POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Master's Degree Thesis

Multifidelity modelling of damaged aerospace structures: A transfer learning based approach

Supervisors

Dott. Ing. Laura MAININI

Prof. Marco GHERLONE

Candidate

Mirko ERMACORA

YEAR 2021/2022

"Science can amuse and fascinate us all,
but it is engineering that changes the world."
Isacc Asimov

Abstract

A critical part of structural health monitoring is accurate detection of damages in the structure. Simulation based optimization for damage detection and identification requires numerous iterations with expensive simulated models which is impractical for real-time assessment.

This thesis proposes a multi-fidelity Reduced Order Modeling (ROM) method based on transfer learning, to develop an emulator for fast online data generation. The online data are obtained using a machine learning algorithm trained with an offline database which is the result of a transfer learning method. In this method, finite element simulations with different fidelities are combined using Reduced Order models and manifold alignment to determine a common space where the accuracy of high fidelity simulations is fused with the spatial resolution of the low fidelity simulations.

Then, it is proposed a comparison between the machine learning algorithms of gaussian regression, cokriging, regression trees and Self Organizing Maps (SOMs) to determine the most suited for cut damage detection in composite plates for aerospace application.

Acknowledgements

Desidero ringraziare innanzitutto, la Dottoressa Laura Mainini e il Professore Marco Gherlone per la disponibilità, l'attenzione e la gentilezza dimostrate durante la stesura di questo lavoro, oltre che per il sostegno dato nei momenti di sconsolatezza in cui la tesi sembrava essere arrivata a un punto morto.
Inoltre, i miei più preziosi ringraziamenti vanno alla mia famiglia che, anche a distanza, mi ha sempre supportato e ha contribuito al raggiungimento di questo importante traguardo.
Un ringraziamento speciale va ai miei amici, che in un modo o nell'altro, hanno condiviso tutti questi anni di gioie e sacrifici.

Table of Contents

List of Tables

List of Figures

Acronyms

BMU Best Matching Unit

DOE Design Of Experiment

DOF Degrees Of Freedom

FEM Finite Element Method

GP Gaussian Process

HF High Fidelity

LF Low Fidelity

LHS Latin Hypercube Sampling

MF Medium Fidelity

ML Machine Learning

MOR Model Order Reduction

NRMSE Normalized Root Mean Squared Error

PCL Patran Command Language

POD Proper Orthogonal Decomposition

POM Proper Orthogonal Mode

POV Proper Orthogonal Value

ROM Reduced Order Model

SOM Self Organizing Map

SVD Singular Value Decomposition

Chapter 1

Introduction

Structure's health monitoring and damage detection has always been a fundamental topic in aerospace design. Simulation based optimization for damage detection and identification requires numerous iterations with expensive simulated models which is impractical for real-time assessment.

At the moment, there are several studies which try to address this problem using a vibration based approach [1] [2] or using a static response approach [3] [4], which has become more interesting due to the introduction of real time structural monitoring. This thesis proposes an original multi-fidelity Reduced Order Model (ROM) [5] method based on transfer learning, to develop an emulator, or surrogate model, for fast online data generation.

A surrogate model [6] can be thought as a black box function, tuned with the information from the data considered the true response of the system to predict the behavior of the simulation model as closely as possible while being computationally cheaper to evaluate [7]. The data obtained from the surrogate models are approximate and intended to do first estimates of structure's health.

To train the surrogate model is necessary to create a database of high fidelity data which explores the full field of study of the phenomena and with an accuracy similar to the real data. This high fidelity data usually are very expensive to produce, in either time (computer simulation) or resources (experimental data), so it is introduced a multi-fidelity approach. Multi-fidelity algorithms combine data obtained from different sources: high fidelity data, which have high accuracy but

are extremely expensive to produce; and low fidelity data, which have low accuracy but are cheap to produce and have better spatial resolution.
The case study is a small section of a representative composite wing which is modelled as a composite plate made of four layers with a fiber-cut damage perpendicular to the loading direction. Each specimen has a single cut localized in the third layer.
The first phase is based on creation of the database with three different levels of fidelity: high, medium and low. The models were created using the software MSC Patran and analysed with the software MSC Nastran. The high-fidelity models were used as reference data because, due to Covid-19 epidemic, it was impossible to access to the laboratory for the experimental data.
In the second/third phase, we initially tried to crate a surrogate model of the original strain field using CoKriging [8]. This was very time-demanding due to the high dimensionality of the data, so we determined a Reduced Order Model (ROM) using a Proper Orthogonal Decomposition (POD) [9] and we determined a lower dimensionality basis for each of three levels of fidelity.
In the fourth phase, to combine the information from the different levels of fidelity, we projected the high and medium fidelity basis to the lower fidelity base (which has the greater number of models and so, it has the best spatial exploration of the model) with a manifold alignment algorithm [10] called Procrustes analysis [11] and we determined the transformation matrices. This matrices are then used, in the fifth phase, to obtain the unified basis coefficients of each fidelity model for the low fidelity basis.
In the sixth phase, we tested the ability of different algorithms such as Gaussian Process regression [12], CoKriging, Regression Trees [13] and Self Organizing Maps (SOMs) [14], to estimate the damage position and its strain values.

Chapter 2

Physics's Based Model

In this chapter is described the development path of the model design. The case of study is based on real specimens designed to simulate a damage, with fiber cut, in a composite plate constituting the skin of a wing's structure. The specimens are modelled as composite plates with dimensions of $102 \times 458\ mm$, shown in figure 2.1. They are constituted of four layers of plain weave fabric of carbon prepreg (IM7/8552 AS4) laminated with the order [45°/0°/0°/45°].
The material properties are obtained from the data sheet published from NCAMP (National Center for Advanced Material) [15] and they are summarized in table 2.1.
The specimen were designed to be tested using a uni-axial tensile load applied

E_1	64121 MPa
E_2	64121 MPa
G_{12}	5722 MPa
ν_{12}	0.031
ρ	1570 kg/m^3

Table 2.1: Hexcel IM7/8552 AS4 properties

along the major dimension which represents a simplification of wing panel's load conditions during flight. In this case, the majority of the load is supported by the 0° layers, therefore layer 2 and 3 are the most critical, so the damage was inserted in one of these layers, specifically the layer 3. The damage consist in a cut of the fibers orthogonal to the major axis.

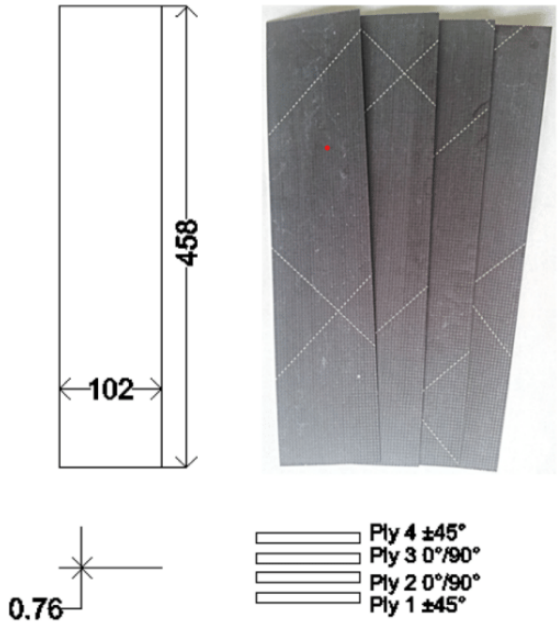

Figure 2.1: Real specimen dimensions and sequence of lamination[1]

The experimental part of the thesis did not happen for Covid restrictions, so the data from the real specimens were not available. Therefore, using the software *MSC Patran*, were modelled three different FEM models and we will refer to them as:

- **high fidelity model**
- **medium fidelity model**
- **low fidelity model**

These models are the source of information that will be combined to produce a surrogate model using a multi-fidelity approach.

[1]the specimen illustrated in the figure were produced under the visiting professor project "Multisource framework to support real-time structural assessment and autonomous decision making" at Politecnico di Torino

2.1 High Fidelity Model

The High fidelity model is used as reference model substituting the experimental data, hence it has to be as closer to reality as possible. In order to create this model are used 3D elements in particular HEXA [16] [17] elements (figure 2.5). There are two common types of thees elements: HEXA8 and HEXA20.

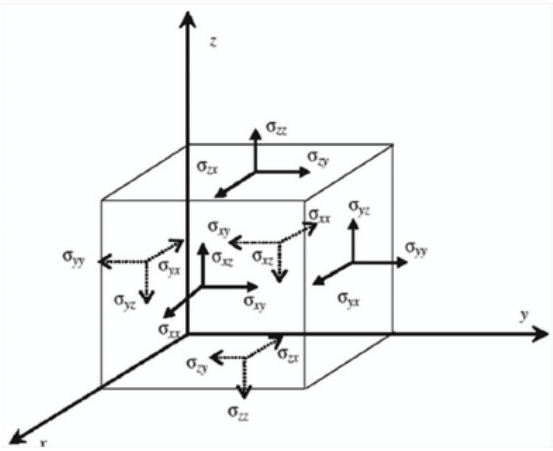

Figure 2.2: Stress components on HEXA8 elements [18]

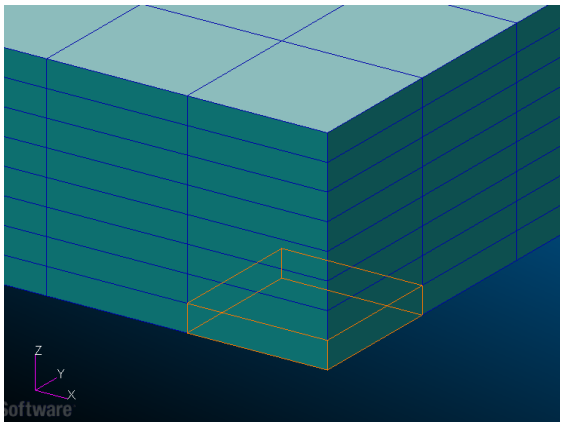

Figure 2.3: High fidelity model elements

HEXA8 elements are characterized by a linear shape function, hence the displacements of the mesh region between the nodes vary linearly with the distance between the nodes and they don't capture bending. HEXA20 elements are characterized by

a quadratic shape function and have more degrees of freedom (DOF) compared to the previous which means that they are more expensive to calculate. In this thesis the specimen is not subjected to bending loads so the we decided to use HEXA8 elements.

The dimensions of each element are fixed to 2 mm for the sides (x and y dimensions) and the height of the element is determined as half the thickness of a single layer, due to the necessity to fit two of them in each layer to achieve better results, hence the height is 0.38 mm (figure 2.3).

Then, the model is created with slightly different dimensions from the real specimen to simplify the creation of the mesh, the new dimensions are 104×456 mm, which mean that are created a total of 94848 elements.

The cut damage can be considered as an area where the stresses are not transmitted by the fibers but only by matrix, so it can be modelled with an area of only epoxy resin, as shown in the right figure 2.4. The damage is discretized as a rectangular of one element in the y-direction and n elements in the x-direction, depending on its extension. The elements belonging to to the damaged area are associated with an isotropic material with $E = 4.6\ MPa$, $\nu = 0.3$ and $\rho = 1300\ kg/m^3$, while the other elements are associated to an orthotropic material with $E_1 = 64121\ MPa$, $E_2 = 64121\ MPa$, $E_3 = 4.6\ MPa$, $G_{12} = 5722\ MPa$, $\nu_{12} = 0.031$ and $\rho = 1570\ kg/m^3$.

To simulate the load conditions of a universal testing machine (left figure 2.4) is applied a clamped constrain (*blue*) to the lower border of the model, while on the upper border is imposed a displacement (*red*).

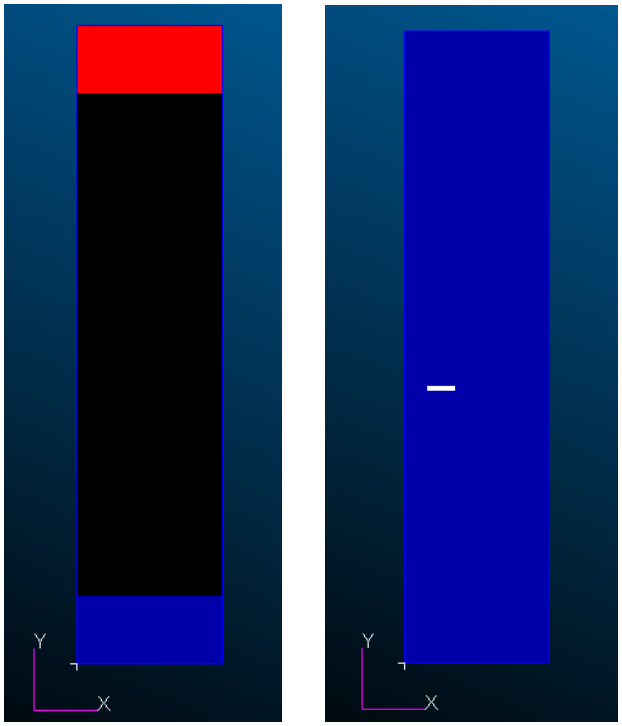

Figure 2.4: Load conditions (center) and damage model (right)

2.2 Medium Fidelity Model

The medium fidelity model is designed to be a less accurate model but faster to compute compared to the high fidelity. In order to do so, the specimen is modelled as a 2D plate and then meshed with 2D elements in particular QUAD [16] [17] elements (figure 2.5) which are less expensive to calculate compared to HEXA because they have less DOF. There are two common types of these elements QUAD4 and QUAD8, the former is a linear element while the latter is quadratic. In this thesis the specimen is not subjected to bending loads so the we decided to

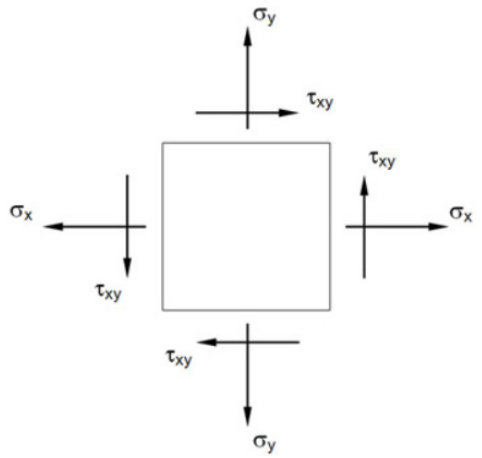

Figure 2.5: Stress components on QUAD 4 elements [19]

use QUAD4 elements with 1 mm side (figure 2.6. Then, the model is created with the same dimensions of the high fidelity which means that are created a total of 47424 elements.

The cut damage is simulated the same way of the high fidelity with the only difference being in the two rows of elements in the y-direction to model the same damage of the high fidelity. The load conditions are the same of the high fidelity. To the elements is assigned the property of laminate with the lamination stacking of the layers and the mechanical properties resumed in table 2.1. The elements belonging to the damage area are associated to the property of laminate with the third layer assigned to the matrix material with $E = 4.6\ MPa$, $\nu = 0.3$ and $\rho = 1300\ kg/m^3$.

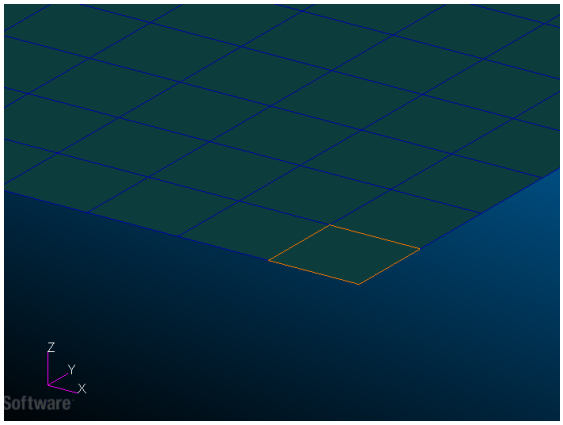

Figure 2.6: Medium fidelity model elements

2.3 Low Fidelity Model

The low fidelity model is designed to be the coarser model of the three but also the fastest to compute. It is created the same way of the medium fidelity model with the difference in dimension of the elements, which have a side of 4 mm. This modification affects the extension of the damage in the y-direction from 2 mm to 4 mm, so the damage in low fidelity is overestimated. The number of elements in this model are 2964.

Chapter 3

Method at glance

In this chapter is resumed the original method developed in this thesis, see figure 3.1. The proposed method is divided in **offline** phase and **online** phase.

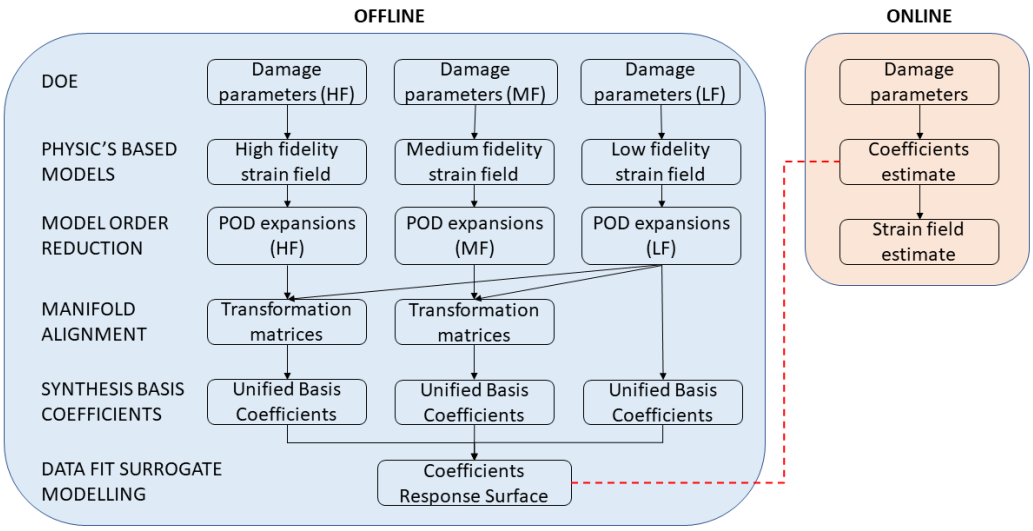

Figure 3.1: Method at glance

3.1 Offline Phase

In the offline phase physic's based models are used to obtain information about the response of the system at the variation of the damage parameters and train a surrogate model for fast data generation.

3.1.1 Design of experiment

The *first* step of the method is the generation of the design parameters

$$X = [x, y, \delta x] \tag{3.1}$$

where x is the position of the lower angle of the cut in the x-axis, y is the position of the lower angle of the cut in the y-axis and δx is the extension of the cut in the x-axis, using the Latin Hypercube Sampling [20] (see section 4.2). This technique allows the best spatial exploration in the range of the design parameters for the given number of variables.
In this step are determined three different sets of design parameters

$$\boldsymbol{X} = [X_1, X_2, ..., X_N] \tag{3.2}$$

one for each fidelity model: $\boldsymbol{X_h}$ for high fidelity, $\boldsymbol{X_m}$ for medium fidelity and $\boldsymbol{X_l}$ for low fidelity. The only constraint that must be respected is that each set must have a number of common design parameters, in order to correlate the models generated from different fidelities.

3.1.2 Physic's Based Model

The *second* step of the method consist in the generation of the strain fields for each fidelity using the Finite Element models designed in chapter 2.
For each triplet of design parameters it is automatically created a new Finite Element model for the desired fidelity using a *Matlab* script. This script modifies the session file that gives instructions to the software *MSC Patran* to produce the information needed to the solver *MSC Nastran*. The output of the solver is a file punch (.pch) containing the data about the strains of each element of the

model. Lastly the punch file is read with another *Matlab* script to obtain the strain field in the y-direction and saving it in the column vector S_i called *snapshot*. The snapshots are then collected it the snapshots matrix

$$\boldsymbol{S} = [S_1, S_2, ..., S_N] \tag{3.3}$$

In the end, from the three set of damage parameters $\boldsymbol{X_h}$, $\boldsymbol{X_m}$ and $\boldsymbol{X_l}$ are obtained the three snapshot matrices $\boldsymbol{S_h}$, $\boldsymbol{S_m}$ and $\boldsymbol{S_l}$.

3.1.3 Model Order Reduction

The *third* step of the method consist in the determination of POD expansions (composed of the POD basis vectors and the POD modal coefficients) of the snapshots matrices using the Proper Orthogonal Decomposition (POD) [9] via Singular Value Decomposition (SVD) [21].
POD is a method for low-order approximation of some high dimensional processes. It provides an efficient way of capturing the dominant components of high-dimensional processes with a small number of basis Φ. It is widely used in the situations where model reduction is required like study of turbulent flows [22], structural dynamics [23] and thermal analysis [24].
For each snapshot matrix $\boldsymbol{S_h}$, $\boldsymbol{S_m}$ and $\boldsymbol{S_l}$, it is first subtracted their mean value (to improve the numerical conditioning) and then is computed the SVD to determine the dominant left singular vectors as the POD basis vectors

$$\boldsymbol{\Phi} = [\Phi_1, \Phi_2, ..., \Phi_N] \tag{3.4}$$

ranked in order of importance by the magnitude of their corresponding singular value. Then through the singular values and the right singular vectors are determined the POD modal coefficients

$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_N] \tag{3.5}$$

Afterwards, the POD basis are truncated evaluating to relative energy associated in each mode in order to retain only the fundamental basis to describe the system. The obtained POD basis vectors are $\boldsymbol{\Phi_h}$, $\boldsymbol{\Phi_m}$ and $\boldsymbol{\Phi_l}$ with the associated POD modal coefficients $\boldsymbol{\alpha_h}$, $\boldsymbol{\alpha_m}$ and $\boldsymbol{\alpha_l}$.

3.1.4 Manifold Alignment

The *fourth* step of the method consist in the calculation, through Procrustes Analysis [25], of the transformation matrices that link high fidelity and medium fidelity POD basis to the low fidelity POD basis.
Manifold alignment techniques are used to transfer information from multiple sources to improve the learning process such as in robotics integrating data from various robots [26], multi-fidelity modelling, automatic translation, image recognition [27] and biomedics [28]. In particular, *Perron et al.* [10] developed a method for multi-fidelity modelling transonic airfoil based on information transfer from low fidelity database to high fidelity database.
In this thesis we developed an **original** method, based on Procrustes Analysis, to transfer information from the high and medium fidelity spaces into the low fidelity space. The choice of using the low fidelity basis as the main basis is due to the fact that it has been produced from the database with the highest number of simulations. This translates in the best spatial investigation of the response of the Physic's based models through the range of the damage parameters.
After implementing this step we obtain the transformation matrices $\boldsymbol{b_{hl}}$, as the scaling matrix, $\boldsymbol{T_{hl}}$, as the orthogonal rotation matrix, and $\boldsymbol{c_{hl}}$, as the translation matrix for the high fidelity POD basis $\boldsymbol{\Phi_h}$ projected to the low fidelity basis $\boldsymbol{\Phi_l}$ and the transformation matrices $\boldsymbol{b_{ml}}$, $\boldsymbol{T_{ml}}$ and $\boldsymbol{c_{ml}}$ or the medium fidelity POD basis $\boldsymbol{\Phi_m}$ projected to the low fidelity basis $\boldsymbol{\Phi_l}$.

3.1.5 Synthesis Basis Coefficients

The *fifth* step of the method consist in the calculation of the unified basis coefficients, which are the POD modal coefficients of the high and medium fidelity databases, $\boldsymbol{\alpha_h}$ and $\boldsymbol{\alpha_m}$, projected into the low fidelity POD basis $\boldsymbol{\Phi_l}$ using the transformation matrices $\boldsymbol{b}$, $\boldsymbol{T}$ and $\boldsymbol{c}$. This step is implemented using a mathematical function developed during the the thesis, which returns the projected high fidelity POD modal coefficients $\boldsymbol{\beta_h}$ and the projected medium fidelity POD modal coefficients $\boldsymbol{\beta_m}$. The $\boldsymbol{\beta_h}$, $\boldsymbol{\beta_m}$ and $\boldsymbol{\alpha_l}$ lie in the same space, so it is possible to compare them each other to determine the differences between the three fidelities. These differences $\boldsymbol{\delta_{hm}}$ between $\boldsymbol{\beta_h}$ and $\boldsymbol{\beta_m}$, and $\boldsymbol{\delta_{ml}}$, between $\boldsymbol{\beta_m}$ and $\boldsymbol{\alpha_l}$ are used to correct the $\boldsymbol{\alpha_l}$.

3.1.6 Data Fit Surrogate Modelling

The *sixth* step of the method consist in training the surrogate models to obtain the coefficients response surface. For each coefficient α_{l_i}, δ_{hm_i} and δ_{ml_i} is determined a different response surface using a machine learning algorithm trained with the $\boldsymbol{\alpha_l}$, $\boldsymbol{\delta_{hm}}$, $\boldsymbol{\delta_{ml}}$ which represent the training database. In literature there are several methods to train a response surface, we decided to focus on four methods Gaussian Process regression (GP) [29], CoKriging [8], Regression Tree [13] and Self Organizing Maps (SOMs)[3].
The results obtained from the trained models were not satisfactory for the localization of the damage, so it is determined a fifth method based on Regression Trees and SOMs which has given acceptable results.
In this method the α_{l_i} response surfaces are obtained using Regression Trees, while the corrective coefficients, δ_{hm_i} and δ_{ml_i}, response surfaces are produced using SOMs. With this step is concluded the mapping of the Damage parameters to the unified basis coefficients.

3.2 Online Phase

In the online phase the surrogate model is used for estimate a strain field related to input damage parameters.
The Damage parameters are given to the surrogate model determined in section 3.1.6 which estimates the POD modal coefficients. These coefficients are then multiplied to the low fidelity POD basis vectors and then is added the mean value of the low fidelity database to estimate the Strain field associated to the new Damage prameters.

Chapter 4

Design Of Experiment

In this chapter we determined the Damage parameters for the creation of the databases to implement a multi-fidelity approach.
The choice of the Damage parameters is a decisive factor for the creation of the surrogate model. In [6], are described various critical issues which may arise if the sampling process is done incorrectly.
Firstly, the surrogate models depend on the observations used to build them, so a sampling plan must take in account a possible critical behavior of studied system and must explore all the design parameters selected range.
Secondly, the number of design variables must be selected accurately to avoid variables that have little to none impact on the response of the model.
Thirdly, the number of variables also affects the number of samples needed to build an accurate prediction, as example, if a certain prediction accuracy for one design variable is obtained using N samples, then for k variables are necessary N^k samples. This is referred as *"dimensionality course"*. Taking into account these problems we preceded into the selection of the Damage Parameters.

4.1 Damage Parameters

During the development of the thesis, have been tested three combinations of Damage parameters.
At the beginning, we created models with **5** design parameters, which were:

- **x**, representing the position of the lower left angle of the matrix region in the x axis;
- **y**, representing the position of the lower left angle of the matrix region in the y axis;
- $\boldsymbol{\delta x}$, representing the extension of the cut in the x axis;
- **layer**, representing the number of layers affected by the cut from the first layer to the fourth;
- **displacement**, representing the displacement imposed;

Therefore the input vector was in the form of

$$X = [x, y, \delta x, layer, displacement] \tag{4.1}$$

The results obtained from the first model were not satisfactory because it was possible to localize the cut damage in the validation models however the values of the strains in that area were underestimated by a magnitude of $50 \sim 100$. This issue was brought back to the displacement variable, it introduced elevated variability in the peak strains' values which were not possible to characterize with the number of models created. So we incurred in the dimensionality course.
Thus, a new model was designed with **4** parameters, leaving the *displacement* fixed at 5 mm.

$$X = [x, y, \delta x, layer] \tag{4.2}$$

In this case, with the results it was possible to recreate the cut with the appropriate values of strain. However, this model was useful only for exposed cut damages, because the layers affected started from the outer and increased up until the other outer.
To recreate a model able to estimate the internal damages of the structure, the *layer* variable was fixed, positioning the cut only in the third layer. With this update, there were maintained **3** design parameters:

- **x**, with a range of $[0 : 104 - \delta x]$ mm;
- **y**, with a range of $[60 : 394]$ mm;

- $\boldsymbol{\delta x}$, with a range of $[10 : 48]$ mm;

The *displacement* was maintained at 5 mm. In the end, the input vector representing the **Damage parameters** for each simulation is created as

$$X = [x, y, \delta x] \tag{4.3}$$

4.2 Latin Hypercube Sampling

In this section is illustrated the implementation of the Latin Hypercube Sampling (LHS) as the sampling plan used for the creation of the Damage parameters.
LHS[20] is a statistical method to generate quasi-random samples of parameters values. It is widely used in Monte Carlo simulations (which utilize randomized variables to statistically analyze the outputs of the problem and analyzing its trends) to reduce the number of runs necessary to achieve a reasonably accurate result.
LHS is based on the Latin square design (figure 4.1), which has a single sample in each row and column, and is generalized to an arbitrary number of dimensions (hyper-planes).

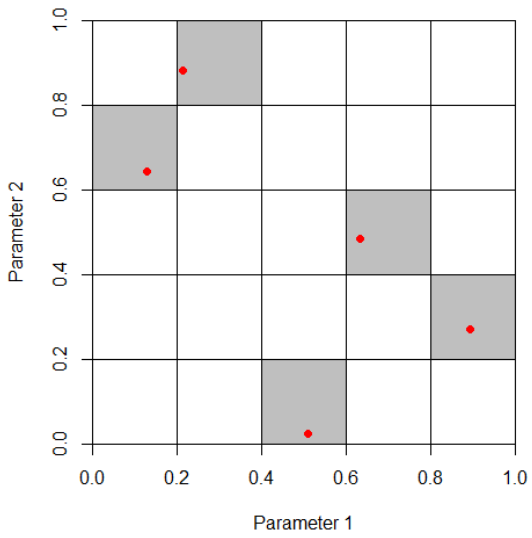

Figure 4.1: Example of Latin square design [30]

To perform the sampling, the cumulative probability (100 %) is divided into n

(number of samples) segments, then a probability is randomly picked within each segment using a uniform distribution, and then mapped to the correct representative value in the actual distribution.

4.3 Database Generation

The generation of the various databases is done automatically exploiting the advantages of using *MSC Patran*, which integrates a proprietary programming language called Patran Command Language (PCL). Starting from the physics's based models session files (*.ses*), which contain the instructions to create the models in PCL, we parametrized the commands in order to automatically generate a new model for every Damage parameter used as input.
In order to generate the models it is implemented a *Matlab* script which determines the various Damage parameters using Latin Hypercube Sampling (LHS) space filling. Then it updates the parameters of the session files, maintaining constant the others parameters (geometry,materials and constrains). Afterwards, the generated models are analyzed with the program MSC Nastran. The time required for each simulation is summarized in table 4.1.

The results of the analysis are collected in a *punch* file (*.pch*), which is parsed

	Time
High fidelity	1615 s
Medium fidelity	175 s
Low fidelity	20 s

Table 4.1: Fem analysis time with chipset Intel 7700HQ and 16 GB of RAM

using another *Matlab* script to obtain the value of the strains in each layer for each element, in the y-direction, and saving them in the columns S of the *snapshots* matrix (see chapter 5).

$$\mathbf{S} = [S_1, S_2, ..., S_N] \tag{4.4}$$

A snapshot example is shown in figure 4.2. The snapshots obtained from different fidelity have different dimensionality due to the variation of the dimensions and the type of the elements, hence, to compare them, it is necessary to manipulate

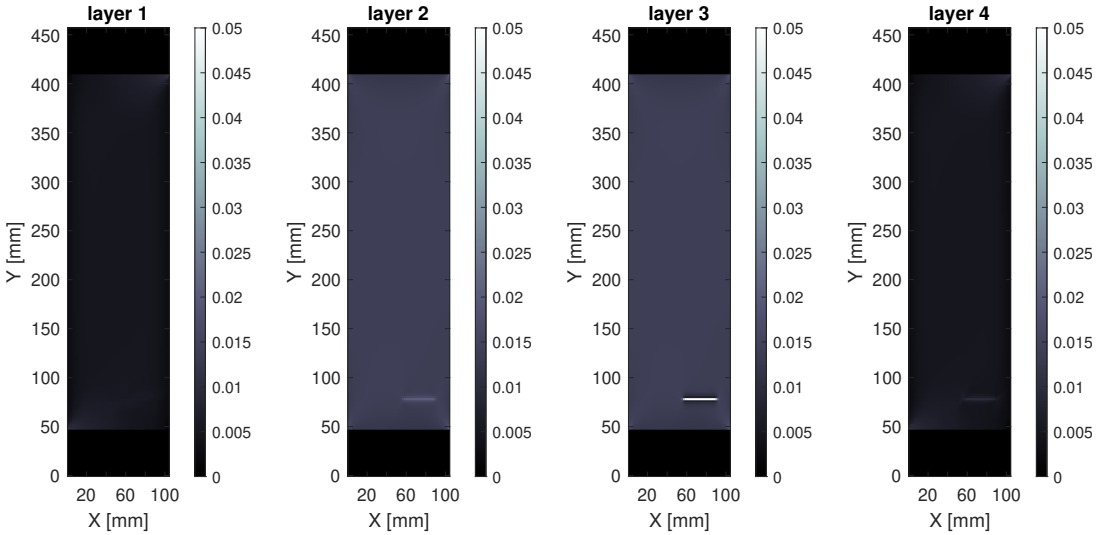

Figure 4.2: Example of High fidelity model snapshot

the data as follows:

- in the high fidelity snapshot, it was taken the average value between the upper and lower element of the same layer;
- in the medium fidelity snapshot, for every layer, it was taken the average value between the four elements corresponding to the same element in the high fidelity grid;
- in the low fidelity snapshot, for every layer, every element was split in four elements and its value has been assigned to all of them;

The Normalized Root Mean Squared Error (NRMSE) between the three fidelities for 100 samples, calculated as

$$NRMSE = \frac{\sqrt{\sum_{i=1}^{n_e} (\boldsymbol{S_h} - \boldsymbol{S_m})^2}}{\sqrt{n_e}\,(\boldsymbol{S_{h_{max}}} - \boldsymbol{S_{h_{min}}})} \tag{4.5}$$

where S_h and S_m are the strain values form the high fidelity and the medium fidelity and n_e is the number of elements of each snapshot. The NRMSE of layer 3 between the high and medium snapshots (*left* and between the medium and low snapshots (*right*) is displayed in figure 4.3.

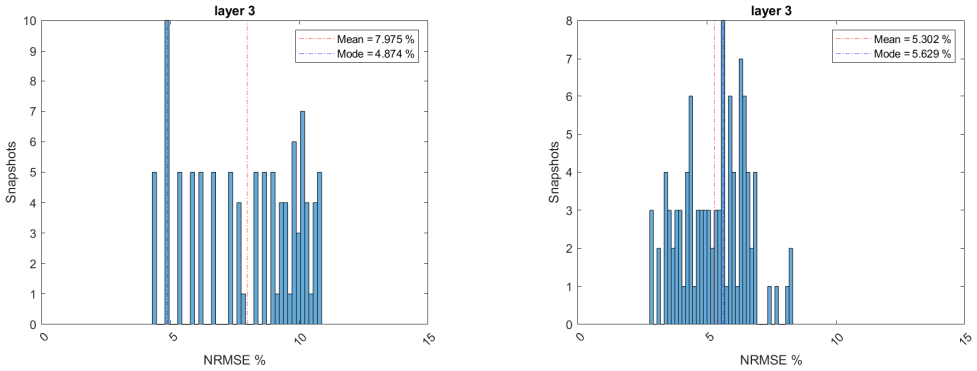

Figure 4.3: RMSE between high and medium fidelity (left) and between medium and low fidelity (right)

Using the described automatic scripts are created a **training** and a **validation** database. The former is composed by:

- $N_H = 100$ high fidelity simulations;
- $N_M = 200$ medium fidelity simulations, with N_H simulations made with the same parameters of the high fidelity ones;
- $N_L = 1500$ low fidelity simulations with N_M simulations made with the same parameters of the medium fidelity ones;

While the validation database is made with only $N_{HV} = 100$ high fidelity simulations.

Chapter 5

Model Order Reduction

After the creation of the database with all the simulations, the issue of finding a method to create the data for different inputs emerged. In fact it wasn't feasible to train an accurate machine learning method with an input of only three variables and an output of over ten thousand. So, it was necessary to find a Reduction Method to decrease the information needed to reconstruct every model. In the next section there is a brief introduction to these methods.

5.1 Introduction

Computational modelling and simulation of nonlinear complex systems with millions of degrees of freedom (DOFs), like finite elements or finite difference, are in high demand with computational and storage resources. When multiple analysis are required to investigate the behaviour of a system, to reduce the computational time minimizing the information losses, it is necessary to condense the information in lower order models that can serve as the basis for additional analysis. These lower order models are called reduced order models (ROMs). The ROMs capture the essential behaviour of the phenomena reducing the DOFs. The high-dimensional solution of a given model is thus reduced to a linear combination of few dominant modes.

In [31] and [32], is proposed a classification between Model Order Reduction methods which can be divided in two categories: methods for linear systems

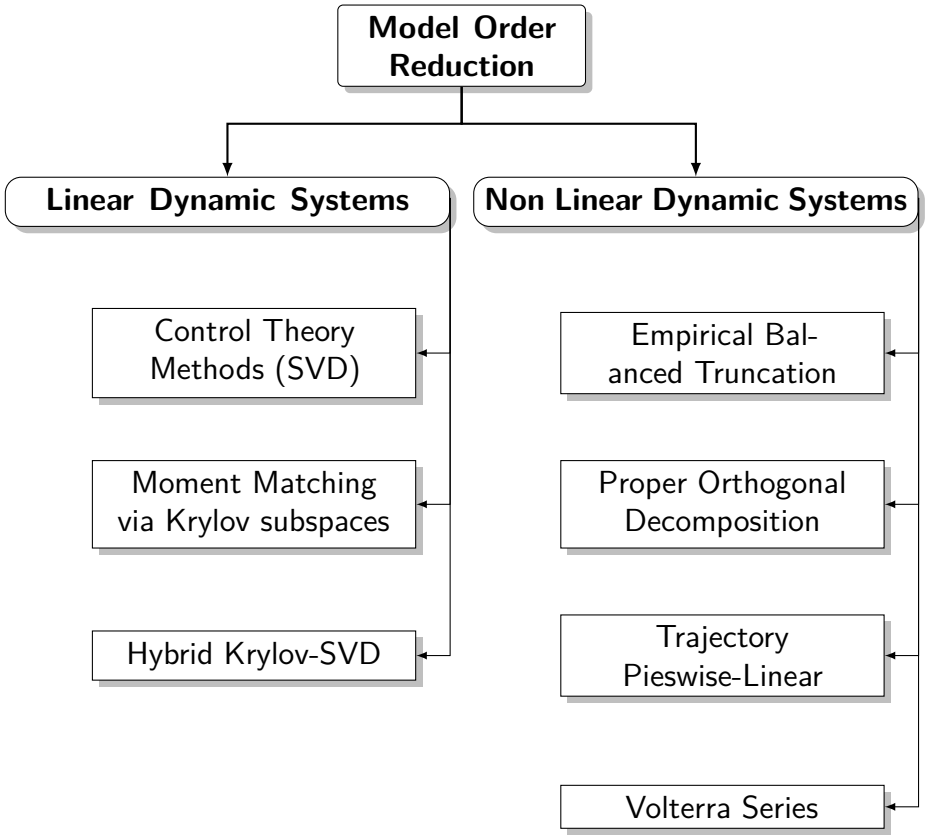

Figure 5.1: Model Order Reduction classification

(Control Theory methods, moment matching via Krylov subspaces and hybrid Krylov-SVD methods) and for non-linear systems (Empirical Balanced Truncation, Proper Orthogonal Decomposition, Trajectory Pieswise-Linear and Volterra Series). Control theory methods can be used to reduce complex systems matrices to lower rank matrices using their singular values to determine the dimension of the reduced model. They are difficult to implement with high rank problems due to the necessity of calculation of the SVD (computational complexity of n^3).

Krylov subspaces moment matching methods [33] are implemented in reductions of large quantity of data. The moments are defined as the coefficients of the Taylor series expansion of the transfer function around a certain complex frequency and the reduction is performed eliminating the higher order components. The main advantage of this approach is that it involves the solution of sparse linear systems of equations and so results require low computational effort and small memory

requirements. The main disadvantage of Krylov methods is that they do not have a global error estimate.
In case of non-linear dynamic systems, the field of model reduction remains a big challenge mainly because the reduction of the non-linear system approximates the original for a single input and the calculation of the reduced order model requires the simulation of the original model. So, the calculation of the ROM can require more time than the original, hence it's utilized for re-obtaining the data already calculated.
The Proper Orthogonal Decomposition (POD) [9] is a very popular method, which projects the original system into a basis subspace with smaller dimension. In this thesis we will focus on the Proper Orthogonal Decomposition (POD) method, so a detailed explanation will be given in the next chapter.

5.2 Proper Orthogonal Decomposition

Proper Orthogonal Decomposition was first introduced by Pearson in 1901 [9] and then revised by Lumley [34] and Sirovich [35]. POD is an efficient way to reduce system complexity, it can capture the essential behaviour of the studied phenomena using a database of already known solutions and identify a lower order subspace composed by orthonormal basis.
A generic vector S which can be written as a linear combination of some basis

$$S = \sum_{i=1}^{N} \alpha_i \varphi_i = \boldsymbol{\alpha}\boldsymbol{\Phi} \tag{5.1}$$

where $\boldsymbol{\alpha}$ is the coefficients vector and $\boldsymbol{\Phi}$ is the basis matrix such as $\boldsymbol{\Phi} = [\phi_1, \phi_2, ..., \phi_N]$, where ϕ_i are the basis vectors called Proper Orthogonal Modes (POMs). From 5.1, a reduced version of the model can be represented with

$$S \approx \sum_{i=1}^{n} \alpha_i \varphi_i = \boldsymbol{\alpha_n}\boldsymbol{\Phi_n} \tag{5.2}$$

where $n < N$. The Φ_n matrix is composed by the most representative POMs for the reconstruction of the system and grants the approximation to the original model

through the lest mean square

$$min \left\| S - \sum_{i=1}^{n} \alpha_i \varphi_i \right\|_{L2} \tag{5.3}$$

The issue is the determination of the basis which create the matrix $\boldsymbol{\Phi}$. We can introduce the auto-correlation matrix $\boldsymbol{C} \in \mathbb{R}^{N \times N}$ such as

$$\boldsymbol{C} = \sum_{i=1}^{N} S_i S_i^T \tag{5.4}$$

which is symmetric, positive definite with real positive eigenvalues sorted according to $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ... \geq \lambda_N > 0$ called proper orthogonal values (POVs). To obtain the eigenvectors associated to the reduced model, it is necessary to solve the problem

$$\boldsymbol{C}\varphi_i = \lambda_i \varphi_i \qquad i = 1, ..., N \tag{5.5}$$

Solving the eigenvalue problem 5.5 is in general computationally intractable because the dimension of the matrix $\boldsymbol{C}$ is usually large and this matrix is usually dense. So, it was developed the snapshots method [35]. Where the response of the system, for each input, is collected in a column vector S_i (called snapshot) and collected in the snapshots matrix $\boldsymbol{S} \in \mathbb{R}^{N \times N_{snap}}$

$$\boldsymbol{S} = \left[S_1, S_2, ..., S_{N_{snap}} \right] \tag{5.6}$$

where usually $N_{snap} \ll N$. Then $\boldsymbol{C}$ can be replaced with the covariance matrix $\overline{\boldsymbol{C}} \in \mathbb{R}^{N \times N}$

$$\overline{\boldsymbol{C}} = \boldsymbol{S}\boldsymbol{S}^{\boldsymbol{T}} \tag{5.7}$$

The non-zero eigenvalues of $\overline{\boldsymbol{C}}$ are the same of those of the matrix

$$\widehat{\boldsymbol{C}} = \boldsymbol{S}^{\boldsymbol{T}}\boldsymbol{S} \tag{5.8}$$

with significant lower dimension $\widehat{\boldsymbol{C}} \in \mathbb{R}^{N_{snap} \times N_{snap}}$. So, the eigenvectors (POMs) can be obtained solving

$$\boldsymbol{S}^{\boldsymbol{T}}\boldsymbol{S}\varphi_i = \lambda_i \varphi_i \qquad i = 1, ..., N_{snap} \tag{5.9}$$

5.3 Singular Value Decomposition

Singular Value Decomposition (SVD) [21] is one of the most used methods to obtain orthonormal basis. The SVD factorizes the matrix $\boldsymbol{S}$ into three matrices

$$\boldsymbol{S} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} \tag{5.10}$$

where $\boldsymbol{U} \in \mathbb{R}^{N_{snap} \times N_{snap}}$ is the left singular vectors matrix, $\boldsymbol{V} \in \mathbb{R}^{N \times N}$ is the right singular vectors matrix and $\boldsymbol{\Sigma} \in \mathbb{R}^{N_{snap} \times N}$ is the singular values (σ) diagonal matrix. The auto-correlation matrix $\overline{\boldsymbol{C}}$ and the covariance matrix $\overline{\boldsymbol{C}}$ can be written as

$$\overline{\boldsymbol{C}} = \boldsymbol{S}\boldsymbol{S}^{\boldsymbol{T}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}}\boldsymbol{V}\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{U}^{\boldsymbol{T}} = \boldsymbol{U}\left(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\boldsymbol{T}}\right)\boldsymbol{U}^{\boldsymbol{T}} \tag{5.11}$$

$$\widehat{\boldsymbol{C}} = \boldsymbol{S}^{\boldsymbol{T}}\boldsymbol{S} = \boldsymbol{V}\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\boldsymbol{T}} = \boldsymbol{V}\left(\boldsymbol{\Sigma}^{\boldsymbol{T}}\boldsymbol{\Sigma}\right)\boldsymbol{V}^{\boldsymbol{T}} \tag{5.12}$$

Hence, from the SVD is possible to determine $\boldsymbol{U}$ as the left eigenvectors of $\overline{\boldsymbol{C}}$, $\boldsymbol{V}$ as the right eigenvectors of $\widehat{\boldsymbol{C}}$ and $\boldsymbol{\Sigma^2}$ as the eigenvalues of $\boldsymbol{S}$.
A relevant characteristic of the SVD is the determination of relative importance of the i^{th} POM in the approximation of the matrix $\boldsymbol{S}$. It is determined by the relative energy of that mode ([36],[37])

$$E_i = \frac{\sigma_i^2}{\sum_{i=1}^{N} \sigma_i^2} = \frac{\lambda_i}{\sum_{i=1}^{N} \lambda_i} \tag{5.13}$$

and it is possible to determine the relative energy maintained during the k^{th} reduction of the original basis as

$$E_{retained} = \sum_{i=1}^{k} E_i \tag{5.14}$$

Typically, the number of modes required to capture all of the energy is very large and it does not result in a significant dimensionality reduction. However, it is possible to accurately approximate a matrix to 99% of its energy by using a significant lower number of POMs .

5.4 Implementation

To obtain the ROM of the snapshots matrix $\boldsymbol{S} = [S_1, S_2, .., S_N]$, where S_i is the column vector containing the strain data of every element for a single simulation, at the start $\boldsymbol{S}$ is centered subtracting the row vector S_m containing its the average value

$$\widehat{\boldsymbol{S}} = \boldsymbol{S} - S_m \tag{5.15}$$

Then, the SVD of $\widehat{\boldsymbol{S}}$ is calculated (see table 5.1) to obtain $\boldsymbol{U}$ (left eigenvectors matrix), $\boldsymbol{\Sigma}$ (eigenvalues matrix) and $\boldsymbol{V}$ (right eigenvectors matrix). Afterwards,

	Time			
	layer 1	layer 2	layer 3	layer 4
High fidelity	39.5 s	40.9 s	43.4 s	36.6 s
Medium fidelity	35.4 s	36.2 s	37.5 s	33.9 s
Low fidelity	29.5 s	30.3 s	26.9 s	20.6 s

Table 5.1: SVD time with chipset Intel 7700HQ and 16 GB of RAM

using

$$E_{retained} = \sum_{i=1}^{k} \left(\frac{\lambda_i}{\sum_{i=1}^{N} \lambda_i} \right) \tag{5.16}$$

it is possible to determine the number of POD modes k necessary to preserve a determinate relative energy . An example is shown in figure 5.2, where are displayed the POD eigenvalue spectra, and in figure 5.3, where the dimensionality reduction to retain the 99% in the third layer for the high fidelity database is only of 10 modes, however in the low fidelity database the reduction is more than half of the total modes.

In table 5.2 are displayed the number of POD modes retained for each fidelity model.

Finally it is possible to reconstruct the reduced snapshot as

$$\tilde{S}_i(x) = S_m + \sum_{j=1}^{k} \alpha_j^i(x)\varphi_j^i \qquad i = 1, ..., M \tag{5.17}$$

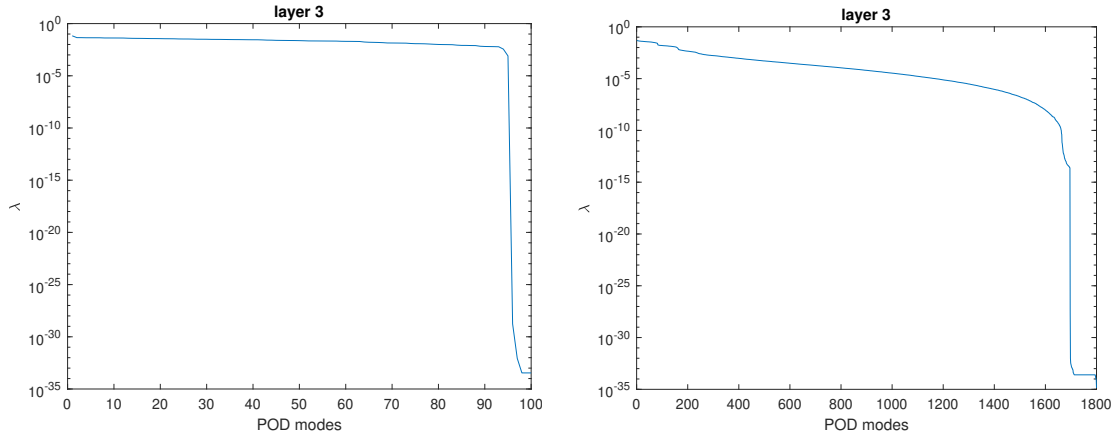

Figure 5.2: POD eigenvalues for the computed modes in the 3^{rd} layer for high fidelity models left and low fidelity models right

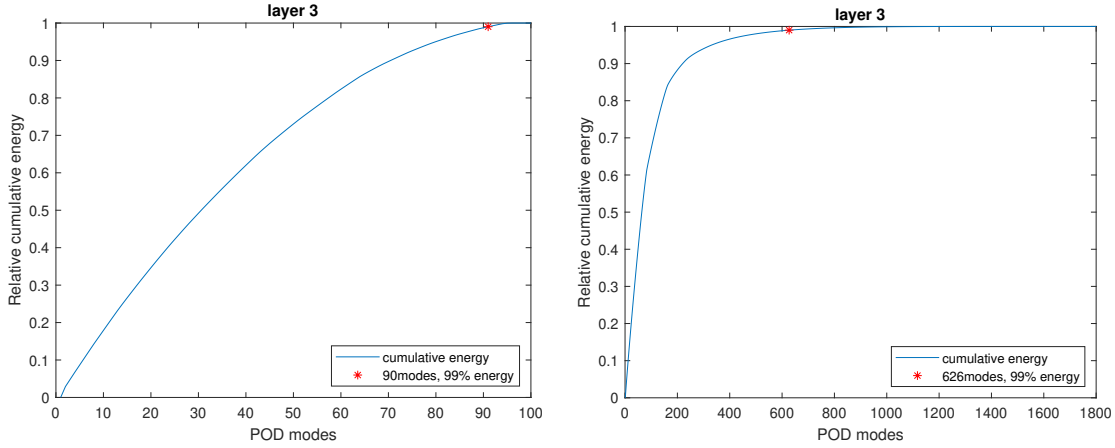

Figure 5.3: Cumulative energy retained in the 3^{rd} layer for high fidelity models (left) and low fidelity models (right)

where M is the number of elements of the model, α_j^i is the POD modal coefficient for the φ_j^i component of the POD basis vector and x is the vector of the design variables. Figure 5.4 and 5.5 display the truncation error for the three fidelities calculated as

$$NRMSE = \frac{\sqrt{\sum_{i=1}^{n_e} \left(\tilde{\boldsymbol{S}} - \boldsymbol{S}\right)^2}}{\sqrt{n_e}\left(\boldsymbol{S_{max}} - \boldsymbol{S_{min}}\right)} \tag{5.18}$$

n_e is the number of elements of each column. For each model the error is comprised between 0 and 1 % with a mean value lower than 0.5%.

	POD modes			
	layer 1	layer 2	layer 3	layer 4
High fidelity	88	90	90	89
Medium fidelity	259	261	261	259
Low fidelity	657	637	626	598

Table 5.2: POD modes retained

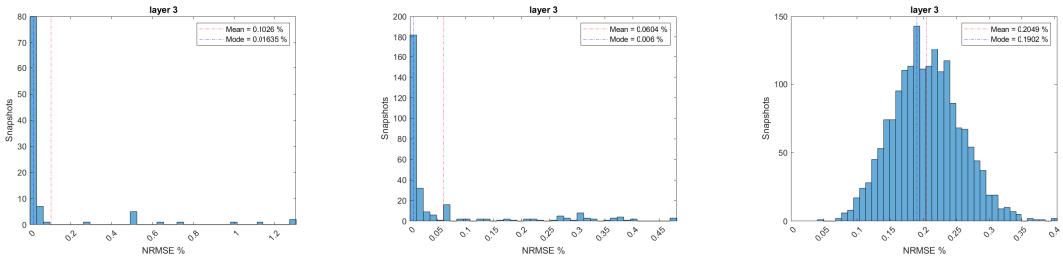

Figure 5.4: Normalized rms error (NRMSE) for 99% of cumulative energy retained in the 3^{rd} layer for high fidelity models (left), medium fidelity models (center) and low fidelity models (right)

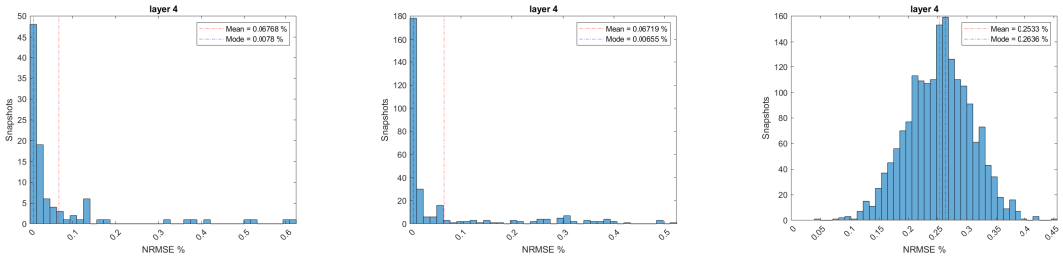

Figure 5.5: Normalized rms error (NRMSE) for 99% of cumulative energy retained in the 4^{th} layer for high fidelity models (left), medium fidelity models (center) and low fidelity models (right)

Despite the determination of $\mathbf{\Phi}$ as the POD basis matrix which describe the entire phenomenon as

$$\mathbf{\Phi} = [\varphi_1, \varphi_2, ..., \varphi_k] = \boldsymbol{U}[:, 1:k] \tag{5.19}$$

it is possible to directly determine POD modal coefficients

$$\boldsymbol{\alpha} = [\alpha_1, \alpha_2, ..., \alpha_k] = \left(\boldsymbol{\Sigma}\boldsymbol{V}^T\right)[1:k, :] \tag{5.20}$$

only for the x inputs of $\widehat{\boldsymbol{S}}$. Hence, it is mandatory to determine a regression model to obtain correct α for a random input in the allowed range (see chapter 7).

Chapter 6

Manifold Alignment

POD modal coefficients matrices obtained in the previous chapter for each fidelity database are not compatible with each other because they lay in different spaces. So, to transfer information from different fidelity models we must first find a method to determine how the basis that characterize each of them are connected.
As described in [38], in mathematics, "a manifold is a topological space that locally resembles Euclidean space near each point". Manifolds can be many types, and euclidean space is an example of a particular kind of manifold. Hence, the POD basis can be considered manifolds and, to connect them, exist a family of machine learning methods called **manifold alignment** algorithms which produce projections between two sets of data, under the assumption they are correlated. Manifold alignment methods are used to accelerating learning in robotics integrating data from various robots [26], multi-fidelity modelling [10], automatic translation, image recognition and biomedics. In [28] is proposed a classification in these methods. In particular, *Perron et al.* [10] developed a method for multi-fidelity modelling transonic airfoil based on information transfer from low fidelity database to high fidelity database.
In this thesis we developed an original method, based on Procrustes Analysis, to transfer information from the high and medium fidelity spaces into the low fidelity space. The choice of using the low fidelity basis as the main basis is due to the fact that it has been produced from the database with the highest number of simulations. This translates in the best spatial investigation of the response of the

Physic's based models through the range of the damage parameters.
In this thesis we will focus on Procrustes analysis.

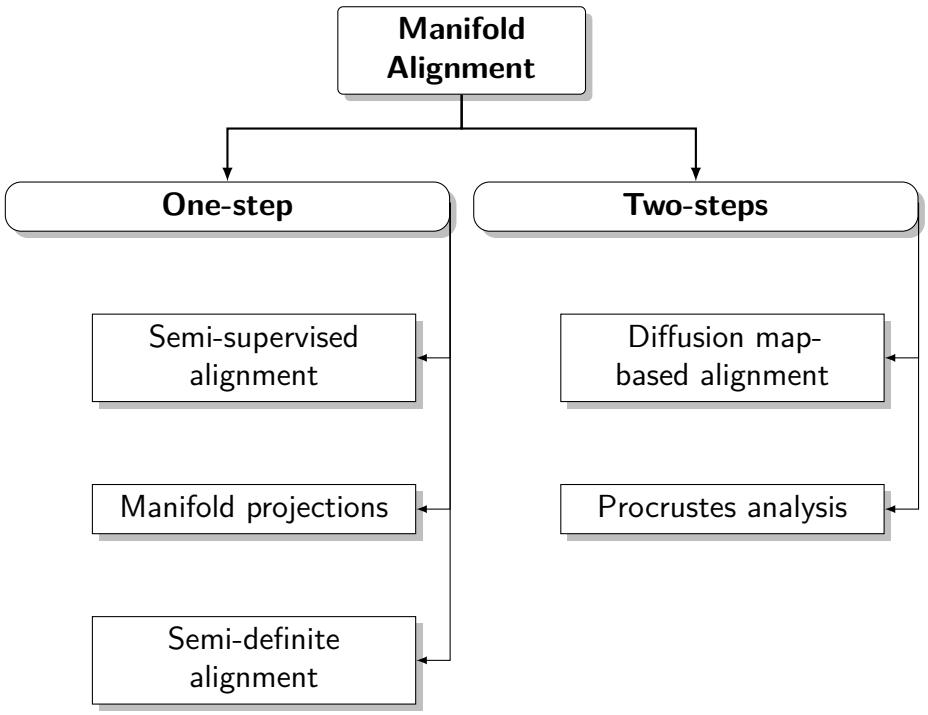

Figure 6.1: Manifold Alignment classification

6.1 Procrustes Analysis

The name Procrustes refers to a bandit from Greek mythology who made his victims fit his bed either by stretching their limbs or cutting them off. Similarly the Procrustes analysis [25] determines a linear transformation, shown in eq 6.1, composed of translation $\boldsymbol{c}$, orthogonal rotation $\boldsymbol{T}$ and scaling $\boldsymbol{b}$ of the points in the matrix $\boldsymbol{Y}$ to best conform them to the points in the matrix $\boldsymbol{X}$. In [11] are presented the steps to apply the method.

$$\boldsymbol{X} \simeq \boldsymbol{bYT} + \boldsymbol{c} \tag{6.1}$$

In the first step is performed the **translation** of the matrices to a common centre. Hence is calculated the centroid of every column as

$$\overline{X} = \left[\begin{array}{ccc} \frac{1}{n}\sum_{i=1}^{n}\varphi_{1i} & \dots & \frac{1}{n}\sum_{i=1}^{n}\varphi_{mi} \end{array} \right] \tag{6.2}$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{n\times m}$, and it is subtracted from each element in the matrix $\boldsymbol{U}$. Considering the truncated basis obtained in chapter 5:

- $\boldsymbol{\Phi_h}$, representing the basis of the high fidelity database;
- $\boldsymbol{\Phi_l}$, representing the basis of the low fidelity database;

the centered basis are

$$\overline{\boldsymbol{\Phi}}_{\boldsymbol{h}} = \boldsymbol{\Phi}_{\boldsymbol{h}} - \begin{bmatrix} \overline{X}_h \\ \vdots \\ \overline{X}_h \end{bmatrix} \qquad\qquad \overline{\boldsymbol{\Phi}}_{\boldsymbol{l}} = \boldsymbol{\Phi}_{\boldsymbol{l}} - \begin{bmatrix} \overline{X}_l \\ \vdots \\ \overline{X}_l \end{bmatrix} \tag{6.3}$$

In the second step is performed the **scaling** of the matrices using the Frobenius norm such as

$$\hat{\boldsymbol{\Phi}}_{\boldsymbol{h}} = \frac{\overline{\boldsymbol{\Phi}}_{\boldsymbol{h}}}{\left\|\overline{\boldsymbol{\Phi}}_{\boldsymbol{h}}\right\|_F} \qquad\qquad \hat{\boldsymbol{\Phi}}_{\boldsymbol{l}} = \frac{\overline{\boldsymbol{\Phi}}_{\boldsymbol{l}}}{\left\|\overline{\boldsymbol{\Phi}}_{\boldsymbol{l}}\right\|_F} \tag{6.4}$$

where the Frobenius norm is calculated with

$$\left\|\overline{\boldsymbol{\Phi}}\right\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m}\left|\overline{\varphi}_{ij}\right|} \tag{6.5}$$

The translated and scaled basis are then **rotated**. In this step, firstly is determined the matrix $\boldsymbol{A} \in \mathbb{R}^{N_l\times N_l}$, where N_l is the number of retained modes in $\boldsymbol{\Phi_l}$, to maximize the correlation between the two matrices.

$$\boldsymbol{A} = \hat{\boldsymbol{\Phi}}_{\boldsymbol{l}}^{\boldsymbol{T}}\hat{\boldsymbol{\Phi}}_{\boldsymbol{h}} \tag{6.6}$$

Must be noted that the $\hat{\boldsymbol{\Phi}}_{\boldsymbol{h}}$ has been extended with columns of zeros to match the columns of $\hat{\boldsymbol{\Phi}}_{\boldsymbol{l}}$. Then, in eq 6.7 is calculated the SVD of the matrix $\boldsymbol{A}$ to obtain the left singular vectors matrix $\boldsymbol{L}$, the right singular vector matrix $\boldsymbol{M}$ and the

singular values diagonal matrix $\boldsymbol{D}$.

$$[\boldsymbol{L}, \boldsymbol{D}, \boldsymbol{M}] = \text{svd}(\boldsymbol{A}) \tag{6.7}$$

From this decomposition it is possible to obtain the rotation matrix $\boldsymbol{T}$

$$\boldsymbol{T} = \boldsymbol{M}\boldsymbol{L}^{\boldsymbol{T}} \tag{6.8}$$

the scaling matrix $\boldsymbol{b}$

$$\boldsymbol{b} = \left(\sum_{i=1}^{nl} D_{ii}\right) \frac{\left\|\overline{\boldsymbol{\Phi}}_{\boldsymbol{l}}\right\|_F}{\left\|\overline{\boldsymbol{\Phi}}_{\boldsymbol{h}}\right\|_F} \tag{6.9}$$

and the translation matrix $\boldsymbol{c}$

$$\boldsymbol{c} = \begin{bmatrix} \overline{X}_l \\ \vdots \\ \overline{X}_l \end{bmatrix} - \begin{bmatrix} \overline{X}_{new} \\ \vdots \\ \overline{X}_{new} \end{bmatrix} \tag{6.10}$$

where $\overline{X}_{new}$ is obtained from

$$\overline{X}_{new} = \boldsymbol{b}\overline{X}_h\boldsymbol{T} \tag{6.11}$$

The same steps are also performed between $\boldsymbol{\Phi_m}$, representing the basis of the medium fidelity database and $\boldsymbol{\Phi_l}$.

6.2 Synthesis Basis Coefficients

After obtaining the relations between the POD basis of the three fidelities, it is possible to project the high and medium POD modal coefficients into the low fidelity POD basis. Starting from

$$\boldsymbol{\Phi_l} = \boldsymbol{b}\boldsymbol{\Phi_h}\boldsymbol{T} + \boldsymbol{c} \qquad\qquad \boldsymbol{\Phi_l}\boldsymbol{\beta_h} = \boldsymbol{\Phi_h}\boldsymbol{\alpha_h} \tag{6.12}$$

where $\boldsymbol{\beta} \in \mathbb{R}^{N_l \times N_{snap}}$ is the matrix composed by the high fidelity POD modal coefficients matrix in the low fidelity POD basis. By substituting the left equation

into the right we obtain

$$(\boldsymbol{b}\boldsymbol{\Phi_h}\boldsymbol{T} + \boldsymbol{c})\,\boldsymbol{\beta_h} = \boldsymbol{\Phi_h}\boldsymbol{\alpha_h} \tag{6.13}$$

Both terms are then multiplied with $\boldsymbol{\Phi_h^T}$. Since $\boldsymbol{\Phi_h}$ is an orthogonal matrix, $\boldsymbol{\Phi_h^T}\boldsymbol{\Phi_h} = I$, where I is the identity matrix, the equation 6.13 can be written as

$$\left((\boldsymbol{bT}) + \left(\Phi_h^T \boldsymbol{c}\right)\right)\boldsymbol{\beta} = \boldsymbol{\alpha_h} \tag{6.14}$$

Finally, by solving the linear system in equation 6.14, it is possible to obtain $\boldsymbol{\beta}$. The obtained POD modal coefficients are:

- $\boldsymbol{\beta_h}$, obtained transferring $\boldsymbol{\alpha_h}$ from the high fidelity basis into the low fidelity basis;
- $\boldsymbol{\beta_m}$, obtained transferring $\boldsymbol{\alpha_m}$ from the medium fidelity basis into the low fidelity basis.

In table 6.1 is resumed the calculation time for the Procustes analysis and the coefficients $\boldsymbol{\beta}$.

Figures 6.2 and 6.3 show the NRMS error between the original database and

	Time			
	layer 1	layer 2	layer 3	layer 4
$\boldsymbol{\beta_h}$	0.682 s	0.571 s	0.7 s	0.588 s
$\boldsymbol{\beta_m}$	0.942 s	0.717 s	0.761 s	0.832 s

Table 6.1: Procrustes Analysis and Coefficients alignment time with chipset Intel 7700HQ and 16 GB of RAM

the database reconstructed using the β coefficients and the low fidelity basis. The error maintains values in the order of the 10% and it is higher for the high fidelity snapshots because they have greater value difference with the low fidelity compared to the low fidelity. The coefficients $\boldsymbol{\alpha_h}$ for the high fidelity database can also be projected to the low fidelity basis with an intermediate projection to the medium fidelity basis. This two-step projection should have the same mean error compared to the the one-step method but, as shown in figure 6.4, the mean error is higher due to loss of information in the iterative methods used during the projection.

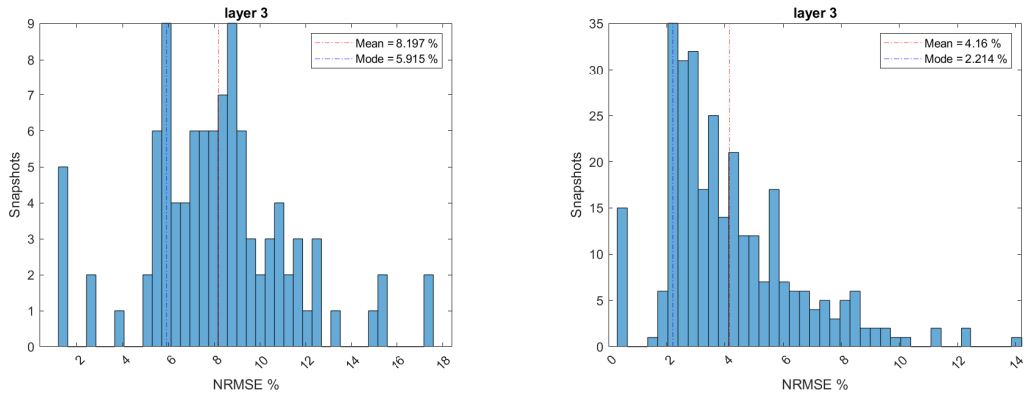

Figure 6.2: 3^{rd} layer Normalized rms error (NRMSE) for high fidelity models (left) and medium fidelity models (right) using POD coefficients projected into the lower basis

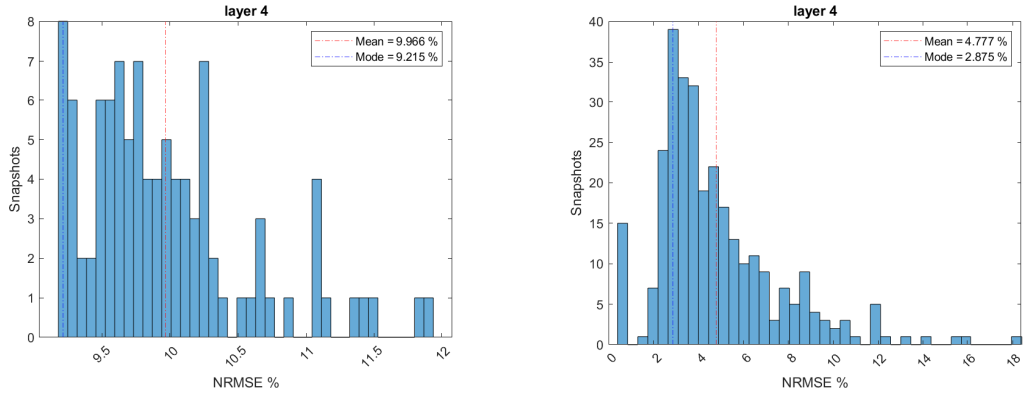

Figure 6.3: 4^{th} layer Normalized rms error (NRMSE) for high fidelity models (left) and medium fidelity models (right) using POD coefficients projected into the lower basis

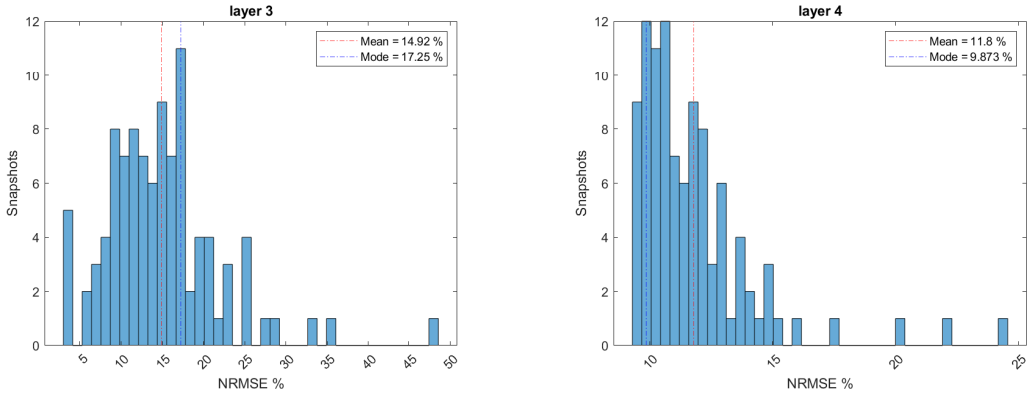

Figure 6.4: Normalized rms error (NRMSE) for high fidelity models 3^{rd} layer (left) and 4^{th} layer (right) using POD coefficients with two-step projection

Chapter 7

Data Fit Surrogate Modelling

In this chapter are trained the surrogate models, created with different Machine Learning (ML) algorithms to obtain the coefficients response surfaces of the unified basis coefficients. ML algorithms are algorithms which can improve automatically through experience and by the use of data [39]. Basically they determine a model using a set of input data, which tries to predict or exploit the behavior of the studied phenomena. These methods can be divided in supervised and unsupervised learning.

In **supervised** learning, a model is trained to obtain the desired outputs using as set of input data, training data, through the iterative optimization of an objective function. Supervised learning can be further divided in classification and regression, in the former the outputs are limited to a finite number of values, while, in the latter the outputs can assume any value in a certain range. Some examples are Gaussian Process regression, CoKriging, Regression Trees, Support Vector Machines and linear regression.

In **unsupervised** learning, a model is trained to extract patterns or structures from a set of input data. Some examples are self organizing maps and k-means.

In this thesis is done a comparison between Gaussian Process regression, CoKriging, Regression Trees and Self Organizing Maps to determine the most suitable, in terms of training time and NRMSE, to be used in non-smooth problems such ours.

To integrate the multi-fidelity information are implemented two machine learning models:

- the first model $\boldsymbol{M_1}$ is trained with the $\boldsymbol{\alpha_l}$ because it has the most number of variables and so the best spatial exploration of the variables;
- the second model $\boldsymbol{M_2}$ is trained with the difference $\boldsymbol{\delta}$ between the $\boldsymbol{\alpha}$ of two different fidelities for the snapshots obtained with the same design variables.

The idea is to improve the online values of $\boldsymbol{\alpha}$ obtained from M_1 with the $\boldsymbol{\delta}$ obtained from M_2 as

$$\boldsymbol{\alpha_h} = \boldsymbol{\alpha_l} + \boldsymbol{\delta_{hm}} + \boldsymbol{\delta_{ml}} \tag{7.1}$$

with $\boldsymbol{\delta_{hm}}$ and $\boldsymbol{\delta_{ml}}$ defined as

$$\boldsymbol{\delta_{hm}} = \boldsymbol{\beta_h} - \boldsymbol{\beta_m} \tag{7.2}$$

$$\boldsymbol{\delta_{ml}} = \boldsymbol{\beta_m} - \boldsymbol{\alpha_l} \tag{7.3}$$

respectively obtained from M_{2hm} and M_{2ml}.

7.1 Gaussian Process Regression

A Gaussian Process can be considered a generalization of a Gaussian probability distribution over infinite possible functions that fit a set of points [29] [12] [40] [41]. It is a supervised non-parametric technique, which means that nearly no assumptions about the mapping function are made. Compared to parametric algorithms, non-parametric algorithms learn more from data. This is because the learning of parametric algorithms may be limited by the assumptions that they make. A Gaussian process is completely defined as a random Gaussian function with a mean function $m(x_i)$ and a covariance function $k(x_i, x_j)$ as

$$f(x) \sim GP(m(x_i), k(x_i, x_j)) \tag{7.4}$$

The mean function $m(x)$ is usually assumed to be zero, while the covariance function $k(x_i, x_j)$ can be any function. The choice of $k()$ make implicit assumptions on the

data modelled, a popular choice is the squared exponential covariance function

$$k(x_i, x_j) = \sigma_f^2 \exp\left[\frac{-(x_i - x_j)^2}{2l^2}\right] \tag{7.5}$$

where σ_f is the maximum allowable covariance which should be high for functions that cover a big interval in the output data. The reliability of the regression model depends on how is fitted the covariance function $k(x_i, x_j)$. This function is parameterized by a set of hyper-parameters, $\boldsymbol{\theta}$, which in this case are σ_f and l. To determine this hyper-parameters it is necessary to maximize the log marginal likelihood

$$\log p\left(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}\right) = -\frac{1}{2}\boldsymbol{y}^{\boldsymbol{T}} K^{-1}\boldsymbol{y} - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi \tag{7.6}$$

K is the covariance matrix as $k(\boldsymbol{x}, \boldsymbol{x})$. The problem of learning with Gaussian processes is exactly the problem of learning $\boldsymbol{\theta}$. This can be done with a standard gradient based technique as long as in 7.6 is possible to determine the partial derivatives.

To predict the output y_* from a new input x_*, it is possible to represent the problem as a Gaussian distribution with mean zero

$$\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \sim G\left(0, \begin{bmatrix} k(\boldsymbol{x}, \boldsymbol{x}) & k(\boldsymbol{x}, x_*) \\ k(x_*, \boldsymbol{x}) & k(x_*, x_*) \end{bmatrix}\right) \tag{7.7}$$

From it is possible to obtain 7.8 which returns the estimated output mean value $\overline{y}_*$

$$\overline{y}_* = k(\boldsymbol{x}, x_*)^T k(\boldsymbol{x}, \boldsymbol{x})\boldsymbol{y} \tag{7.8}$$

An important result of this method is the estimation of the variance of the prediction which is shown in figure 7.1.

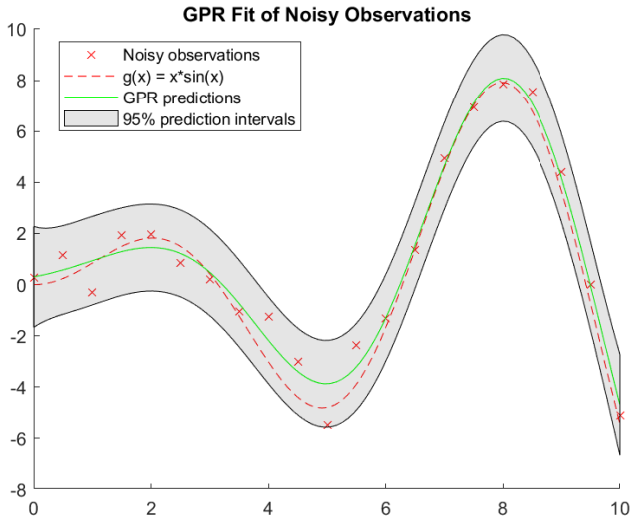

Figure 7.1: Example of variance prediction obtained with GP [42]

7.2 CoKriging

CoKriging [8] is an interpolation model, based on Kriging [43], which correlates two sets of data (high fidelity and low fidelity). In this thesis, it is implemented the *auto-regressive* CoKriging [44], which assumes that the information of low fidelity model can not improve the high fidelity data for the high fidelity design variables. The inputs of the model are the design variables $\boldsymbol{X_k}$ and the training variables $\boldsymbol{\alpha_k}$

$$\boldsymbol{X_k} = \begin{bmatrix} \boldsymbol{X_l} \\ \boldsymbol{X_h} \end{bmatrix} \quad \boldsymbol{\alpha_k} = \begin{bmatrix} \boldsymbol{\alpha_l} \\ \boldsymbol{\beta_h} \end{bmatrix} \tag{7.9}$$

Creating a CoKriging model can be interpreted as constructing two Kriging models in sequence, the first is constructed with the low fidelity data $(\boldsymbol{X_l}, \boldsymbol{\alpha_l})$, while the second is constructed with $(\boldsymbol{X_h}, \boldsymbol{\alpha_d})$, calculated as

$$\boldsymbol{\alpha_d} = \boldsymbol{\beta_h} - \rho\hat{\boldsymbol{\alpha}}_l \tag{7.10}$$

where $\hat{\boldsymbol{\alpha}}_l$ is the estimated mean of the low fidelity Kriging model. The prediction can be written as

$$\hat{\alpha} = M\gamma + r(x)\boldsymbol{\Psi}^{-1}(\boldsymbol{\alpha_k} - F\gamma) \tag{7.11}$$

where $r(x)$ is

$$r(x) = \begin{bmatrix} \rho\sigma_l^2 r_l(x) & \rho^2\sigma_l^2 r_l(x, \boldsymbol{X_h}) + \sigma_d^2 r_d(x) \end{bmatrix} \tag{7.12}$$

$\boldsymbol{\Psi}$ is

$$\boldsymbol{\Psi} = \begin{bmatrix} \sigma_l^2 \boldsymbol{\Psi_l} & \rho\sigma_l^2 \boldsymbol{\Psi_l}\left(\boldsymbol{X_l}, \boldsymbol{X_h}\right) \\ \mathbf{0} & \rho^2\sigma_l^2 \boldsymbol{\Psi_l}\left(\boldsymbol{X_h}, \boldsymbol{X_h}\right) + \sigma_d^2 \boldsymbol{\Psi_d} \end{bmatrix} \tag{7.13}$$

and F is the regression basis function matrix.
The correlation matrices $\boldsymbol{\Psi_l}$ and $\boldsymbol{\Psi_l}$ are obtained from

$$\psi(\boldsymbol{x_1}, \boldsymbol{x_2}) = exp\left(-\sum_{i=1}^{d} \theta_i |x_1^i - x_2^i|^p\right) \tag{7.14}$$

where σ is the process variance; θ and p are the hyper-parameters of the correlation function and are obtained estimating the Maximum Likelihood of the function. In 7.11, M is the model matrix of the predicting point x and γ is the matrix of interpolation coefficients.

7.3 Regression Trees

A Regression Decision Tree [13] is a statistical non-parametric supervised technique based on recursive partitioning. Decision Trees are simple to interpret, they can be trained well with large data sets, they don't make assumptions on the type of phenomena analyzed, they are fast in making predictions and they can handle non-smooth problems (like a cut damage models); on the other hand, they can grow to an over complex structure and they have difficulties to represent linear problems. In a decision tree, the training data, also called parent node, is split between two child nodes using the boolean response to a threshold k (like $is\ X > k$?), as shown in figure 7.2, until obtaining the values τ, called leaves, which are the output of the function.

To build a Regression tree, is applied the approach top to down, starting with grouping all the training data into the same partition. The algorithm then begins allocating the data into every possible threshold in the variables range. The algorithm selects the split that minimizes the sum of the squared deviations R

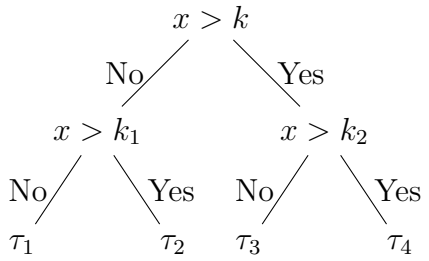

Figure 7.2: Regression Tree example

from the mean in the two separate partitions, R_1 and R_2, calculated as

$$R = \sum_{i=1}^{n_n} (y_i - \overline{y})^2 \tag{7.15}$$

where n_n is the number of observations from the training data which fall in the evaluated branch and $\overline{y}$ is the mean value of the observations in the branch. This process continues until each node reaches a specified minimum node size and becomes a leaf or the sum of squared deviations is zero.

7.4 Self Organizing Maps

A Self Organizing Map (SOM), or Kohonen Map [14], is a non-supervised non-parametric clustering (set of techniques used to group elements with similar features and separate them from other elements) technique used to group a higher dimensional data set in a low-dimensional representation preserving the topological structure of the data.
SOMs are neural networks [45], they try to replicate the structure and function of human brains using a set of nodes, connected to each other, called artificial neurons. Each neuron, like the synapses of a brain, can transmit a signal to the other connected neurons. The output signal of a neuron is obtained using its inputs processed by some function. Neurons have a weight that adjusts as learning proceeds which increases or decreases the strength of the output signal(figure 7.3).

SOMs are trained using competitive learning, in which the neurons compete amongst themselves to be activated, with the result that only one is activated at

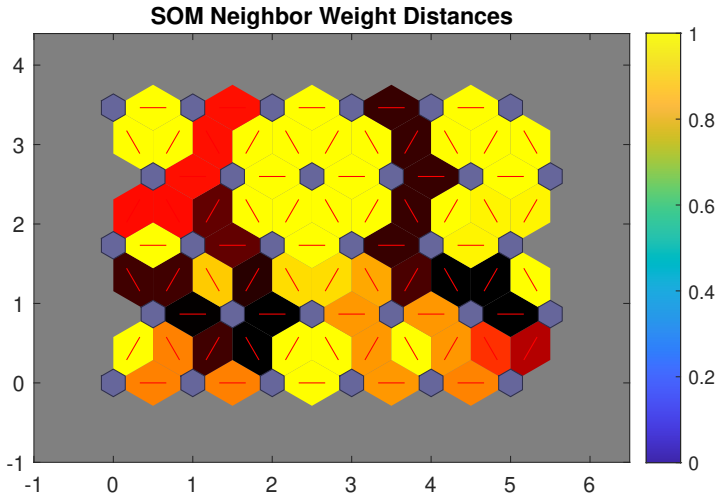

Figure 7.3: Example of weight distances for a trained SOM

any one time. This activated neuron is called best matching unit (BMU).
All the SOMs have four common steps [46]:

1. **initialization**, the weights are initialized with small random values.
2. **competition**, for each training input, the neurons compute the value of the discriminant function, which assume the highest or the lowest value (depending of the function) in the neuron most similar to the input vector. This neuron is called BMU.
3. **cooperation**, the BMU determines the spatial location of a topological neighbourhood of excited neurons, which can cooperate.
4. **adaptation**, the excited neurons alter their discrimination function value through the tuning of the weights in relation to the input vector.

In the training phase, the input matrices $\boldsymbol{T} = [t_1, ..., t_{N_s}]$ are created as

$$\boldsymbol{T_L} = \begin{bmatrix} \boldsymbol{x_L} \\ \boldsymbol{\alpha_L} \end{bmatrix} \quad \boldsymbol{T_{HM}} = \begin{bmatrix} \boldsymbol{x_H} \\ \boldsymbol{\delta_{HM}} \end{bmatrix} \quad \boldsymbol{T_{ML}} = \begin{bmatrix} \boldsymbol{x_M} \\ \boldsymbol{\delta_{ML}} \end{bmatrix} \tag{7.16}$$

and 30 weight vectors w_j with the dimensions of a column of $\boldsymbol{T}$ (N_c) to fit a grid map 6×5. Then, an input vector is introduced to determine the value of the discriminant function [3] for every neuron j as

$$d_j = \|t_j - w_j\|_\Lambda = \sqrt{(t_j - w_j)^T \, \Lambda \, (t_j - w_j)} \qquad j = 1, ..., D \tag{7.17}$$

where D is the number of neurons, M is the input dimension and t is the input vector. To highlight the first 6 values of t_j, is created the diagonal matrix

$$\boldsymbol{\Lambda} = diag\left(\left[0{,}06 \cdot ones(1{,}6) \quad , \quad \frac{1 - 0{,}36}{N_c - 6} \cdot ones(1, N_c - 6)\right]\right) \tag{7.18}$$

The unit with the minimum d is the winner (BMU) and the input is assigned to it (figure 7.4). Then, the weights of the BMU and neurons close to it in the SOM grid are adjusted towards the input vector. The magnitude of the change decreases with time and with the grid-distance from the BMU.

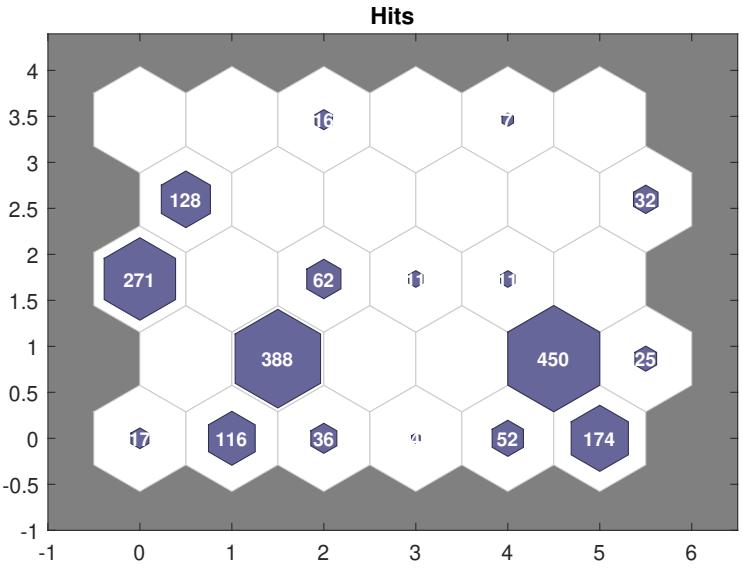

Figure 7.4: Example of Sample Hits for a trained SOM

Chapter 8

Results

The estimated coefficients, obtained from the trained surrogate models of chapter 7 with inputs the validation Damage parameters, are used to calculate the estimated snapshots

$$\boldsymbol{S} = \boldsymbol{\Phi_l}\left(\boldsymbol{\alpha_l} + \boldsymbol{\delta_{ml}} + \boldsymbol{\delta_{hm}}\right) + mean_l \tag{8.1}$$

where $\boldsymbol{\Phi_l}$ is the low fidelity POD basis and $mean_l$ is the mean value of the low fidelity training database. The estimated snapshots are then confronted with the validation set, of which example is presented in figure 8.1, where are only displayed the third and the fourth layer to have a better presentation of the results. Also the time records are obtained using a chipset intel 7700HQ and 16 GB of RAM.
The evaluation error for the ML methods considered is calculated as

$$NRMSE = \frac{\sqrt{\sum_{i=1}^{n_e}\left(\boldsymbol{S_h} - \boldsymbol{S_{estimate}}\right)^2}}{\sqrt{n_e}\left(\boldsymbol{S_{h_{max}}} - \boldsymbol{S_{h_{min}}}\right)} \tag{8.2}$$

S_h are the strain values for the high fidelity snapshot in the validation set, $S_{estimate}$ are the estimate strain values with the ML algorithm and n_e is the number of elements of each column.

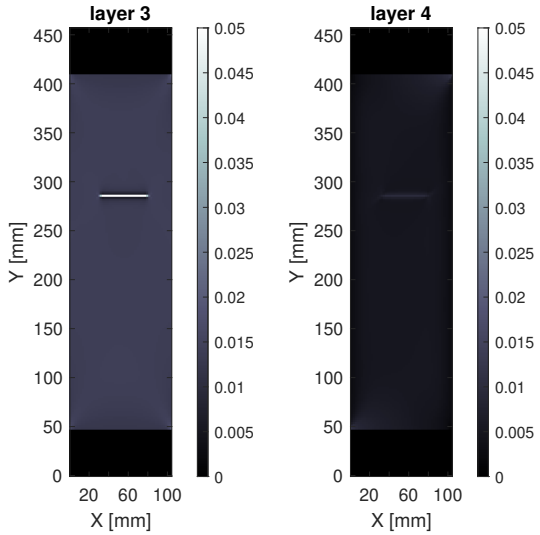

Figure 8.1: High fidelity validation snapshot

8.1 GP Results

Using as inputs $\boldsymbol{\alpha_l}$, $\boldsymbol{\delta_{hm}}$ and $\boldsymbol{\delta_{ml}}$ for the models M_1, M_{2hm} and M_{2ml} and combing the results with the equation 8.1 are predicted the snapshots for the validation data inputs.

In table 8.1 is summarized the training time of the GP for the three models and the time necessary for a new prediction, cumulative of the three parameters. It is possible to observe the training time which is approximately ten times compared to high fidelity FEM analysis, however with an evaluation time of $\sim 0.26\ s$ in just ten iterations in it more time efficient than the high fidelity.

	Time			
	layer 1	layer 2	layer 3	layer 4
Training $\boldsymbol{\alpha_l}$	13980 s	12905 s	13380 s	13015 s
Training $\boldsymbol{\delta_{hm}}$	35 s	31 s	31 s	33 s
Training $\boldsymbol{\delta_{ml}}$	167 s	150 s	140 s	161 s
Evaluation new $\boldsymbol{\alpha}$	0.2594 s	0.2554 s	0.2930 s	0.2455 s

Table 8.1: GP training and evaluation time

The snapshot displayed in figure 8.2 shows on the evaluated snapshot with

the GP. In layer 4 there isn't any cut damage however in layer 3 there is the opposite situation, there is too much background noise and the damage can be barely detected. The NRMSE (using as reference data the validation data set) displayed in figure 8.3, is less than 10% in both cases.

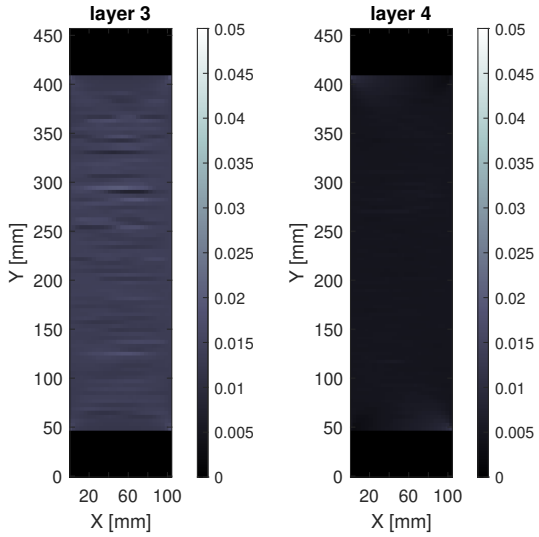

Figure 8.2: High fidelity snapshot obtained with GP

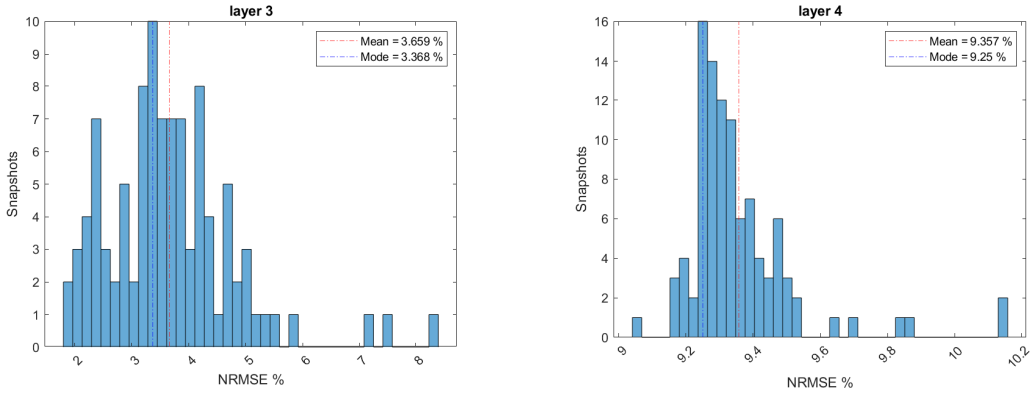

Figure 8.3: NRMSE High fidelity snapshot obtained with GP

8.2 CoKriging Results

Using as inputs $\boldsymbol{\alpha_l}$ and $\boldsymbol{\beta_h}$ for the CoKriging model are predicted the snapshots for the validation data inputs. In table 8.2 is summarized the training time and the time necessary for a new prediction. In this case the training time is not negligible expecially for layer 2. The snapshot displayed in figure 8.4, shows a plate filled with

	Time			
	layer 1	layer 2	layer 3	layer 4
Training	77974 s	426089 s	11811 s	147469 s
Evaluation new $\boldsymbol{\alpha}$	11.43 s	11.15 s	9.14 s	10.23 s

Table 8.2: CoKriging training and evaluation time

cuts due to the excessive noise and the difficulty of determinate a small localized discontinuity using an interpolation model. The NRMSE (using as reference data the validation data set) displayed in figure 8.5 is highest of the four models, with peaks of over the 80% for the layer 4.

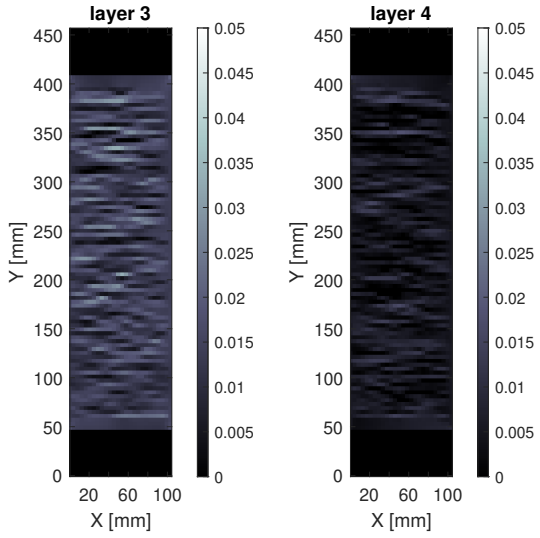

Figure 8.4: High fidelity snapshot obtained with CoKriging

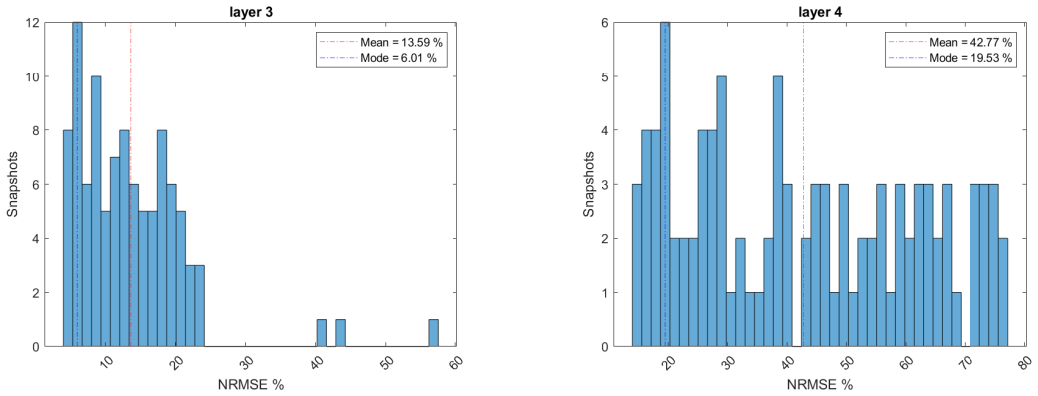

Figure 8.5: NRMSE High fidelity snapshot obtained with CoKriging

This results demonstrate that the CoKriging is not suitable for the reproduction of data with small localized perturbations.

8.3 Regression Trees Results

Using as inputs $\boldsymbol{\alpha_l}$, $\boldsymbol{\delta_{hm}}$ and $\boldsymbol{\delta_{ml}}$ for the models M_1, M_{2hm} and M_{2ml} and combing the results with the equation 8.1 are predicted the snapshots for the validation data inputs. In table 8.3 is summarized the training time of the Regression tree and the time necessary for a new prediction which is extremely lower compared to high fidelity FEM analysis and the other methods in this comparison.

	Time			
	layer 1	layer 2	layer 3	layer 4
Training $\boldsymbol{\alpha_l}$	5.98 s	6.31 s	6.57 s	5.38 s
Training $\boldsymbol{\delta_{hm}}$	2.10 s	1.97 s	1.94 s	1.91 s
Training $\boldsymbol{\delta_{ml}}$	2.30 s	2.10 s	2.19 s	2.15 s
Evaluation new $\boldsymbol{\alpha}$	0.34 s	0.34 s	0.34 s	0.29 s

Table 8.3: Regression Trees training and evaluation time

The snapshot displayed in figure 8.6, shows a light evidence of the cut in layer 4, while in layer 3 there is to much noise. Compared to the GP, the NRMSE (using as reference data the validation data set) displayed in figure 8.7 is greater but this model can localize the damage in the layer 4.

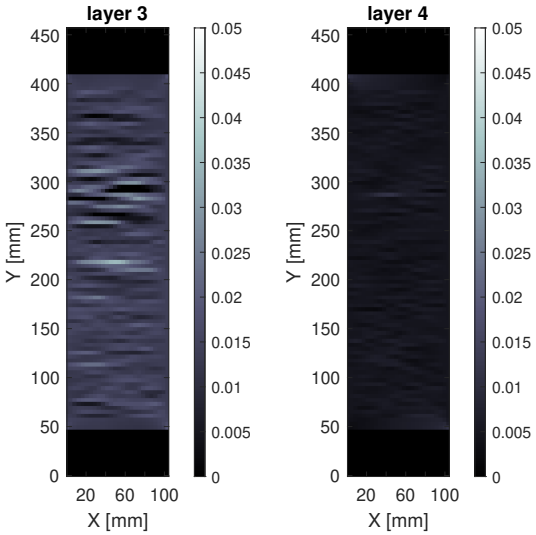

Figure 8.6: High fidelity snapshot obtained with Regression Trees

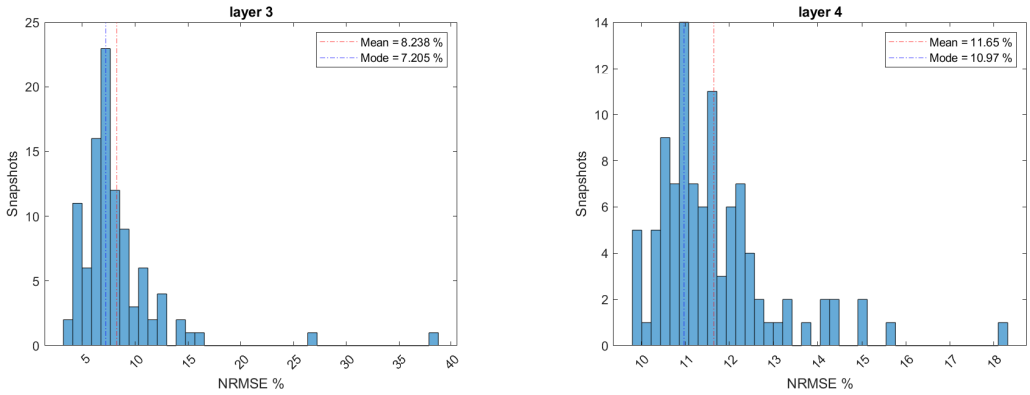

Figure 8.7: NRMSE High fidelity snapshot obtained with Regression Trees

8.4 SOMs Results

Using as inputs $\boldsymbol{\alpha_l}$, $\boldsymbol{\delta_{hm}}$ and $\boldsymbol{\delta_{ml}}$ for the models M_1, M_{2hm} and M_{2ml} and combing the results with the equation 8.1 are predicted the snapshots for the validation data inputs. In table 8.4 is summarized the training time of the SOM and the time necessary for a new prediction which is extremely lower compared to high fidelity FEM analysis. The high training time it is a consequence of the 1000 training epochs.

	Time			
	layer 1	layer 2	layer 3	layer 4
Training α_l	20193 s	2074 s	2147 s	2903 s
Training δ_{hm}	46 s	46 s	80 s	54 s
Training δ_{ml}	160 s	160 s	235 s	184 s
Evaluation new α	0.0153 s	0.0163 s	0.0354 s	0.0174 s

Table 8.4: SOM training and evaluation time

The snapshot displayed in figure 8.8, doesn't show a evidence of the cut in the layers. In fact, it is impossible to determine the cut damage in the layers, even thought the NRMSE (using as reference data the validation data set) displayed in figure 8.9 is low. The low error is a consequence of the snapshots strains being nearly zero.

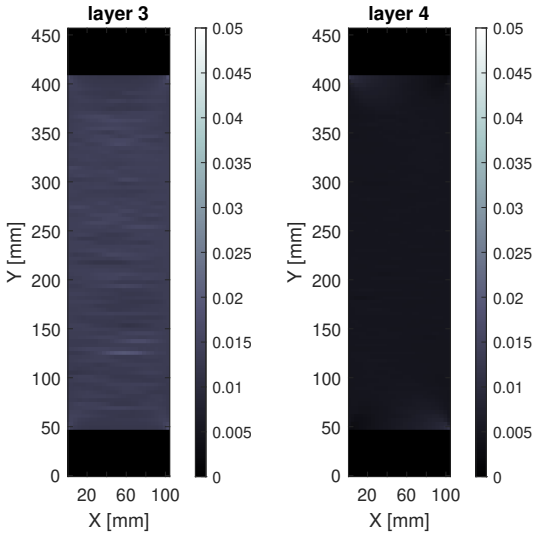

Figure 8.8: High fidelity snapshot obtained with SOMs

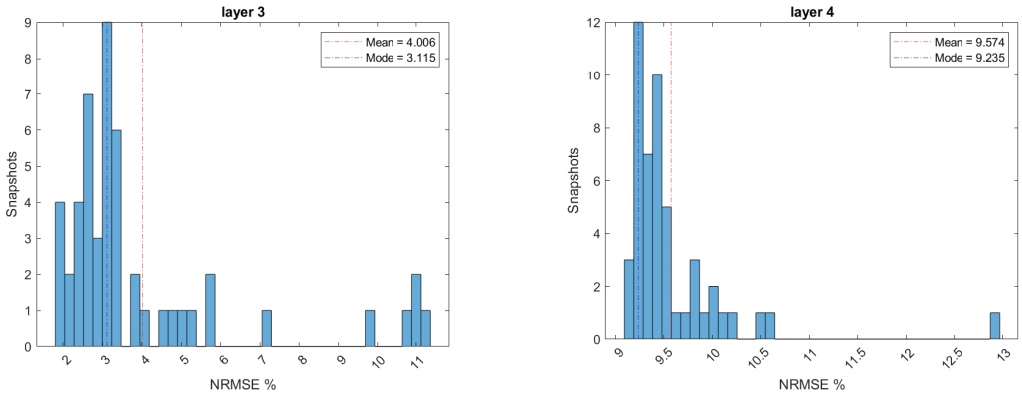

Figure 8.9: NRMSE High fidelity snapshot obtained with SOMs

8.5 Mixed method

The ML models from chapter 7 are unsatisfactory due to their the poor capacity to reproduce a localized damage so it is proposed a method to improve the results.

Starting for the table 8.5, the best performances in terms of training time and

ML method	NRMSE	Training Time	Evaluation Time
GP	3.659 %	13551 s	0.2930 s
CoKriging	13.59 %	11811 s	8.14 s
Regression Tree	8.238 %	10.7 s	0.34 s
SOM	4.006 %	2462 s	0.0174 s

Table 8.5: Mean results comparison

localization of the cut are represented by the Regression Trees while the best evaluation time is represented by the SOMs. So, it was thought to combine the two methods to achieve better overhaul results and improve the quality of the snapshots obtained through the surrogate model.

Firstly the Regression Trees are trained for the model M_1 using as inputs $\boldsymbol{\alpha_l}$ in order to reduce the training time, then two SOMs are trained for the modelsM_{2hm} and M_{2ml} using as inputs, respectively, $\boldsymbol{\delta_{hm}}$ and $\boldsymbol{\delta_{ml}}$. Afterwards to predict the new snapshots, the results are combined using the equation 8.1.

In table 8.6 is summarized the training time and the time necessary for a new

prediction. The training time for each model is comparable with the previous result however the evaluation time for a new snapshot is reduced tenfold compared to Regression tree.

The major improvement of the method is highlighted in the snapshot displayed in

	Time			
	layer 1	layer 2	layer 3	layer 4
Training α_l	13 s	7.7 s	9.6 s	8.1 s
Training δ_{hm}	72.5 s	64.5 s	78.4 s	89 s
Training δ_{ml}	228.7 s	207.5 s	235.6 s	214 s
Evaluation new α	0.0229 s	0.0218 s	0.0257 s	0.0236 s

Table 8.6: Regression Tree and SOMs training and evaluation time

figure 8.10. The damage is clearly visible in layer 4 and layer 3, even if the latter maintains a false detection in the lower right of the plate. Also, in layer 3, judging from the intensity of the colour, the results are underestimated.

Considering the NRMSE (using as reference data the validation data set) displayed

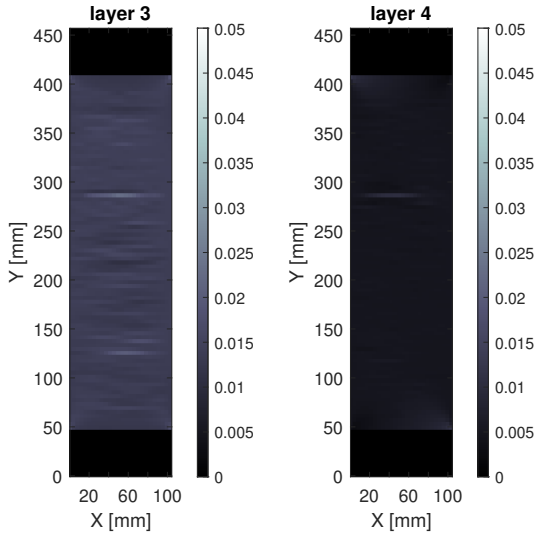

Figure 8.10: High fidelity snapshot obtained with Regression Trees and SOMs

in figure 8.11, the error has decreased to a value comparable to the SOMs with snapshots that can effectively represent the damage.

Comparing the results obtained with the combination of the two methods to a

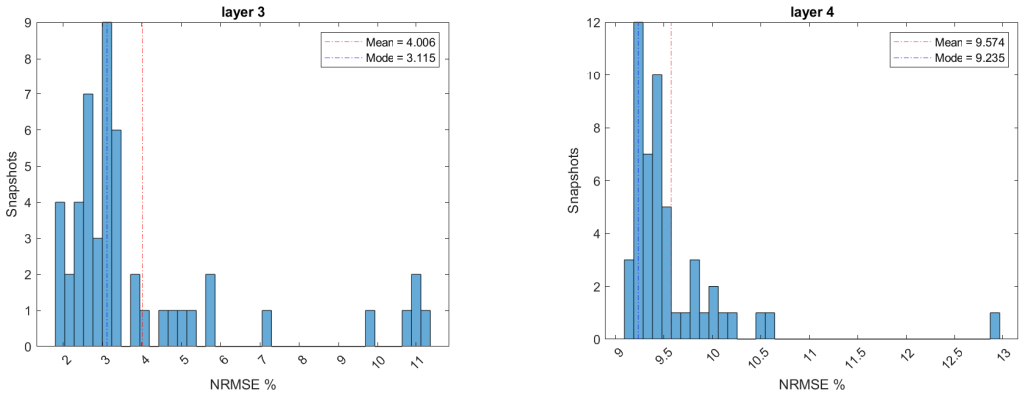

Figure 8.11: NRMSE High fidelity snapshot obtained with Regression Trees and SOMs

snapshot created only with a Regression tree trained with the high fidelity database, the multi-fidelity method improves the localization of the damage, decreasing the false positive occurrences.

Chapter 9

Conclusions

In this thesis is proposed an original surrogate modelling strategy for the determination of the strain field in a cut damaged structure. This method make use of different sources of information consisting of three finite element model simulations with different approximations of the real world case of study, with the objective of improving the final estimates via transfer learning. The approach to develop the surrogate combines POD , Procrustes Analysis and a combination of Regression Trees and SOMs.

The thesis work started with the creation of three physic's based models, called high fidelity, medium fidelity and low fidelity, from a common specimen with dimensions of $104 \times 456\ mm$ made of four carbon fiber plain-weave prepreg plies stacked with a lamination sequence of $[45°/0°/0°/45°]$. The different models vary from each other due to the different discretization of finite elements.

Afterwards are created the training databases constituted of the strain fields of 100 high fidelity simulations, 300 medium fidelity simulations and 1800 low fidelity simulations. Afterwards, it is calculated the POD to determine the truncated POD basis and POD modal coefficients in order to retain a 99% of the original cumulative energy. The ROM introduced a truncation mean NRMSE in the third layer of $\sim 0.05\%$ compared to the simulations.

Then, implementing a Procrustes analysis algorithm, we determined the transformation matrices to project the the high and medium fidelity POD modal coefficients into the low fidelity basis. This step introduced a mean NRMSE in the third layer

of $\sim 5\%$ compared to the simulations.
Finally, we trained three surrogate models, one for the low fidelity POD modal coefficients estimation based on Regression Trees and two the correction coefficients based on SOMs. The results are confronted to a high fidelity validation set of 100 simulations. We obtained a mean NRMSE in the third layer of $\sim 4\%$ and we were able to correctly estimate the damage in terms of position and strain values.
This strategy can be adapted to fit different structural problems varying the training database and fine tuning the data fit algorithm of the surrogate model.

9.1 Future Developments

In this section are presented possible future developments of the thesis, which could be the improvement of the database using experimental data.
The test process will involve placing the test specimen in a universal testing machine and slowly extending it to a displacement value of 5 mm (in order to remain consistent with the simulation data), the values of the strain field are then collected using several strain gauges mounted on the fourth layer of the specimen. The experimental data will be used to validate the physic's based models and to improve the surrogate model training process.
Another possible future development is the development of a real-time structure integrity assessment configuration for non exposed damage detection using data obtained from aircraft sensors [3] by linking the surrogate models of each layer.

Bibliography

[1] Diogo Montalvao, N. Maia, and A. Ribeiro. «A Review of Vibration-based Structural Health Monitoring with Special Emphasis on Composite Materials». In: *The Shock and Vibration Digest* 38 (July 2006) (cit. on p. 1).

[2] Peter Cawley and R. Adams. «The Location of Defects in Structures From Measurements of Natural Frequencies». In: *Journal of Strain Analysis for Engineering Design* 14 (Apr. 1979) (cit. on p. 1).

[3] Laura Mainini and Karen Willcox. «Data to decisions: Real-time structural assessment from sparse measurements affected by uncertainty». In: *Computers Structures* 182 (Apr. 2017), pp. 296–312 (cit. on pp. 1, 16, 49, 63).

[4] Masoud Sanayei and Michael Saletnik. «Parameter Estimation of Structures from Static Strain Measurements. I: Formulation». In: *Journal of Structural Engineering* 122 (May 1996) (cit. on p. 1).

[5] Alfio Quarteroni and Gianluigi Rozza. *Reduced Order Methods for Modeling and Computational Reduction*. Jan. 2014 (cit. on p. 1).

[6] Alexander I. J. Forrester, Andras Sobester, and Andy J. Keane. *Engineering Design via Surrogate Modelling - A Practical Guide*. Wiley, 2008, pp. I–XVIII, 1–210 (cit. on pp. 1, 18).

[7] X. Fang, H. Luo, and J. Tang. «Structural damage detection using neural network with learning rate improvement». In: *Computers Structures* 83.25 (2005), pp. 2150–2161 (cit. on p. 1).

[8] Alexander Forrester, Andras Sobester, and Andy Keane. «Multi-fidelity optimization via surrogate modelling». In: *Proc. R. Soc. A* 463 (2007), pp. 3251–3269 (cit. on pp. 2, 16, 45).

[9] Karl Pearson F.R.S. «LIII. On lines and planes of closest fit to systems of points in space». In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572 (cit. on pp. 2, 14, 27).

[10] Christian Perron, Dushhyanth Rajaram, and Dimitri Mavris. «Development of a Multi-Fidelity Reduced-Order Model Based on Manifold Alignment». In: (June 2020) (cit. on pp. 2, 15, 35).

[11] Amy Ross. «Procrustes Analysis». In: () (cit. on pp. 2, 36).

[12] Mark Ebden. «Gaussian Processes: A Quick Introduction». In: (2015) (cit. on pp. 2, 43).

[13] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, 1984 (cit. on pp. 2, 16, 46).

[14] T. Kohonen. «The self-organizing map». In: 78 (1990) (cit. on pp. 2, 47).

[15] *Excel 8552 AS4 data sheet*. Wichita State University, USA: Nation Center for Advanced Materials Performance (cit. on p. 4).

[16] R.H. MacNeal, United States. National Aeronautics, Space Administration. Scientific, and Technical Information Office. *The NASTRAN Theoretical Manual*. NASA SP v. 1. Scientific, Technical Information Office, National Aeronautics, and Space Administration, 1970 (cit. on pp. 6, 9).

[17] Siemens. *Element library reference* (cit. on pp. 6, 9).

[18] Sreelatha Kilambi and Steven Tipton. «Numerical evaluation of the original "Neuber's Rule" for pure out-of-plane shear loading». In: *The Journal of Strain Analysis for Engineering Design* (Aug. 2013) (cit. on p. 6).

[19] Inc. Autodesk. URL: `https://knowledge.autodesk.com/search-result/caas/CloudHelp/cloudhelp/2019/ENU/NSTRN-Reference/files/GUID-1E3BBBCF-AC88-48FA-BC8B-0168C497B31F-htm.html` (cit. on p. 9).

[20] M D McKay, R J Beckman, and W J Conover. «Comparison the three methods for selecting values of input variable in the analysis of output from a computer code». In: (May 1979) (cit. on pp. 13, 20).

[21] G. H. Golub and C. Reinsch. «Singular Value Decomposition and Least Squares Solutions». In: *Numer. Math.* 14.5 (Apr. 1970), pp. 403–420 (cit. on pp. 14, 29).

[22] Nadine Aubry, Philip Holmes, John L Lumley, and Emily Stone. «The dynamics of coherent structures in the wall region of a turbulent boundary layer». In: *Journal of fluid Mechanics* 192 (1988), pp. 115–173 (cit. on p. 14).

[23] Gaëtan Kerschen, J.-C Golinval, ALEXANDER VAKAKIS, and LAWRENCE BERGMAN. «The Method of Proper Orthogonal Decomposition for Dynamical Characterization and Order Reduction of Mechanical Systems: An Overview». In: *Nonlinear Dynamics* 41 (Aug. 2005), pp. 147–169 (cit. on p. 14).

[24] Ryszard Bialecki, Alain Kassab, and Adam Fic. «Proper orthogonal decomposition and modal analysis for acceleration of transient FEM thermal analysis». In: *International Journal for Numerical Methods in Engineering* 62 (Feb. 2005), pp. 774–797 (cit. on p. 14).

[25] John C Gower and Garmt B Dijksterhuis. *Procrustes problems*. Vol. 30. OUP Oxford, 2004 (cit. on pp. 15, 36).

[26] Ndivhuwo Makondo, Benjamin Rosman, and Osamu Hasegawa. «Knowledge transfer for learning robot models via Local Procrustes Analysis». In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)* (2015), pp. 1075–1082 (cit. on pp. 15, 35).

[27] Devis Tuia, Michele Volpi, Maxime Trolliet, and Gustau Camps-Valls. «Semisupervised Manifold Alignment of Multimodal Remote Sensing Images». In: *Geoscience and Remote Sensing, IEEE Transactions on* 52 (Dec. 2014), pp. 7708–7720 (cit. on p. 15).

[28] Chang Wang and Sridhar Mahadevan. «Manifold alignment using Procrustes analysis». In: (2008) (cit. on pp. 15, 35).

[29] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005 (cit. on pp. 16, 43).

[30] Rob Carnell. *Basic Latin hypercube samples and designs with package lhs*. Feb. 10, 2022. URL: https://cran.r-project.org/web/packages/lhs/vignettes/lhs_basics.html (cit. on p. 20).

[31] Tamara Bechtold, A. Verhoeven, E. MATEN, and T Voss. «Model order reduction: An advanced, efficient and automated computational tool for microsystems». In: (Jan. 2007) (cit. on p. 25).

[32] Pietro La Mantia. *Modellazione surrogata multi-fedeltà per piastre danneggiate in materiale composito*. 2018 (cit. on p. 25).

[33] Roland Freund. «Model Reduction Methods Based on Krylov Subspaces». In: *Acta Numerica* 12 (Mar. 2003). DOI: 10.1017/S0962492902000120 (cit. on p. 26).

[34] J. L. LUMLEY. «The structure of inhomogeneous turbulent flows». In: *Atmospheric Turbulence and Radio Wave Propagation* (1967) (cit. on p. 27).

[35] L. SIROVICH. «Turbulence and the Dynamics of coherent structures Part 1: Coherent Structures». In: *Quarterly of Applied Mathematics* 45.3 (1987), pp. 561–571 (cit. on pp. 27, 28).

[36] Philip Holmes, John L. Lumley, and Gal Berkooz. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge Monographs on Mechanics. Cambridge University Press, 1996 (cit. on p. 29).

[37] Anindya Chatterjee. «An introduction to the proper orthogonal decomposition». In: *Current Science* 78.7 (2000), pp. 808–817 (cit. on p. 29).

[38] Wikipedia contributors. *Manifold — Wikipedia, The Free Encyclopedia*. [Online; accessed 21-February-2022]. 2022 (cit. on p. 35).

[39] Tom M. Mitchell. *Machine Learning*. New York: McGraw-Hill, 1997 (cit. on p. 42).

[40] Christopher Williams and Carl Rasmussen. «Gaussian Processes for Regression». In: 8 (1995) (cit. on p. 43).

[41] Jie Wang. «An Intuitive Tutorial to Gaussian Processes Regression». In: (Sept. 2020) (cit. on p. 43).

[42] Inc. The MathWorks. *Gaussian Process Regression Models Documentation* (cit. on p. 45).

[43] Jerome Sacks, William Welch, Toby Mitchell, and Henry Wynn. «Design and analysis of computer experiments. With comments and a rejoinder by the authors». In: *Statistical Science* 4 (1989) (cit. on p. 45).

[44] Marc Kennedy and A O'Hagan. «Predicting the Output from a Complex Computer Code When Fast Approximations Are Available». In: *Biometrika* 87 (1998) (cit. on p. 45).

[45] Jürgen Schmidhuber. «Deep Learning in Neural Networks: An Overview». In: *Neural Networks* 61 (2015), pp. 85–117 (cit. on p. 47).

[46] J. A. Bullinaria. *Introduction to Neural Networks - Course Material: Lecture 16*. URL: `https://www.cs.bham.ac.uk/~jxb/NN/l16.pdf` (cit. on p. 48).