NEURAL NETWORK CONTROLLER FOR DC MOTOR USING MATLAB APPLICATION

NORAZLINA BINTI AB. RAHMAN

This Thesis is Part Fulfillment of the Requirement for a Bachelor Degree of Electrical Engineering (Power System)

Faculty of Electrical & Electronic Engineering
University Malaysia Pahang

NOVEMBER 2008

DECLARATION

"I declare that this thesis entitled 'Neural Network Controller for DC Motor using Matlab Application' is the result of my own research except as cited in references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree",

Signature :…….

Name of candidate : Norazlina binti Ab. Rahman

Date : November 8, 2008

DEDICATION

Special dedicated to my family, my friends, my fellow colleague,
and to all faculty members

For all your care, support, and believe in me.

Sincerely;
Norazlina binti Ab. Rahman

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and appreciation to my supervisor Mr. Ahmad Nor Kasruddin Nasir for his guidance, encouragement and advice throughout the preparation of this thesis. His influence has helped me learn the practicalities of this project.

I would like to sincerely thank the Universiti Malaysia Pahang and Fakulti Kejuruteraan Elektrik & Elektronik (FKEE) for providing good facilities in campus and laboratory specifically. A very big thank you dedicated to all the staff of Faculty of Electrical and Electronics. In addition, I would like to acknowledge Mr. Mohd Salmizan for his expertise and assistance with the implementation of the lab equipment.

Finally, I would like to thank to my family for their support and encouragement. Without their support I doubt it would have been possible for me to complete this study.

ABSTRACT

The purpose of this study is to control the speed of direct current (DC) motor with Artificial Neural Network (ANN) controller using MATLAB application. The Artificial Neural Network Controller will be design and must be tune, so the comparison between simulation result and experimental result can be made. The scopes includes the simulation and modeling of direct current (DC) motor, implementation of Artificial Neural Network Controller into actual DC motor and comparison of MATLAB simulation result with the experimental result. This research was about introducing the new ability of in estimating speed and controlling the permanent magnet direct current (PMDC) motor. In this project, ANN Controller will be used to control the speed of DC motor. The ANN Controller will be programmed to control the speed of DC motor at certain speed level. The data from ANN Controller is sent to the DC motor through an interface circuit or a medium called DAQ card. The sensor will be used to detect the speed of motor. Then, the result from sensor is fed back to ANN Controller to find the comparison between the desired output and measured output.

ABSTRAK

Tujuan utama kajian ini adalah untuk mengawal kelajuan *Direct Current (DC) Motor*, di mana *Artificial Neural Network (ANN)* akan menjadi pengawal kelajuan utama dan diaplikasi menggunakan MATLAB. *Artificial Neural Network Controller* akan direka bentuk dan harus disesuaikan nilai komponennya supaya perbezaan di antara keputusan simulasi dapat dibandingkan dengan keputusan eksperimen. Skop tugasan kajian ini termasuklah simulasi dan model *direct current (DC) motor*, perlaksanaan *Artificial Neural Network Controller* ke dalam DC motor yang sebenar dan perbandingan keputusan simulasi MATLAB dengan keputusan eksperimen. Kajian ini adalah untuk memperkenalkan keupayaan baru dalam menaksir dan mengawal kelajuan *Permanent Magnet Direct Current (PMDC) motor*. Di dalam projek ini, *ANN Controller* akan digunakan untuk mengawal kelajuan DC motor. *ANN Controller* juga akan diprogramkan untuk mengawal kelajuan motor melalui simulasi MATLAB pada kadar kelajuan yang telah ditetapkan. Data daripada *ANN Controller* akan dihantar kepada DC motor melalui litar penghubung atau medium yang dikenali sebagai Kad DAQ. Alat pengesan (*Encoder*) akan mengesan tahap kelajuan motor. Selepas itu, keputusan daripada alat pengesan akan di suap kembali kepada *ANN Controller* untuk mencari perbandingan di antara keputusan yang kehendaki dengan keputusan sebenar eksperimen.

TABLE OF CONTENT

CHAPTER	TITLE	PAGE
	DECLARATION	ii
	DEDICATION	iii
	ACKNOWLEDGEMENT	iv
	ABSTRACT	v
	ABSTRAK	vi
	TABLE OF CONTENTS	vii
	LIST OF TABLES	xi
	LIST OF FIGURES	xii
	LIST OF SYMBOLS/ABBREVIATIONS	xiii
1	**INTRODUCTION**	**1**
	1.1 General Introduction to Motor Drives	1
	1.1.1 DC Motor Drives	1
	1.2 Permanent Magnet Direct Current (PMDC) Motor	2
	1.2.1 Introduction	2
	1.2.2 Classification of PM Motor	4
	1.2.3 Brushless DC (BLDC) Motor	4
	1.2 Problem Statement	7
	1.3 Problems encountered and solutions	7
	1.4 Objectives	8
	1.5 Scopes	8

2	**LITERATURE REVIEW**	9
	2.1 Artificial Neural Network Controller	9
	Introduction	9
	2.1.2 Classification of ANNs	10
	2.1.3 Neuron Structures	11
	2.1.3.1 Neurons	11
	2.1.3.2 Artificial Neural Networks	12
	2.1.4 Learning Algorithm	13
	2.1.4.1 Back Propagation Algorithm	14
	2.1.5 Application of Neural Network	15
	2.1.6 Neural Network Software	16
	2.2 Matlab 7.5	17
3	**METHODOLOGY**	19
	3.1 Methodology	19
	3.2 PMDC Motor	21
	3.2.1 Modeling of DC Motor	22
	3.2.1.1 Schematic Diagram	22
	3.2.1.2 Equations	23
	3.2.1.3 S-Domain block diagram of PMDC Motors	25
	3.2.1.4 Values of motor modeling component	26
	3.2.2 Cliffton Precision	26
	3.2.2.1 Features	26
	Driver Motor	27
	3.2.3.1 Mosfet-Drive Driver Motor	27

Data Acquisition Card (DAQ Card) 28
3.2.4.1 PCI1710HG 28
Step Response of control system 30
3.3
3.3.1 Step response 31
3.3.2 step response formula analysis 31
3.3.2.1 Maximum Overshoot 31
3.3.2.2 Settling Time 32
3.3.2.3 Rise Time 33
3.3.2.5 Steady-state error 34
Model Reference Controller 35
3.4
3.4.1 Steps to generate Model Reference Controller 35
3.4.1.1 Steps 1 36
3.4.1.2 Steps 2 36
3.4.1.3 Steps 3 37
3.4.1.4 Steps 4 41
3.4.1.5 Steps 5 42
Interfacing of Matlab Simulink with DAQ Card 52
3.5
3.5.1 DAQ Card Interfacing 52

4 RESULT AND DISCUSSION

55
4.1 Simulation Result 55
4.1.1 The result without Neural Network Controller 55
4.1.1.1 Simulink Structure 55
Graph 56
4.1.2 The result with Neural Network Controller 57

4.1.2.1 Simulink Structure 57
4.1.2.3 Graph 55

4.2 Discussion 60
Comparison between without controller and with controller 60
Result without Neural Network controller 60
Result with Neural Network controller 61
Disturbance factor 62

5 **COSTING AND COMMERCIALIZATION, CONCLUSION AND RECOMMENDATION** 64

5.1 Costing and commercialization 64
5.1.1 Costing 64
5.1.2 Commercialization 64
5.2 Conclusion 65
5.3 Recommendation 65

REFERENCES 66

APPENDIX 67

LIST OF TABLES

TABLE	TITLE	PAGE
1	Values of motor modeling component	26
2	Cliffton Precision Technical Data	26
3	Total cost of project	64

LIST OF FIGURES

FIGURE	TITLE	PAGE
1	Simple neural network	11
2	Example of a two-layer neural network	13
3	Developed system with PC interfacing through DAQ card	19
4	Simulink Block Diagram of DC motor with Artificial Neural Network (ANN) controller	20
5	Basic structure of DC motor	21
6	Schematic Diagram	22
7	Block diagram of permanent magnet DC motors	25
8	Driver IR2109 circuit diagram	28
9	PCI1710HG	29
10	Typical Response of a control system	30
11	Figure 1	36
12	Figure 2	37
13	Figure 3	38
14	Figure 4	39
15	Figure 5	40
16	Figure 6	41
17	Figure 7	42
18	Figure 8	43
19	Figure 9	44
20	Figure 10	45
21	Figure 11	46
22	Figure 12	47
23	Figure 13	48
24	Figure 14	49

25	Figure 15	50
26	Figure 16	51
27	Complete system of ANN Controller	52
28	Figure 17	53
29	Figure 18	54
30	Figure 19	54
30	Simulink Structure	55
31	DC Motor Modeling Simulink Block diagram	56
32	Graph of result without Neural Network Controller	56
33	Simulink Structure of ANN Controller	57
34	Figure 20	58
35	From DC motor modeling box	58
36	Graph with ANN Controller	59
37	Figure 21	60
38	Figure 22	61
39	Figure 23	62
40	Figure 24	63

LIST OF SYMBOLS/ABBREVIATIONS

ANN	-	Artificial Neural Network
BLDC	-	Brushless Direct Current
DAQ	-	Data Acquisition Card
DC	-	Direct Current

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Data Sheets	63

CHAPTER 1

INTRODUCTION

1.1 General Introduction to DC Motor Drives

1.1.1 DC Motor Drives

Conventional direct current electric machines and alternating current induction and synchronous electric machines have traditionally been the three cornerstones serving daily electric motors needs from small household appliances to large industrial plants.

Recent technological advances in computing power and motor drive systems have allowed an even further increase in application demands on electric motors. Through the years, even AC power system clearly winning out over DC system, DC motors still continued to be significant fraction in machinery purchased each year.

There were several reasons for the continued popularity of DC motors. One was the DC power systems are still common in cars and trucks. Another application for DC motors was a situation in which wide variations in speed in needed. Most DC machines are like AC machines in that they have AC voltages and currents within them, DC machines have a DC output only because a mechanism exists that converts the internal AC voltages to DC voltages at their terminals.

The greatest advantage of DC motors may be speed control. Since speed is directly proportional to armature voltage and inversely proportional to the magnetic flux produced by the poles, adjusting the armature voltage and/or the field current will change the rotor speed. Today, adjustable frequency drives can provide precise speed control for AC motors, but they do so at the expense of power quality, as the solid-state switching devices in the drives produce a rich harmonic spectrum. The DC motor has no adverse effects on power quality.

1.2 Permanent Magnet Direct Current (PMDC) motor

1.2.1 Introduction

In a Permanent Magnet motor a coil of wire (called the armature) is arranged in the magnetic field of a permanent magnet in such a way that it rotates when a current is passed through it. Now, when a coil of wire is moving in a magnetic field a voltage is induced in the coil - so the current (which is caused by applying a voltage to the coil) causes the armature to rotate and so generate a voltage. It is the nature of cause and effect in physics that the effect tends to cancel the cause, so the induced voltage tends to cancel out the applied voltage.

Voltage is electrical pressure. Current is electrical flow. Pressure tends to cause movement, or flow so an electrical pressure is a force which moves electricity or an 'electromotive force' (EMF). The induced voltage caused by the armature's movement is a 'back EMF' where 'back' because it tends to cancel out the applied voltage so that the actual voltage or pressure across the armature is the difference between the applied voltage and the back EMF.

The value of the back EMF is determined by the speed of rotation and the strength of the magnet such that if the magnet is strong the back EMF increases and if the speed increases, so too does the back EMF.

1.2.2 Classification of PM Motor

In Permanent Magnet motors, it falls into two categories: PM DC motors and PM AC motors. PM DC motors are separately excited DC motors with permanent magnets as the excitation source. In industry, they are widely used as control motors, often in precision applications such as computer disk drives. Permanent Magnet motor classification can be based on control strategy, which produces more classification of PM motors, the brushless DC (BLDC) motor and conventional permanent magnet synchronous motor.

1.2.3 Brushless DC (BLDC) Motor

A brushless DC motor (BLDC) is a synchronous electric motor which is power-driven by direct-current electricity (DC) and which has an electronically controlled commutation system, instead of a mechanical commutation system based on brushes. In such motors, current and torque, voltage and rpm are linearly related. In BLDC motor, there are two sub-types used which are the Stepper Motor type that may have more poles on the stator and the Reluctance Motor.

In a conventional (brushed) DC motor, the brushes make mechanical contact with a set of electrical contacts on the rotor or also called the commutator, forming an electrical circuit between the DC electrical source and the armature coil-windings. As the armature rotates on axis, the stationary brushes come into contact with different sections of the rotating commutator. The commutator and brush system form a set of electrical switches, each firing in sequence, such that electrical-power always flows through the armature coil closest to the permanent magnet that is used as a stationary stator.

In a BLDC motor, the electromagnets do not move; but, the permanent magnets rotate and the armature remains static. This gets around the problem of how to transfer current to a moving armature. The method which is the brush-system/commutator assembly is replaced by an electronic controller is used. The controller performs the same power distribution found in a brushed DC motor, but using a solid-state circuit rather than a commutator/brush system.

In this motor, the mechanical "rotating switch" or commutator/brush gear assembly is replaced by an external electronic switch synchronized to the rotor's position. Brushless motors are typically 85-90% efficient, whereas DC motors with brush gear are typically 75-80% efficient. BLDC motors also have several advantages over brushed DC motors, including higher efficiency and reliability, reduced noise, longer lifetime caused by no brush erosion in it; elimination of ionizing sparks from the commutator, and overall reduction of electromagnetic interference (EMI). With no windings on the rotor, they are not subjected to centrifugal forces, and because the electromagnets are located around the perimeter, the electromagnets can be cooled by conduction to the motor casing, requiring no airflow inside the motor for cooling. This means that the motor's internals can be entirely enclosed and protected from dirt or other foreign matter. [2]

The maximum power that can be applied to a BLDC motor is exceptionally high, limited almost exclusively by heat, which can damage the magnets. BLDC's main disadvantage is higher cost, which arises from two issues. First, BLDC motors require complex electronic speed controllers to run. Brushed DC motors can be regulated by a comparatively trivial variable resistor (potentiometer or rheostat), which is inefficient but also satisfactory for cost-sensitive applications. Second, many practical uses have not been well developed in the commercial sector.

BLDC motors are considered to be more efficient than brushed DC motors. This means that for the same input power, a BLDC motor will convert more electrical power into mechanical power than a brushed motor, mostly due to the absence of friction of brushes. The enhanced efficiency is greatest in the no-load and low-load region of the

motor's performance curve. Under high mechanical loads, BLDC motors and high-quality brushed motors are comparable in efficiency. Brushless DC motors are commonly used where precise speed control is necessary, as in computer disk drives or in video cassette recorders, the spindles within CD, and etc. [2]

1.3 Problem Statement

When commerce with DC motor, the problem come across with it are efficiency and losses. In order for DC motor to function efficiently on a job, it must have some special controller with it. Thus, the Artificial Neural Network Controller will be used.

There are too many types of controller nowadays, but ANN Controller is chosen to interface with the DC motor because in ANN, Non-adaptive control systems have fixed parameters that are used to control a system. These types of controllers have proven to be very successful in controlling linear, or almost linear, systems.

1.4 Problems encountered and solutions

Problem encountered:-

i) Control of DC motor speed;
ii) Interface of DC motor with software (MATLAB/SIMULINK);
iii) To acquire data from the DC motor

Solutions:-

i) Use of ANN controller to the system;
ii) Implementation of DAQ card to the control board;
iii) Use of encoder from the DC motor to the control board;

1.5 Objectives

The objective of the Artificial Neural Network Controller Design for DC motor using MATLAB an application is it must control the speed of DC motor with Artificial Neural Network controller using MATLAB application which the design of the ANN controller is provided and can be tune. Each of the experimental result must be compared to the result of simulation, as a way to attain the closely approximation value that can be achieved in this system.

1.6 Scopes

The scopes that will be figure out in this research are:

i) Simulation and Modeling of DC motor;
ii) Implementation of ANN controller to actual DC motor;
iii) Comparison of MATLAB simulation result with the experimental result.

CHAPTER 2

LITERATURE REVIEW

2.1 Artificial Neural Network Controller

2.1.1 Introduction

Nowadays, the field of electrical power system control in general and motor control in particular has been researching broadly. The new technologies are applied to these in order to design the complicated technology system. One of these new technologies is Artificial Neural Network (ANNs) which based on the operating principle of human being nerve neural. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well.

There are a number of articles that use ANNs applications to identify the mathematical DC motor model. Then, this model is applied to control the motor speed. The inverting forward ANN with two input parameters for adaptive control of DC motor ANNs are applied broadly because all the ANN signal are transmitted in one direction, the same as in automatically control system, the ability of ANNs to learn the sample, the

ability to creating the parallel signals in analog as well as in discrete system and also because adaptive ability in ANNs. They can be used to model complex relationships between inputs and outputs or to find patterns in data. Neural network technologies have been reported superior in numerous application areas in medical, motor drives, business, stock and share etc.

2.1.2 Classification of ANNs

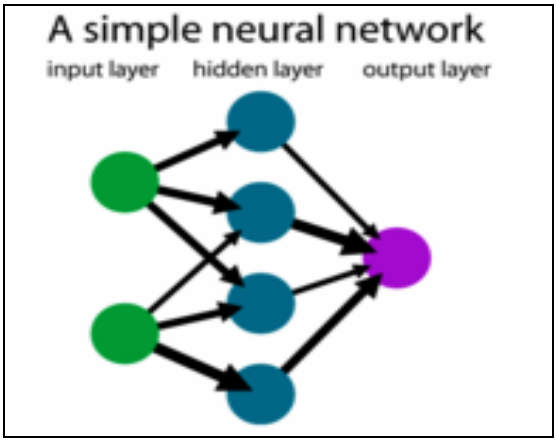
The modern usage of the term often refers to artificial neural networks, which are composed of artificial neurons or nodes. Thus the term 'Neural Network' has two distinct usages which are biological neural networks are made up of real biological neurons that are connected or functionally-related in the peripheral nervous system or the central nervous system. In the field of neuroscience, they are often identified as groups of neurons that perform a specific physiological function in laboratory analysis and Artificial neural networks are made up of interconnecting artificial neurons (programming constructs that mimic the properties of biological neurons). Artificial neural networks may either be used to gain an understanding of biological neural networks, or for solving artificial intelligence problems without necessarily creating a model of a real biological system.

An artificial neural network (ANN), also called a simulated neural network (SNN) or commonly just neural network (NN) is an interconnected group of artificial neurons that uses a mathematical or computational model for information processing based on a connections approach to computation. In most cases an ANN is an adaptive system that changes its structure based on external or internal information that flows through the network. In more practical terms neural networks are non-linear statistical data modeling or decision making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data. [2]

An artificial neural network involves a network of simple processing elements (artificial neurons) which can exhibit complex global behavior, determined by the connections between the processing elements and element parameters. One classical type of artificial neural network is the Hopfield net.

2.1.3 Neuron Structures

2.1.3.1 Neurons



simple neural network

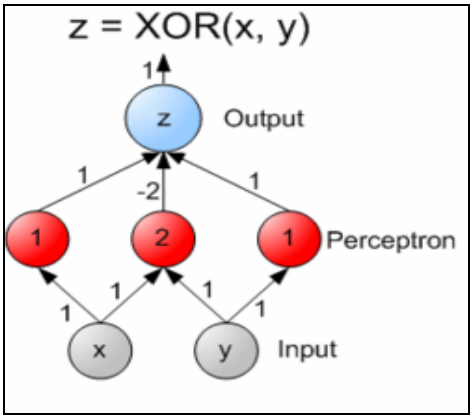
In a neural network model simple nodes, which can be called "neurons", "neurodes", "Processing Elements" (PE) or "units", are connected together to form a network of nodes and then being called "neural network". In some of these systems, neural networks, or parts of neural networks (such as artificial neurons) are used as components in larger systems that combine both adaptive and non-adaptive elements. [3]

2.1.3.2 Artificial Neural Networks

An ANN consists of an input layer, an output layer and one or more hidden layers. There is no transfer function is applied at the input layer and direct inputs are transferred as outputs from this layer. The input layer acts like the biological sensory system, providing information about the surrounding environment. Activations are calculated from the next layer onwards (the hidden layers) and fed into higher layers till it reaches the final output layer. This kind of an ANN structure is called a *feed forward.* There also have feedback networks, commonly called a recurrent network.

ANNs are capable of learning very high-order statistical correlations that are present in a training environment. Learning algorithms provide a powerful mechanism for generalizing behavior to new environments. However, the purpose of the networks then merely reduces to the simulation of a set of goals. Networks in which endogenous goals plays an important role in determining behavior are difficult to train using usual training algorithms. In such cases evolutionary methodologies are the appropriate mechanism for developing goals and purposeful behavior, rather than goal-directed explicit training.

A feed-forward neural network is an artificial neural network where connections between the units do not form a directed cycle. This is different from recurrent neural networks. The feed-forward neural network was the first and arguably simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes if any and to the output nodes. There are no cycles or loops in the network.



Example of a two-layer neural network

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. Multi-layer networks use a variety of learning techniques, the most popular being back-propagation.

2.1.4 Learning algorithms

There are many algorithms for training neural networks; most of them can be viewed as a straightforward application of optimization theory and statistical estimation. Evolutionary computation methods simulated annealing, expectation maximization and non-parametric methods are among other commonly used methods for training neural networks. One of the learning algorithms used in non linear control system is back propagation algorithm. [3]

2.1.4.1 Back Propagation Algorithm

Back propagation, or propagation of error, is a common method of teaching artificial neural networks how to perform a given task. The techniques uses in this algorithm are present a training sample to the neural network, Compare the network's output to the desired output from that sample, calculate the error in each output neuron, for each neuron, calculate what the output should have been, and a scaling factor, how much lower or higher the output must be adjusted to match the desired output. This is known as the local error, Adjust the weights of each neuron to lower the local error, Assign "blame" for the local error to neurons at the previous level, giving greater responsibility to neurons connected by stronger weights, Repeat the steps above on the neurons at the previous level, using each one's "blame" as its error.

Back propagation usually allows quick convergence on satisfactory local minima for error in the kind of networks to which it is suited. Here, the output values are compared with the correct answer to compute the value of some predefined error-function. By various techniques, the error is then fed back through the network. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function by some small amount. After repeating this process for a sufficiently large number of training cycles, the network will usually converge to some state where the error of the calculations is small.

In this case, one would say that the network has learned a certain target function. To adjust weights properly, one applies a general method for non-linear optimization that is called gradient descent. For this, the derivative of the error function with respect to the network weights is calculated, and the weights are then changed such that the error decreases (thus going downhill on the surface of the error function). For this reason, back-propagation can only be applied on networks with differentiable activation functions.

2.1.5 Application of Neural Network

The tasks to which artificial neural networks are applied tend to fall within the following broad categories:

- Function approximation, or regression analysis, including time series prediction and modeling.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.
- Data processing, including filtering, clustering, blind signal separation and compression.
-

System identification and control (vehicle control, process control), game-playing and decision making (backgammon, chess, racing), pattern recognition (radar systems, face identification, object recognition, etc.), sequence recognition (gesture, speech, handwritten text recognition), medical diagnosis, financial applications, data mining (or knowledge discovery in databases, "KDD"), visualization and e-mail spam filtering are the application areas that neural network can works on. [3]

2.1.6 Neural Network Software

Neural network software is used to simulate research, develop and apply artificial neural networks, biological neural networks and in some cases a wider array of adaptive systems. Historically, the most common type of neural network software was intended for researching neural network structures and algorithms. The primary purpose of this type of software is to, through simulation; gain a better understanding of the behavior and properties of neural networks. Today in the study of artificial neural networks, simulators have largely been replaced by more general component based development environments as research platforms.

Commonly used artificial neural network simulators include the Stuttgart Neural Network Simulator (SNNS), Emergent and JavaNNS. Commonly used biological network simulators include XNBC and the BNN Toolbox for MATLAB.

Development environments for neural networks differ from the software described above, they can be used to develop custom types of neural networks and they support deployment of the neural network outside the environment. In some cases they have advanced preprocessing, analysis and visualization capabilities.

Popular development environments are:

- the Wolfram Research Neural Network package which is based on the Mathematica computing environment;
- the AMORE and neural packages of the R programming language;
- the MathWorks Neural Network Toolbox which is based on the MATLAB computing environment;
- The GBLearn2 library of the Lush programming language.

2.2 Matlab 7.5

MATLAB® is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. Typical uses include Math and computation Algorithm development Data acquisition Modeling, simulation, and prototyping Data analysis, exploration, and visualization Scientific and engineering graphics Application development, including graphical user interface building

MATLAB is an interactive system whose basic data element is an array that does not require dimensioning. This allows solving many technical computing problems, especially those with matrix and vector formulations, in a fraction of the time it would take to write a program in a scalar no interactive language such as C or FORTRAN.

The name MATLAB stands for matrix laboratory. MATLAB was originally written to provide easy access to matrix software developed by the LINPACK and EISPACK projects. Today, MATLAB engines incorporate the LAPACK and BLAS libraries, embedding the state of the art in software for matrix computation.

MATLAB has evolved over a period of years with input from many users. In university environments, it is the standard instructional tool for introductory and advanced courses in mathematics, engineering, and science. In industry, MATLAB is the tool of choice for high-productivity research, development, and analysis.

MATLAB features a family of add-on application-specific solutions called toolboxes. Very important to most users of MATLAB, toolboxes allow you to learn and apply specialized technology. Toolboxes are comprehensive collections of MATLAB

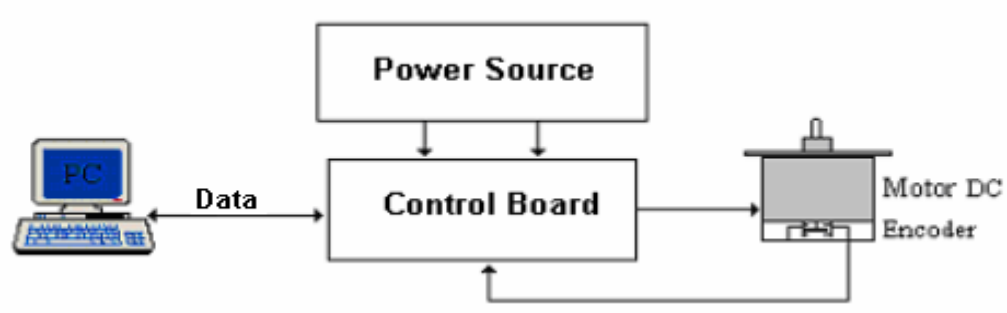
functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

CHAPTER 3

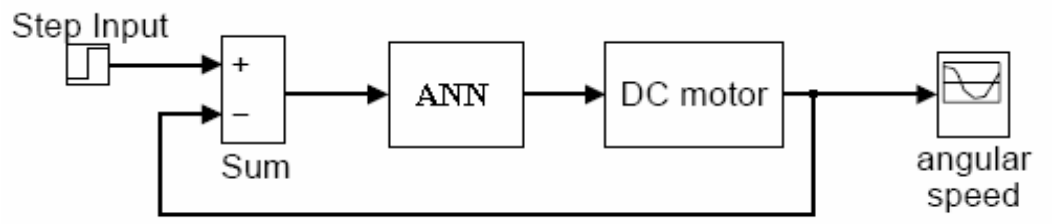
METHODOLOGY

3.1.1 Methodology

This paper describes a performance control method in brushless DC motor drive system applied on modern control theory. Firstly, a model of a permanent magnet synchronous motor is derived as one-input and one-output controlled object. The control system has been driven by control algorithm with full state feedback. The validities of the proposed method are confirmed by experimental results.



Developed system with PC interfacing through DAQ card



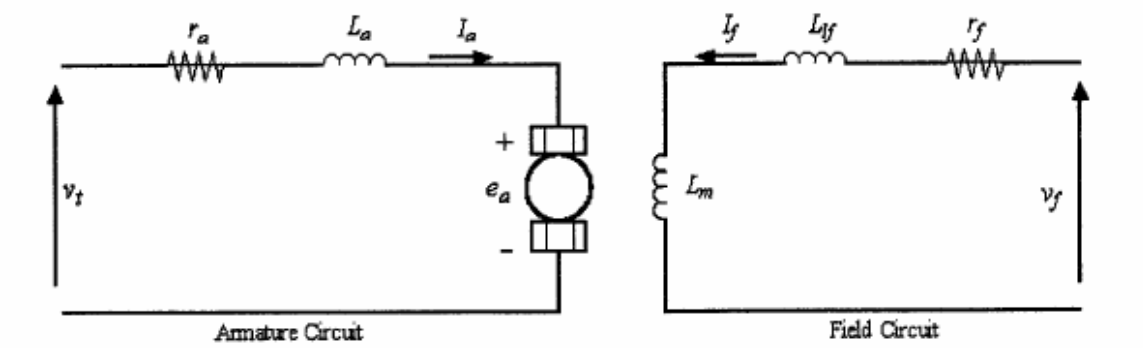
Simulink Block Diagram of DC motor with Artificial Neural Network (ANN) controller

Artificial Neural Network (ANN) or Neural Network is an interconnected of a very simple calculating unit called 'Neuron'. The whole network can be represented using a directed graph. In this project, ANN is used as a controller to control the speed of a DC motor. The MATLAB software is used to execute Neural Network controller block design, focusing on a Simulink in MATLAB software.

The main purpose of this project is to control the speed of DC motor using ANN controller. The ANN Controller will be programmed to control the speed of DC motor at certain speed level. The data from ANN Controller is sent to the DC motor through an interface circuit or medium called DAQ card. The Encoder will be used to detect the speed of motor. The result from sensor then is fed back to ANN Controller to find the comparison between the desired output and measured output.

3.2 PMDC Motor

Several types of controllers, including conventional fixed types such as proportional-integral (PI), proportional-integral derivative (PID) and pseudo-derivative-feedback (PDF); adaptive types such as model reference adaptive controllers (MRAC), sliding mode controllers (SMC) and variable structure controllers (VSC); and intelligent types such as fuzzy logic controllers (FLC), Artificial Neural Network (ANN) controllers and neuro-fuzzy controllers have been used for moderate to high performance drive systems.



Basic structure of DC motor

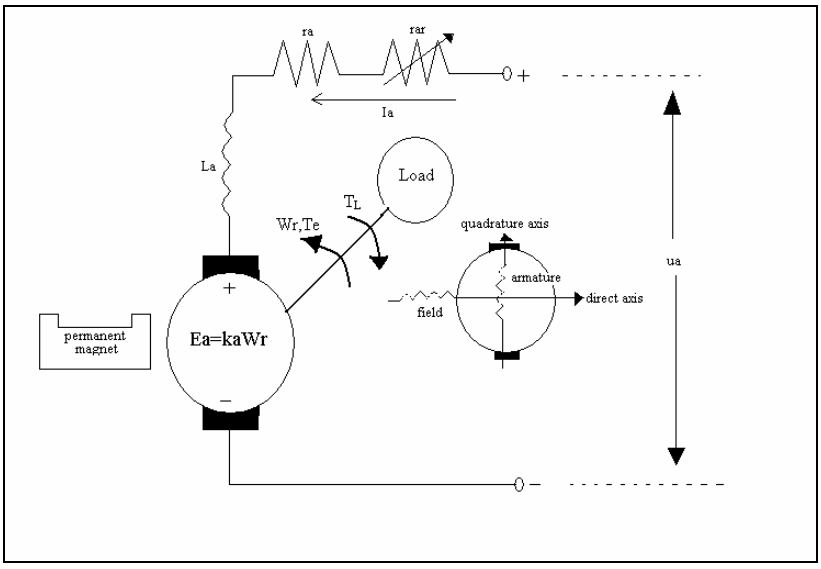
The basic structure of a DC motor consists of a rotating armature winding supplied through a commutator and brushes, and a stationary field structure, which uses either a DC excited winding or permanent magnets. This structure is shown in figure b respectively. As the coils pass the brush position, the action of the commutator is to reverse the direction of the armature winding currents such that the armature current

distribution is fixed in space no matter what rotor speed exists. The flux and mmf are maintained in a mutually perpendicular orientation independent of rotor speed.

This will results in the field flux being unaffected by the armature current and maximum possible torque. The control variables I_a, armature current (torque component) and I_f, field current (flux component) can be considered orthogonal or decoupled vectors. During normal operation, the field current is set to maintain the rated flux, and changing the armature current varies torque. The torque sensitivity remains maximums in both transient and steady-state operation since the stator current components responsible for rotor flux and torque production are decoupled.

3.2.1 Modeling of DC Motor

3.2.1.1 Schematic Diagram



Schematic Diagram

Where,

$$Ea = KaWr \text{ ---------- (1)}$$

3.2.1.2 Equations

The equation of DC motor can be produce using Kirchoff's voltage law and Newton's second law of motion, the differential equations for permanent magnet DC motor can be easily derived. In particular, by using the Kirchhoff law, the equation below is obtains

$$I = \frac{ua - Ea}{ra} \text{ ---------- (2)}$$

By considering the motor magnetic system, the flux linkages in the armature and field windings are produced by the armature current. According to Kirchhoff's voltage law, the applied voltage is expressed as, (rar =0)

$$ua - LafIaWr = (ra + rf)Ia + (La + Lf)\frac{dIa}{dt} \text{ ---------- (3)}$$

Using differential equation (3) and equating $\frac{dIa}{dt}$ to zero (for steady-state analysis), the following expression for armature current is as below

$$ia = \frac{ua}{LafWr + ra + rf} \text{ ---------- (4)}$$

Substituting this equation into expression for electromagnetic torque $Te = LafIa^2$, one obtains

$$Te = LafIa^2 = [Laf / (LafWr + ra + rf)^2] ua^2 \text{----------(5)}$$

Assuming that the susceptibility is constant, one supposes that the flux, established by the permanent magnet poles, is constant. Then, denoting the back emf and torque constant as Ka, the following equations describing the transient behavior of the armature winding and torsional-mechanical dynamics is obtain;

$$\frac{dia}{dt} = -\frac{ra}{La}Ia - \frac{ka}{La}wr + \frac{1}{La}Ua \quad \text{---------- (6)}$$

$$\frac{dwr}{dt} = \frac{ka}{J}Ia - \frac{Bm}{J}wr - \frac{1}{J}Tl \quad \text{---------- (7)}$$

The model in matrix form is obtained;

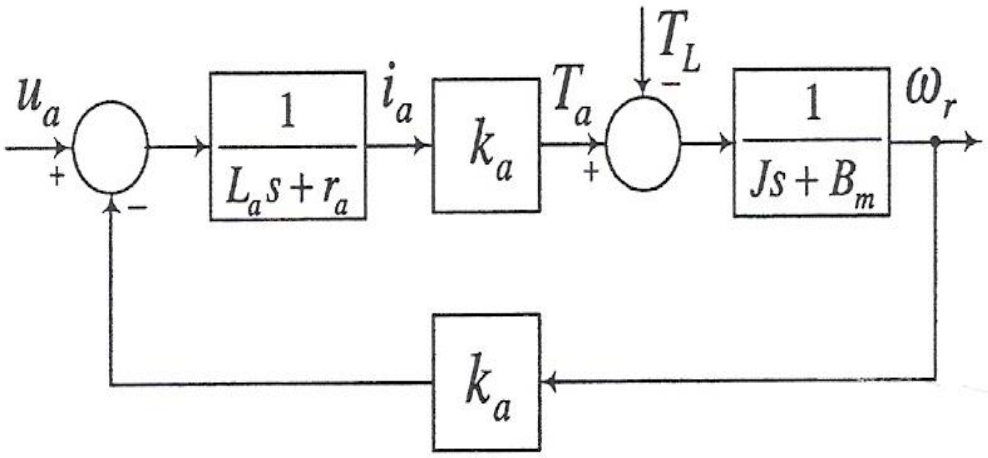
$$\begin{bmatrix} \frac{dia}{dt} \\ \frac{dwr}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{ra}{La} & -\frac{ka}{La} \\ \frac{ka}{J} & -\frac{Bm}{J} \end{bmatrix} + \begin{bmatrix} \frac{1}{La} \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ \frac{1}{J} \end{bmatrix} Tl$$

The Transfer Function/Laplace equation is obtained;

$$\left(s+\frac{B_m}{J}\right)\omega_r(s)=-\frac{1}{J}k_a i_a(s)+\frac{1}{J}T_L(s)$$

$$\left(s+\frac{r_a}{L_a}\right)i_a(s)=-\frac{k_a}{L_a}\omega_r(s)+\frac{1}{L_a}u_a(s)$$

3.2.1.3 S-Domain block diagram of PMDC Motor



Block diagram of permanent magnet DC motors

An S-domain block diagram of permanent magnet DC motors is illustrated in figure above. The angular velocity can be reversed if the polarity of the applied voltage is changed (the direction of the field flux cannot be changed). The steady-state torque-speed characteristic curves obey the following equation

$$Wr = \frac{ua - raIa}{ka} = \frac{ua}{ka} - [Ra/ka2]Te$$ ---------- (8)

3.2.1.4 Values of motor modeling component

Values of motor parameter	
Components	Values
Ra	2.7Ω
La	0.004H
Ka	0.105V-s/rad
Ka	0.105 N-m/A
J	0.0001kgm2
Bm	0.0000093 Nms/rad

Values of motor parameter

3.2.2 Cliffon Precision

3.2.1.1 Features

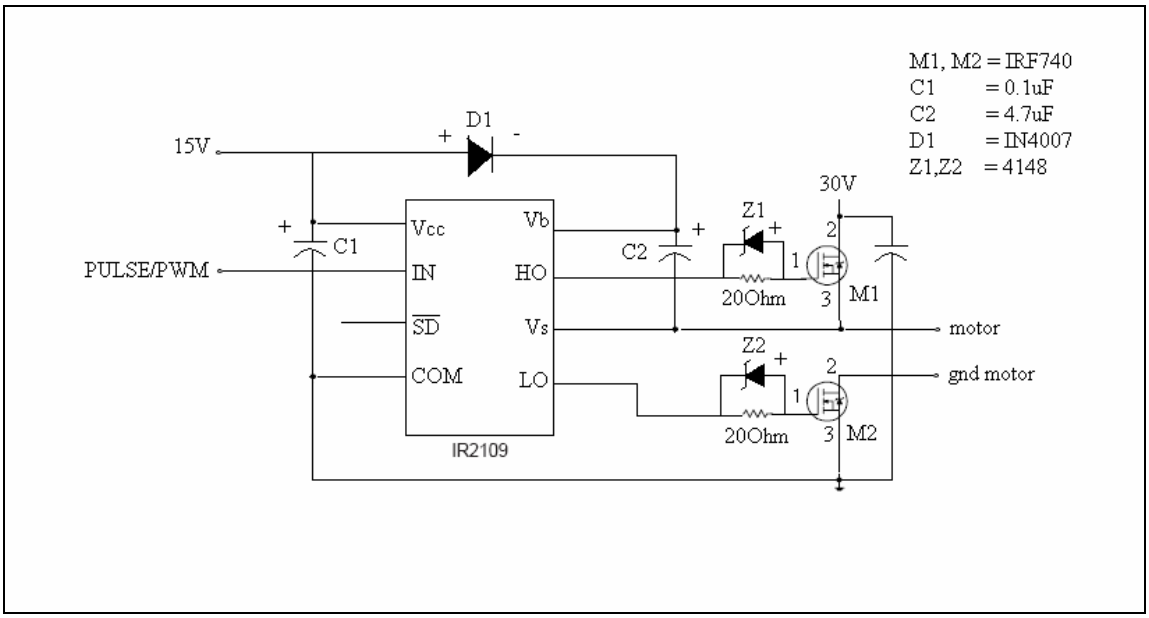
Name	Cliffton Precision
Type	Brushless Permanent Magnet DC Motor
U/T	JDH-2250-HF-2C-E
P/N	000-053479-002
D/C	9444
Technical Specification	Voltage 11-30V, 11A, 2500RPM

Cliffton Precision Technical Data

3.2.3 Driver Motor

3.2.3.1 Mosfet-Drive driver motor

The Driver for drive the Brushless Permanent Magnet DC Motor**,** Cliffton Precision type is made using the circuit diagram given from the IC Driver 2109 datasheet. The IR2109 are high voltage, high speed power MOSFET and IGBT drivers with dependent high and low side referenced output channels. Proprietary HVIC and latch immune CMOS technologies enable ruggedized monolithic construction. The logic input is compatible with standard CMOS or LSTTL output, down to 3.3V logic. The output drivers feature a high pulse current buffer stage designed for minimum driver cross-conduction. The floating channel can be used to drive an N-channel power MOSFET or IGBT in the high side configuration which operates up to 600 volts. The following circuit diagram is as below. [7]

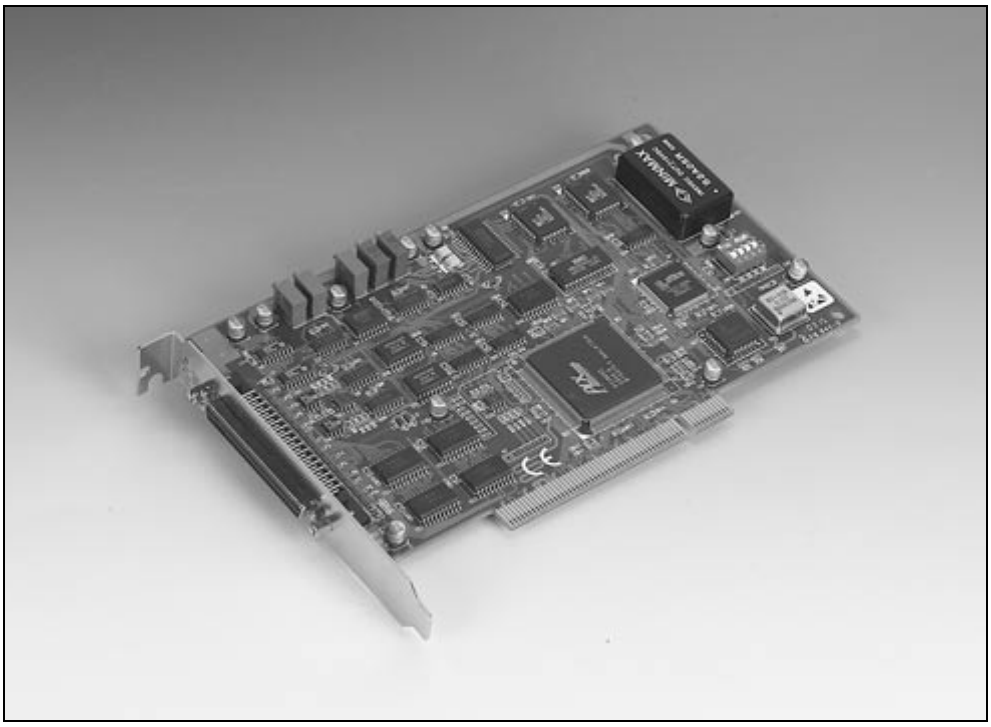


Driver IR2109 circuit diagram

3.2.4 Data Acquisition Card (DAQ Card)

3.2.4.1 PCI1710HG

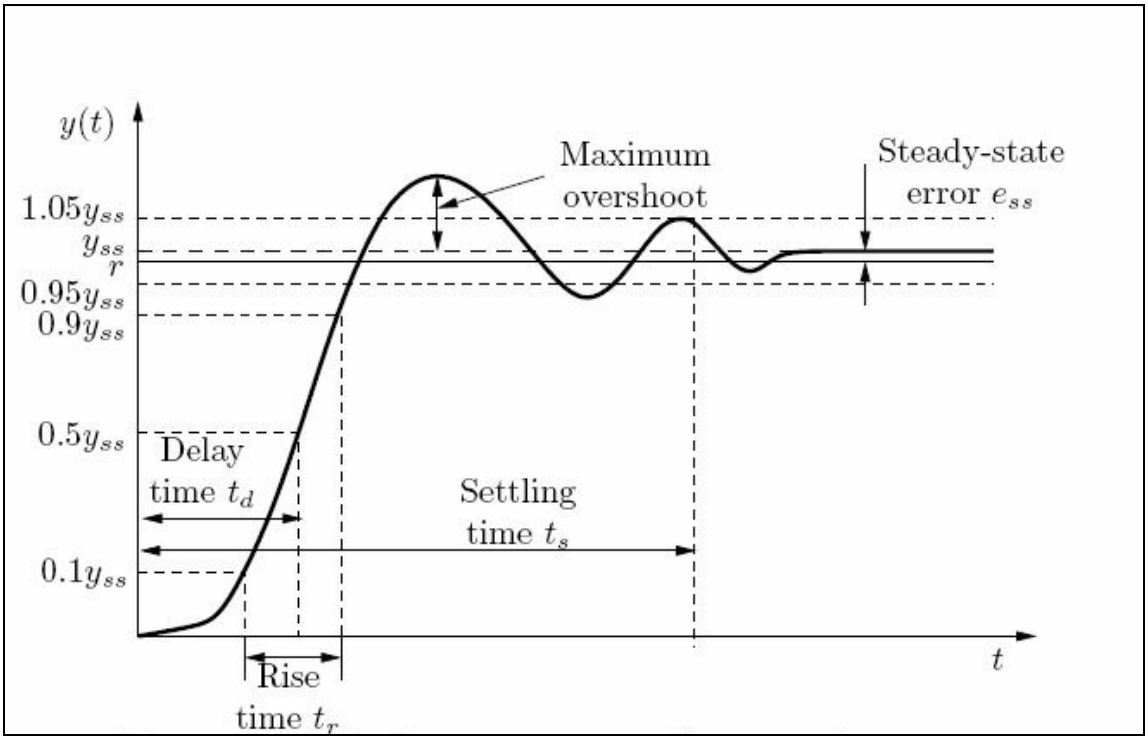
The PCI-1710 Series are 100 kS/s, 12-bit high-gain multifunction cards for the PCI bus. Their advanced circuit design provides higher quality and more functions, including the five most desired measurement and control functions: 12-bit A/D conversion, D/A conversion, digital input, digital output, and counter/timer. The requirement and data for it is all stated at the datasheet.



PCI1710HG

3.3 Step Response of control system

3.3.1 Step Response



Typical Response of a control system

The response above is a reference response used in this system. The way to find the value of each component is by referring to the figure above. The standard maximum overshoot value for response graph is less or equal 2% from the maximum value from y(t). The rise time, Tr value calculated from 0.1yss until 0.9yss, the delay time, Td calculated from 0yss to 0.5yss, the settling time, Ts is calculated from 0seconds until the higher peak of second wave. Mean that, all the components above must be within the standard limit like as shown above.

3.3.2 Step Response formula analysis

3.3.2.1 Maximum Overshoot

The overshoot is the maximum swing above final value, and clearly increases with μ. It is represent a distortion of the signal. In circuit design, the goals of minimizing overshoot and of decreasing circuit rise time can conflict. The magnitude of overshoot depends on time through a phenomenon called *"damping."*. Its often is associated with settling time. Maximum overshoot is the maximum value minus the step value divided by the step value. In the case of the unit step, the *overshoot* is just the maximum value of the step response minus one.

Let y_{max} denote the maximum value of y(t). The maximum overshoot of the step response y(t) is defined as

Maximum overshoot: $= y_{max} - y_{ss}$

The maximum overshoot is often represented as a percentage of the steady-state value,

Percent maximum overshoot, %OS = (maximum overshoot – y_{ss}) × 100%.

The maximum overshoot is often used to measure the relative stability of a system. A system with a large overshoot is usually undesirable.

3.3.2.2 Settling Time

The settling time is the time for departures from final value to sink below some specified level, say 10% of final value. Settling time depends on the system response and time constant.The settling time for a 2nd order, closed loop system responding to a step response is

Ts = − ln(tolerancefraction) / (naturalfreq * dampingratio)

Thus, settling time to within 2%=0.02 is:

Ts = ln(50) / ($\eta\omega_n$)

Or by analysis, say Δ that is:

$$S(t) \leq 1 + \Delta \ ,$$

This condition is satisfied regardless of the value of β A_{OL} provided the time is longer than the settling time, say t_S, given by:

$$\Delta = e^{-\rho t_S} \quad \text{Or} \quad t_S = \frac{\ln\left(\frac{1}{\Delta}\right)}{\rho} = \tau_2 \, \frac{2 \ln\left(\frac{1}{\Delta}\right)}{1 + \frac{\tau_2}{\tau_1}} \approx 2 \, \tau_2 \, \ln\left(\frac{1}{\Delta}\right) \ ,$$

Where the approximation $\tau_1 >> \tau_2$ is applicable because of the overshoot control condition, which makes $\tau_1 = \alpha \beta A_{OL} \tau_2$. Often the settling time condition is referred to by saying the settling period is inversely proportional to the unity gain bandwidth, because $1/(2\pi \tau_2)$ is close to this bandwidth for an amplifier with typical dominant pole compensation. However, this result is more precise than this rule of thumb. As an example of this formula, if $\Delta = 1/e^4 = 1.8$ %, the settling time condition is $t_S = 8 \tau_2$. In general, control of overshoot sets the time constant ratio, and settling time t_S sets τ_2.

3.3.2.3 Rise Time

The rise time tr is defined as the time required for the step response to rise from 10% to 90% of its steady-state value. Rise time is often used to measure the response speed of a system. Its desired value is related to a particular system characteristic and therefore it is often given as a design specification. In general, a small value is more desirable. Delay time and settling time are somehow similar concepts that serve for the same purpose, but they will not be covered in this experiment.

In control theory, it is often defined as the 10% to 90% time from a former setpoint to new setpoint. The quadratic approximation for normalized **rise time** for a 2nd-order system, step response, no zeros is:

$$t_r \cdot \omega_0 = 2.230\zeta^2 - 0.078\zeta + 1.12$$

Where ζ is the damping ratio and ω_0 is the natural frequency of the network.

However, the proper calculation for rise time of a system of this type is:

$$t_r \cdot \omega_n = \frac{1}{\sqrt{1-\zeta^2}} \tan^{-1}\left(\frac{\sqrt{1-\zeta^2}}{\zeta}\right)$$

Where ζ is the damping ratio and ω_n is the natural frequency of the network.

3.3.2.4 Steady –state error

Steady-state error stated that the steady-state value of the step response y(t) is defined as

$$yss := \lim_{t \to \infty} y(t).$$

For a servo control system, we always want the output, y(t), to follow a desired reference signal, r(t). Thus we can define the error as

$$e(t) := r(t) - y(t),$$

And consequently, the steady-state error is given by

$$ess = \lim_{t \to \infty} e(t)$$

3.4 Model Reference Controller

The neural model reference control architecture uses two neural networks: a controller network and a plant model network, as shown in the following figure. The plant model is identified first, and then the controller is trained so that the plant output follows the reference model output.

The figure on the following page shows the details of the neural network plant model and the neural network controller, as they are implemented in the Neural Network Toolbox. Each network has two layers, and the numerous values of number of neurons can be selected to use in the hidden layers. There are three sets of controller inputs: Delayed reference inputs, Delayed controller outputs and Delayed plant outputs.

For each of these inputs, the number of delayed values can select to use. Typically, the number of delays increases with the order of the plant. There are two sets of inputs to the neural network plant model: Delayed controller outputs and Delayed plant outputs.

3.4.1 Steps to generate Model reference Controller

This section demonstrates how the neural network controller is trained. The first step is to copy the Model Reference Control block from the Neural Network Toolbox block set to model window.

A demo model is provided with the Neural Network Toolbox to demonstrate the model reference controller. The objective is to control the speed of the permanent magnet DC motor brushless types, as shown in the following figure.

This reference controller uses a neural network controller with a 5-13-1 (input, hidden input, output) architecture. The inputs to the controller consist of two delayed reference inputs, two delayed plant outputs, and one delayed controller output. A sampling interval of 0.2 seconds is used. To run this method, following steps below is required:-

3.4.1.1 Steps 1:

Start MATLAB.

3.4.1.2 Step 2:

Run the demo model by typing *Model Reference Control* in the MATLAB command window. This command starts Simulink and creates the following model window. The Model Reference Control block has already been placed in the model as shown in figure 1 below.

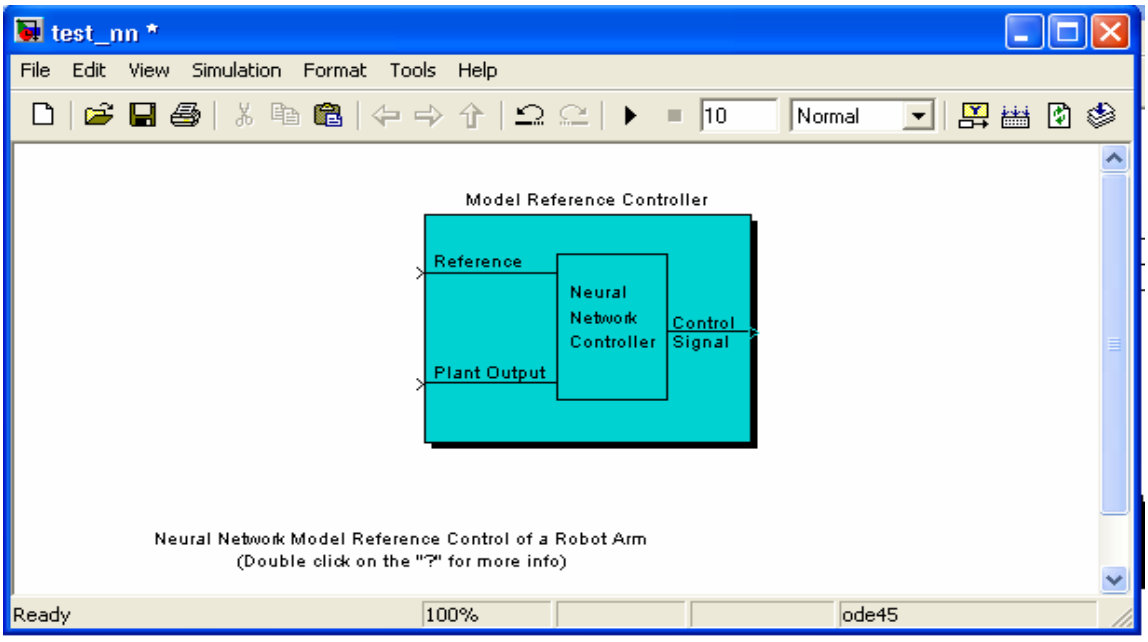


Figure 1

3.4.1.3 Step 3:

Create the DC motor modeling using state space matrix, then demonstrated it in Simulink form as shown in figure 2, and then embedded it in a Simulink Masked Block like shown in figure 2.

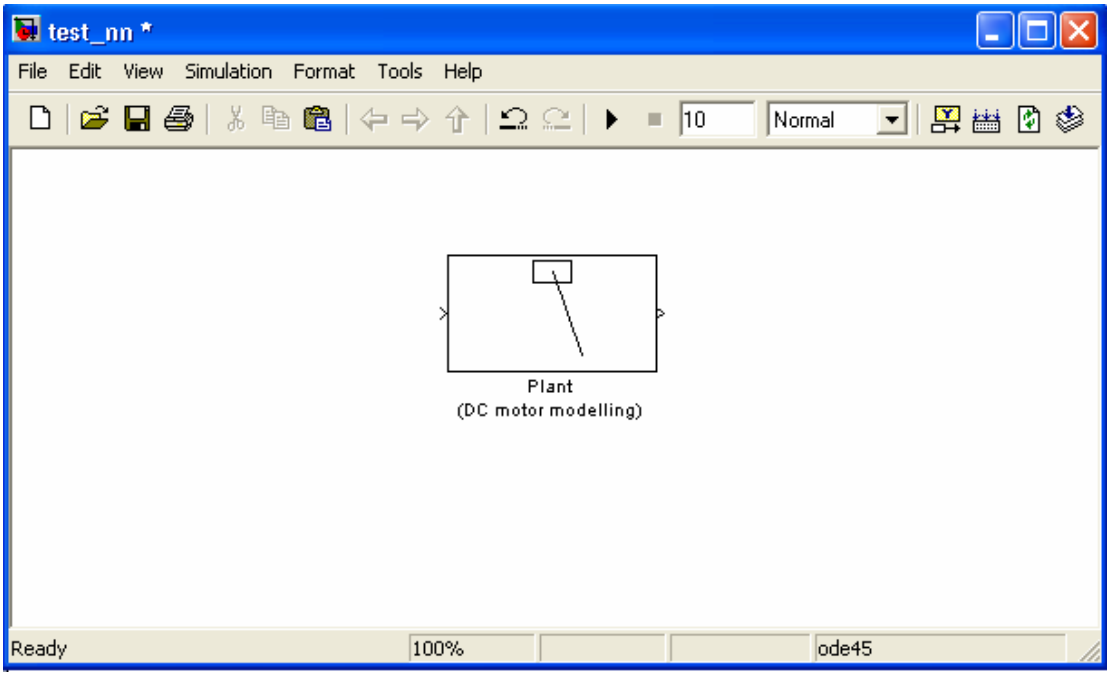


Figure 2

To seek the values of each matrixes and components used in DC Motor Modeling. First, double click the Simulink Masked Block. The figure as shown below will appears.

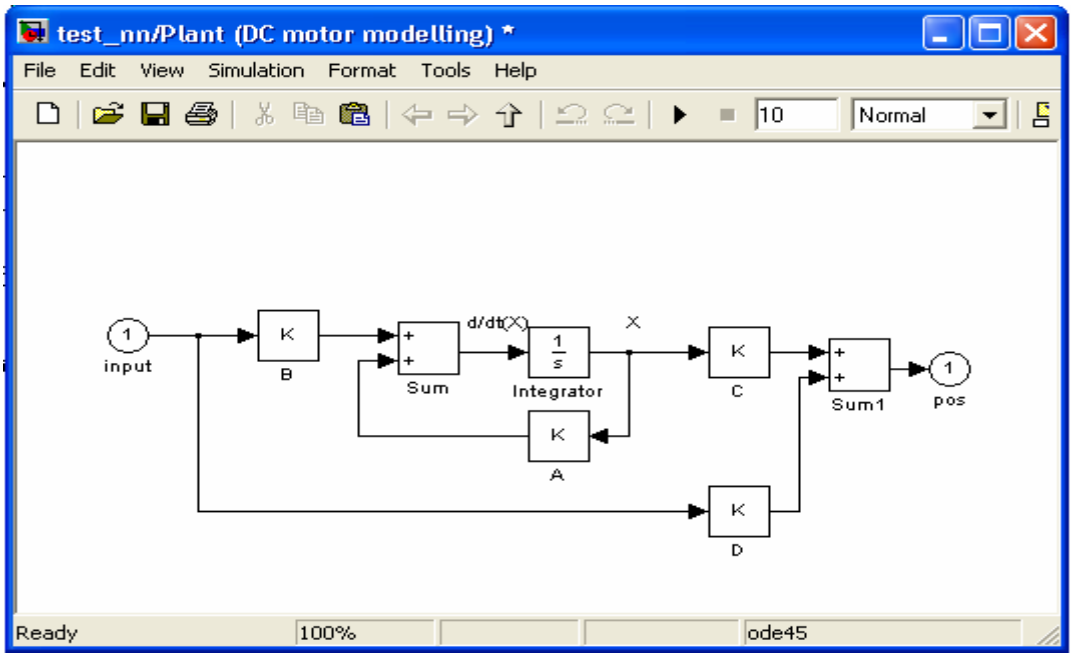


Figure 3

Then, go to *File* stated at the above menu and select *preferences*, the *Preferences* window will pop up automatically.

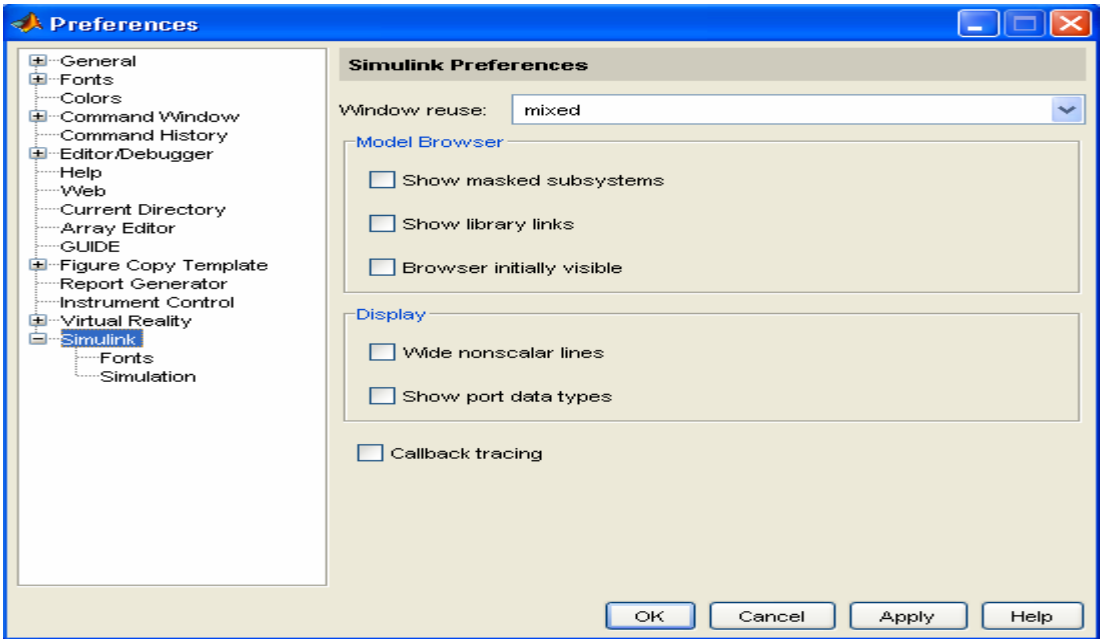


Figure 4

Afterward, go to Simulink which at the right side of the window and select *simulation*, the figure below will appear and then click the *Launch Model Explorer*. The *Model Explorer* window will appear as shown in figure 6.

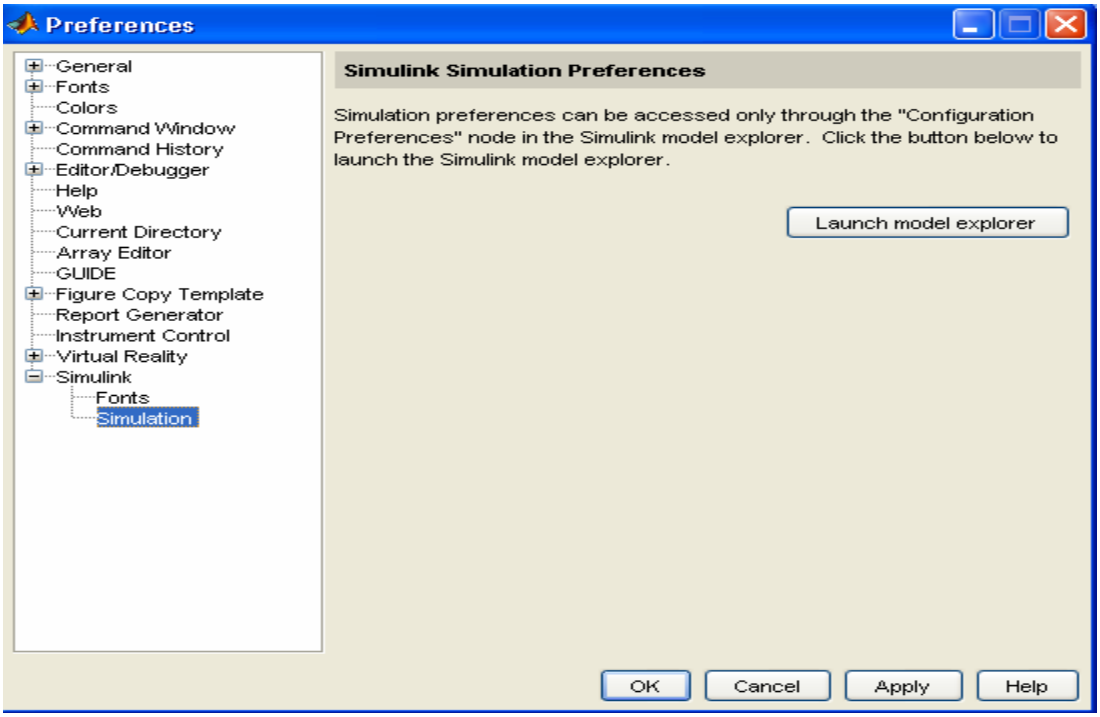


Figure 5

After that, choose the *Model Workspace* by clicking the *dcmotor_nn.* At the middle of the window, all the content of DC Motor Modeling system is shown. The values of all the data can be corrected or adjusted by just double clicking at each component at figure 3.

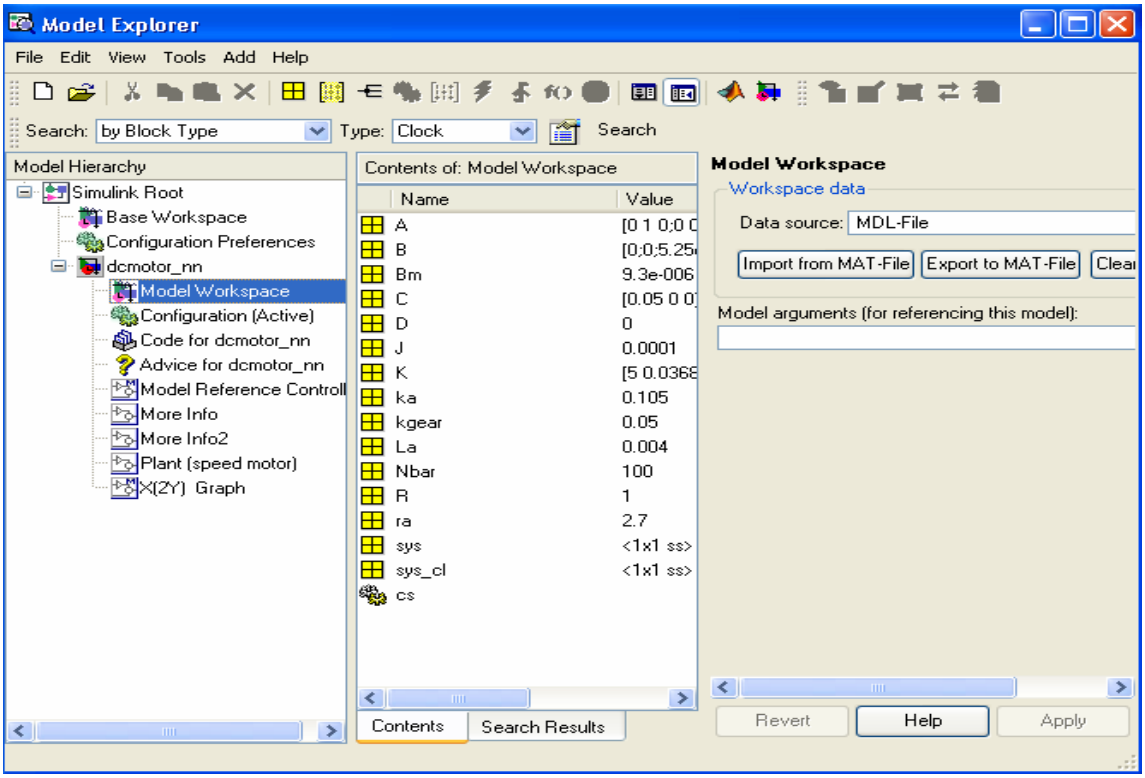


Figure 6

3.4.1.4 Step 4:

At steps 4, save the DC Motor Modeling as a *dcmotor_plant.mdl* or *othername.mdl.* This Simulink block diagram is important in order to fulfill the Training Data criteria at *Model Reference Controller* at Step 5.

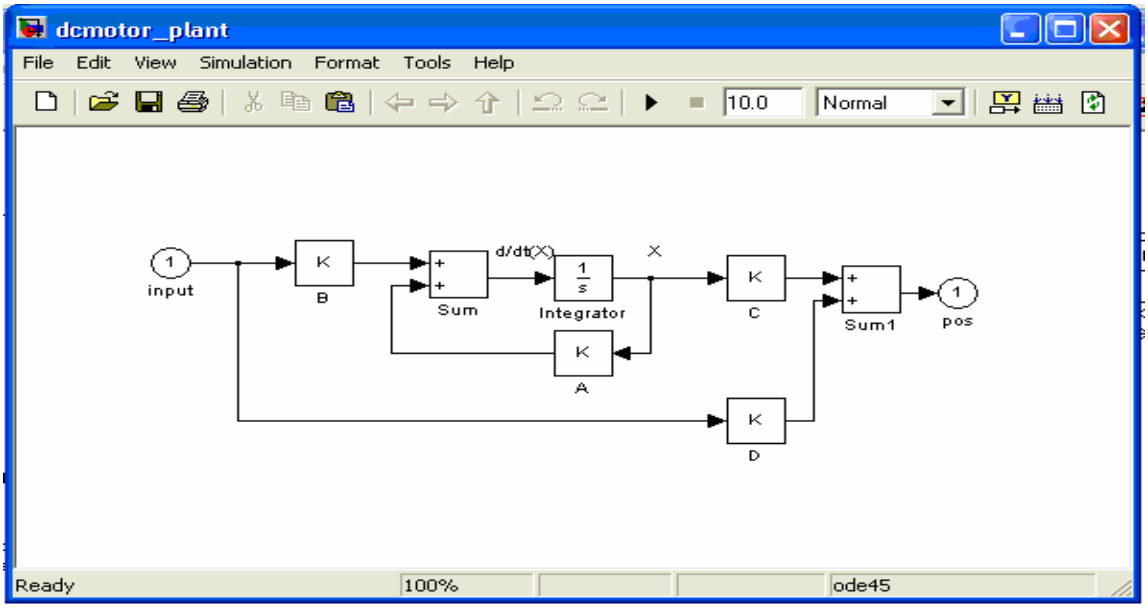


Figure 7

3.4.1.5 Step 5:

Create the complete Simulink block diagram of the Artificial Neural Network Controller for DC motor system as shown below.

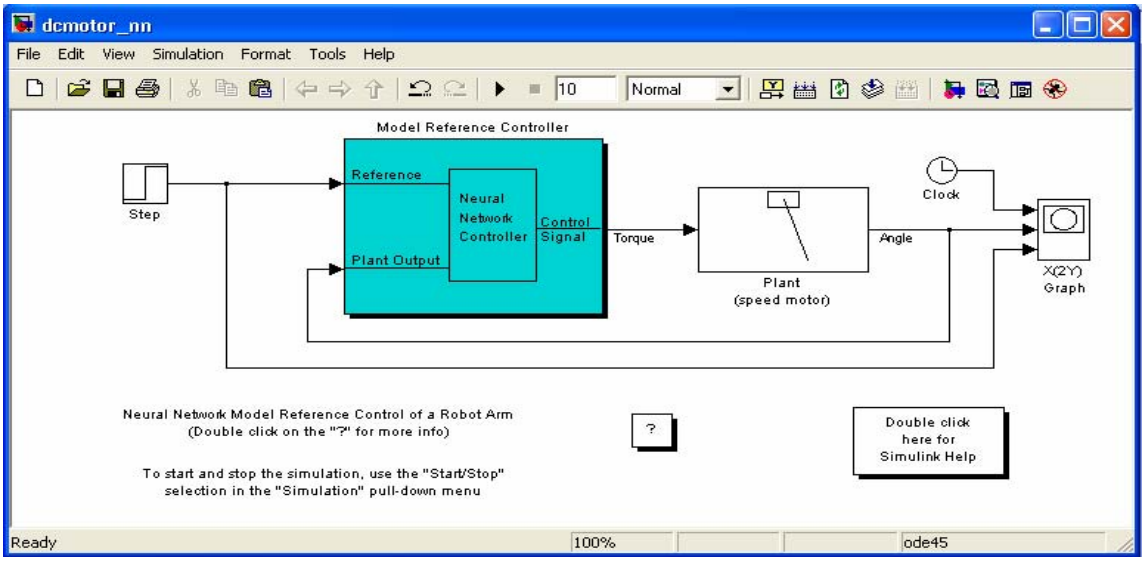


Figure 8

Now, the parameter of Model Reference Controller must be varied and adjusted in order to get the desired output speed of DC motor. To do that, double click at the model reference controller and the box of *model reference control* as below will appear,

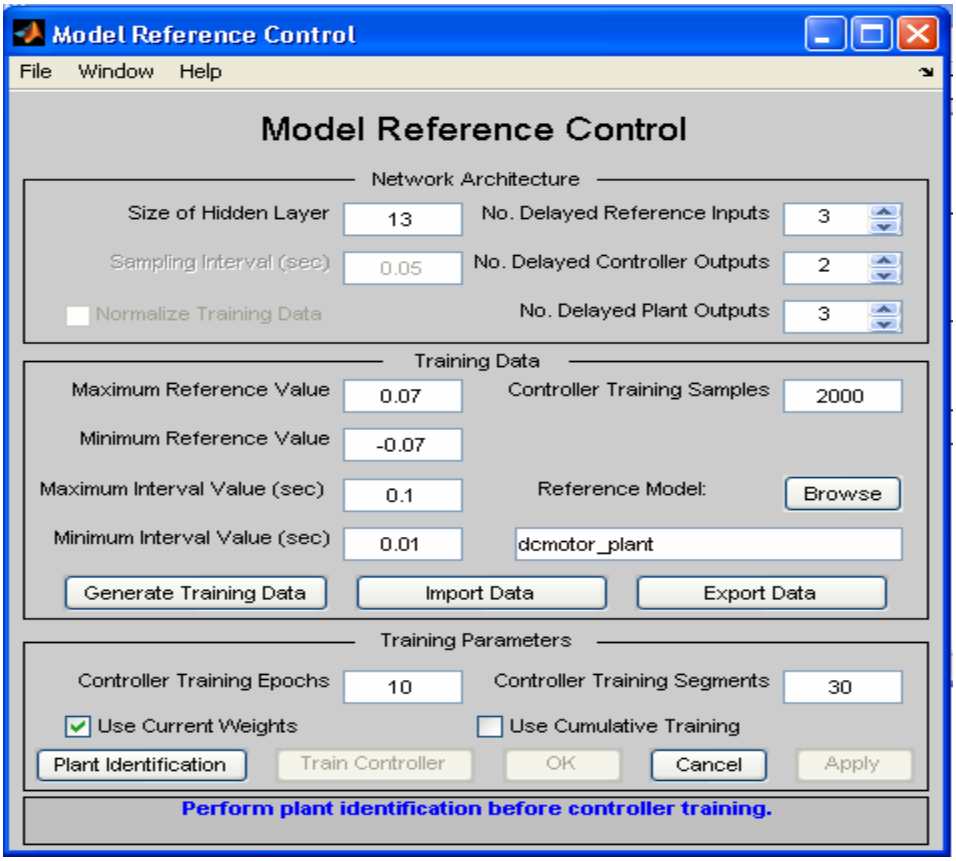


Figure 9

Then, the first steps to generate the required data are to insert all required value in *Network Architecture*. After that, the *Training data* must be inserted. In this section, the *reference model* at the right side and below *Controller Training Samples* must be inserted with *dcmotor_plant.mdl* created at steps 4. After all data is inserted, *Generate Training Data* button must then be selected. The box of *input-output data for NN Model Reference Control* will appears, it will shown the graph for desired values putted in *Controller Training Samples* for example 2000 as shown in figure above. But, in order

for it to do so, the time of 10 to 20 seconds or more is needed in order for it to accomplish the task.

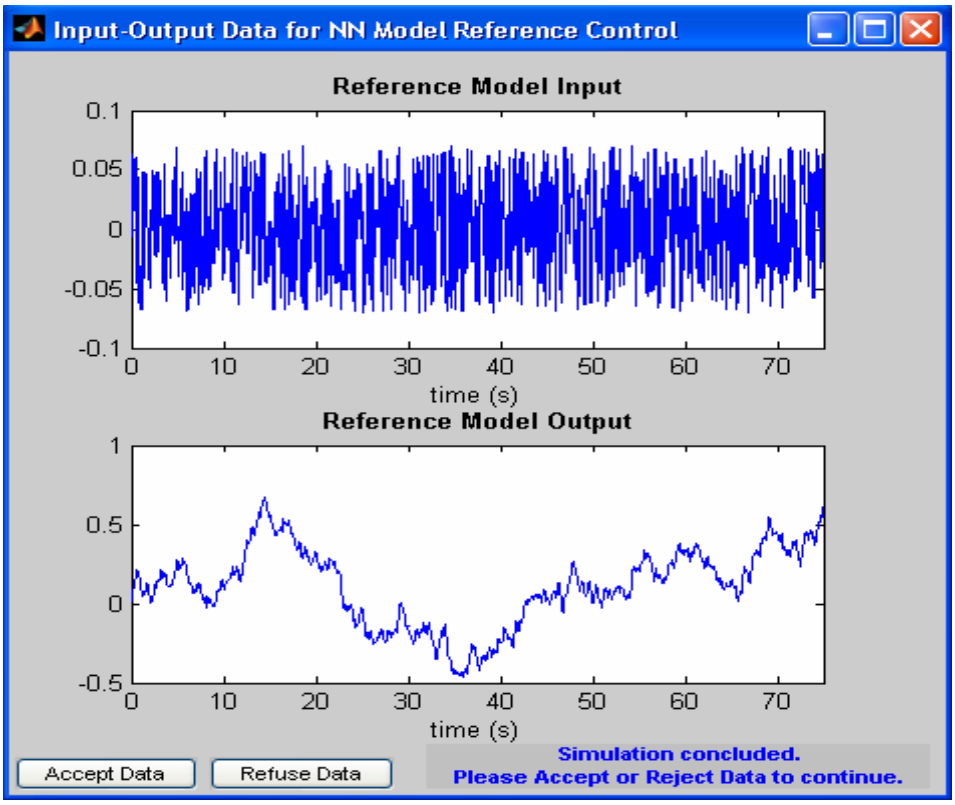


Figure 10

After that, click the *Accept Data* button. Here, the data will be approving as being accepted by the system. Then, go back to the first steps box, *Model Reference Control* and insert all the required data at *Training Parameters* area and tick only at *Use current weights,* and then click *Plant Identification* button.

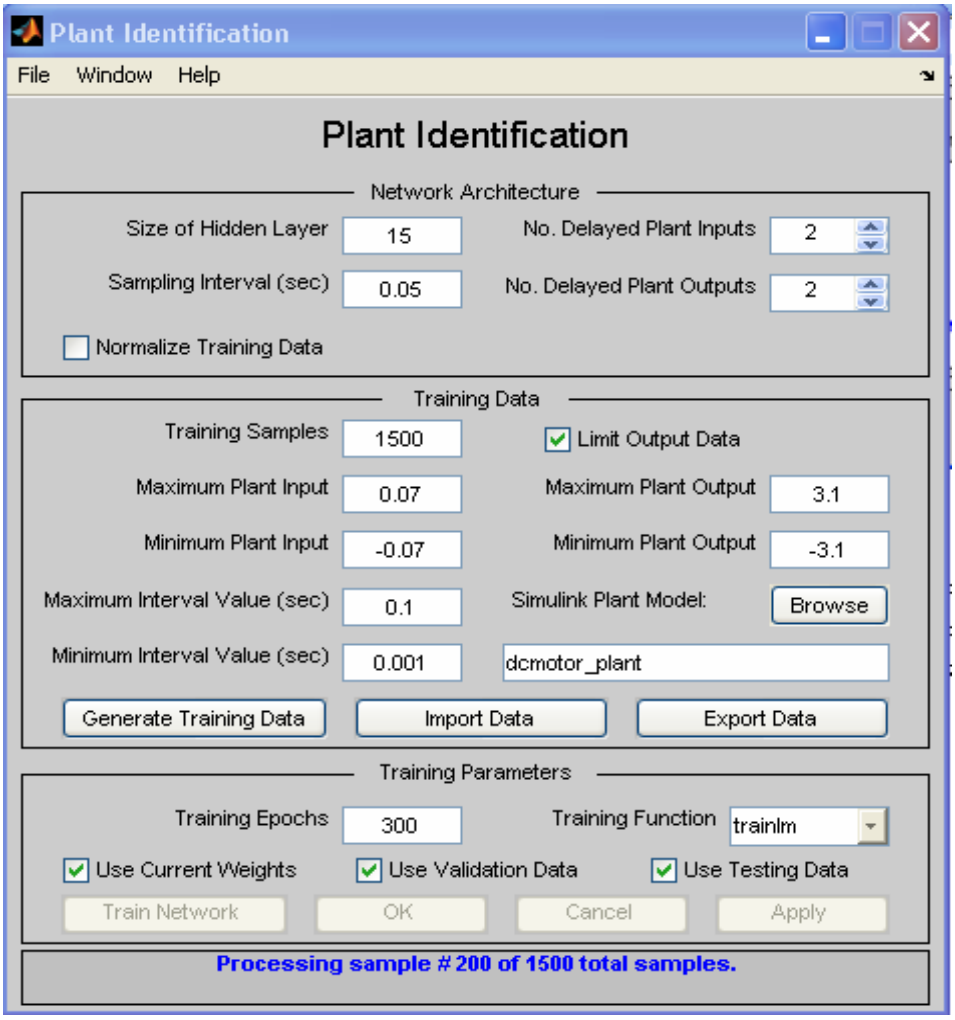


Figure 11

The *plant identification* box will appear as shown below, insert all required data once again and click *generate Training Data.* This will cause the *Plan input-output data* appears as shown below, and click *accept data* button.

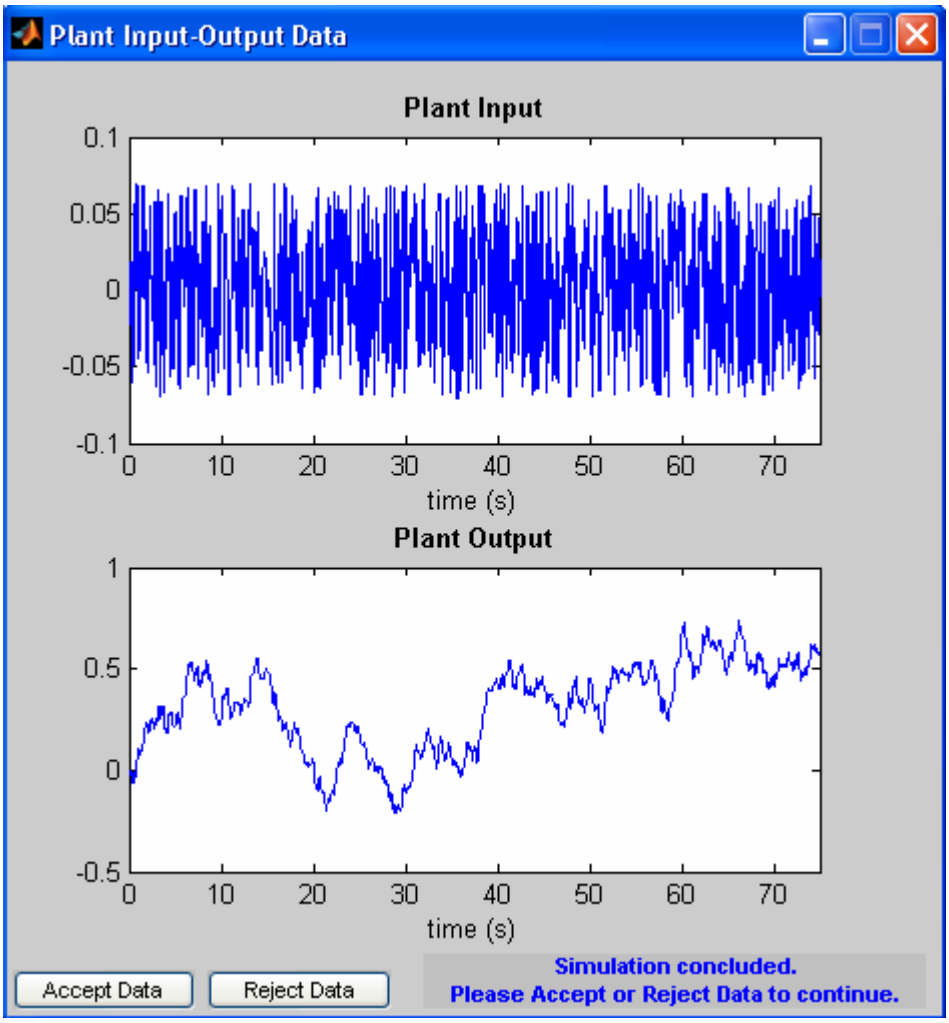


Figure 12

Then, go back to the first steps box, *Model Reference Control* and click *Train Network* at *Training Parameters*. Ensure that all parameter at Training Parameters like *training Epochs, Training function, use current weights, use validation data* and *use testing data* is fulfill. Then, click *Train Network,* the below graph will appears and wait until it finish train the system.

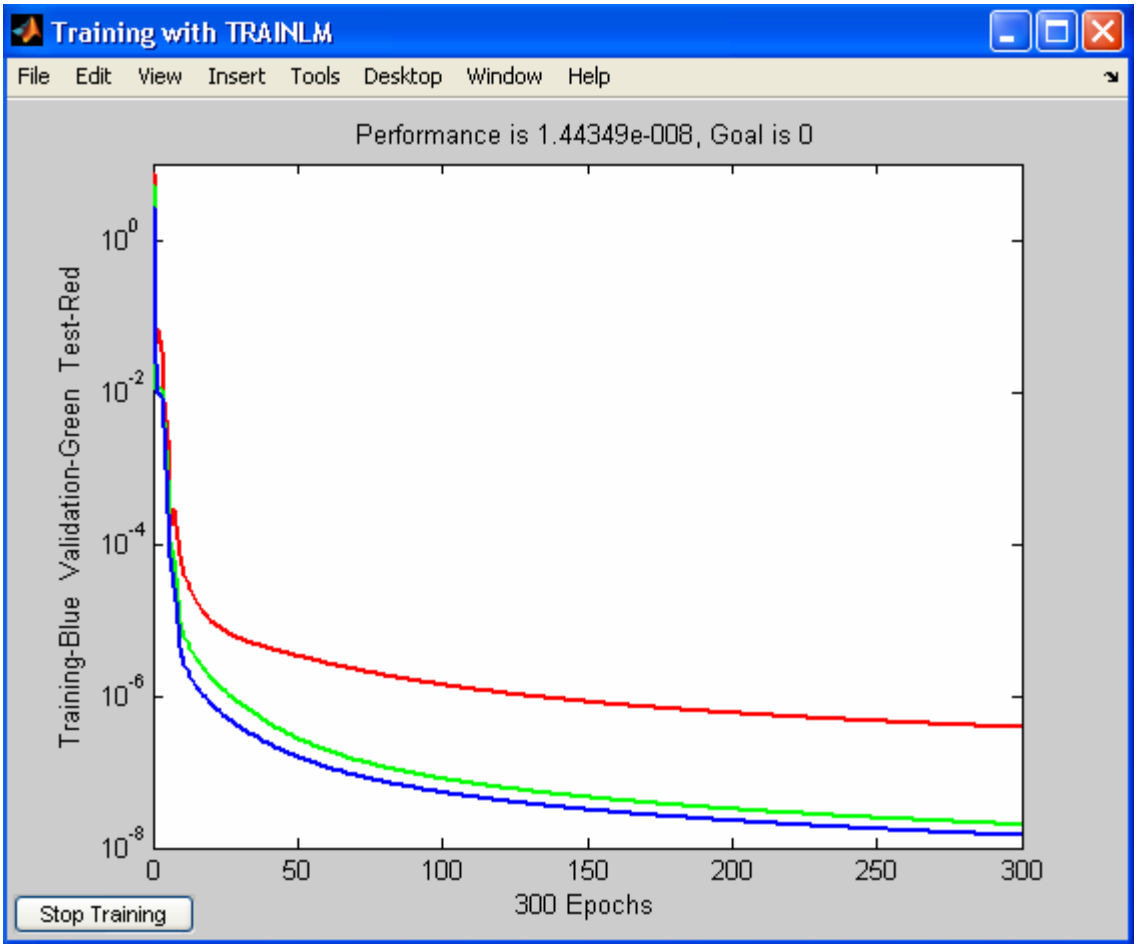


Figure 13

Then, go back to the first steps box, *Model Reference Control* and click *OK* button at Training Parameters. The three graph below will appears and shown all the required data and desired graph. The graphs are *Validation data, Training data and testing data.*

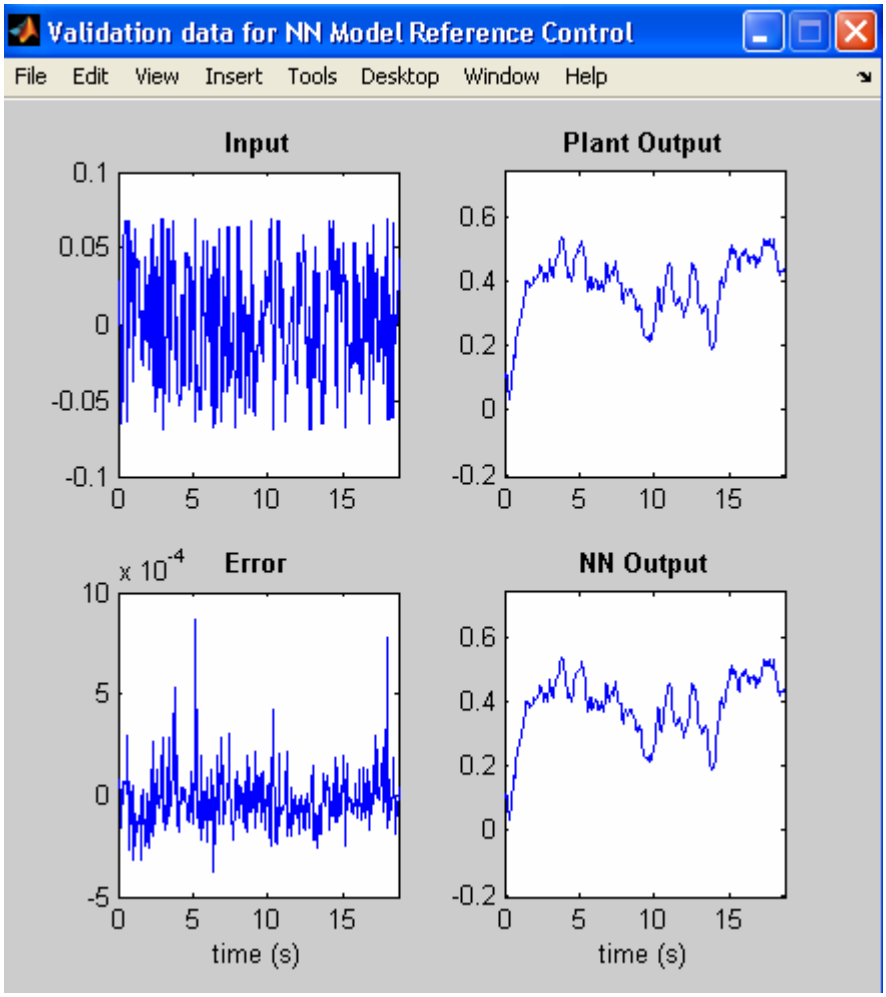

Validation data for NN Model Reference Control
File Edit View Insert Tools Desktop Window Help
Input
Plant Output
Error
NN Output
x 10^-4
time (s)
time (s)

Figure 14

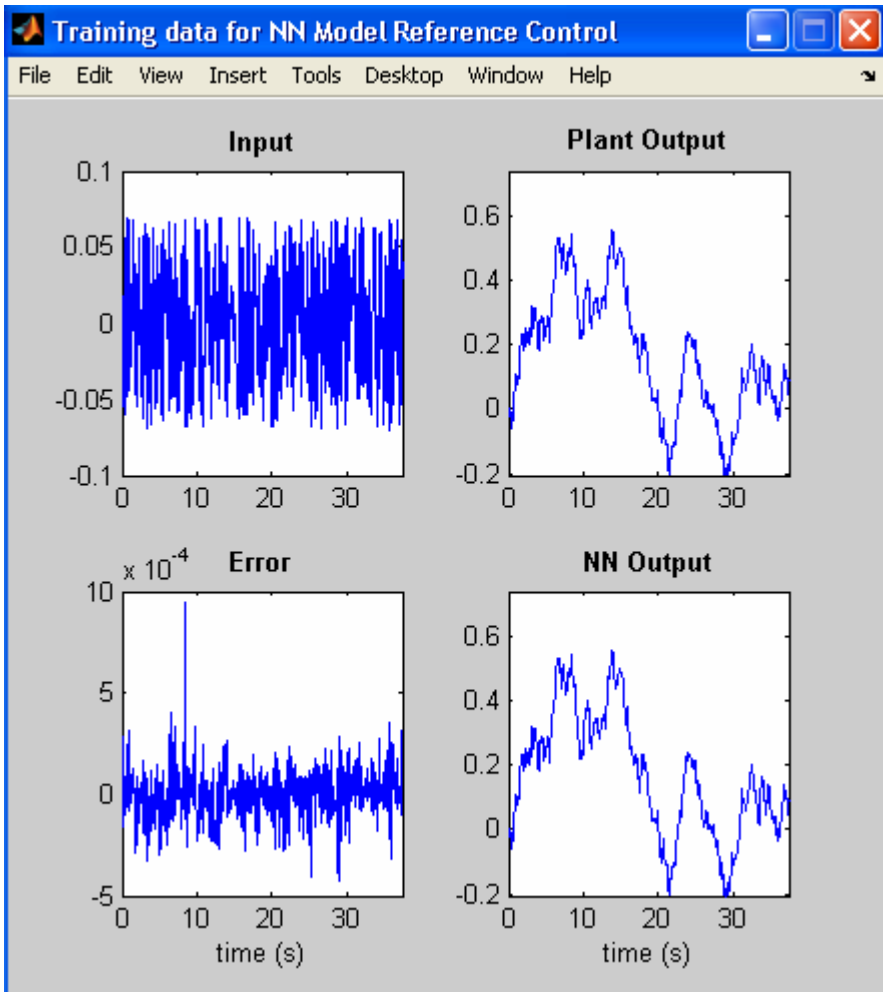

Training data for NN Model Reference Control
File Edit View Insert Tools Desktop Window Help
Input
Plant Output
Error
NN Output
x 10^-4
time (s)
time (s)

Figure 15

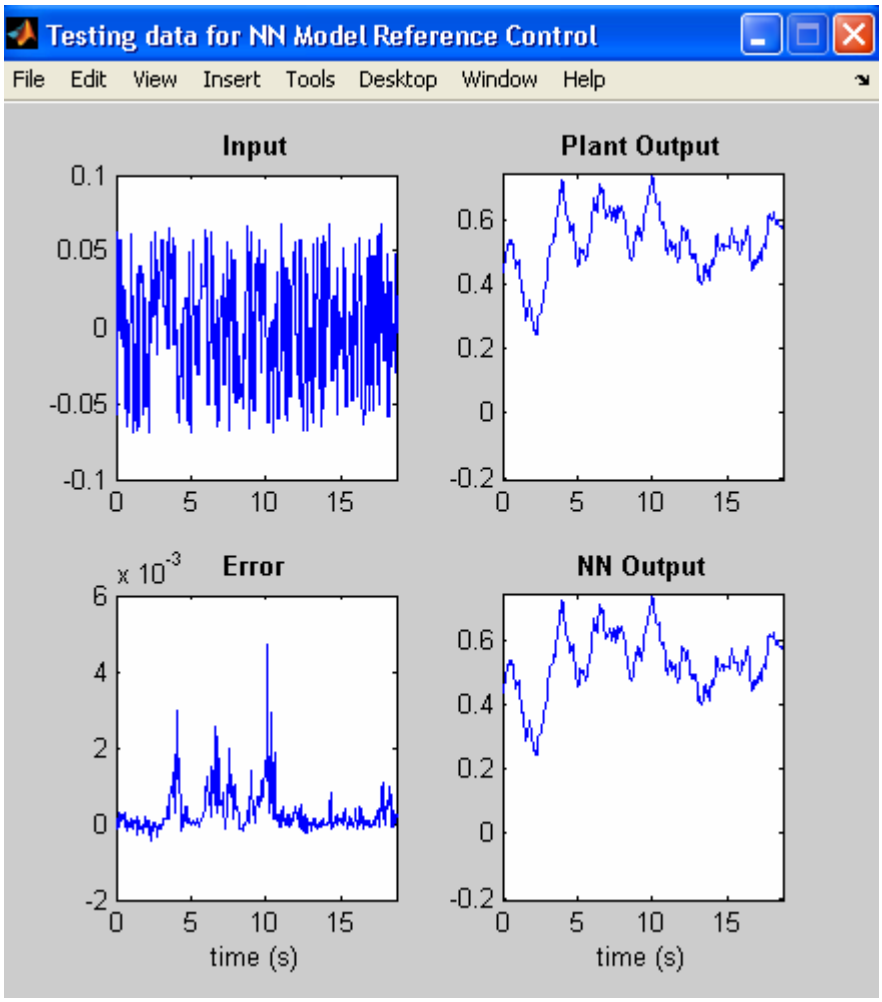


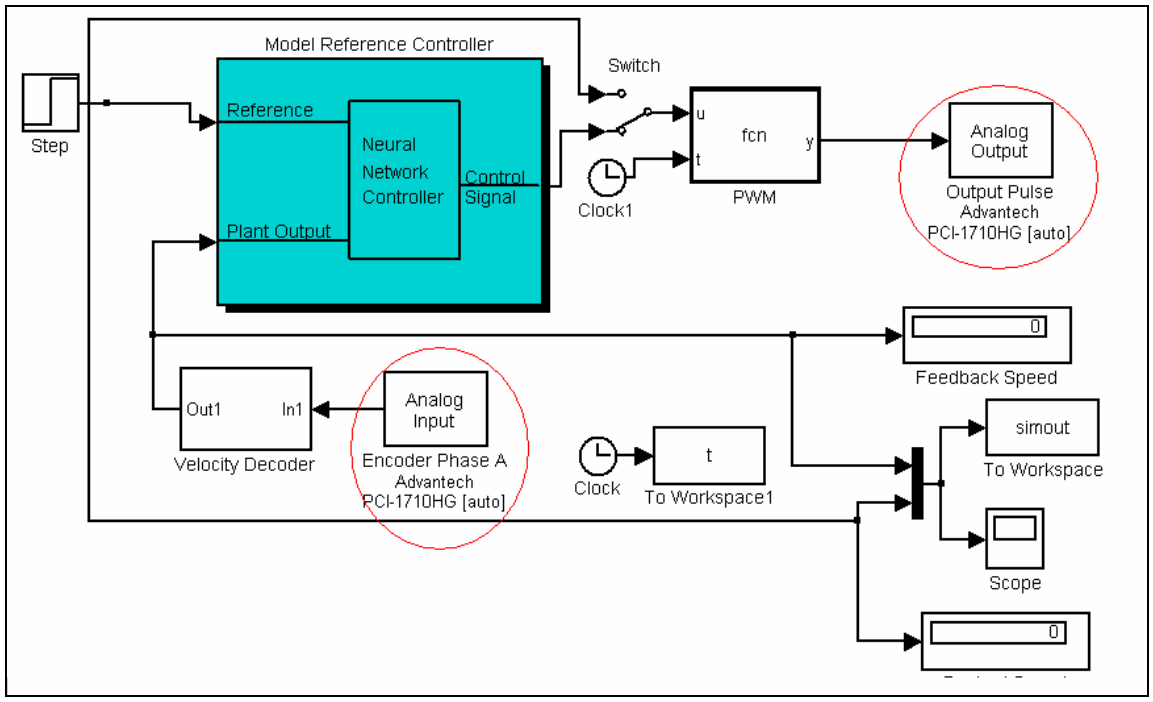
Figure 16

After all this steps is followed, the complete system of Artificial Neural Network controller now is ready to simulate. All steps from one (1) to five (5) should be repeatedly done if the output of the motor speed graph is not as required.

3.5 Interfacing of Matlab Simulink with DAQ Card

3.5.1 DAQ Card interfacing

Thus, In order to interface the Matlab software with the motor, the DAQ Card must be used. In doing so, the few parameters at Simulink block diagram of the controller system must be analyzed. The complete Simulink block diagram of the system is shown below.



Complete system of ANN Controller

The *analog Output* block diagram can be getting from the *Simulink Library Browser*. First of all, double click at the *Analog Input* block and the box of Block *Parameters: Analog Output* as shown below will appear.

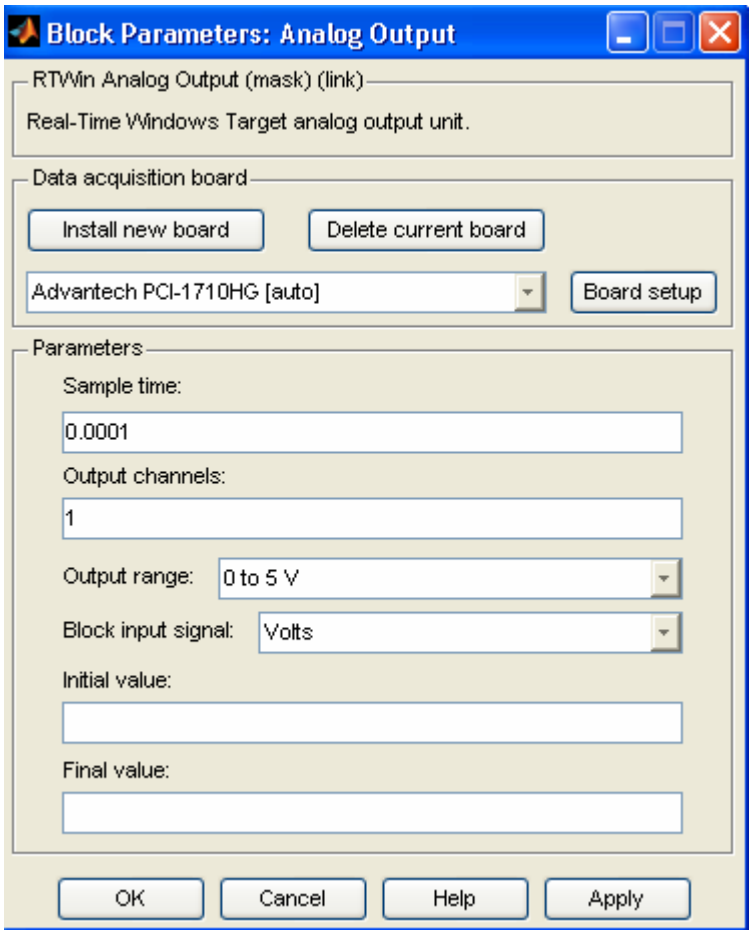


Figure 17

Then, click *Install new board* button and select the Advantech >> PCI-1710HG like as shown in figure 17. By clicking Board Setup, the box at figure 18 wills popup. Select the PCI Slot (DAQ Card Terminal Pin identification) or by select auto-detect for easiness. After that, back to *Parameters: Analog Output* and setting the rest of parameter. Each Analog input/Output must be initialized by doing this step. So, the Analog Input block should face the same step. This step will allow the interfacing between Matlab and motor.

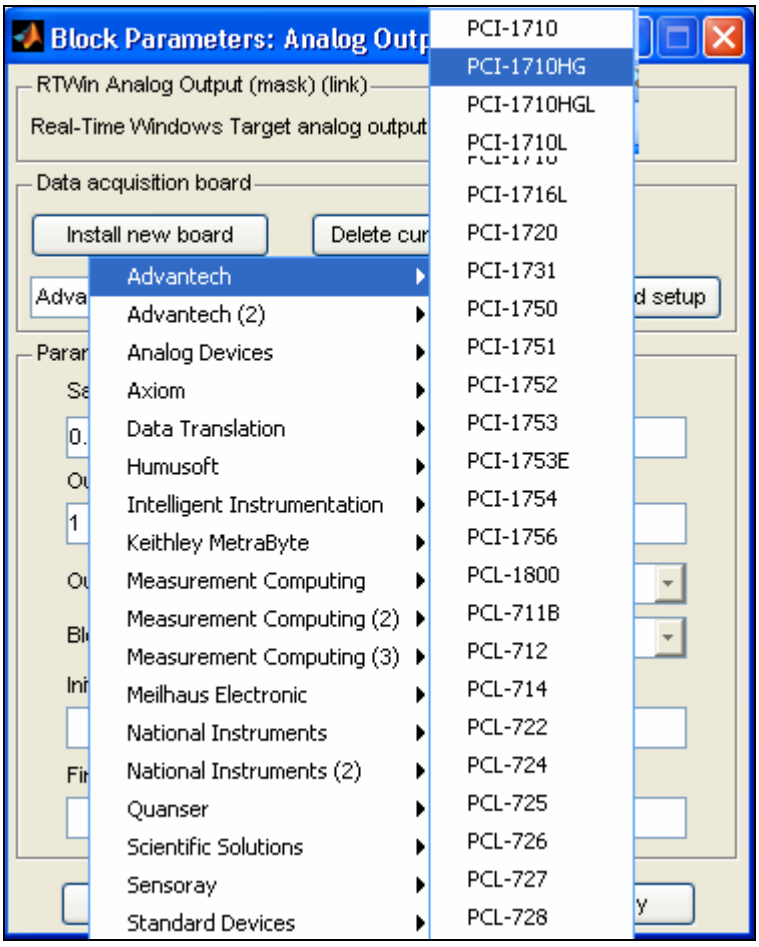


Figure 18

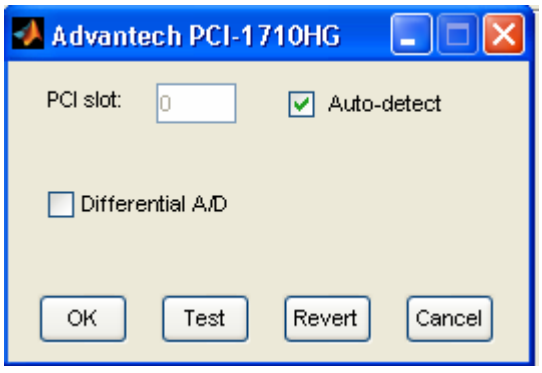


Figure 19

CHAPTER 4

RESULT AND DISCUSSION

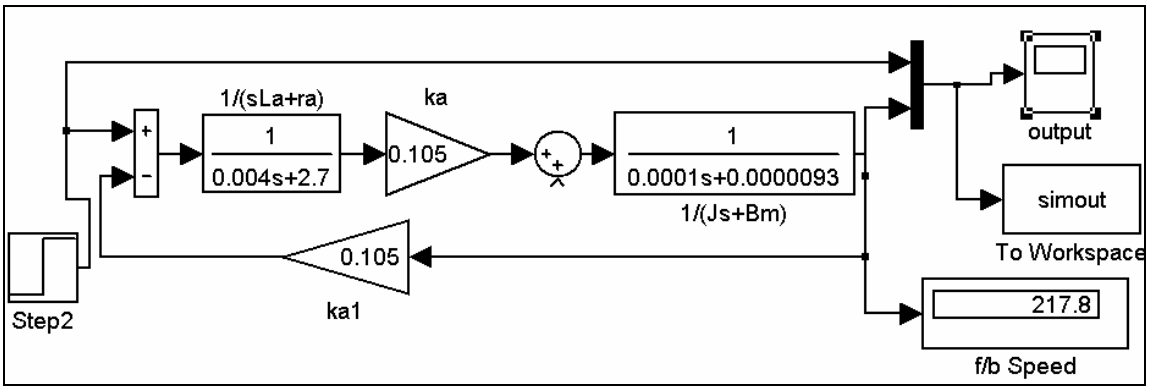
4.1 Simulation Result

4.1.1 The result without Neural Network Controller

4.1.1.1 Simulink Structure

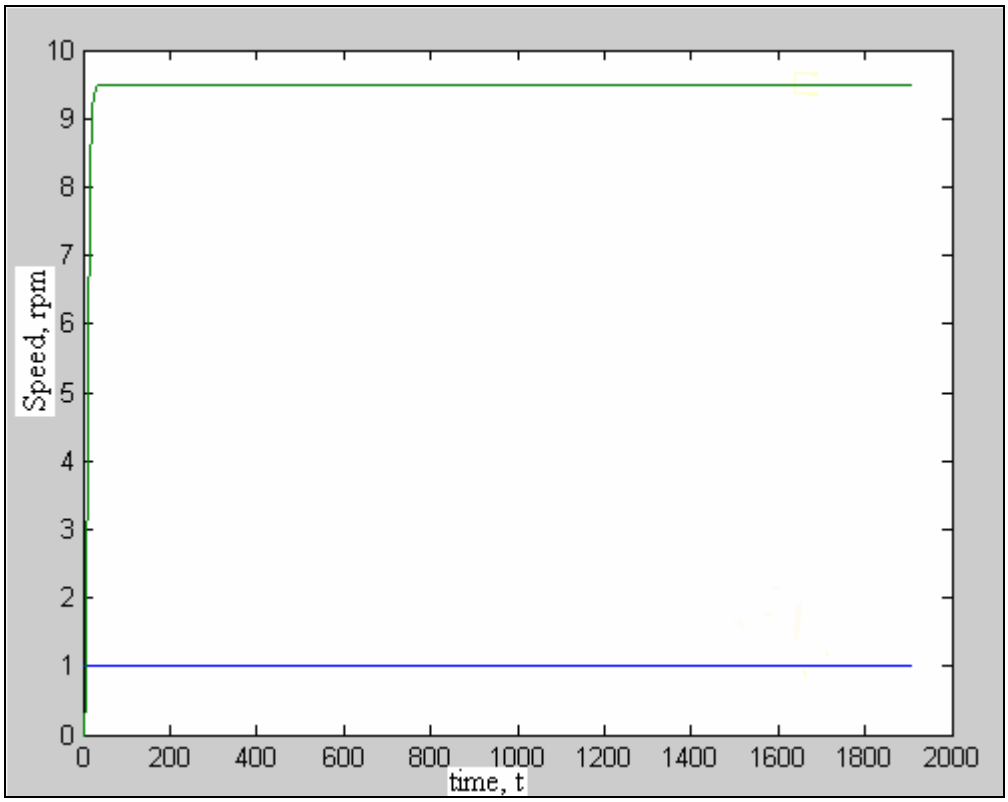
```
  To get started, select MATLAB Help or Demos from the Help menu.

>> ra=2.7;
>> La=0.004;
>> Bm=0.0000093;
>> ka=0.105;
>> kgear=0.05;
>> J=0.0001;
Warning: Using a default value of 0.2 for maximum step size. The simulation step size will
>>
```

Simulink Structure



DC Motor Modeling Simulink Block diagram

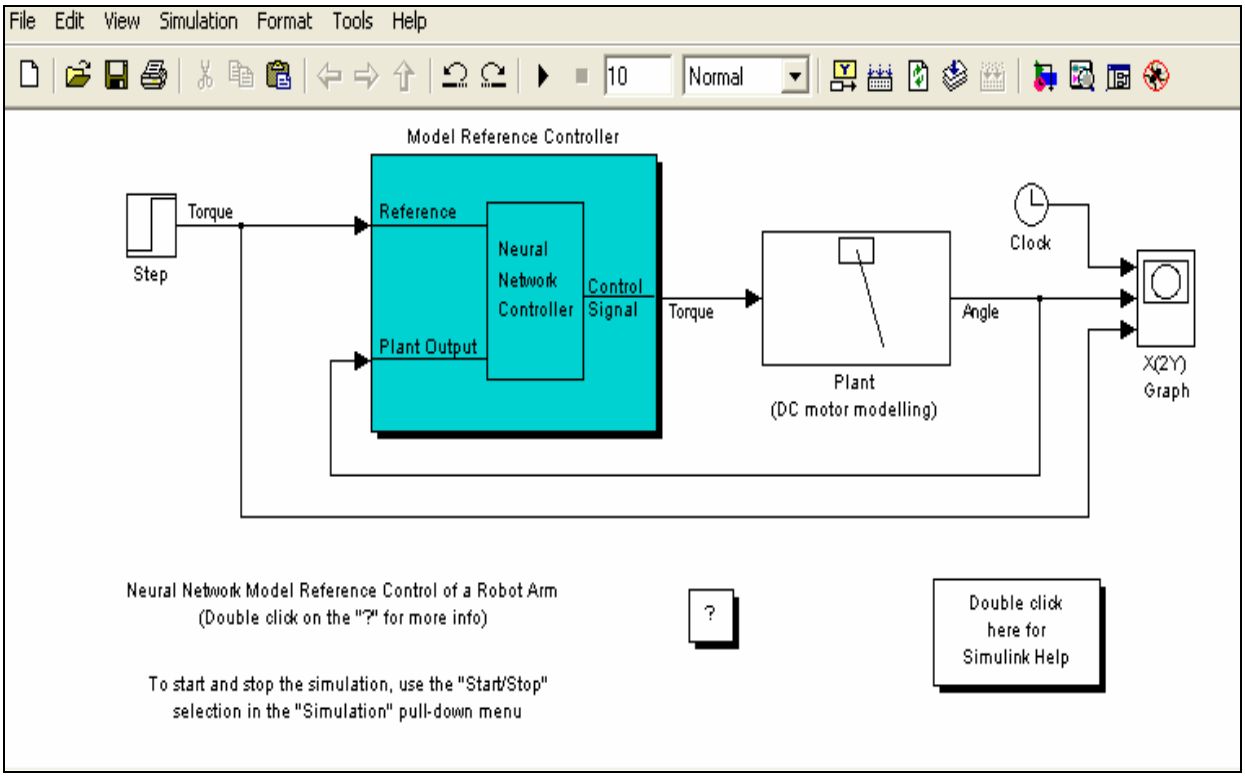
4.1.1.2 Graph



Graph of result without Neural Network Controller

4.1.2 The result with Neural Network Controller

4.1.2.1 Simulink Structure



Simulink Structure of ANN Controller

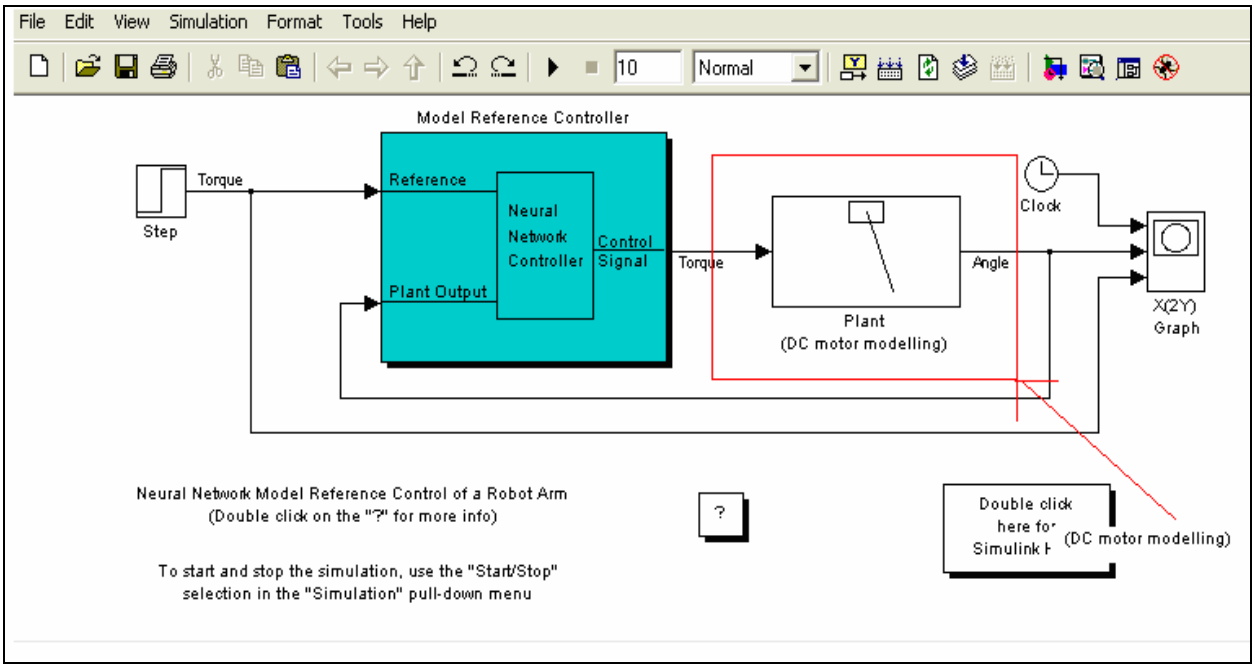
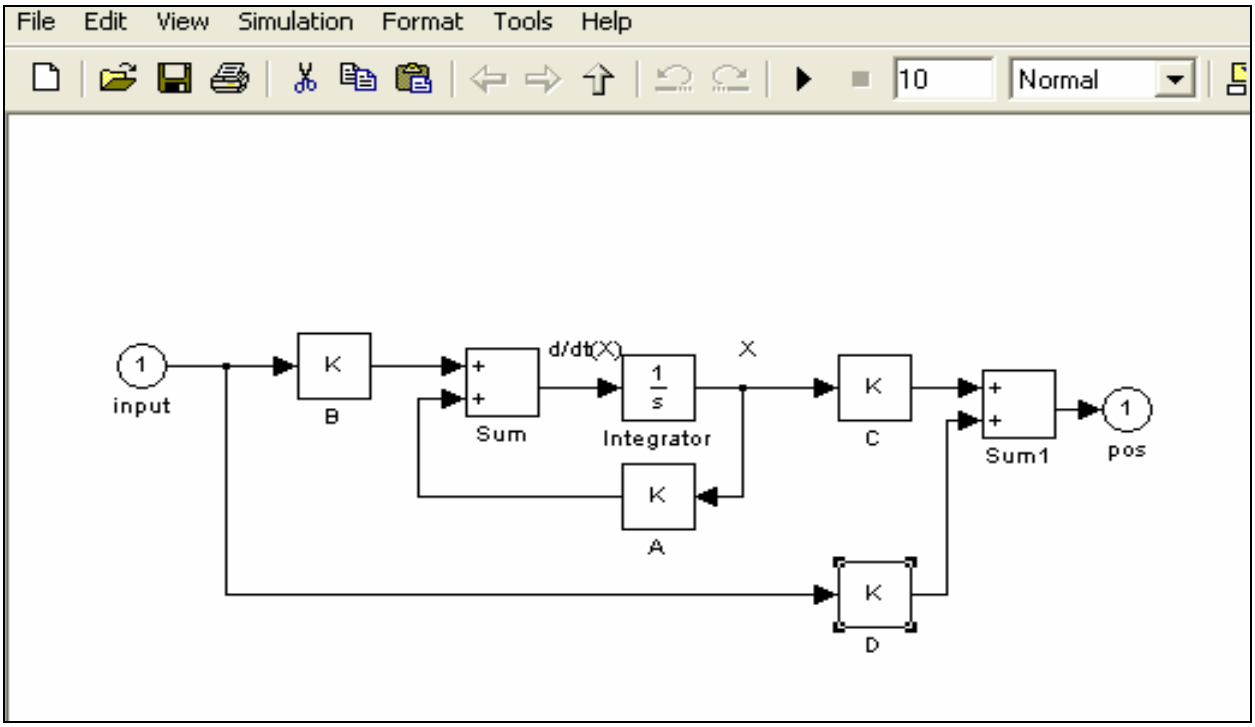
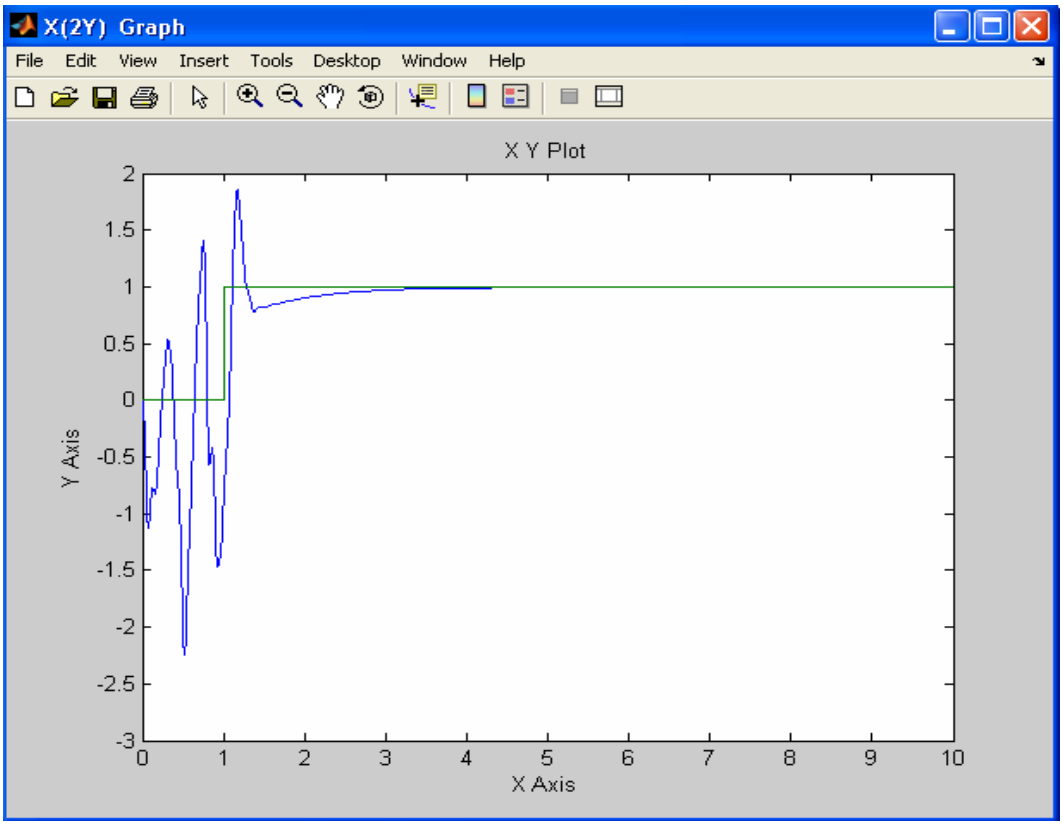


Figure 20



From DC motor modeling box

4.1.2.2 Graph



Graph with ANN Controller

4.2 Discussion

4.2.1 Comparison between without controller and with controller

4.2.1.1 Result without Neural Network Controller

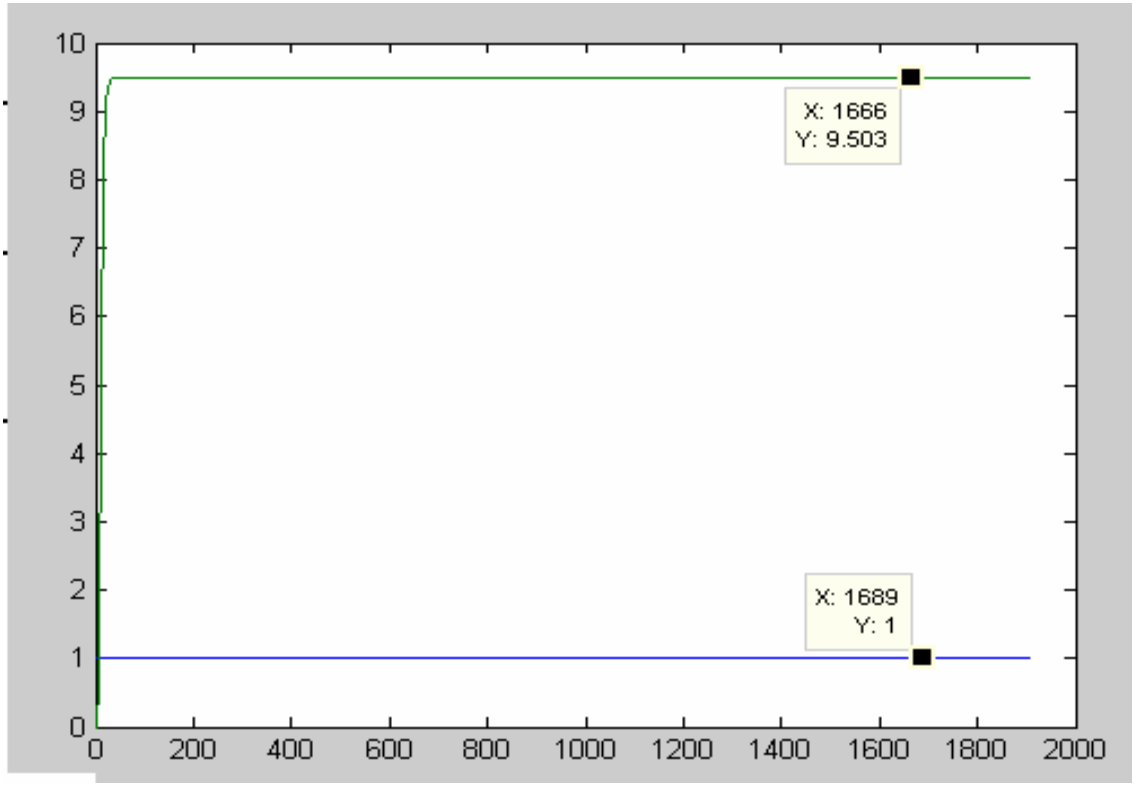


Figure 21

Maximum Overshoot, %OS : 2% of the system

Steady-state Error, Ess : 8.503rpm

Time delay, Td : -

Settling Time, Ts : -

Rise Time, Tr : -

4.2.1.2 Result with Neural Network Controller

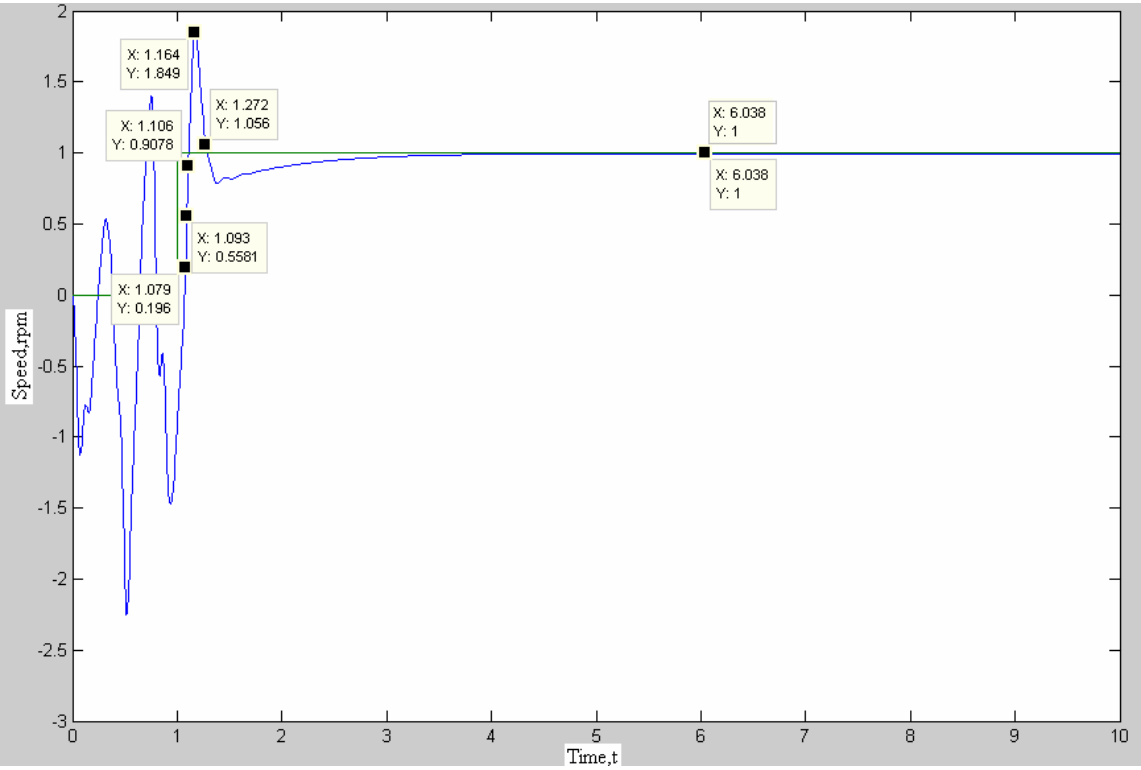


Figure 22

Maximum Overshoot, %OS	: 80%
Steady-state Error, Ess	: 0 rpm
Time delay, Td	: 0.014s
Settling Time, Ts	: 0.193s
Rise Time, Tr	: 0.017s

4.2.1.3 Disturbances factor

The disturbances occurred during starting the response and higher maximum overshoot are the effects created from adjusting the parameter of ANN Controller. To reduce the disturbances, the adjustment of the ANN controller must be done in many times, so that the right value can be achieved.

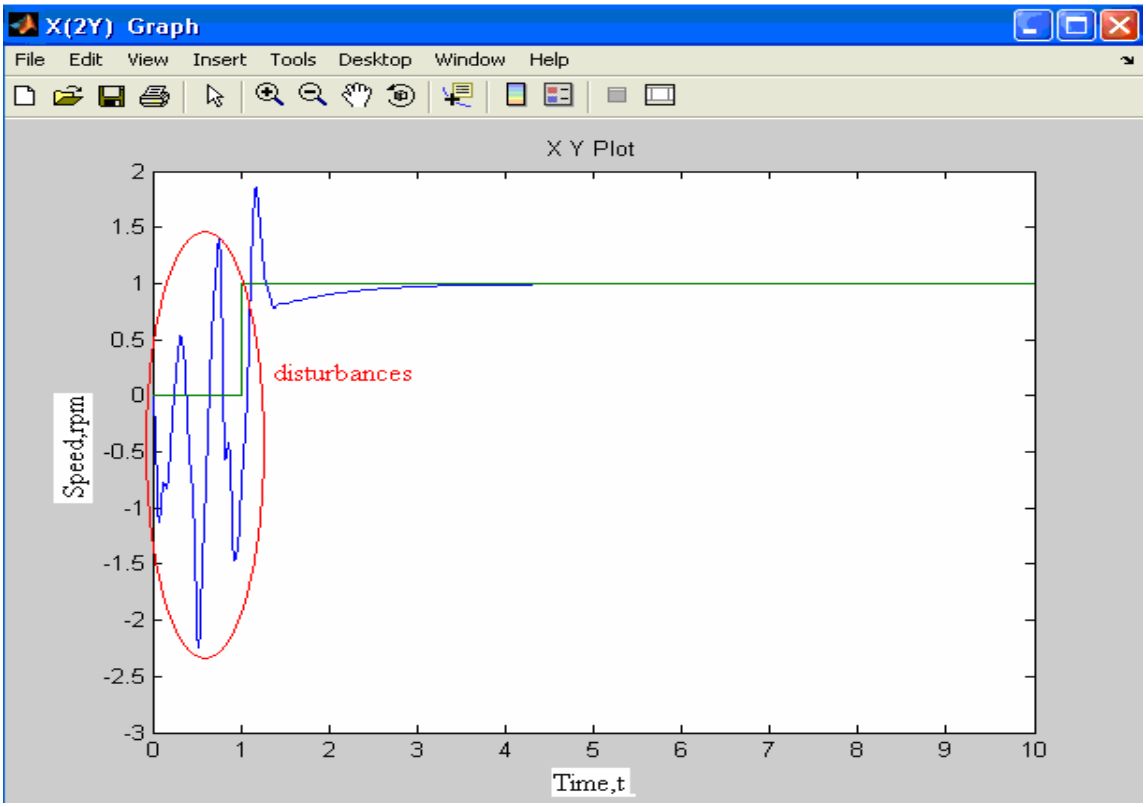


Figure 23

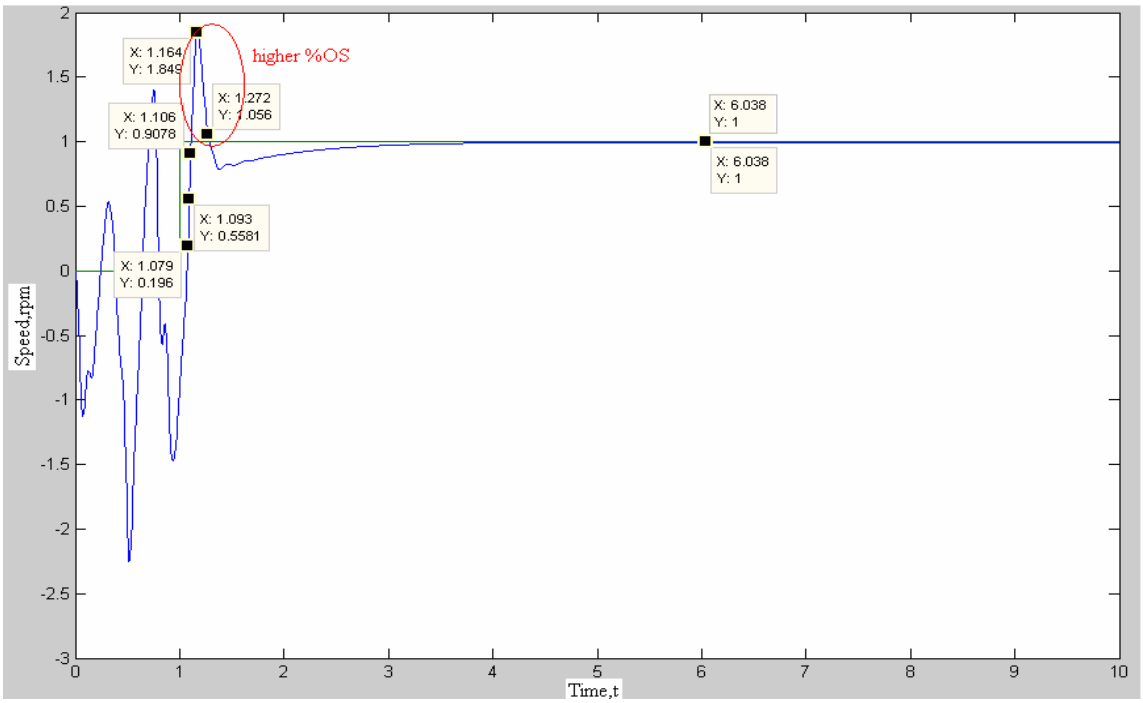

X: 1.164
Y: 1.849
higher %OS
X: 1.272
Y: 1.056
X: 1.106
Y: 0.9078
X: 6.038
Y: 1
X: 6.038
Y: 1
X: 1.093
Y: 0.5581
X: 1.079
Y: 0.196
Speed,rpm
Time,t

Figure 24

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Costing and commercialization

5.1.1 Costing

No	Costing types	Price	Other
1.	Components	RM152.30	-
2.	DAQ Card	-	Provided
3.	DC Motor	-	Provided
4.	Driver Motor	-	Provided
	TOTAL	RM152.30	

Total cost of project

5.1.2 Commercialization

This product can be commercialized for student's exposition and observation learns and it is suitable for universities field.

5.2 Conclusion

As for conclusion, the simulation of Artificial Neural Network Controller with DC motor model is achievable and it was able to control the speed of DC motor. Even interfacing the DC motor with MATLAB is successful but the modification is needed with the software design

5.2 Recommendation

Introduce motor control through internet when it can be implemented for student run experimental simulation without going to the laboratory.

REFERENCES

(i) Articles in internet

[1] http://en.wikipedia.org/wiki/Brushless_DC_electric_motor. (February 18, 2008)

[2] http://en.wikipedia.org/wiki/Neural_network. (February 18, 2008)

[3] http://ai-depot.com/articles/evolutionary-neural-networks-design-methodologies (February 18, 2008)

[4] http://en.wikipedia.org/wiki/Neural_network_software (February 18, 2008)

[5] http://sciencelinks.jp/j-east/article/200201/000020020101A0734606.php (February 18, 2008)

[6] http://www.alldatasheet.com (July 29, 2008)

(ii) BOOK

[7] William Palm III (2005), "*Introduction to MATLAB 7 for Engineers*", McGraw Hill.

[8] Sergey E. Lyshevski (2000), Electromechanical System, Electric Machines, and Applied Mechatronics, CRC Press LCC.

(iii) Software

[9] Matlab 7.1 software, 'Neural Network Toolbox', Mathworks