Partial Least Squares and Neural Network Based Identification of Process Dynamics & Design of Neural Controllers

Project Report Submitted in partial fulfillment of the requirements for the

Degree of

BACHELOR OF TECHNOLOGY

in

CHEMICAL ENGINEERING

By

K S Kaushikaram

Roll 10600022

Under the guidance of

Prof. M. Kundu

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA ,ORISSA -769 008, INDIA

Session 2006-2010

DEPARTMENT OF CHEMICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

ORISSA -769 008,INDIA

CERTIFICATE

This is to certify that the project entitled **Partial Least Squares and Neural Network Based identification of Process Dynamics & Design of Neural Controllers** submitted by **K S Kaushikaram (10600022)** in partial fulfillment of the requirements for the award of Bachelor of Technology Degree in Chemical Engineering at NIT Rourkela is an authentic work carried out by him under my supervision and guidance.

_____

Date: ***(Prof. M Kundu)***

Department of Chemical Engineering
National Institute of Technology
Rourkela - 769008
INDIA

ABSTRACT

Present study implemented the Neural network (NN) and Partial least squares (PLS) based identification of process dynamics for single-input single output (SISO) as well as multi-input multi-output (MIMO) systems. In the present study, the Neural network (NN) based controller design has been implemented for a non-linear continuous bioreactor process. Multilayer feed forward networks (FFNN) were used as direct inverse neural network (DINN) controllers as well as IMC based NN controllers. The training as well as testing database was created by perturbing the open loop process with pseudo random signals (PRS). DINN controllers performed effectively for set-point tracking. To address the disturbance rejection problems, which are very likely to be faced by the bioreactors, the IMC based neural control architecture was proposed with suitable choice of filter and disturbance transfer function. To assess the controllability of the various configurations, like conventional turbidostat and nutristat& concentration turbidostat and nutristat, the offset or degree of disturbance rejection by the proposed IMC based NN controllers were utilized. The 'concentration turbidostat' using the feed substrate concentration as the manipulated variable was found to be the best control configuration among the continuous bioreactor configurations.

A (2×2) distillation column was simulated to generate the time series data consisting various inputs and outputs of the process. Multivariate statistical technique PLS was used to relate the scores of input and output matrices. ARX as well as linear least squares techniques were used for inner relation development between the input-output scores. The PLS model of the distillation column dynamics could simulate the process with reasonable accuracy.

Keywords: turbidostat; nutristat; DINN, IMC, controllability, FFNN, Filter

ACKNOWLEDGEMENT

First and foremost I take this opportunity to express my deepest sense of gratitude to my guide Prof. M Kundu for her able guidance during this project work. This project would not have been possible without her help and the valuable time that she has given me amidst her busy schedule.

I thank the coordinators Prof R.K Singh and Prof. H. M. Jena. I am also grateful to Prof. (Dr.) S. K. Agarwal, Head of the Department, Chemical Engineering for providing me the necessary facilities for successful completion of the project.

I would like to thank all the staff members of my department who have been very cooperative with me. I would like to thank NIT Rourkela for giving me the opportunity to use its resources and work in such a challenging environment.

K S Kaushikaram **Date:**
Roll No. 10600022
B.Tech, 8th semester
Chemical Engineering

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

$\mu, \mu max$	Specific growth rate, maximum growth rate
C_i	Concentration of ith species
D	Dilution rate
F	Volumetric flow rate
k_1, k_2, k_3	Reaction rate constants
k_m, k_1	constants
Ni	i^{th} layer of Neurons in a network
$u(t), y(t)$	Inputs and Outputs of a process
V	Volume of Reactor
x_1, x_2	Concentration of biomass and substrate
Y	Yield of biomass

CHAPTER 1

INTRODUCTION AND LITERATURE REVIEW

Introduction and Literature Review

This chapter presents the theoretical foundation of Neural network and Partial least squares in identification of process Dynamics of CSTR, Bioreactor & Distillation processes. This chapter also documents the relevant previous work. The design of neural controllers especially for various continuous bioreactor configurations, related R&D activity undertaken over the last decade has been documented.

1.1 Artificial Neural Networks

As one branch of artificial intelligence, the theory of artificial neural networks (ANN) was first introduced in the middle of the 20th century and numerous advances have been made since then. ANNs have helped solve various problems, including pattern recognition, optimization, control, forecasting and prediction, etc. and is an area of continued research in diversifying the applications. In this section, important concepts and facts related to artificial neural networks shall be briefly reviewed.

1.1.1 Basis of Artificial Neural Networks

Artificial neural networks are inspired by the natural neurons that are in the human brain. Mathematically, they are very similar. A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity to for storing experimental knowledge and making it available for use. It resembles a brain in two respects (Haykin, 1999):

- Knowledge is acquired by the network from its environment through a learning process.
- Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

The natural neuron has a certain number of connections, called *dendrites*. These dendrites receive thousands of electrical stimulations called local potentials. A signal with a positive coefficient is called an excitation signal, while an inhibitor signal has negative coefficients. The dendrites meet at

a common point: the cellular body. If the sum of all local potentials is above a certain threshold to excite the cellular body, these are transformed into a big electrical stimulation. The stimulation then propagates quickly on the output part of the neuron: *the axon*. The axon is then spanned into several little links, which lead to synaptic connections with other neurons. All these operations are chemically realized. Billions of neurons are interconnected with each other to form the human brain's neural network. These send information back and forth to each other; the result is an intelligent being capable of learning, analysis, prediction and recognition.

Artificial neural networks are formed from up to thousands of simulated neurons that are connected in much the same way as the brain's neurons and are thus able to learn in a similar manner to human beings. In artificial neural networks, we reproduce the chemical reactions within biological neuron with a computational model: each neuron receives signals from the others. Before entering a neuron, a signal is multiplied by a coefficient, called a synaptic coefficient. Then, the signals are added together. A particular function is applied to this sum. The output response from the neuron is obtained, which become the input for other neurons.

1.1.2 Advantages of Artificial Neural Networks

Artificial neural networks are good at pattern recognition, trend prediction, modeling, control, signal filtering, noise reduction, image analysis, classification, and evaluation. In fact, the uses for neural networks are so numerous and diverse that these applications may seem to have nothing in common. However, they all share the ability to make associations between known inputs and outputs by observing a large number of examples (Lawrence, 1994). The use of ANNs offers the following useful properties and capabilities:

- Nonlinearity
- Input-Output Mapping
- Adaptivity

- Evidence Response
- Contextual Information
- Fault Tolerance
- Neurobilogical Analogy

An artificial neural network is excellent for any application requiring pattern recognition. A pattern may consist of visual, numeric, or symbolic data. Artificial neural networks are able to recognize patterns even when the data have inherent noise, or have a great amount of variation. To train a neural network to recognize patterns, large input sample space with the correct identification are needed. Pattern recognition is the easiest thing to train a neural network to do. If the problem involves recognition or classification, a neural network will do it faster, more consistently, and often better than a person.

Artificial neural networks are clever and intuitive, they learn by example rather than by following programmed rules. An artificial neural network can be used to solve a problem, by having only a general understanding of what the important factors are. One must know which information is important so that a good variety of data can be selected to train a network with. One does not need to be certain how important each type of data is. Once the network is trained and the relationships between the factors have been learnt, the network can predict which factors are most important or have the greatest effect on the output.

Often engineering problems are difficult to compute and do not require perfect answers, but quick good answers. ANNs also handle these circumstances very well. Precision (A feature of traditional computational techniques) is not always desirable. It is often more important, for example, that a robot arm be moved quickly in some general direction rather than slowly in exactly the right direction. A program that uses complicated formulae to calculate direction, speed, volume, or any other quantity can be a lot slower to respond than a neural network. A neural network does not

need detailed measurements or calculations. It learns the positions and angles in space directly. It can make generalizations about the spatial relationships after learning from a few examples, and it can relearn with new values as the equipment wears and changes with age. ANNs are thus used instead of traditional programming methods when the rules are not certain or when they change over time. They are a supplement to, not a replacement for conventional computer programs. They are currently ineffective in performing serial logic and precise complicated arithmetic. Serious attempts are being made to better understand the biological mechanisms of thought and to incorporate them into artificial neural networks.

1.1.3 Computational Models of Neuron

Neurons process input and produce output. Each neuron takes in the output from many other neurons. Once inside the neuron, the weighted signals are summed to a net value. In most models, they are simply added together. The inhibitory signals have a negative weight value. Thus, when added in with excitatory signals they reduce to the overall signal input. The equation below is basic to all neural networks. (Haykin, 1999):

$$net_i = \sum_{j=0}^{p} (w_{ij} x_j) \tag{1.1}$$

The equation means the net value for neuron i, net_i equals the sum of the weight times the input signal for all the inputs to the neuron i from neuron j starting at output of neuron j = 1 and ending at j = p. It is the addition of the signals that are coming into this neuron, taking the connection strengths of each signal into account. After finding the weighted sum of its inputs net_i, the neuron calculates its output by applying an activation function, which produces an activation level a_i inside the neuron. The activation is passed through an output, or transfer function f_i, which produces the actual output for that neuron for that time, $y_i(t)$.

In the simplest models, the activation function is the weighted sum of the neuron's inputs; the previous state is not taken into account. In more complicated models, the activation function also uses the previous output value of the neuron, so that the neuron can self-excite. These activation functions slowly decay over time. Sometimes the activation function is stochastic, i.e . it includes a random noise factor. The state of activation is a way to refer to the state of the neural network at a given time. Each neuron has an individual activation value which can be written as $a_i(t)$, where a means activation, i is the neuron and t is a particular time. The activation function specifies what the neuron is to do with the signals after the weights have had their effect. The activation function could even be used to do some sort of time integration of the inputs, so that the neuron and the network exhibit time dependent behavior. This behavior is an area of active research, but there are generally no useful results or understanding yet.

The activation is passed through a transfer function, which produces the actual output for that neuron. The transfer function of a neuron defines how the activation value is output. McCulloch and Pitts, (1943) proposed a binary threshold unit as the transfer function. Their model is shown in Figure 1.1. Mathematically, it can be represented as:

$$y_i = \theta \sum_{j=0}^{p} (w_{ij} x_j - u) \tag{1.2}$$

where y_i is the output of the neuron i, θ is a unit step function, and w_{ij} is the synapse weight associated with jth input. For simplicity of notation, the threshold u can be considered as another weight w_0 and be attached to the sum with a constant input $x_0 = 1$. McCulloch & Pitts(1943) proved that, in principle, suitably chosen weights let a synchronous arrangement of such neurons perform universal computation. This model contains a number of simplifying assumptions. This neuron model has been generalized in many ways. One of them is to use a transfer function other

than the threshold function, such as piecewise linear, sigmoidal, or Gaussian. The most common is the sigmoid function. It is a strictly increasing function that exhibits smoothness and has the desired asymptotic properties. The standard form is the logistic function, defined by

$$f(x) = \frac{1}{(1 + exp\{-\beta x\})} \tag{1.3}$$

where, β is the slope parameter.

The sigmoid function is a particularly useful nonlinear transfer function. This transfer function is a saturation function; excitation above some maximum firing level has no further effect. The sigmoid function has a high and a low saturation limit, and a proportional range in between. This function usually produces a 0 when the activation value is a large negative number and a 1 when the activation value is a large positive number, and makes a smooth transition in between. The sigmoid transfer function thus produces an output from -1 to +1 in some networks. Regardless of the exact transfer function, a neuron fires when it recognizes a particular value combination of incoming signals. In other words, the operation of a neuron is defined by the match between the input vector, consisting of incoming signals, and a weight vector of internal parameter set.

1.1.4 Neural Network Architecture

Artificial neural networks can be viewed as weighted and directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs (Jain, 1996). Based on their connection patterns, neural networks can be grouped into two categories:

- Feed-forward networks, in which no loop exists.
- Feedback (recurrent) networks, in which loops occur because of feedback connections.

Artificial neural networks consist of different types of layers. There is the input-layer, one or more hidden layers and an output layer. All these layers can consist of one or more neurons. A neuron in a particular layer is connected to all neurons in the next layer, which is why this is called a feed-forward network. In other networks the neurons might be connected in other ways. An example of a different network is a recurrent neural network where there are also links that connect neurons to other neurons in a previous layer. Feed forward neural networks can be supervised or unsupervised. A supervised network compares its answers during training to known correct answers, whereas and unsupervised network (self-organizing) does not. Different network architectures require different learning algorithms (Lawrence, 1994). The next section will discuss the most common learning processes.

1.1.5 Learning Algorithms

The ability to learn is a fundamental trait of artificial neural networks. The most attractive characteristics of artificial neural networks is their ability to mathematically learn by examples and repetitions.

There are basically two learning paradigms: supervised learning and unsupervised learning. Supervised learning is the most elementary form of adaptation. During training, it requires an a priori knowledge of what the result should be. Output neurons are told what the ideal response to input signals should be. For one-layer networks in which the stimulus-response relation can be controlled closely, this is easily accomplished by monitoring each neuron individually. In multi-layer networks, supervised learning is more difficult. It is harder to correct the hidden layers. On the contrary, unsupervised learning does not have specific corrections made by an observer. Supervised and unsupervised learning are methods used exclusive of each other.

In the supervised learning, there exists a “teacher”, which may be implemented in various ways. This trainer corrects the network’s responses to a set of inputs. Pairs of inputs and outputs are

presented to the network. The network takes each input and produces an output, then it compares to the correct output. The trainer causes the network to construct an internal representation that captures the regularities of the data in a generalized way. This is the form of learning which is best understood, and is presently most suitable to real applications.

In unsupervised learning, no "teacher" is involved. Instead, the network is simply exposed to a number of inputs. The network organizes itself in such a way as to come up with its own classifications for inputs.

Back-propagation

The gradient-descent back-propagation originates from error-correction rule. The idea behind this rule is that the learning algorithm modifies the parameters of the network in the direction in which the total error decreases most rapidly for the current point. The algorithm modifies synaptic weights and biases of the network in search of the global minimum of the objective function. It moves across the error surface in the direction of steepest descent. This is a stochastic process, which means that it zig-zags its way about the true direction to the minimum of the error surface. For a multiple-layer linear feed-forward network using back-propagation learning, there is only one minimum and all other critical points are saddle points.

Gradient-descent back-propagation may not reach the optimum in all cases, because it can be caught in a local minimum or diverge. It is also a relatively slow learning algorithm. For those reasons, there are modifications of the gradient-descent back-propagation algorithm that improve the convergence. A momentum term is added in order to prevent the training from getting stuck in local minima. This momentum term makes the algorithm take the running average of the gradient to make it less sensitive to small fluctuations. Also, the learning rate can be modified dynamically. Dynamic modification of the learning rate allows the network to learn faster when the error gradient is large but keeps it from overshooting when the gradient is small. Finally, batch training

can be used instead of sequential training to make the training process less sensitive to anomalous data points. Batch mode training is generally more stable than pattern-mode training because the effect of anomalous data points is small when they are aggregated with a large number of "normal" data points.

In the learning process, the multiple-layer perceptron network uses two different kinds of signals: function signals and error signals. Function signals are signals that enter the network through the input nodes, propagate through the network, and emerge as output signals. Error signals are signals that originate at the output layer of the network and propagate backwards though the network (Haykin, 1999). To train a multiple-layer perceptron, a function signal is placed through the network, and its output is compared to some desired output. The resulting error signal is passed backward through the network. In the forward pass, when the function signal is passing through the network, the weights and biases of the network are fixed. The function signal appearing at the output of neuron *i* at iteration *n* is computed as

$$y_i(n) = \theta\big(a_i(n)\big) \tag{1.4}$$

where $a_i(n)$ is the net internal activation level of neuron *i* , defined by

$$a_i(n) = \sum_{j=0}^{p} \big(w_{ji}(n) y_j(n)\big) \tag{1.5}$$

where, p is the total number of inputs (excluding the threshold) applied to neuron j , and w_{ji}(n) is the synaptic weight connecting neuron j to neuron i, and $y_j(n)$ is the input signal of neuron i or, equivalently, the function signal appearing at the output of neuron j (Haykin, 1999). This can also be written as the dot product of w and y vectors.

In the backward pass, when the error signal is passing backward through the network, the weights and biases of the network are modified. The synaptic weights of the network in layer l are adjusted for the n + 1 iteration according to the rule:

$$w_{ij}^{(l)}(n+1) = w_{ij}^{(l)}(n) + \gamma\left[w_{ij}^{(l)}(n) - w_{ij}^{(l)}(n-1)\right] + \eta\delta_i^{(l)}(n)y_j^{(l-1)}(n-1) \tag{1.6}$$

where, δ and η are constants and $\delta_i^{(l)}$ is the local gradient of network for the i^{th} neuron in the l^{th} layer and is computed from

$$\delta_i^{(l)}(n) = y_i^{(l)}(n)\left[1 - y_i^{(l)}(n)\right]\sum_k \delta_k^{(l+1)}(n)w_{kj}^{(l+1)}(n) \tag{1.7}$$

for any layer l and

$$\delta_i^{(L)}(n) = e_i^{(L)}(n)[1 - o_i(n)] \tag{1.8}$$

for output layer L where, $o_i(n)$ is the ith output at iteration n and $e_i^{(L)}(n)$ is the error of that output defined as

$$e_i^{(L)}(n) = d_i(n) - y_i(n) \tag{1.9}$$

with, the desired output $d_i(n)$ at iteration n.

The gradient descent back-propagation algorithm is the most commonly used training algorithm. It is a recursive algorithm, which means that the points can be given to the algorithm one at a time. It is mostly stable, but it is slow.

1.1.6 Steps in Design of a Network

• Creating a network.

• Training the network.

• Simulating the network.

For training a network using back-propagation algorithm the commonly used inbuilt MATLAB functions are train, traingd, traingda, trainrp. There is also different performance parameters used while training a network using MATLAB simulator like epochs, show, goal, time, min grad etc. It is an iterative trial and error procedure. In this work the optimum architecture evolved out through an elaborate trial and error procedure of selecting the right number of nodes, activation function and inputs. After a network has been created and trained with an optimal number of data, the network can be used to simulate unknown inputs to predict the result.

1.1.7 Limitations of ANN

Although artificial neural networks are very powerful tools for dealing with complex problems they are not cure-all. ANNs heavily rely on their training samples. If the training samples are insufficient or do not cover all the typical conditions of the problem, errors can be large with testing samples. If the training samples are too much, they can also cause the over fitting problem. The most important limitation of ANNs is that they do not reveal the exact nature of the relationship between inputs and outputs; Thus ANN models are hard to convert to rules. Besides, confidence intervals for predictions are not always available. Multiple models can be created from the same training data, as the nonlinear multivariable optimization for weight and biases is a ”hard problem” with no guarantee of finding the global optimum. Due to compounding nonlinearity, the model behavior could be erratic in localized regions of the multidimensional input space.

1.1.8 Continuous Stirred Tank Reactor (CSTR) & NN Controller

CSTR is extensively used in industry because of its flexibility in operation. It is a preferred reactor for high volume manufacturing processes. Hugo and Steinbach (1986) described CSTR as open dynamic system and there may be other side reactions; if it is not properly controlled. This out of control situation leads to runaway conditions, whereby the temperature and pressure of the reactor may exceed endurance limit. The NN based identification of CSTR operation and design of DINN SISO controllers have been one of the objectivity of the present Project.

1.1.9 Design of Neural Controllers & Bioreactor

The Neural network stores knowledge in two forms a) the connection between the nodes b) the weight factors of these connections, Neural networks are better suited for processing noisy, incomplete, or inconsistent data and Neural networks mimic human learning processes. In the recent years, there have been significant advances in control system design for non-linear processes. One such method is the non-linear inverse model based control strategy. This method is dependent on the availability of the inverse of the system model. Neural networks (NN) have the potential to approximate any non-linear system including their forward & inverse dynamics. Inverse neural network have been utilized as the controller. For training the neural network, the process input-output data is generated by applying a pseudo random signal to the open loop process and the learning is carried out by considering the future process outputs as the reference set point. IMC (Internal model control) strategy integrates the plant model and its inverse in a feedback control loop. NN based IMC scheme is used; especially for disturbance rejection problem. Application of NN based controllers in chemical processes have gained huge momentum as a result of focused R&D activities taken up by several researchers over the decade. (Donat, 1990); (Ydstie, 1990); (Hernandez, 1990); (Dimitris, 1991); (Dirion, 1995) (Hussain, 2001) (Varshney, 2009).

Bioreactor control has been an active area of research over a decade or so. For optimization of cell mass growth and product formation continuous mode of operation of bioreactors are desirable not

the traditional fed batch bioreactors. Several researchers like Edwards(1972), Agrawal(1984), Menawat (1991), have studied the continuous bioreactor problem. A (2×2) bioreactor process have been taken up to state its different desirable control configuration at the various operating points of the bio-process. The parameters like specific growth rate (μ), yield constant (Y) , & saturation rate constant (k_1, k_m) of the kinetic models are either inadequately determined or vary from time to time regarding the process operation. The aforesaid parameters have been considered as disturbance to the process. The disturbance rejection has given a consideration in selecting suitable control configuration for the continuous bioreactors. The primary aim of a continuous bioreactor is to avoid wash out condition which ceases reaction. This may be done either by controlling cell mass (X g/L) or substrate concentrations (S or x_{2f} g/L). In order to maintain the reaction rate and product quality both of them may be controlled with dilution rate ($D=F/V$ (h^{-1})) and feed substrate concentration (S_f or x_{2fs} g/L) as manipulated variables, thus two degrees of freedom is available for control. However this is expensive and redundant probably, because microorganisms have intracellular regulatory mechanisms, and there exist strong interaction between the two outputs (Zhao, 1997) Four numbers of (1×1) control configurations are possible which are as follows:

- Conventional turbidostat ($D \rightarrow X$): Dilution rate is used to control cell concentration
- Conventional nutristat ($D \rightarrow S$): Dilution rate is used to control substrate concentration
- Concentration turbidostat $(S_f \rightarrow X)$: Feed substrate concentration is used to control biomass or cell concentration
- Concentration nutristat $(S_f \rightarrow S)$: Feed substrate concentration is used to control substrate concentration

Considering the immense commercial significance of the continuous bioreactor process the ANN based non-linear controller design have been implemented for various configurations of it. In the present study, the direct inverse neural network controllers (DINN) were designed for conventional turbidostat and nutristat for set point tracking at an operating condition where the cell growth is substrate limited. IMC based NN controllers were designed for conventional turbidostat and

nutristat & concentration turbidostat and nutristat with which their disturbance rejection performance were tested; as well as controllability of those configurations were assessed.

1.2 Partial Least squares

Partial Least Squares (PLS) attempts to find so-called latent variables that capture the variance in the data and at the same time achieves maximum correlation between predicted variables Y and predictor variables X. Originally, PLS was a technique that would produce a static linear model, although also non-linear and dynamic versions have been published in the literature. As principal components in principal component analysis, the use of latent variables in PLS can reduce the dimensionality of the problem considerably. (Roffel & Betlam, 2006)

The design of controllers for MIMO process is possible only after the development of complete model describing the effects of all the process inputs on all the process outputs. The first principles based models are not always available especially when the process is complex and they are not suitable in control system design. The multivariable process model is usually obtained empirically by performing an identification experiment and analyzing the recorded plant input-output data. The presence of different time scales, different delays, different orders of inputs and outputs in MIMO system poses a challenge in their identification. Several commercial identification and control packages MATLAB System Identification Toolbox (1992), ADATPx (1992) are capable of estimating linear SISO and/or MIMO dynamic models from observed plant data. In the identification of MIMO processes, a high degree of correlation is often observed between process variables. In such cases, use of identification software based on ordinary least squares will result in parameter estimates with large variance owing to the ill-conditioned nature of the problem. One way to circumvent these problems is to use multivariate statistical techniques such as PCA and PLS, which have been proposed and applied by several researchers like Kresta (1992); Qin & McAvoy (1992a); Qin (1993) in Chemical Engineering problems. The data compression features offered by

those techniques provide a low-dimensional subspace/window to look into the process and facilitates the process monitoring, fault detection & diagnosis. Ku et al. (1995) proposed an extension of the standard PCA technique in order to handle dynamic auto correlated data. Kaspar and Ray (1992, 1993) proposed the PLS technique in identifying process dynamics with its possible merits and demerits. Lakshminarayanan et al. (1997) proposed a strategy for the design of multivariable feed forward controller in the PLS frame work.

1.2.1 PLS Architecture and Process dynamics

The idea of PLS is to develop a model that relates the scores of the X data to the scores of the Y data. The PLS model consists of outer relations (X and Y data individually) and an inner relation that links the X data to the Y data. Two attributes are important while choosing the latent variables for an estimator: (i) stability (ii) obtaining good fit.

The outer relationship for the input matrix or matrix with predictor variables is written as:

$$X = t_1 p_1^T + t_2 p_2^T + \cdots + t_n p_n^T + E = TP^T + E \tag{1.10}$$

Similarly, the outer relationship for the output matrix or matrix with predicted variables can be written as:

$$Y = u_1 q_1^T + u_2 q_2^T + \cdots + u_n q_n^T + F = UQ^T + F \tag{1.11}$$

Where, T and U represent the matrices of scores, P and Q represent the loading matrices for the X and Y data. If all components are described, E and F will become zero.

Through the inner relationship, X and Y should be correlated to the best extent, therefore ||F|| should be minimized. The simplest model relating X to Y is one that relates the scores T to the scores U:

$$U = TB \tag{1.12}$$

in which B is the regression matrix. Combining (1.11) with (1.12):

$$Y = TBQ^T + F \tag{1.13}$$

To determine the dominant directions in which the data should be projected, we calculate the covariance within X and Y. The first set of loading vectors p_1 and q_1, representing the dominant direction, is obtained by maximizing the covariance between X and Y. Projection of the X data on p_1 and Y data on q_1 results in the first set of score vectors t_1 and u_1. This procedure is known as establishing the outer relation (Lakshminarayanan, 1997). The matrices X and Y are related through their respective scores, which is called the inner model, representing a linear regression between t1 and u_1: $\hat{u}_1 = t_1 b_1$.The first two dimensions are shown in Figure 1.2. The residuals are then computed as:

$$E_1 = X - t_1 p_1^T; \qquad F_1 = Y - b_1 t_1 q_1^T \tag{1.13}$$

Using the newly computed residuals, the procedure of determining scores and loading vectors is repeated, until the residuals are below a set tolerance. In practical problems, the number of PLS dimensions (number of latent variables) is determined based on the percentage variance explained. Irrelevant directions due to noise and redundancy are confined to errors E and F. PLS is a technique that breaks a multivariate regression problem into a series of univariate regression problems.

1.2.2 The PLS Algorithm

The procedure for calculating the PLS model given by (Geladi, 1986). The procedure starts with assuming a score vector chosen as any column of matrix X, say x_1, and a score vector u which is any column of matrix Y, say y_1. Once X and Y are auto-scaled, the following steps are carried out: for each component take the following starting values:

1. $u = y_1$

2. t_old = x_1

Calculations for the X matrix:

3. $w = (u^T * X / (u^T * u))^T$

4. $w = w / ||w||$

5. $t = X * w / (w^T * w)$

Calculations for the Y matrix:

if number of Y variables > 1

6. $q = (t^T * Y / (t^T * t))^T$

7. $q = q / ||q||$

8. $u = Y * q / (q^T * q)$

else

q=1

end if

Improvement last step > threshold

9. if ||t-t_told|| > threshold

t_old = t

return to step 3

end if

Calculate the X loadings and rescale the scores and weights:

10. p = (tT * X / (tT * t))T

11. p = p / ||p||

12. w = w * ||p||

13. t = t / ||p||

Calculate for each component i:

14. $b_i = u_i^T t_i / (t_i^T t_i)$

Calculate the residuals.

15. $X = X - t_i p_i^T$

16. $Y = Y - b_i t_i q_i^T$

1.2.3 Dynamic Extensions of PLS

If the U block contains past values of the process inputs and outputs, the PLS model would be a dynamic model of the form (Roffel & Betlam, 2006):

$$Y = G_1(t_1)q_1^T + G_2(t_2)q_2^T + \cdots + F = GQ^T + F \tag{1.14}$$

(Kaspar, 1993) developed dynamic PLS models by filtering the process inputs and subsequent application of the standard PLS algorithm. (Lakshminarayanan, 1997) proposed the modification of the PLS inner relation to be able to identify dynamic models. Instead of using the input scores t_i and output scores u_i, a dynamic ARX model G_i was used. The general ARX model structure can be described by:

$$A(z^{-1})y(k) = B(z-1)x(k-f) + e(k) \tag{1.15}$$

where y is the process output and x the process input $G_i(t_i)q_i^T$ is the measure of the Y space that is explained by the i-th PLS dimension. The matrix G is a diagonal matrix, comprising the dynamic elements identified at each of the n PLS dimensions:

$$G = \begin{bmatrix} G_1 & 0 & 0 & 0 & \dots & 0 \\ 0 & G_2 & 0 & 0 & \dots & 0 \\ \dots & & & & & \\ 0 & 0 & 0 & 0 & \dots & G_n \end{bmatrix} \tag{1.16}$$

in Figure 1.3 z^{-1} is the backward shift operator ($x_{k-1}=z^{-1}x_k$).

The output variable Y can be calculated from:

$$Y(z^{-1}) = [QG(z^{-1})P]X(z^{-1}) \tag{1.17}$$

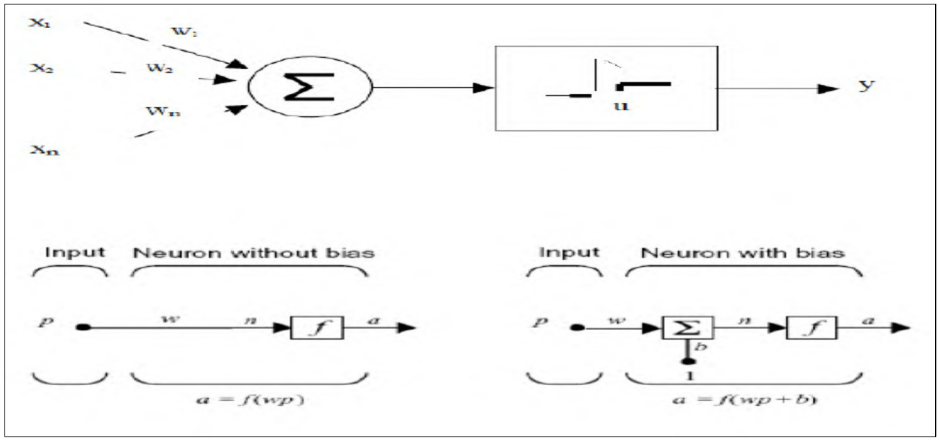

Figure 1.1: McCulloh Pitts model of neuron

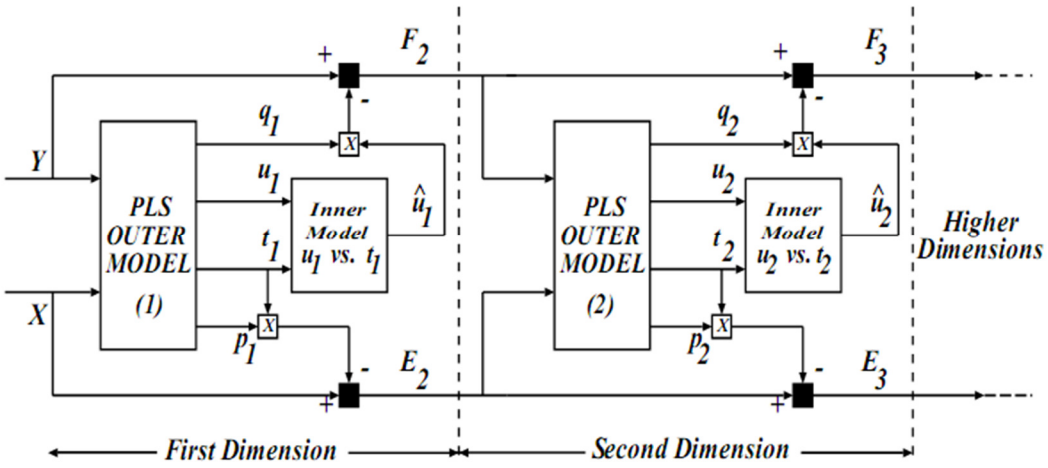

Figure 1.2: Standard linear PLS Algorithm

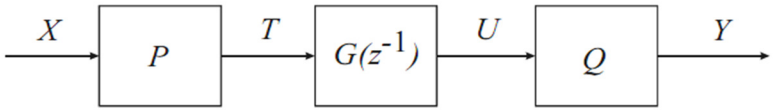

Figure 1.3: PLS based dynamic model

CHAPTER 2

DESIGN OF NEURAL NETWORK CONTROLLER AND APPLICATION

Design of Neural Network Controller & Application

2.1 Continuous Stirred Tank Reactor

Chemical reactors are generally the most important unit operations in a chemical plant. Chemical reactors come in many forms, but two of the most common idealizations are continuous stirred tank reactor Figure 2.1 (CSTR) and the plug flow reactor (PFR). These two serve as limiting hounds for the behavior of many operating reactors. The CSTR is used in dynamic modeling studies, as it can be modeled as a lumped parameter system. (William, 1996)

2.1.1 Van De Vusse Reaction

Reaction schemes often exhibit a maximum in the concentration of product versus flow rate. Consider a system consisting of the following reactions (Bequette, 1998) (Varshney, 2009):

$$A \xrightarrow{k1} B \xrightarrow{k2} C; \quad 2A \xrightarrow{k3} D$$

The desired product is *B.* We assume that the feed stream contains pure component *A.*

2.1.2 Material Balance

Assuming constant density and constant volume

$$\frac{dV}{dt} = 0; \; F_i = F_o \tag{2.1}$$

for the four components,

$$\frac{d(C_A)}{dt} = \frac{F}{V}\left(C_{Af} - C_A\right) - k_1 C_A - k_3 C_A^2 \tag{2.2}$$

$$\frac{d(C_B)}{dt} = \frac{F}{V} C_B + k_1 C_A - k_2 C_B \tag{2.3}$$

$$\frac{d(C_c)}{dt} = -\frac{F}{V}C_c + k_2 C_B \tag{2.4}$$

$$\frac{d(C_D)}{dt} = -\frac{F}{V}C_D + \frac{1}{2}k_3 C_A^2 \tag{2.5}$$

Solving the steady state equations for components *A* and *B* we get,

$$C_{As} = \frac{-\left(k_1 + \frac{F_s}{V}\right)}{2k_3} + \frac{\sqrt{\left(k_1 + \frac{F_s}{V}\right)^2 + 4k_3\left(\frac{F_s}{V}\right)}}{2k_3} \tag{2.6}$$

$$C_{Bs} = \frac{k_1 C_{As}}{\frac{F_s}{V} + k_2} \tag{2.7}$$

2.1.3 State Space Model

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$

$$\boldsymbol{y} = C\boldsymbol{x}$$

Where,

$$\boldsymbol{x} = \begin{bmatrix} C_A - C_{As} \\ C_B - C_{Bs} \end{bmatrix} \tag{2.8}$$

$$u = \begin{bmatrix} \frac{F}{V} - \frac{F_s}{V} \end{bmatrix} \tag{2.9}$$

$$\boldsymbol{y} = \begin{bmatrix} C_A - C_{As} \\ C_B - C_{Bs} \end{bmatrix} \tag{2.10}$$

For this system the state-space matrices are:

$$A = \begin{bmatrix} \frac{F_s}{V} - k_1 - 2k_3 C_{As} & 0 \\ k_1 & -\frac{F_s}{V} - k_2 \end{bmatrix} \tag{2.11}$$

$$B = \begin{bmatrix} C_{Afs} - C_{As} \\ -C_{Bs} \end{bmatrix} \tag{2.12}$$

2.1.4 Direct Inverse Control

For the first step of training a network, the input-output pairs of data is generated using the CSTR model developed earlier. The input is the space velocity and the output is the concentration of species B (Both values in deviation form). At the steady state operating point C_{Bs} = 1.117mol/L for F /V = 4/7min^{-1} other parameters are k_1 = 5/6min^{-1} k_2 = 5/3min^{-1} k_3 = 1/6Lmol^{-1}min^{-1}. The input signal is a Pseudo Random Signal generated using Matlab. The sampling time used is 0.8s refer Figure 2.2 & Figure 2.3. The data generated is divided into three parts each for training, validation and testing of the network.

The inverse neural model shown in Figure 2.4 is a structure representing the inverse of system dynamics once training is complete. During training the network is fed with past inputs and outputs of the system and the present input to system is the target set. The network predicts the output of the controller (space velocity) which is the input to the process. Here we chose four inputs to the controller and the network can be represented as,

$$u(t) = f^{-1}[y(t+1), y(t), y(t-1), u(t-1)] \tag{2.13}$$

The network used was Feed-forward network, training algorithm being Levenberg-Marquardt backpropagation method. The choice of network architecture was based on the performance criteria (i.e) the mean squared error (MSE) between the network output and target. The number of

nodes in the hidden layer is such that the network gives best performance. Upon completion of training, the network was tested. . The closed loop block diagram for servo is shown in Figure 2.5

2.1.5 NN Based Internal Model control

The regulatory problem was studied by giving the disturbance in the feed concentration, C_{A0}. A unit step disturbance was given to the process. The setpoint was kept constant. The performance of the DIN controller was not good as it gave an offset (Figure 2.6). To combat this, a NN forward model was placed across the process see Figure 2.7. Thus becoming an IMC structured closed loop (Anuradha 2009) (Hussain, 2001).Here the disturbance transfer function was taken to be same as the process transfer function.

2.1.6 Results & Discussion

The online performance of the DINN as a controller was compared with a PID based controller for servo problem. The control parameters for PID are taken as Kc = 1.87, τ_I = 1.1 and τ_D = 0.26 (Anuradha 2009).The response is shown in Figure 2.8. The DIN controller shows perfect set point tracking. The ANN based IMC also showed good regulatory performance (Figure 2.9).

2.2 A First Order System

In this section the process model has a first order transfer function (Dirion, 1995). The chosen equation is:

$$\frac{dy}{dt} + \frac{1}{3.5}y = \frac{1}{3.5}u \tag{2.14}$$

The learning database is generated using a pseudo random binary signal PRBS (Figure 2.10 & Figure 2.11).The network structure had three layers with four inputs and one output.

The number of neurons in the hidden layer was similarly chosen which minimizes the MSE. Once the training was complete the network was tested for tracking time varying setpoint (Figure 2.12).

2.3 Bioreactor

Biochemical Reactors (Bequette, 1998) are used to produce products that include pharmaceuticals, food and beverages. Biochemical reactor models are similar to chemical reactor models. We develop the dynamic model by writing the material balances on the biomass and substrate. Biomass grows by feeding on the substrate. The study is based on single biomass-single substrate process. The following are the model equation based on first principle.

2.3.1 Material Balance

RATE OF ACCUMULATION= INFLOW-OUTFLOW + GENERATION - CONSUMPTION

For biomass

$$\frac{d(Vx_1)}{dt} = Fx_{1f} - Fx_1 + Vr_1 \tag{2.15}$$

For substrate

$$\frac{d(Vx_2)}{dt} = Fx_{2f} - Fx_2 + Vr_2 \tag{2.16}$$

The reaction rate is given by

$$r_1 = \mu x_1 \tag{2.17}$$

Where x_{1f} & x_{2f} are the biomass concentration and the feed substrate concentration, respectively. x_1 & x_2 are the biomass and substrate composition, respectively. μ , the specific growth is a function of substrate concentration and given by the substrate inhibition model growth rate expression:

$$\mu = \frac{\mu_{max} x_2}{k_m + x_2 + k_1 x_2^2} \tag{2.18}$$

The relation between the rate of generation of cells and consumption of nutrients is defined by the yield Y

$$Y = \frac{r_1}{r_2} \tag{2.19}$$

Introducing the dilution rate ($D = \frac{F}{V}$) and assuming there is no biomass in the feed

We get the following model equations

$$\frac{dx_1}{dt} = (\mu - D)x_1 \tag{2.20}$$

$$\frac{dx_2}{dt} = D(x_{2f} - x_2) - \frac{\mu x_1}{Y} \tag{2.21}$$

The inputs are dilution rate and substrate concentration and the outputs are the concentrations of substrate and biomass (All values in deviation form). The values of steady state dilution rate (D_s), feed substrate concentration (x_{2fs}), and the various parameters are presented in Table.

2.3.2 State Space Model

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$

$$\boldsymbol{y} = C\boldsymbol{x}$$

Where,

$$\boldsymbol{x} = \begin{bmatrix} x_1 - x_{1s} \\ x_2 - x_{2s} \end{bmatrix} \tag{2.22}$$

$$\boldsymbol{u} = \begin{bmatrix} D - D_s \\ x_{2f} - x_{2fs} \end{bmatrix} \tag{2.23}$$

$$\boldsymbol{y} = \begin{bmatrix} x_1 - x_{1s} \\ x_2 - x_{2s} \end{bmatrix} \tag{2.24}$$

The state-space matrices are as follows:

$$A = \begin{bmatrix} \mu - D_s & x_{1s}\mu_s' \\ -\frac{\mu}{Y} & -D_s - \frac{x_{1s}\mu_s'}{Y} \end{bmatrix} \tag{2.25}$$

$$B = \begin{bmatrix} x_{1s} & 0 \\ x_{2f} - x_{2s} & D_s \end{bmatrix} \tag{2.26}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.27}$$

μ_s' represents the derivative of growth rate with respect to substrate concentration at steady state and given by

$$\mu_s' = \frac{d\mu_s}{dx_{2s}} \tag{2.28}$$

$$\mu_s' = \frac{\mu_{max}(k_m - k_1 x_{2s}^2)}{(k_m + x_{2s} + k_1 x_{2s}^2)^2} \tag{2.29}$$

The above state space matrices were used to find turbidostat and nutristat transfer functions for set point tracking problems. Solving the steady state equations (2.20) & (2.21), we get three different equilibrium points depending on the initial conditions.

- When biomass concentration is zero, it is a washout condition with zero gain, trivial solution.
- When both the concentrations (biomass & substrate) are high it leads to unstable equilibrium
- When there is substrate limiting condition it is a stable equilibrium.

The system model around the second equilibrium point renders unbounded outputs when excited with pseudo random binary signals (PRBS). So the steady state values of third equilibrium; x_{1s} = 1.5302g/L, x_{2s} = 0.1746g/L were considered for the database development required for training the NNs. For disturbance rejection problems, which were implemented using IMC based NN scheme, a state space model was developed to determine a (2×5) disturbance transfer function matrix. The various disturbances considered were μ_{max}, k_m Y, x_{2fs}, & D_s and the disturbance transfer function transfer functions were of same order to that of the process. Following are the state space matrices.

$$A = \begin{bmatrix} \mu - D_s & x_{1s}\mu_s' \\ -\frac{\mu}{Y} & -D_s - \frac{x_{1s}\mu_s'}{Y} \end{bmatrix} \tag{2.30}$$

$$B = \begin{bmatrix} \frac{\mu}{\mu_m} & \frac{-\mu_m x_{2s}}{(k_m + x_{2s} + k_1 x_{2s}^2)} & 0 & -x_{1s} & 0 \\ -x_{1s}\frac{\mu}{\mu_m} & \frac{-\mu_m x_{2s}}{(k_m + x_{2s} + k_1 x_{2s}^2)}\frac{x_{1s}}{Y} & \mu\frac{x_{1s}}{Y^2} & -(x_{2s} - x_{2fs}) & D_s \end{bmatrix} \quad (2.31)$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (2.32)$$

$$D = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (2.33)$$

Where load matrix is

$$u = \begin{pmatrix} \mu_{max} \\ k_m \\ Y \\ D \\ x_{2fs} \end{pmatrix} \quad (2.34)$$

2.3.3 Direct Inverse Control

In the direct method, a NN is trained with observed input-output data from the open loop process to represent its inverse dynamics. Hence the resulting inverse NN model can be used as a controller typically in a feed forward fashion. In an IMC based NN scheme, a NN based process model is placed parallel with the process. The difference between the process and the network output is used for the feedback purpose. This feedback signal is then processed by the inverse NN in the forward path. It is to be noted that the implementation of IMC based NN is limited only to open loop stable processes. The learning phase of the network is an off-line process and the historic data base of the process is used for training and testing the networks. In the present study, the training as well as

testing database was created by exciting the open loop process with pseudo random binary signals (PRBS).

In order to develop DINN controller, the training of the proposed multi layer FF NN (4, 3, and 1) was performed using the gradient based Levenberg-Marquardt method. Performance criterion was MSE between the network output and target. The network predicted the outputs of the controller (D_s, & x_{2fs}) which actually are the manipulated variable to the process. The inputs and outputs of NN (4, 3.1 or N1, N2 & N3) regarding the training & control phase were as follows,

Training Phase

$$N1 = \{y(t), y(t-1), y(t-2), u(t-2)\} \tag{2.35}$$

$$N3 = u(t-1) \tag{2.36}$$

Control Phase

$$N1 = \{y(t+1), y(t), y(t-1), u(t-1)\} \tag{2.37}$$

$$N3 = u(t) \tag{2.38}$$

Figure 2.4 presents the network architecture in the control mode. Sampling time of 0.8 time unit and 2 time unit were used for training the two kinds of turbidostat ($D \rightarrow X$) servo networks, each of the networks demonstrated minimum offset of 0.001 with simulation intervals of 2 time unit and 4 time unit respectively. For nutristat ($D \rightarrow S$) servo networks, sampling time of 0.8 time unit and 2 time unit were used for training, each of them demonstrated minimum offset of 0.001 with simulation intervals of 4 time unit and 3 time unit, respectively.

2.3.4 NN Based Internal Model control

An IMC structured closed loop was used with a disturbance transfer function of same order to that of process transfer function. The various disturbances considered were $\mu_{max}, Y,\ k_m, D_s, \& \ x_{2fs}$) . In IMC scheme, the proposed FF network (6, 3, and 1), which is actually the process model, used the following inputs & outputs,

Training phase

$$N1 = \{y(t-3), y(t-2), u(t-3), u(t-2), u(t-1), u(t)\} \tag{2.39}$$

$$N3 = y(t-1) \tag{2.40}$$

Simulation Phase

$$N1 = \{y(t-2), y(t-1), u(t-3), u(t-2), u(t-1), u(t)\} \tag{2.41}$$

$$N3 = y(t) \tag{2.42}$$

Figure 2.13 represents the block diagram of closed loop IMC scheme. The following filter transfer function was used in the closed loop simulation.

$$G_f = \frac{\gamma S + 1}{(\lambda s + 1)^2} \tag{2.43}$$

Where $\lambda = {}^{\tau_p}/_{5}$, and $\gamma = \frac{(2\lambda\tau_p - \lambda^2)}{\tau_p}$

2.3.5 Results & Discussion

The performance of the developed controllers was tested of their set point tracking ability. The closed loop response in biomass concentration for unit step change in dilution rate at t=20 th time instant using the conventional turbidostat servo controllers are shown in Figure 2.14 & Figure 2.15, which reflect a perfect set point tracking. For monitoring substrate concentrations, the closed loop response of the substrate concentration for unit step change in dilution rate at t=24 th time instant using designed conventional nutristat controllers are shown in Figure 2.16 & Figure 2.17, which also ensures the perfect set point tracking.

To assess the controllability of each of the continuous bioreactor configurations, the closed loop disturbance rejection performance of them were taken in to consideration. Table.2 represents the offset in disturbance rejection by all 4 continuous bioreactor configurations in closed loop without any adjustment of bias. For the present equilibrium point where the cell growth is substrate limited, the concentration turbidostat using the feed substrate concentration as the manipulated variable seems the best control configuration. The performance of conventional turbidostat is poor in rejecting the disturbance in the yield Y. This study reveals that conventional nutristat is unacceptable control configuration when it is the case of disturbance rejection in μ_{max} & k_m. Concentration nutristat is incapable of disturbance rejection in D_s $(dilution\,rate)$. Figure 2.18 shows the open loop responses in biomass and substrate concentration for unit step changes in all the load variables (disturbance + manipulated variables) as mentioned in eq. (2.34). The disturbance rejections of conventional turbidostat and nutristat & concentration turbidostat and nutristat in closed loop are shown in Figure 2.19 through Figure 2.38, respectively.

Table 2.1: Parameters for substrate inhibition kinetics & steady state values of manipulated variables

Disturbances	Value
μ_{max}	0.53 h^{-1}
k_m	0.12 g/L
K_1	0.4545 L/g
Y	0.4
x_{2fs}	4.0 g/L
D_s	0.3 h^{-1}

Table 2.2: Disturbance Rejection performance by closed loop bioreactor configurations

Configurations/ Disturbances	D—>X	D—>S	x_{2fs}—>X	x_{2fs}—>S
μ_{max}	0.4711	8.3554	0.3551	-0.239
k_m	-3.15	10.3195	-3.021	-0.392
Y	1.2874	-5.6397	1.0027	0.5639
D_s	-0.608	-6.0281	-0.458	1.4473
x_{2fs}	0.1342	-1.1808	0.1011	0.0035

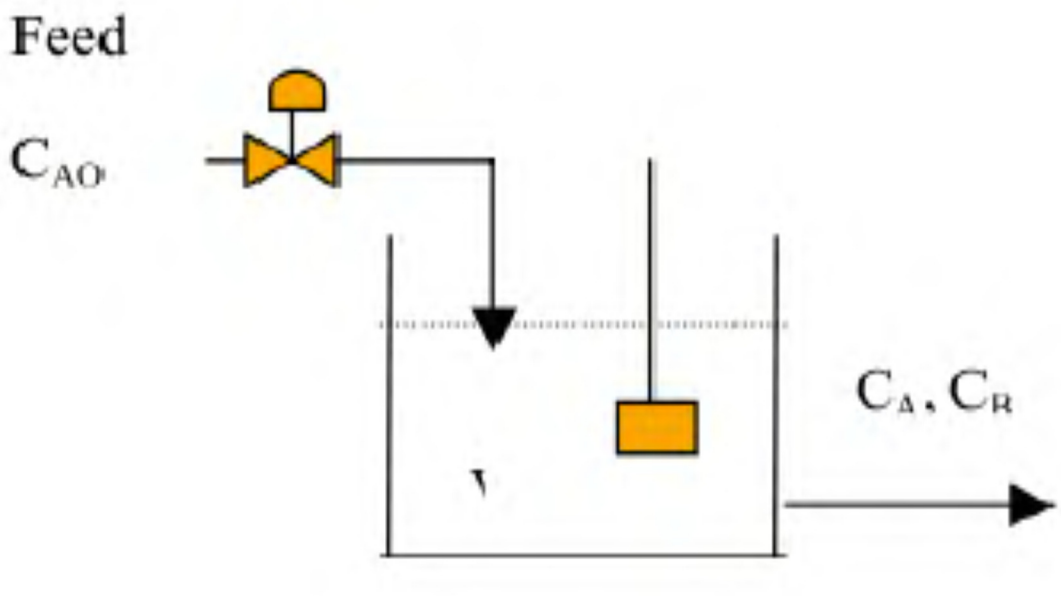

Figure 2.1: Schematic of CSTR

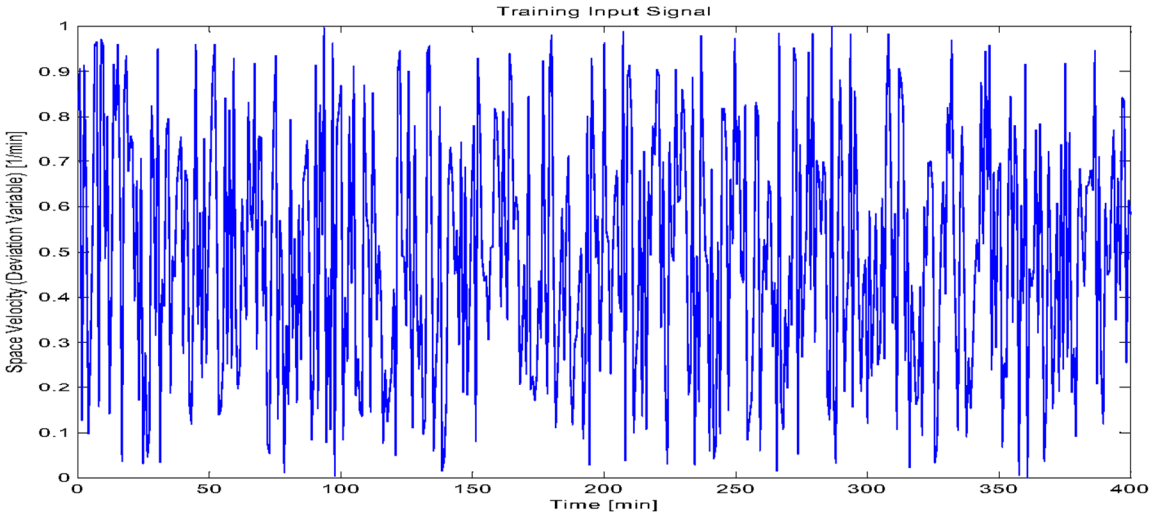

Figure 2.2: Input PSR for training network

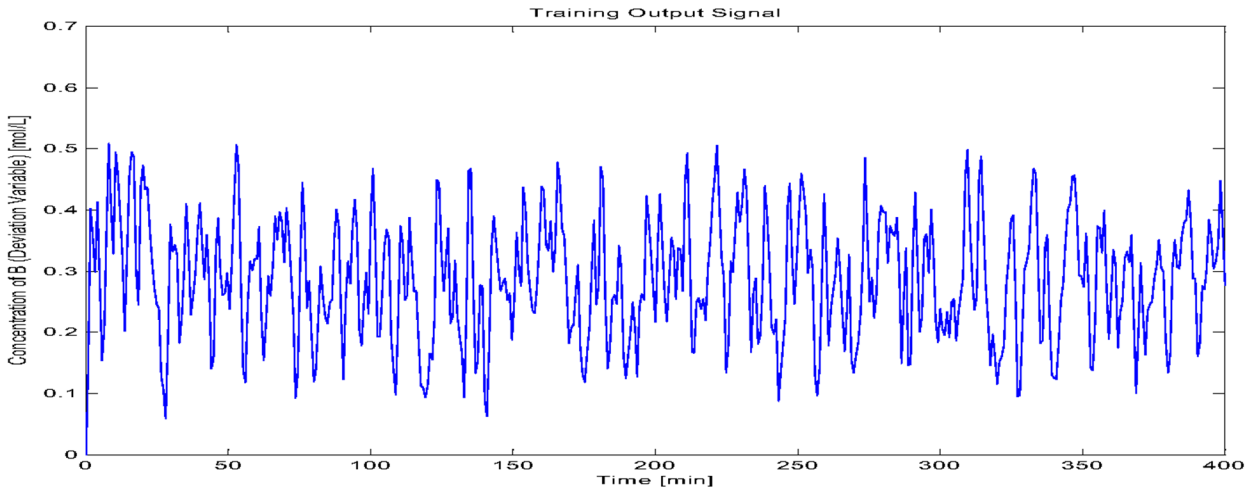

Figure 2.3: System output (Target) for training

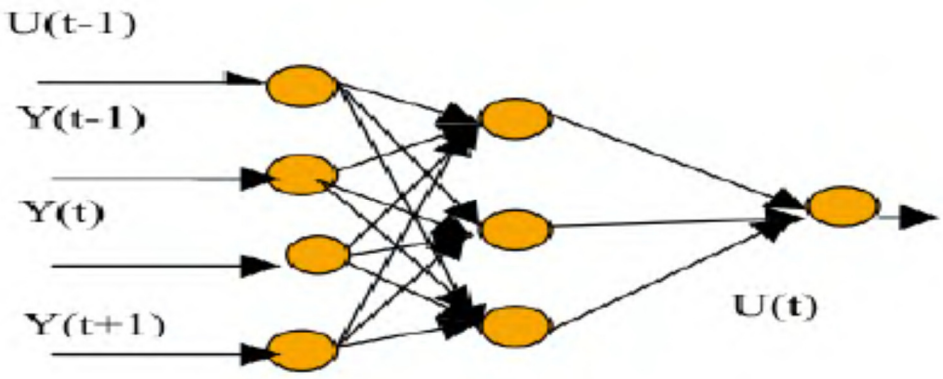

Figure 2.4: DINN structure at completion of training

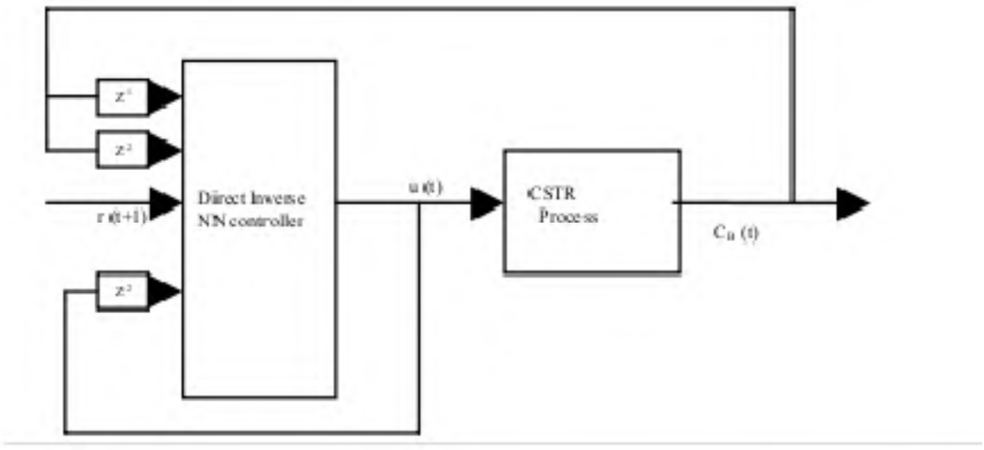

Figure 2.5: Closed loop block for Servo configuration

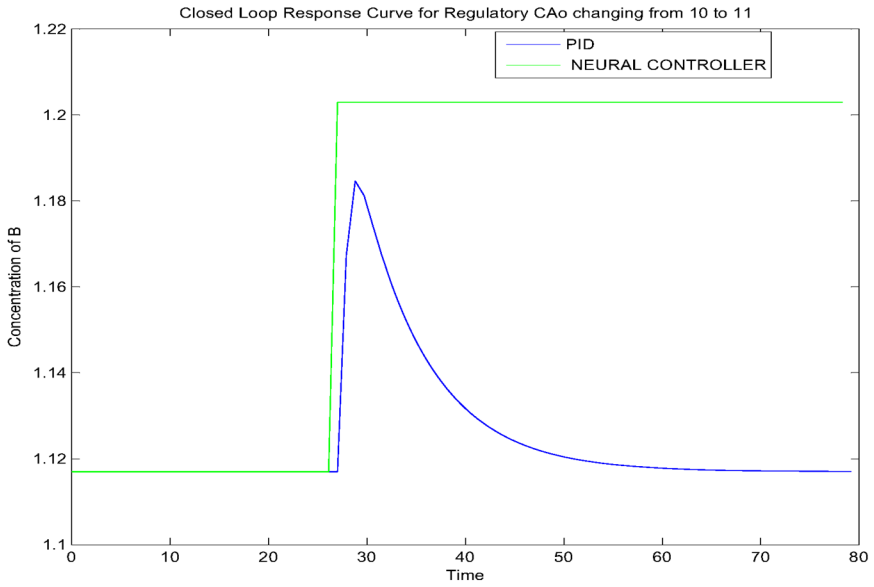

Figure 2.6: Disturbance rejection of DINN controller

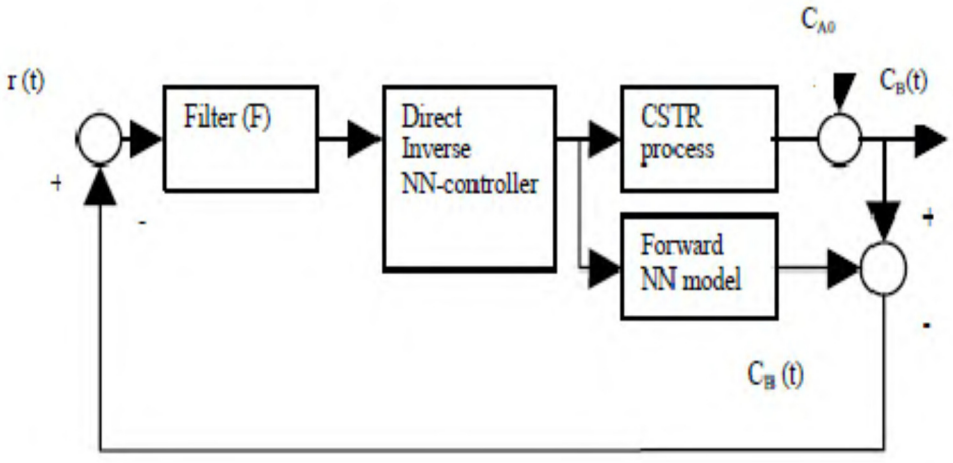

Figure 2.7: Closed loop block for Regulatory configuration

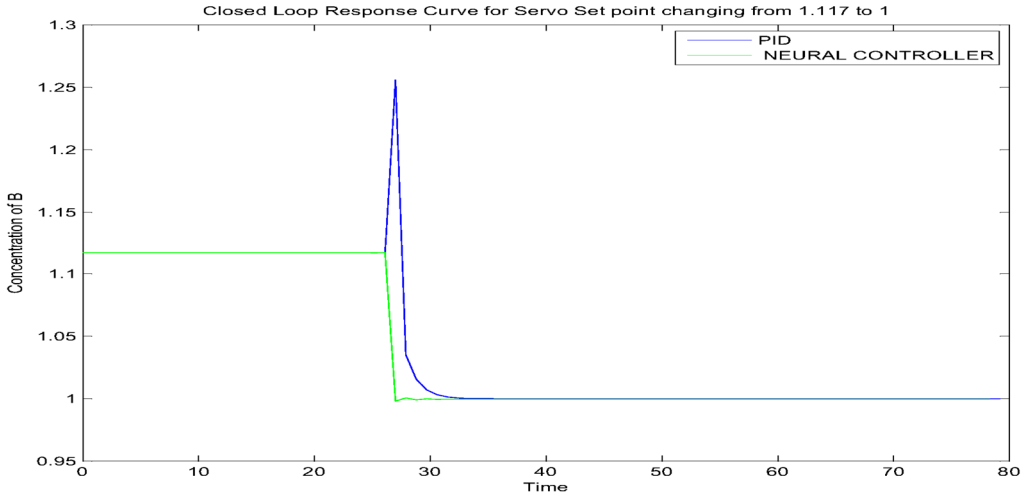

Figure 2.8: Servo response of DINN controller

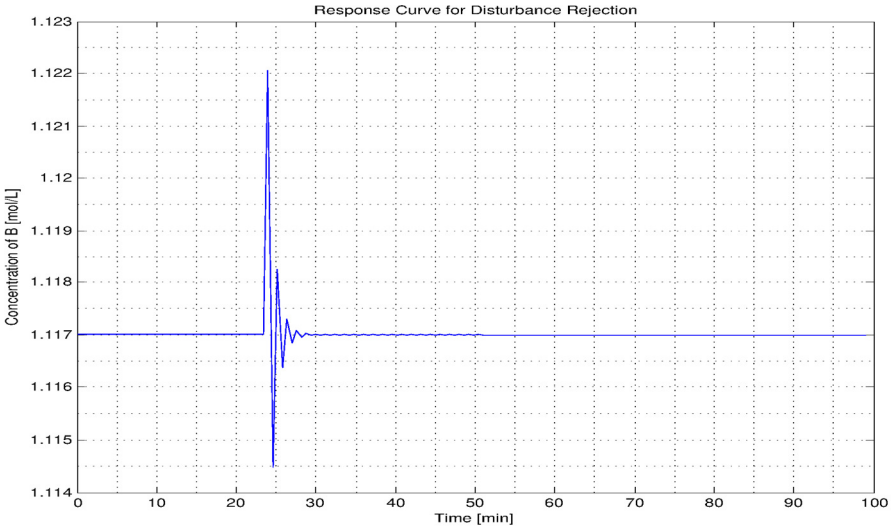

Figure 2.9: Regulatory response of ANN based IMC

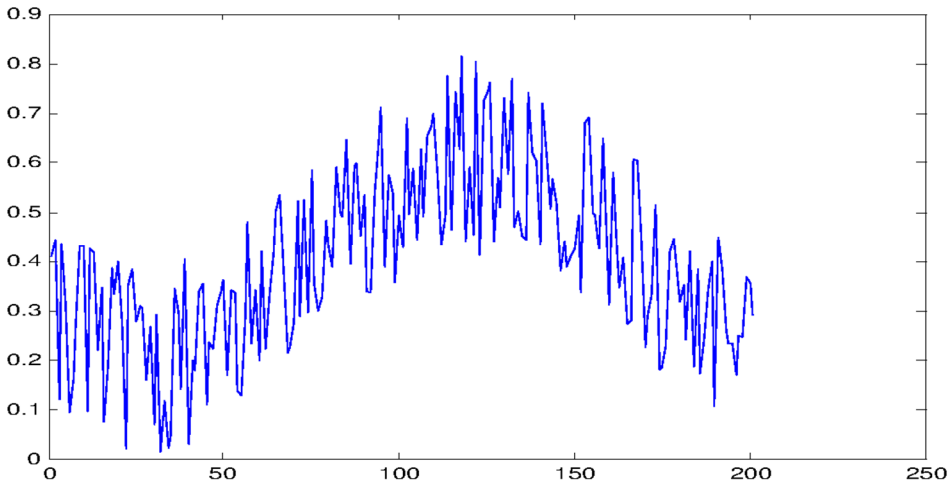

Figure 2.10: Input PRBS for training

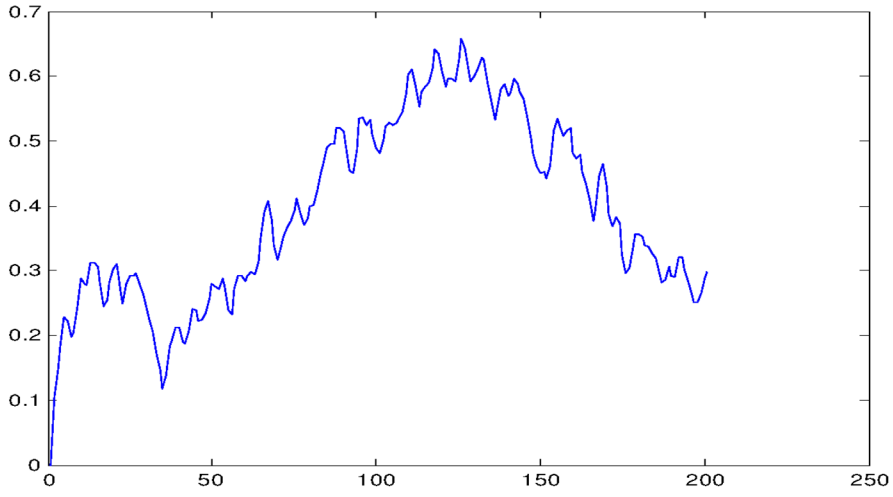

Figure 2.11: Output of system (target) for training

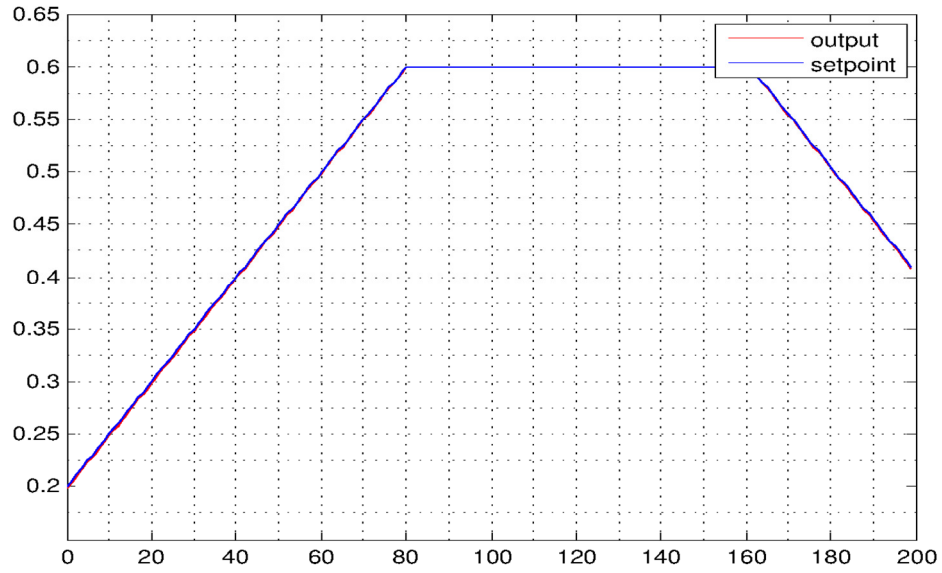

Figure 2.12: Setpoint tracking of DINN

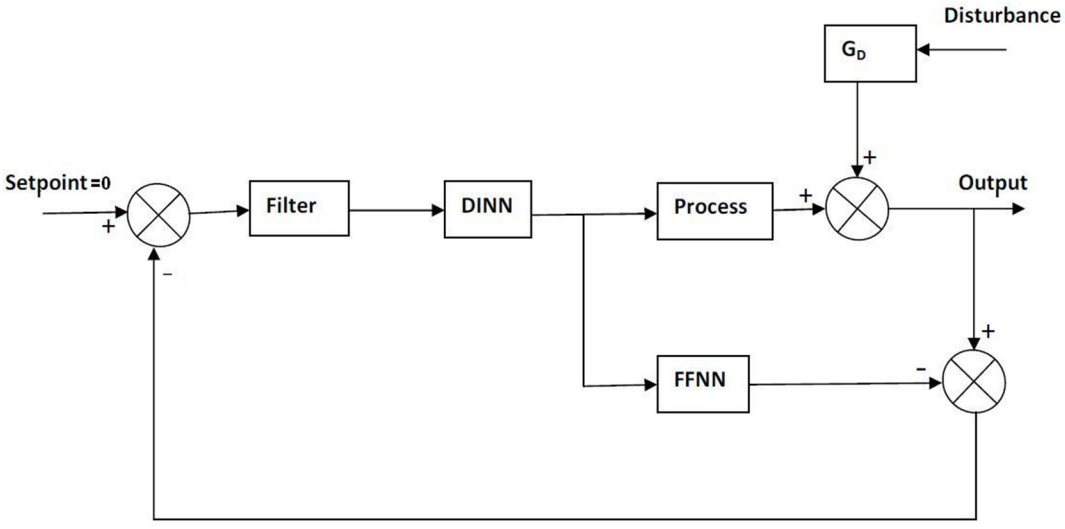

Figure 2.13: Block diagram of closed loop IMC scheme for Biorector

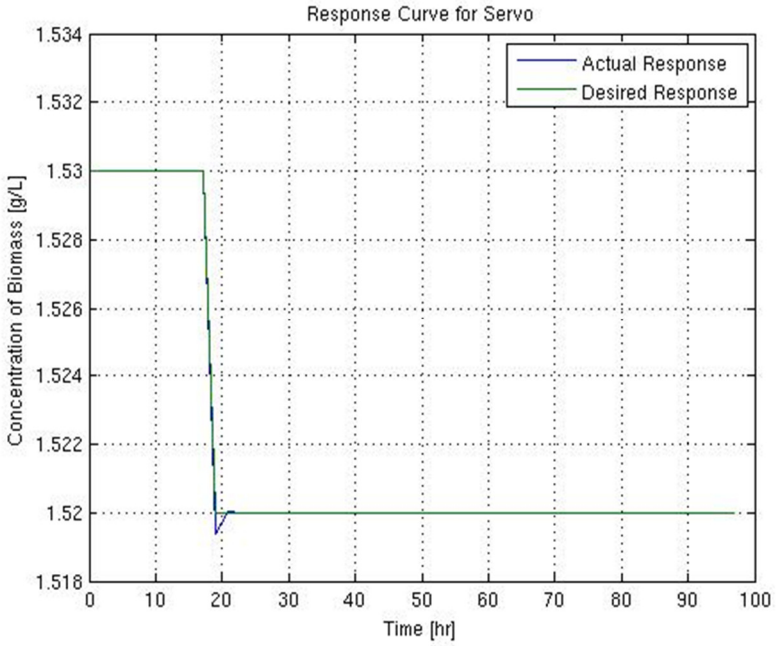

Figure 2.14: Servo response of DINN in (D-X) configuration (Sampling Time=0.8hrs)

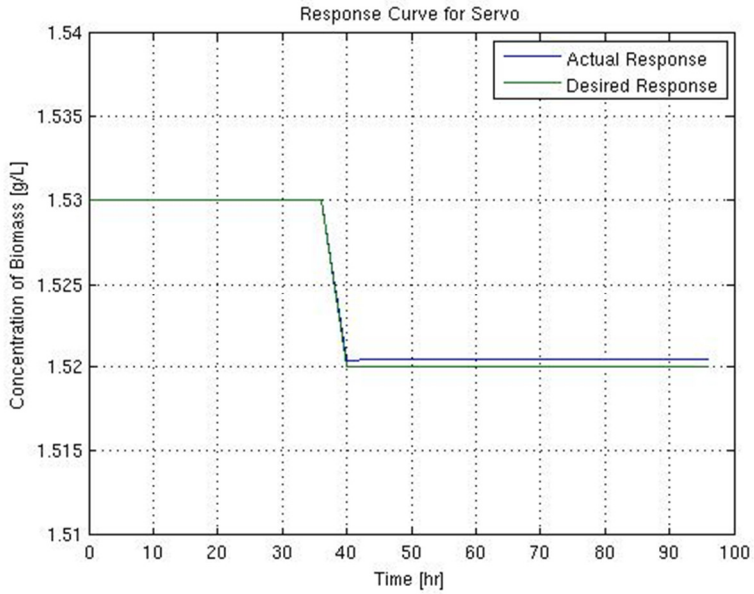

Figure 2.15: Servo response of DINN in (D-X) configuration (Sampling Time=2hrs)

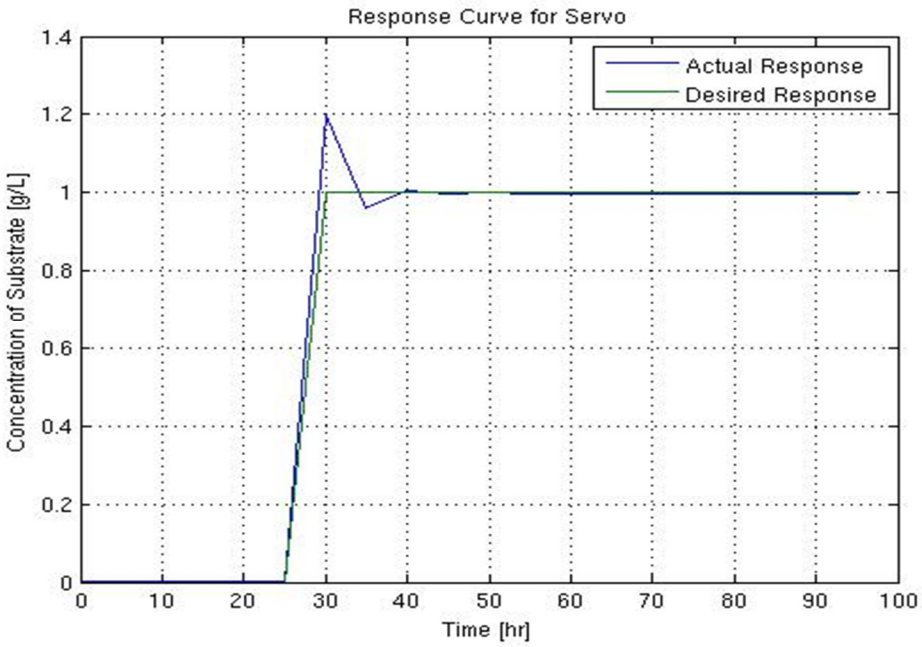

Figure 2.16: Servo response of DINN in (D-S) configuration (Sampling Time=2hrs)

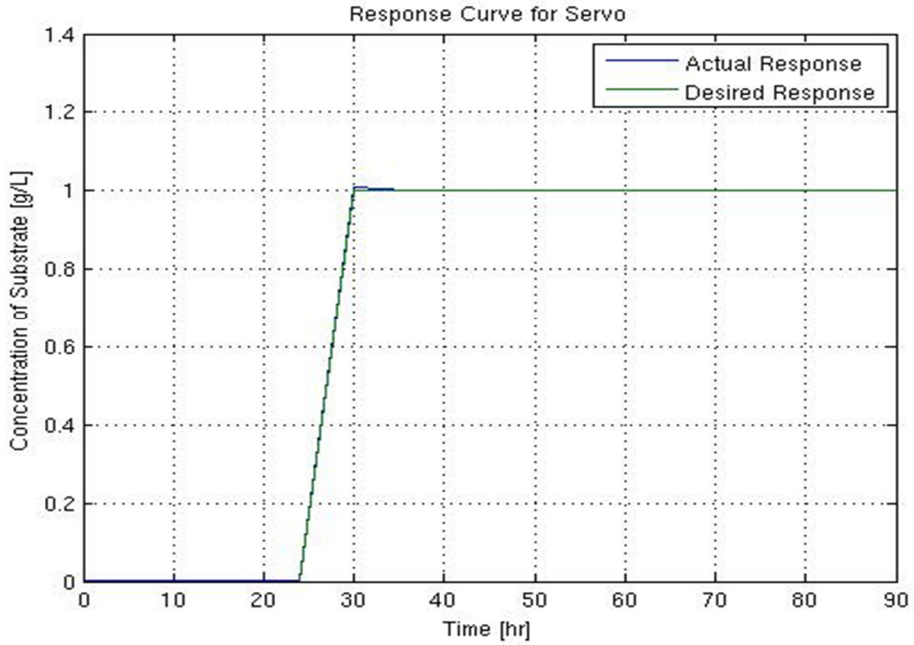

Figure 2.17: Servo response of DINN in (D-S) configuration (Sampling Time=0.8hrs)

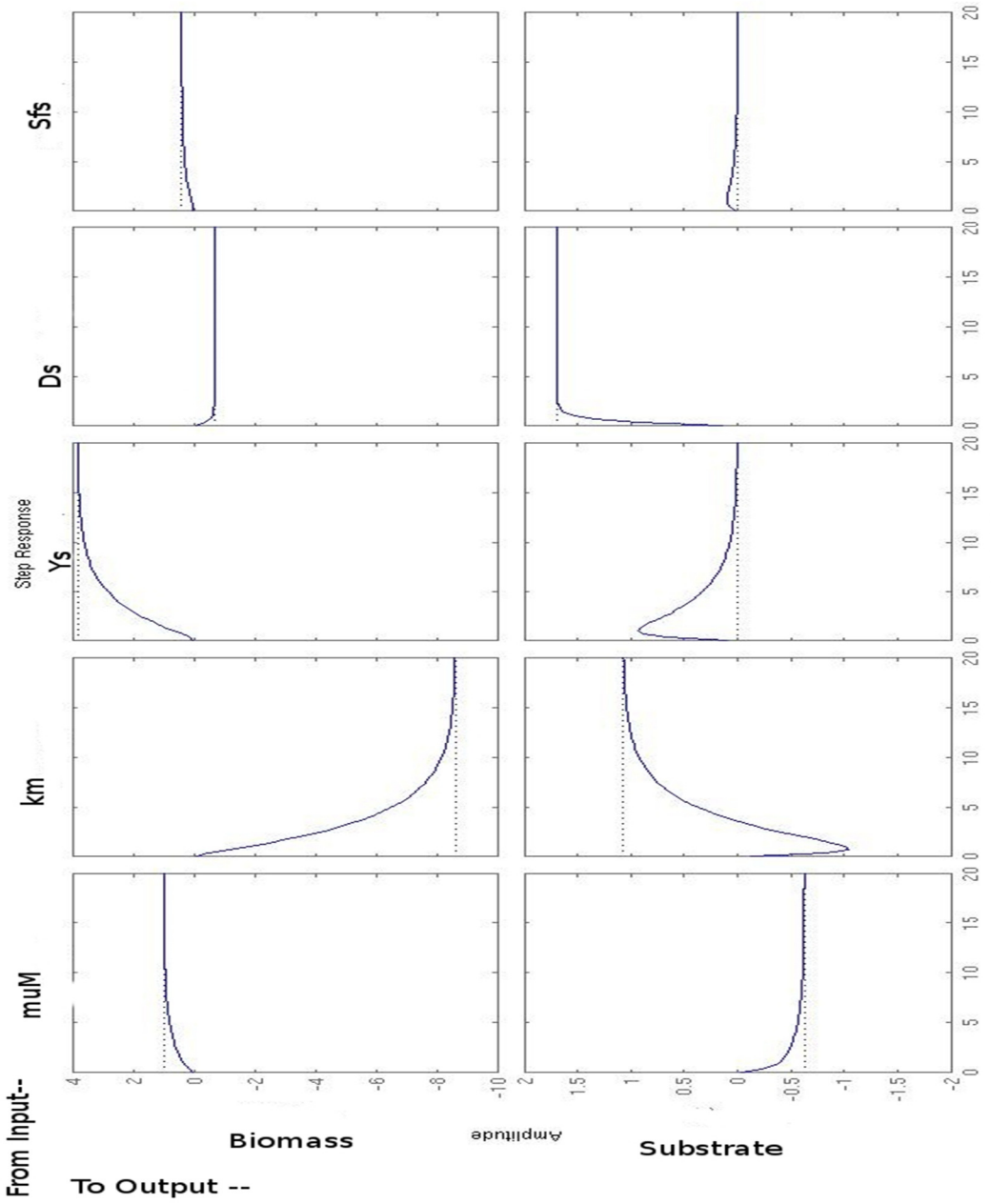

Figure 2.18: Open Loop Respose

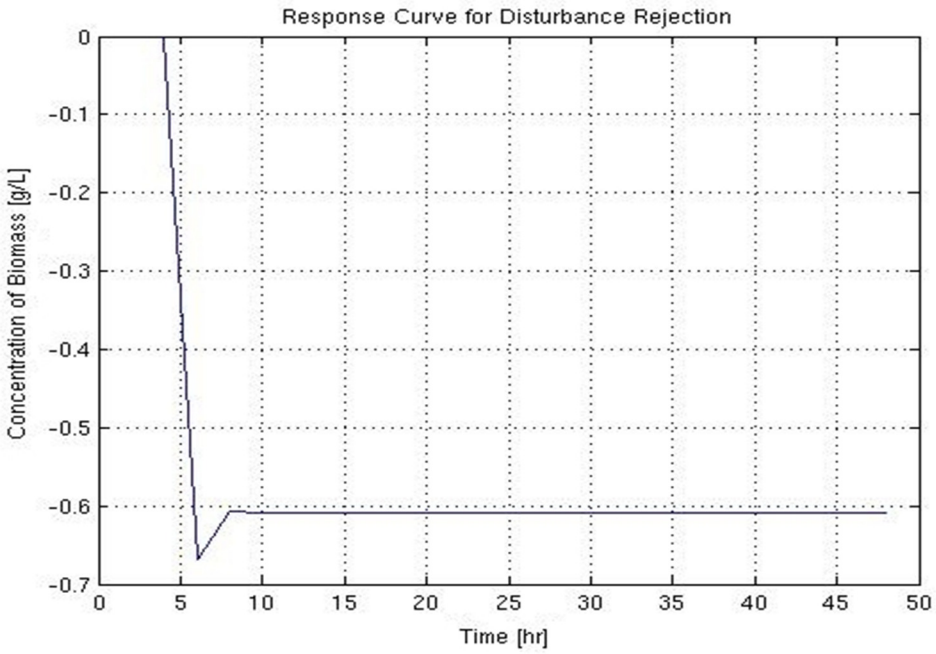

Figure 2.19: Rejection of disturbance in Ds by NN-IMC in (D-X) configuration

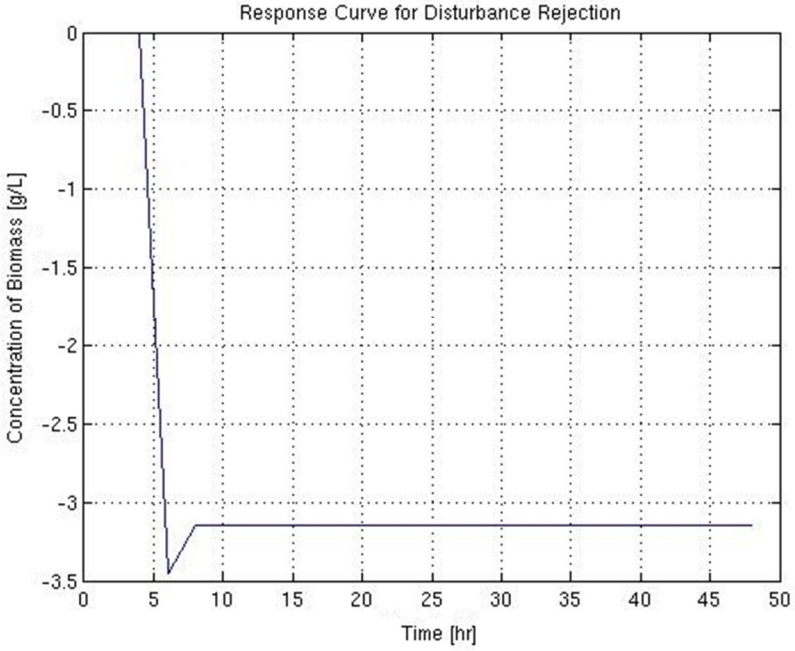

Figure 2.20: Rejection of disturbance in km by NN-IMC in (D-X) configuration

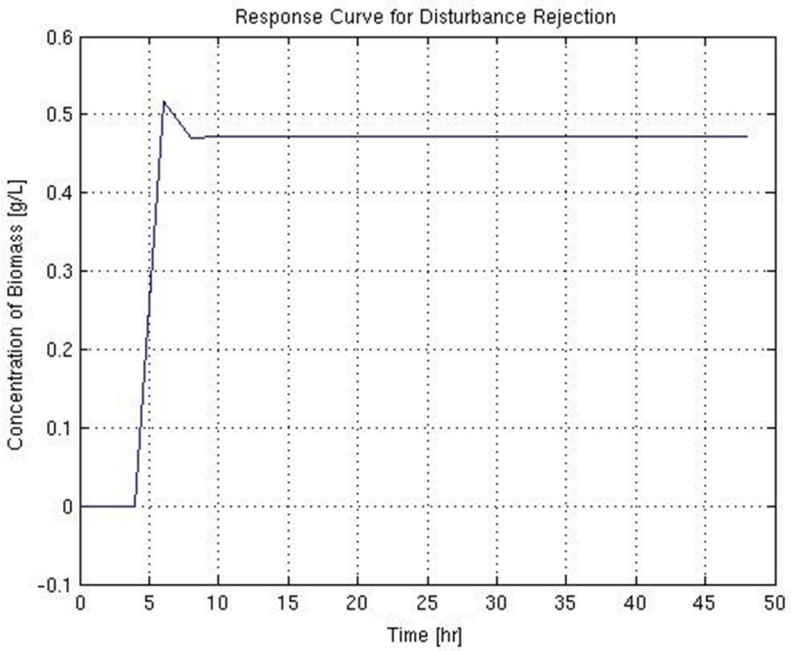

Figure 2.21: Rejection of disturbance in by NN-IMC in (D-X) configuration

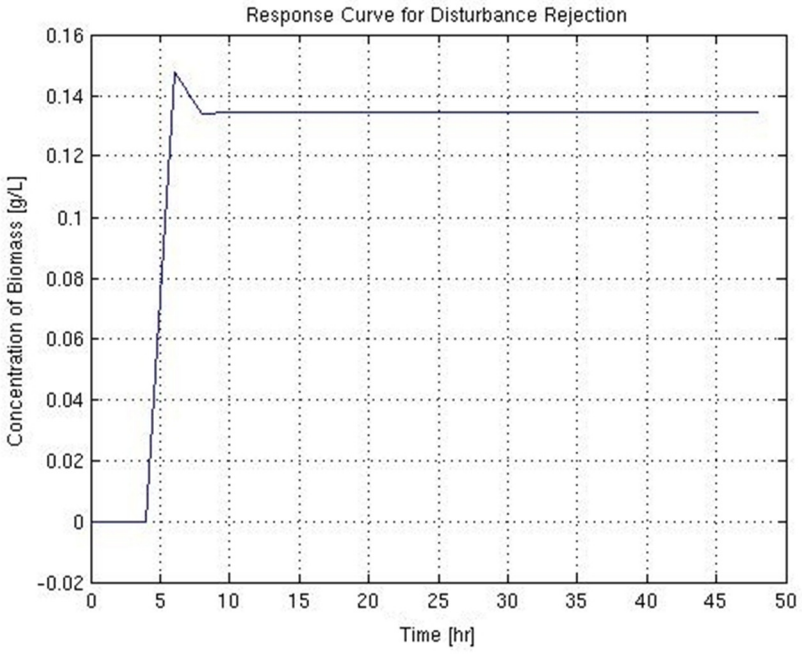

Figure 2.22: Rejection of disturbance in Sf by NN-IMC in (D-X) configuration

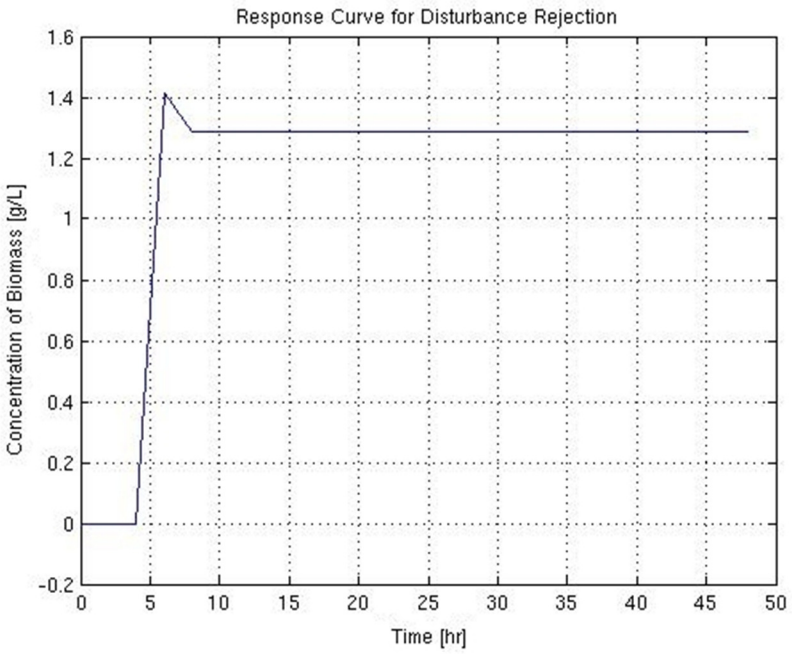

Figure 2.23: Rejection of disturbance in Y by NN-IMC in (D-X) configuration

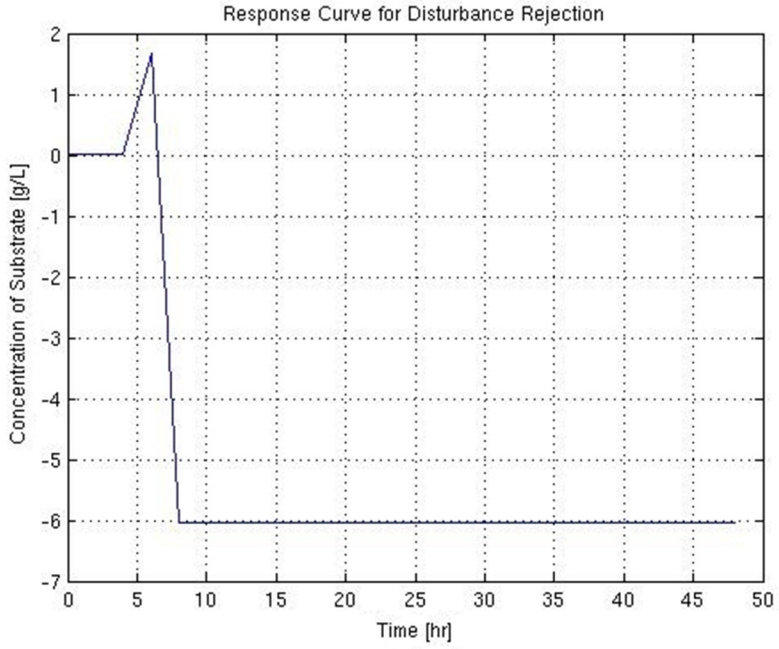

Figure 2.24: Rejection of disturbance in Ds by NN-IMC in (D-S) configuration

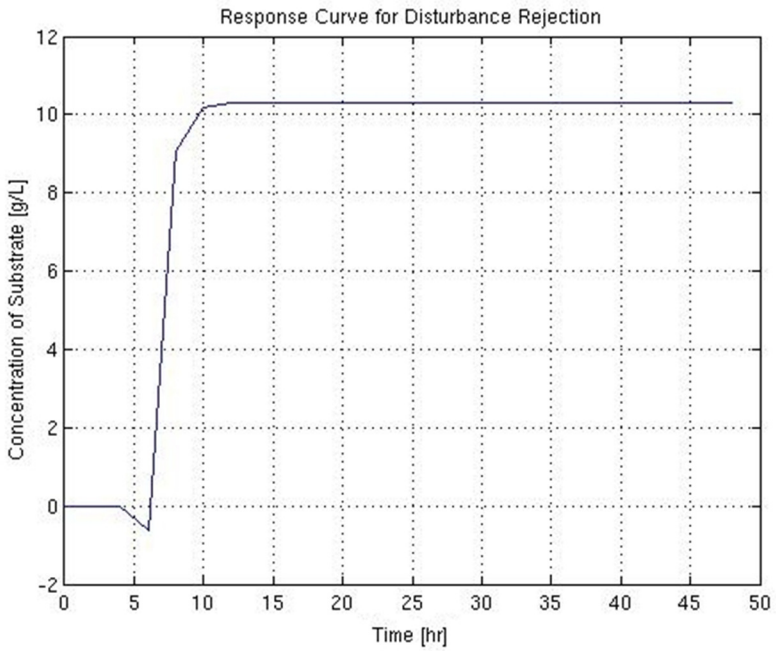

Figure 2.25: Rejection of disturbance in km by NN-IMC in (D-S) configuration

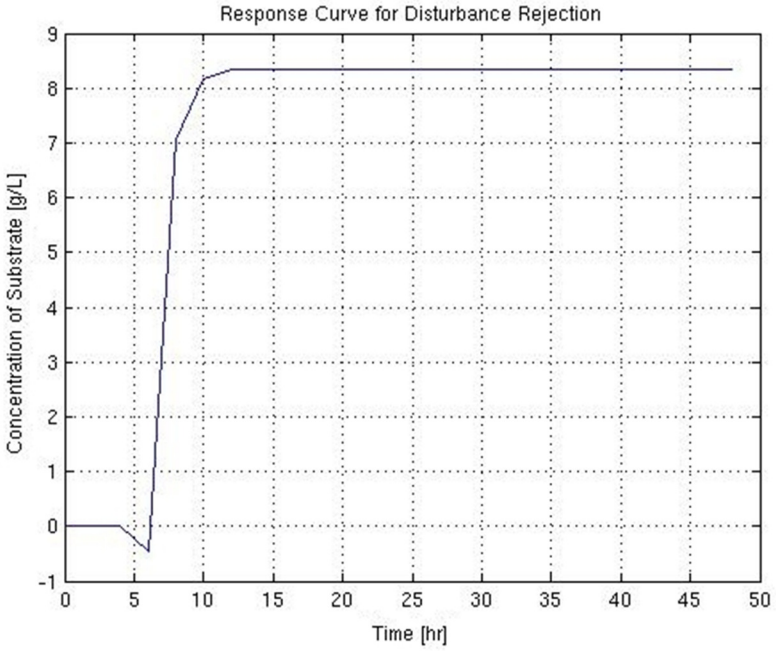

Figure 2.26: Rejection of disturbance in
by NN-IMC in (D-S) configuration

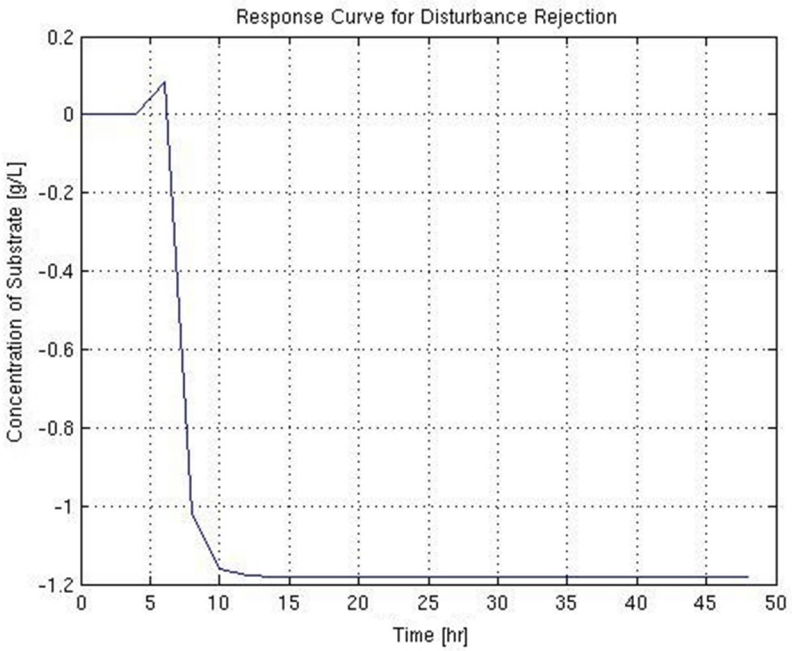

Figure 2.27: Rejection of disturbance in Sf by NN-IMC in (D-S) configuration

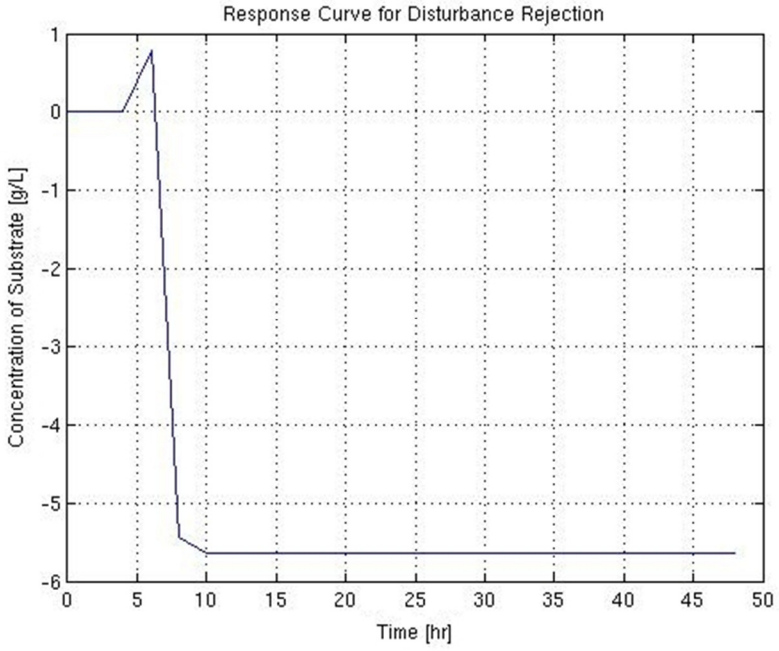

Figure 2.28: Rejection of disturbance in Y by NN-IMC in (D-S) configuration

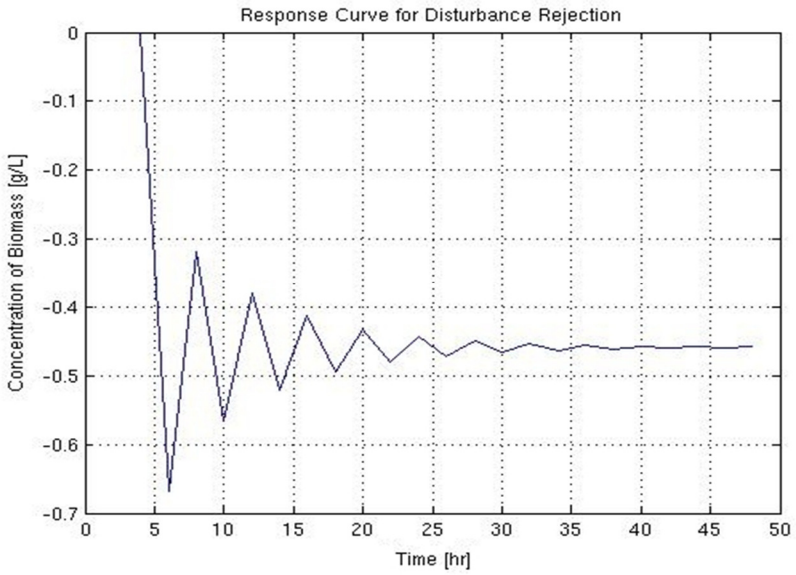

Figure 2.29: Rejection of disturbance in Ds by NN-IMC in (Sf-X) configuration

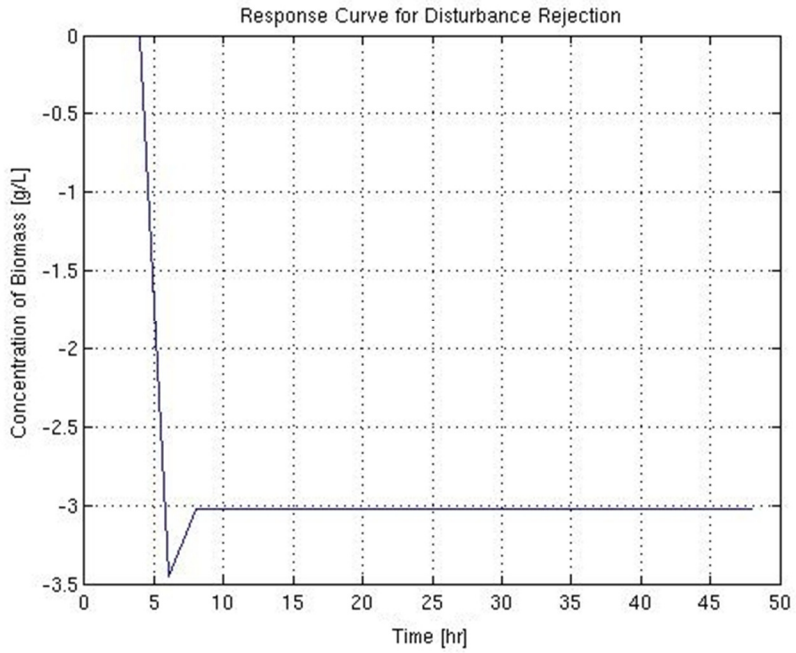

Figure 2.30: Rejection of disturbance in km by NN-IMC in (Sf-X) configuration

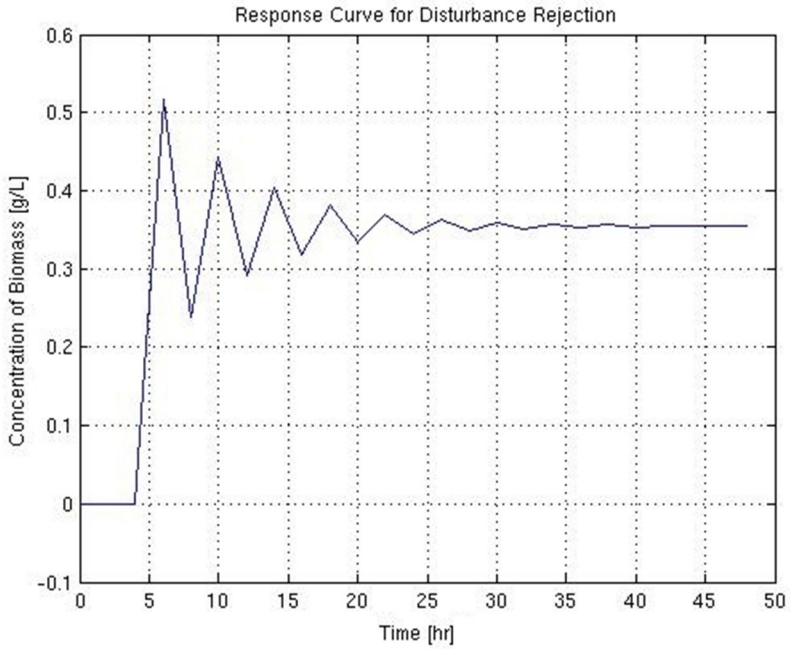

Figure 2.31: Rejection of disturbance in
by NN-IMC in (Sf-X) configuratio

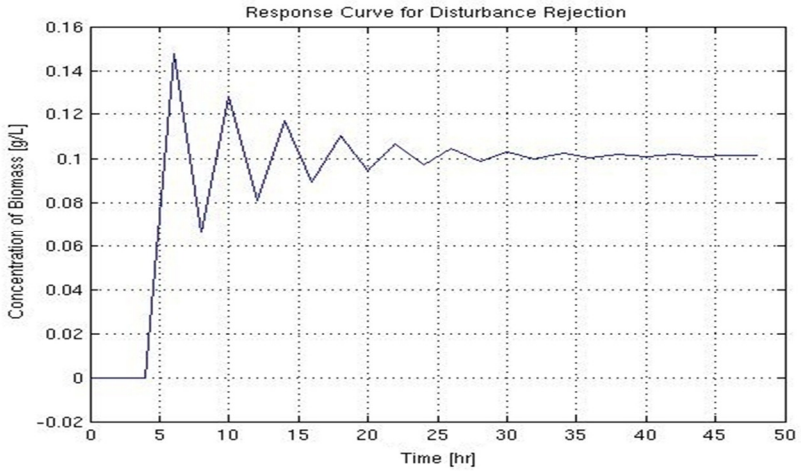

Figure 2.32: Rejection of disturbance in Sf by NN-IMC in (Sf-X) configuration

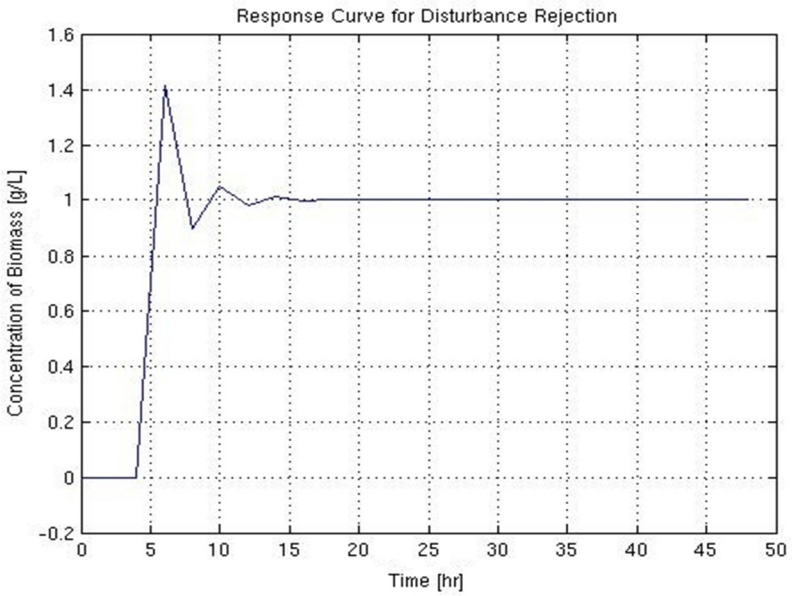

Figure 2.33: Rejection of disturbance in Y by NN-IMC in (Sf-X) configuration

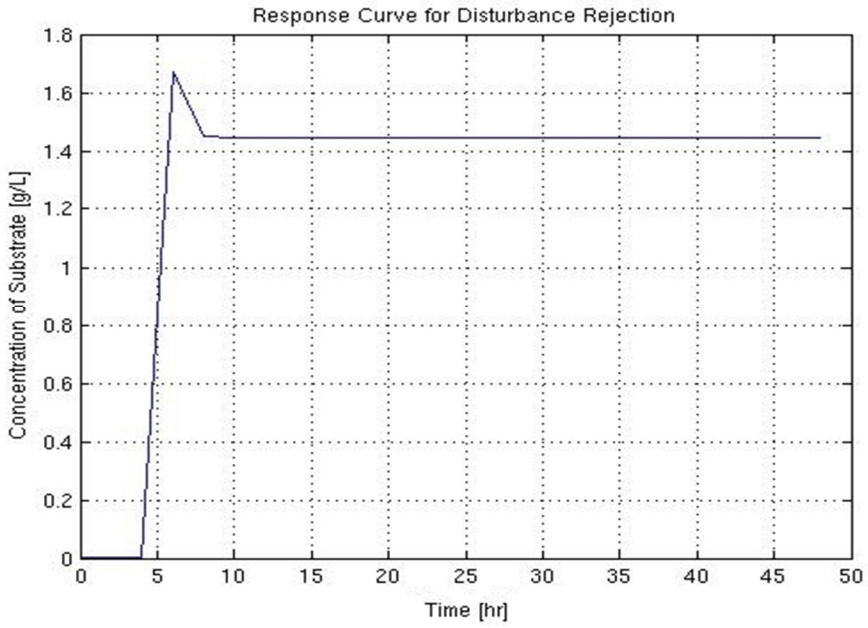

Figure 2.34: Rejection of disturbance in Ds by NN-IMC in (Sf-S) configuration

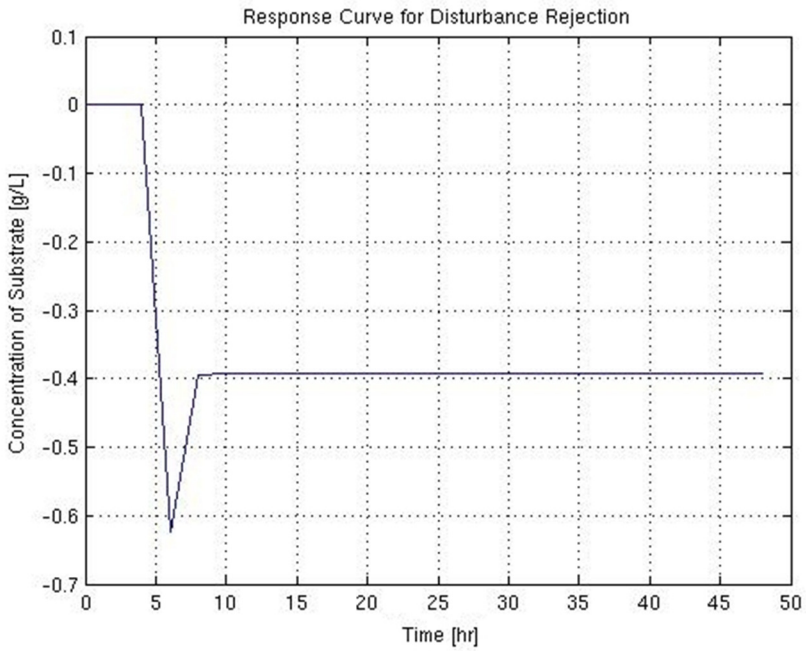

Figure 2.35: Rejection of disturbance in km by NN-IMC in (Sf-S) configuration

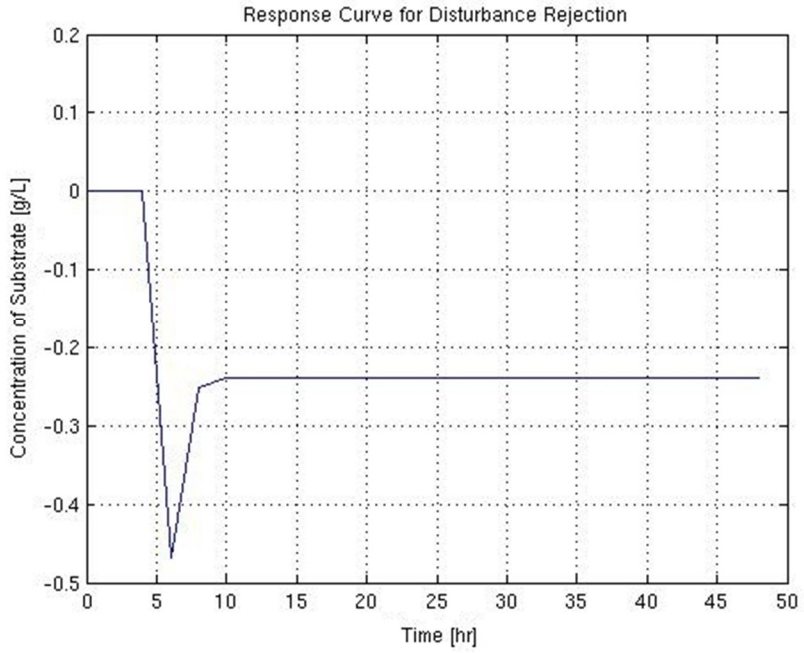

Figure 2.36: Rejection of disturbance in
by NN-IMC in (Sf-S) configuration

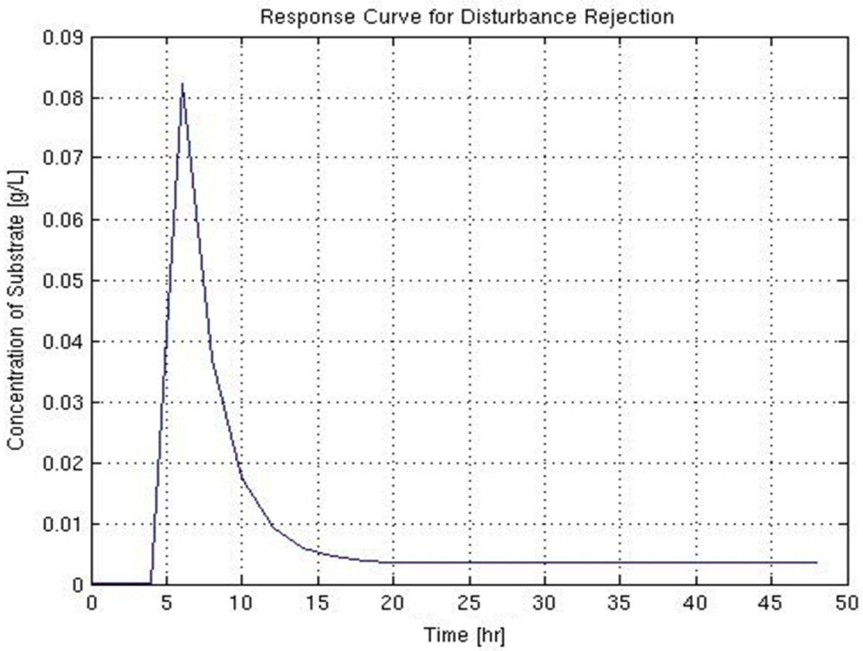

Figure 2.37: Rejection of disturbance in S_f by NN-IMC in (Sf-S) configuration

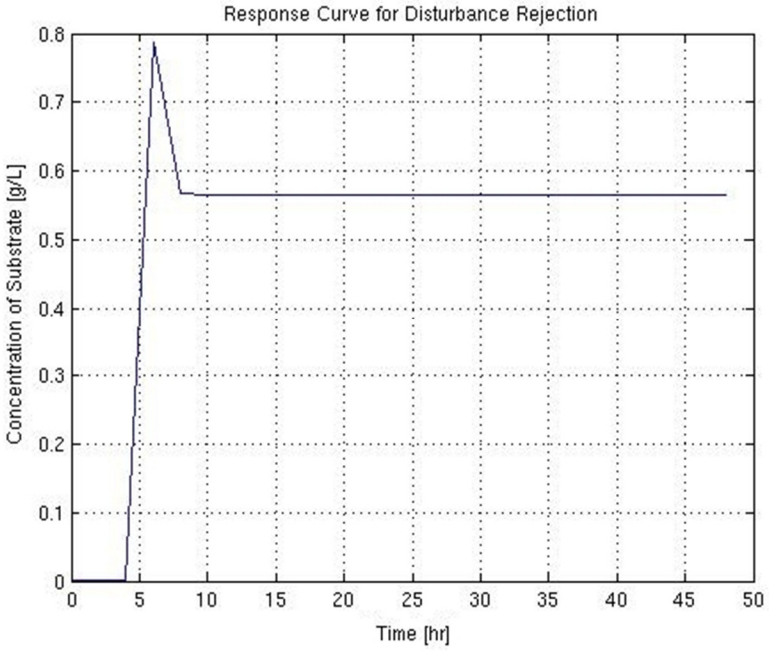

Figure 2.38: Rejection of disturbance in Y by NN-IMC in (Sf-S) configuration

CHAPTER 3

PARTIAL LEAST SQUARE BASED IDENTIFICATION OF PROCESS DYNAMICS

Partial Least Square Based Identification of Process Dynamics

3.1 Problem Specification

An attempt on modeling a distillation column for separating methanol-water was made.(Wood, 1973) The composition of top and bottom products is the controlled variable both expressed in weight percentage of methanol and the manipulated inputs are the reflux and reboiler steam flow rates (lb/min). Time is in minutes. They gave the following transfer function

$$\begin{bmatrix} y_1(s) \\ y_2(s) \end{bmatrix} = \begin{pmatrix} \dfrac{12.8e^{-s}}{16.7s+1} & \dfrac{-18.9e^{-3s}}{21s+1} \\ \dfrac{6.6e^{-7s}}{10.9s+1} & \dfrac{-19.4e^{-3s}}{14.4s+1} \end{pmatrix} \begin{bmatrix} x_1(s) \\ x_1(s) \end{bmatrix} \tag{3.1}$$

The relationship of the model was extracted by exciting the plant with a series of step changes to inputs. The signal to noise ratio (SNR) was set to 10 by adding measurement noise. (Lakshminarayanan, 1997) The inputs ranged between +1 and -1. Delay was incorporated in both dimensions of PLS. The scaling procedure was to simply divide each variable dimension by the range of that variable. The scaling matrixes were found to be the following:

$$S_x = \begin{bmatrix} 3.7 & 0 \\ 0 & 3.8264 \end{bmatrix}; \qquad S_y = \begin{bmatrix} 11.315 & 0 \\ 0 & 18.163 \end{bmatrix} \tag{3.2}$$

The values of loading matrixes are

$$P = \begin{bmatrix} 0.6939 & -0.9118 \\ 0.7201 & 0.4107 \end{bmatrix}; \quad Q = \begin{bmatrix} 0.6093 & 0.9276 \\ 0.7929 & 0.3736 \end{bmatrix} \tag{3.3}$$

3.2 Results & Discussion

The open loop simulation of the system is shown in Figure 3.1. The inner relation was first developed using ARX second order model. The prediction is given in Figure 3.2 and Figure 3.3. The fit for idealized decoupled PLS model case is shown in Figure 3.4 through Figure 3.7.

The input and output scores were correlated by $^{4\ th}$ order polynomial to describe the static non-linearity of the dynamic. The resulted scores were then correlated by linear least squares as a part of the dynamic inner model. The input matrix used by the least square technique consisted of lagged inputs-outputs (historical data up to previous 4 time intervals). The PLS model simulations are given in Figures 3.8 through Figure 3.11 with a reasonable accuracy.

The incorporation of a non-linear static element also improved the prediction fit of the least square fit as seen in Figure 3.12 through Figure 3.15

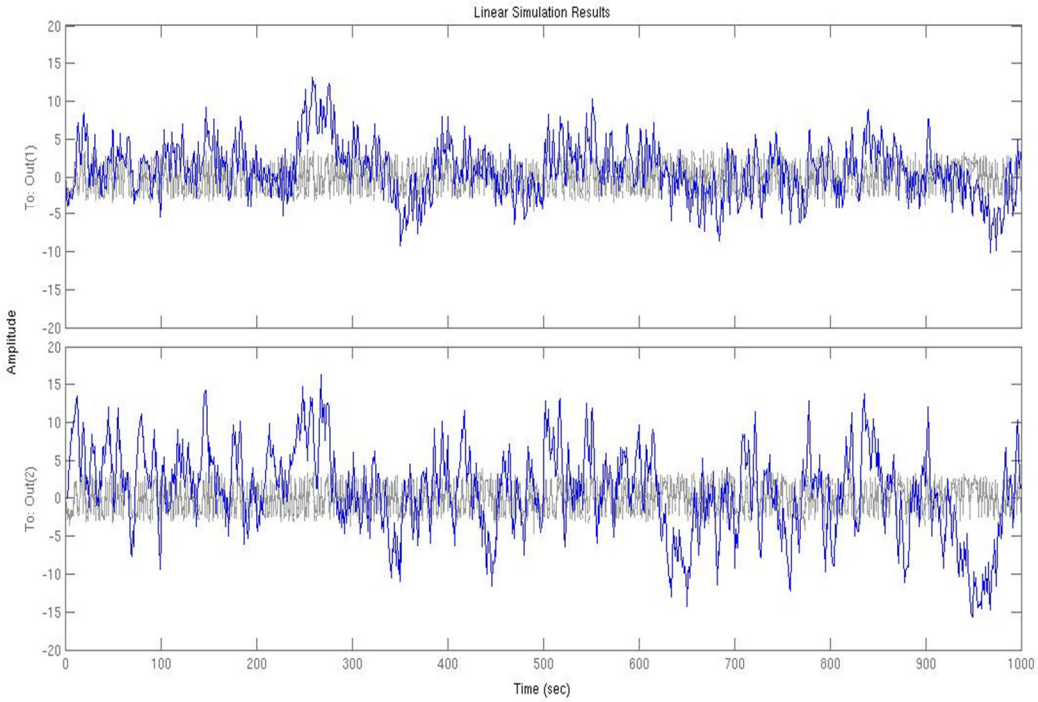

Figure 3.1 Linear simulation of Wood Berry Column model

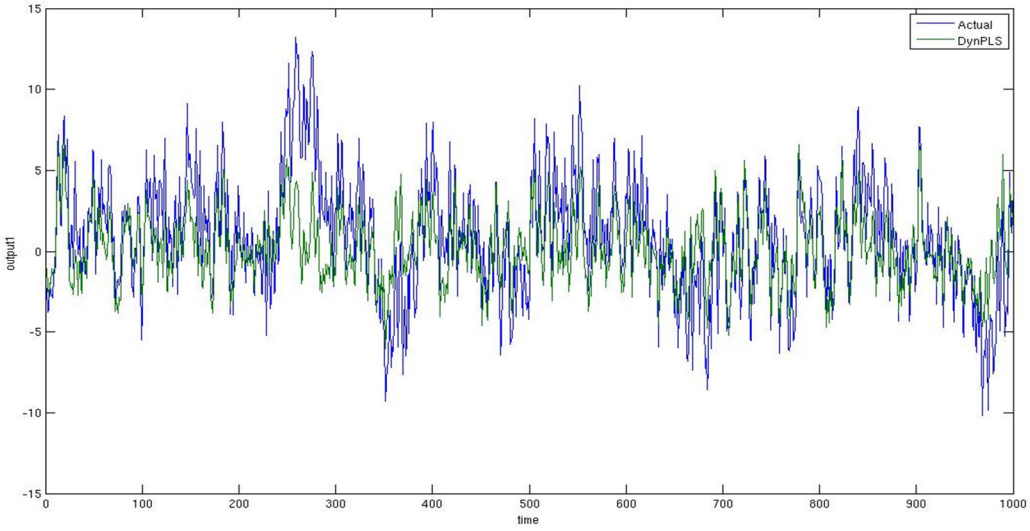

Figure 3.2: Top product composition prediction by PLS(ARX inner model)

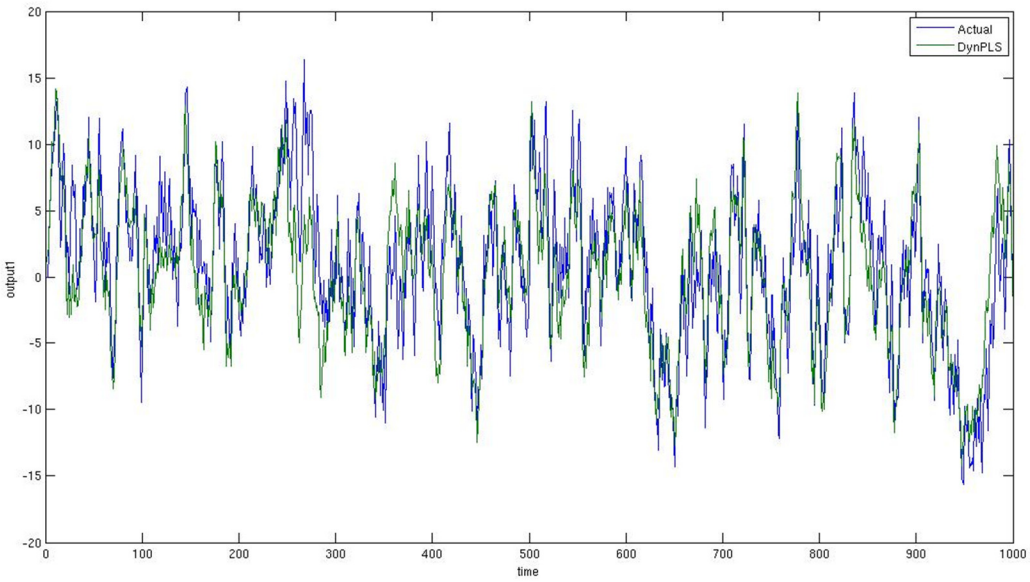

Figure 3.3: Bottom product composition prediction by PLS(ARX inner model)

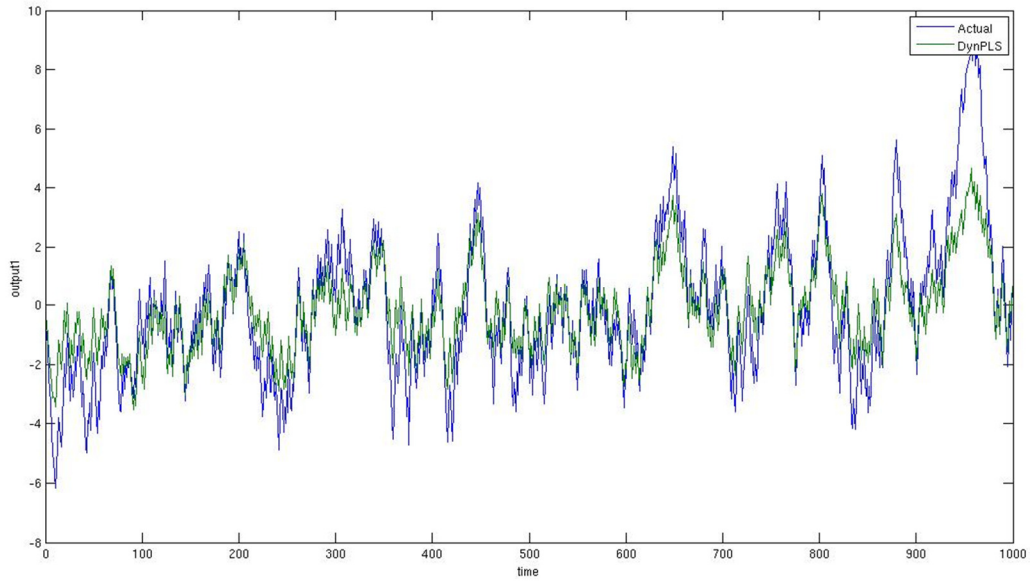

Figure 3.4: Top product composition prediction by PLS (ideal decoupling) (ARX inner model)

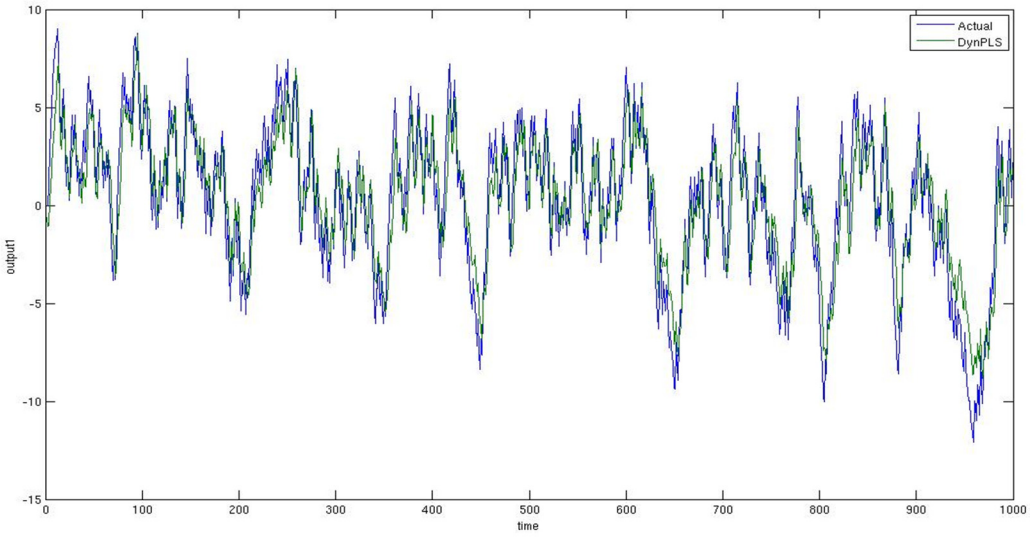

Figure 3.5: Bottom product composition prediction by PLS (ideal decoupling)(ARX inner model)

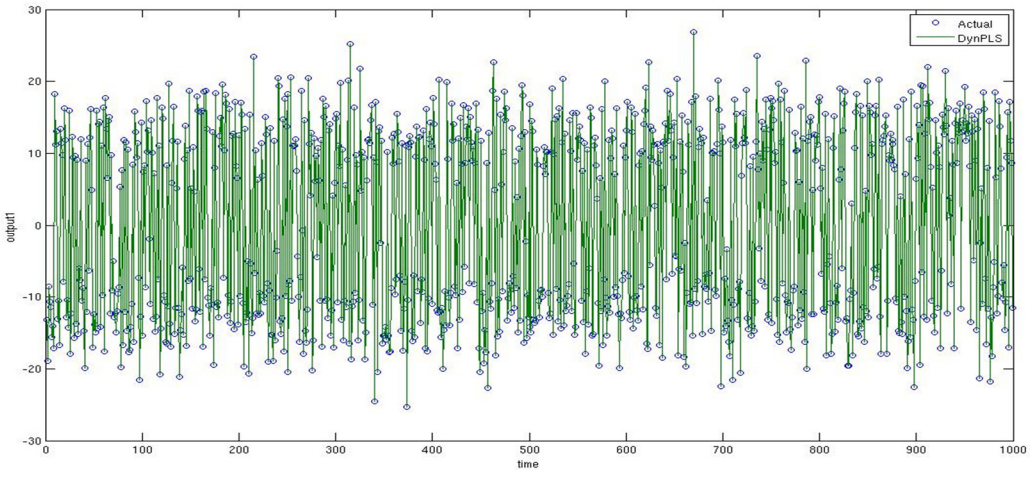

Figure 3.6: Top product composition prediction by PLS (steady state decoupling)(ARX inner model)

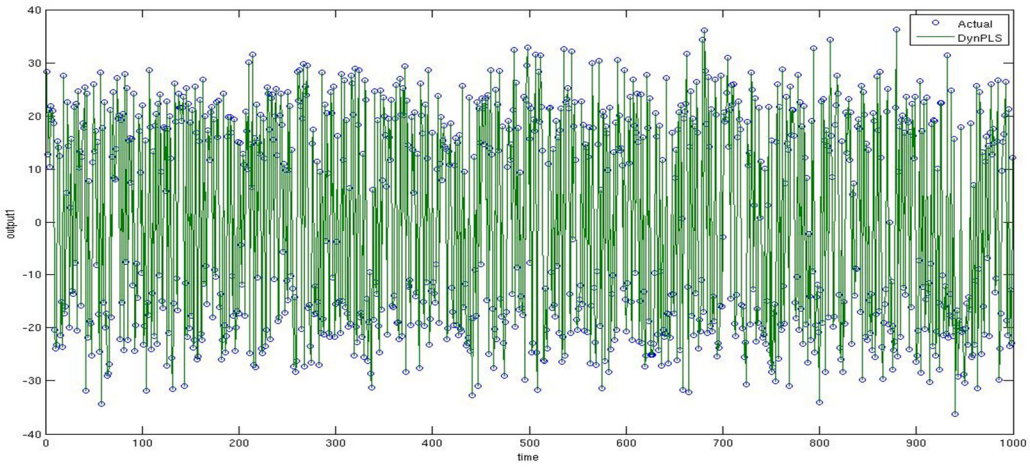

Figure 3.7: Bottom product composition prediction by PLS (steady state decoupling (ARX inner model)

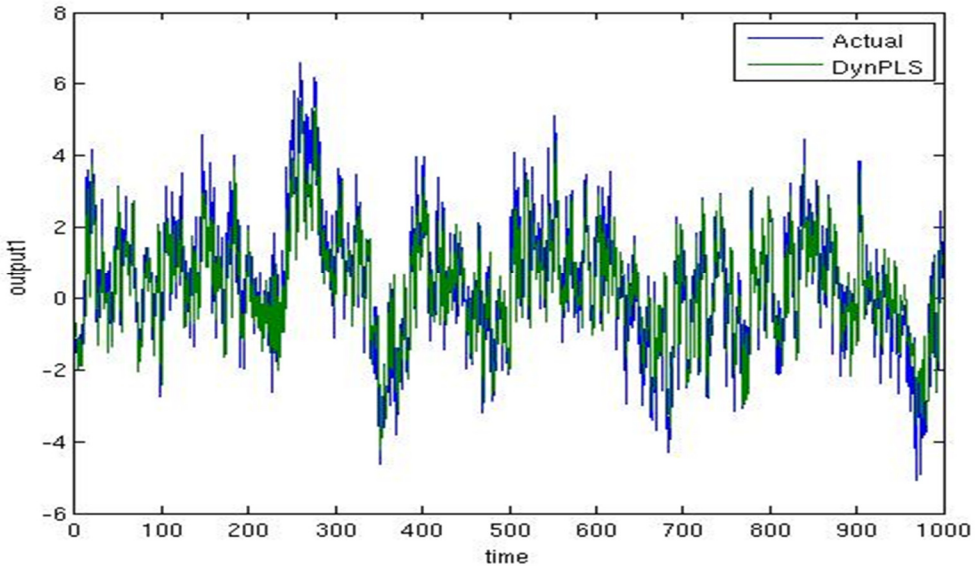

Figure 3.8: Top product composition prediction by PLS (LS inner model)

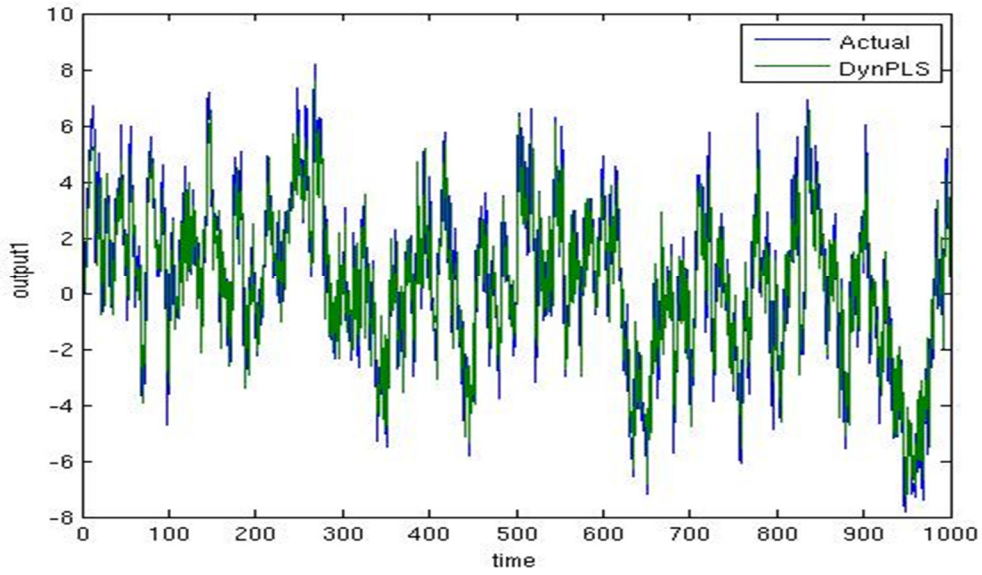

Figure 3.9: Bottom product composition prediction by PLS (LS inner model)

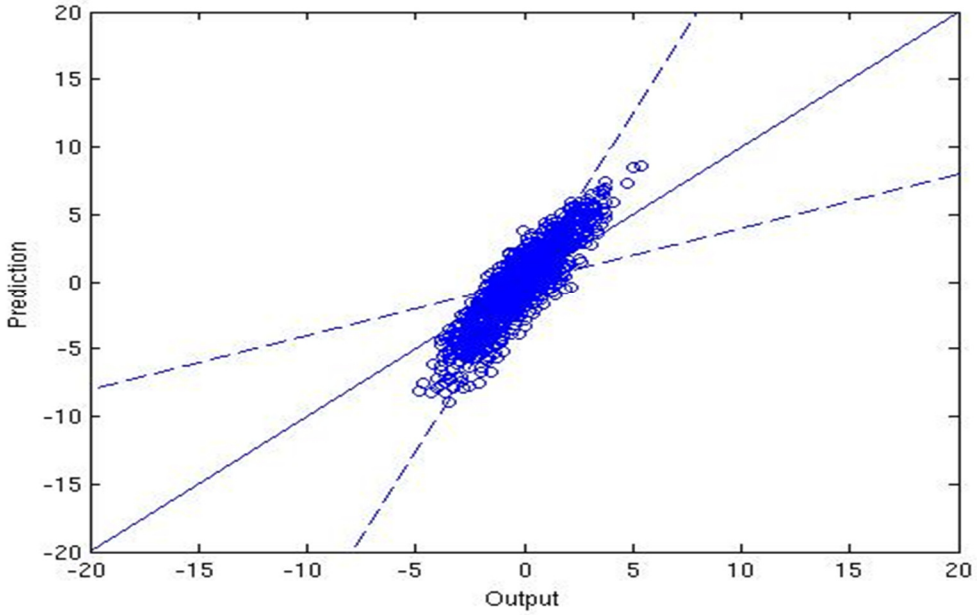

Figure 3.10 Data fit for Top product composition

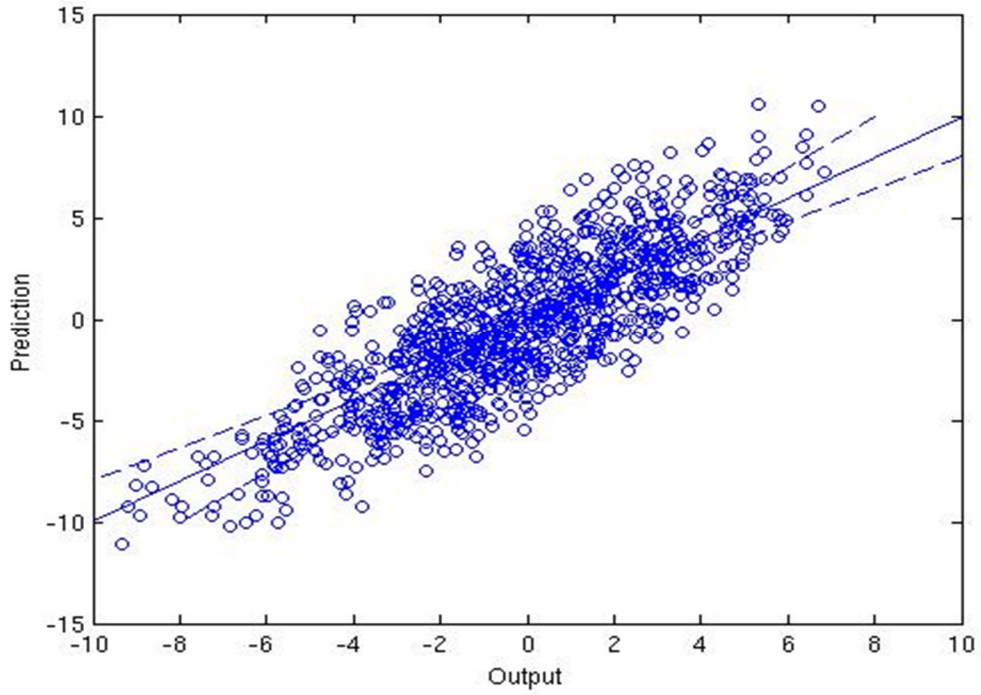

Figure 3.11 Data fit for Bottom product composition

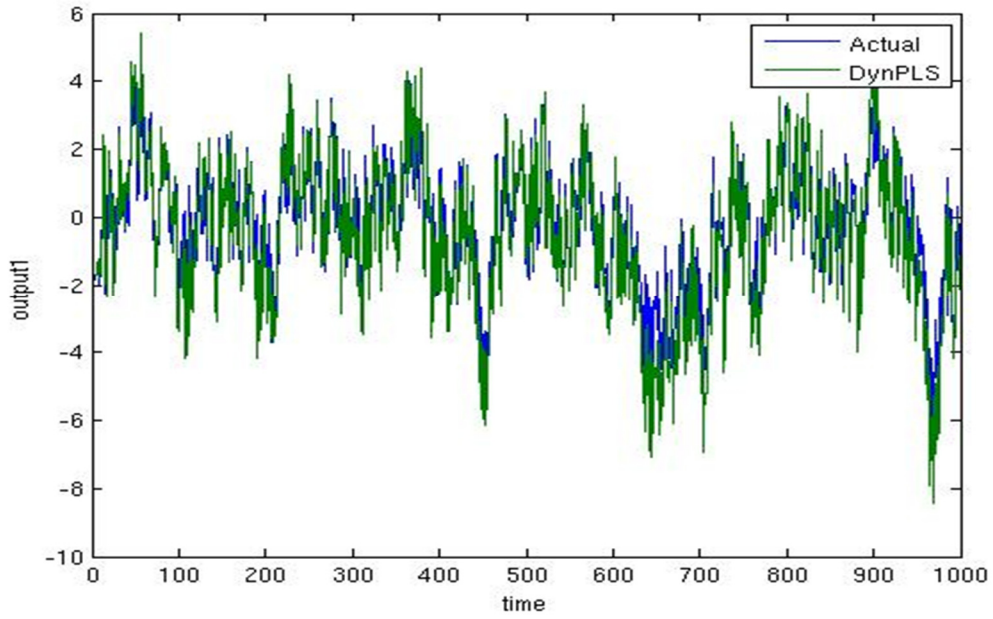

Figure 3.12 Top product composition prediction by PLS (LS inner model with nonlinear static element)

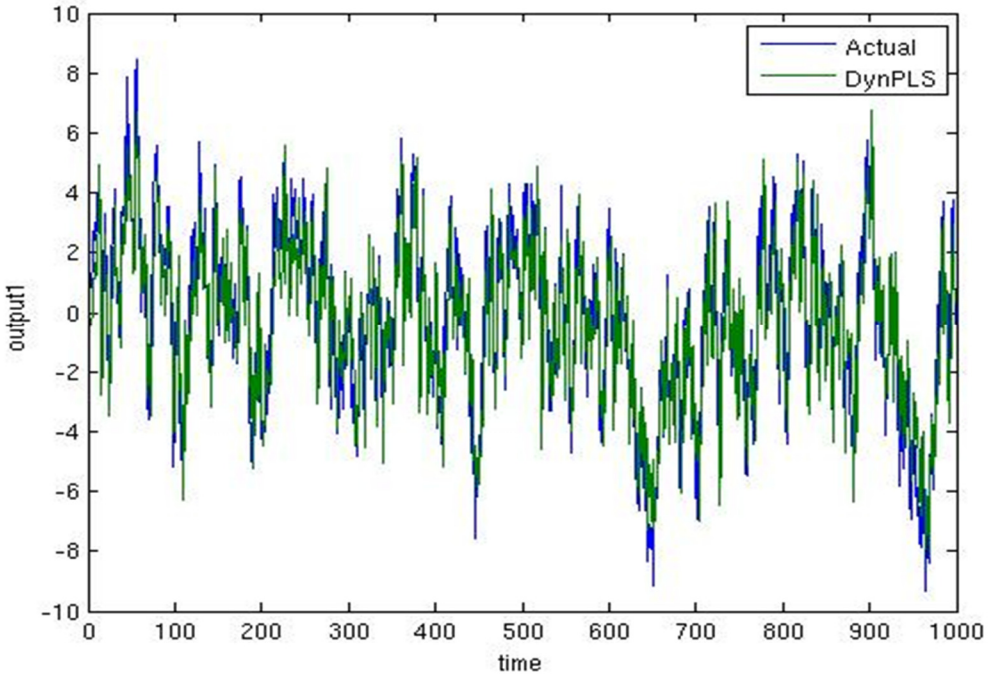

Figure 3.13 Bottom product composition prediction by PLS (LS inner model with nonlinear static element)

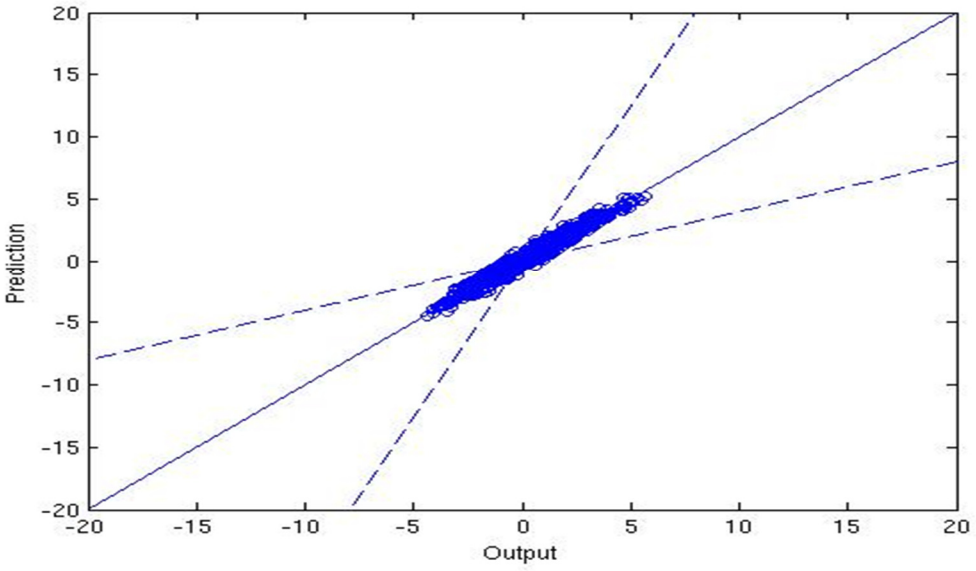

Figure 3.14 Data fit for Top product composition (with nonlinear static element))

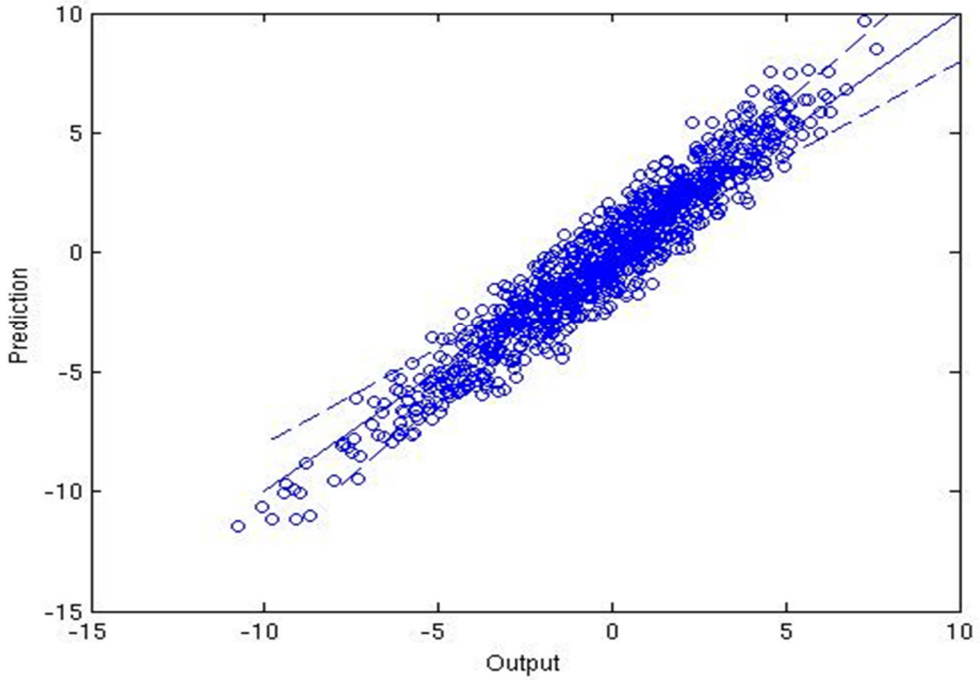

Figure 3.15 Data fit for Bottom product composition (with nonlinear static element))

CHAPTER 4

CONCLUSIONS & FUTURE RECOMMENDATION

Conclusions & Future Recommendation

The executed works under the scope of the present dissertation project are as follows,

- Neural network based identification of the first order process dynamics with and without lag.
- Neural network based identification of CSTR process.
- Neural network based identification of continuous Bioreactor Process.
- Design of DINN controllers for CSTR .
- Design of DINN controllers for various bioreactor configurations.
- Design of IMC based neural controllers for various bioreactor configurations.
- PLS based identification of distillation column dynamics.

  The following recommendations can be made for future course of work:

  - PLS based identification of minimum phase of Process dynamics.
  - Design of multivariable neural controllers.
  - Design of multivariable controllers in PLS framework.

References

Agrawal, P. a. (1984). Analysis of various control schemes for continuous bioreactors. *Adv. Biochem. Eng./Biotechnol.,, 30*, 61-90.

Anuradha B.D, R. G. (2009). Direct inverse Neural Network Control of a CSTR. *International MultiConference of Engineers and Computer Scientists*, *II.* Hong Kong.

Bequette, B. (1998). *Process Dynamics Modeling,Analysis and Simulation.* New Jersey: Prentice Hall PTR.

Dimitris, C. P. (1991). Direct and Indirect Model Based Control Using Artificial Neural Networks. *Ind. Eng. Chem. Res., 30*, 2564-2573.

Dirion, J. L. (1995). Design Of A Neural Controller By Inverse Modelling. *Computers Them. Engng Suppl., 19*, S797-S802.

Donat, J. S. (1990). Optimizing Neural Net Based Predictive Control. *Proceedings of the American Control Conference* (pp. 2466-2471). Piscataway, NJ: IEEE Service Center.

Edwards, V. H. (1972). Dynamics and control of continuous microbial propagators to subject substrate inhibition. *Biotechnol. Bioeng., 14*, 939-974.

Geladi, P. B. (1986). Partial least squares regression: a tutorial. *Analytica Chimica Acta, 187*, 1-17.

Haykin, S. (1999). *Neural Networks a Comprehensive Foundation.* New Jearsy: Prentice-Hall (Pearson Education).

Hernandez, E. a. (1990). Neural Network Modeling and an Extended DMC Algorithm to Control Nonlinear Systems. *American Control Conference* (pp. 2454-2459). San Diego, CA, USA: IEEE Xplore.

Hugo, P., Steinbach, J., A comparison of the limits of safe operation of a SBR and a CTSR. *Chem. Eng. Sci., 41*, 1081-87.

Hussain, M. A. (2001). Implementation of Neural-Network-Based Inverse-Model Control Strategies on an Exothermic Reactor. *ScienceAsia, 27*, 41-50.

Jain, A. K. (1996). Artificial Neural Networks: A Tutorial. *IEEE Computer*, 31-44.

Kaspar, H. M. (1993). Dynamic PLS Modelling for Process Control. *Chemical Engg. Science, 48*(20), 3447-3461.

Kresta, J. (1992). Applications of Partial Least square Regression. *Phd Thesis*. Mc Master University.

Ku, W., Storer, R. H and Geogakis C (1995). Disturbance Detection and Isolation by Dynamic Principal Component Analysis. *Chemo. Intell. Lab Sys., 30*, 179-196.

Lakshminarayanan, S. S. (1997). Modeling and control of multivariable process: Dynamic PLS approach. *American Institute of Chemical Engineers Journal, 43*(9), 2307-22.

Lawrence, J. (1994). *Introduction to Neural Networks: Design, Theory and Applications.* Seattle: California Scientific Software Press.

McAvoy, T. J. (1992). A Comparison of Neural Networks and Partial Least Squares For Deconvoluting Fluorescence Spectra. *Bioengineering and Biotechnology, 40*, 53-62.

McCulloch, W. S. , Pitts. (1943). A logical calculus of ideas immanent in nervous activity. M*athematical Biophysics*(5), 115-133.

Menawat, A. S. (1991). Alternate control structures for chemostat. *AIChE J., 37*, 302-306.

Qin. (1993). Partial Least Squares Regression for Recursive system identification. *32nd Conference on Decision and Control.*

Qin, J. a. (1992). Nonlinear PLS Modeling Using Neural Networks. *Computers and Chemical Engineering, 16*, 379-391.

Roffel, B., & Betlam, B. (2006). *Process Dynamics and Control: Modeling for Control & Prediction.* Chichester, West Sussex: John Wiley & Sons.

Varshney, T. V. (2009). ANN Based IMC Scheme for CSTR. *International Conference on Advances in Computing, Communication and Control*, 543-546. Mumbai.

William, L. N. (1996). *Process Modelling Simulation and Control for Chemical Engineers.* Singapore: McGraw-Hill Chemical Engineering Series .

Wise, B. M. (1991). adapting multivariate analysis for monitoring and modeling dynamic systems. *Phd Thesis*. University of Washington.

Wood, R. a. (1973). Terminal Composition control of a Binary Distillation Column. *Chemical Engg Science, 29*, 1808.

Ydstie, B. E. (1990). Forecasting and Control Using Adaptive Connectionist Networks. *Comput.*

Chem. Eng., 14, 583-599.

Zhao, Y. a. (1997). Comparison of Various Control Configurations for Continuous Bioreactors. *Eng. Chem. Res., 36*, 697-705.