# Detection of Faults in Subsea Crude Oil Pipelines by Machine Learning Assisted Process Monitoring

By

Daniel Theodore Eastvedt

A thesis submitted to the School of Graduate Studies

in partial fulfillment of the requirements for the degree of

**Master of Engineering**

**Faculty of Engineering and Applied Science**

Memorial University of Newfoundland

May 2021

St. John's Newfoundland Canada

# Abstract

Crude oil pipelines are a major infrastructure investment for subsea oilfields and for platform-to-shore transportation and continue to be the preferred method of transporting crude oil over short to medium distances due to low operating costs and high safety record compared to other transport methods. Their remote and often extreme locations limit their access and increases risks of human and environmental hazards in the event of failure. Due to their frequent placement in deep water and hostile environments, inspection and identification of faults (such as leaks and flow restrictions) are difficult, expensive, and hazardous. Faults are often only identified in accidents or upon routine inspection and after significant material losses and/or environmental damage has occurred. The oil industry would benefit from a low-cost and timely method of fault detection. This thesis proposes such a method by augmenting process monitoring with Machine Learning (ML).

This thesis investigates the relationship between pressure change, velocity change, and temperature of crude oil through a pipeline. A representative dataset of crude oil flow is generated by computational fluid dynamics (CFD) and used to train a ML algorithm to develop a model of fluid behavior under normal pipeline operations over a range of typical flow rates and temperatures. CFD data are then collected under several simulated fault conditions: leaks of 10 and 20% of the inner cross-sectional area of the pipe, and a restriction to flow of 50% of the cross-sectional area. This thesis demonstrates that the ML algorithm can be trained to model the system under normal conditions, thereby successfully recognizing a fault condition as non-conforming and indicative of a statistically significant change in pipeline operation. It is further able to identify the fault type based on the pattern observed in the new data.

This work demonstrates that ML may be a low-risk, low-cost, and accurate method of monitoring a subsea crude oil pipeline for optimal performance and fault detection without the need to introduce special equipment to a subsea pipeline network, assuming flowmeters and temperature probes are employed for process monitoring. This thesis develops the model algorithm, and it is hoped that the results of this study provide a basis for the integration of machine learning and further “big data” techniques in loss prevention and health and environmental protection in the offshore oil industry and elsewhere.

## Acknowledgments

I would like to sincerely express my appreciation and gratitude to my supervisors, Dr. Xili Duan and Dr. Greg Naterer, for their counsel, support, and interest in my project throughout the process of my research and education. They took the time to meet with me in person upon expressing my interest in joining their research team and found a project that matched my interests. They strongly encouraged me to learn programming skills and ML coding both at Memorial University and through online resources, and worked with my steep learning curve on these topics. I want to give special thanks to Dr. Duan for taking on the primary supervisory role and being incredibly flexible during the strange situation that the COVID-19 pandemic placed us in. And special thanks to Dr. Naterer, who took time out of his personal life to assist me with research goals and review of this report and others.

I would like to also thank my best friend Joseph Smokey, who provided advice and support on topics of graduate work, as well as making sure I remained healthy and entertained during COVID-19 lockdowns.

Finally, I want to thank my parents, Chuck and Linda Eastvedt, who have always supported me in my endeavors throughout my life, graduate work being no exception. I love you both greatly and appreciate all that you have done to educate, encourage, and support me.

# Table of Contents

# List of Tables

# List of Figures

# Nomenclature

| Symbol | Name [units] |
|---|---|
| **English** | |
| $a, b, c, d, i, j$ | constants |
| $A$ | cross-sectional area of pipe [$m^2$] |
| $C_v$ | specific heat capacity [J/(kg-K)] |
| $D$ | data-dependent statistical distribution |
| $g_z$ | acceleration due to gravity [$m/s^2$] |
| $h, h_{w,b}, h_i$ | predicted output; general, linear, incremental |
| $k$ | thermal conductivity [W/(m-K)] |
| $L$ | loss function |
| $p$ | static pressure [Pa, kg/(m-$s^2$)] |
| $\Delta p$ | pressure change, $p_1$-$p_2$ [Pa, kg/(m-$s^2$)] |
| $P$ | probability |
| $Q_{max}, Q_l$ | volumetric flow rate; maximum pipe and leak [$m^3/s$] |
| $r$ | radius [m] |
| $R$ | risk function |
| $S$ | training set |
| $S_T$ | test set |
| $t$ | time [s] |
| $T$ | temperature [°C] |
| $U, U_{max}$ | mean velocity; general and max [m/s] |
| $\Delta U$ | mean velocity change, $U_1$-$U_2$ [m/s] |

| | |
|---|---|
| $v_r$, $v_\theta$, $v_z$ | r, θ, and z component velocities [m/s] |
| $w$ | input parameters |
| $x$, $x_i$ | dataset inputs; general, incremental |
| $y$, $y_i$ | dataset outputs; general, incremental |
| $z$ | axial distance [m] |

**Greek**

| | |
|---|---|
| $\rho$ | density [kg/m$^3$] |
| $\mu$ | dynamic viscosity [kg/(m-s)] |
| $\varphi$ | dataset structure |
| $\eta$ | learning rate |
| $\theta$ | radial angle [rads] |

# CHAPTER 1: INTRODUCTION

## 1.1 Background of Subsea Pipelines

Subsea pipelines are important components in petroleum production and transportation. The safe and efficient operation of these pipelines is critical to minimizing risk to personnel, structural assets, and the environment. Petroleum in the form of crude oil continues to be a major source of energy and hydrocarbon products and will for the foreseeable future. The U.S. Energy Information Administration estimates that by 2025 global oil consumption will reach 102.4 million barrels of oil per day (mn b/d) then peak at approximately 106.3mn b/d by 2040, after which global demand is expected to slump as energy alternatives attain greater market share [Lauerman 2019]. While this represents an overall decline in the long-term outlook of oil consumption, an estimated 40mn b/d of new crude oil supply must come from offshore and shale sources to offset the depletion of current terrestrial and near-shore wells, necessitating exploitation of deeper and further off-shore reserves [McKinsey 2019].

In 2015 offshore production accounted for 29% of total global oil production and over the 2005-2015 period investment in deepwater (125-1500 meter depth) and ultra-deepwater (greater than 1500 meters) sources increased from approximately 6.7 to 7.9mn b/d and 0.31 to 1.7mn b/d, respectively, and this trend is expected to continue [Manning 2019]. At present, the great majority of the volumetric output of subsea oil wells is conveyed through pipelines; either as well-to-shore transport systems for near-shore operations, or as piping networks in off-shore and deepwater subsea systems. The UK Offshore Operators Association estimates that a total length of about 25,000 km of subsea pipelines with 1567 subsea pipes, and 542 failure events of these have been reported up to the year 2000 [Davis and Brockhurst 2015]. Deepwater systems typically consist of multiple structures incorporating pipeline networks: short “jumpers” that connect various

structures such as wellheads and manifolds; flowlines that transport oil from the manifolds to a riser; and the riser(s), which elevates the oil to the surface [Baker 1998, Palmer et al. 2008]. These networks may present an enormous piping infrastructure. As an example, Shell's Perdido Platform, the deepest operating well in the world at 2450 meters, consists of a 22 well network and 44 kilometers of oil pipelines [Pacella 2009].

Operational loss of a subsea pipeline can include flow reduction due to internal restrictions and containment failure. Pipeline failure consists of situations in which structural loss results in oil leakage into the surrounding sea environment. These range in severity from minor and barely detectable, to catastrophic. A very notable example of catastrophic loss is the April 2010 Deepwater Horizon incident in the Gulf of Mexico that resulted from the safety failure of a deep-water well-head. The resulting explosion killed 11 workers and the resulting release of oil devastated the environment and economic activity of the gulf coast [Levy and Gopalakrishnan 2010]. The great majority of subsea oil spills are relatively minor. Particularly in the case of corrosion failure leaks may take decades to form and many go unnoticed for long periods [Barker et al. 2014, Ilman and Kusmono 2008, Yun and Zhijun 2013]. Srivastava et al. (2008) describe a 30-inch pipeline at a depth of 80 meters that, after 25 years of service life, was found to have a 120 mm by 1250 mm rupture due to corrosion (approximately 30% of the cross-sectional area). Barker et al. (2014) document a case study of a North Sea subsea pipeline found to have sustained uniform corrosion and regions of severe localized corrosion leading to leaks that may have persisted unnoticed for up to five years.

As the need for deeper and further offshore development of subsea pipeline systems increases, so does the need for improved safety and real-time monitoring to avoid and minimize potentially costly and harmful failures of these pipelines. Process monitoring and routine

inspection are currently the primary means of identifying faults in subsea pipeline systems [Baker 1998, Fu et al. 2020, Lyons 2010, L. Yang et al. 2019, Palmer and King 2008, Sa'aadah et al. 2014]. Leaks are often identified only by incidental discovery when losses in flow rates are extreme, or upon visual discovery by remotely operated vehicle (ROV), diver, or sonar. Process monitoring is in general a system of using real-time data from an industrial, chemical, or manufacturing process for analysis and optimization. Process monitoring of flow parameters (e.g., pressure, temperature, volumetric flow rate) during operation aids industry personnel in identifying significant disruptions but is limited in its ability to identify small differences in operating conditions [Baker 1998, Lyons 2010, Palmer and King 2008].

## 1.2 Machine Learning

Machine learning has become in recent decades an effective tool in solving engineering problems and automating complicated data management processes. As described by Tom Mitchell, "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [Mitchell 1997]. Framed another way, ML is a program that can alter its programming (model) based on new information to better fit its data-environment, thus learning from experience (Figure 1.1). Machine learning is divided into three basic subcategories—supervised learning, unsupervised learning, and semi-supervised learning—and forms the logical basis for much of the field of artificial intelligence [Brunton et al. 2020, Mitchell 1997, Shalev-Shwartz and Ben-David 2014].

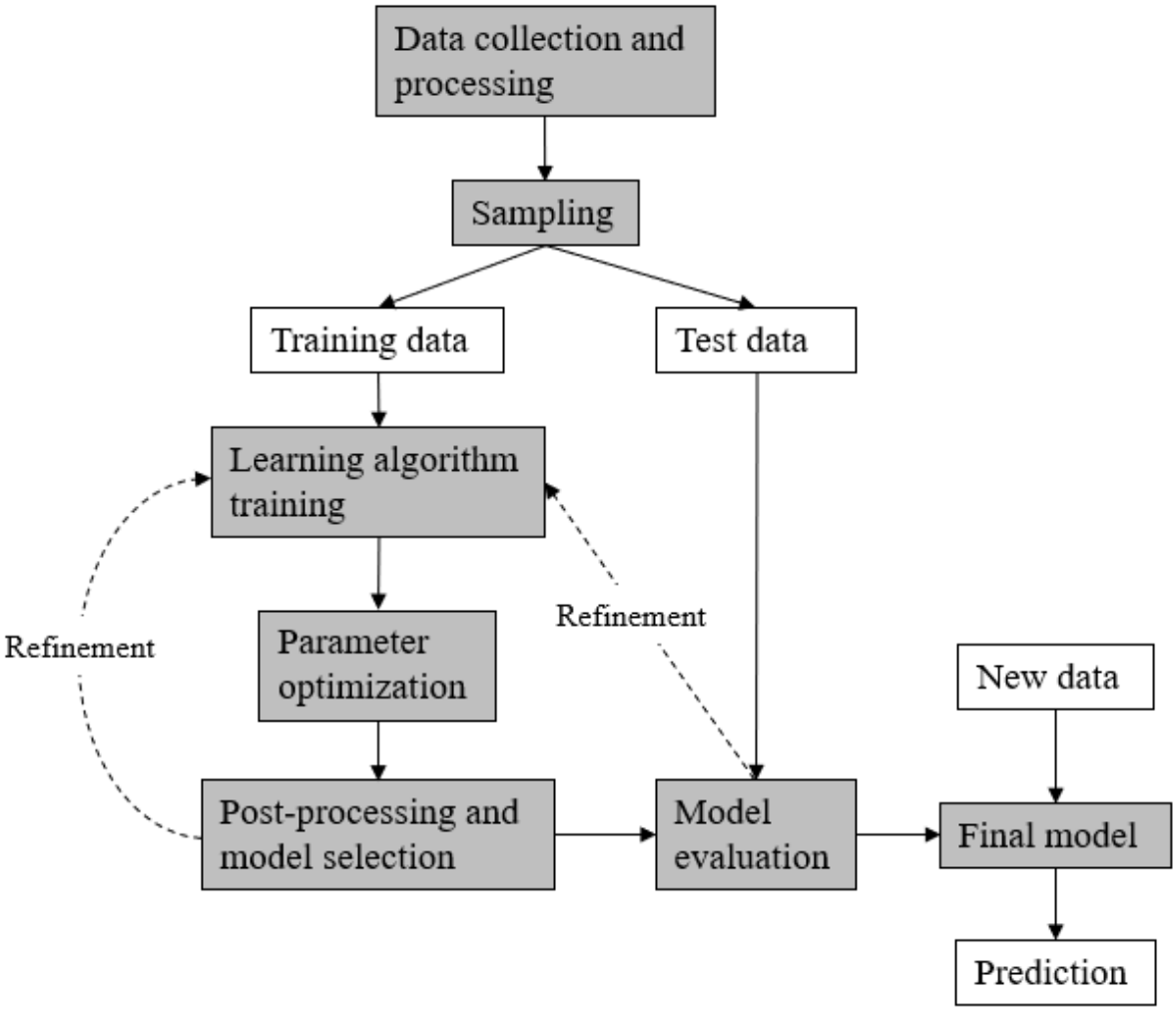



Fig. 1.1: Flowchart of a general supervised machine learning training process.

Supervised learning is the subset of machine learning in which a program is provided with a training set of real input(s) and output data (quantitative or qualitative) and a mathematical or probabilistic model is generated from this training set. This model is tested with a small dataset and refinements are made as necessary to limit generalization error, a measure of how capable a model is of predicting an outcome using new unseen data. Provided generalization error is reasonably low and the bias-variance trade-off is balanced, the final model is presented. The ML

program should then, when presented with a previously unseen input datum, be capable of predicting the corresponding output within a reasonable margin of error. When data is quantitative, data fitting takes the form of regression modelling and the fitting function through recursion identifies the most appropriate order and curve [Brunton et al. 2020, Mitchell 1997, Shalev-Shwartz and Ben-David 2014]. The application of machine learning in pattern recognition has become broad-reaching as it is useful for most forms of data and has been employed in many areas, such as recognizing spam email, price predictions, voice recognition, and manufacturing process control. As this thesis will outline ML has found numerous applications in engineering and in the oil and gas industry, but the full potential has not been realized in the field of process monitoring and fault detection. This thesis will seek to bridge that gap by introducing a program that employs two supervised regression ML algorithms that can learn the normal flow behaviors of a crude oil pipeline and use this learning to accurately identify non-conforming behavior as indicative of a fault or change in the fluid properties.

## 1.3 Objectives of the Thesis

This thesis aims to propose a novel method for monitoring a subsea crude oil pipeline for faults and changes in flow conditions using a supervised ML regression algorithm program that uses real data from a study system. This thesis achieves this goal in the following ways:

- A literature review identifies a gap in the current methodology for monitoring the flow regimes of subsea pipelines.
- Governing models in the underlying fluid mechanics are presented.
- A model pipeline system is simulated under normal operating conditions by computational fluid dynamics (CFD) in ANSYS, forming a set of training data.

- A ML program is trained under two algorithms (univariate linear and multivariate generalized linear regression) using the normal condition flow data generated by the CFD and validated through test data.
- CFD data is obtained under conditions of two types of simulated pipeline faults: leaks at 10% and 20% of nominal pipeline cross-sectional area, and a restriction to flow at 50% of cross-sectional area.
- The ML algorithm is introduced to this fault flow data and is capable of identifying them as abnormal conditions, and uses these data to identify the fault type and provide numerical estimates of severity.

This thesis, being limited in scope to the above conditions of study, serves as a proof of concept that a machine learning program could be developed and employed to detect faults in a subsea oil pipeline. This thesis does not explore the full range of defect types or degrees of severity that a system might experience.

## 1.4 Outline of the Thesis

This thesis presents the research in seven chapters and an appendix as briefly described below:

- Chapter 1. A brief introduction of the use of crude oil pipelines in subsea resource extraction and a brief overview of the concept of machine learning in data analysis.
- Chapter 2. A literature review of the current topics of subsea oil pipeline process monitoring and the present applications of machine learning there-in.
- Chapter 3. Background information on the governing equations of fluid mechanics and the principles of machine learning.

- Chapter 4. An explanation of the methods and assumptions of modeling through computational fluid dynamics the study pipeline under normal conditions and the fault conditions described.
- Chapter 5. An explanation of the ML program used in this thesis.
- Chapter 6. Results and discussion of the training of the ML program using the normal pipeline conditions and its ability to identify fault conditions and report these to the user.
- Chapter 7. The conclusions of this thesis, and recommendations for further research on this topic.
- Appendix A. The ML program coding in full.

# CHAPTER 2: LITERATURE REVIEW

Chapter 2 is divided into three primary sections: Section 2.1 provides a literature review of the fault prevention and process monitoring in subsea pipeline systems. Section 2.2 provides a review of ML: a basic overview, its history, and the applications of ML in engineering and the petroleum industry. Section 2.3 provides an explanation of the literature gap and the justification for the research performed in this thesis.

## 2.1 Pipeline Fault Prevention and Process Monitoring

The majority of failure prevention on subsea crude oil pipelines is performed in the design phase of pipelines and transport systems and involves route planning, material and physical parameter selection, and inspections during construction and installation of pipelines in the operating environment. During pipe design parameters such as material selection, wall thickness, insulation, and diameter are selected based on operational requirements (e.g., flow rates and pressures), the properties of the crude oil (viscosity, temperature, gas fraction, sand entrainment, waxing, etc), the subsea environment and depth, and route planning. Additional considerations are given to incidental stresses such as earthquakes, rock and mudslides, seabed erosion, and impacts from manmade objects (e.g., anchors and fishing nets) [Palmer and King 2008]. The efficacy of design is limited by the knowledge at the time of development and often the actual operational conditions or unexpected event-driven occurrences exceed design parameters. Approximately 38% of subsea pipeline failures occur due to third-party damage and 36% fail due to corrosion in service (Fig. 2.1) [Sa'aadah et al. 2014].

Risk analysis is a tool employed in identifying risk factors for mitigating risk in the design process and incidence response planning. Numerous authors have used historical data to develop probabilistic models in risk analysis. Y. Yang et al. (2017) developed an Object-Oriented Bayesian

Network (OOBN) that used historical data of corrosion occurrences to generate a probabilistic model of future occurrences and their likely locations in target pipeline systems. Li et al. (2016) employed a similar technique to predict the occurrence of leakage in subsea pipelines in general. Fuzzy reasoning algorithms have been proposed for employment in the design of thin-walled pipelines to predict the safety thresholds in the events of corrosion wear and dents [McDonald and Cosham 2005]. The post-failure analysis is another powerful tool employed to identify the root cause of failures after they occur and use the lessons learned in future projects. McDonald and Cosham (2005) have proposed several methodologies of failure and defect analysis using a combined in-lab and in-environment approach. Ilman and Kusmono (2008) provided a multi-analysis examination of a leak in a 27-year service life pipeline caused by electro-chemical and flow-induced corrosion in an elbow and suggested that attention to flow rates and water-phase separation in design could prevent future incidents.

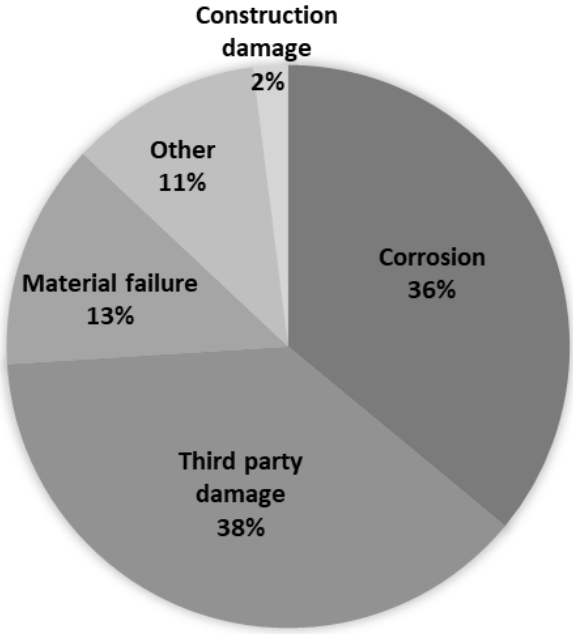



Fig. 2.1: Distribution of failure modes in subsea pipelines [Sa'aadah et al. 2014].

Process monitoring is in general a system of using real-time data from an industrial, chemical, or manufacturing process for analysis and optimization. It is a field that is interested in ensuring the quality and safety of the process of consideration and is a primary tool in fault detection or out-of-control (OOC) conditions. Process monitoring and routine inspection are currently the primary means of identifying faults in subsea pipeline systems [Baker 1998, Fu et al. 2020, Lyons 2010, L. Yang et al. 2019, Palmer and King 2008, Sa'aadah et al. 2014]. Process monitoring of flow parameters (e.g., pressure, temperature, volumetric flow rate) during operation aids industry personnel in identifying significant disruptions but is limited in its ability to distinguish minute differences in operating conditions [Baker 1998, Lyons 2010, Palmer and King 2008]. Minor (and some major) leaks are often identified only by incidental discovery when losses in line pressure or flow rates are extreme, or upon visual discovery of material loss by remotely operated vehicle, diver, or sonar, which are only as frequent as resources, expense, and operating environment permits [McDonald et al. 2005, Palmer and King 2008, X. Li et al. 2016].

Prihtiadi et al. (2016) demonstrated experimentally that a distributed system of pressure sensors in a water pipe could be used to identify the location of a leak by way of analyzing the pressure profiles for hallmark flow disturbances and gradient intersection of the positional pressures upstream and downstream of the leak (see Fig. 4.4). Of note, however, is their approach assumed relatively stable pressures as a function of time and operational fluid pipelines tend to vary considerably. Fu et al. (2020) proposed a method of leak detection in a fluid pipeline by way of monitoring dimensionless pressure drop and dimensionless leakage rate between two positions and supported the viability with experimental results. However, this did not include the effects of temperature on fluid properties and noted the scaling limitation of the method on real conditions. A computational fluid dynamics (CFD) and experimental study published by L. Yang et al. (2019)

proposed a similar dimensionless model to detect flow restrictions/blockage in a fluid pipeline. The authors of this study note that the predictive power is contingent upon calibration of the CFD simulation to the specific conditions of the blockage under study.

## 2.2 Machine Learning

### 2.2.1 Overview

Machine learning (ML) has become in recent decades an effective tool in solving engineering problems and automating complicated data management processes [Mitchell 1997, Shalev-Shwartz et al. 2014]. The application of machine learning in pattern recognition has become broad-reaching as it is useful for most forms of data and has been employed in many areas, including engineering and the petroleum industry. ML is a broad topic. Tom Mitchell (1997), a pioneer in this field, describes ML as such: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E." The key feature of ML is in a program's inherent ability to use data, make an interpretation or prediction of the nature or outcome of new "unseen" data based on previous experience, be able to measure its performance at interpreting or predicting, and learn from the results of that performance measurement. To put it simply, a ML program is capable of identifying how successful its algorithm is at understanding its data-space and adjust its algorithm in order to improve its understanding with little or no guidance from human programmers. This is in contrast to more conventional programming in which a computer program simply follows a set of pre-defined instructions or formulas and the relationship between data in and data out is rigid.

An example of a rigid programming structure related to this thesis would be solving the momentum equation (Eq. 3.6) for a uniform straight pipeline and temperature dependence of

viscosity for Brent crude oil and defining as many known parameters as possible, such as dimensions (length, radius) and fluid density. The resulting mathematical relationship between pressure change, velocity (U), and temperature (T) could then be used by a program to assess whether or not real pressure, velocity, and temperature data collected through empirical measurement conform to the mathematical relationship. In many cases, this approach is an acceptable approximation. It is subject, however, to many limitations: the properties of crude oil are highly variable from well to well; interactions between fluid and stationary wall are difficult to predict and rarely uniform; heat transfer characteristics of fluid and pipe in application may differ from assumptions; and numerous other limitations etc. A ML approach (and that taken by this thesis) would be to remove as many assumptions of the system of study as possible and train a program to define the relationship between pressure, velocity and temperature as it receives real data and define the mathematical model from the data, improve that model as its understanding of the real system grows with experience. This process greatly minimized the effect of errors in assumptions.

### 2.2.2 Historical Context

The earliest iterations of what could be considered machine learning date to the early 1940s when researchers, notably Alan Turing first proposed the use of electro-mechanical computers designed to emulate the human mind toward solving complex problems. The most notable example was the Bombe, an electromechanical computer built to break German naval codes by way of a crib (a fragment of plain text). The Bombe would compare a crib to a potential setting of the encryption devices used to encode a message and a logical deduction would determine whether the setting contradicted the crib or not. If it did, I would move to the next potential setting. If it did not, additional cribs would be compared to the encryption setting [Hodges 2012]. This

technique paved the way for similar machines for both war-time and peace-time functions. The term "machine learning" was first coined in 1959 by Arthur Samuel, an IBM developer in the field of artificial intelligence and its applications in learning programs, particularly games [Samuel 1959]. Samuel demonstrated that a computer game of checkers could, with only 8-10 hours of play time, learn to perform better than its writer with only a basic set of instructions and parameters and a "sense of direction." This involved the program developing a decision tree of many possible moves through a "look-ahead procedure" at each iteration of play and choosing the branch of the decision tree with the greatest likelihood of success given the possible interactions with the player.

Work into ML developed through the 1950s and 60s into what more broadly come to be called artificial intelligence (AI), cybernetic systems designed to mirror human thinking processes. In 1958 Rosenblatt (1958) developed the perceptron, a machine (and later a program) that used linear classification to determine if an input matched a specific class. This was the first use of single-layer supervised binary classification, and would later be expanded into multilayer perceptrons, the forerunners of neural networks [Brunton et al. 2020]. Cybernetics as a field began to stagnate by the 1970s, however. Probabilistic reasoning became plagued by challenges stemming from contemporary limitations on data processing and this resulted in a division in the field between AI and machine learning, the latter of which began to focus on logical data-based approaches relying on passive observation and analysis rather than environmental interaction as with the former [Russell and Norvig 2003]. Since then, however, advancements in computing power and data management have brought much of these two fields back into overlapping research and application space.

In 1986 Rumelhart et al. (1986) advanced the field of ML through the development of backpropagation, a technique that calculates the gradient of a loss function in fitting a neural

network, which greatly increased the efficiency and efficacy of algorithm training over classic methods. The development of more accessible digital computing has spurred a growing interest in ML research that continues to this day [Brunton et al. 2020].

### 2.2.3 Types of Machine Learning Algorithms and Example Applications

Machine learning algorithms are divided into three general categories based on the amount and nature of the information available to the learner: supervised, unsupervised, and semisupervised. Within each, different algorithms are employed depending on the type of data [Brunton et al. 2020, Mitchell 1997, Shalev-Shwartz and Ben-David 2014]. Table 2.1 provides a breakdown of the most employed algorithms in each category.

***Supervised Learning***

Supervised machine learning is that in which the inputs and their corresponding outputs are known to the algorithm during training and typically involve defining a mathematical relationship between inputs and outputs given certain parameters [Shalev-Shwartz and Ben-David 2014]. Through an iterative process of optimizing a loss function, the most appropriate mathematical relationship is established. In cases in which the inputs and outputs are numerical and continuous, regression algorithms such as linear and generalized linear are employed. An algorithm that predicts the price of a house based on the square footage and number of bedrooms by learning from historical real estate records might employ a regression algorithm.

Table 2.1: Basic categories of ML algorithms commonly used based on the availability and type of information [Brunton et al. 2020].

| Supervised | Semisupervised | Unsupervised |
|---|---|---|
| Classification<br>• Support vector machines<br>• Neural networks<br>• k-nearest neighbor | Reinforcement learning<br>• Q-learning<br>• Markov decision processes<br>• Deep reinforcement learning | Dimensionality reduction<br>• Principle component analysis<br>• Self-organizing maps<br>• Diffusion maps |
| Regression<br>• Linear<br>• Generalized linear<br>• Gaussian process | Generative models<br>• Generative adversarial networks | Clustering<br>• k-means<br>• Hierarchical clustering |

When the inputs and outputs are limited to classes of information or sets of specific values, classification algorithms such as support vector machines (SVM) and artificial neural networks (ANN) are used. ANNs are classification algorithms in which associations between numerous factors are weighed and filtered by "hidden layers." An ANN in the process of learning adjusts the weight of factors within the hidden layers through an iterative process of optimizing a loss function. An example of an application is spam filters in which an ANN learns to associate certain words in an email (such as "limited time offer" or "pre-approved") with potential spam activity; however, a hidden layer association might give significantly less weight to emails from senders with previous engagement with the recipient. ANNs have found many applications in areas involving a large amount of data with complex structures.

Supervised machine learning has found many applications in engineering. Teo et al. (1991) introduced ANN as a means to analyze particle image velocimetry (PIV) photographs. Using particle size, shape, position, and gray scale information as factors, they demonstrated that an ANN

could match particles in one frame with particles in another frame, thus enabling a program to determine their velocity due to displacement. Elforjani and Shanbr (2018) developed a method for estimating the service life of bearings by analyzing historical acoustic emission through a combination of regression, SVM, and ANN machine learning techniques. Regan et al. (2017) applied logistic regression and SVM ML to the analysis of vibration of wind turbine blades for detection of damage in structural health monitoring.

***Unsupervised Learning***

Unsupervised learning is that in which only inputs are known to the learning algorithm. The data is unlabelled, meaning it has not been organized into groups or factors. The objective of this type of learning is commonly associated with pattern identification of large amounts of data through quantization or clustering, and it typically involves probabilistic methods. Common clustering algorithms are k-means and hierarchical clustering. In k-means, vector quantization attempts to divide observations into *k*-number of clusters by measuring each observation's distance from a cluster mean (or centroid). This method is common for image processing applications in which a continuous palette of shades is reduced to a finite number of quantized colors for a variety of image processing and analysis applications, such as astronomy [Chavolla et al. 2018] and geospatial data for tracking land use/resources [Kanberoglu and Frakes 2019]. Hierarchical clustering involves dividing data into groups based on a metric (the measure of the distance between observations) and a linkage criterion (the dissimilarity between clusters).

*Semi-Supervised Learning*

Semi-supervised learning involves learning under partial supervision. This typically involves cases in which a set amount of data is labelled, but the majority remains unlabelled, and often involves human interaction as part of the learning process. A significant field of semi-

supervised learning is reinforcement learning in which an agent takes incremental actions, assesses the reward for that action, and either continues forward or changes direction as a consequence of the reward. These techniques are a form of human learning emulation, modelling how a person learns through trial and error using a limited framework of knowledge, much how a toddler learns to balance upright and walk by experiencing movements that work and do not work. This method has been widely applied in game theory and robotics. Zhou et al. (2021) describe the training of a robot to balance and roll on a single ball through deep reinforcement learning. The robot (agent) learns to avoid specific actions that lead to instability such as tilt angles and accelerations. Xie et al. (2020) describe using reinforcement learning to teach a series of animated bi-pedal characters to successfully navigate a course of "stepping stones" in a rough-terrain animated world.

### 2.2.4 ML Applications in Petroleum Industry

The oil and gas industry has seen numerous applications of machine learning. ML algorithms have been developed to estimate the physical properties of crude oil. Creton et al. (2019) demonstrated that a supervised ML model employing quantitative property-property relationships could predict a live well's equivalent alkane carbon number (EACN) through its chemical properties, such as API gravity, $C_{20-}$ fraction, and its chemical fraction properties. Rostami et al. (2013) employed a Gaussian regression supervised ML model to predict the density of crude oil at well heads from historical properties, and Sinha et al. (2020) expanded this to predicting the viscosity of all oil types through a support vector machine learning process. In a very recent analysis Hadavimoghaddam et al. (2021) compared various machine learning methods used to estimate crude oil viscosities at reservoir conditions and determined their performance as capable of simulating the true empirical results with accuracy.

Anifowose et al. (2016) and Otchere et al. (2020) both demonstrated that new oil well characteristics could be successfully predicted by mapping the known well data with their seismic characteristics through a mixed-model supervised ML incorporating support and relevance vector machines. They found that this approach was more predictive than an artificial neural network (ANN). Wood et al. (2019) employed a transparent open box learning network supervised algorithm to predict the oil formation volume factor at bubble point ($B_{ob}$) by using reservoir temperature, gas-oil ratio, and oil gravity and gas specific gravity as predictors from historical well data. Alipour et al. (2020) demonstrated that machine learning could be used to effectively process and analyze imagery of nanoindentations for the accurate characterizing of fracture toughness in samples of shale formations. An ANN using regression analysis was used by Obanijesu and Omidiora (2008) to predict rates of paraffin wax buildup in pipelines in 11 Nigerian oilfields based on a well's viscosity, temperature, and pressure.

There has been literature published on machine learning applied to crude oil pipelines that are within the context of fault detection. Rashid et al. (2015) demonstrated a distributed wireless sensor network in a dry environment pipeline system with support vector ML and Gaussian statistical modeling performed on negative pressure waves to identify adverse events regarding leak detection. The authors did not discuss, however, the potential of such a system being used in a subsea application. Leak detection through ML-integrated acoustic sensing has also been proposed by Ahn et al. (2019) in a dry environment. Mandal et al. (2012) described a method of enhancing hardware-based leak detection (chemical sensor, acoustic sensor, etc) by reducing false-positive indications through a support vector ML and artificial bee colony (ABC) approach. Kerf et al. (2020) proposed a method of crude oil leak detection by ML-assisted infrared image processing.

## 2.3 Research and Literature Gap

This literature review demonstrates that the application of machine learning to the petroleum industry is relatively new and to date has focused primarily on the prediction of crude oil and well conditions and properties from historical information. Very little has been researched on the topic of machine learning in the context of process monitoring in crude oil systems in general, and even less focus on subsea systems. Additionally, most ML applications in the literature related to pipeline process monitoring are dedicated to the detection of leaks; no publications related to other fault types, such as flow restrictions, could be identified in the present literature. This presents a gap in the literature and a significant research opportunity. It has been demonstrated that ML can be successfully employed in the topic of crude oil and its operations, and this thesis seeks to fill some of the gaps in the current literature.

This thesis proposes an expansion of the work performed by Fu et al. (2020) and L. Yang et al. (2019) by way of the application of a machine learning (ML) algorithm that uses live flow velocity, pressure, and temperature data to generate a model of normal operation, and once learned, be able to identify leaks and blockages in a pipeline by way of pattern non-conformance recognition. The research of additional authors support the use of flow data monitoring in the successful detection of fault conditions in pipelines [Adegboye et al. 2019, Ahn et al. 2019, Prihtiadi et al. 2016, Rashid et al. 2015]; however, they do not provide ML as a method of real-time data analysis and system intelligence. Automated systems or processes that can actively monitor flow parameters and, given historical data and fundamental principles, determine atypical or “out of control” conditions before or at the point of failure, rather than after the fact, would greatly enhance efforts in preventing losses and damage to health, safety, and environment.

# CHAPTER 3: PRINCIPLES OF FLUID MECHANICS AND MACHINE LEARNING

Chapter 3 is divided into three primary sections: Section 3.1 provides an overview of the properties of crude oil under investigation in this thesis. Section 3.2 provides an overview of the governing equations in fluid mechanics and the theory of fluid flow through a pipeline under normal conditions. This will include the continuity, momentum, and energy equations, which describe the fundamental behavior of fluid mechanics in a pipeline. Assumptions will be employed to simplify these equations and the basic properties of crude oil will be considered. Governing equations are expressed in cylindrical coordinates unless otherwise noted. Section 3.3 provides an overview of machine learning algorithms and their risk and loss functions. A more detailed description of the loss functions employed in supervised regression ML is provided and the method of model selection is described.

## 3.1 Crude Oil Characteristics and Assumptions

Crude oil is a heterogenous mixture of hydrocarbon compounds that are highly variable in their properties and compositions between sources, and even within the same source, varying over time with dependencies on wellhead pressure, temperature, consistency, and depth. They are usually considered Newtonian fluids with viscosities that are heavily dependent on temperature [Hansen et al. 2019, Lyons 2010, Palmer and King 2008]. This has a significant impact on thermodynamics and flow parameters, especially over significant pipeline distances in which temperature changes may be dramatic. Crude oil is frequently found in multiphase as a mixture of several states of matter, or a mixture of compounds of the same state: solid (e.g., sand, wax), liquid (e.g., hydrocarbon, water), and gas (e.g., methane, carbon dioxide). However, oils with a

negligible to relatively low gas/oil ratio (GOR), or those found at the wellhead to be undersaturated, are considered to be single-phase fluids in the analysis [Lyons 2010, Hansen et al. 2019]. An oil is undersaturated when the gas, if present, is fully dissolved and the pressure is sufficiently high to permit more gas to be dissolved if present. As such oil conveyed in subsea flowlines under high pressure typically exists in single-phase, with phase separation beginning in the riser [Lyons 2010]. To simplify the computation fluid dynamics (CFD) model the current study will assume that the fluid under study is single-phase and Newtonian. This is consistent with much of the literature to date involving CFD simulation of crude oil through pipelines where the multi-phase flow is not under investigation [Rukthong et al. 2015].

The model crude oil that will be used in this thesis is Brent crude oil, a light to medium gravity oil most associated with the North Seas subsea oil fields. This was chosen as the model fluid due to the ready availability of data in the literature. Table 3.1 reports the properties of Brent crude used in the CFD simulation and held constant throughout [Rhodes 1995]. Figure 3.1 shows the relationships between temperature and viscosity [ITOPF 2011].

Table 3.1: Physical and thermal properties of Brent crude oil.

| **Property** | **Value** | **Unit** |
|---|---|---|
| Density | 825 | kg/m^3 |
| Thermal capacity | 2500 | J/(kg-K) |
| Thermal conductivity | 0.130 | W/(m-K) |

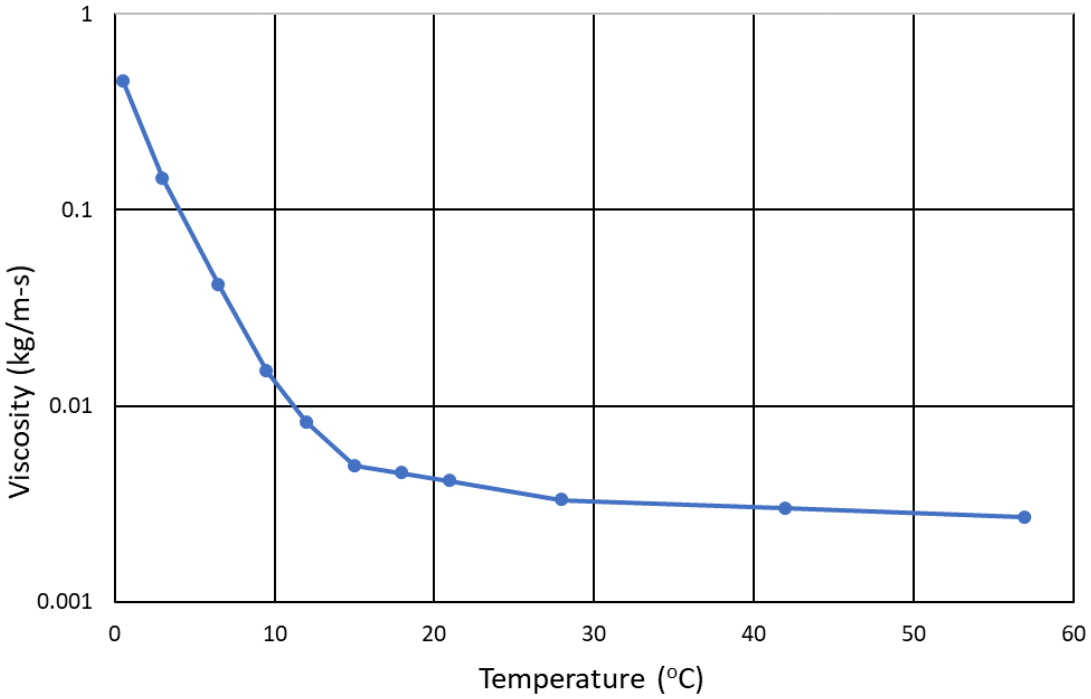



Fig. 3.1: Logarithmic dependence of viscosity of Brent crude oil on temperature.

## 3.2 Fluid Mechanics

This section discusses the continuity, momentum, and energy equations as governing equations used in the study of fluid mechanics. These equations account for the change in mass, momentum, and energy experienced by the fluid through time and across geometries. Under the understanding that the total mass, momentum, and energy in a system must be conserved, accounting mathematically for these three properties permits the researcher to predict the behavior of a fluid under a wide variety of conditions. These equations may be simplified under specific conditions when factors are known to be negligible, and this section addressed these simplifications under the conditions and assumptions of this thesis.

Throughout the present study a pipe will be considered as a horizontal cylinder of uniform radius (Fig. 3.2). Two points of interest representing locations of flow meters and thermocouples, 1 and 2, will be separated by length *l*, at which their respective pressures, mean velocities, and temperatures are represented by *p*, *U*, and *T*. The change in pressure between points 1 and 2, $\Delta p$, is $p_1$-$p_2$, and the change in mean velocity, $\Delta U$, is $U_1$-$U_2$. Pipe characteristics, such as surface roughness, will be held uniform.

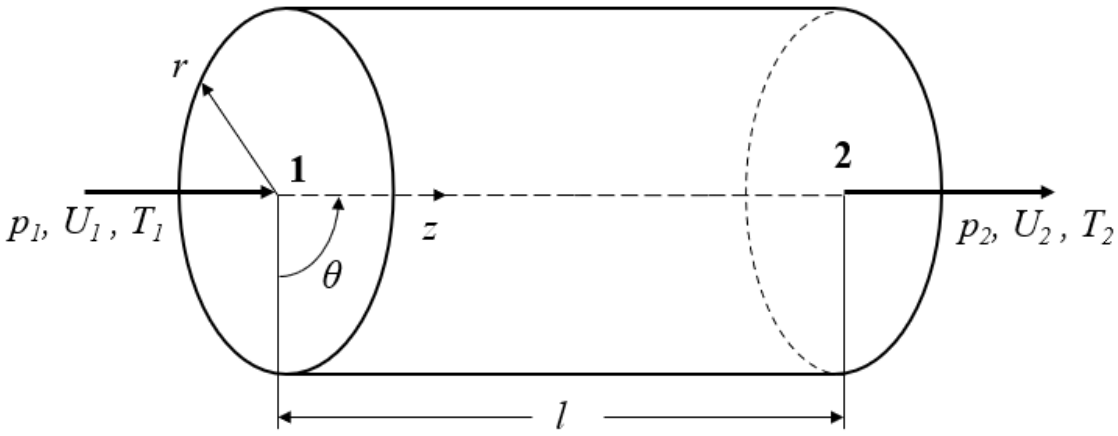



Fig. 3.2: Schematic of a horizontal pipe with a uniform cross section.

### 3.2.1 Continuity Equation

The continuity of mass of a moving fluid is expressed in cylindrical coordinates by the continuity equation [Yuan 1967]:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho v_r)}{\partial r} + \frac{1}{r}\frac{\partial (\rho v_\theta)}{\partial \theta} + \frac{\partial (\rho v_z)}{\partial z} + \frac{\rho v_r}{r} = 0 \qquad (3.1)$$

Because the density of the fluid is constant under the assumption that crude oil is an incompressible fluid the transient term disappears. If the pipe is assumed to be uniform in geometry and flow is fully developed, the radial and circumferential velocity terms are neglected, leaving only velocity in the z-direction. Eq. 3.1 simplifies significantly to:

$$\frac{\partial v_z}{\partial z} = 0 \tag{3.2}$$

A simplified equation for conservation of mass can be written as [Munson et al. 2012]:

$$\frac{\partial}{\partial t}\int \rho d\forall|_{CV} = 0 \tag{3.3}$$

which is integrated across the control volume to yield:

$$\sum|AU|_{in} + \sum|AU|_{out} = 0 \tag{3.4}$$

where *U* is the average velocity across a control surface.

### 3.2.2 Momentum Equation

Flow of a viscous incompressible fluid with a constant density ($\rho$) through a straight horizontal pipe is given in cylindrical coordinates by the continuity of momentum equation [Munson et al. 2012]:

$$\rho\left(\frac{\partial v_z}{\partial t} + v_r\frac{\partial v_z}{\partial r} + \frac{v_\theta}{r}\frac{\partial v_z}{\partial \theta} + v_z\frac{\partial v_z}{\partial z}\right) = -\frac{\partial p}{\partial z} + \mu(T)\left(\frac{\partial^2 v_z}{\partial r^2} + \frac{1}{r}\frac{\partial v_z}{\partial r} + \frac{1}{r^2}\frac{\partial^2 v_z}{\partial \theta^2} + \frac{\partial^2 v_z}{\partial z^2}\right) + \rho g_z \tag{3.5}$$

in the z-direction where viscosity ($\mu$) is a function of only temperature (*T*). If it is assumed that body force in a horizontal pipeline $\rho g_z$ is negligible, that flowrates change sufficiently slow as to be steady-state and fully developed at any given moment, and that radial angle and radius terms are neglected, Eq. 3.5 simplifies to:

$$\frac{dp}{dz} = \mu(T)\left(\frac{d^2 v_z}{dr^2} + \frac{1}{r}\frac{dv_z}{dr}\right) \tag{3.6}$$

### 3.2.3 Energy Equation

Because this study intends to model a temperature dependence on viscosity in the CFD, the energy equation is employed to model the temperature distribution [Yuan 1967]:

$$k\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 T}{\partial \theta^2} + \frac{\partial^2 T}{\partial z^2} + \Phi_i = \rho C_V\frac{\partial T}{\partial t} + \rho C_V\left(v_r\frac{\partial T}{\partial r} + \frac{v_\theta}{r}\frac{\partial T}{\partial \theta} + v_z\frac{\partial T}{\partial z}\right)$$

where

$$\Phi_i = 2\mu(T)\left[\left(\frac{\partial v_r}{\partial r}\right)^2 + \left(\frac{1}{r}\frac{\partial v_\theta}{\partial \theta} + \frac{v_r}{r}\right)^2 + \left(\frac{\partial v_z}{\partial z}\right)^2 + \frac{1}{2}\left(\frac{\partial v_\theta}{r} - \frac{v_\theta}{r} + \frac{1}{r}\frac{\partial v_r}{\partial \theta}\right)^2 + \frac{1}{2}\left(\frac{1}{r}\frac{\partial v_z}{\partial \theta} + \frac{\partial v_\theta}{\partial z}\right)^2 + \frac{1}{2}\left(\frac{\partial v_r}{\partial z} + \frac{\partial v_z}{\partial r}\right)^2\right] \quad (3.7)$$

When simplified by the assumptions made in the case of the momentum equation (Eq. 3.5), Eq. 3.7 is simplified to:

$$k\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + \frac{\partial^2 T}{\partial z^2} + \mu(T)\left(\frac{\partial v_z}{dr}\right)^2 = \rho C_V\left(v_z\frac{\partial T}{\partial z}\right) \quad (3.8)$$

where $k$ is the thermal conductivity and $C_v$ is the specific heat capacity of the fluid.

### 3.2.4 Fluid Flow Model in a Uniform Pipeline

Appropriate model selection will be critical to training the ML algorithm, as will be discussed in the next section of this chapter. It is desirable to train the ML algorithm with flow parameters that are easily obtained from commonly used industry-standard instruments, such as flowmeters and thermocouples. As described in Chapter 2 the parameters of the study will be flow velocity, temperature, and pressure. The momentum equation (Eq. 3.6) will therefore be the basis for the ML regression pressure model. To obtain a generalized model, scaling analysis is applied to Eq. 3.6 to identify the asymptotic behavior of the momentum equation when velocity in the z-direction approaches zero:

$$At\ v_z \to 0:\ \Delta p \sim 0 \quad (3.9)$$

and when velocity in the z-direction approaches infinity:

$$At\ v_z \to\ \infty: \Delta p \sim \frac{z}{r^2}\mu(T)(v_z^2 + v_z) \quad (3.10)$$

Equation 3.9 indicates that at a near-zero velocity the change in pressure approaches zero. Equation 3.10 indicates that as velocity approaches infinity the pressure drop across a span of

uniform pipe resembles a second-order parabolic relationship to velocity scaled to the temperature-dependent viscosity and the geometry. From published viscosity-temperature values of Brent crude in Fig. 3.1 the viscosity can be approximated as a power function of temperature in the following form:

$$\mu(T) = iT^{-j} \tag{3.11}$$

where *i* and *j* are constants. Inserting Eq. 3.11 into Eq. 3.10, the expected pressure-velocity-temperature relationship would be of the form:

$$\Delta p \sim \frac{z}{r^2} iT^{-j}(v_z^2 + v_z) \tag{3.12}$$

To reduce as many a priori assumptions for the ML algorithm as possible, geometry is incorporated into the coefficient and the proportionality of Eq. 3.12 can be expressed as:

$$\Delta p = aT^b U^2 + cT^d U \tag{3.13}$$

where a, b, c, and d are constants for coefficients and powers, and *U* as the mean velocity of the cross-sectional area.

## 3.3 Supervised Regression Machine Learning

From Section 2.2.3 the most appropriate ML algorithm for the application in this thesis is supervised regression. Given that the goal of this study is to use the mathematical relationship between flow parameters of temperature, velocity, and pressure change between two points in a uniform pipeline, the inputs and the outputs will all be known to the algorithms. These flow parameters are continuous numerical data and thus regression fitting will be the methods employed to define the relationships between inputs and outputs.

### 3.3.1 Training the Machine Learning Algorithm

Before the ML algorithm attempts to train with the data, the initial dataset is randomly divided into training set $S$ and test set $S_T$, typically in an 80:20 or 70:30 ratio. The test set is withheld from the training process to serve as a set of known data by which to evaluate the model [Cherkassky and Mulier 2007]. To train a ML algorithm, the program receives a training set, $S$, as input that has been taken from an unknown distribution $D$. A function $f$ is developed to generate a predictor $h_S$ from the training set. The primary focus of the ML algorithm is to define function $f$ (the model) to minimize the training error between the predictor $h_S$ and the known output $y$ with respect to $D$ and $f$ [Shalev-Shwartz and Ben-David 2014]. $D$ may follow a normal, binomial, Weibull, or other distribution depending on the nature of the dataset.

In general, the process of machine learning is one of minimizing a risk function given by:

$$R(w) = \int L[y, \varphi(x, y, w)]P(x, y)dxdy \tag{3.14}$$

where the inputs $x$ and outputs $y$ defined by a structure given as $\varphi$ with parameters $w$. The loss function $L$ depends on the form of $\varphi$ selected for fitting. The selection of the loss function will be discussed in the following sections. The inputs and outputs are sampled from a distribution $D$ and the probability $P$ that any given $x$ and $y$ are not randomly correlated is assessed [Brunton et al. 2020, Cherkassky and Mulier 2007].

### 3.3.2 Regression Loss Functions

A linear regression model can be given by the equation:

$$h_{w,b}(x) = \langle w, x \rangle + b \tag{3.15}$$

in which $h_{w,b}$ represents the hypothetical output of the function given parameter $w$ and intercept $b$. Let a hypothetical output for a given input value be defined as $h_i(x_i)$. The loss function in Eq. 3.14 for a linear regression model can be determined by the squared-loss function:

$$L(h_i, \varphi(x, y, w)) = \|h_i(x_i) - y_i\|^2 \tag{3.16}$$

In the case of a nonlinear regression of polynomial predictors of degree *n*, the model takes the form:

$$h(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{3.17}$$

The loss function in the case of nonlinear regression is the least squares:

$$argmin\, L(h_w) = argmin_w \frac{1}{m} \sum_{i=1}^{m} (\langle w, x_i \rangle - y_i)^2 \tag{3.18}$$

To solve Eq. 3.18 the gradient descent of the objective function is compared to zero [Shalev-Shwartz and Ben-David 2014]:

$$\frac{2}{m} \sum_{i=1}^{m} (\langle w, x_i \rangle - y_i) x_i = 0 \tag{3.19}$$

### 3.3.3 Loss Function Optimization

The defining characteristic of a ML algorithm is the ability to improve performance through the experience of tasks based on a performance metric (see Chapter 2). In the case of regression learning the loss function is the performance metric and the tasks are incremental attempts to fit the training set *S* to a function *f* using iterative values of parameters *w*. The "success" of each attempt is measured by the loss function, which is cataloged, and the parameters *w* are adjusted for a new attempt yielding a new value for the loss function. Following a certain number of attempts (determined through a process of backpropagation), the parameters that achieved the minimal loss function value are assigned to the model. An illustrative example is given in Fig. 3.3 in which a simple linear function (Eq. 3.15) is fitted to a training set *S* *n*-number of times, yielding *n*-number of the parameter (*w*) and intercept (*b*) sets with their respective *L* values calculated through the squared-loss equation (Eq. 3.15). Following *n*-number of iterations, the minimal *L* value determines *w* and *b* assigned to *f*. This process is fundamentally the same regardless of the form of function *f*.

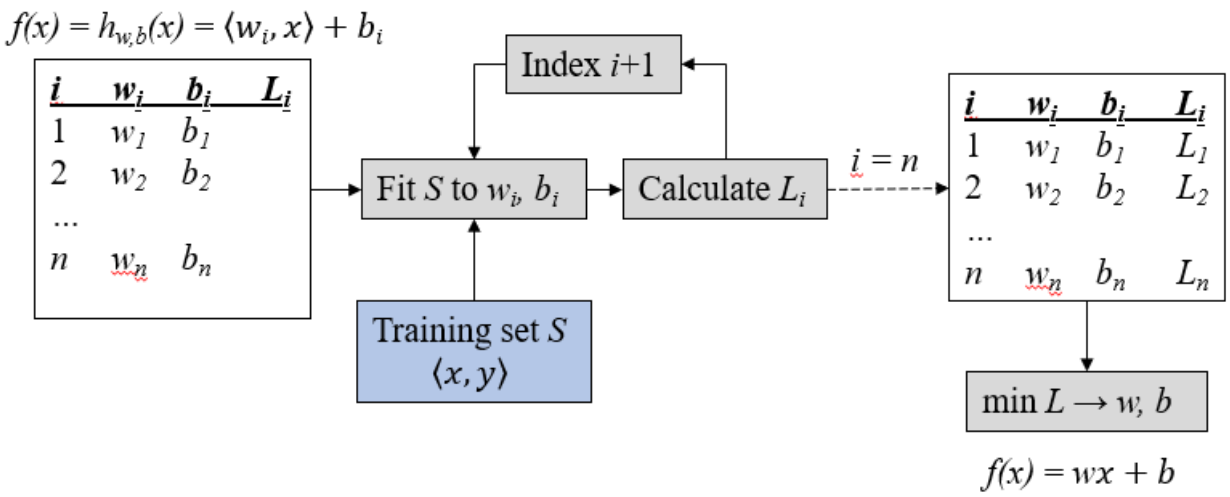



Fig. 3.3: Flow diagram of a generalized linear regression machine learning process.

The number of attempts $n$ in the iterative process may be determined by a "brute force" method of simply assigning a value of $n$ and dividing the parameters stepwise across a range of values and simply pulling those that yield the minimum $L$. This may be useful for estimation, but the risk of generalization error may be higher than necessary. A more efficient method employed in ML is gradient descent, in which the incremental change(s) in $w$ from each previous value is proportional to the gradient of the loss function:

$$w_{n+1} = w_n - \eta \nabla L(w_n) \tag{3.20}$$

where $\eta$ is a numerical value called the learning rate [Shalev-Shwartz and Ben-David 2014]. Figure 3.4 illustrates how a minimal loss function and optimal $w$ can be located, through an iterative process of indexing $w$ by decreasing amounts as $\nabla L$ decreases. Because the initial $\nabla L$ is negative and large in magnitude the first iterative advancement of $w$ is positive and large. In the example situation $\nabla L(w_2)$ is positive and smaller in magnitude than $\nabla L(w_1)$ and thus the next iterative step is negative and reduced. The process continues until $\nabla L \to 0$. Note that Fig. 3.4 is greatly simplified, and gradient descent may involve thousands of iterations.

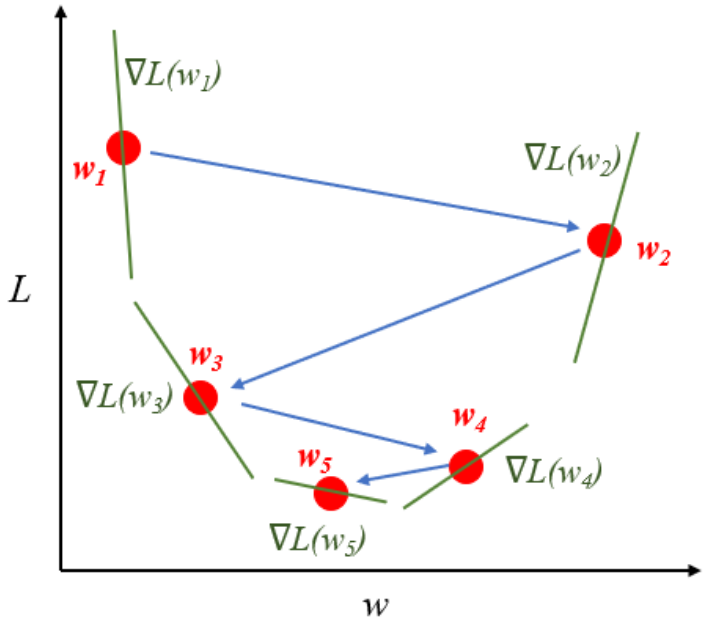



Fig. 3.4: Simplified diagram of the process of minimizing $L$ through gradient descent.

### 3.3.4 Model Selection and Validation

As noted previously the primary function of a ML algorithm is to define the appropriate function $f$ to serve as the model that minimizes training error. Success in properly fitting a function is measured by the generalization error, defined to be the probability that a model does not correctly predict an appropriate output given input(s) $x$ from within the distribution $D$. The error of a prediction $h{:}x \rightarrow y$ is given by:

$$L_{D,f}(h) \stackrel{\text{def}}{=} P_{x\sim D}(h(x) \neq f(x)) \tag{3.21}$$

in which the error is the probability of $h(x)$ not equaling $f(x)$ within the probability distribution $D$ for any random value of $x$ [Shalev-Shwartz and Ben-David 2014].

In many cases simple linear (Eq. 3.15) or polynomial (Eq. 3.17) functions suffice and manage to represent the data set $S$ with minimal error. In regression modeling there are

innumerable forms of functions and when the "true" relationship between inputs and outputs are unknown, appropriate selection of the function $f$ typically involves an iterative process of attempting various functions (such as linear, then polynomial of 1st degree, then 2nd degree, etc) and selecting the model that yields the least generalization error [Shalev-Shwartz and Ben-David 2014]. In some cases, however, it is appropriate to select a specific function for training in those cases in which the general relationship is already known. In this thesis, because the training set $S$ is expected to follow governing principles of fluid mechanics highlighted above, the pressure and velocity algorithms will be trained to the forms of Eqs. 3.4 and 3.13.

Following the selection of the model that best minimizes the loss function, it is evaluated using the test set $S_T$. This performs several functions: It serves as an evaluation of the model in the face of previously unseen data, and it serves to evaluate training bias, also referred to as "overfitting." This occurs when the model conforms too perfectly to the training set and is poorly able to identify real previously unseen data as belonging to the dataset because the ML algorithm was trained only to recognize the training data [Shalev-Shwartz and Ben-David 2014]. As an example of training bias related to this thesis, consider a situation in which a regression model is trained using a dataset that includes velocities ranging from 0 m/s to 0.7 m/s. However, the real operating conditions range from 0 m/s to 1.8 m/s. If the ML algorithm is only shown the data of narrow velocity range, the function $f$ fitted to that dataset will be biased to that range and predictions of flow characteristics (such as $\Delta p$) based on velocity conditions greater than 0.7 m/s will be made based on the forward extrapolation of $f$, not based on fitting to real data. This increases the risk of predictions using previously unseen data that is not within the ranges of the training data poorly fitting to function $f$. To limit the risks of training bias it is important that the training set be representative of the full range of parameters $w$ expected to be encountered. In

events that new data points are observed that are outside the range of the training data, it is a common practice to incorporate them into the training set and re-train the algorithm [Shalev-Shwartz and Ben-David 2014]. If the model is accurate, the predictors $h_T$, generated from the model $f$ from the inputs from the test set, $x_T$, should fall within the distribution of the real outputs $y$ to an acceptable level, typically evaluated as a percentage of test set points correctly predicted.

# CHAPTER 4: COMPUTATIONAL FLUID DYNAMICS MODELLING

Chapter 4 is divided into six primary sections: Section 4.1 explains the setup of the CFD models and solvers. Section 4.2 discusses the setup and results of the CFD simulation of a crude oil pipeline under normal operational conditions. Section 4.3 details the CFD modeling and results of the same pipeline in the event of a 10% and 20% leak under several environmental pressure scenarios. Section 4.4 discusses the CFD modeling and results of the pipeline with a 50% flow restriction. Section 4.5 discusses the CFD modeling and results of an increase in the viscosity profile of the crude oil. Section 4.6 provides a review of the modelling work in this chapter.

## 4.1 CFD Methodology

Flow parameter data for a horizontal, uniform crude oil pipeline under normal conditions were generated using ANSYS Fluent following general practices published by numerous CFD researchers employing energy equation analysis in the study of crude oil pipeline flow [Fu et al. 2020, Kumar et al. 2017, Rukthong et al. 2015, Yang et al. 2019].

### 4.1.1 Model Setup and Material Properties

A three-dimensional model of a straight uniform cross-section steel pipe of 200m in length and 0.300m inner diameter and 0.0125m wall thickness was generated to simulate a simple horizontal subsea pipeline of dimensions common in subsea applications [Palmer and King 2008]. The inner dimension was filled and designated as fluid. Meshing of the fluid elements and pipeline incorporated inflation at the wall interface, and fluid meshes were verified to have at least 200,000

elements before simulation. A no-slip boundary condition was set and a constant surrounding environmental temperature of 5°C was assumed throughout the analysis.

The fluid properties were programmed using Brent crude as the model fluid with the properties listed in Table 3.1. Temperature-dependent viscosity was programmed into Fluent as piecewise-linear using the published values for Brent crude (Fig. 3.1). The steel pipe material had a constant density of 8030 kg/m$^3$, a specific heat of 502.48 J/kgK, and thermal conductivity of 16.27 W/mK.

### 4.1.2 Simulation Solvers

To model the effect of temperature of the environment on the fluid, the energy equation (Eq. 3.7) was turned on and a single-phase, pressure-based solver was selected with an absolute velocity formulation. In this thesis, the turbulent flow field was assumed to be steady-state and the K-epsilon (K-ε) turbulence model was used as this is the most common in the prediction of turbulent energy dissipation under the assumptions used in this thesis [Kays et al. 2018, Fu et al. 2020]. This model employs two transport equations in simulating turbulent behavior: The kinetic energy equation (k) and the dissipation equation (ε).

***Turbulent Energy Equation, k***

The k equation describes the transfer of energy of fluid elements through turbulent effects and is given by [Kays et al. 2018]:

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G_k + G_b - \rho\varepsilon - Y_m + S_k \quad (4.1)$$

where $k$ is the turbulence kinetic energy (not to be confused with k as thermal conductivity used previously), $x_i$ and $x_j$ are corresponding locations with $u_i$ the velocity in the direction of $x_i$, $\mu_t$ is the eddy viscosity, $\sigma_k$ is a constant, $G_k$ is the turbulent flow energy from laminar velocity gradient, $G_b$

is the buoyancy flow energy, $Y_m$ is the fluctuation caused by turbulent diffusion, $\varepsilon$ is the rate of kinetic energy dissipation, and $S_k$ is the turbulent kinetic energy source.

***Dissipation Equation ε***

The ε dissipation equation provides the rate at which turbulent kinetic energy dissipates and is given by an equation very similar to Eq. 4.1 [Kays et al. 2018]:

$$\frac{\partial}{\partial t}(\rho\varepsilon) + \frac{\partial}{\partial x_i}(\rho\varepsilon u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + C_{1\varepsilon}\frac{\varepsilon}{k}(G_k + C_{3\varepsilon}G_b) - C_{2\varepsilon}\rho\frac{\varepsilon^2}{k} + S_\varepsilon \tag{4.2}$$

where $C_{1\varepsilon}$, $C_{2\varepsilon}$, and $C_{3\varepsilon}$ are dissipation parameters, $S_\varepsilon$ is dissipation rate source term, and $\sigma_\varepsilon$ is a constant.

In the Fluent solver, the model Prandtl number $Pr_t$ was set to 0.85, dissipation parameters $C_{1\varepsilon}$, $C_{2\varepsilon}$, and $C_{3\varepsilon}$ as 1.44, 1.92, and 0.9, respectively, $\sigma_k$ as 1.0, and $\sigma_\varepsilon$ as 1.3, as consistent with previously published works of crude oil modeling in Fluent [Kumar et al. 2017, Rukthong et al. 2015, Zhu et al. 2014].

### 4.1.3 Numerical Method

The SIMPLE algorithm was employed in this thesis as the solution method for solving the pressure-velocity coupling. Discretization of the governing equations (Eqs. 4.1 and 4.2) is accomplished through the finite volume method (FVM) with the spatial discretization of the gradient set to least squares cell-based, second-order pressure, second-order upwind momentum, first-order upwind turbulent kinetic energy and dissipation rate, and second-order upwind energy discretization. Convergence criterion of $10^{-6}$ of the residuals of the control volume was selected in all equations. These numerical methods are commonly selected in CFD for pipeline flow as they maximize accuracy while limiting the CPU usage [Fu et al. 2020, Zhu et al. 2014].

## 4.2 Normal Operational Pipeline Conditions

A range of inlet velocities were simulated for each of six inlet temperature groups: 11, 12, 13.5, 22, 37, and 52°C. This temperature range represents those commonly observed in subsea well-head conditions [Barker et al. 2014, Lyons 2010]. To select a range of velocities a set of safety criteria were used [Palmer and King 2008]. The first is a semi-empirical formula meant to minimize the effects of corrosion on steel pipeline walls:

$$U_{max} = \frac{122}{\sqrt{\rho}} \tag{4.3}$$

where $\rho$ is the density of the crude oil. The second is a rule of thumb used in production engineering as a trade-off between capital and operation expenditure:

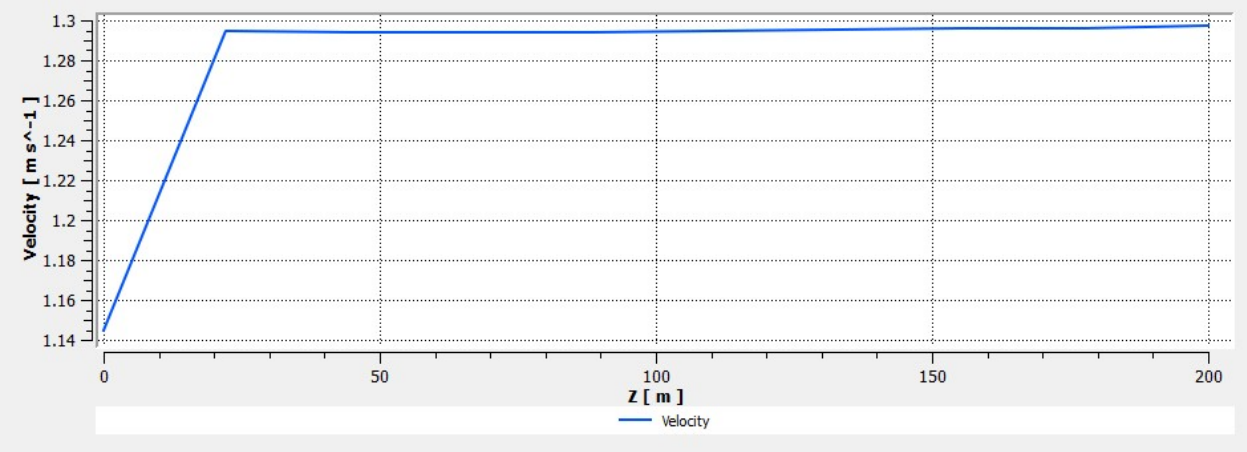
$$Q_{max} = \left(\frac{r}{0.420}\right)^2 \tag{4.4}$$

where $r$ is the inner radius of the pipe. The more conservative result from Eq. 4.3 and 4.4 was the latter, yielding a $U_{max}$ of 1.85 m/s. The flow was simulated under the above conditions and the resultant pressures, temperatures, and mean velocities were recorded at z = 50 m ("point 1") and the exit z = 200 m ("point 2") to represent the locations of flow meters and temperature probes in a pipeline that are spaced 150 m apart. The position of point 1 provides a 50 m entry length to ensure the CFD flow is fully developed before this reaches the point of the first sensors at point 1.

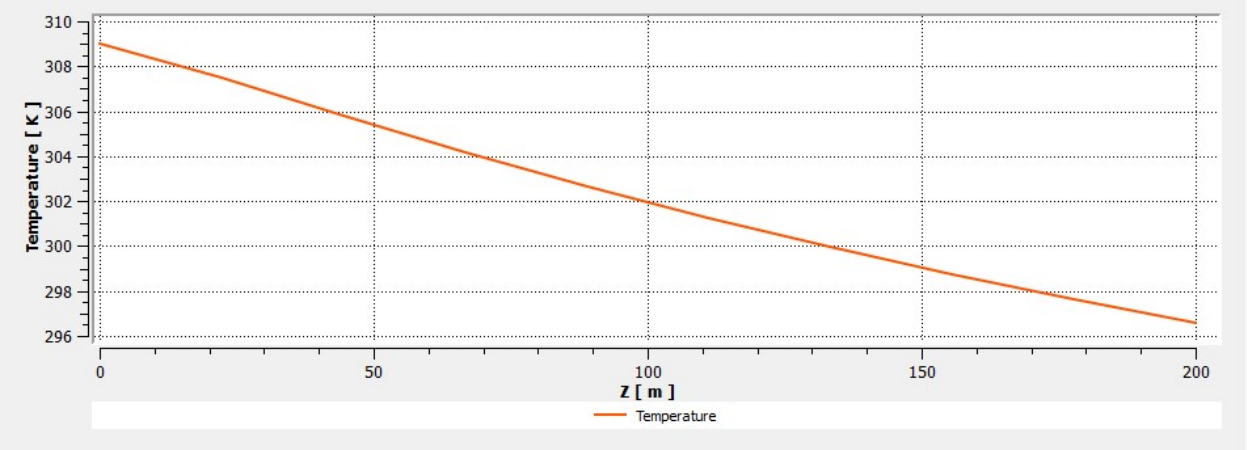
Figure 4.1 shows the velocity, temperature, and pressure along the center-line axis of the pipeline from the inlet (z=0 m) to the outlet (z=200 m) of an example observation under normal conditions with an inlet velocity of 1.145 m/s and temperature 36°C. The center-line velocity enters the pipe at the inlet velocity and increases linearly and reaches a constant magnitude of about 1.295 m/s at approximately 20 m. Note the velocity graphed in this figure is not the mean velocity at the z location but the instantaneous velocity at the center-line. This period of linear incline is consistent with a developing flow in which the cross-sectional profile is approximately

flat at entry, and viscous effects begin to elongate the velocity profile until it reaches a fully developed profile that remains constant for the rest of the length of the pipe. This center-line profile indicates that the flow is fully developed before reaching point 1 at 50 meters as intended by the design. The inlet temperature of 36$^{o}$C gradually decreases to approximately 24$^{o}$C as an effect of heat transfer to the environment. The loss in pressure follows a generally linear decrease from the inlet at about 6800 Pa to the outlet (gage pressure) with distance along the central z-axis (Fig. 4.1c). Equation 3.13 predicts that for a constant velocity $\Delta p$ is proportional to the distance of travel and these results support that prediction, as well as confirmed by the empirical observations of Prihtiadi et al. (2016) in Fig. 4.4, which validates these CFD results.

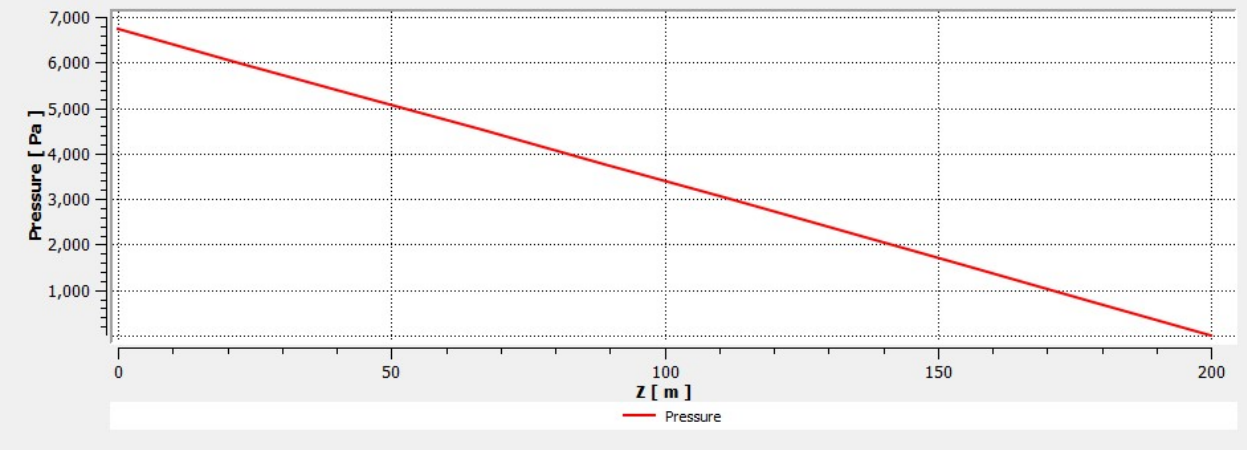
The mean velocities for each observation at point 1 and point 2 demonstrated very little change in velocity ($\Delta U \approx 0$) for all observations (Fig. 4.2), with greatest magnitude being about 1 mm/s and represents merely measurement error. This observation is consistent with conservation of mass per Eq. 3.4 and confirms the assumption that the mean velocity in the pipeline at point 1, $U_1$, may be used to represent the mean velocity at any fully developed location greater than z=50 m under normal operating conditions for all temperatures and velocities. Figure 4.3 demonstrates that temperature had a significant effect on $\Delta p$ as a function of $U_1$, particularly in higher velocities, and greater temperatures tend to lessen the loss in pressure along the pipeline. Eq. 3.13 suggests that $\Delta p$ is proportional to a second-order parabolic function of the z-velocity and the findings of this CFD analysis demonstrate this is true for $U_1$. Further, the impact of temperature-dependent viscosity is not negligible, and that decreasing temperature (increasing viscosity, Fig. 3.1) increases the pressure difference between points 1 and 2. These CFD results support the use of Eq. 3.13 as a model for pressure loss between points 1 and 2 as a function of both the temperature and velocity at point 1, and may serve as a model for training a ML regression algorithm.




(a)




(b)




(c)

Fig. 4.1: Velocity (a), temperature (b), and pressure (c) along the central axis of the model pipeline with inlet conditions of velocity 1.145 m/s and temperature 36ºC.

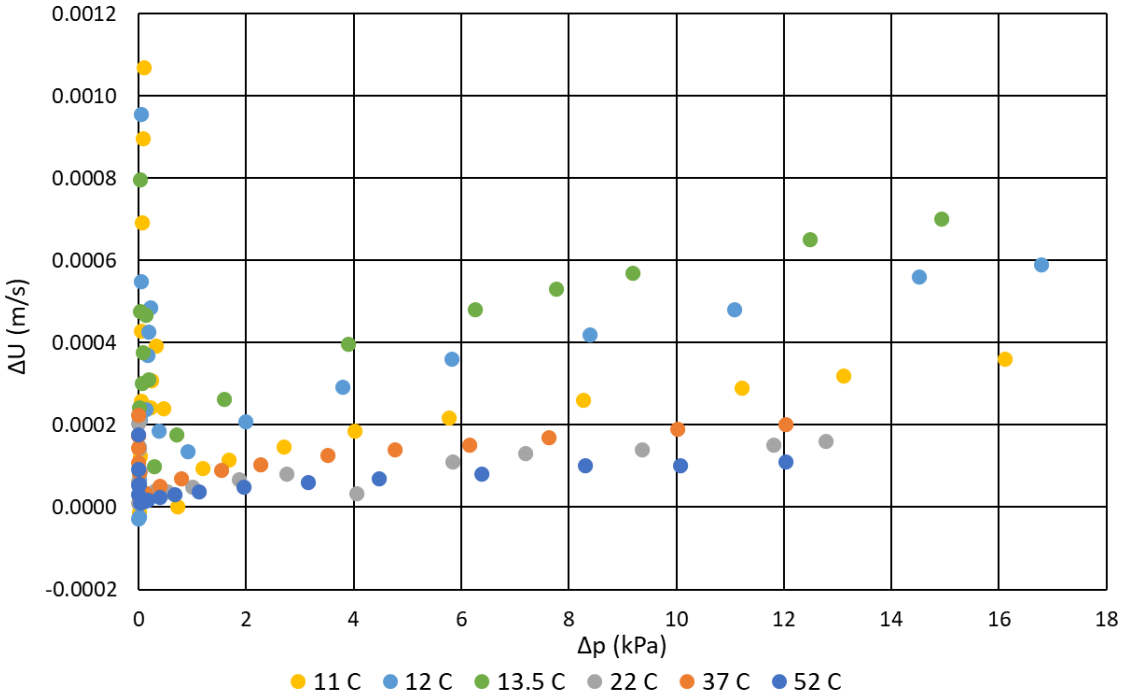



Fig. 4.2: Effect of temperature on $\Delta U$ as a function of $\Delta p$ between points 1 and 2 under normal pipeline conditions.

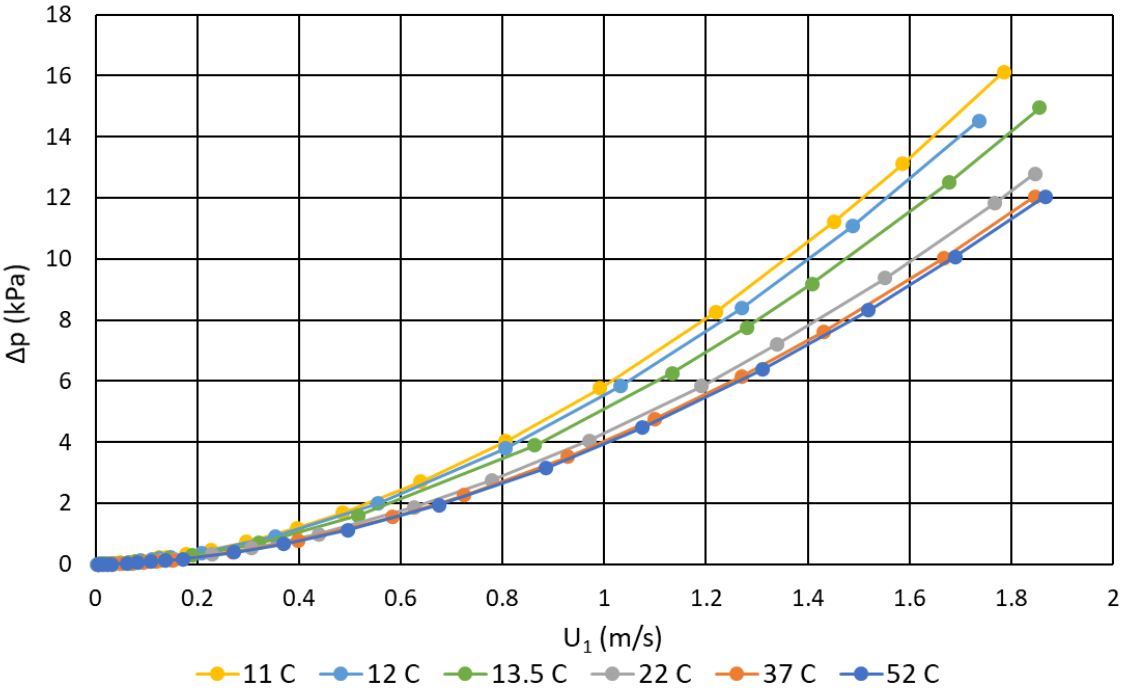



Fig. 4.3: Effect of $T_1$ and $U_1$ on $\Delta p$ under normal pipeline conditions.

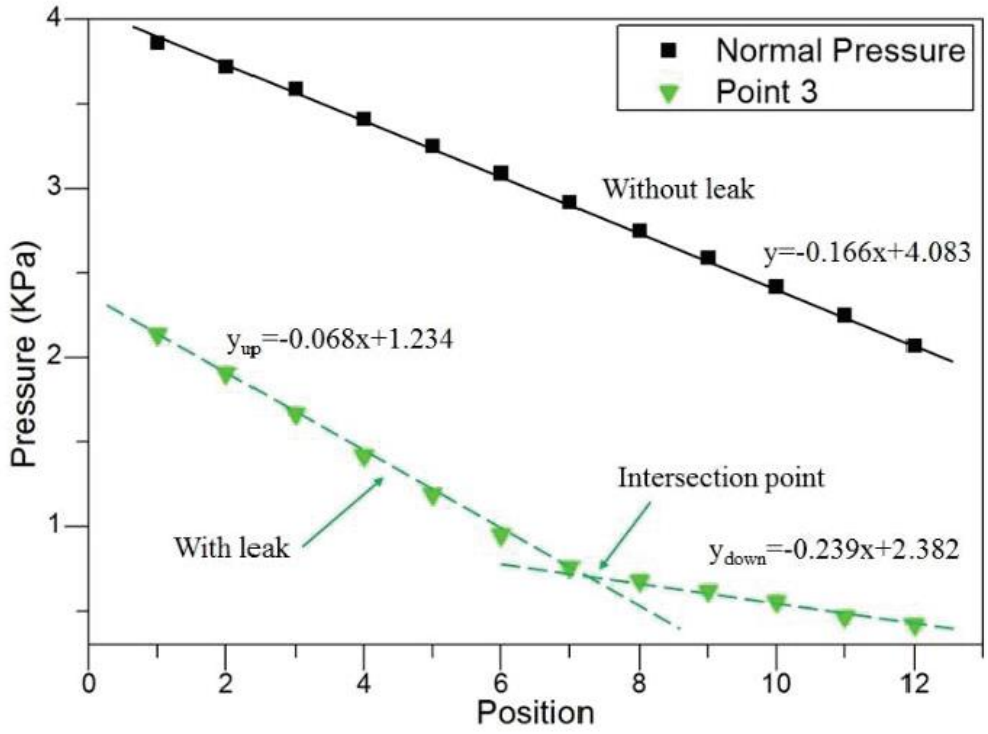



Fig. 4. 4: Prihtiadi et al. (2016) experimental results demonstrating the change in pressure profile of a pipe before and after a leak, and the intersection of two pressure gradients that indicate the location of a leak between positions 7 and 8.

## 4.3 Single Localized Leak

To simulate the condition of a leak an open outlet was introduced to the pipeline at z = 125m, equidistant from points 1 and 2, to limit transitional effects at either point of the simulated data collection (Fig. 4.5). The hole was modelled to have a diameter of 0.134 m to represent an opening of 20% of the inner cross-sectional area of the pipe. Ten observations of initial inlet velocities and temperatures were modelled across the operational range. Three environmental pressure conditions at the hole were simulated at these initial conditions to identify the effect of variable external pressure on the internal flow mechanics: -500 Pa, equal to (0 Pa), and +500 Pa

relative to the outlet pressure (point 2). To examine the effect of leak size on the pipe flow a fourth condition was modeled: a 10% leak (diameter 0.0949 m) under external pressure equal to the point 2 pressure (0 Pa). Fluid and pipeline properties and geometries, as well as simulation solvers and numerical methods, were otherwise unchanged in all leak conditions from simulating the normal conditions as in section 4.2.

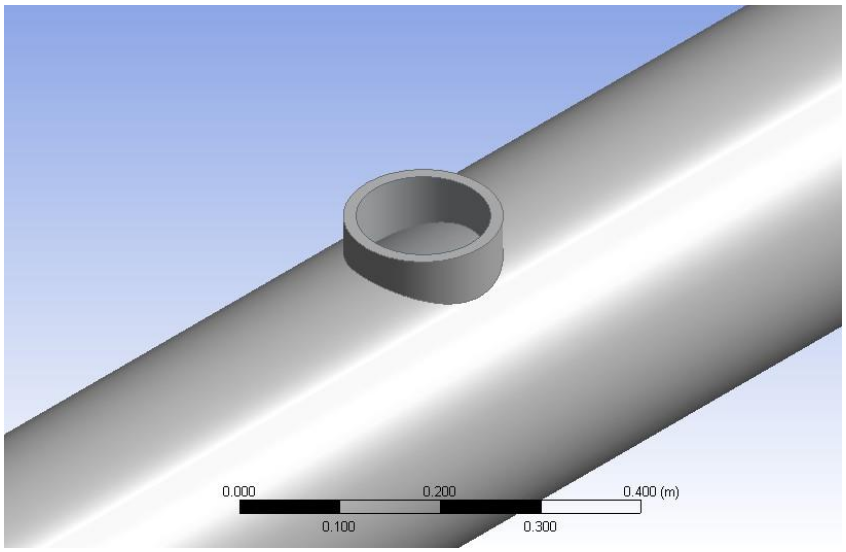


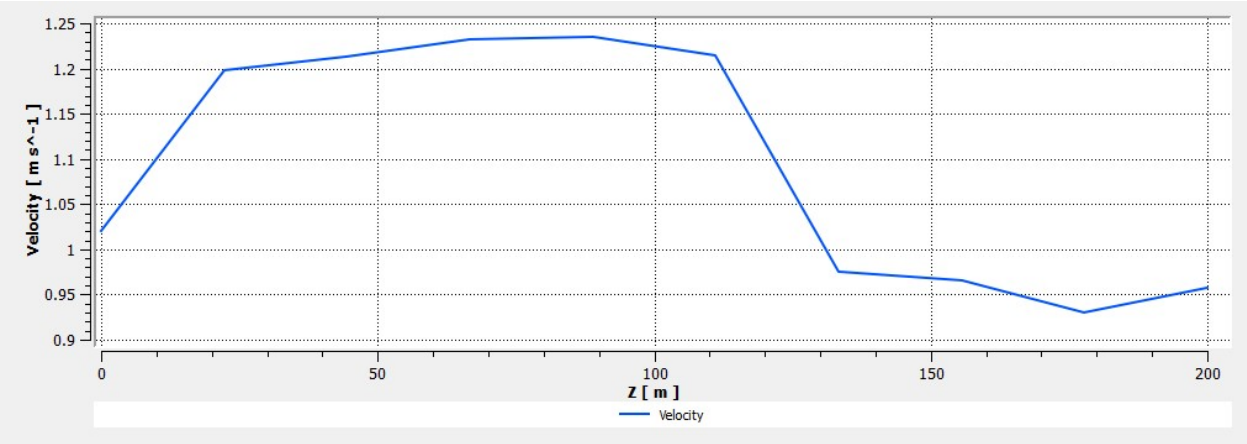

Fig. 4.5: An opening of 0.134m diameter (20% leak) at z = 125 m.

The center-line z-axial velocity and pressure profiles of an example observation under a 20% leak (0 Pa) are given in Fig. 4.6, which demonstrates a significant drop in velocity across the leak position (z = 125 m) and a discontinuity in the velocity gradient at the same point mirrors that observed by Prihtiadi et al. (2016) and by Fu et al. (2020) (Fig. 4.4) under conditions of a leak, which validates the results of this CFD simulation. The temperature gradient observed in this case is not shown, but the gradient resembled that of Fig. 4.1b in general behavior. Most notably in the event of leaks, the pipeline shows a significant divergence in $\Delta U$ behavior between points 1 and 2 compared to normal as a function of $\Delta p$ (Fig. 4.7). In all the leak conditions examined, the data demonstrates that velocity loss is significant. In the event of a leak, the continuity equation (Eq. 3.4) can be adjusted to account for the new outlet:

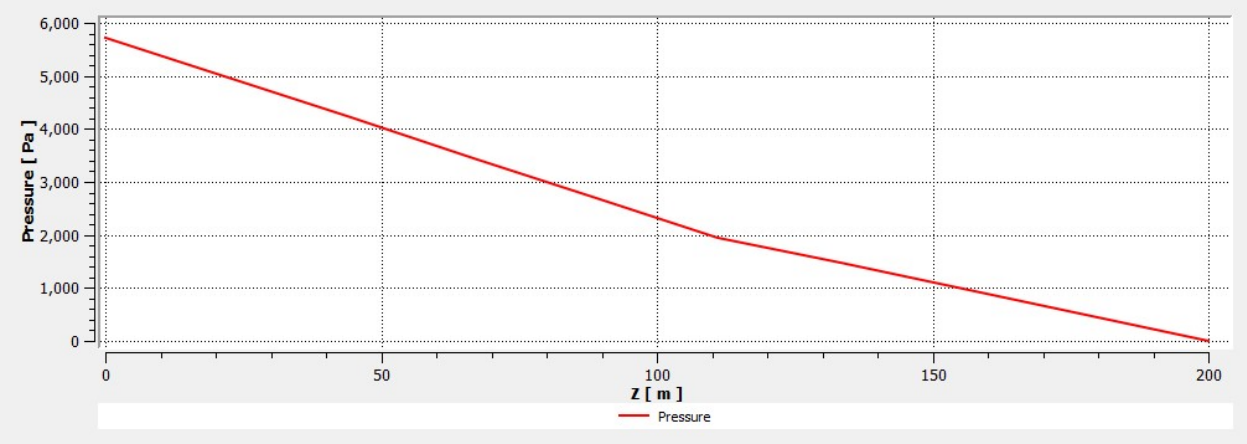
$$\sum|AU|_{in} + \sum|AU|_{out} = AU_1 - (AU_2 + A_l U_l) = 0 \quad (4.5)$$

$$A\Delta U = Q_l \quad (4.6)$$

where $A_l$ and $U_l$ are the control volume area and velocity of the leak location, respectively. In cases in which neither may be known directly, $A_l U_l = Q_l$. Eq. 4.6 shows that the flow rate through the leak can be calculated directly as the product of the inner nominal cross-sectional area of the pipe and the change in mean velocity between points 1 and 2.




(a)




(b)

Fig. 4.6: Velocity (a) and pressure (b) along the central axis of the model pipeline with inlet conditions of velocity 1.02 m/s and temperature 18°C and 0 Pa leak pressure.

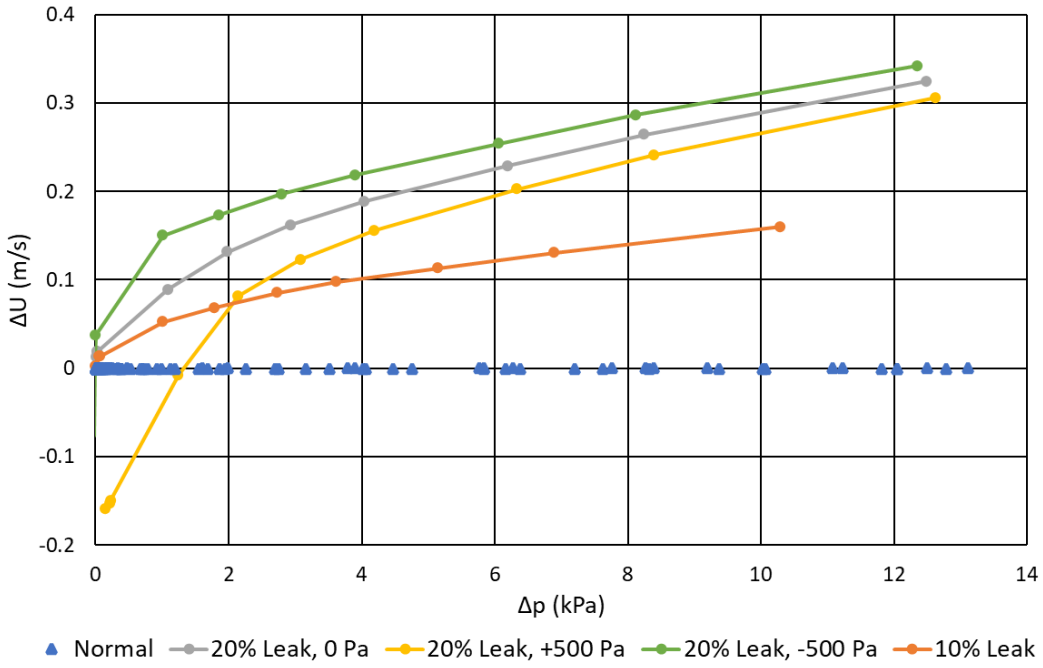



Fig. 4.7: Divergence of velocity behavior from the normal pipeline conditions in the event of several different leak conditions.

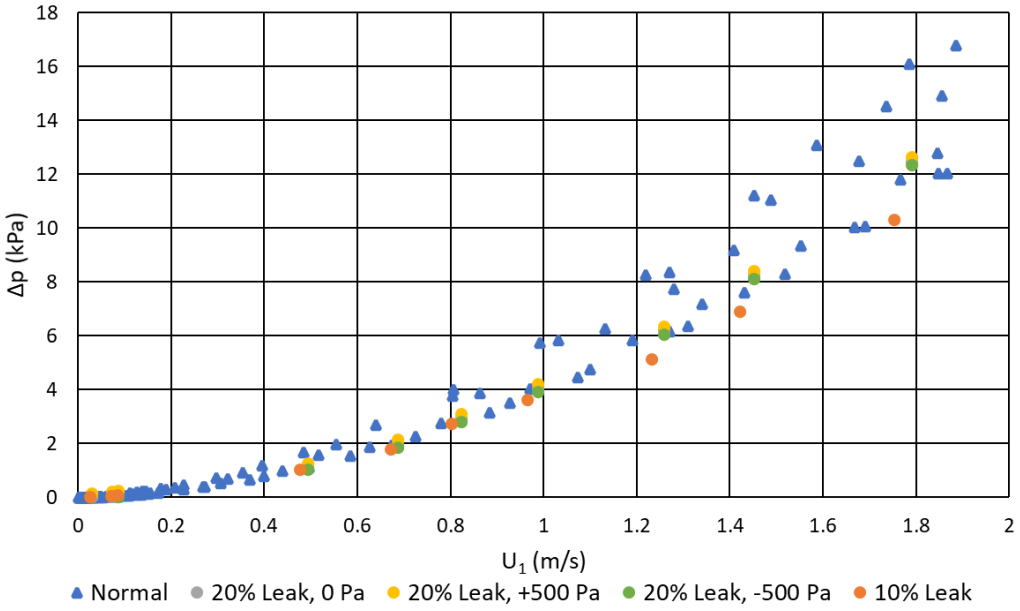



Fig. 4.8: $\Delta p$ as a function of $U_1$ in the event of a 20% leak under variable leak environmental pressures, compared to the normal pipeline conditions. Note that the effect of $T_1$ on $\Delta p$ is present, but not given its own axis.

Equation 4.5 shows that the magnitude of $\Delta U$ is proportional to the volumetric flow rate of the leak, and by extension, the area of the leak aperture. Figure 4.7 shows that the $\Delta U$ profile increased from 10% to 20% leak condition when considering the 0 Pa and -500 Pa environmental conditions. Eq. 4.5 also suggests that when the expected flow through the leak is positive (into the sea environment) and entrainment of sea water into the pipe (negative $Q_l$) would occur in events of negative $\Delta U$ ($U_1$-$U_2$). Fig. 4.7 demonstrates that this is dependent upon both the pressure gradient in the pipe and the external pressure and suggests that leaks of a more significant flow rate than those examined in this study would likely exhibit greater divergence from normal. This claim is further supported by Eq. 4.6.

$\Delta p$ as a function of $U_1$ did not change significantly in the event of all leak conditions, however, when compared to the normal condition (Fig. 4.8). Note that this graph of $\Delta p$ as a function of $U_1$ is two-dimensional and does not fully represent the effect of temperature on pressure change. This relative lack of significant change in the pressure loss under different leak conditions relative to normal is explained by the upstream loss in pressure potential. This concurs with the empirical observations published by Prihtiadi et al. (2016) (Fig. 4.4).

## 4.4 Single Localized Flow Restriction

Restrictions to flow may be experienced in a subsea pipeline through one or more mechanisms, including the deposition of wax or sand, a stagnation point of water separation, or crushing of the pipe due to external third-party damage. To simulate this, a semi-torus was modelled at z = 125 m of a pipe, to limit transitional effects on points 1 and 2, with an inner diameter of 0.212 m such that the minimum cross-sectional area is 50% that of the nominal inner pipe area (Fig. 4.9). Fluid and pipeline properties, geometries, and simulation conditions were

otherwise unchanged from those used under normal conditions as in section 4.2. Thirteen observations spanning the range of velocity and temperature conditions were simulated.

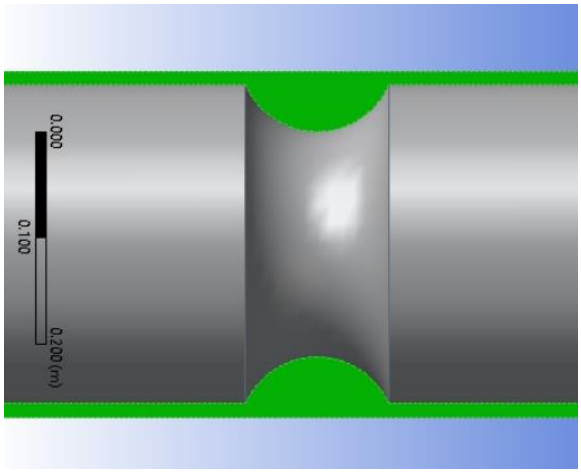



Fig. 4.9: Cross-sectional view of a 50% flow restriction. The wall of the pipe (with the restoration) is highlighted in green and the fluid pathway in grey.

The results of the flow restriction simulation demonstrate in a graph of $\Delta p$ as a function of $\Delta U$ that the increase in mean velocity between points 1 and 2 was significant in comparison to the behavior observed under normal conditions. Fig. 4.10 shows that with increasing pressure change the change in mean velocity decreases in the form of a negative power function. The magnitude of $\Delta p$ is extended in the positive direction with respect to $\Delta U$ relative to observations of normal behavior. Velocity is observed to increase from point 1 to 2 ($U_1$-$U_2 < 0$); however, these $\Delta U$ values are very small relative to $U_1$ (less than 2%). Several authors considering flows through restrictions [Sun and Wenquan 2012, L. Yang et al. 2019, Zhang 2017] have noted similar small variances in readings likely the result of de-stabilization or flow-reversal effects on the streamline flow in the CFD solutions. Compared to the effect of a leak on $\Delta U$ (Fig. 4.7), this effect may be considered negligible and likely within the standard error of the instrumentation.

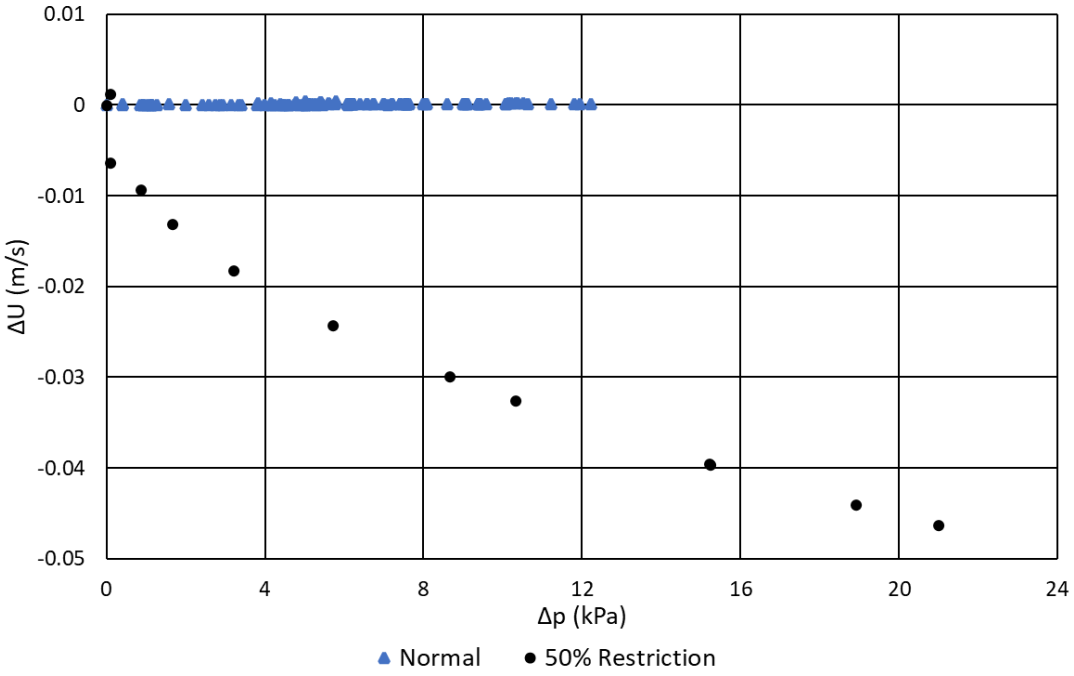



Fig. 4.10: Divergence of velocity behavior in the event of a 50% flow restriction from the normal pipeline conditions.

Fig. 4.11 shows that the upward shift in pressure difference across the flow restriction is also evident as a function of point 1 velocity; however, the magnitude of mean velocity is not showing an increasing or decreasing behavior. The interpretation of these two models (Figs. 4.10 and 4.11) suggests that the increase in velocity (relative to normal operational conditions) was primarily a downstream effect of the restriction, and that the excess pressure loss was permanent. Although this thesis did not examine various geometries of restriction or variable sizes, other authors examining the shape factors of venturi tubes [Sun and Wenquan 2012, Zhang 2017] and sudden channel diameter restrictions [Yang et al. 2019] in pipes have demonstrated that pressure loss is approximately proportional to the degree of restriction relative to the nominal pipe diameter. By this, it can be assumed that restrictions greater than 50% would yield a greater divergence in the pressure curve.

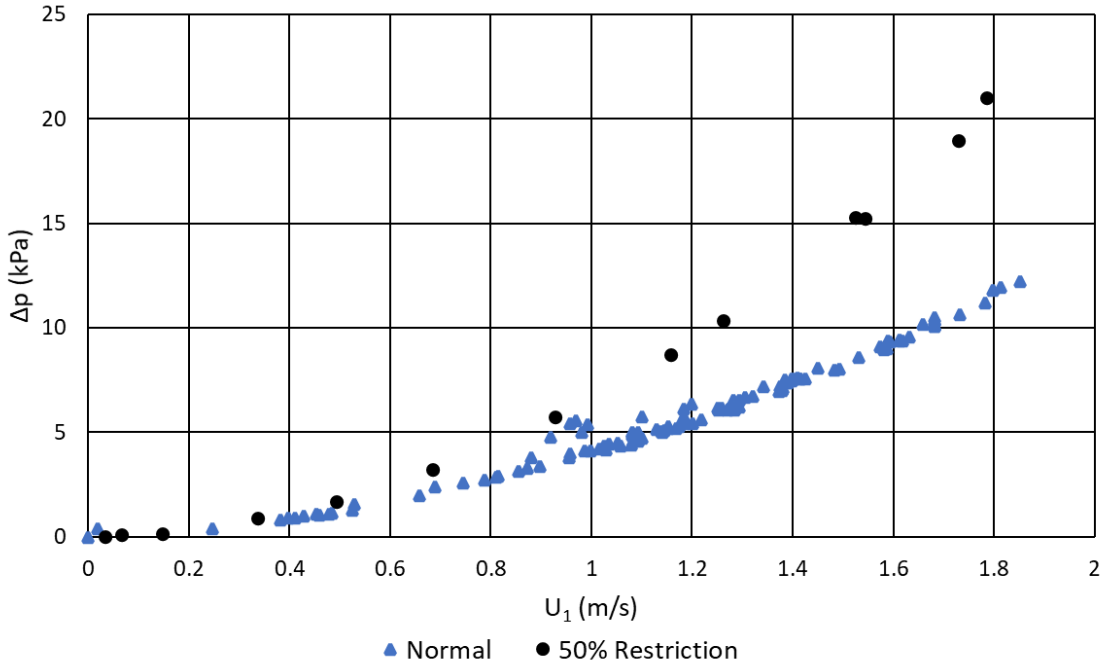



Fig. 4.11: Δ*p* as a function of mean velocity at point 1 in the event of a 50% flow restriction compared to the normal pipeline conditions.

## 4.5 Shift in Crude Oil Viscosity

To this point, the viscosity curve of the fluid has been considered constant throughout the analysis. However, when considering the momentum equation, the fluid mechanics are governed by the viscosity-temperature relationship, and a change in the viscosity of the crude oil would yield an expected proportional shift in the pressure change. As discussed in Chapter 3, the values of the viscosity of a particular well-head may not remain constant for significant periods and do shift over time. To simulate the effect of a change in the viscosity of the crude oil on the fluid mechanics the pipeline system in section 4.2 was simulated with the initial conditions used in section 4.3 with the only exception that it was simulated under two new temperature-dependent viscosity conditions: one in which the curve was shifted upward by a factor of five ("high viscosity") and a second shifted down from normal by a factor of 0.5 ("low viscosity"). These adjustments are

intentionally extreme compared to viscosities typically observed of Brent crude of specific APIs [Rhodes 1995].

The $\Delta p$-$\Delta U$ relationship did not diverge from that of normal operating conditions with any practical significance in terms of the velocity for either the high or low viscosity condition, as mass conservation would suggest. The $U_1$ - $\Delta p$ relationship does, however, demonstrate a slight upward shift in the parabolic velocity profile as compared to the point 1 velocity in the case of high viscosity, and a slight downward shift below the normal behavior in the case of low viscosity (Fig. 4.12). These resultant shifts in the $\Delta p$ behavior with varying viscosity would be expected as the momentum equation demonstrates. However, this proves challenging as a diagnostic tool. Thus far neither leaks nor flow restrictions have depressed the pressure curve significantly, as in the case of the low viscosity condition, and the observation of this behavior may be diagnostic of a decrease in the viscous effects in the pipe flow system. However, the shape of the curve of the high viscosity condition may be indistinguishable in a practical sense from that of a flow restriction given the data available (Fig. 4.11). This presents a limitation on this method: since the intent of this thesis is to use commonly employed instrumentation (flow meters, thermocouples) without additional diagnostic equipment, the risk of a false-positive flow restriction determination as a result of a significant upward shift in the viscosity will need to be taken into account in operational considerations when investigating.

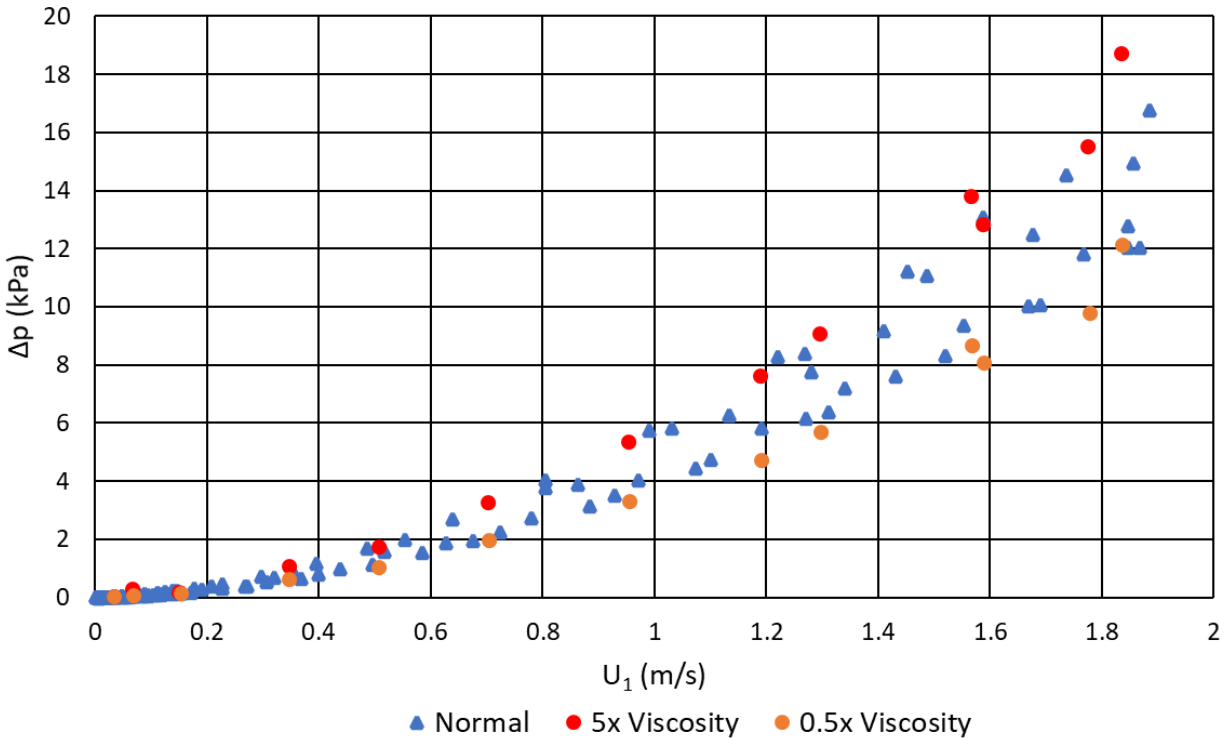



Fig. 4.12: $\Delta p$ as a function of $U_1$ under higher viscosity compared to normal.

## 4.6 Summary of CFD Findings

The behavior observed in the fluid flow parameters under conditions of normal operation (section 4.2), a leak in the center of the pipeline (section 4.3), a flow restriction in the center of the pipeline (section 4.4), and significant shifts in the temperature-dependent viscosity curve of the study fluid (section 4.5) demonstrate three primary aspects important to the goals of this study: 1) The behavior of crude oil under normal conditions behaves in a pattern consistent with the mechanics and governing equations discussed in Chapter 3; 2) Under the two fault conditions modelled (leak and flow restriction) there is a demonstrable and significant divergence from the behavior observed under normal conditions, suggesting that divergence from this predicted behavior can be diagnostic of adverse conditions; and 3) The $\Delta p$-$\Delta U$ and $U_1$-$\Delta p$ behavior

identified under these different fault conditions are distinct, suggesting the flow behavior can be diagnostic in the identification of the defect type. In the event of leaks the $\Delta p$-$\Delta U$ relationship diverges greatly from normal conditions. A leak under several conditions presented a relationship resembling a positive power curve (Fig. 4.7). In the case of a flow restriction the $\Delta p$-$\Delta U$ relationship did not change significantly from normal (Fig. 4.10), yet the $U_1$-$\Delta p$ experienced a significant shift upward (Fig. 4.11).

Chapters 5 and 6 will demonstrate how these behaviors can be exploited by a machine learning program by recognizing that, under the assumptions given in this chapter and Chapter 3, the behaviors should follow a mathematical model under normal conditions. When the system is observed to deviate significantly from that model, a new and adverse condition in the system can be identified. The data of the new condition can then be used to infer the most likely situation (leak, flow restriction, or fluid property shift) and report this situation to the user.

# CHAPTER 5: MACHINE LEARNING AND PROCESS MONITORING PROGRAM

Chapter 5 is divided into four primary sections: Section 5.1 details the coding used to train the machine learning algorithms. Section 5.2 details the program functions of assessing new data for conformance to the ML models. Section 5.3 details the method used by the program to infer the probable cause of a fault or non-conformance in new data. Section 5.4 summarizes this chapter's key findings.

## 5.1 Training the Machine Learning Algorithms

Pre-processing of the CFD data was performed prior to importation to the ML program. This involved converting temperatures from Kelvin to Celsius and pressure from Pa to kPa. This was done in order to bring the data sets into closer orders of magnitude to improve the fitting (for example, 1.5 kPa is within the magnitude of 1.2 m/s while 1500 Pa is not). The ML program generated for this thesis was written in Python on a development platform called Jupyter, and these include standard libraries and functions for ML programming that is very commonly used in academia and industry. The program coding in full can be found in Appendix A. To train the program on the pipelines flow characteristics under normal conditions, the 120 observations acquired in the ANSYS simulations were imported as a dataset in a CSV file. Rows of the dataset were then split randomly into two groups, one for training (dataset $S$) and one for testing (dataset $S_T$), in a 70:30 ratio by a train_test_split function:

```
#break the datasets into train/test sets
vel_train, vel_test, veldiff_train, veldiff_test, temp_train, temp_test, press_train, \
    press_test = train_test_split(vel1_set, veldiff_set, temp_set, press_set, test_size=0.3)
```

### 5.1.1 Pressure Model Training

The training set was fitted to a customized function called "pressureFunc" by the curve_fit function included in the SCIPY library. Curve_fit uses the least squares loss function (Eq. 3.19) and a gradient descent methodology to return the best-fitted parameters to a user-designated function. In this project, that function was pressureFunc, which uses Eq. 3.13, with the point 1 velocities ($U_1$) and temperatures ($T_1$) as inputs and the pressure difference between points 1 and 2 ($\Delta p$) as the output:

```
#multivariate fit function with x[0] as point 1 velocity and x[1] as point 1 temperature.
def pressureFunc(x, a, b, c, d):
   return a*(x[0]**2)*(x[1]**b) + c*(x[1]**d)*x[0]

#generate the model using curve_fit function and pressureFunc
p0 = [2.50, 0.100, 30, -1]   #provide initial parameters to fit function to guide least squares
param, fitCov = curve_fit(pressureFunc, temp_array_pr, press_train, p0)
```

The parameters that minimize the loss function in the curve_fit function are stored as param[ ] values in an array. Param[0], param[1], param[2], and param[3] values correspond to the coefficients of Eq. 3.13 a, b, c, and d, respectively.

The success of the fitting was assessed by calculating the coefficient of determination, or $R^2$ score, from the predicted outputs ($h_i$) for all observations calculated by their inputs and the real outputs ($y_i$). This coefficient reports the proportion of the variance in the output that is predicted from the inputs. If the $R^2$ score is less than 0.80 (that is to say, less than 80% of the variance observed in $\Delta p$ is the result of variance in $U_1$ and temperatures $T_1$) the model is considered to have an inadequate fit and this is reported to the user. The $R^2$ score is related significantly to the loss function in that minimizing the loss function should maximize the $R^2$ score for a given set of parameters. However, as the loss function is not by itself a measure of how well the model fits the data in an absolute sense, the $R^2$ score is used as an indicator of the "degree of fit." If an $R^2$ score

is reported less than 0.80, several issues may be the cause. 1) The data are not well correlated to the function, resulting in high generalization error. 2) Outliers in the data are skewing the curve fitting. Examination of the graphical output may be diagnostic in determining the cause of a poor fit, and either resampling or additional sampling would be at the discretion of the user. If outlying data is suspected statistical tools for identifying outliers such as random sample consensus (RANSAC) may be used.

To test the model's ability to predict unseen data the $\Delta p$ values of the test set are compared to predicted values from the test set inputs. Because the real values are not expected to fall exactly on the predicted, their conformance to the model is assessed by assuming the data follows a normal distribution about the model. Test set data that falls within a 95% confidence interval of the predicted are considered to belong to the distribution. This is achieved in the program by calculating the standard deviation of the training set by the sum of errors of each real $\Delta p$ to the hypothetical output given by the model. The percent of test set points that fall within the confidence intervals are calculated and reported as the predictive power of the model. In the event the predictive power is less than 85%, the user is alerted. The model values are reported to the user and the results of the ML fit to the pressure model are shown graphically (see Chapter 6 for examples).

### 5.1.2 Velocity Model Training

The relationship between the change in velocities ($\Delta U$) and the change in pressures ($\Delta p$) between points 1 and 2 are assumed to follow a linear relationship as per Eq. 3.4 in which $\Delta U$ does not vary with $\Delta p$ under normal conditions and is independent of temperature. As a note, iterations of this formula including temperature was tested in this thesis, but it was confirmed that temperature was not predictive of the $\Delta U$ behavior and was not included in the ML algorithm. A

linear regression model (Eq. 3.15) was trained from the training dataset using the linear_model function from the SKLEARN library:

```
#generate the velocity model using linear regression function from sklearn
vel_model = linear_model.LinearRegression()
vel_model.fit(press_rs, veldiff_rs)
```

This function produces a best-fit for the data by minimizing the squared-loss function (Eq. 3.16). The parameters that minimize the loss function in the linear_model function are stored as vel_model.coef_[0] and vel_model.intercept_ values, which correspond to the slope and intercept of a straight line, respectively.

Because $\Delta U$ is not expected to vary significantly with respect to $\Delta p$, the use of the $R^2$ score to assess the model is not appropriate as it was with the pressure model. Instead, the success of the fit is assessed directly by the predictive power of the model to the test dataset under the normal distribution assumption and the 95% confidence interval. As with the pressure model the threshold for predictive power was chosen to be 85%. The model values are reported to the user and the results of the ML fit to the velocity model are shown graphically (see Chapter 6 for examples).

## 5.2 Monitoring New Data for Non-Conformance

In process monitoring, a control limit is used to determine the point at which a process has shifted from a previous trend to a new one [Kalpakjian and Schmid 2006, Miller and Freund 1965]. In the case of processes that follow a normal distribution with a population size greater than 20, an upper limit is commonly placed at plus 3 standard deviations (+3SD) of the model prediction, and a lower control limit at minus 3 standard deviations (-3SD) of the model prediction [Kalpakjian and Schmid 2006]. A new observation that is greater than +3SD or less than -3SD of the behavior predicted by the model is considered "out of control" (OOC) and likely indicative of a significant

event. Repeating OOC values are indicative of a new trend in the behavior. In the case of the pressure model, which is expected to follow a normal distribution, the +/-3SD criterion is used in the program. However, in the example of the velocity model, this is not practical as the distribution is so narrow (a SD of approximately $4.7 \times 10^{-5}$ m/s from the CFD results) and likely within the measurement error of most flowmeters. In cases in which statistical assignment of control limits in process monitoring is not practical, it becomes a matter of practical significance. Therefore, more practical lower and upper control limits for this model were established at -0.05 and 0.05 m/s, respectively, as the results of the CFD simulations in which mass was conservated (all cases except leaks) fell within these limits.

When the ML algorithms can demonstrate their ability to predict unseen data under normal operating conditions, the program is considered to be trained and can be used to monitor new unseen data. When a new observation of fluid conditions at points 1 and 2 is received, the real $\Delta p$ is compared to the predicted $h$ calculated in the pressure model using the new $U_1$ and $T_1$ values. If the real $\Delta p$ is within the control limits of the normal distribution about $h$ the new observation is considered to conform to the pressure model. This is repeated for the velocity model: he real $\Delta U$ observed is checked for whether it falls between the control limits. Observations for either model that are not in control are flagged by the program, either as HIGH or LOW. Each new observation is checked for conformance to the algorithms. When five OOC flags are counted from new observations for each model the program determines that a new trend has been observed.

## 5.3 Inferential Analysis of New Trends

CFD simulation of several defect conditions found that the fluid parameters between two points on either side of a leak will behave differently from that of a flow restriction (see Chapter 4). A series of conditional criteria in the program are used to infer the cause of a new trend based

on these differences in flow behaviors. This process of identifying the potential fault condition is summarized in Fig. 5.1. When the new behavior is OOC to the velocity model, a leak is suspected as this is the only fault observed to significantly break mass continuity. Because the $\Delta U$-$\Delta p$ relationship of a leak under all conditions resembled a power function of positive exponent in CFD data (Fig. 4.7), the program uses a third curve_fit function:

```
#fit function for fault analysis
def faultFunc(x, a, b, c):
   return a*(x**b) + c
#fit the fault data to a power curve in faultFunc
   f_param, f_fitCov = curve_fit(faultFunc, fault_press, fault_vel_dif)
```

The value of the parameters are returned as an array, f_param[ ] and the exponent (as f_param[1]), if its sign is greater than zero, a leak is suspected. This is reported to the user as "LOSS IN FLOWRATE FOUND. LEAK SUSPECTED." Using Eq. 4.6 the volumetric flow rates of the loss based on $U_1$ are estimated from the data and these estimates are reported to the user graphically (see Chapter 6). If the exponent (f_param[1]) is less than zero, indicating that velocity is increasing, the program reports that an unknown fault or error has been identified and that investigation is required: "UNKNOWN CHANGE IN FLOWRATE. INVESTIGATION REQUIRED."

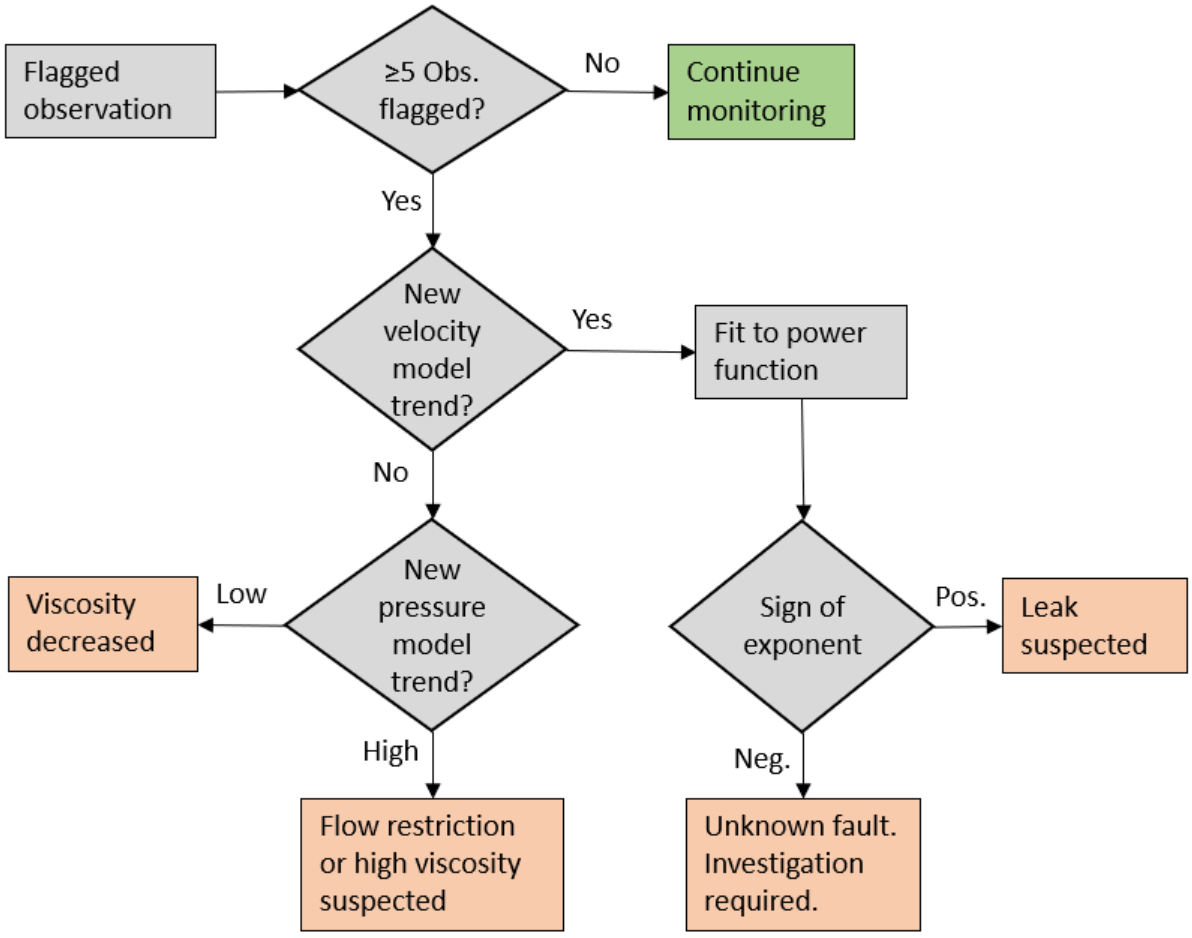



Fig. 5.1: Flow chart of the out-of-control decision-making process of the monitoring phase.

Continuing through Fig. 5.1, if there is not a new trend in the velocity model, but there is in the pressure model, the logic pathway splits based upon whether the new pressure trend is higher or lower than the model. If high, excess pressure loss between points 1 and 2 is assumed to be the result of either a flow restriction or an increase in viscosity of the oil and the message received by the operator is "EXCESS PRESSURE LOSS FOUND. FLOW RESTRICTION OR VISCOSITY INCREASE SUSPECTED." The excess pressure loss is calculated as the difference between the observed $\Delta p$ and the hypothetical $h$ from the model, and these estimated excess losses are reported to the user graphically (see Chapter 6). If low, the decrease in expected pressure loss is assumed

to be indicative of a decrease in fluid viscosity and the operator is alerted: “DECREASED PRESSURE LOSS FOUND. VISCOSITY DECREASE SUSPECTED.”

## 5.4 Summary of Machine Learning Program

This chapter discussed the programming and coding of the machine learning and the process monitoring algorithms developed in this thesis. It covered the methods by which the two ML algorithms (i.e. velocity and pressure) were trained using normal operational data obtained by CFD modelling (see Chapter 4). A mathematical pressure model of the relationship between $U_1$ and $\Delta p$ was developed using the curve_fit function and a mathematical velocity model of the relationship between $\Delta p$ and $\Delta U$ was developed using the LinearRegression function. These models are validated using the test dataset to assess the fit and limit training bias. New data points are then compared to each of these models to determine if the real output values are statistically in control with respect to the models. If they are not, a series of logic statements are used to determine the probable nature of the fault based on inference from the CFD. An OOC condition will yield one of four potential fault conditions:

1) Leak suspected.
2) Flow restriction or high viscosity suspected.
3) Viscosity decrease suspected.
4) Unknown change in flowrate.

Chapter 6 will use this ML program and the CFD-derived data in Chapter 4 to demonstrate that this program is capable of accurately learning the normal flow behavior of the pipeline system under study. Then, when introduced to new observations, this program is capable of recognizing

data from a fault condition as distinct from the ML models and infer the specific type of fault most likely under observation.

# CHAPTER 6: RESULTS OF MACHINE LEARNING AND FAULT IDENTIFICATION

Chapter 6 is divided into three primary sections: Section 6.1 details the results of ML algorithm training with data of simulated normal operating conditions and discusses the success metrics. Section 6.2 details the program's ability to identify and characterize the different fault conditions. Section 6.3 provides a discussion of the significance of the findings. A detailed description of the background of ML can be found in Section 3.3 and a detailed explanation of the ML program methodology in Chapter 5.

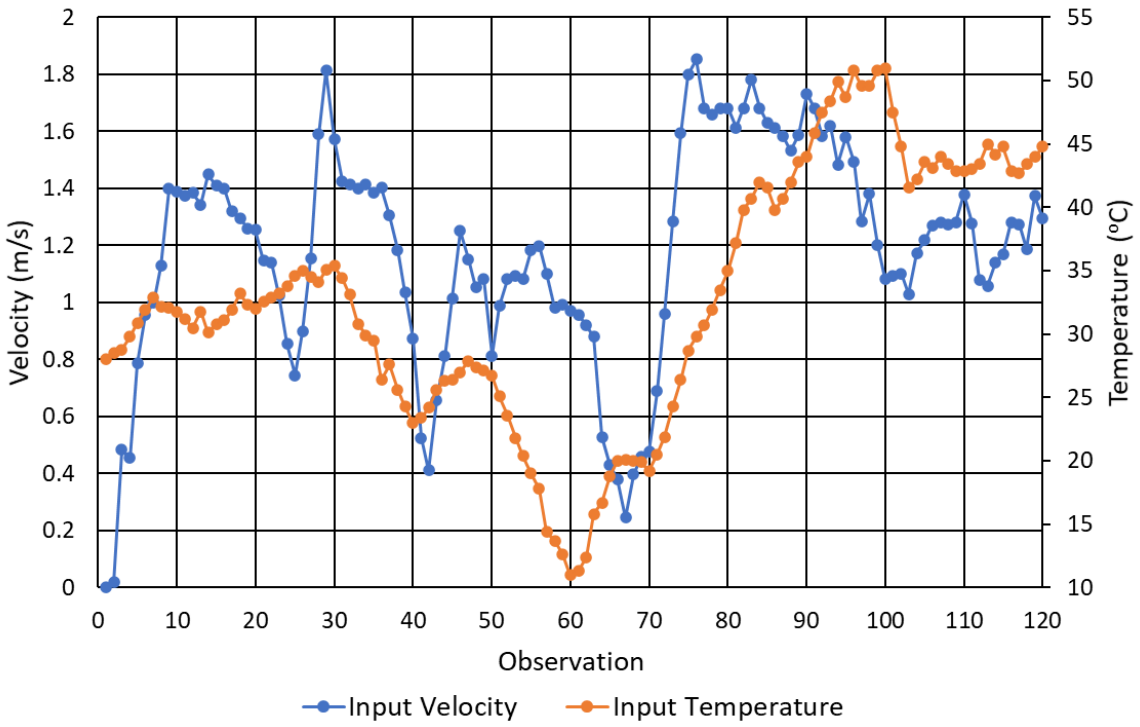



Fig. 6.1: Input velocity and temperature of a "real situation" for CFD simulation and ML training.

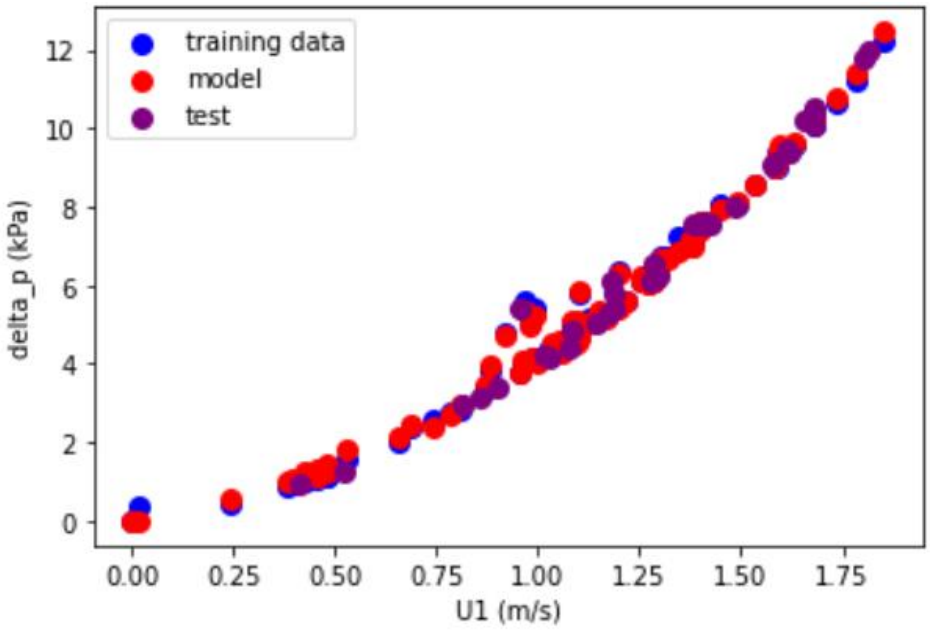
## 6.1 Machine Learning Training Under Normal Pipeline Conditions

To provide a realistic dataset to train the ML algorithms a time-series of 120 observations was generated that represent a progression of readings that a crude oil pipeline system in a subsea

environment may report, with velocity and temperature readings within the range of typical operations discussed in section 4.2 (Fig. 6.1). Real data from pipeline systems were not available, but these ranges are very similar to what has been published in the literature [Barker et al. 2014]. Figure 6.1 shows the input velocities and temperatures used to simulate the flow of this hypothetical "real situation" in which flow velocities at the input (z = 0 m) range between 0 and 1.85 m/s and temperatures vary between 11 $^{o}$C and 51 $^{o}$C. This data-series was simulated in Fluent under normal conditions described in Chapter 4 and the resultant point 1 and 2 data collected onto a CSV for import to the program.

```
Pressure Model:
   delta_p = 2.301*T1^0.0893*U1^2 + 37.677*T1^-1.0744*U1
   Model R^2 score: 0.998
   Predictive power: 1.00
```

(a)

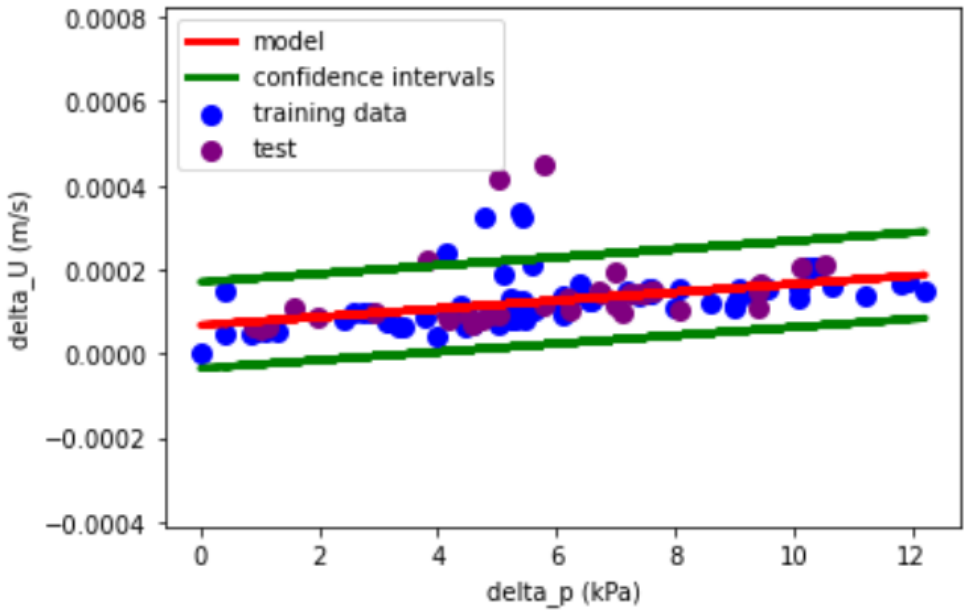



(b)

Fig. 6.2: Regression ML fitting results for the pressure algorithm: reported model (a) and the graphical output (b).

```
Velocity Model:
   delta_U = 0.000009224*delta_p + 0.000076158
   Predictive power: 0.97
```

(a)




(b)

Fig. 6.3: Regression ML fitting results for the velocity algorithm: reported model (a) and the graphical output (b).

The ML learning algorithm successfully generated pressure and velocity models of the pipeline system under normal flow conditions with predictive powers greater than 85% for both models, and an $R^2$ score greater than 80% in the case of the pressure model. The pressure model results shown in Fig. 6.2 shows the model with the parameters generated from the fitting function curve_fit in the form of Eq. 3.13. In this iteration the $R^2$ score is very high, indicating that the model described the training set with very little error. This result suggests the assumption in this thesis that, given the governing equations, the process of model selection could be restricted to the form of Eq. 3.13 and fitting additional models (linear, logarithmic, parabolic, etc) was not necessary. A strong fitting result does, however, present the risk of training bias. That is to say,

the model is so well fitted to the training data—and only the training data—that other real values will not be recognized. This is why the reserved 30% test data (36 points of the 120 observations) is used to assess the predictive power. The results in Fig. 6.2 demonstrate that the predictive power was excellent at 100%. Training bias is reduced primarily by ensuring the training data is well representative of the system under study, and the training-test split is random. Fig. 6.2b displays the fitting of the test data (purple) to the model (red) in comparison to the training set (blue). The training data, test data, and model overlap well, providing visual conformation of the quality of the fit, and confirmation of the parabolic relationship between $U_1$ and $\Delta p$. Note that this figure does not include an axis for temperature on the model as its representation is not visually significant.

Figure 6.3 displays the results of the LinearRegression fit to the velocity model. As noted in Chapter 5 an $R^2$ score is not included. However, the predictive power of 97% is significant, suggesting low generalization error and low training bias. The red model line in the graphical representation (Fig. 6.3b) describes the linear nature of the fit, and the training and test data are shown to overlap well. Between the 4 and 6 kPa values of $\Delta p$, several data points are shown to rise above the upper confidence level (green). When examining these points it was identified that they likely represent a divergence from the velocity model due to extreme effects of temperature on turbulent effects in the CFD. However, this effect is negligible in a practical sense as the $\Delta U$ in these several data points are not more than about 0.5 mm/s, well below the resolution of a typical industrial flowmeter.

These results in the fitting process demonstrate that the ML program successfully fit both the pressure and velocity models to the training data with minimal generalization error and training bias.

## 6.2 Identification of Out of Control Conditions and Fault Characterization

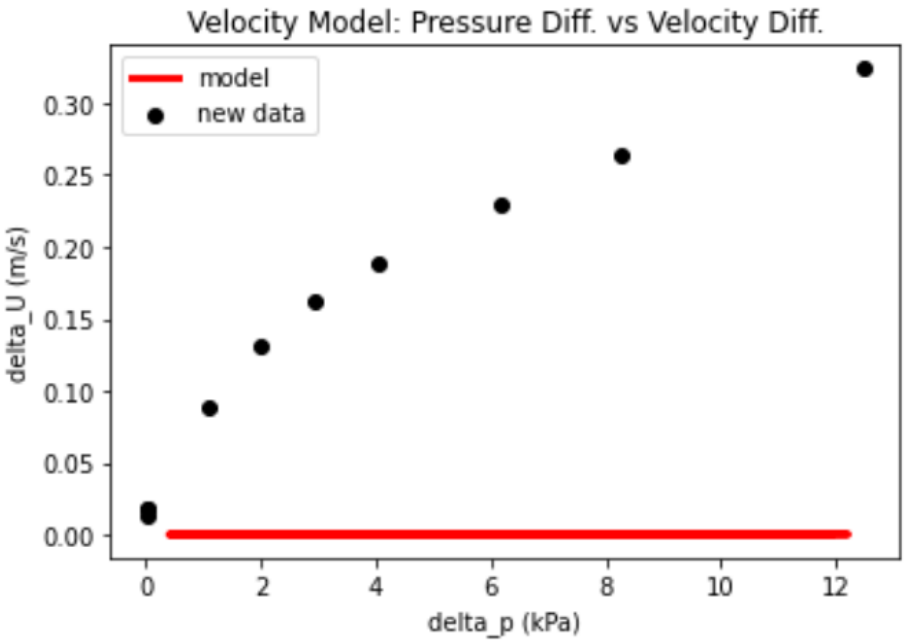
### 6.2.1 Leak Condition

When presented with new observations from all four conditions of a leak at the 125 m position of the pipeline, the ML could identify the observed behaviors as OOC to the learned models' predictions and could identify each as a potential leak from the loss of flowrate between points 1 and 2. When presented with new data generated under the 20% leak condition with a 0 Pa pressure difference between the outlet and the leak location, the program categorized each new observation as being in control (a Boolean "True") or OOC "HIGH" or "LOW" for the velocity and pressure models (Fig. 6.4). In this case, the majority of the observations were in control of the pressure model. However, as expected from mass conservation, the majority were identified as OOC HIGH in the velocity model except for those with very low velocities. Because more than five data points were identified as OOC HIGH in the velocity model the program reports "New trend in VELOCITY model" with the divergence of the observations from the velocity model indicated graphically in which the power relationship is evident (Fig. 6.5). Using Eq. 4.6 the loss in material to the environment based on velocity is estimated and reported graphically.

| Item | U1 | T1 | U2 | delta_p | Vel. Model | Press. Model |
|---|---|---|---|---|---|---|
| 1 | 1.792 | 19.2 | 1.468 | 12.4889 | HIGH | True |
| 2 | 0.493 | 12.3 | 0.405 | 1.0861 | HIGH | LOW |
| 3 | 1.259 | 31.8 | 1.030 | 6.1934 | HIGH | True |
| 4 | 0.030 | 26.7 | 0.017 | 0.0128 | True | True |
| 5 | 0.087 | 40.2 | 0.069 | 0.0327 | True | True |
| 6 | 0.687 | 23.3 | 0.556 | 1.9840 | HIGH | True |
| 7 | 0.988 | 16.5 | 0.799 | 4.0430 | HIGH | LOW |
| 8 | 1.453 | 25.1 | 1.189 | 8.2511 | HIGH | True |
| 9 | 0.823 | 14.8 | 0.661 | 2.9299 | HIGH | LOW |
| 10 | 0.075 | 35.0 | 0.056 | 0.0299 | True | True |

Fig. 6.4: Tabular read-out of the new data points under a 20% (0 Pa) leak with the OOC determination for each model. Units: velocities (U1, U2) in m/s, temperature T1 in $^{o}$C, and pressure in kPa.

```
New trend in VELOCITY model.
```




```
LOSS IN FLOWRATE FOUND
LEAK SUSPECTED
```

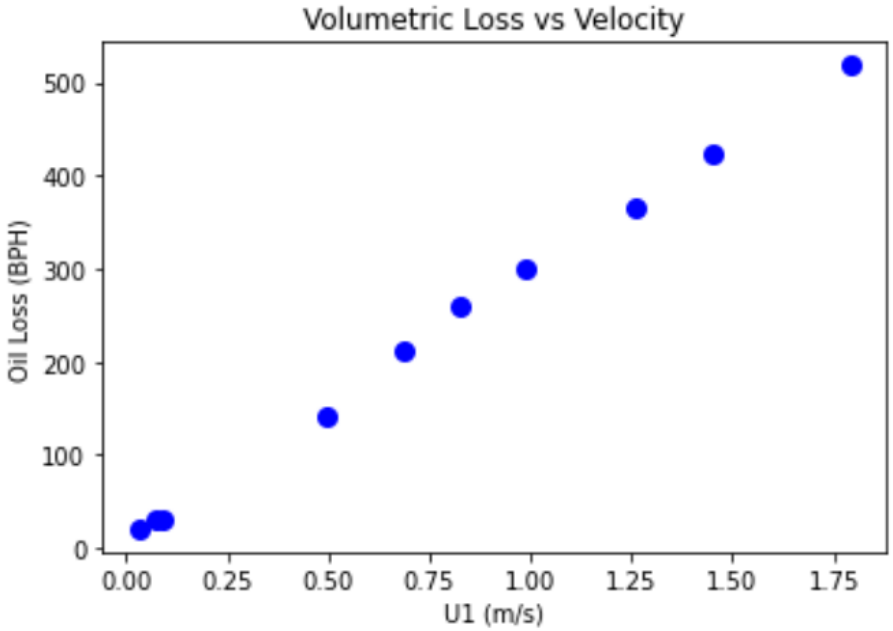
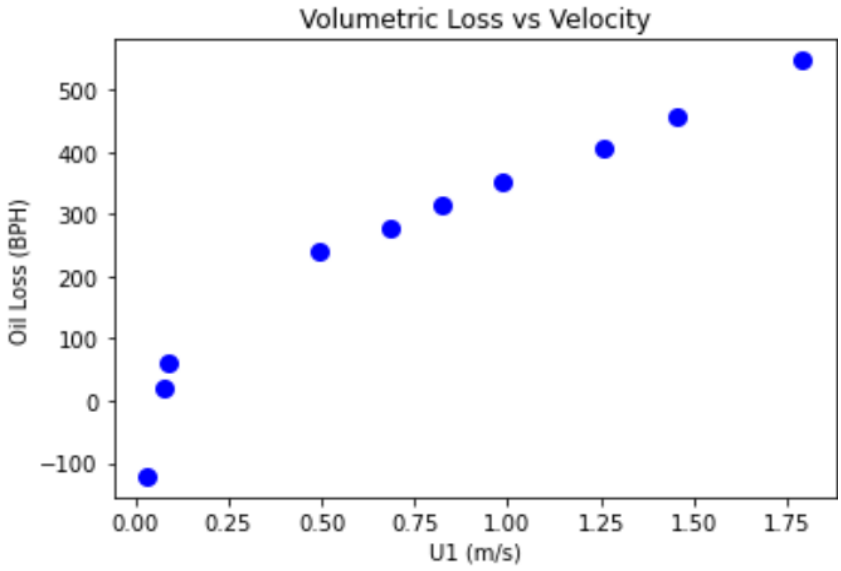



Fig. 6.5: Reports of the ML program in a 20% leak condition (0 Pa) indicating the velocity model observed a new trend and the suspected cause is a leak of the given magnitude.

Under the 20% leak conditions with outlet-leak pressure differences of -500 Pa and +500 Pa the program was also successful in identifying these as probable leaks based on the OOC condition of their respective velocity models and could estimate the volumetric losses in both cases (Fig. 6.6). Consistent with the behavior noted in Fig. 4.7, the volumetric loss under the +500 Pa environmental pressure (Fig. 6.6b) dips somewhat significantly into the negative region at velocities of about 0.50 m/s and less. This indicates that in this velocity region the environmental pressure at the region of the leak was sufficient to overcome the oil fluid pressure at this point and might entrain sea water into the oil flow. It should be noted, however, that the purpose of this thesis is not to predict nor identify seawater entrainment, but that this may be an opportunity for further research in this field.

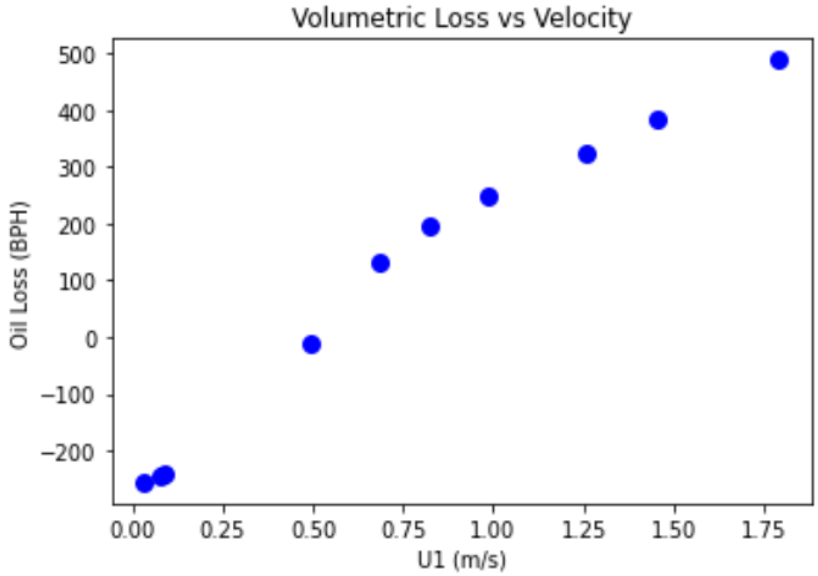
When considering the 10% leak condition the ML program found that both the velocity model and the pressure model were OOC, with the pressure model over-predicting the $\Delta p$ behavior under this condition, suggesting that in this condition the leak slightly decreased the change in pressure from normal. To identify the potential fault condition, the ML program was still successful at reporting this as a leak with the estimated volumetric losses characterized. As shown in Chapter 4, minor effects in the pressure profile were observed in all cases of leak conditions (Fig. 4.8), and as explained in Chapter 5 (Fig. 5.1) the velocity model is the diagnostic model in determining leak condition and estimating the characteristic loss. This illustrates how sensitive this machine learning technique can be in identifying new trends in data. Even a minor shift in the pressure profile was identified as not conforming to the model of normal conditions. When the pressure model control limits were increased from +/-3SD to +/-4SD, the program no longer found the 10% leak condition as OOC for the pressure model.

```
LOSS IN FLOWRATE FOUND
LEAK SUSPECTED
```




(a)

```
LOSS IN FLOWRATE FOUND
LEAK SUSPECTED
```




(b)

Fig. 6.6: Volumetric loss estimates of the 20% leak condition at -500 Pa (a) and +500 Pa (b) as reported by the ML program.

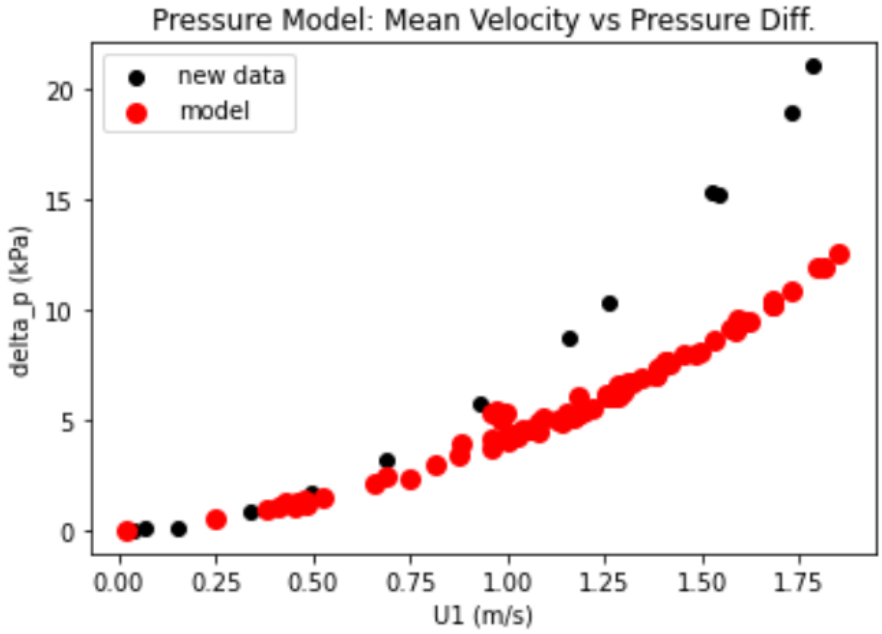
### 6.2.2 Identification of an Increase in Pressure Loss

Under the conditions of a 50% obstruction at the 125 m position of the pipeline the program could determine that the new observations were OOC for the pressure model and that a new trend in the data had occurred. From the tabular read out in Fig. 6.7, all new observations were in control of the velocity model while a majority were OOC HIGH for the pressure model. Those in control represent observations of velocities less than 0.4 m/s where the difference in $\Delta p$ is negligible. This is further visualized in the ML readout of the pressure model displaying the "New HIGH trend in PRESSURE model" with the upward shift in the pressure curve relative to the model of normal behavior in which the pressure model underpredicted $\Delta p$ significantly (Fig. 6.8). Having identified the OOC trend in the pressure model the program could quantify the extent of the change as an expression of excess pressure loss (the new observed $\Delta p$ minus the model $\Delta p$) as a function of velocity and reports this as "EXCESS PRESSURE LOSS FOUND, FLOW RESTRICTION OR VISCOSITY INCREASE SUSPECTED." The relationship is shown to be roughly parabolic with increasing excess pressure loss with increasing pipe flow velocity.

| Item | U1 | T1 | U2 | delta_p | Vel. Model | Press. Model |
|---|---|---|---|---|---|---|
| 1 | 0.685 | 23.3 | 0.703 | 3.2258 | True | HIGH |
| 2 | 1.159 | 45.3 | 1.189 | 8.6827 | True | HIGH |
| 3 | 0.067 | 9.3 | 0.066 | 0.1090 | True | True |
| 4 | 0.033 | 27.9 | 0.034 | 0.0158 | True | True |
| 5 | 1.525 | 17.4 | 1.565 | 15.2443 | True | HIGH |
| 6 | 0.337 | 15.5 | 0.347 | 0.8761 | True | True |
| 7 | 0.148 | 19.6 | 0.154 | 0.1242 | True | True |
| 8 | 0.929 | 28.3 | 0.953 | 5.7312 | True | HIGH |
| 9 | 1.262 | 30.9 | 1.295 | 10.3341 | True | HIGH |
| 10 | 1.546 | 37.0 | 1.585 | 15.2396 | True | HIGH |
| 11 | 1.729 | 41.2 | 1.774 | 18.9350 | True | HIGH |
| 12 | 1.786 | 14.8 | 1.833 | 21.0046 | True | HIGH |
| 13 | 0.494 | 38.2 | 0.507 | 1.6831 | True | HIGH |

Fig. 6.7: Read-out of the new data points under a 50% flow restriction with the OOC determination for each model. Units: velocities (U1, U2) in m/s, temperature T1 in $^{o}$C, and pressure in kPa.

```
New HIGH trend in PRESSURE model.
```




```
EXCESS PRESSURE LOSS FOUND
FLOW RESTRICTION OR VISCOSITY INCREASE SUSPECTED
```

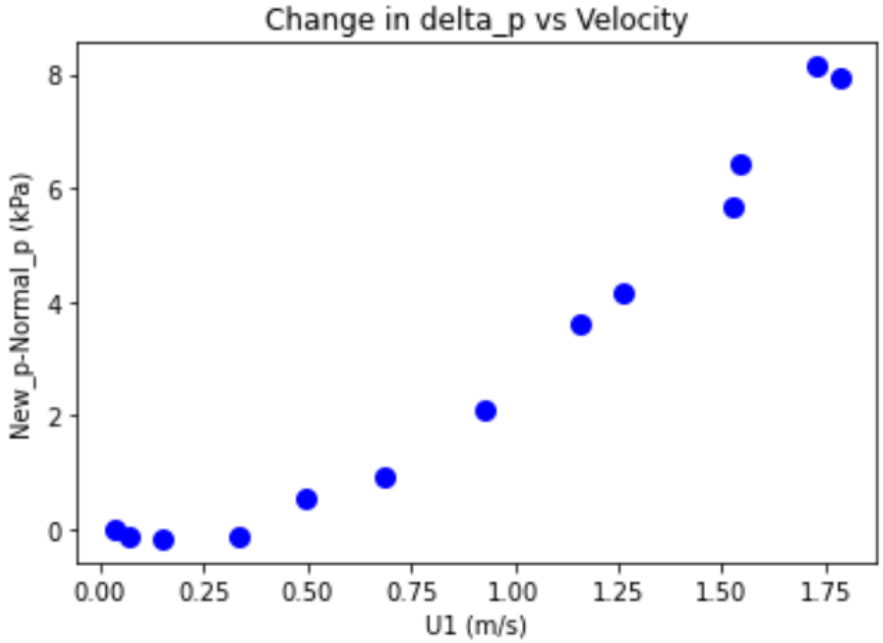



Fig. 6.8: Reports of the ML program in a 50% flow restriction indicating the pressure model observed a new trend and the suspected cause is a flow restriction or increase in fluid viscosity.

When the data developed under a five-fold increase in the viscosity of the model fluid is presented to the ML program, the results are very similar to that of the 50% flow restriction except for a less significant increase in pressure loss in the case of the viscosity shift (Fig. 6.9). As noted previously, this presents a challenge in identifying definitively the cause of an excess pressure loss in the pipeline using strictly the flow data available. This thesis is not exhaustive, however, in the examination of the variable conditions, and further work may identify data patterns that may be exploited as further diagnostic in this respect. In a practical sense, the exact nature of an unexplained loss in pressure may not be of direct importance to a ML application, but simply the ability to identify such a condition.

```
EXCESS PRESSURE LOSS FOUND
FLOW RESTRICTION OR VISCOSITY INCREASE SUSPECTED
```

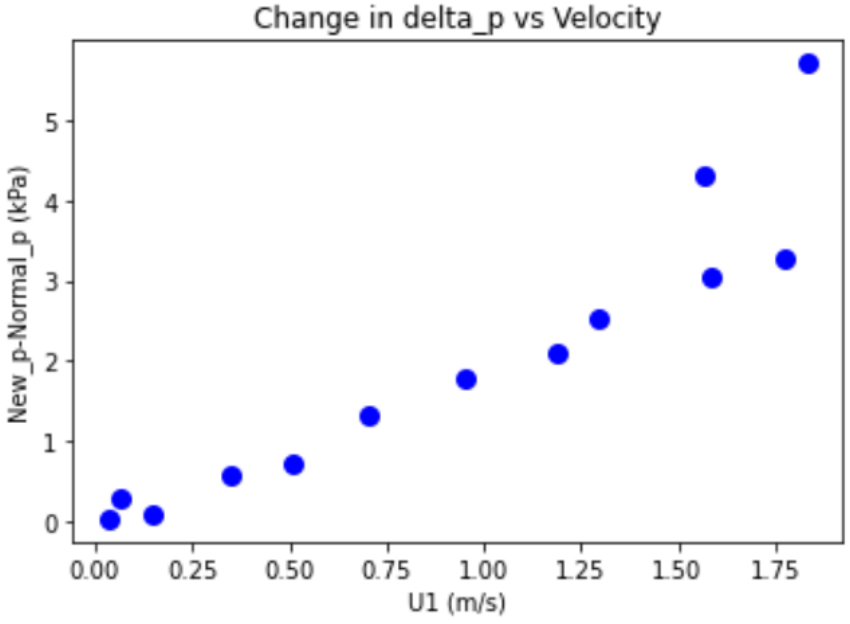



Fig. 6.9: ML program's estimation of excess pressure loss in the case of a five-fold increase in the viscosity of the crude oil.

Following the identification of a shift in the pressure profile, investigation of the rheological properties or composition of the fluid following the change may indicate if a change of material property is a likely cause. If the properties are found unchanged, or not significantly changed, further investigation would be warranted. If additional points along the pipeline (for example, a point 2 to point 3 span) or additional branches carrying the same oil are similarly under monitoring, it is expected that a change in pressure profile would similarly be observed if the cause is related to the fluid property. If the change remains localized to only one span this would support the suspicion of a localized flow restriction, and additional inspection may be necessary.

### 6.2.3 Identification of a Decrease in Pressure Loss

The final condition investigated in this thesis is the case of a decrease in the temperature-dependent viscosity of the crude oil. When presented with the simulated data of a normal pipeline with a viscosity profile reduced by half, the program was capable of identifying this as an OOC LOW trend in the pressure model (Fig. 6.10). The program reports "DECREASED PRESSURE LOSS FOUND. VISCOSITY DECREASE SUSPECTED" and the degree of change in the pressure loss if displayed graphically in which the trend in the change is negative (Fig. 6.11). As with the case of the high-pressure trend in the previous section, additional investigation into the rheological properties, or the behavior of other spans under monitoring, would confirm the characterization.

```
New LOW trend in PRESSURE model.
```

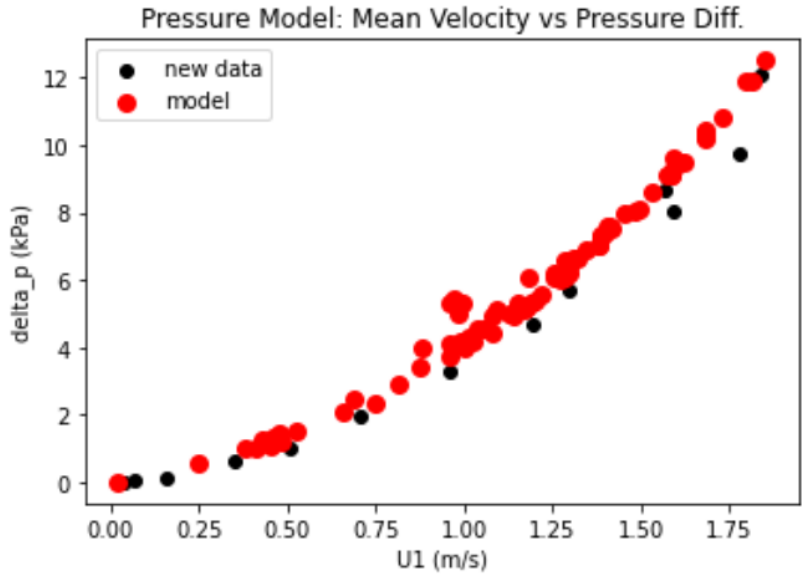



Fig. 6.10: Report of the ML program with the viscosity reduced by half indicating the pressure model observed a new LOW trend.

```
DECREASED PRESSURE LOSS FOUND
VISCOSITY DECREASE SUSPECTED
```

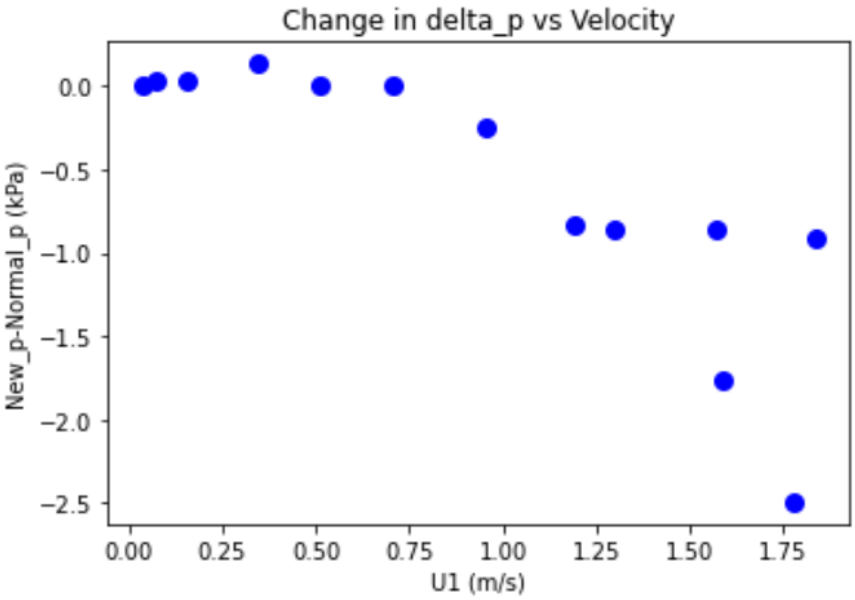



Fig. 6.11: Report of the ML program with the viscosity reduced by half quantifying the extent of the change in pressure loss as a function of $U_1$.

## 6.3 Discussion of Results

A distinct benefit for the application of machine learning in analyzing a complex and dynamic system is that, in general, they require little or no a priori understanding of the specific system under study, but instead identify patterns in the data provided. This thesis modelled a straight pipeline with an inner diameter of 0.3 m with sensors spaced 150 m apart as a representative example. The ML algorithm is effectively dimensionless as it incorporates geometry into the fitting of parameters during training based on the data, not a priori inputs, the algorithms used in this study could be trained using data collected from pipelines of different geometries (and different fluids). The ML models generated from the learning would take this into account and so long as the system itself does not significantly change (i.e., constant geometry and fluid) the general behavior of the fluid through that pathway should follow the governing equations.

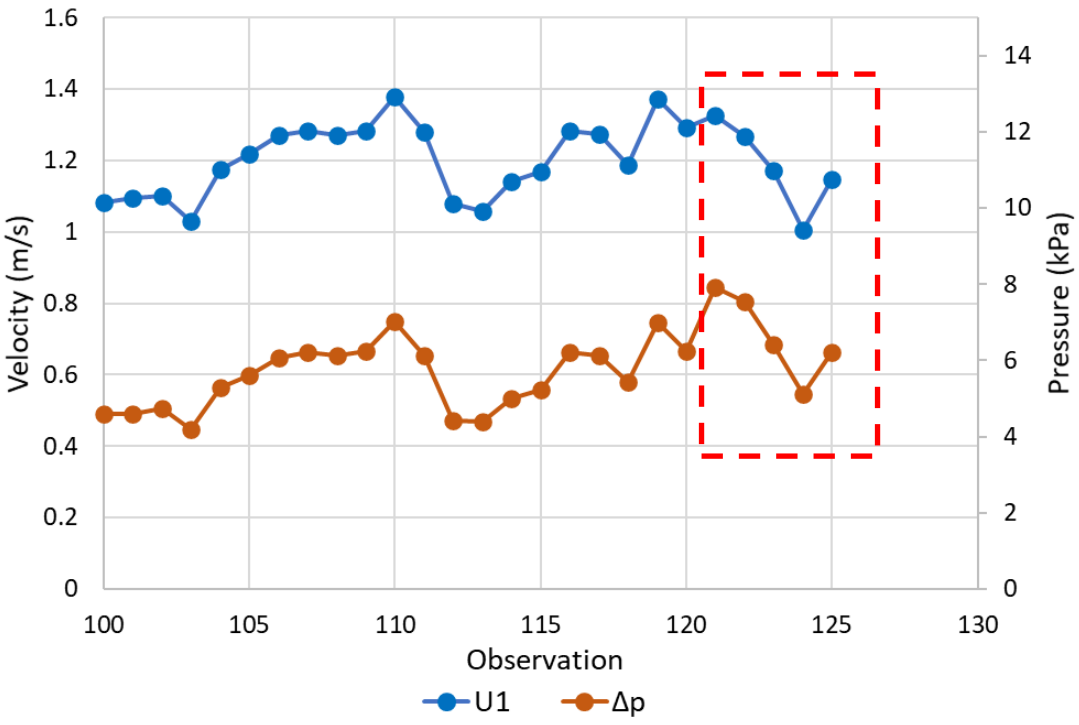



Fig. 6.12: $U_1$ and $\Delta p$ by observation under normal conditions up to observation 120, followed by a 10% leak in observations 121-125 (boxed).

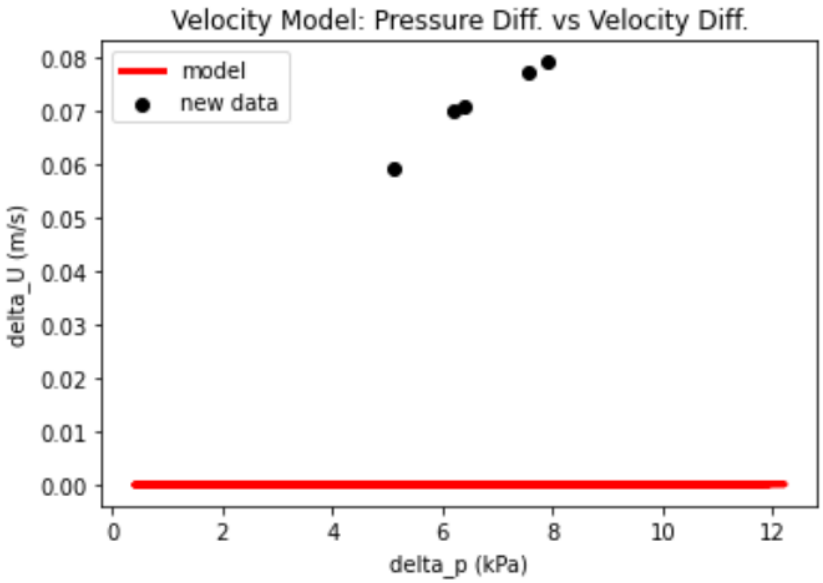
This thesis was limited to the use of pressure, temperature, and velocity because these are parameters commonly monitored in subsea pipeline systems through flowmeters and thermocouples and would not require the installation of special equipment. The ML multivariate regression algorithm for the pressure model in this study was instructed to fit training data to the form of Eq. 3.13 to simplify model selection to a straight uniform pipe; however, this model may apply to curved pipes of moderate radii. This principle would apply to applications of the velocity model as well. This is not exclusive of crude oil as the model fluid either; however, as the momentum and energy equations under investigation in this thesis assumed viscous flow, care would be required in applying this to inviscid flow.

At present, traditional methods of monitoring a subsea pipeline primarily involve monitoring oil volume output for significant changes using limits and alarms to warn operation management of potential faults, or through direct observation of faults. These methods have their limitations in that they are optimized for normally distributed processes or are not sufficiently sensitive to disturbances in dynamic systems. In practice, minor and moderate faults are often not found until a routine visual inspection or by the aid of ROVs and sonar sensors.

Figure 6.12 demonstrates an example situation in which a 10% leak forms immediately following observation 120 of the dataset examined in section 6.2. From the data in this figure it is not readily apparent that the five subsequent observations (121-125) occurred under a fault condition as the change in the velocity and pressure behavior is minute. Traditional monitoring is unlikely to identify this as a potential fault. The ML algorithm is, however, capable of identifying that these five points do not conform to the model established during the learning process and flags them very easily as indicative of a suspected leak (Fig. 6.13). The sensitivity of the algorithms to learn from the real data of the system and identify when observations do not conform to the learned

behavior/model enables it to be a very powerful tool in detecting adverse conditions. The example in Figs. 6.12 and 6.13 required only five data points to identify the relatively small leak. The speed at which the program is able to make this determination would permit significantly faster responses to faults.

```
New trend in VELOCITY model.
```




```
LOSS IN FLOWRATE FOUND
LEAK SUSPECTED
```

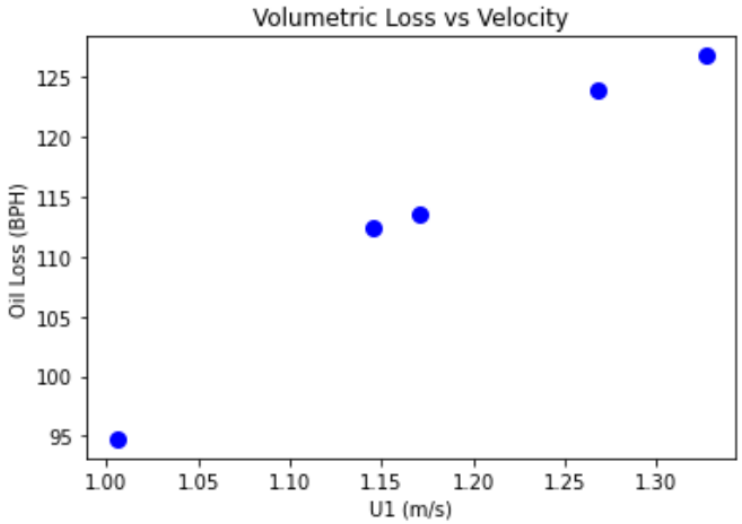



Fig. 6.13: ML output of the 10% leak example in Fig 6.12: Velocity model and estimated volumetric loss.

The ML program developed in this thesis does not rely on an understanding of the physical principles involved in the flow and it cannot determine the nature of an OOC condition by itself. Given the governing equations and the behavior observed in the CFD, however, it can be assisted in inferring the cause of OOC situations. Inferences of the probable nature of a pipeline fault can be made given patterns observed in the data of simulated faults. The fluid behavior observed in the ANSYS Fluent simulations of a leak exhibited a positive power relationship between $\Delta p$ and $\Delta U$ demonstrating a relative decrease in mean velocity between points 1 and 2 compared to the continuity of velocity under normal conditions, as would be expected in the case of loss of mass to the environment (Eqs. 3.4 and 4.6). The pressure model found statistically higher $\Delta p$ as a function of $U_1$ compared to the model of normal conditions. The presence of an obstruction to flow presented or an increase in the viscosity of the oil resulted in a rise in $\Delta p$ in the pressure model relative to normal conditions. A decrease in the fluid viscosity was observed to suppress the pressure curve. These characteristic differences in behavior become diagnostic in defining the probable nature of a new trend of non-conforming observations (Fig. 5.1).

Figure 6.14 outlines this study's scheme of ML algorithm training and pipeline process monitoring. To successfully train the ML a sufficient quantity of data points should be collected under conditions known to be free of faults and encompasses the range of expected operational parameters (flow rates, well-head pressures, temperatures, etc). Once a representative training data set has been collected, the ML algorithm should be trained and evaluated with the test set of known data. Following the successful training of the model new observations should be evaluated for conformance to the model and those that do not conform to either the pressure model or the velocity model are flagged by the algorithm. Conforming observations are logged but are not incorporated into the trained model. Once the trained model has been set it should remain static

for the remainder of the monitoring phase to prevent model drift over time, as might occur with delayed fowling or wax deposition. In this way, the pipeline system is always compared to the initial optimal condition.

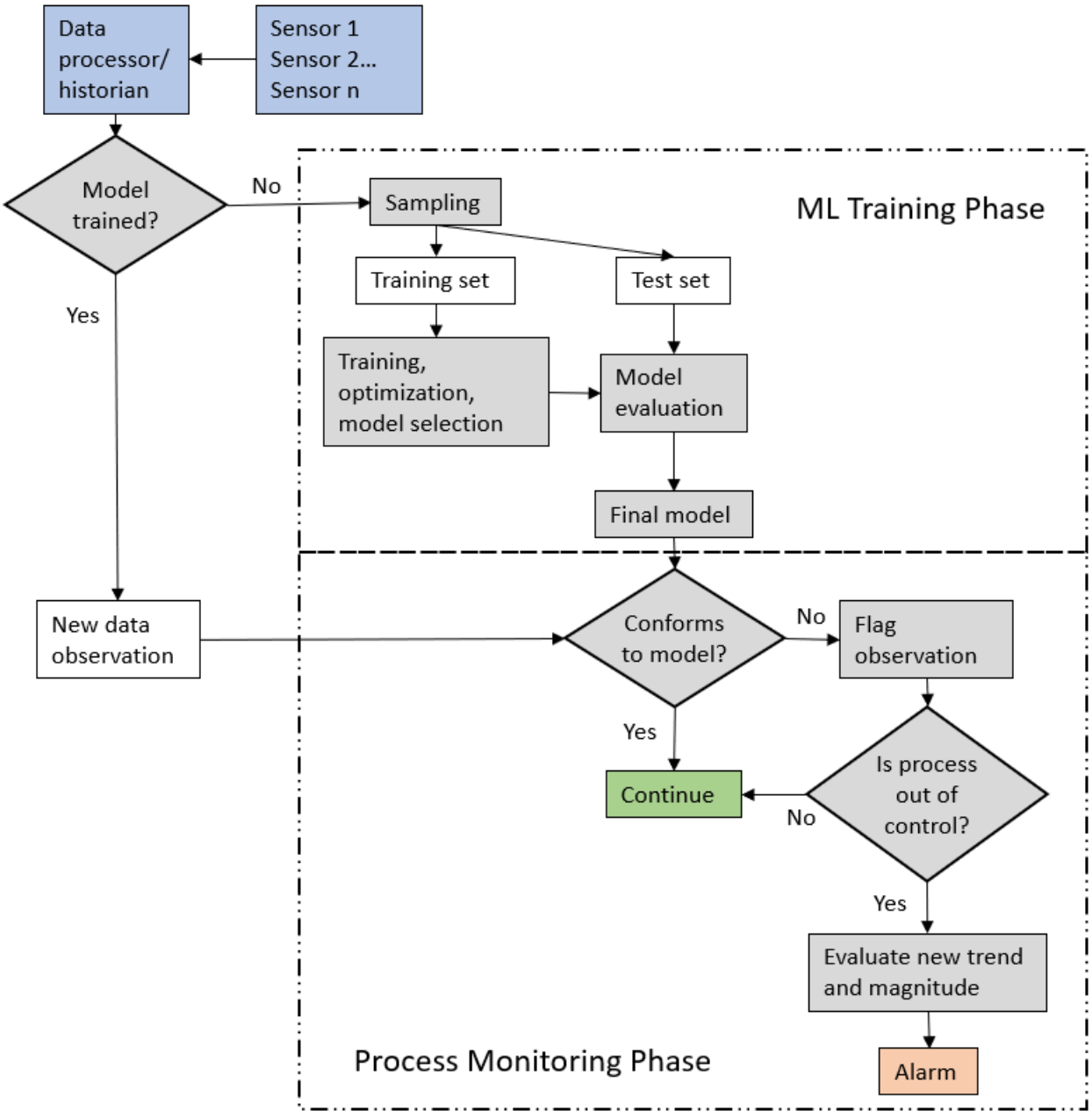



Fig. 6.14: Flow chart of the thesis ML program, training and process monitoring phases.

Figure 5.1 outlines the process monitoring method of determining if the process is out of control and whether this is indicative of a fault. The program counts flags up to five as random outliers and establish a new pattern of behavior. The ML algorithm will alarm operations personnel of a potential fault. The ML algorithm developed for this thesis demonstrates using simulated flow data that machine learning could be a powerful tool in monitoring a dynamic process such as the flow of crude oil through a pipeline. This study was not exhaustive in terms of the potential faults a pipeline may encounter, or the numerous conditions under which a pipeline may experience a leak/rupture or a restriction to flow. It does, however, present a proof of concept that merits further investigation and development.

This thesis was limited in scope to several theoretical fault conditions of specific extent of a pipeline of selected geometry. This was to serve as a proof of concept: that a machine learning program could be developed and employed to detect faults in a subsea oil pipeline. This thesis recognizes that it does not represent a broad-ranging sample of potential fault conditions and quantities, nor does it account for minimal detection limits or potential sources of noise. These would be highly dependent upon a specific application: the location and environment of a pipeline; the fluid properties; the quality, sensitivity, and accuracy of the instruments; and the operational conditions. Additional considerations would be acceptable risk thresholds tolerated by petroleum operations engineers and managers.

# CHAPTER 7: CONCLUSIONS AND FUTURE WORK

## 7.1 Conclusions

This study has proposed a method of monitoring a subsea oil pipeline for fault detection by way of a machine learning program that is trained using real data collected from the pipeline system under normal operations. Two supervised ML algorithms were developed that used temperature, pressure, and flow velocity data derived from an ANSYS Fluent simulation of a uniform pipeline to characterize the fluid mechanics of oil flow under normal conditions as a multivariate regression pressure model and a univariate linear regression velocity model. These trained models were verified by a test set of known results to demonstrated that the ML algorithm could accurately predict outputs. It was then demonstrated that this algorithm could identify new observations under simulated faults as non-conforming to the model, indicating that a ML program is capable of distinguishing between normal operating conditions and conditions of a pipeline fault by recognising an out of control (OOC) situation using the trained models. It was further demonstrated that inferences of the nature of the OOC data in the velocity and pressure model profiles could be used to infer the type of fault and the degree of deviation from normal.

This thesis examined four general fault or adverse situations:

1. Leaks of 10% and 20% of the nominal cross section of the pipeline.
2. Flow restriction of 50% of the nominal cross section of the pipeline.
3. Increase of the viscosity of the fluid.
4. Decrease of the viscosity of the fluid.

The use of such a ML program presents several benefits over traditional methods of process monitoring: 1) The analysis of data is automatic and in real-time as new observations are recorded from the environment. 2) The machine learning program requires no prior assumptions of the

system's geometry or fluid properties. 3) Indications of potential faults are detected early and oil well operators are alerted immediately once a new pattern is confirmed. 4) No special equipment installation is required apart from meters already common to the industry. 5) The potential fault can be isolated between two points, reducing investigation costs and risks. 6) This system is scalable and can operate as a network of pipeline segments across oil fields and transport lines.

## 7.2 Recommendations for Future Work

This study provided a proof of concept for the conditions discussed but was not intended to be an exhaustive analysis of the potential applications or types of defects, adverse situations, and environmental conditions that a crude oil pipeline may experience. There are several areas of further study that may advance this field of research:

1. Various geometries and operational conditions.
2. Training and testing of the ML using real-world flow data.
3. Installation of an ML system into an empirical environment.
4. Expansion into networks of ML-monitored segments of pipeline and other subsea components.

# REFERENCES


Adegboye, M.A., Fung, W., Karnik, A., “Recent advances in pipeline monitoring and oil leakage detection technologies: principles and approaches,” 19, 2548, 2019.

Ahn, B., Kim, J., “Artificial intelligence-based machine learning considering flow and temperature of the pipeline for leak early detection using acoustic emission,” Eng. Fracture Mech., 210, pp. 381-392, 2018.

Alipour, M., Esatyana, E., Sakhaee-Pour, A., Sadooni, F.N., and Al-Kuwari, H.A., “Characterizing fracture toughness using machine learning,” Journal of Petroleum Science and Engineering, 2020.

Anifowose, F., Adeniye, S., Abdulraheem, A., Al-Shuhail, A., “Integrated seismic and log data for improved petroleum reservoir properties estimation using non-linear feature-selection based hybrid computational intelligence models,” Journal of Petroleum Science and Engineering, 145, pp 230-237, 2016.

Baker, R., A Primer of Offshore Operations, Ed. 3. 1998. University of Texas at Austin, Austin, Texas.

Barker, R., Hu, X., Neville, A., and Cushnaghan, S., “Empirical prediction of carbon-steel degradation rates on an offshore oil and gas facility: Predicting CO2 erosion-corrosion pipeline failures before they occur,” Society of Petroleum Engineers Journal, June 2014.

Bird, R., Stewart, W., Lightfoot, E., Transport Phenomena, Ed. 2, Wiley Publishing, 2006.

Brunton, S.L., Noack, B.R., Koumoutsakos, P., “Machine learning for fluid mechanics,” Annual Review of Fluid Mechanics, 52, pp. 477-508, 2020.

Chavolla, E., Valdivia, A., Diaz, P., Zaldivar, D., Cuevas, E., and Perez, M.A., "Improved unsupervised color segmentation using a modified HSV color model and a bagging procedure in K-means++ algorithm," Mathematical Problems in Engineering, 2018.

Cherkassky, V., Mulier F.M., Learning from Data: Concepts, Theory, and Methods. 2007. John Wiley & Sons Publishing, Hoboken, New Jersey.

Creton, B., Leveque, I., Oukhemanou, F., "Equivalent alkane carbon number of crude oils: a predictive model based on machine learning," Oil & Gas Sci. and Tech., 74(30), 2019.

Davis, P., Brockhurst, J., "Subsea pipeline infrastructure monitoring: A framework for technology review and selection," Ocean Engineering, 104, pp. 540-548, 2015.

Elforjani, M., and Shanbr, S., "Prognosis of bearing acoustic emission signals using supervised machine learning," IEEE Transactions on Industrial Electronics, 65(7), 2018.

Fu, H., Yang, L., Liang, H., Wang, S., Ling, K., "Diagnosis of the single leakage in the fluid pipeline through experimental study and CFD simulation," Journal of Petroleum Science and Engineering, 193, 2020.

Hadavimoghaddam, F., Ostadhassan, M., Heidaryan, E., Sadri, M. A., Chapanova, I., Popov, E., Cheremisin, A., Rafieepour, S., "Prediction of dead oil viscosity: machine learning vs. classical correlations," Energies, 14, p. 930, 2021.

Hansen, L.S., Pedersen, S., Durdevic, P., "Multi-phase flow metering in offshore oil and gas transportation pipelines: trends and perspectives," Sensors, 19(9), pp. 2184, 2019.

Hodges, A., Alan Turing: The Enigma. Princeton University Press, 2012.

Ilman, M.N., Kusmono. "Analysis of internal corrosion in subsea oil pipeline." Case Studies in Engineering Failure Analysis., 2, pp. 1-8, 2008.

ITOPF, "Fate of marine oil spills," Technical Information Papers, ITOPF Ltd., 2011.

Kalpakjian, S., Schmid, S., Manufacturing Engineering and Technology, Ed. 5, Pearson Prentice Hall, Upper Saddle River, NJ, 2006.

Kanberoglu, B., and Frakes, D., "Improving the accuracy of two-color Multiview (2CMV) advanced geospacial information (AGI) products using unsupervised feature learning and optical flow," Sensors, 19, 2019.

Kays, W.M., Crawford, M.E., Weigand, B., Convective Heat and Mass Transfer, 4th ed., McGraw Hill Education, New York, 2018.

Kerf, T.D., Gladines, J., Sels, S., Vanlanduit, S., "Oil spill detection using machine learning and infrared images," Remote Sensing, 12, 4090, 2020.

Kumar, R., Banerjee, S., Banik, A., Bandyopadhyay, T.K., and Naiya, T.K., "Simulation of single phase non-Newtonian flow characteristics of heavy crude oil through horizontal pipelines," Petro. Sci. and Tech., 35.6, pp. 615-624, 2017.

Lauerman, V, "Global oil demand to disappoint." Petroleum Economist. Oct 7, 2019. Accessed Mar 29, 2019. https://www.petroleum-economist.com/articles/markets/trends/ 2019/global-oil-demand- to-disappoint.

Levy, J.K., and Gopalakrishnan, C., "Promoting Ecological Sustainability and Community Resilience in the US Gulf Coast after the 2010 Deepwater Horizon oil spill," J. Nat. Resources Policy Review, 2(3), pp. 297-315, 2010.

Lyons, William, Working Guide to Petroleum and Natural Gas Production Engineering, Elsevier Publishing, Burlington, Massachusetts, 2010.

Li, X., Chen, G., Zhu, H., "Quantitative risk analysis on leakage failure of submarine oil and gas pipelines using Bayesian network," Proc. Safety and Env. Prot., 103 (2016), pp. 163-173.

Mandal, S.K., Chan, F.T.S., Tiwari, M.K., “Leak detection of pipeline: an integrated approach of rough set theory and artificial bee colony trained SVM,” Expert Systems with Applications, 39, pp. 3071-3080, 2012.

Manning, M. “Offshore oil production in deepwater and ultra-deepwater is increasing.” U.S. Energy Information Administration. Oct 28, 2016. Accessed Mar 29, 2019. https://www.eia.gov/todayinenergy/detail.php?id=28552.

McDonald, K.A., Cosham, A. “Best practice for the assessment of defects in pipelines – gouges and dents.” Engineering Failure Analysis., 12, pp. 720–745, 2005.

McKinsey, “Global Oil Supply and Demand Outlook 2019.” Energy Insights. Accessed Mar 29, 2019. https://www.mckinsey.com/solutions/energy-insights/global-oil-supply-demand-outlook-to-2035/~/media/231FB01E4937431B8BA070CC55AA572E.ashx.

Miller, I, Freund, J.E., Probability and Statistics for Engineers, Prentice-Hall, Inc., Endglewood Cliffs, NJ, 1965.

Mitchell, T., Machine learning, McGraw-Hill Publishers, 1997.

Munson, B., Rothmayer, A., Okiishi, T., Huebsch, W., Fundamentals of Fluid Mechanics, Ed 7., Wiley Publishing, 2012.

Obanijesu, E.O., Omidiora, E.O., “Artificial neural network's prediction of wax deposition potential of nigerian crude oil for pipeline safety,” Petroleum Science and Technology, 26(16), 1977-1991, 2008.

Otchere, D.A., Ganat, T.O.A., Gholami, R., Syahrir, R., “Application of supervised machine learning paradigms in the prediction of petroleum reservoir properties: comparative analysis of ANN and SVM models,” Journal of Petroleum Science and Engineering, 2020.

Pacella, R., “Extreme Engineering: The Deepest Oil Well.” Popular Science. Feb 24, 2009. Accessed Nov 14, 2019. https://www.popsci.com/scitech/article/2009-02/extreme-engineering-deepest-oil-well/.

Palmer, A.C. and King, R.A. Subsea Pipeline Engineering, Ed. 2. PennWell, Tulsa, Oklahoma, 2008.

Prihtiadi, H., Azwar, A., Djamal, M., “A simple method to determine leakage location in water distribution based on pressure profiles,” AIP Conference Proceedings, 1719, 030045, March 2016.

Rashid, S., Akram, U., Khan, S., “WML: wireless sensor network based machine learning for leakage detection and size estimation,” Procedia Comp. Sci., 63, pp 171-176, 2015.

Regan, T., Beale, C., Inalpolat, M., “Wind turbine blade damage detection using supervised machine learning algorithms,” Journal of Vibration and Acoustics, vol 139, 2017.

Rhodes, A.K., “Brent blend, U.K. North Sea marker crude, assayed,” Oil & Gas Journal, 93(6), 63-64, 1995.

Rosenblatt, F., “The perceptron: a probabilistic model for information storage and organization in the brain,” Psychological Review, 65(6), 1958.

Rostami, H., Azin, R., Dianat, R., “Prediction of undersaturated crude oil density using Gaussian process regression,” Pet. Sci. and Tech., 31, pp. 418-427, 2013.

Rukthong, W., Weerapakkaroon, W., Wongsiriwan, U., Piumsomboon, P., Chalermsinsuwan, B., “Integration of computational fluid dynamics simulation and statistical factoral experimental design of thick-wall crude oil pipeline with heat loss,” Adv. in Eng. Software, 86, pp. 49-54, 2015.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J., "Learning representations by back-propagation errors," Nature, vol 323, 1986.

Russell, S., and Norvig, P., Artificial Intelligence: A Modern Approach, 2nd ed., Prentice Hall, 2003.

Sa'aadah, S.N., Tan, H. "Third party damages of offshore pipeline." J Energy Chall Mech, 1(1)3, 2014.

Samuel, A.L., "Some studies in machine learning using the game of checkers," IBM Journal of Research and Development, 3(3), pp 210-229, 1959.

Shalev-Shwartz, Shai and Ben-David, Shai, Understanding machine learning: from theory to algorithms, Cambridge University Press, 2014.

Sinha, U., Dindoruk, B., Soliman, M., "Machine learning augmented dead oil viscosity model for all oil types," Journal of Petroleum Science and Engineering, 195, 2020.

Srivastava, S.K., Katarki, M. V., and Cherian, V., "Failure analysis of a 30-in subsea oil pipeline," Materials Performance, 47(12), pg. 52, 2008.

Sun, Y., Wenquan, N., "Simulating the effects of structural parameters on the hydraulic performance of venturi tube," Modeling and Simulation in Engineering, vol. 2012, 2012.

Teo, C.L., Lim, K.B., Hong, G.S., and Yeo, M.H.T., "A neural net approach in analyzing photography in PIV," Conference Proceedings 1991 IEEE International Conference on Systems, Man, and Cybernetics, vol. 3, pp. 1535-1538, 1991.

Wood, D., Choubineh, A., "Reliable predictions of oil formation volume factor based on transparent and auditable machine learning approaches," Adv. In Geo-Energy Res., 3(3), pp. 225-241, 2019.

Yang, L., Fu. H., Liang, H., Wang, Y., Han, G., Ling, K., “Detection of pipeline blockage using lab experiment and computational fluid dynamic simulation,” Journal of Petroleum Science and Engineering, 183, 2019.

Yang, Y., Khan F., Thodi, P., Abbassi, R., “Corrosion induced failure analysis of subsea pipelines,” Reliability Eng. And Sys. Safety, 159, pp. 214-222, 2017.

Yuan, S.W., Foundations of Fluid Mechanics, Prentice-Hall Publishing, 1967.

Yun, S., and Zhijun, W., “Aged submarine pipeline failure analysis based on accident tree and fuzzy algorithm,” Forth International Conference on Digital Manufacturing and Automation, 28, 2013.

Zhang, J., “Analysis on the effect of venturi tube structural parameters on fluid flow,” AIP Advances, 7, 065315, 2017.

Zhou, Y., Lin, J., Wang, S., and Zhang, C., “Learning ball-balancing robot through deep reinforcement learning,” 2021 International Conference on Computer, Control and Robotics, 2021.

Zhu, H., Lin, P., and Pan, Q., “A CFD (computational fluid dynamic) simulation for oil leak from damaged submarine pipeline,” Energy, 64, pp. 887-899, 2014).

Xie, Z., Ling, H. Y., Kim, N. H., and van de Panne, M., “ALLSTEPS: Curriculum-driven learning of stepping stone skills,” Eurographic Symposium on Computer Animation 2020, 39(8), 2020.

# APPENDIX A: MACHINE LEARNING PROGRAM CODE

```
#Developed by Daniel Eastvedt in support of a master's thesis, Memorial University of Newfoundland

###THIS FIRST SECTION OF CODE TRAINS THE ML ALGORITHMS BASED ON TRAINING DATA AND TESTS
WITH TEST DATA

#initialize libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
#import seaborn as seabornInstance
from numpy import array, reshape, mean, std, sum, sqrt
from sklearn.model_selection import train_test_split
from sklearn import metrics, linear_model
from scipy.optimize import curve_fit
%matplotlib inline
from scipy.stats import sem, t
from colorama import Fore, Back, Style


#global variables here
predict_thresh = 0.85   #minimum model threshold (in proportion)
r2_thresh = 0.80        #minimum model R^2 threshold accepted for pressure model
flag_length = 5         #number of flag points to trigger alert
confidence = 0.95       #this is the confidence level for models
xs_area = 0.0706858     #constant cross-section area (m^2) of pipe with diam=0.3m


#multivariate fit function with x[0] as point 1 velocity and x[1] as point 1 temperature.
def pressureFunc(x, a, b, c, d):
   return a*(x[0]**2)*(x[1]**b) + c*(x[1]**d)*x[0]


#import raw data from CSV file, assign datasets, and divide into training/test sets at 70:30 ratio
dataset = pd.read_csv('MLTrainingDataFile.csv')    #import CSV spreadsheet file with raw data
vel1_set = dataset['Vel1']                         #assign each column in CSV to a dataset
vel2_set = dataset['Vel2']
veldiff_set = dataset['VelDiff']
temp_set = dataset['Temp1']
press_set = dataset['Press']


#break the datasets into train/test sets
vel_train, vel_test, veldiff_train, veldiff_test, temp_train, temp_test, press_train, press_test = \
   train_test_split(vel1_set, veldiff_set, temp_set, press_set, test_size=0.3)
n_test = len(vel_test)                             #find length of test set
```

```
n_train = len(vel_train)                    #find length of training set


###Generate PRESSURE model to predict delta_p using temperature and velocity at point 1

#form datasets into arrays that can be used by the training function
vel_train_pr = array(vel_train)
vel_train_pr_flat = vel_train_pr.flatten()
temp_train_pr = array(temp_train)
temp_train_pr_flat = temp_train_pr.flatten()
temp_array_pr = array((vel_train_pr_flat, temp_train_pr_flat))


#generate the model using curve_fit function and pressureFunc
p0 = [2.50, 0.100, 30, -1]   #provide initial parameters to fit function to guide least squares
param, fitCov = curve_fit(pressureFunc, temp_array_pr, press_train, p0)


#form a set of predictions of output pressures based on the temp/vel inputs from the model
pr_model_predict =[]
q = 0
for items in vel_train_pr:
   y_hat = (param[0]*(vel_train_pr[q]**2)*(temp_train_pr[q]**param[1])) + \
      (param[2]*(temp_train_pr[q]**param[3])*vel_train_pr[q])
   pr_model_predict.append(y_hat)
   q += 1


#find the R^2 score of prediction compared to actual pressure from training data
pr_model_r2score = metrics.r2_score(press_train, pr_model_predict)


###Test PRESSURE model against the test set to verify model

#calculate confidence intervals of the training data
pr_sum_errs = sum((press_train - pr_model_predict)**2)   #sum of errors
pr_stdev = sqrt(1/(n_train-1) * pr_sum_errs)          #standard deviation
pr_t = t.ppf((1 + confidence)/2, n_train - 1)         #find t-value from statistical table
pr_CI = pr_stdev * pr_t                               #confidence interval


#form arrays for test data
vel_test_pr = array(vel_test)
temp_test_pr = array(temp_test)


#predict pressures for test set velocities and temperatures based on the model
pr_test_predict = []
```

```
j = 0
for items in vel_test_pr:
   test_h = (param[0]*(vel_test_pr[j]**2)*(temp_test_pr[j]**param[1])) + \
         (param[2]*(temp_test_pr[j]**param[3])*vel_test_pr[j])
   pr_test_predict.append(test_h)
   j += 1


#determine how many actual pressures fall within the confidence intervals of the model
lower_CI_pr = []
upper_CI_pr = []
i = 0
p = 0
pr_y = array(press_test)
for items in pr_test_predict:
   pr_lower = pr_test_predict[i] - pr_CI
   lower_CI_pr.append(pr_lower)
   pr_upper = pr_test_predict[i] + pr_CI
   upper_CI_pr.append(pr_upper)
   if pr_y[i] > upper_CI_pr[i]:
      p += 1
   if pr_y[i] < lower_CI_pr[i]:
      p += 1
   i += 1
pr_test_within = (n_test - p) / n_test


#test if the pressure model R^2 score meets the threshold
if pr_model_r2score >= r2_thresh:
   #Print a message giving the pressure model
   print('Pressure Model:')
   print('   delta_p = %0.3f*T1^%0.4f*U1^2 + %0.3f*T1^%0.4f*U1' %(param[0], param[1], param[2],
param[3]))
   print('   Model R^2 score: %0.3f' %(pr_model_r2score))
   print('   Predictive power: %0.2f' %(pr_test_within))
   if pr_test_within < predict_thresh:
      print('Pressure Model predictive power less than %0.0f percent' %((predict_thresh * 100)))
elif pr_model_r2score < r2_thresh:
   print('Pressure model R^2 score below 0.80.')
print(' ')


#make a plot of the pressure data
plt.scatter(vel_train, press_train, color='blue', linewidth=3, label='training data')
plt.scatter(vel_train, pr_model_predict, color='red', linewidth=3, label='model')
plt.scatter(vel_test, press_test, color='purple', linewidth=3, label='test')
plt.title('Pressure Model: Mean Velocity vs Pressure Diff.')
plt.xlabel('U1 (m/s)')
```

```
plt.ylabel('delta_p (kPa)')
plt.legend(loc='upper left')
plt.show()
print(' ')


###Generate VELOCITY model to predict delta_U using delta_p

#form datasets into arrays that can be used by the training function
veldiff_rs = array(veldiff_train)
veldiff_rs = veldiff_rs.reshape(-1, 1)
press_rs = array(press_train)
press_rs = press_rs.reshape(-1, 1)


#generate the velocity model using linear regression function from sklearn
vel_model = linear_model.LinearRegression()
vel_model.fit(press_rs, veldiff_rs)
vel_model_predict = vel_model.predict(press_rs)


###Test VELOCITY model against the test set to verify model

#calculate confidence intervals of the training data
vel_model_predict_flat = vel_model_predict.flatten()
vel_sum_errs = sum((veldiff_train - vel_model_predict_flat)**2)
vel_stdev = sqrt(1/(n_train-1) * vel_sum_errs)
vel_t = t.ppf((1 + confidence)/2, n_train - 1)
vel_CI = vel_stdev * vel_t


#form arrays for test data
press_test_array = array(press_test)
press_test_rs = press_test_array.reshape(-1, 1)


#predict delta_U for test set delta_p based on the model
vel_test_predict = vel_model.predict(press_test_rs)
vel_test_predict_flat = vel_test_predict.flatten()
lower_CI_vel = []
upper_CI_vel = []
vi = 0
vp = 0
vel_y = array(veldiff_test)
for items in press_test_rs:
   vel_lower = vel_test_predict_flat[vi] - vel_CI
   lower_CI_vel.append(vel_lower)
   vel_upper = vel_test_predict_flat[vi] + vel_CI
```

```
    upper_CI_vel.append(vel_upper)
    if vel_y[vi] > upper_CI_vel[vi]:
        vp += 1
    if vel_y[vi] < lower_CI_vel[vi]:
        vp += 1
    vi += 1
vel_test_within = (n_test - vp) / n_test


#test if the velocity model predictive power meets the threshold
if vel_test_within >= predict_thresh:
    #print a message giving the velocity model
    print('Velocity Model:')
    print('   delta_U = %0.9f*delta_p + %0.9f' %(vel_model.coef_[0], vel_model.intercept_))
    #print('   Model R^2 score: %0.3f' %(vel_model_r2score))
    print('   Predictive power: %0.2f' %(vel_test_within))
elif vel_test_within < predict_thresh:
    print('Velocity Model predictive power less than %0.0f percent' %((predict_thresh * 100)))


#make a plot of the kinetic data
low_conf_inter = []
up_conf_inter = []
m = 0
for items in press_rs:
    new_low = vel_model_predict[m] - vel_CI
    new_high = vel_model_predict[m] + vel_CI
    low_conf_inter.append(new_low)
    up_conf_inter.append(new_high)
    m += 1
y_lim_low = 0 - (vel_CI * 4)
y_lim_high = 0 + (vel_CI * 8)

plt.scatter(press_train, veldiff_train, color='blue', linewidth=3, label='training data')
plt.plot(press_train, vel_model_predict, color='red', linewidth=3, label='model')
plt.plot(press_train, low_conf_inter, color='green', linewidth=3, label='confidence intervals')
plt.plot(press_train, up_conf_inter, color='green', linewidth=3)
plt.scatter(press_test, veldiff_test, color='purple', linewidth=3, label='test')
plt.title('Velocity Model: Pressure Diff. vs Velocity Diff.')
plt.xlabel('delta_p (kPa)')
plt.ylabel('delta_U (m/s)')
plt.ylim(y_lim_low, y_lim_high)
plt.legend(loc='upper left')
plt.show()


####THIS SECTION OF CODE TAKES NEW DATA POINTS AFTER TRAINING AND DETERMINES IF THEY
CONFORM TO THE MODELS.
```

```
#Control limits for velocity model
vel_upper_CL = 0.05                #upper control limit for velocity model in process monitoring
vel_lower_CL = -0.05               #lower control limit for velocity model
pr_upper_control_interval = 3*pr_stdev  #upper control interval for pressure model
pr_lower_control_interval = 3*pr_stdev  #lower control interval for pressure model


#assign the fault type for the purpose of process monitoring
fault_type = ''
print('Type a fault/condition option:')
print('   A for 20% Leak at 0 Pa')
print('   B for 20% Leak at -500 Pa')
print('   C for 20% Leak at +500 Pa')
print('   D for 10% Leak at 0 Pa')
print('   E for 50% Restriction')
print('   F for High Viscosity')
print('   G for Low Viscosity')
fault_type = input('>').lower()
if fault_type == 'a':
   new_dataset = pd.read_csv('FaultData20Leak0Pa.csv')    #import CSV spreadsheet file with 20% leak, 0
Pa
elif fault_type == 'b':
   new_dataset = pd.read_csv('FaultData20Leak-500Pa.csv') #import CSV spreadsheet file with 20% leak,
-500 Pa
elif fault_type == 'c':
   new_dataset = pd.read_csv('FaultData20Leak+500Pa.csv') #import CSV spreadsheet file with 20%
leak, +500 Pa
elif fault_type == 'd':
   new_dataset = pd.read_csv('FaultData10Leak.csv') #import CSV spreadsheet file with 10% Leak
elif fault_type == 'e':
   new_dataset = pd.read_csv('FaultData50Restriction.csv') #import CSV spreadsheet file with 50%
restriction
elif fault_type == 'f':
   new_dataset = pd.read_csv('HigherViscosity.csv') #import CSV spreadsheet file with 5x viscosity
elif fault_type == 'g':
   new_dataset = pd.read_csv('LowerViscosity.csv') #import CSV spreadsheet file with 0.5x viscosity
new_vel1 = new_dataset['Vel1']
new_vel2 = new_dataset['Vel2']
new_veldiff = new_dataset['VelDiff']
new_temp1 = new_dataset['Temp1']
new_press = new_dataset['Press']
pr_model_conform = []
vel_model_conform = []


#test each new observation for conformance to the velocity model
b = 0
```

```
for items in new_vel1:
   h_vel_dif = vel_model.coef_[0]*new_press[b] + vel_model.intercept_
   if new_veldiff[b] > vel_upper_CL:
      vel_model_conform.append('HIGH')
   elif new_veldiff[b] < vel_lower_CL:
      vel_model_conform.append('LOW')
   elif h_vel_dif:
      vel_model_conform.append(True)
   b += 1


#test each new observation for conformance to the pressure model
k = 0
for items in new_vel1:
   h_press_dif = ((param[0])*(new_vel1[k]**2)*(new_temp1[k]**param[1])) + \
         ((param[2])*(new_temp1[k]**param[3])*new_vel1[k])
   if new_press[k] > (h_press_dif + pr_upper_control_interval):
      pr_model_conform.append('HIGH')
   elif new_press[k] < (h_press_dif - pr_lower_control_interval):
      pr_model_conform.append('LOW')
   elif h_press_dif:
      pr_model_conform.append(True)
   k += 1


#prints a list of the new data
print('Item' + '\t' + 'U1' + '\t' 'T1' + '\t' + 'U2' + '\t' + 'delta_p' + '\t' + '\t' + \
    'Vel. Model' + '\t' + 'Press. Model')
i = 0
for items in new_vel1:
   n = i+1
   print(str(n) + '      %1.3f   %1.1f    %1.3f   %1.4f  ' %(new_vel1[i], new_temp1[i], new_vel2[i],
new_press[i])  \
       + '\t' + str(vel_model_conform[i])  + '          ' + str(pr_model_conform[i]))
   i += 1
print(' ')


#check for a new trend in the data
vel_flag_count = 0
pr_high_flag_count = 0
pr_low_flag_count = 0
v = 0
for items in new_vel1:
   if pr_model_conform[v] == 'HIGH':
      pr_high_flag_count += 1
   if pr_model_conform[v] == 'LOW':
      pr_low_flag_count += 1
```

```
    if vel_model_conform[v] != True:
        vel_flag_count += 1
    v += 1

#Print a message of the new trend with relevant graphs
if vel_flag_count >= flag_length:
    print('New trend in VELOCITY model.')
    plt.scatter(new_press, new_veldiff, color='black', label='new data')
    plt.title('Velocity Model: Pressure Diff. vs Velocity Diff.')
    plt.xlabel('delta_p (kPa)')
    plt.ylabel('delta_U (m/s)')
    plt.plot(press_train, vel_model_predict, color='red', linewidth=3, label='model')
    plt.legend(loc='upper left')
    plt.show()
if pr_high_flag_count >= flag_length:
    print('New HIGH trend in PRESSURE model.')
    plt.scatter(new_vel1, new_press, color='black', label='new data')
    plt.title('Pressure Model: Mean Velocity vs Pressure Diff.')
    plt.xlabel('U1 (m/s)')
    plt.ylabel('delta_p (kPa)')
    plt.scatter(vel_train, pr_model_predict, color='red', linewidth=3, label='model')
    plt.legend(loc='upper left')
    plt.show()
if pr_low_flag_count >= flag_length:
    print('New LOW trend in PRESSURE model.')
    plt.scatter(new_vel1, new_press, color='black', label='new data')
    plt.title('Pressure Model: Mean Velocity vs Pressure Diff.')
    plt.xlabel('U1 (m/s)')
    plt.ylabel('delta_p (kPa)')
    plt.scatter(vel_train, pr_model_predict, color='red', linewidth=3, label='model')
    plt.legend(loc='upper left')
    plt.show()
print(' ')


####THIS SECTION USES ENGINEERING INFERENCE TO ATTEMPT TO CATEGORIZE THE NON-
CONFORMANCE

#fit function for fault analysis
def faultFunc(x, a, b, c):
    return a*(x**b) + c


#decision tree for fault characterization


#if the velocity model is OOC it may be due to a leak
if vel_flag_count >= flag_length:
```

```
#create arrays of fault data for analysis
fault_vel_dif = []
fault_press = []
f = 0
for items in new_veldiff:
   if new_press[f] >= 0 and vel_model_conform[f] == 'HIGH':
      fault_vel_dif.append(new_veldiff[f])
      fault_press.append(new_press[f])
   f += 1


#fit the fault data to a power curve in faultFunc
f_param, f_fitCov = curve_fit(faultFunc, fault_press, fault_vel_dif)


#if the power function is positive, and the pressure model is in control, suspect a leak
if f_param[1] > 0:
   print('LOSS IN FLOWRATE FOUND')
   print('LEAK SUSPECTED')
   vel_loss = []
   vfr_loss = []
   bph_loss = []
   norm_inlet_vel = []
   b = 0
   for items in new_veldiff:
      #if vel_model_conform[b] == 'HIGH':
      vel_loss.append(new_veldiff[b])
      vfr = new_veldiff[b]*xs_area
      bph = vfr*22642
      vfr_loss.append(vfr)
      bph_loss.append(bph)
      norm_inlet_vel.append(new_vel1[b])
      b += 1
   plt.scatter(norm_inlet_vel, bph_loss, color='blue', linewidth=3, label='loss')
   plt.title('Volumetric Loss vs Velocity')
   plt.xlabel('U1 (m/s)')
   plt.ylabel('Oil Loss (BPH)')
   plt.show()
   u = 0
   print('Inlet Vel    Barrels lost per hour')
   for items in bph_loss:
      print('%1.3f       %3.0f    ' %(norm_inlet_vel[u], bph_loss[u]))
      u += 1
else:
   print('UNKNOWN CHANGE IN FLOWRATE')
   print('INVESTIGATION REQUIRED')
```

```
#if the pressure is OOC but velocity is not, unexpected pressure loss or gain has occured
elif vel_flag_count < flag_length:
    if pr_high_flag_count >= flag_length:
        print('EXCESS PRESSURE LOSS FOUND')
        print('FLOW RESTRICTION OR VISCOSITY INCREASE SUSPECTED')
    elif pr_low_flag_count >= flag_length:
        print('DECREASED PRESSURE LOSS FOUND')
        print('VISCOSITY DECREASE SUSPECTED')
    pressure_loss = []
    norm_inlet_vel = []
    z = 0
    for items in new_veldiff:
        #if pr_model_conform[z] != True:
        norm_inlet_vel.append(new_vel1[z])
        expected_press = (param[0]*(new_vel1[z]**2)*(new_temp1[z]**param[1])) + (param[2]*\
              (new_temp1[z]**param[3])*new_vel1[z])
        excess_press_loss = new_press[z] - expected_press
        pressure_loss.append(excess_press_loss)
        z += 1
    plt.scatter(norm_inlet_vel, pressure_loss, color='blue', linewidth=3, label='loss')
    plt.title('Change in delta_p vs Velocity')
    plt.xlabel('U1 (m/s)')
    plt.ylabel('New_p-Normal_p (kPa)')
    plt.show()
    r = 0
    print('Inlet Vel    Excess Pressure Gain (kPa)')
    for items in pressure_loss:
        print('%1.3f       %3.2f    ' %(norm_inlet_vel[r], pressure_loss[r]))
        r += 1

else:
    print('Unknown error occured.')
```