Predicting the concentration of residual methanol in industrial formalin using machine learning

Förutspå koncentrationen av resterande metanol i industriell formalin med hjälp av maskininlärning

William Heidkamp

Faculty of Health, Science and Technology

Engineering Physics, Bachelor Degree Project

15 ECTS credits

Supervisor: Thijs Jan Holleboom

Examiner: Lars Johansson

October 2016

This page intentionally left blank

Abstract

In this thesis, a machine learning approach was used to develop a predictive model for residual methanol concentration in industrial formalin produced at the Akzo Nobel factory in Kristinehamn, Sweden. The MATLAB™ computational environment supplemented with the Statistics and Machine Learning™ toolbox from the MathWorks were used to test various machine learning algorithms on the formalin production data from Akzo Nobel. As a result, the Gaussian Process Regression algorithm was found to provide the best results and was used to create the predictive model. The model was compiled to a stand-alone application with a graphical user interface using the MATLAB Compiler™.

Acknowledgements

I would like to sincerely thank my supervisor; Thijs Jan Holleboom at Karlstad Univeristy for his help during this work. I also want to thank the companies I collaborated with; Akzo Nobel AB and The MathWorks AB.

I would like to give very special thanks to my supervisor at Akzo Nobel AB, Stefan Kvarth and his colleagues; for allowing me to use their data and to work with them at their office, not to mention the discussions and all the other help they have given me.

Many thanks to Rafal Rogowski for his great guidance and for sharing all his knowledge of machine learning during this work. I would also want to thank his colleagues at The MathWorks AB for providing me with a MATLAB license, the two day long course in machine learning and showing much interest in my project.

Contents

Abstract **I**

Acknowledgements **II**

List of Figures **1**

1 Introduction **3**

2 Description of the machine learning algorithms used **6**
2.1 Principal component analysis 6
2.2 Regression models 6
2.2.1 Linear regression 6
2.2.2 Support vector regression machines 7
2.2.3 Gaussian process regression 7
2.2.4 Artificial neural network 8

3 Method and Results **10**
3.1 Preprocessing the data 10
3.2 Unsupervised learning 11
3.2.1 Correlation 11
3.2.2 Principal component analysis 12
3.3 Supervised learning 19
3.3.1 Preparing the data for regression based learning 19
3.3.2 Regression and predictions 19
3.3.3 Stand-alone application 27

4 Discussion **29**
4.1 Unsupervised learning 29
4.2 Supervised learning 30
4.2.1 Stand-alone application 30

5 Conclusion **31**

Bibliography **32**

Appendix **34**
Gaussian process regression 34
Main MATLAB code 38
MATLAB functions 43
Stand-alone application: MATLAB code 46
MATLAB function used in the application 50

List of Figures

1.1 An overview of the reactors' parameters. 4
1.2 An overview of the absorbtion column's parameters. 4

2.1 Structure of a typical artificial neural network. 8

3.1 Computed correlation coefficients between the parameters for each reactor, highest possible correlation is 1. 11
3.2 Visualization of both reactors' eight dimensional data sets with the first two principal components. 13
3.3 Visualization of both reactors' eight dimensional data sets with the first three principal components. 14
3.4 Heatmap of the first output from the *pca* function to visualize the relations between the principal components and the original parameters. 14
3.5 Scatter plot of the first three principal components from the 'not transformed' data. 16
3.6 Heatmap showing the relations between the eight first prinicpal components and original parameters ('not transformed'). 16
3.7 Scatter plot of the first three principal components from the 'transformed' data. 17
3.8 Heatmap showing the relation between the first eight principal components and the original parameters for the 'transformed' data. . . 18
3.9 The predicted concentration of methanol using linear models (the measured methanol concentration in blue). 20
3.10 Error calculations for the linear models. The red is the error for the 'transformed' data and the the black for the 'not transformed'. . 20
3.11 The predicted concentration of methanol using SVRMs models (the measured methanol concentration in blue). 21
3.12 Error calculations for the SVRMs models. The red is the error for the 'transformed' data and the the black for the 'not transformed'. . 22
3.13 The predicted concentration of methanol using GPR models (the measured methanol concentration in blue). 22
3.14 Error calculations for the GPR models. The error is depicted in red for the 'transformed' data and black for the 'not transformed'. . . . 23
3.15 The predicted methanol concentration by the GPR models for May and the first six days in June. 24
3.16 The predicted methanol concentration by the neural network models for May and the first six days in June. 24

3.17 The measured methanol concentration before and after the moving average filter. 25
3.18 The measured methanol concentration for May and the first few days in June, for the GPR model fitted to the filtered concentration of methanol. 26
3.19 The measured methanol concentration for May and the first few days in June, for the neural network model fitted to the filtered concentration of methanol. 26
3.20 The measured methanol concentration for May and the first few days in June, for the GPR model fitted to the filtered concentration of methanol, without the parameters gas temperature, cooler and temperature before absorber. 27
3.21 The measured methanol concentration for May and the first few days in June, for the neural network model fitted to the filtered concentration of methanol, without the parameters gas temperature, cooler and temperature before absorber 27
3.22 The graphical user interface for the stand-alone application on OS X operating system. 28

A.1 Predicted concentration of methanol for the GPR models, trained using the filtered methanol concentration (blue). 34
A.2 Error calculations for the GPR models, trained with the filtered methanol concentration. The red is the error for the 'transformed' data and the the black for the 'not transformed'. 35
A.3 Predicted concentration of methanol for the GPR models, trained using the filtered methanol concentration (blue) with reduced parameters. 36
A.4 Error calculations for the GPR models, trained with the filtered methanol concentration and reduced parameters. The red is the error for the 'transformed' data and the the black for the 'not transformed'. 37

1 Introduction

In recent decades, there has been a growing interest in utilizing machine learning methods in various industrial, scientific and business applications e.g. Google, Amazon or WEKA. Machine learning is applied within a broad area of engineering and science including a set of well established algorithms and best practises as well as a rapidly developing discipline. With the progress in computation technologies, it has become possible to use machine learning in solving some of the biggest problems of today e.g. analyzing the ever-growing volumes of data in a structured way to make rational and knowledgeable business decissions.

Machine learning could roughly be divided into two subfields; unsupervised and supervised machine learning, although more subcategories could also be distinguished. While the unsupervised machine learning is used for discovering natural patterns in complex data sets, the supervised machine learning aims at building predictive models that can be utilized to make predictions for new sets of data. [1]

In this thesis, the formalin process data obtained from Akzo Nobel AB* will be analyzed using both the unsupervised and supervised machine learning algorithms. The main objective of this analysis is to develop a model able to predict the concentration of residual methanol in the final product (formalin). MATLAB and the Statistics and Machine Learning toolbox from the MathWorks** were used to perform the analysis and develop the predictive model.

The raw material used for synthesizing formalin in the process under consideration is methanol. The reactor system consists of two parallel reactors connected to an absorption column. Figure 1.1 shows a basic overview of the system for one reactor only. Each reactor has 1000 small tubes with internal ceramic rings of two types i.e. inert and catalyst (iron-molybdenum) rings. The rings are ordered in different layers, where the number of layers and the length of the tubes are different for each reactor.

*The formalin production factory in Kristinehamn, Sweden.
**The MathWorks Swedish Office in Kista.

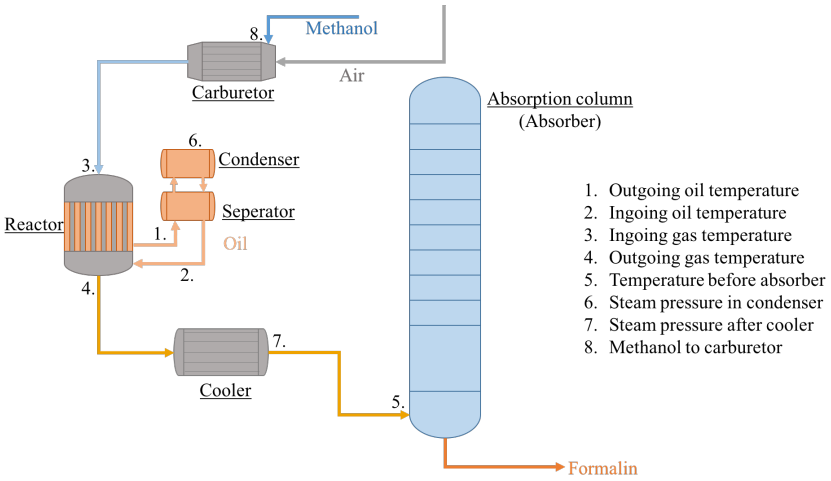


Figure 1.1: An overview of the reactors' parameters.

The underlying chemical process taking place in both reactors can be described by the following chemical reaction equation:

$$Methanol + Oxygen \rightarrow Formaldehyde + Water + Heat$$

$$\mathrm{CH_3OH} + \frac{1}{2}\,\mathrm{O_2} \longrightarrow \mathrm{CH_2O} + \mathrm{H_2O} + \text{heat}$$

Since the reaction is higly exotermic, the system needs to be cooled down with a special oil coolant. The main product of the reaction (formaldehyde) is transported into the absorption column where it is dissolved in water forming the final product (formalin), see Figure 1.2. As a consequence of the reactions, the catalyst rings of each reactor deteriorate and need to be exchanged to maintain the optimal reaction efficiency. Until this happens, the ingoing oil temperature needs to be increased proportionally. After the catalysts rings have been exchanged, the oil temperature is lowered back to its initial value.

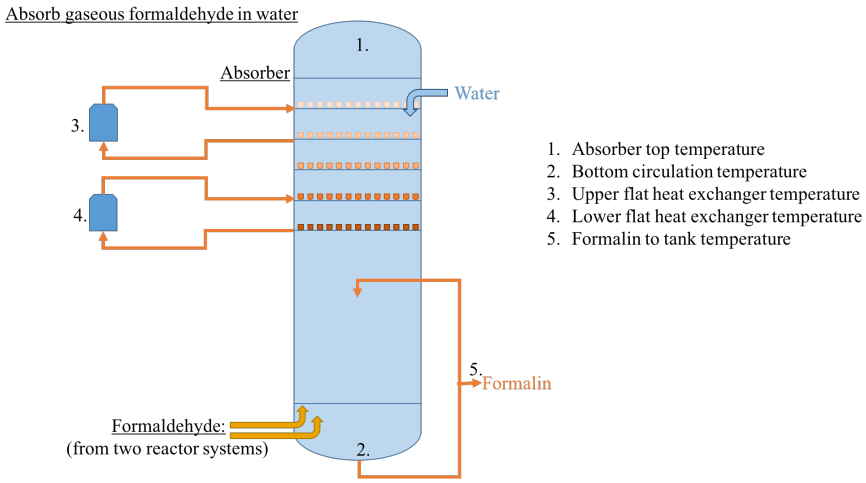


Figure 1.2: An overview of the absorbtion column's parameters.

Formalin produced using this method contains small amounts of methanol (up to 1.7% at the extreme), which significantly lowers its quality. Therefore, reducing the concentration of the remnant methanol or idealy eliminating it completely is of great importance to ensure the highest quality of the product. A question arises then, how could one control the system's parameters (Figures 1.1 and 1.2) to lower the concentration of residual methanol in the final product as much as possible? As the formalin production process is complex and many factors of different nature are involved, it is difficult to develop a traditional mathematical model that could be used to accurately predict the concentration of residual methanol. An interesting alternative to traditional approaches seems to be machine learning for the following reasons; it can be used to reveal patterns and relationships hidden in data to understand the processes involved (unsupervised machine learning), and apply that knowledge to develop models capable of making predictions on new sets of data (supervised machine learning).

The goal of this thesis is to use first unsupervised machine learning to investigate pattern and relationships in the data, and then create a model that could accurately predict the concentration of methanol in formalin based on new or anticipated process data using supervised machine learning.

2 Description of the machine learning algorithms used

This chapter provides an overview of the machine learning algorithms used throughout the work, without including their formal mathematical discussion. The reader should consult each section's references for more comprehensive discussion of the methods. In particular; linear regression, support vector regression machines, gaussian process regression and artificial neural network, have been selected as the most promising candidates for creating the predictive model.

2.1 Principal component analysis

Principal component analysis (PCA) is a method often used to search for patterns in multivariate data sets (unsupervised learning). It is a statistical method that can be used to transform a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. It uses an orthogonal transformation to linearly transform a data set into a set of principal components, while preserving as much as possible of the variation in the original data. The method does not ignore covariances and correlations but focuses on variances. The principal components are independent of each other and are arranged such that the first few principal components explain most of the variation of all the original variables. The principal componets can be viewed as axes in a new space, where they are orthogonal to each other. If the first few principal components explain the majority of the data, it is possible to visualize a highly dimensional data set in fewer dimensions without losing extensive amount of information. [2]

2.2 Regression models

2.2.1 Linear regression

Linear regression is a method extensively used for numerous different problems to model relationships between variables, considered to be both dependent and independent, by fitting a linear equation to observed data. A possible linear

regression model can be written in the following general form:

$$y_i = b_0 + \sum_{k=1}^{K} b_k f_k(x_{i1}, x_{i2}, ..., x_{ip}) + \epsilon_i \qquad i = 1, \cdots, n, \tag{2.1}$$

where y_i is the dependent variable, b_0 is a constant term while b_k is the kth coefficient, f_k is a function of the independent variables, x_{ij}, that may have any form, x_{ij} is the ith measurement of the jth variable ($j = 1, 2, ..., p$) and ϵ_i is the ith noise term (a random error). The noise terms ϵ_i have all the same normal distributions with zero mean and constant variance [3], [4]. The MATLAB function *fitlm* computes, if no arguments are given, a linear regression model that contains an intercept and linear terms for each independent variable. The fitted linear function has the same structure as the equation (2.1), where y_i is the predicted response and b_k are the fitted coefficients. The coefficients are determined, so that the error of the least squares of the predicted response vector, $\tilde{y}$, and the true response vector, y, is as small as possible [5].

2.2.2 Support vector regression machines

Support vector machines (SVMs) can be used for classification, which in this method is the processes of separating different groups in the data by finding the hyperplane that has the biggest margin between the groups. A hyperplane is a subspace of the original n-dimensional space whose dimension is $n - 1$. SVMs are binary by nature, meaning they are only directly applicable to data with two classes. Cortes and Vapnik have introduced the *soft margin hyperplane* concept to make it possible to separate data that is not strictly linearly separable [8].

Later Drucker et al. introduced a regression technique based on Vapnik's concept of support vectors referred to as support vector regression machines. The MATLAB's support vector regression machines algorithm implements ϵ-insensitive loss, which is a loss function that treats errors within ϵ distance of an observed value as equal to zero. Essentially the loss is measured based on the distance between observed values y and the ϵ boundary. The algorithm aims to find a function $f(x)$ that does not lie further from the responses y_i than ϵ for each training point x. If no function $f(x)$ satisfies the constraints for all points, slack variables* are introduced for each point, similar to the *soft margin* concept mentioned above. [9], [10]

2.2.3 Gaussian process regression

Gaussian process regression (GPR) is a very versatile generic supervised learning method that also allows for specifying custom stationary models, including using probabilistic techniques based on covariance functions**. In short, the algorithm

*Slack variables are defined to transform an inequality expression into an equality expression.

**The covariance functions, also referred to as kernels, will determine how the response at one point, x_i is affected by responses at other points, x_j.

uses a probability distribution of basis functions to determine the best distribution of functions that fits the observed data. Assuming that the observed data can be represented as a sample from a multivariate Gaussian distribution, this technique can be combined with a Gaussian process (GP). In such a process every point in the data is associated with random variables that are normally distributed. It is also assumed that the mean of the GP is zero everywhere i.e observations are related to each other through the covariance function $k(x, x'|\theta)$. The covariance function may be any valid kernel function and $|\theta$, indicates that the functions are parameterized by a set of hyperparameters θ. [6]

$$P(y_i|f(x_i), x_i) \sim N(y_i|h(x_i)^T\beta + f(x_i), \sigma^2) \tag{2.2}$$

(2.2) represent a GPR model (in non-vector form) for one response y_i, where ^{T} is the transpose, $f(x_i)$ are the latent variables introduced by the GP for each observation x_i, $h(x_i)$ is the basis function, β are the coefficients for the corresponding basis function and σ^2 is the error variance. The probability equation (2.2), relates the probability of y_i to the given data. The MATLAB function *fitrgp* can estimate the coefficients β, the noise variance σ^2 and the hyperparameters θ of a kernel function from the data while training the model, the default kernel function is the squared exponential kernel. [6], [7]

2.2.4 Artificial neural network

Artificial neural networks models (inspired by biological nervous systems) are used to approximate, generally unkown, functions that are commonly dependent on a considerable number of inputs. It is constructed by interconnected neurons that map inputs to responses. Figure 2.1 shows how a typical neural network is structured.

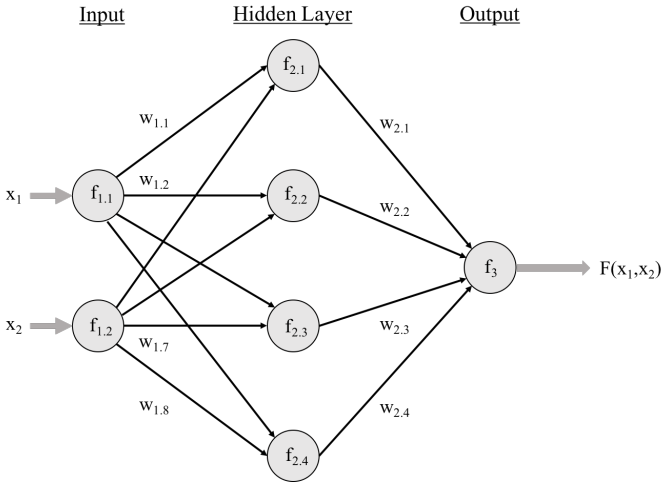


Figure 2.1: Structure of a typical artificial neural network.

The inputs, x_i, have an associated weight, $w_{i,j}$ and then with the use of *transfer functions*, shown as black arrows in Figure 2.1, the information in x_i is multiplied by the corresponding weight. It is a linear combination of the values from the previous neurons, whose coefficients are the weights $w_{i,j}$. This information is integrated into the neuron and the primitive functions $f_{i,j}$ are calculated, e.g. $f_{2,1}(w_{1,1}x_1 + w_{1,5}x_2)$. The difference between different artificial neural network models lies primarily in the assumptions about the primitive functions $f(x)$, the interconnection pattern and timing of the transmission of information. The function $F(x_1, x_2)$ in Figure 2.1 is the *network function*. The primitive functions are combined to create the *network function*. By adjusting the different weights, a new *network function* will be produced. In short, training an artificial neural network model requires, iteratively adjusting every weight until the desired outputs are obtained for the inputs. [11]
The neural network implementation in MATLAB is a feedforward network. Each time the feedforward network is initialized, the network parameters will in general be different and the network may produce different results [12].

3 Method and Results

3.1 Preprocessing the data

The formalin production data set is divided into three sections; reactor one, reactor two and the absorption column (Figures 1.1 and 1.2), where the respective parameters are most likely to have the greatest effect of the methanol concentration in formalin. All values of the control parameters from the three sections were recorded at the same time by a computer system. Each measurement set (at a specified time) in the data reflects a change in the value of one of the control parameters. The process data must be linked to the concentration of methanol in formalin which is normally analyzed once a day. This restricts the amount of process measurements to 396 days, i.e the period from April 1, 2015 to April 30, 2016, when the measurements were done.

Machine learning algorithms work best for large data sets. Since the sample of 396 measurements is clearly not a big data set, a risk of overfitting the data arises. Overfitting means that a regression model is customized too much in the learning process to describe the relationships between the training data and the responses. Thus the model becomes to complex which leads to inaccurate predictions when new data is introduced. For this reason only the measurements recorded when both reactors were active, were taken into account in the analysis. The *methanol to carburetor** parameter is used to remove insignificant measurments from the data. That is, all measurements corresponding to the values of *methanol to carburetor* lower than 15 kg/min were removed as any value lower than 15 kg/min means inactive reactor. The measurements for which the methanol concentration was not recorded were also removed. In effect the number of significant measurements shrank to 329.

Both reactors' are characterized by the same eight parameters (Figure 1.1) whilst the absorption column by five (Figure 1.2). By merging the three sets of parameters, a new set of data is constructed containing every parameter for both reactors and the absorption column, in total 21 parameters. In addition to that, another set of data was constructed, in an attempt to improve the accuracy of the predictive models, by treating the two reactors as one, i.e. by taking the mean of their corresponding eight parameters. By adding the five parameters in absorption column's data, the resulting parameters where 13 in total. From now on this 13 dimensional data will be referred as 'transformed' data, while the previous 21

*This parameter describes the amount of methanol transported into the carburetor per unit time.

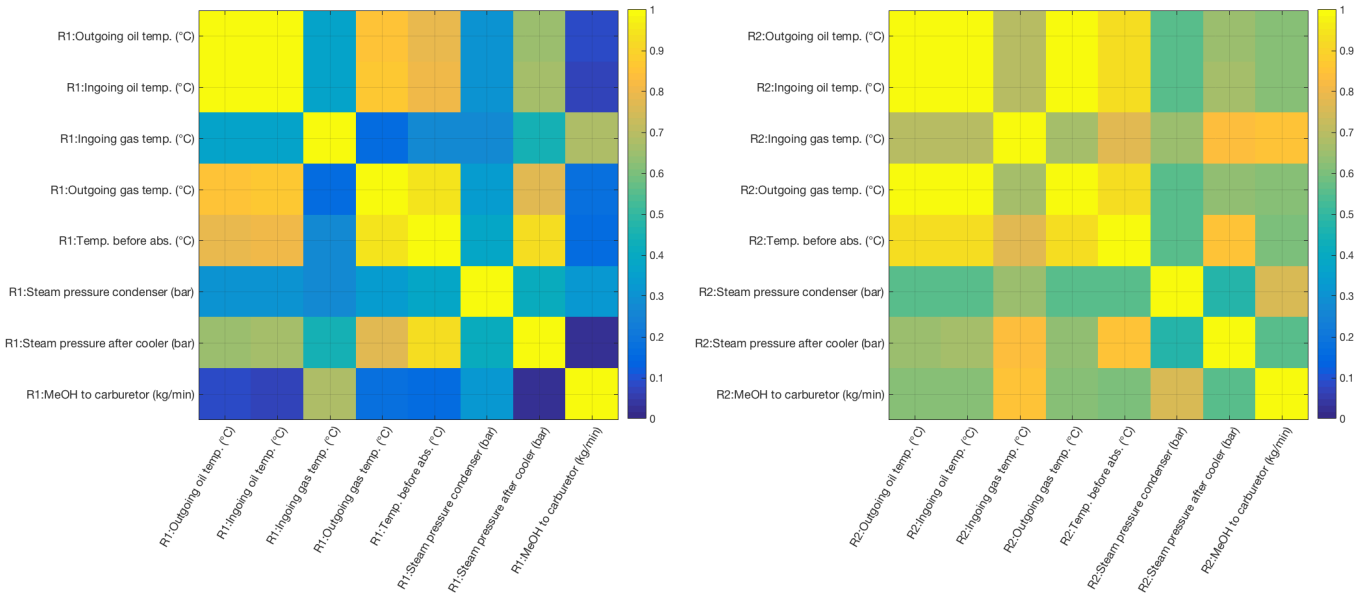
dimensional data is referred as 'not transformed'.

3.2 Unsupervised learning

Unsupervised learning can be used to approach different problems with limited or no knowledge. It is possible to derive structures, in the form of clusters, based on relationships among the variables in the data where the effect of the different variables is not necessarily known in the beginning.

3.2.1 Correlation

Correlation can be used to test for the existence of linear relationships between parameters in a data set. The correlation for each reactor was computed to investigate relationships between their parameters. The MATLAB's function *corr(X)* computes, by default, the widely used Pearson's linear correlation coefficients and returns a matrix of pairwise linear correlation coefficients between each pair of columns in the matrix X[13]. The correlation matrices were plotted as heatmaps* to visualize the correlations of the reactors' parameters and make it easier to distinguish the differences between the two reactors, Figures 3.1a and 3.1b.



(a) Reactor one (b) Reactor two

Figure 3.1: Computed correlation coefficients between the parameters for each reactor, highest possible correlation is 1.

*$abs(imagesc(X))$, where X is the plotted matrix.

The values in the figures indicate the degree of linear relationships between the parameters and zero indicates lack of such a relationship.

It is seen that the outgoing and ingoing oil temperature together with the other related parameters, have high correlations, forming squares of similar colour. Note that the colorbars in both figures are the same scale and the difference between them is caused by different values of correlation. No correlation value in Figure 3.1b is lower than 0.4881 compared to Figure 3.1a, where the darkest blue is as low as 0.0276. While similarites between the two figures exists, the results indicate that there is a stronger relationship between most of the parameters of reactor two than reactor one.

3.2.2 Principal component analysis

It might be difficult to identify hidden patterns in high dimensional data sets without thorough analysis. PCA is used to identify patterns by finding principal components using the method described in 2.2.1. The principal components are calculated with the MATLAB's *pca(N)* function, where N is the matrix representing the normalized data. The industrial process data used in this section is normalized using the function *zscore(X)*, which normalizes data by centering each column (parameter) of the original data, X, so the resulting data set has a mean of 0 and a standard deviation of 1 [14].

Furthermore, the *pca* function can return a number of output arguments;
$[pcs, scrs, \ , \ , pexp] = pca(N)$, where *pcs* is a n-by-n matrix of principal components. It defines the linear transformation between the untransformed (original) normalized data N and the transformed data returned by the second output $scrs$, so $scrs = N * pcs$ or $N = scrs * pcs^T$. The output *pexp* is a vector of the percentage of variance explained by each component. [15]

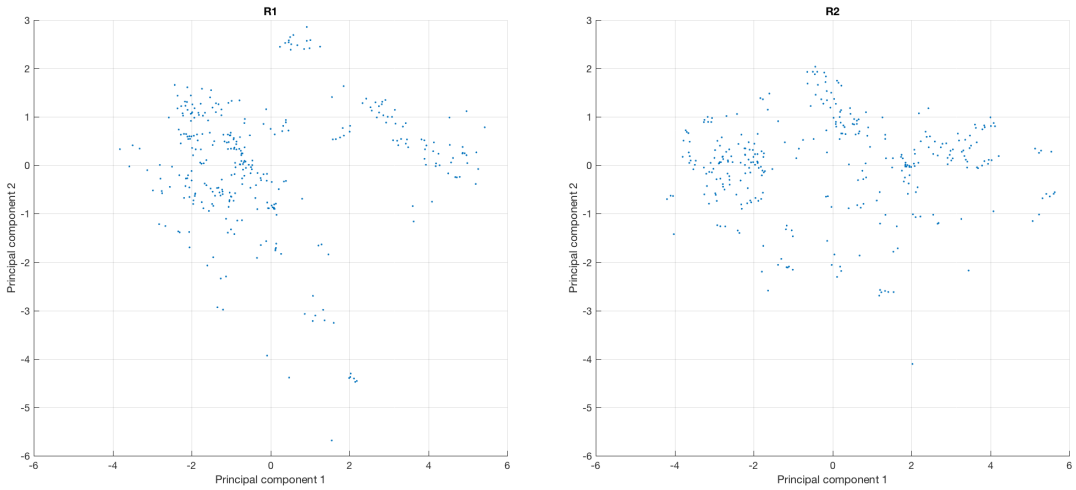
Reactor data

The principal components for each reactor are calculated with the *pca* function and the variances explained by each principal component are shown in Table 3.1.

Table 3.1: Variance explained by the principal components for reactor one and two.

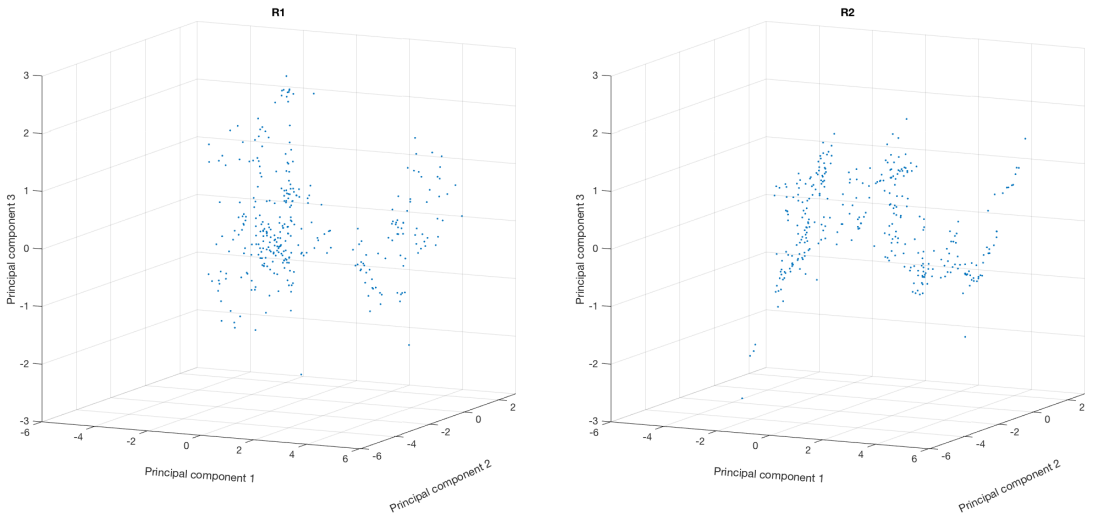
Principal component	Variance (%): Reactor one	Variance (%): Reactor two
1	58.70	76.17
2	22.22	11.88
3	9.51	7.38
4	7.12	3.45
5	1.69	0.73
6	0.59	0.28
7	0.12	0.1
8	0.05	0.01

The cumulative sums of the first three principal components of reactor one and two are 90.43% and 95.43% respectively. The eight dimensional data sets for each reactor are visualized in two and three dimensions using the first two and three columns in the matrix *scrs*, the second output from the *pca* function, Figures 3.2a to 3.3b. To relate the principal components to the original parameters the *pcs* matricies were visualized as heatmaps, Figures 3.4a and 3.4b.



(a) Reactor one (b) Reactor two

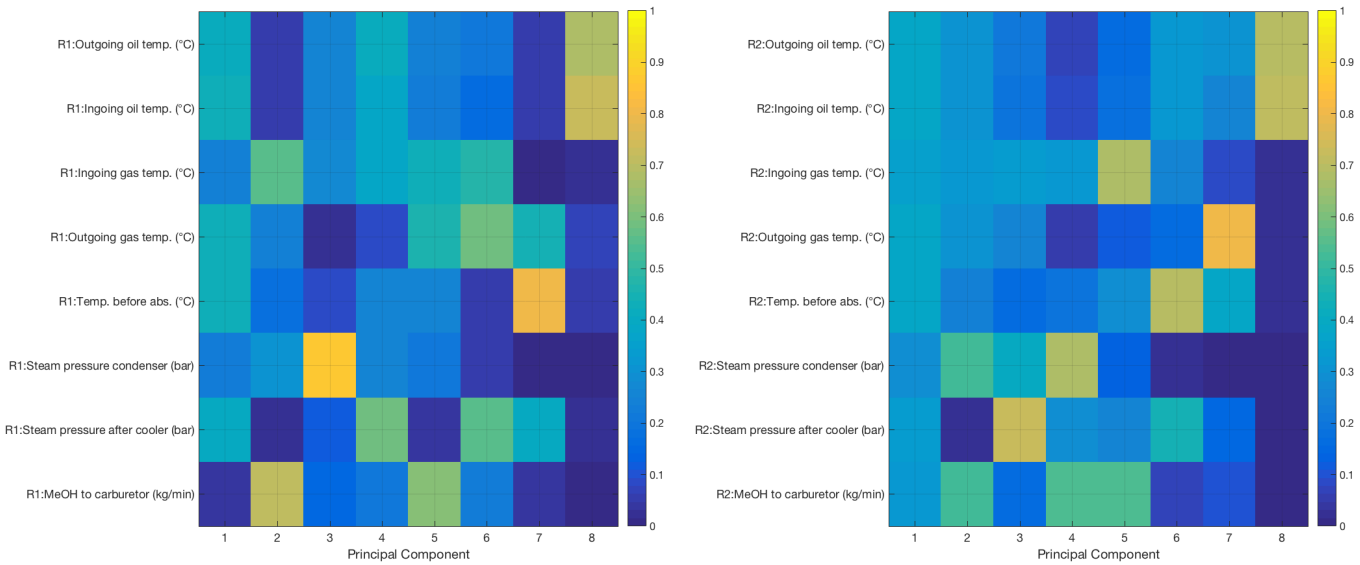
Figure 3.2: Visualization of both reactors' eight dimensional data sets with the first two principal components.



(a) Reactor one (b) Reactor two

Figure 3.3: Visualization of both reactors' eight dimensional data sets with the first three principal components.

In Figures 3.2a and 3.2b similar patterns of two clusters can be identified. More prenounced patterns in the data can be revealed using the first three principal components, Figures 3.3a and 3.3b, where the difference in the cluster structures seen will be adressed in the discussion section.



(a) Reactor one (b) Reactor two

Figure 3.4: Heatmap of the first output from the *pca* function to visualize the relations between the principal components and the original parameters.

The three first principal components, $1-3$ in Figure 3.4, explain over 90% of the data and are therefore the most interesting components to focus on. The high values indicate strong relation between the corresponding parameters and the principal component. Thus, they help to identify how the original parameters affect the structure present in the data revealed by the principal components.

The correlation heatmap (Figure 3.4b) exhibit nearly similar values of the correlation between the temperature parameters and the first three principal components. The possible causes of the difference between the two reactors will be analyzed in the discussion section.

'Not transformed' and 'transformed' data

Table 3.2 presents the variance explained in the eight first principal components for both sets of data. Figures 3.5 to 3.8 presents the visualization of the data in three dimensions and the relations of the principal components to the original parameters.

Table 3.2: Variance explained by the first eight principal components for the 'not transformed' and 'transformed' data.

Principal component component	Variance (%): 'Not transformed'	Variance (%): 'Transformed'
1	35.99	37.48
2	23.25	21.23
3	14.15	15.87
4	11.25	10.61
5	4.27	4.71
6	3.25	4.09
7	2.30	3.10
8	2.17	1.83

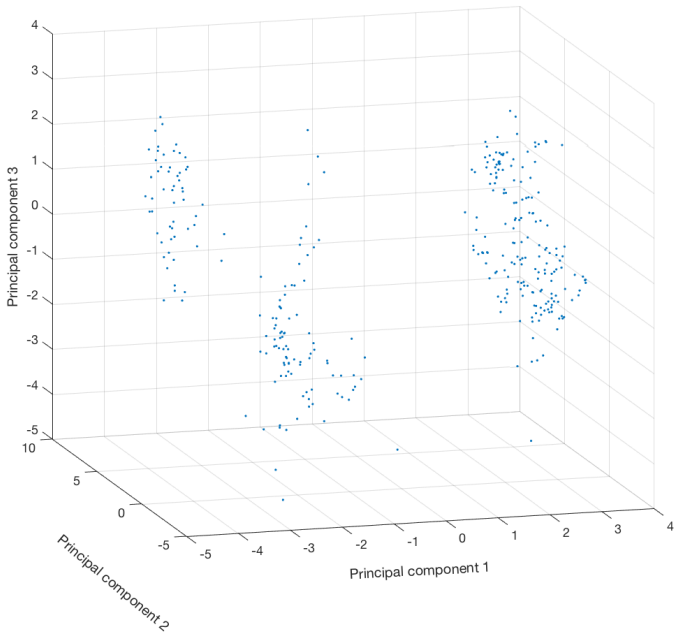


Figure 3.5: Scatter plot of the first three principal components from the 'not transformed' data.

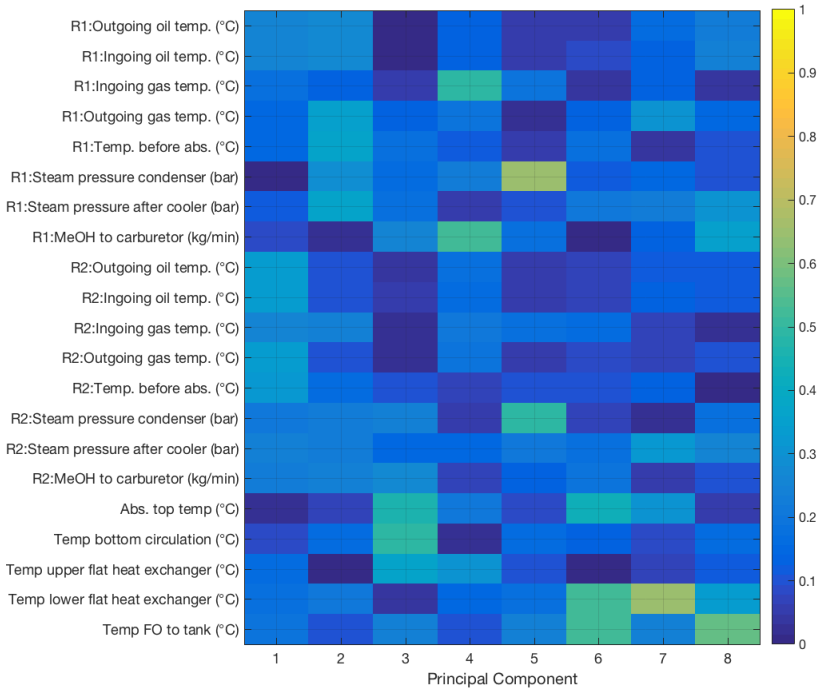


Figure 3.6: Heatmap showing the relations between the eight first prinicpal components and original parameters ('not transformed').

Compared to previous results, applying PCA analysis on to the 'not transformed' data set reveals that only 73.4% is explained by the first three principal components. It is not clear at the moment what kind of an underlying pattern the clusters seen in Figure 3.5 reveal. Some suggestions will be provided in the discussion section.

Figure 3.6 presents the dependence of the first eight principal components on the system's parameters (reactor one and two plus the absorption column). As can be seen from the figure, the first principal component shows a stronger correlation with the second reactor's parameters, while the second principal component seems more correlated with the first reactor's parameters. On the other hand the third principal component is mostly affected by the absorption column's parameters. However, all dependencies are relatively small for all parameters related to the principal components that explain most of the variance.

By comparing the results of PCA applied to the 'transformed' and 'not transformed' data, a better insight into the pattern formation revealed by Figure 3.5 can be achieved. The first three principal components explain 74.6% of the original data, as shown in Figure 3.7. Figure 3.8 shows results for the 'transformed' data.

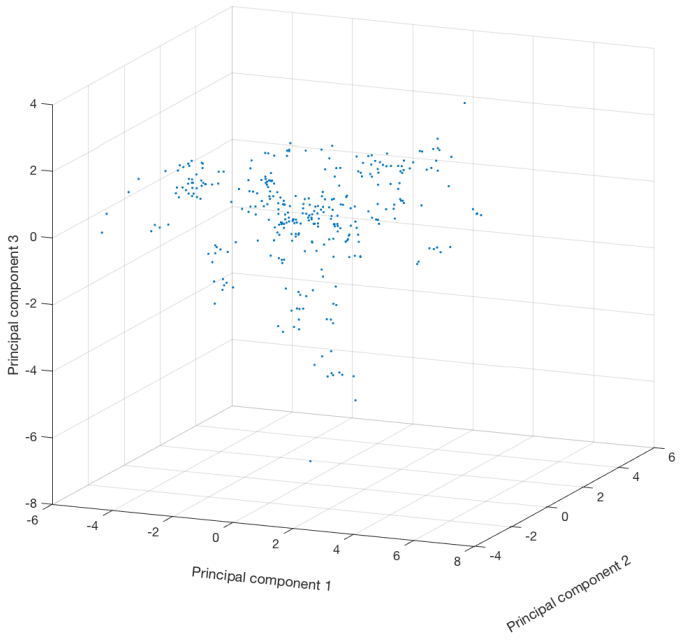


Figure 3.7: Scatter plot of the first three principal components from the 'transformed' data.

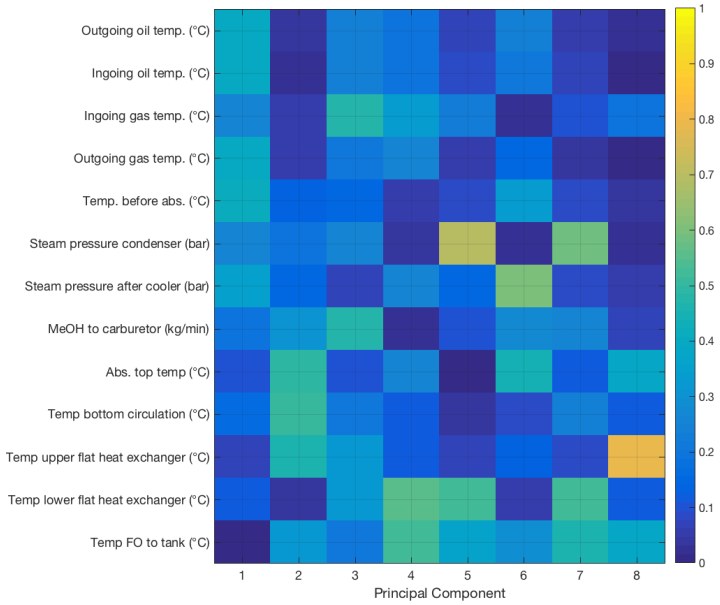


Figure 3.8: Heatmap showing the relation between the first eight principal components and the original parameters for the 'transformed' data.

No clear cluster pattern can be seen in Figure 3.7. One could understand it by considering the pattern revealed in Figure 3.5, which could contain only two groups, is related to each of the two reactors, taken that the 'transformed' data represents a linear combination of the two reactors, whilst no transformation was applied to the absorption column's data. Correlation pattern in Figure 3.8 and 3.4a are similar, with exception to the absorption column's parameters.

The analysis has revealed patterns and clusters in the data, suggesting that there is a difference between the two reactors. Furthermore, no single parameter can be distinguished to affect the majority of the entire processing system.

3.3 Supervised learning

The goal of supervised learning is to build a predictive model in terms of predictor variables, where knowledge obtained from the unsupervised learning algorithms can assist in finding and excluding nonessential predictors.

3.3.1 Preparing the data for regression based learning

To train regression models, data needs to be split into two sets; the training and validation set. Each of the two non-normalized sets of data ('not transformed' and 'transformed') are divided into the two seperate sets using the MATLAB's function *cvpartition* with the argument $'Holdout'$, which specifies the percentage of data that will be put into the validation set. The function will randomly divide the rows of measurements for one selected parameter, while attempting to evenly distribute measurements with equal values between the two sets. To set the seed of the random number generator, the MATLAB's function $rng(n)$ is used. To create the training and validation sets, the seed (n) was set to 1234. The parameter of choice for the *cvpartition* function was the concentration of methanol, where the percentage of measurements put into the validation set was 30%, i.e. 98 randomly selected rows from the formalin data and 231 rows in the training set. The randomly selected rows of measurements obtained from the *cvpartition* function are used to divide both the 'transformed' and the 'not transformed' data into seperate training and validation sets.

3.3.2 Regression and predictions

The linear regression algorithm has been applied to both the 'not transformed' and 'transformed' data sets using the *fitlm(X,Y)* function with the default model specification *'linear'*, where X is the training set and Y is the corresponding vector of responses (concentration of methanol). After the training, the MATLAB's function *predict(mdl,valid)*, where *mdl* is a MATLAB model object containing the regression model and *valid* is the validation data set, is used to predict the concentration of methanol for all the following regression models. The error for each corresponding model is calculated by mean square error (MSE) and by taking the mean of the absolute values for all percentage error for each prediction (Mean Abs. Perc. Error).

Figure 3.9 displays the predicted values of the linear regression model for the validation set, comparing them to the analyzed methanol concentration, shown in blue. The predictions follow the methanol conentration fairly well considering that the difference between the measurements are relatively small. The error calculation for both models are presented in Figure 3.10 where the red bars is the 'transformed' data, the black bars are the 'not transformed' and the vertical axes display the number of measurements included in each bar.

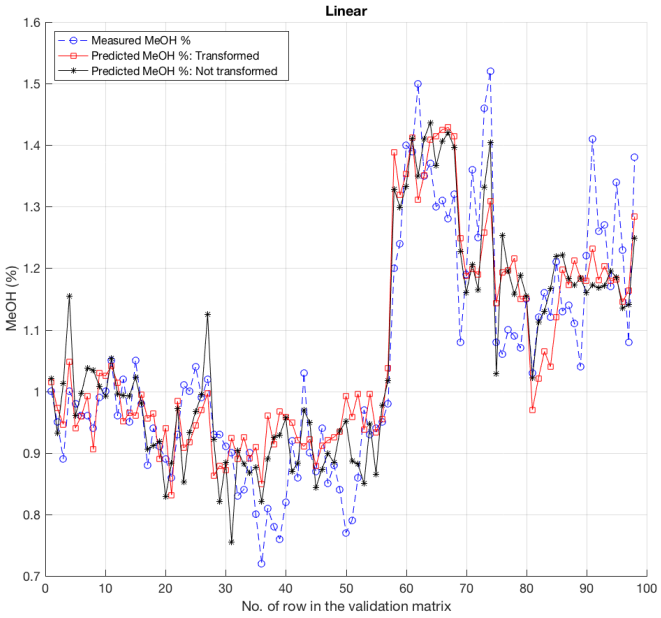


Figure 3.9: The predicted concentration of methanol using linear models (the measured methanol concentration in blue).

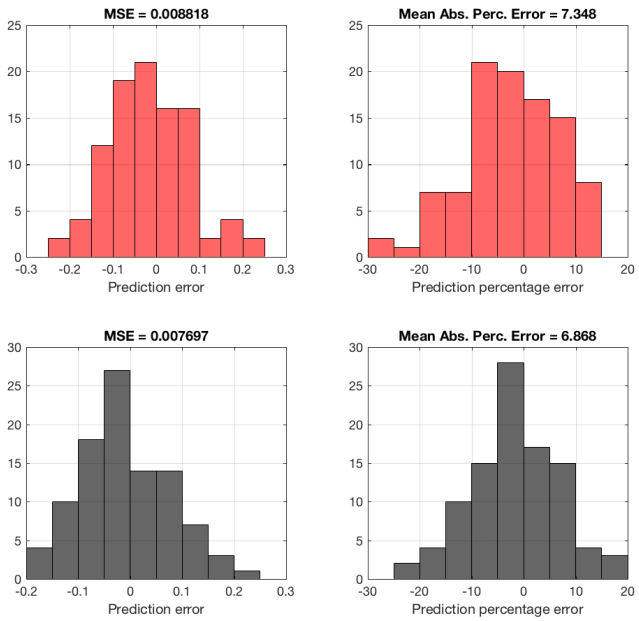


Figure 3.10: Error calculations for the linear models. The red is the error for the 'transformed' data and the the black for the 'not transformed'.

The two support vector regression machines (SVRMs) models are trained using the function $fitrsvm(X, Y)$ and the predictions for the validation data are presented in Figure 3.11, whilst the error in Figure 3.12. Comparing the results obtained from the linear regression models with the results from the SVRMs shows that the linear regression exhibit the smallest error and predicts the concentration of methanol more accurately.

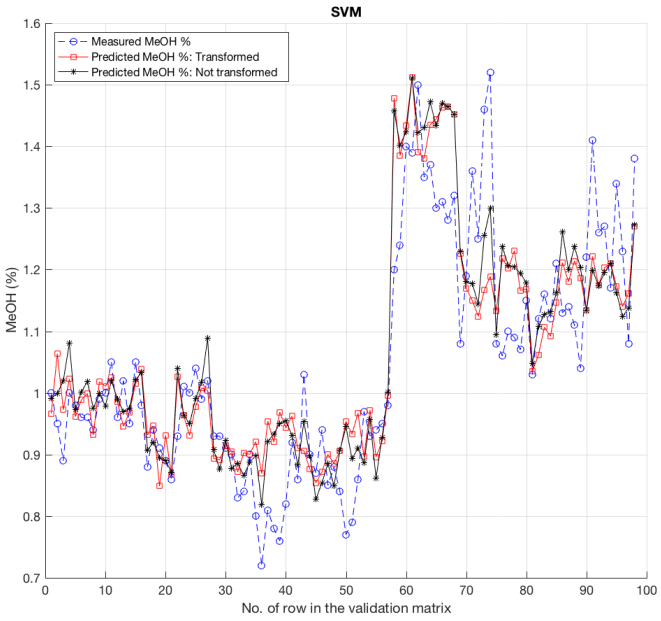


Figure 3.11: The predicted concentration of methanol using SVRMs models (the measured methanol concentration in blue).

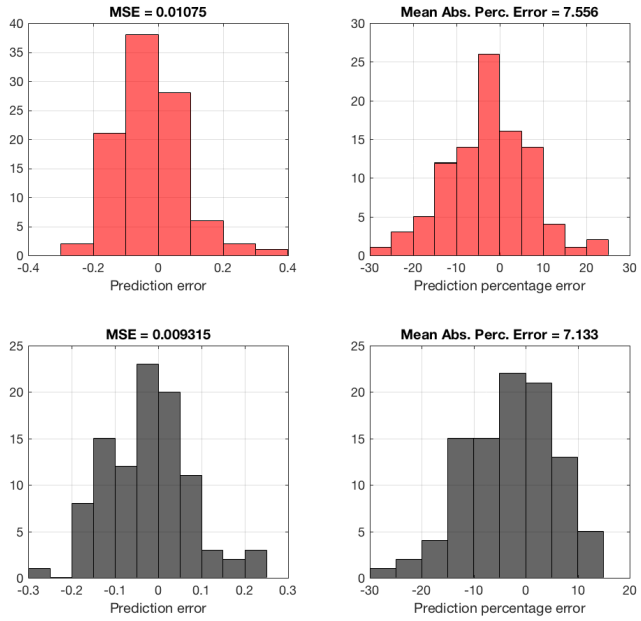


Figure 3.12: Error calculations for the SVRMs models. The red is the error for the 'transformed' data and the the black for the 'not transformed'.

The GPR models are trained using the MATLAB's function $fitrgp(X, Y)$. Figures 3.13 and 3.14 display the predicted concentration of methanol and the prediction error respectively.

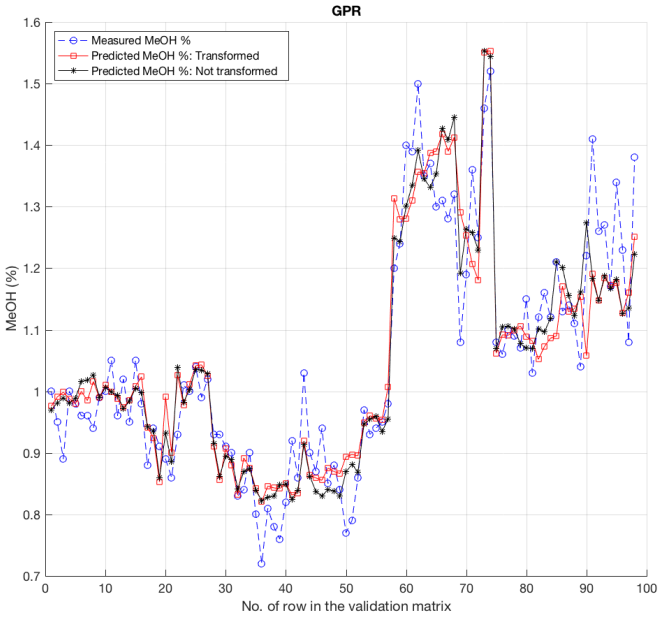


Figure 3.13: The predicted concentration of methanol using GPR models (the measured methanol concentration in blue).

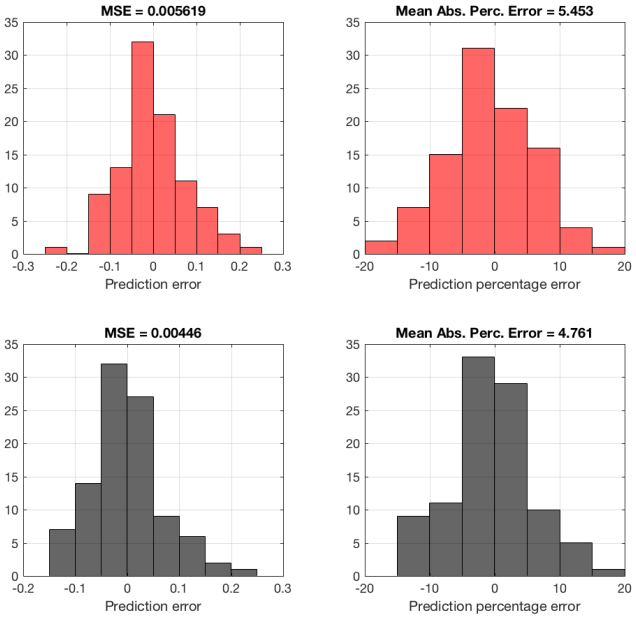


Figure 3.14: Error calculations for the GPR models. The error is depicted in red for the 'transformed' data and black for the 'not transformed'.

Based on the analysis of the different regression models, it has been found that the GPR model provides the most accurate predictions for the methanol concentration.

Fitting of the neural network model is done using the application Neural Net Fitting^{TM}, which is part of the Statistics and Machine Learning toolbox. The data fed to the application is partitioned automatically. To train the neural network models the following settings were used; 70% of the data is selected for training, 15% for validation and 15% for testing, with 10 hidden layers and the training algorithm *Bayesian Regularization*. The validation data is used to assess the network generalization, and the application halts the training when the generalization does not change. The testing data set is independent of the training procedure, thus provides an indepentent measure of the models performance during and after training. The neural network model were trained only once for the 'transformed' and 'not transformed' data sets.

Because of the way neural network models are trained, it cannot be determined what measurements from the data set will be selected by the application for the training, validation and testing sets. It is therefore very likely that many of the measurements in the previously created validation sets have also been used in the training of the neural network models. Thus, it would be quite risky to estimate the quality of the neural network and regression models by comparing the predicted concentrations of methanol based on the same validation sets. Instead, the quality

of the neural network model will be asserted by comparing the model's predictions with those of the GPR models for the same validation set not included in the training data of any of the models. The data used for the analysis were recorded between May and June 2016. The corresponding predictions are presented in Figure 3.15 and 3.16.

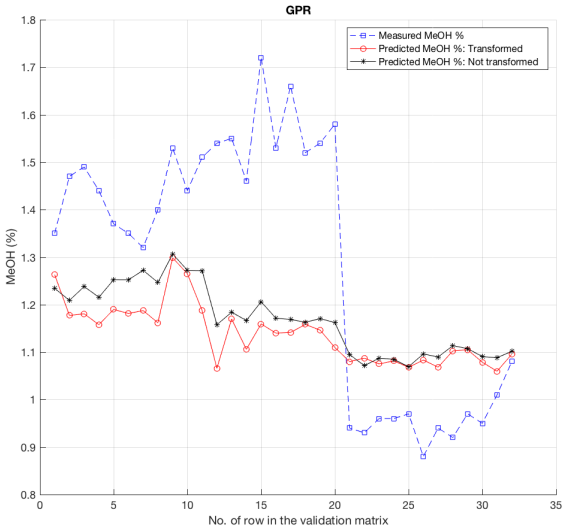


Figure 3.15: The predicted methanol concentration by the GPR models for May and the first six days in June.

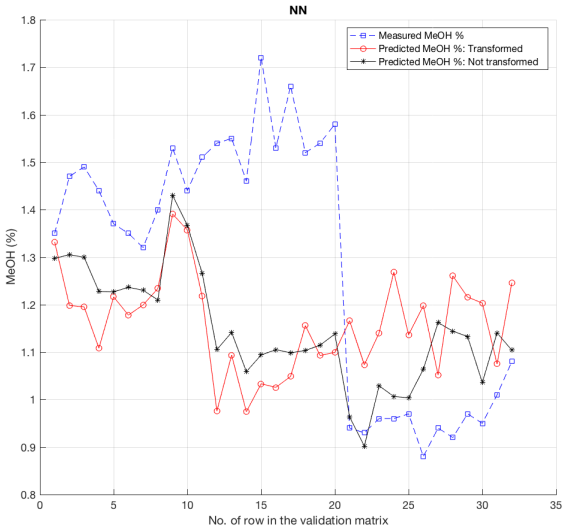


Figure 3.16: The predicted methanol concentration by the neural network models for May and the first six days in June.

The predicted concentration of methanol does not exactly follow the experimental concentration, as seen in the previous figures. Moreover, the regression models are not capable of predicting the signifiquent drop in methanol concentration seen in the experimental data, most likely due to overfitting of the training data set. The risk of overfitting data is especially high for small data sets. The accuracy of the predictions based on the regression models is also affected by a high frequency in the variation of the measured data which the predictions are compared, Figure 3.17. Since the variation is unavoidable due to the measuring technique the MATLAB's function *filter* was used to smoothen the measured methanol concentration, using the moving average method. Figure 3.17 presents the originally measured methanol concentration (blue) and the smoothen data (red).

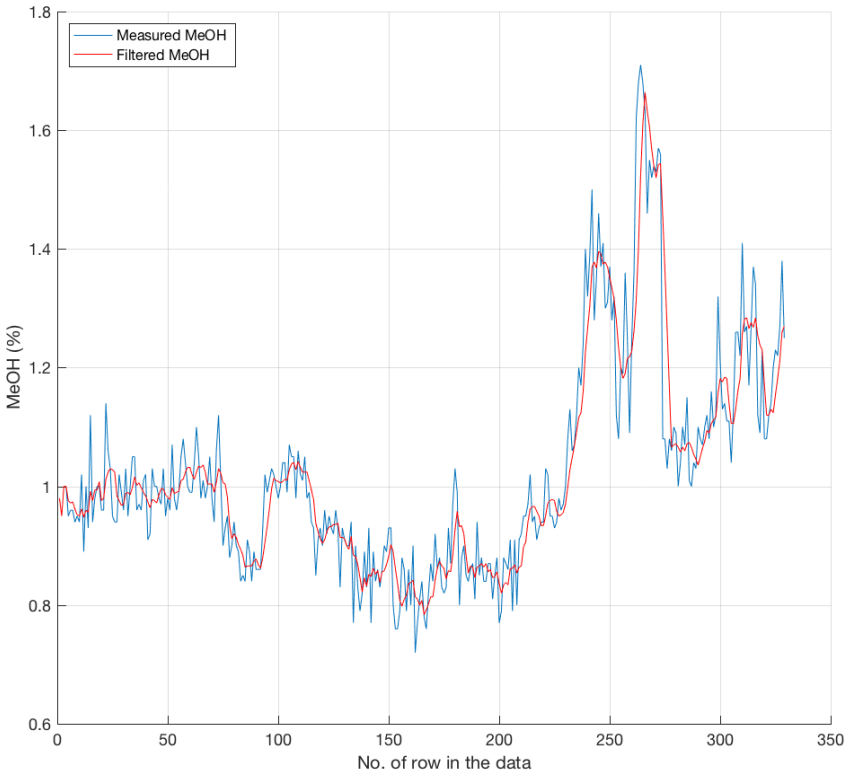


Figure 3.17: The measured methanol concentration before and after the moving average filter.

Next both the GPR models and the neural network models were trained using the same predictor data and the filtered response data (smoothen methanol concentration).

As shown in Figures 3.18 and 3.19, the accuracy of the predictions of the different models have been improved, i.e. the significant drop in the methanol concentration is now well predicted, however there are still quite considerable differences between the predicted and measures concentration of methanol. Nonetheless, the GPR model is clearly more accurate than the neural network model, Figure 3.19, and the 'transformed' data is less accurate than the 'not transformed', suggesting that representing both reactors as one ('transformed' data) does not improve the quality of the model's predictions. In the Appendix (Figure A.1 to A.4), the accuracy of the GPR models' predictions for the validation set can be seen with the filtered methanol concentration, supporting the claim that the variation seen in Figure 3.17 was affecting the quality of the predictive models.

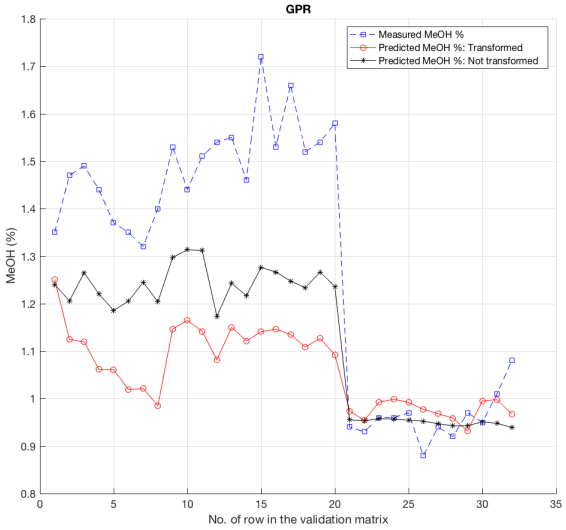


Figure 3.18: The measured methanol concentration for May and the first few days in June, for the GPR model fitted to the filtered concentration of methanol.

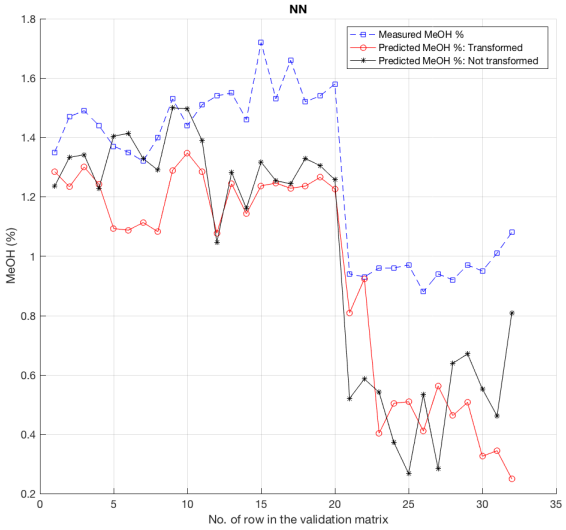


Figure 3.19: The measured methanol concentration for May and the first few days in June, for the neural network model fitted to the filtered concentration of methanol.

It seems reasonable to assume that removing some of the parameters from the data set, which do not affect the concentration of methanol, could improve the accuracy of the models. The PCA analysis can in general be used to determine which parameters can be removed from the orginial data set in the data preprocessing phase. However, it is not always possible to clearly decide which parameters to be removed from the original data set before training a regression model. Therefore deep knowledge of the industrial systems is extremely valuable too. As a result of discussions involving the process engineering group from Akzo Nobel, a subset of the reactors parameters; *in- and outgoing gas temperature, steam pressure after cooler and temperature before the absorber*, have been selected which most likely will not affect the concentration of methanol. Through trial and error, the models were retrained with the same data, each time one or more of the mentioned parameters would be removed until all possible variations had been tested. In the end the removal of all the mentioned parameters; *gas temperatures, temperature before absorber and steam pressure after cooler*, provided the most accurate model, Figures 3.20 and 3.21. Without any doubt the GPR model is much more stable and accurate than the neural network model. Furthermore, the other two models (linear regression and SVRMs) were also fitted to the filtered methanol data, but none of them would give nearly as good predictions as the GPR model, Figure 3.20.

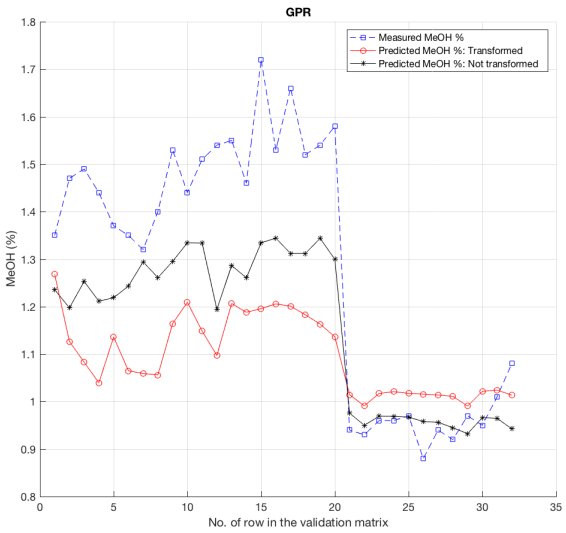


Figure 3.20: The measured methanol concentration for May and the first few days in June, for the GPR model fitted to the filtered concentration of methanol, without the parameters gas temperature, cooler and temperature before absorber.

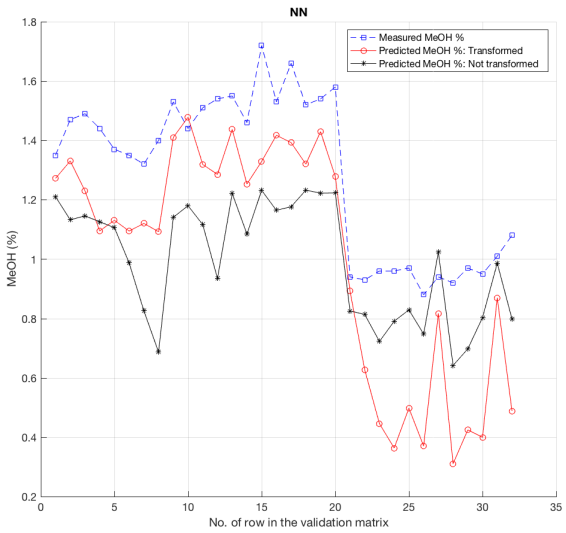


Figure 3.21: The measured methanol concentration for May and the first few days in June, for the neural network model fitted to the filtered concentration of methanol, without the parameters gas temperature, cooler and temperature before absorber

The analysis has clearly shown that the most accurate predictions for the methanol concetration are provided by the GPR models, trained on filtered methanol concentration data and reduced number of predictors.

3.3.3 Stand-alone application

A graphical user interface (Figure 3.22) was added to the MATLAB code implementing the model and into a stand-alone application using the MATLAB Compiler. The application can be used to import and automatically partition the training data from the excel file 'Training.xlsx', and the data used for the predictions from the file 'Predict.xlsx'. The user can then select how many predictor variables that will be used in the training process of the GPR model. When the training is completed, the model can be used to predict the concentration of methanol based on the prediction data. The excel file 'Predictions.xlsx' must be selected as it will contain the predicted concentration of methanol. All actions are forced to be performed in a specific order to ensure the robustness of the application.

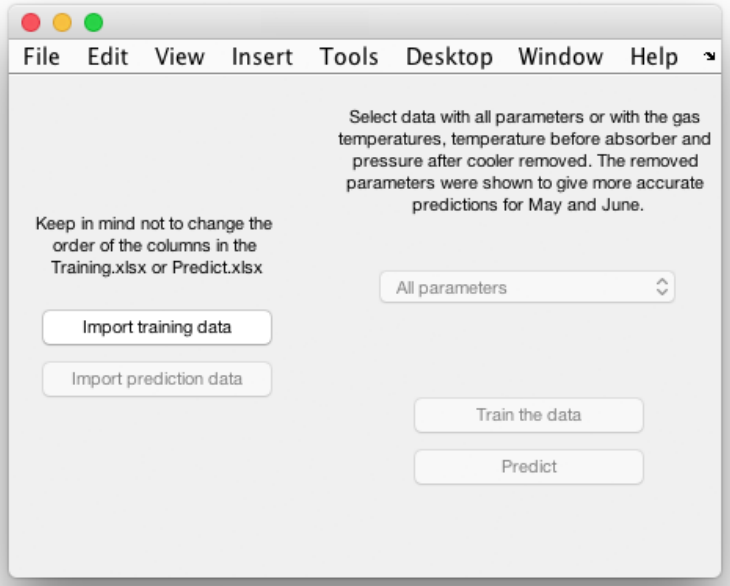


Figure 3.22: The graphical user interface for the stand-alone application on OS X operating system.

4 Discussion

4.1 Unsupervised learning

In the previous sections, various interesting results were obtained using the unsupervised learning methods. The difference in the computed correlations indicate stronger relationships between all parameters in the second reactor's system (Figure 3.1b) than the first reactor's system (Figure 3.1a). This difference could be explained by the different lengths of the tubes, the different amount of the ceramic mixture layers* and their respective age.

The PCA analysis reveals natural patterns and clustering in various sets of the analyze data, e.g. Figures 3.3a, 3.3b and 3.5. The patterns in Figures 3.3a and 3.3b illustrate the difference between the reactor systems, where the data points representing reactor two form a "denser" cluster. Since reactor two is newer, it may provide higher processing efficiency compared to reactor one, therefore the "denser" cluster. However, there are several other components in both systems that may also affect the pattern, e.g. the cooler or the isolated oil coolant system. Additional pattern structure is also seen in Figures 3.3a and 3.3b, most likely depending on the *ingoing and outgoing oil temperature* parameters. The other parameters are mostly kept at constant levels, whereas the oil temperatures are continuously increased until the respective reactor's catalysts are replaced and then the temperatures are decreased again, i.e. the largest change of values between all measurements is seen in the oil temperature parameters. However, the *methanol to carburetor* parameter could also be affecting the pattern depending on the amount of methanol pumped into each reactor, as this parameter also changes notably in value. To get a deeper understanding of the relationships in the reactor systems, other unsupervised learning methods can be used, but those will not be considered in this thesis.

The clusters seen in Figure 3.5 can be considered seperate as they disappear when the 'transformed' data is constructed, see Figure 3.7. Thus suggesting that the cluster patterns are due to the difference in technical parameters of the two reactor systems. The concept of the 'transformed' data aimed at lowering the number of predictors in the regression models, hence increasing the accuracy by representing the two reactors as one, considering that the parameters of both systems are the same. However the results from PCA and correlation analysis suggests that the two parallel reactor systems differ more than what was originally assumed.

*The different layers are made of mixtures between the catalyst rings and the inert rings.

To control the processing system, both reactors' parameters would need to be regulated simultaneously, Figure 3.6. Additonally, Figure 3.8 shows stronger relationships of the parameters for the 'transformed' data because of the reduced number of parameters, corresponding to the results obtained for reactor one (Figure 3.4a). Hence, reactor one may affect the operation of the whole system stronger than reactor two. Thus, it seems to be possible to fine-tune reactor two more easily than reactor one, by adjusting all of its parameters simultaneously.

4.2 Supervised learning

The GPR model trained on the 'not transformed' data (reduced parameters and smoothen methanol concentration) has been found to provide the most accurate predictions of the residual methanol concentration.

It is possible to improve the created neural network model to provide more accurate predictions. However, as it is a feedforward network and every time the neural network model is trained it is likely to be different from the previous one, it is therefore difficult to produce the same results repeateadly when larger training sets are used. This is not a major concern as to GPR model because only different training data sets are used.

Since the GPR model has been trained on a relatively small set of data (329 measurements in a 21 dimensional parameter space), it may not provide as accurate predictions for very large data sets. The possible ways of improving the model could be; using larger training sets which will decrease the risk of overfitting the data and increase the accuracy of predictions based on the model.
Removing outliers from the methanol concentration could additionally decrease the risk of overfitting and hence improve the models accuracy.

4.2.1 Stand-alone application

The application can be used to predict the methanol concentration on new or anticipated data, providing an alternative way of fine-tuning the control parameters before they are changed in the processing system. It will provide the user with a fast and reliable method for changing and experimenting with the control parameters without affecting the production processing system, which was not available prior to this work. The application is highly customizable, i.e. many different MATLAB functions can be supplemented to the application if the need arises.

5 Conclusion

Processing the formalin production data using unsupervised learning methods and analyzing the results in a 3-dimensional space of the first three, most variance explaining principal components has revealed a distinct pattern in the data, consisting of two groupings (clusters). The clusters can be related to the two reactors, which together with the absorption column constitute the formalin production system used in the Akzo Nobel's factory. Their existence may also indicate differences between the reactors resulting from age (reactor one is older than reactor two) and technical details (different length of the tubes and different number of mixture layers). Any attempt to control and fine tune the system to lower the concentration of residual methanol in formalin would need to involve setting the two reactors' parameters independently i.e. the two reactors should not be treated as one reactor system described by the same set of parameters.

It has been found that the Gaussian Process Regression method provided the best predictions of the concentration of residual methanol in formalin and hence was used as the base algorithm for the predictive model. The model could also be used in a next-step analysis to find out how the concentration of residual methanol could be related to the production system's control parameters. Knowing the relations would help to determine a set of the parameters to be tweaked to lower the methanol concentration.

The MATLAB code implementing the model was equipped with a GUI and compiled into a stand-alone application using the MATLAB Compiler. The application can now be used as part of the production process, modified and extended if a need be.

Bibliography

[1] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*. Englewood, NJ: Prentice Hall., 1995.

[2] Jolliffe, I.T., *Principal Component Analysis*. 2nd ed. New York: Springer., 2002.

[3] Kutner, M. H., Nachtsheim, C. J., Neter, J. and Li, W., *Applied Linear Statistical Models*. 5th ed. New York: McGraw-Hill., 2005.

[4] MathWorks documentation: *Linear Regression*. Available from: http://se.mathworks.com/help/stats/linear-regression-model-workflow.html# btb50qx.
Retrived 23 May 2016

[5] MathWorks documentation: *What Are Linear Regression Models?*. Available from: http://se.mathworks.com/help/stats/what-is-linear-regression.html.
Retrived 30 May 2016

[6] Rasmussen, C. E. and Williams, C. K. I., *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: MIT Press., 2006.

[7] MathWorks documentation: *Gaussian Process Regression Models*. Available from: http://se.mathworks.com/help/stats/gaussian-process-regression-models.html.
Retrived 01 June 2016

[8] Cortes, C. and Vapnik, V., *Support-Vector Networks*. Machine Learning, vol. 20, issue 3, 273-297, 1995.

[9] Drucker, H., et al., *Support Vector Regression Machines*. Advances in Neural Information Processing Systems 9, 155-161, 1996.

[10] MathWorks documentation: *Understanding Support Vector Machine Regression*. Available from: http://se.mathworks.com/help/stats/understanding-support-vector-machine-regression.html# buytaw5.
Retrived 01 June 2016

[11] Raúl Rojas., *Neural Networks A Systematic Introduction*. Berlin: Springer., 1996., Available from: https://page.mi.fu-berlin.de/rojas/neural/neuron.pdf.
Retrived 28 May 2016

[12] MathWorks documentation: *Neural Network Toolbox*. Available from: http://se.mathworks.com/help/nnet/index.html
Retrived 05 June 2016

[13] MathWorks documentation: *corr*. Available from: http://se.mathworks.com/help/stats/corr.html.
Retrived 30 May 2016

[14] MathWorks documentation: *zscore*. Available from: http://se.mathworks.com/help/stats/zscore.html.
Retrived 28 May 2016

[15] MathWorks documentation: *pca*. Available from: http://se.mathworks.com/help/stats/pca.html.
Retrived 28 May 2016

Appendix

If the reader is interested in obtaining the excel files 'Training.xlsx' and 'Predict.xlsx' which this thesis uses, please contact Stefan Kvarth at; Stefan.kvarth@akzonobel.com

Gaussian process regression

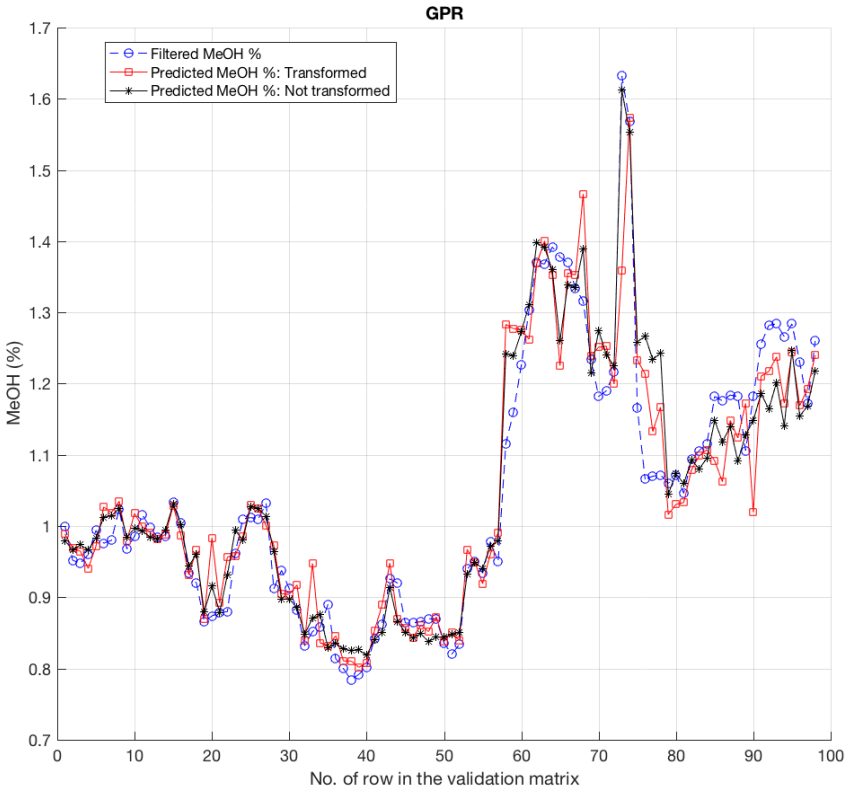


Figure A.1: Predicted concentration of methanol for the GPR models, trained using the filtered methanol concentration (blue).

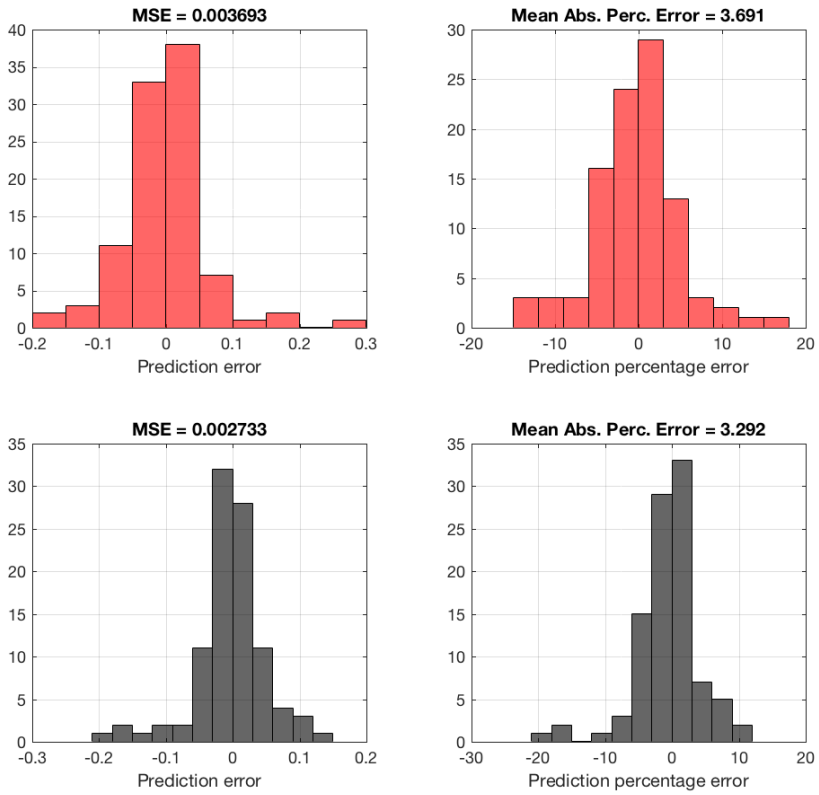


Figure A.2: Error calculations for the GPR models, trained with the filtered methanol concentration. The red is the error for the 'transformed' data and the the black for the 'not transformed'.

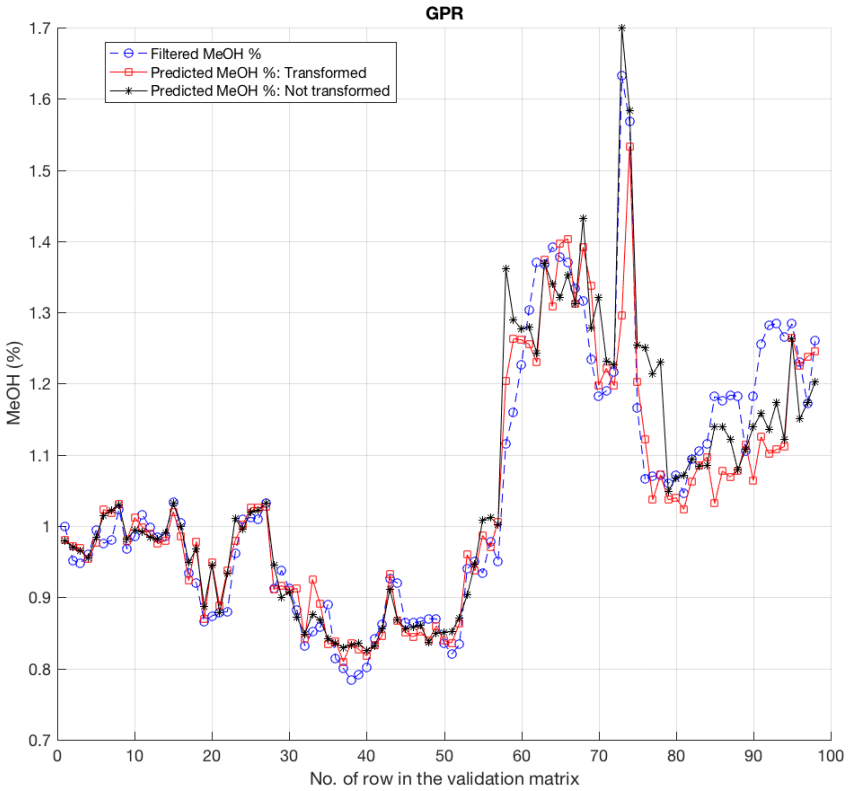


Figure A.3: Predicted concentration of methanol for the GPR models, trained using the filtered methanol concentration (blue) with reduced parameters.

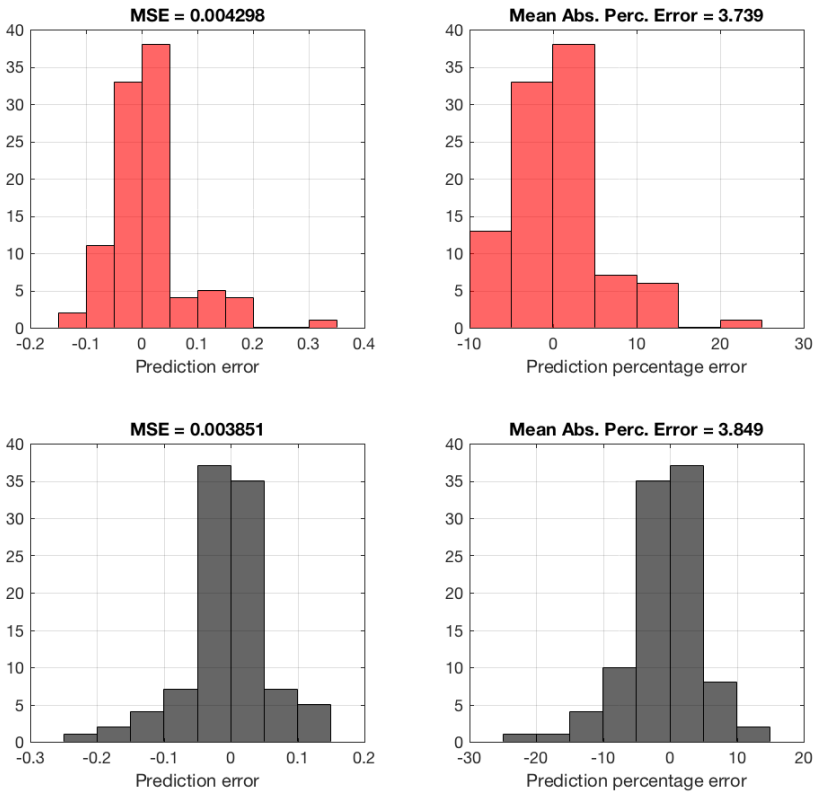


Figure A.4: Error calculations for the GPR models, trained with the filtered methanol concentration and reduced parameters. The red is the error for the 'transformed' data and the the black for the 'not transformed'.

Main MATLAB code

```
%% Import data
     [data,varnames,~] = Import('Training.xlsx');
     [datapred,varnamespred,idxP] = Import('Predict.xlsx');
% Vectors for each section of the data
     idxtot = {[1:8], [9:16], [1:21]};
% Normalizing the data
     R1norm = zscore(data(:,idxtot{1}));
     R2norm = zscore(data(:,idxtot{2}));
     datanorm = zscore(data(:,idxtot{3}));
%%%%%%%%%%%%%%%%%%%%%%%%%

%% Combine both reactors through linear transformation for both sets of data
% Training set
     Rtrans = (data(:,idxtot{1}) + data(:,idxtot{2}))./2;
     datatrans = [Rtrans data(:,17:21)];
     datatransnorm = zscore(datatrans);
% Prediction data set for May and June
     Rtranspred = (datapred(:,idxtot{1}) + datapred(:,idxtot{2}))./2;
     datatranspred = [Rtranspred datapred(:,17:21)];
%%%%%%%%%%%%%%%%%%%%%%%%%

%% Corr R1 and R2
     Rstr = {'R1: ','R2: '};
     for i = 1:2
          figure(i)
          imagesc(abs(corr(data(:,idxtot{i}))))
          labelXTicks(strcat(Rstr{i},varnames(1:8)))
 labelYTicks(strcat(Rstr{i},varnames(1:8)))
          colorbar
          caxis([0 1]) % change colorbar limits
          plotsettings
     end
%%%%%%%%%%%%%%%%%%%%%%%%%

%% PCA for each reactor
     [R1pcs,R1scrs,~,~,R1pexp] = pca(R1norm);
     figure
     pareto(R1pexp)
     title('R1')
     cumsum(R1pexp)
```

```
      [R2pcs,R2scrs,~,~,R2pexp] = pca(R2norm);
      figure
      pareto(R2pexp)
      title('R2')
      cumsum(R2pexp)
%%%%%%%%%%%%%%%%%%%%%%%%

%% Visualize the data
      figure
      scatter3(R1scrs(:,1),R1scrs(:,2),R1scrs(:,3),'.')
      title('R1')
      xlabel('Principal component 1')
      ylabel('Principal component 2')
      zlabel('Principal component 3')
      set(gca,'Xlim',[-6 6],'Ylim',[-6 3],'Zlim',[-3 3])
      plotsettings
      view(26,10)
      set(get(gca,'xlabel'),'rotation',-4)
      set(get(gca,'ylabel'),'rotation',25)
      figure
      scatter3(R2scrs(:,1),R2scrs(:,2),R2scrs(:,3),'.')
      title('R2')
      xlabel('Principal component 1')
      ylabel('Principal component 2')
      zlabel('Principal component 3')
      set(gca,'Xlim',[-6 6],'Ylim',[-6 3],'Zlim',[-3 3])
      plotsettings
      view(26,10)
      set(get(gca,'xlabel'),'rotation',-4)
      set(get(gca,'ylabel'),'rotation',25)
%%%%%%%%%%%%%%%%%%%%%%%%

%% Visualize the relationship between the variables and the first few
% principal components, which represent the greatest amount of variance in
% the data.
      figure
      imagesc(abs(R1pcs))
      labelYTicks(strcat(Rstr{1},varnames(1:8)))
      xlabel('Principal Component')
      colorbar
      caxis([0 1])
      plotsettings
      figure
      imagesc(abs(R2pcs))
      labelYTicks(strcat(Rstr{2},varnames(1:8)))
```

```
     xlabel('Principal Component')
     colorbar
     caxis([0 1])
     plotsettings
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%% PCA on the complete normalized data sets transformed
% and not transformed
     [pcs,scrs,~,~,pexp] = pca(datanorm);
     pareto(pexp)

% Visualize
     figure
     scatter3(scrs(:,1),scrs(:,2),scrs(:,3),'.')
     xlabel('Principal component 1')
     ylabel('Principal component 2')
     zlabel('Principal component 3')
     set(get(gca,'xlabel'),'rotation',3)
     set(get(gca,'ylabel'),'rotation',-40)
     view(-16,14)
     plotsettings

%% Visualize the relationship between the variables and the first few
% principal components, which represent the greatest amount of variance in
% the data.
     figure
     imagesc(abs(pcs(:,1:8)))
     labelXTicks(1:8,0)
     labelYTicks([strcat(Rstr{1},varnames(1:8)), strcat(Rstr{2},varnames(9:16)),..
          varnames(17:21)])
     xlabel('Principal Component')
     colorbar
     caxis([0 1])
     plotsettings

%% PCA on the normalized tranformed data
     [datatranspcs,datatransscrs,~,~,datatranspexp] = pca(datatransnorm);
     figure
     pareto(datatranspexp)

% Visualize
     figure
     scatter3(datatransscrs(:,1),datatransscrs(:,2),datatransscrs(:,3),'.')
     view(23,15)
     xlabel('Principal component 1')
```

```
      ylabel('Principal component 2')
      zlabel('Principal component 3')
      set(get(gca,'xlabel'),'rotation',-6)
      set(get(gca,'ylabel'),'rotation',30)
      plotsettings
      figure
      imagesc(abs(datatranspcs(:,1:8)))
      labelYTicks(varnames([1:8,17:21]))
      xlabel('Principal Component')
      colorbar
      caxis([0 1])
      plotsettings
%%%%%%%%%%%%%%%%%%%

%% Filter Moving average
      rng(1234)
      windowSize = 5;
      b = (1/5)*ones(1,windowSize)
      a = 1;
      respfilter = filter(b,a,data(:,22));
      respfilter(1:4,:) = data(1:4,22);
      figure(1)
      hold on
      plot(data(:,22))
      plot(respfilter,'r')
      hold off
      legend('Measured MeOH','Filtered MeOH','Location','NorthWest')
      xlabel('No. of row in the data')
      ylabel ('MeOH (%)','Rotation',90)
      plotsettings
%%%%%%%%%%%%%%%%%%%

%% Split transformed data into training and validation sets
      rng(1234)
      part = cvpartition(data(:,22),'Holdout',0.30);
      trainidx = training(part);
% Vector for removing parameters, change to [1:2,6,8:10,14,16:21] for idxpara21
% and idxpara to [1:2,6,8:13]
      idxpara21 = [1:21];
      idxpara = [1:13];
% Training set
      datatrain21 = data(trainidx,idxpara21);
      datatrain = datatrans(trainidx,idxpara);

% Test set
```

```
     datatest21 = data(~trainidx,idxpara21);
     datatest = datatrans(~trainidx,idxpara);
% Unfiltered responses
     % resptrain = data(trainidx,22);
     % resptest = data(~trainidx,22);
% Filtered responses
     resptrain = respfilter(trainidx,:);
     resptest = respfilter(~trainidx,:);
%%%%%%%%%%%%%%%%%%%

%% Fit a multivariate linear model, Transformed
     mdl = fitlm(datatrain,resptrain,'linear');
     mdl21 = fitlm(datatrain21,resptrain);
     mdlname = 'Linear';
%% Fit an SVM regression model
     %mdl = fitrsvm(datatrain,resptrain);
     %mdl21 = fitrsvm(datatrain21,resptrain);
     %mdlname = 'SVM';
%% Fit a Gaussian process model
     %mdl = fitrgp(datatrain,resptrain);
     %mdl21 = fitrgp(datatrain21,resptrain);
     %mdlname = 'GPR';
%% Evaluate model at test predictor values Transformed
     pred = predict(mdl,datatest);
% Evaluate model at test predictor values Not transformed
     pred21 = predict(mdl21,datatest21);
%% Plot the predicted test and real respones
     figure
     hold on
     plot(1:numel(resptest),resptest,'b--o','MarkerSize',7);
     plot(1:numel(resptest),pred,'r-sq','MarkerSize',7);
     plot(1:numel(resptest),pred21,'k-*','MarkerSize',6);
     legend('Filtered MeOH %','Predicted MeOH %: Transformed'...
          ,'Predicted MeOH %: Not transformed','Location','Best')
     xlabel('No. of row in the validation matrix')
     ylabel ('MeOH (%)','Rotation',90)
     title(mdlname)
     plotsettings
     hold off
     evaluatefitnew(resptest,pred,pred21)
%%%%%%%%%%%%%%%%%%%%%%%%

%% Prediction for data in May and June
     validpredtrans = predict(mdl,datatranspred(:,idxpara));
     validpredtot = predict(mdl21,datapred(:,idxpara21));
```

```
     figure
     hold on
     plot(datapred(:,22), 'b-sq','MarkerSize',7)
     plot(validpredtrans,'r-o','MarkerSize',7)
     plot(validpredtot,'k-*','MarkerSize',6)
     legend('Measured MeOH %','Predicted MeOH %: Transformed',...
          'Predicted MeOH %: Not transformed','Location','NorthEast')
     xlabel('No. of row in the validation matrix')
     ylabel ('MeOH (%)','Rotation',90)
     title(mdlname)
     plotsettings
%%%%%%%%%%%%%%%%%%%%%%%%
```

MATLAB functions

```
function [data,varnames,idxT] = Import(name)

     if strcmp(name,'Training.xlsx')
           [type,sheetname] = xlsfinfo(name);
          m=size(sheetname,2);
          x = cell(1, m);
          y = cell(1, m);
          for(i=1:1:m);
               Sheet = char(sheetname(1,i)) ;
               [x{i},y{i}] = xlsread(name, Sheet);
          end
          R1 = x{1}; R2 = x{2}; A = x{3}; MeOH = x{4};
          varR1 = y{1}; varR2 = y{2}; varA = y{3}; varMeOH = y{4};
% Remove bad measurements
          idxR1 = R1(:,9) < 15;
          idxR2 = R2(:,9) < 15;
          idxMeOH = isnan(MeOH(:,3));
% Combine data
          idxT = ~(idxR1+idxR2+idxMeOH);
          data = [R1(idxT,2:end) R2(idxT,2:end) A(idxT,2:end) MeOH(idxT,3)];
          varnames = [varR1(:,2:end) varR2(:,2:end) varA(:,2:end) varMeOH(:,3)];
     elseif strcmp(name,'Predict.xlsx')
          [type,sheetname] = xlsfinfo(name);
          m=size(sheetname,2);
          x = cell(1, m);
          y = cell(1, m);
          for(i=1:1:m);
               Sheet = char(sheetname(1,i)) ;
               [x{i},y{i}] = xlsread(name, Sheet);
          end
          R1 = x{1}; R2 = x{2}; A = x{3}; MeOH = x{4};
```

```
          varR1 = y{1}; varR2 = y{2}; varA = y{3}; varMeOH = y{4};
% Remove bad measurements
          idxR1 = R1(:,8) < 15;
          idxR2 = R2(:,8) < 15;
% Combine data
          idxT = ~(idxR1+idxR2);
          data = [R1(idxT,1:end) R2(idxT,1:end) A(idxT,1:end)  MeOH(idxT,1)];
          varnames = [varR1(:,1:end) varR2(:,1:end) varA(:,1:end) varMeOH(:,1)];
     else
          data = 'Wrong';
     end

end

function plotsettings
     width = 8;
     height = 7;
     alw = 0.75;
     fsz = 13;
     pos = get(gcf, 'Position');
     set(gcf, 'Position', [pos(1) pos(2) width*100, height*100]);
     set(gcf, 'PaperPositionMode','auto')
     set(gca, 'FontSize', fsz, 'LineWidth', alw);
     grid on
end

function labelXTicks(labels,rot)

     if nargin < 2
          rot = 60;
     end
     ax = gca;
     ax.XTick = 1:numel(labels);
     ax.XTickLabel = labels;
     ax.XTickLabelRotation = rot;
end

function labelYTicks(labels,rot)
     if nargin < 2
          rot = 0;
     end
     ax = gca;
     ax.YTick = 1:numel(labels);
     ax.YTickLabel = labels;
     ax.YTickLabelRotation = rot;
```

```
end

function evaluatefitnew(y,bpred,ypred)
     figure
% Distribution of errors
     subplot(2,2,1)
     err1 = y-bpred;
     MSE1 = mean(err1.^2,'omitnan');
     histogram(err1)
     set(get(gca,'child'),'FaceColor','r','EdgeColor','k');
     title(['MSE = ',num2str(MSE1,4)])
     xlabel('Prediction error')
     plotsettings
% Distribution of percentage errors
     subplot(2,2,2)
     err1 = 100*err1./y;
     MAPE1 = mean(abs(err1),'omitnan');
     histogram(err1)
     title(['Mean Abs. Perc. Error = ',num2str(MAPE1,4)])
     xlabel('Prediction percentage error')
     set(get(gca,'child'),'FaceColor','r','EdgeColor','k');
     plotsettings
% Distribution of errors
     subplot(2,2,3)
     err = y-ypred;
     MSE = mean(err.^2,'omitnan');
     histogram(err)
     title(['MSE = ',num2str(MSE,4)])
     xlabel('Prediction error')
     set(get(gca,'child'),'FaceColor','k','EdgeColor','k');
     plotsettings
% Distribution of percentage errors
     subplot(2,2,4)
     err = 100*err./y;
     MAPE = mean(abs(err),'omitnan');
     histogram(err)
     title(['Mean Abs. Perc. Error = ',num2str(MAPE,4)])
     xlabel('Prediction percentage error')
     set(get(gca,'child'),'FaceColor','k','EdgeColor','k');
     plotsettings
end
```

Stand-alone application: MATLAB code

```
function Menu

   %  Create and then hide the UI as it is being constructed.
   f = figure('Name','Machine Learning','Toolbar','none','Visible','off',...
       'Position',[360,500,450,285],'NumberTitle','off');

   %  Construct the components.
   % Text box
   htextdata = uicontrol('Style','text','String',...
       ['Keep in mind not to change the order '...
       'of the columns in the Training.xlsx or Predict.xlsx'],'Position',...
       [10,150,170,60],'Units','characters','FontSize',11);
   % Import data buttons
   hdatatrain = uicontrol('Style','pushbutton','String',...
         'Import training data','Enable','on','Position',...
         [30,130,150,25],'Callback',{@datatrainbutton_Callback},'FontSize',11);

     hdatapred = uicontrol('Style','pushbutton','String',...
         'Import prediction data','Enable','off','Position',...
         [30,100,150,25],'Callback',{@datapredbutton_Callback},'FontSize',11);
    % Popup button
      htext = uicontrol('Style','text','String',['Select data with all ' ...
       'parameters or with the gas temperatures, temperature before absorber '...
       'and pressure after cooler removed. The removed parameters were ' ...
       'shown to give more accurate predictions for May and June.'], ...
           'Position',[260,180,250,90],'Units','characters','FontSize',11);
   hpopup = uicontrol('Style','popupmenu',...
           'String',{'All parameters','Removed parameters'},...
            'Enable','off','Position',[150,150,200,25],...
           'Callback',@popup_Callback,'FontSize',11);
    % Training button
  htrain = uicontrol('Style','pushbutton','String','Train the data',...
         'Enable','off','Position',[150,80,150,25],...
        'Callback',{@trainbutton_Callback},'FontSize',11);
    % Predict button
     hpredict = uicontrol('Style','pushbutton','String','Predict',...
         'Enable','off','Position',[150,50,150,25],...
        'Callback',{@predictbutton_Callback},'FontSize',11);

   align([htext,hpopup,htrain,hpredict],'Center','None');
   align([htextdata,hdatatrain,hdatapred],'Center','None');

   % Initialize the UI.
```

```
% Change units to normalized so components resize automatically.
f.Units = 'normalized';
htextdata.Units = 'normalized';
hdatatrain.Units = 'normalized';
hdatapred.Units = 'normalized';
htrain.Units = 'normalized';
hpredict.Units = 'normalized';
htext.Units = 'normalized';
hpopup.Units = 'normalized';

   % Assign the a name to appear in the window title.
% Move the window to the center of the screen.
movegui(f,'center')

   %Make the UI visible.
   f.Visible = 'on';

    function datatrainbutton_Callback(hObject,eventdata,handles)
          handles = guidata(hObject);
  [FileName,PathName,FilterIndex] = uigetfile();
  if strcmp(FileName,'Training.xlsx')
fullAddress = strcat(PathName,FileName);
[data,~] = ImportData(FileName,fullAddress);
% Split data into training and validation sets
rng(1234)
part = cvpartition(data(:,22),'Holdout',0.30);
 trainidx = training(part);

% Filter Moving average
rng(1234)
b = (1/5)*ones(1,5);
a = 1;
MeOHfilter = filter(b,a,data(:,22));
MeOHfilter(1:4,:) = data(1:4,22);

 % Create handles
 handles.data = data;
 handles.MeOHfilter = MeOHfilter;
 handles.trainidx = trainidx;
guidata(hObject,handles) %Update handles

set(hdatapred,'Enable','on');

msgbox('Importing Completed');
```

```
  else
      errordlg('Training.xlsx was not selected!')
  end
    end

function datapredbutton_Callback(hObject,eventdata,handles)
      handles = guidata(hObject);
  [FileName,PathName,FilterIndex] = uigetfile();
  if strcmp(FileName,'Predict.xlsx')
fullAddress = strcat(PathName,FileName);
[datapred,idxP] = ImportData(FileName,fullAddress);
 handles.datapred = datapred;
 handles.idxP = idxP;
 guidata(hObject,handles) %Update handles
 set(hpopup,'Enable','on');
 % Import the data.
msgbox('Importing Completed');
      else
      errordlg('Predict.xlsx was not selected!')
  end

end

%  Pop-up menu callback. Read the pop-up menu Value property to
%  determine which item is currently displayed and make it the
%  current data. This callback automatically has access to
%  current_data because this function is nested at a lower level.
   function popup_Callback(hObject,eventdata,handles)
    % Retrive GUI data
     handles = guidata(hObject);
     data =  handles.data;
     MeOHfilter = handles.MeOHfilter;
     trainidx = handles.trainidx;
 % Determine the selected data set.
      str = get(hObject, 'String');
      val = get(hObject,'Value');
      % Set current data to the selected data set.
      switch str{val};
      case 'All parameters' % User selects All Parameters.
          idx = [1:21];
          idxR1 = [1:8];
          idxR2 = [9:16];
          %Training set
         datatrain = data(trainidx,idx);
         resptrain = MeOHfilter(trainidx,1);
```

```
         %Validation set
        datatest = data(~trainidx,idx);
        resptest = MeOHfilter(~trainidx,1);
      case 'Removed parameters' % User selects Removed Parameters.
          idx = [1:2,6,8:10,14,16:21];
          idxR1 = [1:2,6,8];
          idxR2 = [9:10,14,16];
         % Train set
        datatrain = data(trainidx,idx);
        resptrain = MeOHfilter(trainidx,1);
        %Validation set
        datatest = data(~trainidx,idx);
        resptest = MeOHfilter(~trainidx,1);
      end

      % Create more handles
 handles.datatrain = datatrain;
 handles.datatest = datatest;
 handles.resptrain = resptrain;
 handles.resptest = resptest;
 handles.idxparameter = idx;
 handles.idxR1 = idxR1;
 handles.idxR2 = idxR2;
guidata(hObject, handles) %Update handles

set(htrain,'Enable','on');
set(hpredict,'Enable','off');
   end

   function trainbutton_Callback(hObject,eventdata,handles)
   % Retrive GUI data
     handles = guidata(hObject);
     datatrain = handles.datatrain;
     resptrain = handles.resptrain;
     datatest = handles.datatest;
     % Train the GPR model
    mdl = fitrgp(datatrain,resptrain);
   % Evaluate model at test predictor values
  [pred,~,~] = predict(mdl,datatest);
  msgbox('Training Completed');
  % Declare handles
  handles.mdl = mdl;
  handles.pred = pred;
  guidata(hObject,handles) % Update handles
```

```
  set(hpredict,'Enable','on');
   end

function predictbutton_Callback(hObject,eventdata,handles)
    [FileName,PathName,FilterIndex] = uigetfile();
    fullAddress = strcat(PathName,FileName);
   % Retrive GUI data
     handles = guidata(hObject);
     datapred = handles.datapred;
     mdl = handles.mdl;
     idx = handles.idxparameter;
     idxP = handles.idxP;

   % Predict values
  [pred,~,~] = predict(mdl,datapred(:,idx));

% Fill the 0 in idxP with NaN and 1 with pred
  n = size(idxP);
Main = NaN(n);
j = 0;
for i = 1:n(1,1)
    if idxP(i,1) == 1
        j = j+1;
        Main(i,1) = pred(j,1);
    end
end

 T = array2table(Main,'VariableNames',{'Predicted_MeOH'});

writetable(T,fullAddress,'Sheet','Predictions','Range','C4')

msgbox('Predictions Completed');
   end
end
```

MATLAB function used in the application

```
function [data,idxT] = ImportData(name,fullAddress)

     if strcmp(name,'Training.xlsx')
           [~,sheetname] = xlsfinfo(fullAddress);
          m=size(sheetname,2);
          x = cell(1, m);
          for i=1:m
               Sheet1 = char(sheetname(1,i)) ;
```

```
            x{i} = readtable(fullAddress, 'Sheet',Sheet1);
        end
        R1 = x{1,1}{:,2:end}; R2 = x{1,2}{:,2:end};
        A = x{1,3}{:,2:end}; MeOH = x{1,4}{:,3};
% Remove bad measurements
        idxR1 = R1(:,end) < 15;
        idxR2 = R2(:,end) < 15;
        idxMeOH = isnan(MeOH(:,1));
% Combine data
        idxT = ~(idxR1+idxR2+idxMeOH);
        data = [R1(idxT,:) R2(idxT,:) A(idxT,:) MeOH(idxT,:)];

    elseif strcmp(name,'Predict.xlsx')
        [type,sheetname] = xlsfinfo(fullAddress);
        m=size(sheetname,2);
        x = cell(1, m);
        for i=1:m
            Sheet1 = char(sheetname(1,i)) ;
            x{i} = readtable(fullAddress, 'Sheet',Sheet1);
        end
        R1 = x{1,1}{:,1:end}; R2 = x{1,2}{:,1:end};
        A = x{1,3}{:,1:end};
% Remove bad measurements
        idxR1 = R1(:,8) < 15;
        idxR2 = R2(:,8) < 15;
% Combine data
        idxT = ~(idxR1+idxR2);
        data = [R1(idxT,:) R2(idxT,:) A(idxT,:)];

    else
        data = 'Wrong';
end
```