

Treball de Fi de Màster

**Màster Universitari en Enginyeria Industrial**

**Machine Learning based prediction of the effect of lay-up  
defects in the Automated Fiber Placement**

**MEMÒRIA**

**Autor:** Marcal Iborra Escarré  
**Director:** Javier Anyeto Gubert  
**Ponent:** -  
**Convocatòria:** Gener 2023



Escola Tècnica Superior  
d'Enginyeria Industrial de Barcelona





## Acknowledgement

I would like to extend my sincere gratitude to the Technical University of Munich and the Department of Composite Materials for their unwavering support in the completion of my engineering project. Their invaluable guidance, resources, and expertise helped me to overcome numerous challenges and achieve a successful outcome. I am truly grateful for the opportunities they provided me with and I am confident that the knowledge and skills I have acquired will serve me well in my future endeavors. I am honored to have been associated with such a prestigious institution, thank you for your ongoing support. I am especially grateful to Fabian Diemar, who supervised this project and pushed me to work with scientific rigor and presenting the results in a clear and precise way. He gave me all the advises and guidance that I needed, which helped me during the project to obtain the final project results

I would like also to thank my family and my friends for all the support received during the project development. Specially to Roger Aylagas, Christian Fernandez and Enric Gil for their helpful suggestions in the many hours of discussion in the field of artificial intelligence and machine learning, which have helped me to reach a deep understanding on this field..

## Resum

L'ús del *Automated Fiber Placement* està estenent-se en la indústria aeroespacial. La necessitat de fabricar components estructurals compostos grans i complexes, fa que l'ús d'aquesta tecnologia sigui molt més eficient que la fabricació convencional amb col·locació manual. No obstant això, aquests components encara s'estan inspeccionant manualment i es calcula l'efecte dels defectes trobats amb software de simulació.

L'abast d'aquesta tesi és crear un model de *Machine Learning* que sigui capaç de calcular l'efecte en la rigidesa efectiva per diferents configuracions de defectes. Aquest model d'aprenentatge automàtic hauria de rebre les característiques geomètriques dels defectes en el laminat i de ser capaç de predir, amb un alt nivell de precisió, la rigidesa efectiva del laminat. Entrenar aquest model amb una gran quantitat de configuracions de defectes diferents genera la necessitat de crear un model FE parametritzat d'una laminació composta en el nivell de cupó.

Els resultats mostren que una arquitectura de *Multilayer Perceptron* amb dues *hidden layers*. La primera amb 281 nodes i la segona amb 76 nodes, és capaç de predir la rigidesa efectiva d'un laminat defectuós amb una precisió de 0,1 GPa.

## Resumen

El uso del *Automated Fiber Placement* se está expandiendo en la industria aeroespacial. La necesidad de fabricar grandes y complejos componentes estructurales de materiales compuestos, hace que el uso de esta tecnología sea mucho más eficiente que la fabricación manual convencional. Sin embargo, estos componentes siguen siendo inspeccionados manualmente y se calcula el efecto de los defectos encontrados con un software de simulación.

El objetivo de esta tesis es crear un modelo de *Machine Learning* que sea capaz de calcular el efecto sobre la rigidez efectiva para diferentes configuraciones de defectos. A este modelo de aprendizaje automático se le deben proporcionar las características geométricas del defecto en el laminado y tiene que ser capaz de predecir, con un alto nivel de precisión, la rigidez efectiva del laminado. El entrenamiento de este modelo se debe de realizar con una gran cantidad de configuraciones de defectos diferentes. Este hecho genera la necesidad de crear un modelo de elementos finitos parametrizado de un laminado a nivel de cupón.

Los resultados muestran que una arquitectura *Multilayer Perceptron* con dos *hidden layers*. La primera con 281 nodos y la segunda con 76 nodos que es capaz de predecir la rigidez efectiva de un coupon laminado defectuoso con una precisión de 0,1 GPa.

## Abstract

The use of Automated Fiber Placement is being widespread in the aerospace industry. The need of manufacturing large and complex structural composite components, it makes the use of this technology much more efficient than the conventional hand lay-up manufacturing. However, these components are still being manually inspected and the effect of the defects found is calculated with a simulation software.

The scope of this thesis is to create a Machine Learning model that is able to calculate the effect on the effective stiffness for different defect configuration. This Machine Learning model should be provided with the geometrical defect characteristics in the laminate and it has to be able to predict, with a high level of accuracy, the effective stiffness of the laminate. Training this model with a big amount of different configuration defects generates the need to create a parametrized FE model of a composite laminate on the coupon level.

The results show that a Multi Layer Perceptron architecture with two hidden layers. The first one with 281 nodes and the second one with 76 nodes which is able to predict the effective stiffness of a defective laminate coupon with an accuracy of 0,1 GPa.

# Contents list

<b>Acknowledgement .....</b>	<b>III</b>
<b>Resum .....</b>	<b>IV</b>
<b>Resumen .....</b>	<b>V</b>
<b>Abstract .....</b>	<b>VI</b>
<b>Contents list .....</b>	<b>VII</b>
<b>Nomenclature .....</b>	<b>X</b>
<b>Abbreviations .....</b>	<b>XI</b>
<b>1 Introduction.....</b>	<b>1</b>
<b>2 Fundamentals .....</b>	<b>3</b>
2.1 Composite materials.....	3
2.2 Automatic fiber placement (AFP).....	5
2.3 Finite Element Method (FEM).....	7
2.4 Machine learning .....	9
2.4.1 Introduction .....	9
2.4.2 Methods .....	10
2.4.3 Regression models.....	11
2.4.4 Data pre-processing .....	14
2.4.5 Training .....	15
2.4.6 Validation .....	17
<b>3 State of art.....</b>	<b>20</b>
3.1 FEM models for AFP defects .....	20
3.2 ML application for FEM replacement .....	23
<b>4 Data generation.....</b>	<b>27</b>
4.1 FEM model .....	27
4.1.1 Geometry and material properties .....	27
4.1.2 Defect properties .....	29
4.1.3 Part and assembly generation .....	30

4.1.4 Boundary conditions and loading .....	33
4.1.5 Mesh .....	33
4.1.6 Results.....	41
4.2 FEM Parametrization .....	42
<b>5 Implementation of Machine Learning Models.....</b>	<b>49</b>
5.1 Data pre-processing .....	49
5.2 Model definition .....	50
5.2.1 Multilinear model.....	50
5.2.2 Polynomial model .....	51
5.2.3 MLP with 1 hidden layer .....	52
5.2.4 MLP with 2 hidden layers.....	53
5.3 Model training .....	54
5.4 Hyperparameter tuning.....	57
<b>6 Results .....</b>	<b>61</b>
6.1 Multilinear regression.....	61
6.2 Polynomial regression .....	63
6.3 MLP with one hidden layer .....	65
6.4 MLP with two hidden layers .....	68
<b>7 Economic study .....</b>	<b>71</b>
<b>8 Environmental study.....</b>	<b>72</b>
<b>9 Social and gender equality study .....</b>	<b>72</b>
<b>10 Conclusions and outlook .....</b>	<b>73</b>
10.1 Conclusions .....	73
10.2 Outlook.....	75
<b>11 References.....</b>	<b>77</b>
<b>List of Figures.....</b>	<b>81</b>
<b>List of tables .....</b>	<b>83</b>
<b>Appendix .....</b>	<b>84</b>
A USB-Data .....	84





## Nomenclature

Formelzeichen	Units	Description
$A$	$[mm^2]$	Cross section
$[A]$	$[N/mm]$	Extensional stiffness matrix
$b$	$[-]$	Bias term in the ANN
$E_{11}$	$[MPa]$	Stiffness in longitudinal direction
$E_{22}$	$[MPa]$	Stiffness in transversal direction
$G_{12}$	$[MPa]$	Shear modulus
$h_{middle-bottom}$	$[mm]$	Distance between the middle and bottom shells
$h_{top-middle}$	$[mm]$	Distance between the top and middle shells
$l$	$[mm]$	Length of the laminate
$[Q]$	$[MPa]$	Reduced stiffness matrix
$RF_1$	$[N]$	Reaction force in longitudinal direction
$t$	$[mm]$	Ply thickness
$t_{defect}$	$[mm]$	Thickness of the defective ply
$\nu_{12}$	$[-]$	Poisson's ratio
$w_i$	$[-]$	Weights of the ANN
$\sigma_{eff,11}$	$[MPa]$	Effective stress in longitudinal direction

## Abbreviations

Abbreviation	Description
AFP	Automatic fiber placement
AI	Artificial Intelligence
ANN	Artificial neural network
CFRP	Carbon fibre reinforced plastic
CLT	Clasic laminate theorie
FEA	Finite element analysis
FEM	Finite element method
ML	Machine Learning
MLP	Multi Layer Perceptron
ReLU	Rectified Linear Unit
RMSE	Root mean squared error
RVE	Representative volume element



# 1 Introduction and motivation

Nowadays the use of carbon fiber reinforced plastic (CFRP) in the manufacturing of structural components has become more prominent. An example of this extended use can be found in the structure of commercial airplanes which consists at least of 50% CFRP [2]. Therefore, traditional materials like wood, aluminum or steel are losing their use in favor of composite materials that have better specific strength and modulus. In this context, the Automated Fiber Placement (AFP) method is developed in order to manufacture carbon fiber structural components. The use of this method is widespread in the aerospace industry, where large and complex components make hand lay-up manufacturing very time consuming or impractical [3]. The process consists in computer-controlled placement of pre-impregnated fiber tapes (prepreg) on the surface of a mold. During the placement process some defects may occur, such as an empty space (gaps) or superposition (overlap) between consecutive prepreg tapes. These defects led to fiber waviness in the composite layup and consequently to a reduction of its mechanical properties and load carrying capacity which can be critical in the component stability and compromise the safety requirements [4].

Since mechanical testing of the defective lay-ups, taking into account all the possible defect configurations is impossible, Finite Element (FE) models can be created in order to decrease this effort on mechanical testing. The problem is that the preparation and computation of the model is still very time consuming. In this context, the present project objective is to create a machine learning model that is using all the data from Finite Element Analysis in order to approximate the effect of a certain defect on the laminate mechanical properties. This can facilitate the evaluation of the defects effect in the laminate without using an FEA model, which it will save a lot of computational time.

The machine learning model has to be trained with data that will be generated via ABAQUS with Finite Element Analysis (FEA). For this purpose, a FE model consisting in a tensile of a composite laminate coupon is created in the laminate, a defect is

introduced in one of the plies and the longitudinal stiffness is calculated. In order to do that it is required a parametrized Finite Element Model, this model has to be simple enough to avoid excessive long computational times to generate each one of the samples, but accurate enough to capture the effects of the defective plies in the laminate. Instead of using solid 3D elements like Daniel Del Rossi in [4], a shell model will be created in order to simplify the geometry and reduce the number of nodes, only one defect in the laminate will be considered. These simplifications will reduce drastically the computational time, which will make the data generation more efficient. The model has to be verified comparing the mechanical properties of the Finite Element Model with the real values obtained via analytical calculations or experimental results. Once the model is verified, the parametrized model will be added in a loop where the input parameters will be changed, and the output calculated at each iteration.

Before generating the data, the number of generated samples has to be chosen. Since the amount of data needed to train the model depends on the model used, an initial guess of samples will be set. Then using the learning curves (accuracy vs number of samples) the need to generate more data will be evaluated. After the data is generated, the data will be split into two sets: one for training and the other for test. A pre-processing of the data will be needed in order to obtain an enhanced performance of the machine learning algorithms. With the processed data, some machine learning algorithms will be trained, and its performance optimized by selecting the adequate value of the parameters for each of them. Feature selection may be also required to avoid overfitting. Once all the models are trained its performance will be tested using the test dataset. The model with the best performance has to be chosen. It is also important to understand and evaluate the model in order to predict the knock-down-factor of a laminate with a certain defect.

## 2 Fundamentals

### 2.1 Composite materials

Composite materials are described as a material that is made with the combination of two or more materials, which are physically and chemically differentiable and mechanically separable. The resultant material has better mechanical properties than the simple sum of the properties of its components. In almost every cases the composite materials are made of fibers, which confer the strength to the laminate and resin, as well as holding all the fibers together. Since the fibers are just able to carry loads in one direction, the load carrying capacity of a ply depends on the fiber orientation. A composite part consists of a stacking of layers (plies) with different fiber orientations which are consolidated together in one piece during manufacturing [4].

In order to validate the results obtained with the finite element simulation the classic laminate theory (CLT) will be used as an analytical approach. CLT is a method used to calculate the stress and strain distribution in the laminate depending on its material properties and stacking sequence [5]. This theory is used prior to more advanced mathematical models to get an approximation of the strength, stiffness, and thickness of the laminate [5]. To calculate the effective stiffness of the laminate, the reduced stiffness matrix  $Q_{ij}$  for each material used in the laminate need has to be calculated following the equations (2-2) and (2-3). With this matrix, the elastic behaviour of the ply in plane loading is described.

$$Q_{ij} = \begin{bmatrix} Q_{11} & Q_{12} & 0 \\ Q_{21} & Q_{22} & 0 \\ 0 & 0 & Q_{66} \end{bmatrix} \quad (2-1)$$

$$Q_{11} = \frac{E_{11}^2}{(E_{11} - \nu_{12} \cdot E_{22})} ; Q_{12} = \frac{\nu_{12} \cdot E_{11} \cdot E_{22}}{(E_{11} - \nu_{12}^2 \cdot E_{22})} \quad (2-2)$$

$$Q_{12} = \frac{E_{11} \cdot E_{22}}{(E_{11} - \nu_{12}^2 \cdot E_{22})} ; Q_{66} = G_{12} \quad (2-3)$$

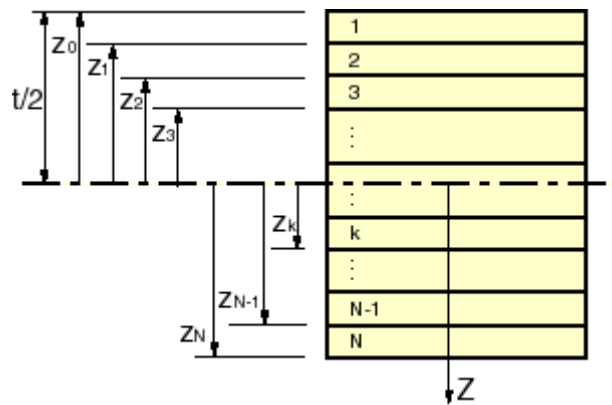
Then the transformed reduced stiffness matrix has to be calculated, taking into account the fiber orientation, as it can be seen in equation (2-3).

$$[Q]_{fo} = [T]^{-1} \cdot [Q] ; \text{where } [T] = \begin{bmatrix} c^2 & s^2 & 2sc \\ s^2 & c^2 & -2sc \\ -sc & sc & c^2 - s^2 \end{bmatrix} \quad (2-4)$$

The terms  $c$  and  $s$  refer to the sine and cosine of the fiber orientation angle. After calculating the transformed reduced stiffness matrix, the  $A$  matrix can be calculated using equation (2-5).

$$A_{ij} = \sum_{k=1}^n [Q]_{fo,k} \cdot (z_k - z_{k-1}) \quad (2-5)$$

The  $z_k$  terms are the distance of the upper and lower ply surface to the half plane in the laminate thickness, as it is shown in Fig 2-1.



**Fig 2-1 Schematic representation of the half plane distances [6]**

Once the  $A$  matrix is calculated, following the guidelines of [6] the effective stiffness on the longitudinal direction of the laminate can be calculated using equation (2-6).



$$E_{11, \text{ eff}} = \frac{1}{h} \left( A_{11} - \frac{A_{12}^2}{A_{22}} \right) \quad (2-6)$$

Where  $h$  is the laminate thickness.

## 2.2 Automatic fiber placement (AFP)

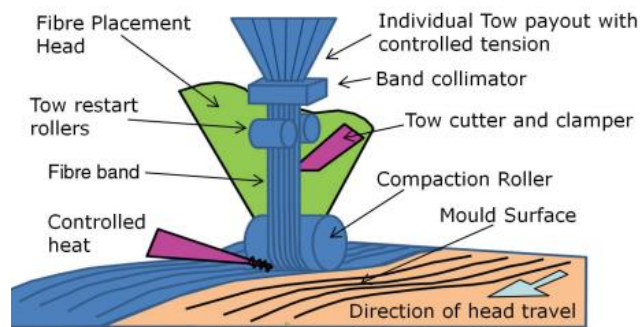
In its beginnings, the manufacture of carbon fiber elements was done using partially cured epoxy resin prepreg material sheets that were placed on a large table. The prepreg sheet pieces were then cut out and placed on top of each other in a mold to make a laminate. The current rates of demand have generated that carbon fiber part manufacturing has had to move away from its more artisanal side to start finding efficient and reliable ways of automation. AFP technology is a process that uses computer-guided robotics to place one or more layers of resin-impregnated continuous fiber tape over a mold to create a part or structure. This technology reduces the cycle time cost for large parts, the manufacturing of flat or curved parts and one-off structures can be easily done and as a result it increases accuracy, repeatability and quality.

There are several manufacturing techniques for composite parts, all of them start with either dry fibers or with pre-impregnated fibers. With dry fibers the resin is introduced after all the plies are placed on the mold, while pre-impregnated fibers are already coated with resin and are placed ply by ply on the mold. After all plies are placed a curing process consolidates all the layers in one solid. If the placing of the fibers is done by hand, then it's called hand layup. This can also be done with a machine, which is the case of automated fiber placement (AFP).

The AFP machine is divided in three parts:

- Motion components, which allow the motion of the machine
- Machine head, which handles the layup process

- Tool support, which is the support of the part mold



**Fig 2-2: Automated fiber placement head. [1]**

The AFP process is essentially based on a mechanical-chemical bonding reaction. It consists in the automated placing of narrow stripes of prepreg material (tows) in such a way that a surface is covered by multiple layers of CFRP material. First of all, the tape feed unit, which is composed of directional rollers, delivers the tows at the top of the fiber placement head with tension. Since the placement head Fig 2-2 can just place a limited number of tows, the layer has to be laid up step by step. Then the cutting unit ensures the cut of the tows at the right time to obtain the needed layup. Before the tow is placed on the mold surface or on the previous ply the heating source ensures that the temperature of the tow surface is high enough to guarantee a correct adherence. After that, the compaction roller presses the tow onto the surface of the previous ply, this process ensures the elimination of the trapped air and inner gaps and the quick adherence of the tow [2]. The correct placement of the tow depends on the process parameters speed, temperature and roller pressure. If these parameters are not well adjusted during the placement process of the tows multiple defects may occur. This is the case of gaps and overlaps, which take place between two consecutive tows or when multiple tows are placed at a time between courses. On the one side, if the tows are superposed which is the case of an overlap and on the other side, if an empty space between the tows is left which is the case of a gap, as it can be seen in Fig 2-3. These defects can weaken the structure and can compromise the component stability. To quantify the effect on the mechanical strength of the defects mechanical tests can be done [4].

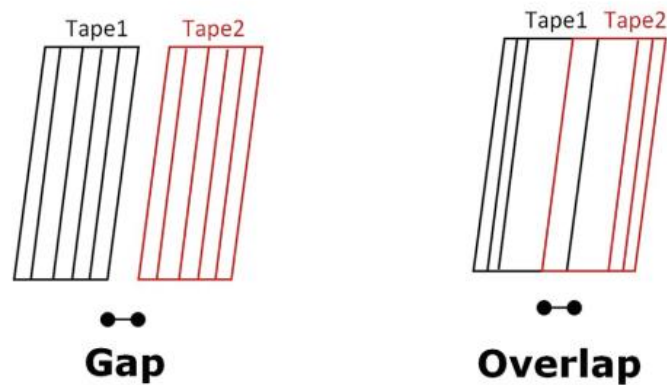


Fig 2-3: Schematic representation of gaps and overlaps. [1]

## 2.3 Finite Element Method (FEM)

When classifying structures, they are usually divided into discrete or reticular structures and continuous structures. The first ones are those which are made up of an assembly of elements clearly differentiated from each other and joined at specific points. The fundamental characteristic of discrete structures is that their deformation can be defined exactly by means of a finite number of parameters, such as the deformations of the junction points of some elements and others. In this way the equilibrium of the whole structure can be represented by the equations of equilibrium in the directions of these deformations [7].

On the other hand, in continuous systems it is not possible to separate, a priori, the system into a finite number of discrete structural elements. If any part of the system is taken, the number of junction points between that part and the rest of the structure is infinite, and it is therefore impossible to use the same method as in discrete systems [7].

A continuous medium has infinitely many possible ways of deforming, independent of each other, since each point can be displaced while keeping any finite number of the remaining points fixed. That's the reason why it is only possible to obtain analytical solutions for systems with very simple geometry and/or simple boundary conditions.

To solve this problem, the Finite Element Method is used, which is based on [8]:

- Transforming a body of continuous nature in an approximate discrete model, this transformation is called element discretization of the model.

- Approximating the variables of the problem in each element by combinations of functions defined in the nodes (junction points between adjacent elements), thus reducing the variables to a determined and finite number of points. This is known as mathematical discretization.

To perform this operations FE software can be used, which have the following framework [7]:

- Pre-processing: Includes geometry definition, material selection, mesh generation and application of loads and model boundary conditions.
- Calculator: This is the part of the program that performs the entire FEM calculation and generates the solutions.
- Post-processing: provides a visual representation of the results and allows a simple interpretation of the results.

In creating a FEM model, one must strive for accuracy and computational efficiency. In most cases, the use of a complex model is likely to generate greater computational accuracy at the expense of an unnecessary increase in processing time. For this reason, the understanding of the system behaviour it's crucial during the pre-processing, since the model is discretized into a finite amount of elements. The element shape, dimensions and the quantity will define the accuracy of the FEA.

The elements used in the model discretization depend on its geometrical complexity. This elements can be classified in three groups [9], see in Fig 2-4:

- Linear (1-D) elements: These elements are characterized by having one dimension much larger than the other two. The length is defined with the junction of 2 nodes with a line, while the other dimensions are defined by assigning a cross section to the line. Linear elements may be subjected to transverse loads and/or bending moments in addition to tension and compression. Since there is just one element across the cross section there is a significant reduction of meshing effort and computational time compared with 2-D and 3-D elements [9], [10].
- Shell (2-D) elements: These elements are characterized by having two dimensions much larger than the third one. The first two dimensions are defined with a surface (which can be triangular or quadrilateral), while the third dimension is defined by assigning the thickness to it. Shell elements may be suitable for models subjected

to plane stress and plane deformation. It has advantages like 1-D elements in terms of less modeling efforts and faster simulation compared to 3D Elements, but it has some limitations when the model has irregular surfaces with different features on two sides [10].

- Solid (3-D) elements: All the dimensions are comparable. They can be used to model three-dimensional structures. They can provide information about the three-dimensional variation of the stresses and strains of the element [10].

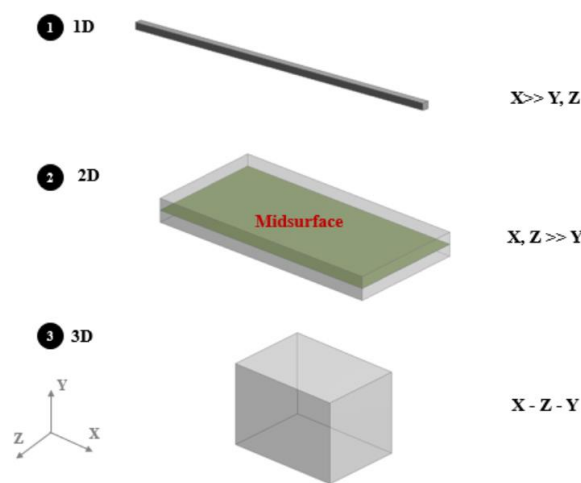


Fig 2-4: 1-D, 2-D and 3-D element representation [11]

## 2.4 Machine learning

### 2.4.1 Introduction

Machine Learning is a field of Artificial Intelligence (AI) that enables the computer to learn and improve its performance without being explicitly programmed. The term Machine Learning (ML) refers to the automated detection of meaningful patterns in data. In the last few years, the use of ML to extract information from large data sets it has become one of the most popular tools. Nowadays, machine learning based technology can be found everywhere: search engines learn how to bring us the best results, e-commerce and entertainment companies to recommend the most suitable products, anti-spam software learns to filter our email messages. Face detection in digital cameras and speech recognition which allows intelligent personal assistance applications on smart-

phones to follow voice instructions. Cars are equipped with accident prevention systems that are built using machine learning algorithms. Machine learning is also widely used in scientific applications such as disease diagnosis in medical science and telescope image clearing in astronomy [12].

## 2.4.2 Methods

Machine Learning methods are divided in two categories depending on whether the output values are required to be present in the training data or not:

Unsupervised learning techniques require only the input feature values in the training data and the learning algorithm discovers hidden patterns or data groupings in the training data based on its implicitness. Unsupervised learning models are used for three main tasks: clustering, association and dimensionality reduction. Clustering techniques try to partition the data into coherent groups based on the similarities or differences, it can be used for categorization of articles according to their topics. Association is a rule-based method that finds relationships between variables in the dataset, these are frequently used for market basket analysis. Dimensionality reduction is a technique used when the number of features, or dimensions, of the dataset is too high. It reduces the number of data inputs to a manageable size trying to preserve the integrity of the dataset as much as possible [13].

Supervised learning methods require knowing the value of the output variable for each training sample. That's why, each training sample has to be represented in the form of a pair of input and output values. The algorithm then trains a model adjusting its parameters and predicts the value of the output variables from the input variables. If the output variables have a continuous value, then the predictive model is called a regression function. For instance, predicting the price of a house given house features is a regression problem. If the output variable is a discrete then the predictive model is called a classifier. A typical classification problem is automated medical diagnosis for which a patient's data needs to be classified as having a certain disease or not [13].

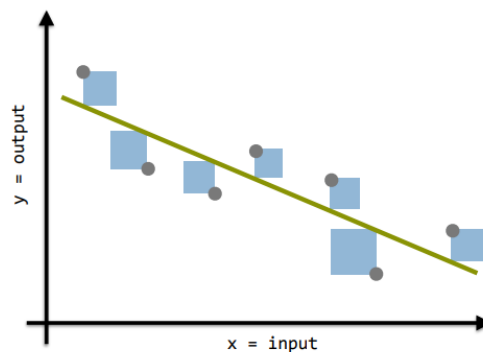
### 2.4.3 Regression models

The following are some of the most commonly used regression ML models:

Multilinear Least squares regression: is a linear regression model with one dependent variable and more than one independent variables [14]. With the form:

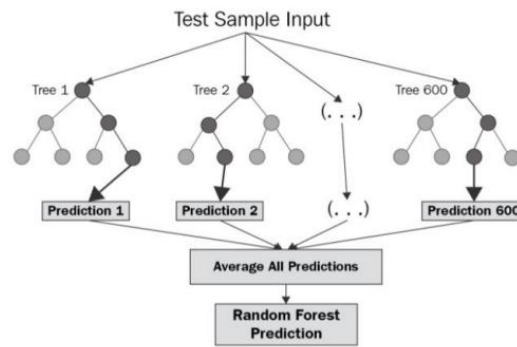
$$y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + \cdots + \beta_n \cdot x_n \quad (2-7)$$

Where  $y$  is the dependent variable,  $\beta_0, \beta_1, \dots, \beta_n$  are the regression coefficients and  $x_1, x_2, \dots, x_n$  are the independent variables. The algorithm fits the linear model with coefficients  $\beta = \beta_0, \beta_1, \dots, \beta_n$  to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation [15].



**Fig 2-5: Graphical representation of Least squares method. [16]**

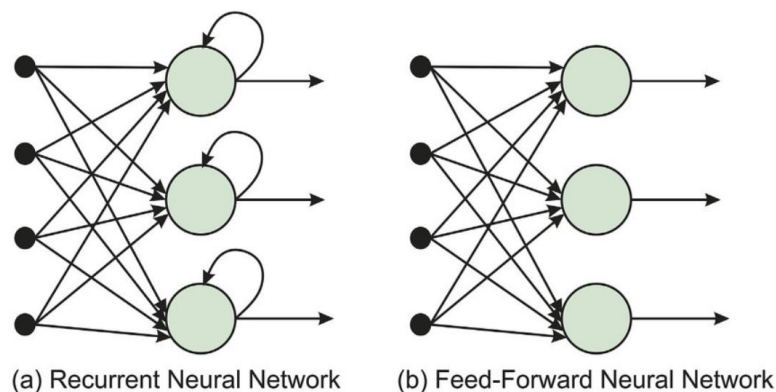
Random Forest Regression: This model is a supervised learning algorithm that uses the ensemble learning method for regression, see architecture in Fig 2-6. As a short description, ensemble learning is a technique that averages the predictions of multiple machine learning algorithms to make a more accurate prediction than a single model. Thus, this approach works by creating and training several decision trees, predicting the output values of each decision tree in the model. The final output consists of the average of the different predicted values of all the decision trees. These decision trees are developed in parallel, avoiding any influence between them and using different parts of the training data set for each tree. The number of trees generated and its depth have to be well chosen in order to obtain good results [17].



**Fig 2-6 Random forest regression architecture [17]**

Artificial Neural Networks (ANNs): are computational models that consist of several processing elements that receive inputs and deliver outputs based on their predefined activation functions [18].

There are two main categories of network architecture depending on the type of the connections between the neurons. If there is no feedback from the outputs of the neurons towards the inputs throughout the network, the information is propagated in one direction. Then it is referred to a Feed Forward Neural Network (FFNN). Otherwise, when this feedback exists, the network is called a Recurrent Neural Network (RNN) [19].



**Fig 2-7: Feed-forward and Recurrent neural networks representation. Source [20]**

One of the architectures of FFNN is the Multilayer Perceptron (MLP). MLP uses the work principle of the biological neurons, which are activated by electrical impulses from linked neurons and can transmit this activation state to further neurons [21]. The concept of an artificial neuron, at least consists on:

- One or more inputs

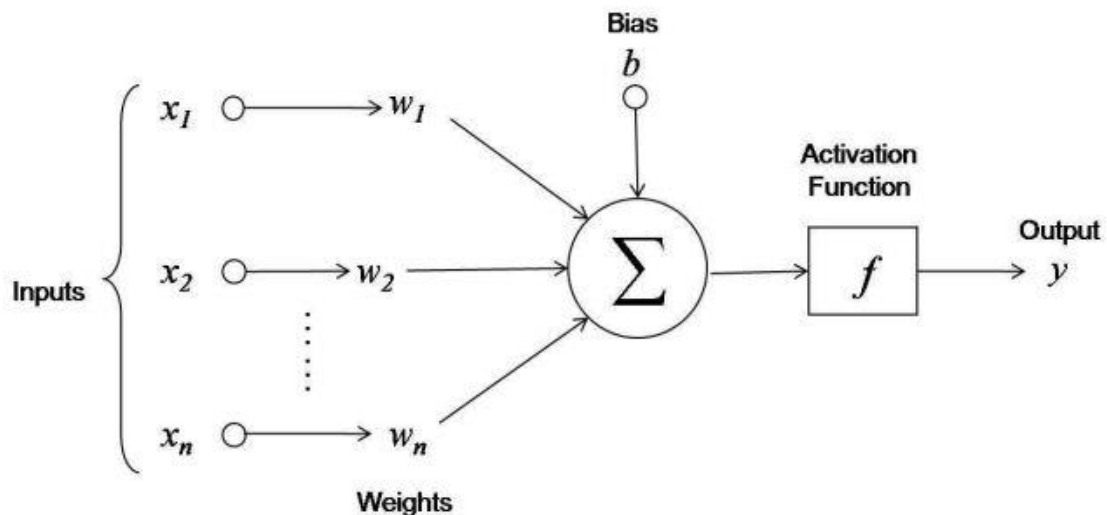


- The neuron itself
- An output

Each entry is multiplied by a learnable weight parameter and after adding the bias parameter, a non-linear operation is performed by applying an activation function see in Fig 2-8. This leads to the following equation for the output  $y$ .

$$y = \sigma \left( \sum_{i=1}^n x_i w_i + b \right) \quad (2-8)$$

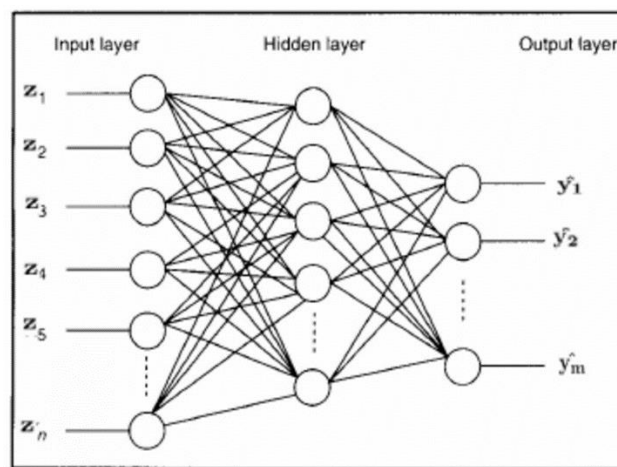
With  $\sigma$  as the activation function,  $w_i$  the weight parameters,  $x_i$  as the input data, and  $b$  as the bias. The activation function is chosen considering the type of the output, some examples can be Hyperbolic Tangent function, Rectified linear unit (ReLU) or Sigmoid function [16]. By choosing an activation function, its hyperparameter would have to be tuned in order to obtain the best possible accuracy on the predictions. The weights and biases, on the other hand, are optimized during the training process [22].



**Fig 2-8: Neuron representation [23]**

As an example of FFNN, Multilayer Perceptron (MLP) is an architecture, which consists in set of neurons organized in layers and connected to each other see in Fig 2-9. Three types of layers can be distinguished in MLPs:

- Input layer: Takes the data from the network. The number of nodes equals the number of inputs
- Hidden layer/s: Receive the inputs or the outputs from previous hidden layers, and processes them. It is where the learning process will take place. The number of nodes of the hidden layer has to be adjusted in order to obtain the best predictions.
- Output layer: will also process the information from the hidden layer and give the output. It can have one more nodes depending on how many parameters the model is predicting.



**Fig 2-9: Artificial neural network schematic representation**

On the one hand ANNs are particularly characterized by high accuracy with short inference time, also ANNs are insensitive to noise and have a good generalization. On the other hand, the long training time, a learning process that is difficult to interpret and the high number of hyperparameters, such as number of hidden layers and number of neurons in each one, are disadvantages [22].

## 2.4.4 Data pre-processing

Machine learning has great potential to develop data-driven systems. Nevertheless, a great model doesn't ensure good performance when data quality is not good enough. Polyzotis in [24] defined the main challenges of data management: Data understanding, data validation, data cleaning and data enrichment

- Data understanding: This process involves generating and visualizing salient features about the data, identifying any anomalies or outliers that exist in the data and recognizing implicit and explicit data dependencies. It is also important to encode the data into features that can be processed by the model.
- Data validation: It has a crucial influence on the quality of the generated model. The notion of validity has several facets, including: ensuring that training data have the expected features, these features have the expected values, features are correlated as expected and serving data does not deviate from training data.
- Data cleaning: Once a validation error is detected, the next logical step is to clean the data. This task can be split into three sequential subtasks: understanding where the error occurred, understanding the impact of the errors and fixing them.
- Data Enrichment: it refers to the augmentation of the training data with new features in order to improve the quality of the generated model. A common form of enrichment is to join in a new data source in order to augment the existing features with new signals. Another form is using the same signals with different transformations, e.g., using a new embedding for text data.

## 2.4.5 Training

Once the data is preprocessed, the model has to be trained, there is a vast quantity of machine learning algorithms that can do the same task. The challenge is to find the model that adjusts the training data the best. To quantify this measure of the best the so-called loss function can be used, which it can be minimized with some optimization method. The loss function measures the accuracy of a model based on the training dataset, so the best model that it can be obtained is the model with the minimum loss. In regression problems, the most used loss functions are:

- Mean squared loss (MSE or L2): is the mean of the squared difference between the predicted and the real value. Since the loss is squared, this loss function penalizes the model for large differences between the predicted and the real value. This means that the MSE error function is not robust against outliers, but its solution can be obtained analytically.

$$MSE = \frac{1}{N} \cdot \sum_{i=1}^N (y_{i \text{ pred}} - y_{i \text{ true}})^2 \quad (2-9)$$

- Mean absolute loss (MAE or L1): is the mean of the absolute value of difference between the predicted and the real value. This loss function is more robust against outliers than MSE, but its analytical solution can not be obtained and it is not differentiable at the origin.

$$MAE = \frac{1}{N} \cdot \sum_{i=1}^N |y_{i \text{ pred}} - y_{i \text{ true}}| \quad (2-10)$$

- Huber Loss: Combines the strengths and weaknesses of L1 and L2 loss functions. It is defined as a combination of L1, for loss values larger than a predefined hyperparameter  $\delta$ , and L2 loss function for loss values lower than  $\delta$ . The resultant loss function is robust against the outliers and differentiable at the origin, but has more hyperparameters and its solution can not be analytically obtained.

$$L = \frac{1}{N} \cdot \sum_{i=1}^N \begin{cases} 0.5 \cdot (y_{i \text{ pred}} - y_{i \text{ true}})^2, & \text{for } |y_{i \text{ pred}} - y_{i \text{ true}}| \leq \delta \\ \delta \cdot |y_{i \text{ pred}} - y_{i \text{ true}}| - 0.5 \cdot \delta^2, & \text{otherwise} \end{cases} \quad (2-11)$$

The training of machine learning model refers to the optimization of the parameters, which have to lead to a minimization of the error function [16]. So the goal of the training is to find the set of parameters  $\theta_* = [w_1, w_2, \dots, w_n]$  for which the loss function is minimal. In order to achieve this goal an iterative method is used, that given an initial guess of the parameters generates a sequence such as:

$$\theta_{i+1} = \theta_i + \alpha \cdot p_i \quad (2-12)$$

The vector  $p_i$  is the update direction, while the scalar parameter  $\alpha$  is the learning rate.

One of the most used optimization method is the Gradient Descent. This method assumes that the update direction that minimizes the most the error is equal to the negative gradient direction [16]. The update rule has the following form:

$$\theta_{i+1} = \theta_i - \alpha \cdot \nabla E(\theta_i) \quad (2-13)$$

The learning rate value  $\alpha$  has to be set small enough to not have a slow convergence and not too large to avoid oscillations on the parameters.

This method is especially efficient for convex error functions, but in the case of neural networks, the error is generally far from being convex. In this context some optimization methods are developed to overcome the problems of the gradient descent method and make optimization more efficient:

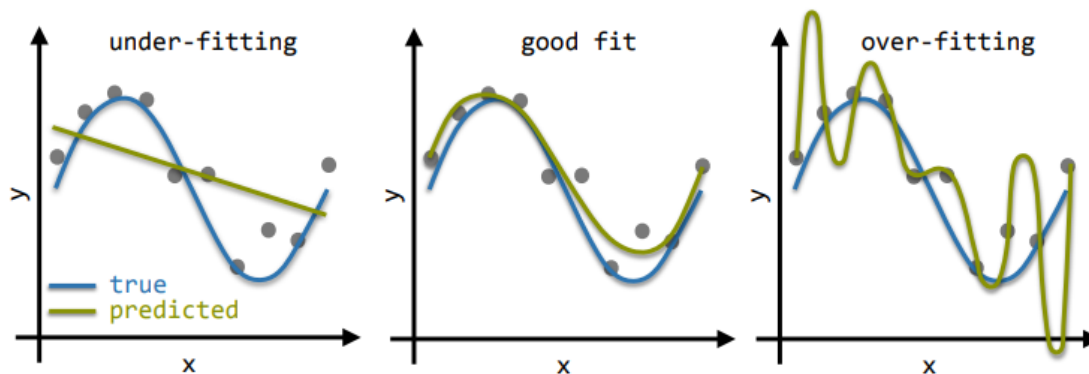
- Gradient descent with momentum: Uses the past gradients as well as the current ones to define the update direction as the sum of exponentially weighted past gradients, giving more relevance to more recent gradients [16].
- RMSProp: extends the gradient descent with momentum method by adding a scaling factor of the learning rate, which makes that each component have a different effective learning rate [16].
- Adam: Combines the heuristics of RMSProp and gradient descent with momentum method [16].

To compute the gradients of the error function and update the parameters of the NN the backpropagation method is used. The backpropagation algorithm works by computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient one layer at a time, iterating backward from the last layer to avoid redundant calculations of intermediate terms in the chain rule [25].

## 2.4.6 Validation

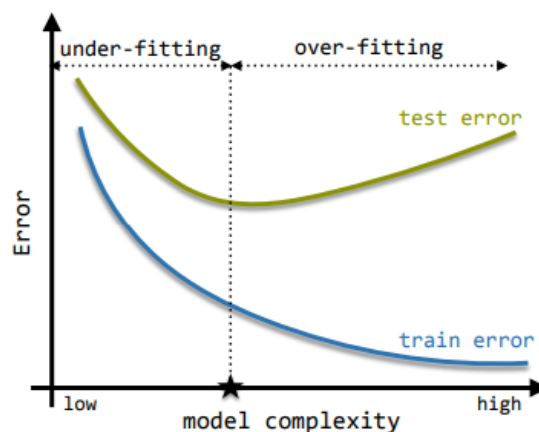
In order to choose the most suitable model, it is mandatory that the model after the training stage is able to give good predictions for unseen inputs, then it can be said that the model generalizes well. In the figure Fig 2-10 it can be seen a regression machine learning model with three different polynomial function. After the training the left model is not flexible enough to capture the relevant relations between the input and the output, which is the so called underfitting. On the right side there is a case of overfitting, where the algorithm is

trying to model the random noise, irrelevant features become too important and there is a failure of generalization between data points.



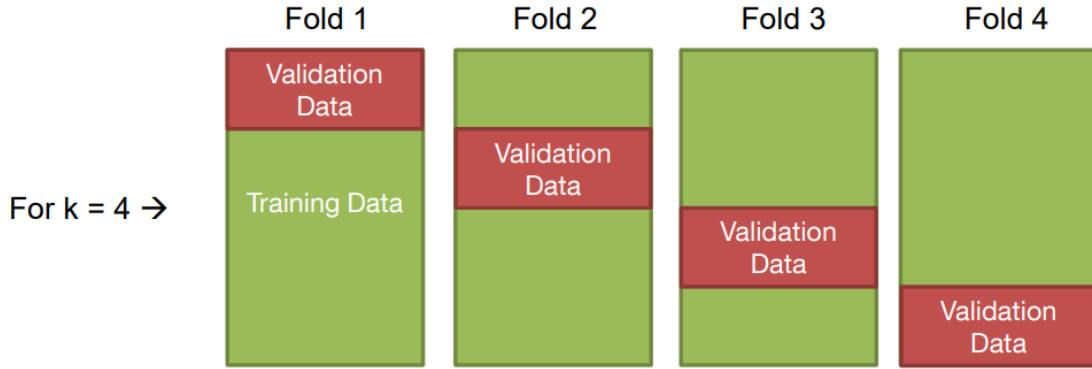
**Fig 2-10: Examples of over- and underfitting. [16]**

Since it is difficult to judge overfitting in high dimensional spaces, a standard method is to separate the data into training and test data. This will allow to calculate the test data loss which can help to judge overfitting, as it can be seen in the figure below.



**Fig 2-11: Variation of the test and training error depending on the model complexity. [16]**

In case of having a small dataset, it can be possible that by removing a part of the training data, the model can not be trained sufficiently well. In this situation k-Fold Cross-Validation can be used, this technique consists in splitting the data in  $k$  subsets “folds” of roughly equal size. Use  $k-1$  subsets as training data and compute the expected loss with the remaining one as test data. By repeating the process  $k$  times an average error can be calculated and the variance of the estimation error is an indicator for model stability.



**Fig 2-12: K-fold cross validation schematic [16]**

Other metrics to evaluate the accuracy of the model can be the MAE explained in page 15, the Root mean squared error (RMSE), and  $R^2$  score.

- RMSE: It shows how far predictions are from measured true values using Euclidean distance.

$$RMSE = \sqrt{\frac{1}{N} \cdot \sum_{i=1}^N (y_{i \text{ pred}} - y_{i \text{ true}})^2} \quad (2-14)$$

- $R^2$  score: It shows the proportion of the variation in the dependent variable that can be predicted from the independent variable(s) [26].

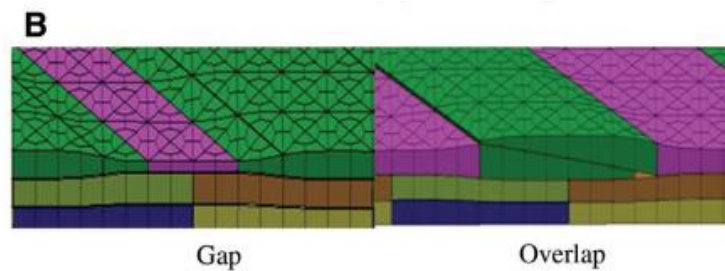
$$R^2 = 1 - \frac{\sum_{i=1}^N (y_{i \text{ pred}} - y_{i \text{ true}})^2}{\sum_{i=1}^N (y_{i \text{ true}} - y_{\text{average true}})^2} \quad (2-15)$$

Also, machine learning models can be used as a substitute of time-consuming and expensive finite element analysis. Once trained, the machine learning model is able to predict the output of the process without relying into the finite element analysis. This characteristic allows to run the calculations much faster, but one of the main problems is data acquisition and the amount of that required to train the model, which can take a lot of simulation time.

### 3 State of art

#### 3.1 FEM models for AFP defects

Although the mechanical properties, like Young modulus, are obtained through mechanical testing, not every defect configuration can be tested. That's why the creation of a finite element model, which can simulate laminates with gaps and overlaps, it's beneficial. FEA can reduce the costs of iterating in these defective laminates FE models before prototyping and testing, but its results are approximated. Li et. al. carried out a study where the effects of gaps and overlaps are studied [1]. The study consists of a 3D finite element model with an adaptive mesh, which reflects the change of thickness that the defect generates into the laminate.



**Fig 3-1: Mesh adaptation to the change of thickness. [1]**

To investigate the features of gaps and overlaps, trial specimens using IM7/8552 pre-preg with layup  $[45, 90, -45, 0]_{3s}$ , each of the plies is 0.25 mm thick. Two-millimeter gaps and overlaps were introduced in the innermost 45-degree plies. It is found that overlapping plies merged at the overlap zone, and plies at gaps have a tendency to flow into and fill the gaps, for this reason the model takes into account the fiber waviness that is generated because of the defects. It is also stated that the defects have a minor effect on the shear, transverse and through-thickness modulus of the plies, so the only relevant effect of the defects occurs just in the fiber direction. A model of 30 mm x 70 mm is created, using constant stress solid elements with one integration point and local orthotropic material axes, which are determined in correspondence with the waviness of the fiber. For the simulation, a mesh size no larger than the design parameters of the defects is selected in order to capture sufficiently the features of the defects. Four millimeters is set as the

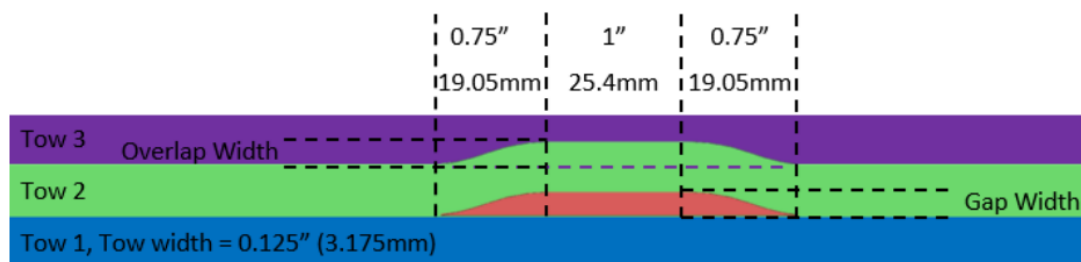


maximum defect size, since it is considered the maximum that can possibly occur in production by the project industrial partner. Defects were located in 90 or 45/-45 degrees plies with or without stagger. Using the Weibull statistical failure criterion and as load case compression or tension tests were simulated. The simulation was run in a high-performance computer. After nine hours of computation, the simulation shows good results in correlation with the experimental tests. The following conclusions are obtained:

- Defects in the 45° or -45° plies have a larger effect on the failure than defects in the 90° plies.
- Defects in 45° and -45° plies have a similar effect on the failure. In 45° and -45° plies, gaps cause a larger knock-down than overlaps in both tension and compression. On the contrary, in 90° plies, overlaps cause a larger knock-down than gaps in both tension and compression

Even though the results are very accurate, nine hours of computation is unacceptable in the present project. The goal of the finite element model is to generate as much data as possible, ideally around one sample every five minutes. But some of the conclusions obtained about the effect of the defects on the material properties can be useful to approximate them. The effects of the defects on the knock-down factor will be also interesting to check the results of the created model.

The creation of a model with a solid element representing each ply will lead to a complex model, which will take a long time to complete simulations. For this reason, in 2019 an experimental work was carried out by Daniel Del Rossi, where the author tested the effect of half gaps and overlaps via finite element analysis and experimental work [4]. In the study a simplified FE model was created, where the adjacent plies with no defects were grouped together in one part. This would reduce the number of interactions between parts and therefore the computational time. The laminate used was [45, 45, 0, -45, 90, 45, 0, -45, 90]<sub>s</sub>, with a number of defects between 4 and 10.



**Fig 3-2: Schematic representation of a gap. [4]**

As it is shown in Fig 3-2, for the FE model, the defect length is fixed at  $2,5'' = 63,5 \text{ mm}$  and width is varied between  $0,05'' = 1,27 \text{ mm}$  and  $0,1'' = 2,54 \text{ mm}$ . Resin properties were assigned to the gap defects. For overlap defects prepreg material, with some modifications to reflect the change in the fiber volume fraction, was assigned. Both defects are modeled with a trapezoidal shape, to simplify the model fiber steering is not considered.

Continuum shell elements SC8R with second order accuracy were used for the whole model. A simple uniform rectangular grid was chosen for the plies with no defects. For the defective plies, in the defect zone, hexahedral continuum shell elements were used allowing easier node generation around the thin defects, outside the defect zone a uniform rectangular mesh is still used.

The boundary conditions are chosen to be as similar as possible to the experimental tests: one end of the coupon has x-displacement fixed while the other has an x-displacement prescribed to induce the loading. Then in order to prevent the coupon from translating or rotating a few points on either end were fixed in the z or the y directions. Once the boundary conditions are applied and using the Hashin Failure Criterion, the following results are obtained:

defect qty	Defect angle / width of defect					
	0/0.05	0/0.1	90/0.05	45/0.05	45/0.1	mix/0.05
4	-3.16%	1.84%	0.73%	-0.79%	-3.44%	-4.96%
7	1.54%	4.03%	2.67%	3.32%	-0.60%	N/A
10	5.37%	N/A	N/A	-7.50%	N/A	N/A

**Tab 3-1: Difference between experimental and simulation results. [4]**

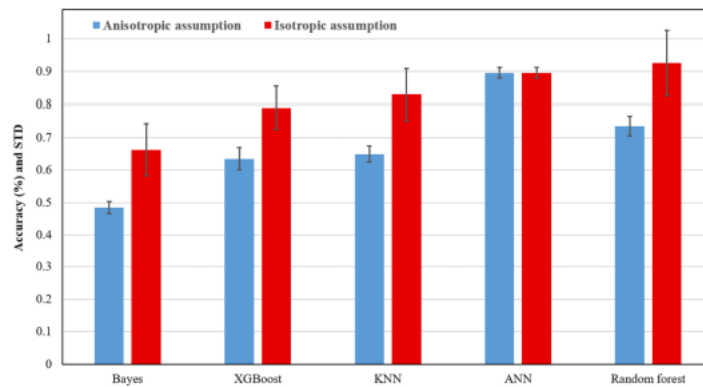
As it can be seen in the Tab 3-1, the tests performed showed that the results of the simulation were matching the experimental results with a difference lower than a 5%,

except for the coupons with high number of defects. This study can have an application in this project, even though the computational time is mentioned to be a few hours. Probably further simplification of the model will be necessary to reduce the computational time.

## 3.2 ML application for FEM replacement

In 2021 Mahziyar Darvishi et. al. carried out a study where the proper cellular structure for the internal mechanical characteristics was predicted [27], by implementing machine learning algorithms based on finite element analysis results of cellular structures. To do that a simplified ABAQUS model, using B31 beam element instead of solid 3D elements, is created and 200 data points were created, consisting of 3 inputs (relative density, elastic modulus and relative yield stresses) and 1 output (cellular structure). Once the data is generated, the following algorithms were trained and analyzed:

- Simple Naïve Bayesian classification model: assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature [15].
- Random Forest: Model based on decision trees, creates multiples trees and averages the results to improve the predictive accuracy and control over-fitting [15].
- K-Nearest Neighbors: the algorithm finds and labels K equal to the input parameter from the nearest neighbors of the data observed in the training phase. Then, it finds the most repeated label number using the entire data set and examining the K neighbor's labels [15].
- XG Boost: is an ML algorithm based on the decision tree but uses the Gradient Boosting framework. This algorithm optimizes the gradient boosting algorithm using parallel processing, tree pruning, handling missing values, and regularization to prevent overfitting or bias.
- Artificial Neural Network (ANN): A fully connected feed forward NN is used with one hidden layer, the number of neurons in the hidden layer is determined via iterative methods and the one with better performance is selected.



**Fig 3-3: Accuracy of the trained models [27]**

As it can be seen on the graph, the ANN is the most accurate among the implemented algorithms. The random forests method has also a great accuracy greater than 70%, while the accuracy of K-nearest neighbors and Bayes classification are below 65%. In this study the use of finite element analysis data is successfully used to create a classification machine learning model able to predict the proper cellular structure, it is also possible that in the current project a regression machine learning is required. It is also interesting the simplification of a 3D element to a 1D-beam element to reduce the computational time.

Along these lines, Qi Zhenchao performed a study where the mechanical properties of a fiber monofilament could be predicted with a machine learning model given the mechanical properties of the laminate and the resin [28]. A parametrized finite element model is created to generate 500 samples to train and validate the machine learning model. The dataset consists of 5 inputs and 4 outputs, as it can be seen in the table.

Number of the features.									
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
$E_1/E_2$	$E_3$	$G_{12}$	$G_{23}/G_{13}$	$mE$	$mv$	$fE_1$	$fE_2/fE_3$	$fG_{12}/fG_{13}$	$fG_{23}$
CFRP Input data				Matrix		Fiber Output data			

**Tab 3-2: Input / Output parameter definition [28]**

In the study feature selection as a data processing method is used, in order to obtain the features with the greatest relevance from all feature sets and reduce overfitting. The property prediction is based on a decision tree regression algorithm. The tree depth represents the complexity of the model; thus, selecting the optimal model selects the

optimal tree depth. Deeper trees produce models that can more precisely describe the details of the training set. However, a tree that is too deep will cause over-fitting. Therefore, in order to obtain an appropriate model, experiments are carried out on model trees with multiple depths. By calculating the root mean squared error and the error rate, the comparison between the value of the model and the true value of the test set can be done. With the correct depth, the model is able to predict sufficiently accurately the properties of the carbon fiber monofilament. It is important to note that performing feature selection methods on models with a reduced number of samples will avoid overfitting.

Further study in this area is performed in 2019 by Sang Ye [29], where a Deep Neural Network (DNN) is proposed to predict the mechanical properties of composites with complex microstructures. The goal of this DNN is to substitute the usually used FEM, which calculates the mechanical properties by defining a representative volume element (RVE). The RVE is the minimal representation of the basic microstructure of the composite. Because of the complexity of the microstructures studied, the FEM calculation is usually tedious and highly time-consuming.

Ye [29] generates a large number of RVEs with various types of complex structures. The microstructural images of composites, and the mechanical properties of the constituent materials as well, are stored in the database. Using simulation their effective mechanical properties are obtained. With this method multiple data sets are generated, consisting in pairs of images describing the microstructures, the properties of constituents, and the mechanical properties of the composites. With these data sets, a DNN model is trained to predict the mechanical properties of a composite from its microstructural image. 20,000 samples are created, which are randomly divided into training, validation, and test parts in the ratio of 3:1:1.

The DNN architecture consists of a sequence of computationally nonlinear layers, which are able to gradually extract representations of images with higher-level abstractions. It has five convolutional layers connected by four fully connected layers. The gray image corresponding to the microstructure of a composite is input at the beginning of the ConvNet, and then the corresponding mechanical properties (Young's modulus and Poisson's ratio) are derived at the output layer of the DNN. Different activations functions are used for the output layer:

- For Young's modulus, the rectified linear unit (ReLU) function is chosen
- For Poisson's ratio, a variant form of Sigmoid function is used

Batch normalization method is also used to avoid exploding and vanishing gradients. This method consist in normalizing the output of a hidden layer so the subsequent layer will learn better [16]. To quantify the difference between the effective mechanical properties of the DNN's prediction and the real value and then adjust the parameters, mean square error (MSE) is used since predicting the mechanical properties is a regression problem.

After the training phase the model is able to predict the Young's modulus with a relative error of 12%, but almost all samples have a relative error lower than 4%. The predicted Poisson's ratio is also sufficiently accurate, with the relative errors smaller than 3%

It is demonstrated that this DNN model can accurately and efficiently predict the mechanical properties of composites with various complex microstructures.

## 4 Data generation

The following chapter aims to develop a method that will be capable of generating data extracted from the tensile testing simulation of a defective composite laminate coupon. This data will be used as input of a machine learning algorithm in order to predict the effects of the defects on the effective longitudinal stiffness of a composite laminate. To create such methodology, the approach will be divided in two main parts:

- The need of having proper data as an input for the ML algorithm, creates the necessity of developing a FE model in ABAQUS that can simulate the effects of the defects in a coupon. It is also required understanding in composite materials and FEA in order to interpret the results of the simulation and extract effective stiffness.
- Once the FE model is created and validated, it will be parametrized in order to represent every possible defect configuration. Then it will be inserted into a loop to generate random samples of the FEA.

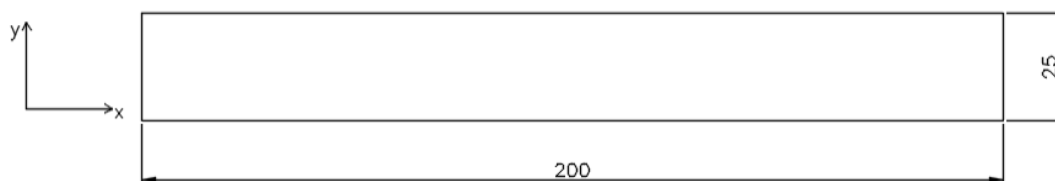
### 4.1 FEM model

This chapter includes the description of the coupon geometry as well as its lay-up, how to characterize a defect and the defect configurations that will be studied. It will also be defined the applied boundary conditions for tensile testing, and the suitable mesh at each study case. Finally, a detailed description of the calculation of the effective stiffness of the laminate will be given.

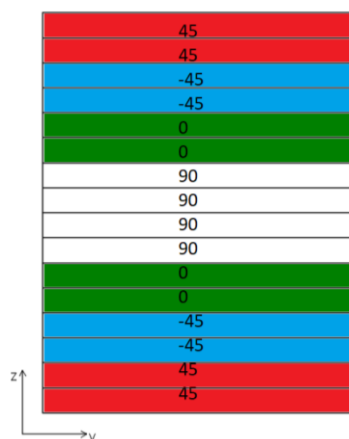
#### 4.1.1 Geometry and material properties

Multiple specimen geometries can be used to create a FEM for tensile testing. Haug [30] performed a study where the mechanical properties of a defective composite coupon were

calculated via mechanical testing. For better comparability the same laminate as Haug [30] is used. This consists in a quasi-isotropic and symmetric laminate with 0,125 mm ply thickness and 16 plies:  $[45_2, -45_2, 0_2, 90_2]_s$ , see in Fig 4-2. In this research the tensile testing is set up according to DIN EN ISO 527-1 [31], where the geometric dimensions testing specimen are 200 x 25 x 2 mm, as shown in Fig 4-1.



**Fig 4-1 Coupon geometry**



**Fig 4-2 Ply stacking sequence**

For reasons of comparability, the coupon geometry and the ply stacking sequence used will be the same as in [30]. The prepreg material used will be IM7/8552 [32], its properties can be seen in Tab 4-1 and in Tab 4-2.



Tab 4-1 IM7/8552 prepreg ply properties [32]

$E_{11}$ (GPa)	$E_{22}=E_{33}$ (GPa)	$G_{12}=G_{13}$ (GPa)	$G_{23}$ (GPa)	$\mu_{12}=\mu_{13}$	$\mu_{23}$
161	11,4	5,17	3,98	0,32	0,435

Tab 4-2 Resin material properties [32]

E (GPa)	$\mu$	Material Type
3,7	0,4	Isotropic

### 4.1.2 Defect properties

To characterize the defects the guidelines of Li et. al. in [1], were followed. For the defective gap regions resin material properties were used. For the overlap defects Li concluded that the overlaps defects have a minor influence on the shear and transverse-to-thickness moduli and can be neglected. The changes in the material properties are reflected in the equations (4-1), (4-2), (4-3) and (4-4). The overlap resultant material properties are listed in Tab 4-3.

$$E_{11\_overlap} = 2 * E_{11\_prepreg} \quad (4-1)$$

$$E_{22\_overlap} = E_{22\_prepreg} ; E_{33\_overlap} = E_{33\_prepreg} \quad (4-2)$$

$$G_{12\_overlap} = G_{12\_prepreg} ; G_{13\_overlap} = G_{13\_prepreg} \quad (4-3)$$

$$G_{23\_overlap} = G_{23\_prepreg} \quad (4-4)$$

Tab 4-3 Overlap material properties

$E_{11}$	$E_{22}=E_{33}$	$G_{12}=G_{13}$	$G_{23}$	$\nu_{12}=\nu_{13}$	$\nu_{23}$
322	11,4	5,17	3,98	0,32	0,435

### 4.1.3 Part and assembly generation

Once the geometry and the material properties are defined, the model has to be generated in ABAQUS. The approach of Rossi in [4], which validates a FEM where all the consecutive plies without defects can be grouped together in one part, will be used. One of the main problems of this approach was the large computational time, to reduce it, shell elements instead of solid elements will be used. With regard to the number of defects, the following situations will be studied:

- No defective plies, pristine condition coupon
- One defective ply in laminate
- Two defective plies, just if the plies are consecutive and have the same fiber orientation. The defect of the second ply is exactly the same as in the first one

For the plies with no defects a rectangular shell with dimensions 200 x 25 x 2 mm will be generated. For the modelling of the defective plies part, the defect section will be added according to the following guidelines. For 90, 45 and -45 degrees fiber orientation plies, see at Fig 4-3:

- The defect position inside the coupon will be the distance between the right edge of the coupon and the right top corner of the defect.
- The defect width will be set with the distance between the right top corner and the left top corner of the defect.

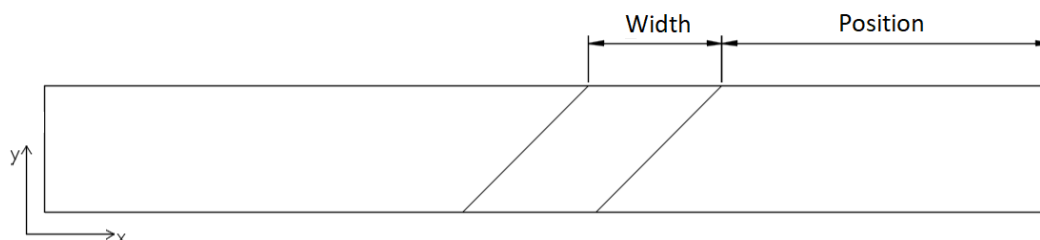
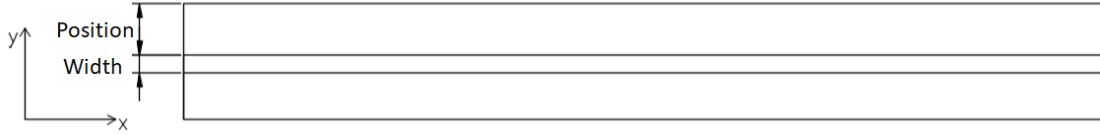


Fig 4-3 Defect characterization for the 45 degree ply

For the 0 degrees ply, see in Fig 4-4:

- The defect position inside the coupon will be the distance between the top edge of the coupon and the top left corner of the defect.
- The defect width will be set with the distance between the top left corner and the bottom left corner of the defect.



**Fig 4-4 Defect characterization for the 0 degree ply**

In the pristine condition case two FE models will be generated. On the first one the whole laminate will be modelled as one shell, where the laminate properties will be assigned. The second pristine coupon FE model will be a three-shell model, where the middle shell will just contain ply 8 and the top and bottom shell the remaining plies.

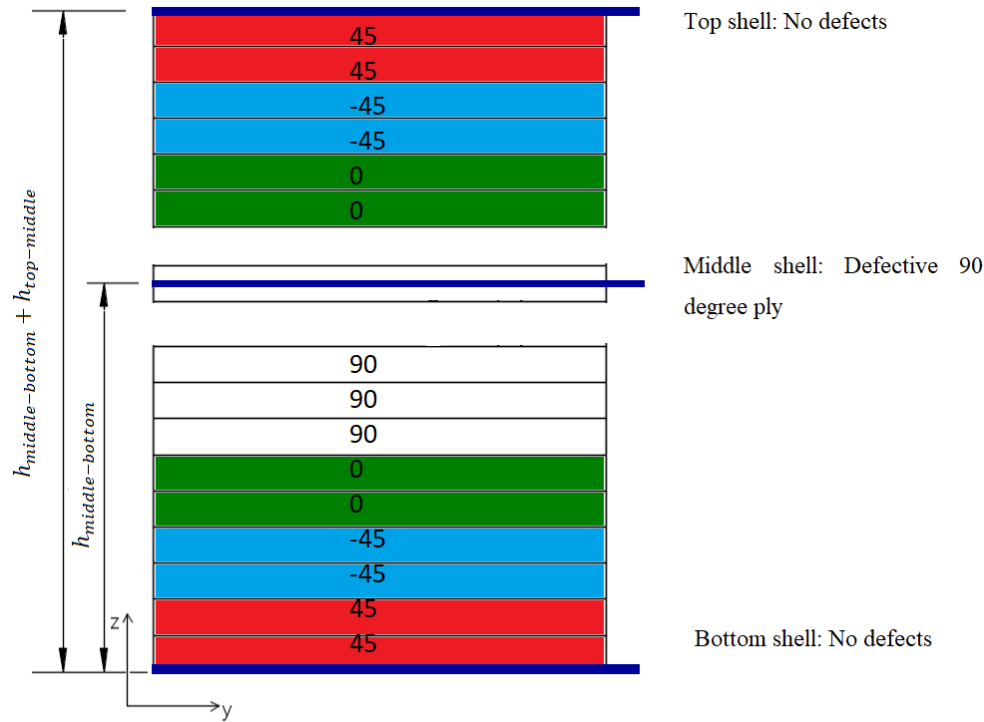
In the defective condition, the shell model of the defective plies is created. Depending on the ply where the defect is located two possible assemblies will be generated:

- When the defective ply is not at the top or bottom ply, a three-shell assembly will be created. The middle shell will contain the defective ply, while the top shell and bottom shell will contain the remaining non defective plies. The top, bottom and middle shells will be assigned as top, bottom and middle surface respectively. By assigning this properties ABAQUS will generate the laminate according to Fig 4-5. The top shell will be located in the assembly at top surface of the laminate, 2 mm above the bottom surface, while the middle shell will located following equations (4-4) and (4-5). A tie constraint between the shells is applied to keep the shells attached to each other during the simulation.

$$h_{middle-bottom} = (16 - ply\_n) * t + \frac{t_{defect}}{2} \quad (4-5)$$

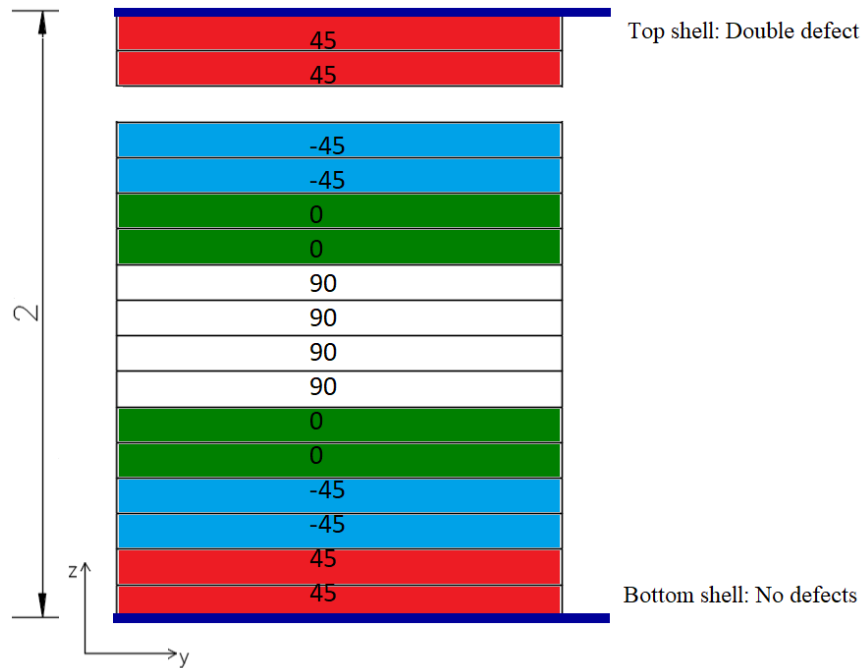
$$h_{top-middle} = (ply\_n) * t + \frac{t_{defect}}{2} \quad (4-6)$$

Where,  $ply\_n$  is the number of ply where the defect occurs,  $t$  is the thickness of the ply and  $t_{defect}$  is the thickness of the defective ply, which is equal to  $t$  in case of single defect or  $2 * t$  when the defect is double.



**Fig 4-5 Three-shell assembly model**

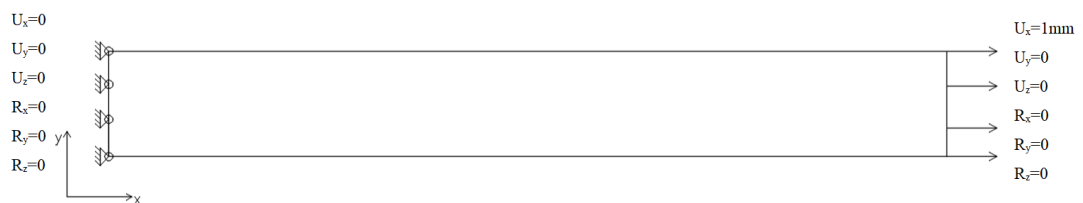
- When the defective ply is at the top or bottom ply, a two-shell assembly will be created. The top shell will contain the defective ply, while the bottom shell will contain the remaining non defective plies, as it is shown in Fig 4-6. The top and bottom shells will be assigned as top, bottom surface respectively. The distance between the shells in the assembly is set as the laminate thickness, 2 mm, and a tie constraint between the shells is applied to keep the shells attached to each other during the simulation.



**Fig 4-6 Two shell assembly model with double defect**

#### 4.1.4 Boundary conditions and loading

Once the assembly is set up, the boundary conditions and the loads will be applied. Once the constraints between the shells are created, the encastre boundary condition is applied at the left edge of all the shells, as it is shown in Fig 4-7. Since the tensile test is performed, following the indications of [4], the load is set up as a displacement of 1 mm in the x-direction on the right edge of the coupon. All other displacements and rotations of the right edge are fixed to 0. The boundary conditions and the load are shown in Fig 4-7.



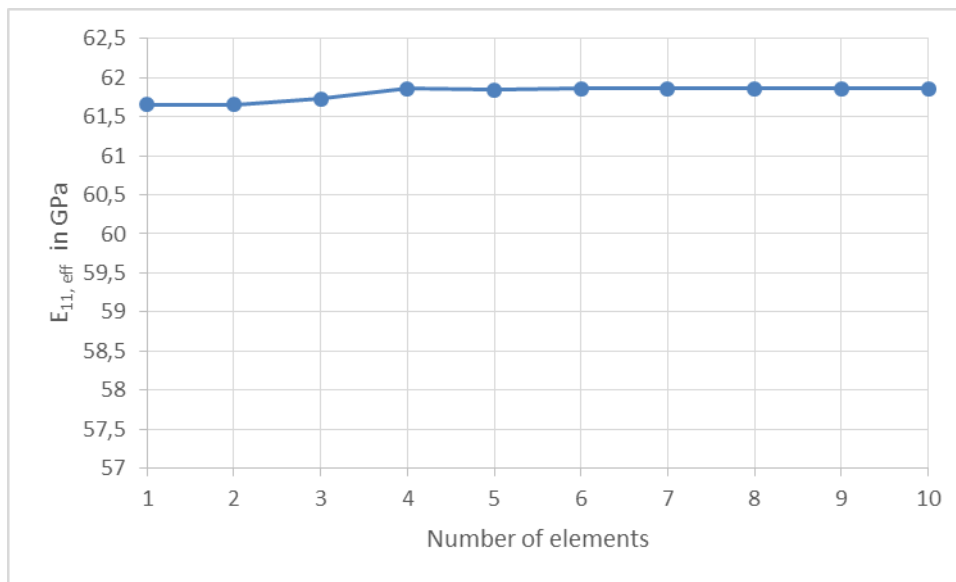
**Fig 4-7 Boundary conditions and load**

#### 4.1.5 Mesh

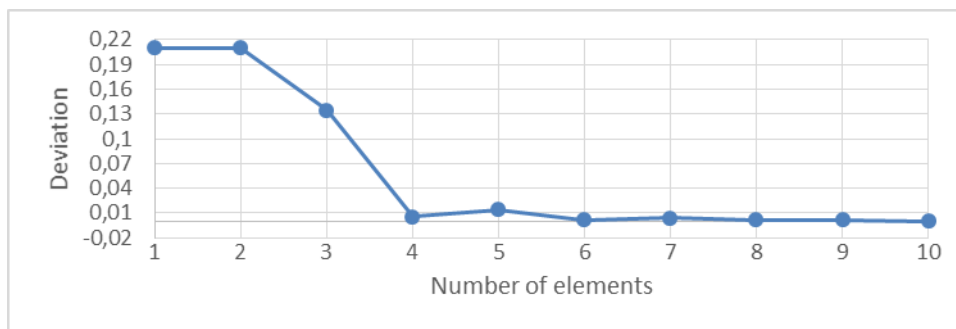
In this section a convergence study of the mesh is performed for each shell configuration, a S4R element is selected during this procedure and the applied defect is a gap. Since the

effective stiffness is predicted with a precision of 2 decimal positions, when the deviation of the output with the reference value is lower than 0,01 it will :

- Pristine coupon: In this model the whole laminate is defined with one shell. By augmenting the number of elements on the shell width, an optimal mesh size of 4,17 mm = 6 elements, see in Fig 4-8 and Fig 4-9.

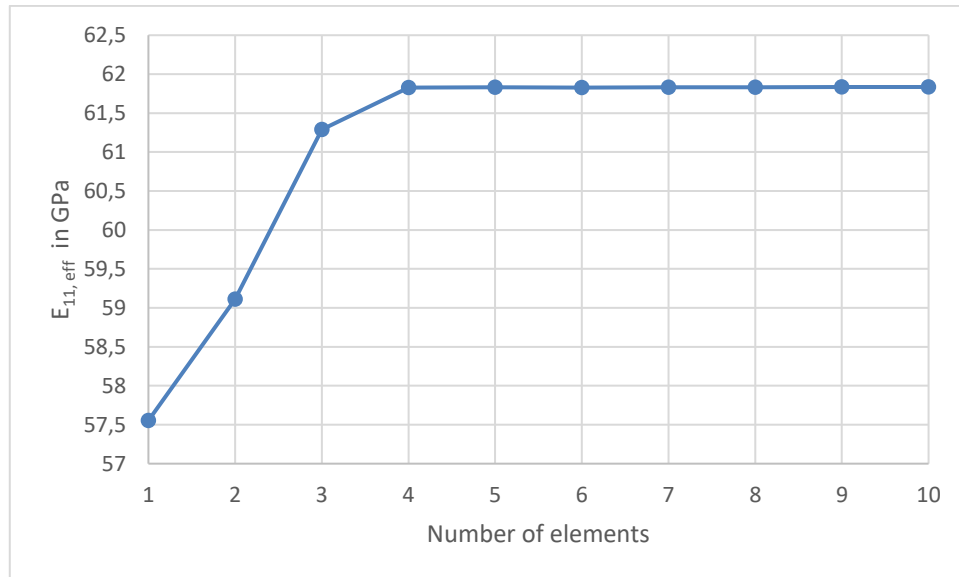


**Fig 4-8 Convergence study 1-shell model in pristine condition**



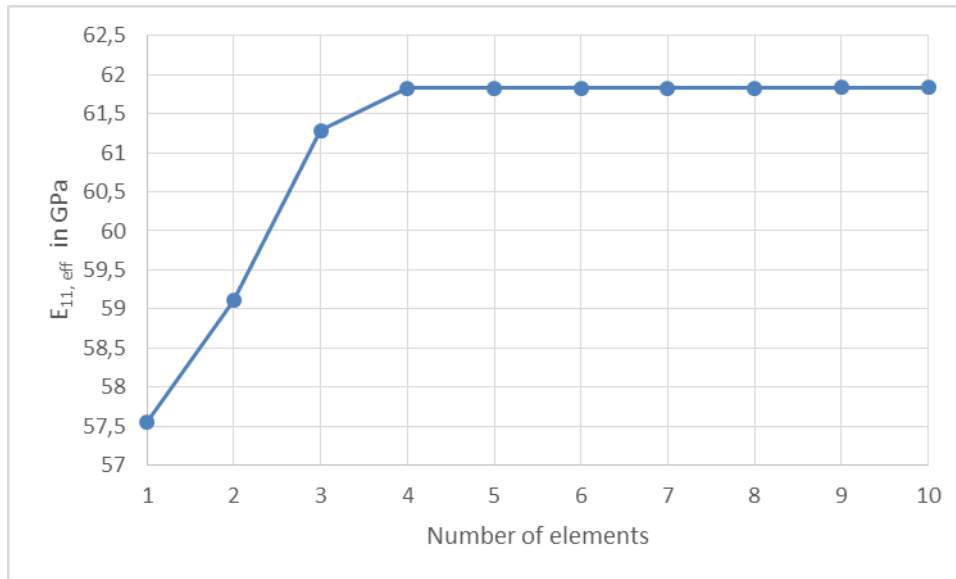
**Fig 4-9 Deviation of the output: 1-shell model in pristine condition**

- Pristine coupon: To validate the shell model, a three shell model in pristine condition is also studied. The mesh size is decreased by adding elements in the width of the shell and the change of the output is monitored. As it can be seen in Fig 4-10 and Fig 4-12, the output value converges at 5 elements in the width of the shell, which correspond to a mesh size of 5 mm.

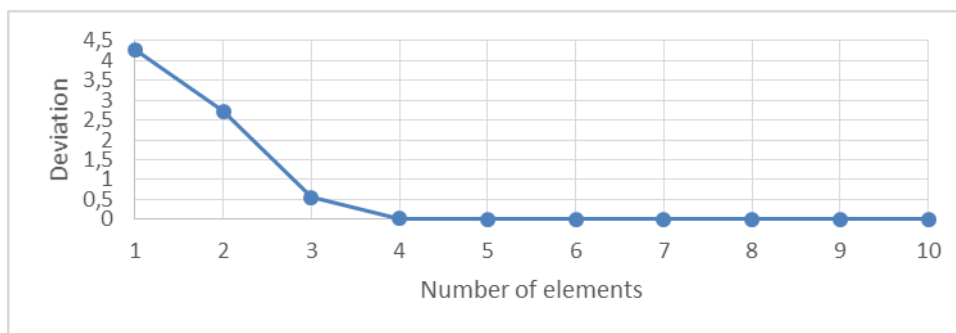


**Fig 4-10 Convergence study 3-shell model pristine condition**

- 90 degree defective plies: A defect of 5 mm in ply 8 was introduced. A first study is performed decreasing the global mesh size for the whole shell. As it can be seen in the Fig 4-11 and Fig 4-12, the output value converges at 5 elements in the width of the shell, which correspond to a mesh size of 5 mm.

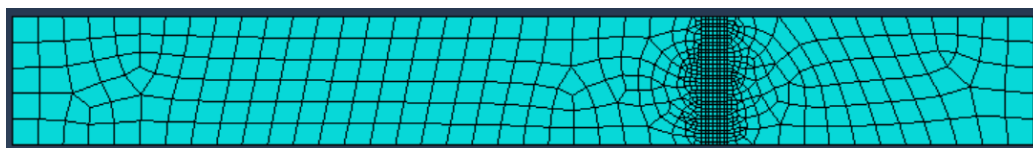


**Fig 4-11 Global mesh convergence study of 90 degree 5 mm defect in ply 8**



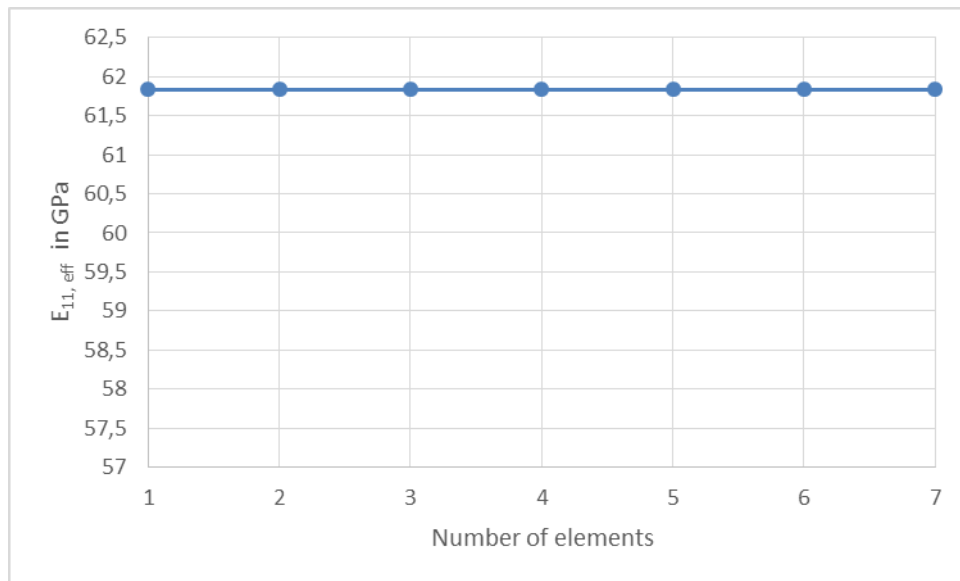
**Fig 4-12 Deviation of the of the output, 5 mm defect in ply 8**

- 90 degree defective plies: A second study is performed: maintaining global mesh size of 5 mm on the shell, but applying a finer mesh is on the defective region, as it can be seen in Fig 4-13. The mesh sizes is decreased by adding elements at the defect width. As it can be seen in Fig 4-14 and Fig 4-15, the finer mesh doesn't have an influence on the output. The mesh size for 90 degrees defects is set at 5 mm.

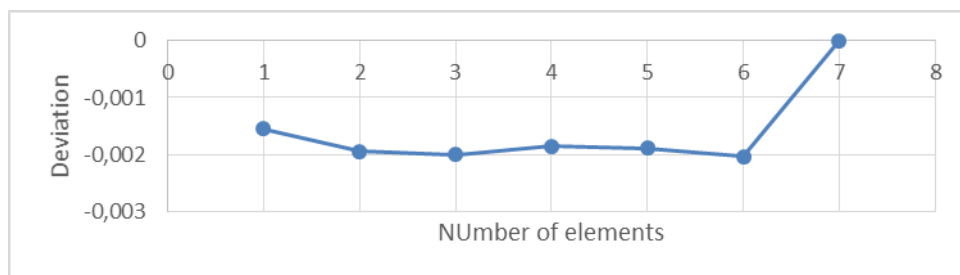


**Fig 4-13 Finer mesh on the defective region**



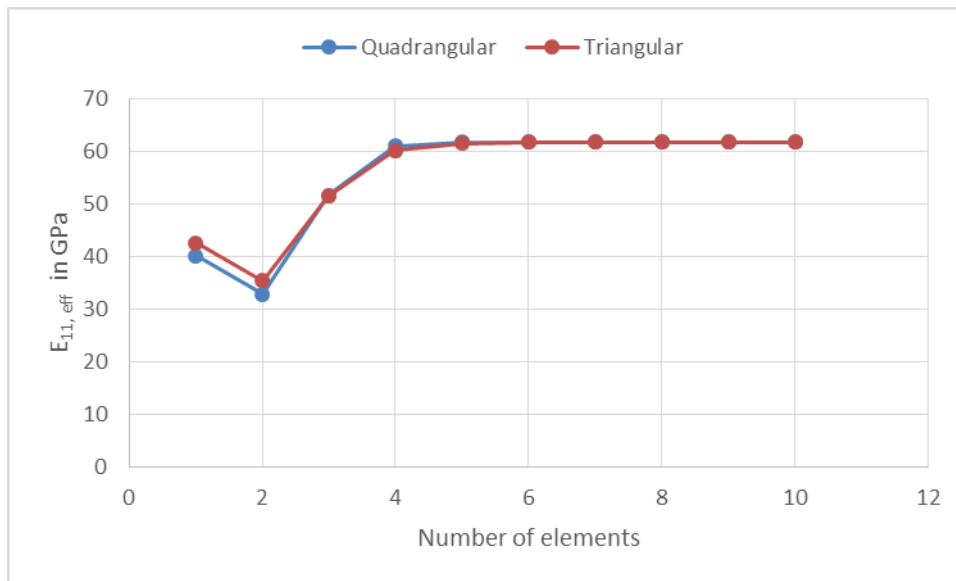


**Fig 4-14 Finer mesh convergence of 90 degree 5 mm defect in ply 8**

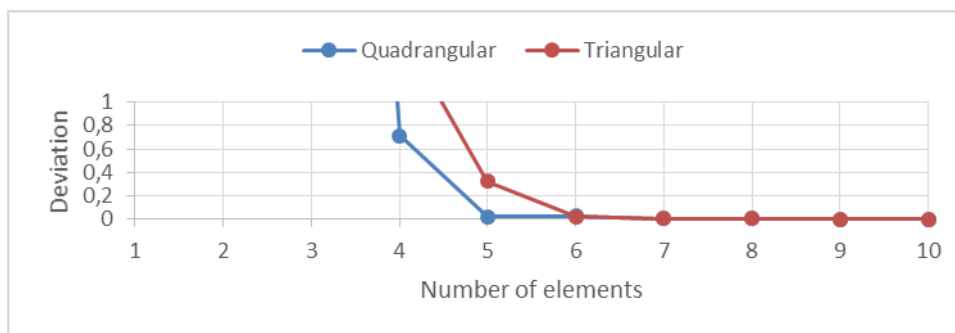


**Fig 4-15 Deviation of the of the output, 5 mm defect in ply 8**

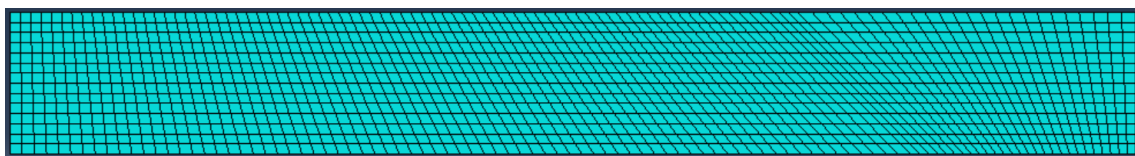
- -45 degree defective plies: A defect of 5 mm in ply 4 was introduced. A first study is performed decreasing the global mesh size for the whole shell. As it is shown in Fig 4-16 and Fig 4-17, the output value converges at 7 elements in the shell width, which corresponds to a mesh size of 3,5 mm. But as it can be seen in Fig 4-18, the quadrangular mesh elements are deformed around the inclined defective region.
- -45 degree defective plies: A second study is performed with triangular elements which are less susceptible to this deformation. The results in Fig 4-16 and Fig 4-17, show that the results of the triangular mesh have almost no difference compared to the quadrangular elements. Since a low computational time is important for the present project, the quadrangular mesh element with a mesh size of 3,5 mm is selected.



**Fig 4-16 Mesh convergence study of 5 mm defect in ply 4**

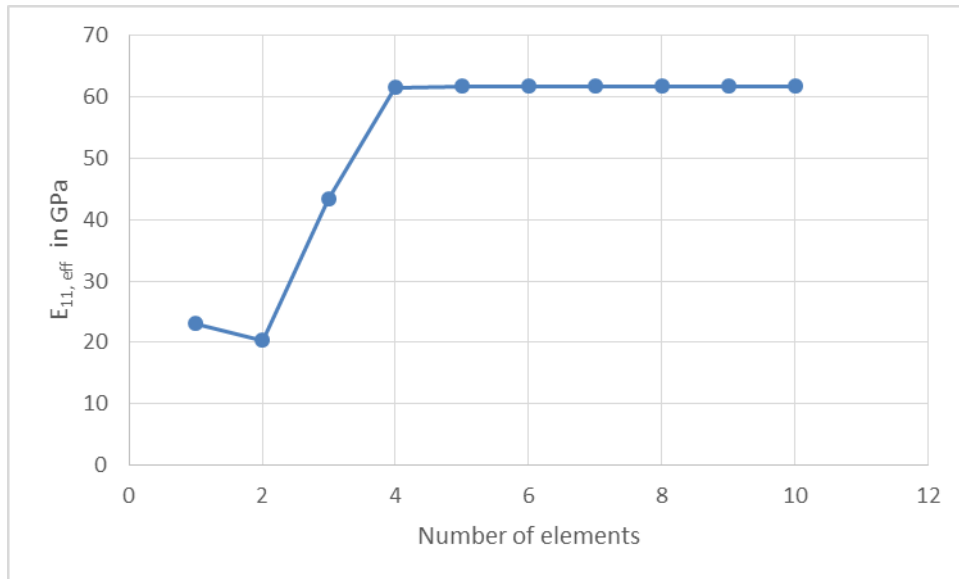


**Fig 4-17 Deviation of the of the output, 5 mm defect in ply 4**

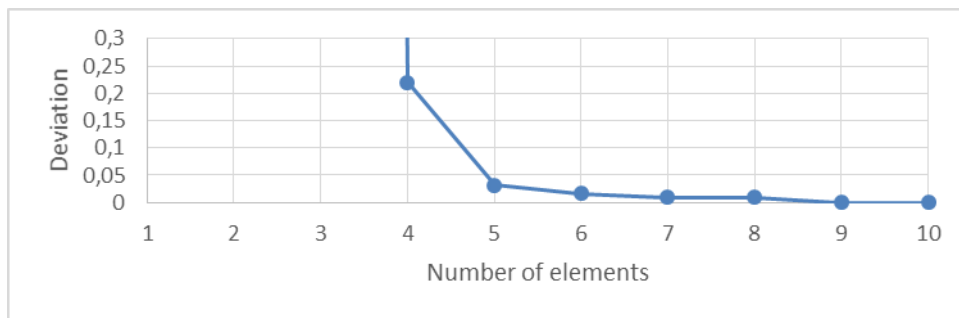


**Fig 4-18 Quadrangular mesh deformation**

- 45 degree defective plies: Since this case is similar to the -45 degree defective ply, just the quadrangular meh will be studied. A defect of 5 mm in ply 2 is introduced. The study is performed decreasing the global mesh size for the whole shell. As it is shown in Fig 4-16 and Fig 4-17, the output value converges at 7 elements in the shell width, which corresponds to a mesh size of 3,5 mm. But as it can be seen in Fig 4-18, the quadrangular mesh elements are deformed around the inclined defective region.

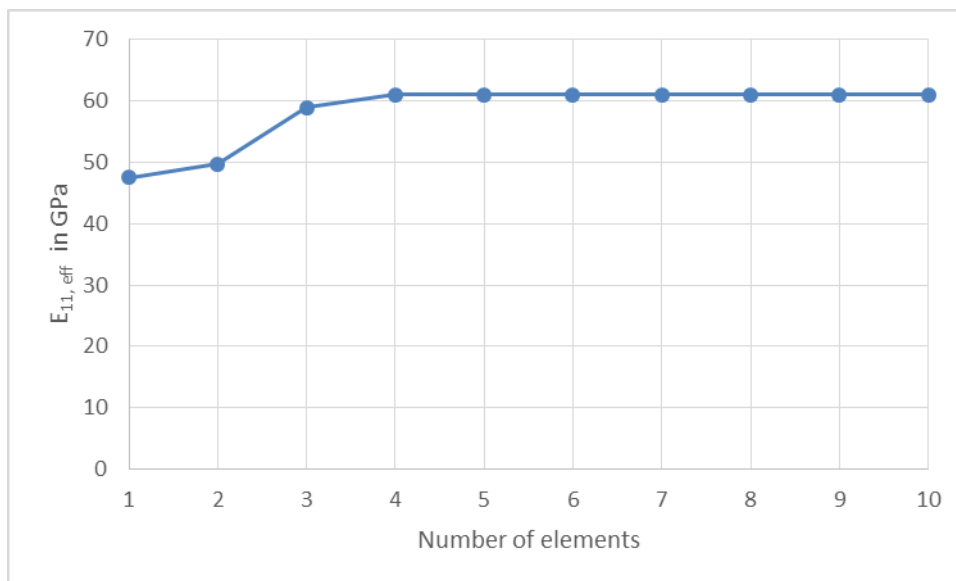


**Fig 4-19 Global mesh convergence study of 5 mm defect in ply 2**

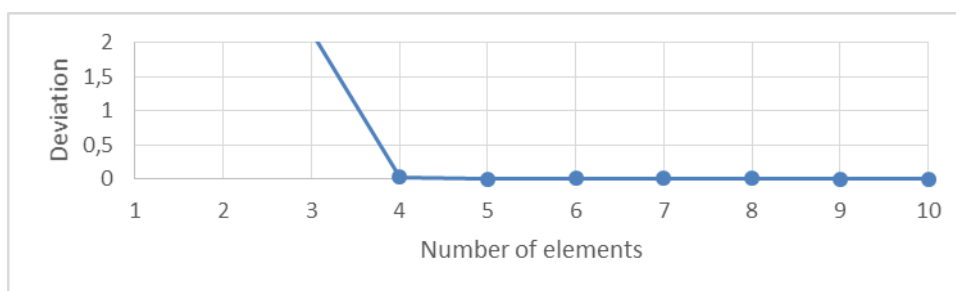


**Fig 4-20 Deviation of the output of 5 mm defect in ply 2**

- 0 degree defective plies: A 2 mm width defect at ply 6 is introduced. The study is performed to decrease the global mesh size for the whole shell. As it can be seen in the Fig 4-21 and Fig 4-22, the convergence is achieved at 4 elements in the shell width, which corresponds to a mesh size of 6,25 mm.



**Fig 4-21 Mesh convergence study of 2 mm defect in ply 6**



**Fig 4-22 Deviation of the output, 2 mm defect in ply 6**

On the following table a summary of the mesh size used in each case. Each study case will have its own mesh depending on the ply fiber orientation where the defect occurs and the ply location in the layup.

Shell/Defect	No defect	90 degree	45/-45 degree	0 degree
Top	5	/	3,5	/
Middle	5	5	3,5	6,25
Bottom	5	/	3,5	/

### 4.1.6 Results

Once the mesh is applied to each shell, the simulation can be run. Since the objective of the project is to predict the effects of the defects on the laminate, the longitudinal effective stiffness ( $E_{11,eff}$ ) of the laminate will be calculated following the equation (4-3).

$$E_{11,eff} = \frac{\sigma_{11,eff}}{\epsilon_{11}} \quad (4-7)$$

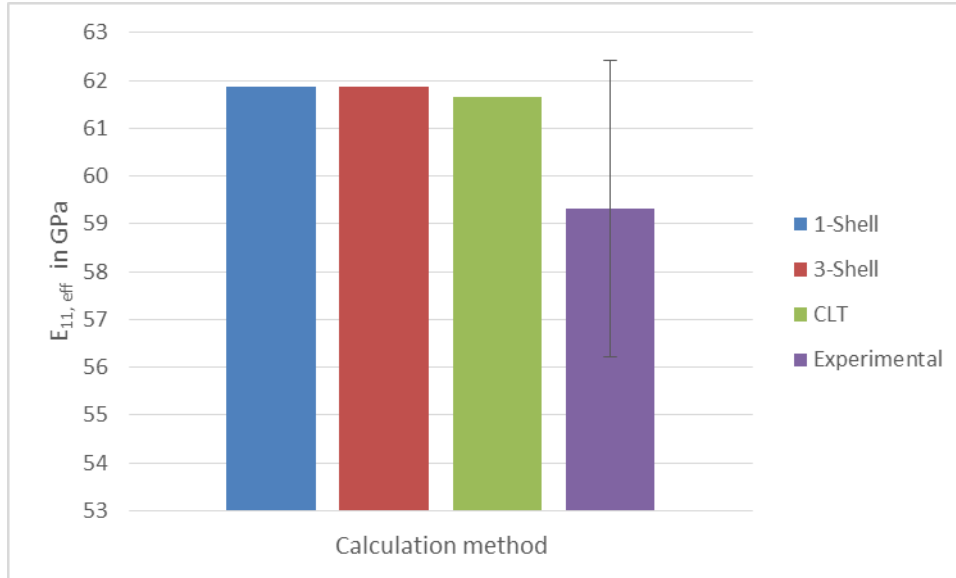
Where  $\sigma_{11,eff}$  and  $\epsilon_{11}$  are the stress and the strain in longitudinal direction. To calculate  $\sigma_{11,eff}$ , the reaction force in longitudinal direction ( $RF_1$ ) at the loaded edge will be extracted as the output of the simulation. Using the equation (4-4) and (4-5),  $\sigma_{11,eff}$  and  $\epsilon_{11}$  can be respectively calculated.

$$\sigma_{11,eff} = \frac{RF_1}{A} \quad (4-8)$$

$$\epsilon = \frac{\Delta l}{l_0} \quad (4-9)$$

Where  $A$  is the area of the cross section of the laminate,  $\Delta l$  is the variation of length of the laminate after applying the load and  $l_0$  is the initial length of the laminate.

In order to validate the ABAQUS shell model created the effective longitudinal stiffness of the laminate in the pristine condition will be compared with the analytical and experimental values. The analytical values are calculated using the CLT, see in chapter 2.1, and the experimental results are extracted from the Haug's study in [30]. As it can be seen in Fig 4-23, between the 1-Shell and 3-Shell FE models there is almost no difference, then it can be concluded that the 3-Shell model is a good substitute for the 1-Shell model. Using FE models, the results obtained are closer to the CLT than to the mean of the experimental testing as expected, since in experimental testing human errors during testing or laminate preparation are introduced. With this comparison the 3-Shell FE model is validated.

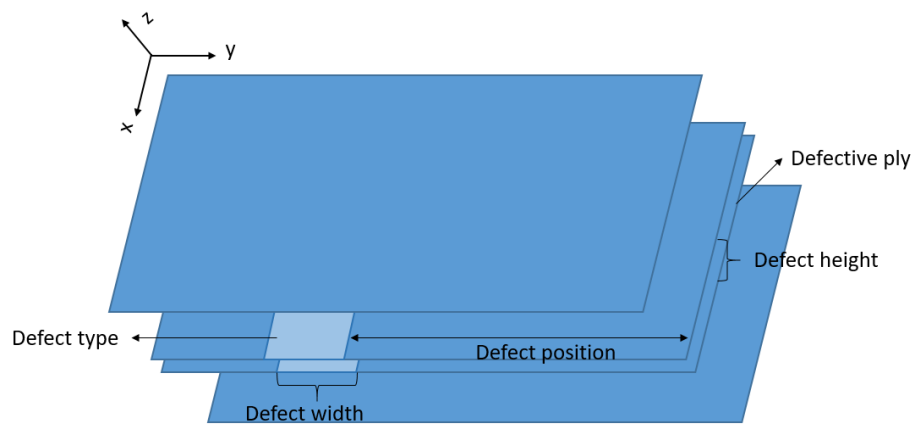


**Fig 4-23 Comparison between calculations methods**

## 4.2 FEM Parametrization

Once the shell model is validated, a parametrization of the FE model it has been done. In this chapter the input parameters of the parametrization will be defined and its limits set. Also, a sensitivity analysis will be performed to determine the dominant parameters. Via this parametrization the dataset that will be used to train a machine learning model will be created.

In this chapter the input parameters of the FE model will be defined. The parameters in Fig 4-24 will be studied:

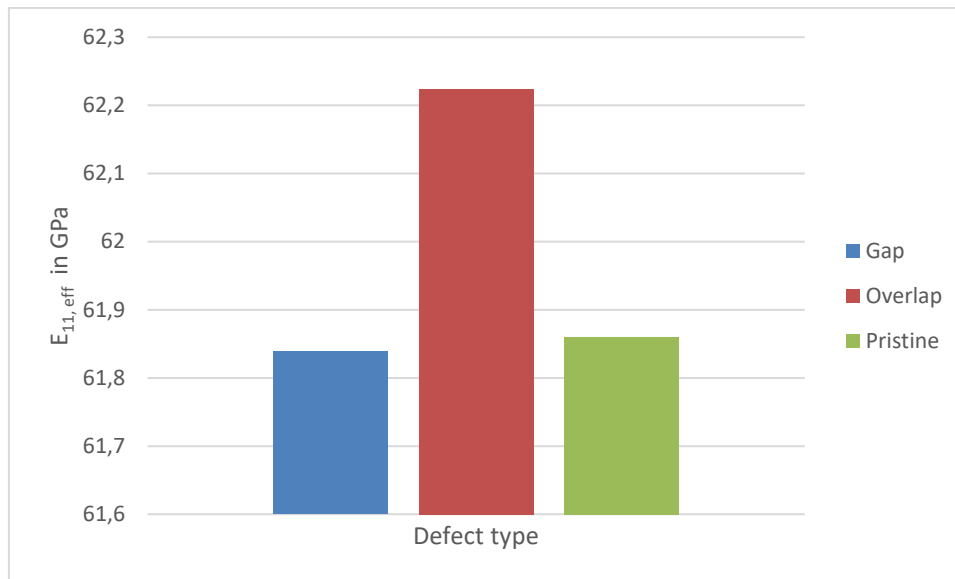


**Fig 4-24 Parameters of the FE model**

Type of defect: It represents the type of defect found between gaps and overlaps. It only changes the material properties of the defective region using the material properties in Tab 4-2 for gaps or using the material properties in Tab 4-3 for overlaps. A defect of 10 mm is introduced on ply 8. As it can be seen in the Fig 4-25, the gap defects slightly lower the effective stiffness, around 0,1 MPa. In the overlap defect case, the effective stiffness is incremented around 0,5 MPa. In Haug's [30] study a defect of width = 3,175 or 6,350 mm is inserted in ply 8, the experimental results show that:

- Gap, single defect: lead to an increment of the effective stiffness.
- Gap, double defect: lead to a decrement of the effective stiffness.
- Overlaps, single and double defects: lead to decrement of the effective stiffness.

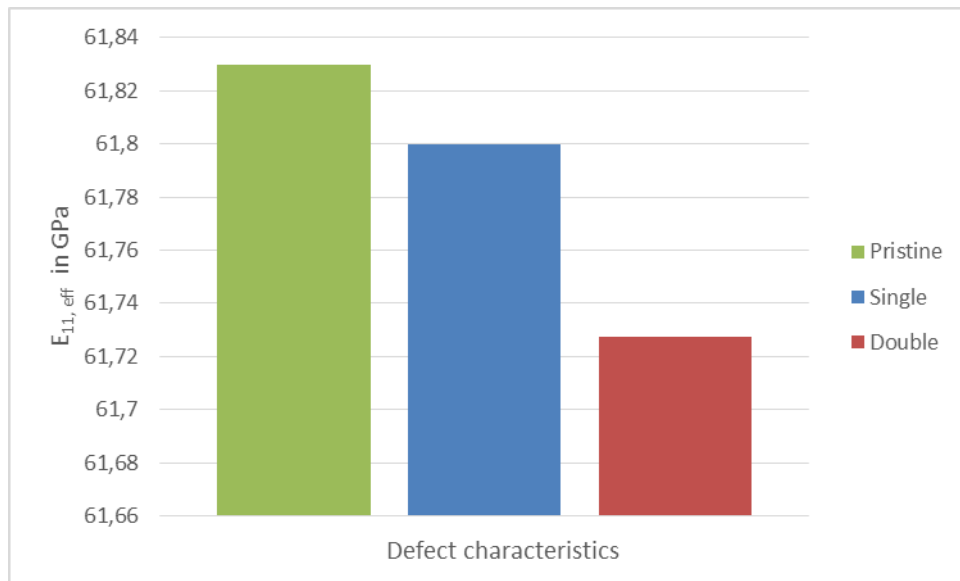
This difference between the simulation of the 3-Shell model and the experimental test can be explained because of the variability of the coupon fabrication, which is around  $\pm 3$  MPa of the mean effective stiffness. With this variability the small variation that introduce the defects (around 0,1 – 1 GPa) turns to not be relevant in front of the manufacturing variability.



**Fig 4-25 Effective stiffness for gap and overlap defects: 10 mm defect in ply 8**

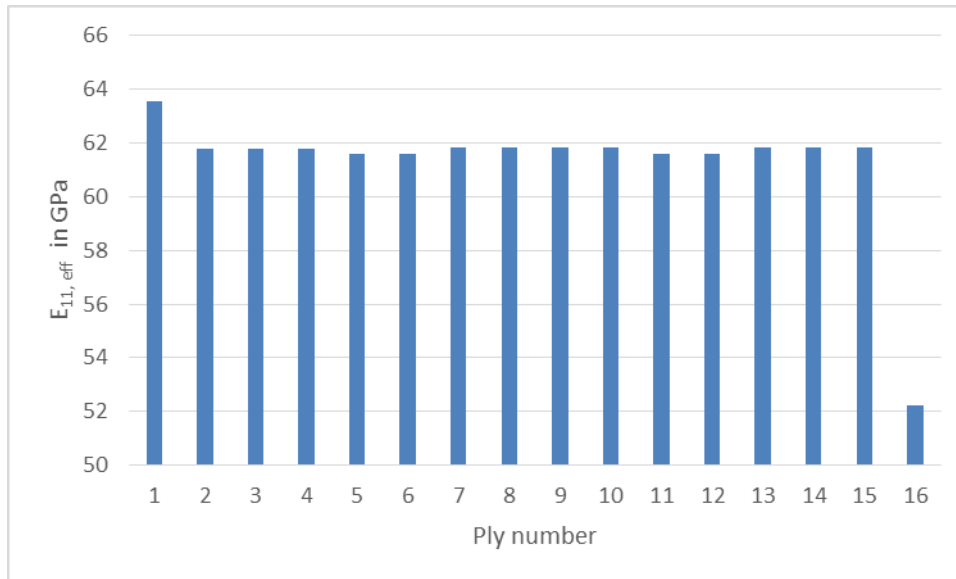
- Double/Single defect: This parameter is represented in the parametrization as a 1 in case of having a single defect or as a 2 in the case that the defect is double. A gap defect of 10 mm is introduced on ply 8. As it can be seen in the Fig 4-26, a double defect reduces even more the effective stiffness of the laminate, but not proportionally as it could be expected. The location of the defect may also have an influence on the effective stiffness. Haug in [30] also mentioned that the defects located in consecutive plies have a higher impact in the laminate stiffness.





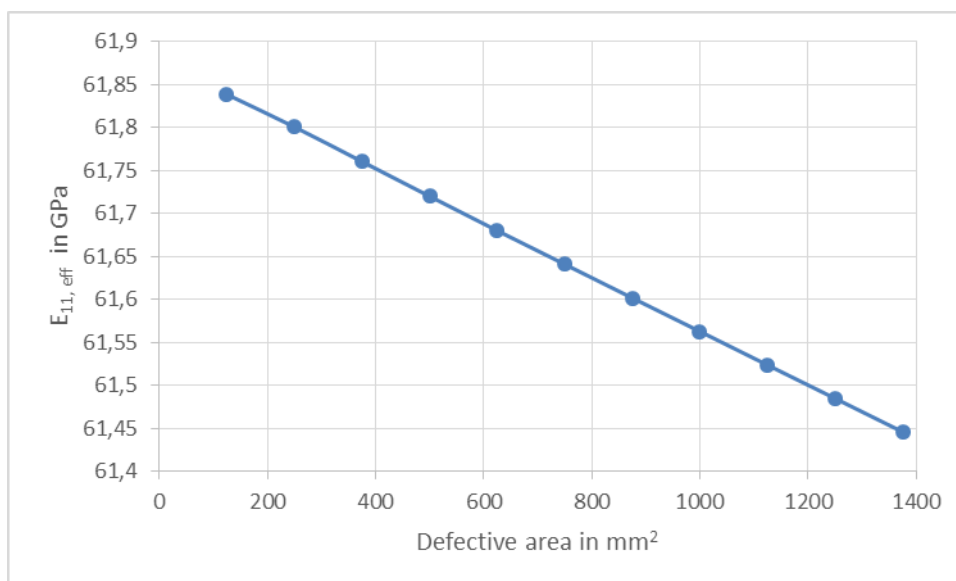
**Fig 4-26 Effective stiffness for single and double defects: 10 mm defect in ply 8**

- Defective ply: This ply is represented in the parametrization as a natural number between one and sixteen. This number indicates in which ply is the defect located, being one the ply number of the ply on bottom of the laminate and 16 the ply number of the ply at the top. A gap defect of 125,0 mm<sup>2</sup> is introduced over all plies. As it can be seen in the Fig 4-27, the effective stiffness is slightly influenced for the location of the defect in the laminate. Being more relevant the effects produced of a defect located on the outer surfaces of the laminate, than in the middle. Also the 0 degrees defects have a higher influence on the effective strength for the same defect width. Because of the configuration of the 0 degree defects, see in Fig 4-4, the defects in these plies are much bigger than the other ones for the same width. For this reason the area of the defect will be studied as a parameter instead of the defect width. It is also relevant to mention that defects located on plies 1 and 16 produce an abnormal behaviour of the FE model, that's why defects on both top and bottom shells will not be considered in the study.



**Fig 4-27 Effective stiffness of a 125,0 mm<sup>2</sup> gap at different plies**

- Defect area: This parameter is defined as the area of the defective region. In order to not have disturbances at obtaining the output results, the limits of the defects have to be set: The minimum distance between the left/right edge of the coupon and the left/right corner of the defect, will be equal to the size of the mesh element. A gap defect is introduced in ply 8 with variable area. In the Fig 4-28 it can be seen, that for increasing defective area, the effective strength of the laminate will decrease linearly.



**Fig 4-28 Effective stiffness for increasing defect area in ply 8**

- Defect position: This parameter is defined as the distance between the right edge of the coupon and the right corner of the defect. A defect of 10 mm width is implemented at the ply 8 and its position inside the defect is arbitrarily changed. As it can be seen in Fig 4-29, the position has no influence on the output. This parameter is eliminated of the FEM parametrization.



**Fig 4-29 Effective stiffness for variation of the defect position: 10 mm defect in ply 8**

In the following table, Tab 4-4, an overview of the parameters with its available range it's shown.

**Tab 4-4 FE model parameter overview**

Parameter	Variable type	Value range	units
Defect type	Categorical	[ Gap, Overlap]	/
Defet height	Categorical	[ Single, Double]	/
Defect Area	Numerical	[2, 38000]	mm <sup>2</sup>
Ply number	Categorical	[2,15]	/

Once the input parameters are defined, and its limits well described. Following the guidelines of [29], a loop is created to generate 25000 models and compute its effective stiffness. As a result of this loop a file with the following data will be generated: Defect type; Defect height; Defect area; Defect ply number;  $E_{\text{eff}, 11}$ . These 25000 data samples will be used to train the machine learning algorithms described in the following chapter.



## 5 Implementation of Machine Learning Models

This chapter aims to develop machine learning models that can predict the effects of the defects using the data extracted via simulation with ABAQUS. It consists of three sections. In the first section the dataset obtained with ABAQUS will be preprocessed. In the following sections the ML models are defined and finally the process for the hyperparameter tuning will be explained.

### 5.1 Data pre-processing

After the data generation process is completed, a pre-processing phase is needed to transform the input data into data that the ML algorithm could understand better and enhance its performance.

Since ML algorithms don't support categorical variables, they will be transformed to a numerical variable using one hot encoding technique. The categorical variables can be either nominal, for example the defect type variable, or ordinal, like the ply number. Ply number is clearly a categorical variable since its values represent categories that classify in which ply the defect has appeared. Using ordinal encoding can create an ordinal relationship in the data, which is not present on the original data [33].

To perform this pre-process of the data, Pandas library [34] in python is used. Because of the above mentioned processes, the initial variables will be transformed into the following ones:

- Each ply will be now an input variable, which can have the value 1 when the defect is in the ply or 0, when there is no defect on the ply.
- Defect type variable will have value 0 for gaps and 1 for overlaps.
- Defect height variable will have value 0 for single defects and 1 for double defects.

In the case of the defect area a normalization is required since its range of values differs a lot from the range of the other variables. To obtain the defect area in the zero-one range, the min-max-normalization [15] will be used. These transformations will enhance the performance of the ML model. In the Tab 5-1 it can be seen an example of the pre-processed input data.

<b>Input variables</b>																
Def. Type	Def. Height	Def. Area	Ply 2	Ply 3	Ply 4	Ply 5	Ply 6	Ply 7	Ply 8	Ply 9	Ply 10	Ply 11	Ply 12	Ply 13	Ply 14	Ply 15
0	1	0,32	0	0	0	0	1	0	0	0	0	0	0	0	0	0

**Tab 5-1 Pre-processed input data sample**

After pre-processing the dataset is split into training, test and validation. Following the guidelines of [29] the data is split in a proportion of 70% for training, 20% for test and 10% for validation. The segmentation was performed by randomly splitting the data using sklearn python library [15]. The main objective was to obtain a well-distributed test dataset that could be representative enough of the simulations gathered, aiming to validate the different configurations extracted. By saving the training and test datasets, the different models used can be easily compared and evaluated.

## 5.2 Model definition

In this chapter the model implementation in pytorch will be described and the hyperparameters for each model will be defined. The following models will be studied: Multilinear model, Polynomial model, MLP with one hidden layer and MLP with two hidden layers. The first two will be studied in order to be sure that the relationship between the input variables and the output is or is not following this simple models. Afterwards the best model of [27], a MLP with one hidden layer is chosen for the study and finally the MLP with two hidden layers will also be considered.

### 5.2.1 Multilinear model

As explained in 2.4.5 the multilinear regression is defined with a set of parameters and the bias. To create the model, pytorch [35] framework will be used. The pytorch library has to be imported and the class *MultiLinearRegression* has to be created. This class is a

subclass of *torch.nn.module*, it will receive as input the dimension of the input. The class contains the parameters *A* and *b*, which refer to the set of weights and the bias respectively, as is shown in the Code 5-1. The forward pass is defined following the equation (5-1).

```
from torch import nn
class MultiLinearRegression(nn.Module):
    def __init__(self, n):
        super(MultiLinearRegression, self).__init__()
        self.A = torch.randn((1, n), requires_grad=True)
        self.b = torch.randn(1, requires_grad=True)
        # Says to the optimizer that the parameters A and b are parameters
        self.A = torch.nn.Parameter(self.A)
        self.b = torch.nn.Parameter(self.b)

    def forward(self, x_input):
        return self.A @ x_input + self.b
```

**Code 5-1 Class initialization**

$$forward = A * x + b \quad (5-1)$$

In the Tab 5-2 are listed the hyperparameters that will be optimized in order to obtain the best model.

**Tab 5-2 Multi linear regression hyperparameters**

Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 - 0,0001]
Batch_size	[2 – 512]

## 5.2.2 Polynomial model

As an extension of the multilinear model, a regressor with a polynomial approximation will be defined. The *pytorch* library has to be imported and the class *MultiPolynRegression* has to be created. This class is a subclass of *torch.nn.module*, it will receive as input the dimension of the input and the polynomial degree. The class contains the parameters list “*params*”, where the sets of weights and the bias are saved, as is shown in the Code 5-2. The forward pass is defined following the equation (5-2).

```

class MultiPolyRegression(nn.Module):
    def __init__(self, degree, n):
        super(MultiPolyRegression, self).__init__()
        self.params=[]
        self.degree=degree
        self.params.append(nn.Parameter(torch.randn(1, requires_grad=True)))
        for _ in range(1, degree+1):
            self.params.append(nn.Parameter(torch.randn((1, n), requires_grad=True)))
        self.params=nn.ParameterList(self.params)

    def forward(self, x_input):
        poly=self.params[0]
        for d in range(1, self.degree+1):
            poly = poly+self.params[d]@x_input**d
        return poly

```

**Code 5-2 Polynomial regressor class**

$$forward = \sum_{i=2}^{i=degree} w_i * x^i + b \quad (5-2)$$

In Tab 5-3 are listed the hyperparameters that will be optimized in order to obtain the best model.

**Tab 5-3 Polynomial regression hyperparameters**

Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 – 0,0001]
Batch size	[2-512]
Polynomial degree	[1-500]

### 5.2.3 MLP with 1 hidden layer

A MLP model, explained in chapter 2.4.3, is created with one hidden layer, as it is shown in the Code 5-3. Bircanoğlu et al. in [36] studied the performance of different activation functions in different ANN models. In the study it is found that the activation function with better performance for regression models is ReLU, which will be used for the present



project. Since ReLU is used as activation function, following the guidelines of [37], Kaiming He weight initialization [38] will be used.

```
class NeuralNetwork(nn.Module):
    def __init__(self, input_dim, hidden_dim1, output_dim):
        super(NeuralNetwork, self).__init__()
        self.layer_1 = nn.Linear(input_dim, hidden_dim1)
        nn.init.kaiming_uniform_(self.layer_1.weight, nonlinearity=
"relu")
        self.layer_2 = nn.Linear(hidden_dim1, output_dim)
        nn.init.kaiming_uniform_(self.layer_2.weight, nonlinearity=
"relu")

    def forward(self, x):
        x = torch.nn.functional.relu(self.layer_1(x))
        x = torch.nn.functional.relu(self.layer_2(x))
        return x
```

**Code 5-3 MLP with one hidden layer class**

In the Tab 5-4 are listed the hyperparameters that will be optimized in order to obtain the best model.

**Tab 5-4 MLP with one hidden layer hyperparameters**

Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 – 0,0001]
Batch size	[2-512]
Nodes in hidden layer	[1-500]

## 5.2.4 MLP with 2 hidden layers

A MLP model with two hidden layers is defined, as it is shown in the Code 5-4. Following the conclusions extracted in chapter 5.2.3, ReLU activation function and Kaiming He weight initialization will be used.

```
class NeuralNetwork(nn.Module):
    def __init__(self, input_dim, hidden_dim1, hidden_dim2, output_
dim):
        super(NeuralNetwork, self).__init__()
        self.layer_1 = nn.Linear(input_dim, hidden_dim1)
```

```

        nn.init.kaiming_uniform_(self.layer_1.weight, nonlinearity=
"relu")
        self.layer_2 = nn.Linear(hidden_dim1,hidden_dim2)
        nn.init.kaiming_uniform_(self.layer_2.weight, nonlinearity=
"relu")
        self.layer_3 = nn.Linear(hidden_dim2,output_dim)
        nn.init.kaiming_uniform_(self.layer_3.weight, nonlinearity=
"relu")

    def forward(self, x):
        x = torch.nn.functional.relu(self.layer_1(x))
        x = torch.nn.functional.relu(self.layer_2(x))
        x = torch.nn.functional.relu(self.layer_3(x))
        return x

```

**Code 5-4 MLP with two hidden layer class**

## 5.3 Model training

To train the previously defined models, an object of this model is created, see in the Code 5-5. After this, the optimizer and the error function are selected. Here, the mean squared error (MSE) as the error function is used following the guidelines of [39], the optimizer and the learning rate are set as hyperparameters. For the training of the final models, a learning rate decay is added in order to have a smoother convergence at the minima.

```

# Setting the loss function
loss = nn.MSELoss()
#Initialize the model
model = NeuralNetwork(input_dim, hidden_dim1, hidden_dim2, output_d
im)
# Initialize optimizer
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=25
0, gamma=0.5)

```

**Code 5-5 Model, error function and optimizer initialization**

The data is split into train and test using sklearn library [15], 70% of the data is used to train the model, 20% as test and 10% is used for validation, as it is shown in Code 5-6.

```

from sklearn.model_selection import train_test_split
#Split train/test
train1, test = train_test_split(result, test_size=0.2, random_state
=42, shuffle=True)

```

```
train, val = train_test_split(train1, test_size=0.125, random_state
=42, shuffle=True)
```

#### Code 5-6 Training-test data split

Once the data is split, the training data is divided into batches, as it is shown in Code 5-7. The batch size is selected as hyperparameter, following the guidelines of [40] a batch size between 2 and 512 will be considered.

```
from torch.utils.data import DataLoader, TensorDataset
#Batch size
batch=DataLoader(TensorDataset(x_train,y_train), batch_size=batch_s
, shuffle=True)
for id_batch, (x_batch, y_batch) in enumerate(batch):
```

#### Code 5-7 Training data batch split

The losses are set to 0 and the lists to save the loss value at each epoch are created. The number of epochs is incremented until the training loss converges. Initiate the training by resetting all the gradients to 0. Perform a forward pass by passing the training data and calculating the predicted value of the output. Following the guidelines of Schischka in [22], to avoid overfitting an L-2 regularization with a  $\lambda = 0,001$  will be applied. After applying regularization compute the loss, perform the backpropagation and then, update the weights. Save the training loss value. See the steps in Code 5-8.

```
# Reset losses Current loss
current_loss = 0
current_test_loss=0

train_loss=[]
test_loss=[]

# Main optimization loop
for t in range(epoch):
    batch_loss=[]
    for id_batch, (x_batch, y_batch) in enumerate(batch):
        x_batch = x_batch.to(device)
        # Set the gradients to 0.
        optimizer.zero_grad()
        pred = model(x_batch)
        l2_lambda = 0.001
        l2_norm = sum(p.pow(2.0).sum()
            for p in model.parameters())
        current_loss_no_reg=loss(pred.cpu(), y_batch.unsqueeze(-1))
```

```

        current_loss = loss(pred.cpu(), y_batch.unsqueeze(-
1)) + l2_lambda*l2_norm
        current_loss.backward()
        optimizer.step()
        scheduler.step()
        batch_loss.append(current_loss_no_reg.detach().numpy())
    train_loss.append(sum(batch_loss)/len(batch))

```

### Code 5-8 Model training

Once the training is completed, the test data is used to test if for an unknown value of the inputs the model is able to generalize. The test loss is computed and its value saved, see in Code 5-9.

```

#Computing test loss
y_pred_test = model(x_test)
current_test_loss = loss(y_pred_test.cpu(), y_test.unsqueeze(-
1))
test_loss.append(current_test_loss.detach().numpy())

```

### Code 5-9 Model testing

After the testing phase, both test loss and training loss over the epochs are plotted, in order to judge overfitting, see in Code 5-10.

```

import matplotlib.pyplot as plt
plt.plot(x_values_train,train_loss,'-')
plt.plot(x_values_test,test_loss,'-')
plt.yscale('log')
plt.grid()
plt.xlabel('epoch')
plt.ylabel('losses')
plt.legend(['Train', 'Test'])
plt.title('Train vs Valid Losses')
plt.show()

```

### Code 5-10 Plotting test and training losses

Finally, to evaluate the performance of the model, cross validation is performed, also the RMSE and the  $R^2$  score, described in chapter 2.4.6, are calculated, see in Code 5-11.

```
!pip install torchmetrics
```

```

from torchmetrics import R2Score
from torchmetrics.functional import mean_squared_error
import numpy as np
from sklearn.model_selection import KFold
### Cross Validation ###

```

```

# Configuration options
k_folds = 5
num_epochs = 1
fold_result={}

torch.manual_seed(42)

# Define the K-fold Cross Validator
kfold = KFold(n_splits=k_folds, shuffle=True)
fold=0
kf = KFold(n_splits = 5, shuffle = True, random_state = 2)

for train_index, test_index in kf.split(result):
    ...
    # Training loop
    ...
    r2score = R2Score()
metric=r2score(y_pred_val.cpu(), y_val.unsqueeze(-1))
metric2=mean_squared_error(y_pred_val.cpu(), y_val.unsqueeze(-
1), squared=False)

```

**Code 5-11 Model validation metrics**

## 5.4 Hyperparameter tuning

The hyperparameter tuning will be performed with Optuna [41], which is a python library that enables the automatic tuning of machine learning models. In Code 5-12. Optuna library is installed in windows cmd and then imported in python.

```
!pip install optuna
```

```
import optuna
```

**Code 5-12 Import Optuna library**

To perform the tuning an objective function, which will allow the comparison of the model performance, has to be defined. Since the main objective of the machine learning model is to predict the output value with the maximum precision, the test loss will be selected as the objective function. Then the search space will be defined. This consists in the hyperparameters that have to be tuned with its range of possible values. In the Code 5-13, the hyperparameter tuning of the MLP with one hidden layer model can be seen.

```

def objective(trial):
    params = {

```

```

    'learning_rate': trial.suggest_int('learning_rate', 1, 4),
    'optimizer': trial.suggest_categorical('optimizer', ["Adam", "R
MSprop", "SGD"]),
    'hidden_dim1': trial.suggest_int("hidden_dim1", 270, 290),
    'batch_s':trial.suggest_int('batch_s',1,9)
}

model = NeuralNetwork(input_dim, params, output_dim)

accuracy = train_and_evaluate(model,params)

return accuracy

```

### Code 5-13 Hyperparameter tuning on MLP with one hidden layer

As it can be seen in Code 5-13 and due to restrictions in the definition of the hyperparameters, the following changes have to be done in the optimizer initialization, the batch size definition and the layer definition of the model, see in Code 5-14.

```

# Setting the optimizer
optimizer = getattr(optim, param['optimizer'])(model.parameters()
, lr = 1*10**(-param['learning_rate']))
#Batch size
batch=DataLoader(TensorDataset(x_train,y_train), batch_size=2**pa
ram['batch_s'], shuffle=True)

class NeuralNetwork(nn.Module):
    def __init__(self, input_dim, params, output_dim):
        super(NeuralNetwork, self).__init__()
        self.layer_1 = nn.Linear(input_dim, params['hidden_dim1'])
        nn.init.kaiming_uniform_(self.layer_1.weight, nonlinearity=
"relu")
        self.layer_2 = nn.Linear(params['hidden_dim1'],output_dim)
        nn.init.kaiming_uniform_(self.layer_2.weight, nonlinearity=
"relu")

```

### Code 5-14 Changes in the optimizer, batch size and model layer definition

After defining the hyperparameters to tune and its range of values, the hyperparameter tuning session can start. An Optuna hyperparameter tuning session is called study. A study session is created by calling *create\_study* method. The direction argument indicates weather the tuning is performed to maximize or minimize the objective function, as it is shown in Code 5-15, minimization of the objective function will be selected. The sampler argument indicates the optimization strategy, *TPESampler()* will be used. The optimization strategy starts off like random sampler, but it suggests the set of

hyperparameter values for the next trial based on the set with better objective values from past trials. The *optimize* method is called with the objective function and the number of trials as arguments. After each trial the hyperparameter selection, the value of the objective function and the best trial will be displayed.

```
study = optuna.create_study(direction="minimize", sampler=optuna.samplers.TPESampler())
study.optimize(objective, n_trials=150)

best_trial = study.best_trial

for key, value in best_trial.params.items():
    print("{}: {}".format(key, value))
```

**Code 5-15 Optuna study creation**





## 6 Results

In this chapter the results of the tuned models are presented with its hyperparameter choice. The hyperparameter tuning has been performed with a smaller representative dataset of 1000 samples in order to speed up the tuning process. Then the best models are selected and trained with the full dataset of 25000 samples.

### 6.1 Multilinear regression

The hyperparameters are tuned using Optuna, in the Tab 6-1 it can be seen the range of values that are studied.

**Tab 6-1 Hyperparameter value range for Optuna tuning; Linear regression**

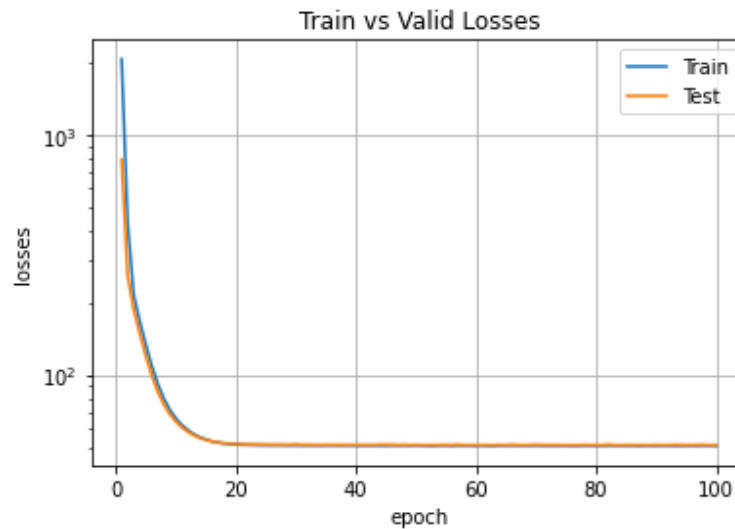
Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 – 0,0001]
Batch size	[2-512]

Once the study is performed, the best trial has the following hyperparameters:

**Tab 6-2 Hyperparameter choice for the linear regression**

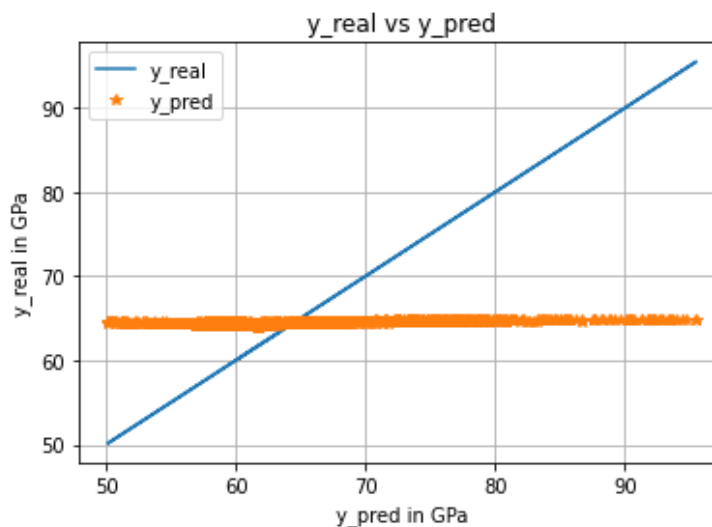
Hyperparameter	
Optimizer	Adam
Learning rate	0,1
Batch size	128

The model with the before mentioned hyperparameters is trained with 100 epochs. As it can be seen in Fig 6-1, the test/training curve shows that the model is training correctly and there is no overfitting.



**Fig 6-1 Train and test loss over the epochs; linear regression**

In the following graphs a comparison between the predicted values and real output values is shown for the different input values. As it can be seen in the Fig 6-2, the model is not able to predict the output.



**Fig 6-2 Real vs predicted effective stiffness over the different parameters; linear regression**

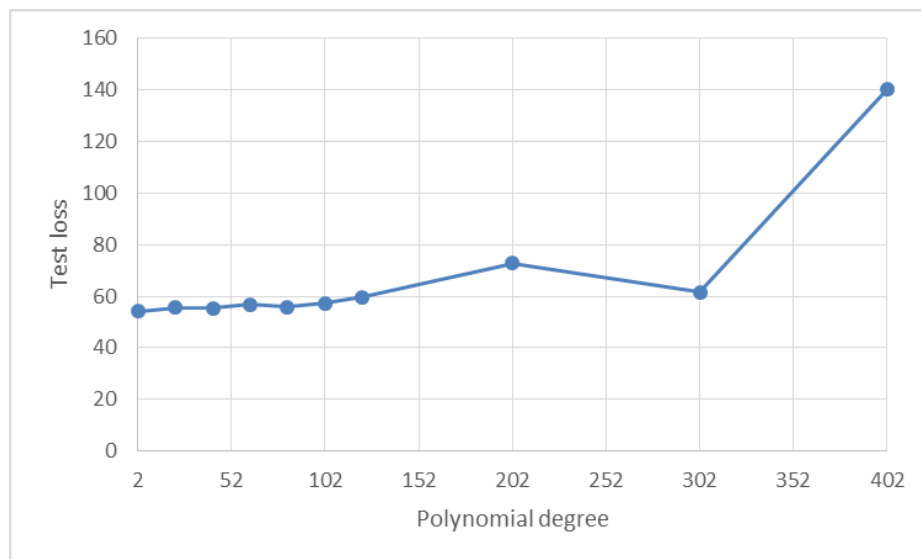
To compare this model with the other ones the metrics listed in the Tab 6-3 are obtained:

**Tab 6-3 Linear regression metrics**

Metrics	
R <sup>2</sup> score	0,0345
RMSE	6,9232
K-fold mean test_loss	50,6413
K-fold standard deviation	1,6835

## 6.2 Polynomial regression

The hyperparameter choice of batch size, optimizer and learning rate obtained in the multilinear regression will be used in this section to generate a first approximation of the polynomial degree. The test loss is monitored for degree steps of 100, as it can be seen in the Fig 6-3, the interval between [2,102] has the lower values of loss. Then this interval is studied with degrees steps of 20, obtaining the lower loss between degree 82 and 102. Finally an Optuna study with the hyperparameters listed in the Tab 6-4.

**Fig 6-3 Test loss over polynomial degree**

**Tab 6-4 Hyperparameter value range for Optuna tuning; Polynomial regression**

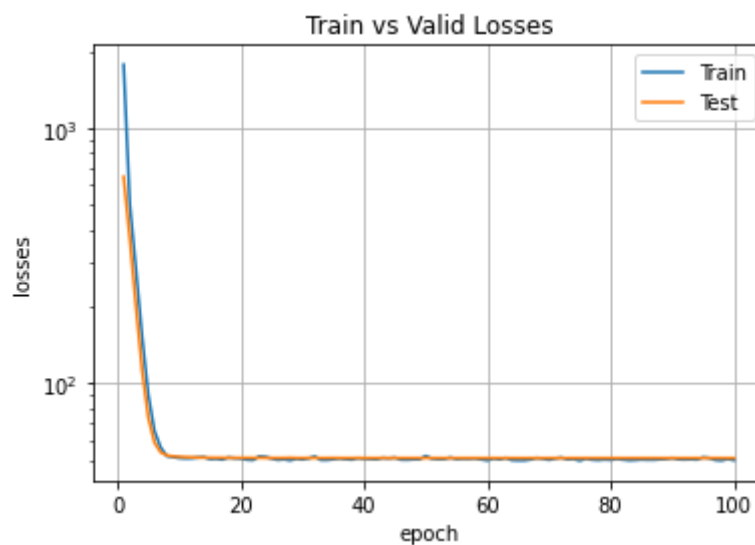
Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 – 0,0001]
Batch size	[2-512]
Polynomial degree	[82-102]

Once the study is performed, the best trial has the following hyperparameters:

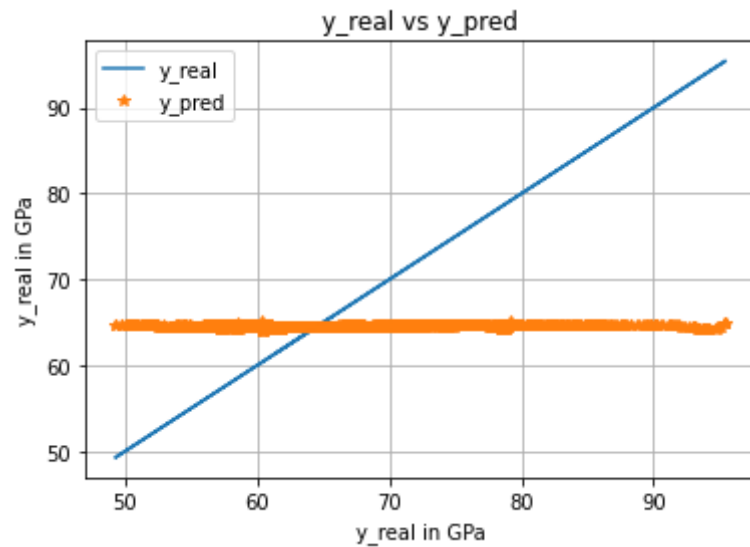
**Tab 6-5 Hyperparameter choice for the polynomial regression**

Hyperparameter	
Optimizer	Adam
Learning rate	0,01
Batch size	512
Polynomial degree	89

The tuned model is trained in 100 epochs. As it can be seen in Fig 6-4, the test/training curve shows that there is no overfitting

**Fig 6-4 Train and test loss over the epochs; polynomial regression**

In the following graphs a comparison between the predicted values and real output values is shown for the different input values. As it can be seen in the Fig 6-5 the model is not able to predict the output.



**Fig 6-5 Real vs predicted effective stiffness over the different parameters; polynomial regression**

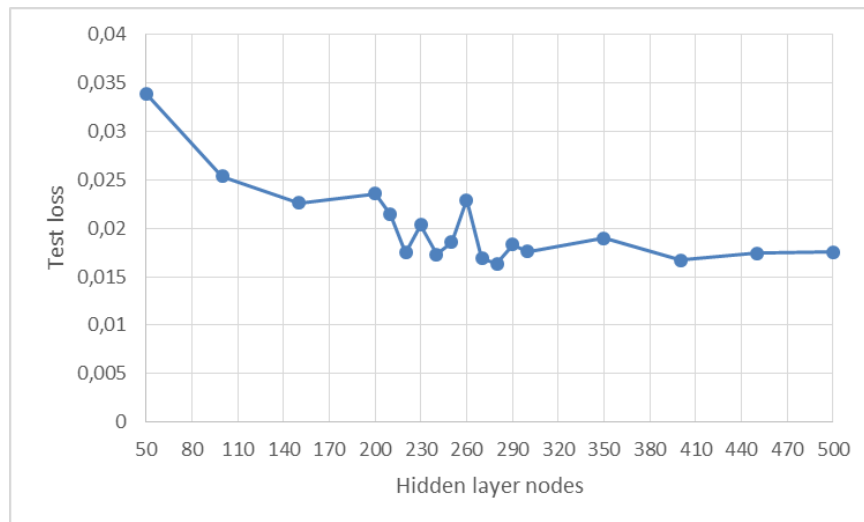
To compare this model with the other ones the metrics listed in the Tab 6-6 are obtained:

**Tab 6-6 Polynomial regression metrics**

Metrics	
R <sup>2</sup> score	0.0103
RMSE	7.1008
K-fold mean test_loss	50,6292
K-fold standard deviation	1,6638

## 6.3 MLP with one hidden layer

The hyperparameter choice of batch size, optimizer and learning rate obtained in the polynomial regression will be used in this section to generate a first approximation of the hidden layer dimension. The test loss is monitored for hidden layer nodes steps of 50, as it can be seen in the Fig 6-6, the interval between [200,300] has the lower values of loss. Then this interval is studied with degrees steps of 10, obtaining the lower loss between 270 and 290 nodes in the hidden layer.



**Fig 6-6 Test loss over hidden layer dimension; MLP model with 2 hidden layer**

Finally an Optuna study with the hyperparameters listed in the Tab 6-7.

**Tab 6-7 Hyperparameter value range for Optuna tuning; MLP model with 1 hidden layer**

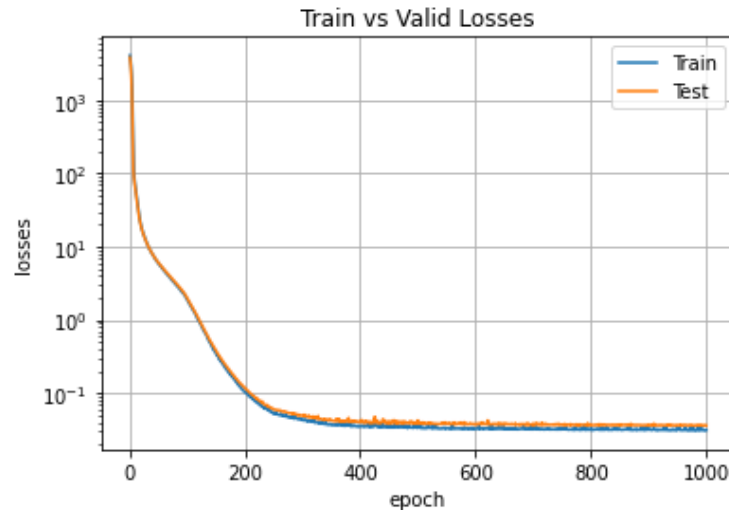
Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 – 0,0001]
Batch size	[2-512]
Nodes in hidden layer	[270-290]

Once the study is finished the following hyperparameters are obtained

**Tab 6-8 Hyperparameter choice for the MLP model with 1 hidden layer**

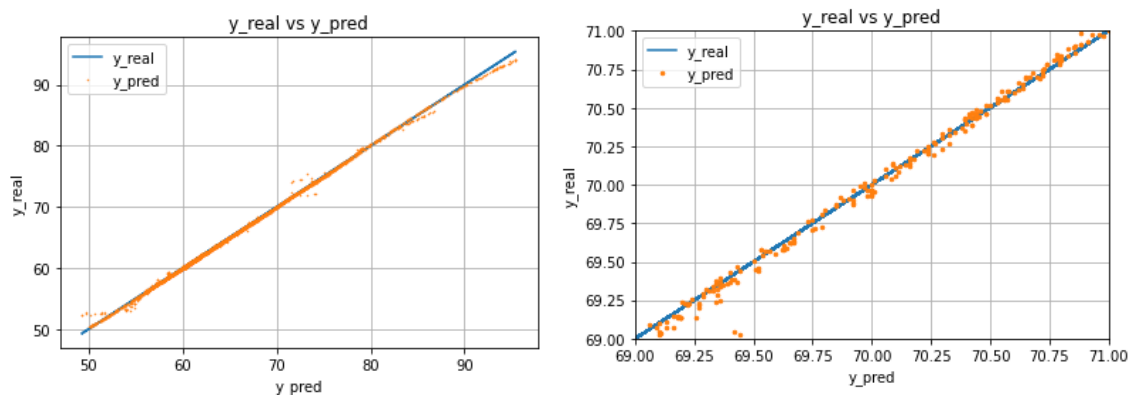
Hyperparameter	
Optimizer	Adam
Learning rate	0,001
Batch size	512
Hidden layer dimension	281

The model with the mentioned hyperparameter choice is trained with 1000 epochs. As it can be seen in Fig 6-7, the test/training curve the model is training correctly and there is no overfitting.



**Fig 6-7 Train and test loss over the epochs; MLP with 1 hidden layers**

Using the described model, the longitudinal effective stiffness of the defective laminates could be calculated with high accuracy. Fig 6-8 shows the correct and the predicted modulus of elasticity of all test data. It can be seen that especially in the range of a middle-lower effective stiffness, the predictions of the MLP with one hidden layer had a high accuracy. In the higher and lower range of values the model loses accuracy.



**Fig 6-8 Real vs predicted effective stiffness over the different parameters; MLP with one hidden layer**

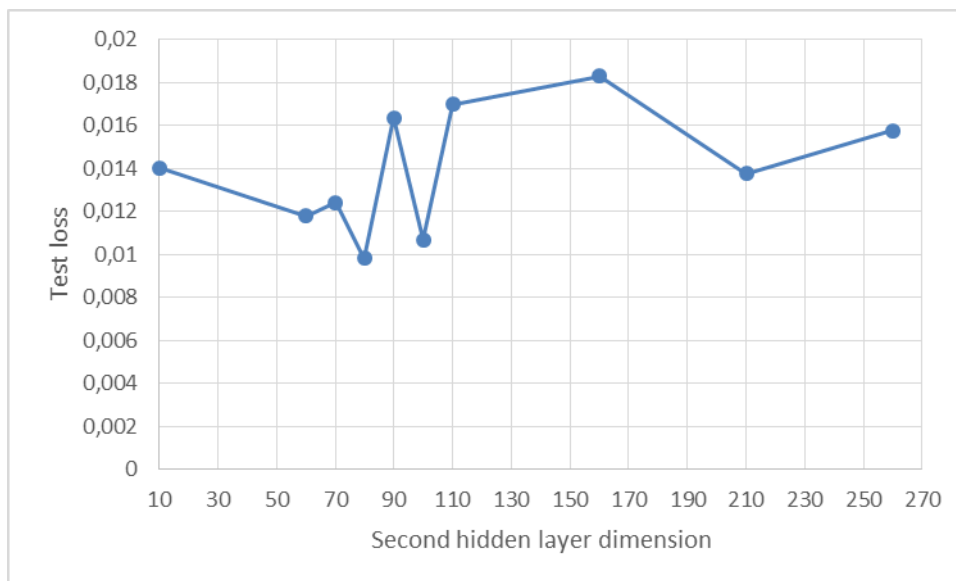
To compare this model with the other ones the metrics listed in the Fig 6-10 are obtained:

**Tab 6-9 MLP with one hidden layer metrics**

Metrics	
R <sup>2</sup> score	0,9995
RMSE	0,1564
K-fold mean test_loss	0,0028
K-fold standard deviation	0,0043

## 6.4 MLP with two hidden layers

The hyperparameter choice of dimension of hidden layer one, batch size, optimizer and learning rate obtained in the MLP with one hidden layer will be used in this section to generate a first approximation of the hidden layer two dimension. The test loss is monitored for hidden layer nodes steps of 50, as it can be seen in the Fig 6-9, the interval between [60,110] has the lower values of loss. Then this interval is studied with degrees steps of 10, obtaining the lower loss between 70 and 90 nodes in the hidden layer.

**Fig 6-9 Test loss over hidden layer dimension; MLP model with 2 hidden layer**

Finally an Optuna study with the hyperparameters listed in the Tab 6-10.



**Tab 6-10 Hyperparameter value range for Optuna tuning MLP model with 2 hidden layer**

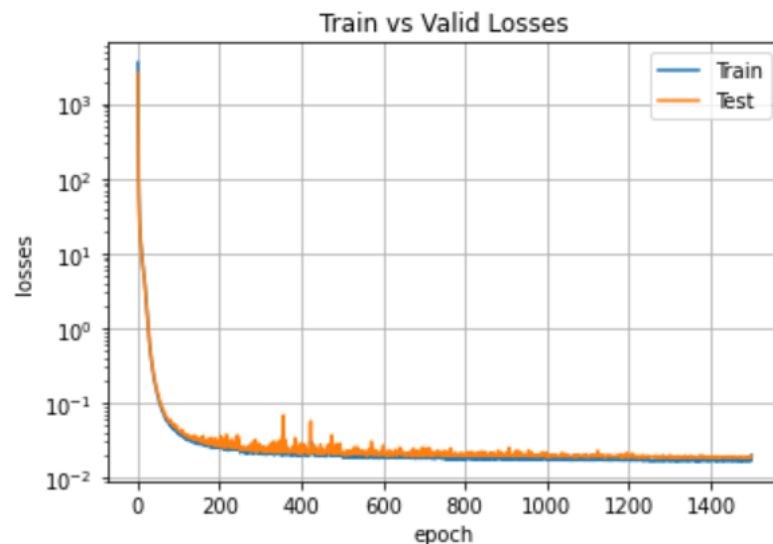
Hyperparameter	
Optimizer	Stochastic Gradient Descent, RMSProp, Adam
Learning rate	[0,1 – 0,0001]
Batch size	[2-512]
Nodes in hidden layer one	281
Nodes in hidden layer two	[70-90]

Once the study is finished the following hyperparameters are obtained:

**Tab 6-11 Hyperparameter choice for the MLP model with 1 hidden layer**

Hyperparameter	
Optimizer	Adam
Learning rate	0,001
Batch size	512
First hidden layer dimension	281
First hidden layer dimension	76

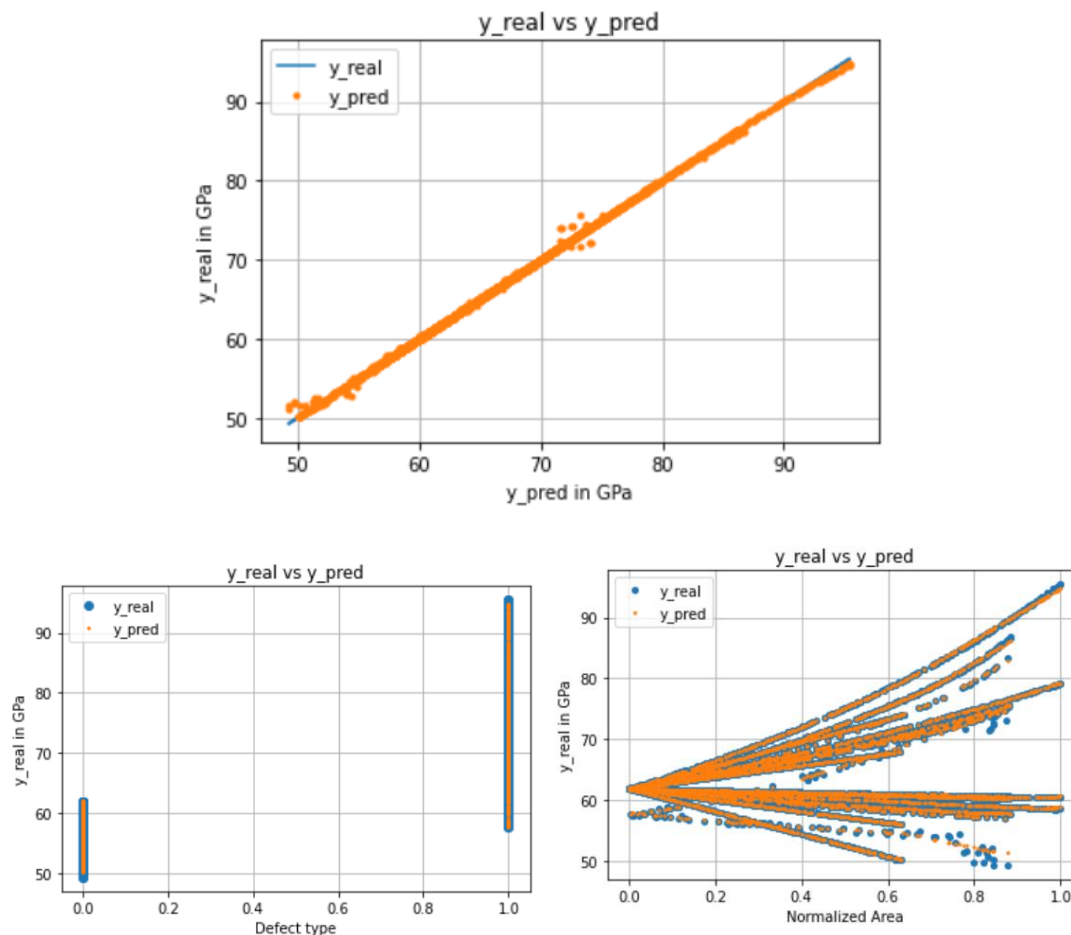
The tuned model is trained with 1500 epochs. As it can be seen in Fig 6-10, the test/training curve shows that there is no overfitting.



**Fig 6-10 Train and test loss over the epochs; MLP with 2 hidden layers**

Using the described in model, the longitudinal effective stiffness of the defective laminates could be calculated with high accuracy. Fig 6-11 shows the correct and the predicted modulus of elasticity of all test data. It can be seen that especially in the range

of a lower effective stiffness, the predictions of the MLP with two hidden layers had a high accuracy. In the range of 72-73 GPa the model a higher deviation on the predictions can be seen. Also for the higher and lower values of effective stiffness the model loses accuracy.



**Fig 6-11 Real vs predicted effective stiffness over the different parameters; MLP with two hidden layers**

To compare this model with the other ones the metrics listed in the Tab 6-12 are obtained:

**Tab 6-12 MLP with two hidden layers metrics**

Metrics	
R <sup>2</sup> score	0,9998
RMSE	0,1032
K-fold mean test_loss	0,01773
K-fold standard deviation	0,0032

## 7 Economic study

This economical study provides a detailed analysis of the costs associated with an engineering project. It includes direct costs such as labor, equipment and software, as well as indirect costs such as overhead expenses. Additionally, it takes into account an entrepreneurial margin to ensure profitability. The study serves as a valuable tool for project budgeting and decision making, providing insight into the financial feasibility of the project.

Direct costs: Includes the working hours for the development of the project, its supervision, the software licenses used and informatic equipment:

- Student salary: 675 worked hours at 36 €/h = 23.300 €
- Supervisor salary: 100 worked hours at 60 €/h = 6.000 €
- ABAQUS Software license = 9.500 €
- Computer = 200 €

Total Direct costs: 39.000 €

Indirect costs:

- 20% of indirect costs = 7.800 €

Total Indirect costs: 7.800 €

Total Direct costs + Indirect costs: 46,800 €

Entrepreneurial margin:

- 10% entrepreneurial margin = 4.680 €

Total project cost: 51.480 €

These calculations are based on the data provided and the assumption that the computer and software are necessary for the execution of the project, and that the amortization of the computer is done in five years and used 6 months. Additionally, the indirect costs and the entrepreneurial margin are estimates based on typical industry standards.

## 8 Environmental study

Given the character of the project, focused on the field of machine learning and FE simulation research, this section does not apply beyond the basic criteria of rational use of energy and the use of recyclable material during the preparation of the simulations and drafting of the report final.

## 9 Social and gender equality study

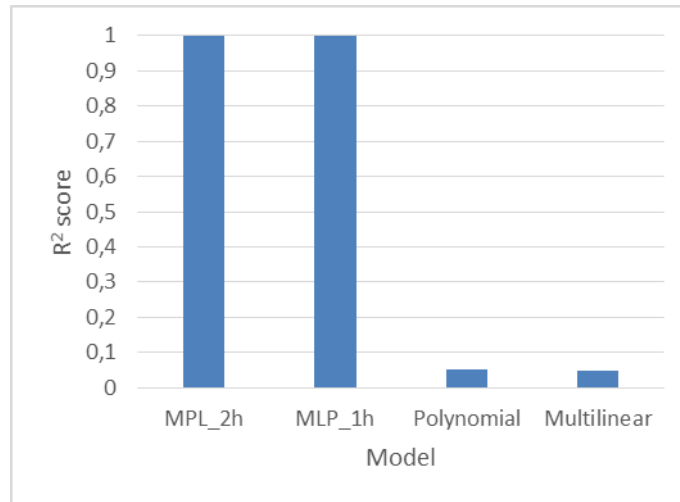
The present project based on finite element simulation and machine learning is based on mathematical models and physical laws, not on subjective opinions or personal biases. The simulation results depend on the input data and assumptions, not on the gender of the engineer running the simulation. Therefore, there is no reason for gender bias in a finite element simulation and machine leaning project.

## 10 Conclusions and outlook

### 10.1 Conclusions

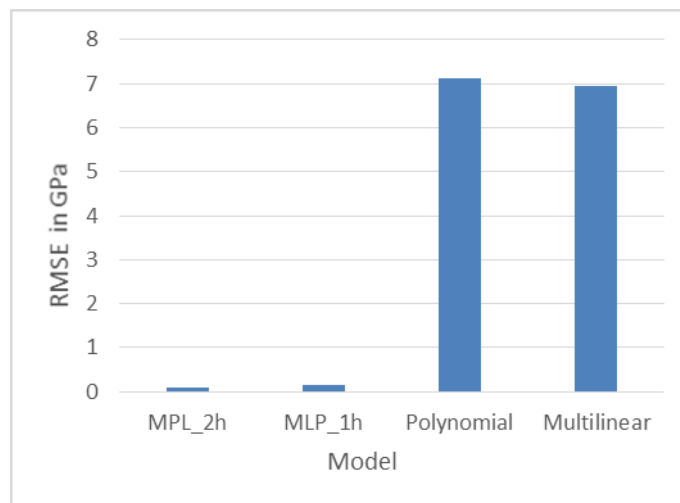
In the present project, a parametrized ABAQUS FE model of a composite laminate on a coupon level has been successfully implemented. The FE model is created using 3-Shells, where just one of them contains the defective ply and the other two the rest of the laminate. This approach is validated by comparing the simulation results with the experimental ones. By applying multiple parameter variations at the model, a data set of 25000 samples is generated. From the vast quantity of regression ML models, four models are selected and its hyperparameters tuned to obtain the best performance of each model.

Once all the models are trained, they can be compared to each other with the metrics obtained. In the Fig 10-1, the  $R^2$  score of the different models is shown. As it can be seen, the multilinear and the polynomial model have a value near 0, this means that these two models are not able to capture the relationship between the inputs and the output value and therefore are not able to generalize. On the other side the MLP models have  $R^2$  score near 1, which means that these models are able to generalize well for unseen input values. Regarding the RMSE values of the models, the multilinear and the polynomial models have a high value, since they are not able to predict the output.



**Fig 10-1 R<sup>2</sup> score of the different models**

The MLP models have a lower RMSE value than the polynomial and multilinear models as it can be seen in the Fig 10-2. This was the expected result since these latter two are not able to predict the output for the given inputs.



**Fig 10-2 RMSE for the different studied models**

Both MLP models have similar results in the R<sup>2</sup> score and RMSE. The model with two hidden layers performs slightly better in terms of these metrics, but also it has a higher training time due to the higher number of parameters that it has. Since the present project is aiming for accuracy before computational time the MLP model with two hidden layers is chosen as the best model that can predict the effects on the effective stiffness of the defects that occur during the manufacturing process in the AFP. Also comparing the robustness of the models, the cross validation results show that the MLP model with two hidden layers is less sensible at the change of the data at the different folds. It is also

important to remark that these high accuracy values are only achieved because the data is generated with a simulation software. With the use of experimental data, the maximum achievable accuracy will be lower.

## 10.2 Outlook

Using the ML model, the effect of a defect in a composite laminate at the effective stiffness can be calculated given the type of defect found, the ply where the defect is found, the area of the defect and if the defect comprises just one or two plies.

As it is explained the model is able to interpret the relationships between the inputs and the output and is able to give high accurate predictions of it. However, the data used to train the ML model is extracted from a simple FE model that does not take into account some of the minor effects of these defects, also some of the defect cases were excluded of the study because of discordant results with the experimental results. For this reason, more investigation in the FE models that can calculate the effect of the defects with higher accuracy has to be done.

In addition, by creating FE models that calculate other mechanical properties, the ML model can also be useful to predict these mechanical properties of the composite laminates and how they are affected by the defects.





## 11 References

- [1] Li X. Modelling the effect of gaps and overlaps in automated fibre placement (AFP)-manufactured laminates. *Sci Eng Compos Mater* 2015;22:115–29.
- [2] Rousseau G. Automated Fiber Placement Path Planning: A state-of-the-art review. *Computer-Aided Design & Applications* 2019;16(2):172–203.
- [3] A. Crosky. Fibre placement processes for composites manufacture. *Advances in Composites Manufacturing and Process Design* 2015:79–92.
- [4] Rossi D de. Effect of Half Gap/Overlap Defects on the Strength of Composite Structures Fabricated with Automated Fiber Placement Methods. *Structures and Composite Materials Laboratory Department of Mechanical Engineering McGill University* 2019:1–16.
- [5] J.N. Reddy. *Mechanics of Laminated Composite Plates and Shells. Theory and Analysis* 1997:109–61.
- [6] Mittelstedt C. *Flächentragwerke*. Berlin, Heidelberg: Springer Berlin Heidelberg; 2022.
- [7] Juan Tomás Celigüeta Lizarza. *Metodo de los elementos finitos para analisis estructural* 2000:1–8.
- [8] Eduardo Frías Valero. Aportaciones al estudio de las maquinas electricas de flujo axial mediante la aplicacion del metodo de los elementos finitos 2004:111–44.
- [9] Alfonso Cubillos. *Introducción al método de los elementos finitos* 2000:1–15.
- [10] Dr. Matthias Goelke. *Practical Aspects of Finite Element Simulation A Study Guide* 2019:21–100.
- [11] graspengineering. Different Types of FEA Elements / How to Decide the Element Type. [November 19, 2022]; Available from:

- <https://www.graspengineering.com/different-types-of-fea-elements-how-to-decide-the-element-type/>.
- [12] Shai Shalev-Shwartz, Shai Ben-David. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press 2014:1–20.
- [13] Rastogi K. Machine Learning approach for Enterprise Data with a focus on SAP. Otto-von-Guericke-University Magdeburg 2018:11–23.
- [14] Xin Yan, Xiao Gang Su. Linear Regression Analysis Theory and Computing. World Scientific 2009:41–115.
- [15] Pedregosa et al. Scikit-learn: Machine Learning in Python. JMLR 12 2011:2825–30.
- [16] Julija Zavadlav. Physics-Informed Machine Learning. TUM 2021.
- [17] Amanoul SV, Abdulazeez AM, Zeebare DQ, Ahmed FYH. Intrusion Detection Systems Based on Machine Learning Algorithms. In: 2021 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS): 26th June 2021, Shah Alam, Malaysia proceedings virtual conference, Shah Alam, Malaysia, 6/26/2021 - 6/26/2021.
- [18] Eshragh F, Pooyandeh M, Marceau DJ. Automated negotiation in environmental resource management: Review and assessment. Journal of Environmental Management 2015;162:148–57.
- [19] Murat H. Salzi. A brief review of feed-forward neural networks. Commun. Fac. Sci. Univ. Ank. 2006:11–7.
- [20] ResearchGate. Fig. 2. The comparison between Recurrent Neural Network (RNN) and. [October 20, 2022]; Available from:  
[https://www.researchgate.net/figure/The-comparison-between-Recurrent-Neural-Network-RNN-and-Feed-Forward-Neural-Network\\_fig1\\_338672883](https://www.researchgate.net/figure/The-comparison-between-Recurrent-Neural-Network-RNN-and-Feed-Forward-Neural-Network_fig1_338672883).
- [21] Stephan Matzka. Künstliche Intelligenz in den Ingenieurwissenschaften: Maschinelles Lernen verstehen und bewerten. Springer Nature(2021):99–166.
- [22] Nicolas Schischka. Artificial Intelligence in the Material Development of Composites 2022:3–20.

- [23] Bravo M. Desarrollo de un sensor virtual de flujo usando Redes Neuronales Artificiales para Bombas de Cavidades Progresivas en el campo San Diego de Cabrutica. 3er congreso suramericano de petróleo y gas 2015 2015.
- [24] Polyzotis N. Data Management Challenges in Production Machine Learning. SIGMOD'17 2017:1723–6.
- [25] Wikipedia. Backpropagation. [October 20, 2022]; Available from: <https://en.wikipedia.org/w/index.php?title=Backpropagation&oldid=1115388493>.
- [26] Draper NR, Smith H. Applied regression analysis. 3rd ed. New York: John Wiley & Sons; 1998.
- [27] Darvishi M. Implementing Machine Learning Algorithms on Finite Element Analyses Data Sets for Selecting Proper Cellular Structure. International Journal of Applied Mechanics 2021;13(6):1–15.
- [28] Zhenchao Q. Prediction of mechanical properties of carbon fiber based on cross-scale FEM and machine learning. Composite Structures 2019;212:199–206.
- [29] Sang Ye. Deep neural network method for predicting the mechanical properties of composites. Appl. Phys. Lett. 115 2019:1–3.
- [30] Luis Haug. Laser Scanning based Quantification of the Effects of Defects during Automated Fiber Placement 2021:3–86.
- [31] UNE-EN ISO 527-1:2020 Plásticos. Determinación de las propiedades. [December 04, 2022]; Available from: <https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0064896>.
- [32] Gan KW, Hallett SR, Wisnom MR. Measurement and modelling of interlaminar shear strength enhancement under moderate through-thickness compression. Composites Part A: Applied Science and Manufacturing 2013;49:18–25.
- [33] Brownlee J. Ordinal and One-Hot Encodings for Categorical Data. Machine Learning Mastery 2020, 11 June 2020; Available from: <https://machinelearningmastery.com/one-hot-encoding-for-categorical-data/>. [November 29, 2022].
- [34] Proceedings of the 9th Python in Science Conference: SciPy; 2010.

- [35] Pytorch: An imperative style, high-performance deep learning library; 2019.
- [36] Bircanoglu C, Arica N. A comparison of activation functions in artificial neural networks. In: SIU 2018: 26. IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı = 26th IEEE Signal Processing and Communications Applications Conference 2-5 Mayıs, Izmir, 5/2/2018 - 5/5/2018.
- [37] Jason Brownlee. Weight Initialization for Deep Learning Neural Networks. Deep learning 2021.
- [38] He K, Zhang X, Ren S, Sun J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: 2015 IEEE International Conference on Computer Vision (ICCV).
- [39] Yang N, Zheng Z, Wang T. Model Loss and Distribution Analysis of Regression Problems in Machine Learning. In: Unknown, editor. Proceedings of the 2019 11th International Conference on Machine Learning and Computing. New York, NY, United States: Association for Computing Machinery; 2019, p. 1–5.
- [40] Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, Tang PTP. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima; 2016.
- [41] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A Next-generation Hyperparameter Optimization Framework; 2019.

## List of Figures

Fig 2-1 Schematic representation of the half plane distances [6] .....	4
Fig 2-2: Automated fiber placement head. [1].....	6
Fig 2-3: Schematic representation of gaps and overlaps. [1].....	7
Fig 2-4: 1-D, 2-D and 3-D element representation [11] .....	9
Fig 2-5: Graphical representation of Least squares method. [16].....	11
Fig 2-6 Random forest regression architecture [17] .....	12
Fig 2-7: Feed-forward and Recurrent neural networks representation. Source [20] .....	12
Fig 2-8: Neuron representation [23] .....	13
Fig 2-9: Artificial neural network schematic representation .....	14
Fig 2-10: Examples of over- and underfitting. [16].....	18
Fig 2-11: Variation of the test and training error depending on the model complexity. [16].....	18
Fig 2-12: K-fold cross validation schematic [16] .....	19
Fig 3-1: Mesh adaptation to the change of thickness. [1] .....	20
Fig 3-2: Schematic representation of a gap. [4].....	22
Fig 3-3: Accuracy of the trained models [27] .....	24
Fig 4-1 Coupon geometry .....	28
Fig 4-2 Ply stacking sequence .....	28
Fig 4-3 Defect characterization for the 45 degree ply .....	30
Fig 4-4 Defect characterization for the 0 degree ply .....	31
Fig 4-5 Three-shell assembly model.....	32
Fig 4-6 Two shell assembly model with double defect .....	33
Fig 4-7 Boundary conditions and load.....	33
Fig 4-9 Convergence study 1-shell model in pristine condition.....	34
Fig 4-10 Deviation of the output: 1-shell model in pristine condition .....	34
Fig 4-11 Convergence study 3-shell model pristine condition .....	35
Fig 4-12 Global mesh convergence study of 90 degree 5 mm defect in ply 8 .....	36
Fig 4-13 Deviation of the of the output, 5 mm defect in ply 8 .....	36
Fig 4-14 Finer mesh on the defective region .....	36
Fig 4-15 Finer mesh convergence of 90 degree 5 mm defect in ply 8 .....	37
Fig 4-16 Deviation of the of the output, 5 mm defect in ply 8 .....	37

Fig 4-17 Mesh convergence study of 5 mm defect in ply 4.....	38
Fig 4-18 Deviation of the of the output, 5 mm defect in ply 4.....	38
Fig 4-19 Quadrangular mesh deformation .....	38
Fig 4-20 Global mesh convergence study of 5 mm defect in ply 2.....	39
Fig 4-21 Deviation of the output of 5 mm defect in ply 2 .....	39
Fig 4-22 Mesh convergence study of 2 mm defect in ply 6.....	40
Fig 4-23 Deviation of the output, 2 mm defect in ply 6.....	40
Fig 4-24 Comparison between calculations methods.....	42
Fig 4-25 Parameters of the FE model.....	43
Fig 4-26 Effective stiffness for gap and overlap defects: 10 mm defect in ply 8 .....	44
Fig 4-27 Effective stiffness for single and double defects: 10 mm defect in ply 8.....	45
Fig 4-28 Effective stiffness of a 125,0 mm <sup>2</sup> gap at different plies.....	46
Fig 4-29 Effective stiffness for increasing defect area in ply 8 .....	46
Fig 4-30 Effective stiffness for variation of the defect position: 10 mm defect in ply 8 47	
Fig 6-1 Train and test loss over the epochs; linear regression .....	62
Fig 6-2 Real vs predicted effective stiffness over the different parameters; linear regression .....	62
Fig 6-3 Train and test loss over the epochs; polynomial regression .....	64
Fig 6-4 Real vs predicted effective stiffness over the different parameters; polynomial regression .....	65
Fig 6-5 Test loss over hidden layer dimension; MLP model with 2 hidden layer .....	66
Fig 6-6 Train and test loss over the epochs; MLP with 1 hidden layers .....	67
Fig 6-7 Real vs predicted effective stiffness over the different parameters; MLP with one hidden layer .....	67
Fig 6-8 Test loss over hidden layer dimension; MLP model with 2 hidden layer .....	68
Fig 6-9 Train and test loss over the epochs; MLP with 2 hidden layers .....	69
Fig 6-10 Real vs predicted effective stiffness over the different parameters; MLP with two hidden layers .....	70
Fig 7-1 R <sup>2</sup> score of the different models .....	74
Fig 7-2 RMSE for the different studied models .....	74

## List of tables

Tab 3-1: Difference between experimental and simulation results. [4].....	22
Tab 3-2: Input / Output parameter definition [28] .....	24
Tab 4-1 IM7/8552 prepreg ply properties [32] .....	29
Tab 4-2 Resin material properties [32] .....	29
Tab 4-3 Overlap material properties .....	30
Tab 4-4 FE model parameter overview .....	47
Tab 5-1 Pre-processed input data sample .....	50
Tab 5-2 Multi linear regression hyperparameters .....	51
Tab 5-3 Polynomial regression hyperparameters .....	52
Tab 5-4 MLP with one hidden layer hyperparameters .....	53
Tab 6-1 Hyperparameter value range for Oputna tuning; Linear regression.....	61
Tab 6-2 Hyperparameter choice for the linear regression .....	61
Tab 6-3 Linear regression metrics .....	63
Tab 6-4 Hyperparameter value range for Oputna tuning; Polynomial regression.....	64
Tab 6-5 Hyperparameter choice for the polynomial regression .....	64
Tab 6-6 Polynomial regression metrics .....	65
Tab 6-7 Hyperparameter value range for Oputna tuning; MLP model with 1 hidden layer .....	66
Tab 6-8 Hyperparameter choice for the MLP model with 1 hidden layer.....	66
Tab 6-9 MLP with one hidden layer metrics .....	68
Tab 6-10 Hyperparameter value range for Oputna tuning MLP model with 2 hidden layer .....	69
Tab 6-11 Hyperparameter choice for the MLP model with 1 hidden layer.....	69
Tab 6-12 MLP with two hidden layers metrics .....	70

# Appendix

## A USB-Data

Die Daten-CD enthält die folgenden Ordner:

- Schriftliche Ausarbeitung (obligatorisch, Arbeit als Word und PDF, Unterordner mit allen Abbildungen der Arbeit (Nummerierung der Bilder wie in der Arbeit z.B. Abb. 1-2), Citavi Ordner mit PDF-Dateien aller Literaturquellen)
- CAD Modelle (falls zutreffend)
- FE Simulationen (falls zutreffend)
- ML models (Python code)