



Universidad de
Oviedo



POLYTECHNIC SCHOOL OF ENGINEERING OF GIJÓN

**BACHELOR'S DEGREE IN INDUSTRIAL ELECTRONICS AND AUTOMATION
ENGINEERING**

**DEPARTMENT OF ELECTRICAL, ELECTRONICAL, COMPUTER AND SYSTEMS
ENGINEERING**

UNIVERSITY OF OVIEDO

BACHELOR'S THESIS NO. 19010071

**DESIGN OF A VIBRATION DATA ACQUISITION SYSTEM FOR A STRUCTURAL
HEALTH MONITORING TEST BENCH**

MR. ARNAIZ BURGUEÑO, ROBERTO

TUTOR: DR. DÍAZ BLANCO, IGNACIO

COTUTOR: MR. GARCÍA PÉREZ, DIEGO

JUNE 2019

Abstract

Structural Health Monitoring (SHM) is a methodology in development that aims to achieve the autonomous supervision of engineering structures. In this project, which is meant to be a contribution for the implementation of this methodology in industrial processes, a whole SHM test bench has been designed. This test bench consists of a steel bar, which is the structure under study, fixed to a metallic support, an excitation system that provokes vibrations on the bar and a data acquisition system that captures the induced vibrations by means of three accelerometers and processes and organizes the data to store it next. All this process has been automated with the help of the NI USB 6356 data acquisition board and Matlab scripts. In addition, Intelligent Data Analysis (IDA) applied to fault detection and diagnosis has been used to analyze the acquired data in order to make statements about the structural health of the studied system by means of Machine Learning algorithms.

Keywords

Keywords: Structural Health Monitoring, Vibrations, Data Acquisition, Frequency analysis, Intelligent Data Analysis, Machine Learning

Table of contents

	Page
List of Tables	vii
List of Figures	ix
1 Introduction	1
1.1 Motivation	4
1.2 Objectives	5
2 Methods	7
2.1 Structural Health Monitoring principles	9
2.1.1 Evolution of SHM	9
2.1.2 Vibration-based SHM methods	10
2.2 Design of the test bench	11
2.2.1 Physical system	13
2.2.2 Excitation system	13
2.3 Data acquisition system	16
2.3.1 DAQ hardware	16
2.3.2 DAQ software	19
2.3.3 Design of the tests	25
2.4 Data pre-processing: feature extraction	29
2.4.1 Feature extraction: spectral analysis	29
2.4.2 Data visualization: Repeatability tests	32
2.5 Intelligent Data Analysis: Machine Learning algorithms	32
2.5.1 Introduction to Machine Learning	33
2.5.2 Supervised Learning	34
2.5.3 Unsupervised Learning	36
3 Results	39
3.1 Data acquisition	39
3.2 Feature extraction by means of spectral analysis: observations	40

3.3	Outcome of the repeatability tests	41
3.4	Noise	43
3.5	Machine Learning algorithms outcome	45
3.5.1	Classification	46
3.5.2	2D projections	51
4	Discussion	59
4.1	Contributions	59
4.2	Assessment	60
4.3	Future work	61
A	Confusion matrices	63
A.1	Confusion matrices: Normal conditions, loose screws and added weights	63
A.2	Confusion matrices: Different joining plates	68
B	Timeline of work	69
C	Abbreviations	71
D	Matlab scripts	73
D.1	DAQ scripts	73
D.1.1	DAQ main	73
D.1.2	Initializing the experiment	74
D.1.3	Control signal	75
D.1.4	Data generation and acquisition	75
D.1.5	Data pre-processing	76
D.1.6	Data description file	76
D.1.7	Data storage	77
D.1.8	Plotting the acquisitions	77
D.2	Feature extraction scripts	78
D.2.1	Feature extraction main and repeatability visualization	79
D.2.2	Feature extraction function	81
D.3	Machine Learning scripts	82
D.3.1	Common functions	82
D.3.2	Logistic Regression scripts	84
D.3.3	Neural Networks scripts	88
D.3.4	PCA and T-SNE scripts	93
	Bibliography	95

List of Tables

TABLE	Page
2.1 Definition of failure classes of the structure.	26
A.1 Confusion matrix of logistic regression in <i>case 1</i> . LR algorithm trained with data from day one and tested with data from that same day. Training set 20% and test set 80% of the total amount of data. Accuracy of 82.12%	63
A.2 Confusion matrix of neural networks in <i>case 1</i> . NN algorithm trained with data from day one and tested with data from that same day. Training set 20% and test set 80% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of 89.37%	64
A.3 Confusion matrix of logistic regression in <i>case 2</i> . LR algorithm trained with data from day one and tested with data from day 2. Training set 20% and test set 80% of the total amount of data. Accuracy of 56.48%	64
A.4 Confusion matrix of neural networks in <i>case 2</i> . NN algorithm trained with data from day one and tested with data from day 2. Training set 20% and test set 80% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of 60.19% . . .	65
A.5 Confusion matrix of logistic regression in <i>case 3</i> . LR algorithm trained with data from day one and tested with data from that same day. Training set 70% and test set 30% of the total amount of data. Accuracy of 98.61%	65
A.6 Confusion matrix of neural networks in <i>case 3</i> . NN algorithm trained with data from day one and tested with data from that same day. Training set 70% and test set 30% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of 100%	66
A.7 Confusion matrix of logistic regression in <i>case 4</i> . LR algorithm trained with data from day one and tested with data from day 2. Training set 70% and test set 30% of the total amount of data. Accuracy of 81.48%	66
A.8 Confusion matrix of neural networks in <i>case 4</i> . NN algorithm trained with data from day one and tested with data from day 2. Training set 70% and test set 30% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of 83.95% . . .	67
A.9 Confusion matrix of logistic regression in <i>case 5</i> . LR algorithm trained with shuffled data from both days and tested with data from both days too. Training set 70% and test set 30% of the total amount of data. Accuracy of 98.04%	67

A.10 Confusion matrix of neural networks in *case 5*. NN algorithm trained with shuffled data from both days and tested with data from both days too. Training set 70% and test set 30% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of **99.35%** 68

A.11 Confusion matrix of LR algorithm applied to distinguish between classes 1, 10 and 11. Training set 70% and test set 30% of the total amount of data. Accuracy of **100%** . . . 68

List of Figures

FIGURE	Page
1.1 Stages involved in the SHM process.	2
1.2 Industry 4.0 diagram.	3
2.1 Phases of the Current CRISP-DM Process Model for Data Mining.	8
2.2 Overview of the CRISP-DM tasks and their outputs.	8
2.3 Test bench.	12
2.4 Test bench distribution.	12
2.5 Hardware diagram.	13
2.6 Hammer with its 3D-printed cover in orange.	14
2.7 Driver MD-22 and its connections.	15
2.8 Data acquisition board NI-USB 6356.	16
2.9 Conexions in NI-USB 6356.	17
2.10 Wilcoxon Research accelerometer with its 3D-printed cover in orange.	18
2.11 Wilcoxon Research power unit for accelerometers.	19
2.12 Software diagram.	20
2.13 Control signal.	21
2.14 <i>Categories</i> stored in <i>description.mat</i>	23
2.15 Appearance of an <i>Exp_N.mat</i> file.	23
2.16 Group of five hits measured from the three accelerometers.	24
2.17 Zoom over the aquisitions.	24
2.18 Numbering convention used for setting the number of each screw.	25
2.19 Weight over the beam before performing a test labeled in the class no. 7.	27
2.20 Different joining plates used.	28
2.21 Representation of the frequency feature extraction process.	31
2.22 Stages of a Machine Learning project.	33
3.1 Welch's Power Spectral Density of diverse hits from different classes computed for the measurements from each one of the three accelerometers S1, S2 and S3.	40

3.2	Energy evolution over time of the nine chosen frequency bands for hits from classes 1 (red), 2 (blue) and 4 (green) measured from accelerometer number 3.	41
3.3	Repeatability check for several hits from clases 1 (red), 3 (green) and 7 (blue) measured from accelerometer number 1.	42
3.4	Repeatability test performed over 5 hits of each of the first 9 classes.	43
3.5	Time signal of accelerometers 1 and 2 with cover and accelerometer 3 without it. . . .	44
3.6	Spectrograms of accelerometers 1 and 2 with cover and accelerometer 3 without it. . .	45
3.7	Bar chart showing the percentage of accuracy of LR and NN in the five described cases. 48	
3.8	Indicators learnt from data coming from sensor 1. The evalutated frequencies in accelerometer 1 are, in ascending order: 80, 610, 710, 1100, 1590, 1730, 2110, 3330 and 3490 Hz.	49
3.9	Indicators learnt from data coming from sensor 2. The evalutated frequencies in accelerometer 2 are, in ascending order: 80, 360, 610, 1230, 1390, 1590, 2040, 3330 and 3675 Hz.	50
3.10	Indicators learnt from data coming from sensor 3. The evalutated frequencies in accelerometer 3 are, in ascending order: 80, 610, 710, 1390, 1570, 1970, 2030, 3330 and 3470 Hz.	50
3.11	Projected data from classes 1 to 9 in 2D by means of PCA. Training set of 70% of the data set and the remaining 30% used as test set.	52
3.12	Projected data from classes 1 to 9 in 2D by means of T-SNE.	53
3.13	Projected data from classes 1 ,10 and 11 in 2D by means of PCA. Training set of 70% of the data set and the remaining 30% used as test set.	55
3.14	Projected data from classes 1, 10 and 11 in 2D by means of T-SNE.	56
B.1	Project timeline.	69

CHAPTER
1

Introduction

The growing demand for the supervision of a large number of structures with many quality requirements, the need to minimize the economic cost and personnel needed to perform such supervision and the advancement of new technologies led to the birth of a technology that allows autonomy in the supervision of a structure: *Structural Health Monitoring (SHM)*.

SHM consists in the implementation of a damage detection and diagnosis system in engineering structures. This system has to periodically acquire measurements from the structure by means of sensors, extract damage-sensitive features from those measurements and perform a statistical analysis in order to determine the health state of the structure [17]. The main stages of SHM can be seen in figure 1.1.

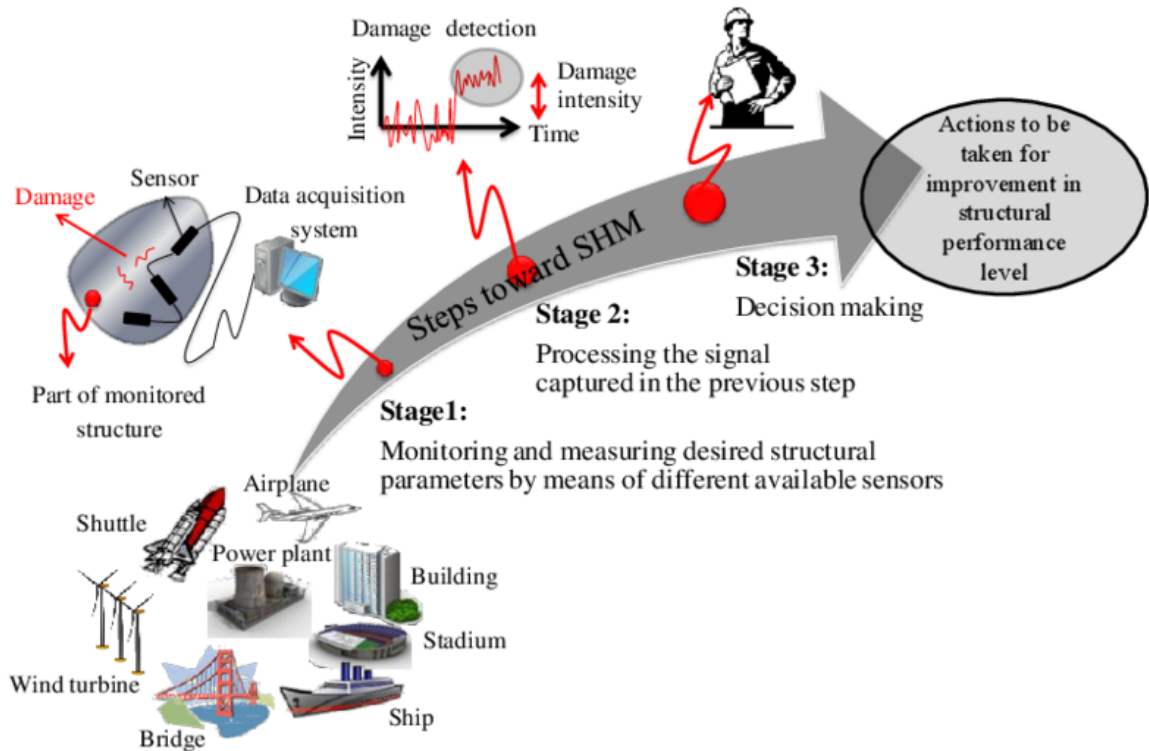


Figure 1.1: Stages involved in the SHM process.

Source: [25]

SHM systems can be analogously compared with the human nervous system, being sensors the equivalent of nerve endings, damage indication in a computer the equivalent of the pain indication in a human mind and decision making for fixing the damage, for instance, maintenance activities the equivalent to the decision taken by the human about going or not to the doctor [26].

SHM is a technology with an infinite number of applications and possibilities which today is more and more present in engineering companies. Some of its current applications are the monitoring of bridges, buildings, dams, tunnels, wind turbines, aerial and maritime transport and industrial facilities. SHM allows great savings in revision tasks, increases safety, reduces uncertainty and aids in planning and designing maintenance activities. The benefits that SHM is able to provide are making its market size grow significantly year by year. As [8] has pointed out: "The global structural health monitoring (SHM) market size was valued at USD 1220 million in 2017 and it is estimated to reach USD 4340 million by 2025, rising at a CAGR of 17.3% during the forecast period".

One of the areas where SHM can make a significant progress is *industrial processes*. SHM can facilitate the supervision of both industrial equipment and the product itself that is being built. This is the chosen field of work for this project, which aims to bring improvements to industrial

processes through SHM systems. An appropriate deployment of SHM systems and an integration of these systems into cloud data access services will be a contributing factor for the consolidation of *industry 4.0*, letting an industrial plant have an almost real time diagnostic of its monitored devices and have access to this information from any location. A diagram of the industry 4.0 model, where SHM plays an important role, is shown in figure 1.2.

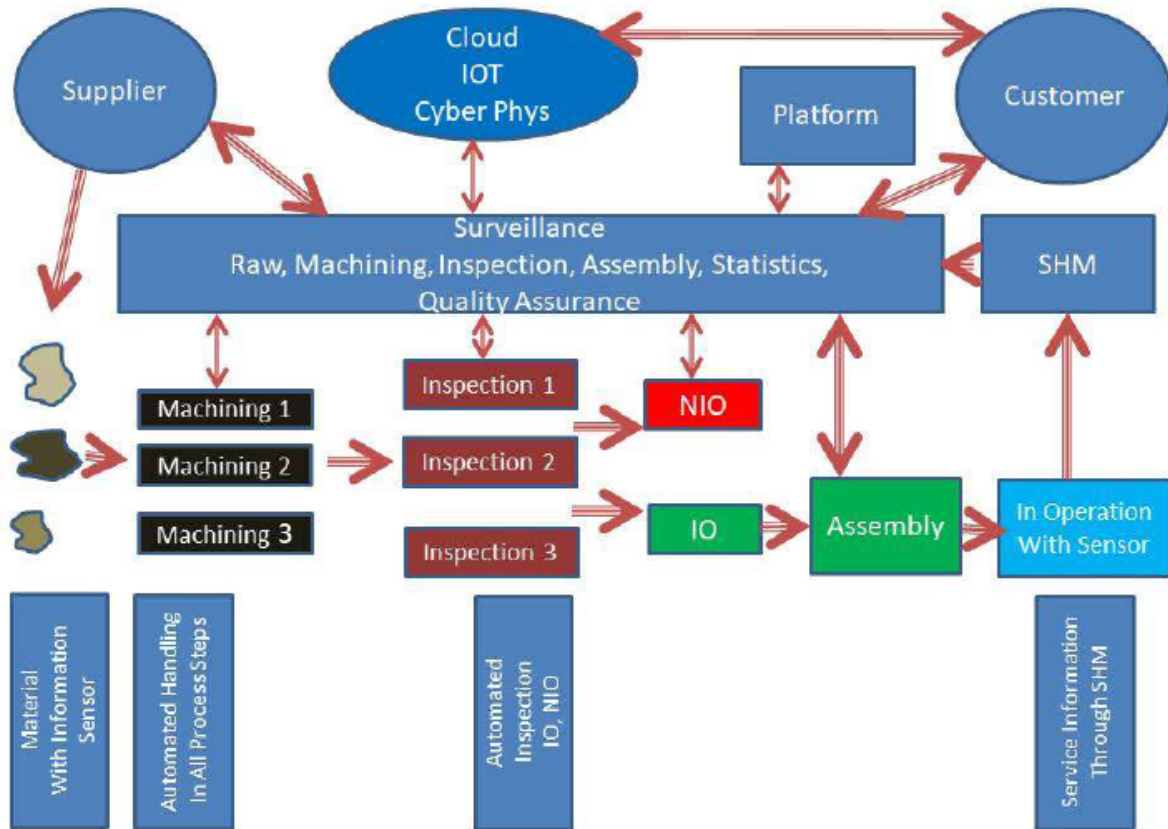


Figure 1.2: Industry 4.0 diagram.

Source: [31]

1.1.- Motivation

GSDPI research group [9] at the University of Oviedo has shown interest in SHM approaches based on machine learning (ML) and its applications to industrial processes over the last few years. The paper [24] written by *GSDPI*, whose main characteristics will be described in this section, can be considered as the antecedent of this project.

As it is shown in [24], the test bench initially proposed by *GSDPI*, which is based on the excitation of a steel bar on which faults can be induced and on the measurement and analysis of vibrations, allows flexibility in its configurations, achieves repeatability in tests and provides an easy access to the measured data. The test bench consisted of: a simple metal structure; an excitation system composed of an electromechanical actuator and H bridge motor driver controlled from an *arduino*; and a vibration signal acquisition system formed by two accelerometers, an oscilloscope receiving the signal from the accelerometers and a PC receiving the acquisitions from the oscilloscope via serial port in a Python environment where data was stored in *JSON* format. Finally, the results of the performed vibration analysis based on the feature extraction in the frequency domain were presented. These results were based on a Principal Component Analysis (PCA) where it was shown that PCA was able to cluster successfully data examples belonging to two different classes, normal conditions and failure (loose bolt).

The limits found in this approach were the following: overheating of the hammer caused by its inverse logic mode of operation, which may limit its prolonged use; the variability present in the acquired data due to the dependence on the environmental conditions, which could affect the repeatability of the tests.

As future work, it was proposed: to improve the existing instrumentation by means of a data acquisition board that could manage the acquisition of data and control the excitation system at the same time; a modular organization of the software that lets perform different groups of tests in an autonomous way and the standardization of the acquired data by adding metadata; the deployment of a remote service that lets the user perform tests through a web page, increasing the flexibility and the access to data.

The successful work done by *GSDPI* and the motivation to go further in this promising area of study, encouraged the proposal of this thesis, which will focus on enhancing the configurable SHM test bench created in 2017. The aim will be to improve the acquisition system and the automation of the tests, as well as the way in which data are stored. Experiments, in which different ways of inducing failures and extracting meaningful features are tested, will also be explored. Finally, different techniques of *Intelligent Data Analysis (IDA)* will be tested, mainly *Machine Learning algorithms* and *visualization techniques*, in order to draw conclusions about the *health state* of

the bar. Special attention will be paid to the following drawbacks found in the state of the art:

- Need for a high degree of automation of the tests so that the measurements do not depend on human errors and thus achieve a greater repeatability.
- High dependence on the environmental conditions of the effectiveness of the used algorithms.
- High dependence of the results on the feature extraction process.

To conclude this section, it has to be mentioned that all the stages present in this project will be addressed following the *CRISP-DM methodology*, which proposes a cyclic view of data mining projects.

1.2.- Objectives

Once the context and the motivation to carry out this thesis have been introduced, the specific objectives of this project will be enumerated:

- In order to improve the existing instrumentation of the SHM test bench, this project proposes the development of a modular software for handling a new high performance NI-USB-6356 board, using high level software (Matlab). The software must manage the automation of the tests, allowing to configure them as a series of impacts by periodically sending signals to the excitation device and measuring the generated vibrations with the help of three accelerometers. The software developed must use the libraries provided by the manufacturer (National Instruments).
- Wrapping the acquisitions in an organized data structure and storage in a standard file (.csv, .mat, .json).
- Test data analysis algorithms (Machine Learning) aimed at monitoring the structural condition and diagnosis.
- As a complement, new improvements of the bench will be explored, such as the use of 3D printing technologies to introduce possible enhancements in the mechanical configuration of the bench and in the design of the tests, through the incorporation of 3D printed parts as parameterizable test elements.

CHAPTER



Methods

The development of this project was not straightforward. It was made following a methodology called *CRISP-DM* (Cross Industry Standard Process for Data Mining) [18] which is based on a cyclic view of the development of a data mining project. The phases of a *CRISP-DM* data mining project and the tasks to carry out in each phase can be seen in figures 2.1 and 2.2, respectively.

As Berry and Linoff (1997) said, "data mining is not finished once a solution is deployed. The lessons learned during the process and from the deployed solution can trigger new, often more focused business questions. Subsequent data mining processes will benefit from the experiences of previous ones" [19].

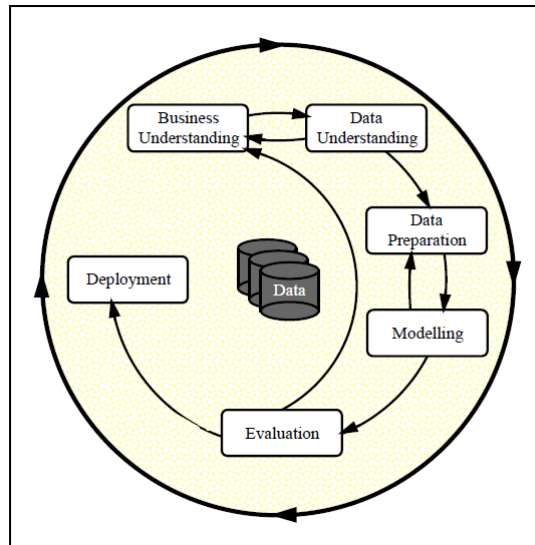


Figure 2.1: Phases of the Current CRISP-DM Process Model for Data Mining.

Source: [18]

Business Understanding	Data Understanding	Data Preparation	Modeling	Evaluation	Deployment
Determine Business Objectives <i>Background Business Objectives Business Success Criteria</i>	Collect Initial Data <i>Initial Data Collection Report</i>	<i>Data Set Data Set Description</i>	Select Modeling Technique <i>Modeling Technique Modeling Assumptions</i>	Evaluate Results <i>Assessment of Data Mining Results w.r.t. Business Success Criteria</i>	Plan Deployment <i>Deployment Plan</i>
Assess Situation <i>Inventory of Resources Requirements, Assumptions, and Constraints Risks and Contingencies Terminology Costs and Benefits</i>	Describe Data <i>Data Description Report</i>	Select Data <i>Rationale for Inclusion/Exclusion</i>	Generate Test Design <i>Test Design</i>	Approved Models	Plan Monitoring and Maintenance <i>Monitoring and Maintenance Plan</i>
Determine Data Mining Goals <i>Data Mining Goals Data Mining Success Criteria</i>	Explore Data <i>Data Exploration Report</i>	Clean Data <i>Data Cleaning Report</i>	Build Model <i>Parameter Settings Models Model Description</i>	Review Process <i>Review of Process</i>	Produce Final Report <i>Final Report Final Presentation</i>
Produce Project Plan <i>Project Plan Initial Assessment of Tools and Techniques</i>	Verify Data Quality <i>Data Quality Report</i>	Construct Data <i>Derived Attributes Generated Records</i>	Assess Model <i>Model Assessment Revised Parameter Settings</i>	Determine Next Steps <i>List of Possible Actions Decision</i>	Review Project Experience <i>Documentation</i>
		Integrate Data <i>Merged Data</i>			
		Format Data <i>Reformatted Data</i>			

Figure 2.2: Overview of the CRISP-DM tasks and their outputs.

Source: [18]

Throughout this chapter, the different techniques that have been taken into account when developing this project will be presented. These techniques have been broken down as follows: SHM principles; design of the test bench; data acquisition system; data pre-processing: feature extraction; Intelligent Data Analysis: Machine Learning algorithms.

2.1.- Structural Health Monitoring principles

The purpose of Structural Health Monitoring systems is to track the performance of mechanical systems (machines, aircrafts, large structures, etc.) and detect divergences from its normal working conditions. These divergences are referred to as damages, which can be either classified as defects, if they only affect the efficiency of the system or failures, if these damages compromise the correct functioning of the system, being no longer suitable for the user [24].

SHM techniques aim to detect both failures and defects in mechanical systems providing, in addition, characteristic information about them. According to [28], one of the ways to classify SHM techniques is according to the quantity of information with which defects are characterized:

- Level 1: they determine the existence of a defect.
- Level 2: they determine the location of the defect.
- Level 3: they quantify the severity of the damage.
- Level 4: they predict the lifetime of the system.

The purpose of this section is to introduce the reader to SHM methodologies and it has been broken down in the following way: evolution of SHM and vibration-based SHM methods.

2.1.1.- Evolution of SHM

Non-Destructive Testing methods (NDT) and visual inspection techniques are considered the background of modern SHM systems. The main advantage of SHM over NTD is the deployment of integrated systems that work online, what makes SHM systems autonomous. In the near future, NDT methods may be replaced by on-line SHM systems, in which sensors are permanently mounted on the structure. This autonomous inspection approach leads to the development of smart structures.

In order to talk about the future of SHM, this vision about the future development of SHM in aeronautics by the Aerospace Industry Steering Committee on Structural Health Monitoring (AISC-SHM) appears to be a good case study. According to [10], AISC-SHM defines these four future phases:

- Phase I (current situation): stand-alone systems for advanced maintenance at manned flights.
- Phase II (short term): evolution towards more automated SHM-support systems for manned and unmanned flights.
- Phase III (medium term): evolution towards fully automated systems, can also be used for remote control.
- Phase IV (long term): evolution where SHM is a design factor that is integrated into all critical elements.

2.1.2.- Vibration-based SHM methods

One of the most common methods in the state of the art of fault detection is the vibration analysis of the system, known as vibration-based methods [32]. These kind of techniques are based on the modal analysis of vibration signals acquired from the monitored structure. Vibration analysis is a consolidated method in the supervision of rotating machines, also referred to as condition monitoring (CM) by [29, 30]. CM is a field of study analogous to SHM and hereby, it is appropriate to make use of similar vibration-based techniques in both fields of study, but they differ in the fact that vibration data acquired from rotatory machines is usually more invariant to environmental conditions than measurements coming from large structures, which increases the challenge faced in SHM [24].

The most widely used paradigm in vibration-based methods is the *statistical pattern recognition*, which is a *knowledge-based methods* and, unlike analytic methods, lets one reach solutions even without mastering the physical phenomena beneath the studied process. According to [33], statistical pattern recognition is divided in four tasks:

- Operational evaluation.
- Data acquisition and normalization.
- Feature extraction.
- Model development.

Every ambit defined by [33], except operational evaluation, is strongly related with Intelligent Data Analysis (IDA). A lot of the techniques proposed in the IDA field, as *Machine Learning algorithms*, can be transferred to SHM with a remarkable potential contribution, such as helping to solve the problem of the variability of environmental conditions with an effective feature extraction.

In order to initiate a line of study in SHM, a baseline data repository is clearly necessary. This data can be obtained from public SHM repositories or collected by oneself from real working systems or from test benches that simulate reality. The availability of an already working test bench in GSDPI's laboratory made the latter choice direct. Having access to an own test bench is a great advantage because it allows one to adapt how experiments are performed to the objectives of the research.

2.2.- Design of the test bench

A compound steel bar will be used as structure of study for researching about the possibilities of SHM applied to industrial and manufacturing processes because it is a relatively simple structure where it is easy to introduce mechanical damages. But it must be borne in mind that the line of work in which this project aims to contribute is the development of fault detection techniques applicable to any type of structure or material. The final objective of this line of research is to make the technique independent of the material or shape of the structure studied thanks to IDA.

In this section, which composes the operational evaluation mentioned in 2.1.2, it will be explained how the *physical system* under study, the *excitation system* and the procedure for the tests have been designed. The appearance of the bench, its overall distribution and its hardware diagram can be seen in figures 2.3, 2.4 and 2.5, respectively. The hardware diagram includes the physical system in red, the excitation system in green and the data acquisition system in blue, which will be described in section 2.3.

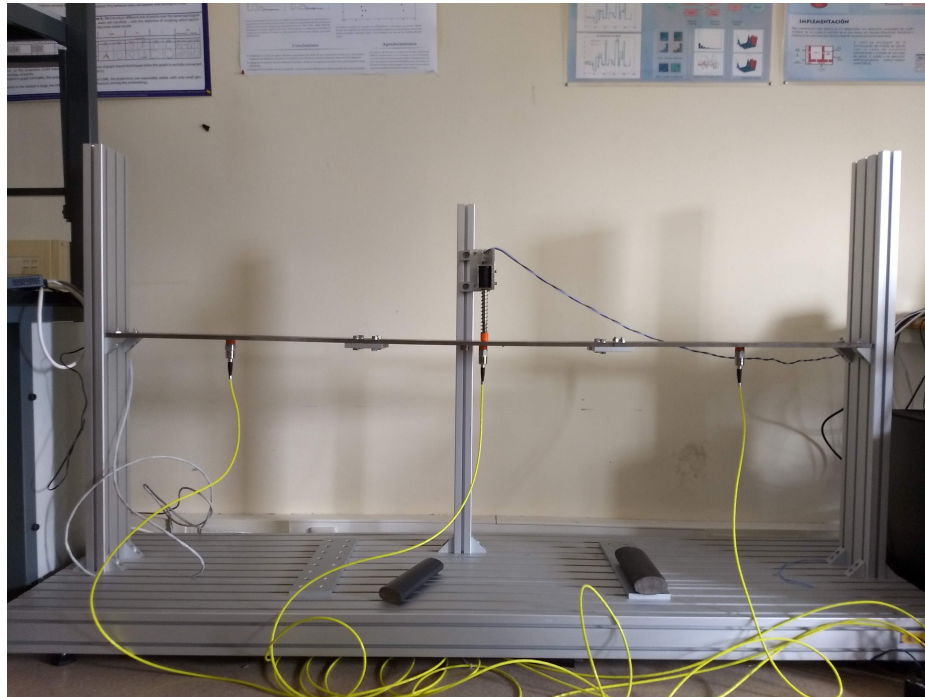


Figure 2.3: Test bench.

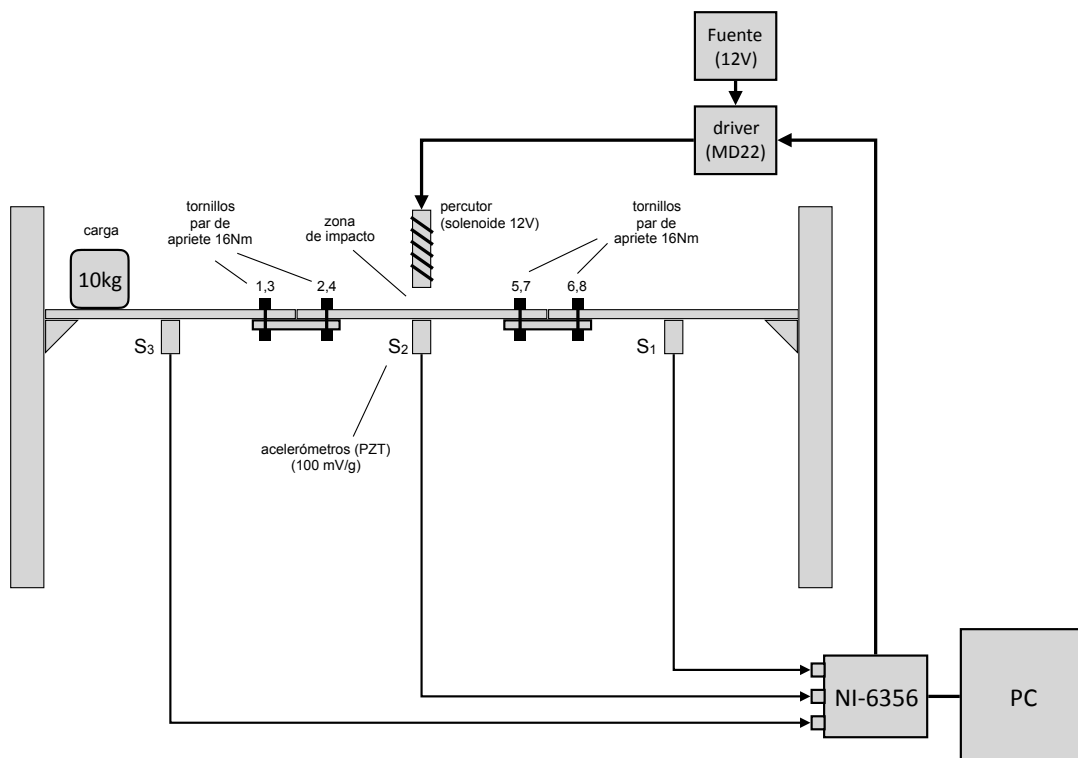


Figure 2.4: Test bench distribution.

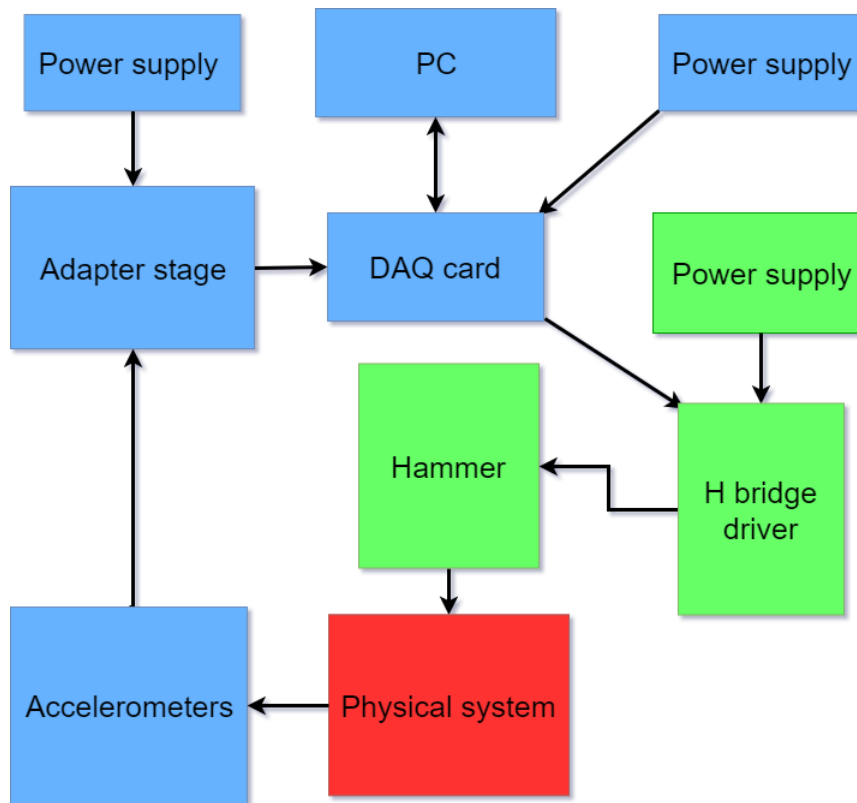


Figure 2.5: Hardware diagram.

2.2.1.- Physical system

A flat compound steel bar supported by a metallic structure composes the physical system under study. This bar is formed by three sections joined by four metallic plates screwed with two screws each one. The support metallic structure consists of a horizontal base and two vertical metal profiles screwed to the base. These metal profiles offer support to the steel bar by means of two support brackets screwed to both the profiles and the bar. This configuration is shown in figure 2.3.

2.2.2.- Excitation system

Although in some SHM tests, usually in big scale tests, it is enough with natural or human excitations to observe the dynamic behaviour of a system, in this case it will be needed to induce an external excitation in the system to be able to observe its dynamic response. An *electromechanical actuator* switched by a *H bridge preactuator* have been used for this purpose.

2.2.2.1.- Electromechanical actuator

The *Ralux electromechanical actuator* or *hammer* is attached to the support structure being fixed to a third vertical metal profile screwed in the center of the horizontal base. It is composed by a metallic axis with a nut screwed at one of its ends, a spring that surrounds the axis, a coil of 24V and 1A where the axis is introduced and a case that holds the coil. Its operating principle is as follows: when the coil receives current because 24V are applied through a bifilar wire, the magnetic field generated attracts the metal axis upwards to its nucleus and when the coil stops receiving current because 0V are applied, the magnetic field decays and the metallic axis is moved downwards due to the potential elastic energy stored by the spring, what generates an impact on the steel bar. The nut works as a top for the spring not to go out.

As the magnetic field generated by the coil has influence on its surroundings, it doesn't only attract the metal axis but sometimes also the steel bar under study, which can ruin completely the performance of the tests. A solution for this problem was found using a 3D-printed plastic cover, which isolates the metal axis of the hammer, as it can be seen in figure 2.6.



Figure 2.6: Hammer with its 3D-printed cover in orange.

2.2.2.2.- Preactor

The *preactor* is an *H bridge motor driver*, specifically a *Devantech MD22* [14], designed for controlling two DC motors but it is also compatible with the control of the coil inside the hammer. In this case it receives a control signal from the *data acquisition board*, which will be described later in section 2.3.1, and outputs a voltage signal adapted to the working conditions of the hammer. The mode selector is put in the *1100* position to select the mode *OV – 2.5V – 5V analog inputs* which permits an all/nothing control. It receives a signal between 2.5V and 5V DC from the data acquisition board between pins *GND* and *SDA*, 5V DC signal in the pin *+5* to feed the microcontroller inside the driver and 24V DC from an external source between pins *V+* and *V-*. Then, it will generate an output signal between pins *M2+* and *M2-* which will alternate between 0V and 24V depending on whether the control signal value is 2.5V or 5V, respectively. This control signal will let the hammer work as desired when it is introduced a proper input signal, which will be described in the section 2.3.2.

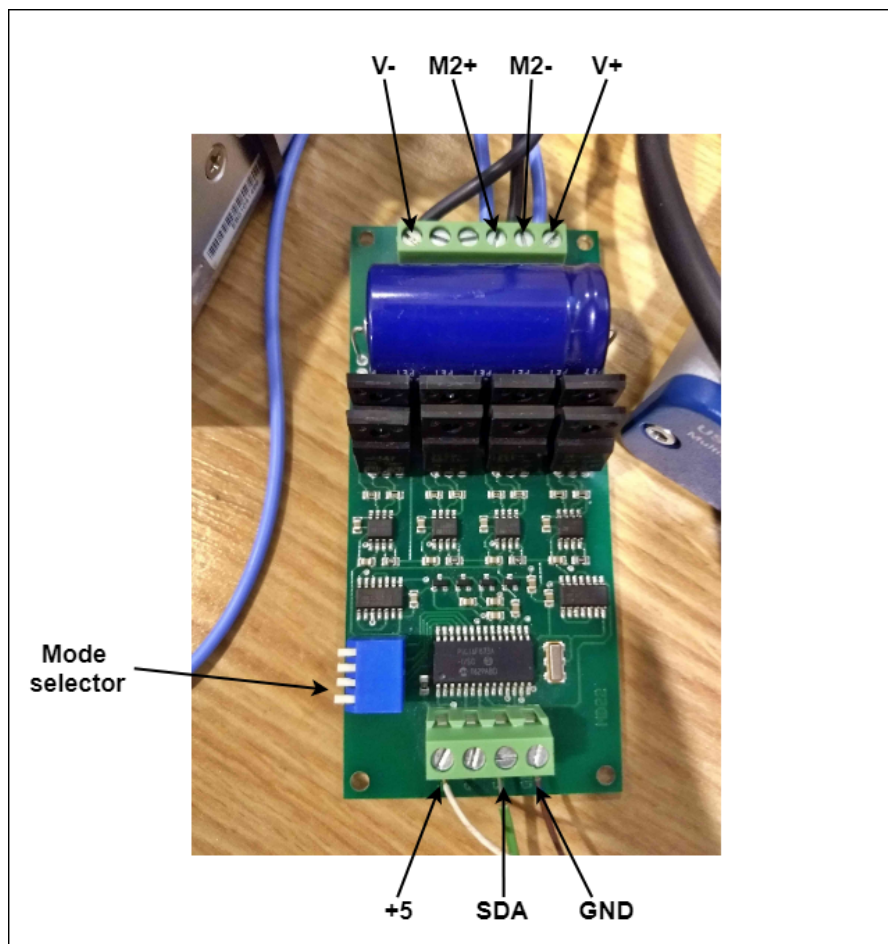


Figure 2.7: Driver MD-22 and its connections.

2.3.- Data acquisition system

In this section, it will be explained how the data acquisition system has been designed and how it is interconnected with the test bench so that the tests can be completely automated by means of a software written in Matlab.

2.3.1.- DAQ hardware

The data acquisition system is composed of three accelerometers that measure the vibration in the bar, a data acquisition board that samples the signals coming from the accelerometers and also generates the control signal for the *H bridge* of the hammer and a PC that manages the traffic of signals from the software Matlab.

2.3.1.1.- Data acquisition board

NI-USB 6356 data acquisition board [11] has been chosen for this test bench because of its high sample rate and high resolution, which are the most relevant features for vibration analysis. It is equipped with 8 simultaneous 16bit AD converters which can handle a maximum input sample rate of $1.25MS/s$ [13].

This board is fed with a DC power supply of $30W$ and it receives the measured signals by the accelerometers in the analog input pins *AI0+* and *AI0-*; *AI2+* and *AI2-*; and *AI5+* and *AI5-*. The output signals from this board are the control signal for the hammer going out from the analog output pins *AO0* and *AO GND* and a digital $5V$ continuous signal produced in pin no. 96 used to feed the microcontroller in the *H bridge*. Figure 2.9 shows this conexions. The analog input signals don't have to be ground-referenced since this board works in differential mode as it is described in the user manual [12]. The *NI-USB 6356* is also connected to the PC through an USB connector so it can exchange information with Matlab and vice versa.



Figure 2.8: Data acquisition board NI-USB 6356.

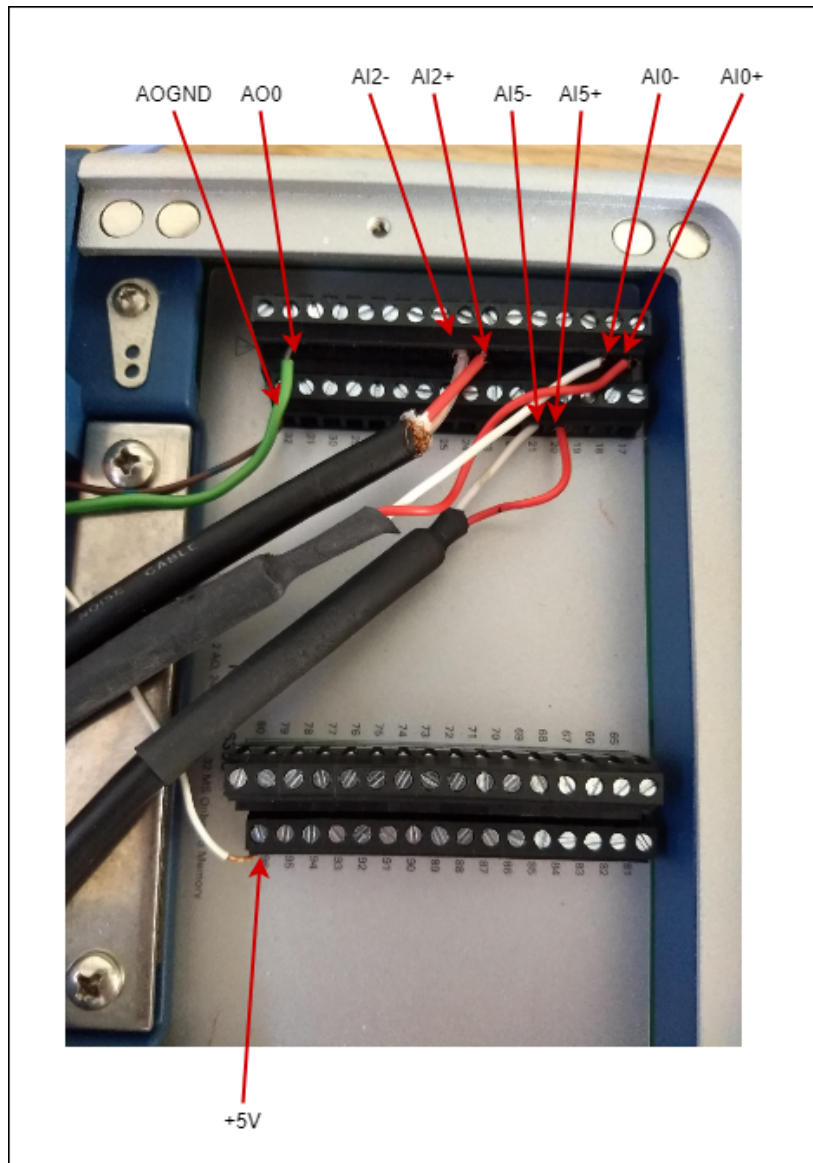


Figure 2.9: Conexions in NI-USB 6356.

2.3.1.2.- Accelerometers

Three Wilcoxon Research *piezoelectric accelerometers*, which are suitable for vibration analysis because of their wide frequency range, are located in the middle of each of the three sections that compose the steel bar under study. Piezoelectric accelerometers working principle consists in a seismic mass, that suffers the same acceleration as the structure it is measuring, and a piezoelectric material attached to it, which outputs a voltage between two electrodes proportional to the acceleration experienced by the mass. The accelerometers will measure vibration acceleration in the vertical direction, transforming it into a voltage signal with a sensitivity of $100mV/g$ [15]

and sending it to the data acquisition board through a supply stage of 27V DC and coaxial wires [16]. A procedure similar to the one used for isolating the hammer with a 3D-printed cover was used to isolate the three accelerometers, in this case because of the presence of noise. This fact will be better described in section 3.4. An accelerometer with the cover and the supply stage are shown in figures 2.10 and 2.11, respectively.



Figure 2.10: Wilcoxon Research accelerometer with its 3D-printed cover in orange.



Figure 2.11: Wilcoxon Research power unit for accelerometers.

2.3.2.- DAQ software

An important concept to understand this module of the system is the theoretical meaning under the *sampling process*, which yields a sequence of values from an analog signal [21].

$$(2.1) \quad x(t) \rightarrow x_k, t = kT_s, \forall k = \dots, -1, 0, 1, 2, \dots$$

$$(2.2) \quad f_s = \frac{1}{T_s}$$

where T_s is the sampling period and f_s is the sampling frequency.

According to the Nyquist-Shannon theorem, which says that $f_s \geq 2F_{max}$, being f_s the sampling frequency and F_{max} the maximum frequency of the signal that has to be measured, it will have to be checked that the chosen sampling frequency f_s is high enough in order not to lose important information in the high frequencies.

To automate the acquisitions, some *Matlab* scripts have been designed. They make a continuous use of the *Data Acquisition or DAQ Toolbox* [1]. It was decided to make the program as modular as possible by isolating every specific purpose in different functions in order to facilitate debugging processes and future modifications. Several functions are called sequentially from the main script so the tests can be performed automatically. The software diagram of the acquisition showing the most representative aspects of these scripts can be seen in figure 2.12, every function will be explained with more detail afterwards.

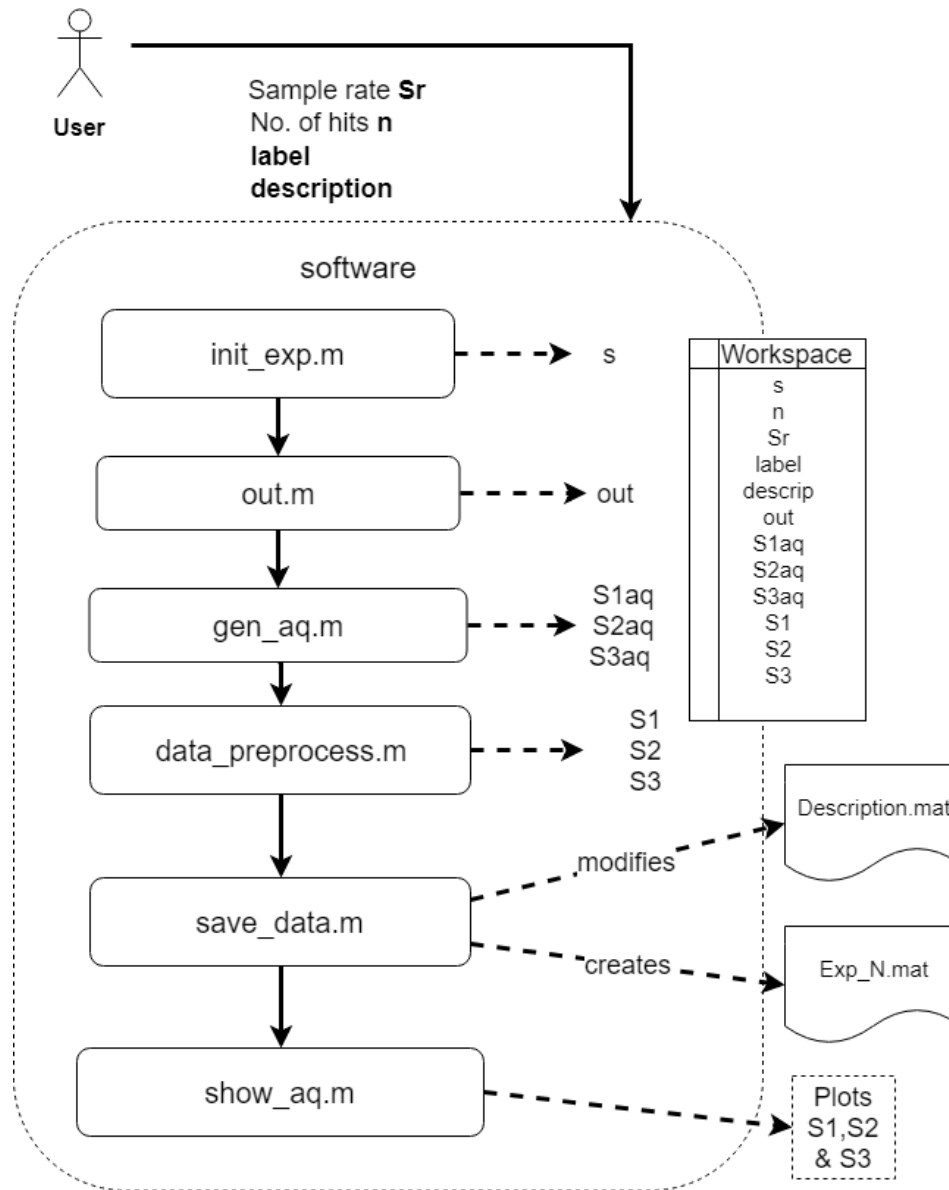


Figure 2.12: Software diagram.

Before every experiment, the program asks the user, by means of a dialog in the command line, for: the number of hits n to be performed; the sampling frequency f_s for the data acquisition, which is denoted as Sr (Sampling rate) in the scripts; a brief description of the test; and the label indicating the previously known condition of the bar.

Then, the functions that are shown in the next list of items, whose code is shown in D.1, are executed in order of appearance:

- `s = init_exp(Sr)`: outputs a *DAQ* session `s` with a given sample rate frequency `Sr` and configures pins `AI0`, `AI2` and `AI5` as analog inputs and `AO0` as an analog output pin.
- `out = control_signal(Sr)`: outputs the necessary signal to be sent to the *H bridge* to make the hammer hit once. This signal has a duration of 15 s and the hit will happen at the 5th second. This configuration has proven experimentally to be an effective compromise between attenuation of the oscillations in the bar before the next hit and fluency to perform big quantities of experiments. The signal returned by `control_signal` alternates between 5V and 2.5V, as it was said in 2.2.2, which are the necessary values to retract and release the hammer, respectively. The shape of the generated control signal can be seen in figure 2.13

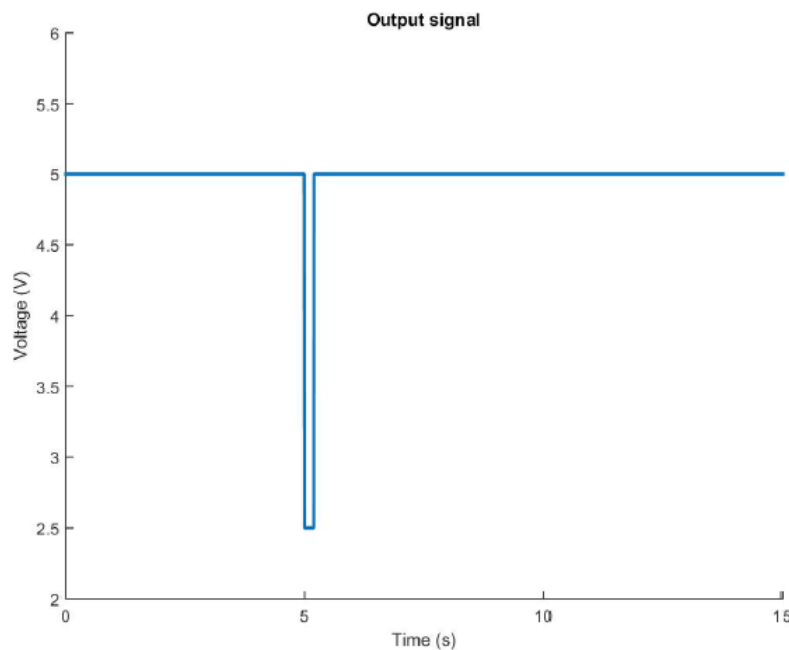


Figure 2.13: Control signal.

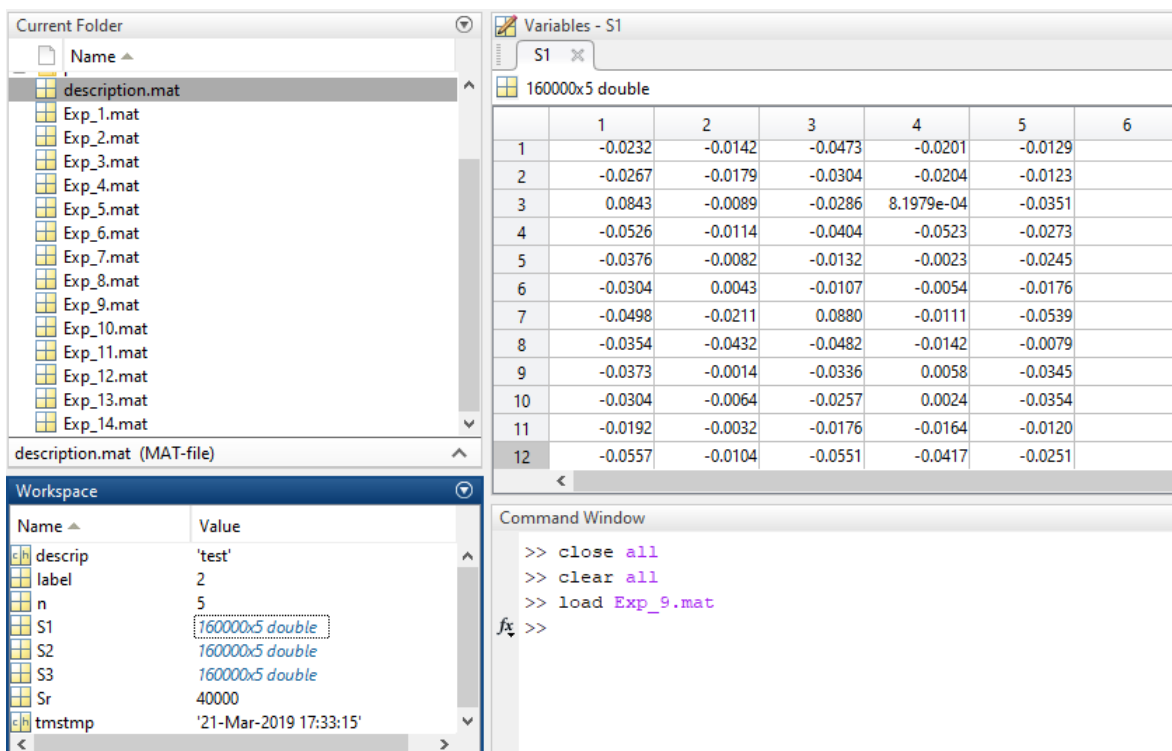
- `[S1aq, S2aq, S3aq] = gen_aq(n, Sr, out, s)`: this function runs `n` times a for loop that repeats the generation of the control signal and captures the signal coming from the accelerometers with the sample rate `Sr`. Each time the function `s.startForeground()` is executed, the control signal for a single hit is sent so the hammer hits the bar, the program samples the signals arriving to the configured analog inputs and, after 15 seconds, they are saved in the variable `data`. Then, the measurements are reshaped concatenating columns of acquisitions in three different matrices `S1aq`, `S2aq` and `S3aq`, one for each sensor, so they contain samples along rows and different hits along columns. After the last hit, the

control signal is set to 5V so the hammer stops consuming power while it waits for a new experiment to be performed and thus, its excessive overheating is avoided. Finally, the three matrices S1aq, S2aq and S3aq are output.

- `[S1, S2, S3] = data_preprocess(n, Sr, S1aq, S2aq, S3aq)`: acquired data S1aq, S2aq and S3aq are cut to reduce necessary memory and it is also synchronized to make easier the posterior processing and for a better visualization of the repeatability in the experiments. Concretely, it finds the first point that overpasses, in absolute value, a threshold of 0.5V in every different hit in the central accelerometer (no. 2) and saves data from 0.5 s before each hit until 3.5 s after it, so the total duration of the saved data is 4 seconds. This information is saved in three new variables S1, S2 and S3, which again contain samples along rows and different hits along columns.
- `data_descrip()`: this function creates a ".mat" file containing an array categories with the fields Number of exp, number of hits, Sample rate, Label, Description and timestamp where these characteristics about each test will be stored just after they are performed. The `save_data` function, which will be described next, will be in charge of saving this information together with the measurements. Every time that it is necessary to know information about experiments performed in the past, it can be checked in this *description.mat* file. Notice that it must be only executed once, at the beginning of all the experiments.
- `save_data(n, Sr, S1, S2, S3, label, descrip)`: this function creates, after each experiment finishes, a new ".mat" file in the same path as *description.mat* with the name *Exp_N.mat*, being N the number of the experiment in the history of the *description.mat* file. The capture matrices S1, S2 and S3, the number of hits n in the experiment, the sample rate Sr and the variable label will be stored in the *Exp_N.mat* file. `save_data()` also fills a row in the categories array of *description.mat* file with the characteristic data of the experiment. The appearance of the categories inside a *description.mat* file and the matrix S1 inside an *Exp_N.mat* are shown in figures 2.14 and 2.15, respectively. Note that the variables appearing in the workspace in figure 2.15 are the ones saved inside each *Exp_N.mat* file and that the files appearing in the top left of that same figure are the files generated after a group of 14 different experiments.

	1	2	3	4	5	6
1	'Number of Exp'	'number of hits'	'Sample Rate'	'Label'	'Description'	'Timestamp'
2	1	30	40000	1	'30 strikes in normal conditions'	'08-Mar-2019 10:40:27'
3	2	30	40000	2	'30 strikes with loose screw (1)'	'08-Mar-2019 10:52:27'
4	3	30	40000	3	'30 strikes with loose screw'	'08-Mar-2019 11:03:44'
5	4	30	40000	4	'30 strikes loose screw 5'	'08-Mar-2019 11:28:14'
6	5	30	40000	5	'30 strikes loose screw (8)'	'08-Mar-2019 11:56:12'
7	6	30	40000	6	'30 strikes with weight in the left side'	'08-Mar-2019 12:16:50'
8	7	30	40000	7	'30 strikes with a weight in the right side'	'08-Mar-2019 12:27:10'
9	8	5	40000	1	'test'	'21-Mar-2019 17:28:59'
10	9	5	40000	2	'test'	'21-Mar-2019 17:33:15'
11	10	5	40000	3	'test'	'21-Mar-2019 17:36:41'
12	11	5	40000	4	'test'	'21-Mar-2019 17:54:44'
13	12	5	40000	5	'test'	'21-Mar-2019 17:57:21'
14	13	5	40000	6	'test'	'21-Mar-2019 17:59:30'
15	14	5	40000	7	'test'	'21-Mar-2019 18:01:26'
16						

Figure 2.14: *Categories* stored in *description.mat*.



The image shows the MATLAB environment. On the left, the 'Current Folder' pane displays a directory structure with files like *description.mat*, *Exp_1.mat*, ..., *Exp_14.mat*. The 'Workspace' pane shows variables: *descrip* (value: 'test'), *label* (value: 2), *n* (value: 5), *S1* (value: 160000x5 double), *S2* (value: 160000x5 double), *S3* (value: 160000x5 double), *Sr* (value: 40000), and *tmstmp* (value: '21-Mar-2019 17:33:15'). The 'Command Window' shows the following commands:

```
>> close all
>> clear all
>> load Exp_9.mat
fx >>
```

The 'Variables - S1' pane shows a 12x6 matrix of double values:

	1	2	3	4	5	6
1	-0.0232	-0.0142	-0.0473	-0.0201	-0.0129	
2	-0.0267	-0.0179	-0.0304	-0.0204	-0.0123	
3	0.0843	-0.0089	-0.0286	8.1979e-04	-0.0351	
4	-0.0526	-0.0114	-0.0404	-0.0523	-0.0273	
5	-0.0376	-0.0082	-0.0132	-0.0023	-0.0245	
6	-0.0304	0.0043	-0.0107	-0.0054	-0.0176	
7	-0.0498	-0.0211	0.0880	-0.0111	-0.0539	
8	-0.0354	-0.0432	-0.0482	-0.0142	-0.0079	
9	-0.0373	-0.0014	-0.0336	0.0058	-0.0345	
10	-0.0304	-0.0064	-0.0257	0.0024	-0.0354	
11	-0.0192	-0.0032	-0.0176	-0.0164	-0.0120	
12	-0.0557	-0.0104	-0.0551	-0.0417	-0.0251	

Figure 2.15: Appearance of an *Exp_N.mat* file.

- `show_aq(n, Sr, S1, S2, S3)`: It shows three subplots with the n overlapped acquisitions of the three sensors with a proper time vector according to the sample rate Sr of the experiment. The shape of the acquired signals of five hits and a zoom over those signals can be seen, respectively, in figures 2.16 and 2.17.

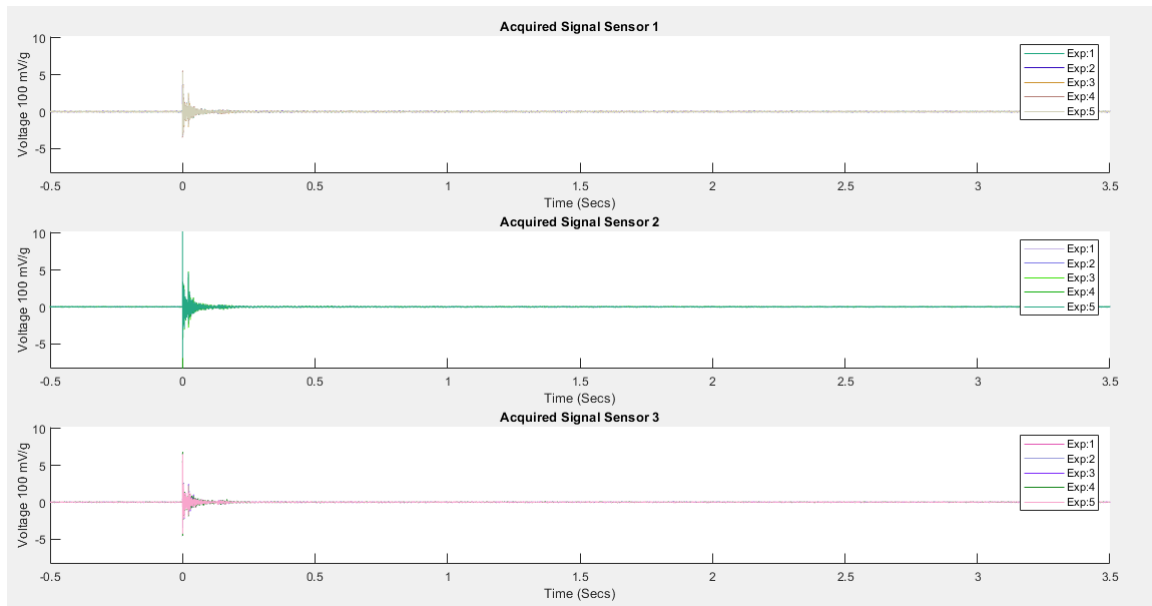


Figure 2.16: Group of five hits measured from the three accelerometers.

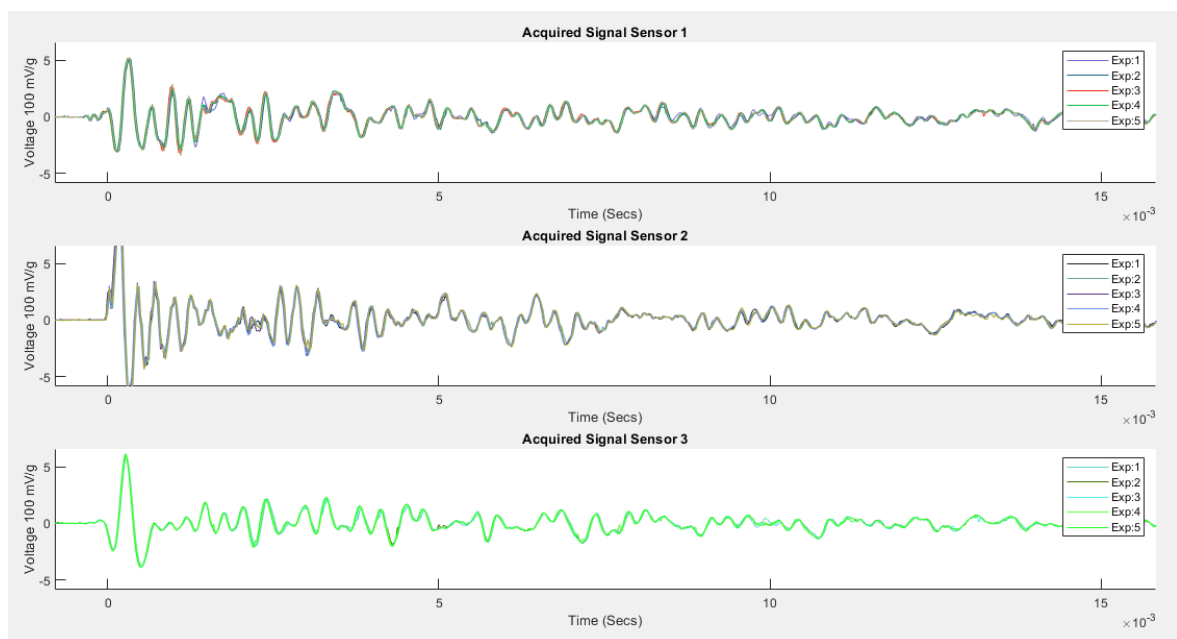


Figure 2.17: Zoom over the acquisitions.

2.3.3.- Design of the tests

Before starting the tests, some procedural conventions had to be defined for the sake of clarity. As the goal of the tests to be performed is to detect failures in the structure, it has been decided to deliberately induce several kind of failures in the bar by putting weights, loosening some screw or using different plates for joining the sections of the steel bar. Each screw has been assigned a number from one to eight, as it is shown in figure 2.18. Eleven labels have been defined to refer to each kind of working condition of the beam, they can be seen in table 2.1.

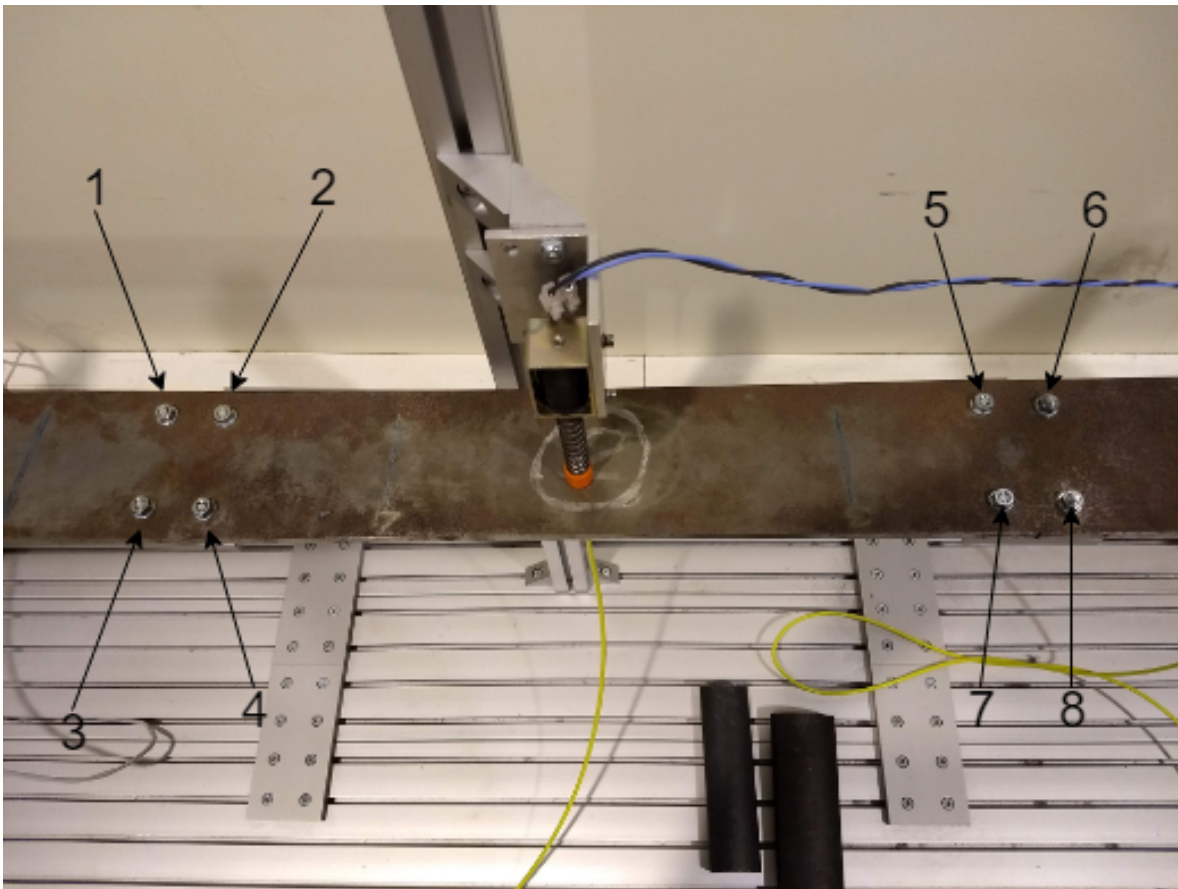


Figure 2.18: Numbering convention used for setting the number of each screw.

label no.	failure	no. of loose screw
1	normal conditions	
2	loose screw	1
3	loose screw	4
4	loose screw	5
5	loose screw	8
6	10 kg weight over left side	
7	10 kg weight over right side	
8	8 kg weight over left side	
9	8 kg weight over right side	
10	3D-printed plastic join	
11	3D-printed plastic join with narrowness	

Table 2.1: Definition of failure classes of the structure.

In the case of normal conditions it has been checked with a *torque wrench* that each screw has been screwed with a torque of around $16 N \cdot m$.

In the case of classes 2, 3, 4 and 5, when a screw is said to be loose, it's been tightened to $4 N \cdot m$. The rest of the screws are tightened to $16 N \cdot m$.

In classes 6, 7, 8 and 9, weights have been put over the screws of either one side or the other as it is shown in figure 2.19 and all the screws are tightened to around $16 N \cdot m$.

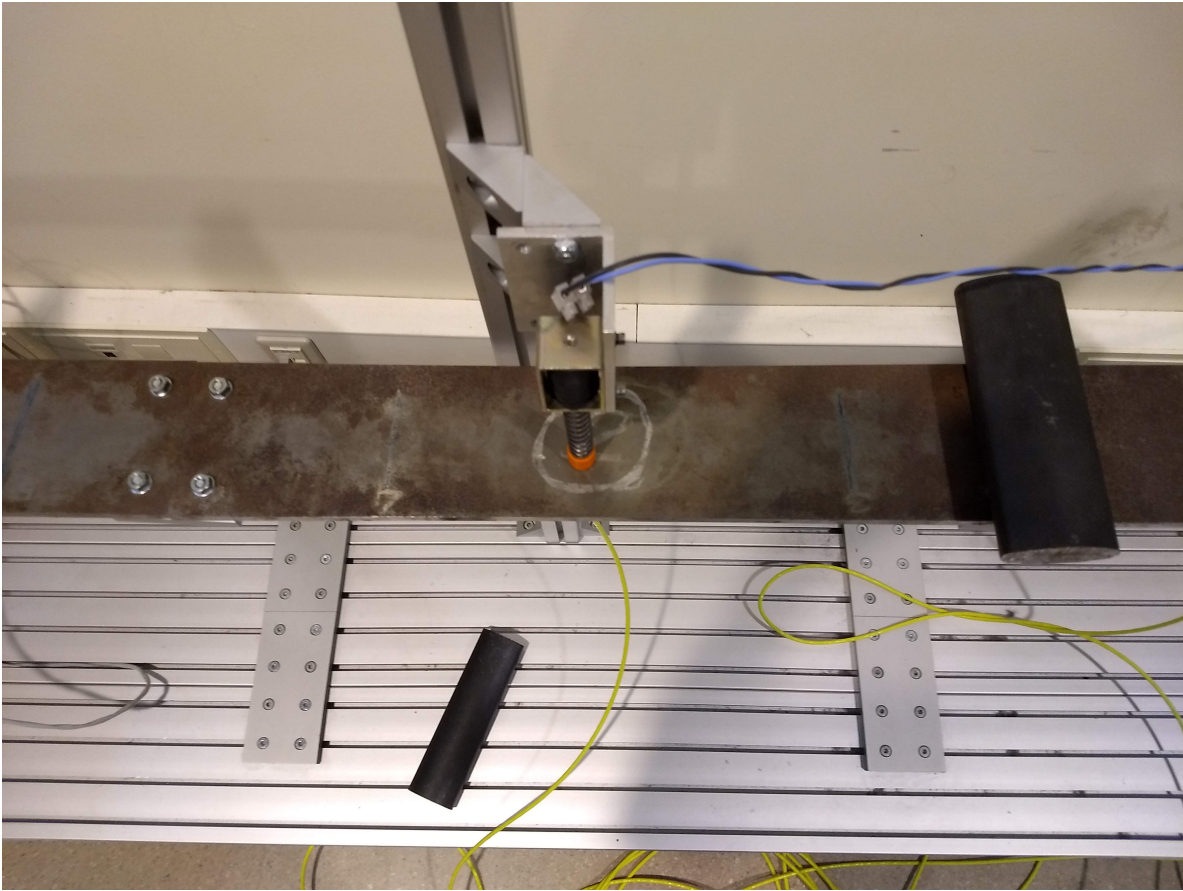


Figure 2.19: Weight over the beam before performing a test labeled in the class no. 7.

In classes 10 and 11, 3D-printed plastic joins in the center and right of figure 2.20, respectively, have been used to join the steel bar just in the left side, where screws 1 to 4 are located, and all the screws are tightened to around $16 N \cdot m$. Note that each shown plate has a couple in order to screw in four screws. The steel plate shown in the left of figure 2.20 has been used for classes 1 to 9.

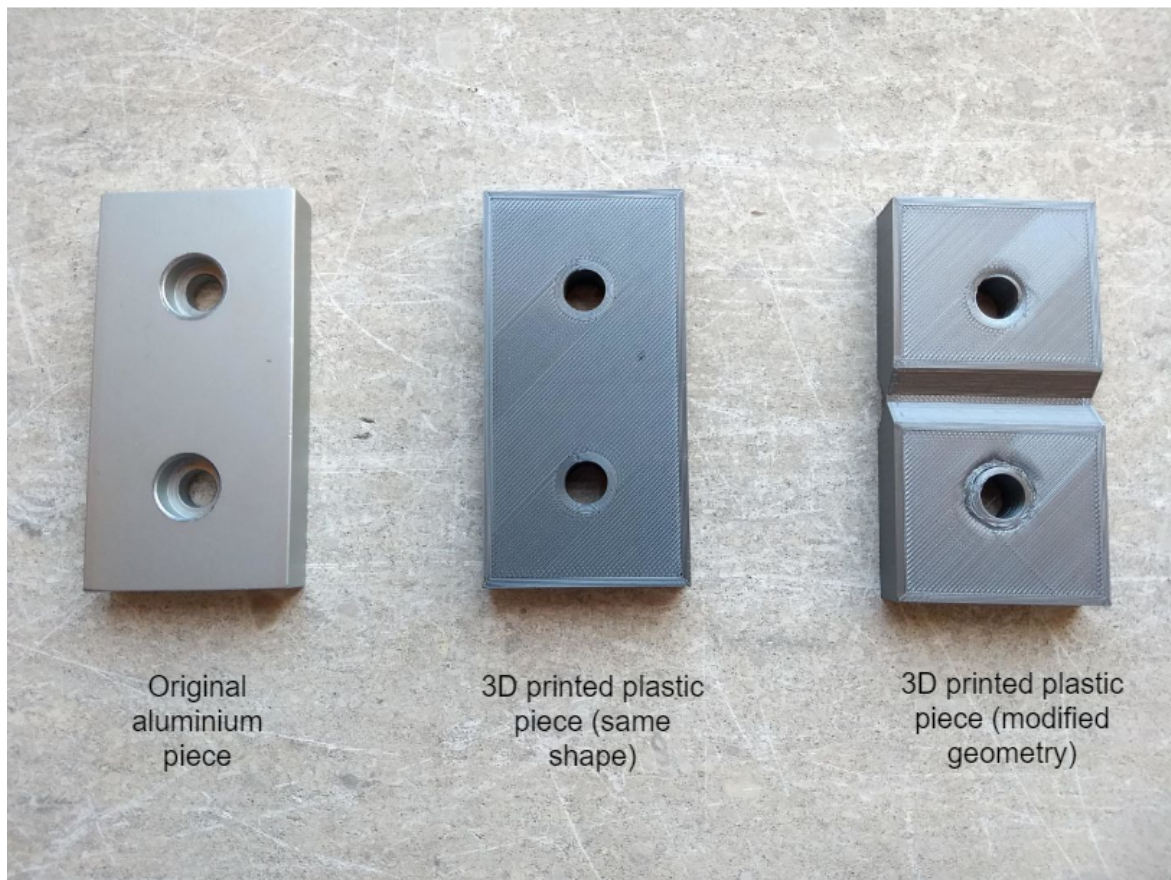


Figure 2.20: Different joining plates used.

This will be the way to proceed for each experiment or set of hits in the same condition:

- Induce one of the predefined failures in the system.
- Start the experiment: The user runs the Matlab scripts and types the parameters of the test in the command line. Then, the hammer will hit the steel bar each 15 seconds for the selected number of times while the acquisition system stores the measurements and metadata.

2.4.- Data pre-processing: feature extraction

According to [20], *Feature extraction*, seen from the *SHM* perspective, "is the process of identifying damage-sensitive properties derived from the measured vibration response, which allows one to distinguish between the undamaged and damaged structure". This process reduces drastically the memory needed to store the data and the computational capacity needed to process it and, at the same time, tries to preserve as much information as possible. It will also make much easier the subsequent application of *ML* algorithms.

These damage-sensitive properties or *features* can be experimentally chosen by observing the dynamic response of the system by means of *frequency analysis*, which will provide us with reliable indicators.

In addition, a *visualization technique* will be tested with the results of the feature extraction in order to check the repeatability of the experiments.

2.4.1.- Feature extraction: spectral analysis

To gain some intuition about which frequency bands should get more attention, *Fourier transforms* are very useful tools that decompose a waveform in individual sinusoidal components with an specific amplitude, frequency and phase. *Fast Fourier Transform (FFT)*, which is an efficient version of the *Discrete Fourier Transform (DFT)*, will be used to analyze signals in the frequency domain. DFT is defined as:

$$(2.3) \quad Y_n = \sum_{k=0}^{N-1} y_k e^{-jn\theta_0 k}, \quad n = \{0, 1, \dots, N-1\}$$

where Y_n is the output sequence in the frequency domain, y_k the input sequence in the time domain, N the number of samples in a period, k the position in the time sequence, n the number of harmonic and $\theta_0 = 2\pi/N$ the normalized fundamental frequency (which has the relationship $\theta_0 = \omega T_s$)[22].

The *RMS (Root Mean Square)* value of a signal is also an essential tool when working with waveforms because it can express the energy of a signal over a period of time. Taking into account that a signal can be decomposed into single components with an specific frequency by means of *FFT*, *RMS* value of a grouping of these components can be performed, obtaining the energy of a signal in an specific frequency band.

Computing the RMS value of a discrete periodic signal y_k in a frequency band, with a period of N samples, can be deduced from the *Parseval* theorem, which states that the power of the

signal in both the frequency and time domains is equal:

$$\sum_{k=0}^{N-1} |y_k|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |Y_n|^2$$

where Y_n are the Fourier (DFT) coefficients of the signal. Dividing by N and obtaining the square root:

$$\sqrt{\frac{1}{N} \sum_{k=0}^{N-1} |y_k|^2} = \frac{1}{N} \sqrt{\sum_{n=0}^{N-1} |Y_n|^2}$$

the expressions of the RMS value in both the frequency and time domains is obtained, respectively. By restricting the summation in the frequency domain expression to the terms corresponding to the frequencies inside an specific frequency band and multiplying by a factor of 2 in all terms, except the DC term and the Nyquist term, it is possible to get the RMS value of that band [23].

A windowing technique is applied to perform the feature extraction. It consists in computing the *FFT* of several overlapped windows of size N samples and displacement δ and performing the *RMS* values of the selected frequency bands at each window. The results of this operation can be concatenated obtaining the evolution in time of the energy of each frequency band. A condensed representation of this first process can be seen in figure 2.21. Later, three particular time intervals are selected and it is performed the mean of the *RMS* values of each frequency band at each interval. After performing the whole feature extraction process, a feature vector will be output for each hit and it will have the following shape: $\mathbf{x} \in \mathbb{R}^n$, being $n = \text{number of sensors} \times \text{number of frequency bands} \times \text{number of time intervals}$. In this way, *ML algorithms* will be able to compare information between different hits in both frequency and time domains in order to extract conclusions.

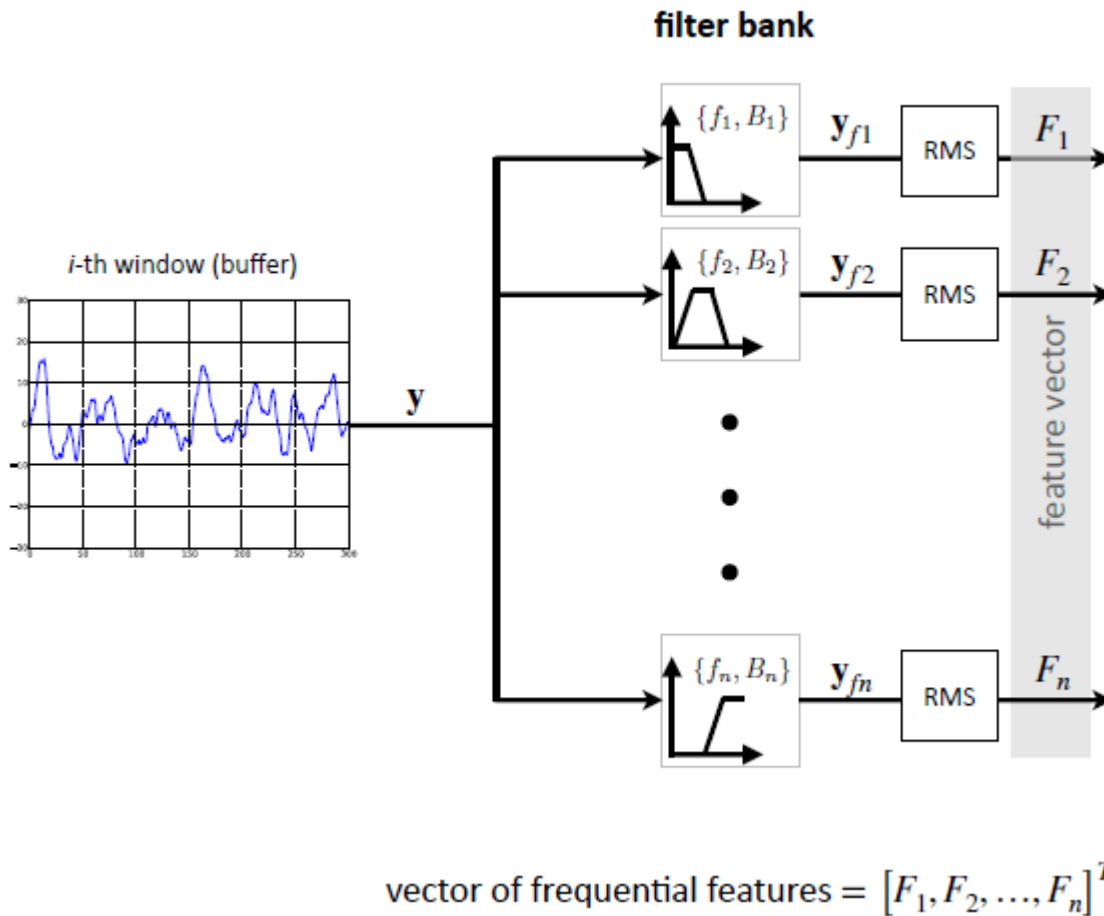


Figure 2.21: Representation of the frequency feature extraction process.

Source: [21]

It is impossible to get all the possible information in the time domain and in the frequency domain at the same time. So, the selection of the size N of the windows has to be done making a compromise. A too high value of N would provide a very rich information in the frequency domain, but a very poor information in the time domain. The opposite would happen choosing a too low value of N .

In order to choose the adequate frequency bands for the feature extraction, the frequency content of time signals of hits can be experimentally checked with the help of the above described technique or performing *Welch's Power Spectral Density estimate*. The first one lets one see the evolution of the energy over time in different frequency bands while the second one, by averaging the spectrum of FFTs of overlapped windows, reduces random variations due to noise, resulting in a cleaner spectrum. *Welch's PSD* does not let one check the energy evolution over time, but it allows one to appreciate the most active frequency regions of a signal. The frequency peaks found with either method indicate the characteristic frequencies of the vibration signal and if

they are properly selected, their variation in energy over time can be a very precise indicator of the working condition of a structure.

2.4.2.- Data visualization: Repeatability tests

Data visualization, which implies the creation and study of the visual representation of data, is considered as a modern field of visual communication [27].

The most simple approach for checking the repeatability of tests is to observe time signals of several hits from different classes and try to spot similarities and discrepancies between them.

In order to try to take a deeper advantage of data visualization techniques to study the repeatability of the tests, it has been proposed to develop an algorithm that produces an image that contains condensed information about a simple hit, so anyone who doesn't know about SHM or spectral analysis could spot differences or similarities between various hits by finding similarities or differences between their corresponding images. To do this, the feature vector has to be reshaped so it gets a similar number of rows and columns. Then the value of each element of the new matrix is assigned to a colour proportional to its value so the human eye can recognize patterns easier.

2.5.- Intelligent Data Analysis: Machine Learning algorithms

To develop Intelligent Data Analysis algorithms applied to SHM, which are currently helping to make supervision systems more and more autonomous, is among the main goals of GSDPI research group. In fact, the data acquired during the development of this project will be used in the future to test innovative IDA algorithms that allow a reliable *fault detection and diagnosis* and a subsequent robust *decision making*.

This section corresponds to the *model development* step presented in section 2.1.2. ML algorithms, that permit the implementation of IDA techniques to fields such as SHM, will be the subject of study along this section. The upgrades that ML brings are truly meaningful, apart from making systems more autonomous, it lets one make inferences about the health state of a structure without having a great idea about the physics that govern it. Figure 2.22 shows the different stages to be addressed when facing a problem from a Machine Learning approach. Notice the value that is placed on the feedback that further steps in the process can provide to previous ones, as it is also stated in CRISP-DM methodology.

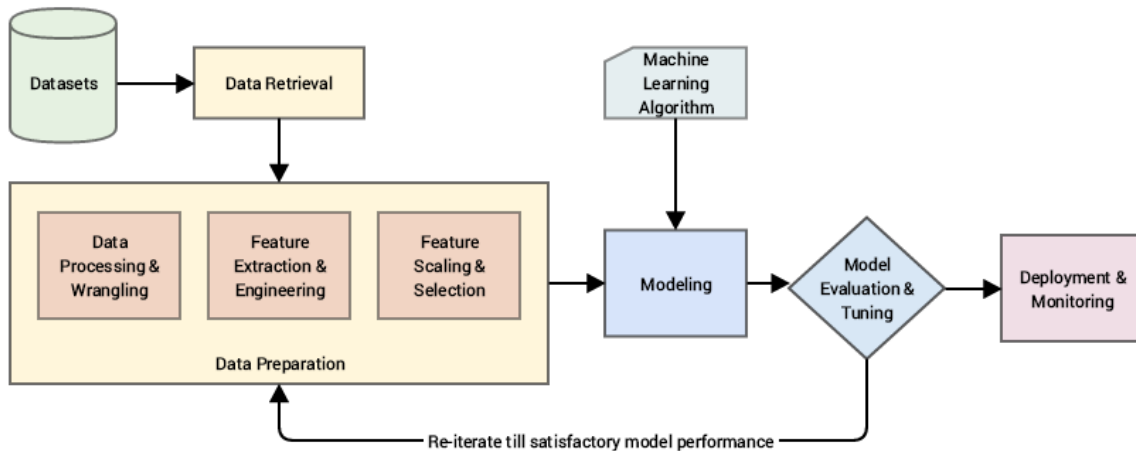


Figure 2.22: Stages of a Machine Learning project.

Source: [4]

A brief introduction about *Machine Learning algorithms* and an explanation about *supervised and unsupervised learning algorithms* used in this project will be the topics in this section.

2.5.1.- Introduction to Machine Learning

Machine Learning (ML), which is seen as a subset of Artificial Intelligence (AI), consists of the scientific study of algorithms and statistical models used by computer systems to efficiently carry out a certain task by relying on patterns and inference and without having received explicit instructions [34, 35].

To better understand the concept of *ML*, two more perspectives are provided in [5]. Arthur Samuel described it as: "the field of study that gives computers the ability to learn without being explicitly programmed," although it is an older and informal definition. Tom Mitchell provided a more modern definition: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ."

ML algorithms, that have been taking over in the last years, have become one of the most powerful tools available for fault detection and diagnosis. Although there are several branches in *ML* algorithms, this project will focus on *supervised learning* and *unsupervised learning* algorithms applied to damage detection.

The general approach when it comes to applying *ML* is to train an algorithm, which will build a mathematical model with a *training set* of sample data; then, test the algorithm with a *test*

set of data; and finally check if the algorithm made the desired predictions or decisions. It is also often used a *cross validation set* which lets evaluate parameters of the algorithm without making it particularize over the training set, this possible situation is what is called *overfitting* or *high variance* and it is unwanted. The opposite situation is called *underfitting* or *high bias* and consists in the lack of precision of the algorithm due to its low complexity. The goal is to look for an algorithm that generalizes well for every possible acquired data from the studied process. The optimal situation is a balance between both *overfitting* and *underfitting* and it can be achieved with the help of a *regularization* technique which will be explained later.

2.5.2.- Supervised Learning

In *supervised learning*, the handled data set contains the actual output of the system apart from the feature vector or input vector. These algorithms will aim to make predictions over new examples outputting the predicted value for each new evaluated test example.

Supervised learning problems are divided into *regression* and *classification* problems. In a *regression* problem, the aim is to predict results within a continuous output, trying to map input variables to some continuous function. In a *classification* problem, the aim is to predict results in a discrete output, trying to map input variables into discrete categories [5]. Regression algorithms are able to find the existence of anomalies, but they are not very efficient at characterizing the damages. In contrast, *classification* algorithms are able to detect and characterize damages when they are trained with a data set that contains enough information about the possible failures.

The notation for dealing with these algorithms follows the next rules as it is defined in [5]. It will be used $\mathbf{x}^{(i)}$ to denote the input or *feature vector* and $\mathbf{y}^{(i)}$ to denote the output or *target variable* to be predicted. A pair $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is called a *training example*, and the data set used to learn—a list of m training examples $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$; $i = 1, \dots, m$ —is called a *training set*. Being m the number of training examples and n the number of available features. The superscript $(i) \in \{1, 2, \dots, m\}$ and the subscript $j \in \{0, 1, \dots, n\}$ can be used in the next way to denote the j^{th} feature of the i^{th} example: $x_j^{(i)}$. As a convention, a new feature $x_0 = 1$ is added to every example, which makes easier the subsequent operations with matrices. $\mathbf{X} \in \mathbb{R}^{m \times (n+1)}$ is used to denote the space of input values and $\mathbf{Y} \in \mathbb{R}^{m \times 1}$ to denote the space of output values. In case there are only two different categories, it is called *binary classification* and $\mathbf{y}^{(i)} \in \{0, 1\}$ expresses if the example belongs to the *negative* or to the *positive* class, respectively. In the current case, which is an example of *multiclass classification*, the output can take as much values as existing categories, which are eleven: $\mathbf{y}^{(i)} \in \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$.

Once extracted the feature matrix \mathbf{X} as explained in section 2.4, a common and useful practice is to perform *mean normalization* and *feature scaling* in order to optimize the performance of

the algorithms. *Mean normalization* consists in subtracting the mean of each input feature \mathbf{x}_j along the data set $\boldsymbol{\mu}_j$ from the corresponding input feature $\mathbf{x}_j^{(i)}$ and *feature scaling* consists in dividing the input features $\mathbf{x}_j^{(i)}$ by the standard deviation σ_j of each of the correspondent input features \mathbf{x}_j along the data set. This results on a feature matrix $\hat{\mathbf{X}}$ with zero-centered features and standard deviation equals 1. In case the features are already in similar scales *feature scaling* is not essential. This transformation is computed in this way:

$$(2.4) \quad \boldsymbol{\mu}_j = \frac{1}{m} \sum_{i=1}^m \mathbf{x}_j^{(i)} \quad \sigma_j = \sqrt{\frac{\sum_{i=1}^m (\mathbf{x}_j^{(i)} - \boldsymbol{\mu}_j)^2}{m}} \quad \hat{\mathbf{x}}_j^{(i)} = \frac{\mathbf{x}_j^{(i)} - \boldsymbol{\mu}_j}{\sigma_j}$$

Linear methods like *logistic regression* and non-linear methods like *neural networks*, which can be used to perform multiclass classification, will be used to try to predict categories of damage in the structure.

2.5.2.1.- Logistic Regression

Logistic regression is a linear method based on the optimization of a parameter vector $\boldsymbol{\theta}$, that is used to predict whether an example belongs to one class or another through the hypothesis function $h_{\boldsymbol{\theta}}(\mathbf{x})$ shown in equation 2.5. This hypothesis expresses the probability that a data example belongs to the *positive class* given its feature vector $\mathbf{x}^{(i)}$ and it makes use of the sigmoid function $g(z)$. Equations 2.6 define the boundary conditions of the hypothesis. In the current case of *multiclass classification*, a *one-vs-all* approach is used, which consists in choosing one class and then grouping all the rest into a single second class. This is done over and over, applying *binary logistic regression* to each case, and then using the hypothesis that outputs the highest value as the final prediction $\mathbf{y}^{(i)}$ [5]. $\boldsymbol{\theta}$ can be trained by means of a gradient descent (equation 2.8) over the cost function defined in equation 2.7. *Feature scaling* is necessary to optimize this process. Gradient descent is performed for multiple iterations until convergence in order to make the algorithm as accurate as possible. The parameter α is the step of the gradient descent and λ is the regularization parameter, which can be used to avoid overfitting of the training set.

$$(2.5) \quad h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x}) \quad z = \boldsymbol{\theta}^T \mathbf{x} \quad g(z) = \frac{1}{1 + e^{-z}}$$

$$(2.6) \quad \begin{array}{lll} h_{\boldsymbol{\theta}}(\mathbf{x}) \geq 0.5 \rightarrow y = 1 & g(z) \geq 0.5 & \boldsymbol{\theta}^T \mathbf{x} \geq 0 \rightarrow y = 1 \\ h_{\boldsymbol{\theta}}(\mathbf{x}) < 0.5 \rightarrow y = 0 & \text{when } z \geq 0 & \boldsymbol{\theta}^T \mathbf{x} < 0 \rightarrow y = 0 \end{array}$$

$$(2.7) \quad J(\theta) = -\frac{1}{m} \sum_{i=1}^m [\mathbf{y}^{(i)} \log(h_{\theta}(\mathbf{x}^{(i)})) + (1 - \mathbf{y}^{(i)}) \log(1 - h_{\theta}(\mathbf{x}^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

$$(2.8) \quad \theta_j := \theta_j - \alpha \frac{\delta}{\delta \theta_j} J(\theta)$$

2.5.2.2.- Neural Networks

Neural Networks (NN) are a very powerful kind of supervised learning non-linear algorithm. Their performance will be compared to the one of logistic regression in this project, but, as the results obtained with logistic regression were succesful, NN won't be described in depth. A more complex vibration analysis problem may necessarily need the application of NN.

Artificial Neural Networks are inspired by the behaviour of the human brain itself. NN consist of a group of artificial neurons, connected to each other in a way that an input feature vector can be mapped to an output vector by means of the trained weights that define the function mapping from each layer to the next one. The layers in between the input layer and the output layer are known as hidden layers. The process of predicting an output for a given input vector is called forward propagation. NN can be trained as any other ML algorithm, in this case following the backpropagation method for upgrading the weights of the network during the gradient descent that minimizes the cost function (in case the reader wants to get a deeper comprehension of froward and backpropagation, additional information can be found in [5, 7]). Once the NN have been trained, they can be applied to new data examples in order to predict whether the output is positive or negative or, as it is the case in this project, produce a one-hot encoded output associated to the class that the evaluted data example is most likely to belong to. This last case is the multiclass approach, where the output unit with the biggest value will be considered as the predicted class, similarly to how it was done in logistic regression.

2.5.3.- Unsupervised Learning

As it is said in [5]:

"Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables. We can derive this structure by clustering the data

based on relationships among the variables in the data. With unsupervised learning there is no feedback based on the prediction results".

Methods like *clustering*, which consists in grouping together sets of data examples that are more similar to each other than to the rest of data, or *dimensionality reduction (DR)*, which consists in reducing the size of the feature vector while trying to maintain a high percentage of the differences and similarities between data examples, belong to this field.

The notation used to deal with these algorithms is almost the same as the one used in the case of *supervised learning*, section 2.5.2, with a few tweaks. In this case, the algorithms only manage feature vectors $\mathbf{x}^{(i)}$, the output variable $\mathbf{y}^{(i)}$ is not taken into account. In addition, the column $\mathbf{x}_0 = 1$ is not added to the input vector as it is done in the case of *supervised learning*. *Feature scaling* and *mean normalization* have to be performed in the same way as it was explained in section 2.5.2 in order to optimize the performance of the algorithms.

This project emphasizes the benefit of dimensionality reduction applied to fault detection and data visualization, using linear techniques like *Principal Component Analysis (PCA)* and non-linear techniques like *T-distributed Stochastic Neighbor Embedding (t-SNE)*.

2.5.3.1.- Principal Component Analysis

Principal Component Analysis (PCA) is a statistical method that maps, by means of an orthogonal transformation, a set of sample data, whose measured variables are possibly correlated, into a new space founded on linearly uncorrelated variables known as *principal components* [3].

PCA can be used for converting $\mathbf{x}^{(i)} \in \mathbb{R}^n$ into a lower dimension vector $\mathbf{z}^{(i)} \in \mathbb{R}^k$ being n the dimensions of the original feature vector and k the number of dimensions to which the data will be reduced. It is very useful for visualization purposes as it is able to reduce the dimensions to a sufficiently small value so data can be plotted, while at the same time it preserves as much information as possible. It is desirable that the resulting low dimensional vector basis keeps a high percentage of the total variance in the data [5].

Before performing the transformation, the *covariance* (equation 2.9) of the input matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$, $\mathbf{\Sigma} \in \mathbb{R}^{n \times n}$, has to be decomposed in singular values (SVD) (equation 2.10):

$$(2.9) \quad \mathbf{\Sigma} = \frac{1}{m} \sum_{i=1}^m (\mathbf{x}^{(i)})(\mathbf{x}^{(i)})^T$$

$$(2.10) \quad \Sigma = USV^T$$

where $U \in \mathbb{R}^{m \times m}$ contains the left singular vectors or principal components, $S \in \mathbb{R}^{m \times n}$ contains the singular values of Σ on its diagonal or the variance along each component and ordered by size and $V \in \mathbb{R}^{n \times n}$ contains the right singular vectors.

After the SVD is computed, the transformation, which consists in keeping the singular vectors associated to the highest singular values, is performed. A new matrix $U_{reduced} \in \mathbb{R}^{m \times k}$, which is extracted from U , is the result of this operation. Then, data can be mapped from $X \in \mathbb{R}^{m \times n}$ into $Z \in \mathbb{R}^{m \times k}$ by means of this equation:

$$(2.11) \quad \mathbf{z}^{(i)} = U_{reduced}^T \mathbf{x}^{(i)}$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^{n \times 1}$ and $\mathbf{z}^{(i)} \in \mathbb{R}^{k \times 1}$.

2.5.3.2.- T-SNE

T-distributed Stochastic Neighbor Embedding (T-SNE) is an unsupervised learning algorithm able to perform a non-linear dimensionality reduction for visualization purposes. T-SNE models each data example in a way that similar points in the high dimensional space are close in the low dimensional space and dissimilar points are distant [6].

An important parameter to have into account when training T-SNE is the *perplexity*, which indicates whether the algorithm will try to find more local differences within the data for low values of this parameter or more global differences for high values. A proper value of the perplexity has to be achieved by checking how output data are clustered.

As in the case of NN, T-SNE hasn't been studied in depth in this project but it will be used to compare its results with PCA ones.

CHAPTER



Results

This section is dedicated to present the output of the methods previously explained. Observations about the data acquisition and the feature extraction processes, the results of the repeatability tests, a digression about problems with noise and the output of Machine Learning algorithms will be shown below.

3.1.- Data acquisition

After checking the *FFT* of several acquisitions it was seen that information was condensed in the 0 to $10kHz$ band. So according to Nyquist-Shannon theorem, it is more than enough to choose a sample rate S_r of $40kHz$ for all the acquisitions.

Concretely, data used for computing the below shown results was acquired in three different days. On the first day, 270 total hits were recorded in nine experiments of 30 hits belonging to each of the nine first classes defined in section 2.3.3. Due to a mistake in the acquisition one hit of each class had to be erased resulting in a 261 data set. On the second day, same approach was taken, 30 hits belonging to each one of the first nine classes were acquired, obtaining 270 new data examples. On the third day, 30 hits of classes 1, 10 and 11 were captured, in order to compare the influence of different joining plates. This resulted in a new data set of 90 hits.

3.2.- Feature extraction by means of spectral analysis: observations

Welch's PSD was used for analyzing several hits from different classes in order to see which frequency bands to choose for the feature extraction. One thing was noticed here, the selected bands could be different for each of the three accelerometers. Even if some active regions in frequency coincide for the three sensors, such as 80 Hz and 610 Hz, others do not. Selecting different bands for each sensor can be an advantage, allowing to make a more efficient use of the feature vector. As it can be seen in figure 3.1, some peaks coincide but others do not.

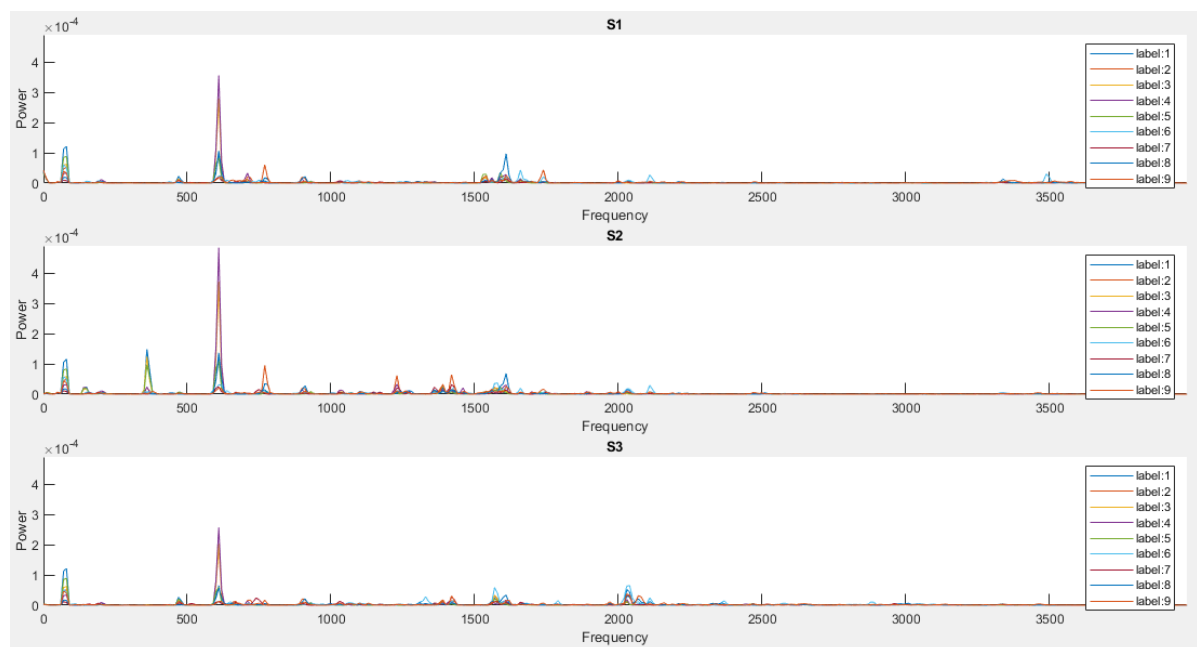


Figure 3.1: Welch's Power Spectral Density of diverse hits from different classes computed for the measurements from each one of the three accelerometers S1, S2 and S3.

For the feature extraction algorithm, size N of the window was set to 4000 samples, which was found to be a good balance between information got from time and frequency and δ was calculated for an overlapping of 90 %; nine 40Hz wide frequency bands were chosen independently for each of the three sensors; and the three time intervals where the energy in this bands was evaluated were: 0.4s to 0.5s, 0.5s to 0.6s and 0.6s to 0.8s (being 0.5s the moment in which the hammer hits the bar). The explanation for choosing the first interval before the hit happens is because the windowing technique displaces the information in time. The energy evolution over time in the nine chosen bands for accelerometer number 3 can be seen in figure 3.2, where different behaviours for different conditions of the steel bar can be spotted.

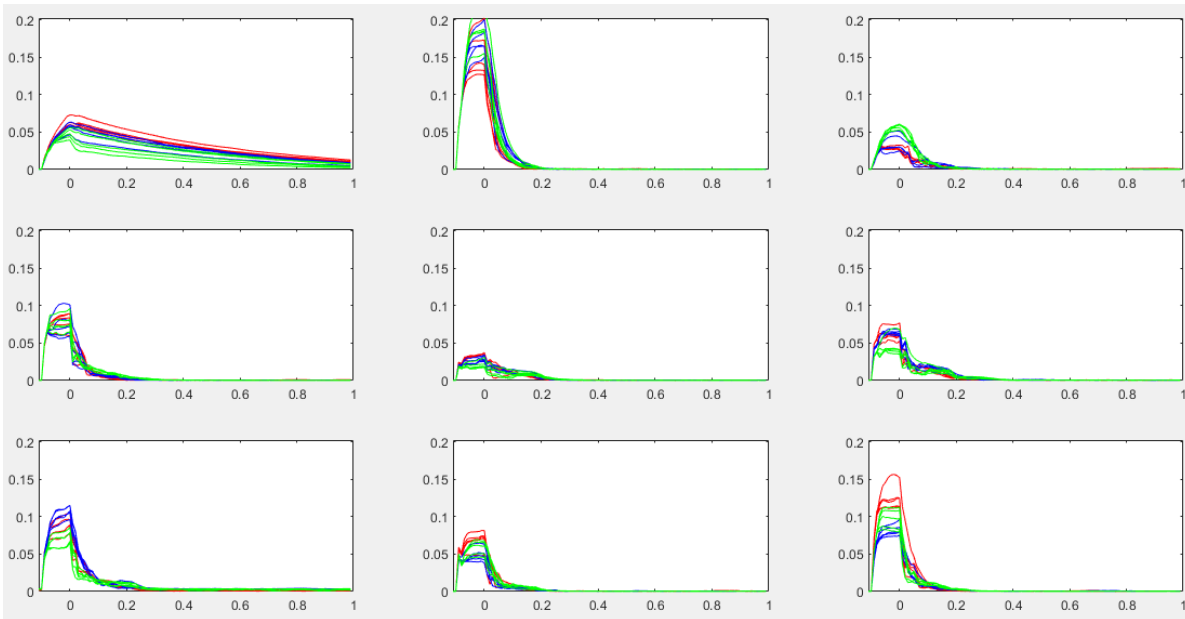


Figure 3.2: Energy evolution over time of the nine chosen frequency bands for hits from classes 1 (red), 2 (blue) and 4 (green) measured from accelerometer number 3.

Performing the feature extraction with this technique over nine frequency bands and three time intervals in the three sensors produced a feature vector $\mathbf{x}^{(i)}$ of $9 \times 3 \times 3 = 81$ dimensions. If this feature extraction is performed over all the acquisitions, a feature matrix $X \in \mathbb{R}^{m \times n}$ will be obtained, being m the total number of hits and n the total number of features. The code that has been used in this feature extraction process can be seen in D.2.

3.3.- Outcome of the repeatability tests

It was accidentally noticed that the addition of the 3D-printed part to the hammer helped increasing the repeatability of the tests. It could be checked by listening to the sounds the hits produced, which became more similar one to another and also by observing the time signals of the acquisitions. The similarity of hits from the same class and the differences between hits from different classes can be seen in figure 3.3.

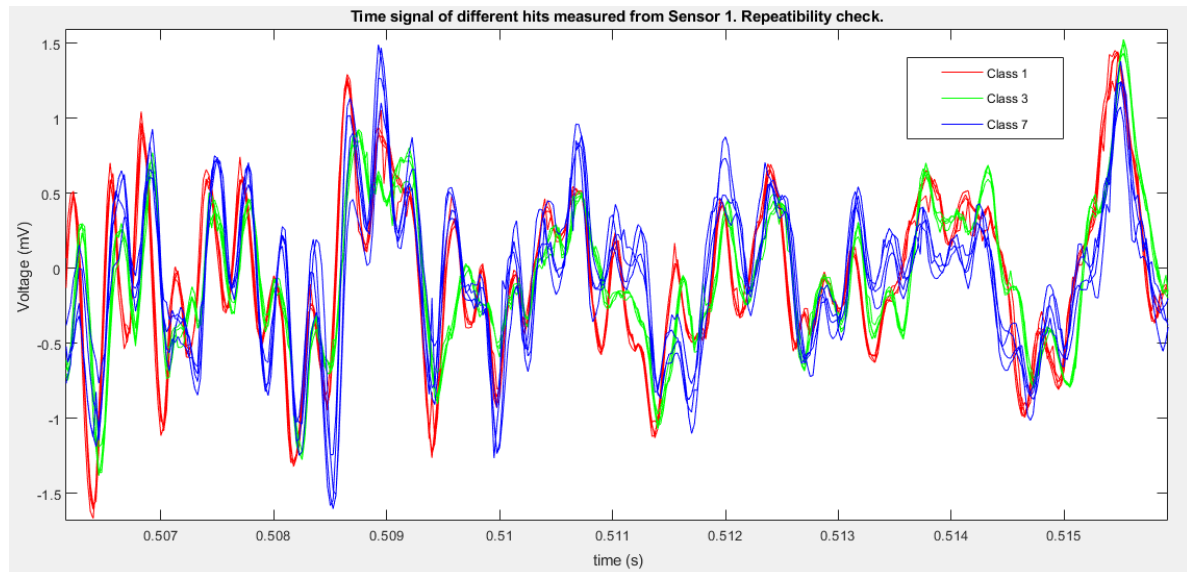


Figure 3.3: Repeatability check for several hits from classes 1 (red), 3 (green) and 7 (blue) measured from accelerometer number 1.

The algorithm explained in section 2.4.2 has been tested with data coming from 45 hits from the same day, where 5 hits were carried out for each one of the first nine classes. The normalized feature vector, that is composed by 81 elements, has been reshaped so that it forms a squared matrix of size 9×9 . The results of this algorithm are shown in figure 3.4. Different classes are distributed in columns and different hits in rows, so the purpose is to find similarities in the columns of this figure and differences between different columns.

It is checked that, even if some blurry patterns could be guessed, it is not an effective method for this exact purpose.

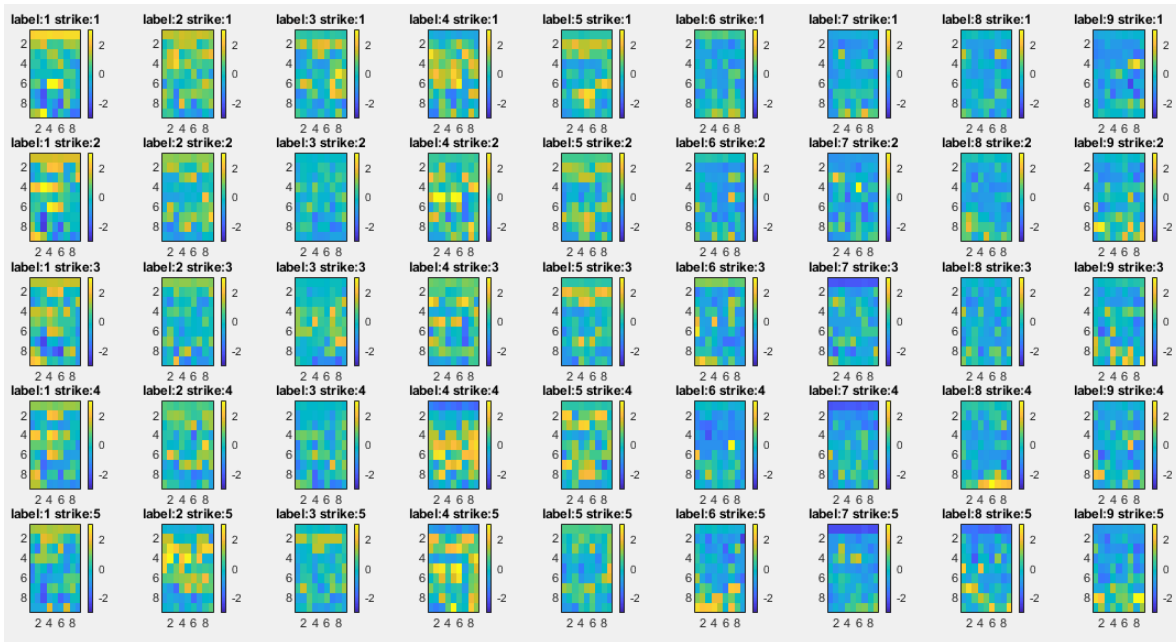


Figure 3.4: Repeatability test performed over 5 hits of each of the first 9 classes.

3.4.- Noise

While observing the first measurements, when the test bench was being developed, it appeared a constant noise in the capture of the accelerometers whose source wasn't possible to be found at a first glance. The noise was structured and constant, but it was split in all the frequency domain so it was imposible to filter it out. Lots of possible sources of noise were proposed but none of them was proved to be the main one. Some options like electromagnetic noise captured by the steel bar directly transmitted to the accelerometers or the magnetic field created by the hammer were taken into account. Every device was plugged to the same connector of the electrical network so they could share the ground connection, but there were no noticeable improvements. Apart from the main noise, it was possible to appreciate the noise coming from the electrical network at around $50Hz$ but this one was much smaller and also harder to eliminate.

It was deduced that these accelerometers aren't working in their optimal conditions. Using some accelerometers with a bigger sensitivity and less susceptible to noise would improve the performance of these acquisitions.

Finally, it was decided to put the 3D-printed plastic cover shown in section 2.3.1.2 in all the three accelerometers. Although it acts as a low pass filter and has some resonance peaks, it considerably reduces the noise captured by the sensors and it respects the frequencies in which the algorithms will focus. In any case, when it came to the feature extraction, it was avoided to

choose the main frequencies of the remaining noise. Even if it could seem that the usage of the plastic covers in the accelerometers is not as clear as in the case of the hammer, it was decided together with the research team to keep them just in case some different feature extraction technique which is sensitive to the noise is applied in the future to the obtained acquisitions during this research.

Time signals and the *spectrogram* of one hit captured by the accelerometers can be seen in figures 3.5 and 3.6, respectively. A spectrogram is the representation of the amplitude of a signal with a colour code for different time intervals and frequency bands and it is also based on the windowing technique and *FFT*. First and second accelerometers are isolated with the plastic cover and the third one is not in order to highlight its influence on the measured vibrations. It can be clearly seen that the third window in both figures shows a bigger level of noise.

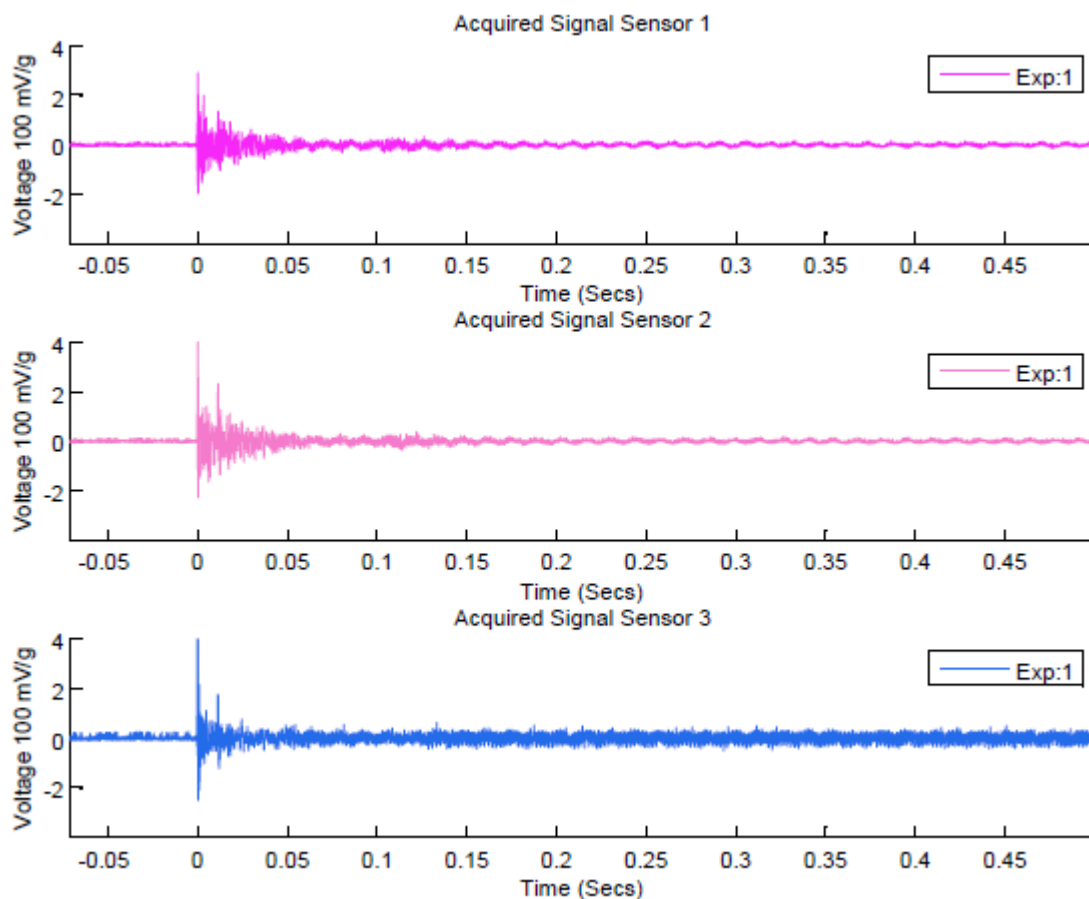


Figure 3.5: Time signal of accelerometers 1 and 2 with cover and accelerometer 3 without it.

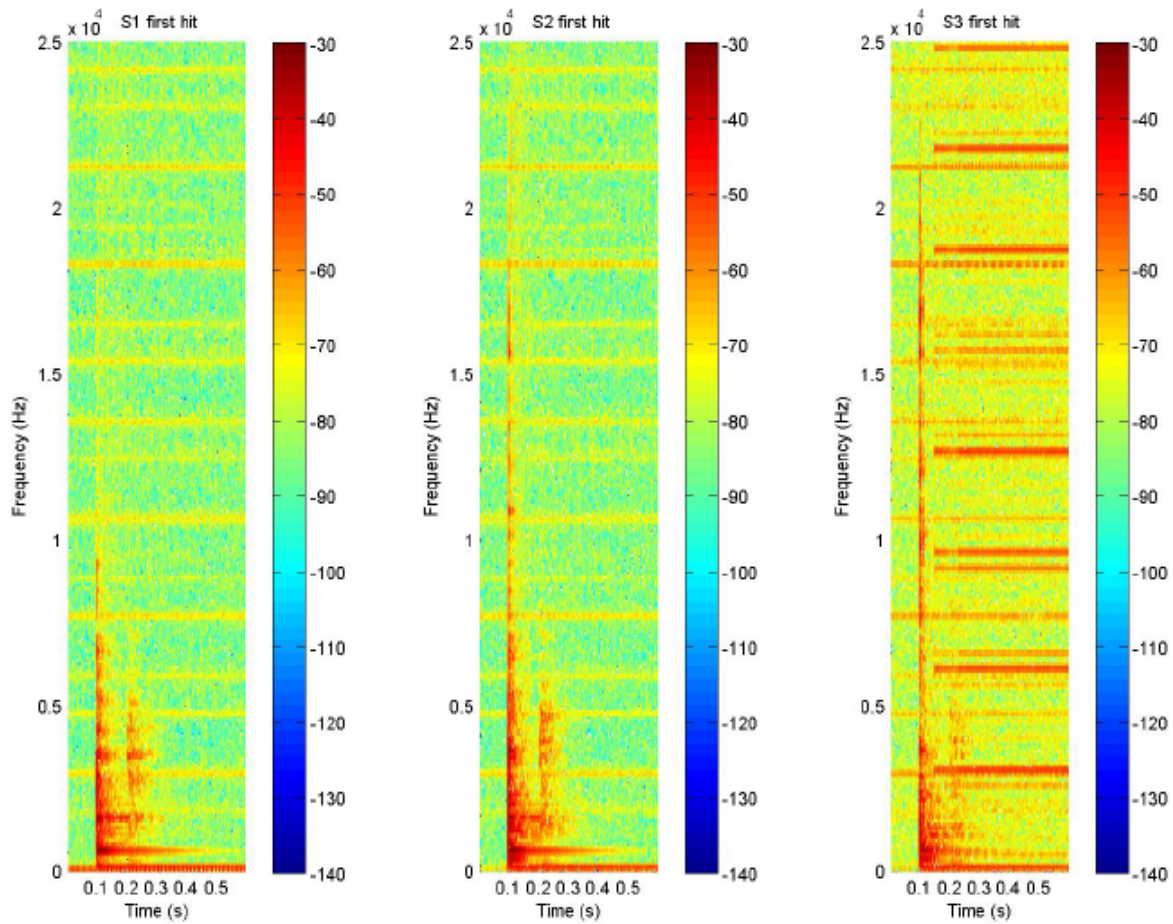


Figure 3.6: Spectrograms of accelerometers 1 and 2 with cover and accelerometer 3 without it.

3.5.- Machine Learning algorithms outcome

Machine Learning algorithms described in section 2.5 have been applied to the extracted feature vectors. The results of the applied classification and dimensionality reduction algorithms will be analyzed during this section.

It must be said that the code used for implementing *ML algorithms*, which is shown in D.3, was mainly extracted from the Machine Learning Course by Andrew Ng [5].

3.5.1.- Classification

Classification by means of *logistic regression* and *neural networks* has been applied to the data captured in the test bench. Once *feature scaling and mean normalization* were applied to all the data, the algorithms were trained in Matlab different ways in order to remark the dependence of the accuracy on the size of the training set and on the variability of the environmental conditions.

First, it will be shown the performance of LR and NN diagnosing classes from 1 to 9 in five different cases to show how the algorithms can make differences between loose screws, added weights and normal conditions. Later, examples belonging to classes 1, 10 and 11 will be diagnosed with LR to show how the algorithms make differences between different joining plates. The performance of these algorithms will be evaluated by means of the percentage of accuracy in the predictions and confusion matrices (shown in appendix A), which are tables that compare the true class of tested data examples versus their predicted class.

3.5.1.1.- Normal conditions, loose screws and added weights

Initially, both LR and NN algorithms were used following two different procedures: in *case 1*, training with data from the first day and testing with data from that same day and in *case 2*, training with data from the first day and testing with data from the second day. The training set size was set to 20% of the total amount of data and the remaining 80% was used as test set. The topology of the used neural network consists of: an input layer of 81 units, which is the size of the feature vector; a hidden layer of 200 units, which was experimentally found to be a good balance between computing speed and algorithm's prediction accuracy; and an output layer of 9 units, which is the number of evaluated classes. The obtained results are shown in the next item list:

- LR in case 1. Accuracy of 82.12%.
- NN in case 1. Accuracy of 89.37%.
- LR in case 2. Accuracy of 56.48%.
- NN in case 2. Accuracy of 60.19%.

Looking at the previous results and the associated confusion matrices shown in tables A.1 to A.4, three things can be said: a low size of the training set has made the algorithms overfit the few data examples that they were given during the training, causing a poor accuracy in the prediction. It can be also said that when it comes to predict data acquired in a different day than the day when the training set was acquired, it is clearly seen that the accuracy is decreased. The

reason for this is the high dependence of the dynamic response of structures on environmental conditions like temperature or moisture, which had been previously pointed out by Sohn et al. (2003) [20]. Lastly, it can be said that Neural Networks are able to achieve a little higher accuracy because they can learn non-linearities from the input data.

To improve the performance of the algorithms, the same procedure was used but this time choosing a training set size of 70% of the total amount of data and the remaining 30% was used as test set. The obtained results in *case 3*, training and testing with data from the same day, and *case 4*, training and testing with data from different days, can be seen below:

- LR in case 3. Accuracy of 98.61%.
- NN in case 3. Accuracy of 100%.
- LR in case 4. Accuracy of 81.48%.
- NN in case 4. Accuracy of 83.95%.

Looking at these results and tables A.5 to A.8, it can be seen that increasing the size of the training set, in comparison with *cases 1 and 2*, clearly helps improving the accuracy of the algorithms. This is because the algorithms get a richer information about the variability of data belonging to the same class and about the difference between data from different classes. It can be said that the algorithms are able to generalize better for new tested data examples. The results have improved, but still, when it comes to predict the class of data acquired in a different day than the training set, the accuracy decreases due to the variability of the environmental conditions. Also, it can be checked again the little higher accuracy of Neural Networks in comparison to Logistic Regression.

Finally, in order to avoid the dependence of the algorithms on the environment and achieve the optimal possible performance, LR and the same NN presented in the previous cases, were trained with data from both days and tested with data also from both days. The size of the training set was 70% of the total amount of data and the remaining 30% was used as test set. The results of this final upgrade, *case 5*, are shown below:

- LR in case 5. Accuracy of 98.04%.
- NN in case 5. Accuracy of 99.35%.

Looking at these final results and tables A.9 and A.10, it can be observed the unquestionable enhancement of the performance of the algorithms when they are trained with data acquired in

different days, learning to properly classify damage classes neglecting the circumstances of the moment of the acquisition. It can also be seen again how Neural Networks perform a little bit better than Logistic Regression, but the difference is not very significant in this case.

A bar chart showing the progressive improvement of the performance of Logistic Regression and Neural Networks in the five previously explained cases can be seen below:

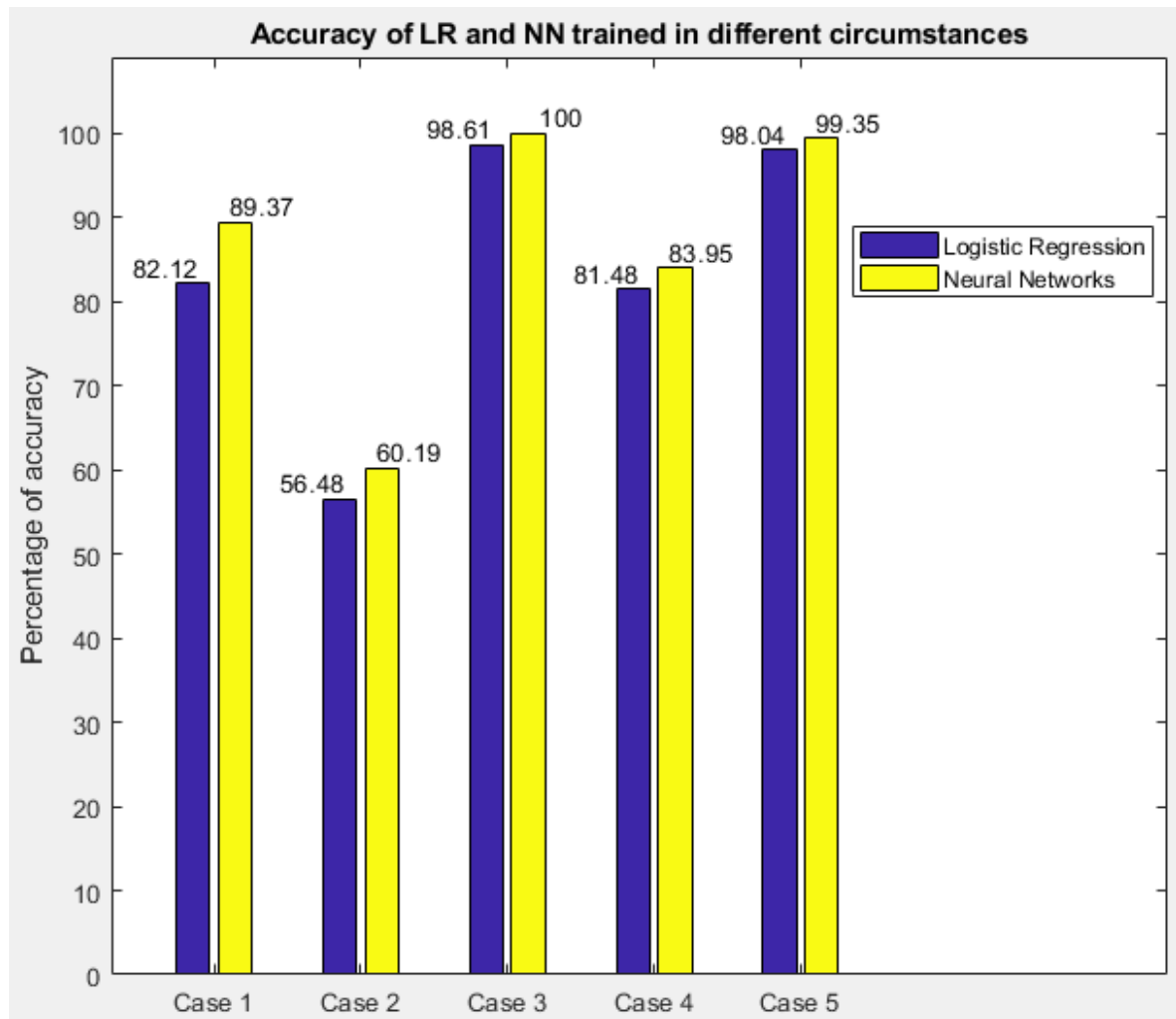


Figure 3.7: Bar chart showing the percentage of accuracy of LR and NN in the five described cases.

A condensation of the weights θ learnt by the trained Logistic Regression algorithm in *case 5* has been done from a data visualization approach. The weights have been separated in three different matrices associated to the data coming from each one of the three accelerometers. Then, the average of each group of three weights associated to the three evaluated intervals of time for a given frequency and accelerometer has been computed in order to show just classes versus frequencies. The results are presented in a discrete three-color scale in order to show low values

(lighter), medium values and high values (darker). Note that, before the averaging, the absolute value of each weight was performed, so darker squares are reliable indicators, but they can indicate that high values of energy in a frequency band make a hit belong to one class or the opposite, not belong to it. This is a very useful technique because it can let one know which features are contributing to the decisions taken by the algorithm with respect to each class. The three following figures, 3.8, 3.9 and 3.10, show the learnt indicators by LR algorithm from the three accelerometers.

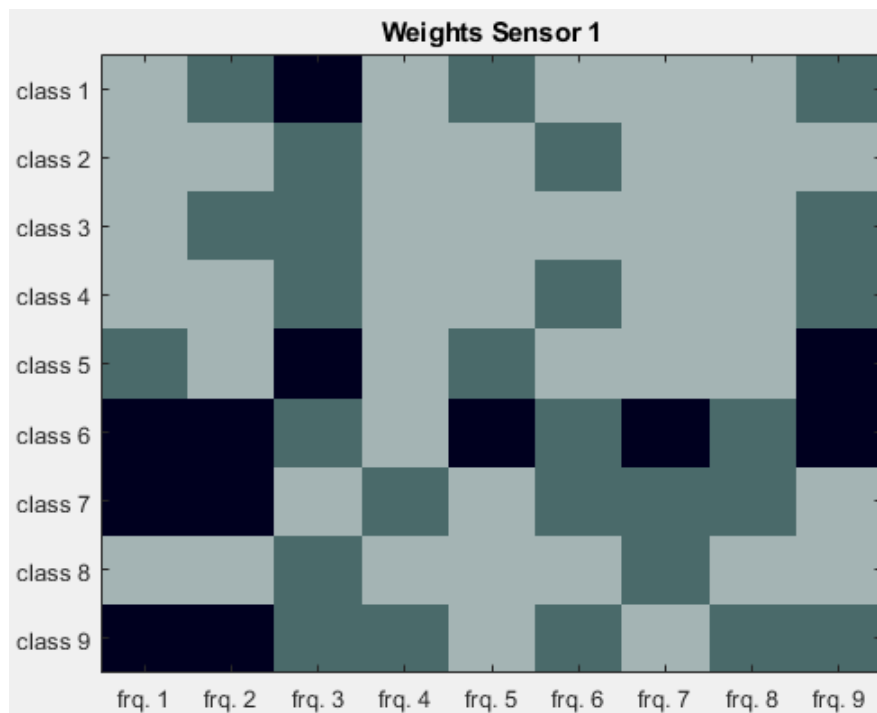


Figure 3.8: Indicators learnt from data coming from sensor 1. The evaluated frequencies in accelerometer 1 are, in ascending order: 80, 610, 710, 1100, 1590, 1730, 2110, 3330 and 3490 Hz.

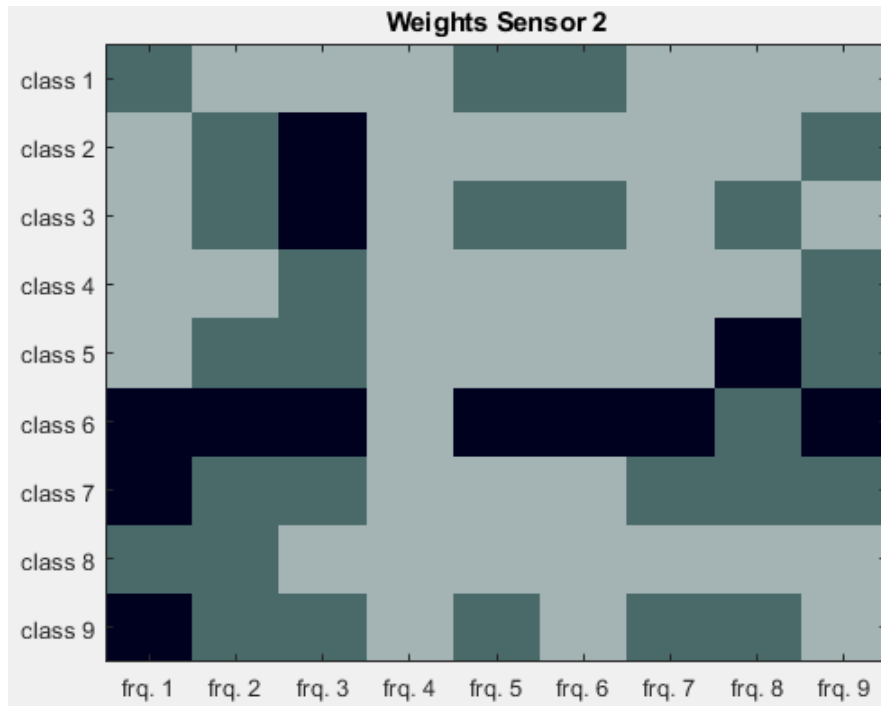


Figure 3.9: Indicators learnt from data coming from sensor 2. The evaluated frequencies in accelerometer 2 are, in ascending order: 80, 360, 610, 1230, 1390, 1590, 2040, 3330 and 3675 Hz.

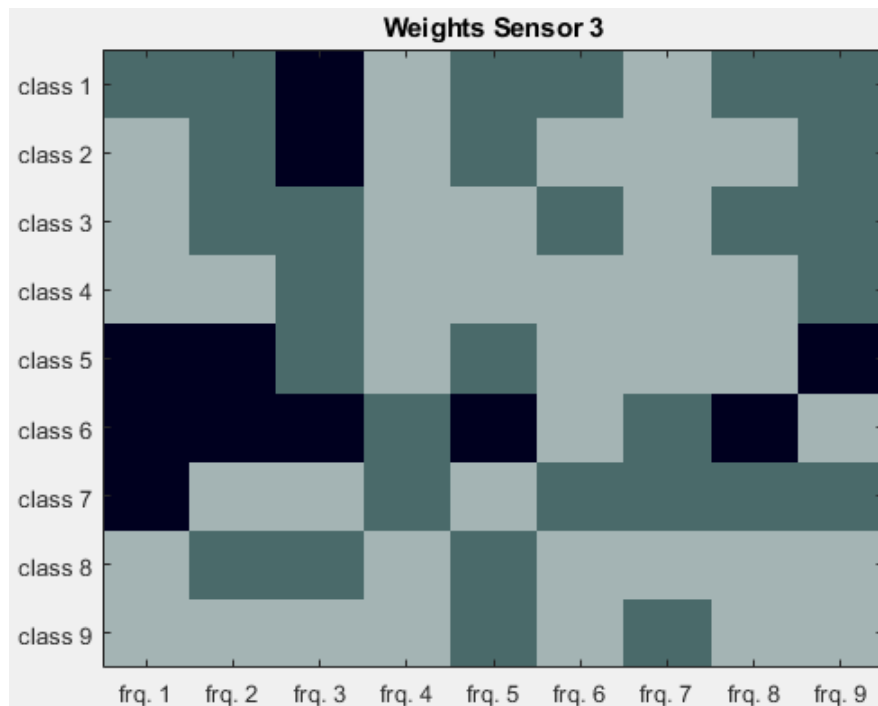


Figure 3.10: Indicators learnt from data coming from sensor 3. The evaluated frequencies in accelerometer 3 are, in ascending order: 80, 610, 710, 1390, 1570, 1970, 2030, 3330 and 3470 Hz.

Finally, it has to be said that, as it was experienced, in the case the feature vector is composed just with data coming from one or two accelerometers, the loss of information is not too large and classification algorithms keep performing with high success rates. This fact should be taken into account in the case a company wants to install this system. It should be studied how many sensors are necessary to achieve the needed accuracy, making sure the economic cost is not greater than necessary.

3.5.1.2.- Different joining plates

In order to check if classification algorithms were also able to distinguish between different joining plates, LR algorithm was trained with the 70% of data acquired on the third day and tested with the remaining 30%. The obtained results were optimal. As it can be seen in table A.11, an accuracy of 100% was achieved. These results are due to the noticeable deviation induced when substituting essential parts of the structure with pieces made of different materials or with different shapes.

3.5.2.- 2D projections

PCA and *T-SNE* have been trained in order to map data examples acquired in the same day from their high dimensional space to a lower dimensional space, which allows the visualization of data and helps to find clusters within the data. *Feature scaling and mean normalization* of the feature matrix have been computed before training the algorithms.

As it was done before in the case of classification, the results will be divided in two groups: clustering normal conditions, loose screw and added weight classes and clustering different joining plate classes.

3.5.2.1.- Normal conditions, loose screws and added weights

2D projections of data obtained by means of *PCA* and *T-SNE* are shown in 3.11 and 3.12, respectively. In the case of *PCA*, 70% of the evaluated data was used to train the *PCA* and the remaining 30% was used to check if new examples fall close to training examples belonging to their same class or not. In the case of *T-SNE*, all the data set has to be provided to train the algorithm. The parameter *perplexity* was set to 20.

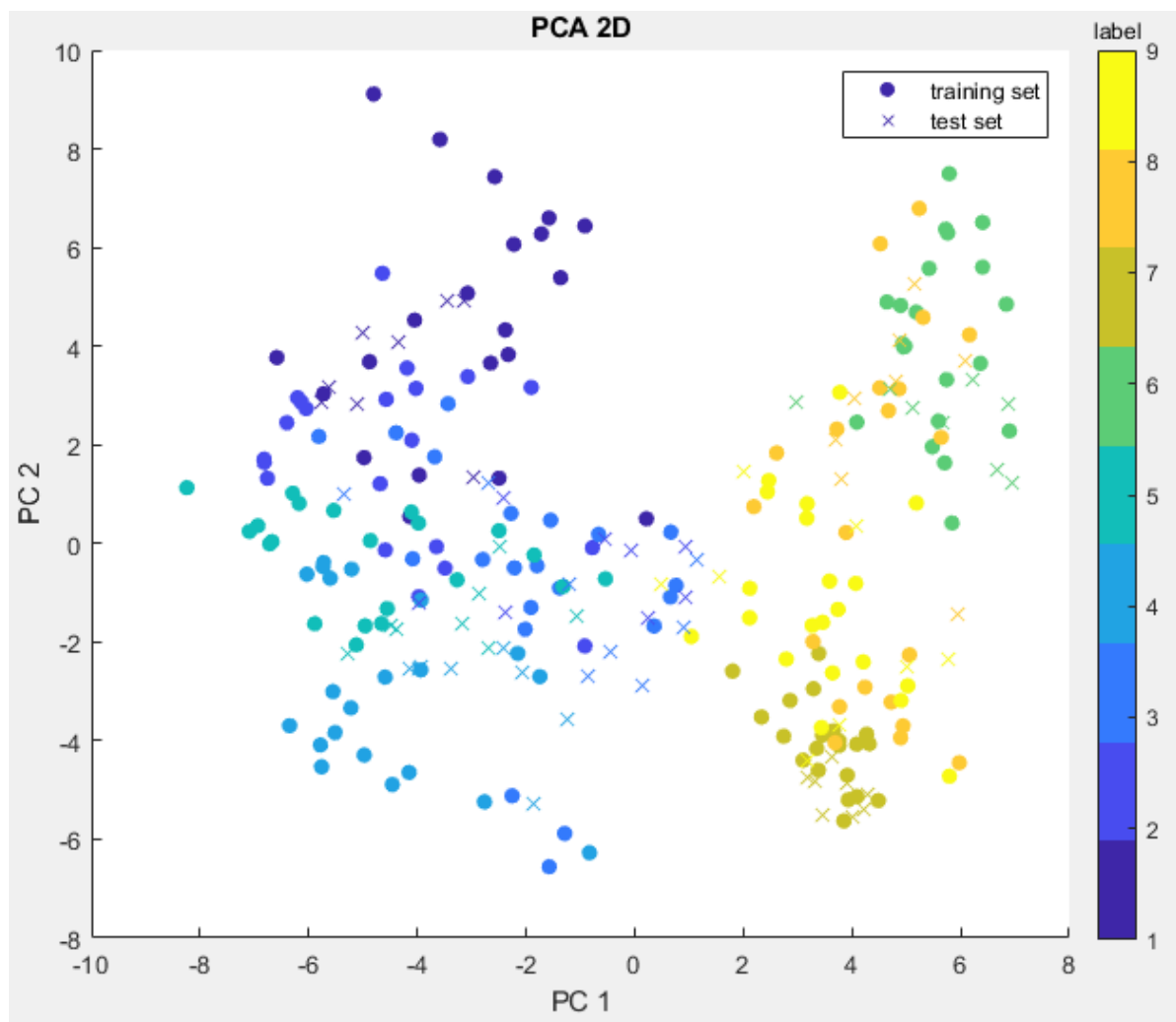


Figure 3.11: Projected data from classes 1 to 9 in 2D by means of PCA. Training set of 70% of the data set and the remaining 30% used as test set.

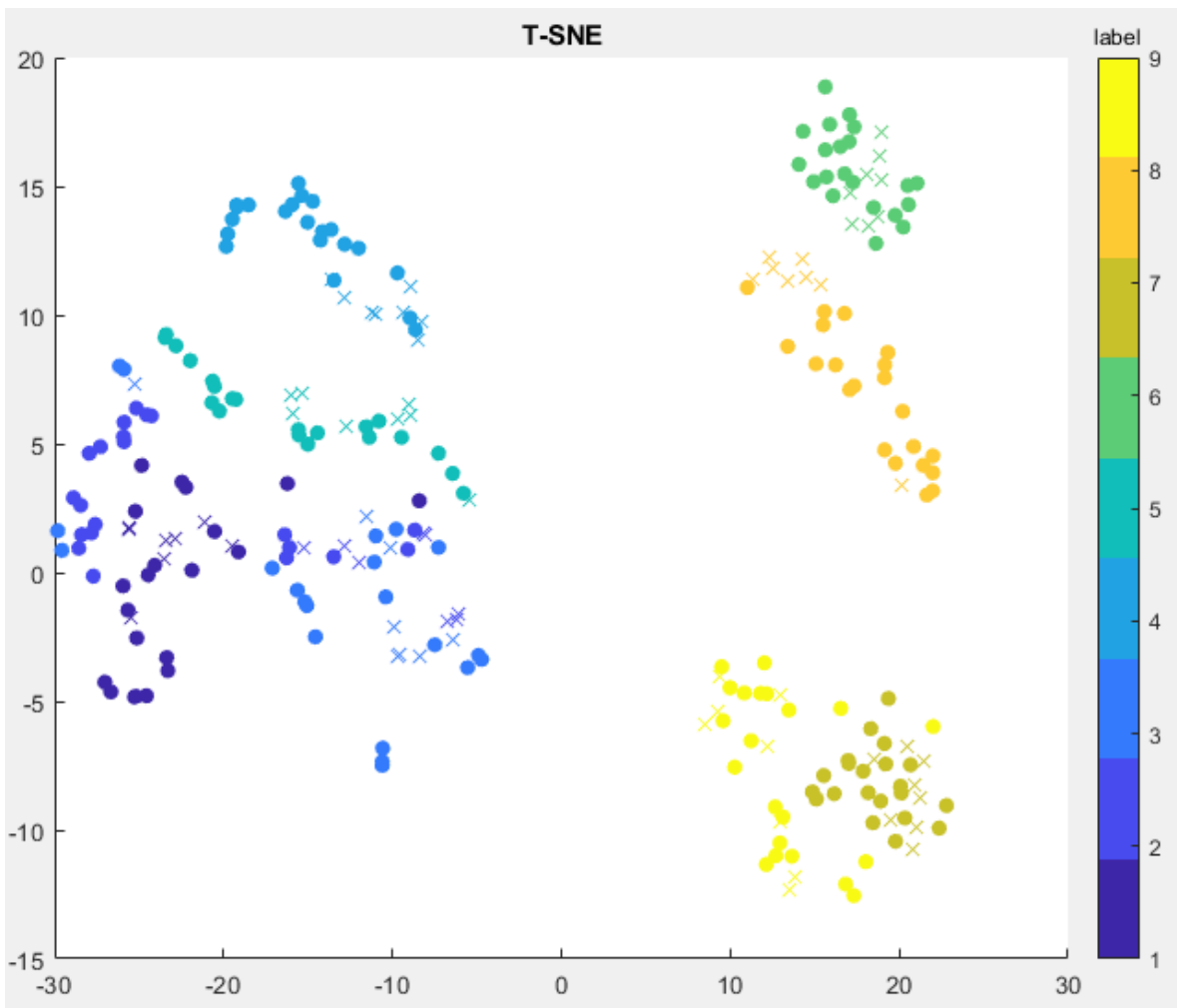


Figure 3.12: Projected data from classes 1 to 9 in 2D by means of T-SNE.

It can be seen that PCA was able to represent data examples that belong to the same class close one to another, but the boundaries between different classes are not very clear. It can also be seen that the tested data was mapped into close positions to the training data from its same class. The poor separation of the classes is due to the fact that the first two principal components just contain the 37.25% of the variance in the data.

T-SNE, in contrast, is able to not only locate data examples belonging to the same class close one to another, but also defines clearer boundaries between different classes. This is due to the fact that T-SNE is able to find non-linearities in the data while PCA algorithm is linear. The same data examples used for testing the PCA are represented with crosses in the T-SNE plot so they can be identified. Anyway, notice that in this case they have been used for training the T-SNE.

3.5.2.2.- Different joining plates

In this section, PCA and T-SNE were trained with data from classes where different joining plates are assembled to the bar. The results can be seen in figures 3.13 and 3.14, respectively. Again, 70% of the evaluated data was used to train the PCA and the remaining 30% was used as test set. The parameter *perplexity* of T-SNE algorithm was set to 20.

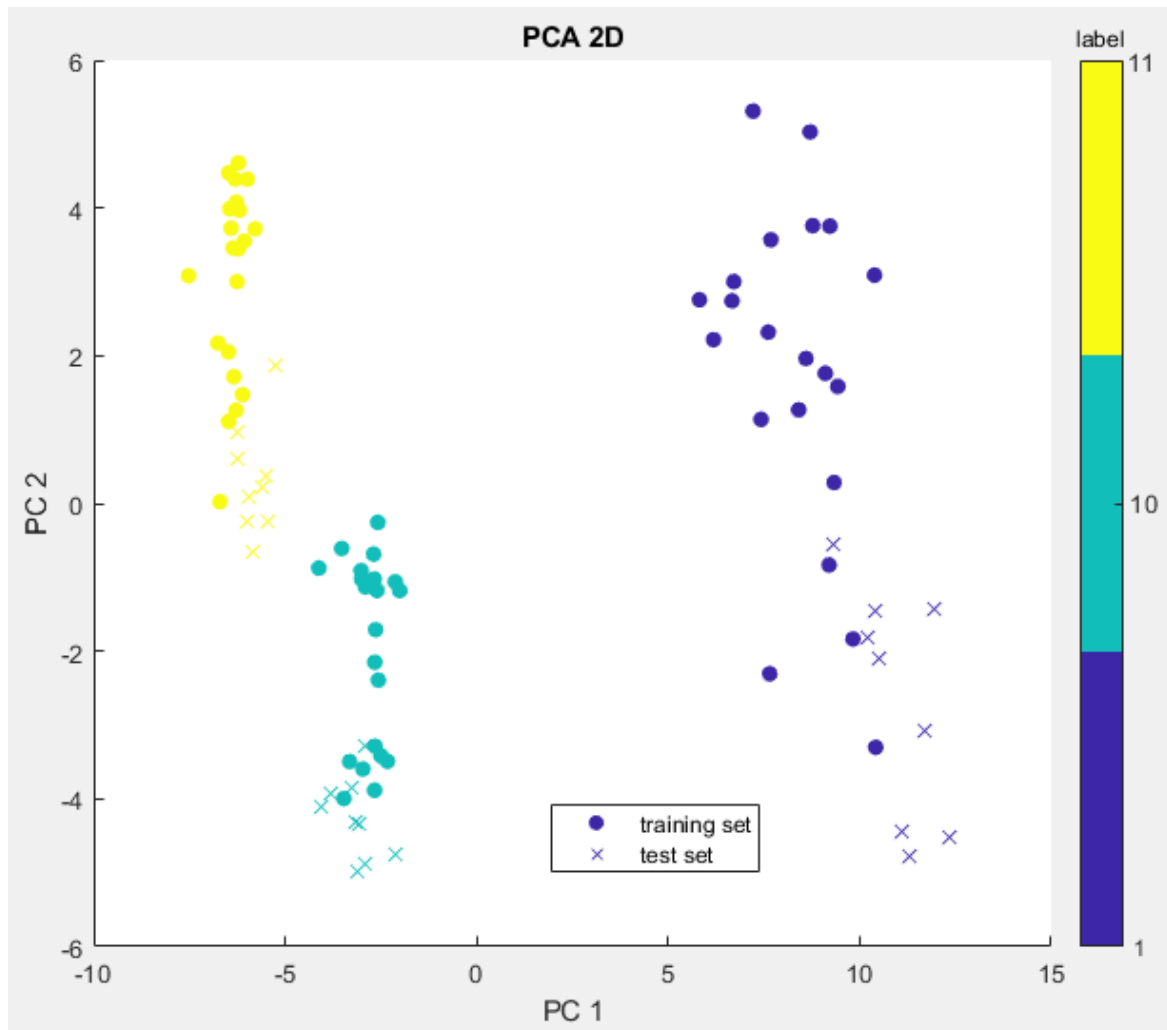


Figure 3.13: Projected data from classes 1 ,10 and 11 in 2D by means of PCA. Training set of 70% of the data set and the remaining 30% used as test set.

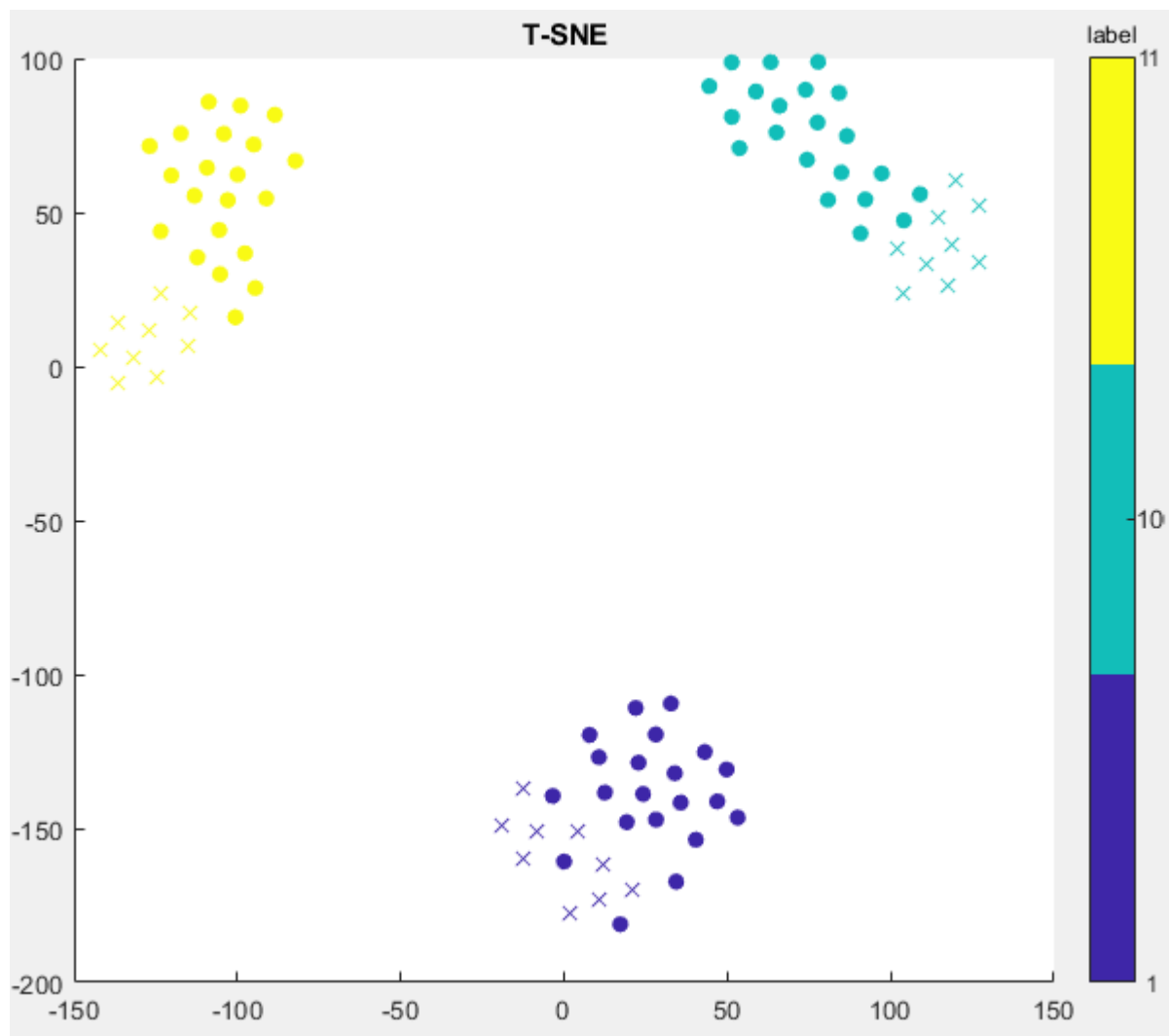


Figure 3.14: Projected data from classes 1, 10 and 11 in 2D by means of T-SNE.

In this case, PCA algorithm is able to perform better than in the 9 classes analysis. This is because the two first principal components contain a 75.85% of the total variance in the evaluated data set and also because of the lower number of classes being evaluated.

T-SNE algorithm is able to perform again better than PCA, not only defining boundaries between classes but also concentrating data examples from the same class in very close locations in the low dimensional space.

CHAPTER



Discussion

After presenting the results of this project: the contributions it has made, an assessment about the obtained results and ideas about future lines of work will be exposed.

4.1.- Contributions

The contributions that this work has brought are the following:

- Review of the SHM bibliography and its potential benefits in industrial processes.
- Automation of a Structural Health Monitoring test bench by means of the high level language software Matlab, that handles the NI-USB 6356 data acquisition board, which improves the old test bench in aspects such as the possibility of increasing drastically the sample rate for the acquisitions or making easier the signal management. The software has been written in a modular way so it can be reused and expanded in the future.
- Structured organization of the acquired data and posterior storage in an intuitive way in ".mat" files, making the subsequent data processing easier.
- Design of a convention for SHM tests with which large repositories can be made and shared in order to contribute to the SHM research.

- Analysis of the possibilities of Machine Learning applied to frequency features in SHM problems. In particular, usefulness of classification and dimensionality reduction algorithms applied to fault detection was proved.

4.2.- Assessment

After analyzing the development of the project and the obtained results, the next ideas can be remarked:

- The design of every aspect of an SHM test bench, ranging from instrumentation and test automation to data processing and diagnostic, makes a considerable contribution.
- The design of a robust test bench like this brings the possibility to do future work without worrying about design defects and in which efforts can focus on improving the applications of Intelligent Data Analysis algorithms in SHM.
- High repeatability in the measurements was achieved in tests performed in the same time frame, partly thanks to the addition of the 3D-printed part on the hammer.
- It has been checked that noise problems can be a real obstacle for the data acquisition and that laboratory conditioning and isolation of the instrumentation play an important role.
- It was experienced that the fact of having a 3D printer allows you to obtain any type of piece with the desired characteristics in a matter of minutes, which brings flexibility, wealth to the information extracted from the experiments and the possibility of modifying mechanical aspects of any element of the system to taste.
- The dependence of the results on the chosen parameters was observed during the feature extraction process, where a proper selection of the frequency bands and time intervals has a crucial role.
- The dependence of the results of the Machine Learning algorithms on the environmental conditions was verified. Clearly, it is an aspect that deserves to be taken into account and it is recommended to collect data from different days, so that the algorithms are able to generalize when they face new situations.

4.3.- Future work

In accordance with the experience accumulated after carrying out this project, the analysis of the tasks that took more time and the motivation found to develop some aspects for which there was not available time, the following future lines of work are proposed:

- Development of an algorithm that automatically optimizes the feature extraction process (center and width of frequency bands, window size, overlapping, etc.) from the available sampled signal, using the prediction error of the applied *Machine Learning* algorithm as a way to evaluate it.
- Modal analysis and damage detection algorithms proposed in this field.
- Test other systems of excitation such as electrodynamic shakers in order to cause different responses on the structure.
- Test other kind of accelerometers which have a higher sensitivity and are less susceptible to noise.
- Further research into the enhancement that 3D printing technologies can bring to SHM test benches design and performance.
- Build a 3D-printed case to make the test bench hardware more compact.
- Integrate the test bench into a cloud data access service in order to contribute to the settlement of industry 4.0.



Confusion matrices

In this appendix, the confusion matrices associated to each of the situations explained in section 3.5.1 will be shown.

A.1.- Confusion matrices: Normal conditions, loose screws and added weights

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	15	4	0	0	2	0	1	1	0
	2	0	23	0	0	0	0	0	0	0
	3	0	8	15	0	0	0	0	0	0
	4	0	0	0	23	0	0	0	0	0
	5	0	2	3	0	18	0	0	0	0
	6	0	4	0	0	0	19	0	0	0
	7	0	0	0	0	0	0	22	0	1
	8	0	0	1	0	0	5	0	14	3
	9	1	0	0	0	0	0	1	0	21

Table A.1: Confusion matrix of logistic regression in *case 1*. LR algorithm trained with data from day one and tested with data from that same day. Training set 20% and test set 80% of the total amount of data. Accuracy of **82.12%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	16	6	0	0	1	0	0	0	0
	2	0	23	0	0	0	0	0	0	0
	3	0	3	18	0	0	0	0	0	2
	4	0	0	0	23	0	0	0	0	0
	5	0	0	0	0	23	0	0	0	0
	6	0	0	0	0	0	23	0	0	0
	7	0	0	0	0	0	0	23	0	0
	8	0	0	1	0	0	5	0	13	4
	9	0	0	0	0	0	0	0	0	23

Table A.2: Confusion matrix of neural networks in *case 1*. NN algorithm trained with data from day one and tested with data from that same day. Training set 20% and test set 80% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of **89.37%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	6	1	10	0	1	0	0	5	1
	2	0	24	0	0	0	0	0	0	0
	3	5	1	18	0	0	0	0	0	0
	4	1	0	3	15	5	0	0	0	0
	5	1	1	4	0	18	0	0	0	0
	6	0	0	1	0	0	22	0	1	0
	7	0	17	0	1	0	0	5	0	1
	8	1	10	0	2	0	1	1	8	1
	9	0	6	0	3	0	0	6	3	6

Table A.3: Confusion matrix of logistic regression in *case 2*. LR algorithm trained with data from day one and tested with data from day 2. Training set 20% and test set 80% of the total amount of data. Accuracy of **56.48%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	7	1	7	0	4	0	1	1	3
	2	0	22	1	0	0	0	0	0	1
	3	4	0	18	1	0	0	0	0	1
	4	1	0	0	18	5	0	0	0	0
	5	0	0	0	0	16	0	0	0	8
	6	0	0	0	0	0	17	0	7	0
	7	0	9	0	0	0	3	11	0	1
	8	1	2	0	0	0	12	0	9	0
	9	0	6	0	0	0	0	5	1	12

Table A.4: Confusion matrix of neural networks in *case 2*. NN algorithm trained with data from day one and tested with data from day 2. Training set 20% and test set 80% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of **60.19%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	8	0	0	0	0	0	0	0	0
	2	0	8	0	0	0	0	0	0	0
	3	0	0	8	0	0	0	0	0	0
	4	0	0	0	8	0	0	0	0	0
	5	0	0	0	0	8	0	0	0	0
	6	0	0	0	0	0	8	0	0	0
	7	0	0	0	0	0	0	8	0	0
	8	0	0	0	0	0	0	0	7	1
	9	0	0	0	0	0	0	0	0	8

Table A.5: Confusion matrix of logistic regression in *case 3*. LR algorithm trained with data from day one and tested with data from that same day. Training set 70% and test set 30% of the total amount of data. Accuracy of **98.61%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	8	0	0	0	0	0	0	0	0
	2	0	8	0	0	0	0	0	0	0
	3	0	0	8	0	0	0	0	0	0
	4	0	0	0	8	0	0	0	0	0
	5	0	0	0	0	8	0	0	0	0
	6	0	0	0	0	0	8	0	0	0
	7	0	0	0	0	0	0	8	0	0
	8	0	0	0	0	0	0	0	8	0
	9	0	0	0	0	0	0	0	0	8

Table A.6: Confusion matrix of neural networks in *case 3*. NN algorithm trained with data from day one and tested with data from that same day. Training set 70% and test set 30% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of **100%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	3	0	2	0	0	0	0	4	0
	2	0	9	0	0	0	0	0	0	0
	3	0	0	9	0	0	0	0	0	0
	4	0	0	0	6	2	0	0	1	0
	5	0	0	0	0	9	0	0	0	0
	6	0	0	0	0	0	9	0	0	0
	7	0	0	0	0	0	0	8	0	1
	8	1	0	0	3	0	1	0	4	0
	9	0	0	0	0	0	0	0	0	9

Table A.7: Confusion matrix of logistic regression in *case 4*. LR algorithm trained with data from day one and tested with data from day 2. Training set 70% and test set 30% of the total amount of data. Accuracy of **81.48%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	2	0	6	0	0	0	0	1	0
	2	0	9	0	0	0	0	0	0	0
	3	0	0	9	0	0	0	0	0	0
	4	0	0	0	9	0	0	0	0	0
	5	0	0	0	0	9	0	0	0	0
	6	0	0	0	0	0	8	0	1	0
	7	0	2	0	0	0	0	7	0	0
	8	0	0	0	0	0	3	0	6	0
	9	0	0	0	0	0	0	0	0	9

Table A.8: Confusion matrix of neural networks in *case 4*. NN algorithm trained with data from day one and tested with data from day 2. Training set 70% and test set 30% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of **83.95%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	17	0	0	0	0	0	0	0	0
	2	0	17	0	0	0	0	0	0	0
	3	0	0	17	0	0	0	0	0	0
	4	0	0	0	17	0	0	0	0	0
	5	0	0	0	0	17	0	0	0	0
	6	0	0	0	0	0	16	0	1	0
	7	0	0	0	0	0	0	17	0	0
	8	0	0	0	0	0	2	0	15	0
	9	0	0	0	0	0	0	0	0	17

Table A.9: Confusion matrix of logistic regression in *case 5*. LR algorithm trained with shuffled data from both days and tested with data from both days too. Training set 70% and test set 30% of the total amount of data. Accuracy of **98.04%**

		Predicted class								
		1	2	3	4	5	6	7	8	9
True class	1	17	0	0	0	0	0	0	0	0
	2	0	17	0	0	0	0	0	0	0
	3	0	0	17	0	0	0	0	0	0
	4	0	0	0	17	0	0	0	0	0
	5	0	0	0	0	17	0	0	0	0
	6	0	0	0	0	0	17	0	0	0
	7	0	0	0	0	0	0	17	0	0
	8	0	0	0	0	0	1	0	16	0
	9	0	0	0	0	0	0	0	0	17

Table A.10: Confusion matrix of neural networks in *case 5*. NN algorithm trained with shuffled data from both days and tested with data from both days too. Training set 70% and test set 30% of the total amount of data. NN with one hidden layer of 200 units. Accuracy of **99.35%**

A.2.- Confusion matrices: Different joining plates

		Predicted class		
		1	10	11
True class	1	9	0	0
	10	0	9	0
	11	0	0	9

Table A.11: Confusion matrix of LR algorithm applied to distinguish between classes 1, 10 and 11. Training set 70% and test set 30% of the total amount of data. Accuracy of **100%**



Abbreviations

- AD** Analog Digital
- AI** Artificial Intelligence
- CAGR** Compound Annual Growth Rate
- CM** Condition Monitoring
- CRISP-DM** Cross Industry Standard Process for Data Mining
- DAQ** Data Acquisition
- DC** Direct Current
- DFT** Discrete Fourier Transform
- FFT** Fast Fourier Transform
- GSDPI** Group of Supervision and Diagnosis of Industrial Processes
- IDA** Intelligent Data Analysis
- LR** Logistic Regression
- ML** Machine Learning
- NDT** Non Destructive Testing
- NN** Neural Networks
- PC** Personal Computer
- PCA** Principal Component Analysis
- PSD** Power Spectral Density
- RMS** Root Mean Square
- SHM** Structural Health Monitoring
- SVD** Singular Value Decomposition
- T-SNE** T-distributed Stochastic Neighbor Embedding

USB Universal Serial Bus

USD United States Dollar



Matlab scripts

The Matlab code used for obtaining the results shown in this document will be presented along this appendix. The code has been classified in DAQ (Data Acquisition), feature extraction and Machine Learning scripts.

D.1.- DAQ scripts

The code used for automating the data acquisition and storage can be seen next.

D.1.1.- DAQ main

```
%% Data Acquisition
clear all;
close all;

ask_sr = 'Type the sample rate in Hz: ';
Sr = input(ask_sr);

ask_n = 'Type the number of hits to be performed: ';
n = input(ask_n);

label1 = ('normal conditions');
```

```
label2 = ('Loose screw (1)');
label3 = ('Loose screw (4)');
label4 = ('Loose screw (5)');
label5 = ('Loose screw (8)');
label6 = ('10Kg weight in the left side');
label7 = ('10Kg weight in the right side');
label8 = ('8Kg weight in the left side');
label9 = ('8Kg weight in the right side');
label10 = ('3D-printed plastic join');
label11 = ('3D-printed plastic join with narrowness');
fprintf(strcat('Possible labels: \n 1-->', label1, '\n 2-->',
    label2, '\n 3-->', label3, '\n 4-->', label4, '\n 5-->', label5
    , '\n 6-->', label6, '\n 7-->', label7, '\n 8-->', label8, '\n
    9-->', label9, '\n 10-->', label10, '\n 11-->', label11, '\n'));
ask_label = 'Type the label of the experiment: ';
label = input(ask_label);

ask_descripcion = 'Write a description for the experiment: ';
descrip = input(ask_descripcion, 's');

s = init_exp(Sr);
out = control_signal(Sr);

message_aq = ['Wait ' num2str(15*n) ' seconds while the test is
    performed...'];
disp(message_aq);

[S1aq, S2aq, S3aq] = gen_aq(n, Sr, out, s);
[S1, S2, S3] = data_preprocess(n, Sr, S1aq, S2aq, S3aq);
save_data(n, Sr, S1, S2, S3, label, descrip);
show_aq(n, Sr, S1, S2, S3);

final_message = ['The test has finished'];
disp(final_message);
```

D.1.2.- Initializing the experiment

```
function s = init_exp(Sr)
% Necessary config to perform experiments with a given sample
```

```
    rate Sr
% Detect connected devices
d = daq.getDevices;
% Create session s
s = daq.createSession('ni');
% Add AI --> accelerometers
s.addAnalogInputChannel('Dev2', 0, 'Voltage');
s.addAnalogInputChannel('Dev2', 2, 'Voltage');
s.addAnalogInputChannel('Dev2', 5, 'Voltage');
% Add AO driver
s.addAnalogOutputChannel('Dev2',0, 'Voltage');
% Sample rate of the experiment
s.Rate = Sr;
end
```

D.1.3.- Control signal

```
function out = control_signal(Sr)
% Square wave close to impulse(200 ms)
% waveform duration: 15 s.
out = 5*(ones(15*Sr,1)); % 5 V, hammer is contracted
out(5*Sr:5.2*Sr) = 2.5; % 2.5 V, hammer hits
end
```

D.1.4.- Data generation and acquisition

```
function [S1aq, S2aq, S3aq] = gen_aq(n,Sr,out,s)
% Generates the control signal and captures the data for n
  experiments
% Stores acquisitions of sensors in matrices S1aq, S2aq and S3aq
S1aq = [];
S2aq = [];
S3aq = [];
for k=1:n
% control signal to output buffer
    if k == n
        out(end)=2.5; % last element of control signal = 2.5,
            so the hammer stops receiving voltage after the last
            hit
    end
end
```

```

else
    out(end)=5;
end

s.queueOutputData([out]);
% Start generation and acquisition in the foreground
[data,time] = s.startForeground();
% Storage of data coming from each sensor (rows=samples,
    columns =no. experiment)
S1aq = [S1aq,data(1:15*Sr,1)];
S2aq = [S2aq,data(1:15*Sr,2)];
S3aq = [S3aq,data(1:15*Sr,3)];
end
end

```

D.1.5.- Data pre-processing

```

function [S1, S2, S3] = data_preprocess(n,Sr,S1aq,S2aq,S3aq)
% The starting instant of vibration is saved in S2_start in order
    to synchronize the different hits and check repeatability
% In SX, data from the accelerometer X is saved, in a way that
    measurements are distributed along rows and hits along columns
    from 0.5 s before the hit until 3.5 s after the hit
S2_start = [];
S1 = [];
S2 = [];
S3 = [];
for i=1:n
    S2_start = [S2_start, find(abs(S2aq(:,i))>0.5,1)];
    S1 = [S1, S1aq(S2_start(i)-0.5*Sr:S2_start(i)+3.5*Sr-1,i)];
    S2 = [S2, S2aq(S2_start(i)-0.5*Sr:S2_start(i)+3.5*Sr-1,i)];
    S3 = [S3, S3aq(S2_start(i)-0.5*Sr:S2_start(i)+3.5*Sr-1,i)];
end
end

```

D.1.6.- Data description file

```

function data_descrip()

```

```
% This function creates the file where the description of each
    experiment will be stored
% This function has to be executed only once, at the beginning of
    all the experiments
categories = {'Number of Exp' 'number of hits' 'Sample Rate' '
    Label' 'Description' 'Timestamp'};
save 'descripcion.mat' categories;
end
```

D.1.7.- Data storage

```
function save_data(n, Sr, S1, S2, S3, label, descrip, tmstamp)
description = matfile('descripcion.mat','Writable',true);
N = size(description.categories,1);
tmstamp = datestr(now);
new_categories = {N n Sr label descrip tmstamp};
description.categories = [description.categories; new_categories
    ];
file = strcat('Exp_',num2str(N),'.mat');
save (file, 'S1', 'S2', 'S3', 'n', 'Sr', 'label', 'descrip', '
    tmstamp');
end
```

D.1.8.- Plotting the acquisitions

```
function show_aq(n, Sr, S1, S2, S3)
% Shows all the measurements after an experiment

time_sync = linspace(-0.5,3.5,4*Sr);
Legend=cell(n,1);
figure;
clf;
for i = 1:n
    subplot(3,1,1)
    hold on
    plot(time_sync',S1(:,i),'color',rand(1,3));
    hold off
    subplot(3,1,2)
    hold on
```

```
    plot(time_sync',S2(:,i),'color',rand(1,3));
    hold off;
    subplot(3,1,3)
    hold on
    plot(time_sync',S3(:,i),'color',rand(1,3));
    hold off;
    Legend{i}=strcat('Exp:', num2str(i));
end

ax1 = subplot(3,1,1);
hold on
ylabel('Voltage 100 mV/g');
xlabel('Time (Secs)');
title('Acquired Signal Sensor 1');
legend(Legend);
hold off

ax2 = subplot(3,1,2);
hold on
ylabel('Voltage 100 mV/g');
xlabel('Time (Secs)');
title('Acquired Signal Sensor 2');
legend(Legend);
hold off

ax3 = subplot(3,1,3);
hold on
ylabel('Voltage 100 mV/g');
xlabel('Time (Secs)');
title('Acquired Signal Sensor 3');
legend(Legend);
hold off

linkaxes([ax1 ax2 ax3], 'xy')
```

D.2.- Feature extraction scripts

The scripts used in the feature extraction process are presented below.

D.2.1.- Feature extraction main and repeatability visualization

```
%% This script takes the collection of tests and computes the
    feature extraction
% Sr has to be the same for every different training subset
clear all;
% load captures from 9 different classes
data{1} = load('Exp_15.mat');
data{2} = load('Exp_16.mat');
data{3} = load('Exp_17.mat');
data{4} = load('Exp_18.mat');
data{5} = load('Exp_19.mat');
data{6} = load('Exp_20.mat');
data{7} = load('Exp_21.mat');
data{8} = load('Exp_22.mat');
data{9} = load('Exp_23.mat');

n = data{1}.n;           %number of hits inside each label
Sr = data{1}.Sr;        %sample rate

%% Y (Output)
num_labels = 9;
Y = zeros(num_labels*n,1);
k=1;
for i=1:num_labels
    for j=1:n
        Y(k) = data{i}.label;
        k = k+1;
    end
end

%% X. Input. feature extraction.
% 9 freq bands x 3 sensors x 3 time intervals. X vector of 81
    elements for each example
X = [];                 %Feature vector

%frequencies to be analyzed in each sensor
freq = [80 610 710 1100 1590 1730 2110 3330 3490;
        80 360 610 1230 1390 1590 2040 3330 3675;
```

```

80 610 710 1390 1570 1970 2030 3330 3470];

N=4000;                                %window size
k = 1;
%i = number of labels  e = number of hits belonging to each label
for i = 1:num_labels
    for e=1:n
        [x, f] = feat_ex(data{i},e,N,freq);
        X = [X; x];
        k = k+1;
    end
end

%% saving data (not normalized)
file = strcat('Data_joints_0.4_0.5_0.6_0.8.mat');
save (file, 'X', 'Y', 'n', 'Sr', 'freq', 'W' );

%% Repetibility test with feature scaling and mean normalization
im = {};
[X_norm, mu, sigma] = featureNormalize(X);
k=1;
for i = 1:num_labels
    for w=1:n
        im{k} = reshape(X_norm((i-1)*n+w,:),9,9);
        k = k+1;
    end
end

for i=1:num_labels
figure;
for w = 1:n
    subplot(6,5,w)
    image(im{(i-1)*n+w},'CDataMapping','scaled')
    title(strcat('label: ',num2str(i),' strike: ',num2str(w)));
    colorbar
    caxis([-3 3])
end
end

```


D.2.2.- Feature extraction function

```
function [X] = feat_ex(data, e, N, freq)
% This function performs the feature extraction of a given sample
  test.
Sr = data.Sr;
S1 = data.S1(:,e);
S2 = data.S2(:,e);
S3 = data.S3(:,e);
tm = 1/Sr;
delta = 0.1*N; % 90% overlapping
Srf = Sr/delta; % feature sample rate
Q = size(S1,1);
time = linspace(-0.5,3.5,4*Sr); %time vector
f = 0:(Sr/N):(Sr/N)*(N-1); % Frequency vector
marg = 20; %20 Hz margin
%time intervals to pay attention to
t1 = 0.4;
t2 = 0.5;
t3 = 0.6;
t4 = 0.8;
idx = {};
for j = 1:3
for i = 1:size(freq,2)
    idx{j,i} = find((f>(freq(j,i)-marg)) & (f<(freq(j,i)+marg)));
end
end

%% Feature extraction: j for each sensor, k for each window (new
  sample) and h for each feature (band)
F = {};
S = {S1 S2 S3};
for j=1:3
    i=1;
    for k=N:delta:Q
        yv = S{j}(k-N+1:k);
        Y = abs(fft(yv));
        for h = 1:size(freq,2)
            F{j}(i,h) = sqrt(sum(2*Y(idx{j,h}).^2))/N;
```

```
        end
        i = i+1;
    end
end

%% rms average in the time intervals: j for each sensor, i for
    each freq band
R = {};
for j=1:3
    for i=1:size(freq,2)
        R{i,j} = [mean(F{j}(uint16(t1*Srf):uint16(t2*Srf-1),i)),
                mean(F{j}(uint16(t2*Srf):uint16(t3*Srf-1),i)), mean(F{j}
                )(uint16(t3*Srf):uint16(t4*Srf-1),i))];
    end
end
I = cell2mat(R);
X = (I(:))';
end
```

D.3.- Machine Learning scripts

The Matlab scripts used to implement Machine Learning algorithms are shown in this section. Some common functions to several algorithms, Logistic Regression scripts, Neural Networks scripts and PCA and T-SNE scripts will be presented.

D.3.1.- Common functions

D.3.1.1.- Feature normalization

```
function [X_norm, mu, sigma] = featureNormalize(X)
%FEATURENORMALIZE Normalizes the features in X
X_norm = [];
mu = zeros(1, size(X, 2));
sigma = zeros(1, size(X, 2));
for i=1:size(X,2)
    mu(i) = mean(X(:,i));
    sigma(i) = std(X(:,i));
```

```
X_norm = [X_norm, (X(:,i) - mu(i))./sigma(i)];  
end  
end
```

D.3.1.2.- Shuffling and getting training and test set

```
function [X,y,Xtest,ytest] = train_test(X,Y,n1,n2,num_labels)  
% training set and test set  
% n1 is the number of examples per class and n2 is the number of  
%   examples per class that are used as training set. the rest will  
%   be used for the test set.  
% this function choses the training set and test set randomly  
%   from the examples of each class  
%num labels is the number of different classes  
  
Xrand = [];  
for i = 1:num_labels  
    rps = randperm(n1)+n1*(i-1);  
    Xrand = [Xrand;X(rps,:)];  
end  
  
aux = []; aux_t = []; auy = []; auy_t = [];  
for i=1:num_labels  
    aux = [aux;X((i-1)*n1+(1:n2),:)];  
    aux_t = [aux_t;X((i-1)*n1+(n2+1:n1),:)];  
    auy = [auy;Y((i-1)*n1+(1:n2),:)];  
    auy_t = [auy_t;Y((i-1)*n1+(n2+1:n1),:)];  
end  
X = aux;  
y = auy;  
Xtest = aux_t;  
ytest = auy_t;  
end
```

D.3.1.3.- Sigmoid function

```
function g = sigmoid(z)  
%Computes the sigmoid of z.  
g = 1.0 ./ (1.0 + exp(-z));
```

end

D.3.1.4.- Sigmoid gradient

```
function g = sigmoidGradient(z)
%SIGMOIDGRADIENT returns the gradient of the sigmoid function
    evaluated at z
g = zeros(size(z));
g = sigmoid(z).*(1-sigmoid(z));
end
```

D.3.2.- Logistic Regression scripts

D.3.2.1.- Logistic regression main and visualization of the weights

```
%% LOGISTIC REGRESSION
%% Initialization
clear; clc; close all;
num_labels = 9;
n_ex_class = 30;

% load data set 1
data{1} = load('Data_N4000_0.4_0.5_0.6_0.8_indfreqs3.mat'); %
    data stored in arrays X, y
X = data{1}.X(:,:,:); % in case i just want to use info from one
    sensor
Y = data{1}.Y;
[X_norm, mu, sigma] = featureNormalize(X);

% load dataset 2
data{2} = load ('Data2_N4000_0.4_0.5_0.6_0.8_indfreqs3.mat')
X_b = data{2}.X(:,:,:);
y_b = data{2}.Y;
n2 = data{2}.n
[Xtest_b, mu_b, sigma_b] = featureNormalize(X_b);

%% training set y test set
[Xtrain_a, Ytrain_a, Xtest_a, Ytest_a] = train_test(X_norm, Y,
    n_ex_class, 21, num_labels);
```

```
[Xtrain_b, Ytrain_b, Xtest_b, Ytest_b] = train_test(Xtest_b, y_b, n2
    , 21, num_labels);
%m = size(Xtrain, 1);
Xtrain = [Xtrain_a; Xtrain_b];
Ytrain = [Ytrain_a; Ytrain_b];
Xtest = [Xtest_a; Xtest_b];
Ytest = [Ytest_a; Ytest_b];

%% One-vs-All Training
lambda = 0.0;
[all_theta] = oneVsAll(Xtrain, Ytrain, num_labels, lambda);
%[all_theta] = oneVsAll([Xtrain(1:10,:); Xtest_b(1:3,:)], [ytrain
    (1:10); ytest_b(1:3)], num_labels, lambda);

% Prediction A
pred = predictOneVsAll(all_theta, Xtest);
fprintf('\nTest Set A Accuracy: %f\n', mean(double(pred == Ytest)
    ) * 100);
CM = confusionmat(Ytest, pred)

% Prediction B
pred_b = predictOneVsAll(all_theta, Xtest_b);
fprintf('\nTraining Set B Accuracy: %f\n', mean(double(pred_b ==
    Ytest_b)) * 100);
CM_b = confusionmat(Ytest_b, pred_b)

%% Visualization of the Weights
S1_theta = abs(all_theta(:, 2:28));
S2_theta = abs(all_theta(:, 29:55));
S3_theta = abs(all_theta(:, 56:82));
S1_theta_avg = []; S2_theta_avg = []; S3_theta_avg = [];
for i = 0:9-1
S1_theta_avg = [S1_theta_avg, mean(S1_theta(:, (i*3+1):(i*3+3)), 2)
    ];
S2_theta_avg = [S2_theta_avg, mean(S2_theta(:, (i*3+1):(i*3+3)), 2)
    ];
S3_theta_avg = [S3_theta_avg, mean(S3_theta(:, (i*3+1):(i*3+3)), 2)
    ];
```

```

end
maximum = max(max(S2_theta_avg));
minimum = min(min(S2_theta_avg));
classes = {'class 1' 'class 2' 'class 3' 'class 4' 'class 5' '
          class 6' 'class 7' 'class 8' 'class 9'};
frequencies = {'frq. 1' 'frq. 2' 'frq. 3' 'frq. 4' 'frq. 5' 'frq.
              6' 'frq. 7' 'frq. 8' 'frq. 9'};
mymap = flip(bone(4));

figure; image(S1_theta_avg+(maximum-minimum)/3, 'CDataMapping', '
          scaled'); colormap(mymap); caxis([minimum maximum]);
%figure; imagesc(S1_theta_avg>2.5); colormap(mymap);
title('Weights Sensor 1')
set(gca, 'Ytick', [1:9], 'YTickLabel', classes);
set(gca, 'Xtick', [1:9], 'XTickLabel', frequencies);

figure; image(S2_theta_avg+(maximum-minimum)/3, 'CDataMapping', '
          scaled'); colormap(mymap); caxis([minimum maximum]);
%figure; imagesc(S2_theta_avg>2.5); colormap(mymap);
title('Weights Sensor 2')
set(gca, 'Ytick', [1:9], 'YTickLabel', classes);
set(gca, 'Xtick', [1:9], 'XTickLabel', frequencies);

figure; image(S3_theta_avg+(maximum-minimum)/3, 'CDataMapping', '
          scaled'); colormap(mymap); caxis([minimum maximum]);
%figure; imagesc(S3_theta_avg>2.5); colormap(mymap);
title('Weights Sensor 3')
set(gca, 'Ytick', [1:9], 'YTickLabel', classes);
set(gca, 'Xtick', [1:9], 'XTickLabel', frequencies);

```

D.3.2.2.- Logistic regression cost function

```

function [J, grad] = lrCostFunction(theta, X, y, lambda)
%LRCOSTFUNCTION Compute cost and gradient for logistic regression
    with regularization
m = length(y); % number of training examples
J = 0;
grad = zeros(size(theta));
h_theta = sigmoid(X*theta);

```

```
J = (1/m)*sum((-y)'*log(h_theta)-(1-y)'*log(1-h_theta))+lambda
    /(2*m)*sum((theta(2:end)).^2);
grad(1) = (1/m).*X(:,1)'*(h_theta-y);
grad(2:end) = (1/m).*X(:,2:end)'*(h_theta-y)+(lambda/m)*theta(2:
    end);
grad = grad(:);
end
```

D.3.2.3.- One vs all gradient descent

```
function [all_theta] = oneVsAll(X, y, num_labels, lambda)
%ONEVSALL trains multiple logistic regression classifiers and
    returns all the classifiers in a matrix all_theta, where the i-
    th row of all_theta corresponds to the classifier for label i
m = size(X, 1);
n = size(X, 2);
all_theta = zeros(num_labels, n + 1);
% Add ones to the X data matrix
X = [ones(m, 1) X];

options = optimset('GradObj', 'on', 'MaxIter', 50);
% fmincg is an optimizer that will perform the gradient descent
    provided: the cost function, the initial theta and the maximum
    number of iterations
for c = 1:num_labels
    initial_theta = zeros(n + 1, 1);
    [theta] = ...
        fmincg (@(t)(lrCostFunction(t, X, (y == c), lambda)), ...
            initial_theta, options);
    all_theta(c,:) = theta;
end
end
```

D.3.2.4.- Predict one vs all

```
function p = predictOneVsAll(all_theta, X)
%PREDICT Predict the label for a trained one-vs-all classifier.
    The labels are in the range 1..K, where K = size(all_theta, 1).
m = size(X, 1);
```

```
num_labels = size(all_theta, 1);  
p = zeros(size(X, 1), 1);  
% Add ones to the X data matrix  
X = [ones(m, 1) X];  
hs_theta = sigmoid(X*all_theta');  
[M,p] = max(hs_theta, [], 2);  
end
```

D.3.3.- Neural Networks scripts

D.3.3.1.- Neural Networks main

```
%% Neural Networks  
%% Initialization  
clear ; close all; clc  
input_layer_size =81; % 20x20 Input Images of Digits % feature  
^4  
hidden_layer_size = 200; % 200 hidden units  
num_labels = 9; % 9 labels  
  
%% Loading and normalizing data  
data{1} = load('Data_N4000_0.4_0.5_0.6_0.8_indfreqs3.mat');  
X = data{1}.X(:,:,:);  
n1 = data{1}.n;  
Y_a = data{1}.Y;  
[X_norm, mu, sigma] = featureNormalize(X);  
  
%% load data from another day  
data{2} = load ('Data2_N4000_0.4_0.5_0.6_0.8_indfreqs3.mat');  
X_b = data{2}.X;  
y_b = data{2}.Y;  
n2 = data{2}.n;  
[Xtest_b, mu_b, sigma_b] = featureNormalize(X_b);  
  
%% training set and test set  
[Xtrain_a,Ytrain_a,Xtest_a,Ytest_a] = train_test(X_norm,Y_a,n1  
,21,num_labels);  
[Xtrain_b,Ytrain_b,Xtest_b,Ytest_b] = train_test(Xtest_b,y_b,n2  
,21,num_labels);
```



```
Xtrain = [Xtrain_a; Xtrain_b];
Ytrain = [Ytrain_a; Ytrain_b];
Xtest = [Xtest_a; Xtest_b];
Ytest = [Ytest_a; Ytest_b];
m = size(Xtrain, 1);

%% Initializing Parameters
initial_Theta1 = randInitializeWeights(input_layer_size,
    hidden_layer_size);
initial_Theta2 = randInitializeWeights(hidden_layer_size,
    num_labels);

% Unroll parameters
initial_nn_params = [initial_Theta1(:) ; initial_Theta2(:)];

%% Training NN
options = optimset('MaxIter', 550);
lambda = 0.00;
costFunction = @(p) nnCostFunction(p, ...
    input_layer_size, ...
    hidden_layer_size, ...
    num_labels, Xtrain_a, Ytrain_a
    , lambda);
[nn_params, cost] = fmincg(costFunction, initial_nn_params,
    options);

% Obtain Theta1 and Theta2 back from nn_params
Theta1 = reshape(nn_params(1:hidden_layer_size * (
    input_layer_size + 1)), ...
    hidden_layer_size, (input_layer_size + 1));
Theta2 = reshape(nn_params((1 + (hidden_layer_size * (
    input_layer_size + 1))):end), ...
    num_labels, (hidden_layer_size + 1));

%% Predictions
%Prediction A
pred = predict(Theta1, Theta2, Xtest_a);
fprintf('\nTraining Set A Accuracy: %f\n', mean(double(pred ==
```

```
Ytest_a)) * 100);
CM = confusionmat(Ytest_a,pred)

% Prediction B
pred_b = predict(Theta1, Theta2, Xtest_b);
fprintf('\nTest Set B Accuracy: %f\n', mean(double(pred_b ==
    Ytest_b)) * 100);
CM_b = confusionmat(Ytest_b,pred_b)
```

D.3.3.2.- NN random initialization

```
function W = randInitializeWeights(L_in, L_out)
%RANDINITIALIZEWEIGHTS Randomly initialize the weights of a layer
    with L_in incoming connections and L_out outgoing connections
W = zeros(L_out, 1 + L_in);
E = 0.12;
W = rand(L_out, L_in+1)*(2*E)-E;
end
```

D.3.3.3.- NN cost function

```
function [J grad] = nnCostFunction(nn_params, ...
    input_layer_size, ...
    hidden_layer_size, ...
    num_labels, ...
    X, y, lambda)
%NNCOSTFUNCTION Implements the neural network cost function for a
    two layer neural network which performs classification

% Reshaping nn_params back into the parameters Theta1 and Theta2,
    the weight matrices for our 2 layer neural network
Theta1 = reshape(nn_params(1:hidden_layer_size * (
    input_layer_size + 1)), ...
    hidden_layer_size, (input_layer_size + 1));
Theta2 = reshape(nn_params((1 + (hidden_layer_size * (
    input_layer_size + 1))):end), ...
    num_labels, (hidden_layer_size + 1));

m = size(X, 1);
```

```

X = [ones(m,1) X];
hyp = zeros(m,num_labels);
yk = zeros(m,num_labels);
for i=1:m
    yk(i,y(i)) = 1;
end

J = 0;
A1 = zeros(size(Theta1));
A2 = zeros(size(Theta2));
Theta1_grad = zeros(size(Theta1));
Theta2_grad = zeros(size(Theta2));

for i = 1:m
% forward propagation
a1 = X(i,:)';
z2 = Theta1 * a1;
a2 = sigmoid(z2);
a2 = [1;a2];
z3 = Theta2 * a2;
hyp(i,:) = sigmoid(z3);
% backprop
delta3 = (hyp(i,:)-yk(i,:))';
delta2 = Theta2'*delta3.*sigmoidGradient([0;z2]); % add a term
    in z that will be discarded after
delta2 = delta2(2:end); %avoid first term
% accumulate gradients from the different experiments
A1 = A1 + delta2*a1';
A2 = A2 + delta3*a2';
end
% divide by m to get the gradient of the NN
Theta1_grad = (1/m)*A1;
Theta2_grad = (1/m)*A2;
% adding regularization term
Theta1_grad(:,2:end) = Theta1_grad(:,2:end) + (lambda/m)*Theta1
    (:,2:end);
Theta2_grad(:,2:end) = Theta2_grad(:,2:end) + (lambda/m)*Theta2
    (:,2:end);

```

```

for i = 1:m
    for k = 1:num_labels
J = J + (1/m)*(-yk(i,k)*log(hyp(i,k))-(1-yk(i,k))*log(1-hyp(i,k))
    );
    end
end

for j=1:hidden_layer_size
    for k = 2:input_layer_size+1
        J = J + (lambda/(2*m))*(Theta1(j,k)^2);
    end
end

for j=1:num_labels
    for k = 2:hidden_layer_size+1
        J = J + (lambda/(2*m))*(Theta2(j,k)^2);
    end
end

% Unroll gradients
grad = [Theta1_grad(:) ; Theta2_grad(:)];

end

```

D.3.3.4.- Prediction

```

function p = predict(Theta1, Theta2, X)
%Predicts the label of an input given a trained neural network
m = size(X, 1);
num_labels = size(Theta2, 1);
p = zeros(size(X, 1), 1);
h1 = sigmoid([ones(m, 1) X] * Theta1');
h2 = sigmoid([ones(m, 1) h1] * Theta2');
[dummy, p] = max(h2, [], 2);
end

```

D.3.4.- PCA and T-SNE scripts

D.3.4.1.- 2D projections main

```
%% PCA and T-SNE
clear all
close all
data{1} = load ('Data_N4000_0.4_0.5_0.6_0.8_indfreqs3.mat')
X_a = data{1}.X(:, :);
Y_a = data{1}.Y(:);
n1 = data{1}.n;
num_labels = 9;
[X_a_norm, mu_a, sigma_a] = featureNormalize(X_a);

%% training set y test set
[Xtrain_a, Ytrain_a, Xtest_a, Ytest_a] = train_test(X_a_norm, Y_a, n1
    , 21, num_labels);

% Project the data onto K = 3 dimension
[U, S, V] = pca_1(Xtrain_a);
K = 3;
Z1 = projectData(Xtrain_a, U, K);
Z2 = projectData(Xtest_a, U, K);

% PCA scatter 2D
figure;
scatter(Z1(:, 1), Z1(:, 2), 40, Ytrain_a, 'filled');
hold on
scatter(Z2(:, 1), Z2(:, 2), 60, Ytest_a, 'x');
colormap(parula(9));
hc = colorbar;
set(get(hc, 'title'), 'string', 'label');
title('PCA 2D')
xlabel('PC 1')
ylabel('PC 2')
legend;

%% T-SNE
a = tsne([Xtrain_a; Xtest_a], 'perplexity', 20);
```

```
figure;  
title('T-SNE')  
hold on  
f = size(Xtrain_a,1);  
scatter(a(1:f,1),a(1:f,2),40,Ytrain_a,'filled');  
scatter(a(f+1:end,1),a(f+1:end,2),60,Ytest_a,'x');  
colormap(parula(9));  
hc = colorbar;  
set(get(hc,'title'),'string','label');
```

D.3.4.2.- PCA

```
function [U, S, V] = pca_1(X)  
%PCA Run principal component analysis on the dataset X  
[m, n] = size(X);  
U = zeros(n);  
S = zeros(n);  
Sigma = (1/m)*X'*X;  
[U, S, V] = svd(Sigma);  
end
```

D.3.4.3.- PCA projection

```
function Z = projectData(X, U, K)  
%PROJECTDATA Computes the reduced data representation when  
    projecting only on to the top k eigenvectors  
Z = zeros(size(X, 1), K);  
Z = X * U(:,1:K);  
end
```

Bibliography

- [1] *Mathworks tutorials for Matlab DAQ Toolbox.*
<https://www.mathworks.com/products/daq.html>.
[Online. Checked on November, 2018].
- [2] *Matlab for student use.*
https://mathworks.com/academia/student_version.html.
[Online. Checked on June, 2019].
- [3] *Principal Component Analysis, Wikipedia.*
https://en.wikipedia.org/wiki/Principal_component_analysis.
[Online. Checked on April, 2019].
- [4] *Understanding Feature Engineering (Part 1).*
<https://bit.ly/2rLYtEM>.
[Online. Checked on June, 2019].
- [5] *Andrew Ng Machine Learning course available in coursera, Stanford University.*
<https://www.coursera.org/learn/machine-learning/home/info>.
[Online. Checked on April, 2019].
- [6] *T-SNE, Wikipedia.*
https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding.
[Online. Checked on June, 2019].
- [7] *Dive into Deep Learning by Aston Zhang et al..*
https://www.d2l.ai/chapter_multilayer-perceptrons/backprop.html.
[Online. Checked on June, 2019].
- [8] *Structural Health Monitoring Market Size Worth \$4.34 Billion by 2025: Grand View Research, Inc.*
<https://prn.to/31kNnEJ>.
[Online. Checked on May 2019].

- [9] *Group of Supervision and Diagnosis of Industrial Processes (GSDPI)*, University of Oviedo.
<http://isa.uniovi.es/GSDPI/>.
[Online. Checked on May 2019].
- [10] *State-of-the-art and new developments in the field of structural health monitoring*.
<https://bit.ly/2wMjMpB>.
[Online. Checked on May 2019].
- [11] *USB 6356, National Instruments*.
<http://www.ni.com/es-es/support/model.usb-6356.html>.
[Online. Checked on May 2019].
- [12] *X Series User Manual, DAQ X Series*, NI 632x/634x/635x/636x/637x Devices, National Instruments, no. August, 2018.
- [13] *NI 6356 Device Specifications, S Series Data Acquisition*, National Instruments, June, 2016.
- [14] *Devantech MD22 24V 5A Dual H-Bridge Motor Driver*, Robot Electronics, 2014.
- [15] *Models 786F General purpose , integral cable accelerometer*, Wilcoxon Research, 98504 Rev.C.3 12/07.
- [16] *Three Channel Power Unit Model P703B Operating Instructions, Product Features and Connections*, Wilcoxon Sensing Technologies, 98446 Rev.I.2 09/18.
- [17] DAWSON, BRYAN, *A Vibration condition monitoring techniques for rotating machinery*, The Shock and Vibration Digest, London, SpringerLink, 1976.
- [18] WIRTH, RÜDIGER; HIPPE, JOCHEN, *CRISP-DM: Towards a standard process model for data mining*, in Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining. Citeseer, 2000. p. 29-39.
- [19] BERRY, MICHAEL J.; LINOFF, GORDON, *Data Mining Techniques: For Marketing, Sales, and Customer Support*, John Wiley and Sons, Inc. New York, NY, USA 1997.
- [20] SOHN, H., FARRAR, C. R., HEMEZ, F. M., SHUNK, D. D., STINEMATOS, D. W., NADLER, B. R., AND CZARNECKI, J. J., *A review of structural health monitoring literature: 1996–2001*, Los Alamos National Laboratory, USA, 2003.
- [21] CUADRADO, ABEL AND DIAZ IGNACIO, *Basic Principles of Digital Signal Processing*, slides of Supervision and Control Processes subject, University of Oviedo, Spain, 2019.
- [22] CUADRADO, ABEL AND DIAZ IGNACIO, *Fourier Analysis*, slides of Supervision and Control Processes subject, University of Oviedo, Spain, 2019.

- [23] CUADRADO, ABEL AND DIAZ IGNACIO, *Exercise: Extraction of frequential features from a rolling mill motor*, slides of Supervision and Control Processes subject, University of Oviedo, Spain, 2019.
- [24] PÉREZ LÓPEZ, DANIEL; GARCÍA PÉREZ, DIEGO; CUADRADO VEGA, ABEL ALBERTO AND DÍAZ BLANCO IGNACIO, *Bancada para análisis inteligente de datos en monitorización de salud estructural*, University of Oviedo, Spain, 2017.
- [25] MALEKZADEH, M.; GUL, M.; KWON, I. AND CATBAS, N., *An integrated approach for Structural Health Monitoring using an in-house built fiber-optic system and non-parametric data analysis*, Smart Structures and Systems 14(5):917-942, 2014.
- [26] SPECKMANN, H.; HENRICH, R., *Structural Health Monitoring (SHM) - Overview on Airbus activities*, Airbus, Bremen, Germany, 2004.
- [27] FRIENDLY, M., *Milestones in the history of thematic cartography, statistical graphics, and data visualization*, York University, 2009.
- [28] RYTER, A., *Vibrational based inspection of civil engineering structures*, PhD thesis, Dept. of Building Technology and Structural Engineering, Aalborg University, 1993.
- [29] D. E. BENTLY AND T. HATCH'CHARLES, *Fundamentals of rotating machinery diagnostics*, Mechanical Engineering-CIME, 125(12):53-54, 2003.
- [30] E. P. CARDEN AND P. FANNING, *Vibration based condition monitoring: a review*, Structural health monitoring, 3(4):355-377, 2004.
- [31] LINK, R.; RIESS, N., *NDT 4.0 - Significance and Implications to NDT – Automated Magnetic Particle Testing as an Example*, 12th ECNDT, Gothenburg, Sweeden, 2018.
- [32] S. W. DOEBLING, C. R. FARRAR, M. B. PRIME, AND OTHERS., *A summary review of vibration-based damage identification methods*, Shock and vibration digest, 30(2):91-105, 1998.
- [33] C. R. FARRAR, S. W. DOEBLING, AND D. A. NIX, *Vibration-based structural damage identification*, Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 359(1778):131-149, 2001.
- [34] BISHOP, C. M., *Pattern Recognition and Machine Learning*, 2006, Springer.
- [35] KOZA, JOHN R.; BENNET, FORREST H.; ANDRE, DAVID; KEANE, MARTIN A., *Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming*, Artificial Intelligence in Design 1996. Springer, Dordrecht. pp. 151–170.



Universidad de
Oviedo



POLYTECHNIC SCHOOL OF ENGINEERING OF GIJON

**BACHELOR'S DEGREE IN INDUSTRIAL ELECTRONICS AND AUTOMATION
ENGINEERING**

**DEPARTMENT OF ELECTRICAL, ELECTRONICAL, COMPUTER AND SYSTEMS
ENGINEERING**

UNIVERSITY OF OVIEDO

BACHELOR'S THESIS NO. 19010071

BUDGET

**DESIGN OF A VIBRATION DATA ACQUISITION SYSTEM FOR A STRUCTURAL
HEALTH MONITORING TEST BENCH**

MR. ARNAIZ BURGUEÑO, ROBERTO

TUTOR: DR. DÍAZ BLANCO, IGNACIO

COTUTOR: MR. GARCÍA PÉREZ, DIEGO

JUNE 2019

Table of contents

	Page
List of Tables	iii
1 Budget	1
1.1 Budget breakdown	1
1.1.1 Labour	1
1.1.2 Necessary material for the implementation of the system	2
1.1.3 Software licenses	2
1.2 Total cost	3

List of Tables

Table	Page
1.1 Labour cost.	1
1.2 Cost of the necessary material for the implementation of the system.	2
1.3 Software licenses cost.	2
1.4 Total cost.	3

Budget

In this document, the cost associated to the development of this thesis will be presented. It has been divided in three sections: labour, necessary material for the implementation of the system and software licenses.

1.1.- Budget breakdown

1.1.1.- Labour

The labour has been defined as *engineering*, where the development of the thesis itself and the work done by tutor and cotutor supervising the correct evolution of the project and reviewing the documentation have been included. Table 1.1 shows the cost associated to the necessary labour to carry out this project.

Type of labour	Quantity [hours]	Unit cost[€/h]	Full cost[€]
Engineering	350	10.00	3,500.00
SUBTOTAL			3,500.00

Table 1.1: Labour cost.

1.1.2.- Necessary material for the implementation of the system

The costs associated to the necessary material for the implementation of the test bench, excitation system and data acquisition system are shown in table 1.2.

Material for system implementation	Quantity	Unit cost[€/u]	Full cost[€]
Data acquisition board NI-USB 6356	1	4,233.00	4,233.00
Accelerometers Wilcoxon Research 786F	3	490.00	1,470.00
Power unit Wilcoxon Research	1	680.00	680.00
Electromechanical actuator Ralux	1	40.00	40.00
Dual motor driver Devantech MD22	1	70.00	70.00
3D printer filament roll	0.172 Kg	22.95 €/Kg	3.95
Torque wrench	1	105.00	105.00
3 Steel plates (10 mm x 150 mm x 500 mm)	18 Kg	0.96 €/Kg	17.28
2 Steel weights (8 Kg and 10 Kg)	18 Kg	0.96 €/Kg	17.28
Aluminium profile (700 mm)	3	13.30	39.90
Aluminium joining plates	4	39.00	156.00
Aluminium support base	1	130.00	130.00
12V Voltage source	1	19.90	19.90
SUBTOTAL			6,982.31

Table 1.2: Cost of the necessary material for the implementation of the system.

Apart from the here mentioned material, it was necessary a PC and some devices or material like oscilloscopes or wire, but it is supposed this is available in an electrical engineering laboratory.

1.1.3.- Software licenses

The cost associated to the necessary software licenses to develop this project is shown in table 1.3

Software license	Type	Quantity	Unit cost[€/h]	Full cost[€]
Matlab R2018a	student license	1	69.00	69.00
SUBTOTAL				69.00

Table 1.3: Software licenses cost.

1.2.- Total cost

The estimated total cost, which can be computed by adding the cost generated by the three previously explained concepts, is shown in table 1.4.

Type of cost	Full cost[€]
Labour	3,500.00
Material for system implementation	6,982.31
Software licenses	69.00
TOTAL	10,551.31

Table 1.4: Total cost.

It must be borne in mind that the total cost could vary depending on the store where the material is acquired or on the lack of basic material available in the laboratory.

