Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

Jan Henrik Jahren

Data-driven modelling of subsea equipment degradation using simulated and experimental case studies

Master's thesis in Nanotechnology
Supervisor: Johannes Jäschke
December 2020

NTNU
Norwegian University of Science and Technology

Jan Henrik Jahren

Data-driven modelling of subsea equipment degradation using simulated and experimental case studies

Master's thesis in Nanotechnology
Supervisor: Johannes Jäschke
December 2020

Norwegian University of Science and Technology
Faculty of Natural Sciences
Department of Chemical Engineering

NTNU
Norwegian University of
Science and Technology

Sammendrag

I all industri finnes det visse essensielle komponenter som utsettes for slitasje fra flere ulike kilder. I denne oppgaven studeres spesifikt erosjonen fra sand i den såkalte strupeventilen (”choke valve”) i et subsea oljeproduksjonssystem. Målet med oppgaven er å undersøke muligheten til å erstatte tradisjonelle fenomenologiske modeller for degradering med data-drevne modeller basert på statistisk læring. Dette gjøres gjennom fire ulike casestudier, tre simulerte case studier som tar for seg henholdsvis konstant, eksponentiell og logistisk sand produksjon fra en oljebrønn og hvordan dette degraderer strupeventilen. Til slutt har vi brukt et eksperimentelt oppsett ved instituttet som emulerer et subsea oljeproduksjonssystem. Her har vi studert erosjonen av 3D-printede prøver utsatt for en strøm av vann (uten sand). Fra alle fire casestudiene ble det konstruert datasett og med disse datasettene ble maskinlæringsmodeller trent. Her er det både studert både tradisjonelle statistiske modeller som stegvis lineær regresjon, regresjonstrær, ensembler av trær og støttevektormaskiner samt mer moderne nevrale nettverk. I de første tre casestudiene ble erosjonsraten estimert ved hjelp av prosessvariabler som samles under simuleringen. Dette gav svært gode resultater på den simulerte dataen.

Det å direkte måle erosjonen i det eksperimentelle oppsettet var en stor utfordring, først ble det forsøkt med kamera og telling av piksler som fortsatt var tilstede, men det gav svært ustabil data. Til slutt bestemte vi oss for å representere erosjonen av prøven med endringen av trykkforskjellen over erosjonskammeret. Ettersom prøven blokkerer flyt, vil trykkforskjellen jevnes mer og mer ut ettersom prøven forsvinner. Disse degraderingsprofilene ble også relativt suksessfullt modellert ved bruk av maskinlæringsmodeller. Gjennom oppgaven diskuteres fordeler og ulemper med de ulike metodene både kvantitativt og kvalitativt i de ulike casestudiene.

Summary

In all industry, there are certain essential components that are subject to degradation from various sources. In this thesis, specifically the erosion of a choke valve from sand production in a subsea oil production network is studied. The goal of this thesis is to explore the possibility of replacing traditional phenomenological models and expensive inspections with more accurate data-driven degradation models based on statistical learning. This is done through four case studies, three simulated case studies dealing with respectively constant, exponential and logistic sand production from the oil well. Finally, this is complemented by an experimental case study where a rig emulating a subsea oil well network in the department of chemical engineering is used. Here we have studied the erosion of 3D-printed probes exposed to flows of water (without sand). From all four case studies, data sets were constructed, which were then used to train machine learning models. Both traditional machine learning models like stepwise linear regression, regression trees, ensembles of trees and support vector regression as well as more modern methods such as neural networks were tested. In the first three case studies the erosion rate was estimated using process variables recorded during the simulations, this yielded very strong predictions, particularly (in case studies 2 & 3) when accurate sand production data was available.

To directly measure the erosion of the probe in the experimental setup was a significant challenge, initially we tried to use a camera and count pixels present in the images, but this led to noisy data of low quality. Finally, we decided to use the change in differential pressure across the erosion chamber to represent the degradation of the probe. The probe was positioned in a way where it blocks the flow through the erosion chamber, and as such, as its area is reduced the pressure across the erosion chamber equalises. These degradation profiles were quite successfully modelled using machine learning models. Throughout this thesis, the methods applied are discussed in terms of benefits and drawbacks, both quantitatively and qualitatively.

Preface

This work is the product of the final semester of my integrated master's degree in nanotechnology, where a 30 ECTS independent research project is done. My thesis theme of data-driven modelling is a quite unconventional choice for students in my study programme, the choice is a result of a strong interest in statistics and modelling after taking classes in chemometrics and statistical learning here at NTNU in my 4th year. My main supervisor for this project has been assoc. prof Johannes Jäschke at the Process Systems Engineering group at the Department of Chemical Engineering. Co-supervisor Jose O. A. Matias provided most of the day to day supervision being available any time for advice on zoom and on email, for which I am extremely grateful.

Jose also built the rig where the experimental work has been carried out, throughout the last months we have spent countless hours waiting for a diverse selection of 3D printed probes to erode away. This project has been an extension of my specialisation project where most of the work was based on the case study of constant sand production rate. In this master project the work has been extended to simulated data with exponential and logistic sand production rates as well as actually performing experiments to investigate the applicability of these methods for real world data. A vast amount of time has been spent optimising the process of gathering experimental data, some of it made it into the thesis but most of it will pass quietly into the dustbin of things that were tried and did not work. The analysis and modelling in this work has been done independently by me, with advise from Jose, while the experimental work has been done jointly. Based on the work in this thesis, a paper was submitted to the ESCAPE-31 conference, currently pending approval, co-authored by my supervisors titled "Data-driven modelling of choke valve erosion usingdata simulated from a first principles model". The paper is included in Appendix E.

Table of Contents

List of Tables

List of Figures

Abbreviations

ANN =	=	Artificial Neural Network
CV	=	Cross validation
dP	=	Differential pressure
GOR	=	Gas Oil Ratio
MLR	=	Multiple linear regression
MSE	=	Mean square error
NARX	=	Non-linear autoregressive exogenous model
NLIO	=	Non-linear input/output model
OLS	=	Ordinary least squares
PCA	=	Principal Components Analysis
PI	=	Productivity Index
PLS(R)	=	Partial least squares (Regression)
RMSE	=	Root mean square error
RSS	=	Residual sum of squares
SGD	=	Stochastic Gradient Descent
SPE	=	Sum of prediction errors
SVM	=	Support vector machine
SVR	=	Support vector regression
TSS	=	Total sum of squares

Chapter 1

Introduction

Failure to detect faults in large scale, expensive or critical equipment can have immense consequences. Both financial, environmental and in the most extreme cases, loss of human life. One of the main mechanisms in equipment degradation in subsea oil extraction, is sand erosion. Accurately modelling this process is vital for monitoring of equipment health.[4][5] Erosion by sand is a very complex process and, thus, difficult to model using physical domain knowledge. Because of this difficulty, this thesis proposes the use of data-driven approaches for modelling erosion in critical equipment of a subsea oil production rig. In such systems, a multitude of available process measurements, such as flowrates and pressures, can be combined to a soft-sensor for component degradation. This approach could save significant amounts of resources by allowing fewer cost intensive inspections and monitoring schemes as well as improving safety.

1.1 Motivation

In modern production systems, advanced process control is often applied to optimise production and profit under certain constraints. One significant constraint that has to be taken into account in all process control systems is the safety of the equipment and potential degradation of said equipment which may cause operational halts, economic losses or asset damage when not properly handled. Control algorithms, being subject to constraints from degradation of components need good and certain estimates of this degradation for optimal operation. In cases where there is a large amount of uncertainty in the models, worst case operation is often used where one assumes the worst case of degradation just to be absolutely safe that no equipment failure will occur, this leads to lost profits from too conservative operation. More accurate models, as well as specific estimates of the uncertainties have the potential to allow for less restrictive operational control which in turn means larger profits and more efficient operations. In this work it will be investigated if statistical learning methods from basic linear regressions to complex neural networks can provide these accurate models that are needed.

1.2 Methods

To investigate the usefulness of a data-driven modelling approach, statistical learning methods will be tested on simulated data using the subsea gas lifted oil well network model proposed by Krishnamoorthy et al.[6] and adapted by Verheylewegen and Jäschke[2]. A final case study using experimental data gathered from a laboratory rig at the Department of Chemical Engineering at NTNU. This work will explore four different case studies, each generating a separate data set that is independently analysed and modelled using the data-driven methods to investigate the applicability of such methods to equipment degradation:

- **Case study 1**: Simulated erosion of choke valve in gas lifted subsea oil production system, sand production rate is held constant. Useful primarily for initial exploration of methods to see if further studies are merited.
- **Case study 2**: Simulated erosion of choke valve in gas lifted subsea oil production system with a non-constant sand production rate. Real reservoirs do not produce the same amount of sand at all times, the profile of the sand production rate is usually increasing as a field matures[7][8], in case study two a exponential sand production rate is investigated. No random variance was added to the sand production rate and the sand production rate was sampled every 50 days.
- **Case study 3**: Simulated erosion of choke valve in gas lifted subsea oil production system with a non-constant sand production rate. Once again a non-constant sand production rate was tested, but this time a logistic profile was investigated and some random low magnitude stochastic noise was added to the sand production and the sand production was still sampled every 50 days.
- **Case study 4**: Experimental validation of methods, using a laboratory rig that emulates a three well gas lifted oil well production system. Multiple iterations of the data acquisition and probe setups were tested before a final experimental setup was found that yielded usable data.

1.3 Outline

This master's thesis is split up into several chapters where the different case studies are handled. First the theory for analysing, pre-processing and building regression models of the data will be presented in Chapter 2. In Chapter 3, the model of the gas lifted subsea oil production system will be presented as well as the adaptions for simulating erosion and handling non-constant sand production rates, a quick overview of the data resulting from the simulations will also be given. Chapters 4 and **??** will present the results from applying the statistical learning methods to the simulated data from the *in silico* case studies. In Chapter 6 the experimental rig will be presented, the variables being recorded, how these variables are used to measure erosion and a quick overview of the resulting data. Chapter 7 will present the results from applying the statistical learning methods to the experimental data gathered. Finally, Chapter 8 will sum up the results and present the conclusions that can be drawn from the results as well as suggest future work that can be done to further

develop the approach. The code used to generate the the simulated data, as well as for pre-processing and modelling of all the data for the different case studies are found in the Appendix along with the paper that resulted from the work with first Chapters of this thesis.

Chapter 2

Methods and theory for data processing

This chapter will introduce the methods that will be used in the modelling and analysis of both simulated and experimental data. First data needs to be pre-processed, for this we will discuss centring and scaling, interpolation, normalisation and principal components analysis. Then the different regression methods that are tested and compared will be introduced, including classical statistical learning methods like linear regression and more modern methods such as artificial neural networks (ANN). Then a brief overview of model selection and model evaluation as well as the optimisation algorithms used to tune the models.

2.1 Data Pre-Processing

This section will mainly deal with the various ways the data has to be treated before analysis is performed. This can for instance be centring and scaling so that no variable dominates the statistical method simply due to having the largest scale and thus larger variances. Another way to solve the problem of different scales for different variables is normalisation. Normalisation is very commonly used when dealing with different scales, but does ”waste” the information inherent in the scales of variables. For any case where one variable has a much larger scale and is also a dominant influence, scaling it down to the same value as other variables might not be beneficial for the modelling. When dealing with high dimensional data, principal component analysis (PCA) can be a beneficial way to reduce the feature space to a more manageable size, while maintaining most of the information. PCA is very dependent on the scale of variances which in turn scales with variable scale and thus should in general be applied to normalised data. In general throughout this thesis I will stick to that the data matrix $\mathbf{X}$ contains p different variables and n different samples resulting in a data matrix of dimensions n by p.

2.1.1 Centering and Scaling

Centering

Centering is performed by calculating the mean of each column, constructing a mean row made up of each of the column means then subtracting this from every row in the data matrix **X**:

$$\mathbf{X}_{centered} = \mathbf{X} - \mu \tag{2.1}$$

where μ is a mean vector where each row contains column means of **X**, such that this subtraction is equal to subtracting the mean value for each measurement from each sample of that measurement.

Scaling

Scaling of the data can be done in various ways. When using latent variable methods on data of very different scales, one should scale the data so that the variables with the largest numerical values don't automatically dominate as their variance is significantly scaled up. One of these scaling methods is called min-max scaling, here every value is scaled from 0 to 1, removing any bias in favour of the variables with the largest absolute values.

$$\mathbf{X}_{scaled} = \frac{\mathbf{X} - min(\mathbf{X})}{max(\mathbf{X}) - min(\mathbf{X})} \tag{2.2}$$

2.1.2 Normalisation

Another even simpler approach, compensating for varying orders of magnitude and units of measurement in the data is to apply normalisation. This pre-processing approach scales all the data to be centred with unit variance. This is done using the standard score formula (Equation 2.3:

$$Z = \frac{X - \mu}{\sigma} \tag{2.3}$$

Here Z is the standard score, X is the original data value, μ is the mean.

2.1.3 Linear interpolation

To handle the difference in sampling rates leading to uneven numbers of data points, linear interpolation can be applied. Linear interpolation allows for the creation of "synthetic" points in between measured data points. Given sample points (x_0, y_0) and (x_1, y_1), a point (x, y) that lies in between can be calculated. Choosing a value x between x_0 to x_1, the corresponding y value can be calculated as:

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0} \tag{2.4}$$

2.1.4 Moving mean and median

For very noisy data, a way of smoothing the stochastic noise is often desired. A simple and widely used method for smoothing is called a moving average, which essentially replaces each value of a signal y_i, with the average of a moving "window" centred on point y_i. For a window of width k, Equation 2.5 gives the values of y_i.

$$y_i = \frac{1}{k}\Sigma_{j=i-\frac{k}{2}}^{j=i+\frac{k}{2}} y_j \tag{2.5}$$

A similar approach can be used to calculate the moving median, which may be more useful if the data contains radical outliers which are clearly errors that biases the average. This would then lead to a lower bias smoothing than the average that can be highly affected by a single erroneous measurement.

2.1.5 Principal Components Analysis

PCA is an unsupervised learning technique that takes only the input data and performs a dimensionality reduction that constructs new axes that optimises the variance between the observations[9]. This allows for low-dimensional representation of high-dimensional data that still contain "most" of the original information. The first principal component is the normalised linear combination of the features:

$$Z_1 = v_{11}\,\mathbf{X}_1 + v_{21}\mathbf{X}_2 + ... + v_{p1}\mathbf{X}_p \tag{2.6}$$

Normalisation in this case is given as $\Sigma_{j=1}^{p} v_{j1}^2 = 1$. Together these loadings (v) constitute the loadings vector. The data $\mathbf{X}$ also should be normalised[9]. The normalised data matrix is $\mathbf{Z}$ containing the standard scores calculated from $\mathbf{X}$ as previously described. $\mathbf{B}$ has a correlation matrix $\mathbf{C} = \mathbf{Z}^T\mathbf{Z}$. Calculating the eigenvalues and eigenvectors of this correlation matrix will give $\mathbf{CV} = \mathbf{VD}$ where $\mathbf{V}$ and $\mathbf{D}$ are the eigenvectors and eigenvalues respectively. These eigenvectors are the "loadings", the loadings along with the normalised data matrix $\mathbf{B}$ make up the principal component scores $\mathbf{T}$.

$$\mathbf{T} = \mathbf{ZV} \tag{2.7}$$

The eigenvalues λ in $\mathbf{D}$ are equivalent to the variances σ^2[9]. This fact is very useful when deciding how many Principal Components one is going to use as the relative amount of variance explained by a set number of PC's r can be calculated as $\frac{\Sigma_{j=1}^{r}\lambda_j}{\Sigma_{j=1}^{n}\lambda_j}$ where λ is the eigenvalues. To choose the appropriate amount of PC's a scree plot is often used. Plotting eigenvalues in decreasing order where one can look for an "elbow", where the amount of variance explained levels off and the additional benefit of including additional PC's is very low compared to the preceding PC's.

2.2 Regression Methods

In this section the regression methods that will be applied throughout this work will be introduced. The methods in this section share the common features of using a data matrix

X or equivalently a normalised data matrix **Z**, to predict a matrix or vector of responses **Y**. For simplicity in this chapter, **X** will be used even if most of the work is done on normalised data. This $n * p$, **X** data matrix, is used to predict the corresponding y_i value for the sample $n = i$. In this thesis the statistical learning methods that will be applied are:

- **Multiple linear regression** (MLR) using both ordinary least squares, regularisation, stepwise subset selection and interaction terms.
- **Latent variable regressions** such as PLSR with the NIPALS algorithm, as well as PCA pre-processing applied prior to modelling with other methods such as trees, ensembles and MLR.
- Simple **regression trees** with binary splits and simple leaf node models like arithmetic mean of each sample in the node.
- **Ensemble methods** using both gradient boosting (LSBoost) and bootstrap aggregation (bagging) will be presented, applied in this work on regression trees.
- **Support vector regression** (SVR) will be presented as an adaption of the famous support vector machine classification method and used with different kernels.
- **Artificial Neural Networks** (ANN) will first be presented in the simplest approach with a single input layer and single output node which can do simple regression tasks, as well as two modifications that allows the network to create time series models, these are called non-linear input output and non-linear auto-regressive exogenous models.

All the above methods have several hyperparameters that needs to be optimised such as number of learners for ensemble methods or the slack for support vector regression, all hyperparameter optimisation in this work is done using cross validation or Bayesian optimisation. The Levenberg-Marquardt algorithm was used for some initial testing for speed, but since it was not used in the final work it will not be further described in this theory chapter.

2.2.1 Linear Regression

Simple linear regression assumes that there is a linear relationship between the response **Y** and the predictors **X** along with some noise ϵ, which is assumed to be normally distributed with mean zero.

$$\mathbf{Y} = \mathbf{XB} + \epsilon\epsilon\epsilon_{\epsilon(2.8)}$$

With training data we can produce the estimate for **B**, that is $\hat{\mathbf{B}}$. From these an estimated relationship between **Y** and **X** can be found as:

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\mathbf{B}}_{LR} \tag{2.9}$$

The hat symbol denotes that its an estimated coefficient or response rather than a true value. The most common way to train a linear regression model is to minimise a sum of

squared errors cost function, giving rise to the name "least squares regression". This cost function is given as Equation 2.10, where RSS is the residual sum of squares, $\hat{y}_i$ is the prediction of **Y** for sample i, and y_i is the true value of **Y** for sample i.

$$RSS = (\hat{y}_i - y_i)^2 \tag{2.10}$$

$$\hat{\mathbf{B}}_{LR} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \tag{2.11}$$

Regularisation

As protection against overfitting, less complex models than those fit by ordinary least squares might be needed, particularily in high-dimensional feature space. There are two widespread regularisation techniques for high-dimensional feature spaces when doing linear regression, these are the lasso (L1) and ridge (L2) regularisation. Both adapt the ordinary least squares cost function with a penalty term for more complex models as shown in Equations 2.12 (Ridge / L2) and 2.13 (Lasso / L1). Note that, if λ were to be set to 0, the ordinary least squares cost function would remain.

$$cost = \Sigma_{i=n}^{n}(y_i - \Sigma_{j=1}^{p} b_j X_{ij})^2 + \lambda \Sigma_{j=1}^{p} b_i^2 \tag{2.12}$$

$$cost = \Sigma_{i=n}^{n}(y_i - \Sigma_{j=1}^{p} b_j X_{ij})^2 + \lambda \Sigma_{j=1}^{p} |b| \tag{2.13}$$

Both approaches reduces complexity, one significant difference is that the L1 regularisation sets the unimportant predictor coefficients to zero, making it a potent tool for model selection while ridge regression will only shrink coefficients "close" to zero. An illustration of why this is can be seen by plotting the optimum **B** on axes of the coefficients **b1** and **b2**, the lasso regularisation is plotted in red and the ridge regularisation in blue for a given λ, as one can see the intersection with the lasso regularisation sets **b2** to zero while the ridge regularisation makes **b2** small compared to **b1**, but not zero, this is shown in Figure 2.1.To decide on the value of lambda, cross validation is usually used but it can also feature as a hyperparameter in Bayesian optimisation. Even if using Bayesian optimisation on a single regularisation coefficient is unnecessary, since the regression is a low cost function to evaluate in general, as Bayesian optimisation is usually applied for functions that are computationally intensive to evaluate multiple times.

Linear Interactions Regression

The standard MLR regression has one bias term (B_0) and one term corresponding to each dimension of the data ($B_1, B_2...B_n$ corresponding to $X_1, X_2...X_n$). Sometimes there is also interest in combinations of terms, so that interactions between variables are also modelled. In the simplest case, each of the first degree linear terms (B_iX_i) also has one interaction term for each other term in the model, that is: $B_{i,j}X_iX_j$. A illustrative example, making the case for using interaction terms in regressions is the case where modelling fluid flow pumped by a rotating pump through a valve. If the data then includes a measured flow as the response as well as valve opening and pump rotation as predictors, the most significant term might be a combination of the valve opening and the pump rotation.

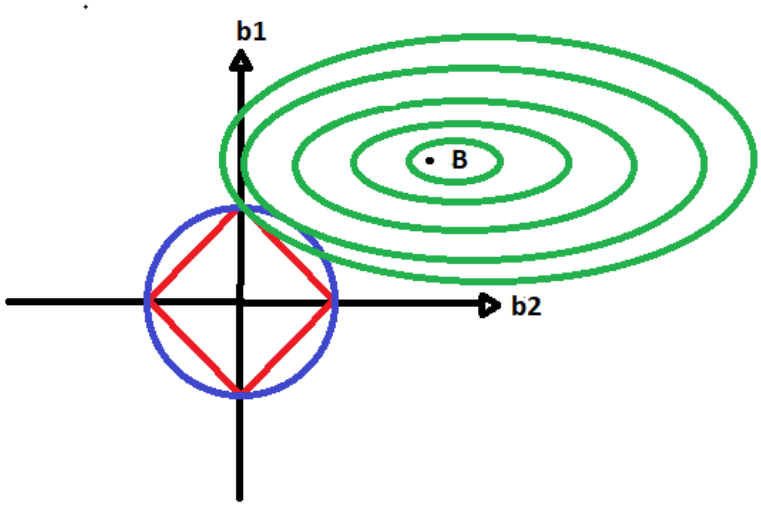


Figure 2.1: Illustration of L1 and L2 regularisation's and the intersections giving the solutions to optimising **B** under L1 / L2 constraints. L1 regularisation is shown in red and the L2 regularisation is shown in blue. As this figure shows the L1 regularisation tends to have intersections setting variables to zero (red and green intersect at $b2 = 0$), while L2 regularisation sets $b2$ to be just "small" compared to $b1$.

Since pumping with a closed valve will not generate flow and opening a valve without any fluid being pumped would also not generate any flow on its own. Thus, the most important term will be a combination of pump rotation and valve opening. Higher order interactions such as $B_{i,j,k}X_i, X_j, X_k$ can also be included but is seldom used due to the exponential increase in model complexity from higher order interactions [10].

Stepwise Linear Regression

Using every available term is often not optimal, then a method for choosing the best subset of models is needed. Stepwise addition of terms to regression models is one such method for best subset selection. This algorithm is based on starting with a initial model, then adding or subtracting terms to search for the best model. There are three common ways to perform this selection: forward, backwards or both. In forwards selection, one starts with a "empty" model, then tests every term for which terms improves model performance the most (common metrics to evaluate model performance here is AIC and Validation error), the best term is picked and added. This is repeated until further additions do not provide a significant improvement. The backwards selection is very similar except that one starts with a full model, including all terms and terms are removed step by step until removing further terms degrades model performance significantly. These methods can be combined, where the algorithm is allowed to both remove and add terms at each step in search of the best model. Significant in this case can either mean statistically significant improvement by a F-test using the following statistic:

$$F = \frac{\frac{RSS_1 - RSS_2}{p_2 - p_1}}{\frac{RSS_2}{n - p_2}} \tag{2.14}$$

RSS is the residual sum of squares, p is the number of parameters in the models and n is the number of samples. This statistic follows an F distribution with $(p_2 - p_1, n - p_2)$ degrees of freedom. The other option for determining what constitutes a significant improvement is to set a minimum level of improvement in either the selection criterion such as the Akaike information criterion or validation MSE [10].

Principal Components Regression

PCR combines the dimension reduction of PCA with the supervised learning technique of regression. PCR changes ordinary least squares (OLS) MLR, instead of using **X** one uses the PC scores **T**. In addition to calculating the optimal regression coefficients **B** one has to find the optimal number of PC's to use. This is often done using a cross validation method (see section 2.3.1) method such as K-fold CV [11].

2.2.2 Partial Least Squares

When one has data with significant co-linearity, the use of OLS MLR is often undesirable. Two predictors are said to be colinear if there exists a linear relationship $X_2 = \lambda_0 + \lambda_1 X_1$. Co-linearity is quite common in industrial process data, in this case study for instance one sees significant co-linearity between inputs and measurements (see Section 4.1). PLS has

been successfully applied for several purposes in the literature[12]. PLS finds the **B** in the regression Equation 2.11 in a supervised way corresponding to latent variables instead of the normal data matrix **X**.

The PLSR method is based on two matricies **X** (dimensions $N*m$) and **Y** (dimensions $N*p$), where N is the number of observations, where p is the dimensions of the dependent variables and m is the dimensions of the independent variables. PLSR selects a number of latent variables that best captures the covariance between **X** and **Y** (i.e instead of finding the latent variables that maximize $Cov(\mathbf{X}, \mathbf{X})$ (as in normal PCA), one finds the latent variables that maximize $Cov(\mathbf{X}, \mathbf{Y})$[12]). This is done using a joint decomposition similar to PCA in the form:

$$\mathbf{X} = \mathbf{T}\mathbf{P}^T \tag{2.15}$$

$$\mathbf{Y} = \mathbf{T}\mathbf{C}^T \tag{2.16}$$

Where **T**, with dimensions $N * r$ is a matrix that is commonly referred to as the "scores", which denote the coordinate in the latent variable space. **C** and **P** are respectively the **X** and **Y** loadings, with dimensions $p*r$ and $m*r$. The rank of the input matrix (**X**) is r, r is bounded by $r \leq m$. The loadings represent the composition of the latent variables in terms of original predictor values.

The NIPALS algorithm

To find these matrices a commonly used algorithm is the non-linear iterative PLS algorithm (see algorithm 1) which calculates the eigenvector matrix **W** (of dimensions $m * r$) and the aforementioned matrices. The matrices **W**, **T**, **P** and **C** are calculated using Equation 2.17a and the regression coefficient itself is calculated using Equations 2.18 and 2.19.[13]

$$\boldsymbol{W} = [\boldsymbol{w}_1, \ \ldots, \ \boldsymbol{w}_r] \tag{2.17a}$$

$$\boldsymbol{T} = [\boldsymbol{t}_1, \ \ldots, \ \boldsymbol{t}_r] \tag{2.17b}$$

$$\boldsymbol{P} = [\boldsymbol{p}_{1_7} \ldots, \boldsymbol{p}_r] \tag{2.17c}$$

$$\boldsymbol{C} = [\boldsymbol{c}_1, \ \ldots, \ \boldsymbol{c}_r] \tag{2.17d}$$

After the matrices are calculated the coefficient matrix $\mathbf{B}_{PLS}$ can be found using Equations 2.18 and 2.19.[13]

$$\tilde{\boldsymbol{X}}^+ = \boldsymbol{W}(\boldsymbol{P}^\mathsf{T}\boldsymbol{W})^{-1}\boldsymbol{T}^\mathsf{T} \tag{2.18}$$

$$\hat{\boldsymbol{B}}_{\text{PLS}} = \tilde{\boldsymbol{X}}^+\boldsymbol{Y} \tag{2.19}$$

Algorithm 1: NIPALS algorithm [13]

Initialise: $\mathbf{X}_1 \leftarrow \mathbf{X}$, $\mathbf{Y}_1 \leftarrow \mathbf{Y}$;
for *d = 1, ... , r* **do**
  $\mathbf{S}_d = \mathbf{X}_d^T \mathbf{Y}_d$;
  $w_d = Eig_{max}[\mathbf{S}_d \mathbf{S}_d^T]$;
  $t_d = \mathbf{X}_d w_d$;
  $c_d = \mathbf{Y}_d^T t_d / t_d^T t_d$;
  $p_d = \mathbf{X}_d^T \mathbf{t}_d / \mathbf{t}_d^T \mathbf{t}_d$;
  $\mathbf{X}_{d+1} = \mathbf{X}_d - t_d p_d^T$, $\mathbf{Y}_{d+1} = \mathbf{Y}_d - t_d c_d^T$;
end
Create matrices **W**, **T**, **P** and **C** using equation 2.17a;
Find B_{PLS} using equation 2.19;

2.2.3 Support Vector Machines

Support vector machine is a statistical method for constructing a non-linear decision boundary that allows classification of data. SVM's are an extension of support vector classifier's. First, support vector classifiers and support vector machines will be discussed. Then, it is shown how these can be modified for building regression models.

Support vector classification

The basis of support vector classification is attempting to create a separating hyperplane in the input data space for discrimination between classes. When the data is not separable, a plane that discriminates the classes perfectly cannot be created. A hyperplane in p-dimensions is given as:

$$\mathbf{b}_0 + \mathbf{b}\,\mathbf{X}^T = 0 \tag{2.20}$$

This separating hyperplane has the property that if $f(\mathbf{x}_i) = \mathbf{b}_0 + \mathbf{b}\,\mathbf{X}_i^T > 0$ then the point given by $\mathbf{X}_i$ lies on one side of the boundary and if $f(\mathbf{x}_i) = \mathbf{b}_0 + \mathbf{b}\mathbf{X}_i^T < 0$ is true it lies on the opposite side. The optimal separating hyperplane for classification is found such that it maximises the distance from the decision boundary to the closest points. This is called the maximal margin classifier and can be found from the following optimisation problem:

$$\max_{b^T} M$$

$$subject\ to: \ \Sigma_{i=1}^{p} b^2 = 1, \quad y_i(b_0 + \mathbf{b}\mathbf{x}_i^T) \geq M \ \ where\ i = 1, 2...n \tag{2.21}$$

Where M is the width of the margin, that is the distance from the hyperplane to the closest sample. This clearly does not work if the data is not separable. Introducing a slack variable ϵ_i and a regularisation parameter C limiting the total amount of slack, modify the method for this case. This gives a new optimisation problem that can be expressed as:

$$\max_{b_0, b_1 b \ldots b_n, \epsilon_1, \epsilon_2 \ldots . \epsilon_n} M$$

$$subject\ to: \ \Sigma_{i=1}^{p} b^2 = 1$$

$$y_i(b_0 + \mathbf{b}\mathbf{x}_i^T) \geq M(1 - \epsilon_i) \ where \ i = 1, 2 \ldots n, \quad \epsilon_i > 0, \quad \Sigma_{i=1}^{n} \epsilon_i \leq C \tag{2.22}$$

Where $\epsilon_i = 0$ means that the sample is on the correct side of the margin, $\epsilon_i > 0$ means that it is on the wrong side of the margin and $\epsilon_i > 1$ means that it is on the wrong side of the hyperplane and will be missclassified. The ability to accept certain missclassifications distinguishes soft margin classification from hard margin classification. Soft margin classification allows the classification of a real, non-separable data set with a linear decision boundary. It can be shown using optimization theory that the support vector classification (SVC) optimization [14] problem has a solution for a test sample $\mathbf{x}^*$:

$$f(\mathbf{x}^*) = b_0 + \Sigma_{i=1}^{n} \alpha_i \langle x^*, x_i \rangle \tag{2.23}$$

Replacing the inner product $\langle x^*, x_i \rangle$ with a kernel allows for non-linear decision boundaries, so that Equations 2.23 can be re-written as:

$$f(\mathbf{x}^*) = b_0 + \Sigma_{i=1}^{p} \alpha_i K(x^*, x_i) \tag{2.24}$$

Two widely used kernels for non-linear SVM are the radial ($K(x_i, x_i) = exp(^{p}_{j=1}(x_{ij} x_{ij})^2)$ and polynomial ($K(x_i, x_i) = (1 + \Sigma_{j=1}^{p} x_{ij} x_{ij})$). An example a radial kernel decision boundary can be seen in Figure 2.2.

Support Vector Regression

For the applications in this project an adapted version of the SVM method will be used. Instead of creating a decision boundary for classification, the method creates a maximal margin separating hyperplane for prediction. One looks for a linear model as with linear regression with a hyperplane on the form [15]:

$$f(x) = \beta x^T + \beta_0 \tag{2.25}$$

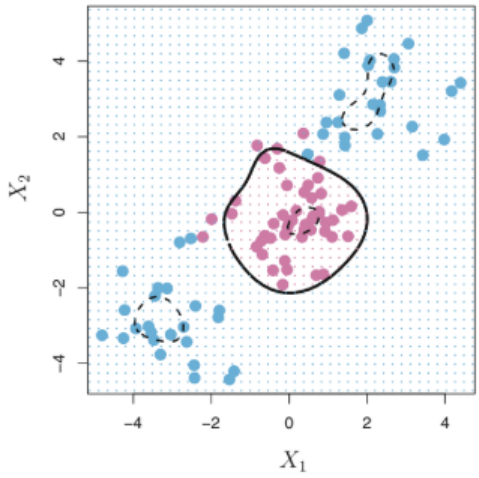
In this case, the estimate $\hat{\beta}$ is found by solving the unconstrained minimisation problem:

$$\min_{\beta, \beta_0} H$$

where H is:

$$H(\beta, \beta_0) = \Sigma_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2} ||\beta||^2 \tag{2.26}$$

This $V(y_i - f(x_i))$ function can take on widely varying shapes, some ignoring errors where the distance from the margin, $|r| < \epsilon$ while applying a linear penalty to errors larger than said tolerance, λ is a regularisation parameter. A famous version of this V function



.

Figure 2.2: Illustration image a radial kernel SVM decision boundary as a solid line and the margins as a dotted line from introduction to statistical learning [1]

was presented by Huber (1964) cited by Hastie et al [14], using quadratic errors for errors smaller than regularisation parameter c and linear for errors greater than c.

$$V_H(r) = \frac{r^2}{2}, |r| \leq c \tag{2.27}$$

$$V_H(r) = c|r| - \frac{c^2}{2} \tag{2.28}$$

This approach gives some significant advantages in handling outliers, quadratic error measures weigh single outliers that are very far off heavily.A single outlier sample could heavily bias the model, particularly with quadratic error, using a linear error for samples far away from the margin creates models more robust to outlier samples.

2.2.4 Regression trees

Regression trees create partitions in the data for either classification or regression purposes. The trees discussed here will be binary trees where every partition is a two ways split decided by a single variable j at a splitting point s. The terminal nodes of such a tree are called leaves, when applied to classification a leaf usually corresponds to a particular class, but for regression a (usually) simple model is applied within the leaf. A common model to use within a leaf is the arithmetic mean of the observations within the leaf, but more complicated models can be used [16].

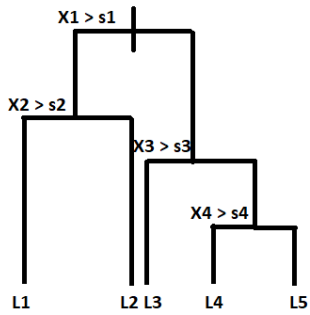


Figure 2.3: Illustration of a how a regression tree might look. Each level of the tree branches, splitting at thresholds where X1,X2,X3 and X4 respectively equal s1, s2, s3 and s4. At the bottom there are leaves L1, L2, L3, L4 and L5. For a regression tree each leaf will correspond to a simply model/estimate of the data points in that leaf, while a classification tree would attach a specific class to each leaf.

2.2.5 Ensemble methods

Ensembles are a powerful statistical tools that can make several weak learners into a single strong learner. In this work two ensemble methods will be used : bootstrap aggregation (commonly referred to as ”bagging”) and gradient boosting (”boosting”). Both take several weak learners and aggregate them into an ensemble method, but in different ways.

Gradient Boosting

Gradient boosting is as based on teaching successive learners to aggregate into a single good learner. The algorithm starts with an initial learner, then starts fitting new learners on the residuals from the previous predictors on the data. Thus each successive learner will compensate for the weakness of its predecessors, until a single strong aggregate predictor has been trained. Each successive learner will have a different weight, which is also a trainable parameter that determines how influential it will be in the prediction. All models can in theory be aggregated with gradient boosting, but the most common are simple linear regressions and tree regressions. However complex model ensembles, such as neural ensembles have gotten some traction recently. The fitting algorithm for gradient boosting as presented by Friedman [17] is given in Algorithm 2, where **F** is the function being approximated, y is the response, $h(x; a)$ is weak learner and ρ is a parameter giving the weight

to the learners being added.

Algorithm 2: LSBoost [17]

$\mathbf{F}_0(x) = \bar{y}$
for *m = 1, ... , M* **do**
  $\tilde{y}_i = y_i - F_{m-1}(\mathbf{x}_i), i = 1, N$
  $(\rho_m, \mathbf{a}_m) = argmin_{a,\rho} \Sigma_{i=1}^{N} [\tilde{y}_i - \rho h(\mathbf{x}_i; \mathbf{a})]^2$
  $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
end

Bootstrap Aggregation

Bootstrapping suggests that estimating the underlying "true" population can be done by sampling from a representative sample. Bootstrapping consists of creating B boostrapped data sets, which are created by drawing N samples with replacement with uniform probability. Having created these B bootstrapped samples a regression tree is grown for each of them. Normally the prediction $\hat{f}(x)$ is just equal to the prediction of the single tree grown from all of the available training data, while the bagged tree prediction is:

$$\hat{f}_{bag}(x) = \frac{1}{B} \Sigma_{b=1}^{B} \hat{f}^{b*}(x) \tag{2.29}$$

That is a averaged prediction of all the bagged trees, counting on the bootstrapped samples to lead to a lower variance of prediction and thus a more powerful prediction.

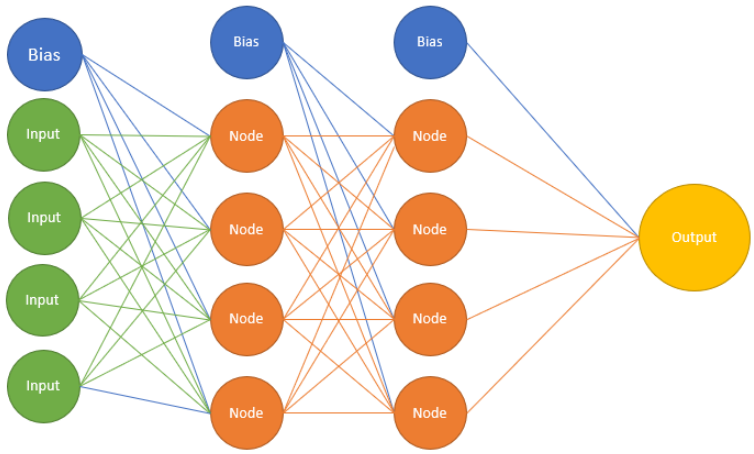


Figure 2.4: An illustration of a general basic network with 4 inputs, 2 hidden layers of width 4, with a single output and bias terms for every non-output layer.

2.2.6 Neural networks

A more modern approach to statistical modelling is the today omnipresent Artificial Neural Network, applicable to marketing, finance, healthcare and basically every other task where classification and prediction from data is useful. Of all the models discussed Neural Networks have the greatest internal variance as a class of models. Neural Networks can be either simple perceptrons or deep networks with up to 175 billion parameters such as DeepMinds GPT-3 natural language model[18]. A good description of what a neural network is given by this quote[19]: "The central idea is to extract linear combinations of the inputs as derived features, and then model the target as a nonlinear function of these features."

Basics of Neural Networks

A basic neural network has some important features: inputs, hidden layer nodes, output nodes, bias nodes, weights and activation functions. Figure 2.4 shows a simple neural network with two hidden layers between the input layer and output layer. The illustrated network is a fully connected feedforward neural network, that is, connections are only forward (left to right) and every node in layer n is connected to every node in layer n+1. Each of these network edges have an associated weight, which is learned in training. Every node also has an associated activation function which transforms the weighted sum of the inputs, into the output passed into the following layer. Several common activation functions exist, some of the most widely used ones are the ReLU and Sigmoid (Equation 2.30 and 2.31).

$$f(x) = x^{+} = max(0, x) \tag{2.30}$$

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.31}$$

Figure 2.4 shows each non-output layer with an associated bias term, the bias terms feed directly to the following layer without any inputs. The bias magnitudes and the network weights that are learned during training are usually found using optimisation algorithms such as gradient descent[20].

NLIO and NARX models

In this work two types of shallow neural networks are used, the non-linear input-output model (NLIO) that takes in a driving (exogenous) time series with lag $x(t)_{(t-k):t}$ to make predictions on a target time series $y(t)$. The second type is the non-linear auto-regressive exogenous (NARX) model which also uses previous values $y(t)_{t=(t-k):t-1}$ in addition to make predictions of the target time series $y(t)$. The order of the lag, k, indicate how many previous values of the time series will be used as inputs. In general the models trained with the auto-regressive property are superior to those without it, but previous values of the target series $y(t)$ might not always be available, requiring the use of only the driving time series as inputs.

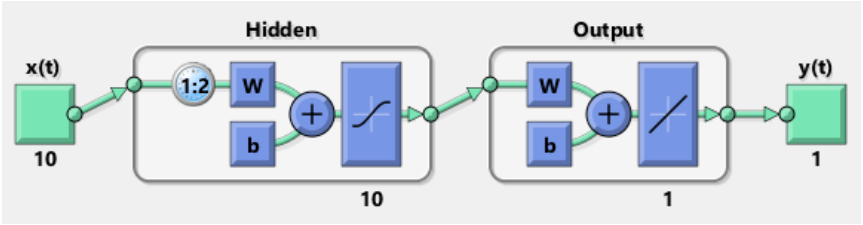


Figure 2.5: Illustration of the structure of a NLIO neural network, with predictors $x_{(t-2):t}(t)$ as inputs (with k'th order lag), a hidden layer using a softmax activation function and an output layer using a linear activation function.

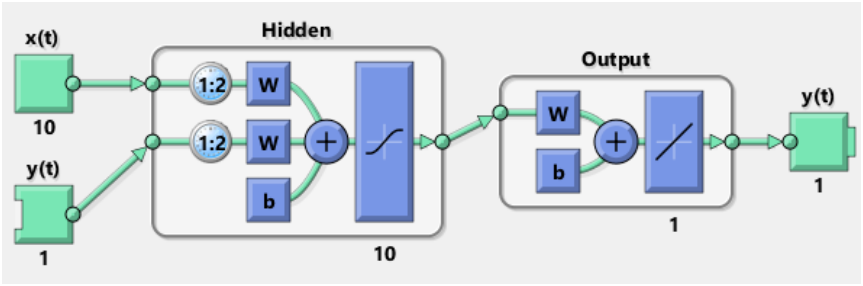


Figure 2.6: Illustration of the structure of a NARX neural network, with predictors and preceding targets, $x_{(t-2):t}(t)$ and $y_{(t-2):t-1}(t)$) as inputs with k'th order lag, a hidden layer. The network uses a softmax activation function in the hidden layer and a output layer using a linear activation function.

2.3 Model Selection and Evaluation

Deciding which data-driven models perform the best in this work is an important step and using appropriate metrics to select the best performing models is a important step in this. What we want is to have a measure of the expected performance on completely new data if the model is deployed. Some of the ways to approach this are described below.

2.3.1 Model Validation

Test Set Validation

The simplest approach to estimating the performance of a model is to only allow the model to see a certain, randomly selected portion of the data during training, then use the model to make predictions on a unseen set that we can call the test set. Often one would want to split the data in three parts: a training set, a validation set and a test set. First the model is trained on the training data. Then performance on the validation data is used to select a model. At last a test is done a independent unseen test set to estimate performance on new unseen data.[21] [20] An illustration of this data split is shown in figure 2.7.

Figure 2.7: Illustrative split of data into training set, validation set and test set.

Cross Validation

The above approach depends on having enough data to train the model without using the test set or the validation set. A way around this problem is using cross validation (CV). K-fold cross validation involves splitting the available data into k ”folds” or partitions of the data, this split is shown in Figure 2.8. All but one of the folds are used for training the data, while the last fold is used for validation. This process of training and validation is repeated k-times so that all k partitions have been used for validation exactly once[21]. The model that has the best average validation performance should then be selected and tested for performance on a test set. Keeping the test set out of the model selection avoids biasing the estimated prediction error[20].

2.3.2 Model Performance

To evaluate which models performs the best and if said model has a good enough performance to be used one needs some metrics that allow quantification of performance of a model. The following metrics can be calculated on the validation hold out set, training set and the independent test set.

	ALL DATA (5-fold cross validation)				
1	Training set	Training set	Training set	Training set	Validation set
2	Training set	Training set	Training set	Validation set	Training set
3	Training set	Training set	Validation set	Training set	Training set
4	Training set	Validation set	Training set	Training set	Training set
5	Validation set	Training set	Training set	Training set	Training set

Figure 2.8: Illustration of 5-fold cross validation, showing 5 iterations each using a different 1/5'th of the data as the validation set.

- **MSE** - Mean squared error, in regression analysis the quality of predictions are usually measured by their distance from the true value. A common metric to use for this is called MSE, of particular interest is the test MSE as this gives an indication of the models performance on currently unseen data. The MSE is calculated as:

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.32}$$

  It is always a positive number with lower errors indicating a higher predictive value. The use of the square term more severely punishes larger misses and removes the issue of errors in positive/negative directions.

- **RMSE** - Root mean squared error of prediction, equivalent to the square root of the MSE, RMSE is used as a validation performance measure in this thesis.

$$RMSE = \sqrt{MSE} \tag{2.33}$$

- $\mathbf{R}^2$ - This metric can be defined in multiple ways, but the idea is to calculate a score of how well the predictors explains the response. For simple linear regression this is just the correlation between the predictor and response, a score as close to 1 as possible is desired.

$$R^2 = 1 - \frac{SSR}{TSS} \tag{2.34}$$

  Where RSS is the regression sum of squares $SSR = \Sigma_i(\hat{y}_i - \bar{y})^2$ and TSS is the total sum of squares $TSS = \Sigma_i(y_i - \bar{y})^2$, $\bar{y}$ is the mean value of y.

2.4 Model optimization

Ordinary least squares coefficients can be easily obtained, since the OLS regresision has an analytical solution. However, for more complex cases, such as neural networks, ensembles

and SVR, hyperparameters such as weights, learning rates, number of leaves, need to be optimised. There is a multitude of algorithms used to solve these machine learning problems, the most basic of which is gradient descent. In addition to gradient descent, I will briefly introduce the algorithm for Bayesian optimisation which was used in this work.

2.4.1 Gradient descent

The basic idea of gradient descent is very simple, one must first define a goal of the machine learning algorithm, and then apply the gradient descent algorithm to try to optimise that goal.

Conventionally in machine learning this is done by introducing a loss function. As this thesis concerns itself with regression problems and not classification problems, the most common loss function is the least squares loss, where $L(f(x_i, \theta), y_i)$ is the squared error of prediction. This is equivalent to minimising the mean square error of prediction which has been used as the main performance indicator in this work. Once such a loss function has been define, the gradient of the loss function in parameter space can be calculated $\nabla_\theta L(f(x, \theta), y)$. With knowledge of the gradient, the parameters θ can be altered to give better predictions by moving θ in the gradient descent direction. This is done by updating the parameters a given amount (the learning rate ϵ), this update will move the parameters towards a lower $L(f(x, \theta), y)$, leading to better predictions. This process continues until a defined stopping criterion is met. Common criteria include a maximum number of iterations, maximum number of seconds training, minimum increase in performance, minimum step distance or targeted performance. The most common version of the gradient descent algorithm is the stochastic gradient descent (SGD), which is given as Algorithm 3. In this case some stochasticity is introduced by computing the gradient on a subset of training samples instead of the whole. The minibatch sampling leads to some issues in terms of convergence if the training rate is held fixed as the gradient may never get close "enough" to zero for the algorithm to be satisfied because of the noise introduced by the sampling. To remedy this, the learning rate ϵ can be limited, if the conditions given in Equations 2.35 and 2.36 SGD will generally converge.[22]

$$\Sigma_{k=1}^{\infty} \epsilon_k = \infty \tag{2.35}$$

$$\Sigma_{k=1}^{\infty} \epsilon_k^2 < \infty \tag{2.36}$$

Algorithm 3: Stochastic Gradient Descent [22]

Require: Learning schedule $\epsilon_1, \epsilon_2...$
Require: Initial parameter θ
k $\leftarrow 1$
while *stopping criterion not met* **do**
  Sample a minibatch of m examples from the training set $x_1, ..., x_m$ with corresponding targets y_i.
  Compute gradient estimate: $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \Sigma_i L(f(x_i, \theta), y_i)$
  Apply update: $\theta = \theta - \epsilon_k \hat{g}$
  k $\leftarrow k + 1$
end

Algorithm 4: Bayesian Optimisation [22]

Place a Gaussian prior on $f(x)$
Observe $f(x)$ at n_0 points according to an initial experimental design.
Set $n = n_0$
while $n \leq N$ **do**
  Update the posterior probability on $f(x)$ using all available samples.
  Let x_n be a maximiser of the acquisition function overr x, where the acquisition function is computed using the current posterior distribution.
  Observe $y_n = f(x_n)$
  $n \leftarrow n + 1$
end
Return a solution: either the point evaluated with the largest value of the objective f, or the point with the largest posterior mean.

2.4.2 Bayesian optimisation

The Bayesian optimisation algorithm is a very powerful and commonly used algorithm for optimising parameters in machine learning models because it efficiently allocates the computational budget when evaluating the target function is expensive, for instance when the target function requires training and testing of a full scale machine learning model. When obtaining the derivative is hard or impossible, making gradient descent infeasible this is very strong algorithm, it is relatively slow compared to the Levenberg–Marquardt algorithm, but provides better generalisation and in this testing better results all around.

In this work Bayesian optimisation was used both in hyperparameter optimisation for statistical methods as well as training of neural networks. The Bayesian optimisation algorithm has two main components, a statistical model of the objective function and an acquisition function for choosing where to sample next. Initially the objective function is sampled with a starting experimental design, where n_o of the total N samples budgeted are spent getting an overview of the objective function. The initial experimental design is often a uniform random sampling. After this, every new point will be sampled intelligently using an acquisition function, in this work this is the expected improvement function.

The expected improvement function uses the existing posterior to look for the x that

will lead to the largest improvement in $f(x^*)$, where x^* is the value of x at the current optimum of $f(x)$. The statistical model is a Gaussian process, which provides a Bayesian posterior probability distribution that describes the potential values for $f(x)$ at point x with the corresponding normal probability distribution at point. Each new observation is used to update the posterior probability distribution. When the budget of function evaluations has been spent, the current x^* is returned (value of x with the highest evaluated $f(x)$), or another search for the highest expected value of $f(x)$ is done and the value of x with the highest posterior mean for $f(x)$ is returned instead. A more detailed view of how this Gaussian process regression can be used to optimise a function can be found in the paper of Frazier (2018)[23], a simplified pseudocode for the algorithm is given as Algorithm 4.

Chapter 3

Simulations

Since erosion is not a fully understood process, representing it with a first principles model has proven difficult. In this thesis, data-driven modelling approaches instead. For testing the capabilities of these approaches, we us in silico and experimental case studies. In this chapter, the models for simulating the degradation of choke valves in gas lifted oil wells due to sand. The model is based on the papers of Krishnamoorthy et al.[6] as well as Verheyleweghen and Jäaschke.[2]

3.1 Gas lifted oil well network

In some cases oil wells do not have sufficient reservoir pressure to lift fluids to the top-side facility. Gas lift, the injection of compressed gas through the annulus, can be used to overcome this. The annulus is the void between the well piping and the casing. The gas injection leads to a reduction in the fluid mixture density. That, in turn reduces the hydrostatic pressure loss in the well. Consequently, the pressure at the well bottom decreases, compensating for the low reservoir pressure.

In Figure 3.1 the different parts of a gas lifted oil well network are shown. The oil flows from a single reservoir into three wells, then, into a manifold before the riser takes the fluid mixture to the surface. However a reservoir does not only produce oil but also gas, water and sand. Maturing fields experience a significant increase over time in sand production, which are approximated by exponential and logistic functions for demonstrative purposes based qualitative evaluation of the shape of sand production rate profiles shown in the literature.[8][7] The model for describing gas lifted wells is given in four parts: The mass balance of different phases, density models, pressure models and flow models. The mass balance of the wells (Eq. 3.1 a-c) and the manifold/riser (Eq. 3.1 d-e) is given by Equations 3.1, the following equations will give the model for a single well, but it is identical for additional wells.

$$\dot{\mathrm{m}}_{ga} = w_{gl} - w_{iv} \tag{3.1a}$$

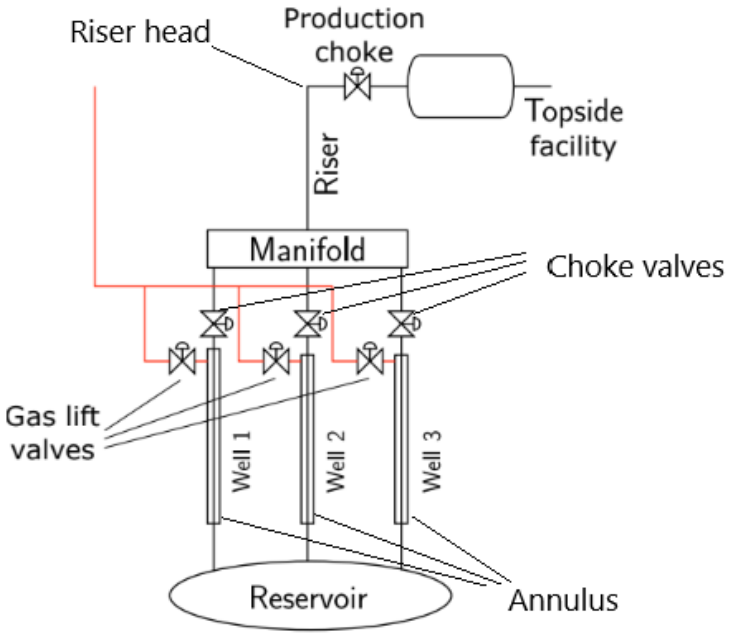


Figure 3.1: Illustration of a gas lifted oil production system with three wells showing how production goes from reservoir to topside facility. Figure adapted from Verheyleweghen and Jäaschke[2].

$$\dot{\mathrm{m}}_{gt} = w_{iv} - w_{pg} + w_{rg} \tag{3.1b}$$

$$\dot{\mathrm{m}}_{ot} = w_{ro} - w_{po} \tag{3.1c}$$

$$\dot{\mathrm{m}}_{rg} = \Sigma(w_{pg} - w_{tg}) \tag{3.1d}$$

$$\dot{\mathrm{m}}_{ro} = \Sigma(w_{po} - w_{to}) \tag{3.1e}$$

Here the $\mathrm{m_{ga}}$ is the mass of the gas annulus, $\mathrm{m_{gt}}$ is the mass of gas in the well tubing, $\mathrm{m_{ot}}$ is the mass of oil in the well tubing, w_{gl} is the gas lift injection rate, w_{iv} is the gas flow from the annulus to the tubing, w_{pg} is the is the flow rate of produced gas, w_{rg} is the flow rate of gas from the reservoir and w_{ro} is the flow rate of oil from the reservoir. The density model is given as ρ_a which is the density of the gas in the annulus and ρ_m which is the density of the fluid mixture in the tubing by Equations 3.2.

$$\rho_a = \frac{M_w p_a}{T_a R} \tag{3.2a}$$

$$\rho_w = \frac{m_{gt} + m_{ot} - \rho_0 L_r A_r}{L_w A_w} \tag{3.2b}$$

Where M_w is the molecular weight of the gas, R is the gas constant, ρ_0 is the density of the oil in the reservoir, T_a is the annulus temperature, L_r is length of the well above the

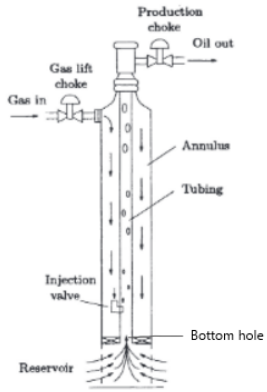


Figure 3.2: Illustration of a single gas lifted well, showing the important components of a gas lifted well: production choke, gas lift choke, annulus, tube, injection valve and the bottom hole.[3].

injection point and L_w is the length below the injection point, A_r is the cross-sectional area above the injection point and A_w is the cross-sectional area below the injection point. The pressure model is given by the equations for the annulus pressure (p_a), wellhead pressure (p_{wh}), well injection point pressure (w_{iv}) and the bottomhole pressure (p_{bh}). These are given by Equations 3.3.

$$p_a = (\frac{T_a R}{V_a M_w} + \frac{g L_a}{L_a A_a}) m_{ga} \tag{3.3a}$$

$$p_{wh} = \frac{T_w R}{M_w} (\frac{m_{gt}}{L_w A_w + L_r A_r - \frac{m_o t}{\rho_0}}) \tag{3.3b}$$

$$p_{wi} = p_{wh} + \frac{g}{A_w L_w} (m_{ot} + m_{gt} - \rho_0 L_r A_r) H_w \tag{3.3c}$$

$$p_{bh} = p_{wi} + \rho_w g H_r \tag{3.3d}$$

In these equations L_a is it is the length of the annulus, A_a is the cross-sectional area of the annulus, T_w is the temperature of the well tubing, H_r is the vertical height of the well tubing below the injection point, H_w is the height of the well tubing above the injection point, g is the gravitational acceleration[6]. In this model we assume that only hydrostatic pressure drop is taken into account. The last part of this model is based on flowrate modelling, Equations 3.4 give the flowrate through the gas lift injection valve (w_{iv}), total flow through the production choke (w_{pc}), produced gas flow rate (w_{pg}), produced oil flow rate (w_{po}), reservoir oil flow (w_{ro}) and reservoir gas flow rate (w_{rg}).

$$w_{iv} = C_{iv} \sqrt{\rho_a max(0, p_{ai} - p_{wi}} \tag{3.4a}$$

$$w_{pc} = C_{pc} \sqrt{\rho_w max(0, p_{wh} - p_m)} \tag{3.4b}$$

$$w_{pg} = \frac{m_{ot}}{m_{gt} + m_{ot}} w_{pc} \tag{3.4c}$$

$$w_{po} = \frac{m_{gt}}{m_{gt} + m_{ot}} w_{pc} \tag{3.4d}$$

$$w_{ro} = PI(p_r - p_{bh}) \tag{3.4e}$$

$$w_{rg} = GOR * w_{ro} \tag{3.4f}$$

Where C_{iv} is the injection valve coefficient, C_{pc} is the production choke valve coefficient, PI is the reservoir productivity index, reservoir pressure is p_r, manifold pressure is p_m and GOR is the gas to oil ratio. In following equations the wells are indexed as n_i. Expressed as a set of differential equations (3.1a-c) and a set of algebraic equations (3.2a - 3.4f) this can be written as shown in equations 3.7.

$$\dot{\mathrm{x}}_i = f_i(x_i, z_i, u_i, p_i) \tag{3.5a}$$

$$g_i(x_i, z_i, u_i, p_i) = 0 \tag{3.5b}$$

The differential states, algebraic states, decision variables and model parameters are given by the following vectors:

$$x_i = [m_{ga_i} \;\; m_{gt_i} \;\; m_{ot_i}]^T \tag{3.6a}$$

$$z_i = [\rho_{a_i} \;\; \rho_{m_i} \;\; p_{a_i} \;\; p_{wi_i} \;\; p_{wh_i} \;\; p_{bh_i} \;\; w_{iv_i} \;\; w_{pc_i} \;\; w_{pg_i} \;\; w_{po_i}]^T \tag{3.6b}$$

$$u_i = [w_{gl_i}]^T \tag{3.6c}$$

$$p_i = [GOR_i]^T \tag{3.6d}$$

Using these vectors the combined model for multiple wells can be expressed as:

$$\dot{\mathbf{x}}_i = f_i(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{p}) \tag{3.7a}$$

$$g_i(\mathbf{x}, \mathbf{z}, \mathbf{u}, \mathbf{p}) = 0 \tag{3.7b}$$

3.2 Erosion model

A model for the erosion in the choke valves can be constructed as shown by Verheyleweghen and Jäschke. This erosion model is in turn based on work by DNV published in 2015[24]. The erosion model presented is given by Equation 3.8. For the remainder of the case study with simulated data we are assuming that this equation actually represents the erosion rate, and thus will investigate statistical learning methods' ability to replicate results from this model.

$$\dot{\mathrm{E}} = \frac{KF(\alpha)U_p^n}{\rho_t A_t} G * C_1 * GF * \dot{\mathrm{m}}_{sand} * C_{unit} \tag{3.8}$$

Here E is the erosion while K, n, $C1$, GF and C_{unit} are empirical constants. Calculation of the parameters A and G is shown in detail in the paper by Verheyleweghen and Jäschke, $F(\alpha)$ is the ductility and U_p^n is the particle impact velocity, calculation can be found in the aforementioned paper. From this equation, one should note that the erosion is a function of the sand production rate.

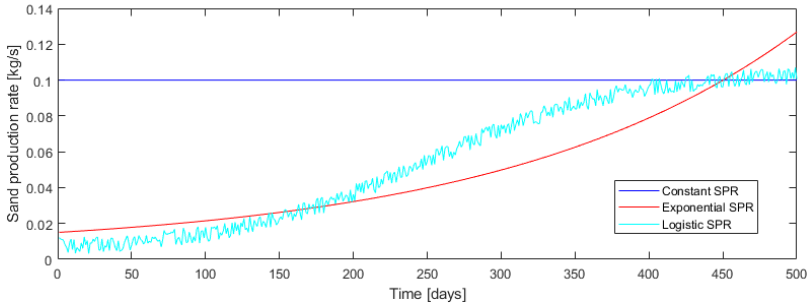


Figure 3.3: Illustration of the three different sand rates for the different case studies, given in Equations 3.9, 3.10 and 3.11. The logistic sand production rate also shows the added noise that was added for that case study.

3.2.1 Sand Production Rate

In real oil well's the sand production rate is not a constant factor, but a dynamically changing property of the maturing field. In general, there is a significant increase in the amount of sand a field outputs over its lifetime, a lot of work has been done in terms of accurately modelling this phenomenon.[7][8] Without delving further into the specifics of these attempts at modelling, this thesis only concerns itself with approximating the functional shape of the sand production profile.

Initial exploration was done on constant sand production data corresponding to Equation 3.9. For the more complex simulated use case we consider two functions that yield a non-linear increase over time as the field matures, the first one is a standard exponential function (Equation 3.10) and the second is the logistic function which exhibits a similar shape to the exponential function initially, but whose rapid increase levels off eventually, providing a similar functional shape as empirical data shown in the paper by Pham.[8]

$$m_{sand}(t) = m_{sand}(0) \tag{3.9}$$

$$m_{sand}(t) = m_{sand}(0) * e^{k*t} \tag{3.10}$$

$$m_{sand}(t) = \frac{L * m_{sand}(0)}{1 + e^{-k*(t-t_0)}} \tag{3.11}$$

3.3 Simulated data

In this section, a brief description of the simulated data sets used for obtaining the data-driven models is given. The section is split into two sections, with Subsection 3.3.1 describing how the simulation is set up for the constant sand production rate, while Subsection 3.3.2 describes how the simulation is adapted for exponential and logistic sand production rate.

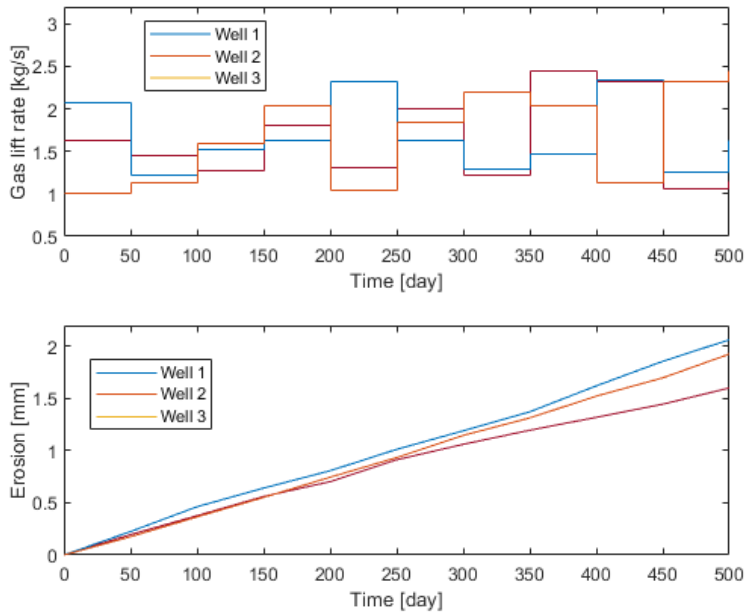


Figure 3.4: Gas lift rate and erosion in mm plotted against time in days for 3 wells for the constant sand production rate study.

3.3.1 Data from constant sand production rate simulations

Simulations were carried out using the model described in Section 3.1. The gas lift rates, which are the system manipulated variables, are varied randomly within the range of $U_{min} + 1 * (U_{max} - U_{min})$ to $U_{min} + 3 * (U_{max} - U_{min})$. A example of this fully known gas lift profile can be seen in Figure 3.4. In addition to the random variations of the input, normally distributed noise was added to the measurements, except for the gas lift rate which is assumed to be fully known. For an overview of the measurements simulated for a single well see Table 4.1. The simulations are run over 500 time steps, which is equal to 500 days, with the gas lift injection changing independently every 50 days for each time series. 200 time series were simulated to be used as training and test data, with an equal split between the two data sets. Validation is done by using holdout and cross validation on the training set, holding out part of that data for model selection.

3.3.2 Data from exponential and logistic sand production rate simulations

The same approach is used for simulating the exponential data with one addition, the sand production rate is allowed to vary. Since the goal of this research is to establish models that can function as a soft sensor without requiring constant expensive measurements to be

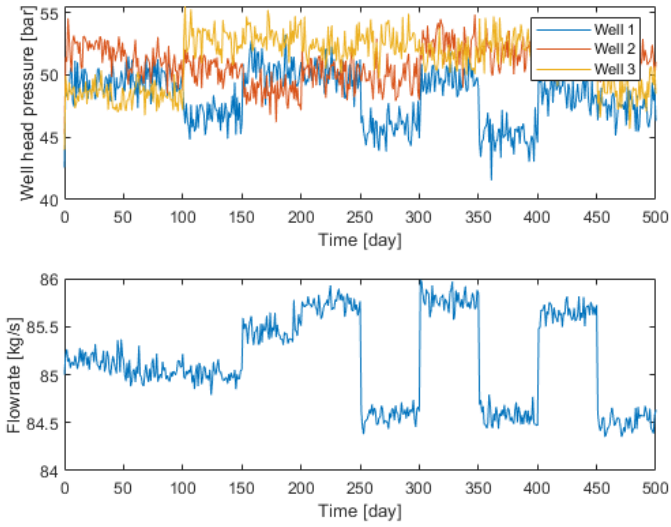


Figure 3.5: Flowrate and well head pressure plotted against time in days for the constant sand production rate case study.

taken, the model can not be allowed to see the constant changes in the sand production rate. To an demonstrate the use case, a sand production sampling interval of 50 days is used for model training here, that is the model will assume the sand production rate to be constant until a new sample is done. This leads to an erosion rate profile similar to the one in Figure 3.6, this profile is the one that the models will try to emulate. An illustration of how different the cumulative erosion is, is shown in Figure 3.7. What this shows very clearly is that an older field, which in general will have a higher sand production rate, has a vastly higher rate of erosion than a younger field. The exponential rate was chosen arbitrarily to demonstrate that models also extend to the less simplified case. The sampling rate was chosen based on industrial experience of how often the sand production is measured, that it is on the order of a month or two. In the results section, the effect of the sampling rate on model performance will be discussed. For the logistic sand production rate simulations, one further adaption was made, the sand rate that was used to simulate the erosion was given some noise, corresponding to true stochastic variance in the sand production of the well. It was established that this did not have a significant impact on the resulting erosion profile, since the stochastic noise was of a modest magnitude and over time, this averages out since the erosion happens over a long period. Nonetheless this establishes further robustness of the analysis as a valid case study for real processes. Another observation that is made from Figure 3.7 is that while the erosion rate is clearly increasing as the sand production increases, at any one time the dominating influence of the gas lift rate is clear which is to be expected from what we have seen in the case study of constant sand production.

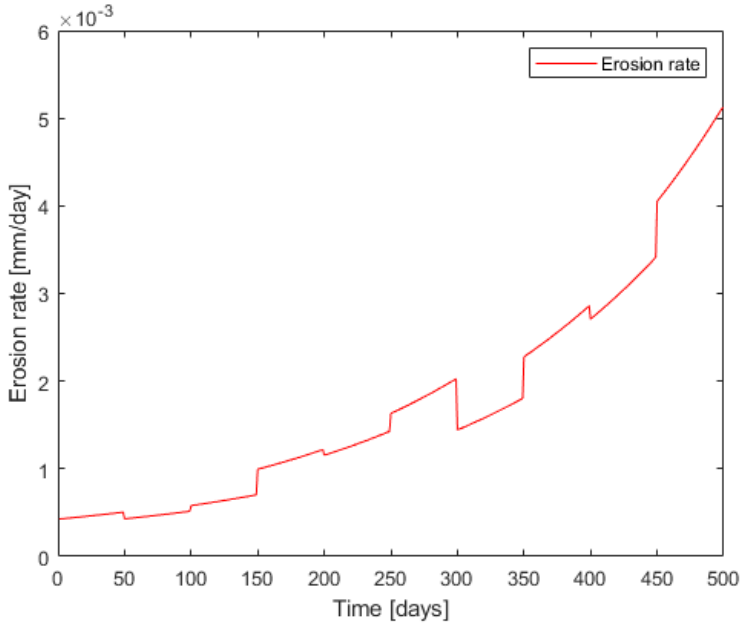


Figure 3.6: Illustration of the erosion rate when simulated with exponential sand production rate. A step profile in the erosion rate every 50 days is still observed due to variations in gas lift rate.

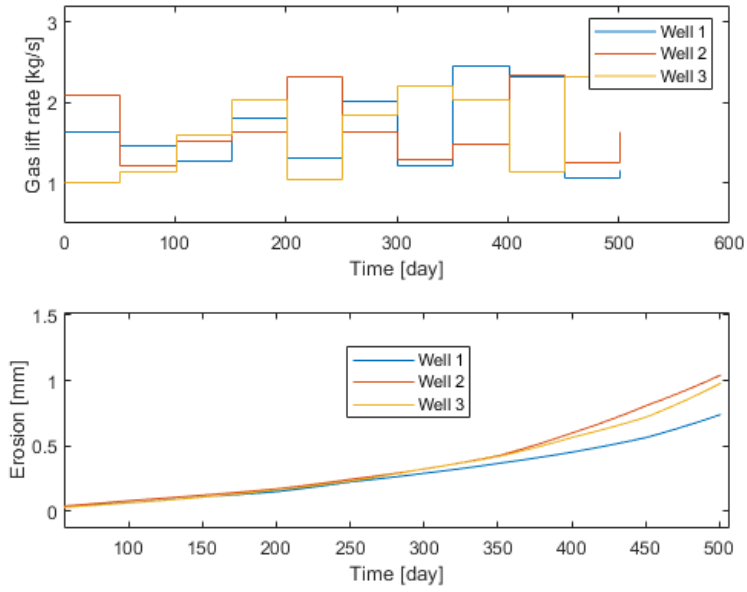


Figure 3.7: Gas lift rate and erosion when simulated using exponential sand production.

Chapter 4

Case study 1: Constant sand production rate

In this chapter, and the following chapter, I will discuss the *in-silico* case studies, where the data-driven machine learning methods discussed in Chapter 2 are applied to data simulated with the model presented in Chapter 3. The code used to generate the data, to train and test models can be found in Appendix A and B respectively. In this section, a brief overview of the initial case study will be given in terms of the simulated data. Then the preprocessing and regression models will be presented, the regression models are presented in roughly increasing order of complexity from stepwise selected linear regression to neural networks. Several of the methods tested here stem from the specialisation project work done this spring, while some new methods like Neural Networks and Regularised Regression are new additions. Then, in the next chapter, the two case studies with exponential and logistic sand production rates are discussed. In terms of model performance, training time, complexity as well explainability and interpretability. Because throughout all case studies, applying PCA has a detrimental impact, not every model was tested using PCA.

The simulated data consists of 200 time series of 501 points with a total of 100200 points in the data. Each time series is generated independently using a separate instance of the simulation code. The data was divided evenly into a test set and a training set. All models are optimised and trained using holdout validation, with a hold out set of 30% of the training data. The trained models are then tested on the independent test set. When working with this simulated data we are in a very data-rich environment, as such one can afford to both use holdout validation for the model selection and optimisation as well as an independent test set. Testing on an independent test set will give the most accurate estimate of true predictive power. At each time step in the time series, several process variables are recorded, in addition to the erosion, these are given in Table 4.1. The final data sets then contain $9 * 50100$ predictors and $1 * 50100$ responses.

Table 4.1: The predictors and response used for regression modelling of the simulated data with constant sand production rate.

Number	Symbol	Name
Predictor 1	p_a	Annulus pressure
Predictor 2	p_{wh}	Well head pressure
Predictor 3	WHO	Well head oil production rate
Predictor 4	WHG	Well head gas production rate
Predictor 5	p_{rh}	Riser head pressure
Predictor 6	p_m	Manifold pressure
Predictor 7	RHO	Riser head total oil production rate
Predictor 8	RHG	Riser head total gas production rate
Predictor 9	GLR	Gas lift rate
Response	ER	Erosion rate

4.1 Exploratory analysis of simulated data with constant sand production rate

With the simulated data at hand I explore the correlations in the data to see if regression modelling is likely to be a good approach. It is quite clear in Figure 3.4 that the erosion is behaving like a linear phenomena (as is expected from the simulations when $\dot{\mathrm{m}}_{sand}$ is constant). Thus, models that are good at fitting linear phenomena will probably perform well. Another easy observation when looking at the rate of change of the erosion is that the trend of the erosion profile seems to be strongly correlated

with the gas lift rate as well as the flow rate (see figures 3.5 and 3.4). To illustrate these correlations more clearly a full correlation plot of all 10 variables has been included in Figure 4.2. We observe several strong correlations between covariates, of particular interest are the gas lift, annulus pressure and the various flowrates (riser head and well head production rates) to the erosion rate. These strong correlations form a solid basis for regression modelling.

4.2 Pre-processing

Before this data was used for modelling, some pre-processing was needed. The predictor variables (see Table 4.1) were all subjected to normalisation (see section 2.1.2), so that all variables has a standard deviation of one as well as a mean of zero. As some form of latent variable method was going to be used, this was necessary to avoid unreasonable results from variables with differing units of measurement. Instead of using the direct erosion measurement the erosion rate was used as input data. This is necessary as predictions are made at snapshots in time from the given process measurements at $t = t_k$. The cumulative erosion has significant auto-correlations, as expected there is a very high correlation between total erosion at $t = k$ and $t = k + 1$. In order to apply methods not designed for time series analysis the erosion rate is used instead, with an accurate model of the erosion rate for every moment of elapsed time, numeric integration can be used to compute the

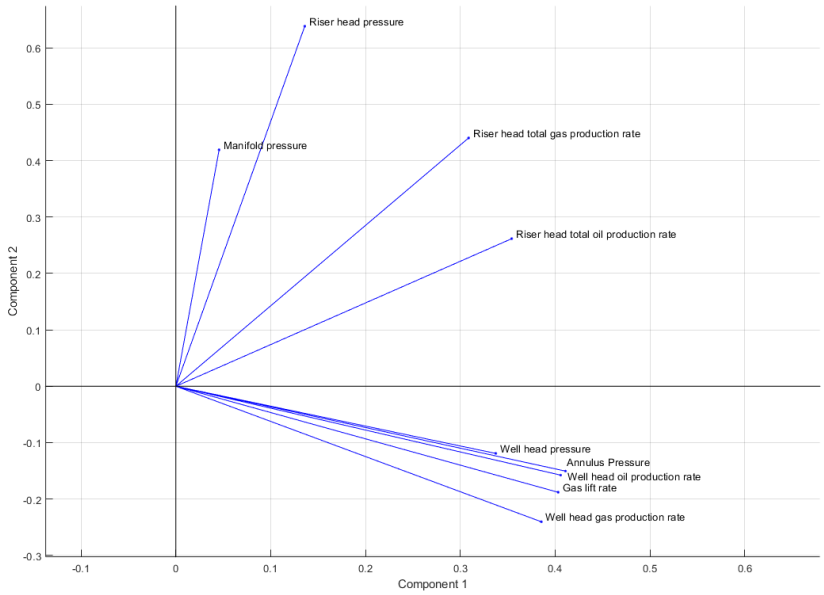


Figure 4.1: Biplot (first 2 PC's) of PCA carried out using the 9 predictors used for regression models for each well: Annulus Pressure, Well head pressure, Well head oil production rate, Well head gas production rate, Riser head pressure, Manifold pressure, Riser head total oil production rate, Riser head total gas production rate and Gas lift rate.

total erosion. For the third case study, a brief look will be taken at using a time-series approach to model the cumulative erosion using a neural network called NARX (Non-linear auto-regressive) with exogenous inputs as described in Chapter 2.

4.3 Regression results

In this section a brief overview of the performance will be given for each of the tested regression methods: linear regression, PLSR, regression trees, support vector regression and neural regression networks. The discussion of these results is held brief as most of the work relating to the first case study was done in the specialisation project conducted this spring as well as the task of modelling a linear phenomena being a quite trivial task. A summary of these results is given in Section 4.4.

4.3.1 Linear regression

For the testing of linear regression models both a latent variable approach using PCA preprocessing as well as a normalised data input was tried. For the PCA data the best approach proved to be a normal MLR model with a test set MSE (calculated from performance on 100 time series) of 0.0166 which is a quite strong performance. For using the raw data, a

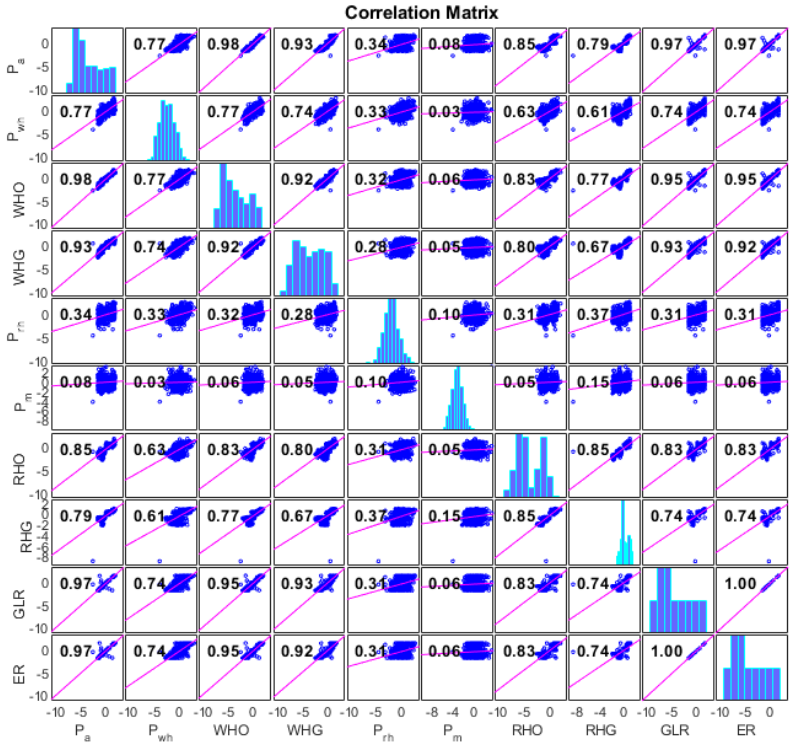


Figure 4.2: Correlation plot of simulated variables for a single well. Showing as expected extremely strong correlations between certain variables and the the erosion rate, like wellhead gas production rate and the gas lift rate.

stepwise selected linear model with interactions gave the best result with a test set MSE 0.0096. As stated previously, since the choke valve erosion here is a linear phenomena, it is expected that linear regression will perform this well. Regularised linear regression was also tested with similar success, in this case L2 or ridge regularisation was used.

4.3.2 Partial Least Squares Regression

Since partial least squares regression is based making latent variables with the strongest correlations to the response, it is expected to perform well when the predictors importance cannot be inferred from knowledge about the system. For the constant sand production rate data, PLSR had the strongest performance of all methods, having a MSE of 0.0076. This is probably due to being able to correctly pick out the latent variable that explained almost all the variance in the response and base its predictions on that. When PLSR is among the top performing models, it is a very strong choice for implementation in industrial applications. Because it gives a thorough understanding of how it makes its predictions based on the weights given to each latent variable as well as the composition of said latent variables and how much of the target variance they explain.

4.3.3 Regression trees

When first inspecting the shape of the erosion rate it seems to follow a step wise shape corresponding to the control input (the gas lift rate). This is not surprising, but it provides a strong argument for trying a tree based model as tree's naturally fit step-wise data very nicely, since each "level" can correspond to one or more leaf nodes. A coarse regression tree performing well on this is a quite expected as the data is divided into approximate levels corresponding to different gas lift rates. Since these are discrete with significant intervals even a coarse model is able to perform very well, producing groups of leaf nodes corresponding to the different gas lift rates. The optimised tree had a test set MSE of 0.0125. When using PCA at 95% explained variance, an ensemble bagged trees perform the best among the tree models. This is not unexpected as the PCA data is significantly more noisy than certain parts of the direct simulated data like the gas lift rate and ensemble methods are more accurate for noisy data. The reason that the tree models suffer so much from PCA pre-processing may be that the gas lift rate gets confounded with other variables, making it less accurate when splits have to be performed on PC1 instead.

4.3.4 Support Vector Regression

Support vector regression also yielded very strong results and had comparable performance to stepwise linear regression and optimised ensemble regression. The best performing kernel was a radial basis function, the Gaussian kernel. This is a very flexible basis which allows the fitting of most function shapes, even highly non-linear ones. The performance of the SVR model was in the upper tier of methods for both raw data with a MSE of 0.0100 and with PCA a 0.0164.

Table 4.2: Model Performance, in terms of mean square error of prediction for several statistical learning techniques when applied to data simulated with the constant sand production rate given in Equation 3.9 in Chapter 3.

Case Study 1: Constant SPR		
Method	**W/o PCA**	**W/ PCA**
Stepwise linear	0.0096	0.0166
Partial Least Squares Regression	0.0076	N/A
Optimised tree	0.0124	0.0244
Optimised ensemble (LSBoost/Bag)	0.0100	0.0165
Gaussian SVR	0.0100	0.0164
ANN Regression	0.0090	N/A

4.3.5 ANN Regression

The ANN Regression model performed quite well, but had some more instability in its predictions than other models suggesting that it did not depend on the exact same underlying variables. The regression delivered a test set MSE of 0.090 on the constant sand production rate data. By plotting of the ANN regression output, one can see there is some noise, but that it almost perfectly captures the step profile. Due to the blackbox nature of the neural network predictions, ascertaining exactly why it has moderate residuals (i.e prediction errors) seemingly suddenly is very difficult. The errors are high compared to the other methods for constant data but not very high on an absolute scale with the erosion rate varying from approximately -1.5 to 1.5 when normalised, meaning in absolute terms the error is only 0.3% of the total range it predicts in.

4.4 Summary

The performance result of every method that was tested on the constant data can be seen in Table 4.2, where it is easy to see that predicting the results from the constant sand production rate is a trivial task as first thought with mean square errors of prediction of around 0.01 on normalised data without PCA. A somewhat worse performance was observed within the data that was subjected to PCA pre-processing, there are a couple of possible explanations for this, first PCA does fundamentally remove information from the data, even at 95% variance. At these low error levels this could explain some of it. Second, it was observed in the linear regressions that using interaction terms was preferable, PCA obscures interactions to some degree by confounding variables that might interact with each other into shared PC's which makes the models unable to capture interactions. The strongest model for constant sand production rate was partial least squares regression with a test set MSE of 0.076.

Chapter 5

Case studies 2 & 3: Exponential & Logistic sand production rate

?? Case studies two and three are similar to the first case study, changing the shape of the sand production profile and, for Case Study 3, adding some noise in the underlying sand production rate. Since the sand production rate is not a fully known variable and as such a sampling interval will use used. As both of these case studies are very similar, the results for both case study 2 & 3 will be presented alongside each other to avoid unnecessary repetition. The negligible impact of adding stochastic noise to the underlying true sand production can be attributed to erosion being a long term process and thus the noise is averaged out by the law of large numbers. The data sets used for these case studies are almost exactly the same as the one in the first case study where the only difference here is the addition of one covariate, the sampled sand production rate, sampled at the set interval. While not strictly pre-processing, the sampling rate of the sand production rate has a very significant impact on the model performance, this is to be expected as the sand production rate is an important predictor and more accurate data for it should thus improve model performance. This section will first do a brief exploratory data analysis of the two data sets looking at correlations between variables as well as PLSR loadings, then proceed to examine the results for each of the methods tested, the figures of the predicted vs true response in this case are reduced in size so that case studies two and three can be viewed side by side. Finally, in this section, an attempt at modelling the cumulative data directly with a time series approach instead of modelling the erosion rate will be made using an auto-regressive NARX neural network model.

Table 5.1: The predictors and response used for regression modelling of the simulated data with exponential and logistic sand production rate. These are the same covariates as in the first case study, with the addition of the sand production rate.

Number	Symbol	Name
Predictor 1	m_{sand}	Sand production rate
Predictor 2	p_a	Annulus pressure
Predictor 3	p_{wh}	Well head pressure
Predictor 4	WHO	Well head oil production rate
Predictor 5	WHG	Well head gas production rate
Predictor 6	p_{rh}	Riser head pressure
Predictor 7	p_m	Manifold pressure
Predictor 8	RHO	Riser head total oil production rate
Predictor 9	RHG	Riser head total gas production rate
Predictor 10	GLR	Gas lift rate
Response	ER	Erosion rate

5.1 Exploratory analysis of simulated data with changing sand production

The exploratory data analysis in this section will be done with respect to the correlations within the exponential and logistic sand production data sets. I will also examine the PLS loadings, this analysis gives some insight as to how much of the variance in the response is explained by each latent variable with a given composition of underlying predictors.

The correlation plots for the exponential and logistic sand production rate data sets are shown in Figure 5.2 and 5.3 respectively.

It is known from Case Study 1 that the total production rate (i.e fluid flow) was the main driver of the erosion, which in turn is controlled by the gas lift injection rate. The highly correlated well head production rate and well gas lift injection rates are expected, since the gas lift is the driver of the production rate. In the new data sets where the sand production rate is allowed to vary, the sand production has the strongest correlation to the response with 0.90 (exponential) and 0.97 (logistic) correlation coefficients respectively, these are strong correlations, but also expected since sand is what erodes the choke valve in the first place.

The difference between the case studies can be attributed to the sampling rate (note that the correlation is calculated between true erosion and *sampled* sand production rate). Since the exponential sand production rate function has a much steeper gradient when time increases and it becomes more affected by the low frequency of sampling. On the other hand, the logistic sand production rate functions tapers of its growth and the sampled values is thus closer to the true values even towards both ends. Even if the sand production has the strongest correlation, the gas lift rate and other variables also have quite significant correlations. This is expected, the first case study shows that total fluid flow is main driver of erosion while sand production is constant.

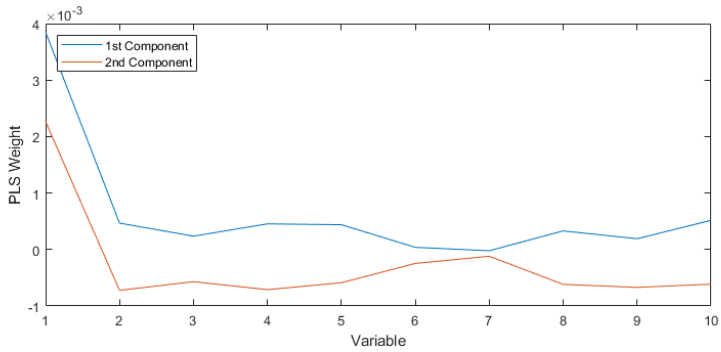


Figure 5.1: Loadings plot of the three PLS components explaining the largest amount of variance in the correlation plot

5.1.1 PLS analysis of data

PLS will be used for exploratory analysis of the simulated data by looking at the relevant loadings and explained variance to investigate which variables explain the most variance in the response variable. The PLS components turned out very similar for both exponential and logistic sand production data, loadings for the logistic data, PLS components are shown in Figure 5.1. The two first components explain almost all the variance in the response, with the first component explaining 80% of the variance and the second explaining 16%. The sand production rate is the dominant variable in both components, but to a significantly larger degree in the first PLS component.

Since these components are chosen based on explained y variance (i.e response / erosion rate), one would expect explain a significant amount of the variance in x (i.e the state of the system with process measurements like pressures, production rates, gas lift etc.). The first component explains only 20% of the variance in x while the second component explains 40%. Although complementary to the results from the correlation plot (Figures 5.2 and 5.3), the insights from identifying the components is useful for identifying the variables driving the response.

In terms of model performance PLSR did poorly, being by a fairly wide margin the worst of the tested methods, as can be seen in Tables 5.2 and 5.3, therefore the predictive results will not be further discussed.

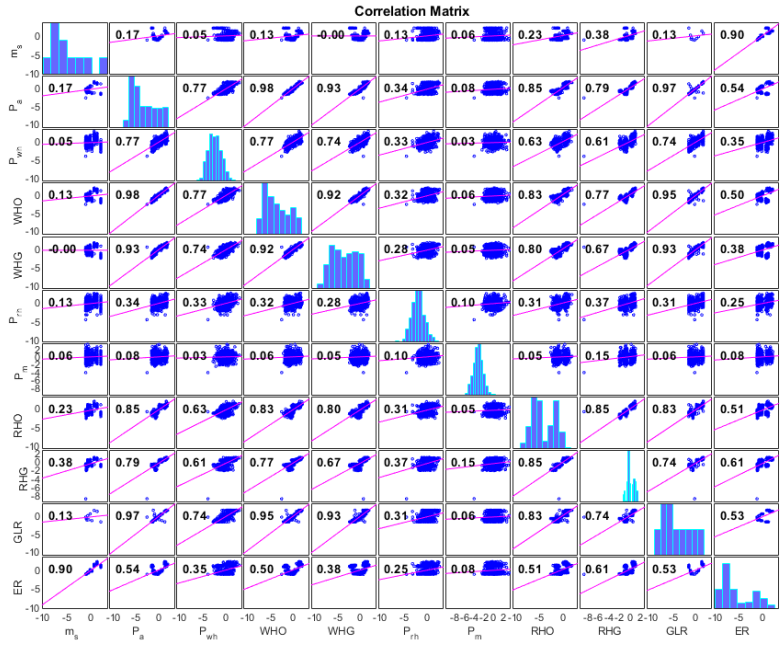


Figure 5.2: Correlation plot of the simulated data with a **exponential** sand production rate. These are the same regression variables given in Table 5.1

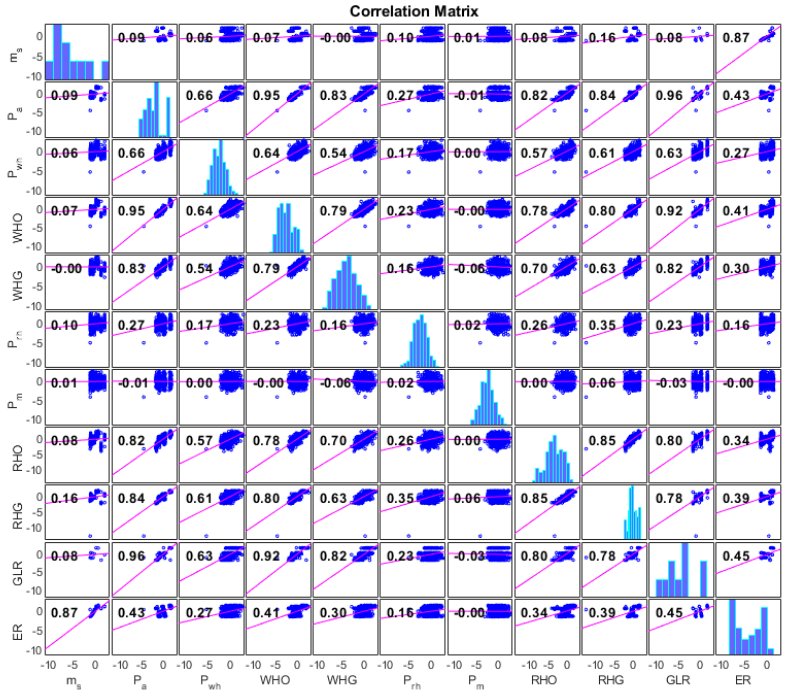


Figure 5.3: Correlation plot of the simulated data with a **logistic** sand production rate. These are the same regression variables given in Table 5.1

5.2 Regression results

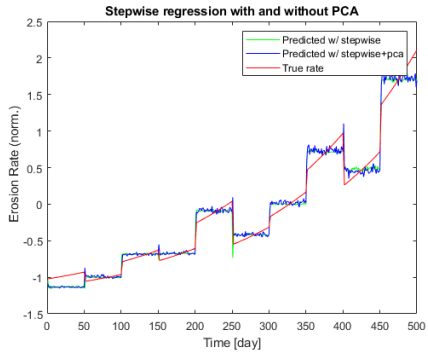
In this section, the same methods that were used for Case Study 1 will be tested on the more complex data from case studies 2 & 3. The methods that will be presented are as follows, multiple linear regression, regression trees, ensemble methods, support vector regression and NLIO neural network. The performance of the methods are shown in Tables 5.2 and 5.3. The added stochastic noise added for Case Study 3, did not prevent strong predictions from being made, but may have had some impact on the accuracy of the models. This was observed when certain models struggled more with the logistic profile despite it being less steep at its steepest.

5.2.1 Linear regression

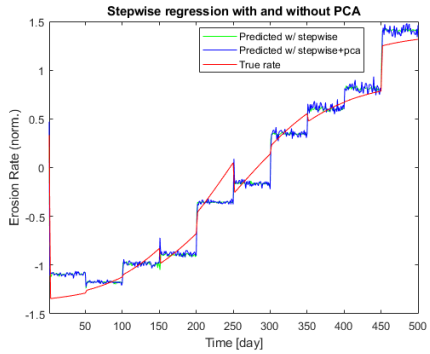
As in the first case study, the MLR models that were allowed to use interaction terms performed the best. Stepwise iterative selection was used for selection of predictors, plots of the predictions made can be seen in Figure 5.4. In case study 2, with exponential sand production rate, the test MSE was 0.0191 and 0.0182 with and without PCA pre-processing respectively. In case study 3, with logistic sand production rate, the test MSE was 0.0245 and 0.0241 with and without PCA pre-processing respectively. Relative to the other methods, linear regression performed worse on non-constant sand production rate data, while it was among the best methods for the constant data. Since these methods are better suited to approximate linear phenomena, the approximations of an erosion rate that increases over time as a non-stationary process were significantly worse. Observing both the predictions made in Case Study 2 and Case Study 3, one can start to discern part of the cause for why the performance on the logistic sand production data is worse than for the exponential. In the exponential data, at each step the prediction is at values that are about the average across the sampling period, this is possible since the sand production is sampled accurately. While what we observe for case study 3 is that it frequently misses this mean value within each sampling period, that is because the sampling method only samples at a single time point, and with the presence of noise the sampling may not be as representative as when the underlying sand rate can be sampled without the presence of noise.

The results when applying PCA compared to not using it is similar to the first case study, that is similar but slightly worse. This again could be explained by loss of information when converting the data to PC's (95% explained variance was used), as well as potentially confounding interaction effects.

Using L2 regularisation does not improve model performance, but instead made it significantly worse with a MSE of 0.0354. On the other hand, the regularised regression has better generalisability as regularisation is less prone to overfitting, which gives better generalisability. The regularisation was optimised using Bayesian optimisation on the λ coefficient and whether to use ridge or lasso regression. Note that stepwise selection is computationally heavier than regressions not based on evaluating every possible model at several steps compared to a single optimisation step with a L2 regularised linear regression. The training time is thus a significant advantage for ridge regression versus stepwise selection.



(a) Case study 2: exponential sand production rate.



(b) Case study 3: logistic sand production rate.

Figure 5.4: Plots of multiple linear regression model predictions with and without PCA pre-processing for case studies 2 & 3. In case study 2, with exponential sand production rate, the test MSE was 0.0191 and 0.0182 with and without PCA pre-processing respectively. In case study 3, with logistic sand production rate, the test MSE was 0.0245 and 0.0241 with and without PCA pre-processing respectively.

5.2.2 Regression trees

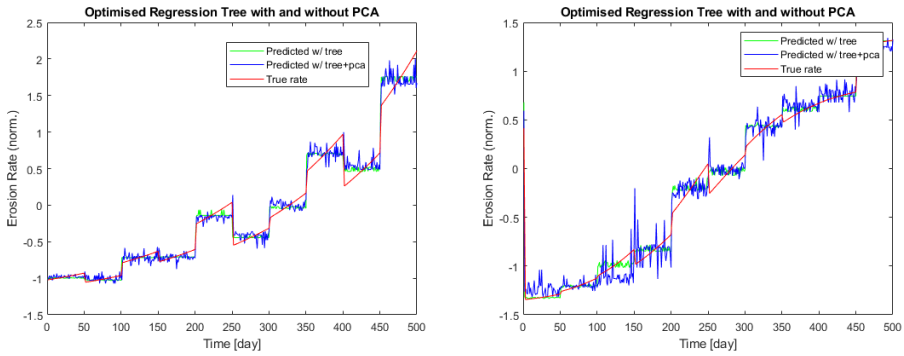
Regression trees are a very flexible method in terms of the functional shape that they can emulate. But have little to no value when predicting outside the range of the known data. The predictions made by an optimised regression tree using Bayesian optimisation can be seen in Figure 5.5. The model performance without PCA pre-processing was quite good at 0.0173 and 0.0112 for the second and third case studies respectively. With PCA pre-processing the model performance was slightly worse for case study 2 at 0.0212 and for case study 3 at 0.0216. The Bayesian optimisation algorithm used to optimise the hyperparameters found that the optimal binary tree has a minimum leaf size of 68 with a total of 1121 nodes in case study 3. This is a very large optimal tree for something that essentially contains 10 ”levels” corresponding to the shifts in gas lift injection rate changing every 50 days. This means that a the predictions can be quite noisy due to the sheer amount of different possible leaf nodes.

Regression trees have several strong points, strong predictive power in terms of performance, semi-transparent, method since you can trace along the tree to understand exactly how a given prediction is made, and very quick to train. But trade-off that regression trees have very poor extrapolation capacity, if one tries to predict something that it has not seen before it will make potentially very erroneous predictions.

When comparing regression trees performance on Case Studies 2 & 3 compared to stepwise linear regression, a significant accuracy advantage is obtained from using regression trees on Case Study 3, while the performance on Case Study 2 was relatively similar. This is attributed to the noise in the sand production data and the way in which regression trees make their predictions. For MLR, the least squares algorithm will assign a direct coefficient to the sand production, making the erosion directly proportional. While the regression tree is built up using binary splits, as such the prediction will follow the same

path if the sand production is within a given range given by the split and small deviations in the sampled values from the true values will thus not bias predictions significantly.

There is a very strong argument for choosing regression trees in this case because of the high transparency, simple structure, short training times at the cost of only a minor reduction in accuracy when compared with Gaussian SVR and bagged trees. Both of which are only slightly better in the exponential sand production rate case and, in the logistic sand production rate case only the bagged ensemble performed better.



(a) Case study 2: exponential sand production rate.

(b) Case study 3: logistic sand production rate.

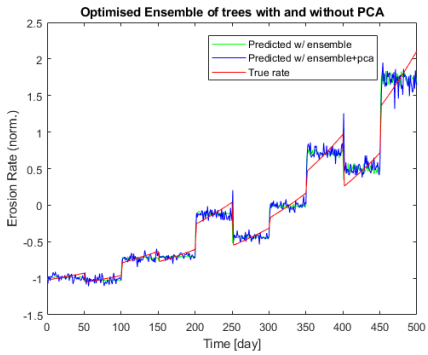
Figure 5.5: Plots of predictions from regression trees optimised with Bayesian optimisation, with and without PCA pre-processing for case studies 2 & 3. In case study 2, with exponential sand production rate, the test MSE was 0.0212 and 0.0173 with and without PCA pre-processing respectively. In case study 3, with logistic sand production rate, the test MSE was 0.0216 and 0.0112 with and without PCA pre-processing respectively.

5.2.3 Ensemble methods

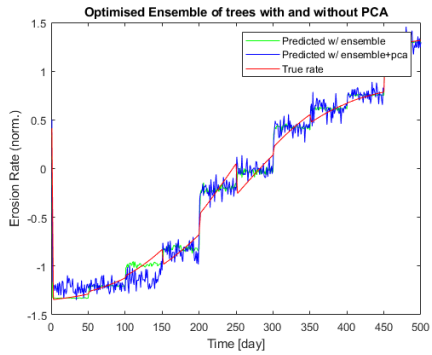
To improve performance of regression trees, several trees can be aggregated. As observed in the results, the optimised ensemble had a higher level of accuracy than the optimised single regression tree for all cases. The optimised ensemble used bootstrap aggregation for both case studies 2 & 3, this was chosen by Bayesian optimisation along with the other hyperparameters such as number of learners (408 trees), weights for averaging and the parameters of the individual trees. Variance in the predictors was introduced by training them each on a different subset of the training data as described in Chapter 2. For both Case Study 2 & 3, a improvement in performance is observed when applying an ensemble approach instead of a single regression tree. In Case Study 2, the performance without PCA improved from $0.0173$to 0.0158 and with PCA from $0.0212$to 0.0193. For Case Study 3, the performance without PCA improved from 0.0112 to 0.0105 and with PCA from 0.0216 to 0.0175. The ensemble, itself being made up of trees have the same strengths as regression trees when compared to MLR based methods in terms of robustness against noise in the sample data.

However, this improvement in accuracy comes at the cost of a significant reduction in model transparency. Predictions using a single tree can be trusted. When using several

smaller trees, the predictions from each of them needs to be traced and taken a weighted sum off to understand exactly how the prediction is made. The component trees are still binary regression trees and as such, the method is not black box. But inference about the underlying phenomena is harder than with normal regression trees. In cases where the performance is the same as that of an ordinary regression tree, the single regression tree should be preferred for increased transparency and ease of training and for use in a decision support system for decision makers. If only the exactness of the prediction is of interest, the ensemble is a very strong method for fitting functions with a wide variety of shapes and has many degrees of freedom to achieve this. In comparison to other methods that are good at complex function shapes, such as neural networks, it is quick to train and more transparent. There is also the benefit of increased robustness and lower variance of predictions, which can be seen when comparing the plots of the optimised trees in Figure 5.5 with the plots of the optimised ensembles of trees in Figure 5.6.



(a) Case study 2: exponential sand production rate.



(b) Case study 3: logistic sand production rate.

Figure 5.6: Plots of predictions from an ensemble with hyperparameters and the bagged ensemble optimised with Bayesian optimisation, with and without PCA pre-processing for case studies 2 & 3. In case study 2, with exponential sand production rate, the test MSE was 0.0193 and 0.0158 with and without PCA pre-processing respectively. In case study 3, with logistic sand production rate, the test MSE was 0.0175 and 0.0105 with and without PCA pre-processing respectively.

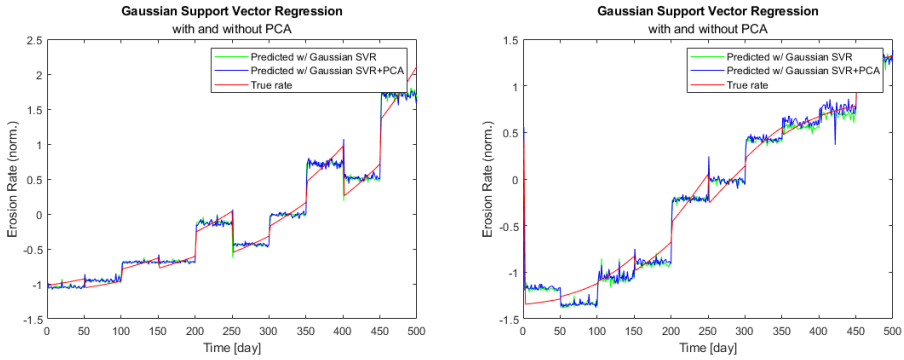
5.2.4 Support vector regression

Support vector regressions have several advantages in terms of generalisability and robustness, with very low variance of predictions in general. This can easily be seen by comparing plots of the SVR predictions (Figure 5.7) to the plots of the predictions from regression trees (Figure 5.5) and ensembles (Figure 5.6). The performance in terms of accuracy was also very good for both case studies, with a MSE of 0.0167 and 0.0173 without PCA for case studies 2 & 3 respectively. PCA proved to be detrimental to the performance, with test MSE of 0.0171 and 0.0177.

As with MLR, SVR is competitive with the tree based methods in terms of model accuracy for Case Study 2, but drops of relative to the tree based methods when noise is introduced, this is attributed to the same cause as with MLR. SVR produces a hyperplane,

containing weights for each variable in the same fashion as MLR does. Thus, a small change due to noise in the sampled data will necessarily lead to a change in the prediction as the prediction is directly proportional to the measured sand production rate.

One of the main advantages in using support vector regression over other methods is if we expect that the real data might have more outliers than the simulated data. The support vector methodology protects against outliers influencing the model unduly, as only the set of support vectors will define the predicting hyperplane. For the applications in the *in-silico* studies, we have data without too many outliers and the dimensionality is not excessively high with 10 predictors and one response. Thus, the full benefit of the high-dimensional tolerance of SVR is not needed, nor is the robustness versus outliers in the training response data points.



(a) Case study 2: exponential sand production rate.

(b) Case study 3: logistic sand production rate.

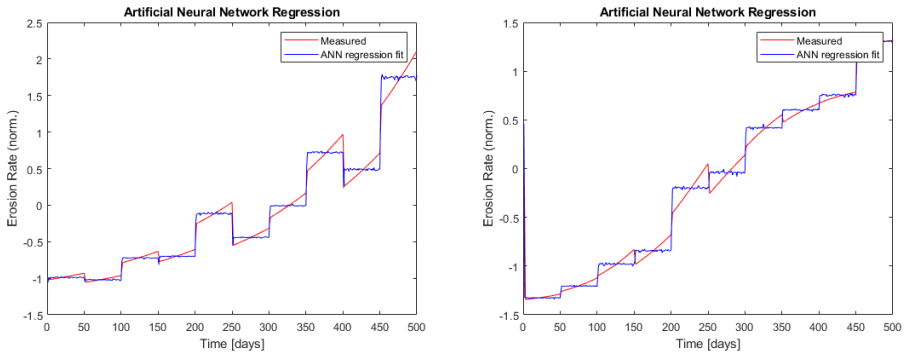
Figure 5.7: Plots of predictions from a Gaussian support vector regression, with and without PCA pre-processing for case studies 2 & 3. In case study 2, with exponential sand production rate, the test MSE was 0.0171 and 0.0167 with and without PCA pre-processing respectively. In case study 3, with logistic sand production rate, the test MSE was 0.0177 and 0.0173 with and without PCA pre-processing respectively.

5.2.5 ANN Regression

The most advanced method discussed yet is the artificial neural network. The trained neural net has two hidden layers each with 15 nodes. In general it should be expected to be more flexible to fit functions that are non-linear. To be able to capture non-linearities the activation function is a sigmoid function. ANN regression performs quite well being the most accurate predictor for the Case Study 2 data and only falling behind the regression tree and ensemble methods for the Case Study 3 data with MSE's of 0.0150 and 0.0102 for the case studies respectively.

The drop in performance from Case Study 2 to Case Study 3 can be attributed to the presence of noise, as with stepwise linear regression and support vector regression, a weight for each variable into the network will be learned. This means that any change in the model input from noise will cause a change in the prediction, this in contrast to regression tree based models (and ensembles of them).

Adjusting the size of the network did not improve test performance. Increasing network size yielded a small decrease in training MSE but also a decrease in the performance on the independent test set, this increase can be attributed to overfitting the data. As expected ANN's have the best performances when the function to be emulated becomes more complex, but due to the importance of the sand production to erosion, the sensitivity to noise in this measurements becomes quite high as the sand production is heavily weighed from the input layer, having a detrimental effect on performance in Case Study 3. This improved performance of course comes with the cost that the model does not give very strong insights into how its predictions are made. Another drawback is the requirement of large amounts of training data as even quite small, shallow networks such as these have a large amount of parameters that need to be trained compared to simpler methods such as MLR.



(a) Case study 2: exponential sand production rate.

(b) Case study 3: logistic sand production rate.

Figure 5.8: Plots of predictions ANN regression, for case studies 2 & 3. In case study 2, with exponential sand production rate, the test MSE was 0.0102 and in case study 3, with logistic sand production rate, the test MSE was 0.0150.

5.2.6 Summary

Overall the models performed very well in case studies 2 & 3 as well, the results for Case Study 2 are shown in Table 5.2. The results for Case Study 3, including results with a shortened sampling rate, which will be motivated in the next section, are shown in Table 5.3. A comparison of the performances of all of the methods across all three case studies is shown in Figure 5.9.

Stepwise regression and support vector regression performed better in case study 2 than in case study 3, while the opposite was true for the ensemble and tree based methods. This is attributed to tree based methods being less sensitive to the noise in the sampled sand production values as a range of sand values can lead to the same binary split decisions and thus same prediction. This is not the case for MLR, SVR and ANN regression as all of these methods directly relate input values through weights to an output, and as such a change in the sampled sand production will always affect the predictions, all else being equal, thus making them more prone to noisy predictions from small perturbations in the

Table 5.2: Model Performance, in terms of mean square error of prediction for several statistical learning techniques when applied to data simulated with the exponential sand production rate given in Equation 3.10 in Chapter 3.

Case Study 2: Exponential SPR		
Method	**W/o PCA**	**W/ PCA**
Stepwise linear	0.0182	0.0191
Optimised tree	0.0173	0.0212
Optimised ensemble	0.0158	0.0193
Gaussian SVR	0.0167	0.0171
ANN Regression	0.0138	N/A
PLSR	0.1840	N/A

observations. The regularised regression performed poorly compared to the other methods, but might have the benefit of being more robust against overfitting in real cases. PCA-pre processing was still in general detrimental to performance in all cases, this is likely due to confounding of interaction effects in the data and loss of information, since the feature space was not of very high dimensionality application of PCA-pre processing is deemed unnecessary.

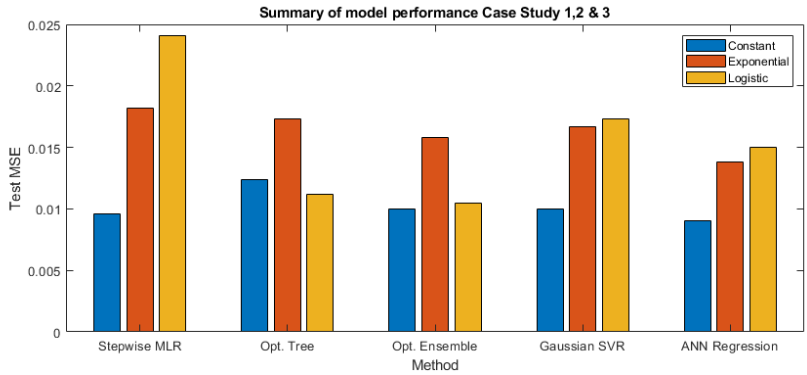


Figure 5.9: Bar chart of the performance of each of the methods for all three case studies (without PCA pre-processing). The methods shown are those that are primarily discussed, PLSR is left out as it was not viable for the non-constant case studies.

Table 5.3: Model Performance, in terms of mean square error of prediction for several statistical learning techniques when applied to data simulated with the logistic sand production rate given in Equation 3.11 in Chapter 3. Reducing the sampling rate of the sand production rate is also tested, reducing the sampling rate from every 50 days to every 14 days.

Case Study 3: Logistic SPR	Sampling rate: 50		Sampling rate: 14
Method	**W/o PCA**	**W/ PCA**	**W/o PCA**
Stepwise linear	0.0241	0.0245	0.0043
Optimised tree	0.0112	0.0216	0.0037
Optimised ensemble	0.0105	0.0175	0.0032
Gaussian SVR	0.0173	0.0177	0.0090
ANN Regression	0.0150	N/A	0.0032
PLSR	0.0765	N/A	0.0583

5.3 Improving predictions by modelling sand production rate

Looking at table 5.3, we observe that the reduced sampling interval, makes predictions significantly more accurate. This is expected given the results of the exploratory data analysis, and the strong correlation between the erosion rate and the sand production rate. Increasing sampling frequency is not always feasible, but there are other methods that can yield more accurate data for the sand production. Using domain knowledge for *feature engineering* is one such approach[25], in this subsection a linear model of the sand rate will be used to improve model input.

5.3.1 Linearly modelling sand production

It is known that sand production rates tend to change, and that the change from one period to another is generally quite similar. Using this knowledge, we make the assumption that the sand production profile increases linearly from each sample point, at the same rate as it did between the two most recent sample points. Then the sand production rate some time Δt after the previous sample point k is given by Equation 5.1.

$$m_{sand}(t) = m_{sand}(k-1) + \frac{m_{sand}(k) - m_{sand}(k-T)}{T}\Delta t \tag{5.1}$$

Where T is the sampling period (in this case study, 50 days), making $m_{sand}(k)$ the last known known sample and $m_{sand}(k-T)$ the one preceding it. Figure 5.10 shows the true sand production rate as well as the values when held constant between samples and using this modelling approach between samples.

Using this linear model instead of assuming the sand production stays constant during the sampling window, increases the accuracy of the model feature by an order of magnitude in terms of MSE, when compared to the true sand production rate. A ten fold increase in feature accuracy, by making a simple and reasonable assumption about the profile, without any additional resources spent on sampling, will be very beneficial to the model performance. Figure 5.10 shows the results of the linearly modelled sand production rate, the true sand production rate and the sand production variable when assumed constant between samples, using the sand production profile from Case Study 3.

5.3.2 Results using modelled sand production rate

The results from using this *feature engineering* method to improve predictions are shown in Table 5.4. What is immediately obvious is that predictions are vastly more accurate when this more accurate, engineered, sand production rate is used as model input. The most accurate model using this approach was the ensemble (bagged) approach, its performance was increased, for Case Study 3, from 0.0105 to 0.0030 which is over a three fold increase in accuracy. Similar, and greater improvements in model accuracy is seen across all methods tested, which is not unexpected given the importance of the sand production to the erosion of the choke valve.

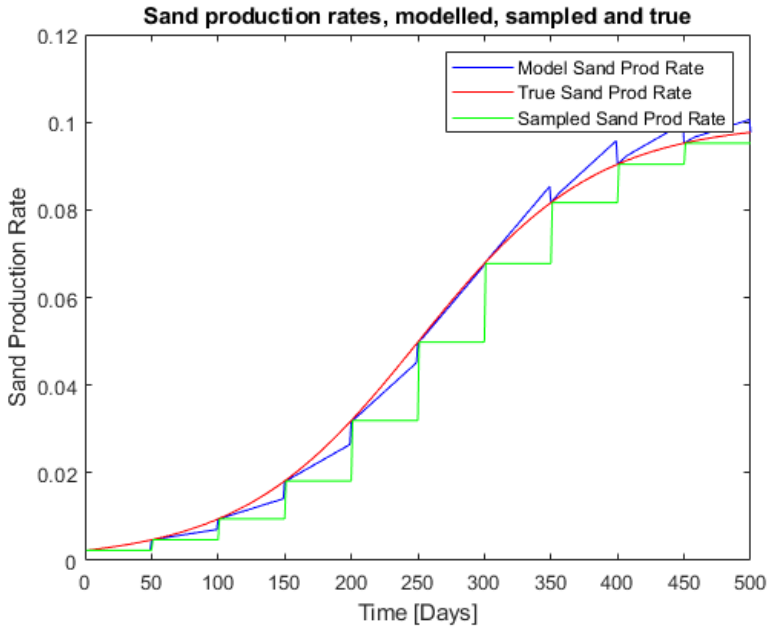


Figure 5.10: This figure shows plots of the true sand production rate from case study 2, the result of using the simple modelling approach as well as the results when held constant at sample values with a 50 day sampling rate.

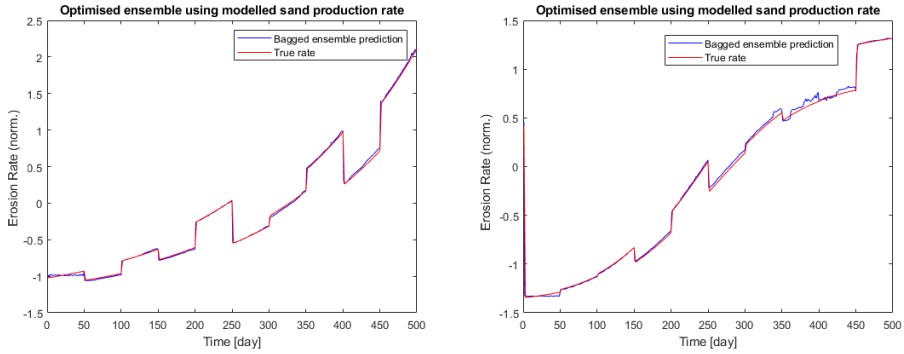
Table 5.4: Model Performance, in terms of mean square error of prediction for several statistical learning techniques when applied to data simulated with the exponential sand production rate given in Equation 3.10 in Chapter 3. No PCA pre-processing was applied in these tests.

Performance using modelled sand production rate		
Method	**Case study 2**	**Case study 3**
Stepwise linear	0.0019	0.0038
Ridge regression	0.0290	0.0169
Optimised tree	0.0039	0.0038
Gaussian SVR	0.0037	0.0062
Optimised ensemble	0.0012	0.0029
ANN Regression	0.0013	0.0030

One thing to note is that using the modelled sand production rate, gives the best results for Case study 2, this is attributed to the exponential function not having an inflection point ($\frac{d^2}{dx^2}f(x) = 0$). Thus the linear approximations don't end up with the significant overshoots that are observed in the logistic profile. If the exponential profile were extended further into the future, thus making the gradient even steeper, larger residuals between the modelled and true sand production rate would be seen there as well.

Observing the different profiles for each of the methods, it is clear to see that the hyperplane methods (SVR and MLR) produce low variance predictions but with some bias. Tree predictions have a fair amount of noise, but this is vastly improved by using an ensemble approach, bagging in this case. Neural regression networks are able to provide quite low variance and bias predictions, performing about as accurately as an ensemble of trees. Plots of prediction versus true profile, for the most accurate method which was ensembles for both case studies, are shown in Figure 5.11.

No tests were carried out using PCA pre-processing as this was shown unnecessary for this data in the previous studies of these Case Studies.



(a) Case study 2: exponential sand production rate.

(b) Case study 3: logistic sand production rate.

Figure 5.11: Plots of predictions made using the modelled sand production data, showing the best method for case studies 2 & 3, the bagged ensemble was the most accurate model for predictions in these cases.

5.4 Modelling cumulative erosion with a NARX model

Finally, using the auto-regressive time series approach, I show the results when modelling the cumulative erosion directly. For this section, I will only consider Case Study 3, but similar results are obtained for the first and second case studies. As has been stated previously, it is not practical with traditional time series approaches such as AR, MA or ARMA models, as they are designed for stationary time-series. For non-linear sand production rates, the cumulative erosion will not only be non-stationary, it will also be somewhat difficult to make stationary using simple approaches. For non-stationary time series an ARIMA model (auto-regressive integrated moving average), but this is not explored in this thesis.

There are ways to turn exponential series into linear ones by for instance, combining a log transform, which will turn it into a linearly increasing time series and then transforming the output y_t into $z_t = y_t - y_{t-1}$. But this would for instance not work for a logistic profile. Thus, for modelling the cumulative erosion as a time series, a model that deals well with non-linearity is required. Here, I use the NARX model, since it is able to make strong predictions directly on non-linear time series data.[26][27] As discussed in Chapter 2, the NARX neural network can operate in two different configurations, one called "open loop" and one called "closed loop". Open loop is generally not very interesting when making predictions about the future as it requires knowledge of targets (true y values), which in general is not available. The closed loop prediction changes the auto-regressive feedback from the true target values to the networks prior predictions, allowing for forecasts multiple time steps ahead.

The amount of lag needed for accurate predictions are indicated (but not necessarily known with certainty) by looking at the partial auto-correlation function (PACF). Figure 5.12 shows the PACF's for the cumulative erosion. As expected for this phenomena, the only information that should be needed to predict total erosion is how much the system is already eroded as well as the parameters with information about how fast the component is eroding. How fast the erosion was up to this point (i.e erosion rate), and the shape of the profile up to that point should in theory not matter for how eroded it will be at the next time step and this is what is seen in the PACF plot.

5.4.1 Cumulative model results

As stated above, the only lag order with a significant partial auto-correlation was the first order lag, which in turn had a very strong auto-correlation, very close to 1. Because of the data rich situation with a lot of available data to use from simulations, a network is chosen with a width of the hidden layers of 20 neurons. The lag order was set to only include first order lags, as those were the only ones with significant partial auto-correlations, this yielded a network with very good performance. The network was tested for prediction 250 days ahead, the results of this test can be seen in Figure 5.13. In this test the loop was kept open the first 200 days and then closed and predictions made using only the exogenous inputs as well as the feedback of its own predictions. The result was that after 250 days of closed loop predictions the error was 0.0392, which means that for a prediction very far in the future the model was off by about 1.6% of the total erosion that happened in this time, the units here are quite meaningless as the data is normalised, but the relative size of the

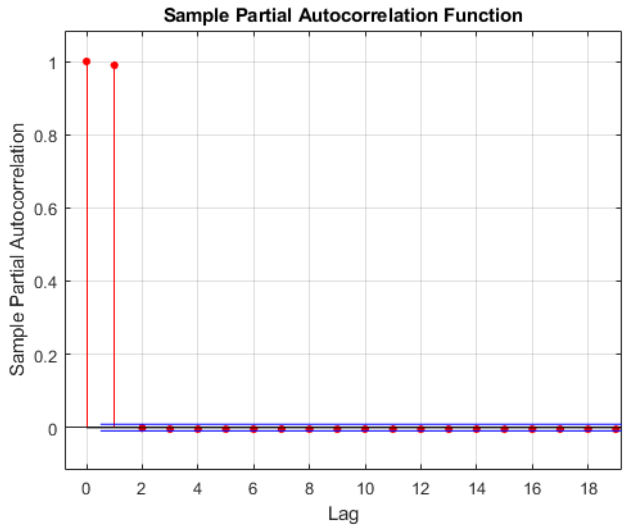


Figure 5.12: Sample partial auto-correlation function for the cumulative erosion arising from case study 3, this shows as expected that the only significant partial auto-correlation for $t = t_k$ is $t = t_{k-1}$. Since this is a cumulative process each step only adds to the previous without any further information in the preceding time steps. What we observe is that the first order lag has a correlation very close to 1, which is as expected, the 0th order lag is the correlation of y_t to y_t and must of course be 1. This serves as justification for setting the lag order of the NARX model to 1 as that is the only statistically significant lag order.

error is interesting and very promising for this application. Even though the model was kept in open loop for 200 time steps in this case, this is not necessary since the model only uses a single lagged value of the cumulative erosion.

5.5 Summary of results for simulated data

As known from Case Study 1, constant sand production erosion is a simple problem using simulations. For the non-linear sand production rate profiles, the problem was complicated both by the non-linear shape of the function that is to be learned and the lack of accurate information about the sand production. In Case Study 3, this was further complicated by the addition of noise to the underlying sand production. Results were still somewhat accurate, but with a significant loss in accuracy whenever the sand production was changing quickly, that is, after a lot of time has passed in the exponential profile and around the inflection point of the logistic profile. To compensate for this, feature engineering with a linear model of the sand production is used successfully to drastically improve accuracy of models. Finally, a NARX neural network is applied to the cumulative erosion time series, successfully predicting 250 days ahead, missing the true value by only 1.6%.

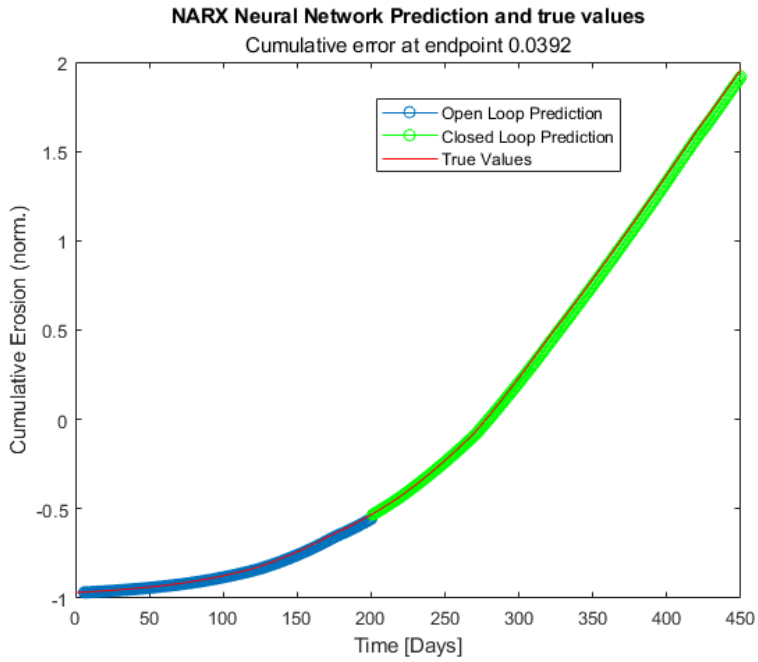


Figure 5.13: Figure illustrating the 450 day long test of a NARX neural network for predicting, the blue markers show the predictions made in open loop format with true values being fed back to the model. The green markers show the closed loop predictions where the model is fed only the exogenous input and feeding back its own predictions. The red line shows the true cumulative erosion. After 250 days in closed loop, the networks prediction is only off by 0.0392 while the erosion has been 2.5 (normalised and thus arbitrary units) in the same time frame, meaning over 250 days the total error is only about 1.6%.

Chapter 6

Experiments

In this chapter, the setup and design of experiments will be discussed for real world testing of the soft sensor approach to equipment degradation. The work was carried out over several months in laboratory facilities at the Department of Chemical Engineering, with multiple phases of tuning and testing before arriving at the final experimental procedure. The sections of this chapter will describe an overview of the experimental rig, a quick description of the fluid mechanics used to justify usage of differential pressure as a stand in for erosion and finally a description of the experimental data, including brief descriptions of how the approach has changed during the months of experimental work.

6.1 Overview of the rig

The experimental rig is designed to emulate the behaviour of a subsea gas lifted oil well network with the same layout as the simulated case study with three gas lifted wells. The system is constructed with one pump as the driving force of the flow. Then three controllers immediately after the pump regulate fluid flow to each well. The wells are represented by three paralell pipelines, each one containing an injection valve and an erosion chamber. The air valves are used for injecting gas into the system to obtain the artificial lifting effect. The erosion chamber is where the eroding sample, discussed later, is placed.

Then the fluid passes through a sand trap and then a riser to a ”topside” chamber. Finally the fluid is recirculated through a large tank. All gas flows and liquid flows are measured as well as the pressure at the top of the riser and the differential pressure over the erosion chamber. A schematic of the rig can be seen in Figure 6.2.

The rig is constructed to facilitate multiple experiments for control, optimisation and accelerated remaining useful life testing. In this work only experiments to measure the erosion based degradation of probes are performed. Because of the long time scales of erosion using standard subsea materials such as steel, the rig was set up to use 3D-printed PVA (polyvinyl alcohol) probes. In addition to the 3D printed probes, custom built probe holders were also built to accommodate them, a probe in its holder during an experiment can be seen in Figure 6.3a.

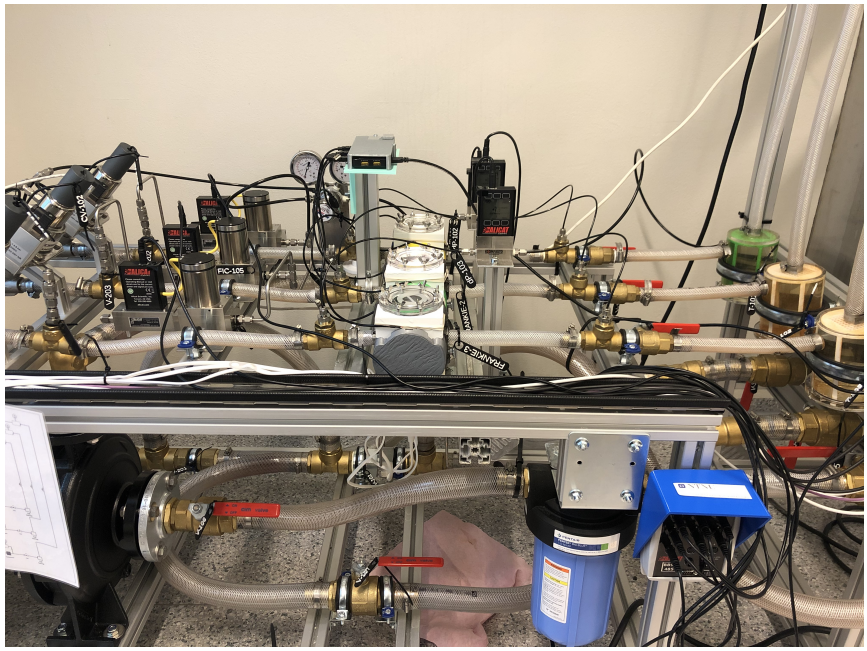


Figure 6.1: Experimental rig. The pump, controllers and air supply are visible on the left hand side, the erosion chambers in the center, and the sand traps at the bottom of the riser to the right.

6.1.1 Measuring erosion

The main goal of this experimental setup is to perform accelerated remaining useful life testing, in this case what we desire is to accelerate the erosion of a probe. The geometry and materials of these probes as well as the holders of the probes went through several refining iterations. Initially, a more complex geometry was tested with a inner part attempting a somewhat faithful reproduction of the geometry of a choke valve. The first generation of probes were made with a quite brittle polymer material which lead to, rather than a somewhat continuous and smooth erosion as real components experience, a very uneven degradation, dominated by breaks and fractures of large blocks. To try to compensate for this issue, PVA was used instead, which yielded a far smoother degradation. An illustration of the PVA probes before and after some degradation can be seen in Figure 6.3.

Initially, the experiments were designed to record images of the probes with two cameras every 30 seconds. From these images, we tried to calculate the visible area of the probe as a measure of how much of the probe was remaining. Since the probe was multi-layered with an internal structure this proved far too difficult in terms of getting good and reliable measurements. In order to simplify this process, we altered the probe geometry to a simple ”wall” shape that was attached to a probe holder. Even with multiple improvements to image quality (for instance, lighting and colouring of the inside chamber) calculating the visible area in a reliable way proved a non-trivial task, an illustration of the data after some pre-processing in the early stages can be seen in figure 6.4.

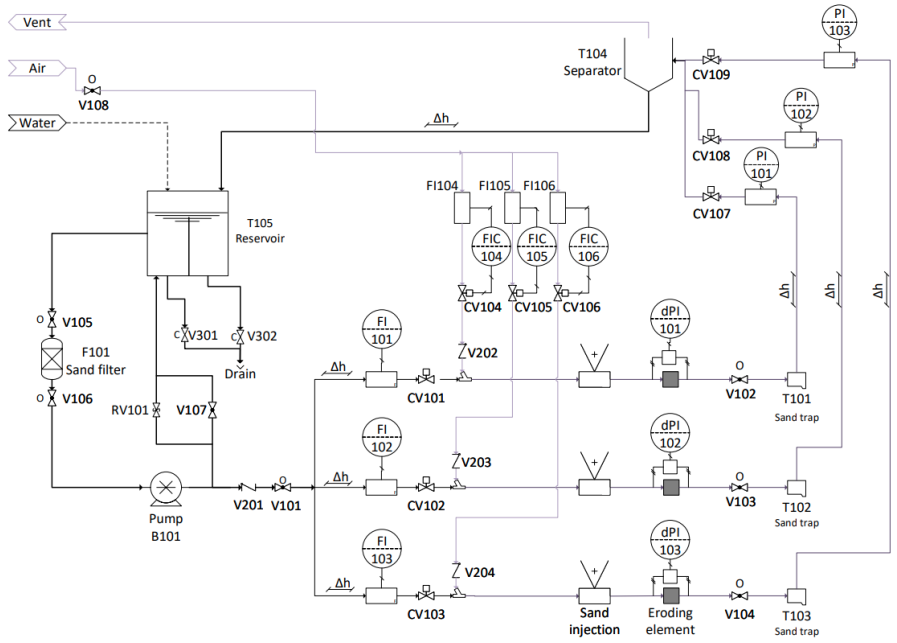
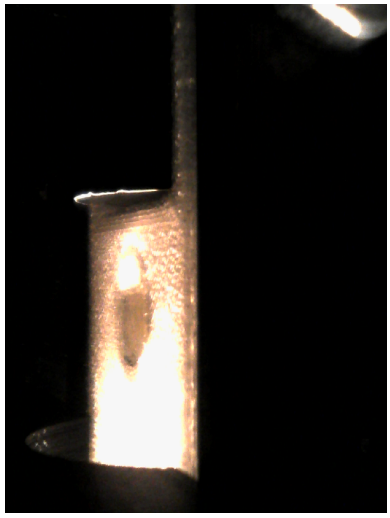


Figure 6.2: Schematic showing the setup of the experimental rig used to gather experimental data. The setup features 3 simulated ”wells”, each with a eroding element chamber and a air supply and later possibility for sand injection. Prior to the wells the rig has a sand filter and a pump after the reservoir. After the wells each well has a sand trap, to be used if sand is included in later experiments. On each well there is also pressure gauges to measure the differential pressure over the eroding element chamber.

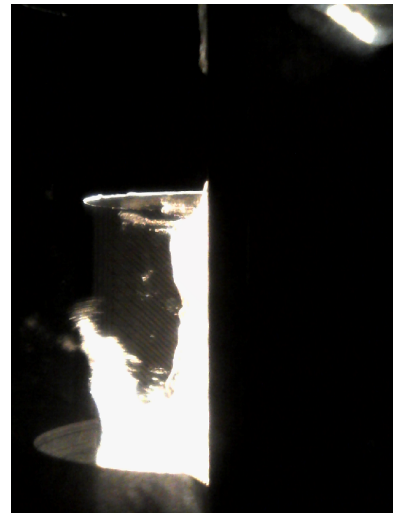
The final approach that we ended up using for the experimental measurements was using a ”stand in” measurement for the erosion. The theory was that if we placed the probe and holder in such a way that the intact probe impeded fluid flow through the chamber, the degradation of the probe would correspond directly to a decrease in measured differential pressure across the erosion chamber. This approach has been significantly more fruitful since it has removed the need for advanced image processing as well as gathering very high quality images, in the final iteration the images are only used for validation of observations. A mechanical justification for using the measured differential pressure as a stand in for erosion as described here is given below.

Using differential pressure as a measure of degradation

Here, we show the correlation between pressure drop and the cross-sectional area, corresponding to the degraded probe area. Two equations are needed to make this argument, the Bernoulli equation and the continuity equation. By assuming that the fluid is incompressible and that the flow is steady, the continuity equation can be written as Equation 6.1.



(a) Probe prior to erosion taking place.



(b) Probe after significant erosion has happened.

Figure 6.3: Before and after images of the erosion probe, showing one picture immediately after the start of the experiment and one when the probe is severely eroded.

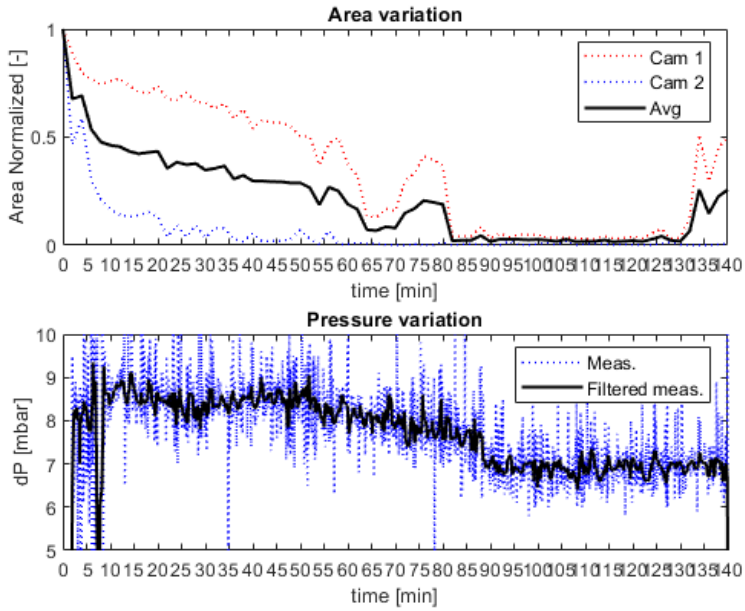


Figure 6.4: Plots of the preliminary experimental data, plotting area, differential pressure, top pressure and flowrates as functions of time for about 3 hours. Initial inspection shows quite clearly that the strong correlation between flowrates and pressures to the area seem to not be present in these, making statistical modelling a unfeasible prospect.

Table 6.1: List of the six experimental variables that were used in modelling. The differential pressure was used as the target variable instead of direct erosion measurements as discussed in Section 6.1.1. Tags corresponding to the schematic in Figure 6.2 are given in parenthesis, and units of measurements are given in brackets. The currents are the output control signals that decide pump effect, valve openings etc.

Number	Symbol	Name
Predictor 1	t	Relative time **[s]**
Predictor 2	q	Flowrate measured (FI101-103) **[L/min]**
Predictor 3	p_t	Topside pressure (PI 101-103) **[mbar]**
Predictor 4	T	Temperature (TI 101-103) **[C]**
Predictor 5	I_{cv}	Current bottom valve (CV101-103) **[A]**
Predictor 6	dP	Differential pressure (dP 101-103) **[mbar]**

$$A_2 * v_2 = A_1 * v_1 \tag{6.1}$$

Where A_2, v_2, A_1 and v_1 are the areas and velocities of the fluid in two differing segments of a continuous flow. This tells us that when the cross sectional area of the fluid flow is increased, the velocity is decreased. Using Bernoulli's principle stated as Equation 6.2, we see that as the velocity decreases, the pressure p will increase.

$$\frac{v^2}{2} + gz + \frac{p}{\rho} = constant \tag{6.2}$$

This will bring the pressures at both sides of the erosion box closer together as the flow through the box becomes less constricted. Therefore, as the probe erosion evolves, the differential pressure over the erosion chamber decreases.

6.2 Experimental Data

In this section, a brief description of the gathered experimental data will be given. For information about how the data is processed and analysed see Chapter **??**. Several sensors are used to gather 34 different measurements at each time step (time steps are set to 1 sec). A complete list of all the measurements can be found in Appendix C. When obtaining the data-driven models, not all measurements are relevant, The eight measurements per "well" that are used in the analysis in Chapter 7 are listed in Table 6.1.

As said earlier in Section 6.1.1 the decrease in differential pressure is going to be used as the response variable, instead of direct measurements of the probe degradation.

Plots of the differential pressure of two wells throughout an experimental run are shown in Figure 6.5. As can clearly be seen here, there is a much more stable, strictly declining trend in this data compared to the data that was gathered previously (Figure 6.4. This strict downward profile matches the expected trend, since it has been shown that the differential pressure behaviour is inversely proportional to the probe erosion.

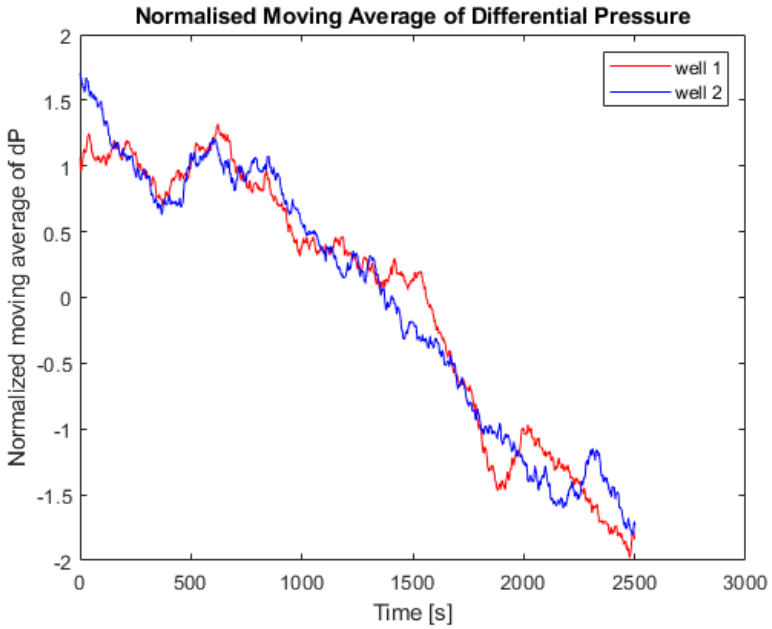


Figure 6.5: Plot of normalised data after application of a 150 seconds wide moving average filter, a clear, almost strictly declining trend is observed. The very wide moving average filter had to be used to circumvent a very high level of noise in the raw data.

Chapter 7

Case study 4: Experimental data

The fourth and final case study in this thesis is an investigation of the previously tested statistical learning methods. In this chapter it will be tested if the erosion process can be accurately modelled in the real world using the same methods as in the *in-silico* case studies. Because getting proper data of component degradation from a real subsea plant is prohibitively expensive, a smaller scale lab rig was used. The method of the experiments is described in detail in Chapter 4. In summary, since the reduction in probe area covering the fluid flow parth is proportional to the differential pressure across the erosion chamber, this measurement was used as a stand in for probe degradation. This change in differential pressure will then be predicted with aforementioned statistical learning methods on the basis of process measurements (Table 6.1). This chapter will first discuss the exploratory analysis of the data as well as how pre-processing is used to make the data fit for regression modelling. Then the results of applying the previously tested regression methods will be discussed and compared.

7.1 Exploratory data analysis and pre-processing

7.1.1 Pre-processing

The experimental data is collected in six operating conditions, represented by different liquid flowrates (2, 3, 4, 5, 6 and 9 L/min). For each condition, we gather data from the three wells, leading to 18 independent probe degradation profiles. The experiments were run until the probes in the three wells looked qualitatively ”destroyed” on the camera pictures. Since the liquid flowrate is held constant during operation in this experimental design, capturing the effect on the erosion profile is hard as it will not correlate to the degradation. To capture this effect properly a design could for instance capture fluid flowrate and the rates at which the probe degrades. The noise level of the data made using the gradient of the dP profile difficult. For training the models, we concatenated the time series of all operating conditions such that we obtained three data sets, one for each well. Wells one and two were combined to create a training set, while well three became the independent test set.

Validation was done splitting out part of the training set with cross and holdout validation as described in Chapter 2. The data was then normalised before analysis was performed. PCA pre-processing was not performed for this case, as the data we are dealing with is of relatively low dimensionality and the performance in the previous case studies as well as initial testing did not indicate that it would positively impact the models.

7.1.2 Exploratory data analysis

The most significant difference between real and simulated data is that there is a large amount of noise and variance and it is difficult to determine their source. While in the simulated case studies the noise was introduced in a controlled fashion and followed a normal distribution. No such guarantee exists for real world data. In particular the differential pressure measurements were very noisy, which is a problem when you want to use it to to represent the strictly increasing trend in the probe erosion. Figure 7.1 illustrates the high level of noise in the differential pressure measurements when compared to the size of the effect from the eroding probe. Since the trend of decrease in the differential pressure is what is assumed to represent the erosion, a high level of smoothing is imposed on the data with a 150 seconds moving average removing most of the measurement noise while retaining most of the functional shape of the trend. Another option would have been to use a trend line or other line fit and try to predict this trend. However, this would impose an arbitrary linearity on the erosion that may not be realistic. A wide moving average window of 150 also removes a lot of fluctuation in the data, some of which may contain important information. By visual inspection, this maintains as much of the original shape as possible while resembling the strictly decreasing trend in the differential pressure that is used to represent the strictly increasing erosion. The knowledge that the remaining area, which is represented by the differential pressure, is strictly decreasing and relatively smooth provides a strong argument that applying such a wide moving average window is acceptable.

After smoothing the data variation, we explore the correlation between the measurements. Checking that there are significant correlations between the predictor data and the response variables is an important preliminary step for obtaining a proper data-driven model. A correlation plot including individual correlation values for each pair of variables, including the 5 predictors and the response, is shown in Table 6.1. The most pertinent correlations to study is the correlations between each predictor and the response. However, correlation between predictors must also be noted, since it can be problematic for model identification. The first variable is the relative time, which is time elapsed since the start of the experiment. This variable is as expected strongly negatively correlated (correlation coefficient: -0.71) with the differential pressure since the probe erodes over time and each time step should see the probe eroded further than the last. The second variable is the measured flowrate, while this may be the single most important variable in the system, it shows almost no correlation to the response. This is due to the fact that the flowrate was kept constant by control valves throughout the experimental runs. On the other hand, it is fair to assume that a significant correlation between these variables would be observed, if the flowrates were allowed to vary freely as the probe eroded. The topside pressure seems to be basically independent of all the other measurements including the response and time elapsed in the experiment with no correlation coefficients of magnitude greater

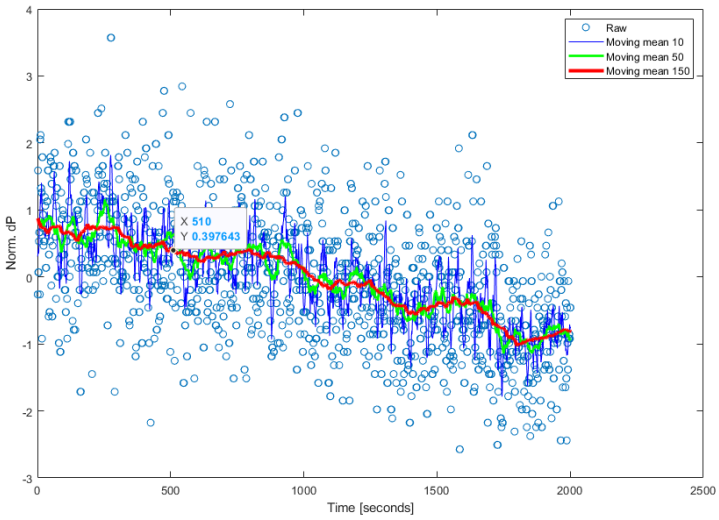


Figure 7.1: Plot of the experimental data from the run of the rig with the controlled flowrate being set to 5 [sl/min] showing a high level of noise in the raw data, as well as the data after being subjected to several different moving averages using window widths of 10, 50 and 150.

than 0.06. The fourth variable, the temperature is also significantly negatively correlated to the response (correlation coefficient: -0.56), this is somewhat surprising as one would not think that a increase in the temperature of absolute magnitude smaller than one full Kelvin, would have a significant impact on for instance the solubility of PVA in the water or something like that. This is probably a case of correlation without causation as both variables have a strong correlation to the time elapsed. The final predictor variable, the control current to the valve that controls the liquid flowrate has a significant correlation (correlation coefficient: 0.50). This is likely due to the fact that the control valve behaviour changes depending on the hindrance, i.e. the amount of un-eroded probe blocking the flow.

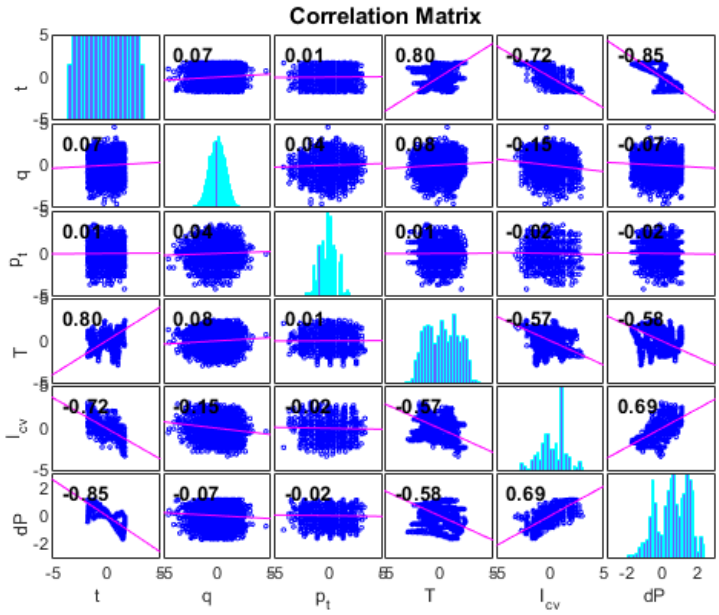


Figure 7.2: Correlation matrix of the six variables used for modelling the experimental change in differential pressure as the probes eroded, variables are numbered the same as in Table 6.1 in Chapter 4 with variables one through 5 being the predictors and variable 6 being the response.

7.2 Regression results

In this section, the results from applying the statistical learning methods to the data from this fourth case study. This includes results from linear regression, regression trees, ensemble methods, support vector regression and artificial neural networks. For the neural networks, direct prediction is done using a simple regression network, but results from NLIO and NARX time series models are also shown. The performances of all of these methods in terms of MSE are shown in Table 7.1 and Figure 7.10.

7.2.1 Linear regression

The stepwise linear regression managed to capture the downward trend of the differential pressure but not the fluctuations, this is clearly visible in the plotted response shown in Figure 7.3. The MSE was only 0.0854 over a full test set. The full model selected by the stepwise selection algorithm contains 10 terms, 1 bias term, 5 variable terms and 4 interaction terms that significantly benefited the model, according to the stepwise selection algorithm described in Chapter 2. It should be noted that the variance of the predictions is very low for this this model, which is a result of the trained model being a continuous hyperplane. This is different from the tree based models which do not have any sort of continuity requirement and by their very nature are discrete.

For improving this model performance, one can consider using additional higher order and/or different non-linear terms in order to properly describe the system dynamics. Then, L1 regularisation can be applied to discover the relevant terms, similarly to Brunton et al.[28]. This approach has the advantage of possibly learning an equation model describing fundamental dynamics of the system, which none of the other methods are able to do.

7.2.2 Regression trees

Compared to the linear model, the tree is able to capture some of the non-linearity in the differential pressure profile, this is shown in Figure 7.4. The performance of the tree alone is about the same as the linear regression, with a test set MSE of 0.1213. The tree prediction was made by applying Bayesian optimisation to hyperparameters, selecting minimum leaf size, whether pruning should be turned on and which splits should be considered. The hyperparameter optimisation was limited to 30 iterations.

The CART algorithm[29] was used to train the trees with the optimised hyperparameters. The main weakness of the tree prediction model is its high prediction variance. In Figure 7.4 you can see that at 8500 seconds, when the true value is close to one, the outlier predictions span from about 0.25 to -1.5. Such variance can significantly impact any decision-making method based on the tree method's predictions. For example, from the results in Figure 7.4, the most positive predictions show a probe only partially degraded while the reality is that the probe is almost fully degraded which could lead to dangerous decisions. Due to the nature of this system, this drawback of the tree model can be compensated. For example, let us say that our goal is state estimation, one might not need second by second state estimates, because in reality erosion is a slow process when dealing with real materials. Thus we can apply a moving average to the predictions, and as it is

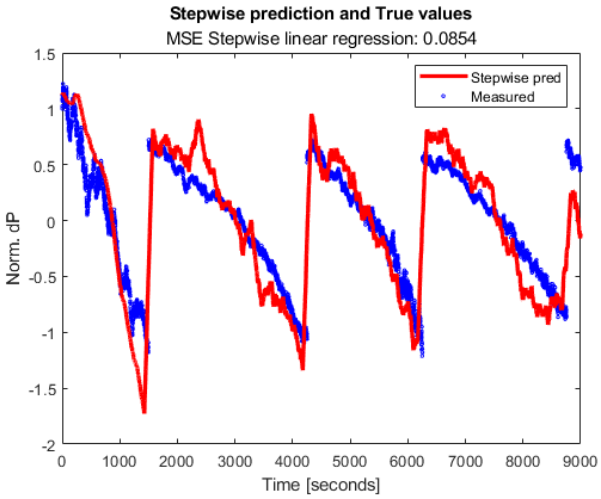


Figure 7.3: Regression result for experimental data when applying stepwise linear regression to the training data.

shown in Figure 7.4, 30 second wide windows yield much more accurate predictions, reducing the MSE to 0.0958 and significantly decrease the variance of predictions. Another approach that could further reduce the variance of tree predictions is using an ensemble approach with multiple trees, which is shown in the next section.

7.2.3 Ensemble methods

As expected the ensemble approach gives a significantly smaller prediction variance by averaging several trees. The same Bayesian hyperparameter optimisation was applied to the ensemble as with the tree. The optimisation chose bootstrap aggregation as the preferred ensemble algorithm, using 496 bagged trees with equal weight. The number of predictors for each tree, minimum leaf size and number of learners were chosen in the same way. The ensemble of trees improves upon the prediction of the tree based methods, with a test set MSE of 0.0998. In Figure 7.5, we see that, even with averaging 496 learners, there is a significant amount of model prediction outliers. Applying the same 15 second moving average, the performance is somewhat improved, becoming 0.0852. As with the previous case studies, note that the ensemble method is generally very accurate, but this comes at the expense of any meaningful inference about the underlying dynamics of the system and the transparency of the model.

7.2.4 Support vector regression

Initial expectations for support vector regression were quite low due to the low dimensionality of this data. For SVR to make sense in applications it needs to provide very accurate predictions as it is a very black box approach and thus offer little in terms of inference and explainability. The resulting predictions from applying SVR to the experimental data

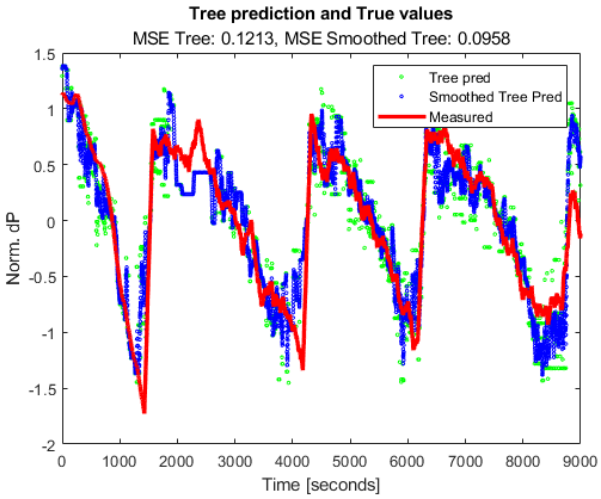


Figure 7.4: Regression result for experimental data when applying a single regression tree (green) as well as the prediction when the predictions are smoothed with a 15 second moving average (blue). The true value is also shown in red.

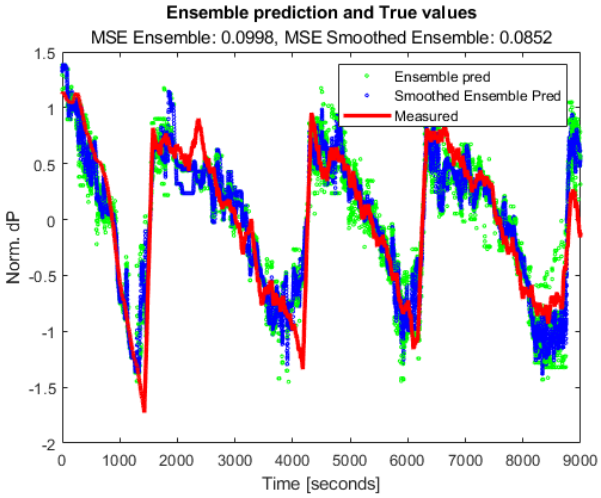


Figure 7.5: Regression result for experimental data when applying a bagged ensemble of trees (green) as well as the prediction when the predictions are smoothed with a 15 second moving average (blue). The true value is also shown in red.

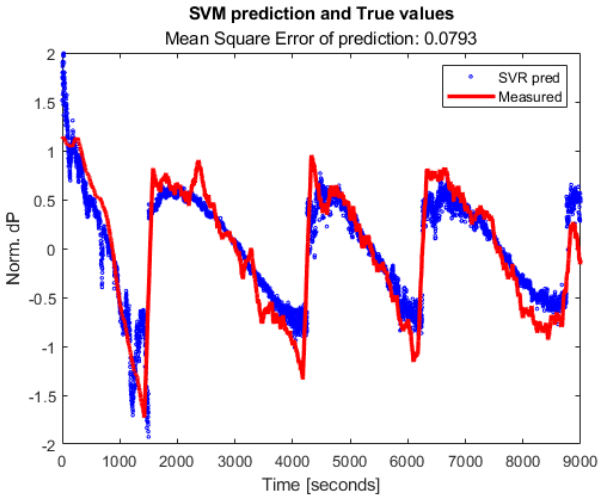


Figure 7.6: Regression result for experimental data when applying Gaussian kernel SVR (blue) and the true value (red).

had a MSE of 0.0793 and can be seen in Figure 7.6. Compared to tree and ensemble methods, the variance of predictions from SVR are significantly lower, with few significant outliers. Compared to the stepwise regression, another method based on creating a hyperplane, SVR is more able to capture the non-linear profile of the differential pressure profile. This is one of the most significant benefits of using SVR, as it allows the use of non-linear kernels, in this case the Gaussian kernel which allows emulations of a variety of non-linear profiles.

7.2.5 Neural networks

The experimental data should be very well suited for neural network applications, since the system dynamics are complex and nonlinear, and the amount of available data is relatively large. The data is a time series, but as has been shown in the previous subsections, models are able to perform rather well on it without using any auto-regressive properties. A simple regression network, making direct predictions like the previously discussed models is tested first. Afterwards, two time series based models will be investigated, the NLIO and NARX models.

ANN Regression

A simple regression network was able to capture the dynamics of the degradation, it quite accurately models the changes in differential pressures across the experiments, as shown in Figure 7.7. The regression network had similar performances for different sizes, thus a smaller network of 10-15 nodes is preferred as it is sufficient to achieve strong performances without being slow to train. The variance of predictions were also significantly lower than the ensemble methods but with comparable accuracy, having a MSE of 0.0764.

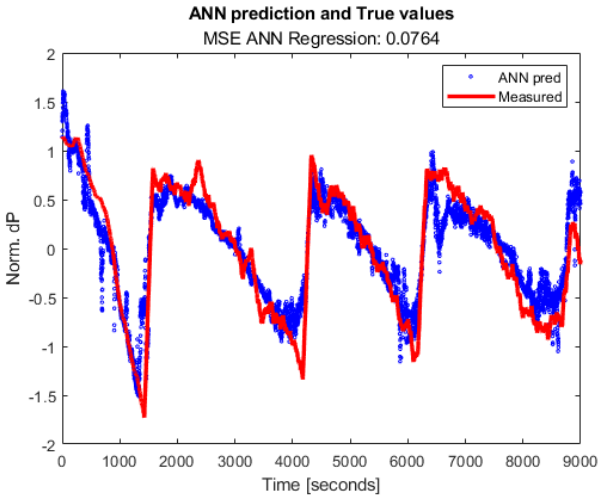


Figure 7.7: Regression result for experimental data when applying a simple regression network (blue) and the true value (red).

NLIO Network

The first time series model that is tested is the NLIO network, it can also be thought of as a regression with the previous (lagged) predictor values merely being additional predictors. The NLIO network is specifically designed to deal with non-linear dynamics, and as expected it has a very strong performance in terms of capturing the non-linearity in the data as shown in Figure 7.8. The MSE of the NLIO network was 0.0523, making it stronger than all the non-time series models. This could be expected as it has more information to use when making its predictions. This information has value when making predictions, this is clear from looking at the comparison to the ordinary regression network.

NARX Network

The NARX model shows significant potential, it is first tested in open loop format being fed the true (measured) lagged values of the response, then changed to closed loop format. This was to show how this can be used to monitor a process that cannot be continuously observed directly, either due to cost or physical limitations. As expected the open loop predictions are almost perfect, as the change from time step to time step is very small, and thus knowing the previous time step makes the predictions very strong. A positive indication, is that when the loop is closed and it only relies on its own previous predictions rather than true values, it still performs well with a closed loop MSE of 0.0384 in the interval from 5000 to 9000 seconds. For comparison the ensemble model has a MSE of 0.0646 and the NLIO network has a MSE of 0.0537 in that time interval. It is not surprising that a time series model does perform better since what we are dealing with is time series data, but the time series is ”seasonal” in terms of experiments and non-stationary with large differences in variance across time. Thus it is a very positive indication for the viability of

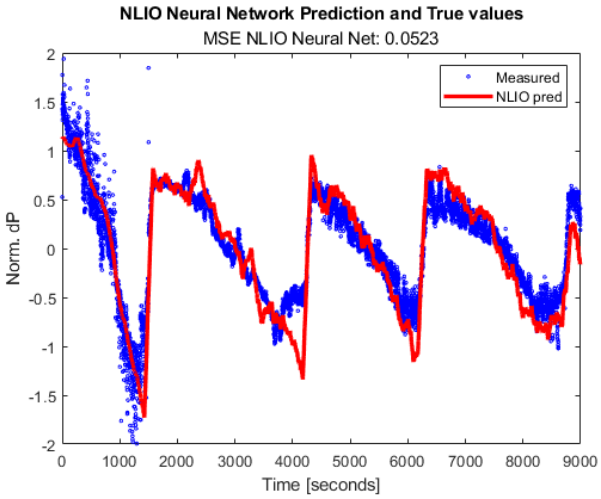


Figure 7.8: Regression result for experimental data when applying a NLIO neural net (green) and the true value (red).

this method that the NARX model still delivers very accurate predictions with the lowest variance of predictions out of all models. The low variance of predictions are due to the auto-regressive property, limiting the amount of change from one step to the next when the first order partial autocorrelation is high as in this case.

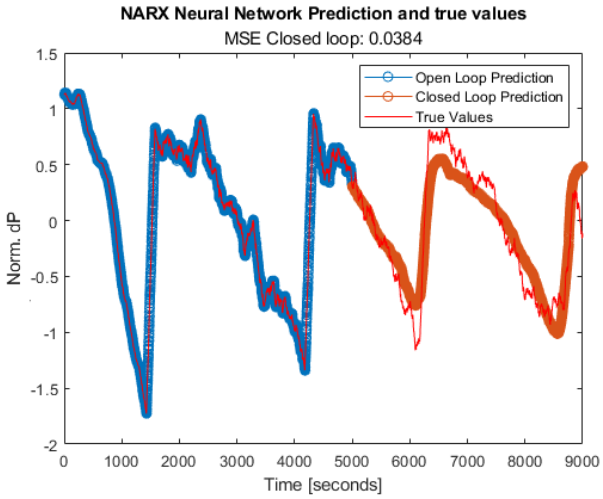


Figure 7.9: Results of open loop predictions of the NARX network from 0 to 5000 seconds (blue), closed loop predictions from 5000 to 9000 seconds (orange) and the true values from 1 to 9000 seconds (red).

7.3 Summary

The stepwise linear model performed well when compared to the other traditional statistical learning methods like regression trees, ensembles and support vector regression. Other advantages of the stepwise MLR is that it is more interpretable, transparent and faster at making predictions. Surprisingly, given the results of the previous case studies, ensembles performed relatively poorly. This may be due to the real data being very noisy with predictions being made on a very smoothed profile, the same argument can be made for regression trees. Both of these methods' predictions have a very high variance somewhat similar to the real data. Linear regressions avoid a lot of the noise in predictions by enforcing a continuity and in general having far fewer degrees of freedom and thus less likely to learn patterns in the data instead of the underlying phenomenon. Stepwise regression has a certain inbuilt protection against overfitting from the significance criteria for terms in the model using the F-test as described in Chapter 2. When working with real data, it is expected that the modern approaches to machine learning will perform better, but at the cost of a far higher model complexity and being less transparent. The ANN network was the best regression approach that did not use the temporal features of the data, its performance is further improved upon by adding lagged predictor values to the predictor data in a NLIO network. Finally using the NARX network, where also the auto-regressive property is exploited yielded the strongest predictions in its closed loop form when making predictions based on its past predictions.

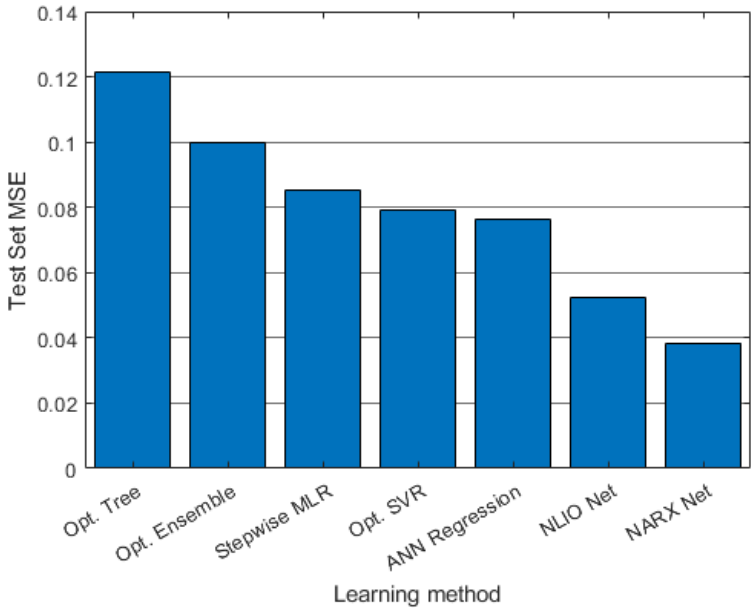


Figure 7.10: Bar plot over performances by the statistical learning methods in terms of test set MSE, from best to worst from left to right.

Table 7.1: Model Performance, in terms of mean square error of prediction on unseen test data for several statistical learning techniques applied to experimental data recorded from the lab rig. Both results for direct predictions, as well as for 15 second moving average windows for the methods were this was tested. For the NARX model, the MSE was computed using only the closed loop predictions made in the range 5000-9000 seconds.

Case Study 4: Experimental Data		
Method	Direct predictions	Smoothed predictions
Stepwise MLR	0.0854	N/A
Optimised tree	0.1213	0.0958
Optimised Ensemble (LSBoost/Bag)	0.0998	0.0852
Optimised SVR	0.0793	N/A
ANN Regression	0.0764	N/A
NLIO Neural Net	0.0523	N/A
NARX Neural Net	0.0384	N/A

Chapter 8

Conclusion

In this thesis, we discuss the use of statistical learning methods for modelling equipment degradation. This is motivated by necessity, given the difficulty in obtaining first principles models of this phenomenon. We present the advantages and challenges of this approach using both simulated and experimental case studies.

8.1 Case study 1: Constant sand production

In the first case study, we study the erosion of a choke valve in a subsea gas lifted oil production network under constant sand production from the well. The model is presented in Chapter 3, looking at the model, the erosion under constant sand production should be a linear phenomena and thus simple to predict. This is what was observed in the results, with every model being able to make predictions that were very accurate. In this case, simple and transparent models like partial least squares regression (PLSR) and multiple linear regression (MLR), and should be preferred over more complex methods such as support vector regression (SVR) and neural network regression.

8.2 Case studies 2 & 3: Exponential and logistic sand production

The second and third case studies are similar to the first case study, except for the addition of variation in the sand production in the simulation. Since in reality, the sand production is not fully known, the models are fed a sampled sand production rate with a given sampling interval. Making the sand production exponential, gave the erosion profile an exponential trend, but with oscillations corresponding to changes in the control input. In this case study the accuracy benefits of using more complex models were observed, but with acceptable performance from simple MLR and regression trees.

The third case study changed the sand production used in the simulations to a logistic profile. Additionally, stochastic noise was added to the underlying sand production. The

added noise was observed to not exhibit a significant impact on the macroscopic trend of the erosion. The order of the prediction accuracies observed in the exponential case was somewhat altered. This is attributed to the noise added in Case Study 3. Regression trees and ensembles of trees are more robust against small predictor noise as a small change in a predictor value will not change the prediction directly if it still satisfies the same splits. Thus, for sand rates within a certain interval, the prediction will be the same all else being equal. This is not true for MLR, SVR and neural networks, where a direct proportionality from weights is observed, meaning even a small change will perturb the prediction.

Studying the prediction profiles versus the true erosion profiles show us the weakness of the methods are the inaccurate sand production data coming from the sampling interval. This was confirmed by seeing a large improvement in accuracy by lowering sampling intervals, but this is not cost effective or practical in the real world. Domain knowledge tells us that sand production rates have trends that generally hold across multiple sampling intervals. We thus replace the sampled sand production at each point with a modelled sand production. The modelled values come from linear extrapolation using the two previously observed values until a new sample is observed.

Up to this point, we have showed how to accurately model the erosion rate, we then investigate a time series approach to modelling the cumulative erosion with a non-linear auto-regressive exogenous neural network (NARX). To be useful for these applications, long and accurate forecasts are necessary while in closed loop configurations, that is without using true values of the previous response, but rather the networks own previous predictions. Forecasting 250 days into the future, using only the exogenous variables observed over this time (closed loop configuration), the cumulative error is only 1.6%.

Conclusions from these results are that modelling the erosion rate using statistical learning models is convenient. For inference, the accuracy of stepwise MLR and simple regression trees perform well, being only hindered by the strong co-linearity's in the data, PCA can help with this problem. Modern machine learning methods were more accurate, but at a significant cost to interpretability. The approaches discussed above have shown themselves to be accurate, and could improve control strategies by forecasting degradation by statistical learning, for instance in a model predictive control setting. Having shown that the approach performs well on *in-silico* case studies, we investigate if the statistical methods also perform well on real world data.

8.3 Case study 4: Experimental data

Having shown that statistical learning methods perform well on simulated degradation data, we in this section test the merits of this approach experimental data from a laboratory rig. We went to several different set ups to obtain usable experimental data (see 4), the final approach uses the differential pressure to represent the degradation of a probe blocking the fluid flow. The differential pressure data was very noisy, and to capture the trend caused by the degradation of the probe a 150 second moving average was used on the measured differential pressure.

Linear regression is able to capture the overall trend, but does not perfectly capture the non-linearity, this is unsurprising for a first order interactions model. Note that the variance of predictions are very low. SVR was better able to capture the non-linearity in

the differential pressure profile, maintaining a low variance of prediction. This is attributed to the fact that these are models based on forming a continuous hyperplane.

Trees and ensembles of trees both had a high variance in their respective predictions. This can be attributed to applying discrete models to data with a large amount of noise. To compensate for this, a moving average was applied to the predictions. Thus yielding state estimates at a later time since one needs to wait for the moving average window to be centred on a given point to know the value of the smoothed prediction. Even applying this smoothing filter, the predictions do not outperform the aforementioned hyperplane methods (SVR & MLR) and is thus not advisable for any application for this case study.

Neural networks were then tested. First a simple regression network, as with SVR it is able to capture the non-linearity in the profile. This is to be expected as a non-linear (sigmoid) activation function was used. The neural network had slightly more accurate predictions than the SVR model. Building upon the neural network, we then tested a NLIO model, using lagged predictor values to make predictions. The hypothesis that these lagged predictors carry useful information is supported by their inclusion improving accuracy over a simple regression network.

Improving further on the NLIO results, there seems to be a significant auto-correlation in the response data, this is utilised in the model by using a NARX model. This model yielded far smoother predictions by using the previous prediction value as a important covariate, making sure predictions close to each other in time have similar values.

The NARX network provided the strongest predictions, and should be used if prediction accuracy is most important and generalisation is not too important. If inference and transparency is the objective, stepwise linear regression with interactions was able to give accurate enough predictions to be useful for applications.

8.4 Further Work

In the experimental case study, the most important future work that should be done to further this approach is to set up a stronger methodology for directly measuring the erosion instead of indirectly. This could be done by improving upon the pixel counting method with better image processing, better cameras and probes with a stronger contrast to the background. Another possibility could be the use of weighing after set time intervals to measure lost mass, but this method would require a lot of time to make a large data set.

Other modelling approaches could also be tested. To be useful for the applications we are investigating, the ability to make accurate predictions about the state of the system without observing it often is important. For time series models, this means that we need models able to make forecasts accurately far into the future. In this work, the only method with an auto-regressive component tested was the NARX neural net, it was chosen due to its ability to use exogenous variables and past predictions to forecast into the future. There are however several more time-series methods that may have merit and should be investigated, such as the ARIMAX model.

Bibliography

[1] G James, D. Witten, T Haste, and R Tibshirani. *An Introduction to Statistical Learning*. Springer, 2017.

[2] A Verheyleweghen and J Jäschke. Oil production optimization of several wells subject to choke degradation. *IFAC-PapersOnLine*, 51, 2018.

[3] Gisle Otto Eikrem, Lars Imsland, and Bjarne Foss. Stabilization of gas lifted wells based on state estimation. *IFAC Proceedings Volumes*, 37(1):323 – 328, 2004. 7th International Symposium on Advanced Control of Chemical Processes (ADCHEM 2003), Hong-Kong, 11-14 January 2004.

[4] Siri Kildal Hansen. *Modelling Failure Mechanisms in Subsea Equipment*. PhD thesis, NTNU, 2016.

[5] X. Si, W. Wang, C. Hu, D. Zhou, and M. G. Pecht. Remaining useful life estimation based on a nonlinear diffusion degradation process. *IEEE Transactions on Reliability*, 61(1):50–67, 2012.

[6] D Krishnamoorthy, B Foss, and S Skogestad. Real-time optimization under uncertainty applied to a gas lifted well network. *Processes, Real-Time Optimization (Special Issue)*, 2016.

[7] M. H. Hettema, J. S. Andrews, E. Papamichos, and M. Blaasmo. The relative importance of drawdown and depletion in sanding wells: Predictive models compared with data from the statfjord field. *Journal of Petroleum Technology*, 2006.

[8] Son Tung Pham. Estimation of sand production rate using geomechanical and hydromechanical models. *Advances in Materials Science and Engineering*, 2017.

[9] G James, D. Witten, T Haste, and R Tibshirani. *An Introduction to Statistical Learning*. Springer, 2017.

[10] J Friedman, T Hastie, and R Tibshirani. *The Elements of statistical learning, Data Mining, Inference, and Prediction*. Springer, 2017.

[11] G James, D. Witten, T Haste, and R Tibshirani. *An Introduction to Statistical Learning*. Springer, 2017.

[12] S. Wold and Sjöwström. Pls-regression: a basic tool of chemometrics. *Chemometrics and Intelligent Laboratory Systems*, 58:109–130, 2001.

[13] A. E. Stott, S. Kanna, D. P. Mandic, and W. T. Pike. "an online nipals algorithm for partial least squares,". In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),*, New Orleans, LA, 2017, pp. 4177-4181., 2017.

[14] J Friedman, T Hastie, and R Tibshirani. *The Elements of statistical learning, Data Mining, Inference, and Prediction*. Springer, 2017.

[15] Harris Drucker, Chris C, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. *Advances in Neural Information Processing Systems*, 9, 11 2003.

[16] J Friedman, T Hastie, and R Tibshirani. *The Elements of statistical learning, Data Mining, Inference, and Prediction*. Springer, 2017.

[17] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[19] J Friedman, T Hastie, and R Tibshirani. *The Elements of statistical learning, Data Mining, Inference, and Prediction*. Springer, 2017.

[20] P. Norvig and S. Russel. *‚Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd edition, 2009.

[21] J Friedman, T Hastie, and R Tibshirani. *The Elements of statistical learning, Data Mining, Inference, and Prediction*. Springer, 2017.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[23] Peter I. Frazier. A tutorial on bayesian optimization, 2018.

[24] DNV-GL. Recommended practice rp-o501: Managing sand production and erosion., 2015. `https://rules.dnvgl.com/docs/pdf/dnvgl/RP/2015-08/DNVGL-RP-O501.pdf`.

[25] Timur Bikmukhametov and Johannes Jäschke. Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models. *Computers Chemical Engineering*, 138:106834, 2020.

[26] Eugen Diaconescu. The use of narx neural networks to predict chaotic time series. *WSEAS Transactions on Computer Research*, 3, 03 2008.

[27] Tamal Datta Chaudhuri and Indranil Ghosh. Artificial neural network and time series modeling based approach to forecasting the exchange rate in a multivariate framework, 2016.

[28] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Sparse identification of nonlinear dynamics with control (sindyc). *IFAC-PapersOnLine*, 49(18):710 – 715, 2016. 10th IFAC Symposium on Nonlinear Control Systems NOLCOS 2016.

[29] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984.

[30] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

Appendix A

Simulation model

In this appendix, the scripts that were used to simulate erosion data as discussed in Section 3. The appedix contains 6 scripts, one containing the initial conditions of the gas lifted system (Section A.1), one containing parameters (Section A.2), function creating sand production profile (Section A.3), a function for linearly modelling the sand production (Section A.4), the model itself implemented using CasADi[30] (Section A.5) as well as the script that runs the model (Section A.6).

A.1 InitialConditionGasLift

```
1
2  function [dx0,z0,u0] = InitialConditionGasLift_5
3
4  %% Differential states
5  %well erosion
6  ER0 = [1,1,1]'/(365*24*3600); %[mm] x(1-3)
7
8  dx0 = vertcat(ER0);
9
10 %% Algebraic states
11 %pressure - annulus
12 p_ai0 = [61.9230, 62.1454, 62]';%[bar] z(1-3)
13 %pressure - well head
14 p_wh0 = [42.5851, 45.3082, 44]';%[bar] z(4-6)
15 %pressure - injection point
16 p_wi0 = [56.8713, 57.1119, 57]';%[bar] z(7-9)
17 %pressure - below injection point (bottom hole)
18 p_bh0 = [96.1433, 98.3867, 96.25]';%[bar] z(10-12)
19 %density - annulus
20 rho_ai0 = [0.4949, 0.4967, 0.4955]';%[100 kg/m3] z(13-15)
21 %mixture density in tubing
22 rho_m0 = [2.3400, 2.2359, 2.3]';%[100 kg/m3] z(16-18)
23 %well injection flow rate
```

```
w_iv0 = [1.5000, 1.5000,1.5000]';%[kg/s] z(19-21)
%wellhead total production rate
w_pc0 = [30.1212, 33.3235,32]';%[kg/s] z(22-24)
%wellhead gas production rate
w_pg0 = [3.1928, 4.0168, 4]';%[kg/s] z(25-27)
%wellhead oil production rate
w_po0 = [26.9283, 29.3067, 28]';%[kg/s] z(28-30)
%oil rate from reservoir
w_ro0 = [26.9283, 29.3067, 28]';%[kg/s] z(31-33)
%gas rate from reservoir
w_rg0 = [26.9283, 35.1680, 28]';%[0.1 kg/s] z(34-36)
%riser head pressure
p_rh0 = 22.9558;%[bar] z(37)
%mixture density in riser
rho_r0 = 1.3618;%[100 kg/m3] z(38)
%manifold pressure
p_m0 = 32.8920;%[bar] z(39)
%riser head total production rate
w_pr0 = 63.4446;%[kg/s] z(40)
%riser head total oil production rate
w_to0 = 85;%[kg/s] z(41)
%riser head total gas production rate
w_tg0 = 7.2096;%[kg/s] z(42)

%gas holdup @ annulus
m_ga0 = [1.0568, 1.0606, 1.0644]';%[ton] z(43-45)
%gas holdup @ well
m_gt0 = [0.7470, 0.7956, 0.8442]';%[ton] z(46-48)
%oil holdup @ well
m_ot0 = [6.3000, 5.8047, 5.3094]';%[ton] z(49-51)
%gas holdup @ riser
m_gr0 = 0.1265;%[ton] z(52)
%oil holdup @ riser
m_or0 = 0.9863;%[ton] z(53)
%particle impact velocity
V_p0 = [2,2,2]'; % z(54-56)
%mixed dynamic viscosity
mu_f0 = [0.001,0.001,0.001]'; % z(57-59)
%g1
g10 = [0.2,0.2,0.2]'; % z(60-62)

z0 = vertcat(p_ai0,p_wh0,p_wi0,p_bh0,rho_ai0,
    rho_m0,w_iv0,w_pc0,w_pg0,w_po0,...
    w_ro0,w_rg0,p_rh0,rho_r0,p_m0,w_pr0,w_to0,
    w_tg0,m_ga0,m_gt0,m_ot0,m_gr0,m_or0,V_p0,mu_f0,g10);

%% Inputs
%gas lift rate
w_gl0 = [0.5,0.5,0.5]'; %[kg/s]

u0 = vertcat(w_gl0);
```

A.2 ParametersGasLift

```
function  par = ParametersGasLift(n,sandArray)

%number of wells
par.n_w = 3;
%gas constant
par.R = 8.314; %[m3 Pa/(K mol)]
%molecular weigth
par.Mw = 20e-3; %[kg/mol] -- Attention: this unit is not usual

%% Properties
%density of oil - dim:  nwells x 1
par.rho_o = 8*1e2; %[kg/m3]
%riser oil density
par.rho_ro = par.rho_o; %[kg/m3]
%1cP oil viscosity
par.mu_oil = 1*0.001; %[Pa s or kg/(m s)]

%% Project
%well parameters - dim:  nwells x 1
%length
par.L_w = [1500;1500;1500]; %[m]
%height
par.H_w  = [1000;1000;1000]; %[m]
%diameter
par.D_w = [0.121;0.121;0.121]; %[m]
%well transversal area
par.A_w = pi.*(par.D_w/2).^2;%[m2]

%well below injection - [m]
par.L_bh = [500;500;500];
par.H_bh = [500;500;500];
par.D_bh = [0.121;0.121;0.121];
par.A_bh = pi.*(par.D_bh/2).^2;%[m2]

%annulus - [m]
par.L_a = par.L_w;
par.H_a = par.H_w;
par.D_a = [0.189;0.189;0.189];
%volume of the annulus
par.V_a = par.L_a.*(pi.*(par.D_a/2).^2 - pi.*(par.D_w/2).^2); %[m3]

%riser - [m]
par.L_r = 500;
par.H_r = 500;
par.D_r = 0.121;
%riser areas
par.A_r = pi.*(par.D_r/2).^2;%[m2]

%injection valve characteristics  - dim:  nwells x 1
par.C_iv = [0.1e-3;0.1e-3;0.1e-3];%[m2]
%production valve characteristics  - dim:  nwells x 1
par.C_pc = [2e-3;2e-3;2e-3];%[m2]
%riser valve characteristics
```

```
par.C_pr = [10e-3];%[m2]

%parameters
%reservoir pressure
par.p_res = [150;155;160]; % [bar]
%Annulus temperature
par.T_a = [28+273;28+273;28+273]; %[K]
%well temperature
par.T_w = [32+273;32+273;32+273]; %[K]
%riser temperature
par.T_r = 30+273; %[K]
%separator pressure
par.p_s = 20; %[bar]

%% Reservoir parameters
%System parameters for nominal model
par.GOR = [0.10;0.12;0.11];
par.PI = [5;5;5];

%% For scaling the noise
% pressure meters = 1
% flow meters = 0.1
par.scale = [1,1,1,1,1,1,0.1,0.1,0.1,0.1,0.1,0.1,1,1,0.1,0.1]';

%% For erosion model
% Sand
par.d_p = 2.5*10^(-4); %[m] particle diameter
par.rho_p = 2.5*10^3; %[kg/m3] particle density
par.mdot_p = sandArray(n+1); %[kg/s] sand rate

% Choke
par.K = 2*10^(-9); %[-] material erosion constant
par.rho_t = 7800; %[kg/m3] sensity CS
par.r = 0.2; %[m] radius of curvature
par.D = 0.05; %[m] Gap between body and cage
par.H = 0.15; %[m] Height of gallery

% Constants
par.C_unit = 1000; % Unit conversion factor: now in mm/s
par.C_1 = 1.25; %[-] Model/geometry factor
par.n = 2.6; %[-] Velocity coefficient
par.GF = 2; %[-] Geometry factor

% Precalculations of erosion in choke:
par.alpha = atan(1/sqrt(2*par.r));
par.F = 0.6*(sin(par.alpha) + 7.2*(sin(par.alpha) - ...
    sin(par.alpha)^2))^0.6 * (1-exp(-20*par.alpha));
par.A_g = 2*par.H*par.D; %[m2] Effective gallery area
par.A_t = par.D_w(1)^2*pi/(4*sin(par.alpha)); % Area exposed to erosion
par.ER_constant = par.K*par.F*par.C_1*par.GF*par.mdot_p ...
    *par.C_unit/(par.rho_t*par.A_t);
```

A.3 Sandproductionrate

```
1
2  function [sandArray,sandArrayNoise,stepSandArray] = ...
       sandproductionrate(initial_mdot,days,type,k)
3
4  sandArray = [initial_mdot];
5
6
7  if (type == 'exp')
8      for n = 1:days
9       sandArray = [sandArray initial_mdot*(0.5+exp(k*n))];
10     end
11
12     stepSandArray = [];
13     step = 1;
14
15     for n = 1:days
16        stepSandArray = [stepSandArray sandArray(step)];
17        if mod(n,30) == 0
18            step = n
19
20        end
21
22     end
23 end
24
25 if (type == 'log')
26     for n = 1:days
27      newVal = (10*initial_mdot)/(1+exp(-k*(n-days/2)));
28      sandArray = [sandArray newVal];
29     end
30
31     stepSandArray = [];
32     step = 1;
33
34     for n = 1:days
35        stepSandArray = [stepSandArray sandArray(step)];
36        if mod(n,30) == 0
37            step = n
38
39        end
40
41     end
42 end
43
44 sandArrayNoise = sandArray + 0.1 * rand(1, length(sandArray));
45
46 end
```

A.4 ModelSandArray

```
%% Model sand production rate with interpolation

function modelSandArray = modelSandProdRate(sandArray,dt)

modelSandArray = []

days = length(stepSandArray)

modelSandArray(1) = sandArray(1);
ΔSand = 0;

for i = 2:days

    if i ≤ dt
       modelSandArray(i) = sandArray(i);
    end

    modelSandArray(i) = modelSandArray(i-1);

    if mod(i,dt) == 0
        ΔSand = (sandArray(i) - sandArray(i-dt+1))/dt;
    end

    modelSandArray(i) = modelSandArray(i-1) + ΔSand;

    if mod(i,dt) == 0
        modelSandArray(i) = sandArray(i)
    end

end



figure(1)
plot(1:500,modelSandArray(1:500),'b')
hold on
plot(1:500,sandArray(1:500),'r')
legend({'Model Sand Prod Rate','Sample Sand Prod Rate'})
title('Sand production rates, modelled, sampled and true')

end
```

A.5 WellPlantModel

```


function [xk,zk] = WellPlantModel(dx0,z0,u0,par)


```

```
7       %% Parameters
8       %number of wells
9       n_w = par.n_w; %[]
10      %gas constant
11      R = par.R; %[m3 Pa /K /mol]
12      %molecular weigth
13      Mw = par.Mw; %[kg/mol?]
14
15      %properties
16      %density of oil - dim:  nwells x 1
17      rho_o = par.rho_o; %[kg/m3]
18      %riser oil density
19      rho_ro = par.rho_ro;%[kg/m3]
20      %1cP oil viscosity
21      mu_oil = par.mu_oil;% [Pa s]
22
23      %project
24      %well parameters - dim:  nwells x 1
25      L_w = par.L_w; %[m]
26      H_w = par.H_w; %[m]
27      D_w = par.D_w; %[m]
28      A_w = par.A_w;%[m2]
29
30      %well below injection - [m]
31      L_bh = par.L_bh;
32      H_bh = par.H_bh;
33      D_bh = par.D_bh;
34      A_bh = par.A_bh;%[m2]
35
36      %annulus - [m]
37      H_a = par.H_a;
38      V_a = par.V_a; %[m3]
39
40      %riser - [m]
41      L_r = par.L_r;
42      H_r = par.H_r;
43      D_r = par.D_r;
44      A_r = par.A_r;%[m2]
45
46      %injection valve characteristics  - dim:  nwells x 1
47      C_iv = par.C_iv;%[m2]
48      %production valve characteristics  - dim:  nwells x 1
49      C_pc = par.C_pc;%[m2]
50      %riser valve characteristics
51      C_pr = par.C_pr;%[m2]
52
53      %% For erosion model
54      % Sand
55      d_p = par.d_p; %[m] particle diameter
56
57      % Choke
58      K = par.K; %[-] material erosion constant
59      rho_t = par.rho_t; %[kg/m3] sensity CS
60      r = par.r; %[m] radius of curvature
61      D = par.D; %[m] Gap between body and cage
62      H = par.H; %[m] Height of gallery
63
```

```
64      % Constants
65      C_unit = par.C_unit; % Unit conversion factor: now in mm/s
66      C_1 = par.C_1; %[-] Model/geometry factor
67      n = par.n; %[-] Velocity coefficient
68      GF = par.GF; %[-] Geometry factor
69
70      % Precalculations of erosion in choke:
71      alpha = par.alpha;
72      F = par.F;
73      A_g = par.A_g; %[m2] Effective gallery area
74      G = 1; % THIS MUST BE CHANGED
75      ER_constant = par.ER_constant;
76
77      gma = d_p./D;
78
79      %% CasADi Variables
80      %%%%%%%%%%%%%%%%%%%%%
81      % Differential states
82      %%%%%%%%%%%%%%%%%%%%%
83
84      %erosion rate
85      ER = MX.sym('ER',n_w); %[s] x(1-3)
86
87      %%%%%%%%%%%%%%%%%%%%%
88      % Algebraic states
89      %%%%%%%%%%%%%%%%%%%%%
90
91      %pressure - annulus
92      p_ai = MX.sym('p_ai',n_w);%[bar] z(1-3)
93      %pressure - well head
94      p_wh = MX.sym('p_wh',n_w);%[bar] z(4-6)
95      %pressure - injection point
96      p_wi = MX.sym('p_wi',n_w);%[bar] z(7-9)
97      %pressure - below injection point (bottom hole)
98      p_bh = MX.sym('p_bh',n_w);%[bar] z(10-12)
99      %density - annulus
100     rho_ai = MX.sym('rho_ai',n_w);%[100 kg/m3] z(13-15)
101     %mixture density in tubing
102     rho_m = MX.sym('rho_m',n_w);%[100 kg/m3] z(16-18)
103     %well injection flow rate
104     w_iv = MX.sym('w_iv',n_w);%[kg/s] z(19-21)
105     %wellhead total production rate
106     w_pc = MX.sym('w_pc',n_w);%[kg/s] z(22-24)
107     %wellhead gas production rate
108     w_pg = MX.sym('w_pg',n_w);%[kg/s] z(25-27)
109     %wellhead oil production rate
110     w_po = MX.sym('w_po',n_w);%[kg/s] z(28-30)
111     %oil rate from reservoir
112     w_ro = MX.sym('w_ro',n_w);%[kg/s] z(31-33)
113     %gas rate from reservoir
114     w_rg = MX.sym('w_rg',n_w);%[0.1 kg/s] z(34-36)
115     %riser head pressure
116     p_rh = MX.sym('p_rh',1);%[bar] z(37)
117     %mixture density in riser
118     rho_r =  MX.sym('rho_r',1);%[100 kg/m3] z(38)
119     %manifold pressure
120     p_m = MX.sym('p_m',1);%[bar] z(39)
```

```
121     %riser head total production rate
122     w_pr = MX.sym('w_pr',1);%[kg/s] z(40)
123     %riser head total oil production rate
124     w_to = MX.sym('w_to',1);%[kg/s] z(41)
125     %riser head total gas production rate
126     w_tg = MX.sym('w_tg',1);%[kg/s] z(42)
127     %gas holdup @ annulus
128     m_ga = MX.sym('m_ga',n_w);%[ton] z(43-45)
129     %gas holdup @ well
130     m_gt = MX.sym('m_gt',n_w);%[ton] z(46-48)
131     %oil holdup @ well
132     m_ot = MX.sym('m_ot',n_w);%[ton] z(49-51)
133     %gas holdup @ riser
134     m_gr = MX.sym('m_gr',1);%[ton] z(52)
135     %oil holdup @ riser
136     m_or = MX.sym('m_or',1);%[ton] z(53)
137     %particle impact velocity
138     V_p = MX.sym('V_p',n_w);%       z(54-56)
139     %mixed dynamic viscosity
140     mu_f = MX.sym('mu_f',n_w);%     z(57-59)
141     %g1
142     g1 = MX.sym('g1',n_w);%         z(60-62)
143
144     %%%%%%%%%%%%%%%%%%%%%
145     % Control inputs
146     %%%%%%%%%%%%%%%%%%%%%
147
148     %gas lift rate
149     w_gl = MX.sym('w_gl',n_w); %[kg/s]
150
151     %%%%%%%%%%%%%%%%%%%%%
152     % Parameters
153     %%%%%%%%%%%%%%%%%%%%%
154
155     %reservoir pressure
156     p_res = MX.sym('p_res',n_w);
157     %productivity index
158     PI = MX.sym('PI',n_w); %[kg s^-1 bar-1]
159     %GasOil ratio
160     GOR = MX.sym('GOR',n_w); %[kg/kg]
161     %Annulus temperature
162     T_a = MX.sym('T_a',n_w); %[oC]
163     %well temperature
164     T_w = MX.sym('T_w',n_w); %[oC]
165     %riser temperature
166     T_r = MX.sym('T_r',1); %[oC]
167     %separator pressure
168     p_s = MX.sym('p_s',1); %[bar]
169
170
171     %time transformation: CASADI integrates always from 0 to 1 and the ...
            USER does the time
172     %scaling with T.
173     T = MX.sym('T',1); %[s]
174
175
176     %% Modeling
```

```
177     %gas fraction (mass) of the well holdup - avoiding zero division
178     xGw = (m_gt.*1e3./max(1e-3,(m_gt.*1e3+m_ot.*1e3)));
179     xOw = 1 - xGw;
180     %gas fraction (mass) of the riser holdup
181     xGr = (m_gr.*1e3./(m_gr.*1e3+m_or.*1e3));
182     xOr = 1 - xGr;
183
184     % ===================================
185     %     Well model
186     % ===================================
187     % algebraic equations (all symbolic)
188     %annulus pressure - %g = 9.81
189     f1 = -p_ai.*1e5 + ((R.*T_a./(V_a.*Mw) + ...
            9.81.*H_a./V_a).*m_ga.*1e3) + ...
            (Mw./(R.*T_a).*((R.*T_a./(V_a.*Mw) + ...
            9.81.*H_a./V_a).*m_ga.*1e3)).*9.81.*H_a;
190     %well head pressure
191     f2 = -p_wh.*1e5 + ((R.*T_w./Mw).*(m_gt.*1e3./(L_w.*A_w + ...
            L_bh.*A_bh - m_ot.*1e3./rho_o))) - ((m_gt.*1e3+m_ot.*1e3 ...
            )./(L_w.*A_w)).*9.81.*H_w/2;
192     %well injection point pressure
193     f3 = -p_wi.*1e5 + (p_wh.*1e5 + ...
            9.81./(A_w.*L_w).*max(0,(m_ot.*1e3+m_gt.*1e3-rho_o.*L_bh.*A_bh)).*H_w) ...
            + (128.*mu_oil.*L_w.*w_pc./(3.14.*D_w.^4.*((m_gt.*1e3
194     + m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o)./(m_ot.*1e3.*p_wh.*1e5.*Mw + ...
            rho_o.*R.*T_w.*m_gt.*1e3)));
195     %bottom hole pressure
196     f4 = -p_bh.*1e5 + (p_wi.*1e5 + rho_o.*9.81.*H_bh + ...
            128.*mu_oil.*L_bh.*w_ro./(3.14.*D_bh.^4.*rho_o));
197     %gas density in annulus
198     f5 = -rho_ai.*1e2 +(Mw./(R.*T_a).*p_ai.*1e5);
199     %fluid mixture density in well
200     f6 = -rho_m.*1e2 + ((m_gt.*1e3 + ...
            m_ot.*1e3).*p_wh.*1e5.*Mw.*rho_o)./(m_ot.*1e3.*p_wh.*1e5.*Mw + ...
            rho_o.*R.*T_w.*m_gt.*1e3);
201     %well injection flow rate
202     f7 = -w_iv + C_iv.*sqrt(rho_ai.*1e2.*max(0,(p_ai.*1e5 - p_wi.*1e5)));
203     %wellhead prodution rate
204     f8 = -w_pc + 1.*C_pc.*sqrt(rho_m.*1e2.*max(0,(p_wh.*1e5 - p_m.*1e5)));
205     %wellhead gas production rate
206     f9 = -w_pg + xGw.*w_pc;
207     %wellhead oil prodution rate
208     f10 = -w_po + xOw.*w_pc;
209     %oil from reservoir flowrate
210     f11 = -w_ro + PI.*1e-6.*(p_res.*1e5 - p_bh.*1e5);
211     %gas from reservoir production rate
212     f12 = -w_rg.*1e-1 + GOR.*w_ro;
213     %riser head pressure
214     f13 = -p_rh.*1e5 + ((R.*T_r./Mw).*(m_gr.*1e3./(L_r.*A_r))) - ...
            ((m_gr.*1e3+m_or.*1e3 )./(L_r.*A_r)).*9.81.*H_r/2;
215     %riser density
216     f14 = -rho_r.*1e2 + ((m_gr.*1e3 + ...
            m_or.*1e3).*p_rh.*1e5.*Mw.*rho_ro)./(m_or.*1e3.*p_rh.*1e5.*Mw ...
            + rho_ro.*R.*T_r.*m_gr.*1e3);
217     %manifold pressure
218     f15 = -p_m.*1e5 + (p_rh.*1e5 + ...
            9.81./(A_r.*L_r).*(m_or.*1e3+m_gr.*1e3).*H_r) + ...
```

```
        (128.*mu_oil.*L_r.*w_pr./(3.14.*D_r.^4.*((m_gr.*1e3 + ...
        m_or.*1e3).*p_rh.*1e5.*Mw.*rho_ro)./(m_or.*1e3.*p_rh.*1e5.*Mw ...
        + rho_ro.*R.*T_r.*m_gr.*1e3)));
    %total production rate of well
    f16 = -w_pr + 1.*C_pr.*sqrt(rho_r.*1e2.*(p_rh.*1e5 - p_s.*1e5));
    %oil total production rate
    f17 = -w_to + xOr.*w_pr;
    %gas total production rate
    f18 = -w_tg + xGr.*w_pr;
    % setting mass balances as algebraic equations since the
    % dynamics of ER is on a much larger time scale
    f19 = (w_gl - w_iv).*1e-3;
    f20 = (w_iv + w_rg.*1e-1 - w_pg).*1e-3;
    f21 = (w_ro - w_po).*1e-3;
    f22 = (sum(w_pg) - w_tg).*1e-3 ;
    f23 = (sum(w_po) - w_to).*1e-3 ;
    f24 = - V_p + 3/(4*A_g)*(w_po/rho_o + R*T_w.*w_pg./(p_wh.*10^5*Mw));
    f25 = - mu_f + mu_oil.*(w_po/rho_o)./(w_po./rho_o + ...
        R.*T_w.*w_pg./(p_wh.*10^5*Mw));
    % Assuming that gamma < 0 (checked in main)
    f26 = -g1 + gma/0.1;

    % differential equations - (all symbolic) - [ton]
    % Erosion rate
    df1 =  ER_constant.*g1.*(V_p).^n;

    %% Building system for integration
    % Form the DAE system
    diff = vertcat(df1);
    alg = vertcat(f1,f2,f3,f4,f5,f6,f7,f8,f9,f10,f11,f12,f13,f14,f15,f16
    ,f17,f18,f19,f20,f21,f22,f23,f24,f25,f26);

    % give parameter values
    alg = substitute(alg,p_res,par.p_res);
    alg = substitute(alg,p_s,par.p_s);
    alg = substitute(alg,T_a,par.T_a);
    alg = substitute(alg,T_w,par.T_w);
    alg = substitute(alg,T_r,par.T_r);

    diff = substitute(diff,p_res,par.p_res);
    diff = substitute(diff,T_w,par.T_w);

    % concatenate the differential and algebraic states
    x_var = vertcat(ER);
    z_var = vertcat(p_ai,p_wh,p_wi,p_bh,rho_ai,rho_m,w_iv,
    w_pc,w_pg,w_po,w_ro,w_rg,p_rh,rho_r,p_m,...
        w_pr,w_to,w_tg,m_ga,m_gt,m_ot,m_gr,m_or,V_p, mu_f,g1);
    p_var = vertcat(w_gl,GOR,PI,T);

    %end modeling
    %% Casadi commands for integration
    %declaring function in standard DAE form (scaled time)
    dae = struct('x',x_var,'z',z_var,'p',p_var,'ode',T*diff,'alg',alg);

    %calling the integrator, the necessary inputs are: label; ...
        integrator; function with IO scheme of a DAE (formalized); ...
        struct (options)
```

```
270     F = integrator('F','idas',dae);
271
272     %integration results
273     Fend = F('x0',dx0,'z0',z0,'p',[u0;par.GOR;par.PI;par.T]);
274
275     %extracting the results (from symbolic to numerical)
276     xk = full(Fend.xf);
277     zk = full(Fend.zf);
278 end
```

A.6 MainErodingValves

```
1
2  %%
3  %
4
5  %
6  clear
7  close all
8  clc
9
10 %noise --> For reproducibility
11
12 rng(1)
13
14 %%
15
16 %% Initializing the table to store multiple time series:
17
18 nTimeseries = 200;
19 dataSz = [nTimeseries 10];
20 varNames = {'Num','uArray', 'yMeas','erosionArray', 'H', 'x0', 'xk', ...
       'yk', 'z0', 'zk'};
21 varTypes = {'double', 'cell', 'cell', 'cell', 'cell', 'cell', 'cell', ...
       'cell', 'cell', 'cell'};
22 Data = table('Size', dataSz, 'VariableNames', varNames, ...
       'VariableTypes', varTypes);
23
24
25
26 %% Model initialization
27 sandArray = sandproductionrate(0.01,500);
28
29 for i_timeseries = 1:nTimeseries
30
31
32
33 % initial condition (pre-computed)
34 [x0,z0,u0] = InitialConditionGasLift_5;
35
36 % model parameters
37 par = ParametersGasLift(1,sandArray);
38
39 %states to measurement mapping function
40 H = zeros(16,length(z0));
41 %pai - annulus pressure, well 1-3
42 H(1,1) = 1;
43 H(2,2) = 1;
44 H(3,3) = 1;
45 %pwh - well head pressure, well 1-3
46 H(4,4) = 1;
47 H(5,5) = 1;
48 H(6,6) = 1;
49 %wro - wellhead gas production rate, well 1-3
50 H(7,25) = 1;
```

```
H(8,26) = 1;
H(9,27) = 1;
%wrg - wellhead oil production rate, well 1-3
H(10,28) = 1;
H(11,29) = 1;
H(12,30) = 1;
%prh - riser head pressure
H(13,37) = 1;
%pm - manifold pressure
H(14,39) = 1;
%wto - riser head total oil production rate
H(15,41) = 1;
%wtg - riser head total gas production rate
H(16,42) = 1;


%% Simulation parameters

% number of simulation steps
nSim = 500; % time_total = 3600*24*500; %[s]



[dx0,z0,u0] = InitialConditionGasLift_5

%sampling time /control interval /1 simulation iteration time
par.T = 3600*24; % [s]




%% Simulation initialization
xk = x0;
zk = z0;
uk = u0;
yk = H*z0;



%%


fprintf('Time series number: >>> %0.0f \n',i_timeseries)
%creating random array for the inputs
uArray = [];
%bounds on the inputs
uMin = 1.0;
uMax = 2.5;

for t = 0:nSim
    if rem(t,50) == 0 %every 50 days we change the inputs
        uk = (uMax - uMin).*rand(3,1) + uMin;
    end
    uArray = [uArray, uk];
end

% measurements (for plotting)
```

```
yMeas = yk;
erosionArray = xk;


for u = 1:nSim

    %fprintf('     iteration >>> %0.0f \n',t)

    % integrating the system
    [xk,zk] = WellPlantModel(xk,zk,uArray(:,u),par);
    par = ParametersGasLift(u,sandArray);
    par.T = 3600*24;
    % Adding noise to the measurements
    yMeas = [yMeas, H*zk + par.scale.*randn(length(yk),1)];
    erosionArray = [erosionArray, xk];
end


Data(i_timeseries, :) = {i_timeseries, uArray, yMeas, erosionArray, H, ...
    x0, xk, yk, z0, zk};

end

%%
filename = 'datamatrix_'+string(nTimeseries)+'.mat';
save(filename, 'Data')
%% Plotting
figure(1)


time = 0:1:500; %[days]


% system inputs
subplot(2,1,1)
    stairs(time,uArray(1,:));
    hold on
    stairs(time,uArray(2,:));
    stairs(time,uArray(3,:));

    legend('Well 1','Well 2','Well 3');

    ylim([0,2.2]);
    xlabel('Time [day]');
    ylabel('Gas lift rate [kg/s]');

% erosion
subplot(2,1,2);
    plot(time,transpose(Data.erosionArray{1,1}(1,:)));
    hold on
    plot(time,transpose(Data.erosionArray{1,1}(2,:)));
    plot(time,transpose(Data.erosionArray{1,1}(3,:)));
    ylim([0,50.2]);

    legend('Well 1','Well 2','Well 3');

    xlabel('Time [day]');
```

```
164     ylabel('Erosion [mm]');
165
166 figure(2)
167
168 % Pressure
169 subplot(2,1,1)
170     plot(time,yMeas(4,:));
171     hold on
172     plot(time,yMeas(5,:));
173     plot(time,yMeas(6,:));
174
175     legend('Well 1','Well 2','Well 3');
176
177     %ylim([0,2.2]);
178     xlabel('Time [day]');
179     ylabel('Well head pressure [bar]');
180
181 % erosion
182 subplot(2,1,2);
183     plot(time,yMeas(15,:)); %oil
184     %plot(time,yMeas(14,:)); %gas
185
186
187     xlabel('Time [day]');
188     ylabel('Flowrate [kg/s]');
```

Appendix B

Training and testing script for simulated data

B.1 Analysis script for Exponential data

To not include two other scripts that are almost identical, only the script used to generate the plots and models for exponential data will be included in the appendix.

```
1
2  %% Regression Analysis Script
3  % In this script there are two significant parts, one part that ...
       initialises
4  % and pre-processes the training data and one part that takes this ...
       data and
5  % feeds it to models for training and testing of machine learning methods.
6
7  %% Initialize training data
8
9  clc
10
11 %pai - annulus pressure, well 1-3
12
13 pai_1 = [];  %y1
14
15 %pwh - well head pressure, well 1-3
16
17 pwh_1 = []; %y4
18
19 %wro - wellhead gas production rate, well 1-3
20
21 wro_1 = []; %y7
22
23 %wrg - wellhead oil production rate, well 1-3
24
25 wrg_1 = []; %y10
```

```
26
27  %prh - riser head pressure
28
29  prh = []; %y13
30
31  %pm - manifold pressure
32
33  pm = []; %y14
34
35  %wto - riser head total oil production rate
36
37  wto = []; %y15
38
39  %wtg - riser head total gas production rate
40
41  wtg = [];  %y16
42
43  % Erosion rate of change
44
45  rocArray = []; %erosion well 1
46
47  % Gas Lift Rate
48
49  gLift = []; %control input well 1
50
51  % Sand Rate
52
53  sand = [];
54
55  % Time variable
56
57  time = 1:1:501;
58
59  varNames = {'Sand rate'; 'Annulus Pressure'; 'Well head pressure'; ...
        'Well head oil production rate'; 'Well head gas production rate';
60      'Riser head pressure';'Manifold pressure';'Riser head total oil ...
            production rate';
61      'Riser head total gas production rate';'Gas lift rate'};
62  %varNamesShorthand = {'P_A '; 'P_wh'; 'WH_OP'; 'WH_GP'; ...
        'P_RH';'P_M';'RH_totOP';'RH_totGP';'GLR','ErosROC'};
63
64  responseName = {'Erosion rate of change'};
65
66  [sandArray,sandArrayNoise,stepSandArray] = ...
        sandproductionrate(0.01,501,'log',0.015,50); % The function used ...
        for generating logit data
67
68  % [sandArray,sandArrayNoise,stepSandArray] =
69  % sandproductionrate(0.01,501,'exp',0.005,50); The function used to make
70  % exponential data
71  %%
72  for i = 1:100
73
74      % Adding 100 timerseries of yMeas to the end of eachother to give more
75      % training data
76
77      %yMeas
```

```
78      pai_1 = [pai_1 Data.yMeas{i,1}(1,:)];
79      pwh_1 = [pwh_1 Data.yMeas{i,1}(4,:)];
80      wro_1 = [wro_1 Data.yMeas{i,1}(7,:)];
81      wrg_1 = [wrg_1 Data.yMeas{i,1}(10,:)];
82      prh = [prh Data.yMeas{i,1}(13,:)];
83      pm = [pm Data.yMeas{i,1}(14,:)];
84      wto = [wto Data.yMeas{i,1}(15,:)];
85      wtg = [wtg Data.yMeas{i,1}(16,:)];
86      sand = [sand stepSandArray];
87      %uArray
88      gLift = [gLift Data.uArray{i,1}(1,:)];
89
90      %erosionArray - Calculate erosion rate of change
91
92
93      erosArray = Data.erosionArray{i,1}(1,:);
94      roc = diff(erosArray)./diff(time);
95      roc = [roc, roc(500)];
96      rocArray = [rocArray, roc];
97
98
99  end
100
101 %Data without sand, use if working with constant spr data set
102 %RegData = [];
103 %RegData = [transpose(pai_1), transpose(pwh_1), transpose(wro_1), ...
        transpose(wrg_1), transpose(prh), transpose(pm), transpose(wto), ...
        transpose(wtg), transpose(gLift), transpose(rocArray)];
104 %NNRegData = [transpose(pai_1), transpose(pwh_1), transpose(wro_1), ...
        transpose(wrg_1), transpose(prh), transpose(pm), transpose(wto), ...
        transpose(wtg), transpose(gLift)];
105 %NNRegResponse = [transpose(rocArray)];
106
107
108 %Data with sandproduction rate, use for non-constant spr
109 RegData = [];
110 RegData = [transpose(sand), transpose(pai_1), transpose(pwh_1), ...
        transpose(wro_1), transpose(wrg_1), transpose(prh), transpose(pm), ...
        transpose(wto), transpose(wtg), transpose(gLift), ...
        transpose(rocArray)];
111 NNRegData = [transpose(sand), transpose(pai_1), transpose(pwh_1), ...
        transpose(wro_1), transpose(wrg_1), transpose(prh), transpose(pm), ...
        transpose(wto), transpose(wtg), transpose(gLift)];
112 NNRegResponse = [transpose(rocArray)];
113
114 NNRegData = normalize(NNRegData);
115 NNRegResponse = normalize(NNRegResponse);
116 NormRegData = normalize(RegData);
117
118 %% Initialize Test Data
119
120 %pai - annulus pressure, well 1-3
121
122 pai_1_T = [];  %y1
123
124 %pwh - well head pressure, well 1-3
125
```

```
126 pwh_1_T = []; %y4
127
128 %wro - wellhead gas production rate, well 1-3
129
130 wro_1_T = []; %y7
131
132 %wrg - wellhead oil production rate, well 1-3
133
134 wrg_1_T = []; %y10
135
136 %prh - riser head pressure
137
138 prh_T = []; %y13
139
140 %pm - manifold pressure
141
142 pm_T = []; %y14
143
144 %wto - riser head total oil production rate
145
146 wto_T = []; %y15
147
148 %wtg - riser head total gas production rate
149
150 wtg_T = [];  %y16
151
152 % Erosion rate of change
153
154 rocArray_T = []; %erosion well 1
155 roc_T = [];
156
157 % Gas Lift Rate
158
159 gLift_T = []; %control input well 1
160
161
162 for i = 101:200
163
164     pai_1_T = [pai_1_T Data.yMeas{i,1}(1,:)];
165     pwh_1_T = [pwh_1_T Data.yMeas{i,1}(4,:)];
166     wro_1_T = [wro_1_T Data.yMeas{i,1}(7,:)];
167     wrg_1_T = [wrg_1_T Data.yMeas{i,1}(10,:)];
168     prh_T = [prh_T Data.yMeas{i,1}(13,:)];
169     pm_T = [pm_T Data.yMeas{i,1}(14,:)];
170     wto_T = [wto_T Data.yMeas{i,1}(15,:)];
171     wtg_T = [wtg_T Data.yMeas{i,1}(16,:)];
172
173     gLift_T = [gLift_T Data.uArray{i,1}(1,:)];
174
175
176
177     erosArray_T = Data.erosionArray{i,1}(1,:);
178     roc_T = [roc_T gradient(erosArray_T)];
179
180 end
181 %Use when testing performance of training data with spr
```

```
TestData = [transpose(sand), transpose(pai_1_T), transpose(pwh_1_T), ...
    transpose(wro_1_T), transpose(wrg_1_T), transpose(prh_T), ...
    transpose(pm_T), transpose(wto_T), transpose(wtg_T), ...
    transpose(gLift_T)];

%Use when testing performance of training data without spr
%TestData = [transpose(pai_1_T), transpose(pwh_1_T), ...
    transpose(wro_1_T), transpose(wrg_1_T), transpose(prh_T), ...
    transpose(pm_T), transpose(wto_T), transpose(wtg_T), ...
    transpose(gLift_T)];

NormTestResponse = transpose(normalize(roc_T));

NormTestData = normalize(TestData);


%% Training and Testing Stepwise Model

%Without PCA
[StepwiseModel, StepwiseValidationRMSE] = ...
    trainStepwiseRegression(NNRegData,NNRegResponse);

%With PCA
[StepwiseModelPCA, StepwisePCAValidationRMSE] = ...
    trainStepwiseRegressionPCA(NNRegData,NNRegResponse);

yhat_stepwise = StepwiseModel.predictFcn(NormTestData);
yhat_stepwise_pca = StepwiseModelPCA.predictFcn(NormTestData);


immse(yhat_stepwise,NormTestResponse)
immse(yhat_stepwise_pca,NormTestResponse)


%% Plot sample results
figure(1)

plot(1:500,yhat_stepwise(1:500),'g')
hold on
plot(1:500,yhat_stepwise_pca(1:500),'b')
hold on
plot(1:500,NormTestResponse(1:500),'r')
title('Stepwise regression with and without PCA')
legend('Predicted w/ stepwise','Predicted w/ stepwise+pca','True rate')

    %ylim([0,2.2]);
xlabel('Time [day]');
ylabel('Rate of change');


%% Training and Testing Tree Model

%Without PCA
[OptTreeModel, OptTreeValidationRMSE] = ...
    trainOptTreeRegression(NNRegData,NNRegResponse);

%With PCA
```

```
[OptTreeModelPCA, OptTreePCAValidationRMSE] = ...
    trainOptTreeRegressionPCA(NNRegData,NNRegResponse);

yhat_opttree = OptTreeModel.predictFcn(NormTestData);
yhat_opttree_pca = OptTreeModelPCA.predictFcn(NormTestData);


immse(yhat_opttree,NormTestResponse)
immse(yhat_opttree_pca,NormTestResponse)

%% Plot sample results
figure(2)

plot(1:500,yhat_opttree(1:500),'g')
hold on
plot(1:500,yhat_opttree_pca(1:500),'b')
hold on
plot(1:500,NormTestResponse(1:500),'r')
title('Optimised Regression Tree with and without PCA')
legend('Predicted w/ tree','Predicted w/ tree+pca','True rate')

    %ylim([0,2.2]);
xlabel('Time [day]');
ylabel('Rate of change');

%% Training and Testing SVM Model


%Without PCA
[SvmMedGausModel, SvmMedGausValidationRMSE] = ...
    trainSvmMedGausRegression(NNRegData,NNRegResponse);

%With PCA
[SvmMedGausModelPCA, SvmMedGausPCAValidationRMSE] = ...
    trainSvmMedGausRegressionPCA(NNRegData,NNRegResponse);

%Independent Test
yhat_svm = SvmMedGausModel.predictFcn(NormTestData);
yhat_svm_pca = SvmMedGausModelPCA.predictFcn(NormTestData);

%MSE calculation
immse(yhat_svm,NormTestResponse)
immse(yhat_svm_pca,NormTestResponse)


%% Plot sample results
figure(3)

plot(1:500,yhat_svm(1:500),'g')
hold on
plot(1:500,yhat_svm_pca(1:500),'b')
hold on
plot(1:500,NormTestResponse(1:500),'r')

legend('Predicted w/ Gaussian SVR','Predicted w/ Gaussian ...
    SVR+PCA','True rate')
title('Medium Gaussian Support Vector Regression')
```

```
subtitle('with and without PCA')

    %ylim([0,2.2]);
xlabel('Time [day]');
ylabel('Rate of change');

%% Training and Testing Ensemble Model


%Without PCA
[EnsembleModel, EnsembleValidationRMSE] = ...
    trainEnsembleRegression(NNRegData,NNRegResponse);

%With PCA
[EnsembleModelPCA, EnsemblePCAValidationRMSE] = ...
    trainEnsembleRegressionPCA(NNRegData,NNRegResponse);

%Independent Test
yhat_ensemble = EnsembleModel.predictFcn(NormTestData);
yhat_ensemble_pca = EnsembleModelPCA.predictFcn(NormTestData);

%MSE calculation
immse(yhat_ensemble,NormTestResponse)
immse(yhat_ensemble_pca,NormTestResponse)

%% Plot sample results
figure(4)

plot(1:500,yhat_ensemble(1:500),'g')
hold on
plot(1:500,yhat_ensemble_pca(1:500),'b')
hold on
plot(1:500,NormTestResponse(1:500),'r')
title('Optimised Ensemble of trees with and without PCA')
legend('Predicted w/ ensemble','Predicted w/ ensemble+pca','True rate')

    %ylim([0,2.2]);
xlabel('Time [day]');
ylabel('Rate of change');

%% PLSR

X = NormRegData(:,1:10);
y = NormRegData(:,11);

[XL,yl,XS,YS,beta,PCTVAR,mse,stats] = plsregress(X,y,5);

%plot(1:9,cumsum(100*PCTVAR(2,:)),'-bo');
%xlabel('Number of PLS components');
%ylabel('Percent Variance Explained in y');

yfit = [ones(size(X,1),1) X]*beta;
residuals = y - yfit;
%stem(residuals)


Xtestplot = NormTestData(1:500,1:10);
```

```
yfitTestPlot = [ones(size(Xtestplot,1),1) Xtestplot]*beta;

immse(yfit,NormTestResponse)

figure(5)

plot(1:500,yfitTestPlot(1:500),'b')
hold on
plot(1:500,NormTestResponse(1:500),'r')
title('Partial Least Squares')
legend('PLSR','True rate')

    %ylim([0,2.2]);
xlabel('Time [day]');
ylabel('Rate of change');


figure(6)
plot(1:5 ,cumsum(100*PCTVAR(2,:)),'-bo');
xlabel('Number of PLS components');
ylabel('Percent Variance Explained in Y');


figure(7)
plot(1:10,stats.W,'-');
xlabel('Variable');
ylabel('PLS Weight');
legend({'1st Component' '2nd Component' '3rd Component'},'location','NW');
%% GLM Modelling

[b,dev,stats] = glmfit(NNRegData,NNRegResponse,'normal');
yhat = glmval(b,NormTestData,'identity');

mse_glm = immse(yhat,NormTestResponse)


figure(8)
plot(1:500,NormTestResponse(1:500),'r',1:500,yhat(1:500),'b')
title('GLM predictions and True values')
legend({'Measured','GLM Prediction'})


%% Neural Network Modelling (Non-linear Input/Output)
x1 = transpose(NormTestData(4:50100,:));
xi1 = transpose(NormTestData(1:3,:));


[y1,xf1] = NLIOnet(x1,xi1)

immse(transpose(y1),NormTestResponse(4:50100))

figure(9)
plot(4:500,NormTestResponse(4:500),'r',4:500,y1(4:500),'b')
title('Non-linear I/O Neural Network')
legend({'Measured','Neural Network Prediction'})

%% Optimised Linear Model
```

```
hyperopts = struct('AcquisitionFunctionName','expected-improvement-plus');
[Mdl,FitInfo,HyperparameterOptimizationResults] = ...
    fitrlinear(NNRegData,NNRegResponse,'OptimizeHyperparameters','auto','HyperparameterOpt


yfit_regularised = predict(Mdl,NormTestData);

immse(yfit_regularised,NormTestResponse)

figure(10)
plot(1:500,NormTestResponse(1:500),'r',1:500,yfit_regularised(1:500),'b')
title('Optimised Regularised Linear Regression')
subtitle('Least squares, expected-improvement, ridge regularisation')
legend({'Measured','Regularised Linear Regression'})


%% Function Declarations of training models for stepwise, optimised ...
    tree, medium gaussian svr and optimised ensemble are collapsed for ...
    space reasons, detailed documentation is available from Mathworks, ...
    the algorithms chosen for training, loss functions, validation etc ...
    is given earlier in the thesis.

% stepwise functions w and w/o pca

function [trainedModel, validationRMSE] = ...
    trainStepwiseRegression(trainingData, responseData)
function [trainedModel, validationRMSE] = ...
    trainStepwiseRegressionPCA(trainingData, responseData)

% optimised tree functions w and w/o pca

function [trainedModel, validationRMSE] = ...
    trainOptTreeRegression(trainingData, responseData)
function [trainedModel, validationRMSE] = ...
    trainOptTreeRegressionPCA(trainingData, responseData)

% medium gaussian svm w and w/o pca

function [trainedModel, validationRMSE] = ...
    trainSvmMedGausRegression(trainingData, responseData)
function [trainedModel, validationRMSE] = ...
    trainSvmMedGausRegressionPCA(trainingData, responseData)

% optimised ensemble functions w and w/o pca

function [trainedModel, validationRMSE] = ...
    trainEnsembleRegression(trainingData, responseData)
function [trainedModel, validationRMSE] = ...
    trainEnsembleRegressionPCA(trainingData, responseData)
```

Appendix C

Experimental Data

In this section of the appendix, possibly interesting information about the experimental data that will be given.

C.1 Variable List

These variables were recorded by LabView during experimental runs of the erosion rig, not all are used in prediction but for completeness everything that was recorded is listed here.

1. Date
2. Time (clock)
3. Relative time [s]
4. Air flow set point [sl/min] (well 1-3)
5. Air flow measured [sl/min] (well 1-3)
6. Fluid flow rate set point control valve [l/min] (well 1-3)
7. Fluid flow rate measured control valve [l/min] (well 1-3)
8. Differential pressure (dP) [mbar] (well 1-3)
9. Pressure setpoint control valve [mbar] (well 1-3)
10. Pressure measured control valve [mbar] (well 1-3)
11. Temperature measured [C°] (well 1-3)
12. Pressure measured topside [mbar] (well 1-3)
13. Control currents [A] (pump, control valves)

C.2 Script for making training and test sets from imported data

This script takes the tables of data recorded by LabView from the sensors around the rig. The script turns these 6 tables from six experimental runs into a training set consisting of the measured data from two wells and a testing set with the data from the third well.

```
%% Creating a Full dataset of experimental data
%In this script, the code takes several tables, imported from .txt files
%output from LabView to

%% Working with experimental Data

%Working Notes on where to limit data Q2: 1000:13000, Q3: 3500:5000, ...
    Q4: 500:3000 Q5: 500:2500, Q6:
%250:3000, Q9: 500:2000
%Throw away Well 1 for Q4

% Q1 1:12000, Q2 12000:13500, Q4 13500:16000, Q5 16000:18000, Q6
% 18000:20750, Q9 20750:22250

%% Initialise start and end of each variable
Q2start = 1000;
Q2end = 13000;
Q3start = 3500;
Q3end = 5000;
Q4start = 500;
Q4end = 3000;
Q5start = 500;
Q5end = 2500;
Q6start = 250;
Q6end = 3000;
Q9start = 500;
Q9end = 2000;

%% Adding data from experiments Q2, initialising data matrices

table = Q2;

DataColumnsWell1 = [3, 11, 16, 20, 25, 28, 31, 34];
DataColumnsWell2 = [3, 13, 17, 22, 26, 29, 32, 34];
DataColumnsWell3 = [3, 15, 18, 24, 27, 30, 33, 34];


Well1Data = [];
Well2Data = [];
Well3Data = [];

for i = 1:length(DataColumnsWell1)

    vector1 = table2array(table(:,DataColumnsWell1(i)));
    vector2 = table2array(table(:,DataColumnsWell2(i)));
    vector3 = table2array(table(:,DataColumnsWell3(i)));


```

```
    Well1Data = [Well1Data rescale(vector1,0,10)];
    Well2Data = [Well2Data rescale(vector2,0,10)];
    Well3Data = [Well3Data rescale(vector3,0,10)];


end

Well1Data = Well1Data(Q2start:Q2end,:);
Well2Data = Well2Data(Q2start:Q2end,:);
Well3Data = Well3Data(Q2start:Q2end,:);

q2w1 = Well1Data;
q2w2 = Well2Data;
q2w3 = Well3Data;


%% Repeating for Q3 - identical except no initialization of array

table = Q3;

Well1Data = [];
Well2Data = [];
Well3Data = [];

for i = 1:length(DataColumnsWell1)

    vector1 = table2array(table(:,DataColumnsWell1(i)));
    vector2 = table2array(table(:,DataColumnsWell2(i)));
    vector3 = table2array(table(:,DataColumnsWell3(i)));


    Well1Data = [Well1Data rescale(vector1,0,10)];
    Well2Data = [Well2Data rescale(vector2,0,10)];
    Well3Data = [Well3Data rescale(vector3,0,10)];


end

Well1Data = Well1Data(Q3start:Q3end,:);
Well2Data = Well2Data(Q3start:Q3end,:);
Well3Data = Well3Data(Q3start:Q3end,:);

q3w1 = Well1Data;
q3w2 = Well2Data;
q3w3 = Well3Data;


%% Repeating for Q4 - identical except no initialization of array

table = Q4;

Well1Data = [];
Well2Data = [];
Well3Data = [];

for i = 1:length(DataColumnsWell1)

```

```
105     vector1 = table2array(table(:,DataColumnsWell1(i)));
106     vector2 = table2array(table(:,DataColumnsWell2(i)));
107     vector3 = table2array(table(:,DataColumnsWell3(i)));
108 
109 
110     Well1Data = [Well1Data rescale(vector1,0,10)];
111     Well2Data = [Well2Data rescale(vector2,0,10)];
112     Well3Data = [Well3Data rescale(vector3,0,10)];
113 
114 
115 end
116 % Because well 3 is broken here we copy well 2,
117 
118 Well1Data = Well2Data;
119 
120 Well1Data = Well1Data(Q4start:Q4end,:);
121 Well2Data = Well2Data(Q4start:Q4end,:);
122 Well3Data = Well3Data(Q4start:Q4end,:);
123 
124 q4w1 = Well1Data;
125 q4w2 = Well2Data;
126 q4w3 = Well3Data;
127 
128 %% Repeating for Q5 - identical except no initialization of array
129 
130 table = Q5;
131 
132 
133 Well1Data = [];
134 Well2Data = [];
135 Well3Data = [];
136 
137 for i = 1:length(DataColumnsWell1)
138 
139     vector1 = table2array(table(:,DataColumnsWell1(i)));
140     vector2 = table2array(table(:,DataColumnsWell2(i)));
141     vector3 = table2array(table(:,DataColumnsWell3(i)));
142 
143 
144     Well1Data = [Well1Data rescale(vector1,0,10)];
145     Well2Data = [Well2Data rescale(vector2,0,10)];
146     Well3Data = [Well3Data rescale(vector3,0,10)];
147 
148 
149 end
150 
151 Well1Data = Well1Data(Q5start:Q5end,:);
152 Well2Data = Well2Data(Q5start:Q5end,:);
153 Well3Data = Well3Data(Q5start:Q5end,:);
154 
155 q5w1 = Well1Data;
156 q5w2 = Well2Data;
157 q5w3 = Well3Data;
158 
159 %% Repeating for Q6 - identical except no initialization of array
160 
161 table = Q6;
```

```
162
163 Well1Data = [];
164 Well2Data = [];
165 Well3Data = [];
166
167 for i = 1:length(DataColumnsWell1)
168
169     vector1 = table2array(table(:,DataColumnsWell1(i)));
170     vector2 = table2array(table(:,DataColumnsWell2(i)));
171     vector3 = table2array(table(:,DataColumnsWell3(i)));
172
173
174     Well1Data = [Well1Data rescale(vector1,0,10)];
175     Well2Data = [Well2Data rescale(vector2,0,10)];
176     Well3Data = [Well3Data rescale(vector3,0,10)];
177
178
179 end
180
181 Well1Data = Well1Data(Q6start:Q6end,:);
182 Well2Data = Well2Data(Q6start:Q6end,:);
183 Well3Data = Well3Data(Q6start:Q6end,:);
184
185 q6w1 = Well1Data;
186 q6w2 = Well2Data;
187 q6w3 = Well3Data;
188
189 %% Repeating for Q9 - identical except no initialization of array
190
191 table = Q9;
192
193
194 Well1Data = [];
195 Well2Data = [];
196 Well3Data = [];
197
198 for i = 1:length(DataColumnsWell1)
199
200     vector1 = table2array(table(:,DataColumnsWell1(i)));
201     vector2 = table2array(table(:,DataColumnsWell2(i)));
202     vector3 = table2array(table(:,DataColumnsWell3(i)));
203
204
205     Well1Data = [Well1Data rescale(vector1,0,10)];
206     Well2Data = [Well2Data rescale(vector2,0,10)];
207     Well3Data = [Well3Data rescale(vector3,0,10)];
208
209
210 end
211
212 Well1Data = Well1Data(Q9start:Q9end,:);
213 Well2Data = Well2Data(Q9start:Q9end,:);
214 Well3Data = Well3Data(Q9start:Q9end,:);
215
216 q9w1 = Well1Data;
217 q9w2 = Well2Data;
218 q9w3 = Well3Data;
```

```
219
220 %% Playing with scaling and normalisation
221
222 q2w1 = normalize(q2w1);
223 q2w2 = normalize(q2w2);
224 q2w3 = normalize(q2w3);
225
226 q3w1 = normalize(q3w1);
227 q3w2 = normalize(q3w2);
228 q3w3 = normalize(q3w3);
229
230 q4w1 = normalize(q4w1);
231 q4w2 = normalize(q4w2);
232 q4w3 = normalize(q4w3);
233
234 q5w1 = normalize(q5w1);
235 q5w2 = normalize(q5w2);
236 q5w3 = normalize(q5w3);
237
238 q6w1 = normalize(q6w1);
239 q6w2 = normalize(q6w2);
240 q6w3 = normalize(q6w3);
241
242 q9w1 = normalize(q9w1);
243 q9w2 = normalize(q9w2);
244 q9w3 = normalize(q9w3);
245
246 %% Combining all experiments
247 FullDataW1 = [];
248 FullDataW2 = [];
249 FullDataW3 = [];
250
251 FullDataW1 = [q9w1; q6w1; q5w1; q4w1; q3w1; q2w1];
252 FullDataW2 = [q9w2; q6w2; q5w2; q4w2; q3w2; q2w2];
253 FullDataW3 = [q9w3; q6w3; q5w3; q4w3; q3w3; q2w3];
254
255 %% Combining into test and training set, using 2 wells for training ...
        and 1 for testing
256
257 RegVars1 = FullDataW1(1:length(FullDataW1(:,3)),[1, 2, 4, 5, 6]);
258 RegVars2 = FullDataW2(1:length(FullDataW2(:,3)),[1, 2, 4, 5, 6]);
259 RegVars3 = FullDataW3(1:length(FullDataW3(:,3)),[1, 2, 4, 5, 6]);
260
261 RegResponse1 = movmean(FullDataW1(1:length(FullDataW1(:,3)),3),150);
262 RegResponse2 = movmean(FullDataW2(1:length(FullDataW2(:,3)),3),150);
263 RegResponse3 = movmean(FullDataW3(1:length(FullDataW3(:,3)),3),150);
264
265 TrainSetVars = [RegVars1; RegVars2];
266 TrainSetResponse = [RegResponse1; RegResponse2];
267
268 TestSetVars = RegVars3;
269 TestSetResponse = RegResponse3;
270
271 %% Plotting for show
272
273 figure(1)
274 plot(1:length(RegResponse1),RegResponse1,1:length(RegResponse2),RegResponse2,1:length(RegR
```

```
275 legend('Well 1','Well 2','Well 3')
```

Appendix D

Training and testing script for experimental data

D.1 Analysis script for experimental data

```

%% Modelling experimental data
% In this file I will make an effort to train and test models on the
% experimental data using various methods within the field of supervised
% statistical learning

%% GLM Modelling

[b,dev,stats] = glmfit(TrainSetVars,TrainSetResponse,'normal');
yfit = glmval(b,TestSetVars,'identity');

mse_glm = immse(yfit(5000:9000),TestSetResponse(5000:9000))

figure(1)
plot(5001:9000,TestSetResponse(5001:9000),'r')
hold on
plot(5001:9000,yfit(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
title('Stepwise prediction and True values')
subtitle('Mean Square Error of prediction: 0.1129')
legend({'Measured','Tree pred'})



%% Stepwise prediction

yfit = stepwise.predictFcn(TestSetVars);

mse_stepwise = immse(yfit(5000:9000),TestSetResponse(5000:9000))
```

```
30
31 figure(2)
32 plot(5001:9000,TestSetResponse(5001:9000),'r')
33 hold on
34 plot(5001:9000,yfit(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
35 title('Stepwise prediction and True values')
36 subtitle('Mean Square Error of prediction: 0.1069')
37 legend({'Measured','Stepwise pred'})
38
39 %% Tree prediction
40
41 yfit = opttree.predictFcn(TestSetVars);
42
43 mse_tree = immse(yfit(5000:9000),TestSetResponse(5000:9000))
44
45 figure(3)
46 plot(5001:9000,TestSetResponse(5001:9000),'r')
47 hold on
48 plot(5001:9000,yfit(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
49 title('Tree prediction and True values')
50 subtitle('Mean Square Error of prediction: 0.1072')
51 legend({'Measured','Tree pred'})
52
53 %% Ensemble
54
55 yfit = optensemble.predictFcn(TestSetVars);
56
57 mse_ensemble = immse(yfit(5000:9000),TestSetResponse(5000:9000))
58
59 figure(4)
60 plot(5001:9000,TestSetResponse(5001:9000),'r')
61 hold on
62 plot(5001:9000,yfit(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
63 title('Ensemble prediction and True values')
64 subtitle('Mean Square Error of prediction: 0.0798 (Bagging)')
65 legend({'Measured','Ensemble pred'})
66
67
68 %% SVM prediction
69
70
71 yfit = optsvm.predictFcn(TestSetVars);
72
73 mse_svm = immse(yfit(5000:9000),TestSetResponse(5000:9000))
74 %%
75 figure(5)
76 plot(5001:9000,TestSetResponse(5001:9000),'r')
77 hold on
78 plot(5001:9000,yfit(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
79 title('SVM prediction and True values')
80 subtitle('Mean Square Error of prediction: 0.1175')
81 legend({'Measured','SVM pred'})
82
83
84 %%  NARX prediction
85
86 NARX_ExperimentalData;
```

```
87
88  X = tonndata(TestSetVars(1:9000,:),false,false);
89  T = tonndata(TestSetResponse(1:9000),false,false);
90
91  [x,xi,ai,t] = preparets(net,X,{},T);
92
93  numTimesteps = size(x,2);
94  knownOutputTimesteps = 1:(numTimesteps-4000);
95  predictOutputTimesteps = (numTimesteps-3999):numTimesteps;
96  X1 = X(:,knownOutputTimesteps);
97  T1 = T(:,knownOutputTimesteps);
98  [x1,xio,aio] = preparets(net,X1,{},T1);
99  [y1,xfo,afo] = net(x1,xio,aio);
100
101 x2 = X(1,predictOutputTimesteps);
102 [netc,xic,aic] = closeloop(net,xfo,afo);
103 [y2,xfc,afc] = netc(x2,xic,aic);
104
105 y1 = cell2mat(y1);
106 y2 = cell2mat(y2);
107
108 mse_narx = immse(transpose(y2),TestSetResponse(5001:9000));
109
110 figure(6)
111 plot(7:5000,transpose(y1),'-o')
112 hold on
113 plot(5001:9000,transpose(y2(1:4000)),'-o')
114 hold on
115 plot(1:9000,TestSetResponse(1:9000),'r')
116 title('NARX Neural Network Prediction and true values')
117 subtitle('Mean Square Error of closed loop prediction: 0.0797')
118 legend({'Open Loop Prediction','Closed Loop Prediction','True Values'})
119
120 %% NLIO prediction
121
122 NLIOExperimentalData;
123
124 X = tonndata(TestSetVars(1:9000,:),false,false);
125 T = tonndata(TestSetResponse(1:9000),false,false);
126
127 [x,xi,ai,t] = preparets(net,X,T);
128
129 [y1,xf1] = NLIOnet(x1,xi1);
130
131
132
133 mse_nlio = immse(transpose(y1(5001:9000)),TestSetResponse(5001:9000));
134
135
136 figure(7)
137 plot(5001:9000,TestSetResponse(5001:9000),'r')
138 hold on
139 plot(5001:9000,y1(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
140 title('NLIO Neural Network Prediction and True values')
141 subtitle('Mean Square Error of prediction: 0.0709')
142 legend({'Measured','NLIO pred'})
143
```

```
%% PLSR

[XL,yl,XS,YS,beta,PCTVAR,MSE,stats] = ...
    plsregress(TrainSetVars,TrainSetResponse,4,'cv',4);


figure(8)
plot(1:4,cumsum(100*PCTVAR(1,:)),'-bo');
xlabel('Number of PLS components');
ylabel('Percent Variance Explained in y');


yfit = [ones(size(TestSetVars,1),1) TestSetVars]*beta;

figure(9)
plot(1:5,stats.W,'-');
xlabel('Variable');
ylabel('PLS Weight');
legend({'1st Component' '2nd Component' '3rd Component'},'location','NW')


mse_plsr = immse(yfit(5000:9000),TestSetResponse(5000:9000))

figure(10)
plot(5001:9000,TestSetResponse(5001:9000),'r')
hold on
plot(5001:9000,yfit(5001:9000),'o','MarkerSize',2,'MarkerEdgeColor','blue')
title('PLSR prediction and True values')
subtitle('Mean Square Error of prediction: 0.1135')
legend({'Measured','PLSR pred'})


%% MSE BAR PLOT


X = ...
    categorical({'Ensemble','GLM','NARX','NLIO','PLSR','Stepwise','SVM','Tree'});
Y = [0.0798 0.1129 0.0797 0.0709 0.1135 0.1069 0.1175  0.1072];

bar(X,Y)
```

Appendix E

Escape-31 Paper

As a part of this work, a paper was submitted to the 31st European Symposium on Computer Aided Process Engineering. The paper co-authored by my supervisors is included below and provides a shorter version of the research related to the simulated data and the modelling of the simulated data with traditional statistical learning methods.

Metin Türkay Metin Trkay, Erdal Aydin (Eds.)
Proceedings of the 31th European Symposium on Computer Aided Process Engineering
June 6th to 9th, 2021, Istanbul, Turkey.

Data-driven modelling of choke valve erosion using data simulated from a first principles model

Jan Henrik Jahren[a], Jose Matias[a] and Johannes Jäschke[a*]

[a]*Department of Chemical Engineering, Norwegian University of Science and technology, Sem Sælands vei 4, 7034 Trondheim, Norway*
johannes.jaschke@ntnu.no

Abstract

On the Norwegian continental shelf there are, according to Norwegian Petroleum (DNV-GL (2015), approximately 87 fields operating, with operational costs of 60 billion NOK per year. Maintenance is a large portion of this, costing tens of billions of NOK. In addition to the economic burden, degradation can lead to critical equipment failure, as happened in 2011 in the Deepwater Horizons disaster where 11 people lost their lives and 4.9 billion barrels of oil were released into the environment.Pallardy (2010) Hence, failure to detect faults can have immense financial, environmental and human consequences. One of the principal mechanisms of degradation in subsea process equipment is erosion by sand, which is a very complex process and, thus, difficult to be modelled using physical domain knowledge. Because of this difficulty, we propose in this paper the use of data-driven approaches for modelling erosion in critical equipment of a subsea oil production rig. In such systems, a multitude of available process measurements such as flowrates, gas lift injection rates, pressures etc. can be combined to a soft-sensor for component degradation. This approach could save significant amounts of resources by allowing fewer cost intensive inspections and monitoring schemes. This soft-sensor approach was tested with simulated data created using the model proposed by Krishnamoorthy et al.Krishnamoorthy et al. (2016) and adapted by Verheyleweghen and Jäsckhke Verheyleweghen and Jäschke (2018). This model considers sand degradation of a choke valve in a gas lifted well oil production system with three wells. Regression models were trained on the simulated data for obtaining the component erosion soft-sensor and their accuracy validated with unseen data. The tested methods were MLR (normal, piecewise and interactions), trees (bagged, boosted and normal regression trees) and support vector regressions with various kernels. The approaches were tested in two case studies, the first with constant sand outflow from the reservoir, and the second with a more realistic profile. In both, we compare the regression modelling techniques in terms of their basic requirements as well as accuracy and convergence characteristics. Given the good performance of these approaches in the case studies, testing using real world data is merited to investigate applicability to industrial processes.

Keywords: equipment degradation, data-driven modelling, machine learning

1. Introduction

Failure to detect faults in large scale, expensive or critical equipment can have immense consequences. Both financial and in the most extreme cases, loss of life. A famous ex-

ample of such a failure is the Deepwater Horizon oil spill of 2010. This disaster released an estimated 4.9 million barrels (Pallardy (2010)) of oil and killed 11 people (Kaufman (2010)). While there where multiple failures to take appropriate actions leading up to the disaster, with financial pressure mounting and corner's being cut, the ultimate cause of the disaster was the failure of the blowout valve. The blowout valve served as the last line of defence, application of prognostics and health management to monitor this vital component could have prevented this disaster that came to be one of the worst environmental disasters of modern history (Si et al. (2012)).

One of the principle mechanisms in equipment degradation is erosion and as such accurately modelling it is vital in monitoring of equipment health (Hansen (2016)). The most obvious approach would be to create a first principle model to accurately predict erosion but this has proved difficultDNV-GL (2015). If first principles modelling does not provide satisfactory accuracy, data-driven regression modelling could be a solution. We propose in this paper the use of data-driven approaches for modelling erosion in critical equipment of a subsea oil production rig. In such systems, a multitude of available process measurements such as flowrates, gas lift injection rates, pressures etc. can be combined to a soft-sensor for component degradation. This approach could save significant amounts of resources by allowing fewer cost intensive inspections and monitoring schemes as well as improving safety. To investigate the voracity of such a data-driven modelling approach models will be tested on simulated data using the model proposed by Krishnamoorthy et al. (2016) and adapted by Verheyleweghen and Jäschke (2018).

2. Simulations

The simulated data in this work is based on a model for a gas lifted oil production facility presented by Krishnamoorthy et al. (2016), this model is adapted by Verheyleweghen and Jäschke (2018) to facilitate the simulation of the erosion of a choke valve. Krishnamoorthy et al. (2016) gives the model for describing gas lifted wells in four parts: The mass balance of different phases, density models, pressure models and flow models. The erosion rate is given by equation , where $\frac{dE}{dt}$ is the erosion rate while K, n, $C1$, GF and C_{unit} are constants. Calculation of the parameters A and G is shown in detail in the paper by Verheyleweghen and Jäschke, $F(\alpha)$ is the ductility and U_p^n is the particle impact velocity.

$$\frac{dE}{dt} = \frac{KF(\alpha)U_p^n}{\rho_t A_t} G * C_1 * GF * \frac{dm_{sand}}{dt} * C_{unit} \tag{1}$$

An implementation of this adapted model using casADi (Andersson et al. (2019)) in Matlab version 2019b is used to simulate a large amount of data to test models on. This was done by simulating 200 time series of length 500 days, variability in the model is created by letting the control input (gas lift rate) vary randomly within a given range. Two of these data sets were created, first one where the sand production rate was held constant and then one where the sand production rate was increased exponentially according to equation 2 as this more closely resembled the sand production in an actual reservoir.

$$m_{sand,t} = m_{sand,0} * \exp(0.005 * t) \tag{2}$$

Recorded variables from the simulations		
Sand production rate	Annulus pressure	Well head pressure
Well head oil production rate	Well head gas production rate	Riser head pressure
Manifold pressure	Riser head total oil production rate	Riser head total gas production rate
Gas lift rate	Measured erosion (response)	

Table 1: Table listing recorded variables from simulations used for training and testing the data-driven models.

Ten process variables are recorded for each time step of the simulation (one timestep is one day) as well as a the measured erosion at the given time.

2.1. Pre-processing

The simulations record directly the measured erosion, but since prediction is done using process variable values at a given time it is more sensible to predict the rate of erosion at the given time. To get this the gradient of the time series of erosion measurements is used as the response variable while the remaining recorded variables are used for prediction. Since sand production rate is normally not a constantly measured process variable but rather something that is sampled at longer intervals, the models are trained using a sampling rate of once per 50 days with respect to the sand production rate. In addition to using the gradient values of the response, principal components analysis as pre-processing was used for some of the models to test performance when using PC scores versus raw predictor data.

3. Data-driven models

The proposal of this paper is that ordinary statistical learning methods can provide a good stand in for measurements of erosion acting as a soft sensor. In this paper results will be shown for three categories of common data-driven models: linear regressions, three based models and support vector regressions. A brief description of the methods as well as the specific versions applied in this work is given below. All the methods described below are implemented in Matlab MATLAB (2020) using the Machine Learning and Statistics toolbox The MathWorks (2020). All training optimization and selection of models is done using holdout validation to select models. Tuning of hyperparameters for ensemble and support vector methods are done with bayesian optimization.

3.1. Linear regressions

The most basic of all statistical models, assuming a linear relationship between variables and the response. In this work it is found that the best performing method in terms of test set MSE is a linear interactions model that includes not only the predictor variables but first order interactions between the predictors (Friedman et al. (2017)). Normal multiple linear regression and stepwise linear regression methods also performed well.

3.2. Tree based methods

Using tree based models both simple decision trees as well as ensemble approaches perform well in terms of test set MSE. All the methods are based on building a tree with splits on a given variable at a given value. The ensemble methods simply use multiple trees to make their predictions in slightly different ways. Boosting uses successive weak learners to create a single strong learner, letting the training importance on wrongly classified samples get larger compared to correctly classified samples during training to make a collection of weak learners that average to a strong learner. Bagging or bootstrap aggregation is a method used to reduce variance by training m models on m different bootstrapped data sets containing $\sim 63\%$ of the data and using a average of the predictions (Friedman et al. (2017)).

3.3. Support vector regression

Support vector regression (SVR) is a kernel method, that adapts the maximal margin classification which forms the basis of support vector machines (SVMs) to regression problems. In the regression mode, one still attempts a similar feat of creating a separating hyperplane of the form $f(x) = \beta x^T + \beta_0$ by solving this minimization problem: $\min_{\beta,\beta_0} H : H(\beta,\beta_0) = \Sigma_{i=1}^{N} V(y_i - f(x_i)) + \frac{\lambda}{2}||\beta||^2$. Where V is a function deciding how errors are penalised and λ is a regularization parameter (Friedman et al. (2017)).

4. Results and discussion

A set of six specific learning methods was tested on both the data simulated from constant sand production rate and exponential sand production rate, the model performance as measured by MSE for constant sand production rate is given in table 2 and performance for exponential sand production rate data is given in table 3. There are some considerations outside of model performance that needs to be considered when selecting a model for actual applications. There are significant differences in model training time, with the outlier being SVM based models requiring significantly more training than tree ensembles and MLR based models. The other consideration that needs to be made is that to be acceptable in real world scenarios "black box" models of safety critical components is undesirable and a premium is placed on model transparency and interpretability. None of the models considered here are completely black box, in general models with transparent coefficients showing how a prediction is made is easier to interpret such as the MLR models. As such when performance is close the preferred methods will be MLR models due to fast training times and superior interpretability.

4.1. Constant sand production rate data

In the simplest test case where a constant sand production rate was used the models performed (as expected) extremely well as the phenomena behaves linearly. Linear phenomena are very well suited to being modelled by simple statistical learning methods. The equivalent results from interactions MLR and stepwise MLR are certainly due to both algorithms producing very similar models as the stepwise regression algorithm adds linear interaction terms as well. While the performance of the MLR based models is degraded by application of PCA pre-processing there is a significant gain in model simplicity with

far fewer features being used for prediction which provides advantages in terms of data needed for training and a lower model variance.

Method	MSE w/o PCA	MSE w/ PCA
MLR w/ interactions	0.0096	0.0166
Stepwise MLR	0.0096	0.0166
Bagged trees	0.0091	0.0162
Boosted trees	0.0125	0.0320
Linear kernel SVR	0.0123	0.0181
Cubic kernel SVR	0.0120	0.0129

Table 2: Table showing the test set performance of the models on simulated data with a constant sand production rate.

4.2. Exponential sand production rate

For the data simulated with exponential sand production rate, a significant drop in model performance was observed, as expected with a more complex function being emulated. As with the data simulated from constant production rate, there is a drop in performance when PCA is applied, but this effect is relatively weaker for the exponential data. In this case cubic kernel support vector regression proved to have the strongest performance, but in general bagged trees, boosted trees, interactions and stepwise MLR all provided fairly accurate predictions of the erosion rate. The sampling rate of the sand production rate measurement as expected has a very significant impact on the model accuracy with exponential distribution, increasing sampling to once every 30 days instead of 50 for example reduces the MSE of a optimised bagged ensemble to 0.0066 from 0.0184.

Method	MSE w/o PCA	MSE w/ PCA
MLR w/ interactions	0.0182	0.0191
Stepwise MLR	0.0182	0.0191
Bagged trees	0.0184	0.0192
Boosted trees	0.0184	0.0239
Linear kernel SVR	0.0465	0.0473
Cubic kernel SVR	0.0164	0.0169

Table 3: Table showing the test set performance of the models on simulated data using an exponential sand production rate.

5. Conclusion

It is observed that for constant sand production rate very accurate predictions of the erosion rates are made, a significant degradation of model accuracy is seen for all the constant sand production rate models except for cubic kernel SVR when PCA pre-processing is applied to the data. On the data simulated from an exponential sand production rate the performance is overall worse as expected since the function that the models are attempting to reproduce is more complex. The methods are still relatively accurate with under 0.02 MSE on normalised unseen test data, what is obvious is that when the slope gets very

high the models suffer quite significantly from the sampling rate of once every 50 days on the sand production rate. With most of the models, except for linear kernel SVR which overall performed very poorly performing well also with PCA pre-processing. With current model performance there is a strong case to be made for selecting linear regression based methods as they provide superior model transparency and interpretability. Having observed that simulated data can be predicted well using statistical models, further investigation on real world data is merited to ascertain applicability to real industrial facilities.

6. Future work

The obvious next step for this work is testing the methods in a laboratory setup, where we can become more confident that the approach merits testing in a full scale industrial project. This is work we are currently undertaking at NTNU, using a lab rig which emulates the studied oil production system for remaining useful life estimation, initial results show some promise. With successful reproduction and validation of results, industrial trials are the next step.

7. Acknowledgments

This work was carried out as a part of SUBPRO, a Research-based Innovation Centre within Subsea Production and Processing. The authors gratefully acknowledge the financial support from SUBPRO, which is financed by the Research Council of Norway, major industry partners, and NTNU.

References

Andersson, J. A. E., Gillis, J., Horn, G., Rawlings, J. B. and Diehl, M. (2019), 'CasADi – A software framework for nonlinear optimization and optimal control', *Mathematical Programming Computation* **11**(1), 1–36.

DNV-GL (2015), 'Recommended practice rp-o501: Managing sand production and erosion.'. https://rules.dnvgl.com/docs/pdf/dnvgl/RP/2015-08/DNVGL-RP-O501.pdf.

Friedman, J., Hastie, T. and Tibshirani, R. (2017), *The Elements of statistical learning, Data Mining, Inference, and Prediction*, Springer.

Hansen, S. K. (2016), Modelling failure mechanisms in subsea equipment, Master's thesis, NTNU.

Kaufman, L. (2010), 'Search ends for missing oil rig workers', *New York Times* .
URL: *https://www.nytimes.com/2010/04/24/us/24spill.html*

Krishnamoorthy, D., Foss, B. and Skogestad, S. (2016), 'Real-time optimization under uncertainty applied to a gas lifted well network', *Processes, Real-Time Optimization (Special Issue)* .

MATLAB (2020), *version (R2020b)*, The MathWorks Inc., Natick, Massachusetts.

Pallardy, R. (2010), 'Deepwater horizon oil spill', *Encyclopaedia Britannica* .
URL: *https://www.britannica.com/event/Deepwater-Horizon-oil-spill*

Si, X., Wang, W., Hu, C., Zhou, D. and Pecht, M. G. (2012), 'Remaining useful life estimation based on a nonlinear diffusion degradation process', *IEEE Transactions on Reliability* **61**(1), 50–67.

The MathWorks, I. (2020), *Statistics and Machine Learning Toolbox*, Natick, Massachusetts, United State.
URL: *https://www.mathworks.com/help/stats/*

Verheyleweghen, A. and Jäschke, J. (2018), 'Oil production optimization of several wells subject to choke degradation', *IFAC-PapersOnLine* **51**.

Jan Henrik Jahren

Data-driven modelling of subsea equipment degradation using simulated and experimental case studies

NTNU
Norwegian University of
Science and Technology