A DISSERTATION

ON

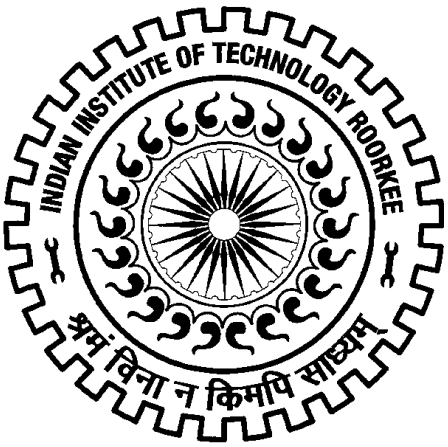
MODEL REFERENCE ADAPTIVE CONTROL USING NEURAL NETWORKS

Submitted in partial fulfillment of Credit of Course Work
Requirement for the award of the degree
of

MASTER OF TECHNOLOGY

By

ROHIT SHUKLA
M.Tech II year S&C
(Enrolment No:-13530018)



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247667 (INDIA)
JULY-2016

CANDIDATE DECLARATION

I hereby declare that this thesis report entitled Model reference adaptive control using neural network, submitted to the Department of Electrical Engineering, Indian Institute of Technology, Roorkee, India, in partial fulfillment of the requirements for the award of the Degree of Master of Technology in Electrical Engineering with specialization in Systems & Control Engineering is an authentic record of the work carried out by me during the period June 2014 to July 2016, under the supervision of **Dr. G.N. Pillai**, **Professor, Electrical Engineering Department, Indian Institute of Technology Roorkee.** The matter presented in this thesis report has not been submitted by me for award of any other degree of institute or any other institutes.

Date- 12.07.16 **Rohit Shukla**
Place- Roorkee

CERTIFICATE

This is to certify that the above statement made by the candidate is true to best of my knowledge and belief.

Dr. G. N. Pillai
Professor
Department Of Electrical Engineering
Indian Institute of Technology, Roorkee

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude and indebtedness to my supervisor**Dr. G.N Pillai, Professor, Electrical Engineering Department, Indian Institute of Technology Roorkee** for guiding me to undertake this seminar work as well as providing me all the necessaryguidance and support throughout this work. He has displayed unique tolerance and understandingat every step of progress, without which this work would not have been in the present shape.

I would also be thankful to all staff of **Electrical Engineering Department** for their constantsupport at and all my friends, for their help and encouragement at the hour of need.

Date: 12.07.2016
Place: Roorkee (ROHIT SHUKLA)

ABSTRACT

The thesis presents a quick overview of Model Reference Adaptive Control (MRAC). The thesis presents model reference based neural network structure that can be used for adaptive control of linear and nonlinear processes. The proposed MRAC neural network scheme is simulated to control the movement of a simple, single-link robot arm. The simulation results show that neural network based MRAC can give satisfactory performance requirements.

Table of Contents

LIST OF FIGURES

Chapter 1

INTRODUCTION

Research in adaptive control began in the mid 1950's– autopilot outline for superior flying machine. But interest got reduced because of the accident in an experimental run. Previous decade saw the advancement of a sound hypothesis and numerous useful applications. Adaptive control has for just about forty years been considered as a standout amongst the most appealing, captivating, and frequently misconstrued ranges inside the field of programmed control. In truth, regardless of its inadequacies {to address execution issues} adaptive control has come a significant long route since initially considered. Prior adding up to somewhat more than a gathering of apparently random general thoughts, adaptive control now lays on a true blue foundational hypothesis which serves to clarify essential ideas and developments in a principled way.

We can understand the importance of use of Adaptive control by an example. Suppose there is container in which two different chemicals are getting mixed which results into exothermic reaction and we have to keep the temperature of the solution below a set value by the help of a coolant and also to maintain a requisite concentration of the solution. In this we will use a control system with two control loops. First control loop is used to regulate the flow of the coolant according to the temperature and second control loop will be used to change the controller gain to regulate the concentration of the solution. This system shows that it can adapt to any type of condition thus elucidates the perception of adaptive control.

Various key neural ideal models have been effectively connected in the zone of adaptive control. Specifically, dynamic systems. It is widely known fact that neural networks offers lots of benefits that can be fructified in the area on control system engineering. The main characteristics of neural network that is applicable in control systems is good approximation of function with non-linear characteristics, it can easily learn by some training data sets and its ability to conglomerate huge amount of data for pattern recognition and drawing inferences.

But at the same time, because of their capability to adapt to approximate random nonlinear functions, feed forward networks which includes radial basis function networks, multilayer perceptron and have allured people in the area of adaptive control. It has been proposed that a simpler, flexible control law can be provided by making use non-linear fuction approximation property of neural networks. We have to train that type of neural controller online so as to provide a nonlinear control law, make it capable of adequate control of the nonlinear plant over the operating range.

1.1 Problem Statement

The control principles used in configuration of most of the system of control are without a doubt linear, perhaps being shaped by state variables' linear combination. To have a good control of non-linear system the adaptive control system should be used in which the gains of the linear controller are varied over course of time. In this controller gains are converged to make the plant follow the response of the reference model over the non-linear range of operation. Irrespective of that probability of the model, plant and control law to be linear, the whole MRAC system is absolutely nonlinear with changing parameters of controller with time.

The technology involving neural networks offers great benefits while dealing with non-linear direct model reference adaptive control (MRAC). It has been proposed that a more general, law with more flexibility can be supplied using this nonlinear function simulation property. A neural controller like that could be trained online to give a nonlinear control law, capable of requisite control of the nonlinear plant over the range of operation. This type of techniques have been successfully used in a number of simulation studies ranging from aircraft control to turbo generator regulation.

1.2 Objectives of Dissertation Work:

The objective of this dissertation work includes studying and analysing the adaptive control system using neural networks, studying the simulation of neural networks and observe the performance of neural network based model reference adaptive control and discover the changes that can make the system more accurate and stable and implement this in another system.

1.3 Organization of the Report:

This report is organised as follows: Chapter 2 introduces the basic model reference adaptive control. Chapter 3 describes the role of neural network in adaptive control and neural MRAC architecture. Chapter 4 deals with learning algorithm of neural network.In Chapter 5 we have seen the simulation of MRAC.

Chapter 2

BASIC MODEL REFERENCE ADAPTIVE CONTROL

Adaptive control is a control method used by a controller, with varying or uncertain parameters, which have to adapt itself to a controlled system. In adaptive control basic idea is to estimate plant / controller uncertain parameters on-line, while making use of measured form of system signals. For example as the aeroplane flies the fuel quantity stored in the plane decreases and weight of the plane decreases thus plane control system has to adjust their controller parameters to keep plane in balanced position. Controller in Adaptive control system is a vibrant system with real time parameter estimation and is absolutely nonlinear. Ordinarily, an adaptive controller have two loops, first is the controller loop with gain adjustment and the second one is normal control feedback loop. The basic diagram of Model reference adaptive control is shown in figure1.

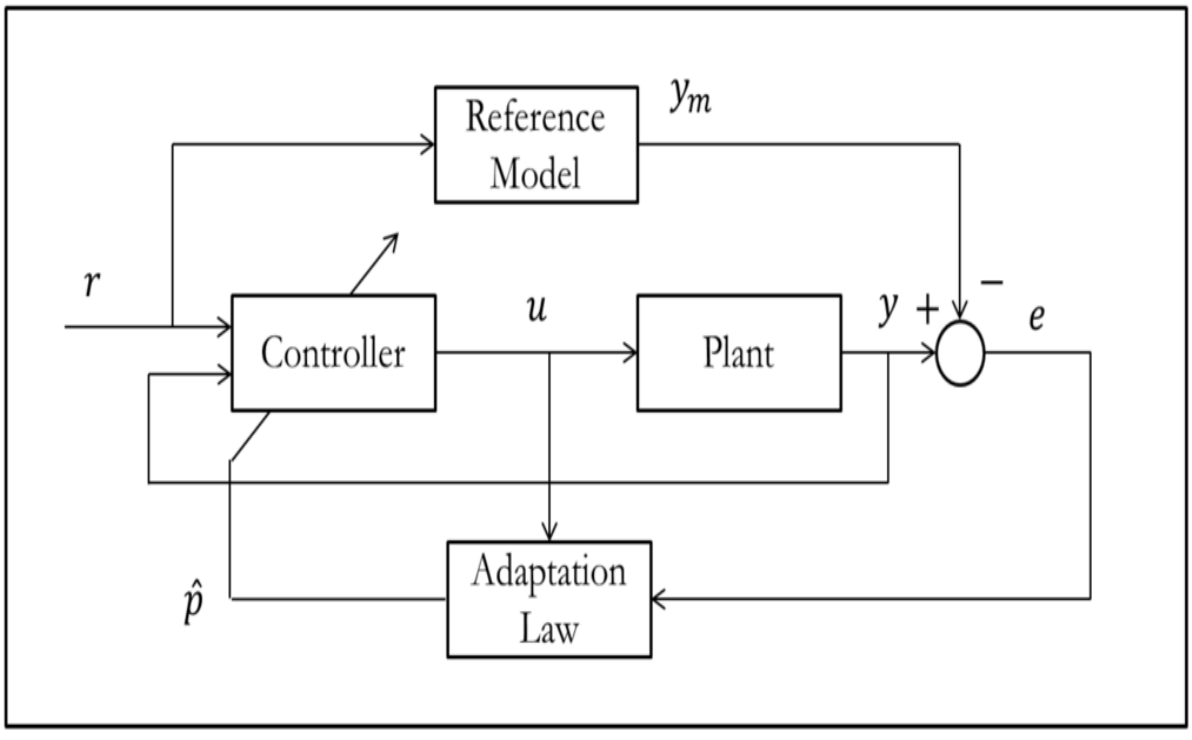


Figure 1: Basic Adaptive Control Model [1]

Plant – has a known structure with unknown parameters.

Reference Model – specifies the required response y_m to the command r given externally.

Controller - having uncertain parameters and have to adapt itself according to varying conditions and to keep balance between the reference model output and plant output .

Adaptation Law - a mechanism which uses the error between the reference model output and plant output to control the controller gain.

These techniques have been applied in a number of simulation studies ranging from aircraft control to turbo generator regulation.

Chapter -3

NEURAL NETWORKS FOR ADAPTIVE CONTROL

In systems and control engineering, feed forward networks like as radial basis function networks and multilayer perceptron have been alluring mostly. This is definitely due to the powerful feature like arbitrary multidimensional non-linear mappings which can be easily used within nonlinear modelling and control structures. In the figure 2, the structure of a general feed forward network is shown.

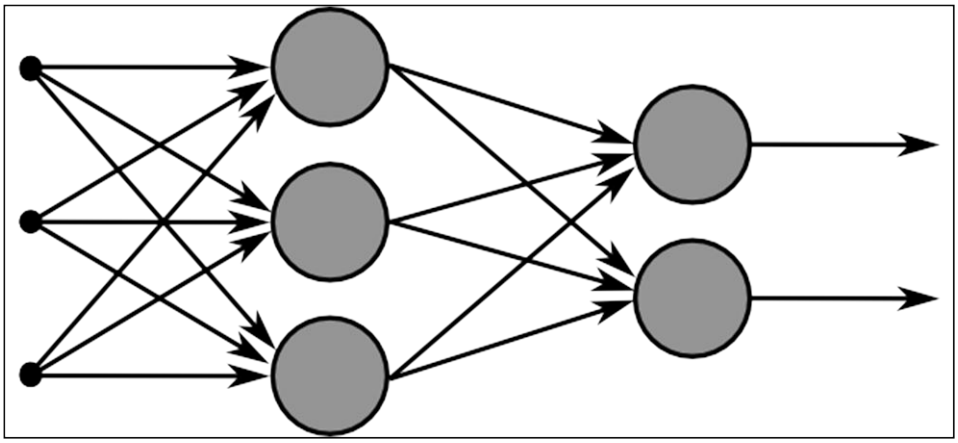


Figure 2: A simple Feed forward Network [2]

The neurons in different layers can have different activation functions for example the hidden layer neurons can have sigmoidal activation function and output layer can have linear transfer function. Inter neural links have weights and these weights are the parameters that are adjusted at time of neural network training. Such NNs are also known as 'Multi-Layer Perceptrons'.

In the feed forward network weighted sum of the output of one layer is passed as information and after that it went as input to a nonlinear function known as the activation function and which gives the output. In case of multilayer perceptron network, the activation function is mainly sigmoidal in shape. Given the neurons in a layer have same activation function, in equation 1 we have given the basic functions of feed forward networks in terms of the activation

function ψ (.), here c_q represents input weight vector and Θq is scalar bias for the qth node in the hidden layer.

The basic function of a feed forward network can be explained in below eqn. as the weighted sum of N_h basis functions.

$$f(v) = \sum_{q=1}^{Nh} \zeta_q \phi_q(v) \qquad - \quad (1)$$

Here, the input vector v € R^{Ni} and ϕq (.) is the qth basis function cone and output is connected to it by weight ζq.

3.1 Neural MRAC Architecture

As shown in figure 3, in the neural model reference adaptive control system, we use controller based on neural network in place of conventional controller which provides more accurate results and fast convergence in case of highly non-linear plants. Model reference architecture includes twin neural based networks which is shown in the figure 3-

1) Controller Network

2) Plant Model

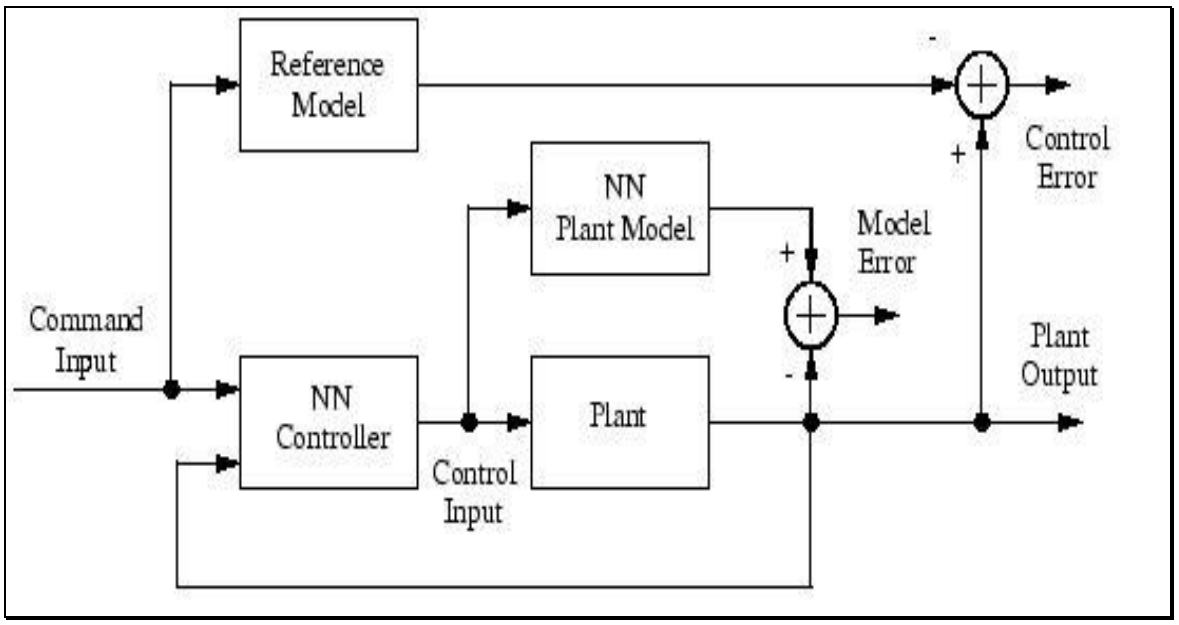


Figure 3: Neural Model Reference Adaptive Control System [3]

3.1.1 Controller Network

In this we have used a NARX network. If previous data for a time series is given then the NARX networks can be easily trained to estimate complete time series which is the feedback input, and another time series, known as external time series. In this we have used 'trainlm' (Levenberg-Marquardt back propagation) as training function to train the network which uses a fast back propagation algorithm. The error between the ref. model o/p and plant o/p is used to adjust the weight of the neural controller.

3.1.2 Plant Model Network

It is also a NARX network. The plant model is based between the error signal and neural controller network, hence neural plant model is used by which the error is conveyed to provide a suitable descent direction at the output of the neural based control network. In Figure 4, the detailed view of the adaptive control system is shown.

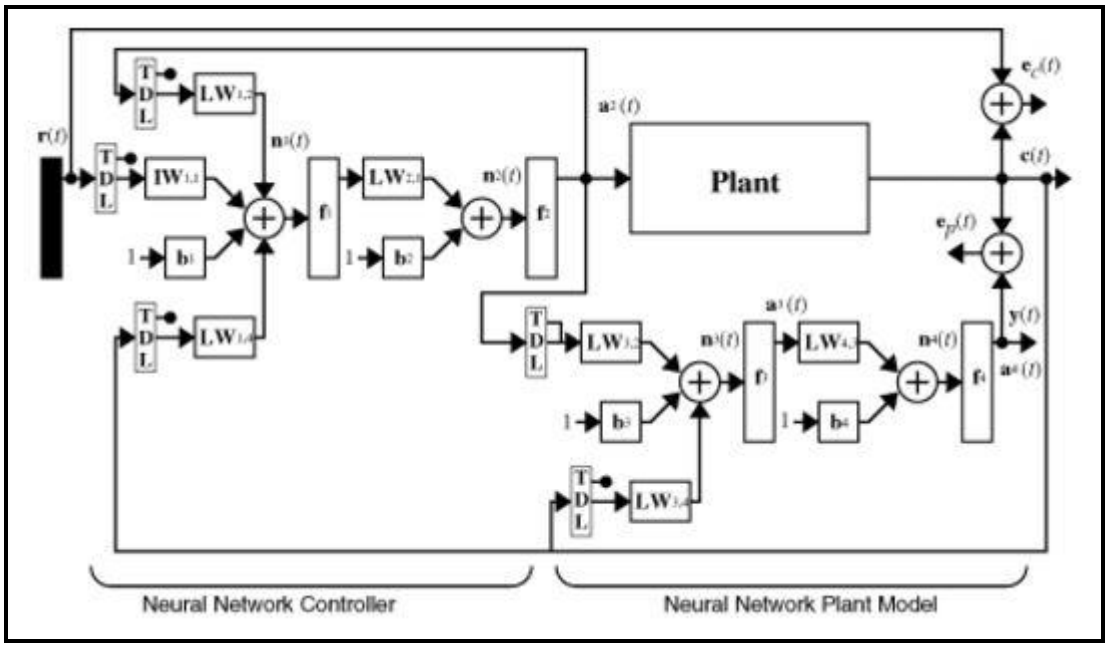


Figure 4: Internal Structure OF MRAC system [4]

Each network have two layers, and we can select the numbers of neurons that is to be used in the hidden layers.

Chapter -4

DIRECT NEURAL MRAC

We have seen that basic ability of neural network to approximate the nonlinear has made it lucrative. Specifically, following control law can be conceived by the help of a neural network:

$$u(t) = f(x_p(t), r(t), w(t)) \qquad \text{-(2)}$$

The neural network can be used to signify a large variety of control laws with non-linearity, having the same structure, by simply adjusting the weights of controller in training for getting desired accuracy in results. One probable MRAC structure, based on the capability of the neural network to form a nonlinear control law, is in figure 5.

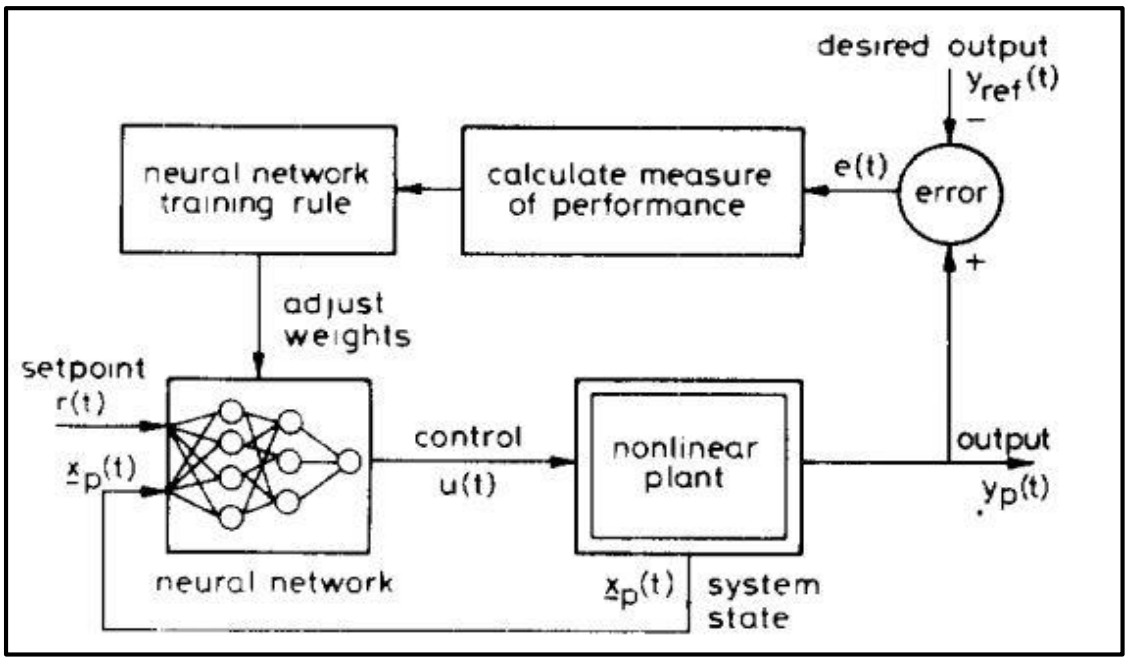


Figure 5: Training Mechanism for Controller [5]

4.1 LEARNING ALGORITHMS FOR ANN

Learning algorithm is the rule for adjusting the inter neural weights during training of the feed forward network. Main learning algorithms for multilayer feed forward networks are:

- Back propagation algorithm.
- Extreme learning machine

Back propagation algorithm is a supervised gradient descent method for weight adjustment of inter neural linkages of multilayer feed forward network. The main idea of this algorithm is that the hidden layer neurons do not make any error, rather they contribute to the final error and depending upon their contribution to the error their weights with previous layer is adjusted.

This is the most widely used ANN training method because of its simplicity but, it has two major disadvantages:

- Its slow rate of convergence
- Second being the possibility of error function to settle at local minimum rather than the global minimum.

Extreme learning machine or ELM is a method for weight adjustment of hidden layer of a single hidden layer feed forward network. In this method the weights of the hidden layer neurons are randomly set and the weights of the output layer are calculated using pseudo inverse.

In training a network to get the required function approximation we have to change all the weights and biases so as to minimise the cost function over a batch of suppose N_p training patterns. Here f_{des} (i) and f (i) represent the the desired function and network approximation respectively.

$$J(w(k)) = \frac{1}{2N_p} \sum_{i=1}^{Np} (f(i) - f_{des}(i))^2 \qquad - \quad (3)$$

Earlier, method of gradient-descent was suggested in which the back propagation method was used to determine the gradient of the cost function of equation above with respect to the network weights. The weights were then modified in the direction of the negative gradient as shown in eqn below.

$$g(k) = \frac{\partial J(w(k))}{\partial w(k)} \quad - \quad (4)$$

$$w(k+1) = w(k) - \eta(k) g(k) \quad - \quad (5)$$

To issue the problem associated with this structure, following procedure we have to use:

1) We use the concept of generalized learning and then specialized learning of a neural control principle. In start in generalized learning we start with a rough approximate model and trained that to the desired control law,. In this way, the neural controller can drive the plant over the operating range and with better stability.
2) Then we perform on-line specialized learning so as to improve the control provided by the neural network controller.

4.2 Steps for Better Stability

To provide better stability and better control neural control law is applied with a constant gain controller in parallel. Thus the plant can be then comfortably driven over the range of operation, with the online training of neural network to better the control process. Thus the network now formed can be used for neural model reference adaptive control of SISO nonlinear plant as shown in figure 6.

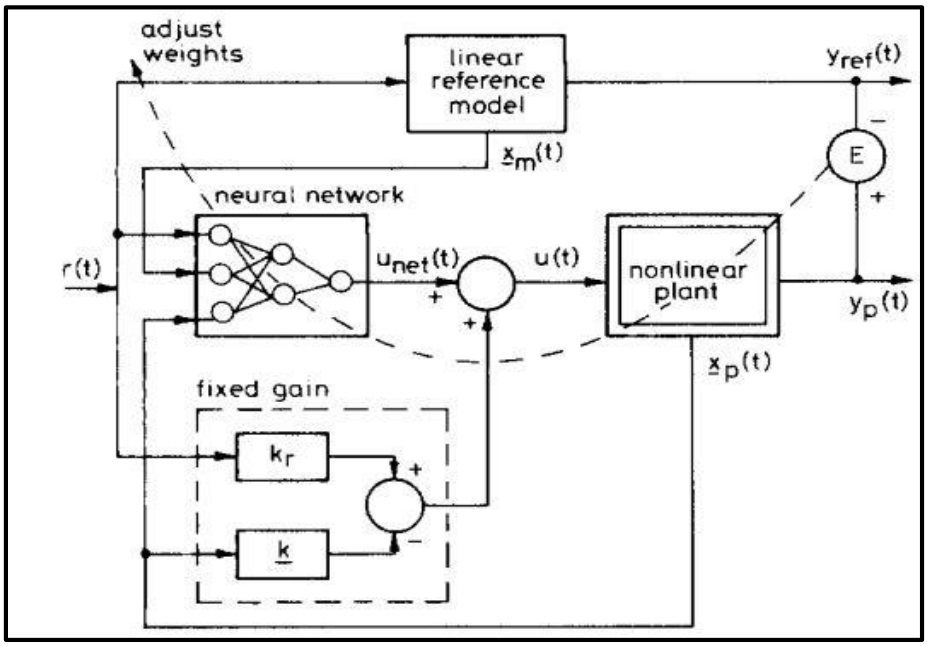


Figure 6: Neural MRAC with Fixed gain [5]

Suppose K_r and k represents a nominal state space control principle with constant gain. After applying a fixed gain in parallel, the whole nonlinear control provided by the network in is expressed in eqn below, including the state of the reference model x_m (t).

$$u(t) = kr(t) - k^t x_p(t) + u_{net}(t) \quad - \quad (6)$$

$$u_{net}(t) = f(X_p(t), x_m(t), r(t)) \quad - \quad (7)$$

Because of the nonlinearities both within the neural network and within the plant, suitable training rules based on stability for direct neural controllers are still not completely developed. Thus, here we have used gradient-based adaptive control principle, which uses the back propagation method to determine the gradients of suitable cost function with respect to every weight in the network. The following single-pattern version of the cost function of eqn. 8 is usually adopted for direct neural control:

$$J(k, w(k)) = \frac{1}{2}(y_p(k) - y_{ref}(k))^2 = \frac{1}{2}e(k)^2 \quad - \quad (8)$$

The weight vector can then be changed by use of the gradient descent method of eqn. 4. However, for control structure proposed in Fig. 2, the plant is based between the error and the neural controller network, hence it is necessary to develop a method by which the error at the output of the plant could be easily conveyed to provide a correct direction at the output of the neural network. Note that generally the step size is adjusted, according to the magnitude of the gradient vector, to to aid convergence and effectively normalize the update direction.

$$w(k+1) = w(k) - \eta(k)[\frac{\partial y_p(k)}{\partial u_{net}(k)} e(k)] \frac{\partial u_{net}(k)}{\partial w(k)} \quad - \quad (9)$$

But at the same time, if less information of the nonlinear plant is present, it is tough to obtain an expression for the Jacobian.

Chapter -5

SIMULATION OF MRAC

In this, we will see the example of robot single link arm control shown in the figure 7, the objective is to control the simple, robot single link arm. When we give input to the robot to move the arm to a certain position, since the controller cannot directly make robot arm to reach the required position, thus, here we have used adaptive control mechanism. The reference model will give the required position of the arm for a certain input and the neural controller weights will be continuously adjusted according to the error between the arm actual position and the reference model output.

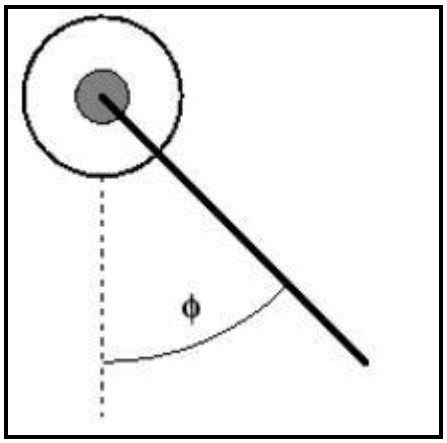


Figure 7: Robot Arm dynamics [4]

The equation of the motion of the arm is:

$$\frac{d^2\phi}{dt^2} = -10\sin\phi - 2\frac{d\phi}{dt} + u \qquad - \quad (10)$$

Here ϕ is showing arm angle, and u represents DC motor supplied torque.

We have to train the neural controller so that the arm chase the reference model

$$\frac{d^2 y_r}{dt^2} = -9y_r - 6\frac{dy_r}{dt} + 9r \qquad - \quad (11)$$

Where y_r is the output of the reference model, and r is the input reference signal.

In this example a narx architecture based neural network controller is used. The two delayed reference inputs are inputs to the controller, are one controller output and two plant outputs.

5.1 Procedure for creating the MRAC network

1) We will start by training a NARX network for the plant model sub network (robot arm) . By the following script we will get plant model of figure 8.

```
[u,y] = robotarm_dataset;
d1 = [1:2];
d2 = [1:2];
S1 = 5;
robo_net = narxnet (d1, d2, S1);
robo_net.trainParam.min_grad = 1e-10;
[P, Pi, Ai, t] = preparets (robo_net, u, {}, y);
robo_net = train (robo_net, p, t,Pi);
robo_net_closed = closeloop(robo_net);
view (robo_net_closed)
```

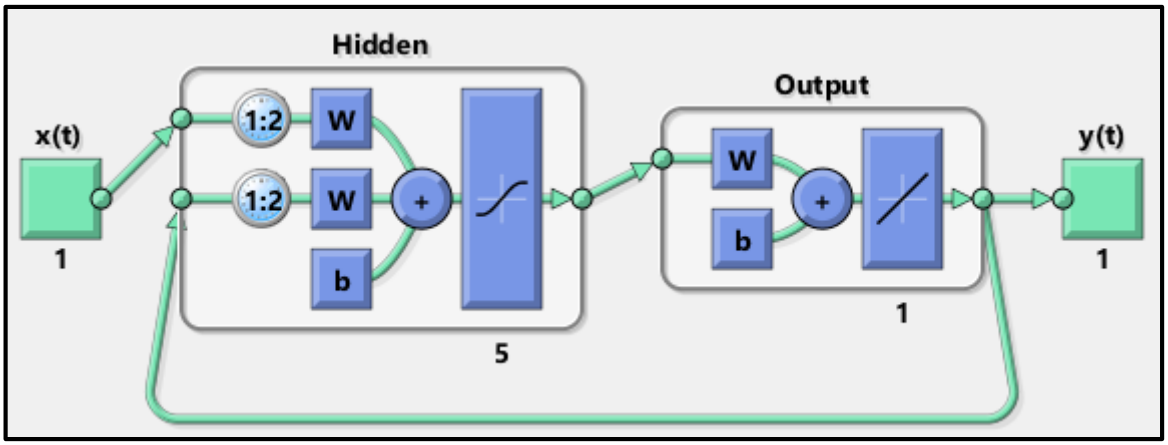


Figure 8: Neural Plant Model

2) We have trained the NARX plant model, now we can make the complete MRAC system and insert the NARX model in that. We have to first make a feed forward network, and then we will add feedback connections. Also, learning is to be turn off in the plant model sub network, since it has been trained already. The controller sub network will be trained in the next stage of training only.

```
adap_control_net = feedforwardnet([S1 1 S1]);
adap_control_net.layerConnect = [0 1 0 1;1 0 0 0;0 1 0 1;0 0 1 0];
adap_control_net.outputs{4}.feedbackMode = 'closed';
adap_control_net.layers{2}.transferFcn = 'purelin';
adap_control_net.layerWeights{3,4}.delays = 1:2;
adap_control_net.layerWeights {3,2}.delays = 1:2;
adap_control_net.layerWeights{3,2}.learn = 0;
adap_control_net.layerWeights{3,4}.learn = 0;
adap_control_net.layerWeights{4,3}.learn = 0;
adap_control_net.biases{3}.learn = 0;
adap_control_net.biases{4}.learn = 0;
```

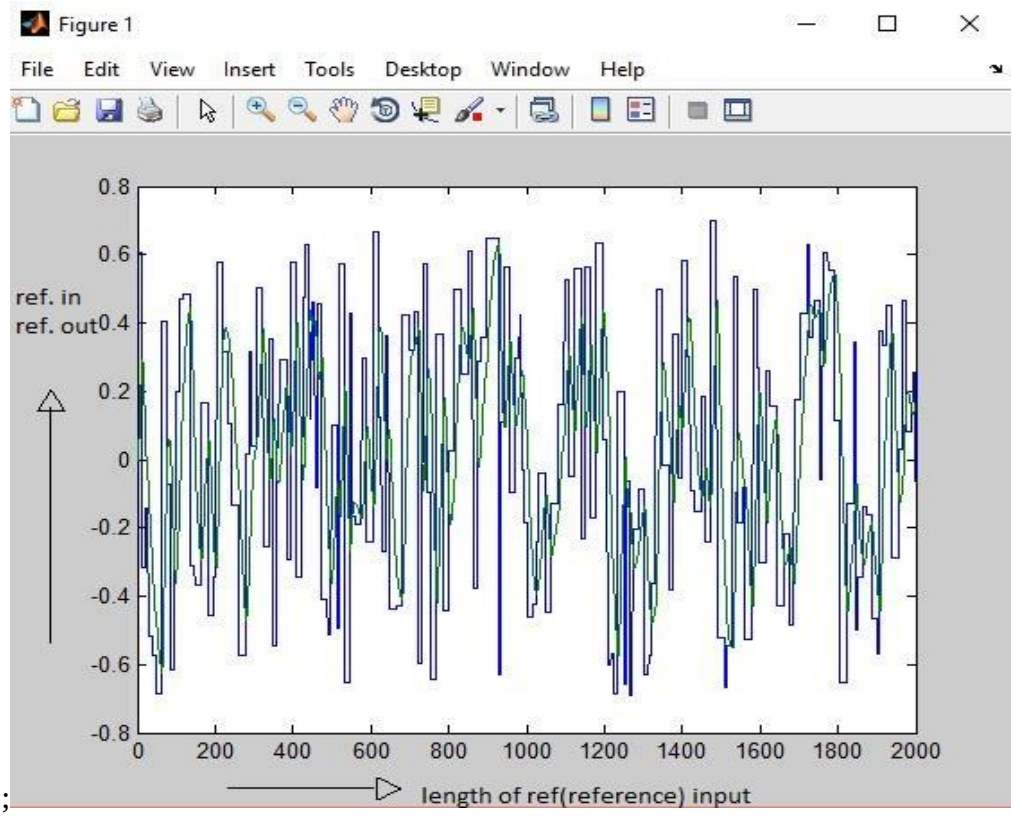
3) The below code is used to turn off preprocessing and data division, which are not needed in this example problem. Delays required for certain layers and names the network will also be set by this.

```
adap_control_net.divideFcn = '';
adap_control_net.inputs{1}.processFcns = {};
adap_control_net.outputs{4}.processFcns = {};
adap_control_net.name = 'Model Reference Adaptive Control Network';
adap_control_net.layerWeights{1,2}.delays = 1:2;
adap_control_net.layerWeights{1,4}.delays = 1:2;
adap_control_net.inputWeights{1}.delays = 1:2;
```

4) For the network configuration, we need sample training data. The following code is used for loading and plotting the training data as shown in figure 9 ,i.e. reference input and reference model output over the length of reference input , the network is configured:

```
[refin,refout] = refmodel_dataset;
ind = 1:length(refin);
```

```
plot(ind,cell2mat(refin),ind,cell2mat(refout))
dap_control_net = configure(adap_control_net,refin,refout);
```



;

Figure 9: Plot of Training Data

5) Now in the appropriate location location of the system weights from the traine plant model will be inserted.

```
adap_control_net.LW{3,2} = robo_net_closed.IW{1};
adap_control_net.LW{3,4} = robo_net_closed.LW{1,2};
adap_control_net.b{3} = robo_net_closed.b{1};
adap_control_net.LW{4,3} = robo_net_closed.LW{2,1};
adap_control_net.b{4} = robo_net_closed.b{2};
```

6) We will make the output weights of the controller zero, which will give the plant an initial zero input.

```
adap_control _net.LW{2,1} = zeros(size(adap_control _net.LW{2,1}));
adap_control _net.b{2} = 0;
```

7) We can view the complete MRAC network with the following command as shown in the figure 10.

view(adap_control _net)

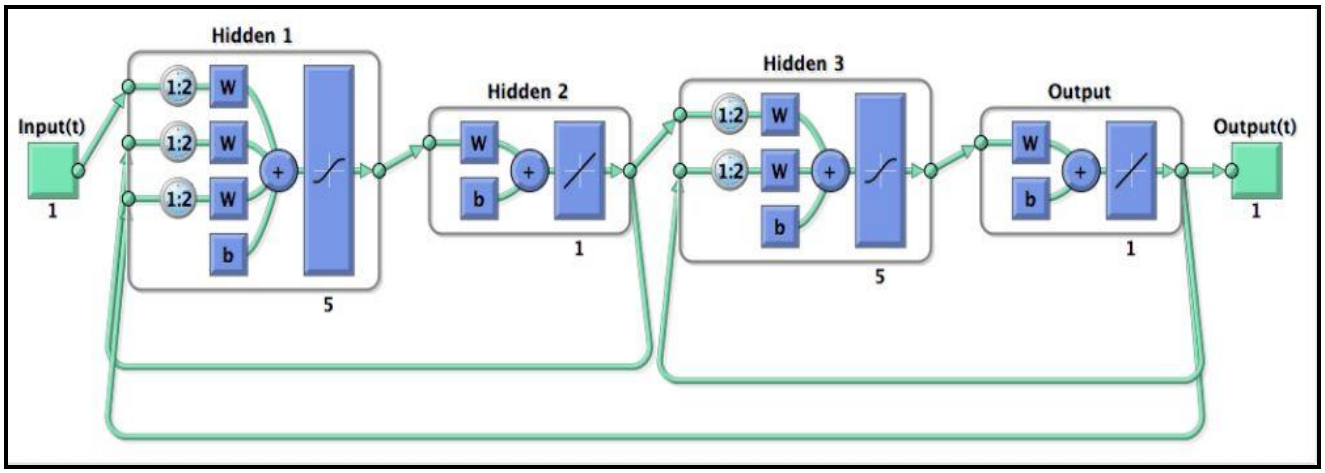


Figure 10: Complete MRAC for Robot Arm

Plant model sub network- Layer 3 and layer 4 (output).

Controller - Layer 1 and layer 2 .

8) We can then prepare the training data and start the network training.

```
[x_tot,xi_tot,ai_tot,t_tot] = preparets(adap_control _net,refin,{},refout);
adap_control _net.trainParam.epochs = 50;
adap_control _net.trainParam.min_grad = 1e-10;
[adap_control _net,tr] = train (adap_control _net,x_tot,t_tot,xi_tot,ai_tot);
```

It will take more time in training of the MRAC system in comparison of training of the NARX plant model. Due to the recurrence in network and use of dynamic back propagation.

After training the network, we can check the operation by applying a test input to the MRAC network, which is a series of steps of random height and width, and applies it to the trained MRAC network and the output can be seen in the figure 11.

```
testin = skyline(60,10,20,-1,1);
testinseq = con2seq(testin);
testoutseq = mrac_net(testinseq);
```

```
testout = cell2mat(testoutseq);
figure
plot([testin' testout'])
```

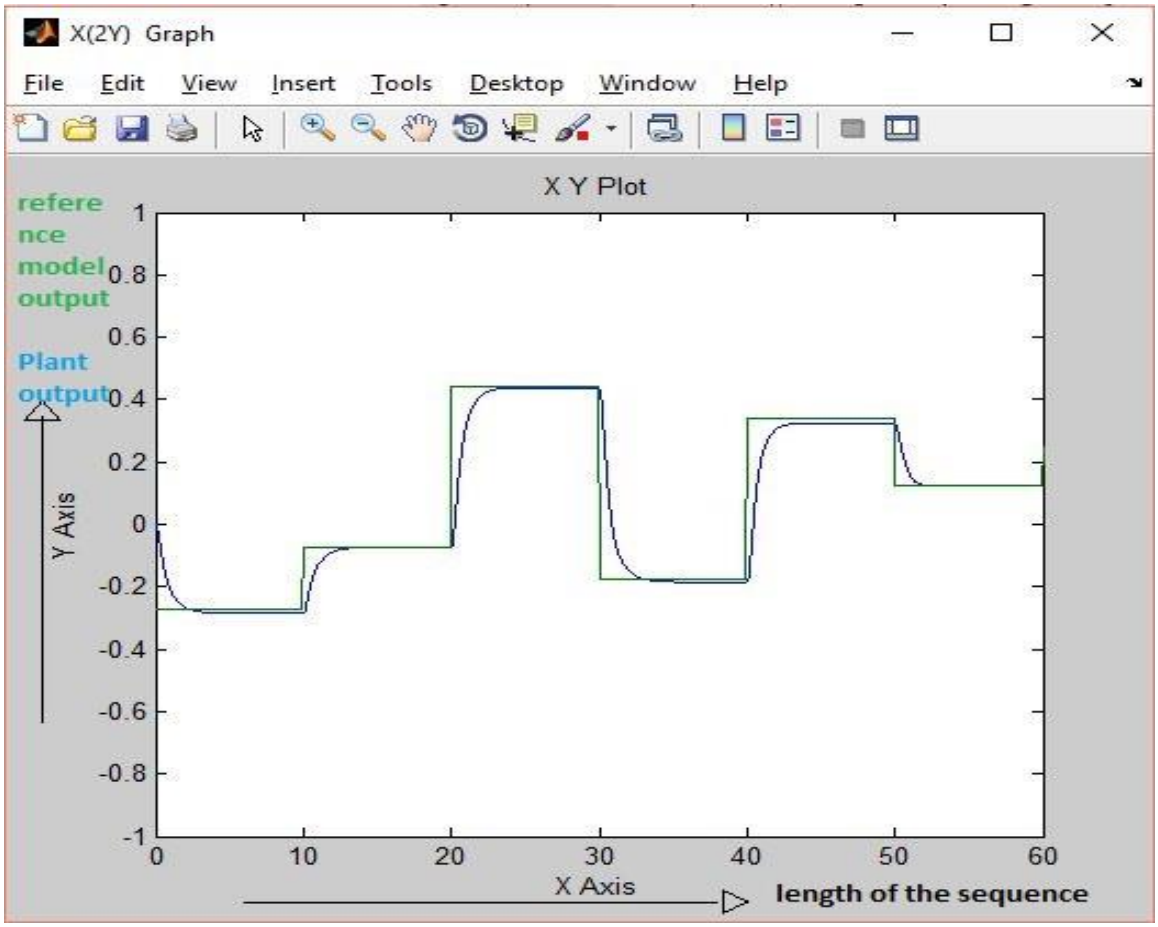


Figure 11: O/p of MRAC system

In figure11, it is clear that the plant model output is following the reference output like correct critically damped response, even if the sequence of i/p and the sequence of input in the training data are not the same. The steady state response for every step is not perfect and to improve this big training set has to be taken and perhaps more no. of hidden layer neurons.

In MATLAB tool box, there is a block for MRAC, shown in figure 12, which we can use directly.

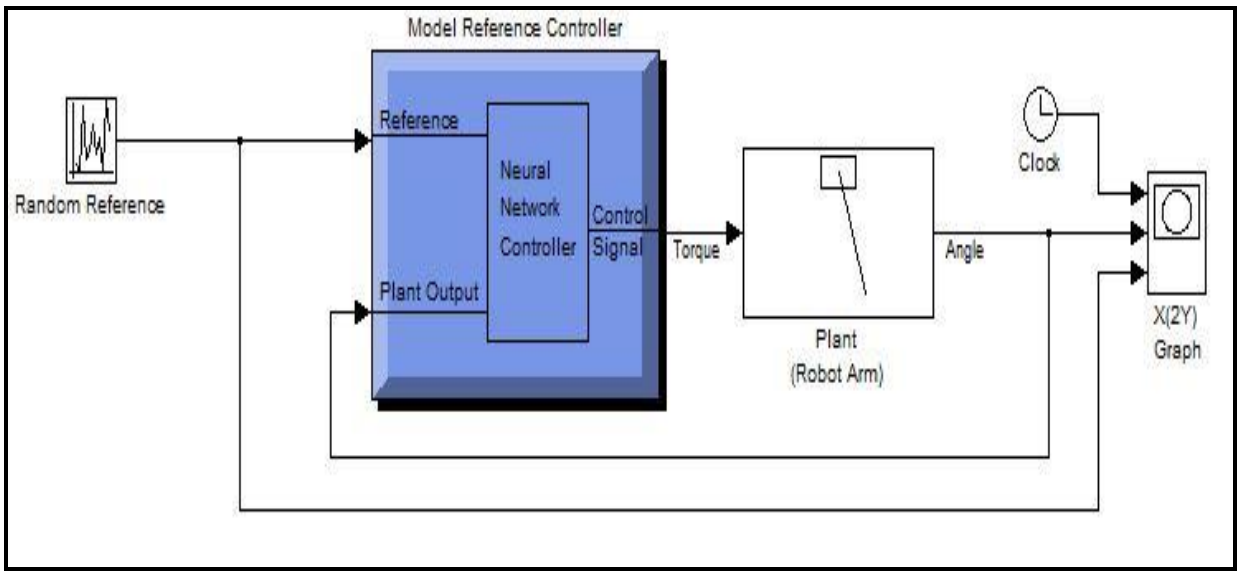


Figure 12: MRAC Block

5.2 SIMULATION OF FIRST ORDER SYSTEM

We have taken the example of a first order system from a paper [5], in this we will check the performance of the adaptive control system using neural network with the help of MRAC block in MATLAB. We will also use a parallel constant gain in parallel with the MRAC block to reduce the error and improve stability of the system. The plant and reference model equations are as follows:

$$\frac{dy_p(t)}{dt} = 2y_p \cos(y_p(t)) + u(t)$$

$$u(t) = -4y_p(t) + r(t)$$

$$\frac{dy_p(t)}{dt} = -3y_p(t) + r(t)$$

1) First we will make the Simulink model for the plant as shown in figure 13 and reference model as shown in figure 14.

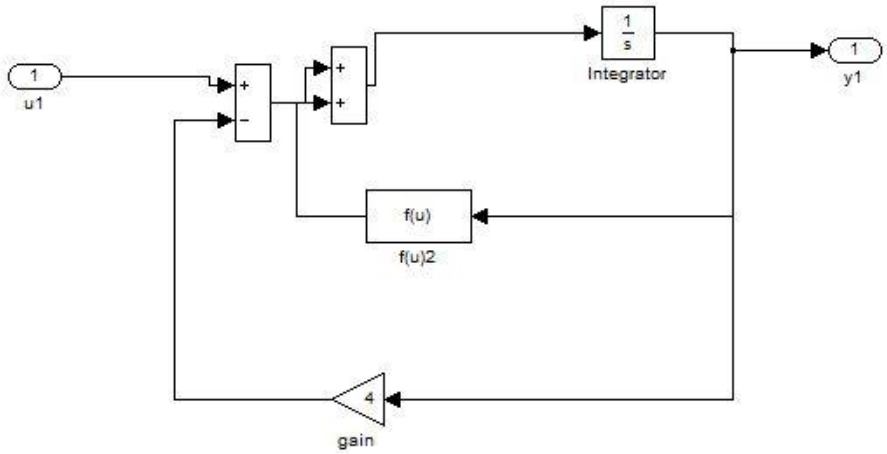


Figure 13: Simulink Model of Plant

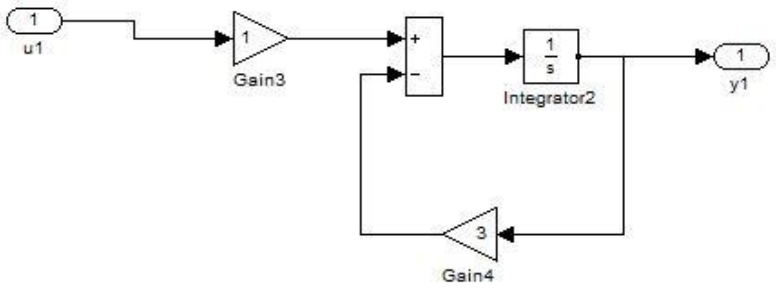


Figure 14: Simulink Model of Reference Model

1) When we will double click on the MRAC block of figure 12. The window of figure 15 will get open. In the window of figure 15, we can fetch the Simulink model of plant and reference model and then select the requisite no of the hidden layer neurons and other credentials

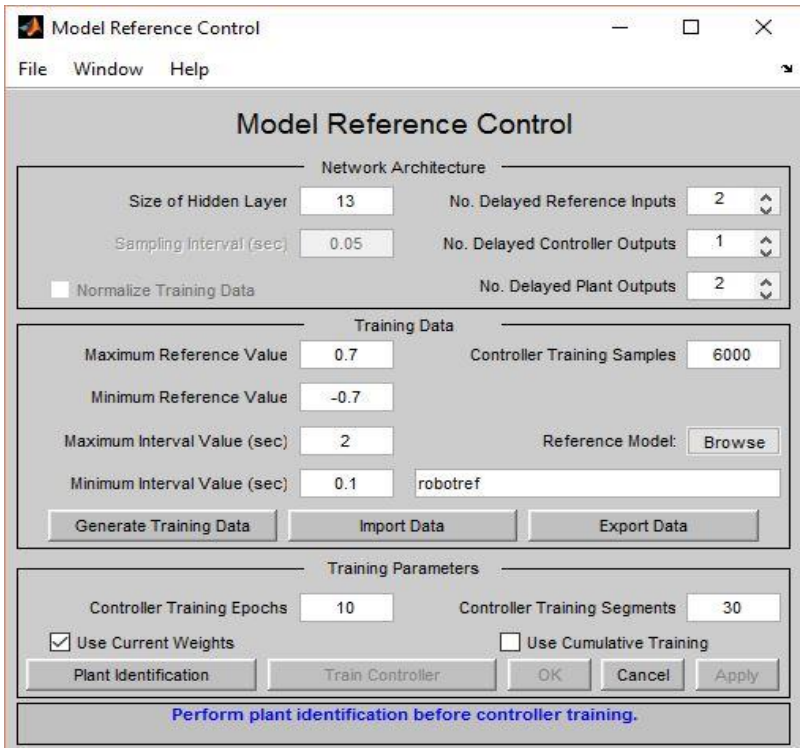


Figure 15: MRAC block configuration window

When we will click on tab of generate training data the system will generate the training data on the basis of previous values. First we will see the training data for plant model in figure 16.

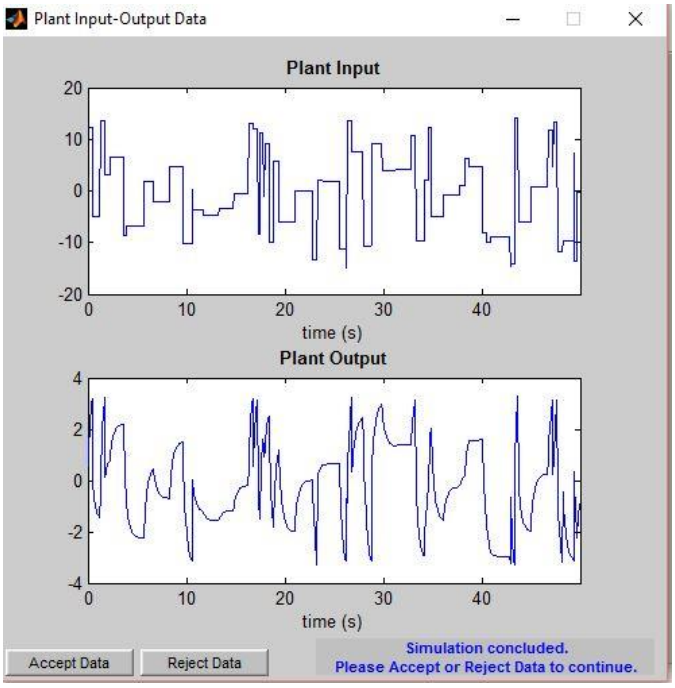


Figure 16: Training data of Neural plant

Once the training data for plant got generated, we will accept the data and train the neural plant and the testing results will be like figure 17.

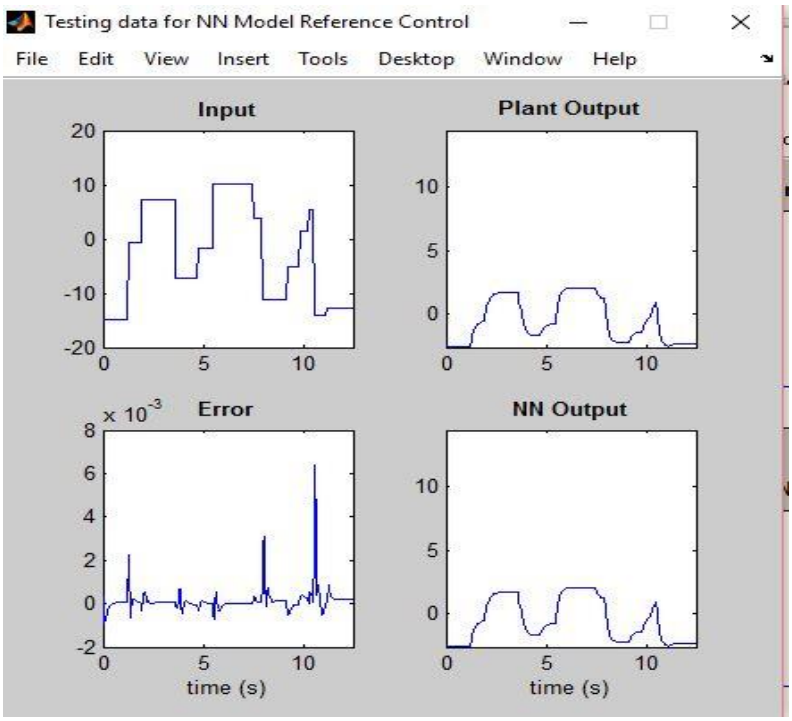


Figure 17: Testing data for NN plant model

Once the plant neural model gets trained we can generate the training data for neural model reference control in which neural plant is also included as figure 18.

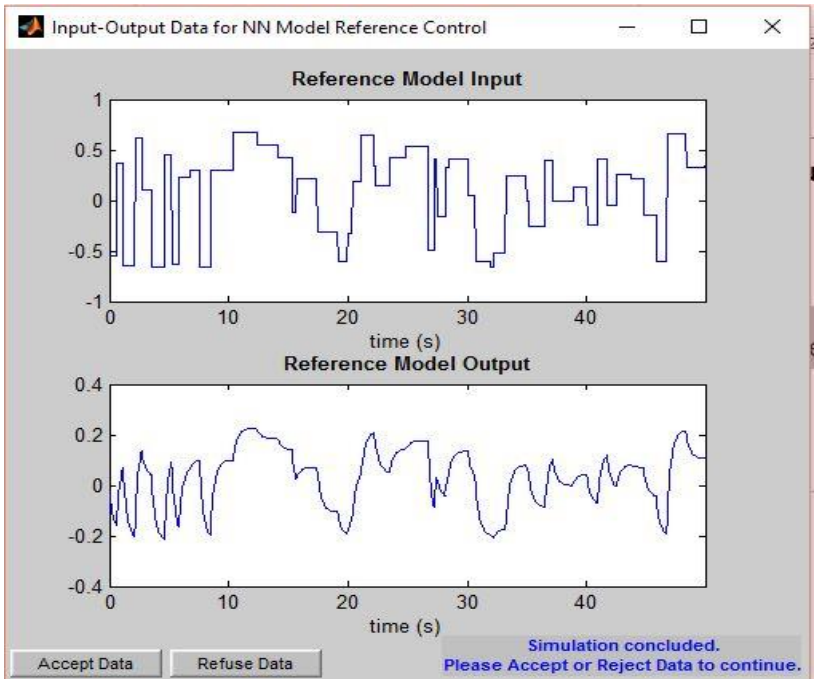


Figure 18: Input O/p for NN Model Reference Control

Then, we will accept the training data and train the NN model reference control. We can select the no. of hidden layer neuron as required. The performance of the model while testing is as shown in figure 19.

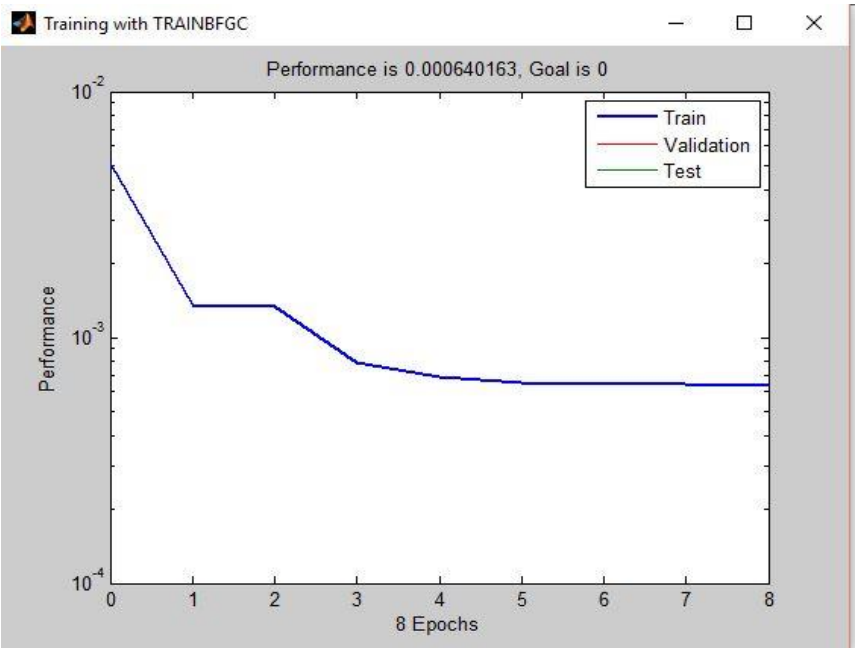


Figure 19: Performance during Training

The results of the training in terms of plant response is shown in figure 20.

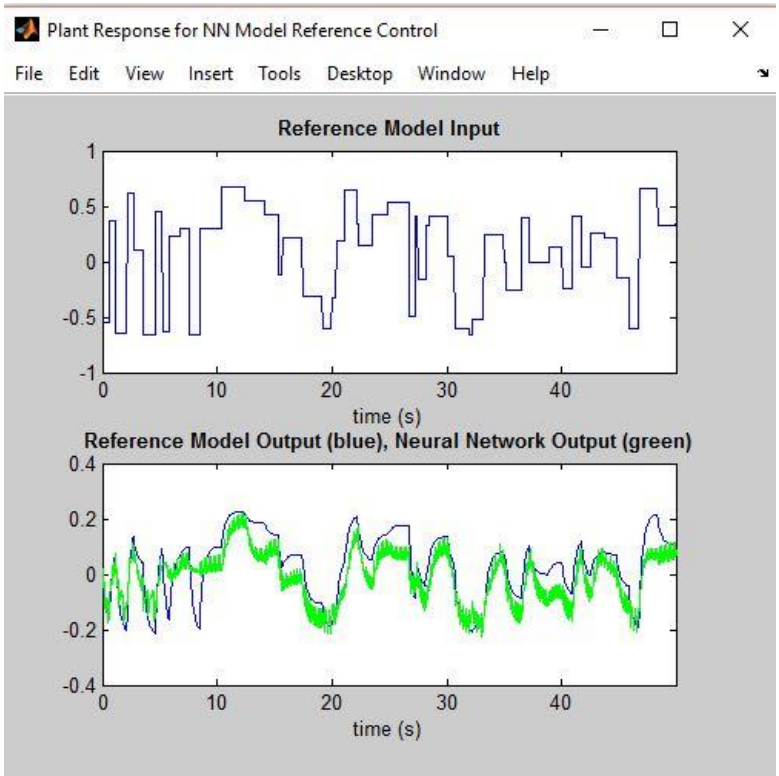


Figure 20: Plant response during training

Then we will check the output of the plant for randomly generated input signal within a limited range and will be like figure 21. in this on Y – axis the output of reference model and plant are compared and on x- axis the length of generated i/p string is taken.

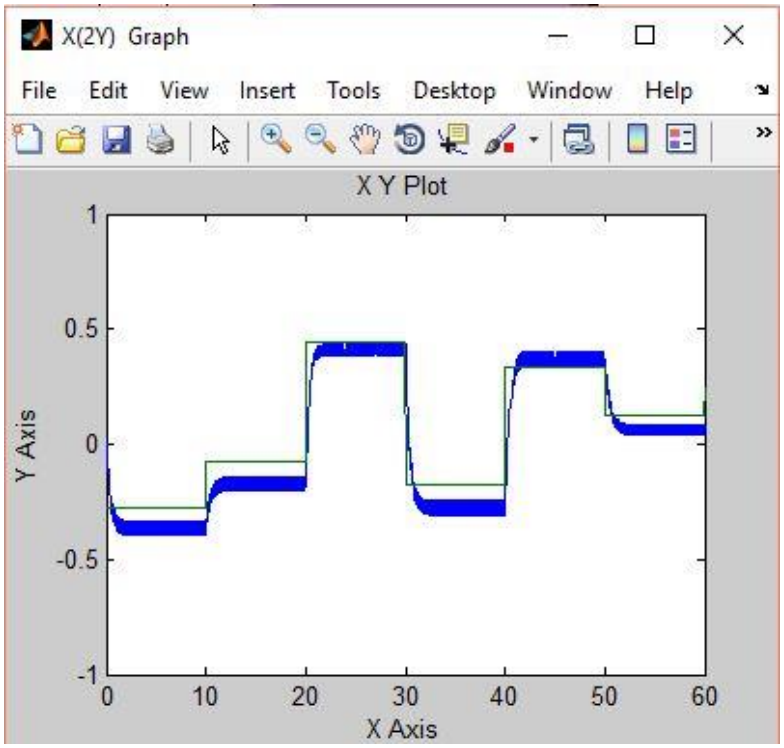


Figure 21: Plant Output Vs. Reference Model O/P

Thus we can see the performance of the neural based adaptive control is optimal for this system.

Chapter – 6

CONCLUSION

As we have seen in this report that the adaptive control is very useful in controlling the non-linear plant. If even the plant, control law and model are linear, the whole MRAC scheme is absolutely nonlinear because of the variation of the gains of controller with time. From theory of adaptive, the gains of controller are best adjusted.

Neural Network has many specific characteristics like nonlinear function approximation and ability to combine bulk of data to draw pattern and inferences are applicable in control and are very helpful in adaptive control. In particular, neural based technology offers much more advantages in the area of non-linear direct model reference adaptive control (MRAC)

Adaptive Control tackles complex systems whose parameter deviations are unpredictable and uncertain and the main objective is to keep system performance consistent in the uncertain ambience and variations in parameters of plant. Adaptive control is superior to robust control while dealing with uncertainties in slow-varying or constant parameters. Vigorous control has focal points in managing unsettling influences, unmodeled elements and rapidly differing parameters.

REFERENCES

1) http://www.pages.drexel.edu
2) http://en.wikibooks.org/wiki/Artificial_Neural_Networks/Feed-Forward_Networks
3) http://in.mathworks.com/help/nnet/ug/design-model-reference-neural-controller-in-simulink.html?s_tid=srchtitle
4) http://in.mathworks.com/help/nnet/ug/design-model-reference-neural-controller-in-simulink.html?s_tid=srchtitle
5) G.Lightbody, G.W. Irwin, 'Direct neural model reference adaptive control' ;IEE Proc.-Control Theory Appl., Vol. 142, No. I , January 1995
6) K. S. Narendra, Y. H. Lin, and L. S. Valavani. Stable adaptive controller design, part ii: Proof of stability. Technical report, Yale University, April 1979.
7) I. Kanellak Opoulos, P. V. Kokotovic, and A. S. Morse. Systematic design of adaptive controllers for feedback linearizable systems. IEEE Transactions on Automatic Control, 36(11):1241-1253, November 1991.
8) GUEZ, A., EILBERT, J.L., and KAM, M.: 'Neural network architecture for control', IEEE Contr. Sys. Mag., April 1988, pp. 22-25
9) M. T. Hagan, H. D. Demouth, "An introduction to the use of neural networks in control systems", International Journal of Robust and Nonlinear Control 12(11):959 - 985 · September 2002