Soft Sensor Development Using Artificial Intelligence and Statistical Multivariate Methods

Radu Platon

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Applied Science (Mechanical Engineering) at

Concordia University

Montreal, Quebec, Canada

July 2009

Your file Votre référence
ISBN: 978-0-494-63086-0
Our file Notre référence
ISBN: 978-0-494-63086-0

Abstract

Soft Sensor Development Using Artificial Intelligence and Statistical Multivariate Methods

Radu Platon

The lack of real-time measurement of certain critical product and process characteristics is a major problem in the manufacturing industry, and it can lead to an out of specification production. A soft sensor is a predictive model that uses readily available process measurements to infer variables that are impossible or difficult to obtain in real-time.

In this work, historical process data related to the black liquor recovery circuit from a Canadian kraft pulp and paper mill is used to develop soft sensor models for the black liquor solid content at the concentrator feed. Prior to modeling, irrelevant variables and observations not representative of a normal operating regime are eliminated from the dataset. For practical reasons related to modeling restrictions and soft sensor industrial implementation, is proposed that a limited number of variables be used as model inputs. Two Partial Least Squares-based selection criteria are used to select the most relevant predictors. Two different sets of ten variables are obtained and used to develop Sugeno-type fuzzy logic, neural network and Partial Least Regression models. Their predictive performance is compared in order to determine the best model configuration and input selection method.

Currently, the black liquor solid content at the concentrator feed is measured once every eight hours, by performing a laboratory analysis. The proposed soft sensor model can be used to provide a real-time value of the solid content, allowing operators to monitor the process and act timely if corrective actions are required.

Acknowledgements

I thank my supervisor, Dr. Kudret Demirli for his wholehearted support. His able guidance has been essential in the completion of this work, and without it, I would not have been able to come this far.

I thank my parents and my grandmother for their love and support. I cannot adequately express my gratitude to them.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial intelligence
ANFIS	Adaptive neural fuzzy inference system
ANN	Artificial neural network
BL	Black liquor
COEM	Controller Output Error Method
LSE	Least Square Error
MIN	Minimum
MSE	Mean square error
MVA	Multivariate data analysis
PC	Principal Component
PCA	Principal Component Analysis
PLS	Projection to Latent Structures by means of Partial Least Squares
PROD	Product
WL	White liquor

List of Symbols

$\mathbf{I}$	identity matrix
$\mathbf{V}$	covariance matrix of $\mathbf{X}$
$\mathbf{T}$	matrix containing the scores of the $\mathbf{X}$ data
$\mathbf{P}$	matrix containing the variable loadings
$\mathbf{U}$	matrix containing the scores of the $\mathbf{Y}$ data
$\mathbf{Q}$	matrix of PLS weights
$\mathbf{E}$	residuals of the PLS $\mathbf{X}$ model
$\mathbf{F}$	residuals of the PLS $\mathbf{Y}$ model
$\mathbf{H}$	regression error
μ_x	membership value of element x to the fuzzy set
P	clustering potential
r_a	cluster radius
r_b	penalty radius
$\acute{\eta}$	squash factor
Z	Sugeno model output

Zi	Sugeno rule output
σ^2	variance
σ	standard deviation
η	learning rate
α	momentum

1 INTRODUCTION

1.1 Background

A soft sensor is a predictive model that uses readily available process measurements in order to infer process state and product quality variables that are impossible or difficult to obtain in real-time. The inferential algorithm is developed using historical process data and it identifies an accurate relationship between the variable to be predicted – typically a critical variable not measured on-line – and other process variables measured on-line.

In this study, the development of a soft sensor for the pulp and paper manufacturing process is presented. The model is developed using actual industrial data from a Canadian pulp and paper mill, and it predicts the value of the black liquor solid content, a critical variable of the chemical pulping process. A detailed description of this process is presented below.

1.2 Soft sensors in the manufacturing industry

The lack of real-time measurement of certain critical product and process characteristics is a major problem in the manufacturing industry, and it can lead to an out of specification production.

In some cases, product quality variables and other key indicators of process performance can only be measured through off-line sample analyses, thus introducing discontinuity and significant delays in obtaining information relevant to process behaviour. In some cases, the available measuring devices are not as accurate as the laboratory analysis. For

some applications where accurate measuring instruments are available, financial considerations such as the costs associated with purchasing, installing and maintaining the instrument, as well as process-related issues, such as performance degradation and rigorous maintenance protocols due to harsh operating environments might prevent the implementation or practical use of these hardware instruments.

Not having access to real-time measurements of key process performance indicators can lead to potential operability problems, such as reduced efficiency of control policies and late detection of abnormal process behaviour. Additional expenses, such as labour and material cost, occur if the product has to be reprocessed due to an out of specification quality. This also increases the energy consumption of the process and its environmental impact, since more waste is produced and it will have to be disposed.

Soft sensor technology provides an accurate real-time estimate of the values of critical process variables not measured on-line, allowing operators to supervise in real-time the process behaviour, timely detect deviations from the normal operating range and take early corrective actions in order to avoid production not meeting the required specifications. In addition to process variable real-time estimation, soft sensors offer a number of other advantages over conventional measuring devices: they represent a low-cost alternative to expensive measuring instruments; they can work in parallel with other measuring devices, thus allowing the implementation of more comprehensive process monitoring systems; the soft sensor model can be implemented on existing hardware and retuned when the process operation conditions change. Soft sensors also play a significant role in more complex systems used for process optimization. In such systems, the soft sensor represents an individual component whose output is used as an input to

another component of the system. For example, it can be used to provide the value of the variable used in the control feedback loop. It can be used to detect a process disturbance that will be analysed by a fault detection and diagnosis system. It can also be used to detect measuring instrument faults: the soft sensor infers the value of a process variable, and this value is compared with the actual value recorded by the measuring device. A large difference between these two values can indicate deterioration in the instrument's measuring accuracy, and its potential malfunction [1].

Soft sensors represent a valuable tool for industrial users to monitor the operation of their processes in order to increase their productivity, energy efficiency and profitability. They can be used to solve a number of problems such as real-time variable estimation, process monitoring, on-line prediction for plant control, sensor validation and improvement of fault detection and diagnosis strategies.

1.3 Soft sensor modeling techniques

There are three main approaches for building the predictive models used as soft sensors: first principles (physical) modeling, statistical modeling and artificial intelligence modeling. Statistical and artificial intelligence methods are data-driven, since they use historical data in order to build the model. Estimates of a process variable's values are obtained on the basis of their correlation with other variables, as identified in the historical process dataset. From an industrial application perspective, this is a useful approach, since very often obtaining an accurate physical model of the plant can be difficult [1].

1.3.1 First principles models

First principles models are based on the physics of the process, such as mathematical and qualitative relationships between process variables. They can be very robust, but difficult to obtain since they require an accurate physical model of the process, which is often a very complicated task due to the complexity of the process dynamics.

1.3.2 Statistical based models

Statistical models use regression methods. Multivariate data analyses techniques, such as Principal Component Analysis (PCA) and Projection to Latent Structures by means of Partial Least Squares (PLS), identify correlations between variables and transform a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. These principal components are linear combinations of the original variables, and they account for most of the variation in the original variables, summarizing the data with little information loss [2].

1.3.3 Artificial intelligence methods

Artificial intelligence models are typically based on neural network or fuzzy logic techniques. Hybrid systems, containing a combination of these two methods, are also common. Both these techniques try to mimic human reasoning. Fuzzy logic uses gradients of true and false in order to produce an approach that recognizes more than simple true and false values; it is used to generate rules of association that can be either linguistic or numerical, in order to describe the relationship between the input and output of a system. Artificial neural network (ANN) models also describe the relationship between the input and output of a system. They represent information processing

structures consisting of a number of input and output units connected in a systematic fashion. A neural network model learns to identify patterns in the data set and predict or classify variables.

1.4 Description of the kraft pulping process

Pulp and paper mills use wood chips as the raw material for producing the pulp that will be processed into paper. Inside the wood, the fibers used for papermaking are bound together by an organic polymer called lignin. Lignin is a natural component of plants, together with cellulose and hemicellulose, but it is undesirable in paper, leading to yellowing and rapid degradation. The objective of pulping is to dissolve away the lignin and leave most of the cellulose and hemicellulose in the form of intact fibers. The kraft pulping process is a method for chemically dissolving the lignin by treating wood chips at elevated temperature and pressure with an aqueous solution of sodium hydroxide (NaOH) and sodium sulphide (Na_2S), known as "white liquor".

Woodchips are fed into large vessels called digesters, and then impregnated with the white liquor. At high pressure and temperature, delignification is achieved after a few hours. This process is also known as "digesting" or "cooking". The next step involves separating the cooking liquor from the pulp. The wood pulp containing the papermaking fibers proceeds through various stages of washing, and possibly bleaching, after which is pressed and dried into the finished paper product. The mixture of white liquor and organics, containing the used process chemicals and organic materials (mainly the dissolved lignin) is known as "black liquor". The solid content of the black liquor typically ranges from 13% to 20%, and after leaving the digester is concentrated to about

65% solids through several stages of evaporation before being burned in the recovery boiler. Combustion of the organics dissolved in the black liquor provides heat for generating steam that can be used in a turbine to produce electricity, or in different stages of the process. Inorganic chemicals present in the black liquor collect as a molten smelt at the bottom of the furnace. The smelt is dissolved into water, and then transferred to a causticizing tank where quicklime (calcium oxide) is added to convert the solution back to white liquor for return to the digester system. A lime mud precipitates from the causticizing tank, after which it is calcined in a lime kiln to regenerate quicklime.

In kraft pulping, the process of recovering the inorganic cooking materials from the black liquor, and their regeneration into fresh pulping chemicals, as well the production of energy via burning of the organic materials is known as the chemical recovery process. A simplified diagram of the paper making process involving kraft pulping is shown in Figure 1-1. The white and black liquor circuits are denoted by the WL and BL abbreviations, respectively.

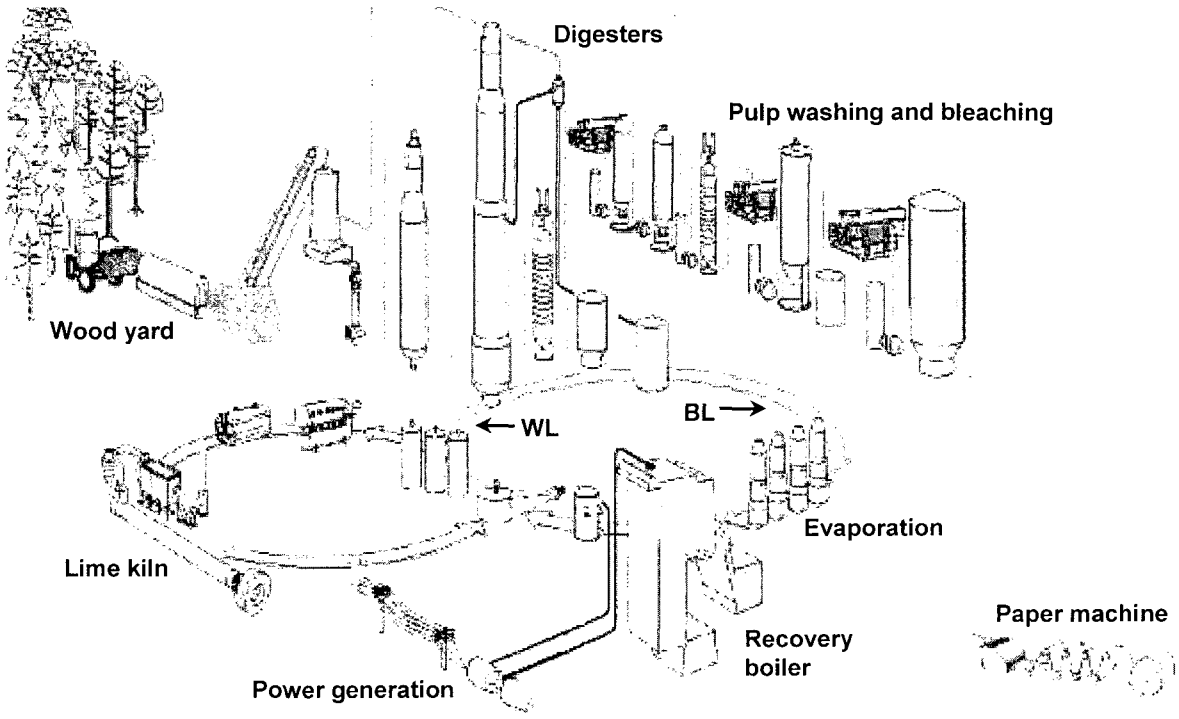


Figure 1-1 Simplified diagram of the kraft pulp and paper making process

1.5 Plant description

This study presents the development of a soft sensor for the chemical recovery process of a Canadian kraft mill. The mill daily production rate is over 1,000 tons. The production lines include both continuous and batch digesters, washing and bleaching lines and two paper machines. The steam network includes three recovery furnaces that provide over 50% of the steam mill production, and two boilers supplying the remaining steam requirements. The black liquor recovery circuit is composed of two six-effect evaporators and two concentrators. A diagram of a six-effect evaporator is shown in Figure 1-2: the black liquor is fed at the inlet of effect 6, and it is evaporated in the following effects; the concentrated black liquor exits the evaporator at effect 1.

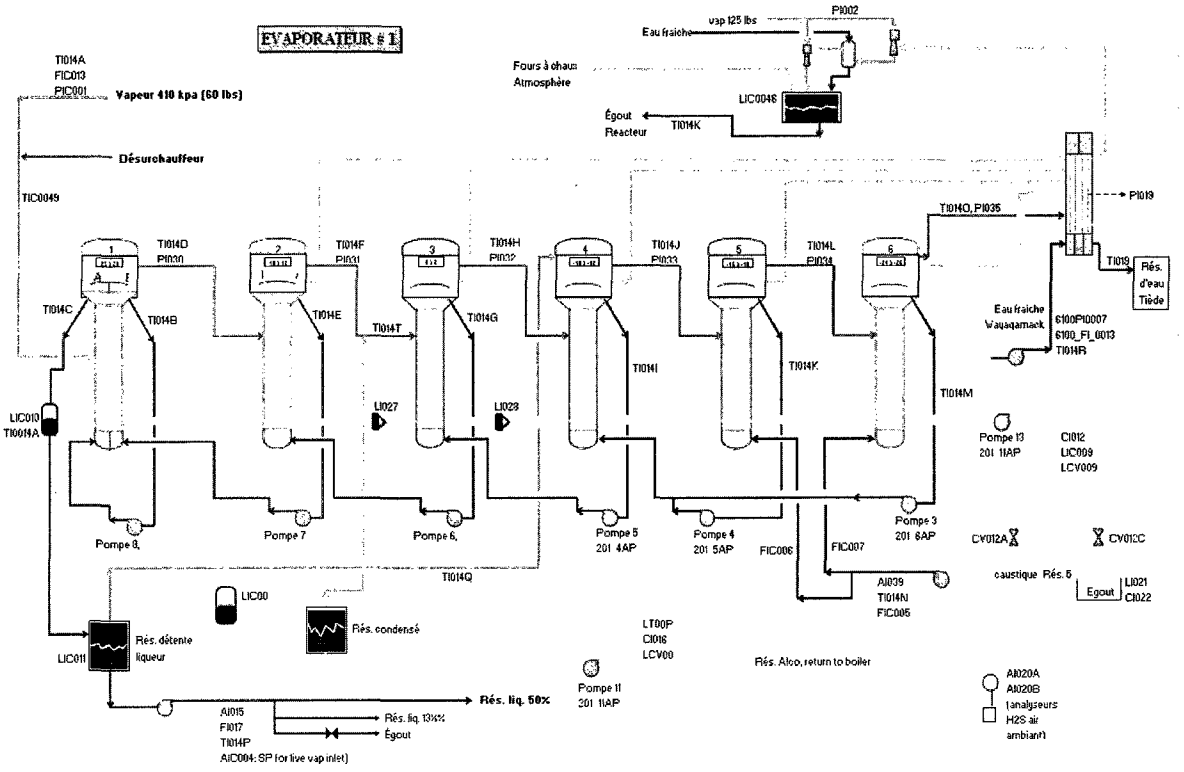


Figure 1-2 Simplified diagram of a multiple-effect evaporator

After leaving the digesters, the black liquor has a concentration lower than 15% solid content. Water from the black liquor is removed using two lines of multiple-effect evaporators operating in parallel. This operation increases the solid content to approximately 50%. The black liquor exiting the evaporators is collected in storage tanks and its solid content is increased to about 65%, by further evaporating water in two concentrators operating in series. Figure 1-3 shows a schematic of the plant's black liquor concentration process.

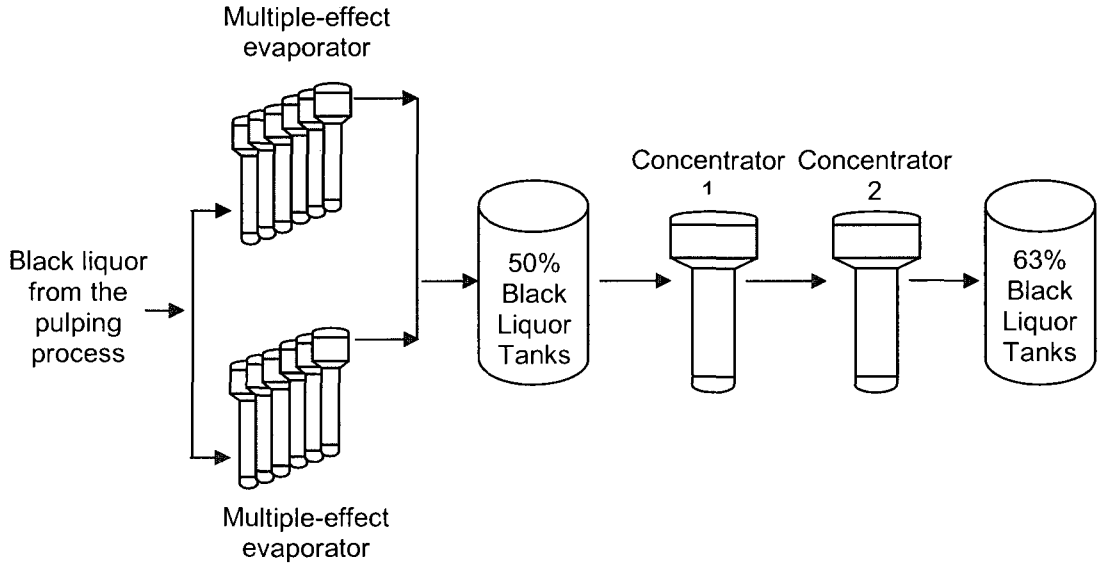


Figure 1-3 Schematic of the plant black liquor concentration process

Sodium carbonate and sodium sulfate salts (Na_2CO_3, Na_2SO_4) are the major inorganic compounds in black liquor, and they represent a significant cause of scaling in black liquor evaporators and concentrators. In the case of concentrators, if the solid content is below a critical threshold, the sodium salts crystallise on heat transfer surfaces, leading to a loss in concentrator efficiency. A solid content above the critical thresholds ensures that enough solids are present to act as seed crystals to promote additional salt precipitation in the black liquor solution rather than deposition on heat transfer surfaces. A common method used for minimizing this scale deposition is to increase the solid content at the concentrator inlet by recirculation of concentrated black liquor.

1.6 Problem definition and motivation

At the plant, salt deposits on the concentrator heat exchange areas occur when the solid content drops below 49%. The black liquor solid concentration at the concentrator feed is not measured in real-time. It is measured once per work shift (eight hours), by manually sampling the black liquor and performing a chemical analysis using the beaumé degree technique. This method relates the black liquor density to its solid content. The lack of real-time measurement of the black liquor solid content at the concentrator inlet delays the application of corrective actions required to avoid scaling, and a decrease in concentrator efficiency. Hardware devices that provide a real-time measure of the solid content exist, but they require a strict maintenance schedule that is not always respected. Also, the cost associated with their purchase and maintenance could prevent their implementation. At the plant, no measuring instrument is used to provide a real-time value of the black solid content at the concentrator feed. The appropriate corrective actions required in order to increase the solid content are well known at the mill, but their timely application is prevented by the lack of solid content real-time measurement.

Considering this major shortcoming, the objective is to develop a soft sensor model that infers, in real-time, the solid content of the black liquor at the concentrator inlet by using real-time measurements of other process variables. This soft sensor provides an accurate real-time estimation of the solid content, allowing operators to monitor the solid content value and respond as soon as it drops below the critical threshold. This prevents an increased steam consumption caused by concentrator scaling and improves the productivity of black liquor circuit, since the frequency of shutdowns for cleaning purposes is reduced.

1.7 Proposed methodology

Soft sensor models are developed in order to infer the value of the black liquor solid content at the concentrator inlet. Given the complexity of the plant constraints and dynamics, building an accurate physical model of the black liquor recovery circuit is a difficult and time-consuming task. Therefore, only multivariate statistical and artificial intelligence methods are used. Measurements related to the black liquor chemical recovery circuit are collected from the mill: the historical process dataset contains hourly measurements of 400 variables. A particular attention is paid to the data pre-processing step, since the presence of invalid data can result in the development of inaccurate models. Given the practical considerations related to a soft sensor industrial application, as well as the limitations imposed by managing a model with a very large number of inputs, it is proposed that a reduced number of variables be used as inputs to the predictive model. A predictor set of 10 variables is used to develop the models. These input variables have to be carefully selected, as they could have a significant impact on the model's performance. A Partial Least Squares (PLS) analysis is performed on the dataset containing all 400 variables, and two different criteria are used to obtain two different sets of input variables. Models are developed using each dataset, and a comparison of each model's predictive performance is carried out in order to determine the best model configuration and input selection criterion.

The objective of this study is three-fold:

- to determine if 10 input variables are sufficient for developing a soft sensor for this application

- to determine which PLS-based input selection criterion leads to the development of the most accurate model
- to determine which modeling method generates the smallest predictive error

The following methodology is used:

- data collection
- data pre-processing
- input variable selection
- development of multivariate statistical models
- development of Sugeno-type fuzzy logic models
- development of neural network models
- development of adaptive neural fuzzy inference models
- comparison of the models
- selection of the best input selection method and model configuration

1.7.1 Number of model inputs

Since the soft sensor model uses on-line process measurements to infer the desired process variable, its performance can be seriously affected by measuring instrument performance degradation. This degradation can be caused by harsh operating process conditions and lack of proper cleaning and re-calibration, for example. The instruments

providing the measurements for the process variables used as inputs to the soft sensor model have to follow a rigorous maintenance and re-calibration schedule, to ensure that accurate values are used by the inferential algorithm. This represents capital and manpower costs to the plant; therefore, given the financial considerations related to a soft sensor industrial implementation, it is desirable to use a relatively low number of process variables as model inputs.

The number of inputs can represent a challenge from a modeling perspective, since managing a model with a large number of inputs, and therefore a large number of model parameters, can become cumbersome and make the model impractical to use. Also, in an industrial context, data-driven models have to be re-computed if process changes occur, as the training data used originally for developing the model is no longer representative of the current operating conditions; using few inputs can reduce the modeling time.

On the other hand, using a small number of input variables might compromise the soft sensor performance. When dealing with large data sets, it is impractical to use all the variables as inputs, especially in the context of a soft sensor industrial application; therefore, it is important that a trade-off be achieved in terms of the number of inputs and the model's predictive accuracy. In this study, a 10-variable input dataset is proposed to develop the models, and one of the objectives is to determine if this number of inputs is sufficient to attain a proper predictive performance.

1.7.2 Input selection

Selection of input variables is an essential task in ensuring superior performance of the predictive model. The PLS method is used to identify, among all 400 variables collected from the plant, the 10 most relevant variables to be used as soft sensor inputs. Two selection criteria are used:

- the variable's ability to simultaneously model the input dataset and to predict the output
- the variable's ability to solely predict the output

Two different input datasets are obtained and predictive models using each dataset are developed. The performance of these models is compared in order to determine the input selection criteria leading to the best predictive accuracy.

1.7.3 Model performance criteria

The model performance is judged on a modeling and validation predictive error basis. The validation error is computed on a set of data not used during the modeling process. As the model parameters are calculated to specifically fit the modeling data, it is important to evaluate the model performance on "unseen" data. An industrial soft sensor uses real-time measurements as inputs to the inferential model; since these values are not used during modeling, a superior validation performance is critical. Models displaying the best modeling and validation error trade-off are considered as most accurate.

1.7.4 Data pre-processing

Irrelevant variables are identified and removed from the dataset, and data filtering techniques are used to pre-process each variable in order to identify and eliminate measurements related to abnormal variable operation, such as equipment cleaning or malfunction, for example. The principal component analysis is used to perform a global analysis of the process, in order to detect deviations from the normal operating range. The variables and measurements related to these deviations are investigated, and eliminated if necessary.

1.7.5 Process knowledge

Process knowledge is a major requirement for a successful soft sensor development project [3]. Process knowledge is critical when deciding if outliers identified in the data pre-processing step represent deviations from a normal process operating regime and therefore should be eliminated from the modeling dataset. Process knowledge is also essential when deciding upon the significance of the variables to be used as predictors. The statistical analysis determines the degree of significance of a variable by looking at the amount of variability that it contains. Sometimes, the variability might be artificially introduced, and is not necessarily representative of a normal process behaviour: frequent equipment shutdowns and start-ups, measuring instrument malfunction, frequent communication problems between the instrument and data historian and irregular updating of manual logs are examples of such occurrences. The statistical analysis might determine that these variables have a significant impact on the prediction, but process knowledge is required to confirm it, since the variables might be important from a statistical point of view, and not necessarily from a process point of view. During this

study, close collaboration with plant personnel is maintained in order to ensure that their process expertise is properly used to develop the model.

1.8 Literature survey

Artificial intelligence and multivariate data modeling techniques have found extensive applications in various fields of engineering and industrial applications, such as soft sensors, control, process optimization and fault detection and diagnosis. The objective of this literature survey is to identify soft sensor industrial applications, as well as the modeling methods used. The survey does not include only stand-alone soft sensor applications. It is extended to more complex systems in which the soft sensor represents one individual component whose output is used as an input by another component, such as control loops and fault identification and diagnosis systems. The literature survey is not limited only to the pulp and paper industry, it includes different manufacturing sectors. Some of the soft sensor applications described in these publications are already implemented on-line in industrial installations, and some of them have been successfully tested off-line using actual process data.

Ahvenlampi *et al* [4] present a combination of a Takagi-Sugeno fuzzy logic system identification and a neural network-based model to predict the Kappa number in a pulp and paper manufacturing process. The modeling historical data is obtained from an industrial continuous wood chip digesters. The model is developed using a combination of the Takagi-Sugeno clustering method for system identification and the Kohonen self-organized map neural network for classification purposes. This model is tested off-line

using data from a continuous digester, and proves to be effective for variable monitoring and prediction purposes.

Abonyi *et al* [5] use a Takagi-Sugeno fuzzy model to control the temperature of a batch polystyrene batch reactor. The Controller Output Error Method (COEM) is used for on-line tuning of a fuzzy controller. The Takagi-Sugeno fuzzy model combined with the COEM method out-performs conventional control solutions.

Bonissone and Goebel [6] use PCA and an adaptive neural fuzzy inference system (ANFIS) to build a break tendency indicator that predicts time-to-break margins in paper machines. The principal components identified by the PCA are used as inputs to the ANFIS model. When tested with actual industrial data, this system generates an accurate break tendency indicator with enough lead time to help in the overall control of the paper making cycle by minimizing down time and improving productivity.

Merikoski *et al* [7] use the ANFIS method to develop a soft sensor for viscosity in the rubber mixing process. Since in the rubber mixing process the viscosity cannot be measured in real-time, the need for soft sensor development is explained. The predictive performance of the proposed model is high enough so the soft sensor can be used in the control strategy of the process.

Wold and Keitaneh-Wold [8] discuss the benefits of applying multivariate data analysis techniques in the pulp and paper industry, mainly for process monitoring, fault detection and soft sensor applications. The benefits of using these techniques to improve process operational stability, decrease variability in product quality, improved efficiency in the

use of raw materials and obtain superior understanding of the overall process behaviour are highlighted.

Engin *et al* [9] model a nonlinear coupled-tank liquid-level system using the fuzzy Sugeno and ANFIS modeling techniques. An input-output dataset is used to represent the changes in the control valve's adjustments for this tank system. This modelling method proves to be superior to the conventional mathematical model when used in the control strategy of the coupled tank system.

Macías-Hernández *et al* [10] propose a method based on Takagi-Sugeno fuzzy models for predicting the properties of the crude oil distillation side streams. Instrument data and laboratory analysis from a crude unit operation are used in this study in order to predict temperature profiles of the crude oil streams. The performance of this model is superior to that of statistical-based models. The soft sensor is implemented on-line at an oil refinery in Spain.

Dayal *et al* [11] investigate the use of PLS and neural networks to build empirical models for Kappa number using historical data from a Kamyr wood chip digester. The Kappa number is a measure of the pulp quality. The neural network models perform slightly better than the PLS models; however, no insight into the process could be obtained from the neural network model, while the PLS model can be used to improve process understanding. The PLS model identifies variables most responsible for the variation in Kappa number, and these variables are used as inputs to the models.

Aminian and Shahhosseini [12] use a feedforward backpropagation neural network model for the prediction of crude oil fouling behaviour in preheat exchangers of crude

distillation units. The predictive performance of the NN model proves to be superior to that of conventional models used to evaluate fouling.

Radhakrishnan *et al* [13] present a neural-network based model to predict the rate of the fouling and the decrease in heat transfer efficiency in heat exchangers from the crude oil preheat train in a petroleum refinery. A PCA analysis is used to identify and remove outliers, as well as to reduce the number of predictors: highly correlated variables are identified, and some of these variables are eliminated from the modeling dataset. A methodology for developing a preventive maintenance scheduling tool using this soft sensor is also proposed.

Ahmed *et al* [14] propose an empirical model for the prediction of the melt index during grade change operations in a high density polyethylene plant. The model is developed using a recursive partial least square scheme combined with model output bias updating. In this work two different schemes have been proposed. The prediction accuracy is measured using the root mean square error. A procedure for minimizing the number of iterations is also presented.

Qin *et al* [15] propose a soft sensor to predict boiler emissions that uses a PCA model to select and validate the sensor inputs. A neural network and a linear regression model on the principal components (PCR) are used for prediction purposes, while PCA is used for input selection.

Bolf *et al* [16] present the development of neural network-based soft sensors for quality estimation of kerosene, a refinery crude distillation unit side product. Using temperature and flow measurements, two neural network models estimate the kerosene distillation end point and freezing point. The results show possibilities of applying soft sensors for refinery product quality estimation and inferential control as an alternative for laboratory analyzers.

Delgado *et al* [17] present a novel approach to design soft sensors for industrial applications, by identifying second-order Takagi–Sugeno–Kang fuzzy models by means of a co-evolutionary genetic algorithm and a neuro-based technique. The input variables of the fuzzy model are pre-selected by considering nonlinear relations among the input and output variables, and a co-evolutionary methodology is used to identify the fuzzy model itself. Membership functions, individual rules, rule-bases and fuzzy inference parameters are optimized using a neuro-genetic algorithm.

Keski-Säntti [18] explores the use of neural network-based modeling techniques for the production optimization of a kraft bleach plant. The industrial data is collected during normal mill operation at a Finish pulp mill. This study highlights the fact that first principle modeling is a complicated time-consuming task, which needs very deep knowledge of the process and of mathematics. Also, these kinds of models are quite restricted in use, impractical, and the updating is almost as laborious as the model-building process. Neural network models prove to be an efficient optimization method for this application.

Facco *et al* [19] propose the development of multivariate statistical soft sensors for the online estimation of product quality in an industrial batch polymerization process. It is shown that the estimation accuracy can be improved if dynamic information is included into the models, either by using lagged variables, or by averaging the observations on a moving window of fixed length.

A government report produced in 1995 [20] assesses the use of artificial intelligence (AI) tools in the industry: it identifies where and how AI-based tools may increase productivity, quality and energy efficiency in industry. The study also targets the application of these methods in five industrial sectors: iron and steel, cement, mining and metallurgy, oil and gas, and .pulp and paper. The report contains examples of AI systems implemented in the above-mentioned industrial sectors, along with their technical description. Process monitoring and diagnosis, online decision support, soft sensors, scheduling and planning, fault diagnosis and maintenance, systematic analysis of information about equipment failure, and design (generation of alternative designs) are identified as promising areas for the application of AI techniques. These techniques are shown to represent significant methods in improving the productivity, final product quality and energy efficiency of the targeted industrial sectors. Neural networks, expert systems and fuzzy logic are the main methods used in the development of the artificial intelligence-based tools presented in this study.

This literary review provides information about soft sensor technology applications in the industrial manufacturing sector and development trends. Different approaches for building the predictive model are presented: artificial intelligence techniques, statistical analysis, or a combination of both. The literature survey shows that the soft sensor

development can be improved by using a hybrid configuration containing more than one method. The literature survey also shows that PLS, Takagi-Sugeno fuzzy models and feedforward backpropagation neural networks are extensively used for soft sensor model development.

The majority of the soft sensor models presented in the reviewed literature use PCA for selecting the model inputs: the principal components are identified by the PCA, and used as inputs for feedforward backpropagation neural networks or ANFIS models. PCA models the global behaviour of the process, and it ranks variables according to the degree of which they describe the global variability of the process. PCA does not analyse the degree of importance of a set of variables for predicting another variable. The PLS method grades variables according to their significance in predicting another variable. Since the objective of a soft sensor is to estimate the value of a variable by using other available process measurements, it is important to analyse the performance of a predictive model developed using inputs selected according to their importance in predicting that variable. The selection of predictors according to a criterion determined by a PLS analysis is found in one reviewed publication [11]; however, only one input selection method is studied. A comparison of the predictive performance of different Sugeno-type model configurations – first order, second order, different implication functions – was not found in the reviewed literature, and neither was the development of Sugeno-type fuzzy models using inputs selected according to criteria determined by a PLS analysis.

In the study presented in this work, an extensive predictive performance comparative study of different configurations of Sugeno-type fuzzy, neural networks and PLS models

is carried out. Two different PLS-based variable significance criteria are used to identify relevant predictors. The models are built using two different input sets, in order to determine which input selection method leads to the development of more accurate soft sensors.

1.9 Thesis outline

Chapters 2, 3, and 4 contain, respectively, a review of the statistical multivariate, Sugeno-type fuzzy logic, feedforward backpropagation neural network and adaptive neuro-fuzzy modeling techniques used in this study. Chapter 5 contains a description of the procedure employed for collecting and pre-processing the industrial data used for developing the models. The methodology used to select the soft sensor inputs is presented in chapter 6.

The development of the PLS, fuzzy logic, neural network and ANFIS models is presented in chapter 7. Chapter 8 contains a brief analysis of the effects of data scarcity on the validation errors of Sugeno fuzzy models. Finally, the summary, conclusions and future work are presented in chapter 9.

2 MULTIVARIATE DATA ANALYSIS METHODS

The multivariate data analysis (MVDA) techniques used in this study are the Principal Components Analysis (PCA) and Projection to Latent Structures by means of Partial Least Regression (PLS). These techniques are called projection methods, since they return a compact representation of the original data by projecting it onto lower-dimensional planes. The coordinates of the points on these planes provide a compressed representation of the data [2].

2.1 Review of the PCA method

PCA is a multivariate projection method designed to extract and display the systematic variation in a data matrix, and it is used in this study to model the process in order to identify data representing deviations from the normal process operating range. PCA identifies correlations between variables, and expresses the data in such a way as to highlight similarities and differences. Once these patterns in the data are identified, the variation in the data set is summarized using new variables, called principal components (PCs), which are particular linear combinations of the original variables. Each principal component approximates a data grouping as well as possible in the least squares sense. The components are extracted in decreasing order of importance so that the first PC reflects the greatest source of variance in the original data, and each succeeding component accounts for as much of the remaining variability as possible. PCA reduces the dimensionality of the data set by extracting the smallest number of PCs that account for most of the variation in the original multivariate data and summarize the data with little loss of information. The principal components are orthogonal to one another. By

projecting the observations on the PC lines, coordinate values in the model plane are obtained. These values are known as scores, and they summarize the information in the original variables that is relevant of the data variation [2].

Figure 2-1 illustrates a geometrical representation of a two-principal component model for a three-dimensional dataset: the first principal component (PC1) represents the most variation in the dataset, while the second component (PC2) explains the remaining variation; the scores represent the projection of the original observations on the plane formed by the principal components.

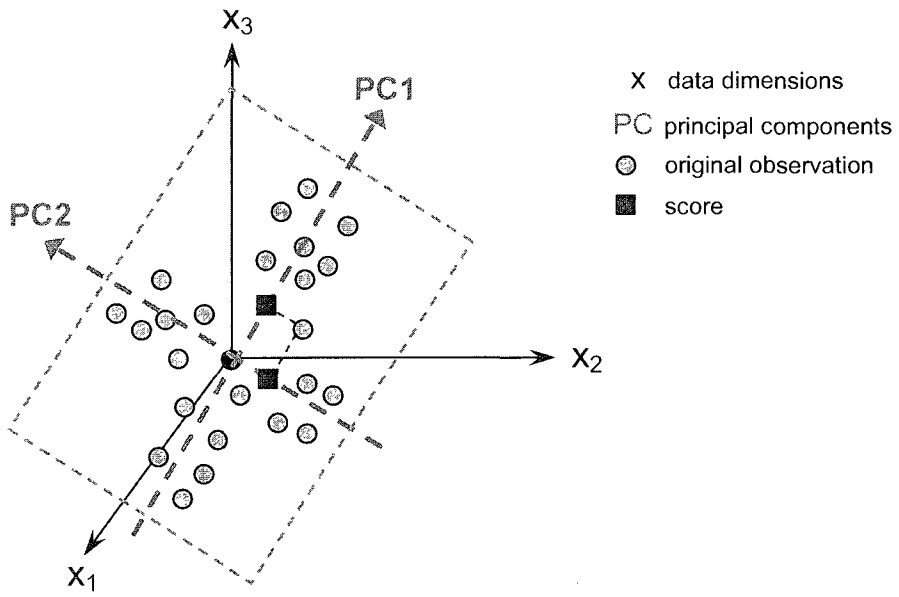


Figure 2-1Geometrical representation of the principal components

2.1.1 Mathematical representation of PCA

The algebraic solution to PCA using the covariance method is presented in this section. The first step in computing the PCA model is the scaling and mean-centering of the data. Scaling involves equalizing the variance of each variable in order to equalize their influence in the data set. Most common technique is the unit variance scaling, where each variable has an equal, unit variance. Mean-centering corresponds to a re-positioning of the data set coordinate system, such that the data average point is located at the origin; it is achieved by calculating and then subtracting from the each data point the average value of its corresponding variable. The unit variance scaling and mean-centering are essential to improve the interpretability of the model. Figure 2-2 illustrates the effect of unit variance and scaling of a set of variables: it is seen that all variables will have the same "length" and a common origin [2].

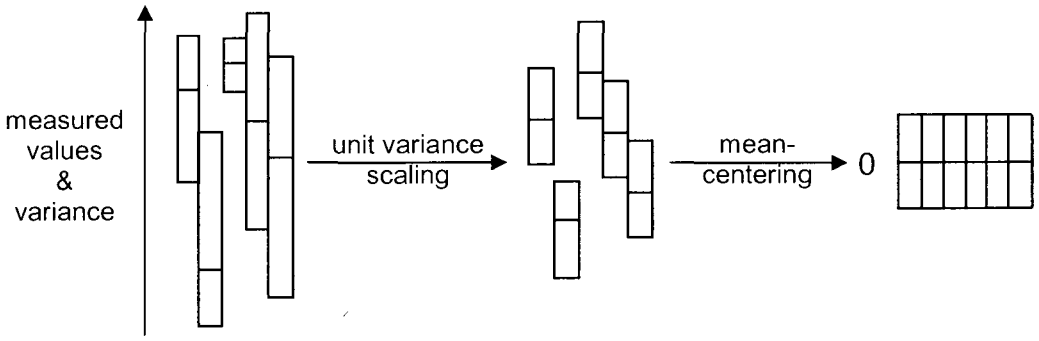


Figure 2-2 The effect of unit variance scaling and mean-centering

Let **X** be the n x p matrix containing the scaled and mean-centered data. Each column of the matrix **X** represents a variable. The objective is to obtain a vector $\boldsymbol{a}$, which is a n x 1 column vector of projection weights (unknown at this point) that result in the largest variance when the data X are projected along $\boldsymbol{a}$. This vector will represent a principal component, which explains the variance in the data.

The projection of any particular data vector is a linear combination that can be expressed as

$$\boldsymbol{a}^{T}\mathbf{X} = \sum_{j=1}^{p} \boldsymbol{a}_j \mathbf{x}_j \qquad (2.1)$$

where:

p = number of columns (variables) in the **X** data

$\mathbf{x}_j$ = **X** variable

$\boldsymbol{a}_j$ = vector resulting in the largest variance when $\mathbf{x}_j$ is projected along $\boldsymbol{a}_j$

$\boldsymbol{a}^{T}$ = transpose of $\boldsymbol{a}$

The variance is a measure of variability, defined as the average of the squared differences between the mean and the individual data values:

$$\sigma^2 = \sum_{i=1}^{p} \frac{(\mathbf{x}(i) - \bar{\mathbf{x}})^2}{n} \qquad (2.2)$$

where:

σ^2 = variance

x(i) = individual data values of variable **x**

$\bar{\mathbf{x}}$ = mean of variable **x**

p = number of **x** values

n = number of individual data values

Let $\mathbf{X}\boldsymbol{a}$ be a column vector containing the projected values onto $\boldsymbol{a}$ of all data vectors in **X**. The variance along $\boldsymbol{a}$ is calculated by the expression

$$\sigma_a^2 = (\mathbf{X}\boldsymbol{a})^T(\mathbf{X}\boldsymbol{a}) \qquad (2.3)$$

$$= \boldsymbol{a}^T\mathbf{X}^T\mathbf{X}\boldsymbol{a}$$

$$= \boldsymbol{a}^T\mathbf{V}\boldsymbol{a},$$

where **V** = covariance matrix of **X**

The covariance measures the degree of the linear relationship between variables. A positive value indicates positively correlated data, and a negative value denotes negatively correlated data. Diagonalizing the covariance matrix ensures that redundancy, measured by the absolute magnitude of the covariance, is minimized; it also maximizes the signal, measured by the variance.

The objective is to maximize σ_a^2 in order to maximize the variability of the data when projected onto the principal components.

Let $\boldsymbol{a}^T\boldsymbol{a} = 1$ be an optimization constraint. With this normalization constraint, the optimization problem can be re-written as maximizing the quantity given the expression

$$\sigma_a^2 = \boldsymbol{a}^T\mathbf{V}\boldsymbol{a} - \lambda(\boldsymbol{a}^T\boldsymbol{a} - 1) \qquad (2.4)$$

where λ is a Lagrange multiplier.

Differentiating with respect to $\boldsymbol{a}$ reduces Equation 2.4 to the eigenvalue form of

$$(\mathbf{V} - \lambda\,\mathbf{I})\boldsymbol{a} = 0 \qquad (2.5)$$

where:

$\mathbf{I} = p \times p$ identity matrix

p = number of columns (variables) in the **X** data

The first principal component $\boldsymbol{a}$ is the eigenvector associated with the largest eigenvalue of the covariance matrix **V**, and it has the largest projected variance: it explains most of the variability in the X data. The second principal component has a direction orthogonal to the first component, and is the eigenvector corresponding to the second largest eigenvalue of **V**, and so on for each subsequent component [21].

The eigenvalues indicate the amount of variance explained by each principal component. The eigenvectors contain the variable loadings, which indicate the influence of each variable in the linear combination forming each principal component.

2.1.2 Detection of process deviations

PCA can be used for detecting process deviations from the normal operating range determined by the statistical analysis. The Hotteling's T^2 plot summarizes all process variables and all model dimensions (the principal components), by calculating the distance from model origin for each observation. It is used for detecting process deviations, since it provides, on a single plot, a global picture of the process behavior. Figure 2-3 shows a Hotteling's T2 plot for a process displaying deviations from the normal operating range: the spikes seen on this plot correspond to process data related to an operation outside the normal range.

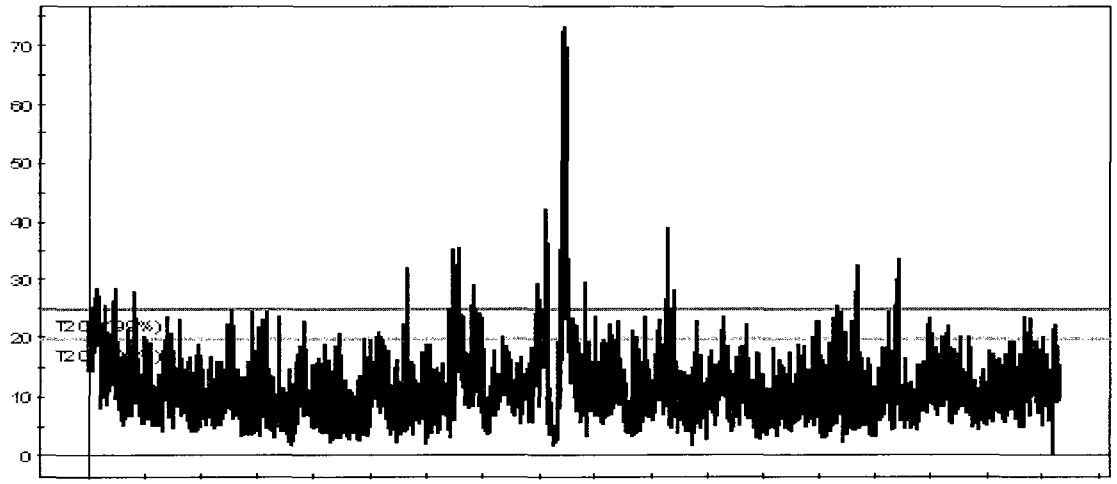


Figure 2-3 Hotteling's T2 plot

2.2 Review of the PLS method

The Projections to Latent Structures by means of Partial Least Squares (PLS) is used to build a predictive model for the black liquor concentration. PLS can be considered a regression extension of PCA, since it fits two "PCA-like" models at the same time by

modeling the input data X and the output data Y, and simultaneously aligning these models to represent the association between X and Y. The PLS objectives are to model X and Y, and to predict Y from X. In the X space, the components represent lines that approximate the data groupings and provide a good correlation with the Y values. As is the case for PCA, the components are extracted in decreasing order of importance, so each component improves the description of the X data, while providing a good correlation with the Y left unexplained by previous components. In a PLS model, there are two sets of score vectors, one for X and one for Y. The scores represent the projections of the observations on the X and Y component planes, respectively [2].

2.2.1 Mathematical representation of PLS

The same pre-processing steps of unit variance scaling and mean-centering described for the PCA analysis are required before computing the PLS model. The algebraic solution to PLS using the NIPALS (Nonlinear Iterative Partial Least Squares) algorithm is presented in this section.

The models of the X and Y data as represented by the PLS model are given by the expression

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{T} + \mathbf{E} \quad \text{and} \quad \mathbf{Y} = \mathbf{U}\mathbf{Q}^{T} + \mathbf{F} \qquad (2.6)$$

where:

$\mathbf{T}$ = matrix containing the scores of the $\mathbf{X}$ data

$\mathbf{P}$ = loadings showing the influence of the $\mathbf{X}$ variables in the principal components

$\mathbf{TP}^T$ = principal components of the **X** space

U = matrix containing the scores of the **Y** data

Q = matrix of weights expressing the correlation between **Y** and **T**

$\mathbf{UQ}^T$ = principal components of the **Y** space

E, F = residuals of the **X** and **Y** models, respectively

The columns of the matrix **T** containing the **X** scores are updated at each algorithm iteration as follows

$$\mathbf{t}_i = \mathbf{X}_i\mathbf{w}_{i+1} \qquad (2.7)$$

where:

$\mathbf{t}_i$ = columns of the matrix **T** containing the **X** scores

$\mathbf{X}_i$ = X variables

w = the weights expressing the correlation between **X** and **U**

The relationship between X and Y is given by the expression

$$\mathbf{Y}_{estimated} = \text{sum}(\mathbf{t}_i\boldsymbol{b}_i) + \mathbf{H} \qquad (2.8)$$

where:

$\mathbf{Y}_{estimated}$ = predictive output of the PLS model

$\boldsymbol{b}$ = regression coefficient used for the **Y** prediction

H = model residuals

The weights **w** expressing the correlation between **X** and matrix **U** containing the scores of the **Y** data are calculated as follows

$$\mathbf{w}_i = \mathbf{X}^T \mathbf{u}_i / \|\mathbf{X}^T \mathbf{u}_i\| \qquad (2.9)$$

where $\|\mathbf{X}^T \mathbf{u}_i\|$ = norm of $\mathbf{X}^T \mathbf{u}_i$

For the first iteration, $\mathbf{u}_i$ can be chosen as a column of **Y**. The weights expressing the correlation between **Y** and **T** is calculated using the **X** and **Y** scores, contained in the **T** and **U** matrix, respectively, by the expression

$$\mathbf{q}_i = \mathbf{u}_i^T \mathbf{t}_i / \| \mathbf{u}_i^T \mathbf{t}_i\| \qquad (2.10)$$

where:

q = columns of the matrix **Q** of weights expressing the correlation between **Y** and **T**

$\| \mathbf{u}_1^T \mathbf{t}_1\|$ = norm of $\mathbf{u}_1^T \mathbf{t}_1$

After the first iteration, the scores of the **Y** data are calculated using the expression

$$\mathbf{u}_i = \mathbf{Y}\mathbf{q}_i \qquad (2.11)$$

where:

$\mathbf{u}_i$ = columns of the **Y** score matrix **U**

q = columns of the matrix **Q** of weights expressing the correlation between **Y** and **T**

The regression coefficient ***b*** can now be calculated as

$$\boldsymbol{b} = \mathbf{u}_i^T \mathbf{t}_i \, (\mathbf{t}_i^T \mathbf{t}_i)^{-1} \qquad (2.12)$$

The previous calculations determine the scores and loadings for the first **X** and **Y** principal components. The residuals for the **X** and **Y** models are then calculated:

$$\mathbf{E}_i = \mathbf{X} - \mathbf{t}_i \mathbf{p}_i^T \quad \text{and} \quad \mathbf{F}_i = \mathbf{Y} - \mathbf{u}_i \mathbf{q}_i^T \qquad (2.13)$$

where:

X = residual of the **X** model

Y = residual of the **X** model

After each iteration, the entire procedure is repeated by replacing X and Y with their residuals until the algorithm converges [22].

Two different sets of weights are computed by the PLS model. The Y weights, $\mathbf{q}_i$, stored in the **Q** matrix indicate how the Y variable is summarized by the score vector **U**. The **W** matrix contains the X weights $\mathbf{w}_i$, which show the linear combinations forming the score vector **T**. They are optimized so as to maximize the covariance between **T** and **U**, thereby indirectly **T** and Y. X variables that are highly correlated to the Y variable have high weights $\mathbf{w}_i$. These two sets of weights provide information about the quantitative relation between X and Y.

2.2.2 PLS regression coefficients

The PLS models the Y variable using the X variables as follows

$$\mathbf{Y} = \mathbf{XB} + \mathbf{H} \qquad (2.14)$$

where:

B = matrix with regression coefficients

H = regression error

The regression coefficients are calculated using the X weights:

$$\mathbf{B} = \mathbf{W}(\mathbf{P}^T\mathbf{W})^1\mathbf{Q}^T \qquad (2.15)$$

where:

W = matrix contains the X weights $\mathbf{w}_i$

P = matrix containing the loadings of the **X** variables in the principal components

Q = matrix containing the Y weights $\mathbf{q}_i$

The model output is a linear combination of the X variables:

$$\mathbf{Y}_{estimated} = \mathbf{b}_1\mathbf{X}_1 + \mathbf{b}_2\mathbf{X}_2 + \ldots\ldots + \mathbf{b}_n\mathbf{X}_n + k \qquad (2.16)$$

where:

$\mathbf{Y}_{estimated}$ = PLS model output

b = regression coefficient

X = X variable

k = numerical constant

2.2.3 Variable influence on projection (VIP)

The variable influence on projection (VIP) parameter summarizes the importance of the variables both to explain X and predict Y. For a given model, there is only one VIP vector summarizing all X and Y components. Each input variable has its own VIP value. The VIP value is calculated as the sum of squares of the X weights ($\mathbf{w}_i$), multiplied by the residual sum of squares of the Y component – this term indicates the amount of Y variance explained by the PLS principal components. The sum of squares of all VIP's is equal to the number of X variables; hence the average VIP would be equal to 1. X variables with VIP values greater than 1 can be considered as most relevant for explaining Y [2].

3 SUGENO-TYPE FUZZY MODELING

3.1 Introduction to fuzzy logic theory

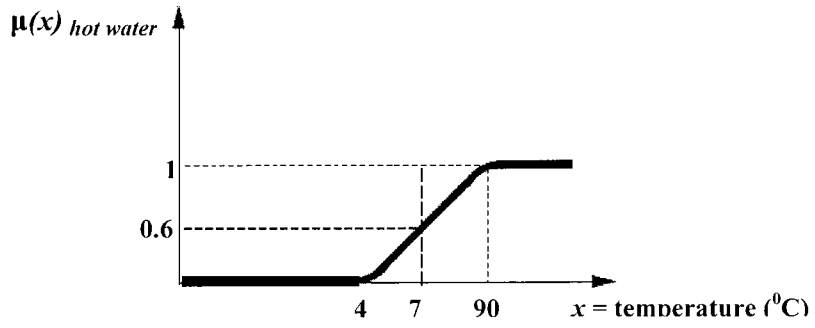
Fuzzy logic modeling proved to be a powerful tool for handling imprecision and vagueness. Traditional binary logic allows only two truth values, 0 or 1, while fuzzy logic allows any intermediate values between 0 and 1 to be directly represented. Fuzzy logic was introduced by Dr. Lotfi Zadeh in 1965, as an extension from the characteristic function of binary logic to multivalued membership functions [23]. The underlying concept in the development of this new theory was that meaning in natural language is a matter of degree. For example, consider the statement "*The water is hot*". In classical logic, this proposition can be represented only as true or false, or 1 or 0, respectively; also, the term "*hot*" has to be defined in precise numbers. In reality, the understanding of the term "*hot*" is subjective and varies with context. This suggests that if the value of water temperature is given, then its degree of compatibility with the term *"hot"* should be a matter of degree between false and true, or 0 and 1, respectively. Fuzzy set theory allows for the term "*hot*" to be defined mathematically, create a "*hot water*" fuzzy set, and evaluate to what degree a given temperature belongs to this set.

3.2 Fuzzy sets

Let X be the universal set. A fuzzy set A of the universal set X is a function $\mu_A \rightarrow [0\text{-}1]$, where μ_A represents the membership function that assigns a membership value between 0 and 1 to elements x from the universal set X in the fuzzy set A. For example, a function determining the membership value to the fuzzy set *"hot water"* used previously can be

defined using mathematical terms that assign a degree of membership of a given temperature to this fuzzy set.

Figure 3-1 illustrates the membership function for the "*The water is hot*" statement, as well as the equations calculating the membership of given temperature to the *"hot water"* fuzzy set: *"hot water"* is defined on the interval [40 90]^{0}C, and temperatures lower than 40^{0}C are not compatible with the statement "*The water is hot*" – they do not belong to the *"hot water"* fuzzy set – while temperatures greater than 90^{0}C have a membership value of 1. The memberships of temperatures with any other values are calculated according to the specified formula; for example, a temperature of 70^{0}C has a membership value of 0.6 to this fuzzy set. The term $\mu(x)_{\text{hot water}}$ represents the membership value of a given temperature x to the *"hot water"* fuzzy set.



$$\mu(x)_{\text{hot water}} = \begin{cases} 1 & \text{if } x \geq 100\ ^0\text{C} \\ (x-40)/50 & \text{if } 40 \leq x < 100 \\ 0 & \text{if } x < 40\ ^0\text{C} \end{cases} \qquad (3.1)$$

Figure 3-1 Membership function of the « hot water » fuzzy set

In contrast with a fuzzy set, a conventional set based on classical logic would have allowed only two membership values: 0, or false, for temperatures equal or below 40^0C, and 1, or true, for temperatures equal or greater than 90^0C. A fuzzy set admits the possibility of partial membership, and the membership function associated with a given fuzzy set maps an input to its appropriate membership value.

3.3 Subjective and objective fuzzy knowledge - based models

Fuzzy knowledge-based modeling can be particularly useful when the relations between the inputs and outputs of a system are not exactly known, or there is no analytical model for these relations. In these cases, the only basis for modeling is expert knowledge, which is often uncertain and imprecise. Such knowledge can be represented by a set of fuzzy rules which describe the relations between the inputs and outputs of a model.

Fuzzy knowledge-based models can be divided into two groups: subjective and objective models. The information contained in subjective models is directly solicited from experts; these models attempt to mimic the reasoning process of an expert. The Mamdani-type fuzzy methodology is extensively used for developing subjective fuzzy knowledge-based models. In Mamdani models, the output variables are defined as fuzzy sets and the final output value is converted to a scalar by a defuzzication method.

The objective models are developed from a set of input and output data by using a systematic data analysis process to generate the fuzzy rules describing the information contained in the dataset. Introduced in 1985 [24], the Sugeno-type fuzzy methodology (also known as Takagi-Sugeno-Kang) is extensively used for developing objective fuzzy knowledge-based models. In Sugeno models, the output variables are scalars that are

computed using a polynomial function of input variables. The Sugeno-type fuzzy modelling proved to be a powerful tool for developing predictive models using a dataset containing input and output variables, and it is used in this study to develop soft sensor models.

3.4 Sugeno-type fuzzy rule

The general syntax of a fuzzy rule is

$$\textit{If } \boldsymbol{x} \textit{ is } \boldsymbol{A} \textbf{ then } \boldsymbol{y} \textit{ is } \boldsymbol{B} \qquad (3.2)$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are the values defined by the fuzzy sets on the ranges $\boldsymbol{X}$ and $\boldsymbol{Y}$, respectively [25]. The if-part of the rule – *if* $\boldsymbol{x}$ *is* $\boldsymbol{A}$ – is called the *antecedent*, and the then-part of the rule – **then** $\boldsymbol{y}$ *is* $\boldsymbol{B}$ – is called the *consequent*. The consequent has to appropriately describe the rule output within the fuzzy region specified by the antecedent.

In a Sugeno rule, rule consequent is expressed as a polynomial function of the inputs, and the order of the polynomial determines the order of the model [24].

A typical first-order Sugeno rule is:

$$\textit{If } \boldsymbol{x}_1 \text{ is } \boldsymbol{A}_1 \textit{ and } \boldsymbol{x}_2 \text{ is } \boldsymbol{A}_2 \ldots \text{and } \boldsymbol{x}_i \text{ is } \boldsymbol{A}_i \textit{ then } z(u) = p_1 u_1 + p_2 u_2 + \ldots + p_i u_i + k \qquad (3.3)$$

where:

i = number of input dimensions

A = rule antecedents

u = input value

z = rule output

p_i = rule consequent associated with the i^{th} input dimension

k = numerical constant

The output of a Sugeno rule is calculated simply by applying the input values to the polynomial function.

3.5 Sugeno-type fuzzy inference system

The fuzzy inference system maps input variables to output variables through a fuzzy rule base. A fuzzy rule base is the collection of rules that the model uses to properly cover the space defined by the input and output variables.

3.5.1 Rule weights

For a given input, each rule has a certain impact on determining the model result for that input; this is called the *rule weight* – also called the *firing strength* of the rule. The membership values of each input dimension indicate the degree to which each part of the antecedent is satisfied, and they are used to determine the rule weight. The calculation of the rule antecedents and input membership functions is presented in the next section.

The function that transforms the individual membership values of each input dimension into a rule weight for the multi-dimensional input is called an implication function. Two

well known fuzzy implication functions are used in this study, and they are presented below:

- the *MIN* (Minimum) implication function: it interprets the fuzzy implication as the minimum operation, and the rule weight is equal to the lowest membership value among all input dimensions

- the *PROD* (Product) implication function: it implements the fuzzy implication by the product operation, and the rule weight is equal to the product between the membership values for all input dimensions

3.5.2 Rule output aggregation

Combining the output and weight of each rule in order to calculate the model result is called aggregation. The output of each rule is multiplied by the rule weight, and the final output of the Sugeno system is the weighted rule average of all rule outputs. It is computed as the sum of products between each rule output and the corresponding rule weight, divided by the sum of weights of all rules:

$$Z = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} w_i} \qquad (3.4)$$

where

i = number of input dimensions

N = number of rules

w_i = rule weight

z_i = output of rule i

Z = Sugeno model output

The calculation of the rule consequents, required to compute the rule output, is presented in the next section.

3.6 First order Sugeno model building

In modeling from input and output data, the parameters of the Sugeno model are typically identified by using fuzzy clustering techniques which involves grouping of data into clusters of similar behaviour. These clusters are used to determine the rule antecedents, as each cluster represents a fuzzy rule. The membership value of a data point is a measure of the point's distance from the cluster center. The rule consequents for a given set of clusters are obtained using a least square estimation method, such that the error between the Sugeno model output and the training output data is minimized. The main steps required for computing a first order multiple-input single-output Sugeno-type fuzzy model are presented below.

3.6.1 Normalizing the data

Before starting the Sugeno modeling process, the input-output data is normalized on the interval [0 1]. This ensures that each variable will have the same numerical range, thus equalizing their influence in the dataset. After normalization, the minimum value of each

variable will be 0, and the maximum value will be 1. The normalization is performed as follows

$$u_{\text{normalized}} = \frac{u - \min(u)}{\max(u) - \min(u)} \qquad (3.5)$$

where:

$u_{\text{normalized}}$ = normalized value of each observation of the variable

u = actual value of the observation, before normalization

$\min(u)$ = minimum value of the variable

$\max(u)$ = maximum value of the variable

3.6.2 Identifying the rule antecedents using the subtractive clustering algorithm

The purpose of clustering is to extract natural groupings of data from a large data set, such that a concise representation of system's behaviour is produced. Clustering is a process of partitioning a set using resemblance or dissemblance measures, such as the distance between elements of the set. Fuzzy clustering methods include the fuzzy C-means clustering approach, the mountain clustering method and the subtractive clustering method.

In this study, the subtractive clustering method introduced by Chiu [26] is used to estimate the number of clusters and cluster centers in a database. The computational effort of this algorithm is independent of the dimension of the space to be clustered, thus

improving computing speed. The clustering is performed on the combined input and output spaces, and the behaviour of each cluster is modeled by a fuzzy rule. The clusters can be disjoint or partial overlapped, and they indicate groupings of data points with related behaviour; the center is the data point with the highest measure in the cluster. The clusters are projected on each dimension in the input space, and each projection forms the fuzzy rule antecedent associated with an input dimension.

The subtractive clustering method considers the data points themselves as potential locations for cluster centers. The potential of a data point to become a cluster center is considered to be a function of its distances to all data points, and the subtractive clustering algorithm calculates a measure of the potential for each data point based on the density of surrounding data points. It is assumed that each point in the data space has equal contribution towards system identification. The *cluster radius* indicates the range of influence of a cluster in the data space.

The potential P for each data point is calculated according to its location to all other data points, using the expression

$$P_i = \sum_{j=1}^{n} e^{-\alpha \left\| x_i - x_j \right\|^2} \qquad (3.6)$$

where:

$\alpha = 4/ r_a^2$

r_a = cluster radius

n = total number of data points

x_i, x_j = vectors in combined data space of input and output dimensions

P_i = potential of the i^{th} data point

The data point with the highest potential as the first cluster center is selected. The potentials of data points located near this cluster center are penalized in order to facilitate the emergence of new cluster centers. A clustering parameter, the *squash factor*, controls how close or how far clusters are formed. This parameter is used to compute a so-called penalty radius, which determines the neighbourhood of a cluster center within which the existence of other clusters is to be discouraged. The penalty radius is computed as follows

$$r_b = \acute{\eta} r_a \qquad (3.7)$$

where:

r_b = penalty radius

r_a = cluster radius

$\acute{\eta}$ = squash factor

Each time a cluster is obtained, the potential of all data points is revised according to the expression

$$P_i = P_i - P_k^* \, e^{-\beta \|x_i - x_k^*\|^2} \qquad (3.8)$$

where:

$$\beta = \frac{4}{r_b^2}$$

P_k^* = the potential of the k^{th} cluster center

i = the i^{th} data point being subtracted

x_k^* = the k^{th} cluster center

The identification of other cluster centers is carried out through the subtraction process, as the algorithm selects the data point with the highest remaining potential as the next cluster center and revises the potential of the data points. This new cluster center is selected based on its revised potential value in relationship to an upper acceptance threshold value, called the *accept ratio*, and a lower rejection threshold value, called *the reject ratio*. The accept ratio sets the new potential as a fraction of the potential of the first cluster center, above which another data point will be accepted as a cluster center. The reject ratio sets the new potential as a fraction of the potential of the first cluster center, below which a data point will be rejected as cluster center [25]. This process of acquiring a new cluster center and revising the potential of remaining points repeats until the potential of all data points falls bellow the reject ratio value.

3.6.3 Computing membership values

The clustering is performed on the combined input-output space, and each cluster represents a fuzzy rule. The membership value of a data point is a measure of the point's distance from the cluster center, and the cluster center has a membership value of 1.

For a given input point, the membership value associated with each input dimension corresponds to the distance between the point's coordinate on the input dimension and the coordinate of the cluster center on that dimension. The membership values of all data points are assigned exponentially with respect to all cluster centers as follows

$$\mu_{ij} = e^{\frac{-4}{r_a^2}\|x_i - x_j\|^2} \qquad (3.9)$$

where:

μ_{ij} = membership value of the ith input point with respect to the jth cluster coordinate in the input dimension

r_a = cluster radius

$\|x_i - x_j\|$ = Euclidean distance between the i^{th} data point and j^{th} cluster center

The membership functions can be calculated using the same radius value for all of the data dimensions, or using a different radius value for each of the data dimensions.

When using the same radius value for all data dimensions, the cluster has the same range of influence in each data dimension. This results in obtaining membership functions with

the same spread (width) for all dimensions. For example, the membership values for a two-dimensional input will be computed as follows

$$\mu(u) = e^{\left[\frac{-4}{r_a^2}(u_{x1}-C_{x1})^2 + \frac{-4}{r_a^2}(u_{x2}-C_{x2})^2\right]} \qquad (3.10)$$

where:

x_1, x_2 = input dimension 1 and 2, respectively

$\mu(u)$ = membership function of input u

u_{x1} = value in the input dimension 1

u_{x2} = value in the input dimension 2

r_a = cluster radius

C_{x1} = cluster center coordinate in input dimension 1

C_{x2} = cluster center coordinate in input dimension 2

It can be seen that the same cluster radius value r_a is used to compute the membership values associated with each input dimension.

When using a different radius for each of the data dimensions, the cluster has different ranges of influence in each dimension. This results in obtaining membership functions with a different spread for each dimension. For example, the membership values for a two-dimensional input will be computed as follows

$$\mu(u) = e^{\left[\frac{-4}{r_{a1}^2}(u_{x1}-C_{x1})^2 + \frac{-4}{r_{a2}^2}(u_{x2}-C_{x2})^2\right]} \qquad (3.11)$$

where:

$\mu(u)$ = membership function of input u

u_{x1} = value in the input dimension 1

u_{x2} = value in the input dimension 2

r_{a1} = cluster radius for input dimension 1

r_{a2} = cluster radius for input dimension 2

C_{x1} = cluster center coordinate in input dimension 1

C_{x2} = cluster center coordinate in input dimension 2

It can be seen that different cluster radii are used to compute the membership values associated with each input dimension: r_{a1} for input dimension 1 and r_{a2} for input dimension 2. Using different cluster radii can allow the model more flexibility and improve its performance, as an optimal radius can be determined for each of the data dimensions.

3.6.4 Computing the rule consequents

The rule consequents are calculated using a least square optimization method that minimizes the error between the Sugeno model output and the values of the output variable.

Combining Equations 3.3 and 3.4 – describing the output of a rule and the output of the model, respectively – the output of a first order multiple-input single-output Sugeno model is calculated as follows

$$Z = \beta_1 z_1(u) + \beta_2 z_2(u) + \ldots + \beta_j z_j(u) \qquad (3.12)$$

$$= \beta_1[p_{11} u_1 + p_{21} u_2 + \ldots + p_{i1} u_i + k_1] + \beta_2 [p_{22} u_1 + p_{22} u_2 + \ldots$$

$$+ p_{i2} u_i + k_2] + \ldots + \beta_j [p_{1j} u_1 + p_{2j} u_2 + \ldots + p_{ij} u_i + k_j]$$

where:

Z = model output

z_j = output of rule j

u_i = input dimension i

p_{11} = first consequent of the first rule, corresponding to the first input dimension

p_{21} = second consequent of the first rule, corresponding to the second input dimension

p_{ij} = i^{th} consequent of the j^{th} rule, associated with the i^{th} input dimension

k_j = last consequent of the j^{th} rule

β_j = term associated with rule j

The term β associated with rule j takes into account the weights w of all of the rules, as follows

$$\beta_j = \frac{w_j(u)}{w_1(u) + w_2(u) + \ldots + w_j(u)} \qquad (3.13)$$

For a single output Sugeno-type system, the rule base will have a number of consequents equal to the product between the number of rules and number of consequents per rule, as shown in the expression

$$N_{\text{rule base}} = j(i + 1) \qquad (3.14)$$

where:

$N_{\text{rule base}}$ = number of rule consequents in the rule base

j = number of rules

i = number of input dimensions

$i + 1$ = number of consequents per rule

The number of β terms introduced previously is equal to the number of consequents contained in the rule base. These β terms are arranged in a matrix $n \times j$, where n is the number of observations in the dataset and j is the number of rules, as in the expression

$$B = \begin{pmatrix} \beta_{11}u_{11}\ \beta 1_1 u_{21}\ ..\beta_{11}u_{i1}\ \beta_{11}\ \beta_{21}u_{11}\ \beta_{21}u_{11}\ ..\beta_{21}u_{i1}\ \beta_{21}\ ..\beta_{j1}u_{11}\ \beta_{j1}u_{21}\ ..\beta_{j1}u_{i1}\ \beta_{j1} \\ \beta_{12}u_{12}\ \beta_{12}u_{22}\ ..\beta_{12}u_{i2}\ \beta_{12}\ \beta_{22}u_{12}\ \beta_{22}u_{12}\ ..\beta_{22}u_{i2}\ \beta_{22}\ ..\beta_{j2}u_{12}\ \beta_{j2}u_{22}\ ..\beta_{j2}u_{i2}\ \beta_{j2} \\ \quad .\quad .\quad .\quad .\quad .\quad .\quad .\quad .\quad .\quad . \\ \beta_{1n}u_{1n}\ \beta_{1n}u_{2n}\ ..\beta_{1n}u_{in}\ \beta_{1n}\ \beta_{2n}u_{1n}\ \beta_{2n}u_{1n}\ ..\beta_{2n}u_{in}\ \beta_{2n}\ ..\beta_{jn}u_{1n}\ \beta_{jn}u_{2n}\ ..\beta_{jn}u_{in}\ \beta_{jn} \end{pmatrix}$$

where:

β_{jn} = the β term corresponding to the j^{th} rule and n^{th} observation

u_{in} = the input value corresponding to the i^{th} dimension and n^{th} observation

n = number of observations

If the values of the output variable and the consequents for the rule base are arranged in separate vectors, the problem can be expressed as

$$BX = Y \qquad (3.15)$$

where

B = matrix containing the β terms

X = vector containing the rule consequents

Y = vector containing the training values of the output variable

Using this notation, the calculation of the consequents becomes a least square estimation problem. The consequents are calculated such that the difference between the product BX, representing the model output, and the output variable Y is minimized.

3.6.5 Identifying the optimal model parameters

A given combination of the clustering parameters presented previously determines one model configuration. In order to determine the most accurate model, different combinations of clustering parameters have to be tested. This is typically carried out by varying the clustering parameters on the intervals shown in Table 3-1.

Clustering parameter	Cluster radius	Reject ratio	Accept ratio	Squash factor
Interval	0.1 – 1	0 – 0.9	0 – 1	0 .1– 2

Table 3-1 Intervals for varying the clustering parameters

An enumerative search is carried out trough all possible combinations of clustering parameters in order to identify the combination leading to the model with the lowest predictive error. The user selects the desired step sizes for incrementing the clustering parameters.

3.7 Second order Sugeno models

In Sugeno systems, the order of the polynomial function describing the rule consequent determines the order of a system. A typical second order single-output fuzzy rule of a Sugeno model would be:

$$If\ x_1 \text{ is } A_1 \ldots \text{and } x_i \text{ is } A_i \ then\ z(u) = p_{11}\,u_1 + p_{12}\,u_1^2 + \ldots + p_{i1}\,u_i + p_{i2}\,u_1^2 + k \quad (3.16)$$

where:

z = rule output

i = number of input dimensions

A = rule antecedents

u = input value

p_i = rule consequent associated with the i^{th} input dimension

k = numerical constant

Higher order models, compared to lower order models, can identify a system with less error for the same number of rules, or could achieve the same performance with less number of rules [27].

The procedure to compute a second order Sugeno-type model is the same as for the first-order model. The matrices used to identify the rule consequents are slightly larger, in order to accommodate the additional terms contained in the second order polynomial.

4 FEEDFORWARD BACKPROPAGATION ARTIFICIAL NEURAL NETWORK MODELING

4.1 Artificial neural networks

An artificial neural network (ANN) is an artificial intelligence-based model that mimics the behaviour of the human brain. Similar to a human neural system, an artificial neural network is an information processing structure that consists of a number of input and output units connected in a systematic fashion. Between the input and output, there may be one or more hidden layers, each consisting of a number of processing units called neurons. The connections between units lying on different layers are assigned with varying values known as weights. Input data are fed in from the input layer, and they follow all possible connection paths to reach the next layer. At each neuron, the signal suffers a transformation, and eventually it reaches the output layer. A simplified neural network configuration is shown in Figure 4-1: there are 3 input variables, and one output; the input layer contains 3 neurons, the hidden layer contains 4 neurons, and there is one output neuron.

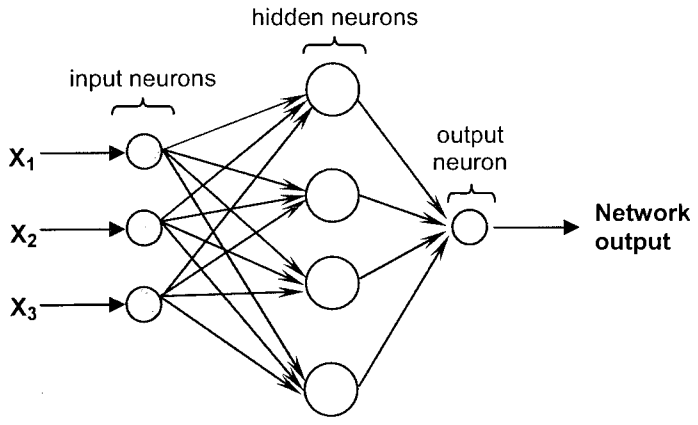


Figure 4-1 Simplified representation of an ANN

Neural networks can be adjusted, or trained, so that they learn how a particular input leads to a specific target output. The network weights are adjusted based on the comparison between the model result and the target output, until the network result matches the target. Although other procedures can be used, the modification of the weights provides the traditional method for the design of neural networks [28]. The procedure used to perform this learning process is called a learning algorithm, and it represents the function used to modify the weights in a systematic manner until the desired output is achieved. Neural networks perform very well in regression and classification problems, and it was proved that a neural network model can practically fit any function [29].

4.2 Feedforward backpropagation neural networks

A class of neural networks extensively used for soft sensor development are the feedforward networks using the backpropagation algorithm. They proved to be efficient to model nonlinear correlations between the input and output data. The input signal propagates through this type of network in a forward direction, on a layer-by-layer basis. The backpropagation method, also called the error backpropagation algorithm, is a neural network learning procedure consisting of two passes through the network layers: a forward pass and a backward pass. In the forward pass, a vector of input data is presented to the network. This data is subjected to different transformations in each layer, and the network output is calculated. During this pass, the network weights are fixed. During the backward pass, the weights are modified according to an error-correction rule, called the delta rule. The error is obtained as the difference between the desired output and the network output, and is propagated backward through the network. The objective is to

modify the weights so that the error is minimized. Standard backpropagation is a gradient descent algorithm, in which the network weights are moved along the negative of the gradient of the performance function [29].

4.2.1 Layers of neurons

Two or more neurons can be combined in a layer. A network can have multiple layers. The outputs of each layer represent the inputs to the following layer. It is common for different layers to have different numbers of neurons. The layers have different functions, according to their location in the network:

- the input layer receives the data to be modelled by the network, and the output of this layer is fed into subsequent layers, called the hidden layers
- the output layer produces the network output
- the hidden layers are the layers located between the input and output layers, and they perform most of the computations required to produce the network output. Their output represents the input to the output layer. A network can have one or more hidden layers.

Each neuron is connected to all the neurons in the following layer. The input of a neuron consists of the sum of the products between all outputs from the previous layer's neurons and the corresponding weight for each connection. Another scalar value, called bias, is added to this sum. Inside each neuron, the signal suffers a transformation. The function performing this transformation is called a transfer function.

The neuron input vector can contain input data to the network, in the case of the input layer, or the outputs of neurons from previous layers, in the case of hidden neurons. These values are summed after being multiplied with the weight of each neuron connection, and the bias is also added. Therefore, the input to the neuron becomes

$$n = w_{1,1}\, p_1 + w_{1,2}\, p_2 + \ldots. w_{1,R}\, p_R + b \qquad (4.1)$$

where:

n = neuron input

R = number of inputs

w = connection weights

p = neuron input

b = bias

The model of a single neuron is shown in Figure 4-2: the transfer function is denotes by f, and the neuron output is the scalar a.

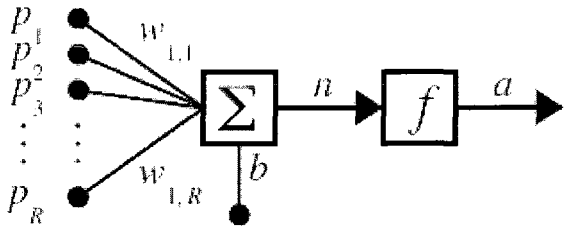


Figure 4-2 Neuron model

For the input layer, there is no transfer function. This layer receives the input data to the network. After being transformed according to Equation 4.1, the data is transmitted to the hidden layer(s).

A three-layer feedforward network is shown in Figure 4-3: a weight matrix ***W***, a bias vector ***b*** and an output vector ***a*** is associated with each layer.

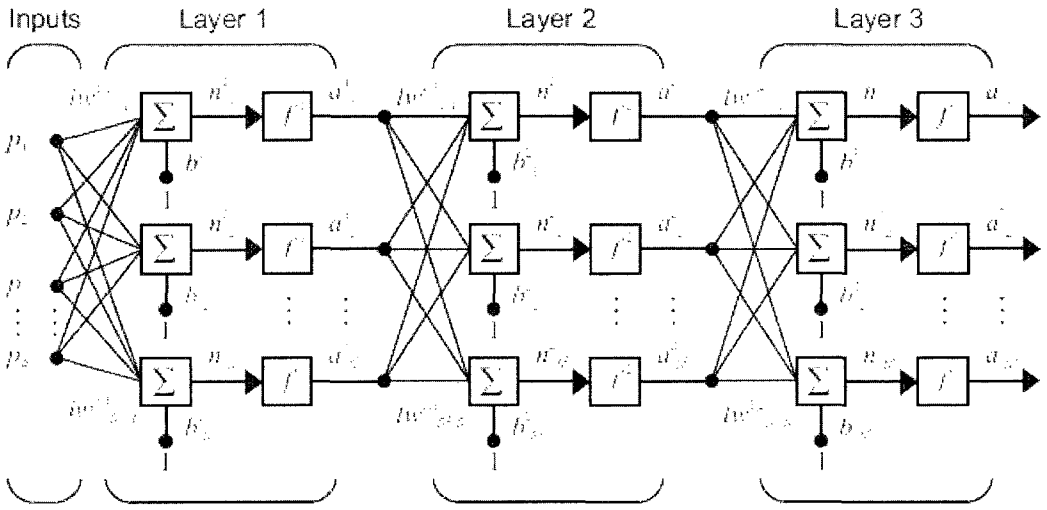


Figure 4-3 Three layer network

4.2.2 Transfer functions

The transfer functions use the neuron's inputs to calculate its output, and they are an essential component of the network's processing capabilities. The transfer functions can also be called activation functions. In backpropagation, it is critical that the derivates of the transfer functions can be calculated; the non-linear activation functions have to be continuously differentiable [28]. Backpropagation networks often use transfer functions

in the hidden layers that compress an infinite input range into a finite output range [29]. Some of these functions are presented next.

The logarithmic sigmoid function – also called the *log-sigmoid* function – uses the following formula to generate outputs between 0 and 1

$$logsig(n) = \frac{1}{1+e^{-n}} \qquad (4.2)$$

This function is represented in Figure 4-4.

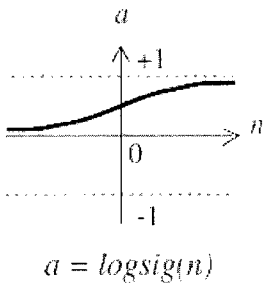


Figure 4-4 The log-sigmoid function

The hyperbolic tangent function – also called the *tan-sigmoid* function – uses the following formula to generate outputs between -1 and 1

$$tansig(n) = \frac{2}{(1+e^{-2n})-1} \qquad (4.3)$$

This function is represented in Figure 4-5.

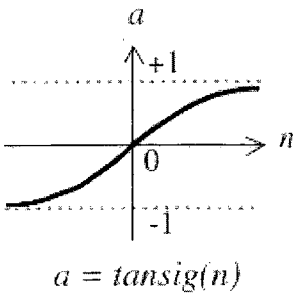


Figure 4-5 The tan-sigmoid function

If the last layer of a feedforward network has sigmoid neurons, then the network output will be limited to a small range; this is why often networks have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. The output linear activation function allows the network to produce values outside the [-1 1] interval [29].

4.2.3 Backpropagation algorithm

The backpropagation algorithm performs a gradient descent minimisation of the absolute modelling error, adjusting the network weights such as the difference between the network result and target output is minimized. The main steps of this algorithm are shown next. Initially, the weights are chosen randomly, and the neuron output is computed. The modelling error is used to adjust the weights according to the delta rule

$$\Delta W = \eta \frac{\delta E^2}{\delta W} \qquad (4.4)$$

$$W_{new} = W - \Delta W \qquad (4.5)$$

where:

ΔW = weight correction factor used to adjust the weights

E = neuron modelling error

W = weights

W_{new} = weight values after adjustment

η = learning rate parameter

$\frac{\delta E^2}{\delta W}$ = error sensitivity factor

The learning rate parameter determines the rate of network learning, and it will be presented in the next section.

The partial derivative $\frac{\delta E^2}{\delta W}$ represents the sensitivity of the error, indicating the direction of search in the weight space for the adjusted value of the weight:

- if it has a negative value, then the weight value has to be increased in order to minimize the error

- if it has a positive value, then the weight value has to be decreased in order to minimize the error

Using the chain rule, the error sensitivity factor can be written as

$$\frac{\delta E^2}{\delta W} = \frac{\delta E^2}{\delta I}\frac{\delta I}{\delta W} \qquad (4.6)$$

where I represents the neuron input.

Eventually, the complete delta can be written as

$$\Delta W_{ij} = \eta \frac{\delta E^2}{\delta I_j} I_j \qquad (4.7)$$

where:

ΔW_{ij} = correction factor for adjusting the weight between neuron i and neuron j

I_j = input to neuron j

E = neuron modelling error

η = learning rate

The gradient $\frac{\delta E^2}{\delta I_j}$ depends on whether neuron j is a hidden or an output neuron [28]:

- if neuron j is an output unit, the local gradient equals to the product of the derivative of the activation function and the error of neuron j
- if neuron j is a hidden unit, the local gradient equals to the product of the derivative of the activation function of neuron j and the weighted sum of the gradients computed for the neurons in the next layer that are connected to node j

Different error criteria can be employed in order to define convergence for the backpropagation algorithm, such as:

- if the Euclidean norm of the gradient vector reaches a certain threshold
- if the absolute rate of change in the error computed at each training iteration is sufficiently small

The backpropagation method is an iterative procedure, as the weight updating process is repeated for a number of fixed iterations, called training epochs, or until the error criteria is met.

4.2.4 Learning modes

Traditionally, backpropagation learning might be performed in two ways: sequential and batch modes of training.

In the sequential training mode, the weight updating step is performed after each observation from the training dataset is presented to the network. This learning mode is also called incremental training.

In the batch mode, all the training observations are applied to the network before the weights are updated. The gradients calculated at each observation are summed to determine the change in the weights and biases.

4.2.5 Learning rate and momentum

The learning rate controls the magnitude of change applied to the weights during the backpropagation algorithm. The learning rate is positively correlated to the weight correction factor: the smaller it is, the smaller the changes to the weights from one iteration to the next. A small learning rate leads to a more stable evolution of the weights, but it slows down the network learning process. Faster convergence can be achieved by increasing the learning rate value. This will result in increasing the changes in the weights, but it can lead to an unstable network, as the weights display an oscillatory behaviour during the training. The learning rate can be increased without compromising

the network stability by introducing in the backpropagation algorithm a training parameter called momentum. The momentum has a stabilizing effect in the directions of the error gradient that oscillate in sign [28]. With momentum, the delta rule for weight adjustment is expressed as

$$\Delta W_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta W_{ji}(n\text{-}1) \qquad (4.8)$$

where

$\Delta W_{ji}(n)$ = correction factor for adjusting the weight between neuron j of the upper layer and neuron I of the lower layer

η = learning rate

α = momentum

δ_j = error signal of the jth neuron

$y_i(n)$ = output value of the ith neuron in the previous layer

Momentum can also prevent the training process to converge on a shallow local minimum on the error surface. Momentum allows a model to react not only to the local error gradient, but also to recent changes in the error surface. If the momentum constant is 0, the weight adjustment is performed using solely the gradient; if the momentum is 1, the gradient is ignored [29].

4.2.6 Data pre-processing

Before training, it is useful to scale the inputs and targets so that they all fall within the same range. Most of the nonlinear activation functions used in backpropagation produce a neuron output restricted to certain intervals, such as [0 1] or [-1 1]. Therefore, normalization of the data over these intervals is critical.

4.3 Adaptive neuro-fuzzy inference systems

The adaptive neuro-fuzzy inference (ANFIS) technique can be used to tune the membership functions of a fuzzy inference system using artificial neural network-based optimization methods. This learning method works similarly to that of neural networks: through an iterative procedure, the fuzzy model learns information about the system, in order to estimate the membership function parameters minimizing the error between the model output and the desired objective.

The membership function parameters are adjusted according to the system's error gradient vector which indicates the performance of the system in tracking the given input-output data. Typically, ANFIS uses either backpropagation or a hybrid method consisting of a combination of least squares estimation and backpropagation for membership function optimization. This hybrid method uses backpropagation to optimize the parameters associated with the input membership functions, and least squares estimation for the output membership functions.

The more the initial membership functions result in a fuzzy system with a good predictive performance, the easier it is for the ANFIS training to converge. In case of a Sugeno-type fuzzy model, ANFIS is usually applied after the combination of clustering parameters

leading to the best prediction performance is identified. The ANFIS training can be set for a number of fixed training epochs, or a training stopping criterion related to the error size can be defined [25].

5 DATA COLLECTION AND PRE-PROCESSING

5.1 Data collection

Historical process data from a Canadian kraft pulp and paper mill is collected in order to develop a soft sensor for the black liquor solid content at the concentrator feed. Four hundred process variables, mostly related to the black liquor recovery circuit, are retrieved for a three-month period, starting at the beginning of October 2007. This time period was chosen since the process operation was stable during this interval. There were no major operational changes or significant equipment shutdowns or malfunctions, so it is considered that this period is representative of a normal process operation.

The measurements are retrieved from the plant data historian as hourly averages, resulting in a dataset containing 2521 observations per variable. Discussions with plant personnel confirm that this is an acceptable averaging, given the relative slow dynamics of the process and the multitude of storage tanks used between key operation units.

5.2 Data pre-processing

The successful development of a data-driven model is highly dependent on the quality of the data. Data-driven modeling techniques are highly susceptible to the adage "garbage in/garbage-out". The steps preceding the development of the model are critical to ensure that useful knowledge is derived from the data. These steps are referred to as data pre-processing, and their objective is to increase the data quality before the modeling procedures can be successfully applied. It is not unusual that upwards of 80% of time and

effort in a project dealing with data analysis be spent on data understanding and pre-processing [30].

In the industrial process context, data quality is a key issue, since historical process data can be incomplete, inconsistent and noisy. Missing variable values, aberrant or out-of-range measurements, outliers and inaccurate values can be present in a database containing process variables. This can be caused by various process or equipment related factors, such as drift and malfunction of measuring instruments, starts and stops of key unit operations, aberrant process behaviour, relative infrequent laboratory analyses or product quality sampling and dubious periods of operation, such as shutdowns, low production periods and equipment cleaning.

The data pre-processing steps performed in this study are data reduction and data cleaning. Data reduction involves removing irrelevant variables such that a reduced data set is obtained, without a significant loss of information from the original data. Data cleaning involves identifying and removing outliers.

A good comprehension of the process and close collaboration with plant personnel during the data pre-processing step is of key importance for developing a proper understanding of the data.

5.2.1 Data reduction

Irrelevant and redundant variables are eliminated from the dataset. A close inspection of the database revealed some redundancy, such as variables derived from other variables. For example, there were three measurements related to the flow of black liquor to one of

the evaporators. One of these measurements represents the total flow rate, and it is the sum of the other two measurements. In this case, only the variable representing the total flow rate is kept in the modeling database.

The data contained variables related to control loops, such as the set point values, remote set point values, manual entries and controller output values. If the controlled variable was present in the database, all other entries related to the control loop were deemed irrelevant and were eliminated. Following discussions with plant personnel, variables not related to the process, such as measurements used for safety alarm purposes or manual verification tests are identified and eliminated from the database.

Variables not available on-line and manually entered in the data historian, such as laboratory analysis and other manual measurements, are also eliminated. Since the soft sensor predicts in real-time the solid content at the concentrator feed, it is critical that variables that are used as model inputs be available in real-time.

A significant number of variables are identified as redundant or irrelevant and subsequently eliminated: 267 variables are removed from the dataset. The dataset now contains 144 variables: 143 input variables, and the output variable – the solid content of the black liquor at the concentrator inlet.

5.2.2 Data cleaning on a variable basis

Data cleaning consists of the identification and removal of outliers and other unwanted deviations. Outliers and other observations related to abnormal operating conditions, such as equipment shutdowns or malfunctions and low-production periods, should be

removed, since the model has to capture information representative of a normal operating regime.

The first step performed in order to separate noise from meaningful data is to clean the variables: a filtering range is calculated for each variable, and observations outside this range are considered outliers and are eliminated from the dataset. The filtering range is calculated using the standard deviation and average of each variable, as follows:

$$\overline{X_n} \pm k\sigma \qquad (5.1)$$

where:

$\overline{X_n}$ = moving average of variable X over the interval n

k = numerical constant

σ = standard deviation of variable X

A moving average is used to emphasize the direction of a trend and to smooth out fluctuations in the data series. The moving average is calculated over 720 hours, since this interval represents approximately a month's worth of measurements. The numerical constant used to multiply the standard deviations is set at 2. All observations falling outside the above-mentioned range are considered outliers and are eliminated.

This procedure uses two passes in order to eliminate outliers. The first cleaning pass eliminates severe outliers that could lead to standard deviation and average values not representative of the general trend of the variable. The second pass uses the standard

deviation and average values of the data cleaned after the first filtering run, in order to further refine the cleaning.

A variable with severe outliers is shown in Figure 5-1, and it can be seen that there are extreme values in the order of 10^{-4}. Figure 5-2 shows a close-up view of the trend of the same variable, and it can be clearly seen that the average value is around 30. Such extreme outliers are probably due to instrument malfunctions or communication problems between the instrument and data historian.

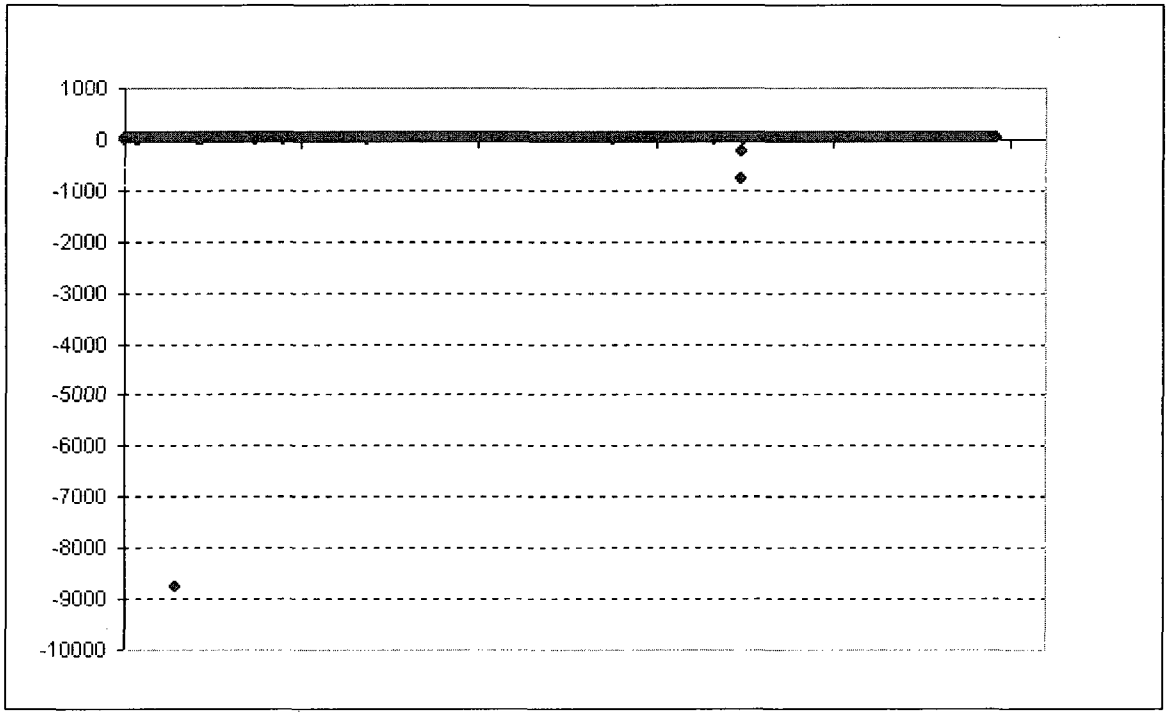


Figure 5-1 Variable containing extreme outliers in the range of 10^{-4}

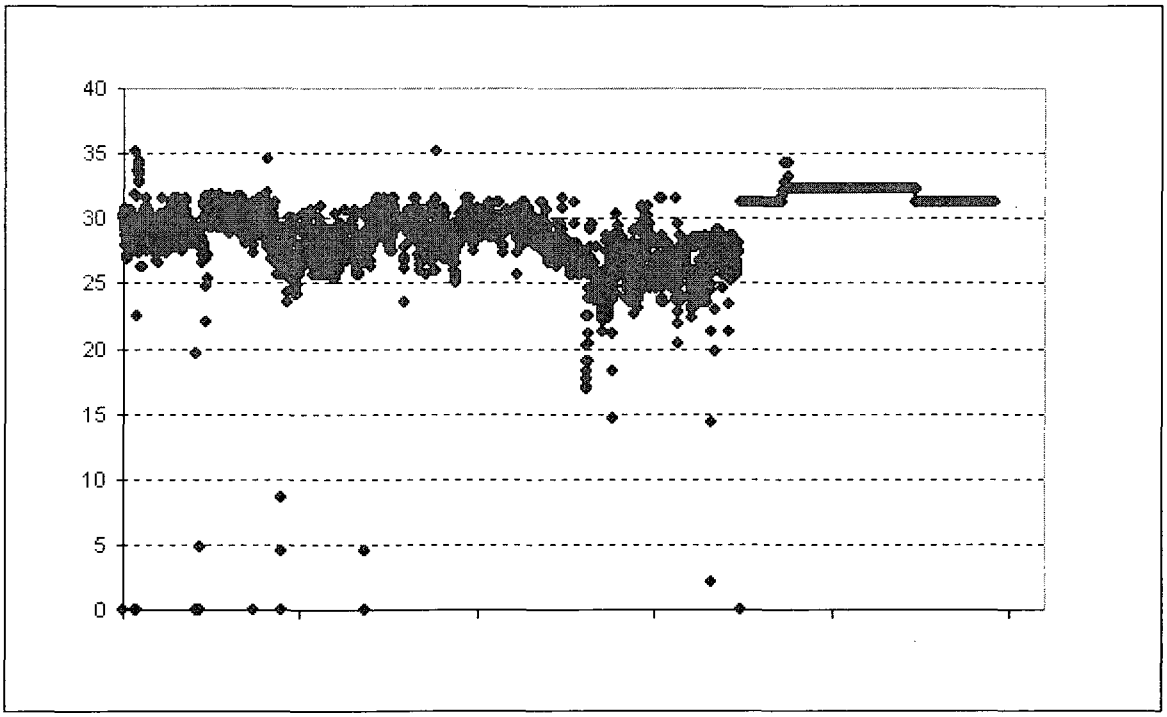


Figure 5-2 Close-up view of the trend of a variable containing extreme outliers in the range of 10^{-4}

Typically, for data cleaning purposes, it can be considered that observations outside the interval defined by plus and minus 3 standard deviations ($\pm 3\sigma$) are outliers. If the data are normally distributed, 99.7% of the points will lie within $\pm 3\sigma$, and it is considered that any values outside this interval do not follow the statistical distribution of the bulk of the data. The historical process dataset used in this study contained periods of equipment shutdowns, and the numerical range of the measurements related to that equipment increases, due to the presence of intermediate measurements from the operating regime to the shutdown level. Because of this, a filtering range calculated with a $\pm 3\sigma$ value would have been too wide and not useful in cleaning the data. Using a $\pm 2\sigma$ value ensures that the filtering range eliminates not only outliers, but also observations related to shutdowns. A variable containing shutdown data and cleaned using a filtering range based on a $\pm 3\sigma$ value is shown in Figure 5-3: very few observations – 0.69% – are eliminated, while virtually all outliers and observations related to the shutdown periods are still present. Figure 5-4 shows the same variable, but cleaned using a filtering range based on $\pm 2\sigma$ value: it can be seen that almost all outliers and observations related to shutdowns are eliminated – 12.95% of the original data is removed.

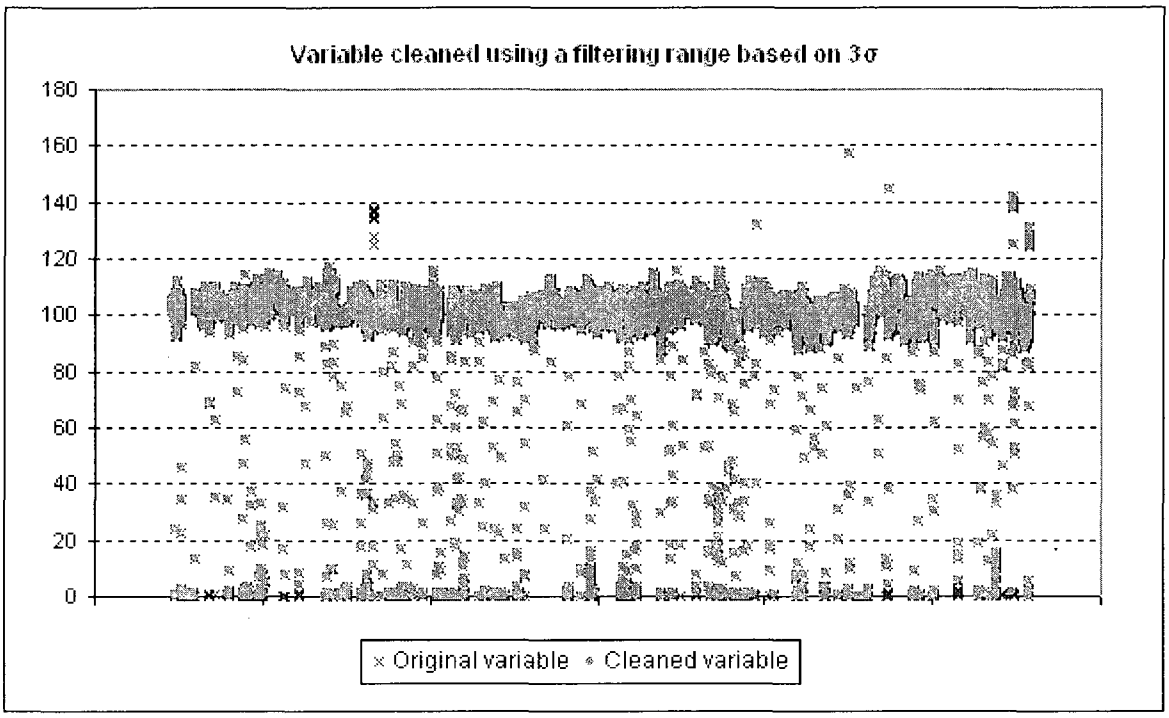


Figure 5-3 Variable containing shutdowns and cleaned using a filtering range based on a ±3σ value

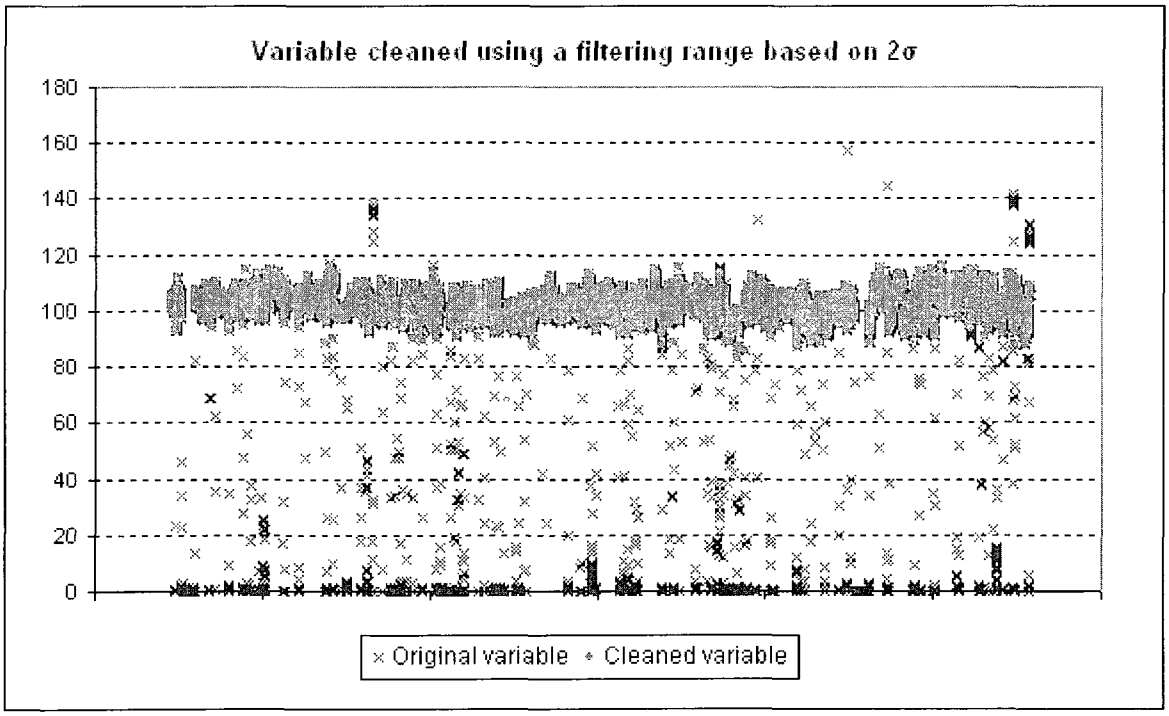


Figure 5-4 Variable containing shutdowns and cleaned using a filtering range based on a ±2σ value

A moving average is used to enable the filtering range to follow the trend of the data more closely than a fixed average. Using a moving average can also limit the detrimental effect that an outliner has on cleaning a variable, as its magnitude influences only the filtering range of the interval containing the outlier, and not the filtering range for the whole variable. Figure 5-5 shows a variable cleaned using a fixed average, and it can be seen that observations consistent with the general run of the data are eliminated. This is not the case when the cleaning is performed using a moving average, as shown in Figure 5-6.

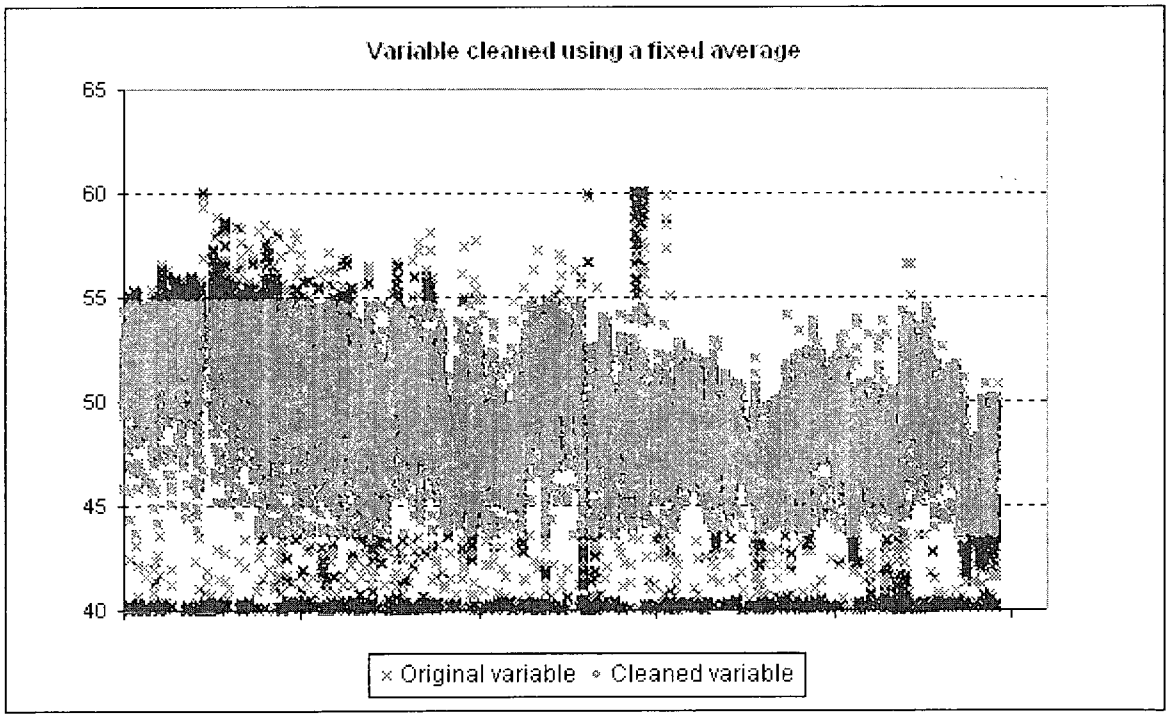


Figure 5-5 Variable cleaned using a fixed average

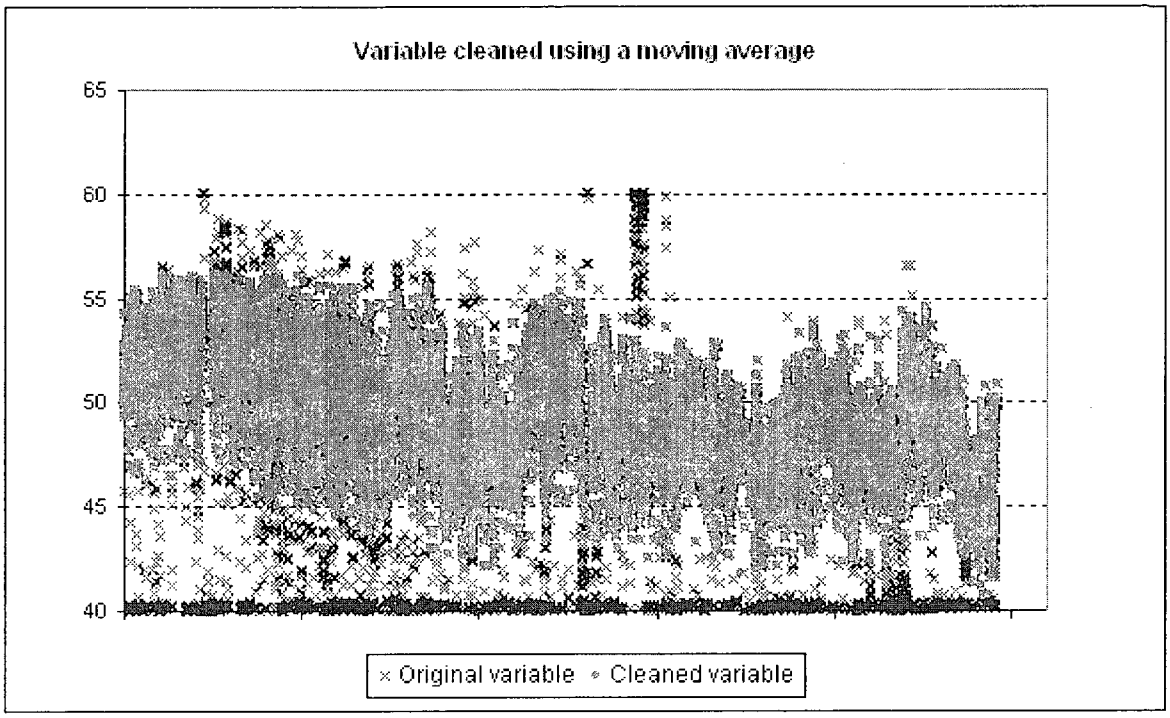


Figure 5-6 Variable cleaned using a moving average

The procedure described previously is applied automatically to the entire database. However, each variable might display different fluctuations, according to the frequency of shutdowns and other changes that might have influenced its behaviour. Since the same cleaning parameters are used for the entire database, each variable is individually inspected after the automatic cleaning process is completed. The objective is to verify that the cleaning procedure performs satisfactory: clear outliers are removed, and valid observations that seemed consistent with the general run of the data are not eliminated. If the data filtering performance is deemed to be unsatisfactory, the cleaning parameters are modified in order to achieve a more efficient filtering.

After inspection of each cleaned variable, it is considered that the cleaning procedure applied to the historical process data is successful in removing outliners, inconsistent and

aberrant values, and data related to abnormal equipment and process operation conditions. However, measurements of the black liquor solid content at the concentrator feed measurements that are below the critical threshold are kept in the modeling dataset in order to enable the model to recognize and predict such occurrences.

Different outlier detection strategies are commonly employed for data filtering. Some of them are the 3-Sigma edit rule, the Jolliffe method and the residual analysis of linear regression [1]. However, the cleaning procedure presented previously proved to be efficient in removing inconsistent data and measurements related to abnormal process behaviour.

5.2.3 Data cleaning on a global process basis

The data cleaning procedure described previously is applied on an individual process variable basis, in the sense that it identifies deviations from a normal operating range for each variable separately – each variable has a filtering interval corresponding to its own statistical behaviour.

After performing the data cleaning on a variable basis, a principal component analysis is performed to model the global process behaviour and determine if any deviations from a normal process operating range are present. The Hotteling's T2 plot is obtained. This plot summarizes all process values and model components, displaying the distance from the origin in the model plane for each observation. Critical limits are determined based on the general grouping of the data, and values far above the critical limits represent outliers that might influence the model in a detrimental way. For databases containing multiple variables, it is practical to visualize all process parameters and measurements on one

combined plot, since it reduces the risk of not detecting deviations when inspecting many individual plots. An inspection of this plot revealed that very few significant process deviations are present in the dataset. The variables most responsible for these deviations are identified, and a closer inspection showed that the corresponding observations are somewhat farther away from the general trend of each individual variable. Once these observations are eliminated, the process deviations disappear.

The Hotelling's T2 plot showing process deviations, as spikes from the general trend of the data, is presented in Figure 5-7. The same plot after the deviations are eliminated is shown in Figure 5-8; it can be seen that the process does not display severe deviations anymore.

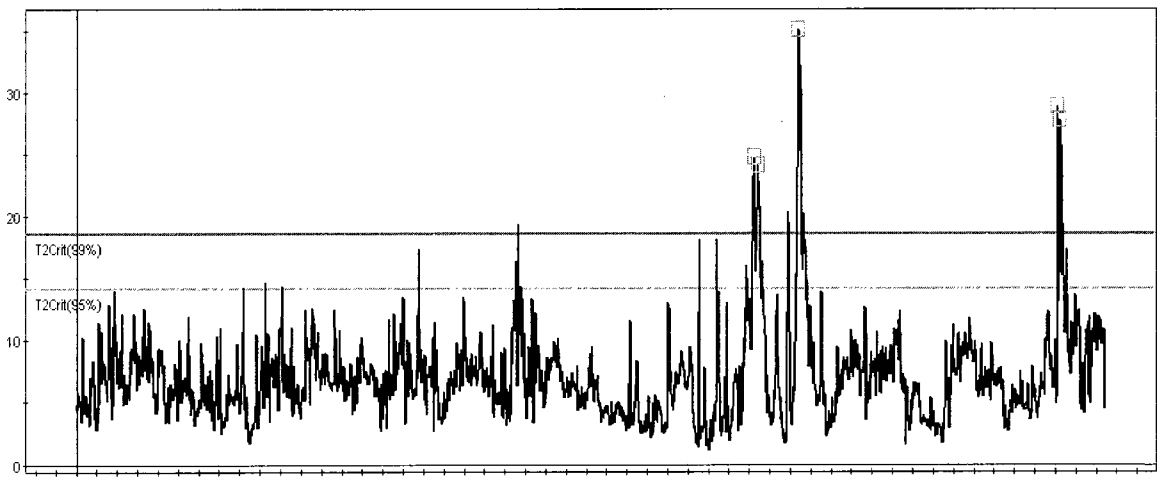


Figure 5-7 Hotteling's T2 plot showing process deviations

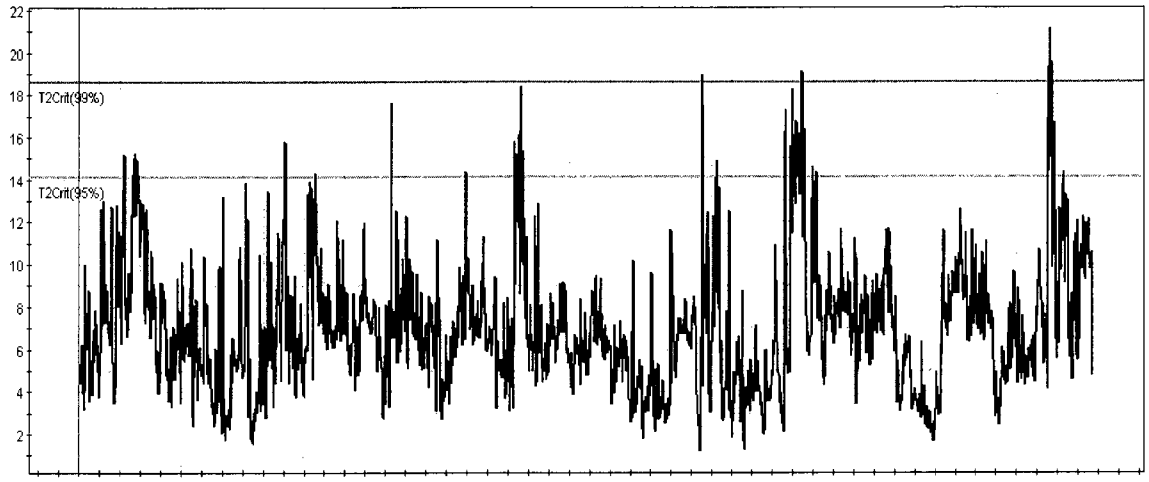


Figure 5-8 Hotteling's T2 plot without process deviations

6 SOFT SENSOR INPUT VARIABLES

The soft sensor performance is directly affected by the quality of real-time measurements used as inputs to the predictive model. This implies that the instruments associated with the input variables have to be rigorously maintained in order to detect any measuring performance degradation. Therefore, if fewer input variables are used, less effort is required to maintain the measuring instruments. The cost associated with the maintenance and re-calibration of these sensors also decreases. From a modeling perspective, using more predictors increases the complexity of the model, making it difficult to manage and less practical to use, and it also can considerably increase the time required to re-compute the model if changes occur in the process. Given these practical reasons related to a soft sensor industrial implementation, it is proposed that 10 input variables be used to develop the soft sensor model. The literature review also revealed that typically a small number of predictors is used to develop industrial soft sensors.

6.1 Input selection methodology

The following methodology is proposed to extract subsets of 10 relevant predictors from the cleaned dataset of 143 input variables:

- a PLS model for inferring the black liquor solid content at the concentrator feed using all 143 variables as inputs is developed

- the correlation between the input and output variables is tested to determine if strong nonlinearities and process lagging are present; if this is the case, they need to be taken into account prior to model development

- two PLS-based criteria are used to identify the most significant variables for predicting the black liquor solid content: the significance in modeling the input data and predicting the output variable, and the significance solely in predicting the output
- two different datasets containing the top 10 variables as identified by each PLS-based criterion will be used to develop the soft sensor models

6.2 PLS analysis on the complete dataset

After the data pre-processing step is completed, a PLS analysis is performed in order to obtain a predictive model of the black liquor solid content at the concentrator feed. All remaining 143 variables from the historical process database are used as inputs to this model. The objective is to estimate the importance of input variables from a regression point of view.

6.2.1 Modeling and validation datasets

Prior to computing the model, the database is separated into modelling and validation subsets. The modeling dataset, also called training dataset, is used for developing the model, while the validation dataset is used to validate the model performance. If only the training data is used to determine the prediction performance, the accuracy can be over-estimated, since the model is specifically tuned to fit the training data. The validation dataset is obtained by setting aside a portion of the original dataset that will not be used during the training process. After the model is fitted on the training data, its performance is tested on the validation data.

For the objective of this study, the dataset is randomly partitioned to obtain the training and validation data. Out of 2521 observations (or lines in the database), 504 lines were randomly selected, in a uniformly distributed manner, as validation data; this represents 20% of the complete database.

6.2.2 *Verification of strong nonlinearities and lagging*

Multivariate data analysis methods, such as PLS, assume linear or quasi-linear relationships between variables. When computing predictive models, the correlation between the input and output variables has to be tested in order to determine if strong nonlinearities are present, since they will detrimental to the predictive performance. The input variables are tested for nonlinearities by introducing in the database new variables containing their respective squared and cubed values, and determining their regression coefficient. A high value of the regression coefficient would show that these new variables have a significant impact on predicting the output variable, thus indicating the presence of strong nonlinearities.

Manufacturing processes can be subjected to lag times caused by the process dynamics. Lag intervals between the time a change occurs in a process variable and the time when the effect of this change impacts the output variable should be taken into account when developing the model. Process lags must be synchronized beforehand, since the modeling methods used in this study are not time series techniques, treating all observations as separate events. Process lagging is verified by introducing lags of 1, 2 and 3 hours between the input variables and the output variable. This means that each observation of the input variables at a time T is correlated with the output variable value at the time T +

1 hour, *T + 2 hours* and *T + 3 hours*, respectively. A high value of the regression coefficients of the lagged variables would show that they have a significant impact on predicting the output variable, thus indicating the presence of lagging.

The results indicate that there is no significant nonlinearity or lagging between the input and output variables. This can be explained by a relatively stable operating regime and slow dynamics of the process, as well as the presence of storage tanks between key processing units.

6.2.3 Model performance

The best PLS model developed using all 143 available input variables contains 7 principal components and models 51.7% of the input (X) data variation and 97.1% of the output (Y) data variation; the model can predict 96.8% of the Y variable variation. Table 6-1 shows this model performance, along with the percentages of X and Y data variation modeled by each component, and the percentage of the Y variable variation predicted by the number of specified components. The first principal component models 22.2% and 51% of the X and Y data, respectively; this means that the first principal component captures almost half of the variation of the X and Y data explained by the 7-component model: 42.9% and 52.5%, respectively. It can also be seen that a 3-component model is able to predict 79.4% of the Y variable variation; this represents 82% of the predictive performance of the 7-component model.

Component number	% of X variation modeled in the component	% of Y variation modeled in the component	% of Y variation predicted by the number of specified components
1	22.2	51	50.9
2	7.1	19.7	70.4
3	7.3	9.1	79.4
4	7.1	6.2	85.7
5	3.1	6	91.3
6	2.8	3	94.7
7	2.1	2	96.8
	% of X variation modeled by the 7-PC model	% of Y variation modeled by the 7-PC model	% of Y variation predicted by the 7-PC model
	51.7	97.1	96.8

Table 6-1 Influence of each component in the PLS model developed using 143 input variables

The mean square error (MSE) is used to measure the model's predictive performance on the training and validation datasets. These results are shown in Table 6-2. It can be seen that the error is very small; this is not surprising, given the considerable number of model inputs.

Number of inputs	Modeling MSE	Validation MSE
143	0.057	0.098

Table 6-2 Performance of the PLS model developed using all 143 inputs

Figure 6-1 illustrates the modeling performance of this model, while Figure 6-2 shows its validation performance. In both cases, it can be seen that the model generates values very close to the actual data.

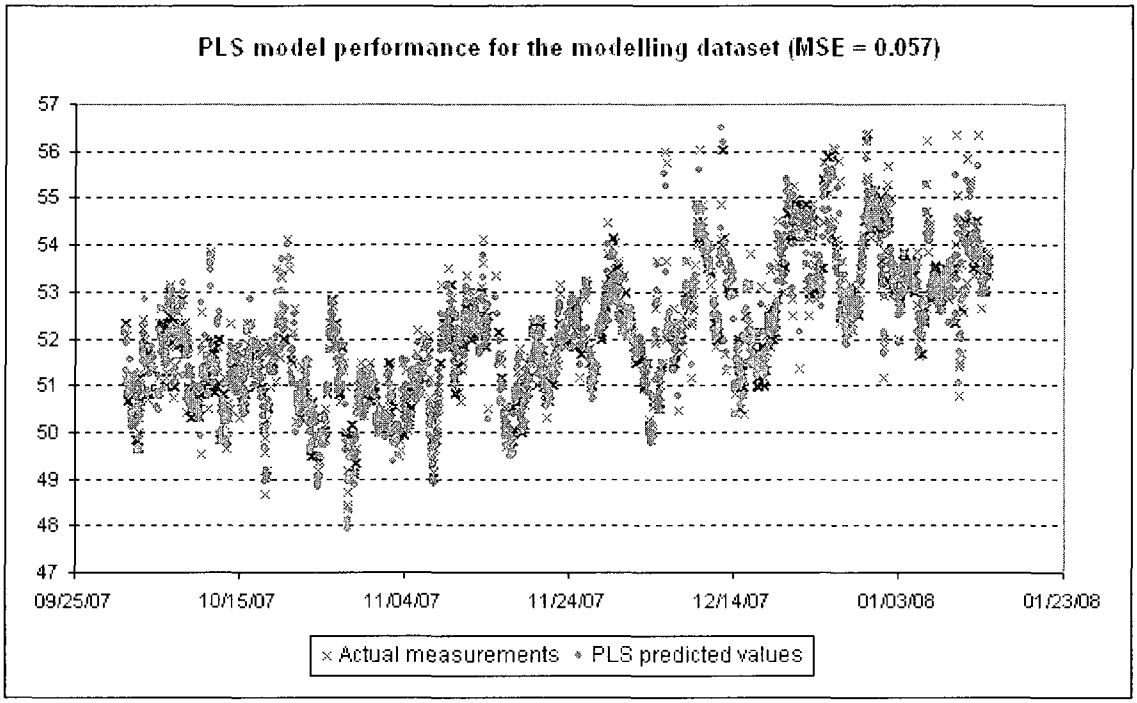


Figure 6-1 Modeling performance of the PLS model using the complete dataset

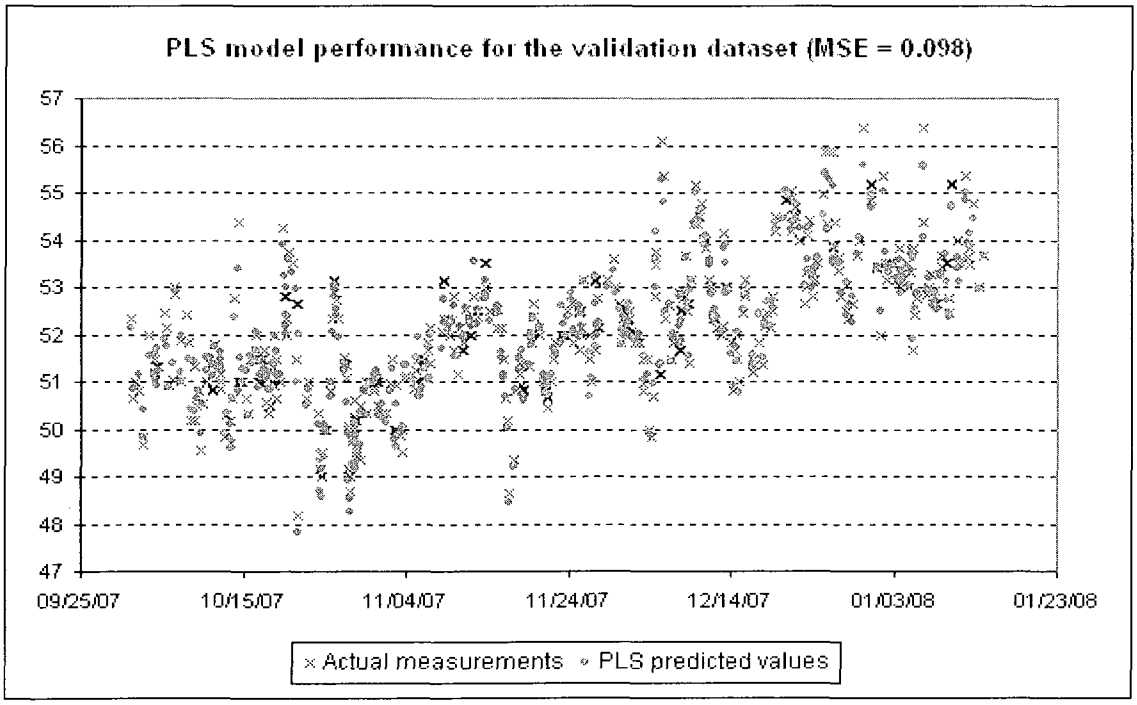


Figure 6-2 Validation performance of the PLS model using the complete dataset

6.3 Input selection criteria

The PLS analysis performed previously is used to identify the most significant variables for inferring the black liquor solid content. Two selection criteria are used:

- the significance in modeling the input dataset and predicting the output, as measured by the VIP value – the Variable Influence on Projection
- the significance in predicting the output, as measured by the absolute magnitude of the coefficients of regression

The VIP and the regression coefficient values are computed for each variable.

Input variables with VIP values greater than 1 can be considered as most relevant for explaining the output variable, since the average VIP is equal to 1. Out of all 143 input variables, 57 have a VIP value greater than 1. Table 6-3 shows the variables having the 20 largest VIP values; the top 10 variables are selected as inputs to the soft sensor models – due to confidentiality issues, the variable description is not included and only the variable number, as entered in the data historian, is shown.

Variable number	VIP value
316	1.64971
83	1.53649
434	1.53281
100	1.48366
804	1.48135
82	1.37923
338	1.37902
800	1.37393
796	1.36617
435	1.36535
793	1.35508
679	1.34278
497	1.31075
366	1.30841
445	1.29364
493	1.29287
217	1.27232
806	1.25313
827	1.25023
828	1.23845

Table 6-3 Variables with the 20 largest VIP values

The size of the regression coefficient indicates the size of the effect that the input variable is having on the output variable, and the sign on the coefficient (positive or negative) indicates the direction of the effect. Table 6-4 shows the variables having the 20 largest absolute regression coefficient values; the top 10 variables are selected as inputs to the soft sensor models.

Variable number	Absolute value of the regression coefficient
316	0.103269
435	0.101454
679	0.0638193
795	0.0496987
366	0.0421756
223	0.0363415
838	0.0350992
830	0.0348395
450	0.0336553
242	0.033317
379	0.0329404
454	0.0328659
281	0.0326359
839	0.0315617
428	0.0303781
92	0.0303355
661	0.0300577
835	0.0299866
434	0.0299355
338	0.0297899

Table 6-4 Variables with the 20 largest regression coefficient values

Two different sets of predictors are selected: the top 10 variables with the largest VIP values, and the top 10 variables with the highest absolute magnitude of the regression coefficient. Each dataset is used to develop the soft sensor models. Both datasets are made up of variables related to the flow of steam and black liquor in the evaporator train, but their composition is quite different, since they contain only two common variables.

Process knowledge and close collaboration with plant personnel are essential in ensuring that the variables deemed significant by the statistical analysis are indeed relevant from a process point of view. The variables displaying a great amount of variability are determined by the statistical model as important in explaining the global dataset variability. The data cleaning performed previously proved to be efficient in removing

measurements not representative of normal process behaviour, such as equipment shutdowns for example. However, it is important to inspect the significant variables, as artificial variability – not representative of a normal process operation – might still be present in the dataset. Variables manually entered in the data historian that were not identified and subsequently removed in the data pre-processing step, measuring instrument failures or communication errors with the data historian are a few examples of causes responsible for the presence of artificial variability in the dataset. Such an example is shown in Figure 6-3: the data consists of a manual log that is not updated on a regular basis. The same value is entered for a certain time interval, and then a different value is repeated over a time interval. The variable is identified as statistically significant, even though the measurements are not valid.

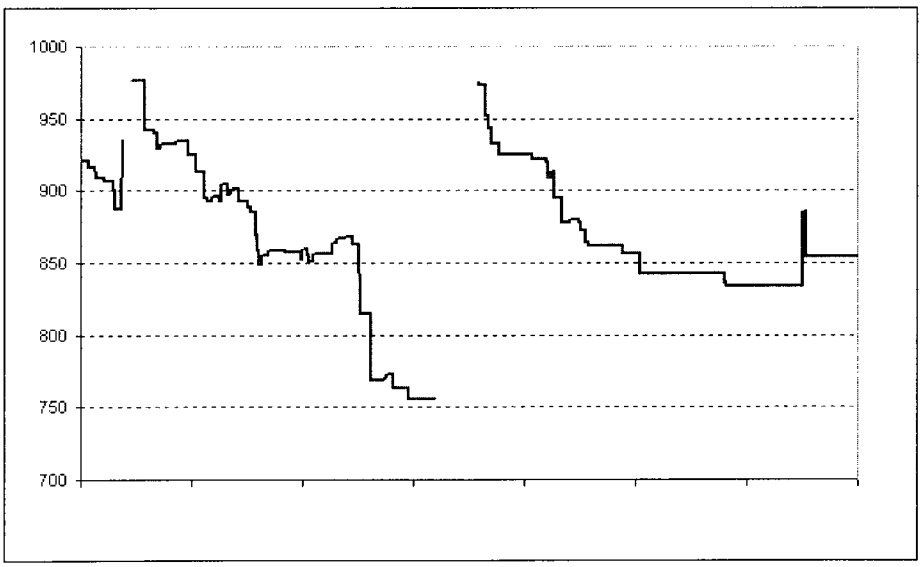


Figure 6-3 Manual log displaying variability not representative of normal process behavior

A close inspection of the 10-input sets presented previously revealed that all the variables are representative of normal process behaviour.

7 SOFT SENSOR MODEL DEVELOPMENT

The 10-input variable datasets described in the previous chapter are used as inputs to the soft sensor. Prior to model development, all observations containing missing values are removed from the dataset, and the training and validation data are then selected. The predictive models are developed using PLS, Sugeno-type fuzzy logic and neural network methods. The predictive performance is measured using the training and validation mean square errors. Models displaying the best training and validation error trade-off are selected. The PLS models are computed using the Umetrics SIMCA-P multivariate data analysis software. All the other models are developed using the Matlab software package, notably the Fuzzy Logic and Neural Networks toolboxes. All the computations are executed using an AMD Athlon 64X2 dual core processor, with a computer processor speed of 2.41 GHz and 2.00 GB of RAM.

7.1 Removal of missing values and validation data selection

The data cleaning procedure identifies and removes outliers and other invalid data, leading to the presence of missing values in the dataset. Since the PLS method is able to handle missing values, this issue is not addressed prior to the development of the PLS model using the entire dataset. However, fuzzy logic and neural network-based methods cannot cope with missing values, and therefore all the rows containing at least one missing value are removed from each 10-input dataset. For each input set, the validation dataset is formed by randomly selecting, in a uniformly distributed manner, 20% of the observations. After the removal of missing values, the number of training and validation observations is slightly different for each input set, as they contain only two common

variables: the inputs selected according to the VIP values contain approximately 10% more training and validation data. These observations are removed, in a uniformly distributed manner, so both datasets have the same number of training and validation observations. The training and validation data partitioning is presented in Table 7-1.

Total number of observations	Training observations	Validation observations
1200	960 – 80%	240 – 20%

Table 7-1 Training and validation data partitioning for each set of input variables

7.2 PLS models

Two PLS models are computed, one for each dataset. The execution time is less than 1 minute for each model.

The best model using the inputs selected according to the VIP values contains 3 principal components and models 84.5% of the input (X) data and 63.3% of the output (Y) data; the model can predict 63.2% of the variation in Y. The mean square modeling and validation errors are 0.689 and 0.627, respectively. Table 7-2 shows the fraction of X and Y data variation modeled by each component, and the fraction of the Y variable variation predicted by the number of specified components. It is seen that the first principal component captures more than half of the variation of the X and Y data explained by the 3-component model: 59.2% and 50.1%, respectively; a 1-component model would be able to predict 50.1% of the Y variable variation.

Component number	% of X variation modeled in the component	% of Y variation modeled in the component	% of Y variation predicted by the number of specified components
1	59.2	50.1	50.1
2	17.2	9.1	59.2
3	8.1	4.0	63.2

Table 7-2 Influence of each component in the PLS model developed using the VIP-based input variables

The best model using the inputs selected according to the regression coefficient values contains 3 principal components and models 73% of X data and 60.8% of the Y; it predicts 60.5% of the variation in Y. The modeling and validation errors are 0.792 and 0.799, respectively. Table 7-3 shows the fraction of X and Y data variation modeled by each component, and the fraction of the Y variable variation predicted by the number of specified components. It is seen that the first principal component captures slightly less than half of the variation of the X and Y data explained by the 3-component model: 44.2% and 44.8%, respectively; a 1-component model would be able to predict 44.7% of the Y variable variation.

Component number	% of X variation modeled in the component	% of Y variation modeled in the component	% of Y variation predicted by the number of specified components
1	44.2	44.8	44.7
2	21.2	13.9	58.6
3	7.6	2.1	60.5

Table 7-3 Influence of each component in the PLS model developed using the regression coefficient-based input variables

The characteristics, modeling and validation performance, and computing time of the PLS models developed using each input set are compared in table 7-4.

Model characteristics	Input dataset	
	VIP	Regression coefficients
number of principal components	3	3
fraction of X data explained by the model	84.5%	73%
fraction of Y data explained by the model	63.3%	60.8%
fraction of Y variation explained by the model	63.2%	60.5%
modeling MSE	0.689	0.792
validation MSE	0.627	0.799
execution time	< 1 minute	< 1 minute

Table 7-4 Characteristics and performance of the PLS models

The model based on the VIP inputs is able to explain more of the X data variation than the model using the regression coefficient-based inputs; this can be explained by the fact that the VIP variables are significant not only for predicting Y, but also for modeling the X space. This model also outperforms the model using the regression coefficient-based inputs in predicting the Y variable. Table 7-5 presents the reduction in error when the model is developed using the VIP-based inputs, compared to the error of the model using the regression coefficient-based inputs.

training MSE reduction when using the VIP-based inputs	13%
validation MSE reduction when using the VIP-based inputs	21.5%

Table 7-5 Error reduction of the PLS model when the VIP input-based are used

7.3 Sugeno-type fuzzy models

First and second order Sugeno-type fuzzy models are developed using each set of inputs. All the models are developed by clustering the training data on the combined input-output space using the subtractive clustering algorithm. The following model configurations are used:

- first order models, with the same cluster radius for all data dimensions, and using the *Product* (*PROD*) implication function for determining the rule weights
- first order models, with the same cluster radius for all data dimensions, and using the *Minimum* (*MIN*) implication function
- first order models, with a different cluster radius for each data dimension, and using the *Product* (*PROD*) implication function
- second order models, with the same cluster radius for all data dimensions, and using the *Product* (*PROD*) implication function

The *PROD* implication function uses the product between the membership values of each input dimension to determine the rule weight associated with that input, while the *MIN* function uses the minimum of all membership values to determine the rule weight for that input. The order of the polynomial describing the rule consequents determines the order of the system.

Different combinations of clustering parameters are used to generate the Sugeno models, and their predictive performance is compared. Based on the training and validation errors, the combination of clustering parameters that generates the best model is

identified. As mentioned previously, the radius determines the cluster's influence in the data space, the squash factor indicates the distance between clusters, and the accept and reject ratios indicate the degree of which each data point has the potential of becoming a cluster center.

When using the same radius value for all data dimensions, the cluster has the same range of influence in each dimension. For the models obtained by using the same radius value for all data dimensions, the values of the clustering parameters are increased on the intervals presented in Table 7-6, with an incremental step size of 0.1 in each case. Given all possible combinations, 22000 different models are obtained when the clustering is carried out using a common radius for all data dimensions.

Clustering parameter	Cluster radius	Reject ratio	Accept ratio	Squash factor
Range	0.1 – 1	0 – 0.9	0 – 1	0 .1– 2
Step size	0.1	0.1	0.1	0.1

Table 7-6 Parameters used for clustering with the same radius for all data dimensions

When using a different radius value for each of the data dimensions, the cluster has different ranges of influence in each dimension and the number of clustering combinations increases considerably. For a 10-input dataset, if the clustering ranges and incremental step sizes are maintained the same as for the clustering using the same radius for all data dimensions, 2.2×10^{13} different models would be obtained, resulting in an extremely long computing time. To avoid this, fewer combinations of the clustering parameters are used to perform the clustering with a different radius value for each data dimension. These parameters are presented in Table 7-7.

Clustering parameter	Cluster radius	Reject ratio	Accept ratio	Squash factor
Values	0.2, 0.5, 0.7	0.1	0.3	0.7

Table 7-7 Parameters used for clustering with a different radius for each data dimension

This results in obtaining 177147 different models, and it is considered that a relative representative picture of a Sugeno model with clustering performed using a different cluster radius for each data dimension could be obtained using this reduced number of parameter combinations.

7.3.1 1st order models with the same cluster radius for all data dimensions, and PROD implication function

For each input set, 22000 first order models with the same cluster radius for all data dimensions and using the *Product* implication function for calculating the rule weights are obtained.

The best Sugeno model using the inputs selected according to the VIP values contains 21 rules, and generates training and validation errors of 0.236 and 0.358, respectively. The execution time for computing all the models is 36.1 hours.

The best Sugeno model using the inputs selected according to the regression coefficient values contains 17 rules, and generates training and validation errors of 0.299 and 0.404, respectively. The execution time for computing all the models is 34.2 hours.

These results, as well as the clustering parameters for each best model are summarized in Table 7-8.

Clustering parameters	Input dataset	
	VIP	Regression coefficients
number of rules	21	17
cluster radius	0.6	0.6
reject ratio	0.1	0.1
accept ratio	0.6	0.1
squash factor	0.5	1.1
training MSE	0.236	0.299
validation MSE	0.358	0.404
execution time for all models	36.1 hours	34.2 hours

Table 7-8 Characteristics of the best 1st order Sugeno model with the same cluster radius for all data dimensions, and using the *PROD* implication function

The model using the VIP-based inputs outperforms the model using the regression coefficient-based inputs, for both the modeling and validation errors. The execution time for computing all the models is slightly shorter for the regression coefficients model. The reduction in error when the model is developed using the VIP-based inputs, compared to the model using the regression coefficient-based inputs, is presented in Table 7-9.

training MSE reduction when using the VIP-based inputs	21.1%
validation MSE reduction when using the VIP-based inputs	11.4%

Table 7-9 Error comparison between the VIP and regression coefficient-based 1st order Sugeno models with the *PROD* implication function and the same radius for all data dimensions

7.3.2 1st order models with the same cluster radius for all data dimensions, and MIN implication function

For each input set, 22000 first order models with the same cluster radius for all data dimensions and using the *Minimum* implication function for calculating the rule weights are obtained.

The best Sugeno model using the inputs selected according to the VIP values contains 28 rules, and generates training and validation errors of 0.185 and 0.393, respectively. The execution time for computing all the models is 39.2 hours.

The best Sugeno model using the inputs selected according to the regression coefficient values contains 35 rules, and generates training and validation errors of 0.162 and 0.42, respectively. The execution time for computing all the models is 39.6 hours.

These results, as well as the clustering parameters for each best model are summarized in Table 7-10.

Clustering parameters	Input dataset	
	VIP	Regression coefficients
number of rules	28	35
cluster radius	0.7	0.8
reject ratio	0.1	0
accept ratio	0.4	0.1
squash factor	0.5	0.8
training MSE	0.185	0.162
validation MSE	0.393	0.42
execution time for all models	39.2 hours	39.6 hours

Table 7-10 Characteristics of the best 1st order Sugeno models with the same cluster radius for all data dimensions and using the *MIN* implication function

For the training error, the model using the regression coefficient-based inputs performs better than the model using the VIP-based inputs, reducing the error by 12.4%. For the validation error, the model using the VIP-based inputs performs better than the model using the regression coefficient-based inputs, reducing the error by 6.4%.

These results are summarized in Table 7-11.

training MSE reduction when using the regression coefficient-based inputs	12.4%
validation MSE reduction when using the VIP-based inputs	6.4%

Table 7-11 Error comparison between the VIP and regression coefficient-based 1st order Sugeno models with the *MIN* implication function and the same radius for all data dimensions

7.3.3 1st order models with a different cluster radius for each data dimension, and PROD implication function

For each input set, 177147 models with a different cluster radius for each data dimension, and using the *Product* implication function for calculating the rule weights, are obtained.

The best Sugeno model using the inputs selected according to the VIP values contains 24 rules and generates modeling and validation errors of 0.196 and 0.331, respectively. The execution time for computing all the models is 167.4 hours.

The best Sugeno model using the inputs selected according to the regression coefficients values contains 27 rules and generates modeling and validation errors of 0.18 and 0.461, respectively. The execution time for computing all the models is 184.3 hours.

The results are summarized in Table 7-12.

Clustering parameters	Input dataset	
	VIP	Regression coefficients
number of rules	24	27
input 1 cluster radius	0.7	0.7
input 2 cluster radius	0.7	0.5
input 3 cluster radius	0.7	0.5
input 4 cluster radius	0.7	0.5
input 5 cluster radius	0.7	0.7
input 6 cluster radius	0.2	0.7
input 7 cluster radius	0.5	0.7
input 8 cluster radius	0.2	0.7
input 9 cluster radius	0.2	0.7
input 10 cluster radius	0.7	0.5
reject ratio	0.1	0.1
accept ratio	0.3	0.3
squash factor	0.7	0.7
training MSE	0.196	0.180
validation MSE	0.331	0.461
execution time for all models	167.4 hours	184.3 hours

Table 7-12 Configuration and performance of the best 1st order Sugeno models with a different cluster radius for each data dimension and using the *PROD* implication function

For the training error, the model using the regression coefficient-based inputs performs slightly better than the model using the VIP-based inputs, by an 8.2% margin. For the validation error, the VIP-based model significantly outperforms the regression coefficient-based model, by a 28.2% margin. These results are summarized in Table 7-13.

training MSE reduction when using the regression coefficient-based inputs	8.2%
validation MSE reduction when using the VIP-based inputs	28.2%

Table 7-13 Error comparison between the VIP and regression coefficient-based 1st order Sugeno models with a different cluster radius for each data dimension

7.3.4 2nd order models with the same radius for all data dimensions, and PROD implication function

For each input set, 22000 second order models with the same cluster radius for all data dimensions and using the *Product* implication function for calculating the rule weights are obtained.

The best Sugeno model using the inputs selected according to the VIP values contains 15 rules and generates modeling and validation errors of 0.19 and 0.365, respectively. The execution time for computing all the models is 71.3 hours.

The best Sugeno model using the inputs selected according to the regression coefficient values contains 13 rules and generates modeling and validation errors of 0.22 and 0.415, respectively. The execution time for computing all the models is 81.9 hours.

The results are summarized in Table 7-14.

Clustering parameters	Input dataset	
	VIP	Regression coefficients
number of rules	15	13
cluster radius	0.5	0.3
reject ratio	0.1	0.5
accept ratio	0.2	0.5
squash factor	1.2	1.3
training MSE	0.190	0.22
validation MSE	0.365	0.415
execution time for all models	71.3 hours	81.9 hours

Table 7-14 Characteristics of the best 2nd order Sugeno models with the same cluster radius for all data dimensions and using the *PROD* implication function

The model using the VIP-based inputs outperforms the model using the regression coefficient-based values, for both the modeling and validation error. The reduction in

error when the model is developed using the VIP-based inputs, compared to the error of the model using the regression coefficient-based inputs, is shown in Table 7-15.

training MSE reduction when using the VIP-based inputs	13.6%
validation MSE reduction when using the VIP-based inputs	12%

Table 7-15 Error reduction of the 2nd order Sugeno models with the same cluster radius for all data dimensions when the VIP-based inputs are used

7.3.5 The best Sugeno fuzzy models

The performance of the best Sugeno models obtained previously is presented in Table 7-16; for each model type, the execution time required to compute all possible clustering combinations is also shown.

Sugeno model type	Error (MSE) and execution time (hours)	Input dataset	
		VIP	Regression coefficients
1st order, same cluster radius for all data dimensions, *PROD* implication	training	0.236	0.299
	validation	0.358	0.404
	execution time for all models	36.1	34.2
1st order, same cluster radius for all data dimensions, *MIN* implication	training	0.185	0.162
	validation	0.393	0.420
	execution time for all models	39.2	39.6
1st order, different cluster radius for each data dimension, *PROD* implication	training	0.196	0.180
	validation	0.331	0.461
	execution time for all models	167.4	184.3
2nd order, same cluster radius for all data dimensions, *PROD* implication	training	0.190	0.220
	validation	0.365	0.415
	execution time for all models	71.3	81.9

Table 7-16 Performance of the best Sugeno models

The *PROD* implication function uses the product between the membership values of each input dimension in order to determine the weight of the rule associated with the input. If these membership values are relatively low, and there are many input dimensions, the

rule weights can have very low values. This can lead to a poor predictive performance, since there might be an insufficient number of rules with a high enough firing strength to accurately compute the model output; in order to have a complete rule-base, any input data vector should fire at least one rule with a relative high weight [31]. This situation did not occur in our case, as the models developed using the *PROD* implication function are fairly accurate. The first order models developed using the *MIN* implication function display a lower training error and a slightly higher validation error than that of models developed using the *PROD* implication. Second order models display a training performance superior to that of first order models developed using the same cluster radius for all data dimensions and the *PROD* implication function; their training performance is inferior to that of first order models developed using the *MIN* implication function.

The first order model with the same cluster radius for all data dimensions, the *MIN* implication function and developed using the regression coefficient-based inputs has the smallest training error among all configurations. The first order model with a different cluster radius for each data dimension and *PROD* implication function, developed using the VIP-based inputs, displays the lowest validation error. These results are summarized in Table 7-17.

Sugeno model type	Errors (MSE)		Input dataset
1st order, same cluster radius for all data dimensions, *MIN* implication	training	0.162	Regression coefficients
	validation	0.420	
1st order, different cluster radius for each data dimension, *PROD* implication	training	0.196	VIP
	validation	0.331	

Table 7-17 Sugeno models with the lowest training and validation errors

In most of the cases, the models developed using the input variables selected according to the VIP values outperform the models developed using inputs selected according to the regression coefficient values. In the cases when the models using the regression coefficient-based inputs have a smaller training error, the models using the VIP-based inputs display superior generalization ability, as shown by their validation performance. A comparison of the performance of each method, in terms of the predictor selection method, is shown in Table 7-18.

Model Type	Superior training error	Superior validation error
1st order, same cluster radius for all data dimensions, PROD implication	VIP	VIP
1st order, same cluster radius for all data dimensions, MIN implication	Regression coefficients	VIP
1st order, different cluster radius for each data dimension, *PROD* implication	Regression coefficients	VIP
2nd order, same cluster radius for all data dimensions, *PROD* implication	VIP	VIP

Table 7-18 Performance of the Sugeno models in terms of predictor selection method

7.4 Artificial neural network models

Two feedforward backpropagation neural network architectures are developed for each input set: a single hidden layer network, and a two hidden layer network. In both cases, the output layer contains one neuron with a linear transfer function. The Levenberg-Marquardt optimization method is selected as the training function for updating weight and bias values; this algorithm is fast, and it is used extensively in backpropagation networks [29]. The batch learning mode is used to train all the networks. The learning rate and momentum constant are set at 0.01 and 0.9, respectively, in order to ensure

network stability and minimize the risk of convergence on a local minimum. The maximum number of training epochs is set at 300.

For both network architectures, the number of neurons in each hidden layer is increased from 5 to 20. The transfer functions are either the logarithmic sigmoid or hyperbolic tangent functions; these functions are commonly used in backpropagation algorithms [28]. All possible combinations of these parameters are used to compute the models and the networks displaying the best training and validation error trade-off are considered as the best models.

7.4.1 One-layer neural network models

For the one-hidden layer network, there are 32 possible combinations of different number of hidden neurons and transfer functions.

The best network configuration using the inputs selected according to the VIP values contains 18 hidden neurons with the hyperbolic tangent transfer function and generates training and validation errors of 0.272 and 0.413, respectively. The execution time for computing all the models is 0.16 hours.

The best network configuration using the inputs selected according to the regression coefficient values contains 19 hidden neurons with the hyperbolic tangent transfer function, and generates training and validation errors of 0.288 and 0.429, respectively. The execution time for computing all the models is 0.14 hours.

The results are summarized in Table 7-19.

Network parameters	Input dataset	
	VIP	Regression coefficients
number of hidden neurons	18	19
neuron activation function	hyperbolic tangent	logarithmic sigmoid
training MSE	0.272	0.288
validation MSE	0.413	0.429
execution time for all models	0.16 hours	0.14 hours

Table 7-19 The best one-layer standard backpropagation networks

The model using the VIP-based inputs outperforms the model using the regression coefficient-based inputs, in terms of both training and validation errors.

7.4.2 Two-layer neural network models

For the two-hidden layer network, there are 1024 possible combinations of different number of hidden neurons and transfer functions.

The best network configuration using the inputs selected according to the VIP values contains 14 neurons with the hyperbolic tangent function in the first layer, and 20 neurons with the hyperbolic tangent function in the second layer; the training and validation errors are 0.157 and 0.401, respectively. The execution time for computing all the models is 9.8 hours.

The best network configuration using the inputs selected according to the regression coefficient values contains 18 neurons with the logarithmic sigmoid function in the first layer, and 11 neurons with the logarithmic sigmoid function in the second layer; the training and validation errors are 0.174 and 0.420, respectively. The execution time for computing all the models is 10.9 hours.

The results are summarized in Table 7-20.

Network parameters	Input dataset	
	VIP	Regression coefficients
number of neurons in the 1st layer	14	18
1st layer activation function	hyperbolic tangent	logarithmic sigmoid
number of neurons in the 2nd layer	20	11
2nd layer activation function	hyperbolic tangent	logarithmic sigmoid
training MSE	0.157	0.174
validation MSE	0.401	0.420
execution time for all models	9.8 hours	10.9 hours

Table 7-20 The best two-layer standard backpropagation networks

The model using the VIP-based inputs outperforms the model using the regression coefficient-based inputs, in terms of both training and validation errors.

7.4.3 The best neural network models

The two-layer networks generate a smaller training error than the one-layer networks; however, the validation errors of the one and two-layer models are very close. The two-layer model obtained using the VIP inputs displays the smallest training and validation errors among all neural network configurations. In both cases, the network using the inputs selected according to the VIP values outperforms the network using the inputs selected according to the regression coefficient values. However, this difference is not significant, being less than 10%.

The reduction in error when the model is developed using the VIP-based inputs, compared to that of the model using the regression coefficient-based inputs, is presented in Table 7-21.

	1-layer network	2-layer network
training MSE reduction when using the VIP-based inputs	5.55%	9.77%
validation MSE reduction when using the VIP-based inputs	3.73%	4.52%

Table 7-21 Error reduction of the best neural model when the VIP-based inputs are used

7.5 ANFIS models

An *ANFIS* model is developed for each input dataset. The membership functions of the best first order Sugeno models with the same cluster radius for all data dimensions and using the *PROD* implication function are further tuned using the *ANFIS* method. This optimization is carried out using a hybrid method: the backpropagation algorithm is used to improve the parameters associated with the input membership functions, while a least squares optimization is performed for the output membership functions. The *ANFIS* method combines the advantages of fuzzy and neural networks learning, allowing the Sugeno system, trough an iterative process, to adjust the membership functions in order to minimize the error. The number of maximum training epochs is set at 3000. The *ANFIS* optimization can be carried out according to a minimum modeling performance criterion – the model displaying the lowest training error is considered the best – or according a minimum validation performance criterion – the model displaying the lowest validation error is considered the best. Compared to the original Sugeno models, the ANFIS optimization method according to the validation error results in minimal improvement of both training and validation errors – less than 1% for both input sets.

The results of the *ANFIS* optimization according to the training error, along with the original Sugeno models, are presented in Table 7-22.

Model type	Input dataset			
	VIP		Regression coefficients	
	Modelling MSE	Validation MSE	Modelling MSE	Validation MSE
Sugeno	0.236	0.358	0.299	0.404
ANFIS	0.163	0.48	0.262	0.402
ANFIS execution time	2.23 hours		1.4 hours	

Table 7-22 *ANFIS* models

The *ANFIS* method improves the training error of the model using the inputs selected according to the VIP values by 30.9% when compared to the corresponding Sugeno model; however, this comes at the detriment of the validation error, which increases by 25.4%. For model using the inputs selected according to the regression coefficient values, the *ANFIS* method optimizes both the training and modeling errors, when compared to the corresponding Sugeno model: 12.4% and 0.5%, respectively.

The ANFIS model using the VIP-based inputs presents a modeling performance increase of 37.8% when compared to the modeling error of the model using the regression coefficient-based inputs; the validation error of the VIP model is 16.2% bigger than that of the regression coefficients model. These results are summarized in Table 7-23.

training MSE reduction when using the VIP-based inputs	37.8%
validation MSE reduction when using the regression coefficient-based inputs	16.2%

Table 7-23 Error difference between the VIP and regression coefficient-based *ANFIS* models

7.6 Discussion of results

All models using 10 predictors generate relatively small training and validation errors. This indicates that, for the application presented in this work, 10 process variables are sufficient to develop an accurate black liquor solid content soft sensor, and both the VIP and regression coefficient-based input selection methods are accurate enough to obtain a good prediction performance.

Since a small number of model inputs is used, their selection is critical. The soft sensor performance might suffer if variables not relevant to the prediction of the output variable are used to develop the model. Both the predictor set selected according to the VIP value, and the set selected according to the absolute magnitude of the regression coefficients perform well as predictive model inputs. For the models developed using a 10 variable input set, the Sugeno-type fuzzy logic and neural network-based models outperform the PLS models; this can be explained by their ability to explain nonlinearities better than the PLS method, which is a linear modeling technique. The best model configurations, for each method used in this study, sorted according to the lowest training and validation error are presented in Tables 7-24 and 7-25, respectively. The performance of the PLS model developed using all 143 input variables is also shown, and it can be seen that it outperforms the other models; however, as mentioned previously, due to modeling restrictions and issues related to an industrial implementation, it is not practical to use so many predictors for soft sensor development.

Modeling technique	Input selection method	Number of inputs	Best model configuration	Training error	Validation error
PLS	N/A	143	7 principal components	0.057	0.098
Neural networks	VIP	10	2 layers backpropagation	0.157	0.401
Sugeno fuzzy logic	Regression coefficients	10	1st order, same cluster radius for all dimensions, *MIN* implication	0.162	0.42
ANFIS	VIP	10	hybrid optimization	0.163	0.480
Sugeno fuzzy logic	Regression coefficients	10	1st order, different cluster radius for each dimension, *PROD* implication	0.180	0.461
Sugeno fuzzy logic	VIP	10	2nd order, same cluster radius for all dimensions, *PROD* implication	0.190	0.365
Sugeno fuzzy logic	VIP	10	1st order, same cluster radius for all dimensions, *PROD* implication	0.236	0.358
Neural networks	VIP	10	1 layer backpropagation	0.272	0.413
PLS	VIP	10	3 principal components	0.689	0.627

Table 7-24 Best models sorted according to the lowest training error

Modeling technique	Input selection method	Number of inputs	Best model configuration	Training error	Validation error
PLS	N/A	143	7 principal components	0.057	0.098
Sugeno fuzzy logic	VIP	10	1st order, different cluster radius for each dimension, *PROD* implication	0.196	0.331
Sugeno fuzzy logic	VIP	10	1st order, same cluster radius for all dimensions, *PROD* implication	0.236	0.358
Sugeno fuzzy logic	VIP	10	2nd order, same cluster radius for all dimensions, *PROD* implication	0.190	0.365
Sugeno fuzzy logic	VIP	10	1st order, same cluster radius for all dimensions, *MIN* implication	0.185	0.393
Neural networks	VIP	10	2 layers backpropagation	0.157	0.401
ANFIS	Regression coefficients	10	Hybrid optimization	0.262	0.402
Neural networks	VIP	10	1 layer backpropagation	0.272	0.413
PLS	VIP	10	3 principal components	0.689	0.627

Table 7-25 Best models sorted according to the lowest training error

The models developed using inputs selected according to the VIP values consistently outperform the models using the regression coefficients values: out of 8 model configurations computed for each input dataset, the VIP-based models generate both training and validation errors lower than those of the regression coefficient-based models in 5 cases. For 2 Sugeno models – the first order model with the same cluster radius for all data dimensions and *MIN* implication function, and the first order model with a different cluster radius for each data dimension and *PROD* implication function – the regression coefficient-based models display a slightly better modeling performance, but the VIP-based models have lower validation errors. In the case of the ANFIS method, the regression coefficient-based model displays a better validation performance, but the VIP-based model has a lower training error.

For an industrial implementation, the ability to use real-time measurements not used during model training for accurately predicting the output variable is critical in measuring the soft sensor performance and its degree of usefulness for optimizing the process operation. If the validation error is considered as the main performance measure, the top 5 models are developed using input variables selected according to the VIP values.

Given these results, it is considered that, for this application, predictors selected according to the VIP value lead to the development of more accurate models than predictors selected according to the regression coefficient values.

Using variables with high VIP values for the development of industrial soft sensors offer certain advantages, as these variables describe simultaneously both the X space and the

correlation between the X and Y variables, while the regression coefficient variables describe only the regression relationship between X and Y:

- datasets of industrial process measurements often contain a significant number of correlated variables, and the dataset variability is driven by relatively few process variables. In other words, only a few variables might be responsible for the global variability displayed by the process. In these situations, variables with high VIP values can successfully reduce the X space dimension while retaining most of the information pertinent to the process variability, thus allowing the development of an accurate regression model with fewer inputs

- the simultaneous modeling of both the X and Y spaces leads to an unique solution to the predictive problem, as opposed to a number of different solutions that regression methods dealing solely with the Y prediction can provide. For example, for the same input dataset, different partial least regression algorithms will assign different regression coefficient values to the input variables; this multiple choice of answers can lead to low confidence regarding the choice of model parameters [32].

For each modelling technique, the execution times required to compute all possible model configurations are summarized in Tables 7-26 and 7-27. Table 7-26 shows the computing times required to obtain all model configurations in order to identify the training-error best models, while Table 7-27 shows the computing times required to obtain all model configurations on order to identify the validation-error best models.

Modeling technique	Input selection method	Number of inputs	Best model configuration	Number of all possible model configurations	Computing time
PLS	N/A	143	7 principal components	1	< 1 minute
PLS	VIP	10	3 principal components	1	< 1 minute
Neural networks	VIP	10	1 layer backpropagation	32	0.16 hours
ANFIS	VIP	10	hybrid optimization	1	2.23 hours
Neural networks	VIP	10	2 layers backpropagation	1024	9.8 hours
Sugeno fuzzy logic	VIP	10	1^{st} order, same cluster radius for all dimensions, *PROD* implication	22000	36.1 hours
Sugeno fuzzy logic	Regression coefficients	10	1^{st} order, same cluster radius for all dimensions, *MIN* implication	22000	39.6 hours
Sugeno fuzzy logic	VIP	10	2^{nd} order, same cluster radius for all dimensions, *PROD* implication	22000	71.3 hours
Sugeno fuzzy logic	Regression coefficients	10	1^{st} order, different cluster radius for each dimension, *PROD* implication	177147	184.3 hours

Table 7-26 Best training-error models sorted according to the lowest computing time required to obtain all model configurations

Modeling technique	Input selection method	Number of inputs	Best model configuration	Number of all possible model configurations	Computing time
PLS	N/A	143	7 principal components	1	< 1 minute
PLS	VIP	10	3 principal components	1	< 1 minute
Neural networks	VIP	10	1 layer backpropagation	32	0.16 hours
ANFIS	Regression coefficients	10	Hybrid optimization	1	1.4 hours
Neural networks	VIP	10	2 layers backpropagation	1024	9.8 hours
Sugeno fuzzy logic	VIP	10	1^{st} order, same cluster radius for all dimensions, *PROD* implication	22000	36.1 hours
Sugeno fuzzy logic	VIP	10	1^{st} order, same cluster radius for all dimensions, *MIN* implication	22000	39.2 hours
Sugeno fuzzy logic	VIP	10	2^{nd} order, same cluster radius for all dimensions, *PROD* implication	22000	71.3 hours
Sugeno fuzzy logic	VIP	10	1^{st} order, different cluster radius for each dimension, *PROD* implication	177147	167.4 hours

Table 7-27 Best validation-error models sorted according to the lowest computing time required to obtain all model configurations

The PLS models display the shortest computing time, followed by the neural network models and the Sugeno models. The ANFIS models display a relatively short computing time. However, prior to computing the ANFIS models, the best first order Sugeno model with the same cluster radius for all data dimensions has to be identified; this computing time is not included in the previous tables.

The training is performed off-line, and the search for the best model configuration is carried out one time. Once the best configuration is identified, the real-time calculation of the soft sensor output is done very fast: for all the best models identified in this study, it takes less than 5 seconds to calculate the output of one input observation. Therefore, it can be considered that the computing time required to identify the best model configuration is a less relevant performance criterion.

The best two-hidden layer neural network model generates the lowest training error, but the first order Sugeno fuzzy model with clustering performed using a different cluster radius for each data dimension generates the lowest validation error. In both cases, the model inputs are selected according to the VIP criterion. Identifying the best modeling method for this soft sensor application becomes somewhat subjective, in the sense that different criteria can be considered:

- if the main performance criteria is the training error and computational time, the two-layer neural network model can be considered as the best modeling method
- if the main performance criterion is the validation error, the Sugeno fuzzy model with clustering performed using different cluster radii for each data dimension can be considered as the best modeling method

As mentioned before, the validation error is essential in evaluating the soft sensor performance, and the model training time can be considered as being less important. The Sugeno fuzzy model obtained by using a different cluster radius for each data dimension generates a very low modeling error, and if the practical aspects mentioned previously are taken into consideration, it represents the best predictive model configuration. An

analysis of a fuzzy rule base can also provide insights into the model behaviour, allowing early detection of instances when the prediction is not accurate.

Also, for the first order Sugeno model with clustering performed using a different cluster radius for each data dimension, the search interval for the clustering parameters was narrowed to avoid an extremely long computing time for identifying the best model. A more precise model might have been obtained had the combinations of clustering parameters been performed on the full search interval.

Both the best Sugeno and neural network models presented previously can be further improved by refining certain modeling parameters, such as, among others, the number of hidden layers and neurons, and increasing the search intervals for the clustering parameters. The modeling and validation performances of the best Sugeno model – the actual output values along with the model generated outputs – are illustrated in Figures 7-1 and 7-2, respectively.

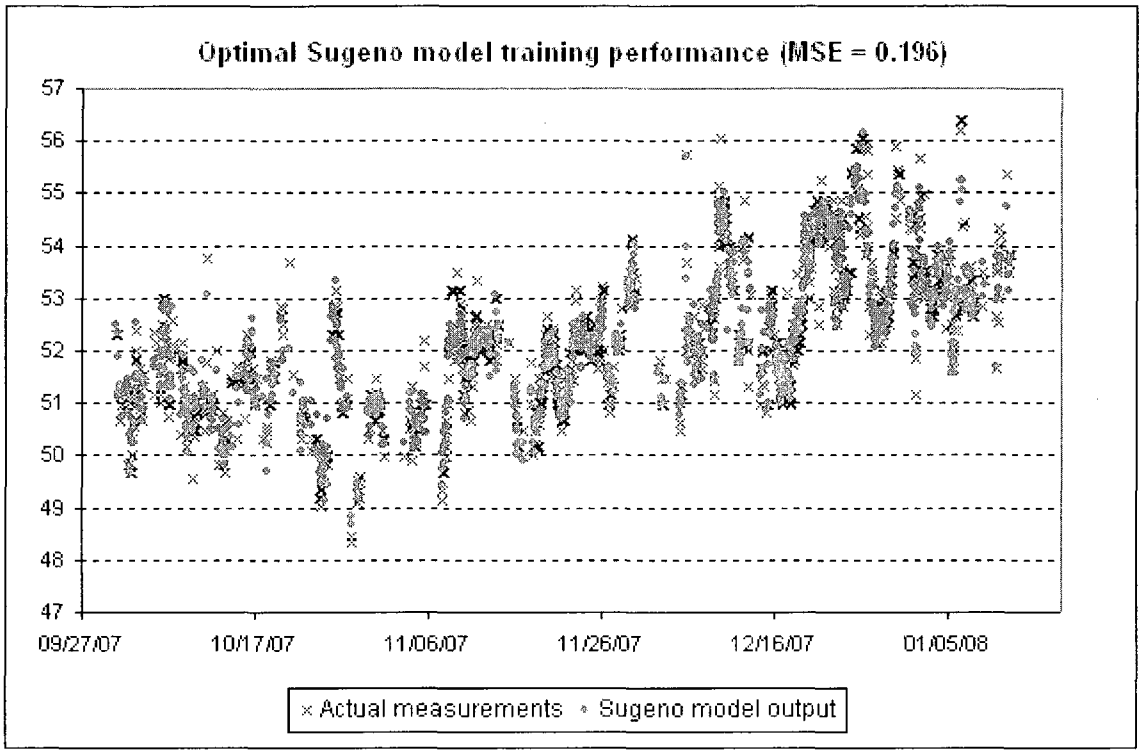


Figure 7-1 Modeling performance of the best Sugeno model

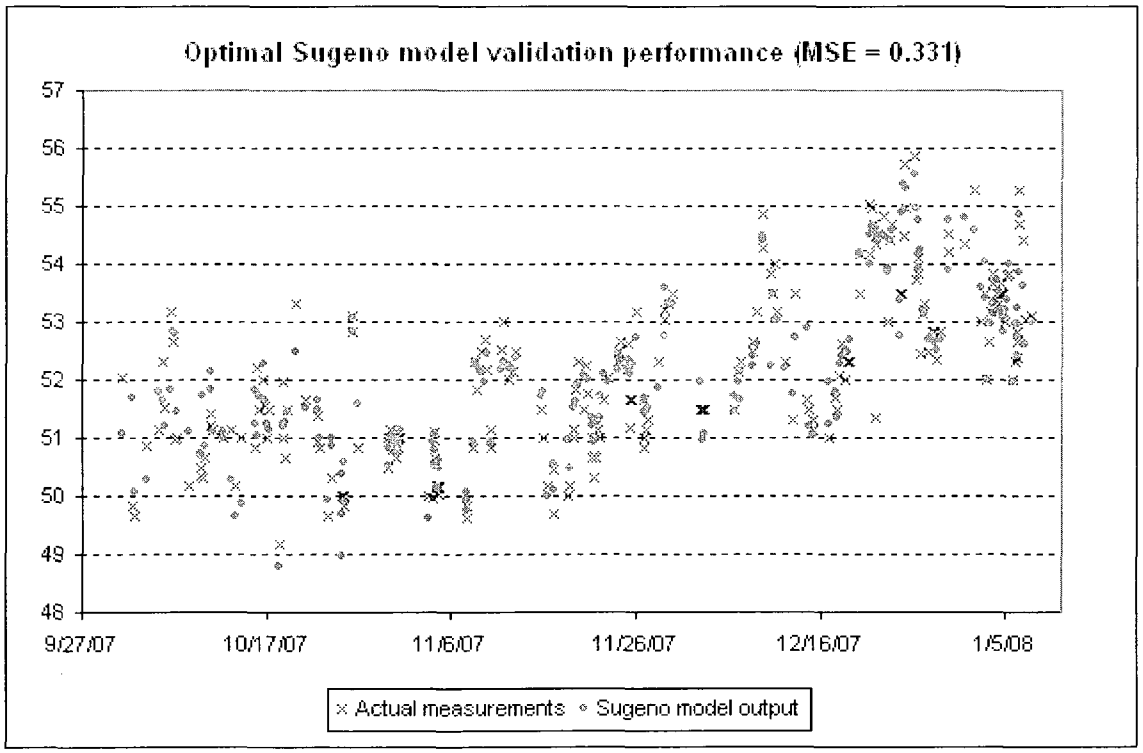


Figure 7-2 Validation performance of the best Sugeno model

8 EFFECTS OF DATA SCARCITY ON THE VALIDATION ERROR OF SUGENO FUZZY MODELS

8.1 Introduction

A closer inspection of the Sugeno fuzzy models revealed that the validation error behaves in an oscillatory manner, in the sense that its values would abruptly increase or decrease from one model to the next. This suggested that some models are not representative of the validation observations. For example, for the first order Sugeno models using the same cluster radius for all data dimensions and developed using the inputs selected according to the VIP values, the validation error increases nearly five times from the 35-rule model to the 36-rule model. This behaviour was also observed for the first order models using the same cluster radius for all data dimension and developed using the regression coefficient-based inputs, as well as for the second order models with both input sets. The magnitude and frequency of the validation error oscillations decrease in the case of the models obtained by using a different cluster radius for each data dimension, but they are still clearly present. An analysis of the validation error evolution for the first order Sugeno models with the same cluster radius for all data dimensions, using the *PROD* implication function and developed using the VIP inputs is presented next.

8.2 Validation error analysis

The evolution of the validation error values, for the 1 to 40-rule models is presented in Figure 8-1. It was considered that models containing more than 40 rules become over-fitted, since the validation error increases considerably for these models.

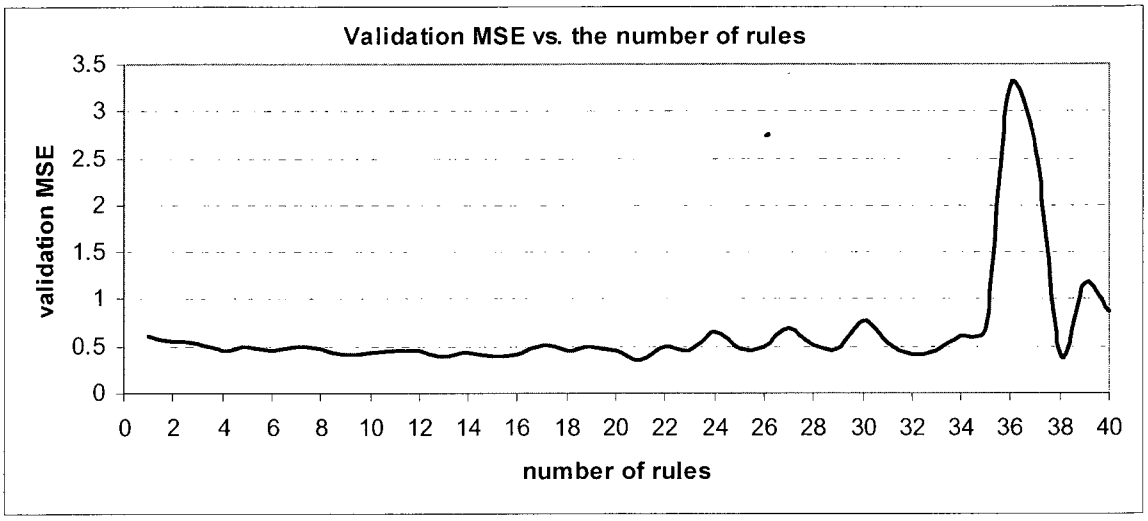


Figure 8-1 Oscillatory behavior of the validation error

The 30, 36, 37 and 39-rule models display the most oscillatory behaviour of the validation error. An inspection of the validation outputs reveals that for each of these models, only one observation displays a significantly higher deviation from its respective target value. This observation is responsible for most of the model's validation error, suggesting that it falls in a region that is not properly covered by the model's rule base. The absolute validation error for these models is presented on the plots below: figures 8-2, 8-4 and 8-5 show that the same observation – observation number 163 from the validation database – displays the greatest error among all validation observations for the 30, 37 and 39-rule models, respectively. Figure 8-3 shows the deviation of the observation 210 being responsible for most of the validation error of the 36-rule model.

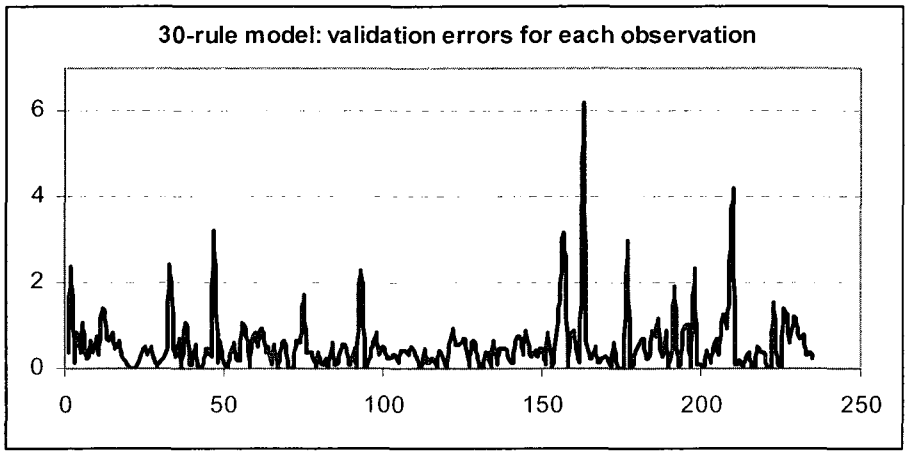


Figure 8-2 Validation error of each observation for the 30-rule model

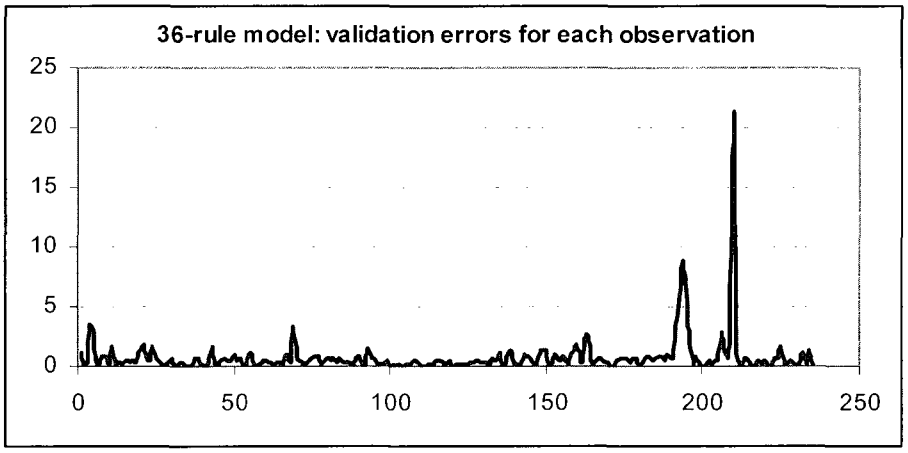


Figure 8-3 Validation error of each observation for the 36-rule model

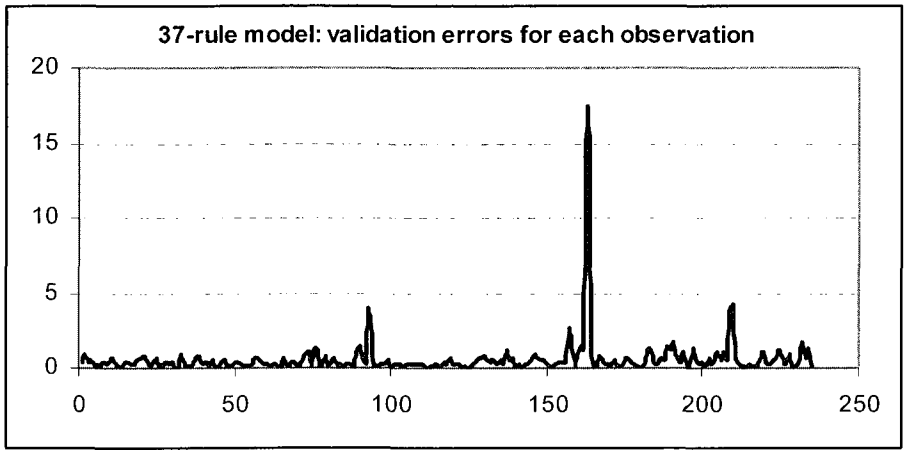


Figure 8-4 Validation error of each observation for the 37-rule model

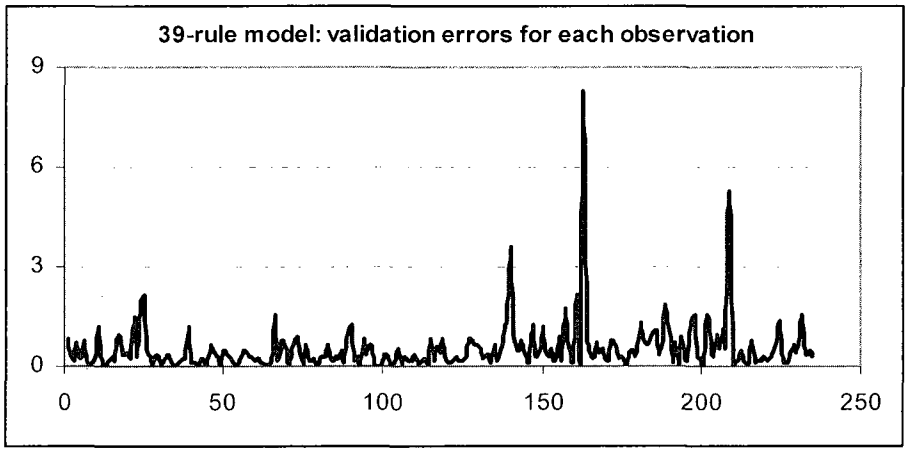


Figure 8-5 Validation error of each observation for the 39-rule model

For each model, the percentage of the validation error caused by the observation displaying the greatest deviation is presented in Table 8-1.

	Models			
	30 rules	36 rules	37 rules	39 rules
MSE of the complete validation dataset	0.706	3.282	2.583	1.175
MSE of the validation dataset with the observation with the largest deviation removed	0.511	1.273	0.606	0.465
% of model validation error caused by the observation with the largest deviation	27.6%	61.2%	76.5%	60.4%

Table 8-1 Impact of the validation observation with the largest deviation on the model validation error

For each of these models, a closer inspection of the rule weights and outputs for each observation identified previously as displaying a severe error reveals the following:

- the rule bases contain one rule with a weight fairly higher than the rest of the rules, as measured by the standard score of the rule – the standard score indicates how many standard deviations an observation is above or below the mean
- the output of the rule with the highest weight is well outside the interval [48.3 56.4], representing the minimum and maximum values of the output variable used in the training dataset
- if the rule with the maximum weight is removed, the mean and standard deviation of the remaining rule weights are fairly low

The rule base behaviour relative to the observation having the biggest validation error is presented in Table 8-2.

	Models			
	30 rules	36 rules	37 rules	39 rules
Output of the rule with the highest weight	85.86	75.95	669.85	-132.29
Weight of the rule with the highest weight	0.133	0.141	0.234	0.232
Standard score of the rule with the highest weight	4.4	5.83	5.76	3.74
Mean of the rule base not containing the rule with the highest weight	0.0055	0.00025	0.00082	0.031
Standard deviation of the rule base not containing the rule with the highest weight	0.016	0.00071	0.0033	0.042

Table 8-2 Rule base behaviour

These findings indicate that for each of these observations, there is only one rule having an impact on the modeling result, as suggested by the low mean and standard deviation values of the rule base not containing the rule with the highest weight. The output of the rule with the maximum weight is not representative of the training data, so therefore the model result will not be accurate.

In order to have a complete rule-base, any input data vector should fire at least one rule with a relative high weight [31]. As presented in Table 8-2, the highest rule weights for the models presented are below 0.25.

The rule base of some models cannot handle certain validation observations that appear to fall in a model region covered mainly by one rule. Therefore, this rule is most responsible for the system output. The presence of regions not properly covered by the model rule base is probably caused by training data scarcity: if there is not enough data to

cover the entire training space, the model does not have the required information to accurate generalize for some validation observations.

For a soft sensor industrial application, this can become an important issue, since the prediction might be inaccurate for the real-time measurements used as the model inputs that happen to fall in a region not properly covered by the rule base. The brief analysis presented in this chapter showed that inspection of the rule behaviour of a fuzzy system can provide an insight into the model. By analyzing the rule weights and outputs of each incoming real-time observation, criteria can be developed for detecting measurements for which the model is not representative. The validation error analysis presented previously – output of the rule with the maximum weight, mean and standard deviation values of the remaining rules – could be further investigated. A deeper analysis of the model rule base can also produce other criteria that can be used in order to provide users with an advanced warning concerning the soft sensor accuracy.

9 SUMMARY, CONCLUSIONS AND FUTURE WORK

9.1 Summary

Soft sensors represent a valuable tool for industrial users to optimize the process operation. Predictive models using measurements readily available can be developed to infer the value of critical process variables not measured in real-time, and allow an early detection of abnormal process behaviour. Historical process data related to the black liquor recovery circuit from a Canadian kraft pulp and paper mill is used to develop soft sensors models for the black liquor solid content at the concentrator feed. Currently, this solid content value is not measured on-line; it is obtained once every 8 hours trough a laboratory analysis of a black liquor sample.

The historical process data is cleaned by removing irrelevant variables and observations not representative of a normal operating regime. The cleaned dataset contains 144 variables: 143 variables to be used as model inputs, and the output variable – the solid content. For practical reasons related to model development and soft sensor industrial implementation, an input set containing 10 variables is used to build the models. Two selection criteria are used to select the top 10 variables most relevant to the prediction: the significance in modeling the input space and predicting the output, indicated by the so-called VIP value, and the significance solely in predicting the output, indicated by the absolute magnitude of the regression coefficients. A PLS analysis is performed on the complete dataset, and 2 subsets containing the ten variables with the highest VIP and regression coefficient values are formed. Their composition is quite different, as only two variables are common to both datasets. Both datasets are made up of variables related to

the flow of steam and black liquor in the evaporator train. From each dataset, 20% of the observations are used as validation data. These 10-variable datasets are used as inputs to predictive models developed using PLS, Sugeno-type fuzzy and neural network methods. The PLS analysis performed on the complete dataset also reveals that there is no significant nonlinearity or lagging between the input and output variables. Missing values are introduced in the dataset during the cleaning process by eliminating unwanted data. Prior to modeling, all observations containing at least one missing value are eliminated. The modeling and validation datasets obtained using both input selection criteria contain 960 and 240 observations, respectively: 80% of the available data is used for training, while the remaining 20% is used for validation.

First and second order Sugeno models are developed. The rule antecedents are obtained by clustering the input-output data, using either the same cluster radius for all data dimensions, or a different cluster radius for each dimension. The rule weights are calculated using either the *Product* or *Minimum* implication function. The best model is selected from all possible combinations of the clustering parameters. A reduced search interval for the clustering parameters is used in the case of clustering with a different cluster radius for each dimension, in order to avoid an extremely long computing time.

Neural network models with one and two hidden layers are computed. Two transfer functions of the hidden layer are tested: logarithmic sigmoid and hyperbolic tangent. A linear output layer transfer function is used in all cases. For each hidden layer, the number of neurons is increased from 5 to 20.

An Adaptive neuro-fuzzy (*ANFIS*) model is developed for each input dataset, in order to further optimize the first order Sugeno models developed using the same cluster radius for all data dimensions and the *PROD* implication function.

No combinations of modeling parameters were required for the PLS models; for each input set, the best PLS model contains 3 principal components.

In total, 8 different model configurations are developed for each set of input variables: 1 PLS configuration, 4 Sugeno-type fuzzy configurations, 2 neural network configurations and 1 ANFIS configuration. The performance of all those models is compared using the modeling and validation errors.

9.2 Conclusions

This work presents the development of a predictive model for the black liquor solid content at the concentrator feed, using historical process data related to the black liquor recovery circuit from a Canadian kraft pulp and paper mill. The objectives are to determine if 10 input variables are sufficient for developing an accurate soft sensor, to determine which PLS-based input selection criterion leads to the development of the best predictive model, and to determine which modeling method generates the smallest error.

The training and validation errors of the models presented are relatively low, indicating that for the application described in this work, 10 predictors are sufficient for developing an accurate soft sensor.

The models developed using the inputs selected according to their ability to both explain the input space and predict the output variable consistently outperform, both from a

training and validation error perspective, the models developed using the inputs selected according to their ability to only predict the output variable. Therefore, is concluded that, for the application described in this study, the former method of input selection leads to the development of the more accurate soft sensor models.

The Sugeno-type fuzzy logic and neural network-based models outperform the PLS models. The best 2-layer neural network generates the lowest training error, but the first order Sugeno fuzzy model with clustering performed using a different cluster radius for each data dimension generates the lowest validation error. Since an industrial soft sensor uses process measurements not used in the training process as inputs to the predictive model, it can be concluded that the first order Sugeno fuzzy model with clustering performed using a different cluster radius for each data dimension is the best model. The computing time can be considered as being a less important criterion, since the model training is performed once, off-line. Moreover, for all the model configurations studied, the real-time calculation of the soft sensor output takes less than 5 seconds.

Since it is shown that 10 predictors can be successfully used to develop an accurate soft sensor, and the input selection method as well as the model configuration leading to the best predictive performance is identified, it is considered that the objectives of this study are met. The developed soft sensor model can be used to provide operators with a real-time estimation of the black liquor solid content, allowing for rapid corrective action if the solid content drops bellow a critical threshold. This prevents loss of equipment efficiency due to scaling, reduced frequency of shutdowns for cleaning purposes, and a reduced energy consumption.

9.3 Future work

The soft sensor models developed in this work use a linear statistical-based method – PLS – to select a relevant set of variables to be used as inputs to predictive models developed using non-linear techniques, such as fuzzy logic and neural networks. Identifying an input selection method specific to the technique used for developing the model might increase its predictive performance; for example, for a fuzzy logic-based predictive model, the significant inputs are selected using fuzzy logic-based criteria.

The computing time required to identify the clustering parameters leading to the best Sugeno model can become an obstacle in developing more precise and complex models. Developing a methodology for efficiently identifying shorter optimal search intervals for the Sugeno model clustering parameters can facilitate the development of more accurate models, by considerably reducing the training time.

The validation error analysis presented previously should be further investigated, to determine precise criteria for identifying real-time measurements for which the model cannot produce an accurate prediction.

References

1. Fortuna, L., Graziani, S., Rizzo, A., and Xibila, M., *Soft Sensors for Monitoring and Control of Industrial Processes.* Advances in Industrial Control. 2007: Springer.

2. Eriksson, L., Johansson, E., Kettaneh-Wold, N., Trygg, J., Wikstrom, C., and Wold, S., *Multivariate and Megavariate Data Analysis, Basic Principles and Applications (Part I)* 2nd ed. 2006: Umetrics AB.

3. Champagne, M., Amazouz, M., and Platon, R., *The application of soft sensors in the pulp and paper and cement manufacturing sectors for process and energy performance improvement.* 2005, CanmetENERGY - Varennes.

4. Ahvenlampi, T., and Kortela., U., *Clustering algorithms in process monitoring and control application to continuous digesters.* Informatica, 2005(29): p. 101-109.

5. Abonyi, J., Nagy, L. and Szeifert, F. *Takagi-Sugeno Fuzzy Control of Batch Polymerization Reactors.* in *Proceedings of the 2nd Online World Conference on Soft Computing.* 1997.

6. Bonissone, P., and Goebel, K. *When will it break? A Hybrid Soft Computing Model to Predict Time-to-break Margins in Paper Machines.* in *Proceedings of SPIE 47th Annual Meeting, International Symposium on Optical Science and Technology.* 2002.

7. Merikoski, E., Laurikkala, S., and Koivisto, H. *An adaptive neuro-fuzzy inference system as a soft sensor for viscosity in rubber mixing process.* in *WSES/IEEE International Conference on Neural, Fuzzy and Evolutionary Computation.* 2001.

8. Wold, S., and Kettaneh-Wold, N. , *Improving pulp and paper process diagnostics and knowledge by means of multivariate analysis (MVA).* Pulp and Paper Canada, 2003. **104**(5): p. 48-50.

9. Engin, S., Kuvulmaz, J., and Ömürlü, V.E. *Modeling of a Coupled Industrial Tank System with ANFIS.* in *Mexican international conference on artificial intelligence.* 2004.

10. Macías-Hernández, J., Angelov, P., and Zhou, X., *Soft Sensor for Predicting Crude Oil Distillation Side Streams using Evolving Takagi-Sugeno Fuzzy Models.* Systems, Man and Cybernetics, 2007. 7(10): p. 3305 - 3310.

11. Dayal B.S., M.J.F., Kildaw, R. and Marcikic, S., *Application of Feedforward Neural Networks and Partial Least Squares Regression for Modelling Kappa Number in a Continuous Kamyr Digester.* Pulp and Paper Canada, 1994. **95**(1): p. 26-32.

12. Aminian, J., and Shahhosseini, S., *Evaluation of ANN modeling for prediction of crude oil fouling behavior.* Applied Thermal Engineering, 2008. **28**(7): p. 668–674.

13. Radhakrishnan, V.R., et al, *Heat exchanger fouling model and preventive maintenance scheduling tool* Applied Thermal Engineering, 2007. **27**(17): p. 2791–2802.

14. Ahmed, F., Nazi , S., and Yeo, Y. K., *A recursive PLS-based soft sensor for prediction of the melt index during grade change operations in HDPE plant* Korean Journal of Chemical Engineering, 2009. **26**(1): p. 14-20.

15. Qin, S.J., Yue, H., and Dunia, R. *A Self-Validating Inferential Sensor for Emission Monitoring.* in *1997 American Control Conference*. 1997.

16. Bolf, N., Ivandic, M., and Galinec, G., *Soft sensors for crude distillation unit product properties estimation and control.* Control and Automation, 2008. **25**(27): p. 1804 - 1809.

17. Delgado, M.R., Naga , E. Y., and de Arruda, L. V. R., *A neuro-coevolutionary genetic fuzzy system to design soft sensors* Soft Computing - A Fusion of Foundations, Methodologies and Applications, 2008. **13**(5): p. 481-495.

18. Keski-Säntti, J., *Neural Networks in the Production Optimization of a Kraft Pulp Bleach Plant*, in *Faculty of Technology, Department of Process and Environmental Engineering* 2007, University of Oulu.

19. Facco, P., Doplicher, F., Bezzo, F. and Barolo , M., *Moving average PLS soft sensor for online product quality estimation in an industrial batch polymerization process.* Journal of Process Control, 2008. **19**(3): p. 520-529.

20. Szladow, A.J., *Application of Artificial Intelligence Technology to Increase Productivity, Quality and Energy Efficiency in Heavy Industry*, D.o.N.R. Canada, Editor. 1995.

21. Hand, D.J., Mannila, H., and Smyth, P., *Principles of Data Mining.* 2001: MIT Press.

22. Wise, B.M. (2004) *Properties of Partial Least Squares (PLS) Regression, and Differences between Algorithms*, http://www.eigenvector.com/Docs/Wise_pls_properties.pdf

23. Zadeh, L.A., *Fuzzy sets.* Information and Control, 1965. **8**: p. 338-353.

24. Sugeno, M., and Takagi, T., *Fuzzy Identification of Systems and its Application to Modeling and Control.* IEEE Transactions on Systems, Man and Cybernetics, 1985. **15**(1): p. 116-132.

25. *Fuzzy Logic Toolbox User's Guide*. 2001: The Mathworks.

26. Chiu, S.L., *Fuzzy Model Identification Based on Cluster Estimation.* Journal of Intelligent and Fuzzy Systems, 1994. **2**: p. 267-278.

27. Demirli, K., and Packirisamy, M., *Higher Order Fuzzy System Identification with Subtracting Clustering.* Journal of Intelligent Fuzzy Systems, 2000. **9**(3-4): p. 129-158.

28. Haykin, S., *Neural Networks: A Comprehensive Foundation*. 2nd ed. 1999: Tom Robbins.

29. Demuth, H., Beale, M., and Hagan, M., *Neural Network Toolbox User's Guide*. 2006: The Mathworks.

30. Chapman, P., et al, *A step-by-step data mining guide*. 2000, CRISP-DM: Cross Industry Standard Process for Data Mining.

31. Demirli, K., Cheng S. K., and Muthukumaran, P., *Subtractive clustering based modeling of job sequencing with parametric search.* Fuzzy Sets and Systems, 2003. **137**(2): p. 235-270.

32. MacGregor, J. (2007) *Using Multivariate Methods for Process Monitoring and Soft Sensors*, http://www.prosensus.ca/white-papers/using-multivariate-methods-for-process-monitoring-and-soft-sensors.