CHALMERS

Selection of preprocessing methods for Partial Least Squares regression of acoustic spectrometry data using a Genetic Algorithm

Master of Science Thesis in Communication Engineering

Sebastian Johansson Mauricio

Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden 2011

Report No. EX062/2011

MASTER OF SCIENCE THESIS 2011

Selection of preprocessing methods for Partial Least Squares regression of acoustic spectrometry data using a Genetic Algorithm

Master of Science Thesis in Communication Engineering

SEBASTIAN JOHANSSON MAURICIO

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY

Göteborg, Sweden 2011

Selection of preprocessing methods for Partial Least Squares regression of acoustic spectrometry data using a Genetic Algorithm
SEBASTIAN JOHANSSON MAURICIO

Master of Science Thesis 2011
Department of Signals and Systems

Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Göteborg, Sweden 2011

Selection of preprocessing methods for Partial Least Squares regression of acoustic spectrometry data using a Genetic Algorithm
Master of Science Thesis in Communication Engineering
SEBASTIAN JOHANSSON MAURICIO
Department of Signals and Systems
Chalmers University of Technology

Abstract

Acoustic spectrometry is a fast and noninvasive means of measuring liquid properties inside tubes in processing plants. Partial least squares regression is a popular method in chemometrics used to treat such multivariate data. However, before applying the regression, various preprocessing techniques can be applied to the collected data. A single preprocessing method or several ones in series can be used. The usefulness of such an approach to acoustic spectrometry data is investigated in this thesis. An optimization algorithm of the type called Genetic algorithm was developed and used in order to choose a maximum of 6 preprocessing techniques in series out of a total of 31 available choices. The data used was acoustic spectrometry data on black liquor collected at the Billerud paper mill in 2010. The Genetic algorithm proved to perform well when solving the optimization problem. The results from the optimization show that it is possible to improve the prediction error for a given data set through the use of preprocessing, however, most of the selected preprocessing sequences did not improve the prediction error when using data collected at different times.

Keywords: Acoustic spectrometry, PLS, Partial Least Squares, preprocessing, Genetic algorithm, optimization

Sammanfattning

Akustisk spektrometri är ett snabbt sätt att mäta egenskaper hos vätskor inuti rör i processindustrin utan att göra ingrepp på ordinarie utrustning. Partial least squares regression är en populär metod inom kemometri som används för att behandla och analysera sådan multivariat data. Innan regressionen kan datan förbehandlas, antingen med hjälp av en ensam metod eller med hjälp av flera metoder applicerade efter varandra. Nyttan av sådana förbehandlingsmetoder applicerade på akustisk spektrometri-data undersöks i detta arbete. En optimeringsalgoritm av typen Genetisk algoritm har utvecklats och använts för att välja ut upp till 6 stycken förbehandlingsmetoder av 31 tillgängliga alternativ. Datan som användes var akustisk spektrometri-data som inhämtades på Billeruds pappersbruk under 2010. Den Genetiska algoritmen visade sig prestera väl när det kom till att lösa optimeringsproblemet. Resultaten från optimeringen visar att det är möjligt att förbättra prediktionsfelet för ett givet dataset men de flesta av de utvalda förbehandlingssekvensenerna gav ingen förbättring när data som var insamlad vid andra tidpunkter användes.

Nyckelord: Akustisk spektrometri, PLS, Partial Least Squares, förbehandling, Genetisk algoritm, optimering

Contents

Acronyms

DFT	Discrete Fourier Transform
DTFT	Discrete-Time Fourier Transform
FFT	Fast Fourier Transform
GA	Genetic Algorithm
KLT	Karhunen-Loève Transform
MA	Moving Average
MLR	Multiple Linear Regression
MSEP	Mean Squared Error of Prediction
PCA	Principal Component Analysis
PCR	Principal Component Regression
PLS	Partial Least Squares
PSD	Power Spectral Density
RMSEP	Root Mean Squared Error of Prediction
RSS	Residual Sum of Squares
SNV	Standard Normal Variate
SUS	Stochastic Universal Sampling
TSS	Total Sum of Squares
WSS	Wide Sense Stationary

Preface

This work was carried out from February to July 2011 at the Department of Signals and Systems at Chalmers University of Technology in Göteborg Sweden. The author is a student with a Bachelor's degree in Electronics engineering attending the Master's programme Communication engineering. The field of study during the education includes topics such as signal processing, multivariable calculus and linear algebra. A similar theoretical background is recommended for the reader.

The work was carried out with the aid and support of Acosense AB in Göteborg.

Aknowledgements

I would like to thank my supervisor at Acosense for his aid and support. I would also like to thank my supervisor at Chalmers for his valuable support and input throughout the course of the work.

Göteborg August 2011
Sebastian Johansson Mauricio

1 Introduction

This report describes a Master's thesis performed at the department of Signals and Systems at Chalmers University of Technology during the spring and summer of 2011. It was part of the company Acosense AB's continuing work on refining the quality of their process parameter measuring instrument.

1.1 Background

As more and more industries look to develop automated processes, the demand for fast, accurate and reliable monitoring equipment is increasing. In addition, many industries handle corrosive or in other ways aggressive liquids that might damage sensitive equipment. In view of this development, the possibility of measuring process parameters without stopping the process and without making contact with hazardous liquids is of great interest. Acoustic spectroscopy is one technique that could meet these requirements.

When manufacturing paper pulp from pulpwood, a substance called black liquor is produced and this substance is normally incinerated in a furnace. In order for the furnace to perform as efficiently as possible, it is important to know the percentage of dry matter content, the *total solids*, in the black liquor. Acoustic spectroscopy has recently been suggested as a method for measuring this quantity within tubes in processing plants without stopping the process itself [15]. Partial Least Squares (PLS) regression has successfully been used to analyze the spectrum of the collected data, however, the usefulness of preprocessing the data has yet to be evaluated.

Studies have suggested the possibility of combining PLS with different preprocessing techniques in order to improve the Root Mean Squared Error of Prediction (RMSEP) of the predictions as well as simplifying the PLS model [6]. Genetic algorithms have been used to find suitable combinations of preprocessing steps performed in series as well as selecting important frequency bands [6, 7].

Acosense AB develops and manufactures an instrument based on active acoustic spectroscopy called Acospector Acoustic Chemometer designed to measure and predict process fluids [1]. In their work to further develop and understand the processes, an investigation of preprocessing techniques was deemed valuable.

1.2 Purpose

This thesis aims to identify and investigate the impact of using a series of preprocessing steps on active acoustic spectroscopy data before performing PLS regression. An optimization algorithm will be developed and used to choose between the preprocessing methods since an exhaustive search of all combinations would be much too time consuming.

1.3 Objective

The objective is to improve the existing measurement procedure. An improvement could mean any of the following possibilities:

- Improvement of measurement accuracy

- Faster/more efficient measurement procedure (less computational time)
- More reliable system (more robust PLS model)

1.4 Limitations

The PLS algorithm itself will not be the focus of the study, ready-made functions in the MATLAB Signal Processing Toolbox will be used. The measurement setup will not be a part of the study. Different excitation sounds will not be investigated and the acoustics of liquids will not be considered. The preprocessing methods chosen in this study will be standard signal processing techniques, custom methods will not be designed. Due to time constraints, a complete investigation of the behaviour of the genetic algorithm will not be performed.

2 Theory

This section aims to provide a theoretical background to the concepts used in this thesis. To being with, acoustic spectrometry is introduced followed by a description of the Power Spectral Density (PSD). Multivariate analysis concepts used in the thesis is subsequently presented followed by the preprocessing methods and optimization with focus on genetic algorithms.

Throughout the thesis, scalars are denoted by italics (x) and column vectors are denoted by bold face ($\mathbf{x}$). Matrices are denoted by bold face and capital letters ($\mathbf{X}$) and transposed vectors (row vectors) and matrices are denoted by a prime sign ($\mathbf{x}'$ and $\mathbf{X}'$). An asterisk (*) denotes complex conjugate if nothing else is specified. $\mathscr{F}\{x(t)\}$ denotes the Fourier transform of $x(t)$ and $\mathscr{DF}\{x[n]\}$ denotes the discrete Fourier transform of $x[n]$. For more details on Fourier transforms, see appendix A.

2.1 Acoustic spectrometry

Acoustic spectrometry is the measurement of acoustic vibrations in order to obtain information about the system interacting with or emitting the vibrations [8]. Two techniques are commonly used, passive and active acoustic spectrometry.

In passive acoustic spectrometry, sensors listen to the vibrations produced by the system itself. When measuring process fluids, an obstruction can also be used in order to create additional turbulence in the liquid, generating more vibrations [15].

In active acoustic spectrometry, vibrations are sent into the system and sensors listen to the vibrations sent out from the system. The recorded signal contains vibrations originating from the system itself as well as those originating from the interaction between the system and the vibrations sent into the system [15].

2.2 Power spectral density

The PSD shows the distribution of power among the frequencies in a signal. The definition is given in equation 2.1, where ω is the angular frequency in radians per second, T is the duration of the truncated time domain signal and $X(\omega)$ is the Fourier transform of $x(t)$ (see appendix A) [10].

$$P(\omega) = \lim_{T\to\infty} \frac{1}{T}|X(\omega)|^2 \tag{2.1}$$

If the signal can be viewed as a Wide Sense Stationary (WSS) random process, the PSD can be calculated according to equation 2.2, where $R_{XX}(\tau)$ is the autocorrelation function of $x(t)$ (see appendix C) [9]:

$$P(\omega) = \mathscr{F}\{R_{XX}(\tau)\} \tag{2.2}$$

In other words, by estimating the autocorrelation function, it is possible to obtain an estimate of the PSD.

For discrete signals, an N-point sample based estimate $\hat{r}_{x,N}[p]$ of the discrete autocorrelation function $r_x[p]$ can be obtained using a truncated part of the original discrete signal $x[n]$

[10]:

$$\hat{r}_{x,N}[p] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[p+n], \quad p = 0, 1, \dots, N-1 \tag{2.3}$$

By using $\hat{r}_{x,N}[p]$ in equation 2.2, a sample based estimate of the PSD called a periodogram can be obtained. (The Discrete-Time Fourier Transform (DTFT) has to be used instead of the continuous Fourier transform.)

The periodogram is a smoothed version of the actual PSD. The smoothing results from using only N samples of the signal $x[n]$ in equation 2.3. This truncation can be achieved by multiplying $x[n]$ by a window function $w[n]$ and the choice of window function will affect the smoothing of the PSD. Simply choosing N samples of $x[n]$ is e.g. equivalent to applying a rectangular window. The relationship between multiplication in the temporal domain and smoothing by convolution in the frequency domain is further developed in appendix B.

If the Discrete Fourier Transform (DFT) is used instead of the DTFT when calculating the periodogram, a sampled version of the periodogram is obtained. This is the most common application since the rightmost part of equation 2.4, which describes this operation, can be efficiently calculated using the Fast Fourier Transform (FFT) [10]. $X[k]$ in equation 2.4 is the DFT of $x[n]$.

$$P[k] = \mathscr{DF}\{\hat{r}_{x,N}[p]\} = \frac{1}{N}|X[k]|^2 \tag{2.4}$$

In order to ascertain the average power of a signal, the autocorrelation function at lag zero, $R_{XX}(0)$, can be used. That value can be obtained by summing the values of the PSD and possibly multiply with a constant, (see also appendix A and C).

2.3 Multivariate analysis

Multivariate analysis is a type of analysis where many variables are observed and treated at a time. In this thesis, a power spectrum is used to characterize the properties of a liquid at a certain time instant. The spectrum contains several variables describing the energy content of the received signal at certain frequencies. In other words, for each time instant there are several variables describing the properties of the liquid and multivariate analysis can thus be used to analyse the spectra.

Principal Component Analysis (PCA), Principal Component Regression (PCR) and PLS are popular techniques used in multivariate analysis and they are described in this section. PCA is described more thoroughly since it is important for the understanding of the other two techniques.

2.3.1 Principal Component Analysis

PCA is a well known technique used in many different fields. In the engineering community it is perhaps more commonly known as the Karhunen-Loève Transform (KLT) [14]. The general idea of PCA is to represent a data matrix $\mathbf{X}$ using a new collection of basis vectors. The first basis vector in the new basis describes the direction in which the data has the largest variance. The second basis vector describes the direction in which the data has the second largest variance while being orthogonal to the first basis vector, and so on. In this new basis, if one wants to reduce the dimensionality of the data, removing basis vectors starting from the end will ensure the smallest possible deviation from the original data. The "principal component" itself is an infinite line through the origin along the direction of a basis vector, the first principal component lies in the direction of the first basis vector and so on.

A data matrix **X** of rank r whose rows can be viewed as points in space can be represented by r matrices of rank 1 [3] as shown in equation 2.5. The points, or rows of **X**, will be referred to as *observations* in the rest of this thesis. The reason for this name is that each row will be a power spectrum recorded at a certain time, thus constituting an observation of the characteristics of the liquid at a certain time.

$$\mathbf{X} = \mathbf{M_1} + \mathbf{M_2} + ... + \mathbf{M_r} \tag{2.5}$$

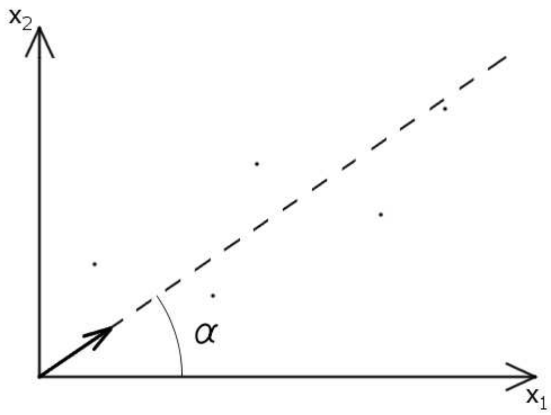
Each matrix $\mathbf{M_h}$ in 2.5 can be represented by a score vector $\mathbf{t_h}$ and a loading vector $\mathbf{p_h}$ [3]:

$$\mathbf{M_h} = \mathbf{t_h p'_h} \tag{2.6}$$

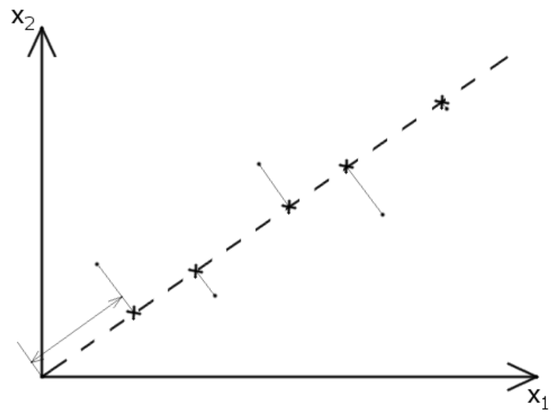
and the data matrix **X** can thus be written **X** = **TP'** by using equations 2.5 and 2.6. If all r components are not used, there is also a residual, **E**, consisting of a combination of the components not used [3]:

$$\mathbf{X} = \mathbf{TP'} + \mathbf{E} \tag{2.7}$$

$\mathbf{p'_h}$ is a unit vector in the direction of principal component h. Its elements are the projections of $\mathbf{p_h}$ onto the axes of the original coordinate system; $\mathbf{p'_h}(1) = \cos(\alpha)$ where α is the angle between component h and the first axis. This is illustrated in figure 2.1a. $\mathbf{t_h}$ is the data projected onto component h; $\mathbf{t_h}(1)$ is the distance from the origin to the first observation (point in space) projected onto component h as shown in figure 2.1b.



(a) A principal component as a dashed line, the loading $\mathbf{p_1}$ as a vector



(b) Data projected onto the first principal component, the double headed arrow shows $\mathbf{t_1}(1)$

Figure 2.1: Principal component analysis

In order to visualize the data, scores plots and loadings plots are often used [2]. Briefly, the scores for the observations (rows of **X**) for component i is plotted against the scores for component j. This helps determining correlations between the observations. The same can be done for the loadings; the loadings plot shows correlations between the variables in the observations [2, 13]. The variables are the elements of each row of **X**.

2.3.2 Principal Component Regression

PCR is very closely related to PCA and to linear regression. In the univariate case, a linear relationship between two variables can be described by the familiar formula $y = kx + m$. If there are many observations of x, each can be stored in a vector resulting in the following expression:

$$\mathbf{y} = k\mathbf{x} + m \tag{2.8}$$

If the observations of x are associated with errors (the observations might be flawed, the linear assumption itself might be erroneous), linear regression can be applied in order to "fit" the data to a linear relationship; the observations of x are projected onto a straight line in a two-dimensional space so that the error in some sense is minimized.

In the multivariate case, this notion is extended to multidimensional data. Instead of a vector of observations for the scalar x, there is a matrix $\mathbf{X}$, where each row is an observation of the vector **x'**. For a single observation, $y = b_1x_1 + b_2x_2 + \ldots + b_nx_n + e = \sum_{i=1}^{n} b_ix_i + e$, where x_i is the i^{th} entry of x' and this can be written as in equation 2.9.

$$y = \mathbf{x}'\mathbf{b} + e \tag{2.9}$$

If there are many observations, many vectors $\mathbf{x}'$, a relationship similar to the one in equation 2.8 can be obtained:

$$\mathbf{y} = \mathbf{X}\mathbf{b} + \mathbf{e} \tag{2.10}$$

It is also possible to have several y-variables in which case the more general expression becomes:

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E} \tag{2.11}$$

where $\mathbf{Y}$, $\mathbf{B}$ and $\mathbf{E}$ are also matrices. This is known as Multiple Linear Regression (MLR) [3]. If instead of using the actual data matrix $\mathbf{X}$, the scores $\mathbf{T}$ from a PCA analysis are used to represent $\mathbf{X}$, PCR can be performed [3]:

$$\mathbf{Y} = \mathbf{T}\mathbf{B} + \mathbf{E} \tag{2.12}$$

2.3.3 Partial Least Squares

Originally designed to be used in economics, PLS is now one of the most popular calibration methods in chemometrics [2, 3]. One of the benefits of using PLS instead of PCR or MLR is that it does not assume the dependant variables $\mathbf{Y}$ to be without noise [2]. The basic operation is explained below.

The $\mathbf{Y}$ matrix from equations 2.11 and 2.12 can be represented in the same way as the $\mathbf{X}$ matrix in equation 2.7 [3]:

$$\mathbf{Y} = \mathbf{U}\mathbf{Q}' + \tilde{\mathbf{F}} \tag{2.13}$$

where $\mathbf{U}$, $\mathbf{Q}$ and $\tilde{\mathbf{F}}$ are the scores, regression coefficients and the residual respectively. The tilde in $\tilde{\mathbf{F}}$ is used to distinguish it from the residual in equation 2.15. It can be shown that there is a connection, an inner relation, between the score vectors for $\mathbf{X}$ and the corresponding vectors for $\mathbf{Y}$ [3]:

$$\hat{\mathbf{u}}_{\mathbf{h}} = b_h\mathbf{t}_{\mathbf{h}} \tag{2.14}$$

where b_h describes the linear relationship between $\mathbf{u}_{\mathbf{h}}$ and $\mathbf{t}_{\mathbf{h}}$.

The idea with PLS, and where it differs from PCR, is to substitute $\mathbf{U}$ in 2.13 with $\hat{\mathbf{U}} = \mathbf{TB}$ from 2.14 in order to arrive at the final expression:

$$\mathbf{Y} = \mathbf{TBQ}' + \mathbf{F} \tag{2.15}$$

The same operation is made for $\mathbf{X}$, i.e. it is given the scores for $\mathbf{Y}$. By doing so, the covariance between $\mathbf{X}$ and $\mathbf{Y}$ is maximized rather than the variance of $\mathbf{X}$ and $\mathbf{Y}$ independently, as would be the case in PCR. The result is that the component, or *latent variable* following the PLS terminology, expressing the strongest tie between $\mathbf{X}$ and $\mathbf{Y}$ is selected [3].

When predicting y-values, the x- and y-loadings $\mathbf{p}'$ and $\mathbf{q}'$, along with $\mathbf{b}$ is calculated using some known x- and y-data. Then, new x-scores are calculated using new x-data without any known corresponding y-values. These scores are then used in equation 2.15 in order to predict the unknown y-values. The x-scores are calculated according to 2.16 where equation 2.7 has been used and where weights $\mathbf{w}'$ are used instead of the x-loadings $\mathbf{p}'$. The reason is that due to the interchange of scores between $\mathbf{X}$ and $\mathbf{Y}$ as described by equation 2.15, the $\mathbf{t}$ values are not orthogonal in PLS as opposed to the $\mathbf{t}$ values in PCR. They are normally orthogonalized by introducing weights which are used instead of $\mathbf{p}'$.

$$\hat{\mathbf{T}} = \mathbf{XW} \tag{2.16}$$

2.3.4 Measuring the quality of a model

Calculating the principal components in PCA or the latent variables in PLS and determining the relationship between $\mathbf{X}$ and $\mathbf{Y}$ is called building a model. Once a model has been built or *trained* using some data, there is often a need to evaluate that model according to some criterion. Two important measures are how well the model describes the data used to build it and how good the model is at predicting new values. A common way of describing how well a regression model fits the data is to use the R^2 measure as defined in equation 2.17.

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} \tag{2.17}$$

where RSS is the residual sum of squares and TSS is the total sum of squares:

$$\text{RSS} = \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.18}$$

$$\text{TSS} = \sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{2.19}$$

y_i is the ith observation of y and $\bar{y}$ is the mean of y. Provided a functioning model, the residual sum of squares is always smaller than the total sum of squares, so R^2 is bounded between zero and one. The closer R^2 is to one, the better the model fits the data.

The Q^2 measure is a variant of the R^2 measure which is commonly used to describe how well a model can predict new values.

$$Q^2 = 1 - \frac{\text{RSS}_{predicted}}{\text{TSS}_{actual}} \tag{2.20}$$

Although defined as R^2 in equation 2.17, Q^2 can take negative values since there is no guarantee that the residual sum of squares of the predicted values is smaller than the total sum of squares of the actual values. In fact, a model that is too specialized on the data used to create it, i.e. has a high R^2 value, might experience difficulties in predicting data; it loses its *generality*. This is known as over-training a model. Typically, R^2 will always increase with training, e.g. by using more latent variables in PLS, whereas Q^2 will increase to a certain point and then start to decrease indicating that the model is becoming too specialized and is losing its predictive capability. If the predicted values are simply the average of the actual values, equations 2.18 and 2.19 will be the same and Q^2 will be zero.

Other ways of measuring the quality of a model is to use the Mean Squared Error of Prediction (MSEP) or RMSEP which is the square root of the MSEP.

$$\text{MSEP} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.21}$$

The downside of using the MSEP or RMSEP is that they do not take into consideration the variance of the data which becomes important especially when using multiple validation or test sets, e.g. in cross-validation. Intuitively, if a data set contains a lot of changes, has a large variance, a larger error would be acceptable than for a data set in which the data hardly changes at all. Variance is further explained in section 2.4.2.

2.3.4.1 Validation set and test set A way of avoiding over-training of a model is to partition the data into a training set which is used to train the model, a validation set which is used to evaluate the predictive ability of the model during the training, and finally a test set which is used to evaluate the final model. Usually, one third of the total data is used as test set and one third of the remaining data is used as validation set. The rest is used as a training set.

2.3.4.2 Cross-validation Cross-validation is another way of avoiding over-training of a model. The initial data set is divided into K subsets and for each subset, a model is trained withholding the subset which is then used to evaluate the model.

2.4 Preprocessing

In this section, some of the preprocessing methods and the reasons for using them are presented. The choices of preprocessing methods were to a large degree influenced by the choices made by Jarvis and Goodacre in [6].

2.4.1 Savitzky-Golay smoothing and derivatives

Smoothing is a method used for reducing noise in data. A common way to smooth data is to use Moving Average (MA) filters but they have the disadvantage of using a straight line fit which risks deteriorating peaks in the data. An alternative is to use a polynomial fit where a curve is fitted to the data within a window as the window slides along the data. A simple and computationally efficient way of realizing this task involving simple summations was devised by Abraham Savitzky and Marcel J. E. Golay in 1964 [2]. Within the window, each datapoint is scaled with a predefined factor and then summed together. The summed up value is the fitted value for the datapoint in the middle of the window.

The filter is characterized by the window size and the order of the polynomial used for the fitting.

Derivatives can be an effective way of detecting overlapping peaks in data [2]. The problem is that they also amplify noise substantially. A way of counteracting this effect is to precede the differentiation by a smoothing filter. Since convolution has the associative property (see appendix B), the smoothing filter and the differentiation filter can be combined into one filter and then applied to the data.

2.4.2 Scaling

Scaling is a quite simple way of preprocessing data but the effect can be significant [2, 6]. Measures such as the mean and variance are often used. The mean is simply the average of the data; given a data sequence $x[n]$ stored in a vector $\mathbf{x}$, the mean can be calculated by summing all of the elements in $\mathbf{x}$ and dividing by the number of elements. $\mathbf{x}$ can e.g. be the samples of a signal or a spectrum. Mean centering a data series means subtracting the mean so that the resulting data only describes the variation around this average value. The expression for the mean is given in equation 2.22.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{2.22}$$

The variance gives a sense of the variability of the data, how much it is spread out around the mean. A large spread around the mean gives a large variance and vice versa. It can be calculated by summing the square of all elements in $\mathbf{x}$ after having removed the mean and then dividing by the number of elements minus one. Sometimes the division is done by exactly the number of elements, this however gives a biased estimate of the variance meaning that its expected value is not the true variance (see appendix C) but $\frac{N-1}{N}\sigma^2$, where σ^2 is the true variance. As can be seen, the difference decreases as N increases.

The square root of the variance is called the standard deviation. Dividing a data series with its standard deviation (after removing the mean) results in both the standard deviation and the variance being one. If this is done for two different data series, the effect is that the variations around the mean for both data series will be roughly on the same scale regardless of how they were related before. The expression for the variance is given in equation 2.23.

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2 \tag{2.23}$$

Often, mean centering a data series and dividing it by its standard deviation is referred to as autoscaling. Another commonly used term is Standard Normal Variate (SNV) transformation. In this thesis, the term SNV is used only to describe autoscaling of the observations (rows of a data matrix), autoscaling of the columns is referred to as autoscaling.

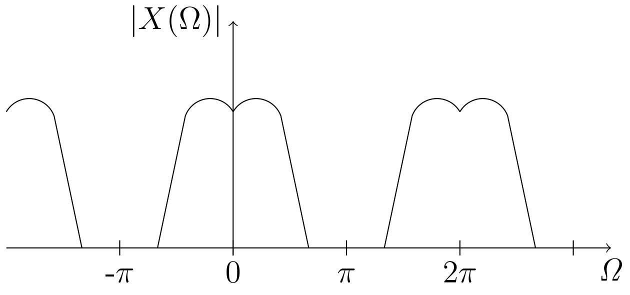
Another way of looking at a data series $\mathbf{x}$ with N elements is as a vector describing the coordinates of a point in an N-dimensional space. The length of this vector can be calculated according to equation 2.24. Dividing by $\mathbf{x}$ by its length makes it possible to compare the direction of different vectors without taking their lengths into consideration.

$$|\mathbf{x}| = \sqrt{\sum_{i=1}^{N} x_i^2} \tag{2.24}$$

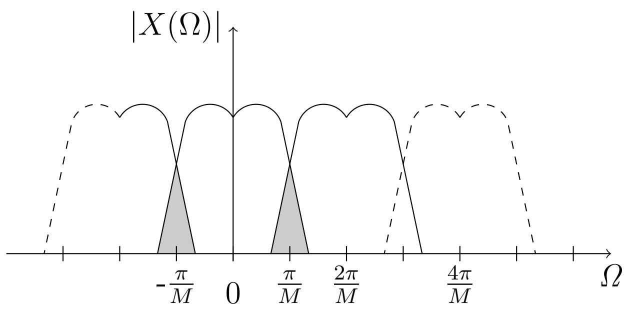
When different signals are compared, sometimes it is more informative to look at the distribution of power among the frequencies when the average power is the same in all signals. In order to achieve this, each sample of the PSD can be divided by the sum of all samples according to what is described in section 2.2. Any constants will be the same for all signals.

2.4.3 Decimation

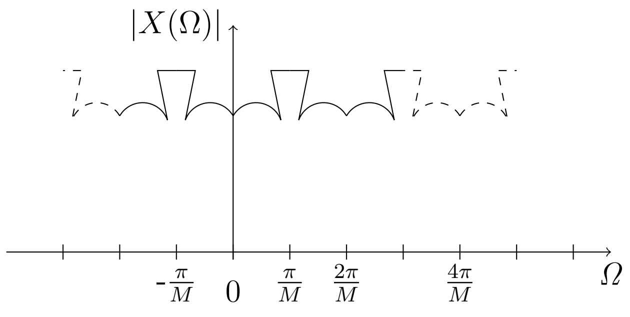
Decimation consists of low-pass-filtering data followed by a downsampling. The low-pass-filtering is performed in order to avoid aliasing of the signal when downsampling. Aliasing is when a signal is distorted due to the sampling frequency being to low. The effect in the frequency domain is illustrated in figure 2.2b. If the data is to be downsampled by M, each M datapoint is retained, the rest are discarded. In order to avoid aliasing of the signal, the cut-off frequency of the low-pass filter should be less than or equal to $\frac{\pi}{M}$. The reason for this is that after downsampling a time domain signal with M, the sampling frequency is lowered by M. Since the frequency content of the original signal has not changed, there will be an aliasing effect if there in the original signal was energy present above half of this new sampling frequency, as is illustrated in figure 2.2. Figure 2.2a displays the frequency spectrum of a signal before downsampling. The spectrum between 0 and π is mirrored around π and then repeated with the period 2π. Figure 2.2b displays the frequency spectrum after downsampling by M. The spectrum is now mirrored around $\frac{\pi}{M}$ so if there was energy above $\frac{\pi}{M}$ it will be added to the spectrum between $\frac{\pi}{M}$ and 0. These overlapping parts are shown in grey in figure 2.2b. The frequency spectrum is then repeated with the period $\frac{2\pi}{M}$. The final spectrum after adding the overlapping parts are shown in figure 2.2c. In the figure, $\Omega = \omega T_s$ where ω is the angular frequency measured in radians per second and T_s is the sampling interval. Consequently, 2π corresponds to the original sampling frequency. The effect of the different steps in decimating a signal is shown in figure 2.3 for a sample signal.



(a) Original spectrum

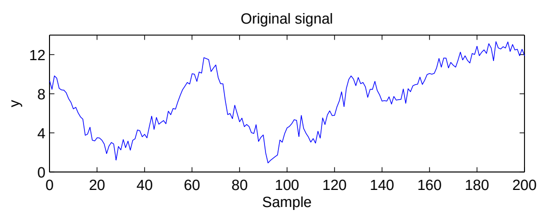


(b) Spectrum after downsampling with $M = 2$

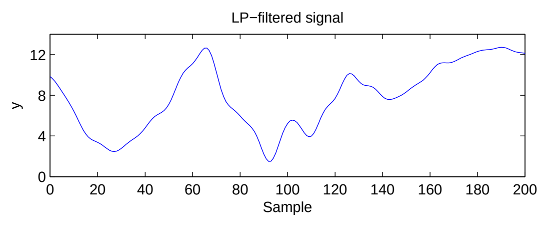


(c) Final spectrum of downsampled signal

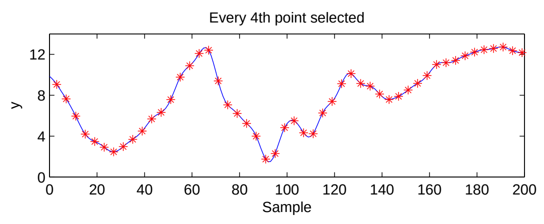
Figure 2.2: The effect of aliasing on a signal



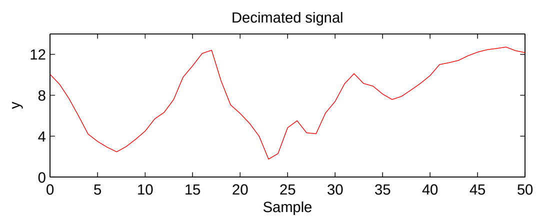
(a) Original signal



(b) LP-filtered signal



(c) Every M$^{\text{th}}$ point selected (M = 4)



(d) Decimated signal

Figure 2.3: The procedure of decimation illustrated in four steps

2.4.4 Wavelets

The discrete wavelet transform has its origins in many different fields but was unified into a single topic in the 1980s by Stéphane Mallat [16]. It has found applications in fields such as image and video compression, numerical analysis and object recognition and is the basis of the JPEG2000 standard. Although a mathematical tool, the operation of the transform can be described as a bank of filters as shown in figure 2.4 [16]. The signal $x[n]$ is fed as input to a lowpass (LP) and a highpass (HP) filter. The output from the lowpass filter is called an approximation of the signal $x_a[n]$ and the output from the highpass filter is called the details of the signal $x_d[n]$. In between the filters and the outputs, there is a downsampling stage in order to keep the total number of samples constant. The frequency content of the signal is kept since at the outputs of the filters, it is only half of that at the inputs for the same sample rate (see also section 2.4.3).

By feeding the average signal $x_a[n]$ into yet another highpass-lowpass filter pair, the average and detail of $x_a[n]$ is obtained. The procedure can then repeated for the new average signal obtained in each stage. For a fixed number of samples at the input, the procedure can be repeated until the final average signal contains only one sample which is related to the average of the original signal. The procedure is referred to as multiresolution analysis [16]. By discarding some of the details, compression (and smoothing) can be realized.

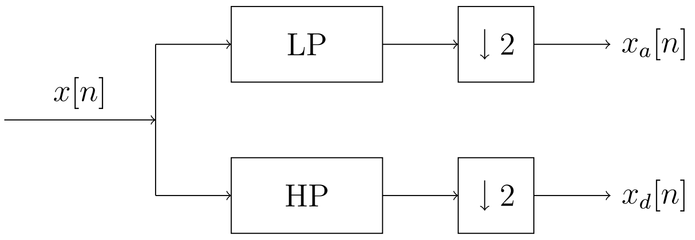


Figure 2.4: Wavelet transform as a bank of filters

2.5 Optimization

Haupt and Haupt describe optimization as “the process of adjusting the inputs to or characteristics of a device, mathematical process, or experiment to find the minimum or maximum output or result” [5]. An example of an optimization problem could be to adjust the antenna of a portable radio. The device would be the radio, the input to the device would be the antenna position and the output would be the sound quality. As the position of the antenna (the input) is changed, the sound quality (the output) changes. The goal or optimization problem would be to find the position of the antenna that produces the best sound quality. In order to find the best set, or at least a good set of inputs to a given experiment without trying all possible combinations, optimization algorithms are used. Typically, the inputs are adjusted according to some rule in order to “search” the cost surface (set of all possible output function values) for good solutions.

2.5.1 Genetic Algorithms

One such optimization technique is the Genetic Algorithm (GA). It was developed during the 1960s and ‘70s by John Holland. The name “genetic algorithm” originates from its inspiration from biology and natural selection. In short, a number of vectors are created with symbols from an alphabet A. Random changes will be done within each vector with a certain probability; in the binary case this translates to a certain probability of flipping a bit. Two vectors can also be combined, e.g. by substituting one half of the content in each vector thus generating two new ones. After each random change and/or recombination, the resulting new vectors are evaluated against a cost function in order to determine their individual suitability. Thereafter a new set is created where each vector is selected from the old set, including the newly generated vectors, according to some scheme taking into consideration their individual suitability [12].

Drawing from its biological inspiration, the vectors are called chromosomes, each position within the chromosomes is called a gene, the values that the symbols can take are called alleles and the set of chromosomes is called a population. The random changes in a chromosome are called mutations and the evaluation and deletion according to the cost function is called selection. The cost function itself is called fitness function referring to the evaluation of the fitness of each individual, whose traits are determined by the chromosome. The process of selection, recombination and mutation is called reproduction, and each population is called a generation [12].

2.5.1.1 Population size It is difficult to make statements about the parameters associated with a genetic algorithm as the settings seem to be very problem specific, as is described in by Reeves and Rowe in [12]. It is however suggested in the same literature that the minimum size of the population is chosen in such a way that the entire search space can be reached from the initial population by crossover only. This means that an instance of each allele should be present at all gene locations somewhere within the population. That is to say, when looking at a certain position (gene) using all the chromosomes within the population, one should be able to find at least one of each of all possible alleles.

For binary chromosomes of length l in a population of N individuals (chromosomes), the probability of being able to reach the entire search space is [12]

$$P = \left(1 - (1/2)^{N-1}\right)^{l} \tag{2.25}$$

For a q-ary alphabet, q being the number of letters in the alphabet, this probability becomes [11]

$$P = \left\{ \frac{q!S(N,q)}{q^N} \right\}^{l} \tag{2.26}$$

where $S(N,q)$ is the Stirling number of the second kind which can be calculated according to equation 2.27 using the initial conditions in equations 2.28, 2.29 and 2.30 [4].

$$S(N,q) = S(N-1,q-1) + qS(N-1,q) \tag{2.27}$$

$$S(n,n) = S(n,1) = 1, \quad n \geq 1 \tag{2.28}$$

$$S(n,0) = \begin{cases} 1 & \text{if } n = 0 \\ 0 & \text{if } n > 0 \end{cases} \tag{2.29}$$

$$S(n,k) = 0, \quad \text{if } k < 0 \text{ or } k > n \tag{2.30}$$

3 Method

In this section, the implementation of the project is described. To begin with, the initial literature study leading up to the choice of method is briefly described followed by a description of how the data was collected. Then the process of constructing data sets is described followed by a description of how the preprocessing methods were implemented. The last section covers the implementation of the genetic algorithm. The MATLAB version used in this project was 7.12.0.635 (R2011a), 64 bit. The computer used for the MATLAB calculations was a 64 bit Windows 7 computer equipped with dual Intel® Xeon® processors at 2.40 GHz and 8 GB of RAM. Some of the multivariate analysis was performed using the multivariate analysis software Simca-P+ 12.

3.1 Literature study

The literature study undertaken at the beginning of the project aimed at finding similar investigations that had been undertaken previously and if found, to assess if the same ideas could be applied in this project. The literature study was performed using resources available through the Chalmers Library such as books and scientific databases, most notably Scopus, a large database containing publications from many different disciplines, and also by using standard internet search engines. It turned out that there were quite a few studies regarding preprocessing methods for PLS regression, however, not many treated acoustic spectrometry data. The method of using a genetic algorithm to select a sequence of preprocessing methods as is done by Jarvis and Goodacre in [6] was deemed applicable for this project as the problem was the same but the type of data was different. Another factor that weighed in when selecting a genetic algorithm as optimization method was the concern that the cost surface would be rather complex when evaluating many different types of preprocessing methods. This could make conventional optimization techniques difficult to use [5].

3.2 Measurement setup

The measurement equipment used to record the data at the Billerud Gruvön paper mill is described below. It was already installed at the mill and the design of this system is not a part of this thesis.

The system consisted of a shaker and two accelerometers fastened on a tube containing black liquor just after the evaporation stage at the mill. The shaker consisted of a piezoelectric disk between two copperplates connected to a voltage source. Outside the copperplates there were two isolating distances. The shaker was attached to the tube with a clamp. The two accelerometers were glued directly to the opposite side of the pipe. The inner diameter of the pipe was 350 mm.

The computer responsible for collecting the data and controlling the shakers was placed adjacent to the pipe. Data acquisition cards from National Instruments were used in order to send and receive signals. These cards and other equipment is listed below.

- DAQ NI PCI-4461
- DAQ NI PCI-6351

- HP Z 600 Work station
- Western Digital MyBook Elite external hard disk drives
- Four Brüel&Kjær DeltaTron 4396 accelerometers
- Custom signal amplifier from 41 Hz Audio
- Two Brüel&Kjær 2647-A Charge-to-DeltaTron® Converters
- Shakers constructed from piezoelectric disks from Piezomechanik GmbH

The 6351 card was used as signal generator and was connected to the shaker through a custom amplifier made by 41 Hz Audio. The accelerometers were connected to the computer via the 4461 card through the Charge-to-DeltaTron® converter. Data acquisition, signal generation and calculation of PSD spectra was performed using National Instruments' software LabVIEW™. The sampling rate used was 102,400 samples per second. 8192 samples were used in order to calculate each PSD which consequently contained 4096 samples. Each final PSD was obtained from averaging 100 subsequent PSD's using an exponential moving average. Only data from one of the accelerometers was used in this thesis.

3.3 Data sets

The data used in this thesis comes from Billerud Paper mill in Gruvön and was collected during 2010 and 2011 using the methods and equipment described in section 3.2. Each power spectrum corresponds to a certain time and several spectra were stored as rows in a matrix. Following the terminology introduced in section 2.3.1, these spectra will be referred to as observations of a variable $\mathbf{x}'$. The y-data is total solids in percent. Two different data sets were used for the optimization, one spanning a long time period with low resolution (few observations) and the other spanning a shorter period in time but with higher resolution.

3.3.1 Low resolution data set

This data set was constructed using 73 observations taken during a rather large time period spanning most of 2010 and early 2011. The observations were unevenly distributed throughout this period as can be seen in figure 3.1 and this resulted in a very low resolution over time although it at certain narrow time intervals was good, at the end of March e.g.

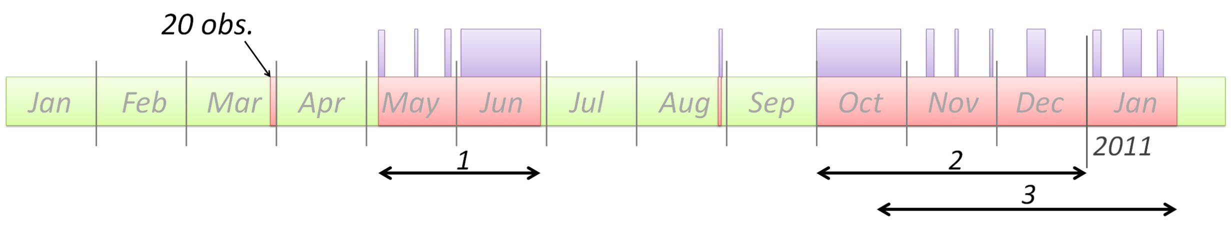


Figure 3.1: Red sections show periods covered by observations, purple sections show the distribution of observations within red sections.

Three validation sets were created. As can be seen by the horizontal arrows at the bottom of figure 3.1, they overlap somewhat and together they do not cover all observations. The first two were selected in order to be easier to predict since they are in the middle of all observations. They also represent fairly continuous series of observations. The last one was selected to be the most difficult one since it is at the very end of all observations.

3.3.1.1 Decimated data set

A variant of the low resolution data set was created by performing decimation on each spectrum. This was done in MATLAB using the parameters described in section 3.4.3. The resulting data set contained 1024 variables for each of the same observations as in the original low resolution data set. The same validation sets were used.

3.3.2 High resolution data set

This data set included observations collected in February 2011. Originally, observations were taken one minute apart covering the whole month of February. This made the data set very large and time consuming to work with. There were also other issues such as missing and faulty y-values. The preparations needed to overcome these issues and construct a workable data set are described below. All preparations were done using MATLAB except the PCA analysis which was done using Simca-P+.

During certain times the system in the paper mill is flushed in order to clean out deposits in the pipes. During these flushings, and during other stops in production, the recorded values for the total solids dropped quite drastically. However, the refractometer used for the measurements had a threshold and showed all values below 50 as being 50. The phenomenon is shown in figure 3.2. Since the y-values were faulty during these times, all observations for which the y-values were below 50.5 were deleted. Some y-values were also missing completely and the corresponding observations for these times were deleted as well. In total, 808 deletions were made as a result of these criteria. One additional observation was deleted after being judged as being an outlier in the PCA analysis as seen in figure 4.5. Within the observations there were also 7 missing values (within individual spectra). These values were interpolated using the average of the two adjacent frequency bins.

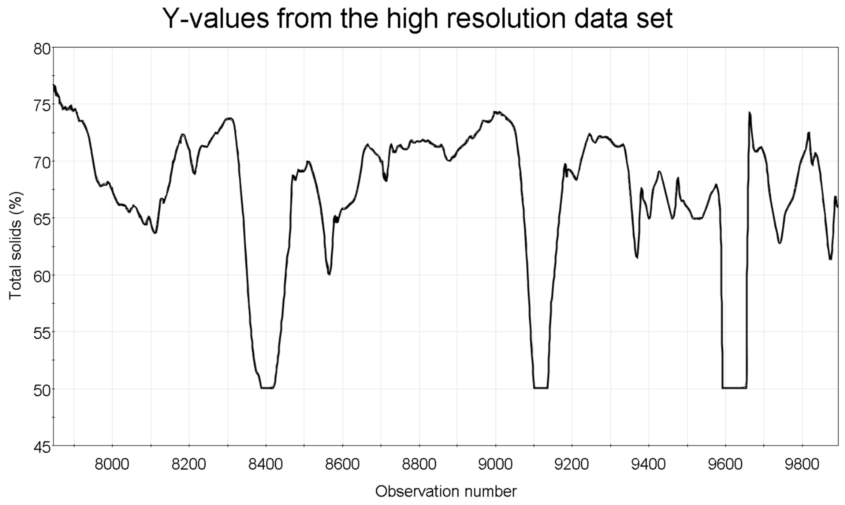


Figure 3.2: Example of y-values not going below 50

In order to reduce the computation time when working with the data set, a smaller subset was used. One week of continuous data was chosen starting the 2nd of February 00:00 and ending on the 8th of February 23:59. In order to further decrease the amount of data, not all observations were used. The choice of which observations to use was made by analyzing how much the y-values changed between samples. If the cumulative change in absolute values between data points in **y** did not exceed a threshold, every 5th observation was used, i.e. one observation every five minutes. If the cumulative change was above the threshold, all 5 observations were used. In order to avoid small fluctuations in the y-values influencing the decision, a polynomial fit was used on the y-values before making the decision. This however modelled the sharp peaks resulting from flushings in the system poorly, so in order to keep the peaks, it was decided to switch from the fitted y-curve to the actual one if the difference between them grew to big. The final threshold for switching between the fitted values and the actual ones was determined by visually inspecting the result of a number of trial thresholds. The threshold that applied to the cumulative change in y-values was also determined visually through trials by verifying that important traits were not lost when choosing a lower sampling rate. The fitting of the y-values is shown in figure 3.3 where a switch from the smoothed curve to the actual one and back can be seen. The sampling of the edited data is shown in figure 3.4 for the same observations and it can be seen that the desired effect is achieved where all data points are used for the sharp drop, but only every fifth one where the curve changes only a little between data points. It was decided to not use less than one observation every five minutes since even though the y-values might have allowed for it at certain intervals, the x-observations might contain errors as well.

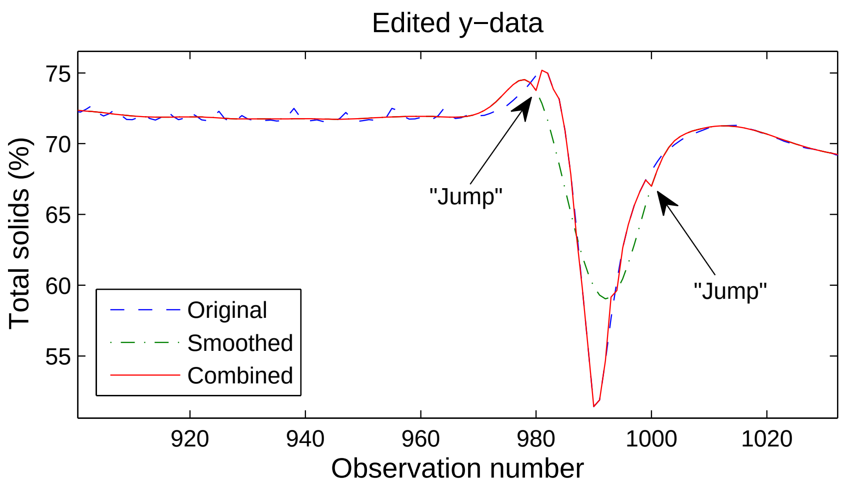


Figure 3.3: Smoothing of the y-data, zoomed in view

The data was then divided into one work set and one test set containing 2/3 and 1/3 of the observations respectively. The work set was then further partitioned into a training set and a validation set containing 2/3 and 1/3 of the work set respectively. The validation set was used for testing the PLS-model during the optimization and determine the fitness of a certain individual. The test set was used to evaluate the results of the optimization after it was completed. The data set partitions are shown in figure 3.5 using the y-values.

3.3.2.1 Wavelet coefficient data set A wavelet coefficient data set was also constructed using a discrete wavelet transformation on observations in the high resolution data set after the data reduction described above had been performed. The method is described in section 3.4.4. The coarsest detail level retained was level 4 containing 512 samples out of an original 4096. The percentage of sum of squares of the original spectra retained was 97 %. Although the reduction of variables would have allowed for an increase

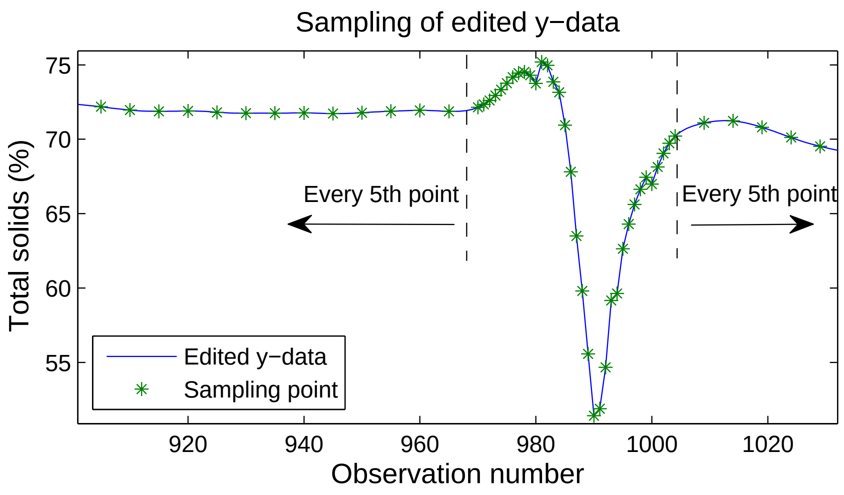


Figure 3.4: Sampling of the edited y-data

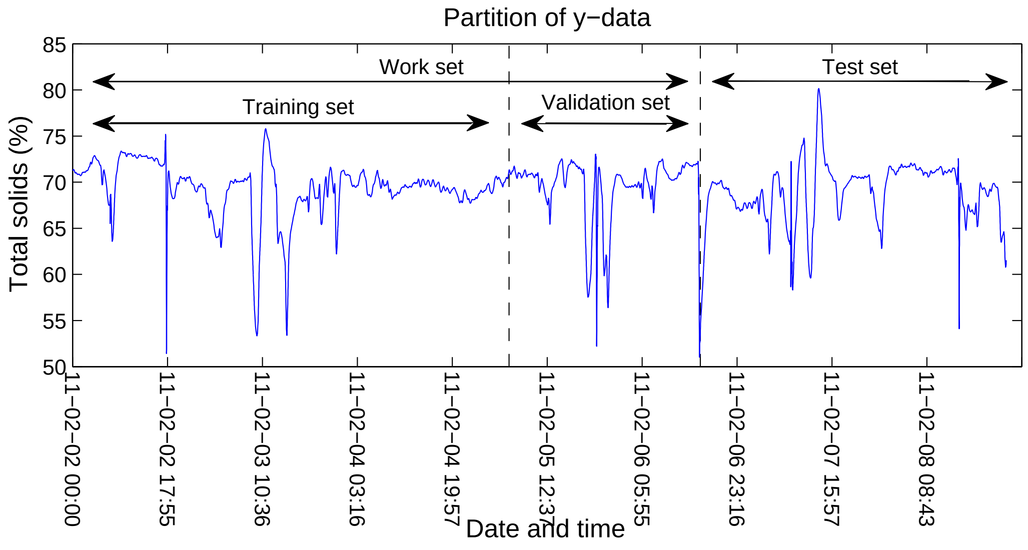


Figure 3.5: Data set partitions showing the y-values

in the number of observations in comparison with the original high resolution data set, the same observations were kept in order for the results to be comparable.

3.3.3 Summary - data sets

The four data sets are summarized below in table 3.1.

Table 3.1: Summary of the different data sets

Data set	Collected	Obs.	Sampl/obs.	Val. sets	Test sets
Low res.	Feb 2010 - Jan 2011	73	4096	3	0
Low res., decim.	Feb 2010 - Jan 2011	73	1024	3	0
High res.	Feb 2011	2553	4096	1	1
High res., wav.	Feb 2011	2553	512	1	1

3.4 Preprocessing

In this section, implementation aspects of some of the preprocessing methods are presented. All methods are summarized in table 3.2. With the exception of wavelet transformations which were performed using Simca-P+ 12, all preprocessing was done in MATLAB.

3.4.1 Savitzky-Golay smoothing and derivatives

In order to apply the Savitzky-Golay smoothing, `sgolayfilt` in the MATLAB Signal Processing Toolbox was used. Savitzky-Golay derivatives were applied using the `sgolay` command in the same toolbox. The input signal was shortened by $W-1$ samples after taking the derivative, where W is the window size of the filter, since the first and last $(W-1)/2$ samples contained transients after the filtering. W is always odd in Savitzky-Golay smoothing.

3.4.2 Scaling

When performing the column based operations such as column centering, a specific variable was isolated and treated as the samples of a signal in time, across the observations. An issue with these methods is that during operation in a real life scenario, predictions are made using a single observation where no column (variable) averages exist. One option is to use a fixed average, another option could be to recalculate statistics such as the average and variance as new observations arrive, continuously or at given intervals. For simplicity, the first choice with a fixed average and variance was used in this work.

That meant that no observations from the validation or test sets were allowed to influence column operations. As described above, they represented future observations and thus could not be allowed to influence the construction of the model which was going to be used for prediction. As a result, when performing e.g. column centering as preprocessing, the average of the training set variables had to be subtracted from the validation or test set variables. The same principle applied in the actual PLS-regression stage, where the average of the training variables had to be used when centering the validation and test set variables.

When scaling the average power to be equal for all spectra, the values had to be transformed to linear scale first since they were stored in decibel units. After the transformation and normalization, the values were recalculated and stored in decibel units again.

3.4.3 Decimation

Early results from the low resolution data set suggested that smoothing might be an effective means of preprocessing the data. Also, using Simca-P+, it was possible to ascertain that most of the variance within the data set was located within the slow variations of the spectra. It is important to note that this does not mean that it was located at low frequencies since the data was already in the frequency domain. It did however suggest that it might be possible to represent most of the differentiating features in the data using fewer samples for each individual observation (spectrum). The resulting spectra would contain fewer frequency bins spaced further apart from one another in frequency. A way of realizing this would be to simply use every M sample in each spectrum, but that would not lower the computational burden when actually constructing the spectra. Another way of lowering the resolution in the frequency domain would be to use a smaller temporal window when constructing the spectra. As this would decrease the computational load it was judged interesting enough to investigate.

Since constructing new spectra from the time domain data would be a very time consuming task (due to the way this data was stored), it was decided that the effect of using a smaller window would be simulated by using decimation on the existing spectra. This will not perfectly simulate the effect of using a smaller window size in the time domain since changing the window size will not only change the resolution in the frequency domain, but also change the windowing effect. Some further descriptions of time domain and frequency domain operations are given in appendix B. The objective however was only to determine if using different window sizes would be worth investigating or not, and for this reason, an exact simulation of the effect was not necessary.

By visually inspecting the frequency content of the spectra, a cut-off frequency of 0.1 was decided upon, where the Nyqvist frequency is 0.5.[1] Since M was restricted to be a power of 2 in order to maintain the window size a power of 2 when constructing the spectra, this allowed for a downsampling of 4 (see section 2.4.3). It was important to keep the window size since the FFT was used for when constructing the spectra as described in section 2.2. The type of filter used was a 66th order linear-phase finite impulse response filter designed using the `firpm` command in the MATLAB Signal Processing Toolbox.

3.4.4 Wavelets

The wavelet transformations were performed using Simca-P+ 12. The type of wavelet used was Symlet of order 10. The choice was made since this was the type of wavelet currently in use by Acosense. The first wavelet coefficient is as discussed in section 2.4.4 related to the average of the original signal. As a result, this coefficient was very different in size than the other ones. Since this would affect most row operations negatively when preprocessing was applied, the first coefficient was excluded from row operations unless the variables, being the columns of the data matrix, were autoscaled first. Two versions of autoscaling was made available, the ordinary one plus a new one that would allow subsequent row operations to include the first wavelet coefficient. Since the observations were no longer PSD spectra in decibel units but wavelet coefficients, preprocessing *S6*, scale average received power to 1, would not perform its intended task. Therefore it was substituted for the new autoscaling method when wavelet coefficient data was used, as seen in table 3.2.

3.4.5 Summary - preprocessing

In total, 31 choices of preprocessing methods were made available: 7 scaling methods, 14 pure Savitzky-Golay smoothing and 10 Savitzky-Golay smoothing with derivatives. All methods are summarized in table 3.2 which also explains the codes used to describe the preprocessing sequences in the results.

[1]The "sampling frequency" was in samples/Hertz since the "signals" were in the frequency domain

Table 3.2: Preprocessing methods and their encoding

Category	Enc. no.[a]	Code[b]	Preprocessing	Attributes
-	0	-	No further steps	-
Scaling	1	S1	Column centering	-
	2	S2	Row centering	-
	3	S3	Column variance to 1	-
	4	S4	Column autoscale	-
	5	S5	Row lengths to 1	-
	6	S6	Scale power to 1[c]	-
	6[d]	S6w	Row variance to 1	-
	7	S7	SNV[e] transformation	-
S-G smoothing	8	G1	Smoothing using a Savitzky-Golay smoothing filter	W[f] = 7, O[g] = 3
	9	G2		W = 11, O = 3
	10	G3		W = 15, O = 3
	11	G4		W = 19, O = 3
	12	G5		W = 23, O = 3
	13	G6		W = 33, O = 3
	14	G7		W = 43, O = 3
	15	G8		W = 7, O = 4
	16	G9		W = 11, O = 4
	17	G10		W = 15, O = 4
	18	G11		W = 19, O = 4
	19	G12		W = 23, O = 4
	20	G13		W = 33, O = 4
	21	G14		W = 43, O = 4
S-G differentiation (1st order)	22	D1	1st order derivatives after smoothing using a Savitzky-Golay smoothing filter	W[f] = 15, O[g] = 3
	23	D2		W = 21, O = 3
	24	D3		W = 27, O = 3
	25	D4		W = 37, O = 3
	26	D5		W = 47, O = 3
	27	D6		W = 15, O = 4
	28	D7		W = 21, O = 4
	29	D8		W = 27, O = 4
	30	D9		W = 37, O = 4
	31	D10		W = 47, O = 4

[a] Encoding number used for the optimization algorithm (translated to decimal numbers)
[b] Used in the Results section
[c] Only scales average power to 1 when selected as the first preprocessing method
[d] Used on wavelet coefficient data set, on other data sets S6 was used
[e] Standard Normal Variate transformation
[f] Window size of filter
[g] Order of polynomial

3.5 Genetic Algorithm

In this section, the implementation of the GA is described. Everything was done using MATLAB. The Genetic Algorithm in the MATLAB Global Optimization Toolbox was not used since it was not available to the author.

The final algorithm allowed for quite a lot of variables to be set such as the population size, crossover rate, mutation rate, elite count etc. In addition, different types of crossover, mutation and selection functions could be specified. Thoroughly investigating the effect of all of these parameters and functions for this particular application would have been quite time consuming and was also regarded as being outside the scope of the project. Consequently, it is quite possible that a better combination of parameters than that which is used in this project exists, but no further work was done to investigate this.

3.5.1 Pseudocode

The pseudocode for the GA is presented below. Here, it realizes a minimization; the smaller the fitness score, the better.

Algorithm 1 Pseudocode for the Genetic Algorithm

```
popSize = K;
create initial population
evaluate initial population
currentGen = 0;
maxGen = N;
stallGen = 0;
stallGenMax = M;
tolerance = t;
improvement = 0;
prevBestScore = "large";
while currentGen < maxGen AND stallGen < stallGenMax do
  select subset of population for crossover
  recombine selected individuals
  mutate parents and offspring, elite individuals excluded
  evaluate entire population
  keep the K most fit individuals
  improvement += prevBestScore - currentBestScore;
  prevBestScore = currentBestScore;
  if improvement < tolerance then
    stallGen++;
  else
    improvement = 0;
    stallGen = 0;
  end if
end while
```

3.5.2 The chromosomes

The chromosomes consisted of preprocessing methods in the order they were to be applied to the data. To code the preprocessing methods, each was assigned a number and a sequence of these numbers in their binary form formed a chromosome, as is shown in figure 3.6. The decimal number zero was used as a stop character so that all steps after the zero were neglected. This meant that any number of preprocessing steps up to a maximum number, which could be set as a variable in the code, could be chosen by the algorithm.

The chromosomes were stored as rows in a matrix, as is shown in figure 3.7, which made up the population.

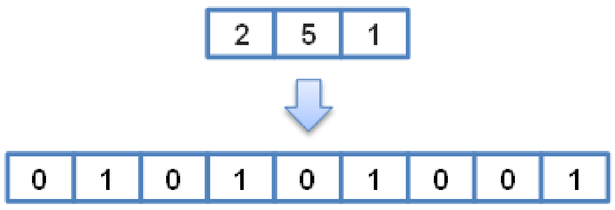


Figure 3.6: A sequence of preprocessing steps stored in a binary vector (3 bits per decimal number)

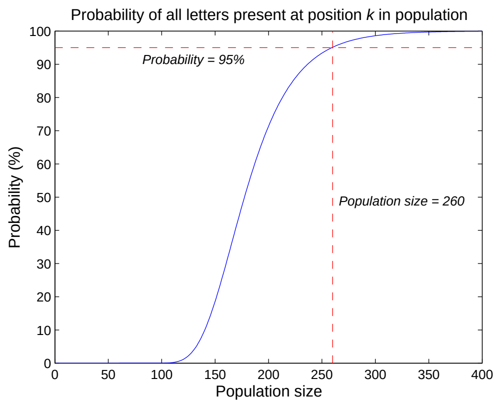
1
2
3
4
5
6
7
8
9
10

Figure 3.7: A population of ten individuals, each individual is a binary vector as in fig. 3.6

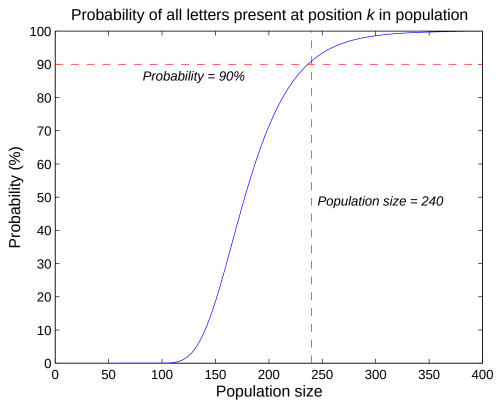
3.5.3 The population

The initial population was created randomly as a default but it was also possible to choose all or some of the individuals in the population. This possibility however was used very conservatively, as there are cautions against using this technique in both [5] and [12]. Care had to be taken since the random number generator in MATLAB is initialized to the same state every time MATLAB starts up. This means that the `rand` command, used for generating all random numbers in the algorithm (for initial population, selection, crossover points, mutation. . .), will generate the same sequence of numbers in each session if the state is not changed. This will cause the genetic algorithm to produce the same solution each time a session is initiated. This was avoided by generating different number of random numbers at the start of each session before the genetic algorithm was initiated. The sequence of numbers was still the same but different parts of the sequence were used in different places of the genetic algorithm which meant that each solution was unique and not just a copy of a previous result.

In order to get some guidance as to the size of the population, equation 2.26 had to be used instead of 2.25. The reason was that the crossover points were limited to locations between two numbers in their *decimal* form making the number of possible "letters" 32. The reasons behind this limitation to the crossover point locations will be explained in section 3.5.4. This however meant that the population size had to be increased considerably when compared to the purely binary case. A graph showing the probability of covering the entire search space from the initial population as a function of population size is shown in figure 3.8. In the graph, the chromosome length (the maximum number of preprocessing steps), l, is 6. The number of letters in the alphabet, q, is 32. This was the setup used for the final tests. As can be seen, an initial population of size 260 would give a 95% probability of covering the entire search space using only crossover (figure 3.8a) and this was used for the low resolution data set. A population size of 240 would give slightly more than a 90% probability (figure 3.8b) and this was used for the high resolution data set due to the large size and long computation time for that data set.



(a) Population size of 260 and 95 % prob. marked as dashed red lines



(b) Population size of 240 and 90 % prob. marked as dashed red lines

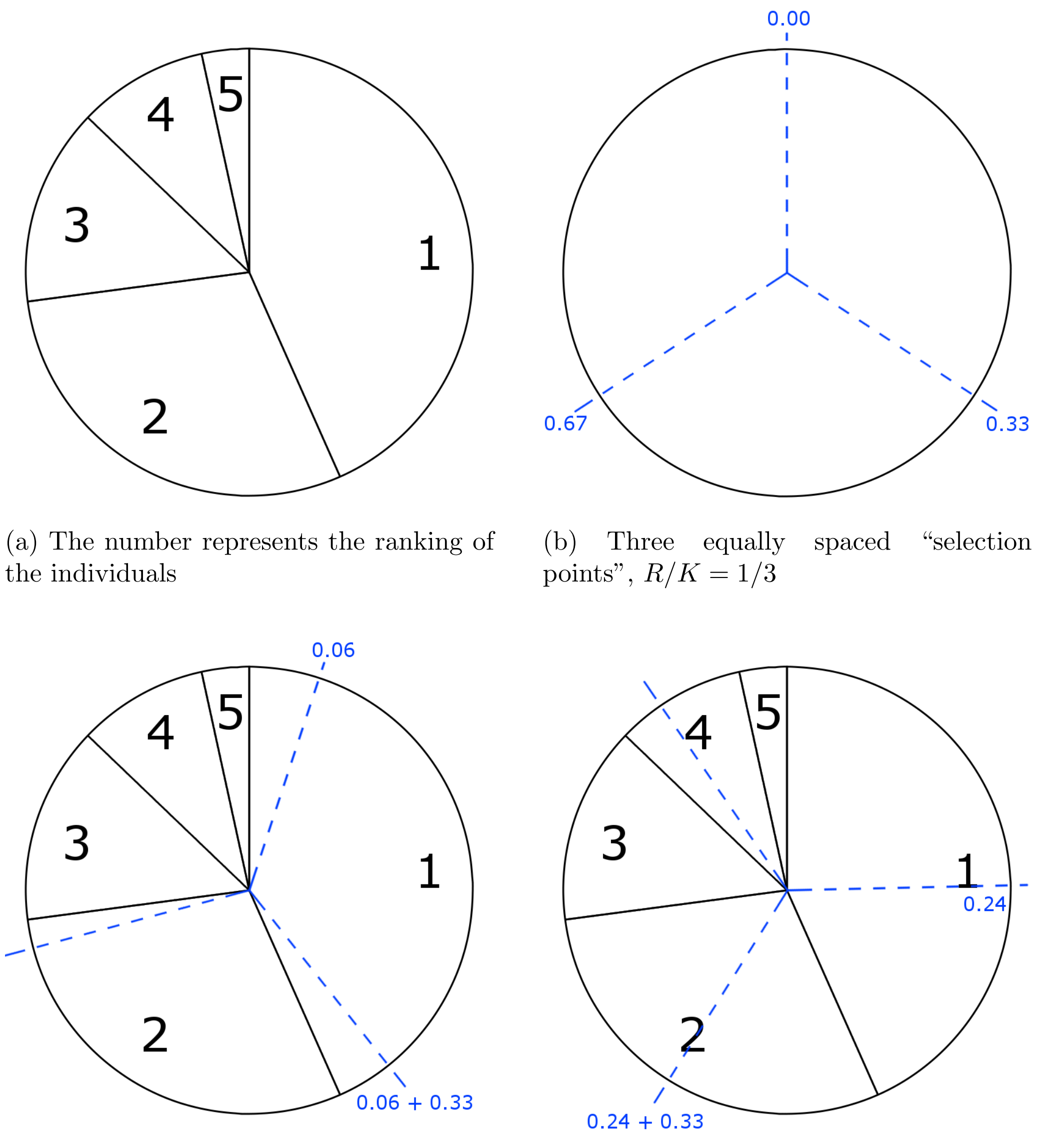
Figure 3.8: Probability of reaching the entire search space from the initial population using only crossover

3.5.4 Crossover

In order to select individuals for crossover, a crossover rate was set. This was the ratio of the population used for crossover. Once set, Stochastic Universal Sampling (SUS) was used to select the individuals. The same technique is used by Jarvis and Goodacre in [6], and it is also what is recommended in by Reeves and Rowe in [12]. It should however be noted that other possibilities such as roulette wheel and tournament selection exist [5, 12], but they were not investigated.

The process of SUS is illustrated in fig 3.9. All individuals in the population were evaluated for their fitness and given a ranking score. For simplicity, the ranking score was simply the inverse of the fitness score. A low fitness score meant a good preprocessing sequence as will be described in section 3.5.6. The only restriction was that all fitness scores had to be positive. Using the fitness score to calculate the ranking score is known as cost ranking. The individuals were then sorted according to their ranking score, from highest to lowest. This is shown in figure 3.9a where the circle represent the sum of the ranking scores, R, and the size of each sector in the circle represents the ranking score of an isolated individual. The boundaries between sectors are the cumulative ranking scores, i.e. boundary two is located at ranking score 1 plus ranking score 2.

Next, the first individual for crossover was to be selected. The first step in doing so was to divide the cumulative ranking score R for all individuals by the number of individuals to be selected K. Figure 3.9b shows this when R is 1 and K is 3. The next step in choosing the first individual was to generate a starting point, a random number within the range from zero to R/K. The individual corresponding to the cumulative ranking score indicated by the starting point was then selected. If e.g. the starting point were to end up between zero and the first circle sector boundary in figure 3.9a, individual 1 would be selected. The second individual was chosen by adding R/K to the first random number, the third by adding $2R/K$ and so on. The operation is equivalent to turning the pointers in figure 3.9b by the amount determined by the starting point. This is illustrated for two different starting points in figures 3.9c and 3.9d.



(a) The number represents the ranking of the individuals

(b) Three equally spaced "selection points", $R/K = 1/3$

(c) Starting point: 0.06, selection outcome: 1 1 2

(d) Starting point: 0.24, selection outcome: 1 2 4

Figure 3.9: Illustration of Stochastic Universal Sampling

For the actual crossover, single point crossover as illustrated in fig. 3.10 was used. The choice was made primarily due to its simplicity and the fact that it is easy to understand what is happening. When using single point crossover, care had to be taken to make sure that the crossover point was always located between two groups of bits representing a decimal number, otherwise new preprocessing methods not belonging to either of the parents would have been introduced.

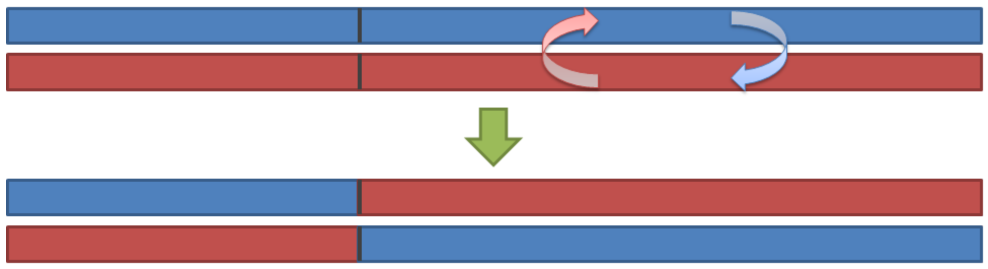


Figure 3.10: The process of single point crossover; the top two chromosomes are the "parents" and the bottom two their "offspring".

3.5.5 Mutation

After the crossover, all individuals, parents and offspring alike, except a certain number of *elite individuals* were mutated. The elite individuals were the best parents, i.e. the best individuals from the previous generation. Since they were the best ones, these individuals were allowed to pass unaffected through the mutation phase. The number of individuals used as elite individuals could be set, but two was used as a standard.

Since the chromosomes were binary, a mutation was simply the flipping of a single bit. This was done with a certain probability, the mutation rate. The mutation rate was set globally for the whole set of individuals subjected to mutation, meaning that the individuals subjected to mutation was seen as a binary matrix, and the mutation rate applied to the matrix as a whole. As a consequence, it was not possible to explicitly select the ratio of individuals that were going to be mutated. However, given a certain mutation rate, the average probability of flipping a certain number of bits per individual could be ascertained through simulations. The result is shown in figure 3.11 for the input mutation rate used in the final tests. As long as the probability of not flipping any bits was not too small, there would be individuals unaffected by the mutation. This was important since having too much mutations would undo much of the results from the crossover phase. After mutation, all individuals were ranked and the least fit ones were discarded in order to keep a constant population size. In general for genetic algorithms, the process of crossover will cause the

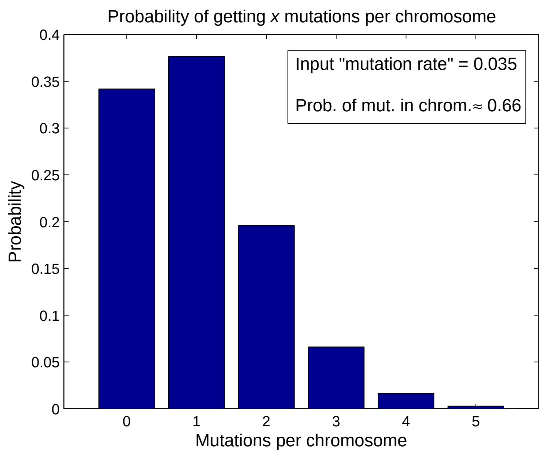


Figure 3.11: The average probability of getting different number of mutations per individual

algorithm to converge on a solution (possibly suboptimal), and the mutation counteracts this convergence by introducing an additional degree of randomness. The successfulness of the optimization depended to a large extent on finding a suitable combination of the parameters associated with these two processes, and this proved to involve some trial and error.

3.5.6 Fitness function

The fitness of a certain individual was based on the predictive ability of the PLS model built on the data after preprocessing. The preprocessing methods specified in the chromosome were performed on the data and the MSEP was calculated according to equation 2.21. In order to take the variance of the validation or test set into consideration, the MSEP was then divided by the variance of the validation or test set at hand:

$$\text{Measure of quality} = \frac{\text{MSEP}}{\sigma^2} \tag{3.1}$$

This expression is simplified when using the biased estimator for the variance

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(y_{val,i} - \bar{y}_{val})^2 \tag{3.2}$$

where $y_{val,i}$ is the i$^{\text{th}}$ observation of y from the validation set and $\bar{y}_{val}$ is the mean of the validation set observations of y. Substituting σ^2 in equation 3.1 for $\hat{\sigma}^2$ from equation 3.2, and using equation 2.21 for the MSEP, the following expression is obtained:

$$\text{Measure of quality} = \frac{\sum_{j=1}^{N}(y_{val,j} - \hat{y}_{val,j})^2}{\sum_{i=1}^{N}(y_{val,i} - \bar{y}_{val})^2} \quad \left(= \frac{\text{RSS}}{\text{TSS}}\right) \tag{3.3}$$

where $\hat{y}_{val,j}$ is the predicted y-value. The Residual Sum of Squares (RSS) and Total Sum of Squares (TSS) are defined in equations 2.18 and 2.19.

This measure of quality, hereafter referred to as the *normalized mean squared error of prediction*, is very closely related to the Q^2 value in equation 2.20 but it is always positive which was convenient when using cost ranking as described previously in section 3.5.4.

The normalized MSEP values for the PLS model using a certain number of latent variables up to a maximum were saved. If more than one validation set was used, a model was built for each set and the normalized MSEP from the different validation sets were added latent variable-wise. That is to say, the normalized MSEP for all models using n latent variables were added together. This method was used since in a real life scenario, a model with a fixed number of latent variables would most often be used. Finally, the lowest normalized MSEP, or the lowest average if several validation sets were used, was saved and used as fitness score for an individual. The procedure is summarized in equation 3.4,

$$\text{Fitness score} = \min_{m} \frac{1}{K}\sum_{j=1}^{K} e_{j,m}, \quad m = 1, \dots, M \tag{3.4}$$

where $e_{j,m}$ is the normalized MSEP from validation set j using m latent variables. K is the number of validation sets.

3.5.6.1 Low resolution data set For the low resolution data set, a variant of cross-validation as described in section 2.3.4.1 with partially overlapping subsets was used. A maximum of 10 latent variables was decided upon after some initial trials.

3.5.6.2 High resolution data set

For the high resolution data set only one validation set as described in section 2.3.4.2 was used due to computation time. A maximum of 15 latent variables was decided upon after some initial trials.

3.5.7 Calculation cost

The number of evaluations of the fitness function for the initial population is the number of individuals in the population, N_{pop}. For subsequent generations, the number of evaluations depends on the population size, the crossover rate, the mutation rate and the number of elite individuals. This leads to the following total number of fitness function evaluations:

$$N_{eval} = N_{pop} + N_{gen}(\tilde{R}_{mut}(N_{pop}(1 + R_{cross}) - N_{elite})) \tag{3.5}$$

where $\tilde{R}_{mut}$ denotes the average ratio of mutated individuals per generation. This differs from the input mutation rate in this implementation of the GA as is described in section 3.5.5. Because of the way mutation was performed, the number of individuals mutated in each generation changed a bit but an average could be obtained through simulations.

For comparison, it is helpful to be able to calculate the total number of possible combinations of preprocessing methods. This would be the number of fitness function evaluations needed in an exhaustive search. Looking only at the first step, there are q possible choices of letters, q being the alphabet size (the number of preprocessing methods plus the stop character). The stop character denotes the "do nothing" option which will also forbid further steps. Consequently, there are $q - 1$ possibilities that allow for a second step. The second step also offers q possible choices of letters, however, the stop character again simply states that there are no further steps, leaving the letter chosen in the first step as the only preprocessing method; $\{a, 0\} = \{a\}$, a being the first step. Since this option was already accounted for in the first step, there are only $q - 1$ choices of letters after the first step that translates into new combinations of preprocessing methods. This means that after two steps, there are $q + (q-1)(q-1)$ or $1 + (q-1) + (q-1)(q-1)$ possible combinations. These calculations are formalized in equation 3.6, where l is the length of the chromosome in decimal form, i.e. the maximum number of preprocessing steps.

$$N_{preproc\_comb} = \sum_{k=0}^{l} (q-1)^k \tag{3.6}$$

4 Results

In this section, results from the investigations are presented. Some brief comments are also included but the main discussion can be found in later sections. For the preprocessing results, one important note is that the data was always mean centered along the columns before PLS regression. This was a built in trait in the `plsregress` command which was used for PLS regression in MATLAB. Consequently, when chosen as the final step before regression, scaling the column variance to one (*S3*) and autoscaling the columns (*S4*) resulted in the same operation.

4.1 Low resolution data set

The explained variance for the original low resolution data set and the decimated low resolution data set without any preprocessing is shown in figure 4.1. As can be seen, there does not seem to be any negative effects resulting from the decimation. However, all values of Q^2 are negative suggesting poor predictive capabilities of the model (the residual sum of squares after prediction is greater than the total sum of squares of the actual values).

The corresponding explained variance for each of the three validation sets used in the cross-validation is shown in figures 4.3a and 4.3b. It is evident that there is no real consistency in the optimum number of latent variables across the validation sets, especially not between validation set 3 and the other two sets. This no doubt contributes to the poor results but even for the individual validation sets, the values for the explained variance are negative.

4.1.1 Optimization

The results from the optimization are given in tables 4.1 and 4.2. For both the original and the decimated data, the common denominator amongst the chosen preprocessing sequences is the use of derivatives. The optimization on the decimated data set did not provide any conclusive results, almost every optimization run produced a different result. Still, the chosen sequences can roughly be divided into two groups: derivatives followed by smoothing and derivatives followed by scaling. The last group contains only one sequence, *D8, S5, S3*, and this is also the only sequence that was chosen more than once by the algorithm. Three sequences do not fall into either of these categories; two contain both smoothing and scaling in addition to derivatives and one sequence starts with smoothing followed by derivatives. It can also be noted that column autoscaling is the final step in the top five solutions. Based on these results, there seems to be a possibility of increasing the quality of the model built on the decimated data set, however, no particular preprocessing sequence was heavily favoured by the algorithm. In addition, the model is still very poor after preprocessing with negative values for the explained variance. Therefore, the analysis that follows will be focused on the original low resolution data set.

For the original data set, the sequence *D2, G7, G7, G7* was chosen most frequently by the algorithm. *D2* represents derivatives using a 3rd order Savitzky-Golay smoothing filter with a window of size 21, and *G7* represents Savitzky-Golay smoothing using a 3rd order filter of size 43 which was also the maximum window size available to the algorithm. The explained variance of the data set before and after this preprocessing sequence is shown in figure 4.2. In order to display the effect on the individual validation sets, the explained variance for the individual validation sets is displayed in figure 4.3c. It seems as if fewer latent variables are needed in order to obtain the best possible model when compared to not using preprocessing. The model is however still very poor as indicated by the negative values of Q^2.

Table 4.1: Results from the optimization on the original data

Preprocessing						Score	Penalty	MSEP$_{norm}$
-	-	-	-	-	-	1.767	-	1.767
D1	D1	S5	G6	G7	G7	1.104	0.09	1.014
D2	S3	G7	S5	G7	-	1.163	0.05	1.113
D2	G7	G7	G7	-	-	1.164	0.01	1.154
D2	G7	G7	G7	-	-	1.164	0.01	1.154
D2	G7	G7	G7	-	-	1.164	0.01	1.154
D2	G7	G7	G7	-	-	1.164	0.01	1.154
D2	G7	G7	G7	-	-	1.164	0.01	1.154
D7	G6	G7	G7	-	-	1.170	0.01	1.160
D7	G7	G7	G6	-	-	1.184	0.01	1.174
D7	G7	G7	-	-	-	1.197	-	1.197

Table 4.2: Results from the optimization on the decimated data

Preprocessing						Score	Penalty	MSEP$_{norm}$
-	-	-	-	-	-	1.682	-	1.682
D6	G13	S7	S3	-	-	1.121	0.01	1.111
D8	S5	S3	-	-	-	1.164	-	1.164
D8	S5	S3	-	-	-	1.164	-	1.164
D8	S5	S4	-	-	-	1.164	-	1.164
D3	S5	G9	S3	-	-	1.174	0.01	1.164
D1	G2	G3	-	-	-	1.187	-	1.187
D1	G2	G12	-	-	-	1.191	-	1.191
G2	D1	G3	-	-	-	1.192	-	1.192
D6	G3	G2	-	-	-	1.193	-	1.193
D1	G11	G3	-	-	-	1.193	-	1.193

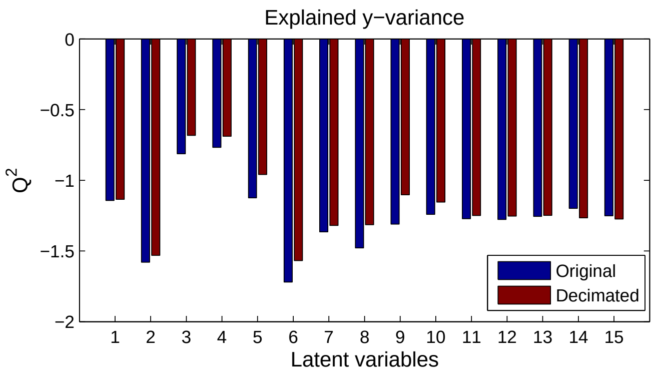


Figure 4.1: Explained variance for the original and decimated low resolution data sets

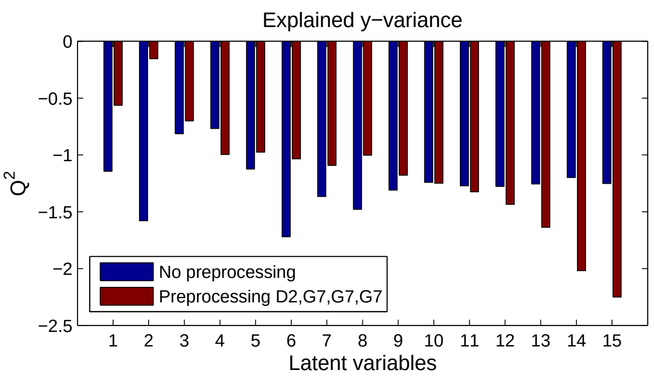
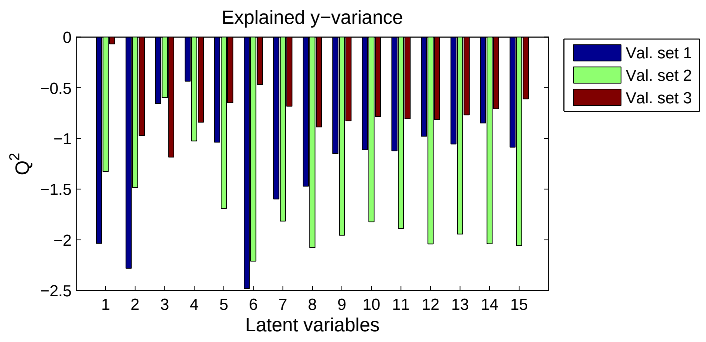
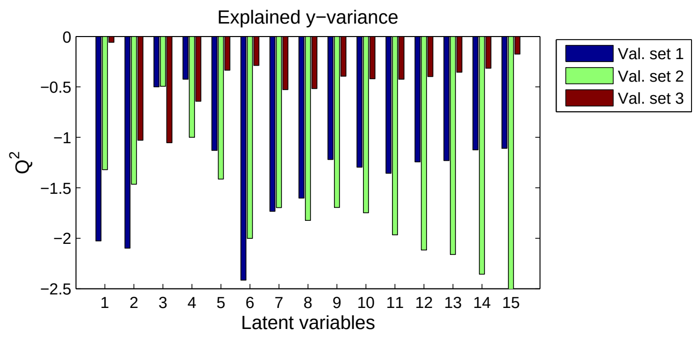


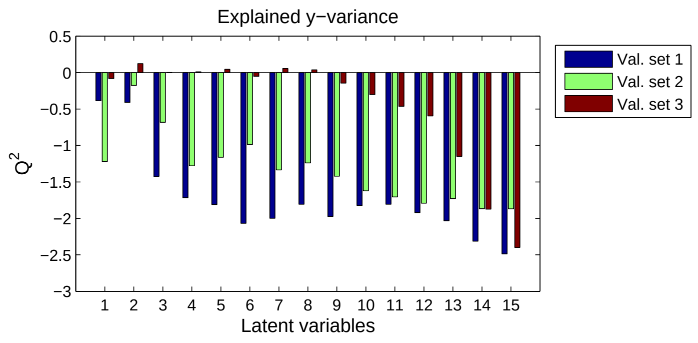
Figure 4.2: Explained variance before and after preprocessing (original low resolution data set)



(a) Original data set without preprocessing



(b) Decimated data set without preprocessing

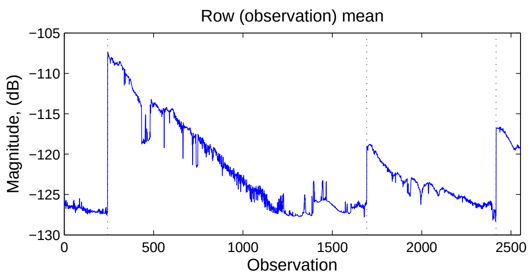


(c) Preprocessed original data set (*D2, G7, G7, G7*)

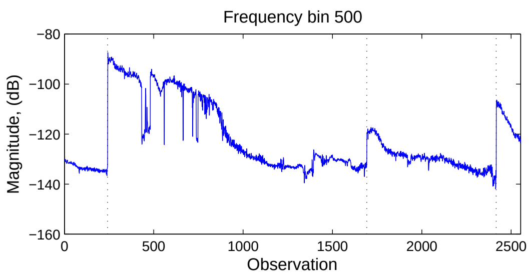
Figure 4.3: Explained variance for the three validation sets

4.2 High resolution data set

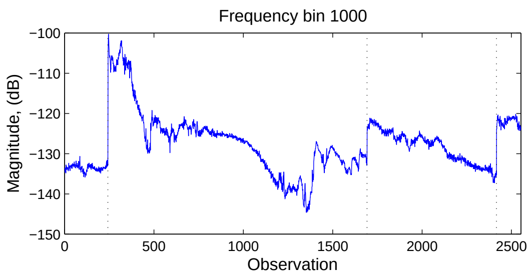
In order to visualize the observations in the high resolution data before any data reduction, a PCA analysis was performed. The data was mean centered before the PCA. A score plot for the first two components is shown in figure 4.5. In these two components, the observations are highly structured, a large part of them describing an “arc looking” pattern. This structure is somewhat correlated with the time when the observations were made. This can be seen coloring the observations according to the date when they were collected. Due to the large number of observations, this is only shown after the data reduction in figure 4.6. The observations seem to be grouped according to the time when they were recorded and they seem to “drift” with time along the first component. An exception is the observations recorded at the 2nd of February, which are grouped in two separate areas. One possible reason for this time correlation could be the accumulation of deposits on the inside of the tube. These deposits absorb some of the vibrations and the received signal power decreases over time as the deposits accumulate. This effect is shown in figure 4.4a where the mean of the observations is plotted over time. The dotted vertical lines indicate when the system has been flushed, actions which clean the pipe from deposits. The graph clearly shows that the received signal power increases directly after the flushings and then decreases over time. This difference in magnitude of the observations might be what is captured in the first components of the PCA score plot. In figures 4.4b and 4.4c, which show how two single variables (PSD magnitudes at different frequencies) evolve over time, it can be seen that the effect of the deposits differ for different frequencies and is not as simple as that which is suggested by simply looking at the mean of each observation (mean of PSD magnitudes over all frequencies at a given time).



(a) Evaluation of observation means over time



(b) Evaluation of frequency bin no. 500 over time



(c) Evaluation of frequency bin no. 1000 over time

Figure 4.4: Different frequency bins over time

4.2.1 Data reduction

A PCA score plot for the observations after the data reduction described in section 3.3.2 is shown in figure 4.6. When comparing figures 4.6 and 4.5, it is clear that the data reduction has an adverse effect on the detail level; much of the structure in the third and fourth quadrant of figure 4.5 has disappeared in figure 4.6. No outliers are present in the first components in the reduced data set.

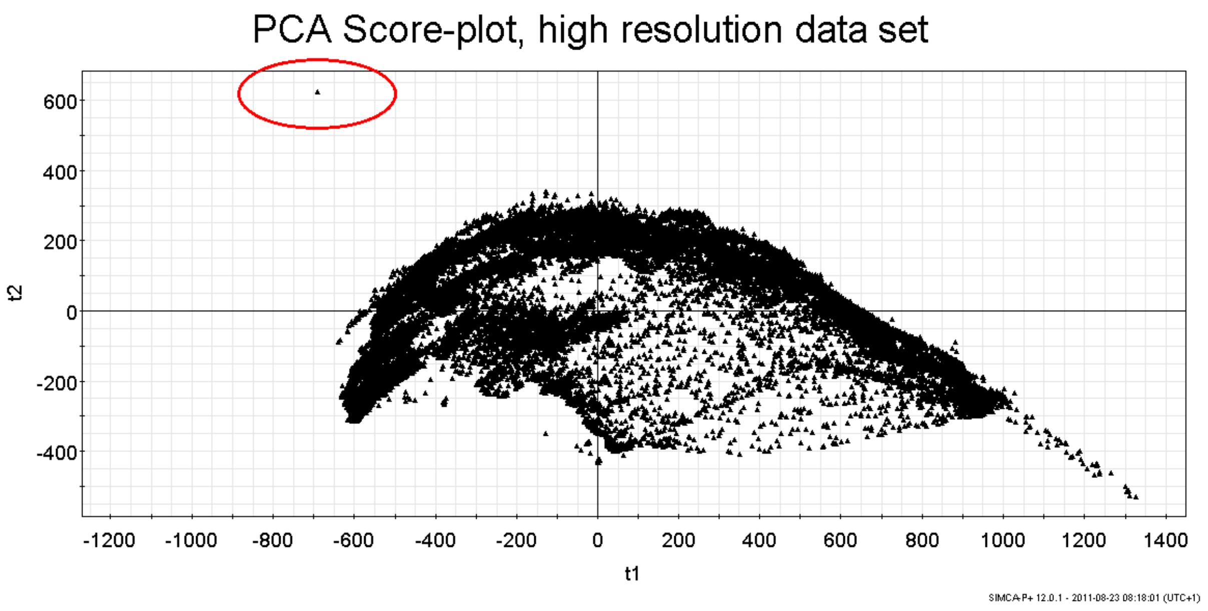


Figure 4.5: PCA score-plot of the high resolution data set before data reduction, encircled: outlier

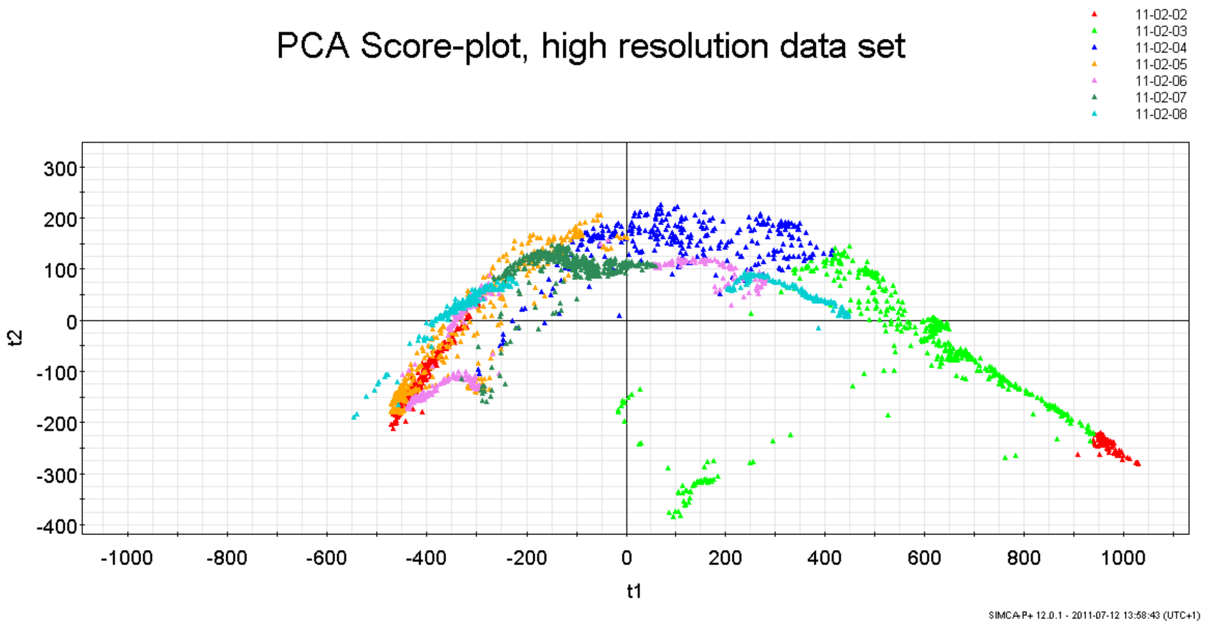


Figure 4.6: PCA score-plot of the high resolution data set after data reduction

4.2.2 Optimization

The results of the optimization on the high resolution data set can be seen in table 4.3. They differ from those in the previous section where the use of derivatives was the common denominator. For the high resolution data set, different scaling methods are dominating. The sequence *S1, S5, S3* in particular was chosen six out of ten times. As a reminder, whether *S3* or *S4* was chosen as the final step made no difference since the data was always mean centered before PLS regression by default. The same sequence can also be found in three of the remaining four solutions, accompanied by an additional step. In the last remaining solution, an additional step has been inserted inside the sequence *S1, S5, S3*. The sequence *S1, S5, S3* (*S1, S5, S4*) denotes column centering, scaling of row vector length to one and autoscaling of columns.

It is somewhat surprising to see *S6* as the final step for the best candidate as it only performs the intended average power normalization when it is located in the first position. During testing however, it was discovered that adding *S6* after *S1, S5, S3* had very little effect on the predicted y-values.

The results of the optimization on the wavelet coefficient high resolution data set can be seen in table 4.4. For this data set, the results show that the preprocessing methods tested in this thesis hardly improves the quality of the model at all. It can also be seen that when compared to the high resolution data set without preprocessing, there does not appear to be any significant adverse effect from using wavelet coefficients and discarding some of the coefficients describing the details.

A comparison between the explained variance for the original high resolution data set and the wavelet coefficient high resolution data set is given in figure 4.7. The values are similar up until around eleven to twelve latent variables.

Table 4.3: Results from the optimization on the original data

Preprocessing						**Score**	**Penalty**	**MSEP**$_{norm}$
-	-	-	-	-	-	0.717	-	0.717
S1	S5	S3	S6	-	-	0.594	0.01	0.584
S1	S5	S2	S3	-	-	0.601	0.01	0.591
S6	S1	S5	S3	-	-	0.606	0.01	0.595
G13	S1	S5	S3	-	-	0.612	0.01	0.602
S1	S5	S3	-	-	-	0.614	-	0.614
S1	S5	S3	-	-	-	0.614	-	0.614
S1	S5	S3	-	-	-	0.614	-	0.614
S1	S5	S3	-	-	-	0.614	-	0.614
S1	S5	S4	-	-	-	0.614	-	0.614
S1	S5	S4	-	-	-	0.614	-	0.614

Table 4.4: Results from the optimization on the wavelet compressed data

Preprocessing						**Score**	**Penalty**	**MSEP**$_{norm}$
-	-	-	-	-	-	0.722	-	0.722
S1	G9	-	-	-	-	0.707	-	0.707
G9	S1	-	-	-	-	0.707	-	0.707
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715
S2	-	-	-	-	-	0.715	-	0.715

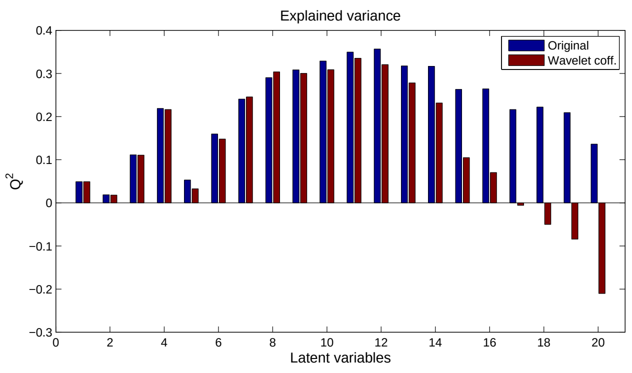


Figure 4.7: Explained variance for the original and wavelet coefficient high resolution data sets (test set)

4.2.3 Testing

Some of the best models from the optimization were chosen for testing in order to determine how well they would predict a different data set as an assessment of their generality. The model built on data without preprocessing was also tested and used for comparison.

For the high resolution data set, three sequences were chosen for testing. The first one was *S1, S5, S3, S6*, chosen since it obtained the highest optimization score. The second sequence was *S1, S5, S3*, chosen since it was selected by the genetic algorithm six out of ten times. The last sequence used for testing was *S6, S1, S5, S3*, chosen since it is easy to understand the effect of using *S6* as the first method. The explained variance from these tests are shown in the graph in figure 4.8a. The only preprocessing sequence that improves the results for both the validation set and the test set is *S6, S1, S5, S3*. The other ones chosen for testing do not perform better than the non-treated data.

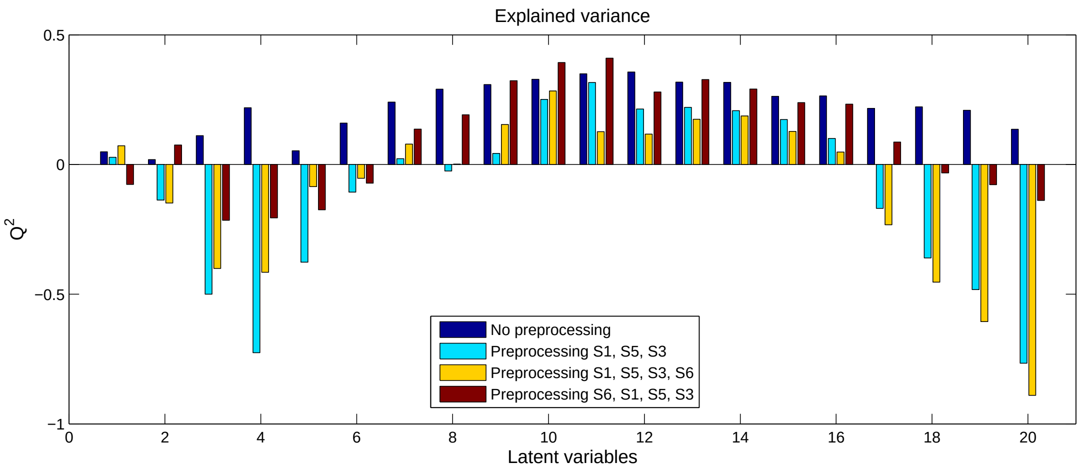
For the wavelet coefficient high resolution data set, all three sequences selected by the algorithm were chosen for testing. Since *S1, G9* and *G9, S1* were found to produce the same results, only those for *S1, G9* are displayed in the figures and graphs that follow. The explained variance is shown in the graph in figure 4.8b. It is only *S2* that improves the results for both the validation set and test set. For *S1, G9*, there is a large difference between the results for the validation set and the test set. Using the validation set, a model built with ten latent variables produced the best model whereas the best model for the test set was built using only four latent variables.

In order to further compare the chosen preprocessing sequences, the predicted y-values were plotted together with the actual y-values for the test set. The results are shown in figures 4.9 and 4.10 for the high resolution and wavelet coefficient high resolution data set respectively. The optimum number of latent variables according to the results using the validation set was used for each sequence.

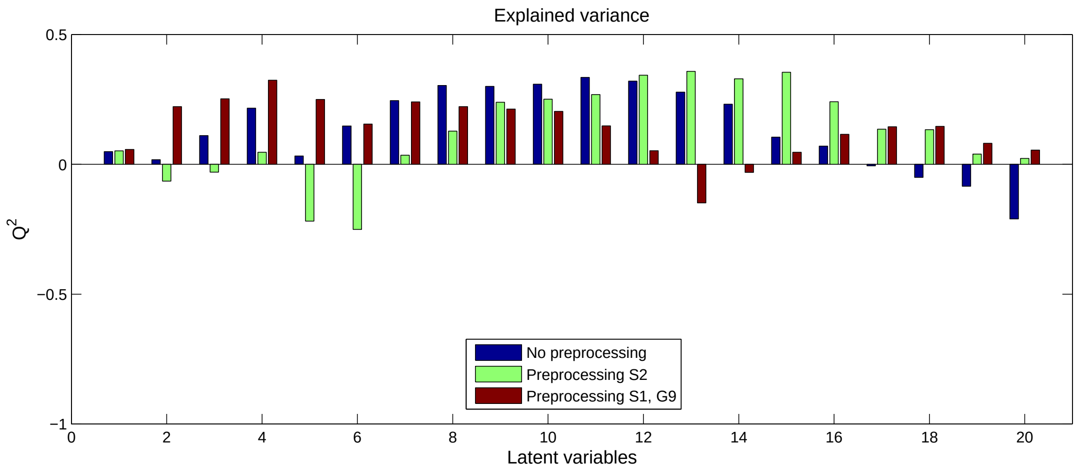
For the high resolution data set, the improvements compared to the non-preprocessed data are mainly on how well peaks are predicted. In all predicted y-observations resulting from preprocessed data depicted in figure 4.9, the peaks seem to be modelled better. This is especially evident right in the beginning, between observations 1850 and 2050, and at the peak around observation 2420. In the range between observation 2050 and 2250 however, the predicted y-values after preprocessing are consistently too low. An interesting range is between observation 2430 up to the end, since this is where the different preprocessing sequences differ the most from one another. Whereas *S1, S5, S3* and *S1, S5, S3, S6* seem to exaggerate the variations, *S6, S1, S5, S3* seem to handle them better and this is probably the main source of its success when looking at the overall prediction error.

For the wavelet coefficient high resolution data set, using preprocessing *S2* does not change the predicted y-values much, as can be seen by comparing the graphs in figures 4.10a and 4.10c. However, using the sequence *S1, G9* gives a result that is different from the other two as can be seen in the graph in figure 4.10b. Peaks, valleys in particular, are not predicted as well as in the other two graphs. However, the predicted values seem closer to the actual ones towards the beginning and the end of the y-value curve. The fact that the predicted values using *S1, G9* might differ was also hinted by the explained variance in figure 4.8b. The Q^2 values for no preprocessing using eleven latent variables and those for preprocessing *S2* using twelve latent variables are quite similar whereas the explained variance for *S1, G9* using ten latent variables is much lower. Ten latent variables produced the best results for *S1, G9* for the validation set. By comparing figures 4.9a and 4.10a, it can also be seen that without further preprocessing, the predictions made using a model built on the wavelet coefficient high resolution data set are very similar to those obtained from using a model built on the original high resolution data set.

The comparisons in figures 4.9 and 4.10 also show the difficulty in describing the quality of a model. The normalized MSEP e.g. is an objective measure not taking into account the type of deviation (small peaks, offsets e.g.) or where the deviation occurs.

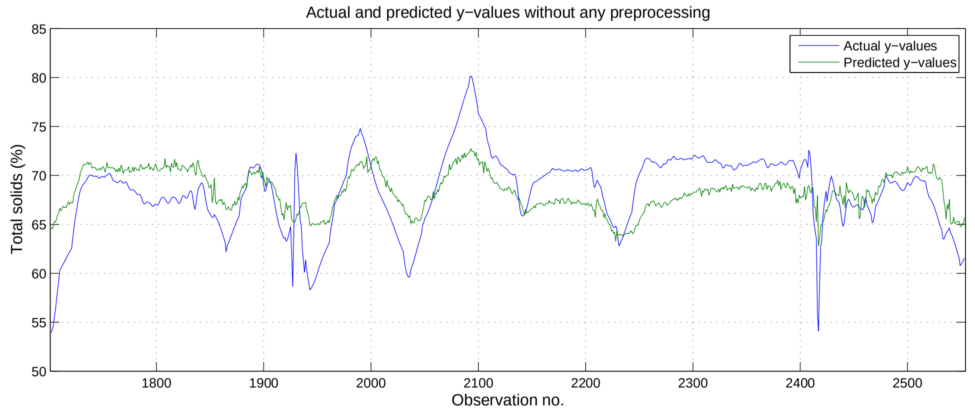


(a) High resolution data set

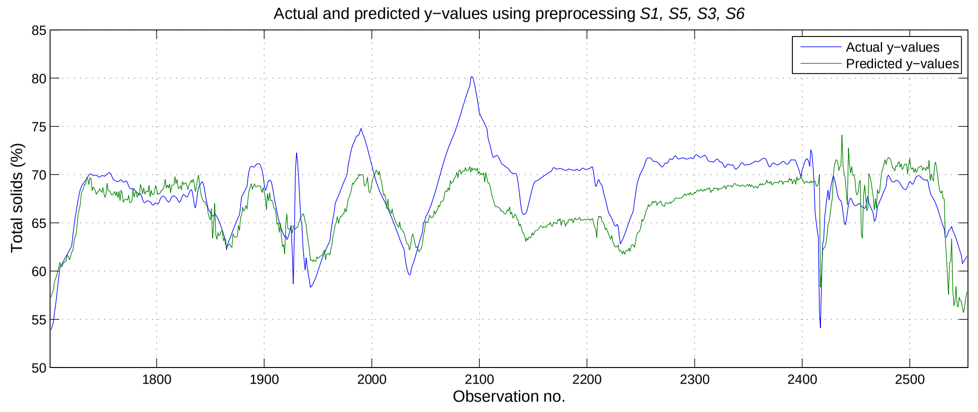


(b) High resolution wavelet coefficient data set

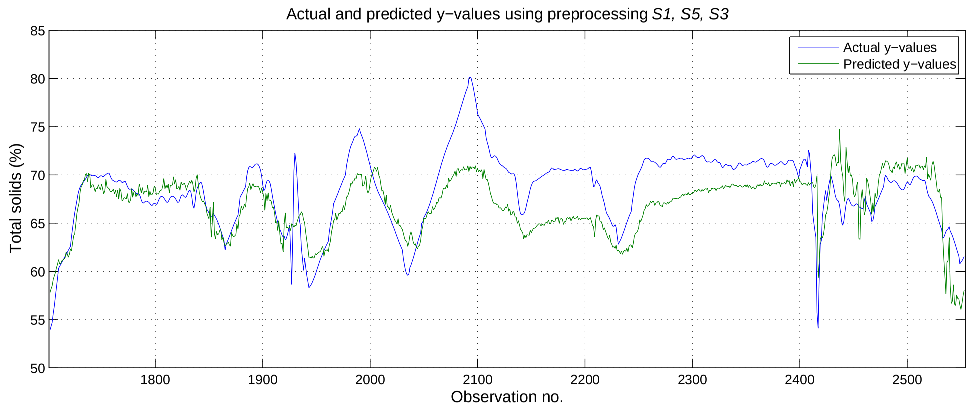
Figure 4.8: Explained variance for the test set using different preprocessing sequences



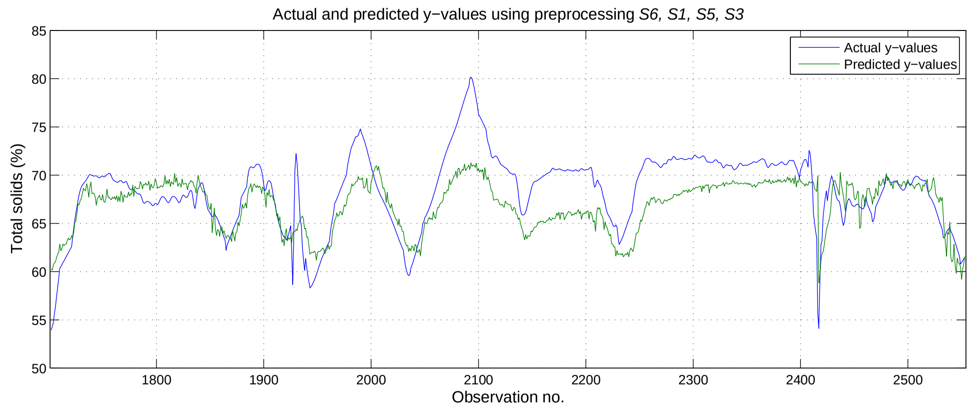
(a) Predicted y-values without preprocessing, 11 latent variables



(b) Predicted y-values after *S1, S5, S3, S6*, 10 latent variables

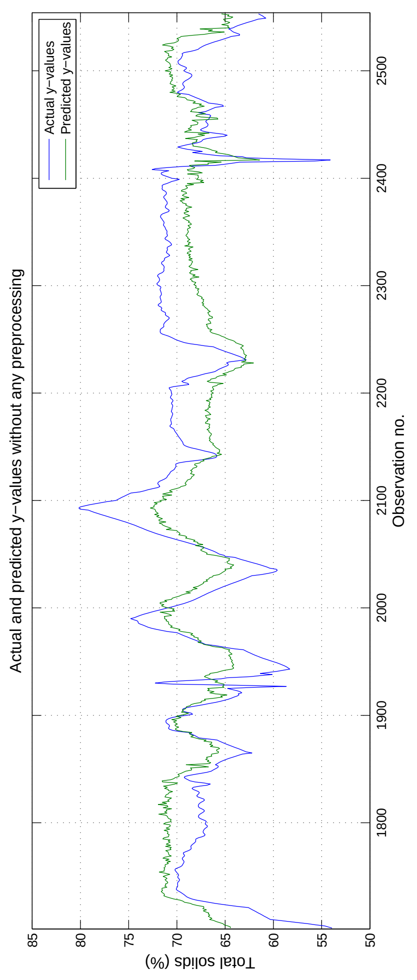


(c) Predicted y-values after *S1, S5, S3*, 11 latent variables

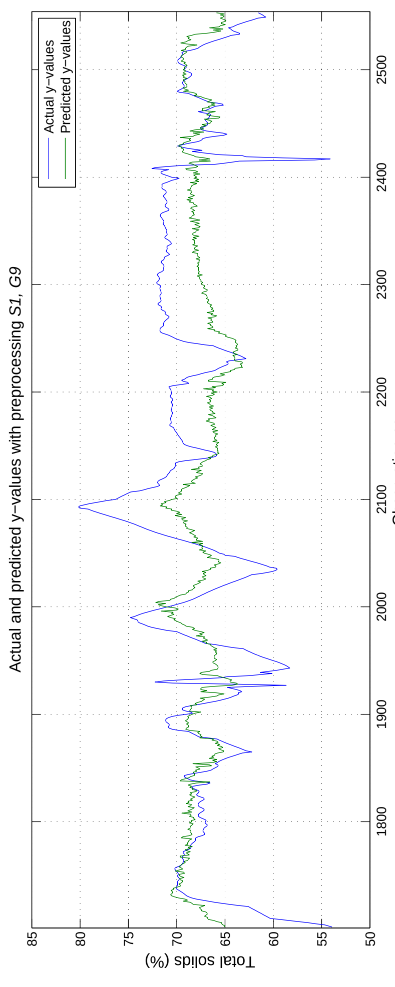


(d) Predicted y-values after *S6, S1, S5, S3*, 11 latent variables

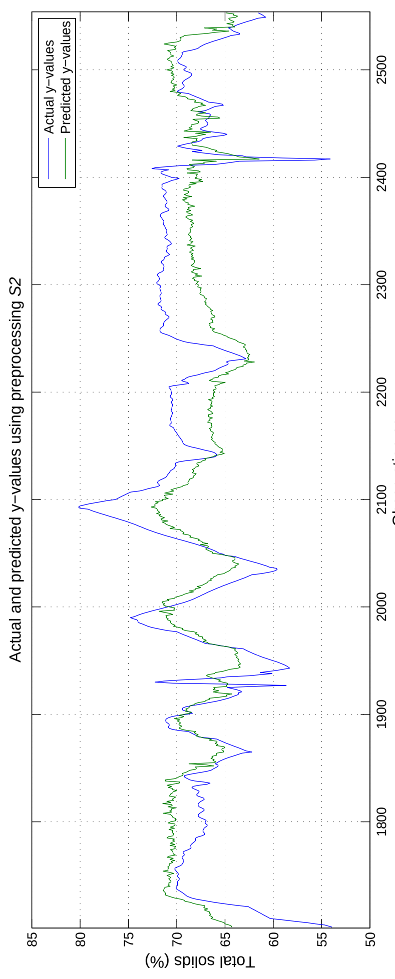
Figure 4.9: Prediction of total solids using the high resolution data set



(a) Predicted y-values without preprocessing, 11 latent variables



(b) Predicted y-values after *S1*, *G9*, 10 latent variables



(c) Predicted y-values after *S2*, 12 latent variables

Figure 4.10: Prediction of total solids using the high resolution wavelet coefficient data set

4.3 Summary - optimization results

For this summary, given in table 4.5, only one preprocessing sequence has been selected for each data set. The selection was made by considering not only the optimization score but also the complexity of each sequence. For the high resolution data sets, the results from the testing were considered rather than those from the training.

Table 4.5: Summary of the tests on different data sets

Data set	**Preprocessing**	$\mathbf{Q^2_{val}}$	$\mathbf{Q^2_{test}}$	$\mathbf{RMSEP_{test}}$	**Latent var.**
High res.	-	0.283	0.349	3.13	11
High res. wav.	-	0.278	0.335	3.16	11
Low res.	-	-0.767	-	-	4
Low res. decim.	-	-0.682	-	-	3
High res.	S6, S1, S5, S3	0.404	0.410	2.98	11
High res. wav.	S2	0.285	0.343	3.14	12
Low res.	D2, G7, G7, G7	-0.154	-	-	2
Low res. decim.	D8, S5, S3	-0.164	-	-	2

4.4 Calculation cost

For all data sets, a total of 10 optimization runs were performed. Using equation 3.5, the total number of evaluations of the fitness function can be estimated. The actual figure will be lower since the algorithm saves the result for all evaluated chromosomes in order to avoid remaking the same calculation twice. $\tilde{R}_{mut}$, the average ratio of mutated individuals per generation was obtained through simulations for a mutation rate of 0.035 which was used for all optimization runs. The results, seen in figure 3.11, shows the probability of having no mutations in a chromosome to be approximately 0.34, giving an $\tilde{R}_{mut}$ value of 0.66. The crossover rate was 0.8, the number of elite individuals was 2 and the population sizes were 260 and 240 for the low resolution and high resolution data sets respectively. The number of generations needed for the algorithm to reach the stopping criterion is given in table 4.6 along with the estimated number of evaluations of the fitness function rounded up to the nearest thousand. As a reference, the total number of fitness function evaluations using an exhaustive search would be approximately $9 \cdot 10^8$, using equation 3.6 with $q = 32$ and $l = 6$.

The most time consuming data set was the high resolution data set. The total calculation time for that data set was roughly 5 days running two optimization algorithms at the same time on parallel processes. This motivates the use of the optimization algorithm since the number of calculations was reduced over 6000 times compared to an exhaustive search. An exact number for the time consumption of an exhaustive search is not possible to obtain from these calculations since the evaluation time of a preprocessing sequence depended on the length of the sequence and the type of preprocessing. Thus, the reduction in total number of evaluations is not necessarily proportional to the reduction in computation time. It is however evident that the reduction in computation time was considerable and that an exhaustive search would not have been feasible.

The complexity of the cost surface is hinted in figure 4.11 showing the cost surface for two preprocessing steps on the high resolution data set. Even if only two steps are used, there are several local minima making the optimization task more difficult. In addition, some of the minima are very small, suggesting that they may be very difficult to find.

Table 4.6: The number of generations needed for the different tests

Data set	Min	Max	Average	Eval. of fitness funct.
High res.	32	79	49	140,000
High res. wav.	15	25	19	55,000
Low res.	28	49	37	115,000
Low res. decim.	20	46	29	88,000

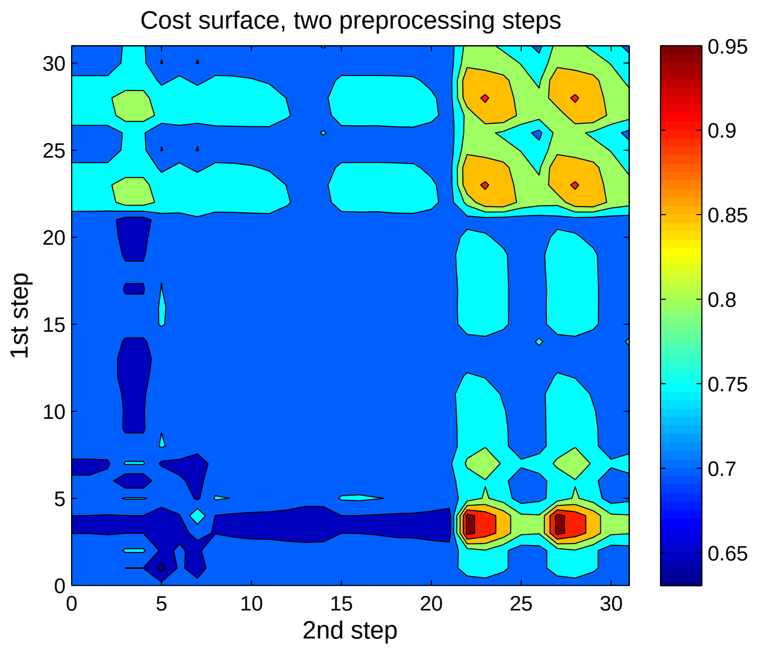


Figure 4.11: Cost surface for two preprocessing steps, high resolution data set. The colorbar values are normalized MSEP.

5 Discussion

The investigation of the genetic algorithm developed in this project was limited. Although not treated in this report, there are e.g. other types of selection, crossover and mutation which might have performed better. Also, ones these parameters had been fixed, inputs such as mutation rate and crossover rate were largely decided on a trial and error basis. It is therefore likely that the performance of the genetic algorithm could be improved by a more thorough investigation towards that end.

One such problem that could have an interesting solution is how to generate the initial population. Due to the high cardinality of the alphabet used for the chromosomes (there are 31 different preprocessing methods to choose from), the required population size for sufficient sampling of the cost surface becomes quite big. It is however possible to provide better sampling by generating initial populations using codes instead of random values [11]. This might improve the performance of the genetic algorithm since the population sizes used in this thesis are small for such a high cardinality alphabet.

When it comes to how the genetic algorithm was used in this thesis, the choice of how many times the optimization was performed on each data set could be discussed. Using ten runs was decided upon based on the time consumption of the largest data set (the high resolution data set). It would however have been possible to use considerably more runs for the smaller data sets and this might have produced clearer results. The reason for why this was not done was to provide consistency between the investigations on the different data sets.

As to the results from the optimization, having a good resolution of observations is clearly preferable as seen when comparing the results from the high and the low resolution data sets. But even for the high resolution data set, the fact that most of the preprocessing sequences that performed well when using the validation set performed poorly for the test set, raises some doubts regarding the sequences that performed well for both sets. It is possible that these sequences were particular to those two particular sets and that a cross-validation approach would have been a better choice. The reason for why this was not used was, as mentioned before, the time it would have taken to evaluate the high resolution data set.

The preprocessing methods investigated in this thesis are standard ones, not adapted specifically for this type of data. Developing more specialized preprocessing methods, feature extraction e.g., might give better results. Also, the investigation was limited by the default column centering performed by the `plsregress` command in MATLAB.

The way the quality of a model was determined in this thesis was also quite standard. Using a more specialized measure of quality could help focusing on specific traits such as peak characterization or steady state offsets.

There was not enough time to investigate how the accumulation of deposits affected the recorded signal and whether this knowledge could be used to improve the predictions. This could be the basis for future work. Using more information as input, such as fluid temperature, might also help improve the predictions.

Another subject that was not considered due to time constraints was the possibility of obtaining a model that used fewer latent variables to achieve a similar predictive capability to that of a model based on non-preprocessed data. This would alleviate some of the computational burden of the digital signal processor used in the Acospector.

6 Conclusions

It has been shown in this thesis that using a genetic algorithm in order to select preprocessing methods for PLS regression of acoustic spectroscopy data works and provides good solutions for the optimization problem as it is stated. The main difficulty lies in finding a good measure of quality for a prediction model in order to use in the fitness function.

Further investigation of the settings in the genetic algorithm is needed to provide a faster optimization and possibly more accurate results.

The results from the optimization show that it is possible to improve the prediction error for a given data set. However, most preprocessing sequences seem to be specific to that particular data set. Consequently, it cannot be said that a preprocessing sequence was found that could improve the prediction results in general or that could provide a more robust model.

There does seem to be possibilities of creating a faster measurement procedure by adjusting the size of the temporal window when constructing the PSD spectra. Using wavelet compressed spectra would also improve the time used for calculating the regression, although the benefit of this depends on the time needed for calculating the wavelet transformation.

References

[1] *Acospector Acoustic Chemometer.* `http://www.acosense.com/technology/product`. 2011.

[2] Richard G. Brereton. *Applied Chemometrics for Scientists.* Chichester, England: John Wiley & Sons, Ltd, 2007. ISBN: 978-0-470-01686-2.

[3] Paul Geladi and Bruce R. Kowalski. “Partial least-squares regression: a tutorial”. In: *Analytica Chimica Acta* 185 (1986), pp. 1–17.

[4] John M. Harris, Jeffry L. Hirst, and Michael J. Mossinghoff. *Combinatorics and Graph Theory.* Springer, 2008. ISBN: 978-0-387-79710-6.

[5] Randy L. Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms.* Hoboken, USA: John Wiley & Sons, Inc, 2004. ISBN: 0-471-45565-2.

[6] Roger M. Jarvis and Royston Goodacre. “Genetic algorithm optimization for pre-processing and variable selection of spectroscopic data”. In: *Bioinformatics* 21.7 (2005).

[7] Riccardo Leardi and Amparo Lupiáñez González. “Genetic algorithms applied to feature selection in PLS regression: how and when to use them”. In: *Chemometrics and Intelligent Laboratory Systems* 41 (1998), pp. 195–207.

[8] A. D. McNaught and A. Wilkinson. *Compendium of Chemical Terminology (the “Gold Book”).* 2nd ed. XML on-line corrected version: http://goldbook.iupac.org (2006-) created by M. Nic, J. Jirat, B. Kosata; updates compiled by A. Jenkins. Blackwell Scientific Publications, 1997. ISBN: 0-9678550-9-8.

[9] Scott L. Miller and Donald G. Childers. *Probability and Random Processes.* Elsevier Academic Press, 2004. ISBN: 0-12-172651-7.

[10] Charles L. Phillips, John M. Parr, and Eve A. Riskin. *Signals, Systems and Transforms.* Pearson Education, Inc., 2003. ISBN: 0-13-041207-4.

[11] Colin R. Reeves. “Using Genetic Algorithms with Small Populations”. In: *Proceedings of the Fifth International Conference on Genetic Algorithms.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993, pp. 92–99. ISBN: 1-55860-299-2.

[12] Colin R. Reeves and Jonathan E. Rowe. *Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory.* Dordrecht, The Netherlands: Kluwer Academic Publishers, 2003. ISBN: 0-4020-7240-6.

[13] *Simca-P+ 12 Tutorial.* `http://www.umetrics.com/default.aspx?id=8170&ptid=0`. 2008.

[14] Anatoli Torokhti and Phil Howlett. *Computational Methods for Modelling of Nonlinear Systems.* Vol. 212. Mathematics in Science and Engineering. Elsevier B.V., 2007. ISBN: 978-0-444-53044-8.

[15] Felix Törner. “Analys av Svartlut med Aktiv Akustisk Spektrometri”. MA thesis. Linköpings universitet.

[16] Michael Weeks. *Digital Signal Processing Using MATLAB and Wavelets.* Infinity Science Press LLC, 2007. ISBN: 0-9778582-0-0.

A Fourier transforms

The following equations describe some basic properties of the Fourier transform. All information was collected from [10].

The continuous Fourier transform and its inverse:

$$\mathscr{F}\{x(t)\} = X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}\,\mathrm{d}t \tag{A.1}$$

$$\mathscr{F}^{-1}\{X(\omega)\} = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}\,\mathrm{d}\omega \tag{A.2}$$

The discrete-time Fourier transform and its inverse:

$$\mathscr{D}\mathscr{T}\mathscr{F}\{x[n]\} = X(\Omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-jn\Omega}, \quad x[n] = x(nT_s), \quad \Omega = \omega T_s \tag{A.3}$$

$$\mathscr{D}\mathscr{T}\mathscr{F}^{-1}\{X(\Omega)\} = \frac{1}{2\pi}\int_{2\pi} X(\Omega)e^{jn\Omega}\,\mathrm{d}\Omega \tag{A.4}$$

where T_s is the sampling interval in seconds.

The discrete Fourier transform and its inverse:

$$\mathscr{D}\mathscr{F}\{x[n]\} = X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N}, \quad k = 0, 1, \dots, N-1 \tag{A.5}$$

$$\mathscr{D}\mathscr{F}^{-1}\{X[k]\} = \frac{1}{N}\sum_{k=0}^{N-1} X[k]e^{j2\pi kn/N}, \quad n = 0, 1, \dots, N-1 \tag{A.6}$$

If a continuous time signal $x(t)$ is sampled at intervals nT_s, the sampled signal is $x_s(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s)$ and its Fourier transform is $\mathscr{F}\{x_s(t)\}$ which using A.1 becomes:

$$\mathscr{F}\{x_s(t)\} = \sum_{n=-\infty}^{\infty} x(nT_s)e^{-jn\omega T_s} \tag{A.7}$$

This gives rise to the Discrete-time Fourier transform given in A.3, where Ω is a continuous variable, periodic with period 2π. $X(\Omega)$ is as a result a continuous function, and cannot be handled in a computer unless it is sampled. If one period of $X(\Omega)$ is sampled at intervals $2\pi k/N$, the sampled function is $X_S(\Omega) = \sum_{k=0}^{N-1} X(2\pi k/N)\delta(\Omega - 2\pi k/N)$. This function can be obtained directly from the sampled time domain signal $x[n]$, and the transform for doing so is the Discrete Fourier transform given in A.5.

B Convolution

The following equations describe some basic properties of convolution. All information was collected from [10].

The convolution integral:

$$x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\,\mathrm{d}\tau \tag{B.1}$$

The associative property of convolution:

$$(x * h_1) * h_2 = x * (h_1 * h_2) \tag{B.2}$$

Convolution and Fourier transforms:

$$\mathscr{F}\{x(t) * h(t)\} = X(\omega)H(\omega) \tag{B.3}$$

$$\mathscr{F}\{x(t)h(t)\} = X(\omega) * H(\omega) \tag{B.4}$$

where $X(\omega)$ is the Fourier transform of $x(t)$.

C Random processes

The information in this appendix is collected from [9].

A random variable X can be described as a function whose values are the result of some experiment. A coin flipping experiment could e.g. generate a random variable with two possible outcomes. A random process $X(t)$ is much the same, only each outcome of the experiment results in a function (in time e.g.) $x(t)$. The coin flipping experiment could e.g. give rise to two sinusoids with different frequencies.

The mean function $\mu_X(t)$ of a random process $X(t)$ is the expected value of the process, $E[x(t)]$ as defined by

$$E[X(t)] = \int\limits_{-\infty}^{\infty} x f_X(x;t)\,\mathrm{d}x \tag{C.1}$$

where $f_X(x;t)$ is the probability density function of $X(t)$. Basically, it is a weighted average of all the possible values that $X(t)$ can take, more probable values having a larger weight.

The variance, σ^2, of $X(t)$ is $E[(x(t) - \mu_x(t))^2]$, where $\mu_x(t) = E[x(t)]$:

$$\sigma^2 = \int\limits_{-\infty}^{\infty} (x - \mu_X)^2 f_X(x;t)\,\mathrm{d}x \tag{C.2}$$

The autocorrelation function $R_{XX}(t, t+\tau)$ of a random process $X(t)$ is defined by

$$R_{XX}(t, t+\tau) \int\limits_{-\infty}^{\infty} x_1 x_2 f_{X_1,X_2}(x_1, x_2; t, t+\tau)\,\mathrm{d}x_1 \mathrm{d}x_2 \tag{C.3}$$

where $f_{X_1,X_2}(x_1, x_2; t, t+\tau)$ is the joint probability density function of $X(t_1)$ and $X(t_2)$. It describes the relationship between two samples of a random process.

If $\mu_X(t)$ is constant, and $R_{XX}(t, t+\tau)$ is a function of only τ, $R_{XX}(\tau)$, the random process $X(t)$ is said to be wide sense stationary.

In order to calculate $\mu_X(t)$ and $R_{XX}(t, t+\tau)$, an ensemble average, an average over all realizations of a random process, is needed. If however a wide sense stationary process is ergodic, then the time average and time-average autocorrelation function for one realization of the process, given in equations C.4 and C.5, are the same as the ensemble averages and can be used instead of C.3 and C.1.

$$\overline{x(t)} = \lim_{T\to\infty} \frac{1}{T} \int\limits_{-T/2}^{T/2} x(t)\,\mathrm{d}t \tag{C.4}$$

$$\mathfrak{R}_{XX}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int\limits_{-T/2}^{T/2} x(t)x(t+\tau)\,\mathrm{d}t \tag{C.5}$$