ซอฟต์เซ็นเซอร์สำหรับการประมาณค่าคุณภาพของผลิตภัณฑ์ในหน่วยไซโคลเฮกซะโนน

นายประพนธ์ เขมะจันตรี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมเคมี ภาควิชาวิศวกรรมเคมี
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549
ISBN 974-14-2094-3

SOFT SENSOR FOR QUALITY ESTIMATION OF PRODUCT IN CYCLOHEXANONE UNIT

Mr. Praphon Kemachuntree

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Chemical Engineering
Department of Chemical Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2006

ISBN 974-14-2094-3

Thesis Title	SOFT SENSOR FOR QUALITY ESTIMATION OF PRODUCT IN CYCLOHEXANONE UNIT
By	Mr. Praphon Kemachuntree
Field of Study	Chemical Engineering
Thesis Advisor	Assistant Professor Montree Wongsri, D.Sc.

Accepted by the Faculty of Engineering, Chulalongkorn Univesity in Partial Fulfillment of the Requirements for the Mater's Degree

..... Dean of the Faculty of Engineering
(Professor Direk Lavansiri, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Professor Piyasan Praserthdam, Dr.Ing.)

..... Thesis Advisor
(Assistant Professor Montree Wongsri, D.Sc.)

..... Member
(Associate Professor Pornpote Piumsomboon, Ph.D.)

..... Member
(Assistant Professor Muenduen Phisalaphong, Ph.D)

..... Member
(Soorathep Kheawhom, Ph.D.)

ประพนธ์ เขมะจันตรี : ซอฟต์เซ็นเซอร์สำหรับการประมาณค่าคุณภาพของผลิตภัณฑ์ในหน่วยไซโคลเฮกซะโนน. (SOFT SENSOR FOR QUALITY ESTIMATION OF PRODUCT IN CYCLOHEXANONE UNIT) อ. ที่ปรึกษา : ผศ. ดร. มนตรี วงศ์ศรี, จำนวนหน้า 111 หน้า. ISBN 974-14-2094-3.

การวัดอย่างแม่นยำของตัวแปรคุณภาพเป็นสิ่งที่สำคัญสำหรับการเฝ้าตรวจติดตามที่สมบูรณ์และงานที่สำคัญทางการควบคุมระบบในกระบวนการทางเคมี น่าเสียดายที่เครื่องมือวิเคราะห์หรือฮาร์ดแวร์เซ็นเซอร์สำหรับการวัดตัวแปรสำคัญที่เหมาะสมตามต้องการมีจำนวนไม่มากและยุ่งยากในการดูแลรักษา ยิ่งกว่านั้นเครื่องมือวัดดังกล่าวก็มีข้อจำกัดด้วยเช่นกันได้แก่ราคาที่สูงและเวลาหน่วงของเครื่องมือที่มาก ดังนั้นจึงจำเป็นต้องใช้ซอฟต์เซ็นเซอร์เพื่อที่จะประมาณค่าตัวแปรคุณภาพโดยใช้ตัวแปรทุติยภูมิอื่นๆที่สามารถวัดค่าได้โดยตรง ในงานวิจัยนี้ใช้วิธีการการประมาณค่าสามวิธีโดยอาศัยแบบจำลองเชิงประสบการณ์ ซึ่งได้แก่ข่ายงานนิวรัลแบบเคลื่อนไปข้างหน้าหลายชั้น การถดถอยแบบกำลังสองน้อยสุดแบ่งส่วนและข่ายงานนิวรัลกำลังสองน้อยสุดแบ่งส่วน ถูกใช้เพื่อที่จะสร้างซอฟต์เซ็นเซอร์ที่สามารถประมาณผลิตภัณฑ์ยอดหอกลั่นในหน่วยไซโคลเฮกซะโนนโดยใช้เครื่องวัดอุณหภูมิที่มีอยู่ภายในหอกลั่น อย่างไรก็ตามการสร้างแบบจำลองซอฟต์เซ็นเซอร์โดยอาศัยข้อมูลจริงจากโรงงานเพียงอย่างเดียวนั้นมีข้อจำกัดที่เป็นปัญหาเพราะไม่สามารถทำงานในช่วงการทำงานที่กว้างได้เนื่องจากข้อมูลจากโรงงานมีการตอบสนองที่ราบเรียบหรือมีการเปลี่ยนแปลงที่น้อย เพื่อที่จะจัดการกับปัญหาดังกล่าว งานวิจัยนี้ใช้สองแหล่งข้อมูลคือข้อมูลโรงงานจริงและข้อมูลประดิษฐ์ช่วงกว้างเพื่อสร้างซอฟต์เซ็นเซอร์ สองแหล่งข้อมูลนี้ถูกผสมรวมกันเพื่อที่จะคำนวณค่าพารามิเตอร์ของแบบจำลองซอฟต์เซ็นเซอร์ชนิดกำลังสองน้อยสุดแบ่งส่วน สำหรับซอฟต์เซ็นเซอร์แบบข่ายงานนิวรัลแบบเคลื่อนไปข้างหน้าหลายชั้นกับแบบข่ายงานนิวรัลกำลังสองน้อยสุดแบ่งส่วน ข้อมูลประดิษฐ์ช่วงกว้างนี้ถูกใช้เพื่อที่จะฝึกสอนขั้นเริ่มให้กับค่าพารามิเตอร์ของแบบจำลองซอฟต์เซ็นเซอร์และใช้ข้อมูลโรงงานจริงสำหรับปรับแก้ค่าพารามิเตอร์อีกครั้ง ผลการทดลองพิสูจน์ว่าซอฟต์เซ็นเซอร์ทุกแบบให้ผลของสมรรถนะการประมาณค่าเป็นที่น่าพอใจและซอฟต์เซ็นเซอร์แบบข่ายงานนิวรัลแบบเคลื่อนไปข้างหน้าหลายชั้นให้ผลของสมรรถนะการประมาณค่าที่แม่นยำกว่าทั้งสองแบบที่ใช้กำลังสองน้อยสุดแบ่งส่วน

ภาควิชา วิศวกรรมเคมี ลายมือชื่อนิสิต.....ประพนธ์ เขมะจันตรี.....
สาขาวิชา วิศวกรรมเคมี ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2549

##4770339021 : MAJOR CHEMICAL ENGINEERING

KEY WORD : SOFT SENSOR/ ARTIFICIAL NEURAL NETWORKS/ PARTIAL LEAST SQUARES REGRESSION

PRAPHON KEMACHUNTREE : SOFT SENSOR FOR QUALITY ESTIMATION OF PRODUCT IN CYCLOHEXANONE UNIT THESIS ADVISOR : ASST. PROF. MONTREE WONGSRI, D.Sc., 111 pp. ISBN 974-14-2094-3.

Accurate measurement of quality variables are importance for the complete quality monitoring and control tasks in chemical process. Unfortunately, few analysers or hardware sensors for measuring the key variables are available and difficult to maintain. Moreover, these have also limitations such as very high cost and the large time delays. Therefore, it is necessary to utilize soft sensors to estimate the quality variables using other directly measurable secondary variables. In this work, three estimating approaches by using empirical models being multilayer feedforward (MLFF) artificial neural networks (ANNs), partial least squares (PLS) regression, and neural network partial least squares (NNPLS) are exploited to build soft sensors able to estimate the top product of the distillation column in the cyclohexanone unit using available temperature measurements in the column. However, buildings of soft sensor models using only real plant data have the hard limitation since they cannot implement in wide range estimation because of the plant data having smooth responses or small data variations. In order to handle this problem, this work used two data sources which were real plant data and wide range simulated data for constructing the soft sensors. These two data sources were mixed together to calculate parameters of the soft sensor model based on PLS model. For MLFF and NNPLS model, the wide range simulated data were used to pre-train or pre-calibrate parameters of soft sensor models and used the real plant data for find-tuning the parameters again. In case study, the results proved that all of the soft sensors showed satisfactory estimating performances and soft sensor based on MLFF method gave better estimating performances than both of the PLS methods.

Department Chemical Engineering Student's Signature Praphon Kemachuntree

Field of Study Chemical Engineering Advisor's Signature Montree Wongsri

Academic Year 2006

ACKNOWLEDGEMENTS

I would like to thanks and expresses my profound gratitude to my advisor, Asst.Prof.Dr. Montree Wongsri for his continuous guidances, suggestions and supports throughout the thesis study.

I would like to thank my parent who encourage and help during the thesis study.

I gratefully appreciate to Thai Caprolactam Public Co., Ltd. for supporting the important case study.

Finally, I would like to thank all of the members of the Control and Systems Engineering Center in Chulalongkorn University. It has been meaningful to be here and work with them.

CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER I

INTRODUCTION

This chapter is an introduction of this research. It consists of importance and reason for research, research objectives, scopes of research, contributions of research, activity plans and the research contents.

1.1 Importance and reason for research

It is a fact that the developed chemical processes of industrial plants must comprise special focus on quality standard production and pollution phenomena in social environments in order to acquire high product quality with the lowest environmental issues. The industrial standards enforce hard constraints of product specification and pollution. According to these limits, increasingly high performance control strategies are therefore required. For increasing efficient control systems, the developments of monitoring, predicting, and primary fault detection are regarded. The reliable measurements of quality variables are one of the main points in these requirements because there can achieve successful real-time monitoring and high control performance in chemical processes operation. Unfortunately, reliable hardware sensors or analyzers have been obstructed by very high investigation with maintenance costs of on-line measurement devices and long time delays, which cause problems to install equipment in processes. Owing to these limitations, many works have been devoted to the development of techniques for reducing the cost of on-line measurements and for designing reliable sensors via suitable algorithms in order to emulate the behavior of particular chemical process for producing real-time reliable estimation of indirectly measurable variables on the basis of available operational data. Such algorithms are usually known as soft sensors.

A soft sensor (inferential model) is mathematical algorithm estimating the qualities of interested variables by using their correlation with available data, such as temperature, pressure, flow rate, and other variables. In the field of soft sensors, the methods for building soft sensors can be grouped into two categories that are a first principal model-based and an empirical model-based. It is unfortunate that if the soft sensors with first principal model-based are enough accurate to represent real chemical systems, the models of first principal must be complicated formulas that result in very difficult calculation to find final solutions. Sometimes, the models cannot provide final solutions or they may take a long time to compute the final solutions. On account of these problems, the soft sensors with empirical model based are wildly used for building soft sensors because they use particular the data which are input data (available data) and output data (estimated data). The key of empirical models are algorithm which finds the relationship between input and output of chemical systems without consideration about inside their systems or they can be called black box models, which certainly find the final solutions.

It is various alternatives for empirical models that the two approaches are the most popular of empirical models being Partial Least Squares (PLS) Regression fields and Multilayer Feedforward with Backpropagation (MLFF with BP) for Artificial Neural Networks (ANNs). For the first method, PLS is a regression method that can handle collinearity problem or highly correlated input data (similar to a soft sensor problem) because this method transforms original data to new data that can represent old data by lower dimension (Geladi & Kowalski, 1986). In another advantage, the PLS can reduce uncertainty or noise of measurement by decreasing the number of input variables. Thus it is important to keep the number of variables as low as possible. In that sense, PLS will give the minimum number of variables that is necessary (Höskuldsson, 1988). The PLS algorithms have been applied to many problems in chemical engineering fields. For instances, Zamprogna, Barolo & Seborg (2005) used the PLS regression method for building one of the soft sensors to estimate compositions in batch distillation. For the second method, MLFF with BP is one of the widespread architecture in ANNs field since this method can tackle high nonlinearity problems by increasing the

number of neurons and/or hidden layers via suitable neural structure for their problems. Moreover, MLFF with BP have many alternatives to improve performance of neural architectures such as activated function, learning function, and so on. The MLFF have also been wildly applied in various fields. For example in chemical engineering field, Fortuna, Graziani & Xibilia (2005) desired neural structures for monitoring quality of product in the debutanizer column. For other interesting works and their applications, PLS methods and ANNs were provided in the CHAPTER II.

In summary, this research built the soft sensors which used mathematical algorithm with PLS methods and MLFF with BP produced reliable estimation models of product quality by using their correlation with available sets of data.

1.2 Research objectives

The objective of our research is to apply the using of soft sensor in order to estimate the product quality for cyclohexanone/cyclohexanol distillation section at cyclohexanone unit in Thai Caprolactam Public Company Limited.

1.3 Scopes of research

1. Description and data of cyclohexanone unit are obtained from the Thai Caprolactam Public Company Limited.

2. Partial least squares (PLS) or linear Partial Least Squares (linear PLS), Neural Network Partial Least Squares (NNPLS), and Multilayer Feedforward with Backpropagation (MLFF with BP) for Artificial Neural Networks (ANNs) are used to build the soft sensor.

3. Program MATLAB and commercial process simulator-HYSYS are used in this work.

1.4 Contributions of Research

The contribution of this work utilizes the selected algorithms for soft sensors in order to apply in the estimation of product quality of production in cyclohexanone/cyclohexanol distillation section. The algorithms are capable of improving accuracy of prediction.

1.5 Activity Plans

1. The first plan, various literatures related to soft sensor building and its applications were studied.
2. We investigated model building techniques which were Partial Least Squares (PLS), Neural Network Partial Least Squares (NNPLS), and Multilayer Feedforward with Backpropagation (MLFF with BP) for building soft sensors.
3. We obtained the important data being input data (measurable variable) and output data (product quality) from the Thai Caprolactam Public Company Limited.
4. We applied the soft sensors based on Partial Least Squares (PLS), Neural Network Partial Least Squares (NNPLS), and Multilayer Feedforward with Backpropagation (MLFF with BP) to cyclohexanone/cyclohexanol distillation section.
5. The results between soft sensors based on Partial Least Squares (PLS), Neural Network Partial Least Squares (NNPLS), and Multilayer Feedforward with Backpropagation (MLFF with BP) were compared.
6. We analyzed and concluded our research.

This thesis is divided into five chapters. Moreover, information of each chapter is shown as a following.

A CHAPTER I is an introduction of this research. This chapter consists of research objective, scope of research, contribution of research, and activity plan.

A CHAPTER II reviews the introduction of soft sensors, the advantage of soft sensors, and applications of soft sensors based on Partial Least Squares (PLS) & Artificial Neural Networks (ANNs) field.

A CHAPTER III covers some background information of soft sensor and theory concerning with statistical method such as Principal Component Analysis (PCA), Partial Least Squares (PLS), Neural Networks Partial Least Square (NNPLS), and fundamentals of Artificial Neural Network (ANNs).

A CHAPTER IV describes about the profile of Thai Caprolactam Public Company Limited, the work description, and the design of soft sensors based on PLS, NNPLS & MLFF with BP. The simulation results and discussion are included in this chapter.

A CHAPTER V presents the conclusions and recommendations of this research.

This is follow by:

REFERENCES

- APPENDIX A: CORRESPONDING MATHEMATICS
- APPENDIX B: NORMALIZATION METHODS
- APPENDIX C: PARAMETER VALUES OF SOFT SENSORS
- APPENDIX D: EXAMPLES FOR SOFT SENSOR CALCULATIONS
- APPENDIX E: SAMPLES OF DATA SETS

VITA

CHAPTER II

LITERATURE REVIEWS

This chapter presents the literature reviews of various case studies for soft sensor applications. We present the interesting methods that are Partial Least Squares (PLS) methods and Multilayer Feedforward with backpropagation (MLFF with BP) for Artificial Neural networks (ANNs).

Soft sensors have been interested as mentioned in previous chapter. For these reasons, people who work in this field have studied many applications, especially empirical models. The general procedure of soft sensors is shown in figure 2.1.

Figure 2.1: The structure of soft sensor.

Form figure 2.1, soft sensors (inferential models) are tools that provide the relationship between available variables and target variables. Available variables are variables that are measured (data from chemical hardware sensors) such as temperature, pressure, flow rate, and so on. Moreover, target variables (unmeasured variables) are key variables or essential variables of quality control which were not directly measured. Then, if we can exactly predict the key variables at real time with some fault occurred, it is important to prevent some malfunction of chemical operation. Advantages of soft sensors were presented as next section.

2.1 Advantages of soft sensors

Some main advantages are presented as following:

- Soft sensors offer low cost alternative,
- Soft sensors can work in parallel with other chemical tools,
- Soft sensors can estimate target variables with real time operation, and
- Soft sensors can overcome the long time delays.

2.2 Applications of soft sensors

2.2.1 Soft sensors based on Partial Least Squares (PLS) Regression Methods

Principal Component Analysis (PCA) and Partial Least Squares (PLS) regression have attracted wide interests as robust methods for constructing empirical models, particularly when there are high dimensionality and collinearities in data. Especially, PLS and its variations have been applied to many practical regression problems in chemical engineering. Thor Mejdell & Sigurd Skogestad (1991a) used PLS method to estimate distillation compositions form multiple temperature measurements. The distillation columns were binary mixture and multicomponent mixture. Moreover, data were generated by model base. Finally, the results indicate that the estimator perform well in wide range of application. In the next paper, Thor Mejdell & Sigurd Skogestad (1991b) used PLS regression to estimate product for compositions on a high-purity pilot-plant distillation column. The result found that an estimators based on experimental data gave well performance over a wide range of operating points. Eliana Zamprogna, Massimiliano Barolo & Dale E. Seborg (2004) used this method to estimating product composition in batch distillation process using available temperature measurements. The result prove that the PLS estimators can provide accurate composition estimations for a batch distillation process. The computational requirements are very low,

which makes the estimators attractive for on-line use. In the next year, Eliana Zamprogna, Massimiliano Barolo & Dale E. Seborg (2005) used PCA to select the most suitable secondary process variables to be used as soft sensor inputs and they are used as input variables for the development of a regression model suitable for on-line implementation in batch distillation. H. Kamohara, A. Takinami, M. Takade, M. Kano, S. Hasebe & I. Hashimoto (2004) used developed soft sensor based on PLS to estimate impurity concentration in the industrial ethylene plant but it did not function well when the process is operated under conditions that have never been observed before. Jong Ku Lee & Chonghun Han (1999) used multivariate statistical methods to monitor and diagnose the operation of two industrial polymerization processes: a) a batch PVC process and b) a continuous slurry HDPE process. Multiway PCA has been used to develop a monitoring system for the batch PVC process. Another monitoring system for the HDPE process has been developed based on PLS techniques. The developed systems will be used to achieve their ultimate goal: to supply their customers with high quality product of minimum variations.

Although multivariate statistical methods based on linear projection, such as PCA and PLS can handle high dimensionality and collinearity, their major restriction is that only linear information is extracted from the data. Since many chemical processes are nonlinear in nature, it is desirable to have a robust method which can model any nonlinearity. At present, there are many nonlinear method available. For example, nonlinear partial least squares, artificial neural networks (ANNs) are well-known nonlinear regression methods. Sungyong Park & Chonghun Han (2000) propose a new nonlinear method that is conceptually simple and quite easy to use. This method has been motivated by locally weighted regression (LWR or loess) that estimates a regression surface through multivariate smoothing. They used this method to build a soft sensor for an industrial splitter column and an industrial crude column. Moreover they have shown that the proposed method gives a better or equal performances over other methods such as PLS, nonlinear PLS and artificial neural network.

2.2.2 Soft sensors based on Artificial Neural Networks (ANNs)

Artificial neural network which is one of the empirical methods has been wildly used in soft sensor field. V.R. Radhakrishnan & A.R. Mohamed (2000) studied the operation and control of blast furnaces because this work is difficult to measuring and controlling. The measurement of hot metal compositions require spectrographic techniques that is performed by off-line. Then, they used soft sensor to handle this problem. They used Artificial Neural Networks (ANNs) to estimate the outputs that are quality of the hot metal (& slug) and their composition. The soft sensor has been able to predict the variables with an error less than 3%. Adilson Jose' de Assis & Rubens Maciel Filho (2000) used this method to built model of soft sensor which provide on-line estimate of unmeasurable process variables from available sensors in bioprocess monitoring and control. Their performances depend on the measurement quality by hardware sensor and the estimation algorithm. The estimation techniques have been developed by four methods , namely, (1) estimation through elemental balances; (2) adaptive observer; (3) filtering techniques (Kalman filter, extended Kalman filter); and (4) artificial neural network. In last section, they studied about hybrid modeling using coupled first principles approach and ANN, and genetic programming. L. Fortuna, A. Rizzo, M. sulfur & M.G. Xibilia (2003) designed and used soft sensor for a Sulfur Recovery Unit (SRU) in a refinery. Four strategies for the soft sensor have been implemented and compared: Multi-Layer Perceptrons (MLP), Radial Basis Function neural network, Neuro-Fuzzy networks, and nonlinear Least-Squares (LSQ) fitting. The best performance is given by MLP and LSQ. L. Fortuna, S.Graziani & M.G. Xibilia (2005) deal with the design of neural based soft sensors to improve product quality monitoring and control in refinery by estimating the stabilized gasoline concentration (C5) in the top flow and the butane (C4) in the bottom flow of a debutanizer column. They studied three-step predictive dynamic neural models to evaluate in real time the top and the bottom product concentrations in column. Increasingly efficient control policies require a large set of data acquired the plants to be processed. The sensors are currently implemented in the plant and are used to monitor the production process, avoid-

ing the long delay introduced by the gas chromatographs. The availability of the gas chromatographs gave us the opportunity to check the quality of soft sensor estimation that is satisfactory by experts. In the same year, Pascal Dufour, Sharad Bhartiya, Prasad S. Dhurjati & Francis J. Doyle III (2005) study about a neural network-based strategy for detection of feedstock variations in a continuous pulp digester. Training and validation data sets are generated using a rigorous first principles model. Moreover, soft sensor can be applied to quality control in chemical process. Masahiro Ohshima & Masataka Tanigaki (2000) used soft sensor for polymer production plant, which integrates optimal control with on-line sensing. They used neural network in attempt to relating the instantaneous polymer properties with process variables.

CHAPTER III

THEORIES AND PRINCIPLES

This chapter presents the fundamentals of Principal Component Analysis (PCA), Partial Least Squares Regression (PLS), Neural Network Partial Least Squares (NNPLS), and Artificial Neural Networks (ANNs). All of the fundamentals are important and used to build soft sensors.

The main goal of a soft sensor is to provide accurate and reliable measurements of a process variable or parameter that are used for monitoring or control purpose in chemical processes (Albertos, P. and Goodwin, G.C. 2002). This goal can indicate that it should

- achieve a (near) relationship between involved input and output,
- reduce the measurement time and eliminate time delays,
- obtain the required accuracy,
- detect malfunction of chemical units, and
- minimize maintenance costs.

Soft sensors are the association of sensors (hardware) which allow on-line measurements of some process variables with an estimation algorithm (software) in order to provide reliable on-line estimation of the unmeasurable variables, model parameters with overcoming time delays of measurements. Three estimation techniques have been proposed in this work, namely Partial Least Squares (PLS) regression, Neural Networks Partial Least Squares (NNPLS), and Multilayer Feedforward with Backpropagation (MLFF with BP) for Artificial Neural Networks (ANNs). For the fundamentals of basic statistics and the basic matrix algebra corresponding to this work, they are provided in APPENDIX A.

3.1 Models building for soft sensors

3.1.1 Data Transformation or preprocessing step

Variables tend to have ranges that vary greatly from each other. Such differences in the ranges will lead to a tendency for the variable with greater range to have undue influence on the results.

Therefore, numerical variables should be normalized, to standardize the scale of effect each variable has in the results. There are several techniques for normalization. Let X refer to our original field value and X^* refer to the normalized value.

- **Min-Max Normalization**

  Min-max normalization works by seeing how much greater the field value is than the minimum value X_{min} and scaling this difference by the range (X_{max} represents the maximum value of X). That is,

  $$X^* = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

- **Z-score Standardization**

  Z-score standardization, which is very widespread in the world of statistical analysis, works by taking the difference between the field values and the field mean value (zero-mean) and scaling (unit-variance) this difference by the standard deviation (*std*) of the field values. That is,

  $$X^* = \frac{X - X_{mean}}{X_{std}} \tag{3.2}$$

In this work, these transformations method were used for preprocessing step. Graphically, an illustration of unit-variance and zero-mean is given in figure 3.1.

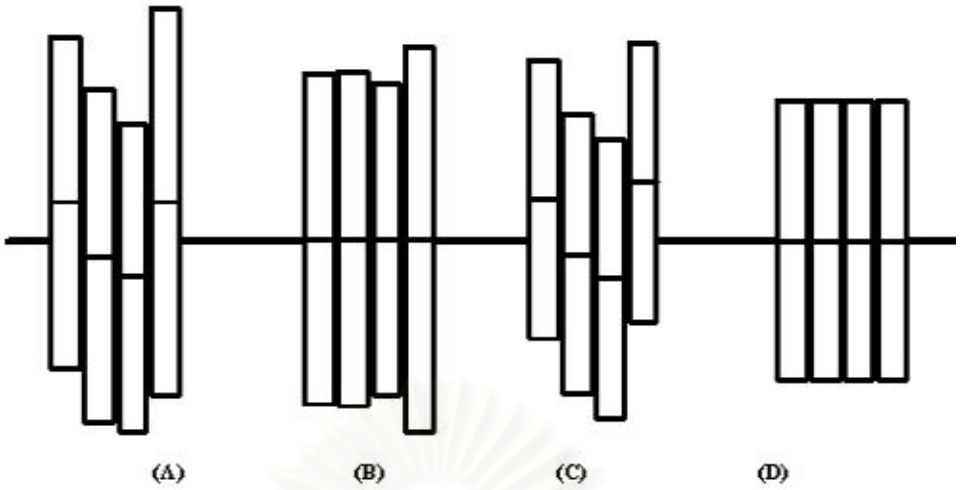

Figure 3.1: Data preprocessing. The data for each variable are represented by a variance bar and its center. (A) Most raw data look like this. (B) The result after zero-mean only. (C) The result after unit-variance only. (D) The result after zero-mean and unit-variance.

3.1.2 Procedures of models building

Before the model is built, it is convenient to tailor the data by preprocessing step. In general case, a procedure for building model consists of two steps.

1. The calibration or training step: This step attempt to make a model for a relationship between two groups of variables, often called dependent variables Y and independent variables X. The data set used for this step is called a calibration or training set. The model parameters are called regression coefficients or sensitivities. The model between Y and X can present as a following:

$$Y = f(X) \tag{3.3}$$

2. The prediction or test step: This step use sensitivities in calibration step and prediction set to predict values for the dependent variables. The data which are used in prediction step are different from data in calibration step.

In this work, the data in PLS, NNPLS, and ANNs are divided to two main set for using in building soft sensors and the data (X and Y) that are used for building are also normalized by scaling methods in section 3.1.1.

3.2 Principal Component Analysis (PCA)

In the some problems, the dimension of the input set is large, but the components of the vectors are highly correlated (redundant). It is useful in this situation to reduce the dimension of the input set. An effective procedure for performing this operation is principal component analysis (PCA). The PCA is one of the multivariate methods of analysis and has been used widely with large multidimensional data sets. The use of PCA allows the number of variables in a multivariate data set to be reduced, whilst retaining as much as possible of the variation present in the data set. This reduction is achieved by taking a variables x_1, x_2, $\cdots$, x_m and finding the combinations of these to produce principal components (PCs) PC_1, PC_2, $\cdots$, PC_a, which are uncorrelated. These PCs are also termed eigenvectors. The lack of correlation is a useful property as it means that the PCs are measuring different dimensions in the data. Nevertheless, PCs are ordered so that PC_1 exhibits the greatest amount of the variation, PC_2 exhibits the second greatest amount of the variation, PC_3 exhibits the third greatest amount of the variation, and so on. That is $\text{var}(PC_1) \geq \text{var}(PC_2) \geq \text{var}(PC_3) \geq \cdots \geq \text{var}(PC_a)$, where var ($PC_h$) expresses the variance of PC_i in the data set being considered. $\text{Var}(PC_h)$ is also called the eigenvalue of PC_h. When using PCA, it is hoped that the eigenvalues of most of the PCs will be so low as to be virtually negligible. Where this is the case, the variation in the data set can be adequately described by means of a few PCs where the eigenvalues are not negligible. Accordingly, some degree of economy is accomplished as the variation in the original number of variables (X variables) can be described using a smaller number of new variables (PCs).

In summary, the PCA technique has three important actions: first, it makes the orthogonal components of the input vectors (so that they are uncorrelated with each other). Second, it orders the resulting orthogonal components (principal components) so that those with the largest variation come first and, third, it eliminates those components that contribute the least to the variation in the data set. In general case, because each variable usually has different unit, the data for implementation must be scaled by following method.

3.2.1 Procedures for a principal components analysis

The analysis is performed on a data set of n variables (x_1, x_2, $\cdots$, x_n) for n individuals (X is scaled by Z-score standardization), as indicated in figure 3.2.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Figure 3.2: A general data matrix; its rows as observations (m), and its columns as variables (n).

From this data set, a corresponding squared covariance or correlation matrix can be calculated. For the covariance matrix, the equation is shown in appendix A.

The covariance matrix then has the following form.

$$C = \begin{bmatrix} S_{11}^2 & S_{12}^2 & S_{13}^3 & \cdots & S_{1n}^2 \\ S_{21}^2 & S_{22}^2 & S_{23}^3 & \cdots & S_{2n}^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ S_{1n}^2 & S_{2n}^2 & S_{2n}^3 & \cdots & S_{nn}^2 \end{bmatrix} \tag{3.4}$$

where S is the covariance matrix, S_{jk} is the covariance of variables x_j and x_k when $j \neq k$ and the diagonal element S_{jj} is the variance of variable x_j when $j = k$. The covariance matrix is used when the variables are measured on comparable scales.

PCA is thus concerned with finding the variances and coefficients of the data set, in other words, finding the eigenvalues and eigenvectors of the sample correlation matrix or the covariance matrix. Eigenvectors (PCs) and their associated eigenvalues can be calculated from the correlation matrix. The matrix C (correlation matrix) is reduced to a diagonal matrix λ by multiplying by particular orthonormal matrix P as following this equation.

$$P'CP = \lambda \tag{3.5}$$

The diagonal elements of λ_1, λ_2, $\cdots$, λ_a are called the characteristic roots, latent roots, or eigenvalues of C. The columns of P are called characteristic vectors, loading vector, or eigenvectors of C. The eigenvalues may be obtained from the following determinant equation called the characteristic equation (I is a identity matrix).

$$|C - \lambda I| = 0 \tag{3.6}$$

For the procedure of calculation a matrix P and matrix λ , reader can read additional detail in the book of Jackson, J. E. (1991) and Kreyszig, E. (1999).

Other important properties of eigenvectors or PCs are:

1. Eigenvectors can only be found for squared matrices.
2. Not all squared matrices have eigenvectors; however a correlation (or covariance) matrix will always have eigenvectors (the same number as there are variables).
3. Length does not affect whether a vector is a particular eigenvector, direction does.
4. Eigenvectors and eigenvalues always come in pairs.

Form previous sentences, assume we obtained.

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{3.7}$$

And each of the columns of P represented the eigenvector of C (C is covariance matrix of X)

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1a} \\ p_{21} & p_{22} & \cdots & p_{2a} \\ \vdots & \vdots & \vdots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{na} \end{bmatrix} \tag{3.8}$$

A principal component axis transformation will transform a correlated variables $x_1, x_2, \cdots, x_a$ into a new uncorrelated variables $t_1, t_2, \cdots, t_a$, the coordinate axes of these new variables being described by the each column of P (eigenvector) which make up the matrix P of direction cosines used in the following general transformation:

$$T = XP \tag{3.9}$$

or

$$T = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1a} \\ t_{21} & t_{22} & \cdots & t_{2a} \\ \vdots & \vdots & \vdots & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{ma} \end{bmatrix} \tag{3.10}$$

$$X = TP'; (P'P = I) \tag{3.11}$$

where the matrix T is called score matrix and P is called loading matrix.

All of the PCs are orthogonal, which can illustrate in geometrical interpretation of components in figure 3.3 (An example for two variables, in the two dimensional plane).

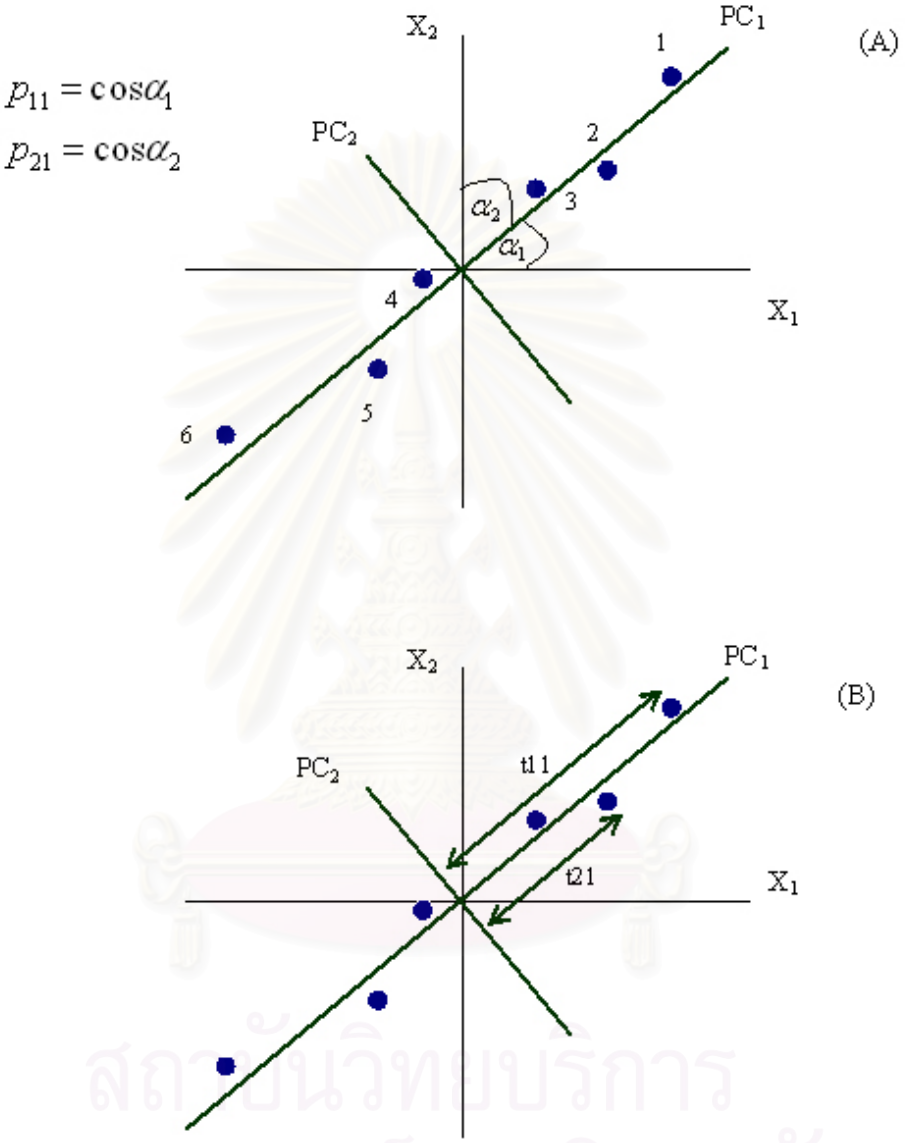

Figure 3.3: A principal component in the case of two variables. (A) loading are the angle cosines of the direction vector; (B) scores are the projections of the sample on the principal component directions (the data are mean-centering).

Another method for calculating a score matrix and a loading matrix is Nonlinear Iterative Partial Least Squares (NIPALS). These technique doses not calculate all the principal components at once. It calculates t_1 and p'_1 (t_h and p_h represent column vector (h) of matrix T and matrix P, respectively) from the X matrix. Then the outer product, $t_1 p'_1$ is subtracted from X and the residual E_1 is calculated. This residual can be used to calculate t_2 and p'_2:

$$E_1 = X - t_1 p'_1, E_2 = E_1 - t_2 p'_2, \cdots, E_h = E_{h-1} - t_h p'_h \tag{3.12}$$

The NIPALS algorithm is as follows:

1. take a vector x_j from X and call it t_h: $t_h = x_j$ (some columns of X)
2. calculate p': $p'_h = \frac{t'_h X}{t'_h t_h}$
3. normalize p'_h to length 1: $p'_{h(new)} = \frac{p'_{h(old)}}{\|p'_{h(old)}\|}$
4. calculate t_h: $t_h = \frac{X p_h}{p'_h p_h}$
5. compare the t_h used in step 2 with that obtained in step 4. If they are the same, stop (the iteration has converged). If they still differ, go to step 2.

After the first component is calculated, X in steps 2 and 4 have to be replaced by its residual. The iterations can continue until number of PCs is equal to number of variables in X.

The principal components of input and output set (scores and loading) are used in partial least squares (PLS) method to build the PLS model. For modeling, PLS approach is proposed in next section. Reader can find more details about principal component analysis and its applications in J. E. Jackson 1991.

3.3 Partial Least Squares Regression (PLS)

The PLS is an acronym for partial least squares projection to latent structures. PLS is a method of modeling relationships between one or more Y variables and one-to-many explanatory variables. PLS is gaining importance in many fields of chemistry, analytical, physical, clinical chemistry and industrial process control can be benefited from the use of this method. Partial least squares (PLS) models are based on principal components of both the independent data X and the dependent data Y. The central idea is to calculate the principal component scores of the X and the Y data matrix and to set up a regression model between the scores (and not the original data).

This work used linear PLS and NNPLS as methods to build soft sensors model. For these methodology, the basics of linear PLS and NNPLS have been proposed as follow:

3.3.1 Linear PLS identification

The PLS method is basically a particular multivariable regression algorithm which can handle correlated inputs and limited data. The detail of linear PLS algorithm are referred to the literature in P. Geladi and B.R. Kowalski 1986, A. Höskuldsson 1988.

Assume output has l variables, y_j $(j = 1, 2, \cdots, l)$ and input has n variables, x_i $(i = 1, 2, \cdots, n)$. Also assume d samples of data are observed, Then two matrices (Output and Input matrix) can be proposed as a following.

$$Y = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1l} \\ y_{21} & y_{22} & \cdots & y_{2l} \\ \vdots & \vdots & \vdots & \vdots \\ y_{m1} & y_{m2} & \cdots & y_{ml} \end{bmatrix} \tag{3.13}$$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{3.14}$$

The matrices X and Y are decomposed into linear models similar to principal components. The linear PLS decomposes matrices X and Y into products plus residual matrices (outer relation):

$$X = t_1 p_1' + t_2 p_2' + \cdots + t_a p_a' + E_a \tag{3.15}$$

$$Y = u_1 q_1' + u_2 q_2' + \cdots + u_a q_a' + F_a \tag{3.16}$$

where t and u are score vectors of principal components, p and q are loading vectors corresponding to corresponding components, a the number of factors extracted in the principal component space. E_a and F_a are residuals for X and Y matrices, respectively after a factors have been extracted. The outer relationship is defined as the relationship between E_h and F_h, ($h = 1, 2, \cdots, a$) where initially $E_0 = X$ and $F_0 = Y$. Generally, Eq. 3.15 and Eq. 3.16 can be written as:

$$X = TP' + E \tag{3.17}$$

$$Y = UQ' + F \tag{3.18}$$

Thus the matrix X is decomposed into a matrix T (the score matrix) and a matrix P' (the loadings matrix) plus an error matrix E. The matrix Y is decomposed into U and Q and the error term F. The above two equations formulate a PLS outer model. After the outer relation has been calculated, the score vectors are related by inner model (inner relation).

If here a linear relation exists between latent variable matrices U and T, i.e. $U = TB$, where matrix B is a diagonal matrix containing the regression coefficient for each PLS component along with its diagonal, it is called a linear PLS model. Moreover, X and Y must be pre-processed by Z-score Standardization (zero mean and unit variance). The approach can be described in the following sentences.

For each factor h

1. Use one column in Y as a starting vector for u_h, $u_h = Y_j$.

2. $w'_h = u'_h X / u'_h u_h$, norm w_h: $\|w_h\| = 1$.

3. $t_h = X w_h / w'_h w_h, q'_h = t'_h Y / t'_h t_h$, norm q_h: $\|q_h\| = 1$.

4. $u_h = Y q_h / q'_h q_h$

5. Check convergence on t_h or u_h, e.g. $\|t_h - t_{hold}\| / \|t\| < e$, if no convergence and if the number of iterations < maxiter, return to step 1.

6. $b_h = u'_h t_h / t'_h t_h$, $p'_h = t'_h X / t'_h t_h$.

7. Residuals: use E_h as X and F_h as Y in the next model dimension calculations. For block Y, $\hat{u}$ is calculated by bt:

$$E_h = X - t_h p'_h \tag{3.19}$$

$$F_h = Y - \hat{u} q'_h. \tag{3.20}$$

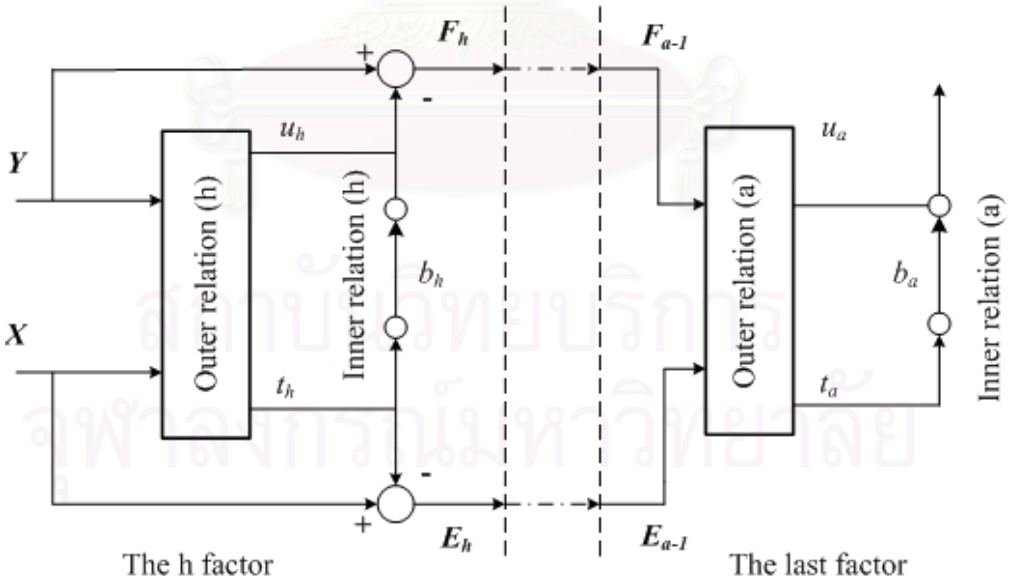

Figure 3.4: A schematic of the linear PLS model: each inner model (b_h relation) is performed by a linear regression.

Figure 3.4 shows a schematic of the PLS model. It shows that the input X is projected into the latent space by the input-loading matrix P obtaining the input scores T. Similarly, the output Y is projected into the latent space by the output-loading matrix Q obtaining the output scores U. A linear model captures the relationship between the input and output latent scores for each factor.

When the chemical process exhibits non-linear behavior, it is necessary to extend the PLS model structure to capture non-linearity. According to this problem, the extended PLS model for modeling static data using a quadratic relationship between the u_h and t_h as inner relation (S. Wold, N. Kettaneh-wold and B. Skagerberg 1989) and spline function for non-linear inner relation (S. Wold 1992). The Neural network partial least squares (S. J. Qin and T. J. MacAvoy 1992, 1996) called NNPLS has been developed. The NNPLS uses neural networks to model the inner relationship.

The NNPLS has been used to build a soft sensor. The static NNPLS combines the robust properties of the PLS and the nonlinear modeling of neural networks. For NNPLS procedures, they have been demonstrated as following.

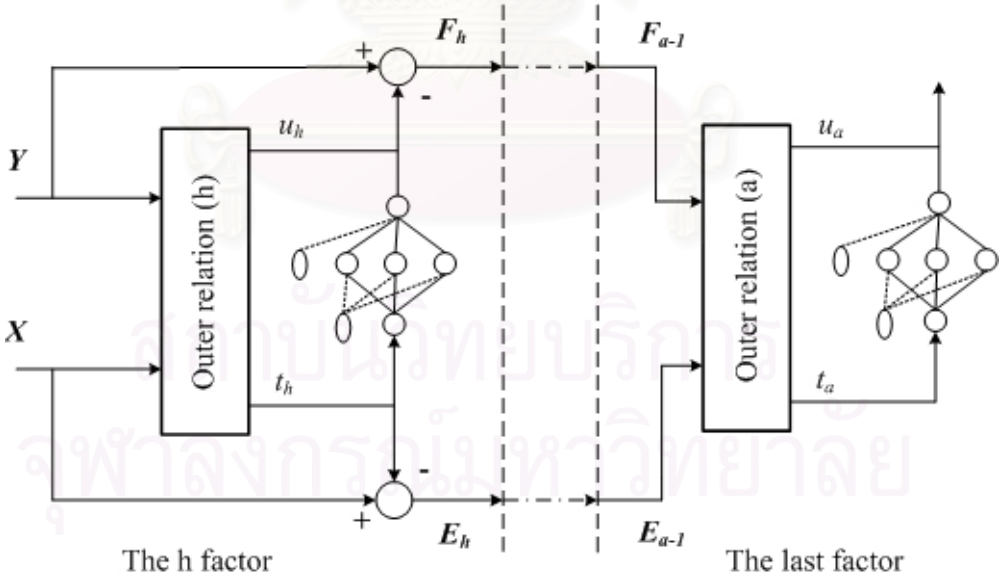

Figure 3.5: A schematic of the NNPLS model: the data are transformed to latent scores, then neural networks are used to learn the scores.

3.3.2 NNPLS identification

The NNPLS algorithm can be constructed based on the NNPLS framework shown in figure 3.5. The NNPLS is composed of linear PLS in outer transform and neural net in inner transform. The algorithm can be proposed as a following:

The algorithm is implemented after the data have been pre-processed; scaling around zero mean and unit variance (Z-score Standardization). Proper scaling prevents the latent variables from being biased towards variables with larger magnitude.

For each factor h

Step 1: Initialization

Set: $u_h = y_j$, $t_h^{old} = x_i$, where i and j are any column of the input matrix X and output matrix Y, respectively, used just for the initialization. These are often taken as the first column.

In the X block:

Step 2:

Define w_h as:

$$w_h' = \frac{u_h' X}{u_h' u_h} \tag{3.21}$$

Step 3:

Normalize w_h to norm of one:

$$w_h = \frac{w_h}{\|w\|} \tag{3.22}$$

Step 4:

Calculate the X matrix scores:

$$t_h = \frac{X w_h}{w_h' w_h} \tag{3.23}$$

In the Y block:

Step 5:

Calculate Y matrix loadings:

$$q_h' = \frac{t_h' Y}{t_h' t_h} \tag{3.24}$$

Step 6:

Normalize he loadings to norm of one:

$$q_h = \frac{q_h}{\|q_h\|} \tag{3.25}$$

Step 7:

Calculate the Y matrix scores:

$$u_h = \frac{Yq_h}{q_h'q_h} \tag{3.26}$$

Step 8:

Check for convergence:

$$\frac{|t_h - t_h^{old}|}{t_h^{old}} \leq \varepsilon \tag{3.27}$$

where ε is a stopping criterion.

If condition Eq. 3.27 is satisfied, continue to step 9 otherwise repeat from step 2 with $t_h^{old} = t_h$

Calculate the X loadings and rescale the scores and weights accordingly:

Step 9:

Calculate the X matrix loadings:

$$p_h' = \frac{t_h'X}{t_h't_h} \tag{3.28}$$

Step 10:

Re-scale the X matrix scores accordingly:

$$t_h = t_h \cdot \|p_h'\| \tag{3.29}$$

Step 11:

Normalize the loadings to norm of one:

$$p_h = \frac{p_h}{\|p_h\|} \tag{3.30}$$

Step 12:

Re-scale w_h accordingly:

$$w_h = w_h \cdot \|p_h\| \tag{3.31}$$

p_h, q_h and w_h are saved for prediction purposes.

Step 13:

Inner model training using neural networks (section 3.3.2.1)

u_h is modeled using the neural network Eq. 3.33. The training is carried out such that $\|\hat{u}_h - u_h\|$ is minimized and the weights are saved for prediction purposes.

Step 14: Calculation of residuals

For each factor h, Eq. 3.19 gives the general outer relation for the X block with E_0 and h is the current factor. For the Y block,

$$F_h = F_{h-1} - \theta(t_h)q'_h \tag{3.32}$$

where $F_0 = Y$ and the inner neural network model has been used in calculating the residual of the output block from Eq. 3.32.

Step 15:

The procedure is repeated recursively with each factor with X and Y replaced by the corresponding residuals E_h and F_h in the algorithm.

3.3.2.1 Inner relationships for NNs

The inner relationships are modeled by the nonlinear model being neural network. This function can write as below equation.

$$\hat{u}_h = \theta(t_h) \tag{3.33}$$

where the function $\theta(.)$ represents the neural network model function. In this work, the network models present output score using the input scores used by multilayer feedforward with Levenberg-Marquandt optimization routine in back-propagation method used to train the network. The details about these method are proposed in section 3.4.8 and 3.4.9 (Artificial Neural Network topic).

3.3.3 Prediction step

This section is important for estimating output variables from input variables. For this purpose, the vectors (p_h, q_h, and w_h) and PLS coefficients from previous section in every PLS factor are used as a following:

Step 1:

Scale input and output sets to zero mean and unit variance by using the scaling parameters used in previous section.

Step 2:

For the X block, t is estimated by multiplying X by w as in the calibration part.

$$t_h = E_{h-1} w_h \tag{3.34}$$

$$E_h = E_{h-1} - t_h p'_h \tag{3.35}$$

For the Y block, $\hat{u}$ is calculated by t and PLS coefficients.

$$Y = F_a = \sum_{h=1}^{a} \hat{u}_h q'_h \tag{3.36}$$

The summation is over h for all the factors (a) one wants to include and $X = E_0$, $Y = F_a$.

3.3.4 Cross-validation

The important point when setting up a PLS model is to make a decision for the optimum number a of principal components involved in the PLS model. PLS the optimum number of components has to be determined empirically by *cross validation* method. This method can be explained that, in over all data, some data are kept out (validation set or test set), the coefficients are calculated from the remaining data (calibration or train set), and finally the y_{Pred} are predicted from the corresponding x-vectors and the model. The squared differences between predicted and actual y-values are added to the **PR**edictive **E**rror **S**um of **S**quares (PRESS). The PRESS value can be used to find the optimum number

of components by a stepwise variable selection procedure. The best model consists of as possible and shows the smallest (or almost the smallest) PRESS value. In the figure 3.6, this show example resulting in the *best* model of five predictor variables. PRESS value can be calculated as:

$$PRESS = \frac{\sum_{i=1}^{m}(y_{i,obs} - y_{i,pred})^2}{m-1} \tag{3.37}$$

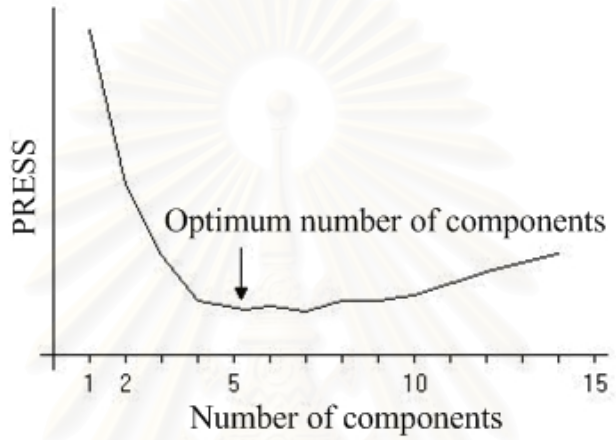

Figure 3.6: Number of components vs PRESS values.

3.4 Artificial Neural Networks fundamentals

An Artificial Neural Networks (ANNs) is information processing paradigm that is inspired by the way of biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs like human brains which learn by many examples. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons.

Table 3.1: The relationship between biological neuron and artificial neuron.

No.	Biological neuron	Artificial neuron
1.	Dendrites	Input variables
2.	Axon	Output variable
3.	Synapse	Weight
4.	Threshold	Bias

3.4.1 Human brain and Artificial Neural Networks

Because the biological neuron is the basic building block of the nervous system, its operation will be briefly discussed for understanding artificial neuron operation and the analogy between ANN and human brain.

In the human brain, a typical neuron collects signals from others through a host of fine structures called dendrites. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. At the end of each branch, a structure called a synapse converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effect that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another will change.

The crude analogy between artificial neuron and biological neuron is that the connections between nodes represent the axons and dendrites, the connection weights represent the synapses, and the threshold approximates the activity in the soma. Figure 3.7 illustrates n biological neurons with various signals of intensity x and synaptic strength w feeding into a neuron with a threshold of b, and the equivalent artificial neurons system. The relationship between biological neuron and artificial neuron is shown in table 3.1.

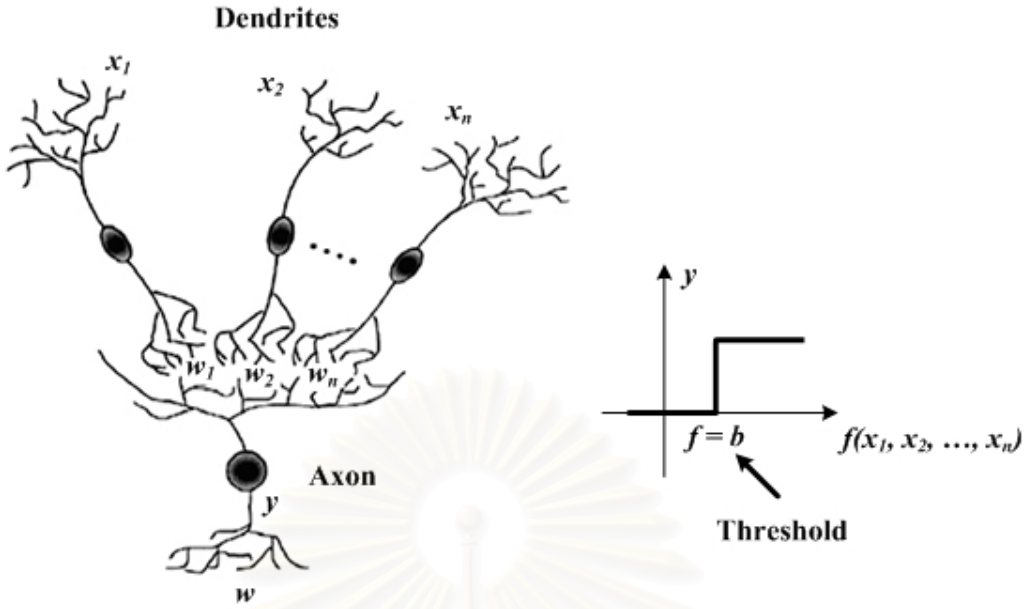

Figure 3.7: Signal interaction from n neurons and threshold signal.

3.4.2 Basic artificial neural networks

Artificial neural networks consist of many interconnected processing elements (artificial neurons or nodes). That a neuron is an information processing unit that roughly resembles its biological counterpart. Figure 3.8 shows a model of an artificial neural. Basic components of neural model are showed as below:

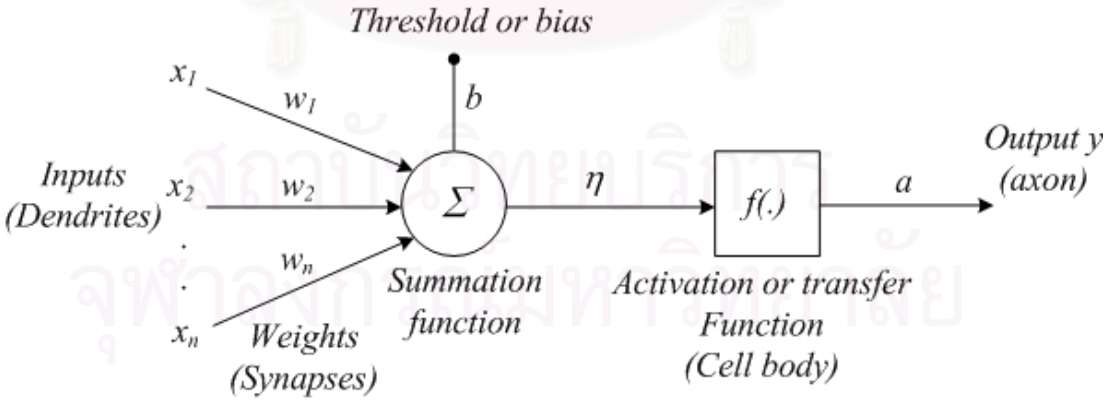

Figure 3.8: Nonlinear model of artificial neuron.

1. There is a set of synapses with associated synaptic or connection weights. As shown in figure 3.8, the continuous-valued input to synapses is a vector signal, with the individual vector components given as x_j for $j = 1, 2, \cdots, n$, Each vector component x_j is input to the synapses and connected to neuron through a synaptic weight w; that is, each of these inputs (X) are multiplied by weight (W) and pulsed by their bias term (b) as shown in below equation.

$$\eta = \sum_{j=1}^{n} w_j x_j + b \tag{3.38}$$

2. Transfer function or activation function, the result of the summation function, almost always the weighted sum, is transformed to a working output through an algorithmic process known as the transfer function. In the transfer function the summation total can be compared with some threshold to determine the neural output. If the sum is greater than the threshold value, the processing element generates a signal. If the sum of the input and weight products is less than the threshold, no signal (or some inhibitory signal) is generated. Both types of response are significant. In this research, the linear and sigmoid transfer functions are utilized.

The linear transfer function is shown in figure 3.9:

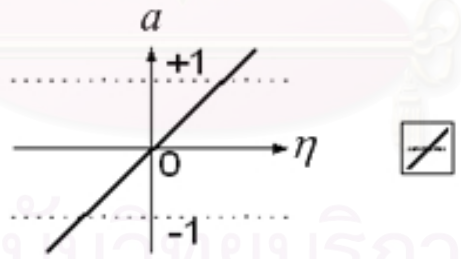

Figure 3.9: Linear transfer function.

The linear transfer functions are utilized in the output layer for outputs expansion purpose, the calculation can be expressed as follows:

$$a = \eta \tag{3.39}$$

According to the expression above, the output values are the same as the corresponding input values but the expansions can be obtained from the adjustable weights (w).

The sigmoid transfer function is shown in figure 3.10:

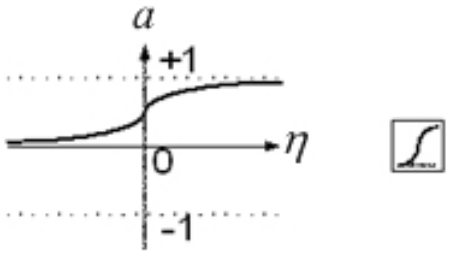

Figure 3.10: Sigmoid transfer function.

The sigmoid transfer functions are utilized in the hidden layers for the nonlinear behavior representation purpose, the calculation can be expressed as follows:

$$a = \frac{1}{1 + e^{-\eta}} \tag{3.40}$$

According to the expression above, the transfer function takes the input and squashes the output into the limited range of 0 to 1; this is the reason why the linear transfer functions are required in the output layer for the outputs expansion purpose.

3.4.3 Architectures of neural networks

Generally, neural networks structure can be divided into 4 types:

1. *Feedforward Connections*: For all the neural models, data from neurons of a lower layer are propagated forward to neurons of an upper layer via feedforward connections networks.

2. *Feedback Connections*: Feedback networks bring data from neurons of an upper layer back to neurons of a lower layer.

3. *Lateral Connections*: n the feature map example, by allowing neurons to interact via the lateral network, a certain topological ordering relationship can be preserved. Another example is the lateral orthogonalization network which forces the network to extract orthogonal components.

4. *Time-delayed Connections*: Delay elements may be incorporated into the connections to yield temporal dynamics models. They are more suitable for temporal pattern recognitions.

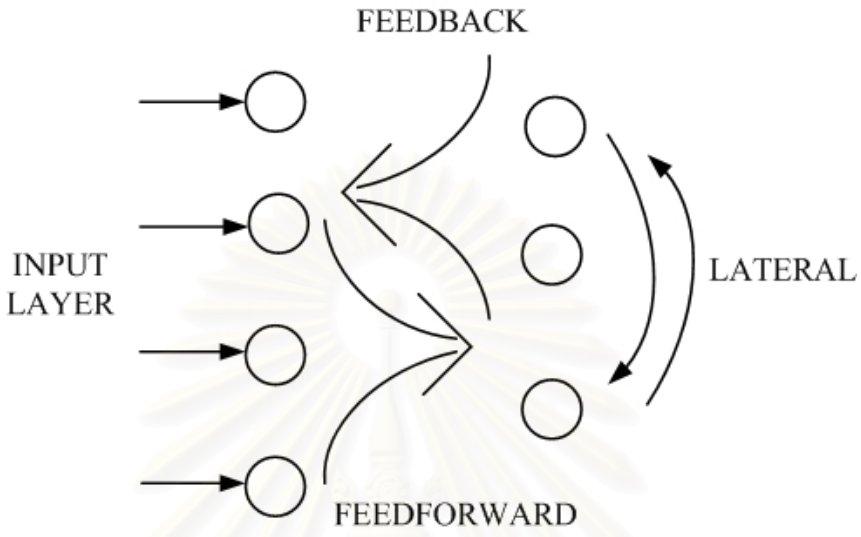

Figure 3.11: Basis structure of the neural network weighted connection.

3.4.4 The Multilayer Feedforward Networks (MLFF)

This architectures is one of the most popular ANN architectures, the multilayer feedforward (MLFF) networks with backpropagation (BP) learning. This type of network is also sometimes called the Multilayer Perceptron because of its similarity to perceptron networks with more than one layer. A general MLFF network is illustrated in figure 3.12 This is a feedforward, fully connected hierarchical network consisting of an input layer, one or more middle or hidden layers and an output layer. The internal layers are called *hidden* because they only receive internal inputs (inputs from other processing units) and produce internal outputs (outputs to other processing units). Consequently, they are *hidden* from the outside world.

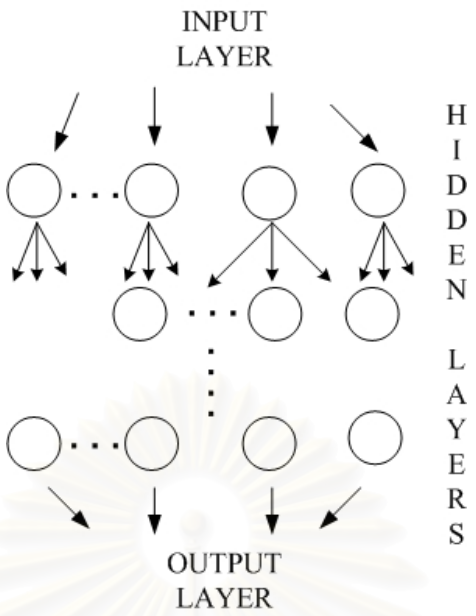

Figure 3.12: A general structure of multilayer feedforward.

3.4.5 Learning Function

The purpose of the learning function is to modify the variable connection weights on the inputs of each processing element according to some neural based algorithm. This process of changing the weights of the input connections to achieve some desired result can also be called the adaptation function, as well as the learning mode. There are two types of learning: supervised and unsupervised. Supervised learning requires a teacher. The teacher may be a training set of data or an observer who grades the performance of the network results. Either way, having a teacher is learning by reinforcement. When there is no external teacher, the system must organize itself by some internal criteria designed into the network. This is learning by doing.

3.4.6 Learning procedures

Learning or training procedures define how exactly the network weights should be adjusted (updated) between successive training cycles (epochs). The error-correction learning (ECL) rule used in this work is used in supervised learning in which the arithmetic difference (error) between the ANN solution at any stage (cycle) during training and the corresponding correct answer is used to modify the connection weights so as to gradually reduce the overall network error. This section will introduce a training procedure called *backpropagation*, which is based on error gradient descent, and then, the several more efficient algorithms including the Levenberg-Marquardt algorithm which is used in this research will be introduced. All of these algorithms require the error between network output and desired output to adjust the weights and biases as shown in Figure 3.13.

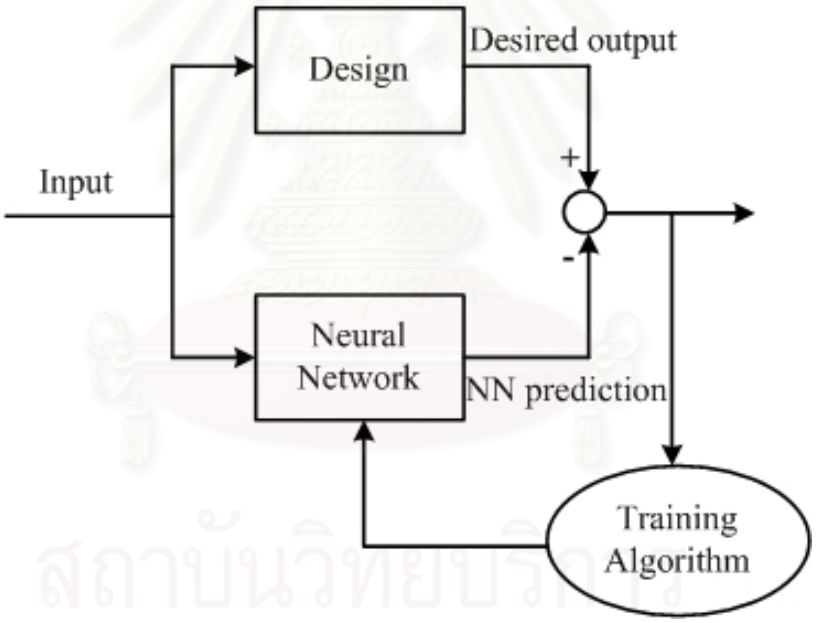

Figure 3.13: The backpropagation method.

3.4.7 Backpropagation neural networks

Backpropagation is the most widely used learning process in neural networks. In this method, the network predicted output is compared with the actual output (target), and the weights are changed in the negative direction of error to minimize the prediction error. This type of learning is known as supervised learning. The algorithm of this network is shown in next section.

3.4.8 The backpropagation procedures

The three-layer network that is case study for description of the backpropagation is shown in figure 3.14. For multilayer networks, the output of one layer becomes the input to the following layer. The equations the describe this operation are

$$a^{m+1} = f^{m+1}(W^{m+1}a^m + b^{m+1}); m = 0, 1, ..., M-1 \tag{3.41}$$

where M is the number of layers in the network. The outputs of the neurons in the first layer receive external input:

$$a^0 = p \tag{3.42}$$

The outputs of the neurons in the last layer are considered the network outputs:

$$a = a^M \tag{3.43}$$

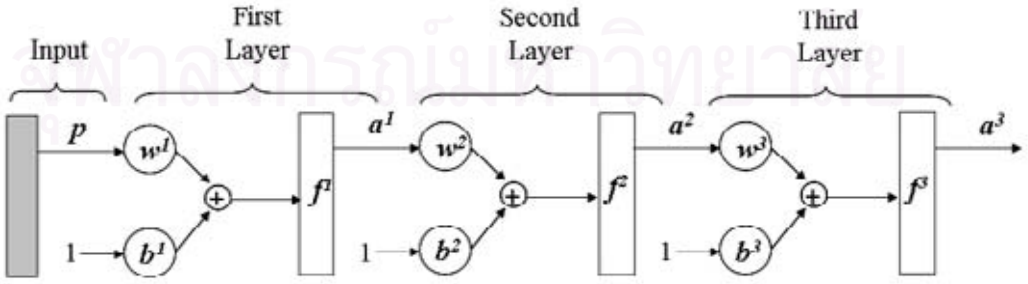

Figure 3.14: Three-Layer Network.

3.8.4.1 Performance Index

The backpropagation algorithm use performance index: mean square error. The algorithm is provided with a set of examples of proper network behavior:

$$(p_1, t_1), (p_2, t_2), \cdots, (p_Q, t_Q) \tag{3.44}$$

where p_q is an input to the network, and t_q is the corresponding target output. As each input is applied to the network, the network output is compared to the target. The algorithm should adjust the network parameters in order to minimize the mean square error:

$$F(x) = E[e^2] = E[(t-a)^2] \tag{3.45}$$

where x is the vector of network weights and biases. If the network has multiple outputs this generalizes to

$$F(x) = E[e'e] = E[(t-a)'(t-a)] \tag{3.46}$$

$$\hat{F}(x) = (t(k)-a(k))'(t(k)-a(k)) = e'(k)e(k) \tag{3.47}$$

where the expectation of the squared error has been replaced by the squared error at iteration k.

The steepest descent algorithm for the approximate mean square error is

$$w^m_{i,j}(k+1) = w^m_{i,j}(k) - \alpha \frac{\partial \hat{F}}{\partial w^m_{i,j}} \tag{3.48}$$

$$b^m_i(k+1) = b^m_i(k) - \alpha \frac{\partial \hat{F}}{\partial b^m_i} \tag{3.49}$$

where α is the learning rate. The $w_{i,j}$ and b_i are defined for the weights from node j in layer $m-1$ to node i in layer m and the bias of node i in layer m, respectively.

For the multilayer network the error is not an explicit function of the weight in the hidden layers, the chain rule of calculus is used to calculate the derivatives. The chain rule is used to find the derivative in Eq. 3.48 and Eq. 3.49:

$$\frac{\partial \hat{F}}{\partial w^m_{i,j}} = \frac{\partial \hat{F}}{\partial n^m_i} \times \frac{\partial n^m_i}{\partial w^m_{i,j}} \tag{3.50}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = \frac{\partial \hat{F}}{\partial n_i^m} \times \frac{\partial n_i^m}{\partial b_i^m} \tag{3.51}$$

The second term in each of these equations can be easily computed, since the net input to layer m is an explicit function of the weights and bias in that layer:

$$n_i^m = \sum_{j=1}^{S^{m-1}} w_{i,j}^m a_j^{m-1} + b_i^m \tag{3.52}$$

Therefore

$$\frac{\partial n_i^m}{\partial w^m, i, j} = a_i^{m-1}, \frac{\partial n_i^m}{\partial b^m, i} = 1 \tag{3.53}$$

Define

$$s_i^m = \frac{\partial \hat{F}}{\partial n_i^m} \tag{3.54}$$

(the sensitivity of $\hat{F}$ to changes in the ith element of the net input at layer m), then Eq. 3.50 and Eq. 3.51 can be simplified to

$$\frac{\partial \hat{F}}{\partial w_{i,j}^m} = s_i^m a_j^{m-1} \tag{3.55}$$

$$\frac{\partial \hat{F}}{\partial b_i^m} = s_i^m \tag{3.56}$$

The approximate steepest descent algorithm is expressed as

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m a_j^{m-1} \tag{3.57}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{3.58}$$

In matrix form this becomes:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha \mathbf{s}^m(\mathbf{a}^{m-1})' \tag{3.59}$$

$$\mathbf{b}_i^m(k+1) = \mathbf{b}_i^m(k) - \alpha \mathbf{s}_i^m \tag{3.60}$$

3.4.8.2 Backpropagating the Sensitivities

It now remains for us to compute the sensitivities s^m, which requires another application of the chain rule. It is this process that gives us the term *backpropagation*, because it describes a recurrence relationship in which the sensitivity at layer m is computed from the sensitivity at layer $m+1$.

$$\mathbf{s}^M = -2\dot{F}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \tag{3.61}$$

$$\mathbf{s}^m = \dot{F}^M(\mathbf{n}^m)(\mathbf{W}^{m+1})'\mathbf{s}^{m+1}; m = M-1, \cdot, 2, 1 \tag{3.62}$$

where

$$\dot{F}^m(\mathbf{n}^m) = \begin{bmatrix} \dot{f}^m(\mathbf{n}_1^m) & 0 & \cdots & 0 \\ 0 & \dot{f}^m(\mathbf{n}_2^m) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \dot{f}^m(\mathbf{n}_3^m) \end{bmatrix} \tag{3.63}$$

The details of calculation about sensitivities were expressed in the book of M. Hagan & H. Demuth (1996)

3.4.8.3 Summary

Let's summarize the backpropagation algorithm. The first step is to propagate the input forward through the network:

$$a^0 = p \tag{3.64}$$

$$a^{m+1} = f^{m+1}(W^{m+1}a^m + b^{m+1}); m = 0, 1, ..., M-1 \tag{3.65}$$

$$a = a^M \tag{3.66}$$

The next step is to propagate the sensitivities backward through the network:

$$\mathbf{s}^M = -2\dot{F}^M(\mathbf{n}^M)(\mathbf{t} - \mathbf{a}) \tag{3.67}$$

$$\mathbf{s}^m = \dot{F}^M(\mathbf{n}^m)(\mathbf{W}^{m+1})'\mathbf{s}^{m+1}; m = M-1, \cdot, 2, 1 \tag{3.68}$$

Finally, the weights and biases are updated by using the approximate steepest descent rule:

$$\mathbf{W}^m(k+1) = \mathbf{W}^m(k) - \alpha\mathbf{s}^m(\mathbf{a}^{m-1})' \tag{3.69}$$

$$\mathbf{b}_i^m(k+1) = \mathbf{b}_i^m(k) - \alpha\mathbf{s}_i^m \tag{3.70}$$

3.4.9 Training Function

3.4.9.1 Conjugate Gradient Method

The basic backpropagation Method adjusts the weights in the steepest descent direction (negative of the gradient). This is the direction in which the performance function is decreasing most rapidly. It turns out that, although the function decreases most rapidly along the negative of the gradient, this does not necessarily produce the fastest convergence. In the conjugate gradient algorithms a search is performed along the conjugate directions, which produces generally faster convergence than steepest descent directions.

All of the conjugate gradient algorithms start out by searching in the steepest descent direction (negative of the gradient) on the first iteration.

$$\mathbf{s}^0 = -\nabla f(\mathbf{x}^0) \tag{3.71}$$

A line search is then performed to determine the optimal distance along the current search direction:

$$\mathbf{x}^1 = \mathbf{x}^0 + \alpha^0 \mathbf{s}^0 \tag{3.72}$$

Then the next search direction is determined so that it is conjugate to various search directions. The general procedure for determining the new search direction is to combine the new steepest descent direction with the previous search directions:

$$\mathbf{s}^1 = -\nabla f(\mathbf{x}^1) + \mathbf{s}^0 \frac{\nabla' f(\mathbf{x}^1)\nabla f(\mathbf{x}^1)}{\nabla' f(\mathbf{x}^0)\nabla f(\mathbf{x}^0)} \tag{3.73}$$

For the kth iteration the relation is

$$\mathbf{s}^{k+1} = -\nabla f(\mathbf{x})^{k+1} + \mathbf{s}^k \frac{\nabla' f(\mathbf{x}^{k+1})\nabla f(\mathbf{x}^{k+1})}{\nabla' f(\mathbf{x}^k)\nabla f(\mathbf{x}^k)} \tag{3.74}$$

3.4.9.2 Newton's Method

Newton's method is an alternative to the conjugate gradient methods for fast optimization. The basic step of Newton's method is

$$\mathbf{x}^{k+1} - \mathbf{x}^k = -[\mathbf{H}(\mathbf{x}^k)]^{-1}\nabla f(\mathbf{x}^k) \tag{3.75}$$

where $\mathbf{H}(\mathbf{x}^k)$, is the Hessian matrix (second derivatives) of the performance index at the current values of the weights and biases. Newton's method often converge faster than conjugate gradient methods. Unfortunately, it is complex and expensive to compute the Hessian matrix for feedforward neural networks.

3.4.9.3 Levenberge-Marquardt

The Levenberge-Marquardt algorithm was designed to approach second order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feedforward networks), then the Hessian matrix can be approximated as

$$\mathbf{H} = \mathbf{J}'\mathbf{J} \tag{3.76}$$

and the gradient can be computed as

$$\nabla f = \mathbf{J}'\mathbf{e} \tag{3.77}$$

where $\mathbf{J}$ is the Jacobian matrix, which contain first derivatives of the network errors with respect to the weights and biases, and e is a vector of network errors. The Jacobian matrix can be computed through a standard backpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton like update:

$$\mathbf{x}^{k+1} = \mathbf{x}^k - [\mathbf{J}'\mathbf{J} + \mu I]^{-1}\mathbf{J}'\mathbf{e} \tag{3.78}$$

When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When is large, this becomes gradient descent with a small

step size. Newton 's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton 's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

The more details of these training algorithms can find in Edgar T. F. and Himmelblau D. M. 2001.

3.4.10 Underfitting and Overfitting

One of the problems occurred when training the neural network is called overfitting. The error on the training set is driven to a very small value, but when the new data set is presented to the network, the error is large. The network has memorized the training examples, but it has not learned to generalize to the new situations.

The training algorithms such as Backpropagation, Backpropagation with Adaptive Learning Rate and Backpropagation with Levenberg-Marquardt Approximation are sensitive to the number of neurons in hidden layers. While, in general, the more neurons in hidden layers, the better data the network can fit, if far too many neurons are used, overfitting can occur. And as general, if too few neurons are examined, underfitting can occur. These problems have been demonstrated in figure 3.15.

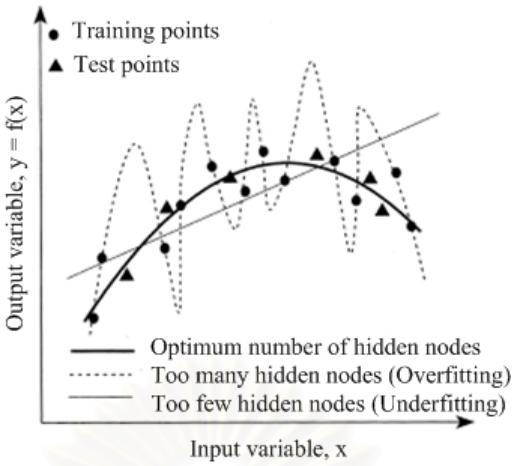

Figure 3.15: Effect of hidden nodes on network generalization.

3.4.11 Neural Networks Performance Improvement

A method used in this research for the network improvement is call *early stopping*. In this technique, there are three sets of data used for building models. The first data set is the training set, used for initially computing the gradient based on the training data set and updating the network weights and biases which are randomly initialized. After the initial training, the trained weights and biases are obtained. If the training is successful, these weights and biases would make the network fit well with the training data set and these become the initial weights and biases for the validation. The trick is in the validation, during the normal training based on the validation data set, the performance of network on the training data set is monitored. During the validation, the error based on the validation data set will come down from the beginning through the end. When the overfitting begins, the error based on the training data set will begin to rise and this is an additional criterion for stopping the validation, this is the reason why it is called *early stopping*. This technique can improve the network by generalizing for two different data sets, not too fit for the only one set which makes the performance go worse for the generalization.

3.4.12 Criteria for choosing the number of hidden nodes

In this work, the criteria to find suitable a number of hidden nodes have been explained as later sentences. The most commonly used stopping criterion in neural network training is the mean of squared errors (MSE) calculated for the training or test subsets. Generally, the error on training data decreases indefinitely with increasing number of hidden nodes or training cycles, as shown in Figure 3.16. The initial large drop in error is due to learning, but the subsequent slow reduction in error may be attributed to (i) network memorization resulting from the excessively large number of training cycles used, and/or (ii) overfitting due to the use of a large number of hidden nodes. During ANN training, the error on test subsets is monitored which generally shows an initial reduction and a subsequent increase due to memorization and overtraining of the trained ANN. The final (optimal) neural network architecture is obtained at the onset of the increase in test data error.

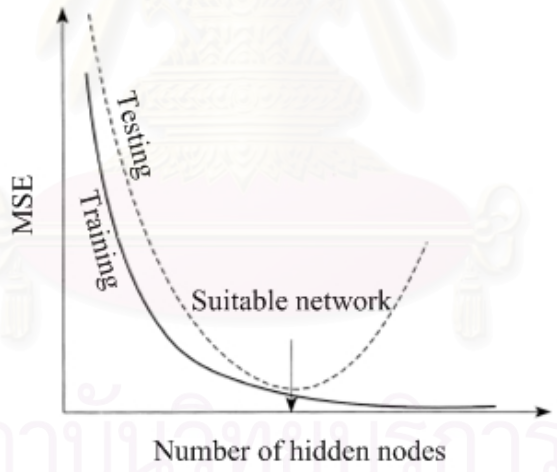

Figure 3.16: Criteria for termination of training and selection of suitable network architecture.

More details about the artificial neural network can be found in Lippman, R.P. 1987; Patterson, D.W. 1996; Hagan M., Demuth H. and Beale M. 1996; Basheer, I.A. and Hajmeer, M. 2000.

CHAPTER IV

SOFT SENSORS FOR CASE STUDY

This chapter presented large scale about the cyclohexanone unit, the involved procedures of this work, and simulation results of soft sensors for cyclohexanone unit.

4.1 The description of the plant

Thai Caprolactam Public Co., Ltd. (TCL), in Rayong (Thailand) is the first producer and distributor of caprolactam and ammonium sulphate in Thailand and Southeast Asian region. Caprolactam is an important raw material in the production of Nylon 6 which is used in various industries, given its special characteristics. It is very durable and can withstand high temperature well, yet still maintains excellent flexibility. Moreover, Caprolactam ($C_6H_{11}NO$) is created by a complex polymerisation process using three major raw materials; sulfur, ammonia and cyclohexane. In and intermediate step sulfur is modified to sulfuric acid liquid, ammonia is processed into hydroxylamine and cyclohexane is altered into cylcohexanone. These three chemicals combine to create two products, the first being caprolactam. The second, ammonium sulfate, is a co-product representing three quarters of the output of polymerisation and is one of the key fertilisers used in Thailand. The Company's production capacity for caprolactam and ammonium sulfate is 70,000 metric tonnes and 280,000 metric tonnes, respectively. Reader can find the more informations about the Thai Caprolactam Public Co., Ltd. (TCL) at http://www.caprolactam.net.

4.2 The work description

This plant comprises many parts of chemical processes. All of the parts are important to efficient production of caprolactam. In this work, soft sensors

have been built for cyclohexanone/cyclohexanol distillation section because this section is highly significant for producing the caprolactam. The details of the section has been shown in next topic.

4.2.1 Cyclohexanone/Cyclohexanol Distillation Section

In this section, it is required to separate high purity cyclohexanone and clycohexanol from the mixture. The cyclohexanone is one of the reactant used to produce caprolactam. The process flowchart of this section has been demonstrated in figure 4.1.

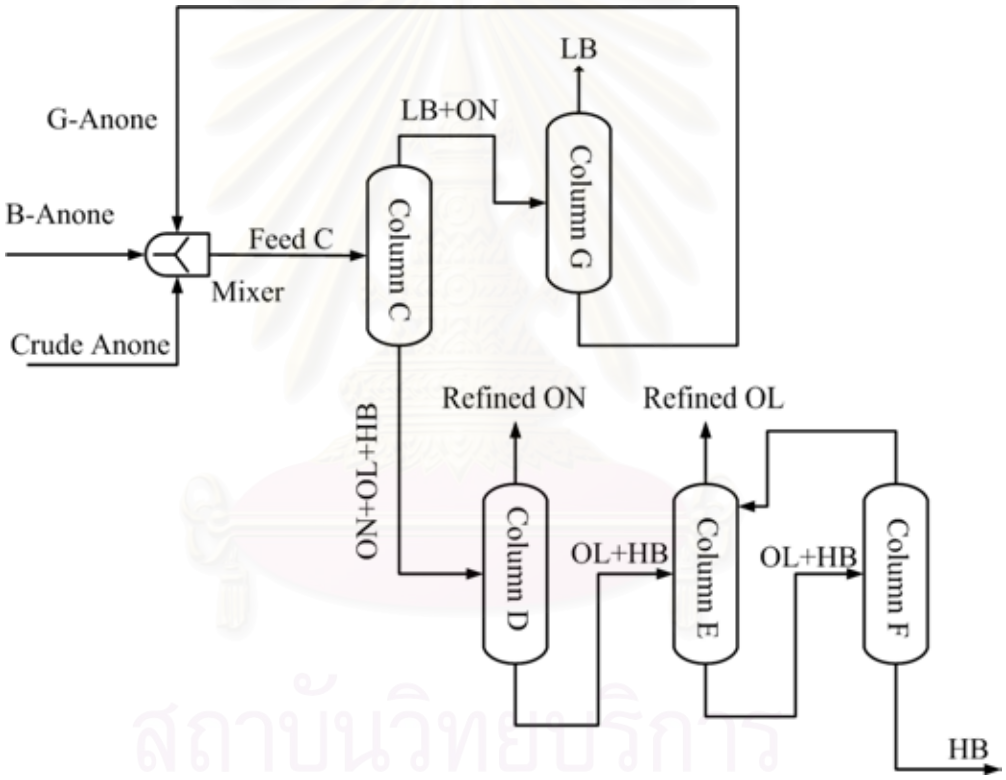

Figure 4.1: Schematic representation of cyclohexanone/cyclohexanol section.

From figure 4.1, this section has five distillation columns. Moreover, these labels of each stream roughly represent the chemical component of this section shown as following. First, LB represented low boiling point components that come from other section in the plant. Second, ON represented cyclohexanone. Third, OL represented cyclohexanol. Forth, HB also represented high boiling point components. The rank of boiling point for each component in the mixture

is presented below.

$$\text{LB} < \text{ON (cyclohexanone)} < \text{OL (cyclohexanol)} < \text{HB}$$

Other relevant components and component details of LB and HB group are described in section 4.4. For more understanding, work approximate information of each distillation is required as below sentences.

The column C is used to separate LB components from B-Anone, crude Anone and G-Anone (recycle from column G). Most of the top products are ON + LB and bottom products are ON + LO + HB. Distillate is fed to the column G for separating ON, and the bottoms are moved to the column D.

The column D is used for separating ON which come from bottom stream of the column C. Top product (pure ON) is fed to refined ON for producing caprolactam in latter stage while bottom stream is OL + HB fed to the column E.

The column E is used to separate OL from bottom stream of column D. Top product (refined OL) is OL and ON being remaining ON from the column D. In another stream, bottom stream consisting of OL + HB is fed to the column F.

The column F is used to separate HB from OL + HB stream. Top product is OL + HB (small fraction) and bottom product composed of mainly of HB + OL is moved to another unit. The top stream is fed to condenser to condense into liquid phase being reflux at bottom of the column F.

The column G is used for separating LB from ON that move from the top stream of the column C. The distillate stream is pure LB for using in latter section while the bottom stream called G-Anone is mainly ON for back feeding to separate in the column C.

4.2.2 The position of the soft sensor

From previous sentences, the column that are interested for building soft sensors is the column D since the top product of this column is mainly cyclohexanone used as primary component for producing caprolactam (as mentioned

previously). The purity of cyclohexanone (ON) is meaningful for the final production so soft sensors for estimating quality of top stream of the column D have been applied for this situation. The key variable estimated by soft sensors is concentration of cyclohexanol (OL) in the top output stream of the column D because concentration of cyclohexanol (OL) combining with cyclohexanone (ON) can represent the impurity of product in distillate stream. In order to build soft sensors, it is necessary to use the available hardware sensors and laboratory analysis for estimating the concentration of cyclohexanol in this column. The sensors installed on the column D to monitor the distillation measured their process variables every one minute. In another data source, laboratory technicians tap the sample of distillate stream of the column D for estimating OL every eight hours. The column D with its hardware sensors have been illustrated in figure. 4.2.

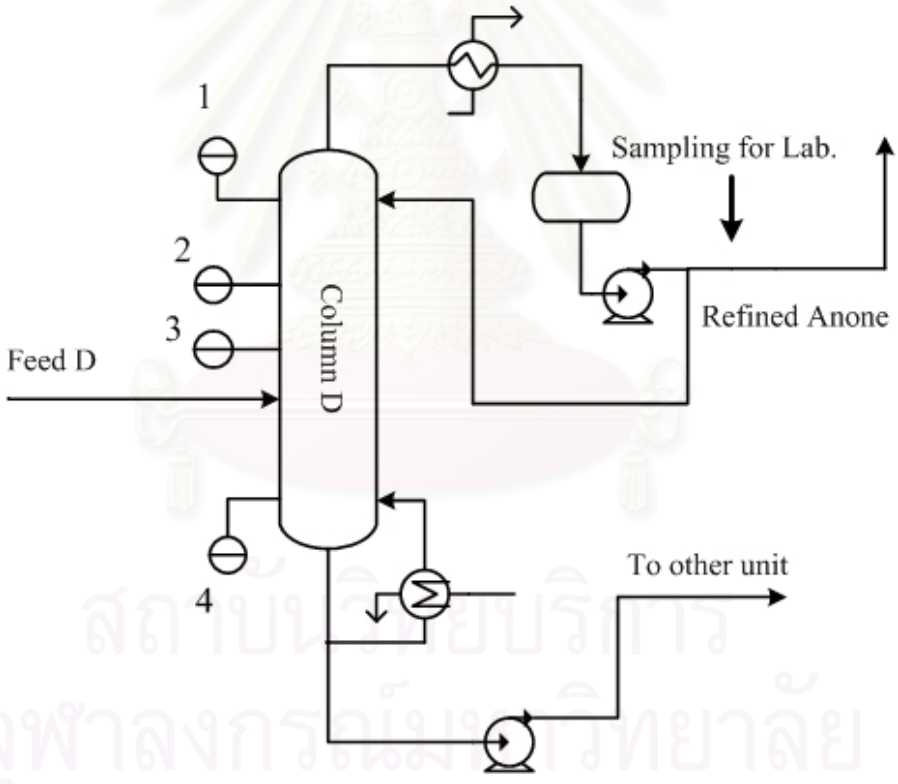

Figure 4.2: The locations of hardware sensors in the column D.

From figure 4.2, These sensors are four input variables for building soft sensors to predict output (OL concentration). The set of sensors relevant to construct soft sensors are listed in table 4.1, together with the corresponding description and observations ranges.

Table 4.1: The used hardware sensors for building soft sensors

Tag	Description	Range	Units
1.	Top Temperature of Column D	57.40 - 57.87	^oC
2.	High Middle Temperature of Column D	66.26 - 67.85	^oC
3.	Low Middle Temperature of Column D	77.42 - 79.64	^oC
4.	Bottom Temperature of Column D	91.66 - 93.62	^oC

4.2.3 Short coming of using pure real plant data

In real plant operations, generally, the data collected from historical database have normally smooth response or small ranges of operation. Therefore, if we used only real plant data for constructing soft sensors, it is limitation to implement in wide ranges operation of processes. When the operating point shifted or external conditions changed from original condition, the model soft sensor cannot accurately implement for estimating performance. For extending the operation ranges, it was necessary to find another source of data that can represent wide range operations of system. By this reason, this work had to construct the rigorous model by fitting with real plant operation and simulated other possible conditions occurring in real plant for generating different data sets. For the rigorous model, the steady state of the multicomponent distillation model of the column D was defined. this work used the commercial simulator-HYSYS program to build the model for generating wide range data. Finally, the simulated data were used to combine with plant data in order to build soft sensors model. The steps for generating the data were demonstrated in section 4.4.1.

4.3 Building an empirical soft sensor

In model building, the inputs and output of soft sensor were listed as follows:

1. x_1 is top temperature of Column D,
2. x_2 is high middle temperature of Column D,
3. x_3 is low middle temperature of Column D,
4. x_4 is bottom temperature of Column D, and
5. y is concentration of cyclohexanol.

As a design approach, the model in the form:

$$y(k) = f(x_1(k), x_2(k), x_3(k), x_4(k)) \tag{4.1}$$

where $y(k)$ is the output at time k and x_j with $j = 1, 2, \ldots, 4$ are input variables. The unknown function $f(\ldots)$ are implemented by MLFF, linear PLS, and NNPLS (the theory introduced previously in CHAPTER III).

4.3.1 Model building

General procedures for building soft sensors can be mainly divided into two steps in both methods.

Step 1 (model determination): In this work, the data were normalized by Z-score Standardization or Min-Max Normalization in preprocessing step. For later step, the model structure, the most important procedure, is determined based on the calibration or training data set.

Step 2 (model testing): After determining a model, its accuracy is usually validated using a new test set.

Since soft sensors were built from two approaches, it was necessary to has the performance criteria for comparing these methods. The performances of soft sensors were calculated by indexes in section 4.3.2.

4.3.2 The performance index of model predictions

In order to demonstrate the efficiency of the model, the estimating performance of the soft sensors being the prediction accuracy and capability to track the process trend is evaluated. The criterion is used in terms the root mean squared error of prediction ($RMSE$), which is calculated as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(y_{i,Obs} - y_{i,Pred})^2}{n}} \tag{4.2}$$

where n is number of data test set, y_{Obs} is output from observation, and y_{Pred} is the corresponding estimate from the soft sensors.

4.4 Work processing

Since we interested top stream of the column D, we need to consider the other related unit operations consisting of the column C ,and D. The components in cyclohexanone/cyclohexanol section were demonstrated as the following:

- Cyclohexanone (ON),
- Cyclohexanol (OL),
- Low boiling point components (LB),
  - Cyclohexene
  - 2-pentanone
  - n-pentanal
  - 1-butanol
  - Mesityloxide
- High boiling point components (HB),
  - 2-cyclohexylcyclohexanone
- A component (n-pentylcyclohexane),

- B component (cyclohexylbutylether),
- Water.

For these components, their formulas, NB (normal boiling point), and MW (molecular weight) have been shown in table 4.2.

Table 4.2: The list of components in case study

Component	**Formula**	**NB(^{o}C)**	**MW.**
Cyclohexanone (ON)	$C_6H_{10}O$	155.65	98.14
Cyclohexanol (OL)	$C_6H_{12}O$	161.15	100.16
Cyclohexene (1st LB)	C_6H_{10}	82.95	82.14
2-pentanone (2nd LB)	$C_5H_{10}O$	102.00	86.13
n-pentanal (3rd LB)	$C_5H_{10}O$	102.85	86.13
1-butanol (4th LB)	C_4H_6O	117.75	120.85
Mesityloxide (5th LB)	$C_6H_{10}O$	129.80	98.14
2-cyclohexylcyclohexanone (HB) (2CyHxCC6one)	$C_{12}H_{20}O$	264.00	180.29
n-pentylcyclohexane (A component)	$C_{11}H_{22}$	203.75	154.30
cyclohexylbutylether (B component)	$C_{10}H_{20}O$	175.63	156.27
Water	H_2O	100.00	18.02

The components in table 4.2 were separated by the column C and columns D. The flows of each component in both distillations were performed in figure 4.3. Furthermore, the details of condition in the column D for using in steady state rigorous model of HYSYS simulator were shown in figure 4.4.

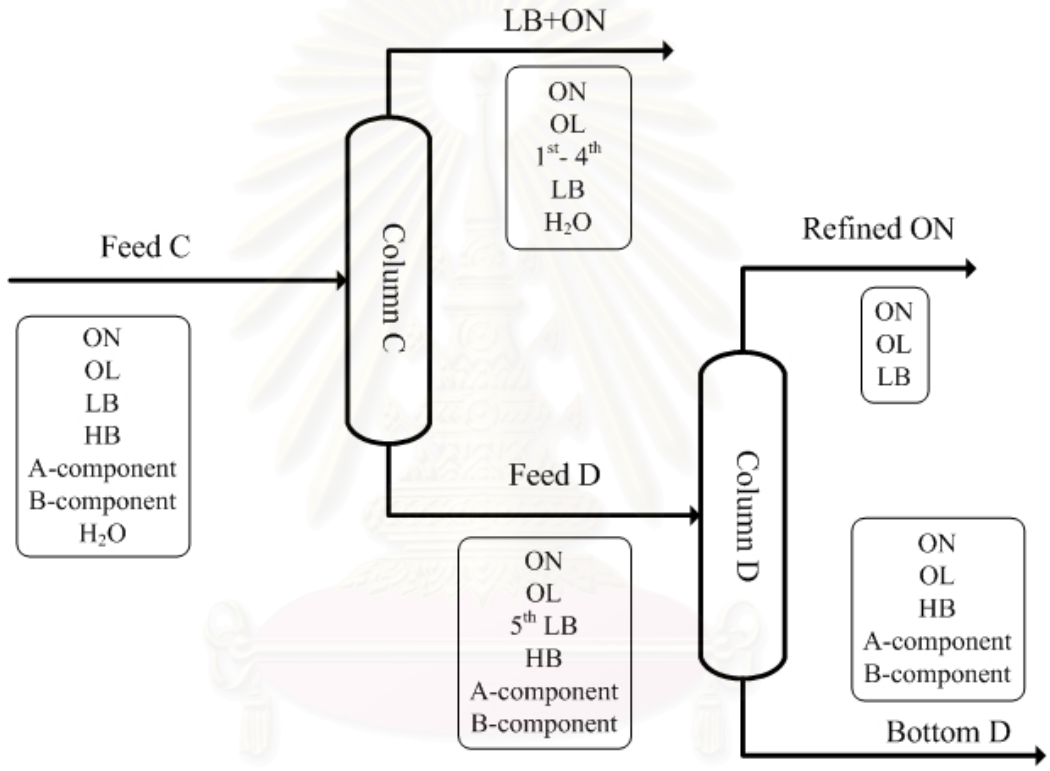

Figure 4.3: The column C and column D with each component stream in real operation.

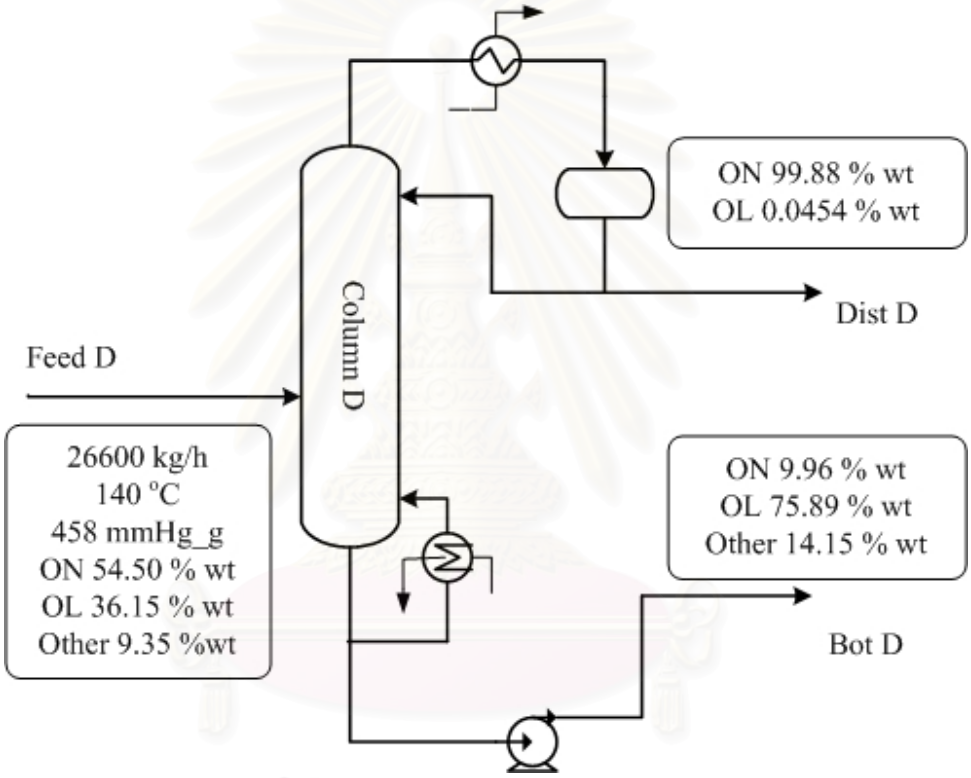

Figure 4.4: The column D with streams condition.

4.4.1 Simulations for the distillation

We simulated stead state model of the column D by the HYSYS simulator with its assumptions. Table 4.3 compared the base case simulation results and the actual plant data. From this table, it could be seen that the simulation results fitted very well against the original plant data collected during the plant operation.

The following assumptions were used in this work:

- feed D was 26,600 kg/hr,
- pressure of feed D was 458 mmHg-g,
- temperature of feed D was 140 ^{o}C,
- test run plant data was the base case, and
- current operating at that time was constraints.

Table 4.3: Base case simulation results

Description	**Unit**	**Simulation**	**Actual**	**% Diff**
OL mass fraction in feed D	-	0.361	0.362	-0.013
ON mass fraction in feed D	-	0.545	0.545	-0.013
OL mass fraction in Dist D	-	4.539×10^{-4}	4.540×10^{-4}	-0.013
ON mass fraction in Dist D	-	0.999	0.999	-0.003
Top Temperature of Column D	^{o}C	57.378	57.573	-0.338
High Middle Temperature of Column D	^{o}C	66.670	66.857	-0.280
Low Middle Temperature of Column D	^{o}C	78.417	78.608	-0.243
Bottom Temperature of Column D	^{o}C	92.466	92.832	-0.395
Top Pressure of Column D	mmHg	19.000	18.986	0.074

Remark: each actual value in table 4.3 was its average value of plant data that was observed in this work.

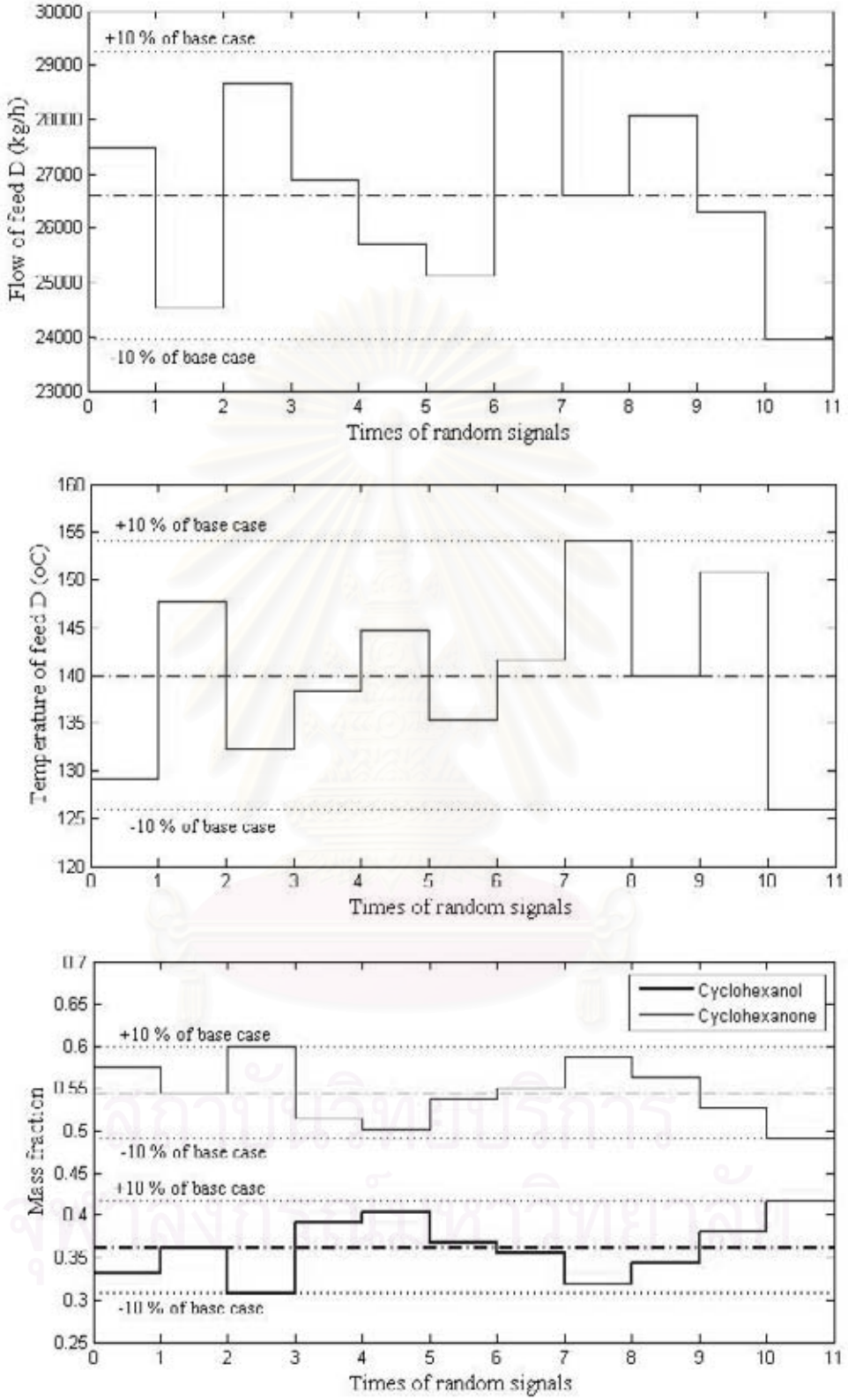

Figure 4.5: Random step changes of three inputs for generating simulated data : each middle dash line represented its base case value.

For generating other conditions, we varied mass flow rate of feed D, feed D temperature and major component (ON and OL) in feed D of base case steady state model. When simulated data was generated, two distillation parameter being reflux ratio and duty of reboiler were fixed at constant or these parameter values were equal to base case condition. The varied variables were be changed from base case to ± 10 % of base case condition. These step changes have been shown in figure 4.5.

Finally, in soft sensor models building, we obtained all of the data for preparing to use in section 4.5. The number of data could be performed in table 4.4.

Table 4.4: The number of data

Kind of data	**Observations**
Plant data	451
Simulated data	11×11×11 = 1331
Total data	1782

4.5 Soft sensor models

In this section, empirical soft sensors for estimating OL were generated by using plant data and simulation data. Two methods have been utilized for building the soft sensors. There were artificial neural networks and partial least squares regression. Both methods used the same data set for calibration (train), and prediction (test) step. We selected randomly data as two sets being calibration sets and prediction sets. The prediction sets consisted of 10 % of plant data, and 10 % of simulated data, approximately and remaining data were used in the calibration sets. In summary, the number of data that were classified have been shown in table 4.5.

Table 4.5: The observations of two subsets

Data set	Observations
Calibration (training) sets	406 (Plant data) + 1197 (Simulated data) = 1603
Prediction (test) sets	45 (Plant data) + 134 (Simulated data) = 179

4.5.1 The constructed soft sensor by using only pure plant data

This issue showed the performance of the soft sensor that was built from only real plant data for the calibration step when other operating condition generated by the HYSYS simulator and partial real plant data were used as prediction sets (as provided in table 4.5) in oder to test the performance of this soft sensor.

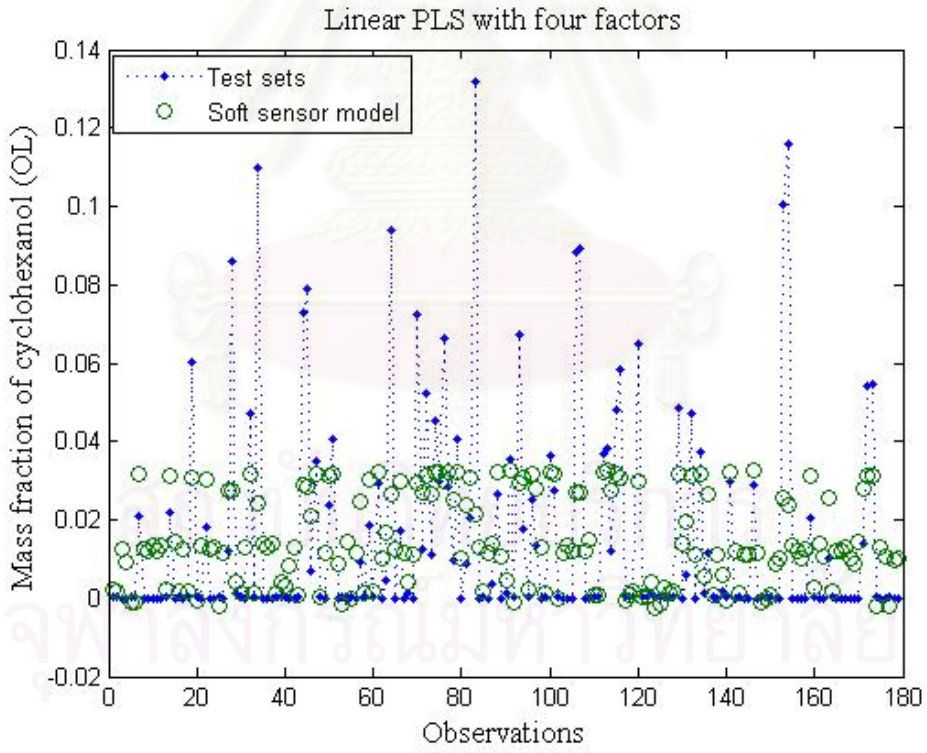

Figure 4.6: Prediction of soft sensor based on linear PLS using only real plant data for model calibration.

The method for building this soft sensor was linear partial least squares and procedures for the building were similar to section 4.5.3. Form figure 4.6, the result illustrated that the soft sensor using only real plant data for the calibration step gave failure to predict mass fraction of cyclohexanol (OL) when operating condition changed or shifted from available normal condition of real plant.

To handle this problem, we could improve the performance of soft sensors by conducting wide range simulated data to the calibration step for extending working ranges of soft sensors. This purpose were utilized for building soft sensors based on NNs, linear PLS, and NNPLS. Consequently, Three soft sensors have been proposed in below section.

4.5.2 Pre-construct neural network soft sensors

A soft sensor model was created using feedforward (MLFF) neural network. Neural network is a universal function approximator that is useful method to find relationships between variables when the data of these variables are available. A network takes in a set of input data, sums them together, takes some function of them, and passes the output through a weighted connection to another neuron. The neuron is thus just a estimated variable, or a function of a nonlinear combination of estimated variables. The connection weights serve as adjustable parameters which are set by a training method. In this work, transfer functions of hidden layer(s) were sigmoid functions and the transfer function of output layer was linear function. The neural models were trained with the Levenberg-Marquardt Backpropagation algorithm by using the training set. The number of hidden neurons was chosen by trial and error with minimizing the Root Mean Squared Error (RMSE) between the model prediction and the test data sets.

The purpose of pre-construct neural network soft sensor is to construct the backbone or skeleton of neural network soft sensor by using the wide range simulated data and we used the real plant data to refine the structure again. The wide range simulated data make the neural soft sensor more robustness or greater robustness. In summary, we used wide range simulated data by generating data in table 4.5 for finding suitable neural structures to support all conditions that

could possibly exist or finding initial weights and bias of each connection. The next step, we trained these structures again by using data plant in order to refine the soft sensor models. This step was to ensure agreement between simulated data analysis laboratory analysis results.

In finding suitable model, the available data in calibration sets was divided into two subsets. The first subset was the training set which was used for computing the gradient and updating the network initial parameter values of weights and biases of neural structures. The second subset was was the validation set. The error on the validation set was monitored during the training process. The validation error will normally decrease during the initial phase of training, as does the training set error. However, when the network begins to overfitting the data, the error on the validation set will typically begin to rise. When the validation error increases for a specified number of iterations, the training is stopped, and the weights and biases at the minimum of the validation error are returned. This algorithm is called *early stopping method*. The test sets were prediction sets. They were used to test performance of models by RMES value.

The procedures for obtaining pre-construct soft sensor based on neural network models were summarized and listed as a following.

Calibration step

- Both input data and output data were normalize by Min-Max normalization (as proposed in APPENDIX B).
- Find suitable structure of network by increasing number of hidden nodes and number of hidden layers.
- Each of the structures in previous step was trained by *early stopping method*.
- Selected structures were pretrained for finding initial neural network parameter (weights and biases) by using simulated data sets of calibration sets (as proposed in table 4.5).
- The parameter of these structures were refined or retrained again by using plant data sets of calibration sets (as proposed in table 4.5).

Prediction step

- Normalize test sets (both inputs and outputs) using the scaling parameters used during the model training.
- All generated structures were tested by these test sets.

The RMSE values of test sets (original data) for various neural network models were demonstrated in table 4.6. The minimum RMSE index was regarded for choosing neural network structures. It indicated that the NN soft sensor which consisted of five nodes for first hidden layer and three nodes for second hidden layer of was suitable structure (4-5-3-1) because this structure gave the least RMSE of other structures varied and selected structure was not quite complex structure that resulted in small computing time for implementing in the real problem. Graphically, the selected neural network structure was shown in figure 4.7.

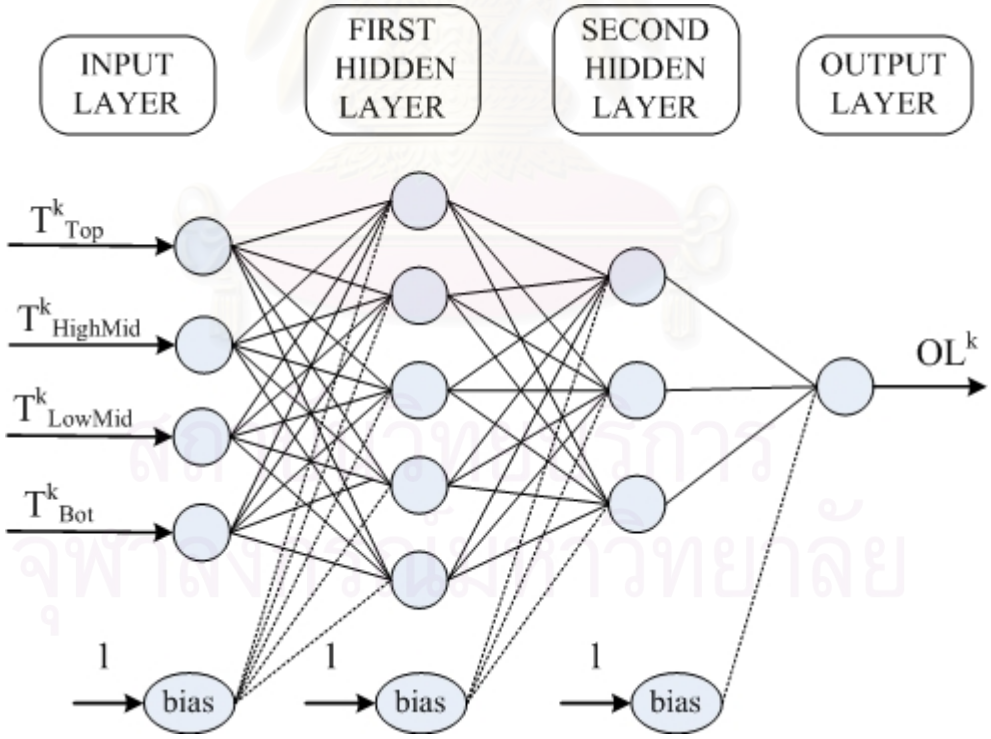

Figure 4.7: The selected structure (4-5-3-1).

Table 4.6: RMSE of test sets for various structures

Layers		**RMSE** of test sets
1st **layer**	**2**nd **layer**	$\times$**10**$^{-5}$
3	0	4.31096
	3	2.66435
	5	1.74723
	7	1.68899
	9	1.85622
5	0	2.02520
	3	1.10988
	5	1.80394
	7	1.67053
	9	1.78454
7	0	1.99014
	3	1.88068
	5	1.92180
	7	1.62779
	9	1.83200
9	0	1.99066
	3	1.27807
	5	1.53762
	7	1.92659
	9	1.79166

4.5.3 Modified partial least squares soft sensors

Partial least squares (PLS) is a linear system identification method that projects the input/output data down into a latent space, extracting a number of principal factors with an orthogonal structure, while capturing most of the variance in the original data. When the process exhibits non-linear behavior, it is desirable to extend the PLS model structure to capture non-linearities. In this work, NNPLS, neural network partial least squares (nonlinear PLS), were used to build soft sensors. In summary, we utilized linear PLS and NNPLS for constructing soft sensors by using the same data of calibration set and prediction sets as soft sensor based on ANNs.

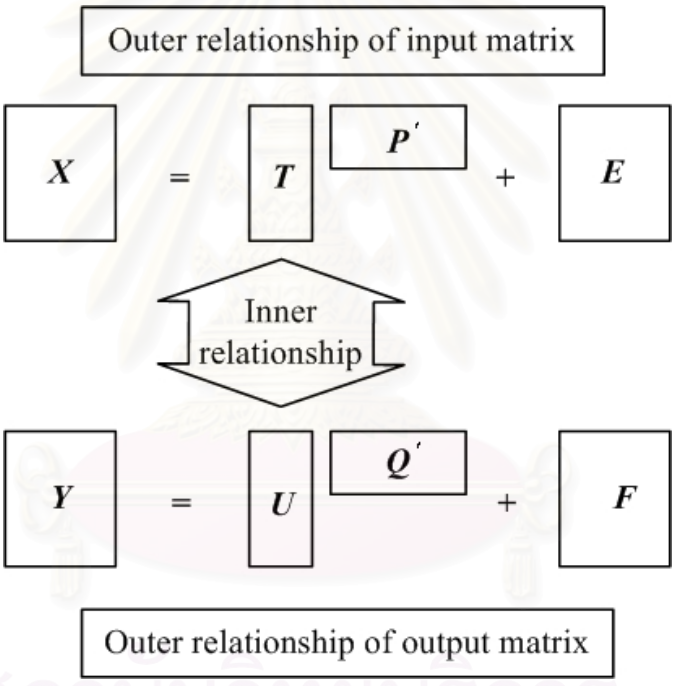

Figure 4.8: PLS structure.

Form figure 4.8, The procedures for obtaining soft sensor based on partial least squares were summarized and listed as a following.

Calibration step

- Both input data and output data were normalize by Z-score Standardization (zero-mean and unit-variance) as proposed in APPENDIX B.
- Find outer relationship of PLS (as provided in CHAPTER III).

- Find both inner relationship of linear PLS and NNPLS.
- For linear PLS, the inner functions of PLS model were found by using mixed data between simulated data and plant data of calibration sets in table 4.5 or these functions were defined at one time.
- For NNPLS, the inner functions of NNPLS model were found by using the similar way of calibration step in section 4.5.2 or, in another word, these functions were defined by two steps. First step, the parameters of inner function were pretrained by simulated data sets. Second step, these parameters were refined again by plant data sets.
- Find number of factors (components) used in PLS model by PRESS value.
- P, Q, and W are saved for prediction purposes.

Prediction step

- Normalize test sets (both inputs and outputs) to zero mean and unit variance using the scaling parameters used during the model calibration.
- PLS structure from calibration step was tested by these test sets.

For Linear PLS, Since input variables had four inputs, four factors were extracted by PLS decomposition. The linear functions of inner relationships between t and u of each factor were calculated and graphical representation of inner relation of only random 100 data points have been shown in figure 4.9.

For NNPLS, NNPLS could also decompose as four factors. The inner relationships were estimated by MLFF neural network and inner neural structures were selected from the structures showing small RMSE values between u and $\hat{u}$ (estimated by neural models). RMSE values of four factors have been demonstrated in table 4.7. The inner neural models of each factor was consisted of three layers (input layer, one hidden layer, and output layer). Furthermore, transfer functions of hidden layer were sigmoid functions and the transfer function of output layer was linear function.

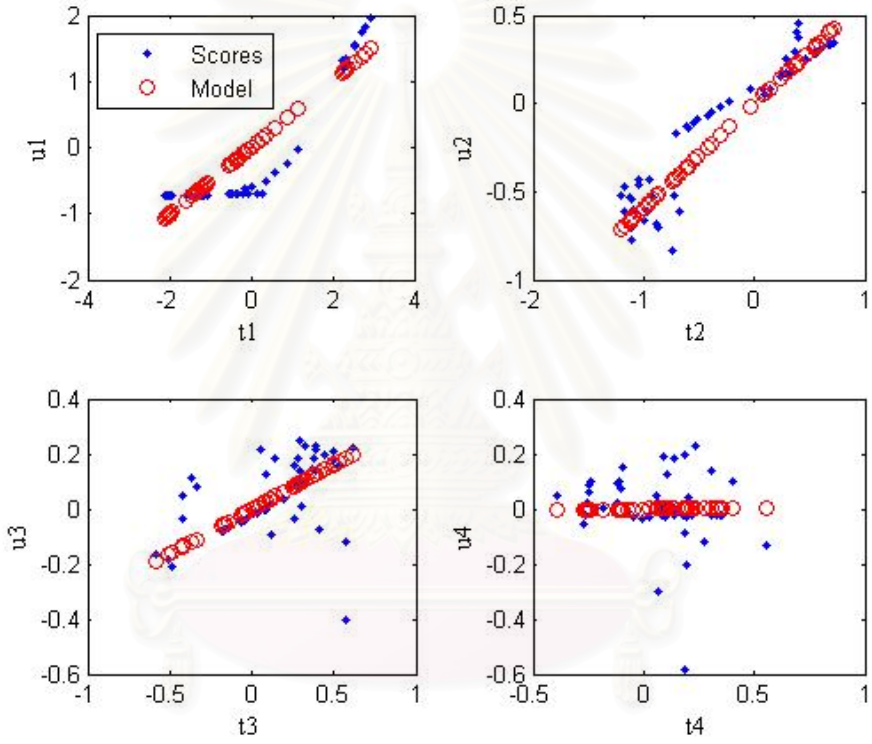

Figure 4.9: Performance of the linear inner models for the four factors extracted.

Table 4.7: RMSE between u and $\hat{u}$

Factor	**Neurons in 1^{nd} layer**	**RMSE $\times 10^{-2}$**
1	2	5.33294
	3	5.17525
	4	5.16767
	5	5.15841
2	2	3.67116
	3	3.48376
	4	3.48008
	5	3.47776
3	2	3.22133
	3	2.91504
	4	2.89216
	5	2.86117
4	2	2.75119
	3	2.72233
	4	2.75223
	5	2.74997

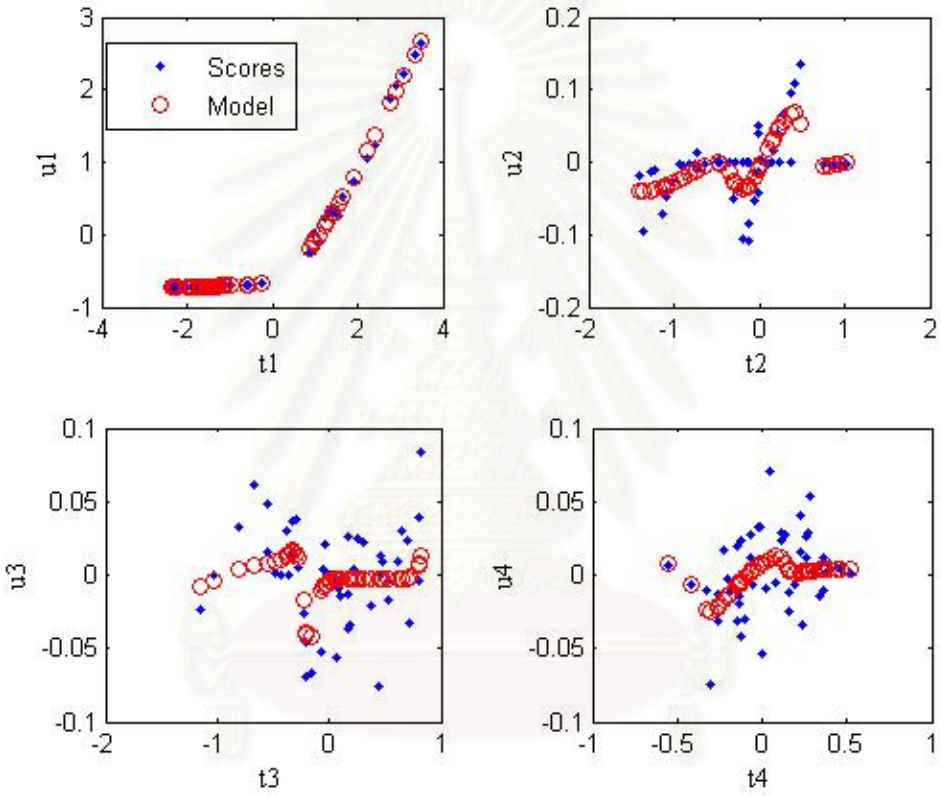

Figure 4.10: Performance of the neural network inner models for the four factors extracted.

Form table 4.7, the number of neurons in hidden layer structures for each factor have been chosen and performed as a following:

- three neuron for first factor (1-3-1),
- four neuron for second factor (1-4-1),
- four neuron for third factor (1-4-1), and
- three neuron for fourth factor (1-3-1).

Figure 4.10 showed the performance of the selected neural network models of inner relation in the latent space on the testing data. Only 100 data points were shown in the figure. As the figure, the neural network models were able to predict the latent scores for each factor.

For linear PLS, the number of factors for building PLS soft sensor model were three factors. From table 4.8, PRESS value of third factor was enough for prediction step since this value was comparatively small with other factors. Thus, we used three factor for prediction step of linear PLS model. For NNPLS, three factors were sufficient for extracting the relevant information between the input and output variables. The PRESS value between the model and the test data, given by factor in table 4.8, also indicated that three factors were efficient; hence the model for prediction purposes used only three factors.

Table 4.8: PRESS of liner PLS and NNPLS

No. of factors	**PRESS (linear PLS)** $\times 10^{-6}$	**PRESS (NNPLS)** $\times 10^{-7}$
1	175.75453	18.84320
2	27.75638	9.49016
3	9.78411	6.55450
4	9.75710	5.80730

4.5.4 Performances of soft sensors

The models was used to predict unseen data, which was generated using different random signals as inputs of soft sensor models. These data are the testing data set in table 4.5. Figure 4.11, figure 4.12, and figure 4.13 showed the soft sensor performances results of the linear PLS with three factors, NNPLS model with four factors, and the neural network model (4-5-3-1), respectively, in predicting the mass fraction of cyclohexanol between test sets and predicting sets.

For parameters of soft sensor models, we also showed corresponding parameters which were constructed and used for predicting the output variable in this research for the selected structure of neural network model (4-5-2-1), the linear PLS model, and the NNPLS model. The parameters contained weights, biases, and matrices in the prediction purposes for partial least squares methods. These values have been demonstrated in APPENDIX C.

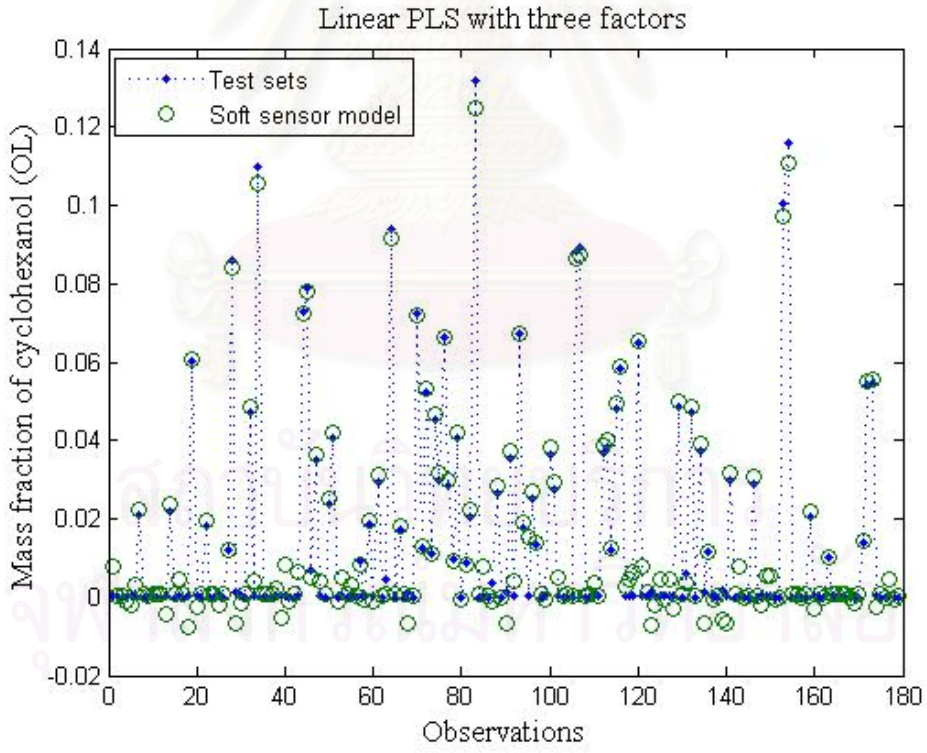

Figure 4.11: Prediction of soft sensor based on linear PLS.

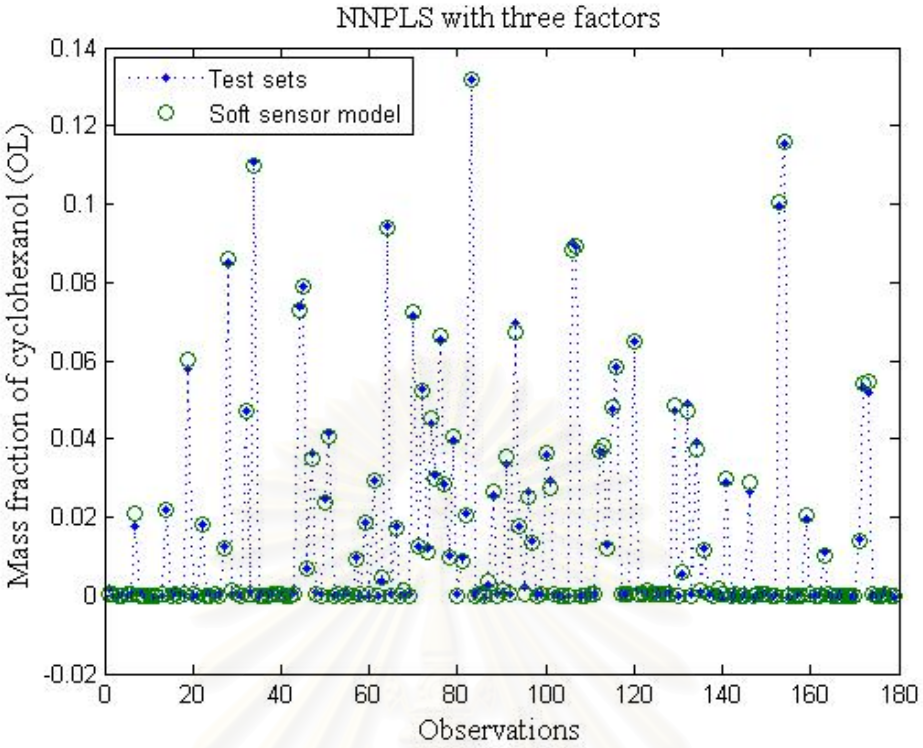

Figure 4.12: Prediction of soft sensor based on NNPLS.

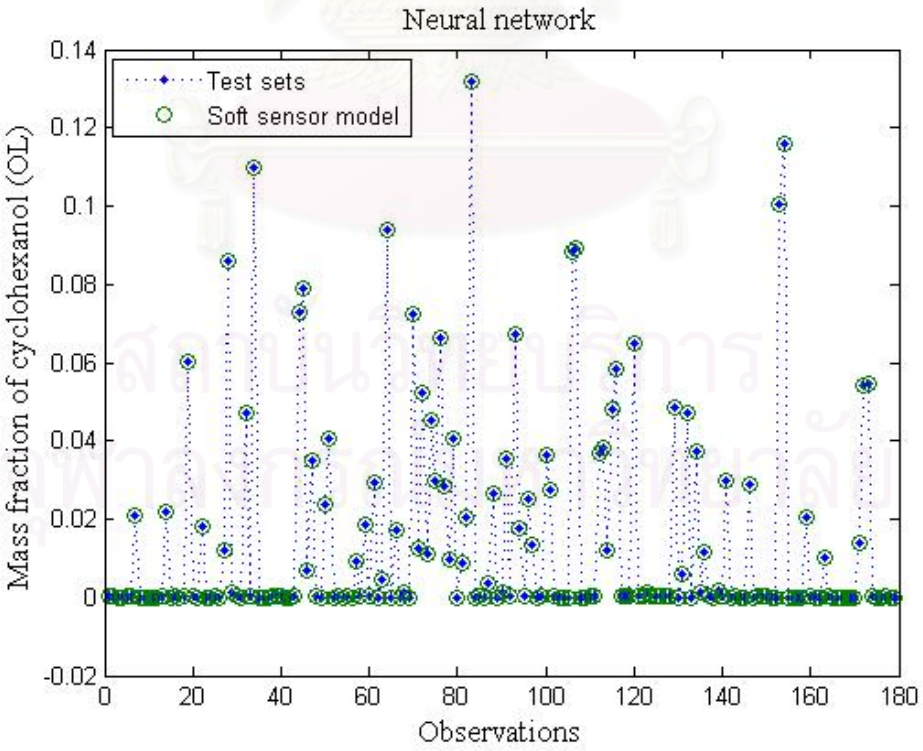

Figure 4.13: Prediction of soft sensor based on NN.

Table 4.8: Performance criteria of each soft sensor

Soft sensors	$RMSE$
Linear PLS	3.11921×10^{-3}
NNPLS	8.07334×10^{-4}
NN	1.10988×10^{-5}

Table 4.8, showed the root mean squared error of prediction (RMSE) of the linear-PLS model, the NNPLS model and the neural network model for the soft sensors. Comparatively, Linear-PLS gave the worst performance for predicting and NN gave the best performance for predicting. Fortunately, NNPLS models predicted adequately for prediction. Three factors were extracted by the NNPLS model for prediction. This meant that the dimension of the MIMO (Multiple Inputs and Multiple Outputs) problem in the original space has been reduced from a four-input by one-output to three series of SISO (Single Input and Single Output) problems in the latent space. Therefore, by using the NNPLS soft sensor model, the training time for neural training step in building model and computing time for prediction step were reduced from original NN model approach and the NNPLS also performed as good as a neural network soft sensor model.

CHAPTER V

CONCLUSIONS AND RECOMMENDATIONS

5.1 Introduction

For successful monitoring and control of chemical plants, the measurement of important variable is essential. An increasing number of expensive measuring devices need to be installed. Therefore, an effective solution to this drawback is represented by the availability of powerful low cost data processing systems. This makes it possible to utilize soft sensors as mentioned in this thesis.

5.2 Conclusions

The approaches for estimation of target variable have been proposed at the column D in cyclohexanone unit. The proposed approaches being soft sensors based on multilayer Feedforward with backpropagation (MLFF with BP) of artificial neural networks (ANNs), partial least squares (PLS) regression, and neural networks artificial partial least squares (NNPLS) were used to predict the concentration of cyclohexanol for top product of the column D at Thai Caprolactam Public Co., Ltd. Generally, real plant data have small variations or smooth responses. Therefore, if we used only real plant data for constructing soft sensors, they have the limitation to implement in wide ranges operation of processes. Consequently, when the operating point shifted or external conditions changed from normal condition, the model soft sensor cannot accurately implement for estimating performance. According to these problems, this work used another wide ranges data source from simulated data that were simulated by HYSYS simulator. These two data sources were mixed together to calculate parameters of the soft sensor model based on linear PLS model. For MLFF and NNPLS model, the wide range simulated data were used to pre-train or pre-calibrate parameters of soft sensor

models and used the real plant data for find-tuning the parameters again. In case study, the results for soft sensors which have four inputs as temperature of the column D indicated that the three the soft sensors showed satisfactory estimating performances and soft sensor based on MLFF method gave better estimating performance over the NNPLS and linear PLS method.

5.3 Recommendations

To develop the efficiency of soft sensors in order to estimate target variable, the data which are reliable are very important to building soft sensor. Since received data may have certainly error, they have effect on accuracy for prediction. Therefore, it is excellent to build soft sensors if available data are highly reliable.

Form these results, although, the soft sensor based on ANNs preformed high performance for prediction, the soft sensor based on NNPLS also showed as good prediction as neural network model. In advantages of NNPLS, in this work, the number of parameters or dimension of the problem were reduced from MIMO in original space to three SISO in latent space. According to these features, they resulted in reduction of calibration and calculation time for operations. In another good thing, the NNPLS model is more suitable for control proposes since almost model base controllers have complex formulations. Then, it is necessary to reduce the dimensionality of the problem for improving performance resulting form less complex formulations. In another word, PLS converts the MIMO regression problem into a SISO regression problem for decreasing the complexities and combines piece the results of the SISO regression problems together to arrive at the result for the MIMO problem.

REFERENCES

Albertos, P. and Goodwin, G.C. Virtual sensors for control applications. Annual Reviews in Control 26 (2002): 101-112.

Assis, A. J. and Filho, R. M. Soft sensors development for on-line bioreactor state estimation.Computers and Chemical Engineering 24 (2000): 1099-1103.

Basheer, I.A. and Hajmeer, M. Artificial neural networks: fundamentals, computing, design, and application. Journal of Microbiological Methods 43 (2000): 3-31.

Dufour, P., Bhartiya, S., Dhurjati, P. S. and Doyle III, F. J. Neural network-based software sensor: training set design and application to a continuous pulp digester. Control Engineering Practice 13 (2005): 135-143.

Edgar, T.F., Himmelblau, D.M. and Lasdon, L.S. Optimization of chemical processes. 2 nd ed. Singapore: McGraw-Hill, 2001.

Fortuna, L., Graziani, S. and Xibilia, M.G. Soft sensors for product quality monitoring in debutanizer distillation columns. Control Engineering Practice 13 (2005): 499-508.

Fortuna, L., Rizzo, A., Sinatra, M. and Xibilia, M.G. Soft analyzers for a sulfur recovery unit. Control Engineering Practice 11 (2003): 1491-1500.

Geladi, P. and Kowalski, B. R. Partial least-squares regression: a tutorial. Analytica Chimica Acta 185 (1986): 1-17.

Hagan M.,Demuth H. and Beale M. Neural Network Design. Boston: PWS, 1996.

Höskuldsson A. PLS regression methods. Journal of chemometrics 2 (1988): 211-228.

Jackson, J. E. A User's Guide to Principal Components. New York: John Wiley & Sons, INC., 1991.

Kamohara, H., Takinami, A., Takeda, M., Kano, M., Hasebe, S. and Hashimoto, I. Product quality estimation and operating condition monitoring for industrial ethylene fractionator. Journal of Chemical Engineering of Japan 37 (2004): 422-428.

Kreyszig, E. Advanced Engineering Mathematics. New York: John Wiley & Sons, 1999.

Lee, J. K., and Han, C. Industrial application of multivariate statistical approaches to polymerization processes. The APCChE Congress 8 (1999): 16-19.

Lippman, R.P. An introduction to computing with neural nets. IEEE ASSP Magazine April (1987): 4-22.

Mejdell, T. and Skogestad, S. Estimation of distillation composition from multiple temperature measurements using partial-least-squares regression. Industrial & Engineering Chemistry Research 30 (1991a): 2543-2555.

Mejdell, T. and Skogestad, S. Composition estimator in a pilot-plant distillation column using multiple temperatures. Industrial & Engineering Chemistry Research 30 (1991b): 2555-2564.

Ohshima, M. and Tanigaki, M. Quality control of polymer production processes. Ohshima, M. and Tanigaki, M. Quality control of polymer production processes. Journal of Process Control 10 (2000): 135-148.

Park, S. and Han, C. A nonlinear soft sensor based on multivariate smoothing procedure for quality estimation in distillation columns. Computers and Chemical Engineering 24 (2000): 871-877.

Patterson, D. W. Artificial Neural Networks: Theory and Application, Singapore: Prentice-Hall, 1996.

Qin, S. J., and McAvoy, T. J. Nonlinear PLS modeling using neural networks. Computers and Chemical Engineering 16 (1992): 379-391.

Qin, S. J., and McAvoy, T. J. Nonlinear FIR modeling via a neural net PLS approach. Computers and Chemical Engineering 20 (1996): 147-159.

Radhakrishnan, V.R. and Mohamed, A.R. Neural networks for the identification and control of blast furnace hot metal quality. Journal of Process Control 10 (2000): 509-524.

Tang, K. and Li, T. Comparison of different partial least-squares methods in quantitative structure-activity relationships. Analytica Chimica Acta 476 (2003): 85-92.

Wold, S., Kettaneh-Wold, N., and Skagerberg, B. Non-linear PLS modelling. Chemometrics and Intelligent Laboratory Systems 7 (1989): 53-65.

Wold, S. Nonlinear partial least squares modeling. II. Spline inner relation. Chemometrics and Intelligent Laboratory Systems 14 (1992): 71-84.

Zamprogna, E., Barolo, M. and Seborg, D. E. Estimating product composition profiles in batch distillation via partial least squares regression. Control Engineering Practice 12 (2004): 917-929.

Zamprogna, E., Barolo, M. and Seborg, D. E. Optimal selection of soft sensor inputs for batch distillation columns using principal component analysis. Journal of Process Control 15 (2005): 39-52.

APPENDICES

APPENDIX A

CORRESPONDING MATHEMATICS

This section will attempt to provide some elementary background mathematical skills that will be required to understand procedure corresponding to this work.

A.1 Matrix algebra

A.1.1 Definitions of Matrices

A matrix is defined as an orderly array of numbers. Examples of matrices are the following:

$$\begin{bmatrix} 1 & 2 \\ 5 & 9 \end{bmatrix} \tag{A.1}$$

$$\begin{bmatrix} 8 & 1 & 0 \\ 7 & 3 & 6 \end{bmatrix} \tag{A.2}$$

In particular, the $r \times c$ matrix A denotes an array of numbers consisting of r rows and c columns.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1c} \\ a_{21} & a_{22} & \cdots & a_{2c} \\ \vdots & \vdots & \vdots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rc} \end{bmatrix} \tag{A.3}$$

If $r = c$, the numbers of rows and columns are the same and the matrix is said to be a square matrix.

A.1.2 Matrix Transpose

If, for any matrix A, a new matrix B is formed by interchanging the rows and columns. The resultant matrix is said to be the transpose of the original matrix and is denoted by A'. For instance, if

$$A = \begin{bmatrix} 2 & 7 & -5 & 1 \\ 8 & 3 & 0 & 9 \end{bmatrix}$$

$$A' = \begin{bmatrix} 2 & 8 \\ 7 & 3 \\ -5 & 0 \\ 1 & 9 \end{bmatrix} \tag{A.4}$$

A.1.3 Diagonal Matrices

A square matrix whose only nonzero elements are on the diagonal is called a diagonal matrix.

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 8 \end{bmatrix} \tag{A.5}$$

In particular, a square matrix that has ones on the diagonal and zeros elsewhere is denoted by I and is called a unit or identity matrix.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.6}$$

A.1.4 Vectors

A matrix that has only a single row is called a row vector and a matrix that has only a single column is called a column vector. The transpose of a row vector is a column vector and vice versa. A matrix with only a single row and column is a scalar.

A.1.5 Orthonormal and Orthogonal Matrices

An orthonormal matrix is a square matrix with the following properties:

1. $\mid A \mid = \pm 1$, where $\mid A \mid$ is the determinant of A.

2. The sum of squares of any row or column is equal to unity.

$$\sum_{i=1}^{p} a_{ij}^2 = \sum_{j=1}^{p} a_{ij}^2 = 1 \tag{A.7}$$

   for i,j.

3. The sum of crossproducts of any two columns is equal to zero and implies that the coordinate axes, which these two columns represent, intersect at an angle of 90^o.

$$\sum_{i=1}^{p} a_{ij} a_{ik} = 0 \tag{A.8}$$

   for all $j \neq k$.

This implies that $AA' = \mathrm{I}$. If A is orthonormal, $A^{-1} = A'$ where A^{-1} is the inverse of A, to be defined in section A.1.6. A matrix that satisfies Condition 3 but not Conditions 1 and 2 is said to be orthogonal.

A.1.6 Inversion

The matrix A must be a square nonsingular matrix so that A has its inverse matrix A^{-1}. A^{-1} is defined as that matrix which when multiplied by the matrix A will yield the identity matrix I:

$$AA^{-1} = I = A^{-1}A \tag{A.9}$$

A.2 Statistics

A.2.1 Standard Deviation

A standard deviation is the positive square root of the variance. It is defined as following:

$$S = \frac{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})}}{n-1} \tag{A.10}$$

with the summation running from 1 to n. x are variables and $\bar{x}$ is average value of x variables. The standard deviation can show that a Set of variable that has a much larger standard deviation indicates that the data much more spread out from the mean.

A.2.2 Variance

A variance is another measure of the spread of data in a data set. In fact it is almost identical to the standard deviation. The formula is this:

$$S^2 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n-1} \tag{A.11}$$

Both these measurements are measures of the spread of the data. Standard deviation is the most common measure, but variance is also used.

A.2.3 Covariance

A standard deviation and a variance can operate on 1 dimension. In fact many kinds of problem have more than one dimension, so we have to find the relationship between one dimension and other dimensions.

A covariance is such a measure of association between two dimensions obtained as the expected value of the product of two variables around their means. If we calculate the covariance between one dimension and itself, we obtain the variance. The formula is this:

$$Cov(x_1, x_2) = \frac{\sum_{i=1}^{n}(x_{i1} - \bar{x_1})(x_{i2} - \bar{x_2})}{n-1} \tag{A.12}$$

A.2.4 Covariance Matrix

A covariance Matrix is a square matrix that contains the variances and covariances among a set of variables, x_1, x_1, $\cdots$, x_n. The main diagonal elements of the matrix are the variances of the variables and the off diagonal elements are the covariances between x_i and x_j. Also called the variance-covariance matrix. For 3 dimensions (x,y,z)

$$Cov(x,y,z) = \begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix} \tag{A.13}$$

Main diagonal can find that the covariance value is between one of the dimensions and itself. These values are the variances for it dimension. Another point is that as $Cov(A,B) = Cov(B,A)$ so, the matrix is symmetrical about the main diagonal.

A.2.5 Eigenvectors and Eigenvalues

From the standpoint of engineering applications, eigenvalue problems are among the most important problems in connection with matrices. The basic concepts are as follows.

Let A be $n \times n$ matrix and consider the vector equation

$$Ax = \lambda x \tag{A.14}$$

where λ is a constant value. It is clear that the zero vector $x = 0$ is a solution of this equation for any value of λ. A value of λ for the equation has a solution $x \neq 0$ is called an eigenvalue or characteristic value of the matrix A. The corresponding solutions $x \neq 0$ of the equation are called eigenvectors or characteristic vectors of A corresponding to that eigenvalue λ. Properties of eigenvectors.

Properties of eigenvectors.

- Eigenvectors can only be found for square matrices and not every square matrix has eigenvectors.
- If A is an $n \times n$ matrix, then there will be n eigenvalues and n corresponding eigen vectors.
- All the eigenvectors of a matrix are perpendicular or orthogonal. In general case, eigenvectors will be normalized to length that equal to one.

APPENDIX B

NORMALIZATION METHODS

This section performed the calculation procedures of scaling or normalizing input and output data sets for using in the calibration step.

B.1 Scaling data for NN soft sensors

For the calibration step, raw or original data were scaled by Min-Max normalization (as proposed in section 3.1.1) using the following equations:

$$Input_{scaled} = \frac{Input_{actual} - (Input_{min} - 1)}{(Input_{max} + 1) - (Input_{min} - 1)} \tag{B.1}$$

$$Output_{scaled} = \frac{Output_{actual} - Output_{min}}{Output_{max} - Output_{min}} \tag{B.2}$$

The scaled up parameters which are the min and max of each variable are showed in table B.1.

Table B.1: The scaled up parameter of the neural network soft sensors

No.	Variables	min	max
Input: 1	T_{Top}	57.367	59.991
Input: 2	$T_{HighMid}$	66.110	75.296
Input: 3	T_{LowMid}	73.217	81.525
Input: 4	T_{Bot}	89.124	94.248
Output	OL	0.000	1.000

For the prediction step, the neural network soft sensor output value which being the mass fraction of cyclohexanol was rescaled to find the value in the original units as described in Eq. B.3.

$$Output_{actual} = Output_{scaled} \times (Output_{max} - Output_{min}) + Output_{min} \tag{B.3}$$

B.2 Scaling data for PLS soft sensors

For the calibration step, raw or original data of linear PLS and NNPLS were scaled by Z-score Standardization (as proposed in section 3.1.1) as the following equations:

$$Input_{scaled} = \frac{Input_{actual} - Input_{mean}}{Input_{std}} \tag{B.4}$$

$$Output_{scaled} = \frac{Output_{actual} - Output_{mean}}{Output_{std}} \tag{B.5}$$

The scaled up parameters which are the mean and standard deviation of each variable are showed in table B.2.

Table B.2: The scaled up parameter of the partial least squares soft sensors

No.	**Variables**	**mean**	**standard deviation (std.)**
Input: 1	T_{Top}	57.775	0.564
Input: 2	$T_{HighMid}$	69.737	3.723
Input: 3	T_{LowMid}	77.152	3.447
Input: 4	T_{Bot}	92.158	1.153
Output	OL	0.0200	0.028

For the prediction step, the partial least squares soft sensor output value which being the mass fraction of cyclohexanol was rescaled to find the value in the original units as described in Eq. B.6.

$$Output_{actual} = Output_{scaled} \times Output_{std} + Output_{mean} \tag{B.6}$$

APPENDIX C

PARAMETER VALUES OF SOFT SENSORS

This section presented the parameters of soft sensor models for neural model, linear PLS, and NNPLS which were suitable selected structures as shown in CHAPTER IV.

C.1 Soft sensor based on neural model

For understanding, we proposed the selected structure of neural networks model (4-5-3-1) in graphically information as below figure. Furthermore, The $w_{i,j}^m$ and b_i^m are defined for the weights from node j in layer $m-1$ to node i in layer m and the bias of node i in layer m, respectively. In this work, transfer functions of hidden layer(s) were sigmoid functions and the transfer function of output layer was linear function (as shown in section 3.4.2).

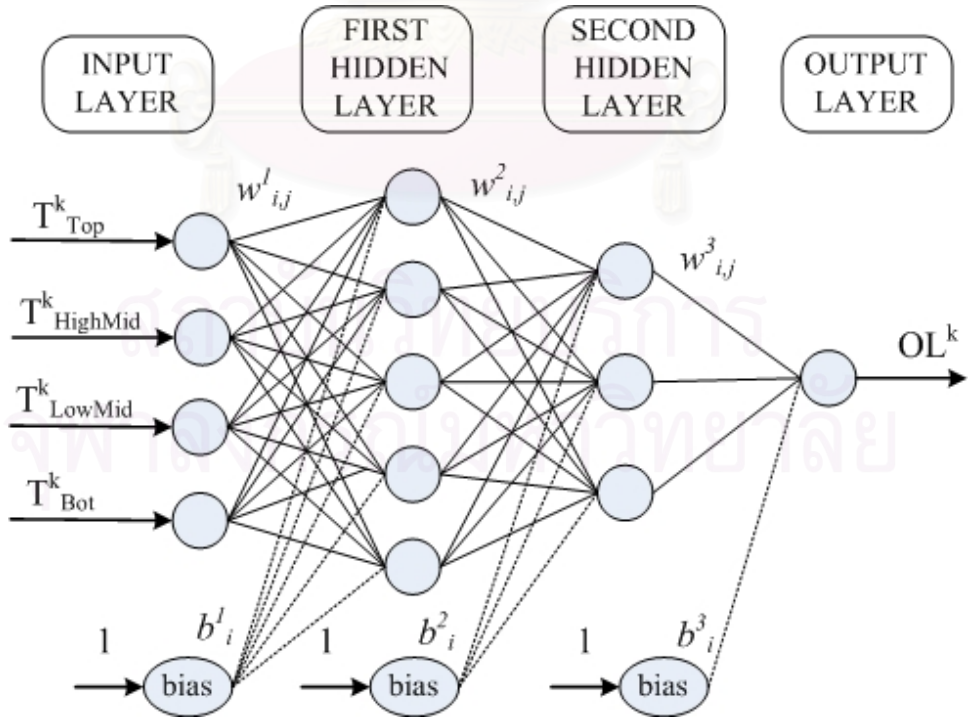

Figure C.1: The selected architecture of the neural network soft sensor.

Table C.1: Weights from input layer to first hidden layer

$w^1_{1,1} = 2.687$	$w^1_{1,2} = 1.373$	$w^1_{1,3} = 4.698$	$w^1_{1,4} = 0.338$
$w^1_{2,1} = 5.159$	$w^1_{2,2} = 1.662$	$w^1_{2,3} = 5.692$	$w^1_{2,4} = 0.203$
$w^1_{3,1} = 5.304$	$w^1_{3,2} = -8.794$	$w^1_{3,3} = -13.613$	$w^1_{3,4} = -0.319$
$w^1_{4,1} = 19.389$	$w^1_{4,2} = -0.963$	$w^1_{4,3} = 9.228$	$w^1_{4,4} = 0.137$
$w^1_{5,1} = 3.615$	$w^1_{5,2} = 3.583$	$w^1_{5,3} = -9.904$	$w^1_{5,4} = -0.347$

Table C.2: Weights from first hidden layer to second hidden layer

$w^2_{1,1} = 1.826$	$w^2_{1,2} = 3.799$	$w^2_{1,3} = -2.646$	$w^2_{1,4} = -1.392$	$w^2_{1,5} = 12.244$
$w^2_{2,1} = -1.182$	$w^2_{2,2} = 2.197$	$w^2_{2,3} = -9.820$	$w^2_{2,4} = -17.876$	$w^2_{2,5} = -3.822$
$w^2_{3,1} = 1.069$	$w^2_{3,2} = 2.335$	$w^2_{3,3} = -13.881$	$w^2_{3,4} = 0.429$	$w^2_{3,5} = 2.806$

Table C.3: Weights from second hidden layer to output layer and all its biases

$w^3_{1,1} = -0.0007$	$b^1_1 = -7.615$	$b^2_1 = -9.225$	$b^3_1 = 0.0004$
$w^3_{1,2} = 1.835$	$b^1_2 = -6.872$	$b^2_2 = 2.519$	
$w^3_{1,3} = 2.459$	$b^1_3 = 9.261$	$b^2_3 = -8.702$	
	$b^1_4 = -11.125$		
	$b^1_5 = 5.298$		

C.2 Soft sensor based on linear PLS

For linear PLS structure, we showed the the structure (with three factors) in graphically information as shown figure C.2.

C.2.1 Calibration step

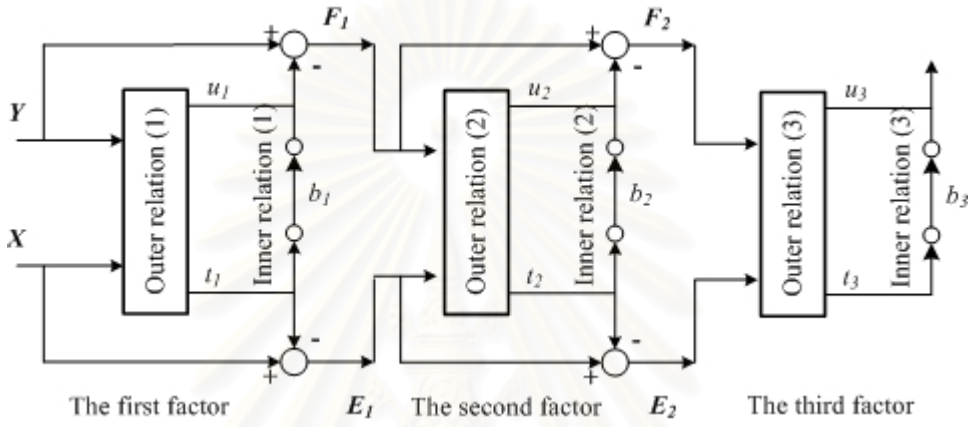

Figure C.2: Linear PLS structure soft sensor with three factors.

C.2.2 Prediction step

This step was done by decomposing the X block and building up the Y block. For this purpose, p' ,q' , w' and b from the calibration part were saved for every PLS factor (as mentioned in section 3.3). The independent blocks are decomposed and the dependent block is built up. For the X block, t is estimated by multiplying X by w as in the model building part. For this work, a was equal to 3.

$$\hat{t}_h = E_{h-1} w_h \tag{C.1}$$

$$E_h = E_{h-1} - \hat{t}_h p'_h \tag{C.2}$$

For the Y block:

$$Y = F_h = \sum_{i=1}^{a} (b_h \hat{t}_h q'_h) \tag{C.3}$$

where the summation is over h for all the factors (a) one wants to include and $X = E_0$, $Y = F_a$.

The corresponding matrices can be listed as a following:

$$P = \begin{bmatrix} 0.49925 & 0.40848 & 0.47749 \\ 0.54455 & 0.30443 & -0.62735 \\ 0.48865 & -0.52551 & -0.29926 \\ 0.46415 & -0.68141 & 0.53748 \end{bmatrix} \tag{C.4}$$

$$W = \begin{bmatrix} 0.63426 & 0.52926 & 0.48094 \\ 0.60792 & 0.18257 & -0.63206 \\ 0.38082 & -0.61118 & -0.29281 \\ 0.35813 & -0.59737 & 0.5325 \end{bmatrix} \tag{C.5}$$

$$Q = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{C.6}$$

$$B = diag \begin{bmatrix} b_1 = 0.50617, & b_2 = 0.59185, & b_3 = 0.32375 \end{bmatrix} \tag{C.7}$$

C.3 Soft sensor based on NNPLS

For NNPLS structure, we also illustrated the the structure (with three factors) in graphically information as shown figure B.3. In the same way, when the NNPLS model was created, we saved p_h, q_h and w_h for prediction proposes. For inner model, the neural structure of the first factor was 1-3-1, the neural structure of the second factor was 1-4-1 and the neural structure of the third factor was 1-4-1. All weights and biases of each factor have been demonstrated in table C.4. Besides, transfer functions of hidden layer were sigmoid functions and the transfer function of output layer was linear function. For this work, a was equal to 3.

C.3.1 Calibration step

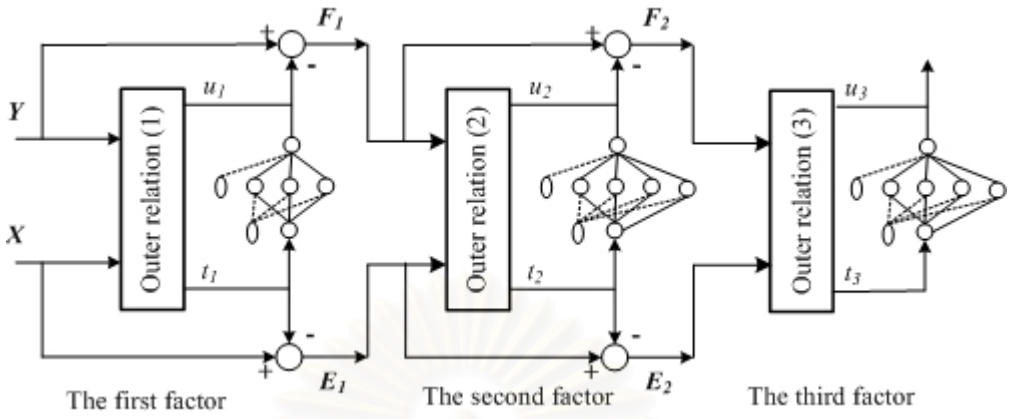

Figure C.3: NNPLS structure soft sensor with three factors.

C.3.2 Prediction step

For the X block, t is estimated by multiplying X by w as in the calibration part.

$$t_h = E_{h-1} w_h \tag{C.8}$$

$$E_h = E_{h-1} - t_h p'_h \tag{C.9}$$

For the Y block, u is calculated by t and PLS coefficients.

$$Y = F_a = \sum_{h=1}^{a} \theta(t_h) q'_h \tag{C.10}$$

where the summation is over h for all the factors (a) one wants to include, $X = E_0$, $Y = F_a$, and $\theta(t_h) = \hat{u}$ for each factor (component) h.

The corresponding matrices can be also listed as a following:

$$P = \begin{bmatrix} 0.50219 & 0.17476 & -0.66995 \\ 0.5444 & 0.48807 & 0.17666 \\ 0.48738 & -0.33442 & 0.7052 \\ 0.46249 & -0.78702 & 0.1505 \end{bmatrix} \tag{C.11}$$

$$W = \begin{bmatrix} 0.63426 & -0.09603 & -0.61667 \\ 0.60621 & 0.65074 & 0.10779 \\ 0.37832 & -0.0711 & 0.78693 \\ 0.35691 & -0.85817 & 0.08555 \end{bmatrix} \tag{C.12}$$

$$Q = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} \tag{C.13}$$

$$B = \begin{bmatrix} \theta^1, & \theta^2, & \theta^3 \end{bmatrix} \tag{C.14}$$

where θ^a = inner neural model of a factor.

Table C.4: Weights and biases of each factor inner model

First factor			
$w^1_{1,1}$ = -1.475	$w^2_{1,1}$ = -37.149	b^1_1 = -2.060	b^2_1 = 44.883
$w^1_{2,1}$ = -0.060	$w^2_{1,2}$ = -83.949	b^1_2 = 0.222	
$w^1_{3,1}$ = -1.3971	$w^2_{1,3}$ = 42.679	b^1_3 = -1.953	
Second factor			
$w^1_{1,1}$ = 24.776	$w^2_{1,1}$ = -0.100	b^1_1 = -13.113	b^2_1 = 0.101
$w^1_{2,1}$ = -4.435	$w^2_{1,2}$ = -0.282	b^1_2 = -0.385	
$w^1_{3,1}$ = -11.200	$w^2_{1,3}$ = 0.158	b^1_3 = -3.409	
$w^1_{4,1}$ = -9.070	$w^2_{1,4}$ = -0.019	b^1_4 = -8.824	
Third factor			
$w^1_{1,1}$ = -37.866	$w^2_{1,1}$ = -0.060	b^1_1 = 32.209	b^2_1 = 0.030
$w^1_{2,1}$ = -27.240	$w^2_{1,2}$ = 5.189	b^1_2 = -5.941	
$w^1_{3,1}$ = -26.708	$w^2_{1,3}$ = -5.177	b^1_3 = -5.799	
$w^1_{4,1}$ = 5.998	$w^2_{1,4}$ = 0.027	b^1_4 = 6.267	

Form these parameters, the example for calculation procedures of soft sensors based on NN, linear PLS, and NNPLS have been proposed in appendix D.

APPENDIX D

EXAMPLES FOR SOFT SENSOR CALCULATIONS

This section illustrated calculation examples of soft sensor models for selected structures of neural model, linear PLS, and NNPLS which were used to estimate mass fraction of cyclohexanol (OL).

D.1 Soft sensor based on neural model

We selected No.8 data set in table E.1 that inputs and output values could be shown as follow:

- OL = 0.0515 (y),
- Top temp. = 58.416 (x_1),
- High middle temp. = 74.210 (x_2),
- Low middle temp. = 80.658 (x_3), and
- Bottom temp. = 93.050 (x_4).

Inputs were normalized by Eq. B.1 and input_{scaled} could be shown as a following:

- $x_{1scaled} = 0.4431$,
- $x_{2scaled} = 0.8136$,
- $x_{3scaled} = 0.8189$, and
- $x_{4scaled} = 0.6914$.

Input layer to first hidden layer

Node 1st at first hidden layer

$$\begin{aligned}\eta_1^1 &= \sum_{j=1}^{4} w_{1,j}^1 x_j + b_1^1 \\ &= 0.4431 \times 2.687 + 0.8136 \times 1.373 + 0.8189 \times 4.698 + 0.6914 \times 0.338 \\ &\quad + - (7.615) \\ &= -1.2267\end{aligned}$$

$$\begin{aligned}x_1^1 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-(-1.2267)}} \\ &= 0.2268\end{aligned}$$

where η_i^k = summation of each input multiplied by its weight with bias and x_i^k = output of η from transfer function at node i in layer k.

Node 2nd at first hidden layer

$$\begin{aligned}\eta_2^1 &= \sum_{j=1}^{4} w_{2,j}^1 x_j + b_2^1 \\ &= 0.4431 \times 5.159 + 0.8136 \times 1.662 + 0.8189 \times 5.692 + 0.6914 \times 0.203 \\ &\quad + (-6.872) \\ &= 1.5674\end{aligned}$$

$$\begin{aligned}x_2^1 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-1.5674}} \\ &= 0.8274\end{aligned}$$

Node 3rd at first hidden layer

$$\begin{aligned}\eta_3^1 &= \sum_{j=1}^{4} w_{3,j}^1 x_j + b_3^1 \\ &= 0.4431 \times 5.304 + 0.8136 \times (-8.794) + 0.8189 \times (-13.613) + 0.6914 \times (-0.319) \\ &\quad + 9.261 \\ &= -6.9110\end{aligned}$$

$$\begin{aligned} x_3^1 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-(-6.9110)}} \\ &= 0.001 \end{aligned}$$

Node 4th at first hidden layer

$$\begin{aligned} \eta_4^1 &= \sum_{j=1}^{4} w_{4,j}^1 x_j + b_4^1 \\ &= 0.4431 \times 19.389 + 0.8136 \times (-0.963) + 0.8189 \times 9.228 + 0.6914 \times 0.137 \\ &\quad +(-11.125) \\ &= 4.3336 \end{aligned}$$

$$\begin{aligned} x_4^1 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-4.3336}} \\ &= 0.9871 \end{aligned}$$

Node 5th at first hidden layer

$$\begin{aligned} \eta_5^1 &= \sum_{j=1}^{4} w_{5,j}^1 x_j + b_5^1 \\ &= 0.4431 \times 3.165 + 0.8136 \times 3.583 + 0.8189 \times (-9.904) + 0.6914 \times (-0.347) \\ &\quad +5.298 \\ &= 1.4654 \end{aligned}$$

$$\begin{aligned} x_5^1 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-1.4654}} \\ &= 0.8124 \end{aligned}$$

First hidden layer to second hidden layer

Node 1st at second hidden layer

$$\begin{aligned} \eta_1^2 &= \sum_{j=1}^{5} w_{1,j}^2 x_j^1 + b_1^2 \\ &= 0.2268 \times 1.826 + 0.8274 \times 3.799 + 0.001 \times (-2.646) + 0.9871 \times (-1.392) \\ &\quad 0.8124 \times 12.244 + (-9.225) \\ &= 2.9020 \end{aligned}$$

$$\begin{aligned} x_1^2 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-2.9020}} \\ &= 0.9480 \end{aligned}$$

Node 2nd at second hidden layer

$$\begin{aligned} \eta_2^2 &= \sum_{j=1}^{5} w_{2,j}^2 x_j^1 + b_2^2 \\ &= 0.2268 \times (-1.182) + 0.8274 \times 2.197 + 0.001 \times (-9.820) + 0.9871 \times (-17.876) \\ & \quad 0.8124 \times (-3.822) + 2.519 \\ &= -16.6900 \end{aligned}$$

$$\begin{aligned} x_2^2 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-(-16.6900)}} \\ &= 0 \end{aligned}$$

Node 3nd at second hidden layer

$$\begin{aligned} \eta_3^2 &= \sum_{j=1}^{5} w_{3,j}^2 x_j^1 + b_3^2 \\ &= 0.2268 \times 1.069 + 0.8274 \times 2.335 + 0.001 \times (-13.881) + 0.9871 \times 0.4289 \\ & \quad 0.8124 \times 2.806 + (-8.702) \\ &= -3.8384 \end{aligned}$$

$$\begin{aligned} x_3^2 &= \frac{1}{1+e^{-\eta}} \\ &= \frac{1}{1+e^{-(-3.8384)}} \\ &= 0.0211 \end{aligned}$$

Second hidden layer to output layer

Node 1st at output layer

$$\begin{aligned}\eta_1^3 &= \sum_{j=1}^{3} w_{1,j}^3 x_j^2 + b_1^3 \\ &= 0.9480 \times -0.0007 + 0 \times 1.836 + 0.0211 \times 2.459 + 0.0004 \\ &= 0.05149\end{aligned}$$

$$\begin{aligned}x_1^3 &= 0.05149 \\ &= y_{scaled}\end{aligned}$$

The y_{scaled} was rescaled to find the value in the original units as described in Eq. B.3.

$$y(OL) = 0.05149$$

The error between y_{actual} and y_{cal} was 0.00001 (0.0515 - 0.05149).

D.2 Soft sensor based on linear PLS

We selected No.8 data set in table E.1 that inputs and output values could be shown as follow:

- OL = 0.0515 (y),
- Top temp. = 58.416 (x_1),
- High middle temp. = 74.210 (x_2),
- Low middle temp. = 80.658 (x_3), and
- Bottom temp. = 93.050 (x_4).

Inputs were normalized by Eq. B.4 and $input_{scaled}$ could be shown as a following:

- $x_{1scaled}$ = 1.1362,

- $x_{2scaled} = 1.2015$,
- $x_{3scaled} = 1.0170$, and
- $x_{4scaled} = 0.7733$.

First factor

For X block:

$$E_0 = X_{Scaled} = \begin{bmatrix} 1.1362 & 1.2015 & 1.0170 & 0.7733 \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 0.63426 \\ 0.60792 \\ 0.38082 \\ 0.35813 \end{bmatrix}$$

$$\begin{aligned} \hat{t}_1 &= E_0 w_1 \\ &= 2.1153 \end{aligned}$$

$$p_1' = \begin{bmatrix} 0.49925 & 0.54455 & 0.48865 & 0.46415 \end{bmatrix}$$

$$\begin{aligned} E_1 &= E_0 - \hat{t}_1 p_1' \\ &= \begin{bmatrix} 0.0802 & 0.0496 & -0.0166 & -0.2085 \end{bmatrix} \end{aligned}$$

For Y block:

$$q_1 = \begin{bmatrix} 1 \end{bmatrix}$$

$$b_1 = 0.50617$$

$$\begin{aligned} F_1 &= b_1 \hat{t}_1 q_1' \\ &= 1.0707 \end{aligned}$$

Second factor

For X block:

$$E_1 = \begin{bmatrix} 0.0802 & 0.0496 & -0.0166 & -0.2085 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} 0.52926 \\ 0.18257 \\ -0.61118 \\ -0.59737 \end{bmatrix}$$

$$\begin{aligned} \hat{t}_2 &= E_1 w_2 \\ &= 0.1862 \end{aligned}$$

$$p_2' = \begin{bmatrix} 0.40848 & 0.30443 & -0.52551 & -0.68141 \end{bmatrix}$$

$$\begin{aligned} E_2 &= E_1 - \hat{t}_2 p_2' \\ &= \begin{bmatrix} 0.0041 & -0.0071 & 0.0812 & -0.0816 \end{bmatrix} \end{aligned}$$

For Y block:

$$q_2 = \begin{bmatrix} 1 \end{bmatrix}$$

$$b_2 = 0.59185$$

$$\begin{aligned} F_2 &= b_1 \hat{t}_1 q_1' + b_2 \hat{t}_2 q_2' \\ &= 1.1809 \end{aligned}$$

Third factor

For X block:

$$E_2 = \begin{bmatrix} 0.0041 & -0.0071 & 0.0812 & -0.0816 \end{bmatrix}$$

$$w_3 = \begin{bmatrix} 0.48094 \\ -0.63206 \\ -0.29281 \\ 0.5325 \end{bmatrix}$$

$$\begin{aligned} \hat{t}_3 &= E_2 w_3 \\ &= -0.0608 \end{aligned}$$

$$p_3' = \begin{bmatrix} 0.47749 & -0.62735 & -0.29926 & 0.53748 \end{bmatrix}$$

$$\begin{aligned} E_3 &= E_2 - \hat{t}_3 p_3' \\ &= \begin{bmatrix} 0.0331 & -0.0452 & 0.0630 & -0.0489 \end{bmatrix} \end{aligned}$$

For Y block:

$$q_3 = \begin{bmatrix} 1 \end{bmatrix}$$

$$b_3 = 0.32375$$

$$\begin{aligned} F_3 &= b_1 \hat{t}_1 q_1' + b_2 \hat{t}_2 q_2' + b_3 \hat{t}_3 q_3' \\ &= 1.1612 \\ &= y_{scaled} \end{aligned}$$

The y_{scaled} was rescaled to find the value in the original units as described in Eq. B.6.

$$y(OL) = 0.0525$$

The error between y_{actual} and y_{cal} was -0.001 (0.0515 - 0.0525).

D.3 Soft sensor based on NNPLS

We selected No.8 data set in table E.1 that inputs and output values could be shown as follow:

- OL = 0.0515 (y),
- Top temp. = 58.416 (x_1),
- High middle temp. = 74.210 (x_2),
- Low middle temp. = 80.658 (x_3), and

- Bottom temp. = 93.050 (x_4).

Inputs were normalized by Eq. B.4 and input$_{scaled}$ could be shown as a following:

- $x_{1scaled} = 1.1362$,
- $x_{2scaled} = 1.2015$,
- $x_{3scaled} = 1.0170$, and
- $x_{4scaled} = 0.7733$.

First factor

For X block:

$$E_0 = X_{Scaled} = \begin{bmatrix} 1.1362 & 1.2015 & 1.0170 & 0.7733 \end{bmatrix}$$

$$w_1 = \begin{bmatrix} 0.63826 \\ 0.60621 \\ 0.37832 \\ 0.35691 \end{bmatrix}$$

$$\begin{aligned} \hat{t}_1 &= E_0 w_1 \\ &= 2.1143 \end{aligned}$$

$$p_1' = \begin{bmatrix} 0.50219 & 0.54440 & 0.48738 & 0.46249 \end{bmatrix}$$

$$\begin{aligned} E_1 &= E_0 - \hat{t}_1 p_1' \\ &= \begin{bmatrix} 0.0744 & 0.0505 & -0.0135 & -0.2046 \end{bmatrix} \end{aligned}$$

For Y block:

$$q_1 = \begin{bmatrix} 1 \end{bmatrix}$$

$$\begin{aligned} F_1 &= \theta^1(t_1) q_1' \\ &= 1.0417 \end{aligned}$$

where θ^a = inner neural model of a factor.

Second factor

For X block:

$$E_1 = \begin{bmatrix} 0.0744 & 0.0505 & -0.0135 & -0.2046 \end{bmatrix}$$

$$w_2 = \begin{bmatrix} -0.09603 \\ 0.65074 \\ -0.071097 \\ -0.85817 \end{bmatrix}$$

$$\begin{aligned} \hat{t}_2 &= E_1 w_2 \\ &= 0.2022 \end{aligned}$$

$$p_2' = \begin{bmatrix} 0.17476 & 0.48807 & -0.33442 & -0.78702 \end{bmatrix}$$

$$\begin{aligned} E_2 &= E_1 - \hat{t}_2 p_2' \\ &= \begin{bmatrix} 0.0391 & -0.0482 & 0.0542 & -0.0454 \end{bmatrix} \end{aligned}$$

For Y block:

$$q_2 = \begin{bmatrix} 1 \end{bmatrix}$$

$$\begin{aligned} F_2 &= \theta^1(t_1) q_1' + \theta^2(t_2) q_2' \\ &= 1.0824 \end{aligned}$$

Third factor

For X block:

$$E_2 = \begin{bmatrix} 0.0391 & -0.0482 & 0.0542 & -0.0454 \end{bmatrix}$$

$$w_3 = \begin{bmatrix} -0.61667 \\ 0.10779 \\ 0.78693 \\ 0.08555 \end{bmatrix}$$

$$\begin{aligned} \hat{t}_3 &= E_2 w_3 \\ &= 0.0094 \end{aligned}$$

$$p_3' = \begin{bmatrix} -0.66995 & 0.17666 & 0.7052 & 0.1505 \end{bmatrix}$$

$$\begin{aligned} E_3 &= E_2 - \hat{t}_3 p_3' \\ &= \begin{bmatrix} 0.0454 & -0.0499 & 0.0475 & -0.0468 \end{bmatrix} \end{aligned}$$

For Y block:

$$q_3 = \begin{bmatrix} 1 \end{bmatrix}$$

$$\begin{aligned} F_3 &= \theta^1(t_1)q_1' + \theta^2(t_2)q_2' + \theta^3(t_3)q_3' \\ &= 1.0785 \\ &= y_{scaled} \end{aligned}$$

The y_{scaled} was rescaled to find the value in the original units as described in Eq. B.6.

$$y(OL) = 0.0502$$

The error between y_{actual} and y_{cal} was 0.0013 (0.0515 - 0.0502).

APPENDIX E

SAMPLES OF DATA SETS

This section showed samples of both simulated data sets and real plant data sets which were used in this work.

Table E.1: Samples of simulated data sets (100 samples from 1331 samples)

No.	OL	Top temp. (^oC)	High middle temp. (^oC)	Low middle temp. (^oC)	Bottom temp. (^oC)
1	5.85E-06	57.36777	66.11334	73.29539	90.15116
2	1.58E-05	57.36949	66.12693	73.64279	92.32539
3	3.00E-05	57.36877	66.14506	74.08925	91.83358
4	2.32E-05	57.36868	66.13616	73.88014	91.89798
5	4.90E-06	57.36886	66.11250	73.26111	90.55725
6	9.18E-03	57.55761	71.46389	80.08286	92.81331
7	4.83E-06	57.36776	66.11201	73.25827	89.78712
8	5.15E-02	58.41580	74.21035	80.65775	93.04961
9	1.34E-05	57.36823	66.12328	73.55976	91.31212
10	4.73E-06	57.36856	66.11216	73.25464	90.19817
11	2.36E-03	57.41739	68.48486	79.76305	92.77546
12	2.75E-02	57.93201	73.45165	80.35938	93.31935
13	7.93E-06	57.36809	66.11616	73.37027	90.82313
14	3.87E-06	57.36771	66.11074	73.22275	89.40206
15	1.99E-05	57.36918	66.13216	73.77807	92.08469
16	8.76E-02	59.13205	74.79194	81.06607	93.74957
17	1.71E-04	57.37091	66.32482	76.71009	91.95862
18	3.18E-02	58.01860	73.64619	80.41517	92.57845
19	9.07E-02	59.19290	74.83204	81.10006	93.71080
20	4.99E-03	57.47126	70.06048	79.96663	92.53632

No.	OL	Top temp. (^{o}C)	High middle temp. (^{o}C)	Low middle temp. (^{o}C)	Bottom temp. (^{o}C)
21	8.64E-02	59.10900	74.77628	81.05463	93.84386
22	2.61E-05	57.36885	66.14006	73.97209	92.06125
23	2.09E-05	57.36835	66.13308	73.80777	91.71068
24	2.46E-03	57.41932	68.56637	79.78121	92.49626
25	5.37E-03	57.47936	70.22787	79.98110	92.72977
26	9.93E-06	57.36842	66.11888	73.44084	91.19929
27	9.49E-05	57.37036	66.22891	75.60705	92.23262
28	1.29E-02	57.63397	72.18958	80.15098	92.53332
29	4.79E-02	58.34425	74.13052	80.61487	93.43903
30	5.10E-06	57.36880	66.11273	73.26825	90.49552
31	3.72E-05	57.36899	66.15446	74.30110	92.15857
32	5.24E-02	58.43384	74.22907	80.66779	93.30218
33	3.84E-02	58.15140	73.87724	80.49775	92.72275
34	1.11E-05	57.36826	66.12041	73.48339	91.37779
35	3.37E-02	58.05661	73.71845	80.43882	92.77672
36	2.07E-05	57.36854	66.13295	73.80269	91.76290
37	2.77E-02	57.93384	73.46010	80.36097	92.60182
38	1.28E-05	57.36778	66.12240	73.54049	91.20704
39	7.02E-06	57.36769	66.11483	73.33748	90.42721
40	3.69E-02	58.12079	73.82910	80.47871	92.74769
41	4.23E-02	58.22969	73.98921	80.54565	92.79577
42	2.49E-05	57.36939	66.13868	73.93453	92.29966
43	2.00E-05	57.36863	66.13204	73.77933	91.90808
44	5.63E-06	57.36784	66.11309	73.28753	90.20779
45	1.03E-02	57.58110	71.72513	80.10595	92.57360
46	4.61E-06	57.36842	66.11196	73.25022	90.04045
47	2.86E-04	57.37413	66.46882	77.66105	92.18893
48	1.87E-02	57.75183	72.86896	80.23907	92.35352
49	7.85E-06	57.36800	66.11602	73.36719	90.72741
50	4.30E-02	58.24499	74.00875	80.55526	93.03910

No.	OL	Top temp. (^{o}C)	High middle temp. (^{o}C)	Low middle temp. (^{o}C)	Bottom temp. (^{o}C)
51	9.37E-06	57.36841	66.11815	73.42129	91.18510
52	5.45E-05	57.37043	66.17730	74.76020	92.71553
53	8.95E-06	57.36841	66.11761	73.40660	91.18629
54	2.66E-02	57.91434	73.40746	80.34778	93.26173
55	4.80E-02	58.34487	74.13185	80.61509	93.13526
56	1.24E-02	57.62251	72.09780	80.14146	92.70268
57	3.34E-02	58.04972	73.70573	80.43475	92.66298
58	1.35E-05	57.36931	66.12389	73.56553	92.05534
59	8.42E-06	57.36929	66.11724	73.38810	91.66388
60	1.59E-05	57.36846	66.12660	73.64366	91.54910
61	5.58E-06	57.36826	66.11317	73.28571	90.25722
62	6.24E-06	57.36873	66.11420	73.30990	90.80659
63	2.40E-02	57.85898	73.25757	80.31179	92.52512
64	7.52E-06	57.36870	66.11585	73.35570	91.16891
65	7.09E-06	57.36936	66.11554	73.34067	91.34449
66	3.51E-02	58.08572	73.76911	80.45712	93.03637
67	4.78E-06	57.36851	66.11221	73.25647	90.14689
68	4.43E-06	57.36743	66.11137	73.24345	89.52721
69	6.95E-06	57.36938	66.11536	73.33554	91.38545
70	1.44E-03	57.39759	67.71044	79.53807	92.07584
71	3.96E-02	58.17474	73.91157	80.51254	92.73215
72	7.05E-06	57.36821	66.11506	73.33889	90.58107
73	2.50E-02	57.88028	73.31867	80.32587	92.69634
74	1.28E-02	57.63198	72.17490	80.14966	92.41703
75	5.25E-02	58.43563	74.23102	80.66925	93.18552

No.	OL	Top temp. (^{o}C)	High middle temp. (^{o}C)	Low middle temp. (^{o}C)	Bottom temp. (^{o}C)
76	6.06E-02	58.59785	74.38731	80.76441	93.06705
77	3.41E-03	57.43890	69.21408	79.87772	92.57761
78	6.65E-03	57.50573	70.72851	80.02317	92.64626
79	1.44E-02	57.66265	72.39283	80.17382	92.29428
80	2.41E-05	57.36940	66.13768	73.91145	92.35520
81	9.10E-02	59.20028	74.83637	81.10475	93.89758
82	2.91E-02	57.96293	73.52931	80.37955	92.48253
83	1.24E-05	57.36925	66.12244	73.52798	91.96985
84	3.88E-06	57.36762	66.11072	73.22322	89.33564
85	5.69E-06	57.36898	66.11358	73.29008	90.74221
86	5.61E-02	58.50838	74.30360	80.71206	93.43829
87	7.44E-02	58.87256	74.61027	80.92176	93.37129
88	6.18E-06	57.36815	66.11391	73.30753	90.51550
89	5.39E-02	58.46306	74.25887	80.68565	93.31452
90	3.96E-05	57.36978	66.15787	74.36835	92.48134
91	3.62E-05	57.36898	66.15318	74.26895	92.17190
92	5.13E-06	57.36894	66.11283	73.26948	90.68106
93	4.41E-06	57.36752	66.11137	73.24269	89.58917
94	4.65E-02	58.31479	74.09728	80.59749	92.77095
95	3.73E-02	58.12978	73.84135	80.48477	93.34874
96	6.10E-06	57.36788	66.11371	73.30454	90.25288
97	1.63E-02	57.70412	72.63127	80.20408	92.97686
98	1.35E-05	57.36922	66.12379	73.56398	91.88444
99	7.60E-06	57.36902	66.11607	73.35884	91.21564
100	5.71E-06	57.36866	66.11348	73.29037	90.71493

Remark: OL represented the mass fraction of cyclohexanol.

Table E.2: Samples of real plant data sets (100 samples from 451 samples)

No.	OL	Top temp. (^oC)	High middle temp. (^oC)	Low middle temp. (^oC)	Bottom temp. (^oC)
1	2.37E-04	57.77376	67.85109	78.60389	92.50938
2	2.60E-04	57.68524	67.71702	78.74191	92.62955
3	2.94E-04	57.68524	67.57255	78.91922	92.72490
4	3.07E-04	57.66616	67.28355	78.61263	92.49154
5	3.72E-04	57.52208	66.88672	78.54277	92.43651
6	2.77E-04	57.50127	66.86496	78.56720	92.56674
7	3.10E-04	57.50474	66.57689	78.32095	92.20082
8	3.99E-04	57.46656	66.66566	78.48686	92.37913
9	3.09E-04	57.56895	66.90587	78.54102	92.60396
10	3.21E-04	57.52208	66.73875	78.35411	92.27912
11	3.31E-04	57.60887	66.88322	78.65977	92.65978
12	3.84E-04	57.50301	66.68480	78.66501	92.49464
13	5.44E-04	57.51861	66.86234	78.92358	92.74352
14	2.76E-04	57.50474	66.76311	78.48686	92.51247
15	3.43E-04	57.46656	66.58038	78.52530	92.38068
16	4.36E-04	57.58718	66.81709	78.87903	92.44813
17	3.33E-04	57.66790	66.91109	78.93755	92.79933
18	3.54E-04	57.48393	66.65349	78.74541	92.46906
19	3.26E-04	57.52729	66.73875	78.63184	92.49232
20	3.34E-04	57.61927	66.69527	78.72616	92.57372
21	3.59E-04	57.64185	66.78227	78.82924	92.55511
22	3.39E-04	57.57068	66.77181	78.76463	92.43495
23	3.25E-04	57.63924	66.84492	78.78121	92.60396
24	3.31E-04	57.50474	66.70396	78.78296	92.43884
25	5.90E-05	57.66963	66.74847	78.82570	92.58706

No.	OL	Top temp. (^{o}C)	High middle temp. (^{o}C)	Low middle temp. (^{o}C)	Bottom temp. (^{o}C)
26	4.29E-04	57.66963	66.75569	78.83778	92.63208
27	3.18E-04	57.66963	66.76291	78.84986	92.67709
28	3.72E-04	57.66963	66.77013	78.86194	92.72211
29	3.00E-04	57.66963	66.77735	78.87402	92.76714
30	3.55E-04	57.66963	66.78457	78.88609	92.81216
31	3.44E-04	57.66963	66.79179	78.89817	92.85718
32	4.32E-04	57.66963	66.79900	78.91025	92.90220
33	3.97E-04	57.50648	66.64653	78.81526	92.77141
34	4.13E-04	57.62794	67.08860	79.23102	92.94120
35	3.81E-04	57.58805	66.70309	78.92358	92.77141
36	3.72E-04	57.56200	66.72307	78.79604	92.82491
37	3.82E-04	57.52729	66.62563	78.76113	92.66133
38	4.09E-04	57.52555	66.73265	78.81439	92.93422
39	4.49E-04	57.50474	66.69788	78.79779	92.82491
40	4.66E-04	57.52208	66.53860	78.63010	92.72025
41	4.31E-04	57.50474	66.59953	78.83273	92.85282
42	4.19E-04	57.66616	66.98243	79.32714	93.19936
43	4.33E-04	57.57938	66.66219	78.75589	92.77606
44	4.45E-04	57.52035	66.59779	78.89912	92.82724
45	4.04E-04	57.52035	66.48817	79.02838	92.74118
46	3.42E-04	57.73384	67.00854	79.45468	92.94817
47	3.58E-04	57.62794	66.90238	79.30268	92.77141
48	3.51E-04	57.68871	66.88670	79.19608	92.77141
49	4.13E-04	57.48393	66.66566	79.33762	92.72025
50	4.34E-04	57.52295	66.63435	79.22927	92.74118

No.	OL	Top temp. (^{o}C)	High middle temp. (^{o}C)	Low middle temp. (^{o}C)	Bottom temp. (^{o}C)
51	4.17E-04	57.50301	66.60824	79.28345	92.78381
52	4.34E-04	57.50127	66.63522	79.32365	92.80863
53	2.90E-04	57.52555	66.60127	79.08953	92.76289
54	3.49E-04	57.48393	66.70744	79.29045	92.93732
55	1.91E-04	57.52208	66.32802	78.67025	92.06747
56	2.95E-04	57.57068	66.50035	77.87470	92.86755
57	3.46E-04	57.62448	66.57689	77.86859	93.02725
58	4.02E-04	57.52208	66.61868	78.05547	92.92104
59	2.87E-04	57.52555	66.46901	77.53507	93.08772
60	2.68E-04	57.52729	66.47075	77.42419	93.14355
61	3.61E-04	57.52208	66.47249	77.74635	92.96678
62	3.29E-04	57.46656	66.43245	77.57697	92.98616
63	2.88E-04	57.50387	66.53860	77.59619	93.14355
64	3.30E-04	57.48046	66.45161	77.68174	92.97608
65	3.17E-04	57.48046	66.45770	77.72539	93.06602
66	3.50E-04	57.46829	66.34718	77.68698	92.97608
67	3.83E-04	57.52555	66.45509	77.86859	93.16216
68	3.82E-04	57.52382	66.26714	77.72364	92.93732
69	3.88E-04	57.59931	66.30540	77.78912	93.03191
70	3.69E-04	57.51428	66.45074	77.90353	93.25519
71	4.93E-04	57.51861	66.48643	78.28775	93.11331
72	4.02E-04	57.48044	66.34370	77.93146	93.03423
73	4.07E-04	57.48046	66.25670	77.86859	92.91872
74	3.92E-04	57.52382	66.30540	77.86334	92.92027
75	2.87E-04	57.46482	66.43245	77.78476	93.00167

No.	OL	Top temp. (^{o}C)	High middle temp. (^{o}C)	Low middle temp. (^{o}C)	Bottom temp. (^{o}C)
76	3.18E-04	57.48219	66.45509	77.85287	93.03191
77	4.20E-04	57.53946	66.57341	78.17947	93.03191
78	4.17E-04	57.52035	66.46988	78.14105	93.08927
79	2.46E-04	57.48219	66.35616	77.73106	92.63279
80	3.24E-04	57.48219	66.41837	77.82468	92.83598
81	5.57E-04	57.40319	66.47250	78.27115	93.05749
82	4.24E-04	57.48391	66.28978	78.15852	93.03191
83	4.43E-04	57.51861	66.48643	78.23187	93.14355
84	4.88E-04	57.43880	66.42375	78.15678	93.13425
85	4.53E-04	57.50214	66.52815	78.43448	93.25053
86	4.39E-04	57.46309	66.43071	78.21440	93.03191
87	4.53E-04	57.50127	66.48817	78.24758	93.06214
88	4.80E-04	57.52295	66.47075	78.37333	93.14433
89	4.55E-04	57.44401	66.32628	78.18297	93.02725
90	5.53E-04	57.50474	66.63435	78.54452	93.19936
91	5.35E-04	57.50127	66.36458	78.51831	92.80941
92	4.23E-04	57.46135	66.42375	78.28950	92.97608
93	4.20E-04	57.49347	66.43245	78.27115	93.02725
94	4.25E-04	57.56200	66.61171	78.51831	93.02958
95	4.00E-04	57.48044	66.51076	78.39779	92.97608
96	5.00E-04	57.50474	66.47510	78.50433	92.92492
97	4.29E-04	57.51861	66.36284	78.30086	92.92027
98	4.55E-04	57.46482	66.47771	78.38905	93.00632
99	5.21E-04	57.50127	66.47249	78.56720	93.03191
100	4.32E-04	57.44574	66.29932	78.23537	92.64583

Remark: OL represented the mass fraction of cyclohexanol.

VITA

Mr.Praphon Kemachuntree was born in Bangkok, Thailand, on November 11, 1981. He received a Bachelor Degree of Science in Department of Chemical Technology from Chulalongkorn University, Bangkok in 2004. In the same year, he studied for Master Degree of engineering in Chemical Engineering, Chulalongkorn University in May 2004.