RWTH AACHEN UNIVERSITY

Edge-based Vibration Monitoring System for Textile Machinery

Master Thesis

presented by
B.Sc. Victor Lorhan Loiola Costa

The present work was submitted to the Institute for Man-Machine Interaction.

Examiner: Univ.-Prof. Dr.-Ing. Jürgen Roßmann

Aachen, 18. Jul. 2022

Edge-based Vibration Monitoring System for Textile Machinery

Edge-basiertes Vibrationsüberwachungssystem für Textilmaschinen

presented by
B.Sc. Victor Lorhan Loiola Costa
390048

Advisors:
M.Sc. Björn Eberhardt (DELTA Systems GbR)
M.Sc. Jiahang Chen (Institute for Man-Machine Interaction)

Abstract

Industrial machinery are capital-intensive assets. In order to achieve the return on the investment in their aquisition and operation, it is important that they operate in the best condition possible.

Achieving such best condition requires maintenance efforts. Modern maintenance techniques, based on the Predictive Maintenance concept, aim to maximize machinery operation time by constantly monitoring the machinery and then using statistical methods to determine the point in time when the machine will stop performing satisfactorily. A maintenance stop is then scheduled just before this point, so that the machine does not fail and also is not submitted to unecessary maintenance.

In this work, statistical models are then developed to analyse machine vibration data and determine the machine's condition in a way to facilitate Predictive Maintenance programs.

In carrying out this task, a research on a range of techniques is carried out in order to understand how to interpret vibration data, how to extract meaningful information from it, and how to use this information to assess a machine's condition.

Another central task carried out in this work is the optimization of an already operating vibration signal aquisition setup in order to produce a higher quantity of data using the same resources.

Finally, a user friendly method of displaying the information to the end-user is proposed.

Contents

Lists

Figures

Tables

File Excerpts

Acronyms

FT Fourier Transform 16

FFT Fast Fourier Transform 17

DFT Discrete Fourier Transform 17

MVGD Multi-Variate Gaussian Distribution 31

PCA Principal Component Analysis 26

PC Principal Component 27

AD Anomaly Detection 13

PM Predictive Maintenance 2

OEE Overall Equipment Effectiveness 1

RUL Remaining Useful Life 2

IoT Internet of Things 4

IIoT Industrial Internet of Things 4

RMS Root Mean Square 14

WCSS Within-Cluster Sum of Squares 24

Nomenclature

This page summarizes the nomenclature of mathematical symbols and formulas used in this work.

As a general note, scalars are written in italic (as in k), matrices are written in bold capital letters (as in $\mathbf{X}$), and vectors are written in bold-italic small letters (as in $\boldsymbol{x}$). Also, a vector that represents a matrix row has the row number indication as a superscript in parenthesis (as in $\boldsymbol{x}^{(i)}$), while a vector that represents a matrix column has the column number in indication as a subscript (as in $\boldsymbol{x}_i$).

i, k	General integers used for indexing, counting, etc
j	Unit imaginary number, square root of -1
$\mathbf{X}$	Input dataset matrix with datapoints as rows and features as columns
$\boldsymbol{x^{(i)}}$	i^{th} datapoint/sample (or row) of dataset
$\boldsymbol{x_i}$	i^{th} feature/dimension (or column) of dataset
$\mathbf{C}$	Covariance matrix of input dataset
m	Number of samples in dataset
n	Dimension size (number of features) of samples in dataset. In other contexts, also represents an array indexing variable
λ	General notation for eigenvalue
$\boldsymbol{\lambda}$	Vector-form collection of a matrix's associated eigenvalues
$\boldsymbol{u}$	General notation for eigenvector
$\mathbf{U}$	Matrix-form collection of a matrix's associated eigenvectors
ω	Angular velocity in radians per second
s	Laplace-transform continuous-time complex frequency
z	Z-transform discrete-time complex frequency
v_i	Digital filter's input sample array
v_o	Digital filter's output array
f_{low}	Transfer function for the low-frequency-range digital filter designed
f_{mid}	Transfer function for the middle-frequency-range digital filter designed
f_{high}	Transfer function for the high-frequency-range digital filter designed

CHAPTER 1

Introduction and motivationdasdasdas

1.1 The Way to Predictive Maintenance

Industrial machinery are usually capital-intensive assets. Thus, efforts to keep such equipment in optimal working condition, that is, with the highest availability, performance and quality are an important part of ensuring the return of the investment in their aquisition and operation. A key parameter that combines the metrics mentioned is the Overall Equipment Effectiveness (OEE), which relies on machine maintenance as one of its main drivers of improvement [CNG15].

As the complexity of the machinery increased over time, so did maintenance strategies, in order to optimize OEE levels. The first strategies for maintenance of industrial equipment were simply based on fixing the machine when it presented a problem, i.e., machines would run until failure. This strategy, known as Reactive Maintenance, however simple, has its drawbacks: failures were numerous and unpredictable, and could also be catastrophic, which incurred in high maintenance costs, production losses, and even safety hazards. Due to these issues, reactive-only maintenance programs are today considered generally obsolete and segregated for the niche cases of industrial plants characterized by a large number of small and comparatively inexpensive machinery, where the failure of individual machines does not stop the whole production and usually does not lead to catastrophic consequences [Ran21].

Eventually, with the equipment failure experience accumulated by machine producers and operators, maintenance programs evolved to include also the so-called Preventive Maintenance actions, in which the machine is regularly stopped for maitenance. This new paradigm provided great reduction in the probability of failure between repairs, but at the cost of greater use of spare parts and higher maitenance work than strictly required. Also, it lead to an increased number of machine stops, in contrast to the reactive-only approach, which ensures maximum operating time between shutdowns. The addition

of preventive actions to maintenance programs is, nevertheless, the standard practice in industry. The cost-effectiveness of this approach, as a result of increased OEE and reduced overall maintenance costs, is well established by literature and practice [LPdB06].

Finally, with the accumulation of not only further experience, but also of data about machine maintenance, combined with the statistical analysis of such data, the so-called Predictive Maintenance (PM), or also Condition Monitoring, methodologies started to take form. These methodologies are based on regular inspection of machine conditions by means of sensors and other sources of data and the evaluation of this data by means of mathematical methods to determine the optimum time for a maintenance stop.

Figure 1.1 presents an overview of the maintenance types discussed so far, also ranking them by potential OEE gains offered.

The PM approach combines the both the advantages of avoiding unnecessary maintenance stops and unscheduled machine downtime. It already presents decades of successful implementation in industries such as power and petrochemical [Car06]. In a 1978 study, it was estimated that approximately 65% of the maintenance costs in a selection of British industries could be saved by adequate implementation of PM [NW78].

Idealy, PM aims to assess the machine's Remaining Useful Life (RUL), i.e., how long the machine will be able to perform acceptably before failing. This enables maintenance stops to be performed shortly before then, maximizing the machine uptime and avoiding the failure. A data-based approach for the RUL determination can be achieved by tracking, for a as large as possible number of same-model machines, a condition-indicator variable,

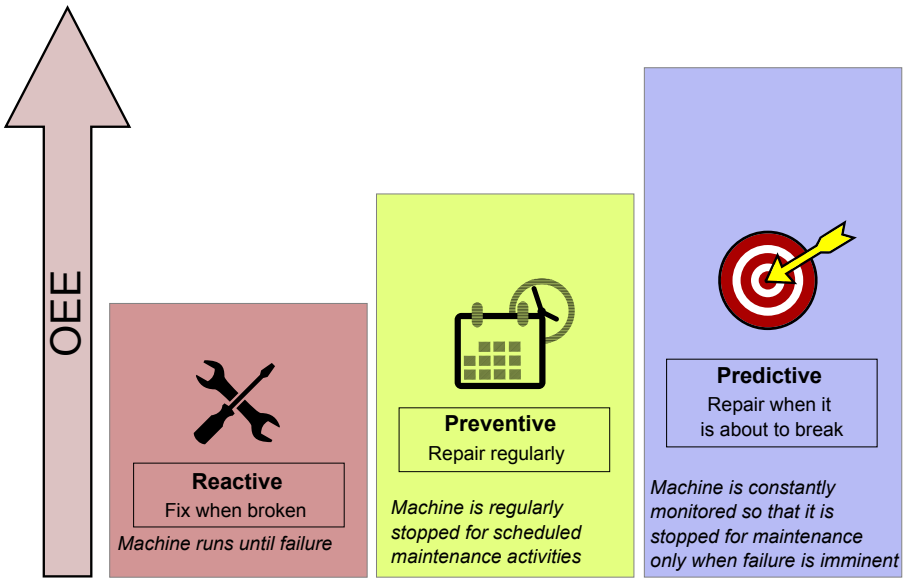

Figure 1.1: *Overview of maintenance types.*

which decreases over the operation and/or calendaric time or number of operation cycles, recording also the moment when the machine failed or stopped working satisfactorily. Then, for a given machine in operation, its RUL can be estimated by checking which historical example from the dataset most closely matches the condition-indicator variable's time-evolution of the machine in operation. Figure 1.2 presents a graphical intuition of this procedure.

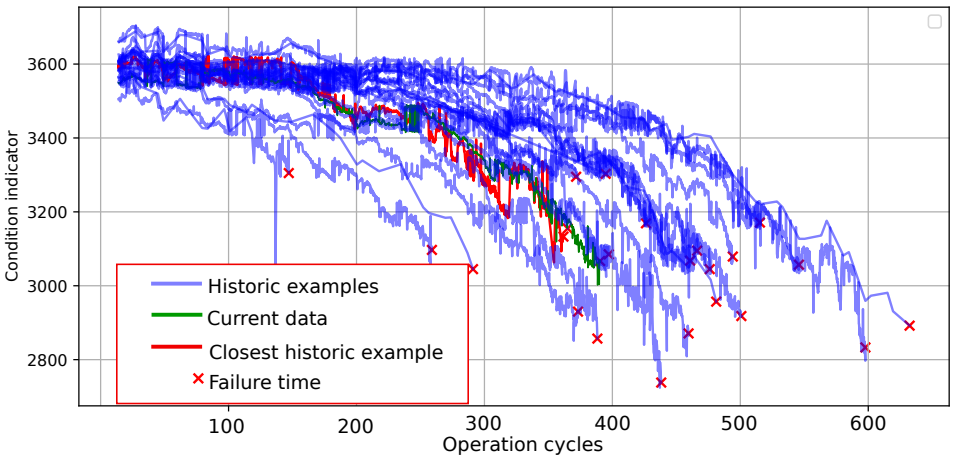

Figure 1.2: *Intuition of data-based RUL estimation*

Developing a model for an accurate RUL determination is, however, quite difficult. Physics-based models are usually prohibitivelly complex, usually requiring highly skilled experts and time-consuming research. Data-based models, as the one just briefly explained, suffers from a variety challenges.

First, naturally, there is a cost factor: PM is based on machine monitoring data. Such data usually comes from sensors, which have an associated aquisition and installation cost. This specific problem, however, has a steadily decreasing relevance, given the developments in wireless edge computing technologies.

Another issue is that of operational variability. Individual machines from a same model might operate in different conditions, in different processes, with different materials, and so on. This lead to the problem that a data-based model built for a machine model in a given type of operation might be significantly less accurate for another machine of the same model, but working in a different type of operation. Thus, there is an additional burden on the models to take into account the operational conditions the machine finds itself in.

One can also point out the fact that the data normally provided by sensors (pressure,

temperature, vibration, etc), while useful for detecting the onset of general and/or specific degradation mechanisms, is insuficient for determining the RUL. In most sucessful solutions, the data provided by sensors, i.e., the data you use to estimate the RUL, is usually later associated to data from maintenance teams indicating the moment in time the machine failed. The resulting dataset, which contains not only the data used to estimate the target varible, but also the target variable associated to each datapoint, is referred to in statistical modelling as a *labeled dataset*, and contains richer information that allows the use of specific types of mathematical models which produce better results. However, in many cases, specially when Preventive Mainenance is employed, there is little information about machine failures, as their usually receive excessive maintenance and failures are rare.

When considering these challenges, it is easy to understand the early success of PM in the power and petrochemical industries. These are highly capital intensive endeavors, so that the cost of monitoring infrastructure is not such a problem. Also, the machines in such plants usually come from a reduced number of manufacturers, operate with reasonably low variation, constant speed and stable loads, so the operational variability problem is much less significant.

Given all that, we can conclude that the sucessfull implementation of PM is an exercise of finding a way within a specific scenario at hand to overcome the challenges mentioned and also of finding intelligent ways to analyze the data that can be reasonably obtained and produce useful results.

1.2 IIoT and Edge Computing

In the current context of large number of internetworked devices, the term Internet of Things (IoT) was created to refer to the amalgama of software techniques, communication technologies and individual devices that constitute these networks which, despite the name, may or may not be connected to the internet.

In the specific case of industrial systems, for which the term is adapted to Industrial Internet of Things (IIoT) the sharp rise in the connectivity and software intelligence employed to deal with the amalgama of computerized sensors, actuators and controllers is said to mark a new phase in industrial technology, also called "fourth industrial revolution" or Industry 4.0.

A key elements found in the architecture of these networks are centralized cloud units, capable of storing and processing large quantities of data. These central units allow other devices in the network to access their computational resources. Such devices, which access the resources from the central units, are called *edge devices*. These edge devices,

on their turn, extend the network to other devices, which exist in large quantities, and are characterized by a wide variety of applications, such as cellphones, game controllers, industrial equipment, and so on.

With the development of chip technology, the processing and storage capabilites of the edge devices increases steadily. With that, although it is simpler to do most of the data processing/storage in the cloud units, there are numerous advantages to the paradigm of shifting some of this task to the edge devices, also called *Edge Computing*:

- **Reduced latency:** The larger the quantity of data being transferred over the network and processed in the cloud units, the longer the data requests should take. Thus, if a device is able to process by itself, reduced traffic enables faster requests.
- **Improved security:** If all information is stored into a centralized unit, an unauthorized access to this single unit gives the invader access to all of the data. The distribution of the data between multiple devices greatly increases the effort required to steal the data.
- **Reduced infrastructure costs:** The more the edge devices process/store data, the less the requirements for the network infrastructure and central units, and thus the smaller their cost.
- **Improved reliability:** The less the edge devices must rely on the central units, the better they are able to keep performing their functions when the units fail, so the system as a whole is still able to function.
- **Larger autonomy for battery-powered devices:** For battery-powered devices, a concious use of energy is required, as the device stops working when the battery is drained, until it is replaced. Wireless communication is frequently the most energy-intensive task in such devices, so less communication means longer battery-life and thus less maintenance efforts in replacing batteries.

1.3 Tasks of the work

Given the context of state-of-the-art industrial practices based on PM and their associated challenges, this work aims to make use of modern industrial wireless sensor networks to extract an analyze data to be used in assessing machinery health and allowing maintenance managers to optimize maintenance schedules.

In carrying out this task, this work aims to optimize an existing wireless vibration monitoring system developed by the company DELTA Systems and currently in operation

in a textile manufacturing plant in the city of Malatya in Turkey. More details about this setup are given in Chapter 2.

The main technology and research gaps that this work intends to help filling can be summarized into two points:

1. **Sensor data analysis:** An algorithm to analyse the real-world data aquired by the sensors should be developed with the objective of giving the best indication possible of the health of the machinery monitored.

2. **Exploitation of hardware infrastructure:** The software and firmware infrastructure should be obtimized in other to better exploit the capabilities of the hardware employed. In more specific terms, it is desired that the wireless sensors produce more data given their limited energy supply before their batteries must be replaced.

CHAPTER 2

The Pollux Vibration Monitoring System

The Pollux Vibration Monitoring System is a wireless solution that aims to facilitate the implementation of PM in industrial machinery. The system is developed by the company DELTA Systems, where this work is being produced.

This works foccuses on an implementation of this system installed in a textile manufacturing plant of the company Calik Denim, in the city of Malatya, in Turkey.

In more specific terms, the setup consists of two gateways and 85 sensors and installed in a carding machine. The installation took place in July 2020, and the sensors have been producing data since then. Figure 2.1 depicts one of the electronic sensors attached to the referred carding machine.

Figure 2.1: *Textile machine with electronic vibration sensor attached (Source: DELTA Systems GbR)*

The system consists of a set of wireless electronic vibration sensors which communicate with a gateway. The gateway can then be acessed through a variety of TCP/IP based technologies in order to access the system's graphical user interface and the measurement database. This architecture is illustrated in Figure 2.2.

This section provides a more detailed overview of the system, first introducing the company which develops it, then its components, and finally the vibration measurements.

2.1 DELTA Systems

DELTA Systems GbR is an electronic design company located in the city of Aachen, Germany. The company was founded in 1996 and designs and produces equipment with applications that range from medical to industrial, with an expertise focus in wireless sensor systems. The company's logo is depicted in Figure 2.3. Figure 2.4 displays a few of the hardware products developed by them, presenting also the types of battery used by the equipment and also some unassembled printed circuit boards as an illustration for the company's activity in the device's design chain.

The company develops a large portion of the elements in the whole chain of electronics design and production. In an overview, the company's development expertise includes:

- **Hardware:** electronics schematic and printed circuit board design and prototyping. Theres is also possibility for small-scale production. The products associated mechanics are also usually designed in-house.
- **Firmware:** microcontroller systems, embedded Linux.

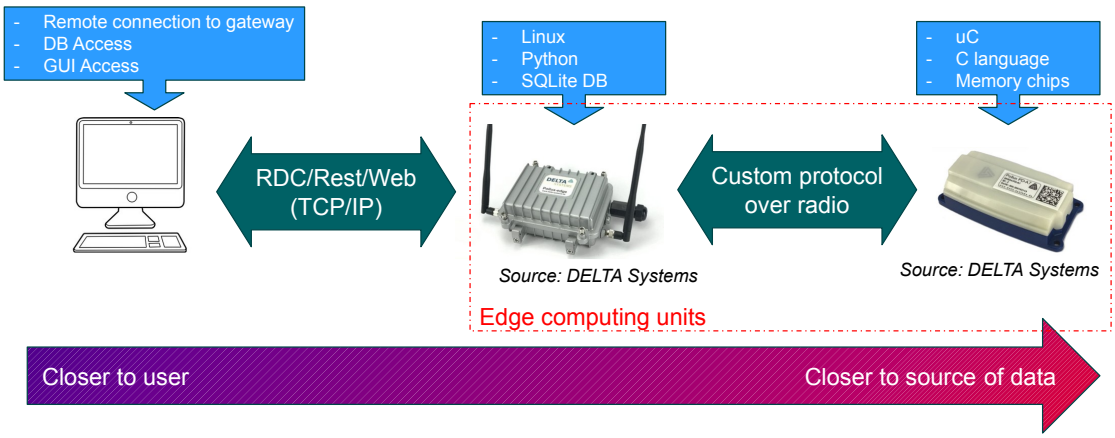

Figure 2.2: *Overview of the hardware elements of the Pollux Vibration Monitoring System*

Figure 2.3: *Logo of DELTA Systems GbR*

Figure 2.4: *Examples of electronic hardware developed by DELTA Systems (Source: DELTA Systems GbR)*

- **Software:** graphical user interfaces, database and cloud connectivity, etc.

2.2 Pollux Vibration Sensor

The Pollux Vibration Sensor, depicted in Figure 2.5 consists of a wireless electronic system controlled by a microcontroller. The microcontroller program is written in C language, and the compiled machine code is stored in the microcontroller itself. The system also comprises an external flash memory used to store vibration measurements for later analysis and/or transmission.

The sensor communicates with the gateway through a custom protocol over radio on the 868 Mhz frequency band.

The sensor is powered by two 3V batteries in series, with the resulting voltage being converter to logic levels by means of a buck converter. It is important to mention that, from the user side, the attractiveness of such sensors is strongly tied to low maintenance requirements. Thus, the sensors are designed so that the batteries must be replaced as

rarely as possible. This requirement is directly translated into a need for an operation with the lowest energy consumption as possible. For this purpose, the sensor system finds itself in *sleep-mode* most of the time, waking up regularly to listen to a sync signal over radio to respond to eventual commands sent by the gateway. Also, overall, the use of radio is energy intensive in such systens and should be avoided in order to conserve battery energy.

As a final note, it is worth mentioning that the microcontroller is not optimized for complex math operations. Due to limitations on both raw processing power and available memory, some operations might take a long time to process and thus increase the energy consumption and also restrict the *real-time* capabilites.

2.3 Pollux Gateway

The Pollux Gateway, depicted in Figure 2.6 consists in a single board computer equiped with multiple communication interfaces.

The single board computer is a linux system that runs a Python application responsible for, among other activities:

Figure 2.5: *DELTA Systems Pollux Vibration Sensor (Source: DELTA Systems GbR)*

Figure 2.6: *DELTA Systems Pollux Vibration Gateway (Source: DELTA Systems GbR)*

- Managing SQLite database where the measurement data is stored
- Sending sync signal to the sensors
- Requesting measurements from the sensors
- Providing a graphical user interface that enables:
  - Adjustments of configurations
  - Programming measurement schedules
- Uploading measurement data to cloud applications, when available

As for the communication interfaces, it has:

- An ethernet port for TCP/IP networks
- A radio interface by means of an external USB transceiver used to communicate with the sensors

While it is not the case for this particular vibration monitoring project, the gateway can optionaly include interfaces for communication over industrial PLC buses.

As for the power supply, it usually comes from mains voltage, but can also make use of *power-over-ethernet*. In any case, the gateway has a cabled power supply and therefore does not suffer from the power restrictions associated with the electronic sensors.

Due to the more powerfull processor, the gateway is also able to handle complex math operations.

While the architecture of both the processor and operational system does not enable a reliable real-time operation, the external radio interface has such capability. This capability can then be extended to the gateway as a whole by means of astute expoitation of the USB communication on the software side.

2.4 Vibration signals

As for the vibration measurements in the system, there are two types:

1. **Long measurements:** These consist of signals 2 secods of duration, sampled at $4.5kHz$, with 16 bit resolution. A measurement is comprised of three of such signals, one for the sensor's x axis, one for the y axis and one for the z axis. The signals of the three axes can be summarized as a single signal, referred to as the a axis. This fictional axis, obtained as per 2.1, contains frequency components of the

three original axes. In this type of measurement, the three signals are sent in their entirety over radio, which consumes a significant ammount of power and greatly compromise battery autonomy. Therefore, these measurements are aquired only twice per day. Figure 2.7 depicts the x, y, z and a axis of a long measurement. In these measurements, the a axis has an offset of $-1g$ applied in order to remove the influence of the constant gravity acceleration on the signal.

2. **Short measurements:** In this type of measurement, the sensor transmits only the maximum value of each of the three signals. They represent a much less detailed information, but require much less energy for transmission. Thus, these measurements are sent much more frequently, aproximately once every 15 minutes. In this work, we aim to use signal processing techniques to send more significant data instead of simply the maximum values.

$$a = \sqrt{x^2 + y^2 + z^2} \tag{2.1}$$

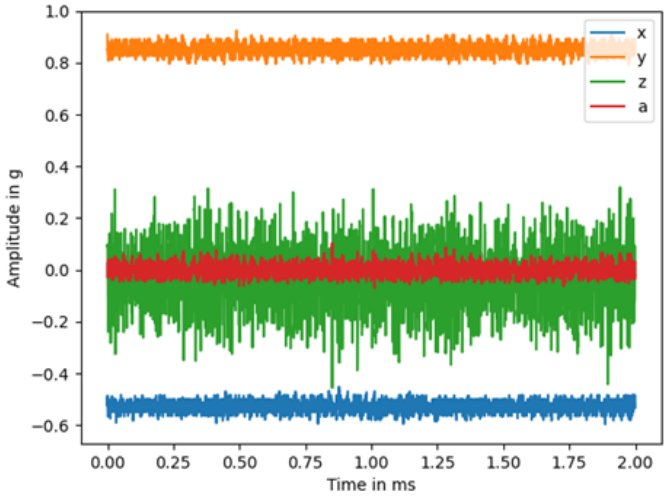

Figure 2.7: *Example of vibration signals aquired by the sensors*

CHAPTER 3

Theory Review

The vibration monitoring solution proposed in this work employs a range of techniques.

In Section 3.1, a research on vibration analysis literature is required in order to gain insights on how to interpret and analyze vibration data, which features to extract from the waveforms for this purpose, and the results that can be reasonably achieved.

In Section 3.2, signal processing techniques are explored in order to understand the computations used to extract information from vibration waveforms.

In Section 3.3, clustering techniques are presented in order to understand if the machinery under study presents a constant and stable operation or if it presents multiple operating modes, and how we can segregate them. This is an important step in order to enable an adequate implementation of Anomaly Detection (AD) techniques.

Finally, in Section 3.3, AD techniques are explored in order to understand their principles and how they can be best applied to the scenario at hand.

Figure 3.1 presents an overview of the knowledge areas and their technologies employed in this work.

3.1 Vibration Analysis

In the context of PM of rotating machinery, vibration analysis is widely considered the single most important technique available [Ran21]. It is a remarkably powerful method for monitoring the general health of an equipment by indicating the onset and/or presence of many fault mechanisms, including [Ran21]:

- Shaft misalignment
- Rotor unbalance
- Mechanical looseness

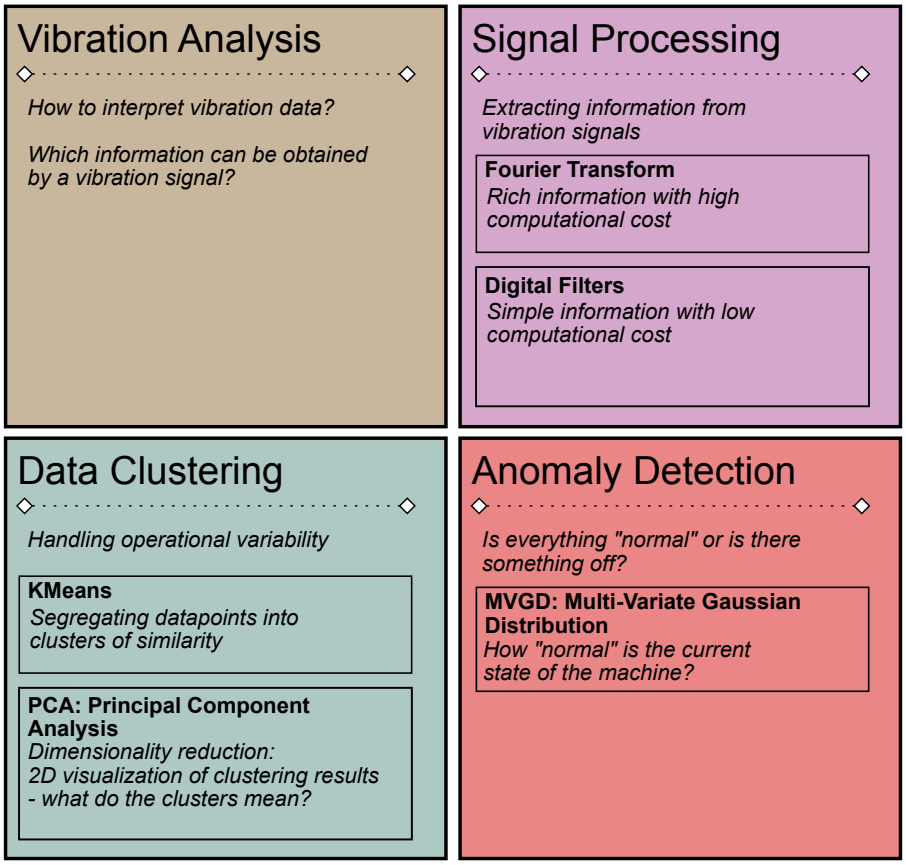

Figure 3.1: *Overview of techniques used in the proposed vibration monitoring system*

- Bearing and gear damage/degradation
- Inadequate reassembly after maintenance

In order to carry out this kind of analysis, the process starts with the aquisition of vibration signals of the machine under test by means of a vibration transducer. Then, the vibration waveforms can be analysed by multiple techniques.

In the simplest approach, one takes into account that, in general, vibration in rotating machinery is to be avoided. Although it is physically impossible to have a rotating machine without producing, the intensity of this vibration should be kept within acceptable levels dictated by the mechanical structure of the machine and the environment where it is located. Thus, the Root Mean Square (RMS) value of the vibration signal is a useful indicator of the machine condition.

This approach, while useful, gives little to no indication of what the root cause of an underlying problem might be. A more sophisticated approach is the frequency analysis of the vibration waveform. The decomposition of the vibration signal into its frequency

components is explained in more details in Section 3.2.1. Once the frequency components are obtained, one can associate the amplitudes over the frequency spectra to the frequencies where mechanical problems such as the ones listed in this section are expected to appear. Also, while the frequency indicates the source of the vibration, the amplitude indicates the severity of the problem.

The process of associating specific mechanical problems to a machine's vibration frequency spectra involves knowledge of a plethora of characteristics such as bearing and gear geometry and materials, type of grease, maintenance history and so on.

A basic characteristic is the rotating speed of the motors. A very common type of motor used in industrial rotating machinery is the four-pole three-phase electric induction motor [Tsy17]. This type of machine presents a mechanical rotation slightly below half that of it's alternate current electric supply. Thus, for a $50Hz$ network, we expect a peak around $25Hz$ for rotating machinery which employ such motors. Thus, for the case of machine speed control by means of the motor, the position where the peak around $25Hz$ appears is directly related to the speed the machine is being operated at. This is specially useful, as the machine's speed can exert a high influence on its vibration pattern, and thus the vibration analysis should take the machine's speed into account [BG12].

The associated frequency of other problems, such as those with bearings, gears and oil, require information which is more difficult to obtain. Thus, pointing out such problems requires more human expertise. As a consequence, computer-solutions for fault predictions perform best for clearly defined problems [BG12].

Another method for using vibration data to monitor rotating machinery is by means of compliance with technical standards, such as *ISO 20816 Mechanical vibration – Measurement and evaluation of machine vibration*. This family of standards, in a general sense, presents procedures for determining the expected vibration pattern of an equipment, and then for judging the deviations from this signature. One drawback of this approach results in the fact that, similar to the expert analysis, it requires significant manual effort into aqcuiring machine specifications. Also, the methods proposed in the standards are not well suited for all types of machines, as they mostly concern with single axis machinery such as large turbines, or reciprocating machinery, such as many types of compressors. As a result, this approach also hinders the suitability of an automated solution for the vibration analysis procedure.

A third method for the analysis of vibration signals is based on establishing the machine's expected vibration signature by means of statistical data. This technique, which is based on so-called AD algorithms from the machine learning realm, aims to using historical signals to establish a pattern of similarity. Then, for new vibration signals, the machine condition that they represent can be estimated as how well they fit into this

similarity pattern.

The AD approach for vibration monitoring, however, presents some drawbacks. First, it is prone to false positives. The cause for this is the fact that innocuous events, such as the replacement of a part for another another one from another model, can potentially result in a change in vibration pattern. The AD algorithm will flag this pattern anomaly as a potential problem, while it offers no harm to the machine operation. Second, the historical dataset is expected to contain vibration signals associated to a well functioning machine, with little to no datapoints corresponding to poblematic behavior. If that is not the case, the AD algorithm might establish a similarity pattern that does not segregate the healthy from the problematic behavior. In the worst case of a dataset with only poblematic behavior data, the algorithm will flag examples of healthy behavior as potentially problematic anomalies.

Despite the drawbacks, the AD approach finds merit to make it worth of real-world applications. The main reason for that is the fact that, in many cases, the cost of hiring an expert to establish a carefully constructed vibration pattern for the machine in question is prohibitively high. Also, for the cases of less standardized machinery configurations, there is insuficient literature on how to establish adequate patterns. Then, AD presents a cost-effective and technically feasible alternative that enables vibration monitoring for a wider range of scenarios.

Beside all sensors and data analysis, it is important to keep in mind that a significant number of the failures in complex mechanical equipment happen accidentally and without any relation to their life-cycle period [Kab16].

3.2 Signal Processing

3.2.1 Fourier Transform

The Fourier Transform (FT) is a mathematical transform widely used to decompose a time-function into multiple sinusoidal functions distributed over a frequency spectrum. The result of this transform is then not based on time, but on frequency [TJ18].

The transformation of a time-function $g(t)$ to it's frequency representation $G(\omega)$ is mathematically expressed as per Equation 3.1:

$$\mathcal{F}\{g(t)\} = G(\omega) = \int_{-\infty}^{\infty} g(t)e^{-j\omega t}dt \tag{3.1}$$

where t represents time and ω represents frequancy in rad/s.

The continuous-time integral from 3.1 can also be represented in the discrete-time sum

expressed in Equation 3.2, which is also known as the Discrete Fourier Transform (DFT):

$$G(\omega) = \sum_{k=-\infty}^{\infty} g[k]e^{-j\omega k} \tag{3.2}$$

The $G(\omega)$ represented in both Equations 3.1 and 3.2 is a function whose domain is constituted of frequencies $\omega \in \mathbb{N}$. Its image, however, is constituted of complex numbers whose amplitude and phase correspond to the amplitude and phase of sinusoidal functions.

For the purpose of engineering applications in signal processing, the discrete-time form given in Equation 3.2 is more relevant for multiple reasons.

First, due to the discrete-time nature of the sampled signals used in signal processing, there is a time-difference T_s between any two consecutive samples. Such T_s value is called *sampling time*, and its inverse $f_s = 1/T_s$ is called *sampling frequency*. As stated by the *Nyquist-Shannon sampling theorem*, the highest-frequency component in the image of $G(\omega)$ will have frequency $f_s/2$ [Jer77]. Thus, the sampling frequency f_s of a signal defines the highest frequency component that can be obtained by the Fourier Transform of this signal.

Also, just like the input signal is of discrete-time, the output signal is of discrete-frequency. The time-resolution of the input signal is equivalent to the sampling time T_s and is thus determined solely by the sampling rate f_s. The frequency-resolution Δf of the output, however, is influenced also by the number N of the samples in the input signal and is expressed Equation 3.3:

$$\Delta f = f_s/N \tag{3.3}$$

The Fast Fourier Transform

The Fast Fourier Transform (FFT) is the name given to a familiy of algorithms that compute de DFT for a finite-legth signal in a computationaly optimized way.

The basic idea behind this optimization comes from the fact that the computation of a signal's DFT as per Equation 3.2 involves many symmetries. Such symmetries, which arise naturally from the mathematical pattern of the transform, represent a computational redundancy, as they consist of calculations that had already been carried out in previous steps. The FFT algorithms then exploit these symmetries in order to reduce the number of computations employed. While the DFT is of complexity $O(N^2)$, an FFT is of complexity of $O(N \cdot log_2(N))$.

It is important to say that the result of an FFT computation is not an approximation of the DFT; it actually yields the exact same result.

The invention of the first FFT algorithm is credited to Cooley and Tukey, who published it in a 1965 paper [CT65]. This algorithm is still popular, being implemented in famous signal processing tools such as in the Python library Numpy [Dev].

One disadvantage of all FFT algorithms, such as the one from Cooley and Tukey, is the fact that it requires the input data to be parsed non-sequentially. This means that the computations does not take each input sample in sequence, but rather in another order predefined by the number of samples. This detail represents a challenge for memory restrained computers, whose Random Access Memory (RAM) is not large enough to store both the input and the output arrays. Section 5.2 describes how this problem impacted the present work and what had to be done to work around it.

3.2.2 Digital Filters

Fundamentals

Signal filters are systems designed to reduce or enhance certain frequency ranges of signals. Such systems are an important part of signal processing and have many applications in fields including telecommunications, audio and video systems, medical devices and automatic control systems [TJ18].

In their most fundamental classification, signal filters can be either analog or digital. Analog filters are constituted of physical elements, usually electronic components. Digital filters are constituted by computer systems that perform mathematical operations to modify the input signal.

In a comparison against analog filters, their digital counterparts offer more flexibility, as their configuration can by changed by software, without any hardware modification [LZ06]. Also, in the context of modern electronics, many signals are already introduced into the system in digital form, in such a way that analog circuits are unable to process them in any form, and only digital filtering can be implemented.

In another fundamental classification, signal filters can be divided according to the frequencies they operate on into one of the following four types:

- **Low-pass:** Allows frequencies lower than the cutoff frequency ω_0, rejecting the ones above it
- **High-pass:** Allows frequencies higher than the cutoff frequency ω_0, rejecting the ones below it
- **Band-pass:** Allows frequencies between a range defined by an inferior boundary ω_{inf}, and a superior one ω_{sup}

- **Band-stop:** Rejects frequencies between a range defined by an inferior boundary ω_{inf}, and a superior one ω_{sup}

Figure 3.2 presents a graphical intuition of the behavior of these four types of filters.

The frequency responses presented in Figure 3.2 represent the behavior of *ideal filters*. Such filters present an instantaneous transition between the bands they accept and reject. Also, they have gain equal to one in their whole passband range, and equal to zero in the whole stopband range. Physically implementable filters, however, present non-idealities such as ripple and transition bands.

For real-world filters, however, the band-type and cutoff frequencies are not the only characteristics of interest. The design of such filters usually takes into account additional factors such as [TJ18]:

- **Flatness of bands:** Ideally, bands should be flat, whether they are passbands or stopbands. However, this is not always possible, and some bands might contain ripple. A ripple in a passband has the undesired effect of applying a reduction to some passband frequencies. A ripple in a stopband has the undesired effect of applying amplification some stopband frequencies, leaving the output with stopband contamination.

- **Width of transition band:** The transition band is the range of frequencies along with the passband transitions to the stopband. It is desirable that transition bands are as narrow as possible.

- **Phase distortion:** The output of the filter differs from the input not only by the amplitude of the frequencies contained, but also by their phase. Phase distortion is usually undesirable when it comes to signal integrity, but might also desired for some applications such as specific audio effects.

Figure 3.3 illustrates the concepts of band flatness and transition band for the example of a lowpass filter with flat stopband and ripple on the passband.

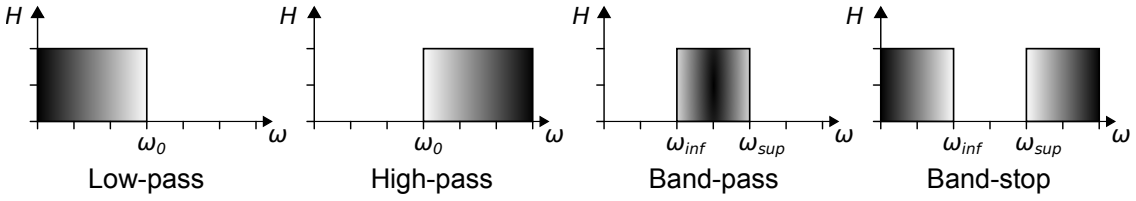

Figure 3.2: *Graphical intuition of band-characteristic classification of signal filters*

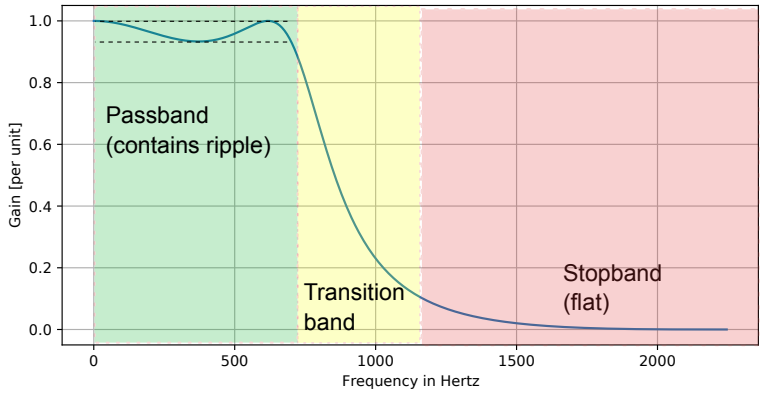

Figure 3.3: *Plot of the response of a lowpass filter with a flat stopband and a passband with ripple*

For most types of filters, these three characteristics can be improved by increasing the complexity of the filter implementation. This action, however, presents some drawbacks. In the analog case, more complex filters usually consume more power, require more components, reduce overall system reliability and so on. In the digital case, higher complexity results in more computations required, which then increases the time required to calculate the filter output, thus the power consumption as well.

Filters can also be classified into their associated circuit topology, or the circuit topology response they emulate, for the case of digital filters. The most common topologies, excluding simple RLC combinations, are compared in Table 3.1.

Topology	**Advantages**	**Disadvantages**
Butterworth	Flat passband Flat stopband	Wide transition band High phase distortion
Chebyshev	Flat stopband Narrow transition band	Passband ripple High phase distortion
Inverse Chebyshev	Flat passband Narrow transition band	Stopband ripple High phase distortion
Bessel	Low phase distortion	Wide transition band

Table 3.1: *Comparison of signal filter topologies*

Design methodology example

As an example of the methodology employed in this work for the design of digital filters, this section will present the design of a simple low-pass filter. The analog representation consists of a resistor in series with a capacitor, in a voltage divider configuration, as per Figure 3.4.

The output voltage V_o can be written as per Equation 3.4:

$$V_o = V_i \times \frac{1/(j\omega C)}{R + 1/(j\omega C)} \tag{3.4}$$

We can then describe the filter's frequency response, F_{RCLP}, as per Equation 3.5:

$$\frac{V_o}{V_i} = F_{RCLP}(j\omega) = \frac{1}{1 + j\omega RC} \tag{3.5}$$

It can be observed in Equation 3.5 that, for a $\omega = 1/(RC)$, F_{RCLP} will have a gain of 0.5. We then define the cutoff frequence $\omega_0 = 1/(RC)$. The substitution of ω_0 into Equation 3.5 results in Equation 3.6:

$$F_{RCLP}(j\omega) = \frac{1}{1 + j(\omega/\omega_0)} \tag{3.6}$$

Now, replacing $j\omega$ by the Laplace complex frequency s, we get Equation 3.6.

$$F_{RCLP}(s) = \frac{\omega_0}{s + \omega_0} \tag{3.7}$$

It is important to mention that Equation 3.7 is called the *transfer function* of the referred filter topology. It is usually the *design entry point* for the development of signal filters. In the task of designing a filter, the usual approach is to consult specialized literature to obtain the transfer function for the desired topology instead of deriving it manually as done in this section.

Now, note that Equation 3.7 refers to an analog implementation of the referred filter. A digital implementation, which can be written in form of code, requires a process called *discretization*. This process maps the transfer function from the continuous complex

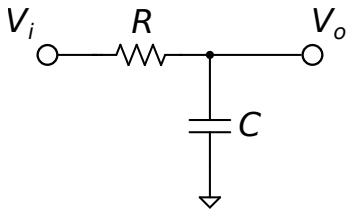

Figure 3.4: *Analog RC low-pass filter*

frequency domain s to the discrete complex frequency domain z. A popular method to implement this mapping is the *bilinear transform*, which allows us to write z as a function of s as per Equation 3.8:

$$s \mapsto \frac{2}{T_s} \cdot \frac{1 - z^{-1}}{1 + z^{-1}} \tag{3.8}$$

where T_s is the sampling frequency of the filter's input signal.

Now, it is also important to mention that the transform in Equation 3.8 produces a phenomenon known as *frequency warping*. Due to this phenomenon, the discrete filter's equivalent cutoff frequency will be different. Fortunatelly, this problem can be worked around by substituting the frequency ω_0 in Equation 3.7 by a "pre-warped" frequency $\tilde{\omega}_0$, which can be obtained as per Equation 3.9:

$$\tilde{\omega}_0 = \frac{2}{T_s} \cdot \arctan\left(\omega_0 \cdot \frac{T_s}{2}\right) \tag{3.9}$$

Now, in Equation 3.7, we first substitute ω_0 for $\tilde{\omega}_0$, and then substitute s per z as described in 3.8. We then obtain the discrete-time transfer function of the referred filter, as per Equation 3.10

$$F_{RCLP}(z) = \frac{T_s\tilde{\omega}_0 + T_s\tilde{\omega}_0 z^{-1}}{(2 + T_s\tilde{\omega}_0) + (T_s\tilde{\omega}_0 - 2)z^{-1}} \tag{3.10}$$

We can then rewrite 3.10 in a simplified form as per 3.11:

$$F_{RCLP}(z) = \frac{b_0 + b_1 z^{-1}}{a_0 + a_1 z^{-1}} \tag{3.11}$$

Once we have the discrete-time transfer function, the next step is to produce the *difference equation*. The difference equation is the method used to write computer code to emulate the analog filter behavior. It works by mapping the z complex frequency to an array index. For example, in the case of the filter's input voltage V_i, the mapping is expressed as follows:

$$V_i \cdot z^{-k} \mapsto v_i[n - k] \tag{3.12}$$

where n is a counting variable indicating the current index of the array containing the sampled v_i values.

Now, going back to the output to input voltage relation of our filter based on 3.11, we rewrite it as:

$$V_o(z) \cdot (a_0 + a_1 z^{-1}) = V_i(z) \cdot (b_0 + b_1 z^{-1}) \tag{3.13}$$

Then, the difference equation is achieved as:

$$a_0 v_o[n] + a_1 v_o[n-1] = b_0 v_i[n] + b_1 v_i[n-1] \tag{3.14}$$

It can then be written in the more intelligible form which can then be turned into code for a computerized digital filter function:

$$v_o[n] = \frac{b_0}{a_0} v_i[n] + \frac{b_1}{a_0} v_i[n-1] - \frac{a_1}{a_0} v_o[n-1] \tag{3.15}$$

By examining Equation 3.15, a problem emerges when $n = 0$, as the computations will try to access $v_o[-1]$. First, in most programming languages, array indexes are supposed to be integers larger then zero, so the referred access attempt would result in an execution error. Also, even if the language implementation accepts negative array indexes, there is still the issue that the value of the array at the referred index was not defined.

Both these problems can be solved by a data structure known as *ring buffer*. This structure can be described as a *first-in-first-out circuilar queue*, with one index referred to as *head*, representing the last value to enter the array, and another index referred to as *tail*, representing the first of the current values to enter the array. When initializing this structure, all the values can be set to zero, so the initialization problem described can be easily handled. Also, when the desired array index is computed as a negative number, which happends during the few first filter output calculations, this desired array index can be simply set to zero. Another benefit of ring buffers is that they optimize code execution, as the addition of a value into the buffer does not need to change all the values of the buffer, just a single value and the buffer's associated head and tail indexes.

To gain more practical insight into the application of ring buffers for digital filters, let us consider a hypothetical difference equation similar to Equation 3.15. In this hypothetical equation, however, we need to access v_i from indexes n to $n-3$ and v_o from n to $n-4$. In such a case, the filter implementation would required a ring buffer with length 4 for the v_i values and a second ring buffer with length 5 for the v_o values.

Figure 3.5 presents an intuition of the working principle of a ring buffer.

3.3 Clustering

In statistical data analysis, the term "clustering" refers to the task of dividing a group of data points into subgroups, also called *clusters*, of data points which are considered similar according to some specification.

In this context, the concept of "similarity" can be defined in different forms. A simple and intuitive method is the Euclidean distance between the data points, meaning that

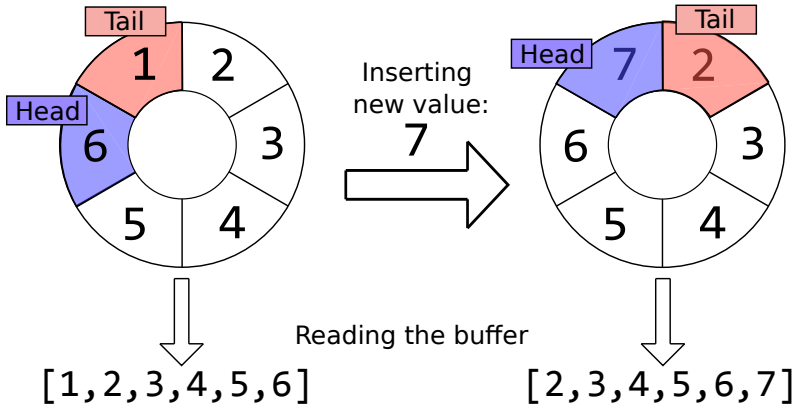

Figure 3.5: *Intuition of the operation of a ring buffer*

points located close to each other are considered similar.

As already explained in Section 1.1, industrial machinery usually presents different operational modes. These different types of operations produce differences in vibration signals which must be taken into account in the analysis of the acquired signals. This work then makes use of clustering techniques in order to identify aggregation patterns in the vibration signals which can be associated with different operational modes.

This section then describes the two specific clustering techniques employed in this task: the K-Means algorithm and the Principal Component Analysis.

3.3.1 K-Means

K-Means is a simple and popular technique within the realm of data clustering. This technique works, in essence, by identifying a certain number of positions around which the data points tend to aggregate themselves and then labeling the data points according to the closest of the positions of aggregation identified [ASI20].

Figure 3.6 provides an intuition for the working principle behind this technique. It illustrates a toy example with multiple datapoints which aggregate themeselves in two clusters. The objective of K-Means is to find points C_A and C_B, namely the centroids, which minimizes the Within-Cluster Sum of Squares (WCSS), which is determined as per expressed in Equation 3.16:

$$WCSS = \sum_{i=1}^{m} l_i^2 \tag{3.16}$$

where m is the number of points in the dataset and l is the Euclidean distance between

each point p_i and the nearest centroid to it, as per Equation 3.17:

$$l_i = min(dist(C_A, p_i), dist(C_B, p_i)) \tag{3.17}$$

The task of finding such centroids, however, is not simple. For a problem with n samples, k clusters and d dimensions, the global optimum solution can be exactly solved in time $O(n^{dk+1})$. As this is usually prohibitivelly computationaly expensive, even for simple problems, heuristic methods, which are not guaranteed to find the global optima, are usually employed. One such method is Lloyd's algorithm heuristic, which has worst-case complexity $O(nkdi)$ [ADH+09]. In this algorithm, we start with initial guesses for the centroids and then enter an iterative mode composed of two steps:

1. **Assignment step:** each point is assigned to the cluster with nearest current centroid (considering the distance calculated through Equation 3.17)
2. **Update step:** the position of each centroid is recalculated as the average position of the points currently assigned to it

Convergence is reached when the the points no longer change their assigned clusters.

The convergence speed can also be improved by an educated initial guess for the centroids by means of prior knowledge of the data or also by the k-means++ technique. In this later approach, a probabilistic analysis is carried out in the data points to be clustered in order to define the optimized initial position for the centroids in the K-Means procedure [AV07].

At this point, it is useful to remember that the K-Means technique does not specify the number of clusters in the data. This is an information that must be given as an input to the problem. However, this information is usually not known beforehand, and figuring

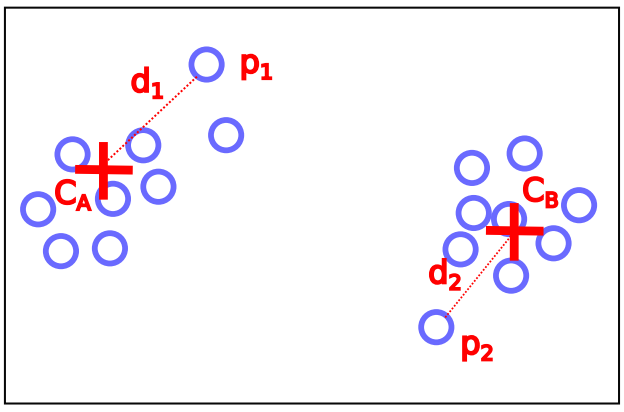

Figure 3.6: *Intuition of K-Means clustering technique*

out how many clusters the underlying patterns in the data present is actually part of what is expected of a data clustering procedure.

Fortunatelly, there are methods to to solve this problem. One of them is the so-called "elbow-method", illustrated in Figure 3.7. In this figure, the plot on the left depicts a group of data points which lend themselves to a subgrouping into three clusters. The right plot depicts the resulting $WCSS$ versus the number of clusters given as input to the K-Means technique. The optimum number of clusters is that after which no significant reduction in $WCSS$ is obtained; the point where this happens forms an "elbow-shaped" curve in the graph, which can be detected by visual inspection or also by some automatic method. In the case of the referred figure, this sharp decrease in the $WCSS$ variation can be easily detected when the number of clusters is three.

Despite the ease of comprehension and simplicity of implementation of the "elbow-method", the Principal Component Analysis (PCA), described in 3.3.2, provides additional insights into the data and is therefore also widely used.

The K-Means technique works well for cluster patterns in which the points aggregate themselves in the forms of n-dimensional spheres. When the points aggregate themselves in more complex patterns, however, the results tend to be less satisfactory, and that is the main disadvantage of the K-Means technique. This shortcoming might result in mislabeling of points, i.e., in points being assigned to a cluster that does not match the condition actually represented by the point.

However, in conclusion, k-means still finds its merit in being a simple and industry-tested technique [ASI20].

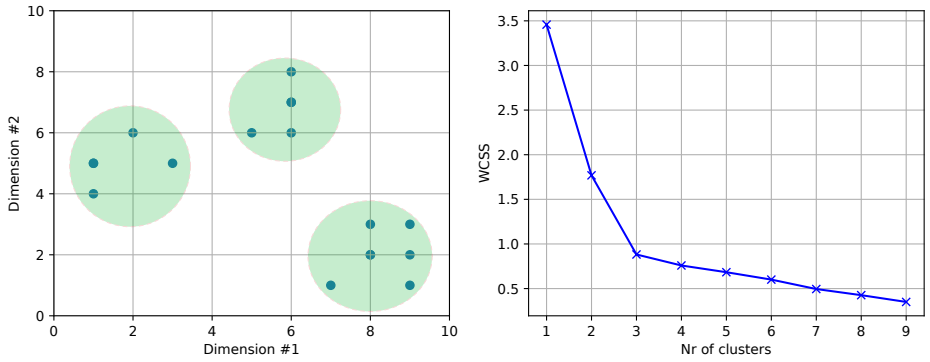

Figure 3.7: *Example of the use of the "elbow-method" plot for determination of optimum number of clusters through K-Means*

3.3.2 Principal Component Analysis

PCA is a data dimensionality reduction technique widely used to represent data with more than three dimensions in a graphical way in two or three dimensions. This enables humans to use their sense of vision to try to find insights in the data.

The intuition behind this technique can be explained as applying a change of basis in the data, where this new base is composed of vectors that point in the directions of maximum variance in the data. These vectors, also called Principal Component (PC), are hierarchically sorted based on how much of the total data variance they explain.

Such principal component vectors are equivalent to the eigenvectors of the data's covariance matrix, and the variance explained by them is directly related to their associated eigenvalues [WRR03]. Thus, the math behind PCA can be summarized by building a linear transformation matrix with the eigenvectors of the covariance matrix and then applying this linear transformation to the data in question.

Before diving into the details of the math behind PCA, it is useful to recap that, in the notation of this text, our dataset matrix is $\mathbf{X}$, with each column $\boldsymbol{x_i}$ representing a dimension (also called a "feature" in the context of this thesis), and each row $\boldsymbol{x^{(i)}}$ representing a data point (also referred to as "measurement" in this thesis). The number of measurements in the dataset is m, and the number of dimensions in them is n. As the objective of PCA is to find a representation of the data with a lower dimension, let d be the targeted number of dimensions, for which, naturally, $d < n$.

This technique requires the data to be centered around the origin, so the first step in order to apply this technique is to find the average vector $\boldsymbol{\mu}$ of the dataset, as per Equation 3.18:

$$\boldsymbol{\mu} = \frac{1}{m}\sum_{i=1}^{m}\boldsymbol{x^{(i)}} \tag{3.18}$$

The next step is to obtain the dataset's covariance matrix $\mathbf{C}$, of dimensions $m \times m$, as per Equation 3.19:

$$\mathbf{C} = \frac{1}{m}\sum_{i=1}^{m}(\boldsymbol{x^{(i)}} - \boldsymbol{\mu})(\boldsymbol{x^{(i)}} - \boldsymbol{\mu})^T \tag{3.19}$$

Once the covariance matrix is obtained, we proceed to find its eigenvectors and eigenvalues. As per definition, a real number λ and a n-sized vector $\boldsymbol{u}$ are, respectively, an eigenvalue and its associated eigenvector of our matrix $\mathbf{C_{n\times n}}$ if they satisfy Equation 3.20:

$$\mathbf{C}\boldsymbol{u} = \lambda\boldsymbol{u} \tag{3.20}$$

Rearranging Equation 3.20, we get Equation 3.21:

$$\mathbf{C}\boldsymbol{u} - \lambda\boldsymbol{u} = 0 \tag{3.21}$$

Considering that $\mathbf{C}$ is a matrix and $\mathbf{I}$ is the identity matrix, we can put $\boldsymbol{u}$ in evidence, as per Equation 3.22:

$$(\mathbf{C} - \lambda\mathbf{I})\boldsymbol{u} = 0 \tag{3.22}$$

The non-trivial solution of Equation 3.22 can be found by Equation 3.23:

$$det(\mathbf{C} - \lambda\boldsymbol{I}) = 0 \tag{3.23}$$

Equation 3.23 leads to a polynomial of λ known as the *characteristic polynomial* of $\mathbf{C}$. The roots of this polynomial are the eigenvalues of $\mathbf{C}$.

Now, for the purposes of the PCA, it is required that each solution λ_i is sorted in descending order, i.e., $\lambda_n > \lambda_{n+1}$. This is important because it makes sure that their corresponding eigenvectors will then be sorted from the largest to the smallest spread of data accounted for. It is worth noting here that covariance matrices, as is the case of our matrix $\mathbf{C}$, have the property of being positive definite, thus their eigenvalues are always positive [Joh70].

Then, we move to find each eigenvector $\boldsymbol{u_i}$ of matrix $\mathbf{C}$ by solving the linear system resulting from Equation 3.24 for each λ_i:

$$\mathbf{C}\boldsymbol{u_i} - \lambda_i\boldsymbol{u_i} = 0 \tag{3.24}$$

Now, let $\mathbf{U}$ be the eigenvector matrix of $\mathbf{C}$, with each column constituted by the eigenvectors $\boldsymbol{u_i}$ in unit vector form and sorted by descending order of associated eigenvalue. The $\mathbf{U}$ matrix is then called the Principal Components Matrix of $\mathbf{C}$. The proportion τ_k of the variation explained by PC_k to the total variation of the dataset $\mathbf{X}$ is given by:

$$\tau_k = \frac{\lambda_k}{\sum_{i=0}^{n} \lambda_i} \tag{3.25}$$

We can then build a matrix $\mathbf{U_{reduced}}$ by taking only the first d eigenvectors of $\mathbf{C}$ or, in other words, taking only the first d columns of $\mathbf{U}$.

Finally, a sample $\boldsymbol{x^{(i)}} \in \mathbb{R}^m$ can be mapped to dimension-reduced-sample $\boldsymbol{x^{(i)}_{reduced}} \in \mathbb{R}^d$ according to Equation 3.26:

$$\boldsymbol{x^{(i)}_{reduced}} = \mathbf{U^T_{reduced}} \cdot \boldsymbol{x^{(i)}} \tag{3.26}$$

By using Equation 3.25, for our target dimension d, the proportion $\tau_{x_{reduced}}$ of the variation that the dimension-reduced dataset $\boldsymbol{x_{reduced}}$ explains from the total variation in x is given by Equation 3.27:

$$\tau_{x_{reduced}} = \frac{\sum_{i=0}^{d} \lambda_i}{\sum_{i=0}^{n} \lambda_i} \tag{3.27}$$

3.4 Anomaly Detection

Anomaly Detection AD is a term used to refer to a number of mathematical techniques which aim to detect outliers in datasets.

As briefly explained in Section 3.1, these techniques can be summarized in two basic pillars:

1. Providing a mathematical model that captures the underlying patterns in a dataset
2. Using this model to declare if a given datapoint conforms to the model-associated pattern or not, that is, if it is a normal or an anomalous point

For these techniques to work properly, some caveats must be taken into account. First, it is important that the data characteristics present in the dataset (that is, the features of the datapoints) contain useful information with regards on how to contrast normal and anomalous behaviour. In our case of monitoring of vibration signals, this requirement dictates that a research in vibration analysis literature is used to best identify the vibration characteristics of interest (such as RMS, frequency spectra) where an anomaly is expected to manifest itself.

Second, also as already mentioned in Section 3.1, it is important that the dataset contains most or even exclusively data considered normal. This ensures that the pattern resulting from the algorithm fits the representation of normal data only, and easily segregates anomalous data.

As a general context for the area of application of AD, it is important to mention that they are classified as *unsupervised learning* algorithms. This classification is associated with advantages and disadvantages. The main advantage is the fact that it enables the use of *unlabeled datasets*. Such datasets contain the features used to estimate the target variable, but not the target variable itself.

On the other hand, the lack of the target variable information precludes the use of feedback-based mathematical models which greatly enhance the algorithm's performance. This drawback demands higher efforts in designing data features that help highlight the difference between normal and anomalous behavior in order to help the algorithm perform better.

Also, the very concept of performance for such algorithm has to be understood in a different context. In supervised learning methods, the labeled data can be used to quantify how well the algorithm's estimations match the recorded results. For AD with unlabeled data, however, the performance is usually determined by examining the results produced by the algorithm during a certain time of field operation. A useful performance metric, in the case of industrial machinery, is the increase in OEE.

Another interesting design caveat of AD algorithms is the tradeoff between false positives and false negatives. As a general rule, such algorithms tend to occasionally flag normal datapoints as anomalous. The algorithm can then be tuned to accept a more strict definition of data anomaly, but that would come at the cost of potentially missing some machine anomalies that the system falsely classify as normal.

Finally, it is important to mention that problems in industrial machinery and processes can have unexpected causes, unrelated to what the data used by the AD algorithm could possibly point out. This is useful to keep in mind in order to understand the limits of the technique and how to design a solution aiming at realistic results.

We now proceed to explain the AD technique, which we use to determine how well a given vibration signal fits the statistic profile presented by the machine's vibration history.

3.4.1 Multi-Variate Gaussian Distribution

Given a single-variate Gaussian distribution with mean μ and variance σ^2, the probability that a sample x belongs to this distribution is given by Equation 3.28:

$$p(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right) \tag{3.28}$$

where μ is determined as per Equation 3.18.

The determination of σ^2 is given as per Equation 3.29:

$$\sigma^2 = \frac{\sum_{i=1}^{m}(x^{(i)}-\mu)^2}{m} \tag{3.29}$$

The graph of $p(x|\mu,\sigma^2)$ vs. x is given in Figure 3.8, which is commonly called "bell-curve" due to its shape. As can be easily deducted from the graph, the further from μ that a given x presents itself, the smaller the probability that x belongs to the depicted distribution.

For a simple system which can be described by a single variable, as long as this variable is normally distributed, Equation 3.28 is sufficient to develop an anomaly detection procedure. By considering that a "normal" observation is one with high probability and that,

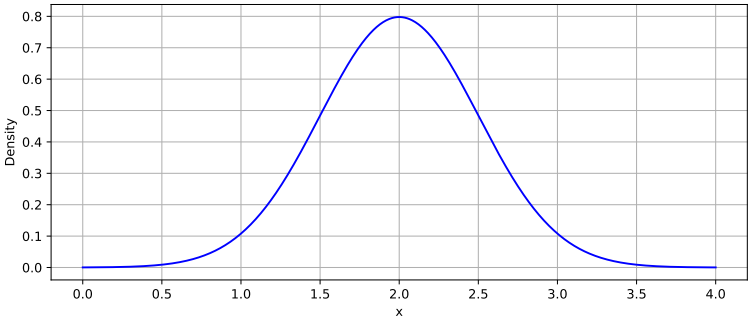

Figure 3.8: *Gaussian probability distribution - single variable case*

likewise, an "anomalous" observation has low probability, for a given new observation x, if $p(x|\mu, \sigma^2) < \varepsilon$, where ε is a minimum allowed probability threshold, we can flag this new observation as a potential anomaly. The optimum value for ε can be determined experimentally, usually also taking into account the cost of false negatives and of false positives for the classification system at hand.

Although such simple system is widely and sucessfully used in inumerous applications in industry [Oak07], one quickly realises that many systems are more complex and therefore cannot be satisfactorily described by a single variable. Nevertheless, for a system with n variables (which, in this context, we can also call "features" or also "dimensions"), we can adapt the strategy just described and define the probability of a n-dimensional sample $\boldsymbol{x}$ as the product of the probabilities of each variable x_i, as expressed in Equation 3.30:

$$p(x) = \prod_{i=1}^{n} p(x_i|\mu_i, \sigma_i^2) \tag{3.30}$$

The effectiveness of this strategy, however, potentially degrades when the different random variables are *correlated* to each other. As an illustration, we can analyse the simple two-dimensional example depicted in Figure 3.9. A visual inspection easily leads to the conclusion that $\boldsymbol{x_{anom}}$ does not conform to the aggregation pattern displayed by the other points. However, as $\boldsymbol{x_{anom}}$ presents x_1 and x_2 coordinates well within the ranges considered normal, Equation 3.30 would assign it a high probability, i.e., it would fail to flag it as a potential anomaly.

To solve this problem, the Multi-Variate Gaussian Distribution (MVGD) comes into play. It does so by taking the correlations between the variables into account through the use of a covariance matrix $\mathbf{C}$ instead of a single variance σ^2. The determination of matrix C was already presented in Equation 3.19. Then, the probability of a n-dimensional point x belonging to a n-variate gaussian distribution with mean given by n-dimensional vector

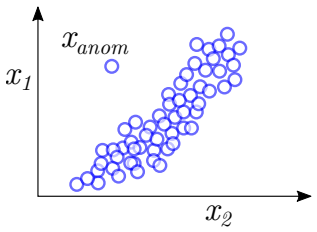

Figure 3.9: *Illustration of anomaly that single variate gaussian anomaly detection strategy fails to detect*

$\boldsymbol{\mu}$ and n-order covariance matrix $\mathbf{C}$ is expressed by Equation 3.31:

$$p(\boldsymbol{x}|\boldsymbol{\mu}, \mathbf{C}) = \frac{1}{(2\pi)^{n/2}|\mathbf{C}|^{1/2}} \exp\left(\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \boldsymbol{C}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right) \tag{3.31}$$

The covariance matrix $\mathbf{C}$ is determined in the same Equation 3.19.

The resulting probability can then be compared with a minimum probability threshold ε just like in the previous strategy.

It is worth noting that the probability that results from Equation 3.30 is equivalent to the particular case of Equation 3.31 in which the covariance matrix $\mathbf{C}$ is diagonal, i.e., it has the off-diagonal elements all equal to zero. In this case, the variables present zero correlation to each other, and the computation of Equation 3.31 results in the same productory of Equation 3.30.

The properties of the covariance matrix $\mathbf{C}$ which enable the detection of outliers such as the one depicted in Figure 3.9 can be illustrated geometrically. For this purpose, we can employ the four two-dimensional distributions depicted in Figure 3.10.

In this table-like illustration, each column represents a different covariance matrix example and its effects on the data distribution. The 2D heatplot in the center row represents the MVGD probability resulting from the associated $\mathbf{C}$, with the color blue representing the lowest values and the color red representing the highest ones. Note that the probability values are all distributed along elispoidal contours of same probability. The lower row depicts the same heatmap association with the addition of a third dimension axis, also representing the probability associated. This 3D representation then depicts an extension of the "bell-curve" from Figure 3.8 for the case of a distribution with two random variables.

We can analyse the four cases, going from left to right, as following:

1. In the first case, the covariance matrix $\mathbf{C}$ is diagonal, therefore we can conclude that x_1 and x_2 are not correlated. In addition to that, as their respective variances are equivalent, the ellipsoids are of circular shape.

2. In the second case, $\mathbf{C}$ is also diagonal and the variances are also equivalent, so the observations from the previous case are still valid. The difference lies in the fact that, for the current case, as the variances are smaller, the "bump" shape has smaller height, but larger area, although the volume under the curve is still unity.

3. In the third case, x_2 has a higher variance then x_1, which results in ellipsoids which are longer along the direction of x_2. Note that the variables are still not correlated.

4. Finaly, in the last case, we have a non-diagonal $\mathbf{C}$, which represents a correlation between x_1 and x_2. As a result, the ellipsoidal shapes are tilted. Note that this distribution is similar to the example depicted in Figure 3.9. In this case, however, the heatmap shows that point $\boldsymbol{x_{anom}}$ would be situated in a region with much smaller associated probability that the other points, and then we could sucessfully flag it as a potential anomaly based on the ε-threshold strategy discussed.

As already explained, the MVGD technique is able to automatically capture correlations between the variables in the data, which is clearly an advantage in comparison to the single-variate case. However, the single-variate case is also able to capture correlations between variables if new variables, which explicitely reflect such currelations, are manualy inserted into the model. Considering the example of Figure 3.9, the design of a new variable $x_3 = \frac{x_1}{x_2}$ would greatly facilitate the identification of outliers such as x_{anom}, and this approach is also less computationaly expensive then the matrix computations

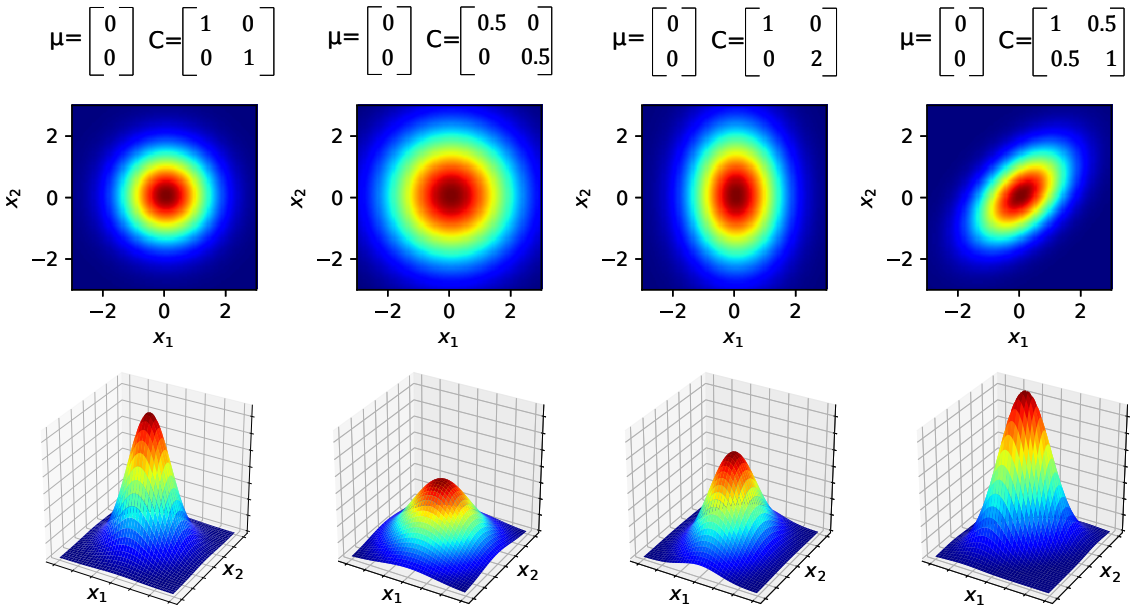

Figure 3.10: *Effects of variation of the elements in the covariance matrix*

required for Equation 3.31, especially for a higher number of variables. However, the necessity of a manual design of such variables compromises the generality of the final solution developed in this work. This is undesirable, given that it is in the interest of the company where this work is produced to easily adapt this solution to other projects.

It is also worth mentioning that it is generaly advisable not to use too many variables in a MVGD model. One of the reasons for that, as already explained, is that it increases the computational cost of Equation 3.31. Another reason is the fact that, if the number of dimensions is higher than the number of samples, than the covariance matrix C is non-invertible.

When it comes to the ability of a gaussian-based model to sucessfully segregate outliers, its performance tends to increase with larger dataset size and lower proportion of anomalous samples in it. Such datasets tend to provide a greater separation between the probability of normal and anomalous samples, allowing a larger range for the design of the ε-threshold to find the best result within the false-positive vs. false-negative tradeoff.

Quantile transform

For a gaussian-based anomaly detection algorithm to give adequate results, it is necessary that the input random variables present an unimodal gaussian distribution, as Equations 3.28 and 3.31 are based on this assumption. This assumption might, however, not hold true for some cases.

In some cases, we might have, for example, a multi-modal distribution. This is a common situation for data which contain clusters. Given the work's task of monitoring industrial assets, such situation is usually indicative of an asset that presents multiple operation modes, which is common in industry. This problem can potentialy be solved by clustering the data previously and making the AD models cluster-specific. In other words, each cluster should have it's own $\boldsymbol{\mu}$ and $\mathbf{C}$ to be used in Equation 3.31 for determining the probability that a given observation $\boldsymbol{x}$ belongs to it. The approaches for clustering are explained in Section 3.3.

On other cases, it is not unusual that a given distribution has a mean value which is different from the median, i.e., the distribution is skewed. For this such cases, we can use the *quantile transform* to make the data more Gaussian-like. This transform works by using the *quantile regression* method to estimate the gaussian cumulative distribution of the variable [KH01]. The data points are then mapped to this equivalent gaussian distribution. An example of the result of the application of this transform is depicted in Figure 3.11

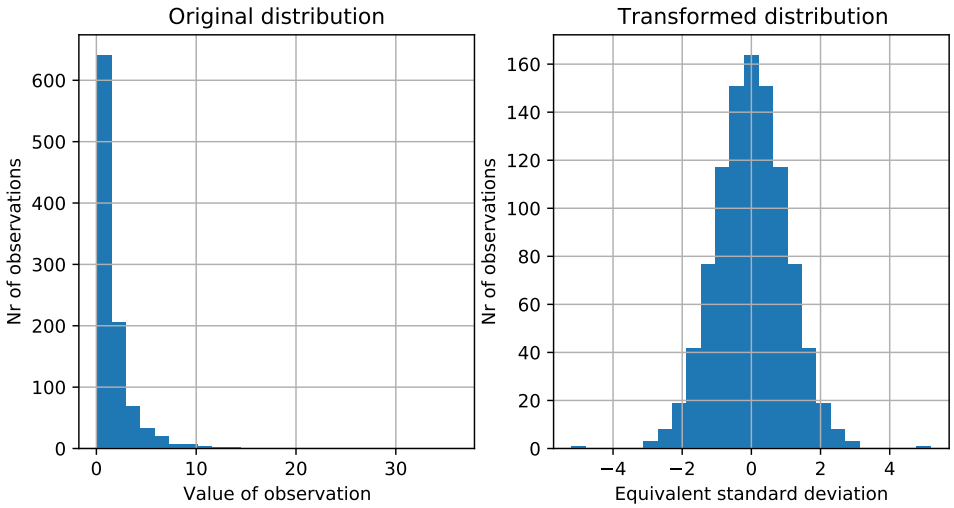

Figure 3.11: *Example of gaussian-mapping of non-gaussian data through quantile transform*

CHAPTER 4

Sensor data analysis and Anomaly Detection algorithm development

This section illustrates the data analysis procedure usuing the data as acquired on site by the sensor with the ID 5. The same procedure applies for the other sensors.

4.1 Basic data exploration

First, as an a more detailed illustration of the signals we are working with, we build axis a. The, we take the fft for the axes x, y, z and a. Figure 4.1 depicts such signals.

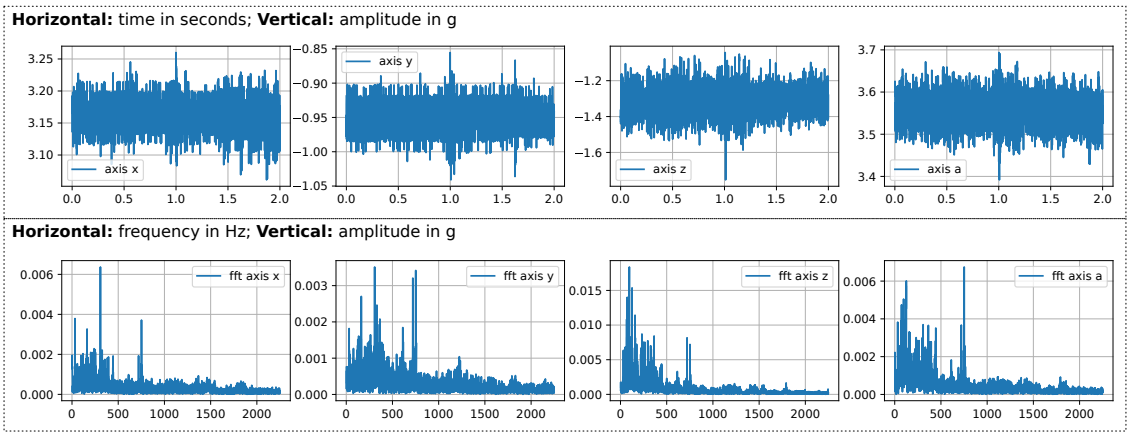

Figure 4.1: *Example of plot of the four vibration axes and their respective FFT's*

4.2 Motor rotation frequency

For multiple measurements, we observed the FFT in the range between 25 Hz and 35 Hz. This is plotted in Figure 4.2.

As it can be seen, the peak presents itself in the 30 Hz. It is worth mentioning that the network frequency in Turkey is 50 Hz, which should correspond to a 25 Hz mechanical frequency for an electrical 4-pole-induction motor. One explanation for this disparity is the possibility of using frequency inverters to control the speed of the motor. One speculation about the need for this adjusment on site is that the mechanical equipment the motor has to drive might be designed to rotate at 1800 rpm instead of 1500 rpm.

An important takeaway from this analysis is the fact that the frequency range that contains the indication of the expected motor speed is from 25 Hz to 35 Hz. This information will be used in Section 5.2 in order to optimize the fourier transform computation by only computing this specific range.

4.3 Digital filtering

In order to acquire frequency domain information at a computational cost compatible with the power restriction of the sensors, it was decided to use digital filters. The frequency range from 0 to 2250 Hz was then divided into three ranges:

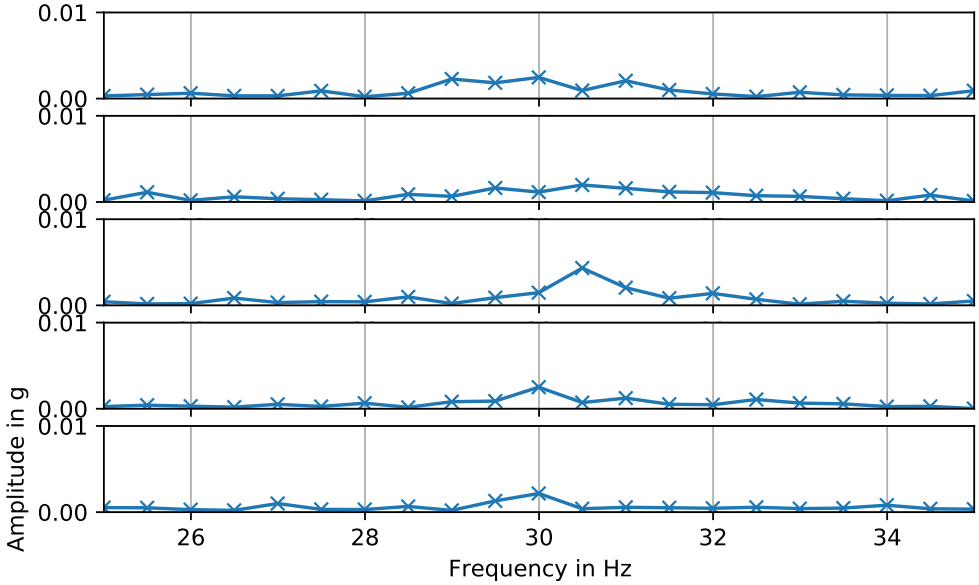

Figure 4.2: *Frequency peaks in the motor rotation expected frequency range*

- a lower range up to 500 Hz
- a middle range betwen 500 and 1250 Hz
- a higher range from 1250 Hz on

Then, using the approach described in Section 3.2.2, three discrete filters were devised. As for the design decisions taken for the development of these filters, they can be summarized as:

- **Topology:** The topology chosen was the inverse Chebyshev. The first reason for this choice is the narrow transition band, which is always desirable. The tradeoff for the narrower transition band is a larger phase distortion, which is irrelevant for our case, as we do not take phase into account in our modelling. Finally, we desired a flat passband in order to preserve as much as possible the information of the passband for our modelling.
- **Order:** As a general rule, the overall performance of the filter is improved with higher order, at the cost of more computations required. By experimentation, the order value was set to 3.
- **Stop-band attenuation:** This parameter is specific for the design of inverse Chebyshev filters. The higher its value, the less stop-band contamination will be present in the output, at the cost of a wider transition band. Through experimentation, its value was set to 26.
- **Design entry point:** The continuous-time transfer function used as the design entry point was obtained through the Python library Scipy, using the *signal.cheby2* class.
- **Overlap:** Some overlap was included in the cutoff frequencies of the three filters. This was a decision taken in orther to achieve an optimum configuration within the numerous tradeoffs in filter design.
- **Sampling time:** The sampling period used in the discretization for all the filters was the same as that of the input signal, which is $1/4.5kHz$.

The lower-frequency range filter is of low-pass type, with cutoff frequency of 700 Hz and discrete-time transfer function given by 4.1:

$$F_{low}[z] = \frac{0.0609 + 0.00579283z^{-1} + 0.00579283z^{-2} + 0.06095431z^{-3}}{1 - 1.84322701z^{-1} + 1.27884077z^{-2} - 0.30211946z^{-3}} \tag{4.1}$$

The middle-frequency range filter is of band-pass type, with lower cutoff frequency of 400 Hz and higher of 1350 Hz. Its discrete-time transfer function given by Equation 4.2:

$$F_{mid}[z] = \frac{0.01 \cdot (9.0352 - 8.6479z^{-1} - 3.4275z^{-2} + 4.4409z^{-3} + 3.4275z^{-4} + 8.6479z^{-5} - 9.0352z^{-6})}{1 - 1.892652z^{-1} + 2.599724z^{-2} - 2.201339z^{-3} + 1.531801z^{-4} - 0.594503z^{-5} + 0.171189z^{-6}} \tag{4.2}$$

The higher-frequency range filter is of high-pass type, with cutoff frequency of 1150 Hz and discrete-time transfer function given by 4.3:

$$F_{high}[z] = \frac{0.11416988 - 0.13894442z^{-1} + 0.13894442z^{-2} - 0.11416988z^{-3}}{1 + 1.03760923z^{-1} + 0.65334312z^{-2} + 0.10950528z^{-3}} \tag{4.3}$$

It is useful to note that the band-pass filter of Equation 4.2 contains much more terms than the low and high-pass filters of Equations 4.1 and 4.3, respectively. The reason for that is the fact that the band-pass filter is implemented by a cascade association of a low-pass and a high-pass.

Figure 4.3 depicts the frequency response of the three filters designed, in the form of gain and phase versus frequency. Figure 4.4 depicts the outputs of these filters for an input that consists of the a axis of a vibration measurement.

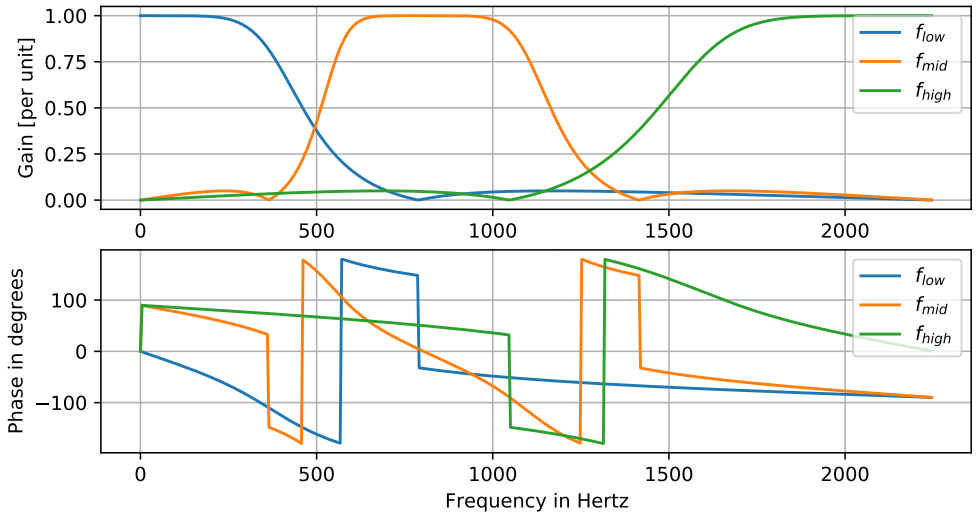

Figure 4.3: *Frequency response of the three signal filters designed*

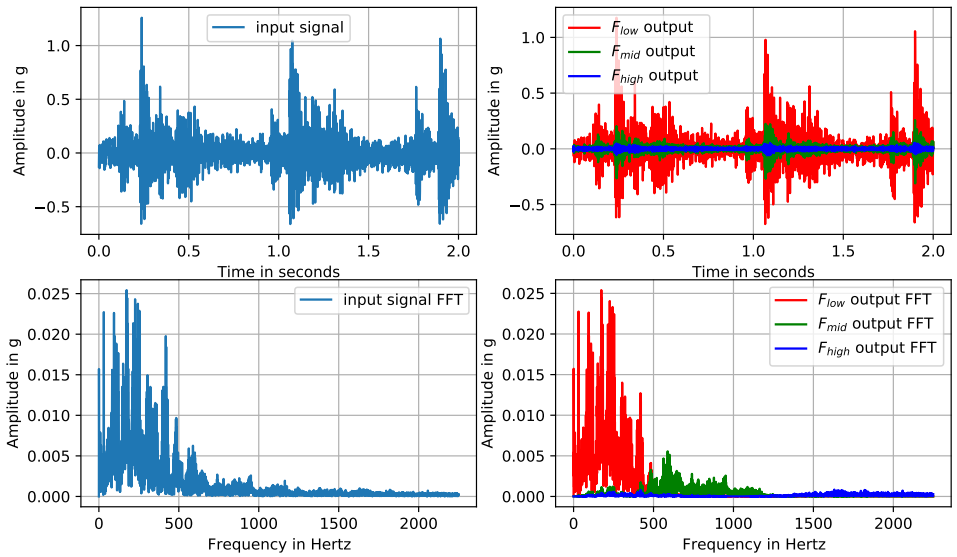

Figure 4.4: *Analysis of the output of the three signal filters designed to an example vibration input*

4.4 Feature engineering

In order to design the features to be used in the clustering and anomaly detection models, one must take some points into consideration. Those points were already summarized throughout Section 3. Now, we make use of these generalized ideas about statistical methods to establish the following guidelines for the clustering and anomaly detection models for the work at hand:

- **Reduced number of features:** A higher number of features results in a higher model complexity. This has the main drawback of increasing the requirements on the user side of familiarity with machine learning techniques. This is highly undesirable. The first reason for this is that such system is likely to perform poorly in real-world scenario with an uncontrolled environment. It also makes the end solution commercially unfeasible, as clients avoid aquiring, installing and operating systems they have innadequate understanding of.

- **Features should carry physical meaning:** As the model is based on unlabeled data, there is a higher effort on the system design side to make use of both domain-knowledge and previous knowledge about the data. Also, the physical meaning of the data is very useful for the end user to try to find what physical anomaly in the

machine can be associated to the data anomaly reported by the model.

- **Features must be feasibly obtainable:** Several limitations might hinder the suitability of a given potential feature, in different levels. First, there are the limitations imposed by the sensors used, such as physical quantity measured (pressure, vibration, etc) and sampling frequency, for example. One must also take into accound the cost associated with acquiring and processing the data that constitutes the features. In the current case, the frequency spectrum information obtained by means of Fourier Transform is particularly rich for means of PM. However, it requires much more computational effort (and therefore battery energy) than aglomerating a large range of frequencies under a single digital filter.

Under these guidelines, the first variable elected to be included in the models is the motor speed, which can be obtained by a range-restricted Fourier Transform, as explained in Section 4.2.

The second elected variable was the total vibration of the a axis, expressed as the RMS value of the referred signal.

The three other variables to be elected were the RMS values of the output of the digital filters designed in Section 4.3. The main purpose of these variables is to help the end user identify the physical source of a problem indicated by a high general vibration level.

More specific details about how to use these variables to build features for machine learning models are given in Section 4.5 and Section 4.6.

4.5 Clustering

In order to understand the underlying patterns in the data, the first step was to carry out a Principal Component Analysis on the dataset. The input to this analysis contained five features:

- f_{motor}: peak frequency in the 25Hz to 35 Hz range
- RMS_{total}: RMS value of the a axis
- RMS_{low}: RMS value of the output of the f_{low} filter
- RMS_{mid}: RMS value of the output of the f_{mid} filter
- RMS_{high}: RMS value of the output of the f_{high} filter

The resulting PCA plot is presented in Figure 4.5. This same plot also shows a seggregation of the points based on the results of a Kmeans clustering procedure. The KMeans

implementation employed makes use of Lloyd's heuristic and kmeans++ initial centroid assignment.

While analysing the measurements assigned to each cluster, it was noted that one of the clusters was characterised by lower vibration levels in general than the other one. This lead to the conclusion that the former represents the "operational" state for when the machine is off, and the later for when the machine is on. This conclusion is supported by the region with large variation, encircled in a green dashed shape. When the machine is off, the motors are not running, so they do not produce the 30 Hz peak, thus the peak in the 25 Hz to 35 Hz range ends up being mostly randomly defined by noise.

In order to confirm that the referred variation corresponds to the f_{motor} variable, a second PCA was carried out, this time not including f_{motor}. The result can be seen in Figure 4.6, which does not present the referred variation pattern.

This alternative dataset without f_{motor} serves the purpose of illustrating the effect of the referref variable only. For all other purposes, the dataset used contains f_{motor} and corresponds to the PCA and clustering presented in Figure 4.5.

A more detailed comparison between the two clusters is presented in the histograms of Figure 4.7.

The "elbow" plot for the referred data is depicted in Figure 4.8, where the "elbow"

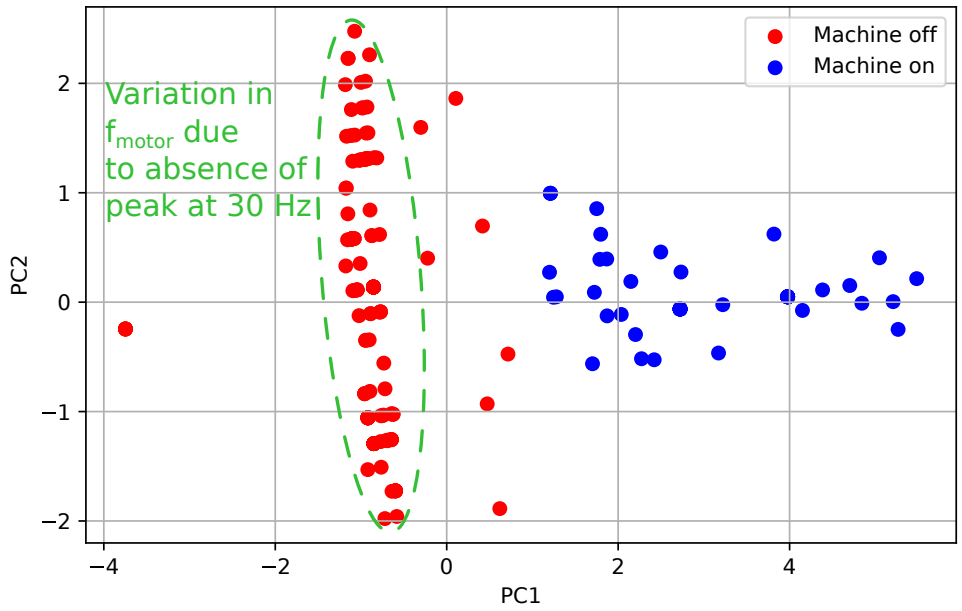

Figure 4.5: *PCA plot for the described dataset*

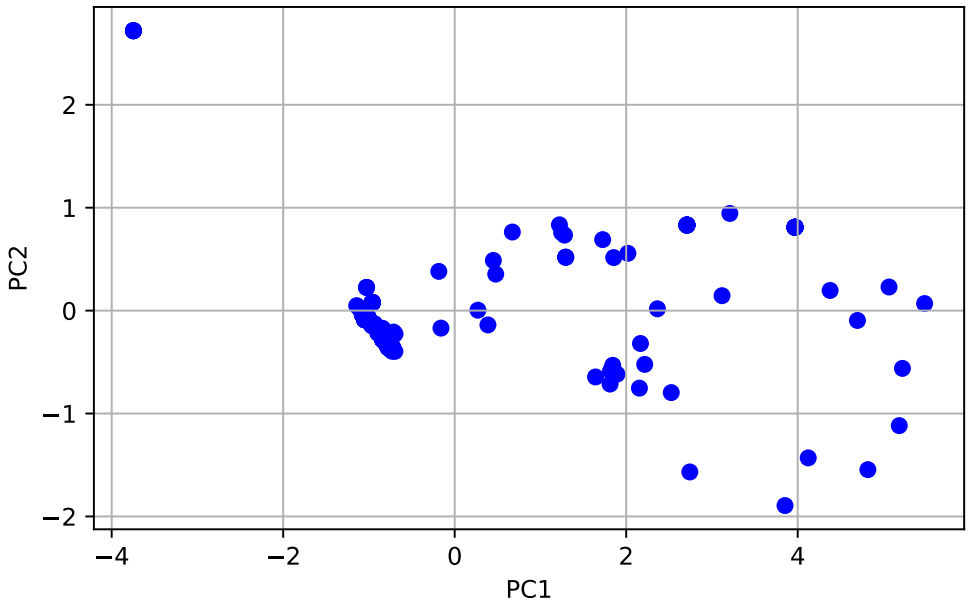

Figure 4.6: *Alternative PCA plot without* f_{motor}

shape can be seen for a number of clusters equal to 2.

When the machine is off, its vibration signature is of low relevance for PM. Thus, the analysis from this point on focuses on the cluster that represents the machine in its "on" state. The segregation can be easily carried out by thresholding the RMS_{total} value. This threshold value was set to $0.015g$. It is worth at this point to recap that the a axis signal has its average value removed, so the $1g$ gravity vibration is not present in the signal.

4.6 Concept of Anomaly Detection algorithm

At this point in the data analysis, the data is already clustered, and a single cluster was selected for further analysis. Thus, the variability left in the data is expected not to result in multimodal distributions for a given variable. In order to implement the Multi-Variate Gaussian Distribution AD strategy discussed in Section 3.4.1, it is required that the variables are Gaussian-distributed. In order to make sure this is the case, we apply a normalization transform to the data unsing the quantile transform described in Section 3.4.1.

Figure 4.9 depicts the variables before and after the transformation.

Then, for the actual AD approach, it was decided to build a concept based on the evaluation of the MVGD probability for four distinct models:

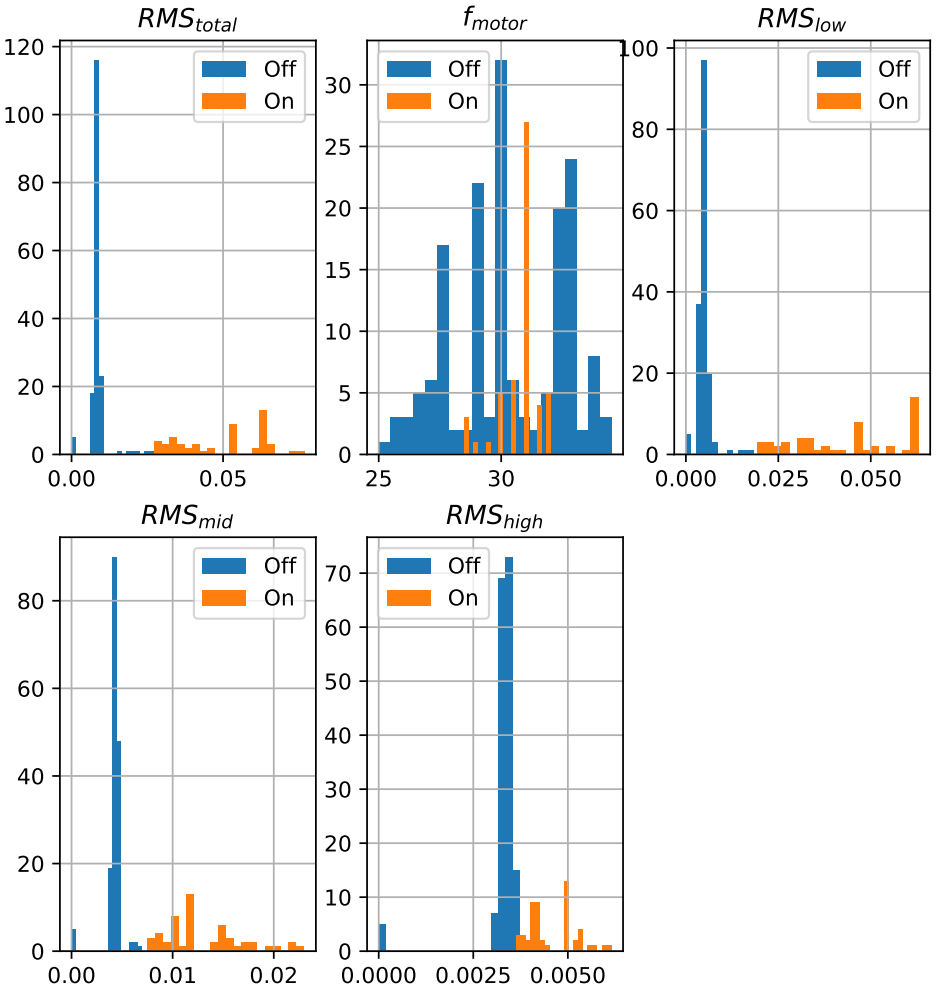

Figure 4.7: *Histogram comparison for the distribution of the variables in the two clusters*

1. Covariance between f_{motor} and RMS_{total}
2. Covariance between f_{motor}, RMS_{total} and RMS_{low}
3. Covariance between f_{motor}, RMS_{total} and RMS_{mid}
4. Covariance between f_{motor}, RMS_{total} and RMS_{high}

The "normality" values for the four covariance models designed are then tracked over time in days in Figure 4.10. As it can be seen in the plots, it is, unfortunately, not

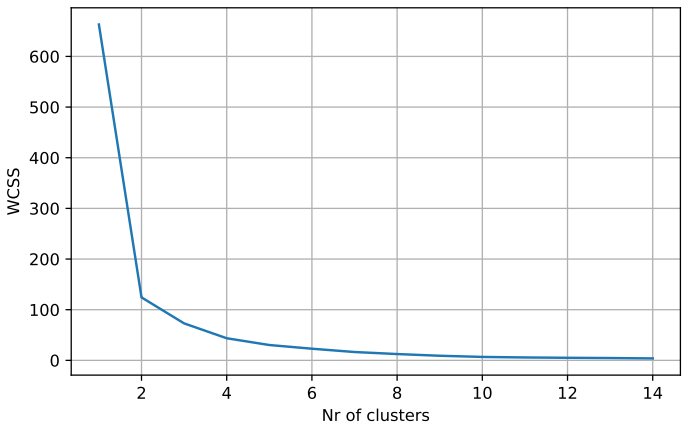

Figure 4.8: *"Elbow" plot for the clustering procedure*

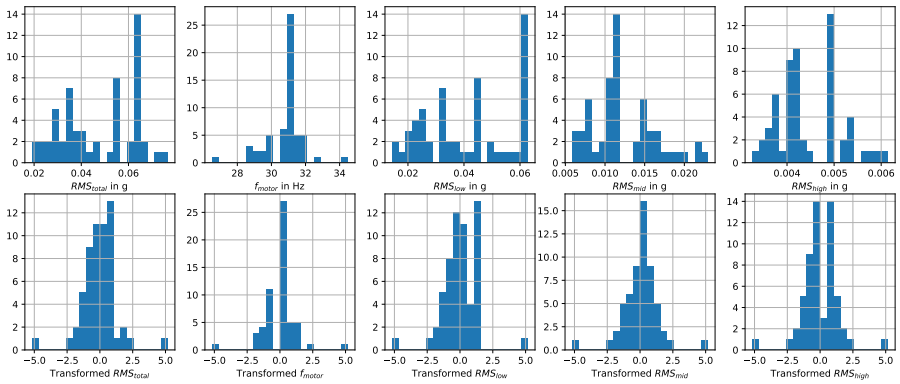

Figure 4.9: *Variables before and after quantile transform*

possible to establish a significant deterioration trend or any clearly alarming anomaly. The main cause for this problem is the fact that there are long periods of time without available measurements. With additional data regularly aquired, it should be possible to detect trends more clearly.

The resulting "normality" values are also depicted in sequence from Figure 4.11 to Figure 4.14, each of them presenting a 3D-heatmap plot for each of the covariance models designed.

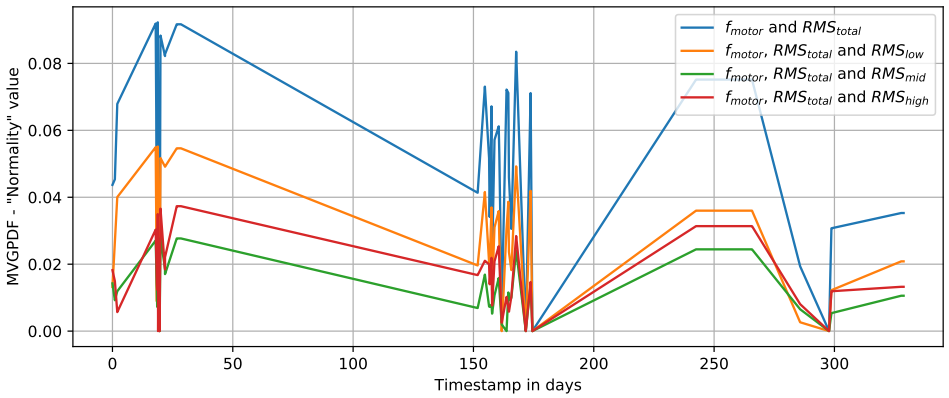

Figure 4.10: *Time-evolution of the "normality" values for the different covariance models designed*

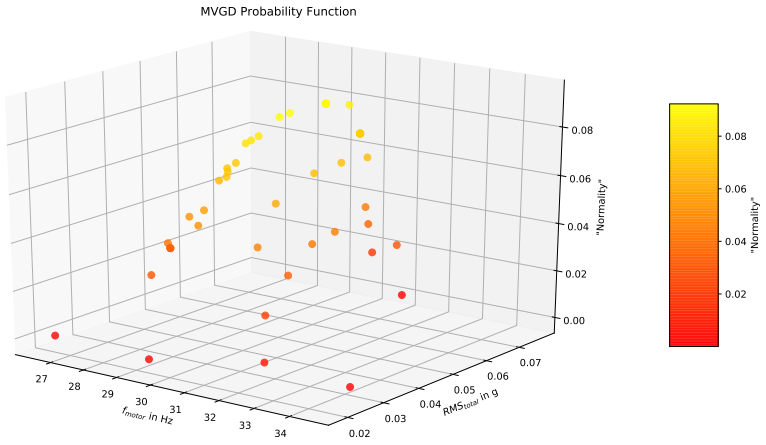

Figure 4.11: *MVGD plot for f_{motor} and RMS_{total}*

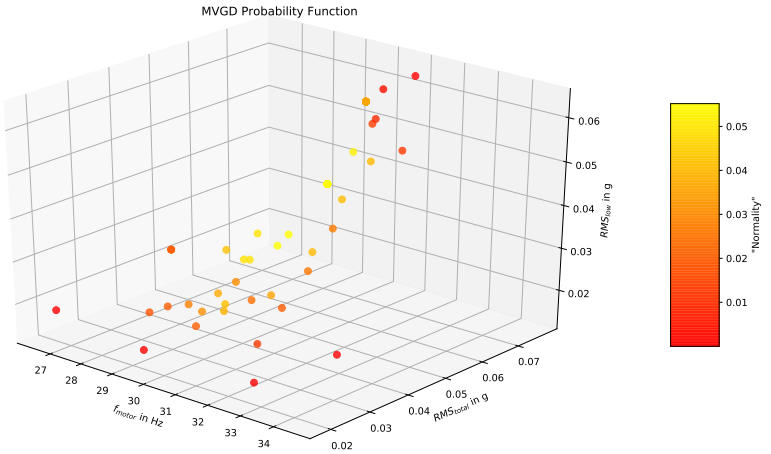

Figure 4.12: *MVGD plot for f_{motor}, RMS_{total} and RMS_{low}*

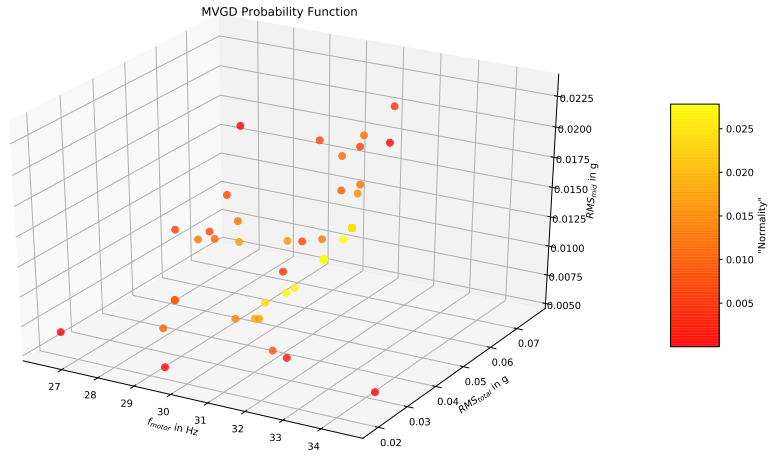

Figure 4.13: *MVGD plot for f_{motor}, RMS_{total} and RMS_{mid}*

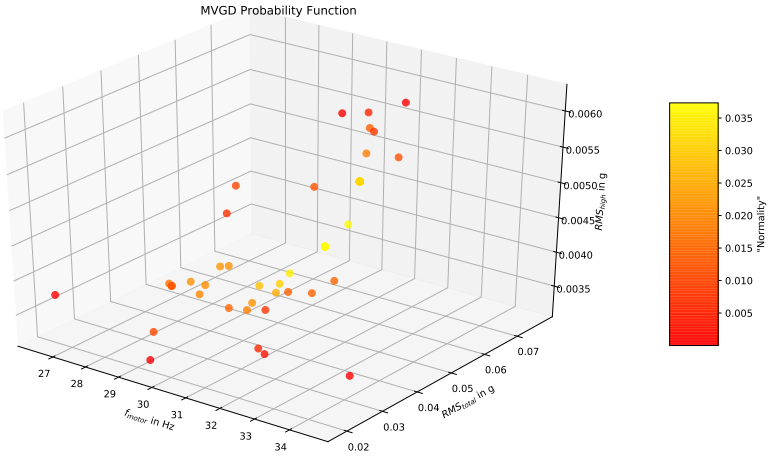

Figure 4.14: *MVGD plot for f_{motor}, RMS_{total} and RMS_{high}*

CHAPTER 5

Data processing on the Edge

In this work, in addition to developing the algorithms for analysis of the vibration data, we also aim to optimize the system in two aspects: data throughput and battery autonomy. These two aspects are conflicting, as sending data through radio consumes a significant ammount of energy.

A potential solution for this problem is processing data in the elementary sensors. Then, the sensors transmit only the desired features from the data, and not the whole measurement signal. However, due to limitations in computational power and memory of the microcontroller, the electronic sensors cannot feasibly perform the all the computations employed in this work so far. The gateway computer, however, is powered by cable and presents a fully fledged Linux system capable of running Python scripts, interfacing databases, connecting to internet, etc. Thus, a delegation of the software tasks among the different hardware devices has to be taken with the limitations and capabilities of each device in mind.

In order for the sensor to achieve its task extracting the data features from the vibration signals, it must employ the following techniques:

- Fourier Transform
- Digital Filters

All other tasks should then be delegated to the gateway:

- K-Means clustering classification
- Multi-Variate Gaussian Probability computation
- Dashboard application, which will be explained in larger details in Section 6

Finally, user-side desktop computers can be used to connect to the gateway and access the system's graphical user interface.

This delegation of tasks is illustrated in Figure 5.1.

The largest challenge in this part of the work is then enabling the sensor's microcontroller to perform the computations required for the Fourier Transform and for the Digital Filters. This section then first presents some general guidelines for scientific computations in microcontroller systems, then describes how the referred techniques were implemented in the electronic sensor used in the project.

Next, some calculations are presented to determine the battery power saved by the present strategy of sending only the data features instead of the whole vibration signals.

Finally, in the chapter's conclusion, it is presented a discussion about the next steps required for installing the recently developed signal processing features into the electronic sensors on site.

5.1 General optimization approaches for numeric computations in microcontroller systems

Due to the transistor-level implementation of digital computers, some mathematical operations in such systems are performed faster than others [Whi11]. For example, addition and subtraction are fast, i.e., they are accomplished in a small number of system clock cycles. Multiplication is usually based on consecutive additions, so it takes longer. A division, however, is different from these other operations, as it is not performed by dedicated hardware. It is usually based on library function calls, and is therefore much slower

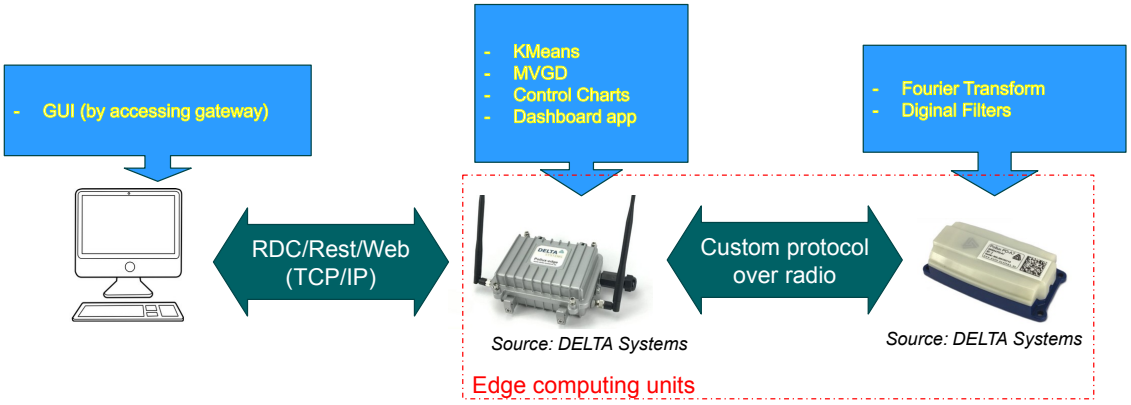

Figure 5.1: *Delegation of SW tasks among the HW components of the Pollux Vibration Monitoring System based on the capabilities and restrictions of each HW device*

[Whi11].

Similarly to the division operation, floating-point number operations are also not built into hardware. Floating-point arithmetic is specially slow in comparison to integer arithmetic. Thus, the substitution of floating-point operations by integer-based ones which deliver similar results can then deliver high code execution speed gains.

Also, given the nature of binary integer number representation, a multiplication by two is equivalent to the operation of a bit-shift to the left, which is computationaly fast. Likewise, a division by two is equivalent to the also fast operation of a bit-shift to the right. This reasoning can be expanded to any power of two. For example, a division by 16, which is equal to 2^4, is equivalent to four bit-shifts to the right.

The approaches described so far can then be combined in order to yield larger execution speed gains. For example, suppose a given mathematical expression requires a multiplication by 0.247. This operation can be calculated (or approximated) in different ways. A first approach would be a multiplication operation by the floating-point number 0.247. This approach, while direct, is much slower in comparison to the alternatives which will be presented shortly.

A second approach for the referred multiplication is represented by a multiplication by 247 followed by a division by 100. The absence of floating-point operation should already provides significant a significant speed increase.

However, a third approach is definitely faster. It consists of two bit-shifts to the right, which is equivalent to a division by 4, or a multiplication by 0.250. Note that the result is an approximation, with the error corresponding to $100\% \cdot (250 - 247)/247 \approx 1.2\%$.

The exact measure of how slower the first approach would be in comparison with the last one in an actual hardware execution is difficult to know beforehand. It largely depends on many factors such as hardware architecture and compilation process configurations [SL13]. These approaches can, however, be reliably compared by actually implementing them in the target computer system and measuring the time execution for a single operation, or for the whole series of calculations desired, or by many loops of the specific portion of the code which one aims to optimize.

Another important technique for the implementation of mathematical functions in computer systems is the use of approximations for functions of difficult computation, such as the trigonometric functions. Such approximations can be done, for example, in the form of polynomial regression models [Mor92].

5.2 Fourier Transform

The Fourier Transform was introduced in Section 3.2.1. In this section, we present a an approximation of the Fourier Transform which can be implemented in a microcontroller system delivering satisfactory results at a reasonable computational cost.

We start by reexamining Equation 3.2, which is a function of angular frequency ω. We then first need to find the mapping between ω and the sample indexes n. In order to do that, one must take into account that the Fourier Transform works by assigning each sample n out of N a frequency f_n expressed as in 5.1:

$$f_n = f_s \cdot n/N \tag{5.1}$$

where f_s is the sampling frequency of the signal.

Thus, we can replace ω by $2\pi n/N$ and then rewrite Equation 3.2 as a function of the index n, resulting in 5.2.

$$G(n) = \sum_{k=-\infty}^{\infty} g[k]e^{-jk\frac{2\pi n}{N}} \tag{5.2}$$

Now, let the auxiliary variable w_N be equal to $2\pi/N$. Then, substituting it in Equation 5.2 results in Equation 5.3:

$$G(n) = \sum_{k=-\infty}^{\infty} g[k]e^{-jw_N kn} \tag{5.3}$$

which can be rewritten as Equation 5.4:

$$G(n) = \sum_{k=-\infty}^{\infty} g[k]cos(w_N kn) - j\sum_{k=-\infty}^{\infty} g[k]sin(w_N kn) \tag{5.4}$$

Then, it can be observed that the product kn can be mapped to an angle, for when $nk = N$ we have that the exponential in Equation 5.3 represents a 2π rotation. Thus, let r_{kn} represent the remainder of nk/N, so that $cos(w_N r_{kn})$ and $sin(w_N r_{kn})$ are equal to, respectively, $cos(w_N kn)$ and $sin(w_N kn)$, and similarly for any other trigonometric function. This is useful because r_{kn} is always less then or equal to N. This property enables significant simplifications on the computations of the trigonometric functions in Equation 5.4, as will be explained in further details along this section.

Now, by means of the trigonometric identities given by Equations 5.5 to 5.7, it is possible to find the cosine and the sine of any angle $0 \leq \theta \leq 2\pi$ through the cosine of some angle in the first quadrant. This is also useful, as it enables a cosine representation for a single quadrant to generate the sines and cosines for any angle. Table 5.1 details

the equivalent first-quadrant-cosine for all the four quadrants sine and cosines.

$$sin(\theta) = -sin(-\theta) \tag{5.5}$$

$$sin(\theta) = cos(\pi - \theta) \tag{5.6}$$

$$cos(\theta) = cos(-\theta) \tag{5.7}$$

As a next step, we aim to find an approximation for $cos(\theta)$, with $0 < \theta \leq \frac{\pi}{2}$. In order to avoid time-consuming mathematical operations, it is advisable to avoid high-order polynomials and, most importantly, to avoid floating-point arithmetic. Thus, it was decided to adopt an 8-bit resolution for the 0 to 1 range of cosine values, so that the value 1 corresponds to the decimal representation of 255. Furthermore, it was decided to use piecewise-first-order polynomials for the considered range.

Considering that the signals we work with have $N = 9000$ samples, we have that the angle 2π corresponds to sample number 9000, and thus the first quadrant, which goes up to $\pi/2$ radians, equivalently, goes up to sample number 2250. Then, Figure 5.2 depicts our "computer-friendly" first-quadrant mapping of the cosine function, which we call $repcos(r_{kn})$, with the horizontal axis representing the r_{kn} number varying from 0 to 2250, and with the vertical axis representing the scaled cosine value, varying from 0 to 255. The same figure also shows the piecewise-line approximation for the same range, which is mathematically described in Equation 5.8.

Quadrant	**Equivalent cos(θ)**	**Equivalent sin(θ)**
1: $0 < \theta \leqq \frac{\pi}{2}$	$cos(\theta)$	$cos(\frac{\pi}{2} - \theta)$
2: $\frac{\pi}{2} < \theta \leqq \pi$	$-cos(\pi - \theta)$	$cos(\theta - \frac{\pi}{2})$
3: $\pi < \theta \leqq \frac{3\pi}{2}$	$-cos(\theta - \pi)$	$-cos(\frac{3\pi}{2} - \theta)$
4: $\frac{3\pi}{2} < \theta \leqq 2\pi$	$cos(2\pi - \theta)$	$-cos(\theta - \frac{3\pi}{2})$

Table 5.1: *Equivalent first-quadrant-cosine representation for cosines and sines for θ between 0 and 2π*

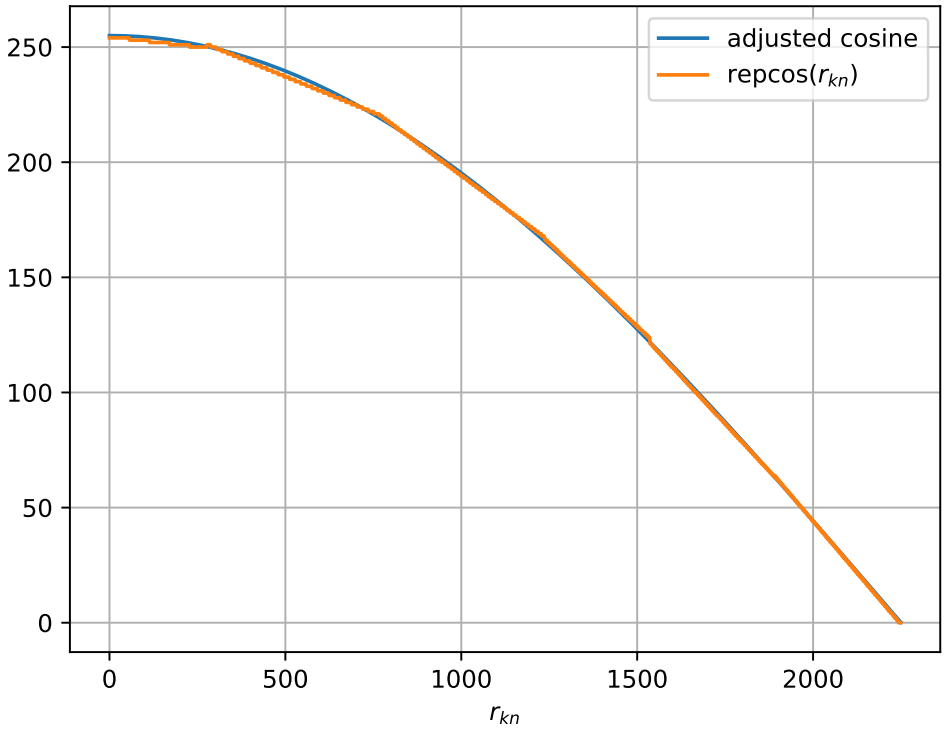

Figure 5.2: *Graph of* $repcos(r_{kn})$ *for* $0 < n \leq N/4$

$$
\begin{aligned}
repcos(r_{kn}) &= 254 - \frac{9 \cdot r_{kn}}{2^9}, \quad for \quad 0 \leq r_{kn} \leq 276 \\
&= 268 - \frac{16 \cdot r_{kn}}{2^8}, \quad for \quad 277 \leq r_{kn} \leq 773 \\
&= 307 - \frac{29 \cdot r_{kn}}{2^8}, \quad for \quad 774 \leq r_{kn} \leq 1236 \\
&= 345 - \frac{37 \cdot r_{kn}}{2^8}, \quad for \quad 1237 \leq r_{kn} \leq 1536 \\
&= 373 - \frac{21 \cdot r_{kn}}{2^7}, \quad for \quad 1537 \leq r_{kn} \leq 1891 \\
&= 403 - \frac{23 \cdot r_{kn}}{2^7}, \quad for \quad 1892 \leq r_{kn} \leq 2250
\end{aligned}
\tag{5.8}
$$

At this point, we are able to show in Figure 5.3 the plot of functions $repcos(r_{kn})$ and $repsin(r_{kn})$ for $0 < r_{kn} \leq N$, i.e, for all quadrants, by using Equation 5.8 and Table 5.1.

Finally, considering the functions $repcos(r_{kn})$ and $repsin(r_{kn})$ recently described, and also considering that the finite-length input signal to our Fourier transform is 0 for any

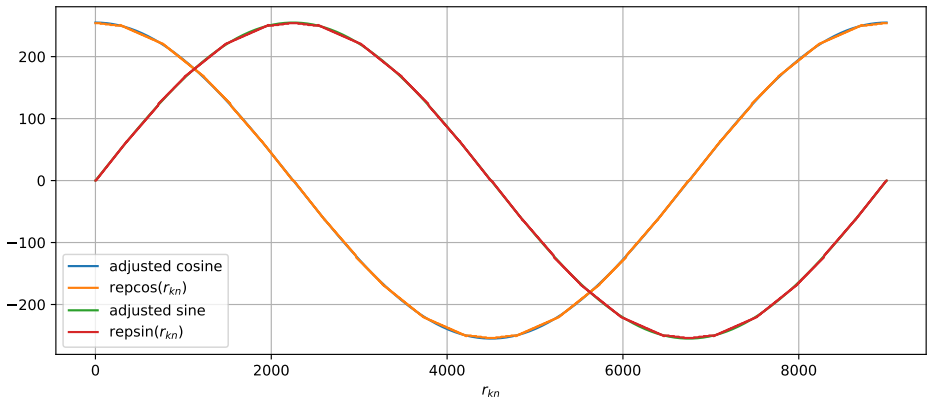

Figure 5.3: *Graph of* $repcos(r_{kn})$ *and* $repsin(r_{kn})$ *for* $0 < r_{kn} \leq N$

value before $k = 0$ and after $k = N - 1$, Equation 5.4 can be approximated by Equation 5.9.

$$G(n) \approx \sum_{k=0}^{N-1} g[k] \frac{repcos(r_{kn})}{2^8} - j \sum_{k=0}^{N-1} g[k] \frac{repsin(r_{kn})}{2^8} \tag{5.9}$$

As explained in Section 3.2.1, the DFT of a finite-time signal can be computed with less calculations by means of the FFT. The approximation in Equation 5.9 serves the FFT calculations just as well as for the DFT. However, FFT's requirement of parsing the input data in a non-sequenatial manner is problematic for an implementation in the microcontroller at hand.

The referred microcontroller has only a few kilobytes of RAM. The input signal, having 9000 samples, with each sample containing 6 bytes, does not fit into the device's RAM. Thus, the input signal is stored in an external memory chip, which is accessed through an SPI bus. This external memory chip is optimized for sequential read and write operations. Reading the bytes out of sequence, although possible, results in much slower read operations, in addition to increasing the software engineering efforts required to manage the parsing of different memory blocks inside the external memory chip. For these reasons, it was decided to implement the normal DFT procedure instead of an FFT algorithm.

Given that the complexity of the DFT computation is $O(N^2)$, the 9000 samples would require $9000^2 = 81 \cdot 10^6$ calculations of n and k term combinations in Equation 5.9. Such a high number of calculations would take a prohibitivelly long time on the microcontroller system, in addition to not having enough RAM to store the results.

As described in Section 4.2, however, the frequence range of interest in the DFT is only

the range which contains the machine's motor rotation frequency, which goes only from 25 Hz to 35 Hz. For a sampling frequency $f_s = 4500Hz$ and a number of samples $N = 9000$, Equation 3.3 determines that the frequency resolution $\Delta f = 4500Hz/9000 = 0.5Hz$. Thus, the range 25 Hz to 35 Hz corresponds to calculating $G(n)$ in Equation 5.9 only for $n = (25Hz/0.5Hz) = 50$ to $n = (35Hz/0.5Hz) = 70$.

As a test of the computer-friendly Fourier Transform proposed in Equation 5.9, we used it on an example input signal. The result is illustrated in Figure 5.4. As it can be seen from the illustration, the proposed FT approximation satisfactorily matches the standard floating-point implementation of the FFT algorithm from Python's Numpy library.

5.3 Digital Filters

Unlike the FT, digital filters lend themselves easily to implementation in microcontroller systems. As per the filter design example presented in Section 3.2.2, the discrete-time transfer equations from filters f_{low}, f_{mid} and f_{high}, presented in Equations 4.1 to 4.3 can be writen in difference equation form, which can then be implemented in microcontroller systems.

Equation 5.10 to presents the difference equations for f_{low}. In this equation, the array v_i represents the input signal samples, which is the a vibration axis in our case. The v_o array represents the output of the filter. These arrays are implemented in ring-buffer form, with all entries being initialized with the value zero.

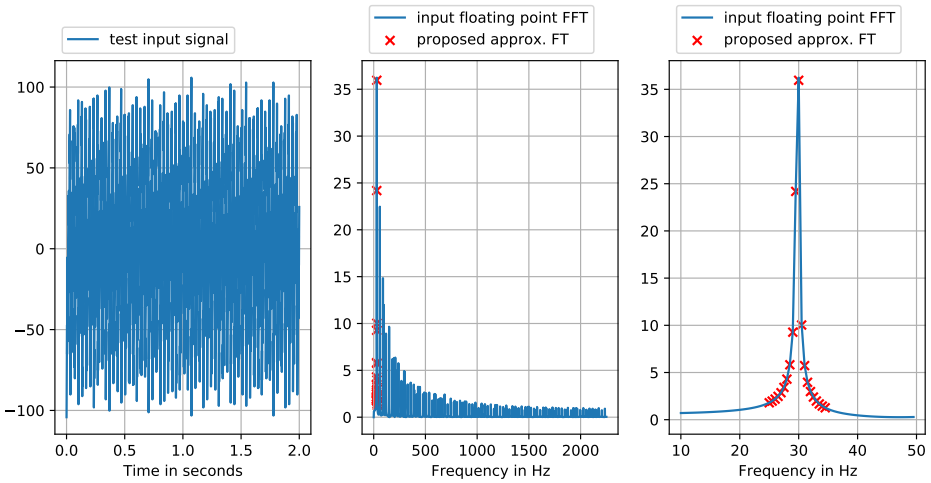

Figure 5.4: *Test of proposed FT: comparison with floating-point implementation*

$$v_o[n] = 0.060954 \cdot v_i[n] + 0.005793 \cdot v_i[n-1] + 0.005793 \cdot v_i[n-2] + 0.060954 \cdot v_i[n-3] \tag{5.10}$$
$$+ 1.843227 \cdot v_o[n-1] - 1.278841 \cdot v_o[n-2] + 0.302119 \cdot v_o[n-3]$$

Then, by employing the optimization approaches in 5.1, the difference equation of f_{low} is approximated as the "computer-friendly" expression in Equation 5.11:

$$v_o[n] \approx \frac{16}{2^8} \cdot v_i[n] + \frac{3}{2^9} \cdot v_i[n-1] + \frac{3}{2^9} \cdot v_i[n-2] + \frac{16}{2^8} \cdot v_i[n-3] \tag{5.11}$$
$$+ \frac{15}{2^3} \cdot v_o[n-1] - \frac{10}{2^3} \cdot v_o[n-2] + \frac{19}{2^6} \cdot v_o[n-3]$$

Similarly, Equation 5.12 presents the difference equation for f_{mid}, which is the approximated as the optimized form in 5.13:

$$v_o[n] = 0.090352 \cdot v_i[n] - 0.086479 \cdot v_i[n-1] - 0.034275 \cdot v_i[n-2] + (0.000000) \cdot v_i[n-3] \tag{5.12}$$
$$+ 0.034275 \cdot v_i[n-4] + 0.086479 \cdot v_i[n-5] - 0.090352 \cdot v_i[n-6]$$
$$+ 1.892652 \cdot v_o[n-1] - 2.599724 \cdot v_o[n-2] + 2.201339 \cdot v_o[n-3]$$
$$- 1.531801 \cdot v_o[n-4] + 0.594503 \cdot v_o[n-5] - 0.171189 \cdot v_o[n-6]$$

$$v_o[n] \approx \frac{23}{2^8} \cdot v_i[n] - \frac{22}{2^8} \cdot v_i[n-1] - \frac{9}{2^8} \cdot v_i[n-2] \tag{5.13}$$
$$+ \frac{9}{2^8} \cdot v_i[n-4] + \frac{22}{2^8} \cdot v_i[n-5] - \frac{23}{2^8} \cdot v_i[n-6]$$
$$+ \frac{15}{2^3} \cdot v_o[n-1] - \frac{21}{2^3} \cdot v_o[n-2] + \frac{18}{2^3} \cdot v_o[n-3]$$
$$- \frac{12}{2^3} \cdot v_o[n-4] + \frac{19}{2^5} \cdot v_o[n-5] - \frac{44}{2^8} \cdot v_o[n-6]$$

Finaly, Equation 5.14 presents the difference equation for f_{high}, which is the approximated as the optimized form in 5.15:

$$v_o[n] = 0.114170 \cdot v_i[n] - 0.138944 \cdot v_i[n-1] + 0.138944 \cdot v_i[n-2] - 0.114170 \cdot v_i[n-3] \\ - 1.037609 \cdot v_o[n-1] - 0.653343 \cdot v_o[n-2] - 0.109505 \cdot v_o[n-3] \tag{5.14}$$

$$v_o[n] \approx \frac{29}{2^8} \cdot v_i[n] - \frac{36}{2^8} \cdot v_i[n-1] + \frac{36}{2^8} \cdot v_i[n-2] - \frac{29}{2^8} \cdot v_i[n-3] \\ - \frac{17}{2^4} \cdot v_o[n-1] - \frac{21}{2^5} \cdot v_o[n-2] - \frac{28}{2^8} \cdot v_o[n-3] \tag{5.15}$$

Figure 5.5 presents a frequency response comparison between the original difference equations and the approximated-optimized ones for a selection of frequencies. The comparions allows the conclusion that the approximations replicate the original frequency response with high accuracy.

5.4 Improvements in the tradeoff between battery autonomy and sensor throughput

As explained in Section 2.4, the measurements sent over radio are sampled at $4.5kHz$ and have a duration of $2s$. Each individual measurement point consists of three axes, each of them enconded in 16 bits, or 2 bytes. The number of bytes $N_{bytesmm}$ to be transmitted

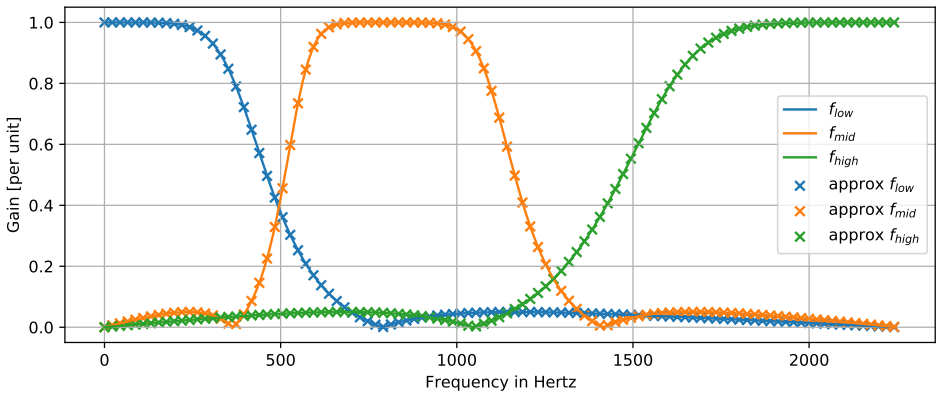

Figure 5.5: *Frequency responses for approximated difference equations for the filters designed*

in a measurement is then determined per Equation 5.17:

$$N_{bytesmm} = 4500 \cdot \frac{samples}{second} \times 2 \cdot seconds \times 3 \cdot \frac{axes}{sample} \times 2 \cdot \frac{bytes}{axis} = 54kbytes \quad (5.16)$$

A radio message in the protocol implemented in the system has 48 bytes of payload. The electronic sensor takes $30ms$ to transmit such a message, while drawing $18mA$ at $6V$. The energy $E_{bytetrans}$ required for the transmission of a byte of information can then be determined by Equation 5.17:

$$E_{bytestrans} = \frac{6V \cdot 18mA \cdot 30ms}{message} \times \left(48\frac{bytes}{message}\right)^{-1} \approx 67.5\mu Ws/byte \quad (5.17)$$

The energy E_{longmm} required to send a long measurement is then expressed as per Equation 5.18:

$$E_{longmm} = 54kbytes \times 67.5\mu Ws/byte \approx 3.64Ws \quad (5.18)$$

Through lab experiments, the time required for the execution of the Fourier Transform approximation from Section 5.2 for the selected frequency range and the digital filters from Section 5.3 for a $2s$-$4kHz$ measurement was determined as $17s$. During this time, the electronic sensor draws $5mA$ at $6V$. The energy $E_{sigprocess}$ required for the signal-processing-feature-extraction procedure is then determined as per Equation 5.19:

$$E_{sigprocess} = 6V \cdot 5mA \cdot 17s \approx 0.51Ws \quad (5.19)$$

Given that the signal processing procedure extracts five features (f_{motor}, RMS_{total}, RMS_{low}, RMS_{mid} and RMS_{high}), while also considering that each feature can be satisfactorily expressed in 2 bytes, the features can be contained in 10 bytes. This amount of information can be transmitted in the 48 bytes payload of a single radio message. A single message consumes $48bytes \times 67.5\mu Ws/byte \approx 3.24mWs$, which is negligible in comparison to either E_{longmm} or $E_{sigprocess}$.

Given that $E_{longmm}/E_{sigprocess} \approx 7.14$, the signal processing approach promotes a seven times reduction in the energy cost per measurement.

An additional energy cost saving takes place by rejection of measurements for which the machine is off. During the signal processing computations in the sensor, it is possible to estimate RMS_{total} by considering only the first half of the samples. Then, in case the resulting RMS_{total} is below the $0.015g$ threshold defined in Section 4.5, it is possible to quit the data processing and save the energy that would be wasted in a measurement considered irrelevant for our model.

5.5 Sensor firmware update

The signal processing functions described in this section have been sucessfully implemented in lab tests with a sensor of the same model as the ones operating in the field.

The field sensors, however, still have not received an update with the new functionalities. The reasons for that include logistic and contractual issues with the client company that aquired the sensors which are beyond the scope of this work.

Therefore, the update of the field sensors and the training of the models with new field data are left as future steps in the continuation of the work developed in this thesis.

CHAPTER 6

Dashboard

In order to help the end-user of the vibration monitoring solution better understand the data output of the system, a dashboard for data visualization was conceptualized.

This dashboard, illustrated in Figure 6.1, presents a heatmap plot of one of the variables tracked by the system over time. The slider allows the user to observe the time evolution of the values tracked and associate patterns to the spacial position of the sensors on the machine. The sensors are represented by the heatmap boxes.

The dashboard is developed with the Python library Plotly, which enables easy integration with the software infrastructure of the vibration monitoring system.

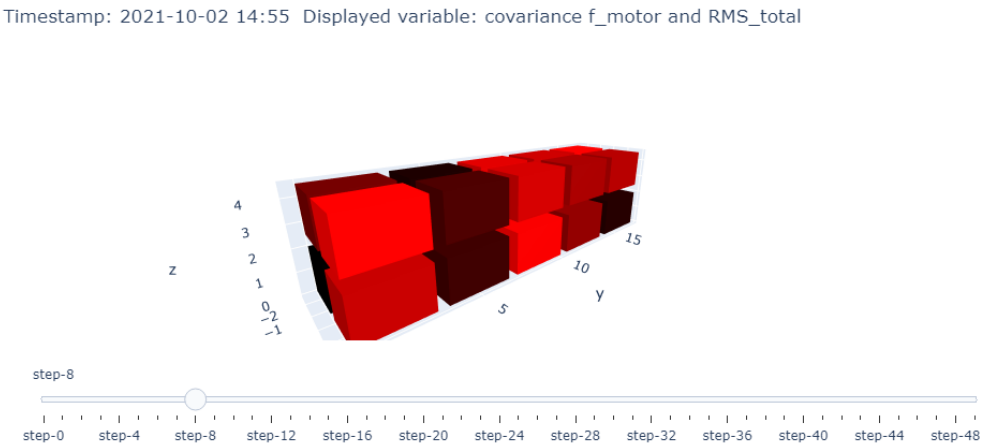

Figure 6.1: *Concept of the proposed dashboard*

CHAPTER 7

Conclusion

In this work, we designed improvements to a vibration monitoring system which currently operates in a textile manufacturing plant in Turkey.

First, a research on a range of technologies was carried out in order to understand how to best analyse vibration signals.

Then, a clustering and an Anomaly Detection model were developed and tested in the vibration signals from the system's database. Although it could be confirmed that the models are functional from a data pipeline perspective, more conclusive results about their performance require more data.

In order to facilitate the aquisition of more data with a same hardware setup, signal processing techniques were implemented to enable the field sensors to process their aquired measurements autonomously. This approach represents a more intelligent way of constructing the data pipeline requiring less maintenance for the sensors.

Finally, a dashboard was developed to help end-users of the system to make sense of the results produced. This knowledge should then help them optimize the maintenance schedules of the equipment they work with.

The work developed in this Thesis should be continued by an update of the vibration monitoring system which will add the functionalities developed here developed. Then, with the aquisition of new data, the Anomaly Detection models can potentially be tuned to detect deterioration trends in the machinery more clearly.

Bibliography

[ADH+09] D. Aloise, A. Deshpande, P. Hansen, et al.: *NP-hardness of Euclidean sum-of-squares clustering*. In: *Machine Learning*. Vol. 75. 2009, pp. 245–248. DOI: 10.1007/s10994-009-5103-0.

[ASI20] M. Ahmed, R. Seraj, and S. M. S. Islam: *The k-means algorithm: A comprehensive survey and performance evaluation*. In: *Electronics* 9.8 (2020), p. 1295.

[AV07] D. Arthur and S. Vassilvitskii: *k-means++: the advantages of careful seeding*. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. 2007, pp. 1027–1035.

[BG12] H. P. Bloch and F. K. Geitner: *Machinery Failure Analysis and Troubleshooting: Practical Machinery Management for Process Plants*. Vol. 2. Elsevier, 2012.

[Car06] M. Carnero: *An evaluation system of the setting up of predictive maintenance programmes*. In: *Reliability Engineering & System Safety* 91.8 (2006), pp. 945–963.

[CNG15] K. Chong, K. Ng, and G. Goh: *Improving Overall Equipment Effectiveness (OEE) through integration of Maintenance Failure Mode and Effect Analysis (maintenance-FMEA) in a semiconductor manufacturer: A case study*. In: *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*. 2015, pp. 1427–1431. DOI: 10.1109/IEEM.2015.7385883.

[CT65] J. W. Cooley and J. W. Tukey: *An algorithm for the machine calculation of complex Fourier series*. In: *Math. Comput.* 19.2 (1965).

[Dev] *numpy.fft.fft (function documentation)*. URL: https://numpy.org/devdocs/reference/generated/numpy.fft.fft.html (visited on 07/10/2022).

[Jer77] A. Jerri: *The Shannon sampling theorem—Its various extensions and applications: A tutorial review*. In: *Proceedings of the IEEE* 65.11 (1977), pp. 1565–1596. DOI: 10.1109/PROC.1977.10771.

[Joh70] C. R. Johnson: *Positive definite matrices*. In: *The American Mathematical Monthly* 77.3 (1970), pp. 259–264.

[Kab16] S. Kaboli: *Reliability in power electronics and electrical machines: industrial applications and performance models: industrial applications and performance models*. IGI Global, 2016.

[KH01] R. Koenker and K. F. Hallock: *Quantile regression*. In: *Journal of economic perspectives* 15.4 (2001), pp. 143–156.

[LPdB06] C. M. F. Lapa, C. M. N. Pereira, and M. P. de Barros: *A model for preventive maintenance planning by genetic algorithms based in cost and reliability*. In: *Reliability Engineering & System Safety* 91.2 (2006), pp. 233–240.

[LZ06] I. D. Landau and G. Zito: *Digital control systems: design, identification and implementation*. Vol. 130. Springer, 2006.

[Mor92] D. Morgan: *Numerical Methods/Real-Time and Embedded Systems Programming*. Hungry Minds Inc, 1992.

[NW78] M. Neale and B. Woodley: *A guide to the condition monitoring of machinery*. Report TRD 223 for the British Department of Industry, 1978.

[Oak07] J. S. Oakland: *Statistical process control*. Routledge, 2007.

[Ran21] R. B. Randall: *Vibration-based condition monitoring: industrial, automotive and aerospace applications*. John Wiley & Sons, 2021.

[SL13] J. P. Shen and M. H. Lipasti: *Modern processor design: fundamentals of superscalar processors*. Waveland Press, 2013.

[TJ18] L. Tan and J. Jiang: *Digital signal processing: fundamentals and applications*. Academic Press, 2018.

[Tsy17] M. Tsypkin: *The Origin of the Electromagnetic Vibration of Induction Motors Operating in Modern Industry: Practical Experience—Analysis and Diagnostics*. In: *IEEE Transactions on Industry Applications* 53.2 (2017), pp. 1669–1676. doi: 10.1109/TIA.2016.2633946.

[Whi11] E. White: *Making Embedded Systems: Design Patterns for Great Software*. " O'Reilly Media, Inc.", 2011.

[WRR03] M. E. Wall, A. Rechtsteiner, and L. M. Rocha: *Singular value decomposition and principal component analysis*. In: *A practical approach to microarray data analysis*. Springer, 2003, pp. 91–109.

Acknowledgement

I would like to thank my Parents, Lourenzo and Safira, for always enabling me to study and pursue a path of continuous personal and professional improvement.

I also would like to thank my supervisors, M.Sc. Björn Eberhardt and M.Sc. Jiahang Chen, for their most-valuable guidance in the conduction of this work.

The present translation is for your convenience only.
Only the German version is legally binding.

Statutory Declaration in Lieu of an Oath

Loiola Costa, Victor Lorhan	390048
Last Name, First Name	Matriculation No. *(optional)*

I hereby declare in lieu of an oath that I have completed the Master Thesis entitled

Edge-based Vibration Monitoring System for Textile Machinery

independently and without illegitimate assistance from third parties. I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 18. Jul. 2022	The German version has to be signed.
Location/City, Date	Signature

Official Notification:

Para. 156 StGB (German Criminal Code): False Statutory Declarations

Whosoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

Para. 161 StGB (German Criminal Code): False Statutory Declarations due to Negligence

(1) If a person commits one of the offences listed in sections 154 to 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.

(2) The offender shall be exempt form liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

I have read and understood the above official notification:

Aachen, 18. Jul. 2022	The German version has to be signed.
Location/City, Date	Signature