UNIVERSITÀ DI PAVIA

Swansea University
Prifysgol Abertawe

Predictive Maintenance of an External Gear Pump using Machine Learning Algorithms

by

Kayalvizhi Lakshmanan

A Thesis submitted

in Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

at

Università degli Studi di Pavia, Italy

and

Swansea University, UK

Supervisors:

Prof. F Auricchio, Università degli Studi di Pavia, Italy

Prof. A J Gil, Swansea University, UK

July 2021

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ...... [redacted] .. (candidate)

Date .... 08/07/2021 ....

STATEMENT 1

This thesis is the result of my own investigations, except where otherwise stated. When correction services have been used, the extent and nature of the correction is clearly marked in a footnote(s). Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed .... [redacted] .... (candidate)

Date .... 08/07/2021 ....

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .... [redacted] .... (candidate)

Date .... 08/07/2021 ....

Acknowledgements

I would like to express my sincere gratitude to my supervisors Prof. Ferdinando Auricchio and Prof. Antonio J. Gil who have always motivated me and provided excellent supervision throughout the course of this work. I would also like to thank the company F-Lab and Dr. Fabrizio Tessicini for all the useful interactions we had during this project.

I am grateful to the European Commission EACEA Agency for providing me the financial assistance under the Framework Partnership Agreement 2013-0043 Erasmus Mundus Action 1b as a part of the EM Joint Doctorate Simulation in Engineering and Entrepreneurship Development (SEED). Moreover, this programme has given me the exposure and opportunity to undertake research in two top Universities in the field of computational mechanics which has consequently helped me to grow professionally.

I would also like to acknowledge the open source and commercial software community, in particular to MATLAB, Python, Keras, TensorFlow, PumpLinx, and LaTeX projects. I would like to thank the CFD online website for providing answers to CFD related questions and Python Forum and MaTLAB Answers for providing answers to programming related questions.

I take this opportunity to thank all my collegues in the Zienkiewicz Centre for the Computational Engineering (ZCCE), UK and Computational Mechanics & Advanced Materials Group (CompMech), Italy. I would like to thank my friends Nikitha, Radha Krishna, Kalyan, Chithra and Geethu for their support throughout my studies, and Jyostna, Vandana, Meena, and Anagha for the encouragement. I would like to also thank Osama, Francisco, Sanjay, Emilio, Sky, Marianna, Ken and Nidhal, who made my stay brighter in Swansea. I want to thank Davide for his support and for being a wonderful companion throughout this period. Last but certainly not least, I would like to thank my father Lakshmanan and my mother Usha for believing in me and letting me pursue my dreams. I want to thank my brother Manikandan for his support all these years. I also want to thank my uncle and his family for their endless concern and support.

Abstract

The importance of Predictive Maintenance is critical for engineering industries, such as manufacturing, aerospace and energy. Unexpected failures cause unpredictable downtime, which can be disruptive and high costs due to reduced productivity. This forces industries to ensure the reliability of their equipment. In order to increase the reliability of equipment, maintenance actions, such as repairs, replacements, equipment updates, and corrective actions are employed. These actions affect the flexibility, quality of operation and manufacturing time. It is therefore essential to plan maintenance before failure occurs.

Traditional maintenance techniques rely on checks conducted routinely based on running hours of the machine. The drawback of this approach is that maintenance is sometimes performed before it is required. Therefore, conducting maintenance based on the actual condition of the equipment is the optimal solution. This requires collecting real-time data on the condition of the equipment, using sensors (to detect events and send information to computer processor).

Predictive Maintenance uses these types of techniques or analytics to inform about the current, and future state of the equipment. In the last decade, with the introduction of the Internet of Things (IoT), Machine Learning (ML), cloud computing and Big Data Analytics, manufacturing industry has moved forward towards implementing Predictive Maintenance, resulting in increased uptime and quality control, optimisation of maintenance routes, improved worker safety and greater productivity.

The present thesis describes a novel computational strategy of Predictive Maintenance (fault diagnosis and fault prognosis) with ML and Deep Learning applications for an FG304 series external gear pump, also known as a domino pump. In the absence of a comprehensive set of experimental data, synthetic data generation techniques are implemented for Predictive Maintenance by perturbing the frequency content of time series generated using High-Fidelity computational techniques. In addition, various types of feature extraction methods considered to extract most discriminatory informations from the data. For fault diagnosis, three types of ML classification algorithms are employed, namely Multilayer Perceptron (MLP), Support Vector Machine (SVM) and

Naive Bayes (NB) algorithms. For prognosis, ML regression algorithms, such as MLP and SVM, are utilised. Although significant work has been reported by previous authors, it remains difficult to optimise the choice of hyper-parameters (important parameters whose value is used to control the learning process) for each specific ML algorithm. For instance, the type of SVM kernel function or the selection of the MLP activation function and the optimum number of hidden layers (and neurons).

It is widely understood that the reliability of ML algorithms is strongly dependent upon the existence of a sufficiently large quantity of high-quality training data. In the present thesis, due to the unavailability of experimental data, a novel high-fidelity in-silico dataset is generated via a Computational Fluid Dynamic (CFD) model, which has been used for the training of the underlying ML metamodel. In addition, a large number of scenarios are recreated, ranging from healthy to faulty ones (e.g. clogging, radial gap variations, axial gap variations, viscosity variations, speed variations). Furthermore, the high-fidelity dataset is re-enacted by using degradation functions to predict the remaining useful life (fault prognosis) of an external gear pump.

The thesis explores and compares the performance of MLP, SVM and NB algorithms for fault diagnosis and MLP and SVM for fault prognosis. In order to enable fast training and reliable testing of the MLP algorithm, some predefined network architectures, like 2^n neurons per hidden layer, are used to speed up the identification of the precise number of neurons (shown to be useful when the sample data set is sufficiently large). Finally, a series of benchmark tests are presented, enabling to conclude that for fault diagnosis, the use of wavelet features and a MLP algorithm can provide the best accuracy, and the MLP algorithm provides the best prediction results for fault prognosis. In addition, benchmark examples are simulated to demonstrate the mesh convergence for the CFD model whereas, quantification analysis and noise influence on training data are performed for ML algorithms.

Keywords: Fault diagnosis, Fault prognosis, Wavelet transform, Noise perturbation, Support Vector Machine, MultiLayer Perceptron, Naive Bayes

Research Output

Journal publications

- ☐ **K. Lakshmanan**, F. Tessicini, A.J. Gil, and F. Auricchio "Machine Learning Approach for the Predictive Maintenance methodology of an External Gear Pump", Journal paper [in preparation]
- ☐ **K. Lakshmanan**, F. Tessicini, A.J. Gil, and F. Auricchio, "Methodology and data generation method for fault prognosis using Machine Learning algorithms", Journal paper 2021 [in preparation]

Conference publications

- ☐ **K. Lakshmanan**, A.J. Gil, F. Auricchio, and F. Tessicini. "A fault diagnosis methodology for an external gear pump with the use of Machine Learning classification algorithms: Support Vector Machine and Multilayer Perceptron". In proceedings of the *UK Association for Computational Mechanics (UKACM)*, Loughborough, United Kingdom, 1-3 April, 2020. https://doi.org/10.17028/rd.lboro.12097668.v1
- ☐ **K. Lakshmanan**, A.J. Gil, F. Auricchio, and F. Tessicini. "A predictive maintenance methodology for an external gear pump with the use of Machine Learning classification algorithms: Support Vector Machine and Multilayer Perceptron". In proceedings of the *14th World Congress on Computational Mechanics and 8th European Congress on Computational Methods in Applied Sciences and Engineering, (WCCM-ECCOMAS)*, Virtual congress, 11-15 January, 2021.

Posters

☐ **K. Lakshmanan**, F. Auricchio, A.J. Gil, and F. Tessicini , "Fault diagnosis of an external gear pump using Machine Learning algorithms". Poster at The Annual Zienkiewicz Centre for Computational Engineering (ZCCE) Postgraduate Workshop, Swansea University, United Kingdom, 13-14 January, 2020.

☐ **K. Lakshmanan**, F. Auricchio, A.J. Gil, and F. Tessicini , "Predictive maintenance of an external gear pump using Machine Learning algorithms". Poster at The Annual Zienkiewicz Centre for Computational Engineering (ZCCE) Postgraduate Workshop, Swansea University, United Kingdom, 18 January, 2021.

"Logic will get you from A to Z;

imagination will get you everywhere.

Albert Einstein

CONTENTS

LIST OF FIGURES

Title **Page No.**

LIST OF TABLES

Title **Page No.**

ABBREVIATIONS

AI	**A**rtificial **I**ntelligence
ALE	**A**rbitrary **L**agrangian **E**ulerian
CAD	**C**omputer **A**ided **D**esign
CFD	**C**omputational **F**luid **D**ynamics
CWT	**C**ontinuous **W**avelet **T**ransform
DFT	**D**iscrete **F**ourier **T**ransform
DWT	**D**iscrete **W**avelet **T**ransform
DWPT	**D**iscrete **W**avelet **P**acket **T**ransform
FFT	**F**ast **F**ouier **T**ransform
FVM	**F**inite **V**olume **M**ethod
KKT	**K**arush **K**uhn **T**ucker
MODWT	**M**aximal **O**verlap **D**iscrete **W**avelet **T**ransform
MODWPT	**M**aximal **O**verlap **D**iscrete **W**avelet **P**acket **T**ransform
ML	**M**achine **L**earning
MLP	**M**ultilayer **P**erceptrons
NCG	**N**on-**C**ondensable **G**ases
NB	**N**aive **B**ayes
NN	**N**eural **N**etwork
PSD	**P**ower **S**pectral **D**ensity
ReLU	**Re**ctified **L**inear **U**nit
RUL	**R**emaining **U**seful **L**ife

RPM	**R**evolution **P**er **M**inute
SK	**S**pectral **K**urtosis
SVM	**S**upport **V**ector **M**achine
SPSD	**S**ymmetric **P**ositive **S**emi-**D**efinite
STFT	**S**hort **T**ime **F**ourier **T**ransform

NOMENCLATURE

$\mathbb{R}$	Real numbers
$\mathbb{C}$	Complex numbers
$\mathbb{H}$	Hilbert space

<u>Scalars</u>

t	Time	[s]
ω	Frequency	[Hz]
p	Pressure	[bar]
ρ	Fluid density	$\left[\mathrm{kg/m^3}\right]$
μ	Fluid viscosity	[Pa-s]
E	Total energy	[J]
r	Rate of volumetric heat generation	$\left[\mathrm{J/(K\ m^3)}\right]$
e	Internal energy	[J]
T	Temperature	[K]
R	Universal gas constant	[J/(mol K)]
f_v	Vapor mass fraction	$[-]$
α_v	Vapor volume fraction	$[-]$
σ_{f_v}	Turbulent schmidt number	$[-]$
μ_t	Turbulent kinetic energy	[J/kg]
ρ_v	Vapor density	$[-]$
ρ_l	Liquid density	$\left[\mathrm{kg/m^3}\right]$
α_v	Vapor volume	$[-]$
α_l	Liquid volume	$[\mathrm{m^3}]$

α_g	Gas volume	$[-]$
P	Cell centroid	$[-]$
N	Neighbouring cell	$[-]$
f	Face centre	$[-]$
d	Distance	$[\mathrm{m}]$
m_f	Mass flux	$\left[\mathrm{kg\ m^{-2}s^{-1}}\right]$
x	Variables	
ϵ	Noise	
$\mathcal{W}$	Set of frequencies	
$\tilde{\omega}$	Selected frequency	
$\hat{\mathcal{F}}$	Fourier transform	
$\hat{\mathcal{F}}^{-1}$	Inverse Fourier transform	
α	Scalar parameter	
m	Model parameter	
μ	Mean / average	
σ	Standard deviation	
s	Skewness	
k	Kurtosis	
sk	Spectral kurtosis	
μ_1	Spectral centroid	
μ_2	Spectral spread	
S_{xx}	Power spectral density	
$\mathcal{X}$	Input space	
$\mathcal{Y}$	Output space	
S	Training sample set	
$\mathcal{H}$	Hypothesis set	

b	Bias	
M_h	Geometric margin	
M	Maximum margin	
α_j	Lagrange variables	
ζ_j	Slack variables	
C, λ	Regularisation parameter	
Φ	Feature space	
$\mathcal{K}$	Kernel	
z	Net input	
ϕ	Activation function	

Vectors

$\boldsymbol{v}$	Velocity	[m/s]
$\boldsymbol{v}_s$	Mesh velocity	[m/s]
$\boldsymbol{\tau}$	Viscous shear stress	$\left[\mathrm{N/m^2}\right]$
$\boldsymbol{f}$	Body force	$\left[\mathrm{N/m^3}\right]$
$\boldsymbol{\sigma}$	Stress tensor	$\left[\mathrm{N/m^2}\right]$

$\boldsymbol{x}$	Input data
$\boldsymbol{y}$	Output data
$\hat{\boldsymbol{x}}$	Discrete representation
$\boldsymbol{\epsilon}$	Noise vector
$\boldsymbol{x}_h$	Healthy data
$\boldsymbol{x}_f$	Faulty data
$\boldsymbol{w}$	Weight vector
$\boldsymbol{\alpha}$	Lagrange multiplier

$\boldsymbol{X}$	Matrix representation of $\boldsymbol{x}$

$\boldsymbol{W}$	Weight matrix
$\boldsymbol{E}_j$	Unit vector

Chapter 1

INTRODUCTION

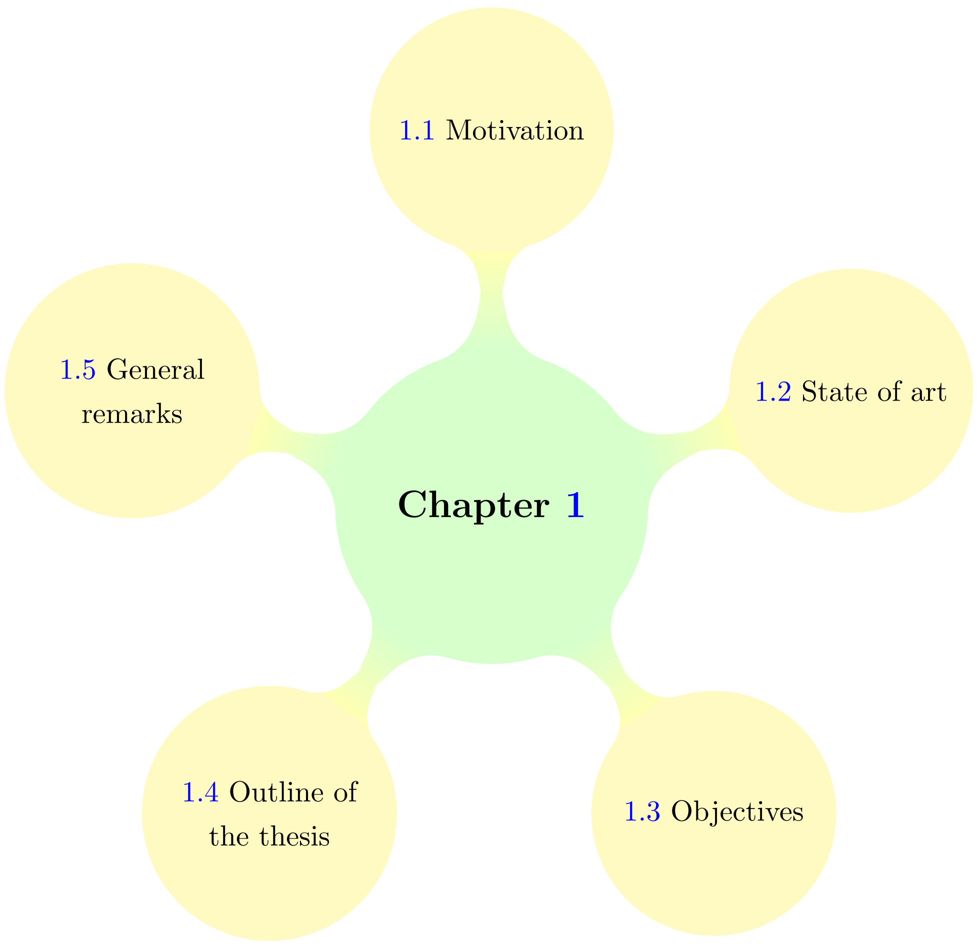


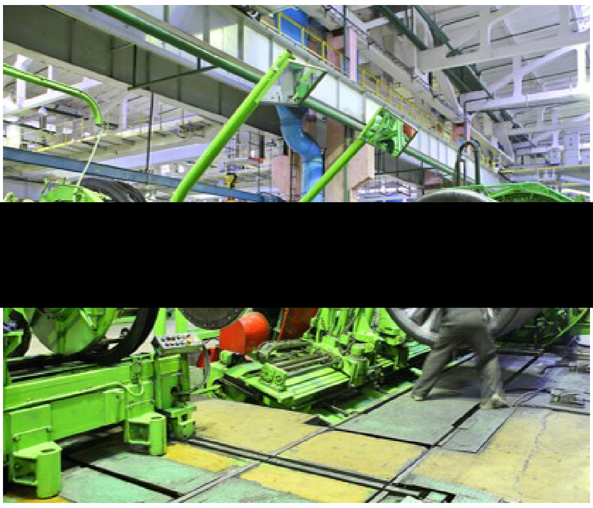
FIGURE 1.1: Structure of Chapter 1.

1.1 Motivation

For many major industries (such as manufacturing, aerospace, chemical, information technology, food and beverage, oil and gas, and energy) performing Predictive Maintenance is essential to prevent failures which may cause large-scale catastrophes (Figure 1.2). Flixborough Disaster (1974), Seveso Disaster (1976), Alexander Kielland Disaster (1980), Bhopal Gas Tragedy (1984), Sandoz Chemical Spill (1986), Piper Alpha Disaster (1988), Philips 66 Disaster (1989), Esso Longford Gas Explosion (1998), Texas City Refinery Explosion (2005), and the Macondo Blowout (2010) are some examples of engineering accidents for which there were deadly consequences [7–11]. According to [12], in the USA and Europe, among the major engineering accidents to have occurred 44% were due to maintenance related issues. In recent years, more attention has been paid to the topic of maintenance and efforts have been directed towards its systematisation in order to prevent catastrophic failures.

Economic implications caused by machinery failure can be severe. Manufacturing companies can experience substantial economic losses due to machinery failure and the resulting downtime. A notable example of the importance of maintenance involved the failure of the gear box in a conveyor steel manufacturing company; due to overheating, the conveyor remained out of service for two weeks with a resultant cost of $2.5 million for production restoration [13]. While maintenance in any type of organisations may require considerable financial investment, if strategically managed, it can contribute substantially to the competitiveness of the industry.

Traditionally, maintenance is divided into corrective and preventive maintenance. The corrective maintenance involves performing maintenance only when failure occurs [14–18]. This approach is expensive, and it is therefore only suitable for non-critical equipment with low cost, swift identification of failure, and fast repairs. Conversely, preventive maintenance involves routine maintenance activities scheduled on a regular basis designed to prevent failure from occurring; this reduces operational costs, but it typically involves an increase in maintenance costs [16–22]. In order to reduce maintenance costs, condition-based maintenance has been developed which resides in between the two types of maintenance strategy. The condition based maintenance is performed as and when required, depending upon the current condition of the equipment [23, 24]. In condition-based maintenance, the equipment is checked through regular inspection



(a) Mechanical

(b) Aerospace

[redacted]

(c) Power plant

(d) Energy

[redacted]

(e) Chemical

(f) Transportation

[redacted]

FIGURE 1.2: Predictive Maintenance can help to prevent several catastrophic failures in mechanical (a), aerospace (b), power plant (c), energy (d), chemical (e), transportation (f) industries.

and/or via sensor data; this information identifies failure timelines in order that maintenance can be scheduled. However, condition-based maintenance lacks predictive models that can be used to interpret different failure trends and help decide when maintenance should be scheduled [25, 26]. Figure 1.3 shows the most important types of maintenance techniques.

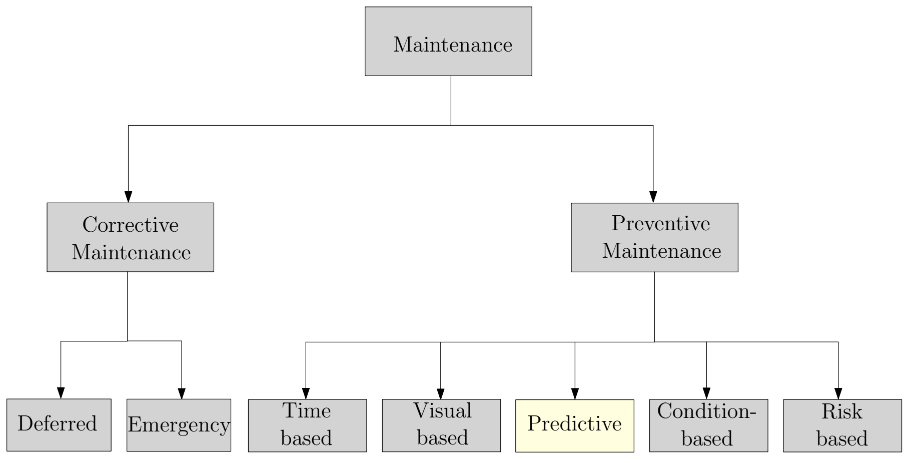


FIGURE 1.3: Types of maintenance techniques. Predictive Maintenance is the motivation of the present thesis.

Predictive Maintenance combines condition-based maintenance with a predictive model in order to predict when an item of equipment may fail. Specifically, Predictive Maintenance employs sensors to monitor equipment key parameters to continuously evaluate the overall health and predict changes in the physical condition of the equipment [27–30]. This strategy allows maintenance to be performed more efficiently, because more data is available concerning proximity of the equipment to the point of potential failure. With an appropriate predictive model-based on the availability of a sufficiently high-fidelity data, superior performance has been shown in the field of Predictive Maintenance. Predictive Maintenance is effectively an enhanced form of condition-based maintenance. Figure 1.4 depicts the operating and maintenance costs of different maintenance techniques. It can be seen that Predictive Maintenance lowers both maintenance and operating costs and optimises their combine cost, making it the preferred option.

Industry 4.0 revolutionises the maintenance domain by integrating Predictive Maintenance with production, logistics and services in industrial practices. Industry 4.0 recognises the existence of three previous industrial revolutions and suggests that its impact in transforming the world in which we live will be comparable to the prior revolutions [31]. Figure 1.5 shows the different types of maintenance techniques used in each industrial revolution. In the context of Predictive Maintenance, the fourth industrial revolution has substantially more data available through the use of technology such as Internet of

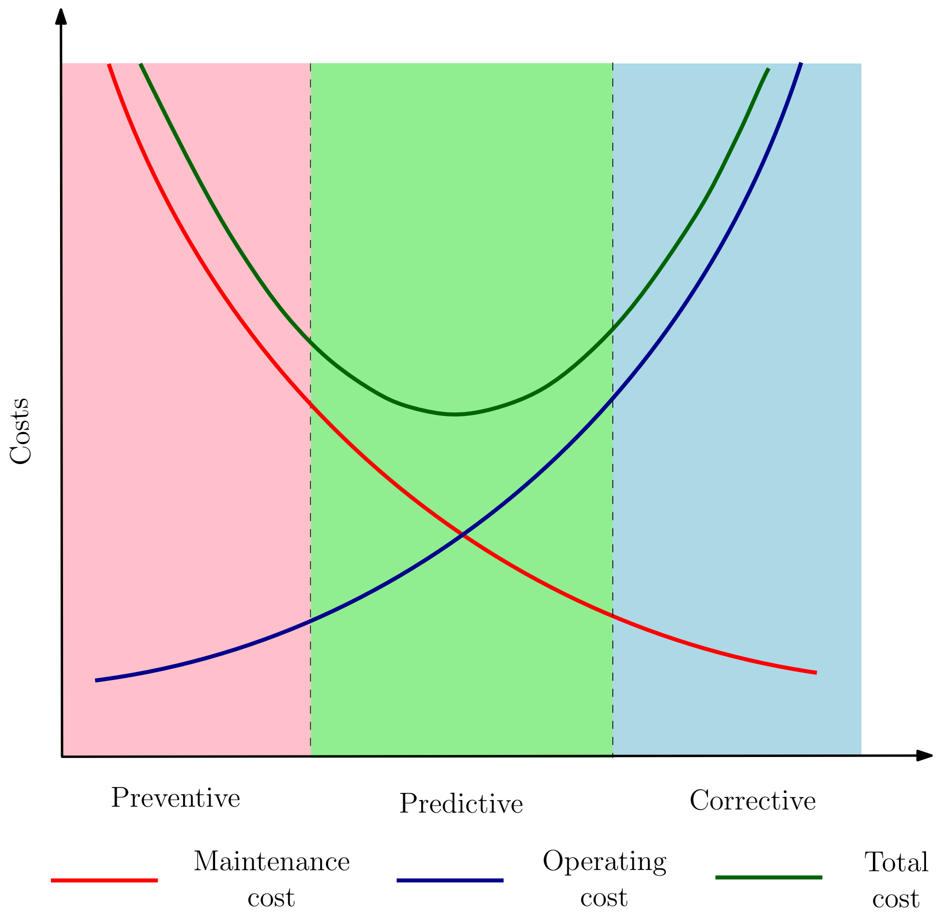


FIGURE 1.4: Operating and maintenance cost chart.

Things (IoT) sensors. These IoT sensors can identify maintenance issues in real-time, allowing perform cost effective maintenance by determining failure before it actually occurs. The use of Predictive Maintenance has gained in popularity in recent years due to the digitalisation in specific industrial sectors.

Smart and connected devices have a number of features that can reshape competitive advantage. Those are: (i) via the monitoring and reporting on themselves and their environment, new data and insights are created, (ii) with remote control through embedded or cloud software, users are presented with an unprecedented opportunity to tailor product function and personalise interactions, (iii) optimisation of product operation, capacity utilisation, and Predictive Maintenance can be enabled by analytics and algorithms, and (iv) autonomous operation, self coordination and self diagnosis is possible by access to monitoring data, remote control and optimisation algorithms [32]. Three of these capabilities (monitor, optimise and self diagnose) improve the development of Predictive Maintenance. The advances in Information Technology have reduced the complexity of gathering and analysing data, and this has improved the use of analytics for Predictive Maintenance [33]. In this way, collecting maintenance related data has become easier

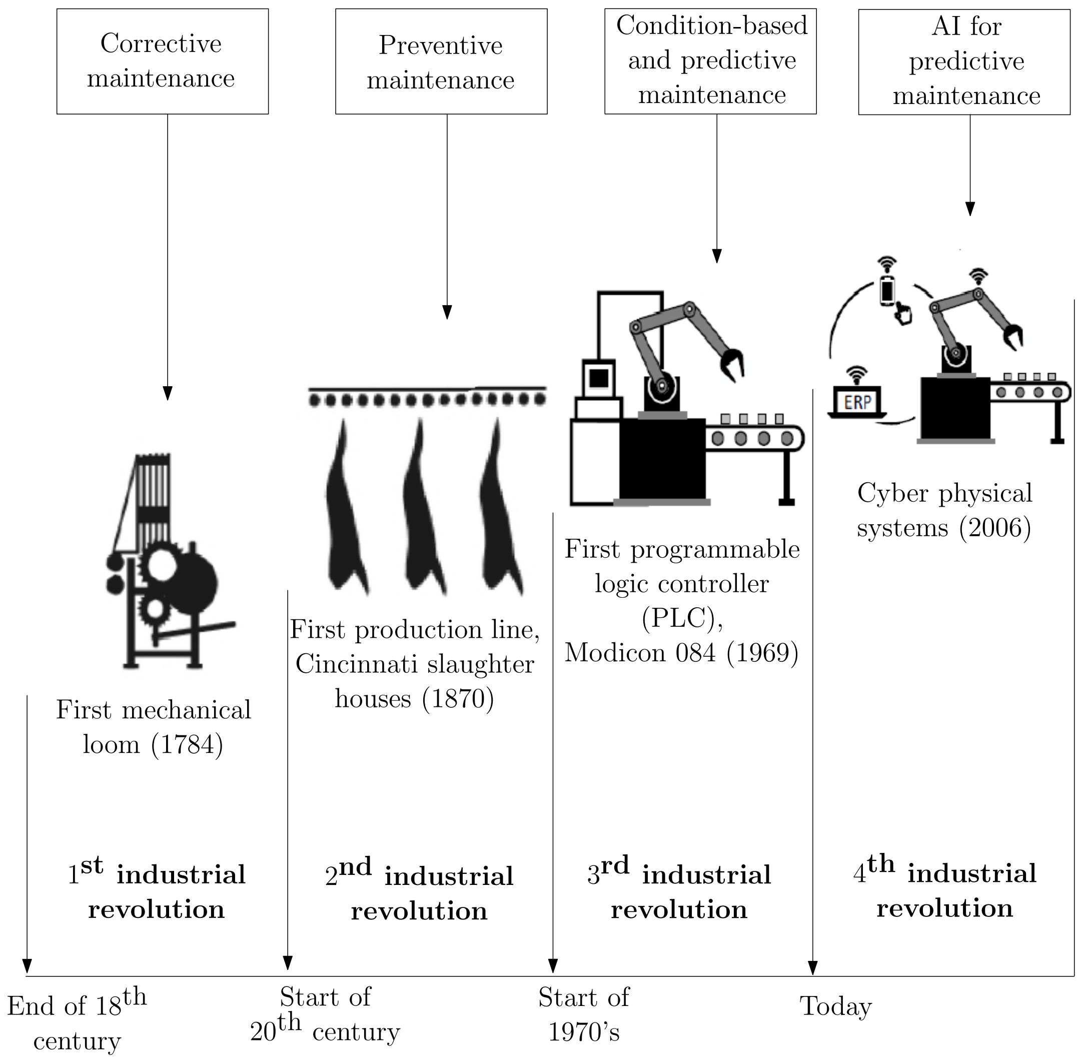

Corrective maintenance
Preventive maintenance
Condition-based and predictive maintenance
AI for predictive maintenance
ERP
First mechanical loom (1784)
First production line, Cincinnati slaughter houses (1870)
First programmable logic controller (PLC), Modicon 084 (1969)
Cyber physical systems (2006)
1st industrial revolution
2nd industrial revolution
3rd industrial revolution
4th industrial revolution
End of 18th century
Start of 20th century
Start of 1970's
Today

FIGURE 1.5: Industrial revolutions from first to fourth and respective developments. Partially adapted from [1].

through the use of sensors, micro-processors and computerised maintenance management systems. These collected datasets can be of different types, for instance, integer, boolean or strings. The traditional computational mechanics techniques often cannot handle the complexity of those datasets, for this reason, Machine Learning (ML) has emerged as a solution to this problem. As a result, ML is currently the fastest growing research field [34] and is being successfully used in the field of Predictive Maintenance in multiple major engineering industries [35].

The initial installation of condition monitoring systems in an organisation can incur high investment and operational costs. As a result, some industries can be sceptical about the investment required for initial installations. In such situations, data may be generated synthetically. In this way, it can be demonstrated to the industry that

Predictive Maintenance with ML application can be beneficial in the longer term. The layout of this chapter is depicted in Figure 1.1.

1.2 State of art

1.2.1 State of art in Predictive Maintenance

The sudden failure of industrial components can have adverse consequences for an organisation in terms of time, cost and workflow. It is, therefore, critical to maintain equipment components in their optimal condition to avoid downtime that may cause significant disruption. As modern technologies allow for increasingly complex equipment, maintenance regimes need to progressively adapt more advanced techniques. Predictive Maintenance has emerged as a highly useful tool to predict failure and minimise industrial impact. The use of Predictive Maintenance techniques helps to determine the condition of in-service equipment to predict when maintenance should be performed. Two fundamental techniques of Predictive Maintenance are *fault diagnosis* and *fault prognosis*.

- *Fault diagnosis* is related to the detection of faults that are emerging in an equipment component; these are usually predicted using data gathered regarding the condition of the equipment.
- *Fault prognosis* is related to the determination of the Remaining Useful Life (RUL); this is typically forecasted by utilising life history data or run-to-failure history data of the equipment [23].

In order to perform fault diagnosis and fault prognosis, it is essential to monitor the actual mechanical condition of the equipment's health and life history in regular intervals. In recent years, the requirement for Predictive Maintenance has been enhanced by several findings across multiple applications which demonstrate that diagnosis and prognosis can reduce the number and impact of failures, extend the time between maintenance intervals, increase operational performance, and reduce overall costs [23, 36–40]. The primary comparison of fault diagnosis and prognosis is given in Table 1.1. The fault diagnosis and fault prognosis methods are discussed broadly in the following sections.

	Fault diagnosis	Fault prognosis
Definition	Detection of fault	Prediction of RUL
Required information	Mechanical condition of the equipment	Mechanical condition, fault propagation process, failure mechanism of the equipment
Main categories	model-based, statistical, ML approach	model-based, statistical, ML approach
Model-based approach	Mechanistic knowledge and explicit mathematical model	Mechanistic knowledge and explicit mathematical model
Statistical approach	Null hypothesis problem, statistical process control, etc.	Weibull distributions, ARMA model, Markov model, etc.
ML approach	ML classification algorithms (SVM, MLP, Naive Bayes, decision trees, K-Nearest Neighbour (KNN), discriminant analysis etc.)	ML regression algorithms (SVM regression, MLP, logistic regression, KNN, decision tree, etc.)

Table 1.1: Comparison of fault diagnosis and fault prognosis.

1.2.1.1 Fault diagnosis

Fault diagnosis involves the interpretation of the current status of the equipment given sensor readings and process knowledge. Early fault detection or diagnosis can help avoid abnormal events progression and reduce productivity losses while the equipment is still operating in a controllable region [41]. Based on [23], fault diagnosis approaches can be grouped into three primary categories, namely *model-based approaches, statistical approaches* and *ML approaches*. All three techniques are briefly discussed as follows.

Model-based approaches require specific mechanistic knowledge and an explicit mathematical model of the monitored equipment [42, 43]. Based on this explicit mathematical model, Kalman filter, parameter estimation or system identification methods can be used to indicate whether a fault is present within the equipment. Model-based approaches can be more effective than other approaches if an accurate mathematical model exists. However, an accurate mathematical model is frequently unfeasible for complex equipment, as it would often be highly convoluted or potentially impossible to build mathematical models for intricate component interactions. Model-based approaches have been applied to a variety of mechanical devices such as bearings [44, 45], gearboxes [46] and rotating machinery [47, 48].

Statistical approaches are common methods for fault diagnosis that use available condition monitoring data. A fault diagnosis problem can be described as a null hypothesis test problem; this approach uses condition monitoring information to determine whether to accept or reject the null hypothesis. Examples of the use of hypothesis testing for fault diagnosis can be found in [49, 50]. Another conventional approach called statistical process control, which was developed in quality control theory, and has been widely used in fault diagnosis. The principle of statistical process control is to measure the deviation of the current dataset from a reference dataset representing the normal condition to see whether the current dataset is within the control limits or not [51].

ML approaches follow the concept of pattern recognition approach. Pattern recognition is a process of mapping the information obtained from the measurement space (or features space) to equipment faults in the fault space [52]. Traditionally, pattern recognition can be performed manually using representations such as power spectrum graphs, cepstrum graphs, spectrograms, wavelet scalograms and autoregressive spectrum graphs. However, manual pattern recognition requires the skills of a highly experienced person in the discipline of fault diagnosis. As a result, automatic pattern recognition is emerging; this introduces the classification of a given measurements based on the features or information extracted from them. Following this approach, ML and deep learning has emerged as a powerful tool for developing intelligent predictive algorithms for many applications. ML approaches have the ability to manage high dimensional and multivariate data and to extract hidden relationships in complex and dynamic environments [53]. In this way, ML provides powerful predictive approaches for Predictive Maintenance applications. However, the performance of these applications crucially depends on the appropriate choice of ML technique. ML methods, generally, involve a number of steps, including *historical data selection, data pre-processing, model selection, model training and validation* and *model maintenance* [54]. All of these steps are briefly described below.

- ☐ *Historical data selection step* identifies how and what types of data are to be collected and stored for the ML model. For Predictive Maintenance, this step is also called data acquisition step, and it aims to obtain exclusively data that is relevant to equipment health [23]. Several studies in fault diagnosis have been largely based upon the examination of vibration signals extracted from sensors distributed within the relevant equipment [55–57].

- ☐ *Data pre-processing step* processes and transforms data so that they can be efficiently processed within the ML model. This step includes data transformation, data cleaning (treatment of missing data and outlier removal in the case of experimental data) and feature extraction or feature selection methods (selecting the most valuable information).

Some of the feature extraction methods that are employed in performing fault diagnosis are below. Statistical measures, namely mean, variance, skewness, kurtosis, and fifth-to-ninth central moment, have been used as features for fault diagnosis for gears [58] and centrifugal pumps [59, 60]. Although the output from the statistical measures may be regarded as moderately acceptable, an enhanced solution would be advantageous. Applying advanced signal processing methods is a popular strategy for performing fault diagnosis in any type of machinery [56]. In applying such methods, spectral kurtosis and power spectral density are two important techniques for vibration signal analysis. Spectral kurtosis has proven that it is capable of highlighting local fault induced impulses in noisy backgrounds [61], and it can detect gear faults early by analysing its signal [62]. Power spectral density has proven effective in the detection of faults in hydraulic pumps [63]. Subsequently, wavelet has emerged as the most popular technique for non-stationary signals providing time and frequency information simultaneously [64, 65]. The continuous wavelet transform, with three distinct wavelet functions for fault diagnosis of a centrifugal pump, are presented in [66, 67] and Discrete Wavelet Transform (DWT) has been widely used for its superior gear fault detection features [68]. However, the DWT algorithm requires the sample size to be powers of 2, due to its down-sampling (re-sampling) step. To overcome this disadvantage (i.e. to remove the down-sampling step), maximal overlap discrete wavelet transform (MODWT) has been developed. Both DWT and MODWT have excellent frequency resolution in the low frequencies, but poor frequency resolution in the high frequencies. As an enhancement to the MODWT, maximal overlap discrete wavelet packet transform (MODWPT) has emerged, providing a better frequency resolution [69, 70]. Fault diagnosis has been performed using MODWPT for gears [69] and bearings [71] and it is achieved optimal results.

- ☐ *Model selection step* involves the selection of the appropriate ML model.

  A number of ML approaches in relation to fault diagnosis exist. Cluster analysis is a statistical classification approach, that groups signals into different fault categories on the basis of the similarity of the characteristics or features they possess. A fault diagnosis of mechanical equipment with cluster analysis application is discussed in [72, 73]. The ML algorithms commonly used in fault diagnosis are the nearest neighbour algorithm [74], Support Vector Machine (SVM) [66, 75, 76] and probabilistic based algorithms such as Bayesian network [58] and Naive Bayes (NB) (simplest form of Bayesian network derived from Bayes theorem) [77, 78].

In the literature, popular ML techniques for fault diagnosis are Artificial Neural Networks (ANN), fuzzy–neural networks, neural–fuzzy systems, evolutionary algorithms and deep learning algorithms [79]. The feedforward neural network is most widely used for machine fault diagnosis [80–83]. An important feedforward neural network called Multilayer Perceptrons (MLP) with backpropagation is the most commonly used neural network model for classification and pattern recognition and is also extensively employed for fault diagnosis [66, 81, 84–88]. Further neural network models, which have shown improved performance over a period of years, are the recurrent neural network [89–94], the convolutional neural network [95–98] and other miscellaneous deep learning neural networks [99–101] for various applications in several industrial sectors.

☐ *Model training and model validation step* involves ML model training and ML model validation and testing. This step is used to perform numerical examples related to fault diagnosis.

☐ *Model maintenance step* referred as the maintenance decision-making step, and it aims to decide the optimal algorithm for the Predictive Maintenance application based on the previous steps.

1.2.1.2 Fault prognosis

In equipment prognosis, there are two main prediction types. The most commonly used type is the prediction of time remaining before a failure occurs, given the current condition of the equipment and its past performance profile. The time left before observing a failure is designated Remaining Useful Life (RUL). In some situations, especially when a failure is potentially catastrophic, it is more than desirable to predict the probability that an equipment operates without fault or failure up to some future point in time (the next inspection interval), given the current equipment condition and its past operational profile. In any situation, the probability that equipment operates without a fault until the next inspection interval is a good reference point for a maintenance engineer to determine the appropriateness of the inspection interval [102, 103]. However, this is not a popular method due to the limitation in gathering the required data for prediction. The particular focus of this thesis is on RUL estimation.

RUL is also called remaining service life, which alludes to the time remaining before observing a failure given the current age and condition of the equipment. The precise definition of failure is essential to the correct interpretation of RUL [23]. To perform prognosis, in addition to the condition monitoring data of equipment, obtaining the fault propagation process and information on the failure mechanism is necessary. However, gaining knowledge of physical processes is not always possible [104]. The fault propagation process is usually tracked by a trending or forecasting model for certain condition variables. There are two methods to describe the failure mechanism. The first one assumes that failure depends only on the condition variables, which reflect the actual fault level, and the predetermined boundary condition (i.e. the pre-established level at which failure will occur). The second method constructs a model for the failure mechanism using historically available data. In this case, different definitions of failure can be used. A failure can be defined as an event where the equipment operates at an unsatisfactory level, or it can be a malfunction in which the equipment is unable to perform as intended, or it can simply reflect a breakdown when the equipment ceases to operate. Similar to diagnosis, the approaches to prognosis fall into three primary categories, namely *model-based approaches, statistical approaches* and *ML approaches.*

Model-based approaches for prognosis requires having a mechanistic knowledge and a theory relevant to the monitored equipment. For example, defect propagation models by means of mechanistic modelling for RUL estimation of bearings are introduced in [105, 106]. Reference [107] presented a physical model for predicting the condition of the equipment in combination with the strength of faults to life model-based on crack growth law to estimate RUL. An alternative means of applying model-based approaches to prognosis is to derive the explicit relationship between the condition variables and the lifetimes through mechanistic modelling. Following this avenue of research, RUL estimation has been performed for an energy processor and a bearing, based on monitored vibration signals [108, 109].

Similar to diagnosis, *statistical approaches* have been widely employed in prognosis. Based on Weibull distributions, [110] proposed a method to predict the potential failures for both installation and operation. Reference [111] proposed a logistic regression model to calculate the probability of failure for given condition variables and an autoregressive moving average time series model to trend the condition variables for failure prediction, [112] used Kalman filtering. The proportional hazards and proportional intensity models are two useful tools for RUL estimation, which can both be used in combination with a trending model for the fault propagation process. RUL estimation in combination with both of these models are discussed with Markov modelling in [113] and hidden Markov modelling in [114].

In the present thesis, the focus is principally based on *ML approaches.* Processes designed using the ML approach for prognosis are similar to those of diagnosis, which are presented in Section 1.2.1.1. ML approaches are widely documented performing fault prognosis in literature. Methodologies employed to predict the RUL of rolling bearings using the gated recurrent unit method and the SVM regression method are discussed in [115] and [116], respectively. SVM regression is extensively used for prognosis within a variety of applications [117–119]. MLP with backpropagation is another method widely applied in the field of prognosis. Reference [120] proposed the application of an echo state network and a recurrent multilayer perceptron. Reference [121] took the approach of comparing the results of applying recurrent neural networks and neural–fuzzy inference systems to predict failure of the machinery, whereas the proposal within Reference [122] was based on the Bayesian regularised radial basis function neural network for an external gear pump. Recent developments in prognosis have also used the deep neural network, recurrent neural network or convolutional neural network [104, 115, 123–127], and have been applied within various industrial sectors.

Predictive Maintenance market

The Predictive Maintenance market has been growing exponentially. The global market for Predictive Maintenance, which was estimated at $1.5 Billion in 2016 and $3.3 Billions in 2018 is projected to reach a revised size of $23.5 Billions by 2024; this represents a compound annual growth rate of 39% over the period 2018-2024 [128]. Figure 1.6 shows

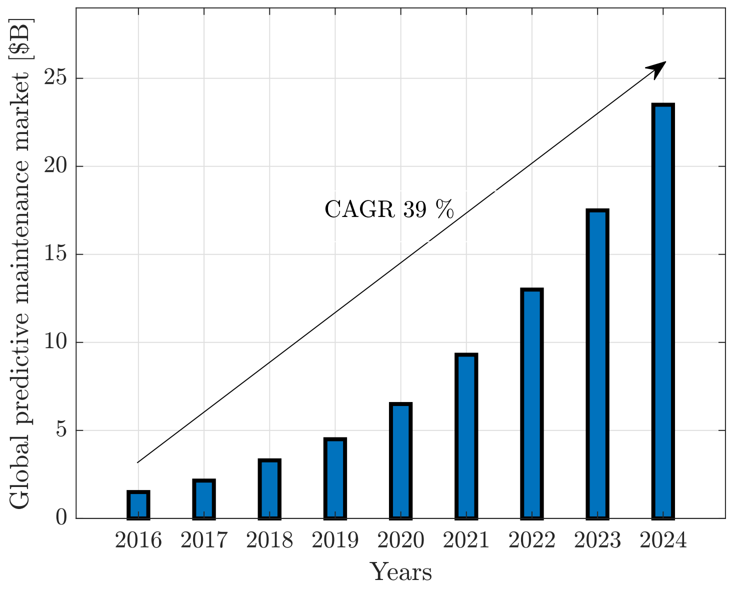


FIGURE 1.6: Global Predictive Maintenance market from 2016 to 2024. CAGR is compound annual growth rate.

the prediction of Predictive Maintenance market growth. The expansion of the market can be attributed by the rapidly increasing field on Big data and emerging concepts such as the IoT.

1.2.2 State of the art in external gear pumps

A pump is responsible for the mechanical to hydraulic energy conversion of a working fluid and is a key component of hydraulic systems. Pumps can be mainly classified into centrifugal pumps and positive displacement pumps. A centrifugal pump consists of one or more rotating impellers that guide the liquid to and from the impeller, increasing the pressure and flow rate of the fluid as it rotates. Positive displacement pumps move a fixed amount of fluid at regular intervals. They are built with internal cavities that fill up at the suction (inlet) side, to be discharged with higher pressure at the outlet. Based on how the fluid is displaced, positive displacement pumps can be categorised into linear, reciprocating and rotary, and are shown in Figure 1.7.

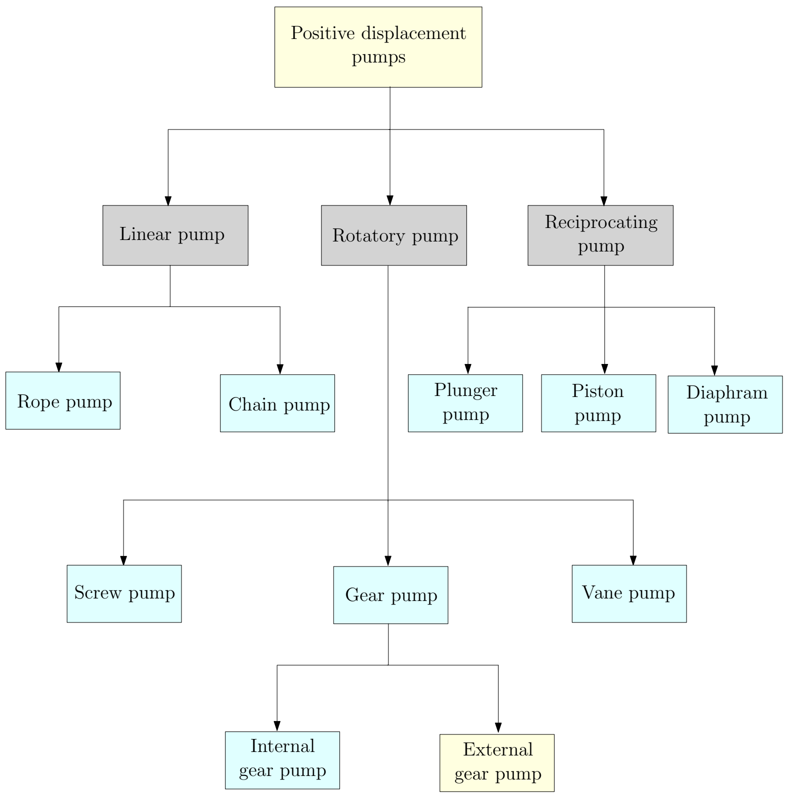


FIGURE 1.7: Various types of positive displacement pumps; An external gear pump is the focus of the present thesis.

Among the positive displacement pumps, the rotatory gear pumps are commonly used within the industry. Gear pumps use a simple mechanism to generate flow and therefore have an associated minimum number of design parts. The simplicity of the gear pump design translates into higher reliability as compared to other positive displacement pumps that use a more complex design. Gear pumps can be operated at high speeds and used in applications where the operating pressure is low to moderate. They are widely used in manufacturing industries due to the advantages of possessing compact designs and minimal manufacturing costs. Gear pumps are categorised into internal gear pumps and external gear pumps. The pumping action of both gear pumps are similar, i.e., two gears come in and out of mesh to produce flow. The present thesis focuses on an external gear pump. Figure 1.8 shows the principle operation of the external gear pump. In Figure 1.8, the fluid is displaced from the inlet side of the pump to the outlet side with the assistance of two identical rotating gears. Among the two identical gears, one gear is driven by a motor (drive gear) and it, in turn, drives the other gear (slave gear). Each gear is supported by a shaft with bearings on both sides of the gear. External gear pumps have several advantages: (i) the fluids can be moved at high speeds and under high pressure, (ii) the operation volume is quiet that avoids noise pollution and (iii) a variety of fluids can be accommodated. However, the disadvantages of the pumps are: (i) the close tolerances between the gears and casing mean that these types of pumps are susceptible to wear particularly when used with abrasive fluids and (ii) the solid part is not allowed in the liquid area because the close tolerances between the gears and casing, a tight gear mesh, and limited tooth volumes make pumping fluids with large and suspended solids difficult.

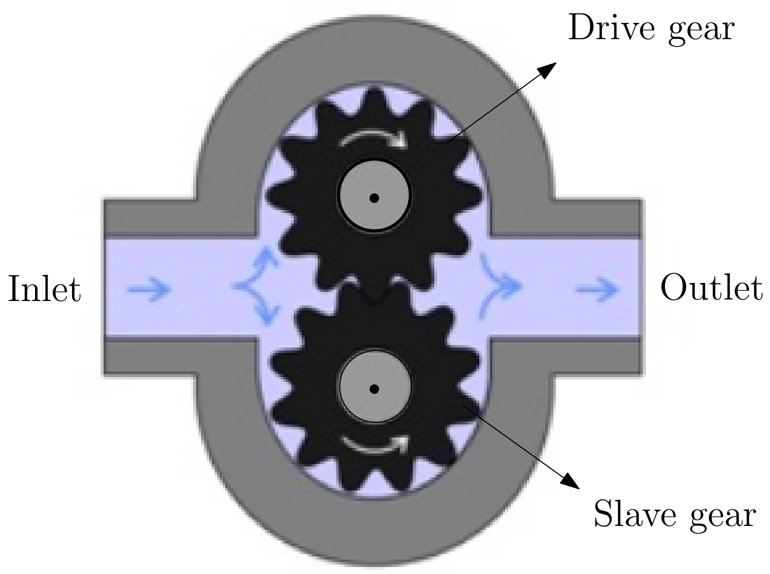


FIGURE 1.8: Principle operation and geometry of an external gear pump. The fluid displaced from inlet side of the pump to outlet side with the help of rotating gears.

An external gear pump is notably simple in its operation. Depending on the pump geometry involved, the complexity of the subject arises, which has typically required numerical analysis to solve the governing equations. An analytical method of understanding a gear pump was first proposed by [129], and the average flow rate of a gear pump, using such a method, has been conducted by [130]. Due to the complexity of geometry, the analysis was performed through computational methods including Computational Fluid Dynamic (CFD) modelling. Several studies using CFD modelling have been performed over the years, including [131] which studied the flow reversing process inside a gear pump, and [132] which modelled experimental validation of a gear pump. Reference [133] explored the effect of cavitation on the volumetric efficiency of an external gear pump in 2D, which later evolved into a computational 3D simulation. Reference [134, 135] investigated the effect of turbulence flow in the external gear pump. Many commercial software, such as StarCCM+, Ansys fluent, openForm, PumpLinx from Simerics, are available to overcome the difficulty of coding complex geometry and computing the intricate 3D simulations. PumpLinx has been developed to specifically address the problems emerging from positive displacement devices [136]. It possesses a particular strength in its explicit accuracy in the analysis and testing of the performance of positive displacement pumps. In addition, for fluid systems, PumpLinx's cavitation module efficiently models the vapour, the free gas and the liquid's compressibility. PumpLinx's uses binary tree meshing that provides tight and leak free surfaces. Reference [137] performed external gear pump analysis using PumpLinx's cavitation module and its accuracy was demonstrated when compared to the experimental results. In addition, PumpLinx has been tested against Ansys CFX and its superior precision in relation to the experimental results was proven on the same mesh [138]. Due to its effective performance, PumpLinx has been chosen to conduct all of the computational simulations in the present thesis. The details of the CFD simulations are discussed in Chapters 2 and 3.

1.3 Objectives

The overall aim of this project is to demonstrate the use of ML in the context of Predictive Maintenance within an industrial framework. This Ph.D. project is conducted in collaboration with an industrial partner, F-Lab, an innovation centre in Fluid-o-tech. Fluid-o-tech produces various types of pumps for several important applications, in which the FG304 series External Gear Pump, also known as Domino pump, (shown in Figure 1.9), has been used within medical equipment, ink-jet printing systems, cooling systems, water treatment felicities, food processing equipment and sanitisation applications [2]. The operating condition of the external gear pumps play a pivotal role in the performance of equipment in which they are installed. Therefore, in order to anticipate the early failure of the equipment in which they reside, Predictive Maintenance of the gear pumps is essential.

[redacted]

FIGURE 1.9: FG200 - FG400 series motor unit. Image reprinted from [2].

The main objectives of this research are as follows:

- ☐ Exploration of Predictive Maintenance strategies and their application to a given gear pump.
- ☐ Development of synthetic data generation methods for Predictive Maintenance in the absence of experimental data
    - Development of detailed CFD models, representative of the given gear pump in various working conditions.
    - Validation of the CFD model against the experimental results.
    - Generation a high-fidelity dataset using the CFD model to recreate a variety of working conditions for the given gear pump.
    - Recreation of possible environmental variations of the working pump conditions
    - Recreation of possible degradation behaviour variations of the pump conditions
- ☐ Comparison of feature extraction methods to identify the optimal feature extraction method.
- ☐ Optimisation of hyper-parameters for ML algorithms

- ☐ Identification of suitable ML algorithms to perform fault diagnosis in the gear pump.
- ☐ Identification of the appropriate ML algorithms to predict the Remaining Useful Life (RUL) of the gear pump.

Each of these objectives is considered and discussed in the subsequent chapters of this thesis.

1.4 Outline of the thesis

This thesis comprises 8 chapters and is supported by 2 appendices, which are as follows:

- ☐ **Chapter 2**: *High-fidelity Computational Fluid Dynamics.*
  The present chapter introduces the computational modelling of an external gear pump. It progresses from introducing the geometry of interest to the mathematical formulations of the CFD governing equations, turbulence model and cavitation model, and then moves on to the computational framework. This is followed by a discussion of the computational implementation, model attributes of the CFD simulation and the implemented fault scenarios.
- ☐ **Chapter 3**: *High-fidelity computational results.*
  The present chapter provides the numerical examples of the CFD simulation for an external gear pump. It begins by presenting the geometry and the computational implementation including the relevant mesh generation using PumpLinx. The mesh sensitivity analysis, the experimental validation of the CFD model, and the gear pump running in healthy and faulty conditions are presented next.
- ☐ **Chapter 4**: *Machine Learning for diagnosis and prognosis.*
  The present chapter presents an overview of Machine Learning, general terminology and its types. In addition, ML algorithms, namely MLP, SVM and NB, chosen to perform fault diagnosis and fault prognosis are presented. For each ML algorithm, their operational mechanism is briefly explained together with the optimisation process of the hyper-parameters.
- ☐ **Chapter 5**: *Data collection and preprocessing.*
  The present chapter focuses on data collection and pre-processing methods for the

ML analyses. It begins by introducing the various methods of data collection and then provides data manufacturing methods to increase the availability of data for ML training. The importance of standardisation in ML and the feature extraction techniques to distil descriptive features from real-time series sample data are presented.

☐ **Chapter 6**: *Numerical examples for diagnosis.*
The present chapter provides the numerical examples of fault diagnosis with the use of the ML classification algorithms (MLP, SVM and NB). It begins by presenting the framework of the proposed fault diagnosis strategy, optimisation of hyper-parameters and progresses to various numerical examples.

☐ **Chapter 7**: *Numerical examples for prognosis.*
The present chapter presents the numerical examples of fault prognosis with ML algorithms, MLP and SVM. The proposed framework of predicting remaining useful life, optimisation of hyper-parameters and numerical examples for different faulty scenarios are presented.

☐ **Chapter 8**: *Conclusion and Remarks.*
The present chapter comprises a summary of the work performed as part of this thesis, together with an associated discussion. Potential directions for future research are also presented.

☐ **Appendix A**: *PumpLinx*
The present appendix presents the simulation workflow of PumpLinx. In particular, pre-processing steps, simulation modelling, and post processing steps are discussed. Some accompanying mathematical formulas and definitions of physical variables are also detailed.

☐ **Appendix B**: *Frequency to wavelet analysis.*
The present appendix discusses a frequency to wavelet analysis, i.e., from Fourier transform to wavelet transform, and further available wavelet transforms. Firstly, wavelet transform and its definition are introduced for the continuous wavelet transform and DWT, and then the evolution of the wavelet analysis from DWT to MODWPT is presented. Finally, their advantages in terms of realising better resolutions in the high frequencies of the signal are considered.

☐ **Appendix C**: *Code description.*
The present appendix presents the code description of synthetic data generation methods and ML are presented. Firstly, the MATLAB code for noise perturbation method to increase the amount of data available for ML training is presented.

Next, to obtain degradation behaviour in the data for fault prognosis, the linear interpolation and the cubic interpolation methods are discussed. Finally, MATLAB and Python code for ML algorithms using specific libraries are presented for SVM, MLP and NB.

1.5 General remarks

Some general remarks applicable to the entirety of this thesis are described below:

- ☐ All quantities mentioned in this work are expressed in *SI units* unless otherwise stated.
- ☐ *Vectors* and *tensors* are represented with **bold** fonts.
- ☐ All CFD simulations and post-processing of CFD simulations are performed using the 3D fluid dynamic software *PumpLinx*.
- ☐ The ML toolbox in *MATLAB* software is employed for the SVM analysis of fault diagnosis.
- ☐ *Python* open source programming software is used for all MLP, SVM and NB analyses. Python libraries such as *TensorFlow* and *Keras* are employed for MLP algorithm and *Scikit-learn* for SVM regression and NB classification.
- ☐ Post-processing of results and generation of the plots have been completed using *MATLAB*.

Chapter 2

HIGH-FIDELITY COMPUTATIONAL FLUID DYNAMICS

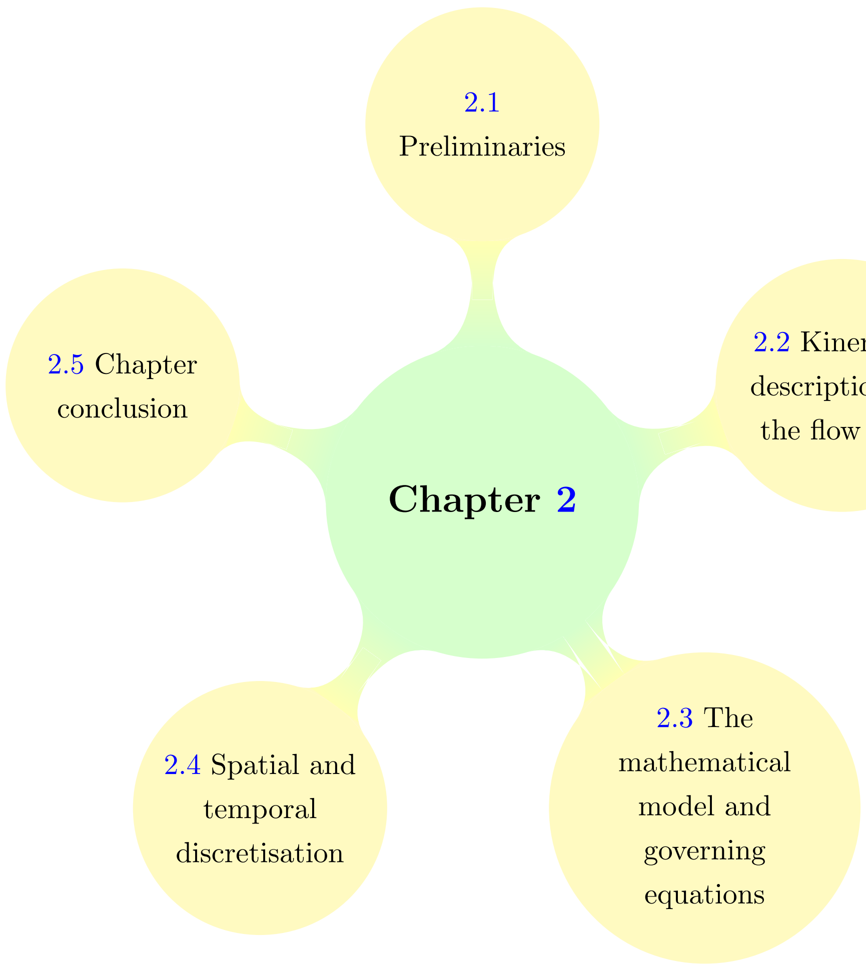

2.1 Preliminaries
2.5 Chapter conclusion
2.2 Kinematic descriptions of the flow field
Chapter 2
2.4 Spatial and temporal discretisation
2.3 The mathematical model and governing equations

FIGURE 2.1: Structure of Chapter 2.

2.1 Preliminaries

The present chapter presents the mathematical equations of high-fidelity Computational Fluid Dynamics (CFD). The kinematic description of the flow field and the governing equations along with the turbulence and cavitation models are presented in Sections 2.2 and 2.3, respectively. The spatial and temporal discretisation of the governing equations using the Finite Volume Method (FVM) and implicit time integration approach are presented in Section 2.4. The structural layout of the present chapter is shown in Figure 2.1.

2.2 Kinematic descriptions of the flow field

An appropriate kinematic description is essential to simulate fluid flow by any numerical method. The choice of description determines the relationship between the deforming continuum and the computational mesh. Three different types of motion description used in the continuum mechanics are the Lagrangian description, the Eulerian description and the Arbitrary Lagrangian-Eulerian (ALE) description [3]. These descriptions are shown in Figure 2.2. The Eulerian description is most widely used in fluid mechanics, and the Lagrangian description widely used in solid mechanics [139].

2.2.1 Eulerian and Lagrangian description

In an Eulerian description, the computational mesh is fixed throughout the numerical simulation where the fluid moves with respect to the computational grid. The Eulerian description allows severe distortion in the fluid motion which is crucial for simulating turbulent flows. However, the downside is the difficulty to track free surfaces and interfaces between different media. A compromise has to be made to add approximate tracking techniques such as the level set method and the volume of fluid method [3].

In case of a Lagrangian description, the computational mesh follows the associated material particle during the deformation. The Lagrangian description allows easy tracking of free surfaces and interfaces between different materials; however, it is not practical

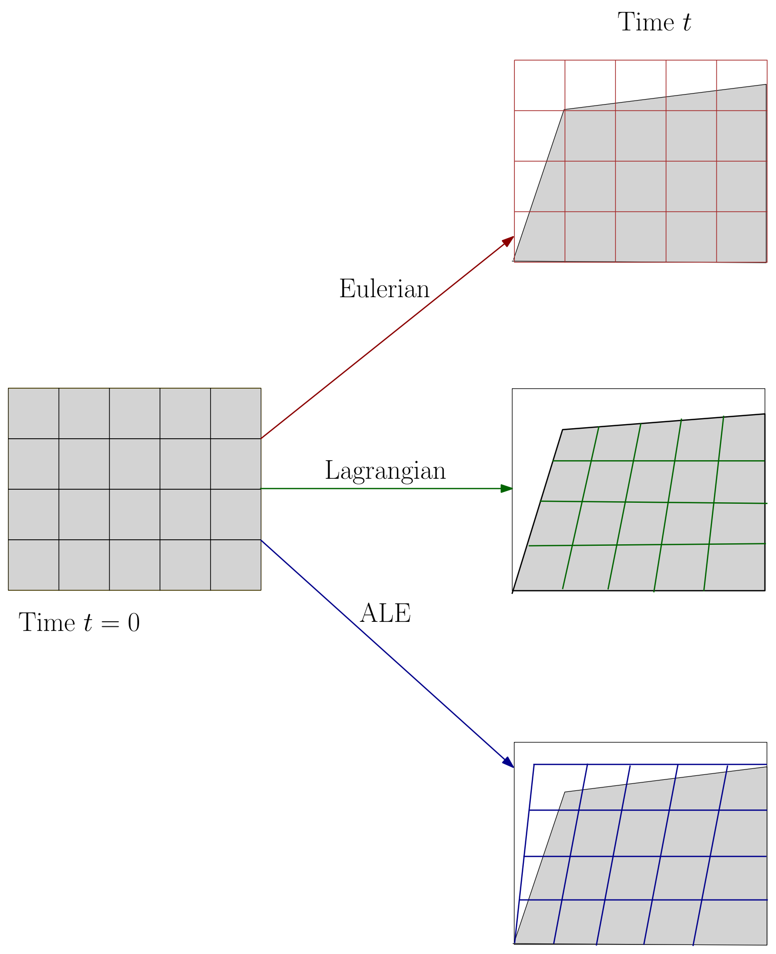


FIGURE 2.2: A Eulerian (computational mesh does not move), a Lagrangian (computational mesh follows the movement of deformed continuum) and a ALE (computational mesh moves but does not follow the motion of the deformed continuum) descriptions of motion.

to follow large distortions of a computational domain without frequent re-meshing operations. The primary application of the Lagrangian description is large deformation problems like modelling of metal forming operations and vehicle crash simulation, where they are used in combination with solid and structural elements [3].

2.2.2 Arbitrary Lagrangian-Eulerian (ALE)

By combining the advantages of both Lagrangian and Eulerian frameworks, the ALE algorithm was introduced. ALE algorithms are particularly useful in flow problems involving large distortions in the presence of deforming boundaries. The key advantage of the ALE formulation is a computational mesh that moves with a velocity that is independent of the material particle velocity [3, 140]. There are two types of ALE approaches, firstly, direct ALE, where the advective terms are explicitly included in the governing equations. On the other hand, the indirect ALE method comprises the

Lagrangian stage, where the numerical solution and the computational mesh are updated, the rezoning stage, where the nodes of the mesh are moved in order to improve grid quality and the remapping stage, where the Lagrangian solution is transferred to the rezoned mesh [141]. In the present study, the direct ALE method has been used.

In the ALE description of motion, neither the material nor spatial configuration, is taken as reference. Another domain called referential configuration is taken as reference, where the referential coordinates $\boldsymbol{\chi}$ is introduced to identify the grid points. Figure 2.3 shows material, spatial, and reference configurations along with their one-to-one transformations. The referential domain is mapped into material and spatial domains. The particle motion $\boldsymbol{\varphi}$ can be expressed as $\boldsymbol{\varphi} = \boldsymbol{\Phi} \circ \Psi^{-1}$ and this shows that these three mappings are not independent. The mapping $\boldsymbol{\Phi}$ from referential domain to the spatial

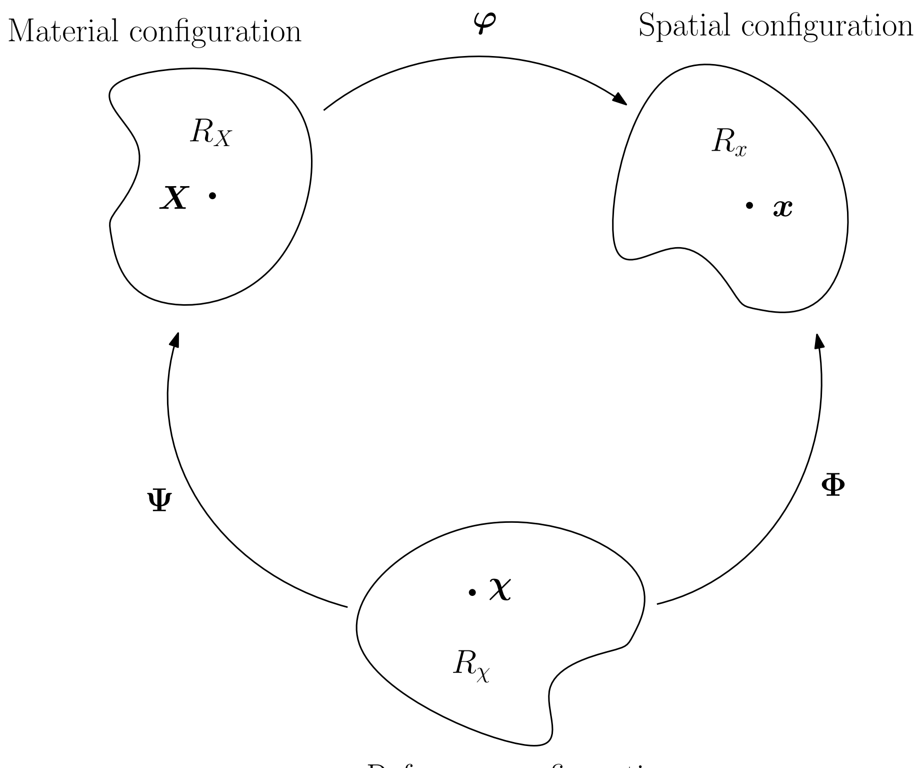


FIGURE 2.3: ALE description of motion. Material, spacial ans referential configurations. The current figure is adapted from [3].

domain, which is the motion of grid points in the spatial domain, is represented by

$$\boldsymbol{\Phi} : R_\chi \times [t_0, t_{\text{final}}] \rightarrow R_x \times [t_0, t_{\text{final}}] \tag{2.1}$$

$$(\boldsymbol{\chi}, t) \mapsto \boldsymbol{\Phi}(\boldsymbol{\chi}, t) = (\boldsymbol{x}, t). \tag{2.2}$$

The mesh velocity $\boldsymbol{v}_s$ is defined as

$$\boldsymbol{v}_s = \frac{\partial \boldsymbol{\Phi}}{\partial t} \mid_{\boldsymbol{\chi}}, \tag{2.3}$$

where $\boldsymbol{\Phi}$ is the spatial configuration coordinate as shown in Figure 2.3.

2.3 The mathematical model and governing equations

The start of the modelling strategy is a mathematical model, in this case given by a the set of partial differential equations (or integral differential equations), initial conditions and boundary conditions. An appropriate model (i.e. compressible or incompressible fluid flow, inviscid or viscous, laminar or turbulent, two dimensional or three dimensional, etc.) for the target application must be chosen. Consider a time-varying control volume $\Omega(t)$ of boundary $\partial\Omega(t)$ and outward unit normal $\boldsymbol{n}$. The conservation of mass principle can be written using an ALE description as follows [136]

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \rho d\Omega + \int_{\partial\Omega(t)} \rho(\boldsymbol{v} - \boldsymbol{v}_s) \cdot \boldsymbol{n} ds = 0, \tag{2.4}$$

where $\boldsymbol{v}$ is the fluid velocity and ρ is the fluid density. The conservation of linear momentum written in ALE description is given as,

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \rho\boldsymbol{v} d\Omega + \int_{\partial\Omega(t)} [\rho\boldsymbol{v} \otimes (\boldsymbol{v} - \boldsymbol{v}_s)] \cdot \boldsymbol{n} ds = \int_{\partial\Omega(t)} \boldsymbol{\tau}\boldsymbol{n} ds - \int_{\partial\Omega(t)} p\boldsymbol{n} ds + \int_{\Omega(t)} \rho\boldsymbol{f} d\Omega, \tag{2.5}$$

where $\boldsymbol{\tau}$ is the viscous shear stress tensor, p is the pressure, and $\boldsymbol{f}$ is a body force. The conservation of energy is written in ALE description as,

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \rho E d\Omega + \int_{\partial\Omega(t)} \rho E(\boldsymbol{v} - \boldsymbol{v}_s) \cdot \boldsymbol{n} ds = \int_{\partial\Omega(t)} k(\boldsymbol{\nabla} T \cdot \boldsymbol{n}) ds + \int_{\Omega(t)} Q_v d\Omega + \int_{\partial\Omega(t)} \boldsymbol{Q_s} \cdot \boldsymbol{n} ds, \tag{2.6}$$

where E is total energy per unit of mass, k is the thermal conductivity, $Q_v = \rho\boldsymbol{f} \cdot \boldsymbol{v} + r$ is a heat source term, $\boldsymbol{Q_s} = -p\boldsymbol{v} + \boldsymbol{\tau}\boldsymbol{v}$ and r is the rate of volumetric heat generation.

The control volume boundary $\partial\Omega(t)$ can be decomposed into non-overlapping regions $\partial\Omega_i(t)$, $\partial\Omega_o(t)$ and $\partial\Omega_w(t)$ representing inlet, outlet and wall, respectively, such that

$$\partial\Omega_i(t) \cup \partial\Omega_o(t) \cup \partial\Omega_w(t) = \partial\Omega(t), \tag{2.7}$$

and

$$\partial\Omega_i(t) \cap \partial\Omega_o(t) = \emptyset, \tag{2.8}$$

$$\partial\Omega_i(t) \cap \partial\Omega_w(t) = \emptyset, \tag{2.9}$$

$$\partial\Omega_o(t) \cap \partial\Omega_w(t) = \emptyset. \tag{2.10}$$

In order to solve all the unknowns in the conservation equations, viscous law, equation of state, and caloric equations need to be additionally considered. The shear stress tensor $\boldsymbol{\tau}$ is a function of the fluid viscosity μ and the velocity gradient and it is defined as

$$\boldsymbol{\tau} = 2\mu\boldsymbol{d} - \frac{2}{3}(\mathrm{tr}\boldsymbol{d})\boldsymbol{I}; \quad \boldsymbol{d} = \frac{1}{2}(\nabla\boldsymbol{v} + (\nabla\boldsymbol{v})^T). \tag{2.11}$$

For the closure of the system, an equation of state is essential to link the pressure to the thermodynamic variables; this requires the conservation of energy equation (Equation (2.6)) and the behaviour of the cavitating mixture to be considered [142, 143]. For compressible fluids, considering T as the absolute temperature, it is postulated that p, ρ and T are related in a definite way. The easiest way to relate these variables is through Boyle's law, given by

$$p = \rho R T, \tag{2.12}$$

where R is a constant for the fluid under consideration. A modified equation of state (Van der Waals) is also possible and given by

$$p = \frac{RT}{\rho - b} - a\rho^2, \tag{2.13}$$

where, b is the volume occupied by one mole of the molecules and a is a constant that depends on the gas. The constants a and b are derived from experimental data on the density, temperature and pressure interdependence. Consequently, it follows that E can be related to ρ and T in a general way, that is

$$E = E(\rho, T), \tag{2.14}$$

this type of equation is called a caloric equation of state. The actual form of the function depends on the fluid properties. The simplest form of the caloric equation is

$$E = c_v T, \tag{2.15}$$

where c_v called the *specific heat at constant volume*, is either a constant or a function of ρ. The above equations from Equations (2.4)-(2.15) define the complete form of compressible flow with thermal effects. However, the CFD simulation model considered in the present thesis does not have a large variation in the temperature field, so the model is considered isothermal. Therefore, along with Equations (2.4) and (2.5), suitable turbulence and cavitation models are included in the CFD simulations. The turbulence and cavitation models will be described in the following sections.

2.3.1 The turbulence model

In the turbulence regime, fluid motion is characterised by chaotic changes in pressure and flow velocity in both space and time. It contrasts with a laminar regime, which occurs when a fluid flows in parallel layers with no disruption between the layers. The turbulence regime is extremely frequent in natural phenomena and real life applications: the aerodynamics of vehicles, most of the terrestrial atmospheric recirculation and many industrial applications [144].

Reynolds Number

The dimensionless Reynolds number is defined as

$$Re = \frac{\rho v_\infty D}{\mu}, \tag{2.16}$$

where D is a characteristic length (or pipe diameter), μ is the dynamic viscosity and v_∞ is the free stream velocity. Turbulent flow occurs when Re exceeds a certain threshold value depending on the application's topology and the flow physics [144]. Figure 2.4 shows the difference between streamlines in turbulent and laminar flows. The straight, parallel lines are streamlines, which are everywhere parallel to the mean flow. In laminar flow, the fluid particles follow the streamlines, and in turbulent flow, eddies of many sizes are superimposed onto the mean flow.

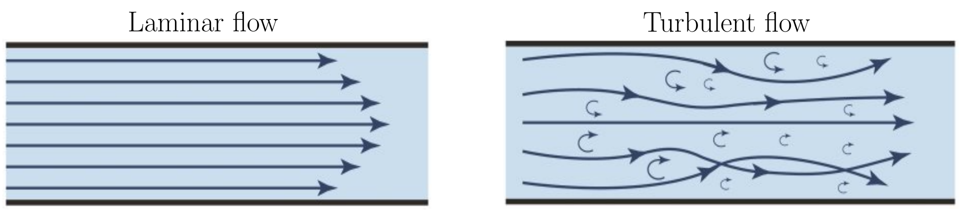


FIGURE 2.4: In laminar flow, the fluid particles follow the streamlines, and in turbulent flow, eddies of many sizes are superimposed onto the mean flow.

Turbulence modelling in Reynolds Average Navier Stokes equations

To obtain the Reynolds Average Navier Stokes (RANS) equations, unsteady Equations (2.4) - (2.5) are averaged in time to smooth the instantaneous turbulent fluctuations in the flow field, while still allowing the capture of the time-dependency in the large time scales of interest. In many engineering problems, this assumption is valid, but this averaging procedure breaks down if the time scale of the physical phenomena of relevance is similar to that of the turbulence itself. For compressible flows, the density weighted *Favre averaging* procedure is mostly employed [145]. The compressible turbulent flow begins with averaging process that effectively partitions the dependent variables, such as density, pressure, temperature and velocity. The flow variable f is decomposed into an averaged part $\overline{f}$ and a fluctuating part f', given as

$$f(\boldsymbol{x}, t) = \overline{f}(\boldsymbol{x}, t) + f'(\boldsymbol{x}, t). \tag{2.17}$$

The time average of the mean flow variable is given as

$$\overline{f}(\boldsymbol{x}, t) = \lim_{T \to \infty} \frac{1}{T} \int_{t}^{t+T} f(\boldsymbol{x}, \tau) d\tau. \tag{2.18}$$

From Equations (2.17) and (2.18), the time average of the fluctuating part f' is zero. The Favre averaging procedure applied to the momentum Equation (2.5) generates the extra convection term called Reynolds stress tensor. The most straightforward approach is to associate the unknown Reynolds stresses with the computed mean flow quantities by means of a turbulence model. The detailed theory of Reynolds averaging and Favre averaging for compressible flows are presented in [146, 147].

Several turbulent models are available in the literature [147]. The Mixing-length model of Prandtl (1925) introduced a straight-forward prescription for computing the eddy viscosity in terms of mixing-length. Models based on the mixing-length hypothesis are called algebraic models or zero-equation models of turbulence. To develop a realistic metamathematical model for turbulent stresses, Prandtl (1945) introduced an exact equation for turbulent kinetic energy k. These types of models are known as the one-equation model. However, the model is incomplete as it does not provide the turbulence length scale. Kolmogorov (1942) proposed the first complete model of turbulence called the $k - \omega$ model, where ω represents the rate of dissipation of energy in unit volume and time. This model is termed the two-equation model. Chou (1945) proposed modelling the exact equation for dissipation rate ε. Rotta (1951) devised a plausible model for the differential governing evolution of the tensor that represents the turbulent stresses, i.e., the Reynolds stress tensor. Such models are refer to as stress-transport models. After 1950's all these methods are evolved with several advantages. Later, another two-equation model called $k - \varepsilon$ model was proposed by Launders (1972), where ε is a dissipation rate of turbulent kinetic energy. Widespread use of the $k - \varepsilon$ model began with the 1972 formulation. Many improvements have been taken place for each of the mentioned methods with the help of computer resources [147]. The models based on $k-\varepsilon$ model are standard $k-\varepsilon$ model [148], Realisable $k-\varepsilon$ model [149] and RNG $k-\varepsilon$ model [150]. In this work, the standard $k-\varepsilon$ turbulence model [148] is used, and the equations are discussed below.

$k-\varepsilon$ *model*

One of the popular turbulence model is the $k-\varepsilon$ model. In CFD modelling there are other resolution methods more accurate than the $k-\varepsilon$ model. For the specific problem, losses due to the viscous stresses are negligible compared to pressure forces. The adoption of higher order turbulence models would have increased the computational time with no relevant improvement of the results. Therefore, the $k-\varepsilon$ model is used as it is numerically robust, computationally efficient and it provides good accuracy [136, 151]. The model parameters k and ε refer to the turbulent kinetic energy and the turbulent kinetic energy dissipation rate, respectively. The standard $k-\varepsilon$ model is based on the following two equations [152]. The first equation is

$$\frac{\partial}{\partial t}\int_{\Omega(t)} \rho k d\Omega + \int_{\partial\Omega(t)} \rho k \left(\boldsymbol{v}-\boldsymbol{v}_s\right)\cdot\boldsymbol{n} ds = \int_{\partial\Omega(t)} \left(\mu+\frac{\mu_t}{\sigma_k}\right)(\nabla k\cdot\boldsymbol{n})ds + \int_{\Omega}\left(G_t-\rho\boldsymbol{\varepsilon}\right)d\Omega, \tag{2.19}$$

where μ_t is the turbulent viscosity, $\sigma_k = 1$ is the turbulent Prandtl number and G_t is the turbulent generation term. The second equation is

$$\begin{aligned}\frac{\partial}{\partial t}\int_{\Omega(t)} \rho\epsilon d\Omega + \int_{\partial\Omega(t)} \rho\varepsilon \left(\boldsymbol{v}-\boldsymbol{v}_s\right)\cdot\boldsymbol{n} ds = &\int_{\partial\Omega(t)} \left(\mu+\frac{\mu_t}{\sigma_\varepsilon}\right)(\nabla\boldsymbol{\varepsilon}\cdot\boldsymbol{n})ds \\ &+ \int_{\Omega}\left(c_1 G_t\frac{\varepsilon}{k} - c_2\rho\frac{\varepsilon^2}{k}\right)d\Omega,\end{aligned} \tag{2.20}$$

where $\sigma_\varepsilon = 1.3$ is turbulent Prandtl numbers for the turbulent kinetic energy dissipation rate and $c_1 = 1.44$ and $c_2 = 1.92$ are constants. The turbulent kinetic energy k is defined as

$$k = \frac{1}{2}\left(\boldsymbol{v}'\cdot\boldsymbol{v}'\right), \tag{2.21}$$

with $\boldsymbol{v}'$ being the turbulent fluctuation velocity, and the dissipation rate ε of the turbulent kinetic energy is defined as

$$\varepsilon = 2\frac{\mu}{\rho}\overline{(\boldsymbol{S}':\boldsymbol{S}')}, \tag{2.22}$$

where the overbar indicates for time average, the colon : represents a contraction over the same tensor and the strain tensor $\boldsymbol{S}'$ is defined as

$$\boldsymbol{S}' = \frac{1}{2}(\nabla\boldsymbol{v}' + (\nabla\boldsymbol{v}')^T). \tag{2.23}$$

The turbulent viscosity μ_t is calculated by

$$\mu_t = \rho C_\mu \frac{k^2}{\epsilon}, \tag{2.24}$$

with $C_\mu = 0.09$. The turbulent generation term G_t can be expressed as a function of the velocity and the shear stress tensor as

$$G_t = \boldsymbol{\tau}' : \nabla \boldsymbol{v}; \quad \boldsymbol{\tau}' = -\rho \overline{\boldsymbol{v}' \otimes \boldsymbol{v}'} \tag{2.25}$$

where $\tau'_{ij} = -\rho \overline{u'_\iota u'_j}$ is the turbulent Reynolds stress, which can be modeled by the Boussinesq hypothesis [153, 154] by relating the mean velocity gradients in almost the same way that the viscous stresses are related to the complete velocity gradients.

$$\boldsymbol{\tau} = 2\mu_t \boldsymbol{d} - \frac{2}{3}(\rho k + \mu_t \operatorname{tr}\boldsymbol{d})\boldsymbol{I}. \tag{2.26}$$

Equations (2.19)-(2.26) are used to build a turbulent model in the present study.

2.3.2 The cavitation model

Cavitation is the formation of vapour followed by an immediate implosion of cavities in a liquid. Cavitation happens as the consequence of forces acting upon the liquid.

- Cavitation occurs, when there is a rapid change in the pressure that causes the formation of cavities in the liquid where the pressure is low.
- When there is a high pressure, the voids implode and generate an intense shock wave.

The practical and general purpose of the cavitation model is as follows:

- In mechanical devices, the low-pressure regions where cavitation occurs are the same regions with high velocity. In such regions, the velocity slip between the liquid phase and vapour phase are small.
- Usually, the generated vapour takes the form of small bubbles. While such flows can be characterised by a more rigorous two-fluid approach, which allows for velocity slip between the liquid and vapor phases. The computed flow fields strongly depend upon the physical models used for the computation of local bubble sizes and interface drag forces.

The cavitation model in [155] describes the cavitation vapour distribution formulation. The basic approach consists of using the standard viscous flow (Navier-Stokes) equations for variable fluid density and a conventional turbulence model. The fluid density ρ is a function of vapour mass fraction f_v, which is computed by solving a transport equation coupled with the mass and momentum conservation equations. The $\rho - f_v$ relationship is defined as

$$\frac{1}{\rho} = \frac{f_v}{\rho_v} + \frac{1 - f_v}{\rho_l}, \tag{2.27}$$

and the vapour volume fraction α_v is defined as,

$$\alpha_v = f_v \frac{\rho}{\rho_v}. \tag{2.28}$$

The cavitation vapour distribution is described using the following evolution equation

$$\frac{\partial}{\partial t} \int_{\Omega(t)} \rho f_v d\Omega + \int_{\partial\Omega(t)} \rho f_v (\boldsymbol{v} - \boldsymbol{v_s}) \cdot \boldsymbol{n} ds = \int_{\partial\Omega(t)} \left(D_{f_v} + \frac{\mu_t}{\sigma_{f_v}} \right) (\nabla f_v \cdot \boldsymbol{n}) ds + \int_{\Omega} (R_e - R_c) d\Omega, \tag{2.29}$$

where D_{f_v} is the diffusivity of the vapour mass fraction f_v, σ_{f_v} is the turbulent Schmidt number and μ_t is the turbulent viscosity. The source terms R_e and R_c are vapour generation and the condensation rate. These source terms are function of flow parameters (pressure, velocity) and fluid properties (liquid and vapour phase densities, saturation pressure, and liquid-vapour surface tension). In particular, R_e and R_c originate form the bubble dynamics equation, which is derived from the Rayleigh-Plesset equation as in [156]. The Rayleigh-Plesset equation provides the physical strategy to introduce the effects of bubble dynamics into the cavitation model. The vapour generation R_e and the condensation rate R_c are described as

$$R_e = C_e \frac{\sqrt{k}}{\sigma} \rho_l \rho_v \left[\frac{2}{3} \frac{(p_v - p)}{\rho_l} \right]^{\frac{1}{2}} (1 - f_v), \tag{2.30}$$

$$R_c = C_c \frac{\sqrt{k}}{\sigma} \rho_l \rho_l \left[\frac{2}{3} \frac{(p - p_v)}{\rho_l} \right]^{\frac{1}{2}} f_v, \tag{2.31}$$

where k is the turbulent kinetic energy and C_e and C_c are model constants. The theory of bubble dynamics is described in detail in [155].

2.3.2.1 Final form of full cavitation model

The full cavitation model also involves the effect of Non-condensable Gases (NCG); as in most engineering equipment, the operating liquid contains a finite amount of NCG in a dissolved state, or due to leakage or by aeration. According to [157], even a small amount of NCG can have a significant effect on the performance of the equipment. The primary effect of NCG is due to the gas expansion at low pressure which can lead to significant values of local gas volume fraction which has an impact on density, velocity and pressure distributions. The secondary effect of NGC can be via an increase in the phase change threshold, which can be neglected due to lack of general correlation. The full cavitation model is derived considering the fluid as a mixture of liquid, liquid-vapor and NCG. The calculation of the mixture density is derived by modifying Equation (2.27) such that

$$\frac{1}{\rho} = \frac{f_v}{\rho_v} + \frac{f_g}{\rho_g} + \frac{1 - f_v - f_g}{\rho_l}. \tag{2.32}$$

The volume fraction of NCG and liquid are modified as

$$\alpha_g = f_g \frac{\rho}{\rho_g}; \quad \alpha_l = 1 - \alpha_v - \alpha_g. \tag{2.33}$$

The density of the NCG is derived as $\rho_g = \frac{Wp}{RT}$, where W is the molecular weight of non-condensible gas and $R = R_e - R_c$. Considering the NCG effect, Equations (2.30) and (2.31) can be rewritten as

$$R_e = C_e \frac{\sqrt{K}}{\sigma_l} \rho_l \rho_v \left[\frac{2}{3} \frac{(p_v - p)}{\rho_l} \right]^{\frac{1}{2}} (1 - f_v - f_g), \tag{2.34}$$

$$R_c = C_c \frac{\sqrt{K}}{\sigma_l} \rho_l \rho_v \left[\frac{2}{3} \frac{(p - p_v)}{\rho_l} \right]^{\frac{1}{2}} f_v, \tag{2.35}$$

According to [155], the model constants are determined by performing many experiments covering a wide range of operating conditions.

2.4 Spatial and temporal discretisation

2.4.1 Finite volume method

Finite Volume Method (FVM) is a numerical technique used for spatial discretisation through which an integral form of the conservation laws is directly discretised in the physical space. FVM was first introduced in the 1970s by McDonald, MacCormack and Paullay for the numerical solution of fluid mechanics problems [158, 159]. Since then the FVM has become one of the most popular discretisation techniques in the

CFD community. FVM, unlike the finite element method, is based on the notion of a locally conservative discretisation which takes into account the integral formulation of the conservation laws; this is perhaps one of its biggest advantages, since it automatically satisfies the local conservation property of the primary variables such as mass, momentum and energy at the discrete level. In FVM, the problem domain is divided into a set of non-overlapping control volumes called finite volumes. FVM is suitable for complex geometries, due to their ability to accommodate any type of grids, either structured (see Figure 2.5a) or unstructured (see Figure 2.5b) [160].

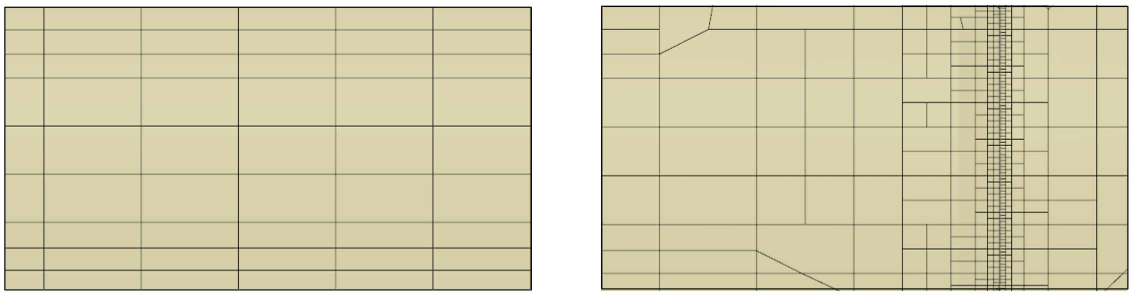
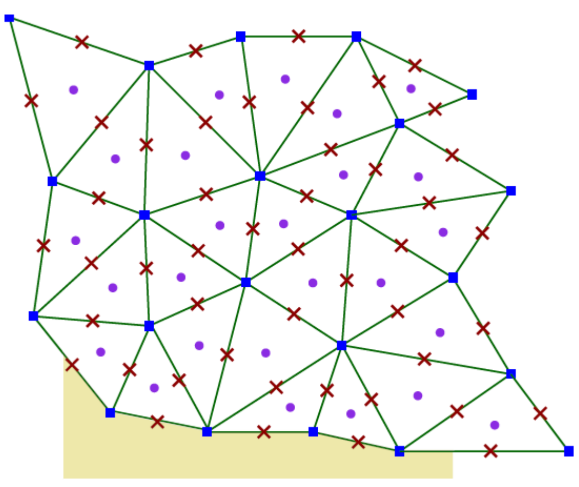

(a) Structured grid (b) Unstructured grid

Figure 2.5: Numerical grid types (a) structured grid and (b) unstructured grid.

There are two major types of finite volume grids, namely cell centred and vertex centred, these are shown in Figures 2.6a and 2.6b, respectively. In the cell centred method, the unknown variables are located at the center of the cell, while in a vertex-centered approach, the variables are located at the grid points. In the latter, the control volume can either be the union of all cells connected to a grid point giving rise to overlapping control volumes or some volume centred around the grid point thereby creating a dual (non-overlapping) control volume. The main advantage of the cell centred method is that dealing with interfaces are simpler, as the control volumes are clearly defined for different mediums; this becomes important in the concept of multi-material flows. The advantage of the vertex centred method is in its exceptional handling of boundaries since the unknown quantities can be explicitly specified at the physical boundary [160]. The focus of the present study is aimed at cell centred FVM which will be explored in the following section.

2.4.1.1 Cell-centred finite volume method

Discretisation of the solution domain produces a computational mesh on which the governing equations are subsequently solved. In the cell centred FVM method, the control volumes coincides with the cell. For ease of understanding, a nomenclature is introduced


(a) Cell centred FVM

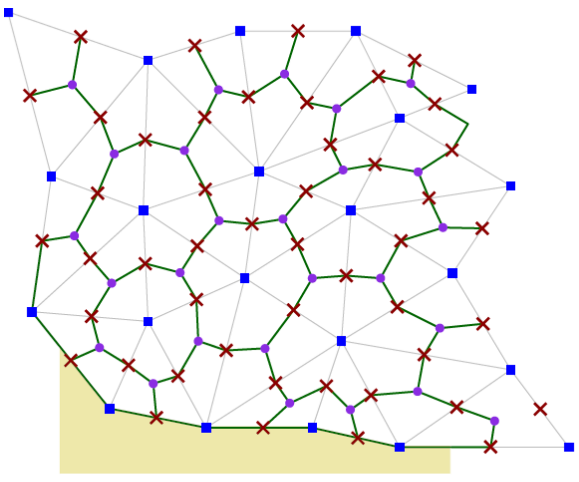

(b) Vertex centred FVM

FIGURE 2.6: The two types of FVMs: (a) Cell centred and (b) vertex centred. The notations are presented in Figure 2.7.

in Figure 2.7 to elaborate the finite volume discretisation and for simplicity a quadrilateral mesh is chosen, where P represents cell centroid of the cell, f represents the face centre and a represents the node. For good accuracy, it is necessary for the order of the discretisation to be equal or higher than the order of the equation that is being discretised [161].

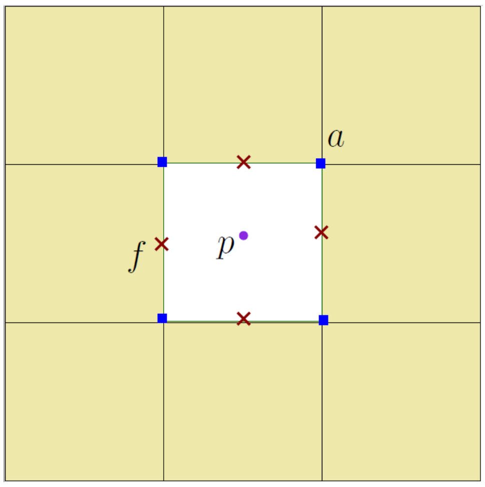


FIGURE 2.7: Cell centred FVM nomenclature.

The integral form of the conservation equations in Equations (2.4) and (2.5) and the cavitation equation in Equation 2.29 are discretised using the standard cell centred FVM as,

$$|\Omega_P| \frac{d(\rho_P)}{dt} + \sum_{f=1}^{n_f} \rho_f (\boldsymbol{v}_f - \boldsymbol{v}_{s,f}) \cdot \boldsymbol{n}_f s_f = 0, \tag{2.36}$$

$$|\Omega_P| \frac{d(\rho_P \boldsymbol{v}_P)}{dt} + \sum_{f=1}^{n_f} \rho_f \boldsymbol{v}_f \otimes (\boldsymbol{v}_f - \boldsymbol{v}_{s,f}) \cdot \boldsymbol{n}_f s_f = \sum_{f=1}^{n_f} \boldsymbol{\tau}_f \cdot \boldsymbol{n}_f s_f - \sum_{f=1}^{n_f} p_f \boldsymbol{n}_f s_f + |\Omega_P| \boldsymbol{f}_P, \tag{2.37}$$

$$|\Omega_P| \frac{d(\rho_P f_{v,P})}{dt} + \sum_{f=1}^{n_f} \rho_f f_{v,f} (\boldsymbol{v}_f - \boldsymbol{v}_{s,f}) \cdot \boldsymbol{n}_f s_f = \sum_{f=1}^{n_f} \left(D_{f_v} + \frac{\mu_t}{\sigma_{f_v}} \right)_f \nabla f_{v,f} \, \boldsymbol{n}_f s_f + |\Omega_P| (R_e - R_c)_P, \tag{2.38}$$

where $\boldsymbol{n}_f s_f = \boldsymbol{S}$, is a face area vector pointing outwards from the control volume Ω_P. Similar to Equation (2.38), the turbulence equations (2.19) and (2.20) can be discretised. The mass flux through the face is defined as,

$$m_f = \rho_f (\boldsymbol{v}_f - \boldsymbol{v}_{s,f}) \cdot \boldsymbol{S}. \tag{2.39}$$

The approximation of these convection fluxes in the transport equations has a decisive influence on the overall accuracy of any numerical solution for fluid flow. Although convection is represented by a simple first order derivative, its numerical representation remains one of the central issues in CFD [162]. The classic first order schemes such as upwind, hybrid, and power-law are unconditionally bounded whereas the higher order schemes, such as the central differencing scheme, the second order upwind and the third-order upwind, offer a route to improve accuracy of the computations. However, higher order schemes suffer from the boundedness problem; that is, the solutions may display unphysical oscillations in regions of steep gradients, which can be sufficiently serious to cause numerical instability. In the present study, a first order accurate upwind scheme is used in order to compromise between accuracy and boundedness. Upwind scheme guarantees the boundedness of the solution through the sufficient boundedness criterion for the system of algebraic equations [161].

Another important term discretised in Equation 2.38 is the diffusion term, which follows as

$$\sum_{f=1}^{n_f} \left(D_{f_v} + \frac{\mu_t}{\sigma_{f_v}} \right)_f \nabla f_{v,f} \, \boldsymbol{n}_f s_f = \sum_{f=1}^{n_f} \left(D_{f_v} + \frac{\mu_t}{\sigma_{f_v}} \right)_f (\nabla f_{v,f}) \cdot \boldsymbol{S}. \tag{2.40}$$

If the mesh is orthogonal, i.e. vectors $\boldsymbol{d}$ and $\boldsymbol{S}$ are parallel, shown in Figure 2.8a. It is possible to use central difference scheme to approximate the first order derivative.

$$\boldsymbol{S} \cdot (\nabla f_{v,f}) = |\boldsymbol{S}| \frac{f_{v,N} - f_{v,P}}{\boldsymbol{d}}. \tag{2.41}$$

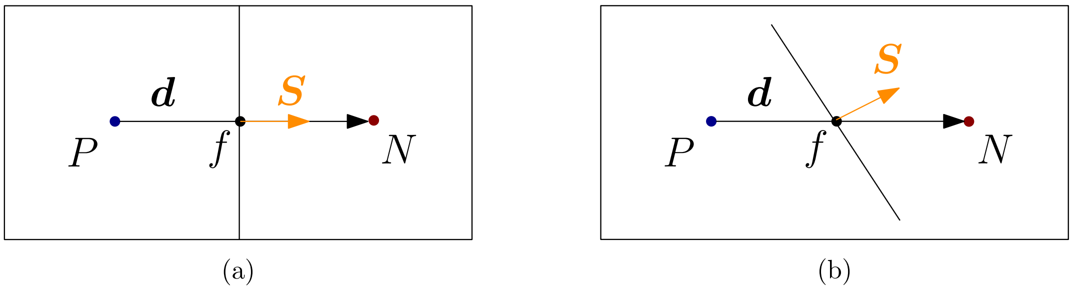


FIGURE 2.8: Vectors $\boldsymbol{d}$ and $\boldsymbol{S}$ on a orthogonal (a) and non-orthogonal (b) mesh.

If the mesh is non-orthogonal, as shown in Figure 2.8b

$$\boldsymbol{S} \cdot (\nabla f_{v,f}) = |\triangle| \frac{f_{v,N} - f_{v,P}}{\boldsymbol{d}} + \boldsymbol{k} \cdot (\nabla f_{v,f}), \tag{2.42}$$

where the first term and second term of the right hand side are orthogonal and non-orthogonal contribution terms, respectively. The two vector introduced in Equation (2.42), $\triangle$ and $\boldsymbol{k}$, have to satisfy the following condition:

$$\boldsymbol{S} = \triangle + \boldsymbol{k}. \tag{2.43}$$

Central difference scheme is second order accurate but involves larger truncation error, that makes it less stable. To overcome the disadvantage of large truncation error, some correction approaches are used when the mesh is non-orthogonal. Many possible approaches are available, including minimum correction approach, orthogonal correction approach and over-relaxed approach. The detailed theory related to the diffusion term can be found in [163]. Similarly, discretisation in space has been performed for all the equations solved in the CFD simulation.

2.4.2 Temporal discretisation

For transient simulations, the governing equations must be discretised in both space and time. The spatial discretisation has been presented in the previous section. Temporal discretisation involves the integration of every term in the differential equations over a time step Δt. If time derivative is discretised using backward differences, the first order accurate temporal discretisation of the temporal term in Equation (2.36) is given by

$$\frac{(\rho_P)^{t+\Delta t} - (\rho_P)^t}{\Delta t} = F(\rho_P)^{t+\Delta t}, \tag{2.44}$$

where F incorporates any spatial discretisation. The second order discretisation for the same equation is given by

$$\frac{(3\rho_P)^{t+\Delta t} - 4(\rho_P)^t + (3\rho_P)^{t-\Delta t}}{2\Delta t} = F(\rho_P)^{t+\Delta t}. \tag{2.45}$$

Time integration can also be conducted using alternatively an explicit approach. In the explicit approach, the function $F(\rho_P)$ is evaluated at current time step. For instance, Equation (2.44) becomes,

$$\frac{(\rho_P)^{t+\Delta t} - (\rho_P)^t}{\Delta t} = F((\rho_P)^t), \tag{2.46}$$

and a similar expression results from Equation (2.45).

In the present thesis, the focus is on fully implicit time discretisation with first order accuracy, obtained through the application of the backward Euler method. Using backward Euler method, $(\rho_P^{t+\Delta t})$ is calculated as,

$$(\rho_P^{t+\Delta t}) = (\rho_P)^t + \Delta t F((\rho_P)^{t+\Delta t}). \tag{2.47}$$

For further details on time interpolation, refer to [163, 164].

2.5 Chapter conclusion

The present chapter has presented the governing equations, the turbulence and cavitation equations solved during the CFD simulations have been discussed along with kinematics descriptions of a flow field. In order to discretise the governing equations, the spatial and temporal discretisation methods have been discussed. With the present chapter, the high-fidelity CFD model has been achieved, which will be used to perform CFD simulation of an external gear pump using PumpLinx®. The workflow of PumpLinx® is presented in Appendix A. The numerical examples of the CFD simulations are discussed in Chapter 3.

Chapter 3

HIGH-FIDELITY COMPUTATIONAL RESULTS

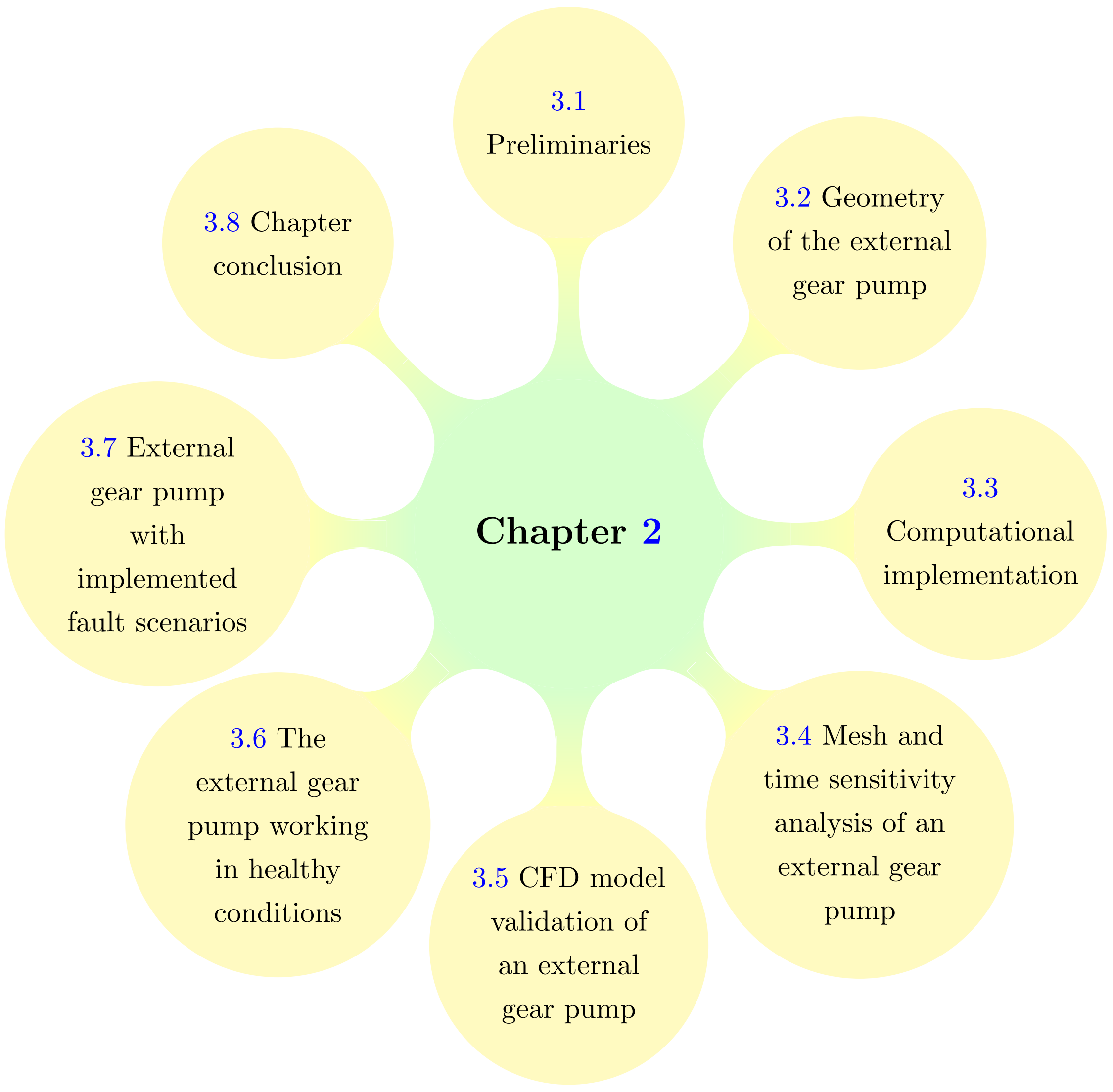

3.1 Preliminaries
3.2 Geometry of the external gear pump
3.3 Computational implementation
3.4 Mesh and time sensitivity analysis of an external gear pump
3.5 CFD model validation of an external gear pump
3.6 The external gear pump working in healthy conditions
3.7 External gear pump with implemented fault scenarios
3.8 Chapter conclusion
Chapter 2

FIGURE 3.1: Structure of Chapter 3.

3.1 Preliminaries

This chapter provides the numerical examples of a CFD simulation for an external gear pump. The generation of data via a CFD model of the pump under various working conditions will be discussed. The background theory of the CFD model has been presented in Chapter 2. In particular, all the provided results are from the 3D transient simulation of the pump using PumpLinx with a specific cavitation model. The chapter is outlined as follows: the geometry of the external gear pump (see Section 3.2), the computational implementation including the relevant mesh generation using PumpLinx (see Section 3.3), the mesh and time sensitivity analysis (see Section 3.4), experimental validation of the CFD model (see Section 3.5), the gear pump running in a healthy (normal) condition (see Section 3.6), and the implemented fault scenarios (see Section 3.7). The structure of this chapter is presented in Figure 3.1.

3.2 Geometry of the external gear pump

FG304 series are rotating positive displacement external gear pumps, where the fluid is displaced from the inlet side of the pump to the outlet side of the pump as a result of the intermeshing of the drive and the slave gear's teeth. The geometry of the external gear pump, that is of the interest to the present thesis, is shown in Figure 3.2.

The gears are located inside a tight and leak free casing, and they are connected to the lubrication channel through the top channel with rotating shafts. Similarly, the back side of the gear is connected to the magnet via a bottom channel with rotating shafts. The inlet and outlet ports are connected to the sides of the gear. The inlet and outlet sensor are placed in the inlet port and outlet port, respectively. The outlet sensor is placed away from the end, in order to avoid the reversing flow and achieve an accurate measurement of the pressure distribution in the outlet port. During the operation, these types of pumps are noiseless, pulsation-free and capable of handling liquids up to 95 °C with pressure up to 12 bar and flow rate up to 210 l/h [2].

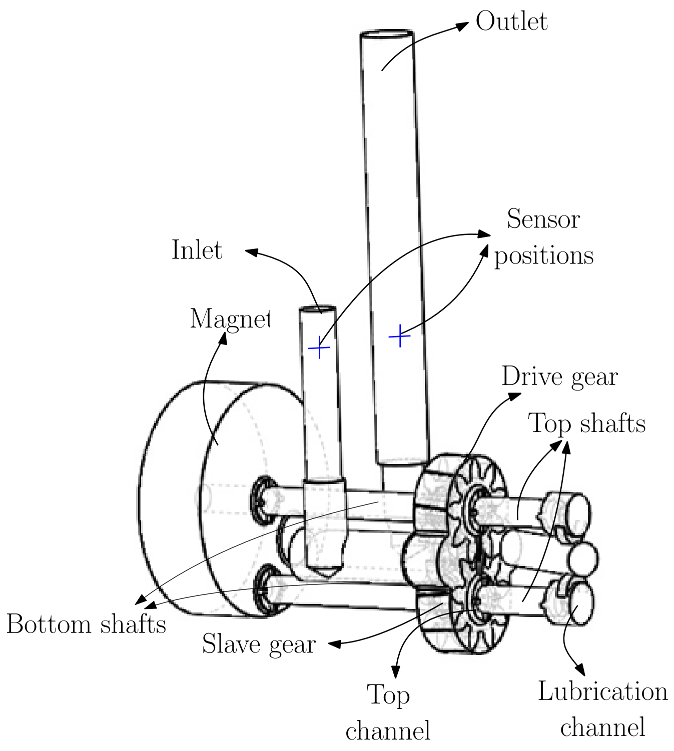


FIGURE 3.2: Geometry of the FG304 series external gear pump.

3.3 Computational implementation

3.3.1 PumpLinx

As stated in Chapter 1, the CFD model has been built using a commercial software PumpLinx (developed by Simerics Inc.). The software has been chosen after precisely assessing its capability to study several common problems in the hydraulic components. PumpLinx solves numerically the fundamental conservation equations of mass, momentum, and energy and includes accurate physical models for turbulence and cavitation [136]. The mathematical model and governing equations are discussed in Section 2.3. Moreover, PumpLinx grids use a body-fitted binary tree approach [136, 137, 165]. These types of grids are accurate and efficient because

- ☐ The parent-child tree architecture allows for an expandable data structure with reduced memory storage.
- ☐ Binary refinement is optimal for transitioning between different length scales and resolutions within the model.

- ☐ The majority of cells are cubes, which is the optimum cell type in terms of orthogonality, aspect ratio, and skewness thereby reducing the influence of numerical errors and improving speed and accuracy.
- ☐ It can be automated, greatly reducing the set-up time.
- ☐ Since the grid is created from a volume, it can tolerate dirty CAD surfaces with small cracks and overlaps.

In the boundary layer, the binary tree approach can easily increase the grid density on the surface without excessively increasing the total cell count. In the regions of high curvature and small details, the grid is subdivided and cut to conform to the surface.

PumpLinx code permits the simultaneous application of moving and stationary grids. The moving grid uses the ALE approach, while the stationary grid uses the Eulerian approach. These aspects are described in Section 2.2. In PumpLinx, each control volume connects to the others through an implicit interface called a mismatched grid interface. A mismatched grid interface is an efficient implicit algorithm that identifies the overlap areas and matches them without interpolation, then the area is treated as the common face connecting the cells on both sides of the interface. The common faces are then treated as two neighbouring cells during the simulation. The solution resulting from this approach is very robust and accurate. The simulation workflow of PumpLinx is presented in Appendix A.

3.3.2 Model attributes

The fluid domain of the external gear pump is extracted from the Computer Aided Design (CAD) of the pump, which is provided by the industrial partner F-Lab and the geometry is imported into PumpLinx. The geometry of the external gear pump can be found in Section 3.2. The mesh of the entire fluid domain is generated using high quality structured and unstructured hexahedral cells in PumpLinx, shown in Figure 3.3. For high curved regions or narrow cut regions, the binary tree unstructured mesh is utilised to create high-quality hexahedral cells. In addition, PumpLinx has a specialised tool for the meshing of the internal geometry of the gear using a rotor gear template mesh, producing high quality structured hexahedral cells [137]. The rotor mesh is adapted automatically according to the angle of the rotation of the gears. A brief section of gear template mesh is shown in Figure 3.4a and the interface between the moving and stationary mesh are displayed in Figure 3.4b.

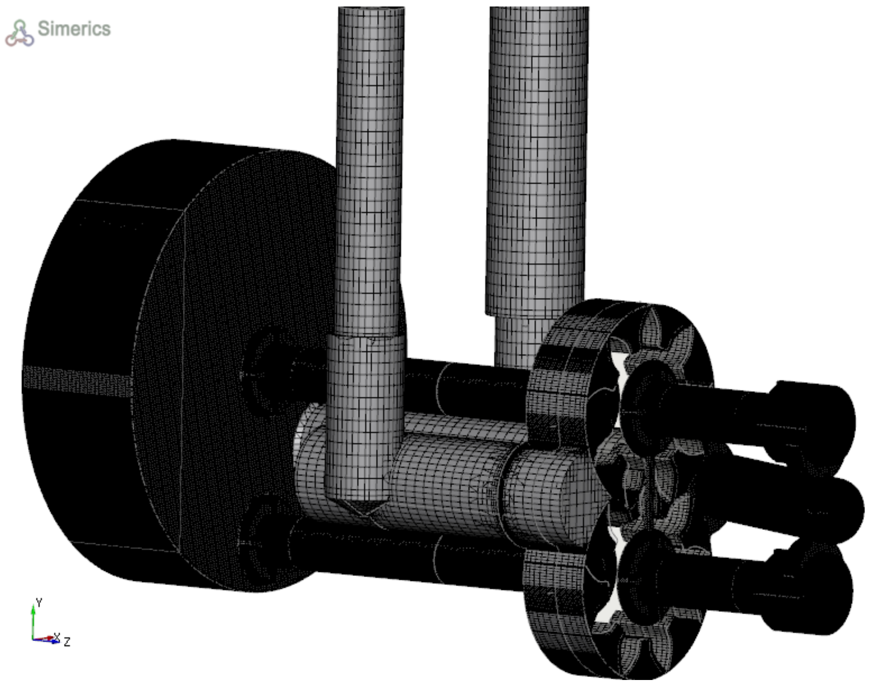


FIGURE 3.3: Mesh of the external gear pump (the entire computational domain) created using PumpLinx.

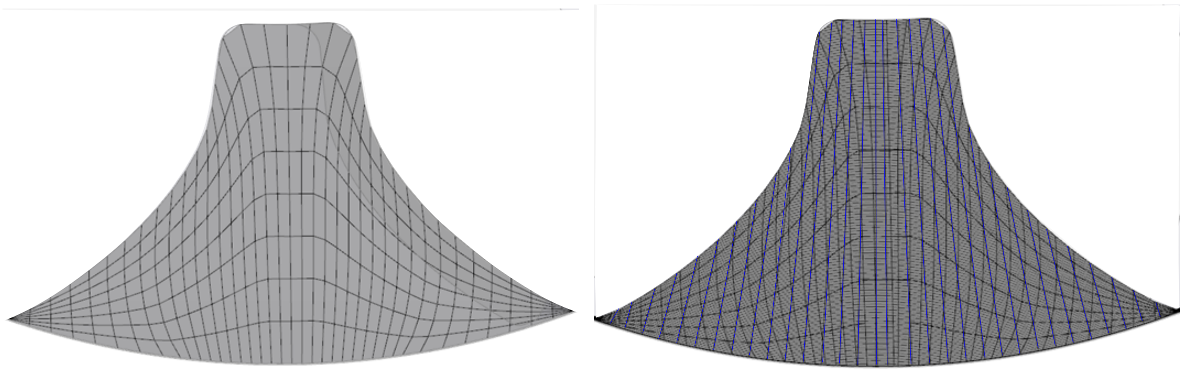


FIGURE 3.4: Gear template mesh (a) and interface between the moving grid (highlighted in blue) and stationary grid (b) of the gear pump using PumpLinx.

In order to perform CFD the simulation of this gear pump, the conservation of mass (Equation (2.4)) and momentum equations (Equation (2.5)) are solved along with the turbulence model equations (Equation (2.19)-(2.26)) and the cavitation model equation (Equation (2.29)). The cavitation model employed is solved taking into account liquid properties, cavitation, aeration, and liquid compressibility. The simulation of an external gear pump running in healthy conditions uses the information presented in Table 3.1. Typical boundary conditions are provided for flow variables, turbulence and cavitation models in Table 3.2. The standard wall function for turbulent boundary condition is discussed in Appendix A. The rotation speed of the pump is specified in RPM. All the

CFD simulations presented in the present thesis utilise the conditions and properties discussed in this section.

Operating conditions of the gear pump	
Speed	1400 rpm
Radial gap	0.03 mm
Number of drive gear teeth	9
Number of slave gear teeth	9
Material properties of the gear pump	
Viscosity	0.0383 Pa-s
Density	800 kg/m^3
Temperature	300 K

TABLE 3.1: Properties of a healthy (normal working condition) gear pump.

	Inlet	Outlet	Wall
Flow:			
Pressure	1.01325 bar	3.15 bar	Zero gradient
Velocity	Zero gradient	Zero gradient	No-slip condition
Turbulence:			
Kinetic energy	0.01 m^2/s^2	0.01 m^2/s^2	Standard wall function[166]
Dissipation rate	1 m^2/s^2	1 m^2/s^2	Standard wall function
Cavitation:			
Vapour mass fraction	0	0	Zero gradient

TABLE 3.2: Boundary conditions of flow variables, turbulence and cavitation models [6].

3.4 Mesh and time sensitivity analysis of an external gear pump

In the present section, firstly, the mesh sensitivity study is discussed then the time sensitivity study are discussed. The main goal of the present study is not only to obtain an accurate solution, but also to investigate its stability and the computational efficiency.

Mesh sensitivity analysis
A mesh sensitivity study is performed on the model considered, whereby a finer and finer mesh is used until a threshold of accuracy is passed going from one model to the next [167]. Alternatively, the optimal mesh is the coarsest mesh possible that produces results within an acceptable error interval. For simplicity and to optimise the number of cells or control volumes in the mesh, a simpler geometry is considered for this analysis. The geometry is shown in Figure 3.2; however, the inlet port, the outlet port, and external gear are only considered. The boundary conditions and material properties are presented in Table 3.1. Simulations are performed for extremely small fixed time step with four different grid sizes (h_1, h_2, h_3, h_4) ranging from coarse to fine mesh. In PumpLinX software, there is no explicit way of changing the cell size, so min and max cell sizes are adopted to perform this analysis. The coarse mesh (h_1) and the finest mesh (h_4) are presented in Figure 3.5. The properties of different grid sizes and their execution time are shown in Table 3.3.

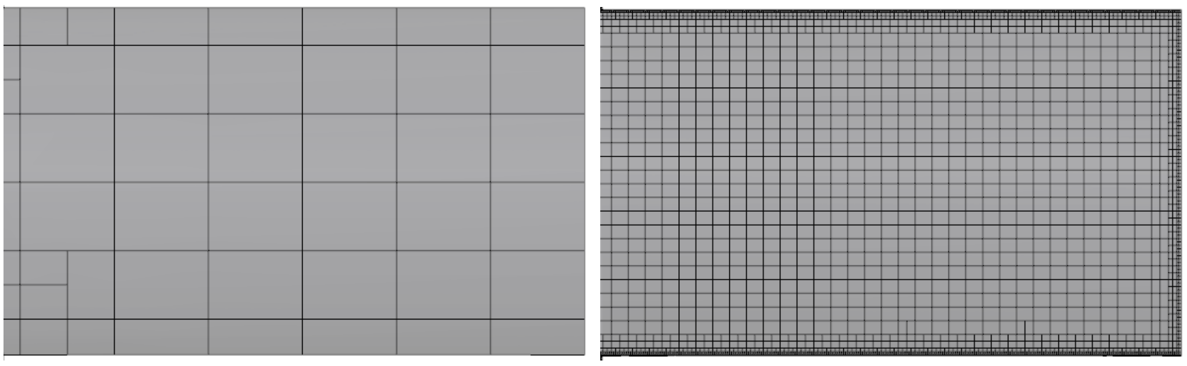

(a) (b)

FIGURE 3.5: Coarse mesh h_1 (a) and fine mesh h_4 (b) created using PumpLinx.

	h_1	h_2	h_3	h_4
No of elements	44518	247495	446393	4605699
Max cell size	0.1	0.07	0.03	0.018
Min cell size	0.005	0.005	0.002	0.001
Execution time	1	3.24	7.57	182.76

TABLE 3.3: Mesh sensitivity analysis: execution time is in a dimensionless quantity, and h_1 is used as a reference.

Figure 3.6 shows the standardised[1] discharge flow rate based on four different mesh sizes. Mesh size increases from coarse (h_1) to fine mesh (h_4). Mesh sizes, h_3 and h_4 have the same behaviour with a very small difference in values. Considering the amount of cost and time for h_3 and h_4, and comparing the results, h_3 can be chosen as an optimal size of the mesh for the rest of the simulations. Figure 3.7 shows the relative error percentage of pressure, flow rate and torque at a chosen time step for the different number of cells. The error is calculated by assuming the finest mesh (h_4) as a benchmark reference. As the number of cells increases, the relative error decreases. In case of h_3 mesh, even though the accuracy of the spatial discretisation is of first order, the accuracy of the solution is beyond second order; this is due to the efficiency of the PumpLinx. This can also be seen in Figure 3.6 that h_3 and h_4 are in close proximity. As a result of this analysis, mesh size h_3 is used to perform all the CFD analysis in this thesis.

Time sensitivity analysis

A time sensitivity study is performed on the gear pump model considered. The geometry is shown in Figure 3.2. The boundary conditions and material properties are presented in Table 3.1. In PumpLinX, there is no explicit way of changing the time step size. However, the revolution time definition method uses the number of revolutions and time steps per Drive Gear Tooth Rotation (DGTR) to determine the size and number of time steps. The number of time step is calculated as

$$\begin{aligned}\text{Number of time steps} =& (\text{time steps per DGTR}) \\ & (\text{number of drive gear teeth})(\text{number of revolutions}),\end{aligned} \tag{3.1}$$

[1] Standardised refers to z-score normalisation and it expressed as $(\boldsymbol{x} - \mu)/\sigma$, where $\boldsymbol{x}$ is the dataset and μ and σ is mean and standard deviation. Further explanations can be found in Section 5.4.

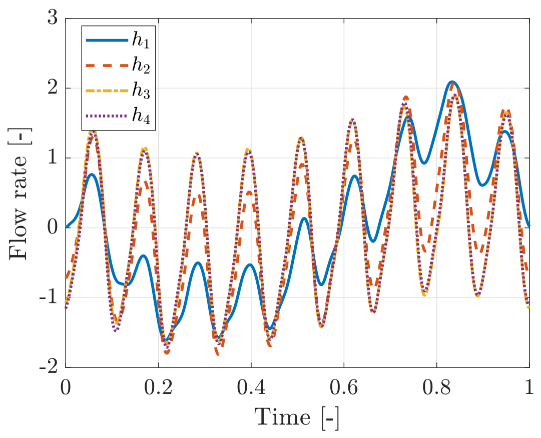


FIGURE 3.6: Standardised discharge flow rate for different mesh sizes (h_1, h_2, h_3, h_4). Mesh size increases from h_1 to h_4. Symbol [-] refers to dimensionless quantity.

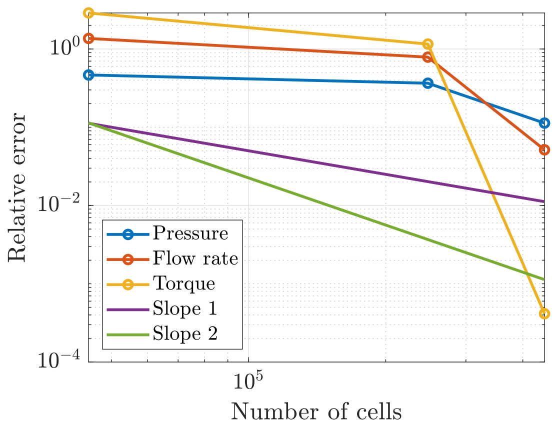


FIGURE 3.7: Relative error of pressure, flow rate and torque (fixed time step) with respect to finest mesh (h_4). Slope 1 is first order slope and Slope 2 is second order slope.

where the time steps per DGTR dictates the number of time steps taken to move the inner tooth one pocket. One pocket corresponds to the average angle between drive gear teeth, based on 360° divided by the specified number of drive gear teeth. The time step size is defined as,

$$\Delta t = \left(\frac{1}{\text{(time steps per DGTR)(number of drive gear teeth)}}\right)(60/rpm). \qquad (3.2)$$

Different sizes of Δt can be achieved by modifying the time steps per DGTR. The numerical simulations are performed for four values of the time steps per DGTR ranging from 10 to 50 while keeping keeping other relevant parameters fixed (i.e, cell size, material properties, etc.). The variation of time steps per DGTR is limited by the computational time required to perform each simulation. Indeed, high value of this variable drastically increase the computational time. This study is intended to evaluate the sensitivity of the choice of time step. All the computations start from a zero initialisation and are stopped at the end of two revolutions. The properties of different time step sizes and their execution time are shown in Table 3.3.

	t_1	t_2	t_3	t_4
Time steps per DGTR	10	30	45	50
Δt	$4.76e^{-04}$	$2.38e^{-04}$	$1.19e^{-04}$	$9.52e^{-05}$
Execution time	1	2.184	2.781	2.915

TABLE 3.4: Time sensitivity analysis: execution time is in a dimensionless quantity, and t_1 is used as a reference.

Figure 3.8 shows the relative error of pressure, flow rate and torque at a chosen time step for the different number of time steps per DGTR. The error is calculated by assuming the highest DGTR (t_4) as a benchmark reference. As the number of time steps per DGTR increases, the relative error decreases. The accuracy of the simulation is first order in time; this has been discussed in Section 2.4. The accuracy of the solution is of the first order for flow rate and torque, and the accuracy of the solution is of the second order for the pressure. As a result of this analysis, time steps per DGTR t_3 is used to perform all the CFD analysis in this thesis.

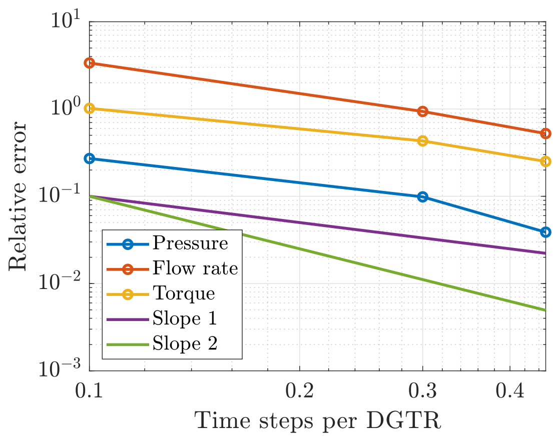


FIGURE 3.8: Relative error of pressure, flow rate and torque (fixed time step) with respect to highest time steps per DGTR (or smallest time step size t_4). Slope 1 is first order slope and Slope 2 is second order slope.

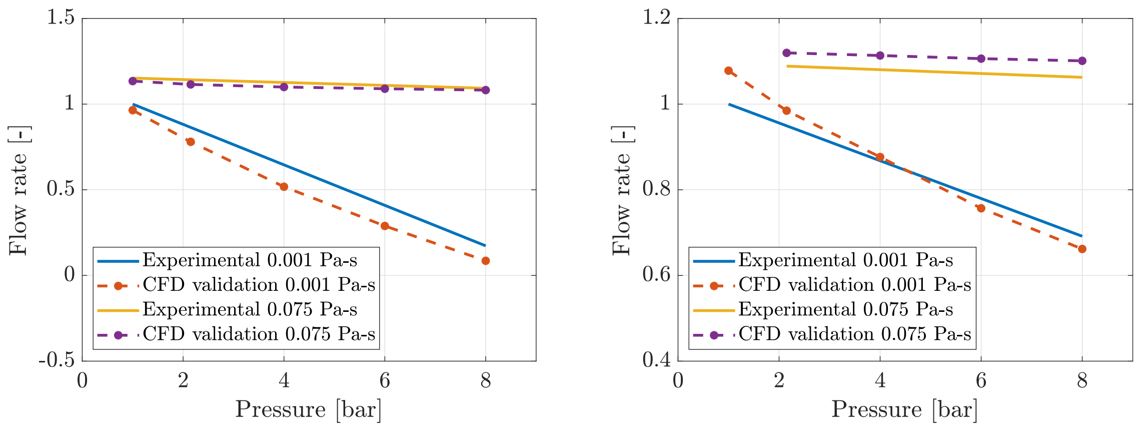
3.5 CFD model validation of an external gear pump

In the present section, the objective is to validate the CFD model against the experimental results of the external gear pump. Similarly, Reference [137] has validated the CFD model against the experimental results of an external gear pump, proving that the CFD simulation model built in PumpLinx® has produced reliable results.

The experiment of the gear pump was performed by the Italian industrial partner F-Lab. The geometry of the gear pump is given in Figure 3.2. The mathematical model of the CFD simulation is presented in Chapter 2. The boundary conditions and material properties are described in Section 3.3.2. The experiments using this gear pump ware conducted by fixing speed and viscosity values for different outlet pressure values $(1, 2, \ldots, 9$ bar). The inlet pressure is assumed to be atmospheric pressure (1.01325) for all the experiments. For each experiment, the discharge flow rate has been measured, and the average of these values have been used for validating the CFD model. In particular, the experiments were performed for two different rotational speed values of the gear, 1450 rpm and 2900 rpm and for two different viscosity values, 0.001 Pa-s and 0.075 Pa-s.

The CFD simulations are conducted similarly to validate their performance with the experimental model. Figures 3.9a and 3.9b show the experimental discharge flow rate values compared with the CFD simulation model for different pressure values of a gear pump with the rotating speed of 1450 rpm and 2900 rpm, respectively. From Figures 3.9a and 3.9b, it can be seen that the simulation results are in very accurate agreement with the experiments. Also, the experimental results related to viscosity 0.075 Pa-s has stronger similarity to the CFD model results than the results related to 0.001 Pa-s. Also, it can be noted that an increase in the viscosity value provides a higher discharge flow rate.

Furthermore, an increase in rotational speed of the gear increases the total gas volume fraction and the velocity in the gear teeth contact regions. Due to confidentiality, the gear region cannot be showed. The total gas volume fraction, which is related to the cavitation model, computing the volume fraction of the free NCG in a liquid for a selected Volume.



(a) 1450 rpm speed (b) 2900 rpm speed

FIGURE 3.9: Mean discharge flow rate of experimental results against CFD model of an external gear pump with speed of 1450 rpm (a) and 2900 rpm (b) for different relative pressure values. In the x-axis, the values are relative pressure calculated using the difference between outlet and inlet pressure.

3.6 The external gear pump working in healthy conditions

In the present section, the objective is to provide the simulation results of the external gear pump under healthy working conditions using PumpLinx. The geometry of an external gear pump is presented in Section 3.2 The model attributes of the simulations including, mesh generation, boundary conditions and materiel properties, are presented in Section 3.3.2. The numerical simulations are conducted using the properties shown in Table 3.1. The gear pump is run under the transient simulation for five gear revolutions with an atmospheric (1.01325 bar) inlet pressure and outlet pressure of 3.15 bar. During the CFD simulations, PumpLinx offers several physical variables in different positions of the pump. Among them, the pressure at the outlet sensor (sensor is shown in Figure 3.2), the discharge flow rate and the sum of drive and slave gear's torque have been considered as primary variables in the present thesis. The physical variables are described in Appendix A.

Figure 3.10 shows the standardised and fully developed outlet pressure, discharge flow rate and torque of the rotating gear for one revolution (x-axis is normalised between 0 and 1). The number of oscillations shown in Figures 3.10a and 3.10b correspond to the number of teeth in the gear, however, in Figure 3.10c, torque has two oscillation per gear tooth referring to the product of velocity and cross-sectional area.

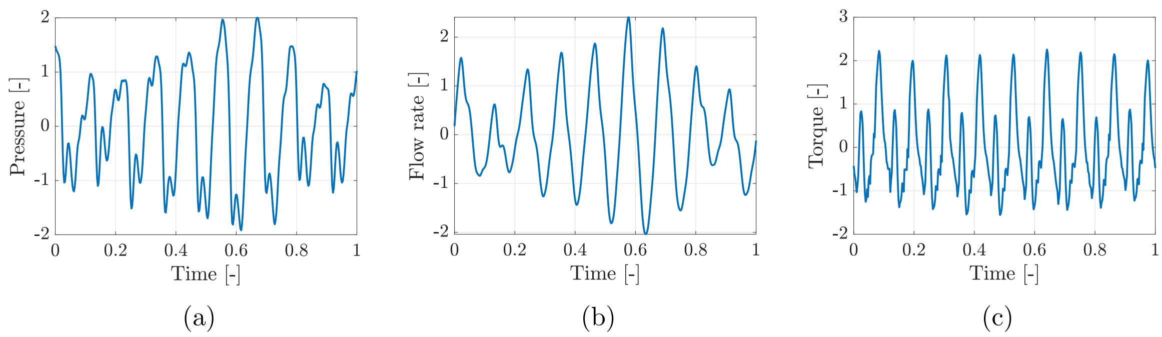


FIGURE 3.10: External gear pump running in a healthy condition; the fully developed pressure (a), flow rate (b) and torque (c) with respect to time for one revolution.

Figure 3.11 shows the pressure distribution of the external gear pump after the end of five revolutions, where the pressure values range from atmospheric pressure (1.01325 bar) to 3.15 bar. The fluid moves from the inlet side of the pump to the outlet side of the pump for each revolution. Figure 3.12 shows the total gas volume fraction, which is obtained by computing the volume fraction of the free NCG in a liquid for a selected volume.

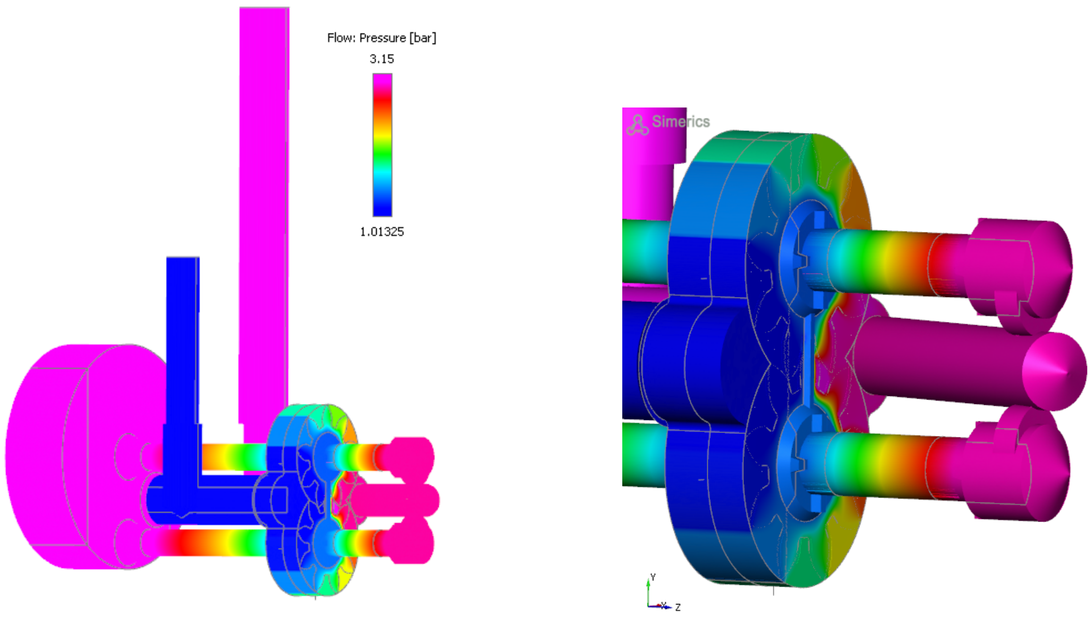


FIGURE 3.11: Pressure distribution of an external gear pump within the range [1.01325 3.15] bar. The result is obtained after five revolutions.

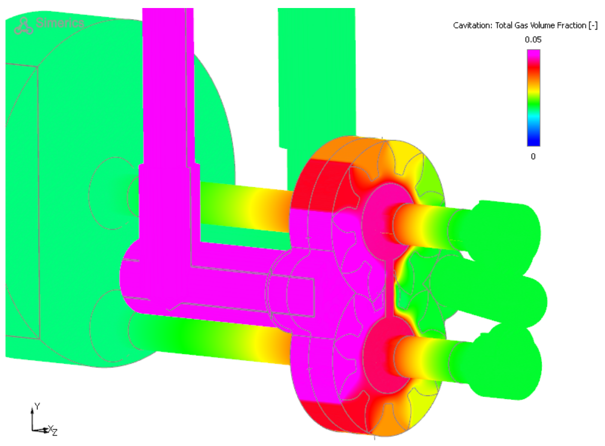


FIGURE 3.12: Total gas volume fraction of an external gear pump within the range [0 0.05]. The result is obtained after five revolutions.

3.7 External gear pump with implemented fault scenarios

Utilising the above-discussed CFD simulation model of a healthy gear pump, in the present section, a variety of working conditions including, several fault scenarios and combinations of fault scenarios, are recreated based on the experience of an industrial partner. Several fault scenarios are considered: namely radial gap degradation, axial gap degradation, speed variations, viscosity variations, temperature variations and clogging/ blocking variations. These scenarios are considered to understand the complete behaviour of the pump when running in an abnormal condition. Figure 3.13 shows the fault scenarios implemented to examine the equipment's health under different working conditions. Table 3.5 shows the fault variables and their different cases (in percentage). Each fault scenario is simulated following boundary conditions and material properties described in Section 3.3.2. Furthermore, the clogging variations and viscosity variations are combined to perform a combination of fault scenarios; this method helps to identify the behaviour of the gear pump when affected by multiple faults.

The CFD simulation is performed for all the scenarios measuring time-varying values of the outlet pressure at the sensor, discharge flow rate and gear's torque. The discharge flow rate is chosen to exhibit the comparisons due to their smooth oscillations. All the results in this section are presented in a dimensionless manner.

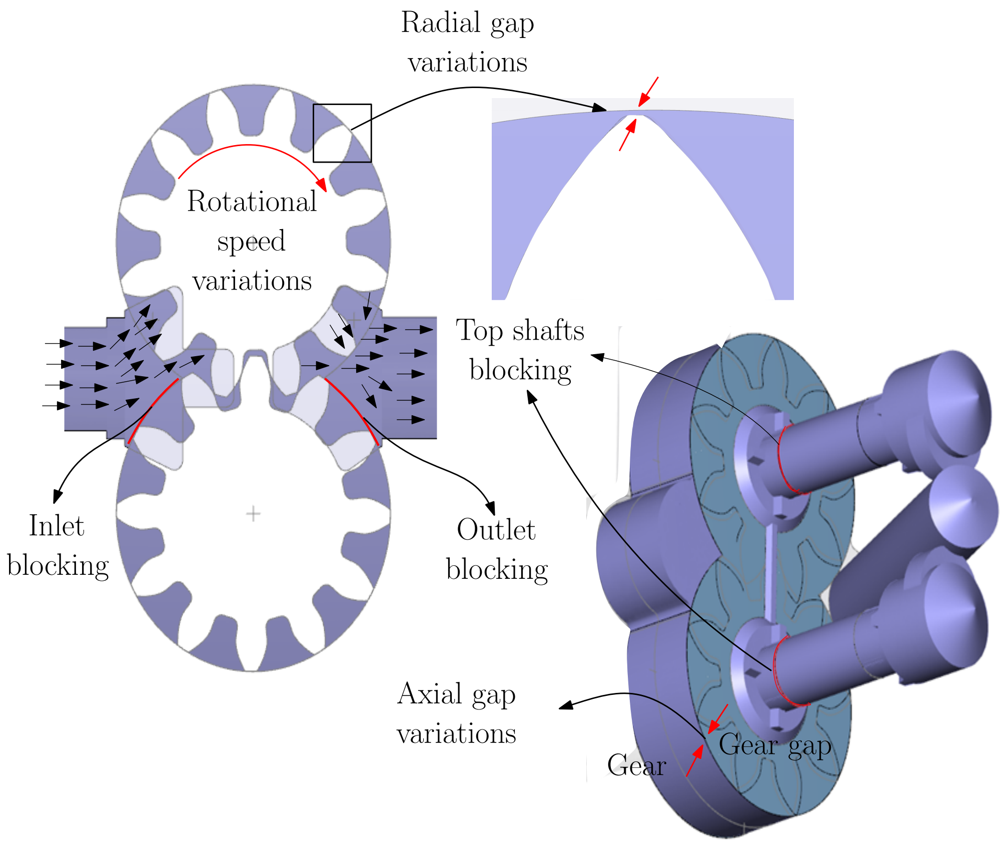


FIGURE 3.13: Implemented fault scenarios considering healthy working condition of the gear pump as a reference.

Fault scenarios	Different cases
Radial gap degradation	$66.66\%, 133.33\%, 200\%$
Axial gap degradation	$2.5\%, 5\%, 7.5\%$
Speed variations	$1.78\%, 3.57\%, 5\%$
Viscosity variations	$0.8\%, 1.6\%, 2.4\%$
Temperature variations	$2.5\%, 5\%, 7.5\%$
Clogging variations	blocked inlet, outlet, top shafts

TABLE 3.5: Implemented fault scenarios; the percentage is calculated considering the healthy working conditions of the gear pump as a reference.

Fault scenarios due to radial gap degradation

The fault scenarios due to radial gap degradation are created based on the concept that mass particles enter the pump causing the gear teeth to erode. The radial gap faults are generated by grinding all the gear teeth. The step-by-step degradation behaviour is generated, namely G1, G2 and G3, to understand the erosion behaviour on the gear teeth. The standardised discharge flow rate of the scenarios (G1, G2, G3) compared with the healthy scenario (H) in time domain and frequency domain is shown in Figure 3.14. Consequently, by increasing the gap size, the mean flow rate decreases. With respect to the healthy scenario, the mean flow rate of G1, G2 and G3 is decreased by 21%, 40% and 56%, respectively. These values are high corresponded to the high percentage of gap introduced for the radial gap degradation.

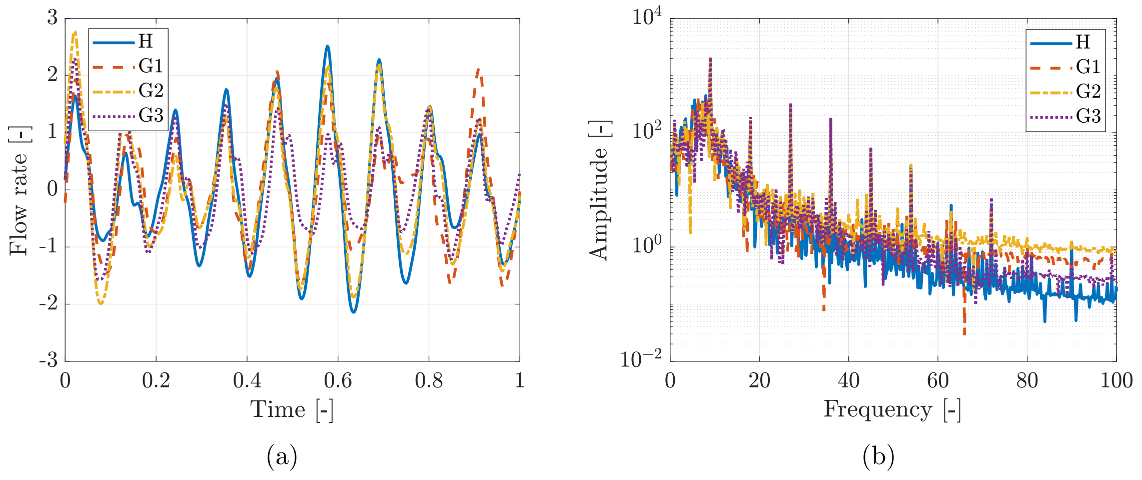


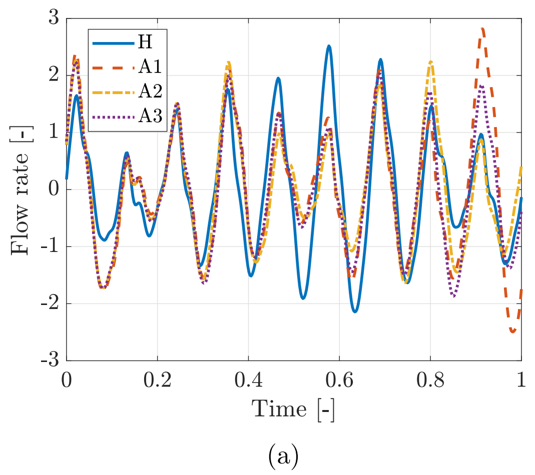
FIGURE 3.14: Discharge flow rate with respect to the time (a) and the frequency (b) for fault scenarios due to radial gap degradation.

Fault scenarios due to axial gap degradation

The fault scenarios due to axial gap degradation are created to comprehend the effect of flow leakage in the fluid volume. The axial gap faults are generated by increasing the gap between the gears and the gear casing (side leakage), called A1, A2, and A3. These scenarios compared with a healthy scenario in the time domain and the frequency domain are displayed in Figure 3.15. Similar to radial gap fault scenarios, the discharge flow rate decreases with the incremental increase of axial gap size. With respect to the healthy scenario, the mean flow rate of A1, A2 and A3 is decreased by 2%, 4% and 6.3%, respectively.

Fault scenarios due to speed variations

The fault scenarios due to speed variations are generated by increasing the rotational speed (in RPM) of the gear pump to comprehend the flow behaviour. The speed variations, S1, S2 and S3 are created to validate that for positive displacement pumps, as the



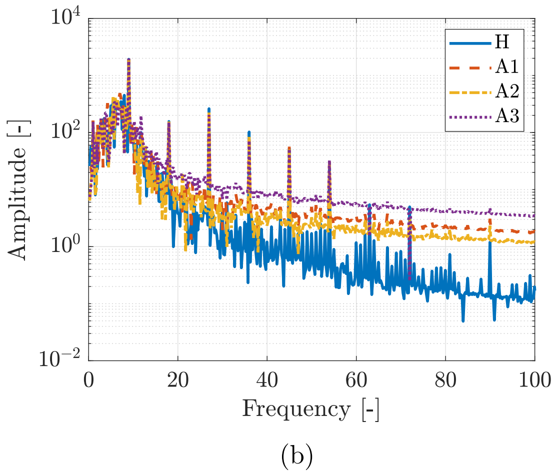
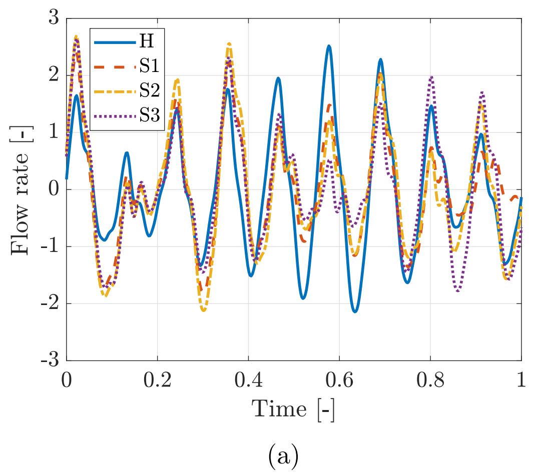


FIGURE 3.15: Discharge flow rate with respect to the time (a) and the frequency (b) for fault scenarios due to axial gap degradation.

rotational speed increases, the discharge flow rate increases in the linear trend, which is proven in [137] for an external gear pump. The discharge flow rate of these scenarios compared against the healthy scenario in the time domain and the frequency domain is shown in Figure 3.16. In respect to the healthy scenario, the mean flow rate of S1, S2 and S3 is increased by 2%, 4.5% and 6.3%, respectively. This aspect is presented in Figure 3.17.



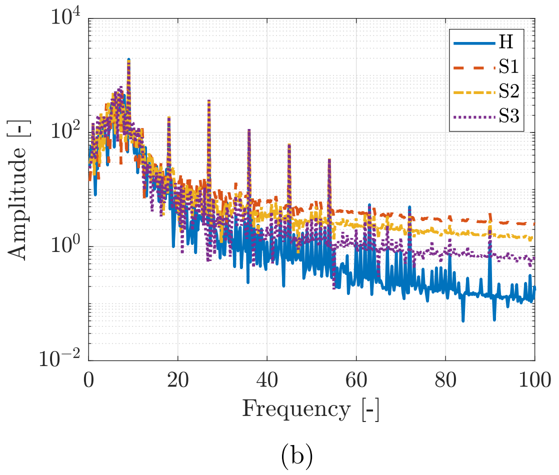


FIGURE 3.16: The discharge flow rate with respect to the time (a) and the frequency (b) for fault scenarios due to speed variations.

Fault scenarios due to viscosity variations

The fault scenarios due to viscosity variations are generated to understand the pump performance in the change of viscosity. Gear pumps are best suitable for high viscosity applications. When the fluid viscosity decreases, slip increases. The slip is when the

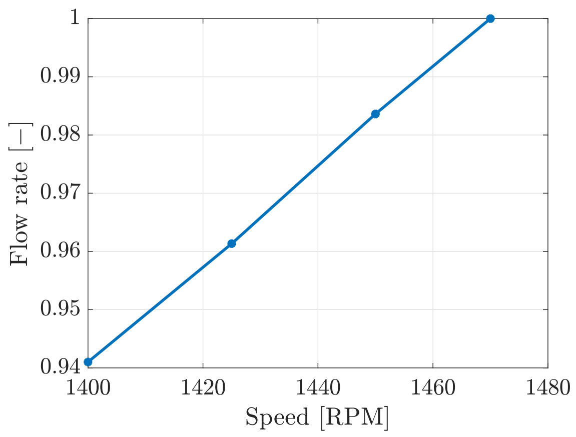


FIGURE 3.17: Average flow rate with respect the increase in the rotational speed.

pump recirculates from the outlet to the inlet side as it escapes through machined clearances while the pump tries to progress forward. The amount of slip is determined by the fluid's viscosity and discharge pressure. A thin fluid with a low viscosity can squeeze through the clearances more easily than the thicker fluids. High discharge pressure can force thinner fluids back through the pump, also causing more slip. Therefore, because thin fluids slip, the efficiency of the pump decreases as less product is progressed with the forward movement, hence the lower discharge flow rate.

The viscosity variations, V1, V2 and V3 are created by increasing the fluid viscosity. Figure 3.19 shows the discharge flow rate of these scenarios compared against the healthy scenario in the time domain and the frequency domain. The mean flow rate of V1, V2 and V3 is increased by 0.4%, 1% and 1.5%, respectively with respect to the healthy scenario.

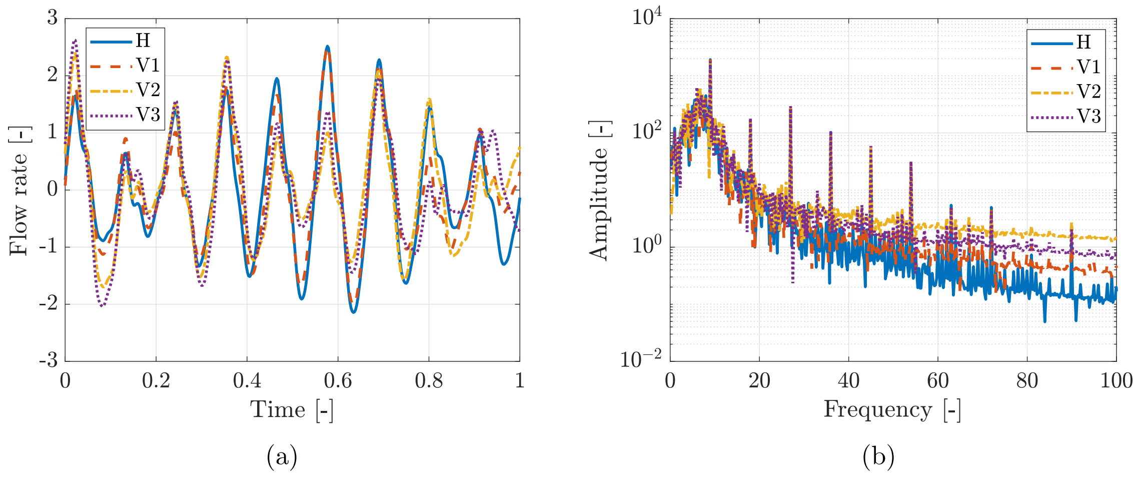


FIGURE 3.18: The discharge flow rate with respect to the time (a) and frequency (b) for the fault scenarios due to viscosity variations.

Fault scenarios due to temperature variations

The fault scenarios due to temperature variations are generated to comprehend the changes in the gear pump's performance with respect to temperature variations. An increase in temperature heats up the fluid transported through the pump, which reduces the fluid viscosity. As discussed above, an increase in viscosity increases the discharge flow rate. So, due to an increase in temperature, the viscosity decreases, and that implies, the discharge flow rate decreases.

The temperature variations, T1, T2 and T3 are created by increasing the constant temperature of the pump gradually. Figure 3.19 shows the discharge flow rate of these scenarios compared against the healthy scenario in the time domain and the frequency domain. With respect to the healthy scenario, the mean flow rate of T1, T2 and T3 is decreased by 0.16%, 0.25% and 0.33%, respectively.

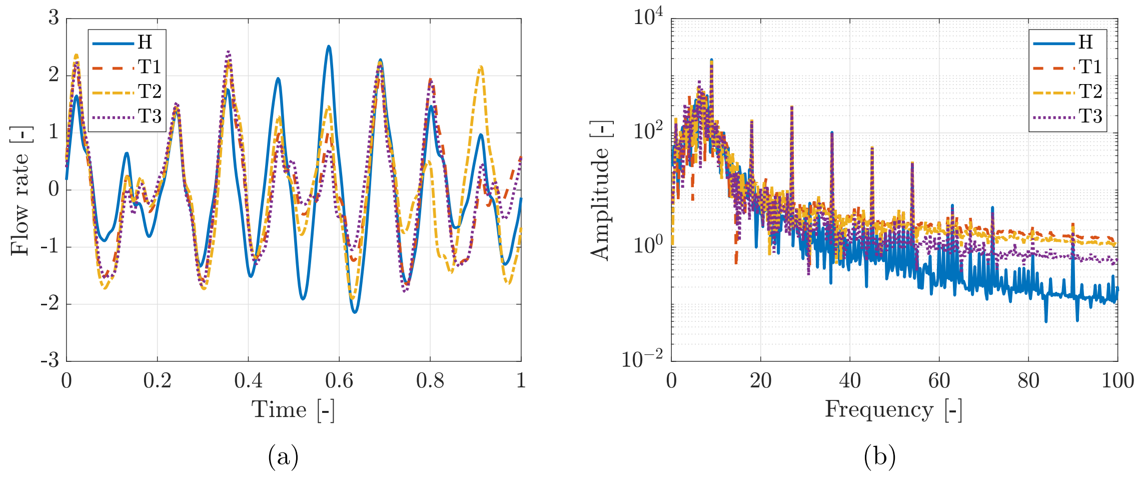


FIGURE 3.19: The discharge flow rate with respect to the time (a) and the frequency (b) for fault scenarios due to temperature variations.

Fault scenarios due to clogging variations

The fault scenarios due to clogging or blocking are implemented to validate the sudden disturbances due to any blockages in the pump. For this reason, various parts of the pump, the outlet (C1), the inlet (C2) and the top shafts (C3) are partially blocked. The CFD simulation is performed, and the discharge flow rate can be measured. Figure 3.20 shows the discharge flow rate with respect to time and frequency for fault scenarios due to clogging. Blocking a part of the outlet (C1) and part of the inlet (C2), the discharge flow rate is decreased by 4% and 0.5%, however, when blocking the top shafts (C3), the discharge flow rate is increased by 0.19%.

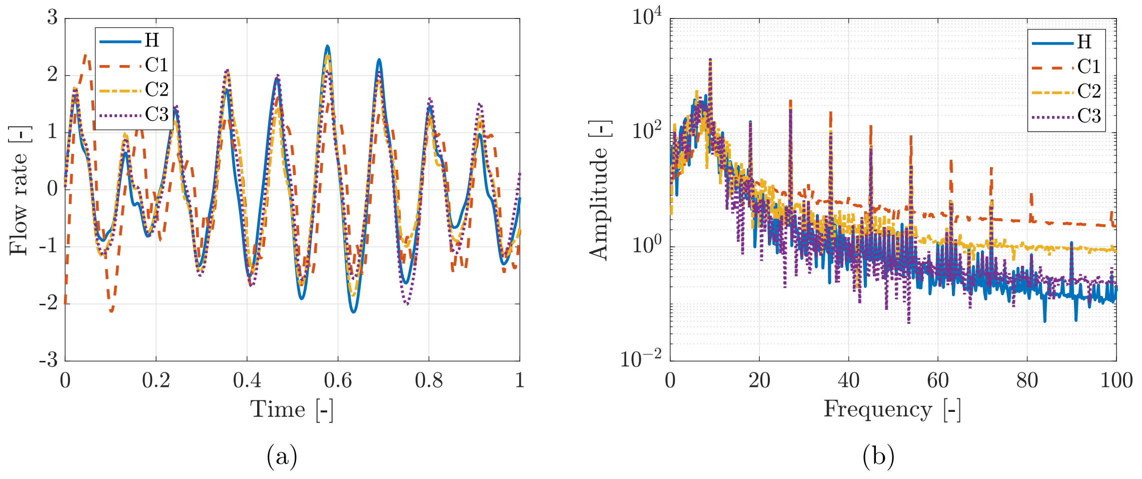


FIGURE 3.20: The discharge flow rate with respect to the time (a) and the frequency (b) for fault scenarios due to clogging or blocking.

3.7.1 Combination of fault scenarios

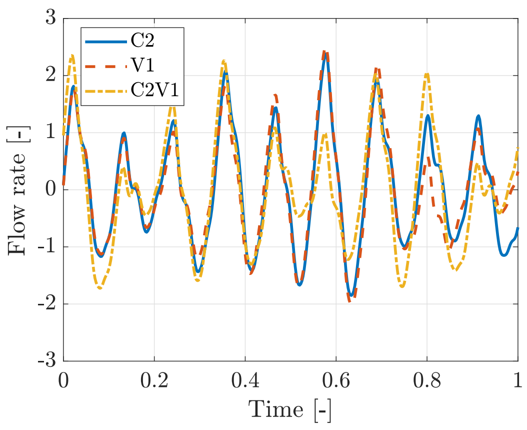
A combination of fault scenarios is implemented, by considering that, in an equipment component, any type of fault can occur, including a combination of faults. Therefore, based on experience from an industrial partner, combining fault scenarios due to clogging (C1, C2, C3) and viscosity variations (V1, V2, V3) together create a combination of fault scenarios (C1V1, C1V2, C1V3, C2V1, C2V2, C2V3, C3V1, C3V2, C3V3). For all these combinations, the CFD simulation is performed, and the discharge flow rate is measured.

Figures 3.21 and 3.22 show the discharge flow rate of the fault scenarios due to combination of clogging and viscosity in time domain (a) and frequency domain (b). In Figure 3.22, the behaviour of C3V3 is very closely related to V3 because, compared to V3, the fault implemented in C3 does not affect the performance of the gear pump.

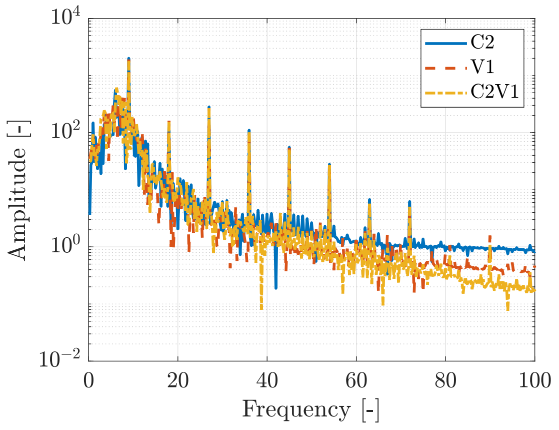
All the fault scenarios and combination of fault scenarios discussed in the current section will be used to perform fault diagnosis in Chapter 6.

3.8 Chapter conclusion

The present chapter has presented the numerical examples related to CFD simulations for an external gear pump. Firstly, the geometry of the external gear pump was introduced. Following that the computational implementation using the CFD commercial

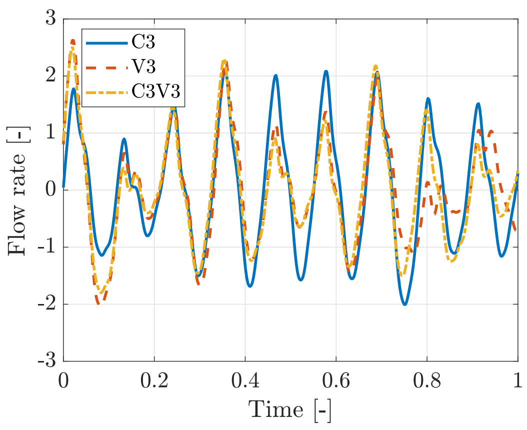


(a)

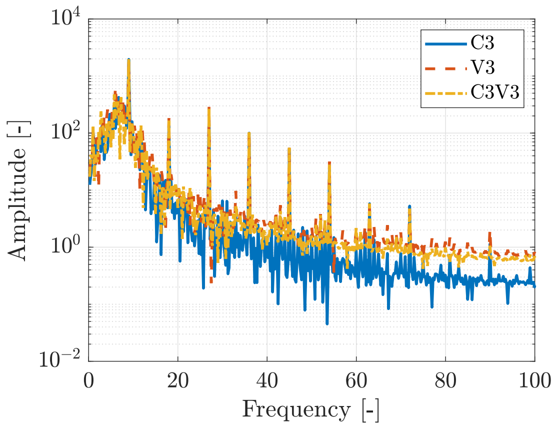


(b)

FIGURE 3.21: The discharge flow rate with respect to the time and the frequency. Combination of clogging and viscosity fault scenarios C2V1 in time domain (a) and frequency domain (b).



(a)



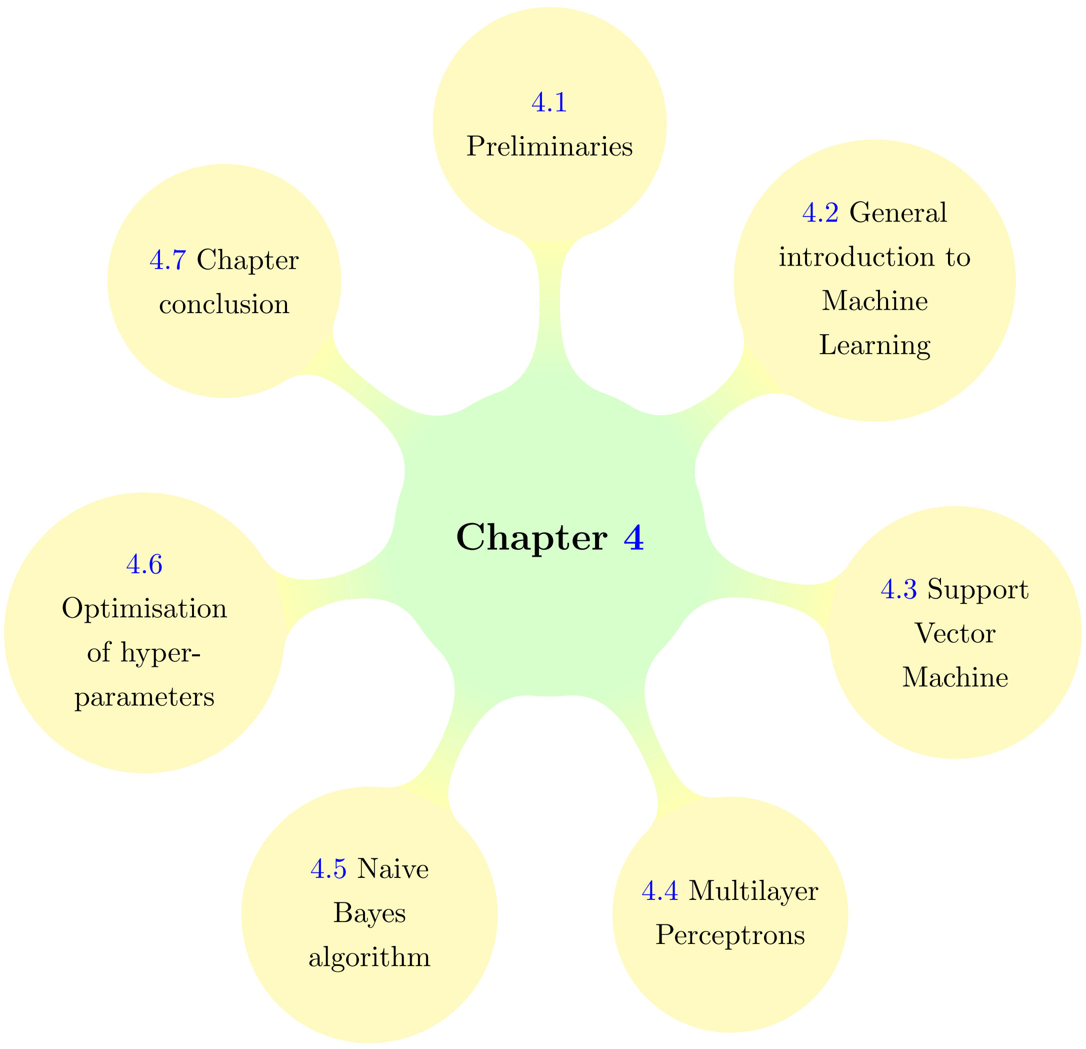
(b)

FIGURE 3.22: The discharge flow rate with respect to time and frequency. Combination of clogging and viscosity fault scenarios C3V3 in the time domain (a) and the frequency domain (b).

software PumpLinx has been presented. The mesh and time sensitivity analysis of an external gear pump has been discussed by showing the spatial and time sensitivity of the CFD model. The CFD model is validated by the experimental results by showing their strong similarities. The gear pump working in a healthy condition has been shown and established as a baseline criterion. Finally, the simulation results of gear pumps with series fault scenarios have been discussed. With the current chapter, the high-fidelity in-silico data have been generated through the CFD model. This is considered as primary data for ML.

Chapter 4

MACHINE LEARNING FOR DIAGNOSIS AND PROGNOSIS



4.1 Preliminaries
4.2 General introduction to Machine Learning
4.7 Chapter conclusion
Chapter 4
4.6 Optimisation of hyper-parameters
4.3 Support Vector Machine
4.5 Naive Bayes algorithm
4.4 Multilayer Perceptrons

Figure 4.1: Structure of Chapter 4.

4.1 Preliminaries

This chapter presents the Machine Learning (ML) algorithms for diagnosis and prognosis employed in the present thesis. Firstly, the general terminology and the types of ML algorithms are introduced in Section 4.2. The derivation of three different supervised ML algorithms, namely Support Vector Machine (SVM), Multilayer Perceptron (MLP) and Naive Bayes (NB), are introduced in Sections 4.3, 4.4 and 4.5, respectively. Finally, the optimisation process of the hyper-parameters for the various ML algorithms is studied in Section 4.6. The layout of the present chapter is summarised in Figure 4.1.

4.2 General introduction to Machine Learning

Machine Learning is a fundamental subdivision of Artificial Intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Figure 4.2 shows that ML is a subfield of AI [168].

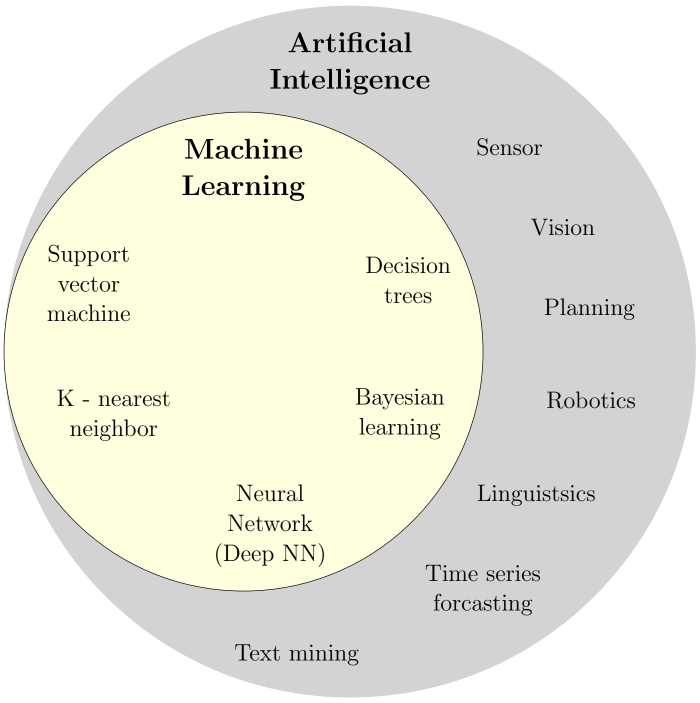


FIGURE 4.2: Artificial intelligence and Machine Learning.

ML focuses on the development of computer programs that can access data and perform tasks automatically through prediction and detection. Figure 4.3 summarises the concept of ML, whereby a machine uses data to learn and understand the problem automatically and make evaluations according to its experiences.

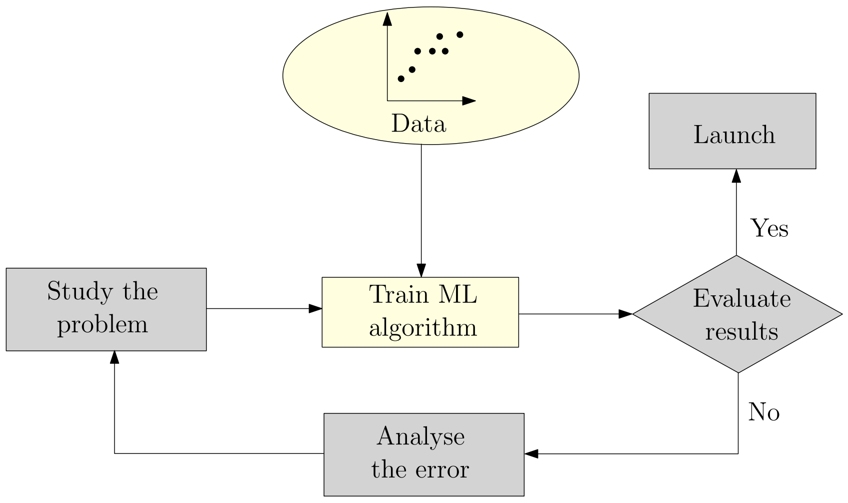


FIGURE 4.3: General working principle of ML.

ML is mainly divided into unsupervised learning and supervised learning. The goal of unsupervised learning is to learn from an input $S = \{\boldsymbol{x}_j\}_{j=1}^{n}$ and find possible patterns in the dataset. Two main methods used in unsupervised learning are clustering analysis and principal component analysis [169]. The goal of supervised learning algorithms is to learn to map from the input $\boldsymbol{x}$ to the output y for a labelled set of pairs $S = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{n}$, where S is called training sample dataset and n is the number of training samples. Figure 4.4 shows the supervised learning approach for a given problem. The input $\boldsymbol{x}_j$ can be a vector, for example, the height and weight of a person, which are called features or attributes. Moreover, $\boldsymbol{x}_j$ can be a complex object, such as an image, a time-series, a sentence, an email message, etc. Similarly, output or response variable y_j can be categorical values from some finite set such as male or female, types of flowers, etc., or y_j can be a real-valued scalar such as income level, temperature, etc. When y_j is real-valued, then the problem is defined as regression, and when y_j is categorical, then the problem is defined as classification. To analyse the results of classification and regression, the accuracy metric and the error metric (Mean Squared Error (MSE), mean absolute error, R2 score) are utilised. The accuracy and MSE are defined as follows:

$$\text{Accuracy} = \frac{1}{n}\sum_{j=1}^{n}\left(\begin{cases} 1 & \text{if} \quad \hat{y}_j = y_j \\ 0 & \text{otherwise} \end{cases}\right), \tag{4.1}$$

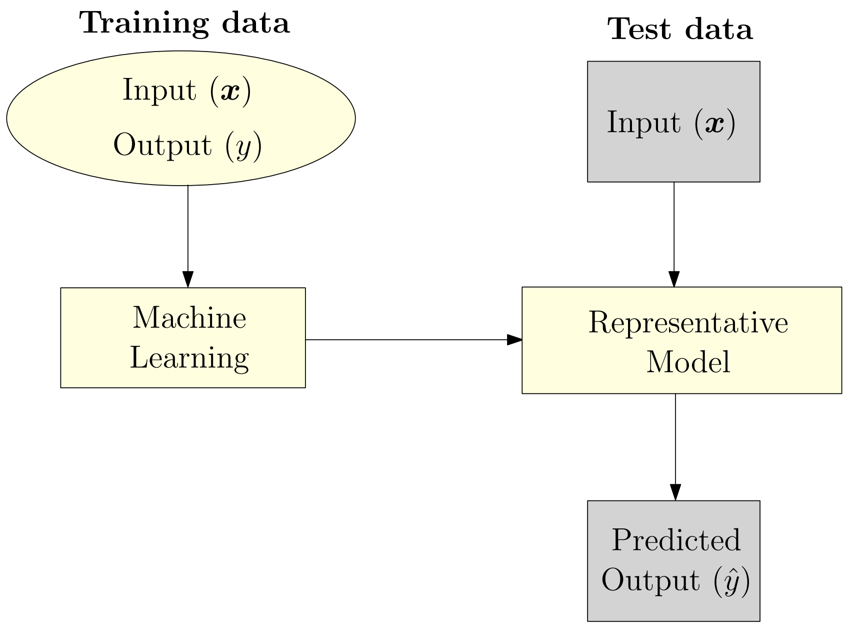


FIGURE 4.4: Supervised learning method.

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n} \left(\hat{y}_j - y_j\right)^2 , \tag{4.2}$$

where $\hat{y}_j$ are predicted values. Figures 4.5a and 4.5b demonstrate the representation of classification and regression, respectively. ML supervised learning algorithms, such as

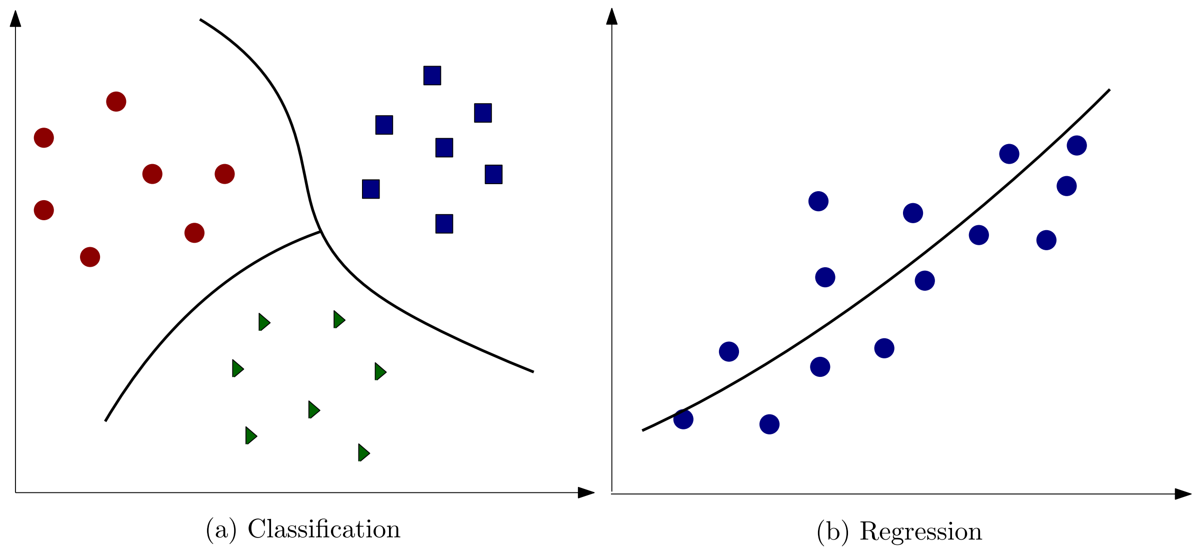


FIGURE 4.5: Representation of classification (a) and regression (b).

SVM, MLP and NB, are considered exclusively in this thesis and they are presented in the following sections.

4.3 Support Vector Machine

This section describes one of the most effective classification algorithms called Support Vector Machine (SVM). Firstly, the linear classification problem and the algorithm for linearly separable case are introduced. Subsequently, a more generic version of the algorithm for non-separable data is presented followed by the formulation for margin theory and the concept of kernel spaces. Finally, SVM regression is briefly discussed.

4.3.1 Linear classification

The simplest form of pattern classification is that of binary classification. Consider an input space $\mathcal{X} \subset \mathbb{R}^m$ with $m \geq 1$, and an output (target) space $\mathcal{Y} = \{-1, 1\}$. Let $f : \mathcal{X} \to \mathcal{Y}$ be an output function. The training sample dataset S drawn from $\mathcal{X}$ is

$$S = ((\boldsymbol{x}_1, y_1), ..., (\boldsymbol{x}_n, y_n)) \in (\mathcal{X} \times \mathcal{Y})^n, \tag{4.3}$$

with $y_j = f(\boldsymbol{x}_j)$ and n the number of samples within the sample dataset S. Consider the hypothesis set $\mathcal{H}$ of a binary classifier h mapping $\mathcal{X}$ to $\mathcal{Y}$ is defined as

$$\mathcal{H} = \{\boldsymbol{x} \mapsto \text{sign}\,(\boldsymbol{w} \cdot \boldsymbol{x} + b) : \boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}\}. \tag{4.4}$$

In above expression, $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ is the general equation of a hyperplane in $\mathbb{R}^m$, where $\boldsymbol{w}$ is a weight vector normal to the hyperplane and b is a scalar. The hypothesis $\mathcal{H}$ in Equation (4.4) labels all the data points either positively (on one side of the hyperplane) or negatively (on the other side of the hyperplane).

4.3.2 SVMs - separable case

Assume that a given training sample dataset S in Equation (4.3) is linearly separable, then there exists a hyperplane that perfectly separates the training samples into two classes of positively and negatively labeled sample data points as shown in Figure 4.6. This is equivalent to the existence of $(\boldsymbol{w}, b) \in (\mathbb{R}^m - \{0\}^m \times \mathbb{R})$ such that

$$y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \geq 0, \quad \forall j = 1, \ldots, n, \tag{4.5}$$

where the parenthesis indicates a multiplication. In Figure 4.6, there can be infinitely many separating hyperplanes for the given problem. The SVM algorithm selects the optimal hyperplane based on the geometric margin, which is stated in Definition 1.

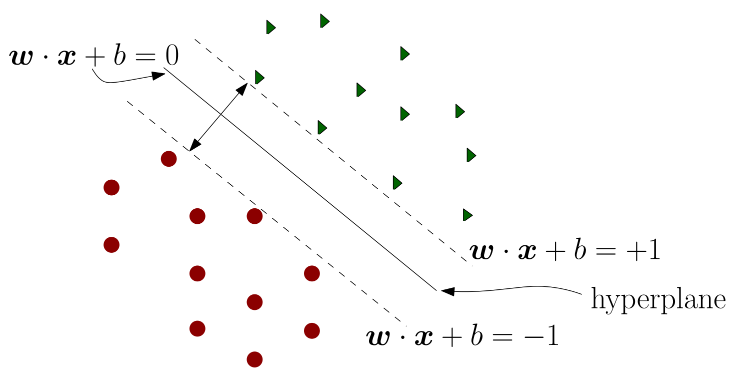

$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = +1$
hyperplane
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$

FIGURE 4.6: The hyperplane is represented by $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ and the marginal hyperplanes are represented by $\boldsymbol{w} \cdot \boldsymbol{x} + b = \pm 1$ (dashed lines).

Definition 1: **(Geometric margin)** The geometric margin $M_h(\boldsymbol{x})$ of a linear classifier $h : \boldsymbol{x} \mapsto \text{sign } (\boldsymbol{w} \cdot \boldsymbol{x} + b)$ at a point $\boldsymbol{x}$ is its Euclidean distance to the hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

$$M_h(\boldsymbol{x}) = \frac{|\boldsymbol{w} \cdot \boldsymbol{x} + b|}{||\boldsymbol{w}||}, \tag{4.6}$$

where geometric margin M of a linear classifier h for a sample dataset S is the minimum geometric margin over the points in the sample dataset,

$$\mathcal{M}_h(S) = \min_{j=1,\ldots,n} M_h(\boldsymbol{x}_j), \tag{4.7}$$

which represents the distance of the hyperplane defining h to the closest sample points [170].

Using the definition of the geometric margin, the maximum-margin M of a separating hyperplane is defined as,

$$M = \max_{\boldsymbol{w},b} \min_{j=1,\ldots,n} \frac{y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b)}{||\boldsymbol{w}||}. \tag{4.8}$$

To maximise the pair $(\boldsymbol{w}, b)$, $y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b)$ must be positive for all $j = 1, \ldots, n$, and invariant under multiplication by a positive scalar. Thus the pairs $(\boldsymbol{w}, b)$ are scaled such that $\min_{j=1,\ldots,n} y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b)||\boldsymbol{w}||= 1$. Then Equation (4.8) becomes

$$M = \max_{\boldsymbol{w},b\colon \forall j=1,\ldots,n,\ y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j+b)\geq 1} \frac{1}{||\boldsymbol{w}||}. \tag{4.9}$$

Figure 4.6 illustrates the maximum-margin hyperplane returned by the SVM solution for the separable case. It also shows the marginal hyperplanes, which are the hyperplanes parallel to the separating hyperplane and passing through the closest points on the negative and positive sides. Since maximising $1/||\boldsymbol{w}||$ is equivalent to minimising $\frac{1}{2}||\boldsymbol{w}||^2$, and the solution of SVM for the separable case can be achieved by the following convex optimisation problem

$$\begin{aligned} &\underset{\boldsymbol{w},b}{\text{minimise}} \quad \frac{1}{2}||\boldsymbol{w}||^2, \\ &\text{subject to} \quad y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j + b) \geq 1, \quad \forall j = 1,\ldots,n. \end{aligned} \tag{4.10}$$

The objective function $F : \boldsymbol{w} \mapsto \frac{1}{2}||\boldsymbol{w}||^2$ is infinitely differentiable, its gradient is $\nabla F(\boldsymbol{w}) = \boldsymbol{w}$, and its Hessian is the identity matrix $\nabla^2 F(\boldsymbol{w}) = \boldsymbol{I}$, whose eigenvalues are strictly positive. Hence, the Hessian is non negative definite matrix ($\nabla^2 F(\boldsymbol{w}) \succ 0$) and F is strictly convex. The constraints are all defined by affine functions $g_j : (\boldsymbol{w}, b) \mapsto 1 - y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j + b)$ and thus qualified[2]. Thus, in view of the results known for convex optimisation, the optimisation problem (4.10) admits a unique solution, an important property that does not hold for other learning algorithms [170].

To solve the above optimisation problem, the quadratic function with linear constraints needs to be optimised. The solution involves constructing a dual optimisation problem where a Lagrange's multiplier $\boldsymbol{\alpha}$ is introduced. Let us define the Lagrange multipliers $\alpha_j \geq 0, \ j = 1,\ldots,n$, associated to the n constraints and referred to by $\boldsymbol{\alpha}$, then the Lagrangian can be defined as

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{j=1}^{n} \alpha_j[y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j + b) - 1] \tag{4.11}$$

$\forall\ \boldsymbol{w} \in \mathbb{R}^m, b \in \mathbb{R}$ and $\boldsymbol{\alpha} \in \mathbb{R}^n_+$.

Theorem 4.1. ***(Karush-Kuhn-Tucker's theorem)*** *Assume that $f, g_j : \mathcal{X} \to \mathbb{R}, \forall j \in \{1, 2, \ldots, n\}$ are convex and differentiable and that the constraints are qualified. Then $\bar{\boldsymbol{x}}$*

[2] Constraint qualifications are assumptions on the algebraic description of the feasible set of an optimisation problem that ensure that the KKT conditions hold at any local minimum[171].

is a solution of constrained program iff $\exists\ \bar{\boldsymbol{\alpha}} \geq 0$ *such that*

$$\begin{aligned} \nabla_{\bar{\boldsymbol{x}}}\mathcal{L}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\alpha}}) =& \nabla_{\bar{\boldsymbol{x}}} f(\bar{\boldsymbol{x}}) + \bar{\boldsymbol{\alpha}} \cdot \nabla_{\bar{\boldsymbol{x}}} g_j(\bar{\boldsymbol{x}}) = 0 \quad (4.12) \\ \nabla_{\boldsymbol{\alpha}}\mathcal{L}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{\alpha}}) =& g_j(\bar{\boldsymbol{x}}) \leq 0 \quad (4.13) \\ \sum_{j=1}^{n} \bar{\boldsymbol{\alpha}}_j \cdot g_j(\bar{\boldsymbol{x}}) =& 0 \quad (4.14) \end{aligned}$$

Equations (4.12) *to* (4.14) *are known as the KKT conditions.*

Support Vectors: The constraints are affine and thus qualified. The objective function as well as the affine constraints are convex and differentiable. Thus, the hypothesis of Theorem 4.1 holds and the Karush-Kuhn-Tucker (KKT) conditions apply at the optimum. These condition can be used to analyse the algorithm and demonstrate crucial properties and subsequently the derive dual Lagrangian problem associated to SVMs.

The KKT conditions are obtained from Equation 4.11 by computing the gradient of the Lagrangian with respect to the variables $\boldsymbol{w}$ and b to zero, and the conditions can be written as:

$$\begin{aligned} \nabla_{\boldsymbol{w}}\mathcal{L} = \boldsymbol{w} - \sum_{j=1}^{n} \alpha_j y_j \boldsymbol{x}_j = 0 \quad &\implies \quad \boldsymbol{w} = \sum_{j=1}^{n} \alpha_j y_j \boldsymbol{x}_j, \quad (4.15) \\ \nabla_b \mathcal{L} = -\sum_{j=1}^{n} \alpha_j y_j = 0 \quad &\implies \quad \sum_{j=1}^{n} \alpha_j y_j = 0, \quad (4.16) \\ \alpha_j[y_j(\boldsymbol{w}\cdot\boldsymbol{x} + b) - 1] = 0 \quad &\implies \quad \alpha_j = 0 \vee y_j(\boldsymbol{w}\cdot\boldsymbol{x}_j + b) = 1, \forall j, \quad (4.17) \end{aligned}$$

where, $\vee$ represents logical disjunction (i.e., OR operator). According to Equation (4.15), the weight vector $\boldsymbol{w}$ is a linear combination of training samples $\boldsymbol{x}_j$ provided $\alpha_j \neq 0$. A vector $\boldsymbol{x}_j$ appears in that expansion iff $\alpha_j \neq 0$. Such vectors are called *support vectors*. By Equation (4.17), if $\alpha_j \neq 0$, then $y_j(\boldsymbol{w}\cdot\boldsymbol{x} + b) = 1$. Thus, support vectors lie on the marginal hyperplanes $\boldsymbol{w}\cdot\boldsymbol{x} + b = \pm 1$. Figure 4.7 shows the support vectors which lie on the marginal hyperplanes. Support vectors fully define the maximum-margin hyperplane or SVM solution. By construction, vectors which are not lying on the marginal hyperplanes do not affect the definition of these hyperplanes. In their absence, the solution to the SVM problem remains unchanged. Due to the difficulty in solving the primal optimisation problem (4.10), the problem is converted to a *dual optimisation problem*. Substituting the conditions (4.15) - (4.16) into the Lagrangian

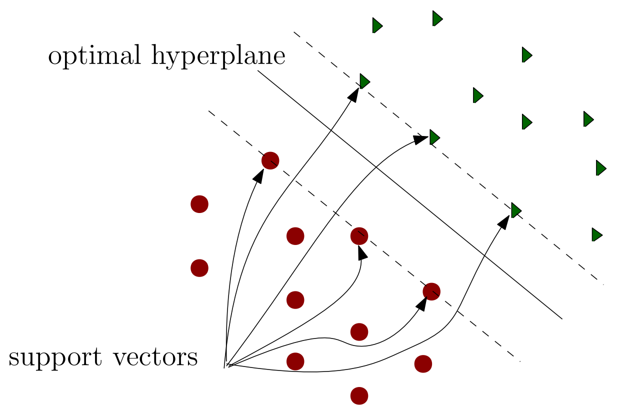


FIGURE 4.7: Representation of support vectors, which are lying on the marginal hyperplanes (dashed lines).

function, leads to the following dual optimisation problem:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{miximise}} \quad & \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{j,k=1}^{n} \alpha_j \alpha_k y_j y_k (\boldsymbol{x}_j \cdot \boldsymbol{x}_k), \\ \text{subject to} \quad & \alpha_j \geq 0 \wedge \sum_{j=1}^{n} \alpha_j y_j = 0, \forall j = 1, \ldots, n. \end{aligned} \tag{4.18}$$

To solve the dual optimisation problem, a quadratic problem with linear constraints needs to be optimised. The objective function G in Equation (4.18) is infinitely differentiable. Its Hessian is given by $\nabla^2 G \preceq 0$, and thus G is a concave function. Since the constraints are affine and convex, the maximization problem (4.18) is equivalent to the convex optimisation problem (4.10). Furthermore, as the constraints are affine, they are qualified and strong duality holds. Thus the primal and dual problems are equivalent. The solution of a dual problem (4.18) can be used directly to determine the hypothesis by the SVMs, using equation (4.15),

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) = \text{sign}\left(\sum_{j=1}^{n} \alpha_j y_j (\boldsymbol{x}_j \cdot \boldsymbol{x}) + b \right). \tag{4.19}$$

To compute the value of b, using the fact that the support vectors lie on the marginal hyperplane, for any support vector $\boldsymbol{x}_j$, $\boldsymbol{w} \cdot \boldsymbol{x}_j + b = y_j$ and thus b can be obtained as

$$b = y_j - \sum_{k=1}^{n} \alpha_k y_k (\boldsymbol{x}_k \cdot \boldsymbol{x}_j). \tag{4.20}$$

The dual optimisation problem (4.18) and the expressions (4.19) and (4.20) reveal that the hypothesis solution depends only on inner products between the vectors and not directly on the vectors themselves.

4.3.3 SVMs - non-separable case

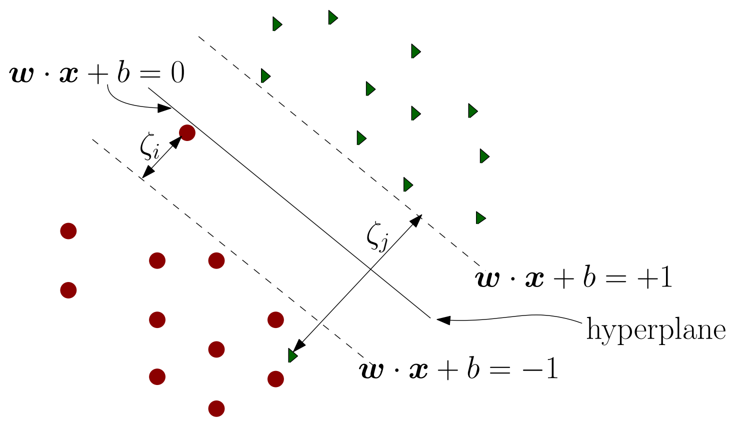
In practical situations, the training dataset are not linearly separable, which implies for any hyperplane $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$, $\exists\, \boldsymbol{x}_j \in S$ such that

$$y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \ngeq 1, \tag{4.21}$$

then the constraints imposed in Section 4.3.2 for linearly separable case cannot hold. However a relaxed version of those constraint holds with the help of slack variables ζ_j. $\forall j = 1, \ldots, n$, $\exists\, \zeta_j \geq 0$ such that

$$y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \geq 1 - \zeta_j. \tag{4.22}$$

The slack variable measures the distance by which the $\boldsymbol{x}_j$ violates the inequality, which is shown in Figure 4.8.



$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
ζ_i
ζ_j
$\boldsymbol{w} \cdot \boldsymbol{x} + b = +1$
hyperplane
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$

FIGURE 4.8: A separating hyperplane with one data sample incorrectly classified and other one classified correctly with margin less than 1.

Similar to the separable case, the optimisation problem which is defining the SVMs in the non-separable case is defined as

$$\begin{aligned} &\underset{\boldsymbol{w},b,\boldsymbol{\zeta}}{\text{minimise}} \quad \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{j=1}^{n} \zeta_j^p, \\ &\text{subject to} \quad y_j(\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \geq 1 - \zeta_j \wedge \zeta_j \geq 0,\ \forall j = 1, \ldots, n, \end{aligned} \tag{4.23}$$

where the regularisation parameter $C \geq 0$ determines minimisation of $||\boldsymbol{w}||^2$ and the minimisation of the slack penalty $\sum_{j=1}^{n} \zeta_j^p$, for some $p \geq 1$. Similar to the separable case, Equation (4.23) is convex optimisation problem since the constraints are affine and thus convex and since the objective function is convex for any $p \geq 1$. The solution of this problem involves constructing a dual optimisation problem, where a Lagrange's multiplier $\boldsymbol{\alpha}$ is associated. The Lagrangian equation is constructed, along with ζ_j. The KKT conditions are obtained by setting the gradient of the Lagrangian with respect to variables $\boldsymbol{w}$, b and ζ_j to a value of zero. For the convenience of solving the optimisation problem, Equation (4.23) is converted to a dual optimisation problem, and it can be written as

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{miximise}} \quad \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{j,k=1}^{n} \alpha_j \alpha_k y_j y_k (\boldsymbol{x}_j \cdot \boldsymbol{x}_k), \\ &\text{subject to} \quad 0 \leq \alpha_j \leq C \wedge \sum_{j=1}^{n} \alpha_j y_j = 0, \ \forall j = 1, \ldots, n, \end{aligned} \tag{4.24}$$

the constraint $\alpha_j \geq 0$ has been changed to $0 \leq \alpha_j \leq C$. A smaller C emphasizes the importance of ζ_j. A similar approach used to solve the dual problem in the separable case (Equation (4.18)) can be applied to the dual problem in the non-separable case (Equation (4.24)). In particular, the objective function is concave and infinitely differentiable and Equation (4.24) is equivalent to a convex problem (Equation (4.23)). The hypothesis returned by the SVMs, using KKT condition is,

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b) = \text{sign}(\sum_{j=1}^{n} \alpha_j y_j (\boldsymbol{x}_j \cdot \boldsymbol{x}) + b). \tag{4.25}$$

To compute b, using the fact that the support vectors lies on the marginal hyperplane, for any vector $\boldsymbol{x}_j$, with $0 \leq \alpha_j \leq C$. For support vectors $\boldsymbol{w} \cdot \boldsymbol{x}_j + b = y_j$, b can be obtained as

$$b = y_j - \sum_{k=1}^{n} \alpha_k y_k (\boldsymbol{x}_k \cdot \boldsymbol{x}_j). \tag{4.26}$$

The dual optimisation problem (4.24) and the expressions (4.25) and (4.26) reveal an important property of SVMs. The hypothesis solution depends only on inner products between vectors and not directly on the vectors themselves. These topics are discussed in detail in [170].

4.3.4 Kernel methods

Often in practical situations, the collected data are far from linear and separable. Kernels are used to non-linearly map the input space to high-dimensional space, where the linear separation is possible; this is illustrated in Figure 4.9. The kernel function is given below in Definition 2.

Definition 2: **(Kernels)** A function $\mathcal{K} = \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called kernel over $\mathcal{X}$.

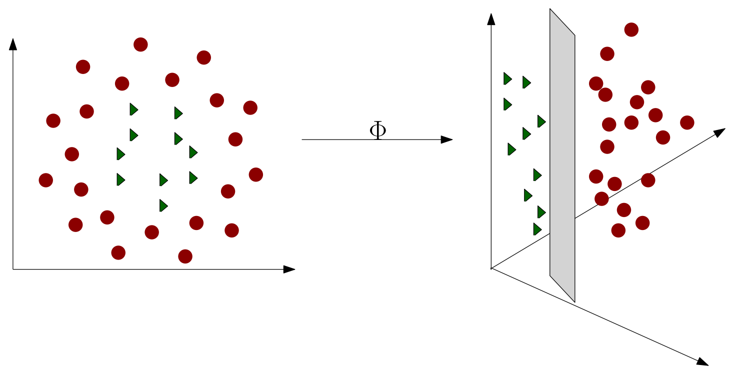


Figure 4.9: Example of a non-linear mapping from 2D to 3D, where the data sample becomes linearly separable.

The idea is to define a kernel $\mathcal{K}$ such that any two points $x_1, x_2 \in \mathcal{X}$, $\mathcal{K}(x_1, x_2)$ be equal to an inner product $(\langle \cdot, \cdot \rangle)$ of vectors $\Phi(x_1)$ and $\Phi(x_2)$

$$\forall x_1, x_2 \in \mathcal{X},\ \mathcal{K}(x_1, x_2) = \langle \Phi(x_1), \Phi(x_2) \rangle, \tag{4.27}$$

for some mapping $\Phi : \mathcal{X} \rightarrow \mathbb{H}$ to a Hilbert space $\mathbb{H}$.

Since an inner product is a measure of the similarity of two vectors, $\mathcal{K}$ is often interpreted as a similarity measure between elements of the input space $\mathcal{X}$. The important advantage of kernel $\mathcal{K}$ is its efficiency (in several cases, computation can be achieved in $O(N)$). Another advantage of kernel function is flexibility (the existence of non-linear mapping is guaranteed, i.e. it satisfies Mercer's condition). Mercer's condition is important to guarantee the convexity of the SVM optimisation problem, thereby ensuring convergence to a global minimum [170].

Definition 3: **(Positive definite symmetric kernels)** A kernel $\mathcal{K} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is said to be positive definite symmetric if for any $\{x_1, ..., x_n\} \subset \mathcal{X}$, the matrix $\boldsymbol{\mathcal{K}} = [\mathcal{K}(x_j, x_k)]_{jk} \in \mathbb{R}^{n \times n}$ is symmetric positive semi-definite (SPSD).

$\boldsymbol{\mathcal{K}}$ is SPSD, if it is symmetric and the eigenvalues of $\boldsymbol{\mathcal{K}}$ are non-negative or for any column vector $\boldsymbol{c} = (c_1, ..., c_n)^\intercal \in \mathbb{R}^{n \times 1}$,

$$\mathbf{c}^T \boldsymbol{\mathcal{K}} \mathbf{c} = \sum_{j,k=1}^{n} c_j c_k \mathcal{K}(x_j, x_k). \tag{4.28}$$

For a sample $S = (x_1, ...x_n), \boldsymbol{\mathcal{K}} = [\mathcal{K}(x_j, x_k)]_{jk} \in \mathbb{R}^{n \times n}$ is called kernel matrix or a Gram matrix associated to $\mathcal{K}$ and the sample S. Let us define some standard examples of the positive definite symmetric kernels, which are commonly used:

☐ **Linear kernel**

A linear kernel $\mathcal{K}$ defined over $\mathbb{R}^m$ by

$$\forall \boldsymbol{x}_j, \boldsymbol{x}_k \in \mathbb{R}^m, \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_k) = \boldsymbol{x}_j \cdot \boldsymbol{x}_k. \tag{4.29}$$

☐ **Polynomial kernel**

For any constant $c_1 > 0$, a polynomial kernel of degree $d \in \mathbb{N}$ is the kernel $\mathcal{K}$ defined over $\mathbb{R}^m$ by,

$$\forall \boldsymbol{x}_j, \boldsymbol{x}_k \in \mathbb{R}^m, \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_k) = (\boldsymbol{x}_j \cdot \boldsymbol{x}_k + c_1)^d. \tag{4.30}$$

☐ **Gaussian kernels or Radial basis function**

For any constant $c_1 > 0$, a Gaussian kernels is the kernel $\mathcal{K}$ defined over $\mathbb{R}^m$ by

$$\forall \boldsymbol{x}_j, \boldsymbol{x}_k \in \mathbb{R}^m, \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_k) = \exp\left(\frac{-||\boldsymbol{x}_k - \boldsymbol{x}_j||^2}{2c_1^2}\right). \tag{4.31}$$

Gaussian kernel is one of the most frequently used kernels in applications.

☐ **Sigmoid kernels**

For any real constant $c_1, c_2 \geq 0$, a sigmoid kernels is the kernel $\mathcal{K}$ defined over $\mathbb{R}^m$ by

$$\forall \boldsymbol{x}_j, \boldsymbol{x}_k \in \mathbb{R}^m, \mathcal{K}(\boldsymbol{x}_j, \boldsymbol{x}_k) = \tan(c_1(\boldsymbol{x}_j \cdot \boldsymbol{x}_k) + c_2). \tag{4.32}$$

Using sigmoid kernel with SVMs is closely related to learning algorithms based on a simple neural network as in [170].

4.3.5 Support Vector Machine Regression

SVM regression is a method to estimate a function that maps from an input object to a real number on the training dataset. Similar to the SVM classification, SVM regression has the same properties of the margin maximisation and kernel for non-linear mapping with only a minor difference [172, 173].

Recall the training sample dataset in Equation (4.3). The SVM regression function maps from an input vector $\boldsymbol{x}_j$ to the output y_j of the form $\boldsymbol{w} \cdot \boldsymbol{x}_j + b$. The goal is to estimate the parameters ($\boldsymbol{w}$ and b) of the function that give the best fit of the dataset. The SVM function approximates all pairs of input and output $(\boldsymbol{x}_j, y_j)$ while maintaining the differences between estimated values and real values under *epsilon* (ϵ) precision. ϵ is called as margin of tolerance. That is, for every input $\boldsymbol{x}_j$ in S,

$$y_j - (\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \leq \epsilon. \tag{4.33}$$

Recalling Equations (4.9) and (4.10), by minimising $||\boldsymbol{w}||^2$ to maximise the margin, the training in SVM regression becomes a constrained optimisation problem.

$$\begin{aligned} &\underset{\boldsymbol{w},b}{\text{minimise}} && \frac{1}{2}||\boldsymbol{w}||^2, \\ &\text{subject to} && y_j - (\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \leq \epsilon, \quad \forall j = 1, \ldots, n. \end{aligned} \tag{4.34}$$

The solution of this problem follows the similar step of solving SVM classification problem. For the non-separable case discussed in Section 4.3.3, the soft margin SVM uses slack variable ζ_j. Then, the optimisation problem (4.23) revised as

$$\begin{aligned} &\underset{\boldsymbol{w},b,\boldsymbol{\zeta}}{\text{minimise}} && \frac{1}{2}||\boldsymbol{w}||^2 + C\sum_{j=1}^{n} \zeta_j^p, \\ &\text{subject to} && y_j - (\boldsymbol{w} \cdot \boldsymbol{x}_j + b) \leq \epsilon + \zeta_j \wedge \epsilon,\ \zeta_j \geq 0,\ \forall j = 1, \ldots, n, \end{aligned} \tag{4.35}$$

To solve the optimisation problem, similar steps are utilised as the SVM classification problem, presented earlier in this chapter.

4.4 Multilayer Perceptrons

This section describes one of the most effective ML algorithms, called Multilayer Perceptron (MLP). MLP is a class of feedforward artificial Neural Network (ANN). The study of ANNs has been inspired by the observation that biological learning systems are built of very complex networks of interconnected neurons. The average human brain consists of nearly 10^{11} neurons of various types, with each neurons connected on average to 10^4 others [174]. However, the fastest neuron switching times are in the order of 10^{-3}, which is quite slow compared to computer switching speed of 10^{-10}. Yet, humans are able to make complex decisions surprisingly quick. This observation has hypothesised that the information-processing abilities of biological neural systems must follow from highly parallel processes operating on representations that are distributed over many neurons. A motivation for ANN is to capture this kind of highly parallel computation based on distributed representations [175].

ANN is a network that is composed of artificial neurons or nodes. Figure 4.10 shows the functioning of a simple ANN architecture comprised of a single input neuron. The scalar input x is multiplied by the scalar weight w and forms wx, and the bias term b is added and sent to the summer $\sum$. The net input $z = wx + b$ goes into the activation function ϕ and produces a scalar output neuron y. The output neuron is thus calculated as

$$y = \phi(z) = \phi(wx + b). \tag{4.36}$$

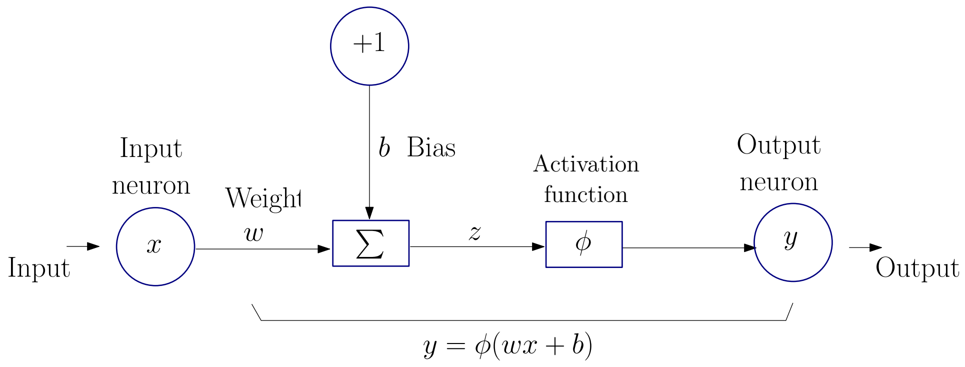


FIGURE 4.10: Single input neuron and single output neuron; z is net input function which is addition of weighted input(wx) and bias (b); ϕ is an activation function.

Before proceeding to further specifications, some necessary technical terms are introduced as follows:

- **Network architecture:** The network architecture is a collection of neurons and connections.
- **Neurons:** The neurons are the nodes of the network, and they are used to receive, send and store information.
- **Connections:** They connect one neuron to another neuron.
- **Weights:** Each network connection always has a weight value associated with it, which represents its importance. To optimise the value of the weights, the backpropagation method is utilised, which will be discussed later in this section.
- **Bias unit:** (Also referred as offset) The Bias unit is another neuron in the ANN. Also, it could be achieved by adding a constant input, usually considered 1. Bias has only outgoing signals to the neurons of the next layer, and it is not influenced by any neurons from the previous layer.
- **Net input:** The net input z computes the product of the input and the weight added to the bias term. When considering one input neuron as in Figure 4.10, a

scalar input x is multiplied by the scalar weight w and added to bias term providing the net input z. Net input for one neuron case is defined as

$$z = wx + b \tag{4.37}$$

When considering multiple input neurons as shown in Figure 4.11, the net input

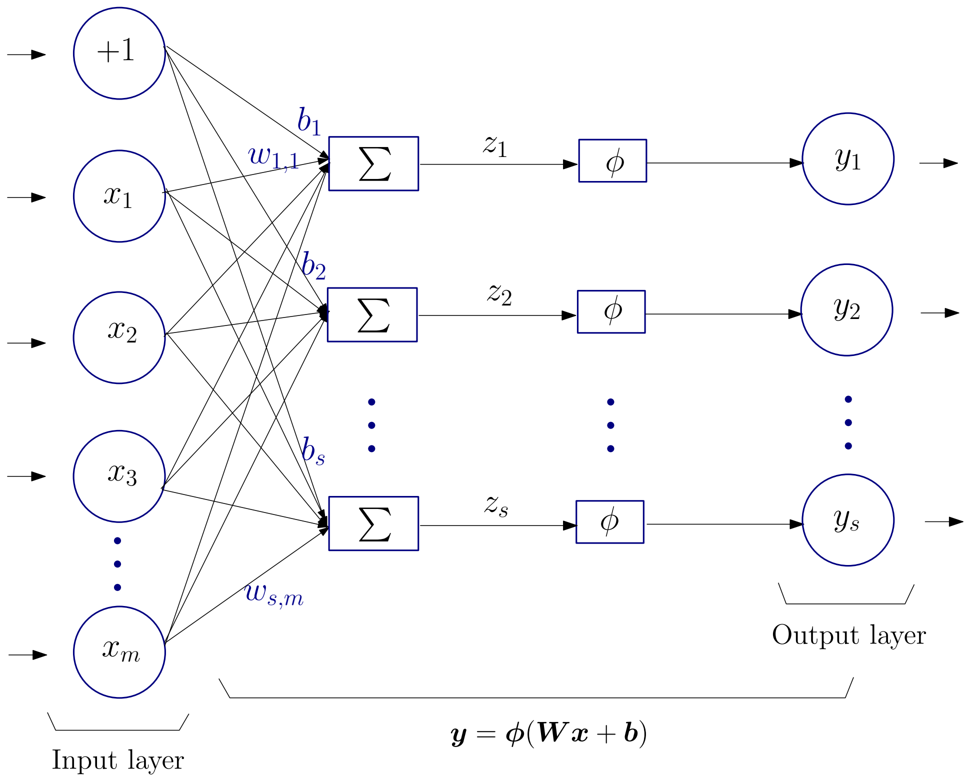


FIGURE 4.11: One layer Neural Network architecture with multiple (m) input neurons and multiple (s) output neurons; z is net input function which is addition of weighted input and bias; ϕ is a activation function.

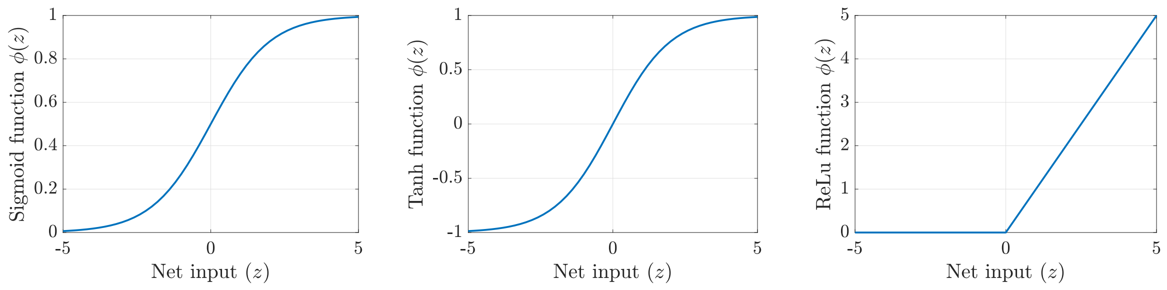
computes the inner product of the input vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$, the weight matrix $\boldsymbol{W}$ and the bias $\boldsymbol{b} = [b_1, b_2, \ldots, b_s]^T$ to produce a net input signal, usually denoted by $\boldsymbol{z} = [z_1, z_2, \ldots, z_s]^T$. Net input for multiple neuron case is defined as

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \tag{4.38}$$

where

$$\boldsymbol{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots & w_{1,m} \\ w_{2,1} & w_{2,2} & \ldots & w_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ w_{s,1} & w_{s,2} & \ldots & w_{s,m} \end{bmatrix}. \tag{4.39}$$

- ☐ **Activation Function:** Activation function is also referred to as transfer function, and it determines the output of the neural network. The activation function is attached to each neuron in the network and determines whether it should be activated or not based on each neuron's input, which is relevant for the model prediction. The function takes the net input signal, then compress the values between a certain range, and outputs the signal. There are different types of activation functions that are used in ANN depending on the application. Some of the common activation functions used in the present thesis are sigmoid function (Figure 4.12a), hyperbolic tangent sigmoid function or tanh function (Figure 4.12b) and Rectified Linear Unit function (ReLU) (Figure 4.12c).



(a) Sigmoid function (b) Tanh function (c) ReLU function

FIGURE 4.12: Plot representation of activation functions; sigmoid function (a) tanh function (b) and ReLU function (c).

Sigmoid function is defined as

$$\phi(z) = \frac{1}{1+e^{-z}}. \tag{4.40}$$

When the activation function is the sigmoid, it takes the net signal and outputs the value between 0 and 1, and it has a smooth gradient. The gradient of the sigmoid function vanishes as the value of the respective inputs increases or decreases, which is known as one of the sources to cause the vanishing gradient problem [176]. Nowadays, the sigmoid function is one of the most widely used activation functions.

Tanh function is similar to sigmoid function. Tanh function is defined as

$$\phi(z) = \frac{1-e^{-z}}{1+e^{-z}}. \tag{4.41}$$

When the activation function is tanh, it takes the net signal and outputs a value between -1 and 1. As it is zero centred, it makes it easier to model inputs that have strongly negative, neutral and strongly positive values. Similar to the sigmoid function, tanh also has the smooth gradient and vanishing gradient problem. Tanh is also a very popular and widely used activation function.

In order to handle the vanishing gradient problem, ReLU function is proposed in [177], and is defined as

$$\phi(z) = \max(0, z). \tag{4.42}$$

The function returns 0 if it receives any negative input, but for any positive value, it returns that value back. ReLu is less computationally expensive than tanh and sigmoid because it involves simpler mathematical operations, which makes it feasible to use in the deep networks. However, mathematically, the ReLU has two problems: non-differentiable at $z = 0$, thus not valid to use it along with gradient-based method and unbounded in the positive side that causes a potential problem for overfitting. Nevertheless, these problems can be tackled. For the first one, in practice, the gradient of the ReLU is set either 0 or 1, and for the later one, regarding unboundedness on the positive side, the regularisation techniques can be helpful to magnitude the weights, thus overcoming the overfitting issue. In practice, for deep neural networks, ReLu function and L_2 regularisation could be trained efficiently [178].

- **Regularisation** This is a form of regression that tends to shrink the weights of each neuron in the ANN towards zero. In other words, this technique discourages learning a complex model so as to avoid the risk of overfitting. A user-defined parameter λ is introduced to address the issue of overfitting and is used in the evaluation of the cost function, that is, when the error between the predicted and real value is calculated. More information will follow in the derivation of the cost function later in this section.

In a feedforward network, the connections between neurons are in one (forward) direction. It is arranged in the form of layers. In such a layered network, there is no connection between the neurons in the same layer and no feedback between the layers. MLP is a fully connected layered feedforward network, i.e., every node in any layer is connected to every node in its adjacent forward layer. The MLP architecture is presented in Figure 4.13, where there is an input layer (first layer), an output layer (last layer), and the layers between them are called hidden layers. The hidden layer is composed of neurons, and the outputs of these neurons would be the input to the next layer until the final output is achieved. There are $h - 1$ hidden layers shown in Figure 4.13.

MLP is trained with a backpropagation learning algorithm. Backpropagation is a method to optimise the connection weights starting from the output layer and propagating backward by adapting the weights, layer by layer, to compensate for error during the training, and efficiently dividing the error among the connections. Technically, backpropagation efficiently computes the gradient of the loss function associated with a given state with

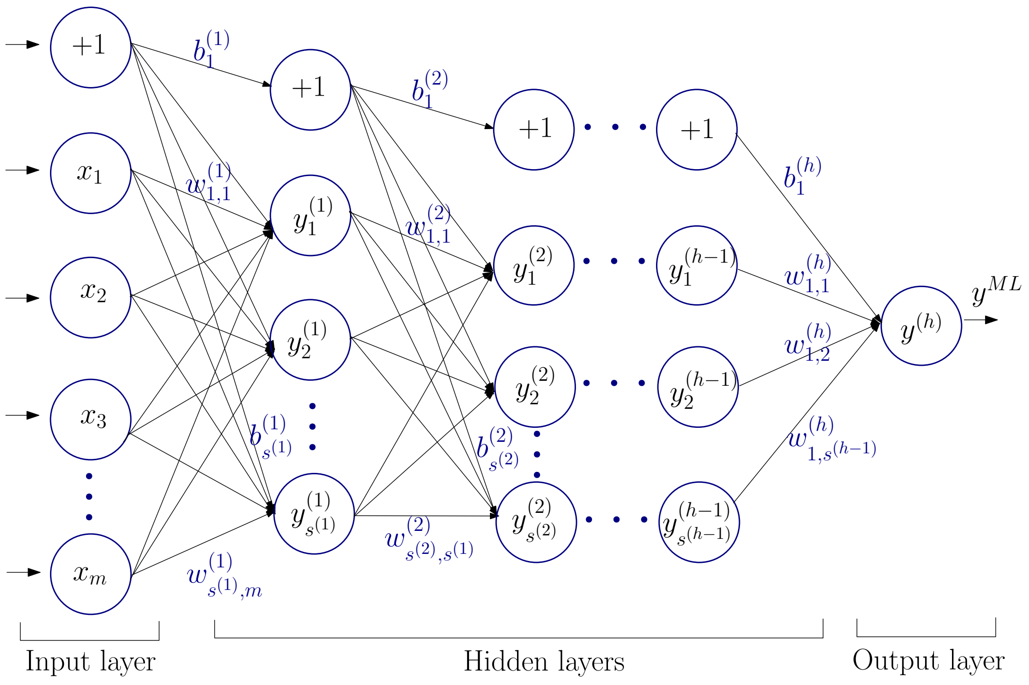


FIGURE 4.13: Fully connected MLP architecture.

respect to the weights. Backpropagation employs the gradient descent method to minimise the squared error between the network output values and desired values for those outputs. The mathematical formulation of this will be discussed later in this section. Figure 4.14 shows the complete cycle of an epoch. In MLP networks, the training term epoch is used to describe a complete pass through all the training patterns [4].

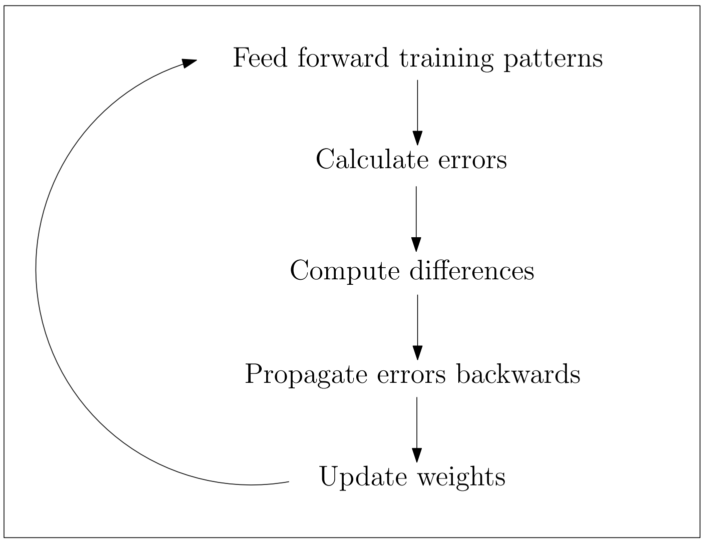


FIGURE 4.14: The epoch cycle of backpropagation MLP architecture. The figure is adapted from [4].

For forward propagation of MLP, the following formulation is used. For m number of training samples, the input layer can be written as

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T \,. \tag{4.43}$$

In order to calculate the net input for the first layer, the linear relationship between weights, input values and bias is calculated, and it is written as

$$z_1^{(1)} = \sum_{i=1}^{m} W_{1,i}^{(1)} x_i + b_1^{(1)}. \tag{4.44}$$

Similarly $y_2^{(1)}$ up to $y_s^{(1)}$ are calculated. With the net input vector $\boldsymbol{z}^{(1)} = [z_1^{(1)}, z_2^{(1)}, \ldots z_s^{(1)}]$, the output of the first hidden layer is therefore

$$\boldsymbol{y}^{(1)} = \boldsymbol{\phi}^{(1)} \left(\boldsymbol{z}^{(1)}\right), \tag{4.45}$$

$$\boldsymbol{y}^{(1)} = \boldsymbol{\phi}^{(1)} \left(\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right), \tag{4.46}$$

where the activation function for first layer is $\boldsymbol{\phi}^{(1)}$, and typically the same activation function is used for each single layer, so it can be expressed as

$$\boldsymbol{y}^{(1)} = \sum_{j=1}^{s^{(1)}} \boldsymbol{E}_j^{(1)} \phi^{(1)} \left(\boldsymbol{E}_j^{(1)} \cdot (\boldsymbol{W}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)})\right), \tag{4.47}$$

Where $\boldsymbol{E}_j^{(1)} = [0, \ldots, 0, 1, 0, \ldots, 0]^T$ is a suitably defined unit vector. Similar to Equation (4.46), for the second hidden layer, the activation function is expressed as

$$\boldsymbol{y}^{(2)} = \boldsymbol{\phi}^{(2)} \left(\boldsymbol{W}^{(2)}\boldsymbol{y}^{(1)} + \boldsymbol{b}^{(2)}\right). \tag{4.48}$$

The procedure is repeated until the last layer is reached. The previous layer's output is the input for the current layer. Likewise, for h layer, a generalised form can be written as

$$\boldsymbol{y}^{(h)} = \boldsymbol{\phi}^{(h)} \left(\boldsymbol{W}^{(h)}\boldsymbol{y}^{(h-1)} + \boldsymbol{b}^{(h)}\right). \tag{4.49}$$

The final output is of the form

$$\boldsymbol{y}^{ML} = \boldsymbol{y}^{(h)}. \tag{4.50}$$

For example when the output layer has only one neuron, like in Figure 4.13

$$y^{(h)} = \phi\left(z^{(h)}\right), \tag{4.51}$$

where

$$z^{(h)} = \boldsymbol{W}^{(h)}\boldsymbol{y}^{(h-1)} + b_1^{(h)}, \tag{4.52}$$

where the matrix $\boldsymbol{W}^{(h)}$ for the single output neuron case only has one row. When the MLP is used for multiclass classification, another activation function called softmax function is used in the MLP network, and it is of the form

$$\phi(z_i) = \frac{e^{z_i}}{\sum_{g=1}^{s} e^{z_g}}, \quad i = 1, 2, \ldots, s. \tag{4.53}$$

The output of the softmax activation function can be interpreted as the probability associated with each class. Each output will fall between 0 and 1, and the sum of the outputs are equal to 1. Typically softmax is used only for the output layer of the MLP that needs to classify multiple categories.

For Equation (4.50), the error between the predicted and the expected value can be calculated by using a loss function (also known as an error function and a cost function) depending on the specific type of application. In the case of regression, the loss function MSE (Equation (4.2)) is widely used. Whereas, in the case of classification, the cross-entropy loss function is preferred.

In binary classification, where the number of classes K is 2, then the cross-entropy with regularisation is calculated as

$$J(\boldsymbol{W}^{(1)} \ldots \boldsymbol{W}^{(h)}; \boldsymbol{b}^{(1)} \ldots \boldsymbol{b}^{(h)}) = \mathcal{J}(\boldsymbol{W}^{(1)} \ldots \boldsymbol{W}^{(h)}; \boldsymbol{b}^{(1)} \ldots \boldsymbol{b}^{(h)}) + \mathcal{R}(\boldsymbol{W}^{(1)} \ldots \boldsymbol{W}^{(h)}), \tag{4.54}$$

$$\begin{aligned} &J(\boldsymbol{W}^{(1)} \ldots \boldsymbol{W}^{(h)}; \boldsymbol{b}^{(1)} \ldots \boldsymbol{b}^{(h)}) \\ &\qquad = \frac{1}{n}\sum_{j=1}^{n}[y_j \operatorname{Ln}(y_j^{ML}) + (1 - y_j) \operatorname{Ln}(1 - y_j^{ML})] + \lambda\frac{1}{n}\sum_{l=1}^{h}\frac{1}{2}||\boldsymbol{W}^{(h)}||^2, \end{aligned} \tag{4.55}$$

where, n is the number of columns in the input matrix (or total number of samples) and $y_j^{ML} = y_j^{(h)}$ found in Equation (4.50). The first half of Equation (4.55) is the cross-entropy loss function and the second half is a regularisation term that addresses the overfitting. λ is the regularisation parameter that can be tuned. Larger weight values will be more penalised if the value of λ is large. Similarly, for a smaller value of λ, the regularisation effect is smaller.

For multiclass classification, when $K > 2$, Equation 4.55 can be extended to

$$J(\boldsymbol{W}^{(1)} \dots \boldsymbol{W}^{(h)}; \boldsymbol{b}^{(1)} \dots \boldsymbol{b}^{(h)}) = \frac{1}{n} \sum_{j=1}^{n} \sum_{nc=1}^{K} [y_{j,nc} \operatorname{Ln}(y_{j,nc}^{ML}) + (1 - y_{j,nc}) \operatorname{Ln}(1 - y_{j,nc}^{ML})] + \frac{\lambda}{2n} \sum_{l=1}^{h} ||\boldsymbol{W}^{(h)}||^2. \tag{4.56}$$

The loss functions J and MSE can be minimised by applying the gradient descent procedure. When minimising J in Equation (4.55), taking the gradient with respect to the weight $\boldsymbol{W}$ is given by

$$\nabla_{\boldsymbol{W}^{(h)}} J(\boldsymbol{W}^{(1)} \dots \boldsymbol{W}^{(h)}; \boldsymbol{b}^{(1)} \dots \boldsymbol{b}^{(h)}) = \frac{1}{n} \sum_{j=1}^{n} y_j^{(h-1)} \left(y_j^{ML} - y_j\right) + \frac{\lambda}{n} \boldsymbol{W}^{(h)}. \tag{4.57}$$

The gradient descent learning rule for weight is written as

$$\boldsymbol{W} \to \boldsymbol{W} - \eta \nabla_{\boldsymbol{W}} J, \tag{4.58}$$

where η is the learning rate, a hyper-parameter that controls how much weight needs to be adjusted with respect to the gradient vector. A smaller value of η slows down the convergence and a larger value of η might fail to converge. Once the weights are updated, the forward propagation is performed to obtain new values for the activation function using updated weights. Since the biases can be treated as special weights, these are usually omitted in practical applications. This procedure is repeated until the loss function is converged.

This procedure is repeated for MSE in the case of regression. More details about the gradient descent optimisation for MLP can be found in [174, 179, 180].

4.5 Naive Bayes algorithm

The present section describes another efficient probability-based classification algorithm called Naive Bayes. Bayesian decision theory is the foundation for Naive Bayes [181]. Before proceeding to present Naive Bayes algorithms, some of the basic concepts and technical terms are as follows:

- ☐ **Probability:** Probability is a measure of the likelihood that an event will occur. The probability of an event A is denoted as $P(A)$ which is defined as

$$P(A) = (\text{number of ways event } A \text{ occurs})/(\text{total number of outcomes}). \tag{4.59}$$

- ☐ **Random variable:** A random variable is a quantity that is produced by a random process.

- ☐ **Probability distribution:** The relationship between each possible outcome for a random variable and their probabilities is called a probability distribution. The type of distribution varies based on the properties of the random variable, such as continuous or discrete. There are many different types of probability distributions, such as normal distribution, uniform distribution, exponential distribution, etc.

- ☐ **Conditional probability:** The conditional probability of event A occurring given that event B has already occurred; it is denoted as $P(A|B)$, notion for the probability of A given B. In the case, where events A and B are independent (where the event B has no effect on the probability of event A), the conditional probability of event A given event B is simply the probability of event A, that is $P(A)$. If events A and B are not independent, then the probability intersection of A and B (the probability that both events occur) is defined by

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \quad P(B) > 0, \tag{4.60}$$

- ☐ **Bayes theorem:** An alternative method for computing conditional probabilities, and it is defined as

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{4.61}$$

where, $P(A)$ and $P(B)$ are prior probability, and $P(A|B)$ is posterior probability.

A classification algorithm can be viewed as a set of discriminant functions $h^*(\boldsymbol{x})$, one for each possible class y, which are to be computed for each observation so that the observation is assigned to the one class, whose function value is maximum [182, 183]. The Naive Bayes method is defined as

$$\hat{\boldsymbol{y}} = \text{argmax}_y \, h^*(\boldsymbol{x}), \tag{4.62}$$

where $\hat{\boldsymbol{y}}$ is the predicted output. Naive Bayes uses the Bayes theorem as a discriminant function, together with probabilities of different values of each input feature and respective class in the training data, that is,

$$h^*(\boldsymbol{x}) = P(y|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|y)P(y)}{P(\boldsymbol{x})}, \tag{4.63}$$

where $P(\boldsymbol{x}), P(y)$ is prior probability and $P(y|\boldsymbol{x})$ is posterior probability. In general, during the training phase, solving $P(\boldsymbol{x}|y)$ is difficult. However, to overcome this difficulty, Naive Bayes classifier assumes conditional independence. Based on that, $P(\boldsymbol{x}|y)$ can be represented as

$$P(\boldsymbol{x}|y) = \prod_{i=1}^{n} P(\boldsymbol{x}_i|y) \tag{4.64}$$

such that

$$P(y|\boldsymbol{x}) = \frac{P(y)\prod_{i=1}^{n} P(\boldsymbol{x}_i|y)}{P(\boldsymbol{x})}, \tag{4.65}$$

$P(\boldsymbol{x})$ in (4.65) is called evidence, which is independent of the label, so it can be neglected without affecting the classification results [184], and it can be written as

$$\hat{\boldsymbol{y}} = \text{argmax}_y P(y|\boldsymbol{x}) = \text{argmax}_y \left[P(y) \prod_{i=1}^{n} P(\boldsymbol{x}_i|y) \right]. \tag{4.66}$$

Naive Bayes makes the assumption that the x_j's in $\boldsymbol{x}$ are conditionally independent for a given y. Even if the Naive Bayes assumption is not true, it often results in classifiers that work well, one reason for this is that the model is quite simple, and hence it is relatively immune to overfitting [185]. In addition, calculating the probability of a given real-value, such as $\boldsymbol{x}$ is arduous. To overcome this issue, it can be assumed that the $\boldsymbol{x}$ is drawn from a probability distribution, namely normal or Gaussian distribution (real-valued features), multinomial distribution (categorical features), and Bernoulli distribution (binary features) depending on the feature values.

The normal distribution is chosen when dealing with real-valued features. The probability distribution can be computed using the probability density function, and it is defined as

$$P(x_j|y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_j-\mu)}{2\sigma^2}} \tag{4.67}$$

where $\boldsymbol{x} = x_j,\ j = 1, \ldots, m,\ \mu$ is mean and σ is standard deviation. This is most the widely used Naive Bayes model [175, 181].

4.6 Optimisation of hyper-parameters

In this section, optimisation of hyper-parameters for the algorithms SVM and MLP will be discussed. Most of the ML and deep learning algorithms have some parameters that need be adjusted, which are called hyper-parameters. The Hyper-parameters are optimised during the training of ML algorithms as they are crucial and directly control the functioning of the algorithm. A good choice of hyper-parameters is essential in order to build a robust and accurate model, thus, preventing the model from overfitting or underfitting [186]. Although several automatic optimisation techniques exist, they have different strengths and drawbacks when applied to different types of problems [187].

Considering SVM classification, the most important hyper-parameters are the type of kernel functions, the kernel scale and the regularisation parameter C in (4.23), which are explained in Section 4.3. Recently, to optimise the hyper-parameters for SVM classification, [188] and [189] used the resampling procedure and the Kalman filter method, respectively. For the SVM regression, the hyper-parameters can be optimised using the analytically assisted genetic algorithm [190] and generalised pattern search algorithms [191]. This is performed in a staggered manner[3] , i.e., by fixing the kernel function and regularisation parameter and optimising the kernel parameter heuristically. The regularisation parameter is given a default value following Reference [194]. In the case of SVM regression, a similar approach is used for optimising hyper-parameters. By fixing the kernel function and regularisation parameter and epsilon value and optimising the kernel parameter heuristically.

For MLP, the main parameters are weights, activation functions, number of hidden layers, and the number of hidden neurons. The weights have been adjusted by the backpropagation algorithm, which has been discussed in Section 4.4. The activation function in MLP determines the output of the network for the given input. The most commonly used activation functions for classification are sigmoid, hyperbolic tangent sigmoid, which are in Section 4.4.

Another important hyper-parameter in MLP is the number of hidden neurons and the number of hidden layers. Hidden layers and hidden layer neurons play a vital role in the performance of the backpropagation neural network [195]. The hidden neuron can

[3] This kind of optimisation can also be referred as trial and error method [192, 193].

influence the error on the nodes to which their output is connected. The stability of the network is estimated by the evaluation of the error. The minimal error reflects better stability, and higher error reflects the worst stability [196]. The excessive number of hidden neurons will cause overfitting, that is, the MLP has overestimated the complexity of the target problem. Few numbers of hidden neurons cause underfitting which increases the training error [197]. In the last decade, there were several methods to select the hidden neuron but there were no generally accepted theories on how many hidden neurons are needed to approximate any function in one hidden layer [196]. To solve any non-linear complex problem one or two hidden layers are sufficient; if the accuracy or the minimal error is the most important criterion then one can adapt the third hidden layer, and this will increase the overall complexity of the network and the training time [195]. Now, choosing the number of hidden neurons in each layer is crucial. According to [198], the number of hidden layer neurons is 2/3 of the size of the input layer and by [199], the number of hidden layer neurons should be less than twice the number of neurons in the input layer. In the present thesis, for both MLP classification algorithm, the activation functions (linear, tanh or sigmoid) and the number of hidden layers are optimised in a staggered manner. The 2^n, $n = 1, \dots 8$ method has been adopted for the number of hidden neurons in each layer. The numbers are adapted by analysing accuracy or error plot for classification and regression, respectively. The results from this section will be discussed for both diagnosis and prognosis in Sections 6.2 and 7.2 respectively.

4.7 Chapter conclusion

The chapter has presented the general introduction of ML and the theoretical formulation of different ML algorithms such as SVM, MLP, and NB. The functionality of each of these algorithms has been broadly discussed, along with their hyper-parameters. Furthermore, optimising hyper-parameters for these different ML algorithms have been discussed. With this chapter, the mathematical background of ML algorithms, which are required for performing fault diagnosis and fault prognosis, has been obtained.

Chapter 5

DATA COLLECTION AND PREPROCESSING

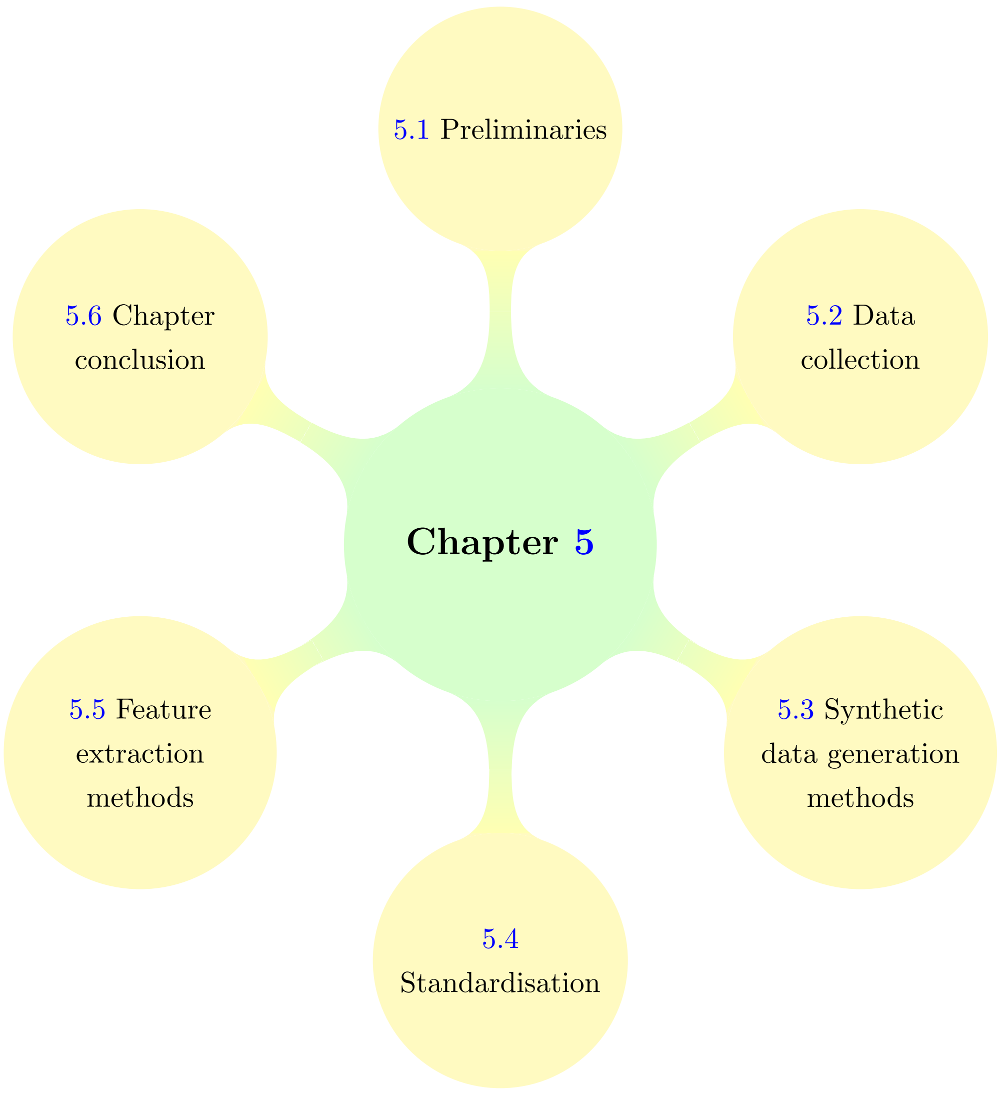


FIGURE 5.1: Structure of Chapter 5.

5.1 Preliminaries

In the present chapter, data collection, synthetic data generation methods and preprocessing methods are discussed. A brief introduction regarding data collection techniques and their types, especially the importance of employing sensor data, is presented in Section 5.2. In case of insufficient data availability, as it is the case for the specific application discussed in the thesis, synthetic data generation methods for fault diagnosis and prognosis need to be implemented; this is presented in Section 5.3. Subsequently, the standardisation method is described in Section 5.4 and the different feature extraction methods are included in Section 5.5. The breakdown of the current chapter is presented in Figure 5.1.

5.2 Data collection

For successful diagnosis and prognosis of a system, data collection is a fundamental part of Predictive Maintenance. The significance of data-driven models is dependent on both data quality and data quantity. There are various sources of data important for the purpose of maintenance, namely maintenance and repair operation data, diagnosis system data, ambient and environmental condition data, and condition monitoring (sensor) data [23, 37, 200]. In recent years, sensors have become smarter, smaller, and easier to implement in existing systems, making them more reliable. Different types of data are collected using different types of sensors, namely micro sensors, ultrasonic sensors, acoustic emission sensors, etc., [201, 202]. Usually, sensors measure mechanical magnitudes such as time-varying values of acceleration, pressure, flow rate, torque, force, vibration, temperature, etc. Vibration signals have been put into practice for diagnosis and prognosis for many years, and they are still a widely employed method [203, 204].

In past years, data has been acquired in different ways while handling different mechanical devices. For the aircraft fault prognosis system, Reference [37] recorded condition monitoring data, environmental data, and incomplete information about past maintenance actions over a period of five years. In order to generate unavailable information on the past maintenance actions, the autoregressive moving average model has been used.

Authors in Reference [205] built a simulation model of a rolling element bearing to produce vibration data, considering typical vibration excitations in addition to the wear. A similar approach has been followed in the thesis, whereby using a high-fidelity CFD simulation model of the external gear pump. Information regarding the performance of the pump is acquired via its simulation under different working conditions. The theory and numerical examples of CFD simulations are presented in Chapters 2 and 3. In addition, synthetic data generation methods are implemented to further increase the quantity of high-fidelity in-silico data, which will be discussed in the following section.

5.3 Synthetic data generation methods

In the present section, synthetic data generation methods used for diagnosis and prognosis are discussed. Firstly, the synthetic data generation method for diagnosis by perturbing the frequency content with noise is presented. Subsequently, the generation of degradation datasets using linear and cubic degradation interpolation functions are discussed.

5.3.1 Synthetic data generation method for fault diagnosis using a noise perturbation method

In order to improve the learning phase of any ML algorithms, an important requirement is to have a large quantity of data. Due to the absence of experimental data in the context of the current thesis, synthetic data must be generated. The CFD model has been used in order to generate high-fidelity data by recreating a variety of working conditions for the gear pump. The high-fidelity data created using the CFD model is perturbed using a noise perturbation method to recreate possible environmental variations of the working pump conditions; this process is shown in Figure 5.2.

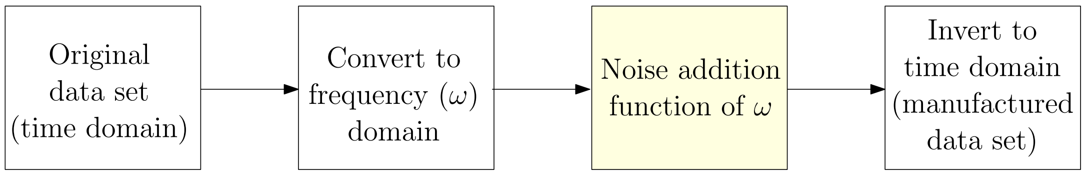

Original data set (time domain)
Convert to frequency (ω) domain
Noise addition function of ω
Invert to time domain (manufactured data set)

FIGURE 5.2: Overview of data preparation using noise addition.

Consider $x(t)$ a continuous time-varying field (i.e. pressure, flow rate, etc.) in the time interval $[0, T]$

$$\begin{aligned} x: \quad & [0,T] \subset \mathbb{R} \to \mathbb{R} \\ & t \mapsto x(t), \end{aligned} \tag{5.1}$$

where $x(t)$ is the value of the field at time t. Let us define a discrete set ($\mathcal{D}$) of time values being chosen for the sampling evaluation of the field x in time, namely

$$\mathcal{D} = \{t_1, t_2, \dots, t_m\}; \quad t_i = (i-1)\Delta t, \quad i = 1, \dots, m; \quad \Delta t = \frac{T}{m-1}. \tag{5.2}$$

Thus, the discrete set $\boldsymbol{x}$ is given by,

$$\boldsymbol{x} = [x(t_1), x(t_2) \dots x(t_m)]^T, \quad \boldsymbol{x} \in \mathbb{R}^m. \tag{5.3}$$

The discrete set $\boldsymbol{x}$ is converted to the frequency domain using a Discrete Fourier Transform (DFT) that computes the discrete Fourier transform ($\tilde{\mathcal{F}}_{\Delta t}$) [206], and it is expressed as

$$\hat{\boldsymbol{x}} = \tilde{\mathcal{F}}_{\Delta t}(\boldsymbol{x}) = [\hat{x}(\omega_1), \hat{x}(\omega_2), \dots \hat{x}(\omega_m)]^T, \quad \hat{\boldsymbol{x}} \in \mathbb{R}^m, \tag{5.4}$$

where $\hat{\boldsymbol{x}}$ represents the discrete approximation of the Fourier transform for a set of discrete frequencies $\mathcal{W} = \{\omega_1, \omega_2, \dots, \omega_m\}$. The objective to perturb the DFT $\hat{\boldsymbol{x}}$, by adding noise ϵ in a selected group of frequencies contained in a set $\tilde{\omega} \subset \mathcal{W}$. Specifically,

$$\begin{aligned} \epsilon \ : \quad & \tilde{\omega} \subset \mathcal{W} \to \mathbb{R} \\ & \tilde{\omega}_j \mapsto \epsilon(\tilde{\omega}_j). \end{aligned} \tag{5.5}$$

As an example, the set $\tilde{\omega}$ is defined on the basis of local maxima observed frequencies contributing to the frequency spectrum. For instance, Figure 5.3 shows the DFT of a given monitored field x, where the set of selected frequencies $\tilde{\omega} = \{\tilde{\omega}_1, \tilde{\omega}_2, \dots \tilde{\omega}_{10}\}$ can be observed.

In the present thesis, a specific strategy has been followed in order to generate synthetic data. The objective is to add noise in the response of the DFT with larger noise content for higher frequencies. One of the ways to achieve this is by considering a monotonically increasing quadratic function,

$$\epsilon(\tilde{\omega}_j) = p_1 \tilde{\omega}_j^2 + p_2 \tilde{\omega}_j + p_3, \quad j = 1, 2, \dots n, \tag{5.6}$$

with unknown coefficients p_1, p_2 and p_3. In order to find these coefficients, system of three equations need to be solved. These three equations are generated by imposing constraints. The constraints are selected to have a balance between the generated time

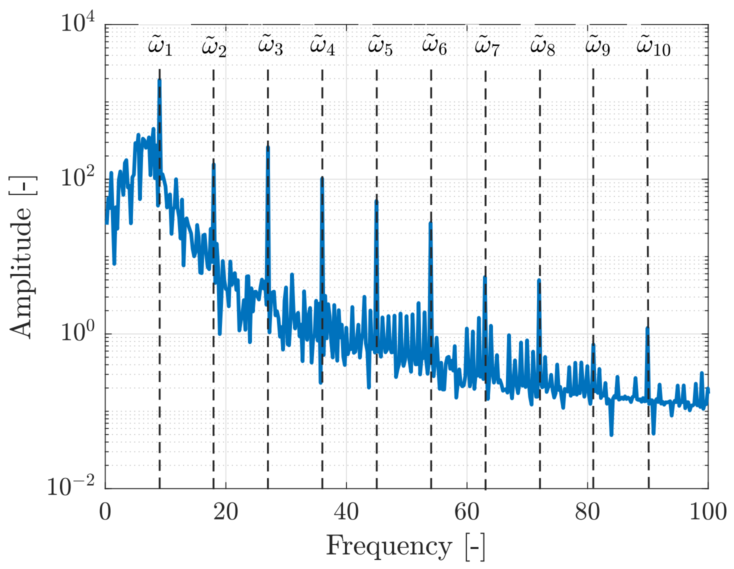


FIGURE 5.3: Pressure as a function (discrete Fourier transform) of frequency in normal working condition of gear pump where high frequencies of frequency spectrum are chosen to add noise.

history and the original time history. A small noise content will not produce any difference in the generated time history but, on the other hand, a larger noise content can produce an unreasonable time history. For example, ϵ is assumed to be zero for $\tilde{\omega}_1$ since adding noise in the first peak can create a very high-level difference with the original data. Similarly, ϵ is assumed to be 1 for $\tilde{\omega}_3$ for the same reason. The third equation is created by assuming the last peak value to be 100 because when the noise is added to the last peak, the effect generated by the noise is very small. In Figure 5.3, only $\tilde{\omega}_1$ to $\tilde{\omega}_{10}$ is displayed.

Thus, we can construct a matrix $\hat{\boldsymbol{X}} \in \mathbb{R}^{m\times(n+1)}$ by using Equations (5.4) and (5.6),

$$\hat{\boldsymbol{X}} = \begin{bmatrix} \hat{x}(\omega_1) & \hat{x}(\omega_1) & \hat{x}(\omega_1) & \dots & \hat{x}(\omega_1) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \hat{x}(\tilde{\omega}_1)(1+\alpha\epsilon(\tilde{\omega}_1)) & \vdots & \dots & \vdots \\ \vdots & \vdots & \hat{x}(\tilde{\omega}_2)(1+\alpha\epsilon(\tilde{\omega}_2)) & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \hat{x}(\tilde{\omega}_n)(1+\alpha\epsilon(\tilde{\omega}_n)) \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \hat{x}(\omega_m) & \hat{x}(\omega_m) & \hat{x}(\omega_m) & \dots & \hat{x}(\omega_m) \end{bmatrix}, \quad (5.7)$$

where α is a scalar parameter. Hence,

$$\hat{\boldsymbol{X}} \cdot \boldsymbol{E}_j = [\hat{\boldsymbol{X}}]_j, \quad \text{with} \quad \boldsymbol{E}_j = [0, \dots, 0, 1, 0, \dots 0]^T, \quad (5.8)$$

where $[\hat{\boldsymbol{X}}]_j$ is the j^{th} column of $\hat{\boldsymbol{X}}$, and in $\boldsymbol{E}_j$, 1 is the j^{th} entry. Inverse Fourier transform ($\tilde{\mathcal{F}}_{\Delta t}^{-1}$) has been used to convert $[\hat{\boldsymbol{X}}]_j$ into time history $[\boldsymbol{X}]_j$, and it is expressed as,

$$[\boldsymbol{X}]_j = \tilde{\mathcal{F}}_{\Delta t}^{-1}([\hat{\boldsymbol{X}}]_j), \tag{5.9}$$

where in Equation (5.9), the first column is equal to the original time-varying discrete set $\boldsymbol{x}$, and the second and the following columns are the noise perturbed dataset. Figure 5.4a shows the time history of the original pressure dataset and two synthetically generated datasets. The original dataset, synthetic dataset 1 and synthetic dataset 2 are the inverse Fourier transform of first column, second column and third column in Equation 5.7, with $\alpha = 1$. Figure 5.4b shows the time history of the synthetic pressure dataset, and inverse Fourier transform of third column in Equation 5.7, by varying the scalar parameter $\alpha = 1, 10, 20$.

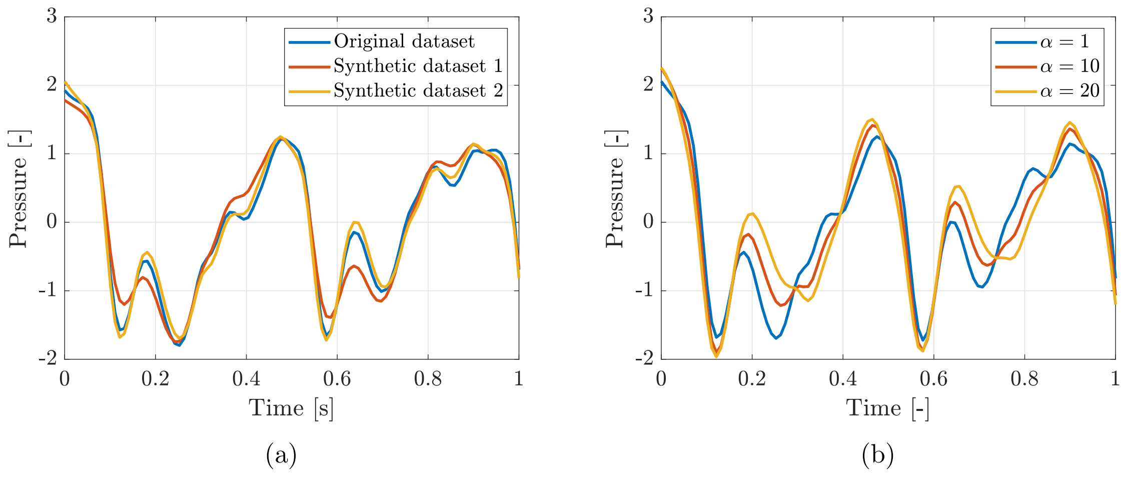


FIGURE 5.4: Time history of the original pressure dataset and two synthetic datasets. The original dataset, synthetic dataset 1 and synthetic dataset 2 are the inverse Fourier transform of first column, second column and third column in Equation 5.7, with $\alpha = 1$ (a). The inverse Fourier transform of third column (synthetic dataset 2) with $\alpha = 1, 10, 20$ in Equation 5.7 (b).

In the present thesis, the ML supervised algorithms are utilised to perform fault diagnosis, which requires the input and output dataset to train the algorithm. The concept of ML and its algorithms are presented in Chapter 4. In the context of fault diagnosis, data generated using the noise perturbation method are employed as an input dataset to extract features using feature extraction methods. Eventually, these extracted features will be used as an input dataset for ML algorithms. Subsequently, the output datasets are created based on the working condition of the gear pumps (healthy, faulty1, faulty2, etc.,) respective to the input dataset.

5.3.2 Synthetic data generation method for fault prognosis

In the present section, data generation for fault prognosis is presented. As discussed in the introduction, fault prognosis requires degradation behaviour to be present in the dataset. To achieve this objective, two methods are being considered. Firstly, using a linear interpolation method, to deteriorate from a healthy dataset (h) to a faulty dataset (f1) (Section 5.3.2.1). The second approach uses a cubic interpolation method to interpolate between a healthy dataset (h) to a series of faulty datasets (f1, f2, f3) (Section 5.3.2.2).

5.3.2.1 Approach 1 - Linear interpolation method

Consider x_h a time-varying continuous function representing a monitored field of a gear pump in a healthy working condition. Similarly, x_{f1} is the time-varying continuous function representing the same monitored field under faulty working condition. In order to obtain time-varying conditions for the situation in between these two scenarios healthy and faulty, the degradation function is presented as

$$x(t,s) = x_{f1}(t) + g(s)\left[x_h(t) - x_{f1}(t)\right], \quad s \in [0,1], \tag{5.10}$$

where

$$\begin{aligned} x(t, s=0) &= \quad x_h(t), && (5.11)\\ x(t, s=1) &= \quad x_{f1}(t), && (5.12) \end{aligned}$$

where $g(s)$ is a degradation function and defined as

$$g(s) = 1 - s. \tag{5.13}$$

In order to generate a discrete set, the time continuous function Equation (5.10) is evaluated in a discrete set of time values, rendering

$$[\boldsymbol{x}]_j = \boldsymbol{x}_{f1} + g(s_j)[\boldsymbol{x}_h - \boldsymbol{x}_{f1}], \quad j = 1, \dots, n, \tag{5.14}$$

Figure 5.5 shows the interpolated dataset between heathy dataset $\boldsymbol{x}_h$ and faulty dataset $\boldsymbol{x}_{f1}$ using the linear degradation function. For instance, if a series of faulty scenarios $(\boldsymbol{x}_{f1}, \boldsymbol{x}_{f2}, \boldsymbol{x}_{f3})$ are available, then the linear interpolation method is employed intervals, i.e., $\boldsymbol{x}_h$ to $\boldsymbol{x}_{f1}$, $\boldsymbol{x}_{f1}$ to $\boldsymbol{x}_{f2}$ and $\boldsymbol{x}_{f2}$ to $\boldsymbol{x}_{f3}$. These interpolated datasets are employed as the input dataset for fault prognosis.

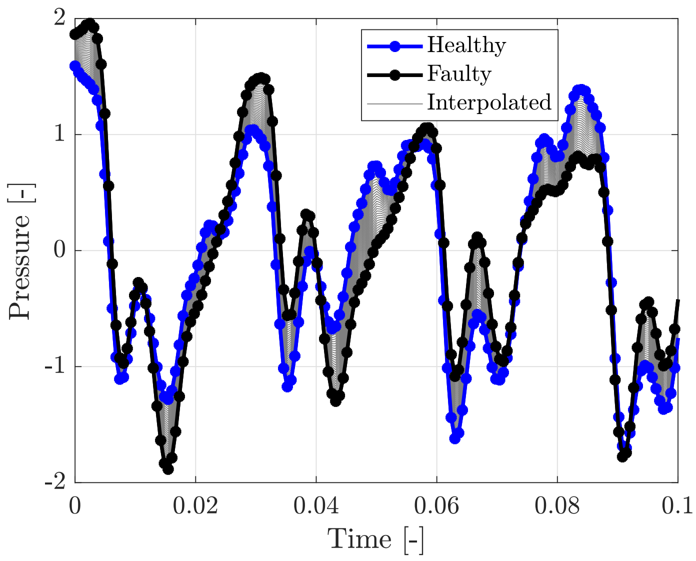


FIGURE 5.5: Interpolated pressure values between time history of healthy working condition and faulty working condition using linear degradation function presented in Equation 5.14.

5.3.2.2 Approach 2 - Cubic interpolation method

Consider x_h a time-varying continuous function representing a monitored field of the gear pump in a healthy working condition. Similarly, x_{f1}, x_{f2} and x_{f3} are the time-varying continuous functions representing the same monitored field under various faulty working conditions. To obtain the time-varying conditions for the situation in between these four scenarios, a cubic interpolation method can be used. A Lagrange interpolation polynomial can be used to yield,

$$x(m,t) = \sum_{i=1}^{4} x_i(t) g_i(m), \quad g_i(m_j) = \delta_{ij}, \quad \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases} \tag{5.15}$$

where x represents the time-varying condition of four working scenarios x_h, x_{f1}, x_{f2}, and x_{f3} of the gear pump, δ_{ij} is Kronecker delta function and m is the model parameter, which is assumed to be gap sizes of radial gap faulty scenarios $m = [m_1, m_2, m_3, m_4] = [0.03, 0.05, 0.07, 0.09]$. The radial gap faulty scenarios are presented in Section 3.7. Four available model parameters can be used to form a system of equations that can be solved to find the unknown coefficients. Figure 5.6 shows the interpolated time-varying pressure values for radial gap faulty scenarios for 1 second (100-time steps).

A similar approach can be used for axial gap faulty scenarios and speed variations faulty scenarios by using model parameters as gap sizes and speed values, respectively. This method can be utilised when a series of fault information is available. These synthetically

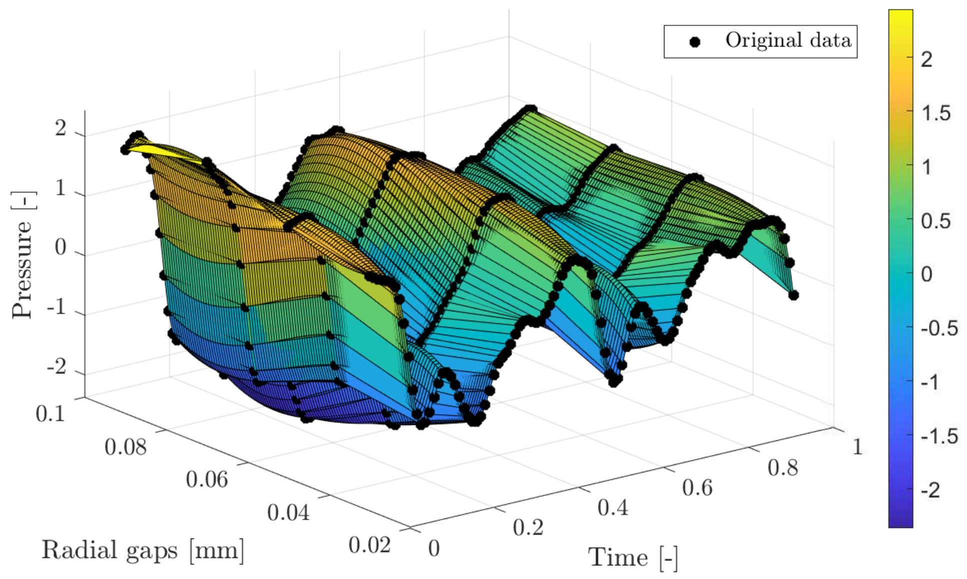


FIGURE 5.6: Time-varying pressure values for radial gap faulty scenarios for 1 second (100-time steps) created using the cubic interpolation method.

generated datasets are directly utilised as the input dataset for ML algorithms to perform prognosis.

5.3.2.3 Cubic decay function

As discussed in the previous section, in the present thesis, ML supervised algorithms are employed to perform fault prognosis. In order to train the ML algorithms, along with the input dataset, the output dataset is also needed. The output dataset of fault prognosis is defined as the RUL, which is a real number, quantifying the remaining life of the gear pump. In the absence of the experimental dataset, RUL can be considered as any decaying function. In this thesis, linear and cubic decaying functions are considered as possible RUL functions. Figure 5.7 shows the possible functions that can be used as RUL of a gear pump, with 100-time units being assumed as the maximum lifetime of the gear pump. It represents that with the increase of life, the RUL is decreasing.

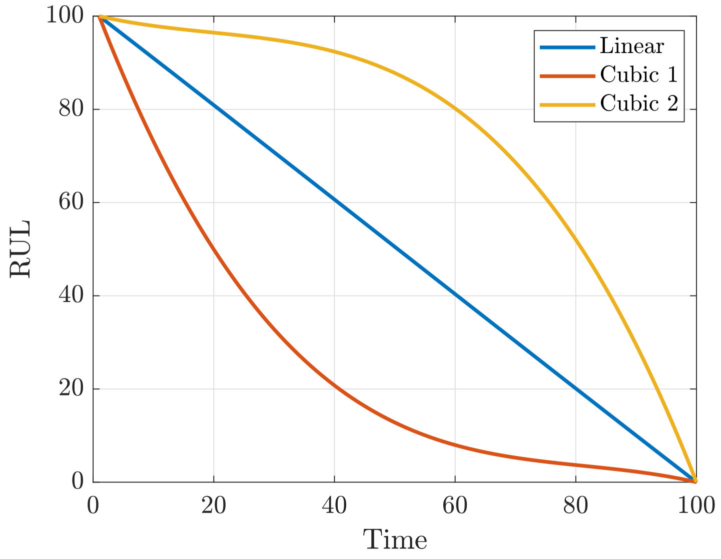


FIGURE 5.7: Possible functions for RUL, linear decay function and cubic decay functions.

5.4 Standardisation

Data standardisation is necessary for ML algorithms since it changes the values of numeric columns in the dataset to a common scale, without distorting differences in the range of values. In some ML algorithms, without the use of standardisation, the minimisation of the objective function may not be efficiently achieved. For instance, when attempting a classification problem, ML algorithms (in this context, known as the classifiers) compute the Euclidean distance between two data points. If one of the data points has a broad range of values, then the distance is governed by this particular data point. Standardisation can be related to preconditioning as it makes the gradient descent converge much faster than without it [104]. The standardisation makes each point in the dataset have zero-mean and unit variance, and it is expressed as

$$[\boldsymbol{Y}]_j = \frac{[\boldsymbol{X}]_j - \mu([\boldsymbol{X}]_j)}{\sigma([\boldsymbol{X}]_j)}, \tag{5.16}$$

where

$$\mu([\boldsymbol{X}]_j) = \frac{1}{m}\sum_{i=1}^{m}[\boldsymbol{X}]_{ij}\ ; \quad \sigma([\boldsymbol{X}]_j) = \frac{1}{m-1}\sum_{i=1}^{m}\left([\boldsymbol{X}]_{ij} - \mu([\boldsymbol{X}]_j)\right)^2, \tag{5.17}$$

and $[\boldsymbol{X}]_j$ is a original dataset, $\mu([\boldsymbol{X}]_j)$ its mean, and $\sigma([\boldsymbol{X}]_j)$ its standard deviation. This method has been widely used for standardisation in ML algorithms such as SVM, Logistic regression and ANN [207]. The standardisation is applied to all datasets prior to ML analyses.

5.5 Feature extraction methods

Feature extraction transforms high dimensional data into lower-dimensional, which contains the most discriminatory information that helps to overcome the problem of excessive computational time. In AI, as well as in other ML algorithms, it is essential to find an appropriate representation of multivariate data [208]. The features are expected to contain the relevant information from the input dataset so that the desired task can be performed by using this reduced representation instead of the complete initial dataset [209]. In this thesis, three types of feature extraction methods are employed to compare their efficiency in the performance of fault diagnosis, namely time features, frequency features and wavelet features.

Time features

Time features, such as mean, standard deviation, skewness, kurtosis, max, shape factor, crest factor, root mean square and energy are extracted from the time-varying monitored field. These extracted features are used as input features for fault diagnosis in various applications [58, 59, 210]. Table 5.1 shows the mathematical definitions of some relevant time-domain features, which are utilised to extract features from the time-varying monitored field, and they are used to perform fault diagnosis. Time-domain features, namely standard deviation, skewness and kurtosis are shown in Figure 5.8 for the healthy working condition (h) and the three different faulty working conditions (f1, f2, and f3).

Frequency features

The frequency features in the present thesis are spectral based features. Spectral kurtosis and Power Spectral Density (PSD) are combined to extract features to perform fault diagnosis. These methods are proven to be very useful in signal analysis [211]. The spectral kurtosis is initially defined as the kurtosis of its frequency components, and it

[4] In skewness equation, $E(\boldsymbol{x})$ represents the “expected” value of $\boldsymbol{x}$ or mean.

Features	Equations
Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
Standard deviation	$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} \lvert x_i - \mu\rvert^2}$
Skewness[4]	$s = \frac{E(\boldsymbol{x}-\mu)^3}{\sigma^3}$
Kurtosis	$k = \frac{E(\boldsymbol{x}-\mu)^4}{\sigma^4}$
Root Mean Square	$\mathrm{RMS}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} x_i^2$
Crest factor	$\mathrm{Max}(\boldsymbol{x})/\mathrm{RMS}(\boldsymbol{x})$
Shape factor	$\mathrm{RMS}(\boldsymbol{x})/\mu$
Impulse factor	$\mathrm{Max}(\boldsymbol{x})/\mu$
Margin factor	$\mathrm{Max}(\boldsymbol{x})/\mu^2$
Energy	$\sum_{i=1}^{N} x_i^2$

TABLE 5.1: Time domain features and their mathematical formulas.

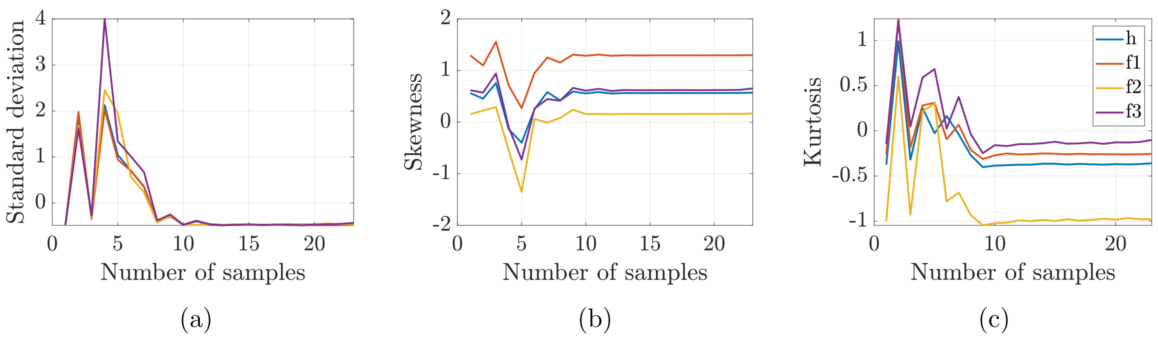


FIGURE 5.8: Time domain features for different working conditions, healthy (h) and various fault scenarios (f1, f2, and f3).

is defined as [61],

$$sk = \frac{\sum_{k=b_1}^{b_2} (f_k - \mu_1)^4 s_k}{(\mu_2)^4 \sum_{k=b_1}^{b_2} s_k}, \tag{5.18}$$

where f_k is the frequency corresponding to bin k (bin refers to the intervals between the samples), s_k is the spectral value at bin k, b_1 and b_2 are band edges in bins, μ_1 is the spectral centroid and μ_2 is the spectral spread. μ_1 and μ_2 are defined as,

$$\mu_1 = \frac{\sum_{k=b_1}^{b_2} f_k s_k}{\sum_{k=b_1}^{b_2} s_k}, \tag{5.19}$$

$$\mu_2 = \sqrt{\frac{\sum_{k=b_1}^{b_2} (f_k - \mu_1)^2 s_k}{\sum_{k=b_1}^{b_2} s_k}} \tag{5.20}$$

When the spectral kurtosis calculated, it returns as a double vector that contains low values where data is stationary and Gaussian, and high values where transients occur. However, depending on the window size chosen, the returned vector can be in a large dimension. So, in this thesis, mean, standard deviation, skewness, and kurtosis (table 5.8) are extracted from spectral kurtosis. Figures 5.9a, 5.9b and 5.9c show the mean, standard deviation and skewness extracted from spectral kurtosis, where the latter samples identify the detailed features of different faulty scenarios.

PSD function shows the strength of the variations (energy) as a function of frequency. The PSD is defined by [63]

$$S_{xx}(\omega) = \lim_{T\to\infty} E|\hat{x}(\omega)|, \tag{5.21}$$

where E denotes expected value, $\hat{x}(\omega)$ is Fourier transform of $x(t)$, and it is defined as

$$\hat{x}(\omega) = \frac{1}{\sqrt{T}} \int_0^T x(t) e^{-i\omega t} dt. \tag{5.22}$$

When the PSD is calculated, it returns the power of a spectrum, which is in a large dimension depending on the signal. So, in this thesis, mean frequency, median frequency (middle value separating the greater and lesser halves of a signal), band power (average power of a signal), and power bandwidth (3-dB (half-power) bandwidth[5]) are extracted

[5] To determine the 3-dB bandwidth, power bandwidth computes a periodogram power spectrum estimate using a rectangular window and takes the maximum of the estimate as a reference level. The bandwidth is the difference in frequency between the points where the spectrum drops at least 3 dB below the reference level. If the signal reaches one of its endpoints before dropping by 3 dB, then power bandwidth uses the endpoint to compute the difference.

from PSD. Figures 5.9d, 5.9e, and 5.9f show the mean, power bandwidth and band power, where the behaviour of features are different from the ones of spectral kurtosis. So, combining features based on spectral kurtosis and PSD together, the information acquired has advantage of both features.

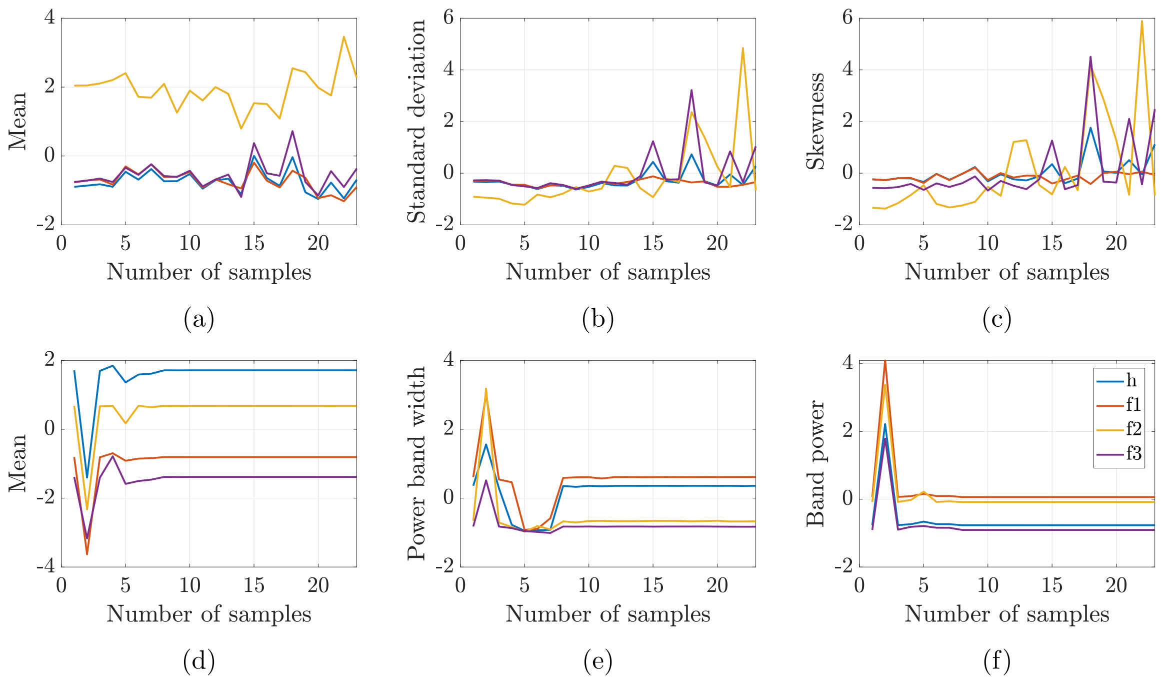


FIGURE 5.9: Spectral kurtosis based features (a, b, c) and PSD based features (d, e, f) for different working conditions, healthy (h) and various fault scenarios (f1, f2, and f3).

The extracted features from spectral kurtosis and PSD are used to perform the fault diagnosis.

Wavelet features

The wavelet features in the present thesis are based on Shannon entropy values of the Maximal Overlap Discrete Wavelet Packet Transform (MODWPT). MODWPT decomposes the time series into its components on different equal length-frequency sub-bands. The theory of frequency to wavelet transforms, including MODWPT, is given in Appendix B. Although MODWPT can reveal the local characteristics of an input signal, the number of such coefficients is usually very large so that the direct use of those coefficients as features to the classification can be difficult. Therefore, some effective features may be derived from these coefficients for better classification. Entropy is a tool to measure the uncertainty of the information contained in the given system, and it is widely used in signal processing and pattern recognition [212]. Shannon entropy, log energy entropy, Renyi entropy and Tsallis entropy are some of the widely used types of entropy methods. Shannon entropy in Definition 4 is a measure of uncertainty associated with random variables in information theory, and it can be calculated based on the probability distribution of energy [213], and that is used in the thesis to extract the informative features from MODWPT. Some of these features extracted using Shannon entropy from the MODWPT are shown in Figure 5.10, where it can be seen that the features are able to capture different resolutions in different samples, which are used as input features for fault diagnosis.

Definition 4: **Shannon's entropy**
The entropy cost function of a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ is

$$SE(\boldsymbol{x}) = -\sum_{i=1}^{n} p_i \mathrm{Ln}(p_i); \quad p_i = |x_i|^2/||\boldsymbol{x}||^2. \tag{5.23}$$

Since $\sum_i p_i = 1$ and $p_i \geq 0$ by construction, the measure SE is Shannon's entropy of the discrete probability distribution $\{p_1, \ldots, p_n\}$.

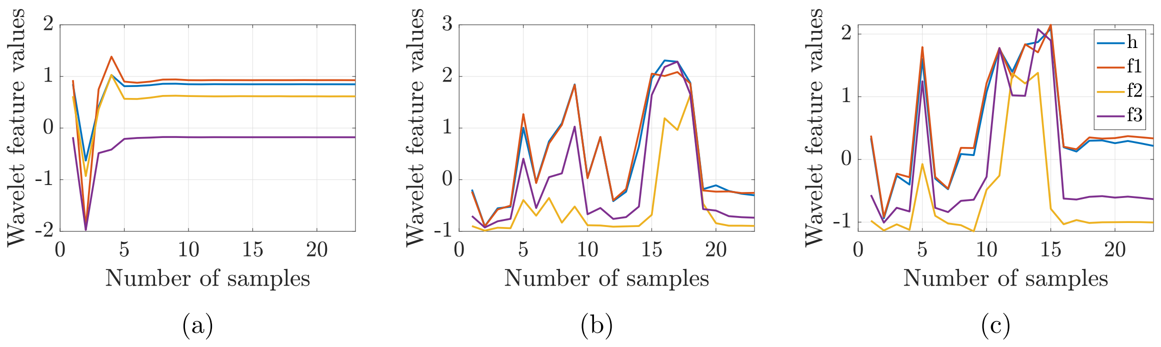


FIGURE 5.10: Wavelet domain features for different working conditions, healthy (h) and various fault scenarios (f1, f2, and f3).

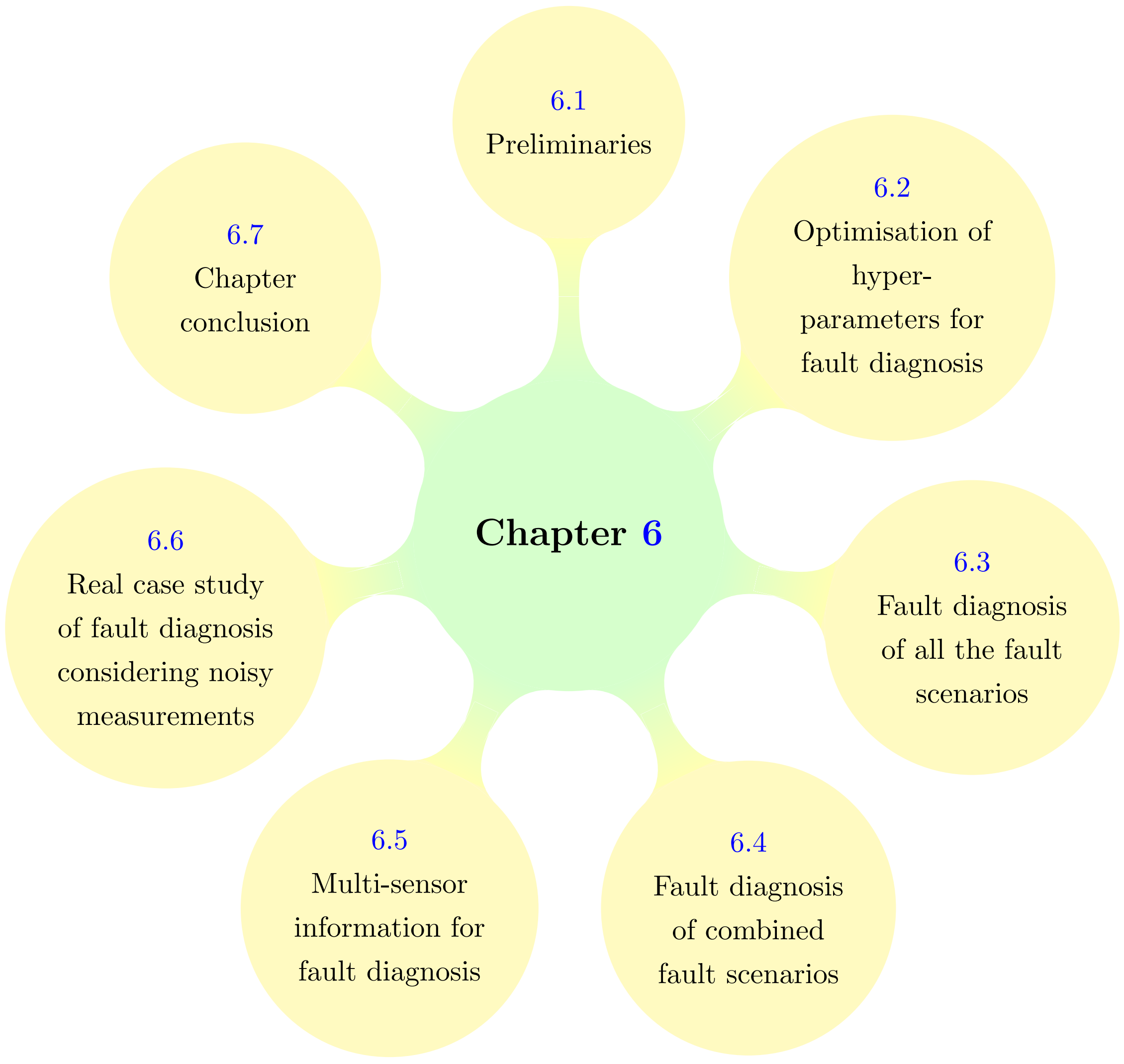
The three feature extraction methods mentioned are compared to each other with the performance of fault diagnosis. The numerical examples related to this section will be presented in Chapter 6.

5.6 Chapter conclusion

The present chapter has presented data collection methods and data preprocessing methods to perform fault diagnosis and fault prognosis using ML algorithms. In the absence of the experimental dataset, synthetic data generation methods are implemented for both diagnosis and prognosis. With this chapter, datasets required for conducting ML applications to Predictive Maintenance have been obtained.

Chapter 6

NUMERICAL EXAMPLES FOR DIAGNOSIS



6.1 Preliminaries
6.2 Optimisation of hyper-parameters for fault diagnosis
6.3 Fault diagnosis of all the fault scenarios
6.4 Fault diagnosis of combined fault scenarios
6.5 Multi-sensor information for fault diagnosis
6.6 Real case study of fault diagnosis considering noisy measurements
6.7 Chapter conclusion
Chapter 6

FIGURE 6.1: Structure of Chapter 6.

6.1 Preliminaries

The objective of the present chapter is the numerical simulation of fault diagnosis (fault detection) of the external gear pump by means of three alternative ML algorithms, namely MLP, SVM and NB. These ML algorithms have been presented in Chapter 4. The overview of the fault diagnosis simulation process is depicted in Figure 6.2. The layout of the present chapter is summarised in Figure 6.1. Fault diagnosis of various industrial equipment components has been recently studied in [66, 96, 214, 215] using ML algorithms: SVM, MLP, generative adversarial network, decision trees, SVM, k-nearest neighbors, ensemble and convolutional NN.

ML predictive capability is strongly dependent upon the availability of large sets of high-fidelity data [216]. In case of absence (or limited availability) of experimental data, these can be superseded using high-fidelity in-silico data. The high-fidelity data has been generated using an accurate CFD model resembling the operations of an external gear pump in healthy and various fault conditions (clogging, radial gap variations, etc.). The aspect of high-fidelity data generation has been discussed in Chapter 3. In order to enhance the predicted response of any ML algorithm, large amounts of data are usually necessary. To greatly facilitate the gathering of data, a synthetic data generation method is employed. A noise perturbation technique is used to recreate more data from the in-silico data, and the process of generating data is presented in Section 5.3.1. The generated datasets are used to extract features using the feature extraction methods.

The methods being employed to extract features from time series are time features (mean, standard deviation, etc.), frequency features (spectral kurtosis, power spectral density) and wavelet transform features (MODWPT). The feature extraction methods are presented in Section 5.5. The extracted features are utilised as the input dataset for the ML classification algorithms, and the respective categories related to input datasets are labelled as output datasets (i.e., healthy1, healthy2, . . ., faulty1, faulty2, etc.). Once the dataset is gathered, it is divided into a training dataset (used to fit the model) comprised of 70% of the dataset and a test dataset (used to provide an unbiased evaluation of a final model fit on the training dataset) comprised of the remaining 30% [217]. The training dataset is used to train each ML algorithm, which is tested against the test dataset, and the accuracy of the model is computed using Equation (4.1).

In addition, to understand the efficiency of ML algorithms and the chosen feature extraction methods, fault diagnosis is performed with the data containing multi-sensor

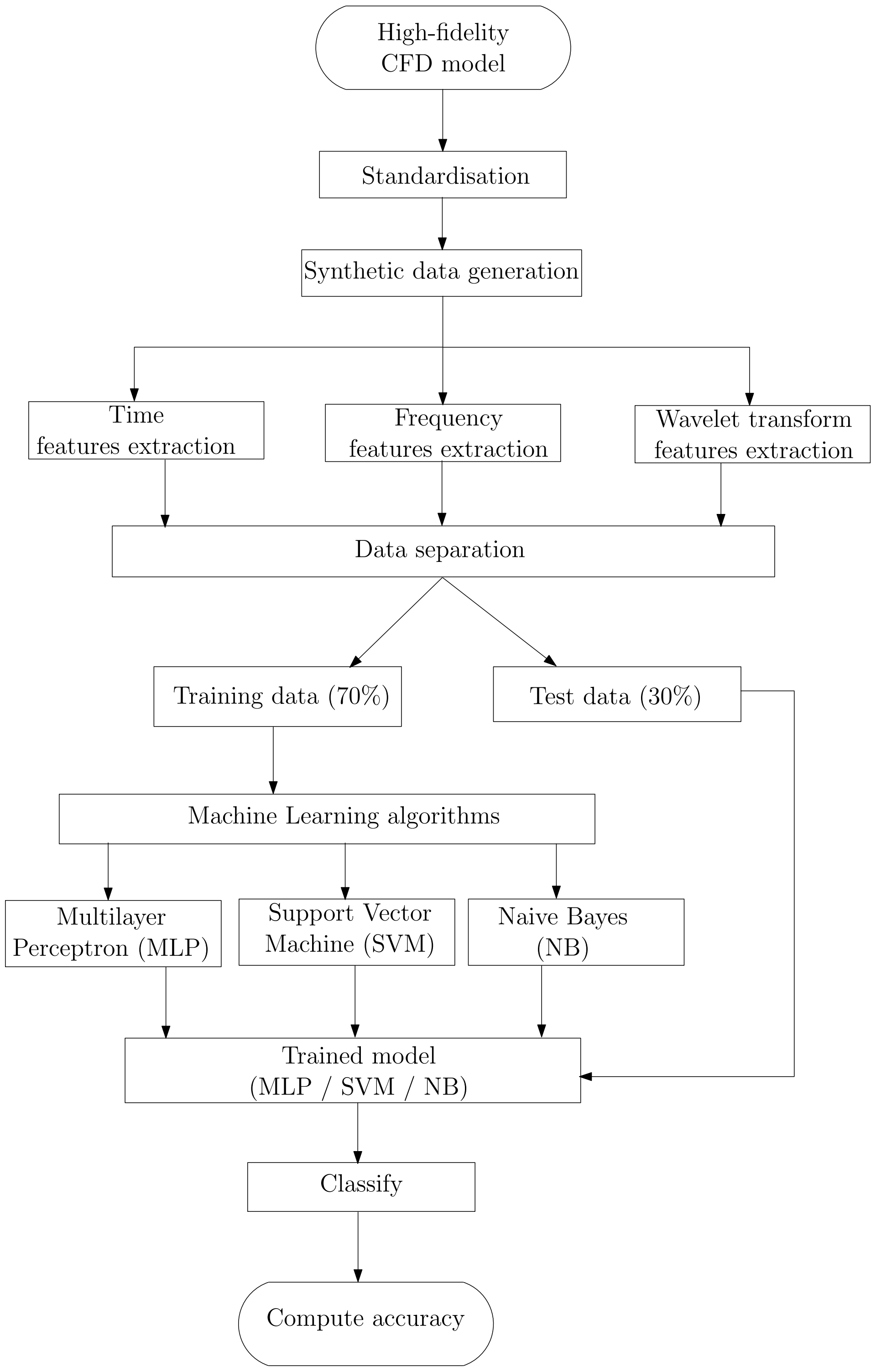

High-fidelity CFD model
Standardisation
Synthetic data generation
Time features extraction
Frequency features extraction
Wavelet transform features extraction
Data separation
Training data (70%)
Test data (30%)
Machine Learning algorithms
Multilayer Perceptron (MLP)
Support Vector Machine (SVM)
Naive Bayes (NB)
Trained model (MLP / SVM / NB)
Classify
Compute accuracy

FIGURE 6.2: Overview process of fault diagnosis using ML classification algorithms.

information. Also, a real case study of fault diagnosis is employed by inducing white noise in the train and test dataset to understand the sensitivity of ML algorithms [218]. This analysis is performed by obtaining the fault diagnosis accuracy using each ML algorithm considered.

The present chapter is summarised as follows: (1) Optimisation of hyper-parameters for fault diagnosis using ML classification algorithms is presented in Section 6.2. (2) Fault diagnosis of all implemented fault scenarios using ML algorithms is presented in Section 6.3 (3) Fault diagnosis of the combination of fault scenarios due to clogging and viscosity variations using ML algorithms is discussed in Section 6.4. (4) Fault diagnosis of all implemented fault scenarios with multi-sensor information using ML classification algorithms is presented in Section 6.5. (5) A real case study of fault diagnosis considering noisy measurements is described in Section 6.6. (6) The chapter conclusions are presented in Section 6.7.

6.2 Optimisation of hyper-parameters for fault diagnosis

The objective of the present section is to describe the methodology used for the optimisation of hyper-parameters to perform fault diagnosis. As an example, time features of the pressure dataset from all implemented fault scenarios are considered for this particular analysis. In order to increase the availability of data to train the ML algorithm for fault diagnosis, the synthetic data generation method is employed (see Section 5.3.1). Similar analyses have been carried out considering flow rate data and torque data with all feature extraction methods.

The methodology used for the optimisation of hyper-parameters has been discussed in Section 4.6. Firstly, the optimisation of hyperparameters is discussed for SVM classification, then for the MLP classification, and then for the NB classification. ML algorithms SVM, MLP and NB have been discussed in Sections 4.3, 4.4 and 4.5, respectively. In the current section, the dataset has been separated into training and validation dataset. The validation dataset is used to provide an unbiased evaluation of a model fit on the training dataset while characterising the model hyper-parameters. The holdout sample validation method ([219]) is used to hold a 30% of the dataset for validation purposes.

Hyper-parameter optimisation for SVM classification

For SVM classification, three hyper-parameters are considered, namely kernel function, kernel parameter (Section 4.3.4) and regularisation parameter (C in Equation (4.23)). To ensure the optimal choice of hyper-parameters for SVM, it is necessary to optimise these parameters. The optimisation of these hyper-parameters is performed in a staggered manner[6], i.e., by fixing the kernel function and regularisation parameter and optimising the kernel parameter heuristically. Kernel functions, namely linear kernel, Gaussian kernel, and polynomial kernel are considered for this analysis. The regularisation parameter is considered with default value of C=1. Even though the regularisation parameter value can range between $[10^{-3}, 10^3]$, C=1 has been chosen as it is sufficient to provide good results [194]. Utilising this method, the optimisation of hyper-parameter is performed for fault diagnosis.

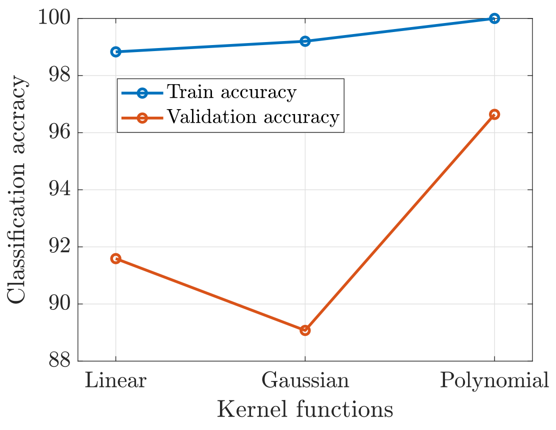
Following the fault diagnosis procedure discussed in the introduction 6.1 of this chapter, fault diagnosis is performed for the desired dataset with standardisation[7] and without standardisation. As a result, the classification accuracy is calculated. Figure 6.3 shows the SVM classification accuracy of pressure (time features) with standardisation (a) and without standardisation (b) for different kernel functions: linear, Gaussian and polynomial. In Figures 6.3a and 6.3b, considering both training and validation accuracy, polynomial kernel performs with highest accuracy compared to linear and Gaussian kernel functions. When using polynomial kernels, considering both training accuracy and validation accuracy, the one with standardisation has higher accuracy than the one without standardisation. Following these results, all the examples presented in this thesis are performed with standardisation.

Hyper-parameter optimisation for MLP classification

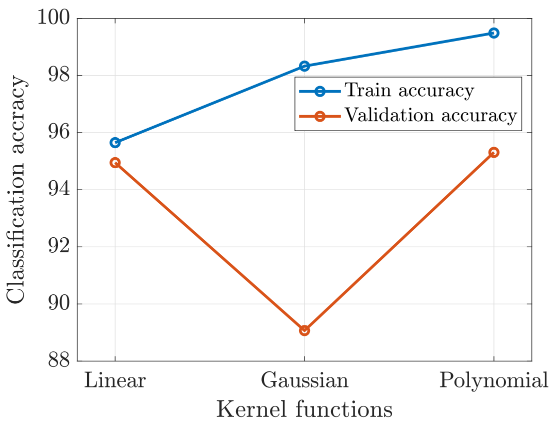
For MLP classification, three hyper-parameters are considered, namely the activation functions, the number of hidden layers and the number of hidden neurons in each hidden layer, which are discussed in Section 4.4. The activation functions (linear, tanh or sigmoid) and the number of hidden layers are optimised in a staggered manner. By fixing the activation function and the number of hidden layers, and optimising the number of hidden neurons using a 2^n method, where $n = 1, \ldots, 8$. The 2^n approach is based on the forward approach [221] and is carried out by fixing the number of hidden neurons, for which 2^n neurons per hidden layer are considered to speed up the identification

[6] This kind of optimisation can also be referred to as trial and error method [192, 193].

[7] Standardised refers to z-score normalisation and it expressed as $x_i - \mu/\sigma$, where x_i is a i^{th} entry in $\boldsymbol{x}$, and μ and σ is mean and standard deviation of $\boldsymbol{x}$ [220]. Further explanations can be found in Section 5.4.



(a) Dataset with standardisation

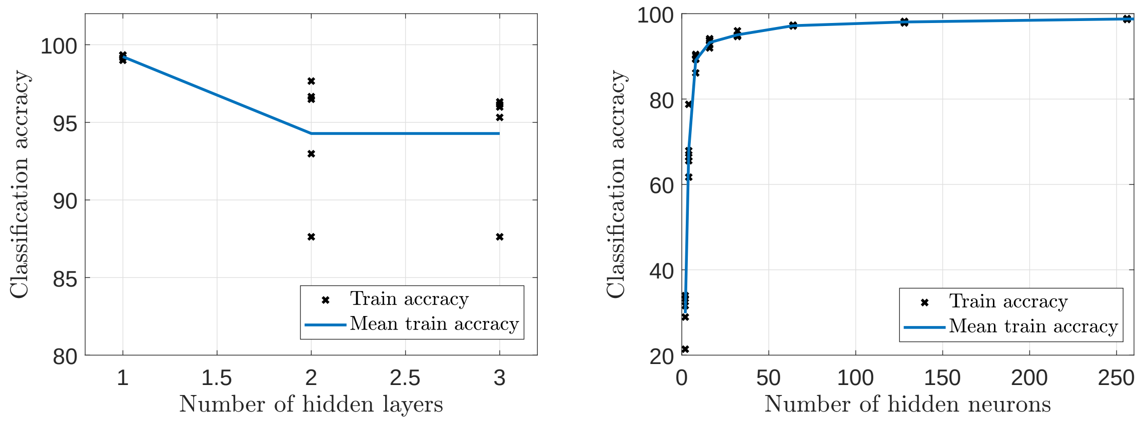


(b) Dataset without standardisation

FIGURE 6.3: SVM Classification accuracy of pressure (time features) with standardisation (a) and without standardisation (b) for different kernel functions, linear, Gaussian and polynomial.

of the precise number of neurons in the layer. When training the MLP network, the backpropagation algorithm is used to optimise the updating of the weights, according to the loss function specified when compiling the model. The backpropagation algorithm requires that the network is trained for a specified number of epochs to the training dataset. This aspect has been discussed in Section 4.4. Each epoch can be partitioned into groups called batches. The batch size defines the number of patterns that the network is exposed to before the weights are updated within an epoch. It is common practice to show the model loss with respect to the number of epochs. This plot can detect if the model is overfitting, underfitting or suitably fitting to the training dataset [4]. In addition, during the training, *Early Stopping* argument is used to stop the training when the chosen performance measure stops improving [222].

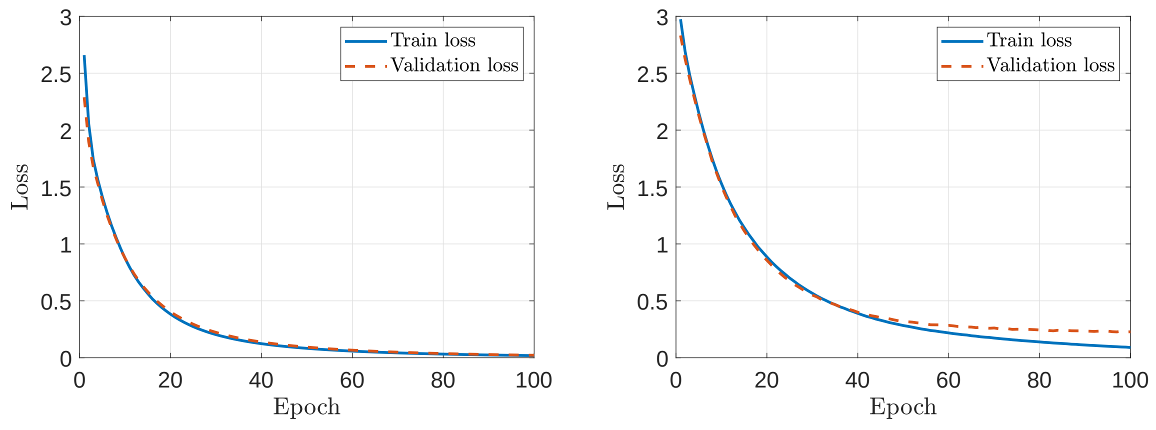
Similar to the previous example, fault diagnosis has been performed using MLP considering standardised time features of the pressure dataset. The optimal choice of hyper-parameters is considered based on the fault diagnosis accuracy obtained. Considering activation functions, the tanh function outperforms the sigmoid function for MLP classification, so this function is chosen to perform fault diagnosis. Figures 6.4a and 6.4b show the accuracy plot for different hidden neurons and hidden layers, which is used to optimise the number of hidden neurons and hidden layers. Each figure shows the accuracy of the training dataset along with their mean accuracy values. Each asterisk symbol in the figure shows each training accuracy of the ML algorithm. The process is repeated five times to acquire an average accuracy since during each MLP training, random weights are initialised and optimised using the backpropagation algorithm. In addition, Figure 6.4 shows that one layer with 256 neurons is sufficient to provide higher accuracy and there is no need for a second or third layer.



(a) (b)

FIGURE 6.4: Classification accuracy of pressure (time features) features) with respect to number of hidden layers (1, 2, 3) and the number of hidden neurons (2, 4, 8, 16, 32, 64, 128, 256) for 1 hidden layer (a) using MLP.

As discussed earlier, the model loss shows how well the model handles the training and avoids overfitting, this is shown in Figure 6.5. The x-axis shows the number of epochs. One of the asterisk symbol in Figure 6.4a is expanded over number of epochs. The model has good performance on both train and validation datasets and avoids to overfit in Figure 6.5a when one hidden layer is considered. The drift between training and validation loss are shown in Figure 6.5b shows the overfitting of the model when two hidden layers are considered, whereas one hidden layer is sufficient to perform the analysis. For all the examples in this thesis, similar approach is conducted.



(a) (b)

FIGURE 6.5: Model loss on training and validation datasets. Model trained with one layer of neurons (a) and two layers of neurons (b) using MLP.

Hyper-parameter optimisation for NB classification

As discussed in Section 4.5, Naive Bayes assumes that all the input features are independent. Even if the naive Bayes assumption is not true, it often results in classifiers that work well. One reason for this is that the model is quite simple, and hence it is relatively immune to overfitting [185]. So, there is no hyper-parameter to be optimised in the case of Naive Bayes with a normal distribution function.

6.3 Fault diagnosis of all the fault scenarios

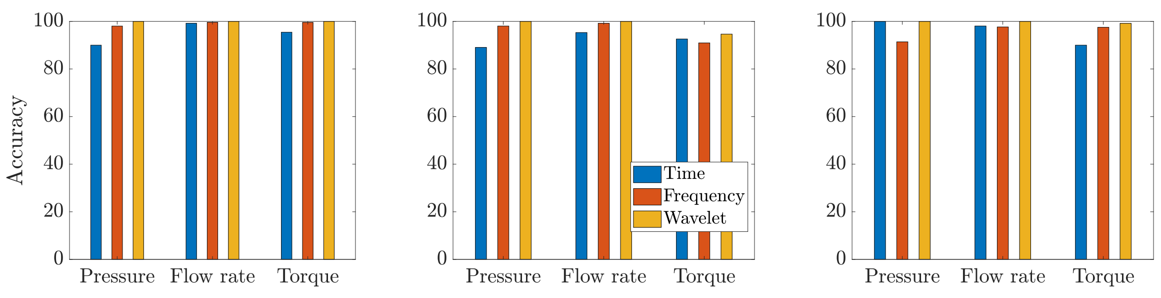
The purpose of the present section is to perform fault diagnosis using MLP, SVM, and NB for all implemented fault scenarios of the external gear pump. This example is quite relevant because, in a practical situation, a piece of equipment can be affected by more than one fault scenario at the same time. Applying this idea, all fault scenarios implemented in the thesis are used to perform the fault diagnosis. The overview designed process of fault diagnosis is shown in Figure 6.2. The ML algorithms used have been discussed in Chapter 4.

For this example, consider the healthy dataset (H) and all six types of fault scenarios implemented with 18 different cases (C1, C2, C3, G1, G2, G3, S1, S2, S3, T1, T2, T3, W1, W2, W3 V1, V2, V3). These fault scenarios have been presented in Section 3.7. Similar to the previous example, the variables such as pressure, flow rate and torque are monitored for all the fault scenarios. These datasets are in the form of a discrete set of time values. In order to enhance the availability of data to train the ML algorithm, the synthetic data generation method is employed (see Section 5.3.1). The generated datasets are used to extract features using feature extraction methods, namely time, frequency and wavelet features, which are discussed in Section 5.5. The extracted features are standardised and used as input dataset for training and testing ML algorithms. The output data is considered as labels (H, C1, C2,...V2, V3). The input and output datasets are divided into a training dataset of 70 % and the remaining 30 % is considered as a test dataset. The training dataset is trained using MLP, SVM and NB and tested against the test dataset, where the accuracy of the test dataset is calculated between model output and desired output.

Following the fault diagnosis procedure discussed in the introduction 6.1 of this chapter, a fault diagnosis is performed for the desired dataset. Table 6.1 and Figure 6.6 show the classification test accuracy of time features, frequency features and wavelet features extracted from pressure, flow rate and torque using ML algorithms MLP, SVM and NB. The combination of wavelet features with all the employed ML algorithms provides the highest accuracy. The combination of NB with time features provide equally high accuracy. From the accuracy obtained using ML algorithms, compared to pressure and flow rate, the torque provide relatively lower accuracy when performing with frequency features. Whereas the use of a flow rate dataset provides the highest fault diagnosis accuracy.

	Features	MLP	SVM	NB
Pressure	Time	90.01 %	89.06 %	100.0 %
	Frequency	98.03 %	98.03 %	91.40 %
	Wavelet	100.0 %	100 %	100 %
Flow rate	Time	99.22 %	95.31 %	98.05 %
	Frequency	99.61 %	99.21 %	97.65 %
	Wavelet	100.0 %	100.0 %	100 %
Torque	Time	95.45 %	92.59 %	90.01 %
	Frequency	99.58 %	90.94 %	97.53 %
	Wavelet	100.0 %	94.65 %	99.17 %

TABLE 6.1: Classification accuracy of pressure, flow rate and torque using MLP, SVM and NB.



(a) MLP (b) SVM (c) NB

FIGURE 6.6: Classification accuracy of time, frequency and wavelet features extracted from pressure, flow rate and torque using ML algorithms MLP (a), SVM (b) and NB (c).

6.4 Fault diagnosis of combined fault scenarios

The objective of the present section is to perform fault diagnosis of an external gear pump when a combination of fault scenarios occurs, using ML algorithms, namely MLP, SVM and NB. In a practical situation, any type of fault or a combination of faults can occur in a piece of equipment. Considering this idea, a combination of fault scenarios was implemented combining fault scenarios due to clogging and due to viscosity variations (H, C1V1, C1V2, C1V3, C2V1, C2V2, C2V3, C3V1, C3V2, C3V3). The overview designed process of fault diagnosis is shown in Figure 6.2. This example is divided into three smaller examples which will be discussed one by one.

Example (1)
For this example, consider the healthy dataset (H), fault scenarios due to clogging (C1, C2, C3), and fault scenarios due to viscosity variations (V1, V2, V3). The C1, C2 and C3 scenarios are outlet blocking, inlet blocking and lubrication blocking, respectively, and V1, V2 and V3 scenarios are 3%, 6% and 10% variations in the viscosity values, respectively. The fault scenarios due to a combination of faults have been discussed in Section 3.7.1. Similar to the previous section, variables pressure, flow rate and torque are monitored for the above scenarios. In order to enhance the availability of data to train the ML algorithm, the synthetic data generation method is employed (see Section 5.3.1). These generated datasets are used to extract features using feature extraction methods, namely time feature, frequency feature and wavelet feature, which are discussed in Section 5.5. The extracted features are standardised and used as an input dataset for training and testing of ML algorithms. The output data is considered as labels (H, C1, C2, C3, V1, V2, V3). The input and output datasets are divided into a training dataset of 70 % and the remaining 30 % is considered as a test dataset. The training dataset is trained using MLP, SVM and NB and tested against the test dataset, where the test accuracy is calculated between the model output and the desired output.

Following the fault diagnosis procedure discussed in the introduction 6.1 of this chapter, fault diagnosis is performed for the desired dataset. Table 6.2 shows the classification accuracy of MLP, SVM and NB for pressure, flow rate and torque using all the feature extraction methods. MLP algorithm provides the best accuracy compared to other ML algorithms. Moreover, all the ML algorithms provide better results with wavelet features and frequency features compared to time features.

	Features	MLP	SVM	NB
Pressure	Time	97.87 %	95.83 %	100.0 %
	Frequency	100.0 %	98.93 %	100.0 %
	Wavelet	100.0 %	100.0 %	95.83 %
Flow rate	Time	100.0 %	97.91 %	100.0 %
	Frequency	100.0 %	97.87 %	100.0 %
	Wavelet	100.0 %	100.0 %	96.80 %
Torque	Time	100.0 %	93.75 %	100.0 %
	Frequency	93.54 %	100.0 %	100.0 %
	Wavelet	100.0 %	100.0 %	95.83 %

Table 6.2: Classification accuracy of pressure, flow rate and torque using MLP, SVM and NB for the fault scenarios due to clogging and viscosity.

Example (2)
For this example, consider the healthy dataset (H), fault scenarios due to clogging (C1, C2, C3) and due to viscosity variations (V1, V2, V3) for training the ML algorithms. A combination of fault scenarios due to clogging and viscosity (H, C1V1, C1V2, C1V3, C2V1, C2V2, C2V3, C3V1, C3V2, C3V3) is considered as a test dataset to test the trained ML model. The pressure is monitored for all the above scenarios to perform this analysis. In order to enhance the availability of data to train the ML algorithm, the synthetic data generation method is employed (see Section 5.3.1). These generated datasets are used to extract features utilising feature extraction methods, namely time, frequency and wavelet features, which are discussed in Section 5.5. The extracted features are standardised and used as input dataset for training and testing the ML algorithms. For the training output data, the labels are considered as H, C1, C2, C3, V1, V2 and V3, and for the testing output data, the labels are considered to be H, C1V1, C1V2, C1V3, C2V1, C2V2, C2V3, C3V1, C3V2 and C3V3. In this case, the accuracy of the test dataset cannot be computed, however, the behavioural match can be found.

Table 6.3 shows the classification of pressure from combined fault scenarios using ML algorithms for three different feature extraction methods. The results depict that the

	MLP			SVM			NB		
Test set	Time	Freq	Wave	Time	Freq	Wave	Time	Freq	Wave
H	H	H	H	H	H	H	C3	V2	H
C1V1	C1	C1	C1	C2	C1	C1	V3	V3	V1
C1V2	C1	C1	C1	C2	C1	C1	V3	C1	V1
C1V3	C1	C1	C1	C2	C1	C1	V3	C1	V1
C2V1	C2	V2	C3	C2	C3	C3	C2	V1	V1
C2V2	V3	C2	V1	V1	C2	V2	V3	V3	V1
C2V3	C2	C3	C2	C2	C3	V3	V3	V1	H
C3V1	H	V3	V1	C3	V3	V2	C3	V3	V2
C3V2	V3	H	V1	C2	V2	V1	V3	V3	V1
C3V3	H	C2	V2	H	C2	V1	C3	V3	V1

Table 6.3: Classification of the pressure dataset using ML algorithms for three different feature extraction methods (Time - time features, Freq - frequency features, Wave - wavelet features). The text highlighted is correctly predicted (H), and a reasonable prediction (C1).

healthy dataset (H) has been predicted perfectly using MLP and SVM. For C1V1, C1V2 and C1V3, by using MLP and SVM, the prediction has been C1 for most cases. This explains the reason behind high sensitivity while blocking the outlet (C1) rather than adding different viscosity values (V1, V2, V3) in the gear pump. In the case of the rest of the combinations, there is no common pattern to derive any conclusions.

Example (3)

Due to the interest in understanding the opposite effect, fault diagnosis is performed. A combination of fault datasets H, C1V1, C1V2, C1V3, C2V1, C2V2, C2V3, C3V1, C3V2 and C3V3 are used for training the ML algorithms, MLP, SVM and NB. Datasets H, C1, C2, C3, V1, V2 and V3 are used for testing the trained ML model. The same procedure discussed in the above example is employed here in terms of the generation of synthetic datasets and standardisation of such datasets. In this case, the test accuracy cannot be computed, however, it would be interesting to visualise the pattern of recognition by the algorithms.

Table 6.4 shows the classification of pressure dataset using ML algorithms, MLP, SVM and NB for three different feature extraction methods. The results show that the healthy dataset (H) is predicted perfectly using MLP, SVM and NB, and they are highlighted in the Table. For C1, the prediction is C1V1, C1V2 and C1V3 in most cases except one. This would imply that in the combination dataset, blocking the outlet (C1) has higher sensitivity than adding different viscosity values (V1, V2, V3) in the gear pump. In the case of the rest of the combinations, there is no common pattern to draw any conclusions.

	MLP			SVM			NB		
Test set	Time	Freq	Wave	Time	Freq	Wave	Time	Freq	Wave
H	H	H	H	H	H	H	H	H	H
C1	C1V1	C1V2	C1V3	C1V1	C1V3	C1V3	C1V1	C2V2	C1V1
C2	C2V1	C3V3	C1V1	C2V1	C3V3	C3V2	C2V1	C2V2	C3V2
C3	H	C2V3	C2V1	H	C2V2	C3V3	H	C2V1	C3V1
V1	C1V2	C3V3	C3V2	C2V1	C2V2	C3V2	C2V1	C2V2	C3V1
V2	H	C3V2	C1V3	C2V1	C1V1	C2V2	H	C3V2	C2V1
V3	C2V2	C2V2	C2V3	C2V3	C3V3	C3V3	C1V3	C3V3	C2V1

Table 6.4: Classification of the pressure dataset using ML algorithms for three different feature extraction methods (Time - time features, Freq - frequency features, Wave - wavelet features). The text highlighted is correctly predicted (H), and a reasonable prediction (C1).

6.5 Multi-sensor information for fault diagnosis

The objective of the present section is to perform fault diagnosis with multi-sensor information datasets using MLP, SVM and NB for all the implemented fault scenarios of the external gear pump. In modern industries, a huge number of sensors can be implanted. The dataset coming from these sensors can be used for performing the fault diagnosis. We derive multi-sensor fault diagnosis in order to understand the efficiency of the ML algorithms and the chosen feature extraction methods when a more complex dataset is involved.

For this example, two sets of datasets are considered: 1) the healthy dataset (H) and fault scenarios due to radial gap degradation (G1, G2 G3). 2) The healthy dataset (H) and all six types of fault scenarios implemented with 18 different cases (C1, C2, C3, G1, G2, G3, S1, S2, S3, T1, T2, T3, W1, W2, W3 V1, V2, V3). These fault scenarios have been presented in Section 3.7. The dataset from different sensor positions of all the above scenarios is achieved by employing in-silico data generation (CFD simulations). In each fault scenario, the variable pressure is monitored in four different sensor positions (the inlet, the inlet side of the gear, the outlet and the outlet side of the gear) of the gear pump.

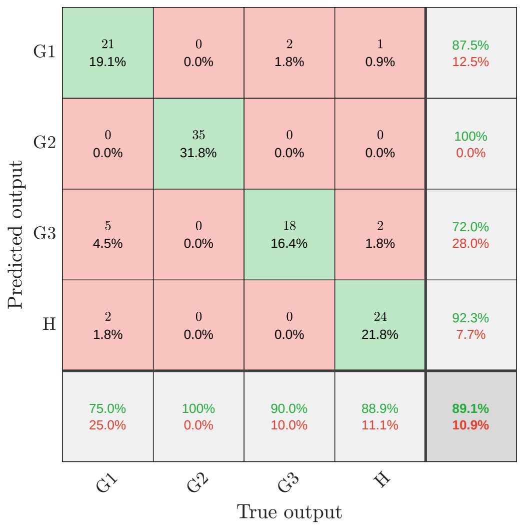
In order to enhance the availability of data to train the ML algorithm, a synthetic data generation method is employed (see Section 5.3.1). The generated datasets are used to extract features utilising feature extraction methods that are discussed in Section 5.5. The extracted features are standardised and used as input dataset for training and testing ML algorithms. The output data is considered as labels (H, C1, C2,...V2, V3). The input and output datasets are divided into a training dataset of 70 % and the remaining 30 % is considered as a test dataset. The training dataset is trained using MLP, SVM and NB and tested against the test dataset, whereas the test accuracy is calculated between model output and the desired output.

Following the fault diagnosis procedure discussed in the introduction 6.1 of this chapter, this procedure is performed for the desired dataset. The computed accuracy of the test dataset for fault scenarios due to degradation is shown in Table 6.5. MLP and SVM algorithms, with all the feature extraction methods, provide higher accuracy compared to the NB algorithm. The NB classification algorithm performs poorly with time and frequency features, whereas with wavelet features, it provides high accuracy. Figure 6.7 shows the classification accuracy of 89.1 % for SVM and 49.1 % for NB in the form of an error matrix for frequency features of the torque dataset. In the error matrix, the x-axis is the true output, and the y-axis is the predicted output. Each element represents the number of samples and the percentage of the predicted output of a corresponding class. The diagonal elements of the error matrix represent the correctly predicted output of the corresponding class. The last element provides the percentage of accuracy and error. In Figure 6.7a, using SVM, it can be seen that G2 has all correct predictions, whereas, G1, G3 and H have misclassifications. Subsequently, when using NB, Figure 6.7b shows that all the variables are misclassified, varying from SVM predictions. Among SVM and NB, SVM provides better classification accuracy.

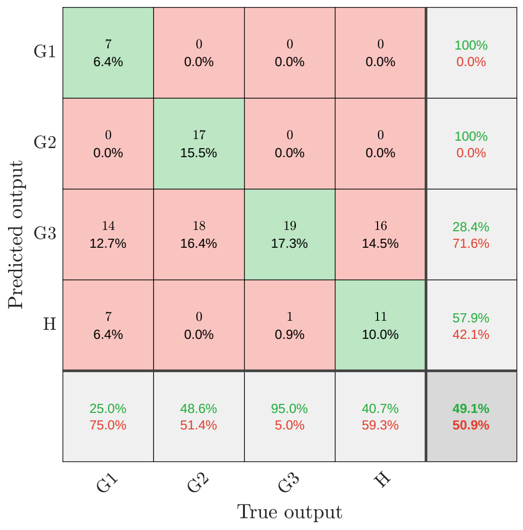
The computed accuracy of the test dataset is shown in Table 6.6 for all the fault scenarios. The NB classification algorithm performs poorly when the dataset is complex, due to its linearity. The MLP algorithm, with all the feature extraction methods, provides a relatively higher accuracy compared to other ML algorithms employed. Comparing all the feature extraction methods, wavelet features provide the highest accuracy with all the ML algorithms.

	Features	MLP	SVM	NB
Pressure	Time	84.40 %	80.90 %	48.18 %
	Frequency	85.32 %	89.09 %	49.10 %
	Wavelet	100.0 %	100.0 %	100.0 %

TABLE 6.5: Classification accuracy of pressure (multiple sensor) using MLP, SVM and NB for healthy and fault scenarios due to radial gap degradation (H, G1, G2, G3). The highlighted numbers are expanded into an error matrix, which is shown in Figure 6.7.



(a)



(b)

FIGURE 6.7: Error matrix of SVM classification (a) and NB classification (b) for the frequency-based pressure feature of four class classification. In the error matrix, each element represents the number of samples and the percentage of predicted output, and the diagonal elements represent the correctly predicted output of the corresponding class. In the last element, the percentage of accuracy and error is presented.

	Features	MLP	SVM	NB
Pressure	Time	79.54 %	65.45 %	25.42 %
	Frequency	69.40 %	59.73 %	22.70 %
	Wavelet	100.0 %	100.0 %	71.18 %

TABLE 6.6: Classification accuracy of pressure (multiple sensor) using MLP, SVM and NB for all the fault scenarios (19 classes).

6.6 Real case study of fault diagnosis considering noisy measurements

In the present section, the objective is to perform the real case study of fault diagnosis considering noisy measurements to understand the sensitivity of the ML algorithms. The noise influence in various ML algorithms have been widely studied [218]. The noise can be added into the input dataset, output dataset, training dataset, testing dataset or a combination of all of these. The generation of noise can be characterised by its distribution and where to introduce it [223]. In general, noisy data may cause biases in the learning process, making it more difficult for learning algorithms to form accurate models from the data. Therefore, developing learning techniques that effectively and efficiently deal with these types of data is a key aspect in ML.

In real case applications, datasets arrive from sensors installed in the device. However, the datasets measured contain noise, so to replicate this scenario, Gaussian white noise [224] is added to the input data of training or test datasets. The noise addition in the input data can be expressed as

$$\hat{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\epsilon}, \tag{6.1}$$

where $\boldsymbol{x}$ is the time history and $\boldsymbol{\epsilon}$ is Gaussian white noise. In order to vary the level of noise added, the Signal to Noise Ratio (SNR) in decibel (dB) [225] is utilised, and it is defined as

$$\text{SNR} = 10 \log_{10} \frac{P_{\boldsymbol{x}}}{P_{\boldsymbol{\epsilon}}}, \tag{6.2}$$

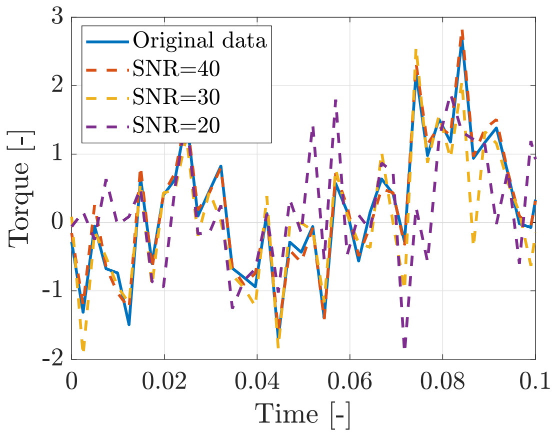
where $P_{\boldsymbol{x}}$ is the power of a signal, which is the sum of the absolute squares of its time history samples divided by the signal length,

$$P_{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i^2. \tag{6.3}$$

Similarly, $P_{\boldsymbol{\epsilon}}$ is the power of a signal and it can be written as

$$P_{\boldsymbol{\epsilon}} = \frac{1}{n}\sum_{i=1}^{n} \epsilon_i^2. \tag{6.4}$$

Figure 6.8a shows torque time histories of the original data and the noisy data obtained using Equation (6.1) for different levels of SNR. As the level of SNR increases, the noise added time signal is approaching the original data. Indeed, form Equation (6.2), when SNR $= 100$, then $P_{\boldsymbol{\epsilon}} = 10^{-10}P_{\boldsymbol{x}}$, so the noise is negligible.



(a) Torque data vs noisy data

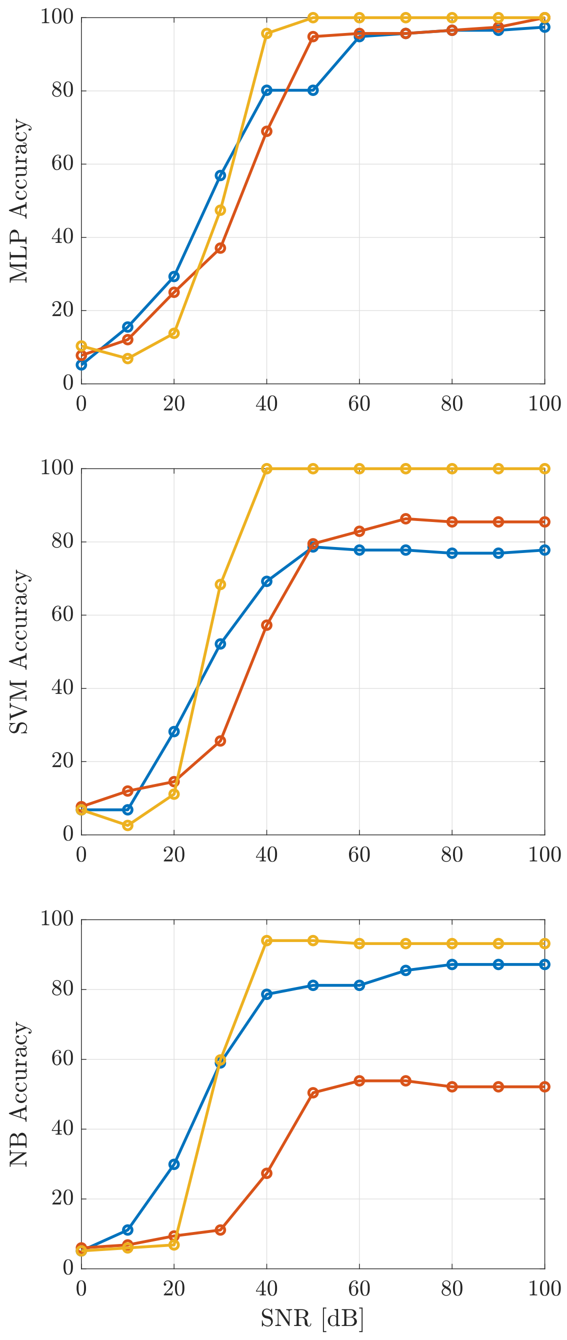
To perform fault diagnosis, consider the healthy dataset (H) and all six types of fault scenarios implemented with 18 different cases (C1, C2, C3, G1, G2, G3, S1, S2, S3, T1, T2, T3, W1, W2, W3 V1, V2, V3). These fault scenarios have been presented in Section 3.7. The variable torque is monitored for all the scenarios mentioned above. As discussed earlier, the availability of data has been increased using the noise perturbation method that is discussed in Section 5.3.1. These generated datasets are divided into a training dataset of 70% and the remaining 30% is considered as a test dataset. Both training and testing datasets are used to extract features utilising feature extraction methods shown in Section 5.5. The extracted features are standardised and used as an input dataset for training and testing of ML algorithms. The output data is considered as labels (H, C1, C2,...V2, V3). The training dataset is used to train each ML algorithms MLP, SVM and NB, which are then tested against the test dataset, where the test accuracy is calculated between the model output and the desired output for each SNR level.

In our case, two examples of fault diagnosis are performed. Firstly, in the 30% of the testing dataset, the Gaussian noise is added using Equation (6.1) that is tested against the ML model trained with a dataset without noise. Later, in the 70% of the training dataset, the Gaussian noise is added and tested against the test dataset without noise.

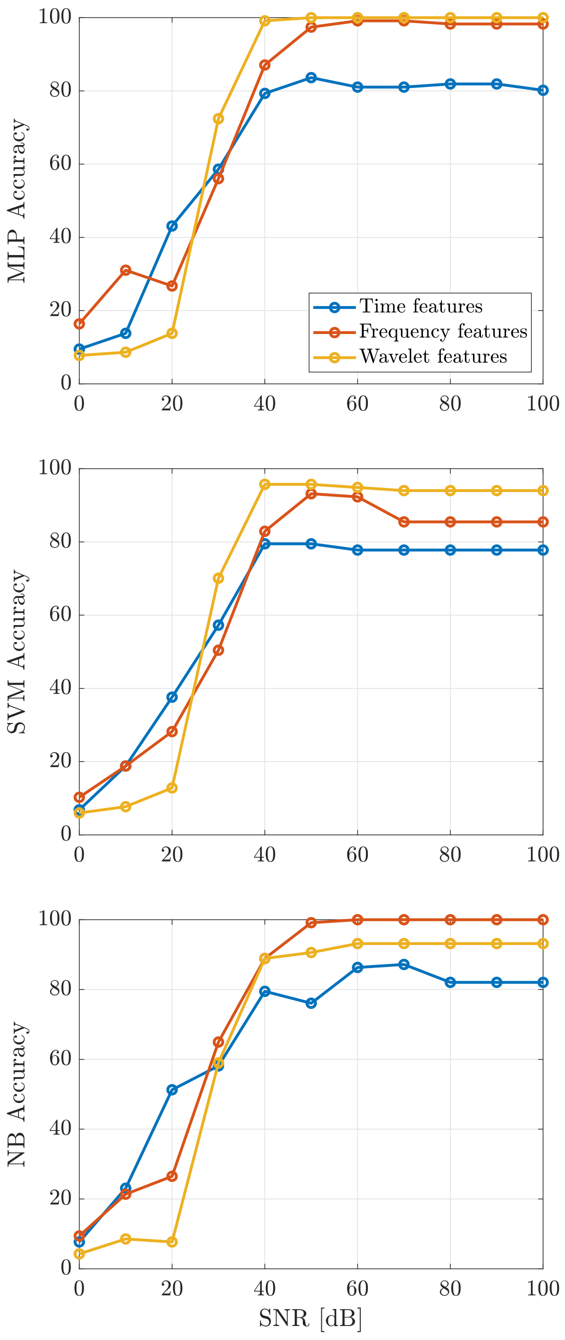
Figures 6.9a and 6.9b show the fault diagnosis accuracy of torque using MLP, SVM and NB for three different feature extraction methods, time features, frequency features and wavelet features for noise addition in the testing dataset and training dataset, respectively. All the considered cases follow the increasing trend with the increase in SNR level, i.e., decrement of the noise level in the dataset. In most of the cases, results related to wavelet features provide the highest accuracy up to the SNR level of 40. However, the frequency feature of the NB algorithm performs better when noise added to the training dataset rather than in the test dataset. For the SNR level below 40, all the algorithms with the combinations of all the feature extraction methods perform poorly.

6.7 Chapter conclusion

The present chapter has presented the numerical examples related to fault diagnosis of an external gear pump. The high-fidelity data generation in chapter 3, along with synthetic data generation through the noise perturbation method (Section 5.3.1), are used as an input dataset for feature extraction methods. These extracted time features, frequency features, and wavelet features are the training and test dataset for fault diagnosis. To begin with, optimisation of hyper-parameters for MLP, SVM and NB has been presented. All the numerical examples have been performed while adapting these parameters. Fault diagnosis has been performed considering all implemented fault scenarios (i.e. 19 classes), and a combination of fault scenarios (combining fault scenarios due to clogging and viscosity) using ML algorithms MLP, SVM, and NB for an external gear pump. Furthermore, fault diagnosis has been performed with multi-sensor information on the pressure dataset using ML algorithms. MLP and SVM compute a better classification accuracy than NB because NB suffers due to its linearity. Finally, a real case study of fault diagnosis was performed to understand the effect of white noise in the dataset and how it affects the performance of ML algorithms. High level of noise addition caused disruption in the signal that decreased the ML classification accuracy.



(a) Noise addition in test dataset

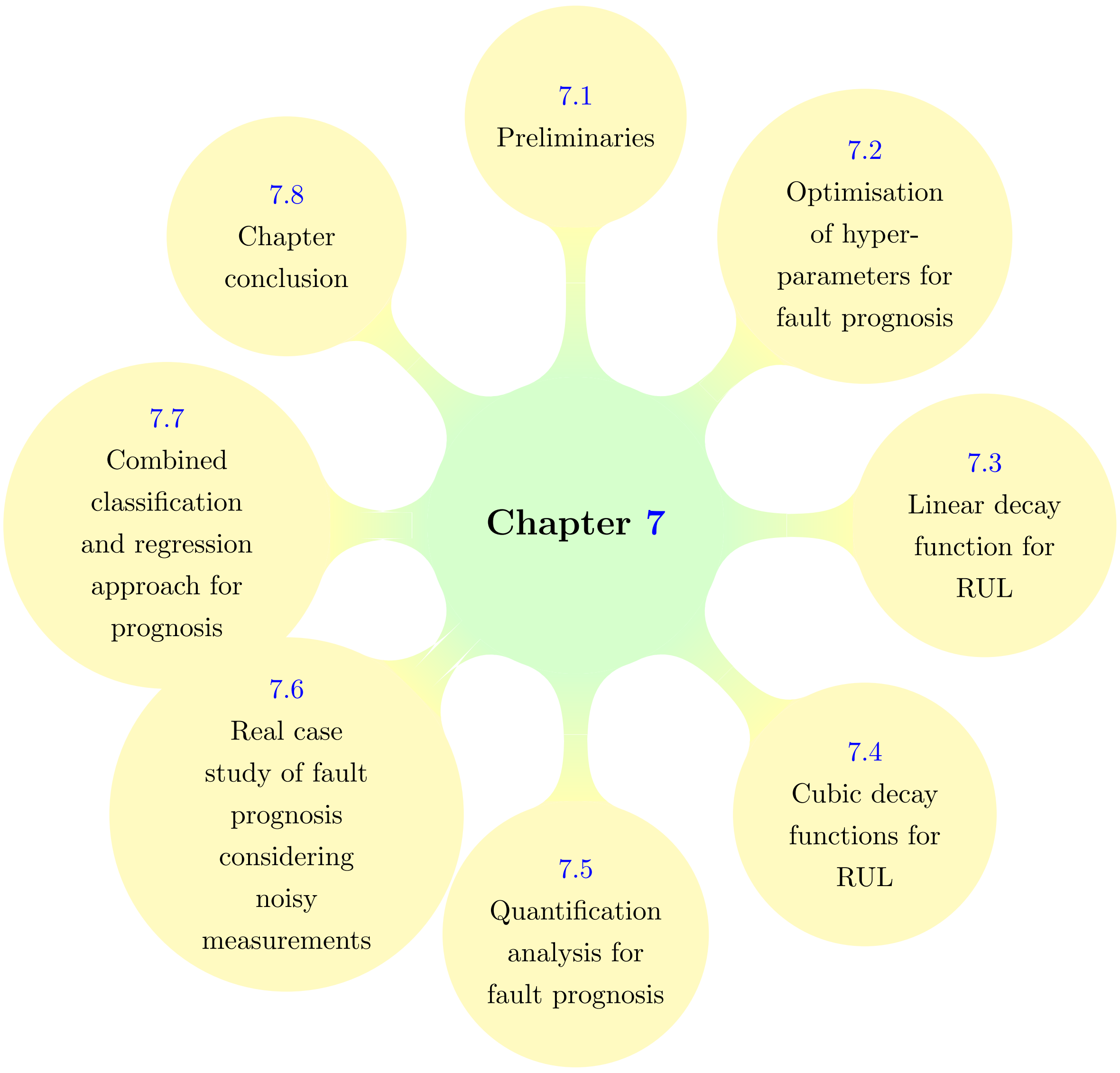


(b) Noise addition in training dataset

FIGURE 6.9: Classification accuracy of Torque using MLP, SVM and NB for time, frequency and wavelet features for noise addition in test dataset (a) and noise addition in training dataset (b).

Chapter 7

NUMERICAL EXAMPLES FOR PROGNOSIS



Chapter 7
7.1 Preliminaries
7.2 Optimisation of hyper-parameters for fault prognosis
7.3 Linear decay function for RUL
7.4 Cubic decay functions for RUL
7.5 Quantification analysis for fault prognosis
7.6 Real case study of fault prognosis considering noisy measurements
7.7 Combined classification and regression approach for prognosis
7.8 Chapter conclusion

FIGURE 7.1: Structure of Chapter 7.

7.1 Preliminaries

The objective of the present chapter is the numerical simulation of fault prognosis (prediction of the RUL) of the external gear pump by means of two alternative ML algorithms, namely MLP and SVM. These ML algorithms have been presented in Chapter 4. The overview of the designed process for fault prognosis is presented in Figure 7.2. Fault prognosis of various components of industrial equipment has been recently studied in [115, 122, 226] using various ML algorithms: Bayesian regularised Radial Basis Function NN, Gated recurrent unit and deep convolutional NN. The structure of the present chapter is presented in Figure 7.1.

ML predictive capability is strongly dependent on the availability of large sets of high-fidelity data [216]. In case of absence (or limited availability) of experimental data, these can be superseded by the use of high-fidelity in-silico data. The high-fidelity data has been generated using an accurate CFD model resembling the operations of an external gear pump in healthy and various fault conditions (clogging, radial gap variations, etc.). The aspect of high-fidelity data generation has been discussed in Chapter 3. In order to enhance the predicted response of any ML algorithm, large amounts of data are usually necessary. To greatly facilitate the gathering of data, the synthetic data generation method is employed. The noise perturbation technique is used to recreate more data from the in-silico data, and the process of generating data is presented in Section 5.3.1. The generated datasets are, in turn, used for generating degradation dataset for fault prognosis.

In the absence of run-to-failure data or the lifetime history of the equipment, the deteriorating data of the equipment component is necessary to carry out fault prognosis. In order to generate deterioration data, in the present thesis, a linear interpolation method and a cubic interpolation method are employed. These methods have been presented in Sections 5.3.2.1 and 5.3.2.2. Datasets generated from the linear and cubic interpolation methods are standardised[8] and used as input dataset for the training and testing of ML regression algorithms. The output datasets consist of RUL, derived from linear and cubic decay functions. These decay functions are presented in Section 5.3.2.

[8] Standardised refers to z-score normalisation and it expressed as $\boldsymbol{x} - \mu/\sigma$, where $\boldsymbol{x}$ is dataset and μ and σ is mean and standardisation of $\boldsymbol{x}$. Further specifications can be found in Section Section 5.4.

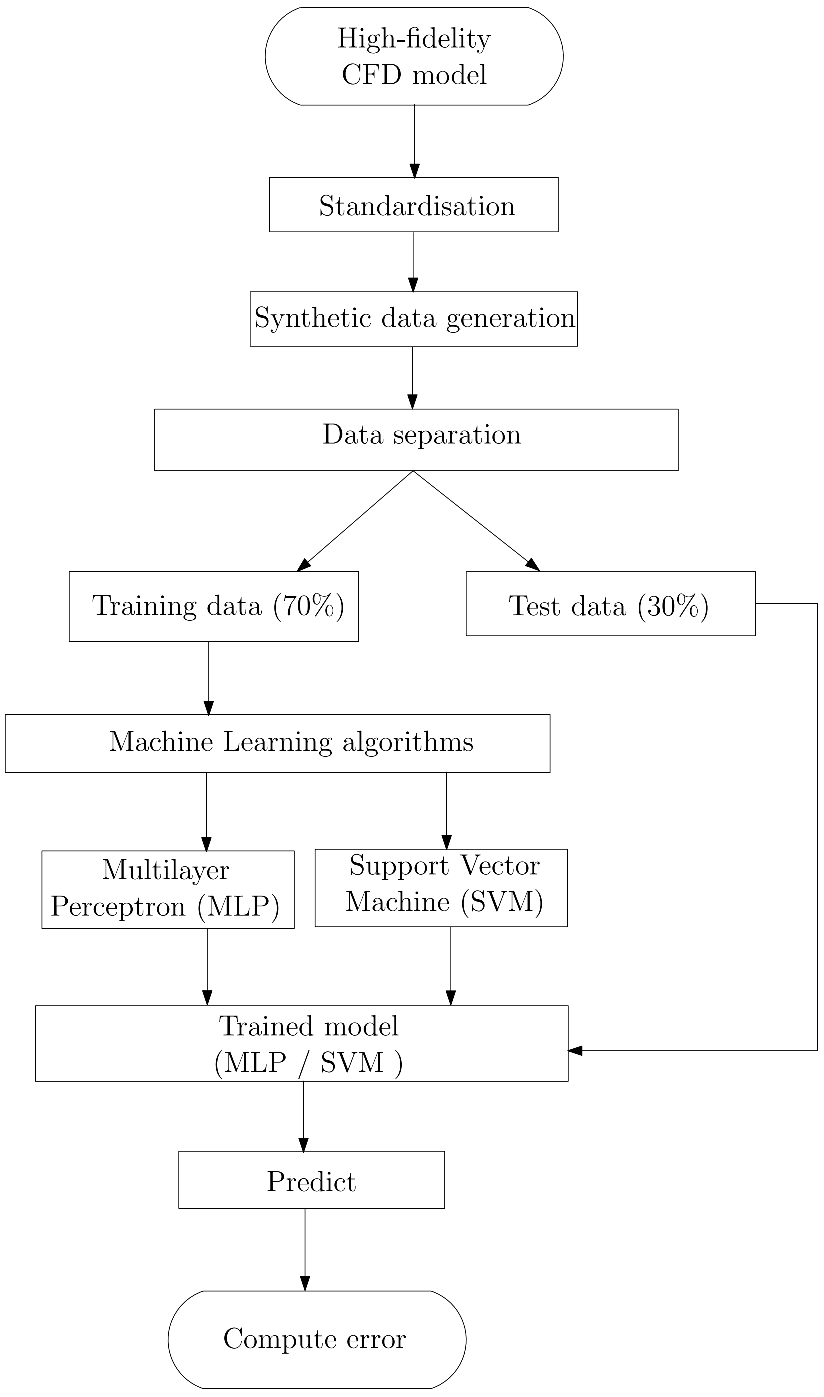

High-fidelity CFD model
Standardisation
Synthetic data generation
Data separation
Training data (70%)
Test data (30%)
Machine Learning algorithms
Multilayer Perceptron (MLP)
Support Vector Machine (SVM)
Trained model (MLP / SVM)
Predict
Compute error

FIGURE 7.2: Overview process of fault prognosis using ML regression algorithms.

Once the dataset is gathered, it is divided into a training dataset (used to fit the model) comprised of 70% of the dataset and a test dataset (used to provide an unbiased evaluation of a final model fit on the training dataset) comprised of the remaining 30% [217]. The training dataset is used to train each of the ML algorithms, which are then tested against the test dataset, and the Mean Squared Error (MSE) of the model is computed using Equation (4.2). All the examples in this chapter will follow the same procedure as discussed so far.

In addition, in order to understand the ML behaviour with respect to an increase in the number of samples in the training dataset, quantification analysis is employed [227]. Next, to recognise the sensitivity of ML algorithms to an increase of noise level in the training dataset, noise influence analysis is considered [218]. Finally, to achieve a better prediction error, a combined classification and regression approach is implemented to perform prognosis. These analyses will be presented in depth.

The chapter is summarised as follows: (1) Optimisation of hyper-parameters for fault prognosis using ML regression algorithms is presented in Section 7.2. (2) The RUL prediction for three different fault scenarios, namely fault scenarios due to radial gap degradation, fault scenarios due to axial gap degradation, and fault scenarios due to speed variations with a linear decay RUL function is presented in Section 7.3. (3) The RUL prediction of fault scenarios due to radial gap degradation with cubic decay RUL functions is discussed in Section 7.4. (4) The quantification analysis for fault prognosis is discussed in Section 7.5. (5) A real case study of fault prognosis considering noisy measurements is presented in Section 7.6. (6) A combination of the classification and regression approach for fault prognosis is discussed in Section 7.7. (7) The chapter conclusion is presented in Section 7.8.

7.2 Optimisation of hyper-parameters for fault prognosis

The objective of the present section is to describe the methodology used for the optimisation of the hyper-parameters to perform fault prognosis. As an example, a standardised torque dataset from fault scenarios due to radial gap degradation is considered as an input dataset for this analysis. In order to increase the availability of data to train the ML algorithm, the synthetic data generation method is employed (see Section 5.3.1). Furthermore, to achieve degradation behaviour in the dataset, a cubic interpolation method (see Section 5.3.2.2) is utilised. The output dataset considered is a linear decay function, described in Section 7.3. Similar analyses have been carried out considering pressure and flow rate datasets.

A brief overview of the optimisation of hyper-parameters has been discussed in Section 4.6. Firstly, the optimisation of hyper-parameters for SVM regression is discussed. Next, optimisation of hyper-parameters for MLP regression is discussed. In the current section, the dataset has been separated into training and validation datasets. The validation dataset is used to provide an unbiased evaluation of a model fit on the training dataset whilst characterising the model hyper-parameters. The holdout sample validation method [219] is used to hold a 30% of the dataset for validation purposes.

Hyper-parameter optimisation for SVM regression

For SVM regression, four hyper-parameters are considered, namely kernel function, kernel parameter (Section 4.3.4), regularisation parameter (C in Equation (4.23)) and epsilon (margin of tolerance). To ensure the optimal choice of hyper-parameters for SVM, it is necessary to optimise these parameters. This is performed in a staggered manner[9], i.e., by fixing first the kernel function, the regularisation parameter and epsilon, and optimising then the kernel parameter heuristically. Kernel functions, namely linear kernel, Gaussian kernel and polynomial kernel, are considered for this analysis. The regularisation parameter is considered as a default value of C=1. Even though the regularisation parameter value can range between $[10^{-3}, 10^{3}]$, C=1 has been chosen as it is sufficient to provide very good results [194]. The epsilon value is considered as a default value of 1 [194].

Following the fault prognosis procedure discussed in Section (7.1), fault prognosis is performed for the desired dataset. As a result, the regression MSE is calculated using Equation (4.2). Figure 7.3 shows the SVM prediction error of the torque for the different kernel functions: linear, Gaussian and polynomial. Considering both training and validation MSE, linear and polynomial kernel display considerably smaller values of MSE when compared to the Gaussian kernel function. Following these results, for all the examples, a polynomial kernel will be used.

[9] This kind of optimisation can also be referred as trial and error method [192, 193].

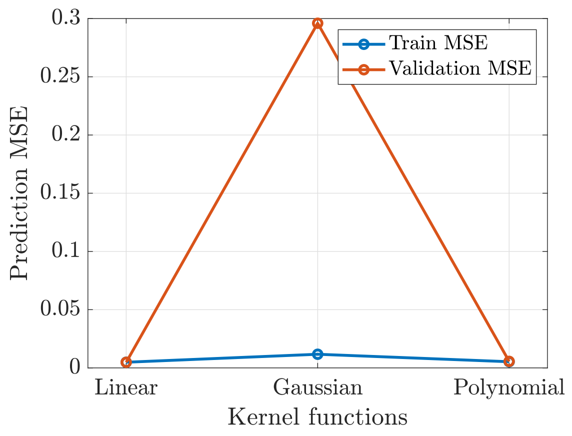


FIGURE 7.3: SVM prediction error of torque for different kernel functions, linear, Gaussian and polynomial.

Hyper-parameter optimisation for MLP regression

For MLP regression, similar to MLP classification, three hyper-parameters are considered, namely the activation functions, the number of hidden layers and the number of hidden neurons in each hidden layer, which are discussed in Section 4.4. The activation function is considered as Relu (Figure 4.12c) for each MLP layer. The number of hidden layers are optimised in a staggered manner, i.e., by fixing the activation function and the number of hidden layers, the number of hidden neurons is optimised using 2^n method, where $n = 1, \dots, 8$. The 2^n method is based on the forward approach method [221] and is carried out by fixing the number of hidden neurons, for which 2^n neurons per hidden layer are considered to speed up the identification of the precise number of neurons in the layer. When training the MLP network, the backpropagation algorithm is used to optimise the updating of the weights, according to the loss or error function specified when compiling the model. The backpropagation algorithm requires that the network is trained for a specified number of epochs to the training dataset. This aspect has been discussed in Section 4.4. It is common practice to show the model loss with respect to the number of epochs and this plot can detect if the model is overfitting, underfitting or suitably fitting to the training dataset [4]. In addition, during the training, *Early Stopping* argument is used to stop the training when the chosen performance measure stops improving [222].

Following the fault prognosis procedure discussed in Section 7.1, the prognosis is performed using the ML algorithm MLP and the prediction error is computed using the MSE. The MSE is calculated between ML predicted values and true output values. Figure 7.4 shows the error plot as a function of the number of hidden layers and hidden neurons using MLP. Figure 7.4a shows that the first layer provides the optimal result as the error increases when considering more layers. This one layer is sufficient to perform this analysis. Within this layer, the number of hidden neurons that must be used to perform fault prognosis is shown in Figure 7.4b. It can be seen that one hidden layer with two neurons is sufficient to perform this analysis. A similar method will be used to optimise the number of hidden neurons and hidden layers for the prognosis simulations shown in the following sections.

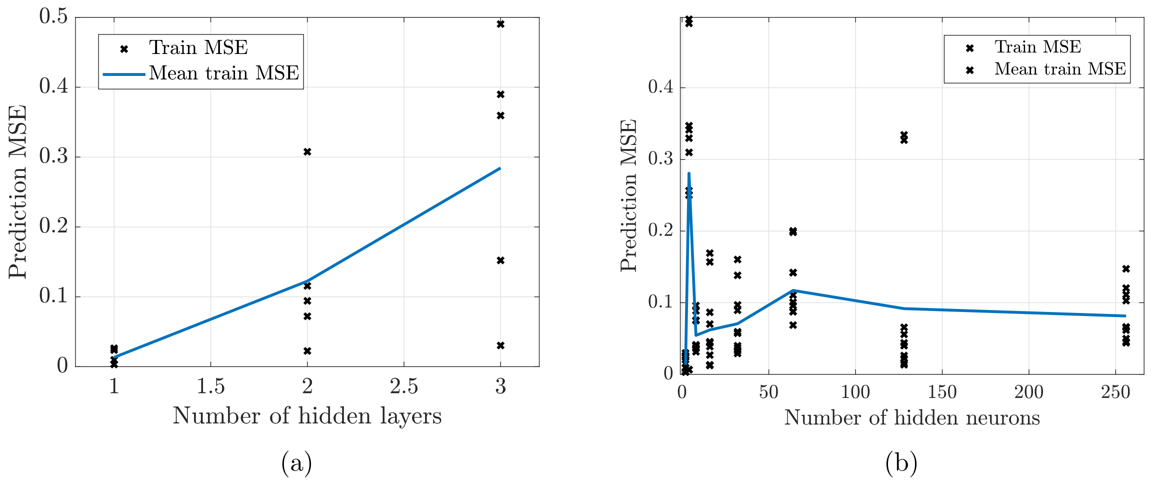
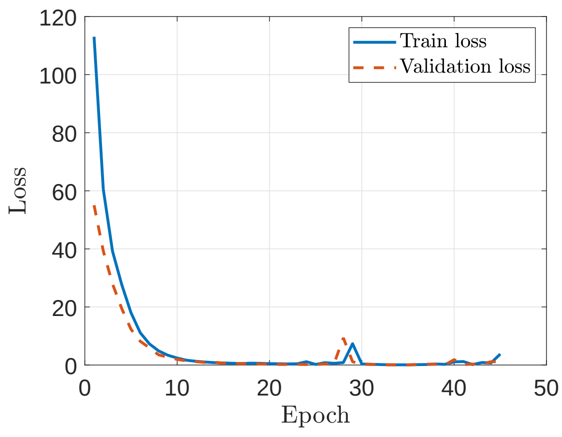
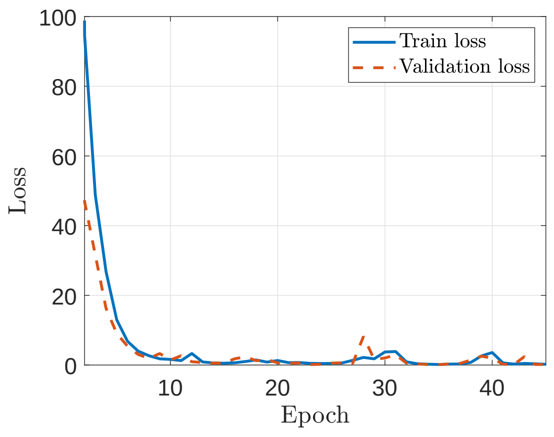


FIGURE 7.4: Prediction MSE of the torque dataset with respect to (a) the number of hidden layers (1, 2, 3) and (b) the number of hidden neurons (2, 4, 8, 16, 32, 64, 128, 256) for 1 hidden layer using MLP.

As discussed earlier, the model loss shows how well the model manages the training and avoids overfitting. This is shown in Figure 7.5a. The x-axis shows the number of epochs. The model has good performance on both train and validation datasets and avoids to overfit when one hidden layer is considered with early stopping. Figure 7.5b shows the model loss when two layers are considered with early stopping. The train and validation loss curves are slightly drifted compared to Figure 7.5a, yet it does not show overfitting. The number of epochs can be stopped during the training, using the early stopping option, when the chosen performance stops improving. The early stopping criteria minimises the computational time up to 50% when the optimisation is carried out without early stopping.



(a) one hidden layer



(b) Two hidden layers

FIGURE 7.5: Model loss on training and validation datasets. Model trained with one layer of neurons (a) and two layers of neurons (b) using MLP.

7.3 Linear decay function for RUL

In the present section, the objective is to perform the fault prognosis considering a linear decay function. Let us consider that the life of the gear pump is 100-time units, i.e., the RUL ranges from 100 to 1. Figure 7.6 shows the linear decay function, considered as the true RUL function, where the x-axis denotes time units, and the y-axis denotes the RUL of the equipment. The RUL is considered as the output dataset for fault prognosis. The

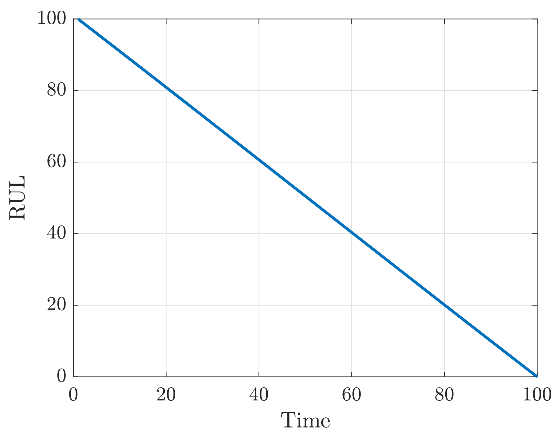


FIGURE 7.6: Representation of true RUL data as a linear decay function.

input datasets are based on three different fault scenarios, namely fault scenario due to radial gap degradation, fault scenario due to axial gap degradation and fault scenario due to speed variations. Each scenario will be discussed in the following.

Fault scenarios due to radial gap degradation

The present example, the RUL prediction of fault scenarios due to radial gap degradation using ML algorithms, MLP and SVM is described. The pressure, flow rate, and torque data from the fault scenario due to radial gap degradation are considered for the analysis.

Following the fault prognosis procedure discussed in Section 7.1, the fault prognosis is performed using the ML algorithms, MLP and SVM, and the prediction error is computed by the MSE. Table 7.1 shows the MSE of pressure, flow rate, and torque of fault scenario due to radial gap degradation using ML algorithms, MLP and SVM utilising the linear interpolation method (a) and cubic interpolation method (b). In all the cases, MLP performs relatively better than SVM. The dataset created with the cubic interpolation method provides lower error when compared to the dataset created with the linear interpolation method. This occurs because the linear interpolation method is used between intervals of two time signal whereas cubic interpolation method is used between four time signal. So, the dataset created using the cubic interpolation method gives the degradation behaviour is smoother than the dataset created using the linear interpolation method.

Figures 7.7, 7.8, and 7.9 show the prediction error of the fault scenarios due to radial gap degradation using the linear interpolation method (a) and the cubic interpolation method (b) for pressure, flow rate, and torque, respectively. The prediction error is computed via the difference between predicted RUL and the true RUL for a number of samples in the test set. For the pressure and the flow rate, dataset generated from the cubic interpolation with the MLP algorithm provides optimal results. On the other hand, in case of torque, dataset generated with the cubic interpolation method with MLP and SVM algorithm provide similar results.

Fault scenarios due to axial gap degradation.

In the present example, the RUL prediction of fault scenarios due to axial gap degradation using ML algorithms, MLP and SVM is described. The variables pressure, flow rate and torque extracted from the fault scenarios due to axial gap degradation, are considered for this analysis.

Following the fault prognosis procedure discussed in Section 7.1, the fault prognosis is performed using the ML algorithms, MLP and SVM, while the prediction error is computed by MSE. Table 7.2 shows the prediction MSE of pressure, flow rate and torque for fault scenarios due to axial gap degradation using ML algorithms, MLP and SVM

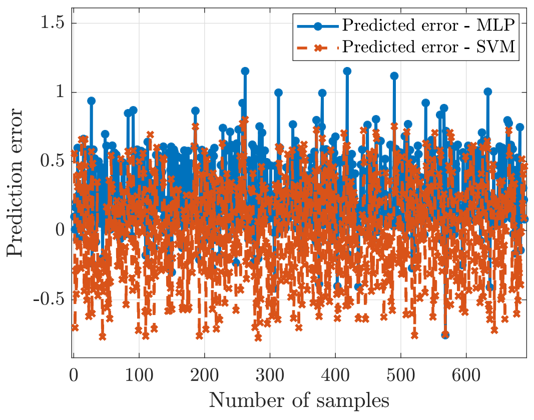
	MLP	SVM
Pressure	0.0563	0.0700
Flow rate	0.0367	0.0422
Torque	0.0711	0.0889

(a) Linear interpolation method

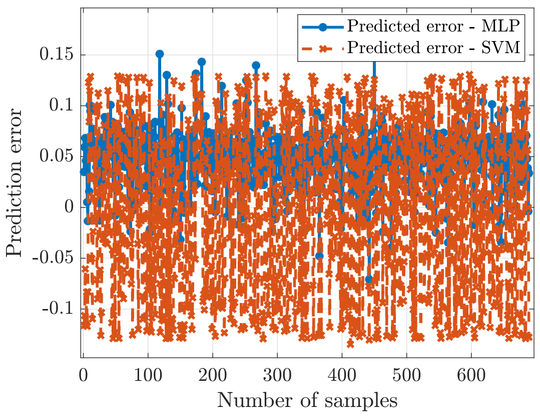
	MLP	SVM
Pressure	0.00041	0.0048
Flow rate	0.00035	0.0059
Torque	0.0024	0.0056

(b) Cubic interpolation method

TABLE 7.1: MSE of pressure, flow rate and torque using MLP and SVM for fault scenarios due to radial gap degradation using the linear interpolation method (a) and cubic interpolation method (b).



(a)

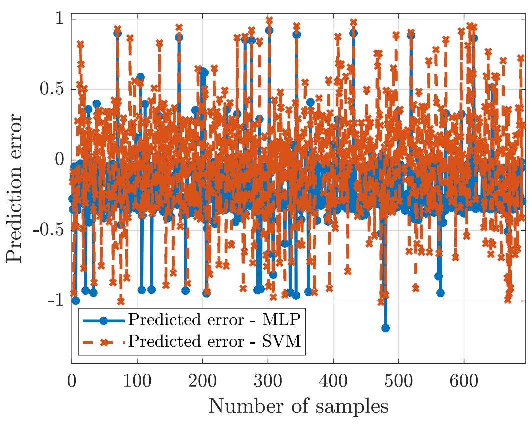


(b)

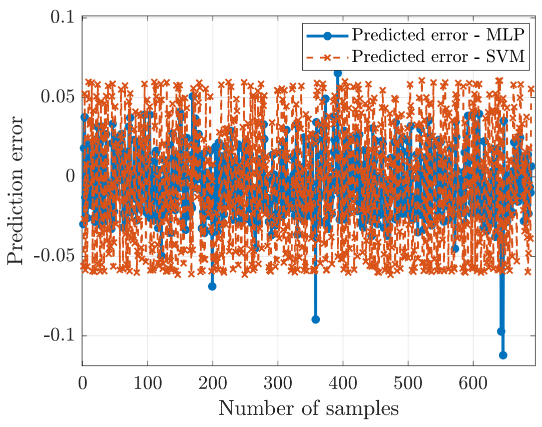
FIGURE 7.7: Prediction error for the pressure due to radial gap degradation using the linear interpolation method (a) and cubic interpolation method (b). The prediction error is computed via the difference between predicted RUL and true RUL for the number of samples in the test dataset.

utilising the dataset generated from the linear interpolation method (a) and the cubic interpolation method (b). The dataset created by the cubic interpolation method provides lower error compared to the dataset created by the linear interpolation method. This occurs because, using the cubic interpolation method, the interpolation is obtained between for time signal rather than two time signals using linear interpolation method, so smoother the degradation behaviour.

As displayed in the previous examples, the prediction error has been carried out for pressure, flow rate, and torque. The behaviour of the prediction error is similar to the previous section. Figure 7.10 shows the prediction error of torque for the fault scenarios due to axial gap degradation using the linear interpolation method (a) and the cubic interpolation method (b).

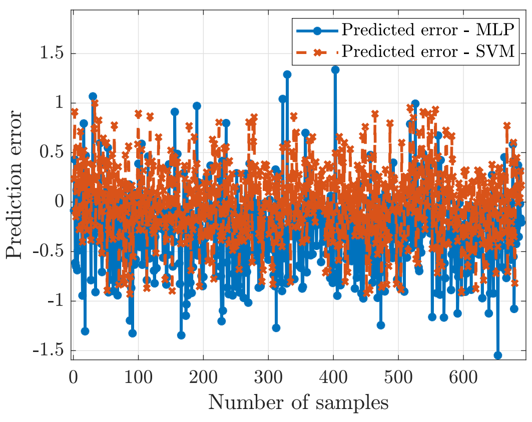


(a)

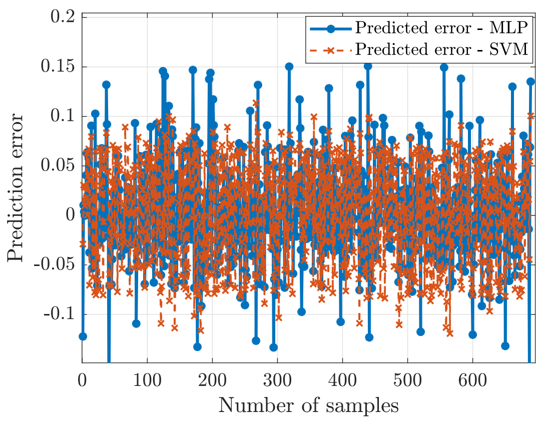


(b)

FIGURE 7.8: Prediction error of flow rate due to radial gap degradation using the linear interpolation method (a) and cubic interpolation method (b). The prediction error is computed via the difference between predicted RUL and true RUL for the number of samples in the test dataset.



(a)



(b)

FIGURE 7.9: Prediction error of torque due to radial gap degradation using the linear interpolation method (a) and cubic interpolation method (b). The prediction error is computed via the difference between predicted RUL and true RUL for the number of samples in the test dataset.

	MLP	SVM
Pressure	0.0401	0.0406
Flow rate	0.0346	0.042
Torque	0.0961	0.1087

(a) Linear interpolation method

	MLP	SVM
Pressure	0.00077	0.0036
Flow rate	0.00011	0.00064
Torque	0.00223	0.0059

(b) Cubic interpolation method

TABLE 7.2: MSE of pressure, flow rate and torque using MLP and SVM for fault scenarios due to axial gap degradation using the linear interpolation method (a) and cubic interpolation method (b).

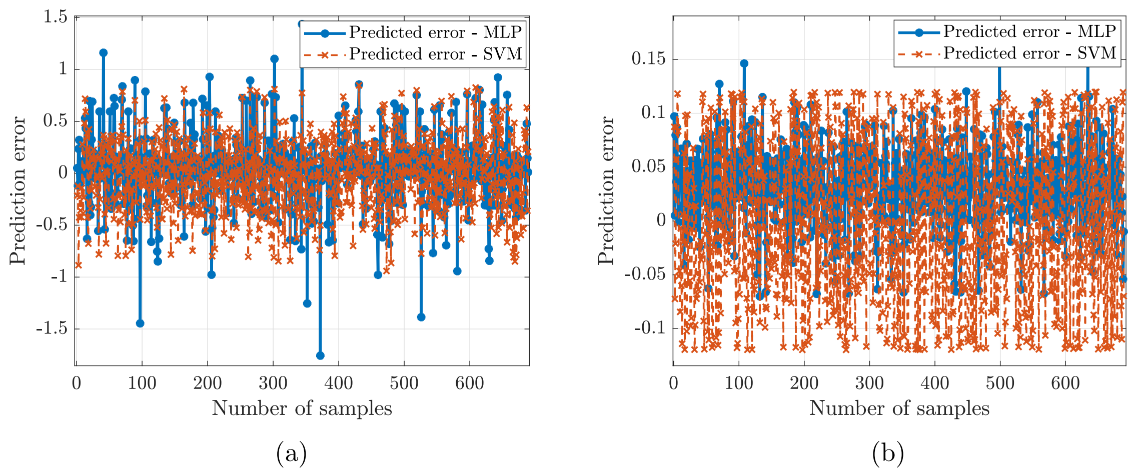


FIGURE 7.10: Prediction error of torque due to radial gap degradation using the linear interpolation method (a) and cubic interpolation method (b). The prediction error is computed via the difference between predicted RUL and true RUL for the number of samples in the test dataset.

Fault scenarios due to speed variations

In the present example, the RUL prediction of fault scenarios due to speed variations will be discussed. The pressure, flow rate, and torque extracted from the fault scenarios due to speed variations, are considered for this analysis.

Following the fault prognosis procedure discussed in Section 7.1, the fault prognosis is performed using the ML algorithms, MLP and SVM, and the prediction error is computed by the MSE. Table 7.3 show the MSE of pressure, flow rate, and torque for fault scenarios due to speed variations using ML algorithms, MLP and SVM utilising the synthetic data generation approaches, the linear interpolation method (a) and the cubic interpolation method (b). For MLP and SVM, the dataset created using the cubic interpolation method provides lower error compared to the dataset created using the linear interpolation method. However, in case of torque, SVM provides a lower prediction error than MLP. This is shown in Figure 7.11, the prediction error of torque for the fault scenarios due to speed variations using the linear interpolation method (a) and cubic interpolation method (b).

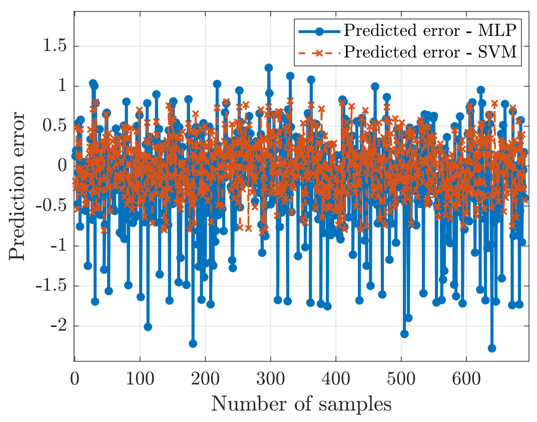
	MLP	SVM
Pressure	0.03857	0.0472
Flow rate	0.03746	0.0490
Torque	0.1401	0.1211

(a) Linear interpolation method

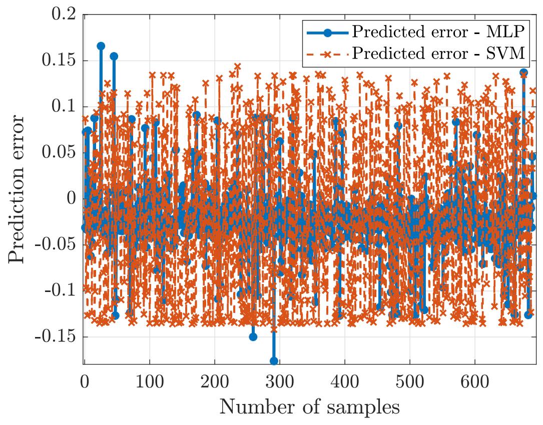
	MLP	SVM
Pressure	0.000752	0.0026
Flow rate	0.00023	0.0084
Torque	0.00062	0.00052

(b) Cubic interpolation method

TABLE 7.3: MSE of pressure, flow rate and torque using MLP and SVM for fault scenarios due to speed variations using the linear interpolation method (a) and cubic interpolation method (b).



(a)



(b)

FIGURE 7.11: Prediction error of torque due to radial gap degradation using the linear interpolation method (a) and cubic interpolation method (b). The prediction error is computed via the difference between predicted RUL and true RUL for the number of samples in the test dataset.

7.4 Cubic decay functions for RUL

In the present section, the objective is to perform fault prognosis considering the cubic decay functions. Let us consider that life of the gear pump is 100-time units, i.e., the RUL ranges from 100 to 1. Fault prognosis considering the linear decay function has been presented in Section 7.3. Subsequently, the prognosis is performed considering cubic decay functions as the RUL. Figure 7.12 shows the true RUL considering cubic functions,

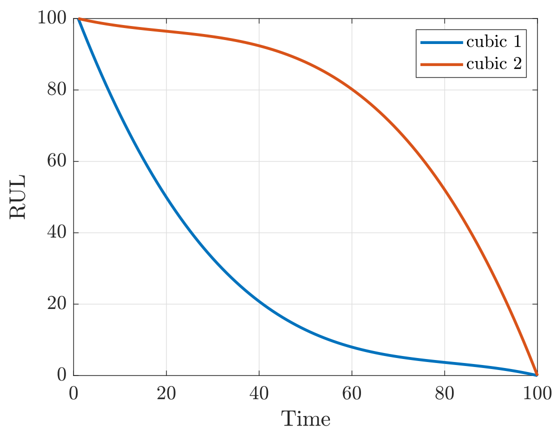


FIGURE 7.12: Representation of true RUL data, and the derived cubic functions are presented. The RUL decreases as the equipment life increases.

where x-axis denotes time units, and y-axis denotes the RUL of an equipment. The RUL is considered as output dataset for fault prognosis. Whereas the input dataset is from the fault scenario due to radial gap degradation. The pressure, flow rate, and torque, extracted from the fault scenarios due to radial gap degradation, are considered for this analysis.

Following the fault prognosis procedure discussed in the introduction (7.1) of this chapter, the fault prognosis is performed using the ML algorithms, MLP and SVM, and the prediction error is computed by the MSE. Tables 7.4 and 7.5 show the MSE of pressure, flow rate, and torque using ML algorithms, MLP and SVM based on the synthetic data generating methods, the linear interpolation method and the cubic interpolation method, respectively. The results with cubic decay function are similar to the one of the linear decay function and the dataset interpolated cubically performs better than the dataset created from the linear interpolation method.

	MLP	SVM
Pressure	0.0883	0.1055
Flow rate	0.0726	0.0813
Torque	0.2775	0.8562

(a) Cubic decay function 1

	MLP	SVM
Pressure	0.1127	0.1601
Flow rate	0.0872	0.0874
Torque	0.2424	0.1703

(b) Cubic decay function 2

TABLE 7.4: MSE of pressure, flow rate and torque using MLP and SVM for fault scenarios due to radial gap degradation using the linear interpolation method and cubic decay functions.

	MLP	SVM
Pressure	0.0037	0.0061
Flow rate	0.0026	0.0066
Torque	0.0048	0.0055

(a) Cubic decay function 1

	MLP	SVM
Pressure	0.00041	0.0063
Flow rate	0.00023	0.0066
Torque	0.0022	0.0071

(b) Cubic decay function 2

Table 7.5: MSE of pressure, flow rate and torque using MLP and SVM for fault scenarios due to radial gap degradation using the cubic interpolation method and cubic decay functions.

7.5 Quantification analysis for fault prognosis

The quantification analysis is employed to understand the behaviour and the performance of ML algorithms with respect to the number of samples used for training. The dataset considered for the analysis of RUL prediction contains 2300 samples. In which 10%, 30%, 50%, 70%, and 90% (230, 690, 1150, 1610, and 2070 samples) are used to train the ML algorithms, MLP and SVM, that are tested against the test dataset consisting of 230 samples. For each of these samples, fault prognosis is performed to identify the behaviour of ML algorithms. A similar analysis to underline the behavioural effect on ML classification algorithms based on the number of samples can be found in [227].

The dataset considered for this analysis is fault scenarios due to radial gap degradation. The pressure, flow rate, and torque, extracted from the fault scenarios due to radial gap degradation, are considered for this analysis. The dataset generated using cubic interpolation method used as an input dataset for ML algorithms has been presented in Section 5.3.2.2. The output dataset is generated using linear decay function Section 7.3.

Following the fault prognosis procedure discussed in Section 7.1, the fault prognosis is performed using the ML algorithms, MLP and SVM, and the prediction error is computed by the MSE. Figures 7.13a and 7.13b show the quantification analysis of pressure, flow rate, and torque for fault scenarios due to radial gap degradation using ML algorithms, MLP and SVM, utilising the dataset generated from the linear interpolation method and the cubic interpolation method, respectively. In most cases, both ML algorithms follow a linear trend, depicting that, an increase in the number of samples causes the machine learning algorithms to provide better results (i.e., low prediction error). However, data generated using the linear interpolation method shows inconsistency in the case of pressure when using both ML algorithms, which is shown in Figure 7.13a. This could be because 2300 samples are not necessary to provide the optimal results, and1750 samples are sufficient to perform the ML training.

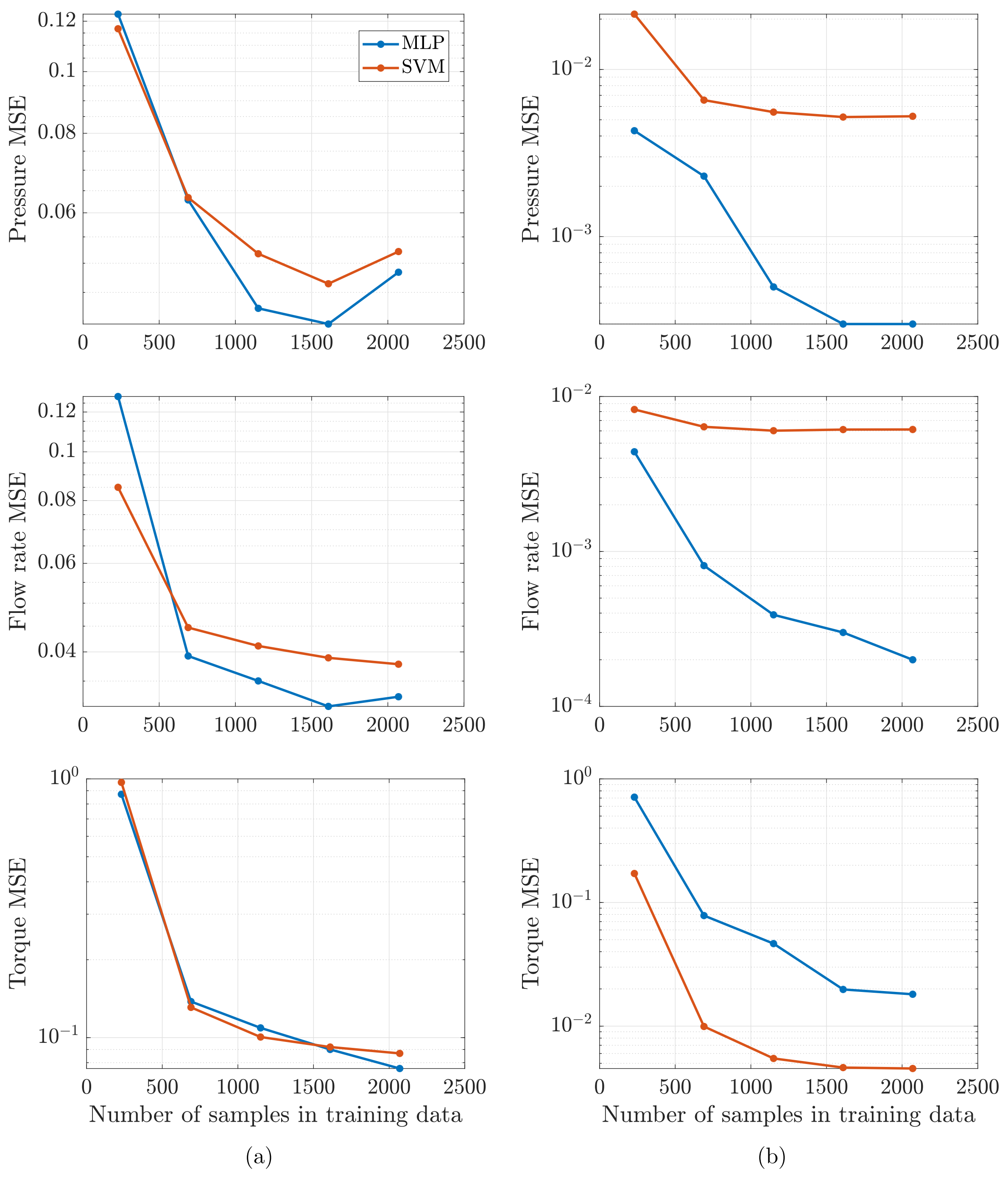


FIGURE 7.13: Quantification analysis of prognosis of fault scenarios due to radial gap degradation using the linear interpolation method (a) and the cubic interpolation method (b).

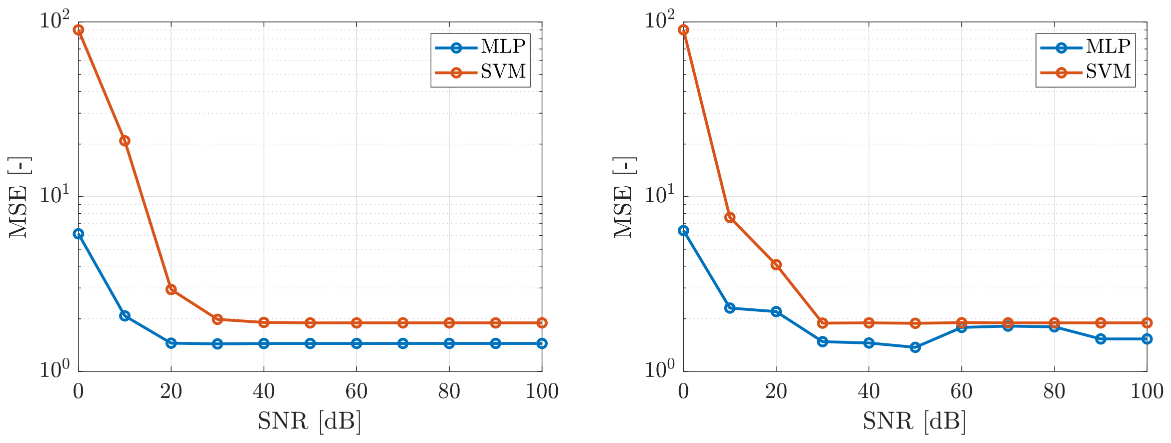
7.6 Real case study of fault prognosis considering noisy measurements

In the present section, the objective is to perform a real case study of fault prognosis considering noisy measurements to understand the sensitivity of the ML algorithms. The present section follows the same approach as in Section 6.6 to represent the real case application for fault prognosis.

The dataset considered for this analysis is fault scenarios due to radial gap faulty degradation. The variable torque is extracted from the fault scenarios due to radial gap degradation, is considered for this analysis. The degradation dataset generated using the cubic interpolation method is used as an input dataset for ML algorithms and this method has been presented in Section 5.3.2.2. The output dataset is generated using linear decay function with RUL between 1 - 100 time units (refer to Section 7.3). These datasets are divided into 70% of training and the remaining 30% of the testing dataset. The training and testing datasets are standardised and used for ML training. The training dataset is used to train each ML algorithms, MLP and SVM, which are then tested against the test dataset with noise, and the prediction MSE of the model is computed using Equation (4.2) between ML predicted values and true output values.

Similar to Section 6.6, the Gaussian white noise is added to the 30% of the testing dataset that is tested against the ML model trained with a dataset without noise. Conversely, the Gaussian white noise is added to the 70% of the training dataset and tested against the test dataset without noise.

Figures 7.14a and 7.14b show the MSE of torque using MLP and SVM for the prediction of RUL for noise addition in the testing dataset and training dataset, respectively. In both cases, the prediction error follows a decreasing trend in the beginning then a constant trend when there is an increase in the SNR level, i.e., a decrease in noise. For overall prediction, MLP performs better than SVM. However, both algorithms perform poorly for SNR levels below 30. This shows that prognosis is more sensible to noise than diagnosis.



(a) Noise addition in test dataset (b) Noise addition in training dataset

FIGURE 7.14: MSE of torque using MLP and SVM for noise addition in test dataset (a) and noise addition in test dataset.

7.7 Combined classification and regression approach for prognosis

The objective of the present section is to combine the classification and regression approach to perform prognosis of a gear pump. The idea is to first classify the dataset into smaller groups, before training the ML algorithms for classification and regression. In a practical situation, when new data arrives from a piece of equipment, it can be first classified to detect which fault has occurred, then that particular fault dataset can be used to find the RUL of the equipment. Modelling of the problem is explained in Figure 7.15.

For this analysis, the dataset of fault scenarios due to radial gap degradation (refer to section 3.7) is considered. The pressure dataset is extracted from the fault scenario due to radial gap degradation. The dataset generated using the cubic interpolation method is used as input dataset for ML algorithms and this method has been presented in Section 5.3.2.2. The output dataset is generated using linear decay function, similar to Section 7.3.

Following the fault prognosis procedure discussed in Section 7.1, the fault prognosis is performed using the ML algorithms, MLP and SVM. The output dataset is the RUL that ranges from 100-time units to 1-time unit, as shown in Figure 7.6. The training

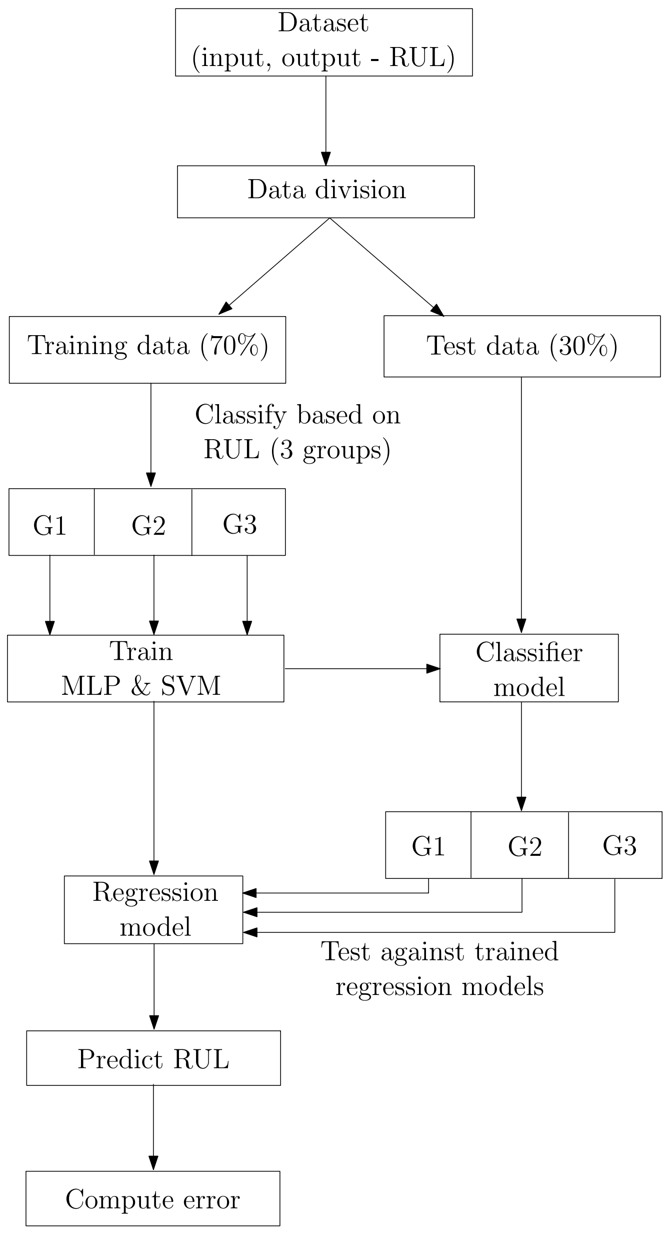


FIGURE 7.15: Problem modelling

dataset is then divided into the dataset into three subgroups based on RUL i.e., a dataset between 100 - 67-time units, 66 - 64-time units and 33 - 1-time unit. These subgroups are called G1, G2 and G3, respectively. The subgroups G1, G2 and G3 are used to train the ML algorithms, MLP and SVM for both classification and regression. The test dataset is tested against the ML classification model to determine the subgroup under which the test dataset would be categorised. Based on the results of ML classification, the RUL prediction is performed in the respective subgroup using the trained ML regression model. The final prediction error is computed using MSE for each individual subgroup.

Table 7.7 and 7.6 shows the MSE of the test dataset against the ML prediction where prior to classification of the training dataset provides a lower prediction error compared to when the entire dataset is considered using SVM. However, classifying the training dataset prior to training the MLP algorithm does not provide a lower prediction error. It remains in the same range of results when compared to the entire dataset. The MLP has a better optimisation range compared to SVM, hence, with (or without) classifying, it provides the same results.

Test dataset based on	Training G1 (RUL-100 to 67)	Training G2 (RUL-66 to 34)	Training G3 (RUL-33 to 1)
G1	9.941e-07		
G2		4.8665e-07	
G3			5.0305e-08

TABLE 7.6: MSE of pressure using SVM for fault scenarios due to radial gap degradation. RUL is in time units.

Test dataset based on	Training G1 (RUL-100 to 67)	Training G2 (RUL-66 to 34)	Training G3 (RUL-33 to 1)
G1	0.00161		
G2		0.0048	
G3			0.00067

TABLE 7.7: MSE of pressure using MLP for fault scenarios due to radial gap degradation. RUL is in time units.

7.8 Chapter conclusion

The present chapter has presented some numerical examples related to fault prognosis (i.e. predicting the RUL) of an external gear pump. The high-fidelity data generation in chapter 3, along with synthetic data generation through the noise perturbation method (Section 5.3.1), are used as a primary dataset for the prognosis. In order to achieve a degradation behaviour in the dataset, the linear interpolation (Section 5.3.2.1) and the cubic interpolation (Section 5.3.2.2) methods are used, which are considered for training and testing dataset for fault prognosis. At the beginning of this chapter, optimisation of the hyper-parameters for MLP and SVM has been elaborated. All the numerical examples of fault prognosis have been performed adapting these parameters. A fault prognosis has been performed for the fault scenarios due to radial gap degradation, axial gap degradation and speed variations using ML algorithms, MLP and SVM for an external gear pump considering the linear decay function as RUL. Subsequently, fault prognosis has been performed for the fault scenarios due to radial gap degradation considering the cubic decay function as RUL. As a result, the dataset created using the cubic interpolation method provides better results compared to the dataset generated using the linear interpolation method, and comparing ML algorithms, in most cases, MLP provides better results. However, SVM also provides comparatively good prediction results.

The quantification analysis has been performed to understand the effect of the performance of ML algorithms on the number of samples given to training the algorithms. As a result, the MSE decreases with an increase in the number of samples. A real case study of fault prognosis was performed to understand the effect of white noise in the dataset and how it affects the performance of ML algorithms. Finally, a combination of classification and regression algorithms has been utilised to perform prognosis. This depicts that classification of the dataset prior to the prediction of RUL, using the SVM algorithm, can provide lower prediction error.

Chapter 8

CONCLUSION AND REMARKS

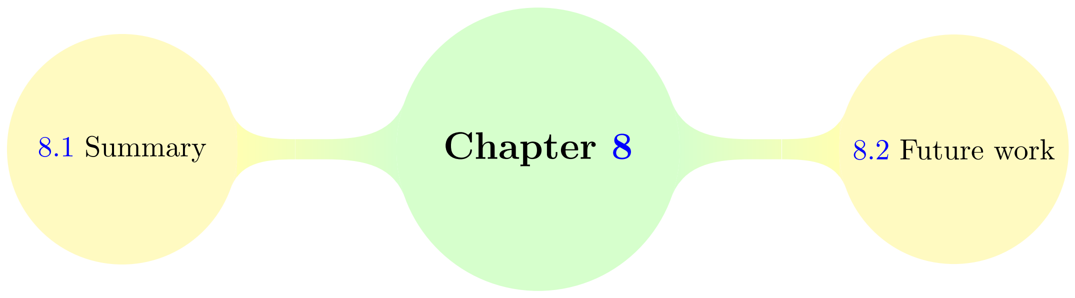


FIGURE 8.1: Structure of Chapter 8.

8.1 Summary

The present thesis has presented a fault diagnosis and fault prognosis strategy with the application of ML algorithms on the FG304 series external gear pump. In this regard, the main novelty of this thesis is the generation of high-fidelity data for an external gear pump, using CFD models and synthetic data generation methods, and a framework of Predictive Maintenance strategy with the application of ML algorithms. In general, data generation methods provide a long-term solution to the problem of insufficient or unavailable data and enable further investigation via ML or other applicable data analysis techniques. Moreover, these methods can be employed where data privacy is a significant concern [223]. The aim of the present doctoral thesis has been the design and implementation of an effective and robust approach to control the condition and to predict the RUL of an external gear pump. To achieve this objective, the methods employed in this thesis have been as follows:

High-fidelity data generation
Due to the unavailability of a dataset for an external gear pump, high-fidelity data have been generated using CFD modelling, to create an accurate physical model of the gear pump. Following the work of [136, 137, 165, 228], the CFD model of the external gear pump is simulated including turbulence and cavitation model using the 3D proprietary commercial software PumPLinx. The CFD model has also been used to recreate a variety of working conditions for the pump. These high-fidelity CFD results are validated against experimental results and their accuracy has been shown. Physical variables such as *pressure, flow rate* and *torque* have been extracted from the CFD simulations. The ensuing high-fidelity datasets are utilised as the primary data to perform fault diagnosis and fault prognosis.

Synthetic data generation methods
In order to extend the high-fidelity dataset, synthetic data generation methods are implemented for both fault diagnosis and fault prognosis. Initially, the dataset is generated using the noise perturbation method by adding a monotonically increasing noise function in selective frequency contents. These datasets are utilised to extract features for fault diagnosis and they also form the initial dataset for fault prognosis. In order to achieve a degradation behaviour in the dataset for prognosis, a linear interpolation method (interpolation in time) and a cubic interpolation method (interpolation in model parameters) are employed. These synthetically generated datasets are considered for fault prognosis.

Machine Learning and hyper-parameter optimisation
Fault diagnosis and fault prognosis use ML classification and ML regression algorithms, respectively. Fault diagnosis employs ML classification algorithms MLP, SVM, and NB, whereas fault prognosis uses ML regression algorithms MLP and SVM. Even though established research exists in the field [55, 66–68, 77], there remains room for improvement in terms of regulating hyper-parameters. Hyper-parameters are generally optimised in a staggered manner for SVM and MLP. In addition, for MLP, the 2^n method is used to optimise the number of neurons in the layer. Employing a staggered manner to select the kernel function for SVM, the polynomial kernel outperforms when compared to the Gaussian kernel and the linear kernel; this has been similarly shown in [66].

Fault diagnosis with feature extraction methods

Fault diagnosis of the gear pump has been performed using feature extraction methods. This has enabled highly discriminatory information to be discerned from the time-varying values. Established feature extraction methods, such as time features (statistical features), frequency features (spectral features), and wavelet transform features (Shannon entropy-based features) are employed for fault diagnosis in various applications [58, 59, 69, 208, 212]. In this thesis, all these feature extraction methods are exploited to determine the optimal combination of feature extraction method with each of the ML algorithms, MLP, SVM and NB. It is demonstrated that the combination of wavelet features with the MLP algorithm provides the highest degree of accuracy for the fault diagnosis of the external gear pump. Notably, Shannon entropy-based wavelet features performance has been confirmed in [69]. Furthermore, time features with the NB algorithm provide the best classification accuracy.

Fault prognosis

Fault prognosis has been performed employing the ML regression algorithms, MLP and SVM. Synthetically generated degradation datasets, using a linear interpolation method and a cubic interpolation method, are applied to perform this analysis. The data generated using the cubic interpolation method provides a superior prediction of the life expectancy of the gear pump compared to the linear interpolation method. Compared to SVM, MLP yields better pressure and flow rate results, whereas MLP and SVM perform similarly for the analysis of torque.

Comparison of variables

In conclusion, a comparison of the physical variables has shown that flow rate is the foremost variable to facilitate the fault diagnosis and fault prognosis: it provides the highest degree of accuracy for classification and the lowest error margin for regression with the synthetic data generation methods employed. A primary interest of the engineering industry is to perform these analyses with minimal cost; this would suggest the use of torque because it is linearly related to current absorption, which can be monitored via the current sensor. Hence, torque analysis is provided for both fault diagnosis and prognosis.

Summary of contributions

Novel contributions to the advancement of knowledge that are intrinsic to this thesis are as follows:

- ☐ CFD analysis of the FG304 series external gear pump (aka domino pump) including turbulence and cavitation effect.

- ☐ Implementation of a greater range of fault scenarios with six different external gear pump conditions.
- ☐ Implementation of synthetic data generation methods for fault diagnosis and fault prognosis.
- ☐ Comparison of three feature extraction methods for fault diagnosis.
- ☐ Implementation of a fault diagnosis strategy in the case of data unavailability via the use of ML algorithms, MLP and SVM and NB.
- ☐ Implementation of a fault prognosis strategy in the case of data unavailability via the use of ML algorithms, MLP and SVM.

8.2 Future work

The findings presented in this thesis unlock several potential avenues of future research. A number of them are outlined next both at an academic and at an industrial research points of view.

8.2.1 Further research developments

Experimental data

Utilisation of sensor-based experimental data from an external gear pump, to replace or enhance the synthetically generated data used in this work, would be beneficial. When experimental data is used, significant importance must be given to its preprocessing. Within the industrial landscape in which this work is situated, the importance of big data brings many opportunities, but challenges also remain: abundant data types and complex data structures (due to the diversity of data sources), complicates data integration and aggregation; difficulties in assessing data quality within reasonable time periods due to high dimensionality and especially short timeliness; lack of unified and approved data quality standards, and elevated requirements for data processing technology [229].

Enhancement of hyper-parameter optimisation

Hyper-parameters require adjustment prior to training the ML algorithms due to their crucial role in directly controlling the functioning of the training algorithm. However, the increasing complexity of ML models requires the optimisation of an ever-increasing number of hyper-parameters. Also, manual manipulation of hyper-parameters is difficult, so an automated process of optimising hyper-parameters is preferable. Such optimisation algorithms can be separated into three primary categories, namely exhaustive search of the space, surrogate models, and evolutionary algorithms [230].

The method of an exhaustive search of the space, such as random search and grid search, is simple to use and can run in parallel but it has no guarantee of finding a local minimum to some degree of precision. Surrogate models, such as Bayesian optimisation, are widely used for the optimisation of hyper-parameters. Bayesian optimisation is efficient in tuning a limited number of hyper-parameters but suffers degradation as the search dimension increases [231]. A particular drawback of this algorithm is non-parallelism. It, therefore, incurs a high computational cost and can only operate on continuous hyper-parameters and not on categorical ones. The genetic algorithm is one of the most commonly used evolutionary algorithms. This algorithm requires less information regarding the problem, but it is potentially computationally expensive and designing an objective function can be complex. All these methods have been used to optimise the hyper-parameters for ML, but there still remains a place for an efficient, automatic and computationally less expensive algorithm.

Employing other deep learning algorithms

Depending upon the availability of experimental data and its processing difficulty, other deep learning algorithms, such as recurrent neural network, convolutional neural network, deep belief network, deep auto-encoder and the gated recurrent unit can be employed for fault diagnosis. Some of these methods have also been used for fault prognosis, but the case of improvement remains [115].

Employing filtering techniques

In Sections 6.6 and 7.6, the influence of noise level on the training and test datasets are analysed for diagnosis and prognosis. With the higher level of noise in the dataset, ML algorithms are performed poorly. In order to handle such situation, filtering techniques can be employed to eliminate or reduce the noise level in the dataset. Many filtering techniques available in the literature, including linear smoothing filter, non-linear filter, etc., [45, 232, 233].

Unsupervised learning

In case of predictive maintenance, even if significant quantities of machine and process data exist, a common problem intrinsic to such data is the lack of correct labellings describing the equipment condition or maintenance history. Additionally, each item of equipment generates highly heterogeneous data, resulting in difficulties in relation to information integration. For these reasons, engineering industries frequently have either completely omitted or have limited the analysis of their available manufacturing data. In situations such as these, unsupervised ML data analysis techniques can be employed for hidden pattern identification. Cluster analysis is the most frequently used method to perform this analysis. A number of recent studies have shown that clustering methods can be effective for fault diagnosis or fault detection. For example, K-means clustering with PCA for dimensionality reduction [234], K-means, density, spectral, agglomerative hierarchical [235] and K-means, fuzzy c-means, model-based, agglomerative hierarchical [236].

Expand the experimental data-vibration data
In the present thesis, data has been generated via CFD modelling. As a result, the monitored physical variables are restricted. However, in gear pump experiments, vibration data is primarily monitored to perform diagnosis and prognosis [55, 56]. The strategy introduced in this thesis can be applied to vibration data or any other measured form of data derived from pump devices.

Uncertainty quantification
The predictions formulated by ML or deep learning models are intrinsically uncertain because they are prone to noise and incorrect model inference. This is besides the inductive assumptions that are inherent in cases of uncertainty. For ML algorithms, there are two principle uncertainties: epistemic (model uncertainty) and aleatoric (data uncertainty). These are particularly important aspects of uncertainty quantification because they determine the reliability of the ML algorithms [237].

Internet of Things (IoT)
Globally, IoT is growing exponentially. IoT has provided abundant new opportunities in the technology sector while also bringing several challenges to the fore. IoT relies upon sensors being directly connected to the cloud, often via a wireless network, for the collection and transfer of data on a continuous basis. These datasets are frequently analysed by AI algorithms in cloud-based systems, providing real-time analysis of the monitored machinery. Such analysis can detect abnormal functionality or predict future failure events. However, the installation of IoT sensors in the equipment is costly.

8.2.2 Transfer to industry developments

The research contributions documented within this thesis have relevance for the industrial partner of this project, F-Lab. However, further work is required to ensure that the developed methodology can be transferred to an industrial context. This is outlined below.

Code optimisation
The developed MATLAB code and Python code should be translated into a unique and efficient programming language and several functions could be optimised by employing the encapsulation methods intrinsic to object-oriented programming.

Graphical User Interface
Implementation of the developed predictive maintenance methodology in a robust, Graphical User Interface (GUI) based, commercially available, software platform would enhance its ease of use.

System deployment
The optimal ML algorithms methods shown to be effective in this thesis can be deployed to predict the state of an industrial external gear pump. The deployment consists of a process of taking a trained ML model and offering its predictions to qualified users. The optimal value is gained when the insights that the ML model can bring are available on a consistent basis.

Appendix A

PUMPLINX

In the present appendix, the simulation workflow of PumpLinx is presented. Firstly, the preprocessing step of the simulation will be shown, then the computational model of the simulation (also known as solver) and finally, the postprocessing step will be discussed. Figure A.1 shows the workflow of PumpLinx. All the information provided here are based on PumpLinx/Semerics manual [6].

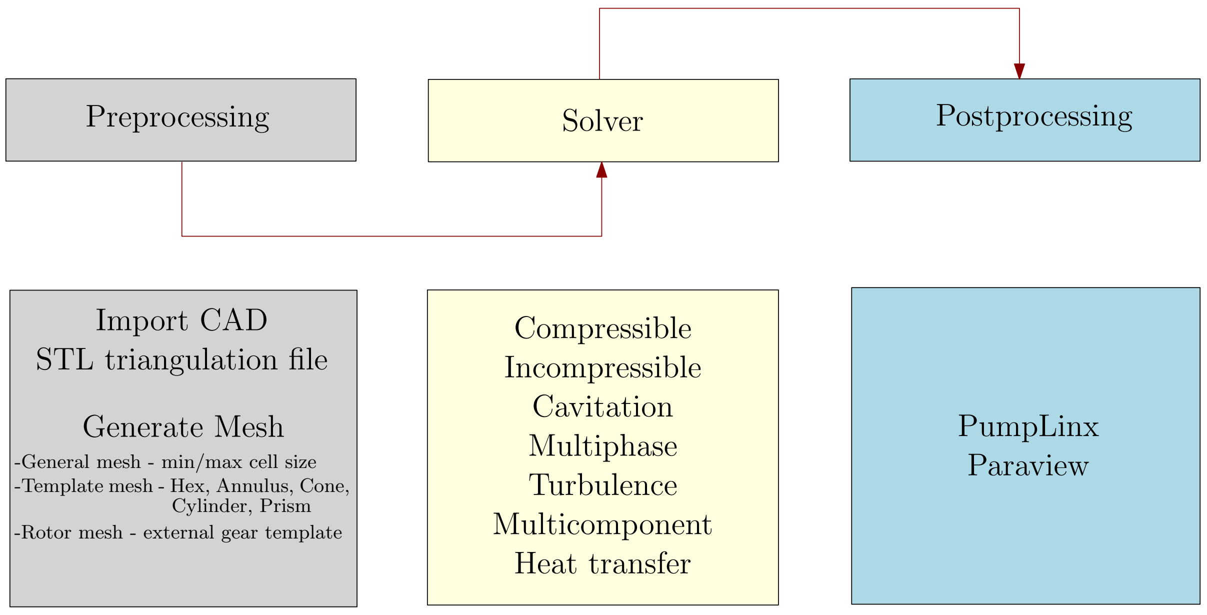


FIGURE A.1: Workflow of PumpLinx.

Preprocessing:
Prior to start building the simulation, one requires the CAD file of the object or geometry of interest. The CAD geometry can be created using several software programs available such as Solidworks, AutoCAD, Tinkercad, FreeCAD, IronCAD, Shapr3D, etc,. PumpLinx only allows importing the STL triangulation file from a CAD surface. In order to directly import the grid file, the software allows the use of a Simerics grid file, a Nastran grid file, a Gambit neutral file, an Ansys CDB file or an Gridpro file. PumpLinx has an in-built highly efficient mesh generator that uses a conformal adaptive binary tree algorithm in order to generate a Cartesian hexahedral cell-based mesh. There are four types of mesh generators in PumpLinx, namely general mesh, template mesh, and rotor template mesh and valve template mesh. The general mesh generator allows us to define the minimum and maximum size of cells, so during the mesh generation, it automatically adjusts the size of the mesh depending on the surface. The template mesh generator creates a mesh of different shapes including, hex, annulus, cone, cylinder, and prism elements. The rotor mesh generator provides templates for the external gear pump, gerotor, crescent, bend axial piston, rolling piston, radial piston, scroll pump and vane pump. The valve template mesh generator provides templates for the spool valve, ball valve, axial valve and circumferential valve. For our geometry, the rotor mesh template generator has been used for the external gear, the template mesh is used to produce an annulus shape mesh, and the rest of the geometry is meshed using the general mesh.

Computational modelling or solver selection:
In PumpLinx, compressible, incompressible, turbulence, cavitation, multiphase, multi-component and heat transfer solvers are available. For the external gear pump, during the CFD simulation, the flow equations, turbulence and cavitation equations are solved. For a gear pump template, the software offers pressure inlet and pressure outlet boundary conditions. Material of the fluid is defined as per the choice of the user. For rotation of the gear, the constant speed to be fixed and the direction of the rotation is given. For a given simulation, PumpLinx under the external gear pump module provides three options to define the time definition for the method of specifying the size and number of time steps are revolution, pockets and the total number of time steps. This is discussed in Section 3.4 under time convergence study.

PumpLinx numerical scheme offers three *spatial interpolation schemes* for the velocity and pressure discretisation, namely first order upwind scheme, central difference scheme and second order upwind scheme.

- First Order Upwind: the upwind scheme sets the value at a cell interface, based on the value from the cell that is upwind (upstream) of the interface of interest.

- Central: the central scheme sets the value at a cell interface using an average value from the cells on both sides of the interface of interest. Central differencing may be used with a *Blending Factor (B)* and a *bounded scheme* to help control the convergence.

- Second Order Upwind: the second order upwind scheme sets the value at a cell interface, based on a stencil of cells neighbouring the interface of interest. Second order upwind may also be used with a *blending factor* and a *bounded scheme* to help control the convergence.

The *blending factor* is used in conjunction with the higher order interpolation schemes (central and second order upwind) to help stabilise the convergence by including the first order upwind scheme using the formula below.

$$\varphi_{interface} = B\varphi_{upwind} + (1 - B)\varphi_{higher\ order\ scheme}. \tag{A.1}$$

Typical values of the blending factor vary from 0.1 to 0.5, and higher values make the solution more stable.

The *bounded scheme* is used in conjunction with the higher order interpolation schemes (central and second order upwind) to help stabilise the convergence by limiting the range value of the interpolation to be no more or less than the maximum or minimum of the cells neighbouring the cell-face of interest.

$$\varphi_{minimum} \leq \varphi_{interface} \leq \varphi_{maximum}. \tag{A.2}$$

For the examples presented in this thesis, the first order upwind method has been utilised.

Pressure-velocity coupling is relevant only for the pressure-based solver. PumpLinx numerical scheme offers three *pressure-velocity coupling methods*: SIMPLE, SIMPLEC and SIMPLES. SIMPLE refers to Semi-Implicit Method for Pressure-Linked equations, this method described in [161]). SIMPLEC refers to SIMPLE Consistent, this method is described in [238]. SIMPLES is Simerics proprietary extension of the SIMPLEC algorithm [6]. SIMPLES method is used in this thesis.

For relatively uncomplicated problems (laminar flows with no additional models activated) in which convergence is limited by the pressure-velocity coupling, often possible to obtain a converged solution more quickly using SIMPLEC. With SIMPLEC, the pressure-correction under-relaxation factor is generally set to 1.0, which aids in convergence speed-up. In some problems, however, increasing the pressure correction under-relaxation to 1.0 can lead to instability due to high mesh skewness. For such cases, use of slightly more conservative under-relaxation value (up to 0.7), or use the SIMPLE algorithm. For complicated flows involving turbulence and (or) additional physical models, SIMPLEC will improve convergence only if it is being limited by the pressure-velocity coupling. Often it will be one of the additional modelling parameters that limits convergence; in this case, SIMPLE and SIMPLEC will give similar convergence rates.

PumpLinx offers three *temporal schemes* for velocity and pressure solution

☐ First order backward scheme: the temporal updates are determined implicitly, using the previous time step alone.

☐ Second order backward scheme: the temporal updates are determined implicitly using the previous two time steps.

☐ Crank-Nicolson scheme: the temporal updates are determined using the previous time step alone, using explicit/implicit combination.

In this thesis, first order time accuracy is employed.

Converge criterion: The solution process in PumpLinx is iterative, so the user can limit the total number of iterations for a steady state simulation or the total number of time iterations per time step for a transient simulation by setting the converge criterion to the desired convergence tolerance. When the correction for the flow module drops below this converge criterion, the code will stop for steady state or move to the next time-step for a transient simulation, assuming that the criteria for all other modules have been met as well. In the flow module, the converge criterion dictates the convergence criterion for both the pressure P and velocity $\boldsymbol{v}$ solutions. In the cavitation module, the converge criterion dictates the convergence criterion of vapour mass fraction f_v. In the turbulence module, the converge criterion dictates the convergence criterion of turbulent kinetic energy k and turbulent kinetic energy dissipation rate ε. A smaller value of convergence criterion implies more precision, however, the cost of a smaller value of convergence criterion is more iterations, resulting in more computational time. The residual norm between iterating "i" and "$i+1$" of pressure, velocity, vapour mass fraction, turbulent kinetic energy and turbulent kinetic energy dissipation rate are calculated as

$$\text{Norm}_p = \frac{|R_p^{i+1} - R_p^i|}{|R_p^0|} < \text{tolerance}, \tag{A.3}$$

where, R_p^0 is initial value of pressure residual and $|\cdot|$ indicates a suitable norm.

$$\text{Norm}_{v_x} = \frac{|R_{v_x}^{i+1} - R_{v_x}^i|}{|R_{v_x}^0|} < \text{tolerance}, \tag{A.4}$$

similarly, Norm_{v_y} and Norm_{v_y} are calculated.

$$\text{Norm}_{f_v} = \frac{|R^{i+1}_{f_v} - R^{i}_{f_v}|}{|R^{0}_{f_v}|} < \text{tolerance}. \tag{A.5}$$

$$\text{Norm}_{k} = \frac{|R^{i+1}_{k} - R^{i}_{k}|}{|R^{0}_{k}|} < \text{tolerance}. \tag{A.6}$$

$$\text{Norm}_{\varepsilon} = \frac{|R^{i+1}_{\varepsilon} - R^{i}_{\varepsilon}|}{|R^{0}_{\varepsilon}|} < \text{tolerance}. \tag{A.7}$$

Hence

$$\max\left(\text{Norm}_p, \text{Norm}_{v_x}, \ldots, \text{Norm}_{\varepsilon}\right) < \text{tolerance}. \tag{A.8}$$

Turbulence modelling near wall

The usage of wall functions is subject to Boundary Layer theory [166], as shown in Figure A.2. The standard wall function defines the turbulent mean velocity near the wall. Turbulence is an instability generated by shear. The maximum turbulence level occurs

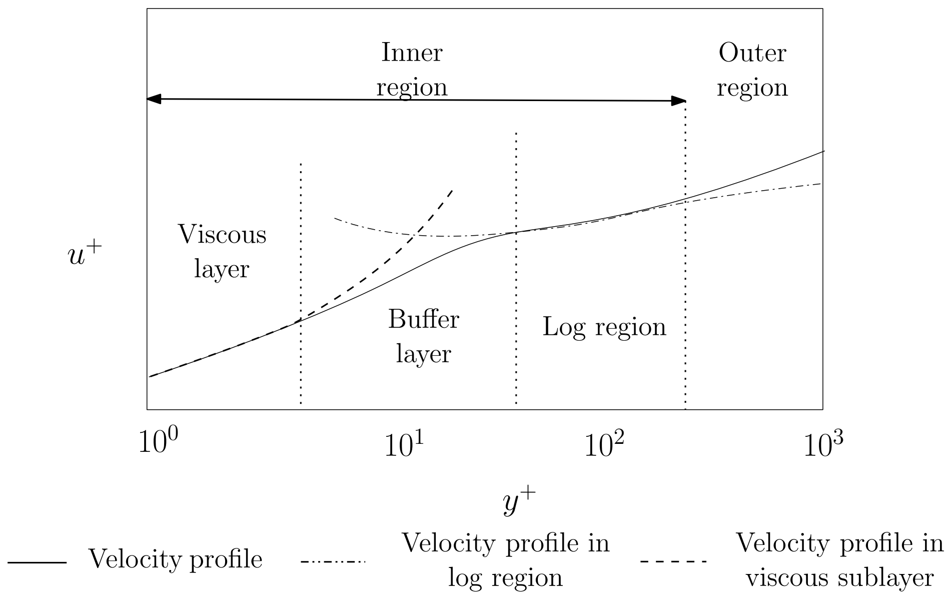


FIGURE A.2: Velocity profile of turbulent boundary layer. The figure is adapted from [5].

at the positions of maximum shear. But this scale relationship is not dimensionally consistent, so velocity scale to represent the shear strength is introduced as friction velocity, also known as shear velocity, and it characterises the shear at the boundary. An appropriate velocity scale for flow in the near-wall region is the friction velocity defined by

$$u_\tau \equiv \sqrt{\frac{\tau_w}{\rho_w}}, \tag{A.9}$$

where τ_w is the wall shear stress and ρ_w is the density at the wall. Using this velocity scale, a non-dimensional velocity and a non-dimensional length are defined by

$$u^+ \equiv \frac{u}{u_\tau}, \tag{A.10}$$

and

$$y^+ \equiv \frac{u_\tau y}{\nu}, \tag{A.11}$$

where u is the velocity component parallel to the wall, y is the distance normal to the wall and ν is the kinematic viscosity. As shown in Figure A.2, the equation for the velocity profile in the log region is

$$u^+ = \frac{1}{\kappa} Ln\left(y^+\right) + B, \tag{A.12}$$

where κ is the von Karman constant and B is an additional constant. In the viscous sublayer,

$$u^+ = y^+. \tag{A.13}$$

The equations describing the velocity profile in the inner region are collectively called *the law of the wall.* Further information related to this topic can be found in [239].

PumpLinx offers Nonequilibrium wall function and unified wall function in addition to the Standard wall function [6].

Postprocessing:

PumpLinx provides a user friendly interface for postprocessing. The contour plots can be visualised and saved in image or in Gif file. A time plot of physical variables can be visualised in the plot section and the data can be saved to any format of files. Furthermore, the PumpLinx result file can be imported into Paraview for postprocessing purpose.

Physical variables:

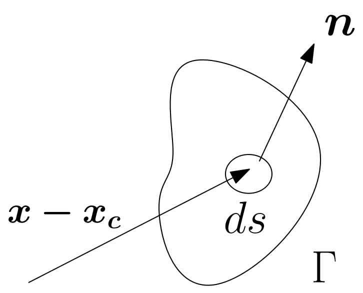
Definitions about how PumpLinx calculates some physical variables of interest are provided below.

Definition 5: **Velocity** refers to the velocity of the fluid. The SI units are metres per second [m/s].

Definition 6: **Pressure** refers to static pressure in the fluid. The SI units of the static pressure are Pascal (In this thesis, bar unit is used, 1 bar = 100000 Pascal).

Definition 7: **Volume flow rate** is the volume of fluid which passes per unit of time through a cross section. The SI units are cubic metres per second [m^3/s].

Definition 8: **Torque $\boldsymbol{T}$** is defined as



$$\boldsymbol{T} = \int_\Gamma \boldsymbol{r} \times \boldsymbol{t} \, ds, \tag{A.14}$$

where $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}_c$ and $\boldsymbol{t} = \boldsymbol{\sigma n}$, where $\boldsymbol{\sigma} = -p\boldsymbol{I} + \boldsymbol{\tau}$.

The component of $\boldsymbol{T}$ along an axis is defined by the unit vector $\boldsymbol{a}$ is:

$$T_a = \boldsymbol{T} \cdot \boldsymbol{a} = \left(\int_\Gamma \boldsymbol{r} \times (-p\boldsymbol{I} + \boldsymbol{\tau})\boldsymbol{n} \, ds \right) \cdot \boldsymbol{a}, \tag{A.15}$$

$$= \left(- \int_\Gamma \boldsymbol{r} \times (p\boldsymbol{n}) \, ds \right) \cdot \boldsymbol{a} + \left(\int_\Gamma \boldsymbol{r} \times (\tau\boldsymbol{n}) \, ds \right) \cdot \boldsymbol{a}, \tag{A.16}$$

$$= \boldsymbol{T}_p \cdot \boldsymbol{a} + \boldsymbol{T}_\tau \cdot \boldsymbol{a}. \tag{A.17}$$

Typically $\boldsymbol{T}_\tau \cdot \boldsymbol{a} <<< \boldsymbol{T}_p \cdot \boldsymbol{a}$, hence,

$$\boldsymbol{T} \cdot \boldsymbol{a} \cong \left(- \int_\Gamma \boldsymbol{r} \times (p\boldsymbol{n}) \, ds \right) \cdot \boldsymbol{a}. \tag{A.18}$$

Definition 9: **Total gas volume fraction** Average volume fraction of free Non-Condensible Gas (NCG) in a selected volume.

Appendix B

FREQUENCY TO WAVELET ANALYSIS

In the present appendix, the theory of wavelet transform will be discussed. In particular, why Maximal Overlap Discrete Wavelet Packet Transform (MODWPT) is very effective in signal processing will be explained.

Definition 10: $\mathbb{L}_p$ **space**
A function f belongs to the Lebesgue space $\mathbb{L}_p(\mathbb{A}), 1 \leq p < \infty$, if

$$||f||_p = \left(\int_{\mathbb{A}} |f(t)|^p dt \right)^{1/p} < \infty, \tag{B.1}$$

$$||f||_\infty = ess \sup_{t \in \mathbb{A}} |f(t)| < \infty. \tag{B.2}$$

In the analysis of any time series, the changes to physical variables are not enough to categorise the dynamic behaviour of the system. In such areas, how the system behaves can be obtained by the changes in frequencies. To compute the frequency response of a time series data, the most commonly used method is the Fourier transform. The FT breaks the non-sinusoidal signals into pieces of sinusoidal functions of varying amplitude and frequency. The Fourier Transform of a function $f \in \mathbb{L}_1(\mathbb{R})$ is defined by

$$\hat{f}(\omega) = \mathcal{F}[f(t)] =< f(t), e^{i\omega t} >= \int_{\mathbb{R}} f(t) e^{-i\omega t} dt. \tag{B.3}$$

The Fourier transform reveals the frequency components present in both the time signal and the magnitude of frequency response. The Fourier transform methods are designed to be used with stationary signals. However, the most practical signals contain time-varying frequency data that are not possible to classify in this manner. To overcome this issue, the Short Time Fourier Transform (STFT) is developed. The STFT of a function $f \in \mathbb{L}_1(\mathbb{R})$ with respect to $g \in \mathbb{R}_+/\{0\}$ is defined by

$$V_g f(\tau, \omega) = \int_{\mathbb{R}} f(t)\overline{g_{\tau,\omega}(t)}dt, \quad \forall\, \tau, \omega \in \mathbb{R}, \tag{B.4}$$

where $g_{\tau,\omega}(t) = g(t-\tau)e^{-i\omega t}$ and $g \neq 0$, real and symmetric window function. Fig. (B.1) shows the working principle of STFT with a window function. STFT uses localised time window to provide frequency information in the localised time width. The accuracy of STFT in time and frequency resolution depends on how large the time window of the transforms. Large windows provide good frequency information/ resolution but poor time information, whereas small windows provide good time information but lack good frequency information. These issues related to the resolution of STFT found to be a result of Heisenberg's uncertainty principle [240], which states that time and frequency cannot be resolved simultaneously. A possible solution for this issue is to adjust the flexible windowing strategy and selectively transform the signal depending on their time or frequency requirements. This strategy is used in wavelet analysis, the techniques firstly introduced in 1984 [241].

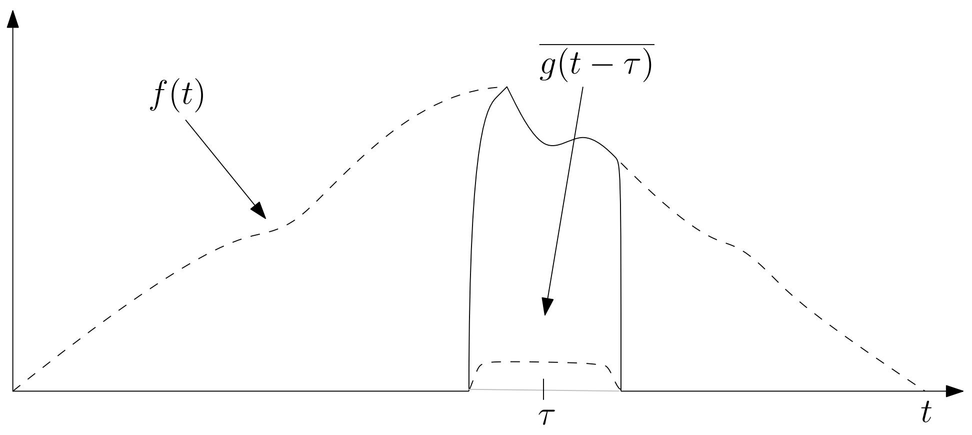


Figure B.1: Time-frequency analysis with STFT

Wavelets are a type of mathematical functions that reveal oscillatory behaviour. With sinusoidal functions, the oscillation dominates the entire signal, where wavelets exhibit only localised oscillation. Wavelet analysis includes breaking a signal into pieces of different frequency components by comparing the signal to several differently sized wavelet functions. Even though wavelet is closely related to Fourier analysis, it has the advantage to overcome some of their limitations. The two most commonly used forms of the wavelet transform are the continuous wavelet transform (CWT) and the discrete wavelet transform (DWT). For any $\mathbb{L}_2$ function $f(t)$, the Continuous Wavelet Transform (CWT) is defined as a function of two variables

$$\mathcal{CWT}_f(a,b) = \langle f, \psi_{a,b} \rangle = \int_{\mathbb{R}} f(t)\overline{\psi_{a,b}}(t)dt \quad \forall a,b \in \mathbb{R}/\{0\} \times \mathbb{R}, \tag{B.5}$$

where a, b are dilation and translation parameter respectively, and $\psi_{a,b}(t)$ is defined as

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi\left(\frac{t-b}{a}\right), \tag{B.6}$$

The CWT is a computationally demanding algorithm, so the DWT is developed but their method of computation differs. The CWT utilises subband coding whereas the DWT utilises pyramidal coding. The CWT is reluctant as it analyses the signal at all scales and translations, also requiring an infinite number of wavelets. Proper sampling of scale and translation and a reduced number of wavelets used for decomposition produces a significantly faster algorithm. In DWT, scale and time are sampled in powers of two $(2^1, 2^2, \ldots)$, this is commonly termed as dyadic sampling [242].

In order to define DWT, let us consider $\boldsymbol{x}$ containing a sequence $\{x_0, \ldots x_{N-1}\}$ of N and $\{\boldsymbol{x}_t, t = 0, \ldots, N-1\}$ real-valued time series and assumed N is the power of 2. For the implementation of the DWT, the decomposition is achieved through filtering of a signal using high-pass filters $\{h_l, l = 0, \ldots, L-1\}$ and low-pass $\{g_l, l = 0, \ldots, L-1\}$ filters. The low-pass and high-pass filters satisfy the unit energy equation in Equation (B.7) and orthonormal to its even shifts in Equation (B.8).

$$\sum_{l=0}^{L-1} g_l^2 = 1, \tag{B.7}$$

$$\sum_{l=0}^{L-1} g_l g_{l+2n} = \sum_{l=-\infty}^{\infty} g_l g_{l+2n} = 0, \quad \forall n \neq 0. \tag{B.8}$$

In addition, the filters are chosen to be quadrature mirror filters satisfying

$$h_l = (-1)^l g_{L-l-1} \quad \text{or} \quad g_l = (-1)^{l+1} h_{L-l-1} \quad \forall l = 0, \ldots, L-1. \tag{B.9}$$

Consider, $V_{0,t} = \boldsymbol{x}_t$, the jth step of input to the pyramid algorithm is $\{V_{j-1,t,\, t=0,\ldots N_{j-1}-1}\}$ where $N_j = N/2^j$. For the DWT pyramid algorithm (introduced in [243]), the jth step output is the jth level scaling and wavelet coefficients, which are given below [69].

$$V_{j,t} = \sum_{l=0}^{l-1} g_l V_{j-1,(2t+1-l) \bmod N_j - 1} \left(t = 0, \ldots, N_j - 1\right), \tag{B.10}$$

$$W_{j,t} = \sum_{l=0}^{l-1} h_l V_{j-1,(2t+1-l) \bmod N_j - 1} \left(t = 0, \ldots, N_j - 1\right), \tag{B.11}$$

where mod means modulus after division. As the DWT requires the sample size to be powers of 2, to overcome this limitation Maximal Overlap Discrete Wavelet Transform (MODWT) is developed. MODWT is a revised version of DWT by [70]. The MODWT of level j is defined for any sample size N. The low-pass and high-pass filters are re-scaled as required to conserve energy,

$$\tilde{g}_l = g_l/\sqrt{2}, \tag{B.12}$$

$$\tilde{h}_l = h_l/\sqrt{2}. \tag{B.13}$$

Now, the filters $\tilde{g}_l, \tilde{h}_l$ satisfies the conditions in Equations (B.7) and (B.8), and it becomes

$$\sum_{l=0}^{L-1} \tilde{g}_l^2 = 1/2, \quad \sum_{l=0}^{L-1} \tilde{g}_l \tilde{g}_{l+2n} = \sum_{l=-\infty}^{\infty} \tilde{g}_l \tilde{g}_{l+2n} = 0, \quad \forall n \neq 0. \tag{B.14}$$

Similar to Equation (B.9), the filters are yet quadrature mirror satisfying

$$\tilde{h}_l = (-1)^l \tilde{g}_{L-l-1} \quad \text{or} \quad \tilde{g}_l = (-1)^{l+1} \tilde{h}_{L-l-1} \quad \forall\, l = 0, \ldots, L-1. \tag{B.15}$$

In order to avoid downsampling in the DWT, the MODWT creates appropriate new filters at each step by inserting $2^{j-1} - 1$ zeros between the elements of $\tilde{h}_l$ and $\tilde{g}_l$. Similar to DWT, the MODWT pyramid algorithm creates scaling and wavelet coefficients, which is defined as

$$V_{j,t} = \sum_{l=0}^{l-1} \tilde{g}_l V_{j-1,(t-2^{j-1}l) \bmod N} \left(t = 0, \ldots, N-1\right), \tag{B.16}$$

$$W_{j,t} = \sum_{l=0}^{l-1} \tilde{h}_l V_{j-1,(t-2^{j-1}l) \bmod N} \left(t = 0, \ldots, N-1\right). \tag{B.17}$$

MODWT has excellent frequency resolution in the low frequencies, however a poor frequency resolution in the high frequencies. To get through this problem, A.T Walden and A. Contreras Cristan introduced MODWPT [70]. Before moving to details of MODWPT, let us define the Discrete Wavelet Packet Transform (DWPT), which is a generalization of the DWT. At level j of DWPT, partitions the frequency into 2^j frequency bands with equal width labelled $n = 0, \ldots 2^j - 1$. Increasing the transform level j increases the frequency resolution. For length N with level j there are only $N/2^j$ DWPT coefficients for each frequency band n. Hence resolution in time is reduced by a factor of 2 at each step (by decimation).

Similar to MODWT, MODWPT filters are related to quadrature mirror filters, Equation (B.15). For the Equations (B.12) and (B.13), the transfer function corresponding to $\tilde{g}_l$ and $\tilde{h}_l$ is given by

$$\tilde{G}(\omega) = \sum_{l=0}^{L-1} \tilde{g}_l e^{-i2\pi\omega l}, \tag{B.18}$$

$$\tilde{H}(\omega) = \sum_{l=0}^{L-1} \tilde{h}_l e^{-i2\pi\omega l}, \tag{B.19}$$

where ω is a frequency. Some of the well known choices of $\tilde{h}_l$ and $\tilde{g}_l$ are Haar wavelets, Daubechies wavelets, Fejér-Korovkin wavelets and others.

The MODWPT is illustrated in Figure B.2, where only positive frequency range is shown. Let $\boldsymbol{x} = x_t$ is a column vector, and it conveys the fact that the support of the power spectrum of $\boldsymbol{x}$ is the standard dimensionless frequency range for unit sample interval [70]. In order to compute maximal overlap wavelet packet coefficients for level j, the wavelet

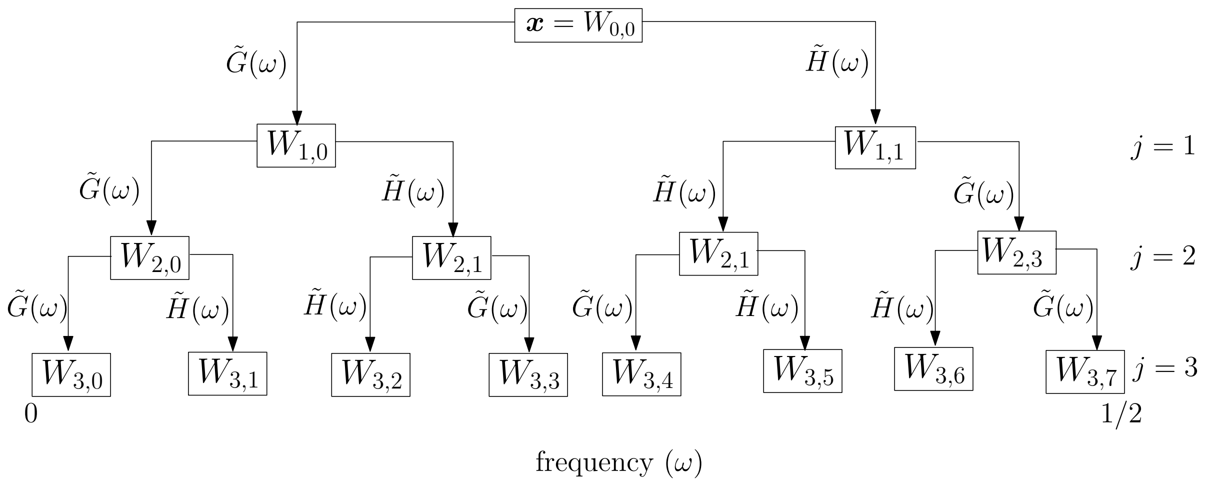


FIGURE B.2: Sequency-ordered wavelet packet tree showing the MODWPT of $\boldsymbol{x}$ into $W_{j,n}$, n= 0,1,... $2^j - 1$, for j =1, 2 and 3.

packet coefficients are circularly filtered in the previous stage. Let $W_{0,0} = \boldsymbol{x}$, given the series $W_{j-1,\lfloor n/2 \rfloor,t}$ of length N. The $W_{j,n,t}$ with n being frequency-index, produced using

$$W_{j,n,t} = \sum_{l=0}^{l-1} \tilde{\Gamma}_{n,l} W_{j-1,\lfloor n/2 \rfloor,(t-2^{j-1}l) \bmod N}, \tag{B.20}$$

where

$$\tilde{\Gamma}_{n,l} = \begin{cases} \tilde{g}_l, & \text{if } n \bmod 4 = 0 \text{ or } 3, \\ \tilde{h}_l, & \text{if } n \bmod 4 = 1 \text{ or } 2. \end{cases} \tag{B.21}$$

Interested reader can find more detail about MODWPT in [70].

Appendix C

CODE DESCRIPTION

In the present appendix, the code description of synthetic data generation methods and ML are presented. Firstly, the MATLAB code for noise perturbation method, then linear interpolation method and cubic interpolation method are discussed. The implementation of the noise perturbation method has been discussed in Section 5.3.1. The dataset generated using this method is used in both fault diagnosis and prognosis. The implementation of linear interpolation and cubic interpolation methods are presented in Section 5.3.2. The dataset generated using this method is used in fault prognosis. Finally, MATLAB and Python code for ML algorithms using specific libraries are presented. MATLAB code for SVM classification and Python code for SVM regression, MLP regression and classification and NB classification are presented. The code descriptions as follows.

Noise perturbation method.

Main.m

```
clear all
close all
clc

load Data.mat % Data size is (1620,1). It can be considred as many ...
    time series as you want
Dataset = normalize(Data) % standardisation or z-score normalisation
[m_Data] = noisePerturbationData(Dataset,Increment); %m_Data should ...
    be of size (1620,23)
```

noisePerturbationData.m

```
function [m_Data] = noisePerturbationData(Dataset)

n_Dataset = size(Dataset,1);
% In frequency domian: [first peak : (second peak - first peak): ...
    n_Dataset/2]
positions = 37 : 36 : n_Dataset/2;
n_positions= length(positions);

% noise generation
x1 = 1; x2 = 3; x3 = n_positions;
b = [0 1 100];       % Predefined conditions
A = [x1^2 x1 1;
x2^2 x2 1;
x3^2 x3 1];
AA = inv(A);
coef = AA * transpose(b);

x1 = 1:1:n_positions; % new values to interpolate
y = @(x) coef(1)*x.^2 + coef(2)*x +  coef(3);
noise = y(x1);

% Convert to Fourier transform, add noise, invert back to time domian
for i = 1:size(Dataset,2)% if there is more than once time series , ...
    otherwise i=1
    q = fft(Dataset(:,i));
    RR(:,1,i) = q;
        for j = 1:n_positions
            RR( :, j+1,i) = q;

            noise_position = positions(j);
            RR(noise_position, j+1,i) = q(noise_position)*(1+ noise(j));
        end

        for j= 1:n_positions +1
            data_now = squeeze(RR(:,j,i));
            invFouriTransform( j+((n_positions +1)*(i-1)),:) = ...
    real(ifft(data_now));
        end

end
m_Data = transpose(invFouriTransform);
```

Linear and cubic interpolation method.

Main.m

```
clear all
close all
clc

load Data.mat %Data size is (1620,4), it contains pressure values of ...
    healthy and 3 gap variations da
Dataset = normalize(Data) % standardisation or z-score normalisation

% Each of these produce (1620,23)
[H_Data]  = noisePerturbationData(Dataset(:,1));
[F1_Data] = noisePerturbationData(Dataset(:,2));
[F2_Data] = noisePerturbationData(Dataset(:,3));
[F3_Data] = noisePerturbationData(Dataset(:,4));

% Linear interpolation through H - F1, F1-F2 and F2-F3
[Interpolated_Data, RUL] = ...
    linearInterpolationData(H_Data,F1_Data,F2_Data, F3_Data);
% Interpolated_Data - (1620,2300), RUL - (1,2300)

% Cubic interpolation through H - F1 - F2 - F3
n_samples = 100; % Number of values to interpolate
[Interpolated_Data, RUL] = ...
    cubicInterpolationData(H_Data,F1_Data,F2_Data, F3_Data, n_samples);
% if n_samples is 100, Interpolated_Data - (1620,2300), RUL - (1,2300)
```

linearInterpolationData.m

```
function [Data_I, RUL] = ...
    linearInterpolationData(H_Data,F1_Data,F2_Data, F3_Data)

d1= H_Data-F1_Data;
d2= F1_Data-F2_Data;
d3= F2_Data-F3_Data;

noise= fliplr(linspace(0.01,0.99,32));

% H_data to F1_data
Rul1=[]; % to store RUL
Data1=[]; % to store the data

for ii = 1:size(d1,2)
    Data_combine=[];
    for i= 1:size(noise,2)
        z(:,i)= F1_Data(:,ii) +(noise(i)* d1(:,ii));
    end
    Data_combine =[H_Data(:,ii) z F1_Data(:,ii)];
    Data1= [Data1 Data_combine];
    Rul_1=  fliplr(67:1:100);
    Rul1= [Rul1 Rul_1];
end

% F1_data to F2_data
Rul2=[];
Data2=[];

for ii = 1:size(d2,2)
    Data_combine=[];
    for i= 1:size(noise,2)
        z1(:,i)= F2_Data(:,ii) +(noise(i)* d2(:,ii));
    end
    Data_combine =[z1 F2_Data(:,ii)];
    Data2= [Data2 Data_combine];
    Rul_2= (fliplr(34:1:66));
    Rul2= [Rul2 Rul_2];
end

% F2_data to F3_data
Rul3=[];
Data3=[];
for ii = 1:size(d3,2)
    Data_combine=[];
    for i= 1:size(noise,2)
        z1(:,i)= F3_Data(:,ii) +(noise(i)* d3(:,ii));
    end
    Data_combine =[z1 F3_Data(:,ii)];
    Data3= [Data3 Data_combine];
    Rul_3= (fliplr(1:1:33));
    Rul3= [Rul3 Rul_3];
end

RUL= [Rul1 Rul2 Rul3];
Data_I = ([Data1 Data2 Data3]);

end
```

cubicInterpolationData.m

```
function [I_data, Rul] = ...
    cubicInterpolationData(H_Data,F1_Data,F2_Data, F3_Data, n_samples)
% model parameters (Gap sizes in radial gap faulty scenarios)
```

```
x1= 0.03;
x2= 0.05;
x3= 0.07;
x4= 0.09;

% solving system of equaitons
A = [x1^3 x1^2 x1 1;
x2^3 x2^2 x2 1;
x3^3 x3^2 x3 1;
x4^3 x4^2 x4 1];
AA= inv(A);
x_new = linspace(x1,x4,n_samples); % new values to interpolate

I_data = [];
Rul= [];
for i= 1:size(H_Data,2)% 23
    input_data = [H_Data(:, i) F1_Data(:, i) F2_Data(:, i) ...
   F3_Data(:, i)];
    for j=1:size(H_Data,1)%1620
        b= input_data(j,:);
        coef= AA*transpose(b);
        y = @(x) coef(1)*x.^3 + coef(2)*x.^2 +  coef(3)*x + coef(4);
        Int_data(j,:)= y(x_new);
    end
    I_data= [I_data Int_data];
    Rul_1=  fliplr(1:1:n_samples);
    Rul= [Rul Rul_1];
end

end
```

SVM classification.

SVM_classification.m

```
template = templateSVM(...
        'KernelFunction','polynomial',...
        'PolynomialOrder',2,...
        'KernelScale','auto',...
        'BoxConstraint',1,...
        'Standardize',true);

 model = fitcecoc(...
         trainX,...
         trainY,...
         'Learners',template,...
         'Coding','onevsone',...
        'ClassNames',[1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 11; 12; 13; ...
    14; 15; 16; 17]);
predLabels = predict(model,testX);
correctprediction =(predLabels == testY);
testAccuracy = sum(correctprediction)/numel(correctprediction)*100;
```

SVM regression

SVM_regression.py

```
from sklearn import metrics
from sklearn.svm import SVR

model = SVR(kernel=kernel_function, C=100, gamma='auto', degree=3, ...
    epsilon=.1, coef0=1)
model.fit(trainX,trainY)
predTest = model.predict(testData)
testMSE = metrics.mean_squared_error(predTest,testY)
```

MLP classification and regression

MLP_classification_regression.py

```
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from sklearn import metrics
from sklearn.metrics import accuracy_score

hn_1= 32 # number of neurons in hidden layer 1
hn_2= 16 # number of neurons in hidden layer 1
model = Sequential()
model.add(Dense(hn_1, input_dim=trainX.shape[1], ...
    activation=activation_fuction))# Hidden 1
model.add(Dense(hn_2, activation=activation_fuction)) # Hidden 2

#%% MLP classification
model.add(Dense(trainX.shape[1],activation='softmax')) # Output
model.compile(loss='categorical_crossentropy', optimizer='adam', ...
    metrics=['accuracy'])
model.fit(trainX,trainY,verbose=0,epochs=300,callbacks=[early_stopping])
predTest = model.predict(testX)
predict_classes = np.argmax(predTest,axis=1)
testAccuracy = accuracy_score(predTest,testY)*100

#%% MLP regression
model.add(Dense(trainData.shape[1],activation='linear')) # Output
model.compile(loss='mean_squared_error', optimizer='adam', ...
    metrics=['MSE'])
model.fit(trainX,trainY,verbose=0,epochs=300,callbacks=[early_stopping])
predTest = model.predict(testData)
testMSE = metrics.mean_squared_error(predTest,testY)
```

NB classification

NB_classification.py

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

model = GaussianNB()
model.fit(trainX,trainY)
testAccuracy = accuracy_score(predTest,testY)*100
```

BIBLIOGRAPHY

[1] Henning Kagermann, Wolfgang Wahlster, and Johannes Helbig. Recommendations for implementing the strategic initiative industrie 4.0 – securing the future of german manufacturing industry. Final report of the industrie 4.0 working group, acatech – National Academy of Science and Engineering.

[2] Fluid-o-tech. URL `https://www.fluidotech.it/en/products/technologies/external-gear-pumps/`.

[3] Jean Donea and Antonio Huerta. *Finite element methods for flow problems*. John Wiley & Sons, 2003.

[4] Koushal Kumar and B Abhishek. *Artificial neural networks for diagnosis of kidney stones disease*. GRIN Verlag, 2012.

[5] DA Anderson, JC Tannehill, and RH Fletcher. Computational fluid mechanics and heat transfer: Series in computational meth-ods in mechanics and thermal. *Sciences, Hemisphere Publishing Corporation, Washington, DC*, 1984.

[6] Semerics INC. *Semerics and PumpLinx Manual*, 2011. URL `https://www.simerics.com/pumplinx/`.

[7] Robin Pitblado. Global process industry initiatives to reduce major accident hazards. *Journal of Loss Prevention in the Process Industries*, 24(1):57–62, 2011.

[8] Peter Okoh and Stein Haugen. Maintenance-related major accidents: Classification of causes and case study. *Journal of Loss Prevention in the Process Industries*, 26 (6):1060–1070, 2013. ISSN 0950-4230.

[9] Til Baalisampang, Rouzbeh Abbassi, Vikram Garaniya, Faisal Khan, and Mohammad Dadashzadeh. Review and analysis of fire and explosion accidents in maritime transportation. *Ocean Engineering*, 158:350–366, 2018.

[10] Til Baalisampang, Rouzbeh Abbassi, and Faisal Khan. Overview of marine and offshore safety. In *Methods in Chemical Process Safety*, volume 2, pages 1–97. Elsevier, 2018.

[11] Til Baalisampang, Rouzbeh Abbassi, Vikram Garaniya, Faisal Khan, and Mohammad Dadashzadeh. Modelling an integrated impact of fire, explosion and combustion products during transitional events caused by an accidental release of lng. *Process Safety and Environmental Protection*, 128:259–272, 2019.

[12] Peter Okoh and Stein Haugen. A study of maintenance-related major accident cases in the 21st century. *Process Safety and Environmental Protection*, 92(4): 346–356, 2014.

[13] Matthew Aladesaye. *Application of predictive maintenance to industry including Cepstrum analysis of a gearbox: a thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy.* PhD thesis, Massey University, 2008.

[14] C Sheut and LJ Krajewski. A decision model for corrective maintenance management. *The International Journal of Production Research*, 32(6):1365–1382, 1994.

[15] Yuanhang Wang, Chao Deng, Jun Wu, Yingchun Wang, and Yao Xiong. A corrective maintenance scheme for engineering equipment. *Engineering Failure Analysis*, 36:269–283, 2014.

[16] A. G. Starr. A structured approach to the selection of condition based maintenance. In *Fifth International Conference on Factory 2000 - The Technology Exploitation Process*, pages 131–138, 1997.

[17] Gang Niu, Bo-Suk Yang, and Michael Pecht. Development of an optimized condition-based maintenance system by data fusion and reliability-centered maintenance. *Reliability Engineering & System Safety*, 95(7):786–796, 2010.

[18] Jean Pierre Kenné and LJ Nkeungoue. Simultaneous control of production, preventive and corrective maintenance rates of a failure-prone manufacturing system. *Applied numerical mathematics*, 58(2):180–194, 2008.

[19] Mazhar Ali Khan Malik. Reliable preventive maintenance scheduling. *AIIE transactions*, 11(3):221–228, 1979.

[20] Ilya Gertsbakh. *Reliability theory: with applications to preventive maintenance.* Springer, 2013.

[21] SH Sim and J Endrenyi. Optimal preventive maintenance with repair. *IEEE Transactions on Reliability*, 37(1):92–96, 1988.

[22] Adiel Teixeira De Almeida, Cristiano Alexandre Virgínio Cavalcante, Marcelo Hazin Alencar, Rodrigo José Pires Ferreira, Adiel Teixeira de Almeida-Filho, and Thalles Vitelli Garcez. Preventive maintenance decisions. In *Multicriteria and Multiobjective Models for Risk, Reliability and Maintenance Decision Analysis*, pages 215–232. Springer, 2015.

[23] Andrew KS Jardine, Daming Lin, and Dragan Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, 20(7):1483–1510, 2006.

[24] Rosmaini Ahmad and Shahrul Kamaruddin. An overview of time-based and condition-based maintenance in industrial application. *Computers & industrial engineering*, 63(1):135–149, 2012.

[25] Albert HC Tsang. Condition-based maintenance: tools and decision making. *Journal of quality in maintenance engineering*, 1995.

[26] John H Williams, Alan Davies, and Paul R Drake. *Condition-based maintenance and machine diagnostics*. Springer Science & Business Media, 1994.

[27] R Keith Mobley. *An introduction to predictive maintenance*. Elsevier, 2002.

[28] HM Hashemian. Wireless sensors for predictive maintenance of rotating equipment in research reactors. *Annals of Nuclear Energy*, 38(2-3):665–680, 2011.

[29] Michael Wetzer. Performing predictive maintenance on equipment, May 18 2004. US Patent 6,738,748.

[30] Antoine Grall, Laurence Dieulle, Christophe Bérenguer, and Michel Roussignol. Continuous-time predictive-maintenance scheduling for a deteriorating system. *IEEE transactions on reliability*, 51(2):141–150, 2002.

[31] Henning Kagermann. Change through digitization—value creation in the age of industry 4.0. In *Management of permanent change*, pages 23–45. Springer, 2015.

[32] Michael E Porter and James E Heppelmann. How smart, connected products are transforming companies. *Harvard business review*, 93(10):96–114, 2015.

[33] Jay Lee, Behrad Bagheri, and Hung-An Kao. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters*, 3:18–23, 2015.

[34] Jana Wäldchen and Patrick Mäder. Machine learning for image based species identification. *Methods in Ecology and Evolution*, 9(11):2216–2225, 2018.

[35] Rebecca Welte, Manfred Estler, and Dominik Lucke. A method for implementation of machine learning solutions for predictive maintenance in small and medium sized enterprises. *Procedia CIRP*, 93:909–914, 2020.

[36] Kamran Javed, Rafael Gouriveau, and Noureddine Zerhouni. State of the art and taxonomy of prognostics approaches, trends of prognostics applications and open issues towards maturity at different technology readiness levels. *Mechanical Systems and Signal Processing*, 94:214–236, 2017.

[37] Marcia Baptista, Elsa MP Henriques, Ivo P de Medeiros, Joao P Malere, Cairo L Nascimento Jr, and Helmut Prendinger. Remaining useful life estimation in aeronautics: Combining data-driven and kalman filtering. *Reliability Engineering & System Safety*, 184:228–239, 2019.

[38] Tangbin Xia, Yifan Dong, Lei Xiao, Shichang Du, Ershun Pan, and Lifeng Xi. Recent advances in prognostics and health management for advanced manufacturing paradigms. *Reliability Engineering & System Safety*, 178:255–268, 2018.

[39] Jay Lee, Jun Ni, Dragan Djurdjanovic, Hai Qiu, and Haitao Liao. Intelligent prognostics tools and e-maintenance. *Computers in industry*, 57(6):476–489, 2006.

[40] Subhasis Nandi, Hamid A Toliyat, and Xiaodong Li. Condition monitoring and fault diagnosis of electrical motors—a review. *IEEE transactions on energy conversion*, 20(4):719–729, 2005.

[41] Mohammadhamed Ardakani, Mahdieh Askarian, Ahmed Shokry, Gerard Escudero, Moisès Graells, and Antonio Espuña. Optimal features selection for designing a fault diagnosis system. In *Computer Aided Chemical Engineering*, volume 38, pages 1111–1116. Elsevier, 2016.

[42] Silvio Simani, Cesare Fantuzzi, and Ronald Jon Patton. Model-based fault diagnosis techniques. In *Model-based Fault Diagnosis in Dynamic Systems Using Identification Techniques*, pages 19–60. Springer, 2003.

[43] Janos Gertler. *Fault detection and diagnosis in engineering systems.* CRC press, 1998.

[44] DC Baillie and J Mathew. Nonlinear model-based fault diagnosis of bearings. *NDT and e International*, 5(30):328, 1997.

[45] Kenneth A Loparo, Maurice L Adams, Wei Lin, M Farouk Abdel-Magied, and Nadar Afshari. Fault detection and diagnosis of rotating machinery. *IEEE Transactions on Industrial Electronics*, 47(5):1005–1014, 2000.

[46] Ian Howard, Shengxiang Jia, and Jiande Wang. The dynamic modelling of a spur gear in mesh including friction and a crack. *Mechanical systems and signal processing*, 15(5):831–853, 2001.

[47] Kenneth A Loparo, Abdulazim H Falah, and Maurice L Adams. Model-based fault detection and diagnosis in rotating machinery. In *Proceedings of the tenth international congress on Sound and vibration, Stockholm, Sweden*, pages 1299–1306, 2003.

[48] Andrea Vania and Paolo Pennacchi. Experimental and theoretical application of fault identification measures of accuracy in rotating machine diagnostics. *Mechanical Systems and Signal Processing*, 18(2):329–352, 2004.

[49] Jun Ma and James C Li. Detection of localised defects in rolling element bearings via composite hypothesis test. *Mechanical Systems and Signal Processing*, 9(1): 63–75, 1995.

[50] Yong-Wha Kim, Giorgio Rizzoni, and Vadim I Utkin. Developing a fault tolerant power-train control system by integrating design of control and diagnostics. *International Journal of Robust and Nonlinear Control: IFAC-Affiliated Journal*, 11 (11):1095–1114, 2001.

[51] Michael L Fugate, Hoon Sohn, and Charles R Farrar. Vibration-based damage detection using statistical process control. *Mechanical systems and signal processing*, 15(4):707–721, 2001.

[52] Tran Van Tung and Bo-Suk Yang. Machine fault diagnosis and prognosis: The state of the art. *International Journal of Fluid Machinery and Systems*, 2(1): 61–71, 2009.

[53] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus-Dieter Thoben. Machine learning in manufacturing: advantages, challenges, and applications. *Production & Manufacturing Research*, 4(1):23–45, 2016.

[54] Symone Gomes Soares and Rui Araújo. An on-line weighted ensemble of regressor models to handle concept drifts. *Engineering Applications of Artificial Intelligence*, 37:392–406, 2015.

[55] Zengkai Liu, Yonghong Liu, Hongkai Shan, Baoping Cai, and Qing Huang. A fault diagnosis methodology for gear pump based on eemd and bayesian network. *PloS one*, 10(5), 2015.

[56] Tianyang Wang, Qinkai Han, Fulei Chu, and Zhipeng Feng. Vibration based condition monitoring and fault diagnosis of wind turbine planetary gearbox: A review. *Mechanical Systems and Signal Processing*, 126:662–685, 2019.

[57] Ruonan Liu, Boyuan Yang, Enrico Zio, and Xuefeng Chen. Artificial intelligence for fault diagnosis of rotating machinery: A review. *Mechanical Systems and Signal Processing*, 108:33–47, 2018.

[58] B Samanta. Gear fault detection using artificial neural networks and support vector machines with genetic algorithms. *Mechanical systems and signal processing*, 18 (3):625–644, 2004.

[59] Saeid Farokhzad. Vibration based fault detection of centrifugal pump by fast fourier transform and adaptive neuro-fuzzy inference system. *Journal of mechanical engineering and technology*, 1(3):82–87, 2013.

[60] Saeid Farokhzad, Hojjat Ahmadi, and Ali Jafary. Fault classification of centrifugal water pump based on decision tree and regression model. *Journal of Science and today's world*, 2(2):170–176, 2013.

[61] Yanxue Wang, Jiawei Xiang, Richard Markert, and Ming Liang. Spectral kurtosis for fault detection, diagnosis and prognostics of rotating machines: A review with applications. *Mechanical Systems and Signal Processing*, 66:679–698, 2016.

[62] Tomasz Barszcz and Robert B Randall. Application of spectral kurtosis for detection of a tooth crack in the planetary gear of a wind turbine. *Mechanical Systems and Signal Processing*, 23(4):1352–1365, 2009.

[63] Kaveh Mollazade, Hojat Ahmadi, Mahmoud Omid, and Reza Alimardani. An intelligent combined method based on power spectral density, decision trees and fuzzy logic for hydraulic pumps fault diagnosis. *International Journal of Intelligent Systems and Technologies*, 3(4):251–263, 2008.

[64] ZK Peng and FL Chu. Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography. *Mechanical systems and signal processing*, 18(2):199–221, 2004.

[65] Ruqiang Yan, Robert X Gao, and Xuefeng Chen. Wavelets for fault diagnosis of rotary machines: A review with applications. *Signal processing*, 96:1–15, 2014.

[66] Maamar Ali Saud ALTobi, Geraint Bevan, Peter Wallace, David Harrison, and KP Ramachandran. Fault diagnosis of a centrifugal pump using mlp-gabp and

svm with cwt. *Engineering Science and Technology, an International Journal*, 22 (3):854–861, 2019.

[67] V Muralidharan and V Sugumaran. Feature extraction using wavelets and classification through decision tree algorithm for fault diagnosis of mono-block centrifugal pump. *Measurement*, 46(1):353–359, 2013.

[68] B Bagheri, H Ahmadi, and R Labbafi. Implementing discrete wavelet transform and artificial neural networks for acoustic condition monitoring of gearbox. *Elixir Mech. Engg*, 35:2909–2911, 2011.

[69] Yu Yang, Yigang He, Junsheng Cheng, and Dejie Yu. A gear fault diagnosis using hilbert spectrum based on modwpt and a comparison with emd approach. *Measurement*, 42(4):542–551, 2009.

[70] Andrew T Walden and A Contreras Cristan. The phase–corrected undecimated discrete wavelet packet transform and its application to interpreting the timing of events. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 454(1976):2243–2266, 1998.

[71] Xiao Yu, Xiaohong Ren, Hong Wan, Shoupeng Wu, and Enjie Ding. Rolling bearing fault feature extraction and diagnosis method based on modwpt and dbn. In *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–7. IEEE, 2019.

[72] VA Skormin, LJ Popyack, VI Gorodetski, ML Araiza, and JD Michel. Applications of cluster analysis in diagnostics-related problems. In *1999 IEEE Aerospace Conference. Proceedings (Cat. No. 99TH8403)*, volume 3, pages 161–168. IEEE, 1999.

[73] Mariano Artés, Lourdes Del Castillo, and Jesús Pérez. Failure prevention and diagnosis in machine elements using cluster. In *Proceedings of the Tenth International Congress on Sound and Vibration, Stockholm, Sweden*, pages 1197–1203, 2003.

[74] Qiao Sun, Ping Chen, Dajun Zhang, and Fengfeng Xi. Pattern recognition for automatic machinery fault diagnosis. *Journal of Vibration and Acoustics*, 126(2): 307–316, 05 2004. ISSN 1048-9002.

[75] Yuanhong Liu, Yansheng Zhang, Zhiwei Yu, and Ming Zeng. Incremental supervised locally linear embedding for machinery fault diagnosis. *Engineering Applications of Artificial Intelligence*, 50:60–70, 2016.

[76] Ming Guo, Lei Xie, Shu-Qing Wang, and Jian-Ming Zhang. Research on an integrated ica-svm based framework for fault diagnosis. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics. Conference Theme-System Security and Assurance (Cat. No. 03CH37483)*, volume 3, pages 2710–2715. IEEE, 2003.

[77] Kiran Vernekar, Hemantha Kumar, and KV Gangadharan. Engine gearbox fault diagnosis using empirical mode decomposition method and naïve bayes algorithm. *Sādhanā*, 42(7):1143–1153, 2017.

[78] Xiao Jian Yi, Yue Feng Chen, and Peng Hou. Fault diagnosis of rolling element bearing using naïve bayes classifier. *Vibroengineering PROCEDIA*, 14:64–69, 2017.

[79] Arfat Siddique, GS Yadava, and Bhim Singh. Applications of artificial intelligence techniques for induction machine stator fault diagnostics. In *4th IEEE International Symposium on Diagnostics for Electric Machines, Power Electronics and Drives, 2003. SDEMPED 2003*, pages 29–34. IEEE, 2003.

[80] M. J. Roemer, C. Hong, and S. H. Hesler. Machine Health Monitoring and Life Management Using Finite-Element-Based Neural Networks. *Journal of Engineering for Gas Turbines and Power*, 118(4):830–835, 10 1996. ISSN 0742-4795.

[81] Yimin Fan and C James Li. Diagnostic rule extraction from trained feedforward neural networks. *Mechanical Systems and Signal Processing*, 16(6):1073–1081, 2002.

[82] EC Larson, DP Wipf, and BE Parker. Gear and bearing diagnostics using neural network-based amplitude and phase demodulation. *A Critical Link: Diagnosis to Prognosis*, pages 511–521, 1997.

[83] Bo Li, M-Y Chow, Yodyium Tipsuwan, and James C Hung. Neural-network-based motor rolling bearing fault diagnosis. *IEEE transactions on industrial electronics*, 47(5):1060–1069, 2000.

[84] D-M Yang, AF Stronach, P MacConnell, and J Penman. Third-order spectral techniques for the diagnosis of motor bearing condition using artificial neural networks. *Mechanical systems and signal processing*, 16(2-3):391–411, 2002.

[85] BA Paya, II Esat, and MNM Badi. Artificial neural network based fault diagnostics of rotating machinery using wavelet transforms as a preprocessor. *Mechanical systems and signal processing*, 11(5):751–765, 1997.

[86] Timo Sorsa, Heikki N Koivo, and Hannu Koivisto. Neural networks in process fault diagnosis. *IEEE Transactions on systems, man, and cybernetics*, 21(4):815–825, 1991.

[87] Krishna Mohan Mishra and Kalevi J Huhtala. Fault detection of elevator systems using multilayer perceptron neural network. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 904–909. IEEE, 2019.

[88] Kingshuk Chatterjee, Subham Dawn, Vinay K Jadoun, and RK Jarial. Novel prediction-reliability based graphical dga technique using multi-layer perceptron network & gas ratio combination algorithm. *IET Science, Measurement & Technology*, 13(6):836–842, 2019.

[89] Dawei W Dong, John J Hopfield, and KP Unnikrishnan. Neural networks for engine fault diagnostics. In *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, pages 636–644. IEEE, 1997.

[90] C James Li and Tung-Yung Huang. Automatic structure and parameter training methods for modeling of mechanical systems by recurrent neural networks. *Applied Mathematical Modelling*, 23(12):933–944, 1999.

[91] Tim De Bruin, Kim Verbert, and Robert Babuška. Railway track circuit fault diagnosis using recurrent neural networks. *IEEE transactions on neural networks and learning systems*, 28(3):523–533, 2016.

[92] Hadi Shahnazari, Prashant Mhaskar, John M House, and Timothy I Salsbury. Modeling and fault diagnosis design for hvac systems using recurrent neural networks. *Computers & Chemical Engineering*, 126:189–203, 2019.

[93] Zenghui An, Shunming Li, Jinrui Wang, and Xingxing Jiang. A novel bearing intelligent fault diagnosis framework under time-varying working conditions using recurrent neural network. *ISA transactions*, 100:155–170, 2020.

[94] Rui Yang, Mengjie Huang, Qidong Lu, and Maiying Zhong. Rotating machinery fault diagnosis using long-short-term memory recurrent neural network. *IFAC-PapersOnLine*, 51(24):228–232, 2018.

[95] Duy-Tang Hoang and Hee-Jun Kang. Rolling element bearing fault diagnosis using convolutional neural network and vibration image. *Cognitive Systems Research*, 53: 42–50, 2019.

[96] Zhuyun Chen, Konstantinos Gryllias, and Weihua Li. Mechanical fault diagnosis using convolutional neural networks and extreme learning machine. *Mechanical Systems and Signal Processing*, 133:106272, 2019.

[97] Pei Cao, Shengli Zhang, and Jiong Tang. Preprocessing-free gear fault diagnosis using small datasets with deep convolutional neural network-based transfer learning. *Ieee Access*, 6:26241–26253, 2018.

[98] Moslem Azamfar, Jaskaran Singh, Inaki Bravo-Imaz, and Jay Lee. Multisensor data fusion for gearbox fault diagnosis using 2-d convolutional neural network and motor current signature analysis. *Mechanical Systems and Signal Processing*, 144: 106861, 2020.

[99] Hossein MehdipourPicha, Rui Bo, Haotian Chen, Md Masud Rana, Jie Huang, and Fengkai Hu. Transformer fault diagnosis using deep neural network. In *2019 IEEE Innovative Smart Grid Technologies-Asia (ISGT Asia)*, pages 4241–4245. IEEE, 2019.

[100] Haidong Shao, Hongkai Jiang, Xun Zhang, and Maogui Niu. Rolling bearing fault diagnosis using an optimization deep belief network. *Measurement Science and Technology*, 26(11):115002, 2015.

[101] Chaolong Zhang, Yigang He, Lifeng Yuan, and Sheng Xiang. Analog circuit incipient fault diagnosis method using dbn based features extraction. *IEEE Access*, 6: 23053–23064, 2018.

[102] Daming Lin and Viliam Makis. Recursive filters for a partially observable system subject to random failure. *Advances in Applied Probability*, pages 207–227, 2003.

[103] Hoon Sohn, Charles R Farrar, Francois M Hemez, Gyuhae Park, Amy N Robertson, and Todd O Williams. A coupled approach to developing damage prognosis solutions. In *Key Engineering Materials*, volume 245, pages 289–306. Trans Tech Publ, 2003.

[104] Xiang Li, Qian Ding, and Jian-Qiao Sun. Remaining useful life estimation in prognostics using deep convolution neural networks. *Reliability Engineering & System Safety*, 172:1–11, 2018.

[105] YSCTS Li, S Billington, C Zhang, T Kurfess, S Danyluk, and S Liang. Adaptive prognostics for rolling element bearing condition. *Mechanical systems and signal processing*, 13(1):103–113, 1999.

[106] Y Li, TR Kurfess, and SY Liang. Stochastic prognostics for rolling element bearings. *Mechanical Systems and Signal Processing*, 14(5):747–762, 2000.

[107] Charles H Oppenheimer and Kenneth A Loparo. Physically based diagnosis and prognosis of cracked rotor shafts. In *Component and Systems Diagnostics, Prognostics, and Health Management II*, volume 4733, pages 122–132. International Society for Optics and Photonics, 2002.

[108] C Cempel, HG Natke, and M Tabaszewski. A passive diagnostic experiment with ergodic properties. *Mechanical systems and signal processing*, 11(1):107–117, 1997.

[109] Jing Qiu, Brij B Seth, Steven Y Liang, and Cheng Zhang. Damage mechanics approach for bearing lifetime prognostics. *Mechanical systems and signal processing*, 16(5):817–829, 2002.

[110] KB Goode, J Moore, and BJ Roylance. Plant machinery working life prediction method utilizing reliability and condition-monitoring data. *Proceedings of the Institution of Mechanical Engineers, Part E: Journal of Process Mechanical Engineering*, 214(2):109–122, 2000.

[111] Jihong Yan, Muammer Koc, and Jay Lee. A prognostic algorithm for machine performance assessment and its application. *Production Planning & Control*, 15 (8):796–801, 2004.

[112] Ethan Phelps, Peter K Willett, and Thiagalingam Kirubarajan. Statistical approach to prognostics. In *Component and Systems Diagnostics, Prognosis, and Health Management*, volume 4389, pages 23–34. International Society for Optics and Photonics, 2001.

[113] Pieter-Jan Vlok, Maciej Wnek, and Maciej Zygmunt. Utilising statistical residual life estimates of bearings to quantify the influence of preventive maintenance actions. *Mechanical systems and signal processing*, 18(4):833–847, 2004.

[114] Chiman Kwan, Xiaodong Zhang, Roger Xu, and Leonard Haynes. A novel approach to fault diagnostics and prognostics. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 604–609. IEEE, 2003.

[115] Shao Haidong, Cheng Junsheng, Jiang Hongkai, Yang Yu, and Wu Zhantao. Enhanced deep gated recurrent unit and complex wavelet packet energy moment entropy for early fault prognosis of bearing. *Knowledge-Based Systems*, 188:105022, 2020.

[116] Akhand Rai and Sanjay H Upadhyay. Intelligent bearing performance degradation assessment and remaining useful life prediction based on self-organising map and support vector regression. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 232(6):1118–1132, 2018.

[117] Bing-Yu Sun, Zhi-Hua Zhu, Jiuyong Li, and Bin Linghu. Combined feature selection and cancer prognosis using support vector machine regression. *IEEE/ACM transactions on computational biology and bioinformatics*, 8(6):1671–1677, 2010.

[118] Tarak Benkedjouh, Kamal Medjaher, Noureddine Zerhouni, and Saïd Rechak. Health assessment and life prediction of cutting tools based on support vector regression. *Journal of Intelligent Manufacturing*, 26(2):213–223, 2015.

[119] Chengliang Li, Zhongsheng Wang, Shuhui Bu, Hongkai Jiang, and Zhenbao Liu. A novel method based on least squares support vector regression combing with strong tracking particle filter for machinery condition prognosis. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 228(6):1048–1062, 2014.

[120] Yuan Xie and Tao Zhang. The application of echo state network and recurrent multilayer perceptron in rotating machinery fault prognosis. In *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pages 2286–2291. IEEE, 2016.

[121] Wilson Q Wang, M Farid Golnaraghi, and Fathy Ismail. Prognosis of machine health condition using neuro-fuzzy systems. *Mechanical Systems and Signal Processing*, 18(4):813–831, 2004.

[122] Rui Guo, Yongtao Li, Lijiang Zhao, Jingyi Zhao, and Dianrong Gao. Remaining useful life prediction based on the bayesian regularized radial basis function neural network for an external gear pump. *IEEE Access*, 8:107498–107509, 2020.

[123] RCM Yam, PW Tse, L Li, and P Tu. Intelligent predictive decision support system for condition-based maintenance. *The International Journal of Advanced Manufacturing Technology*, 17(5):383–391, 2001.

[124] Jiujian Wang, Guilin Wen, Shaopu Yang, and Yongqiang Liu. Remaining useful life estimation in prognostics using deep bidirectional lstm neural network. In *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, pages 1037–1042. IEEE, 2018.

[125] Jichao Hong, Zhenpo Wang, and Yongtao Yao. Fault prognosis of battery system based on accurate voltage abnormity prognosis using long short-term memory neural networks. *Applied Energy*, 251:113381, 2019.

[126] Min Xia, Teng Li, Tongxin Shu, Jiafu Wan, Clarence W De Silva, and Zhongren Wang. A two-stage approach for the remaining useful life prediction of bearings using deep neural networks. *IEEE Transactions on Industrial Informatics*, 15(6): 3703–3711, 2018.

[127] Yongzhi Zhang, Rui Xiong, Hongwen He, and Michael G Pecht. Long short-term memory recurrent neural network for remaining useful life prediction of lithium-ion batteries. *IEEE Transactions on Vehicular Technology*, 67(7):5695–5705, 2018.

[128] Global predictive maintenance market. URL https://iot-analytics.com/predictive-maintenance-companies-landscape-2019/. Accessed: 2020-09-06.

[129] TE Beacham. High-pressure gear pumps. *Proceedings of the Institution of Mechanical Engineers*, 155(1):417–452, 1946.

[130] RH Frith and W Scott. Comparison of an external gear pump wear model with test data. *Wear*, 196(1-2):64–71, 1996.

[131] W Wustmann, S Helduser, and W Wimmer. Cfd-simulation of the reversing process in external gear pumps. In *6th International fluid power conference, Dresden*, volume 31, pages 03–02, 2008.

[132] Andrea Vacca and Marco Guidetti. Modelling and experimental validation of external spur gear machines for fluid power applications. *Simulation Modelling Practice and Theory*, 19(9):2007–2031, 2011.

[133] D Del Campo, R Castilla, GA Raush, PJ Gamez Montero, and E Codina. Numerical analysis of external gear pumps including cavitation. *Journal of fluids engineering*, 134(8), 2012.

[134] Jafar Ghazanfarian and D Ghanbari. Computational fluid dynamics investigation of turbulent flow inside a rotary double external gear pump. *Journal of Fluids Engineering*, 137(2), 2015.

[135] R Castilla, PJ Gamez-Montero, N Ertürk, A Vernet, M Coussirat, and E Codina. Numerical simulation of turbulent flow in the suction chamber of a gearpump using deforming mesh and mesh replacement. *International Journal of Mechanical Sciences*, 52(10):1334–1342, 2010.

[136] Hb Ding, FC Visser, Y Jiang, and M Furmanczyk. Demonstration and validation of a 3d cfd simulation tool predicting pump performance and cavitation for industrial applications. *Journal of fluids engineering*, 133(1):011101, 2011.

[137] Feng Qi, Sujan Dhar, Varun Haresh Nichani, Chiranth Srinivasan, De Ming Wang, Liang Yang, Zhonghui Bing, and Jinming Jim Yang. A cfd study of an electronic hydraulic power steering helical external gear pump: Model development, validation and application. *SAE International Journal of Passenger Cars-Mechanical Systems*, 9(2016-01-1376):346–352, 2016.

[138] Influence of approaches in cfd solvers on performance prediction in screw compressors. In *International compressor Engineering conference*.

[139] Javier Bonet, Antonio J Gil, and Richard D Wood. *Nonlinear continuum mechanics for finite element analysis:Statics*. Cambridge university press, 2016.

[140] J Donea, Antonio Huerta, J-Ph Ponthot, and A Rodriguez-Ferran. Arbitrary lagrangian-eulerian methods, volume 1 of encyclopedia of computational mechanics, chapter 14. *John Wiley & Sons Ltd*, 3:1–25, 2004.

[141] Andrew J Barlow, Pierre-Henri Maire, William J Rider, Robert N Rieben, and Mikhail J Shashkov. Arbitrary lagrangian eulerian methods for modeling high-speed compressible multimaterial flows. *Journal of Computational Physics*, 322: 603–665, 2016.

[142] Yves Dellanoy and J. L. Kueny. Two phase flow approach in unsteady cavitation modeling. 1990.

[143] Jean Decaix and Eric Goncalves. Compressible effects modeling in turbulent cavitating flows. *European Journal of Mechanics-B/Fluids*, 39:11–31, 2013.

[144] Zellman Warhaft. *An introduction to thermal-fluid engineering: the engine and the atmosphere*. Cambridge University Press, 1997.

[145] A. J. Favre. Review on space-time correlations in turbulent fluids. *Journal of Applied Mechanics*, 32(2):241–257, 06 1965.

[146] Thomas B Gatski and Jean-Paul Bonnet. *Compressibility, turbulence and high speed flow*. Academic Press, 2013.

[147] David C Wilcox et al. *Turbulence modeling for CFD*, volume 2. DCW industries La Canada, CA, 1998.

[148] Brian Edward Launder and Bahrat I Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in heat and mass transfer*, 1(2):131–137, 1974.

[149] T-H Shih, William W Liou, Aamir Shabbir, Zhigang Yang, and Jiang Zhu. A new k-epsilon eddy viscosity model for high reynolds number turbulent flows: Model development and validation. *NASA Sti/recon Technical Report N*, 95:11442, 1994.

[150] VSASTBCG Yakhot, SA Orszag, Siva Thangam, TB Gatski, and CG Speziale. Development of turbulence models for shear flows by a double expansion technique. *Physics of Fluids A: Fluid Dynamics*, 4(7):1510–1520, 1992.

[151] Emma Frosina, Adolfo Senatore, Dario Buono, Federico Monterosso, Micaela Olivetti, Luigi Arnone, et al. A tridimensional cfd analysis of the lubrication circuit of a non-road application diesel engine. Technical report, SAE Technical Paper, 2013.

[152] Brian Edward Launder and Dudley Brian Spalding. The numerical computation of turbulent flows. In *Numerical prediction of flow, heat transfer, turbulence and combustion*, pages 96–116. Elsevier, 1983.

[153] Joseph Boussinesq. *Essai sur la théorie des eaux courantes.* Wentworth Press, 1877.

[154] Stewart Glegg and William Devenport. *Aeroacoustics of low Mach number flows: fundamentals, analysis, and measurement.* Academic Press, 2017.

[155] Ashok K Singhal, Mahesh M Athavale, Huiying Li, and Yu Jiang. Mathematical basis and validation of the full cavitation model. *Journal of fluids engineering*, 124 (3):617–624, 2002.

[156] E Brennen Christopher. *Cavitation and bubble dynamics.* Oxford University Press, 1995.

[157] Masao Watanabe and Andrea Prosperetti. The effect of gas diffusion on the nuclei population downstream of a cavitation zone. Technical report, American Society of Mechanical Engineers, New York, NY (United States), 1994.

[158] Nigel P Weatherill and Oubay Hassan. Efficient three-dimensional delaunay triangulation with automatic point creation and imposed boundary constraints. *International Journal for Numerical Methods in Engineering*, 37(12):2005–2039, 1994.

[159] Michael Schäfer. *Computational engineering: Introduction to numerical methods*, volume 321. Springer, 2006.

[160] Charles Hirsch. *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics.* Elsevier, 2007.

[161] Suhas Patankar. *Numerical heat transfer and fluid flow.* Taylor & Francis, 2018.

[162] Ryoichi Amano and Bengt Sundén. *Computational fluid dynamics and heat transfer: emerging topics*, volume 23. WIT Press, 2011.

[163] Hrvoje Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows.* PhD thesis, Imperial College London (University of London), 1996.

[164] Robert Eymard, Thierry Gallouët, and Raphaèle Herbin. Finite volume methods. *Handbook of numerical analysis*, 7:713–1018, 2000.

[165] Emma Frosina, Adolfo Senatore, Dario Buono, Michele Pavanetto, Micaela Olivetti, and Ina Costin. Improving the performance of a two way flow control valve, using a 3d cfd modeling. In *ASME 2014 International Mechanical Engineering Congress and Exposition.* American Society of Mechanical Engineers Digital Collection, 2014.

[166] Hermann Schlichting and Klaus Gersten. *Boundary-layer theory.* Springer, 2016.

[167] Wu-Gui Jiang, Ren-Zhi Zhong, Qing H Qin, and Yong-Gang Tong. Homogenized finite element analysis on effective elastoplastic mechanical behaviors of composite with imperfect interfaces. *International journal of molecular sciences*, 15(12): 23389–23407, 2014.

[168] Vijay Kotu and Bala Deshpande. *Data Science: Concepts and Practice.* Morgan Kaufmann, 2018.

[169] Hong Jia, Yiu-ming Cheung, and Jiming Liu. A new distance metric for unsupervised learning of categorical data. *IEEE transactions on neural networks and learning systems*, 27(5):1065–1079, 2015.

[170] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning.* MIT press, 2018.

[171] Roberto Andreani and Paulo JS Silva. Constant rank constraint qualifications: a geometric introduction. *Pesquisa Operacional*, 34(3):481–494, 2014.

[172] Hwanjo Yu and Sungchul Kim. Svm tutorial-classification, regression and ranking. *Handbook of Natural computing*, 1:479–506, 2012.

[173] Guido F Smits and Elizabeth M Jordaan. Improved svm regression using mixtures of kernels. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, volume 3, pages 2785–2790. IEEE, 2002.

[174] Ke-Lin Du and Madisetti NS Swamy. *Neural networks and statistical learning.* Springer Science & Business Media, 2013.

[175] Tom Mithchell. *Machine Learning.* McGraw-Hill Computer science series, 1997.

[176] Yuhuang Hu, Adrian Huber, Jithendar Anumula, and Shih-Chii Liu. Overcoming the vanishing gradient problem in plain recurrent networks. *arXiv preprint arXiv:1801.06105*, 2018.

[177] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[178] Dinggang Shen, Guorong Wu, and Heung-Il Suk. Deep learning in medical image analysis. *Annual review of biomedical engineering*, 19:221–248, 2017.

[179] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA:, 2015.

[180] Martin T Hagan, Howard B Demuth, and Mark Beale. *Neural network design.* PWS Publishing Co., 1997.

[181] Kevin P Murphy. *Machine learning: a probabilistic perspective.* MIT press, 2012.

[182] Selina SY Ng, Yinjiao Xing, and Kwok L Tsui. A naive bayes model for robust remaining useful life prediction of lithium-ion battery. *Applied Energy*, 118:114–123, 2014.

[183] Aritz Pérez, Pedro Larrañaga, and Iñaki Inza. Bayesian classifiers based on kernel density estimation: Flexible classifiers. *International Journal of Approximate Reasoning*, 50(2):341–362, 2009.

[184] Geoffrey I Webb and Michael J Pazzani. Adjusted probability naive bayesian induction. In *Australian Joint Conference on Artificial Intelligence*, pages 285–295. Springer, 1998.

[185] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[186] Antonio Candelieri, Ilaria Giordani, Francesco Archetti, Konstantin Barkalov, Iosif Meyerov, Alexey Polovinkin, Alexander Sysoyev, and Nikolai Zolotykh. Tuning hyperparameters of a svm-based water demand forecasting system through parallel global optimization. *Computers & Operations Research*, 106:202–209, 2019.

[187] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, pages 2546–2554, 2011.

[188] Jacques Wainer and Gavin Cawley. Empirical evaluation of resampling procedures for optimising svm hyperparameters. *The Journal of Machine Learning Research*, 18(1):475–509, 2017.

[189] Yingsheng Ji, Yushu Chen, Haohuan Fu, and Guangwen Yang. An enkf-based scheme to optimize hyper-parameters and features for svm classifier. *Pattern Recognition*, 62:202–213, 2017.

[190] Panagiotis Tsirikoglou, Simon Abraham, Francesco Contino, Chris Lacor, and Ghader Ghorbaniasl. A hyperparameters selection technique for support vector regression models. *Applied Soft Computing*, 61:139–148, 2017.

[191] R Laref, E Losson, A Sava, and M Siadat. On the optimization of the support vector machine regression hyperparameters setting for gas sensors array applications. *Chemometrics and Intelligent Laboratory Systems*, 184:22–27, 2019.

[192] Shengfa Yuan and Fulei Chu. Fault diagnosis based on support vector machines with parameter optimisation by artificial immunisation algorithm. *Mechanical Systems and Signal Processing*, 21(3):1318–1330, 2007.

[193] Ilhan Aydin, Mehmet Karakose, and Erhan Akin. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. *Applied soft computing*, 11(1):120–129, 2011.

[194] Inc. The MathWorks. *Statistics and Machine Learning Toolbox*. Natick, Massachusetts, USA, 2020. URL https://uk.mathworks.com/help/stats/index.html.

[195] Saurabh Karsoliya. Approximating number of hidden layer neurons in multiple hidden layer bpnn architecture. *International Journal of Engineering Trends and Technology*, 3(6):714–717, 2012.

[196] K Gnana Sheela and Subramaniam N Deepa. Review on methods to fix number of hidden neurons in neural networks. *Mathematical Problems in Engineering*, 2013, 2013.

[197] Jinchuan Ke and Xinzhe Liu. Empirical analysis of optimal hidden neurons in neural network modeling for stock prediction. In *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, volume 2, pages 828–832. IEEE, 2008.

[198] Zvi Boger and Hugo Guterman. Knowledge extraction from artificial neural network models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, volume 4, pages 3030–3035. IEEE, 1997.

[199] Gordon S Linoff and Michael JA Berry. *Data mining techniques: for marketing, sales, and customer relationship management.* John Wiley & Sons, 2011.

[200] Jay Lee, Ramzi Abujamra, Andrew KS Jardine, Daming Lin, and Dragan Banjevic. An integrated platform for diagnostics, prognostics and maintenance optimization. *Proceedings of the intelligent maintenance systems*, pages 15–27, 2004.

[201] Howard Austerlitz. *Data acquisition techniques using PCs.* Academic press, 2002.

[202] Nikolay V Kirianaki, Sergey Y Yurish, Nestor O Shpak, and Vadim P Deynega. *Data acquisition and signal processing for smart sensors.* Wiley Chichester, 2002.

[203] Nagi Z Gebraeel, Mark A Lawley, Rong Li, and Jennifer K Ryan. Residual-life distributions from component degradation signals: A bayesian approach. *IiE Transactions*, 37(6):543–557, 2005.

[204] Jason Deutsch and David He. Using deep learning-based approach to predict remaining useful life of rotating components. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(1):11–20, 2018.

[205] Erkki Jantunen, Jan-Otto Hooghoudt, Yi Yang, and Mark McKay. Predicting the remaining useful life of rolling element bearings. In *2018 IEEE International Conference on Industrial Technology (ICIT)*, pages 2035–2040. IEEE, 2018.

[206] Charles M Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968.

[207] Stavros Tsakalidis, Vlasios Doumpiotis, and William Byrne. Discriminative linear transforms for feature normalization and speaker adaptation in hmm estimation. *IEEE Transactions on Speech and Audio Processing*, 13(3):367–376, 2005.

[208] Silvia Cateni, Marco Vannucci, Marco Vannocci, and Valentina Colla. Variable selection and feature extraction through artificial intelligence techniques. *Multivariate Analysis in Management, Engineering and the Science*, pages 103–118, 2012.

[209] Isabelle Guyon, Steve Gunn, Masoud Nikravesh, and Lofti A Zadeh. *Feature extraction: foundations and applications*, volume 207. Springer, 2008.

[210] Yaguo Lei, Zhengjia He, Yanyang Zi, and Xuefeng Chen. New clustering algorithm-based fault diagnosis using compensation distance evaluation technique. *Mechanical Systems and Signal Processing*, 22(2):419–435, 2008.

[211] Valeriu Vrabie, Pierre Granjon, and Christine Serviere. Spectral kurtosis: from definition to application. In *6th IEEE International Workshop on Nonlinear Signal and Image Processing.*

[212] Taiyong Li and Min Zhou. Ecg classification using wavelet packet entropy and random forests. *Entropy*, 18(8):285, 2016.

[213] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3):379–423, 1948.

[214] Funa Zhou, Shuai Yang, Hamido Fujita, Danmin Chen, and Chenglin Wen. Deep learning fault diagnosis method based on global optimization gan for unbalanced data. *Knowledge-Based Systems*, 187:104837, 2020.

[215] Mohammad Zawad Ali, Md Nasmus Sakib Khan Shabbir, Shafi Md Kawsar Zaman, and Xiaodong Liang. Single-and multi-fault diagnosis using machine learning for variable frequency drive-fed induction motors. *IEEE Transactions on Industry Applications*, 56(3):2324–2337, 2020.

[216] Ziad Obermeyer and Ezekiel J Emanuel. Predicting the future—big data, machine learning, and clinical medicine. *The New England journal of medicine*, 375(13): 1216, 2016.

[217] Derek T Ahneman, Jesús G Estrada, Shishi Lin, Spencer D Dreher, and Abigail G Doyle. Predicting reaction performance in c–n cross-coupling using machine learning. *Science*, 360(6385):186–190, 2018.

[218] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[219] Peter Ebbes, Dominik Papies, and Harald J Van Heerde. The sense and non-sense of holdout sample validation in the presence of endogeneity. *Marketing Science*, 30 (6):1115–1122, 2011.

[220] C Saranya and G Manikandan. A study on normalization techniques for privacy preserving data mining. *International Journal of Engineering and Technology (IJET)*, 5(3):2701–2704, 2013.

[221] Foram S Panchal and Mahesh Panchal. Review on methods of selecting number of hidden nodes in artificial neural network. *International Journal of Computer Science and Mobile Computing*, 3(11):455–464, 2014.

[222] François Chollet et al. Keras, 2015. URL `https://keras.io`. Accessed: 2020-09-06.

[223] Kato Mivule. Utilizing noise addition for data privacy, an overview. 07 2012.

[224] Inc. The MathWorks. *Communications Toolbox*. Natick, Massachusetts, United State, 2019.

[225] David R Pauluzzi and Norman C Beaulieu. A comparison of snr estimation techniques for the awgn channel. *IEEE Transactions on communications*, 48(10):1681–1691, 2000.

[226] Han Li, Wei Zhao, Yuxi Zhang, and Enrico Zio. Remaining useful life prediction using multi-scale deep convolutional neural network. *Applied Soft Computing*, 89: 106113, 2020.

[227] Zaixu Cui and Gaolang Gong. The effect of machine learning regression algorithms and sample size on individualized behavioral prediction with functional connectivity features. *Neuroimage*, 178:622–637, 2018.

[228] Giorgio Altare and Massimo Rundo. Computational fluid dynamics analysis of gerotor lubricating pumps at high-speed: geometric features influencing the filling capability. *Journal of Fluids Engineering*, 138(11), 2016.

[229] Ebru Turanoglu Bekar, Per Nyqvist, and Anders Skoogh. An intelligent approach for data pre-processing and analysis in predictive maintenance with an industrial case study. *Advances in Mechanical Engineering*, 12(5):1687814020919207, 2020.

[230] Steven R Young, Derek C Rose, Thomas P Karnowski, Seung-Hwan Lim, and Robert M Patton. Optimizing deep learning hyper-parameters through an evolutionary algorithm. In *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, pages 1–5, 2015.

[231] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[232] W. Stanley, G. Dougherty, R. Dougherty, and H. Saunders. Digital signal processing (2nd edition). *Journal of Vibration and Acoustics-transactions of The Asme*, 110:126–127, 1988.

[233] Suranai Poungponsri and Xiao-Hua Yu. An adaptive filtering approach for electrocardiogram (ecg) signal noise reduction using neural networks. *Neurocomputing*, 117:206–213, 2013.

[234] Chuan Jiang and Samuel H Huang. A computationally efficient and adaptive approach for online embedded machinery diagnosis in harsh environments. *Advances in Mechanical Engineering*, 5:847612, 2013.

[235] Javier Diaz-Rozo, Concha Bielza, and Predro Larranaga. Machine learning-based cps for clustering high throughput machining cycle conditions. *Procedia Manuf*, 10:997–1008, 2017.

[236] Nagdev Amruthnath and Tarun Gupta. Fault class prediction in unsupervised learning using model-based clustering approach. In *2018 International Conference on Information and Computer Technologies (ICICT)*, pages 5–12. IEEE, 2018.

[237] Yarin Gal. *Uncertainty in deep learning.* PhD thesis, University of Cambridge, 2016.

[238] Jeffrey P Van Doormaal and George D Raithby. Enhancements of the simple method for predicting incompressible fluid flows. *Numerical heat transfer*, 7(2): 147–163, 1984.

[239] Douglas L Sondak. *Wall functions for the k-[epsilon] turbulence model in generalized nonorthogonal curvilinear coordinates.* PhD thesis, Iowa State University, 1992.

[240] Emmanuel C Ifeachor and Barrie W Jervis. *Digital signal processing: a practical approach.* Pearson Education, 2002.

[241] Christopher M Leavey, M Neil James, John Summerscales, and Robert Sutton. An introduction to wavelet transforms: a tutorial approach. *Insight-Non-Destructive Testing and Condition Monitoring*, 45(5):344–353, 2003.

[242] Vidakovic Brani. *Statistical Modelling by Wavelets.* Wiley Series in Probability and Statistics, 1999.

[243] Stephane G Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12): 2091–2110, 1989.