ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Neural Network Based Direct Model Reference Adaptive Control Technique For Improving Tracking Performance in Nonlinear Systems.

By:

Alemie Assefa

Thesis Submitted To Addis Ababa Institute of Technology in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical and Computer Engineering (Control Engineering)

Advisor:

Dr. Dereje Shiferaw

July 2019

Addis Ababa, Ethiopia

Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Thesis Submitted To Addis Ababa Institute of Technology in Partial Fulfillment of the Requirement for the Degree of Master of Science in Electrical and Computer Engineering (Control Engineering)

By:
Alemie Assefa

Approval by Board of Examiners

_____	_____
Chairman Department of Graduate Committee	Signature
Dr.Dereje Shiferaw	_____
Advisor's Name	Signature
_____	_____
Internal Examiner	Signature
_____	_____
External Examiner	Signature

Declaration

I, the undersigned, declared that this MSc thesis is my original work, has not been presented for fulfillment of a degree in this or any other University and all sources and materials used for the thesis is acknowledged.

Alemie Assefa _____

Name Signature

Addis Ababa _____

Place Date of submission

This thesis work has been submitted for examination with my approval as a University Advisor.

Dr.Dereje Shiferaw _____

Advisor's Name Signature

Acknowledgments

First, I would like to thank almighty GOD for his presence with me; I have necessarily completed this final thesis. Secondly, I want to start expressing a sincere acknowledgment to my advisor, Dr. Dereje Shiferaw for giving me the opportunity to research under his guidance and supervision. I received motivation, comments, encouragement and continuous guidance from him during my graduate studies.

Abstract

This thesis investigates the application of a neural network based model reference adaptive intelligent controller for controlling of the nonlinear systems. In this scheme, the intelligent supervisory loop is incorporated into the conventional model reference adaptive controller framework by utilizing an online growing multilayer back propagation neural network structure in parallel with it. The idea is to control the plant by minimizing the tracking error between the desired reference model and the nonlinear system using conventional model reference adaptive controller by estimating the adaptation law using a multilayer back propagation neural network.

In the conventional model reference adaptive controller (MRAC) scheme, the controller is designed to realize the plant output converges to reference model output based on the plant, which is linear. This scheme is effective for controlling the linear plant with unknown parameters. However, using MRAC to control the nonlinear system in real time is difficult.

The Neural Network is used to compensate the nonlinearity of the plant that is not taken into consideration in the conventional MRAC. The proposed neural network based model reference adaptive controller can significantly improve the system behavior and force the system to follow the reference model and minimize the error between the model and the plant output.

Adaptive law using Lyapunov stability criteria for updating the controller parameters online has been formulated.

The effectiveness of the proposed control scheme is verified by developing the simulation results for simple pendulum and Vander poll oscillation as a benchmark study in MATLAB/SIMULINK software. It is observed from the simulation results that the proposed neural network based Direct MRAC has 3.13sec rise time, 5.15sec settling time for 0.1rad disturbance and 3.12sec rise time, 5.21sec settling time for 0.2rad disturbance. Whereas, the conventional direct model reference adaptive control has 5.42sec rise time, 15.5sec settling time for 0.1rad disturbance and 5.01sec rise time, 15.52sec settling time for 0.2rad disturbance**.**

It is shown that the proposed neural network based Direct MRAC has small rising time, steady-state error and settling time for a different disturbance than Conventional DMRAC adaptive control**.**

Keywords: Model reference adaptive control (MRAC), Artificial Neural Network (ANN), Multilayer Backpropagation Neural Network.

Table of Contents page

List of Figures

List of Tables

List of Abbreviations

NN	Neural network
ANN	Artificial neural network
MRAC	Model reference adaptive control
DMRAC	Direct model reference adaptive control
CMRAC	Conventional model reference adaptive control
NNMRAC	Neural Network based model reference adaptive control
MLP	Multilayer Perceptron
RBF	Radial basis function
BPA	Back Propagation algorithm
SMC	Sliding mode control
LTI	Linear Time Invariant system
IP	Inverted Pendulum
VPO	Vander poll oscillator
BPNNs	Backpropagation Neural Networks

List of Symbols

ω_{ij}	Weight of neural network
δ_i	Error gradient of the neuron on the ith layer of NN
τ	Torque in(N.m)
θ_{est}	Adaptive Parameter estimation
φ	Bounded Basis function
e	Error between model reference and nonlinear plant
γ, Γ_θ	Learning rates
ζ	Damping ratio
θ	Angle of the pendulum
Λ	Control input uncertainty
ω_n	Damping frequency
I	Inertia of the bob($kg.m^2$)

Chapter One

1. Introduction

1.1. Background of the study

The design of autopilots for high-performance aircraft was one of the primary motivations for active research in adaptive control in the early 1950s. Aircraft operate over a wide range of speeds and altitudes, and their dynamics are nonlinear and conceptually time varying. For a given operating point, specified by the aircraft speed and altitude, a linear model can approximate the longitudinal nonlinear aircraft dynamics. As the aircraft goes through different flight conditions, the operating point changes. These changes cannot be handled by constant gain feedback control. Since the output response $y(t)$ carries information about the state as well as the parameters, one may argue that in principle, a sophisticated feedback controller should be able to learn about the plant changes by processing the input/output (I/O) measurements and choosing the appropriate controller from a list or design a new one in real-time [1].

In the adaptive control, the question of control of the nonlinear system with present-day sophistication and complexities has often been an important research area due to the difficulty in modeling, nonlinearities, and uncertainties. Model reference adaptive control is one of the main schemes used in the adaptive system. Recently MRAC has received considerable attention and many new approaches have been applied to the practical process. [2]

In the MRAC scheme, the controller is designed to realize the plant output converges to reference model output based on assumption that plant can be linearized [3], [4], and [5]. Therefore, this scheme is effective for controlling linear plants with unknown parameters. However, it may not assure for controlling nonlinear plants with unknown structure. In recent years, an artificial neural network (ANN) has become very popular in many control applications due to their higher computation rate and ability to handle nonlinear systems [6].

1.1.1. The Basics of Neural Networks

The simplest definition of a neural network, more properly referred to as an 'artificial' neural network (ANN), is provided by the inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen. He defined a neural network as a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs. ANNs are processing devices (algorithms or actual hardware) that are

loosely modeled after the neuronal structure of the mammalian cerebral cortex but on much smaller scales. A large ANN might have hundreds or thousands of processor units, whereas a mammalian brain has billions of neurons with a corresponding increase in the magnitude of their overall interaction and emergent behavior. Although ANN researchers are generally not concerned with whether their networks accurately resemble biological systems, some have. For example, researchers have accurately simulated the function of the retina and modeled the eye rather well.

Neural networks are typically organized in layers. Layers are made up of a number of interconnected nodes, which contain an activation function. Patterns are presented to the network via the input layer, which communicates to one or more hidden layers where the actual processing is done via a system of weighted connections. The hidden layers then link to an output layer where the answer is output as shown in the graphic below.

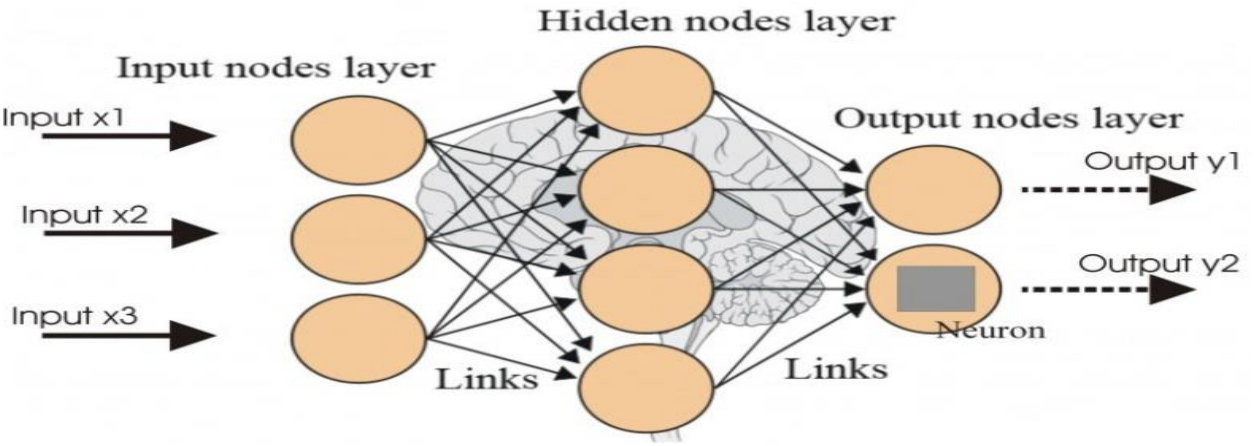


Figure 1. 1 Layers of ANN

Most ANNs contain some form of learning rule, which modifies the weights of the connections according to the input patterns. In a sense, ANNs learn by example as do their biological counterparts; a child learns to recognize dogs from examples of dogs. Although there are many different kinds of learning rules used by neural networks, this demonstration is concerned only with one; the delta rule. The most common class of ANNs called backpropagation neural networks (BPNNs) often utilizes the delta rule. Backpropagation is an abbreviation for the backward propagation of error. With the delta rule, as with other types of backpropagation, learning is a supervised process that occurs with each cycle or epoch (i.e. each time the network is presented with a new input pattern) through a forward activation flow of outputs, and the backward error propagation of weight adjustments. [7]

1.2. Statement of the problem

In the conventional model reference adaptive controller (MRAC) scheme, the controller is designed to realize plant output converges to reference model output based on the plant, which is linear. This scheme is effective for controlling linear plants with unknown parameters.

However, the disturbance and nonlinear components are adding to the output on the plant of the conventional model reference adaptive controller in such a way that the tracking error has not come to zero and the output of the plant is not tracked with the reference model output.

In the industrial process, there are many systems having nonlinear properties like unknown or uncertain parameters and disturbances. Unless such parameter uncertainty and disturbance gradually reduced online by an appropriate adaptation or estimation mechanism it may cause inaccuracy or instability for the control systems.

To overcome the above problem and improve the system performance, a neural network-based model reference adaptive controller is proposed. The controller is designed by using a parallel combination of the conventional MRAC system and neural network.

1.3. Objective of the Thesis

1.3.1. General objective

The general objective of the thesis is to improve the tracking performance of nonlinear systems (Simple Pendulum and VP oscillation) using neural network based direct MRAC using MATLAB software.

1.3.2. Specific objective

- To develop and evaluate the performance of Neural network based direct Model Reference Adaptive Control System through transient analysis.
- To train the NN based on the error generated from the actual output and reference model.
- To compare the simulation results of the conventional MRAC and NN based Direct MRAC on different nonlinear systems (Simple pendulum and VP oscillation).
- To evaluate and analyze the performance of the controller.

1.4. Scope of the thesis

The scope of this project is to simulate the performance of nonlinear systems using MATLAB software and train the neural network to track the reference model to the plant output.

In addition to this, it is developed to compare the effectiveness of the proposed controller to improve tracking performance and modeling of nonlinear systems (Pendulum and VPO) using the neural network base direct model reference adaptive controller in MATLAB software.

1.5. Methodology

The following methodologies have been used for the accomplishment of this thesis:

- ✓ Literature Survey: Different works of literatures that are related to this thesis work are studied and different concepts are adopted.
- ✓ Adaptive controller design using Lyapunov stability criteria and mathematical modeling for a nonlinear system for both simple pendulum and Van der pol oscillation.
- ✓ Controller design: non-linear controller of neural network based model reference adaptive control has been designed which is robust and insensitive to external disturbances and parameter uncertainties.
- ✓ MATLAB/Simulink Simulation has been utilized for comparison of tracking performance for both conventional model reference adaptive control and neural network based model reference adaptive control of nonlinear systems (Simple pendulum and Vander poll oscillators).
- ✓ Neural network based model reference adaptive control has been tested subjected to different effects to ensure the robustness, parameter uncertainty and insensitive to external disturbance while tracking the desired reference signal.

1.6. The Outlines of the thesis

The thesis is organized into five chapters including the introduction in chapter 1. The rest of the thesis is organized as follows.

Chapter 2 describes the literature review and theoretical analysis of both neural network and adaptive control.

Chapter 3 in this chapter mathematical modeling of the nonlinear system and adaptive control design using Lyapunov stability criteria is described.

Chapter 4 this chapter describes the simulation of nonlinear systems (Simple pendulum and VPO) using conventional MRAC technique and neural network based Direct MRAC and compares the simulation result on MATLAB/SIMULINK. Including in this chapter, Simulation results based on disturbance and unknown parameter variation are discussed.

Chapter 5 in this chapter draws the conclusion from the work done in this thesis and recommend further research.

Chapter Two

2. Literature Review

This chapter establishes the essential theory and critical assessment of the related work needed for the development of the research and it presents the main findings by different researchers of intelligence systems and how they can be used to regulate the nonlinear system especially for simple pendulum and Vander, poll oscillation with aid of Model reference adaptive control system. The related literature and similar studies mentioned in this study substantiate the arguments that support the theories and assumptions. The information gathered in this chapter provides compelling motivation to pursue the study due to the Benchmark information and baseline research that reflect meritorious works and scholarly studies conducted by experts that are relevant to the neural network based direct model reference, adaptive controller.

Parks.P.C, Hang.C.C, Kalpesh.B [8], [9], [3] the controller is designed to realize a plant output converges to reference model output based on assumption that the plant can be linearized. The linear MRAC performs well when it is working around the operating point, where a linear model can approximate the plant. Therefore, this scheme is effective for controlling a linear plant with unknown parameters in the ideal case. However, as most industrial processes are highly nonlinear, non-minimum, and with various type of uncertainties and load disturbances the performance of the linear MRAC may deteriorate, and suitable nonlinear control may have to be used.

Swarnkar, Pankaj, Shailendra Jain [10] the effect of adaptation gain for a second order system using the MIT rule is analyzed. Based on the results in this article, an interesting topic for further research would be to discuss second order systems using both MIT rule and Lyapunov rule and focus on changes to reference model parameters.

Zhihong, Man, X. H. Yu, K. Eshraghian [11], the output of neural networks then adaptively adjusts the gain of the sliding mode controller so that the effects of system uncertainties eliminated and the output tracking error between the plant output and the desired reference signal can be asymptotically converging to zero. However, the sliding mode control action can lead to high frequency oscillations called chattering which may excite unmodeled dynamics, energy loss, and system instability and sometimes it may lead to plant damage. Therefore, the drawback of SMC is that unavoidable chattering occurs when the control signal switches sign along the sliding surface.

Hunt, K. Jetal, D.Sbarbaro, R. Żbikowski [12], the recent development of the neural network based

control system for suitable choosing neural network structures, training methods, and sufficient past input and output data of the neural network can be well trained to learn the system forward dynamics to predict the future behavior of the system for the predictive control and model following control or to learn the inverse dynamics for inverse control. However, the stability, error convergence and robustness have not been fully proved for these off-line trained neural network based control system because of the high nonlinearity of the neural network and the lack of feedback.

Zhihong, Man, Hong Ren Wu [13] A Radial basis function(RBF) neural network based adaptive controller have been developed for the compensation for the effects of nonlinearities and system uncertainties in control system so that the system performance such as the stability, convergence, and robustness can be improved. However, once training is completed, an RBF network may be slower to use than a feedforward backpropagation network since more computations are required to arrive at an output.

Munadi, M. Amirullah Akbar [14] presents the application of adaptive control of for a DC motor. The uncertainties include parametric variations in the time-varying disturbance. We chose to use MRAC with the MIT rule method for the adaptation mechanism and estimate the controller parameter to match the reference model. The purpose of this paper is to obtain an adaptive control design that is resistant to disturbance by using Simulink for modeling the dynamics of the system so that it can easily obtain a solution of modeling without having too many calculations of the complex mathematical equations.

K. Pirabakaran and V.M. Becerra [15] describe the application of artificial neural networks for automatic tuning of PID controllers using the Model Reference Adaptive Control (MRAC) approach. The approach is first construct a plant emulator, using a multi-layer perceptron (MLP) network. This emulator is then used together with an on-line trained neural network, which adjusts the PID parameters such that the error between the reference model and process output is reduced. The neural network tuner takes past plant output values, control signal and set point signal as inputs, and produces the required changes in the PID controller parameters.

2.1. Adaptive control

Adaptive control is the control method used by a controller, which must adapt to a controlled system with parameters, which vary or are initially uncertain. For example, as an aircraft flies, its mass will slowly decrease because of fuel consumption; a control law is needed that adapts itself to such changing conditions. Adaptive control is different from robust control in that it does not need a priori information about the bounds on these uncertain or time-varying parameters; robust control guarantees that if the changes are within given bounds the control law need not be changed, while adaptive control is concerned with control law changing itself.

This control is a special kind of non-linear control, and the process can be split into two timelines: rapid time (feedback loop) and slow time (variation of control parameters, which affects to automation).

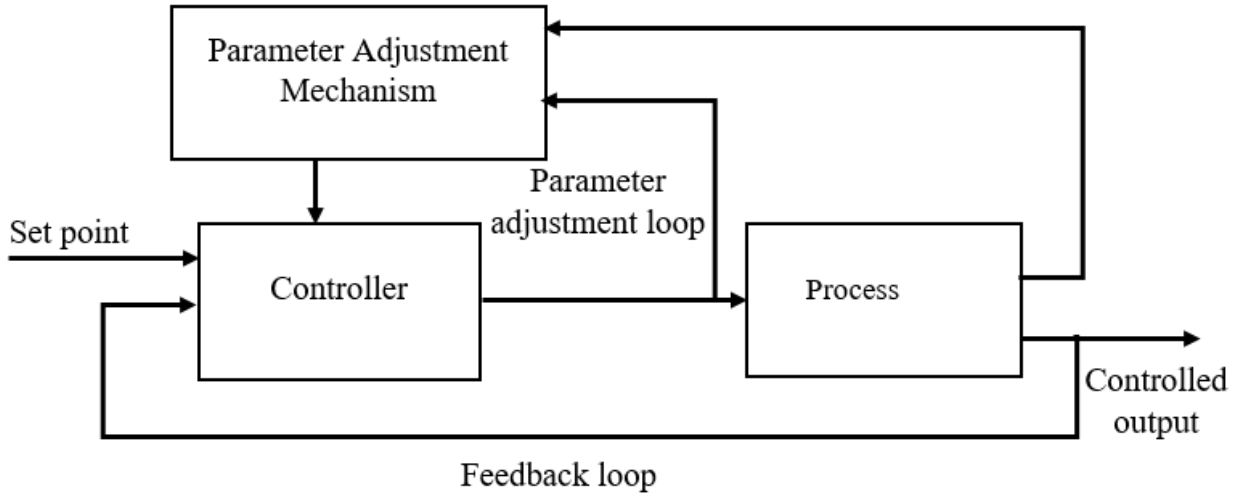


Figure 2.1 Block diagram of Adaptive control.

The foundation of adaptive control is parameter estimation, which is a branch of system identification. Common methods of estimation include recursive least squares and gradient descent. Both of these methods provide update laws that are used to modify estimates in real time (i.e., as the system operates). Lyapunov stability is used to derive these update laws and show convergence criteria (typically persistent excitation; relaxation of this condition are studied in Concurrent Learning adaptive control), Projection (mathematics) and normalization are commonly used to improve the robustness of estimation algorithms.

2.1.1 Model-Reference Adaptive Control

Model reference adaptive control is one of adaptive control classification technique. When designing a controller for a system, a control designer typically would like to know how the system behaves physically. This knowledge is usually captured in the form of a mathematical model. For many real-world applications, modeling of physical systems can never be perfect as systems may have parameter variations due to nonlinearity, parameter uncertainty due to modeling inaccuracy or imprecise measurements, uncertainty in exogenous disturbances coming from the operating environment, or other sources of uncertainty. The role of a modeling specialist is to reduce the system uncertainty as much as practicable [16], [2]. The control designer then uses a mathematical model of the system to design a controller, which may incorporate performance measures and stability margins to account for any residual system uncertainty that cannot be completely accounted for. In situations when the system uncertainty may become significant beyond a level of desired tolerance that can adversely affect the performance of a controller, adaptive control can play an important role in reducing the effects of the system uncertainty on the controller performance.

There are generally two classes of adaptive control: direct adaptive control and indirect adaptive controller [1]. Direct adaptive controller methods adjust the control gains online directly and indirect adaptive controller methods estimate unknown system parameters for use in the update of the control gains. Asymptotic tracking is the fundamental property of model-reference adaptive control, which guarantees that the tracking error tends to zero in the limit.

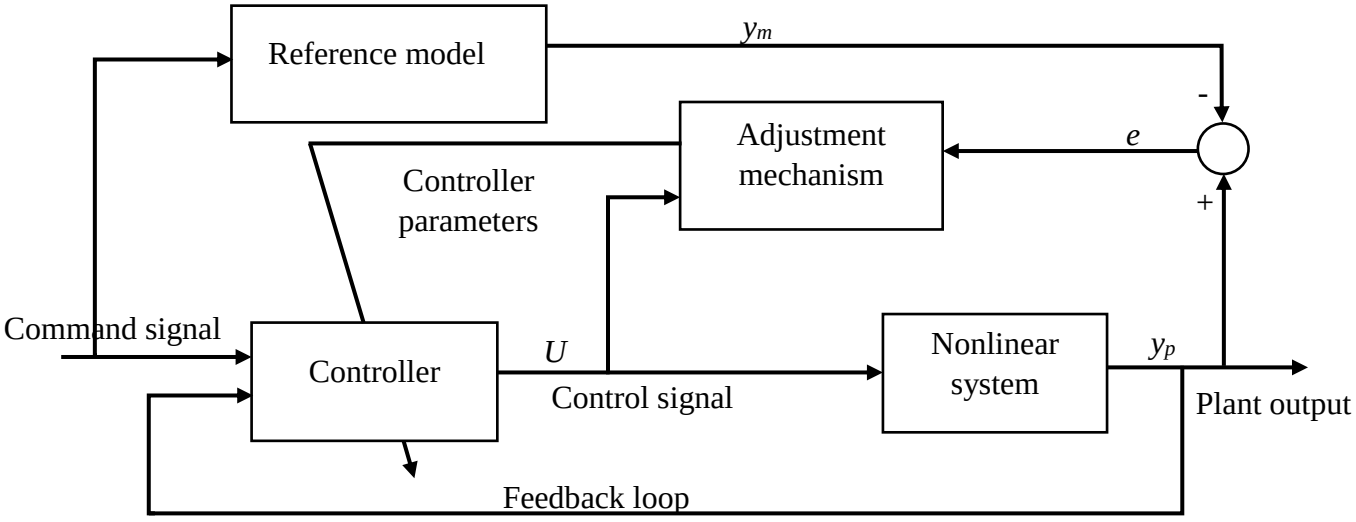


Figure 2. 2 Block diagram of typical MRAC.

2.2. Introduction to Neural network

Neural networks and deep learning are big topics in Computer Science and in the technology industry; they currently provide the best solutions to many problems in image recognition, speech recognition, and natural language processing. The inventor of one of the first neurocomputers, Dr. Robert Hecht-Nielsen, provides a neural network, more properly referred to as an 'artificial' neural network (ANN). He defines a neural network as a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs. On the other hand, you can also think of Artificial Neural Network as a computational model that is inspired by the way biological neural networks in the human brain process information. [19]- [22]

The basic computational unit of the brain is a neuron. Approximately 86 billion neurons can be found in the human nervous system and they are connected with approximately 10^{14}—10^{15} synapses. The diagram below shows a cartoon drawing of a biological neuron.

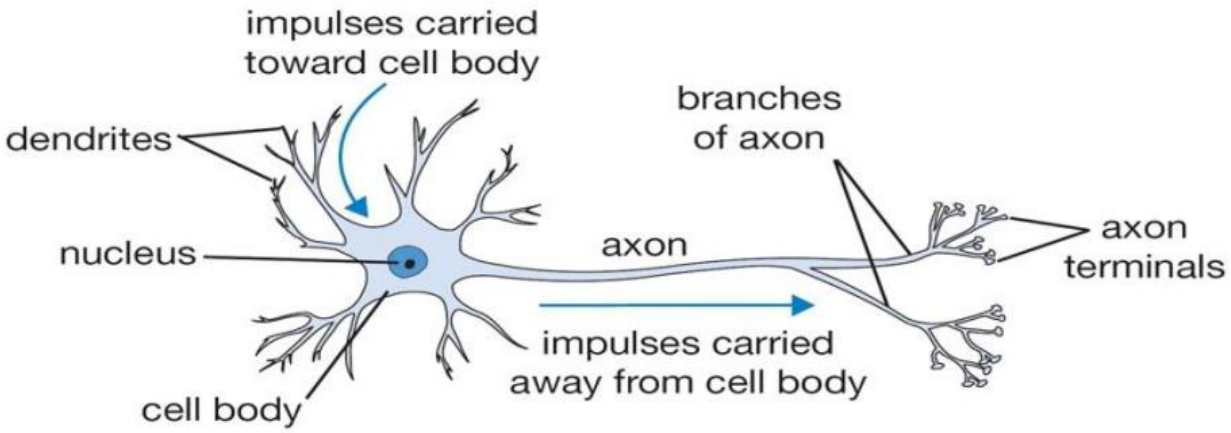


Figure 2.3 Biological neuron model

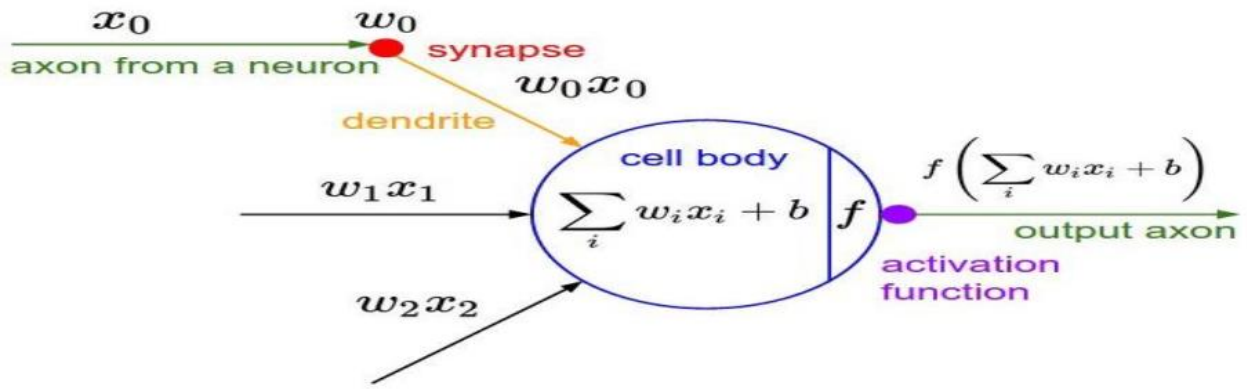


Figure 2.4 Mathematical model for a common biological neuron.

The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes or from an external source and computes an output. Each input has an associated weight (w), which is assigned based on its relative importance to other inputs. The node applies a function to the weighted sum of its inputs.

The idea is that the synaptic strengths (the weights *w*) are learnable and control the strength of influence and its direction: excitory (positive weight) or inhibitory (negative weight) of one neuron on another. In the basic model, the dendrites carry the signal to the cell body where they all are summed. If the final sum is above a certain threshold, the neuron can fire, sending a spike along its axon. In the computational model, we assume that the precise timings of the spikes do not matter and that only the frequency of the firing communicates information. We model the firing rate of the neuron with an activation function *(e.g. sigmoid function)*, which represents the frequency of the spikes along the axon.[7],[12].

Neural Network Architecture

From the above explanation we can conclude that a neural network is made of neurons, biologically the neurons are connected through synapses where information's flows (weights for out computational model), when we train a neural network we want the neurons to fire whenever they learn specific patterns from the data, and we model the fire rate using an activation function.

- Input Nodes (input layer): No computation is done here within this layer; they just pass the information to the next layer (hidden layer most of the time). A block of nodes is also called layer**.**
- Hidden nodes (hidden layer): In hidden layers is where intermediate processing or computation is done, they perform computations and then transfer the weights (signals or

information) from the input layer to the following layer (another hidden layer or to the output layer). It is possible to have a neural network without a hidden layer.

- Output Nodes (output layer): Here we finally use an activation function that maps to the desired output format (e.g. softmax for classification).
- Connections and weights: The network consists of connections, each connection transferring the output of a neuron i to the input of a neuron *j*. In this sense *i* is the predecessor of *j* and *j* is the successor of *i*, each connection is assigned a weight *Wij*.
- Activation function: the activation function of a node defines the output of that node given an input or set of inputs. A standard computer chip circuit can be seen as a digital network of activation functions that can be "ON" (1) or "OFF" (0), depending on the input. This is similar to the behavior of the linear perceptron in neural networks. However, the nonlinear activation function allows such networks to compute nontrivial problems using only a small number of nodes. In artificial neural networks, this function is also called the transfer function.
- Learning rule: The learning rule is a rule or an algorithm, which modifies the parameters of the neural network, in order for a given input to the network to produce a favored output. This learning process typically amounts to modifying the weights and thresholds.

2.3. Types of Neural Networks

There are many classes of neural networks and these classes also have sub-classes, here are the most used ones and make things simple to move on in this journey to learn neural networks [20],[22].

2.3.1. Feedforward Neural Network

A feedforward neural network is an artificial neural network where connections between the units do not form a cycle. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.

We can distinguish two types of feedforward neural networks:

2.3.1.1. Single-layer Perceptron: This is the simplest feedforward neural Network and does not contain any hidden layer, which means it only consists of a single layer of output nodes. This is said to be single because when we count the layers we do not include the input layer, the reason for this is that at the input layer no computations are done, the inputs are fed directly to the outputs via a series of weights.

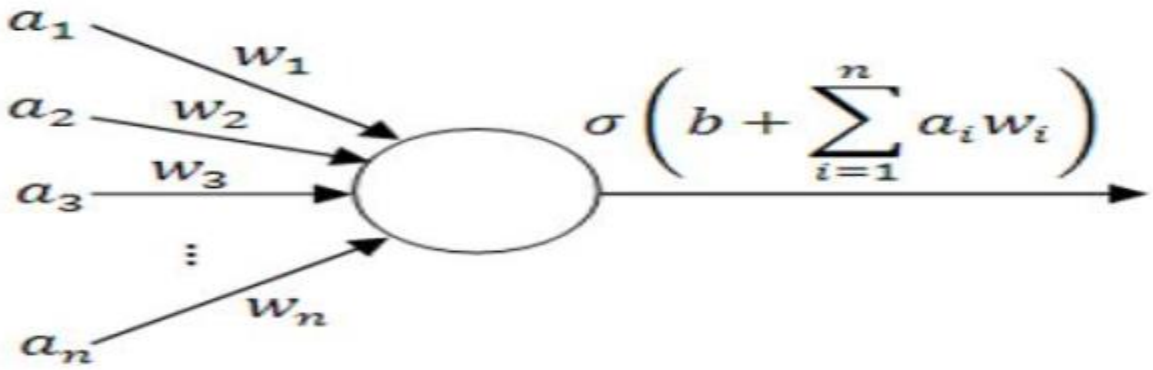


Figure 2.5 Single layer perceptron

2.3.1.2. Multi-layer perceptron (MLP)

This class of networks consists of multiple layers of computational units, usually interconnected in a feed-forward way. Each neuron in one layer has directed connections to the neurons of the subsequent layer. In many applications, the units of these networks apply a sigmoid function as an activation function. MLP is very more useful and one good reason is that they are able to learn non-linear representations (most of the cases the data presented to us are not linearly separable).

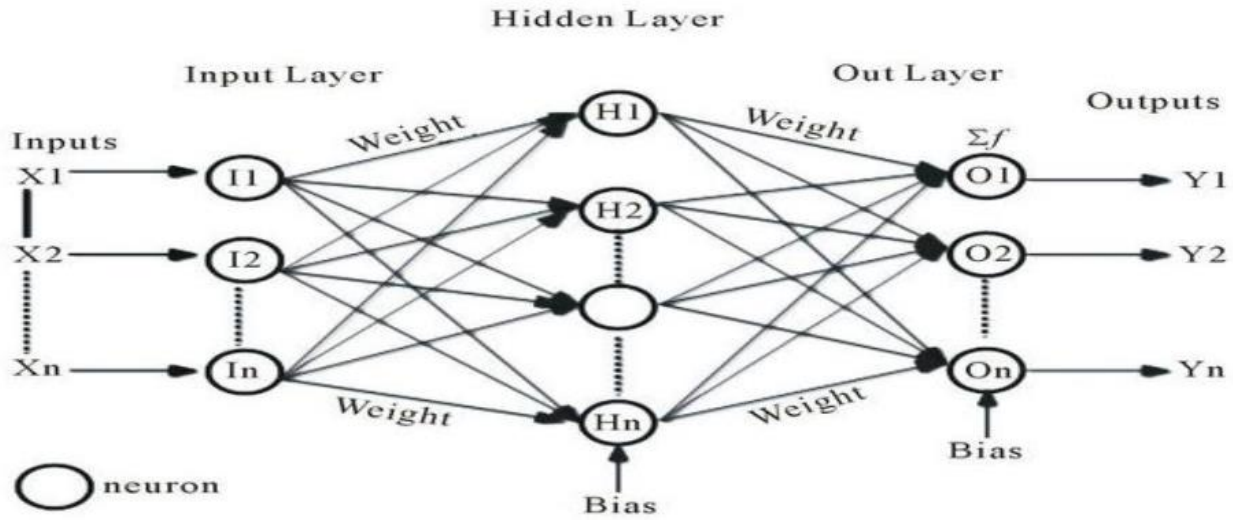


Figure 2.6 Multi-layer perceptron model

2.3.2. Recurrent Neural Networks

In a recurrent neural network (RNN), connections between units form a directed cycle (they propagate data forward, but also backward, from later processing stages to earlier stages). This allows it to exhibit dynamic temporal behavior. Unlike feedforward neural networks, RNNs can use their internal memory to process arbitrary sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition, speech recognition, and other general sequence processors.

Models of Artificial Neural Networks

There are various Artificial Neural Network Model. Main ones are-

Multilayer Perceptron – It is a feedforward, artificial neural network model. It maps sets of input data onto a set of appropriate outputs.

Radial Basis Function Network – A radial basis function network is an artificial neural network. It uses radial basis functions as activation functions. Both of the above are being supervised learning networks used with one or more dependent variables at the output.

Multilayer Perceptron

As we saw above, a multilayer perceptron is a feedforward, artificial neural network model. It maps sets of input data onto a set of appropriate outputs. In feed-forward neural networks, the movement is only possible in the feed-forward direction. MLP consists of many layers of nodes in a directed graph, with each layer connected to the next one. Each neuron is a linear equation like linear regression as shown in the following equation.

$$y_i = w_0 + w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n \quad (2.1)$$

The equation is the transfer function in a neural network. This linear weight sum would be a threshold at some value so that the output of neuron would be either 1 or 0.

The multilayer perceptron networks are suitable for the discovery of complex nonlinear models. On the possibility of approximating any regular function with a sum of sigmoid its power based. MLP utilizes a supervised learning technique called backpropagation for training the network. This requires a known, desired output for each input value to calculate the loss function gradient. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

Radial Basis Function Network

A Radial Basis Function (RBF) network is a supervised learning network like MLP, which it resembles in some ways. However, the RBF network works with only one hidden layer. It accomplishes this by calculating the value of each unit in the hidden layer for an observation. It uses the distance in space between this observation and the center of the unit.

Learning of RBF involves determining the number of units in the hidden layer. Like a number of radial functions, their centers, radii, and coefficients.

Comparison of MLP and RBF Networks

In neural networks, the approximating models relating inputs and outputs are **"**black box**"** models. They also provide very little insight into what these models do. In addition, neural network users must make many modeling assumptions. For example, the number of hidden layers and the number of units in each hidden layer, and usually, there is little guidance on how to do this. Thus, it takes considerable experience to determine the most appropriate representation. Furthermore, back-propagation can be quite slow if the learning constant is not in the correct form. MLPs and RBF networks are the two most common types of a feedforward network. They have much more in common than most of the NN literature would suggest. The only difference is the way in which hidden units combine values coming from preceding layers in the network An MLP has one or more hidden layers for which the combination function is the inner product of the inputs and weights, plus a bias. The activation function is usually a logistic or tanh function. MLP provides better generalization as it has the number of hidden units while RBF has less risk of non-optimal convergence. MLP is faster in model application mode while RBF is faster in model learningmode.

RBF networks usually have only one hidden layer. By which the combination function depends on the Euclidean distance between the input vector and the weight vector. RBF networks do not have anything that is the same as the bias term in an MLP. However, some types of RBFs have a "width" associated with each hidden unit or with the entire hidden layer. Instead of adding it in the combination function as a bias, you divide the Euclidean distance by the width.

2.4. Types of neural network Learning

The Common learning algorithms for the neural network are: [22], [23]

Supervised learning: In this, we can present the computer with example inputs and their desired outputs, given by a "teacher". Its goal is to learn a general rule that maps inputs to outputs. Spam filtering is an example of supervised learning. In particular, classification, the learning algorithm is present with email messages labeled as "spam" or "not spam". This is to produce a computer program that labels unseen messages as either spam or not. The classification problem is another standard formulation of the supervised learning task. Here the learner needs to learn a function that maps a vector into one of the several classes. This he can do by looking at several input-output examples of the function.

Unsupervised Learning: In this, no labels are given to the learning algorithm, leaving it on its own to groups of similar inputs (clustering), density estimates or projections of high-dimensional data that can be visualized effectively. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end. Topic modeling is an example of unsupervised learning, where a program is given a list of human language documents and is tasked to find out which documents cover similar topics. Here, learning takes places by detecting regularities in input data and developing patterns based on these deductions. Regularities are observed for the repetitive occurrence of a pattern. The more frequently occurring pattern is used to make predictions. This approach is also called the density estimation approach. Several methods like clustering can be used for density estimation.

Reinforcement Learning: In this, a computer program interacts with a dynamic environment. In this, it must perform a certain goal, without a teacher explicit telling it whether it has come close to its goal or not. Let us consider the case of robotic navigation. A robot can make very precise movements to perform a task. Yet, the robot has to learn to perform these movements through repeated tests. It applies the knowledge gained from this to improve its efficiency. This is the basis of reinforced learning. In robotic navigation, and other similar systems, such as a self-driving car, sensor doors the output is not restricted to a single action. It may contain a sequence of actions.

Workflow for Neural Network Design

To implement a Neural Network (design process), the following steps must be followed [19].

- Collect data (Load data source).
- Neural Network creation.
- Configure the network (selection of network architecture).
- Initialize the weights and biases.
- Train the network.
- Validate the network (Testing and Performance evaluation).
- Use the network.

Multilayer perceptron networks procedure steps using MATLAB: [19]

- The structure of the network is first defined, activation functions are chosen and weights and biases are initialized.
- The training algorithm parameter like error goal, the maximum number of epochs (iterations), etc., are defined.
- Run the training algorithm.
- Simulate the output of the neural network with the measured input data. This is compared with the measured outputs.
- Final validation must be carried out with independent data.

2.5. Multilayer neural networks (Back-Propagation training algorithm)

The input signals are propagated in a forward direction on a layer-by-layer basis. Learning in a multilayer network proceeds the same way as for a perception. A training set of input patterns is presented to the network and the network computes its output pattern, and if there is an error or in other words, a difference between actual and desired, output patterns the weights are adjusted to reduce this error.

In a back-propagation neural network, the learning algorithm has two phases.

First, a training input pattern is presented to the network input layer. The network propagates the input pattern from layer to layer until the output layer generates the output pattern.

Second, if this pattern is different from the desired output, an error is calculated and then propagated backward through the network from the output layer to the input layer. The weights are modified as the error is propagated.

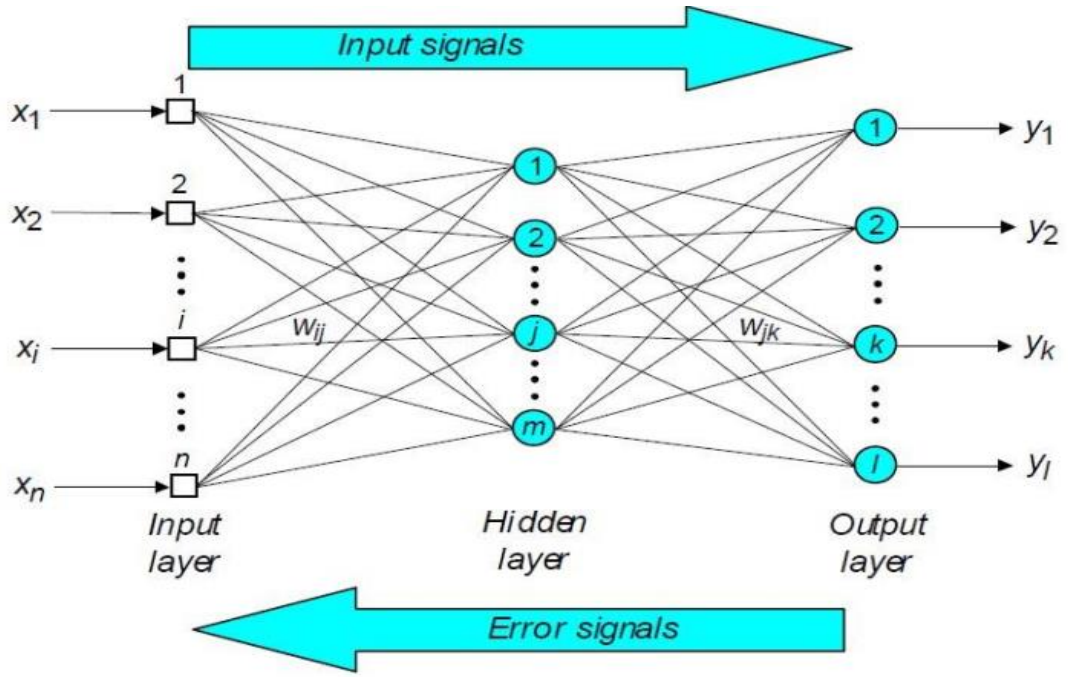


Figure 2.7 Back-propagation neural network

2.6. The back-propagation training algorithm

Backpropagation is a common method for training a neural network, the goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs. [19]

Backpropagation method contains the following steps:

Step 1: Initialization; set all the weights and threshold levels of the network to random numbers uniformly distributed inside a range:

$$\left(-\frac{2.4}{Fi}, \quad +\frac{2.4}{Fi}\right) \tag{2.2}$$

Where Fi is the total number of inputs of neuron $\boldsymbol{i}$ in the network.

Step2: Activation; activate the back-propagation neural network by applying inputs $x1(p), x2(p), \ldots, xn(p)$ and desired outputs $y_{d1}(p), y_{d2}(p), \ldots, y_{dn}(p)$ (forward pass).

Calculate the actual outputs of the neurons in the hidden layer:

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i\,(p).w_{ij}(p) - \theta_j\right] \tag{2.3}$$

Where **n** is the number of inputs of neuron **j** in the hidden layer.

Calculate the actual outputs of the neurons in the output layer:

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}\,(p).w_{jk}(p) - \theta_k\right] \tag{2.4}$$

Where **m** is the number of inputs of neuron **k** in the output layer

Step 3: Weight training (back-propagate):-Update the weights in the back-propagation network propagating backward the errors associated with output neurons.

Calculate the error gradient for the neurons in the output layer:

$$\delta_k(p) = y_k(p).[1 - y_k(p)].e_k(p) \tag{2.5}$$

Where, $e_k(p) = y_{d,k}(p) - y_k(p)$

Calculate the weight corrections:

$$\Delta w_{jk}(p) = \alpha.y_j(p).\delta_k(p) \tag{2.6}$$

Update the weight at the output neurons:

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \tag{2.7}$$

Calculate the error gradient for the neurons in the hidden layer

$$\delta_j(p) = y_j(p).\left[1 - y_j(p)\right].\sum_{k=1}^{l} \delta_k(p)\,w_{jk}(p) \tag{2.8}$$

Calculate the weight corrections:

$$w_{ij}(p) = \alpha.x_i(p).\delta_j(p) \tag{2.9}$$

Update the weights at the hidden neurons:

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \tag{2.10}$$

Step 4: Iteration:-Increase iteration **p** by one, go back to **Step 2** and repeat the process until the selected error criterion is satisfied.

Chapter Three

3. Mathematical Modeling of Nonlinear Systems and Adaptive Controller Design.

3.1. Mathematical modeling of nonlinear systems

A mathematical model of a dynamic system is defined as a set of equations that represents the dynamics of the system accurately or, at least fairly well. A mathematical model is not unique to a given system. The dynamics of many systems may be described in terms of differential equations. Such differential equations may be possibly obtained by using physical laws governing a particular system. Once a mathematical model of a system is obtained, various analytical and computational tools can be used for analysis and synthesis purposes. [24]

Inverted pendulum and Vander poll oscillation are non-linear and unstable. To control the limitations of different researchers, investigate feedback controller based on Taylor series linearization techniques and nonlinear controllers based on nonlinear methods. A system that is not linear is called a nonlinear system. All physical systems are nonlinear so many researchers and designers' study nonlinear control system. The Taylor series linearization technique has been used to design control laws for Inverted pendulum and Vander poll oscillation systems, Taylor series linearization method is used to approximate the nonlinear system by a linear one at nominal operating points; this is used to determine the behavior of the nonlinear system by studying the linear equivalent

3.1.1. Mathematical Modeling of Simple Pendulum System

Consider the simple pendulum shown in Figure 3.1, where l denotes the length of the rod and m denotes the mass of the bob. Assume the rod is rigid and has zero mass. Let θ denote the angle subtended by the rod and the vertical axis through the pivot point. The pendulum is free to swing in the vertical plane. The bob of the pendulum moves in a circle of radius l. To write the equation of motion of the pendulum, let us identify the forces acting on the bob. There is a downward gravitational force equal to mg, where g is the acceleration due to gravity. There is also a frictional force resisting the motion (by air and any other frictions), which we assume to be proportional to the speed of the bob with a coefficient of friction b.

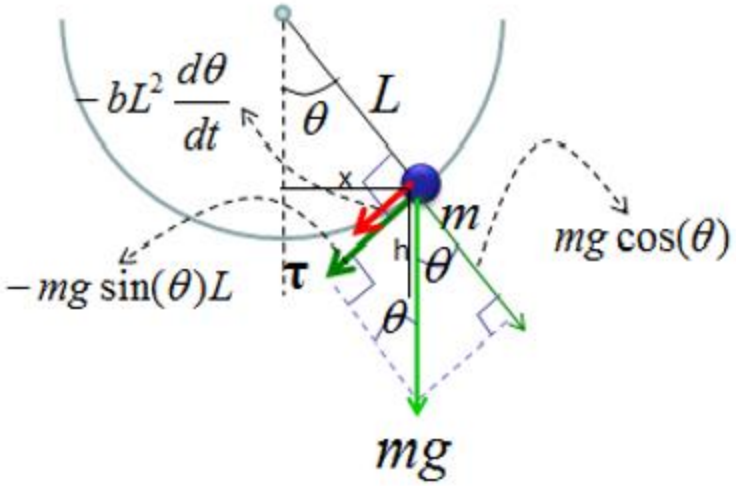


Figure 3. 1 Simple pendulum modeling [31]

$$x = lsin\theta, h = l(1 - cos\theta) \tag{3.1}$$

$$K.E = \frac{1}{2}I\omega^2 = \frac{1}{2}mv^2 \tag{3.2}$$

$$\omega = \frac{v}{l}, I = ml^2, v^2 = v_x^2 + v_y^2 \tag{3.3}$$

$$v_x = \frac{dx}{dt}, v_y = \frac{dy}{dt}, v_x^2 = \left(\frac{dx}{dt}\right)^2, v_y^2 = \left(\frac{dy}{dt}\right)^2 \tag{3.4}$$

Substituting equation (3.1) to equation (3.4)

$$v_x^2 = \left(\frac{dx}{dt}\right)^2 = \left(l\dot{\theta}cos\theta\right)^2, v_y^2 = \left(\frac{dy}{dt}\right)^2 = \left(l\dot{\theta}sin\theta\right)^2 \tag{3.5}$$

$$K.E = \frac{1}{2}m\left(v_x^2 + v_y^2\right) = \frac{1}{2}m(\left(l\dot{\theta}cos\theta\right)^2 + \left(l\dot{\theta}sin\theta\right)^2) \tag{3.6}$$

$$Therefore, K.E = \frac{1}{2}m(\left(l\dot{\theta}\right)^2[cos^2\,\theta + sin^2\,\theta]) \tag{3.7}$$

$$K.E = \frac{1}{2}m(\left(l\dot{\theta}\right)^2), since, cos^2\,\theta + sin^2\,\theta = 1 \tag{3.8}$$

$$P.E = mgh, but\ h = l(1 - cos\theta), P.E = mgl(1 - cos\theta) \tag{3.9}$$

By using Lagrangian equation

$$L = K.E - P.E = \frac{1}{2}m(\left(l\dot{\theta}\right)^2 - mgl(1 - cos\theta)) \tag{3.10}$$

$$\frac{d}{dt}\left(\frac{\partial L}{\partial\dot{\theta}}\right) - \frac{\partial L}{\partial\theta} = \tau(torque) - b\dot{\theta} \tag{3.11}$$

$$\frac{\partial L}{\partial \dot{\theta}} = ml^2\dot{\theta}, \frac{\partial L}{\partial \theta} = mgl(-sin\theta) = -mglsin\theta, \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) = ml^2\ddot{\theta} \tag{3.12}$$

$$Therefore, ml^2\ddot{\theta} + mglsin\theta + b\dot{\theta} = \tau(torque) \tag{3.13}$$

The mathematical model for a simple pendulum is that

$$ml^2\ddot{\theta} + mglsin\theta + b\dot{\theta} = \tau(torque) \tag{3.14}$$

The state space form of the system is given by, Let $x_1 = \theta, x_2 = \dot{\theta} = \omega$

$$\dot{x}_1 = 0x_1 + x_2 \tag{3.15}$$
$$\dot{x}_2 = -\frac{b}{ml^2}x_2 + \frac{1}{ml^2}[\tau - mglsinx_1]$$

On the other hand, we can put in the form of

$$\left.\begin{aligned} \dot{x}_1 &= 0x_1 + x_2 \\ \dot{x_2} &= \frac{1}{ml^2}[\tau - mglsinx_1 - bx_2] \end{aligned}\right\} \tag{3.16}$$

$\dot{x} = Ax + B\,\Lambda\,[u - \theta^T\varphi(x)]$, Λ control input uncertainty

Case I: A and Λ Unknown, but B and Sign of Λ Known.

Let the controller, $\tau = -K_x(t)x + k_r(t)r + \theta^T\varphi(x)$ be the adaptive controller, where$K_x(t)$,$k_r(t)$ and θ (t) are the estimates of Kx (t)*, kr (t)* and θ (t)* respectively.

$$\dot{x}_1 = 0x_1 + x_2 \tag{3.17}$$
$$\dot{x}_2 = -\frac{b}{ml^2}x_2 + \frac{1}{ml^2}[\tau - mglsinx_1]$$

$$\begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & \frac{-b}{ml^2}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \frac{1}{ml^2}\begin{bmatrix}0\\ 1\end{bmatrix}[\tau - mglsinx_1] \tag{3.18}$$

Case II: A, B and Λ=I Known.

Let the controller $\tau = -Kx\,(t)^*x + k_r(t)r + \theta^T\varphi(x)$ be the adaptive controller, where$k_r(t)$,θ (t) are the estimates of kr (t)*and θ (t)* respectively.

$$\begin{bmatrix}\dot{x_1}\\ \dot{x_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \frac{1}{ml^2}\begin{bmatrix}0\\ 1\end{bmatrix}[\tau - bx_2 - mglsinx_1] \tag{3.19}$$

3.1.2. Mathematical Modeling of Vander Poll Oscillation

The van der Pol equation can be represented using the circuit below

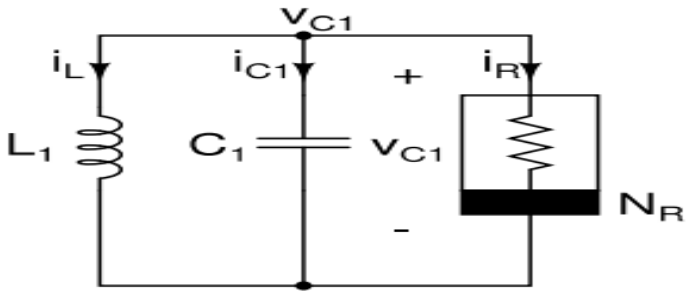


Figure 3. 2 Circuit model of van der Pol oscillator [32]

Using Kirchhoff's current law at v_{C1}gives

$$I_L + I_C + I_R = 0 \tag{3.20}$$

I_L=Inductance current, I_R ==Current on the negative resistance, I_C=capacitance current

However, $I_L = \frac{1}{L}\int V_{C1}\,dt, I_C = C\frac{dV_{C1}}{dt}$ and I_R depends on the voltage V_{C1}.

Negative Resistances is a behavior in which the current and voltage are inversely proportional to each other. A normal circuit with a resistor following Ohm's law has a current decrease when the voltage drops. In the case of negative resistance, the current increases with a voltage drop. Practically, there is nothing as a negative resistor. It is just behavior on the V-I graph, where there is a negative slope, and operating in that region would give you behavior of negative resistance. The I-v characteristics of negative resistance is that

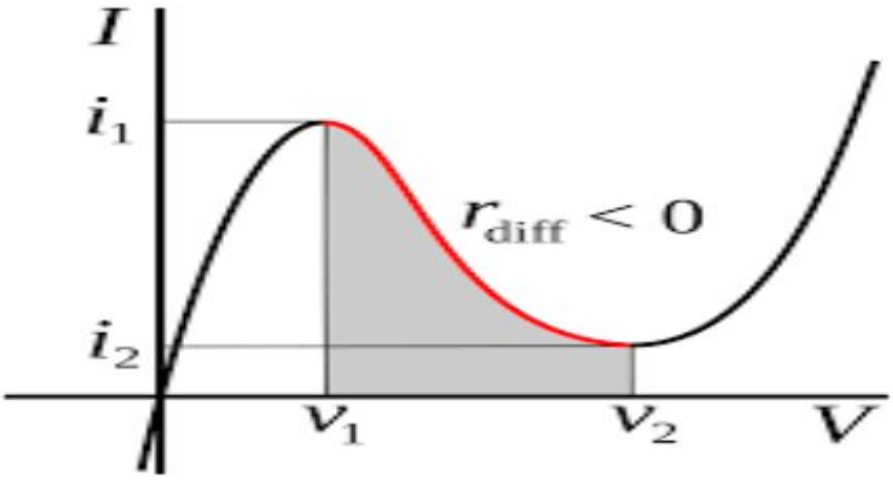


Figure 3. 3 I-v characteristics of negative resistance

Therefore, $I_R = f(V_{C1}), \frac{dI_R}{dt} = \frac{dI_R}{dV_{C1}}\frac{dV_{C1}}{dt}$ using chain rule

$$Therefore, \frac{1}{L}\int V_{C1}\,dt, +C\frac{dV_{C1}}{dt} + I_R = 0 \tag{3.21}$$

Derivation both sides gives

$$\frac{V_{C1}}{L} + C\frac{d^2V_{C1}}{dt^2} + \frac{dI_R}{dt} = 0, \frac{V_{C1}}{L} + C\frac{d^2V_{C1}}{dt^2} + \frac{dI_R}{dV_{C1}}\frac{dV_{C1}}{dt} = 0 \tag{3.22}$$

From the above I-v characteristics $I_R = \frac{V_{C1}}{R} = V_{C1}\frac{1}{R} = V_{C1}\left(-1 + \frac{V_{C1}^2}{3}\right) = -V_{C1} + \frac{V_{C1}^3}{3}$

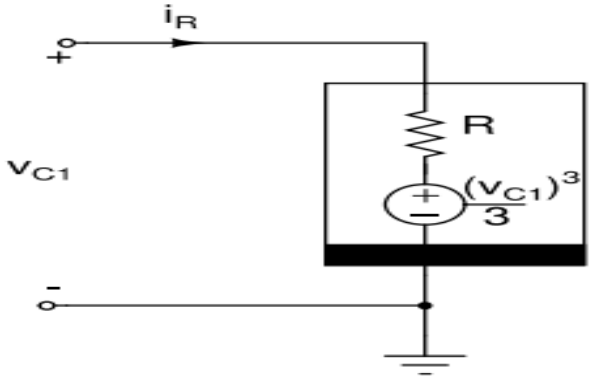


Figure 3. 4 Circuit model of nonlinear resistance for the van der pol oscillation

$$Using\ KVL, -V_{C1} + I_R(R) = 0, I_R = (V_{C1} - \frac{V_{C1}^3}{3})\ \frac{1}{R} \tag{3.23}$$

$$\frac{dI_R}{dt} = \frac{dI_R}{dV_{C1}}\frac{dV_{C1}}{dt}, \frac{dI_R}{dV_{C1}} = \frac{d(-V_{C1} - \frac{V_{C1}^3}{3})}{dV_{C1}} = -1 + V_{C1}^2 \tag{3.24}$$

Therefore, from the above equation $C\frac{d^2V_{C1}}{dt^2} + \frac{V_{C1}}{L} + (-1 + V_{C1}^2)\frac{dV_{C1}}{dt} = 0$

$$\ddot{V}_{c1} + \frac{V_{C1}}{LC} + \frac{1}{C}(-1 + V_{C1}^2)\dot{V}_{c1} = 0 \tag{3.25}$$

Let, $q1 = \frac{1}{C} > 0, q2 = \frac{1}{LC} > 0, q3 = \frac{1}{C}$

$$\ddot{V}_{c1} + q1(-1 + V_{C1}^2)\dot{V}_{c1} + q2V_{C1} = q3\ U(voltage) \tag{3.26}$$

From the above equation, the state space form is that

Let $x_1 = V_{C1}, x_2 = \dot{V}_{c1}$

$$\dot{x}_1 = x_2 \tag{3.27}$$
$$\dot{x}_2 = -q1(x_1^2 - 1)x_2 - q2x_1 + q3U$$

The General state space equation for Vander poll oscillation as shown in below.

$\dot{x} = Ax + B\,\Lambda\,[u - \theta^T\varphi(x)]$, Λ control input uncertainty.

Case I: A and Λ Unknown, but B and Sign of Λ Known.

Let the controller, $u = -K_x(t)x + k_r(t)r + \theta^T\varphi(x)$ be the adaptive controller, where$K_x(t)$,$k_r(t)$ and θ (t) are the estimates of $Kx\ (t)^*, kr\ (t)^*$ and $\theta\ (t)^*$ respectively.

$$\dot{x}_1 = 0x_1 + x_2 \tag{3.28}$$

$$\dot{x}_2 = -\frac{1}{LC}x_1 + \frac{1}{C}[U - (x_1^2 - 1)x_2]$$

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\frac{1}{LC} & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \frac{1}{C}\begin{bmatrix}0\\ 1\end{bmatrix}[U - (x_1^2 - 1)x_2] \tag{3.29}$$

Case II: A, B and Λ=I Known.

Let the controller $u = -Kx\ (t)^*x + k_r(t)r + \theta^T\varphi(x)$ be the adaptive controller, where$k_r(t)$,θ (t) are the estimates of kr $(t)^*$and θ $(t)^*$ respectively.

$$\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix} = \begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\end{bmatrix} + \frac{1}{C}\begin{bmatrix}0\\ 1\end{bmatrix}[U - \frac{1}{L}x_1 - (x_1^2 - 1)x_2] \tag{3.30}$$

3.2. Model Reference Adaptive Control

MRAC has been one of the most popular approaches to adaptive control. The basic structure of the MRAC scheme is shown in Figure 3.5 for the direct scheme. The reference model is chosen to generate the desired trajectory x_m that the plant output x_p has to follow. [2], [16]

The tracking error $e = x_m - x$ represents the deviation of the plant output from the desired trajectory. The closed-loop plant is made up of an ordinary feedback control law that contains the plant and a controller C (θ) and an adjustment mechanism that generates the controller parameter estimates θ (t) online.

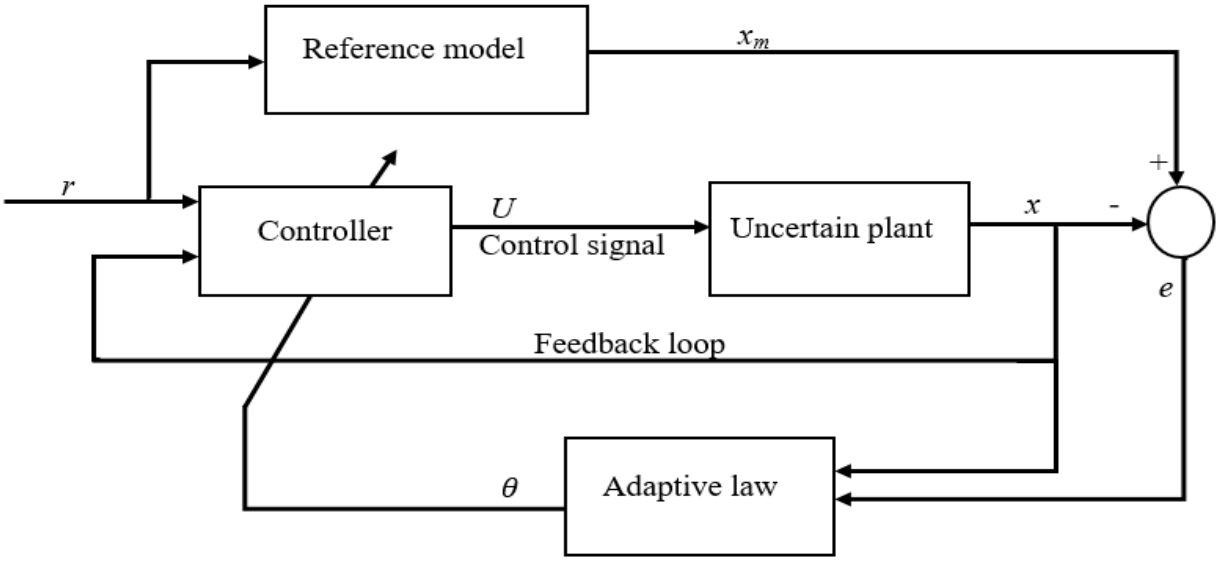


Figure 3. 5 Direct model reference adaptive control.

3.2.1. Reference Model

A reference model is used to specify the desired response of an adaptive control system to command input. It is essentially a command shaping filter to achieve the desired command following. Since adaptive control is formulated as a command following or tracking control, the adaptation is operated on the tracking error between the reference model and the system output. A reference model must be designed properly for an adaptive control system to be able to follow. Typically, a reference model is formulated as an LTI model, but a nonlinear reference model can be used although a nonlinear design always brings up many complex issues. An LTI reference model should capture all important performance specifications such as rise time and settling time, as well as robustness specifications such as phase and gain stability margins.

For example, the reference model for an adaptive control system could be selected to be a second-order system as:

$$\ddot{x}_m + 2\zeta\omega_m\dot{x}_m + \omega_m^2 x_m = \omega_m^2 r \quad (3.31)$$

Where $x_m(t)$ is, a model-reference signal that only depends on the reference command input r (t) The tracking error is defined as;$e = x_m - x$.

The objective of an adaptive control system is to adapt to system uncertainty to keep the tracking error as small as possible. In an ideal case when $e(t) \rightarrow 0$, then the system state follows the model-reference signal perfectly, i.e., $x(t) \rightarrow x_m(t)$.

3.2.2. Controller

A controller must be designed to provide overall system performance and stability for a nominal plant without uncertainty. Thus, it can be thought of as a baseline or nominal controller. The type of controllers is dictated by the objective of control design. A controller can be linear or nonlinear but as always nonlinear controllers are much more difficult to design, analyze, and ultimately certify for operation in real systems. The controller can be a nominal controller augmented with an adaptive controller or a fully adaptive controller. The adaptive augmentation control design is more prevalent and generally should be more robust than a fully adaptive control design.

3.2.3. Adaptive Law

An adaptive law is a mathematical relationship that expresses explicitly how adaptive parameters should be adjusted to keep the tracking error as small as possible. An adaptive law can be either linear time-varying or nonlinear. In any case, the stability of an adaptive control system usually must be analyzed using Lyapunov stability theory. Many different adaptive laws have been developed, and each has its own advantages as well as disadvantages. Ultimately, designing an adaptive control system comes down to a trade-off between performance and robustness. This trade-off can be made by a suitable selection of an adaptive law and a set of tuning parameters that are built into an adaptive law.

3.2.4. Uncertain plant

The uncertain plant is the plant we want to design based on the above controller and adaptive mechanism. Adaptive control can deal with either linear or nonlinear plants with various types of uncertainty, which can be structured uncertainty, unstructured uncertainty, or unmodeled dynamics.

3.3. Direct MRAC for First-Order nonlinear SISO Systems

Consider a first-order nonlinear SISO system

$$\frac{dx}{dt} = ax + b[u + f(x)] \tag{3.32}$$

Subject to $x(0) = x_0$ where f(x) is a structured matched uncertainty that can be linearly parametrized as

$$f(x) = \sum_{i=1}^{p} \theta^*{}_i \, \varphi_i(x) = \theta^{*T}\varphi(x) \tag{3.33}$$

Where, $\theta^* = [\theta_1 \quad \theta_2 \quad \theta_3 \ldots\ldots \theta_P] \in R^P$ is an unknown constant vector and $\varphi(x) = [\varphi_1(x) \quad \varphi_2(x) \quad \varphi_3(x) \ldots\ldots \varphi_P(x)]^T \in R^P$ is a vector of known bounded basis functions.

3.3.1 Case I a and b Unknown but Sign of b Known

A reference model is specified as

$$\frac{dx_m}{dt} = a_m x_m + b_m r \tag{3.34}$$

Subject to $x_m(0) = x_{m0}$, where $a_m < 0$ and $r(t) \in L_\infty$ is a piecewise continuous bounded reference command signal so that $x_m(t)$ is a uniformly bounded model reference signal.

Firstly, define an ideal controller that perfectly cancels out the uncertainty and enables x (t) to follow x_m (t) as

$$u = -k_x{}^* x + k_r{}^* r - \theta^{*T}\varphi(x) \tag{3.35}$$

where the superscript $*$ denotes ideal constant values which are unknown.

Upon substituting into the plant model, we get the ideal closed-loop plant

$$\frac{dx}{dt} = (a - bk_x{}^*)x + bk_r{}^* r \tag{3.36}$$

Comparing the ideal closed-loop plant to the reference model, the ideal gains $k_x{}^*$ and $k_r{}^*$ can be determined by the following model matching conditions:

$$a_m = a - bk_x{}^*, b_m = bk_r{}^* \tag{3.37}$$

It turns out that the solutions for $k_x{}^*$ and $k_r{}^*$ always exist since there are two independent equations with two unknowns.

The actual adaptive controller is an estimate of the ideal controller with a goal that in the limit the adaptive controller approaches the ideal controller.

$$Let\ u = -k_x(t)x + k_r(t)r(t) - \theta^T\varphi(x) \tag{3.38}$$

be the adaptive controller, where $k_x(t)$, $k_r(t)$, and θ (t) are the estimates of k_x^*, k_r^* and θ^*, respectively.

The adaptive controller is a direct adaptive controller since k_x (t), k_r (t), and θ (t) is estimated directly without the knowledge of the unknown system parameters a, b, and θ^*.

Now, define the estimation errors as

$$\left.\begin{aligned}\Delta k_x &= k_x(t) - k_x^* \\ \Delta k_r &= k_r(t) - k_r^* \\ \Delta\theta &= \theta\,(t) - \theta*\end{aligned}\right\} \tag{3.39}$$

Substituting these into the plant model gives

$$\frac{dx}{dt} = (a - bk_x^* - b\Delta k_x)x + (bk_r^* + b\Delta k_r)r - b\Delta\theta^T\varphi(x) \tag{3.40}$$

But, $a_m = a - bk_x^*, b_m = bk_r^*$

Let $e = x_m(t) - x(t)$ be the tracking error. Then, the closed-loop tracking error equation is established as

$$\dot{e} = \dot{x_m}(t) - \dot{x}(t) = a_m e + b\Delta k_x x - b\Delta k_r r + b\Delta\theta^T\varphi(x) \tag{3.41}$$

Note that the tracking error equation is non-autonomous due to r (t).

Now, the task of defining the adaptive laws to adjust k_x (t), k_r (t), and θ (t) are considered next. This can be accomplished by conducting a Lyapunov stability proof as follows:

Choose a Lyapunov candidate function

$$V(e, \Delta k_x, \Delta k_r, \Delta\theta) = e^2 + |b|\left(\frac{\Delta k_x^2}{\gamma_x} + \frac{\Delta k_r^2}{\gamma_r} + \Delta\theta^T\Gamma_\theta{}^{-1}\Delta\theta\right) > 0 \tag{3.42}$$

Where $\gamma_x > 0$ and $\gamma_r > 0$ are called the adaptation (or learning) rates for k_x (t) and k_r (t), and

$\Gamma_\theta = \Gamma_\theta{}^T > 0$ $R^p \times R^p$ is a positive-definite adaptation rate matrix for θ(t).

$\frac{dV(e,\Delta k_x,\Delta k_r,\Delta\theta)}{dt}$ is evaluated as

$$\dot{V} = 2e\dot{e} + |b|(2\frac{\Delta k_x\Delta\dot{k}_x}{\gamma_x} + 2\frac{\Delta k_r\Delta\dot{k}_r}{\gamma_r} + 2\Delta\theta^T\Gamma_\theta{}^{-1}\Delta\dot{\theta}) > 0 \tag{3.43}$$

Substituting (3.41) into (3.43), we get

$$\begin{aligned}\dot{V} = {} & 2a_m e^2 + 2\Delta k_x(-ebx + |b|\frac{\Delta\dot{k}_x}{\gamma_x}) + 2\Delta k_r(-ebr + |b|\frac{\Delta\dot{k}_r}{\gamma_r}) \\ & + 2\Delta\theta^T(e\varphi(x) + |b|\Gamma_\theta{}^{-1}\Delta\dot{\theta})\end{aligned} \tag{3.44}$$

Since $b = |b|sgnb$ then $\dot{V} < 0$ if

$$-ebx + |b|\frac{\Delta\dot{k}_x}{\gamma_x} = 0, -ebr + |b|\frac{\Delta\dot{k}_r}{\gamma_r} = 0, e\varphi(x) + |b|{\Gamma_\theta}^{-1}\Delta\dot{\theta} = 0 \quad (3.45)$$

$$\left.\begin{aligned}\Delta\dot{k}_x &= \gamma_x xesgnb\\ \Delta\dot{k}_r &= \gamma_r resgnb\\ \Delta\dot{\theta} &= -\Gamma_\theta\varphi(x)esgnb\end{aligned}\right\} \quad (3.46)$$

Then $\dot{V} = 2a_m e^2 \leq 0$

Since, $\dot{V} = 2a_m e^2 \leq 0$ then $e(t), k_x(t), k_r(t)$ and θ (t) are bounded.

$$Then, \lim_{t\to\infty} V(e, \Delta k_x, \Delta k_r, \Delta\theta) = V(e_0, \Delta k_{x0}, \Delta k_{r0}, \Delta\theta_0) + 2a_m{||e||_2}^2 \quad (3.47)$$

Where $e(0) = e_0, \Delta k_x(0) = \Delta k_{x0}, \Delta k_r(0) = \Delta k_{r0}, \Delta\theta(0) = \Delta\theta_0$

So, $V(e, \Delta k_x, \Delta k_r, \Delta\theta)$ has a finite limit as t → ∞. Since||e||2 exists, therefore e (t) ∈ $L_2 \cap L_\infty$, but ė ∈ L_∞.

$\dot{V}$ can be shown to be uniformly continuous by examining its derivative to see whether it is bounded. $\frac{\partial^2 V(e,\Delta k_x,\Delta k_r,\Delta\theta)}{\partial t^2}$ is computed as

$$\ddot{V}(e, \Delta k_x, \Delta k_r, \Delta\theta) = 4a_m e\dot{e} = 4a_m e[a_m e + b\Delta k_x x - b\Delta k_r r + b\Delta\theta^T\varphi(x)] \quad (3.48)$$

Since $e(t), k_x, k_r$ and $\theta(t)$ is bounded by the virtue that $\dot{V}(e, \Delta k_x, \Delta k_r, \Delta\theta) \leq 0$, x (t) is bounded because of $e(t)$ and $x_m(t)$ are bounded, $r(t)$ is a bounded reference command signal, and $\varphi(x)$ is bounded because x (t) is bounded; therefore $\ddot{V}(e, \Delta k_x, \Delta k_r, \Delta\theta)$is bounded.

Thus, $\ddot{V}(e, \Delta k_x, \Delta k_r, \Delta\theta)$is uniformly continuous. It follows from the Barbalat's lemma that $\dot{V}(e, \Delta k_x, \Delta k_r, \Delta\theta) \to 0$, hence e(t) → 0 as t → ∞. The tracking error is asymptotically stable, but the whole adaptive control system is not asymptotically stable since k_x (t), k_r (t) and θ (t) can only be shown to be bounded.

3.3.2 Case II a and b Known

If a and b are known, then the gains k_x and k_r need not be estimated since they are known and can be found from the model matching conditions

$${k_x}^* = \frac{a_m - a}{b}, \qquad {k_r}^* = \frac{b_m}{b}$$

Then, the adaptive control is given by

$$u = -{k_x}^* x + {k_r}^* r - \theta^T\varphi(x) \quad (3.49)$$

Where only θ (t) needs to be adjusted.

The tracking error equation is then obtained as

$$\dot{e} = a_m e + b\Delta\theta^T \varphi(x) \tag{3.50}$$

The adaptive law can be found to be

$$\dot{\Delta\theta} = -\Gamma_\theta \varphi(x) eb$$

To show that the adaptive law is stable, choose a Lyapunov candidate function

$$V(e, \Delta\theta) = e^2 + \Delta\theta^T {\Gamma_\theta}^{-1} \Delta\theta > 0. \tag{3.51}$$

$$Then, \dot{V} = 2e\dot{e} + 2\Delta\theta^T {\Gamma_\theta}^{-1} \dot{\Delta\theta} = 2a_m e^2 + 2eb\Delta\theta^T \varphi(x) - 2\Delta\theta^T eb\varphi(x) = 2a_m e^2 \tag{3.52}$$

$$\leq 0$$

The Barbalat's lemma can be used to show that the tracking error is asymptotically stable, i.e., $e(t) \rightarrow 0$ as $t \rightarrow \infty$

3.4. Direct MRAC for Second-Order nonlinear SISO Systems

Consider a second-order nonlinear SISO system

$$\ddot{y} + 2\zeta\omega_n \dot{y} + \omega_n^2 y = b[u + f(y, \dot{y})] \tag{3.53}$$

where ζ and ω_n are unknown and $f(y, \dot{y}) = \theta^{*T} \varphi(y, \dot{y})$ is defined in a manner similar to first or der system above.

Let $x_1 = y(t), x_2 = \dot{y}(t)$, and $x(t) = [x_1(t) \;\; x_2(t)]^T \in R^2$.

The state-space form of the system is

$$\dot{x} = Ax + B[u + \theta^{*T} \varphi(x)] \tag{3.54}$$

Let the adaptive controller be

$$u = -K_x(t)x + k_r(t)r(t) - \theta^T \varphi(x) \tag{3.55}$$

For the model of the system is given by

Let $x_{m1}(t) = y_m(t), x_{m2}(t) = \dot{y_m}(t)$ and $x_m(t) = [x_{m1}(t)\; x_{m2}(t)]^T \in R^2$.

A reference model is given by

$$\frac{dx_m}{dt} = A_m x_m + B_m r \tag{3.56}$$

Where $r(t) \in R$ is a bounded command signal, $A_m \in R^2 \times R^2$ is Hurwitz and known, and $B_m \in R^2$ is also known.

3.4.1. Case I A and B Known

If *A* and *B* are known, then it is assumed that there exist K_x and k_r that satisfy the model matching conditions

$$A_m = A - BK_x^*, B_m = Bk_r^* \tag{3.57}$$

For a second-order system, if A_m and B_m have the same structures as those of A and B, respectively, then K_x and k_r can be determined by using the pseudo-inverse method.

Let the adaptive controller be

$$u = -K_x^* x + k_r^* r(t) - \theta^T \varphi(x) \tag{3.58}$$

Then, the closed-loop plant is

$$\dot{x} = (A - BK_x^*)x + BK_r^* - B\Delta\theta^T \varphi(x) \tag{3.59}$$

$\Delta\theta = \theta\,(t) - \theta *$ is the estimation error and the tracking error equation is $e = x_m(t) - x(t)$

$$\dot{e} = \dot{x_m}(t) - \dot{x}(t) = A_m x_m + B_m r - ((A - BK_x^*)x + BK_r^* - B\Delta\theta^T \varphi(x)) \tag{3.60}$$

But $A_m = A - BK_x^*$, $B_m = BK_r^*$

$$Therefore, \dot{e} = A_m e + B\Delta\theta^T \varphi(x) \tag{3.61}$$

Using lyapunove equation to determine the estimates of the unknown parameter θ(t).

$$V(e, \Delta\theta) = e^T P e + \Delta\theta^T \Gamma_\theta^{-1} \Delta\theta > 0 \tag{3.62}$$

$$\frac{dV(e,\Delta\theta)}{dt} = 2(e^T P e + e^T P \dot{e}) + 2\Delta\theta^T \Gamma_\theta^{-1} \dot{\Delta\theta} \tag{3.63}$$

Substituting the rate of an error on the above equation, we get

$$\dot{V} = -2e^T Q e + 2e^T PB\Delta\theta^T \varphi(x) + 2\Delta\theta^T \Gamma_\theta^{-1} \dot{\Delta\theta} \tag{3.64}$$

$$\dot{V} = -2e^T Q e + 2\Delta\theta^T [\varphi(x) e^T PB + \Gamma_\theta^{-1} \dot{\Delta\theta}] \tag{3.65}$$

$$\dot{V} \leq 0 \; when \; \varphi(x) e^T PB + \Gamma_\theta^{-1} \dot{\Delta\theta} = 0 \tag{3.66}$$

Thus, the following adaptive law is obtained:

$$\dot{\Delta\theta} = \dot{\theta} = -\Gamma_\theta \varphi(x) e^T PB, \; \Gamma_\theta = 2 * 2 \text{ Matrix} \tag{3.67}$$

Then, $\dot{V} = -2e^T Q e \leq 0$

Therefore, e(t) and θ(t) is bounded. Using the same argument with the Barbalat's lemma as in the previous sections, one can conclude that $\ddot{V}(e, \Delta\theta)$ is uniformly continuous so $\dot{V}(e, \Delta\theta) \rightarrow 0$ as $t \rightarrow \infty$. Therefore, the tracking error is asymptotically stable with $e(t) \rightarrow 0$ as $t \rightarrow \infty$.

3.4.2. Case II A and B Unknown but sign of B known

A reference model is specified as

$$\frac{dx_m}{dt} = A_m x_m + B_m r \tag{3.68}$$

Subject to $x_m(0) = x_{m0}$ where $A_m < 0$ and $r(t) \in L_{\infty}$ is a piecewise continuous bounded reference command signal, so that $x_m(t)$ is a uniformly bounded model reference signal.

Firstly, define an ideal controller that perfectly cancels out the uncertainty and enables x(t) to follow $x_m(t)$ as

$$u = -K_x{}^* x + k_r{}^* r(t) - \theta^{*T}\varphi(x) \tag{3.69}$$

Where the superscript $*$ denotes ideal constant values, which are unknown.

Upon substituting into the plant model, we get the ideal closed-loop plant

$$\dot{x} = (A - BK_x{}^*)x + BK_r{}^* r \tag{3.70}$$

The following model matching conditions can determine to compare the ideal closed-loop plant to the reference model, the ideal gains $K_x{}^*$ and $K_r{}^*$

$$A_m = A - BK_x{}^*, B_m = BK_r{}^*$$

It turns out that the solutions for $K_x{}^*$ and $k_r{}^*$ always exists since there are two independent equations with two unknowns.

The actual adaptive controller is an estimate of the ideal controller with a goal that in the limit the adaptive controller approaches the ideal controller.

$$Let\ , u = -K_x(t)x(t) + k_r(t)r(t) - \theta^T(t)\varphi(x) \tag{3.71}$$

Be the adaptive controller, where $K_x(t), k_r(t)$ and $\theta(t)$ are the estimates of $K_x{}^*, k_r{}^*$ and θ^*, respectively.

The adaptive controller is a direct adaptive controller since $K_x(t), k_r(t)$ and $\theta(t)$ are estimated directly without the knowledge of the unknown system parameters A, B and θ^*.

Substituting the estimation errors as from equation (3.39), (3.71) into the equation (3.54) gives

$$\dot{x} = (A - BK_x{}^* - B\Delta K_x(t))x + (Bk_r{}^* + B\Delta k_r(t))r - B\Delta\theta^T(t)\varphi(x) \tag{3.72}$$

$$A_m = A - BK_x{}^*, B_m = BK_r{}^*$$

Let $e = x_m(t) - x(t)$ be the tracking error. Then, the closed-loop tracking error equation is established as

$$\dot{e} = \dot{x_m}(t) - \dot{x}(t) = A_m e + B\Delta K_x(t)x - B\Delta k_r(t)r + B\Delta\theta^T(t)\varphi(x) \tag{3.73}$$

Note that the tracking error equation is non-autonomous due to $r(t)$.

Now, the task of defining the adaptive laws to adjust$K_x(t), K_r(t)$ and $\theta(t)$ is considered next. This can be accomplished by conducting a Lyapunov stability proof as follows:

Choose a Lyapunov candidate function

$$V(e, \Delta K_x, \Delta k_r, \Delta\theta) = e^T Pe + |B|\left(\frac{\Delta K_x \Delta K_x^T}{\Gamma_x} + \frac{\Delta k_r^2}{\gamma_r} + \Delta\theta^T {\Gamma_\theta}^{-1}\Delta\theta\right) > 0 \tag{3.74}$$

Where $\Gamma_x = {\Gamma_x}^T = R^p \times R^p > 0$ and $\gamma_r > 0$ are called the adaptation (or learning) rates for K_x (t) and k_r (t), and $\Gamma_\theta = {\Gamma_\theta}^T > 0 = R^p \times R^p$ is a positive-definite adaptation rate matrix for θ (t).

$\frac{dV(e,\Delta K_x,\Delta k_r,\Delta\theta)}{dt}$ is evaluated as

$$\dot{V} = 2(e^T Pe + e^T P\dot{e}) + 2|B|(\frac{\Delta K_x \Delta\dot{K}_x^T}{\Gamma_x} + \frac{\Delta k_r \Delta\dot{k}_r}{\gamma_r} + \Delta\theta^T {\Gamma_\theta}^{-1}\Delta\dot{\theta}) > 0 \tag{3.75}$$

Substituting (3.73) into the equation (3.75), we get

$$\dot{V} = -2e^T Qe + 2\Delta K_x(-xe^T BP + |B|\frac{\Delta\dot{K}_x^T}{\Gamma_x}) + 2\Delta K_r(-re^T BP + |B|\frac{\Delta\dot{k}_r}{\gamma_r}) + 2\Delta\theta^T(\varphi(x)e^T BP + {\Gamma_\theta}^{-1}\Delta\dot{\theta}) > 0 \tag{3.76}$$

Where, ${A_m}^T P + PA_m = -Q$

Since$B = |B|sgnB$, then $\dot{V} \leq 0$ if

$$-xe^T BP + |B|\frac{\Delta\dot{K}_x^T}{\Gamma_x} = 0, -re^T BP + |B|\frac{\Delta\dot{k}_r}{\gamma_r} = 0, \varphi(x)e^T BP + {\Gamma_\theta}^{-1}\Delta\dot{\theta} = 0 \tag{3.77}$$

$$\left.\begin{aligned}\Delta\dot{K}_x^T = \dot{K}_x^T = \Gamma_x xe^T PsgnB\\ \Delta\dot{k}_r = \dot{k}_r = \gamma_r re^T PsgnB\\ \dot{\theta}^T = -\Gamma_\theta\varphi(x)e^T PsgnB\end{aligned}\right\} \tag{3.78}$$

Then, $\dot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta) = -2e^T Qe \leq 0$

Since, $\dot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta) = -2e^T Qe \leq 0$ then $e(t), K_x(t), \Delta k_r(t)$ and $\theta(t)$ are bounded.

$$Then, \lim_{t\to\infty} V(e, \Delta K_x, \Delta k_r, \Delta\theta) = V(e_0, \Delta K_{x0}, \Delta k_{r0}, \Delta\theta_0) - 2Q||e||_2 \tag{3.79}$$

Where, $e(0) = e_0, \Delta K_x(0) = \Delta K_{x0}, \Delta k_r(0) = \Delta k_{r0}, \Delta\theta(0) = \Delta\theta_0$.

So, $V(e, \Delta K_x, \Delta k_r, \Delta\theta)$ has a finite limit as $t \to \infty$. Since$||e||2$ exists, therefore $e(t) \in L_2 \cap L_\infty$, but $\dot{e} \in L_\infty$.

$\dot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta)$ can be shown to be uniformly continuous by examining its derivative to see whether it is bounded.

$\frac{\partial^2 V(e,\Delta K_x, \Delta k_r,\Delta\theta)}{\partial t^2}$ is computed as

$$\ddot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta) = -Qe\dot{e} = -2Qe[A_m e + B\Delta K_x x - B\Delta k_r r + B\Delta\theta^T \varphi(x)] \quad (3.80)$$

Since $e(t), \Delta K_x, \Delta k_r$ and $\theta(t)$ are bounded by the virtue that $\dot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta) \leq 0$, $x(t)$ is bounded because of $e(t)$and $x_m(t)$ are bounded, $r(t)$ is a bounded reference command signal, and $\varphi(x)$ is bounded because of $x(t)$ is bounded; therefore, $\ddot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta)$ is bounded. Thus, $\ddot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta)$ is uniformly continuous. It follows from the Barbalat's lemma that $\dot{V}(e, \Delta K_x, \Delta k_r, \Delta\theta) \to 0$, hence $e(t) \to 0$ as $t \to \infty$. Therefore, from this we can conclude that the tracking error is asymptotically stable.

Chapter Four

4. MATLAB Simulation Result and Discussion

4.1. Reference model design

To carry out controller design and analysis of nonlinear systems using model reference adaptive control, the obtained non-linear model has to be linearized and design linear controller for the linearized system since to perform a simulation and stability analysis. Linear models are easier to understand than nonlinear models and are necessary for most control system design methods. Using Taylor's series linearization at the equilibrium point (Equilibrium points are solutions of the differential equation, which can be calculated from the following state space equation).

$$\dot{x}_1 = f_1(x_1, x_2) = 0 \tag{4.1}$$
$$\dot{x}_2 = f_2(x_1, x_2) = 0$$

Where, $f_1(x_1, x_2)$ and $f_2(x_1, x_2)$ are state space nonlinear equations.

$$\dot{x}_1 = f_1(x_1, x_2) = 0x_1 + x_2 \tag{4.2}$$
$$\dot{x}_2 = f_2(x_1, x_2) = -\frac{b}{ml^2}x_2 + \frac{1}{ml^2}[\tau - mglsinx_1]$$

To determine the equilibrium point using the above equation we get $\dot{x}_1 = f_1(x_1, x_2) = 0 = x_2$ and $\dot{x}_2 = f_2(x_1, x_2) = -\frac{b}{ml^2}x_2 + \frac{1}{ml^2}[\tau - mglsinx_1] = 0$ but $x_2 = 0$.

Therefore, $sinx_1 = 0, x_1 = sin^{-1}(0), x_1 = n\pi, n = \cdots, -1, 0, 1, 2, \ldots\ldots$

The stable equilibrium point is that $(x_{10}, x_{20}) = (0,0)$.

Using the Taylor series linearization method the new linear state-space model as shown below.

$$\Delta\dot{x}_1 = \frac{\partial f_1(x_1, x_2)}{\partial x_1}|_{(x_{10}, x_{20})}.\Delta x_1 + \frac{\partial f_1(x_1, x_2)}{\partial x_2}|_{(x_{10}, x_{20})}.\Delta x_2 + \frac{\partial f_1(x_1, x_2)}{\partial u}|_{(x_{10}, x_{20})}u \tag{4.3}$$

$$\Delta\dot{x}_2 = \frac{\partial f_2(x_1, x_2)}{\partial x_1}|_{(x_{10}, x_{20})}.\Delta x_1 + \frac{\partial f_2(x_1, x_2)}{\partial x_2}|_{(x_{10}, x_{20})}.\Delta x_2 + \frac{\partial f_2(x_1, x_2)}{\partial u}|_{(x_{10}, x_{20})}.u \tag{4.4}$$

Finally, the new linearized state-space model for simple pendulum as shown in (4.5).

$$\Delta\dot{x}_1 = 0\Delta x_1 + \Delta x_2 \tag{4.5}$$
$$\Delta\dot{x}_2 = -\frac{g}{l}\Delta x_1 - \frac{b}{ml^2}\Delta x_2 + u$$
$$y = \Delta x_1$$

The state space model of simple pendulum as shown below

$$\begin{bmatrix}\dot{\Delta x_1}\\ \dot{\Delta x_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ -\frac{g}{l} & -\frac{b}{ml^2}\end{bmatrix}\begin{bmatrix}\Delta x_1\\ \Delta x_2\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}u \tag{4.6}$$

$$y = [1\ 0]\begin{bmatrix}\Delta x_1\\ \Delta x_2\end{bmatrix} \tag{4.7}$$

The equation ***(***4.6***),*** ***(***4.7***)*** can be described as general sate space model

$$\dot{X} = AX + BU \ \ and\ Y = CX + DU \tag{4.8}$$

In order to obtain the parameters $A, B, C\ and\ D$ matrices for simple pendulum model, we consider the physical parameter values on the following table.

Table 4. 1 Physical parameters of simple pendulum [31]

Parameters	Value	Unit
m(pendulum mass)	0.5	kg
l(length of the rod)	1	m
b(damping coefficient)	0.5	Ns/m or kg/s
g(gravity constant)	9.8	m/s^2

Therefore, from the above table, we can determine the sate space parameters as shown below.

$$\dot{X} = AX + BU \Leftrightarrow \begin{bmatrix}\dot{\Delta x_1}\\ \dot{\Delta x_2}\end{bmatrix} = \begin{bmatrix}0 & 1\\ -9.8 & -1\end{bmatrix}\begin{bmatrix}\Delta x_1\\ \Delta x_2\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}u \tag{4.9}$$

$$Y = CX + DU \Leftrightarrow y = [1\ 0]\begin{bmatrix}\Delta x_1\\ \Delta x_2\end{bmatrix}$$

From the above state space equation, we can determine the response of simple pendulum without controller and with controller.

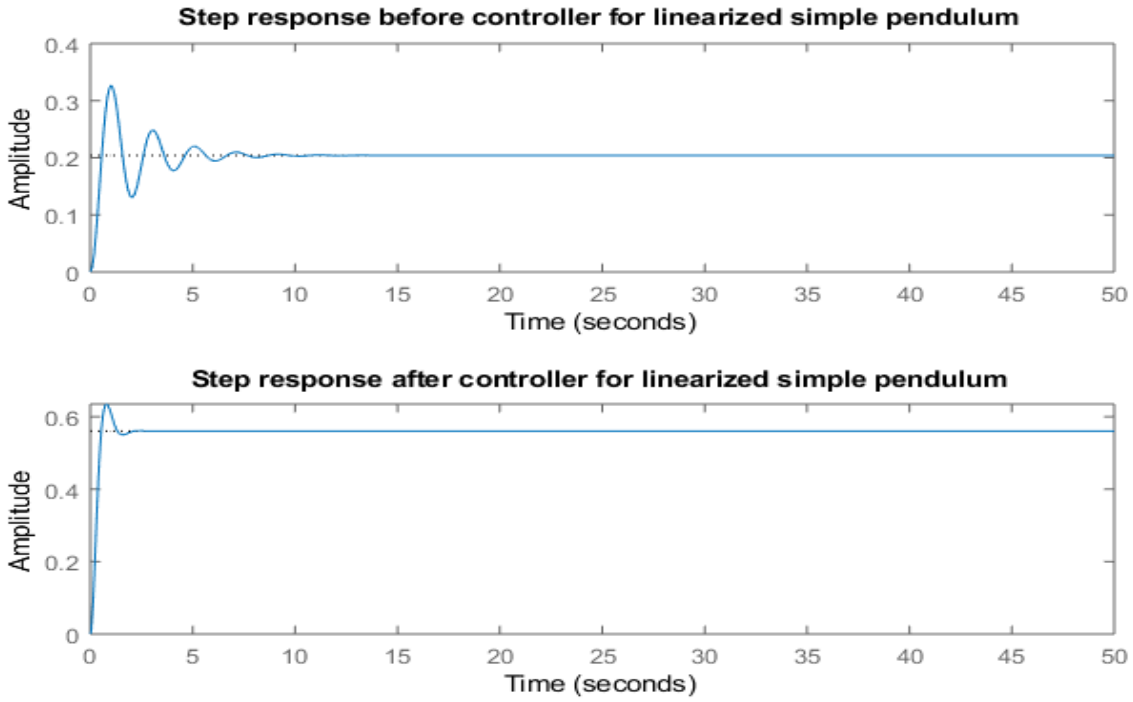


Figure 4. 1 The response of simple pendulum without controller and with controller.

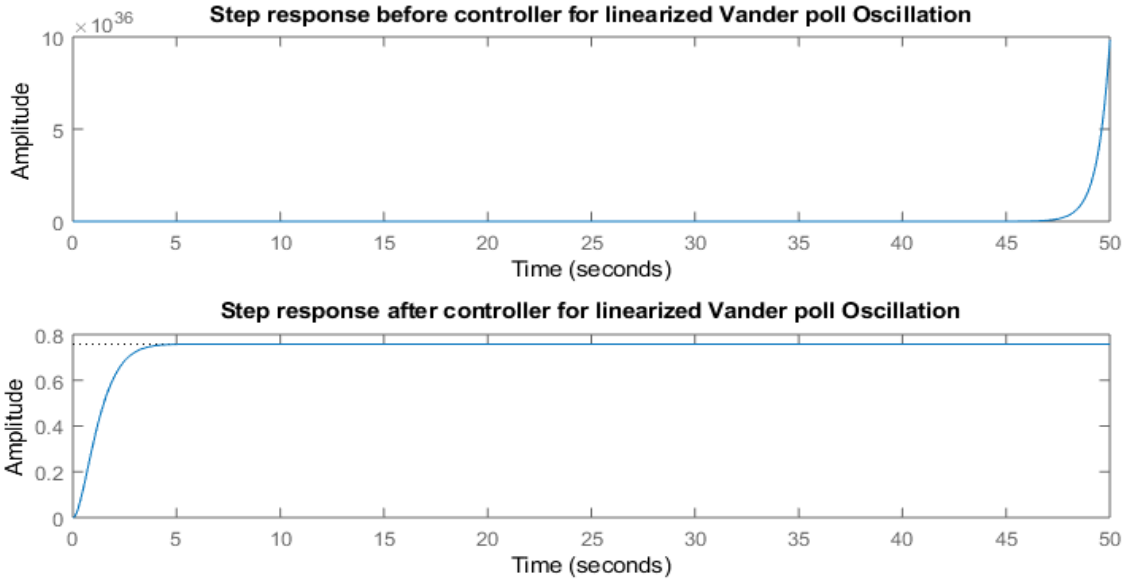


Figure 4. 2 The response of Vander poll Oscillation without controller and with controller.

It is clear from the response that the system is unstable without a controller; hence, a controller is needed in order to stabilize the system. Thus, a PID and LQR controller was implemented for the linearized system.

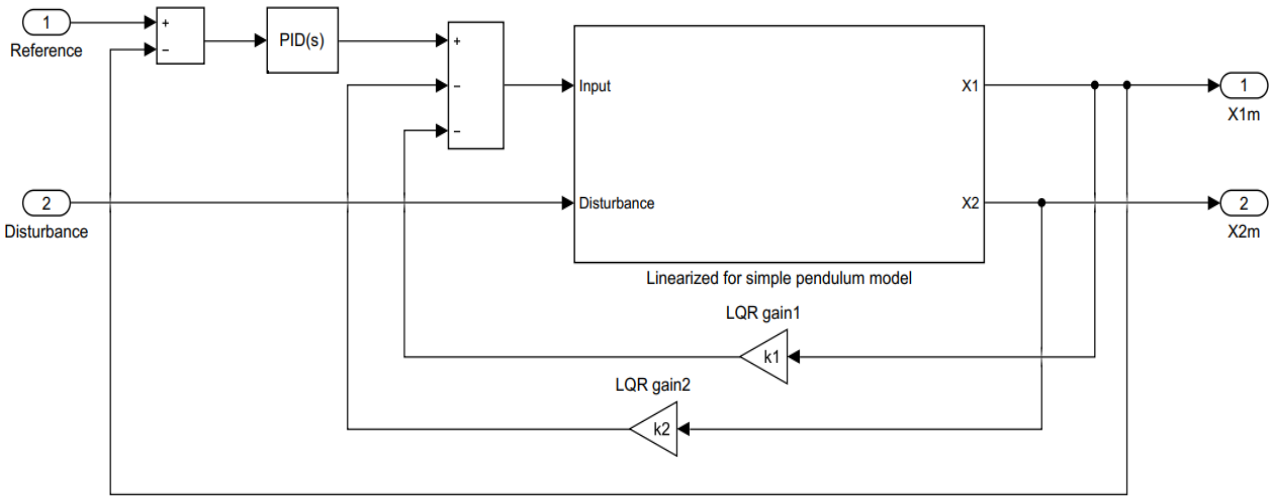


Figure 4. 3 Linearized Reference model block for simple pendulum

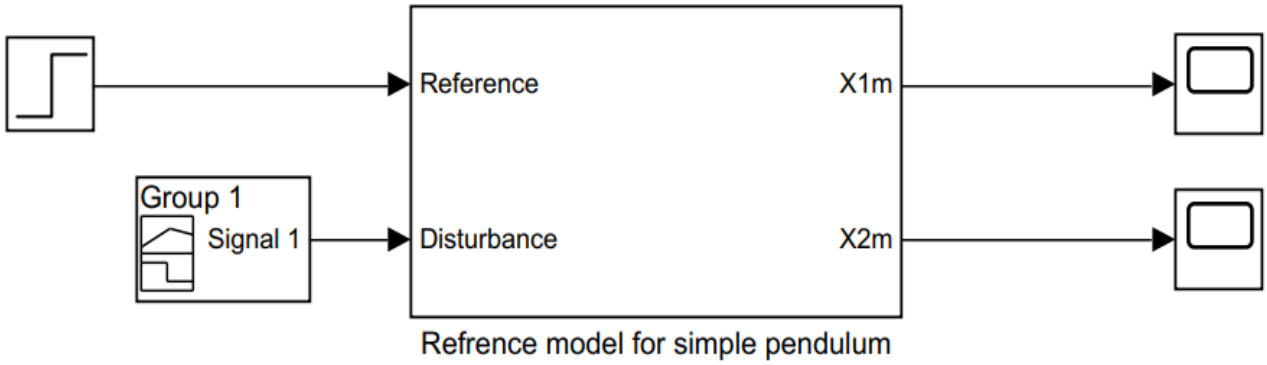


Figure 4. 4 Reference model for Simple pendulum for different reference disturbance.

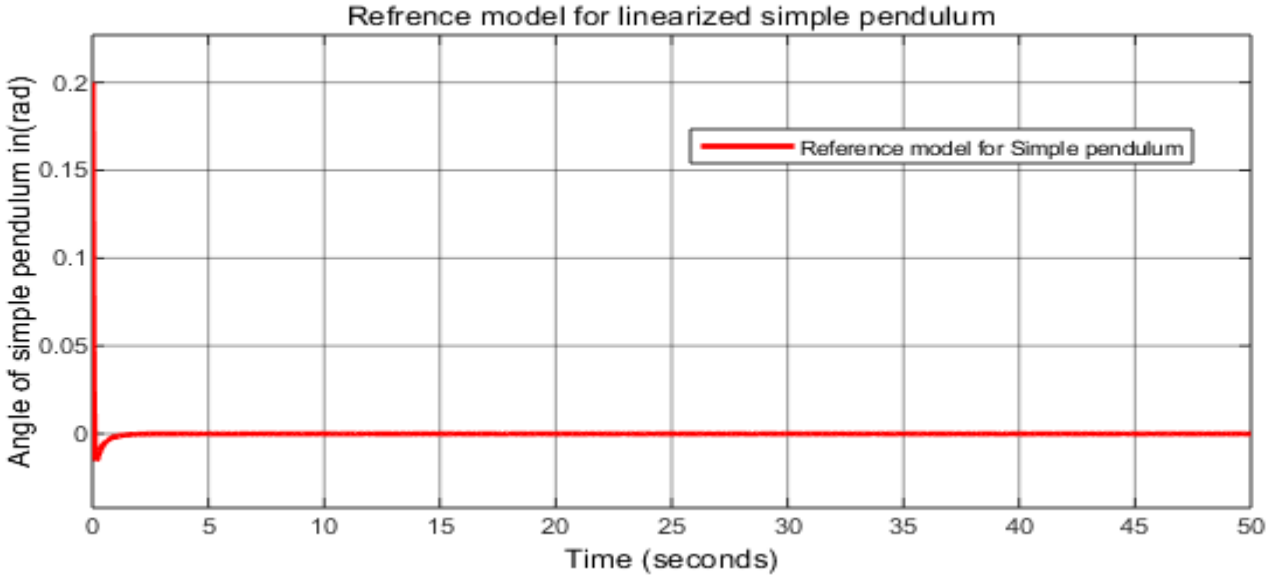


Figure 4. 5 Regulated response disturbed from 0.2rad for Linearized Pendulum.

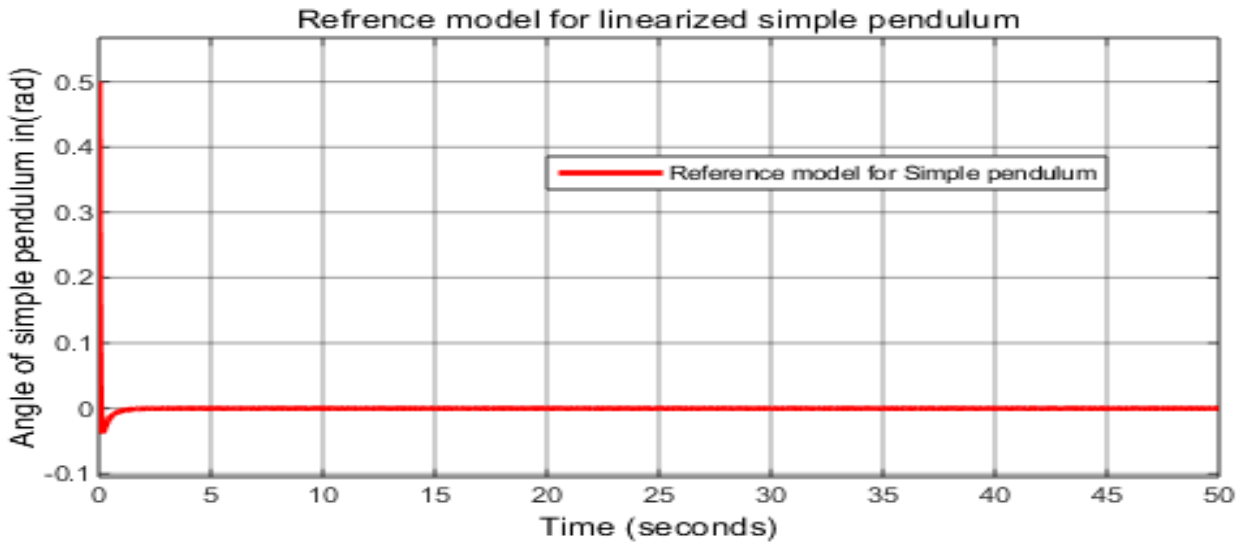


Figure 4. 6 Regulated response disturbed from 0.5rad for Linearized Pendulum.

4.2. Adaptation law design for simple pendulum

From state space equation form of the simple pendulum in the equation shown below.

$$\dot{x}_1 = 0x_1 + x_2 \tag{4.10}$$

$$\dot{x}_2 = 0x + \frac{1}{ml^2}[\tau - mglsinx_1 - bx_2]$$

We can express as a general expression of $\dot{x} = Ax + B\,\Lambda\,[u - \theta^T\varphi(x)]$ because the parameters of mgl and b are unknown or vary with time. Therefore, we can assign as the estimates of the parameter $\theta_1(theta1), \theta_2(theta2)$ respectively and the known basis function $sinx_1$ and x_2 to $\varphi_1(x), \varphi_2(x)$ respectively.

From the above

$$\dot{x}_1 = 0x_1 + x_2 \tag{4.11}$$

$$\dot{x}_2 = 0x + \frac{1}{ml^2}[\tau - [\theta_1(theta1) \quad \theta_2(theta2)]\begin{bmatrix}\varphi_1(x)\\ \varphi_2(x)\end{bmatrix}]$$

Let the adaptive controller

$$\boldsymbol{\tau} = -K_x{}^*.x + k_r(t).r(t) + \theta^T(t)\varphi(x), where, \Delta k_r = k_r(t) - k_r{}^*, \Delta\theta = \theta\,(t) - \theta* \tag{4.12}$$

However, $K_x{}^*$ is the ideal controller gains obtained from Linear Quadratic Regulator.

Based on the Lyapunov adaptive law in chapter 3 we can get the estimates of

$$\Delta\dot{k}_r = \dot{k_r} = \gamma_r r e^T PsgnB \tag{4.13}$$

$$\Delta\dot{\theta} = \dot{\theta} = \Gamma_\theta\varphi(x)e^T PsgnB$$

The dimension of theta and phi as shown

$$\theta = \begin{bmatrix}\theta_1 \\ \theta_2\end{bmatrix} = \begin{bmatrix} mgl \\ b \end{bmatrix}, \varphi(x) = \begin{bmatrix}\varphi_1(x) \\ \varphi_2(x)\end{bmatrix} = \begin{bmatrix} sinx_1 \\ x_2 \end{bmatrix}, e^T = [e_1 \quad e_2] \tag{4.14}$$

Where $e_1 = x_{1m} - x_1$ *and* $e_2 = x_{2m} - x_2$ *and* γ_r, Γ_θ are adaptation rates.

Lastly, by putting this adaptation law in MATLAB Simulink block.

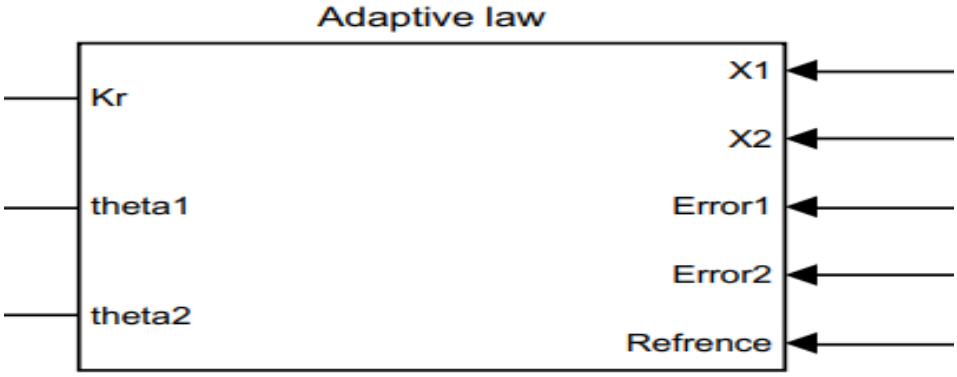


Figure 4. 7 Block diagram of Lyapunov adaptation law.

4.3. Controller Design for Simple Pendulum

Depending on unknown or known of the parameters of the nonlinear plant, choosing the controller is based on the combination of both nominal and adaptive controller for more stabilize the system. Let the controller has been used for this thesis

$$U = U_{nominal} + U_{adaptive} \tag{4.15}$$

$$\tau = -K_x^*.x + k_r(t).r + \theta^T.\varphi(x) \tag{4.16}$$

Based on the adaptation law output the controller parameter updated online until the controller parameter output reaches the ideal values.

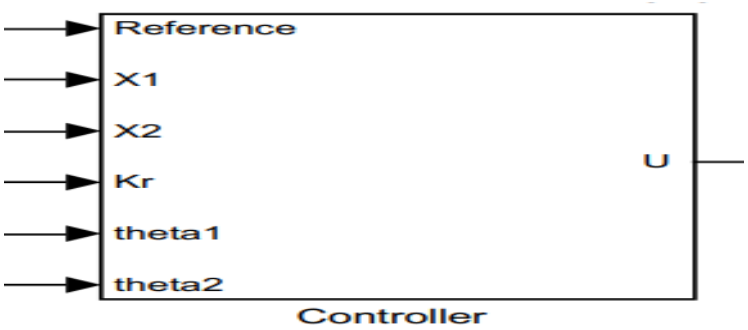


Figure 4. 8 Block diagram for Conventional MRAC.

To track the nonlinear system to the reference model response using conventional model reference adaptive control (MRAC) as shown below.

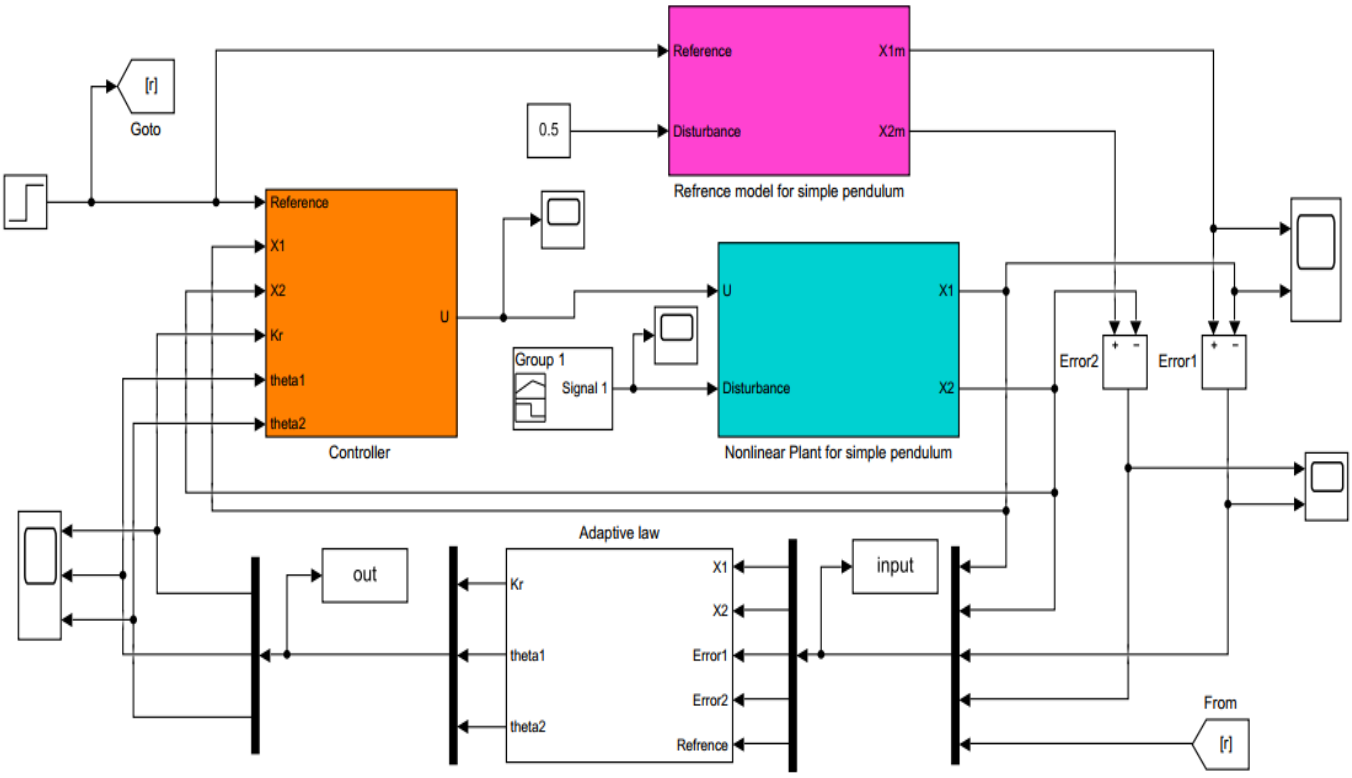


Figure 4. 9 The overall block diagram for Conventional MRAC for simple pendulum.

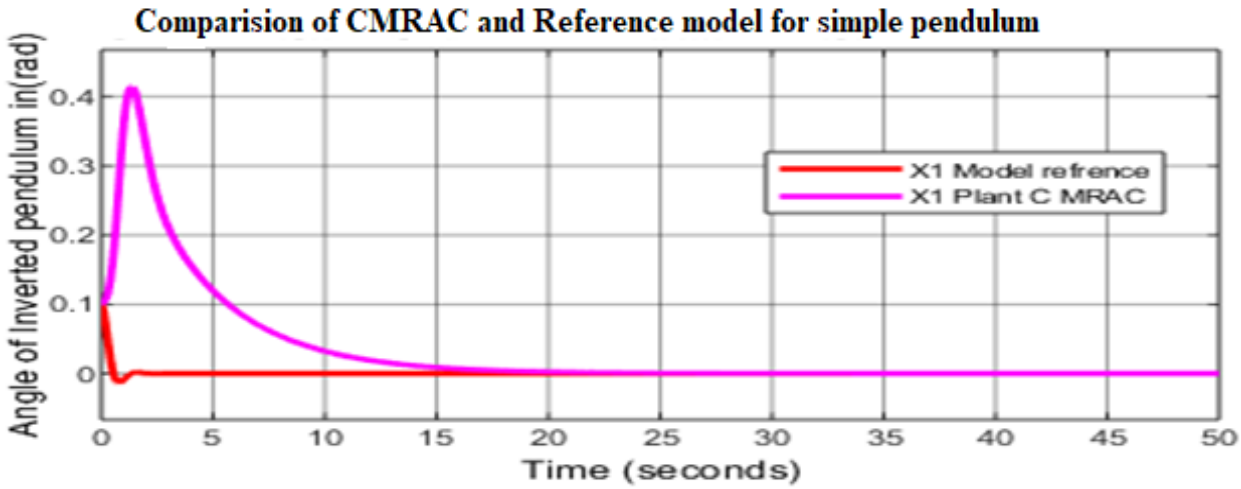


Figure 4. 10 Angle response for IP disturbed from 0.1rad using Conventional MRAC

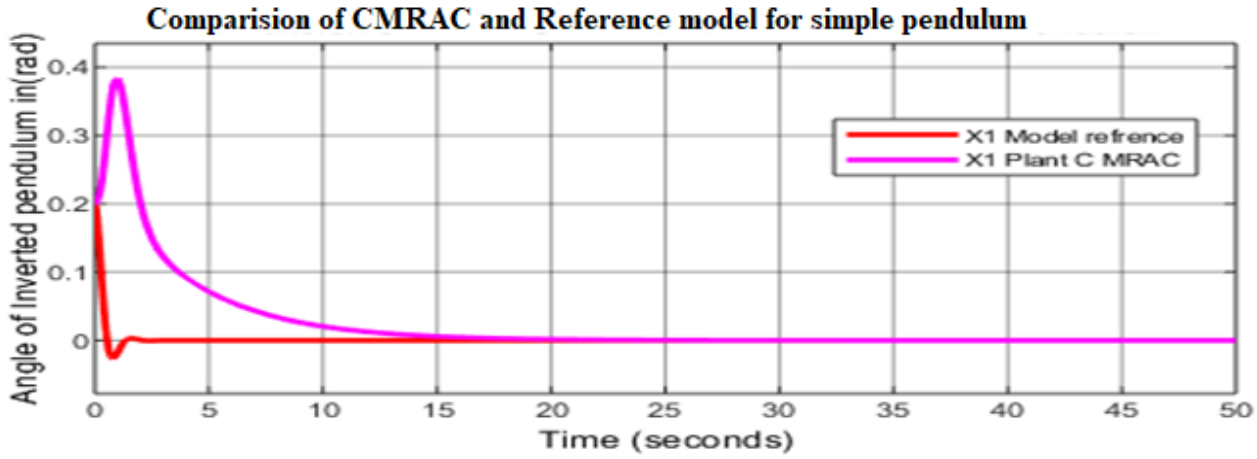


Figure 4. 11 Angle response for IP disturbed from 0.2rad using Conventional MRAC

For a given stable equilibrium, 0deg input as reference the nonlinear system response tracks to the reference model after 15.5 seconds. From the above, response for the nonlinear system the tracking error is large at the beginning so to reduce the tracking error Artificial neural network is preferable.

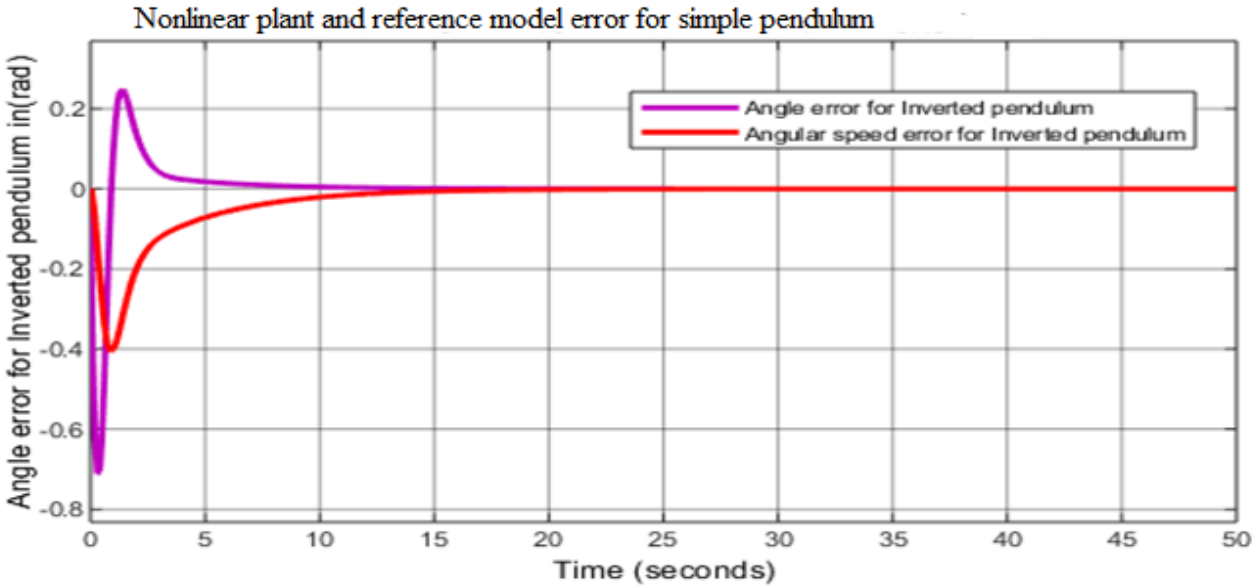


Figure 4. 12 Angle Error between nonlinear plant and reference model to stable equilibrium point.

Table 4. 2 The performance specification of simple pendulum system using a reference model (linearized model) for the desired response.

Performance characteristics	Reference model(desired response)
Rise time	0.3 second
Settling time	2.0 second
Undershoot	0.0002%

Based on Table 4.2 Specification of simple pendulum system using model reference adaptive control system for 0.1rad disturbance as shown below.

Table 4. 3 The performance specification of simple pendulum system using a reference model (linearized model) for 0.1rad disturbance

Performance characteristics	Nonlinear plant using MRAC
Rise time	5.42 second
Settling time	15.5 second
Undershoot	-

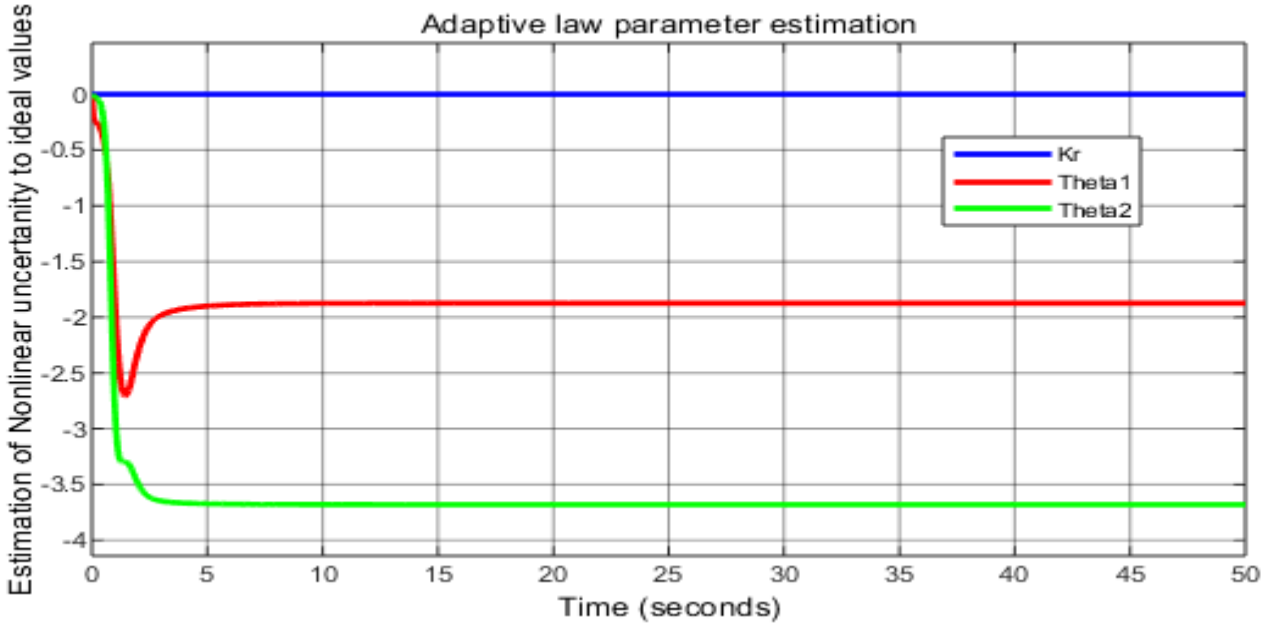


Figure 4. 13 Estimated adaptive law result for 0.1rad disturbance for IP

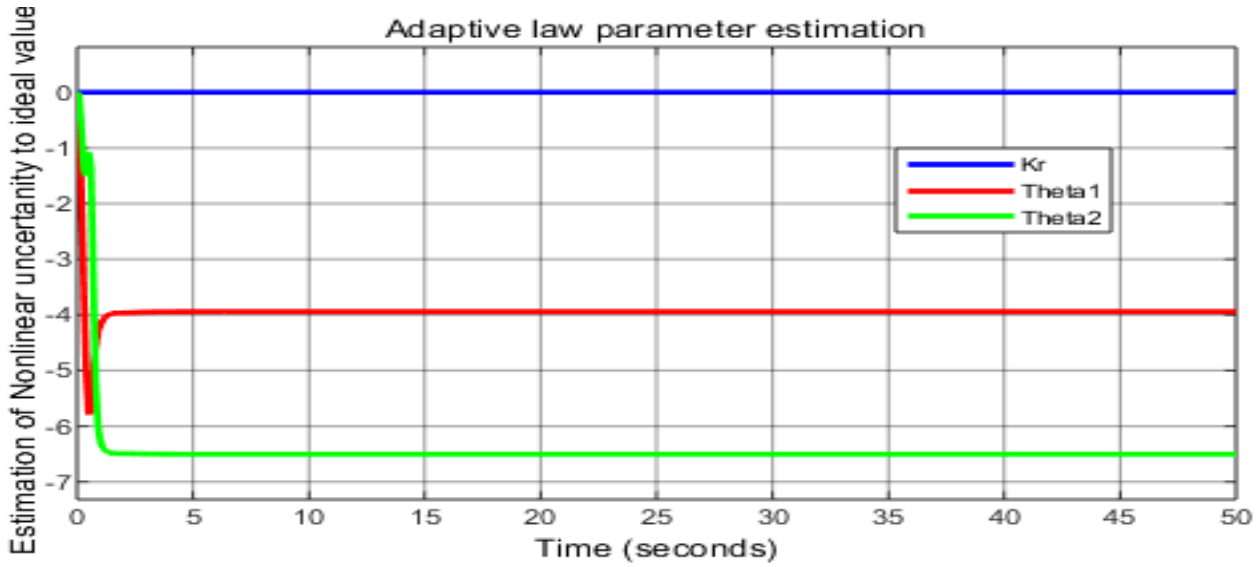


Figure 4.14 Estimated adaptive law result for 0.5rad disturbance for IP

Table 4. 4 The performance specification of simple pendulum system using a reference model (linearized model) for 0.2rad disturbance as shown below

Performance characteristics	Nonlinear plant using MRAC
Rise time	5.01 second
Settling time	15.52 second
Undershoot	-

Using conventional MRAC controller the nonlinear system is not reached a satisfactory performance; we can observe (Table 4.3 and Table 4.4) that conventional MRAC controller shows more settling time and steady state error.

To improve the performance of Conventional MRAC we can design neural network based MRAC. Both simple pendulum and Vander poll equation systems are the nonlinear and uncertain system. Designed for such systems using NN-MRAC controller is better than the conventional MRAC controller does since NN controller ability to learn, generalized and adapt. A controller is a design based on using MATLAB graphical user interface and MATLAB code.

4.5. Neural network based Direct MRAC for simple pendulum result

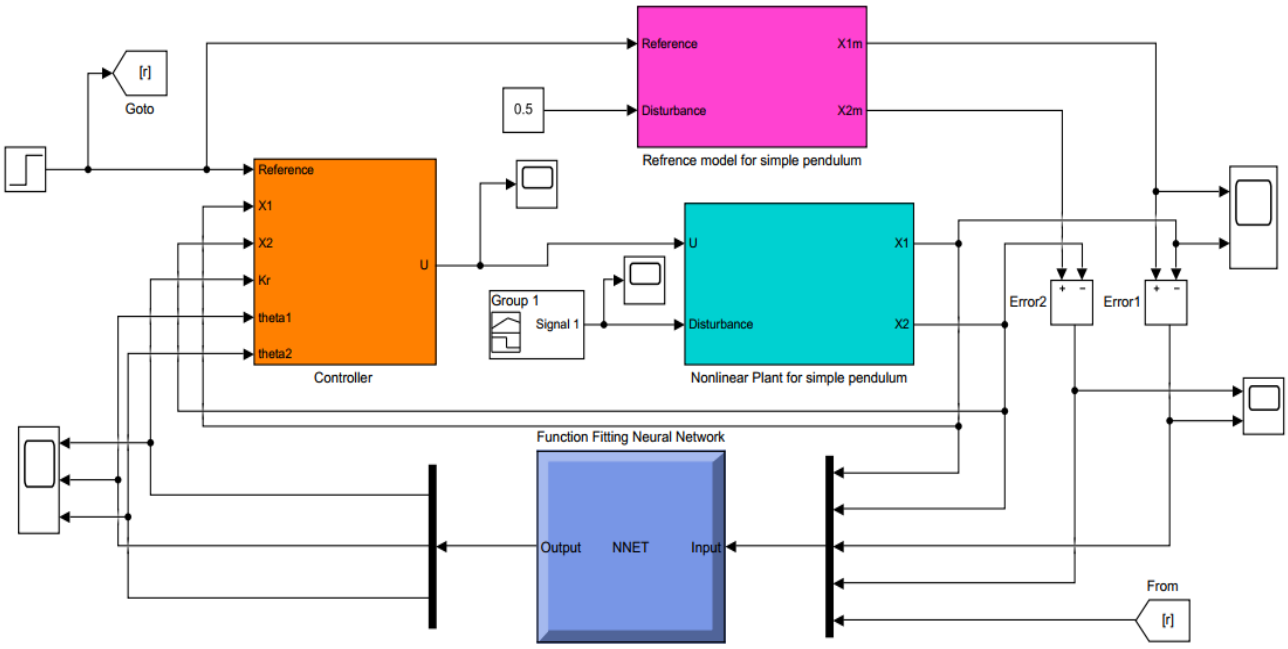


Figure 4. 15 MATLAB/SIMULINK model for simple pendulum using NN-MRAC

In this thesis, Artificial Neural Network is used to improve the performance of nonlinear system by training of adaptation laws to improve the plant due to rapidly estimating of the nonlinear uncertainty and unknown parameters to ideal values. Now NN is dividing into three layers, named the input layer with 5 neurons, the hidden layer with 20 neurons, and the output layer with 3 neurons. Backpropagation learning algorithm (Levenberg - Marquardt) is used to train the network. The ANN training data samples are collected from the input and output of Adaptation law. The hidden layer neurons are activated by using tan sigmoidal activation function and output a pure linear activation function is used.

To implement a Neural Network (design process), the following steps must be followed:

- Collect data (Load data source).
- Neural Network creation.
- Configure the network (selection of network architecture).
- Initialize the weights and biases.
- Train the network.
- Validate the network (Testing and Performance evaluation).
- Use the network.

After training the ANN the regression, error histogram and performance under plots shown in the figure below**.**

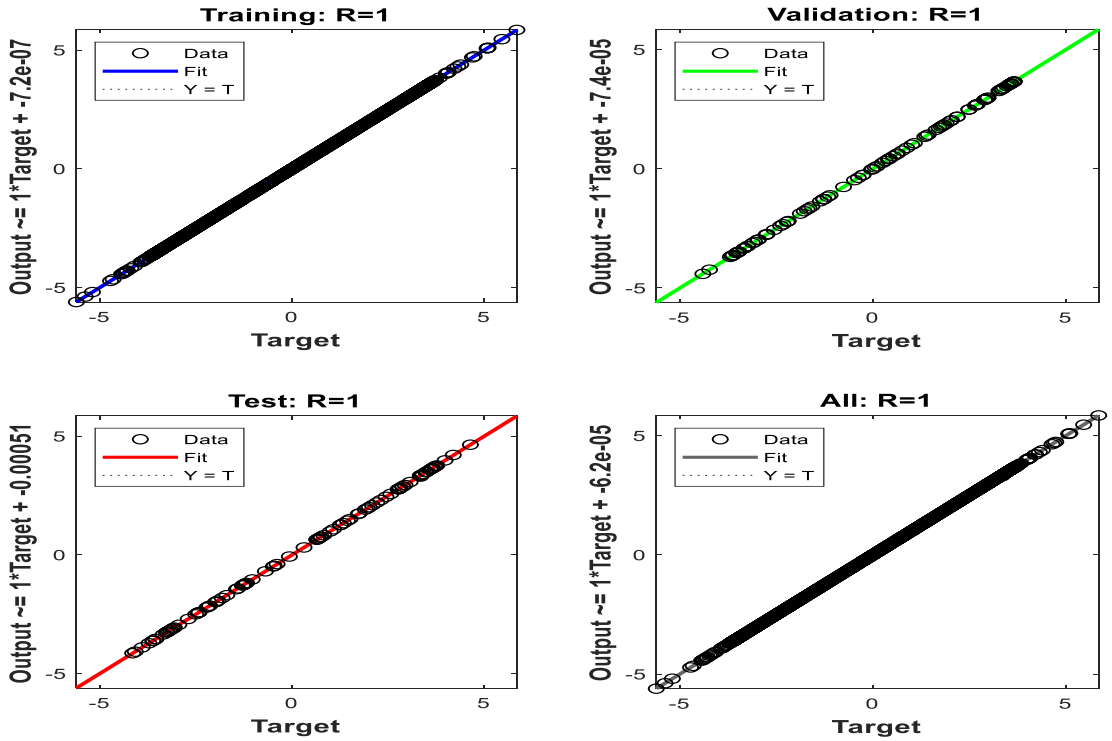


Figure 4. 16 Neural Network Training Regression Graph

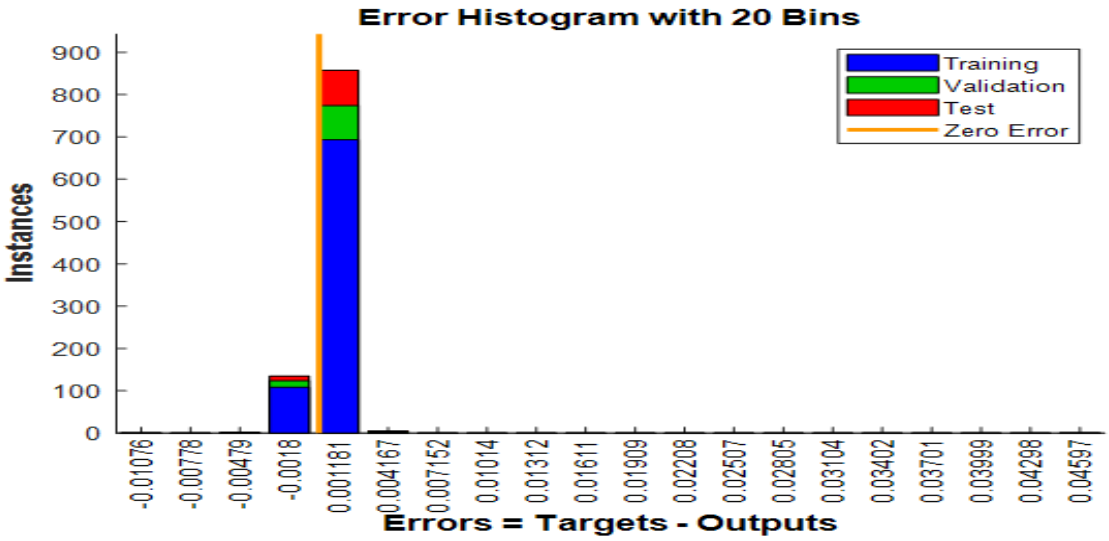


Figure 4. 17 Neural Network Training Histogram

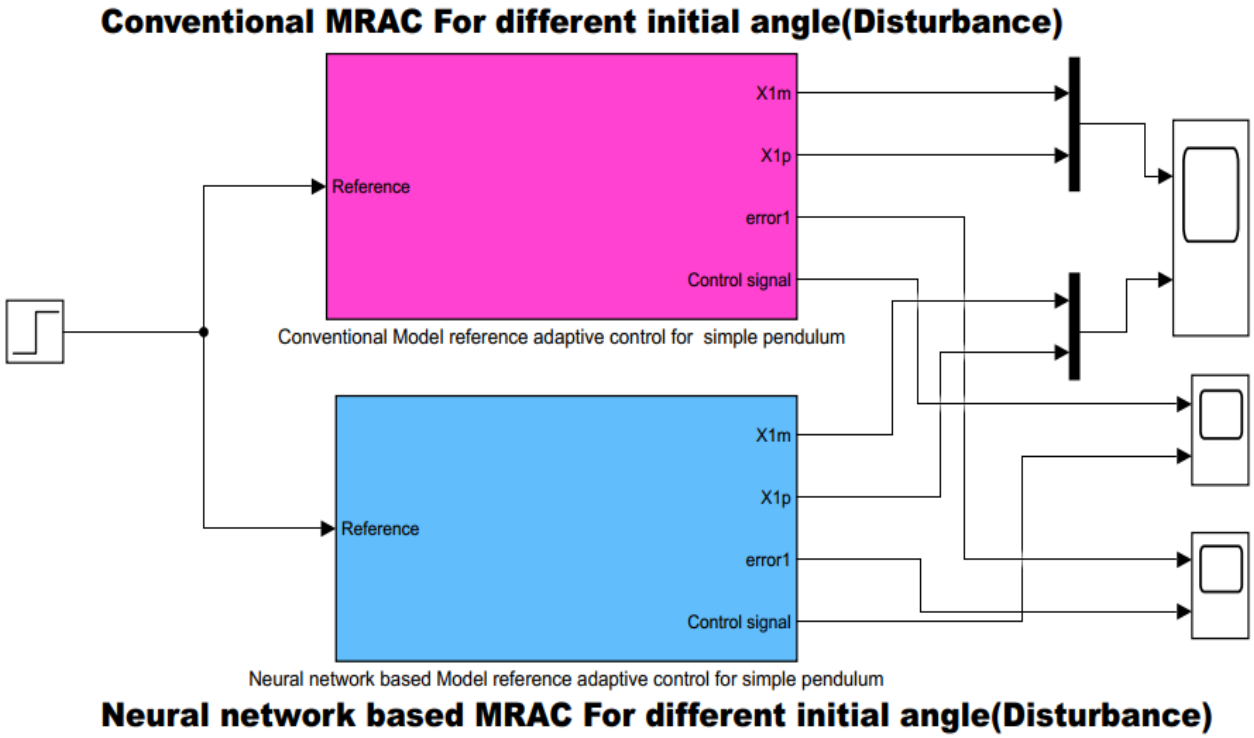


Figure 4. 18 Comparison of both conventional MRAC and Neural network based MRAC for IP.

Using NN based MRAC and Conventional MRAC design the simulation result for the simple pendulum system is illustrated in the following figures with the different disturbances.

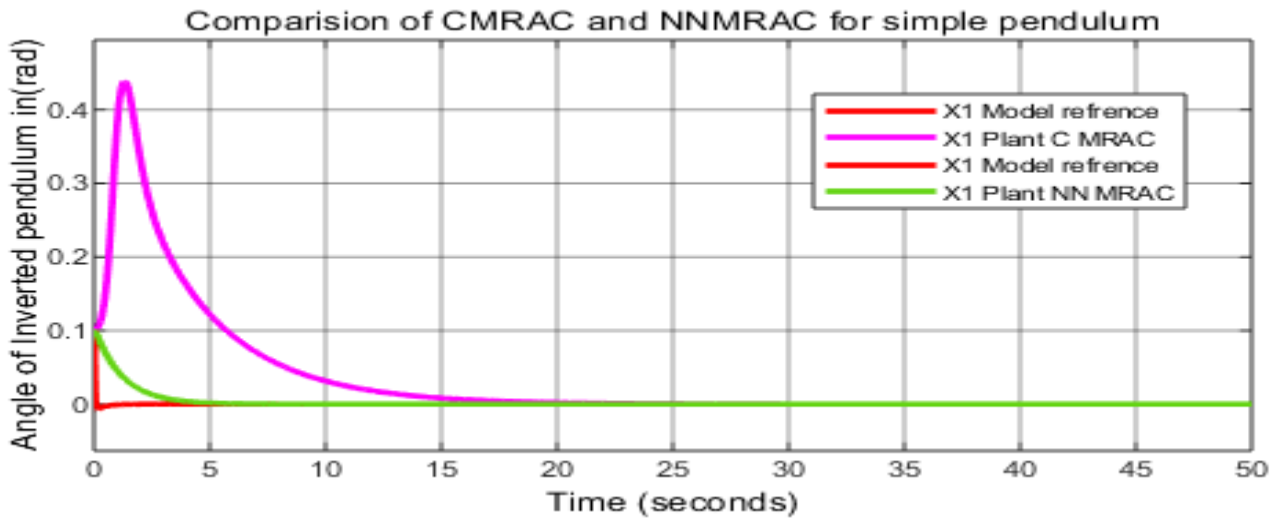


Figure 4. 19 Comparison of Conventional MRAC and NN-MRAC angle response for IP disturbed from 0.1rad.

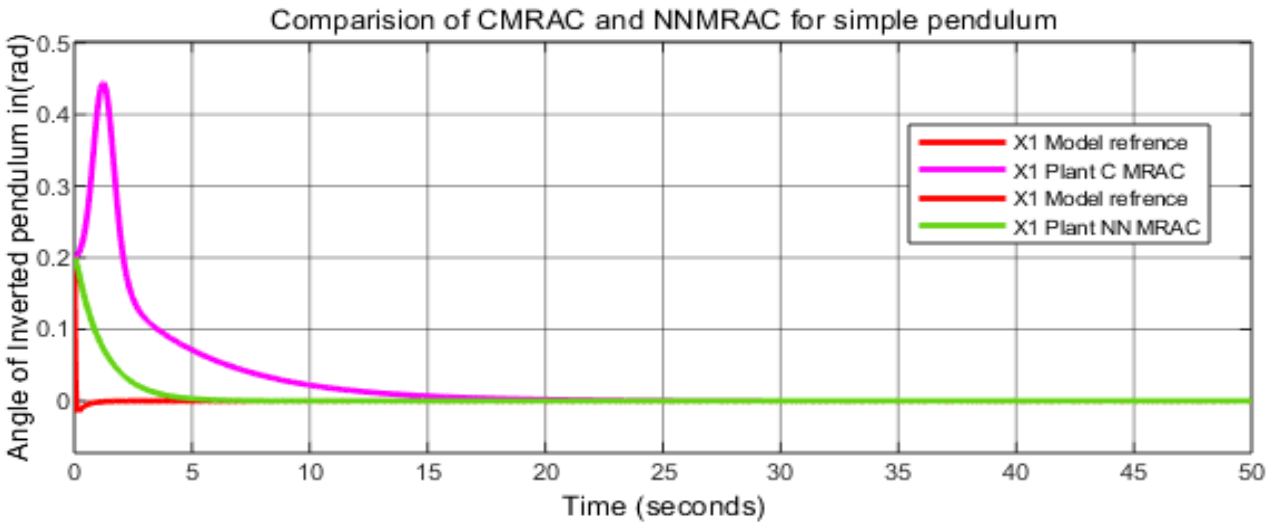


Figure 4. 20 Comparison of Conventional MRAC and NN-MRAC angle response for pendulum disturbed from 0.2rad.

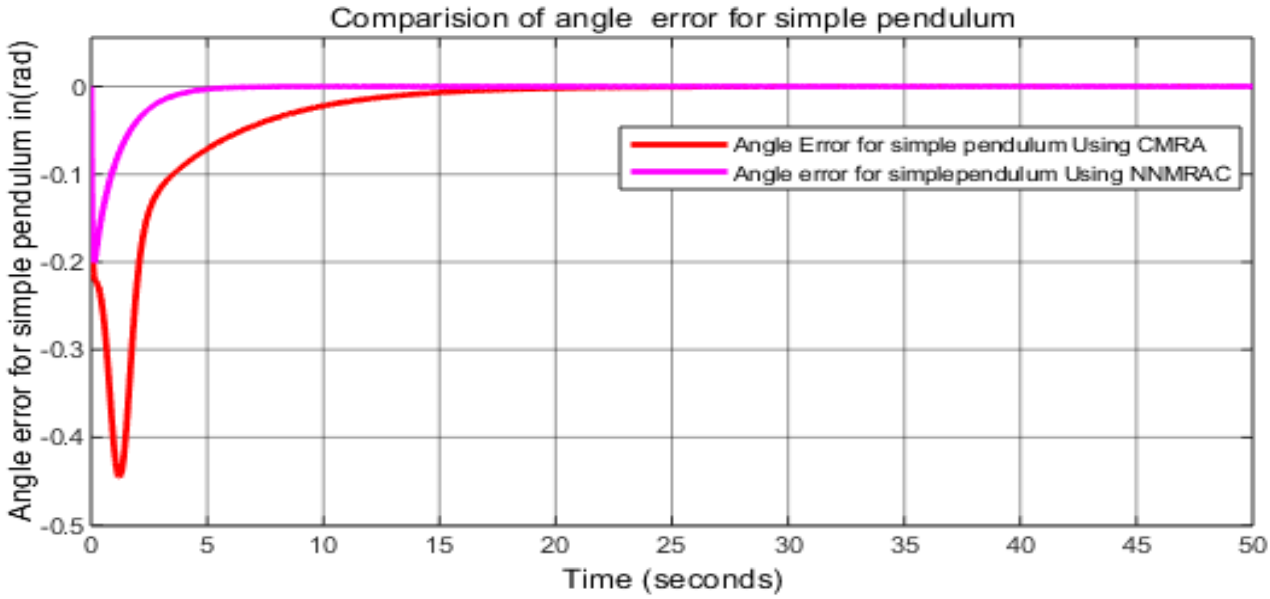


Figure 4. 21 Comparison of angle error for pendulum given 0.2rad disturbance

From the above the NN based direct model reference control is better performance based on time specification i.e. rise time, settling time and steady state error than conventional direct model reference adaptive control for different disturbance.

As we seen from the above Figure 4.21 the angle error for NN based, DMRAC is smaller than Conventional model reference adaptive control.

Generally given Disturbance as pulse input as shown below and see the result both Conventional MRAC and NN based direct model reference adaptive control.

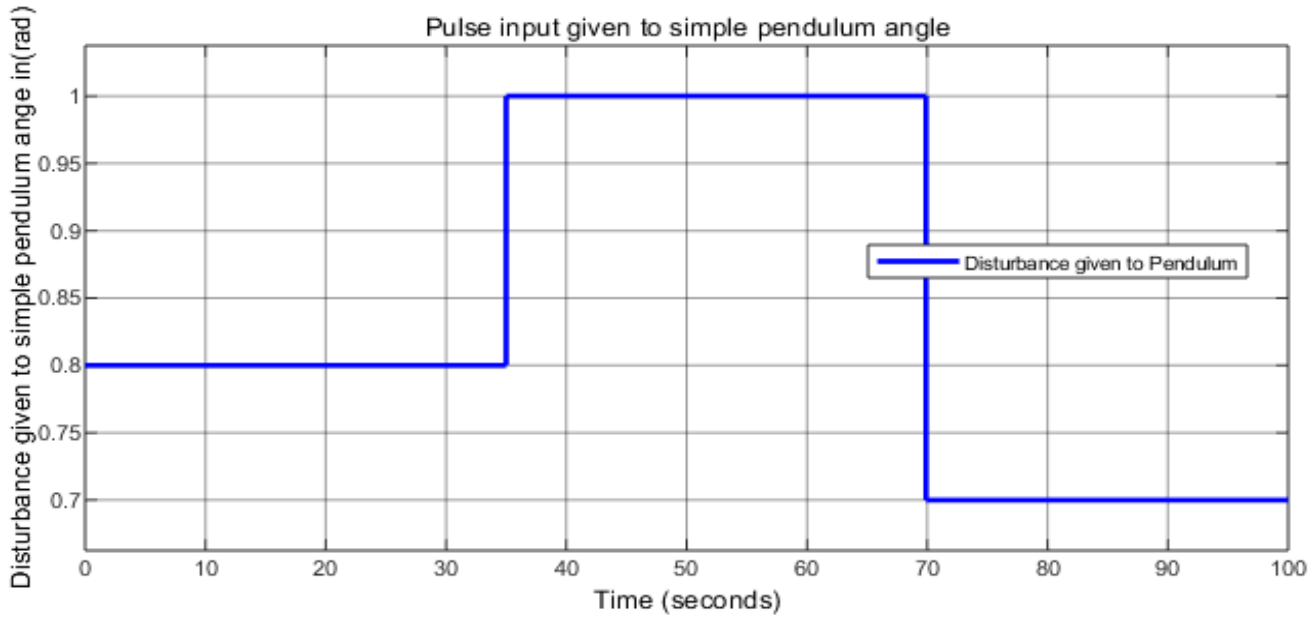


Figure 4. 22 Different disturbance given to the angle of Pendulum.

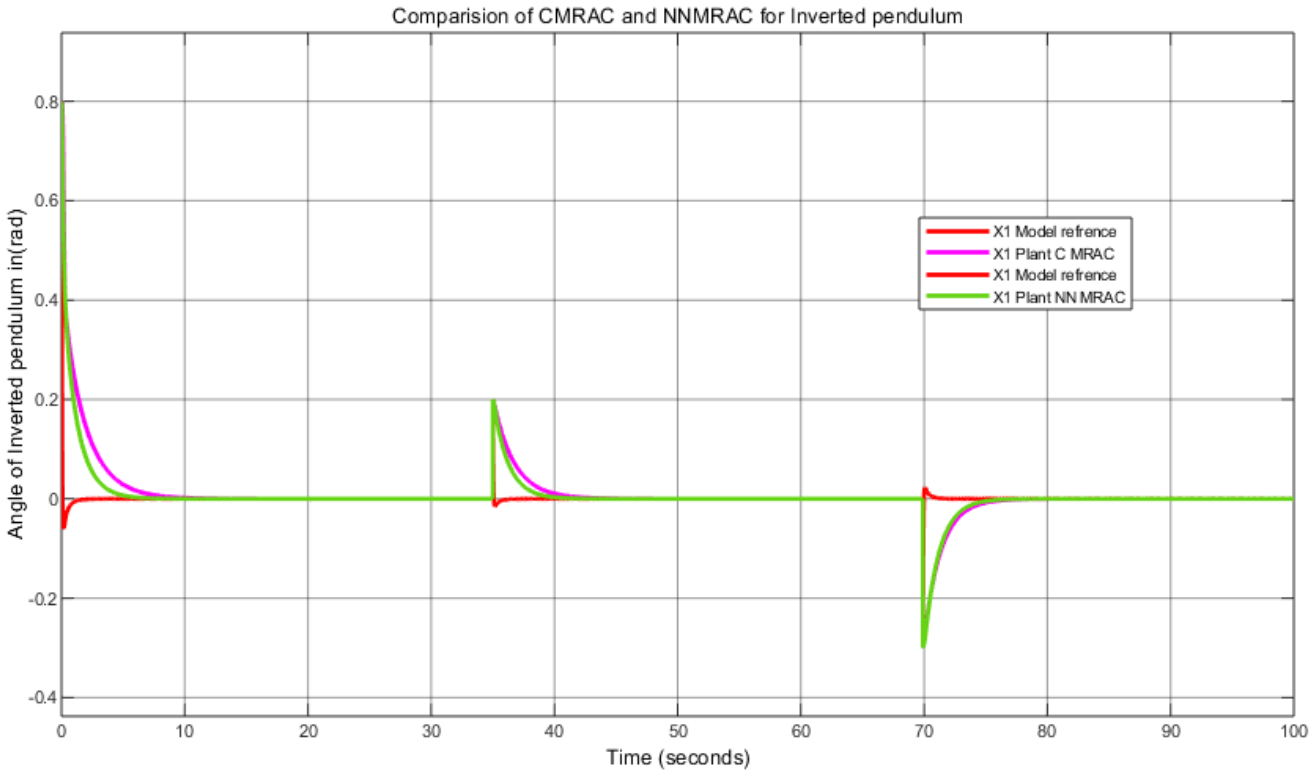


Figure 4. 23 Comparison of angle response for Different disturbance given to angle of IP

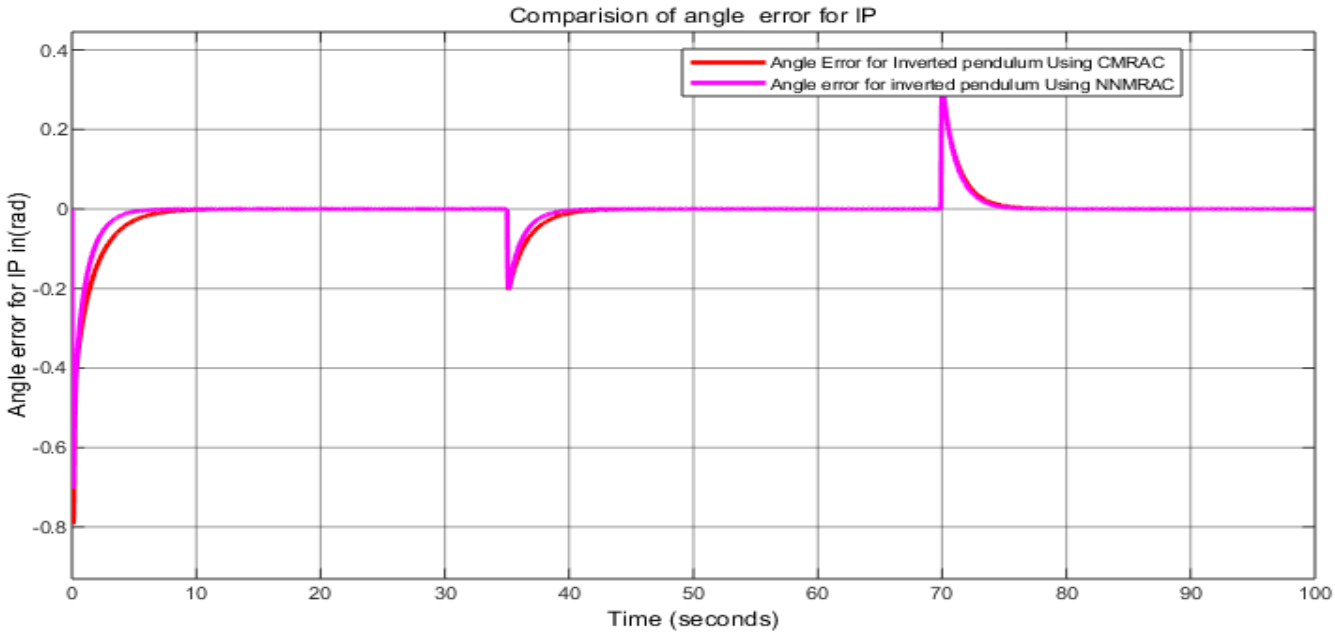


Figure 4. 24 Comparison of angle error for a different disturbance on the angle of IP.

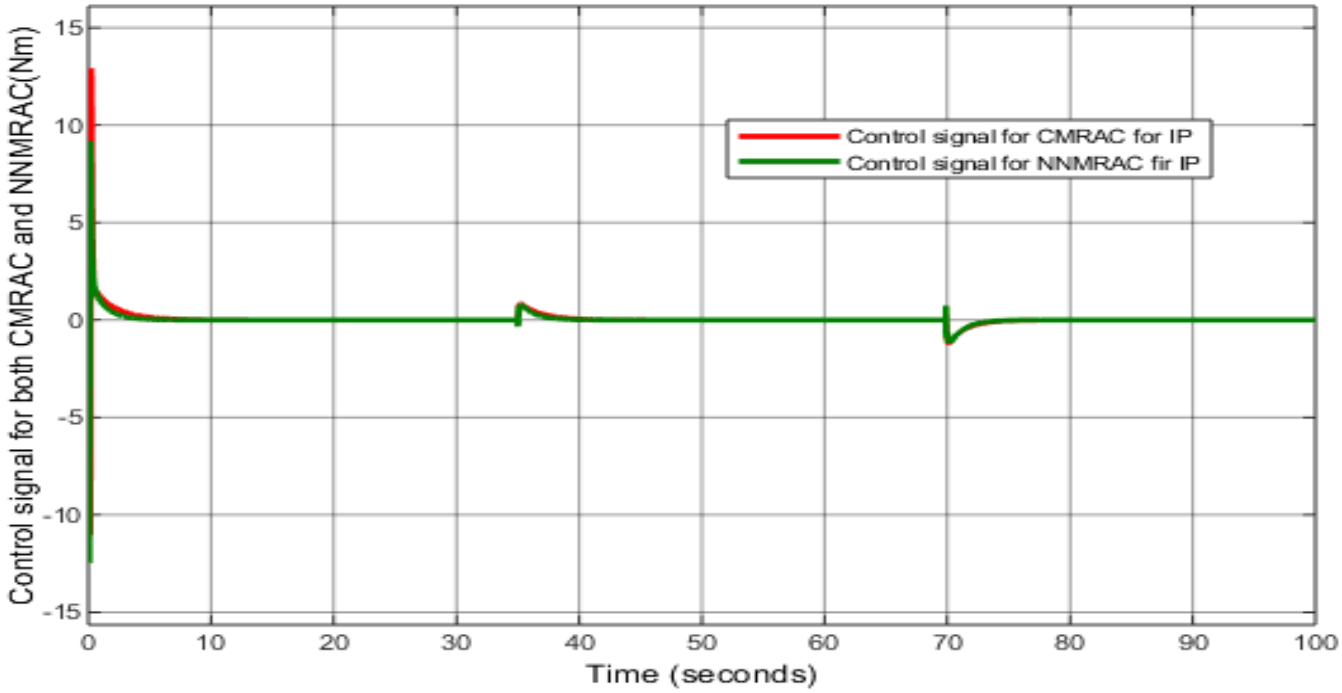


Figure 4. 25 Control signal for simple pendulum for the different disturbance.

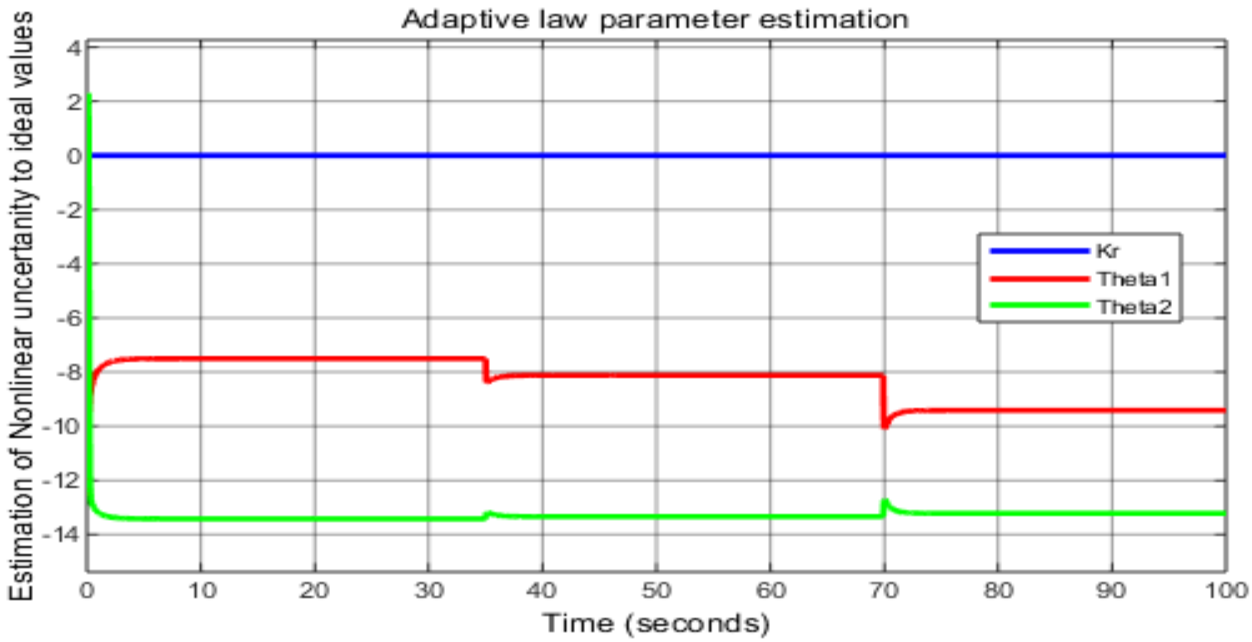


Figure 4. 26 Estimated adaptive law result for different disturbance for IP

Finally, we can observe from the above graphs an ANN controller is a powerful controller to stabilize a nonlinear simple pendulum system when compared to Conventional MRAC.

The conventional MRAC and an artificial neural network controller are designed for the stabilization of simple pendulum systems. The performance characteristics are compared in the following table below.

Table 4. 5 Performance characteristics for the simple pendulum system both Conventional MRAC and Neural network base DMRAC for 0.1rad disturbance.

Performance Characteristics	Conventional MRAC	NN based MRAC
Rise Time	5.42 sec	3.13 sec
Settling Time	15.5 sec	5.15 sec
Undershoot	-	-

Table 4. 6 Performance characteristics for the simple pendulum system both Conventional MRAC and Neural network base DMRAC for 0.2rad disturbance.

Performance Characteristics	Conventional MRAC	NN based MRAC
Rise Time	5.01sec	3.12 sec
Settling Time	15.52sec	5.21sec
Undershoot	-	-

Using ANN controllers, we can observe from the table that the best performance evaluation is obtained when compared to conventional MRAC, since having a very small rise time, overshoot, peak response, steady-state error and settling time.

4.6. MATLAB Simulation result for Vander Poll oscillation

Similarly, for Vander poll oscillator on electric circuit especially tunnel diode.

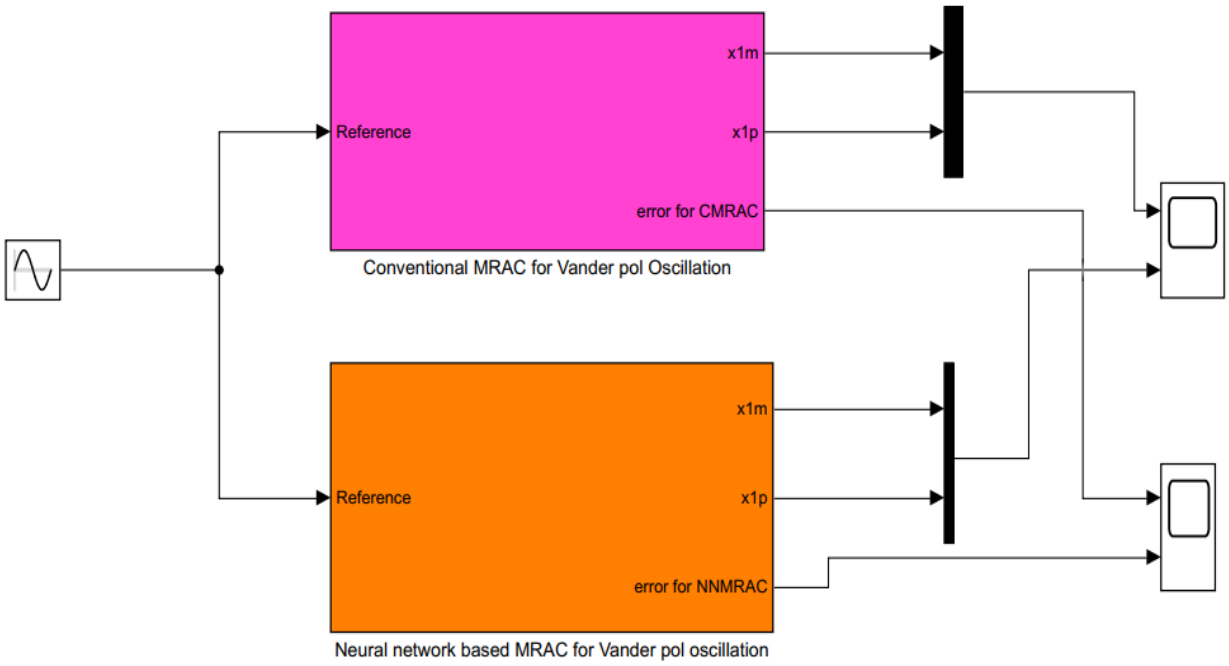


Figure 4. 27 The overall block diagram for Vander poll oscillator

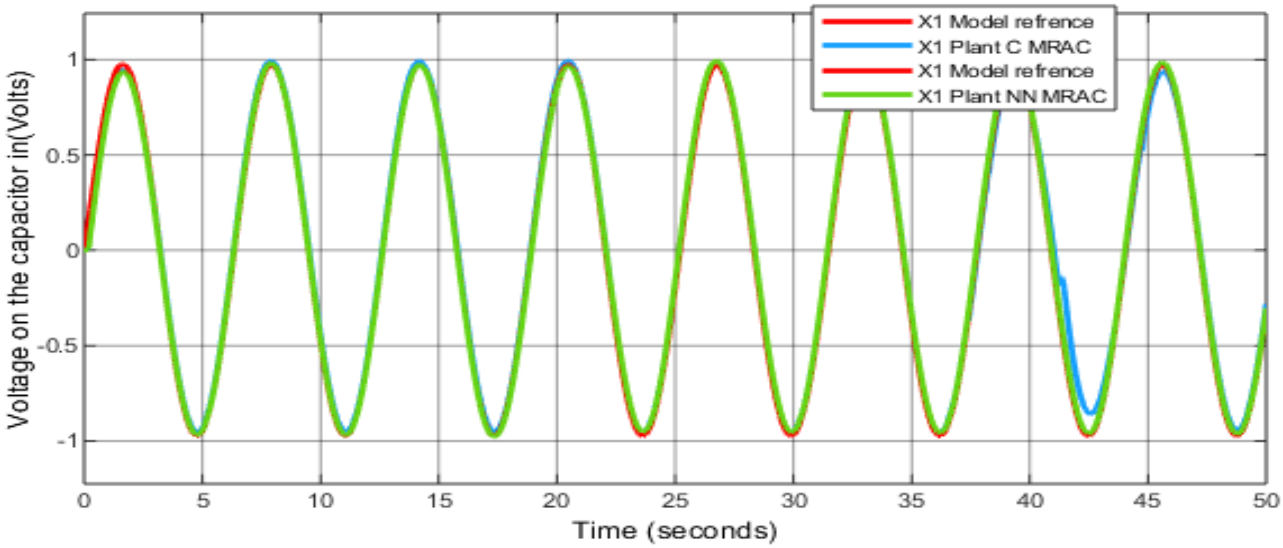


Figure 4. 28 Comparison for Vander poll given reference=sin (t)

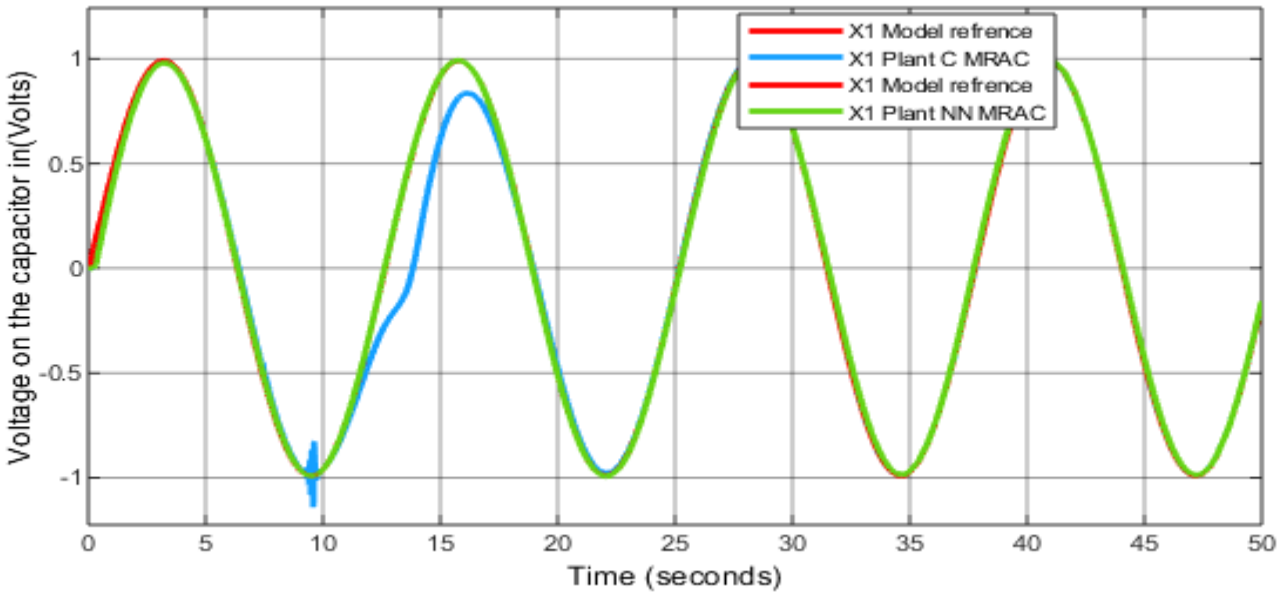


Figure 4. 29 Comparison for Vander poll given reference=sin (0.5t)

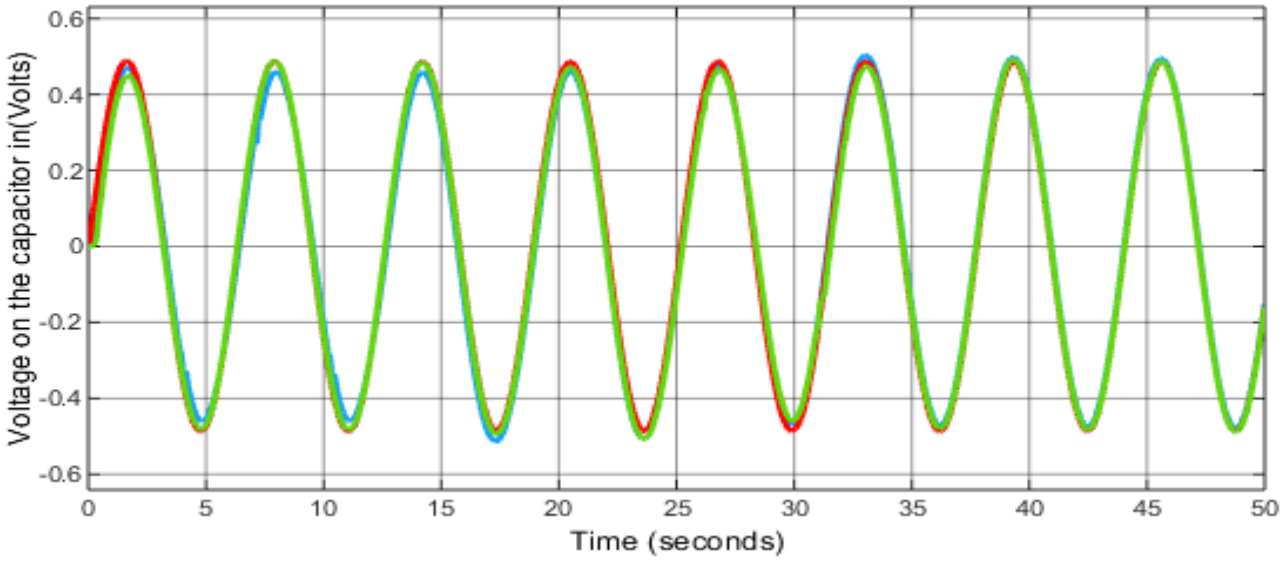


Figure 4. 30 Comparison for Vander poll given reference=0.5sin (t)

From the above Figure, we can conclude that for different sinusoidal reference input to the nonlinear van der pol oscillation; the output voltage on the capacitor perfectly tracks the reference input for neural network based direct MRAC than conventional direct MRAC due to generalize ,adaptively adjust the network parameters and learning ability of the Artificial neural network.

Due to the nonlinearity of the plant, the tracking response for using conventional MRAC is slower than NN based DMRAC by varying the frequency of the reference signal.

Chapter Five

5. Conclusion and Recommendation

5.1 Conclusion

In this thesis, modeling and designing of neural network based direct model reference adaptive control of nonlinear systems using MATLAB software to overcome the tracking performance of conventional direct model reference adaptive control to an equilibrium point for different disturbances have been investigated. Adaptive law using Lyapunov stability criteria for updating the controller parameters online has been formulated.

The simulation result for neural network based direct model reference adaptive control shows good performance compared to conventional model reference adaptive control for regulating output. The result of the conventional model reference adaptive controller is not satisfactory to the highest degree of accuracy condition even it introduces a steady state error in the system.

The neural network controller is proposed to update the parameters of the controller for both simple pendulum and Vander poll oscillation, which has the advantages of conventional MRAC.

Both the transient and steady state performances of the controller are improved by updating the parameters of neural networks (weight and biases).

The performance of the proposed NN based DMRAC and Conventional MRAC is tested through simulation studies using MATLAB/SIMULINK. It is observed from the simulation results that the proposed neural network based Direct MRAC has 3.13sec rise time, 5.15sec settling time for 0.1rad disturbance and 3.12sec rise time, 5.21sec settling time for 0.2rad disturbance. Whereas, the conventional direct model reference adaptive control has 5.42sec rise time, 15.5sec settling time for 0.1rad disturbance and 5.01sec rise time, 15.52sec settling time for 0.2rad disturbance**.** It is shown that the proposed neural network based Direct MRAC has smaller rising time, steady-state error and settling time for a different disturbance than Conventional DMRAC adaptive control. The proposed system is applicable for robotics manipulator to perform its task for minimum energy and time using neural network based direct model reference adaptive control than Conventional MRAC under different disturbance.

The proposed NN based DMRAC is able to maintain the angle of the simple pendulum for different disturbance and different input for Vander poll oscillation within tolerable limits in spite of the unknown parameter variation, unknown nonlinear system and nonlinear uncertainty.

5.2 Recommendation

It can be noted from simulation results that the neural network based model reference adaptive controller for a nonlinear system gives better results as compared to the conventional model reference adaptive controller. Therefore, to validate these simulation results, it is recommend the implementation of the NN based MRAC on the actual operation of a simple pendulum. In addition, further research should focus on extending this scheme to a higher order nonlinear system like autopilot, quadcopters and induction machine modeling for better performance control due to artificial intelligence with that of adaptive control.

Reference

[1] Eugene Laveretsky, Kevin A.Wise.''Robust and Adaptive control with Aerospace Application'' Springer-Verlag London 2013, pp.317-353.

[2] Nhan T. Nguyen," Model reference adaptive control" springer international publishing AG 2018.

[3] Kalpesh B. Pathak, Dipak M. Adhyaru," Survey of Model Reference Adaptive Control" IEEE International Conference, 06-08 Dec 2012.

[4] Arya V A, Aswin R B, Ashni Elisa George." Modified Model Reference Adaptive Control for the Stabilization of Cart Inverted Pendulum System" International Research Journal of Engineering and Technology (IRJET), IRJET Volume: 05 Issue: 04 Apr-2018.

[5] Pawar, R. J., and B. J. Parvat. "Design and implementation of MRAC and modified MRAC technique for the inverted pendulum." In *2015 International Conference on Pervasive Computing (ICPC)*, pp. 1-6. IEEE, 2015.

[6] Prakash, R., and R. Anita. "Design of Model Reference Adaptive Intelligent Controller Using Neural Network for Nonlinear Systems." *International Review of Automatic Control* 4, no. 2 (2011): 153-161.

[7] Caudill, Maureen. "Neural networks primer, part I." *AI expert* 2, no. 12 (1987): 46-52

[8] Parks, P.C, "Lyapunov Redesign of Model reference adaptive control system", *IEEE Trans on Automatic Control, Vol.AC-11,* No.3 (1966). pp.362-367.

[9] Hang, Chang-Chieh, and P. C. Parks. "Comparative studies of model reference adaptive control systems." *IEEE transactions on automatic control* 18, no. 5 (1973): pp 419-428.

[10] Swarnkar, Pankaj, Shailendra Jain, and R. K. Nema. "Effect of adaptation gain in model reference adaptive controlled second order system." *Engineering, Technology & Applied Science Research* 1, no. 3 (2011): pp 70-75

[11] Zhihong, Man, X. H. Yu, K. Eshraghian, and M. Palaniswami. "A robust adaptive sliding mode tracking control using an RBF neural network for robotic manipulators." In *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 5, pp. 2403-2408. IEEE, 1995.

[12] Hunt, K. Jetal, D. Sbarbaro, R. Żbikowski, and Peter J. Gawthrop. "Neural networks for control systems—a survey."*Automatica* 28, no. 6 (1992): pp.1083-1112.

[13] Zhihong, Man, Hong Ren Wu, and Marimuthu Palaniswami. "An adaptive tracking controller using neural networks for a class of nonlinear systems." *IEEE transactions on neural networks* 9, no. 5 (1998):pp 947-955.

[14] Munadi, M. Amirullah Akbar, To mohide Naniwa, Yoshiaki Taniai ," Model Reference Adaptive Control for DC Motor Based on Simulink", *2016 6th International Annual Engineering Seminar (InAES).IEEE Conferences*,2016,pp.101-106.

[15] K. Pirabakaran and V.M. Becerra," PID autotuning using neural networks and model reference adaptive control **",** 15th Triennial World Congress Copyright © 2002 IFAC.

[16] Karl J. Astrom, Bjorn Wittenmark, "Adaptive Control" Pearson Education India, 2nd Edition, 2001.

[17] Adrian-Vasile Duka, Stelian Emilian Oltean, Mircea Dulau." Model Reference Adaptive Control and Fuzzy Model Reference Learning Control for the Inverted Pendulum. Comparative Analysis." *International Conference on dynamical systems and control, Venice, Italy, November 2-4*, 2005 (pp168-173).

[18] NemaR.K, Swarnkar Pankaj." Comparative Analysis of MIT Rule and Lyapunov Rule in Model Reference Adaptive Control Scheme" Innovative Systems Design and Engineering. Vol 2, No 4, 2011.

[19] Dr. Qadri Hamarsheh."Lectures for Neural networks and Fuzzy logic" at Philadelphia University on 2015/2016.

[20] James A. Freeman, David M. Skapura." Neural Networks Algorithms, Applications, and Programming Techniques" Addison-Wesley Publishing Company.

[21] Simon Haykin."Neural networks a comprehensive Foundation second edition."McMaster University Hamilton, Ontario, Canada.

[22] Martin T.Hagan, Howard B.Demuth,MarkBeale."Neural network design" By PWS company in 1996.

[23] Kanada Elektroingenieur, and Simon S. Haykin. "*Neural networks and learning machines*". Vol. 3. Upper Saddle River: Pearson education, 2009.

[24] Khalil, Hassan K., and Jessy W. Grizzle. *Nonlinear systems*. Vol. 3. NJ: Prentice Hall, 2002.

[25] R. Hedjar." Online Adaptive Control of Non-linear Plants Using Neural Networks with Application to Temperature Control System." King Saud Univ., Vol. 19, Comp. & Info. Sci., pp. 75-94, April 2007.

[26] R.M. Sanner and J.J.E. Slotine," Gaussian networks for direct adaptive control*", IEEE Trans. Neural networks, vol.3* pp.837-863, 1992.

[27] Patino, H. Daniel, and Derong Liu. "Neural network-based model reference adaptive control system." *IEEE Transactions on Systems, Man, and Cybernetics*, no.1 (2000): 198-204

[28] S. S. Ge, C. C. Hang, and T. Zhang, "A direct method for robust adaptive nonlinear control with guaranteed transient performance," *Syst. Control Lett.*, vol. 37, no. 5, pp. 275-284, Aug. 1999.

[29] Lavretsky, Eugene (2015). "Robust and Adaptive Control Methods for Aerial Vehicles". Handbook of Unmanned Aerial Vehicles. pp. 675–710.

[30] F.C. Chen and H.K. Khalil," Adaptive control of nonlinear systems using neural networks" In Proc. Decision Contr., 1990, pp.1707-1712.

[31] https://courses.lumenlearning.com/suny-osuniversityphysics/chapter/15-4-pendulums/

[32] Ulasyar, Abasin, Haris Sheh Zad, Adil Zohaib, and Syed Shahzad Hussain. "Adaptive radial basis function neural network based tracking control of Van der Pol oscillator." In *2017 International Conference on Communication Technologies*, pp. 111-115. IEEE, 2017.

[33] Min-an, Tang, Wang Xiao-Ming, Cao Jie, and Cao Li. "On Lyapunov stability theory for model reference adaptive control." In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pp. 1055-1060. IEEE, 2017

Appendix

Appendix A: MATLAB code for linearized Simple Pendulum and VPO Linearized Model for Simple Pendulum

```
M=0.5; %Mass of the pendulum
g=9.8;%Gravity constant g=9.8 m/s^2
len=1;%Length of the rod from the pivot to pendulum mass
b=0.5;%Friction of the pendulum (Damping coefficient)
c=M*(len^2);
d=M*g*len/c;
A=[0 1;-d -b/c];%Linearized state space matrix
B=[0;1/c];
C=[1 0];
D=0;
gammar=10; %Learning rates
gammatheta1=10;
gammatheta2=10;
R=0.01;
Q=C'*C;
[K]=lqr(A,B,Q,R);%LQR controller
k1=K(:,1);
k2=K(:,2);
Ac=A-B*K;
Bc=B*k1;
P=lyap(Ac,Q);%Lyapunove equation
p11=P(1,1);p12=P(1,2);
p21=P(2,1);p22=P(2,2);
[num1,den1]=ss2tf(A,B,C,D);
[num2,den2]=ss2tf(Ac,Bc,C,D);
sys1=tf(num1,den1);
sys2=tf(num2,den2);
subplot(2,1,1)
step(sys1)
xlim([0 50]) %x axis limit from 0 to 50
title('Step response before controller for linearized simple pendulum')
subplot(2,1,2)
hold on
step(sys2)
xlim([0 50]) %x axis limit from 0 to 50
title('Step response after controller for linearized simple pendulum')
```

Linearized model for Vander Poll Oscillation

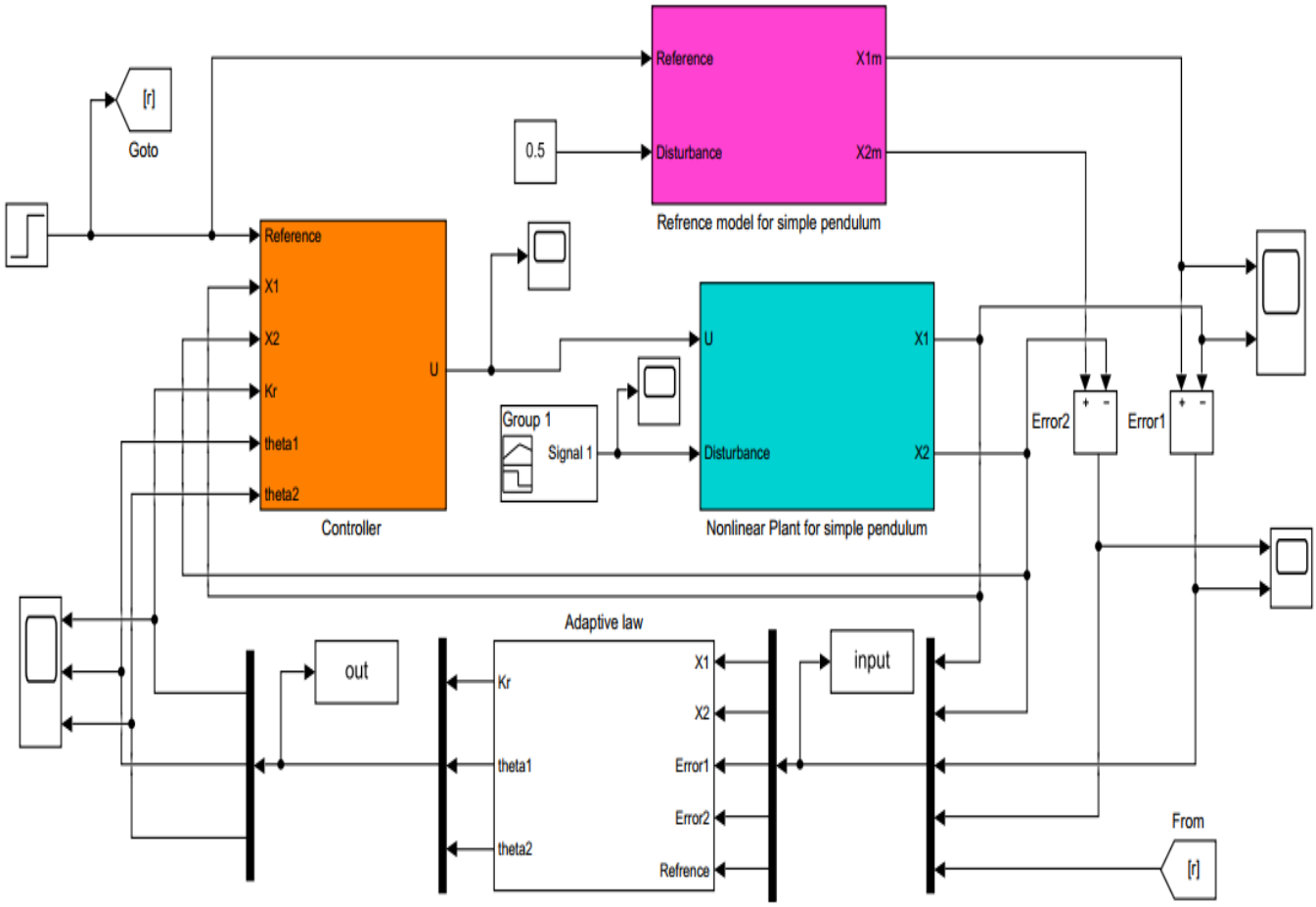
```
%Cap (d^2 V_C1)/(dt^2 )+(-1+V_C1^2)(dV_C1)/dt+V_C1/L=U
Cap=0.1;%Capacitor in micro-henry (µF).
L=100;%Inductor in milli-henry (mH)
q1=1/Cap;
q2=1/(Cap*L);
q3=1/Cap;
gammar=500; %Learning rates
gammatheta1=500;
gammatheta2=500;
A=[0 1;-q2 q1];%Matrix A for linearized system V_O
B=[0;q3];%Input matrix for linearized V_O
C=[1 0];
D=[0];
R=0.5;
Q=C'*C;
K=lqr(A,B,Q,R);%LQR gain calculation
k1=K(:,1);
k2=K(:,2);
Am=A-B*K;
Bm=B*k1;
P=lyap(Am,Q);%Lyapunove equation  for V_O
p11=P(1,1);p12=P(1,2);
p21=P(2,1);p22=P(2,2);

[num1,den1]=ss2tf(A,B,C,D);
[num2,den2]=ss2tf(Am,Bm,C,D);
sys1=tf(num1,den1);
sys2=tf(num2,den2);
subplot(2,1,1)
step(sys1)
xlim([0 25])%X axis limit from 0 to 50
ylim([0 inf])
title('Step response before controller for linearized Vander poll Oscillation')
subplot(2,1,2)
step(sys2)
xlim([0 25])%X axis limit from 0 to 50
title('Step response after controller for linearized Vander poll Oscillation')
```

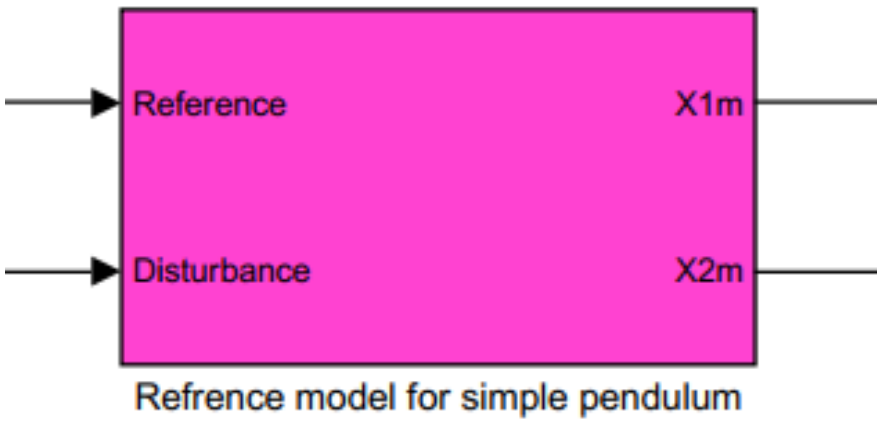
Appendix B: Neural network training code for Simple pendulum and VPO.

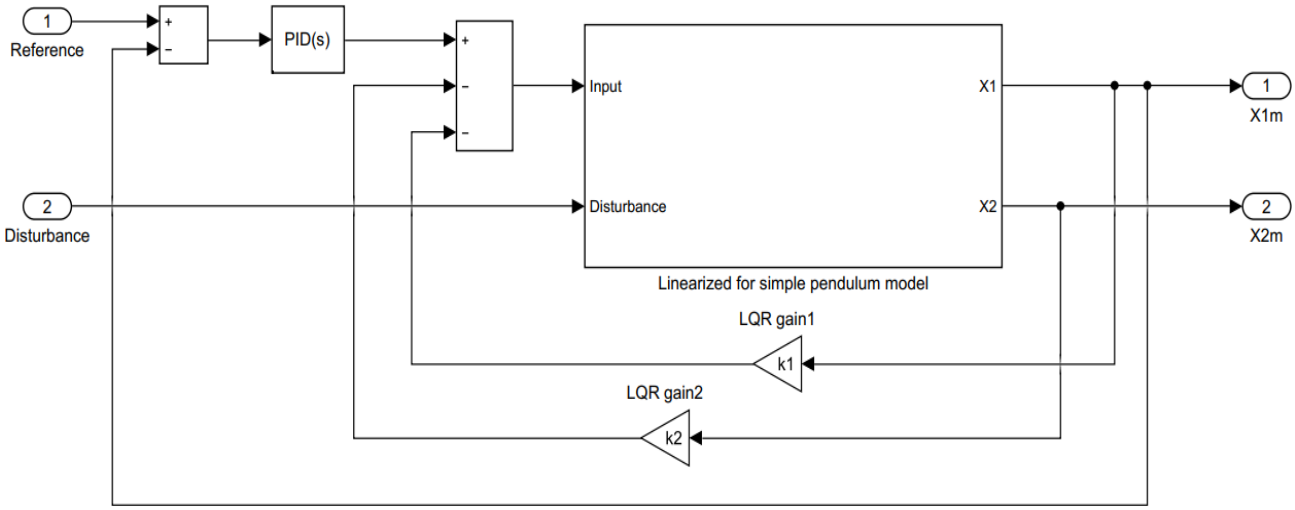
```
x = input'; % input data from work space.
t = out'; % target data from workspace.

% Choose a Training Function
% 'trainlm' is usually fastest.
trainFcn ='trainlm';  % Levenberg-Marquardt backpropagation.
% Create a Fitting Network
hiddenLayerSize = 20;%By varying the number of hidden neuron size
% we can see the performance of the fitting
net = fitnet(hiddenLayerSize,trainFcn);
% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 80/100;
net.divideParam.valRatio = 10/100;
net.divideParam.testRatio = 10/100;
% Train the Network
[net,tr] = train (net, x ,t);
% Test the Network
y = net(x);
e = gsubtract (t,y);%takes two matrices or cell arrays, and subtracts them in an element-wise
manner.
Performance = perform (net,t,y)

% View the Network
%view (net)
genism (net) %To generate neural network block.
```

Appendix C: MATLAB Simulink for NN based Direct MRAC for Pendulum.

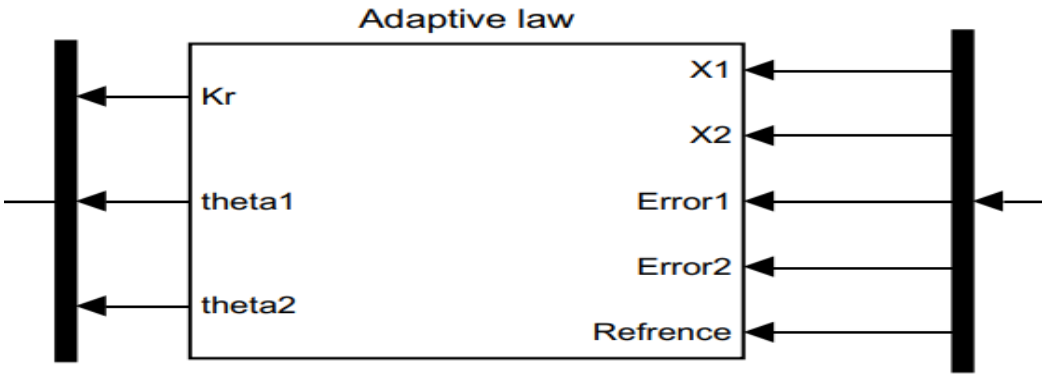
The general block diagram for simple pendulum using CMRAC

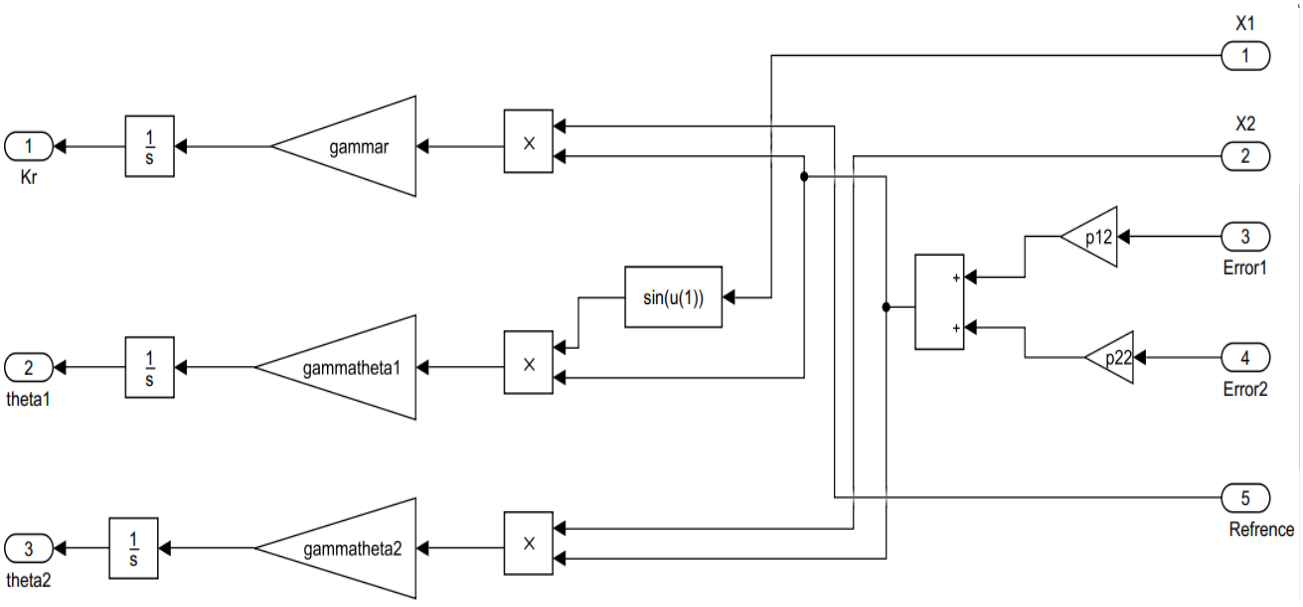




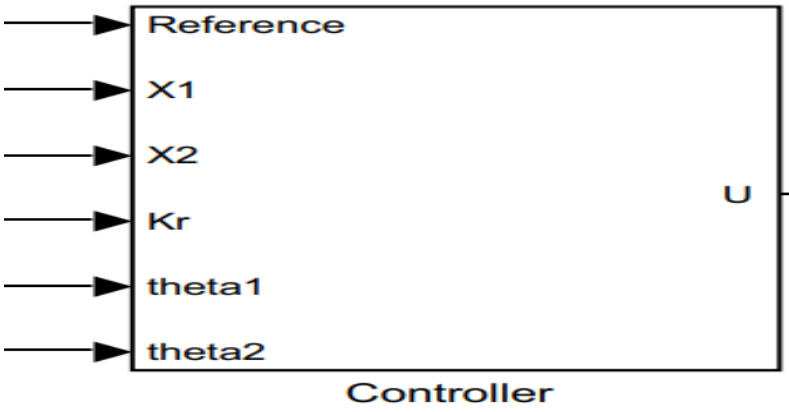


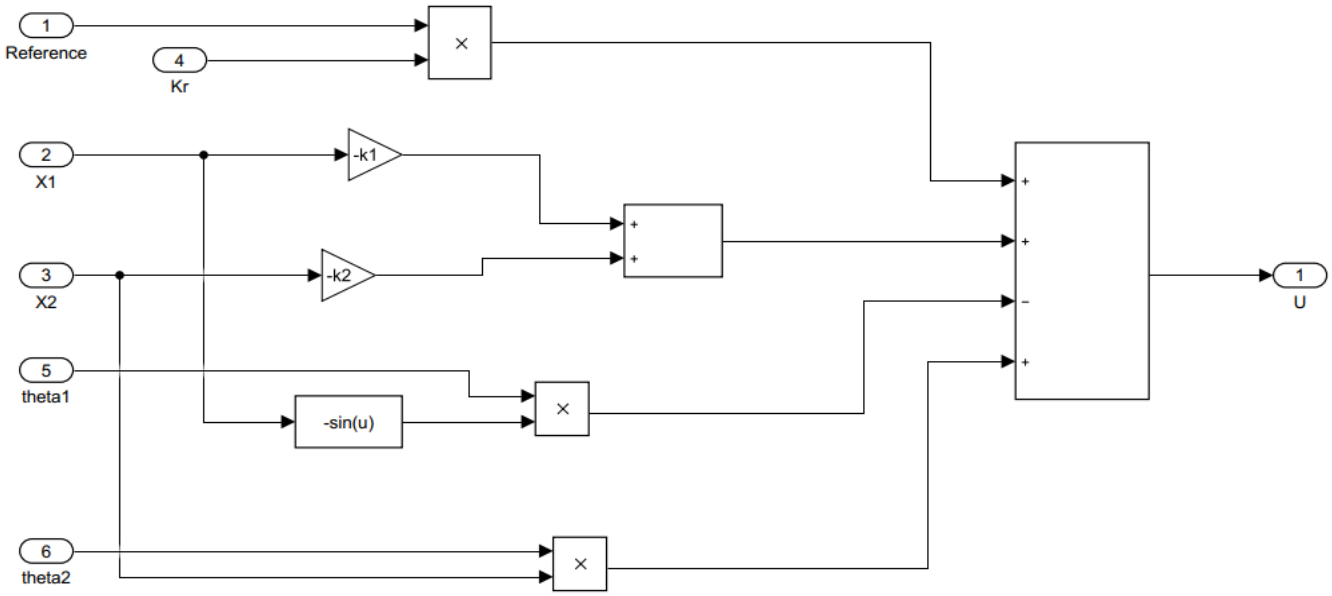
Adaptation law for simple pendulum and detail design



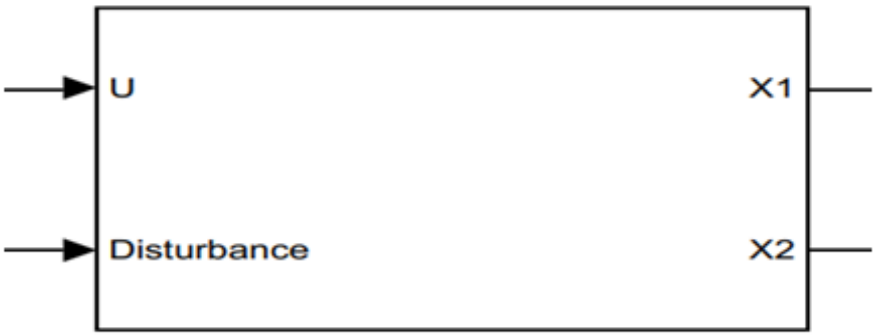


Controller design both nominal and adaptive controller for Pendulum and detail design

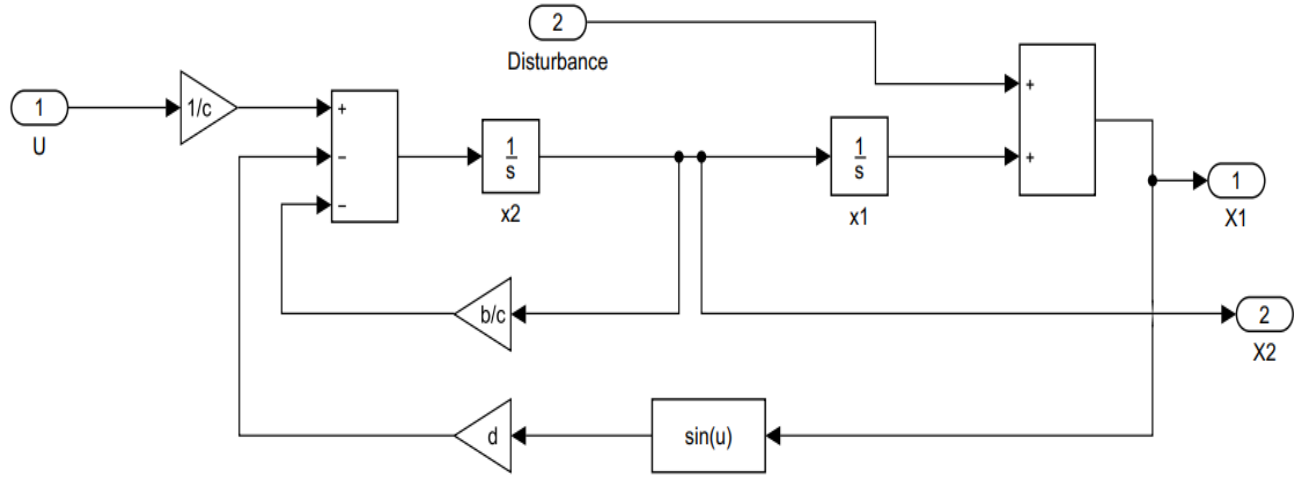




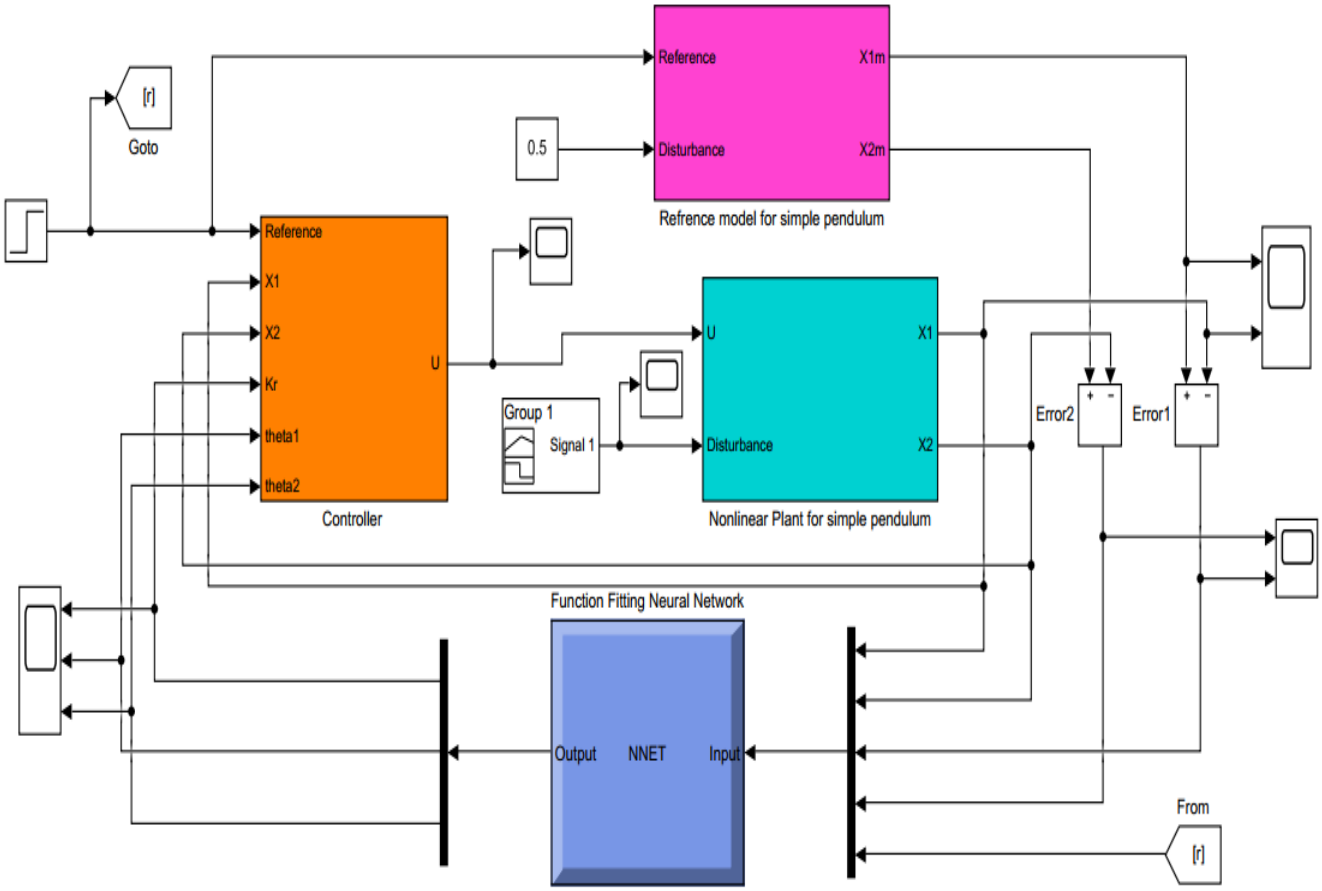
Nonlinear simple pendulum system and detail design.



Nonlinear plant for simple pendulum



The general block diagram for Simple pendulum using NNMRAC



Comparison result for both NNMRAC and CMRAC for Simple Pendulum