POLITECNICO
MILANO 1863

SCHOOL OF INDUSTRIAL AND INFORMATION ENGINEERING

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

DEVELOPMENT OF A STRAIN FIELD DATABASE FOR A HELICOPTER FUSELAGE PANEL UNDER DOUBLE CRACK DAMAGE FOR MACHINE LEARNING-BASED APPLICATIONS

M.Sc. Thesis by:
DIEGO SANCHEZ SANDOVAL
925459

Supervisor:
Francesco Cadini

Co-Supervisor
Luca Lomazzi

Academic Year 2020/2021

ABSTRACT

The analysis of multiple-site damage in the tolerance design of aerospace systems is crucial for the development of accurate and reliable structural health monitoring methods. Currently, with the aid of artificial neural networks, various successful methods have been developed for the diagnosis and prognosis of single crack damage in aeronautical panels. However, there is limited research on the creation of diagnosis algorithms that allow detecting, localizating, and quantifying multiple cracks on such components and that analyze their effect on the remaining life of the panels.

In this thesis, a strain field database is built based on finite element simulations of a helicopter panel under double crack damage. Moreover, an automatization function taken from previous work is optimized to develop a database with different damage scenarios, which includes 5346 strain samples. The database is further employed as a training dataset to train, validate and test an artificial neural network to perform basic tasks of damage detection and localization.

To all the people who supported me in this process and who were always with me until the culmination of this stage of my life, infinite thanks.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 8
1.1. Overview 8
1.2. Structural Health Monitoring 9
1.3. Artificial Neural Networks 13
CHAPTER 2: STATE OF THE ART 16
2.1. Artificial Neural Networks for Structural Health Monitoring 16
2.2. Structural Health Monitoring on Helicopter Fuselage Panels 21
2.3. Automatization of Double Crack Damage Simulations for Helicopter Panels 34
CHAPTER 3: DEVELOPMENT OF THE DATABASE 37
3.1. Limitations of the Current FEA Model 37
3.2. Development of the Algorithm 42
3.2.1 Basic Structure of the Algorithm 42
3.2.2 Panel Symmetries 47
3.2.3 Crack Overlap 56
3.3. Critical Distance Between Cracks 59
3.4. Transition to ABAQUS 72
3.4.1 Simulations Post-processing 73
CHAPTER 4: TRAINING OF THE NEURAL NETWORK 75
4.1. Training Stage 75
4.2. Neural Network Testing 79
CHAPTER 5: CONCLUSIONS 84
CHAPTER 6: BIBLIOGRAPHY 86
CHAPTER 7: Appendices 88
7.1. Appendix A: Code for construction of the database. 88
7.2. Appendix B: Code for ABAQUS automatic post-processing. 91

List of Figures

Figure 1. SHM flowchart. [1] 12
Figure 2. Human brain neuron vs artificial neuron. [9] 14
Figure 3. Human neural network vs artificial neural network structure. [9] 14
Figure 4. Steel beam experimental setup. [2] 19
Figure 5. Decentralized autonomous sensor fault detection based on analytical redundancy scheme. [10] 21
Figure 6. Helicopter tail FE model and sub-models. [13] 24
Figure 7. Typical load specturm of a helicopter in operation. [12] 25
Figure 8. a) Sensor locations and interest area, b) Bay crack FE model, c) Stringer failure FE model. [15] 27
Figure 9. Experimental setup for fatigue damage testing. [16] 28
Figure 10. FE model of the bay crack simulation. [16] 29
Figure 11. Comparison of sensor characteristics. [14] 33
Figure 12. Experimental tension test setup for helicopter panels. [20] 35
Figure 13. Creation process of a linear partition and circular partitions. [20] 36
Figure 14. CAD model of the skin with partitions. 38
Figure 15. Symmetric simulation of two cracks of 20 mm under the stringer. 39
Figure 16. 35 mm crack simulation under the stringer. 40
Figure 17. 130 mm central crack simulation results. 41
Figure 18. 10 mm cracks horizontal overlap. 42
Figure 19. Mesh structure for the panel FE model. 44
Figure 20. FBG sensor location and simulation area. [21] 44
Figure 21. Crack's path according to the nested for-loop. 45
Figure 22. Panel's symmetry with respect to the vertical axis. 47
Figure 23. Single crack symmetric simulation example. 48
Figure 24. Equivalent simulations with switched positions example. 48
Figure 25. Double crack symmetric simulation example. 49
Figure 26. Equivalent simulations by linear superposition principle example. 50

Figure 27. Cracks' path for $L_1 = L_2$. 51
Figure 28. Cracks' path for $L_1 \neq L_2$. 52
Figure 29. Test for panel's symmetry along the horizontal axis. 53
Figure 30. Paths direction for horizontal axis symmetry test. 54
Figure 31. Strain E22 vs distance for the opposite direction paths. 55
Figure 32. Strain E22 vs distance for equal direction paths. 55
Figure 33. Cracks complete overlap example. 56
Figure 34. Cracks partial horizontal overlap example. 58
Figure 35. Details on cracks partial horizontal overlap. 59
Figure 36. Test configuration for critical distance test. 60
Figure 37. Simulation of single cracks for critical distance test. 61
Figure 38. Von Mises stress vs distances for 20 mm cracks. 62
Figure 39. Von Mises stress vs distances for 40 mm cracks. 62
Figure 40. Von Mises stress vs distances comparison between 40 mm and 20 mm cracks. 63
Figure 41. Test configuration for affected zone test. 63
Figure 42. Von Mises stress vs distances for 20 - 50 mm cracks. 64
Figure 43. Test configuration for the critical distance test for different crack lenghts. 65
Figure 44. Von Mises stress vs distances for the critical distance test of 20 mm cracks. 66
Figure 45. Von Mises stress vs distances for the critical distance test of 40 mm cracks. 67
Figure 46. Critical distance vs Crack length. 68
Figure 47. Simulation with relative distance lower than the critical distance for the same x position example. 69
Figure 48. Simulation with relative distance higher than the critical distance for the same x position example. 69
Figure 49. Simulation with relative distance lower than the critical distance for different x position example. 70
Figure 50. Real panel drawing. [21] 74
Figure 51. Paths and sensor locations chosen on the FE model. 75
Figure 52. Classes / labels examples. 76
Figure 53. Matrix arrangement for the training dataset. 79
Figure 54. Training and validation accuracy. 80

Figure 55. Training and validation loss. 80
Figure 56. Label vs signal number prediction performance graph. 82
Figure 57. Prediciton vs signal number prediction performance graph. 82

CHAPTER 1: INTRODUCTION

1.1. Overview

Deep learning techniques are taking over the field of mechanical engineering especially when it comes to the assessment of the damage and the remaining life of a mechanical component, through methods such as structural health monitoring. The development of new sensor technologies and the capacity of processing huge volumes of data gives the possibility to implement machine learning algorithms to perform damage diagnosis and efficiently tackle the issue of ensuring the optimal performance of a structure [1]. However, there are some challenges in the application of such solutions in terms of costs and time consumption, which can be solved with the proper implementation of numerical simulations. Finite element analysis seems a good alternative to overcome these issues, but the transition from real experimental tests to computational models brings some difficulties as well, since the data taken from the simulations need to be accurate enough to enchance and/or replace experimental results [2].

This research focuses on the development of a strain field database based on numerical simulations of a helicopter panel with a double crack damage. The main objective of this database is to train an artificial neural network algorithm to perform damage detection and defect localization for SHM diagnosis, which may be eventually upgraded to deal with prognosis as well. The necessity of this research relies on the need of replacing the costly experimental procedure that has been used for this type of applications by replacing the data measured from sensors with numerical data obtained by several finite element simulations.

This document is organized as follows: Chapter 1 briefly introduces the main topics of the thesis, Chapter 2 contains the state of the art with the most relevant research on SHM and ANNs applications for the aerospace industry, Chapter 3 illustrates the

procedure used to develop the strain field database and explains in detail all the limitations and considerations taken to achieve the result, Chapter 4 explains the training, validation, and testing procedure of the ANN and Chapter 5 gives the conclusions of the present work.

1.2. Structural Health Monitoring

Structural Health Monitoring (SHM) is a method to evaluate the integrity and safety of a component through the collection of data measured from sensors, such as vibrations, acoustic emissions, ultrasonic and thermal imaging, SHM is normally used to identify and characterize damages, but it also allows assessing the integrity of a given structure by analyzing the evolution of the damage detected.

In principle, the scope of SHM can be divided in three main components:

1) Structural Assessment
2) Structural Monitoring
3) Structural Control

These components (also known as SAMCO) represent the main objectives of SHM. In particulas, Structural Assessment deals with the determination of the integrity of the structures in the actual conditions and with the assessment the resistance of the components in the current state.

Instead, Structural Monitoring is related to the supervision of structures in continuous basis by means of sensor data to maintain the optimal performance of the component and assuring the availability of it on demand.

Finally, Structural Control considers the behavior of the structure in unexpected situations and focuses on mitigating or reducing the effect of this undesired load cases, which may be caused by off-design operation conditions or different environmental

loads, by means of control mechanisms used to maintain the response of the system under the preferred limits. [1], [3]

Summarizing, SHM is a set of methods and techniques that basically aims to ensure the performance or utility value of a structure; therefore, maintenance is essential to achieve this goal [4]. However, maintenance is done in different ways, each which gives different results and implies different costs. Currently, most of the components are subjected to what is called periodic maintenance where, as the name suggests, the maintenance procedure is done on a fixed time and normally consists of doing general actions to preserve its well functioning, independently of the degree of deterioration. This is the most common used method, but it has complications when unexpected failures are present in between two periods of maintenance, leading to a greater probability of downtime and, greater operational and maintenance cost. Continuous monitoring in SHM is a useful tool to apply preventive maintenance, according to which repairs are done even when the component does not require it. This allows reducing downtime costs and avoiding catastrophic failure of structures [5]. The networks of sensors that indicate the real time state of the structure allows the implementation of preventive maintenance protocols that reduce the probability of premature failures and permits the execution of timely corrective measures to reduce the structure's shutdown [1]. Currently, SHM is applied mostly on those structures the society depends on, which are often called as strategic or priority structures meaning that they have an important impact on the economy, the environment, the life quality, or they might be structures which offer a high employment prospective for the community, such as bridges, nuclear power plants, hospitals, public buildings and, aerospace systems [4]. Certainly, due to the importance of these structures, a strict maintenance plan must be applied to minimize the risk of failure and to avoid the negative implications of interrupting the operation of this systems, which would impact on the society.

As mentioned above the necessity of SHM relies on the importance of conditioned-based maintenance, that accounts for the unforeseen variations of the loading cycle exerted on certain structures. Time-based maintenance is generally prone to large downtimes and high maintenance costs, which can increase in case of failure due to unpredictable variation of the conditions the structure is operating under. To begin with the application of SHM methodology, first it is important to understand which are the components of it and how each one of these elements is used during the process. A flow chart summarizing the interaction of these components is shown in figure 1. As stated above, one crucial requirement for SHM to work properly is having the possibility to introduce a continuous monitoring procedure to the analyzed component. For this kind of procedures, a huge amount of information must be gathered on time; currently complex sensor networks are used to obtain data on real time basis. Several types of sensors are available on the market, such as acoustic emission, smart or sensor coatings, microwave, thermal imaging, ultrasonic and Fiber Bragg Grating (FGB) sensors [1]. The data measured by these sensor layouts is later received by a data acquisition system (DAQ), which strongly depends on the type of sensor, the type of data transmission and the sensor location. All the data is transferred by communication systems to the data processing devices where generally a statistical analysis is performed. This statistical analysis accounts for the strategies used to extract the parameters needed to perform the diagnosis for structural assessment. Basically, there are two types of statistical models based on their learning framework, supervised and unsupervised. The hardest task inside SHM is to choose a statistical analysis among the existing ones (some examples are presented in [1]) to identify the proper information for assessment and monitoring[1], [6]. Neural networks under supervised learning framework are the ones analyzed in this reseach (refer to the following section of this chapter).

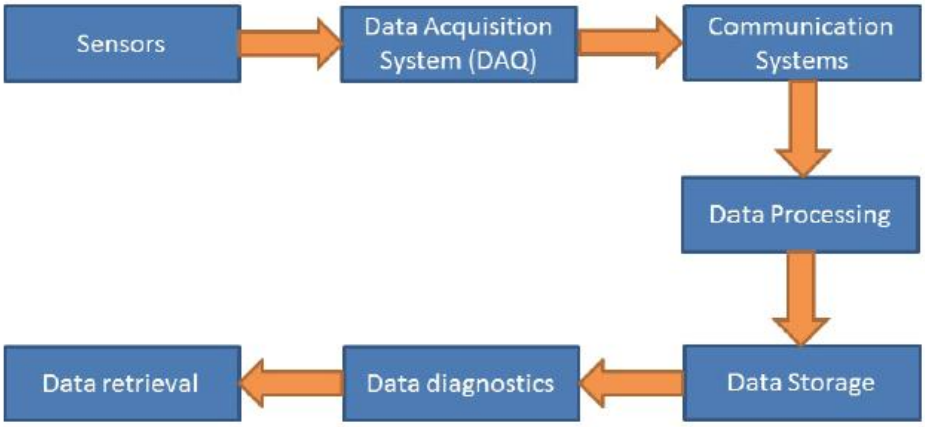

Figure 1. SHM flowchart. [1]

Knowing the structure and the elements required to perform SHM, one must also consider the levels of classification inside SHM that suits for the actual state of the structure. The five levels are:

1) Determine if the structure is damaged or not.
2) If the structure is damaged, try to localize the defect.
3) Quantify the amount of damage according to the data obtained.
4) Predict of the progression of the damage and the remaining life of the component.
5) Recommends corrective measures to recover the strength and functionality of the structure.

The implementation of SHM not only gives the information of the actual state of the structure, due to the continuous monitoring, but it also reduces the downtime present on the time-based maintenance, enhances the overall performance in terms of reliability and durability of the structure and reduce the investment in maintenance and inspection labor [1].

Nevertheless, there are some issues in SHM as well, starting from the technology used, the applications, the kind of data acquired, the costs and the damage detection itself. As mentioned above, several sensor technologies are available to be introduced into the SHM scheme explained before. However, some of these technologies have

exclusively been developed for a given application in SHM, thus requiring additional engineering effort to include such devices into a network of sensors. Moreover, there is no single technology that suits all the existing cases in which SHM can be applied, since there are factors such as materials, component geometry and different damage scenarios that make the use of a single sensing technology impossible to employ in all conditions [7]. Another major challenge is on the data obtained by the sensors, which, in most cases is gathered at a specific position, leading to the necessity of interpolating data to get insight into the critical points between sensors, thus reducing the accuracy and the reliability of the results. The fact that the sensors must be embedded into the structure increases the cost of implementing SHM with respect to other alternatives. On the positive, side these issues have been tackled by improving the sensor layouts used in SHM and in the future it is expected to have better networks with higher production of sensors that will reduce the cost of structural monitoring [1].

1.3. Artificial Neural Networks

Artificial neural networks (ANNs) are computational systems used in the analysis and the processing of information based in the emulation of the structure of the biological neurons found in the human brain, as shown in figure 2. ANNs represent one of the starting points of what is known as Artificial Intelligence (AI), they are widely employed in several fields due to the capacity of fitting complex nonlinear models by gathering information and detecting patterns in the data through a process called *training*, which allows the neural networks learn through experience. Neural networks belong to the family of machine learning algorithms, which is different from the classical type of algorithms that normally require physics-informed programming to solve such problems [8].

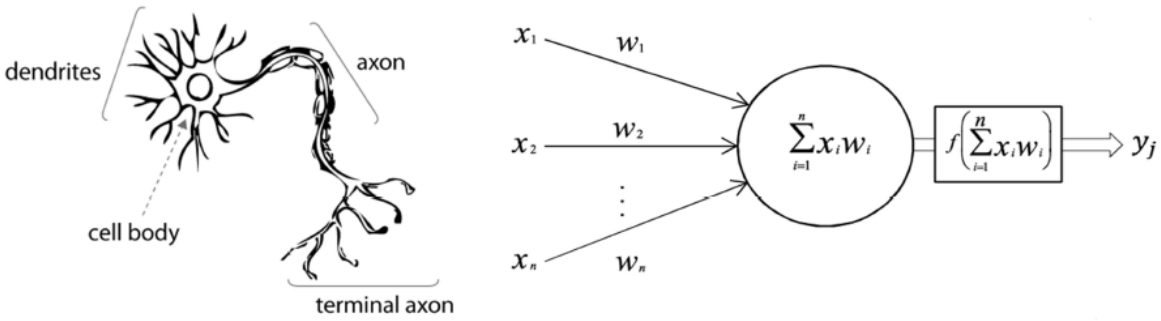

Figure 2. Human brain neuron vs artificial neuron. [9]

ANNs are composed of groups of artificial neurons called *processing units*, which are stacked to form layers. Processing units and layers interact by a set of connections similar to the synapses on the neurons, which are commonly called edges or *connections* [8]. Each of these connections is weighted depending on the respective value learned durin the training process, which is progressibely adjusted to obtain the desired outcome. Generally, processing units are organized to form an input layer, a set of hidden layers, and an output layer, as presented in figure 3 for a classical feed forward multi-layer perceptron neural network. The input data is gradually transformed as it flows through each layer, until it reaches the aouput layer.

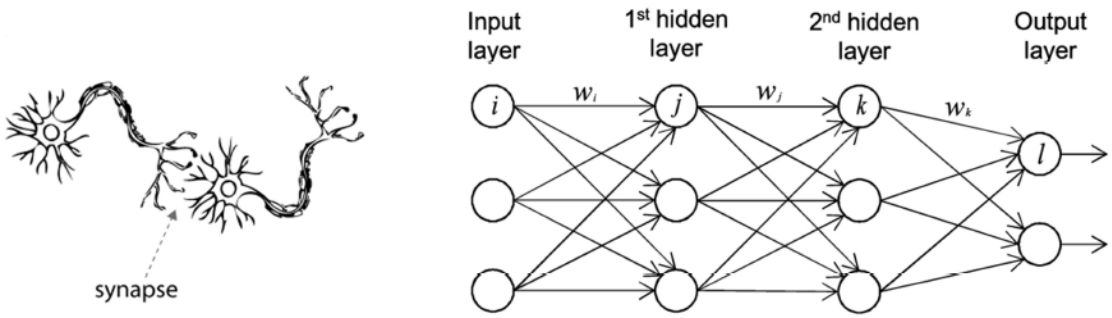

Figure 3. Human neural network vs artificial neural network structure. [9]

As mentioned above, to have a proper performance of the system the neural network must be trained through a process called *supervised training*. This process needs a labeled training dataset to be available, that consists of input data with corresponding known results. The training procedure consists of employing an optimizer which gradually modifies the connections between adjacent neurons to reproduce the

expected output value. During each interaction, once proper weights values have been assigned, the current output obtained by the model is compared with the expected label included in the training dataset (target output) for that corresponding input, the error between the predicted output and the target output is calculated and the weights associations adjusted to reduce it as much as possible. This process of minimizing the error during the training stage is called *error back-propagation* [8], [9].

In mathematical terms the training dataset consist of a group of inputs $\boldsymbol{x}$ and its respective outputs z. The training model is built on a linear function as follows:

$$\hat{\mathbf{z}}(\mathrm{x}) = \mathrm{W}\mathrm{x} + \beta$$

where $\hat{\mathrm{z}}$ is the predicted output obtained by the model, W is the weighting matrix and β is the bias vector. As previously stated, once the comparison between z and $\hat{\mathrm{z}}$ is done an optimization problem is run to reduce the discrepancy between these two values, thus modifying the values W and β during the process [9]. To improve the accuracy of the system additional layers $\boldsymbol{y}$ are introduced as follows:

$$\mathrm{y}(\mathrm{x}) = \sigma(\mathrm{W}_1\mathrm{x} + \beta_1)$$

$$z(\mathrm{y}) = \mathrm{W}_2\mathrm{y} + \beta_2$$

where σ represents an activation function, such as Sigmoid function, to define whether the information inside that processing unit is useful or not. After this procedure is done, a validation process is used to check the if previously trained system works properly and the training procedure can be considered completed. (For further details on NNs please refer to reference [9]).

With a trained ANN it is possible to perform complex tasks in a wide number of fields. Focusing on the scope of this research, ANNs are strongly entering to the field of SHM due to the number of functions that can be done by properlu tuning this type of algorithms. One of the hardest parts of the process of SHM is properly detecting a damage, classifying it and proposing future corrective actions considering the predicted progression of the damage, which are all actions where ANNs stand out

and give successful results. This topic will be covered in detail in the following chapter.

CHAPTER 2: STATE OF THE ART

2.1. Artificial Neural Networks for Structural Health Monitoring

The integration of artificial neural networks and machine learning algorithms in SHM started a couple of decades ago due to the huge potential of processing large amount of sensor data and recognize patterns for SHM purposes [10]. Also, this type of algorithms gives the ability to the system to learn from experience to improve the prediction and decision making make them reliable when properly trained. There are numerous ways to apply these techniques in SHM and normally they depend on the way the algorithm works, ANNs have three types learning schemes that favor different kind of tasks, these learning schemes are: supervised, unsupervised, and semi-supervised [10]. On supervised learning the algorithm is trained by labeled set of data, this data contains the input data plus its respective result to give the neural network a reference or provide a rule to classify new upcoming data. On the other hand, unsupervised learning requires only a set of input data with general rules that allows to train the algorithm by grouping samples datasets. Semi-supervised learning is basically a combination of the latter two. Unsupervised learning can be use for damage detection by clustering of structural response data while supervised learning can be used on classification and severity of the damage. For SHM purposes the most used learning scheme is supervised due to the number of tasks that require a labeled training set to perform properly, specially for techniques such as single/multi-perceptron neural networks.

As mentioned before ANNs have become popular solutions to pair with SHM to assess the well-functioning of civil and mechanical structures, it has a great potential

on diagnosis and prognosis of damage, and it has been successfully applied in several cases but its still moving towards improvement in several aspects. One of the challenges of using this kind of techniques relies on the number of useful data required to train these algorithms by supervised learning scheme; initially this information was extracted (and sometimes still is) from previous sensor data that gives the algorithm the needed guidance to learn and recognize the patterns from measurable features of structural damage and relate them to physical properties of the structure [11]. Nevertheless, this approach has a series of inconveniences, starting from the availability of this data in the right amount to properly train the algorithm to have a level of accuracy, a lot of experimental tests must be done to cover all the possible critical base scenarios plus the healthy conditions of the structure to generate a training dataset big enough to train, validate and test the ANN. This leads to the second issue that is the costs of the experimentation and laboratory testing plus the time required to gather this training data from an empirical way.

To face these issues the option of gathering the data from numerical simulations by means of finite element models raised, these computational tools are able to simulate accurately different healthy and damage scenarios, environmental conditions and other unexpected cases that are harder to replicate in a laboratory [2], [10]. The possibility to recreate different operative conditions of the structures allows to track the effect of numerous types of damages with helps with the direct labeling of them, creating a supervised multiclass SHM system. The drawback of using the approach is related to the fact that if a damage is going to be simulated it must be known, in order to replicate and assess the progression of the damage this one has to be recognized previously, after placing the defect in the model it is possible to simulate several conditions based on the information gathered from the actual sensors on the structure or considering diverse load and environmental conditions.

As mentioned before, the state of the art of ANN on SHM relies on the improvement of the process overall, the sensor technology advancement is allowing the use of more

efficient, low power consumption, accurate and lightweight sensors, also the design of SHM techniques is being optimized by reducing the number of sensors installed maintaining a reliable level of accuracy, on the other hand improving the training techniques making them cost effective, reliable and versatile and finally meliorating the architecture of the ANNs to enhance their performance on duty.

Various research has been developed to test this progression in the field successfully; this section of the document will concentrate on the research focused in other fields but the aerospace industry. As stated on the previous chapter SHM has been used in several fields and is vastly implemented in civil structures such as bridges, energy production plants, offshore structures, etc. For this reason, there is a lot of SHM development on the civil engineering field, for example, on the document presented on reference [2] a deep learning convolutional neural network (CNN) algorithm is trained using an optimal FE model used to measure the dynamic response of a benchmark steel beam which is subjected to vibration by means of an electrodynamic shaker. To check the performance of the CNN trained by the dataset obtained in the FE model, a real experimental test is done with the same type of damages, one damage is a crack on one side of the beam and the other damage is generated by the fixation of a mass on the extreme of the beam as shown in figure 4.

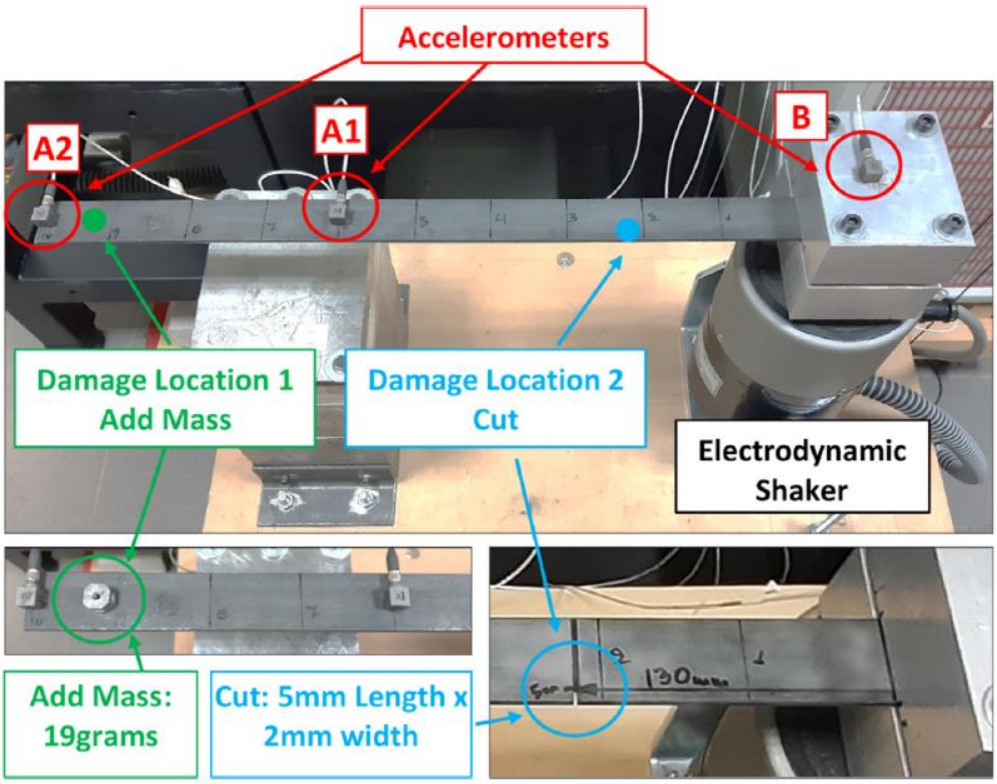

Figure 4. Steel beam experimental setup. [2]

The results shows that optimal FE models are required to perform well on damage identification problems, nominal FE model are not suitable to train algorithms for this type of tasks and can increase the cost of computation if its not well implemented. Also, in the analysis, a comparison between multi-head and single-head parallel filters CNNs was develop showing that the multi-head outperforms the single head in the accuracy of the prediction of the models, those predictions were validated with experimental data [2]. This research proves once again the versatility of this type of numerical tools to collaborate on SHM matters and how fast and cost efficient they are with respect to the experimental tests used conventionally for training deep learning algorithms.

Other kind of research is focusing on the fault detection of the sensors embedded in SHM that may cause wrong measurements that may affect the diagnosis and prognosis of damage on a given structure, especially in bridge structures. As it is known, the sensors are not able to recognize damage by their own, they measure a

physical damage feature on a structure (strain, vibration, wave propagation, temperature, etc.) that is interpreted as flaw on the system by an ANN or any other machine learning algorithms [11]. This leads to the problem mentioned above, if the one sensor is giving wrong measurements, it may lead the algorithm to give a wrong diagnosis of the damage. In order to face this problem, the creation of decentralized autonomous sensor fault detection is presented on the research of K. Smarsly, K. Dragos and J. Wiggenbrock, this system is based on the principle of analytical redundancy where instead of installing multiple sensors over the structure to measure one single parameter, the system uses the relationships and coherences of the sensor available sensor data to detect whether the measurements taken by the devices are correct or incorrect and determine if the sensor need recalibration or change. It is known that after processing the data obtained by the sensors using the Fourier transform, the modal peak amplitude or resonant response of each sensor is correlated with the other ones in the same structure [10]. The deviation of the expected peak and the actual peak shows a miscalibration or wrong measurement of the corresponding sensor, on the document presented in reference [10] a neural network is used to recognize deviation of the expected and the real modal peak amplitude to determine the error on the sensor measurement, then the training dataset is gathered, and the results are compared with and experimental test to evaluate the results. This research shows that this kind of methodologies can be used not only to optimize the sensor layout to reduce the cost and the weight of SHM implementation on the structure, which is relevant in the aerospace industry, but also to improve the reliability of the ANN predictions, thus of the diagnosis of damage for a better calculation of remaining life or defect progression in the system.

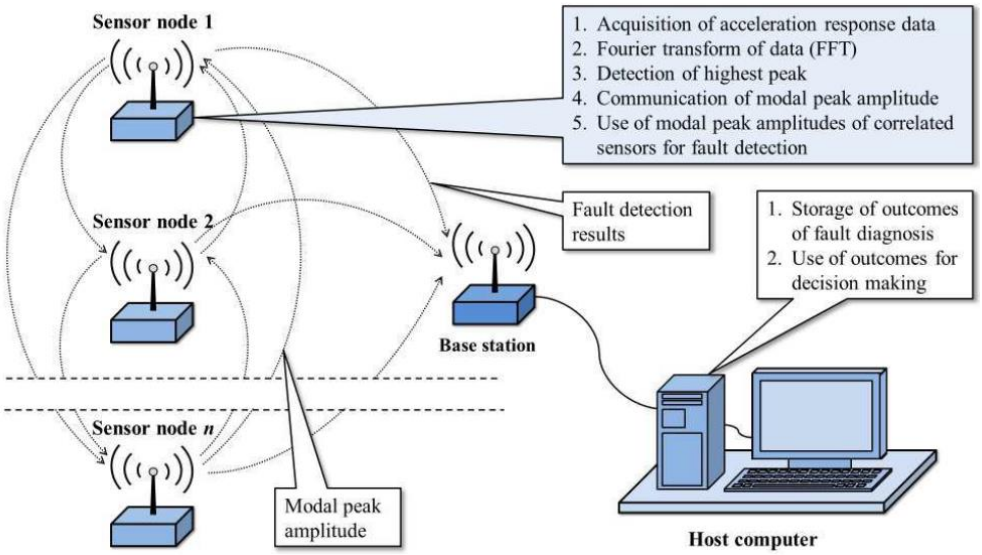

Figure 5. Decentralized autonomous sensor fault detection based on analytical redundancy scheme. [10]

The documents illustrated in this section of this thesis are just some examples of how the SHM processes are improving over time and how the development of optimized neural networks, sensor technology and many other factors are driving the implementation of SHM to many structures and systems in the industry.

2.2. Structural Health Monitoring on Helicopter Fuselage Panels

SHM is taking over several fields in engineering, from construction to nuclear power plants, it has been widely used due to the numerous benefits explained in the previous chapter and it is a great opportunity to increase the availability and reliability of civil and mechanical structures.

Aerospace structures are one of the most complex mechanical systems that operates nowadays, they are considered strategic structures because of their important role in the transportation and military industry. In order to maintain these systems a set of design rules have been used to ensure the optimal performance of the component and to counteract the effect of aging on its parts, however, the design process of each type of system should be covered differently specially when speaking about helicopters.

The difference between the flying principle of conventional fixed wing aircrafts and helicopters implies different design analysis and procedures [12]. The first thing to be considered in helicopter design is the load spectrum to which the system in subjected, starting from the high number of low-amplitude cycles generated by the mechanical rotation of the rotor blades, that generates high vibratory loads, followed by low velocity impact damaged which is related to the harsh environments in which helicopters generally operate. This load history promotes crack nucleation due to accidental damages and reduces the fatigue life of the component considerably; a time-based maintenance is not feasible to for this type of applications due to the short time intervals between the maintenance due to the high fatigue damage caused by the previously described load spectrum [12], [13].

Currently Health and Usage Monitoring Systems (HUMS) has gained popularity among helicopter community to effectively tackle these issues, these methodologies are used to improve the performance and the utility value of life-constrained components in helicopters which ends up in enhanced flight safety and mission reliability as well as reducing maintenance and inspection costs, which represent 25% – 27% of the operational cost of aircrafts and helicopters [12]–[14]. Nevertheless, this type of procedures are generally develop for components that directly affects the flight performance of the system, leaving components like the fuselage out of the scope of the analysis. However, research is aiming at implementing SHM methods to this latter structure, for example, the project "Helicopter fuselage crack monitoring and prognosis through on-board sensor network" (HECTOR) coordinated by the European Defense Agency (EDA) oversees the integration of SHM for diagnosis and prognosis of damage in helicopter fuselages by the combination of sensor data with FE models to understand the progression of the defects within the structure [13].

On the research done by professor Marco Giglio, Andrea Manes and their research team, a proposal done to monitor the panels from the fuselage of the tail of a helicopter is done by means of sensor networks which gather real-time data of the state of the

panel, according to the sensor data an advance FE model of the whole fuselage is done focusing on the tail region and knowing the loading history it is possible to simulate the progression of the defect subjected to the corresponding stresses and define future preventive or correcting actions in flight and maintenance. The tail of the helicopter is chosen because it corresponds to the section of the helicopter subjected to the highest moments due to the reaction of the tail rotor to the torque load exerted by the main rotor, this cyclic loading will generate fatigue damage over the structure formed by the frame, the panels and the stringers.

The research starts with a brief description of the available sensing technologies currently available for SHM for aerospace components and explains the basic principle of each sensor. Several types of sensors are available, among the ones covered on the document are crack gauges, comparative vacuum monitoring (CVM), Acoustic – Ultrasonic (AU), Acoustic Emission (AE), Fiber Bragg Gratings (FBG), Eddy Currents, etc. From the above mentioned, FBG and CVM sensors are highlighted due to the suitability for aerospace applications. For this specific project, a network built out of FBG sensors is used, these devices are optic fiber sensors for strain mapping of the structure that are mounted on the surface via an element called sensor pad that reduces the variation of strain transfer from the surface to the sensor passing through the adhesive needed to bond it, this kind of sensors are accurate, lightweight, and low – power consumption which makes them suitable for flying. (For more information about the sensors refer to reference [12])

After measuring the strains and acquiring the data (the details of the DAQ are presented on reference [13]) the data can be transferred to ABAQUS where the FE model of the tail's fuselage is created like it is depicted in figure 6, the whole structure of the tail is used to define the stress and strain field for the given condition and a sub-model is created to address the specific portion of the structure affected by the damage. Due to the different scale of both analyses, it is necessary to do two separated models to reach a good level of accuracy; on the sub-model an analysis of the crack

growth is done to assess the remaining life of the component, the available flight time, and the following maintenance procedures [12].

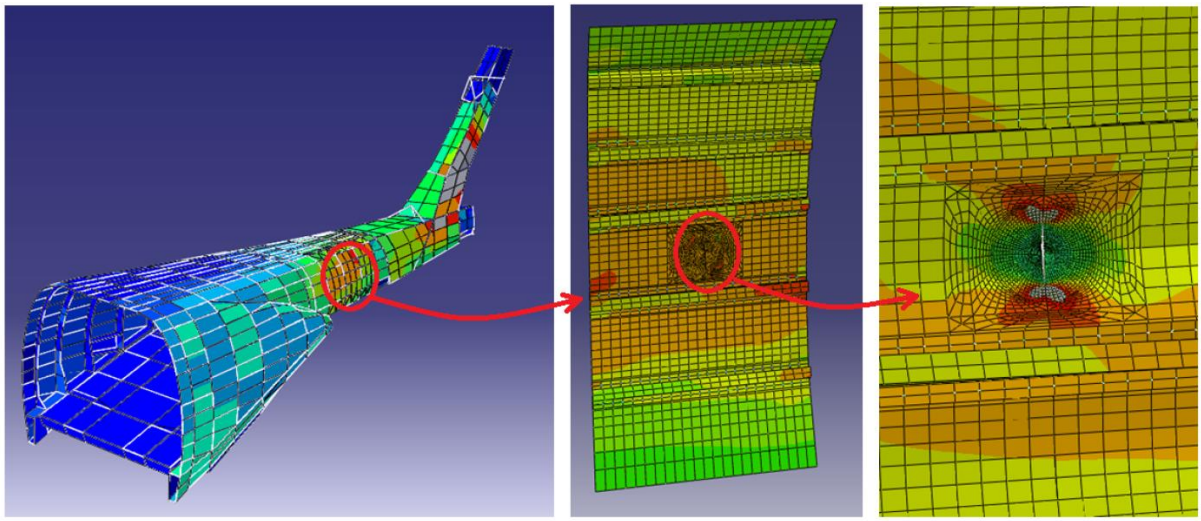

Figure 6. Helicopter tail FE model and sub-models. [13]

The FE analysis is done for both the healthy and the damaged structure, when the structure presents a defect it is displayed on FE model as a crack which is later analyzed by fracture mechanics approach. To define the crack propagation of the damage found in the data measured by sensors, the loading spectrum must be introduced to ABAQUS as a cyclic load considering the forces and moments interacting on the structure, as stated before the main loads that may induce crack propagation of the defects are the reaction moments and the negative lift generated by tail rotor. Figure 7 shows the typical trend of stresses of a helicopter in mission, this information is crucial to calculate the crack growth present on the fuselage.

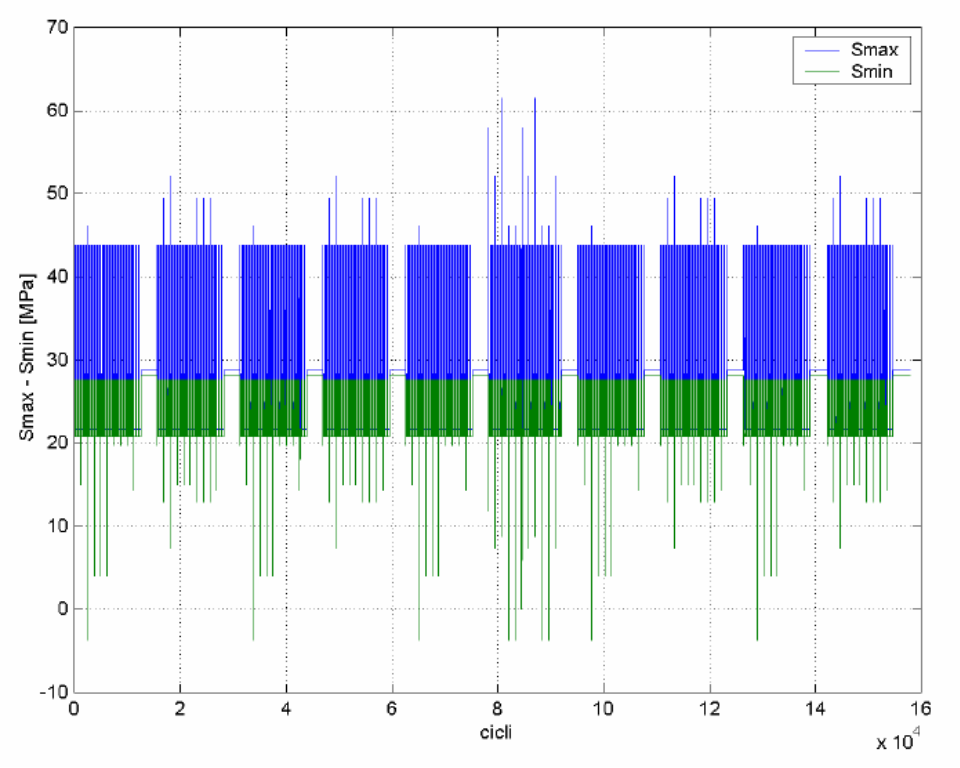

Figure 7. Typical load specturm of a helicopter in operation. [12]

To complement the process an artificial neural network is used for damage detection and classification, this algorithm is able to analyze the strain patter given by the sensors to define the location and the extend of the damage, which is later processed by the FE model to evaluate the degradation of the overall structure caused by the defect and predict the progression of the defects.

The procedure includes the installation of the sensor network on the structures, the data acquisition system, the most suitable communication system for a closed metallic structure in flight, and the proper signal processing and filtering techniques; all of these components together form what is called Embedded Structural Health Monitoring (ESHM). The details of the selection of the components of ESHM is specified in reference [13].

This latter research was one of the first steps in the development of SHM methods for helicopter panels. Agusta Westalnd, now part of Leonardo and the research group of Politecnico di Milano, leaded by Professor Marco Giglio, Claudio Sbarufatti and

Francesco Cadini had studied the behavior of helicopter panels under single crack damage in different scenarios continuing with the investigation mention at the beginning of this section of the document.

Now that the feasibility of FE models in SHM methods was proven, the new research is focused on the behavior of the progression of the cracks on the component, specifically speaking, the objective of the latest studies was to verify these techniques to assess fatigue damage on cracked panels. For the analysis of fatigue several issues have been covered, first the improvement on the architecture of the ANNs was optimized via analysis of variance (ANOVA) to create an neural network with the correct number of hidden neurons in a three-layers ANN that allows the best statistical results in terms of performance and sets a threshold where the number of hidden neurons is limited to an optimized value in which larger number of neurons does not present an statistical evidence of influence of that number on the error function [15]. This optimized algorithm is later used for damage identification, localization, and quantification for the same helicopter panel with FBG sensor network used in this present thesis document (refer to chapter 3). The optimized ANN is trained based on the strain field training dataset obtained by means of numerical simulations of several FE models done in ABAQUS considering two types of damages, a single crack in the skin bay and a single crack on the stringer. Figure 8 shows a) the skin area where the defects can appear according to the positions of the sensors along the panel, b) an example of a bay crack in the middle of the panel, normally caused by accidental damage or battle damage, and c) an example of a stringer failure, that represents the worst case of maximum crack propagation [15]. Then the ANN is tested on a real panel for the two damage scenarios, the bay crack was simulated with an intentionally drilled notch on the center of the central bay with a created crack of 16 mm subjected to a cyclic sinusoidal load while the stringer failure is tested over a panel with two broken stringers that failed naturally under fatigue damage.

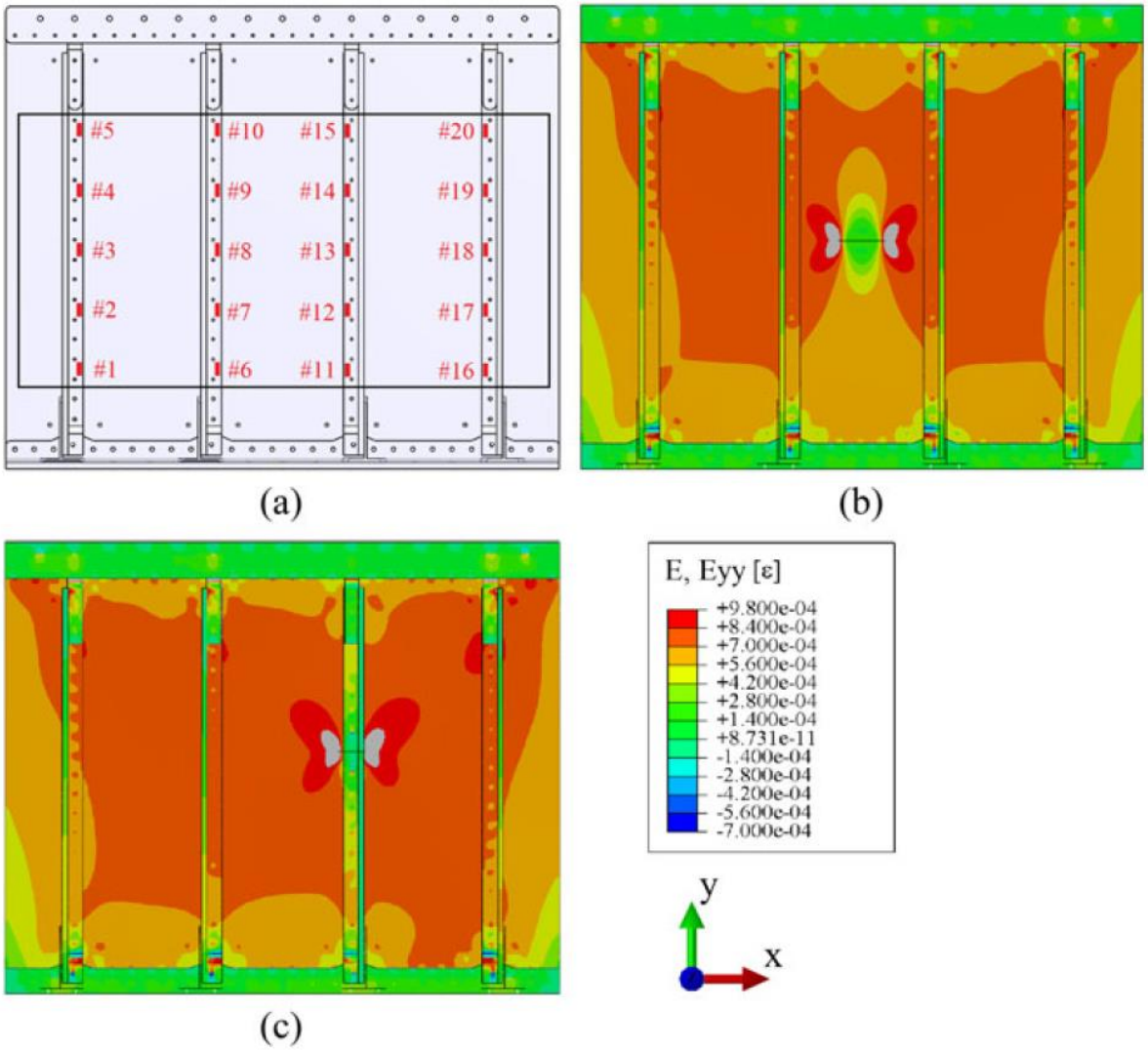

Figure 8. a) Sensor locations and interest area, b) Bay crack FE model, c) Stringer failure FE model. [15]

The results shows that the implementation of the ANOVA for the optimization of the ANN worked properly in the selection of the correct number of hidden neurons in order to improve the performance of the algorithm, also aiming to avoid undertraining and overtraining. The process of training using purely numerical simulations makes harder the identification of overtraining, however once this is solved the ANN performs well in the detection, localization, and the quantification of damage for both damage scenarios analyzed.

The following studies conducted by C. Sbarufatti and professor's Giglio research group are focused on the improvement of fatigue damage prediction of SHM methods paired with ANNs, the previous research presented above is one of the first steps in

the development of these optimized algorithms. After this investigation more experimental setups were built to test the feasibility of these ANN techniques when trained by numerical simulations for SHM applications. The following study of C. Sbarufatti is centered on the verification of a diagnostic algorithm based on a neural network trained by numerical simulations. The validation of the output of the deep learning algorithm is done by the comparison between the predictions of the diagnosis algorithm and the fatigue test performed on the lab, the aim of this experiment was to check the detection capability of the algorithm and compare it with actual results from real panels under cyclic loading with artificial bay cracks intentionally generated on the center of the skin ash shown in figure 9. Also, the performance of the SHM system is qualified by the guidelines of the ARP-6461 "*Aerospace Recommended Practice—Guidelines for Implementation of Structural Health Monitoring on Fixed Wing Aircraft*". The ARP-6461 is intended to assess the performance of the SHM methods assuming that the sensor network is installed permanently on-board, the crack nucleate and propagate on the section analyzed by the SHM [16]. The way to qualify the system depends on the detection of flaws, the minimum detectable crack length (MDCL) defined by the one-sided tolerance intervals for normal distributions (TIND). (For further details on the TIND procedure please refer to reference [16]).

Figure 9. Experimental setup for fatigue damage testing. [16]

As stated on the previous paragraph, the ANN used for the diagnosis of fatigue bay cracks is trained using FE simulations. The process is similar to the one used on the later research done by C. Sbarufatti for the analysis of bay cracks, however, in this case only bay cracks are simulated on 100 random center position and seventeen different crack lengths varying from 20 to 100 mm with a 5 mm step for a total of 1700 simulations [16]. The automatization of the simulation process parameters is done in MATLAB and the strain field is extracted from each numerical simulation to create a training dataset to feed the ANN. The example presented in figure 10 shows the simulation of a crack on the center of the skin exactly on the same position of the one done in the experimental test, the measurements taken by the sensor ID1 and ID2 clearly depend on the position and the extent of the defect which is progressively affecting the strain field of the structure as it becomes larger or as it is closer to any of the sensing devices. The extraction of the strain data is done like on the present thesis and it is explained in detail on the following chapters of the document.

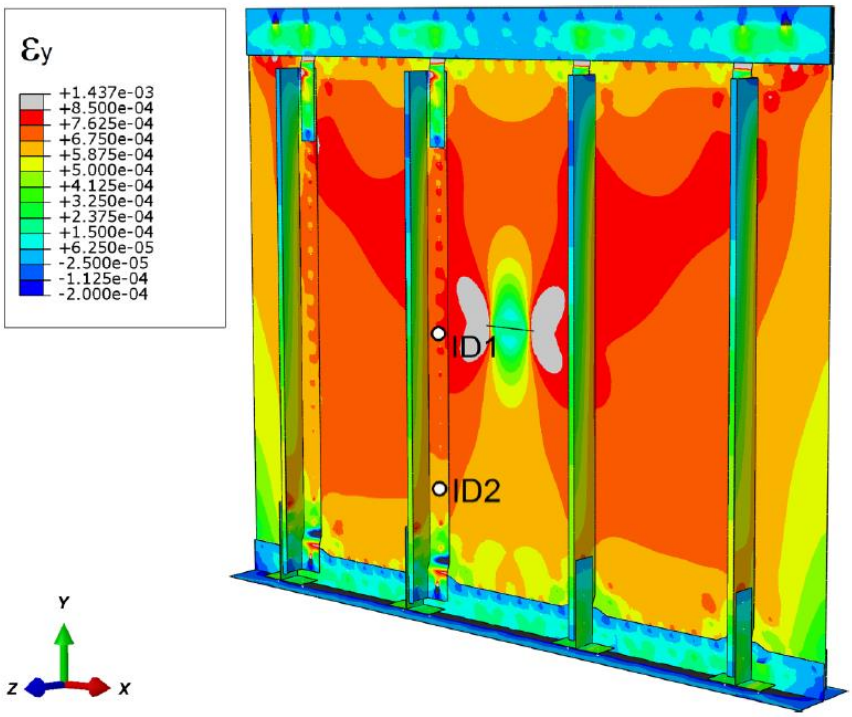

Figure 10. FE model of the bay crack simulation. [16]

The SHM system is tested using the experimental setup with the strain measurements of the FBG sensors used as the input to feed the algorithm. The results shows that the performance of the diagnosis algorithm is good and the TIND method reported on the ARP-6461 is more suitable than the previous non-destructive inspection (NDI) techniques for SHM matters. The MDCL procedure depends on the relative distance of the sensor to the center of the crack and the number of sensors used, for the cases tested experimentally the crack on the center of the skin is the one farthest from the sensor, thus the lower sensitivity is presented in this case. Summarizing the diagnosis algorithms in SHM can bring several benefits compared to the current NDI solutions but there is a lot of improvement to be done. Certainly, this is the future of damage tolerance design and life assessment of structures.

This covers some of the most relevant studies regarding the state of the art of the topics presented in this thesis document, however the progression of SHM seem unstoppable, more and more studies aiming to improve and enhance the application of this methods in the aerospace industry. Part of the latest studies done in the fatigue crack growth (FCG) SHM methods for aeronautical panels is focused on the reduction of the computational burden of these complex algorithms for structural models. An example of this is presented in reference [17] where a diagnosis and prognosis algorithm for damage in helicopter aluminum panels is made by an artificial neural network-based surrogate modelling embedded within a particle filter algorithm giving successful results on the determination of the remaining useful life of these components.

Most of the studies conducted in this area were done considering just a single defect on the component, nevertheless, there are situations in real life where the degradation of the structure can be accounted by not just one defect but the combination of the effects of multiple cracks acting simultaneously over the panel. Fortunately, some of the most recent studies are using SHM method to assess the structural integrity of aerospace panels under multiple site damage (MSD), for example, the document

presented on reference [18] expresses the importance of the analysis of MSD after the incident of Aloha Airlines Flight 243 of 1988. The main concerns of MSD and its effects of the structural integrity of the aeronautical panels is related to three main topics: crack nucleation, crack propagation and residual strength. The document on reference [18] is centered in the prediction of the residual strength, that refers to the capacity of the structure to support stresses before failure, of aircraft aluminum panels with different configurations (unstiffened, stiffened, stiffened with a broken middle stiffener, and bolted lap-joints) and different aluminum alloys (2024-T3, 2524-T3, and 7075-T6) by means a of artificial neural networks using a data-driven approach using 147 datapoints with different material-configuration combination to perform the training and validation of the ANN. The results obtained by Hijazi, Al-Dahidi and Altarazi shows that the ANN is able to predict the residual strength of the panel in all the configurations proposed with an absolute error of 3.82%, which is considerably higher than the ones obtained using other analytical or semi-analytical methodologies[18].

Aluminum alloys are widely used in the aerospace industry; however, composite materials are strongly positioning as crucial material in the industry due to their outstanding mechanical properties and lightweight characteristics. For this reason, some studies had deepened in the use of SHM in this kind of material that now represent a significant percentage of the structure of aerospace systems. The investigation done by in reference [19] aims to determine the best ANN architecture to develop a SHM method for damage detection on composite materials when there is no prion knowledge on the characteristics of the damage, the types of ANNs evaluated are the common feed-forward Multi-Layer Perceptron (MLP) and the Radial Basis Function (RBF) ANNs. The sensor network used on the SHM system is based on piezoelectric sensors to gather data of the location and extend of the damage and define it by a damage index. The training and validation of both ANNs was done by means of a training dataset obtained through a large number of simulations done by Dual Reciprocity Boundary Element Method transient analysis, which allows the

analysis of the upcoming electrical signal from the piezoelectric sensor. The results show that in order to reach the same level of prediction accuracy for both types of ANN the RBF requires a larger amount of data for the training procedure compared to MLP, nevertheless, the time required to train the RBF is significantly lower than for the MLP [19]. This outcome allows the authors to conclude that depending on the amount of training data available one can decide which ANN to choose, if there is a large training dataset the behavior of RBF is superior to MLP, this could be the ideal scenario, but if the data available is limited MLP could be a better option. ANN has a considerable variety of setups and architectures that can be exploited depending on the available data and the application.

On the other hand, another important topic that must be taken into account to define the feasibility of SHM is the cost. A research has studied the impact of the cost of SHM over the operational costs of different aircrafts; as mentioned at the beginning of this section, the inspection and maintenance costs represent around a quarter of the overall operation costs and one of the main goals of SHM is to reduce this value giving the opportunity to stablish condition-based maintenance allowing less downtimes due to unexpected failures, hence improving the availability of the system. However, the implementation of SHM has some additional costs that can make the application of it unfeasible. In most of the cost/benefit analysis done in the aerospace industry on of the key factors is the reduction of the weight, normally this value has an impact on the fuel consumption which is reflected on the saved cost due to that diminution of the weight. Nevertheless, there is not a clear calculation of the effect of a weight increase due to SHM, for this case it is not only the extra fuel consumed but also the required payload that must be sacrificed in other to counteract in the extra weight introduced to the system [14]. Generally, the fuel weight in takeoff is a fixed value, so the weight added due to SHM is directly affecting the payload of the structure, the contribution of the implementation of SHM can be divided in three main parts:

1) Sensor Cost

2) Installation Cost
3) Inspection Cost

On the research done by Ting Dong and Nam Kim a comparison between some sensing technologies is done different relevant features as shown in figure 11. Considering the characteristics evaluated on the table they did a cost/benefit analysis with the implementation of the sensors PWAS (Piezoelectric Wafer Active Sensor) due to the high detection range even if the weight of these sensor is greater than the other two options (CVM and FBG) [14]. On the document is specified the configuration for the proper functioning of a PWAS sensor network mounted for SHM purposes on the entire fuselage of a Boeing 737NG. The results show that for this aircraft the implementation of the SHM will save up to $ 5M on maintenance procedures while the lost of revenue due to the reduction of the payload is about one order of magnitude greater ($ 50M). (For more details on the calculation of the costs please refer to reference [14]).

Performance	CVM Sensor	FBG Sensor	PWAS
Smallest detectable damage size	0.02 in.	N/A	0.2 in.
Weight	Light	Light	Medium
Capability of detecting closed crack	Yes	No	Yes
Detection range	Low	Medium	High

Figure 11. Comparison of sensor characteristics. [14]

It is important to consider that the number of sensors and the distribution of them on a helicopter may differ significantly from the configuration done on the Boeing 737NG, however, the considerations done on Dong's and Kim's research is relevant for all the systems in the aerospace industry. SHM should be reliable but also it should be cost efficient in order to implement it effectively in most of the strategic structures. Also, this allows to conclude that SHM is the path to follow but there is still work to be done, the sensor technology will have to keep improving to make this methodology feasible in terms on of costs.

2.3. Automatization of Double Crack Damage Simulations for Helicopter Panels

Currently most of the research done in SHM matters for helicopters and aircraft damage tolerance design in general has been done considering the degradation of the structures due to the effect of a single defect on a specific component or a whole system. Nonetheless, as stated on the previous section of this chapter, one of the most critical conditions in aerospace systems is the presence of multiple defects on the structure acting together, drastically reducing the remaining life of the component. To tackle this issue, several studies have focused on the integration of SHM methods that are able to detect and localize more than one defect on the structure, this is still in development due to cost and the time consumption that doing experimental test for several cracks require, also the number of combinations and cases is practically infinite which increases the complexity of the training of the widely used neural networks on SHM applications. Once again, the implementation of numerical simulations has been a good alternative to avoid the costly experimental testing and allows the replication of cases giving the possibility to build larger and more reliable training datasets for these deep learning algorithms used for the diagnosis of damage. FE models can be easily modified to introduce any type of known damage in any desired location, allowing the user to gather information of the dagame in several locations along the component. Generally, this process is pretty accurate as long as the physical models, the materials, boundary conditions and the inputs in general are well set and coherent with the real conditions of operation of the structure. Moreover, this approach enables the introduction of ambient effects and other unforeseen effects that may affect the load spectrum that is exerted to the system. Despite that theoretically the introduction of additional defects can be easily done, there is still the issue of how to efficiently simulate thousands of cases to build solid training datasets to implement them in the ANNs.

Following the path of the research done in references [16], [17], G. Verga generated a code to automatize the simulations of a FE model of the helicopter panel under double crack damage. The automatization code is done in MATLAB and allows the user to modify the location and the characteristics of both defects. Using as a base the FE model shown in references [15], [17], that is the computational representation of the experimental test shown in figure 12, G. Verga introduced two bay cracks to the model to analyze the effect of the combination of the defects in the stress/strain field. The modified model generated in ABAQUS outputs a script that contains all the required features to create the FE model which includes, parts, materials, partitions, constraints, loads, steps, and jobs. This script can be processed in MATLAB and can be modified at will to recreate the FE model with the desired changes.

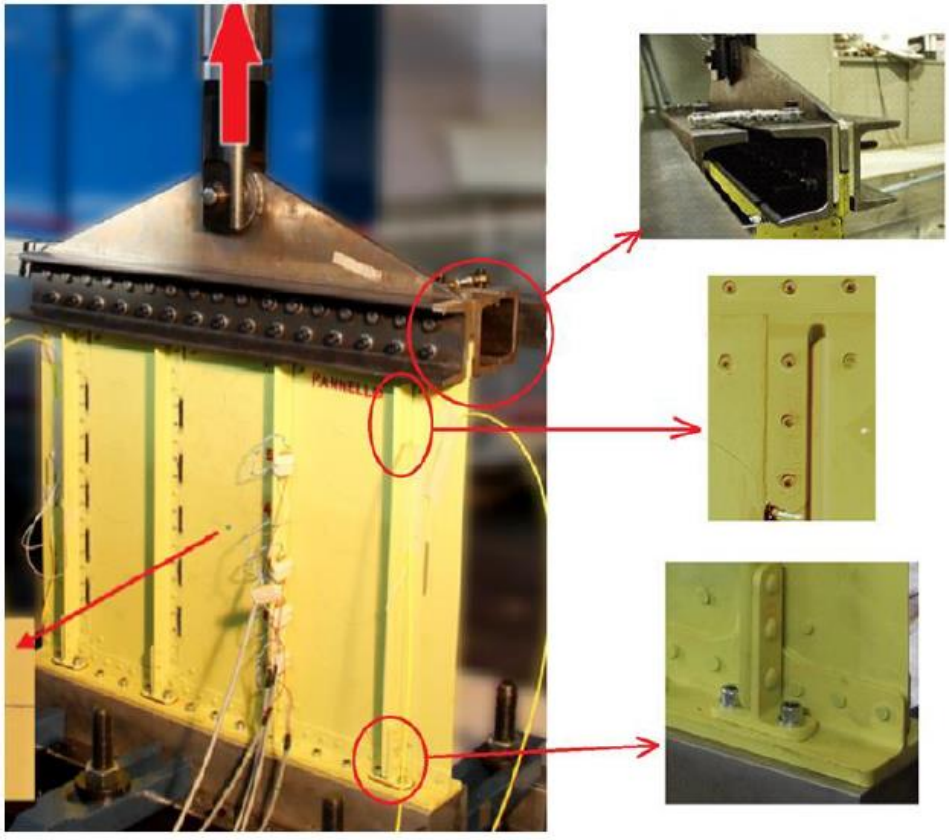

Figure 12. Experimental tension test setup for helicopter panels. [20]

The automatization of the code is done by taking the ABAQUS script with a finished model of two cracks and leaving as a variable all the features regarding the creation of the defects, to do that, first one must understand how the creation of the crack is

done and which features must be modified to be able to run the simulations smoothly. Basically, the crack is created as sketch line over the skin of the panel, to create this line three initial parameters are required: the horizontal position of the center of the crack (X), the vertical position of the center of the crack (Y) and the length of the crack (L). Due to the type of line created the center of the crack is located on the left-hand side of the line and the length is deployed from left to right until it reaches its given length. Once the sketch is done a partition is created over the line and two circular partitions are drawn on the extremities of the segment, these circles are later used to refine the mesh around the crack tips to have higher accuracy on the output stress/strain field also it is the point of propagation of the crack, as shown in figure 13 [20]. Once the partitions are done the interaction of the defect is set on the model, the interaction defines the partition as crack inside the FE model and will allow the software to analyze it as it is. (For further details on the creation of the cracks please refer to reference [20]).

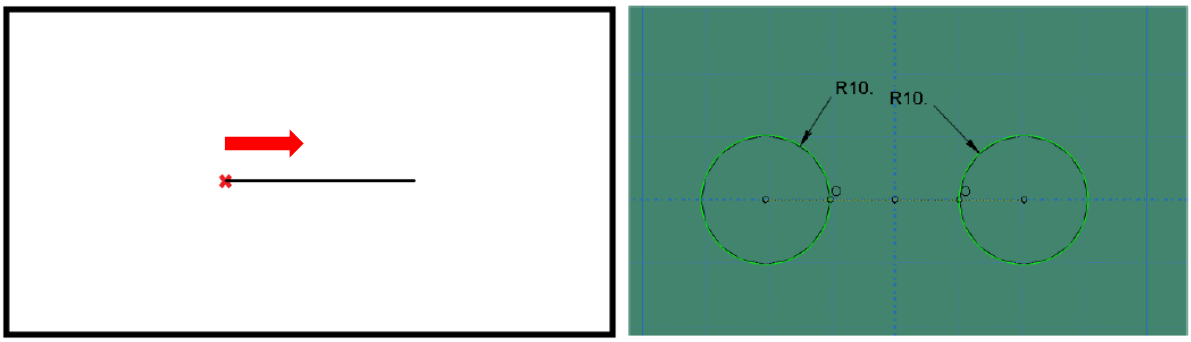

Figure 13. Creation process of a linear partition and circular partitions. [20]

Summarizing, the relevant parameters that have to be set as variables are the coordinates of the centers of both cracks, the size of the cracks and the radius of the circular partition on the extremities, that are related to the length of the defect. This variables can be introduced in a for-loop on MATLAB to do the cycle of variation of the parameters. Additionally, the code is built so that every job.odb file is saved from each simulation finished, this .odb file stores the results of the simulation and is where the stain data is extracted.

The thesis work proposed in this document is the continuation of G. Vergas work and the code built to create the strain fiel database is based upon his code's foundation, the fuction created in his project work is crucial for the development of the present thesis.

CHAPTER 3: DEVELOPMENT OF THE DATABASE

The database presented in this document was built based on the ABAQUS model developed on the previous research as mentioned in the chapter before. Most of the work done in this document was analyzing the requirements for the algorithm that is going to be trained and the limitation of the FE model, aiming to reduce the computational burden that implies the processing of such simulations.

3.1. Limitations of the Current FEA Model

Before the development of the algorithm used to build the database, the limitations of the model were tested experimentally using specific inputs of the automatic simulations code. These initial parameters determined the position and the length that was considered critical in certain scenarios that are going to be explained later in this section of the document. The limitations of the model were those situations where for any reason the process of simulation was not done properly.

Before speaking about the limitations of the model it is pertinent to speak first about the real objective of the creation of it. For the analysis of this helicopter panels we can divide the type of defects in two categories (just for the scope of this research): the first ones are skin cracks, as the name suggests are the defects that appear on the skin on the panel relatively far from the stringers, the second ones are rivet cracks, this type of defects appear when there is crack nucleation and propagation caused by the stress concentration on the notches generated by the permanent joints used between the stringers and the skin of the panel. Considering the previous classification, the model was built in order to analyze only the defects of the skin of the panel by creating cracks

on desired positions at a considerable distance from the stringers. Looking at it in terms of the inputs given to ABAQUS to build the model, partitions are done over the skin towards improving the mesh quality over the region of interest, however, the partitions change on the portions of the skin where the stringers are assembled, this difference between regions impedes the generation of cracks near the vicinity of the stringer. Figure 14 shows the partitions on the CAD model of the skin, notice how the grid done in the region where the stringers are assembled overlaps with the center portion of the skin dividing it in three smaller skin regions (refered as bays from now on), there is where the cracks are going to be analyzed.

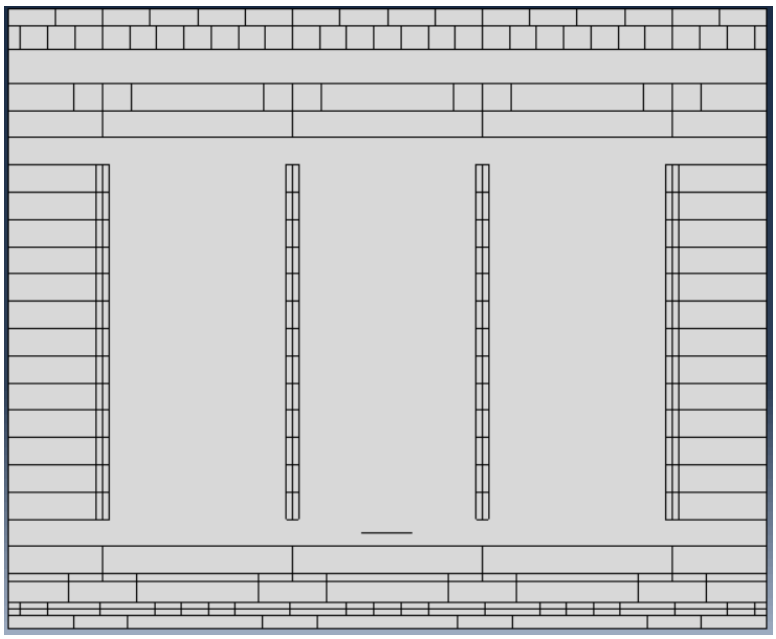

Figure 14. CAD model of the skin with partitions.

As a preliminary test a couple of cracks were located on both sides of the second stringer at distance of 10 mm with an initial length of 20 mm each, the idea was to understand how far the cracks can be placed from the stringer in order to have a proper simulation. The cracks were located as depicted in the figure below. Looking carefully at the figure 15 one can notice that the defects may not look symmetric with respect to the stringer, but this is due to the geometry of it, actually, both cracks were

located symmetrically with respect to the central line of the partition corresponding to the location of this element.

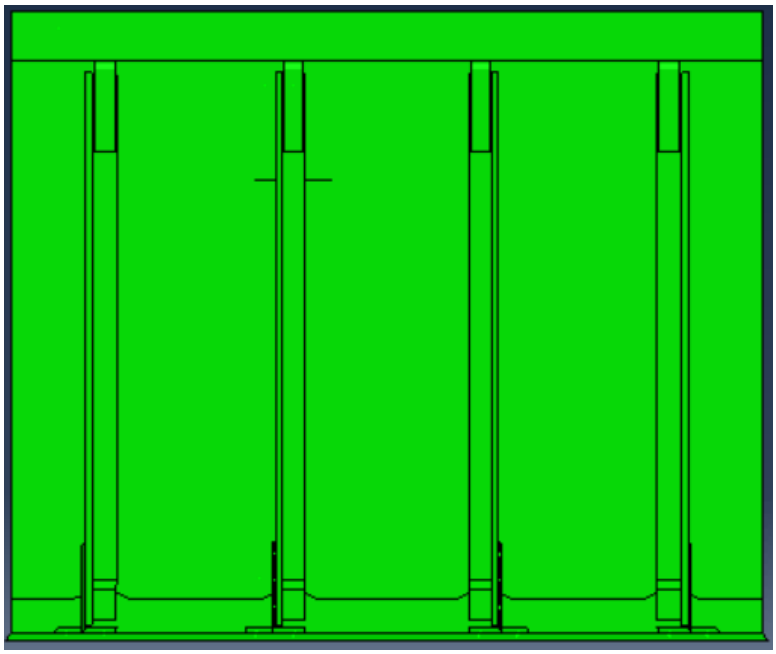

Figure 15. Symmetric simulation of two cracks of 20 mm under the stringer.

For the configuration explained before ran smoothly and coherent results were obtained, the next simulation was performed using cracks of 35 mm where one of the cracks was overlapping with the partition of the stringer and the other crack was left on an arbitrary position in the central skin, as shown in figure 16. For this case, the output of the simulation was abnormal, figure 16 shows the positions where the two cracks were located using the input data (highlighting in red the crack of interest) and presents the output obtained for those initial parameters chosen, notice how the position of the crack below the stringer changes and appears divided in two different parts in random locations in the panel. This unusual behavior of the model is mainly due to the partitions as it was explained before, most of the cracks that nucleate below the stingers are the product of defects generated while assembling the skin with the stringer by means of a rivet, but the distribution of the partitions in this specific model is intended to analyze only the effect of cracks in the skin.

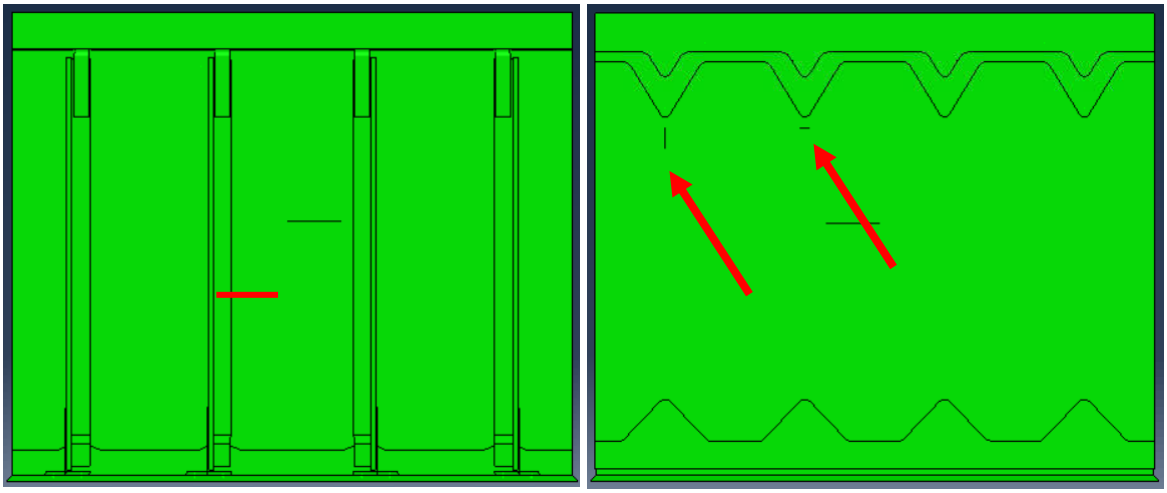

Figure 16. 35 mm crack simulation under the stringer.

This test allowed to set one the first parameters for the creation of the database, the starting and the finishing point in the horizontal direction, the positions of the cracks must start at least 10 mm away from the first stringer and finish at a same distance from the fourth stringer, on the following section of this chapter the process of choosing the boundaries is further explained.

The next limitation identified is related to the one explained before, that is the maximum crack size. Once again experimental tests were done placing one crack in the middle of the of the central skin panel changing its size until it reaches a critical length where the simulation ran with abnormalities. The crack length varied from 90 mm up to 150 mm in steps of 10 mm, the results indicate that for values above 130 mm the simulation's output present errors similar to the ones reported previously for the first limitation of the model. Figure 17 shows the simulation with a crack size of 130 mm, which corresponds to the approximation done for the first issue analyzed, where the minimum distance between the stringer and the crack is 10 mm. Considering that the distance between the vertical axis of the stringers is 150 mm, a maximum crack size of 130 mm is a reasonable value.

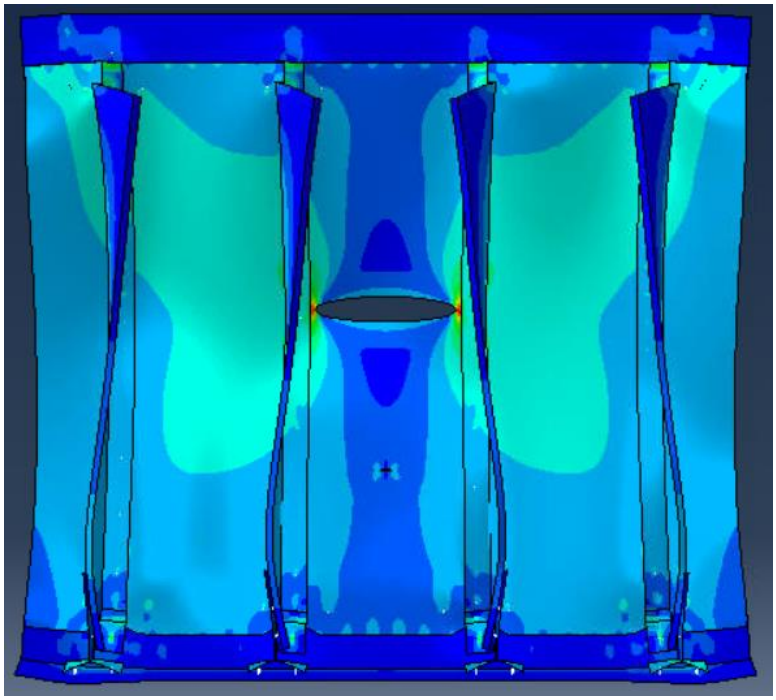

Figure 17. 130 mm central crack simulation results.

This parameter is relevant for the overall knowledge of the model, however this size of crack is never reached in the preliminary database, as presented in the next section of this chapter.

One of the most important limitations, that drove a significant part of the creation of the algorithm to generate the database, is the horizontal overlap of the cracks. When simulating thousands and thousands of possible positions where the cracks can appear there is a high probability of overlapping of those defects. One can expect that if two cracks close enough to reach horizontal overlap will eventually, after a certain extend of crack growth, merge with the closer defect in the surroundings. Nevertheless, figure 18 shows the actual behavior of the model when simulating two cracks of 10 mm with 5 mm of overlap between each other, see how the crack geometry changes, the length of both cracks reduces arbitrarily and the final output an error. The model done for this research is a representation of the case of the panel with defects, in order to keep it as simple as possible this tearing effect of the cracks is not considered, this type of features not only increases the complexity of the model but also increases the computational burden per simulation, that extrapolated to

thousands of configurations means high simulation time, that is a parameter that has to be reduced.

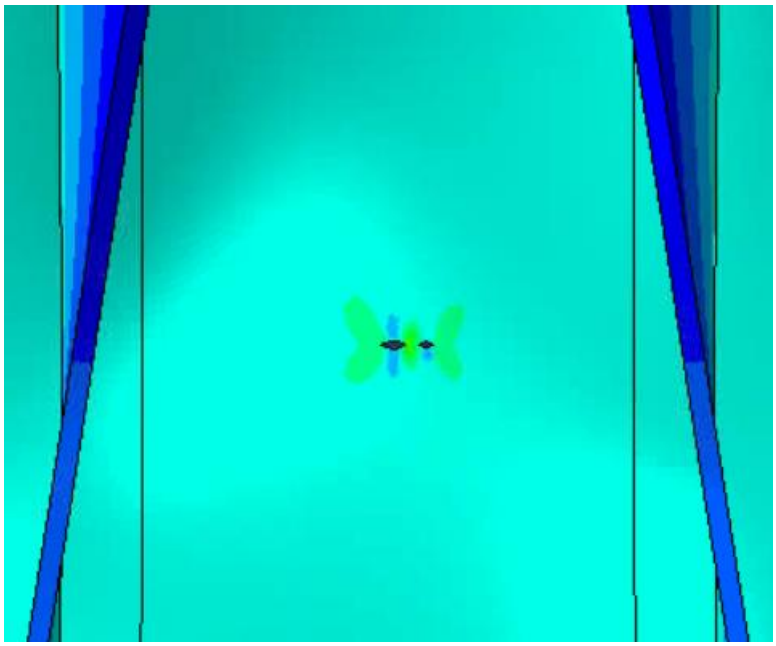

Figure 18. 10 mm cracks horizontal overlap.

Doing the same type of simulation with larger cracks results on a similar situation as the overlap with the stringers, once again, for cracks above 30 mm, after the simulation is done the position and the orientation of the cracks changes randomly and the results obtained are wrong. This limitation introduces a new constrain in the formulation of the algorithm and states the necessity of avoiding these scenarios that may end up in unreasonable data.

3.2. Development of the Algorithm

3.2.1 Basic Structure of the Algorithm

With the identification of the issues related to the model it is possible to start building the code that will rule the creation of the database considering the constraints mentioned before. First the area of interest was delimited, going back to chapter two, it was mentioned that currently the helicopter panel was instrumented with a set of

20 FBG sensors along the stringers. Originally, the distance from the first to the last sensor in the stringers determined de vertical limits for the interest area, while the horizontal boundaries, as mentioned before, were given by the first and fourth stringer. However, the FE model gives an insight of the location where the analysis must be done, taking a closer look to the features of the model it is possible to identify a free mesh where the defects should appear on the panel. Similar to the partitions, the size and the distribution of the mesh is done on purpose to refine the analysis on specific regions of the element, actually partitions are used as way to further improve locally the mesh on desired portions of an element. For this case, figure 19 illustrates the mesh of the overall CAD panel, notice how the central region of the skin is meshed differently from the rest of the component, a free mesh is used due to the presence of the cracks in arbitrary positions inside the skin that will change the mesh in each single simulation. Finally, considering the position of the sensors and the meshed area the simulation region was selected considering that the vertical boundary of the interest area is slightly further than the first and the last senor on each stringer, this is because there is a portion of the skin above and below the first and the last sensor, respectively. Figure 20 shows the position of the sensor in the real panel, highlighted in purple is the interest area where the analysis was done, as mentioned before and in correspondence with the FE model the area where the cracks can appear is delimited horizontally by the stringers and vertically by the boundary of the free mesh in the skin.

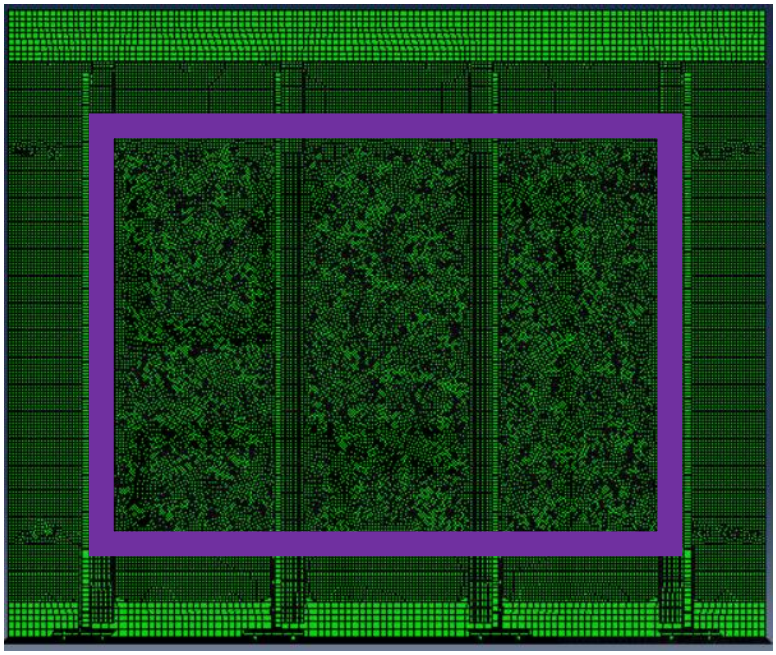

Figure 19. Mesh structure for the panel FE model.

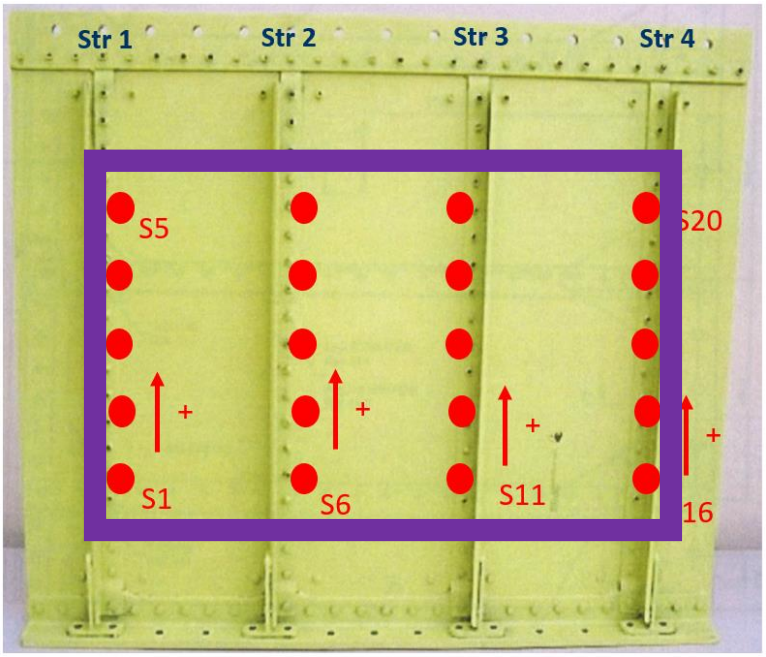

Figure 20. FBG sensor location and simulation area. [21]

Now that the area of interest is well defined, the following step was to decide how the positions of the cracks were going to change in time. On previous research regarding this matter a single crack moved along the panel by changing its position randomly and then evaluating whether this position is feasible or not, for this thesis the approach was different, the cracks start moving following a predefined pattern that is set up by

initial parameters in a code developed in Matlab. In principle the cracks will change their position with respect to a step in the horizontal direction and a step in the vertical direction; by means of for-loops the position and the size of the cracks change in an organized manner, this approach allows the control of undesired positions of the cracks (as its presented on the following section of this chapter), also if a sufficiently small step is applied in in both directions it is possible to simulate approximately all the possible relevant positions inside the interest area (consider that the number of positions are infinite, theoretically speaking).

Figure 21 is a schematic example of how the positions of the crack move according to the iteration loops in the code, the origin of the crack follows the vertical direction (Y direction) a number of times equal to the number of steps in Y, the same logic applies for the horizontal direction (X direction). The for-loop is designed for the origin of the crack to move through all the position in Y before going to the next position in X, once all the position are covered the cycle starts again but this time with a larger crack size.

This is the principle for a single crack, the same applies for the second crack, the only difference relies on the fact that for each cycle of on the first crack the second crack (called stationary crack for the sake of simplicity) only moves one position, in this way it is possible to simulate all the possible combinations between each other.

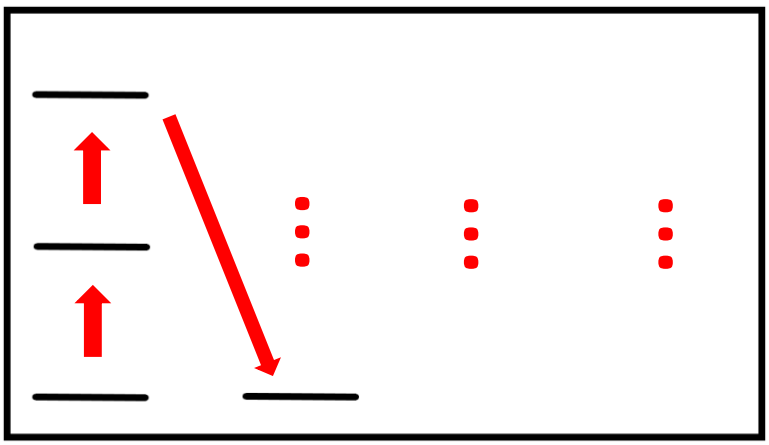

Figure 21. Crack's path according to the nested for-loop.

The cycle inputs were built as follows:

1) A vector containing the horizontal boundaries is created.
2) The number of elements of the vector of horizontal positions is determined by a step in X created as a multiple of 3 (considering that the skin of the panel is divided in three sub-skin bays).
3) A vector containing the vertical boundaries is created.
4) The number of elements of the vector of vertical positions is determined by a step in Y that may change with respect to the critical distance between the cracks (this is further explained in the following sections).
5) A vector containing the initial and the final crack size is created.
6) The number of elements of the vector of crack lengths is determined by a step L which determines the number of changes in the crack size on each cycle.

Once the input vectors and steps are created they are integrated in a set of nested for-loops as mentioned before, the inputs have to be set for each of the cracks independently giving the user the freedom to pick where the position of the cracks are going to be located, however, for the sake of simplicity the preliminary database was built using exactly the same steps in all directions and sizes so that the cracks will follow the same path in all the cycles. This setup was intended to facilitate the analysis of the functioning of the algorithm, also, it helped to identify the main surplus simulations that should be eliminated. Once the code runs the cycle starts with the inner loop in the set of nested for-loops, as typically works for this type of processes, that corresponds with the position Y of the first crack, only until the whole cycle for the first crack is done the stationary crack moves to the next position. After the first cycle is done for both cracks the size of the first defect increases and the process starts all over again.

This algorithm is built in order to obtain the higher number of cases possible to supply enough information to the ANN algorithm to be able to detect the and localize the defects, nevertheless the highest the number of simulations ran, the higher the computational burden and the time required to obtain this information, that is why

there are some features and limitations of the model that aid the reduction of cases, hence reducing the overall simulation time.

3.2.2 Panel Symmetries

Starting from the physical features of the panel, it is possible to exploit the symmetries of the geometry of the component to avoid the repetition of data. Taking a closer look to figure 22, one can identify the symmetry of the element with respect to the vertical axis, this condition gives the possibility to skip several simulations, replacing the existing ones (for example in the first half of the panel) and mirroring the results to the opposite half of the component. This is feasible only if the component is clearly symmetric, otherwise some test must be run to prove this statement.

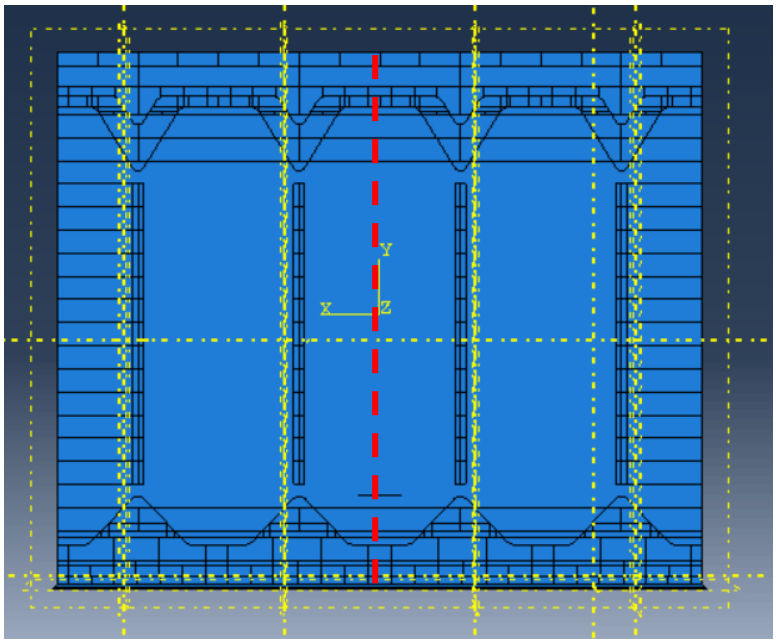

Figure 22. Panel's symmetry with respect to the vertical axis.

To start applying this symmetric condition to the cycle inside the code, first the cases where the symmetry is involved should be identified. For this, the cases are divided in two subgroups: the first one when the crack one ($L_1 = L_2$) and the second one when ($L_1 \neq L_2$).

I. $L_1 = L_2$

Suppose the cycle just started, the inputs of both cracks are the same and both cracks have the save the same length.

For this condition, the following cases are considered as repetitions and should be avoided.

1) One crack on symmetric positions with respect to the vertical axis.

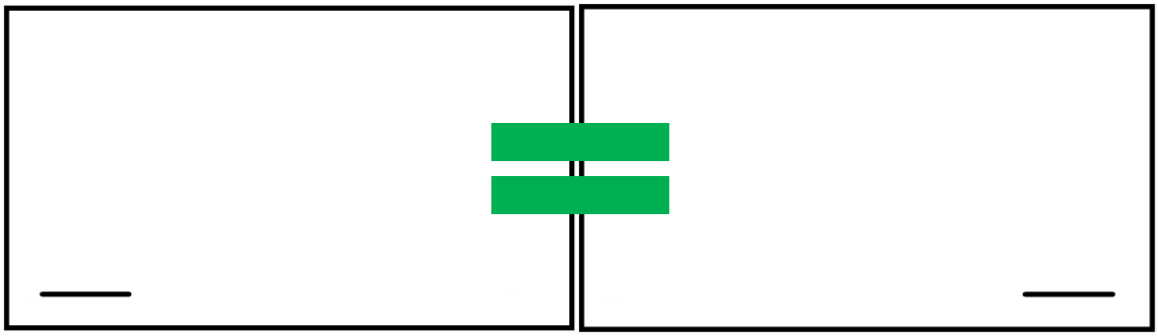

Figure 23. Single crack symmetric simulation example.

Figure 23 is a clear example of a situation where the simulation of one position (for example the one on the left side) can be used to mirror the results to represent the position in the opposite side of the panel. Clearly this case must be complemented with the second defect, still the same logic applies for a pair of cracks with symmetric positions with respect to the vertical axis.

2) Cracks with switched positions between each other.

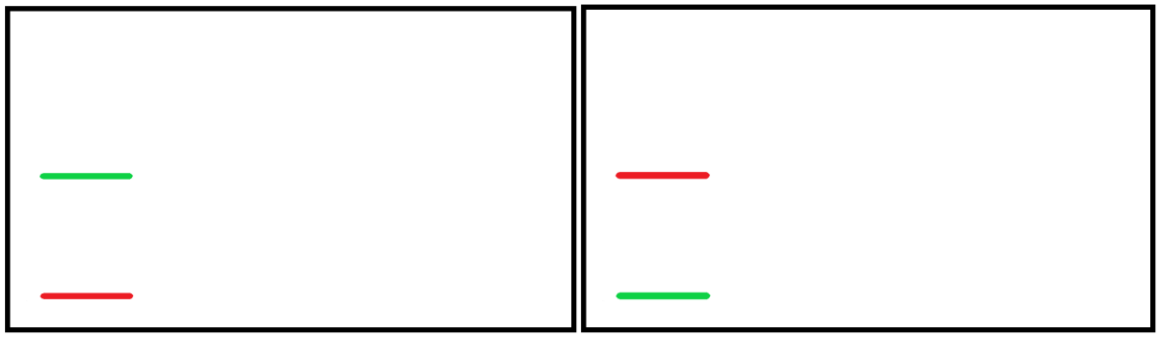

Figure 24. Equivalent simulations with switched positions example.

Figure 24 on the left side depicts the scenario where crack one (red) is on the first position of the cycle and crack two (green) is on the following position, on the other hand, the right side shows the opposite scenario. Considering that both cracks have the same length it is correct to assume that the case on the left and the case on the right are the same, this applies for all the iterations after the second position of the static crack.

II. $L_1 \neq L_2$

Now suppose that the first cycle of the static crack is done and the first change in crack size is set for the moving crack.

For this condition, the following case is considered as repetitions and should be avoided.

3) Two cracks on symmetric positions with respect to the vertical axis

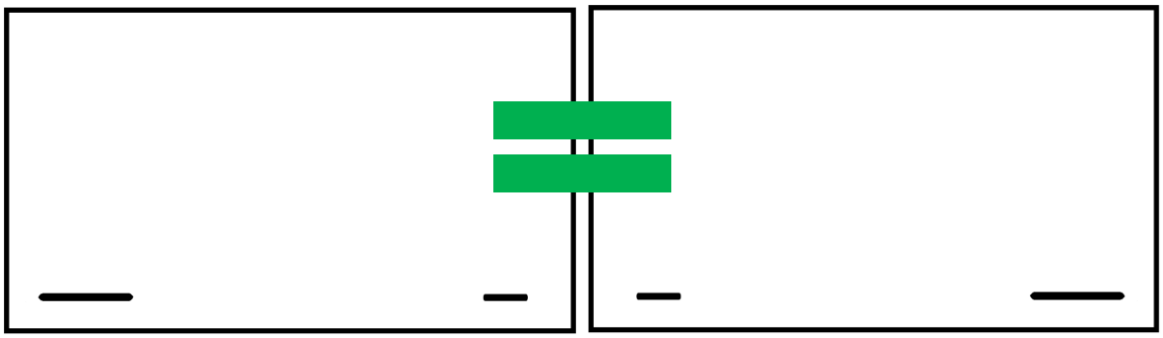

Figure 25. Double crack symmetric simulation example.

Similar to the case in $L_1 = L_2$, the symmetry with respect to the vertical axis for cracks with different lengths generates mirrored simulations which can be avoided, the crack sizes may be coherent in both cases in order to reject this simulation. Figure 25 shows one case where this condition applies, the results on the left side can be used to represent the positions on the right side as in the case of one crack.

In view of the previous mentioned cases, some procedures were developed to solve these issues; the fact that the cycle is built upon a set of nested for-loops made the implementation of these solving procedures not trivial. The idea was to introduce this

portion of the code without deleting useful information from needed simulations, for that, a set of conditional statements were inserted inside the for-loops that were able to neglect the repetitive cases. The conditions used depend on whether the length of the cracks is the same or not, the following figures show the methods used to solve the repetition cases, but, before going to the solutions it is important to look back to Chapter 2 where the database for a single crack is created. This existing information is crucial for the development of the further solutions because these data may be useful in the time of suppressing repetitive simulations, even in the case of the panel with two cracks. Simply explained, one can use the data from two simulations of single cracks to represent a one simulation of two cracks on the corresponding positions depending on the distance between the cracks (this is further explained in the following section of this chapter), an example of this is shown in figure 26. Notice how the addition of the case in the left side plus the case in the right side is equal can represent the case below them. This means that there some scenarios that can be avoided in the two-crack simulation because they can be done by the superposition of the effects of two single cracks.

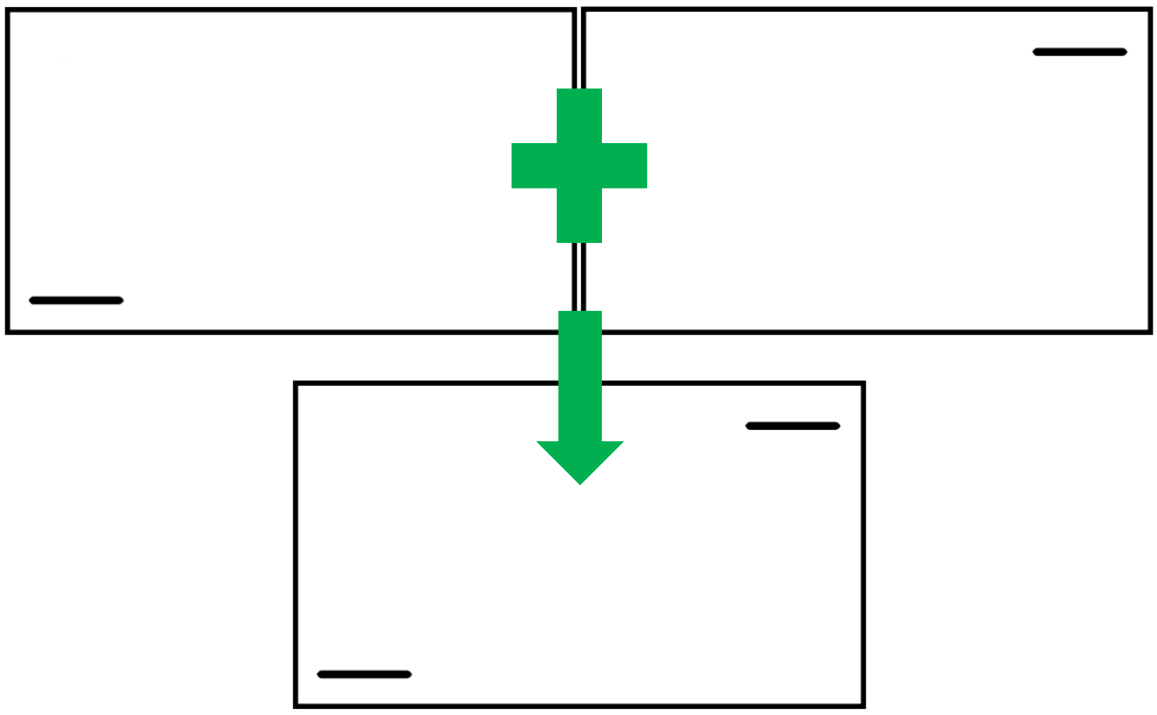

Figure 26. Equivalent simulations by linear superposition principle, example.

Having clear the influence of the previous research done with single cracks, it was possible to develop the methods to solve the undesired scenarios mentioned on the previous paragraph as follows:

1) Accounting for the symmetry in the condition $L_1 = L_2$ all the relevant positions can be covered in just one half of the panel, all the simulations done here can be mirrored in the other half of the component or they can be done by superposition of single-crack simulations. So, for this first case it is necessary to run just the positions that correspond to the left side of the panel.
2) Complementing the solution in point 1 the better way to avoid the undesired case number 2 is by applying a conditional statement where the position of the moving crack has to be always one position after the one of the stationary crack. Normally in the unconstrained cycle the moving crack will appear in position one once it has done the whole path, however, to avoid the switching repetition it is required that the moving crack be always ahead of the one that remains static. Speaking in terms of the structure of the code if the vector of vertical positions contains a number of elements i = (1, 2, …, n), the position Y on the stationary crack goes from y_1 to y_{n-1} while for the moving crack it goes from y_{sc} to y_n, where SC stands for stationary crack. Figure 27 gives a schematic example of the solution implemented for the condition $L_1 = L_2$.

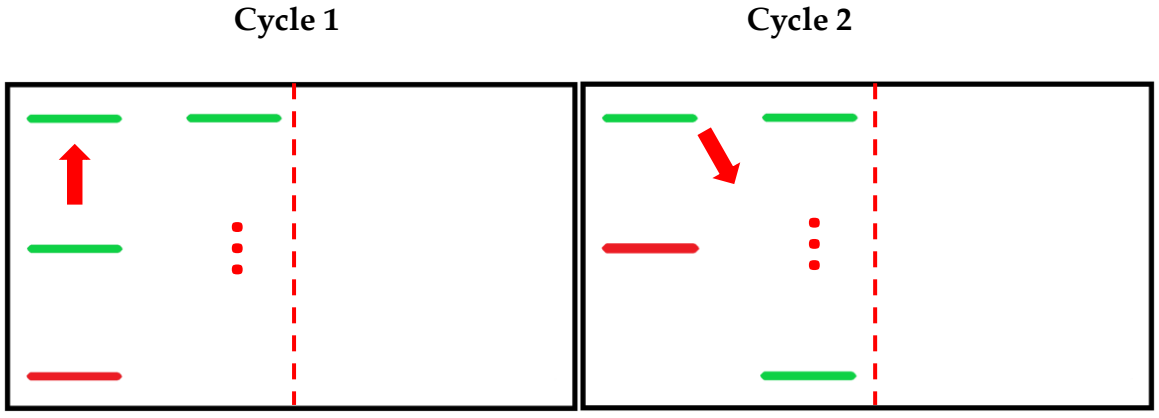

Figure 27. Cracks' path for $L_1 = L_2$.

As mentioned on the previous paragraph, the position of the green crack (moving crack) is always ahead of the red one, the cycle finishes on the top right corner of the left half of the panel where the symmetry line is presented as a dashed line, the other side of the panel remains blank as the simulations of this side can be mirrored form the data obtained in the left side of the component.

3) Similar to the case of equal lengths, a lot of simulations can be avoided by utilizing the existing data of the single cracks, however, for the case $L_1 \neq L_2$ the number of cases that require the presence of two cracks increases significantly. For this condition, the there is no switching repetition so the cracks must start on the very first position at the beginning of each cycle independent from the position of the other one, also, to have a more complete database one of the cracks must cover the positions on the right side of the panel to fill all the possible combinations that will allow a proper training of the ANN. Figure 28 shows the procedure used for cracks with different lengths, notice how the stationary crack (red) remains only in the left half of the panel while the moving crack (green) travel along the whole surface of the component. One can observe that at the end of each cycle of the moving crack, the static crack moves one position forward, but the moving one can start at any iteration before the red one.

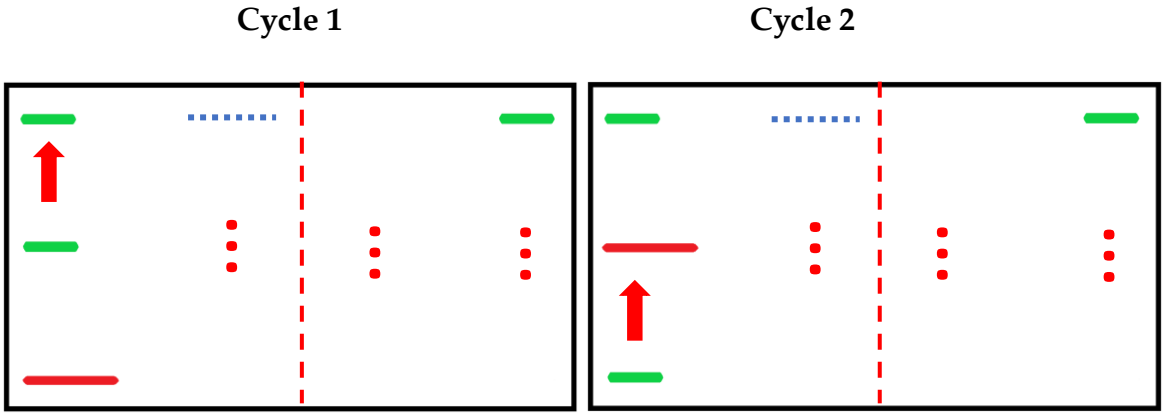

Figure 28. Cracks' path for $L_1 \neq L_2$.

As explained of the solution of point 3) the green defect is able to move along the whole component while the stationary crack will reach its last position over the blue dashed line, this configuration accounts for all the possible combinations for all crack sizes.

Unlike the previous geometric feature, the symmetry with respect of the horizontal axis in not clear due to the assembly of all the components of the panel, the skin itself can be considered as symmetric in all directions, however, once it is assembled with the rest of the elements the overall geometry of the component changes. Certainly, the symmetry should favor the interest area where the cracks are positioned, this way even more simulations can be suppressed, nonetheless this condition is not clearly visible, for this reason some test must be done to prove it. The test done is a simple set of simulations where the cracks are placed first on the top part of the panel and then on the bottom part exactly at the same distance with respect to the origin of the interest area, both tests are shown in figure 29.

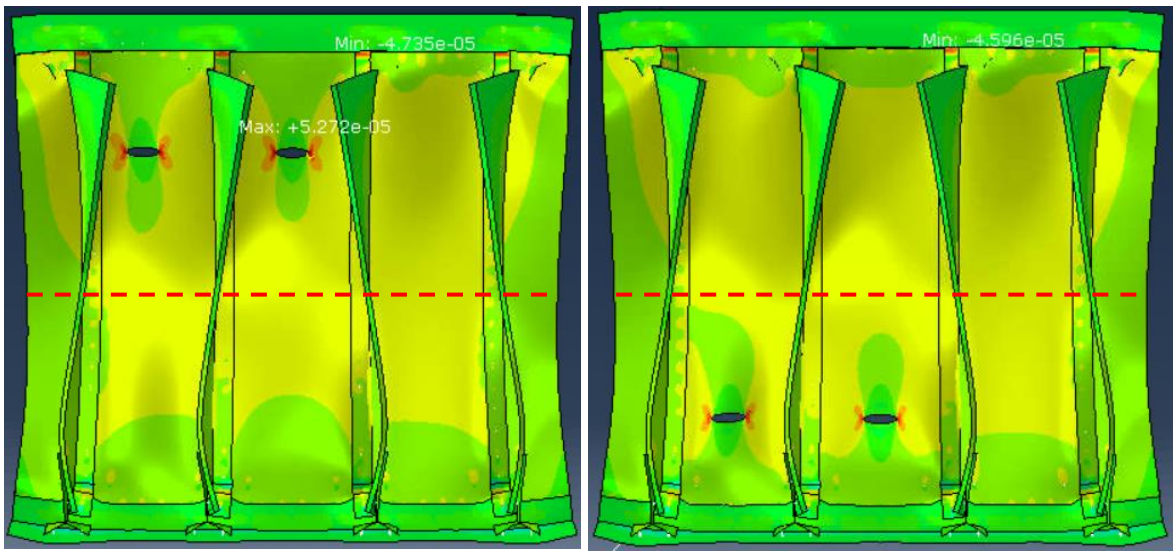

Figure 29. Test for panel's symmetry along the horizontal axis.

After running the simulations one can observe the strain fields (E22) plotted as the countours over the component, notice that the regions below and above the top and bottom cracks (plotted as the light green region), respectively, are not equivalent between them, this gives the first hint that there is no symmetry over the horizontal

axis. However, in order to verify a path is placed in both panels but with opposite directions, by doing this it is possible to compare the actual data in both cases and see if the data fit in both cases to consider the component as symmetric with respect the horizontal axis.

Figure 30 illustrates how the paths are placed over the panel in both simulations, on the left side the panel, the simulation with two top cracks is represented with a path that goes from bottom to top on the first stringer, on the other hand, the right panel with tow bottom cracks has a path going in the opposte direction on the same stringer as well.

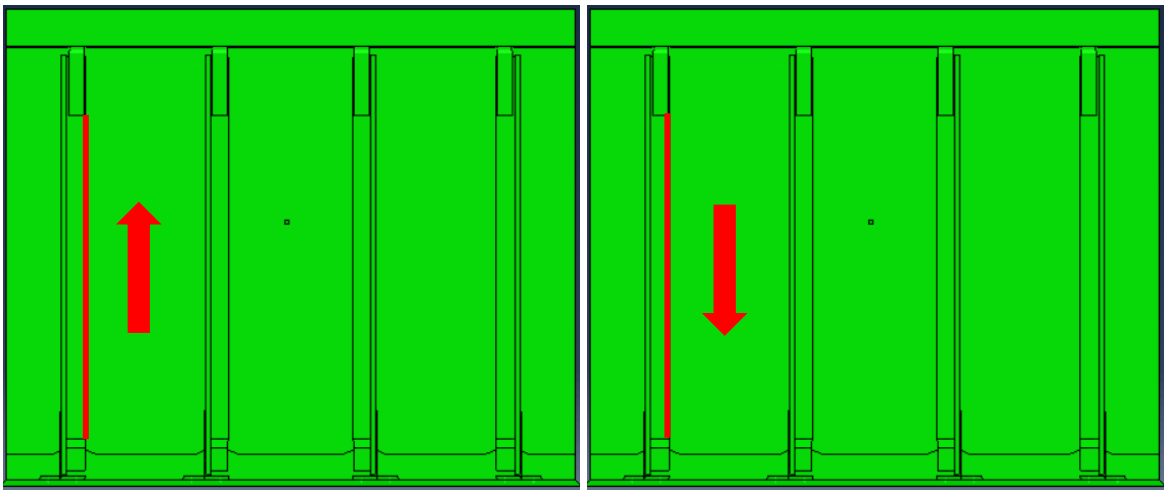

Figure 30. Paths direction for horizontal axis symmetry test.

After placing the paths the data is collected and plotted as shown in figure 31, at first the trend of both simulations is similar but mirrored over the vertical axis, however, if the data is arranged correctly one can notice that the trends do not overlap between them, as shown in figure 32, in theory if the interest area is symmetric both simulations will result in the same trend of strains. Considering the tests done it was possible to determine that the component is not symmetric for this axis, the arrengement of the stringers and the other components make the whole system asymmetric over the horizontal axis. Hence, there is no chance to exploit this geometric feature to reduce the number of simulations even further, all the positions from the bottom boundary to the top boundary should be covered.

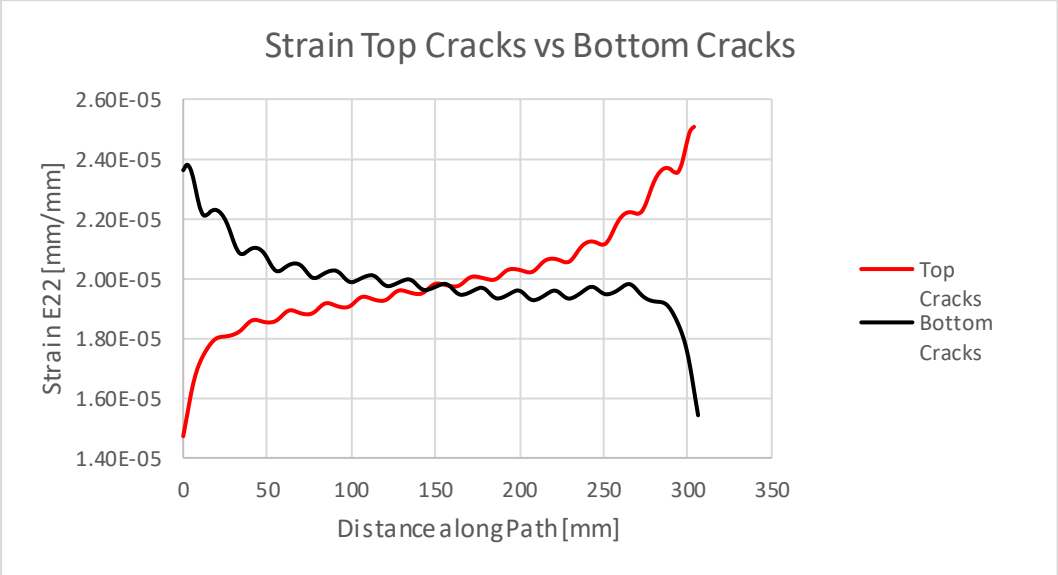

Figure 31. Strain E22 vs distance for the opposite direction paths.

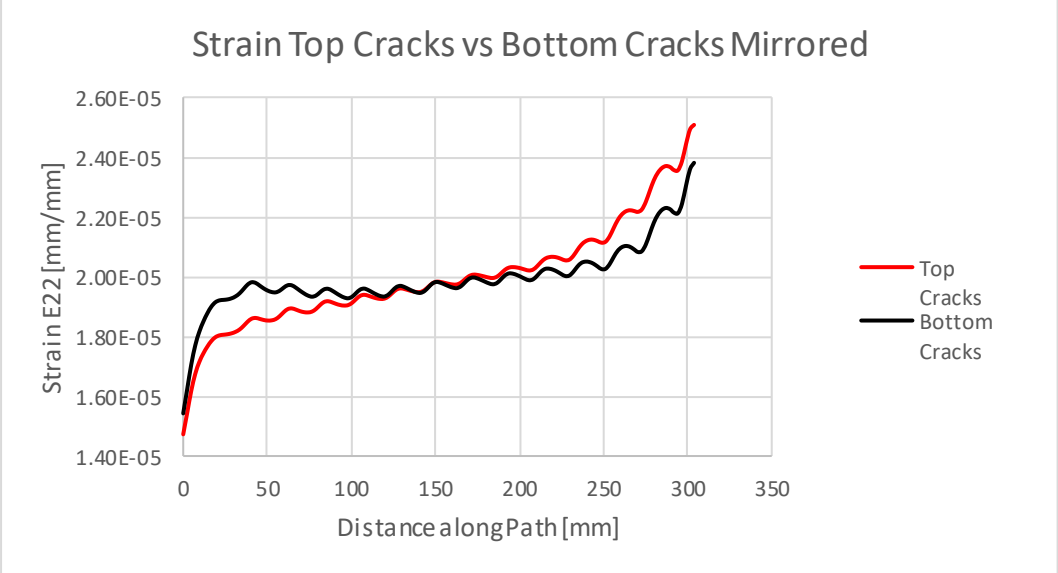

Figure 32. Strain E22 vs distance for equal direction paths.

3.2.3 Crack Overlap

Some limitation of the model may be used as conditions to reduce the number of simulations, as explained in section 3.2.1, the horizontal overlap of the crack causes the simulation to run with abnormal results that are no feasible for training the ANN. In order to solve these undesired scenarios, it is necessary to set conditional statements which allows the software to skip the simulations where an overlap between the cracks is detected.

There are two main cases where an overlap can occur:

1) Complete Overlap

This is the case where the centers of the cracks overlap perfectly, this are obvious scenarios that should be avoided but the FEM is not able to discard them itself, it is necessary to avoid these iterations in the for-loops before introducing them in these positions in the simulation function. Figure 33 shows the green crack over the black one aligned exactly on the center, this condition generates errors on the FE model, but it is easily solved by applying an if-statement where the simulation is skipped when the position X and Y of the cracks coincide for both defects. Whenever this case is identified the code automatically discards it and continues with the following iterations.

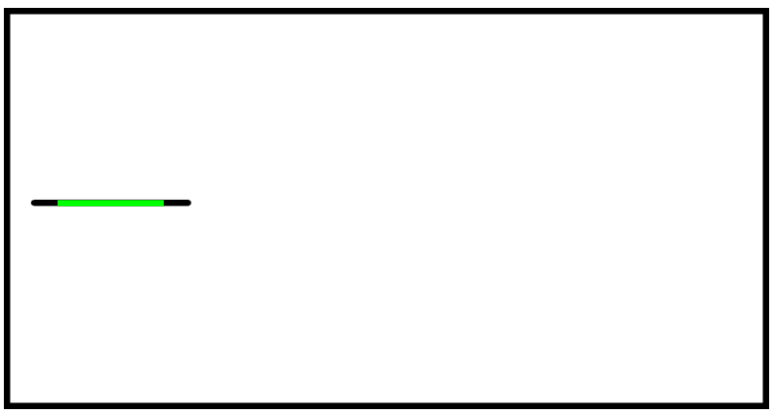

Figure 33. Cracks complete overlap example.

2) Partial Horizontal Overlap

Compared to the previous case, there are situations where two cracks are close together on the horizontal direction and upon loading both cracks start to grow. Once the crack extension is above a threshold both defects tend to merge into a bigger one, however, the FEM is not set to recognize this merging action and the output is an abnormal simulation, as explained on section 3.1. Taking into account the limitations of the model for this scenario it is required that the algorithm avoid the iterations where there is any type of horizontal overlap. Considering that the model used for this training does not account for the growth of the defects on time, the only positions to be suppressed are related to the actual length of the crack in the input data, as shown in figure 34. Notice how there is a clear overlap between the green and the black defect, nevertheless, the centers are not aligned as in case 1), therefore the strategy to solve this issue must be addressed differently. For partial overlap of the defects, recall section 2.3 where the creation of the crack is explained, basically the position (X, Y) of the defect represents the origin from where the crack is deployed, this origin is located on the right-hand side of the defect, subsequently the size of the crack is added on from left to right until it reaches the length given on the input data. Even if the origins do not overlap at first there is a chance that once the length is introduced both cracks may intercept in any point between them. Suppose that the origin of the black defect on figure 34 is given by the coordinates (x_1, y_1) and the origin of the green defect by (x_2, y_2) with crack lengths equal to L_1 and L_2 respectively, in order to have horizontal overlap in the undeformed condition the vertical positions must be equal ($y_1 = y_2$), if this condition is fulfilled this parameter can be neglected and the overlap of the defects will depend only on the horizontal position plus the length of each corresponding crack.

Figure 34. Cracks partial horizontal overlap example.

Now suppose a situation similar to the one depicted in figure 34, as mentioned before there is a clear interception between both defects, if this scenario is further analyzed as shown in figure 35 (the cracks are separated vertically to show their dimensions properly, still they behave as figure 34, this representation is just for explanation purposes), one can identify a condition where all the possible positions can be detected. For example, in the chosen positions the origin of crack 2 (green) is somewhere in the set of points inside $x_1 + L_1$, thus if all the positions inside this range can be skipped it would be possible to prevent any horizontal overlap of the cracks. For this, another conditional if-statement is created where the iterations with points of the origin of the second crack inside the interval ($x_1, x_1 + L_1$) are suppressed and the same applies in case where crack 2 (green) is the one on the left-hand side. The condition to avoid these iterations goes as follows:

$$\text{if} \rightarrow (x_1 < x_2)\ \&\ (x_2 < x_1 + L_1) \rightarrow \text{Continue}$$

Where the "continue" function suppresses the undesired iteration and continues with the following step in the for-loop.

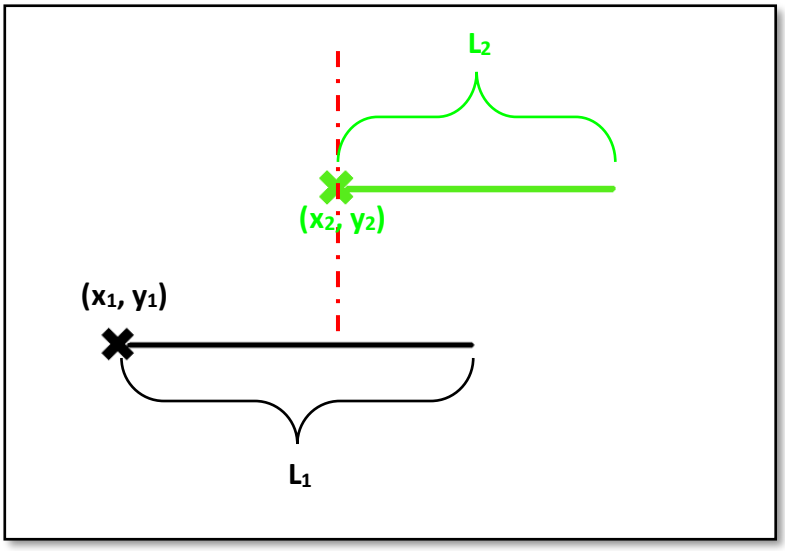

Figure 35. Details on cracks partial horizontal overlap.

With all the undesired conditions regarding the limitations of the model eliminated, the following aspect to be considered for the creation of the database is the mechanical features or limitations of the problem to be addressed. This will be exposed on the following section of this chapter.

3.3. Critical Distance Between Cracks

To reduce the complexity of the model it is assumed that the effect of the crack does not lead to yielding of the component in any point of the loading process, this not only simplifies the analysis but benefits the creation of the database in further reduction of the number of simulations. For the case of a single crack the model is always linear and normally it does not present any eventuality unless is close to the stringers or any boundary of the simulation area which, as explained on section 3.1.1, will end up on an unfeasible simulation.

On the other hand, when the second defect is added to the model, the interaction between them may result on a nonlinear behavior, speaking in terms of the global strain field of the panel. This interaction also suggests that a single crack model is not enough to replicate a real case where the component is damaged by two cracks at the

same time, since the linear superposition principle does not apply when the local strain fields of one crack is affected by the local strain field of the other one, thus affecting the global strain field of the system. The interaction between the defects depends on the relative distance between them, after a given value the effect of the local strain field of one crack does not influence the contribution of the second one, this distance is called from now on as critical distance.

Tests were performed to visualize the consequence of the linear superposition of cracks of 20 mm and 40 mm, in both cases the cracks were separated by 10 mm in the vertical direction, four simulations were done per each test, one for the undamaged panel, one for the panel with two cracks, and two with the single crack panels in the corresponding positions, figure 36 and 37 show the configurations for the damaged cases analyzed with the path used to gather the useful data for the 20 mm cracks. The data obtained in the healthy component is subtracted to the one of the damaged panels to account only for the contribution of the defects. The output data of each simulation is plotted to compare the actual simulation with the replica done by linear superposition (summation of the data of the single crack simulations).

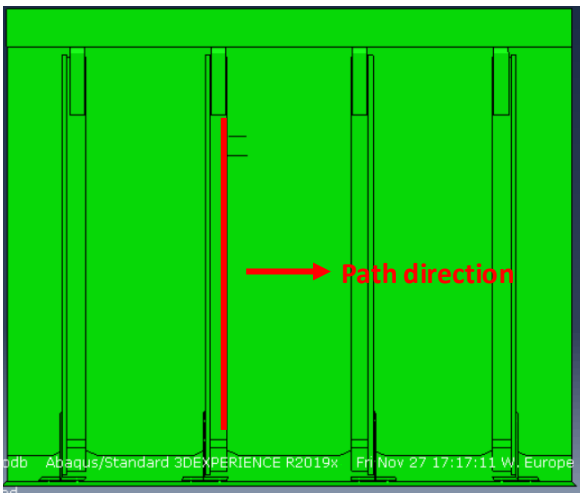

Figure 36. Test configuration for critical distance test.

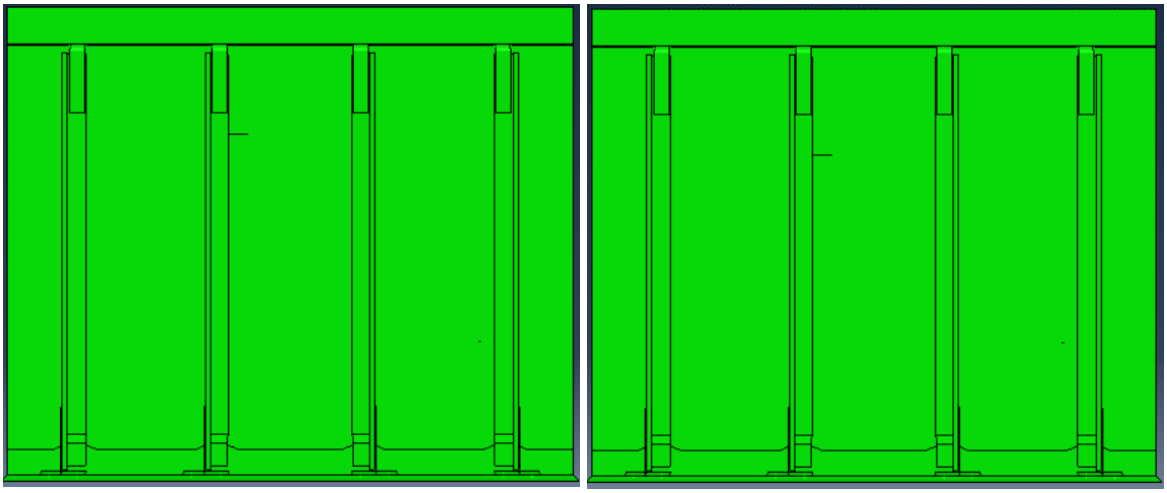

Figure 37. Simulation of single cracks for critical distance test.

Figure 38 shows the stress for 20 mm cracks, stresses are used because the trend is the same that the one for strains, the black line "Top-U" refers to the single crack simulation for the top defect minus the undamaged contribution, the blue dashed line with x markers "Bottom-U" to the other single simulation in the same conditions, the magenta line represents the simulation of the single crack simulations summed up by linear superposition while the cyan dashed line corresponds to the data of the simulation with two cracks. Clearly the maximum stress obtained by linear superposition of the top and the bottom crack is significantly higher than the one from the simulation with two cracks, this suggests that the linear superposition principle is not a feasible option for simulations where the cracks are placed close, below the value of the critical distance. Additionally, the analysis of 40 mm cracks presented on figure 38 shows that for the same distance between cracks there is a strong influence of the crack length on the error while using linear superposition, this is due to the reach that the local stress field of each defect has, which certainly increases as the length of the crack becomes larger.

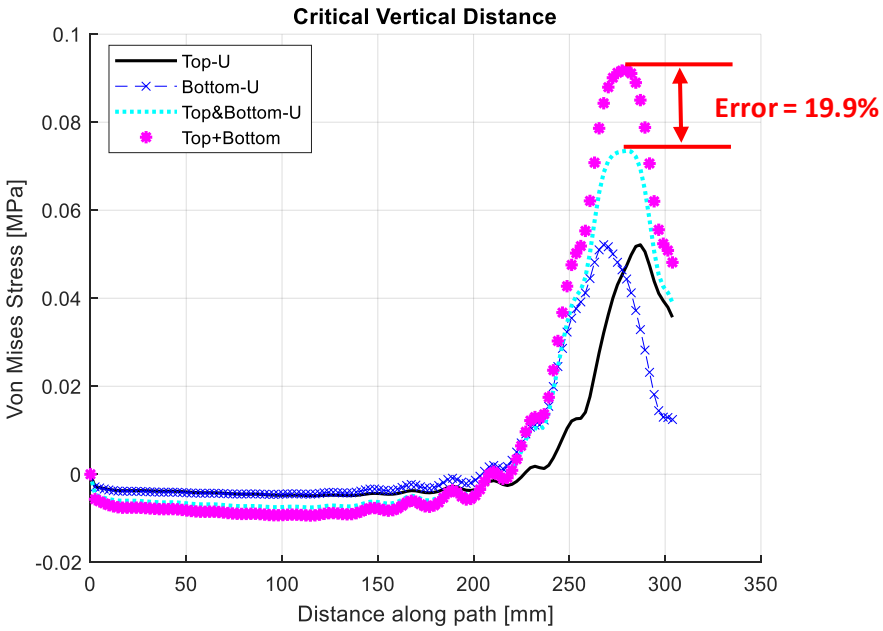

Figure 38. Von Mises stress vs distances for 20 mm cracks.

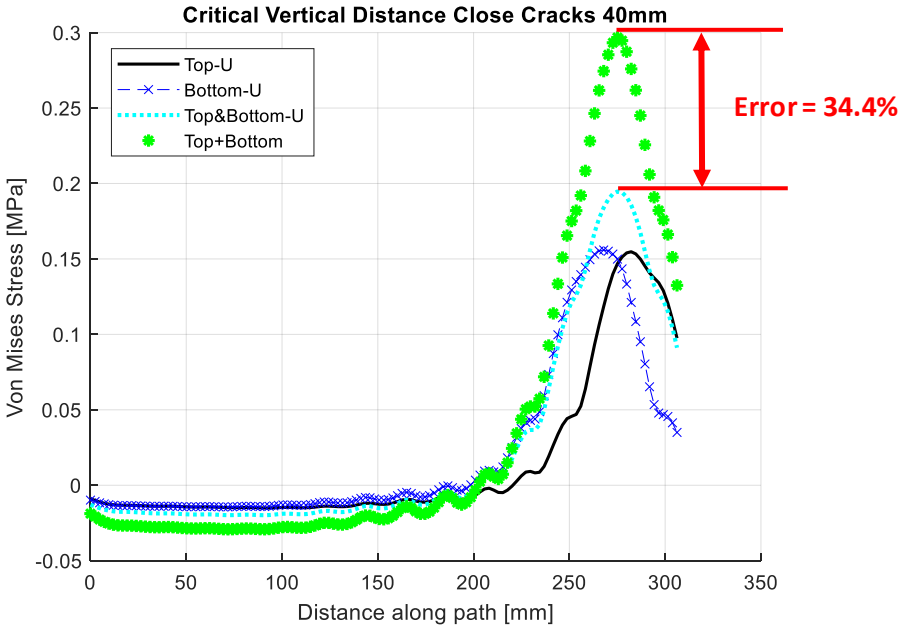

Figure 39. Von Mises stress vs distances for 40 mm cracks.

Figure 40 shows the comparison between 20 mm and 40 mm cracks with more detail, as it was mentioned before the error associated to the linear superposition of the effects of the cracks increases drastically with the crack size. Taking this into account, the critical distance must be computed for each length of the crack, to do so, one must understand how the local stress/strain field of a crack changes when increasing its size.

To illustrate the range of the stress/strain field a set of simulations were done where a single crack was placed static in a random position over the central skin panel while the length increased in each iteration. Figure 41 shows the configuration used for the test where the crack of interest had a variable length that changed from 20 mm to 50 mm with a step of 10 mm.

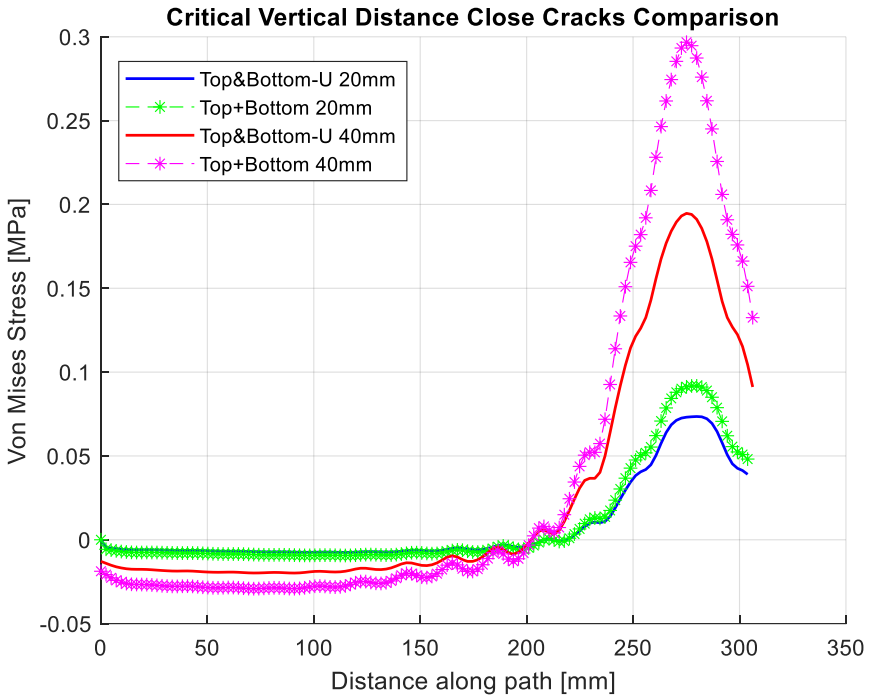

Figure 40. Von Mises stress vs distances comparison between 40 mm and 20 mm cracks.

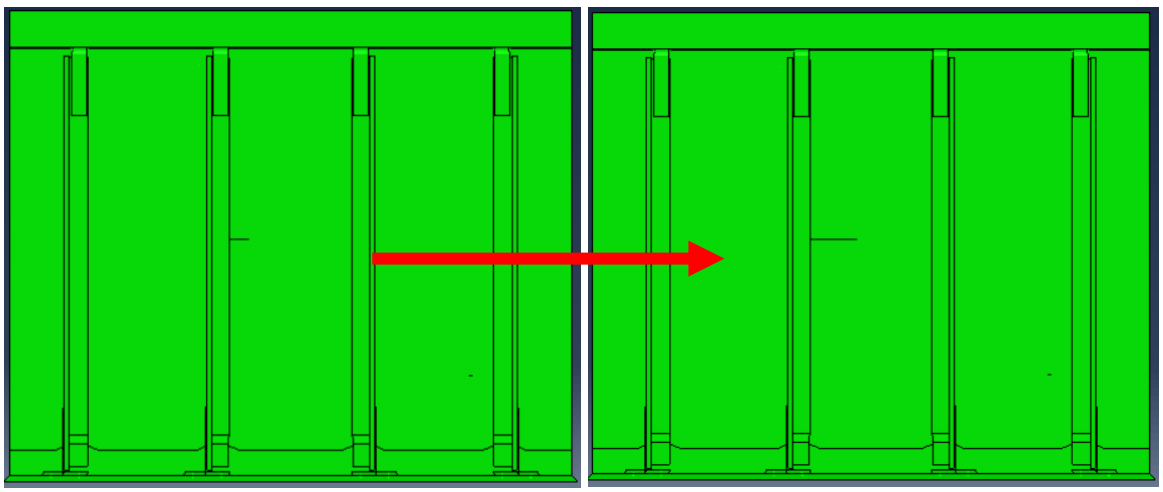

Figure 41. Test configuration for affected zone test.

Figure 42 shows the difference in the affected zone, which is related to the range of the local stress/strain field, depending on the size of the defect, this allows the interaction of bigger cracks in further distances compared to smaller cracks. A model must be created to define the critical distance between cracks depending on the size and the algorithm should be able to identify which simulations are needed to be introduced in the database that are not replicable by linear superposition of single cracks from previous databases.

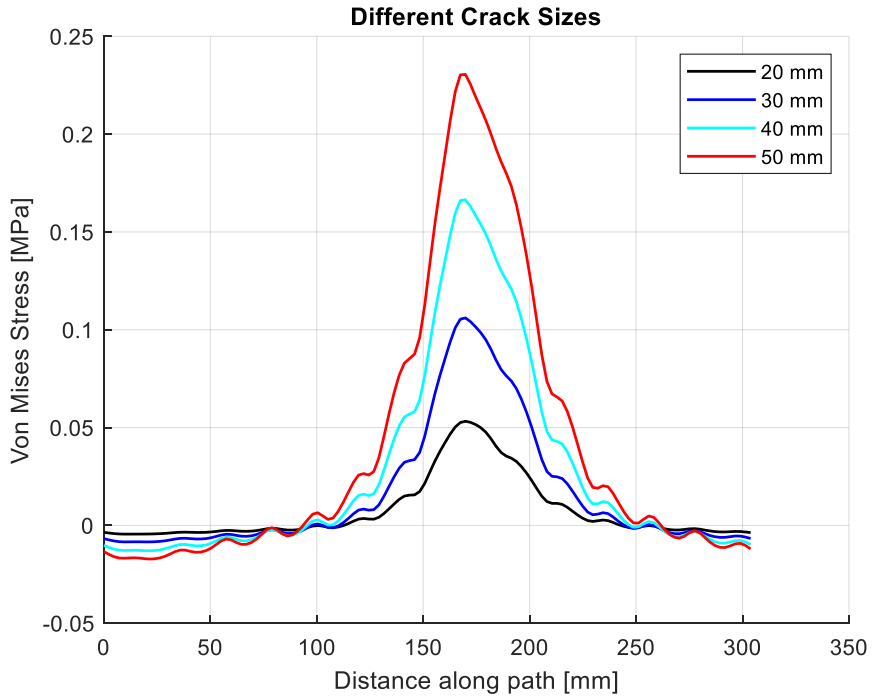

Figure 42. Von Mises stress vs distances for 20 - 50 mm cracks.

The importance of defining the critical distance relies on determining the number of simulations that must be done with the presence of two cracks. Once again, having the database for a single crack allows the replication of two-crack simulations only when the linear superposition principle is valid, i.e., when the distance between the cracks is higher than the critical distance, which, in principle, may be used to reduce the number of simulations reducing the computing time of the two-crack database.

To compute the critical distance without further increasing the complexity of the model, an experimental procedure was developed where the position of one crack

remained stationary while the second one was moving away increasing the vertical distance with respect to the static defect. Six different configurations were simulated as shown in figure 43 in order to test the influence of the distance in the stress/strain field of the component, moreover, to consider only the contribution of the damage, a simulation containing no defects was done as in the test shown before for the case of 20 mm cracks, this healthy condition shows the effect of the exerted load only in the panel without any type of notch effect or stress intensification. Subsequently, the contribution of the healthy panel is subtracted from the one of the damaged panel resulting only in the effect of the defects on the component; with this data is possible to compare the case in which both cracks are present in the panel and the one where the two-crack simulation is replicated with the data of two single cracks in the same positions. This way one can understand the error associated to the linear superpositions of crack in distances below the threshold.

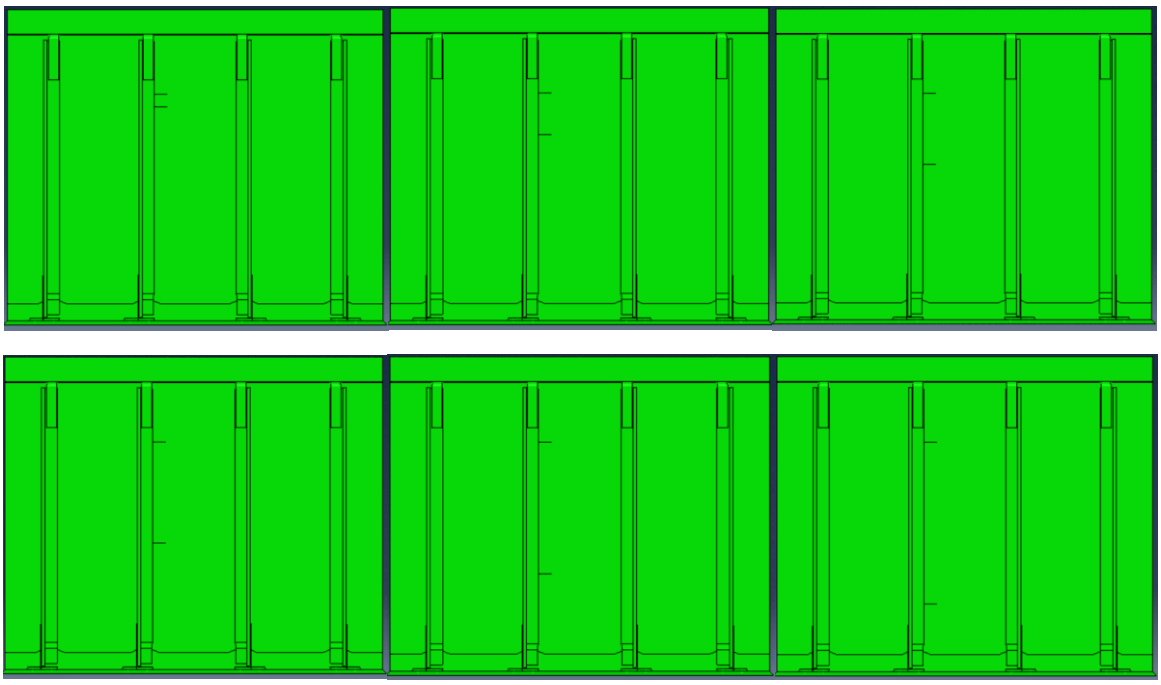

Figure 43. Test configuration for the critical distance test for different crack lenghts.

Initially, the test was done for two cracks of length equals to 20 mm, the results of the simulations are plotted on figure 44, each peak represents a different simulation where all the peaks at the end of the graph represents the static crack while the peaks moving

along the axis represent the defect that is changing its position on each iteration. As shown on the previous tests the peaks from the simulations where the positions of the cracks are close (blue peaks) have higher relative error when comparing the actual simulation with the replica done by linear superposition, however, notice that for the case of 20 mm cracks from the second peaks (red peaks) on this error tends to decrease up to the point where it becomes lower than 1%.

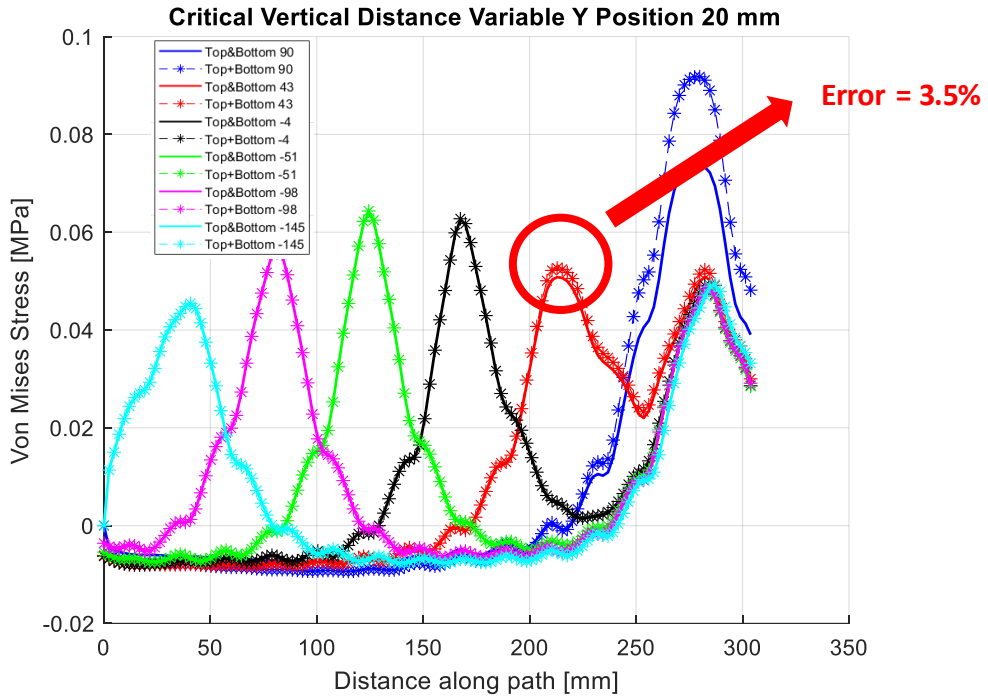

Figure 44. Von Mises stress vs distances for the critical distance test of 20 mm cracks.

As a first approximation, a threshold of 3.5% of error was set to consider linear superposition as a feasible solution to replicate a two-crack simulation, for the first set of cracks of 20 mm the critical distance between cracks corresponds to 47 mm. The same procedure was done for cracks of length equals to 40 mm and the results are presented on figure 45. As it was expected, as the size of the defects increased, the error on each simulation increased as well; for a relative distance of 47 mm (the critical distance for 20 mm cracks) the error this time is approximately 10%, the threshold of 3.5% is reached only until the third peaks (black peaks) to set the critical distance value in 94 mm for this given crack size.

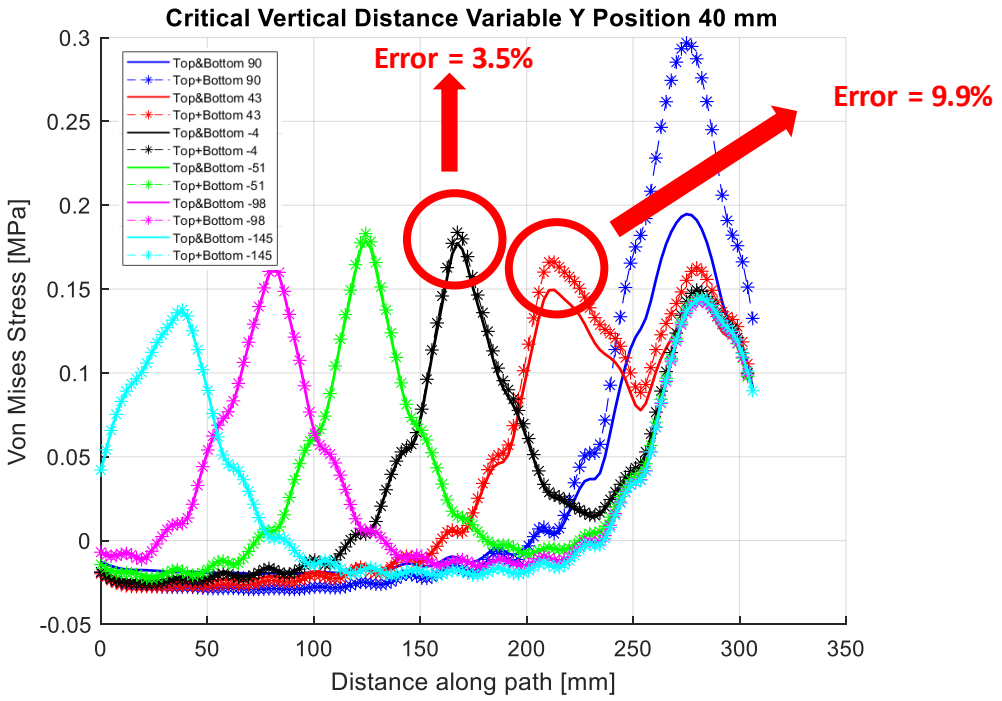

Figure 45. Von Mises stress vs distances for the critical distance test of 40 mm cracks.

The results of these tests reaffirm the influence of the crack size in the determination of the critical size. In order to include this value on the code a simple linear regression was done to relate all the possible crack sizes to a given critical distance value. To obtain more accurate results the same test was performed for 30 mm, 50mm and 80 mm where for the last crack size the critical distance was approximately the same value as the total length of the panel. Once the simulations were done a linear regression with a second order polynomial equation was fitted to the plot shown in figure 46, with the following expression:

$$y = 0.0102x^2 + 2.1959x - 3.2965$$

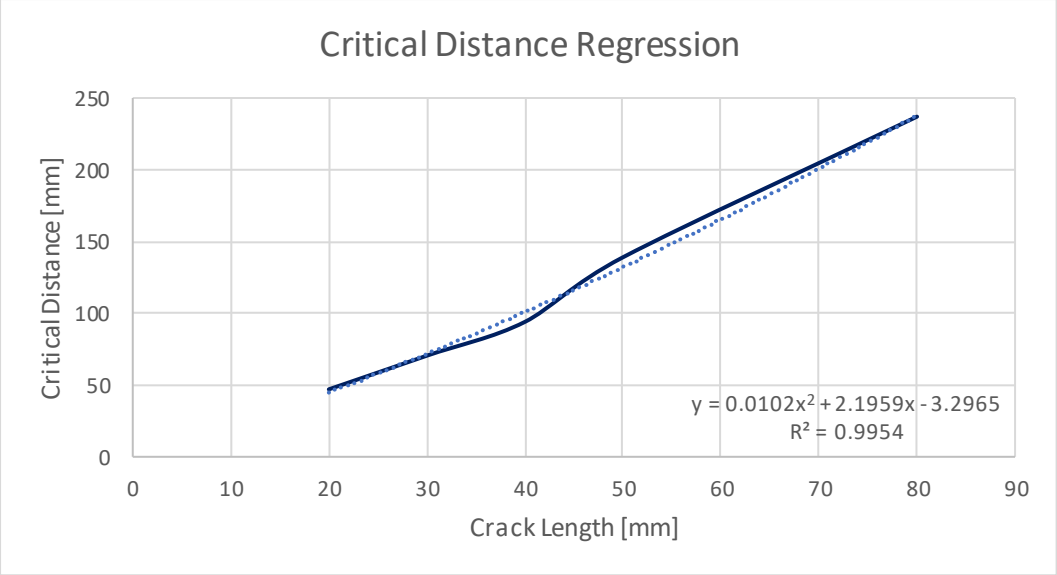

Figure 46. Critical distance vs Crack length.

For the integration of the regression on the code three assumptions were made:

1) The effect of the critical distance on the horizontal direction of the cracks in not considered, this is out of the scope of this thesis work and the interaction of the defects in this configuration should be further studied for more accurate results.
2) The critical distance will be considered in the vertical direction (Y) only if the horizontal direction (X) of the cracks coincides, this to complement what is mentioned in the previous point.
3) To minimize the complexity of the solution, for the case of different crack sizes (i.e., L1 > L2 or vice versa) the critical distance of the largest crack, which corresponds to the one with the largest value, will be considered. This will give a conservative number of simulations.

Example:

- For cracks with **ΔY lower than the critical distance and the same X position**. (the check mark means that the simulation is done)

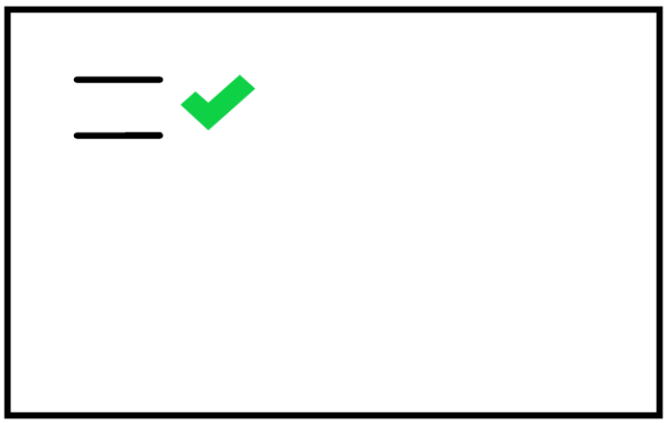

Figure 47. Simulation with relative distance lower than the critical distance for the same x position example.

- For cracks with **ΔY greater than the critical distance and the same X position**. (the x-mark means that the simulation is not done because it can be replicated by the linear superposition of two single cracks)

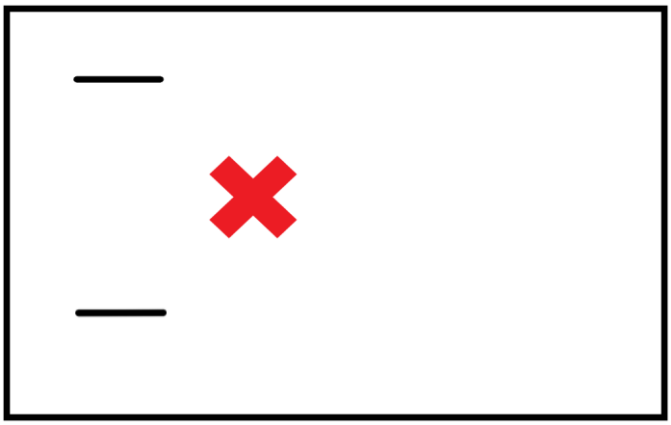

Figure 48. Simulation with relative distance higher than the critical distance for the same x position example.

- For cracks with **ΔY lower than the critical distance but different X position**. (the check mark means that the simulation is done, see assumption number 2)

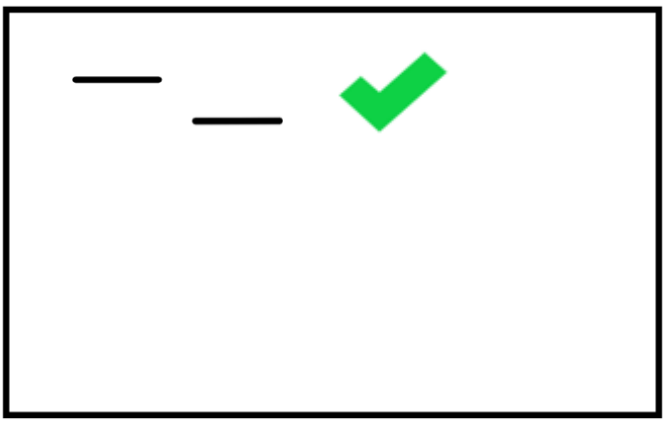

Figure 49. Simulation with relative distance lower than the critical distance for different x position example.

This concludes all the corrective solutions done to reduce the number of simulations and the computational time to build the database for two cracks on the panel, there are some other features (variable crack lengths symmetries along the panel and horizontal critical distance) that can be further exploited that will allow the reduction of even more iterations inside the loops, however they increase the complexity of the extension of the code.

Summarizing, the database is built by as set inputs that correspond to the desired steps the user in the horizontal direction (X), the vertical direction (Y) and the number of crack sizes to be simulated, followed by a set of nested for-loops, starting from the variation of the crack sizes, followed by the variation of the positions of the crack, that depending on the order of the for-loops will define which of the defects is the stationary and which one is the moving one. After determining the order of the loops, the regression that accounts for the critical distance is calculated for both iterations of the crack length in the given time, then, the conditional statements are integrated to the code. First, they define whether the defects are of the same size or not and which of them is larger, the following conditional determine if there is any overlap between the cracks and finally the last conditional evaluates if the vertical position between cracks is above or below the critical distance. At the end of the nested for-loops three matrices are filled: M_1, M_2 and M_3, that correspond for the for loops of ($L_1 = L_2$), ($L_1 > L_2$) and ($L_2 > L_1$) respectively, then all three matrices containing all the possible

positions are summed up into a master matrix called $\underline{M_{final}}$. To observe the difference between the number of simulations of the algorithm with the application of the solutions and the number of simulations of the algorithm without them a fourth matrix containing all the positions without any conditional statement is done, this matrix is called "complete matrix" or M_{com}. (For further details on the structure of the code, please refer to the appendices)

The following examples illustrate the difference between the M_{com} and the M_{final} for different input values.

Example 1)

- Number of steps in X = **6**.
- Number of steps in Y = **5**.
- Distance between Y steps = **64.25 mm**.
- Two crack lengths = **20 mm and 30 mm**.
- Critical distance for 20 mm (two cracks of 20 mm) = **44.7 mm**.
- Critical distance for 30 mm (two cracks of 30 mm) = **71.8 mm.**

1) Number of simulations of the complete matrix = **3480**
2) Number of simulations of the final matrix = **780**

The number of simulations was reduced drastically in this case because **the critical distance of 20 mm is lower than the step in the Y direction**, which means that most the simulations of the crack of 20 mm were bypassed by the code. Also, the number of simulations is low because the after the second step in Y (**128.5 mm** between cracks) the critical distance of both crack sizes is below the threshold, hence, those simulations are also suppressed.

Example 2)

- Number of steps in X = **6**.
- Number of steps in Y = **5**.
- Distance between Y steps = **64.25 mm**.
- Two crack lengths = **30 mm and 50 mm**.
- Critical distance for 30 mm (two cracks of 30 mm) = **71.8 mm**.
- Critical distance for 50 mm (two cracks of 50 mm) = **132 mm.**

1) Number of simulations of the complete matrix = **3480**
2) Number of simulations of the final matrix = **837**

For these initial conditions, the number of simulations increased because **the critical distance for a 50 mm crack (132 mm) is larger than the distance of two steps in the Y direction (128.5 mm)**, which means that the code will bypass the simulations when the cracks have at least three steps difference.

3.4. Transition to ABAQUS

M_{final} is created inserting all the required inputs to simulate in the panel in ABAQUS, all the transition from Matlab to ABAQUS done in the research of Gianmarco Verga was condensed in a single function, this function was called into the code introducing the elements of M_{final} as input values, then, the function was inserted in another for-loop with a total number of iterations equals to the number of rows of the final matrix. With the final version of the code a preliminary database was created as follows:

- Number of steps in X = **6**.
- Number of steps in Y = **7**.
- Initial length = **10 mm**.
- Final length = **50 mm**.

- Number of steps of L = **3**.

Note: Both cracks used the same input values in the creation of the preliminary database.

A total number of 5346 simulations were done and stored. The data gathered in these simulations must be extracted properly to be useful for the training of the ANN.

3.4.1 Simulations Post-processing

The data obtained in the simulations is crucial for training the algorithm, the extraction of the data must be as close as the real configuration of the panel where the data acquisition is done only on the points where the FBG sensors are located. Aiming to replicate this condition, a post-processing procedure of the simulations was developed as follows: first, the job to be processed was opened and the interest points on each stringer were identified according to the drawing of the real component depicted on figure 50. For each stringer, a path was drawn starting from the first sensor up to the last sensor on each stringer, to create the path the nodes where the sensors are located were defined. Knowing the positions of the sensors from the drawing it is possible to located them on the CAD panel, however, for some cases the position of the sensor did not correspond to a given node of the mesh of the FEM, so it was assumed that the sensor was located on the nearest node of the mesh, this assumption also simplifies de analysis of the data, the final paths can be observed in figure 51. After locating the 20 sensors from the four stringers the output strain E22 data was plotted and saved individually for each stringer. (See appendices)

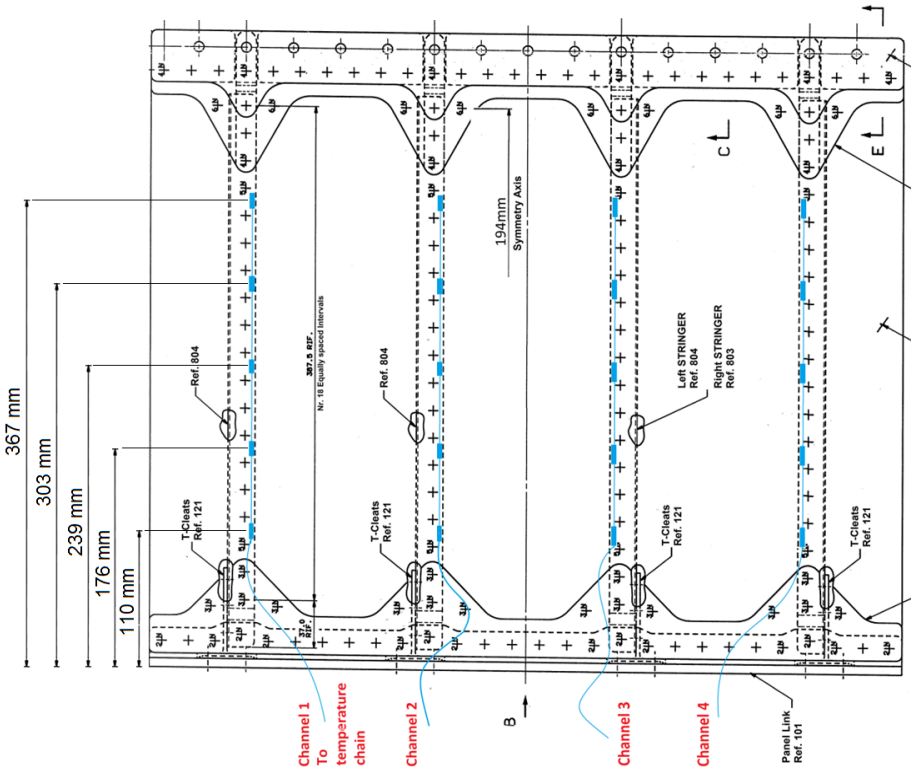

Figure 50. Real panel drawing. [21]

For the purpose of automatizing this post-processing procedure the ABAQUS script was taken from a final processed file and copied on Matlab where it was treated and inserted in a for-loop with a total number of iterations equals to the length of M_{final}, remember that for each simulation a job file is created containing the data of that respective set of positions, so for each simulation a job.odb file is opened by the post-processing code. Once the for-loop is done a set of folders is created, one per each job file, each folder contains four files where the strain data of the paths of the stringers is stored (one file per stringer). Afterwards, the data is extracted from the files and the strain values that correspond to the sensors are organized on a matrix in a way that the rows of the matrix match with the number of jobs and the columns match with the number of sensors. For the preliminary database, the strain matrix obtained after the post-processing of the data has a dimension of 5346 X 20, this arrangement of the data is going to be discussed on the following chapter of this document.

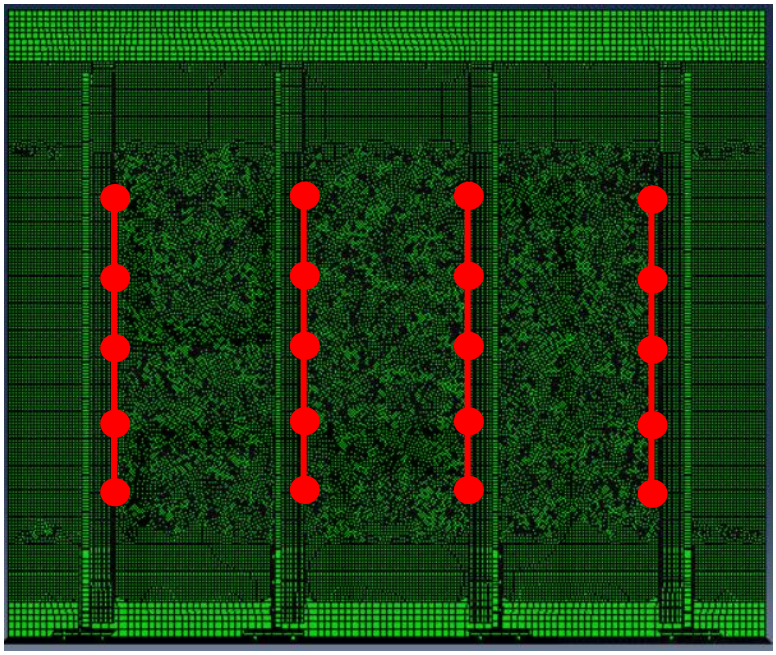

Figure 51. Paths and sensor locations chosen on the FE model.

CHAPTER 4: TRAINING OF THE NEURAL NETWORK

4.1. Training Stage

As the last part of this thesis, an artificial neural network was trained with data obtained from the simulations of the database to fulfill a basic task, this was done to test if the information recollected on the database is useful for future application of this approach in SHM diagnosis and prognosis. For this case, the task for which the ANN was trained was to determine in which bay of the skin of the panel the defects were located. As it was mentioned on the previous chapter, the panel consist of three sub-skin bays between four stringers which the cracks can appear according to the iterations of the nested for-loops, the objective of the ANN is to identify if the panel is damaged and in case it is damaged to recognize in which portion of the skin the cracks are. For this, a set of classes / labels were defined that correspond to each of the possible cases that may occur during the presence of the defects. They were determined as follows:

Classes / Labels:

0) Healthy / undamaged panel
1) Both defects on the first bay of the skin (left skin)
2) Both defects on the second bay of the skin (central skin)
3) Both defects on the third bay of the skin (right skin)
4) One crack on the first bay and the second crack on the second bay of the skin
5) One crack on the first bay and the second crack on the third bay of the skin
6) One crack on the second bay and the second crack on the third bay of the skin

For example:

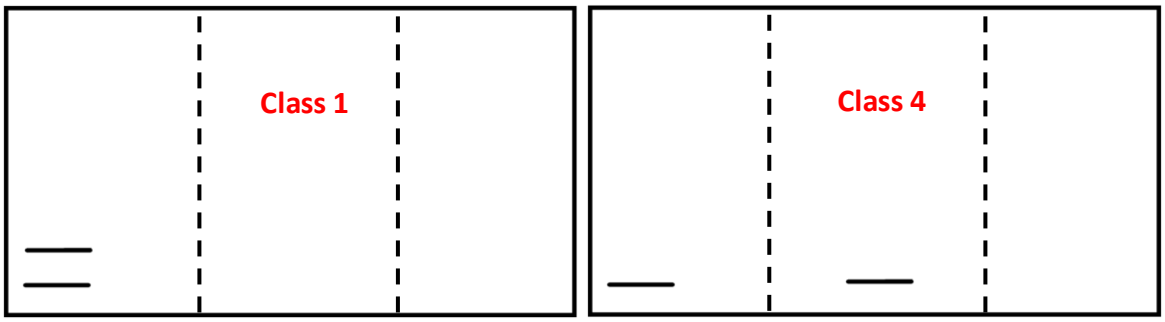

Figure 52. Classes / labels examples.

To achieve this, a classic feed-forward multi-layer perceptron neural network is used which is trained by the set of data gathered form the database, as explained on section 3.4, the final matrix obtained at the end of the post-processing algorithm contains all the strain values that are related to the corresponding position of the defects, it is arranged in a way that the values inserted on the matrix correspond to the strain in the location of the sensors. Additional to this matrix, a healthy condition must be integrated for the algorithm to have a reference value of the panel without defects and understand if any anomaly has been introduced to the system. To include the healthy samples, a single simulation of a healthy panel was performed, then the post-processing procedure was done normally as for the rest of the damaged simulations. Once the strains of the 20 sensors were extracted a noisy signal was added to the data to reproduce a total of 900 samples, a number significant enough to properly train the

ANN. The noisy signal introduced was a set of random values with a gaussian distribution multiplied by the standard distribution of the sensor signal (2 μ strain). The following equation shows the addition of the noise to the undamaged strain data which runs in a for-loop cycle of 900 iterations to generate the healthy samples.

$$\text{undamaged\_matrix(i)} = \text{ud\_sensors} + \text{noise}_{\text{gauss}}(\text{i}) \cdot \sigma_{\text{sensor}}$$

Where the undamaged_matrix(i) stands for the i_{th} sample of a healthy panel, ud_sensors indicate the data obtained from the single undamaged simulation and σ_{sensor} accounts for the standard deviation of the sensor signal as mentioned on the previous paragraph.

To train the ANN a series of inputs and their corresponding outputs must be introduced to the algorithm first, so that once is trained it can deliver a correct output to any new input inserted to the system. Having the strain data of the healthy panels, the strain data of the damaged panels and their respective positions, it is possible to generate an output array that will teach the algorithm which is the desired outcome for those given inputs. First the final input matrix is organized by concatenating the healthy matrix to the damaged strain matrix taken from the database, the following step is to assign an output to each row of the final input matrix, the value designated for each row must match the corresponding class according to the positions of the cracks.

$$[\text{input\_matrix\_ANN}] = \begin{bmatrix} \begin{bmatrix} \text{undamaged\_matrix} \\ 900 \times 20 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \text{damaged\_matrix} \\ 5346 \times 20 \end{bmatrix} \\ \vdots \\ \vdots \end{bmatrix}$$

To build the output vector a simple code was developed where according to the horizontal positions (X) of the cracks taken from M_{final} a value between 1 – 3 is assigned, this number represents the bay of the skin where that specific crack is

located. A for-loop is created and inside it three conditions to identify where the crack is located, for instance, if the crack has a horizontal position X between the first and the second stringer two, the result is number 1, if the crack is between the second and the third stringer, the outcome is number 2 and if the crack is between the third and the fourth stringer, the output is number 3. When the numbers are assigned to both cracks in all the simulations, a second cycle is created having the conditions stated to generate the classes for the output vector, for example, if for simulation (i) crack one and crack two have as designated numbers 1 and 1 respectively, the i_{th} element of the output vector would be 1 as well, if the simulation (i+1) has no defects, the element $i_{th}+1$ of the output vector would be 0 and so on. For the generated database, the dimension of the output vector was 6246 X 1.

The structure of the matrices of what is called the train dataset is presented in figure 53, the dimension of the matrices also gives an insight of how the architecture of the NN should be, for this application it is correct to assume that the input values correspond to every measure taken from the sensors and considering that there are 20 of them, the input layer of the NN consist of 20 nodes. The same analysis can be done for the output layer, taking into account that there are seven classes that can categorize a simulation one can state that the number of output neurons is 7. For the hidden layer, the choice of the number of neurons is based in experience with previous analysis with neural networks, testing with trial and error, making sensibility analysis to find a number of hidden neurons that yields to better results. The final hidden layer had a total number of 15 neurons, this one reached the better outcome for the inputs given. Finally, to obtain more accurate results a dropout layer of 5% obtained empirically to avoid overfitting, the results obtained are presented on the following section of this chapter.

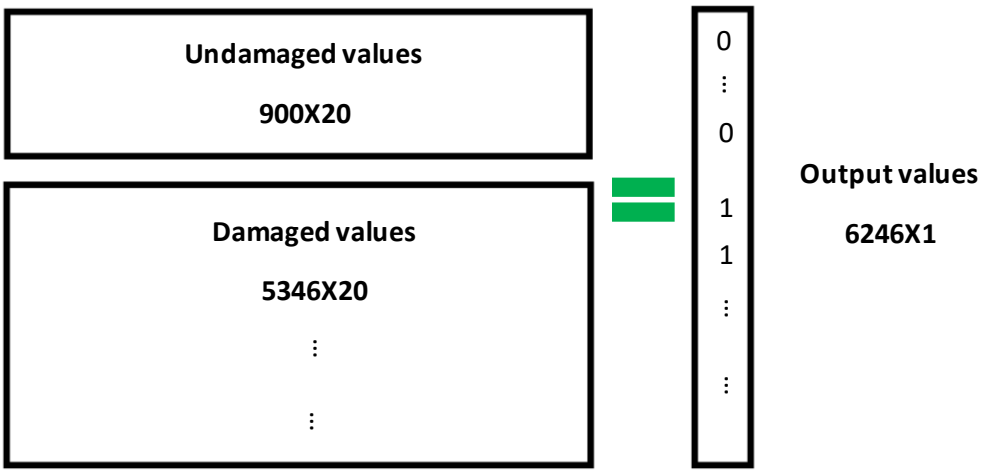

Figure 53. Matrix arrangement for the training dataset.

4.2. Neural Network Testing

The training and validation procedures were done in Python and the results are shown in figure 54 and figure 55. After the validation of the model an accuracy of 97% was obtained, also focusing on the loss function plotted figure 55 one can observe that the trend is always decreasing which represents a positive behavior of the neural network due to the lack of overfitting. Overfitting may occur when the complexity of the algorithm is high and has an outstanding performance on the learning procedure, but once an input value that do not belong to the training dataset is introduced to the system the output achieved by the NN is not accurate. Considering that in the real conditions the cracks may nucleate in random positions along the skin (not necessarily on the positions where the simulations were made) the algorithm must be able to precisely recognize their location for further diagnosis of the damage in the component. This training procedure was done successfully and shows the usefulness of the database created by means of the FEM.

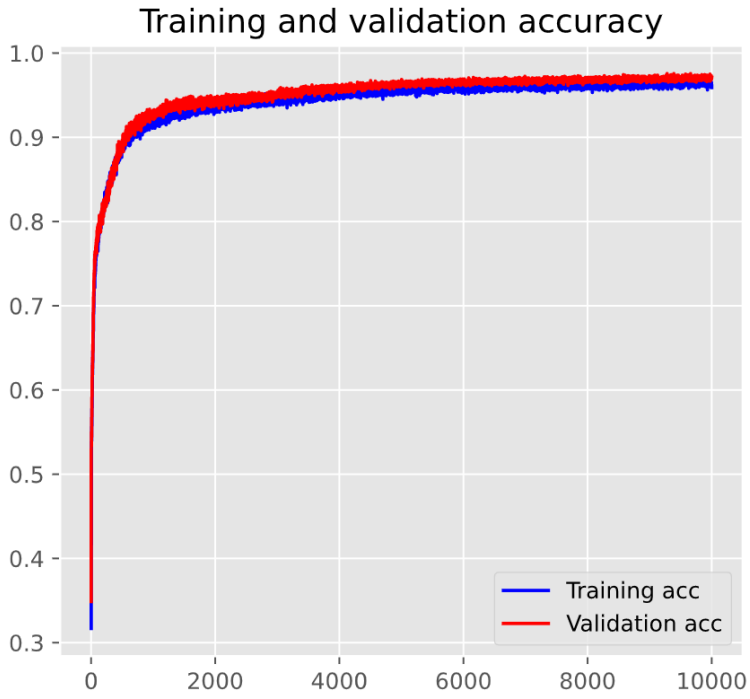

Figure 54. Training and validation accuracy.

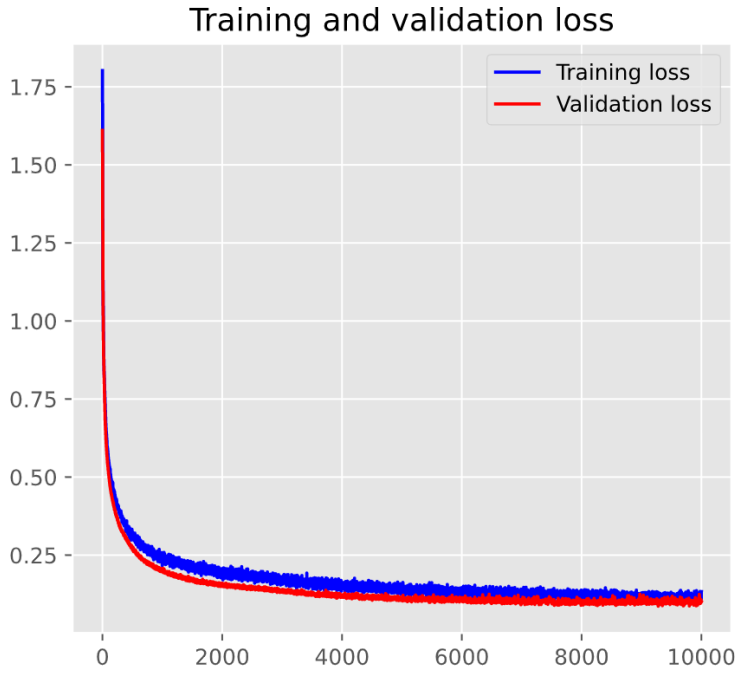

Figure 55. Training and validation loss.

Testing the neural network with the training dataset gave a good classification performance, as mentioned above, around a 97% of accuracy on categorizing the simulations on the respective class was achieved. Plotting the actual label versus the predicted label indicated by algorithm allows the user to understand if the approximations done by the NN are accurate or not and proceed with corrective actions to improve the functioning of the algorithm. To have a clear idea of the recognition capacity of the algorithm two tests were done using the training dataset, figure 56 shows the first experiment where 100 random samples were introduced to ANN to observe the prediction performance. The black x-marks represent the actual label that correspond to the introduced simulations while the red circles are the predictions done by the algorithm after the training and validation process. Notice how, for all the introduced simulations, only three of the x-marks do not match with its corresponding red circle, this means that the algorithm was not able to classify the simulation correctly, moreover, certainly the 97% of accuracy of the NN can be clearly observed in this test. Also, looking to the graph, class number 3 always remains empty for all the possible positions, this is due to the symmetries applied on the creation of the database algorithm, there is no simulations where both of cracks are present on the third bay of the skin of the panel because all this case can be replicated by using the information of the simulations done on the left-hand side of the panel where the first bay of the panel is located.

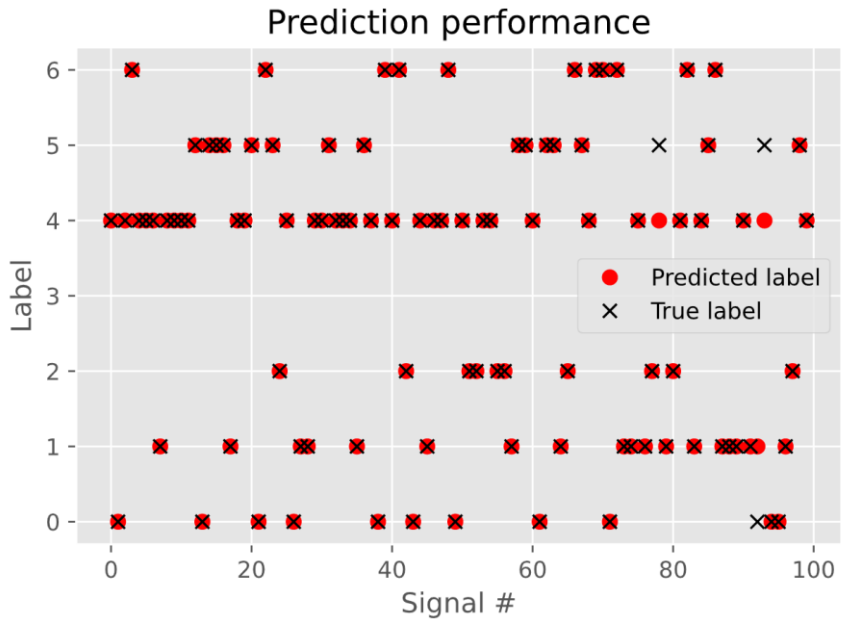

Figure 56. Label vs signal number prediction performance graph.

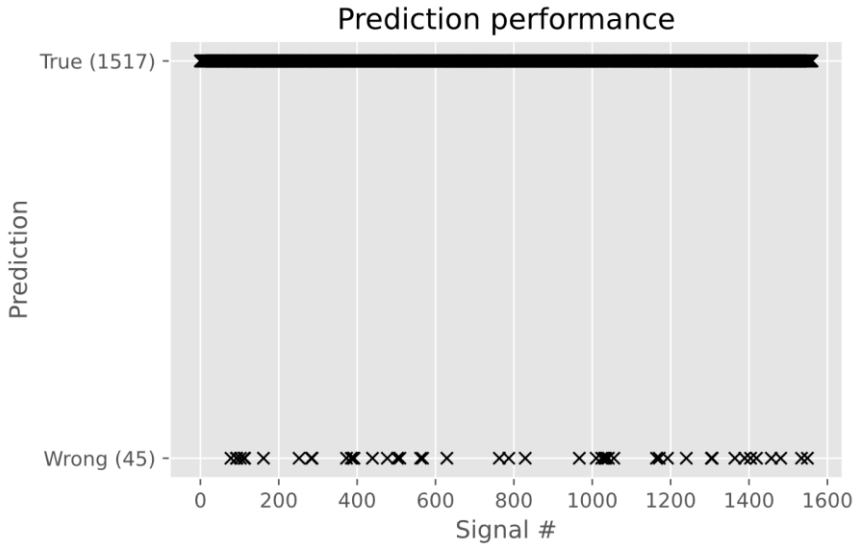

Figure 57. Prediciton vs signal number prediction performance graph.

The second prediction performance indicator was done by simulating a larger amount of data just to observe the consistency of the algorithm in the classification of the simulations. In this test the only criteria evaluated was whether the prediction was correct or incorrect for a total number of 1562 random signals taken from the training dataset. Figure 57 shows the plot that contains the output of the algorithm, in this test, the x-mark represent one simulation that is placed above the graph if the prediction

indicated by the NN is correct, on the contrary, it is place below is the prediction is wrong. For the 1562 simulations tested, only 45 of them were incorrect predictions accounting for a 2.88% of error which is coherent according to the validation plot.

Finally, as a last verification of the neural network six random signals taken from the input matrix of the training dataset were introduced to the algorithm, in this test the classes of the six simulations chosen were known. The idea was to test a single case for each of the possible labels that the NN can choose to classify a simulation, also, it allows the user to understand the outcome of each specific output neuron during the process of classification. The numbers of the samples taken from the input matrix are the following:

- 525 = Class 0
- 905 = Class 1
- 1686 = Class 2
- 1025 = Class 4
- 1928 = Class 5
- 5849 = Class 6

		LABELS						
		0	**1**	**2**	**3**	**4**	**5**	**6**
SIGNAL	**525**	**1**	0	0	1.13E-22	1.60E-15	2.19E-08	0
	905	6.90E-03	**0.91**	0	1.22E-15	0.09	7.42E-07	0
	1686	2.31E-25	0	**0.92**	4.38E-15	0.07	1.59E-05	0.02
	1025	0.04	1.04E-04	6.80E-12	2.98E-10	**0.96**	2.40E-04	0
	1928	0	0	1.84E-15	3.53E-21	6.77E-11	**1**	6.73E-16
	5849	0	0	1.70E-06	4.33E-22	5.73E-11	8.86E-03	**0.99**

The table above shows the results for the samples of the input matrix, as the data was taken from the training dataset it was expected to have a 100% accuracy on the prediction of the classes for this selected data. Looking at the cells, one can notice that the due to the sigmoid function inside the neural network all the output neurons present a number between 0 and 1, the neuron with a number closer to 1 defines the

final prediction of the algorithm, the green numbers correspond to the prediction of the class for the given signal. The green numbers on the table represent the class which the algorithm chose for each simulation, as expected they correspond to the actual classes for each case. The neural network is successfully trained and functional.

CHAPTER 5: CONCLUSIONS

The methodology presented in this thesis work allows implementing an efficient procedure to create a strain field database of a structure of interest by means of an algorithm which automatizes FE simulations on ABAQUS. The algorithm is able to significantly reduce the computational time for the construction of the database by effectively eliminating most of the undesired crack positions, exploiting symmetries and other geometric features, overcoming limitations of the FEM and avoiding inconsistencies with respect to the real setup of the helicopter. The ABAQUS simulations ran smoothly, post-processing and data acquisition procedures are done automatically as well through a script developed to this purpose, letting the user to easily elaborate a dataset from the strain data gathered from the simulations. A database consisting of 5346 strain samples was obtained with this procedure. The data coming from the created database was used to successfully train a feed-forward multi-layer perceptron neural network for classification to perform tasks as damage detection and basic crack localization. The algorithm worked with a 97% of accuracy on the prediction of the location of the defects with no overfitting issues. As presesnted on this document, the neural network allows the localization of both cracks depending on the bay where they appear on the skin of the panel and classifies the output according to a label number that corresponds to the position of the defects.

Furthermore, considering the structure of the database, as a future work a new training dataset could be developed by including the missing symmetric positions along the panel that can be obtained by mirroring the existing data on the present database, but this procedure may be automatized as well coding a new algorithm to

obtain the mirrored data efficiently. Moreover, the code that governs the construction of the database can be further improved by removing the few remaining symmetric cases, but it requires further testing and analysis of all the possible unwanted scenarios. Lastly, it would be useful to develop databases applying the methodology presented in this document to analyze the effect of multi-site damage on the structural integrity of aeronautical panels by introducing more cracks into the structure and studying the degradation of the component affected by actions of the damages simultaiously.

Finally, the possibility of using FE numerical simulations to train machine learning algorithms allows replacing expensive experimental data acquisition and reducing the time required for gathering the amount of information needed to tune this type of systems. This thesis work shows that this is a feasible approach that can be exploited in several fields, such as structural health monitoring of aerospace structures, which has been discussed in this document.

CHAPTER 6: BIBLIOGRAPHY

[1] G. Lu and Y. J. Yang, "Structural health monitoring," *Internet Things Data Anal. Handb.*, pp. 665–674, 2017, doi: 10.1002/9781119173601.ch40.

[2] P. Seventekidis, D. Giagopoulos, A. Arailopoulos, and O. Markogiannaki, "Structural Health Monitoring using deep learning with optimal finite element model generated data," *Mech. Syst. Signal Process.*, vol. 145, p. 106972, 2020, doi: 10.1016/j.ymssp.2020.106972.

[3] S. Chandrasekaran, *Lecture 1 - Part 1: Introduction to SHM*. India: NPTEL, 2019.

[4] S. Chandrasekaran, *Lecture 1 - Part 2: Introduction to SHM*. India: NPTEL, 2019.

[5] F.-G. Yuan, S. A. Zargar, Q. Chen, and S. Wang, "Machine learning for structural health monitoring: challenges and opportunities," no. November, p. 2, 2020, doi: 10.1117/12.2561610.

[6] S. Chandrasekaran, *Lecture 3 - Part 2: Components of SHM*. India: NPTEL, 2019.

[7] S. Chandrasekaran, *Lecture 4 - Part 2: Challenges in SHM*. India: NPTEL, 2019.

[8] Wikipedia, "Artificial neural network," 2021. https://en.wikipedia.org/wiki/Artificial_neural_network (accessed Apr. 12, 2021).

[9] Z. Meng, Y. Hu, and C. Ancey, "Using a data driven approach to predict waves generated by gravity driven mass flows," *Water (Switzerland)*, vol. 12, no. 2, 2020, doi: 10.3390/w12020600.

[10] K. Smarsly, K. Dragos, and J. Wiggenbrock, "Machine Learning Techniques for Structural Health Monitoring," *Lect. Notes Mech. Eng.*, no. July, pp. 3–24, 2020, doi: 10.1007/978-981-13-8331-1_1.

[11] S. Chandrasekaran, *Artificial Neural Network (ANN) in the SHM process.* India: NPTEL, 2019.

[12] M. Giglio, A. Manes, U. Mariani, R. Molinaro, and W. Matta, "Helicopter fuselage crack monitoring and prognosis through on-board sensor network," *6th Int. Conf. Cond. Monit. Mach. Fail. Prev. Technol. 2009*, vol. 2, pp. 1308–1321, 2009.

[13] C. Sbarufatti, A. Manes, M. Giglio, U. Mariani, and ..., "Application of Structural Health Monitoring over a Critical Helicopter Fuselage Component," *Proc. 2nd Int. Conf …*, 2010, [Online]. Available: http://www.vitrociset.it/images/files/Application of Structural Health Monitoring.pdf.

[14] T. Dong and N. H. Kim, "Cost-effectiveness of structural health monitoring in fuselage maintenance of the civil aviation industry," *Aerospace*, vol. 5, no. 3, 2018, doi: 10.3390/aerospace5030087.

[15] C. Sbarufatti, "Optimization of an artificial neural network for fatigue damage identification using analysis of variance," *Struct. Control Heal. Monit.*, vol. 24, no. 9, pp. 1–17, 2017, doi: 10.1002/stc.1964.

[16] C. Sbarufatti and M. Giglio, "Performance qualification of an on-board model-based diagnostic system for fatigue crack monitoring," *J. Am. Helicopter Soc.*, vol. 62, no. 4, pp. 1–10, 2017, doi: 10.4050/JAHS.62.042008.

[17] D. Cristiani, C. Sbarufatti, F. Cadini, and M. Giglio, "Fatigue damage diagnosis and prognosis of an aeronautical structure based on surrogate modelling and particle filter," *Struct. Heal. Monit.*, 2020, doi: 10.1177/1475921720971551.

[18] A. Hijazi, S. Al-Dahidi, and S. Altarazi, "Residual strength prediction of aluminum panels with multiple site damage using artificial neural networks," *Materials (Basel).*, vol. 13, no. 22, pp. 1–25, 2020, doi: 10.3390/ma13225216.

[19] A. Alaimo, A. Barracco, A. Milazzo, and C. Orlando, "Artificial Neural Networks Comparison for a SHM Procedure Applied to Composite Structure," no. September, pp. 9–12, 2013, doi: 10.13140/2.1.1538.4006.

[20] G. Verga, "Gianmarco VERGA 867603," vol. 1, p. 41, 2019.

[21] C. Sbarufatti and F. Cadini, "Practice PANEL - Part 1," Milan.

CHAPTER 7: Appendices

7.1. Appendix A: Code for construction of the database.

```
clear all
close all
clc

%% Input Data
x_piano1 = (1.0);          %coordinate del piano su cui disegnare
y_piano1 = (1.0);
z_piano1 = (0.0);

% x_origine1               %coordinate origine sketch prima cricca
% y_origine1
z_origine1 = (0.0);

L1_x1 = (0.0);             %punti di inizio e fine della prima cricca
L1_y1 = (0.0);
% L1_x2                    %(IL SISTEMA DI RIFERIMENTO è LO SKETCH!)
L1_y2 = (0.0);

% C1_                       %punti di centro e raggio del cerchio della prima cricca
C1_y1 = (0.0);
% C1_x2                     %(IL SISTEMA DI RIFERIMENTO è LO SKETCH!)
C1_y2 = (0.0);

x_piano2 = (1.0);          %coordinate del piano su cui disegnare
y_piano2 = (1.0);
z_piano2 = (0.0);


% x_origine2                %coordinate sketch seconda cricca
% y_origine2
z_origine2 = (0.0);

L2_x1 = (0.0);              %punti di inizio e fine seconda cricca
L2_y1 = (0.0);
% L2_x2                     %(IL SISTEMA DI RIFERIMENTO è LO SKETCH!)
L2_y2 = (0.0);

% C2_x1                     %centro e raggio del cerchio della seconda cricca
C2_y1 = (0.0);
% C2_x2                     %(IL SISTEMA DI RIFERIMENTO è LO SKETCH!)
C2_y2 = (0.0);

Li = 10;                    %Initial crack length [mm]
Lf = 50;                    %Final crack length [mm]

nstep_ox1 = 6;              %Number of steps in direction x for the first crack %Must be multiple of 3
nstep_oy1 = 7;              %Number of steps in direction y for the first crack
nstep_L1 = 3;               %Number of values for the length for the first crack

nstep_ox2 = 6;              %Number of steps in direction x for the second crack %Must be multiple of 3
nstep_oy2 = 7;              %Number of steps in direction y for the second crack
nstep_L2 = 3;               %Number of values for the length for the second crack
```

```
pos_y1 = linspace(-145.5,111.5,nstep_oy1);
pos_y2 = linspace(-145.5,111.5,nstep_oy2);
v_L1 = linspace(Li,Lf,nstep_L1);
v_L2 = linspace(Li,Lf,nstep_L2);


if  rem(nstep_ox1/3,1)==0
else
     disp('The step in the x direction must be a multiple of 3 for crack 1');
     return
end

if rem(nstep_ox2/3,1)==0
else
     disp('The step in the x direction must be a multiple of 3 for crack 2');
     return
end


dev_factor = 0.1;         %caratteristiche della mesh
min_size = 0.1;
max_size = 2.5;

k = 0;
M1 = zeros(1);
M2 = zeros(1);
M3 = zeros(1);
tStart = cputime;
t = zeros(1);
```

```
 %% Cracks position and size nested for-loops

for r = 1:1:length(v_L2)
     L2_x2 = v_L2(r);
     C2_x1 = v_L2(r);
     C2_x2 = v_L2(r) + 2.5;
     pos_x2 = linspace(-210,210-(v_L2(r)),nstep_ox2);
     reg_2 = 0.0102*v_L2(r)^2+2.1959*v_L2(r)-3.2965;

     for q = 1:1:length(v_L1)
             L1_x2 = v_L1(q);
             C1_x1 = v_L1(q);
             C1_x2 = v_L1(q) + 2.5;
             pos_x1 = linspace(-210,210-(v_L1(q)),nstep_ox1);
             reg_1 = 0.0102*v_L1(q)^2+2.1959*v_L1(q)-3.2965;

         if L1_x2 == L2_x2

              for n = 1:1:round(length(pos_x2)/2)...


         elseif L1_x2 > L2_x2

                 for n = 1:1:round(length(pos_x2)/2)...

         elseif L2_x2 > L1_x2

                 for n = 1:1:round(length(pos_x2)/2)...
         end

     end
end
```

```
%% Creation of the final matrix
M2f = zeros(size(M1));
M2f(1:size(M2,1),1:size(M2,2)) = M2;

M3f = zeros(size(M1));
M3f(1:size(M3,1),1:size(M3,2)) = M3;

M_final = M1+M2f+M3f;              %Final matrix with all the feasible positions

M_com = complete_matrix(nstep_ox1,nstep_oy1,nstep_L1,nstep_ox2,nstep_oy2,nstep_L2,Li,Lf); %considering all positions
```

```
%% Automatic Abaqus Simulations
tEnd = cputime - tStart;

xlswrite('M_final.xlsx',M_final)

TM = zeros(1);

wb = waitbar(0,'0%','Name','Simulations Completed',...
    'CreateCancelBtn','setappdata(gcbf,''canceling'',1)');

setappdata(wb,'canceling',0);
```

```
for i = 1:1:length(M_final(:,1))
tic;
A = abaqus_simulation(x_piano1, y_piano1, z_piano1, M_final(i,3), M_final(i,4), z_origine1, L1_x1, L1_y1, M_final(i,7), L1_
t(i) = toc/60;
t_m = mean(t);
t_simulation = length(M_final(:,1))*t_m;
t_rem = (length(M_final(:,1))-i)*t_m;
sim_com = (i/length(M_final(:,1)))*100; %[%]

TM(i,1) = t_rem;
disp('Time Remaining [min]')
disp(TM)

    if getappdata(wb,'canceling')
        break
    end

waitbar(i/length(M_final(:,1)*100),wb,sprintf('%0.2g%%',sim_com));

end

delete(wb);
```

7.2. Appendix B: Code for ABAQUS automatic post-processing.

```
%% Abaqus post-processing

clear all
close all
clc

[position_data] = xlsread('M_final.xlsx');
path_name = 'C:\Users\User\Desktop\Postprocessing\'; % Put your path
path_output = 'C:\Users\User\Desktop\Postprocessing\Output\'; % Put your path

for k=1:length(position_data)
    % Setup file name
    file_name = ['job-00' num2str(k) '.odb'];
    % Setup output name
    strain1_data = ['StrainST1-00'  num2str(k) '.rpt'];
    strain2_data = ['StrainST2-00'  num2str(k) '.rpt'];
    strain3_data = ['StrainST3-00'  num2str(k) '.rpt'];
    strain4_data = ['StrainST4-00'  num2str(k) '.rpt'];
    mkdir(['\Output' 'job-00' num2str(k)])
    % Abaqus function (**TO CODE**)
    abaqus_write(path_name,path_output,file_name,strain1_data,strain2_data,strain3_data,strain4_data);
end
```