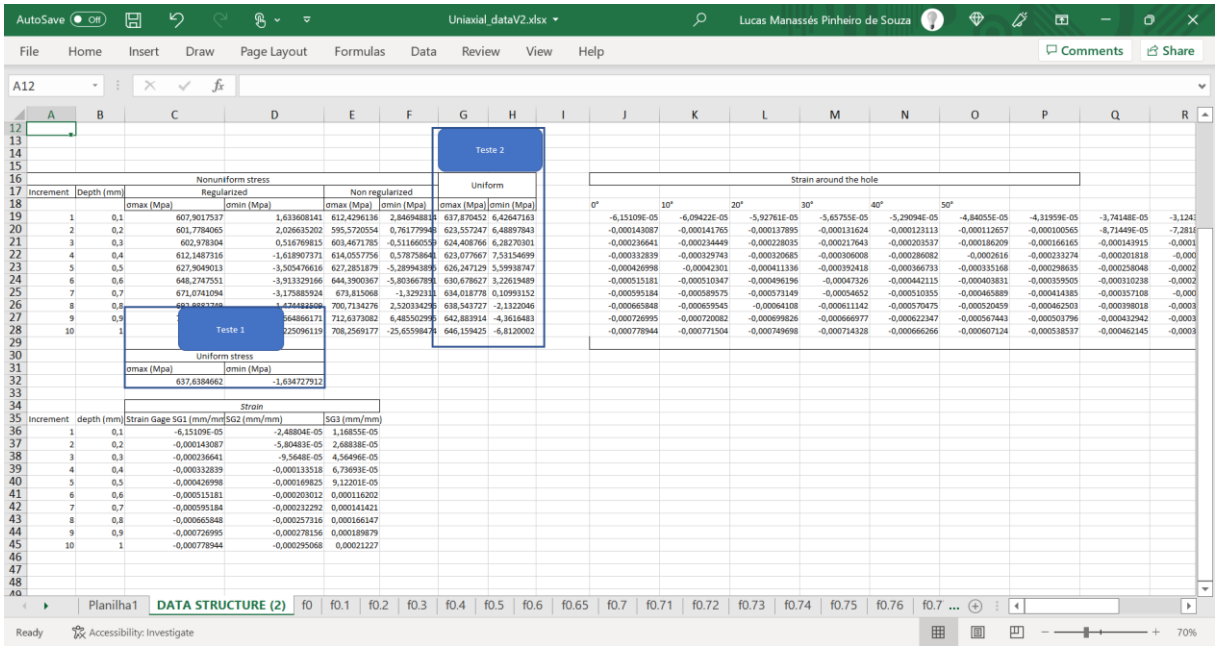
3.2.2 Stress computation using the calibration coefficients

In possession of the txt files with the displacement data of the model's finite element nodes, it was possible to import them via a MATLAB script, developed by the LABMETRO researchers. This code computes the stress values as a function of the deformations, which are also calculated as a function of the supplied displacements. The code was provided by LABMETRO and adapted to the study case of this thesis.

As can be seen in appendix D, this code basically computes the stress values according to the Eq. (2.9)-(2.29) shown in Section 2.2.1. The calibration coefficients (mentioned in Section 2.2) are implemented in the code according to the standard ASTM E837 – 13a [6]. Uniform as well as non-uniform stresses were calculated, even though a constant stress was applied along the hole wall.

The resulting data were arranged in an excel spreadsheet. For each different value of f, a spreadsheet with stress and strain data was allocated and arranged for each increment (row), since for each f value, 10 increments were simulated. In the Figure 3.7 it is possible to visualize how the data was stored in the excel sheets for the case of f=0.4.

Figure 3.7 - Example of the data structure of the excel sheets. The example used the data from f=0.4.



Source: elaborated by the author.

3.3 ESTABLISHMENT OF THE ANN DATABASE

The data manipulation of the simulation results was done by the Python tools in order to establish matrices of data, or more commonly referred to in AI, datasets. These datasets are the main material that feeds the ANNs. Both for training and for testing the ANN model.

Python is a high-level and multi-paradigm programming language, and it has a community development model, open and free. Python also has dynamic typing and one of its main characteristics is that it allows easy code reading and requires few lines of code compared to other languages. Because of its characteristics, it is widely used in processing scientific data. The justification for this choice lies in the ease of integration of the tooling with AI application packages of Machine Learning and Artificial Neural Networks.

The process that will define the datasets involves a crucially important step, the Data Analysis, or also called in Data Science, as the pre-processing of data. Basically, it is the part where engineering analysis is intertwined with statistical analysis to find the best features to feed an ANN with the aim of obtaining reasonable results.

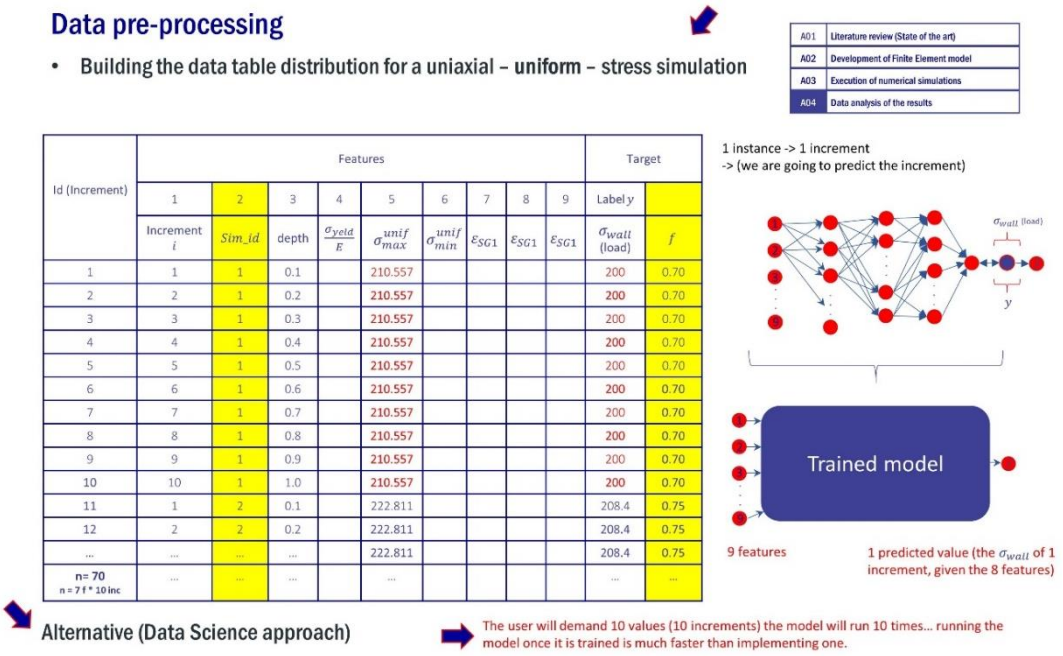
To perform this step some mathematical packages were used in Python, such as Pandas, a library for data manipulation and analysis, NumPy, a package that supports arrays,

multidimensional matrices, and a great collection of mathematical functions, Matplotlib, a library for creating plots and data visualizations in general, and SciPy, an Open-Source library that was made for mathematicians, scientists, and engineers. In addition, these tools help in the manipulation of data with great efficiency and robustness. The integrated development environment (IDE) used for programming the pre-processing of the data was Spyder, which is an open-source cross-platform IDE for scientific programming in the Python language.

From a Data science perspective, the mentioned datasets are prepared in table formats composed of columns with features extracted from statistical analysis of the raw data in general. The rows correspond to data instances, or frequencies of events. In the engineering field, this approach has been adapted for the use of engineering features (input of the model), that is, engineering variables and constants calculated with the classical approach and its correlation with the prediction of the unknown variables (output of the model). This form of pre-selection of features in the engineering approach may contribute to an easy implementation of the model and possible reduction of computational time in the training of ML models.

As brainstorm ideas, some dataset possibilities were explored with small amounts of simulated data. In Figure 3.8, a sketch is shown of the attempts and consideration of some engineering features.

Figure 3.8 - Sketch of the dataset considering some engineering features.

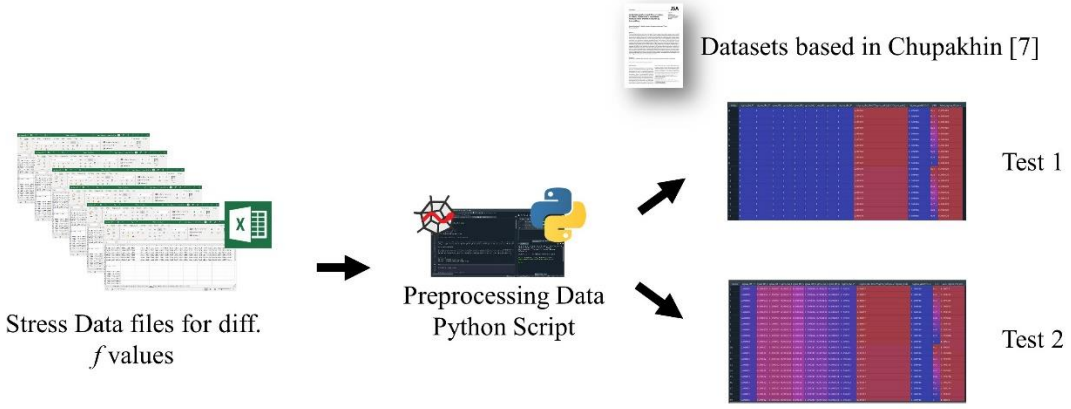


Source: elaborated by the author.

The works from [7] and [22] make use of the engineering features as input values for the model. Chupakhin et al. [7], established its input and output features according to Eq. (2.42)-(2.44), shown in Section 2.4.1.

Under the guidance of the advisors, two tests were structured for this undergraduate thesis based on the approach of Chupakhin's work. The Figure 3.9 illustrate the procedure. A relation based on Eq. (2.44) was explored in these two tests and the construction of the datasets was made based on the engineering features used by Chupkahin. It is noted that the approach of this author was also to normalize the values with engineering features, as can be seen in Eq. (2.40) and (2.41). The same procedure was implemented in this thesis. It is important to note that in this thesis, the stresses have uniaxial characteristics, differing from the equibiaxial approach of [7].

Figure 3.9 - The preprocessing of Data and the establishment of the two datasets



Source: elaborated by the author.

Based on Chupakhin's model, 12 of the 13 features used by the author were used in tests 1 and 2, represented by Eq. (3.1). The Eq. (2.41), which represents the vector **Y** (vector contains the variables or values that the model wishes to predict), remains the same.

$$\mathbf{X} := \left\{ \sigma^{*}_{IM,1}, \sigma^{*}_{IM,2}, \dots, \sigma^{*}_{IM,9}, \frac{\sigma_{IM,10} + 2\,\sigma_{yield}}{2.2\,\sigma_{yield}}, \frac{\sigma_{yield}}{E \cdot 10^{-2}}, \frac{i}{10} \right\} \tag{3.1}$$

In this sense, the trained ANN model in tests 1 and 2 is represented by the function f described in Eq. (3.2).

$$\left(\frac{\sigma_{IM,i} + 2\,\sigma_{yield}}{\sigma_{PD,i} + 2\,\sigma_{yield}}\right) = f\left(\sigma^*_{IM,1}, \sigma^*_{IM,2}, \dots, \sigma^*_{IM,9}, \frac{\sigma_{IM,10} + 2\,\sigma_{yield}}{2.2\,\sigma_{yield}}, \frac{\sigma_{yield}}{E \cdot 10^{-2}}, \frac{i}{10}\right) \quad (3.2)$$

3.3.1 Database with uniform stress values - Test 1

In this test, for every 10 simulations (10 increments) with the same value of f, the calculated values of $\sigma^*_{IM,i}$ and $\sigma^*_{PD,i}$ were the same for the 10 cases. The values of $\sigma^*_{IM,1}, \sigma^*_{IM,2}, \dots, \sigma^*_{IM,9}$ and $\left(\frac{\sigma_{IM,10}+2\,\sigma_{yield}}{2.2\,\sigma_{yield}}\right)$ were calculated according to Eq. (2.42) and (2.43), however as $\sigma_{IM,i}$ has the same value for being a uniform stress value along the hole, the values of $\sigma^*_{IM,i}$ for $i = 1,2,3, \dots,9$ becomes equals to 1. This test was considered a reasonable approach to adapt Chupakhin's methodology [7] to uniform uniaxial stresses. The code for the data preprocessing and feature calculations can be seen in Appendix E.

A screenshot of the dataset is shown in Figure 3.10, where 19 instances out of 420 can be seen. The dataset is exported in a comma-separated values (.CSV) file format, which allows data to be saved in a tabular format.

Figure 3.10 - Dataset established with uniform stress values - Test 1

Index	sigma_IM_1*	sigma_IM_2*	gma_IM_3	gma_IM_4	gma_IM_5	gma_IM_6	gma_IM_7	gma_IM_8	sigma_IM_9*	(sigma_IM_10+2*Sigma_yeld)/(2.2*Sigma_yeld)	Sigma_yeld/E10-2	i/10	label_sigma_PD_incs
0	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.1	0.999069
1	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.2	0.999069
2	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.3	0.999069
3	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.4	0.999069
4	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.5	0.999069
5	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.6	0.999069
6	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.7	0.999069
7	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.8	0.999069
8	1	1	1	1	1	1	1	1	1	1.05962	0.508916	0.9	0.999069
9	1	1	1	1	1	1	1	1	1	1.05962	0.508916	1	0.999069
10	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.1	0.998978
11	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.2	0.998978
12	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.3	0.998978
13	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.4	0.998978
14	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.5	0.998978
15	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.6	0.998978
16	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.7	0.998978
17	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.8	0.998978
18	1	1	1	1	1	1	1	1	1	1.08979	0.508916	0.9	0.998978

Source: elaborated by the author.

It is observable that the variable (output) or label is agglutinated as the last column of the dataset. This variable (called label) is also used for training the model and after the model is trained, this variable is the one to be predicted.

Another important aspect is that the values are repeated every 10 instances for two features, $\left(\frac{\sigma_{IM,10}+2\,\sigma_{yield}}{2.2\,\sigma_{yield}}\right)$ and $\left(\frac{\sigma_{IM,i}+2\,\sigma_{yield}}{\sigma_{PD,i}+2\,\sigma_{yield}}\right)$, the $\frac{i}{10}$ changes for every increment in a crescent order and repeat for every f value, but $\left(\frac{\sigma_{yield}}{E\cdot 10^{-2}}\right)$ remains constant throughout the dataset.

3.3.2 Database with non-uniform stress values - Test 2

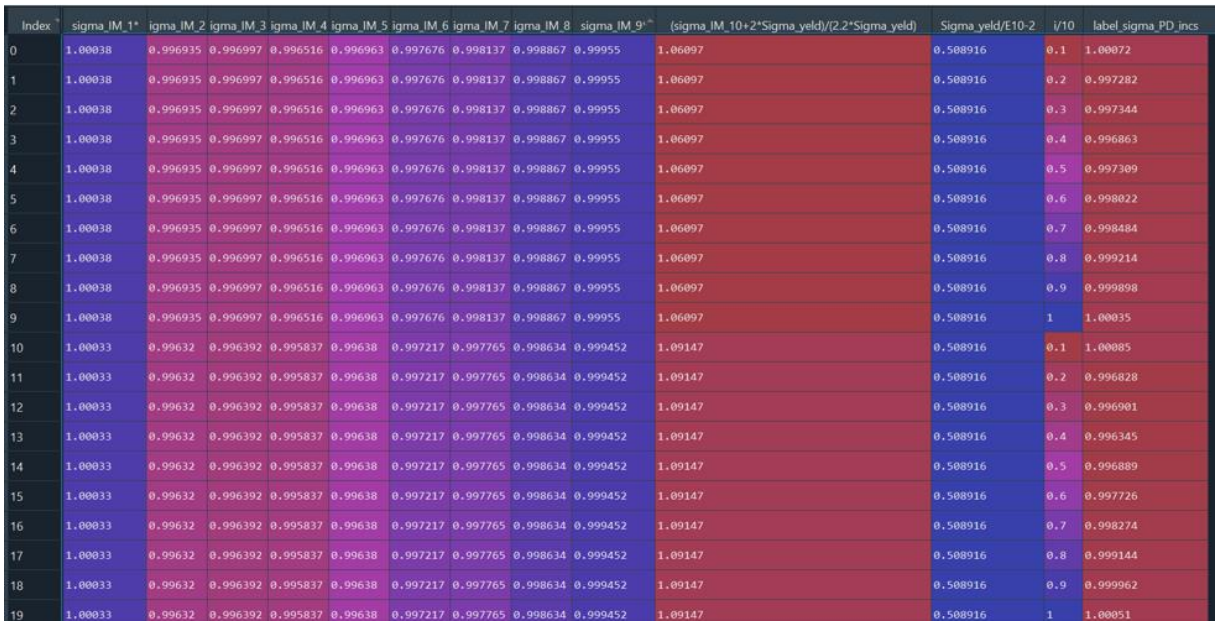
In the same way of the test 1, in this test for every 10 simulations (10 increments) with the same value of f, the calculated values of $\sigma^*_{IM,i}$ and $\sigma^*_{PD,i}$ were the same for the 10 cases. The values of $\sigma^*_{IM,1}, \sigma^*_{IM,2}, \dots, \sigma^*_{IM,9}, \frac{\sigma_{IM,10}+2\,\sigma_{yield}}{2.2\,\sigma_{yield}}$ were also calculated according to Eq. (2.42) and (2.43), however as $\sigma_{IM,i}$ has not the same value for being a non-uniform stress value along the hole, the values of $\sigma^*_{IM,i}$ for $i = 1,2,3,\dots,9$ becomes different than 1. This test was also considered a reasonable approach to adapt Chupakhin's methodology [7] to the uniaxial stresses using the values of the non-uniform stress.

It is worth noting that, the constant stress value along the hole wall for Test 1 is actually calculated by averaging the non-uniform stress values over the 10 increments calculated by the ASTM E837 – 13a [6] standard. In this thesis these values have been calculated by the script in Appendix C.

The code for the data preprocessing and feature calculations for the non-uniform stresses can also be seen in Appendix E.

A screenshot of the dataset is shown in Figure 3.11, where 20 instances out of 420 can be seen. The dataset is exported in a comma-separated values (.CSV) file format, which allows data to be saved in a tabular format.

Figure 3.11 – Dataset established with non-uniform stress values - Test 2



Index	sigma_IM_1*	igma_IM_2	igma_IM_3	igma_IM_4	igma_IM_5	igma_IM_6	igma_IM_7	igma_IM_8	sigma_IM_9	(sigma_IM_10+2*Sigma_yeld)/(2.2*Sigma_yeld)	Sigma_yeld/E10-2	i/10	label_sigma_PD_incs
0	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.1	1.00072
1	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.2	0.997282
2	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.3	0.997344
3	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.4	0.996863
4	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.5	0.997309
5	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.6	0.998022
6	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.7	0.998484
7	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.8	0.999214
8	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	0.9	0.999898
9	1.00038	0.996935	0.996997	0.996516	0.996963	0.997676	0.998137	0.998867	0.99955	1.06097	0.508916	1	1.00035
10	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.1	1.00085
11	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.2	0.996828
12	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.3	0.996901
13	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.4	0.996345
14	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.5	0.996889
15	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.6	0.997726
16	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.7	0.998274
17	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.8	0.999144
18	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	0.9	0.999962
19	1.00033	0.99632	0.996392	0.995837	0.99638	0.997217	0.997765	0.998634	0.999452	1.09147	0.508916	1	1.00051

Source: elaborated by the author.

3.4 BUILDING AND IMPLEMENTING THE ANN MODEL

After the datasets are filtered and tested, the actual construction of the Artificial Neural Network model can be performed. The Figure 3.12 exemplifies the building procedure of the model. This process uses different mathematical and computational tools.

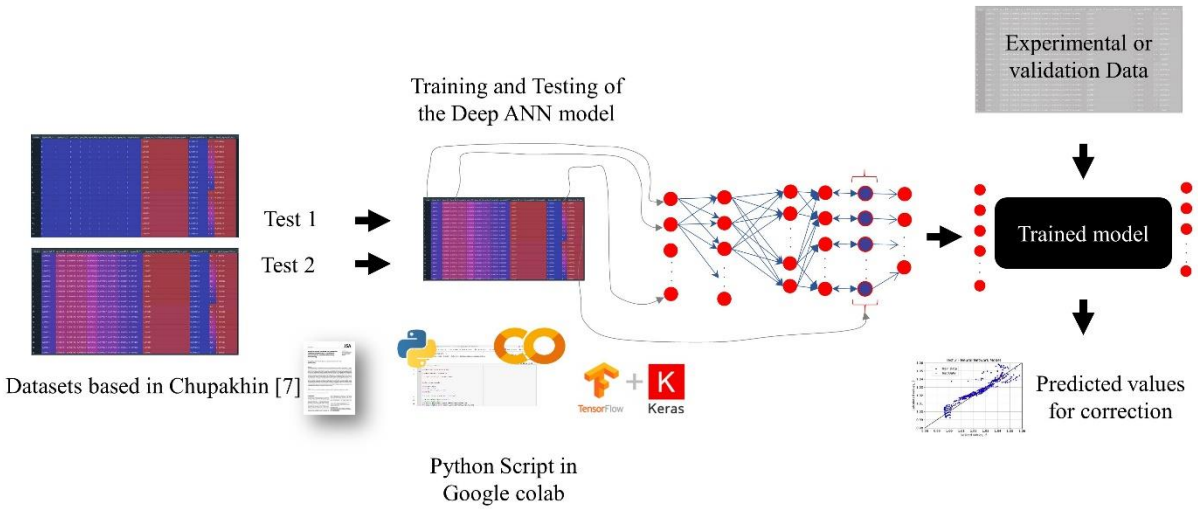
One of the tools is the Scikit-learn toolkit, which is an open-source Machine Learning library, including several classifications, regression, and clustering algorithms, and it is designed to interact with NumPy and SciPy.

For this undergraduate thesis, a sequential model of an ANN will be developed through Python code using Keras tool. The Keras is an open-source neural network library written in Python. It is designed to enable the fast use of DNNs. The choice of this tool is in the convenience and ease of use in implementing the ANN models.

The attempt at parameter optimization of the number of layers, activation functions and number of neurons will lead to an application of more than one hidden layer, so the work will involve Deep Learning concepts. More layers in the model can help solve the complexity of the problem, which can lead to a more robust and accurate model. This process will require the use of TensorFlow, which is an open-source library for ML that works efficiently with complex mathematical computations involving multidimensional matrices.

Given the complexity of the packages used for the installation and implementation of the scripts, especially regarding the combined use of Keras and TensorFlow, which works with complex tensor operations, the choice of using Google Colab Platform for the development of the ANN model scripts seemed advantageous in terms of avoiding incompatibility of package and library versions. The Colaboratory or also known as "Colab," is a Google Research product and it is particularly well suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code through the browser. Basically, Colab is a free Jupyter notebook environment that runs entirely in the cloud, and it does not require a setup of packages installation. Additionally, Colab can support over 40 different programming languages.

Figure 3.12 - Flow chart of the construction of the ANN model.



Source: elaborated by the author.

The coded script for the ANN model implementation can be seen in Appendix F. Note that the code imports the tables in CSV format. The files were imported directly from a Google Drive link.

Before training an ANN model, it was crucial to split the dataset into two subsets: the training set and the test set [18]. As the name follows, the first subset trained the model according to the target values presented in this set, the second subset was used for testing and validation of the model. It was important to keep the set apart and not contaminate the training set with data from the test set, and vice-versa, otherwise the results would not be reliable [8]. A recommendation for the split was given by [18], circa of 80% of the data should be used for training and the remaining for testing. However, to match the Chupakhin-based model, and

following its parameters, a 90% split for model training and 10% for validation or testing was performed.

It is important to point out that in the development of this undergraduate thesis, no experimental data will be used to test or validate the model. Only data from numerical simulations are being used. This means that 90% of the data resulting from numerical simulations that were allocated to datasets are being used for training. Additionally, implementations with different split values can be carried out to evaluate the performance of the models.

In the script the split chooses the 10% of values (42 instances) randomly from the database to maintain consistency of prediction and generality of the model, avoiding a biased trained model.

The Table 3.3 shows the parameter values used in building and training the DANN model. It is observable that for the two tests performed the parameters were kept constant because the objective was also to compare the performance in terms of errors and predictions between the two tests.

Table 3.3 - Parameters of the training of the DNN model. Values based on Chupakhin's model.

Parameters	**Test 1**	**Test 2**
N° of layers	5	5
1° hidden layer		
N° of neurons	4	4
Activation function	Relu	Relu
2° hidden layer		
N° of neurons	4	4
Activation function	Relu	Relu
3° hidden layer		
N° of neurons	1	1
Activation function	Linear	Linear
Optimizer	Adam	Adam
Loss function	MSE	MSE

N° of Epochs	5000	5000
Split ratio (testing/training)	10/90	10/90
Values Approach	Uniform	Non-uniform

Source: elaborated by the author.

After reliable results are achieved, comparisons between models and parameters can be made, and as a result future changes can be implemented.

The validation of the trained models was done using the separate data set for testing. The error analysis was done by comparing the initial vector of **Y** values with the corresponded vector of predicted values $\widehat{\mathbf{Y}}$. The difference indicates how far off the model is in terms of prediction. The validation of the test data only had 42 instances evaluated (corresponding to the 10% of the split).

3.4.1 Evaluation performance of the DANN model by classical comparison.

For the purpose of comparing the performance of a DANN model implementation, a Multivariable Linear Regression (MLR) model based on classical equations was built via code. As defended by [18], the multivariable linear regression model can also be classified as an ML model, in fact, this model is one of the simplest and most classical ML models, more robust models have been based on this approach. Thus, it is justified to choose this classical model as a basis for comparison. The classical equation for the MLR model prediction is shown in Eq. (3.3). Where $\hat{y}$ is the predicted value, n is the number of features (number of variables), x_i is the i^{th} feature value, the θ_j is the j^{th} model parameter, including the bias term θ_0 and the feature weights θ_1, θ_2, ..., θ_n.

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n \tag{3.3}$$

As can be seen in the script in Appendix F, the MLR model was implemented in parallel to the DANN model and followed the same training and validation test procedure in the principles of the ML model training approach.

For the validation of the performance comparison between the DANN model and the MLR model, the 42 instances were evaluated, comparing among these values how many predictions the DANN or the MLR model obtained a lower error, therefore better performance.

4 RESULTS AND DISCUSSIONS

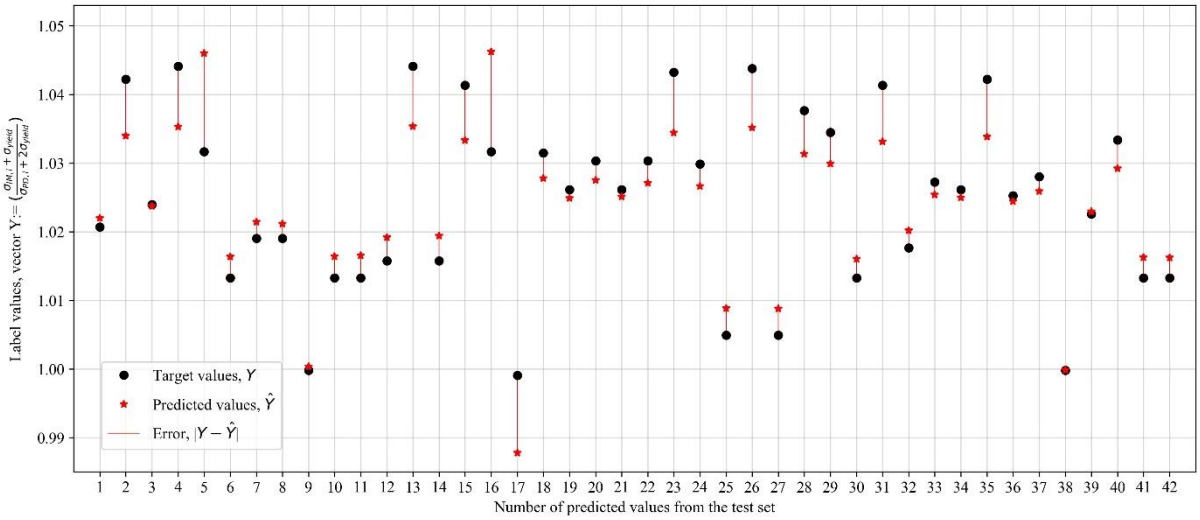
In this section the results obtained from the implementation of the DANN model are presented. Throughout the section, four analyses or discussions are carried out. The first analysis is related to the validation of the data set separated for testing the trained model. The deviations and errors in the prediction of values by the model are presented for the two databases (Test 1 and Test 2). In the second analysis, the prediction of values for the entire database (training and test) are discussed for each database. In the third analysis, the performance comparison of the DANN model with a classical MLR model is discussed, also observing the errors and for each database. In the fourth analysis, the prediction/correction of stress profiles along the hole for some selected plasticity factors are presented.

4.1 VALIDATION OF THE TEST SET

It is important to note that of the 420 instances (simulations), only 42 instances were separated to test the model, thus the 378 instances were used to train the DANN model. The same procedure was done for both databases (Test 1 and Test 2). It is also important to emphasize, as explained in the previous section, that the 42 test values were chosen randomly so as not to contaminate or bias the trained model.

Considering the first database (Test 1) and the Eq. (2.41) and (3.2), used in the calculation of the features, the predicted values in comparison with the target values (the Y vector label) were able to be plotted, as shown in Figure 4.1.

Figure 4.1 - Label values and its Predictions for the test set - Database of Test 1

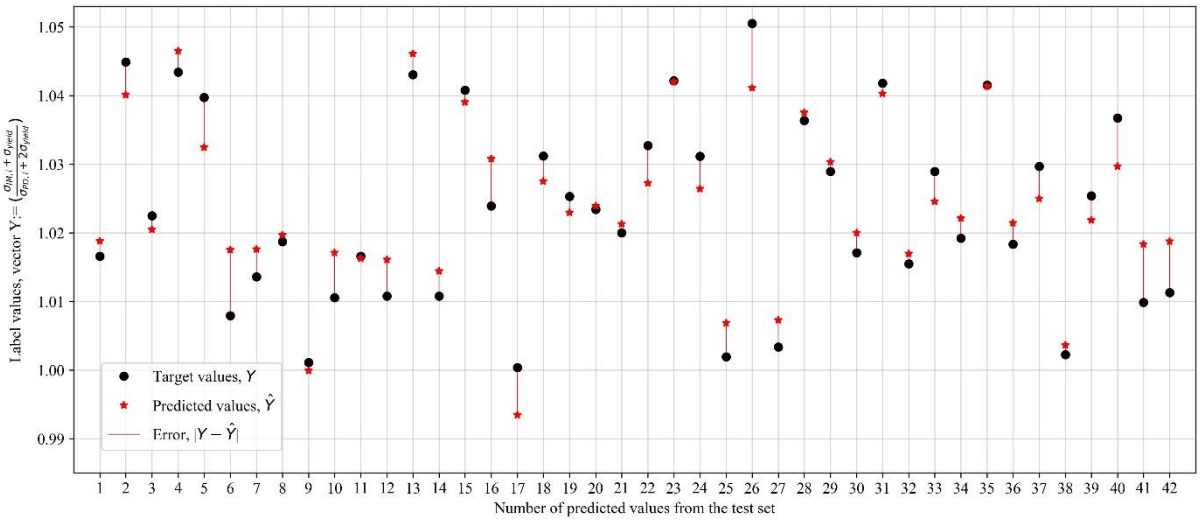


Source: elaborated by the author.

It is possible to note that the 42 values are not displayed in ascending order. The figure is only intended to show the difference in prediction in relation to the target values, i.e., in an ideal model (without errors) the lines and points in red should coincide with those in black.

Considering now the second database (Test 2) and the Eq. (2.41) and (3.2), used in the calculation of the features, the predicted values in comparison with the target values (the Y vector label) were able to be plotted as shown in Figure 4.2.

Figure 4.2 - Label values and its Predictions for the test set - Database of Test 2

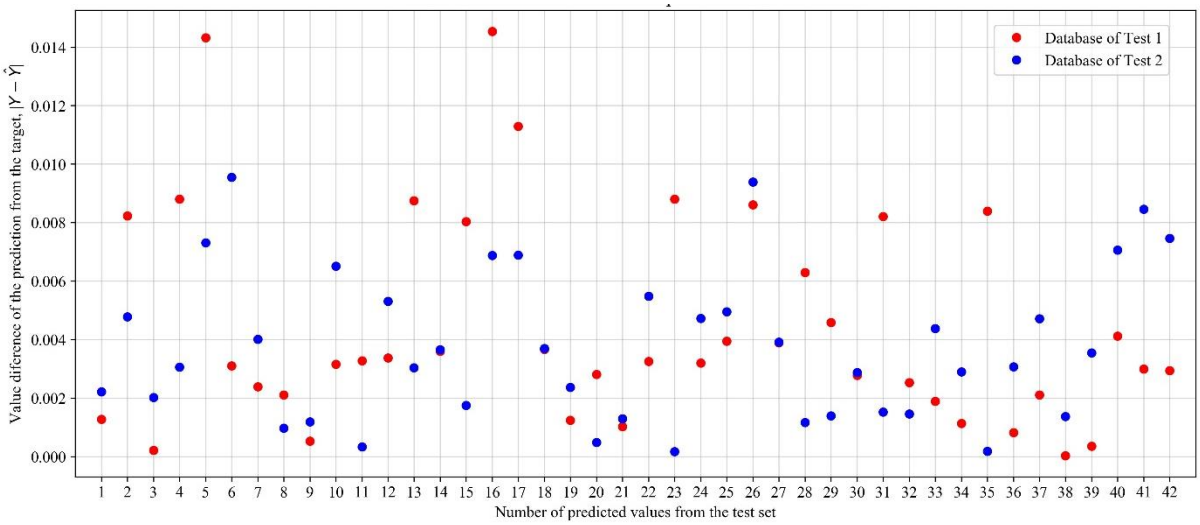


Source: elaborated by the author.

It is noticeable that the values in black are not in fact the same for the two cases, even though the position of the instance in the database was the same. The values changed from one database to the other. It can also be seen that the error projection is very similar in both cases.

In the graph of Figure 4.3, it is shown the plot of the difference (the error) between the estimated values $\widehat{\mathbf{Y}}$ and the target values **Y** of the test set. This graph shows in blue the errors of the 42 instances of the test set for the database of Test 1 and in red for the database of Test 2.

Figure 4.3 – Difference (the error) between the estimated values $\widehat{\mathbf{Y}}$ and the target values **Y** of the test set



Source: elaborated by the author.

Something important to highlight in the prediction of the two databases is in the analysis of errors, it was noticed that both models performed relatively accurately with errors on the order of 4.4×10^{-3} for the database of Test 1 and 3.8×10^{-3} for the database of Test 2.

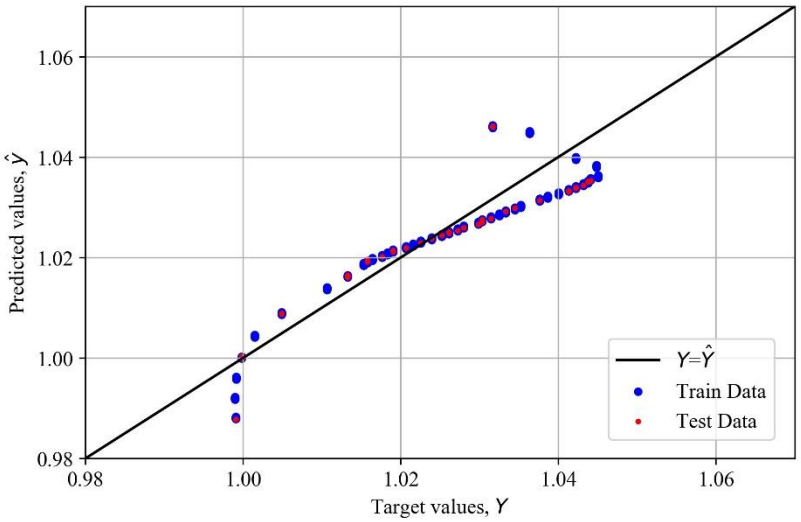
4.2 PREDICTIONS OF THE TRAINED MODEL FOR THE ENTIRE SET

In this section the prediction of DANN model values for the entire set of simulated data is discussed.

Considering the database of Test 1, a plot of the predictions of the DANN model vs. the target values for the entire set of data (instances or simulations) is shown in Figure 4.4. It is noticeable that in red is the separate data set for testing and in blue is the separate data set for training.

By taking the Figure 3.10 as a reference, it is known that every 10 instances the values of the vector are repeated, so when this plot is compared with Figure 4.5, one notices a low dispersion or even an apparent lower volume of data. However, what happens is the repetition of the values, since they are uniform stress values (a constant value along the hole, i.e. for the same f value).

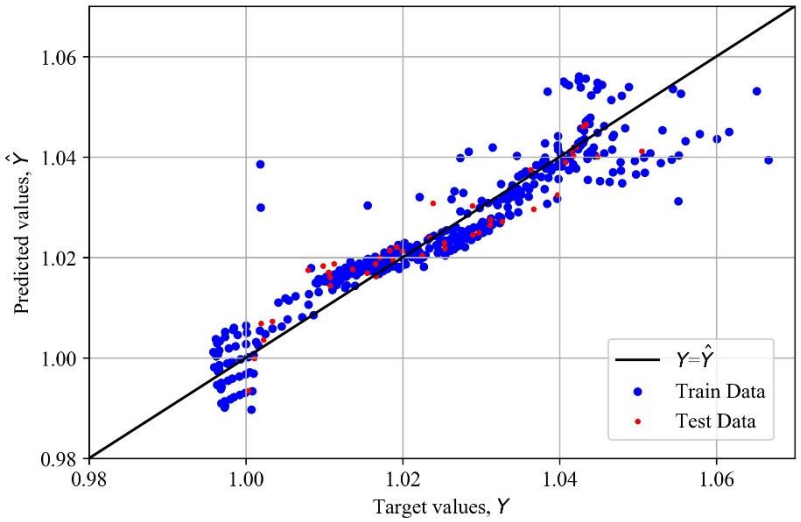
Figure 4.4 - Predictions of the DANN model vs. the target values of Test 1 Database.



Source: elaborated by the author.

Considering the Figure 3.11, it is observable that a greater dispersion and volume of data is plotted in Figure 4.5.

Figure 4.5 - Predictions of the DANN model vs. the target values of Test 2 Database.



Source: elaborated by the author.

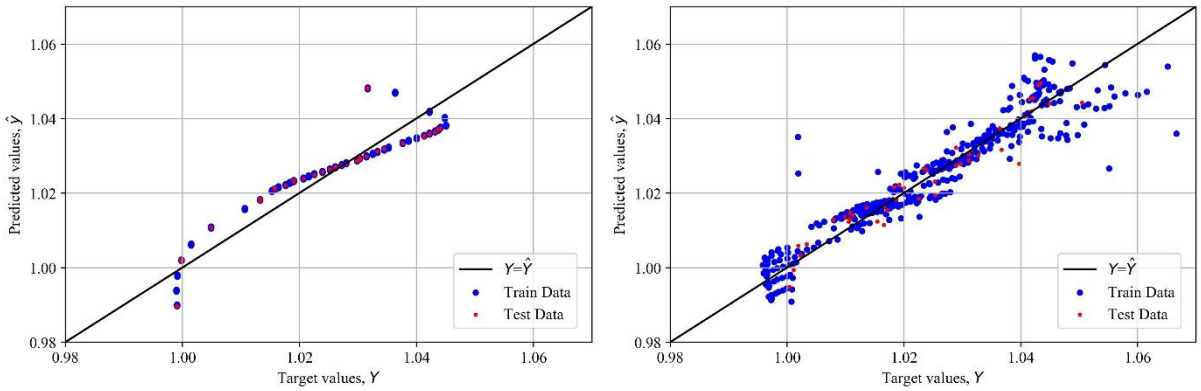
Something important to note is the difference between the Chupakhin *et al*. [7] plots of Figure 2.16(a) and the plots in this section. Chupakhin *et al*. [7] created a specific variable defined by the Eq. (2.43) to compare the values estimated by the model. It is worth noting that in a Data Science approach, the estimated (predicted) values are compared directly with the model's target values (the label represented by the vector **Y**), to match the label that was used to train (supervise) the model. Therefore, the plots above were plotted following this logic.

4.3 EVALUATION PERFORMANCE OF THE DANN MODEL

In this section the comparison of predictions between the DANN model and the MLR model is presented. As well as an error analysis and comparison is presented. As mentioned in Section 3.4.1, a multivariable linear regression model can also be classified as an ML model forming a basis for performance comparison.

The plots shown in Figure 4.6 present the value predictions of the MLR model vs. the target values for the test set and the training set. In a) it is shown the database of Test 1 and in b) is shown the database of Test 2. The data set for training is in blue and in red is the separated set for testing. As mentioned in Section 3.4.1, the MLR model was implemented in parallel to the DANN model and followed the same training and validation test procedure in the principles of the ML model training approach. That includes, the same split for training and testing.

Figure 4.6 - Predictions of the MLR model vs. the target values. The Database of Test 1 is in a) and the Database of Test 2 is in b).



a) b)

Source: elaborated by the author.

What can be seen by comparing the plots of the Figures 4.4, 4.5 and 4.6 is that the MLR model had fewer deviations and less dispersion in the distribution of predictions values than the DANN model.

Using a comparison formula as shown in Eq. (4.1), where it is analyzed by Boolean variables whether the statement is true or false, i.e. whether the MLR model error is greater than the DANN model error.

$$\mathbf{comparison} = \left|\mathbf{Y} - \widehat{\mathbf{Y}}_{\mathrm{DANN}}\right| < \left|\mathbf{Y} - \widehat{\mathbf{Y}}_{\mathrm{MLR}}\right| \tag{4.1}$$

For the analysis of the model test set, considering the database of Test 1, it was computed that from 42 predicted values in the test set, 21 predicted values had less errors when predicted by a linear model (MLR), and the other 21 predicted values had less error when predicted by the DANN model. For the database of Test 2, it was computed that from 42 predicted values in the test set, 27 predicted values had less errors when predicted by the MLR model, whereas only 15 predicted values had less error when predicted by the DANN model. It was expected that the Test 2 database would increase the volume of data instances so that it could generalize the model in order to enhance the performance of the DANN model, but this did not happen. The Table 4.1 shows the Boolean values of the comparison given by Eq. (4.1).

Table 4.1 - Results of Boolean values from Eq. (4.1) for the test set for each database.

Database of Test 1		**Database of Test 2**	
Index	**Comparison**	**Index**	**Comparison**
1	True	**1**	False
2	False	**2**	False
3	True	**3**	True
4	False	**4**	True
5	True	**5**	True
6	True	**6**	False
7	True	**7**	False

8	True	**8**	False
9	True	**9**	True
10	True	**10**	False
11	True	**11**	True
12	True	**12**	False
13	False	**13**	True
14	True	**14**	False
15	False	**15**	False
16	True	**16**	False
17	False	**17**	False
18	False	**18**	False
19	False	**19**	False
20	False	**20**	True
21	False	**21**	True
22	False	**22**	False
23	False	**23**	True
24	False	**24**	False
25	True	**25**	False
26	False	**26**	False
27	True	**27**	False
28	False	**28**	False
29	False	**29**	True
30	True	**30**	False
31	False	**31**	False
32	True	**32**	True

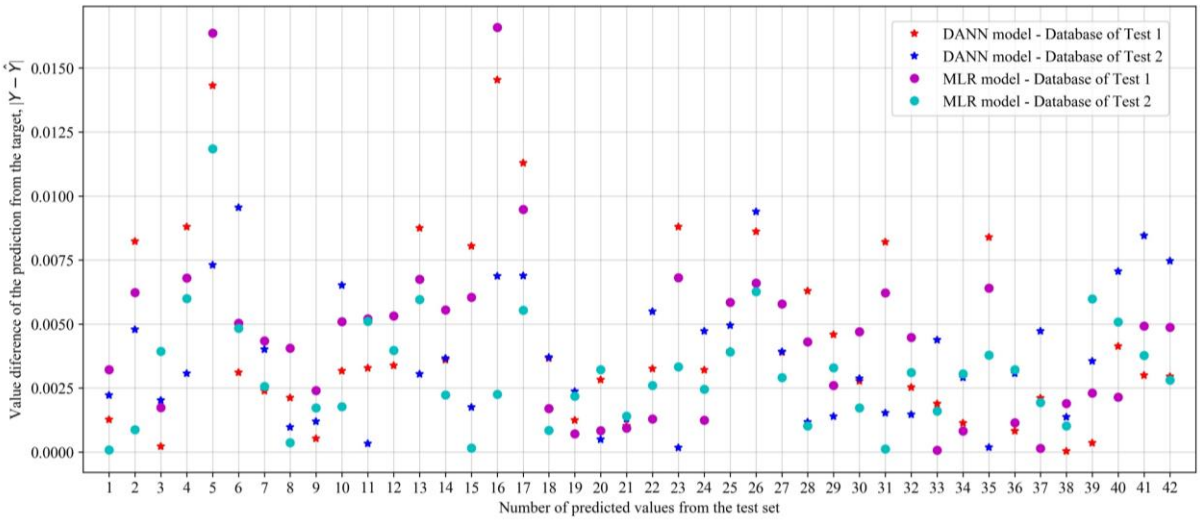
33	False	**33**	False
34	False	**34**	True
35	False	**35**	True
36	True	**36**	True
37	False	**37**	False
38	True	**38**	False
39	True	**39**	True
40	False	**40**	False
41	True	**41**	False
42	True	**42**	False

Source: elaborated by the author.

In the first database (Test 1), for the test set, the DANN model was favored to have lower errors for half the values in the test set, whereas in the second database, which has a higher data dispersion, the DANN model performed worse. The results bring up the discussion about a certain existence of linearity in the databases, which may question the computational efficiency of implementing and using a DANN model. Since the training time for the DANN model under the pre-set conditions took about 150 x longer training time than the MLR model.

In the plot of Figure 4.7 it can be seen the errors for the test data set for the two models. For the MLR model, the errors magnitude are on the order of 4.5×10^{-3} for the database of Test 1 and 3.1×10^{-3} for the database of Test 2. This implies that the error in predicting values for the MLR model is in the same magnitude of the DANN model for the database of Test 1. On the other hand, the error in predicting values for the MLR model is lower than for the DANN model for the database of Test 2, so that the error on average of the DANN model was about 1.2 times larger than the error of the MLR model.

Figure 4.7 - Errors values of the models for the test set.



Source: elaborated by the author.

Additionally, the values of the Plasticity Factors f among the separate data vector for testing were investigated. The values of f corresponding to the test set index can be seen in Table 4.2. The indexes shown in the table correspond to those identified in the plots in Figures 4.2, 4.3 and 4.7.

Table 4.2 - Plasticity Factor values corresponding to the indexes of the test set.

Index	**f**	**Index**	**f**	**Index**	**f**	**Index**	**f**	**Index**	**f**
1	0.76	**11**	0.65	**21**	0.81	**31**	0.95	**41**	0.65
2	0.96	**12**	0.71	**22**	0.85	**32**	0.73	**42**	0.65
3	0.79	**13**	0.99	**23**	0.97	**33**	0.82		
4	0.99	**14**	0.71	**24**	0.84	**34**	0.81		
5	1.30	**15**	0.95	**25**	0.50	**35**	0.96		
6	0.65	**16**	1.30	**26**	0.98	**36**	0.80		
7	0.75	**17**	0.00	**27**	0.50	**37**	0.83		
8	0.75	**18**	0.86	**28**	0.92	**38**	0.30		
9	0.30	**19**	0.81	**29**	0.89	**39**	0.78		
10	0.65	**20**	0.85	**30**	0.65	**40**	0.88		

Source: elaborated by the author.

It is observable that the predictions with the largest errors (index 5 and 16) presented in Figures 4.2, 4.3 and 4.7, were simulated with the same value of f (1.30).

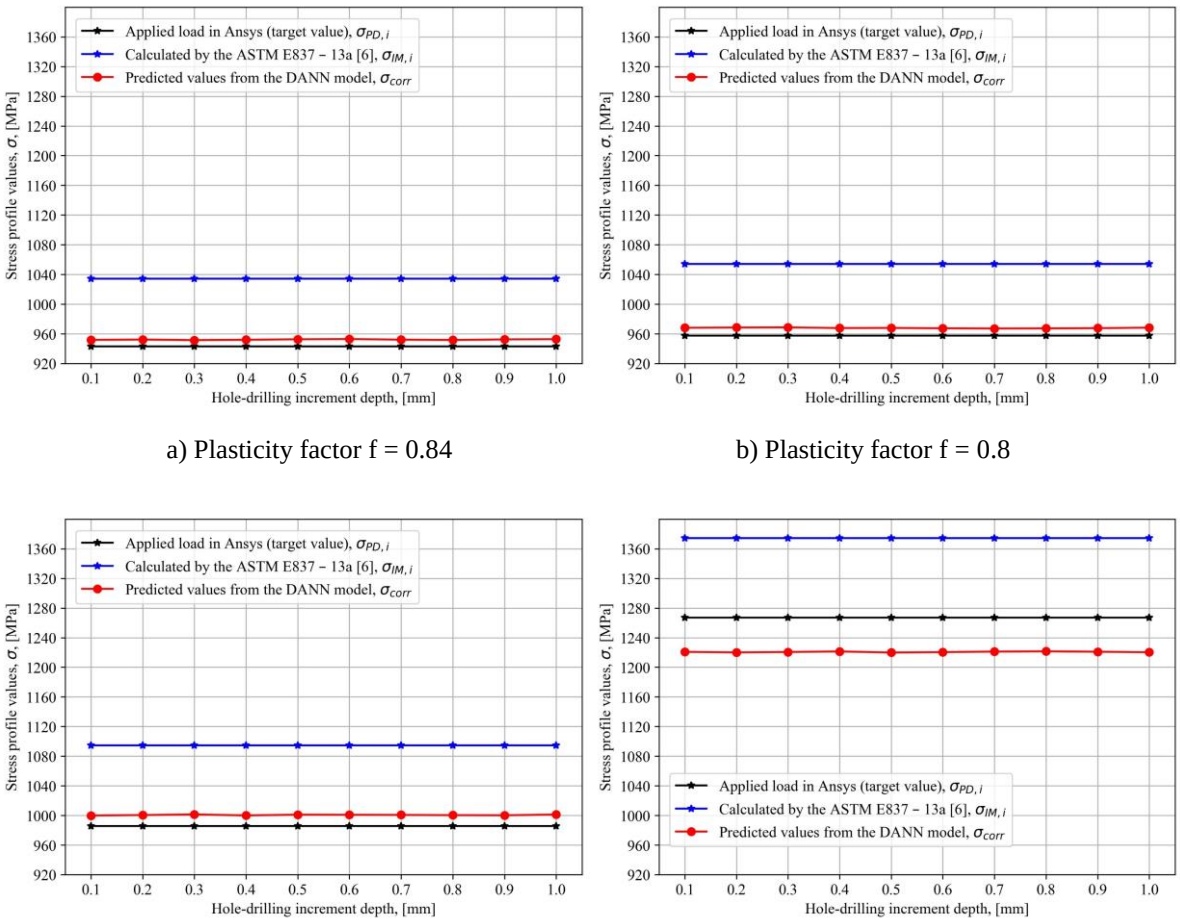
4.4 SELECTED PROFILES FROM THE SET OF DATA

This analysis was performed specifically to evaluate the behavior of the DANN model for a specific number of selected stress profiles (with selected f values).

It was notable in this thesis that no specific stress profile models (with specific f-value) were separated for the test set, the model training approach was designed on a per instance (hole-increment) basis, since for each increment a different simulation was performed. The idea of the implementation was to obtain a prediction for a given instance (hole-increment). The approach in using the DANN model then lies in the perspective of predicting specific values for more than a given stress profile. However, given the generality of the DANN trained model, it can with a certain amount of error fully predict a given profile.

In this sense, for exemplification purposes and not model validity, values of four profiles (with different values of f) were taken from the total database. The profiles were used as input and the correction was given as shown in the plots of Figures 4.8 and 4.9.

Figure 4.8 - Correction of the Stress profile values for a given Plasticity factor value - Database of Test 1



a) Plasticity factor f = 0.84

b) Plasticity factor f = 0.8

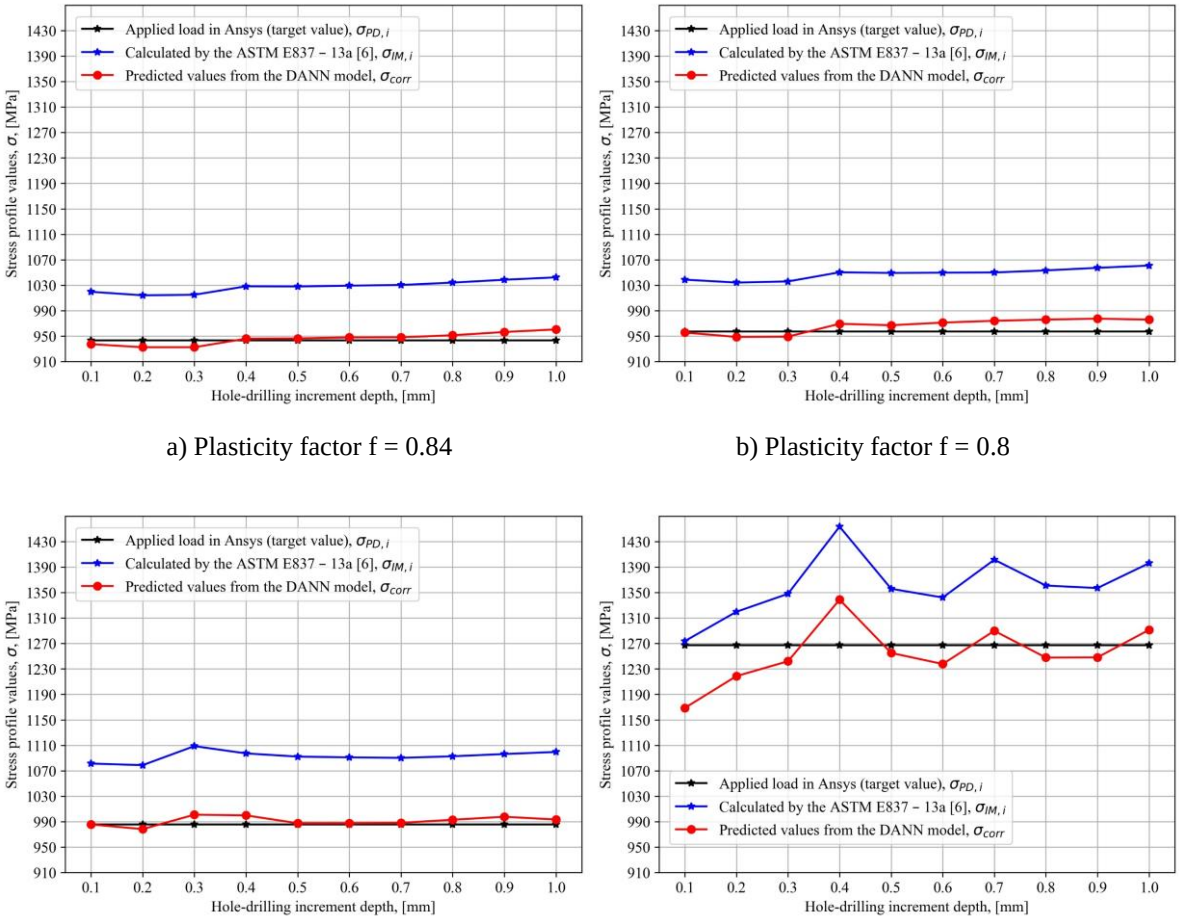
c) Plasticity factor f = 0.90

d) Plasticity factor f = 1.30

Source: elaborated by the author.

The selection of f-values however was based on the results of the analysis in the previous section. An investigation was made upon the values of f which presented peculiarities in the plots of Figures 4.2, 4.3 and 4.7.

Figure 4.9 - Correction of the Stress profile values for a given Plasticity factor value - Database of Test 2



a) Plasticity factor f = 0.84

b) Plasticity factor f = 0.8

c) Plasticity factor f = 0.90

d) Plasticity factor f = 1.30

It is also important to note that the approach here ended up using data that was used for training. Thus, the solutions shown in Figures 4.8 and 4.9 are most likely biased, but they served as examples to demonstrate the application of the DANN model in profile cases.

It is observed that for the plot of f = 1.30 in Figures 4.8(d) and 4.9(d), the error in the prediction is noticeable, which corresponds to the results of the plots in Figures 4.2, 4.3 and 4.7.

5 CONCLUSION AND SUGGESTIONS FOR FUTURE WORKS

In the work of this undergraduate thesis, a DANN model was developed and implemented to perform the correction of plasticity effects in the HDM when measuring high levels of residual stresses in flexible risers, given the limitations of the standard ASTM E837 – 13a [6] in the plastic regime. The model was trained on data resulting from several elastoplastic numerical simulations performed via FEM using Ansys Mechanical APDL. Two databases (named Test 1 and Test 2) were established for the training and implementation of the DANN model, one based on uniform stress values, and another based on non-uniform stress values along the hole wall. The pre-processing of the features of the databases was based on the work of Chupakhin *et al*. [7]. A classical MLR model was also implemented to compare the performance and errors of the DANN model. Both trained models were validated on the separated data set for testing, i.e., as per the split of 10/90 (test/training). That means that of the 420 instances (simulations), 42 instances (randomly chosen) were separated to test the models. The same procedure was done for Test 1 and Test 2.

In the validation of the test set of the two databases, both DANN models performed relatively accurately with prediction errors on the order of 4.4×10^{-3} for the database of Test 1 and 3.8×10^{-3} for the database of Test 2. In the prediction of the DANN trained model for the entire set of data, it was notable an apparent lower volume of data in the plot of Figure 4.4. This was due to the repetition in the database of uniform stress values along the hole (with the same f value), different from the Figure 4.5, where it had non-uniform stress values in the database. When it came to evaluate the performance of the DANN model by means of comparison with the MLR model, from the 42 predicted values in the test set of both Databases, it was found that the prediction error from the MLR model is in the same magnitude of the DANN model for the database of Test 1. On the other hand, the error in predicting values for the MLR model is lower than the DANN model for the database of Test 2, so that the error on average of the DANN model was about 1.2 times larger than the error of the MLR model. The result questioned the computational efficiency of implementing a DANN model, since the training time for the DANN model took 150 x longer than for the MLR model. The values of the Plasticity Factors f in the testing set were investigated. It was observed that the predictions with the largest errors (index 5 and 16) presented in Figures 4.2, 4.3 and 4.7, were simulated with the same value of f (1.30). For exemplification purposes and not model validity, values of four profiles (with different values of f) were taken from the total set of data from each database.

The profiles were used as input and the correction was given as presented in the plots of Figures 4.8 and 4.9. Although, the presented results were most likely biased, it served as an example to demonstrate the application of the DANN model in profile cases. Additionally, it was observed that for the plot of f = 1.30 in Figures 4.8(d) and 4.9(d), the presented error in the prediction corresponds to the results presented in the plots in Figures 4.2, 4.3 and 4.7.

Although considered good results have been obtained in the implementation of the DANN model for the correction of residual stress values affected by plastic effects, the trained model relies heavily on numerical results from controlled simulations under very specific conditions and parameters, which may make the trained model not perform well in generalization. Therefore, a validation via experimental data by means of comparison would be needed. Due to practical and resource difficulties, it was not possible to perform this procedure in this undergraduate thesis. A potential extension of the work done here would be to conduct experiments and collect measurements that could be compared to the predictions of the test sets of the developed models. This way, a greater depth in terms of validation and generality could be discussed, as well as the efficiency of the use of ML techniques in scientific model training.

Another important aspect was the low sampling of data for model training, in a way that both databases were considered poor datasets of 12 features for 420 instances. A fairly standard recommended in Data Science is in the proportion of 10 (features) to 20000 (instances) [18]. That raises again questions about the generality of the model.

The calculated features were based on the equations established by Chupakhin *et al*. [7], hence a larger exploration of statistical features could be performed on the raw data as a suggestion for a different approach, in order to have the displacement data resulting from the simulations as an input for the DANN model. Given a possibility to have a large database with numerous simulations (instances), it might be interesting to also perform a data split based on the predictions of profiles values rather than instance's values, in order to better evaluate the profile prediction. It is important to note that the trained DANN model is able to predict profiles, however it was not validated in this approach on the separate data set used for testing.

In this context and considering the results of the comparison between the MLR and DANN models, exploring other ML models to predict the values of residual stresses would be a great way to investigate better corrections and also to evaluate and decide on a simpler model, or in a model with considerable less processing (training) time. Example of ML regressor models that could be implemented are the Radom Forest regressor, the Decision Tree regressor and the PCA regressor.

Another extension of this thesis work that could be done in terms of training of the DANN model is the possibility of exploring the variation of the parameters established in Table 3.3. These parameters were established according to the approach of [7]. Future work recommendations would be to vary these parameters, evaluating the model error ranges in order to optimize such values. This procedure could be done with a fine-tuned parameter model, however the calculation and processing time may be expensive.

Another research approach would deeply investigate the relationship between the values of the Pasting Factor f and the predicted values from the models. Preliminary notes have been highlighted in sections 4.3 and 4.4 for some notable examples. But no deep investigation was done. Since the literature points to the beginning of plastic effects for $f > 0.6$, an investigation of this upper range with separate databases for testing could be carried out.

REFERENCES

[1] ALBERTAZZI, Jr, A.; VIOTTI, M. R. LABMETRO/EMC/UFSC. 5850.0108248.18.9 (Jurídico) e 4600568091 (SAP). **Relatório Técnico Parcial 1:** Período abrangido pelo relatório: 18/09/2018 a 18/08/2019, Florianópolis, 2019.

[2] MPCNEWS. **FMC Technologies and Petrobras Sign Memorandum of Understanding**. [S. l.], 1 Dec. 2010. Available at: https://offshoredaily.wordpress.com/2010/12/01/fmc-technologies-and-petrobras-sign-memorandum-of-understanding/. Accessed on: 10 Jan. 2022.

[3] VIVIANI, H. B. **Avaliação da aplicabilidade da deflectometria combinada com o método do contorno para medição de tensões residuais em arames de risers flexíveis**. 2020. Masters Dissertation – M.Sc., Mechanical Engineering, Department of Mechanical Engineering, Federal University of Santa Catarina, Florianópolis.

[4] NOV INC. **Flexible Pipe Products**. [S. l.], 2022. Available at: https://www.nov.com/products/flexible-pipe-products. Accessed on: 6 Feb. 2022.

[5] WILVERT, T. **A Methodology to Measure Residual Stresses Close to the Yield Stress**. 2021. Project of the Qualifying Exam for Doctorate Program in Mechanical Engineering, Department of Mechanical Engineering, Federal University of Santa Catarina, Florianópolis.

[6] ASTM. Determining Residual Stresses by the Hole-Drilling Strain-Gage Method. **Standard Test Method E837-13a**, v. i, pp. 1–16, 2013.

[7] CHUPAKHIN, S., et al. Artificial Neural Network for Correction of Effects of Plasticity in Equibiaxial Residual Stress Profiles Measured by Hole Drilling. **The Journal of Strain Analysis for Engineering Design**, Geesthacht, Germany, vol. 52, no. 3, pp. 137–15, 2017.

[8] IOWA STATE UNIVERSITY. Center for Nondestructive Evaluation. **Materials and Processes**: Stress and Strain. [S. l.], 2022. Available at: https://www.nde-ed.org/Physics/Materials/Mechanical/Loading.xhtml. Accessed on: 7 Feb. 2022.

[9] POPOV, E. P. **Engineering Mechanics of Solids**. 1. ed. New Jersey: Prentice-Hall, 1990.

[10] PROTO MANUFACTURING. **Residual Stress Info.** [S. l.], 2020. Available at: https://www.protoxrd.com/knowledge-center/residual-stress-info. Accessed on: 10 Dec. 2021.

[11] SCHAJER, G. S.; WHITEHEAD, P. S. **Hole-Drilling Method for Measuring Residual Stresses**. [s.l.] Morgan & Claypool, 2018. v. 1.

[12] KLOCKE, F. Finite Element Method (FEM). In: _____. (org.). **Manufacturing processes 1: turning, milling, drilling**. RWTH edition ed. Berlin: Springer, 2011. p. 197- 217.

[13] **ANSYS Mechanical APDL Structural Analysis Guide and Command Reference**, 2022. Available at: https://www.mm.bme.hu/~gyebro/files/ans_help_v182/ans_cmd/Hlp_C_CmdTOC.html. Accessed on: 10 Jan. 2022.

[14] ALBERTAZZI, Jr, A.; VIOTTI, M. R. LABMETRO/EMC/UFSC. **Relatório sobre a Malha utilizada nos estudos sobre o Método do Furo-Cego**. Florianópolis, 2020.

[15] DUNNE, F.; PETRINIC, N. **Introduction to Computational Plasticity**. Oxford: Oxford University Press, 2005.

[16] BEGHINI M., BERTINI L., SANTUS C. (2010). A procedure for evaluating high residual stresses using the blind hole drilling method, including the effect of plasticity. **The Journal of Strain Analysis for Engineering Design,** 45:301–317.

[17] NOBRE, J. P., *et al*. Plasticity Effects in the Hole-Drilling Residual Stress Measurement in Peened Surfaces. **Experimental Mechanics**, Johannesburg, South Africa, pp. 12, 2017.

[18] GÉRON, A. **Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 1st ed. Sebastopol, CA: O'Reilly Media, Inc., 2017.

[19] MATHWORKS. **What is a Neural Network?**: 3 things you need to know. [S. l.], 2020. Available at: https://www.mathworks.com/discovery/neural-network.html#how-they-work. Accessed on: 13 Nov. 2020.

[20] IBM. IBM Cloud Education. **What are Neural Networks?**: Neural Networks. [S. l.], 17 Aug. 2020. Available at: https://www.ibm.com/cloud/learn/neural-networks. Accessed on: 12 Dec. 2021.

[21] FERNÁNDEZ-CABÁN, P. L., *et al*. Predicting Roof Pressures on a Low-Rise Structure From Freestream Turbulence Using Artificial Neural Networks. **Frontiers in Built Environment**, Maryland, United States, pp. 16, 2018.

[22] LI, Y., *et al*. Hole-Drilling Method Eccentricity Error Correction Using a Convolutional Neural Network. **Experimental Mechanics**, Chengdu, Sichuan, China, pp. 36, 2020.

[23] THOMPSON, M. K.; THOMPSON, J. M. **ANSYS Mechanical APDL for Finite Element Analysis**. 1. ed. [s.l.] Butterworth-Heinemann, 2017.

[24] DEPARTMENT OF APPLIED MECHANICS MM. Faculty of Mechanical Engineering, Budapest University of Technology and Economics. **SOLID186**, 2021. Available at: https://www.mm.bme.hu/~gyebro/files/ans_help_v182/ans_elem/Hlp_E_SOLID186.html. Accessed on: 10 Feb. 2021.

[25] BROETTO, F. Z. **Avaliação numérica dos campos de deslocamento fora do plano devido ao alívio de tensões residuais pelos métodos do furo cego e furo dentro do furo**. 2015. Undergraduate thesis – B.Sc., Mechanical Engineering, Department of Mechanical Engineering, Federal University of Santa Catarina, Florianópolis.

[26] WILVERT, T. **Numerical assessment of the minimum distance between holes in 3D components and evaluation of the use of twist drills in the hole-drilling method**. 2018. Masters Dissertation – M.Sc., Mechanical Engineering, Department of Mechanical Engineering, Federal University of Santa Catarina, Florianópolis.

[27] SILVA, J. C. **Trabalho de começo de carreira: um guia de coaching para decolar na carreira com seu TCC**. 1st ed. Rio de Janeiro: Gramma, 2018.

[28] MENDELEY. **Modern Language Association**, 8th edition. Help Guides. Available at: https://www.mendeley.com/guides/mla-citation-guide/. Accessed on: 07 Apr. 2021.

[29] NORMAS ABNT. **Normas ABNT 2021 – pré-textuais, textuais e pós-textuais**. Available at: https://www.normasabnt.org/. Accessed on: 07 Apr. 2021.

Appendix A – Tikhonov regularization

As mentioned in section 2.2.1, the Eq. (2.23)-(2.25) from thee non-uniform procedure work well for a small number of increments. For a relatively large number of increments (i.e 10 or 20 increments) the calibration coefficient matrices become numerically ill-conditioned. Small errors in the measured strains could result in large errors for the stress values. The standard ASTM E837 – 13a [6] recommends the use of the Tikhonov regularization to reduce the errors generated in the calculation process.

The Tikhonov second-derivative regularization is implemented by augmenting Eq. (2.23)-(2-25), resulting in Eq. (A.1)-(A.3).

$$(\bar{\mathbf{a}}^{\mathbf{T}}\bar{\mathbf{a}} + \alpha_P \mathbf{c}^T\mathbf{c})\,\mathbf{P} = \frac{E}{(1+v)}\bar{\mathbf{a}}^{\mathbf{T}}\mathbf{p} \tag{A.1}$$

$$(\bar{\mathbf{b}}^{\mathbf{T}}\bar{\mathbf{b}} + \alpha_Q \mathbf{c}^T\mathbf{c})\,\mathbf{Q} = E\,\bar{\mathbf{b}}^{\mathbf{T}}\mathbf{q} \tag{A.2}$$

$$(\bar{\mathbf{b}}^{\mathbf{T}}\bar{\mathbf{b}} + \alpha_T \mathbf{c}^T\mathbf{c})\mathbf{T} = E\,\bar{\mathbf{b}}^{\mathbf{T}}\mathbf{t} \tag{A.3}$$

The matrix **c** represents a tri-diagonal "second derivative" matrix with rows filled with [-1 2 -1], except for the first and last rows. The number of rows corresponds to the number of hole increments used. Four k increments, for example, would demand the usage of the matrix **c** given in Eq. (A.4).

$$\mathbf{c_{kk}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ \mathbf{-1} & \mathbf{2} & \mathbf{-1} & 0 & 0 & \cdots & 0 \\ 0 & \mathbf{-1} & \mathbf{2} & \mathbf{-1} & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & ... & 0 & \mathbf{-1} & \mathbf{2} & \mathbf{-1} & 0 \\ 0 & \cdots & 0 & 0 & \mathbf{-1} & \mathbf{2} & \mathbf{-1} \\ 0 & \cdots & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.4}$$

The parameters α_P, α_Q, and α_T govern the amount of regularization (smoothing). The standard recommends a set interval for these parameters. To update these values and perform the convergence criterion, an iterative technique is proposed. Finally, the combination stresses P, Q, and T are used to calculate the regularized stresses for each increment j.

The calibration coefficients for the three rosette types, A, B, and C (introduced in section 2.2), are included in the standard. The calibration coefficients are used in the

mathematical representation of the relieved strains around the hole. The relieved strains at the surface following the drilling operation can be represented by Eq. (A.5) for the uniform method and Eq. (A.6) for the non-uniform method for a k-increment hole.

$$\varepsilon = \frac{1+v}{2E}\bar{\mathbf{a}}\left(\sigma_x + \sigma_y\right) + \frac{1}{2E}\bar{\mathbf{b}}\left(\left(\sigma_x - \sigma_y\right)\cos(2\theta) + 2\left(\tau_{xy}\right)\sin(2\theta)\right) \tag{A.5}$$

$$\begin{pmatrix}\varepsilon_1\\ \varepsilon_2\\ \vdots\\ \varepsilon_j\end{pmatrix} = \frac{1+\mathrm{v}}{2\mathrm{E}}\begin{bmatrix}\mathbf{a_{11}} & \mathbf{0} & \cdots & \mathbf{0}\\ \mathbf{a_{12}} & \mathbf{a_{22}} & & \mathbf{0}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{a_{1k}} & \mathbf{a_{2k}} & \cdots & \mathbf{a_{jk}}\end{bmatrix}\begin{pmatrix}(\sigma_y)_1 + (\sigma_x)_1\\ (\sigma_y)_2 + (\sigma_x)_2\\ \vdots\\ (\sigma_y)_k + (\sigma_x)_k\end{pmatrix}$$

$$+ \frac{1}{2\mathrm{E}}\begin{bmatrix}b_{11} & 0 & \cdots & 0\\ b_{12} & b_{22} & & 0\\ \vdots & \vdots & \ddots & \vdots\\ b_{1k} & b_{2k} & \cdots & b_{jk}\end{bmatrix}\left(\begin{pmatrix}(\sigma_y)_1 - (\sigma_x)_1\\ (\sigma_y)_2 - (\sigma_x)_2\\ \vdots\\ (\sigma_y)_k - (\sigma_x)_k\end{pmatrix}\cos(2\theta)\right.$$

$$\left. + 2\begin{pmatrix}(\tau_{xy})_1\\ (\tau_{xy})_2\\ \vdots\\ (\tau_{xy})_j\end{pmatrix}\sin(2\theta)\right) \tag{A.6}$$

Appendix B – Developed code in APDL for the execution of simulations

The script below was developed for the execution of the numerical simulations in the Ansys Mechanical APDL software. The script was coded in Ansys APDL parametric language (Fortran based).

txt_inc20.f

```
!Mechanical APDL（ANSYS）
→PlotCtrls→Style→Color→Reverse Video
fini
/clear
/CWD,'D:\lucas.manasses\Changes_of_the_mesh_a
t_the_model_extremity'!SAVE IN D:
/filname, Inc20NewMesh

/prep7

mp,ex,1,200e9

mp,prxy,1,0.30

sy=250e6

tm=10e9

tb,bkin,1,,0,
tbtemp,0.0
tbdata,1,sy,tm

/axlab,x,log strain (mm/mm)
/axlab,y,true stress (N/mm^2)

tbpl,bkin,1

et,1,186

!variables
inc=20
rf=0.9
ra=7.5
c=1e-4
pro=1
n=20
esp=6
h=pro/n
pi=3.14159265359
ea=1
eb=1.2
ec=2
ed=5

!creation of geometry
cylind,0,rf,esp-inc*h,esp,0,90!volume 1
cylind,rf,1.1*rf,esp-eb,esp,0,90!volume 3
cylind,0,rf,esp-eb,esp-inc*h,0,90!volume 2
cylind,1.1*rf,5*rf,esp-eb,esp,0,90!volume 5
cylind,0,1.1*rf,esp-ec,esp-eb,0,90!volume 4
cylind,1.1*rf,5*rf,esp-ec,esp-eb,0,90!volume 6

!cylind,5*rf,ra,esp-eb,esp,0,90!volume 7
!cylind,5*rf,ra,esp-ec,esp-eb,0,90!volume 8
cylind,0,1.1*rf,0,esp-ec,0,90!volume 9
cylind,1.1*rf,5*rf,0,esp-ec,0,90!volume 10
cylind,5*rf,6*rf,0,esp,0,90!volume ** !Coarse
mesh at the boundary 'thiago` 20210325
cylind,0,1.1*rf,0,esp-ec-rf,0,90!volume ** 'thiago`
20210325

cylind,1.1*rf,5*rf,0,esp-ec-
rf,0,90!volume**'thiago` 20210325
cylind,5*rf,6*rf,0,esp-ec-rf,0,90!volume**'thiago`
20210325
cylind,0,1.1*rf,0,esp-ec-rf,0,90!volume**'thiago`
20210325
cylind,5*rf,ra,0,esp-ec,0,90!volume 11
block,0,ra,0,ra,esp-eb,esp!volume 7
block,0,ra,0,ra,esp-ec,esp-eb!volume 8
block,0,ra,0,ra,0,esp-ec

vovlap,all
vsel,all
vglue,all

!Selection of volumes for MESH

!1ºvolume(V18)(folhinha)
asel,s,loc,x,0
asel,r,loc,z,esp-eb,esp-inc*h
asel,r,loc,y,0,rf
*get,area_source,area,,num,max
vsla
*get,volume_sweep,volu,,num,max
asel,s,loc,y,0
asel,r,loc,z,esp-eb,esp-inc*h
asel,r,loc,x,0,rf
*get,area_target,area,,num,max
esize,0.05
mat,1
vsweep,volume_sweep,area_source,area_target,1
```

```
!2ºvolume(V01)
asel,s,loc,x,0
asel,r,loc,z,esp-inc*h,esp
asel,r,loc,y,0,rf
*get,area_source,area,,num,max
vsla
*get,volume_sweep,volu,,num,max
asel,s,loc,y,0
asel,r,loc,z,esp-inc*h,esp
asel,r,loc,x,0,rf
*get,area_target,area,,num,max
esize,0.05
mat,1
vsweep,volume_sweep,area_source,area_target,1

!3ºvolume(V31)
asel,s,loc,x,0
asel,r,loc,z,esp-eb,esp
asel,r,loc,y,rf,1.1*rf
*get,area_source,area,,num,max
vsla
*get,volume_sweep,volu,,num,max
asel,s,loc,y,0
asel,r,loc,z,esp-eb,esp
asel,r,loc,x,rf,1.1*rf
*get,area_target,area,,num,max
esize,0.05
mat,1
vsweep,volume_sweep,area_source,area_target,1

!4ºvolume(V22)
asel,s,loc,x,0
asel,r,loc,z,esp-ec,esp-eb
asel,r,loc,y,0,1.1*rf
*get,area_source,area,,num,max
vsla
*get,volume_sweep,volu,,num,max
asel,s,loc,y,0
asel,r,loc,z,esp-ec,esp-eb
asel,r,loc,x,0,1.1*rf
*get,area_target,area,,num,max
esize,0.2
mat,1
vsweep,volume_sweep,area_source,area_target,1

!5ºvolume(V23)
asel,s,loc,x,0
asel,r,loc,z,esp-eb,esp
asel,r,loc,y,1.1*rf,5*rf
*get,area_source,area,,num,max
vsla
*get,volume_sweep,volu,,num,max
asel,s,loc,y,0
asel,r,loc,z,esp-eb,esp
asel,r,loc,x,1.1*rf,5*rf
*get,area_target,area,,num,max
esize,0.2
mat,1
vsweep,volume_sweep,area_source,area_target,1

!6ºvolume(V25)
asel,s,loc,x,0
asel,r,loc,z,esp-ec,esp-eb
asel,r,loc,y,1.1*rf,5*rf
*get,area_source,area,,num,max
vsla
*get,volume_sweep,volu,,num,max
asel,s,loc,y,0
asel,r,loc,z,esp-ec,esp-eb
asel,r,loc,x,1.1*rf,5*rf
*get,area_target,area,,num,max
esize,0.2
mat,1
vsweep,volume_sweep,area_source,area_target,1

!FIRST cOARSE Mesh (VADD E VMESH COM
ESIZE)
!The goal here is to select the volumes and glue
them for generation the later mesh

!Selecting V20,V27,V28,V21,V32
asel,s,loc,z,0
vsla
!Gluing V20,V27,V28,V21,V32 -> V02
VADD, ALL

asel,s,loc,x,ra
vsla
!Gluing V02,V34,V33 -> V03
VADD, ALL

! V03 is the COARSE Mesh Volume
mshape,1 !Tetrahedral (coarse mesh)
!smrtsize,10
esize,1.5  !4
vmesh,all

!SECOND INTERMEDIATE MESH (SMRTSIZE,
,, 1.2,1.3,,,1.2,,,20)

!selecting volumes V19,V29,V30
asel,s,loc,y,0
asel,r,loc,z,esp-ec-rf,esp-ec
vsla
!Gluing V19,V29,V30 -> V02
VADD, ALL

!selecting volumes V24,V26,V02
asel,s,loc,y,0
asel,r,loc,x,5*rf,6*rf
asel,r,loc,z,esp-ec-rf,esp
vsla
!Gluing V24,V26,V02 -> V04
VADD, ALL

mshape,1 !Tetrahedral (coarse mesh)
smrtsize,10
```

```
esize,1.5
vmesh,all
!------------25.04.2021

lsel,all,all

/solve
!nlgeom,on !03.05.2021
!nsubst,10,20,9

nropt,full

ANTYPE,STATIC
!KBC,1
nsel,all

nsel,s,loc,x,0
d,all,ux,0

nsel,s,loc,y,0
d,all,uy,0

nsel,s,loc,x,0
nsel,r,loc,y,0
nsel,r,loc,z,0
d,all,uz,0

nsel,all,all
!Sigma x
asel,s,loc,x,ra
sfa,all,1,pres,-237.5e6

!Sigma y
asel,s,loc,y,ra
sfa,all,1,pres,-237.5e6

allsel
solve

TIME,1
AUTOTS,ON
OUTRES,ALL,ALL
OUTPR,ALL,ALL

!!!PART 1

nsel,s,loc,x,0
d,all,ux,0

nsel,s,loc,y,0
d,all,uy,0

nsel,s,loc,x,0
nsel,r,loc,y,0
nsel,r,loc,z,0
d,all,uz,0

nsel,all,all
!Sigma x
asel,s,loc,x,ra
sfa,all,1,pres,-237.5e6

!Sigma y
asel,s,loc,y,ra
sfa,all,1,pres,-237.5e6

allsel
! vsel,s,loc,z,esp-inc*h+0.001,esp
! vsel,r,loc,x,0,rf-0.001
! vsel,r,loc,y,0,rf-0.001
asel,s,loc,x,0
asel,r,loc,y,0,rf
asel,r,loc,z,esp-inc*h,esp
vsla
! vsel,r,loc,z,esp-inc*h,esp
! vsel,r,loc,y,0,rf

eslv,s
ekill,all
allsel

solve

TIME,2
AUTOTS,ON
OUTRES,ALL,ALL
OUTPR,ALL,ALL

!PART 2

save

/post1

lcdef,1,1

lcdef,2,2

lcase,2
lcsum,all
lcoper,sub,1
lcwrite,3,sub,,,
rappnd,3

! ! set,1
! lcwrite,2,inc2,,,
! lcfile,1,Semfuro,L01,,
! !lcfile,2,inc2,L02,,
! lcase,2
! lcsum,all
! lcoper,sub,1
! lcwrite,3,sub,,,
! !manually procedure for picture with subtraction
cases===load case->read load case->3->->plot
results->nodal solu->x-component of displacement

nsel,s,loc,z,esp
```

```
allsel
*get,highestnode,node,,num,max
*get,lowestnode,node,,num,min
*get,nnodesl,node,,count

*dim,u20,array,nnodesl,5
*vget,u20(1,1),node,lowestnode,loc,x
*vget,u20(1,2),node,lowestnode,loc,y
*vget,u20(1,3),node,lowestnode,loc,z
*vget,u20(1,4),node,lowestnode,u,x
*vget,u20(1,5),node,lowestnode,u,y
*cfopen,u20NewMesh,txt
```

```
/INPUT,'vwriteu20','txt','D:\lucas.manasses\Changes_of_the_mesh_at_the_model_extremity\vwrite',, 0
*cfclos
!VISUALIZATION OF IMAGE WITH HOLE
! vsel,s,volu,,1
! eslv,r
! esel,inve
! !postprocvse
! plesol,eppl,eqv
! !
! *get,deformacao,node,105,nl,epeq
!!!!!nsubstep
!
```

Appendix C – Uniform stress for each plasticity factor f

In this appendix all values of f employed in the execution of the simulation are described in the Table C.1. In total, 420 simulations were run, where every 10 simulations, a different value of f was implemented. The maximum and minimum uniform stress values calculated according to the standard ASTM E837 – 13a [6] are also indicated.

Table C.1 - Values of Plasticity factor (or Beghini's factor) with the corresponded applied load value and the computed stress value.

Plasticity factor	**Applied Load (MPa)**		**Uniform stress (MPa)**	
f	$\boldsymbol{\sigma_x\,(=\sigma_{eq})}$	$\boldsymbol{\sigma_y}$	$\boldsymbol{\sigma_{max}}$	$\boldsymbol{\sigma_{min}}$
0	352	0	349,7069683	1,47126896
0,1	422,4	0	419,809483	1,68476143
0,2	492,8	0	490,5131412	1,639983837
0,3	563,2	0	562,7021093	0,85623642
0,4	633,6	0	637,6384662	-1,634727912
0,5	704	0	717,8128944	-8,610631337
0,6	774,4	0	805,2141098	-27,13842231
0,65	809,6	0	848,3095345	-42,329725
0,7	844,8	0	890,0440063	-44,06305479
0,71	851,84	0	898,5792458	-46,78401588
0,72	858,88	0	907,6852037	-48,74534962
0,73	865,92	0	918,4227485	-52,27185151
0,74	872,96	0	927,7054105	-55,2402064
0,75	880	0	936,9825089	-59,10588486
0,76	887,04	0	949,0598485	-63,50716498
0,77	894,08	0	958,922451	-66,3351702
0,78	901,12	0	969,1141733	-70,16934398
0,79	908,16	0	980,5003715	-75,100428

0,8	915,2	0	991,6032579	-80,23519783
0,81	922,24	0	1001,480131	-84,26795077
0,82	929,28	0	1012,154016	-88,62940229
0,83	936,32	0	1021,694208	-93,39535089
0,84	943,36	0	1034,51216	-98,59004274
0,85	950,4	0	1043,279677	-101,401651
0,86	957,44	0	1053,97902	-106,757169
0,87	964,48	0	1064,528978	-112,4734972
0,88	971,52	0	1074,329156	-116,5886213
0,89	978,56	0	1085,084838	-122,1303503
0,9	985,6	0	1094,754304	-126,5403038
0,92	999,68	0	1116,785984	-137,9541861
0,93	1006,72	0	1127,249117	-143,4502754
0,94	1013,76	0	1138,955728	-150,5254545
0,95	1020,8	0	1150,298816	-157,6108823
0,96	1027,84	0	1160,358805	-162,9026119
0,97	1034,88	0	1170,852424	-169,4461654
0,98	1041,92	0	1179,908026	-174,9170509
0,99	1048,96	0	1188,312025	-179,6897325
1	1056	0	1198,685306	-186,1795022
1,05	1091,2	0	1234,838518	-210,8168977
1,1	1126,4	0	1263,195975	-222,6001125
1,25	1232	0	1353,576318	-220,1808896
1,3	1267,2	0	1374,204619	-209,680939

Source: elaborated by the author.

Appendix D – Code for the computation of the Stress values

The following MATLAB script is the main file for the computation of the stress values of the standard ASTM E837 – 13a [6]. The script was developed by the LABMETRO researchers. This code computes the stress values as a function of the deformations, which are also calculated as a function of the supplied displacements. The code was provided by LABMETRO and adapted to the use case of this undergraduate thesis. The computation of the stress values was made according to the Eq. (2.9)-(2.29) shown in Section 2.2.1.

The calibration coefficients (mentioned in Section 2.2) are implemented in the code according to the standard ASTM E837 – 13a [6]. Uniform as well as non-uniform stresses were calculated, even though a constant stress was applied along the hole wall of the model.

main_code.m

```
%Main code
clear all;close all

%DF=[9,10,12,16,20,24,28,32,40,50,60,70,132,136,144,148,150,151];%simultaneous
%DF=[9,10,12,16,20,24,28,32,40,50,60,70,132,136,140,144,148,150,151];%preexistinghole
DF=20;%Just for analysing vv
[line, column]=size(DF);
INC=(0.1:0.1:1); %APDL subtraction (no reference)% %0:0.1:1
ref=1;%Harmonic %Reference value (12=df70) vv
%%Saving the raw data from apdl txt's%%
for n_df=1:1:column
%address=['F:\simulation\biaxial_r1mm\thickcase_surface_inc\df',num2str(DF(n_df)/2), 'mm'];
address=['C:\Users\thiag\OneDrive\Área de Trabalho\Cálculo método não-uniforme'];
%address=['E:\simulationdistance3d\harmonicosplasticidade\Inc',num2str(DF(n_df))];
createobject_str = ['df',int2str(DF(n_df)),' =processsurfacedata;'];
cd('C:\Users\thiag\OneDrive\Área de Trabalho\Cálculo método não-uniforme');
eval(createobject_str); % Creating the object (Instance of the class)

    for n_inc=1:1:10 %APDL subtraction (no reference)% %0:1:10
     cd(address);
    %fileID=fopen(['datasurf_inc_',num2str(INC(n_inc+1)),'mm.txt']);%INC(n_inc+1) because n_inc must vary between 0 and 10
    fileID=fopen(['u',num2str(INC(n_inc)*10),'.txt']);%INC(n_inc) because n_inc must vary between 1 and 10.There is no reference in this case
    file=fscanf(fileID,'%f %f %f %f %f ',[5 inf]);
    fclose(fileID);

    %m=['df',int2str(DF(n_df)),'.datainc{',num2str(n_inc+1),'} = transpose(file);']; %INC(n_inc+1) because n_inc must vary between 0 and 10
    m=['df',int2str(DF(n_df)),'.datainc{',num2str(n_inc),'} = transpose(file);']; %INC(n_inc) harmonic

    cd('C:\Users\thiag\OneDrive\Área de Trabalho\Cálculo método não-uniforme');
    eval(m);
```

```
      end%for n_inc=1:1:10
end %for n_df=1:1:column
%%

%Calling funcprocesssurfacedata for every object
 cd('C:\Users\thiag\OneDrive\Área de Trabalho\Cálculo método não-uniforme');
for n_df=1:1:column
m=['df',int2str(DF(n_df)),'.funcprocesssurfacedata;']; %INC(n_inc+1) because n_inc must vary between 0 and 10
      eval(m); %df10.funcprocesssurfacedata; df(x)
end% for n_df=1:1:column
m;

%Linear model(Works for every rosette position)
for n_df=1:1:column
m=['df',int2str(DF(n_df)),'.interpfunc;']; %INC(n_inc+1) because n_inc must vary between 0 and 10
eval(m); %df10.funcprocesssurfacedata; df(x)

end
%% Comands employed to load the raw data
   %cd('F:\simulationdistance3d\OOP treino');
   %save('preexistinghole')% load('preexistinghole') %equibiaxial (F:\simulationdistance3d\OOP treino)
   %save('pureshear')%load('pureshear') %preexisting hole pure shear
   %save('uniaxialy')%load('uniaxialy')% PREEXISTING HOLE UNIAXIAL Y
   %save('uniaxialx')%load('uniaxialx')%PREEXISTING HOLE UNIAXIAL X
%theta1=-90; theta2=-45;theta3=0;%G2 %insert theta values here (0,45,90 means a type b in the first quadrant)
%theta1=0; theta2=45;theta3=90;%G1
%%
%Parameterizing the SG angle to vary from G1 to G2 at increments of 10°
count=1;theta_inc=0;
         %for theta_inc=0:1:360 ;  %180:5:540 hole-hole / 0:5:360 hole-edge / THIS LOOP MUST BE REMOVED
IN ORDER TO ANALYSE THE STRESSES AS A FUNCTION OF DF
         %theta1=-135+theta_inc; theta2=-90+theta_inc;theta3=-45+theta_inc;
         theta1=0+theta_inc; theta2=45+theta_inc;theta3=90+theta_inc;
%% Creating the threee gages for the Rosette 1 (I can leave this so that it will be possible to change these values
whenever it is necessary)

   for n_df=1:1:column
   m=['df',int2str(DF(n_df)),'.sgrotation(',int2str(theta1),',',int2str(theta2),',',int2str(theta3),');'];
%dfxx.sgrotation(-90,-45,0)%Gage location having the point (0,0) as a reference)
   eval(m);
%Gage location considering the position of the hole df(x,y))
  x=40;
   df=DF(n_df);
   y=40+df/4;
   x=0;y=0;

m=['[SIGMA_MAX(n_df),SIGMA_MIN(n_df),SHEARXY(n_df),G2_E(n_df)]=df',int2str(DF(n_df)),'.sgdisp(n
_df,x,y,',int2str(theta1),',',int2str(theta2),',',int2str(theta3),');'];%n_inc should vary from 2 to 11 because n_inc=1
represents the no hole case
%m=['[G2_E(n_df)]=df',int2str(DF(n_df)),'.sgdisp(n_df,x,y,',int2str(theta1),',',int2str(theta2),',',int2str(theta3),');'
];%n_inc should vary from 2 to 11 because n_inc=1 represents the no hole case
   %FORNECER THETA1,2 E 3
   eval(m)
   end % for n_df=1:1:column

 %Non-uniform
 for n_df=1:1:column
    %F from first increment. L from last increment. A from average
```

```
m=['[FSIGMA_MAXN(n_df),FSIGMA_MINN(n_df),LSIGMA_MAXN(n_df),LSIGMA_MINN(n_df),ASIGMA_MAXN(n_df),ASIGMA_MINN(n_df)]=df',int2str(DF(n_df)),'.nuni;'];%n_inc should vary from 2 to 11 because n_inc=1 represents the no hole case

    val=DF(n_df);
    %m=['df',int2str(DF(n_df)),'.nuni'];
    %FORNECER THETA1,2 E 3
  eval(m)
  %pause(2)
  end % for n_df=1:1:column
%Stress non-uniform (f first inc, L last, A average)
```

Appendix E – Scripts in Python for the preprocessing of data

The following scripts were developed in python language for the preprocessing of the uniform and non-uniform voltage data coming from the excel spreadsheets files.

preprocessing_data_teste_01.py

```
"""
Created on Fri Dec 3 10:38:25 2021
@author: Lucas Manassés
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#list_f = ['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.75','f0.8','f0.85','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.98','f0.99','f1','f1.05','f1.1']
list_f = ['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.71','f0.72','f0.73','f0.74','f0.75','f0.76','f0.77','f0.78','f0.79','f0.8','f0.81','f0.82','f0.83','f0.84','f0.85','f0.86','f0.87','f0.88','f0.89','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.98','f0.99','f1','f1.05','f1.1','f1.25','f1.3']

df_1 = pd.DataFrame()

#df_info = pd.read_excel (r'C:\Users\twi\Downloads\Uniaxial_dataV1.xlsx',sheet_name='INFORMATION')
#df_info = pd.read_excel (r'C:\Users\twi\Downloads\Uniaxial_dataV2.xlsx',sheet_name='Planilha1')
#df_info = pd.read_excel (r'D:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name='Planilha1')
df_info = pd.read_excel (r'G:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name='Planilha1')

# filtered
df_info = df_info.iloc[1:43,0:5]
df_info = df_info.reset_index(drop=True)

#%%
# Chupakhin - Data science approach

#list_f = ['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.75','f0.8','f0.85','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.98','f0.99','f1','f1.05','f1.1']
list_f = ['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.71','f0.72','f0.73','f0.74','f0.75','f0.76','f0.77','f0.78','f0.79','f0.8','f0.81','f0.82','f0.83','f0.84','f0.85','f0.86','f0.87','f0.88','f0.89','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.98','f0.99','f1','f1.05','f1.1','f1.25','f1.3']

df_1 = pd.DataFrame()

E = 207.5e3 # MPa
Sigma_yeld = 1056  # MPa

for f_cont in range(len(list_f)):
    #df_f = pd.read_excel (r'C:\Users\twi\Downloads\Uniaxial_dataV2.xlsx',sheet_name=list_f[f_cont])
    #df_f = pd.read_excel (r'D:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name=list_f[f_cont])
    df_f = pd.read_excel (r'G:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name=list_f[f_cont])
```

```
df_f_inc_depth = df_f.iloc[17:27,0:6]
df_f_inc_depth = df_f_inc_depth.reset_index(drop=True)

#df_f_strains = df_f.iloc[34:44,3:5]
#df_f_strains = df_f_strains.reset_index(drop=True)

ppts = np.empty(10); ppts.fill(Sigma_yeld/(E*1e-2)) # Chupakhin -> feature Sigma_yeld / E*10^-2
ft_ppts = pd.DataFrame(ppts)
ft_ppts = ft_ppts.reset_index(drop=True)

df_f_unif_stress = df_f.iloc[30:31,2:4]
stress = df_f_unif_stress.to_numpy()
sigma_IM = np.empty(10); sigma_IM.fill(stress[0][0]) # IN OTHER TESTS WE CHANGE HERE
df_f_unif_sigma_IM = pd.DataFrame(sigma_IM)
stress_min = np.empty(10); stress_min.fill(stress[0][1]) # IN OTHER TESTS WE CHANGE HERE
df_f_unif_stress_min = pd.DataFrame(stress_min)

sigma_IM_10 = df_f.iloc[30:31,2:3]
sigma_IM_10 = sigma_IM_10.to_numpy()
sigma_IM_10_incs = np.empty(10); sigma_IM_10_incs.fill(sigma_IM_10[0][0]) # IN OTHER TESTS WE
CHANGE HERE
sigma_IM_10 = pd.DataFrame(sigma_IM_10_incs)

###
mat =[]
for l in range(10): # line -> increment
   #print(l)
   row = []
   for c in range(9): # column -> feature -> sigma_IM*
      sigma_IM_star = (sigma_IM[c] + 2*Sigma_yeld)/(sigma_IM_10_incs[c] + 2*Sigma_yeld)
      row.append(sigma_IM_star)
   sigma_IM_star = (sigma_IM_10_incs[0] + 2*Sigma_yeld)/(2.2*Sigma_yeld)
   row.append(sigma_IM_star)
   mat.append(row)

###

sigma_IM_star_matrix = pd.DataFrame(mat)

label = df_info.iloc[f_cont:f_cont+1,0:1]
label = label.to_numpy()
label_incs = np.empty(10); label_incs.fill(label[0][0]) # IN OTHER TESTS WE CHANGE HERE
label = pd.DataFrame(label_incs)
#label = label.reset_index(drop=True)

label_sigma_PD_incs = np.empty(10);
for i in range(10):
   label_sigma_PD_incs[i] = (sigma_IM[i] + 2*Sigma_yeld)/(label_incs[i] + 2*Sigma_yeld)
label_sigma_PD_incs = pd.DataFrame(label_sigma_PD_incs)

f_id = df_info.iloc[f_cont:f_cont+1,2:3]
f_id = f_id.to_numpy()
f_id_incs = np.empty(10); f_id_incs.fill(f_id[0][0])
f_id = pd.DataFrame(f_id_incs)

###
###
# depth = i/10
```

```
    df_f = pd.concat([df_f_inc_depth, ft_ppts, sigma_IM_star_matrix, df_f_unif_sigma_IM, df_f_unif_stress_min, sigma_IM_10, f_id, label_sigma_PD_incs], axis=1)  # sigma_IM := stress_max
    df_f.columns = ['Inc','i/10','stress_max_non_uni_reg','stress_min_non_uni_reg','stress_max_non_uni_non_reg','stress_min_non_uni_non_reg','Sigma_yeld/E10-2','sigma_IM_1*','sigma_IM_2*','sigma_IM_3*','sigma_IM_4*','sigma_IM_5*','sigma_IM_6*','sigma_IM_7*','sigma_IM_8*','sigma_IM_9*', '(sigma_IM_10+2*Sigma_yeld)/(2.2*Sigma_yeld)', 'sigma_IM','stress_min','sigma_IM_10','f','label_sigma_PD_incs']
    df_2 = df_f

    df_f = pd.concat([df_1,df_2], axis=0)

    df_1 = df_f

df_f = df_f.reset_index(drop=True)

#df_f.to_csv(r'C:\Users\twi\Desktop\TC_2\A04\pre_processing\dataset_uniform_stress_ds_approach.csv',index=False)

# filters

df_f.drop(['stress_max_non_uni_reg','stress_min_non_uni_reg','stress_max_non_uni_non_reg','stress_min_non_uni_non_reg','stress_min','f'], axis='columns',inplace=True)

# reordering

df_f = df_f[['sigma_IM_1*','sigma_IM_2*','sigma_IM_3*','sigma_IM_4*','sigma_IM_5*','sigma_IM_6*','sigma_IM_7*','sigma_IM_8*','sigma_IM_9*', '(sigma_IM_10+2*Sigma_yeld)/(2.2*Sigma_yeld)', 'Sigma_yeld/E10-2','i/10','label_sigma_PD_incs']]

df_f.to_csv(r'C:\Users\twi\Desktop\TC_2\A04\pre_processing\dataset_uniform_stress_test_1_all_simulations.csv',index=False)
```

---

preprocessing_data_teste_02.py

```
"""
Created on Fri Dec 3 10:38:25 2021
@author: Lucas Manassés
"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#list_f = ['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.75','f0.8','f0.85','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.98','f0.99','f1','f1.05','f1.1']
list_f = ['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.71','f0.72','f0.73','f0.74','f0.75','f0.76','f0.77','f0.78','f0.79','f0.8','f0.81','f0.82','f0.83','f0.84','f0.85','f0.86','f0.87','f0.88','f0.89','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.98','f0.99','f1','f1.05','f1.1','f1.25','f1.3']

df_1 = pd.DataFrame()
```

```
#df_info = pd.read_excel (r'C:\Users\twi\Downloads\Uniaxial_dataV1.xlsx',sheet_name='INFORMATION')
#df_info = pd.read_excel (r'C:\Users\twi\Downloads\Uniaxial_dataV2.xlsx',sheet_name='Planilha1')
#df_info = pd.read_excel (r'D:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name='Planilha1')
df_info = pd.read_excel (r'G:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name='Planilha1')

# filtered
df_info = df_info.iloc[1:43,0:5]
df_info = df_info.reset_index(drop=True)

#%%
#list_f =
['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.75','f0.8','f0.85','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97
','f0.98','f0.99','f1','f1.05','f1.1']
list_f =
['f0','f0.1','f0.2','f0.3','f0.4','f0.5','f0.6','f0.65','f0.7','f0.71','f0.72','f0.73','f0.74','f0.75','f0.76','f0.77','f0.78','f0.79','f0.
8','f0.81','f0.82','f0.83','f0.84','f0.85','f0.86','f0.87','f0.88','f0.89','f0.9','f0.92','f0.93','f0.94','f0.95','f0.96','f0.97','f0.9
8','f0.99','f1','f1.05','f1.1','f1.25','f1.3']

df_1 = pd.DataFrame()

E = 207.5e3 # MPa
Sigma_yeld = 1056  # MPa

for f_cont in range(len(list_f)):

    #df_f = pd.read_excel (r'C:\Users\twi\Downloads\Uniaxial_dataV2.xlsx',sheet_name=list_f[f_cont])
    #df_f = pd.read_excel (r'D:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name=list_f[f_cont])
    df_f = pd.read_excel (r'G:\PC_LABMETRO\TC_2\A04\Uniaxial_dataV2.xlsx',sheet_name=list_f[f_cont])

    df_f_inc_depth = df_f.iloc[17:27,0:6]
    df_f_inc_depth = df_f_inc_depth.reset_index(drop=True)

    #df_f_strains = df_f.iloc[34:44,3:5]
    #df_f_strains = df_f_strains.reset_index(drop=True)

    ppts = np.empty(10); ppts.fill(Sigma_yeld/(E*1e-2)) # Chupakhin -> feature Sigma_yeld / E*10^-2
    ft_ppts = pd.DataFrame(ppts)
    ft_ppts = ft_ppts.reset_index(drop=True)

    df_f_unif_stress = df_f.iloc[17:27,6:8]
    stress = df_f_unif_stress.to_numpy()
    sigma_IM = np.empty(10); sigma_IM = stress[:,0]
    #sigma_IM = np.empty(10); sigma_IM.fill(stress[0][0]) # IN OTHER TESTS WE CHANGE HERE
    df_f_unif_sigma_IM = pd.DataFrame(sigma_IM)

    #stress_min = np.empty(10); stress_min.fill(stress[0][1]) # IN OTHER TESTS WE CHANGE HERE
    stress_min = np.empty(10); stress_min = stress[:,1]
    df_f_unif_stress_min = pd.DataFrame(stress_min)


    sigma_IM_10 = df_f.iloc[26:27,6:7]
    sigma_IM_10 = sigma_IM_10.to_numpy()
    sigma_IM_10_incs = np.empty(10); sigma_IM_10_incs.fill(sigma_IM_10[0][0]) # IN OTHER TESTS WE
CHANGE HERE
    sigma_IM_10 = pd.DataFrame(sigma_IM_10_incs)

    ###
    mat =[]
    for l in range(10): # line -> increment
```

```
        #print(l)
        row = []
        for c in range(9): # column -> feature -> sigma_IM*
            sigma_IM_star = (sigma_IM[c] + 2*Sigma_yeld)/(sigma_IM_10_incs[c] + 2*Sigma_yeld)
            row.append(sigma_IM_star)
        sigma_IM_star = (sigma_IM_10_incs[0] + 2*Sigma_yeld)/(2.2*Sigma_yeld)
        row.append(sigma_IM_star)
        mat.append(row)

    ###
    sigma_IM_star_matrix = pd.DataFrame(mat)

    label = df_info.iloc[f_cont:f_cont+1,0:1]
    label = label.to_numpy()
    label_incs = np.empty(10); label_incs.fill(label[0][0]) # IN OTHER TESTS WE CHANGE HERE
    label = pd.DataFrame(label_incs)
    #label = label.reset_index(drop=True)

    label_sigma_PD_incs = np.empty(10);
    for i in range(10):
        label_sigma_PD_incs[i] = (sigma_IM[i] + 2*Sigma_yeld)/(label_incs[i] + 2*Sigma_yeld)
    label_sigma_PD_incs = pd.DataFrame(label_sigma_PD_incs)

    f_id = df_info.iloc[f_cont:f_cont+1,2:3]
    f_id = f_id.to_numpy()
    f_id_incs = np.empty(10); f_id_incs.fill(f_id[0][0])
    f_id = pd.DataFrame(f_id_incs)

    ###
    # depth = i/10
    df_f = pd.concat([df_f_inc_depth, ft_ppts, sigma_IM_star_matrix, df_f_unif_sigma_IM, df_f_unif_stress_min,
sigma_IM_10, f_id, label_sigma_PD_incs], axis=1)  # sigma_IM := stress_max
    df_f.columns                                                                                                =
['Inc','i/10','stress_max_non_uni_reg','stress_min_non_uni_reg','stress_max_non_uni_non_reg','stress_min_non_
uni_non_reg','Sigma_yeld/E10-
2','sigma_IM_1*','sigma_IM_2*','sigma_IM_3*','sigma_IM_4*','sigma_IM_5*','sigma_IM_6*','sigma_IM_7*','si
gma_IM_8*','sigma_IM_9*',                                  '(sigma_IM_10+2*Sigma_yeld)/(2.2*Sigma_yeld)',
'sigma_IM','stress_min','sigma_IM_10','f','label_sigma_PD_incs']
    df_2 = df_f

    df_f = pd.concat([df_1,df_2], axis=0)

    df_1 = df_f

df_f = df_f.reset_index(drop=True)

#df_f.to_csv(r'C:\Users\twi\Desktop\TC_2\A04\pre_processing\dataset_uniform_stress_ds_approach.csv',index=
False)

# filters
df_f.drop(['stress_max_non_uni_reg','stress_min_non_uni_reg','stress_max_non_uni_non_reg','stress_min_non_
uni_non_reg','stress_min','f'], axis='columns',inplace=True)

# reordering
df_f                                                                                                        =
df_f[['sigma_IM_1*','sigma_IM_2*','sigma_IM_3*','sigma_IM_4*','sigma_IM_5*','sigma_IM_6*','sigma_IM_7
*','sigma_IM_8*','sigma_IM_9*',       '(sigma_IM_10+2*Sigma_yeld)/(2.2*Sigma_yeld)',       'Sigma_yeld/E10-
2','i/10','label_sigma_PD_incs']]
```

```
df_f.to_csv(r'C:\Users\twi\Desktop\TC_2\A04\pre_processing\dataset_uniform_stress_test_2_all_simulations.csv',index=False)
```

Appendix F – Scripts in Python for the implementation of the DNN model

The following script was coded in python language for the training, testing, validation and evaluation of the DNN model. The script was developed on Colab using Keras and TensorFlow. Since the script was coded on Colab, the usual format of the file is .ipynb, which is a Jupyter notebook format. The file can be easily downloaded from Colab as .py file and be executed.

It is important to point out that Colab allows the importing of files directly from a Google Drive link. This way the data tables (dataset) were saved in the author's personal Google Drive and imported through a link in the script.

ds_approach_analysis.py

```
# -*- coding: utf-8 -*-
"""ds_approach_analysis.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1S4efzLuCjqlZMiGQYk5Qpgy-5qw-5jWq
"""

# -*- coding: utf-8 -*-
"""
Created on Mon Jul 19 01:23:41 2021

Updated on Wed Apr 14 11:48:41 2022

Last Update on Mon Sep 12 15:40:24 2022

@author: Lucas Manassés
"""
# Libraries needed to run the code. Note that the script is divided into execution blocks.
# When running in Colab, the imported libraries do not require installation.
# If you are running on a local computer, check that all libraries imported here are installed
#  in your (Anaconda) environment.
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import math
from sklearn.preprocessing import MinMaxScaler # for ML purposes
from sklearn.preprocessing import StandardScaler,Normalizer
from sklearn.metrics import accuracy_score,confusion_matrix,r2_score
from sklearn.model_selection import train_test_split,cross_validate,ShuffleSplit
from sklearn.utils import shuffle
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import LogisticRegression
```

```
from sklearn.ensemble import
ExtraTreesClassifier,RandomForestClassifier,GradientBoostingClassifier,AdaBoostClassifier,RandomForestReg
ressor
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler  # doctest: +SKIP
from sklearn.model_selection import GridSearchCV
from sklearn import svm
from sklearn.svm import SVC

# Preprocess settings

from google.colab import drive
drive.mount('/content/drive')

# Test 1:
#dataset =
pd.read_csv(r'/content/drive/Shareddrives/TCC/TC_2/A04/pre_processing/dataset_uniform_stress_test_1_all_si
mulations.csv')
# Test 2:
dataset =
pd.read_csv(r'/content/drive/Shareddrives/TCC/TC_2/A04/pre_processing/dataset_uniform_stress_test_2_all_si
mulations.csv')

X = dataset.iloc[:, :-1] # last cleaning
y = dataset[['label_sigma_PD_incs']].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)

#%% Models

X = X_train
y= y_train

# Linear Regression
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X, y)

from sklearn.metrics import mean_squared_error
lin_label_predictions = lin_reg.predict(X)
lin_mse = mean_squared_error(y, lin_label_predictions)
lin_rmse = np.sqrt(lin_mse)


# Neural Networks

from keras.models import Sequential
from keras.layers import *

# Define the model
model = Sequential()
model.add(Dense(4, input_dim=12, activation='relu'))
model.add(Dense(4, activation='relu'))
model.add(Dense(1, activation='linear'))
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
# Train the model
model.fit(
    X,
    y,
    epochs=5000,
    shuffle=True,
    verbose=1
)

#%%

# Saving and loading Models (for later plots)
import joblib

# path
#filename_LINEAR = r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/Test_1_Linear_Model.sav"
filename_LINEAR = r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/Test_2_Linear_Model.sav"

#filename_NN = r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/Test_1_NN_Model.sav"
filename_NN = r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/Test_2_NN_Model.sav"

# save the LINEAR model to disk
joblib.dump(lin_reg, filename_LINEAR)

# save the NN model to disk
joblib.dump(model, filename_NN)


# load the models from disk

# Linear Model
lin_reg = joblib.load(filename_LINEAR)

# Neural Network Model
model = joblib.load(filename_NN)


#%% Evaluation

lin_label_predictions_testing_data = lin_reg.predict(X_test)

lin_label_predictions_training_data = lin_reg.predict(X_train)

#nn_prediction

nn_prediction = model.predict(X_test)

nn_prediction_train = model.predict(X_train)


# Saving the predictions
# TEST 1
#np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_1_lin_label_predictions_testing_data.csv", lin_label_predictions_testing_data, delimiter=",")
#np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_1_lin_label_predictions_training_data.csv", lin_label_predictions_training_data, delimiter=",")
#np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_1_nn_prediction_testing_data.csv", nn_prediction, delimiter=",")
```

```
#np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_1_nn_prediction_training_data.csv", nn_prediction_train, delimiter=",")

# TEST 2
np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_2_lin_label_predictions_testing_data.csv", lin_label_predictions_testing_data, delimiter=",")
np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_2_lin_label_predictions_training_data.csv", lin_label_predictions_training_data, delimiter=",")
np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_2_nn_prediction_testing_data.csv", nn_prediction, delimiter=",")
np.savetxt(r"/content/drive/Shareddrives/TCC/TC_2/A04/analysis/test_2_nn_prediction_training_data.csv", nn_prediction_train, delimiter=",")


#comparison TEST 1

y_target = y_test_t1.flatten()
y_hat_dann = nn_prediction_t1.flatten()
y_hat_mlr = lin_label_predictions_testing_data_t1.flatten()

comparison_t1 = abs(y_hat_dann - y_target) < abs(y_hat_mlr - y_target)

print(comparison_t1)
print("sum is: ",np.sum(comparison_t1)) # sum of trues...
len(comparison_t1)

#%%

#comparison TEST 2

y_target = y_test_t2.flatten()
y_hat_dann = nn_prediction_t2.flatten()
y_hat_mlr = lin_label_predictions_testing_data_t2.flatten()

comparison_t2 = abs(y_hat_dann - y_target) < abs(y_hat_mlr - y_target)

print(comparison_t2)
print("sum is: ",np.sum(comparison_t2)) # sum of trues...
len(comparison_t2)
```