Study on Neural Network Predictive Controller: Impact of Network's Architecture and Plant-Model Mismatch

by

Khadijah bt Ismail

8678

Dissertation submitted in fulfillment of
the requirements for the
Bachelor of Engineering (Hons)
(Chemical Engineering)

JANUARY 2009

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

STUDY ON NEURAL NETWORK PREDICTIVE CONTROLLER: IMPACT OF NETWORK'S ARCHITECTURE AND PLANT-MODEL MISMATCH

By

Khadijah bt Ismail

A project dissertation submitted to the
Chemical Engineering Programme
Universiti Teknologi PETRONAS
In partial fulfillment of the requirement for the
Bachelor of Engineering (Hons)
(Chemical Engineering)

Approved by:

....
Haslinda bt Zabiri
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
January 2009

CERTIFICATION OF ORIGINALITY

This is certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgments, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

....

KHADIJAH BT ISMAIL

ABSTRACT

This final report explains about the extended research done on basic concept of the selected topic, which is Study on Neural Network Predictive Controller: Impact of Network's Architecture and Plant-model mismatch. The objective of the project is to study the effect of different network's architecture by manipulating the transfer function, number of neurons, weight and biases on NN-based Predictive Controller's performance. And to study the impact of parameters used in CSTR on the performance of the based model in Plant-Model Mismatch. In literature review section, NN-based Predictive Controller will be further discussed. The successful outcomes of this project can be applied in the industries in order to help reducing the uncertainty or inaccuracy in process control.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 BACKGROUND OF STUDY

Model Predictive Controller, MPC technologies have been widely used in the industries. MPC can be used in either linear or nonlinear system based on the limitation of the controller. NMPC is normally used in highly and moderately nonlinear process. It is also used when the process requires higher product quality specification, increasing productivity demands and the process need to operate system closer to the boundary of operating region. Thus, MPC with NN based is developed, which is NN Predictive Controller and this model are used as a based model in this study. The performance of this model will be monitored by manipulating the transfer function, number of neurons, weight and biases. And the impact of parameters used in CSTR will be studied so that it can be controlled and optimized.

1.2 PROBLEM STATEMENT

Strong nonlinear dynamic behavior is becoming one of the major problems in polymer industry and chemical process. The usage of Nonlinear Model Predictive Controller (LMPC) is not widely used to enhance the performance.

1.3 SIGNIFICANT OF THE PROJECT

Upon completion of the research and simulation, the final simulation will determine the best architecture for NN Predictive Controller. The implementation of best architecture for NN Predictive Controller will optimize the model and reduce the non-linearity of the process and optimize the product produced.

1.4 OBJECTIVES AND SCOPE OF STUDY

The objective of the project is to study the effect of different architecture on NN Predictive Controller's performance. And to perform robustness analysis on developed NN Predictive Controller for Plant-Model mismatch. The scope of study is to applied different transfer functions and number of neurons on NN Predictive Controller.

CHAPTER 2

LITERATURE REVIEW

2.1 MODEL PREDICTIVE CONTROLLER, MPC

In general, the purpose of MPC is to overcome the uncertainty/inaccuracy and unmeasured disturbance that can cause the plant output to behave differently. **Figure 1** shows the structure of typical MPC system.

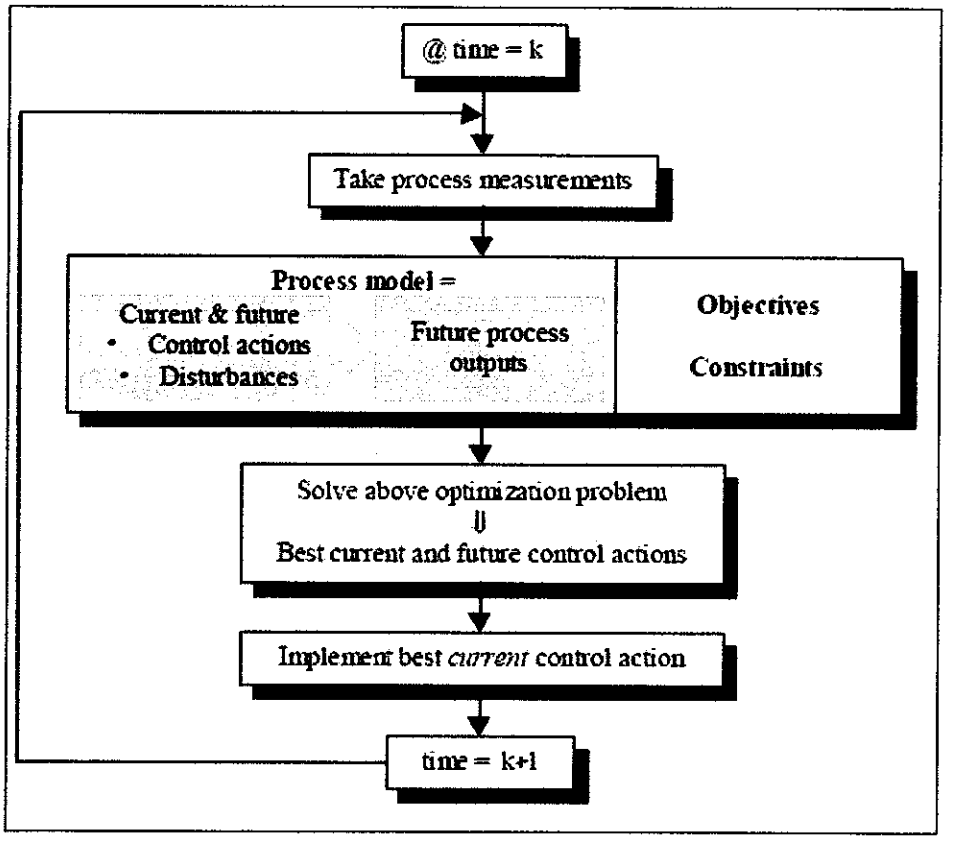


Figure 1: The MPC scheme [1]

MPC is based on iterative, finite horizon optimization of plant model. The current measurement is taken as initial state for each sampling time, k and explicit model will predict the future behavior of the process (see **Figure 2**). At each control intervals, ($u(k)=(k+j|_k)$) the MPC algorithm determined the solution of optimization problem by computing the sequence of optimal future manipulated variable adjustment per a fixed number of control horizon, M. Beyond the control horizon ($M+k-1$), no action will be taken since the manipulated variables is assumed to be constant. Even a lot control move is optimally calculated, only the first input in the optimal sequence will be implemented. At the next sampling, the entire sequence is repeated again and the optimization problem will be reformulated and solved using new measurement. And prediction horizon and control horizon will move or recede ahead by one step as time moves ahead by one step. [2]

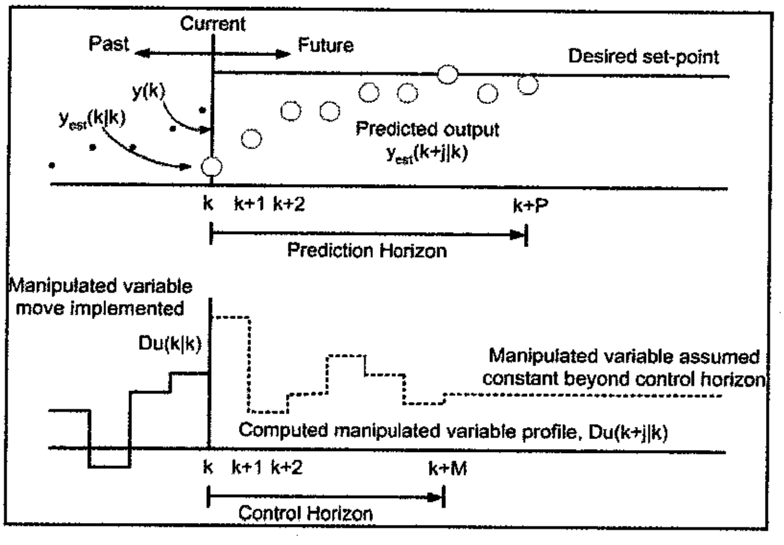


Figure 2: Principle of MPC.

2.2 NONLINEAR MODEL PREDICTIVE CONTROLLER, NMPC

The first steps taken for NMPC to estimate the system states from the output measurement is by obtaining measurements of the system state. The measurement is then compute and optimal signals by minimizing a given cost function over a certain prediction horizon in the future using a model of the system. The first part of the optimal input signal is implemented until new measurement is available.

There are three types of model approaches for NMPC which are:

- Fundamental Model
- Black Box Model
- Discrete Time Model

The first approach is fundamental model which is important when it operates in wide range but it is difficult and time consuming process. It has the advantages over the black box model on extrapolation ability and probability to multiple facilities. The black box model can be described as relatively easy and economically attractive alternative in many situations. It can be developed directly from perturbed plant data. The most appropriate model is discrete time model because plant data is available at discrete time instant.

There is various type of controller model used in industry for nonlinear system. The examples of model of controller used are *Gaussian Process Model* [3], *Wiener-Laguerre Model* [4], *Hammerstein and Wiener Model* [5], *Nonlinear State Space Model* [6], *Partial Least Squares (PLS) models* [7], *Neural Networks Model* [8], *Stochastic Closed Loop Model* [9], *Neuro Fuzzy Hammerstein* [10] and *Radial Basis Function Network (RBFN)* [11]. And this report is focused on Neural Network based model.

2.3 LINEAR MODEL PREDICTIVE CONTROLLER, NMPC

LMPC approaches are used in the majority of application with feedback mechanism of the MPC compensating for prediction errors due to structural mismatch between the model and the plant. The successful application of NMPC can be seen in petroleum refinery, petrochemical, chemical sectors, power plants, pulp and paper and food processing industries and also automobile and aerospace areas.

The *Finite Impulse Response (FIR)* [12], *Multiplex MPC (MMPC)* [13] and *Dynamic Matrix Control (DMC)* [14] are type of models used for linear system in the industry.

2.4 NEURAL NETWORK (NN)

NN are massively processor controllers which consist of nonlinear computer algorithm that learn feedback and have the ability to learn patterns through training experiences. Because of this feature, it is often well suited for modeling complex and non-linear processes such as CSTR process [15]. NN works by training, validation and testing the data to predict the output of the process. The training of a NN involves estimating the unknown parameters; this procedure generally utilized normal operating data which is often to be large data set, taken in the operating region where the model is intended to be used [16]. To train a network, an input vector is applied to the network and the output of the network is calculated and compared to the corresponding target vector with the difference (error) being fed back through the network to change the weights so that the error is minimized. After the parameters are trained, another large set of data can be used to validate that the model is adequate.

The general arrangement of NN is in the form of layer. It is normally consist of input layer, hidden layer and output layer. The number of hidden layer is optimized to get accurate prediction from NN. For this case study, the numbers of hidden layers are fixed at two layers to reduce complication.

2.4.1 NEURONS IN NEURAL NETWORKS

Neural networks are made up of many artificial neurons. The neuron continuously receives signals from these inputs and then sum up the inputs at each layer. Neurons are sometimes referred as nodes. A network can have several layers. Each layer has a weight matrix W, a bias vector b, and an output vector a [17]. And each layer has its own number of neurons. It is common for different layers to have different numbers of neurons. This is one of the most critical parameter affecting the accuracy of prediction in neural network. One can keep on reducing the training error by increasing number of neurons.

In this project, two layers of network are used and the number of neurons at the first layer is varies from 2 to 10 neurons. The performance of each number of neurons is recorded. It is to be noted that often the increase and decrease in performance may not be monotonic with the number of hidden neurons [18].

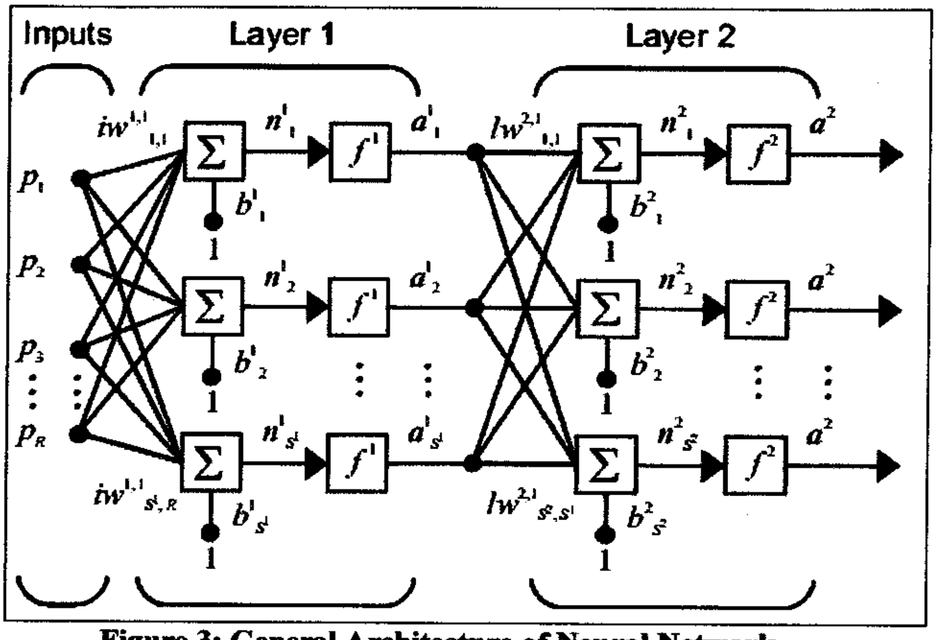


Figure 3: General Architecture of Neural Network

As illustrated in **Figure 3**, the outputs of each layer are the inputs of the following layer. The neuron has a bias *b*, which is summed with the weighted inputs to form the net input *n*. This sum, *n*, is the argument of the transfer function, *f* [17]. In order to improve the prediction, weight will be adjusted accordingly. Therefore, weight can be positive or negative.

2.4.2 ACTIVATION FUNCTION IN NEURAL NETWORKS

Activation function is also known as transfer function. It is one of the criterions to characterize the neural network system. Transfer function is an algebraic expression for dynamic relation between the selected input and output of the process model. It is defined so as to be independent of the initial conditions and of the particular choice of forcing functions [16]. The equation of the activation function can be described as in equation (1) [20].

$$a = f(wp + b)(3) \qquad \ldots(1)$$

The output of a neuron, a will depends on the product of weight, w and its input value, p. The product of this two will be added with the bias, b in order to obtain the overall input for the neurons. The overall input will be multiplied with the transfer function, f which will determine the output of a neuron.

Among the typical activation function used are:

i. Hyperbolic tangent sigmoid

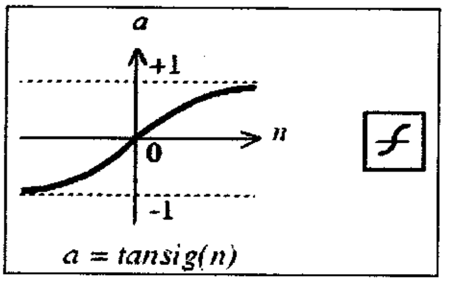


Figure 4: Graph of tansig transfer function [17].

ii. Log-sigmoid

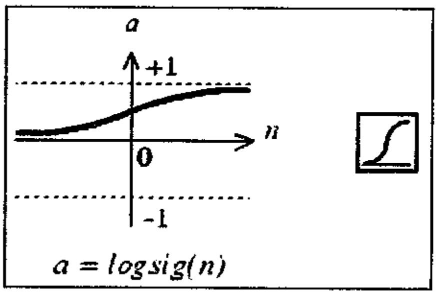


Figure 5: Graph of logsig transfer function [17].

iii. Linear

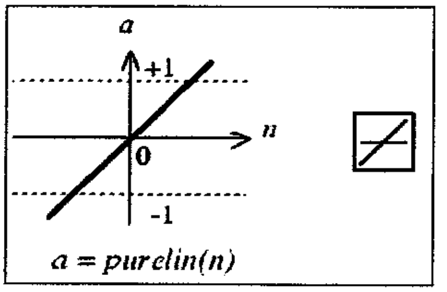


Figure 6: Graph of purelin transfer function [17].

2.4.3 NEURAL NETWORK PREDICTIVE CONTROLLER

NN predictive controller uses NN model of nonlinear plant (CSTR) to predict future plant performance. The controller then calculates the control input that will optimize plant performance over a specified future time horizon. The first step in model predictive control is to determine the neural network plant model (system identification). Next, the plant model is used by the controller to predict future performance. This application is used in Simulink. The performance of plant is optimized using controller by calculating the control input. This controller objective is to control the concentration of the CSTR according to it set points.

The first stage of model predictive control is to train a neural network to represent the future of the plant. The prediction error between the plant output and the neural network output is used as the neural network training signal. The process is represented by the following figure:

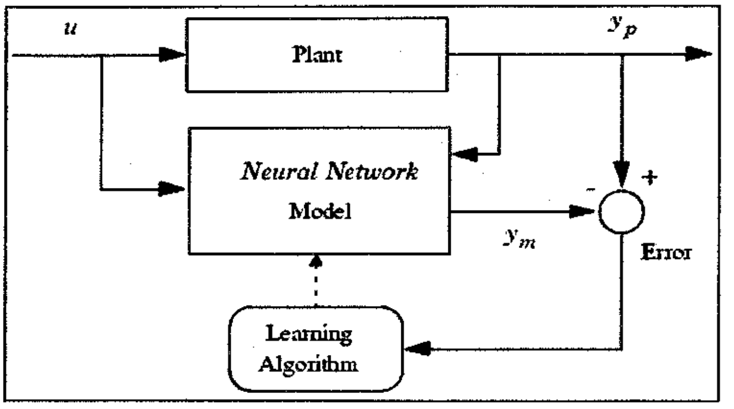


Figure 7: System identification

The neural network plant model uses previous inputs and previous plant outputs to predict future values of the plant output.

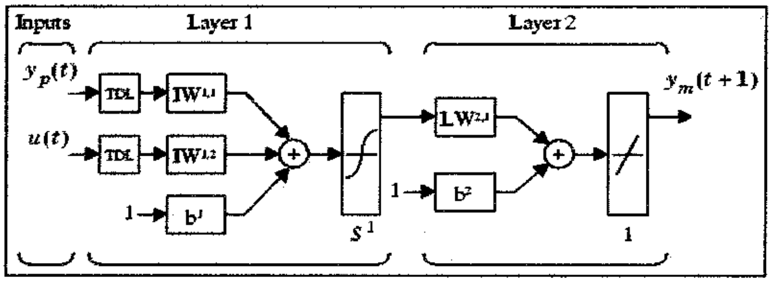


Figure 8: General structure of NN Back-Propagation.

Basically, NN predictive controller is added into CSTR model developed in Simulink. The model used in NN model is state-space model where it simplifies the complicated mathematical expression used in the process. The following block diagram illustrates the model predictive control process. The controller consists of the neural network plant model and the optimization block.

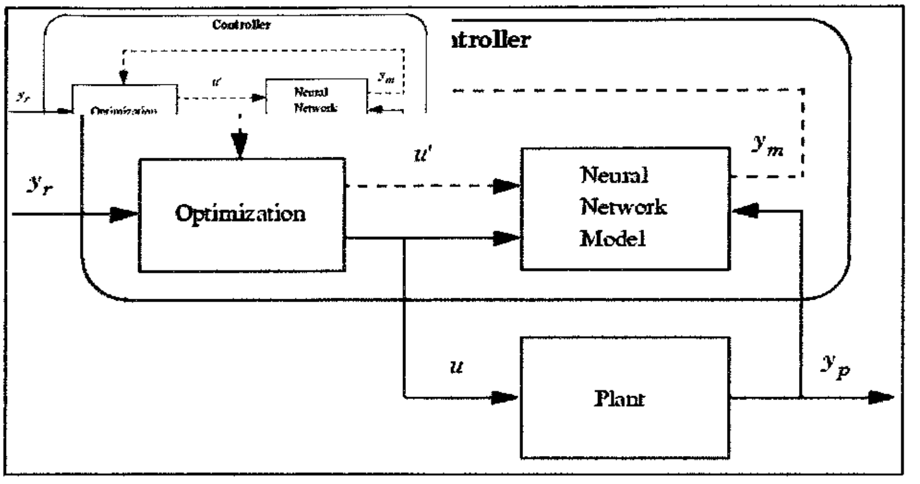
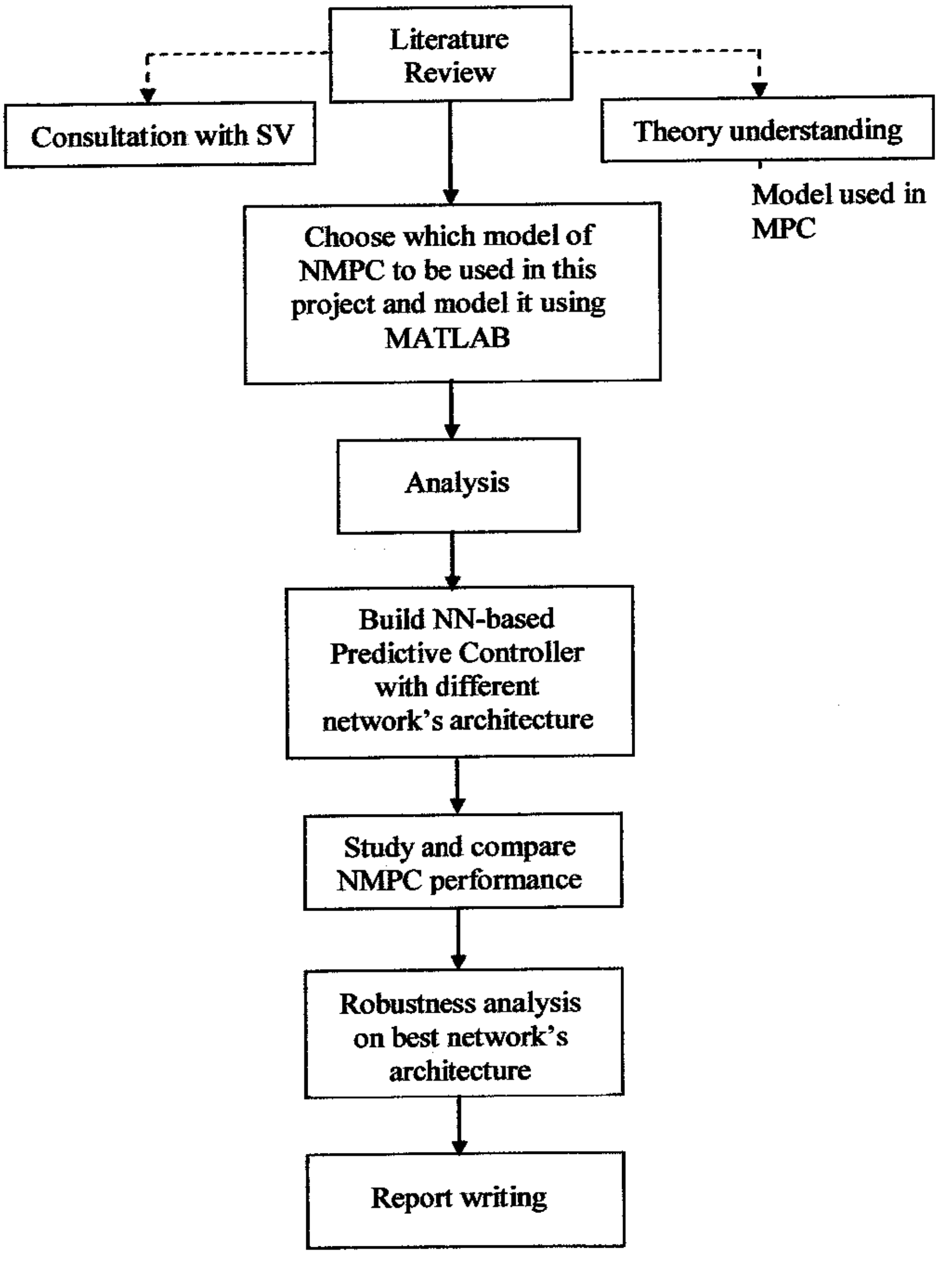


Figure 9: The controller block implemented in Simulink.

CHAPTER 3

METHODOLOGY



CHAPTER 4

RESULT AND DISCUSSION

4.1 CSTR GENERAL MODEL

The CSTR model is developed to control the concentrations within the CSTR at the desired level. CSTR process consists of a constant volume reactor. The diagram of this process is shown in **Figure 12.**

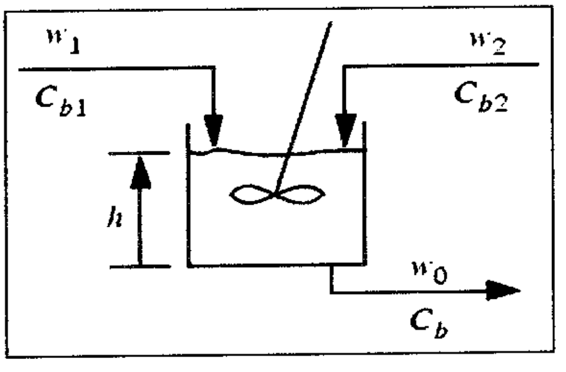


Figure 10: Schematic diagram of a CSTR [17].

The objective of the CSTR is to control the product concentration by controlling the flow of w_1. To simplify the process, the flow rate w_2 is kept constant at $0.1 m^3/s$. In order to model the CSTR, it required a dynamic model. This dynamic model is:

$$\frac{dh(t)}{dt} = w_1(t) + w_2(t) - 0.2\sqrt{h(t)}$$

$$\frac{dC_b(t)}{dt} = \left(C_{b1} - C_b(t)\right)\frac{w_1(t)}{h(t)} + \left(C_{b2} - C_b(t)\right)\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{\left(1 + k_2 C_b(t)\right)^2} \quad \ldots(2)$$

Where $h(t)$ is the liquid level, $C_b(t)$ is the product concentration at the output of the process, $w_1(t)$ is the flow rate of the diluted feed C_{b2} and $k1$ and $k2$ are the constant associated with rate of consumption. The level of the tank *h(t)* is not controlled for this experiment [19]. The values used for this experiment are set as follow:

Table 1: Values used for CSTR modeling

Variable	Value
C_{b1}	24.9mol/L
C_{b2}	0.1mol/L
$k1$	1
$k2$	1
$W_2(t)$	0.1L/min

4.2 BASED MODEL

Based model is the model of CSTR with NN MPC which has been modified in order to follow the project requirements.

The CSTR model simulation objective is to control the measured process concentration by manipulating the flow rate of concentrated feed, C_{b1}. This CSTR model is developed using block diagram in MATLAB Simulink based on the CSTR dynamic model equation. The CSTR model is the typical model used for simulation.

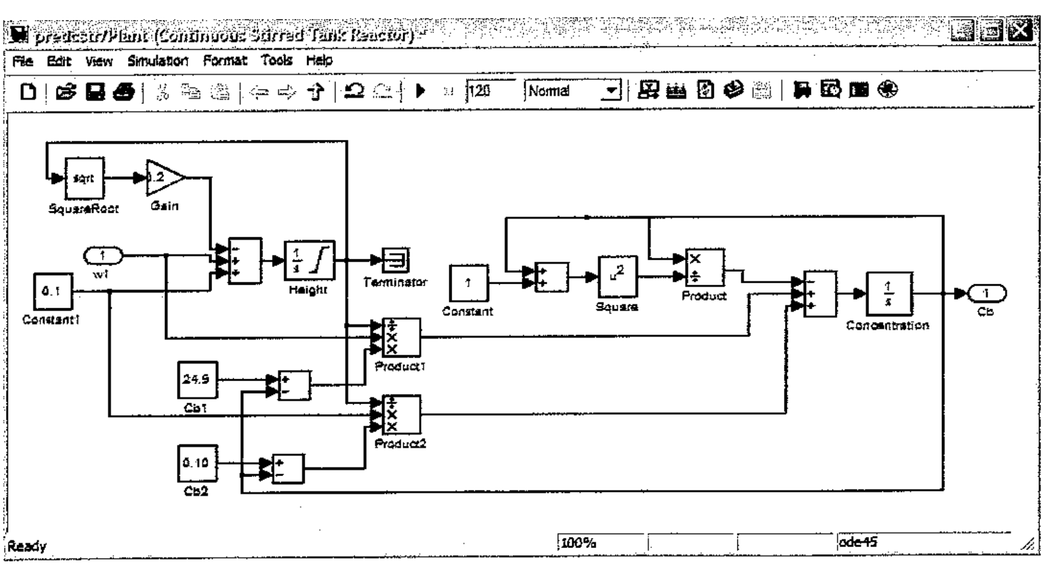


Figure 11: CSTR model in MATLAB Simulink [19].

Below is the figure of based model used in this project which consists of CSTR model and NN MPC. The project objective is to optimize the controller in order to obtain the best performance that can control the nonlinearity of CSTR. The based model structure is shown in **Figure 14.**

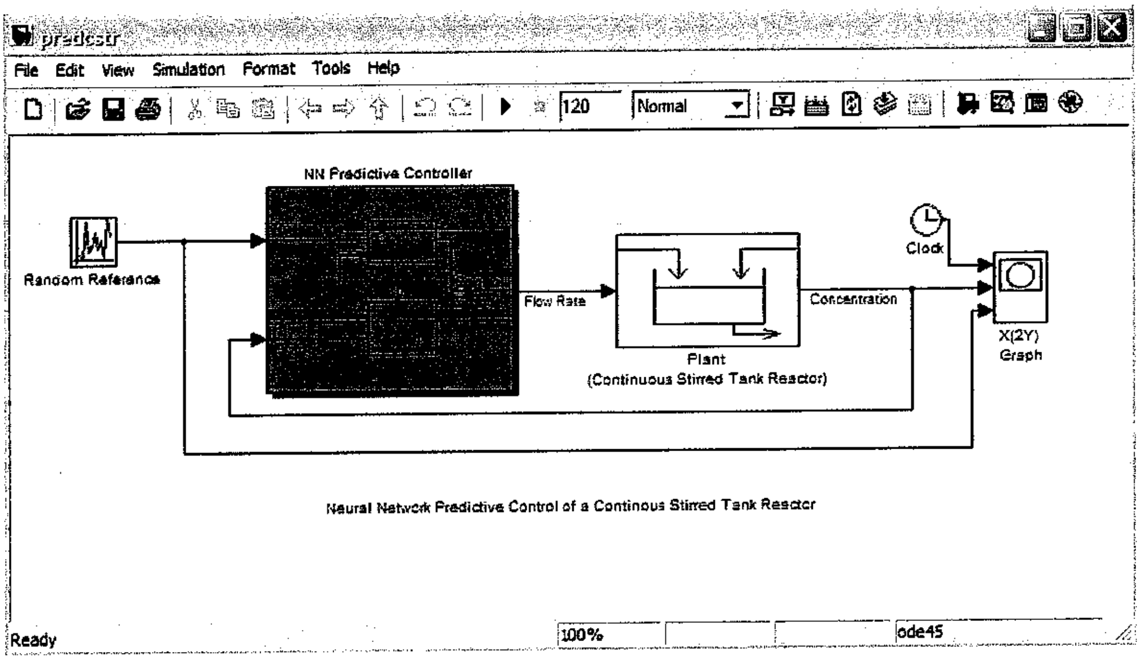


Figure 12: CSTR with NN Predictive Controller in Simulink [19].

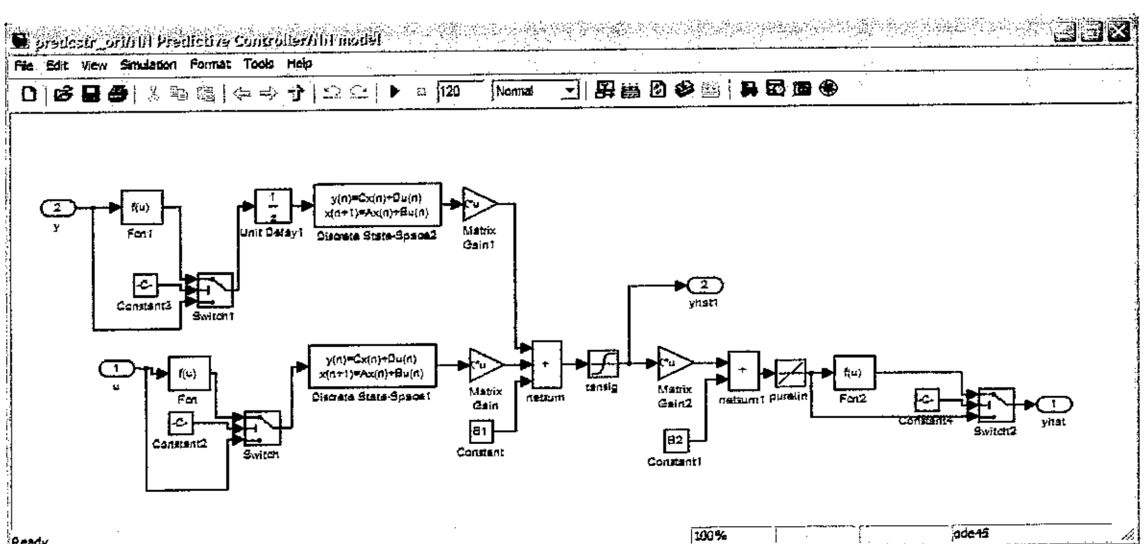


Figure 13: Based model structure [19].

4.3 DIFFERENT NETWORK'S ARCHITECTURE

In order to achieve different architecture, the transfer functions and number of neurons is manipulated. At first, the transfer function is changed to obtain new weight and biases before it is applied on NN-based Predictive Controller, and new data is generated and trained. The performance of each transfer function is recorded. All the results obtained are discussed below:

4.3.1 RESULT FOR DIFFERENT ACTIVATION FUNCTIONS

The activation functions used in this project are log-sigmoid (L), tangent-sigmoid (T) and purelin (P). Since the networks consist of 2 layers, the activation function must be arranged in pair. So, there are nine combination of activation function used in this project. All performance of the nine combinations will be discussed below. On each simulation, new data set will be generated and trained before it is applied into the controller. From the result obtained, it can be concluded that, the NN-based Predictive controller can only perform using combination of activation function of T and P (see **Figure 18**).

i. Log-sigmoid and Log-sigmoid

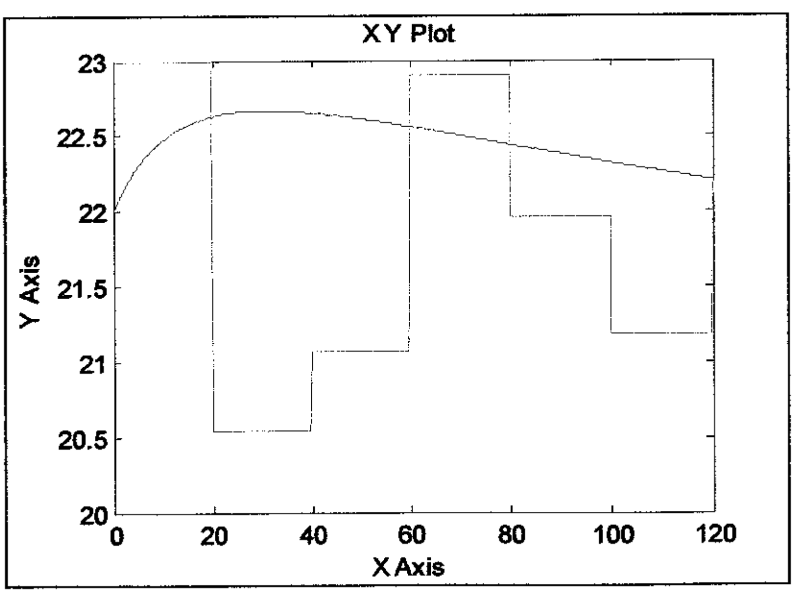


Figure 14: Performance of LL activation function.

ii. Log-sigmoid and Purelin

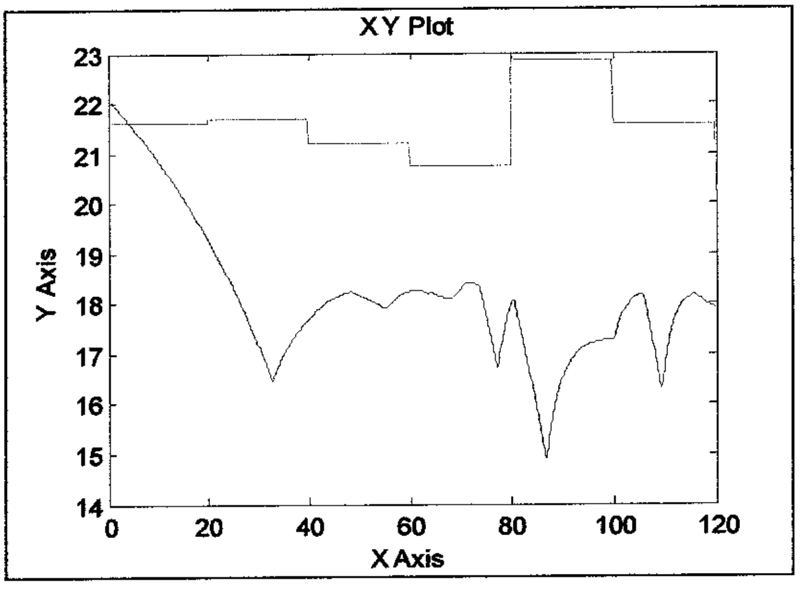


Figure 15: Performance of LP activation function.

iii. Log-sigmoid and Tan-sigmoid

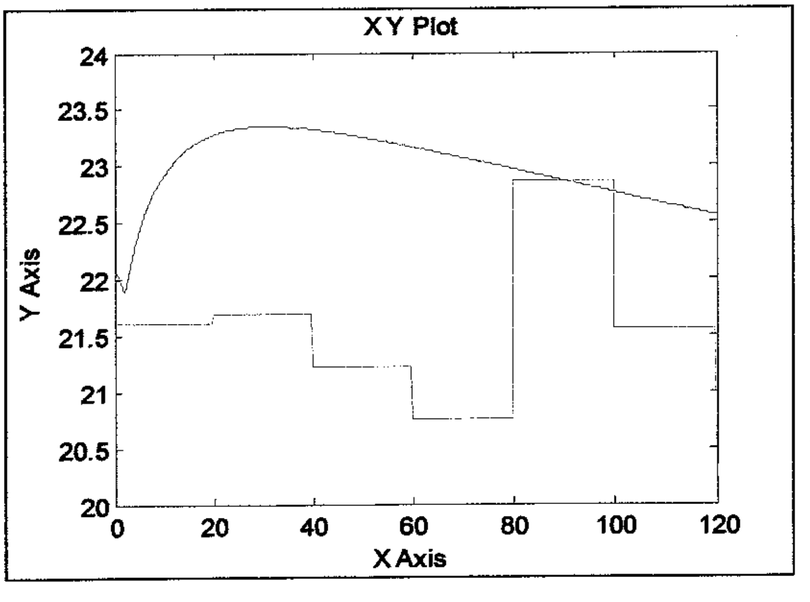


Figure 16: Performance of LT activation function.

iv. Tan-sigmoid and Tan-sigmoid

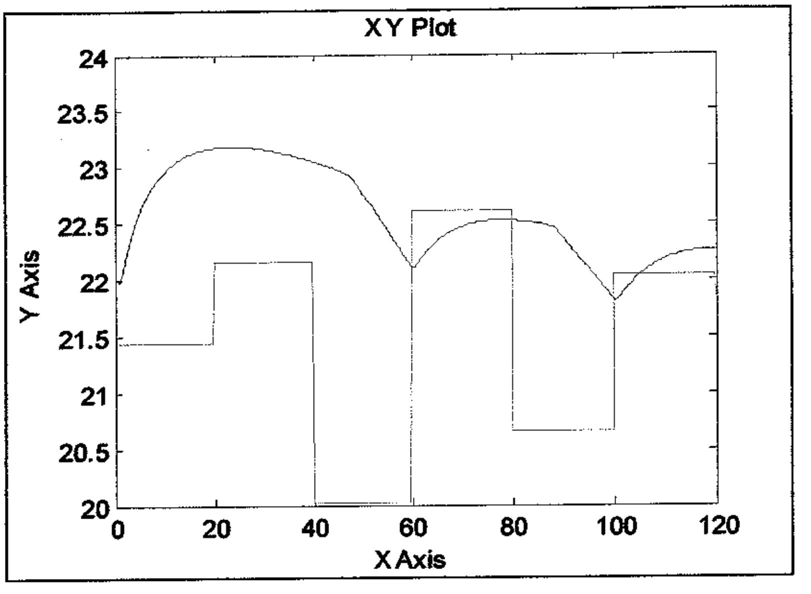


Figure 17: Performance of TT activation function.

v. Tan-sigmoid and Purelin

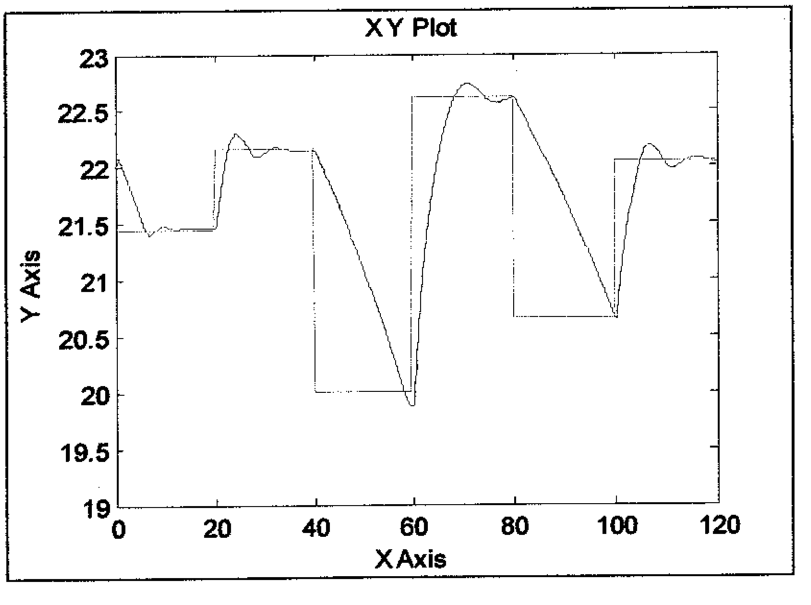


Figure 18: Performance of TP activation function.

vi. Tan-sigmoid and Log-sigmoid

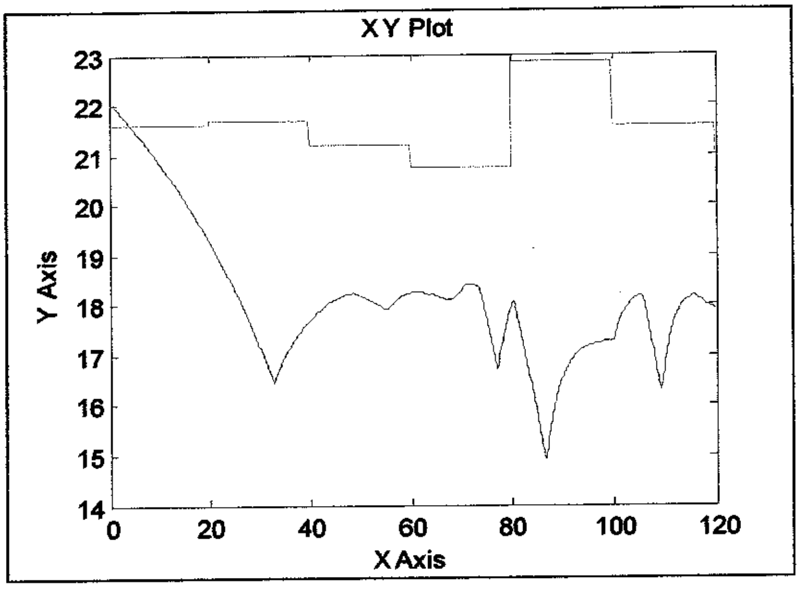


Figure 19: Performance of TL activation function.

vii. Purelin and Purelin

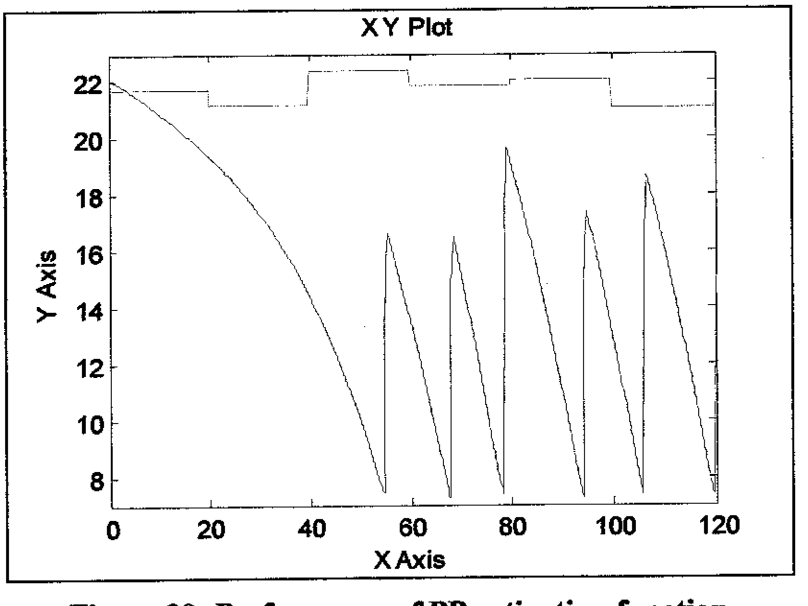


Figure 20: Performance of PP activation function.

viii. Purelin and Tan-sigmoid

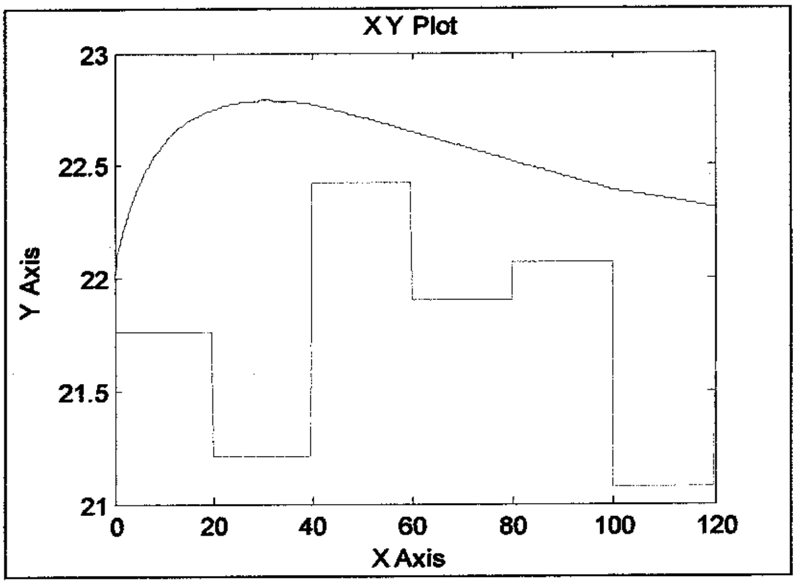


Figure 21: Performance of PT activation function.

ix. Purelin and Log-sigmoid

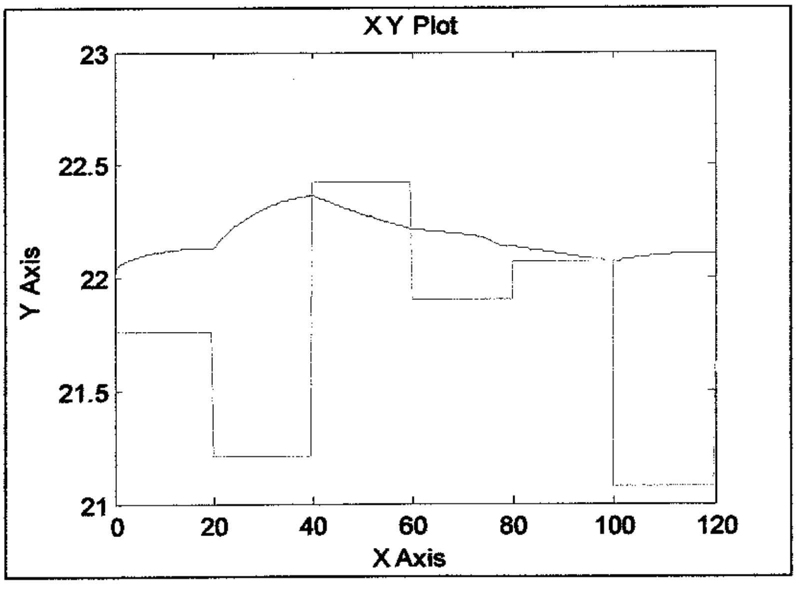


Figure 22: Performance of PL activation function.

4.3.2 RESULT FOR DIFFERENT NUMBER OF NEURONS

For different architecture in number of neurons, each activation function is tested using 2, 4, 6, 8 and 10 neurons with fixed layer of 2. The performance of this different architecture did not show well due to some problem occurred within the based model itself.

Below shows the best performance of best combination of activation function which is tan-sigmoid and purelin for 2, 4, 6, 8 and 10 neurons. Other results are shown in Appendix 2. From the simulation, for combination of activation function Tan-sigmoid and Purelin, it performs really well with 6 numbers of neurons. Result is shown in **Figure 25.**

i. 2 neurons

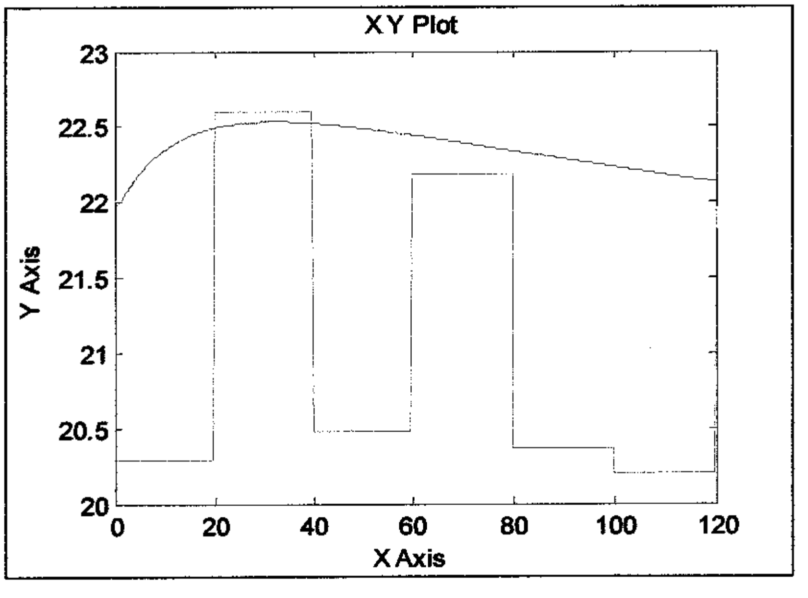


Figure 23: Performance of TP for 2 neurons.

ii. 4 neurons

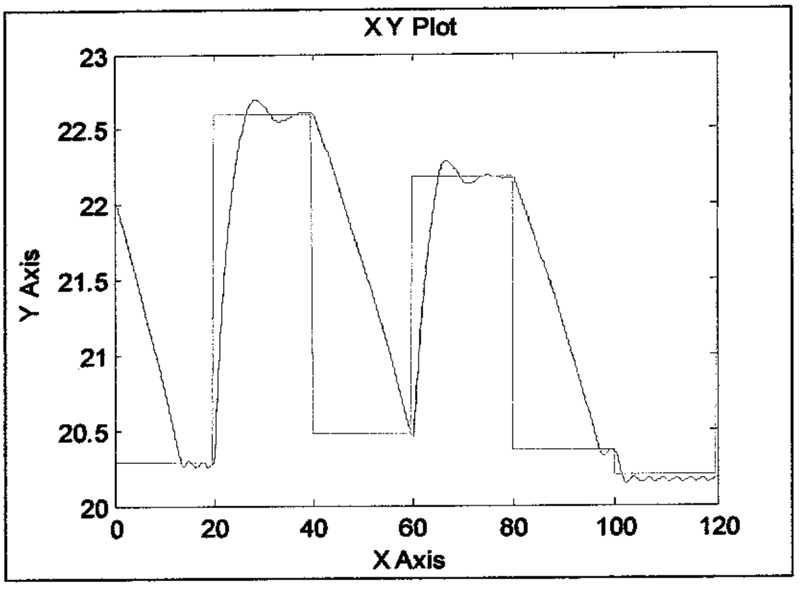


Figure 24: Performance of TP for 4neurons.

iii. 6 neurons

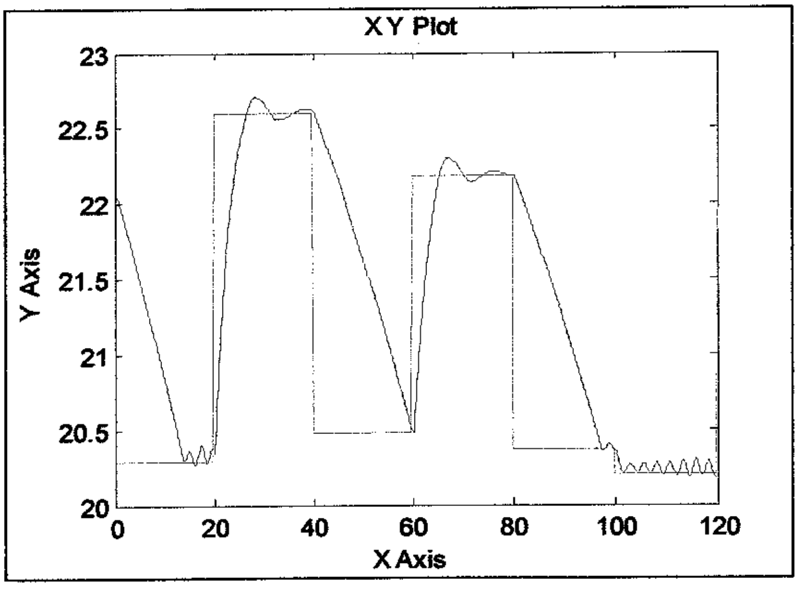


Figure 25: Performance of TP for 6neurons.

iv. 8 neurons

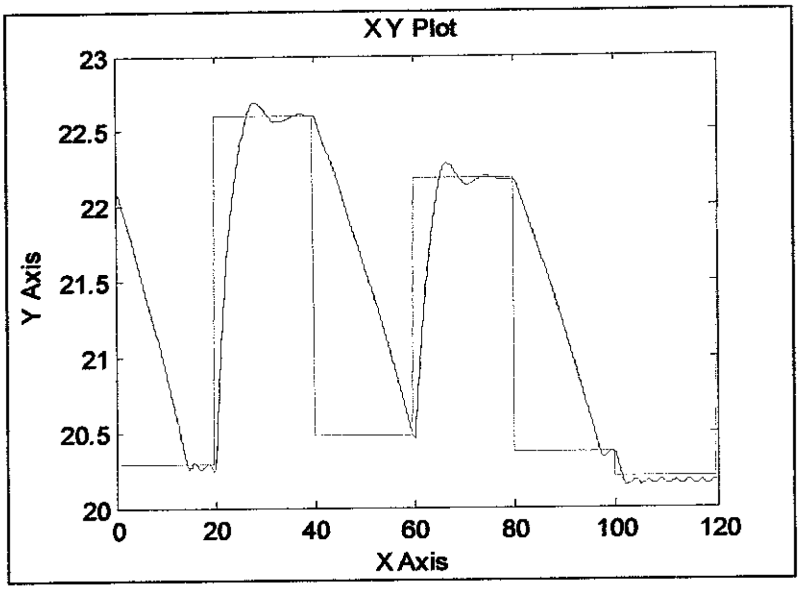


Figure 26: Performance of TP for 8neurons.

v. 10 neurons

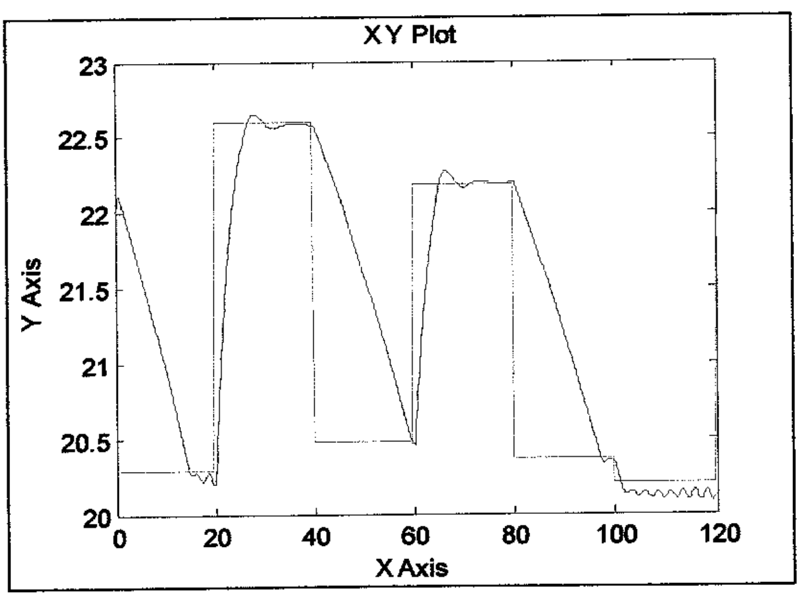


Figure 27: Performance of TP for 10neurons.

4.4 ROBUSTNESS ANALYSIS

Robustness analysis is done to determine the range of parameters used in CSTR at which the NN-based Predictive Controller is capable in maintaining the performance. This analysis is done by manipulating the concentration of both inlets, C_{b1} and C_{b2} respectively.

At first, the concentration of C_{b1} is varies by increasing and decreasing the value of C_{b1} until the performance of the model shows inaccuracy. After running the simulation, it is observed that Cb1 can vary from 24.4mol/L up to 25.9mol/L in order to maintain the performance of the model. At concentration of 25.9mol/L, it shows the best performance.

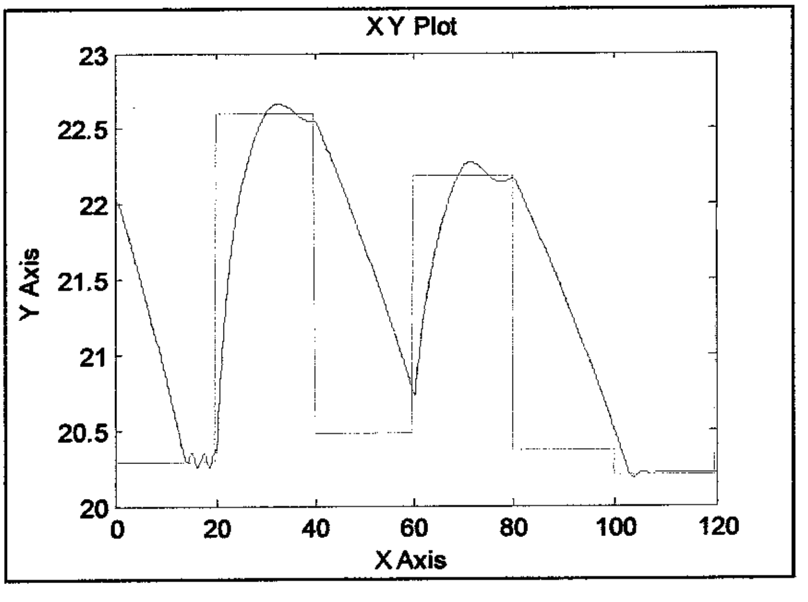


Figure 28: Performance at C_{b1} is 24.4mol/L

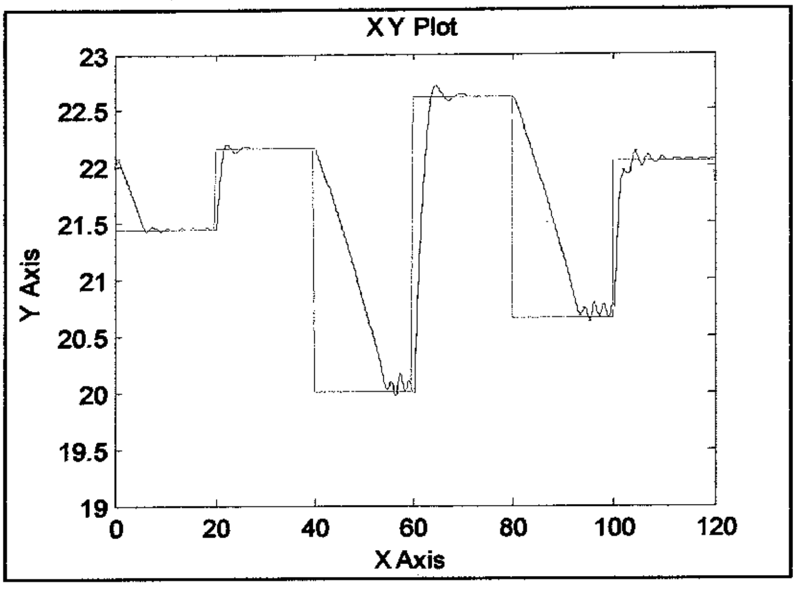


Figure 29: Performance at C_{b1} of 25.9mol/L

When the value of C_{b1} is less than 24.4mol/L or greater that 25.9mol/L, the graph shows inaccuracy in its performance. As shown in **Figure 30,** the prediction which is indicated by the blue line is way below the actual value. Thus at this point, the model is not capable in maintain the model performance.

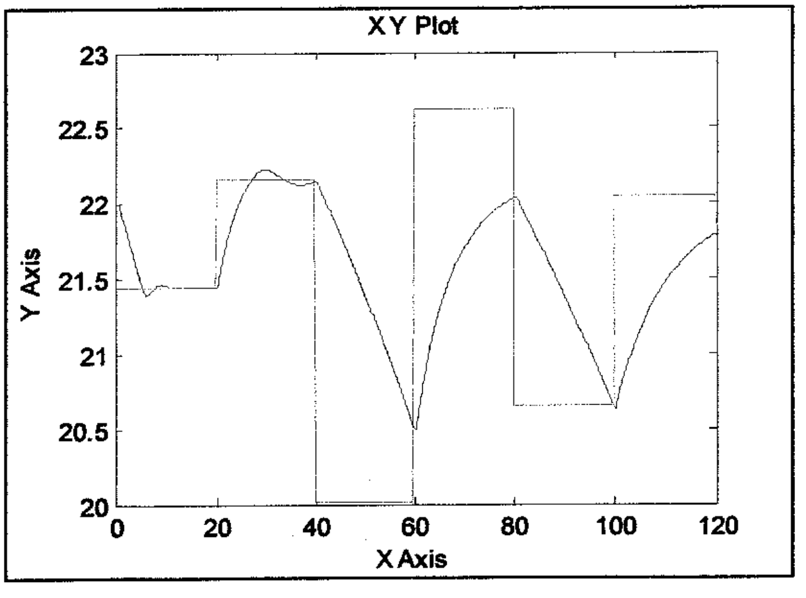


Figure 30: Performance at C_{b1} less than 24.4mol/L

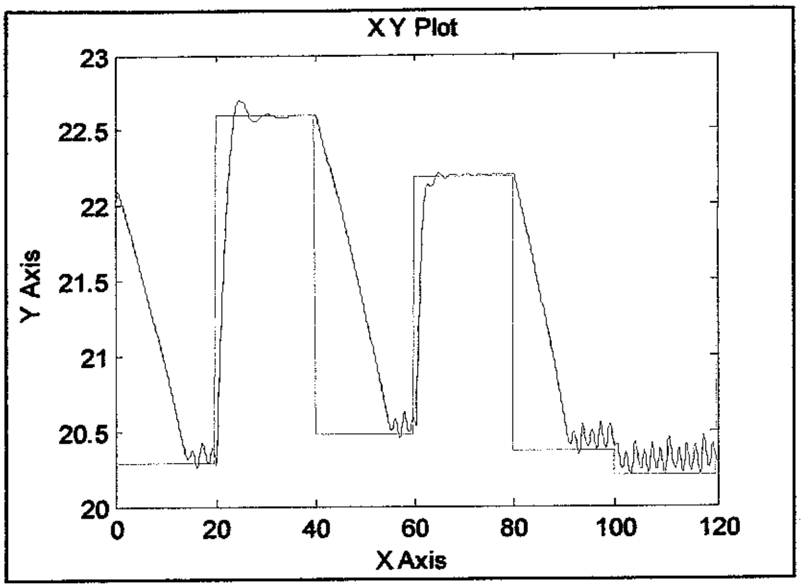


Figure 31: Performance at C_{b1} greater than 25.9mol/L

For the next robustness analysis, the concentration of C_{b2} is varies increasingly and decreasingly in order to obtain the range at which the model can accurately predict and maintain its performance. From the simulation, it is observed that the best performances of this model are between 0mol/L to 10mol/L. But the optimum point of for the system to operates is when C_{b2} at 0.1 and 10mol/L.

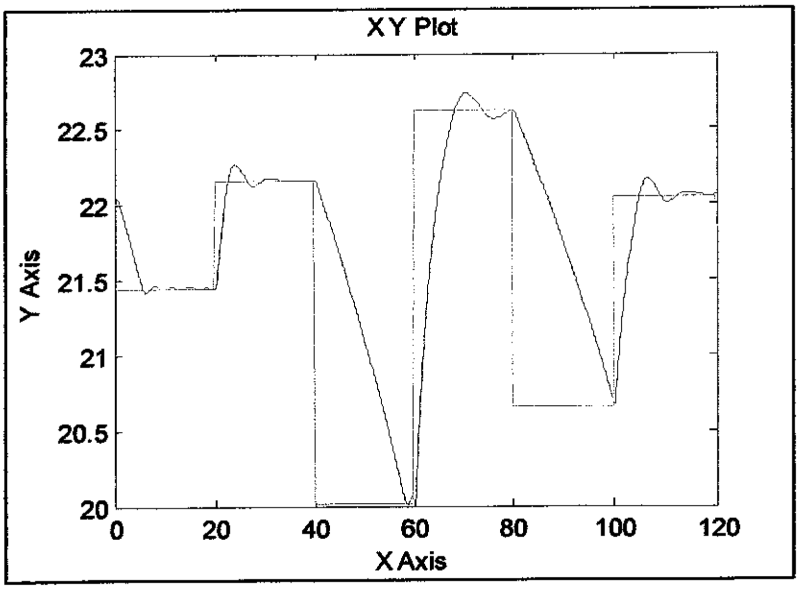


Figure 32: Performance at C_{b2} of 0.1mol/L

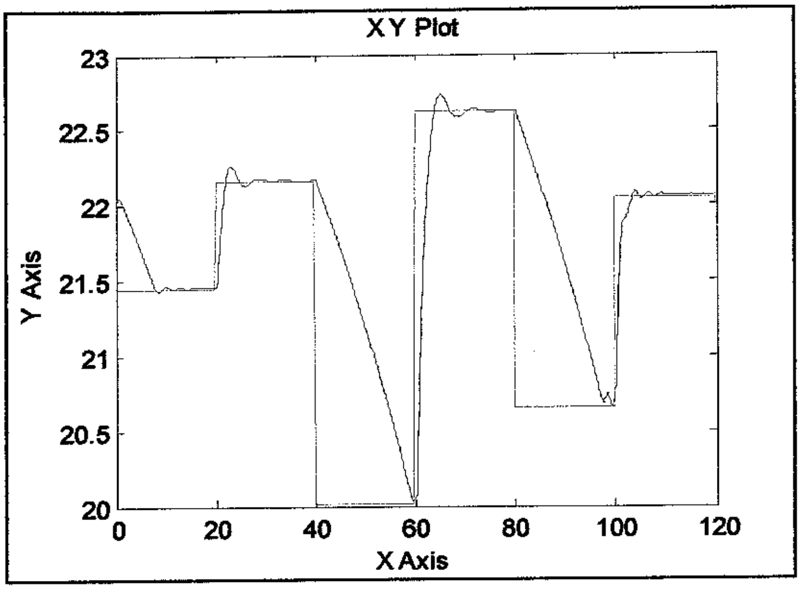


Figure 33: Performance at C_{b2} of 10mol/L

When the value of C_{b2} is less than 0mol/L or greater than 10mol/L, the actual value cannot be predicted properly. The graph will deteriorate from the actual value.

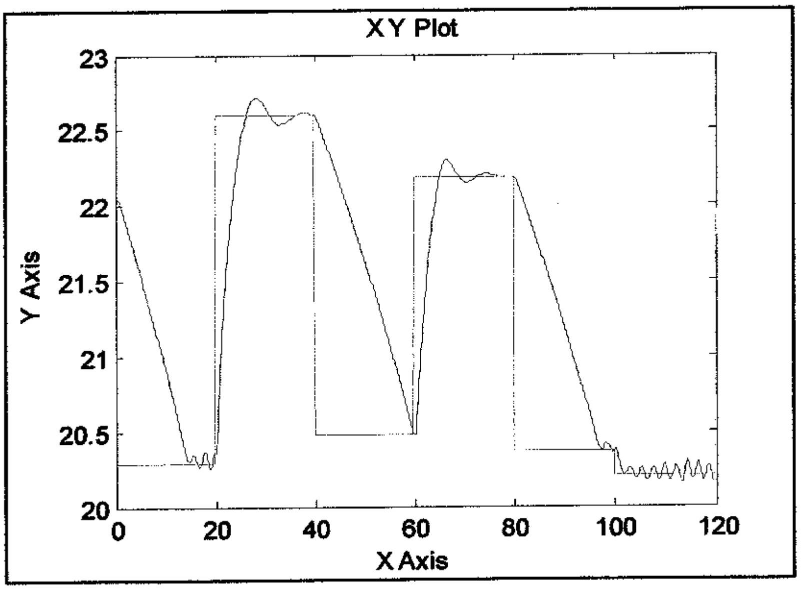


Figure 34: Performance at C_{b2} less than 0mol/L

CHAPTER 5

CONCLUSION

NMPC is an optimal control based method which is one of the techniques that can be used to stabilize processes in the presence of nonlinearity and uncertainty. The performance study for the impact of different architecture on NN-based Predictive Controller has shown that the model gives it best performance for 2 layers with the combination of activation function of Tan-sigmoid and Purelin. And this model with these activation functions performs very well with 6 numbers of neurons. In robustness analysis, the NN-based Predictive Controller is able to cater the changes in C_{b1} from 24.4mol/L up to 25.9mol/L in order to maintain the performance of the model. While for C_{b2}, the concentration must be in the range between 0mol/L to 10mol/L for the model gives good prediction. If these parameters are exceeding the limits, it will reduce the model performance.

REFERENCES

1. Michael Nikolaou, *Model Predictive Controllers: A Critical Synthesis of Theory and Industrial Needs.*

2. Haslinda Zabiri, *PHD Research Proposal – Development of Nonlinear State-Space Model Predictive Controller* (2008).

3. Juˇs Kocijan, Roderick Murray-Smith, Carl Edward Rasmussen & Agathe Girard, *Gaussian Process Model Based Predictive Control* (2004).

4. Sanaz Mahmodi, Javad Poshtan, Mohammad Reza Jahed-Motlagh & Allahyar Montazeri, *Nonlinear Model Predictive Control of a pH Neutralization Process Based on Wiener–Laguerre Model* (27 June 2008).

5. Hai-Bo Huo, Xin-Jian Zhu,Wan-Qi Hu, Heng-Yong Tu, Jian Li & Jie Yang, *Nonlinear Model Predictive Control of SOFC Based on a Hammerstein Model* (2008).

6. Michael Lundh and Mats Molander, *State-Space Models in Model Predictive Control, 2006.*

7. Hong Zhao, John Guiver, Ramesh Neelakantan & Lorenz. T. Biegler, *A nonlinear Industrial Model Predictive Controller Using Integrated PLS and Neural Net State-Space Model* (Control Engineering Practice 9, 2001).

8. Bernt M. Akesson, Hannu T. Toivonen, *A Neural Network Model Predictive Controller*, 2006.

9. Dennis Van Hessem & Okko Bosgra, *Stochastic Closed-Loop Model Predictive Control of Continuous Nonlinear Chemical Processes* (Journal of Process Control 16, 2006).

10. Li Jia, Min-Sen Chiu & Shuzhi Sam Ge, *A Noniterative Neuro-Fuzzy Based Identification Method for Hammerstein Processes* (2005).

11. Ch. Venkateswarlu & K. Venkat Rao, *Dynamic Recurrent Radial Basis Function Network Model Predictive Control of Unstable Nonlinear Processes* (2005).

12. S. Joe Qin, Thomas A. Badgwell, *A Survey of Industrial Model Predictive Control Technology* (Control Engineering Practice 11, 2003).

13. Keck Voon Ling, Jan Maciejowski & Bing Fang Wu, *Multiplexed Model Predictive Control* (2005).

14. G. O¨ zkan, A. U¨ ser, H. Hapoglu & M. Alpbaz, *Experimental Application of Linear/Nonlinear Dynamic Matrix Control to a Reactor of Limestone Slurry Titrated With Acid Sulphuric* (Chemical Engineering Journal 137, 2008).

15. DU Jingjing, SONG Chunyue & LI Ping, *Modeling and Control of a Continuous Stirred Tank Reactor Based on a Mixed Logical Dynamical Model* (Chin. J. Chem. Eng. 2007).

16. Dale E. Seborg, Thomas F. Edgar, Duncan A. Mellichamp, *Process Dynamics and Control,* 2nd edition, 2004.

17. Neuron Model, *http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/nnet.html?BB=1,* 21 July 2008.

18. Prakash Mahadeo Dixit, Uday S. Dixit, *Modeling of Metal Forming and Machining Process*, 2008.

19. Design and Simulate Neural Network, *http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/index.html?/access*

/helpdesk/help/toolbox/nnet/control3.html&http://www.mathworks.com/cgi-bin/texis/webinator/search/?db=MSS&prox=page&rorder=750&rprox=750&rdfreq=500&rwfreq=500&rlead=250&sufs=0&order=r&is_summary_on=1&ResultCount=10&query=nn+CSTR, 21 May 2008.

20. Irfan B. Abd Rahman, *Face Recognition for MyKad Security*, 2003.

21. John Hedengren, *Continuous Stirred Tank Reactor (CSTR) Model*, *http://www.mathworks.com/matlabcentral/fileexchange* (2007).

22. H. Zabiri*, N. Mazuki, *Optimization of Neural Network Model Structures for Valve Stiction Modeling*, 2001.

23. Darell D. Massie, *Neural Network Fundamentals for Scientists and Engineers*, 2001.

APPENDICES

APPENDIX 1: NARX NN Coding

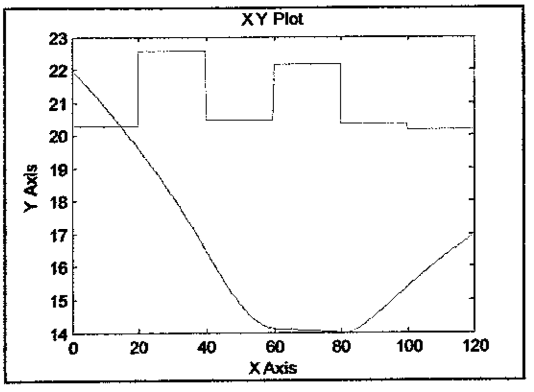
```
clear;clc;close all;
%  Extract data from M-file
A = xlsread('data_file_name');

% Determine size of XY matrix
[row,col] = size(A) ;

%  Allocating input and target columns for Training, Validation and Testing
P_tr = A(1:1500,1)';
T_tr = A(1:1500,2)';
P_v = A(1501:2400,1)';
T_v = A(1501:2400,2)';
P_te = A(2401:3001,1)';
T_te = A(2401:3001,2)';
% P_te = B(1:8000,1)';
% T_te = B(1:8000,2)';

% Define input and output layer matrix
I= [0 1];
O= [1 2];

% naming TF
p = 'purelin'; t = 'tansig'; l = 'logsig';

% Setup network
% net=newcf(minmax(P_tr), [37 20
1],{'logsig','logsig','logsig'},'trainrp','learngdm','mse');
% net = newnarx(PR,ID,OD,[S1 S2...SN1],{TF1 TF2...TFN1},BTF,BLF,PF)
narx_net = newnarx(minmax(P_tr),I,O,[4 1],{p,t});
% narx_net=newnarx(minmax(P_tr),I, O, [2 2
1],{t,p,p},'trainrp','learngdm','mse');

%
narx_net.trainParam.show=5;
narx_net.trainParam.epochs=500;
narx_net.trainparam.goal=1e-4;

% Train network with early stopping
rand('seed',5270000);
narx_net = init(narx_net);

%% Set up the validation and testing sets in a structure form
val.P=P_v; val.T=T_v;
test.P=P_te; test.T=T_te;
[net tr] = train(narx_net,P_tr,T_tr,[],[],val,test);
% [net tr] = train(net,P_tr,T_tr,[],[],[],[]);

% Simulate network
a = sim(net, P_te);


% figure(1)
% [slope,intercept,R] = postreg(a,T_te);

% Actual min max of the data set
T_temin = 0.047362505; T_temax = 0.054611961;

% Unnormalized data set
```

```
[row1,col1] = size(T_te);
unnorm_Tte = zeros(1,1:col1);
for j = 1:col1;
    unnorm_Tte(j) = T_te(j)*(T_temax-T_temin)+T_temin;
    j = j+1;
end
unnorm_a = zeros(1,1:col1);
for jj = 1:col1;
    unnorm_a(jj) = a(jj)*(T_temax-T_temin)+T_temin;
    jj = jj+1;
end

figure(6)
time = 1:length(T_te);
plot(time,T_te,'-',time,a,'d'),...
xlabel('Time (min)'), ylabel('Actual vs predicted Output'),...
legend('Actual','NN')
grid on;

% rmse calculation
[row1,col1] = size(T_te);
error_col = zeros(1,1:col1);
for i = 1:col1;
    error_col(i) = (unnorm_a(i) - unnorm_Tte(i))^2;
    i = i+1;
end
sum_error = sum(error_col);
rmse = sqrt(sum_error/col1)

%%% CDC calculation
d1=zeros(1,col1-1);
ii=2;
for iii=1:col1-1
    ai=unnorm_Tte(:,ii) - unnorm_Tte(:,ii-1);
    bi=unnorm_a(:,ii) - unnorm_a(:,ii-1);
    ci=ai*bi;
    d1(:,ii-1)=ci;
    ii=ii+1;
    iii=iii+1;
end

Dt1=zeros(1,col1-1);
jjj=1;
for jjjj=1:col1-1
    if d1(:,jjj)>0
        Dt1(:,jjj)=1;
    else
        Dt1(:,jjj)=0;
    end
    jjj=jjj+1;
    jjjj=jjjj+1;
end

[row2,col2] = size(Dt1);
CDC = (sum(Dt1))*(100/(col2))
```

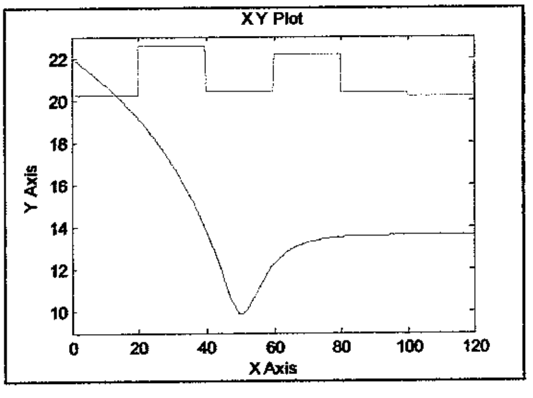
APPENDIX 2: Result of Different Network's Architecture (Different number of neurons)

i. Log-sigmoid and Log-sigmoid

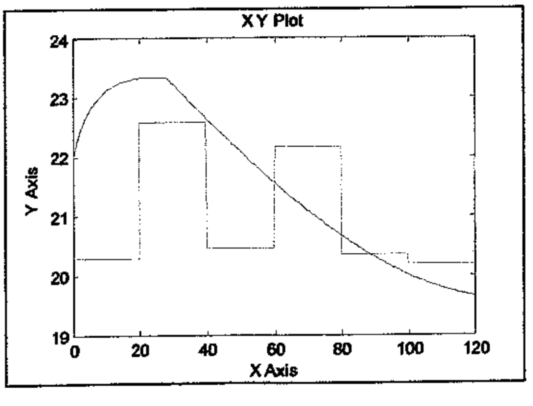
For 2 neurons:



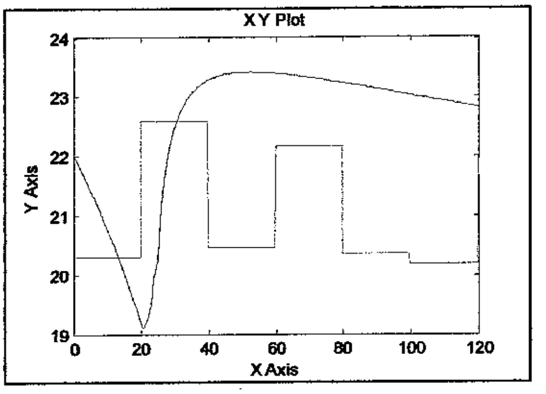
For 4 neurons:



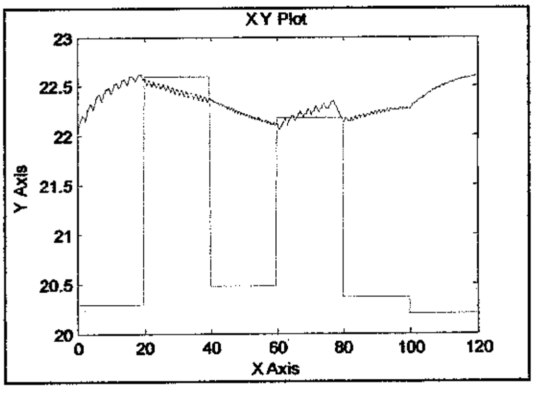
For 6 neurons:



For 8 neurons:

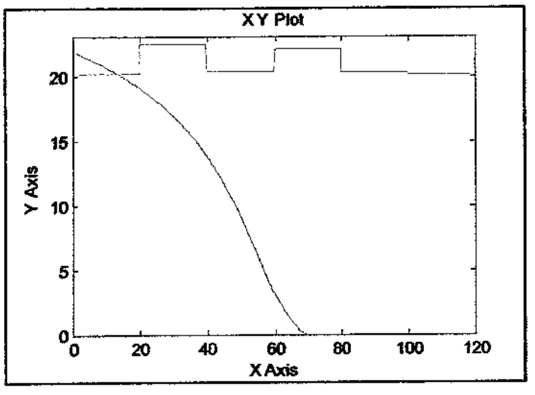


For 10 neurons:

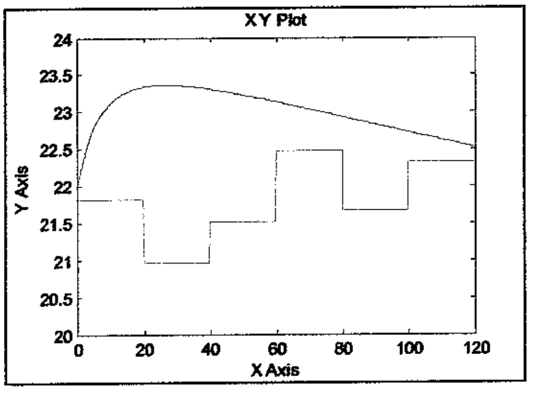


ii. Log-sigmoid and Purelin

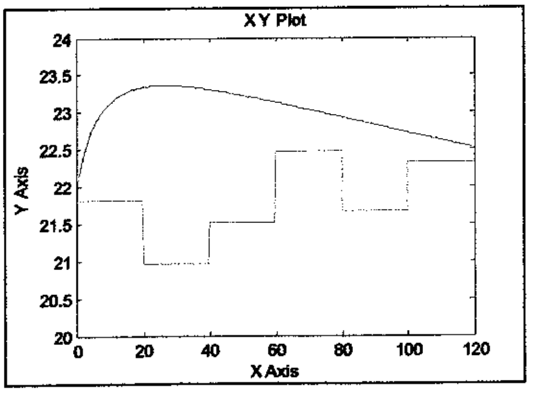
For 2 neurons:



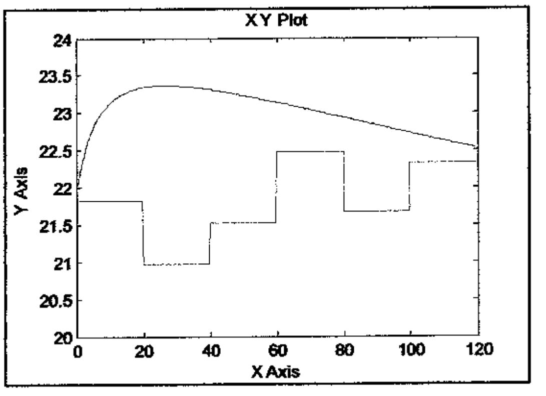
For 4 neurons:



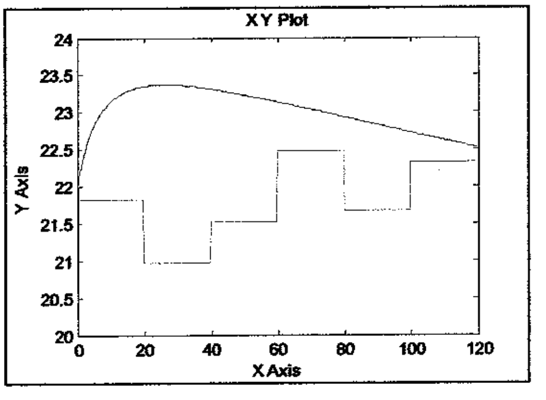
For 6 neurons:



For 8 neurons:

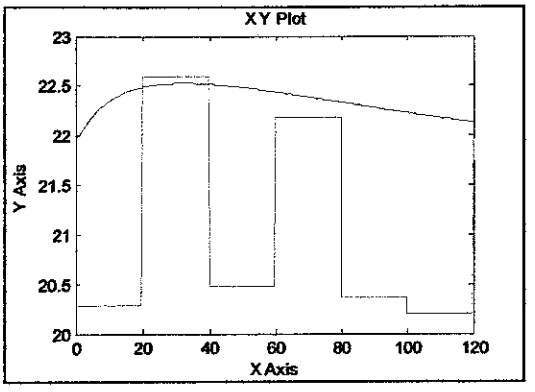


For 10 neurons:

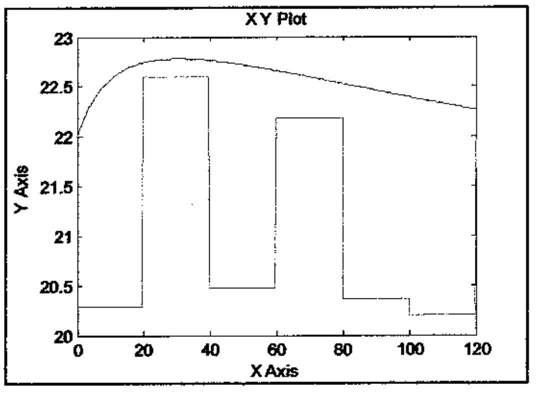


iii. Log-sigmoid and Tan-sigmoid

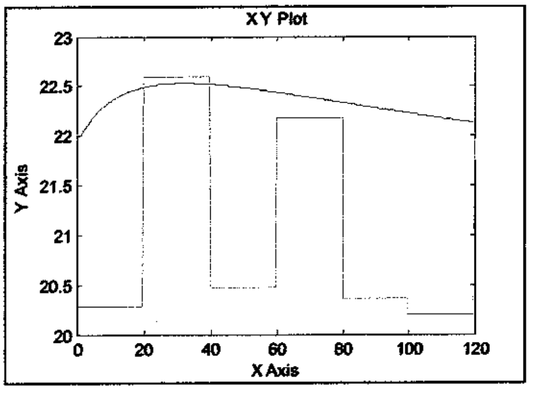
For 2nuerons:



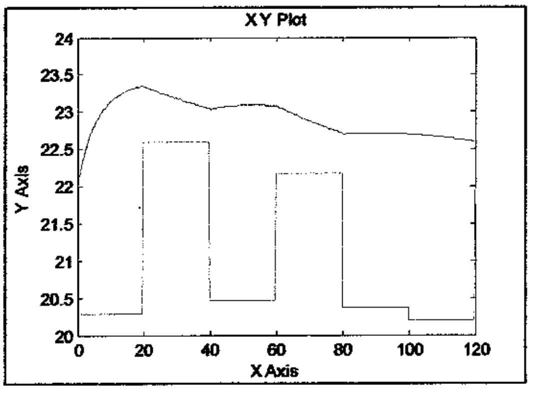
For 8 neurons:



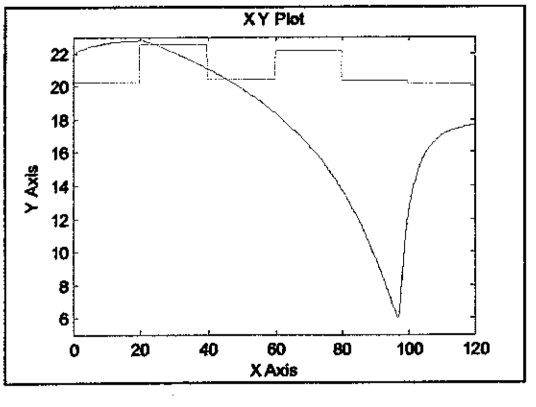
For 4 neurons:



For 10 neurons:

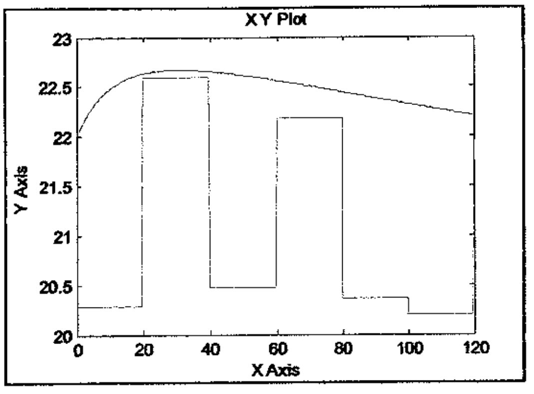


For 6 neurons:

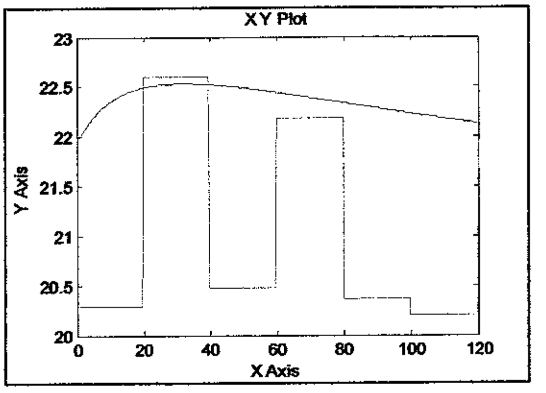


iv. Tan-sigmoid and Tan-sigmoid

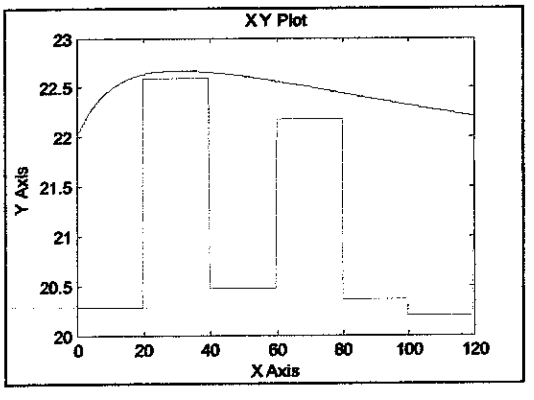
For 2 neurons:



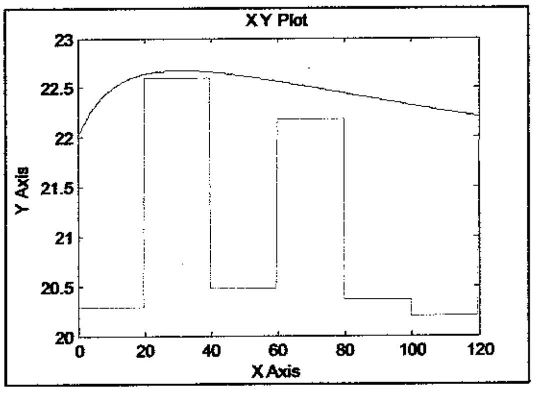
For 8 neurons:



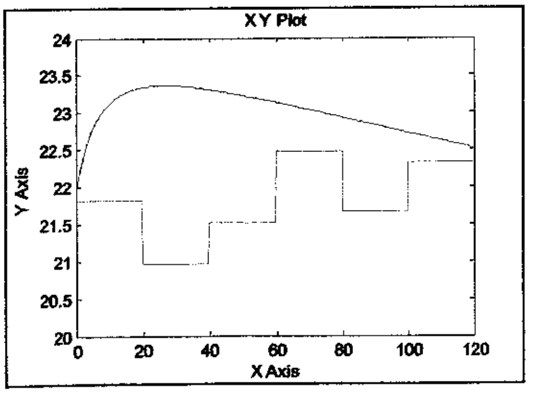
For 4 neurons:



For 10 neurons:

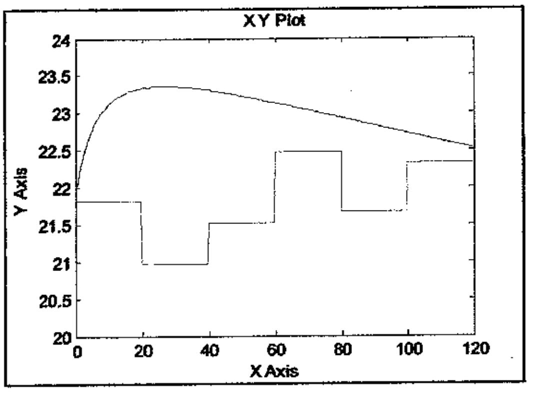


For 6 neurons:

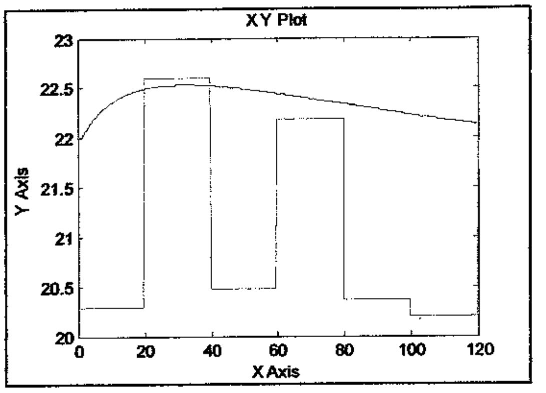


v. Tan-sigmoid and Log-sigmoid

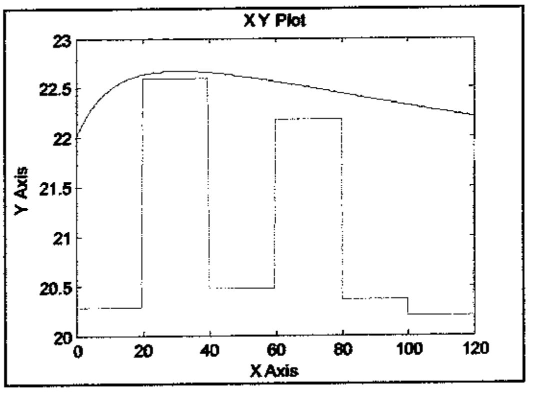
For 2 neurons:



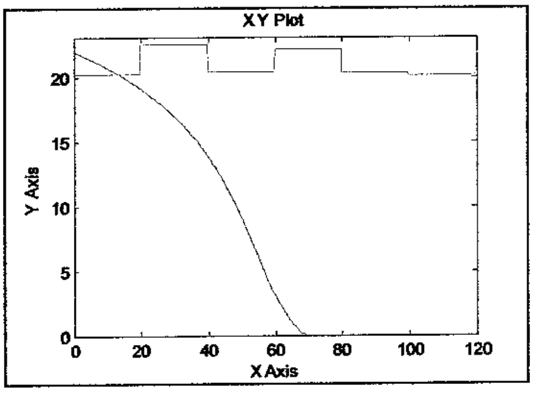
For 8 neurons:



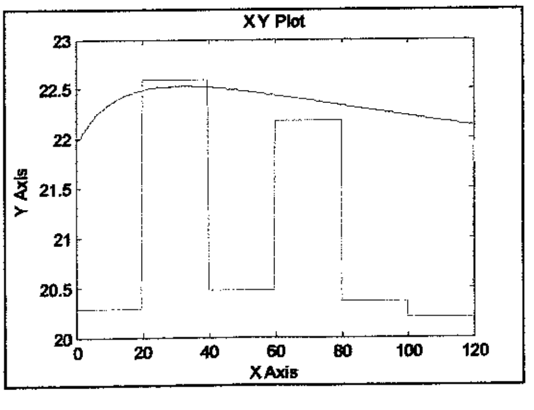
For 4 neurons:



For 10 neurons:

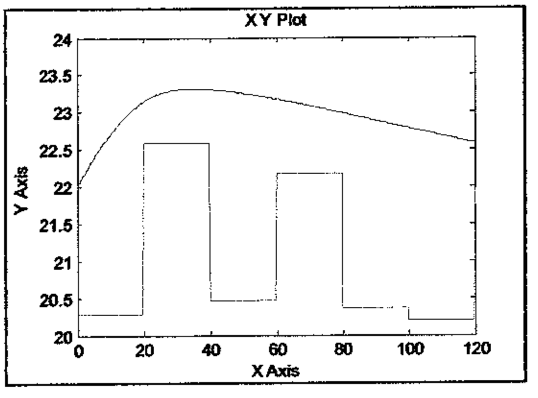


For 6 neurons:

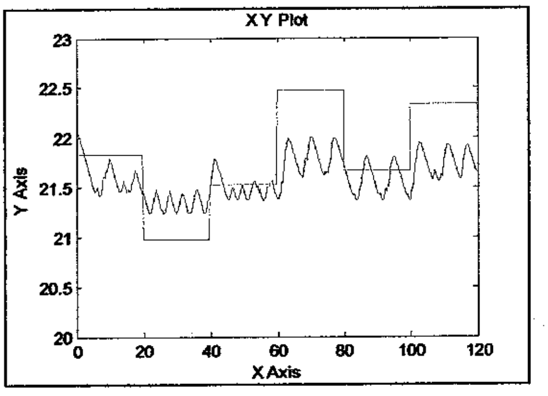


vi. Purelin and Purelin

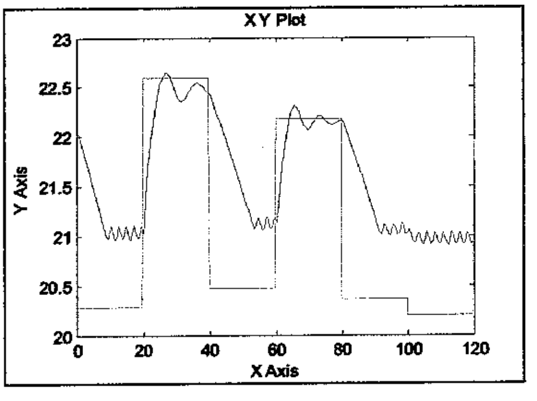
For 2 neurons:



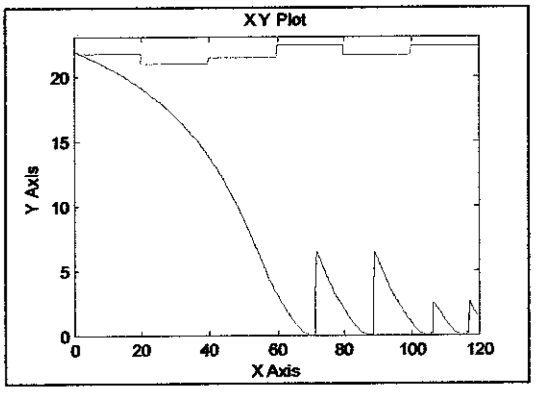
For 8 neurons:



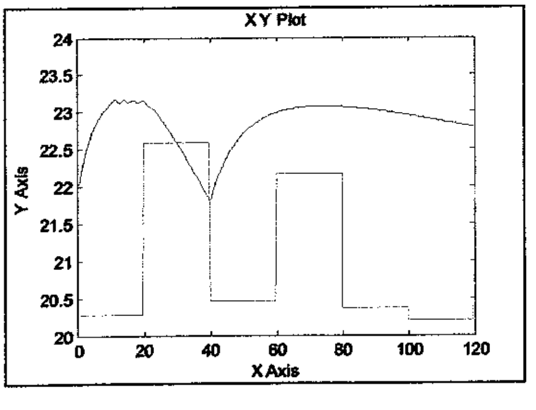
For 4 neurons:



For 10 neurons:

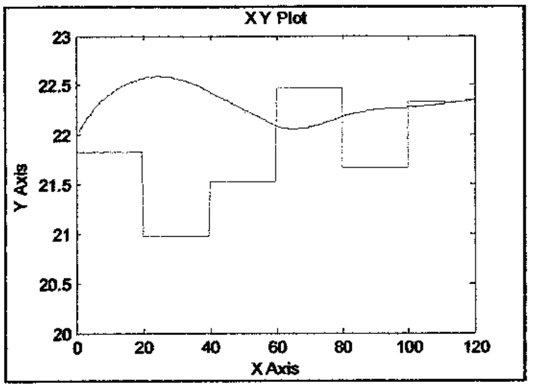


For 6 neurons:

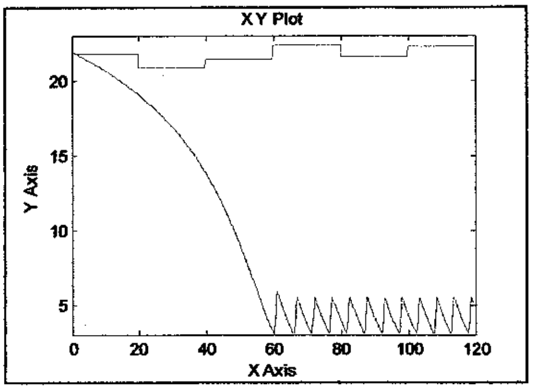


vii. Purelin and Log-sigmoid

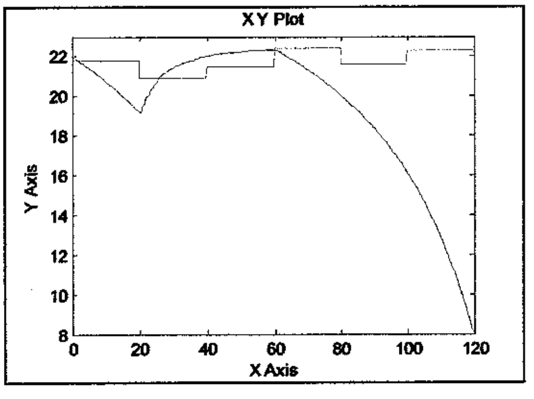
For 2 neurons:



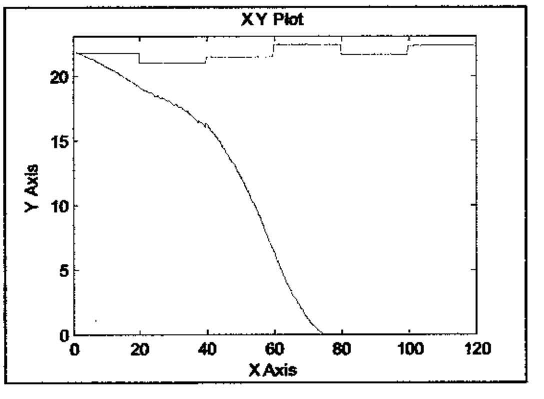
For 8 neurons:



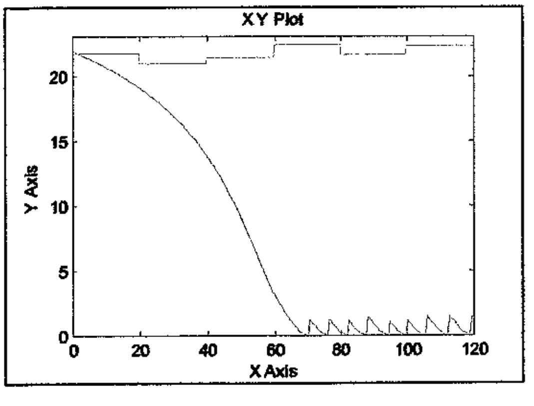
For 4 neurons:



For 10 neurons:

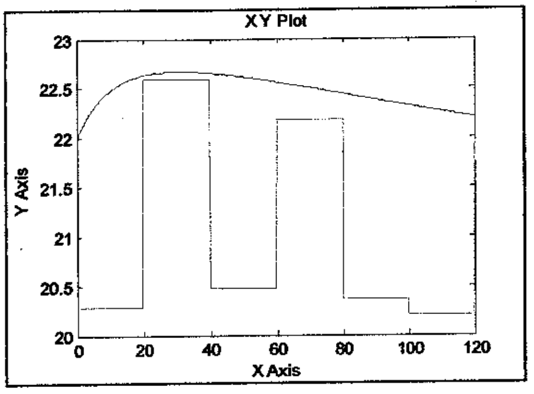


For 6 neurons:

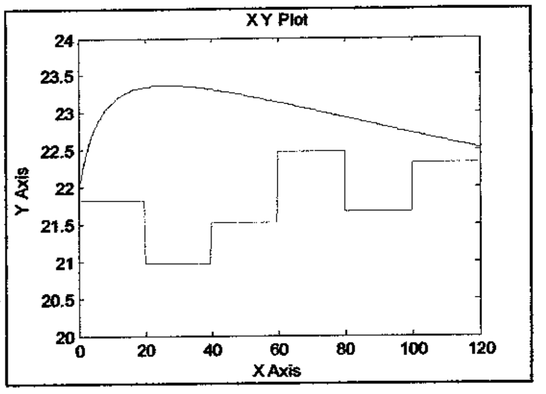


viii. Purelin and Tan-sigmoid

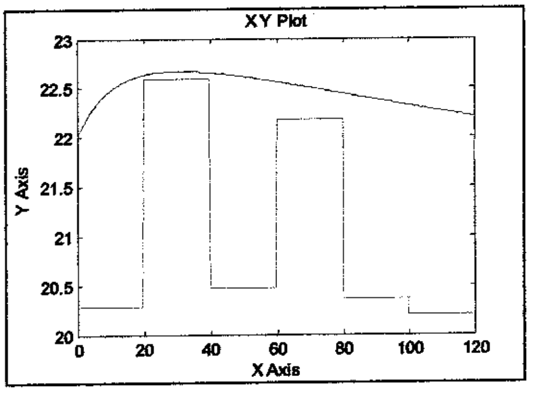
For 2 neurons:



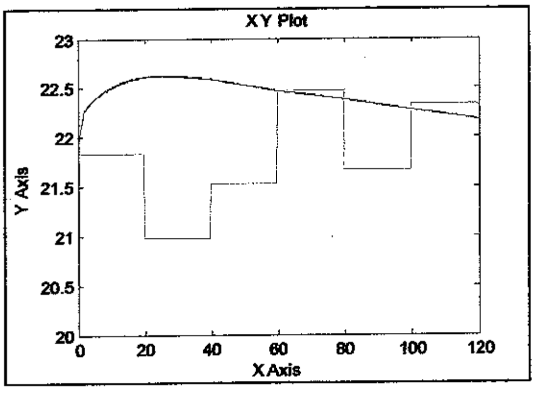
For 8 neurons:



For 4 neurons:



For 10 neurons:



For 6 neurons: