

Spring 2011

Efficient Control of DC Servomotor Systems Using Backpropagation Neural Networks

Yahia Makableh

Follow this and additional works at: <https://digitalcommons.georgiasouthern.edu/etd>

Recommended Citation

Makableh, Yahia, "Efficient Control of DC Servomotor Systems Using Backpropagation Neural Networks" (2011). *Electronic Theses and Dissertations*. 771.
<https://digitalcommons.georgiasouthern.edu/etd/771>

This thesis (open access) is brought to you for free and open access by the Graduate Studies, Jack N. Averitt College of at Digital Commons@Georgia Southern. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of Digital Commons@Georgia Southern. For more information, please contact digitalcommons@georgiasouthern.edu.

EFFICIENT CONTROL OF DC SERVOMOTOR SYSTEMS USING
BACKPROPAGATION NEURAL NETWORKS

by

YAHIA MAKABLEH

(Under the Direction of Fernando Rios-Gutierrez)

ABSTRACT

DC motor systems have played an important role in the improvement and development of the industrial revolution, making them the heart of different applications beside AC motor systems. Therefore, the development of a more efficient control strategy that can be used for the control of a DC servomotor system, and a well defined mathematical model that can be used for off line simulation are essential for this type of systems

Servomotor systems are known to have nonlinear parameters and dynamic factors, such as backlash, dead zone and Coulomb friction that make the systems hard to control using conventional control methods such as PID controllers. Also, the dynamics of the servomotor and outside factors add more complexity to the analysis of the system, for example when the load attached to the control system changes. Due to these parameters and factors new intelligent control techniques such as Neural Networks, genetic algorithms and Fuzzy logic methods are under research consideration in order to solve the complex problems related to the control of these nonlinear systems.

In this research we are using a combination of two multilayer neural networks to implement the control system: a) The first network is used to build a model that mimics the function of DC servomotor system, and b) a second network is used to implement the controller that controls the operation of the model network using backpropagation learning technique. The proposed combination of the two neural networks will be able to deal with the nonlinear parameters and dynamic factors involved in the original servomotor system and hence generate the proper control of the output speed and position. Off line simulation using MATLAB Neural Network toolbox is used to show final results, and to compare them with a conventional PID controller results for the same model.

INDEX WORDS: Neural Networks, Controller, DC Servomotor, Non-Linear Systems Modeling.

EFFICIENT CONTROL OF DC SERVOMOTOR SYSTEMS USING
BACKPROPAGATION NEURAL NETWORKS

by

YAHIA MAKABLEH

B.S, The University of Jordan, Jordan, 2009

A Thesis Submitted to the Graduate Faculty of Georgia Southern University in Partial

Fulfillment

of the Requirements for the Degree

MASTER OF SCIENCE

STATESBORO, GEORGIA

2011

© 2011
YAHIA MAKABLEH
All Rights Reserved

EFFICIENT CONTROL OF DC SERVOMOTOR SYSTEMS USING
BACKPROPAGATION NEURAL NETWORKS

by

YAHIA MAKABLEH

Major Professor: Fernando Rios-Gutierrez

Committee: M. Rocio Alba-Flores

Robert Cook

Electronic Version Approved:

May 2011

DEDICATION

To my Mom and Dad, I will always be as you know me, and I will be always searching for new knowledge and science. I love you both as you supported me all the way to this day. I miss you all.

To my wife Rana, I love you so much, and I will be with you for the rest of my life, as you have been supporting me through my studies. It was a tough year for us, but for the best. With all my Love

ACKNOWLEDGMENTS

I would like to thank all the people that supported me through my academic way, and helped me to understand the engineering concepts needed that were needed for our engineering life. My special thanks to Professor Fernando Rios-Gutiérrez, as he was beside me in every step throughout my research, and for his great effort to show me the best techniques and sources to solve different problems. Also I would like to thank Dr. Frank Goforth, for his great support to the graduate students and for his effort to keep us on track and to be on time. Many thanks also goes to Dr. Biswanath Samanta, as he was leading me through different control techniques and gave me great ideas to help me succeed in my research.

I will never forget the great support from my family, from my parents, brothers, sisters and my loved wife, they helped me a lot, I would like to thank you all to be on my side, and for all of your great support, with you I can always succeed.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS.....	7
LIST OF TABLES.....	10
LIST OF FIGURE.....	11
CHAPTER	
1 INTRODUCTION TO THE STUDY.....	14
DC Servomotor Systems.....	15
Artificial Neural Networks.....	16
Mathematical Modeling of an ANN.....	17
2 REVIEW OF RELATED LITERATURE.....	21
DC Servomotor Literature Review.....	21
Artificial Neural Networks Literature Review.....	23
3 METHODOLOGY.....	27
Multi Layer Neural Network Model.....	27
Neural Network Transfer Functions.....	29
4 Feed Forward ANN and Backpropagation Training Algorithm.....	32
PID Control.....	33
DC SERVOMOTOR.....	37
DC Servomotor Model.....	37
DC Servomotor Response.....	39
5 ANN MODELS STRUCTURES.....	41
Servomotor Neural Network Model.....	41
Neural Network Controller Model.....	52

6 CONTROLLER DESIGN AND SIMULATION RESULTS.....	57
Neural Network System.....	57
Complete System Results with Sine Wave Input.....	60
Complete System Results with Step Input.....	63
PID Controller Response.....	64
NN Controller Response.....	66
7 CONCLUSIONS, DELIVERIES, FUTURE WORK AND SUMMARY.....	70
Conclusions.....	70
Deliveries.....	71
Future Work.....	71
Summary.....	70
REFERENCES.....	71
APPENDIX A.....	72

LIST OF TABLES

	Page
Table 3.1. List of Neural Network Transfer Functions.....	31
Table 5.1 DC Servomotor parameter values.....	42

LIST OF FIGURES

	Page
Figure 1.1. Servomotor circuit diagram.....	15
Figure 1.2 . Biological neuron model.....	17
Figure 1.3. Artificial neuron structure.....	18
Figure 1.4. Multi input neuron.....	19
Figure 1.5. Neural network structure.....	19
Figure 2.1. Servo mechanism basic structure.....	22
Figure 2.2. Servomotor circuit diagram.....	24
Figure 2.3. Structure of single neuron network.....	25
Figure 2.4. single neuron structure.....	25
Figure 2.5. Basic ANN structure.....	26
Figure 3.1. Three layer neural network.....	28
Figure 3.2. Hard-Limit transfer function.....	29
Figure 3.3. The Linear transfer function.....	30
Figure 3.4 The Log-Sigmoid transfer function.....	30
Figure 3.5. Proportional and Derivative terms effect.....	34
Figure 3.6. Integral term effect.....	35
Figure 3.7 Derivative term effect.....	35
Figure 4.1. Servomotor SIMULINK block Diagram.....	39
Figure 4.2. Second order system response.....	40
Figure 5.1. Servomotor SIMULINK model.....	41
Figure 5.2. Input sin wave input.....	43
Figure 5.3. Input signal after dead zone effect.....	43

Figure 5.4. Output Speed.....	44
Figure 5.5 Step input before dead zone.....	45
Figure 5.6. Input signal after dead zone.....	45
Figure 5.7. Speed output signal.....	46
Figure 5.8. Network response before training.....	47
Figure 5.9. Network response after training.....	48
Figure 5.10. Performance Plot.....	49
Figure 5.11. Regression plot.....	49
Figure 5.12. Training tool window of the NN motor model.....	50
Figure 5.13. PID Controller SIMULINK diagram.....	51
Figure 5.14. PID input data.....	52
Figure 5.15. PID output plot.....	53
Figure 5.16. NN controller response before training.....	53
Figure 5.17. NN controller response after training.....	54
Figure 5.18. Performance plot of the NN controller.....	54
Figure 5.19. Training tool window of the NN controller model.....	55
Figure 6.1 NN motor model.....	58
Figure 6.2. Internal NN structure.....	58
Figure 6.3. NN motor model complete diagram.....	59
Figure 6.4. Output signal of NN motor model.....	59
Figure 6.5. Complete system diagram.....	60
Figure 6.6 Input signal.....	61
Figure 6.7. NN Controller Input.....	61
Figure 6.8. NN Controller output.....	62
Figure 6.9. Output speed signal (controlled).....	62

Figure 6.10. Step input response of the NN controller.....	63
Figure 6.11. Output speed with step input.....	64
Figure 6.12. PID controller input and output signals.....	65
Figure 6.13. Output speed with PID controller.....	65

CHAPTER 1

INTRODUCTION TO THE STUDY

Automatic systems are common place in our daily life, they can be found in almost any electronic devices and appliances we use daily, starting from air conditioning systems, automatic doors, and automotive cruise control systems to more advanced technologies such as robotic arms, production lines and thousands of industrial and scientific applications.

DC servomotors are one of the main components of automatic systems; any automatic system should have an actuator module that makes the system to actually perform its function. The most common actuator used to perform this task is the DC servomotor. Historically, DC servomotors also played a vital role in the development of the computer's disk drive system; which make them one of the most important components in our life that we cannot live without it. Due to their importance, the design of controllers for these systems has been an interesting area for researchers from all over the world.

However, even with all of their useful applications and usage, servomotor systems still suffer from several non-linear behaviors and parameters affecting their performance, which may lead for the motor to require more complex controlling schemes, or having higher energy consumption and faulty functions in some cases. For these purposes the controller design of DC servomotor system is an interesting area that still offers multiple topics for research, especially after the discovery of Artificial Neural Networks (ANN) and their possible usage for intelligent control purposes. From this point of view and the importance of having high efficient servomotor systems, the use of ANN played a vital

role in designing smart controllers that can eliminate or cope with the non-linear effects found in servomotor systems and improve the functions they are used for.

DC Servomotor Systems

A Servomotor system consists of different mechanical and electrical components, the different components are integrated together to perform the function of the servomotor, Figure 1.1 below shows a typical model of a servomotor system (Nise, 2008)

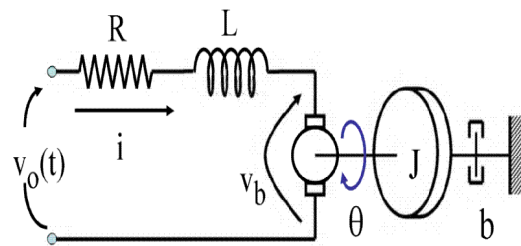


Figure 1.1. Servomotor circuit diagram.

Its clear that the servomotor has two main components, the first is the electrical component; which consists of resistance R , inductance L , input voltage $V_o(t)$ and the back electromotive force V_b . The second component of the servomotor is the mechanical part, from which we get the useful mechanical rotational movement at the shaft. The mechanical parts are the motor's shaft, inertia of the motor and load inertia J and damping b . θ Refers to the angular position of the output shaft which can be used later to find the angular speed of the shaft ω .

DC Servomotors have good torque and speed characteristics; also they have the ability to be controlled by changing the voltage signal connected to the input. These characteristics made them powerful actuators used everywhere. The main concern

about DC servomotors is how to eliminate the non-linear characteristics that affect both the output speed and position. Another important non-linear behavior in servomotors is the saturation effect, in which the output of the motor cannot reach the desired value. For example, if we want to reach a 100 rpm angular speed when we supply a 12 volt input voltage, but the motor can only reach 90 rpm at this voltage. The saturation effect is very common in almost all servomotor systems. Other non-linear effect is the dead zone; in which the motor will not start to rotate until the input voltage reaches a specific minimum value, which makes the response of the system slower and requires more controllability. A mathematical type of non-linear effect found in the servomotors is the backlash in the motor gears. Some of the servomotors use internal gears connections in order to improve their torque and speed characteristics, but this improvement comes over the effect in the output speed and position characteristics.

The goal here is to find a smart controller that is capable of eliminating as much as possible from these non-linearities, so that we will have a better controllability of servomotor drives.

Artificial Neural Networks

Artificial Neural Networks or ANN's is a very powerful technique for solving complex dynamic systems. The idea of developing artificial neural networks was started by the early understanding of the human nervous system in the 1800's, later scientists started to have a clearer image of how the nervous system looks like, and later in the 20th century (J.J Hofield, 1982) proposed the first Neuron model. When we talk about neural networks we need to relate their behavior to the actual biological neural system that

exists, which consists of neuron cell, axons and synapses (Kandel E, Schawrtz JH and Jessel TM. 2000). The neurons act as the processing elements, they receive the message though the axons, process it and then resend another message to the next neuron. The message first is received by the synapses, which produces some kind of chemicals called neurotransmitters.

Any neuron cell can then be excited and will act in return to the message received. The message generated from the neuron will be either a decision that has been made or just resending the same message that has been received. The connection of multiple neurons to other neurons throughout the human body will form the biological neural networks or the nervous system (Kandel E, Schawrtz JH and Jessel TM. 2000).

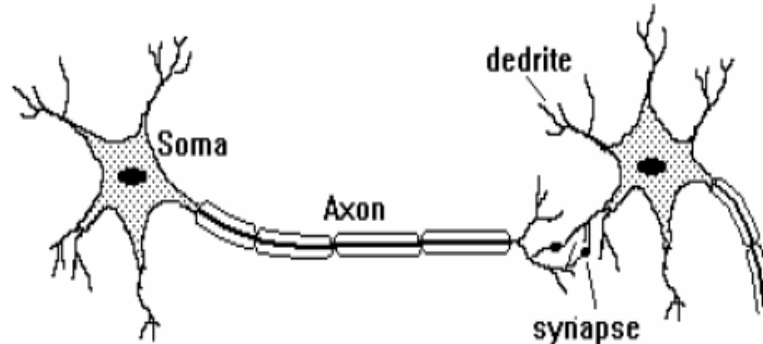


Figure 1.2 . Biological neuron model.

Mathematical Modeling of an ANN

Artificial neurons have almost the same structure as the biological neurons, but with different names and added elements. An ANN mimics the function of a biological neural network and hence it is considered as a very powerful tool that a control engineer can

use to solve difficult non-linear problems. The ANN architecture consists of multiple neurons connected together; each neuron has a similar structure to other neurons (Fukushima Kunihiko, 1975). Figure 1.3 shows a single artificial neuron.

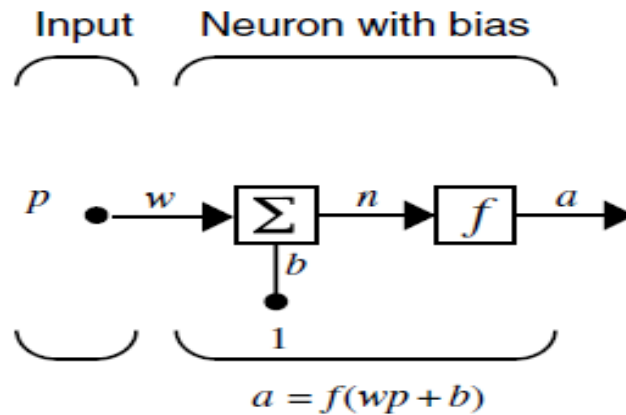


Figure 1.3. Artificial neuron structure.

Each neuron consists of input P , weight w , bias b , transfer function f and the output a . The output of the neuron a can be written in terms of the other elements as (Hagan, 1996):

$$a = f(Wp + b) \quad \text{(Equation 1.1)}$$

The network input can be written as:

$$n = wp + b \quad \text{(Equation 1.2)}$$

f is the transfer function used by the neuron to process the input with the weight and bias to produce the output. There are different types transfer functions used with the ANN's, we will describe them in more details in chapter 3.2. The network input can be a single element, a vector or a matrix consisting of many input values; each one of these

input values has a weight value associated with it. In case of a multi input neuron the weight element will be a weight matrix, and the input will be in vector form.

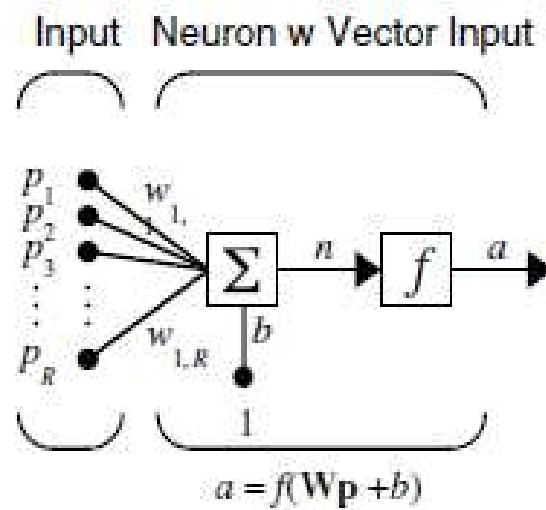


Figure 1.4. Multi input neuron.

The interconnections of single neurons to other neurons form the architecture of the neural network. The interconnection of multiple single neurons to form an ANN is shown in Figure 1.5.

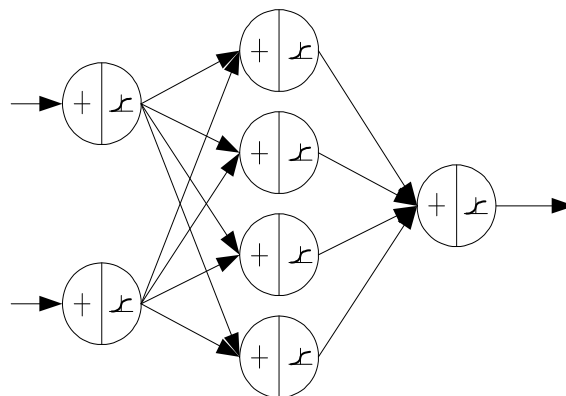


Figure 1.5. Neural network structure.

As mentioned before, ANN's are very powerful tools to solve complex problems and therefore complex control systems. They have a parallel structure that makes them capable to accept large data and process it at one time; which will reduce the time needed for the whole control process to be done. One other important advantage of ANN's is that they are capable to "generalize", which makes them act as a smart controller, this means that the controller is able to deal with new type of input data, this means that the controller has not been trained with that type of data and is let by itself to decide how to deal with any change in the non-linear parameters to reduce their effect. Although, many controllers have been implemented for servomotors, they are not efficient enough in some cases, especially when the servomotor used in a critical application requires a high efficient motor.

CHAPTER 2

REVIEW OF RELATED LITERATURE

DC Servomotor Literature Review

DC motors are widely used in many applications that we use in our daily life. We can find them everywhere, from house appliances to our vehicles, desktops and laptops, and industrial applications such as production lines, remote control airplanes, automatic navigation systems and many other applications. DC motors are well known for their torque-speed characteristics, and their wide operation voltage and current range (David G. Alciatore and Michael B. Hstand, 2007). DC motors can be specified into different types: Permanent magnet motors, Shunt motors, Series motors and Compound motors. For these DC motor types, each one of them has different speed-torque characteristics and different categories of motors. DC servomotors are permanent magnet motors, in which speed and position are typically the most common parameters to control.

DC Servo motors are DC motors that are modified to work using a closed loop control system in which the shaft position or angular velocity are the control variables. A digital or analog controller can be used to direct the operation of the servo motor by sending position or velocity command signals to the amplifier which drives the motor. An integral feedback device (resolver) or devices (encoder or tachometer) are either incorporated within the servo motor or are remotely mounted, often on the load itself. These devices provide the servo motor's position and velocity feedback to the controller, which in turn

use these data to compare them with a programmed motion profile and use them to alter the velocity signal.

A servomechanism, or servo, is an automatic device that uses error sensing negative feedback to correct the performance of a mechanism. The term is correctly applied only to systems where the feedback or error correction signal helps to control a specific parameter of the mechanism. Due to their useful function, servo mechanisms were used long time ago, the Greek were the first to use servo mechanism in their windmills, and they have used this mechanism in adjusting the heading of the windmill (Edward L. Owen, 2002). In 1868 Farcot in his work on hydraulic steam engines for ship steering used the term “Le-Servomoteur” for the first time. A few years later H. Calendar developed his first electro servo mechanism and in 1911 Henry Hobart defined the term “servo-motor” in his electrical engineering dictionary (Otto Mayr, 1970).

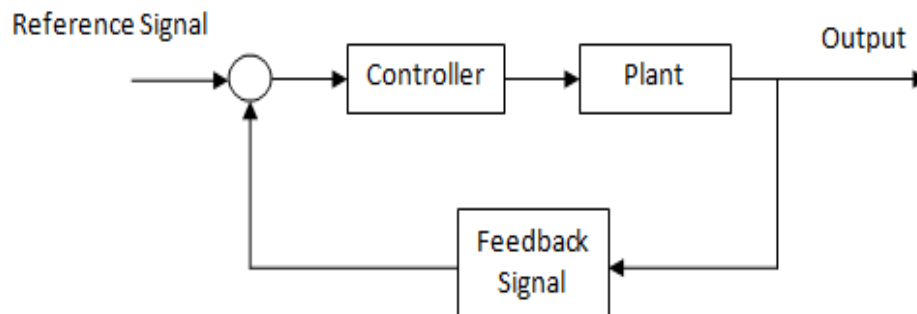


Figure 2.1. Servo mechanism basic structure.

DC servomotors are Permanent magnet motors (PM) in which the stator field is generated by the effect of a permanent magnet. When the PM motor has a position and/or speed control it is called servomotor (David G. Alciatore and Michael B. Hstand,

2007). For this speed and position control advantage of the servomotor, the use of the negative loop feedback can minimize the output error to minimum values.

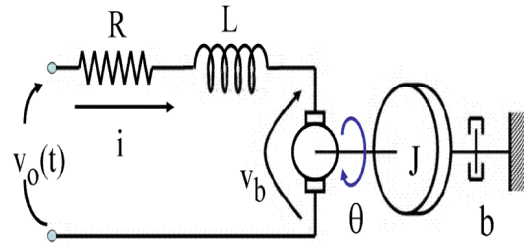


Figure 2.2. Servomotor circuit diagram.

With the use of servomotors, the development of power amplifiers has become also an important component in servomotors; to use them in controlling and powering. Solid state amplifiers are the most used type of amplifiers to power up and control servomotors.

Artificial Neural Network Literature Review

At the beginning of 1800's scientists started to discover the nervous system in the human body, their work on knowing the structure and function of the nervous system continued until 1906 when they started to understand the basic operation of the neuron, and had a clear overview of how it operates and how the basic interconnection of neurons in the nervous systems looks like (J.J Hofield, 1982). After this great discover, McCulloch and Pitts in 1943 came up with the concept of artificial neural networks, when they used it in their primitive artificial neural network (McCulloch, & Pitts, 1943).

The first practical neural network was built in the late 1950's at Cornell University, a neurobiologist called Frank Rosenblatt built it, to be the first one to deal with practical a practical application of a neural network (K. Warwick, G.W Irwin and K.J Hunt, 1992).

From that time neural networks had gained big attention from scientist in different fields. The term Artificial Intelligence (AI) has a strong relationship with neural networks, as the neural network functions and abilities are similar to the nervous system; which in turn has the same function as the human brain. The use of neural network in AI applications was started when John Hopfield first used the ANN in AI applications (W.T. Miller, R.S. Sutton, and P. J. Werbos, 1990).

Since then ANN's started to be one of the most powerful techniques a control engineer may use, many types of neural networks have been invented and used, such as back-propagation and recurrent neural networks.

The basic structure of an artificial neuron is composed of three parts, inputs, weights and outputs, each one of these parts has a unique purpose compared to the other parts. The input layer has to accept the input data. The second components are the weights that are used to modify the values of inputs according to how important they are to produce the corresponding output of the neuron. In this layer the input is propagated to the output through the neuron, each input value is multiplied by the weight value and added to the bias. The output is calculated by applying a particular transfer function to the modified inputs. The overall process performed by the neuron can be defined by the following equation (Martin T. Hagan):

$$a = f(Wp + b) \quad \text{(Equation 2.1)}$$

Where W represents the weights values, p represents the inputs, b the biases and f the function to apply. Then the transfer function calculates the value and sends it to the

output of the neuron. The function of the output layer is to send this final value to a specific location that is defined from the user.

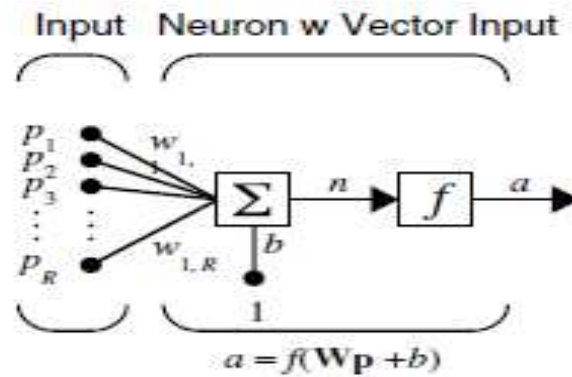


Figure 2.3. Structure of single neuron network.

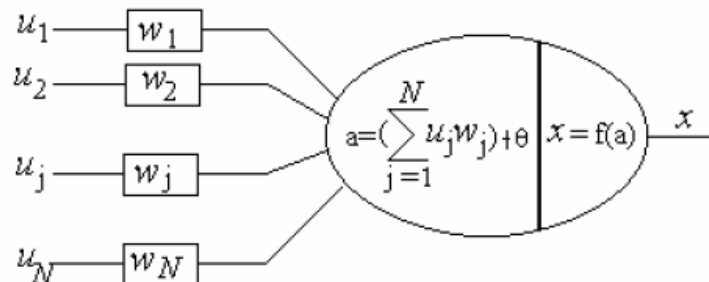


Figure 2.4. Single neuron structure.

An Artificial Neural Network (ANN) is the interconnection of several neurons that are used to solve more complex problems. The basic structure of an ANN is composed of three layers, namely: input layer, hidden layer and output layer, each one of these layers has a unique function. The input layer has to accept the input data; which are the training set of data, or the set of data will be used to simulate the network. The hidden layers, there may be several layers, are used to modify the inputs and define the

interconnections of the neurons; and the final layer is the output layer that produces the single output of the ANN. The basic structure of the ANN is shown in Figure 2.5.

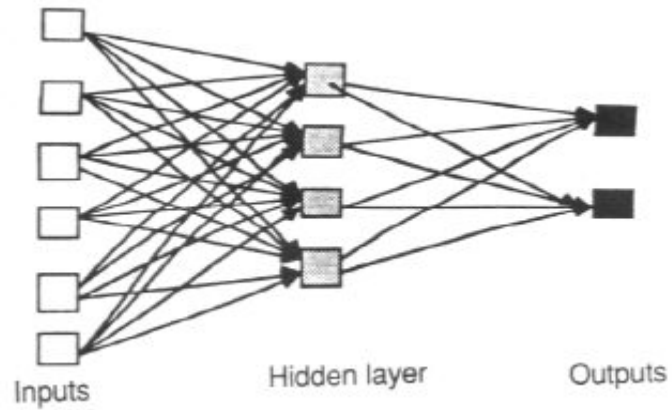


Figure 2.5. Basic ANN structure.

CHAPTER 3

METHODOLOGY

Multi Layer Neural Network Model

Practical neural networks have a structure that traditionally is composed of multi layers. The most common ANN implementation consist of a three layer network: The input layer which accepts the training or simulation data; the hidden layer, which is used to process the input data and modify the weights values; and the output layer, which sends the processed data to a display, control system or any other data storage device. It has been demonstrated that any two-layer network that has a sigmoid transfer function in the first hidden layer and a linear function in the second layer can be used to find the solution to most of the practical applications (Martin T. Hagan, 1996).

Hidden layers are the layers of neurons that exist between the input data and the output layer, these hidden layers have the same structure as any other layer, and each one of them has its own weight matrix, bias and transfer function. Adding more hidden layer to the network makes it more powerful and able to solve more complex problem but it adds more complexity to the controller design, and hence requires more processing time. The control engineer have the tasks to design the network characteristics, and also to think about the complexity and efficiency of the controller, taking into consideration the criticality of the application he is working on. So it is a compromise between the efficiency, time to process the data, cost and complexity of the controller.

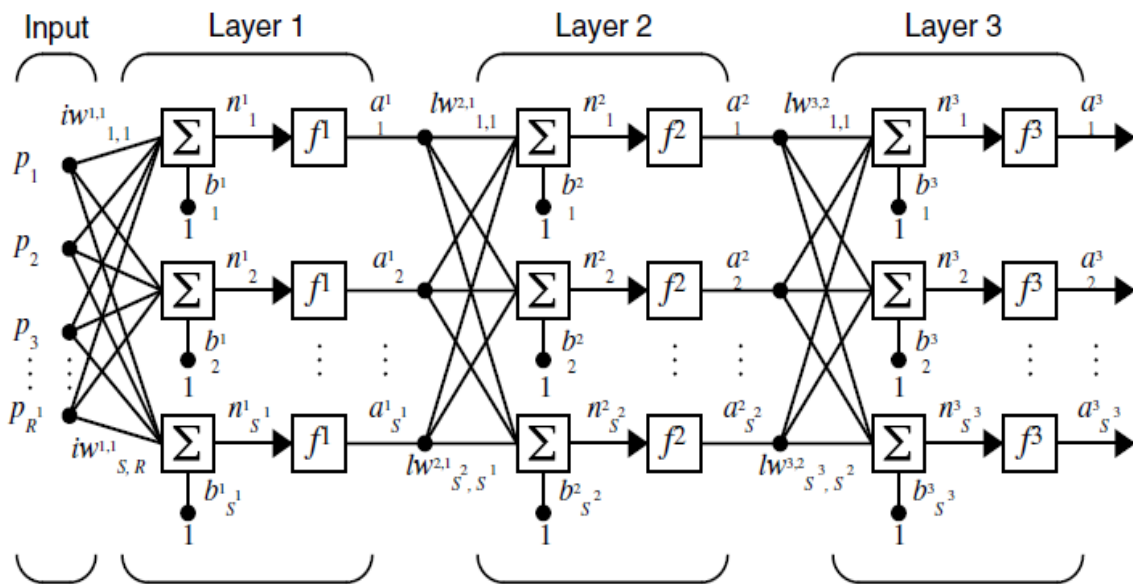


Figure 3.1. Three layer neural network.

Each one of the network layers has an input, weight matrix, bias, transfer function and output. The notation used to specify the output of each different layer is \mathbf{a} with a small number as a subscript to show the number of the layer. For example in Figure 3.1 the output from the first layer is \mathbf{a}^1 . The letter \mathbf{s} is used to specify the number of neurons and \mathbf{R} is the number of inputs. This notation applies to all layers (Howard Demuth and Mark Beale, 2004). The complete output of the neural network can be found using the following equation:

$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3 \mathbf{f}^2(\mathbf{W}^2 \mathbf{f}^1(\mathbf{W}^1 \mathbf{p} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3) \quad \text{(Equation 3.1)}$$

In this equation, \mathbf{a} is the output, \mathbf{f} is the transfer function used to process the values, \mathbf{W} is the weight value and \mathbf{b} is the bias. This equation can be extended to find the output for more layers. There is no rule or law to show the number of neurons in each layer and the number of layers in each network that is best for a specific application, the only

way is to do that by trial and error until we reach the optimal network design for an application.

Neural Network Transfer Functions

The transfer function is one of the main components of an ANN. There are different types of transfer functions used in ANN's, each one of these functions will have a different output from the others. The most common transfer functions used with ANN's are the Hard-Limit, Linear and Log-Sigmoid transfer functions. The Hard-Limit transfer function takes the input data n process it and gives one of two output values, if the value of n is less than zero, then the output is zero, and if the input data more than zero, then the output is one.

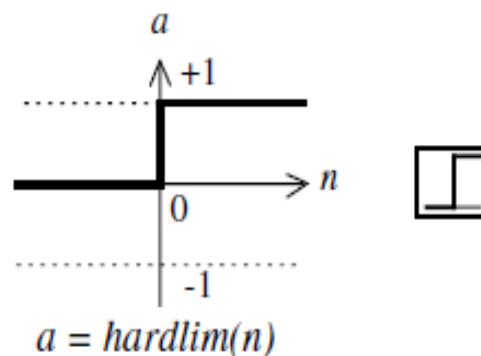


Figure 3.2. Hard-Limit transfer function

The Linear transfer function has an output value that has a linear behavior in respect to the input, regardless of the sign of the input value; this function is very powerful to be used with linear approximation functions. The linear transfer function characteristics are shown in Figure 3.3.

The Log-Sigmoid function (shown in Figure 3.4) takes the input data which may have any value and generates an output in the range of 0 – 1 (or -1, 1), this transfer function is one of the most used in backpropagation neural networks, since it's an infinite differentiable function.

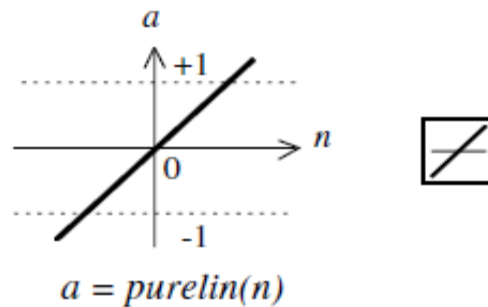


Figure 3.3. The Linear transfer function

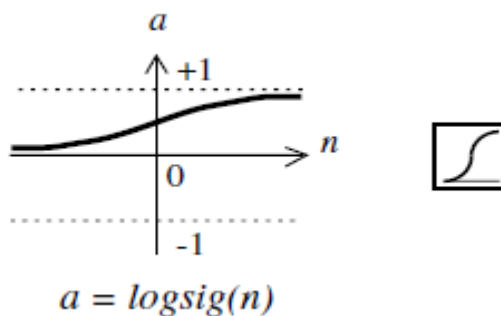











Figure 3.4. The Log-Sigmoid transfer function.

Other transfer functions with different capabilities are used with different neural networks. Table 3.1 shows more transfer functions used with ANN's. The selection of each layer's transfer function is very critical in the network's performance and may lead to both a successful network training and simulation or to fail both. As there is no predefined method to find the number of neurons and layers for each network, also there is no

predefined method to choose the transfer function in each layer. The best match of transfer function for any application comes from initial guess for the functions, then by testing different functions to get the best network performance.

Table 3.1. List of Neural Network Transfer Functions.

Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1 \quad \text{neuron with max } n$ $a = 0 \quad \text{all other neurons}$		compet

Feed Forward ANN and Backpropagation Training Algorithm

A feed forward network, as the one shown in Figure 3.1, is a network in which the data moves only in one forward direction from the input layer to the hidden to the output layer, with no internal cycles or loops inside the network (Simon Haykin, 1998). Backpropagation is a method used in the training process of ANN's. Backpropagation can be described as supervised learning; in which we have an input/output data pairs used to train the network. In Backpropagation each input data has a corresponding output data relates to it. After the training step using backpropagation the network should be able to find a correct output value to a new input value that it has not been trained for it before, so the network act as a classifier.

Backpropagation method works to update the values of the weights and biases in the same direction as the performance function decreases more rapidly. The equations that shows how backpropagation works is:

$$X_{k+1} = X_k + \alpha_k g_k \quad \text{(Equation 3.2)}$$

Where X_k is the vector of current weights values, α_k is the learning rate and g_k is the current gradient.

The way of updating the weights relates to the gradient decent algorithm, in which we have two different ways to implement it. The first way called incremental training; in which the weights and biases are updated after a new input sent to the network. The second way called batch mode training, in which all the inputs are applied to the network before changing the weights and biases values. Batch mode training can be used if the all the input data is ready to be sent to the network, the gradients are

calculated at each training data and then added together to determine the value of the weights and biases, which is the case of our data.

PID Control

The Proportional, Integral, Derivative controller (or the PID controller) is the most popular type of controller used in different engineering applications. The PID controller is a form of control loop that has a feedback mechanism. The PID controller works by calculating the error signal between an output measured value and a reference value, the controller works to minimize the error signal or the difference between the output signal and the reference signal to a minimum value; such that the output measured value will be as close as possible to the input reference signal (Robert N. Baterson, 1999).

The mathematical representation of the PID controller is:

$$U(t) = K_p \cdot e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \quad \text{(Equation 3.3)}$$

Where $U(t)$ is the controller output signal, $e(t)$ is the error signal, K_p is the proportional gain, K_i is the integral gain and K_d is the derivative gain.

As shown in Eq. 3.3 above, the PID controller has three parameters, *P or Proportional term*, *I or Integral term* and *D or Derivative term*, each one of these terms has a gain value related to it, and it makes the controller system to react in a different way from the others. The proportional term depends on the present error value, the proportional gain have a direct relationship to the controller sensitivity, the higher *P gain value leads to faster change for the systems' output*, which makes the controller to be more sensitive.

A pure proportional controller will have a steady state error and depending on the gain it could generate an overshoot in the output signal. Figure 3.5 shows the response of a system when a proportional term and D term is applied, while keeping the integral and derivative values constant.

The integral term depends on summation over time of the present and the previous errors. The action of the integral term makes the system to reduce its steady state error in the output and to have a smoother slope that reached the final value faster. Figure 3.6 shows the integral effect on the output signal.

The last term of the PID controller is the derivative term, which depends on the rate of change of the error. Both the value of the current error and the duration of the error are taken into account, it speeds up the controller response, but can make the overshoot in the system higher.

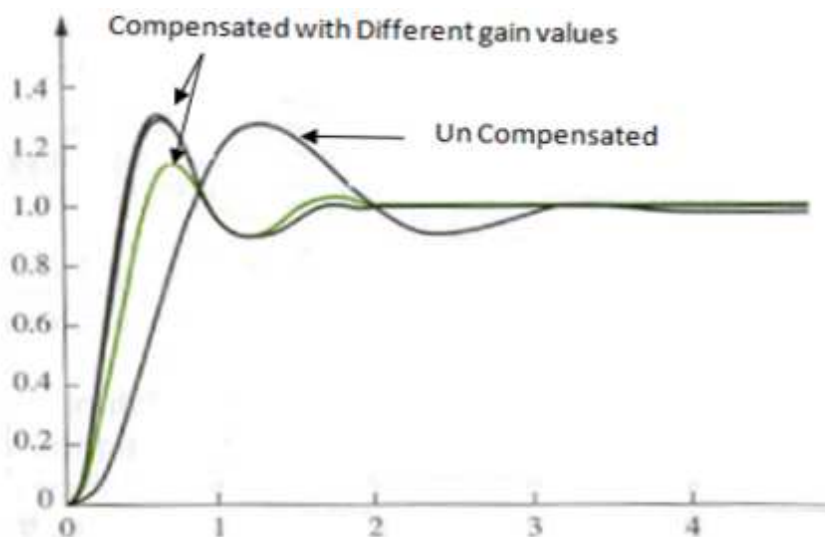


Figure 3.5. Proportional and Derivative terms effect

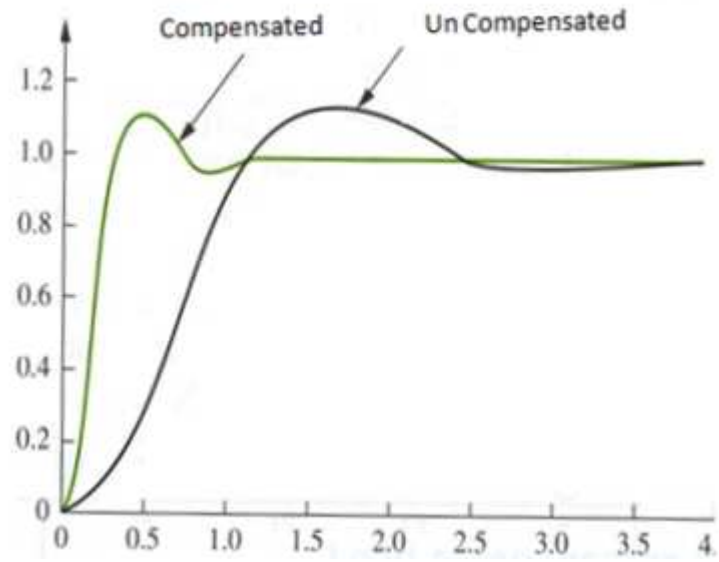


Figure 3.6. Integral term effect.

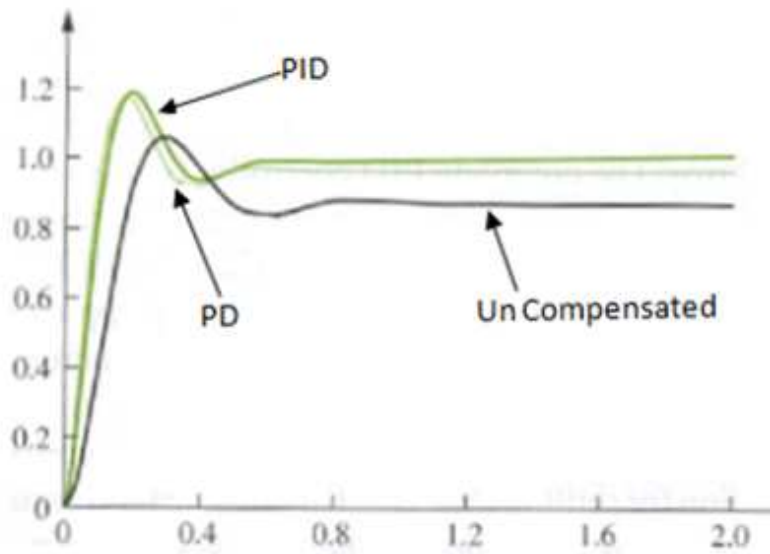


Figure 3.7 Derivative term effect.

By combining the three terms and their effect together we can get the PID controller function. PID controllers can be found everywhere now, in many different industrial process and applications. Even though they are widely used and well developed, but they still not the perfect choice for some critical time and high efficient applications.

CHAPTER 4

DC SERVOMOTOR

DC Servomotor Model

Recalling the DC servomotor diagram from Figure 1.1, the transfer function of the DC servomotor can be derived using Kirchhoff's voltage law and Laplace transforms (Nise, 2008) as the following:

$$V_o = V_m = R_m i_m + L_m \frac{di}{dt} + V_b \quad \text{(Equation 4.1)}$$

The Back-electromotive force (emf) V_b can be found by using the equation shown below.

$$V_b = K_m \frac{d\theta}{dt} = K_m \omega_m \quad \text{(Equation 4.2)}$$

Where V_b is the induced voltage, K_m is the motor torque constant, and ω_m is the angular rotating speed. It can be seen that ω_m can be calculated by the Equation shown below

$$\omega_m = \dot{\theta} \quad \text{(Equation 4.3)}$$

And Using Laplace Transform

$$\omega(s) = s\theta(s) \quad \text{(Equation 4.4)}$$

Our concern in this stage is to control the angular rotating speed ω_m , by controlling the input voltage V_m .

Where:

- J = moment of inertia of the rotor
- b = dampening ratio of the mechanical system
- T = motor torque
- I = Current

V_m	=	back emf
θ	=	shaft position
K	=	electromotive force constant
ω_m	=	Measured angular Speed
R	=	Motor Armature Resistance
L	=	Inductance
V	=	Source Voltage

The transfer function of the output angular speed is derived using Laplace transform using the second order system equation:

$$G(s) = \frac{\omega_n}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad \text{(Equation 4.4)}$$

The resulting transfer function:

$$\frac{\omega}{V} = \frac{K}{(Js + b).(Ls + R) + K^2} \quad \text{(Equation 4.5)}$$

From Equation 4.3, the relationship between the angular position and the speed can be found by multiplying the angular position by $\frac{1}{s}$. Our major concern on this research is the proper control of the angular speed of the motor; since the angular speed is the part that suffers the most from the non-linearities. The angular non-linear effect on the angular position tends to be less due to the term used to derive it $\frac{1}{s}$, which adds an integral effect or filter effect to this part. Figure 4.1 shows the block diagram which represents the servomotor system using MATLAB SIMULINK.

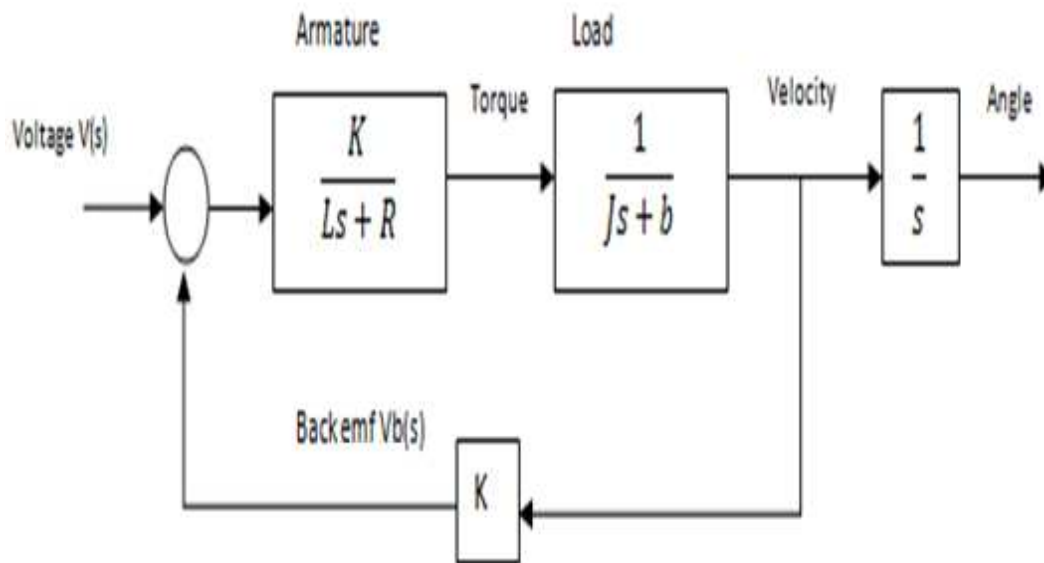


Figure 4.1. Servomotor SIMULINK block diagram.

DC Servomotor Response

The response of the servomotor, as mentioned before in section 4.1, can be considered as a second order system. A second order system will have a natural frequency ω , a damping factor ζ . The general response of a second order system with a step input is shown in Figure 4.6. From the response of the second order system we can get some of the characteristics of the system, and the design criteria can be implemented using these characteristics.

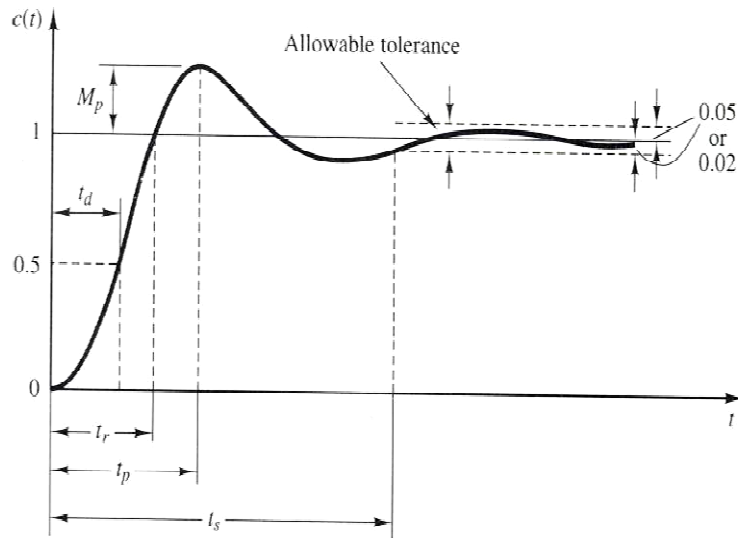


Figure 4.2. Second order system response

Different parameters can be used to evaluate the response of the servomotor; by adjusting the value of these parameters we can reach our design goal. M_p is the overshoot value, t_s is the settling time and β is the allowable error tolerance. These three parameters can define the design criteria and output response for any second order system response.

The ideal system response will have a zero overshoot, zero settling time and zero tolerance, but in real life achieving the ideal response will be hard to achieve and will have a high cost of implementation. So the solution can be found by defining an accepted range for the values of the three parameters mentioned before to reach a good system response for a specific application.

CHAPTER 5

ANN MODELS STRUCTURES

In this research we are proposing a neural network controller design to control the DC servomotor. The training algorithm used in the ANN is the back-propagation method. Two feed-forward neural networks are used, the first neural network is called the Model Network; the function of this network is the same function as the DC servomotor. The second network used is called the PID neural network controller, this network has the same function as a PID tuned controller, but the difference is that this network is capable of updating itself in a manner to improve the controller function this is why it is considered to be a smart controller.

Servomotor Neural Network Model:

In this part we have built a neural network that has same function as the servomotor, input/output data pairs used to train this network and simulate it. The first stage was to build SIMULINK model that represents the servomotor system.

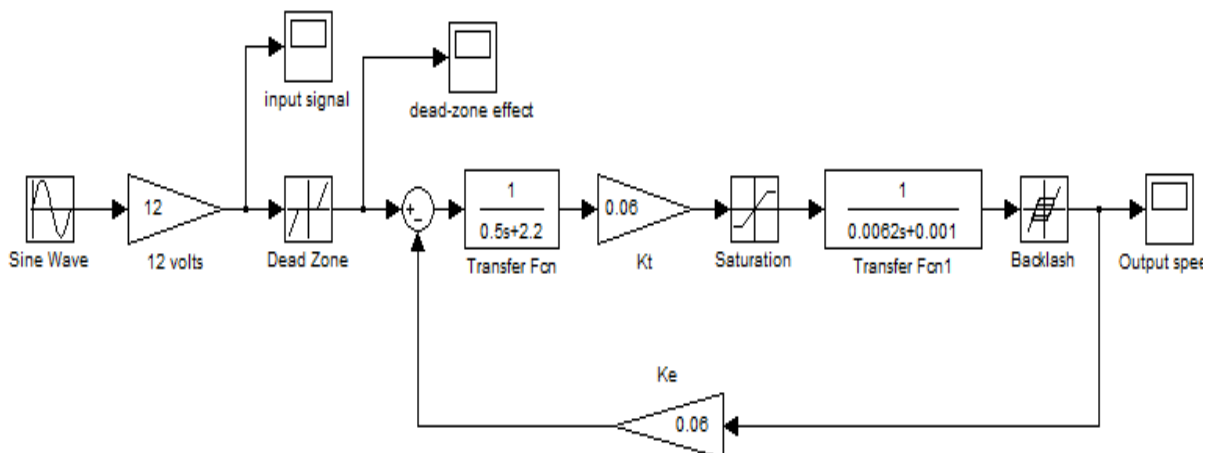


Figure 5.1. Servomotor SIMULINK model

This model has been derived from the Servomotor speed transfer function, Equation

$$3.7: \quad \frac{\omega}{V} = \frac{K}{(Js + b).(Ls + R) + K^2}$$

The parameter values used in the SIMULINK model were taken from a practical servomotor. The parameters are:

Table 5.1 DC Servomotor parameter values.

Parameter	Value
Moment of Inertia J	0.0062 N •m•s ² /rad
Damping Coefficient b	0.001 N•m•s/rad
Torque constant K _t	0.06 N•m/A
Electromotive force constant K _e	0.06 V•s/rad
Electrical Resistance R	2.2 Ohms
Electrical Inductance	0.5 Henry

From the SIMULINK model it is clear that we can divide the transfer function into two transfer functions, the first one is the electrical transfer function; which consists of the electrical resistance and the inductance. The second transfer function is the mechanical transfer function; which consists of the moment of inertia and the damping coefficient. By dividing the transfer function into the two parts mentioned above, we can add the non-linear parameters to the system to see their effect.

The input/output data pairs are generated from this model in order to train the Neural Network model. The data is sent to m files in the form of matrices, stored in a specified location, so we can call them or use their data when needed. The model is simulated for

10 seconds to generate the data. With a predefined sine wave frequency of 6 rad/sec, we can make sure to have enough data points to fully represent the system.

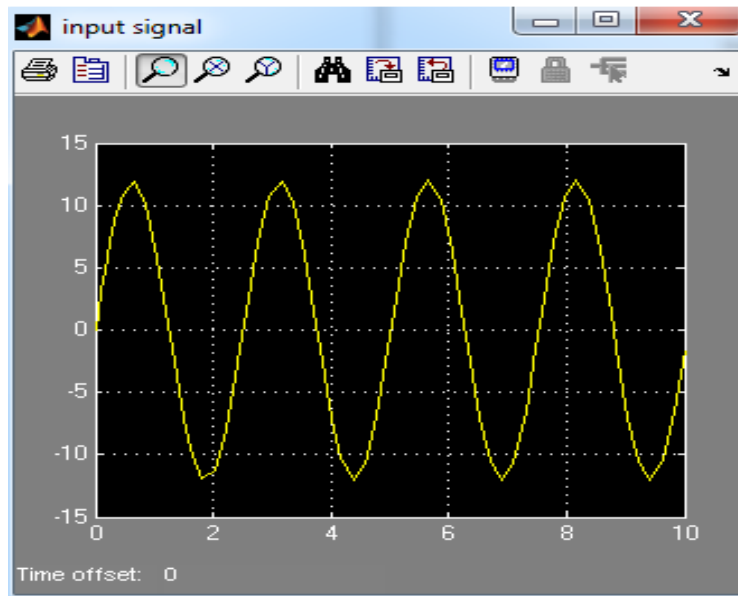


Figure 5.2. Input sine wave input.

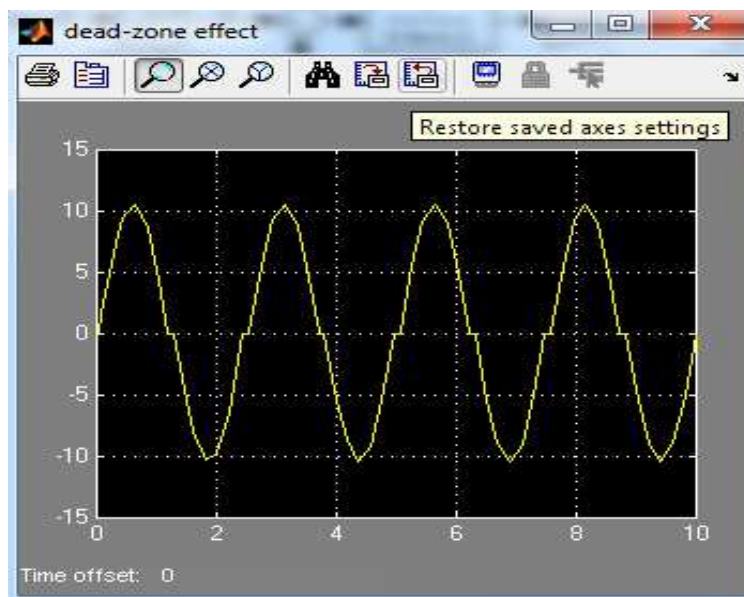


Figure 5.3. Input signal after dead zone effect

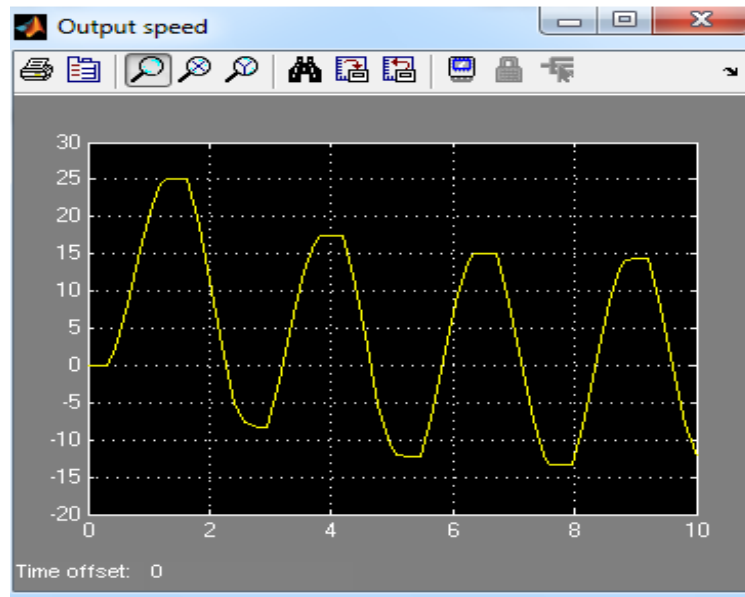


Figure 5.4. Output Speed

In Figures 5.2 to 5.4 we can see the effects of the non-linear parameters affecting the input signal. The first change for the input signal can be seen on Figure 5.3, which represents the signal after the dead zone. The dead zone value is 1.5 volts, which is very common in this size of servomotors. After applying this input signal to the whole system including the other non-linear parameters, we can see the big distortion to the output signal compared to the input signal.

To have a clearer image of the non-linear effect, the systems suffers from, we have used a single step input. The step input has a maximum value of 12 volts, which is the maximum voltage rating of the servomotor we are using. From Figures 5.5 and 5.6 we can see how the dead zone limits the signal to 10.5 volts, which is less than the required voltage, then we can see the slower response of the output signal, which represents the speed signal. This bad response (shown in Figure 5.7) of the system can be worse if one or more of the system's parameters or the outside disturbances are changed.

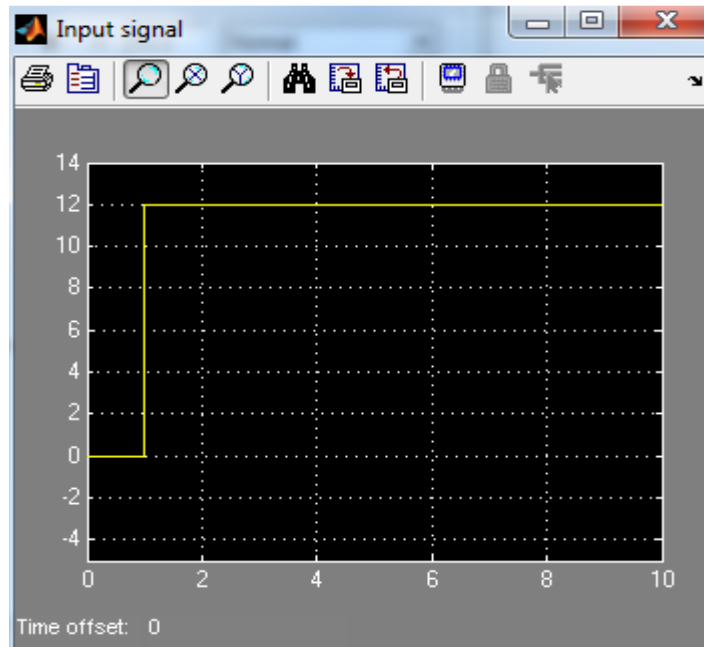


Figure 5.5 Step input before dead zone

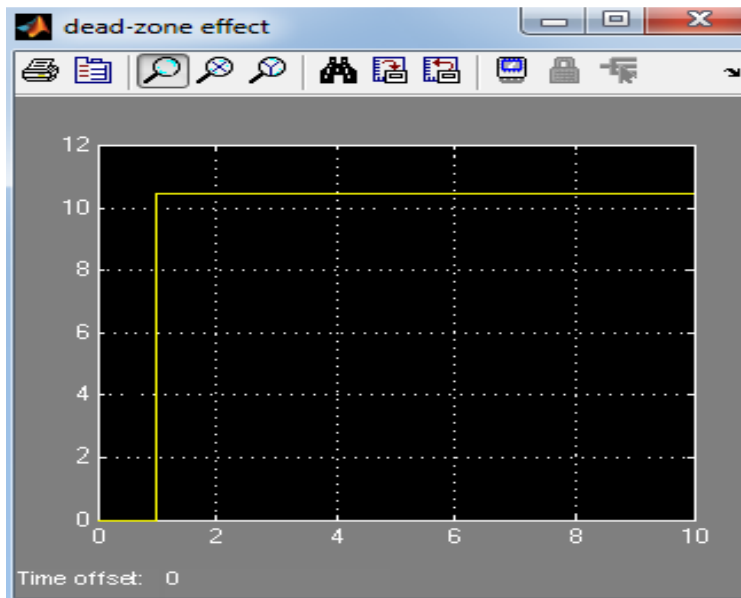


Figure 5.6. Input signal after dead zone.

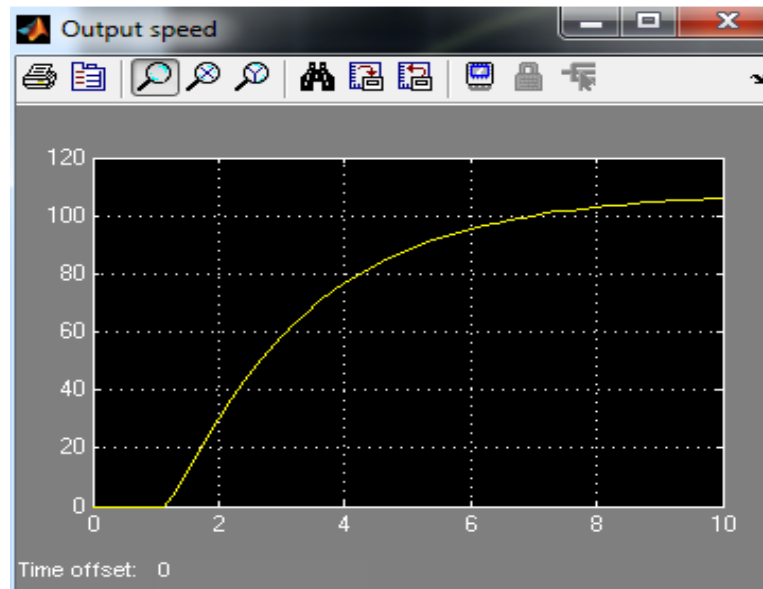


Figure 5.7. Speed output signal.

The neural network model, that mimics the function of the servomotor, has been built using the data recorded from the original sine wave response mentioned earlier. The choice of using the training data from the values obtained is due to nature of the sine wave, since it is the most common wave found in nature, so it has a more general behavior than the other waves. MATLAB SIMULINK and commands have been used to store the data and to generate the neural network model. From the type of data used, sine wave, step input and other signal types, the training data is a two dimensional matrix, one dimension is the time and the other one is the magnitude. The full code used to generate the network is available in Appendix A. The network parameters used to generate the network appear in the MATLAB code bellow:

```
P = sininput;
T = sineoutput;
net = newff(P,T,25);
Y = sim(net,P);
P1 = sininput(1,:);
Figure1 = plot(P1,Y);
net.trainParam.show = 50;
```

```
net.trainParam.epochs = 1000;
net.trainParam.lr = 0.05;
net.trainParam.goal = 1e-5;
net = init(net);
net = train(net,P,T);
Y1 = sim(net,P);
gensim(net,-1)
```

Where P is the input data obtained from the sine wave input signal, T is the target data; which represents the output speed. The maximum number of epochs used is 1000, which is the maximum number of trials the training method can apply before stopping. The training goal was to reach a minimum error value of 0.00005, this error value is between the input signal and the generated output. The learning rate was chosen to be 0.05. These parameters were chosen to best fit the performances of the network, more restrict parameters may take the network to have unstable response and it may cause the training method to diverge. The results from the implemented network are shown bellow (refer to Appendix A for the training data):

- The plot of data before the training of the network:

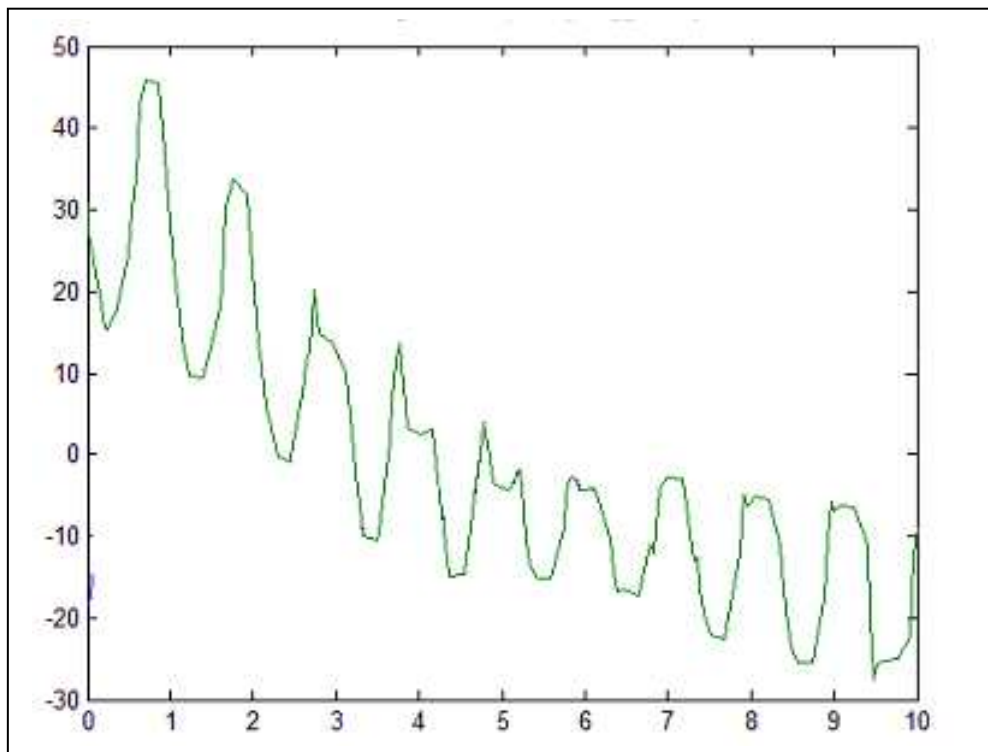


Figure 5.8. Network response before training

- The plot of data after training:

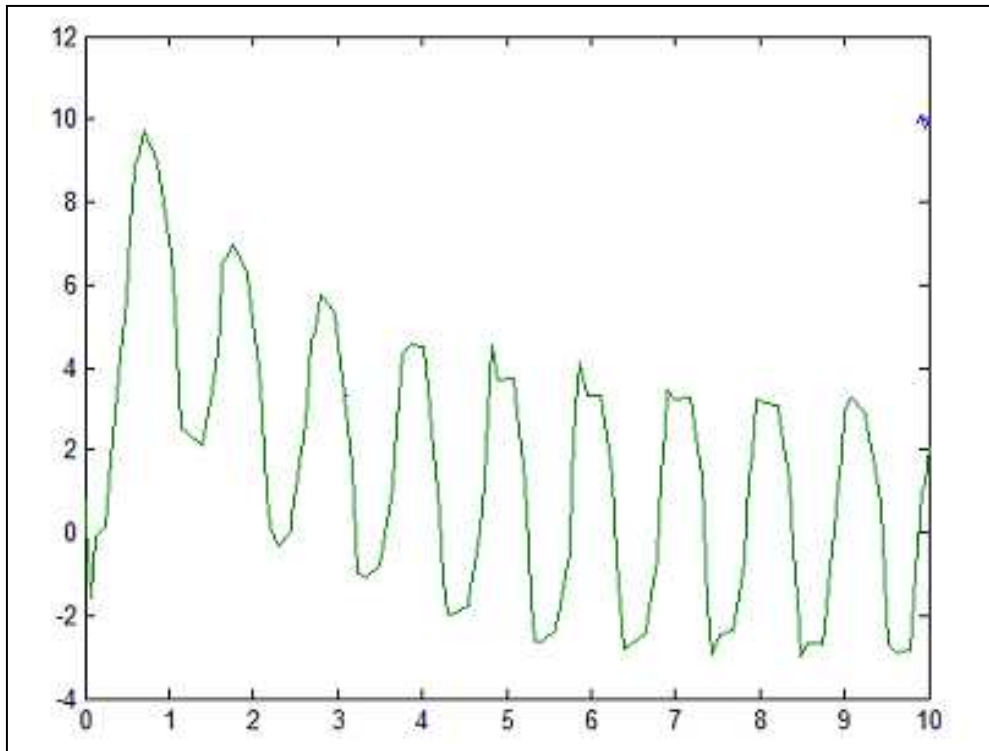


Figure 5.9. Trained Data plot

From Figure 5.9, and comparing to Figure 5.4, we can see how the neural network model has the same response than the original servomotor; from here we can be sure that we can use this network in our system. Figure 5.10 show the performance of the network training, and how the desired error goal has been achieved within the specified number of epochs. Figure 5.12 shows the results of the training, where the number of epochs used to achieve the high performance of the network was 23 epochs, which makes the response of our network model to be very fast compared to any other model that represents the servomotor. The regression plot, shown on Figure 5.11 shows a high data scattering around the regression line, we can see that for $R = 0.98969$

regression line most of the data lies in the range; which is a strong evidence of the successful training.

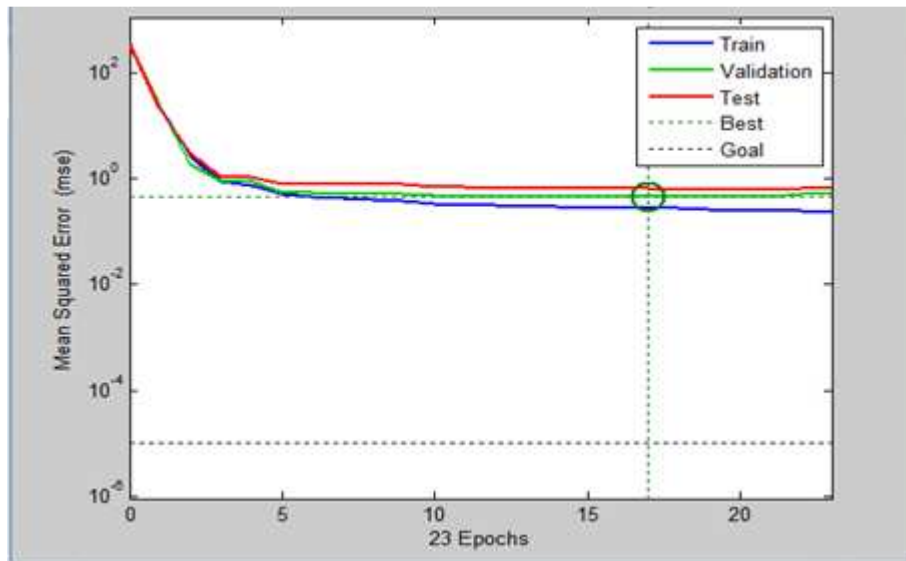


Figure 5.10. Performance Plot.

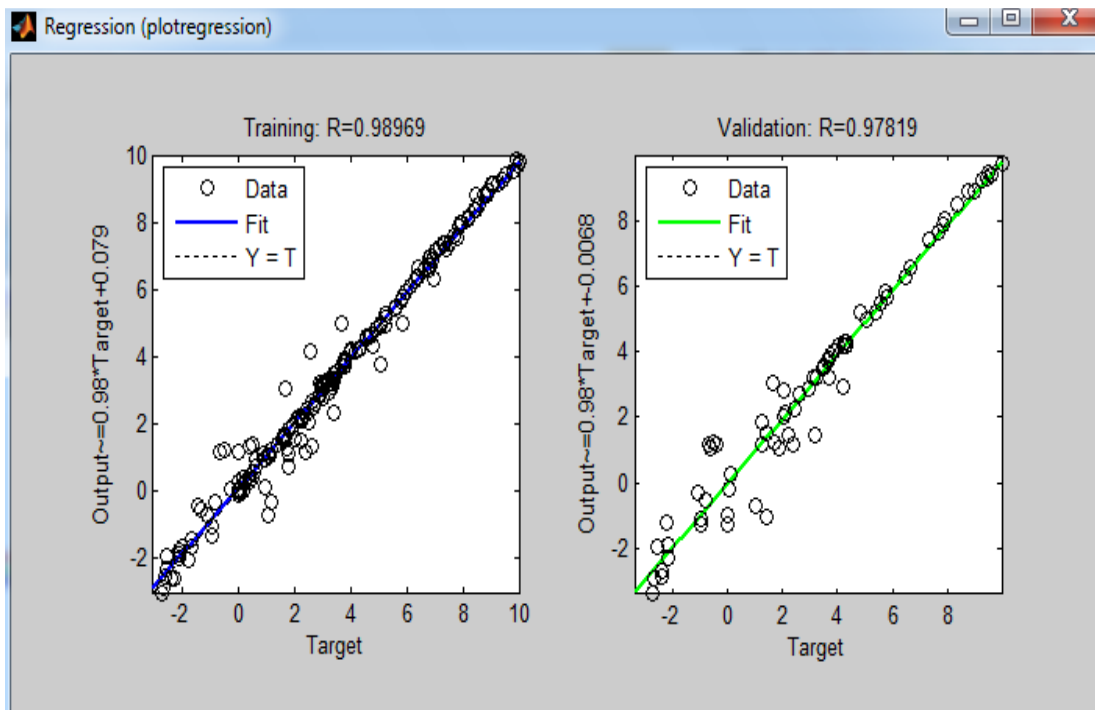


Figure 5.11. Regression plot

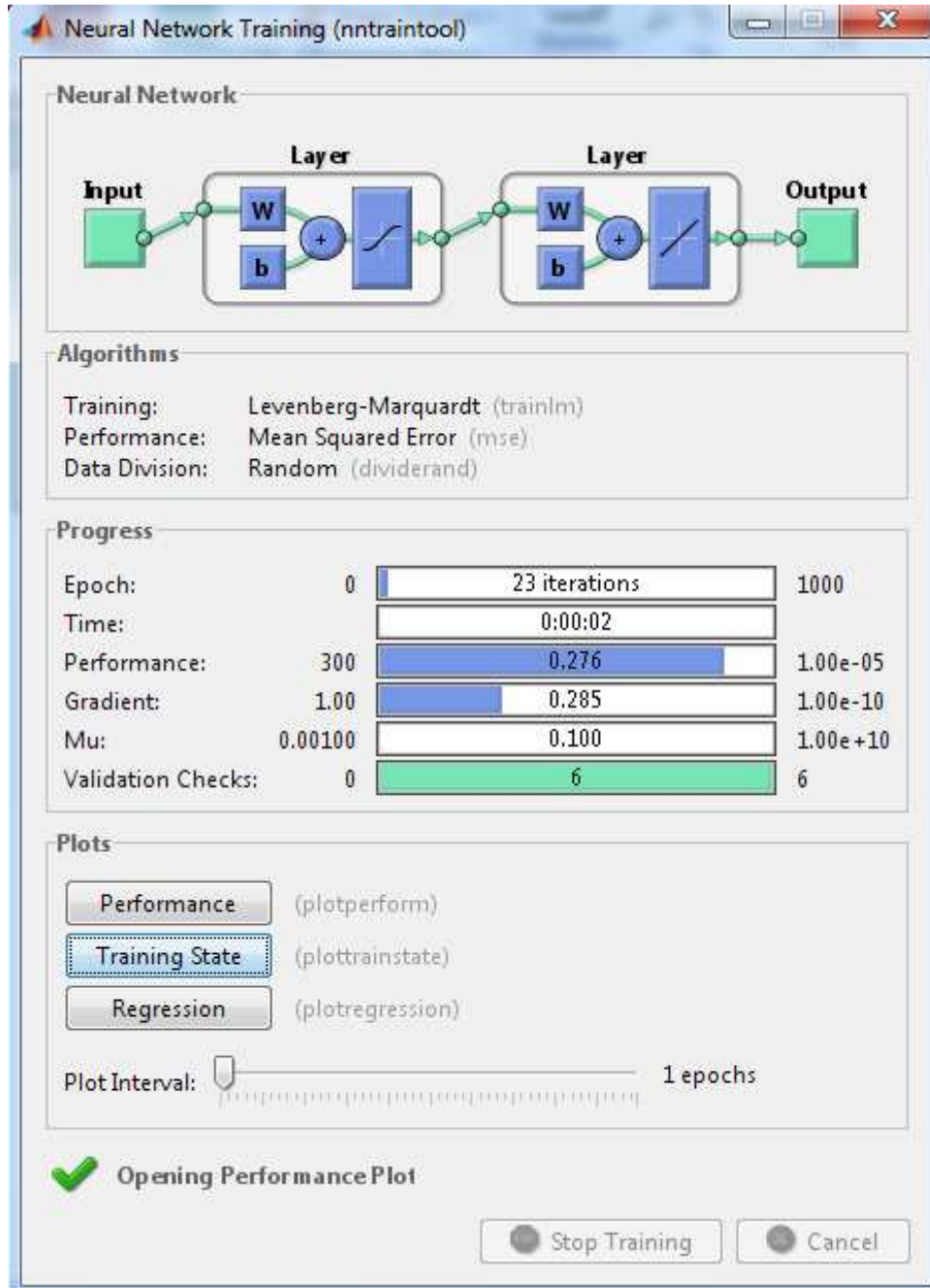


Figure 5.12. Training tool window of the NN motor model

Neural Network Controller Model

The neural network model controller was built based on a tuned PID controller, the first step was to control the servomotor using PID tuning technique, after reaching a better performance of the motor with the PID controller, data have been collected to train the neural network controller. The method of using an embedded PID controller inside the controller function makes the system more powerful, the neural network after training is capable of improving the over performance of the system, the advantage of this smart controller that its able to deal with any new change may occur to the system, and to eliminate the non-linear effect found in the system.

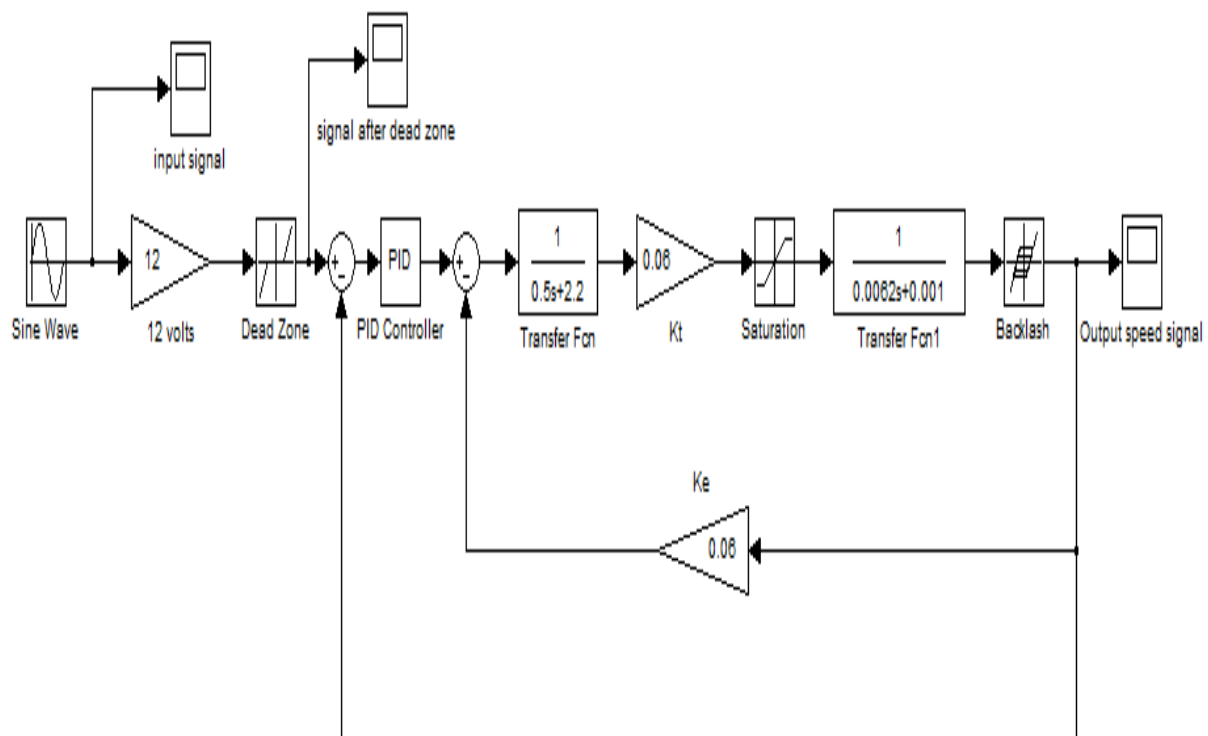


Figure 5.13. PID Controller SIMULINK diagram

The data used for this neural network were obtained from the input of the PID block; which were used as the input training data. The output from the PID block was used as the target data for the training process. The neural network controller has the same structure and training parameters of the servomotor model. It has 25 neurons in the hidden layer, 1000 epochs, 0.05 learning rate and error value of 0.00005. The MATLAB code used for the controller network is:

```
P1 = pidin;
T1 = pidout;
net1 = newff(P1,T1,50);
Ypid = sim(net1,P1);
Plpid = pidin(1,:);
```

```
Figure1pid =
plot(Plpid,Ypid);
```

```
net1.trainParam.show = 50;
net1.trainParam.epochs =
1000;
net1.trainParam.lr = 0.05;
net1.trainParam.goal = 1e-5;
net1 = init(net1);
net1 = train(net1,P1,T1);
Y1pid = sim(net1,P1);
gensim(net1,-1)
Figure4pid =
plot(P1pid,Y1pid);
```

- Simulation Plot of the PID input and output data

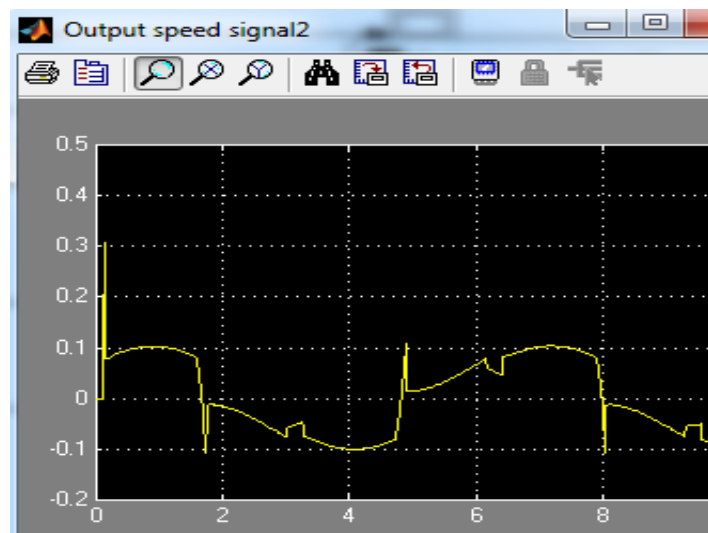


Figure 5.14. PID input data

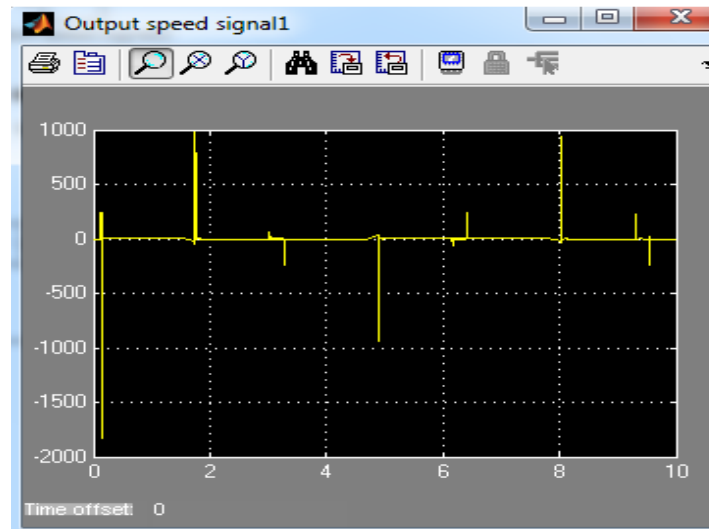


Figure 5.15. PID output plot

- Neural Network Controller plots

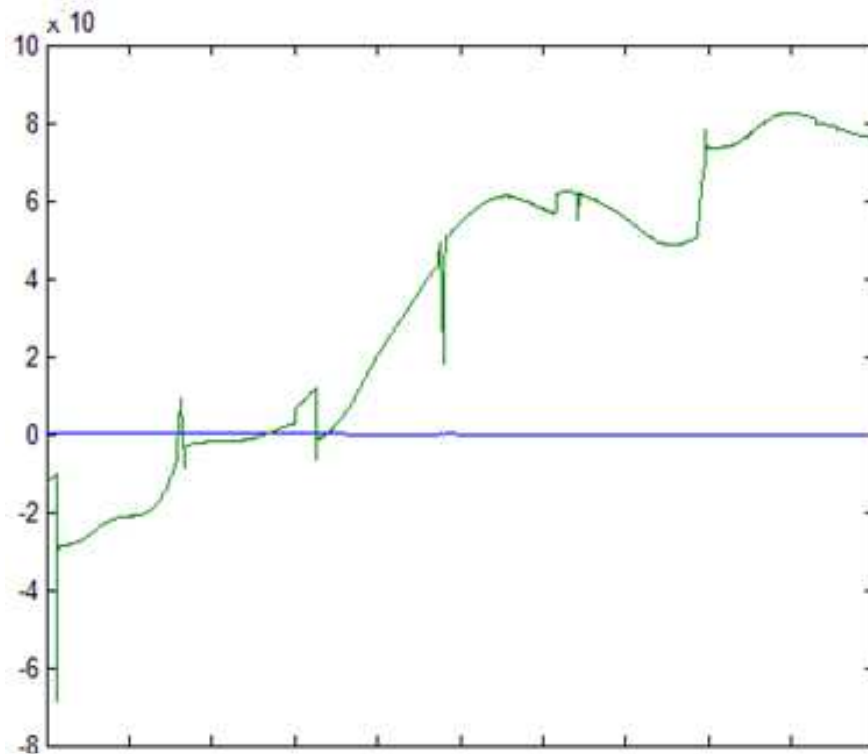


Figure 5.16. NN controller response before training

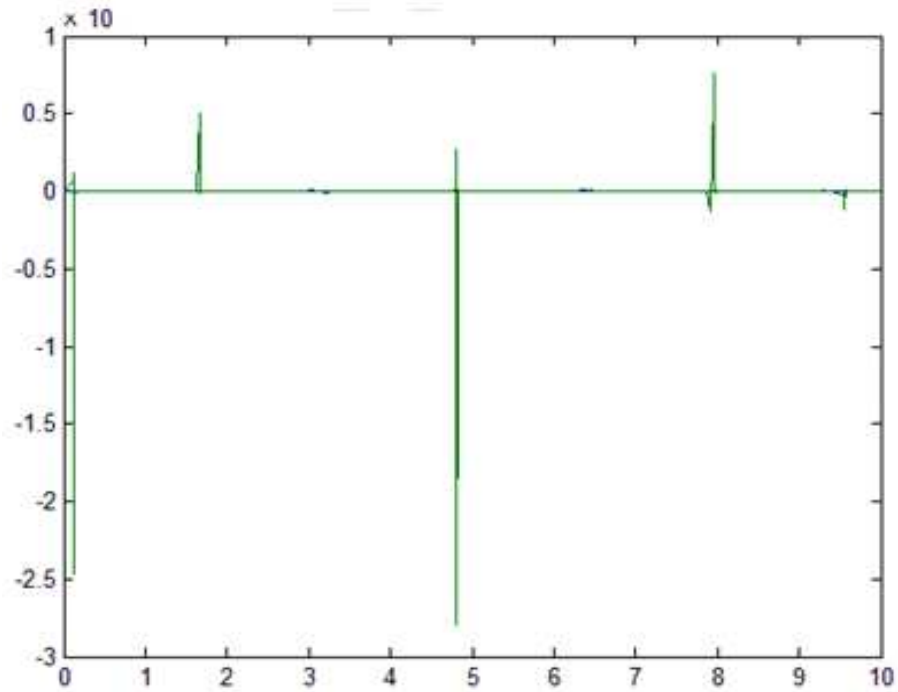


Figure 5.17. NN controller response after training.

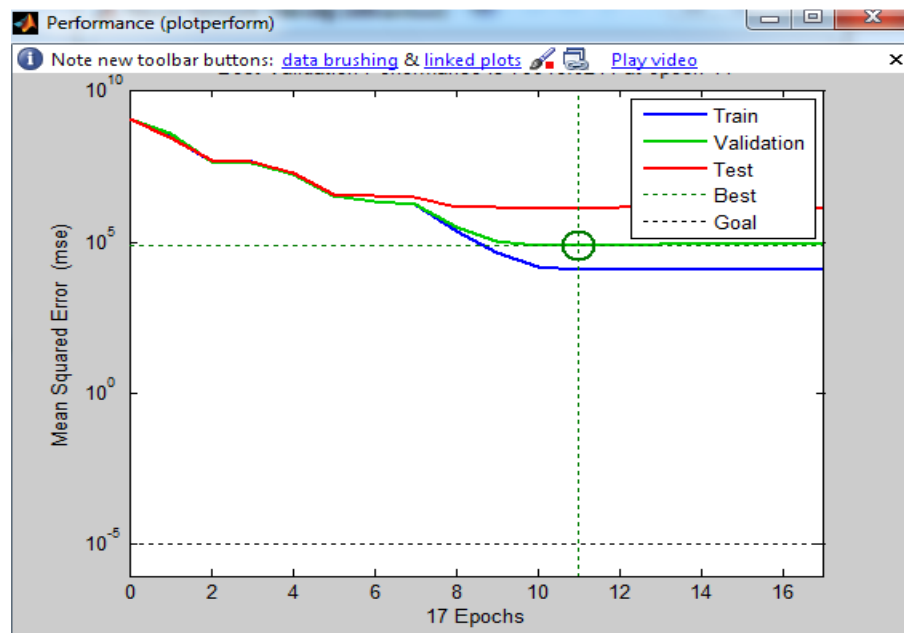


Figure 5.18. Performance plot of the NN controller.

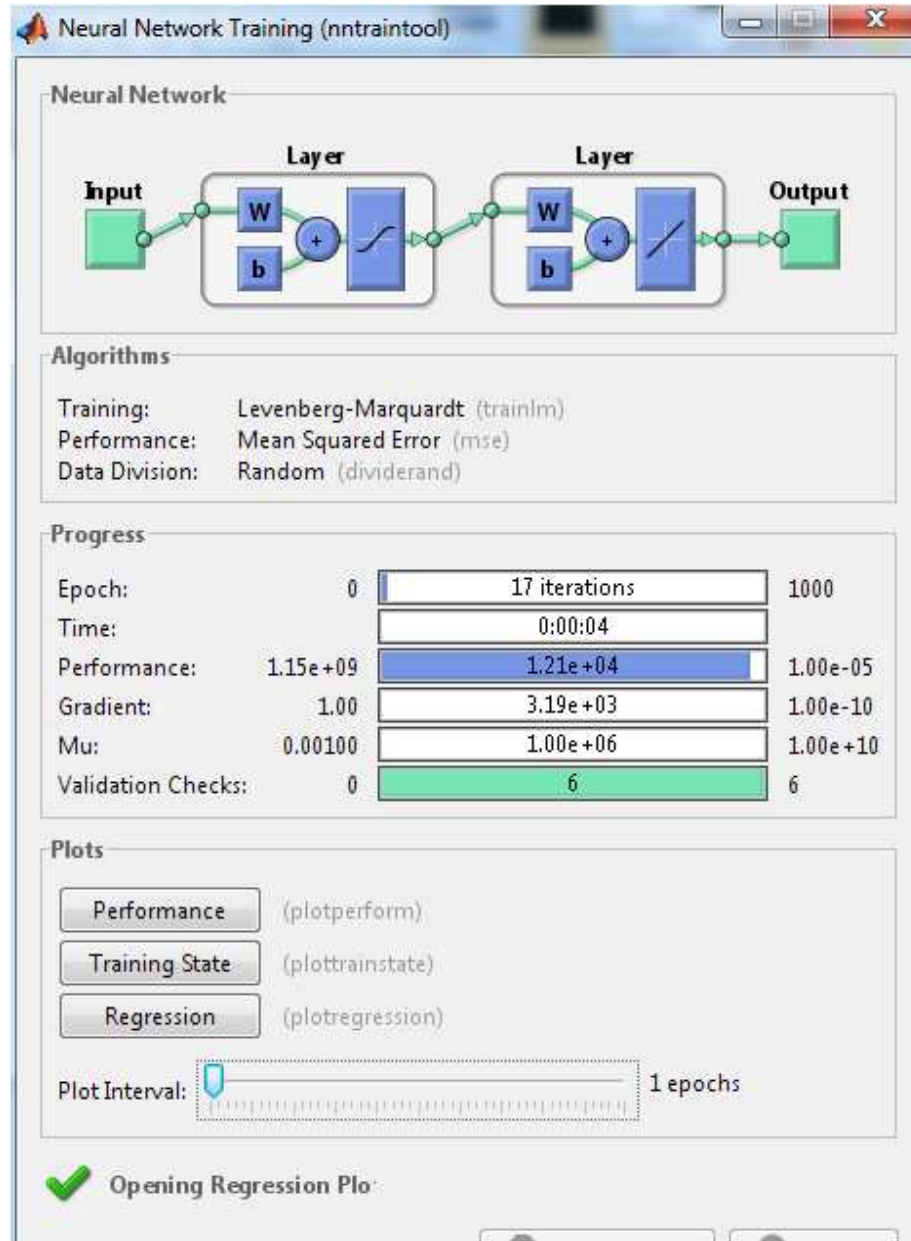


Figure 5.19. Training tool window of the NN controller model.

From Figure 5.19 we can see that training of the controller NN was successful and compatible with the simulation results. Also the performance results were successful,

the training process took 17 epochs to reach the desired error value, which makes the controller response very fast.

CHAPTER 6

CONTROLLER DESIGN AND SIMULATION RESULTS

After we have discussed the different parts of the system and how each neural network succeeded in the training process, we will discuss the system as a whole, when both the NN controller and the NN servomotor model are connected together. In this chapter we will show all the simulation results from the NN controller system, then we will compare the results with the same system but while using a conventional PID controller.

Neural Network System:

After we have used MATLAB commands to build and train our neural networks, we use a very useful command to generate neural network SIMULINK diagram. The function used the ***gensim*** function; this function can be used after successful training of the neural network, the block generated then can be used to build the total system.

The syntax of this function is: $gensim(net, st)$

Where the *net* parameter refers to the trained neural network, and the *st* represents the sampling time, in our case *st* will be set to -1; which means continuous sampling. The two networks can be generated using this function, but the generated network will accept only constant input in its initial configuration. To connect both networks together we will need to add more elements to the SIMULINK model to make sure our networks accept dynamic input. The generated neural network blocks for the motor model before connecting them together is shown in the Figures 6.1 & 6.2.

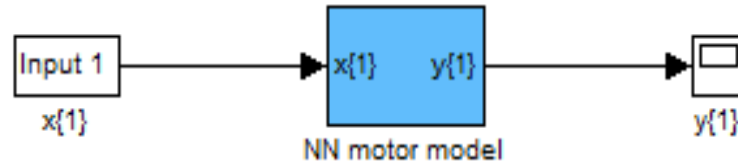


Figure 6.1 NN for motor model

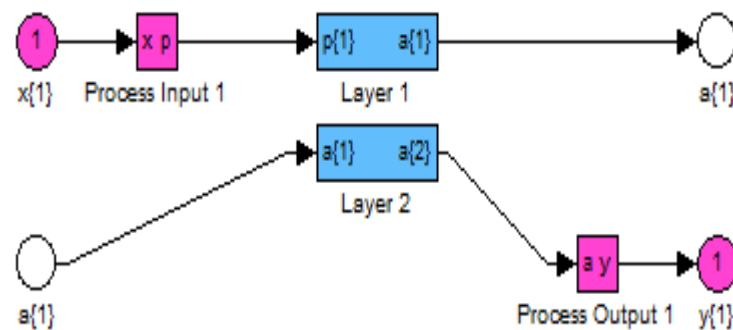


Figure 6.2. Internal NN structure

The SIMULINK model for the controller has the same structure as for the motor model, the only difference that each one has a different function to perform. From the internal structure we can see all the NN components, input, hidden layer and output. From here we will see the simulation results for the NN motor model by itself first, to make sure the network is working correctly then we will build the complete system. In order to simulate this network we need to have a two-dimensional input, this input can be done by using two sine wave signals and a Multiplexor (Mux) block. Figure 6.3 shows the new configuration of the network.

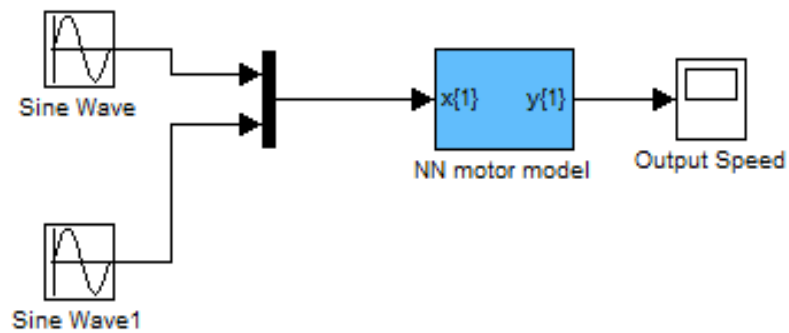


Figure 6.3. NN motor model complete diagram



Figure 6.4. Output signal of NN motor model

From Figure 6.4, we can see the sinusoidal behavior of the system, in this system two sine wave signals are used as input signals, the yellow signal shows the reference signal used to generate the graph and the purple signal shows the output speed.

Complete system results with sine wave input

The complete system; which includes the NN controller and the NN motor model have been built and simulated in MATLAB SIMULINK, the complete diagram is shown bellow.

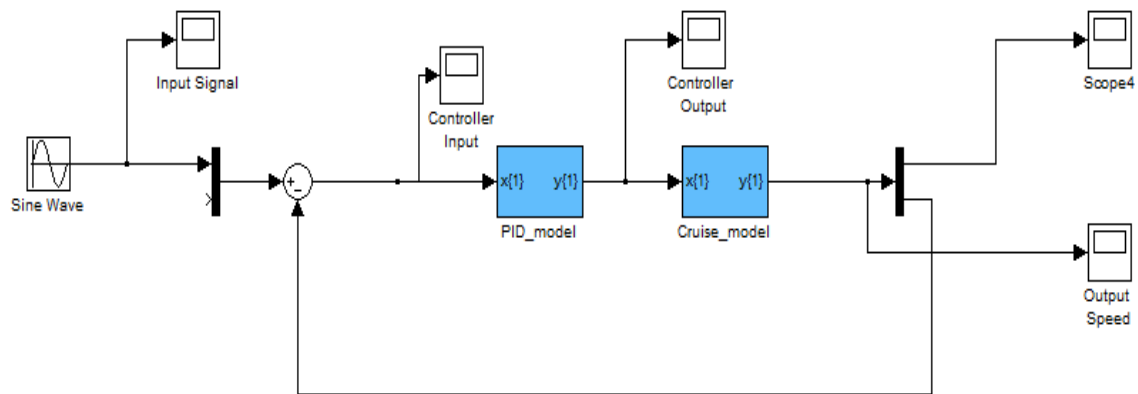


Figure 6.5. Complete system diagram.

In this system we have connected the NN controller to the NN motor model, negative feedback is been used, multi scopes are also used to show the different signals we obtain from the system in order to compare them. The simulation results for this system are shown bellow. From Figures 6.6 to 6.9 we can verify that the output signal is matching the controller output; which confirms that the system is well controlled, from these Figures we can also see that most of the signal distortion due to the non-linear parameters was eliminated. The output speed is completely matching the controller signal.

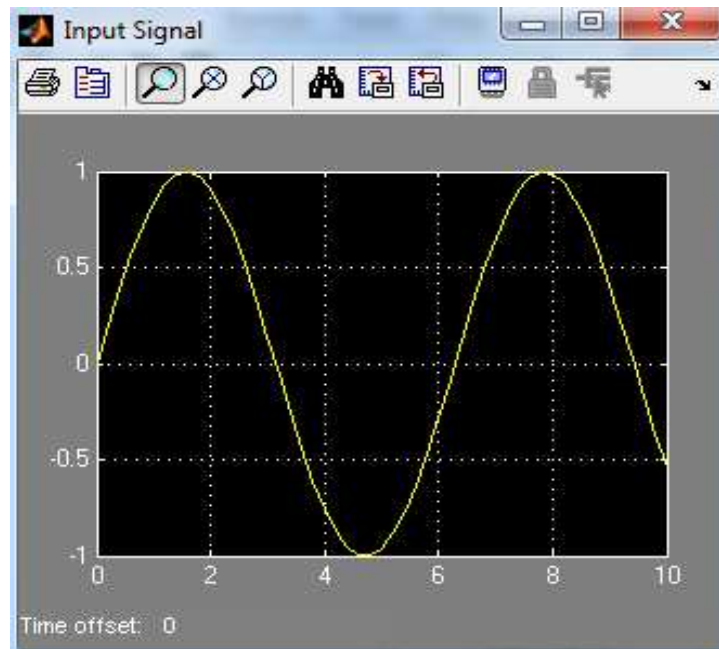


Figure 6.6 Input Signal

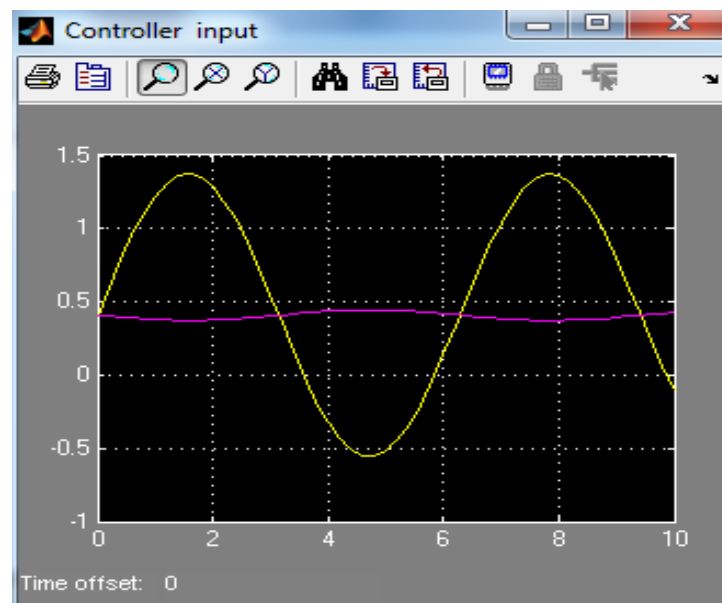


Figure 6.7. NN Controller Input

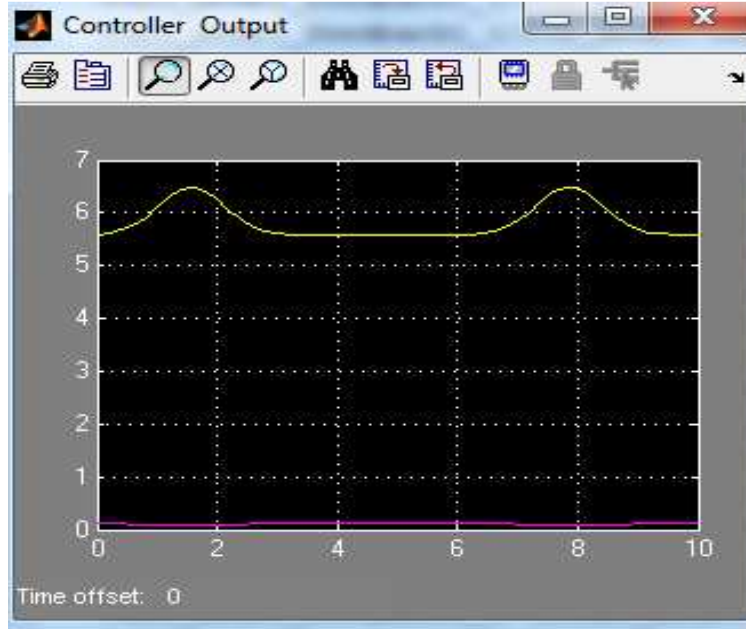


Figure 6.7. NN Controller Output

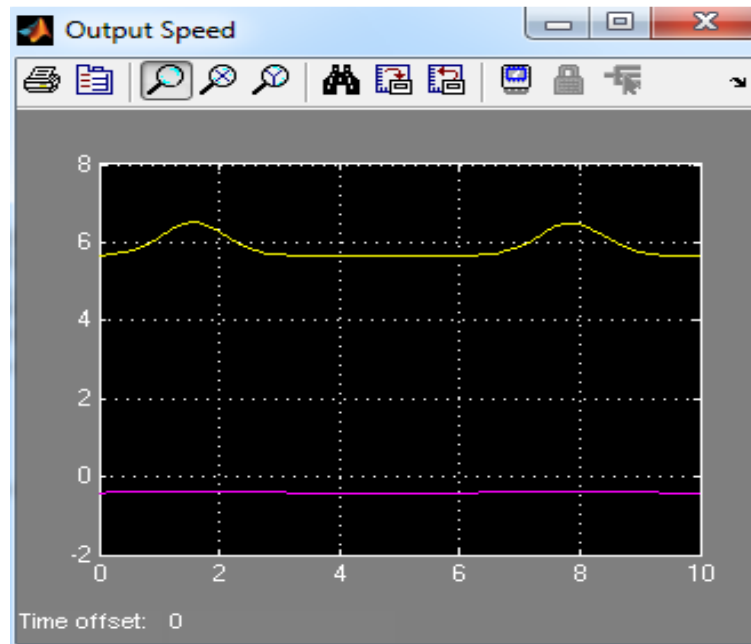


Figure 6.9. Output Speed Signal (Controlled)

Complete System Results with Step Input

To confirm our results, we have used other type of common input signal, which is the step input. All the simulation results obtained from the step input also matching the results that we obtained from the sine wave input. One important result we can get from comparing the system behavior before and after using the NN controller, is that the response time, or the settling time of the system after using the NN controller is less than two seconds compared to the old response with the PID controller itself.

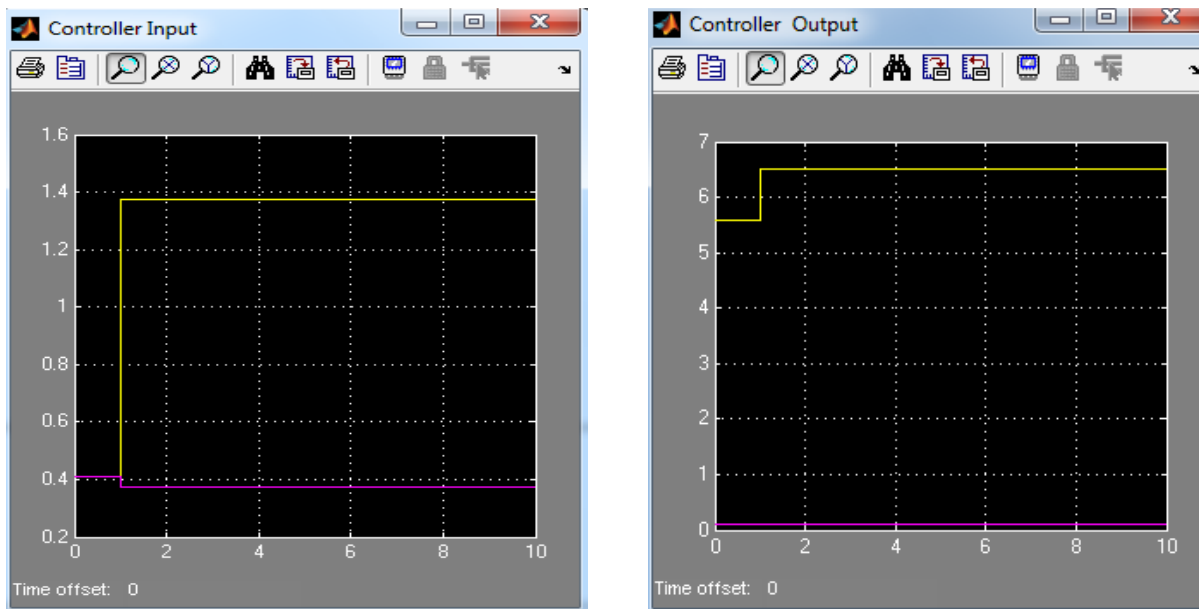


Figure 6.10. Step input response of the NN controller

The difference of the start point is that NN motor model has a 12 volt input signal and the input signal we are using here has value of 1 volt maximum, so that the controller will compensate for the actual value.

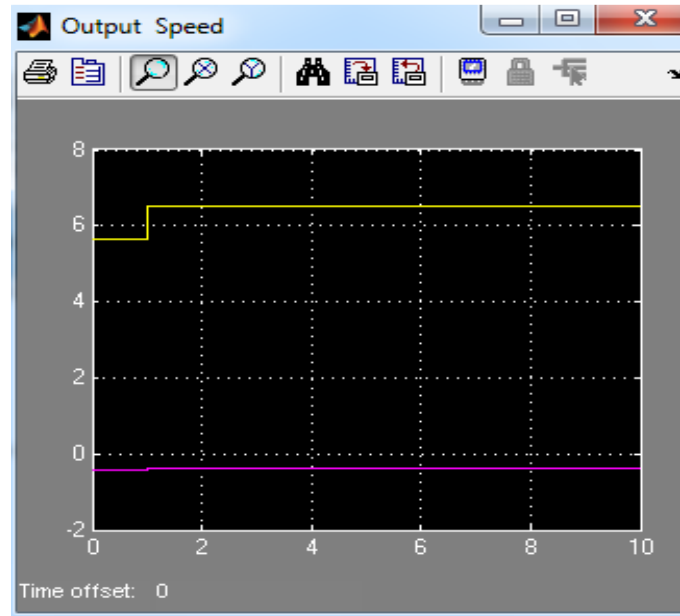


Figure 6.11. Output speed with step input

From the simulation results it is clear to mention the difference of the response of the system for both the time needed to reach the desired output and the shape of the output. The output from the step input has the same as the input signal, which has no distortion.

PID Controller Response

In order to have clearer image about the difference of the two controllers, the conventional PID controller and the NN controller, we have simulated the response of the two systems for the PID controller with step input. The PID controller has been tuned to get the best response possible from the system, the values obtained for the PID parameters values are: $P = 20$. $I = 2$ and $D = 20$. From these values we need to build a controller that consumes more power due to the proportional parameter value. Even though, this controller can be built, due to high gain value this controller may not

be the best solution to our system, taking into consideration the power ratings of the motor; which may not be able to withstand this value of input voltage.

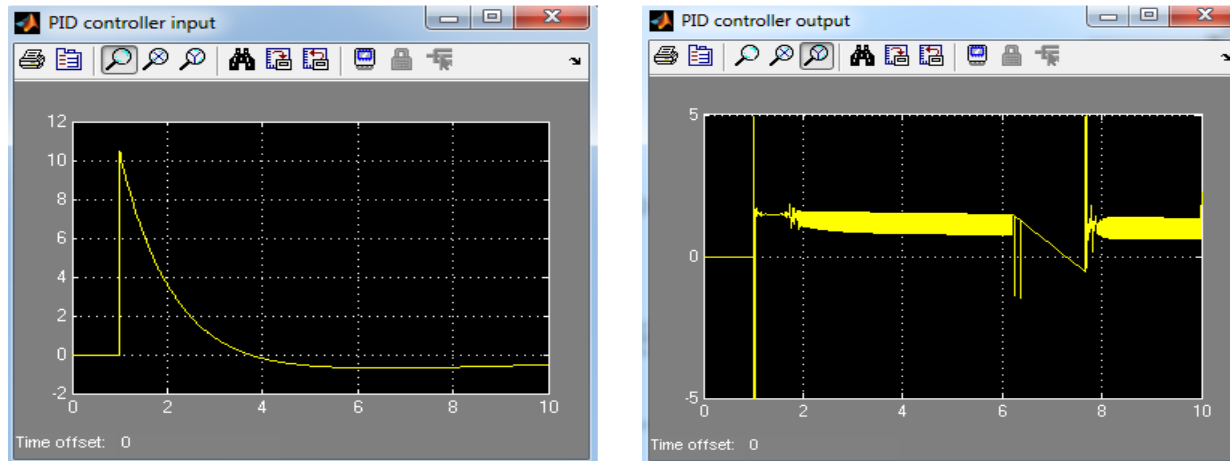


Figure 6.12. PID controller input and output signals.

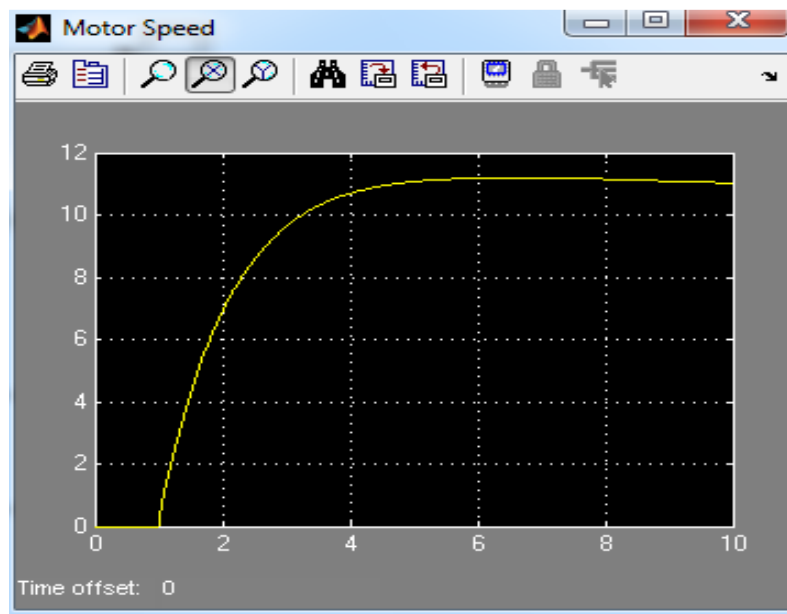


Figure 6.13. Output speed with PID controller

From the response of the PID controller we can mention some disadvantages to use this type of controller. The first one is that we have an overshoot of the input signal to

the PID, this overshoot may cause a failure in the controller function, especially this controller may be running continuously with no stops. The second one is the big difference of the output speed signal compared to the output from the PID controller itself, which means that the controller was not completely successful to eliminate all the non-linear effects. One last important issue of the PID controller is that the response is still slow, even after tuning the PID, which means that if this is a critical time for the function being controlled by the PID then the PID controller will be out of question for this application.

The overall response of the system while using the PID controller is still weak, also this system will not be able to deal with any outside disturbances or any sudden change in the load attached to the servomotor, this is clear from the overshoot of the PID input; in which we can conclude that if any sudden change happen the controller will drain more power leading to burn the components of the controller, and this may lead for motor failure.

NN Controller Response

Recalling the step response of the NN controller shown in Figures 6.10 and 6.11, we can mention the difference in the response compared to the PID controller response. The first difference is that the NN controller input signal, in this signal we can see that there is no overshoot in the signal, which means the controller can keep working in a safe manner without any risk of burning the controller itself. Also this advantage makes the controller to keep the same power level consumption, which is very critical in some remote application if we do not have any kind of power source attached to the system.

The second advantage of the NN controller over the PID is that the output speed signal is following the same shape and values of the controller, this makes the system eliminate any nonlinear effect due to the motor components, or due to any sudden change in the outside environment of the load attached to the motor.

Comparing the time response of the two controllers, we can notice the big difference in the time needed for the signal to reach the final value when using the NN controller, this time tends to be half the time needed when we used the PID controller. This advantage is also very important since in some critical application a time difference of one second may cause unwanted response of the system and may lead to malfunction.

CHAPTER 7

CONCLUSIONS, DELIVERIES, FUTURE WORK AND SUMMARY

Conclusions

From the results obtained for both the PID controller and the NN controller, it is clear that the overall performance of the NN controller proposed was better than the conventional PID controller. The PID controller performance was consistent with old trials of controlling this type of motors, the change in system's parameters does not yield any change in the technique used to tune the PID controller for, but changes the performance of the PID. The PID controller cannot be improved further, since the tuning results were the best to get the output shown on Figure 6.4 and 6.13. The tuning results for the PID controller were best match for the system performance and the ability to build such a controller. The PID controller can be used with servomotors that are not components of very efficient systems or time critical systems, since they will require high power to operate them and may lead to failure in their function due to the high power used by the controller.

On the other hand the NN controller has shown very good results and improvements of the system behavior. The NN controller was able to deal with all the non-linear parameters found on the system, and the output was very consistent with the input of the controller signal. The efficiency of this controller was also better in terms of the response time, as shown in Figures 6.11 and 6.13, the response time to reach the maximum output value was almost 1 second compared to 5 seconds for the PID controller.

The only drawback found in the NN controller is that the output signal was shifted and does not start to rise from zero. This issue of the NN controller response can be easily eliminated and improved by adding a bias to the neural network during training, and with the use of simple components when implementing the controller using hardware.

Deliveries

As deliveries of this research we can mention mainly the following items.

- a) An ANN architecture was developed and trained based on the second order model of a servo motor. This ANN can be used to simulate the operation of a real servomotor system.
- b) A second ANN architecture was developed, that is used to control the operation of the first neural network, this neural network has a better performance than its PID control equivalent, and it was shown to reduce the non-linear parameters and characteristics of a typical servomotor system. It also produces the correct control signals required for the operation of this kind of systems.
- c) Two research papers have been submitted for publication using the results of this project. At the present time, one paper has already been accepted for presentation and the second one is under development to be submitted for acceptance.

Future Work

No further can be done with the PID controller, since the tuning of the controller parameters resulted in the best match of performance and real implementation. While the NN controller still offers some opportunity to continue working with. In particular, the

NN controller can further be improved by first eliminating any signal shift found in the output, also the response time may be improved by using other training techniques, which may be required in some time critical applications.

One Important step to do in the future is to implement both controllers using hardware components, and to test both of them with a real servomotor. This step will also be very helpful to test the performance of both controllers, and may lead to more improvements of the controller function. The most important point about the NN controller is that the more we use it, the more it improves itself and learns how to deal with any new data type and parameter changes.

Summary

In this thesis we are able to test the performance of two types of controllers, PID controller and NN controller, compare them together and show the difference in the performance of each type of them. The NN controller has shown better performance by meeting the problem goals; in which we want to eliminate most of the non-linear effect and to have a kind of a controller that can deal with any new type of data or change in the working environment.

The results show also that the NN controller can be used in high efficient systems and time critical system, in which the PID controller will not be the best choice of a controller for these types of systems.

REFERENCES

- Nise, Norman S.. *Control Systems Engineering*. Fifth edition 2008.
- Hofield, J.J. Neural networks and physical systems with emergent collective computational abilities. 1982.
- Kandel E, Schawrtz JH, Jessel TM. Principles of neural science. 2000
- Halici, Ugur. Artificial neural networks. Ankara 2005
- Kunihiko, Fukushima. A self organizing multilayered neural network. 1975
- Hagan, Martin T.. Neural network design. 1996.
- Owen, Edward L.. "Origins of servomotor". August 2002.
- Mayr, Otto, The Origins of feedback control. MIT press 1970.
- Miller, W.T, Sutton, R.S. and Werbos, P.J. Neural network for control, MIT Press, Cambridge, MA (1990).
- Warwick, K., Irwin, G.W and Hunt, K.J. Neural networks for control and systems, Peter Peregrines Ltd(1992).
- Demuth, Howard and Beale, Mark. Matlab neural network tool box documentation. 2004
- Haykin, Simon, 1999. " An Introduction to Feed Forward Networks". 1999
- Ogata, K, 2005. *Modern control engineering*. Fourth edition. McGraw Hill.

APPENDIX A

MATLAB CODE

```
% The NN motor model design we have 1-25-1, with tansig function
%for hidden layers and purlin for output layer.
```

```
P = sininput;
T = sineoutput;
net = newff(P,T,25);
Y = sim(net,P);
P1 = sininput(1,:);
```

```
Figure1 = plot(P1,Y);
```

```
net.trainParam.show = 50;
net.trainParam.epochs = 1000;
net.trainParam.lr = 0.05;
net.trainParam.goal = 1e-5;
net = init(net);
net = train(net,P,T);
Y1 = sim(net,P);
gensim(net,-1)
Figure4 = plot(P1,Y1);
```

```
% The PID controller NN has also the same structure but with different
% training sets:
```

```
% The NN controller design we have is 1-25-1, with tansig function for
%hidden layers and purlin for output layer.
```

```
P1 = pidin;
T1 = pidout;
net1 = newff(P1,T1,50);
Ypid = sim(net1,P1);
P1pid = pidin(1,:);
```

```
Figure1pid = plot(P1pid,Ypid);
```

```
net1.trainParam.show = 50;
net1.trainParam.epochs = 1000;
net1.trainParam.lr = 0.05;
net1.trainParam.goal = 1e-5;
net1 = init(net1);
net1 = train(net1,P1,T1);
Y1pid = sim(net1,P1);
gensim(net1,-1)
Figure4pid = plot(P1pid,Y1pid);
```