Università Politecnica delle Marche

Department of Industrial Engineering and Mathematical Sciences

Master of Science in Mechanical-Thermomechanical Engineering

Thesis on:

Sviluppo di una procedura automatizzata per la realizzazione di data-set sperimentali simulati ed aumentati per il rilevamento di danni su di un Mockup.

Development of an automated procedure for building simulated and augmented experimental data sets for damage detection on a beam Mockup.

Supervisors:

Prof. Paolo Castellini

Dott. Ing. Emilio Di Lorenzo

Dott. Silvia Vettori

Student:

Simone Colò

ACADEMIC YEAR 2019-2020

TABLE OF CONTENTS

ABSTRACT

Abstract: English version

The goal of this work consist to developing an automated procedure for building simulated and augmented experimental data sets for damage detection on a beam mockup.

Wind turbine infrastructure represents an active area of research in the structural dynamics domain. The project "ReliaBlade": Improving Blade Reliability through Application of Digital Twins over Entire Life Cycle" aims at developing a digital twin of the wind turbine blade covering its entire life-cycle. Within the project, a digital platform has been conceptualised, designed and developed for digital twin applications. The first digital platform use case consists of a desktop size demo mockup. Experimental data acquired during tests on the mockup are streamed online to the IBM Watson IoT Platform cloud and, basing on the transmitted signals, system damages are detected via knowledge driven machine learning applications and numerical models. This work focuses on providing necessary simulated data by running an automated procedure to generate frequency and time data from several mockup Finite Element Model (FEM) damaged configurations. Moreover, it exploits the generated numerical models to enrich experimental data sets through the use of the Augmented Kalman Filter (AKF) for input-state-response prediction.

Abstract: Italian version

Il lavoro di tesi si incentra sullo sviluppo di una procedura automatizzata per creare un set di configurazioni danneggiate di un modello agli elementi finiti. Il fine ultimo consiste poi nell'ottenere le risposte nel tempo e le frequenze proprie del modello, le quali verranno usate come input per algoritmi finalizzati all'identificazione del danno (Damage Detection).

Il progetto è stato sviluppato nell'ambito del consorzio "Reliablade", incentrato sul controllo, manutenzione predittiva e damage detection delle pale eoliche.L'intento è quello di seguire tutta la vita di una pala eolica (Blade) grazie ad una sua riproduzione virtuale, detta Digitalt Twins(Gemello Digitale).

Il lavoro di tesi si colloca all'interno del work-package 1 del progetto, il cui obiettivo risiede nello sviluppo di una piattaforma digitale che permetta la condivisione contemporanea di dati provenienti da test sperimentali e da simulazioni. Tali simulazioni debbono essere eseguite con modelli FEM "danneggiati" per poter permettere a reti neurali di identificare il danno tramite il confronto dei dati simulati con quelli reali. Per questo motivo si è optato per l'utilizzo di un modello semplificato di pala eolica ossia un mockup rappresentativo composto da 4 componenti: un blocco di supporto inferiore rappresentante il suolo, un parallelepipedo a faccia quadrata allungato ed uno schiacciato che servono per simulare la torre di ancoraggio della pala eolica , sostituita con una barra piatta (beam) in acciaio serrata tra i due blocchi precedenti.

Inizialmente il mockup è stato testato in Free-Free con la tecnica del roving accelerometer per ottenere i modi propri di vibrare e permettere poi una validazione del modello 3D FEM. Il mockup è stato poi appoggiato su una scrivania e strumentato con 5 estensimetri e 2 accelerometri ai fini di acquisire le risposte del sistema sottoposto ad

eccitazione del tipo 'pull and release'.

Per ottenere delle reti neurali che funzionino si devono allenare con più campioni possibili,cercando un campione ripetibile e verificabile sperimentalmente.

Dunque per simulare il difetto si è posta una massa magnetica di 8 grammi sulla beam e si sono decise 6 posizioni lungo la sua estensione. È stato poi eseguito un test di tipo 'pull&release' per ogni posizione. Tale test viene comunemente eseguito sulle pale eoliche e consiste nel deflettere la punta della pala, ancorata orizzontalmente, per poi rilasciare ed osservare il fenomeno delle vibrazioni libere (Free Decay). La tipologia di forzante applicata durante questo tipo di test viene definita Gradino o Rampa, secondo la quale l'eccitazione cresce (più o meno velocemente), viene poi mantenuta costante per un periodo di tempo ed infine si azzera instantaneamente.

I modelli virtuali, presenti già nel cloud (fornito da IBM), con i quali la rete neurale sin ora si allena consistono in un modello analitico della barra 1D Euleriano, ed un modello in AMESIM, programma Siemens. L'obiettivo è quello di realizzare un data set proveniente anche dal modello FEM in Simcenter 3D, verificato e provvisto di difetto. Si vuole inoltre estendere i dati forniti non solo ai punti di analisi tramite gli accelerometri e estensimetri, ma a tutti i nodi del Mockup tramite il Kalman Filter che permette di compiere il "Virtual Sensing", ossia simulare sensori in posizioni non strumentate durante i test

Per far ciò, dopo aver realizzato il Mockup in Simcenterd 3D,ed averlo verificato tramite la correlazione con i dati sperimentali, si è passati allo studio del Kalman Filter e dell'Augmented Kalman Filter.

Si è partiti dallo studio teorico per poi implementare l'algoritmo per un sistema a 3 gradi di libertà (gdl) (degrees of freedom) in MATLAB. Ottenuti buoni risultati su esso, si è passati ad importare, le matrici di massa e rigidazza del modello FEM in MATLAB per simulare il sistema. In particolare, la simulazione e' stata implementata su un sistema ridotto al numero di modi di interesse (n°10),cosi da garantire un contenuto tempo computazionale per l'applicazione del Kalman Filter al modello FEM. In questo step l'input, ossia la forzante a gradino, è stato simulato e si sono prese come quantità misurate nel Kalman Filter, le risposte simulate a cui e' stato aggiunto del rumore arbitrario. Cio'è servito per capire se il filtro di Kalman correggesse le risposte simulate con rumore e ne stimasse di meno rumorose. Tutto ciò è stato utile per verificare anche il modello stesso.

Successivamente si è passati a Python con il quale si è automatizzato tutto il processo di generazione dei modelli FE del mockup, con una massa addizionale posta nelle medesime posizioni dei test sul modello fisico, e simulazione di essi tramite il solutore Nastran. Per ottenere il codice Python si sono seguite due strade. La prima realizzando un algoritmo che sfruttasse le librerie dell'interfaccia grafica di NXOpen, e registrando parti di codice direttamente in Simcenter 3D.La seconda, riscrivendo il file .dat del FEM, fornito in output da Simcenter 3D, per aggiungere la massa, e farlo poi risolvere a Nastran.
A causa dell'ingente spazio di archiviazione richiesto dal primo metodo, la scelta e' ricaduta sull'utilizzo della seconda opzione, la quale necessita esclusivamente del salvataggio dei file dat modificati e dei risultati..

Per entrambi si sono svolte le simulazioni sol103 e sol112:

- *sol 103* : risoluzione agli autovalori per l'ottenimento dei modi propri di vibrare e le frequenze corrispondenti.
- *sol 112* : ottenimento della risposta nel tempo in punti specifici(nodi o elementi).

La sol 112 si è risolta sia ponendo direttamente la storia temporale della forzante, generata da python, sia dando le condizioni iniziali di spostamento prese da una soluzione statica con forzante applicata. I file .pch di ogni soluzione sono stati letti e scritti in formato csv tramite Python.

Infine, esportando tutti i risultati in MATLAB tramite i csv, ed inserendo ora anche le misure acquisite sul modello fisico, si e' costruito il data set necessario per implementare l'Augmented Kalman Filter che ha fornito ottimi risultati per i quali si nota una correzione da parte del filtro. Per verificare il funzionamento del filtro, si sono considerate misurate solo le misure di un accelerometro e di 2 estensimetri, mentre le altre sono state usate come confronto con i segnali nel tempo stimati dal filtro di Kalman.

Le risposte stimate in tutti i punti del sistema saranno in grado di fornire un data set completo per gli algoritmi di machine learning atti a determinare la presenza del danno a partire da dati temporali. .

Acknowledgments

I would like to say a big thank you to all the people who have supported me throughout this journey, first and foremost my Parents, my Professor and all my colleagues.

1. INTRODUCTION

The study arises from continuous need to estimate the condition of a wind turbine blade throughout its life-cycle. Since operational measurements are in most cases difficult and time-consuming, there digitalization processes are often put in place in order to reduce the need of testing. Effort are being made to exploit the possibilities of obtaining increasingly high-performance and accurate FEM models (finite element models), combined with experimental measurements and techniques such as the Kalman filter are used to optimise the estimation of results from models. The aim is to obtain virtual models which allow the identification of damage in the real structure. Nowadays, many leading companies use 'Virtual Sensing' and neural networks to optimise the digitalization process.

1.1 Motivation

The work that has been carried out is part of "ReliaBlade" project 1.1, in witch Siemens is involved and working together to DTU (Danmarks Tekniske Universitet).

The main goal of ReliaBlade project is to built a Digital Twin of a physical Wind Turbine blade, that could be update throughout its entire life-cycle and could be used for different activities such as performance prediction and damage detection.

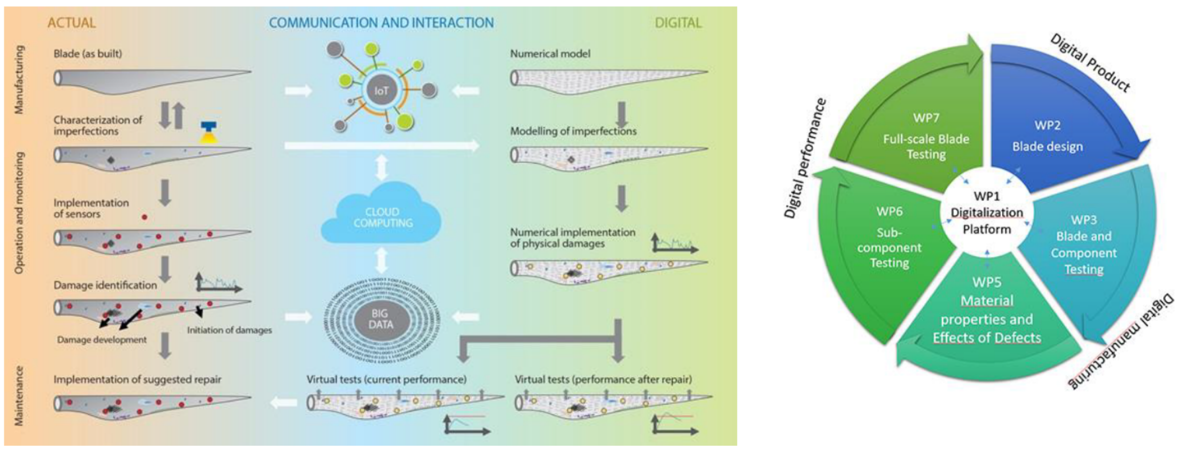

ACTUAL
COMMUNICATION AND INTERACTION
DIGITAL
Manufacturing
Operation and monitoring
Maintenance
Blade (as built)
Characterization of imperfections
Implementation of sensors
Damage identification
Damage development
Initiation of damages
Implementation of suggested repair
IoT
CLOUD COMPUTING
BIG DATA
Numerical model
Modelling of imperfections
Numerical implementation of physical damages
Virtual tests (current performance)
Virtual tests (performance after repair)
Digital Product
Digital performance
Digital manufacturing
WP1 Digitalization Platform
WP2 Blade design
WP3 Blade and Component Testing
WP5 Material properties and Effects of Defects
WP6 Sub-component Testing
WP7 Full-scale Blade Testing

Fig. 1.1: ReliaBlade Project concept map.

In particular, Work-Package 1 (WP1) focuses on building a digitalization platform in which experimental data can be streamed together with simulated data in order to train Machine Learning algorithms with the goal of damage detection.

Computer science defines AI research as the study of "intelligent agents": any device that perceives its environment and takes actions that maximize its chance of successfully achieving its goals.A more elaborate definition characterizes AI as "a system's ability to correctly interpret external data, to learn from such data, and to use those learnings to achieve specific goals and tasks through flexible adaptation." [**1**]
Intelligent agents must be able to set goals and achieve them. They need a way to visualize the future—a representation of the state of the world and be able to make predictions about how their actions will change it—and be able to make choices that maximize the utility (or "value") of available choices. In the specific AI can be classified as in Figure 1.2.

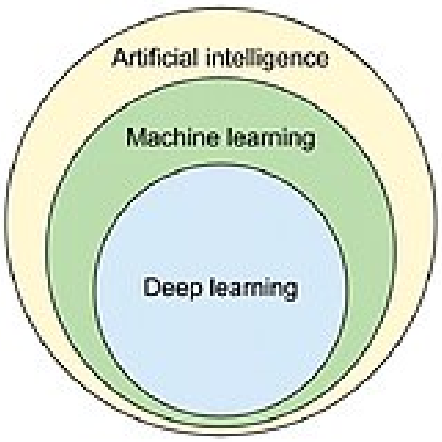


Fig. 1.2: AI classification end sub-classes.

Machine learning is a sub-branch of artificial intelligence. The solution to specific problem lies in the DATA. Algorithms are trained with the dataset, and that results into specific problem. These models are then used for the purpose of predicting the outcomes and solving specific cases/unknowns.
There are another 4 subdivision of Machine Learning algorithms.

Supervised Learning Supervised Learning algorithms are described as learning a target functions (F) that best maps input variables (x) and output variable (Y):

$$Y = F(x)$$

The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output, also known as a supervisory signal. In the mathematical model, each training example is represented by an array or vector, sometimes called a feature vector, and the training data is represented by a matrix. Through iterative optimization of an objective function, supervised learning algorithms learn a function that can be used to predict the output associated with new inputs.An optimal function will allow the algorithm to correctly determine the output for inputs that were not a part of the training data. An algorithm that improves the accuracy of its outputs or predictions over time is said to have learned to perform that task.

After being trained and having reached the set success rate, the network can be used directly on situations detached from the learning cases, without having to employ any more massive computational power.

Unsupervised Learning Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. A central application of unsupervised learning is in the field of density estimation in statistics, such as finding the probability density function. Though unsupervised learning encompasses other domains involving summarizing and explaining data features.

Semi-supervised learning Semi-supervised learning falls between unsupervised learning (without any labeled training data) and supervised learning (with completely labeled training data). Some of the training examples are missing training labels, yet many machine-learning researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy. In weakly supervised learning, the training labels are noisy, limited, or imprecise; however, these labels are often cheaper to obtain, resulting in larger effective training sets.

Reinforcement Learning Reinforcement learning is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. Due to its generality, the field is studied in many other disciplines, such as game theory, control theory, operations research and other. Reinforcement learning algorithms do not assume knowledge of an exact mathematical model of the MDP, and are used when exact models are infeasible. Reinforcement learning algorithms are used in autonomous vehicles or in learning to play a game against a human opponent.

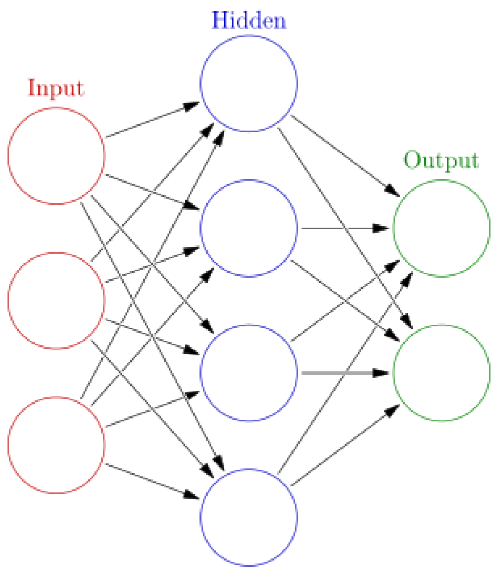


Fig. 1.3: A classic network diagram.

A digital platform has been envisioned within Reliablade WP1 with the purpose of online test data streaming and damage detection.

MOCKUP

To avoid the risk of acting directly on operating wind turbines during the study phase and to simplify the process, the demo Mockup 1.5 has been used instead of a full scale wind turbine blade as the one shown in Figure 1.4.

This has been done to avoid the following disadvantages of the one-to-one scale of the blade:

- Non-portable;
- Difficult to instrument;
- High dimensional model;
- Composite material complexity.

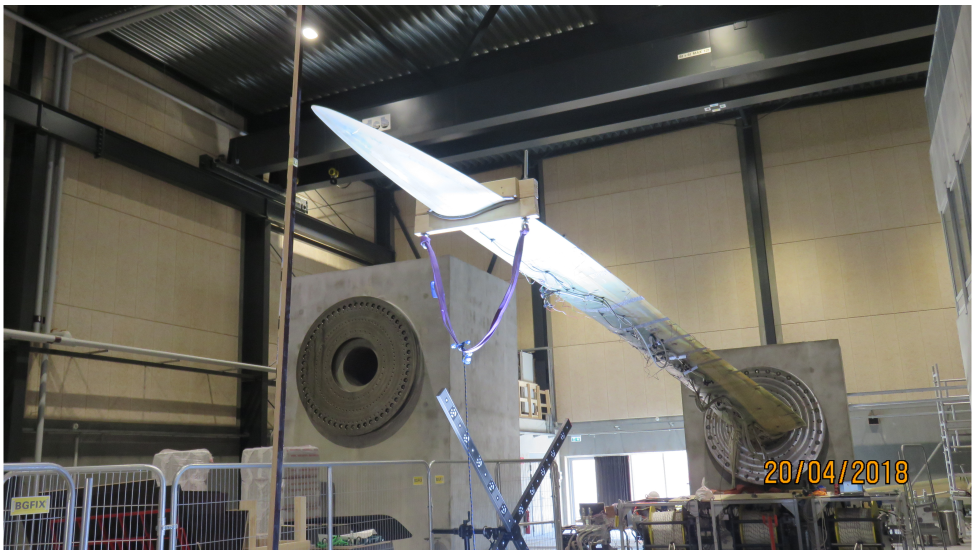
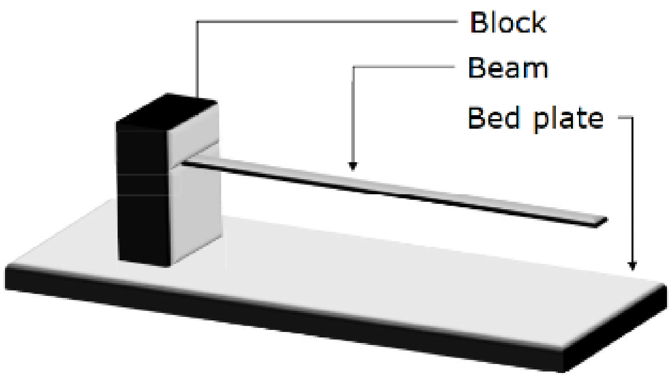


Fig. 1.4: Wind Turbine Blade during a Pull&Release Test, clamped at the root.



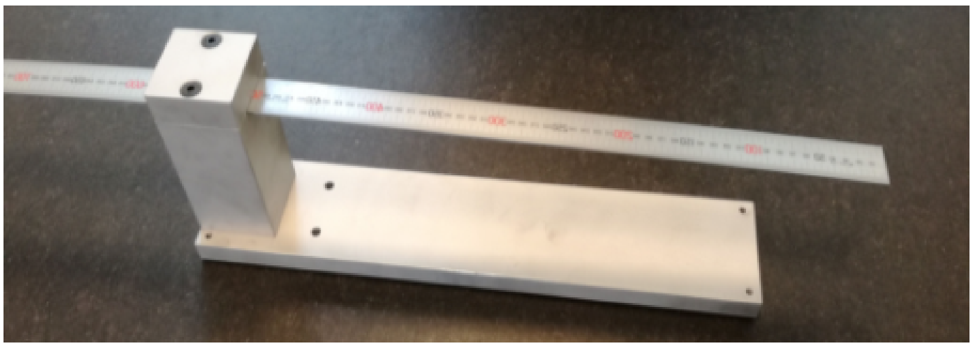


Fig. 1.5: Stylized and physical mockup.

The simple structure of the mockup in Figure 1.5 has the same setup of the blade in a Pull&Release Test. The beam reproduces the blade, which is clamped at the root on the block, while the bed plate is used to simulate the grounding of the block. The materials are:

- Beam : HSS
- Bed plate & Block : AW6082
- Bolted connections : 34CrS4

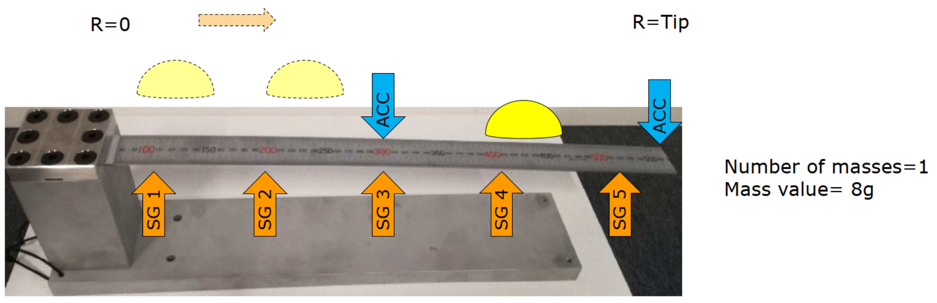


Fig. 1.6: Position of mass and sensors on the mockup.

1.1.1 Pull&Release tests on a beam mockup

Since the goal of the project is to perform damage detection, the mockup has been subjected to a mass damage by adding a 8 grams mass at different positions along the beam length 1.6, while it was tested using a pull and release excitation. This type of excitation has been reproduced by pushing the tip of the beam using the finger 1.11. The mockup was placed resting with rubberised feet on a desk. This would consist of a simply supported constraint but given the presence of the rubberised feet rigid modes in the support plane are prevented and therefore the most appropriate choice for the type of constraint is the Fixed type.

The mokup was instrumented with 5 strain gauges placed on the lower face of the beam, figure1.7, and 2 accelerometers on the upper face, figure 1.8. Regarding accelerometers there is no information while for unidirectional strain gauges there is the data sheet in figure 1.9.

The various locations of the masses for the various tests are listed in the table below:

Mass position	1	2	3	4	5	6
Distance from root (mm)	130	223	330	380	430	480

Fig. 1.7: Strains configuration on the lower face of the beam.

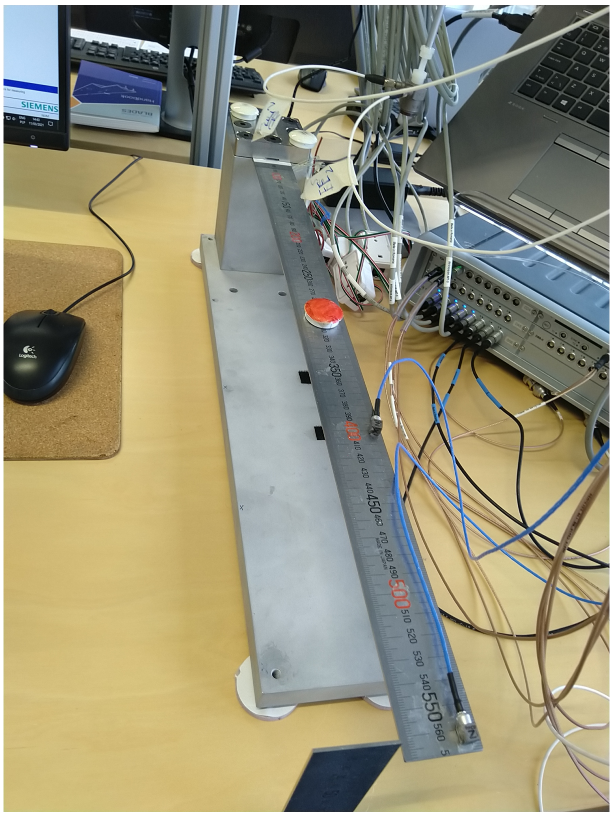
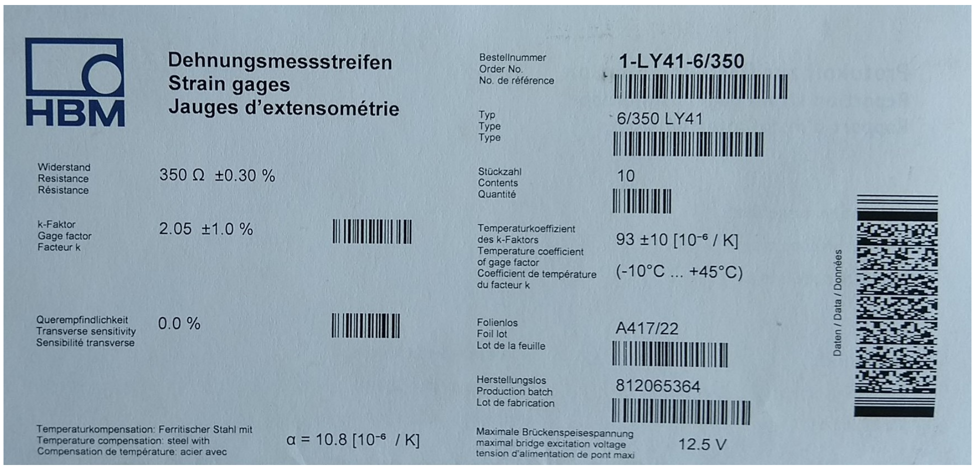


Fig. 1.8: Accelerometers configurations on the upper face of the beam.

Regarding the accelerometers and strain gauges in the table we report the distance from the root by number:

Numbers of sensors	1	2	3	4	5
Accelerometers distance from root [mm]	330	490			
Strains distance from root[mm]	27	127	227	327	417



HBM

Dehnungsmessstreifen
Strain gages
Jauges d'extensométrie

Bestellnummer Order No. No. de référence	1-LY41-6/350
Typ Type Type	6/350 LY41
Stückzahl Contents Quantité	10
Widerstand Resistance Résistance	350 Ω ±0.30 %
k-Faktor Gage factor Facteur k	2.05 ±1.0 %
Temperaturkoeffizient des k-Faktors Temperature coefficient of gage factor Coefficient de température du facteur k	93 ±10 [10^{-6} / K] (-10°C ... +45°C)
Querempfindlichkeit Transverse sensitivity Sensibilité transverse	0.0 %
Folienlos Foil lot Lot de la feuille	A417/22
Herstellungslos Production batch Lot de fabrication	812065364
Temperaturkompensation: Ferritischer Stahl mit Temperature compensation: steel with Compensation de température: acier avec	α = 10.8 [10^{-6} / K]
Maximale Brückenspeisespannung maximal bridge excitation voltage tension d'alimentation de pont maxi	12.5 V

Daten / Data / Données

Fig. 1.9: Strains Data-sheet.

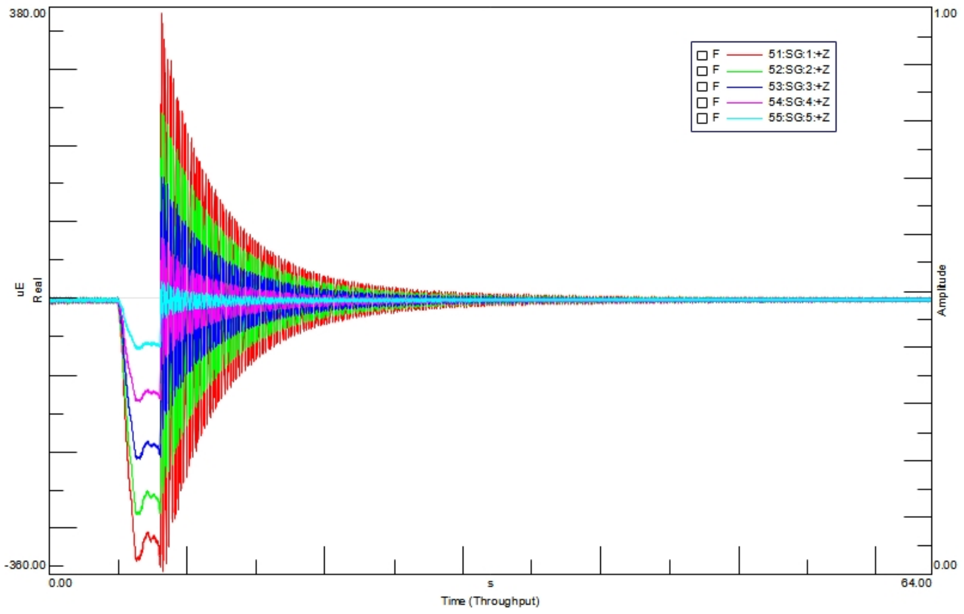


Fig. 1.10: Time-domain responses of strain gauges installed on the mockup.

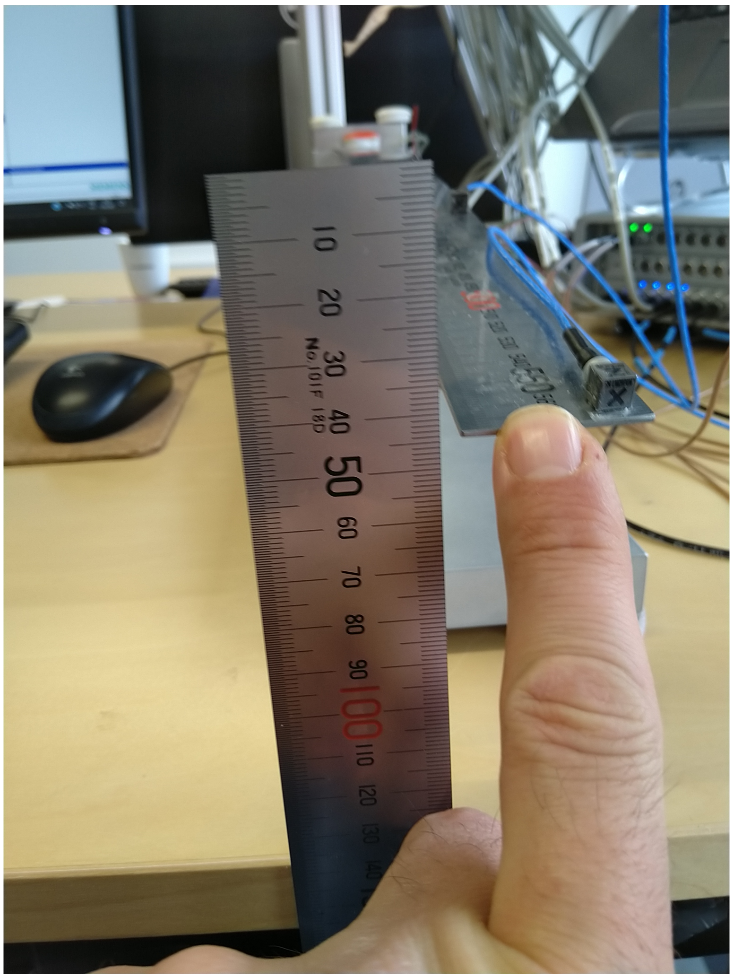


Fig. 1.11: Beam tip deflection during a pull and release test.

1.1.2 Online damage detection

During the test, the experimental data acquired on the mockup are stream online to the cloud during the pull and release test 1.14. The test data have been acquired using the SCADAS system by Siemens. Operational modal analysis is then performed by Polimax Tool in Simcenter TestLab Software. Natural modes and frequencies are therefore computed and stored online. Operational Modal Analysis, Fig. 1.12 can be used online to determine natural frequencies and modes of the damaged physical structure. The resulting data set is compared to simulated ones through machine learning algorithms, which are able to then detect the damage. .

In this case, it was decided to use data from two simulations to train the network:

- 1D Eulerian model;
- 2D Model in AMESIM 1.13 .

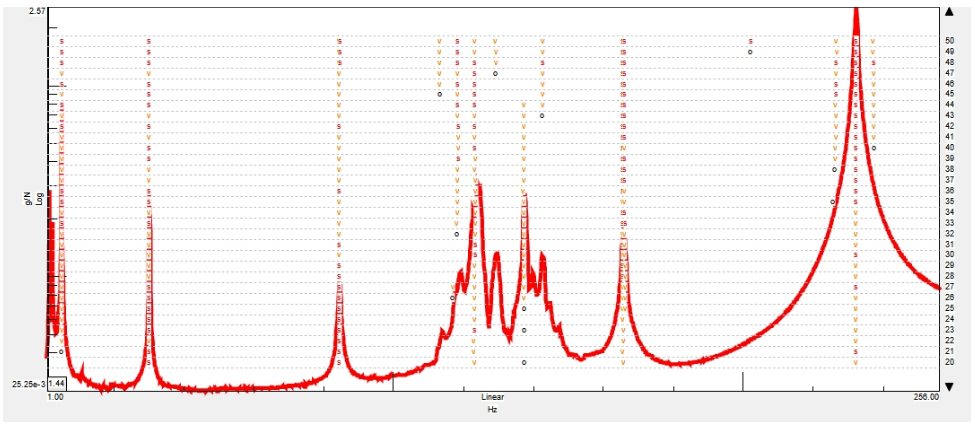


Fig. 1.12: Test mode set computed using PolyMAX tool from TestLab during the Pull&Release test on the mockup.

In Fig. 1.14, a comparison between the experimental frequencies (purple icon) and the corresponding frequencies obtained from a 1D simulation (magenta icon) and the AMESIM

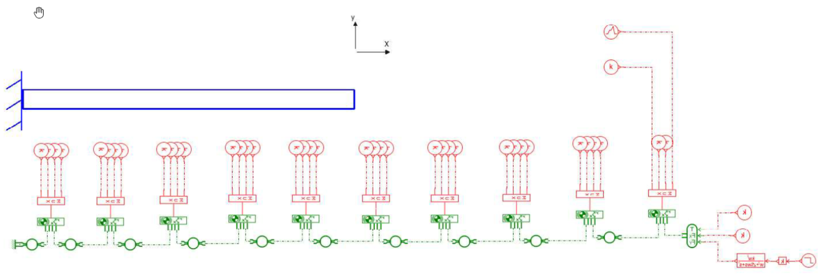


Fig. 1.13: 2D Simulation on AMESIM, Siemens software.

simulation (green icon) is reported. The machine learning algorithm is able to find the correct mass location by comparing the three data sets.

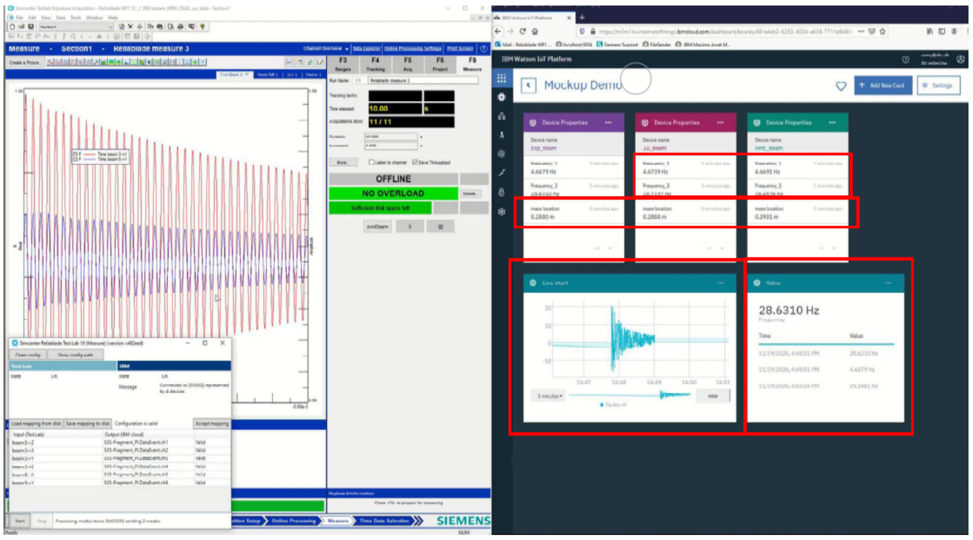
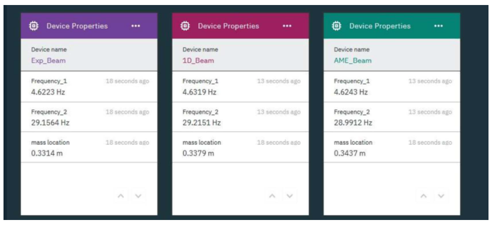


Fig. 1.14: Data streaming and automatic online mass damage localization

1.2 Objective

After a preliminary theoretical study of the Kalman Filter and Augmented K.F. , the work was divided into three parts.

First Objective The objective is to create an automated procedure to simulate a damage presence on the mockup using a 3D model, in particular a finite element model. The idea is to generate a large number of FE models, each one characterized by a different mass location and compute natural modes and frequencies. The resulting data sets can be uploaded on the platform and used to train the machine learning algorithm. In this way, a the fourth block can be added in the graphical interface of the platform 1.16.

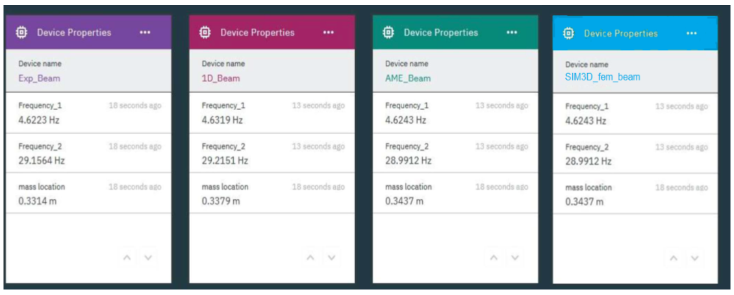


Device Properties
Device name
Exp_Beam
Frequency_1 18 seconds ago
4.6223 Hz
Frequency_2 18 seconds ago
29.1564 Hz
mass location 18 seconds ago
0.3314 m

Device Properties
Device name
1D_Beam
Frequency_1 13 seconds ago
4.6319 Hz
Frequency_2 13 seconds ago
29.2151 Hz
mass location 18 seconds ago
0.3379 m

Device Properties
Device name
AME_Beam
Frequency_1 13 seconds ago
4.6243 Hz
Frequency_2 13 seconds ago
28.9912 Hz
mass location 18 seconds ago
0.3437 m

Fig. 1.15: Comparison between experimental and simulated natural frequencies identified by the machine learning algorithm on the platform.



Device Properties
Device name
Exp_Beam
Frequency_1 18 seconds ago
4.6223 Hz
Frequency_2 18 seconds ago
29.1564 Hz
mass location 18 seconds ago
0.3314 m

Device Properties
Device name
1D_Beam
Frequency_1 13 seconds ago
4.6319 Hz
Frequency_2 13 seconds ago
29.2151 Hz
mass location 18 seconds ago
0.3379 m

Device Properties
Device name
AME_Beam
Frequency_1 13 seconds ago
4.6243 Hz
Frequency_2 13 seconds ago
28.9912 Hz
mass location 18 seconds ago
0.3437 m

Device Properties
Device name
SIM3D_fem_beam
Frequency_1 13 seconds ago
4.6243 Hz
Frequency_2 13 seconds ago
28.9912 Hz
mass location 18 seconds ago
0.3437 m

Fig. 1.16: Additional simulated data set added to the platform in order to train the machine learning algorithm.

Second Objective The objective is to simulate time data 1.17. The interest is on obtaining the transient response for each damaged model configuration. Then data can be stored online and used to train a new type of machine learning algorithm based on time data instead of normal modes and frequencies.

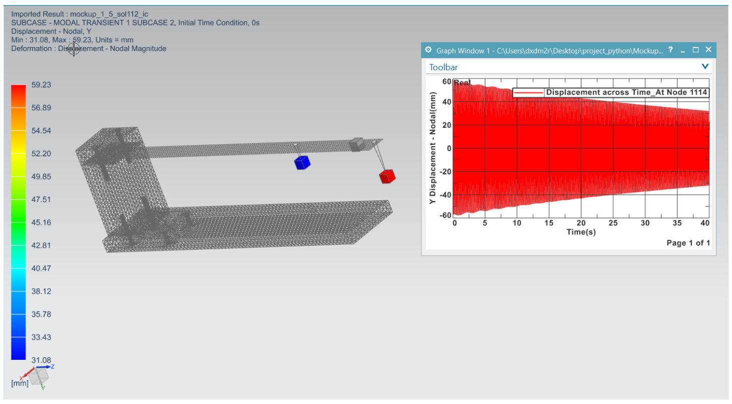

Imported Result : mockup_1_5_sol112_ic
SUBCASE - MODAL TRANSIENT 1 SUBCASE 2, Initial Time Condition, 0s
Displacement - Nodal, Y
Min : 31.08, Max : 59.23, Units = mm
Deformation : Displacement - Nodal Magnitude
59.23
56.89
54.54
52.20
49.85
47.51
45.16
42.81
40.47
38.12
35.78
33.43
31.08
[mm]
Graph Window 1 - C:\Users\dxdm2r\Desktop\project_python\Mockup...
Toolbar
Real
Displacement across Time_At Node 1114
Y Displacement - Nodal(mm)
Time(s)
Page 1 of 1

Fig. 1.17: Simulated time history in Simcenter 3D.

Third Objective The last objective is to use the already available experimental data to obtain the full field response of the mockup using Virtual Sensing. Figure 1.18 shows how an unmeasured response can be estimated using Virtual Sensing methods such as the Augmented Kalman Filter.. In this way the new M.L. algorithm can be train using full field experimental data.

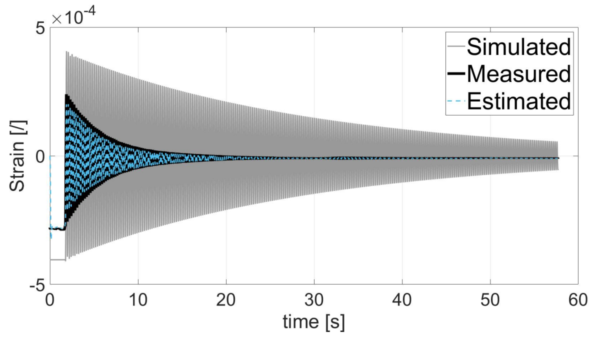


Fig. 1.18: Time data from simulation and estimation by AKF compared to corresponding measured quantity.

2. THEORETICAL BACKGROUND

Filtering is desirable in many situations in engineering and embedded systems. For example, radio communication signals are corrupted with noise. A good filtering algorithm can remove the noise from electromagnetic signals while retaining the useful information. Another example is power supply voltages. Uninterruptible power supplies are devices that filter line voltages in order to smooth out undesirable fluctuations that might otherwise shorten the lifespan of electrical devices such as computers and printers. The Kalman filter is a tool that can estimate the variables of a wide range of processes. In mathematical terms we would say that a Kalman filter estimates the states of a linear system. The Kalman filter not only works well in practice, but it is theoretically attractive because it can be shown that of all possible filters, it is the one that minimizes the variance of the estimation error. Kalman filters are often implemented in embedded control systems because in order to control a process, you first need an accurate estimate of the process variables. The focus of this chapter consists in providing an exhaustive description of the Kalman Filter and Augmented Kalman Filter algorithm used in the project.

The State-Space formulation for a N-Degrees of Freedom (N-DOFs) system, necessary for applying the Kalman Filter, will be at first provided. The Kalman Filter and Augmented Kalman Filter algorithms will be then described. Moreover, results from their application to the N-DOFs system will be reported.

2.1 N-Degrees of Freedom system

2.1.1 State-Space formulation

Many processes in our world can be described by state-space systems. These include processes in engineering, economics, physics, chemistry, biology, and many other areas. If we can derive a mathematical model for a process, then we can use mathematical tool to control the process and obtain related information. If we know the state of a system at the present time, and we know all of the present and future inputs, then we can deduce the values of all future outputs of the system. State-space models can be generally divided into linear models and non-linear models. Although most real processes are non-linear, the mathematical tools that are available for estimation and control are much more accessible and well understood for linear systems. Therefore non-linear systems are often approximated as linear systems.

[2],**[3]** The mathematical formulation for a 1 DOF system is hereby reported and then extended to a N-DOFs system.

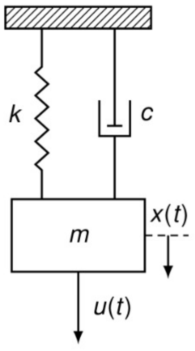


Fig. 2.1: 1 Degree of Freedom mass-spring-damper system.

For a linear time-invariant mechanical system, the equation of motion can be derived as:[4],[5]

$$m\ddot{x}(t) + c\dot{x}(t) + kx(t) = u(t)$$

$$\ddot{x}(t) + 2\zeta_n\omega_n\dot{x}(t) + \omega_n^2 x(t) = \frac{1}{m}u(t)$$

$$\omega_n = \sqrt{\frac{k}{m}} : \ natural\ frequency\ (rad/s)$$

$$\zeta_n = \frac{c}{2\sqrt{km}} : \ damping\ ratio$$

with m : body mass, c : damping, k : stiffness of the system and $u(t)$: input vector. The state-space system can be then derived by using the state vector:

$$\xi = [x\ \dot{x}]^t$$

By using the equation of motion, the state vector derivative can be written as:

$$\dot{\xi}(t) = \begin{Bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{Bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}}_{A} \begin{Bmatrix} x(t) \\ \dot{x}(t) \end{Bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{m} \end{bmatrix}}_{B} u(t).$$

A second equation is added to the system to take into account the measured quantities. Since accelerometers are usually adopted in modal analysis, the output equation is hereby provided by expressing the system acceleration as a function of the state vector and the input:

$$y(t) = \ddot{x}(t) = \underbrace{\begin{bmatrix} -\frac{k}{m} & -\frac{c}{m} \end{bmatrix}}_{H} \begin{Bmatrix} x(t) \\ \dot{x}(t) \end{Bmatrix} + \underbrace{\frac{1}{m}}_{D} u(t). \qquad (1)$$

Using the matrix term given above, the system can be written in a compact form:

$$\begin{cases} \dot{\xi}(t) = \bar{A}_c \xi(t) + \bar{B}_c u(t) \\ y(t) = \bar{H}_c \xi(t) + \bar{D}_c u(t) \end{cases}$$

The concept can be extended for a N-DOFs system in view of the final goal of working with a FE model. Therefore the equations of motion can be written as be: $\bar{M}\ddot{\vec{x}} + \bar{C}\dot{\vec{x}} + \bar{K}\vec{x} = \vec{U}$, where the matrices are :

- mass matrix $\bar{M} = \begin{bmatrix} m_1 & 0 & 0 & \dots & 0 \\ 0 & m_2 & 0 & \dots & 0 \\ 0 & 0 & m_3 & \dots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & m_n \end{bmatrix}$

- stiffness matrix $\bar{K} = \begin{bmatrix} (k_1+k_2) & -k_2 & 0 & \dots & 0 & 0 \\ -k_2 & (k_2+k_3) & -k_3 & \dots & 0 & 0 \\ 0 & -k_3 & (k_3+k_4) & \dots & 0 & 0 \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -k_n & (k_n+k_{n+1}) \end{bmatrix}$

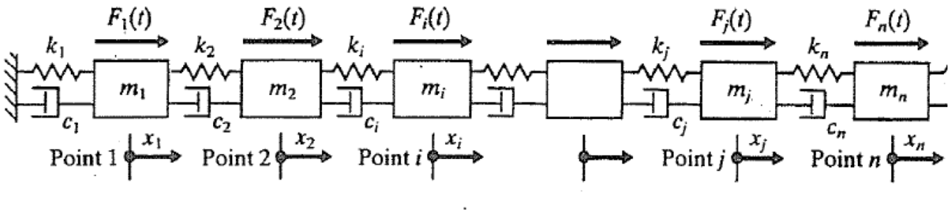


Fig. 2.2: N degree of freedom mass-spring-damper system.

- damping matrix $\bar{C} = \begin{bmatrix} (c_1+c_2) & -c_2 & 0 & \dots & 0 & 0 \\ -c_2 & (c_2+c_3) & -c_3 & \dots & 0 & 0 \\ 0 & -c_3 & (c_3+c_4) & \dots & 0 & 0 \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & -c_n & (c_n+c_{n+1}) \end{bmatrix}$

The state vector can be written as an extension of equation (1) for the N DOFs. At this point it is possible to return to the discussion of the state vector system.

$$\begin{cases} \dot{\xi}(t) = \bar{A}_c \xi(t) + \bar{B}_c u(t) \\ y(t) = \bar{H}_c \xi(t) + \bar{D}_c u(t) \end{cases}$$

In this case the matrices $\bar{A}$, $\bar{B}$, $\bar{H}$ and $\bar{D}$ will be:

$$\bar{A}_c = \begin{bmatrix} [0] & [I] \\ [-[M]^{-1}[K]] & [-[M]^{-1}[C]] \end{bmatrix}^{2n\ x\ 2n}$$

$$\bar{B}_c = \begin{bmatrix} [0] \\ [M]^{-1} \end{bmatrix}^{2n\ x\ ni}$$

$$\bar{H}_c = \begin{bmatrix} [-[M]^{-1}[K]] & [-[M]^{-1}[C]] \end{bmatrix}^{nm\ x\ 2n}$$

$$\bar{D}_c = \begin{bmatrix} [M]^{-1} \end{bmatrix}^{nm\ x\ ni}$$

Where n is the number of bodies, nm is the number of measurements and ni is the number of inputs, for each body.

The system can be formulated in physical coordinates or using a modal reduction basis able to decouple the equations of motions: .

$$\vec{x}(t) = \bar{\Phi}^{n \ x \ m} * \vec{q}(t)$$

In order to be able to observe the system states from the available observations, the observability condition should be satisfied: [**5**].

Observability A system defined by (H,A) matrices is observable if and only if : the rank of the observability matrix O equals the state order (n).

$$\mathcal{O} = \begin{bmatrix} H \\ HA \\ HA^2 \\ \vdots \\ HA^{n-1} \end{bmatrix}$$

2.1.2 Time discretization scheme

The state-space formulation proposed in the previous paragraph describes the system in continuous time. In order for it to be used with physically acquired signal which are discrete, the system needs to undergo a discretization procedure. The following exponential-time discretization scheme as been applied for this purpose, with a sampling period Ts selected according to the measured signals sampling rate. The sampling period for the analytic system under analysis has been set to 0.005. The system matrices can be therefore computed as:

$$A = e^{T_s A_c} = I + T_s A_c + \frac{(T_s A_c)^2}{2!} + \frac{(T_s A_c)^3}{3!} + \frac{(T_s A_c)^4}{4!} + \dots$$

$$B = A_c^{-1}[A - I]B_c$$

$$H = H_c$$

$$D = D_c$$

and the discretized state-space model can be then written as:

$$X_{[k]} + A_1 X_{[k-1]} + \dots + A_n X_{[k-n]} = U_{[k]} + B_1 U_{[k-1]} + \dots + B_m U_{[k-m]}$$

$$\begin{cases} \xi[k+1] = \bar{A}\xi[k] + \bar{B}u[k] \\ y[k] = \bar{H}\xi[k] + \bar{D}u[k] \end{cases}$$

, where k is the time instant.

2.2 Kalman Filter

The state estimation problem aims at calculating $\xi(t)/\xi[k]$ when:

- the matrices $\bar{A}$, $\bar{B}$, $\bar{H}$ and $\bar{D}$ are known;
- measurements of the input $u[k]$ and the output $y[k]$ are available;
- both the state and the output equations are amenable to errors due to modelling, uncertainty, linearization, etc.

Using the Luenberger Observer, let the state equation be approximated by the expression:

$$\hat{\xi}[k+1] = \bar{A}\hat{\xi}[k] + \bar{B}u[k]$$

Then, it is easy to deduce that :

- if $\hat{\xi}[0] = \xi[0]$, the states b $\hat{\xi}[k]$ and $\xi[k]$ will be equal, as both equations are driven by the same input term.
- if $\hat{\xi}[0] \neq \xi[0]$, the states $\hat{\xi}[k]$ and $\xi[k]$ will be equal, as both equations are driven by the same input term, yet, after some time.

To explain this, let's assume an error between the actual value and the estimated value:

$$e[k] = \xi[k] - \hat{\xi}[k]$$

Therefore:

$$e[k+1] = \bar{A}e[k]$$

This leads to three implications:

- The error follows a free response.
- If A is stable e[k + 1] goes to zero: thus after some time the estimated state converges to the true one.
- There is no control of how fast (or slow) the estimated state converges to the true one.

In order to control the error convergence, the estimation of the outputs is also introduced:

$$\hat{y}[k] = \bar{H}\hat{\xi}[k] + \bar{D}u[k]$$

A correction based on the difference between the actual output $y[k]$ and the estimated one can be introduce :

$$\hat{\xi}[k+1] = \bar{A}\hat{\xi}[k] + \bar{B}u[k] + \bar{K}(y[k] - \hat{y}[k])$$

where $\bar{K}$ is a gain matrix.

The error can be therefore expressed as:

$$e[k+1] = (\bar{A} - \bar{K}\bar{H})e[k]$$

One can control the evolution of the state estimation error through an appropriate selection of the gain matrix $\bar{K}$. Thus, the deterministic state estimation problem is reduced at finding whether such a matrix always exists and, if yes, how to calculate it.

The existence of the gain matrix is supported by Kailath's theorem, (1980): "Given matrices $\bar{A}$ and $\bar{H}$, if the pair $(\bar{A}, \bar{H})$ is observable, then there exists a matrix $\bar{K}$ such that $(\bar{A} - \bar{K}\bar{H})$ is asymptotically stable".

For a 1 DOF system, the response shown in Figure 2.3 can be obtained.

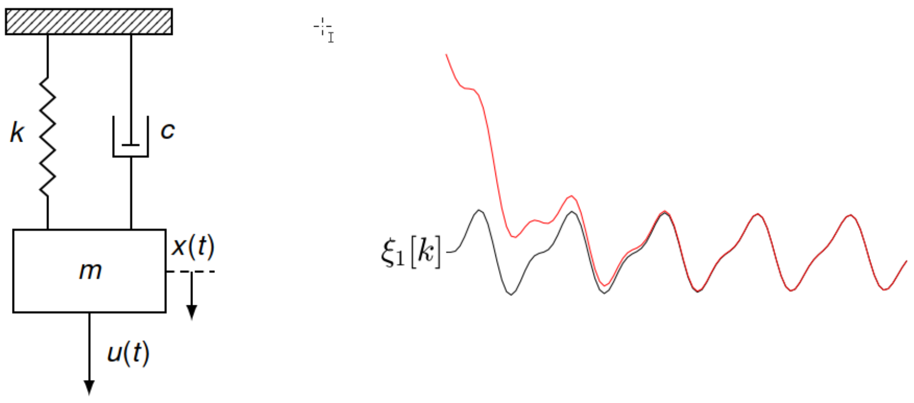


Fig. 2.3: 1 DOF mass-spring-damper system : actual state in black and estimated state in red, with $\hat{\xi}[0] \neq \xi[0]$.

2.2.1 *Algorithm description*

So far, the state system has been analysed in deterministic terms. **[6]**,**[7]**

Real systems cannot be considered deterministic. Kalman's hypothesis consists in taking into account stochastic quantities. Therefore the objective does not rely in estimating the state vector itself but in predicting its statistic moments: mean and covariance.

Uncertainties affecting the model are taken into account by including noise terms on both the model and the measurements. For this reason, rewrite the state-space system is written as:

$$\begin{cases} \xi[k+1] = \bar{A}\xi[k] + \bar{B}u[k] + w[k] \\ y[k] = \bar{H}\xi[k] + \bar{D}u[k] + v[k] \end{cases}$$

where:

- $w[k]$: process noise of covariance matrix $\bar{Q}$;
- $v[k]$: measurement noise of covariance matrix $\bar{R}$.

Both noises are considered to be zero-mean uncorrelated Gaussian noises. Adopting the stochastic formulation, the estimation error can be expressed as:

$$e[k+1] = (\bar{A} - \bar{K}\bar{H})e[k] - w[k] - \bar{K}v[k]$$

In order to estimate the state vector, its mean needs to be investigated:

$$\mu[k] = E\{\xi[k]\} \equiv \hat{\xi}[k]$$

applying the mean value operator to both sides of the state-equation:

$$E\{\xi[k]\} = E\{A\xi[k-1] + Bu[k-1] + w[k-1]\} \Rightarrow$$

$$= E\{A\xi[k-1]\} + E\{Bu[k-1]\} + \underbrace{E\{w[k-1]\}}_{=0\ zero\ mean}\} \quad (A)$$

Thus the estimated quantity is:

$$\hat{\xi}[k] = A\hat{\xi}[k-1] + Bu[k-1]$$

The estimation error covariance matrix can be defined as:

$$\hat{P}[k] = E\{(\xi[k] - \mu[k])(\xi[k] - \mu[k])^T\}$$

by using equation A:

$$\begin{aligned} E\{(\xi[k] - \mu[k])(\dots)^T\} &= E\{(\bar{A}\xi[k-1] + \bar{B}u[k-1] + w[k-1] - \mu[k])(\dots)^T\} \\ &= \dots \\ * \\ &= \bar{A}E\{(\xi[k-1] - \mu[k-1])(\dots)^T\}\bar{A}^T + E\{w[k-1](w[k-1])^T\} \end{aligned} \tag{2.1}$$

Equation 2.1 can be rewritten as:

$$\rightarrow \hat{P}[k] = \bar{A}\hat{P}[k-1]\bar{A}^T + \bar{Q} \quad (2.1)$$

The objective is to calculate the mean $\hat{\xi}[k]$ and the covariance $\hat{P}[k]$ of the state vector as measurements at $k = 1, 2, \dots$ time instants become available.

The following nomenclature will be hereby considered:

- $\hat{\xi}[i|j]$: estimation of the quantity $\hat{\xi}$ at time i, using measurements up to (and including) time j (conditional expectation).

- $\hat{P}[i|j]$: estimation of the quantity $\hat{P}$ at time i, using measurements up to (and including) time j (conditional expectation).

For better understanding an example is reported:

- $\hat{\xi}[k|k-1], \hat{P}[k|k-1] ==$ estimation of $\hat{\xi}[k]$ and $\hat{P}[k]$ using measurements up to (and including) $k-1 \rightarrow$ one-step ahead prediction.

The Kalman Filter (KF) steps used in the iterations for each time instant are reported in the following, by assuming $\bar{A}$, $\bar{B}$, $\bar{H}$, $\bar{D}$, $\bar{Q}$, $\bar{R}$, $\hat{\xi}[0—0]$ and $\hat{P}[0|0]$ known, at time k and introducing the Kalman gain:

- Calculate the Kalman gain: $\hat{K}[k] = \hat{P}[k|k-1]\bar{H}^T(H\hat{P}[k|k-1]\bar{H}^T + \bar{R})^{-1}$

- Update the mean: $\hat{\xi}[k|k] = \hat{\xi}[k|k-1] + \hat{K}[k](y[k] - \bar{H}\hat{\xi}[k|k-1])$

- Update the covariance: $\hat{P}[k|k] = (\bar{I} - \hat{K}\bar{H})\hat{P}[k|k-1]$

- Predict the mean: $\hat{\xi}[k+1|k] = \bar{A}\hat{\xi}[k|k] + \bar{B}u[k]$

- Predict the covariance: $\hat{P}[k+1|k] = \bar{A}\hat{P}[k|k]\bar{A}^T + \bar{Q}$

- Predict the measurements: $\hat{y}[k+1] = \bar{H}\hat{\xi}[k+1|k] + \bar{D}u[k+1]$

2.3 Augmented Kalman Filter

The Augmented Kalman Filter (AKF) is an extension of the KF which allows to estimate the input in addition to the states. The following conditions need to be therefore satisfied :

- the matrices $\bar{A}$, $\bar{B}$, $\bar{H}$and $\bar{D}$ are known;
- measurements of the output $y[k]$ are available;
- $w[k]$ and $v[k]$ are zero-mean, uncorrelated, white noise processes, with covariance matrices $\bar{Q}$ and $\bar{R}$, respectively.

Given u(t) the unknown input, the augmented state vector can be defined as:

$$\tilde{\xi}[k] = [\xi[k]^T, u[k]^T]^T$$

So to take account of input it is need to add another equation usually a random walk model:

$$u[k+1] = u[k] + w_u[k]$$

where:

- $w_u[k]$: zero mean Gaussian process of covariance matrix $\sum_{uu}$
- on $\sum_{uu}$: by tuning covariance matrix $\sum_{uu}$ we can estimate $w_u[k]$

From this, an augmented state-space model can be obtained:

$$\tilde{\xi}[k] = \underbrace{\begin{bmatrix} \bar{A} & \bar{B} \\ \bar{0} & \bar{I} \end{bmatrix}}_{A_a} \tilde{\xi}[k] + \underbrace{\begin{bmatrix} w[k] \\ w_u[k] \end{bmatrix}}_{\tilde{w}}$$

$$\tilde{y}(t) = \underbrace{\begin{bmatrix} \bar{H} & \bar{D} \end{bmatrix}}_{H_a} \tilde{\xi}[k] + v[k]$$

where:

- $\tilde{w}[k]$: augmented process noise of covariance $\bar{Q}_a = diag\{\bar{Q}, \sum_{uu}\}$
- $v[k]$: measurement noise of covariance matrix $\bar{R}$.
- $\bar{A}_a, \bar{H}_a$ *and* $\tilde{w}$: respectively the augmented system matrices and the augmented noise vector.

The implementation steps are analogous to the KF ones, using the augmented state vector instead of the state vector:

- Calculate the Kalman gain: $\hat{K}_a[k] = \hat{P}[k|k-1]\bar{H}_a^T(H_a\hat{P}[k|k-1]\bar{H}_a^T + \bar{R})^{-1}$
- Update the mean: $\tilde{\hat{\xi}}[k|k] = \tilde{\hat{\xi}}[k|k-1] + \hat{K}_a[k](y[k] - \bar{H}_a\tilde{\hat{\xi}}[k|k-1])$
- Update the covariance: $\hat{P}[k|k] = (\bar{I} - \hat{K}_a\bar{H}_a)\hat{P}[k|k-1]$
- Predict the mean: $\tilde{\hat{\xi}}[k+1|k] = \bar{A}_a\tilde{\hat{\xi}}[k|k] + \bar{B}u[k]$
- Predict the covariance: $\hat{P}[k+1|k] = \bar{A}_a\hat{P}[k|k]\bar{A}_a^T + \bar{Q}_a$
- Predict the measurements: $\tilde{\hat{y}}[k+1] = \bar{H}_a\tilde{\hat{\xi}}[k+1|k]$

2.4 Kalman Filter and Augmented Kalman Filter application to a n-Degrees of Freedom system in MATLAB

The results of the implementation in MATLAB of the KF/AKF for a 3 DOFs system are reported in this paragraph.

The system is assumed to be composed of 3 DOFs with equivalent characteristics:

- m = 0.05 kg
- k = 2000 N/m
- c = 0.25 N*s/m

A sinusoidal input with the following characteristics:

- Amplitude = 10 N
- Frequency = 0.5 Hz

Comparing Figures 2.4, 2.5 a full correspondence is found between the system responses simulated using physical and modal coordinates. Figure 2.5 shows the comparison between the actual system responses, i.e., simulated ones, and the ones estimated using the KF.

Since no measurements are available in simulation, they were reconstructed by adding noise to the corresponding simulated quantities. Only one of them was given to the Kalman Filter which was then asked to estimate the others.

The system was simulated for accelerations and displacements and the displacement of the second DOF and the acceleration of the third DOF where taken as observed quantities. . The same procedure was taken for the AKF for which predicted quantities are reported in Figure 2.6. It can be seen that both the KF and the AKF works correctly when a sinusoidal load is applied to in a 3 DOFs system.

Figures 2.7 and 2.8 show the input prediction achieved using the AKF. Even if the estimation is affected by measurements noise, which was set quite high on purpose, it is still able to capture the sinusoidal behavior of the actual input.

The input-response prediction has been iterated for several values of $\bar{Q}$ and $\bar{Q}_a$ to study their influence on the prediction accuracy. An initial value of $w = 10^{-10}$ has been set and varied 15 times by multiplying it by 10, while the input has set $w_u = 1000$.

Root Mean Square Error (RMSE) has been used to quantify the prediction error achieved using the several covariances values and to provide a comparison. It can be defined as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(Predicted_i - Actual_i)^2}{N}}$$

Where N is number of samples. A stiffness error was introduced into the model in order to check how the process noise covariance value could compensate for it.

By giving more confidence to the model, i.e., by reducing the process noise covariance, the RMSE tends to 0 because the estimated quantity converges to the simulated quantity. This result is shown in Figure 2.9.

The introduced stiffness error consist in a K = 1000 N/m additional stiffness Figure 2.10, shows that if the modified model is adopted for the KF, the prediction error will decrease as far as the process noise covariance increases.

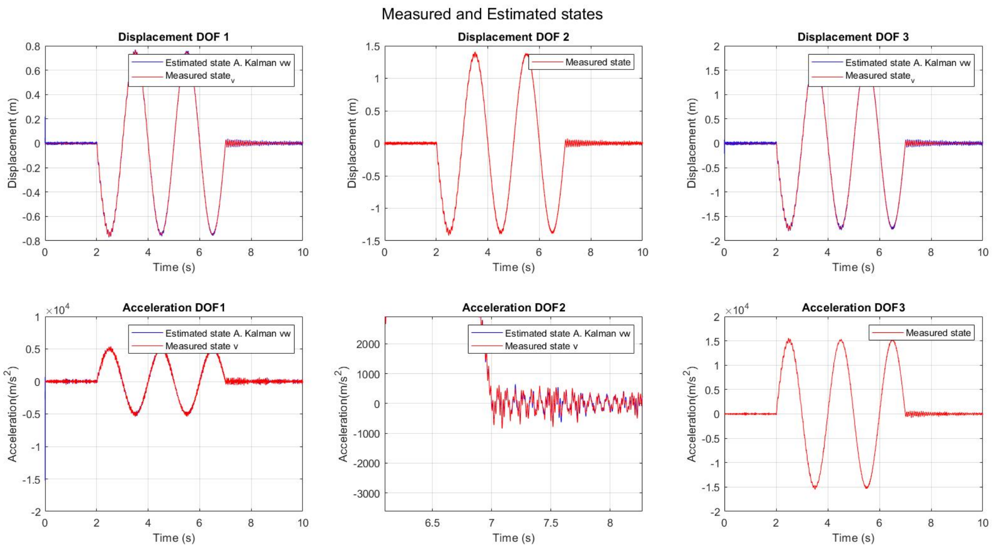


Fig. 2.5: Kalman Filter predictions for a 3 DOFs system solved in in physical coordinates.

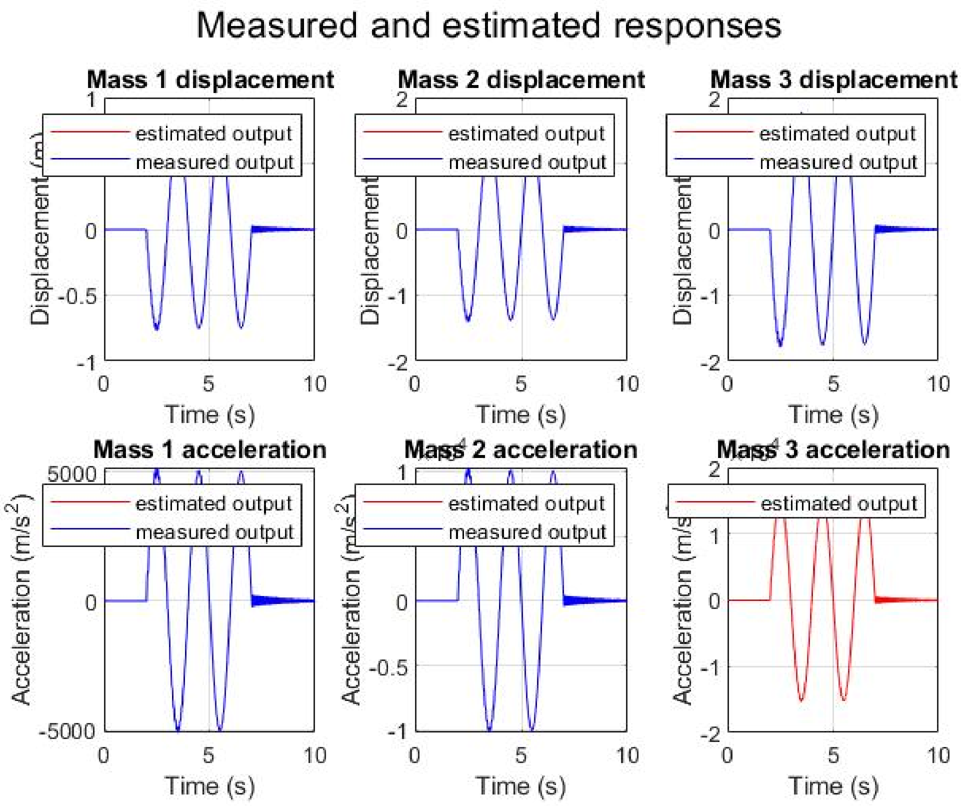


Fig. 2.4: Kalman Filter predictions for a 3 DOFs system solved in modal coordinates.

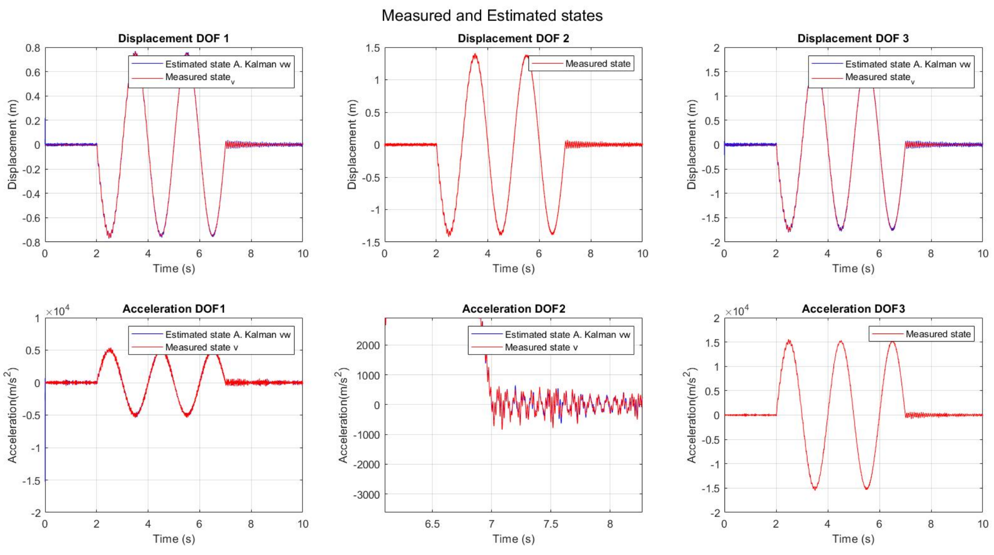


Fig. 2.6: Augmented Kalman Filter predictions for a 3 DOFs solved in in physical coordinates.

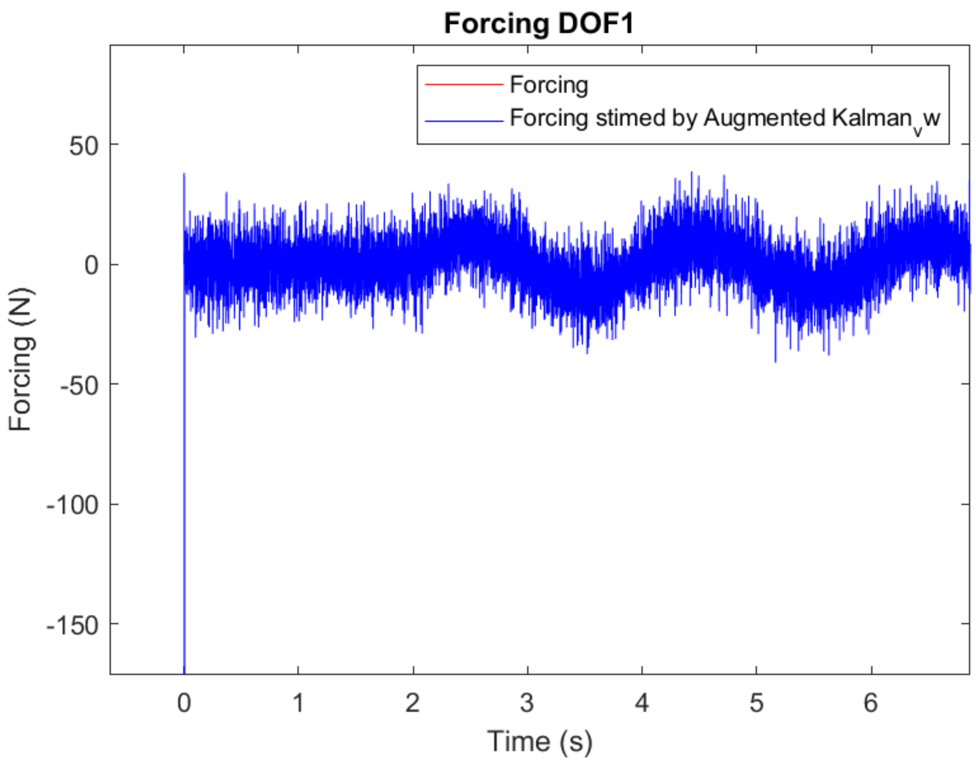


Fig. 2.7: Force estimated by the Augmented Kalman Filter.

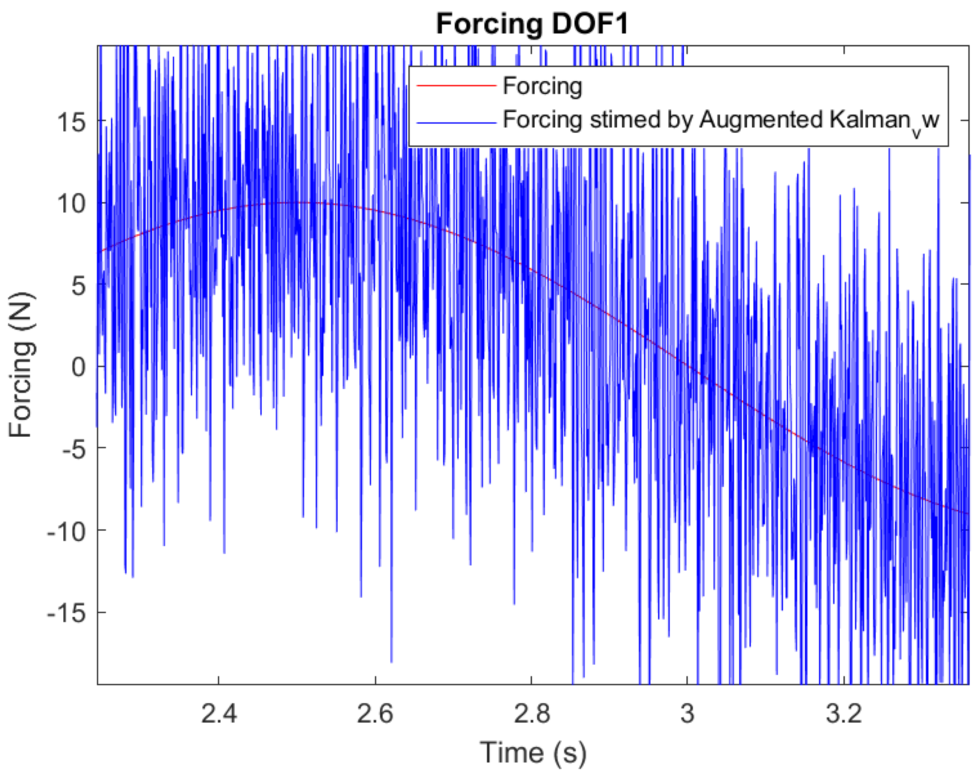


Fig. 2.8: Zoom of Force estimated by Augmented Kalman Filter.

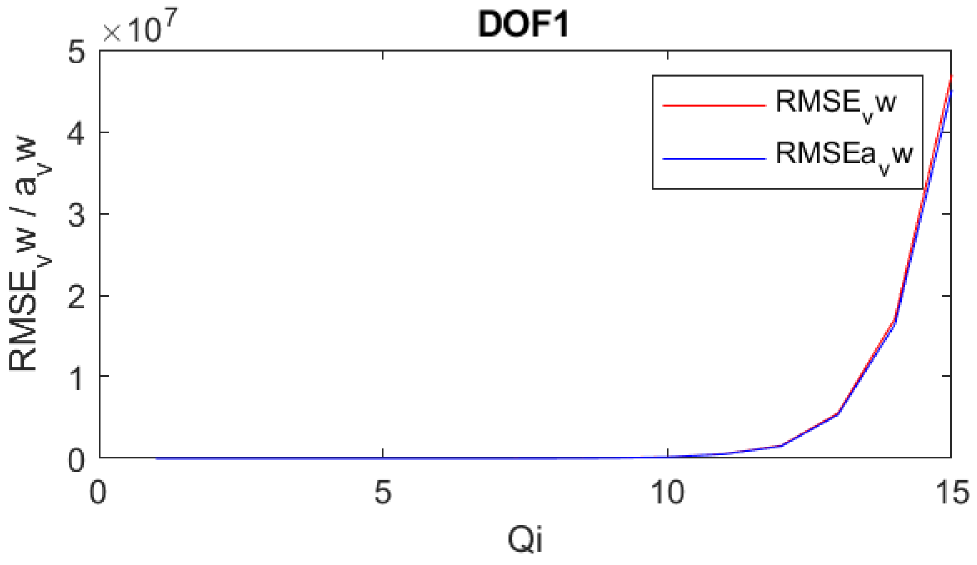


Fig. 2.9: RMSE for both Augmented KF and KF.

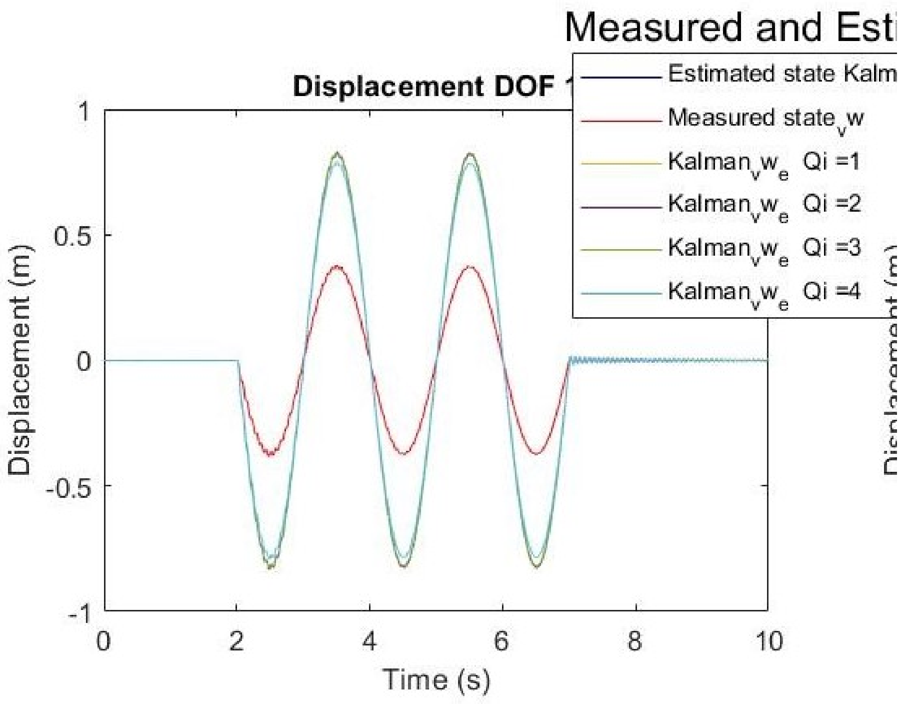


Fig. 2.10: KF predictions as a function of the covariance matrix $\bar{Q}$.

3. FINITE ELEMENT MODELING OF A BEAM MOCKUP IN SIMCENTER3D

As a first step towards the beam mockup digital twin creation consists in creating a model of this system. This task has been accomplished by means of Simcenter 3D software for CAD modeling and Finite Element Analysis (FEA). For FEA, the software generates a source .dat file, which is then processed by NX Nastran solver.

It will be just this .dat file that will be then modified by the automatic procedure for the positioning of the masses, in order to simulate a defect.

The physical model was made available to DTU who acquired the various measurements and performed the various tests.

The modeling process started from creating a model for each part and continued by assembly them into a single model. Then, FE model has been created.

Using NXopen, a Simcenter application, the boundary conditions of the FE simulation could be set. The Free-Free case and the Fixed boundary condition were implemented. The first one has been used for the model validation with the experimental tests data, while the second one has been used as a definitive model for the digital platform application.

In this Chapter, the FE model specifications and its numerical natural frequencies and modes are presented. Moreover, numerical results will be correlated with an experimental data set for model validation purposes.

3.1 FE model: Free-Free Boundary Conditions

Following the physical system structure, in Figure 1.5, the mockup model consists of four parts: the beam, which simulates the blade, the two blocks that clamp the beam, and the supporting bed plate, Figure 3.1. Materials properties used for FEM are:

- Beam : High-speed steal (HSS)
  - Mass Density (ρ) = 7800 kg/m^3
  - Young's Modulus (E) = 200000 MPa
  - Poisson's Ratio (ν) = 0.3
  - Element type = CQUAD4
  - Number of elements = 1287
- Bed plate & Block : Aluminium (EN AW6082)
  - Mass Density (ρ) = 2.71 g/cm^3
  - Young's Modulus (E) = 70000 MPa
  - Shear Modulus (G) = 0.3
  - Element type = CTETRA10
  - Number of elements = 16416
- Bolted connections : Tempered steel (34CrS4)
  - Mass Density (ρ) = 7.83 g/cm^3
  - Young's Modulus (E) = 217000 MPa
  - Poisson's Ratio (ν) = 0.28

- RBE2 connection 1D element type = CBAR

- Number of elements = 16

Gluing command was used to fix the beam between the two blocks, and to fix the main Block at the bed plate. Moreover, bolted connections were introduced to simulate the presence of the screws, Figure 3.2.

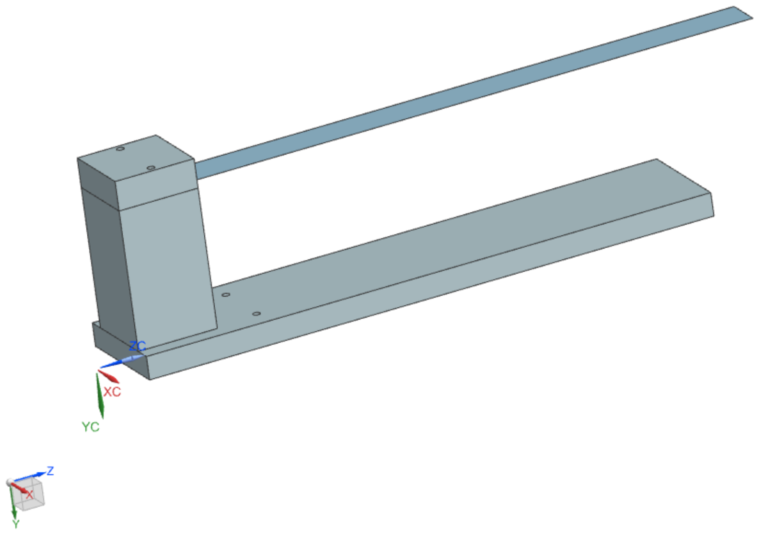


Fig. 3.1: Mockup Assembly.

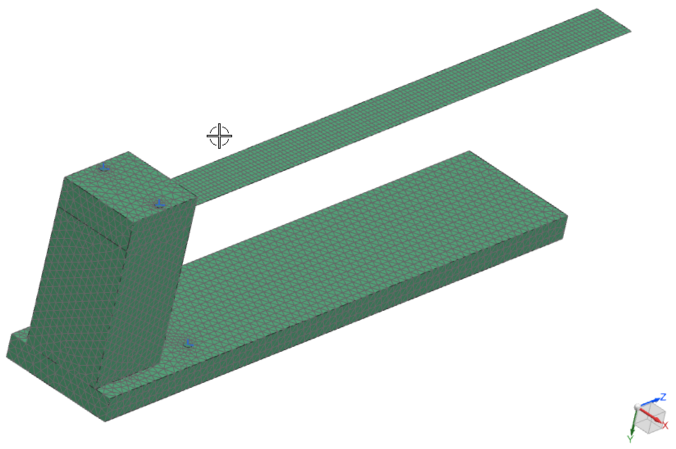


Fig. 3.2: Mockup FE model.

To obtain the free-free vibration modes, the sol-103 solution type was used, Figure 3.3. It computes the eigenvalues and outputs the vibration modes of the structure in terms of nodal displacements, accelerations and stains. The computed first 10 numerical modes are :

Modes (add)	Frequencies (Hz)
1	5.111
2	30.628
3	85.544
4	125.946
5	133.402
6	167.556
7	232.604
8	277.623
9	400.565
10	414.777

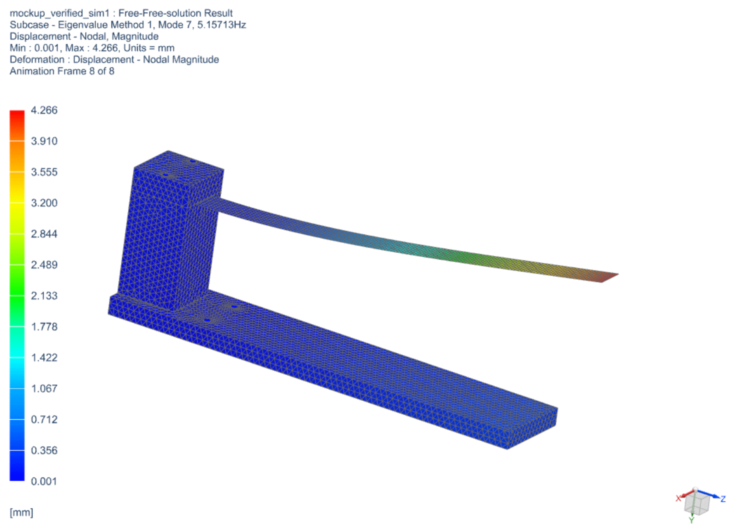

mockup_verified_sim1 : Free-Free-solution Result
Subcase - Eigenvalue Method 1, Mode 7, 5.15713Hz
Displacement - Nodal, Magnitude
Min : 0.001, Max : 4.266, Units = mm
Deformation : Displacement - Nodal Magnitude
Animation Frame 8 of 8
4.266
3.910
3.555
3.200
2.844
2.489
2.133
1.778
1.422
1.067
0.712
0.356
0.001
[mm]
X
Y
Z

Fig. 3.3: First natural modes of Mockup FE model.

3.2 Numerical and experimental mode sets correlation

In order to validate the model, a correlation between physical and numerical modes has been carried out in Simcenter3D. During this test campaign, free-free boundary conditions (BC) where adopted, i.e, the model was hung with two rubber bands to a frame, Figure 3.4.

These two rubber bands were anchored to the holes between the bed plate and the main block.

During this tests, DTU adopted the roving accelerometer technique, using two accelerometers, one fixed at the driving-point, i.e. at the fixed input position during all the measurement runs, and a second one moved over the entire geometry. The position of the fixed accelerometer, and of the input as well, is placed in the left corner of the bed plate of the mockup, as can be seen in figure 3.4, while for the second one the various position are show in figure 3.5. The experimental eigenfrequencies were evaluated through modal analysis using PoliMAX,figure 3.6, resulting in the following Frequencies:

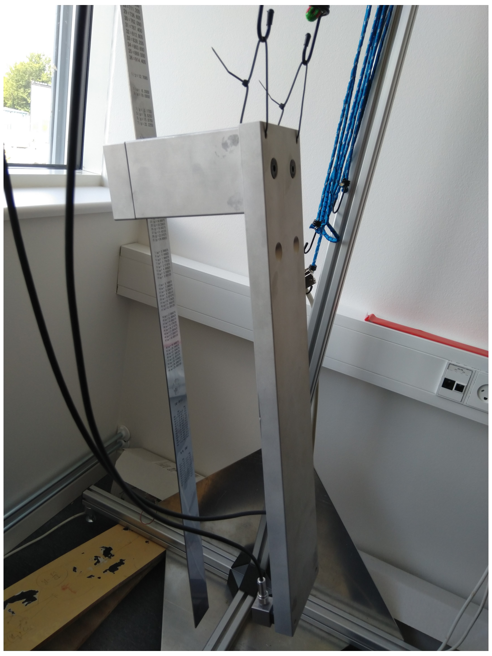


Fig. 3.4: Test setup with the mockup in free-free boundary conditions.

Modes (add)	Frequencies (Hz)
1	5.081
2	30.22
3	84.52
4	123.6
5	137.4
6	165.7
7	232.2

The experimental modes have been loaded into Simcenter 3D by importing the modes .unv file and the test geometry, figure 3.5,, from Simcenter Testlab. This software has been used by DTU to acquire measurements during a test campaign performed on the mockup and to compute its experimental modal parameters through Experimental Modal Analy-

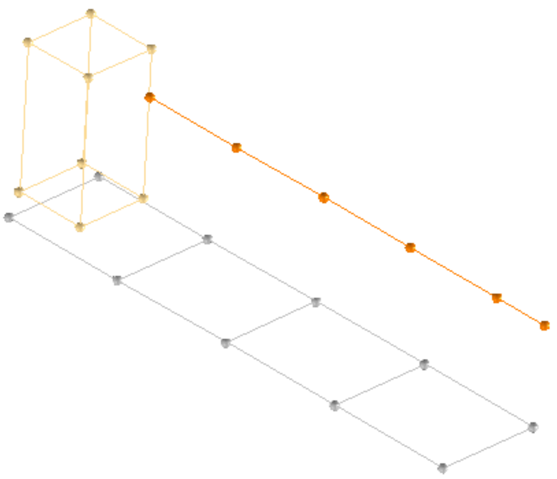


Fig. 3.5: Test geometry.

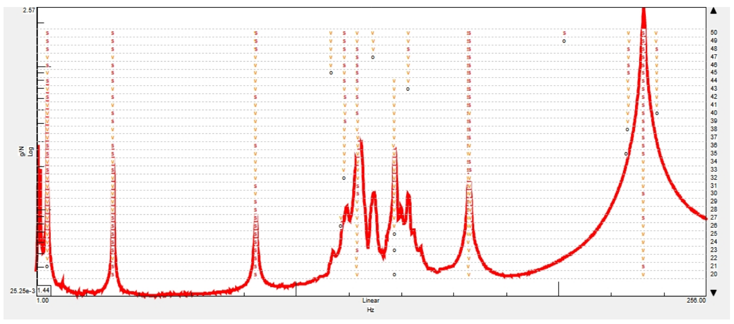


Fig. 3.6: Experimental modes computed using PolyMAX, for the Free-Free test.

sis (EMA). FE model has been aligned to the test geometry to perform mode correlation, figure 3.5 and 3.8. Normally this procedure is used to update the model parameters on the basis of the experimental ones. The Modal Assurance Criterion (MAC) reported in figure 3.9 is a correlation indicator, which shows how much two modes are similar between each other. This indicator is most sensitive to large differences and relatively insensitive to small differences in the mode shapes. The MAC was used to pair modes shapes with those obtained experimentally. It is bounded between 0 and 1, with 1 indicating fully consistent mode shapes and a value near 0 indicating that the shapes are not consistent.

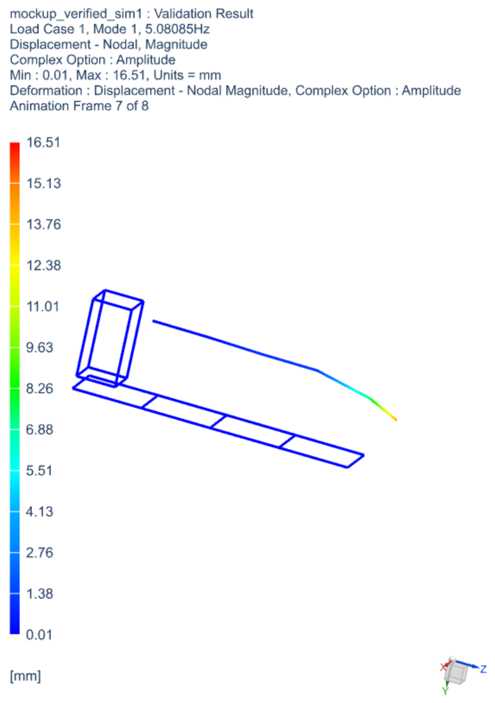

mockup_verified_sim1 : Validation Result
Load Case 1, Mode 1, 5.08085Hz
Displacement - Nodal, Magnitude
Complex Option : Amplitude
Min : 0.01, Max : 16.51, Units = mm
Deformation : Displacement - Nodal Magnitude, Complex Option : Amplitude
Animation Frame 7 of 8

Fig. 3.7: First experimental mode shape.

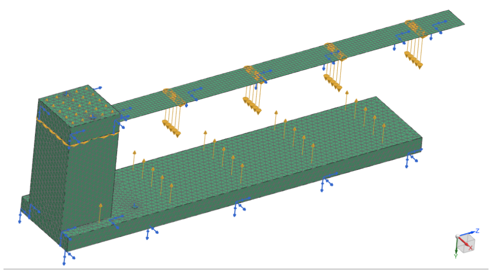


Fig. 3.8: Mockup FE model aligned with Test geometry.

The table below shows the MAC obtained by comparing experimental and simulation data.

Modes	Test Frequencies	Numerical Frequencies	MAC
(add)	(Hz)	(Hz)	(add)
1	5.081	5.2	0.788
2	30.22	30.9	0.613
3	84.52	86.3	0.938
4	123.6	127.6	0.939
5	137.4	134.0	0.422
6	165.7	169.1	0.897
7	232.2	247.8	0.963

Since the achieved correlation satisfactory, the model updating procedure was not carried out.

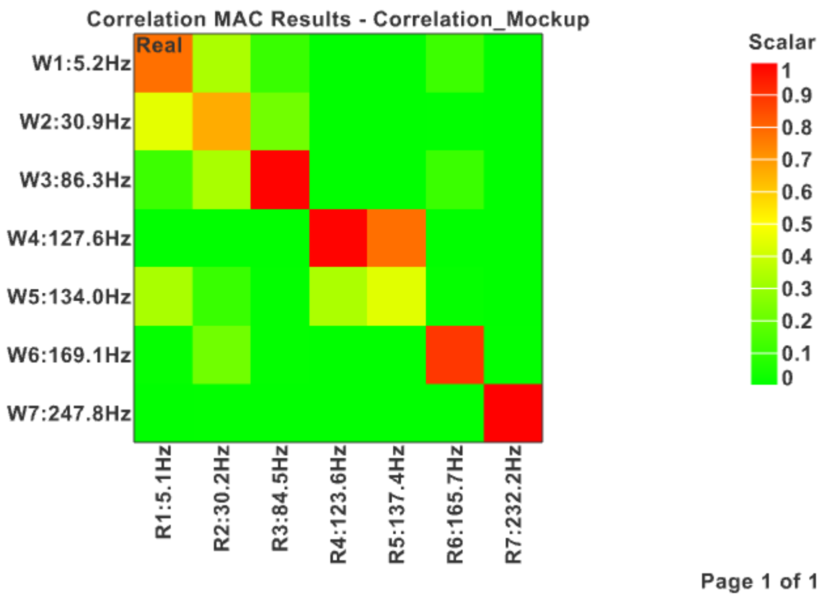


Fig. 3.9: MAC from Correlation.

3.3 FE model: Fixed Boundary Conditions

In order to obtain conditions as close as possible to the experimental ones, a fixed constraint on the 4 independent nodes of the RBE2 connections was chosen, Figure 3.10. This solution has been use to represents the condition in which the mockup is during the pull and release test. During these tests, the mockup is actually is placed on rubber supports, which prevent translations in the xz plane. Therefore, a fix constraint has been adopted to model this condition and at the same time, avoid rigid body motion could be a problem when computing the reduced order model. All the other parameters have been left unchanged. For the sake of completeness, the table of frequencies obtained by solving sol-103 the fixed case is herein reported.

Modes	Frequencies
(add)	(Hz)
1	4.869
2	30.509
3	85.447
4	92.975
5	109.422
6	133.174
7	167.548
8	277.233
9	368.223
10	400.53

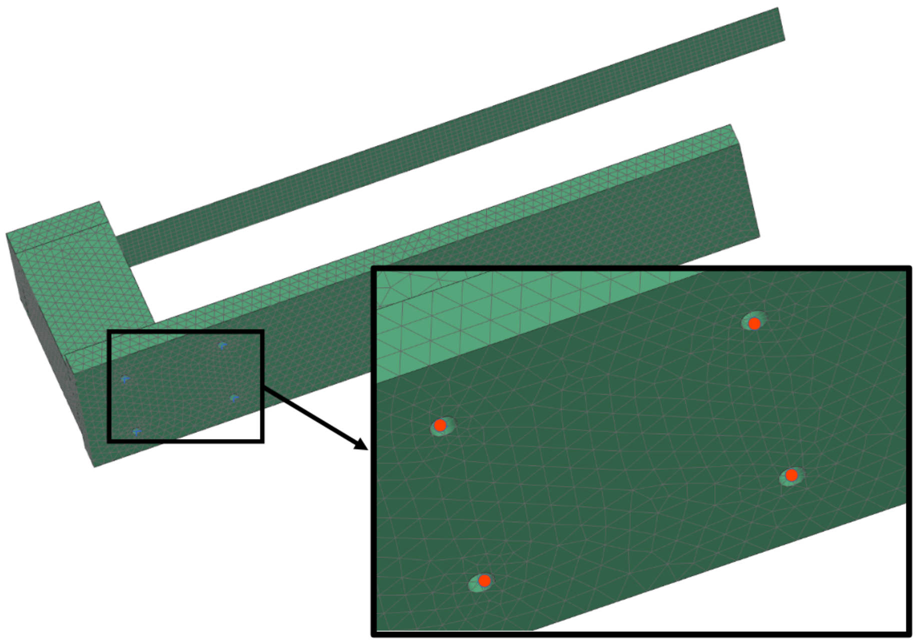


Fig. 3.10: Fixed constraint modeled in Simcenter3D.

4. AUTOMATED DAMAGE SIMULATION ON THE MOCKUP FE MODEL

This chapter will cover the process of automating the mockup FE models generation, each with a different mass position and weight, using Python.

Python was used because of its open source nature, which allowed easier compatibility with tools and application adopted within Reliablade for the digital platform development.

The chapter will start by describing the implemented procedure final goal and the adopted NX Nastran solution. A description of the script itself will be provided and finally results will be presented.

4.1 Key Goal

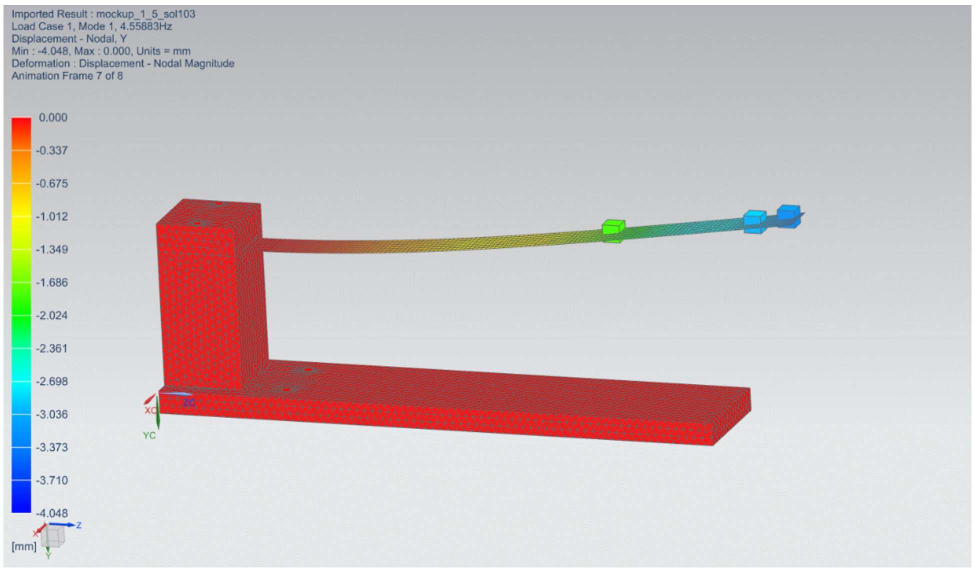
The main objective is to create a procedure that automatically generates damaged configurations of the original FE models, by setting several mass positions and weights to simulate the damage.

The mockup FE model described in chapter 3.3 has been used as original undamaged configuration, to which the additional mass representing the damage could be added.

Two additional 1 gram masses were added at the accelerometers positions to consider their influence, Figure 4.1.

Modal parameters for the modified model and its time response during a simulated pull and release test the pull and release test have been implemented in an automated manner.

Regarding time response, only those elements, for the strains, and nodes, for the accelerations, that correspond to the physical sensorshave been simulated. During pull and release tests, the mockup was instrumented with 5 strain gauges and 2 accelerometers, figure 1.6.



```
Imported Result : mockup_1_5_sol103
Load Case 1, Mode 1, 4.55883Hz
Displacement - Nodal, Y
Min : -4.048, Max : 0.000, Units = mm
Deformation : Displacement - Nodal Magnitude
Animation Frame 7 of 8
```

Fig. 4.1: First mode obtained from Nastran sol 103 with additional mass of 8 grams at 430mm from the root.

NX Nastran SOL 103 has been adopted to compute mode shapes and frequencies for each mockup model damaged configuration.

Two types of simulation are available to simulate the model in the time domain by providing external input, figure 4.2:

- **sol 109** : performs direct transient response analysis. In direct transient response analysis, the structural response is computed by solving the equation of motion using direct numerical integration. The physical solution is then obtained through the summation of the individual modal responses(doesn't support damping).
- **sol 112** : performs modal transient response analysis. Modal transient response analysis is another method for computing the transient response of a structure. This method uses the mode shapes of the structure to reduce the size, uncouple the equations of motion (when no damping or only modal damping is used), and make the numerical integration more efficient (when the equations of motion are uncoupled).

Given that a high number of models need to be simulated within the developed procedure, a physical resolution would required more computational time. Therefore, SOL 112 has been adopted.

In addiction, it gives the possibility of including modal damping into the model.

Advantages can be summarized as follows:

- Reduce the problem size because not all of the modes are typically calculated or retained.
- Uncouple the equations of motion when either no damping or only modal damping is used.

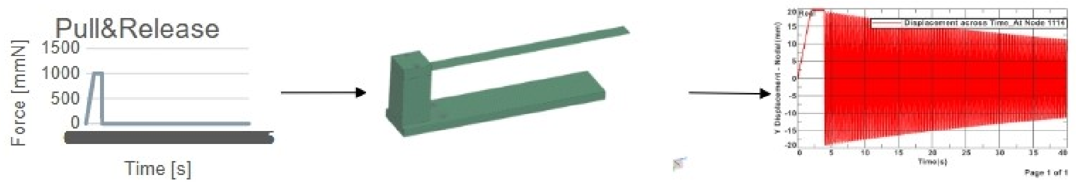

Pull&Release
1500
1000
500
0
Force [mmN]
Time [s]
Displacement across Time_At Node 1114
Y Displacement - Nodal (mm)
Time(s)
Page 1 of 1

Fig. 4.2: Mockup FE model time response (displacement) to a typical pull&release input profile at a specific location.

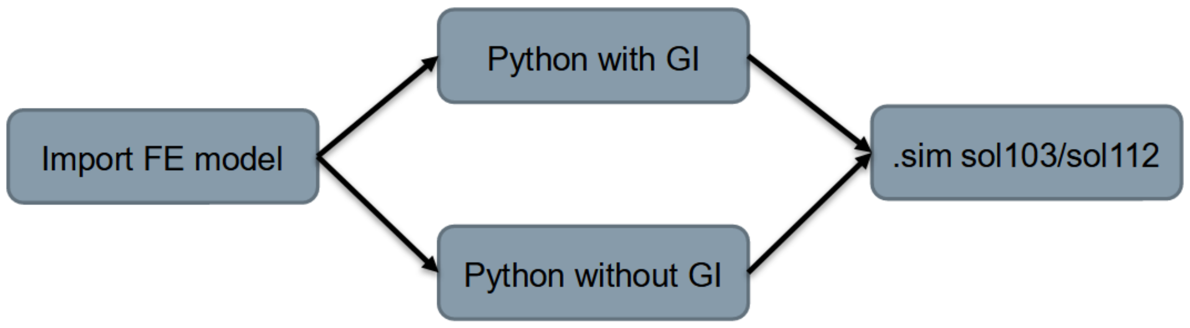


Fig. 4.3: Flow chart for the 2 path.

4.2 Python Script

Tow types of scripts were created, Figure 4.3 : the first uses the graphical interface through NXOpen packages, while the second directly reads and modifies the .dat file.

Both have the following features:

- Import an already existing FEM;
- Add a concentrated mass on a specific node (defined by giving its cartesian coordinates);
- Choose input location and amplitude (both static and pull&release force types);
- Select locations where response has to be recorded (defined in cartesian coordinates) for both accelerometers and strain gauges;
- Solve SOL103, SOL112 with external force applied and/or SOL112 with initial conditions only applied;
- Store results in csv format for all the solutions.

In order to simulate a pull and release test on the mockup, NX Nastran SOL 112 has been developed by following two different methods: the first consists in applying a load (created within Python) at a node of the model, while the second one provides initial conditions taken from a static solution with a static load comparable to the one adopted during the pull and release test.

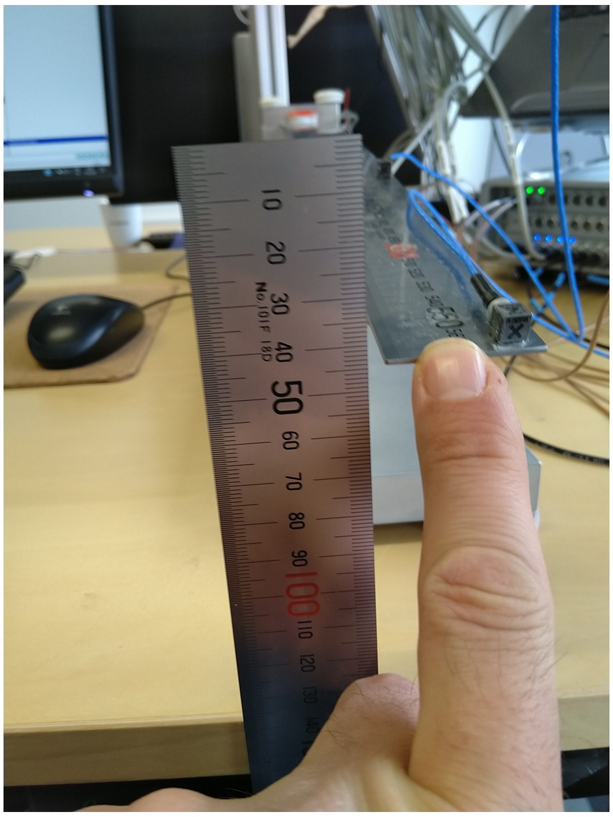
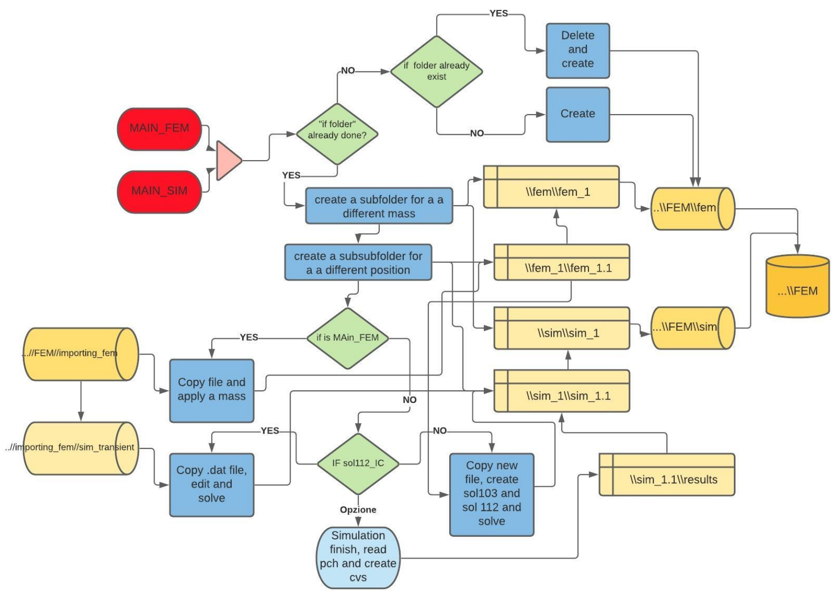


Fig. 4.4: Pull&Release test on the mockup.



MAIN_FEM
MAIN_SIM
"if folder" already done?
YES
NO
if folder already exist
YES
NO
Delete and create
Create
create a subfolder for a a different mass
create a subsubfolder for a a different position
\\fem\\fem_1
\\fem_1\\fem_1.1
\\sim\\sim_1
\\sim_1\\sim_1.1
..\\FEM\\fem
...\\FEM\\sim
...\\FEM
if is MAin_FEM
YES
NO
..//FEM//importing_fem
..//importing_fem//sim_transient
Copy file and apply a mass
Copy .dat file, edit and solve
IF sol112_IC
YES
NO
Copy new file, create sol103 and sol 112 and solve
\\sim_1.1\\results
Opzione
Simulation finish, read pch and create cvs

Fig. 4.5: Flow chart of the first procedure, describing the output.

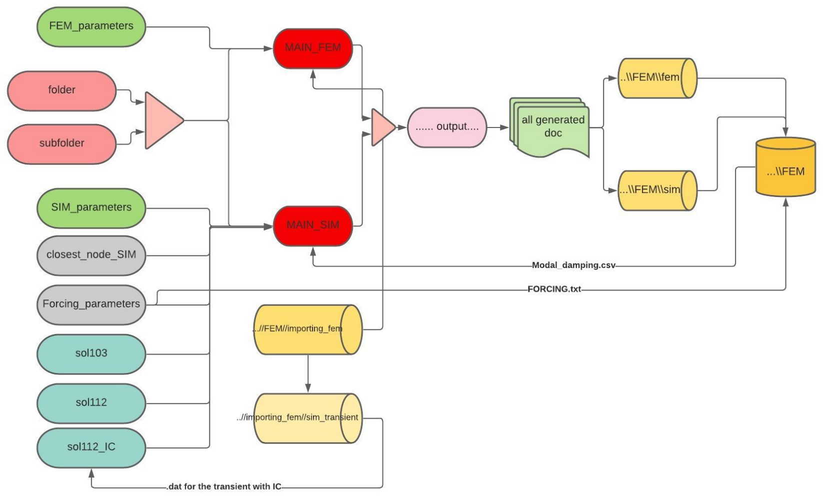


Fig. 4.6: Flow chart of the first procedure, describing the input.

The first path consisted of recording of the actions performed in the graphical interface of NXOpen and with appropriate modifications. This allows to obtain a complete code that imports the FE model and modifies it with the purpose of getting the various required simulations. To do so , CAD, FEM and simulation files need to be stored for each mass case into the corresponding folder, figure 4.5. This implies a large amount of

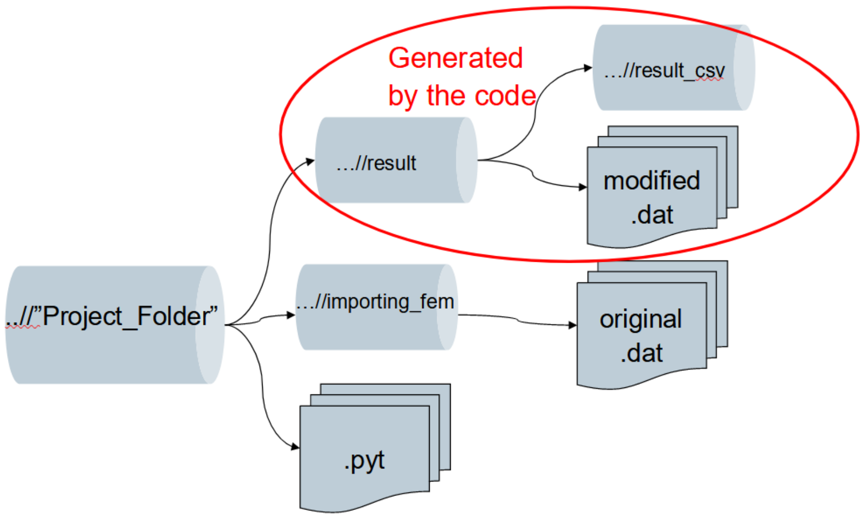


Fig. 4.7: Structure of the project folder.

data to be streamed to the cloud.

Therefore, in order to avoid server overloading, only the second procedure has been developed to be implemented in the cloud. For this reason, the first procedure will not be described in details herein. Figure 4.5, 4.6.

The second script ideally imports the original FEM from the dat source file. The .dat file is a text file containing all the information about the FE model developed using the NXOpen graphical environment.It contains a list of nodes and elements, geometry information, constraints and materials.

Three starting dat files were prepared, one for sol 103, one for sol 112 with force applied and one for sol 112 with initial conditions.

These source files were obtained directly from Simcenter once all the simulations had been set up, without adding any additional mass.

Figure 4.7 shows that the project folder contains the script files and two folders : "importing_fem", where there are the original .dat file is stored, and the "Result" folder, generated by the code, that contains the modified .dat files and their result files. "Result_csv" folder instead, contains the results in csv format.

The csv file have been obtained reading the punch results file, generatedby NX Nastran. The entire script works by running the MAIN_SIM.py code, which takes the necessary dictionaries and functions from the other .py files and then performs the necessary actions to modify and to simulate the model for all masses weights and positions, i.e. for cycles.

By rewriting appropriate lines if the dat source file, was possible to add additional masses, select nodes and elements from which to obtain simulation responses and define required input time histories.

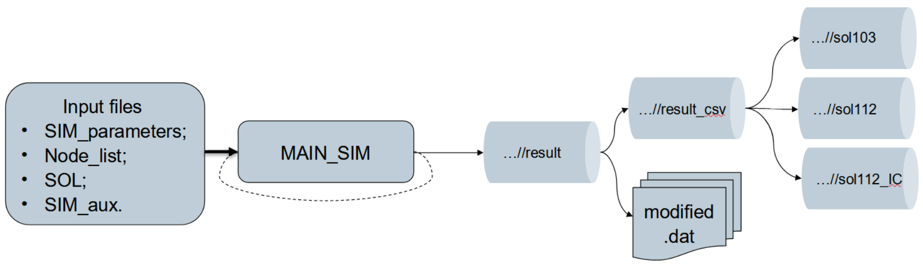


Fig. 4.8: Flow chart of second script.

The various functions that the MAIN_SIM file runs are described in the following. .

SIM_parameters

SIM_parameters.py can be arbitrarily modified by the user, who can:

- Define the nastran.exe path on the local machine;
- Assign several mass values;
- Choose the positions for the various masses;
- Choose strain gauges and accelerometers positions using dictionaries;
- Add a concentrated mass at the accelerometers locations;
- Choose the force position and its amplitude;
- Choose the time step and number of samples for the SOL112/SOL112_IC.

Node_list

It contains the function used to :

- Generate the list of nodes with coordinates, reading the .dat file;
- Find the nearest node given the coordinates (from SIM_parameters);
- Find the nearest element given the coordinates (for strain gauges);

- Generate the input time history and stores it in csv format;
- Compute the rotation matrix to transfer the elemental strain values from local to global coordinates.

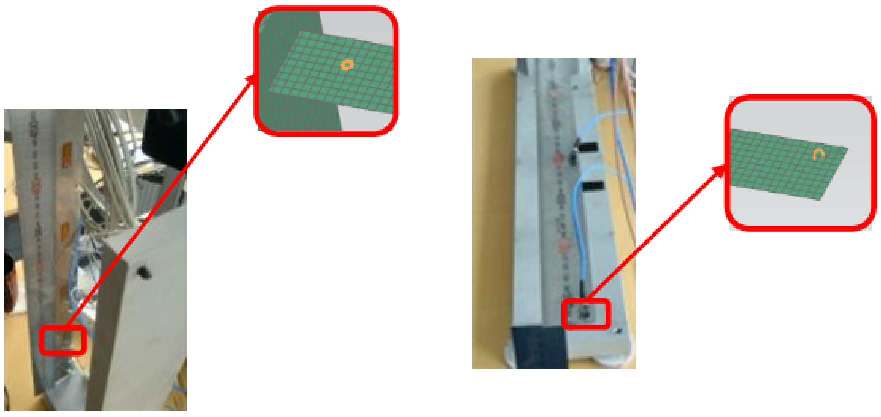


Fig. 4.9: Identification of closest element and nodes.

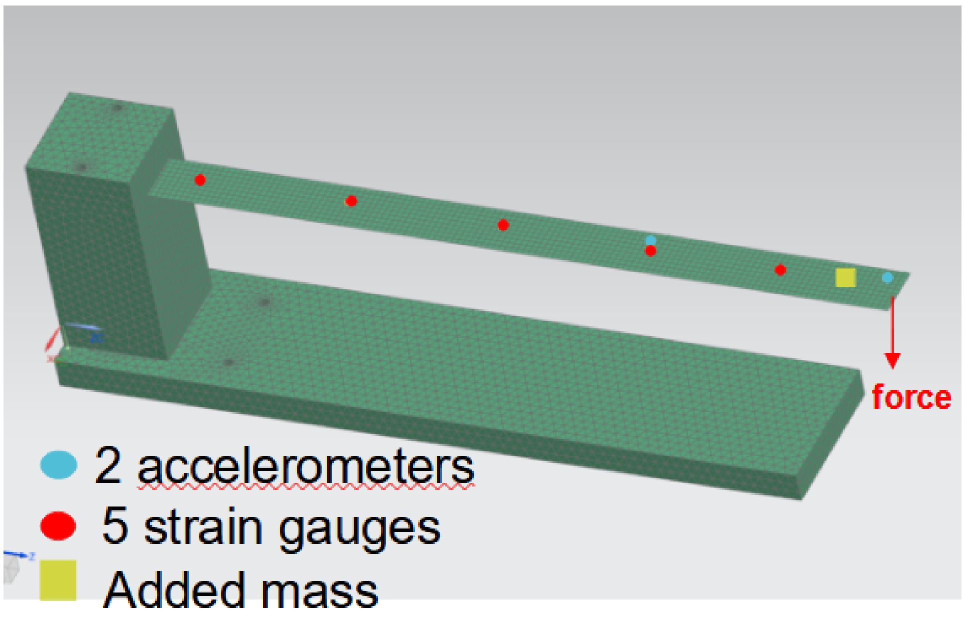


Fig. 4.10: Features implemented by the script on the FE model: input location and amplitude, mass damage location and weight, sensors locations for response simulation.

SOL

It contains all the function for:

- Reading modal damping and force time history from csv files;
- Rewriting the .dat files for SOL103, SOL112 and SOL112_IC;
- Running Nastran with the new .dat.

SIM_aux

This last input file contains the function for generating the output:

- Functions to read the .pch files and create the csv files;
- The function to create subfolders;
- The function to create .log files.

The .log file is simply a txt file in which all the various states of the script, and any errors, are reported.

4.3 Results

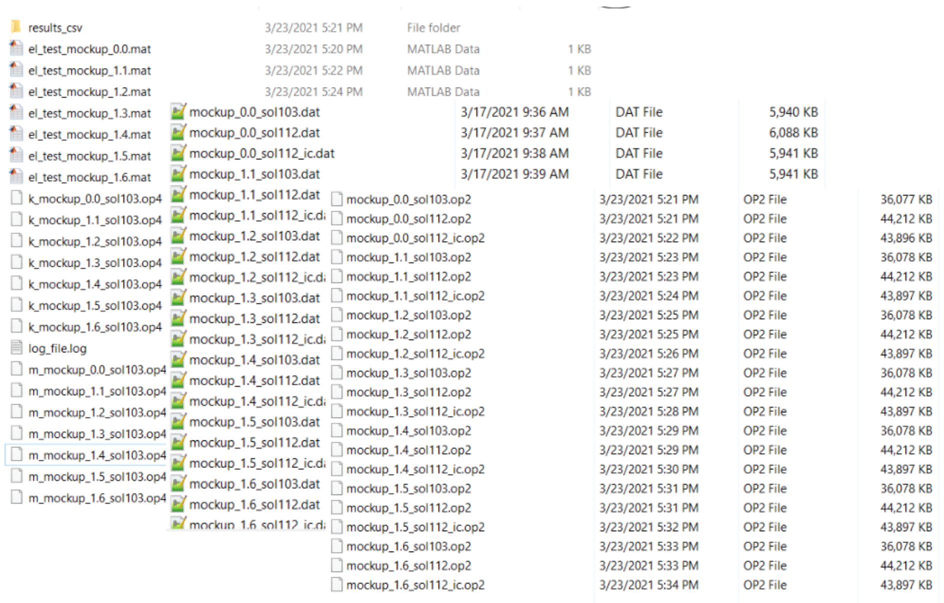
Figure 4.11 shows the various modified .dat files in the result folder, renamed according to mass and position.

In addition, mass and stiffness matrices are also extracted in op4 format. These matrices will be used to build the Reduced Order Model for Virtual Sensing purposes.

It is also possible to choose from python whether to simulate the model without any mass, mockup_0.0_sol103 file in figure 4.11 is an example.

The simulation was performed with a 8 grams mass at 5 different locations. Only the results for the case in which the mass is at the 5th position (430 mm from the beam root) are herein reported. the fifth position.

Figures 4.12 and 4.13 show respectively the model with the added mass and the response time history in terms of displacement at node 1092 of the model, i.e., the position of the second accelerometer. For SOL 112, the experimental damping factor was imported in csv format from TESTLAB.

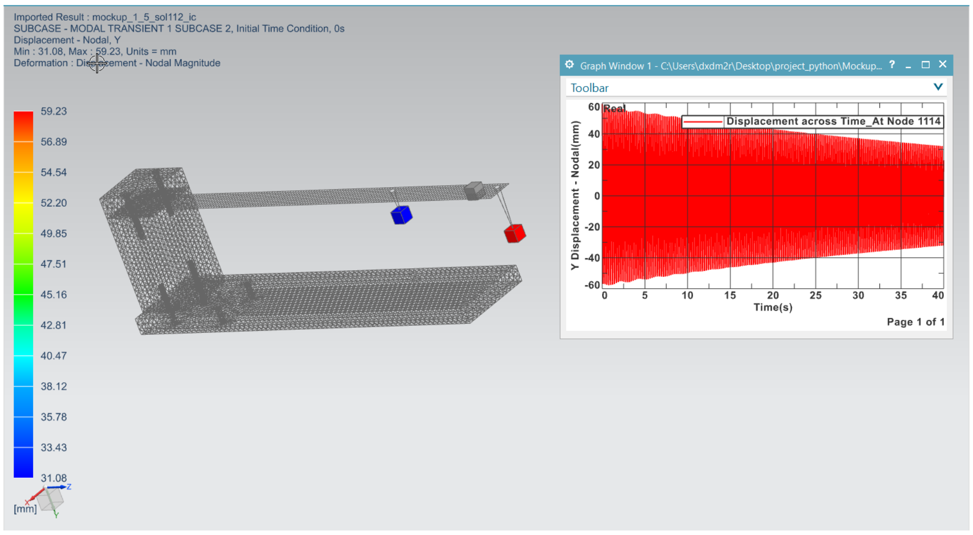


```
results_csv                  3/23/2021 5:21 PM    File folder
el_test_mockup_0.0.mat       3/23/2021 5:20 PM    MATLAB Data    1 KB
el_test_mockup_1.1.mat       3/23/2021 5:22 PM    MATLAB Data    1 KB
el_test_mockup_1.2.mat       3/23/2021 5:24 PM    MATLAB Data    1 KB
el_test_mockup_1.3.mat
el_test_mockup_1.4.mat
el_test_mockup_1.5.mat
el_test_mockup_1.6.mat
k_mockup_0.0_sol103.op4
k_mockup_1.1_sol103.op4
k_mockup_1.2_sol103.op4
k_mockup_1.3_sol103.op4
k_mockup_1.4_sol103.op4
k_mockup_1.5_sol103.op4
k_mockup_1.6_sol103.op4
log_file.log
m_mockup_0.0_sol103.op4
m_mockup_1.1_sol103.op4
m_mockup_1.2_sol103.op4
m_mockup_1.3_sol103.op4
m_mockup_1.4_sol103.op4
m_mockup_1.5_sol103.op4
m_mockup_1.6_sol103.op4
```

```
mockup_0.0_sol103.dat       3/17/2021 9:36 AM    DAT File    5,940 KB
mockup_0.0_sol112.dat       3/17/2021 9:37 AM    DAT File    6,088 KB
mockup_0.0_sol112_ic.dat    3/17/2021 9:38 AM    DAT File    5,941 KB
mockup_1.1_sol103.dat       3/17/2021 9:39 AM    DAT File    5,941 KB
mockup_1.1_sol112.dat
mockup_1.1_sol112_ic.d
mockup_1.2_sol103.dat
mockup_1.2_sol112.dat
mockup_1.2_sol112_ic.d
mockup_1.3_sol103.dat
mockup_1.3_sol112.dat
mockup_1.3_sol112_ic.d
mockup_1.4_sol103.dat
mockup_1.4_sol112.dat
mockup_1.4_sol112_ic.d
mockup_1.5_sol103.dat
mockup_1.5_sol112.dat
mockup_1.5_sol112_ic.d
mockup_1.6_sol103.dat
mockup_1.6_sol112.dat
mockup_1.6_sol112_ic.d
```

```
mockup_0.0_sol103.op2       3/23/2021 5:21 PM    OP2 File    36,077 KB
mockup_0.0_sol112.op2       3/23/2021 5:21 PM    OP2 File    44,212 KB
mockup_0.0_sol112_ic.op2    3/23/2021 5:22 PM    OP2 File    43,896 KB
mockup_1.1_sol103.op2       3/23/2021 5:23 PM    OP2 File    36,078 KB
mockup_1.1_sol112.op2       3/23/2021 5:23 PM    OP2 File    44,212 KB
mockup_1.1_sol112_ic.op2    3/23/2021 5:24 PM    OP2 File    43,897 KB
mockup_1.2_sol103.op2       3/23/2021 5:25 PM    OP2 File    36,078 KB
mockup_1.2_sol112.op2       3/23/2021 5:25 PM    OP2 File    44,212 KB
mockup_1.2_sol112_ic.op2    3/23/2021 5:26 PM    OP2 File    43,897 KB
mockup_1.3_sol103.op2       3/23/2021 5:27 PM    OP2 File    36,078 KB
mockup_1.3_sol112.op2       3/23/2021 5:27 PM    OP2 File    44,212 KB
mockup_1.3_sol112_ic.op2    3/23/2021 5:28 PM    OP2 File    43,897 KB
mockup_1.4_sol103.op2       3/23/2021 5:29 PM    OP2 File    36,078 KB
mockup_1.4_sol112.op2       3/23/2021 5:29 PM    OP2 File    44,212 KB
mockup_1.4_sol112_ic.op2    3/23/2021 5:30 PM    OP2 File    43,897 KB
mockup_1.5_sol103.op2       3/23/2021 5:31 PM    OP2 File    36,078 KB
mockup_1.5_sol112.op2       3/23/2021 5:31 PM    OP2 File    44,212 KB
mockup_1.5_sol112_ic.op2    3/23/2021 5:32 PM    OP2 File    43,897 KB
mockup_1.6_sol103.op2       3/23/2021 5:33 PM    OP2 File    36,078 KB
mockup_1.6_sol112.op2       3/23/2021 5:33 PM    OP2 File    44,212 KB
mockup_1.6_sol112_ic.op2    3/23/2021 5:34 PM    OP2 File    43,897 KB
```

Fig. 4.11: All file generated by the code.


Imported Result : mockup_1_5_sol112_ic
SUBCASE - MODAL TRANSIENT 1 SUBCASE 2, Initial Time Condition, 0s
Displacement - Nodal, Y
Min : 31.08, Max : 59.23, Units = mm
Deformation : Displacement - Nodal Magnitude

59.23
56.89
54.54
52.20
49.85
47.51
45.16
42.81
40.47
38.12
35.78
33.43
31.08
[mm]
Graph Window 1 - C:\Users\dxdm2r\Desktop\project_python\Mockup...
Toolbar
Real
Displacement across Time_At Node 1114
Y Displacement - Nodal(mm)
Time(s)
Page 1 of 1

Fig. 4.12: Generated damaged model (8 grams mass at 430 mm from the blade root) and simulated response time history.

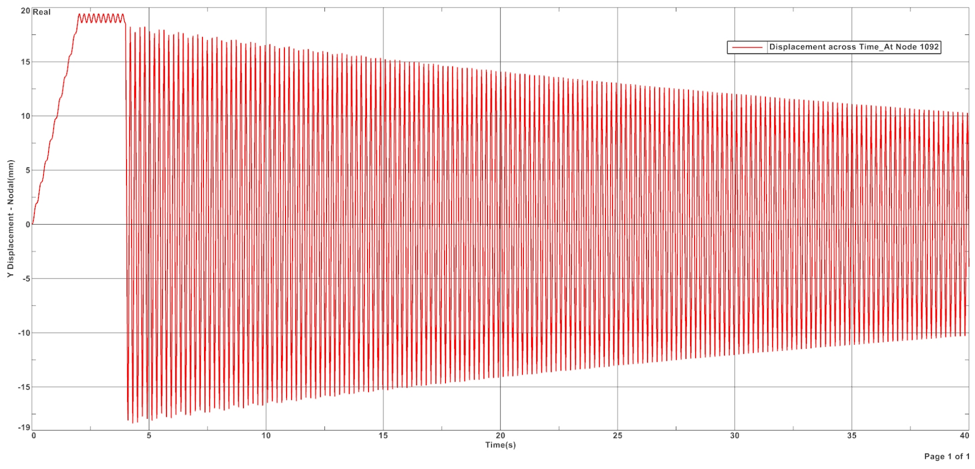


Fig. 4.13: Solution 112 with input time-history , defect of additional mass of 8 grams at 430mm from the root (5Th position).

5. INPUT AND FULL FIELD RESPONSE PREDICTION ON THE MOCKUP

In this chapter we will discuss how the FE models generated by the Python code were imported into Matlab and the implementation of AKF to obtain the full-field response, starting with experimental measurements.

This will be used for the new ML algorithm, which will be based on time data from strain gauges and accelerometers. This concludes the 3 goals.

5.1 Input and full field response prediction using a simulated data set

As a first step, the KF and AKF were applied to the beam mockup in a simulation environment.

Modal coordinates have been used for the beam mockup FE model simulation, transforming therefore all the matrices and vectors thanks to the modal matrix Φ. **[6]**,**[8]**

$$\vec{x}(t) = \bar{\Phi}^{n\ x\ m} * \vec{q}(t)$$

where m is number of modes, $\Phi \in \Re^{ndnof*m}$ is the reduction basis and $q(t) \in \Re^{m}$ is the vector of the generalized coordinates of the system. This will reduce the number of equations to be used by selecting through m, the number of modes. For this application, only the first 10 modes have been taken into account.

To obtain the mass and stiffness matrices for the model, the .op4 files from Simcenter were imported in MATLAB. Then the reduced matrices have been computed using the reduction basis as follows:

$$\bar{M}_r = \bar{\Phi}^T \bar{M} \bar{\Phi}$$

$$\bar{K}_r = \bar{\Phi}^T \bar{K} \bar{\Phi}$$

To obtain the damping matrix instead, a proportionality to $\bar{K}_r$

$$\bar{C}_r = \alpha \bar{K}_r = \frac{2\zeta}{\omega_1} \bar{K}_r$$

Where $\zeta = 0.0012$ is the experimental modal damping ratio and ω_1 is the first natural frequency of the system. A sinusoidal input (10 N amplitude, 0.5 Hz) has been applied to the tip of the beam (node 1124) in the vertical direction to simulate the system responses. Noise has been added to the measured responses and only the nodes showed in figure 5.1 have been used as observations in the filter. The input and the remaining responses have been estimated using the AKF (KF for responses estimation only). Figures 5.2, 5.3, 5.4 show that the KF and AKF can successfully estimate displacement and acceleration at specific nodes of the FE model, as well as the applied input (for the AKF).

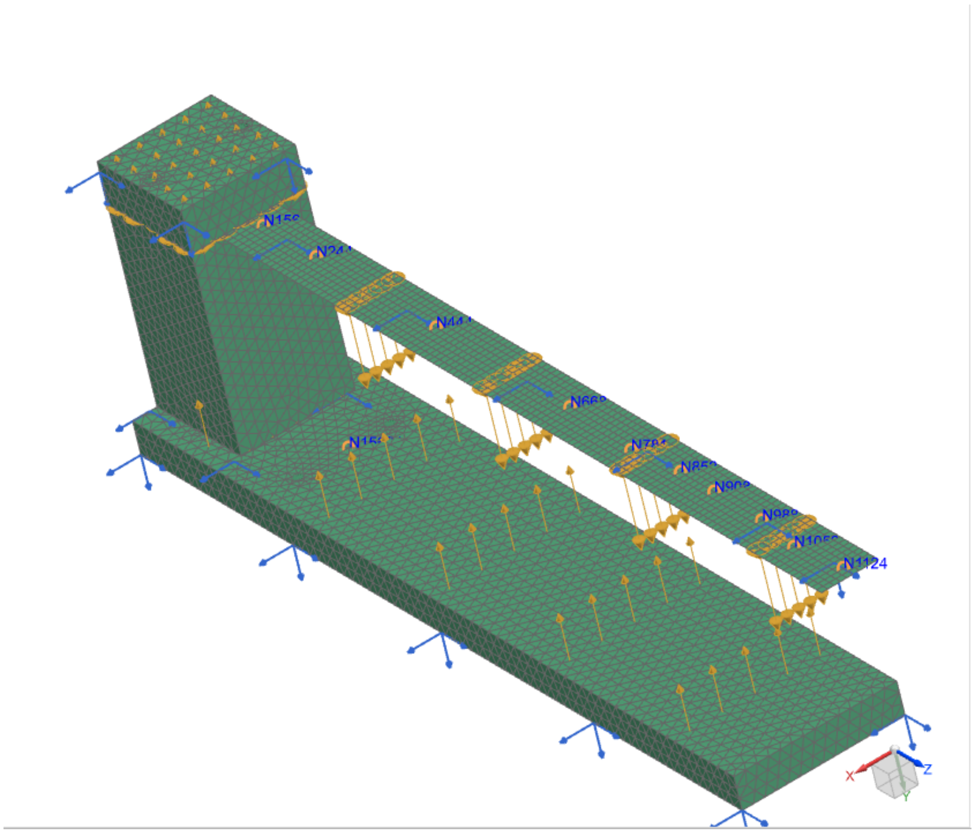


Fig. 5.1: FE model with simulated points highlighted .

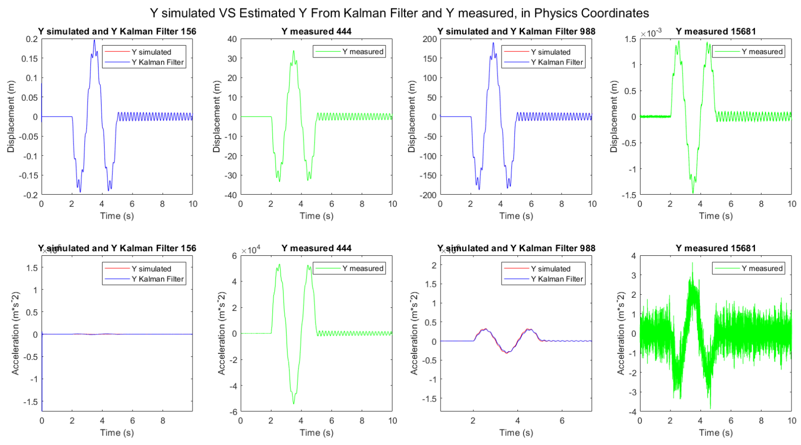


Fig. 5.2: Response predictions by the KF compared to the simulate on the Mockup FE model.

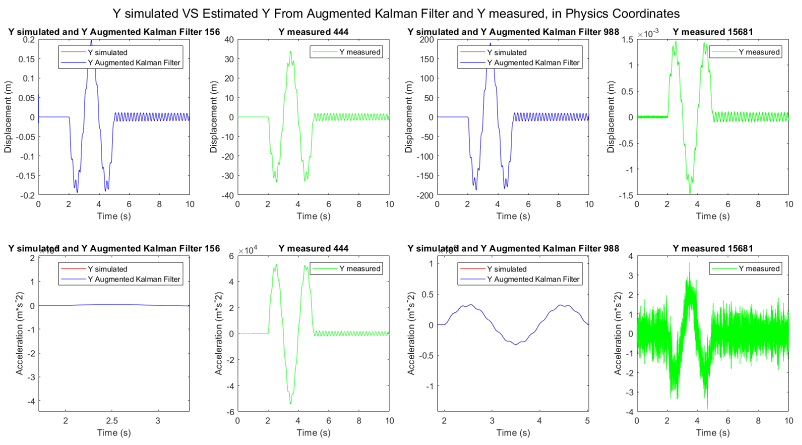


Fig. 5.3: Response predictions by the AKF compared to the simulate on the Mockup FE model.

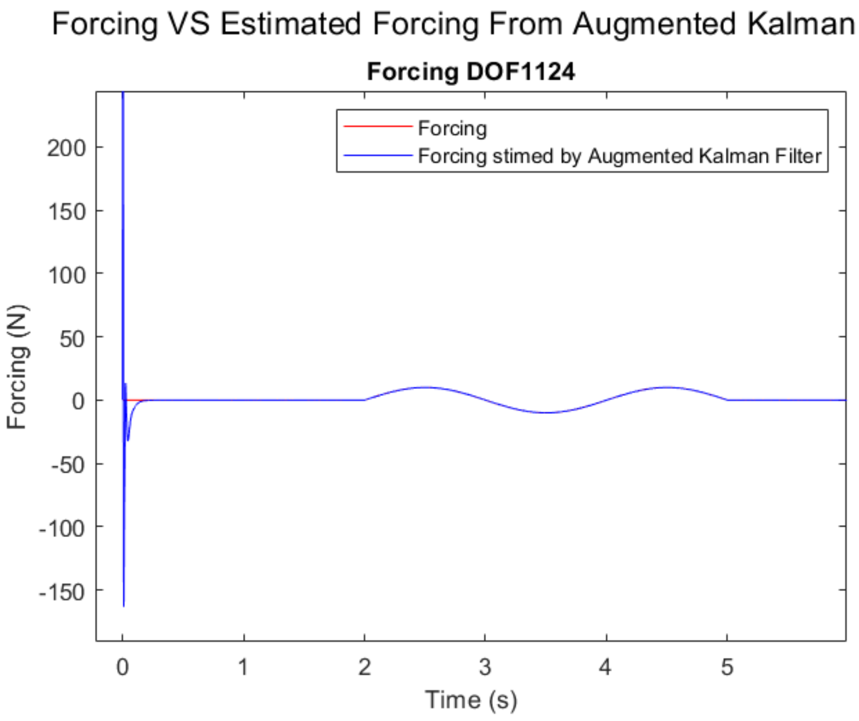


Fig. 5.4: Sinusoidal input estimation by the AKF.

5.2 AKF implementation using MATLAB

A Matlab script was used to iterate the AKF for the different FE model damaged configurations obtained from the Python code. This choice was made since the AKF algorithm had already been implemented in MATLAB for the original model.

In addition to the already available script, the various .op4 files have been imported using a special OP4Reader.m function developed by Siemens for reading and extracting the mass and stiffness matrices of FE models.

To build the reduced model, the first 10 modes obtained from sol103 were considered and merged with the attachment mode, i.e. a static mode obtained from the static solution computed when a unitary input is applied at the DOF at which the actual input is applied during tests.**[6]**,**[8]**

The reduced model is derived by applying the following:

$$x(t) = \Phi^{n_{dof}\ x\ n_{red}} * q(t)$$

Where:

$$x(t) \rightarrow Physical\ cordinates\ with\ n_{dof}\ =\ number\ od\ DOF\ of\ the\ FE\ model$$

$$q(t) \rightarrow Generalized\ coordinates\ with\ n_{red} = n_k + n_a$$

$$n_k =\ number\ of\ normal\ mode$$

$$n_a =\ number\ of\ attachment\ mode$$

And the matrix Φ is :

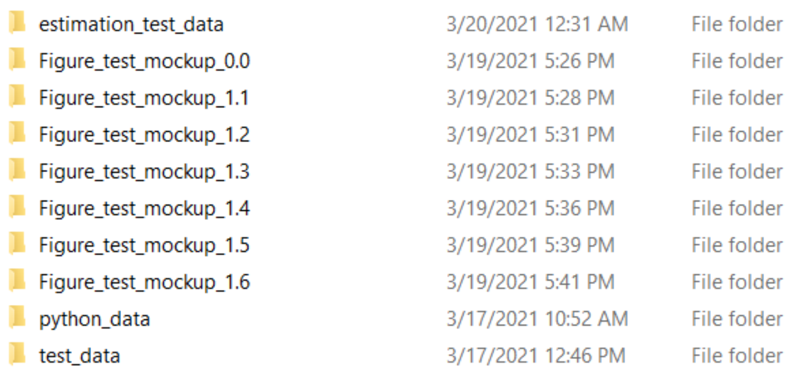
$$\Phi = [\Phi_k\ \Phi_a]$$

with:

- Φ_k : normal modes matrix;
- Φ_a : 1 attachment mode, a static mode resulting by the application of a unitary input the beam tip.

The damping matrix was considered proportional to the stiffness matrix, as discussed in the theoretical section.
The procedure described above has been iterated for all the available matrices belonging to the several damaged model configurations. For this reason, it was decided to rename all cases, as in Python script, "mockup_1.3_sol103", the first digit indicates the weight of the mass placed as a defect, the second its position and then the type of simulation. Input-response estimation results are stored, for each mass case as well, Figure 5.5.

AKF has been applied by using measurements from the pull&release experimental tests described in figure 1.10. To assess the efficiency of the filter only the measurements of two strain gauges and one accelerometer were taken as observations in the AKF, while the others were estimated so that estimated quantities could be compared with measurements. Fore the AKF, measurement noise covariance has been determined from measured time histories and the input process noise covariance has been set to 10^5 by trial and error.



estimation_test_data	3/20/2021 12:31 AM	File folder
Figure_test_mockup_0.0	3/19/2021 5:26 PM	File folder
Figure_test_mockup_1.1	3/19/2021 5:28 PM	File folder
Figure_test_mockup_1.2	3/19/2021 5:31 PM	File folder
Figure_test_mockup_1.3	3/19/2021 5:33 PM	File folder
Figure_test_mockup_1.4	3/19/2021 5:36 PM	File folder
Figure_test_mockup_1.5	3/19/2021 5:39 PM	File folder
Figure_test_mockup_1.6	3/19/2021 5:41 PM	File folder
python_data	3/17/2021 10:52 AM	File folder
test_data	3/17/2021 12:46 PM	File folder

Fig. 5.5: MATLAB folder structures.

5.3 Results

This paragraph shows the prediction results obtained using the AKF during the pull&release test on the mockup with a 8 grams mass damage located at 430 mm from the beam root figure 5.6. It should be noted that the measurements and simulations were cut and adjusted to have the same time step (T_s) and the same start time.

Predictions obtained by AKF are herein reported for the second accelerometer and the second strain gauge, which were not given as observations to the filter, Figure 5.6.

The plots show the comparison between the measured, in black, the simulated from the model, in grey, and estimated by the AKF in light blue (Figures 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, 5.13). Additionally, prediction results achieved by using strains and accelerations as observations have been compared to the ones that can be achieved by only using strains within the observations set (Figure from 5.14 to 5.20).

The FFT was also calculated for each time history with the help of an exponential window, Figures 5.10, 5.11, 5.17, 5.18.

Figures 5.7 and 5.8 demonstrate that the AKF corrects for model error and makes the estimates follow the time histories of the measurements very well. A detailed view, figure 5.9, show how quickly the estimates match the measurements.

It can be seen that as far as FFTs are concerned, strain FFTs are well estimated and the various peaks are identified, figure 5.11, while accelerometer FFTs, figure 5.10 require a better tuning of the exponential window used.

Figure 5.12 compares the estimated quantity RMS to the measured and simulated quantities ones for each sensors. It can be seen that there are notable improvements compared to those of the model simulated in Simcenter. The AKF can correct for the error in the evaluation of the model's modal damping coefficient, and more accurately estimate the responses.

In figure 5.13 the input time history estimated via the AKF is compared to a simulated time history, which reproduces the input profile adopted during the physical pull&release test. Since the actual force applied through the finger could not be measured, the input profile shown in figure 5.13 has been reconstructed by retrieving from the model the static input which could cause the deformation experienced during the first phase of the pull&release test. To quantify the correctness of the input estimation, the following error estimators have been calculated :

- Static Error (SE): mean value of the estimated input profile after the beam release;
- Standard Deviation (SD): standard deviation of the estimated time history after the beam release.

For the other comparisons in general it is inferred that not considering the accelerations in the observations causes a decrease in the filter reliability and the estimations to require more time to match the measurements, as can be seen from the detailed view in figure 5.16. The same conclusion can be drawn by analyzing the RMS comparison for the accelerations reported in figure 5.19. With regards to input estimation, on the other hand, there is a reduction in fluctuations around the mean value after the beam release if only strains are inserted in the observations set figure 5.20. However, more time is needed for the input estimation to converge to a static value before the beam release when strains only are adopted. This can be seen also from SE and SD comparison provided in the table below.

	Strains + accelerometers	Strains only
SE	0.1266	0.1170
SD	0.1033	0.0785

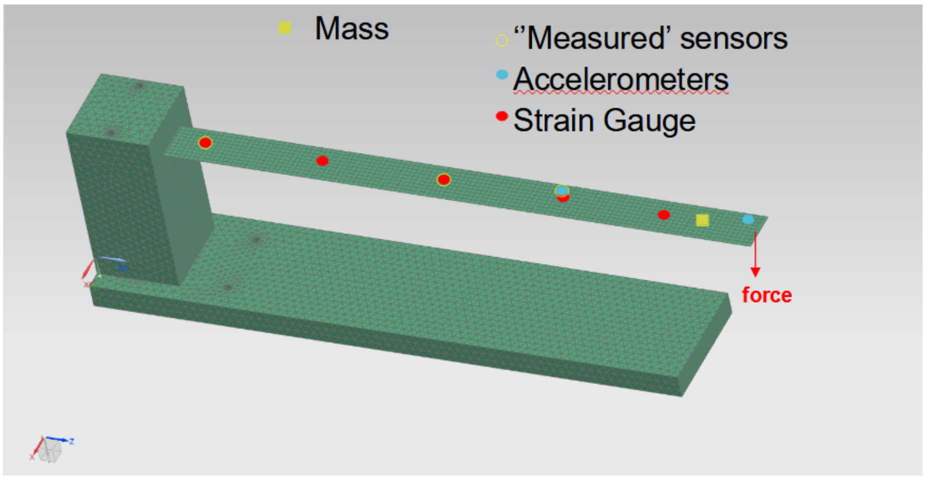


Fig. 5.6: "Measured/unmeasured" locations (both strains and accelerations) and input location and direction during the pull&release test

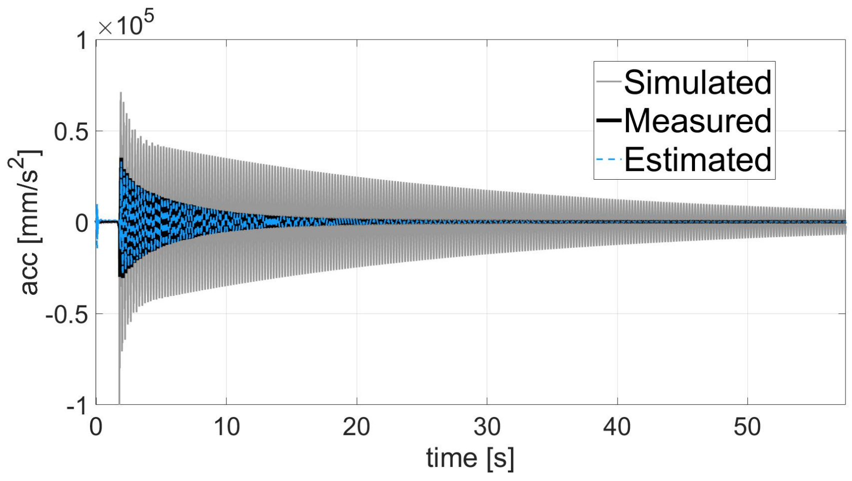


Fig. 5.7: Comparison between measured (black), simulated (grey) and estimated (light blue) time history for the second accelerometer, considering both acceleration and stains in the filter.

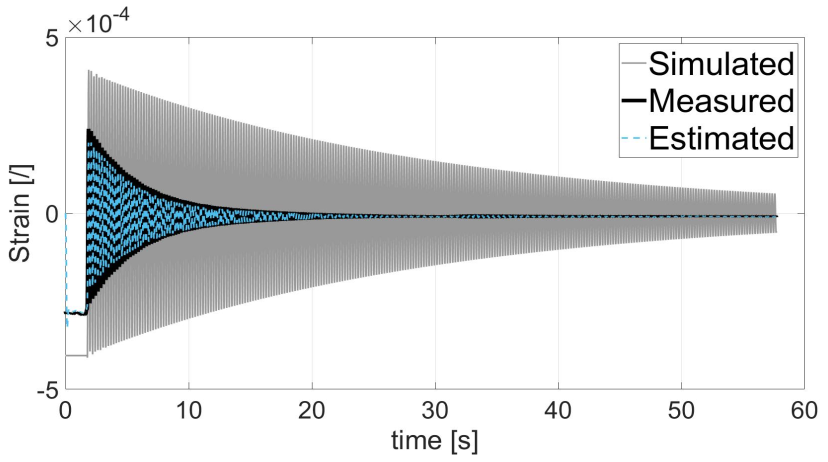


Fig. 5.8: Comparison between measured (black), simulated (grey) and estimated (light blue) time history for the second strain gage, considering both acceleration and stains in the filter.

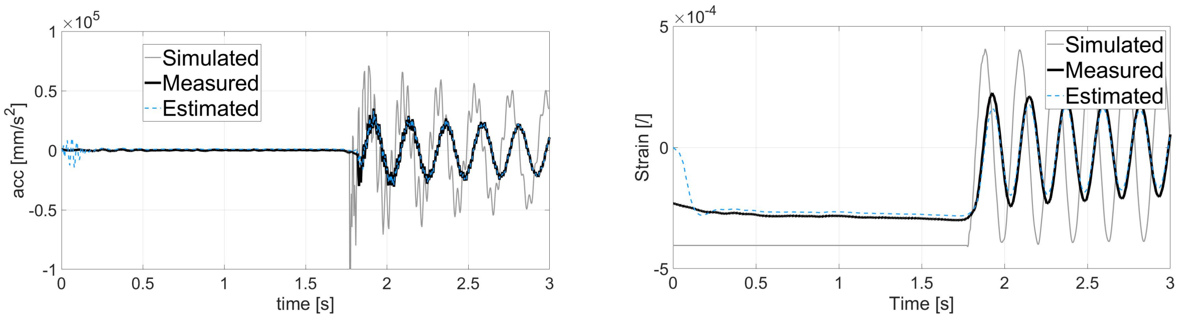


Fig. 5.9: Detailed view: time history of the second strain (left) and time history of the second accelerometer (right).

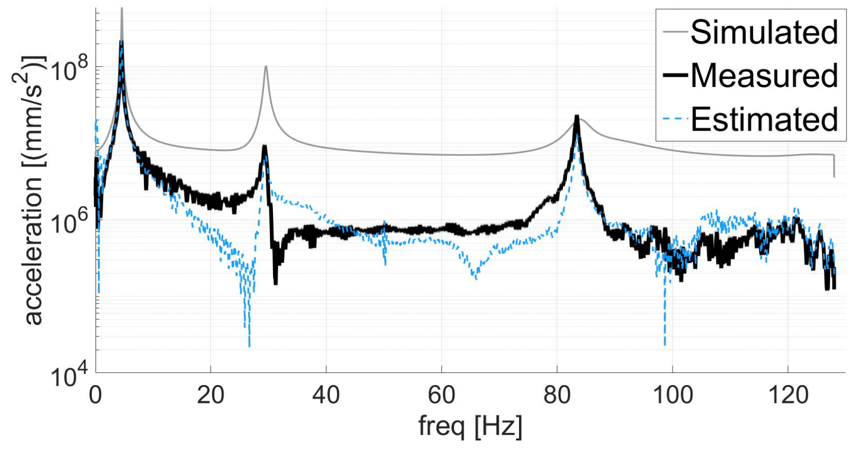


Fig. 5.10: FFT comparison for the second accelerometer, considering both acceleration and stains in the filter.

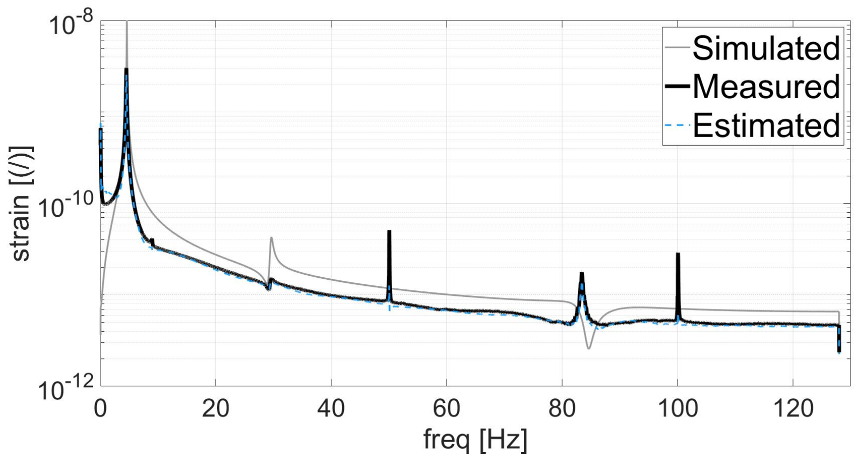


Fig. 5.11: FFT comparison for the second strain, considering both acceleration and stains in the filter.

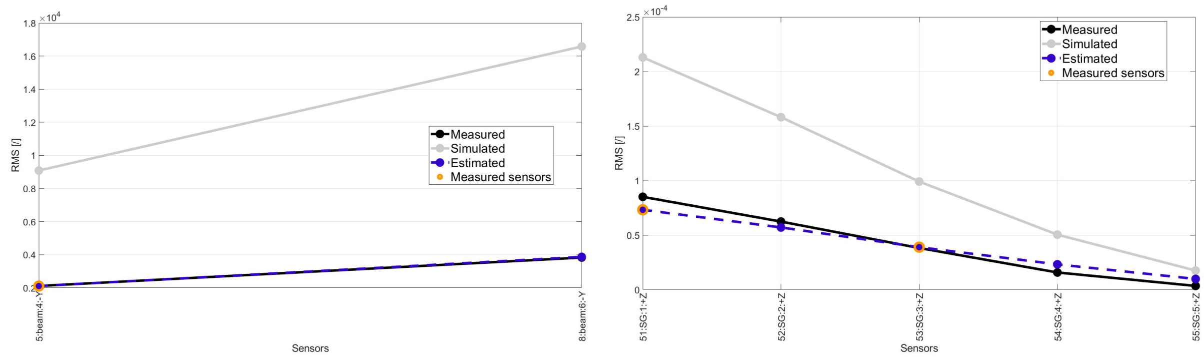


Fig. 5.12: Accelerations (left) and strains (right) measured (black), simulated (grey) and estimated (blue) time histories RMS comparison, considering both accelerometers and strains.

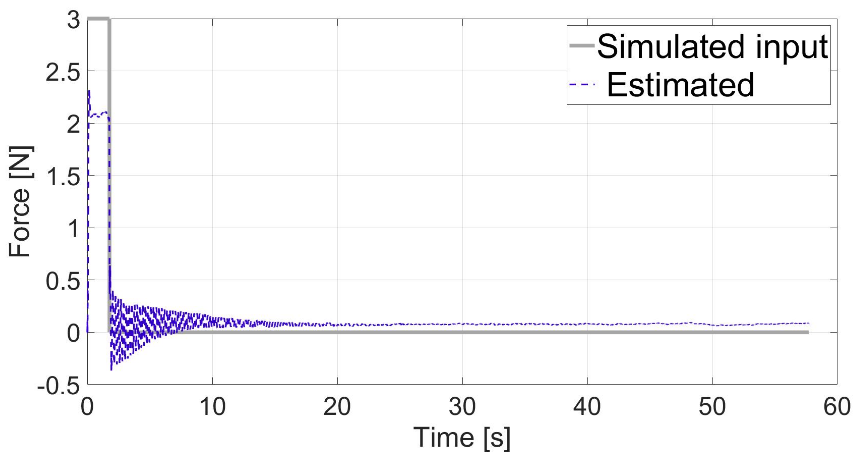


Fig. 5.13: Simulated (grey) and estimated (blue) time history, considering both accelerometers and strains.

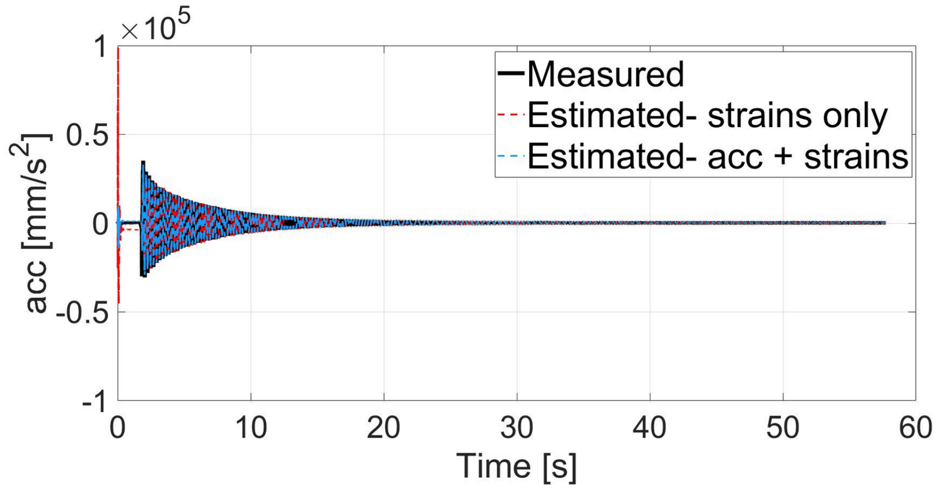


Fig. 5.14: Measured (black), estimated - strains only (red) and estimated - accelerations and strains (light blue) time histories for the second strain sensor.

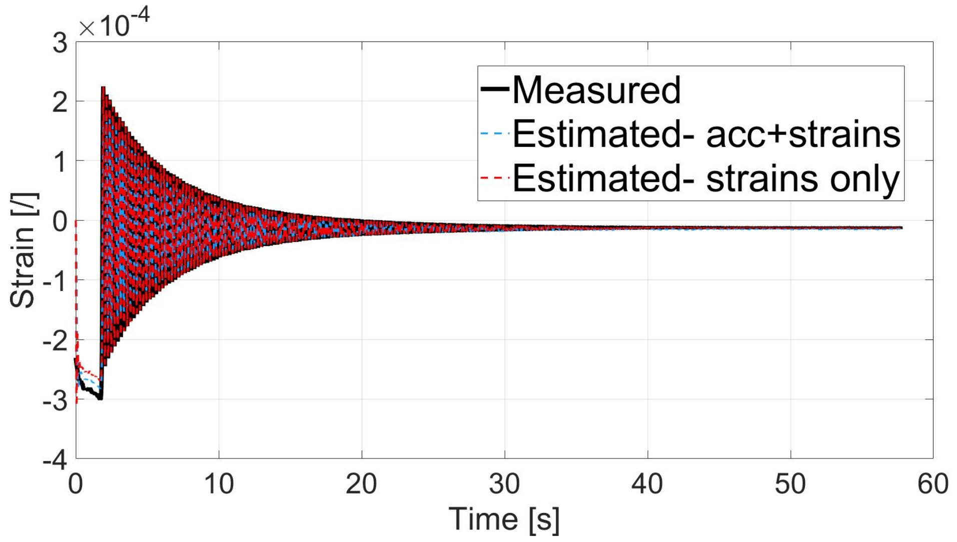


Fig. 5.15: Measured (black), estimated - strains only (red) and estimated - accelerations and strains (light blue) time histories for the second acceleration sensor.

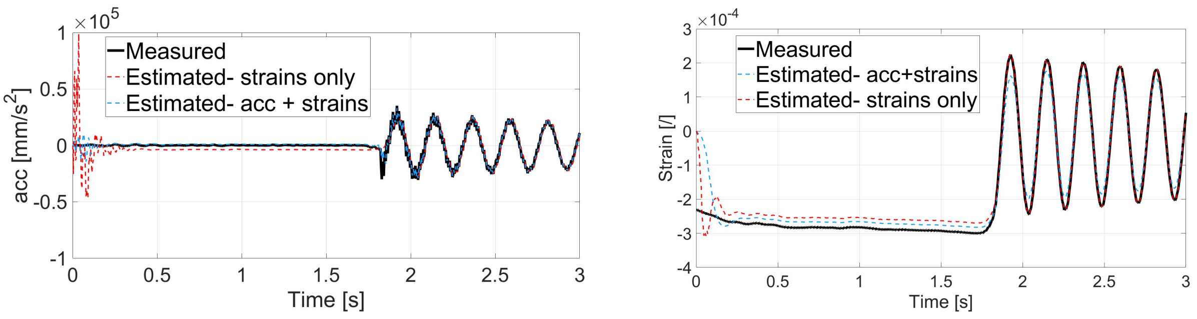


Fig. 5.16: detailed view of the previous graphics .

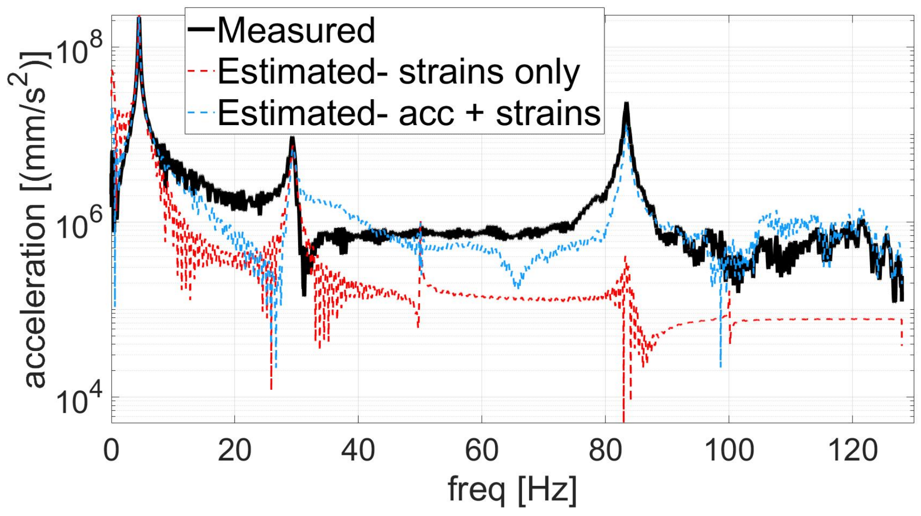


Fig. 5.17: FFT comparison for the second accelerometer, considering both acceleration plus strains and stains only in the filter.

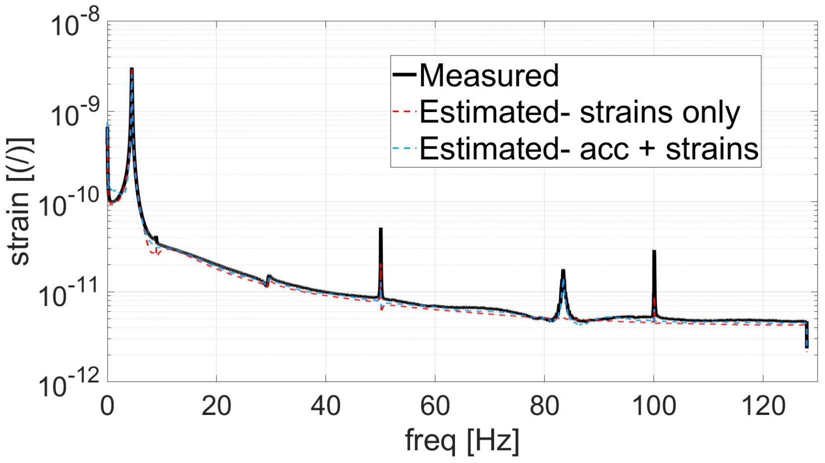


Fig. 5.18: FFT comparison for the second stains, considering both acceleration plus strains and stains only in the filter.

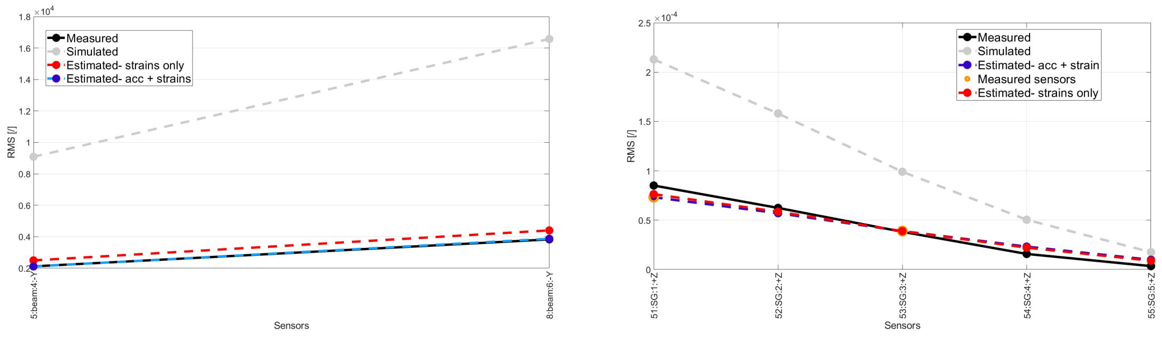


Fig. 5.19: Accelerations (left) and strains (right) measured (black), simulated (grey) and estimated time histories with strains only (red) and accelerations + strains (light blue) RMS comparison.

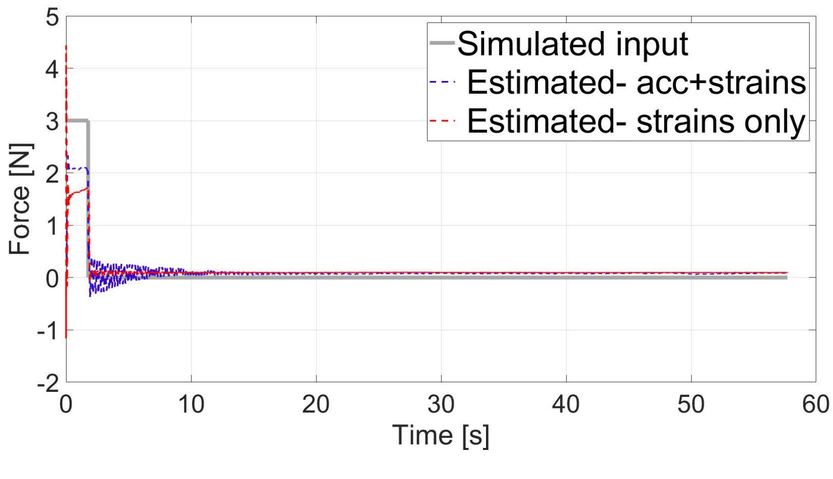


Fig. 5.20: Simulated (grey), estimated- strains only (red) and estimated - accelerations and strains (light blue) input time histories.

6. CONCLUSION

This work has successfully accomplished the objective of developing an automated procedure for automated damage simulation on a beam mockup model. Moreover, it has demonstrated that the full-field response and unknown inputs can be estimated via the AKF during pull&release tests on the mockup.

In particular:

- A flexible way of generating simulated datasets for damaged configurations of the beam mockup has been implemented in Python;
- Simulated data sets are obtained via the Pyhton script;
- Experimental data sets from pull&release tests on the beam mockup have been enriched by using the AKF for input-state-response estimation.

As future steps the author proposes the implementation of a code for simulating other types of defects, such as delaminations in composite materials. Moreover, an optimization of the process noise covariance matrix for improving the AKF predictions could be attempted. Finally, the implemented procedure to models of wind turbine blades in scale or not.

BIBLIOGRAPHY

[1] Wikipedia Web-Site *https://en.wikipedia.org/wiki/Machine_learning*

[2] Dan Simon. *Embedded Systems Programming.* Kalman Filtering, June 2001.

[3] Dan Simon. *Optimal State Estimation Kalman, H_∞, and Nonlinear Approaches.* JOHN WILEY & SONS, INC., PUBLICATION, 206.

[4] E. Lourens, E. Reynders, G. De Roeck, G. Degrande, G. Lombaert. *An augmented Kalman filter for force identification in structural dynamics.* Department of Civil Engineering, K.U. Leuven, Kasteelpark Arenberg 40, B-3001 Leuven, Belgium.

[5] M. Aucejo, O. De Smet, J.-F. Deu. *Practical issues on the applicability of Kalman filtering for reconstructing mechanical sources in structural dynamics.* Journal of Sound and Vibration 442 (2019) 45–70 , ELSEVIERE

[6] S. Vettori, E. Di Lorenzo, B. Peeters, E. Chatzi. *A virtual sensing approach to operational modal analysis for wind turbine blades.* In Proceedings of ISMA2020 International Conference on Noise and Vibration Engineering, Leuven, Belgium, 2020.

[7] Kristof Maesrof, Dr. ir. G. Lombaert Prof, dr. ir. G. De Roeck. *Filtering techniques for force identification and response estimation in structural dynamics.* KU Leuven, April 2016.

[8] R. R. Craig Jr *A review of time-domain and frequency-domain component mode synthesis method.* 1985.