ESTIMATION OF VINYL ACETATE MONOMER CONCENTRATION USING NEURAL NETWORK & PARTIAL LEAST SQUARE

MOHAMAD ZAFARUDIN BIN MOHAMAD

A thesis submitted in fulfillment of the requirements for the award of the degree
of Bachelor of Chemical Engineering

Faculty of Chemical & Natural Resources Engineering
University Malaysia Pahang

MAY 2008

I declare that this thesis entitled "*Estimation of Vinyl Acetate Monomer Concentration Using Neural Network & Partial Least Square*" is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : .....

Name of Candidate : Mohamad Zafarudin bin Mohamad

Date : 15 May 2008

DEDICATION

Special dedication to the memory of my beloved mother and father, Zaiton bt Daud and Mohamad bin Taib, my aunty and uncle and all my family members that always inspire, love and stand besides me, my supervisors, my beloved friends, my fellow colleagues, and all faculty members

For all your love, care, support, and believe in me.Thank you so much.

ACKNOWLEDGEMENT

Praise be to God for His help and guidance that finally I am able to complete this final year project as one of my requirement to complete my study.

First and foremost I would like to extend my deepest gratitude to all the parties involved in this research. First of all, a special thank to my supervisor Mr. Noor Asma Fazli bin Abdul Samad his willingness in overseeing the progress of my research work from its initial phases till the completion of it. I do believe that all their advice and comments are for the benefit of producing the best research work.

Secondly, I would like to extend my words of appreciation to all my lecturers in the Faculty of Chemical Engineering and Natural Resources (FKKSA), for their support and motivation during this project development.

To all my close friends and all my course mates, thank you for believing in me and helping me to go through the difficult time. The experiences and knowledge I gained throughout the process of completing this final project would prove invaluable to better equip me for the challenges which lie ahead. Last but definitely not least to my family members, I can never thank you enough for your love, and for supporting me throughout my studies in University Malaysia Pahang (UMP).

ABSTRACT

This thesis is about the application of Artificial Neural Network (ANN) and Partial Least Square (PLS) on the chemical process plant. At the present time, the process and development in chemical plants are getting more complex and hard to measure. Therefore, the needs for a system that can help to supervise and control the process in the plant have to be accomplished in order to achieve higher performance and quality. As the emergence of Artificial Neural Network and Partial Least Square application nowadays to solve problem in various fields had given a great significant effect such as soft-sensor, lack of on-line measurement, and incorporate the safety issues while maintaining practicality and economic feasibility, both of the system are reliable to be adapted in the chemical plant. Furthermore, this thesis will be focusing more on the application of Artificial Neural Network and Partial Least Square as a estimation scheme in the chemical plant. Estimation by using Artificial Neural Network (ANN) and Partial Least Square (PLS) is popular in the present time as a mechanism to estimate the variables in the chemical plant. By implementing such system, the performance and quality of the plant will increased. For this thesis, the vinyl acetate monomer plant had been chosen as the case study to provide the necessary data and information to run the research. Vinyl acetate monomer process will provides a dependable source of data and an appropriate test for alternative control and optimization strategies for continuous chemical processes.

ABSTRAK

Tesis ini adalah berkenaan aplikasi Rangkaian Saraf Buatan *(Artificial Neural Network)* dan *Partial Least Square (PLS) pada* kilang pemprosesan kimia. Pada masa kini, proses dan pembangunan dalam kilang kimia telah menjadi semakin kompleks dan susah untuk diukur. Oleh itu, satu sistem yang dapat menyelia dan mengawal proses di kilang perlu diadakan untuk mencapai prestasi dan kualiti yang lebih baik. Peningkatan penggunaan Rangkaian Saraf Buatan dan *Partial Least Square (PLS)* untuk menyelesaikan masalah di pelbagai lapangan di zaman ini telah memberi kesan yang positif seperti yg dilakukan pada pengesan lembut, kelemahan dalam pengukuran secara 'on-line', dan penggabungan isu keselamatan di mana kebolehsanaan ekonomi dan praktikalnya dikekalkan, yang mana kedua-dua sistem ini sangat sesuai untuk diadaptasikan di dalam kilang kimia. Selain itu, tesis ini juga akan lebih memfokuskan pengaplikasian Rangkaian Saraf Buatan dan *Partial Least Square (PLS)* sebagai satu skim anggaran di dalam kilang kimia. Penganggaran menggunakan Rangkain Saraf Buatan *(Artificial Neural Network)* dan *Partial Least Square (PLS)* adalah sangat popular pada masa sekarang sebagai satu mekanisme untuk mengukur pembolehubah-pemboleubah yg terdapat di dalam kilang kimia. Dengan pengaplikasian sistem ini, tahap produktiviti dan prestasi di kilang akan bertambah. Dalam tesis ini, kilang Vinyl Acetate Monomer telah dijadikan sebagai rujukan untuk mendapatkan maklumat dan data yang diperlukan untuk menjalankan kajian. Proses asetat vinil akan memberikan sumber data yang tepat dan merupakan ujikaji yang sesuai untuk mengadakan kajian berkenaan pengawalan alternatif dan strategi pengoptimumtasi untuk proses kimia yang berterusan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ANN -	Artificial neural networks
CC -	Composition controller
CSTR -	Continuous stir tank reactor
DNNPLS -	Dynamic neural network partial least squares
EKF -	Extended Kalman filter
FFN -	Feed forward networks
FPM -	First principle method
LC -	Level controller
LCC -	Light-cut column
LM -	Levenberg-Marquardt method
MLR -	Multiple linear regressions
MSE -	Mean squared error of prediction
NIPALS -	Non-linear iterative partial least squares
NNPLS -	Neural network partial least squares
PIC -	Pressure indicator
PI -	Proportional-Integral
PID -	Proportional-Integral-Derivative
PLS -	Partial least squares regression
QPLS -	Quadratic partial least squares
RBFPLS -	Radial basis function partial least squares

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

Stringent product specification, stiff competition among manufacturers and increasingly strict regulation from local authority in the face of full capacity operation with zero accidents and emission have forced many existing plants to revamp their existing control system. More advanced control schemes have been implemented.

Despite these successful implementations, many issues remained as hindrances to efficient process control. For example, the success in the implementation of any optimization scheme requires adequate performance of all control loops. This is however, sometimes hampered by two issues. The first is related to inadequacy of conventional controller used since chemical process dynamics are typically non-linear whilst the controllers are based on linear theory. The second issue is associated with process measurement, the accuracy of which is a prerequisite to successful process control.

Since measurement devices are not one of the main factors in achieving effective process control, selection of appropriate sensors and their location should be properly consider. However, not all variables in a process plant are readily to be measured on-line. Product quality variables such as chemical concentration and their composition are rarely available on-line, and are usually obtained by laboratory

sample analysis. This is usually performed at long intervals and is therefore not practical to be used for process control.

Over the years, various on-line measurement devices have been developed. However, many of these on-line devices are still suffering from problems due to the availability, reliability, complexity and large delays. For some quality variables, existing analytical tools used are simply unavailable for on-line applications. Hence, the development of inferential estimation and control has been advocated as one of the alternative solution to deal with measurement difficulties.

1.1.1 Background of study

For several years, researchers in plant wide design, optimization and control areas have expressed interest in getting realistic test problems for assessing new technologies. The most frequently studied test problem so far is the Tennessee Eastman Challenge (TEC) process which has been utilized by many different researchers in studies ranging from plant wide control to optimization to fault detection.

In 1998, an additional model of a large, industrially relevant system, a vinyl acetate monomer (VAC) manufacturing process, was published by Luyben and Tyreus. The VAC process contains several standard unit operations that are typical of many chemical plants. Both gas and liquid recycle streams are present as well as process-to-process heat integration.

Luyben and Tyreus presented a plant wide control test problem based on the VAC process. The VAC process was modeled in TMODS, which is a proprietary DuPont in-house simulation environment, and thus, it is not available for public use. This research presents a first-principle nonlinear dynamic model for the VAC process as well as associated software, based on the design details provided in references.

The model of the VAC process is developed in MATLAB, and both the steady state and dynamic behavior of the MATLAB model are designed to be close to the TMODS model. Since the MATLAB model does not depend on commercial simulation software and the source code is open to public, the model can be modified for use in a wide variety of process control research areas. To obtain a reasonable simulation speed, compiled model written in C is also available to researchers. Details on how the various programs can be obtained are given at the end of this paper.

In this research, design details of the MATLAB model are presented. For each unit, design assumptions, physical data, and modeling formulations are discussed. There are some differences between the TMODS model and the MATLAB model, and these differences together with the reasons for them are pointed out. Steady state values of the manipulated variables and major measurements in the base operation are given. Production objectives, process constraints, and process variability are summarized based on the earlier publication. All of the physical property, kinetic data, and process flow sheet information in the MATLAB model come from sources in the open literature.

1.2 Problem Statement

The advancement of technology brings new challenges to the chemical industry especially in petrochemical sector. Due to the global nature of the chemical market, any addition of new industry installation throughout the world cause challenges to the existing ones. Additionally, the rapid growth of chemical industry worldwide influences the dynamics of the chemical business. Product quality specifications have been increasingly more difficult to satisfy. The need for reduced variability products has been widespread. Apart from higher demand or quality, the environment and safety regulation imposed on this sector are becoming more stringent. Due to this reason, great emphasis should be put on the method of process control system. It should be incorporate these safety issues while maintaining practicality and economic feasibility. The objective of the process control is to

estimate vinyl acetate monomer concentration by develop estimator using neural network and partial least square in order to achieve the operating requirements of the plant in an optimal manner.

Lack of on-line measurement is an another issue. Efficient Vinyl Acetate Monomer Process is often hampered by the difficulty in measuring some of the key component because of the lack o robust on-line sensors. The inability to provide on-line measurement of the process variable such as product concentration has proved to be a significant obstacle for the implementation of advanced control and optimizations solution. The availability of such measurement is important for establishing optimum operation and minimizing product quality variability. Although off-line measurement via laboratory analysis can be used to provide delayed measurement but sometimes a little bit too late to be useful especially for process control. This is perhaps the main motivation behind the use of various forms of soft-sensor technology.

Soft-sensor is founded on the assumption that data and theoretical information can be use to formulate a model that can represent the measurement of difficult to measure variables. Synonymous to method based on-line estimation; soft-sensors are useful in the process since several key variables such as product, input and output concentration in fact difficult to measure. Several techniques have been reported but much works are still needed especially in making use of available control technology which has somewhat reached certain level of maturity when dealing with continuous chemical processes.

1.3 Objective

The aim of this study is to develop inferential estimator using Neural Network (NN) & Partial Least Square (PLS) in order to measure the concentration of Vinyl Acetate Monomer Process .

1.4 Scope

To achieve the above objectives, the following research scopes had been identified:

i) To simulate Vinyl Acetate Monomer Process as a case study.

ii) Analysis of dynamic response of the process.

iii) To develop base-case control for Vinyl Acetate Monomer Process

iv) To develop of inferential estimator using Partial Least Square (PLS).

v) To develop inferential estimation using Neural Network (NN).

vi) Model Testing

CHAPTER 2

LITERATURE REVIEW

2.1 Artificial Neural Networks (ANNs) Background

Early works in the field of neural networks centred on modelling the behaviour of neurons found in the human brain. Engineering systems are considerably less complex than the brain, hence from an engineering viewpoint ANN can be viewed as nonlinear empirical models that are especially useful in representing input-output data, making predictions in time, classifying data, and recognising patterns.

Despite the above mentioned capabilities, ANNs are not a solution for all modelling problems. Therefore, it is necessary to understand the advantages and disadvantages of ANN in contrast with first principle models or other empirical models to determine their applicability for a particular problem. ANN has several advantages as described by Baughman and Liu (1995):

i) Distribution of information over a field of nodes. This feature allows greater flexibility and robustness of ANN because a slight error or failure in certain sections of the network will not affect the entire system.
ii) Learning ability of ANN. ANN is able to adjust its parameters in order to adapt itself to changes in the surrounding systems by using an error-correction training algorithm.
iii) Extensive knowledge indexing. ANN is also able to store a large amount of information and access it easily when needed. Knowledge is kept in the

network through the connection between nodes and the weights of every connection.

iv) Imitation of the human learning process. The network can be trained iteratively, and by tuning the strengths of the parameters based on observed results. The network can develop its own knowledge base and determine cause and effect relations after repeated training and adjustments.

v) Potential for on-line use. Once trained, ANN can yield results from a given input relatively quickly, which is a desired feature for the on-line use.

Some of the limitation of ANN summarized by Baughman and Liu (1995):

i) Long training time. Training time for ANN can take too much time especially for large networks.

ii) Requires large amount of data. ANN needs large amount of input-output data for a better generalization. Therefore, if there is only a small amount of input-output data available, ANN may not be suitable for modeling the system.

iii) No guarantee to optimal results and reliability. Although the network contains parameters that can be tuned by the training algorithm, there is no guarantee that the resulting model is perfect for the system. The tuned model may be accurate in one region but inaccurate in another

iv) Difficulty in selecting good sets of input variables. Selection of input variables is difficult because too many input variables or wrongly selected input variableswill cause over fitting and poor generalization. Too little or inappropriate input variables will lead to poor mapping of the system.

2.1.1 Neuron (Node) and Neural Networks

Figure 2.1 shows the basic structure of a single processing unit in an ANN which will be referred to as a *node* in this work and is a simplified mimic of a single neuron in the human brain. A node receives one or more input signals, ui, which may come from other nodes or from some other source. Each input is weighted according to the value wi,j that is called *weight*. These weights are similar to the synaptic strength between two connected neurons in the human brain. The weighted signals to

the node are summed and the resulting signal, called the *activation, h,* is sent to the *transfer function, g,* which can be any type of mathematical function, but is usually taken to be a simple bounded differentiable function such as the sigmoid. The resulting output of the node yi, may then be sent to one or more nodes as an input or taken as the output of a ANN model.

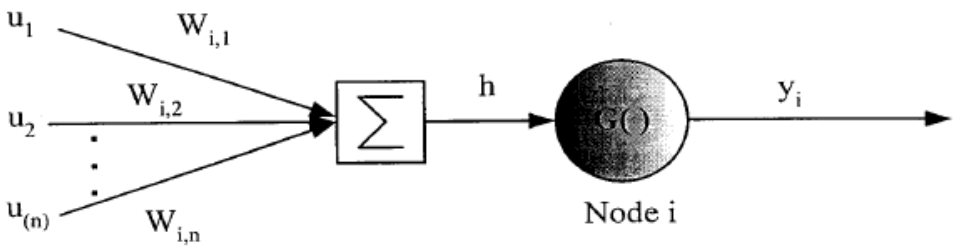

Figure 2.1: Structure of single processing node (Basheer and Hajmeer, 2000)

Neural network consists of interconnected neurons arranged into several layers. A group of nodes called the input layer receives a signal from some external source. In general, this input layer dos not process signal unless it needs scaling. Another group of nodes, called the output layer, return signals to the external environment. The remaining nodes in the network are called hidden nodes because they do not receive any signals from or send a signal to an external source or location. The hidden nodes may be grouped into one or more hidden layers depending on the architecture of the network. Each of the connection between two nodes has a weight associated with it. Figure 2.2 shows a network with fully connected layers. The connections are in forward direction and are known as a multilayer feed forward neural network.

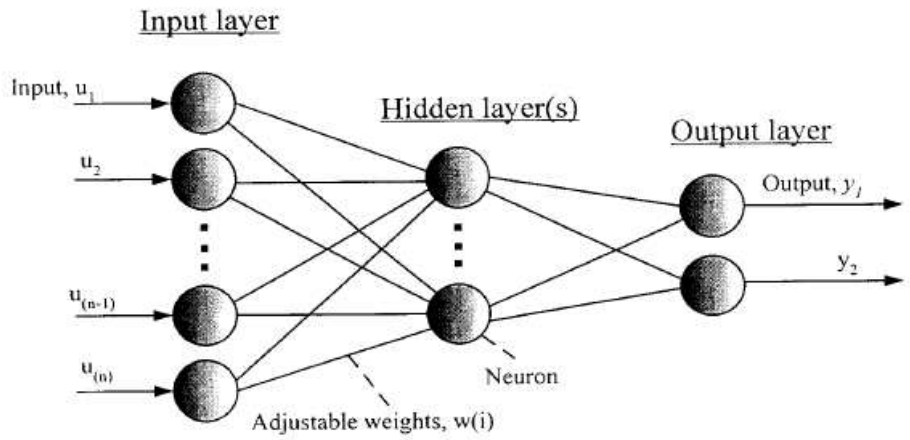

Figure 2.2: Structure of a layered neural network (Basheer and Hajmeer, 2000)

In the principle, connections between nodes can be in any direction for nodes in nonadjacent layers or within the same layer. Feedback connections from a node in one layer to a node in a previous layer can also be made. When feedback connections are involved, the network is referred to as recurrent networks. Due to the complexity of the network, analytical method of calculating the values of the weights for a particular network to represent process behaviour is not discouraging. Instead the network must be *trained* on a set of data (called the *training set*) collected from the process to be modelled.

Training is a procedure of estimating the values of the weights and establishing the network structure, and the algorithm used to do this is called a "learning" algorithm. The learning algorithm is essentially an optimisation algorithm. Once a network is trained, it can be conveniently used as a model to represent the system for various different purposes.

A key difficulty with optimisation for neural network problems is that multiple minima occur especially in large networks. Since most training procedures used for neural networks typically find local minima starting from randomly selected guesses for the parameters, it should be expected that local minima of varying quality will be found. While use of a global optimisation procedure, such as genetic algorithms, branch and bound, or simulated annealing might thus appear to be called for, the training time for such algorithms oftentimes expands to an unacceptable degree.

This practically limits their application. Regardless of what training algorithm is used to calculate the values of the weights, all of the training methods go through the same general steps. First, the available data is divided into a training and test set(s). The following procedure is then used to determine the values of weights of the network:

i) For a given ANN architecture, the values of the weights in the network are initialized as small random numbers;
ii) The inputs of the training set are sent to the network and the resulting outputs are calculated;
iii) Some measure (an objective function) of the error between the outputs of the network and the known correct (target) values are calculated;
iv) The gradient of the objective function with respect to each of the individual weights are calculated;
v) The weights are changed according to the optimization search direction and step
length determined by the optimisation code;
vi) The procedure returns to step 2;
vi) The iteration terminates when the value of the objective function calculated using the data in the test set starts to increase.

As mentioned above, the available data is divided into the training and test set. The purpose of partitioning the available data into the training and test set is to evaluate how well the network *generalises* (predicts) to domains that were not included in the training set. For non-trivial problems it is often difficult to collect all of the possible input-output patterns needed to span the input-output space for a particular behaviour or process to be modelled.

Therefore, the network should be trained with some subset of all the possible input-output patterns. In addition, the training set must also be representative of the domain of interest. If not, the net may not predict well for similar data, and may predict poorly for completely novel data (extrapolate). The test set is used to evaluate how well the neural network generalises using the weights computed during the training exercise. Since it is the ability of the network to predict ‘unseen’ data serves

as a measure of model capability, the performance on the test set is used to select the optimal set of weights as well as network topology.

2.1.2 Feedforward Neural Networks

Two layer (sometimes called three layers) feedforward ANN are commonly encountered models in the literature. Computation nodes are arranged in layers and information feeds forward from layer to layer via weighted connections as illustrated in the figure 2.3. Here, circle represent computation nodes (transfer functions), and lines represent weighted connections. The basis thresholding nodes are represented by squares.

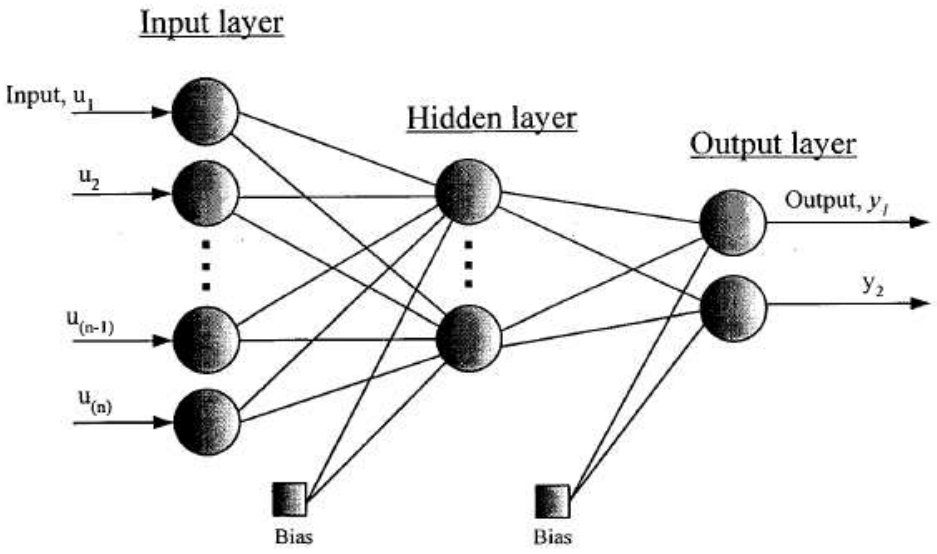

Figure 2.3: Graph of the information flow in a feedforward neural network (Demuth and Beale, 2000)

Mathematically, the typical feedforward network can be expressed as

$$y_i = \varphi_{\circ}\left[C\varphi_h\left(Bu_i + b_h\right) + b_{\circ} \right] \tag{2.1}$$

where y_i is the output vector corresponding to input vector u_i, C is the connection matrix (matrix of weights) represented by arcs (the lines between two nodes) from

the hidden layer to the output layer, *B* is the connection matrix from the input layer to the hidden layer, and b_h and $b_\circ$ are the bias vector for the hidden and output layer, respectively. $\varphi_h(\cdot)$ and $\varphi_\circ(\cdot)$ are the vector valued functions corresponding to the activation (transfer) functions of the nodes in the hidden and output layers, respectively. Thus, feedforward neural network models have the general structure of

$$y_i = f(u) \tag{2.2}$$

where $f(\cdot)$ is a nonlinear mapping. Hence feedforward neural networks are structurally similar to nonlinear regression models, and Eq.(2.2) represents a steady state process. To use models for identification of dynamic systems or prediction of time series, a vector comprised of a moving window of past input values (*delayed coordinates*) must be introduced as inputs to the net. This procedure yields a model analogous to a nonlinear finite impulse response model where

$$y_i = y_t \text{ and } u_i = [u_t, u_{t-1}, ..., u_{t-m}] \text{ or } y_t = ([u_t, u_{t-1}, ..., u_{t-m}]) \tag{2.3}$$

The lengths of the moving window must be long enough to capture the system dynamics for each variable in practice. In practice, the duration of the data windows are determined by trial and error (cross validation) and each individual input and output variable might have a separate data window for optimal performance.

Backpropagation learning algorithm is one of the earliest and the most common method for training multilayer feedforward neural networks. Development of this learning algorithm was one of the main reasons for renewed interest in this area and this learning rule has become central to many current works on learning in ANN. It is used to train nonlinear, multilayered networks to successfully solve difficult and diverse problems.

Standard backpropagation is a gradient descent algorithm in which the network weights are moved along the negative of the gradient of the performance function. The term backpropagation refers to the manner in which the gradient is

computed for nonlinear multilayer networks. There are a number of variations on the basic algorithm that are based on other standard optimisation techniques. These techniques will be discussed below (Demuth and Beale, 2000):

i) Gradient Descent with Momentum Backpropagation Known as traingdm in MATLAB Toolbox. This batch algorithm for feedforward networks provides faster convergence. Momentum allows a network to respond not only to the local gradient, but also to recent trends in the error surface. Acting like a low-pass filter, momentum allows the network to ignore small features in the error surface. Without momentum a network may get stuck in a shallow local minimum. With momentum a network can slide through such a minimum.

ii) Gradient Descent with Momentum and Adaptive Learning Backpropagation is also known as traingdx in MATLAB Toolbox. The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step sizes as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface.

iii) Levenberg-Marquardt Backpropagation is known as trainlm in MATLAB Toolbox, actually a hybrid of the Gauss-Newton Nonlinear Regression method and Steepest Gradient Descent method. Gauss- Newton method is slow converging while Steepest Gradient Descent method suffered local minimum problem. Lavenberg-Marquardt method was designed in such a way that it can choose wisely the direction by crossing between these two methods. This algorithm appears to be the fastest method for training moderatesized feedforward neural networks (up to several hundred weights).

These algorithms are used in this research to train feedforward neural networks and their performances in term of mean squared error will be compared.

2.1.3 Recurrent Neural Networks

Recurrent neural network (RNN) have architectures similar to standard feedforward neural networks with layers of nodes connected via weighed feedforward connections, but also include time delayed feedback or recurrent connections in the network architecture as shown in figure 2.4.

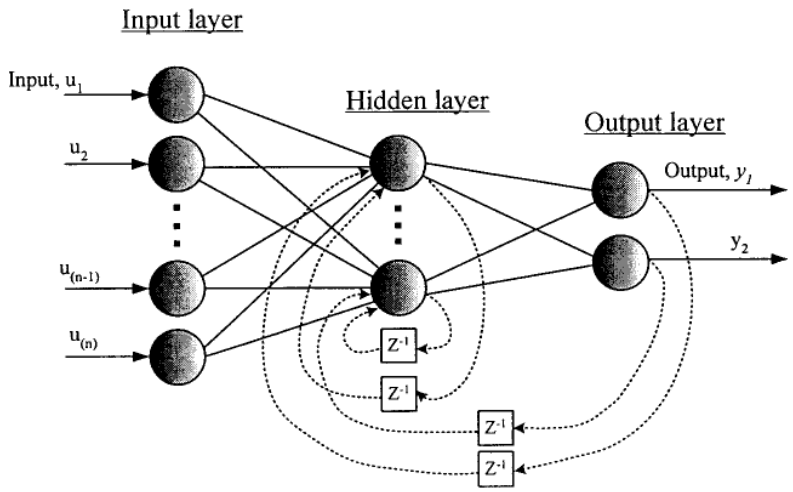

Figure 2.4: Representation of internally/externally recurrent neural networks (Elman,1990)

Figure 2.4 shows a representation of internally/externally recurrent neural networks. Circles represent computation nodes, lines represent weighted connections, z^{-1} indicates time delay. For clarity not all recurrent connection are shown and bias nodes are omitted. Two individual variations of RNN architectures are commonly employed. The first is called internally recurrent network (IRN) that is characterized by time delayed feedback connections from the output of hidden nodes to its input. This feedback path allows IRN to learn to recognize and generate temporal patterns, as well as spatial patterns. The remainder of the network comprises standard feedforward architecture. This structure also known as an Elman network (Elman, 1990). Externally recurrent networks (ERN), on the other hand, contain time delayed feedback connections from the output layer to the hidden layer.

Although the ERN and IRN can exhibit comparable modeling performance, they have different features that make one more desirable that the other for a particular process. Just like the conventional linear state space model, the IRN does not have any structural limit on the number of model states because the number of hidden nodes can be freely varied. The ERN, however, can only have the same number of states as model outputs because the outputs are the states. The IRN thus tends to be more flexible in modeling.

2.1.4 Neural Learning

In neural network, learning refers to the method of modifying the weights of connections between the nodes of a specific network. The training session of the neural network uses the error in the output values to update the weights connecting layers, until the accuracy is within the tolerance level. The training time for a feed forward neural networks using one of the variations of backpropagation can be substantial. For a simple two-input two-output system with 50 training samples, 100 000 training iterations are common. For large-scale systems, memory and computation time required for training a neural network can exceed hardware limits. This has been a bottleneck in developing fault diagnosis algorithms for industrial applications. Like other data-driven methods, the performance of neural networks is determined by the available data. It is highly possible that neural networks will generate unpredictable output when presented with an input out of the range of the training data. This suggests that the neural networks need to be retrained when there is a slight change of the normal operation conditions.

2.1.5 Applications of Artificial Neural Network

The utility of artificial neural network models lies in the fact that they can be used to infer a function from observations. This is particularly useful in applications where the complexity of the data or task makes the design of such a function by hand impractical. The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

i) Function approximation, or regression analysis, including time series prediction and modeling.

ii) Classification, including pattern and sequence recognition, novelty detection and sequential decision making.

iii) Data processing, including filtering, clustering, blind signal separation and compression.

Application areas include system identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition and more), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering.

2.2 Partial Least Square

Partial Least Squares (PLS) is a wide class of methods for modeling relations between sets of observed variables by means of latent variables. It comprises of regression and classification tasks as well as dimension reduction techniques and modeling tools. The underlying assumption of all PLS methods are that the observed data is generated by a system or process which is driven by a small number of latent (not directly observed or measured) variables. Projections of the observed data to its latent structure by means of PLS were developed by Herman Wold and coworkers.

PLS has received a great amount of attention in the field of chemometrics. The algorithm has become a standard tool for processing a wide spectrum of chemical data problems. The success of PLS in chemometrics resulted in a lot of applications in other scientific areas including bioinformatics, food research, medicine, pharmacology, social sciences, physiology–to name but a few.

This chapter introduces the main concepts of PLS and provides an overview of its application to different data analysis problems. In its general form PLS creates orthogonal score vectors (also called latent vectors or components) by maximising the covariance between different sets of variables. PLS dealing with two blocks of variables is considered in this chapter, although the PLS extensions to model relations among a higher number of sets exist. PLS is similar to Canonical Correlation Analysis (CCA) where latent vectors with maximal correlation are extracted. There are different PLS techniques to extract latent vectors, and each of them gives rise to a variant of PLS. PLS can be naturally extended to regression problems.

The predictor and predicted (response) variables are each considered as a block of variables. PLS then extracts the score vectors which serve as a new predictor representation and regresses the response variables on these new predictors. The natural asymmetry between predictor and response variables is reflected in the way in which score vectors are computed. This variant is known under the names of PLS1 (one response variable) and PLS2 (at least two response variables).

PLS regression used to be overlooked by statisticians and is still considered rather an algorithm than a rigorous statistical model. Yet within the last years, interest in the statistical properties of PLS has risen. PLS has been related to other regression methods like Principal Component Regression (PCR) and Ridge Regression (RR) and all these methods can be cast under a unifying approach called continuum regression.

The effectiveness of PLS has been studied theoretically in terms of its variance and its shrinkage properties. The performance of PLS is investigated in several simulation studies. PLS can also be applied to classification problems by encoding the class membership in an appropriate indicator matrix. There is a close connection of PLS for classification to Fisher Discriminant Analysis (FDA). PLS can be applied as a discrimination tool and dimension reduction method–similar to Principal Component Analysis (PCA). After relevant latent vectors are extracted, an

appropriate classifier can be applied. The combination of PLS with Support Vector Machines (SVM) has been studied in.

Finally, the powerful machinery of kernel-based learning can be applied to PLS. Kernel methods are an elegant way of extending linear data analysis tools to nonlinear problems.

2.2.1 Structure of PLS model

A PLS model consists of outer relations and an inner relation. The outer relations are matrices of independent and dependent variables, presented by X and Y, respectively. The input X is projected into the latent space by the input-loading factor, P to obtain the input scores, T. Similarly, the output scores, U is obtained by projecting the output Y into latent space through the output-loading factor, Q. These relations are in matrix form and are written in Equation (2.4) and (2.5).

Outer relations: $X = TP^T + E_f$ (2.4)

$Y = UQ^T + F_f$ (2.5)

The matrices E_f and F_f are residuals of X and Y, respectively. X and Y are linked with a linear regression called inner relation, B to capture the relationship between the inputs and output latent scores. The notation of the inner relation is written in Equation (2.6).

Inner relation: $U = TB$ (2.6)

The procedure of determining the scores and loadings vector is carried out sequentially from the first factor to the *f*th factor. Scores and loading vectors for each factor is calculated from the previous residual matrices as shown in Equation (2.7) and (2.8), where initially $E_0 = X$ and $F_0 = Y$.

For X, $E_f = E_{f-1} - T_f P_f^T$ (2.7)

For Y, $F_f = F_{f-1} - U_f Q_f^T$ (2.8)

Calculation of the inner and outer relations is performed until the last factor, f or when residual matrices are below certain threshold. The algorithm of the PLS model is attached in Table 2.1, while Figure 2.5 illustrated the PLS model schematically.

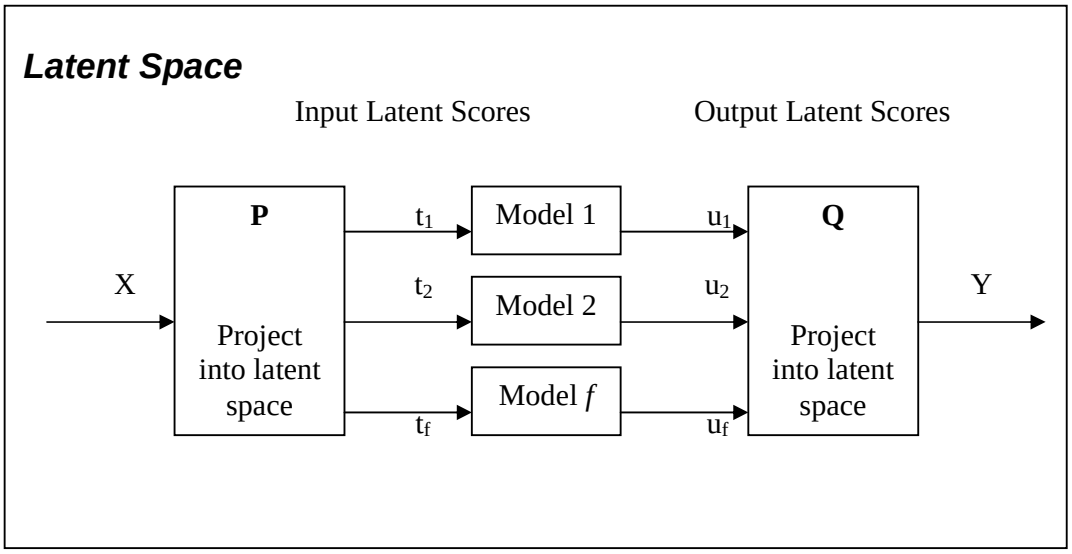

Figure 2.5: Schematic of the PLS model (Adebiyi and Corripio, 2003)

Table 2.1: Algorithm of PLS model (Geladi and Kowalski, 1986)

Step	Summary of Steps	
0	Mean center and scale X and Y	
1	Set the output scores u equal to Y	
2	Compute input weights w by regressing X	$w^T = \frac{u^T \cdot X}{u^T \cdot u}$
3	Normalize w to unit length	$w = w/\|w\|$
4	Calculate the input scores t	$t = \frac{X \cdot w}{w^T \cdot w}$
5	Compute output loadings q	$q^T = \frac{t^T \cdot Y}{t^T \cdot t}$
6	Normalize q to unit length	$q = q/\|q\|$
7	Calculate new output scores u	$u = \frac{Y \cdot q}{q^T \cdot q}$
8	Check convergence on u If yes, go to step 9, else go to step 2	
9	Calculate the input loadings p by regressing X on t	$p^T = \frac{t^T \cdot X}{t^T \cdot t}$
10	Compute inner model regression coefficient b	$b = \frac{t^T \cdot u}{t^T \cdot t}$
11	Calculate input residual matrix	$E = X - t \cdot p^T$ and $F = Y - b \cdot t \times q^T$
12	If additional PLS dimensions are necessary, replace X and Y by E and F, respectively and repeat steps 1 to 12	

2.2.2 Model Development

In this section, development of the inferential estimator based on PLS model is described. The procedure of the PLS model development is as follows:

i) Measurable secondary measurements were selected as input variables of the model
ii) Several sets of data were prepared for training and validation
iii) Data sets were pre-processed using appropriate method
iv) The model was trained using sets of data generated.
v) Performance of the model was investigated. When the performance was not satisfactory, the dimension used in the model was adjusted until the lowest MSE was achieved.
vi) The final model was finally formulated using adjusted dimension and applied for off-line estimation

The procedure shown above can be illustrated in Figure 2.6

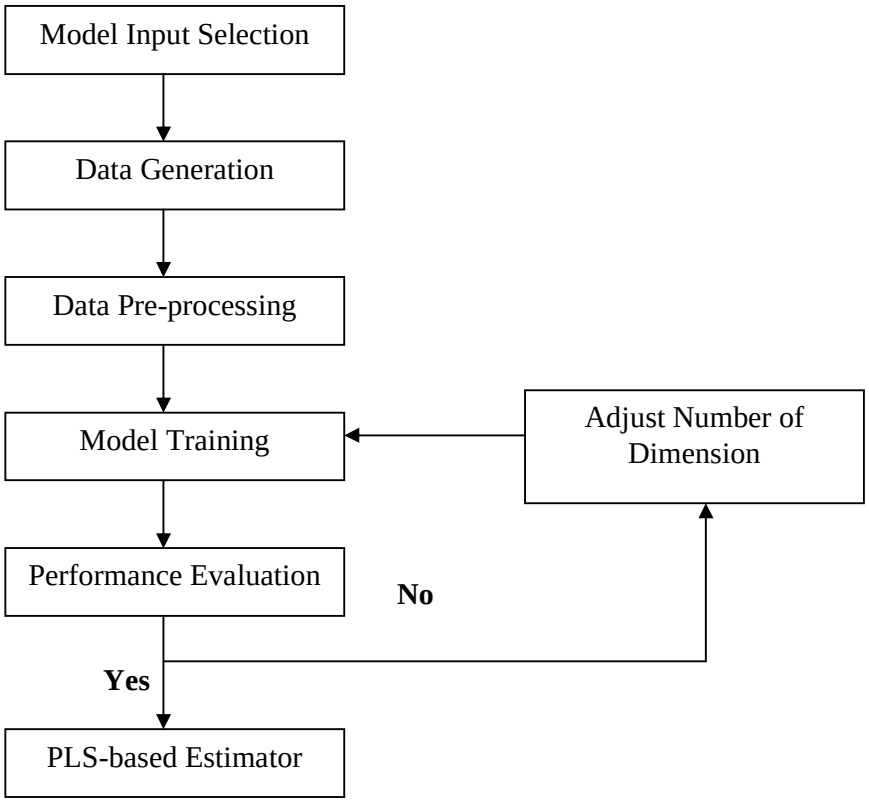

Figure 2.6: Procedure of formulating PLS-based estimator

2.2.3 Data Pre-processing

In order to ensure the model consistency, data pre-processing was implemented in this estimator model. The implementation of data pre-processing also prevents the latent variable from being biased towards variables with larger magnitude. In this work, data pre-processing step can be divided into two parts, i.e., mean-centering and scaling of variables. The data was tailored in mean-centered form prior to scaling. Generally, there are three ways to treat the variables (Geladi and Kowalski, 1986):

i) No scaling is needed when all variables in a block are measured in the same units
ii) Variance scaling is utilized as the variables are measured in different units
iii) Assigning smaller weights to variables with less importance as well as influence on the model

For convenience and simplification, variance scaling was selected among the above method. Mean and variance scaling can be carried out using the following equation:

$$x_m = (x - \bar{x}) / \sigma_x \tag{2.9}$$

Where

x represents the original data;

x_m represents the mean-scaled data;

$\bar{x}$ represents the mean value;

σ_x represents the standard deviation.

2.2.4 Model Training and Validation

In general, the most important and easiest way to evaluate the performance of a model is to measure the estimation accuracy. The estimation accuracy can be defined as the different between the actual and estimated values. Some of the approaches of measuring the accuracy is sum square error (SSE), root mean square error (RMSE) and mean absolute percentage error (MAPE). But the most frequently used is the mean square error of prediction (MSE) (Zhang and Lennox, 2004). The calculation of MSE is shown in Equation (2.10).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i))^2 \tag{2.10}$$

Where

x is the measurement of the product composition;

$\hat{x}$ is its estimation value;

N is the number of measurement.

In addition, explained prediction variance (EPV) as shown in Equation (2-11) that describes the statistical properties of the estimation model was also computed. EPV of X indicates how much of the X block is used in the estimation model and EPV of Y indicates how far the Y block has been estimated.

$$EPV = \left\{1 - \frac{\sum_{n=1}^{N}(x(n) - \hat{x}(n))^2}{\sum_{n=1}^{N}(x(n) - \bar{x})^2}\right\} \times 100\% \tag{2-11}$$

Where

x is the measurement of the product composition;

$\bar{x}$ is the mean value of measurement;

$\hat{x}$ is its estimation value;

N is the number of measurement.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter focuses on the achievement of the conceptual study, simulation work, analyzing and completion of the project. The detailed experimental procedure will be discussed throughout this chapter. There are seven main stages in achieving estimation of vinyl acetate monomer concentration.

3.2 Research Stages

The objective of this work was to develop and implement of process estimator using Neural Network (NN) and Partial Least Square (PLS) scheme for vinyl acetate monomer process. To achieve this target, the research methodologies were divided into several main phases. These were data preparation, NN model development, PLS model development, and finally process estimation. The steps of these phases are summarized in the flowchart as shown in Figure 3.1.