Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Emefon Dan

Failure Analysis of Final Element of Safety Instrumented Systems Using Condition Data

A Statistical and Stochastic Based Model Approach

Master's thesis in Reliability, Availability, Maintainability and Safety
Supervisor: Jørn Vatn
Co-supervisor: Ewa Maria Laskowska
June 2021

NTNU
Kunnskap for en bedre verden

Emefon Dan

Failure Analysis of Final Element of Safety Instrumented Systems Using Condition Data

A Statistical and Stochastic Based Model Approach

Master's thesis in Reliability, Availability, Maintainability and Safety
Supervisor: Jørn Vatn
Co-supervisor: Ewa Maria Laskowska
June 2021

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU

Kunnskap for en bedre verden

Preface

This master thesis was written during the spring semester 2021 at the Department of Mechanical and Industrial Engineering, Faculty of Engineering, Norwegian University of Science and Technology. The thesis is part of the two-year international masters program Reliability, Availability, Maintainability and Safety (RAMS).

The project is carried out in collaboration with my co-supervisor Ewa Maria Laskowska, a PhD candidate in the RAMS group with research interest within condition based maintenance optimization. The thesis which aims to bridge the gap in the oil and gas industry between the collection of condition data and the utilization of this data for predicting the future condition of the system is a continuation of the specialization project carried out in the autumn semester 2020.

This report is partly based on the theory and methods in TPK4120 Safety and Reliability Analysis, TPK5170 Design and Reliability Analysis of Digitalized Safety Systems and in TPK4450 Data Driven Prognostics and Predictive Maintenance. The reader is therefore assumed to have basic knowledge in reliability theory particularly relating to safety instrumented systems, stochastic process theory, data analysis and statistical modelling.

Trondheim, 2021-06-10

Emefon Dan

Acknowledgment

I would like to express my sincere gratitude to my supervisor, Professor Jørn Vatn, for his guidance throughout the period of writing this thesis, my co-supervisor, Ewa Maria Laskowska for her valuable inputs, suggestions and materials which were of much value to support my writing. I also want to acknowledge Xinheng Liu, a post-doc researcher in the RAMS group at NTNU who provided the code used in part of the analysis carried out in this work. Finally, my deepest gratitude goes to my parents and my aunt for their continual love and support throughout the period of my studies at NTNU.

E.D.

Executive Summary

Safety instrumented systems are important independent protection layers used to protect against hazards and accidents associated with hazardous processes and to mitigate their consequences to humans, machines and the environment. They generally consists of three sub components: the sensors, the logic solvers and the final elements. It is important that these systems can carry out their intended function when required. The shutdown valve is one of the most common final element of the safety instrumented system used in the oil and gas and process industries. They may be part of the process control system responsible for keeping the process parameters within set limits or the process shutdown system responsible for isolating a particular section of the plant or the emergency shutdown system responsible for shutting down the whole plant. These systems operate in the so called low demand mode of operation, that is, demand for the activation of the system is no more than once per year. This implies that the system remains in an idle position for long periods of time until there is a demand. While in an idle position, failures may occur which could prevent the system from carrying out its intended function when the need arises. This failure remains hidden until a demand occurs and the system is activated. To detect hidden failures, the system is regularly tested. Proposing an optimal test interval is no easy task. Cost and safety must be balanced. Too frequent testing may be too expensive and may induce stress to the system speeding up the degradation of the valve. Too long interval between test may increase the risk of the system failing before the next test.

In this thesis, the aim is to analyze condition data collected from different valves and use this data to fit a degradation model for the valves to understand how the valves degrade. This understanding can be used to support decision making in maintenance planning and developing optimal testing strategy. The travel time of the valve is an important parameter that describes the condition of the valve. It is required that the valve can close within a given number of seconds less than the time required to achieve process safety. Longer travel time than what is normal can give an indication of an underlying problems. The linear regression model is used to build a predictive model that predicts the next travel time of the valve based on certain factors that are identified to have effects on the travel time such as the time since last activation or maintenance, the travel time during the last activation and the state of the valve at the last activation. The linear model is then used to standardize degradation paths for the valves and degradation models are fit to the path. The Wiener process and the Gamma process with noise are the chosen models to model the degradation of the valves. The Wiener process is a suitable model for modelling non-monotone increasing degradation. The Gamma process on the hand is popular for modelling monotone increasing degradation. To apply the Gamma process to the modelling, it is assumed that the true degradation is monotone increasing and the non-monotonicity observed in the data is due to error in measurements and white noises. Parameters for the models

are estimated from the data and the results are presented and discussed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Industrial systems have been becoming more complex and expensive with less tolerance for performance degradation, productivity decrease, and safety hazards, such as wind farms, aircraft engines, petrochemical production, and metallurgical production. This leads to an ever increasing requirement on reliability and safety of control systems subjected to faults and failures.Dai and Gao [2013]. In the oil and gas and process industry, where safety is of utmost importance, it is common to install additional layers of protection to mitigate the risk of occurrence of hazards and accidents. Safety instrumented system (SIS) are often installed as independent protection layers to reduce the risk associated with hazardous processes. These systems are often the last layer of protection and kicks in when the basic process control system fails. Failure of this system can often lead to disastrous consequences. For example, the fire and gas (F&G) detection system which activates the fire deluge system to suppress the growth of fire in the event of start of fire.

An SIS generally consists of three subsystem: sensors, logic solvers and final elements Rausand and Høyland [2004]. Shutdown valves (ESVs) are the most commonly used final elements of SIS in the process industries Controls [2017]. These ESVs or safety valves are normally operated in a so-called low demand mode of operation. IEC 61508 :2010. This means that the valves are kept idle in open position for long periods and are expected to close and keep tight in case a process demand should occur. Failures may occur while in open position and may cause the valve to "fail to close" or "close too slowly" in a demand situation Lundteigen and Rausand [2008]. Data from OREDA (Offshore and Onshore Reliability Data),ore [2009], lists that as many as 50% of failures within a SIF can be attributed to the final element.

To reduce the uncertainty surrounding the state of the valve, ESVs are periodically tested with

tests ranging from routine inspections to full functional test and data relevant to describing the state of the valve are collected, analysed and decisions are taken based on the results. There are several performance measures and methods applied to assessing the performance and reliability of SIS in general found in the literature. Some of these methods were reviewed in the autumn semester in the specialization project report. Particularly popular amongst these methods were the markovian models Srivastav et al. [2018], Hafver et al. [2019], Ariful et al. [2019], Srivastav et al. [2020]. The markov models utilized failure rates which is based on historical data to asses the performance of the system. According to Srivastav et al. [2020], in order to have a good predictive model, one needs to use real condition monitoring data. In recent years, there has been some efforts in the oil and gas industry to define, collect and analyze real-time condition data to reveal both safe and dangerous failures related to ESD systems. However, some practical challenges exist for such data-driven approaches. Unlike continuously operated systems, early failures of low-demand systems can be rather difficult to be detected as they are not performing their prime functions during normal operations Zhu et al. [2020]. Pengyu Zhu et al attempts to address this problem by developing a logical data-driven approach to enhance the understanding and detectability of ESD system failures Zhu et al. [2020]. Using condition data such as actuator pressure, stem torque and valve travel percentage, Zhu et al were able to identify and link ESD failures to their root causes on different levels of taxonomy Zhu et al. [2020]. This approach however does not attempt to predict the future condition or failure of the ESD.

1.2 Problem Definition

The problem to be addressed in this thesis is how to incorporate system specific data into performance assessment of SIS in general and to build a predictive / degradation model for the system based on real condition data in particular. The system under study is the shutdown valve. Shutdown valve are critical component of the SIS and failures can most time go unnoticed until a demand occurs. In assessing the performance of SIS, it is common to use data from handbooks and expert judgment. This gives a generalized results for SISs. System specific data on the other hand offer valuable information into the operating conditions of the system and is therefore a valuable input to assessing the performance of the system. It is therefore necessary to be able to use the collected data from the system to assess the performance of the system and make predictions about its future performance.

1.3 Objectives

The main objective of this master thesis is to build a predictive model based on the recorded travel time of shutdown valves to predict the travel time of the valves leading up to the delayed

operation failure mode and also to fit a degradation model to capture the degradation pattern of the shutdown valves based on the predictive model. The work is a continuation of the specialization project which was focused on analysing the travel time data to find correlation between the travel time and factors affecting the travel time. To achieve the main objective, the following tasks are identified:

1. Present the basic concepts of an SIS in general.
2. Present the structure, configuration and working principle of the shutdown valves.
3. Explore various statistical model used for data analysis.
4. Explore various models used in prognostics and health management.
5. Identify factors affecting the travel time of the valve.
6. Build a linear predictive model to predict the closing travel time for shutdown valves based on identified factors.
7. Identify a suitable degradation indicator for the valves.
8. Estimate the parameters of the candidate models selected to model the degradation of the valve.
9. Discuss the result.
10. Identify limitations and challenges to the analyses and propose recommendations for future work.

1.4 Approach

Much of the work done in the master thesis involves practical implementation of statistical methods and models. Review of relevant literature is also carried out to get familiarised with state of the art in the field. The research platform used to access relevant literature include *ORIA* and *Google Scholar*. Selection of relevant literature was done based on certain factors such as keywords, date of publication, number of times the article was cited and so on. Some of the keywords used to search for literature include: maintenance optimization, reliability of SIS, stochastic degradation models. In addition weekly discussions and guidance from my supervisor and inputs, suggestions and relevant materials from my co-supervisor contributed greatly to achieving the outlined objectives. The primary tool used for programming was python. Minitab was also used to carry out the analysis of variance to identify important factors affecting travel time of the valve. The baseline code used for parameter estimation for the gamma process with noise was provided by Xingheng Liu, a post-doc researcher in the RAMS research group.

1.5 Limitations

The thesis focuses on incorporating condition data to assess the future performance of final element of safety instrumented systems using regression analysis and stochastic models. To reduce computational complexity and simplify calculations, an engineering approach is used to introduce covariates into the degradation models. This approach is not verified theoretical. However the result from the parameter estimation shows that the approach is reasonable and works well for the data. This may not be the case with other data sets. The full approach is discussed also. Due to limited time, remaining useful life (RUL) prediction and optimal maintenance/testing strategy is not based on the RUL is not developed for the system.

1.6 Outline

The rest of the report is structured as follows: Chapter 2 presents the basic concepts of SIS including the modes of operation, testings and failure classifications. Chapter 3 presents the configuration, working principle and the failure modes associated with the operations of shutdown valves. Chapter 4, 5, and 6 gives a presentation of the statistical framework and stochastic processes implemented in this work. Chapter 7 presents the condition data used and the data pre-processing techniques applied to prepare the data for analysis. Chapter 8 and 9 presents the analysis carried out and Chapter 10 summarizes and discuses what has been done in the master thesis. Recommendation for future work is also outlined in this chapter.

Chapter 2

Basic Concepts of SIS

This chapter gives a background of the basic concepts within safety instrumented systems. The presentation in this chapter is largely based on Rausand and Høyland [2004] and Rausand [2014]

2.1 Safety-Instrumented Systems

A safety-instrumented system (SIS) is an independent protection layer that is installed to mitigate the risk associated with the operation of a specified hazardous system, which is referred to as the equipment under control (EUC) Rausand and Høyland [2004]. A SIS generally consists of at least three subsystems:

- *Sensor subsystem* - detects a potential danger and produces an appropriate electrical signal that is sent to the logic solver. Examples of sensors are pressure transmitters, level transmitters, temperature gauges, and so on.
- *Logic solver subsystem* - detects the electrical signal and compares with a given threshold. If the threshold is exceeded, sends signal to the final elements. Logic solvers can be computers, programmable electronic controllers (PLCs) or relay circuits.
- *Final elements subsystem* - performs the safety function, bringing the protected system to a safe state. Examples of final elements are shutdown valves, circuit breakers, motors, fans, and so on.

These three systems have to work together to detect a deviation in a process variable (pressure, flow or temperature) exceeding a given threshold and bring the EUC to a safe state. A sketch of a SIS is given in Figure 2.1.

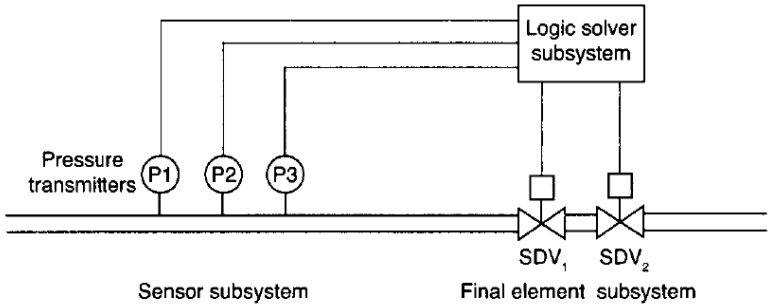


Figure 2.1: Sketch of a simple SIS used as pressure protection system in a pipeline.

Source: Rausand [2014]

The above figure shows three pressure transmitters that monitor the pressure in the pipeline and send this information to the logic solver subsystem. The logic solver compares the received values with predefined set points and, when high pressure occurs, a signal is sent to the two shutdown valves (SDVs) to close the flow in the pipeline. Rausand [2014].

2.2 Safety-Instrumented Function

A safety-instrumented function (SIF) is a function that has been intentionally designed to protect the EUC against a specific demand. The SIF is implemented by a SIS and given a specific safety integrity level (SIL).

2.3 Safety Integrity Levels

Safety integrity level is used as a performance measure for a SIF by the IEC 61508 standard. The standard defines safety integrity as "the probability of a SIS satisfactorily performing the specified SIFs under all the stated conditions within a stated period of time" IEC 61508 :2010. The standard also divides the safety integrity level into four levels, SIL 1, SIL 2, SIL 3 and SIL 4 with SIL 4 being the most reliable and SIL 1 being the least reliable.

2.4 Modes of Operation

IEC 61511 distinguishes between two modes of operation for a SIS:

- demanded mode and
- continuous mode

A SIF in demanded mode is passive in the sense that it does not perform any active function during normal operation but is an add-on to the EUC and is only called upon when something goes wrong, or starts to go wrong.
A SIF operating in continuous mode, on the other hand, plays an active role in the control of the EUC and a hazardous event will occur almost immediately when a dangerous failure of the SIF occurs.
The IEC 61508 standard further splits the demanded mode into two sub-modes:

- Low-demand mode. For this mode, the demand for the SIS is no more than once per year.
- High-demand mode. For this mode, the demand is greater than once per year

2.5 Testing of SIS Functions

Many SISs operates under the low demand mode of operation and are thereby dormant until a specified demand occurs in the EUC. The SIS may fail in the passive position and the failure may remain hidden until the system is activated or tested. The following tests is often carried out to check the status of the SIS:

2.5.1 Diagnostic Self-Testing

The logic solver of modern SISs are often programmable and may carry out diagnostic self-testing during on-line operation. The logic solver may send frequent signals to the detectors and to the actuating items and compare the responses with predefined values. The diagnostic testing can reveal failures of input and output devices, and to an increasing degree, also failures of detectors and actuating items. In many cases the logic solver consists of two or more redundant computers that can carry out diagnostic self-testing of each other. The fraction of failures that can be revealed by diagnostic self-testing is called the diagnostic coverage. The self-testing may be carried out so often that failures are detected almost immediately.

2.5.2 Function Testing

The diagnostic self-testing cannot reveal all failure modes and failure causes, and the various parts of the SIS are therefore often function tested at regular intervals. The objective of a function test is to reveal hidden failures, and to verify that the system is still able to perform the required functions if a process demand should occur.

2.6 Failure Classifications of SIS

The objective of the SIS is to protect the EUC in the event of a deviation in the process parameters. An SIS may perform one or several SIFs. Failure is said to occur when the SIS cannot perform its intended SIF. The following classification based on IEC61508 may be used for an SIS and the SIS subsystems:

1. Dangerous (D). The SIS does not fulfil its required safety-related functions upon demand. These failures may be further split into:

   (a) Dangerous undetected (DU). These dangerous failures prevents activation on demand and are revealed only by testing or when a demands occurs. They are sometimes called dormant failures.

   (b) Dangerous detected (DD). These are dangerous failures that are detected immediately when they occur, for example, by an automatic, built-in self-test.

2. Safe failures (S). These are failures which do not put the SIS in risk of not performing its intended SIF but are nonetheless unwanted. These failures oftentimes lead to spurious trip of the SIS. They may be further split into:

   (a) Safe undetected (SU). Non-dangerous failures that are not detected by automatic self-testing.

   (b) Safe detected (SD). Non-dangerous failures that are detected by automatic self-testing.

Chapter 3

Shutdown Valves

The final element of the SIS (shutdown valve) is looked into in more details in this chapter. The structure, configuration, working principles, possible failure modes and testing strategies are discussed. The presentation is largely based on materials by Rausand and Høyland [2004], Reeson [2006], SAMSON :2017 and Controls [2017]

3.1 Valves

Valves are pressure containing mechanical devices that controls the flow of fluid and pressure within a system Gokilakrishnan et al. [2014]. They are the most common final elements of SIS used in the process control industries Controls [2017]. In the process industry, the safety systems are usually grouped into three categories:

- Process control (PC) system
- Process Shutdown (PSD) system
- Fire and gas detection (FGD) and emergency shutdown (ESD) system

The objective of the process control system is to keep an EUC process within preset limits. Various process control valves are used to control the process, based on signals from temperature, pressure, level and other types of transmitter. When the process deviates from normal values, the process shutdown system is activated and will close down the EUC. The required action for each type of deviation is programmed into the logic solver. The actions may involve activation of alarms, closure of shutdown vales, and opening of relief valves. The process control and process shutdown are local systems that are related to a specific EUC. For some types of process demands that have a potential for a major accident, the ESD system is activated. Several levels of grouping the required ESD actions exist depending on the type of demand and where it occurs.

The top ESD level often involves a total shutdown of the plant and evacuation. In some cases, the same valve is used for both control and shutdown and is connected to all three systems, process control, process shutdown and emergency shutdown as shown in Figure 3.1.

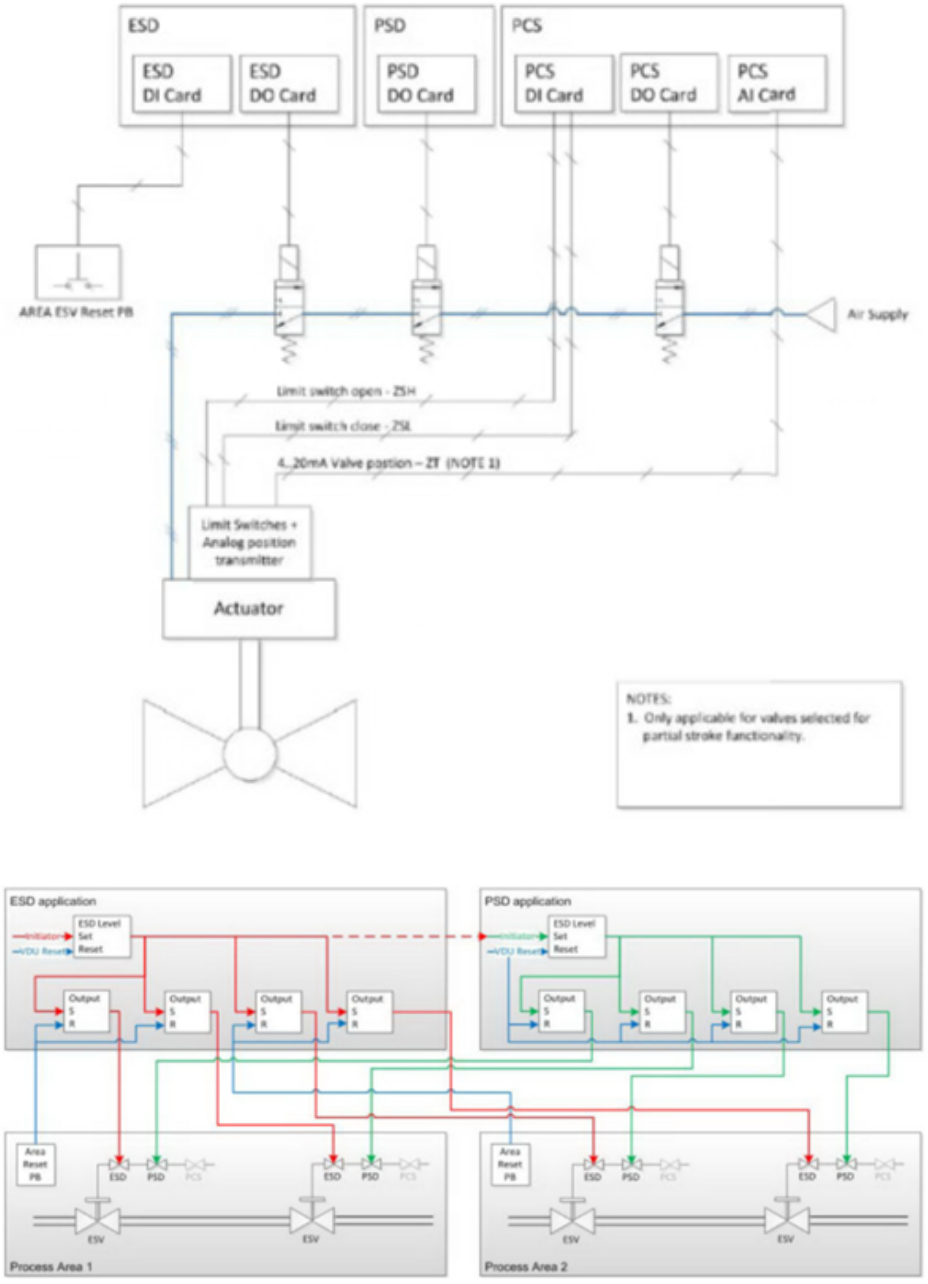


Figure 3.1: Configuration of valve control systems by three solenoids connected to valve actuator

3.2 Valve Configuration

There are many possible physical configurations of shutdown valves depending on their purpose and other factors. A common configuration comprises three main components, Figure 3.2:

- a solenoid (pilot) valve
- an actuator and
- a shutdown valve

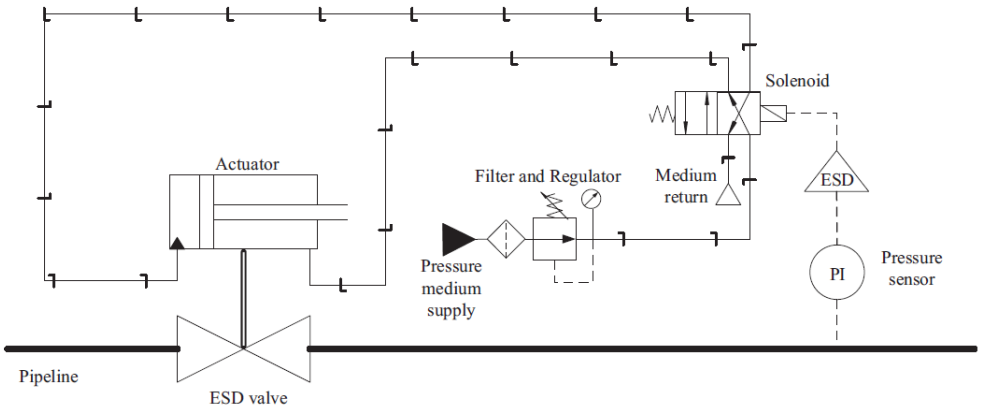


Figure 3.2: Schematic diagram of a typical ESD system.

Source: Zhu et al. [2020]

3.2.1 Solenoid Valve

The solenoid valve is an electrically powered valve which is responsible for the activation of the valve actuator by enabling/disabling air supply to pneumatic actuators. Solenoid valve is a critical element in the pneumatic actuation circuits. In most cases, the solenoid valve always de-energizes to vent air out of the actuator and drive the valve to the safe position with the action of the spring force. In other words, the valve is normally energized. Figure 3.3 shows a solenoid valve connected to an actuator.

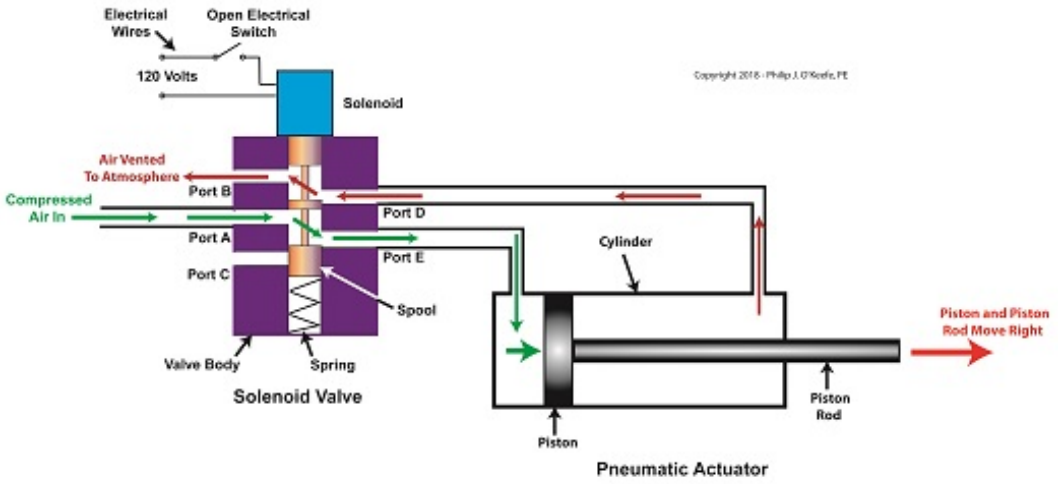


Figure 3.3: Solenoid valve connected with actuator cylinder.

Source: O'Keefe [2018]

3.2.2 Valve actuator

An actuator is basically a control mechanism that is operated by an energy source. This energy which could be hydraulic pressure, pneumatic pressure, or electric current, moves the internal mechanical parts of the actuator to operate the shutdown valve. To obtain fail-safe design of the actuator, so called spring-returns actuators are used. Spring-return actuators have air or liquid supplied to one side of the piston, and the energy to move the mechanisms comes from a spring on the opposite side. This configuration uses pneumatic or hydraulic pressure of the air or liquid to open or close the valve, and a spring affects the opposite motion.

3.2.3 Shutdown valve

The shutdown valve may be a ball, globe or gate valve. The system under considerations uses a ball valve similar to the one shown in Figure 3.4. The valve is made of a perforated and pivoting ball to control flow through it. It is open when the ball's hole is in line with the flow and closed when it is pivoted 90-degrees by the valve stem. This kind of valve enables unrestricted flow of the process fluid when the valve is opened and a tight shut off when in closed position. The valve is moved as a result of the valve stem torque which in connected to the valve actuator and activated by the actuator spring or pressurized air.

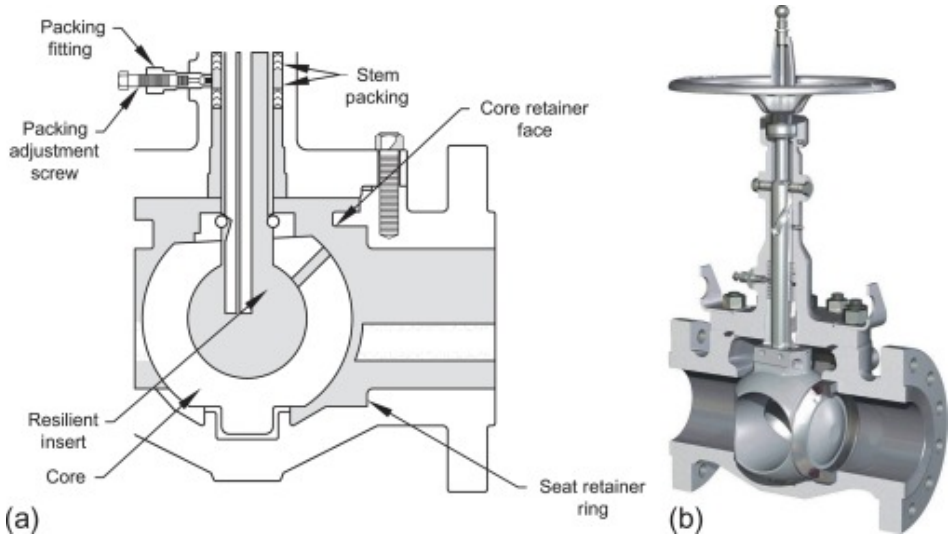


Figure 3.4: Ball valve.

Source: Stewart [2016]

3.3 Working principle

During normal operation the electric power energizes the solenoid valve, keeping it in the position which allows passage of pressurized air from the pressure supply to the valve actuator. This pressurized air on one side of the actuator keeps the actuator's spring compressed keeping the ball valve fully open.

In case of a demand, the logic solver will send information to activate the ESV by cutting of the electrical supply to the solenoid valve. This will cause the movement of solenoid valve opening up the vent port in the actuator to bleed off the the pressurized air. The change of pressure in the actuator cylinder will result in the decompression of the compressed spring. This results in the linear movement of the actuator which is transformed to torque on the ball valve through the valve stem. This leads to the closing of the valve.

3.3.1 Fail safe design

Shutdown valves are normally designed as fail-safe devices. This means that in case of an unpredictable situation such as loss of power, the valve will move to and remain in the safest possible position. In the case of shutdown valves there are usually two possible design options:

- valves which are opened during normal operation of a process plant and required to close in the event of a demand and
- valves which are closed during normal operation and required to open during demand, example pressure relief valves.

The focus is on valves which are opened during normal operation allowing for the flow of the process fluid. In this case the safe position is the closed position of the valve. Since SISs are activated and powered by the means of electric power, in the event of loss of power, the valves need to move to the safe state (closed position) to avoid danger while the power is out.

3.4 Failure Modes

This section describes the most common failure modes associated with shutdown valves based on Rausand [2014].

Fail to close on command (FTC)

This is a DU failure that can only be detected during a function test or during a demand. This failure mode may be caused by a broken spring, blocked return line for the hydraulic fluid, too high friction between the stem and the stem seal, too high friction between the gate and the seats, or by sand, debris, or hydrates in the valve cavity. The failure mode is observed either as a failure event, "the valve does not close," or afterwards, as a failed state, "the valve is open, but should have been in closed position."

Leakage (through the valve) in closed position (LCP)

This is a DU failure mainly caused by corrosion and/or erosion on the gate or the seat. It may also be caused by misalignment between the gate and the seat. The failure mode cannot be observed as a failure event and is only observed as a failed state: "the valve is leaking (more than an acceptable amount)."

Spurious trip (ST)

This is a safe failure which occurs when the valve closes without a closing signal. It is caused by a failure in the hydraulic system or a leakage in the supply line from the control system to the valve. The failure mode is sometimes observed as a failure event, "valve is closing without a signal," or afterwards as a failed state, "the valve is in closed position, but should have been open."

Closing too slowly (CTS)

Also referred to as Delayed Operation, this dangerous undetected failure occurs when the process requires the valve to close within a certain time interval (e.g., 10 seconds) after the ESD signal has been given but fails to do so. Possible causes may be friction between the stem seal

and the stem or a degraded or partly broken spring. If the closing time is monitored, the failure event may be observed. Afterwards, the failure cannot be observed because the valve is found in a normal, closed position after the shutdown action.

Fail to open on command (FTO)

This safe undetected failure occurs when the valve fails to reopen after it is closed. Possible causes may be leakage in the control line, too high friction between the stem seals and the stem, too high friction between the gate and the seats, and sand, debris, or hydrates in the valve cavity. Both the failure event and the failed state can normally be observed.

3.5 Function Testing

Shutdown valves are operated in low demand mode. Thus, most of the dangerous failures remain undetected until tests are carried out or a demand occurs. Data from OREDA (Offshore and Onshore Reliability Data), ore [2009], lists that as many as 50 % of failures within a SIF can be attributed to the final element, Figure 3.5.

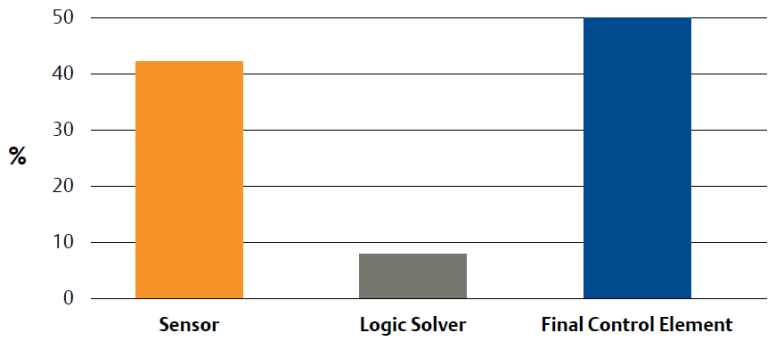


Figure 3.5: Percentage of failure causes of SIS by subsystems

Source: Controls [2017]

Functional testing is required to reveal potential DU failures and involves full stroke operation Lundteigen and Rausand [2008]. The full function test, described in section 2.5, involves sending signal to the valve to close it down completely. This means shutting down the whole process. This is sometimes undesirable as stop in production means loss of revenue. An alternative to the full function test is the partial stroke test. This is done by partially closing the valve, and then returning it to the initial position. The valve movement is so small that the impact on the process flow or pressure is negligible, but the valve movement may still be sufficient to reveal several dangerous failure causes, such as sticking seals and broken signal paths Rausand and Høyland [2004].

Developing an optimal testing strategy often involves striking a balance between cost and achieved safety level. Too frequent testing may be too costly as this involves several stops of the process. Too long interval between test on the hand increases the risk of the valve failing between tests.

Chapter 4

Statistical Framework and Models

This chapter presents the statistical framework and models used in the analysis in this thesis. The presentation is based on materials by Yan et al. [2009], Prokhorov [2020], Pardo [2020] and Frost [2020]

4.1 Regression Analysis Fundamentals

Regression analysis models the relationship between a response variable and one or more predictor variables. It is one of the most commonly used statistical methods in practice. It is often used to understand how changes in the predictor values are associated with changes in the mean of the response. It can also be used to make predictions based on the values of the predictors.

Regression analysis often is the chosen method of analysis when there is not enough information to characterize the distributions of the variables under consideration. Suppose, for instance, that there are reasons to assume that a random variable has a given probability distribution at a fixed value of another variable, so that

$$E(Y|X) = g(X, \beta), \tag{4.1}$$

where β is a set of unknown parameters determining the function, $g(x)$, and that it is required to determine the values of these parameters from results of observations. Depending on the nature of the problem and the aims of the analysis, the results from a set of observations, $[(x_1, y_1), ..., (x_n, y_n)]$, are interpreted in different ways in relation to the variable X. To determine the connection between the variables in the data, it is common to use a model based on simplified assumptions: that X is a controllable variable, whose values are known or can be determined in advance, and the observed value y can be written in the form

$$Y_i = g(X_i, \beta) + \varepsilon_i, \;\; i = 1, ..., n, \tag{4.2}$$

where the variables ε_i characterize the errors in the observed value, which are independent for various observations and identically distributed with mean zero and constant variance.

The purposes of regression analysis often is to:

1. Establish a causal relationship between response variable Y and regressors $X_1, X_2, \ldots, X_n$.
2. Predict y based on a set of values $x_1, x_2, \ldots, x_n$
3. Screen variables $X_1, X_2, \ldots, X_n$ to identify which variables are more important than others to explain the response variable Y so that the causal relationship can be determined more efficiently and accurately.

4.2 Regression Models

There are generally two types of regression models: linear regression models and non-linear regression models. Choice of a regression model often depends on the assumptions about the dependence of the regression function, $g(X, \beta)$ on X and β, but also on the type of response variable and the estimation method.

4.2.1 Linear Regression

Linear regression models are used when there is a suspected linear relationship between the predictor variable(s) and the response variable. In the linear regression model, the function $g(X, \beta)$ is a linear function that maps each value of the independent variable, X, to the response variable, Y, using the vector of parameters, β. In its simplest form, the linear function is given as:

$$Y = X\beta \tag{4.3}$$

with the vector, β, containing the linear regression coefficients. The linear model can be further split into simple linear regression and multiple linear regression.

The simple linear regression is used for modeling the linear relationship between two variables. One of them is the dependent variable, y and the other is the independent variable, x. For example, the simple linear regression can model the relationship between people's weight, y, as a function of their height x. The simple regression model is often written in the following form:

$$y = \beta_0 + \beta_1 x + \varepsilon \tag{4.4}$$

where y is the dependent variable, β_0 is y's intercept, β_1 is the gradient or the slope of the regression line, x is the independent variable, and ε is the random error assumed to be normally

distributed with mean 0 and constant variance, σ^2.

The multiple linear regression is a linear regression model with one dependent variable and more than one independent variables. The multiple linear regression assumes that the response variable is a linear function of the model parameters and there are more than one independent variables in the model. The general form of the multiple linear regression model is as follows:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p + \varepsilon, \tag{4.5}$$

where y is dependent variable, $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ are regression coefficients, and $x_1, x_2, \ldots, x_p$ are independent variables in the model, and ε is the random error. The multiple linear regression usually involves more issues than the simple linear regression such as collinearity, variance inflation and so on.

4.2.2 Nonlinear Regression

Nonlinear regression assumes that the relationship between dependent variable and independent variables is not linear in regression parameters. Example of a nonlinear regression model is the growth model which may be written as:

$$y = \frac{\alpha}{1 + e^{\beta t}} + \varepsilon \tag{4.6}$$

where y is the growth of a particular organism as a function of time t, α and β are model parameters, and ε is the random error. Nonlinear regression model is more complicated than linear regression model in terms of estimation of model parameters, model selection, model diagnosis, variable selection and so on. Nonlinear regression is out of the scope of this work and is not discussed further.

4.3 Linear Regression Framework

4.3.1 Parameter Estimation - Regression Coefficients

The regression parameters or coefficients describes mathematically, the linear relationship between each independent variable and the dependent variable. The sign of a coefficient indicates whether there is a positive or negative correlation between each independent variable and the dependent variable and the value indicates how much the mean of the dependent variable changes with one-unit shift in the independent variable while holding the other variables in the model constant.

The ordinary least squares (OLS) method is commonly used to estimate the regression parameters or coefficients of the linear regression. OLS attempts to find values for the parameters

that minimize the squared error, or difference, between the observed responses, y, and the predicted responses, $X\beta$. In matrix form the squared error for the linear regression model is given as:

$$\varepsilon^T\varepsilon = (Y - X\beta)^T(Y - X\beta) \tag{4.7}$$

Expanding the terms, Equation 4.7 becomes:

$$\varepsilon^T\varepsilon = Y^TY - 2\beta^TX^TY + \beta^TX^TX\beta \tag{4.8}$$

To minimize the square of the errors, the partial derivative of Equation 4.8 is taken with respect to β and the resulting expression is set to equal 0. Solving for β gives the solution for the regression parameters:

$$\beta = [X^TX]^{-1}X^TY \tag{4.9}$$

4.3.2 Test for Significance - p-value

Hypothesis testing involves the process of formulating hypotheses about parameters, and then using data to decide which of the two mutually exclusive hypotheses to believe Hoel [1971]. In linear regression this usually involves the kind of relationship between the dependent variable and the independent variable. The coefficients describes this relationship. However, the relationship could have occurred by chance and it is therefore important to test for the significance.

p-value

The p-value tests the significance of the relationship described by the coefficients for each variable. Often the null hypothesis to be tested is that there is no correlation between the independent variables and the dependent variable. The resulting p-value is then checked against some pre-determined significance level, α, often 0.05. If the p-value for a variable is less than the significance level, then the data provides enough evidence to reject the null hypothesis and conclude that there is a non-zero correlation between the dependent variable and the independent variable. On the other hand, a p-value greater than the significance level indicates that there is insufficient evidence in the data to conclude that there is a non-zero correlation. The null hypothesis can therefore not be rejected.

4.3.3 Goodness of Fit

The goodness of fit of a statistical model describes how well it fits a set of observations. For linear regression, the coefficient of determination or R-squared and standard error of regression

are often used in assessing the goodness-of-fit of the model. Both of these measures give a numeric assessment of how well the model fits the data, however, there are differences between the two statistics.

Standard Error

The standard error of the regression provides the absolute measure of the typical distance that the data points fall from the regression line and is in the unit of the dependent variable. Conveniently, it indicates how wrong the regression model is on average using the units of the response variable with smaller values being better because it indicates that the observations are closer to the fitted line. Mathematically, it is given as

$$\mathrm{S} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right|. \tag{4.10}$$

R-squared

R-squared on the other hand, provides the relative measure of the percentage of the dependent variable's variance that the model explains. Mathematically it can be calculated as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{4.11}$$

R^2 ranges from 0 to 100% with higher values indicating a stronger relationship between the model and the dependent variable. Usually, the larger the R^2, the better the regression model fits the observation. However, large R^2 do not always imply a good model. One could obtain a large R^2 where the model overfits the data. In this case the model may explain most of the variance in the dependent variable for the sample data but may not perform well in predicting the values of the dependent variable for new data points of the independent variable(s).

Adjusted R-squared

R^2 shows how well terms (data points) fit a curve or line. Adjusted R^2 also indicates how well terms fit a curve or line, but adjusts for the number of terms in a model. The formula is given as:

$$R^2_{adj} = 1 - \frac{(1 - R^2)(N-1)}{N - K - 1} \tag{4.12}$$

where N is the number of points in the data sample, k is the number of independent regressors or variables in the model excluding the constant and R^2 is the calculated R-squared.

Both R^2 and the adjusted R^2 gives an idea of how many data points fall within the line of the regression equation. However, there is one main difference between R^2 and the adjusted

R^2. R^2 assumes that every single variable explains the variation in the dependent variable. The adjusted R^2, on the other hand, tells you the percentage of variation explained by only the independent variables that actually affect the dependent variable. Adding more variables to a model always seems like a good way to improve the model. Some of these extra variables may be significant, but one cannot be sure if the significance is just by chance. The adjusted R^2 compensates for this by penalizing those extra variables and is thus, a better indication of goodness of fit than the normal R^2.

The next chapter presents the modelling framework for the stochastic degradation models applied in the thesis

Chapter 5

Prognostics and Health Management

Increasing demand for functionality and quality causes modern systems to be designed with overwhelming complexities. In addition to the increasing complexities of these modern systems, is the high requirement of reliability as failure could result in catastrophic consequences Tsui et al. [2015]. In view of the high impact and extreme costs usually associated with system failures, methods that can predict and prevent such catastrophes have long been investigated. Applications of developed methods are not rare in domains such as electronics-rich systems, aerospace industries, or even public health environment Finch [2009], Bowles [1992]. In general, these methodologies can all be grouped under the framework of prognostics and health management (PHM). Prognostics is used in the industry to manage businesses risks associated with unexpected system failures Sikorska et al. [2011]. According to the international standard for condition monitoring and diagnostics of machines, ISO 13381, prognostics is the "analysis of the symptoms of faults to predict future condition and residual life within design parameters" ISO 13381-1 :2015. Different authors have proposed different definitions for prognostics and according to Sikorska, [Sikorska et al., 2011] these definitions imply that:

- prognostics is, or should be, performed at the component or sub-component level;
- prognostics involves predicting the time progression of a specific failure mode from its incipience to the time of component failure;
- an appreciation of future component operation is required; and
- prognostics is related to but not the same as diagnostics.

Sikorska et al. [2011]

In general, typical workflows in a PHM system can be illustrated, as shown in Figure 5.1. The process generally begins with data collection and condition monitoring. The collected data is then preprocessed and relevant features are extracted using various techniques such as principal component analysis (PCA) and other relevant statistical and engineering knowledge based

methods. Then comes the statistical modelling which involves fitting a mathematical model to the condition data. The model is then used for fault diagnosis, prognosis and condition-based maintenance.

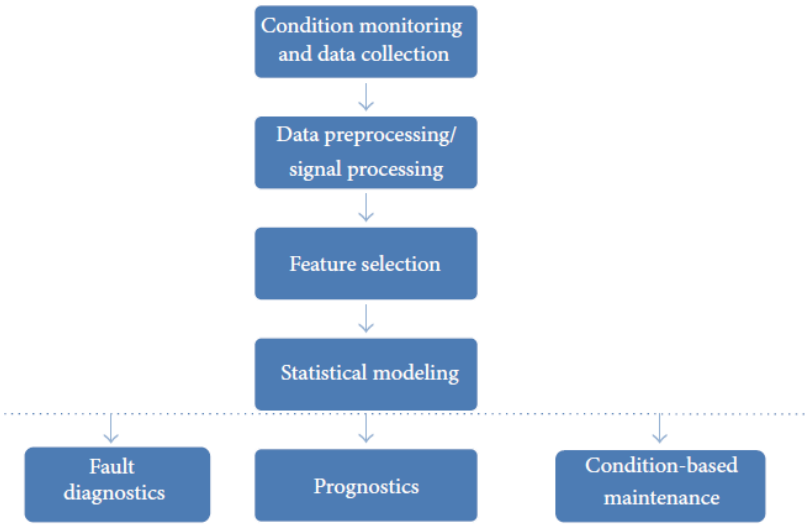


Figure 5.1: An illustration of typical flows of PHM systems.

Source: Tsui et al. [2015]

5.1 Relationship Between Diagnostics and Prognostics

There is a general consensus in the literature that prognostics is related to but not the same thing as diagnostics. According to Sikorska et al. [Sikorska et al., 2011], diagnostics is concerned with identifying and quantifying the damage that has occurred while prognostics involves the prediction of the damage that is yet to occur and relies on the output of diagnostics Sikorska et al. [2011]. The remaining useful life, (RUL), is a typical prediction indicator used within prognostics and is defined as "the length from the current time to the end of the useful life" Si et al. [2011]. The RUL is typically a random variable which must be estimated from the available condition monitoring information. A general definition can be formulated by letting the $RUL(t_j)$ denote a random variable that corresponds to the remaining useful life at time t_j, such that;

$$RUL(t_j) = inf\{h : Y(t_j + h) \in S_L | Y(t_j) < L, Y(s)_{0 \leq s \leq t_j}\} \tag{5.1}$$

where $Y(t_j)$ denotes the condition of the unit at time t_j , which is related to diagnosis. The future health state is denoted by $Y(t_j + h)$, which is the part related to prognosis. Furthermore, S_L denotes a set of unacceptable states representing failure and L represents a fixed threshold limit defining unit or system failure if exceeded.

To properly predict the RUL, Sikorska et al. [Sikorska et al., 2011] suggests looking at a series of questions:

i. Is a component in the degraded state?
ii. Which failure mode has initiated the degradation?
iii. How severe is the degradation?
iv. How quickly is degradation expected to progress from its current state to functional failure?
v. What novel events will change this expected degradation behaviour?
vi. How may other factors (e.g. type of model, measurement noise) affect our RUL prediction?

According to Sikorska et al, Questions (i. - iii.) can be considered as diagnostics questions while the last three are related to prognostics Sikorska et al. [2011].

5.2 Degradation Modelling

Degradation modelling is an important prognostics task. It involves fitting a model to the available condition data to describe the deterioration pattern of the system. Different classification of models used in modelling degradation of systems exists in literature [see for example Sikorska et al. [2011], Tsui et al. [2015]]. The following section presents a some of these models.

5.2.1 Regression-based models

Regression-based methods are commonly used in industry and also in academic fields for lifetime estimation due to their simplicity Si et al. [2011]. Lu and Meeker [1993] present a general nonlinear regression model to characterize the degradation path of a population of units. The general degradation model, given the observed sample degradation $Y(t)$ at time t, can be represented as

$$Y(t) = D(t;\phi,\theta) + \varepsilon(t), \tag{5.2}$$

where $D(t;\phi,\theta)$ is the actual path at time t, ϕ is the fixed effect regression coefficients, common for all units, θ is the random effect representing individual unit characteristics, and $\varepsilon(t)$ is the random error term described by $\mathcal{N}(0,\sigma_\varepsilon)$. Usually, θ and $\varepsilon(t)$ are assumed to be independent of each other.

There have been several applications and extensions to this model in literature such as found in Lu et al. [1997] and Meeker and Escobar [1998]. For a comprehensive review of the applications and extension of the regression-based degradation models in published literature, see Si et al. [2011] and Tsui et al. [2015].

5.2.2 Stochastic-based models

Stochastic-based models usually assume the Markov process Barlow [1965]. A Markov process is a process with the Markov property. Given the value of $X(t)$, the values of $X(s)$, where $s > t$ is independent of the values of $X(u), u < t$. In other words, the conditional distribution of the future states of a system given the present state, is independent of the past states Rausand and Høyland [2004]. Markov processes are usually classified into discrete time Markov processes with finite and countable state space and continuous-time Markov processes with independent increments.

Continuous-time stochastic processes with independent and stationary increments are called Levy processes. A stochastic process has stationary increments if the probability distribution of the increments $X(t+h)-X(t)$ depends only on h for all $t, h \geq 0$ van Noortwijk [2009]. Continuous-time stochastic processes are discussed further in the next chapter.

Chapter 6

Stochastic Degradation Models

Selecting and building an appropriate model not only requires a mathematical understanding of the proposed models but also a familiarization with the system under study. Models, generally are subject to specific assumptions and approximations some of which are mathematical while others relate to practical implementation issues such as the amount of data required to validate and verify a proposed model Sikorska et al. [2011]. Other practical considerations relates to the computational complexity of estimating the design parameters of the model based on observed data.

The following sections presents and overview of some of the popular stochastic models used in literature for degradation modelling of engineering systems and structures.

6.1 Wiener Process

The Wiener process could be considered a type of regression model but with specific properties. A process W is said to be a Wiener process with drift μ and variance σ^2 if:

$$W(t) = \sigma B(t) + \mu t \tag{6.1}$$

where μ is a defined drift parameter, $\sigma > 0$ is a diffusion coefficient and $B(t)$ is a standard Brownian motion. The process W has the following properties:

1. W(0) = 0;
2. W has continuous sample paths;
3. W has independent increments;

4. For any $0 \leq s \leq t$, the random variable W(t) - W(s) has normal distribution with mean $\mu(t-s)$ and variance $\sigma^2(t-s)$

Figure 6.1 shows an illustration of a Wiener process with linear drift with the drift parameter, $\mu = 0.05$ and diffusion coefficient, $\sigma = 0.48$

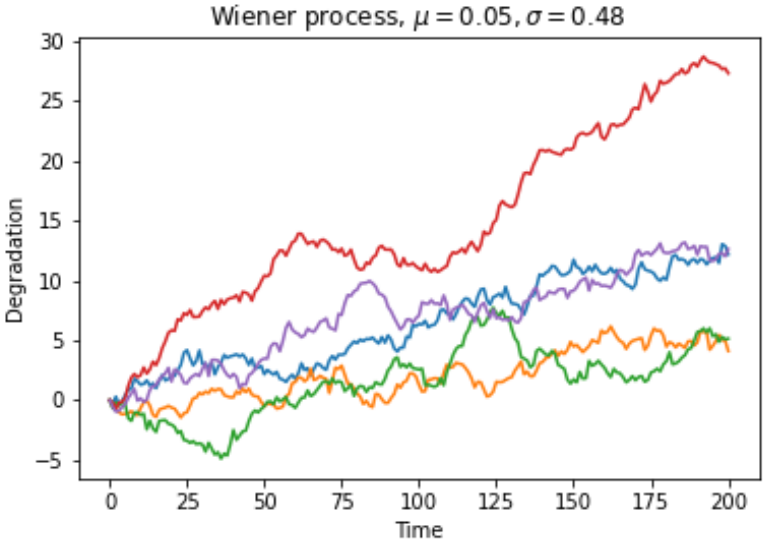


Figure 6.1: Illustration of 5 paths of a wiener process.

Wiener processes for degradation modeling are appropriate for the case that the degradation processes vary bi-directionally over time with Gaussian noises. Modeling degradation processes with Wiener processes has certain mathematical advantages. The most important one is that the distribution of the first passage time can be formulated analytically, known as the inverse Gaussian distribution Si et al. [2011]. The drawback of the Wiener process however, is that it is not suitable to model monotonically increasing degradation.

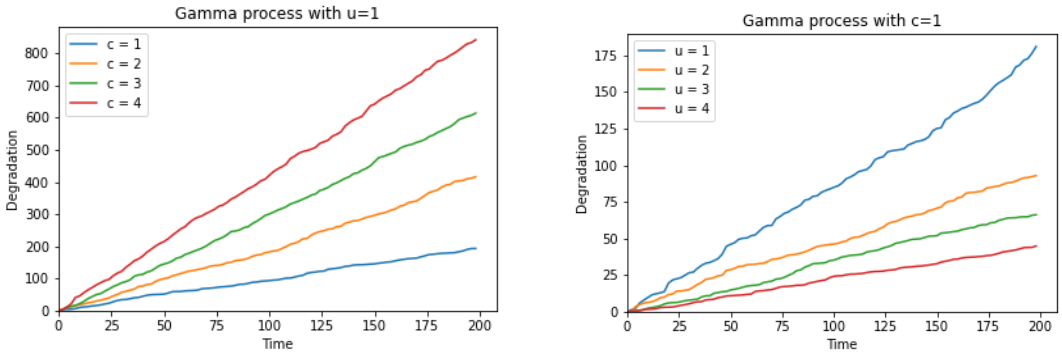
6.2 Gamma Process

The gamma process was originally proposed in 1975 by Abdel-Hameed [1975] in his two page paper called "The gamma wear process", and have since attracted lots of interest and grown in popularity amongst researchers in the area of degradation modelling and remaining useful life estimation. The interest in the gamma process in the literature can be mainly attributed to its suitability to model gradual monotonically accumulating damage and also to the tractability of the required mathematical computations van Noortwijk [2009].
The gamma process is a special case of a Levy process where the increments are independent

and monotonically increasing. Based on this definition, a stochastic process X is said to follow the gamma process with parameters (c, u) if:

1. $X(0) = 0$;
2. X has independent increments;
3. For $0 \leq s < t$, $X(t) - X(s)$ is gamma distributed with parameters $(c_t - c_s, u)$

where c is the shape parameter governing the rate of arrival of jumps and u is the scale parameter which determines the size of the jump in a given interval. Figure 6.2 illustrates the influences of the shape and scale parameter on the gamma process.



Gamma Process: u=1, c=1,2,3,4. Gamma Process: c=1, u=1,2,3,4.

Figure 6.2: Illustration of the effect of shape and scale parameter on the Gamma Process.

The density function of the gamma process is given by:

$$f_{X(t)}(x|c,u) = \frac{u^{c_t}}{\Gamma(c_t)} \cdot x^{c_t-1} \cdot \exp -u \cdot x \tag{6.2}$$

where $0 \leq x < \infty$ and $\Gamma(x) = \int_{z=0}^{\infty} z^{x-1} e^{-z}\, dz$ is the Euler gamma function.
The mean of the gamma process is expressed as:

$$E[X(t)] = \frac{c}{u} \cdot t \tag{6.3}$$

and the variance as:

$$\text{Var}[X(t)] = \frac{c}{u^2} \cdot t \tag{6.4}$$

Figure 6.3 shows different paths of a gamma process and the expected value of the process with shape parameter, c = 0.04 and scale, u = 1.4.

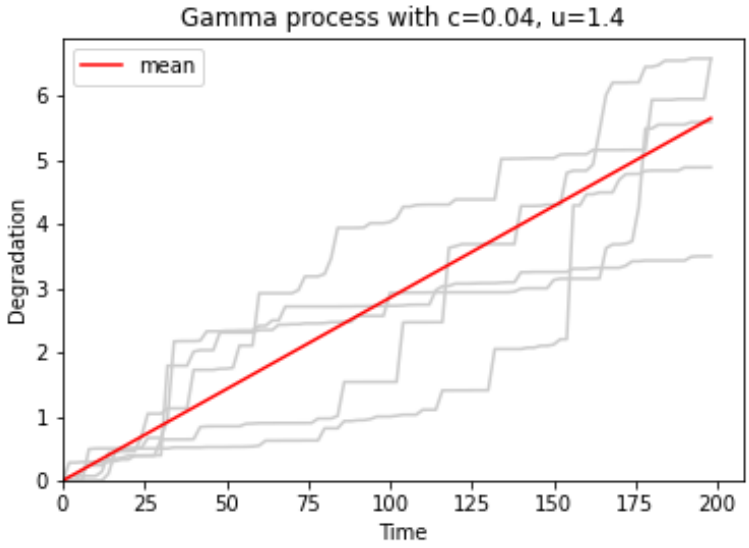


Figure 6.3: Illustration of 5 paths of a gamma process and the mean path.

The homogeneous gamma process has been successfully applied to model gradual degradation phenomena, such as fatigue crack growth [Lawless and Crowder, 2004], thinning due to corrosion [Kallen and van Noortwijk, 2005], corroded steel gates Frangopol et al. [2004], sealing performance of O-rings [Sun et al., 2018]. Zhang et al. [2019] applies the gamma process to model the gradual deterioration of the final elements of the SIS, independent of the damage caused by random demands and high pressure.

6.2.1 Gamma Process with Covariates

In some cases, the degradation process depends on some characteristics of the monitored system such as load, temperature and shock Nabila et al. [2019], the environment Bordes et al. [2010] and individual unit heterogeneity Lawless and Crowder [2004]. This means that the stochastic law that governs the degradation process changes. This has led to the introduction of covariates and explanatory variables by different authors to account for the variations in operating conditions and individual heterogeneity.
A common approach to introduce covariates into the gamma process is to allow the covariates to modify the time scale of the gamma process as in Bagdonavicius and Nikulin [2001]. That is, the degradation at time, t given shape c and scale u, is given by the gamma process:

$$X(t) \sim Ga(c \cdot \int_0^t e^{\beta z_s}\, ds,\ u),\ \forall\ t \geq 0 \tag{6.5}$$

Assuming constant covariates over time:

$$X(t) \sim Ga(c \cdot e^{\beta z_s} t,\ u),\ \ \forall\ t \geq 0 \tag{6.6}$$

where $z^T = (z^{(1)}, z^{(2)}, \ldots, z^{(p)})$ is the vector of covariates and $\beta = (\beta_1, \beta_2, \ldots, \beta_p)$ is a vector of unknown parameters.

To illustrate this process, five degradation paths each were simulated with the assumption that the covariates vector can take one of two possible values corresponding respectively to high stress levels (HSL) and moderate stress levels (MSL), $z \in (1,0), (0,0)$ and $\beta = (0.5, 0)$. This is based on the work done by Nabila et al. [2019] in applying the gamma process to model the deterioration of an actuator exposed to different stress levels. Figure 6.4 shows this illustration.

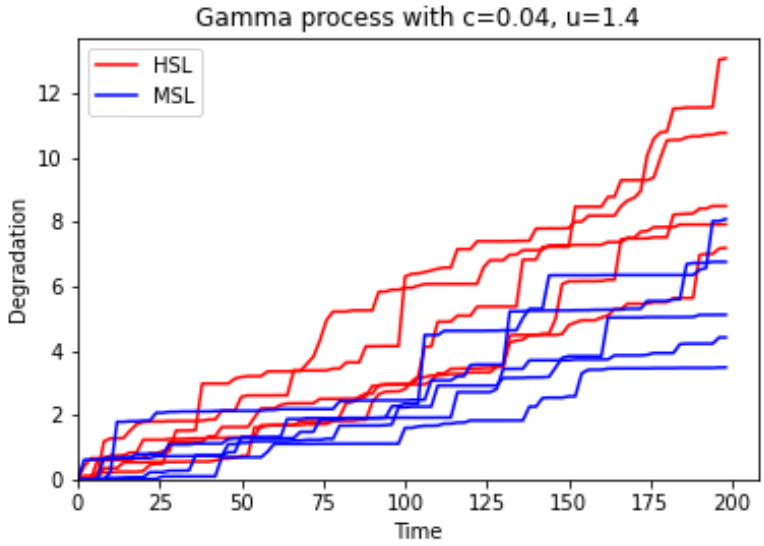


Figure 6.4: Illustration of Degradation paths of a Gamma Process with covariates.

Other alternatives to introducing covariates into the gamma process can be seen in Lawless and Crowder [2004] and Crowder and Lawless [2007]. Here the authors assume that the scale parameter is proportional to a random effect and depends on covariates.

6.2.2 Noisy Gamma Process

The noisy gamma process is an extension of the gamma process incorporating Gaussian noise to account for error in measurements and other sources of white noise. This process is applied to model non-monotonous degradation process. The underlying assumption here is that the true degradation is monotone increasing and follows the gamma process. The observed degradation on the other hand is perturbed by noise which may be due to error in measurements, effects of

minor maintenance actions and so on. The following presentation on the noisy gamma process is based on Le Son et al. [2016]:

Let $X(t)$ be a gamma distributed random variable with a scale parameter u and shape c. The process has the properties listed in section 6.2 with probability density function as given in Equation 6.2.
It is supposed that the deterioration level is not directly observable. Let $Y_n = Y(t_n), n = 1,\ldots,n$, be the degradation indicators observed at inspection times $0 < t_1 < \ldots < t_n$, and $X_n = X(t_n), n = 1,\ldots,n$ is the non-observable state at time t_n modelled by a homogeneous gamma process, thus the relation between X_n and Y_n can be expressed as follows:

$$Y_n = f(X_n, \varepsilon_n) = X_n + \varepsilon_n \tag{6.7}$$

where ε_n are independent Gaussian random variables with standard deviation σ_n and mean equal to zero. It can also be expressed as a function of X_n and Y_n as:

$$\varepsilon_n = g(X_n, Y_n) = Y_n - X_n \tag{6.8}$$

To carry out an efficient prognosis and maintenance planning, it is necessary to approximate the non-observable states of the system from its marginal density. The non-observable states of the system are approximated from the observed state using the conditional density of the non-observable states $X_1,\ldots,X_n$ with respect to the observations $Y_1,\ldots,Y_n$. The conditional density is given as:

$$\mu_{X|Y_1=y_1,\ldots,Y_n=y_n}(x_1,\ldots,x_n) = \frac{\mu_{X,Y}(x_1,\ldots,x_n,y_1,\ldots,y_n)}{\int\ldots\int \mu_{X,Y}(x_1,\ldots,x_n,y_1,\ldots,y_n)dx_1\ldots dx_n} \tag{6.9}$$

with $(x_1,\ldots,x_n)$ and $(y_1,\ldots,y_n)$ a realisation of X and Y, respectively and $\mu_{X,Y}(x_1,\ldots,x_n,y_1,\ldots,y_n)$ is the joint density of the non-observable and observable state. To avoid the difficult calculation of the marginal density of the non-observable state involving many integrals, the Gibbs sampler algorithm is used. The Gibbs sampler algorithm, discussed in section 6.2.2, is a method used to obtains samples from a marginal distributions given the conditional distributions. Through the use of techniques like the Gibbs sampler, we are able to avoid difficult calculations, replacing them instead with a sequence of easier calculations CASELLA and GEORGE [1992].

The Gibbs Sampler

This presentation of the Gibbs sampler algorithm is based on CASELLA and GEORGE [1992]. The Gibbs sampler is a technique for generating random variables from a marginal distribution indirectly, without having to calculate the density. Suppose we are given a joint density

$f(x_1,\ldots,x_n,y_1,\ldots,y_n)$ and the goal is to obtain characteristics of the marginal density

$$f(x) = \int \ldots \int f(x_1,\ldots,x_n,y_1,\ldots,y_n)\,dy_1 \ldots dy_n \tag{6.10}$$

such as the mean and variance. A straight forward approach would be to calculate $f(x)$ and use it to obtain the desired characteristics. However, in some cases the integrations in Equation 6.10 may be difficult to perform, either analytically or numerically. In such cases the Gibbs sampler provides an alternative method for obtaining $f(x)$.

The following illustrates the workings of the Gibbs sampling algorithm: Given a pair of random variables (X, Y), the Gibbs sampler generates a sample from $f(x)$ by sampling instead from the conditional distributions $f(x|y)$ and $f(y|x)$, distributions that are often known in statistical models. This is done by generating a "Gibbs sequence" of random variables

$$Y'_0, X'_0, Y'_1, X'_1, Y'_2, X'_2, \ldots, Y'_k, X'_k.$$

The initial value $Y'_0 = y_0$ is specified, and the rest is obtained iteratively by alternately generating values from

$$\begin{aligned} X'_j &\sim f(x|Y'_j = y_j) \\ Y'_{j+1} &\sim f(y|X'_j = x_j) \end{aligned} \tag{6.11}$$

The distribution of X'_k converges to $f(x)$, the true marginal of X, as $k \to \infty$. Thus for k large enough, the final observation is effectively a sample point from $f(x)$. To obtain iid from $f(x)$, m independent Gibbs sequences of length k can be generated and the final value X'_k from each sequence is used. If k is chosen to be large enough, this yields an approximate iid sample from $f(x)$. Methods for choosing k is discussed in CASELLA and GEORGE [1992].

Chapter 7

System Data and Data Pre-processing

The system under consideration is shutdown valves described in chapter 3. In this chapter, condition data associated with the valves which is used for statistical analysis and degradation modelling is presented.

7.1 Valve activation data

The valves are tested at different intervals and information from the tests are recorded on spreadsheets. One of the collected data is the activation time data. The valve activation data, shown in Figure C.1, contains information about the activations of ninety-two valves. The recorded information include:

- *Time stamp* - the date and time of activation of the valve.
- *Valve* - the name tag of the valve that was activated.
- *Operation* - the kind of operation carried out on the valve, either open or close operation, whether it was a partial stroke (PS) or not and which system initiated the operation, either PCS (Process control system), PSD (Process shutdown) or ESD (Emergency shutdown).
- *Group* - the group the valve belongs to, whether ESD or PPS (Gas Pig launcher).
- *Is To Safe State* - whether the operation puts the valve in a safe state.
- *Description* - brief description of the valve with respect to the EUC.
- *Operation Status* - classification of the response of the valve with respect to the operation carried out either as "OK" or "OK with warning" or "slow".
- *Travel time* - time taken to open or close the valve in seconds.

- *Max travel time* - maximum allowed time in seconds for a particular operation.
- *Have comment* - whether an operation has an associated comment or not.

7.1.1 Raw Data

The raw data contains data for the valves collected over the period of three years between between January 2016 and December 2018. This data is recorded in spreadsheets with each month of each year being stored in a separate file. Some months are split into two spreadsheets depending on the volume of the data resulting in over 40 separate spreadsheet files for the 3 year period. Each spreadsheet contains data for all 92 valves. The tests intervals are highly irregular with no recognizable pattern across the valves. For instance, several strokes may be done on the valve within a 10 minutes interval. Afterwards the valve will remain idle for periods ranging from 1 day to 10 months across the valves.

7.1.2 Data Preprocessing

Data preprocessing is one of the fundamental steps involved in data analytics, independent of the chosen modelling approach. Hence, in order to analyse the system data and build a degradation model for the valves, the raw data needs to be filtered, sorted and down-scaled as much as possible, such that noisy data impacting the performance of the models are reduced Chollet [2018]. The following steps were taken to pre-process the data:

1. **Appending files**. All the separate spreadsheet files were appended together into one file to have all the data in one place for sorting and filtering.
2. **Filtering**. The data was filtered to extract relevant data for the analysis. Majority of the filtering was done in python during the analysis. Before that, the following preliminary filtering was done in the spreadsheet:
    - First the "Operation" column was used as a criteria to clean the data, removing entries that either had no meaning with respect to the proposed analyses or had missing or incomplete information.
        i. All entries having values of "Already in Safe State" or "Already RTN state" were removed. It is understood that this was some sort of check done on the valves to determine its current state. There was no activation of the valves and travel time was recorded as 0.
        ii. Similarly all entries with values of "ESD Close (Interf.)" or "PSD Close (Interf.)" or "PCS Close (Cancelled)" or "PSD Open (Cancelled)" were removed. Although

these entries have meaningful values of travel time, it is understood that operations with the former two values had some sort of interference with the recording device. Operations with the latter two values were cancelled by the operator meaning the activation was incomplete. The recorded travel time cannot then be fully accepted as the true travel time for these operations. To achieve consistency these entries were removed.

- Next the "Operation Status" column was used to remove entries with incomplete information.
    i. All the entries with values, "Error", were removed. This entries had no information under "Timestamp", "Operation", "Travel time" and "Max travel time" columns. It is assumed that these were not real operations and may have been some sort of way to separate entries for different months as these kind of entries were particularly seen just before the first entry of the next month in the data.
    ii. Similarly, entries with values of "Response Missing" were removed. In this case, travel time was not recorded for these operation although they had values for the other columns. It is not fully understood why these values were missing but for consistency in the analysis, these entries were removed.

Chapter 8

Linear Regression Analysis

This chapter presents the regression analysis carried out on the valve activation data. The main aim of the analysis is to identify important factors or variables that can predict the travel time of the valves leading up to delayed operation. This is a continuation of the work done in the autumn semester, therefore, a brief summary of the results from the autumn semester is also presented in this chapter. The following notations are used in this chapter:

Terms

- *Closing operation.* Operation of the valve to close the valve.
- *Test day.* A full calendar day in which a test is carried out. A change in date indicates a separate test day.

Notations

Notations used in this note	
CO_i^k	closing operation i on test day k
$T_{CO_i^k}$	calendar time of closing operation i on test day k
$CT_{CO_i^k}$	Valve travel time for closing operation i on test day k

8.1 Preliminary Hypothesis

In the autumn semester, regression analysis was performed on the valve activation data to check for correlation between the valve travel time for closing operations and some explanatory factors such as the elapsed time since the last operation/maintenance.

The main assumption was that travel time will increase with longer intervals between operations. The proposed linear model was given as:

$$CT_{CO_i^k} = \beta_0 + \beta_1 \cdot (T_{CO_i^k} - T_{CO_{i-1}^k}) + \varepsilon \tag{8.1}$$

where $\varepsilon \sim \mathcal{N}(0, \sigma)$, is random error due to noise and β_0 and β_1 are the regression coefficients.

The result from the regression showed 55 valves having positive trend with 11 being significant while the other 37 valves had negative trend with 2 being significant. Figure 8.1 shows the regression results for all the valves are combined. From this result, it appears that there is a general positive correlation, slope: 5.09e-06, between the travel time and elapsed time since last operation for the valves although from the p-value, 0.3671, this correlation appears not to be significant

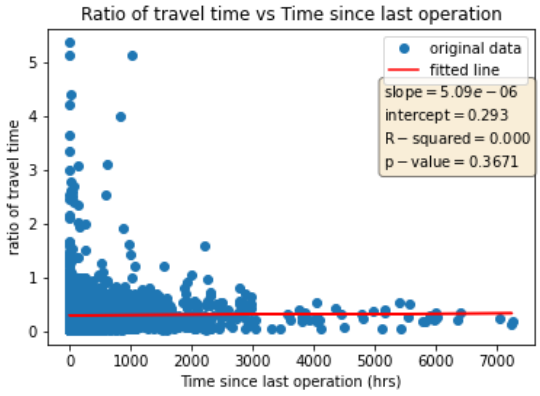


Figure 8.1: Ratio of travel time vs time since last operation

Remark: Response variable used in the regression analysis is the ratio of the travel time to maximum travel time allowed for that operation and not the actual travel time. This allows to normalize the travel time for all the valves so they could be combined for the analysis

8.2 Improved Linear Model

From the observed data, several operations were carried out few seconds apart on a test day. This contributes to having many very short intervals between operations as seen in the scatter plot in Figure 8.1. These subsequent operations offers no new information to the current analysis. As observed by Marvin Rausand, the failure mode "closing too slowly" or "delayed operation" in some cases cannot be observed after the valve has been closed in the first instance be-

cause the movement of the valve has removed the sticking seal problem which normally causes delayed operation Rausand [2014]. The subsequent operations on a test day are therefore assumed to be maintenance actions and filtered out of the analysis. In the improved model only the first, CO_1^k, and last, CO_n^k, closing operation of the test day are considered. The first operation is used as the response variable while the last operation is used to calculate the elapsed time until the next operation which is the first closing operation on the subsequent test day. The improved linear model then becomes:

$$CT_{CO_1^k} = \beta_0 + \beta_1 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) + \varepsilon \tag{8.2}$$

assuming there are n closing operations on a test day k

The scatter plot, Figure 8.2, is similar to the one in the previous section but with improved results.

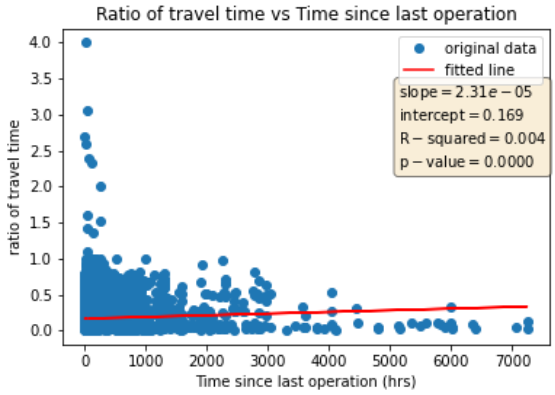


Figure 8.2: Ratio of travel time vs time since last operation result

The p-value shows that there is a significant correlation between travel time and time since last operation of the valves. However, the R-squared indicates that the interval between operations can only explain about 0.4% of the variance present in the travel time.

8.3 Multiple Regression

The plot in Figure 8.3 based on the model in Equation 8.2 shows that the model is not accurate in predicting long travel time. In fact, the model cannot predict travel time for operations where the travel time is greater than the maximum allowed travel time, that is operations with travel time ratio greater than 1 (Maximum predicted travel time is 0.336). This could be as a result of most long travel times occurring after relatively shorter intervals compared to the short travel times as seen in Figure 8.2. An explanation for this could be that thorough maintenance actions, even complete overhaul of parts or the whole valve, which takes longer period, are carried out on

the valve after a slow operation resulting in shorter travel time on the next test day. Conversely, minor or no maintenance actions are carried out on test days when the valve tested OK and thus the valve continues to degrade leading to slower operations on the next test day which is usually after a couple of days for some valves.

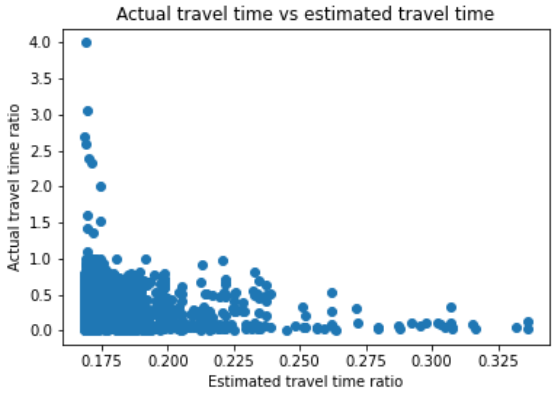


Figure 8.3: Actual vs Predicted Ratio

To study the effect of the interaction between test interval and the state of the valve in the previous test day, an analysis of variance was carried out on a set of valves which showed delayed operation. Valve operations were grouped into two categories:

- OK. operation where the travel time is below the maximum allowed travel time
- NOT OK. operations where the travel time exceeds the maximum allowed travel time

Interval between operations were grouped into three categories:

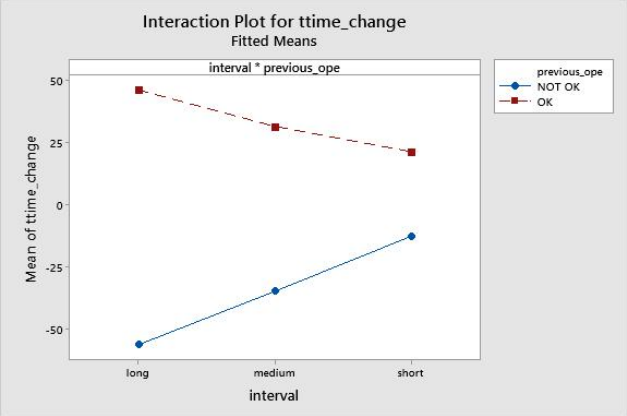
- short. less than 1 month
- medium. 1-3 months
- long. more than 3 months

The null hypothesis, H_0 : the interaction between the state of the valve and the length of the interval until the next operation has no effect on the change in travel time of the valve. In other words the mean change in travel time is the same for all possible combinations of the various categories of the operation status and the length of interval. The results from this analysis is shown in Figure 8.4.

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
interval	2	801	400,7	0,45	0,639
previous_operation	1	27112	27111,6	30,39	0,000
interval*previous_operation	2	7407	3703,6	4,15	0,017
Error	191	170389	892,1		
Total	196	253932			

(a) Analysis of variance.



(b) Interaction between status of previous operation and interval.

Figure 8.4: Analysis of variance

As shown in Figure 8.4b, the travel time increases by about 46 seconds on average for long intervals after the valves were tested OK, 31 seconds for medium interval and 21 seconds for short intervals. For operations where the valves tested NOT OK, the travel time decreases by about 56 seconds on average for long interval, 35 seconds for medium interval and 13 seconds for shorter interval. The p-value in Figure 8.4a, 0.017, for the interaction between the interval and state of valve in previous operation suggests that this result is significant and the null hypothesis can be rejected.

Based on the result from the analysis of variance, the linear model is updated to include the travel time of the previous operation, that is the last operation of the previous test day. The updated model also reflects the interaction between the status of the previous operation and the length of the interval between the operation. To further improve the model, a second step travel time lag was included corresponding to the first travel time of the previous test day. The improved multiple regression model is as shown in Equation 8.3

$$
\begin{aligned}
CT_{CO_1^k} = \beta_0 + \beta_1 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{0,0}(CO_n^{k-1}) + \\
\beta_2 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{0,1}(CO_n^{k-1}) + \beta_3 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{0,2}(CO_n^{k-1}) + \\
\beta_4 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{1,0}(CO_n^{k-1}) + \beta_5 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{1,1}(CO_n^{k-1}) + \\
\beta_6 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{1,2}(CO_n^{k-1}) + \beta_7 \cdot CT_{CO_n^{k-1}} + \beta_8 \cdot CT_{CO_1^{k-1}} + \varepsilon
\end{aligned}
\tag{8.3}
$$

where $I_{i,j}(\mathrm{x}) = 1$ if x is categorized by i and j and
$I_{i,j}(\mathrm{x}) = 0$ otherwise.
i represents the category of the operation status
(0 for OK and 1 for NOT OK) and
j represents the category of the length of the interval
(0 for long, 1 for medium and 2 for short).

The improved model allows the use of the actual closing time as the response variable as opposed to the ratio of travel time to maximum travel time. The previous travel time in the explanatory variable serves as a normalization factor so all valves can be merged to estimate the regression coefficients. The regression coefficients are estimated using ordinary least squares algorithm and the results are shown in Table 8.1. The R-squared and adjusted R-squared are 58.8% and 58.7% respectively. The python codes used in the analysis is explained in Appendix B.

Table 8.1: OLS Regression Results.

Term	Coefficients	p-Value
constant	1.4813	0.000
OK-Long	7.981e-05	0.586
OK-Medium	0.0202	0.325
OK-Short	-0.0002	0.612
Not OK-Long	-0.0157	0.000
Not OK-Medium	-0.0075	0.000
Not OK-Short	-0.0498	0.000
previous test day last travel time	0.3050	0.000
previous test day first travel time	0.5292	0.000

Chapter 9

Degradation Modelling

In this chapter, we attempt to fit degradation models to the valve activation data using the results from the regression analysis.

9.1 Condition monitoring

Condition monitoring is the process of observing certain parameters or variables of a system giving an indication of the current health or status of the system. One of the failure modes associated with shut down valves is delayed operation (DOP) and this failure mode can be observed by monitoring the travel time for closing operation of the valve. The travel time which is measured in seconds gives the time taken from when signal is received to activate the valve to when the valve is in full closed position. For safety reasons, the closing operations are allocated maximum allowed travel time depending on the valve usage and the valve can be said to have failed with failure mode DOP when the travel time exceeds this maximum allowed travel time NOG GL-070 :2018.

In Table 9.1 the maximum allowed travel time, identified usage category based on description, number of valves in the usage category, the average travel time recorded for OK operations and the number of critical faults observed is given. Critical faults here refer to the number of operations where the travel time exceeds the maximum allowed travel time for that operation.

Table 9.1: Maximum allowed travel time and valve usage

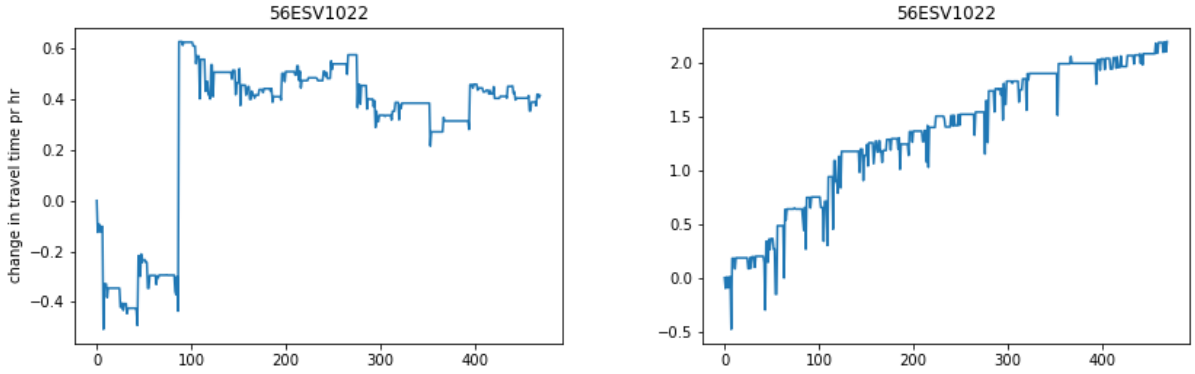
Max allowed travel time (seconds)	Usage	No of valves	Average travel time (OK) (seconds)	Total no of safety operations	No of critical faults
9	Production wing	11	2.34	982	0
	Other usage	1	6.75	82	1
30	Gas lift	11	1.62	719	0
	Chemical injection	10	6.43	640	69
	Upper master valve	11	2.28	1048	0
	Other usage	44	10.05	5378	2
60	Other usage	3	21.65	90	0

9.2 Performance parameter and degradation indicator

Travel time is identified as the performance parameter used to observe DOP failure mode. The health indicator can be given as change in travel time per hour:

$$\Delta CT_k = \frac{CT^k_{CO_1} - CT^{k-1}_{CO_1}}{T^k_{CO_1} - T^{k-1}_{CO_1}} \tag{9.1}$$

Figure 9.1a shows the degradation path based on the given indicator in Equation 9.1 for one of the valves.



(a) Change in travel time per hour.

(b) Standardized path.

Figure 9.1: Degradation path

To account for heterogeneity between the valves, covariates are introduced. It is proposed to use the covariates identified in the regression analysis to account for the different degradation patterns in the valves. In the literature as seen in subsection 6.2.1, the covariates are introduced directly into the degradation model and the coefficients are estimated along with parameters of the selected model. This could prove to be computationally expensive especially dealing with time-dependent covariates as in our case. A pragmatic or engineering approach would be to estimate the coefficients using regression analysis as done in the previous chapter and to use these coefficients to standardize the increments. The standardized increments are given as:

$$\Delta\tilde{CT}_k = \frac{CT_{CO_1^k} - CT_{CO_1^{k-1}}}{\widehat{CT_{CO_1^k}}} \tag{9.2}$$

where $\widehat{CT_{CO_1^k}}$ is the estimated travel time from Equation 8.3. Based on the identified usage in Table 9.1, Equation 8.3 is adjusted to account for the usages. The estimated travel time then becomes:

$$\begin{aligned}\widehat{CT_{CO_1^k}} = {} & \beta_{0,CIV} + \beta_{0,GLV} + \beta_{0,PWV} + \beta_{0,UMV} + \beta_{0,OTH} \\ & \beta_1 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{0,0}(CO_n^{k-1}) + \beta_2 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{0,1}(CO_n^{k-1}) + \\ & \beta_3 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{0,2}(CO_n^{k-1}) + \beta_4 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{1,0}(CO_n^{k-1}) + \\ & \beta_5 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{1,1}(CO_n^{k-1}) + \beta_6 \cdot (T_{CO_1^k} - T_{CO_n^{k-1}}) \cdot I_{1,2}(CO_n^{k-1}) + \\ & \beta_7 \cdot CT_{CO_n^{k-1}} + \beta_8 \cdot CT_{CO_1^{k-1}}\end{aligned} \tag{9.3}$$

Using the model in Equation 9.3 gives a better goodness of fit measure than the model in Equation 8.3. The R-squared and adjusted R-squared are 60.6% and 60.5% respectively (Figure C.3). The now standardized path for the same valve is as shown in Figure 9.1b.

9.3 Degradation models

Based on the standardized path, two models are selected to model the degradation process:

- Wiener process
- Gamma process with noise

The observed deterioration is given as:
$Y_n = \sum_{k=1}^{n} \Delta\tilde{C}T_k$

9.3.1 Wiener Process

The Wiener process obviously is suitable to model the observed degradation because of its suitability to non-monotone increasing degradation. The Wiener process is described in section 6.1. This section describes the estimation of the parameters for the Wiener process.

Parameter Estimation

The method used to estimate the parameters of the Wiener process is the maximum likelihood estimation method.

The maximum likelihood estimation (MLE) is a method of estimating unknown parameters of a probability distribution given samples assumed to have been drawn from that distribution. Suppose we have N independent and identically distributed (iid) variables denoted $Y = Y_1, \ldots, Y_N$, and a corresponding N observations, $y = y_1, \ldots, y_n$, drawn from Y and a joint PDF given by $f(y;\theta)$ where θ is an unknown parameter or vector of parameters of the PDF. The principle of MLE involves finding an estimate of θ such that it maximizes the likelihood of observing those data that were actually observed. Because the elements in Y are independent, the joint distribution may be written as the product of the individual marginal distributions. The resulting likelihood function, Equation 9.4, is then maximized:

$$\Lambda(\theta) = \prod_{i=1}^{N} f(y_i;\theta) \tag{9.4}$$

Due to the computational difficulties of working with products, the log-likelihood function is maximized instead with the knowledge that the log of a function will have its maximum value at the same values of parameters as the function itself. The resulting log-likelihood function, $\ln(\Lambda(\theta)) = \lambda(\theta)$ is given as:

$$\lambda(\theta) = \sum_{i=1}^{N} \ln[f(y_i;\theta)] \tag{9.5}$$

The solution for θ that maximizes the log-likelihood function in Equation 9.5 is called the maximum likelihood estimator (MLE), typically denoted $\hat{\theta}$, the value of which is called the maximum likelihood estimate. Eliason [1993].

For the Wiener process, the observed increments, Δy_i, are normally distributed with parameters $\lambda\Delta t_i$ and $\sigma\Delta t_i$. Barros [2019]. The likelihood function is given as:

$$L(\lambda,\sigma;(\Delta y_1,\ldots,\Delta y_{m_1}),\ldots,(\Delta y_1,\ldots,\Delta y_{m_n})) = \prod_{l=1}^{n}\prod_{j=1}^{m_l} f(\Delta y_{lj};\lambda,\sigma) \tag{9.6}$$

where n is the number of components (valves), m_l, the number of observed increments of the lth valve ($1 \le l \le n$ and $m_l \ge 1$) and Δy_{lj}, the increment of the lth valve at the jth observation ($1 \le j \le m_l$).

The maximum likelihood estimate for the parameters based on the standardized degradation paths of the valves are $\lambda = 0.0354$ and $\sigma = 0.4868$. The codes used in estimating the parameters is given in Appendix B.

9.3.2 Gamma Process with Noise

The Gamma process is popular for modelling monotone increasing degradation. However as discussed in section 6.2, the Gamma process with noise can be applied to model non-monotone degradation. This section describes the estimation of the parameters for the gamma process with noise.

Parameter Estimation

As the true deterioration states are hidden, the maximum likelihood method cannot be directly used. The unknown parameters of the true deterioration obtained through Gibbs sampling are estimated through the Stochastic Expectation-Maximization (SEM) algorithm. Nielsen [2000].

The likelihood function of the gamma distributed deterioration states $X^{(i)} = (x_1^{(i)},\ldots,x_n^{(i)}), i = 1,\ldots,N$ (N:number of components) given the observations $Y^{(i)} = (y_1^{(i)},\ldots,y_n^{(i)}), i = 1,\ldots,N$ is defined as follows:

$$\begin{aligned} L(c,u,\sigma) = \sum_{i=1}^{N}\sum_{j=1}^{n_i} (c(t_j - t_{j-1}) - 1)\,\ln(x_j^{(i)} - x_{j-1}^{(i)}) - u(x_j^{(i)} - x_{j-1}^{(i)}) - \frac{g^2(x_j^{(i)}, y_j^{(i)})}{2\sigma^2} \\ + \ln(|g'(x_j^{(i)}, y_j^{(i)})|) - \ln(\sigma\sqrt{2\pi}) + c(t_j - t_{j-1})\,\ln u - \ln(\Gamma(ct_j - ct_{j-1})) \end{aligned} \tag{9.7}$$

where all the parameters and variables have the usual meaning as defined in subsection 6.2.2.

The random variables $X_j^{(i)}, j = 1,\ldots,n_i$ in the log-likelihood are approximated by the Gibbs sampling (presented in section 6.2.2) and replaced by $Z_j^{(i)}, j = 1,\ldots,n_i$ in the likelihood function to estimate iteratively the parametric inference of the model by the SEM algorithm. The algorithm as described in Le Son et al. [2016] and implemented in the codes in Appendix B is as follows:

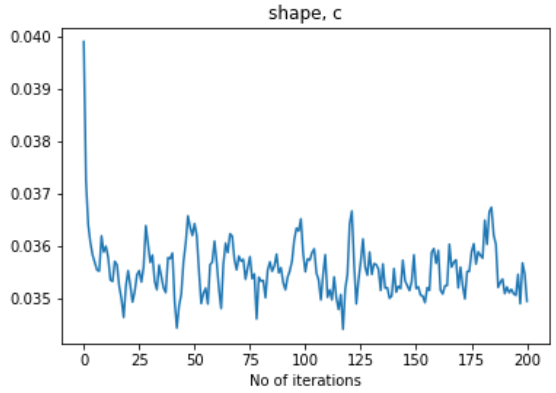
1. The initial parameters (c_0, u_0, σ_0) are choosen and then $Z^1 = (z_{i,1}^1,\ldots,z_{i,r_i}^1)$ simulated by Gibbs sampling. In this work the initial parameters are chosen by maximizing the likelihood function, L, of the monotone path generated using isotonic regression on the observed degradation path.

2. At the q-th iteration, by the samplers vector $Z^q = (z_{i,1}^q,\ldots,z_{i,r_i}^q)$, the following estimation is done:
$$\sigma_q^2 = \frac{\Sigma_{i=1}^N \Sigma_{j=1}^{r_i} g^2(Z_{i,j}^q, Y_{i,j})}{\Sigma_{i=1}^N r_i} \tag{9.8}$$
and $(c_q,\ u_q)$ are obtained by maximizing the likelihood function L.

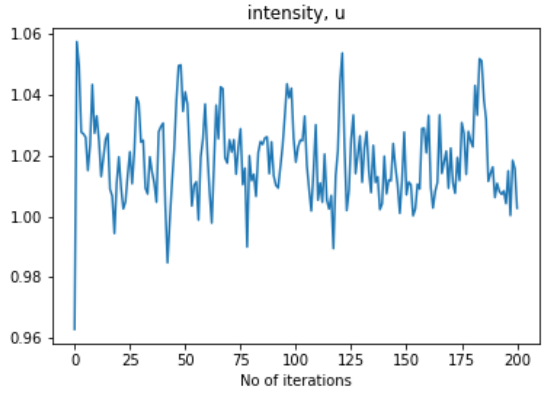
3. At $(q+1)$-th iteration, $(c_q,\ u_q)$ are used to simulate $Z^{q+1} = (z_{i,1}^{q+1},\ldots,z_{i,r_i}^{q+1})$.

4. The parameters of the Gamma process are updated to a sufficient number of iterations Q and a parameter set $(c_q,\ u_q), q = 1,\ldots,Q$ is obtained.

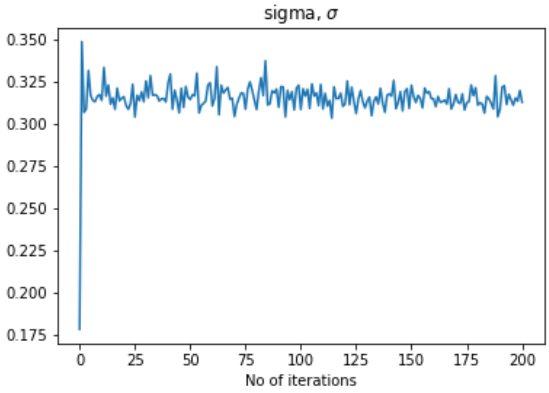
5. The estimated parameters are finally calculated as follows:
$$\hat{c} = \frac{1}{Q}\sum_{q=1}^{Q} c_q, \quad \hat{u} = \frac{1}{Q}\sum_{q=1}^{Q} u_q$$



(a) Shape Parameter.



(b) Intensity or Scale Parameter.



(c) Sigma.

Figure 9.2: Evolution of the Parameters of the Gamma Process Under Stochastic Expectation Maximization Algorithm

Figure 9.2 shows the evolution of the parameters under the stochastic expectation-maximization process. The descriptive statistics of the result from the SEM process for the parameters of the gamma process with noise is given in Table 9.2.

Table 9.2: Parameters of the Gamma Process with Noise.

Statistics	Shape (c)	Scale (u)	Sigma (σ)
Count	201	201	201
Mean	0.03556	1.0187	0.31554
Std	0.00055	0.01359	0.011658
min	0.03440	0.9628	0.17814
25%	0.03441	0.97377	0.24093
50%	0.03443	0.98471	0.30371
75%	0.03451	0.98706	0.30404
max	0.03992	1.05753	0.34915

The codes used for the estimation is given in Appendix B. The original code for estimating the noisy gamma process (section D.1) was written by Xingheng Liu, a post-doc researcher in the RAMS group. The code was modified as explained in section B.4 to be used to estimate the gamma parameters in this thesis. The modification involved adapting the likelihood function of the gamma process from a three-parameter non-homogeneous gamma process to a two-parameter homogeneous gamma process. The Gibbs sampling process, the stochastic expectation maximization algorithm and the likelihood function were also modified to take a joint degradation paths for all the valves as inputs as opposed to one path in the original code.

Chapter 10

Conclusions, Discussion, and Recommendations for Further Work

10.1 Summary and Conclusions

The work done in this thesis was an attempt to bridge the gap between collection of condition data for final element of SIS and the use of this data in predicting the future condition of the system. The failure mode of the SIS under consideration was 'closing too slowly' or 'delayed operation' (DOP). This failure mode of the valve can have undesirable consequences particularly when the response time of the system is required to be less than the process safety time. It can only be detected during activation of the valve and therefore can be classified as a dangerous undetected failure. The main aim of the thesis was to build a model to predict the travel time of the shutdown valve and fit a degradation model to the valve based on the linear model in order to monitor the progression of the DOP failure mode. The thesis has ten chapters. The thesis began in Chapter 1 by giving a brief background and motivation for the work done in the thesis. Chapter 1 also described the objectives, limitations and approach used to accomplish the tasks in this work. In chapter 2, a brief introduction to the main concepts of SIS was presented, concepts such as main components of SIS, modes of operation, testing and so on. Chapter 3 presents the structure, configuration and working principles of shutdown valves, the final element of SIS under study. The most common failure modes and testings of the shutdown valve is also discussed in chapter 3. Statistical framework relevant to the analysis carried out in this thesis is explored in chapter 4 while Chapter 5 and 6 lays the framework for the degradation modelling. Chapter 5 begins by giving a brief introduction of prognostics and health management before looking at the relationship between diagnostics and prognostics and finally the different types of degradation models. Chapter 6 looks into the degradation models used in the analysis, the Wiener process and the Gamma process and its variants. In Chapter 8, a predictive linear model is built for the travel time of the valve. It was identified that the time between operations and

maintenance as well as the previous travel time have significant effects on the travel time of the valve. Based on an analysis of variance carried out on a select group of valves it was further identified that the previous state of the valve has a significant effect on the next travel time. These factors were used to build a predictive model for the valve based on linear regression. Chapter 9 began by identifying different usages for the valves in the data and using this information to update the linear model. The travel time was identified as a suitable performance parameter to observe the DOP failure mode and it was proposed to use the change in travel per hour as a degradation indicator. In order to estimate the parameters for the degradation models, it was necessary to combine the data from all valves and in order to account for heterogeneity between the valves, it was proposed to introduce covariates into the model. The factors identified in the statistical analysis was proposed to be used as the covariates. Estimating the coefficients for the covariates can be difficult and computationally expensive particularly when dealing with time-dependent covariates as is the case in this thesis. A pragmatic or engineering approach is used instead where the coefficients are estimated by regression and these coefficients are used to standardize the increments to obtain the final degradation indicator. The parameters for the Wiener process and the Gamma process with noise are then estimated using maximum likelihood estimation and stochastic expectation maximization methods respectively.

10.2 Discussion

The results from the linear regression analysis is fairly promising with the linear predictive model able to predict the travel time with a prediction accuracy of up to 60%. The most important factor based on the value of the coefficients is the previous travel time. This makes sense as the valves have different ranges of travel time. For instance some valves have a normal travel time of between 1-5s while others can have a normal travel time of up to 25s. Effect of maintenance on the valve can be seen in the coefficients of the variables *NOT OK-long, NOT OK-medium and NOT OK-short* which are -0.0157, -0.0075 and -0.0498 respectively. These variables define the condition of the valve in the previous activation and the length of the interval since that previous activation. The coefficients of these variables which all have significant p-value indicates that the condition of the valve improved after a DOP failure was observed as evident in the reduction in travel time. This is as expected. However, it was also expected that the longer the interval after a DOP failure, the better the condition of the valve, but the results shows that the shorter intervals had the most reducting effect on the travel time. This can be attributed to the fact that in most cases, the DOP failure mode is fixed by the movement of the valve itself and as such cannot be observed after the valve is closed in the first instance Rausand [2014]. It is therefore concluded here that, thorough maintenance which takes longer time is only carried out if the failure mode is still present after several movements of the valve otherwise the valve

is classified as OK. Despite this promising results, there still exists some concerns with utilizing this model for predictions. From an operational point of view, it can be expected that the travel time would increase with the length of the interval since the last activation if the last operation was OK. The variables modelling this are labelled *OK-long, OK-medium and OK-short* and as with the *NOT OK* variables defines the condition of the valve in the previous activation and the length of the interval since that previous activation. These variables, however, do not have significant effects on the valve travel time with the p-values for their coefficients all greater than 0.05. An argument for this could be that, even when the valves tested OK, some minor maintenance actions such as adjustments or lubrication are still carried out on the valve and even the movement of the valve itself improved the valve condition. Another reason for this could be attributed to imbalances in the data between OK operations and NOT-OK operations. There are a total of 8939 operations for all the valves and only 72 of those were delayed operation

In health management and prognostics, it is common practice to combine models to make predictions for systems in what is often called hybrid models. An attempt to fit a degradation model to the travel time data was explored. The degradation models built on the results from the linear regression analysis. The pragmatic/engineering approach proposed in the thesis reduces the computational complexity associated with estimating coefficients for covariates in degradation models. However, this approach of introduction of covariates is yet to be verified from a theoretical point of view. From the resulting degradation path which is non-monotone, the Wiener process with drift is obviously a good candidate model. The estimated parameters shows a general linear drift of 0.0354 with a diffusion of 0.4868. In practice, the deterioration of a system is often monotonous and the non-monotonicity observed in the collected data can be attributed to noise from measurement errors Le Son et al. [2016]. Based on this, the Gamma process with noise is also used to model the degradation path. The estimated parameters for the Gamma process sees the shape and scale parameter and the standard deviation of the Gaussian noise take mean values of 0.036, 1.019 and 0.316 respectively. The diffusion coefficient in the Wiener process and the deviation of the noises for the Gamma process are reasonably low. Based on these results, it may be argued that the engineering approach adopted to introduce covariates into the models was effective in capturing the heterogeneity in the degradation paths amongst the valves.

10.3 Recommendations for Further Work

Due to limited time, RUL predictions and proposal of optimal maintenance policy for the valves based on the RUL were not carried out. Also comparisons between the performance of both models was not done. The following gives possible suggestions on improving or extending the

work carried out.

1. The linear predictive model is fairly promising in terms of its predicting accuracy. Further work needs to be done to improve the model further particularly to improve the ability to predict increase in travel time leading up to delayed operation. More data particularly, data characterizing the delayed operation failure mode should be collected and analyzed. One could also consider other statistical approach such as weighted least squares, partial least squares and principal component analysis to analyze the data.

2. In order to make efficient prognosis and develop optimal maintenance and testing strategy for the valve, RUL predictions based on the degradation models should be developed and a goodness of fit test carried out on the predictions to determine the best model that fits the data. Other factors such as runtime for parameter estimation should also be taken into consideration in choosing the best model.

3. The full theoretical approach of incorporating covariates into the degradation models should be explored. Factors such as size of valves, normal operating speed or torque of the valve amongst others could be considered as time invariant covariates which could be discretized and incorporated into the models.

Appendix A

Acronyms

DOP Delayed operation

ESD Emergency shutdown

ESV Emergency shutdown valve

EUC Equipment under control

FMECA Failure mode effect and criticality analysis

IEC International electrotechnical commission

iid Independent and identically distributed

NO Normally open

PCS Process control system

PFD Probability of failure on demand

PFD$_{avg}$ Probability of failure on demand

PSD Process shutdown

RAMS Reliability, availability, maintainability, and safety

RCA Root cause analysis

SIF Safety instrumented function

SIL Safety integrity level

SIS Safety instrumented system

Appendix B

Python Codes

This chapter contains the python codes used for the analysis in this masters thesis. The program was written in an objected oriented way

B.1 Core

This module contains data structures to store information about the valves including the travel time. The module in python is as follows:

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 26 12:26:09 2021

@author: danem
"""




#%% Required modules

from datetime import datetime

#%% program constans

class Constant:
```

```
    dateFormat = '%Y-%m-%d %H:%M:%S.%f'
    refDate = datetime(1970,1,1)

#%% valve attributes

class Valve:

    def __init__(self, name, category):

        self.name = name
        self.dates = {}
        self.dates_sec = {}
        self.operationStatus = {}
        self.travelTime = {}
        self.category = category
        self.key = 0


    def UpdateValveData(self, date, operationStatus, travelTime):

        self.dates[self.key] = date
        sec = (date - Constant.refDate).total_seconds()
        self.dates_sec[self.key] = sec
        self.operationStatus[self.key] = operationStatus
        self.travelTime[self.key] = travelTime
        self.key += 1

    def SetValveCategory(self, category):
        self.category = category




#%% valve data

class ValveData:
```

```
    def __init__(self):
        self.data = {}

    def AddValve(self, name, category):

        valve = self.LookForValve(name)
        if valve == None:
            valve = Valve(name, category)
            self.data[name] = valve

        return valve

    def LookForValve(self, name):
        return self.data.get(name, None)

    def GetData(self):
        return self.data


#%% valve category

class ValveCategory:

    def __init__(self):
        self.category = dict()
        self.civ = []
        self.glv = []
        self.pwv = []
        self.umv = []
        self.oth = []


    def AddDescription(self, name, category):

        description = self.LookForValve(name)
        if description == None:
            description = category
```

```
            self.category[name] = description
            if description == 'CIV':
                self.civ.append(name)
            elif description == 'GLV':
                self.glv.append(name)
            elif description == 'PWV':
                self.pwv.append(name)
            elif description == 'UMV':
                self.umv.append(name)
            elif description == 'OTH':
                self.oth.append(name)

        return description

    def LookForValve(self, name):
        return self.category.get(name, None)


    def GetCategories(self):
        return self.category
```

B.2 Parser

This module contains functions to import the valve data from the excel sheet into the data structure in the core module.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 26 14:11:45 2021

@author: danem
"""

#%% Required modules

import sys
import openpyxl
```

```
#%%

class XLSParser:

    def ImportValveData(self, valveData, fileName, sheetName,
     ↪  valveCategory):
      try:
        workbook = openpyxl.load_workbook(fileName)
      except:
        sys.stderr.write('Unable to open file "%s"\n' % fileName)
        sys.stderr.flush()
        return 1
      worksheet = workbook[sheetName]
      error = self.DownloadValveData(valveData, worksheet, valveCategory)
      return error

    def DownloadValveData(self, valveData, worksheet, valveCategory):
          row = 1
          #key = 0
          while True:
                row += 1
                cell = worksheet.cell(row=row, column=1)
                timeStamp = cell.value
                #print(type(timeStamp))
                if timeStamp==None:
                    break

                cell = worksheet.cell(row=row, column=2)
                name = cell.value
                if name==None:
                    sys.stderr.write('Row %d, column 2:  not a valve"\n' %
                     ↪  row)
                    sys.stderr.flush()
                    return 1

                if name != 'XXESVTEST':
```

```
                    category = valveCategory.get(name,None)
                    if category==None:
                        sys.stderr.write('category is not specified for
                         ↪  valve %s\n' % name)
                        sys.stderr.flush()
                        return 1


                    valve = valveData.AddValve(name, category)

                    cell = worksheet.cell(row=row, column=7)
                    operationStatus = cell.value
                    if operationStatus==None:
                        sys.stderr.write('Row %d, column 7:  not a valid
                         ↪  entry for operation status"\n' % row)
                        sys.stderr.flush()
                        return 1

                    cell = worksheet.cell(row=row, column=8)
                    travel_time = cell.value
                    if travel_time==None:
                        sys.stderr.write('Row %d, column 8:  not a valid
                         ↪  entry for travel time"\n' % row)
                        sys.stderr.flush()
                        return 1

                    valve.UpdateValveData(timeStamp, operationStatus,
                     ↪  float(travel_time))

            return 0



def ImportValveCategory(self, valveCategory, fileName, sheetName):
    try:
```

```
            workbook = openpyxl.load_workbook(fileName)
        except:
            sys.stderr.write('Unable to open file "%s"\n' % fileName)
            sys.stderr.flush()
            return 1
        worksheet = workbook[sheetName]
        error = self.DownloadValveCategory(valveCategory, worksheet)
        return error

    def DownloadValveCategory(self, valveCategory, worksheet):

        row = 1
        while True:
            row += 1
            cell = worksheet.cell(row=row, column=1)
            valve = cell.value
            if valve==None:
                break

            if valve != '43ESV5021':
                cell = worksheet.cell(row=row, column=3)
                description = cell.value
                if description==None:
                    sys.stderr.write('Row %d, column 3:  not a valve
                     ↪  description"\n' % row)
                    sys.stderr.flush()
                    return 1

                valveCategory.AddDescription(valve,description)

        return 0
```

B.3 Regression Tools

This module contains functions used to extract relevant variables to carry out the regression analysis. It also contain the function used to obtain the standardized increments of the degradation process. However, it does not contain the function to perform the linear regression.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 24 10:19:38 2021

@author: danem
"""

""" Regression Analysis """


#%% Required modules

import numpy as np


#%%

class Regression2:

    def GetExplanatoryVariables(self, vData, vCategory,
     ↪  noOfExplanatoryVariables):

        X = np.zeros((1,noOfExplanatoryVariables))
        y = np.zeros((1,1))



        for valve in vData.keys():

            data = vData[valve]

            category = vCategory[valve]
```

```
            assert len(data.travelTime) == data.key, "length of data is not
             ↪  the same as key %d, %d"%(len(data), data.key)

            x, Y, increments = self.ValveExplanatoryVariable(data, category)

            Y = np.delete(Y, 0)
            Y = Y.reshape(-1,1)

            X = np.vstack((X,x))
            y = np.vstack((y, Y))

        X = np.delete(X,0,axis=0)
        y = np.delete(y,0,axis=0)
        return X, y

    def ValveExplanatoryVariable(self, data, category):

        first_TT = []
        timeStamp_first_TT = []
        op_stat_first = []

        last_TT = []
        timeStamp_last_TT = []
        op_stat_last = []

        #print(str(vData.time_stamp[1])[:10] ==
         ↪  str(vData.time_stamp[3])[:10])

        ttime = data.travelTime
        dates = data.dates
        dates_s = data.dates_sec
        operation_status = data.operationStatus


        for i in range(data.key):
```

```
            if i == 0:
                first_TT.append(ttime[i])
                timeStamp_first_TT.append(dates_s[i])
                op_stat_first.append(operation_status[i])

            elif i == (data.key - 1):
                last_TT.append(ttime[i])
                timeStamp_last_TT.append(dates_s[i])
                op_stat_last.append(operation_status[i])
            else:
                prev = str(dates[i-1])[:10]
                cur = str(dates[i])[:10]

                if prev != cur:
                    first_TT.append(ttime[i])
                    timeStamp_first_TT.append(dates_s[i])
                    op_stat_first.append(operation_status[i])

                    last_TT.append(ttime[i-1])
                    timeStamp_last_TT.append(dates_s[i-1])
                    op_stat_last.append(operation_status[i-1])

        assert len(first_TT) == len(last_TT),"first TT and last TT do not
         ↪  have the same size %d, %d."%(len(first_TT),len(last_TT))

        hours = 60*60

        end = np.asarray(timeStamp_first_TT)[1:]
        start = np.asarray(timeStamp_last_TT)[:-1]

        interval = (end-start)/hours

        x, Y, increments = self.ExtractExplanatoryVariables(interval,
         ↪  first_TT, last_TT, op_stat_last, category)

        return x, Y, increments
```

```
def ExtractExplanatoryVariables(self, interval, first_TT, last_TT,
 ↪  op_stat_last, category):


    z0_civ = np.zeros(len(interval))
    z0_glv = np.zeros(len(interval))
    z0_pwv = np.zeros(len(interval))
    z0_umv = np.zeros(len(interval))
    z0_oth = np.zeros(len(interval))

    if category == 'CIV':
        z0_civ = np.ones(len(interval))
    elif category == 'GLV':
        z0_glv = np.ones(len(interval))
    elif category == 'PWV':
        z0_pwv = np.ones(len(interval))
    elif category == 'UMV':
        z0_umv = np.ones(len(interval))
    elif category == 'OTH':
        z0_oth = np.ones(len(interval))

    z1 = [interval[i] if interval[i] >= 3*730 and op_stat_last[i] !=
     ↪  'Slow' else 0 for i in range(len(interval))]
    z2 = [0 if interval[i] >= 3*730 else( 0 if interval[i] < 730 else(0
     ↪  if op_stat_last[i] == 'Slow' else interval[i])) for i in
     ↪  range(len(interval))]
    z3 = [interval[i] if interval[i] < 730 and op_stat_last[i] != 'Slow'
     ↪  else 0 for i in range(len(interval))]
    z4 = [interval[i] if interval[i] >= 3*730 and op_stat_last[i] ==
     ↪  'Slow' else 0 for i in range(len(interval))]
    z5 = [interval[i] if interval[i] < 3*730 and interval[i] >= 730 and
     ↪  op_stat_last[i] == 'Slow' else 0 for i in range(len(interval))]
    z6 = [interval[i] if interval[i] < 730 and op_stat_last[i] == 'Slow'
     ↪  else 0 for i in range(len(interval))]
```

```
z0_civ = z0_civ.reshape(-1,1)
z0_glv = z0_glv.reshape(-1,1)
z0_pwv = z0_pwv.reshape(-1,1)
z0_umv = z0_umv.reshape(-1,1)
z0_oth = z0_oth.reshape(-1,1)

z1 = np.asarray(z1)
z1 = z1.reshape(-1,1)

z2 = np.asarray(z2)
z2 = z2.reshape(-1,1)

z3 = np.asarray(z3)
z3 = z3.reshape(-1,1)

z4 = np.asarray(z4)
z4 = z4.reshape(-1,1)

z5 = np.asarray(z5)
z5 = z5.reshape(-1,1)

z6 = np.asarray(z6)
z6 = z6.reshape(-1,1)



z7 = np.asarray(last_TT)
z7 = np.delete(z7, -1)
z7 = z7.reshape(-1,1)

z8 = np.asarray(first_TT)
z8 = np.delete(z8, -1)
z8 = z8.reshape(-1,1)



x =
 ↪  np.hstack((z0_civ,z0_glv,z0_pwv,z0_umv,z0_oth,z1,z2,z3,z4,z5,z6,z7,z8))
```

```
        Y = np.asarray(first_TT)


        increments = np.asarray(first_TT[1:]) - np.asarray(last_TT[:-1])

        return x, Y, increments

    def GetStandardizeIncrements(self, incr_dict, valve_set, valve_data,
     ↪  valve_category, regression_result, relevant_variables):

        for valve in valve_set:
            category = valve_category[valve]

            vData = valve_data[valve]
            cov, t_i, increments =
             ↪  self.ValveExplanatoryVariable(vData,category)

            cov_s = cov[:,relevant_variables]
            t_i = np.ravel(t_i)

            E_ti = cov_s @ regression_result.params

            d_t_i = np.diff(t_i)

            #stan = t_i[1:] - E_ti


            stan_incr =  d_t_i/E_ti#[1:]

            #stan_incr = np.diff(stan_incr)

            stan_path = np.cumsum(stan_incr)

            #stan_path = t_i - E_ti

            stan_path = np.insert(stan_path, 0, 0)
```

```
            incr_dict[valve] = stan_path

        return 0
```

B.4 Parameter Estimation

This module contains functions to estimate the parameters for the Wiener process and the Noisy Gamma Process. The code for estimating the Noisy Gamma Process is based on the original code received from Xingheng Liu, a post-doc researcher in the RAMS research department. The original code was written to estimate the parameters for a non-homogeneous gamma process with the shape parameter following the power law $v(t) = at^b$ and scale parameter u, thus estimating three unknown parameters. The original code, given in section D.1, is modified to estimate parameters for a homogeneous gamma process with two unknown parameters. Other modifications include adapting the code to estimate for a joint observation of several degradation paths.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 27 10:12:09 2021

@author: danem
"""

#%% Required Modules

from sklearn.isotonic import IsotonicRegression

import numpy as np
import random as rd
import matplotlib.pyplot as plt
import copy
import math

from scipy.stats import beta
from scipy import optimize
from scipy.special import gamma
from scipy.stats import norm
from scipy.stats import uniform
```

```
from scipy import stats

#%%

class NoisyGamma2Parameters:



    def GenerateNoisyGammab(self, Theta, Timestep):
        c, u, sigma= Theta[0], Theta[1], Theta[2] # u is rate parameter
        Increments = [0]
        Noise = np.array(norm.rvs(loc = 0,scale = sigma, size =
         ↪  len(Timestep)))
        for i in range(1,len(Timestep)):
            Increments.append(rd.gammavariate(c*Timestep[i]-c*Timestep[i-1],
             ↪  1/u))
        GammaPath = np.cumsum(Increments)
        NoisyPath = GammaPath + Noise
        return NoisyPath, GammaPath, Noise

    def kernel1rvs(self, c,d,gam,delt,sigma,Y):
        P=250
        U1 = beta.rvs(a=gam+1,b=delt+1,loc=c,scale=d-c, size=P)
        U2 = uniform.rvs(size=P)
        cpr = U2<=np.exp(-(Y-U1)**2/(2*sigma**2))
        idx = np.argmax(cpr)
        sample = U1[idx]
        return sample


    def kernel2rvs(self, c,bet,gam,sigma,Y):
        P=250
        U1 = stats.gamma.rvs(a=gam+1,loc=c,scale=1/bet, size=P)
        U2 = uniform.rvs(size=P)
        cpr = U2<=np.exp(-(Y-U1)**2/(2*sigma**2))
        idx = np.argmax(cpr)
        sample = U1[idx]
```

```
        return sample


    def GLike(self, Theta, Timestep_dict, Paths_dict):

        if type(Paths_dict) == dict:
            L = []
            keys = Paths_dict.keys()
            for key in keys:
                Paths = Paths_dict[key]
                Timestep = Timestep_dict[key]
                Paths = np.asarray(Paths)
                c, u = Theta[0], Theta[1]
                dX = np.diff(Paths)
                dX[dX<=0]=1e-20 #to avoid numerical errors

                t_start, t_end = Timestep[0:-1], Timestep[1:]
                eta = c*t_end - c*t_start
                el =
                 ↪  (eta*np.log(u)-np.log(gamma(eta))+(eta-1)*np.log(dX)-u*dX)

                l = np.sum(el)

                L.append(l)
            L = np.asarray(L)
        else:
            Paths = Paths_dict
            Timestep = Timestep_dict
            Paths = np.asarray(Paths)
            c, u = Theta[0], Theta[1]
            dX = np.diff(Paths)
            dX[dX<=0]=1e-20 #to avoid numerical errors

            t_start, t_end = Timestep[0:-1], Timestep[1:]
            eta = c*t_end - c*t_start
            L = (eta*np.log(u)-np.log(gamma(eta))+(eta-1)*np.log(dX)-u*dX)
```

```
        return -np.sum(L)



    def GibbsSampling(self, Path_dict, Timestep_dict, Theta, sigma, Qm, Qs,
     ↪ Y0_dict):

        if type(Path_dict) == dict:
            Z_Tab = {}

            keys = Path_dict.keys()

            for key in keys:

                Path = Path_dict[key]
                Timestep = Timestep_dict[key]
                Y0 = Y0_dict[key]

                n = len(Timestep)
                alpha = Theta[0]
                #b = Theta[1]
                bet = Theta[1]
                Z_tab = [Y0]
                for Q in range(1, Qm):
                    z = copy.deepcopy(Z_tab[Q-1])
                    for i in range(n):
                        if i==0:
                            z[i]=0
                        elif i<n-1:
                            if z[i+1]<=z[i-1]: # to avoid numerical errors
                                z[i]=z[i+1]
                            else:
                                c, d = z[i-1], z[i+1]
                                t1, t2, t3 = Timestep[i-1], Timestep[i],
                                 ↪ Timestep[i+1]
                                gam = alpha*(t2-t1)-1
                                delt = alpha*(t3-t2)-1
```

```
                            z[i] =
                             ↪ self.kernel1rvs(c,d,gam,delt,sigma,Path[i])
                    else:
                        c= z[i-1]
                        t1, t2 = Timestep[i-1], Timestep[i]
                        gam = alpha*(t2-t1)-1
                        z[i] = self.kernel2rvs(c,bet,gam,sigma,Path[i])
                Z_tab.append(z)
            Z_Tab[key] = Z_tab[Qs:Qm]

    else:
        Path = Path_dict
        Timestep = Timestep_dict
        Y0 = Y0_dict
        n = len(Timestep)
        alpha = Theta[0]
        #b = Theta[1]
        bet = Theta[1]
        Z_tab = [Y0]
        for Q in range(1, Qm):
            z = copy.deepcopy(Z_tab[Q-1])
            for i in range(n):
                if i==0:
                    z[i]=0
                elif i<n-1:
                    if z[i+1]<=z[i-1]: # to avoid numerical errors
                        z[i]=z[i+1]
                    else:
                        c, d = z[i-1], z[i+1]
                        t1, t2, t3 = Timestep[i-1], Timestep[i],
                         ↪ Timestep[i+1]
                        gam = alpha*(t2-t1)-1
                        delt = alpha*(t3-t2)-1
                        z[i] =
                         ↪ self.kernel1rvs(c,d,gam,delt,sigma,Path[i])
                else:
                    c= z[i-1]
```

```
                    t1, t2 = Timestep[i-1], Timestep[i]
                    gam = alpha*(t2-t1)-1
                    z[i] = self.kernel2rvs(c,bet,gam,sigma,Path[i])
            Z_tab.append(z)
        Z_Tab = Z_tab[Qs:Qm]
    return Z_Tab



def Std_joint(self, path_dict, timestep_dict, y0_dict):



    keys = path_dict.keys()

    chk = path_dict[list(path_dict)[0]]

    if type(chk) == list:

        var = 0
        N = 0

        for key in keys:

            SampledPath = path_dict[key]
            timestep = timestep_dict[key]
            Path = y0_dict[key]

            var += np.sum((SampledPath-Path)**2)
            N += len(SampledPath)*len(Path)

        sigma = np.sqrt(var/N)

    else:
        var = 0
        N = 0
```

```
            for key in keys:

                path = path_dict[key]
                timestep = timestep_dict[key]
                y0 = y0_dict[key]

                var += np.sum((path-y0)**2)

                N += len(timestep)

            sigma = np.sqrt(var/N)

        return sigma

    def EM_Gibbs_us(self, Path_dict, Timestep_dict=None, Qm=200, Qs=50,
     ↪  M=200, x0=None, y0_dict=None):


        if type(Path_dict) == dict and Timestep_dict == None:
            keys = Path_dict.keys()
            Timestep_dict = dict()
            for key in keys:
                path = Path_dict[key]
                timestep = np.asarray(list(range(len(path))))
                Timestep_dict[key] = timestep
        elif Timestep_dict == None and type(Path_dict) != dict:
            Timestep_dict = np.asarray(list(range(len(Path_dict))))


        if x0 == None and y0_dict==None:
            x0, y0_dict = self.IsoX0(Path_dict, Timestep_dict)

        if type(Path_dict) == dict:



            #keys = Path_dict.keys()
```

```
            # SamplePaths = {}

            # for key in keys:
            #     Path = Path_dict[key]
            #     Timestep  = Timestep_dict[key]
            #     y0 = y0_dict[key]

            sigma0 = self.Std_joint(Path_dict, Timestep_dict, y0_dict)
            Tab_sigma=[sigma0]
            Tab_theta=[x0]
            for i in range(M):
                SampledPath = self.GibbsSampling(Path_dict, Timestep_dict,
                 ↪ Tab_theta[-1], Tab_sigma[-1], Qm, Qs, y0_dict)
                #var = np.sum((SampledPath-Path)**2)
                NewSigma = self.Std_joint(SampledPath, Timestep_dict,
                 ↪ Path_dict)
                Param_hat = optimize.minimize(self.GLike,
                              x0,
                              args=(Timestep_dict, SampledPath),
                              method='Nelder-Mead')
                Tab_theta.append(Param_hat.x)
                Tab_sigma.append(NewSigma)
                print('i=',i)


        else:
            Path = Path_dict
            Timestep = Timestep_dict
            y0 = y0_dict

            sigma0 = np.sqrt(np.sum((Path-y0)**2)/len(Timestep))
            Tab_sigma=[sigma0]
            Tab_theta=[x0]
            for i in range(M):
                SampledPath = self.GibbsSampling(Path, Timestep,
                 ↪ Tab_theta[-1], Tab_sigma[-1], Qm, Qs, y0)
                var = np.sum((SampledPath-Path)**2)
```

```
            NewSigma = np.sqrt(var/(len(SampledPath)*len(Path)))
            Param_hat = optimize.minimize(self.GLike,
                          x0,
                          args=(Timestep, SampledPath),
                          method='Nelder-Mead')
            Tab_theta.append(Param_hat.x)
            Tab_sigma.append(NewSigma)
            print('i=',i)
        return Tab_theta, Tab_sigma



    def IsoX0(self, paths,timeStep):


        if type(paths) == dict:
            FilteredY = {}
            new_path = {}
            new_time = {}


            keys = paths.keys()
            for key in keys:
                Path = paths[key]
                Timestep = timeStep[key]
                iso_reg = IsotonicRegression(y_min=0).fit(Timestep, Path)
                y_filt = iso_reg.predict(Timestep)
                FilteredY[key] = y_filt
                unique =  np.unique(y_filt, return_index=True)
                new_path[key] = unique[0]
                new_time[key] = Timestep[unique[1]]
        else:
            Path = paths
            Timestep = timeStep
            iso_reg = IsotonicRegression(y_min=0).fit(Timestep, Path)
            y_filt = iso_reg.predict(Timestep)
            FilteredY = y_filt
```

```
            unique =  np.unique(y_filt, return_index=True)
            new_path = unique[0]
            new_time = Timestep[unique[1]]

        p_hat_iso = optimize.minimize(self.GLike,
                                      [1,1],
                                      args=(new_time, new_path),
                                      method='Nelder-Mead')
        x_isohat = p_hat_iso.x
        return x_isohat, FilteredY




class WienerProcess:

    def WLike(self, Theta, Timestep_dict, Paths_dict):

        if type(Paths_dict) == dict:
            L = []
            keys = Paths_dict.keys()
            for key in keys:
                Paths = Paths_dict[key]
                Timestep = Timestep_dict[key]

                lam, sig = Theta[0], Theta[1]
                dX = np.diff(Paths)

                Timestep = np.asarray(Timestep)
                dT = np.diff(Timestep)

                eta = (dX - lam*dT)/(sig*dT)

                el = -np.log(np.sqrt(2*math.pi)) - np.log(sig) -np.log(dT) -
                 ↪  0.5*(eta**2)
                l = np.sum(el)
```

```
                L.append(l)
            L = np.asarray(L)

        else:
            Paths = Paths_dict
            Timestep = Timestep_dict
            Paths = np.asarray(Paths)
            lam, sig = Theta[0], Theta[1]
            dX = np.diff(Paths)

            Timestep = np.asarray(Timestep)
            dT = np.diff(Timestep)

            eta = (dX - lam*dT)/(sig*dT)

            L = -np.log(np.sqrt(2*math.pi)) - np.log(sig) -np.log(dX) -
             ↪ 0.5*(eta**2)

        return -np.sum(L)


    def Wiener_MLE(self, Path_dict, Timestep_dict=None):

        if type(Path_dict) == dict and Timestep_dict == None:
            keys = Path_dict.keys()
            Timestep_dict = dict()
            for key in keys:
                path = Path_dict[key]
                timestep = np.asarray(list(range(len(path))))
                Timestep_dict[key] = timestep
        elif Timestep_dict == None and type(Path_dict) != dict:
            Timestep_dict = np.asarray(list(range(len(Path_dict))))

        x0 = [1,1]

        Param_hat = optimize.minimize(self.WLike,
                                      x0,
```

```
                                    args=(Timestep_dict, Path_dict),
                                    method='Nelder-Mead')

        return Param_hat.x
```

B.5 Main

This is where everything comes together. All the other modules are imported here and the regression analysis is carried out as well as the parameter estimation for the Wiener and Gamma Processes.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Apr 27 15:46:17 2021

@author: danem
"""

import Parser
import core
import RegressionTools
import ParameterEstimation
import statsmodels.api as sm
import numpy as np

import matplotlib.pyplot as plt


#%% Inputs

#valve descriptions file
v_des_fileName = r"C:\Users\emefo\OneDrive - NTNU\Data for Thesis\Valve
 ↪  Descriptions.xlsx"
v_des_sheetname = 'Sheet1'

#valve operation data file
v_data_fileName = r"C:\Users\emefo\OneDrive - NTNU\Data for Thesis\OPERATION
 ↪  TIMES DATA excel\active.xlsx"
```

```
v_data_sheetname = 'test_closing_no_PS_'


#%% Import Valve Descriptions

valve_Descr = core.ValveCategory()

parser = Parser.XLSParser()

parser.ImportValveCategory(valve_Descr, v_des_fileName, v_des_sheetname)

#%% Import Valve Data

vCat = valve_Descr.GetCategories()

valve_data = core.ValveData()

parser.ImportValveData(valve_data, v_data_fileName, v_data_sheetname, vCat)

#%% Regression Parameters

vData = valve_data.GetData()

regress = RegressionTools.Regression2()

noOfExplanatoryVariables = 13

X, y = regress.GetExplanatoryVariables(vData,vCat, noOfExplanatoryVariables)

#%% Regression

xName = []

relevant_variables = [0,1,2,3,4,5,6,7,8,9,10,11,12]
```

```
xVar = ['CIV','GLV', 'PWV', 'UMV', 'OTH','OK-long', 'OK-medium', 'OK-short',
 ↪  'NOT OK-long', 'Not OK-medium', 'NOT OK-short', 'prev last tt', 'prev
 ↪  first tt']

xName.extend([xVar[i] for i in relevant_variables])

xs =  X[:,relevant_variables]
ys = y

#xs = sm.add_constant(xs)

model = sm.OLS(ys,xs)
results = model.fit()

print(results.summary(yname='closing time',xname=xName))


#%% Standardized increments

civ = valve_Descr.civ
glv = valve_Descr.glv
pwv = valve_Descr.pwv
umv = valve_Descr.umv
oth = valve_Descr.oth

all_valves = []
all_valves.extend(civ)
all_valves.extend(glv)
all_valves.extend(pwv)
all_valves.extend(umv)
all_valves.extend(oth)

incr_dict = {}


regress.GetStandardizeIncrements(incr_dict, all_valves, vData, vCat,
 ↪  results, relevant_variables)
```

```
#%% Parameter Estimation

del incr_dict['13ESV5154']


gamma2 = ParameterEstimation.NoisyGamma2Parameters()

theta_gamma2, sigma_gamma2 = gamma2.EM_Gibbs_us(incr_dict)

c_gamma2 = np.asarray(theta_gamma2)[:,0]
u_gamma2 = np.asarray(theta_gamma2)[:,1]

#%%
a = sigma_gamma2
print(len(a))
print(np.mean(a))
print(np.std(a))
print(min(a))
print(np.percentile(a, 0.25))
print(np.percentile(a, 0.5))
print(np.percentile(a, 0.75))
print(max(a))


#%% Evolution of the gamma parameters under the stochatic expectation
 ↪  maximization

plt.figure()
plt.plot(sigma_gamma2)
plt.title(r'sigma, $\sigma$')
plt.xlabel('No of iterations')
#Tab_theta = np.asarray(Tab_theta)
#plt.plot(Tab_theta[:,2])*
plt.figure()
```

```
plt.plot(u_gamma2)
plt.title('intensity, u')
plt.xlabel('No of iterations')

plt.figure()
plt.plot(c_gamma2)
plt.title('shape, c')
plt.xlabel('No of iterations')


#%% Parameter Estimation - Wiener Process

wiener = ParameterEstimation.WienerProcess()
x_hat = wiener.Wiener_MLE(incr_dict)

print(x_hat)
```

Appendix C

Figures

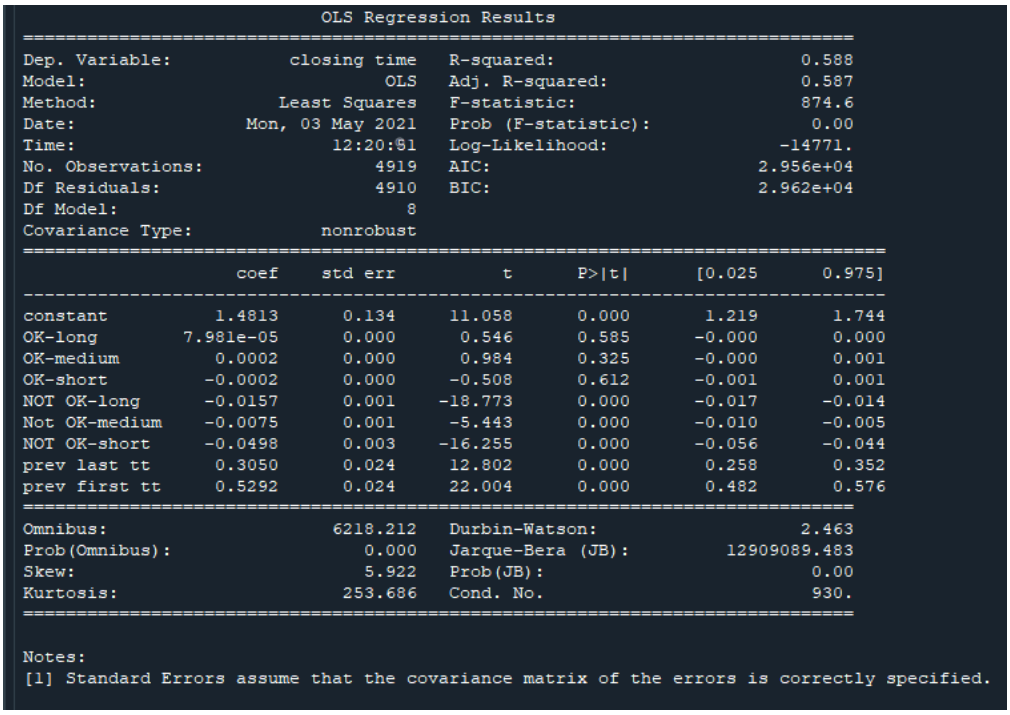
C.1 Valve activation data structure

	A	B	C	D	E	F	G	H	I	J
1	Valve	Timestamp	Operation	Group	Is to safe state	Description	Operation Status	Travel	Max	Have Comment
2	56ESV1022	31.01.2016 03:16	PCS Close	ESD	TRUE	Olje fra avløpsvannbeh til 3 tr sep	OK	9.00	30.00	FALSE
3	56ESV1022	31.01.2016 02:20	PCS Open	ESD	FALSE	Olje fra avløpsvannbeh til 3 tr sep	OK	5.00	150.00	FALSE
4	56ESV1022	31.01.2016 02:17	PCS Open (Cancelled	ESD	FALSE	Olje fra avløpsvannbeh til 3 tr sep	OK	4.00	150.00	FALSE
5	56ESV1022	31.01.2016 02:02	PCS Close	ESD	TRUE	Olje fra avløpsvannbeh til 3 tr sep	OK	9.00	30.00	FALSE
6	56ESV1022	31.01.2016 01:57	PCS Open	ESD	FALSE	Olje fra avløpsvannbeh til 3 tr sep	OK	5.00	150.00	FALSE
7	56ESV1022	31.01.2016 01:56	PCS Close	ESD	TRUE	Olje fra avløpsvannbeh til 3 tr sep	OK	9.00	30.00	FALSE
8	56ESV1022	31.01.2016 01:48	PCS Open	ESD	FALSE	Olje fra avløpsvannbeh til 3 tr sep	OK	5.00	150.00	FALSE

Figure C.1: Valve activation data structure.

C.2 Regression Results

Figure C.2 shows the full regression result for the model in Equation 8.3



OLS Regression Results

Dep. Variable:	closing time	R-squared:	0.588
Model:	OLS	Adj. R-squared:	0.587
Method:	Least Squares	F-statistic:	874.6
Date:	Mon, 03 May 2021	Prob (F-statistic):	0.00
Time:	12:20:51	Log-Likelihood:	-14771.
No. Observations:	4919	AIC:	2.956e+04
Df Residuals:	4910	BIC:	2.962e+04
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
constant	1.4813	0.134	11.058	0.000	1.219	1.744
OK-long	7.981e-05	0.000	0.546	0.585	-0.000	0.000
OK-medium	0.0002	0.000	0.984	0.325	-0.000	0.001
OK-short	-0.0002	0.000	-0.508	0.612	-0.001	0.001
NOT OK-long	-0.0157	0.001	-18.773	0.000	-0.017	-0.014
Not OK-medium	-0.0075	0.001	-5.443	0.000	-0.010	-0.005
NOT OK-short	-0.0498	0.003	-16.255	0.000	-0.056	-0.044
prev last tt	0.3050	0.024	12.802	0.000	0.258	0.352
prev first tt	0.5292	0.024	22.004	0.000	0.482	0.576

Omnibus:	6218.212	Durbin-Watson:	2.463
Prob(Omnibus):	0.000	Jarque-Bera (JB):	12909089.483
Skew:	5.922	Prob(JB):	0.00
Kurtosis:	253.686	Cond. No.	930.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Figure C.2: Regression results from improved model

Figure C.3 shows the estimated coefficients for the model in Equation 9.3

```
main2.py )
                            OLS Regression Results
==============================================================================
Dep. Variable:           closing time   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     629.4
Date:                Tue, 04 May 2021   Prob (F-statistic):               0.00
Time:                        11:49:11   Log-Likelihood:                -14658.
No. Observations:                4919   AIC:                         2.934e+04
Df Residuals:                    4906   BIC:                         2.943e+04
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
CIV              4.9018      0.345     14.189      0.000       4.225       5.579
GLV              0.3567      0.228      1.564      0.118      -0.090       0.804
PWV              0.5745      0.227      2.529      0.011       0.129       1.020
UMV              0.5552      0.234      2.370      0.018       0.096       1.015
OTH              2.8742      0.176     16.325      0.000       2.529       3.219
OK-long      -1.171e-05      0.000     -0.082      0.935      -0.000       0.000
OK-medium        0.0001      0.000      0.534      0.593      -0.000       0.000
OK-short         0.0002      0.000      0.374      0.708      -0.001       0.001
NOT OK-long     -0.0147      0.001    -17.634      0.000      -0.016      -0.013
Not OK-medium   -0.0064      0.001     -4.630      0.000      -0.009      -0.004
NOT OK-short    -0.0442      0.003    -13.903      0.000      -0.050      -0.038
prev last tt     0.2515      0.024     10.658      0.000       0.205       0.298
prev first tt    0.4853      0.024     20.336      0.000       0.439       0.532
==============================================================================
Omnibus:                     6427.401   Durbin-Watson:                   2.369
Prob(Omnibus):                  0.000   Jarque-Bera (JB):         11581163.386
Skew:                           6.392   Prob(JB):                         0.00
Kurtosis:                     240.364   Cond. No.                     2.57e+03
==============================================================================
```

Figure C.3: Coefficients from the adjusted model

Appendix D

Python Codes 2

D.1 Original Noisy Gamma Process Code

This section contains the original codes for estimating the gamma parameters of the noisy gamma process written by Xingheng Liu.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Mar 26 15:02:26 2021

@author: visca
"""

from sklearn.isotonic import IsotonicRegression

import numpy as np
import random as rd
import matplotlib.pyplot as plt
import copy

from scipy.stats import beta
from scipy import optimize
from scipy.special import gamma
from scipy.stats import norm
from scipy.stats import uniform
from scipy import stats
```

```
#%%

# Generate a time-dependent gamma sequence using
# Theta = [c,b,u] the gamma parameters. See Zhang Yun 2016
# Timestep: time sequence

def GenerateNoisyGammab(Theta, Timestep):
    c, b, u, sigma= Theta[0], Theta[1], Theta[2], Theta[3]
    Increments = [0]
    Noise = np.array(norm.rvs(loc = 0,scale = sigma, size = len(Timestep)))
    for i in range(1,len(Timestep)):

        ↪ Increments.append(rd.gammavariate(c*Timestep[i]**b-c*Timestep[i-1]**b,
        ↪ 1/u))
    GammaPath = np.cumsum(Increments)
    NoisyPath = GammaPath + Noise
    return NoisyPath, GammaPath, Noise

# The two functions below are rejection sampling algorithm
# and is an improved version of what has been described in Le Son 2016;
# we changed the proposal distribution so that the envelope fits better the
 ↪ target distribution
# envelope: Beta distribution

def kernel1rvs(c,d,gam,delt,sigma,Y):
    P=250
    U1 = beta.rvs(a=gam+1,b=delt+1,loc=c,scale=d-c, size=P)
    U2 = uniform.rvs(size=P)
    cpr = U2<=np.exp(-(Y-U1)**2/(2*sigma**2))
    idx = np.argmax(cpr)
    sample = U1[idx]
    return sample

# Envelope: Gamma distribution
def kernel2rvs(c,bet,gam,sigma,Y):
    P=250
    U1 = stats.gamma.rvs(a=gam+1,loc=c,scale=1/bet, size=P)
```

```
    U2 = uniform.rvs(size=P)
    cpr = U2<=np.exp(-(Y-U1)**2/(2*sigma**2))
    idx = np.argmax(cpr)
    sample = U1[idx]
    return sample

# Glike: return the opposite value of Log-likelihood given all the sampled
 ↪ paths
# Paths: a array containing all the samples
def GLike(Theta, Timestep, Paths):
    Paths = np.asarray(Paths)
    c, b, u = Theta[0], Theta[1], Theta[2]
    dX = np.diff(Paths)
    dX[dX<=0]=1e-20 #to avoid numerical errors
    t_start, t_end = Timestep[0:-1], Timestep[1:]
    eta = c*t_end**b - c*t_start**b
    L = (eta*np.log(u)-np.log(gamma(eta))+(eta-1)*np.log(dX)-u*dX)
    return -np.sum(L)

# Gibbs sampling, details can be found in Le son 2016
# sigma: standard deviation of the Gaussian noise
# Qm: number of samples
# Qs: samples before Qs are discarded to ensure the stable state of Markov
 ↪ chain
def GibbsSampling(Path, Timestep, Theta, sigma, Qm, Qs, Y0):
    n = len(Timestep)
    alpha = Theta[0]
    b = Theta[1]
    bet = Theta[2]
    Z_tab = [Y0]
    for Q in range(1, Qm):
        z = copy.deepcopy(Z_tab[Q-1])
        for i in range(n):
            if i==0:
                z[i]=0
            elif i<n-1:
                if z[i+1]<=z[i-1]: # to avoid numerical errors
```

```
                    z[i]=z[i+1]
                else:
                    c, d = z[i-1], z[i+1]
                    t1, t2, t3 = Timestep[i-1], Timestep[i], Timestep[i+1]
                    gam = alpha*(t2**b-t1**b)-1
                    delt = alpha*(t3**b-t2**b)-1
                    z[i] = kernel1rvs(c,d,gam,delt,sg,Path[i])
            else:
                c= z[i-1]
                t1, t2 = Timestep[i-1], Timestep[i]
                gam = alpha*(t2**b-t1**b)-1
                z[i] = kernel2rvs(c,bet,gam,sg,Path[i])
        Z_tab.append(z)
    return Z_tab[Qs:Qm]

# Expectation maximization: iterative approach for statistical inference
# M: total number of iteration
# x0: initial guess for the gamma parameter
# y0: initial guess for Gibbs sampling
def EM_Gibbs_us(Path, Timestep, Qm, Qs, M, x0, y0):
    sigma0 = np.sqrt(np.sum((Path-y0)**2)/len(Timestep))
    Tab_sigma=[sigma0]
    Tab_theta=[x0]
    for i in range(M):
        SampledPath = GibbsSampling(Path, Timestep, Tab_theta[-1],
         ↪ Tab_sigma[-1], Qm, Qs, y0)
        var = np.sum((SampledPath-Path)**2)
        NewSigma = np.sqrt(var/(len(SampledPath)*len(Path)))
        Param_hat = optimize.minimize(GLike,
                      x0,
                      args=(Timestep, SampledPath),
                      method='Nelder-Mead')
        Tab_theta.append(Param_hat.x)
        Tab_sigma.append(NewSigma)
        print('i=',i)
    return Tab_theta, Tab_sigma
```

```
# Isotonic regression: derive the initial guess for parameter/gibbs sampling

def IsoX0(Path,Timestep):
    iso_reg = IsotonicRegression(y_min=0).fit(Timestep, Path)
    FilteredY = iso_reg.predict(Timestep)
    unique =  np.unique(FilteredY, return_index=True)
    new_path = unique[0]
    new_time = Timestep[unique[1]]
    p_hat_iso = optimize.minimize(GLike,
                          [1,1,1],
                          args=(new_time, new_path),
                          method='Nelder-Mead')
    x_isohat = p_hat_iso.x
    return x_isohat, FilteredY

#%% Generate Gamma degradation paths
gap = 2
Timestep = np.arange(0,200,gap)
n = len(Timestep)
# Gamma parameters and time
sg = .15
Gamma_par = [0.00059, 2, 3.65543, sg]


Path, GammaPath, noise = GenerateNoisyGammab(Gamma_par, Timestep)
plt.plot(Timestep, Path, label='Observation')
plt.plot(Timestep, GammaPath, label='Hidden path')
plt.plot(Timestep, noise, label='Gaussian noise')
plt.legend()

# Isotonic regression
x0, y0 = IsoX0(Path,Timestep)

# The first 50 samples are discarded
Qm, Qs=150, 50
Theta = [0.00059, 2, 3.65543]
samples = GibbsSampling(Path, Timestep, Theta, sg, Qm, Qs,y0)
```

```
for s in samples:
    plt.plot(Timestep, s, C = 'blue', alpha=0.1)
plt.plot(Timestep, Path, C = 'red', LineWidth=2, label='Observation')



#%% MLE and EM

# MLE based on the true (hidden) degradation path

p_hat_truepath = optimize.minimize(GLike,
                       x0,
                       args=(Timestep, GammaPath),
                       method='Nelder-Mead')
x_truehat = p_hat_truepath.x

plt.plot(Timestep, Path)
plt.plot(Timestep,
 ↪  x_truehat[0]*Timestep**x_truehat[1]/x_truehat[2],label='EX hat')
plt.plot(Timestep, Theta[0]*Timestep**Theta[1]/Theta[2],label='true param')
plt.legend()

# EM based on the noisy degradation path

M=200
Tab_theta, Tab_sigma = EM_Gibbs_us(Path, Timestep, 200, 50, M, x0, y0)
plt.plot(Tab_sigma)
Tab_theta = np.asarray(Tab_theta)
plt.plot(Tab_theta[:,2])
plt.plot(Tab_theta[:,1])
plt.plot(Tab_theta[:,0])

# EM result: last element of the parameter table
theta_hat = Tab_theta[-1]

# Compare the mean E(X)
```

```
plt.plot(Timestep, Path)
plt.plot(Timestep,
 ↪  x_truehat[0]*Timestep**x_truehat[1]/x_truehat[2],label='EX hat')
plt.plot(Timestep,
 ↪  theta_hat[0]*Timestep**theta_hat[1]/theta_hat[2],label='true param')
plt.legend()

# Compare the variance Var(X)
plt.plot(Timestep,
 ↪  x_truehat[0]*Timestep**x_truehat[1]/x_truehat[2]**2,label='EX hat')
plt.plot(Timestep,
 ↪  theta_hat[0]*Timestep**theta_hat[1]/theta_hat[2]**2,label='true param')
plt.legend()
```

Bibliography

Xuewu Dai and Zhiwei Gao. From model, signal to knowledge: A data-driven perspective of fault detection and diagnosis. *IEEE Transactions on Industrial Informatics*, 9(4):2226–2238, 2013. doi: 10.1109/TII.2013.2243743.

Marvin Rausand and Arnljot Høyland. *System Reliability Theory: Models, Statistical Methods, and Applications*. Wiley, Hoboken, NJ, 2nd edition, 2004.

Fisher Emerson Controls. *Control Valve Handbook*. Fisher Controls International, LLC., Mashalltown, Iowa, 2017.

IEC 61508 :2010. Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1-7. Standard, International Electrotechnical Commission, Geneva, apr 2010.

Mary Ann Lundteigen and Marvin Rausand. Partial stroke testing of process shutdown valves: How to determine the test coverage. *Journal of Loss Prevention in the Process Industries*, 21(6): 579 – 588, 2008. ISSN 0950-4230. doi: https://doi.org/10.1016/j.jlp.2008.04.007. URL http://www.sciencedirect.com/science/article/pii/S095042300800034X.

OREDA : offshore reliability data handbook : Vol. 1 : Topside equipment, volume Vol. 1. OREDA Participants, Trondheim, 5th ed. edition, 2009. ISBN 9788214048308.

H. Srivastav, A. V. Guilherme, A. Barros, M. A. Lundteigen, F. B. Pedersen, A. Hafver, and F. L. Oliveira. Optimization of periodic inspection time of sis subject to a regular proof testing. In *Safety and Reliability – Safe Societies in a Changing World*, pages 1125–1131. CRC Press, 2018. ISBN 9781351174664. doi: 10.1201/9781351174664-142. URL https://dx.doi.org/10.1201/9781351174664-142.

Andreas Hafver, Luiz Fernando Oliveira, and Frank Borre Pedersen. Optimal scheduling of tests of safety systems, considering test-induced degradation. In *Proceedings of the 29th European Safety and Reliability Conference.*, pages 4084–4090, Singapore, 09 2019. Research Publishing. ISBN 981-973-0000-00-0. doi: 10.3850/981-973-0000-00-0.

Islam Ariful, Srivastav Himanshu, Vatn Jørn, Anne Barros, and Mary Lundteigen. Time-dependent unavailability assessment of final element of safety instrumented systems- an application of multiphase markov process. In *Proceedings of the 29th European Safety and Reliability Conference (ESREL)*, pages 2583–2590, 01 2019. doi: 10.3850/978-981-11-2724-3_0515-cd.

Himanshu Srivastav, Anne Barros, and Mary Ann Lundteigen. Modelling framework for performance analysis of sis subject to degradation due to proof tests. *Reliability Engineering & System Safety*, 195:106702, 2020. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2019.106702. URL http://www.sciencedirect.com/science/article/pii/S0951832019301450.

Pengyu Zhu, Jayantha Liyanage, and Simon Jeeves. Data-driven failure analysis of emergency shutdown systems in oil and gas industry: Evaluation of detectability from failure progression perspective. *Journal of quality in maintenance engineering*, 26(1):1–18, 2020. ISSN 1355-2511.

Marvin Rausand. *Reiability of Safety-Critical Systems: Theory and Applications*. Wiley, Hoboken, NJ, 2014.

Bill Reeson. Solenoid pilot valves for valve actuation. *Valve Magazine*, 18:C2 – C4, 2006.

SAMSON :2017. Competence in Functional Safety: Application notes for safety-instrumented systems. Manual, SAMSON, Frankfurt, Germany, 2017.

G Gokilakrishnan, R Rajesh, V Selvakumar, and Raajeshkrishna C.R. Operating torque in ball valves-a review. 12 2014.

Philip J. O'Keefe. Engineering Expert Witness Blog the solenoid valve operates a pneumatic actuator. http://www.engineeringexpert.net/Engineering-Expert-Witness-Blog/tag/scotch-yoke, 2018. Accessed: 2020-11-19.

Maurice Stewart. 4 - piping system components. In Maurice Stewart, editor, *Surface Production Operations*, pages 193–300. Gulf Professional Publishing, Boston, 2016. ISBN 978-1-85617-808-2. doi: https://doi.org/10.1016/B978-1-85617-808-2.00004-3. URL https://www.sciencedirect.com/science/article/pii/B9781856178082000043.

Xin Yan, Xin Yan, and Xiao Gang Su. *Linear regression analysis: theory and computing*. World Scientific Publishing Co. Pte. Ltd, 2009. ISBN 9789812834102.

A.V. Prokhorov. Encyclopedia of Mathematics regression analysis. http://encyclopediaofmath.org/index.php?title=Regression_analysis&oldid=48473, 2020. Accessed: 2020-11-23.

Scott Pardo. *Statistical Analysis of Empirical Data: Methods for Applied Sciences*. Springer International Publishing AG, Cham, 2020. ISBN 3030433277.

Jim Frost. Statistics by Jim: making statistic intuitive. http://https://statisticsbyjim.com/jim_frost/, 2020. Accessed: 2020-12-03.

PG Hoel. *Introduction to mathematical statistics*. Wiley, New York, 4th edition, 1971.

Kwok L Tsui, Nan Chen, Qiang Zhou, Yizhen Hai, and Wenbin Wang. Prognostics and health management: A review on data driven approaches. *Mathematical problems in engineering*, 2015:1–17, 2015. ISSN 1024-123X.

Joel Finch. Toyota sudden acceleration: a case study of the national highway traffic safety administration-recalls for change. 22, 01 2009.

J.B Bowles. A survey of reliability-prediction procedures for microelectronic devices. 41(1):2–12, 1992. ISSN 0018-9529.

J.Z Sikorska, M Hodkiewicz, and L Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical systems and signal processing*, 25(5):1803–1836, 2011. ISSN 0888-3270.

ISO 13381-1 :2015. Condition monitoring and diagnostics of machines — Prognostics – Part 1:General guidelines. Standard, International Organization for Standardization, Geneva Switzerland, sep 2015.

Xiao-Sheng Si, Wenbin Wang, Chang-Hua Hu, and Dong-Hua Zhou. Remaining useful life estimation – a review on the statistical data driven approaches. *European Journal of Operational Research*, 213(1):1–14, 2011. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2010.11.018. URL https://www.sciencedirect.com/science/article/pii/S0377221710007903.

C. Joseph Lu and William O Meeker. Using degradation measures to estimate a time-to-failure distribution. *Technometrics*, 35(2):161–174, 1993. ISSN 0040-1706.

Jye-Chyi Lu, Jinho Park, and Qing Yang. Statistical inference of a time-to-failure distribution derived from linear degradation data. *Technometrics*, 39(4):391–400, 1997. ISSN 0040-1706.

William Q Meeker and Luis A Escobar. Statistical methods for reliability data, 1998.

Richard E Barlow. Mathematical theory of reliability, 1965.

J.M. van Noortwijk. A survey of the application of gamma processes in maintenance. *Reliability Engineering & System Safety*, 94(1):2–21, 2009. ISSN 0951-8320. doi: https://doi.org/10.

1016/j.ress.2007.03.019. URL https://www.sciencedirect.com/science/article/pii/S0951832007001111. Maintenance Modeling and Application.

Mohamed Abdel-Hameed. A gamma wear process. *IEEE Transactions on Reliability*, R-24(2): 152–153, 1975. doi: 10.1109/TR.1975.5215123.

J. Lawless and M. Crowder. Covariates and random effects in a gamma process model with application to degradation and failure. *Lifetime Data Analysis*, 10(3):213–227, 2004. ISSN 13807870. doi: 10.1023/B:LIDA.0000036389.14073.dd. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-4043052862&doi=10.1023%2fB%3aLIDA.0000036389.14073.dd&partnerID=40&md5=80ce56fa525024a8c09684ea15b43861. cited By 376.

M.J. Kallen and J.M. van Noortwijk. Optimal maintenance decisions under imperfect inspection. *Reliability Engineering & System Safety*, 90(2):177–185, 2005. ISSN 0951-8320. doi: https://doi.org/10.1016/j.ress.2004.10.004. URL https://www.sciencedirect.com/science/article/pii/S0951832004002443. Selected papers from ESREL 2003.

D. Frangopol, M. Kallen, and J. Van Noortwijk. Probabilistic models for life-cycle performance of deteriorating structures: Review and future directions. *Progress in Structural Engineering and Materials*, 6(4):197–212, 2004. cited By 275.

B. Sun, M. Yan, Q. Feng, Y. Li, Y. Ren, K. Zhou, and W. Zhang. Gamma degradation process and accelerated model combined reliability analysis method for rubber o-rings. *IEEE Access*, 6:10581–10590, 2018. ISSN 21693536. doi: 10.1109/ACCESS.2018.2799853. URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85041381019&doi=10.1109%2fACCESS.2018.2799853&partnerID=40&md5=e4eddd47ebebb3f56eda571aca9b91a5. cited By 15.

Aibo Zhang, Anne Barros, and Yiliu Liu. Performance analysis of redundant safety-instrumented systems subject to degradation and external demands. *Journal of Loss Prevention in the Process Industries*, 62:103946, 2019. ISSN 0950-4230. doi: https://doi.org/10.1016/j.jlp.2019.103946. URL https://www.sciencedirect.com/science/article/pii/S0950423019305741.

Mabrouk Nabila, Moulahi Med Hedi, and Ben Hmida Fayçal. Gamma process with covariets and remaining useful life improvement in an actuator. In *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pages 208–213, 2019. doi: 10.1109/STA.2019.8717206.

Laurent Bordes, Christian Paroissin, and Ali Salami. Combining Gamma and Brownian processes for degradation modeling in presence of explanatory variables. working paper or preprint, November 2010. URL https://hal.archives-ouvertes.fr/hal-00535812.

Vilijandas Bagdonavicius and Mikhail Nikulin. Estimation in degradation models with explanatory variables. *Lifetime Data Analysis*, 7:85–103, 03 2001. doi: 10.1023/A:1009629311100.

Martin Crowder and Jerald Lawless. On a scheme for predictive maintenance. *European Journal of Operational Research*, 176(3):1713–1722, 2007. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2005.10.051. URL https://www.sciencedirect.com/science/article/pii/S0377221705008763.

Khanh Le Son, Mitra Fouladirad, and Anne Barros. Remaining useful lifetime estimation and noisy gamma deterioration process. *Reliability engineering & system safety*, 149:76–87, 2016. ISSN 0951-8320.

G CASELLA and EI GEORGE. Explaining the gibbs sampler. *The American statistician*, 46(3):167–174, 1992. ISSN 0003-1305.

François Chollet. *Deep learning with Python*. Manning Publications Co, Shelter Island, New York, 2018. ISBN 9781617294433.

NOG GL-070 :2018. Application of IEC 61508 and IEC 61511 in the Norwegian Petroleum industry. Standard, Norwegian Oil and Gas, Stavanger, 2016.

Scott R Eliason. *Maximum likelihood estimation : logic and practice*, volume 96 of *Quantitative applications in the social sciences ;*. SAGE, Newbury Park, Calif. ;, 1993. ISBN 9781412984928.

Anne Barros. Data driven prognostic and predictive maintenance, tpk4450 compendium, 2019.

Søren Feodor Nielsen. The stochastic em algorithm: Estimation and asymptotic results. *Bernoulli*, 6(3):457–489, 2000. ISSN 13507265. URL http://www.jstor.org/stable/3318671.

Emefon Dan

Master's thesis

2021

NTNU
Kunnskap for en bedre verden