SYSTEMATIC DESIGN OF SIMPLY STRUCTURED COMPENSATOR

by

Nor Aisyah binti Ghazali

Dissertation submitted in partial fulfillment of

The requirements for the

Bachelor of Engineering (Hons)

(Electrical and Electronic Engineering)

DECEMBER 2010

Universiti Teknologi PETRONAS

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

Systematic Design of Simply Structured Compensator

by

Nor Aisyah binti Ghazali

A project dissertation submitted to the
Electrical and Electronic Engineering Programme:
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL AND ELECTRONIC ENGINEERING)

Approved by,

…….
(MS. ZAZILAH MAY)

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK
December 2010

CERTIFICATION OF ORIGINALITY

This is to 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 acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.

_____

NOR AISYAH BINTI GHAZALI

ABSTRACT

PID Controller is one of the most common controllers used in the industry. It is used to minimize the error between the input and the output in any operation of a plant. In order to get the best performance for the plant, the controller has to be tuned correctly. Currently, there are many tuning methods such as Ziegler-Nichols, Cohen-Coon and many others. This project aims to fine-tune the PID Controller for the most optimum tuning value of a PID Controller so that the output of the plant has the best response. Also, in order to get the most optimum tuning value, this project makes use of a Neural Network model to get that value of a PID Controller. Neural Network was chosen because it has the ability to learn from past error and from that, it will adjust the network accordingly, so that the desired output is reached. Other than using Neural Network for tuning the PID Controller to its most optimum value, Neural Network also can be used as a controller itself. So, this project also aims to compare the output response of a plant by using the Neural Network to fine tune the PID Controller and using the Neural Network alone as a controller. This project revolves around on Laboratory Experiments and MATLAB and Simulink simulations to get the most optimized output plant response.

ACKNOWLEDGEMENTS

First and foremost, I would like to praise Allah the Almighty for with His blessings and guidance I am able to successfully complete my final year project. My appreciation goes to my Supervisor, Ms Zazilah May for the advice, plans, and also for the knowledge and experiences shared during my attachment under her.

My gratitude is extended to all lecturers for their respective professionalism and all the knowledge shared that helped me to go through with this project. Also, a million thanks to all staffs and technicians at lab for the continuous help, support and guidance from the beginning of the final year project until the end.

Finally, my acknowledgement is considered incomplete without thanking all my fellow colleagues, and family who have been giving great support and encouragement for me to complete the project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Background of Study

PID controller is one of the most common controllers used in feedback control for industrial process systems. It is used extensively because of its robustness and simple structure. A survey was done in Japan indicated that more than 90% of the controllers used in process industries are PID controllers and advanced versions of the PID controller [7]. In PID controller calculation, involves three parameters which are the proportional term (K_p), the integral term (K_i), and the derivative term (K_d).

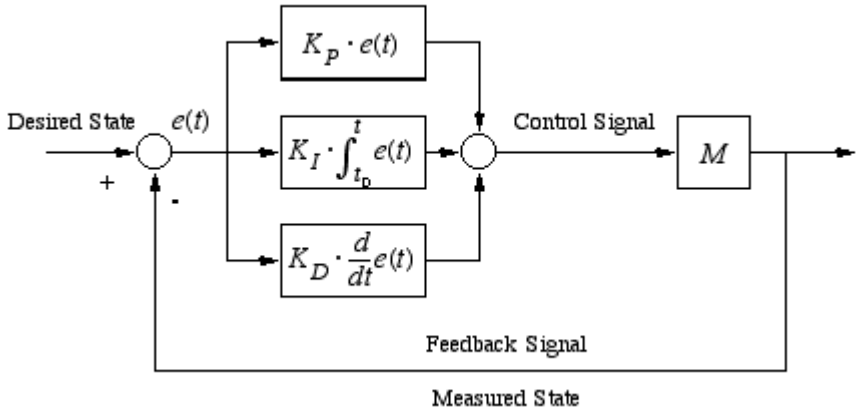


Figure 1 : General Form of a PID Controller.[10]

Tuning of a PID controller is the procedure of adjusting the controller parameters. The tuning of three parameters in PID controller is important for the controller to give the appropriate control action based on the particular

process requirements of the system. Otherwise, the system may become unstable.

This project is based on a previous Final Year Project (FYP) [3] which focused on developing the algorithm of the fine-tuning method based on Nyquist Stability Criterion and demonstrates that it can be used to optimize the performance of a system. Also, a Neural Network model was built to predict the tuning parameters of a PID controller. However, the Neural Network model was not tested on a real system to prove that it could optimize a system's performance.

Hence, this project will focus on testing the Neural Network model based on a real system.

1.2 Problem Statement

The tuning of PID controller's parameters is quite difficult especially in obtaining the desired and most optimize result. The problem is in determining the parameters correctly such that the output response is as preferred.

The existing tuning techniques also have some limitations in the tuning rule itself. So, it might only be suitable only for a small subset of systems [3]. Also, it is also found that the Ziegler–Nichols rules also have severe drawbacks; they use insufficient process information and the design criterion that gives closed loop systems with poor robustness [5].

1.3 Objectives and Scope of Study

1.3.1 Objectives

The objectives of this project are as follows:

1) To develop the Neural Network Model to fine-tune the PID controller.
2) To develop the Neural Network Predictive Controller.
3) To compare the Neural Network model of PID Controller and the Neural Network Predictive Controller.

1.3.2 Scope of study

This study of this project will revolve on building and improving the Neural Network model by using the MATLAB and Simulink simulation based on a real plant system.

CHAPTER 2

LITERATURE REVIEW

2.1 PID Controller

The study on PID controller is done because it is one of the most common algorithms used due to its simplicity of implementation, robustness, applicable in a wide range and low in cost. Even though it is simple in structure, the tuning of PID controller itself is a big challenge [3]. Researches on the best method in tuning a PID controller are still ongoing. There are a few techniques in tuning the PID Controller such as computational methods, intelligent systems, genetic algorithm, fuzzy systems and neural network [6].

Controllers are designed to get rid of the need for constant operator attention and they are used to adjust some variable for the process variable at the set point automatically. Set point is defined as the preferred value while error is defined as the difference between set point and measurement.

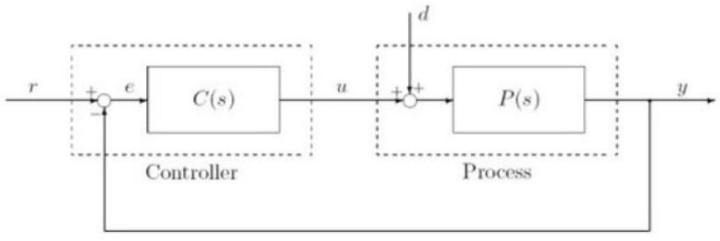


Figure 2: Conventional feedback control system [7]

2.1.1 PID Algorithm

There are three elements of the PID controller which generate outputs with the following condition [2, 7]:

a) Proportional mode is proportional to the error at the instant *t*, which can be interpreted as the "present" error. It provides fast response but does not reduce the offset to zero.

b) Integral mode is proportional to the integral of the error up to the instant *t*, which can be interpreted as the accumulation of the "past" error. It reduces the offset to zero but provides rather slow feedback compensation.

c) Derivative mode element is proportional to the derivative of the error at the instant *t*, which can be interpreted as the prediction of the "future" error. It takes action based on the derivative of the controlled variable.

The combination of the three modes is required to be able to obtain good control [2].

2.1.2 PID Controller Tuning

Tuning is the procedure of adjusting feedback controller parameters to obtain a specified closed-loop response [13]. There are two different approaches in tuning which are the open-loop and the closed-loop. The form of the PID controller is as follows [2]:

$$MV(t) = K_c \left[E(t) + \frac{1}{T_I} \int E(t')\,dt' - T_d \frac{dCV(t)}{dt}\right] + I \text{ -------- (1)}$$

The adjustable parameters are the tuning constants which are K_c, T_I and T_d [13]. To obtain the tuning constants, tuning correlation can be used. There are many tuning correlations such as Ciancone, Ziegler-Nichols, Cohen-Coon etc.

2.2 Neural Network

Neural network is one of the methods that have recently received extensive attention to overcome the difficulties for controller in process control and it offers promising results [4]. Also, there are revolutionary concepts and methods in neural network research which could affect scientific fields such as optimization theory as well as algorithms [1]. So, the study of tuning a PID Controller based on the building of Neural Network model may present with a satisfying result in concern with the optimization of a system.

Neural network are usually used for three reasons which are for fitting a function, recognizing patterns, and clustering of data. In this project, the neural network is used for the clustering of data which means grouping of data based on similarities [9]. Neural Network is useful when the data is very complex and design is too impractical to implement by hand [8].

2.2.1 Neural Network Structure

Neural networks are models of biological neural structures. Figure 3, is a model neuron. The neuron is usually consists of multiple inputs and a single output. Every input is modified by a weight, which multiplies with the input value. The neuron will combine these weighted inputs with reference to a threshold value and activation function. The output is determined by these values. [11]

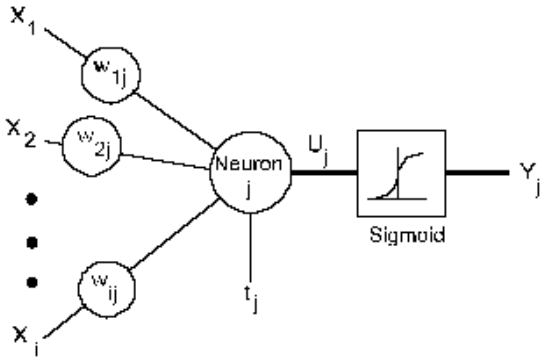


Figure 3: A Model Neuron [11]

2.2.2 Neural Network Learning

Learning in a neural network is also known as training. It would calculate the difference between the desired response and the actual response which is the error. The error is determined and a part of it is propagated backward through the network. At each neuron in the network, the error is used to adjust the weights and threshold values of the neuron, so that the next time, the error would be reduced for the same inputs. [11]

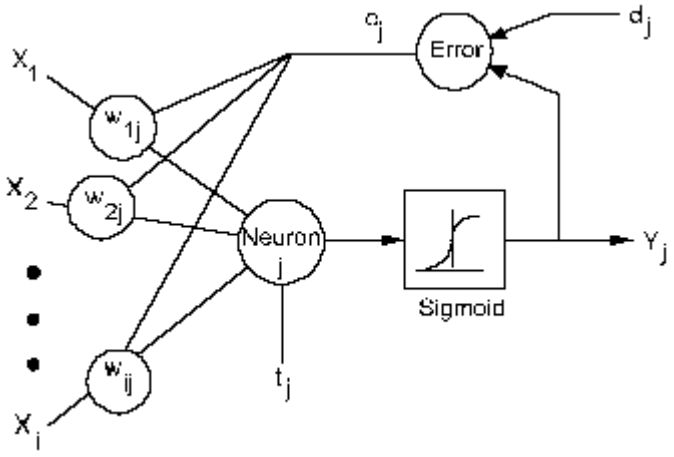


Figure 4: Neuron Weight Adjustment [11]

Backpropagation is the corrective procedure to reduce the error. It is applied repeatedly for each set of inputs and for its resultant outputs.. This procedure continues on as errors in the responses exceed a specified level or until there are no measurable errors. At this point, the neural network has learned the training material.[11]

Backpropagation has multiple-layer networks and nonlinear differentiable transfer functions. Input vectors and its corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors [9].

Properly trained backpropagation network tend to give reasonable answers when presented with inputs that it has never seen. Usually, a new input

leads to an output similar to the correct output for input vectors used in training that are similar to the new input being presented. This makes it possible to train a network on a representative set of input/target pairs and get good results without using all possible input/output pairs to train the network [9].

2.2.3 Neural Network Predictive Controller

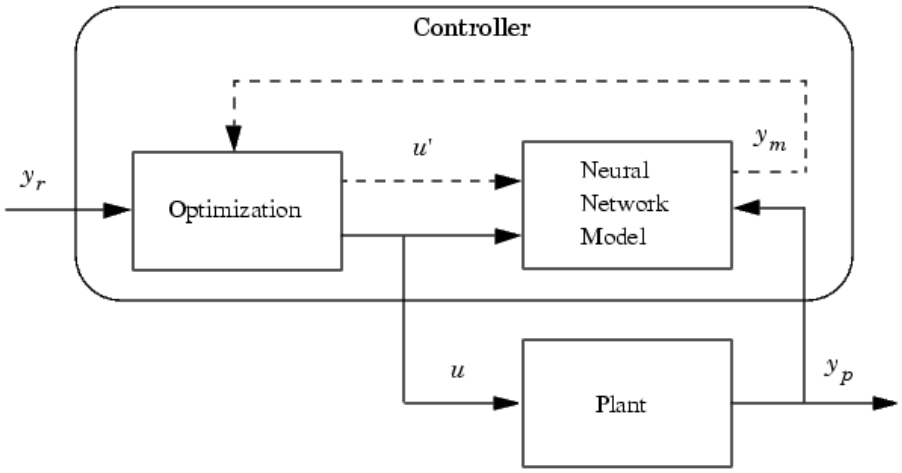


Figure 5: Neural Network Predictive Controller [9]

The neural network predictive controller implements neural network to predict future plant performance. The controller will calculate the input that will optimize plant performance in a particular time. Firstly, the neural network plant model is established. This is done by training the plant using neural network to represent forward dynamics of the plant. The error between the neural network output and the plant output is set as input. Then, the neural network plant model is used by the controller to predict future performance of the plant.[9]

2.3 The Process Reaction Curve

The process reaction curve is the most commonly used method for identifying dynamic models. It is simple and provides adequate models for many

applications. The process reaction curve method involves the following four actions [2]:

1) The process is allowed to reach steady state.
2) A single step change is introduced in the input variable.
3) Input and output response data is collected until the process reaches steady state again.
4) Perform calculations on the graphical process reaction curve.

The calculations would determine the parameters of the system for a first-order with dead time model. There are two different techniques for the graphical calculations which are Method I and Method II.

2.3.1 Method I

The general model for a step input with t ≥ θ is

$$Y'(t) = Kp\ \partial\ [1 - e^{-\frac{t-\theta}{\tau}}]$$ ---------- (2)

So, the slope for the response is

$$\frac{dY'(t)}{dt} = \frac{d}{dt}\left[Kp\ \partial\ \left(1 - e^{-\frac{t-\theta}{\tau}}\right)\right] = \frac{\Delta}{\tau}\ e^{-\frac{t-\theta}{\tau}}$$ ---------- (3)

The maximum slope take place at t = θ. So, the model parameters can be calculated as

$$Kp = \frac{\Delta}{\partial}$$ ---------- (4)

$$\tau = \Delta/S$$ ---------- (5)

θ = intercept of maximum slope with initial value

2.3.2 Method II

The values from the process reaction curve can be related to the model parameters from the general equation which is equation 2. Basically, any two time values can be selected to determine unknown parameters, θ and τ

but usually the times are selected where the transient response is changing rapidly so the parameters can be determined even with noise.[2]

$$Y\ (\theta + \tau) = \ \Delta\ (1 - e^{-1}) = 0.632\Delta \text{ ---------- (6)}$$

$$Y\ (\theta + \tau/3) = \ \Delta\ \left(1 - e^{-1/3}\right) = 0.283\Delta \text{ ---------- (7)}$$

So, to calculate the model parameters, the values of time when the output reaches 28.3 and 63.2 percent are taken.

$$t_{28\%} = \ \theta + \ \tau/3 \text{ ---------- (8)}$$

$$t_{63\%} = \ \theta + \ \tau \text{ ---------- (9)}$$

So, $\tau = 1.5(t_{63\%} - \ t_{28\%})$ and $\theta = \ t_{63\%} - \tau$.

CHAPTER 3

METHODOLOGY

3.1 Project Flow

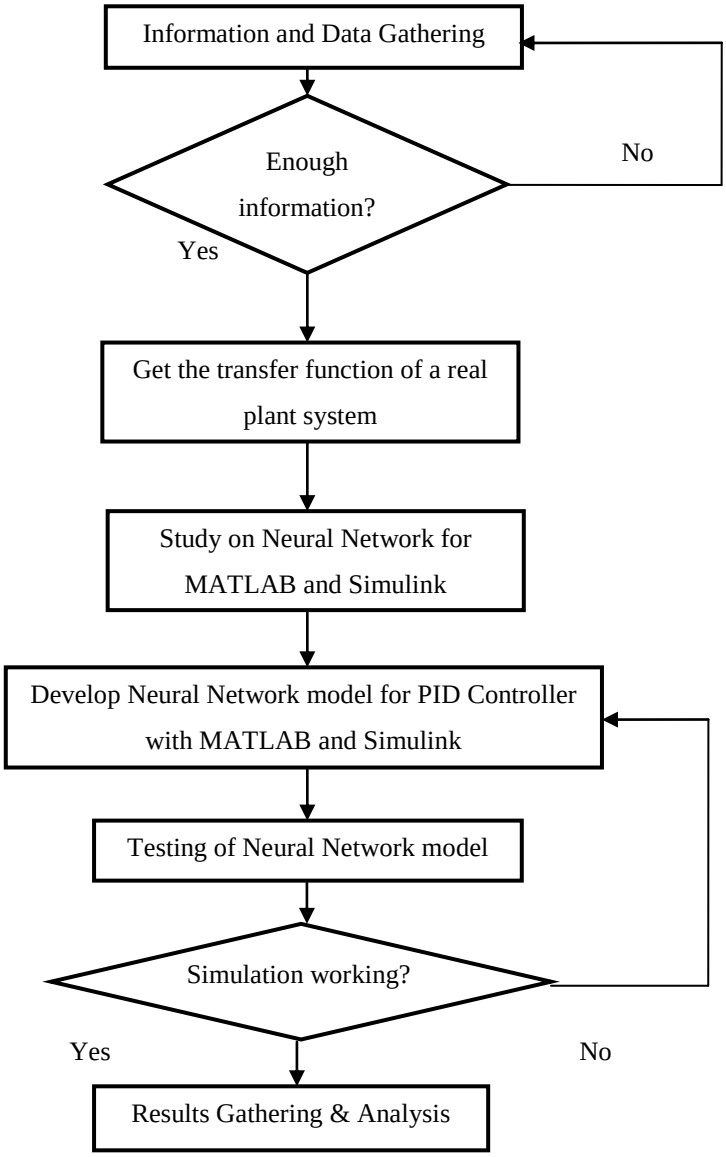


Figure 6: Project Flow for Semester I

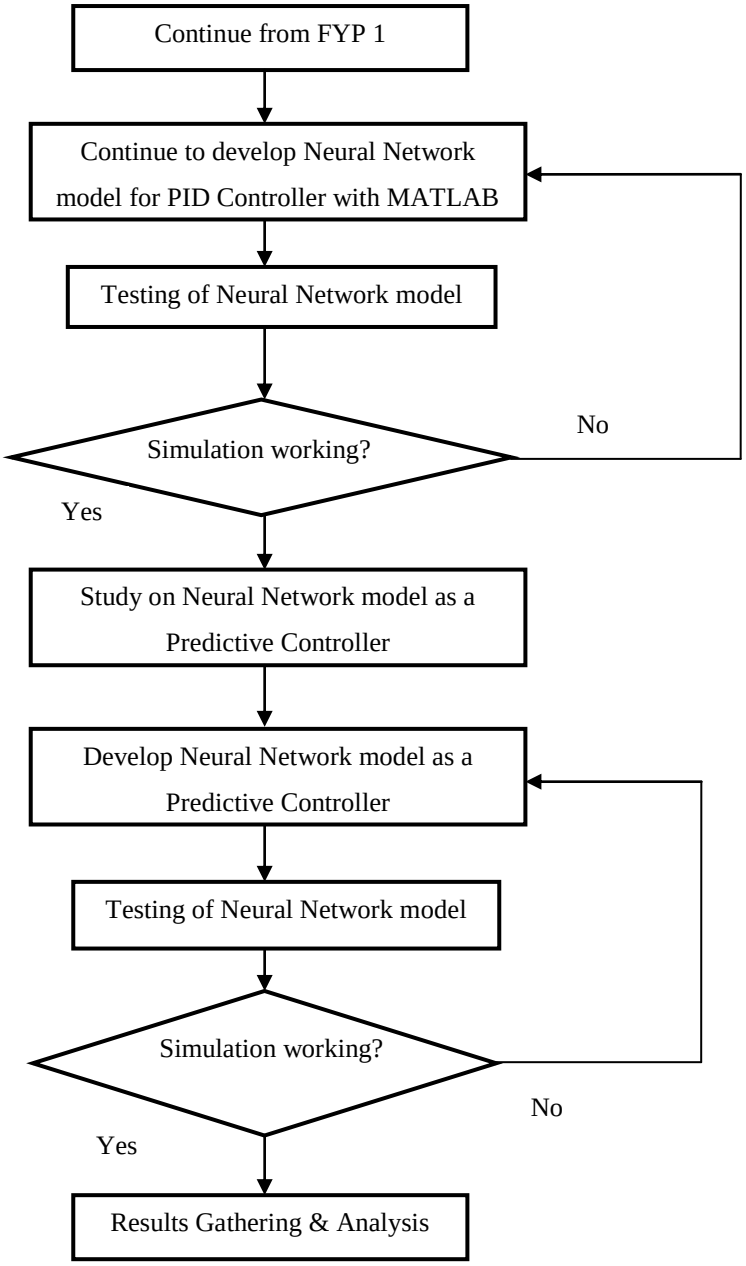


Figure 7: Project flow for Semester II

3.2 Laboratory Experiment Procedure

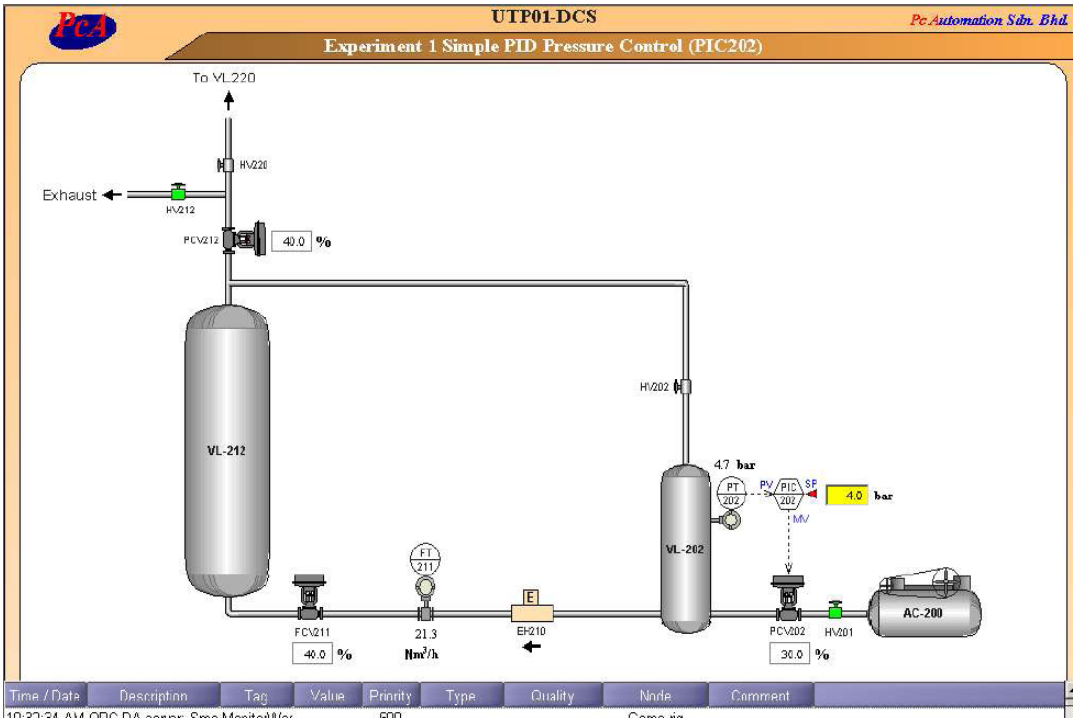


Figure 8: Plant for Simple PID Pressure Control

Based on Figure 8, the experiment is to control the pressure on vessel VL-202. The set point of the pressure in the vessel is set to 4.0 bar. The experiment was done by changing the manipulated variable which is the opening of the valve, PCV-202. The output response would be the actual pressure in VL-202.

The transfer function of the system was obtained by calculating the parameters of the system using the Process Reaction Curve, Method II. This is because Method I is more prone to errors because of the need for the maximum slope calculation. The form of the model is as follows:

$$\frac{Y(s)}{X(s)} = \frac{K_p e^{-\Theta s}}{\tau s+1} \quad \text{---------- (10)}$$

Where, K_p = (change in output)/(change in input)

$\Theta = t_{63\%} - \tau$

$$\tau = 1.5 (t_{63\%} - t_{28\%})$$

3.3 Tools and Software

Tools and software required for this project are:

- MATLAB
- Simulink

3.3.1 Neural Network Graphical User Interface (GUI)

The Graphical User Interface (GUI) of Neural Network as in Figure 9 is used to handle the data to train the Neural Network model. The data for the training, validation and testing of the inputs are imported to the GUI of Network / Data Manager of MATLAB.

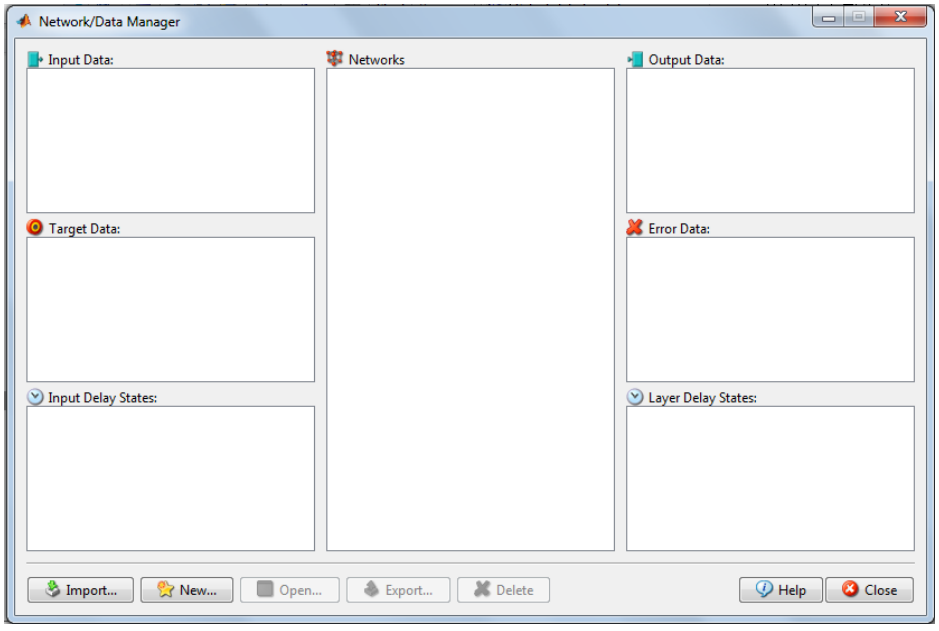


Figure 9: GUI of Neural Network

Then, it will train the network after the network is constructed using the GUI as in Figure 10.

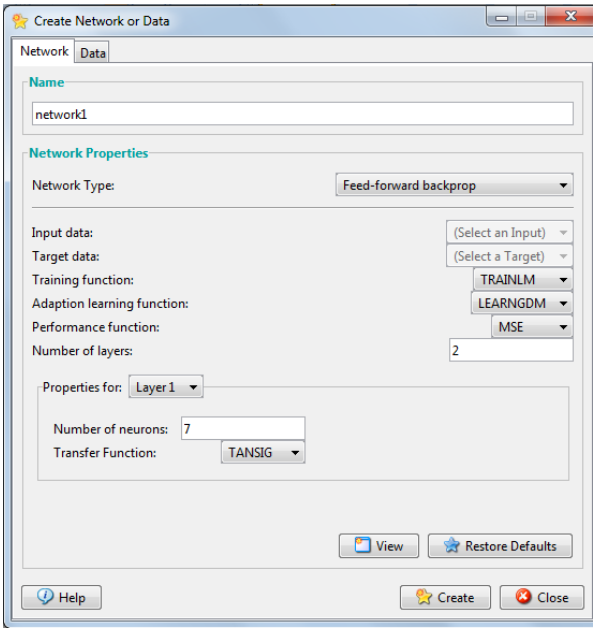


Figure 10: Create a network

3.3.2 Neural Network Predictive Controller

After having the results for the Neural Network model to fine tune the PID Controller, the project is continued with building the Neural Network Predictive Controller where the Neural Network itself act as a controller for the plant. Figure 11 shows how the simulated plant would look like and Figure 12 shows the GUI for the Neural Network Predictive Controller.

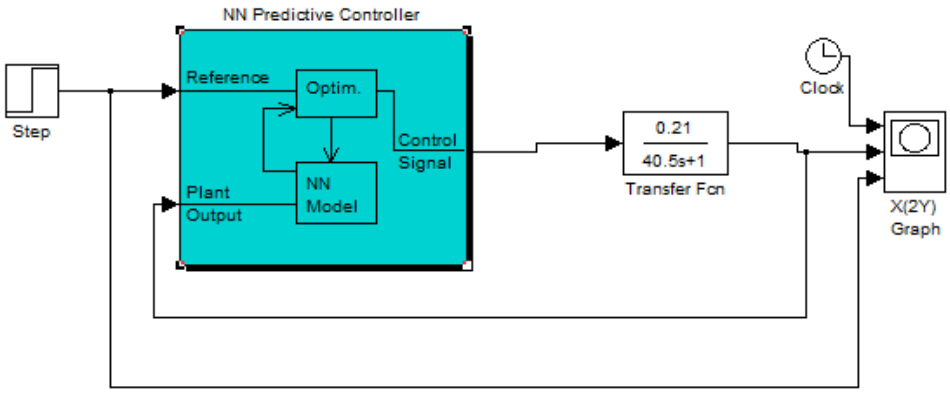


Figure 11: Plant model using NN Predictive Controller.

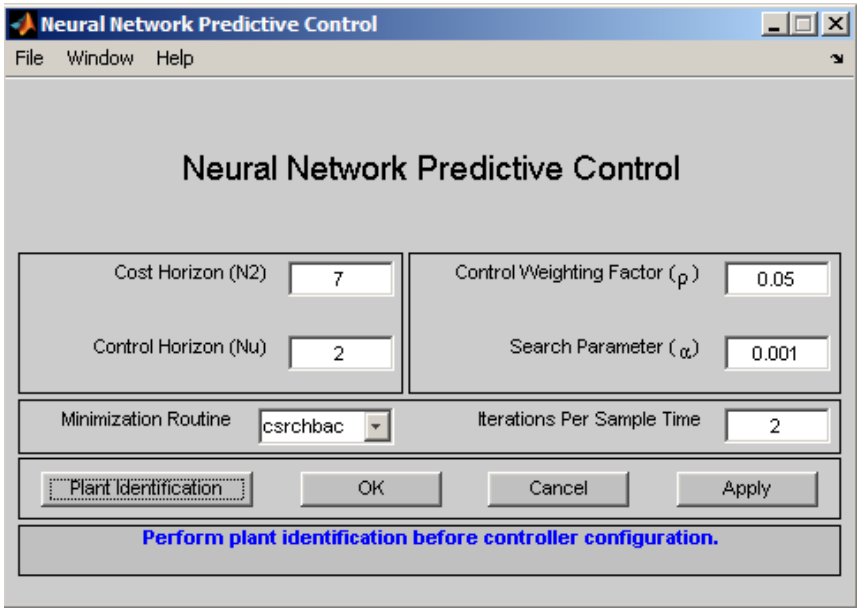


Figure 12: Neural Network Predictive Control GUI

CHAPTER 4

RESULT AND DISCUSSION

4.1 Result from Laboratory Experiment

The result from the laboratory experiment for the Simple PID Pressure Control was obtained. The Process Reaction Curves of the experiment for two different valve opening are as in Figure 13 and Figure 14.

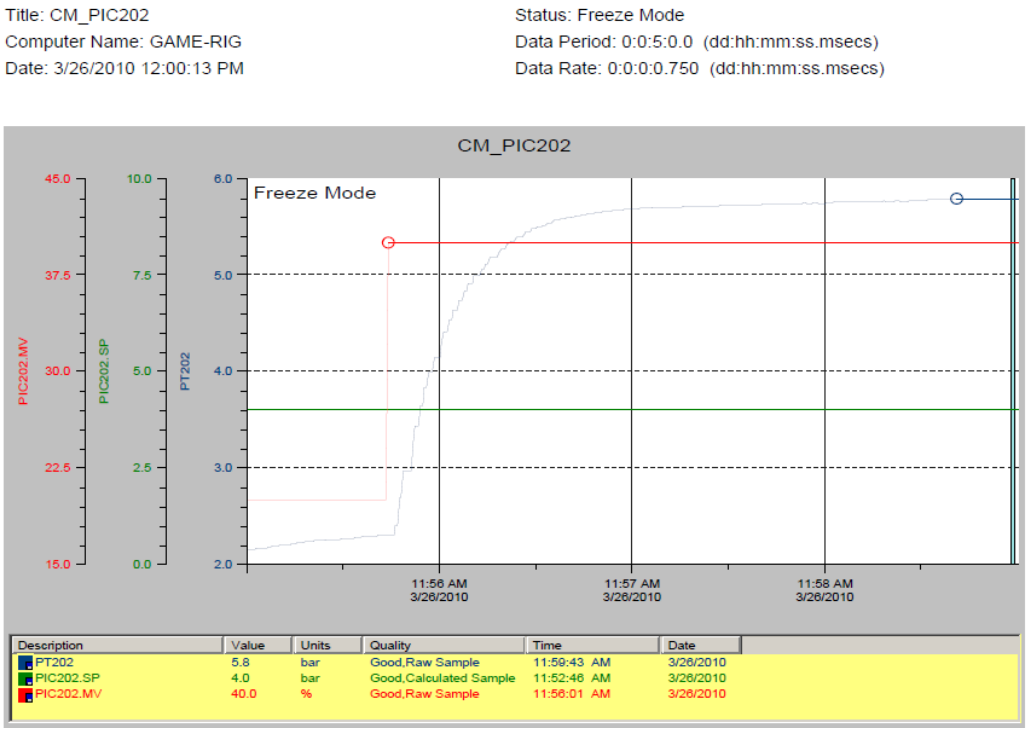


Figure 13: Process Reaction Curve with Valve Opening from 20 – 40%

From the Process Reaction Curve with Valve Opening from 20 – 40% in Figure 13, the parameters were obtained by using the Method II as shown:

$$Kp = \frac{\text{change in output}}{\text{change in input}} = \frac{5.8 - 2.3}{40 - 20} = 0.175 \text{ ---------- (11)}$$

$$\tau = 1.5\,(t_{63\%} - t_{28\%}) = 1.5\,(11.875) = 17.8 \text{ sec ---------- (12)}$$

$$\Theta = t_{63\%} - \tau = 20.625 - 17.8 = 2.8 \text{ sec ----------(13)}$$

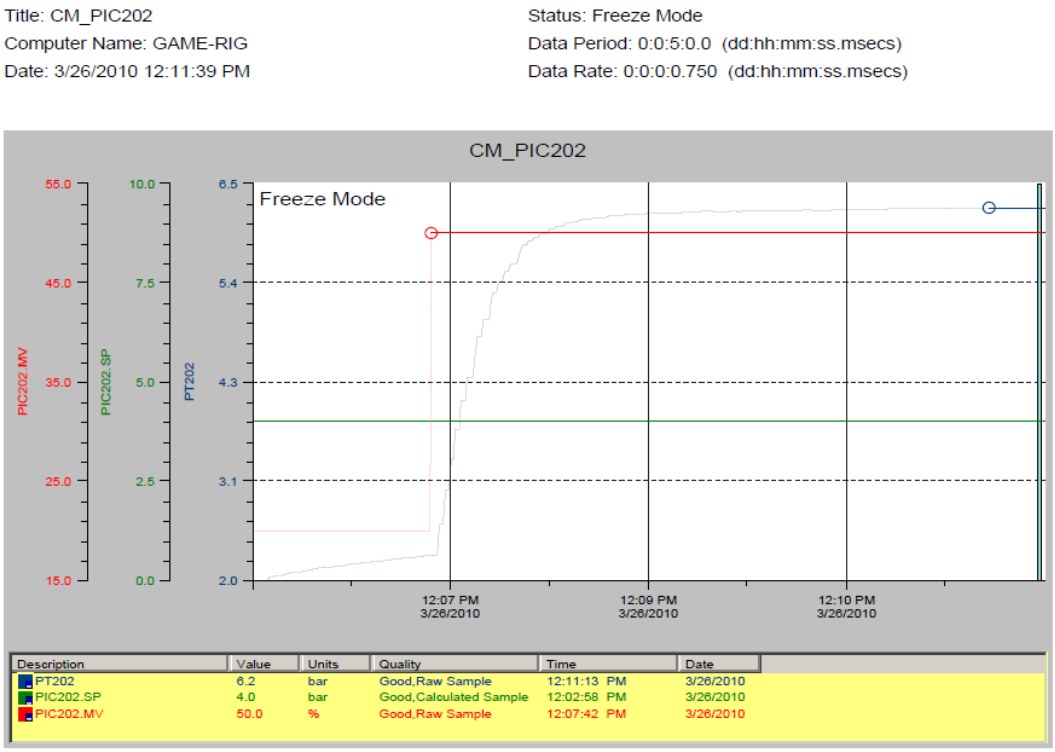


Figure 14: Process Reaction Curve with Valve Opening from 20 – 50%

From the Process Reaction Curve with Valve Opening from 20 – 50% in Figure 14, the parameters were obtained by using the Method II as shown:

$$Kp = \frac{\text{change in output}}{\text{change in input}} = \frac{6.2 - 2.26}{50 - 20} = 0.13 \text{ ---------- (14)}$$

$$\tau = 1.5\,(t_{63\%} - t_{28\%}) = 1.5\,(15) = 22.5 \text{ sec ----------(15)}$$

$$\Theta = t_{63\%} - \tau = 27.5 - 22.5 = 5 \text{ sec} \quad \text{---------- (16)}$$

From all the parameters obtained, the average value is obtained for a more accurate value of the parameters as shown below:

$$Kp = \frac{0.175 + 0.13}{2} = 0.153 \quad \text{---------- (17)}$$

$$\tau = \frac{17.8 + 22.5}{2} = 20.15 \quad \text{---------- (18)}$$

$$\Theta = \frac{2.8 + 5}{2} = 3.9 \quad \text{---------- (19)}$$

From eqn. (10) and the parameters from (17), (18) and (19), the transfer function of the plant is obtained as follows:

$$\frac{Y(s)}{X(s)} = \frac{0.153\, e^{-3.9s}}{20.15s+1} \quad \text{---------- (20)}$$

4.2 Results from MATLAB and Simulink

The transfer function obtained was simulated in Simulink to get the data for the Neural Network building. The model built in Simulink is as in Figure 14.

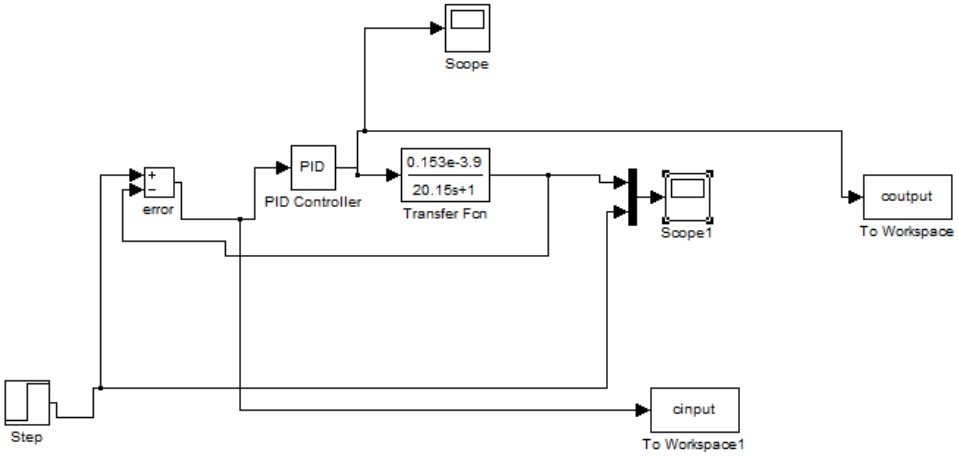


Figure 15: Plant Model in Simulink

To verify the transfer function, simulations on five different step inputs are done. The graphs obtained are about the same for each different step inputs based on Appendix C. Therefore, the transfer function is verified to be functional and can be used as the plant for the neural network training.

The model in Figure 15 was done to obtain the values from the input and output of the PID Controller so that the values could be used for the input and targets of the Neural Network model. The PID Controller was initially tuned using the Ziegler-Nichols Open Loop method based on process reaction curve.

Table 1: Results for Ziegler-Nichols Open Loop Tuning Correlations

	K_C	T_I	T_d
P-only	33.8	-	-
PI	30.4	12.87	-
PID	40.5	7.8	1.95

From the Simulink, the simulation of the plant was done and the result obtained for P-only, PI and PID controller are as in Figure 16, Figure 17 and Figure 18 respectively.

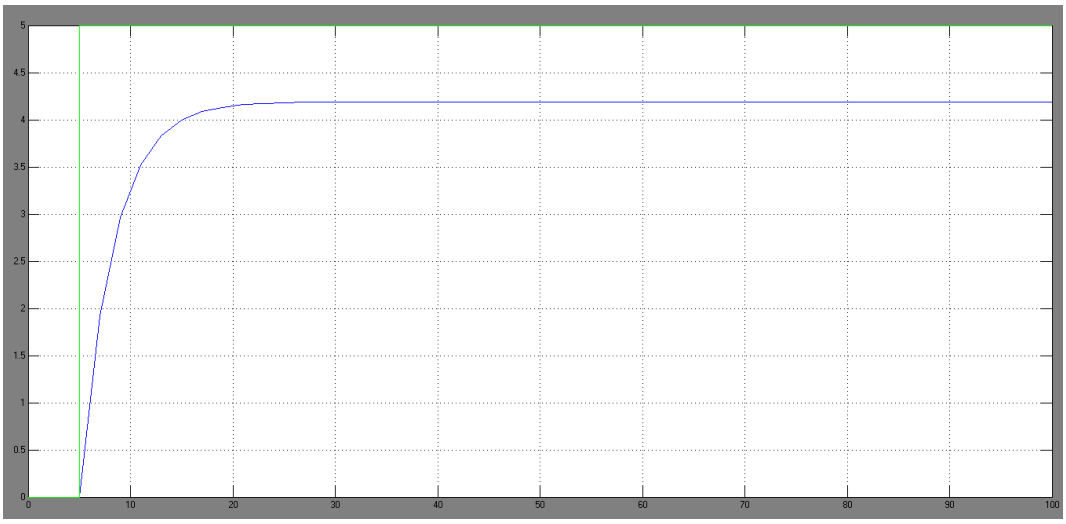


Figure 16: Result from Simulink for P controller

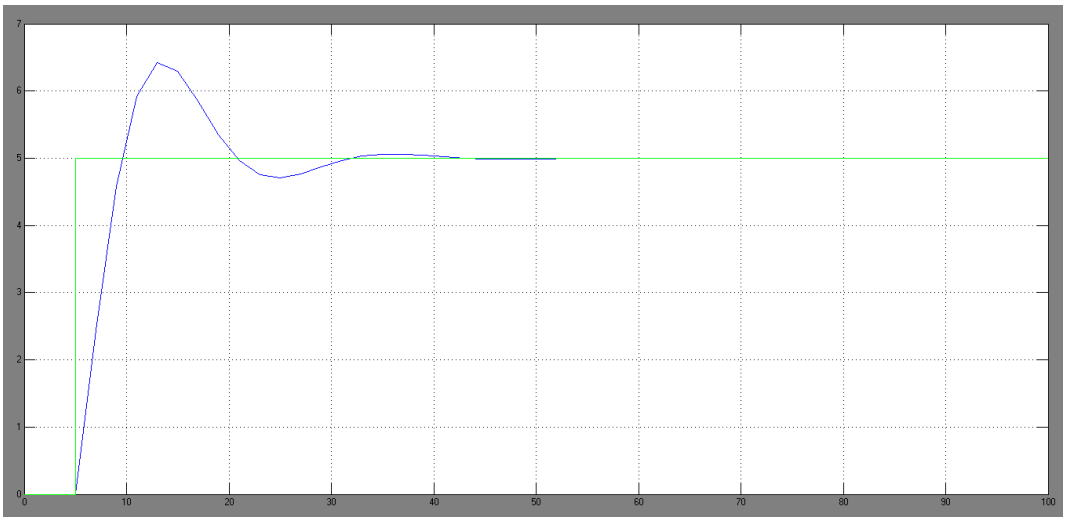


Figure 17: Result from Simulink for PI controller

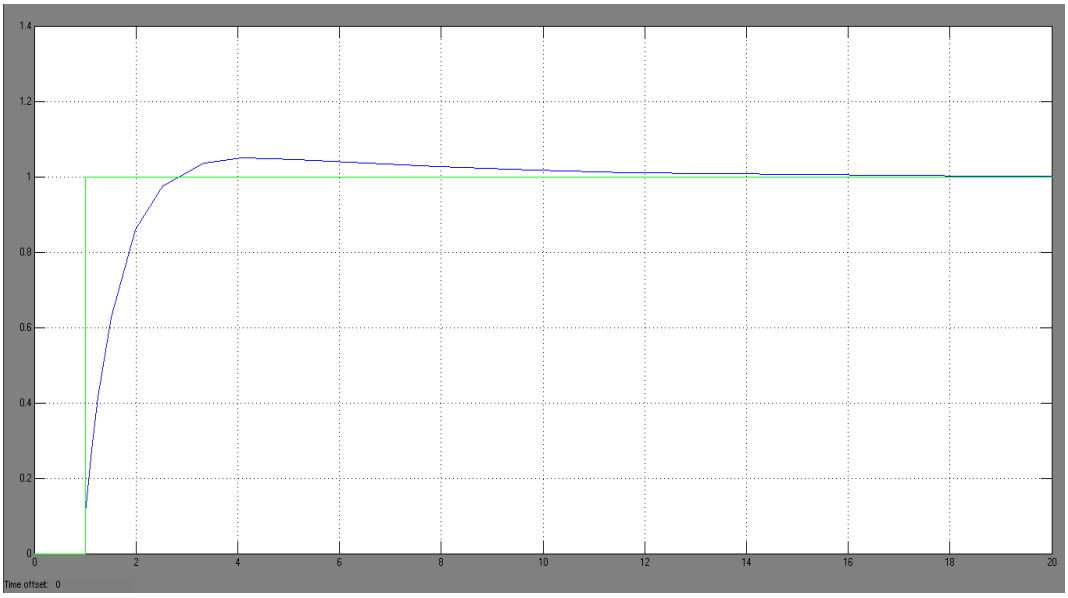


Figure 18: Result from Simulink for PID controller

Depends on the results of each controller, the response for P-only controller is not acceptable as it did not reach the zero offset. The best response for this plant is the PI controller as it has the lowest settling time and rise time which is 9.86 seconds and 1.5 seconds respectively as compared to the PID controller which the settling time is 14.28 seconds and its rise time is 1.8 seconds.

Since the PI controller is the best controller performance, the values obtained from the response will be used for the Neural Network model.

4.3 Neural Network for PID Controller Tuning

The input is the range of error signal while the target is a set of constant proportional and integral values. The initial values used for training was taken from the initial tuning of the PID Controller. For both Kp and Ti training, the data are randomly divided into three sets which are training, testing and validation.

The Neural Network training had the parameters put into as in Figure 10. The parameters are needed to create a network. It is as follows:

Table 2: Neural Network Training Parameters

Network type	Feed-forward Backpropagation
Training algorithm	Levenberg-Marquardt (trainlm)
Adaption learning function	learngdm
Performance function	MSE
Number of layers	2
Transfer function for Layer 1	TANSIG
Transfer function for Layer 2	PURELIN

The training style for this network is the batch training where weights are changed after all the inputs and targets have been computed. Feed-forward backpropagation was chosen as the network type because after the input and target are used to train a network, it can approximate a function that is associated the input with specific output.

The training algorithm used Levenberg-Marquadt or trainlm is used because it is usually able to obtain lower mean-squared error compared to other algorithms and it can compute the fastest convergence.

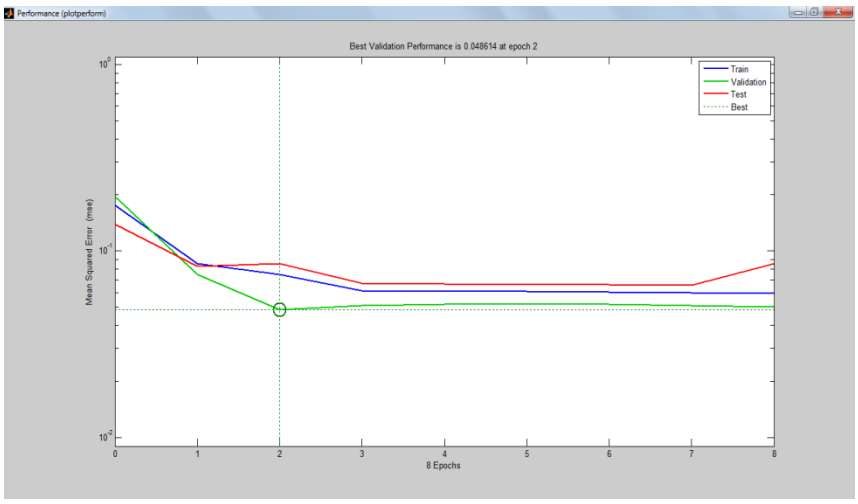


Figure 19: Neural Network Performance Plot for Kp

The mean squared error (MSE) for testing is 0.065, for training is 0.061 and validation is 0.04. The MSE for testing set became 0.1 at the end of the training. This network is acceptable as it shows a small MSE values when training of Neural Network is done. However, since the testing set MSE value became slightly higher at the end of training, it shows that the network can be further improved to obtain a better result.

Based on the training, the output of the network obtained was plotted as shown in Figure 20. Based on the graph, the output for Kp is around 30.07.

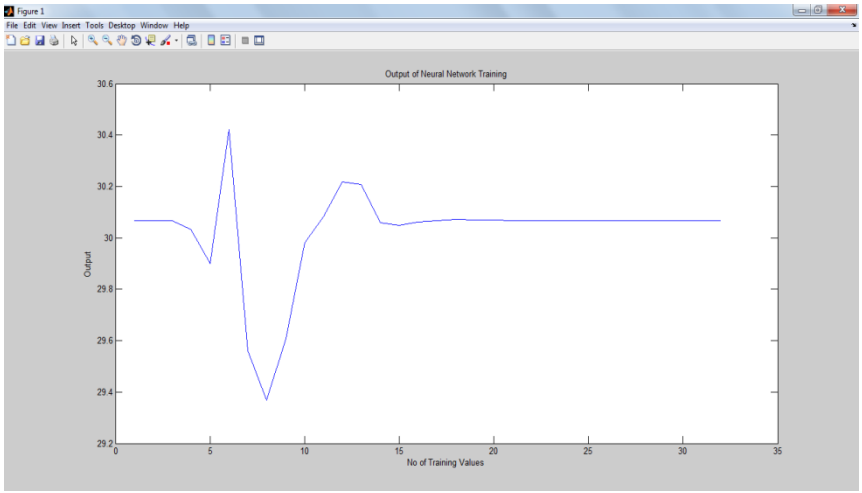


Figure 20: Output of Neural Network training for Kp

The mean squared error (MSE) for testing is 0.9, for training is 0.8 and validation is 1.07. This network has slightly higher error compared to the performance plot for Kp but it is acceptable as it still shows a small MSE values. However, since MSE value is slightly higher, it shows that the network can be further improved to obtain a better result.

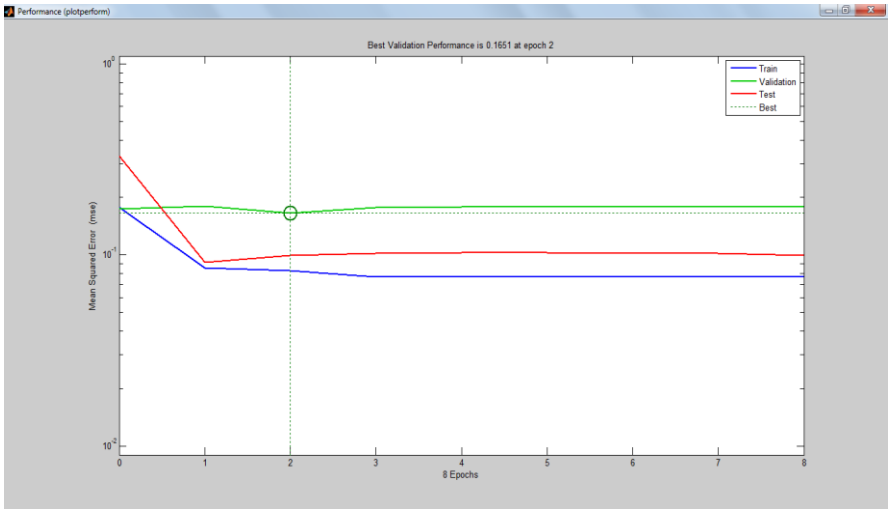


Figure 21: Neural Network Performance Plot for Ti

Based on the training, the output of the network obtained was plotted as shown in Figure 22. Based on the graph, the output for Ti is around 7.39.

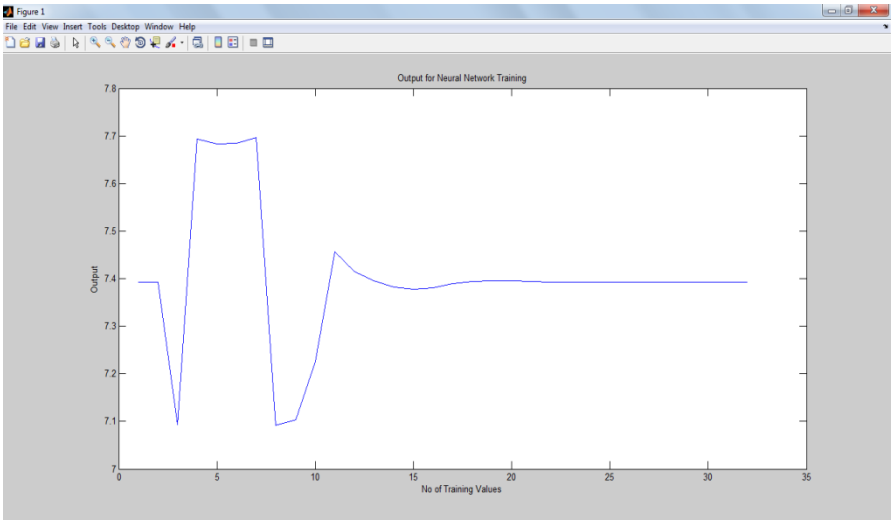


Figure 22: Output of Neural Network training for Ti

The overall performance of the plant can be seen in Figure 23. The decay ratio is lesser than the initial plant performance. The settling time also improved than the initial plant performance. This can be seen from Table 2. Overall, the neural network helped to improve the plant performance.

Table 3 : Comparison of performances between initial tuning and fine-tuning

	Initial	Fine-tune (using NN)
Decay Ratio	0.2/1.5 = 0.13	0.01/0.99 = 0.01
Settling Time	9.86 sec	5.9 sec
Zero offset	Reached	Reached
Kc	30.4	30.07
Ti	12.87	7.39

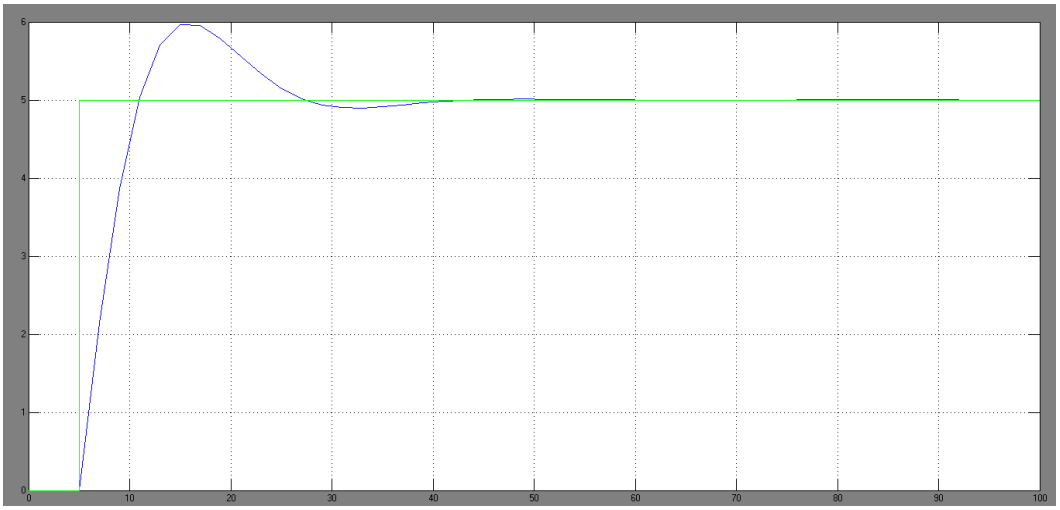


Figure 23: Plant performance after fine-tuning

4.4 Neural Network Predictive Controller

The data used for NN training in Neural Network Predictive Controller is taken from the plant as shown in Figure 11 and the results are as shown in Figure 24, Figure 25 and Figure 26.

The inputs are the random data plant input taken from Figure 11 and they are randomly divided to training, validation and testing data. The plant output is the output of the Simulink model while the NN output is the NN plant model output, a one step ahead prediction of the plant output. The error is the difference between the plant output and the NN plant model output.

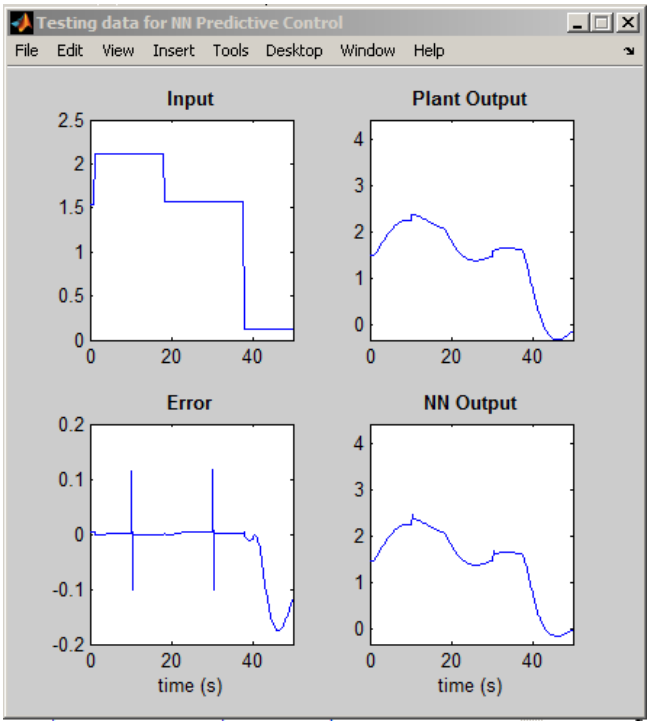


Figure 24: NN Predictive Control Testing Data Results

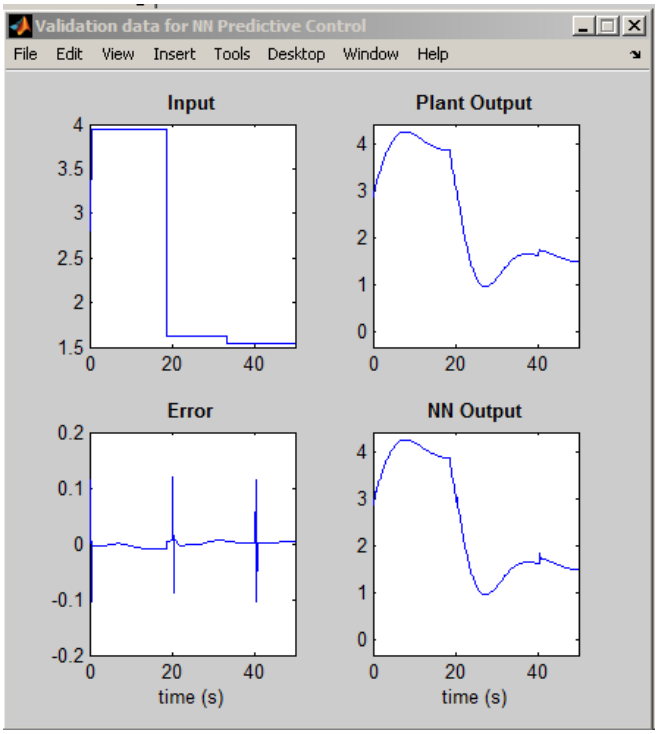


Figure 25: NN Predictive Control Validation Data Results

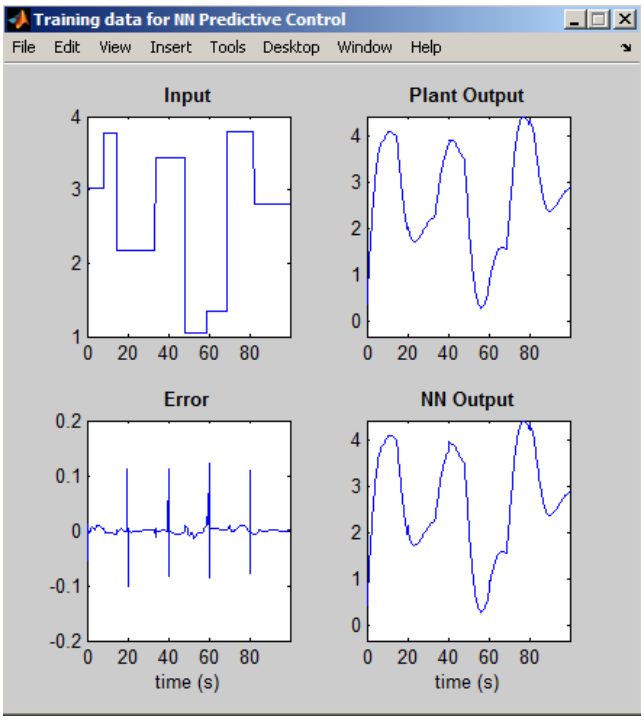


Figure 26: NN Predictive Control Training Data Results

The differences between training, validation and testing data are that training is done to adjust the weights on the neural network, validation is used to minimize over fitting so that the neural network does not need to use all the data to train the network by confirming that there is accuracy in the training data while validation data stays the same. Testing is done to test the final result and verify the predictive power of the network.

The highest error obtained is about 0.1. The goal was for the error to achieved zero. When the random step input is applied, the output would try to get its steady state value which is also the value of the input applied.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

Neural Network is one of the ways to obtain the optimization values of the PID controller tuning parameters. Neural Network model can be built using MATLAB and Simulink. For ease, the Neural Network toolbox can be used to simplify the work of building the model.

The process reaction curve obtained could determine the parameters of the system that is needed to calculate the transfer function of the system. The transfer function calculated is a first-order with dead time model.

After the transfer function is obtained, the plant model is simulated using Simulink. The values obtained from the PID Controller of the plant are used for the building of Neural Network model. It is shown by the results obtained that Neural Network can improve the performance of the plant.

Other than using Neural Network to fine-tune the PID Controller, it is also shown that the Neural Network Predictive Controller can also operate well as a controller for the plant.

5.2 Recommendation

The project can be further improved by testing the proportional, integral and derivatives values obtained to the real plant and analyse the result with the simulation to verify with the values obtained by the Neural Network.

This project does not consider disturbances of the plant. So, the project can be further improved by considering disturbances to more signify the real plant system as in the industries.

Graphical User Interface for this project can also be done for easier simulation.

REFERENCES

[1] Xiang-Sun Zhang, "Neural Networks in Optimization", p. 1-10, 2000

[2] Marlin, Thomas E., "Process Control : Designing Processes and Control Systems For Dynamic Performance, 2nd Edition", p.179 - 181, 1999.

[3] Fung Chun Ting, "Systematic Design of Simply Structured Compensator", Bachelor of Engineering (Hons) Electrical & Electronics Engineering of Universiti Teknologi PETRONAS, June 2005.

[4] Moonyong Lee, Sunwon Park, "Process Control Using a Neural Network Combined with the Conventional PID Controllers", ICASE: The Institute of Control, Automation and Systems Engineers, KOREA Vol. 2, No. 3, September, 2000.

[5] Åström K.J., Hägglund T., "Revisiting the Ziegler–Nichols step response method for PID control", 2004

[6] Mohammad Hassan Shenassa, Kamran Khakpour, "Knowledge Base Expert System for Tuning PID Controllers Using Wireless Technology", Proceedings of the International Conference on Computer and Communication Engineering 2008, Kuala Lumpur, Malaysia, May 13-15 2008.

[7] Araki M., "PID Control", submitted for publications in Control Systems, Robotics, and Automation – Vol. II.

[8] Louis Stowasser, "Neural Networks in PHP – PHP Classes blog – PHP Classes", http://www.phpclasses.org/blog/post/119-Neural-Networks-in-PHP.html, accessed on 6th April 2010 at 4.30 pm.

[9] Howard Demuth, Mark Beale and Martin Hagan, " Neural Network Toolbox™ 6 User's Guide", http://www.mathworks.com/access/helpdesk/help/pdf_doc/nnet/nnet.pdf, accessed on 1st March 2010 at 10.00 am.

[10] "Sensors and Sensing", http://www.cs.brown.edu/~tld/courses/cs148/02/sensors.html, accessed on 25th April 2010 at 4.30 pm.

[11] Ross Berteig, Cheshire Engineering Corporation, "Basic Concepts for Neural Network (Excerpt from the Neuralyst™ User's Guide, Chapter 3)", http://www.cheshireeng.com/Neuralyst/nnbg.htm, accessed on 25th April 2010 at 4.45 pm.

[12] ExperTune Inc., "PID Tuning Tutorial", http://www.expertune.com/tutor.html, accessed on 3rd May 2010 at 2.45 pm.

[13] N.H.H. Mohamad Hanif, Lecture Notes on Plant Process Control System, Electrical & Electronics Engineering Department, Universiti Teknologi PETRONAS, January 2010.

APPENDICES

APPENDIX A

Gantt Chart for FYP 1

Activities	1	2	3	4	5	6	7	Mid-semester break	8	9	10	11	12	13	14
FYP Topic Selection															
Preliminary Research Work															
Submission of Preliminary Report															
Project Work															
- *Transfer function of plant*															
- *Simulation Work*															
- *Result Analysis*															
Submission of Progress Report															
Seminar															
Project Work Continue															
- *Simulation Work*															
- *Result Analysis*															
Submission of Dissertation Draft															
Submission of Project Dissertation															
Oral Presentation															

	Process
	Deadline

APPENDIX B

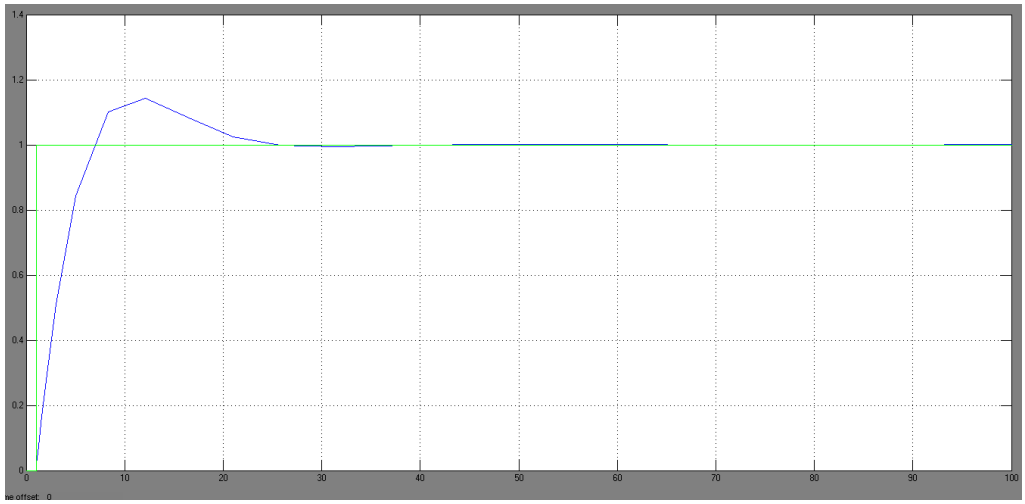
Gantt Chart for FYP 2

Activities	1	2	3	4	5	6	7	Mid-semester break	8	9	10	11	12	13	14
Project Work Continue															
- *Simulation Work*															
- *Result Analysis*															
Submission of Progress Report I															
Project Work Continue															
- *Simulation Work*															
- *Result Analysis*															
Submission of Progress Report II															
Project Work Continue															
- *Simulation Work*															
- *Result Analysis*															
Pre-EDX															
Submission of Dissertation Draft															
Submission of Project Dissertation															
Oral Presentation															

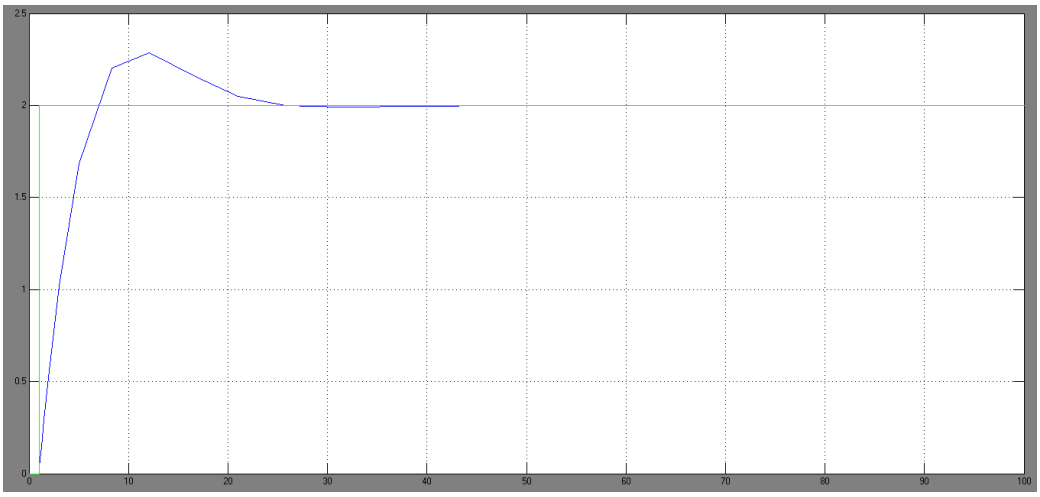
	Process
	Deadline

APPENDIX C

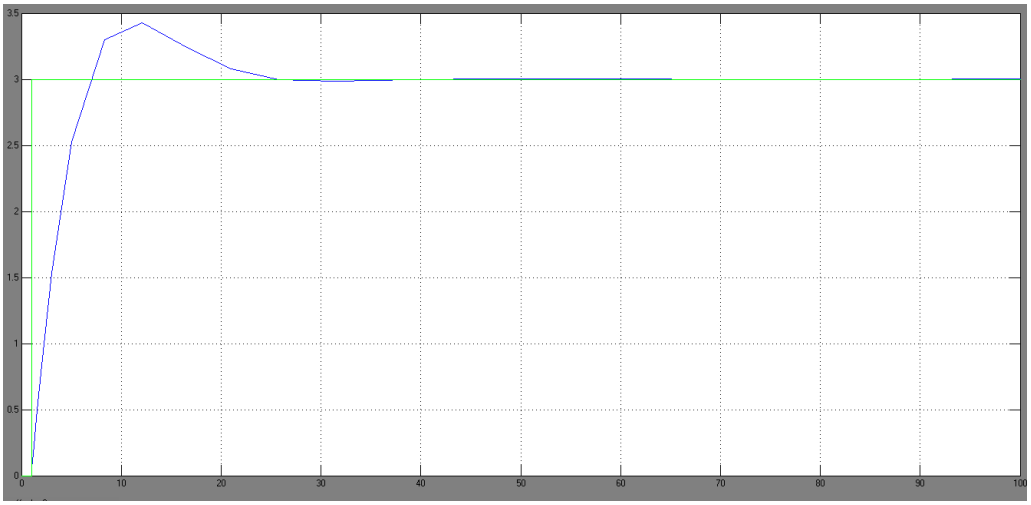
Results for Verification of Transfer Function Based on Different Step Inputs



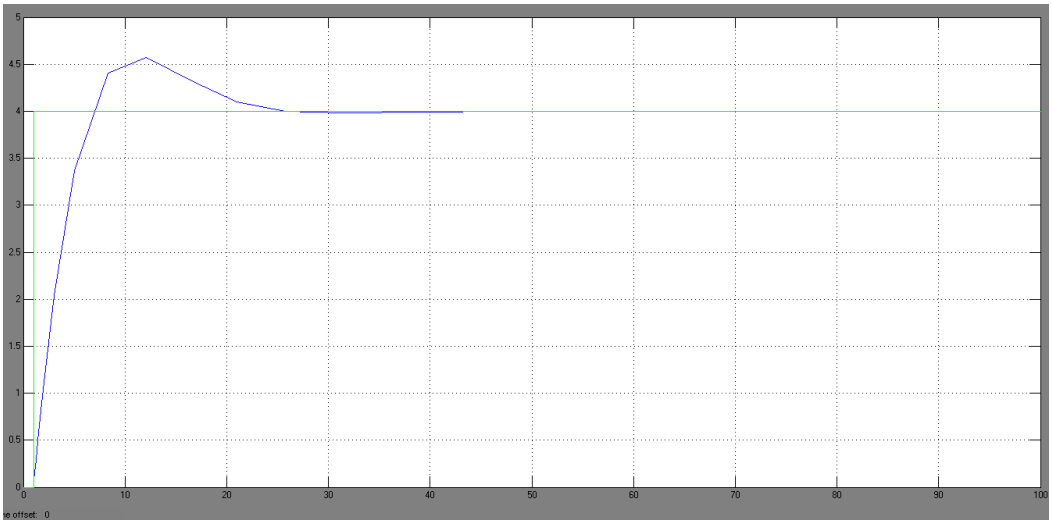
Graph obtained for the transfer function with step input 1



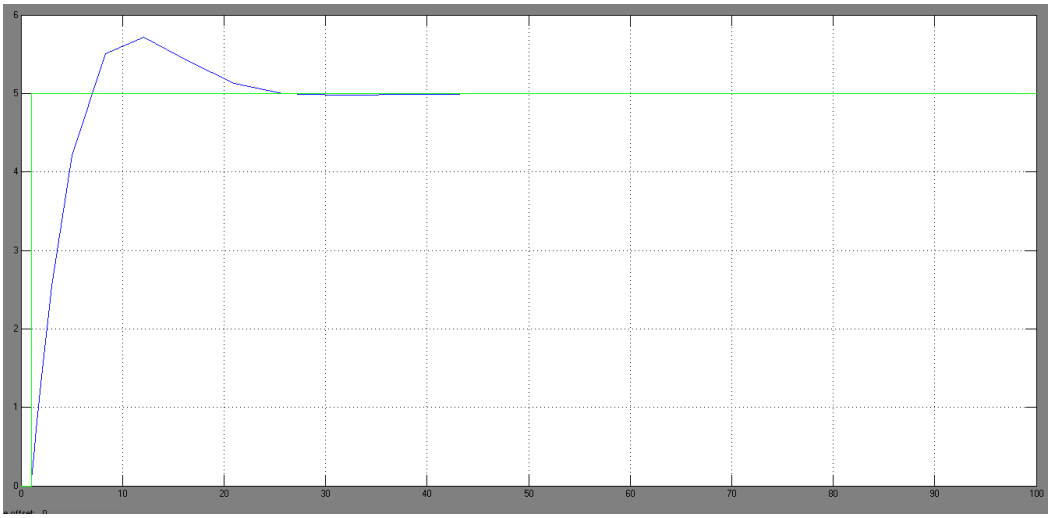
Graph obtained for the transfer function with step input 2



Graph obtained for the transfer function with step input 3



Graph obtained for the transfer function with step input 4



Graph obtained for the transfer function with step input 5

APPENDIX D

Output and error values for Kp neural network training

Output	Error
30.06763	0.032375
30.06763	0.282375
30.06763	0.182375
30.03426	0.005737
29.89986	0.000137
30.42247	-0.41247
29.56065	0.439349
29.36837	0.33163
29.60887	0.191135
29.98095	-0.38095
30.08261	0.117392
30.21776	-0.11776
30.20712	0.192883
30.05811	-0.05811
30.04939	-0.07939
30.06101	-0.22101
30.06704	-0.20704
30.07086	-1.04086
30.07005	0.129952
30.06851	0.231492
30.06739	0.102614
30.06682	0.093179
30.06694	-0.27694
30.06725	-0.37725
30.0678	0.0522
30.06774	0.042256
30.06759	-0.03759
30.06759	-0.15759
30.06763	0.132368
30.06763	0.162367
30.06763	0.032372
30.06763	0.002375

APPENDIX E

Output and error values for Ti neural network training

Output	Error
7.392898	0.607102
7.392898	0.407102
7.093197	0.006803
7.693658	-0.29366
7.683499	-0.0835
7.684527	0.015473
7.696308	0.103692
7.091574	0.208426
7.10421	-0.00421
7.226308	-0.22631
7.457279	-0.11728
7.415486	0.284514
7.394843	0.005157
7.382216	0.117784
7.378056	0.321944
7.380011	0.419989
7.389959	0.610041
7.393696	0.106304
7.39559	0.25441
7.395198	0.194802
7.394136	0.085864
7.392541	0.467459
7.392284	0.457716
7.392316	-0.27232
7.392502	-0.1625
7.392724	-0.15272
7.3929	-0.1429
7.393028	-0.28303
7.392962	-0.37296
7.392888	-0.40289
7.392873	-0.05287
7.392879	-0.26288