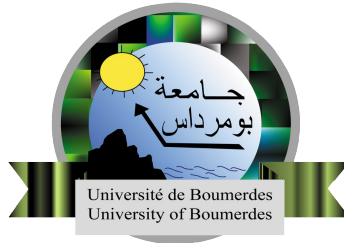


**People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research
University M'Hamed BOUGARA – Boumerdes**



**Institute of Electrical and Electronic Engineering
Department of Power and Control**

*Final Year Project Report Presented in Partial Fulfilment of
the Requirements for the Degree of*

MASTER
In Electrical and Electronic Engineering
Option: Control

Title:

Comparison of Three Neural Network Controllers

Presented by:

CHADLI RAYANE

Supervisor:

Dr. AKROUM MOHAMED

Boumerdes, Algeria

june 2024

List of Figures

1.1	Standards sliding mode controller	3
1.2	Higher order sliding mode controller	4
1.3	Inverted pendulum position control	4
1.4	Nonlinear system (eq1.11) controlled with S.M.C	5
1.5	Non linear system (eq1.12) controlled with S.M.C	5
1.6	Model reference adaptive controller [1]	6
1.7	Non linear system (eq1.15) controlled with MRAC	7
1.8	Non linear system (eq1.24) controlled with linearization feedback	9
1.9	Linear counter-part vs controlled non-linear	9
1.10	Non linear system of polar vehicle controlled with linearization feedback	10
1.11	block diagram of the model reference controlled plant [2]	10
1.12	Model predictive control response	11
1.13	Estimation of the plant with finite past values	12
2.1	Analogy between brain's neuron and ANN [3].	13
2.2	Structure of an artificial neuron network(ANN) [4]	14
2.3	Dynamic ANN	15
2.4	Classical observer	17
2.5	Neural network based Luenberger observer [4]	19
2.6	Estimation using Dynamic systems ANN with one layer	20
2.7	Connection of NEURAL NETWORK observer of a motor	21
2.8	Observer based on R.B.F neural-network [5]	21
2.9	Estimation of Dynamic systems using one layer online ANN	22
2.10	Estimation of Dynamic systems using input-output relationship	23
3.1	NN based MRAC controller block diagram[6]	29
3.2	Pendulum model diagram	31
3.3	Pendulum control	31
3.4	1D car control	32
3.5	NN based NARMA-L2 controller block diagram [5]	35
3.6	Diagram of maglev	36
3.7	Magnetic levitation	36
3.8	CAR SPEED CONTROL with NARMA-L2	37
3.9	Step response of car speed with NARMA	37
3.10	Neural network predictive controller [5]	39
3.11	Continuous steering tank reactor [5]	40
3.12	Continuous steering tank control with NN-MPC	41
3.13	Continuous steering tank control with NN-MPC 1st order ref	41
3.14	Car in 1 dimension system	42
4.1	Liquid level control system [5]	43

4.2	Input flow rate vs tank level of an uncontrolled system	44
4.3	Tank level control with NN controller	45
4.4	High frequency noise on the tank system	46
4.5	High amplitude low frequency noise on the tank system	46
4.6	Error tank level control with NN controller	48
4.7	Tank control signal	49
4.8	Tank output signal in under-trained areas	49
4.9	Tank step response	50
4.10	Motor nonlinear circuit system[7]	51
4.11	Input voltage vs motor speed of an uncontrolled system	52
4.12	Response of the system	53
4.13	High frequency noise on the motor	54
4.14	Low frequency noise on the motor	54
4.15	Motor's speed dynamic error with different NN controller	55
4.16	Speed control signal of a motor	56
4.17	Speed response of a motor	57

List of Tables

3.1	Model Reference Neural Network Control illustration	33
3.2	NARMA-L2 Control illustration	38
4.1	Tank's noise rejection	47
4.2	Error comparison of the tank	47
4.3	Tank's Control Effort	50
4.4	Tank's Performance	50
4.5	Motor's noise rejection	53
4.6	Error comparison of the motor	55
4.7	Motor Control Effort	56
4.8	Motor's Performance	57

Nomenclature

ANN	Artificial Neural Network
ANNC	Artificial Neural Network Controller
AR	Auto-Regressive
ARMA	Auto-Regressive moving average
CSRCHBAC	Cyclic Steered Random Search with Backtracking
LPNN	Linear in-Parameters Neural Network
LQR	Linear Quadratic Controller
LSTMNN	Long Short-Term Memory Neural Network
MA	Moving Average
MIMO	Multiple Inputs Multiple Outputs
MIT	Massachusetts Institute of Technology
MPC	Model predictive controller
MRAC	Model Reference adaptive controllers
NARMA	Nonlinear Auto-Regressive moving average
NARMA L2	Nonlinear Auto-Regressive moving average level 2
NLPNN	Non-Linear in-Parameters Neural Network
NN	Neural Network
NNMPC	Neural Network Model predictive controller
NNMRAC	Neural network Model Reference adaptive controllers
RBF	Radial Base Function
SISO	single Input single Output
SMC	Sliding Mode Controller

List of Algorithms

1	Algorithms for the Model Predictive Control (MPC)	11
2	Weighted Function Algorithm	24
3	NN_estimator(input, output)	25
4	Model Reference Adaptive Control with Neural Network (MRAC-NN)	30
5	NARMA-L2 Neural Network Control Algorithm	35
6	MPC Neural Network Control Algorithm	39
7	CSRCHBAC Algorithm	40

Contents

List of Figures	i
List of Tables	iii
Contents	vi
Abstract	viii
Acknowledgment	ix
Dedication	x
Introduction	xi
1 Nonlinear Controllers	1
1.1 Introduction	1
1.2 Non-Linear Controllers	2
1.2.1 Sliding Mode Controllers (S.M.C)	2
1.2.1.1 Standard (or first-order) sliding mode control	3
1.2.1.2 High-order sliding mode control	3
1.2.2 Adaptive controllers	5
1.2.3 Feedback Linearization	7
1.2.4 Model Predictive Controller (MPC)	10
1.2.5 Artificial Neural Network Controllers (ANNC)	12
2 Introduction to Neural Networks	13
2.1 Introduction	13
2.2 Neural Network (NN)	14
2.2.1 <i>Dynamic</i> neural network:	14
2.2.2 <i>Static</i> neural network:	15
2.2.3 Linear in parameters NN (LPNN)	16
2.2.4 Nonlinear in parameters NN(NLPNN)	16
2.3 Neural Network Estimator	16
2.3.1 Neuro-Observer	17
2.3.1.1 The indirect approach :	18
2.3.1.2 The direct approach	20
2.4 Training and Learning Algorithms	24
2.5 Stability of a Neural Network	26
3 Study of Three Neural Network Controller	28
3.1 Introduction	28
3.2 Model Reference Neural Network Controller	28
3.2.1 Fundamental Concept	28
3.2.2 NN-based MRAC Applications and Illustrations	30
3.3 NARMA L2 Controllers	34

3.3.1	Fundamental Concept	34
3.3.2	NARMA-L2 Applications and Illustrations	36
3.4	Predictive Neural Network Controller	38
3.4.1	Fundamental Concept	38
3.4.2	NN Based MPC Applications and Illustrations	39
4	Simulation and Results	43
4.1	Introduction	43
4.2	Case Study 1 : Liquid Level Control	43
4.2.1	Modeling	43
4.2.2	Simulation and Results	44
4.2.3	Performance Study	45
4.2.3.1	Noise Disturbance Rejection	45
4.2.3.2	Dynamic Error	47
4.2.3.3	Control Effort ΔU	48
4.2.3.4	Response Time & Overshoot	50
4.3	Case Study 2: Motor Nonlinear Control System	51
4.3.1	Modeling	51
4.3.2	Simulation and Results	52
4.3.3	Performance study	52
4.3.3.1	Noise Disturbance Rejection	52
4.3.3.2	Dynamic Error	54
4.3.3.3	Control Effort ΔU	56
4.3.3.4	Response Time & Overshoot	57
4.4	Conclusion	58
	Conclusion	60
	Future Work	60
	Bibliography	62

abstract

The integration of neural networks in control theories has the potential to revolutionize modern control engineering by enhancing performance measures and increasing plant efficiency.

This project aims to explore the performance of three neural network controllers:

Neural Network Predictive controller , Neural Network Model Reference Controller, and the Nonlinear Auto-Regressive Moving Average Controller and evaluate the potential benefits and challenges associated with each type of controller.

A system simulation was created on Simulink and Matlab scripts to compare a set of performance measures. Data were collected from several simulations and averaged to provide estimates.

The findings indicate that neural networks can significantly improve a plant's response and accuracy, especially the Nonlinear Auto-Regressive Moving Average (NARMA-L2) and Neural Network Model Predictive Controllers. However, challenges such as controller training and computational power suggest a need to optimize the network algorithms.

While neural network controllers provided enhanced responses with minimal information about the plants, each type studied has a specific field of application and limitations under certain conditions.

Acknowledgment

I would like to express my deepest gratitude to my academic advisor, Dr. Akroum Mohamed, for his unwavering support, guidance, and mentorship throughout the course of this research project. His invaluable insights, encouragement, and constructive feedback have been instrumental in shaping the direction and scope of this thesis. I am indebted to Dr. Akroum Mohamed, whose expertise and dedication have been instrumental in the successful completion of this research. His invaluable guidance, encouragement, and unwavering support have been a constant source of inspiration and motivation. I would like to acknowledge the institute of electrical & electronic engineering for providing a conducive research environment and access to essential resources and facilities. Their support has been invaluable in facilitating the smooth progress of this research project. I extend my heartfelt appreciation to my parents, brothers and loved ones for their unwavering support, understanding, and encouragement throughout this journey. Their patience, encouragement, and unwavering belief in my abilities have been a constant source of strength and motivation. Finally, I would like to express my profound gratitude to all those who have contributed, directly or indirectly, to the completion of this thesis. Your support, encouragement, and guidance have been instrumental in achieving this milestone, and I am truly grateful for your unwavering support.

Dedication

This thesis is dedicated

To my parents and siblings, whose unwavering love, support, and encouragement have been my greatest strength. Their constant belief in me has been a source of inspiration and motivation.

To my mentors and teachers, whose guidance, wisdom, and patience have been invaluable. Your dedication to my growth and learning has profoundly shaped my journey.

And to all those who have supported me throughout this journey, thank you for your kindness, encouragement, and belief in my potential.

Your support has been a cornerstone in the completion of this work.

To my friends, who have shared in my struggles and triumphs, your companionship has made this journey worthwhile.

To the academic community, whose pursuit of knowledge and truth has inspired me,

your passion for learning has fueled my own.

This work is a testament to the collective effort and unwavering belief of many.

Thank you for being part of my journey.

Introduction

In recent decades, control engineering's field has witnessed a significant developments, striven by the desire to improve the performance and efficiency of several systems. The application of various control strategies, particularly nonlinear controllers, has been proven to be a pivot area of focus, producing revolutionary results that the traditional control methodologies failed to produce, Thus unlocking field in system regulation and optimization.

This project explores the implications of integrating nonlinear control strategies into modern control engineering and more specifically neural network controllers. By focusing in the interplay between theoretical background and simulations, we strive to unravel the potential benefits and challenges of neural network control methodologies, with an illustrations and examples on their application in real-world scenarios.

chapter 1 : is a study covering the theoritical background, real world applications, and assessments of several control system methods. We begin the investigation bygoing through a review of the linear control strategies followed by nonlinear controlled techniques. These complex control strategies such as sliding mode controllers, adaptive controllers, the feedback linearization, model predictive controllers, and the artificial neural network controllers, is made possible by this fundamental knowledge.

chapter 2 : focuses on neural networks, a powerful computational tool that's proving to have immense potential in enhancing system performance and adaptability. By examining the theory of neural networks, their estimation capabilities, and training algorithms, we aim to provide insights into their functionalities.

Chapter 3: we strive to utilize neural networks and the inherent logic in nonlinear control strategies to design and study a neural network-based control methodologies. By a combination of theoretical analyses, simulation studies, and systematical evaluations, we endeavor to shed light on the efficacy, limitations, and future prospects of advanced control techniques in modern control engineering.

Chapter 4 : we strive to uncover the limitation and relative efficiency of each neural network based control strategy. Comparing the results with two different study cases to asses their actual behaviour under different circumstances.

Chapter 1

Nonlinear Controllers

1.1 Introduction

Control engineering is a branch of engineering that deals with the design, analysis and implementation of control systems: Control system is a system created to manipulate the behavior of a plant to satisfy given specifications or to behave in a specific way. This manipulation is applied on a plant using a controller, which is considered a subsystem that directly influence the behaviour of the plant.

These controllers can be classified into two categories :

1. Linear Controllers:

Linear controllers are often used when a plant is accurately modeled by a linear function and acts in a linear fashion with its input.

Most popular linear controllers that can be mentioned are :

- (a) *P.I.D controllers (PROPORTIONAL-INTEGRAL-DERIVATIVE)*
- (b) *LEAD-LAG controllers*
- (c) *STATE FEEDBACK controllers*
- (d) *LINEAR QUADRATIC REGULATOR controllers(L.Q.R)*

2. Non-Linear Controllers

Non-linear controllers are often used when a plant is strongly non-linear, has high complexity and a wide operating range. Some of the most popular non linear controllers are :

- (a) *SLIDING MODE controllers (S.M.C)*
- (b) *ADAPTIVE controllers*
- (c) *LINEARIZATION FEEDBACK controllers*
- (d) *MODEL PREDICTIVE controllers (MPC)*
- (e) *NEURAL NETWORK controllers (ANNC)*

The interest of this chapter lies in non-linear controllers.

1.2 Non-Linear Controllers

Nonlinear controllers manage systems with nonlinear behaviours by applying control policies that are usually not linear functions of the states of the system. They handle complex behaviors functions like saturation, dead zones, and hysteresis. Neural network controllers are based on other non-linear controllers, which is why they will be discussed in details.

1.2.1 Sliding Mode Controllers (S.M.C)

Sliding mode controller is a type of controller that deals with non-linear systems. The popularity of the sliding mode controller is due to its low sensitivity to the plant's parameters and the disturbances.

The idea behind the sliding mode controller is to drive the system into a predefined surface area where it would be maintained.

Let's consider the system :

$$\begin{aligned} \ddot{y} + a_1 \dot{y} + a_2 y &= u \\ J &= \int_0^{t_f} |u| dt \end{aligned} \quad (1.1)$$

With a J performance measure to be minimized as shown in eq.(1.1). By using the pontryagin minimum principal [8] as follows :

$$\begin{aligned} H &= |u| + P(t)^T(\dot{x}) \\ \dot{P}(t) &= -\frac{\partial H}{\partial x} \\ H(u^*, x^*, P(t)^*) &< H(u, x^*, P(t)^*) \end{aligned} \quad (1.2)$$

By expanding on eq.(1.2) and using the system given in eq.(1.1). The control policy is described as follows :

$$U = \begin{cases} U_{max} & P_n(t)^* < 0 \\ -U_{max} & P_n(t)^* > 0 \end{cases} \quad (1.3)$$

In this case, the plane considered is P_n , which minimizes the effort required in the process.

While the equation obtained from using optimal control theory eq.(1.2) is tempting due to its simplicity, there are scenarios in practice when such approach cannot be applied that is mostly due to the fact that the plants are either not modeled accurately or are unknown, which pushed scientist to create another approach that has been popular for the last two decades.

This approach chooses the zero tracking error as its surface. Let's consider the following system:

$$\begin{aligned} \dot{x} &= f(x, t) + g(x, t) \\ y &= h(x, t) \end{aligned} \quad (1.4)$$

And a tracking error as a scalar function [9] that takes the system $\sigma(x) : R^n \rightarrow R$ as follows :

$$\sigma = \sigma(e, \dot{e}, \ddot{e}, \dots, e^k) \quad (1.5)$$

Where $k + 1$ is the relative degree of the system and e is the error. For simplicity's sake and to ensure that the system is completely stable when $\sigma = 0$; A reduced sub-system is chosen to represent eq.(1.5).

$$\sigma = \left(\frac{d}{dt} + p \right)^k e \quad (1.6)$$

With p being an arbitrary positive real number.

There are several approaches based on the sliding mode control approach:

1.2.1.1 Standard (or first-order) sliding mode control

These type of controllers are described by control input policy whose forms are :

$$\begin{cases} U = -U_{max} \text{sign}(\sigma) \\ U = -U_{max} \frac{\sigma}{|\sigma| + \epsilon} & \epsilon \approx 0 \\ U = -U_{max} \frac{e^{\sigma/\epsilon} - e^{-\sigma/\epsilon}}{e^{\sigma/\epsilon} + e^{-\sigma/\epsilon}} & \epsilon \approx 0 \end{cases} \quad (1.7)$$

However, these three forms (eq 1.7) are only valid under certain conditions when uncer-

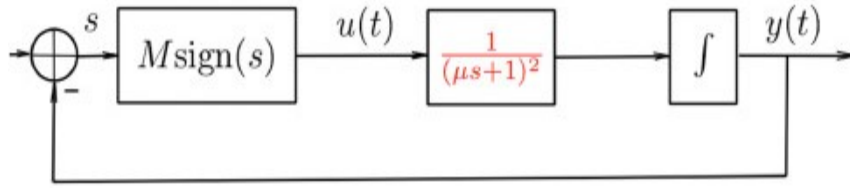


Figure 1.1: Standards sliding mode controller

tainties are not present.

For example ,figure 1.1 illustrates a control block diagram of a first-order system using a sliding mode controller to minimize system effort and error

In case a disturbance is affecting the plant, the higher order SMC are employed.

1.2.1.2 High-order sliding mode control

One popular algorithm for 2^{nd} order sliding mode controller is given [10] by the following control policy :

$$U = -\sqrt{U_{max}|\sigma|} \text{sign}(\sigma) + \int -1.1 U_{max} \text{sign}(\sigma) dt \quad (1.8)$$

The second order sliding mode controller eq.(1.8) is designed to reject all disturbances by choosing suitable parameters through tuning.

The design of higher order sliding mode controller is shown in figure 1.2. This design uses the saturation property of a physical device to control a nonlinear system, which made it raise in popularity even further.

Example: Consider an inverted pendulum project, which involves controlling the force applied to the cart to maintain balance. For the sake of simplicity, the focus will be only on maintaining the pendulum balanced without caring about the distance.

Such system can be described by the non-linear eq.(1.9) :

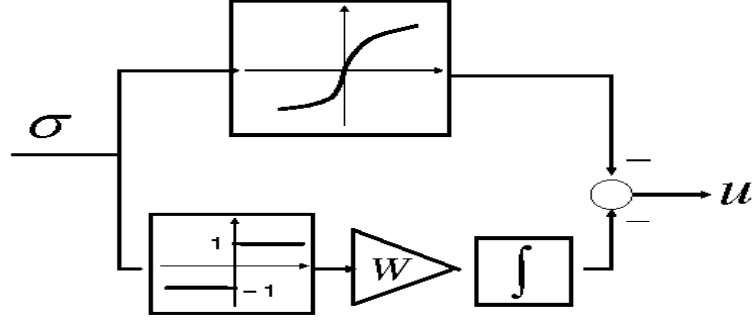
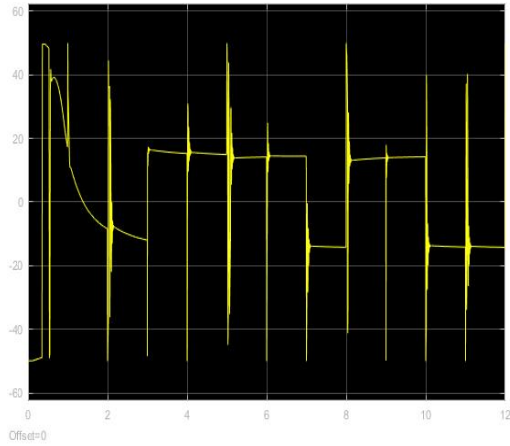


Figure 1.2: Higher order sliding mode controller

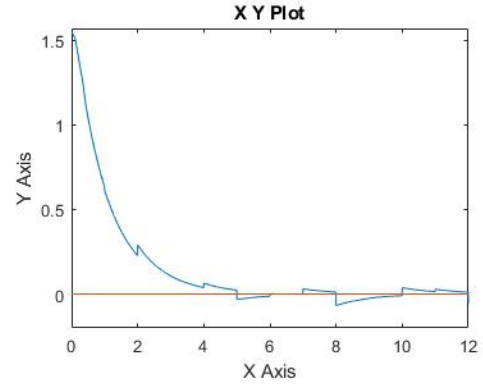
$$\ddot{\theta} = \frac{m \sin \theta (l \dot{\theta}^2 + M_{cart} \cos \theta) + u}{M_{cart} + m \sin^2 \theta} \quad (1.9)$$

In this example $M_{cart} = 10kg$, $m = 0.1kg$ and $l = 0.3m$, thus eq.(1.9) can be expressed as follows:

$$\ddot{\theta} = \frac{0.1 \sin \theta (0.3 \dot{\theta}^2 + 10 \cos \theta) + u}{10_{cart} + 0.1 \sin^2 \theta} \quad (1.10)$$



(a) control signal



(b) inverted pendulum position control with disturbance

Figure 1.3: Inverted pendulum position control

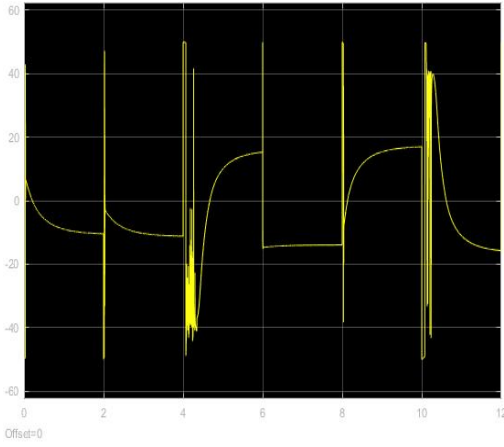
Figure 1.3 shows the control input required to keep the pendulum balanced vertically figure 1.3a and the position angle with a disturbance input to the output 1.3b.

Moreover, the standard sliding mode controller truly shines when controlling either linear or non-disturbed non-linear systems. To show such fact, two more systems described by eq.(1.11) and eq.(1.12) respectively are considered :

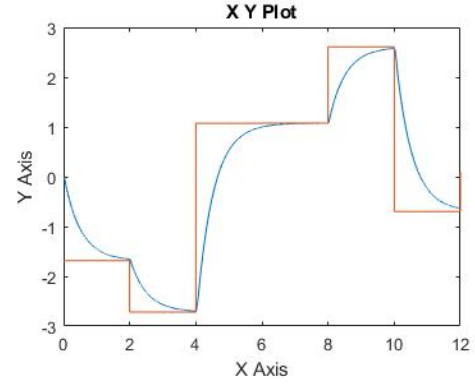
$$\ddot{y} - 2\dot{y} + y = 2u \quad (1.11)$$

$$\ddot{y} + y^3 \dot{y} + y + \dot{y} \cos y - 5u = 0 \quad (1.12)$$

Following the application of the sliding mode controller with the surface defined as the surface for, which the error between the reference and the output to be zero, the following graphs were obtained :

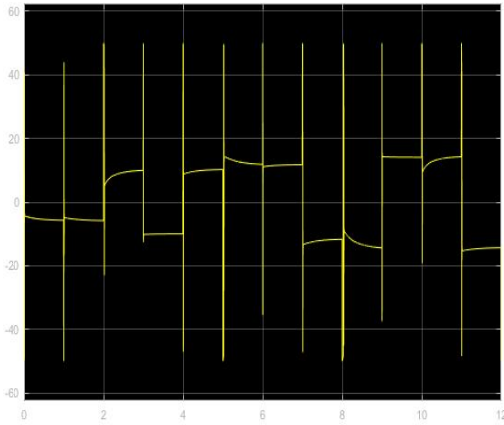


(a) control signal

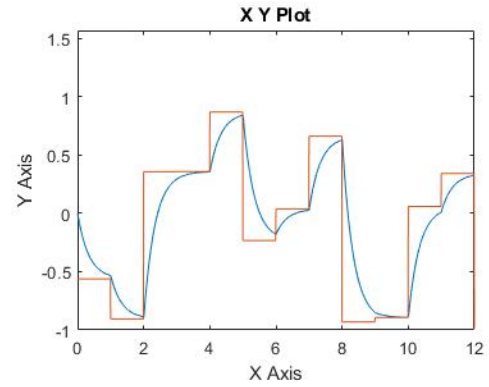


(b) reference vs output linear system

Figure 1.4: Nonlinear system (eq1.11) controlled with S.M.C



(a) control signal



(b) reference vs output non-linear system

Figure 1.5: Non linear system (eq1.12) controlled with S.M.C

1.2.2 Adaptive controllers

There are several adaptive controller that are popular in the industry. In this section, the adaptive controller chosen was the model reference adaptive controller (MRAC), which utilize a reference known model to control a plant.

Figure 1.6 shows the block diagram of a model reference adaptive controller.

Note that the adjustments mechanism depends on the designer. However, the mean square error is often used to create a valid mechanism. Let's consider the error of a system $e = y - y_{model}$ and the mean square error $J = \frac{1}{2}e^2$ [1].

If the MIT algorithm is used to achieve a control policy of the form $U = \theta_1 r$, then the adaptive mechanism would follow this logic

$$\begin{aligned} \frac{d\theta}{dt} &= -\gamma \frac{dJ}{dt} \\ \Rightarrow \frac{d\theta}{dt} &= -\gamma \frac{dJ}{de} \frac{de}{d\theta} \end{aligned} \quad (1.13)$$

Illustration : Figure 1.6 considers that the model's reference transfer function is G_{ref} ,

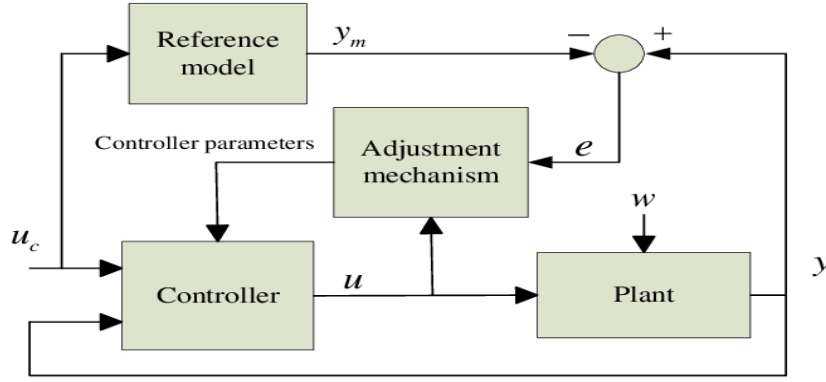


Figure 1.6: Model reference adaptive controller [1]

the plant's transfer function is G , thus eq.(1.13) becomes :

$$\begin{aligned} \frac{d\theta}{dt} &= -\gamma \frac{dJ}{de} \frac{de}{d\theta} \\ &= -\gamma e \frac{de}{d\theta} = -\gamma e y_m \frac{G}{G_{ref}} \end{aligned} \quad (1.14)$$

The parameters of the adaptive control can be chosen arbitrary or through errors and trials, they affect how fast the system converges to the desired reference.

One popular field of engineering that often relies on the adaptive controllers is the aerodynamic system since engineers have a specific behaviors that they would require the system to imitate.

Example:

consider the non-linear system described by :

$$\ddot{y} + y^3 \dot{y} + y + \dot{y} \cos y - 5u = 0 \quad (1.15)$$

and let's take the reference as :

$$G(s) = \frac{3}{s^2 + s + 3} \quad (1.16)$$

By trying to minimize the error between the output of the model system eq.(1.16) and the actual system eq.(1.15), through applying the MIT algorithm. It is possible to obtain an adaptive mechanism with a desired control policy of the form $u = \theta r(t)$, where $r(t)$ is the reference input.

Figure 1.7a represents the control input obtained through designing the adaptive model reference controller, thus forcing the output system to behave as depicted in figure 1.7

The adaptive mechanism used in this example is as follows :

$$\dot{\theta} = -\eta e \frac{-5}{\dot{y} \sin y - 1 - 3y^2 \dot{y}} r(t) \quad (1.17)$$

η represents the learning rate of the adaptive mechanism, which was taken to be equal to $\eta = 0.15$

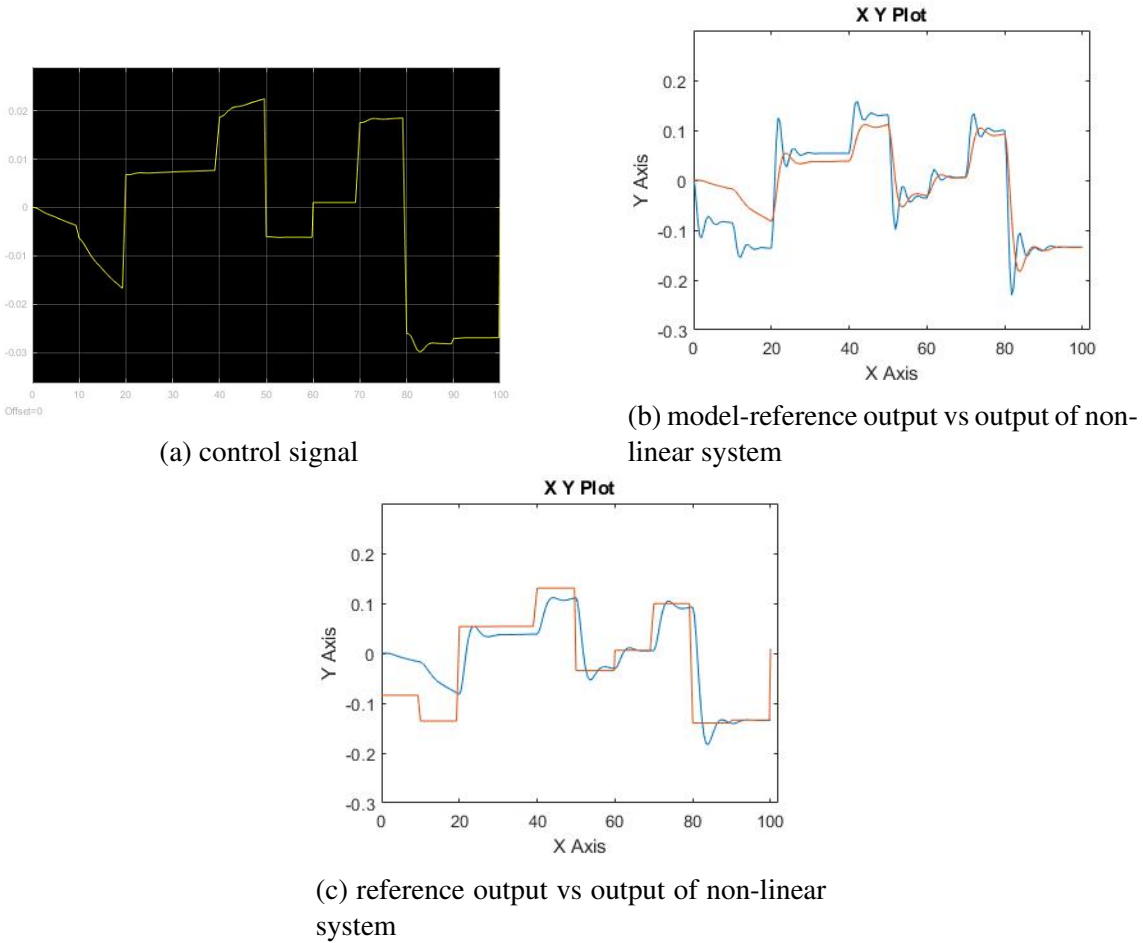


Figure 1.7: Non linear system (eq.1.15) controlled with MRAC

1.2.3 Feedback Linearization

Linear systems are popular due to their simplicity and numerous available approaches in designing and analysing-even though the conditions are quite severe -.Thus scientists and engineers investigated whether or not it was possible to employ the linear system analysis methods to design a non-linear controller.

Relative degree: The relative degree of a system is defined by how many times it is necessary to differentiate the output non-linear function for the effect of the input to appear [11].

Let's consider the following system :

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \quad (1.18)$$

By considering the single-input-single-output eq.(1.18).One can differentiate the output a number of times to find a new state space representation, which is both linear and easy to control using linear controllers.

$$\begin{aligned} y &= h(x) \\ \dot{y} &= \frac{dh(x)}{dx}(f(x) + g(x)u) \\ &\vdots \end{aligned} \quad (1.19)$$

Let's define what is called the lie derivative as follows : $L_f h = (\nabla h)f$

Thus eq.(1.19) can be re-written as :

$$\begin{aligned} y &= h(x) \\ \dot{y} &= L_f h + L_g h u \\ \ddot{y} &= L_f^2 h + L_g(L_f h)u \\ &\vdots \end{aligned} \quad (1.20)$$

One detail to notice when working with the eq.(1.20) is that the input does not influence the first $r-1$ derivative(r is the relative degree), which implies that those derivative do not 'see' the input, henceforth deducing that $L_g(L_f^k h) = 0$ when $k \leq r - 2$. it is possible to Write the eq.(1.20) as :

$$\begin{bmatrix} y \\ \dot{y} \\ \vdots \\ y^{(r)} \end{bmatrix} = \begin{bmatrix} 1 \\ L_f \\ \vdots \\ L_f^r \end{bmatrix} h(x) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ L_g(L_f^{r-1} h(x)) \end{bmatrix} u \quad (1.21)$$

It can be observed that it is possible to alter the states of the eq.(1.21) as :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_{r-1} \\ \dot{x}_r \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 1 & 0 & \dots & 1 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} X + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} V \\ y = [1 \ 0 \ 0 \ \dots 0] X \end{cases} \quad (1.22)$$

With $x_1 = y, x_2 = \dot{y}, \dots, x_r = y^{(r-1)}$ and $V = y^{(r)} = \alpha(x) + \beta(x)u$

The eq.(1.22) is in the controller, which enables us to use a linear controller design approach such as state feedback controller, thus finding a pseudo-control input V , and by extension finding the controller input $u = \frac{-\alpha(x)+V}{\beta(x)}$ [11].

Example: consider the non-linear system described by :

$$\ddot{y} + y^3 \dot{y} + y + \dot{y} \cos y - 5u = 0 \quad (1.23)$$

By expressing the system eq.(1.23) in the state space representation as a result the following description is obtained :

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ 5u - x_2 \cos x_1 - x_1 - x_2 x_1^3 \end{bmatrix} \\ y = x_1 \end{cases} \quad (1.24)$$

Applying the linearization feedback method on the system eq.(1.24), a linear system with a relative degree of 2 is deduced, and is described as follows :

$$\begin{cases} \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} z + \begin{bmatrix} 0 \\ 1 \end{bmatrix} v \\ y = z_1 \end{cases} \quad (1.25)$$

Where $z_1 = y, z_2 = \dot{y}$ and $v = 5u - x_2 \cos x_1 - x_1 - x_2 x_1^3$

It is possible to apply the state feedback to force the system to behave any way desired. In this example, system was made to behave in a similar way to the linear system : $G(s) =$

$$\frac{3}{s^2 + s + 3} \cdot$$

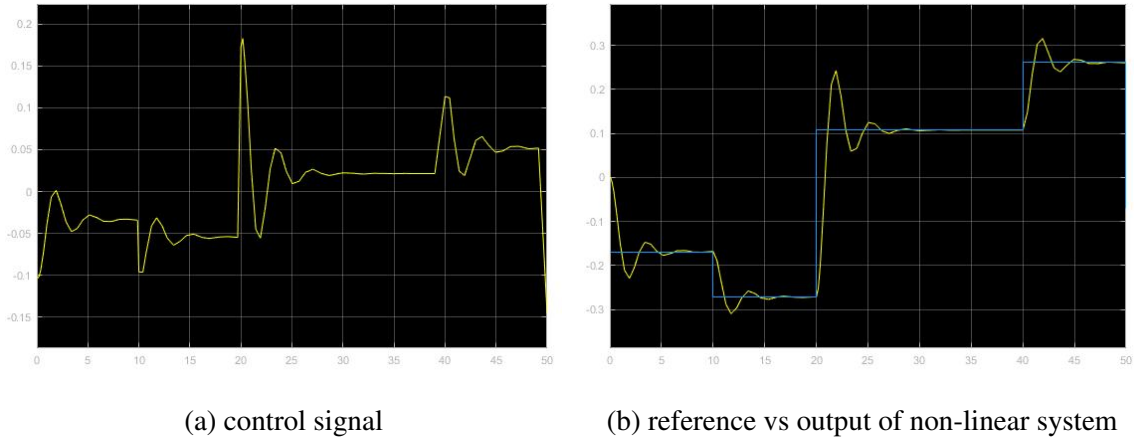


Figure 1.8: Non linear system (eq1.24) controlled with linearization feedback

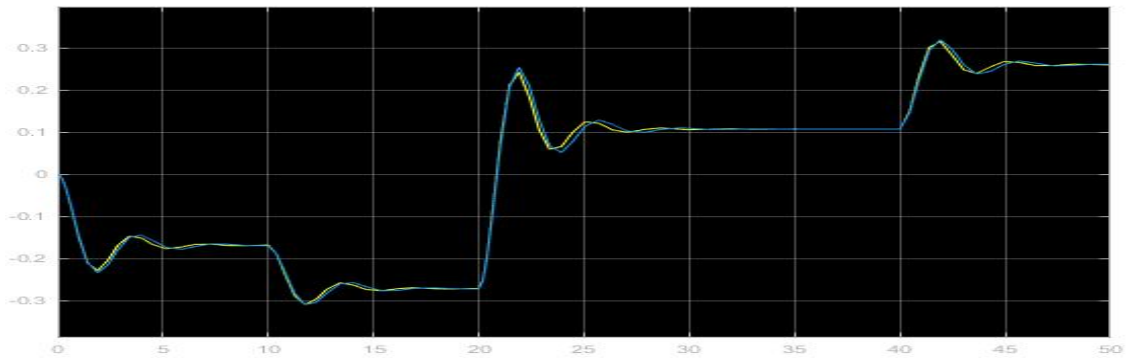


Figure 1.9: Linear counter-part vs controlled non-linear

Figure 1.8a represents the output of the linearization feedback controller, while figure 1.8b shows the desired output(reference) and the actual output

Figure 1.9 represents both the desired output of $G(s) = \frac{3}{s^2+s+3}$ and the actual non-linear output to the controlled plant.

One characteristic that makes linearization feedback controllers popular in the field is its high compatibility with M.I.M.O systems.

As an example; consider the planar vehicle car model described by :

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} u_1 \cos \theta \\ u_1 \sin \theta \\ u_2 \end{bmatrix} \quad (1.26)$$

Manipulating the variables and applying linearization feedback lead to the following results:

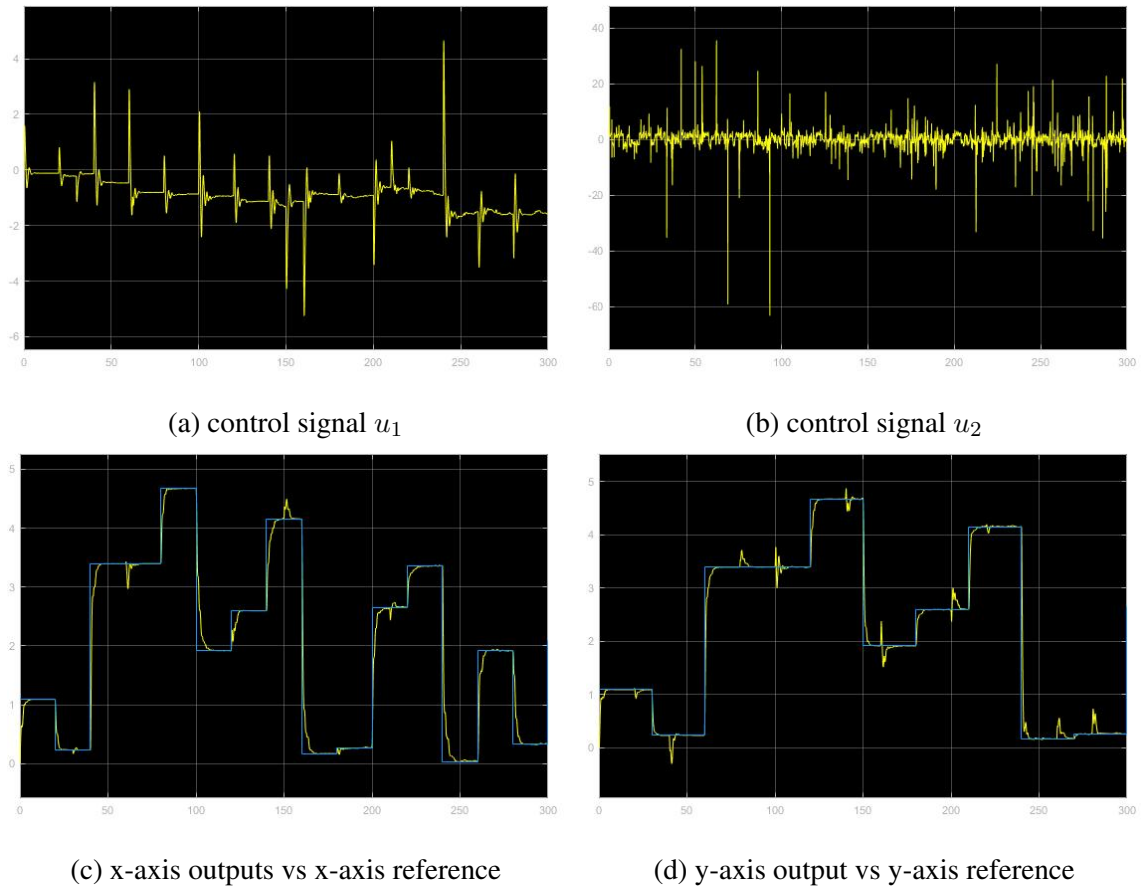


Figure 1.10: Non linear system of polar vehicle controlled with linearization feedback

1.2.4 Model Predictive Controller (MPC)

Model predictive controllers are optimal control based controllers that focus on minimizing a specific performance measure, by relying on the plant's model as well as the provided constraints given to the controller program. These constraints usually describe the limits of the plants such as the physical limits of the actuators as well as the maximum and minimum possible inputs.

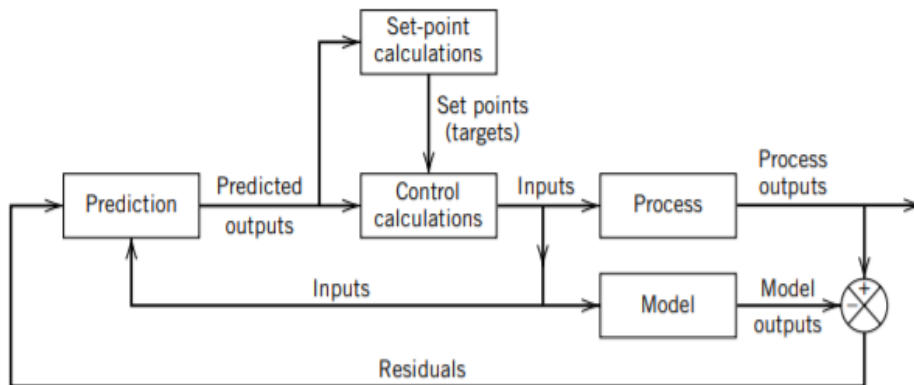


Figure 1.11: block diagram of the model reference controlled plant [2]

figure 1.11 shows a plant controlled using an MPC controller

$$\begin{aligned}
 J = & ||y(T) - y_{ref}||^T H ||y(k) - y_{ref}|| + \\
 & \int_0^T ||y(t) - y_{ref}||^T Q ||y(t) - y_{ref}|| + \\
 & ||u(t) - u_{ref}||^T R ||u(t) - u_{ref}|| \\
 & + ||\Delta u||^T S ||\Delta u||. dt
 \end{aligned} \tag{1.27}$$

The MPC strives to minimize the performance measure described by eq.(1.27). Notice that if a linear system is considered instead of a nonlinear system, the model predictive controller becomes an LQR controller[12] and [13].

One important characteristic and challenging aspect of the model predictive controller is the necessity to design an approximate model that depends on both past and future variable in order to estimate future output of the plant.

Algorithm 1: Algorithms for the Model Predictive Control (MPC)

1. Step 1: Gather Training Data

2. Step 2: Perform least square program

3. Step 3: Use Nonlinear Solver

Use a nonlinear optimization solver such as Hamiltonian approach

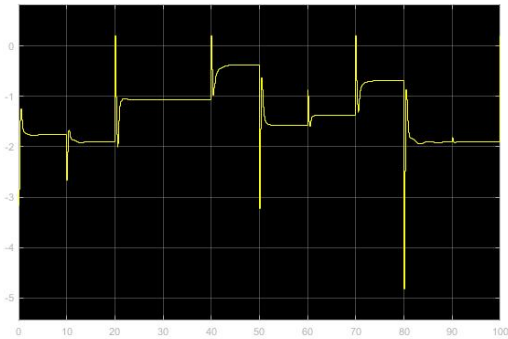
4. Step 4: Apply First Control Input

Apply the first control input from the optimal sequence u_{opt}

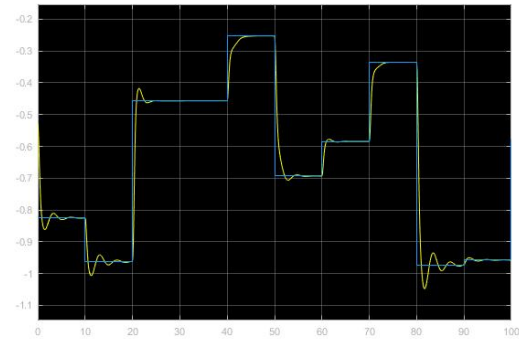
Illustration: consider the following non-linear system [14] :

$$\begin{aligned}
 \dot{x}_1 &= (x_2 - 1) - 2x_1 \\
 \dot{x}_2 &= 2x_2(1 - x_1) - x_1 + u \\
 y &= 1 - e^{1-x_1}
 \end{aligned} \tag{1.28}$$

algorithm 1 describes the different functions necessary to design and implement the



(a) control signal

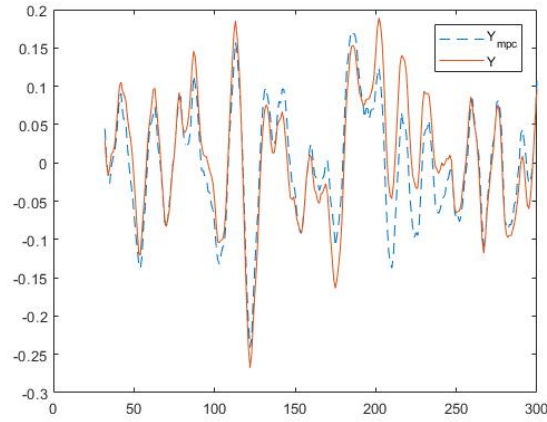


(b) reference vs output of non-linear system

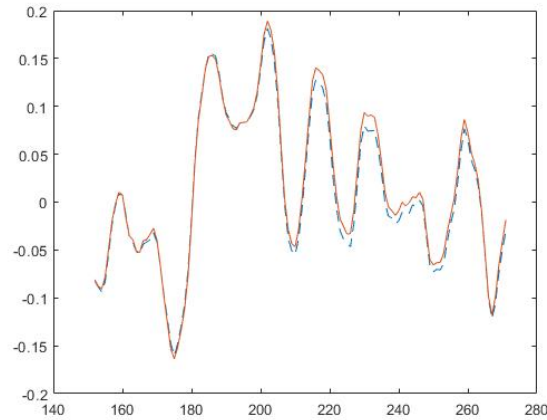
Figure 1.12: Model predictive control response

model predictive control, these steps were employed to design a controller for the system described by the eq.(1.28). The results were as follows:

figure 1.12 represent the control signal as well as the output of the controlled plant in comparison with the reference input considered by the controller. One important aspect of designing the MPC is to design the model as well as a subsystem to predict the future values of the plant.



(a) estimation of the plant with 30 horizon values



(b) estimation of the plant with 150 horizon values

Figure 1.13: Estimation of the plant with finite past values

Figures 1.13 represent the estimation of the output of the plant based on the last few control signals. notice from the graphs that the more data considered the better the estimation is in comparison to the actual system.

1.2.5 Artificial Neural Network Controllers (ANNC)

Neural network controllers can be considered relatively new field of study in the control engineering field. However, it has been gaining popularity, because it doesn't require any knowledge about the plants, which makes it suitable for controlling complex systems with minimum knowledge about their behaviors.

The focus of this report is on the neural network based controllers. Chapter 2 focused on neural network architectures, training methods, and some of their application in control systems, while Chapter 3 covered practical examples and case studies demonstrating the effectiveness of neural network controllers.

Chapter 2

Introduction to Neural Networks

2.1 Introduction

Adapting, learning, and decision making based on experiences and past data are the main characteristics of the brain, which is considered to be the most complex and advanced system in existence.

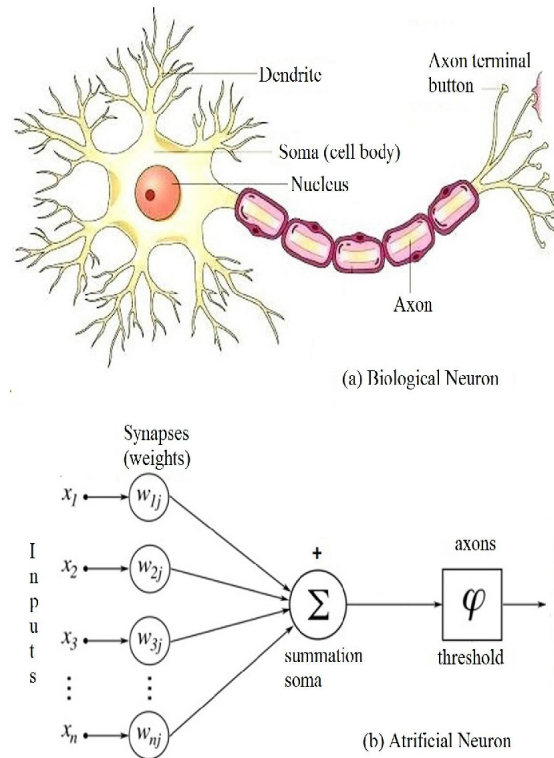


Figure 2.1: Analogy between brain's neuron and ANN [3].

The primary aim of developing neural networks was to emulate human behavior through mathematical modeling. This approach enables the network's output to adapt and learn from past data.

figure 2.1 shows an analogy between a the neuron inside the brain and the mathematical modeling of a single neuron. Thus, it is deduced that the dendrite corresponds to the input in the NN., while the nodes represent the cell nucleus or soma responsible for processing data. The weights play a role in rectifying and adjusting processed data, similar to synapses. Finally, the axon serves as the output in an artificial neural network.

An artificial neural network is organized into layers, with each layer comprising nodes

analogous to neurons, along with weights and biases. The neuron applies an activation function, typically non-linear, to the weighted sum of inputs plus a bias, and then forwards the outputs to the next layer as its input.

The neuron's activation function is generally non-linear and bounded between two values, which implies that the neural network is able to estimate any function or system by training the weights and biases within the hidden layers.

2.2 Neural Network (NN)

A neural network is a model inspired by the structure and function of the human brain. It consists of neurons, which are organized into layers. Each neuron processes the input data, applies an activation function then produces an output. as shown in figure 2.2

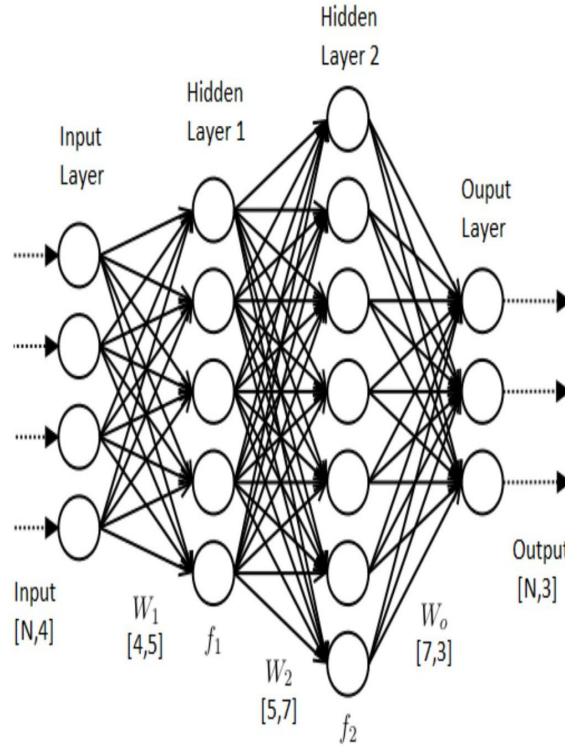


Figure 2.2: Structure of an artificial neuron network(ANN) [4]

Neural network can be divided into several classes. However, The main classes that are studied in control theory are the dynamic and static neural networks.

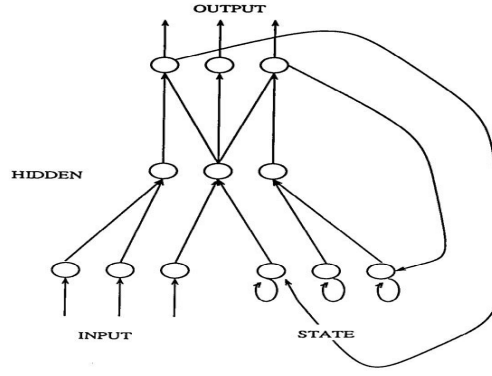
2.2.1 Dynamic neural network:

As the name suggests, the dynamic ANN is designed to process time dependent data; to consider a neural network as dynamic. It is required to have either a memory or a closed loop [15],[16];Examples for dynamic Neural Networks are : recurrent NN figure 2.3a and long-short term NN(2.3b) can be mentioned.

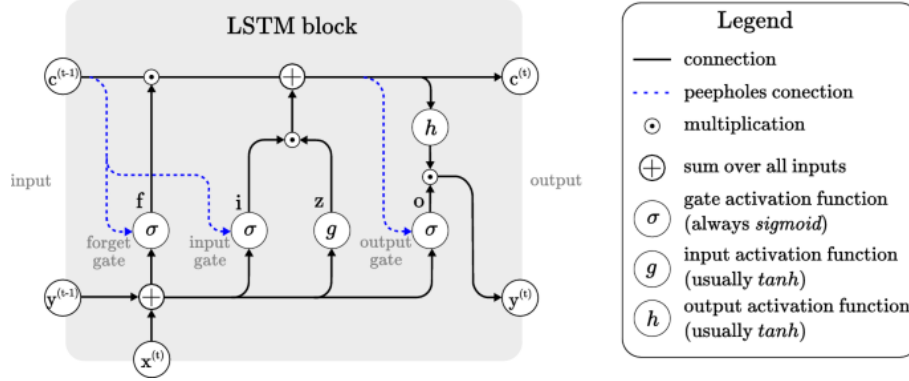
This chapter primarily focuses on a single-layer neural network, which serves as a foundation for theoretical exploration.

Taking into consideration 2.2, it is can be observed that a mathematical modeling of the ANN is as follow:

$$Y = W_0 \sigma(W_1 \sigma(\dots \sigma(W_n X))) \quad (2.1)$$



(a) recursive neural network RNN



(b) long short-term memory neural network L.S.T.M

Figure 2.3: Dynamic ANN

where :

$$W_m = \begin{pmatrix} w_{1,1,m} & w_{1,2,m} & w_{1,3,m} & \dots & w_{1,j,m} & b_{1m} \\ w_{2,1,m} & w_{2,2,m} & w_{2,3,m} & \dots & w_{2,j,m} & b_{2m} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ w_{i,1,m} & w_{i,2,m} & w_{i,3,m} & \dots & w_{i,j,m} & b_{im} \end{pmatrix} \quad (2.2)$$

and

$$X = \bar{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \dots \\ x_{j-1} \\ u \\ 1 \end{pmatrix} = (x^T \quad u \quad 1)^T \quad (2.3)$$

With x being the state and output and u is the input. For simplicity's sake, the only case to consider in the two first chapters is the NN with one hidden layer thus it becomes

$$G(x) = W * \sigma(V * X)) + \epsilon(x) \quad (2.4)$$

In control theories, the interest lies in the dynamic NN, of course there are two types of dynamic NN:

2.2.2 Static neural network:

A prime example of static neural networks is seen in classification and regression tasks where data are independent of time. Static neural networks do not account for time and do not retain memory of past data [15],[16].

2.2.3 Linear in parameters NN (LPNN)

The Linear in-Parameters Neural Network (LPNN) is obtained by fixing the weights of the first layer, e.g. $V = I$ in (2.4) [4]. Hence, the function $G(x)$ can be expressed as

$$G(x) = W\sigma(X) + \epsilon(X) \quad (2.5)$$

2.2.4 Nonlinear in parameters NN(NLPNN)

On the other hand, when the neural network first layer weights are not fixed, i.e., as in equation (2.4) the resulting network is called Nonlinear-in-Parameter Neural Network (NLPNN) [4].

While it is obvious that LPNN is simpler,easier and faster than NLPNN,it cannot represent and is not suitable for a large class of non-linear system

2.3 Neural Network Estimator

mathematical theorem : Let f be a continuous differentiable function, thus it can be approximated by the series.

$$f(x) = f(a) + \frac{df(x)}{dx}\bigg|_{x=a}(x-a) + \cdots + \frac{1}{k!} \frac{d^k f(x)}{dx^k}\bigg|_{x=a}(x-a)^k + \epsilon \quad (2.6)$$

The series (2.6) shows that a function can be approximated using a weighted sum of another function. However, one of its limitation is that the function must be continuously differentiable. That limitation drove mathematicians to look for a way to estimate any function within a small error. That's where neural network estimator came into the picture. Where one would use the error and adjusting weights to obtain a specific result.

Theorem: [17] Let I_n denote the n -dimensional unite cube, i.e., $[0, 1]^n$, and also the space of continuous functions on I_n is represented by $C(I_n)$. By considering a continuous function $\sigma(\cdot)$, which satisfies the following specification

$$\sigma(t) \longrightarrow \begin{cases} 1 & \text{as } t \longrightarrow \infty \\ 0 & \text{as } t \longrightarrow -\infty \end{cases}$$

given any function $f \in C(I_n)$ and $\epsilon > 0$, there exist vectors W_j, V_j and b_j for $j = 1, \dots, N$ and a function $G(x)$ such that

$$|f(x) - G(X)| \leq \epsilon \text{ for all } x \in I_n$$

where

$$G(x) = \sum_{j=1}^N W_j \sigma(V_j x + b_j) = W \sigma(Vx + b) = W \sigma \left((V \ b) \begin{pmatrix} x \\ 1 \end{pmatrix} \right) \quad (2.7)$$

By that definition, it is possible to estimate a large number of states by minimizing the performance measure using the neural network methods The universal approximation theorem stated in shows that a NN with a back-propagation training algorithm has the potential of behaving as a universal approximating algorithm assuming the performance measured an error between two outputs of some kind e.g(mean square error, absolute error ...).

2.3.1 Neuro-Observer

In engineering system; it is imperative to know the states of a plant due to their role in defining the stability and behavior of the overall system. One might even need the internal states to control and stabilize the system.

One popular technique for observer sub-system is known as The Luenberger observer. As a remainder consider :

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX + DU \end{cases} \quad (2.8)$$

and

$$\begin{cases} \dot{\hat{X}} = A\hat{X} + BU + L(Y - \hat{Y}) \\ \hat{Y} = C\hat{X} + DU \end{cases} \quad (2.9)$$

If the performance measure of the observer is defined as follow $E = X - \hat{X}$ then $\dot{E} = \dot{X} - \dot{\hat{X}}$

From eq.(2.9) and eq.(2.8), the following result is obtained :

$$\dot{E} = AX + BU - A\hat{X} + BU + L(Y - \hat{Y}) \quad (2.10)$$

$$\dot{E} = AX + BU - A\hat{X} - BU - L(CX + DU - C\hat{X} - DU) \quad (2.11)$$

By simplifying the eq.(2.11)

$$\begin{aligned} \dot{E} &= AX - LCX - A\hat{X} + LC\hat{X} \\ \dot{E} &= (A - LC)(X - \hat{X}) = (A - LC)E \end{aligned} \quad (2.12)$$

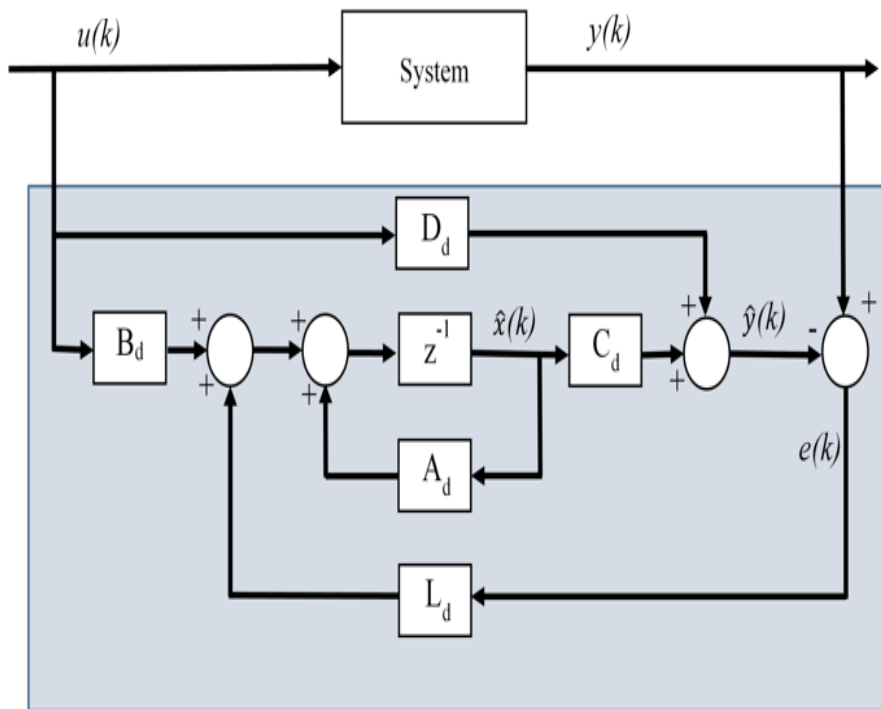


Figure 2.4: Classical observer

Figure 2.4 shows the block diagram implementation of the classical observer. However, this observer is valid only for linear systems or around a fixed-point in a nonlinear system. Moreover, there are cases where the Jacobian of a nonlinear system does not exist or is not observable; on the other hand, linear system might be not observable under the circumstance of the nonexistence of a gain L that enables the error to vanish at some point. *

Unfortunately, most systems have unknown states at all times other than at the initial time thus it would be difficult to choose a suitable control signal.

If only input and outputs of the plant are observable, In this situation, previous outputs may be used as a state vector, which then would allow us to estimate the desired input as well as to observe the internal states at all time.

In numerous pieces of literature : observers are referred to as state estimators. However, it is also true that most of the time and in a number of real-life applications, sensors can only read the output data of the plant, making the states unobservable with traditional methods. This limitation hinders the design of controllers.

Unlike a classical observer: the neural network observer can extract the state data regardless of whether or not the system is linear or nonlinear. In this section, two approaches have been studied

2.3.1.1 The indirect approach :

The indirect approach refers to a combination of a neural network to estimate the unknown nonlinear function of the system dynamics and a classical observer such as Luenberger that was introduced in the section before.

$$\begin{cases} \dot{X} = f(X, u) \\ Y = CX \end{cases} \quad (2.13)$$

Let the system represented by the state space eq.(2.13) describes a plant's behaviour. With $f(X, u)$ being a nonlinear matrix function, and let there be a Hurwitz matrix A such that the pair (A, C) is observable. By adding and subtracting AX from f , the new equation becomes $f(X, u) + AX - AX = g(X, u) + AX$ where g is a nonlinear function, which enable us to analyse the observability of the equation more easily by invoking the Luenberger observer.

$$\begin{cases} \dot{X} = AX + g(X, u) \\ Y = CX \end{cases} \quad (2.14)$$

Now by defining an observer state such that

$$\begin{cases} \dot{\hat{X}} = A\hat{X} + \hat{g}(\hat{X}, u) + L(Y - \hat{Y}) \\ \hat{Y} = C\hat{X} \end{cases} \quad (2.15)$$

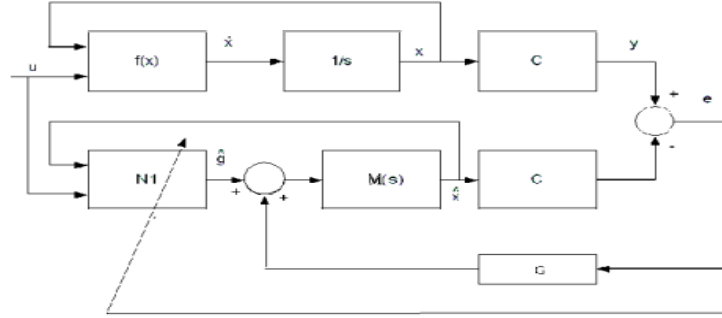
Each of these two equations, eq.(2.14) and eq.(2.15), can be expressed in terms of the neural network.

$$\begin{cases} \dot{X} = AX + W\sigma_K(X, u) + \epsilon(t) \\ Y = CX \end{cases} \quad (2.16)$$

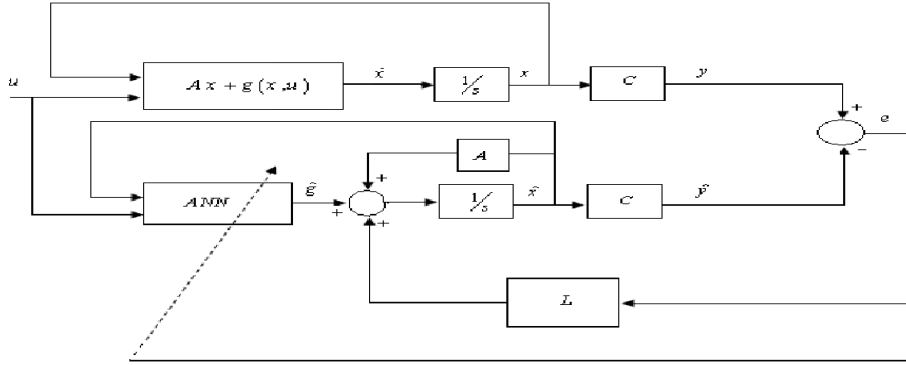
With $\sigma_K = \sigma_{K-1}$ and $\sigma_1 = \sigma(V[X, u]^T)$.

*if the states cannot be observed; the systems are called non-observable, and one can say that the observability matrix is singular

$$\begin{cases} \dot{\hat{X}} = A\hat{X} + \hat{W}\sigma_k(\hat{X}, u) + L(Y - \hat{Y}) \\ \hat{Y} = C\hat{X} \end{cases} \quad (2.17)$$



(a) simplified NN based Luenberger observer



(b) NN based Luenberger observer

Figure 2.5: Neural network based Luenberger observer [4]

Figure 2.5 is a diagram representing the equations(2.17)and(2.16) in a block diagram and their interconnection, where $M(s) = (sI - A)^{-1}$ and $G = L$ [4]

This method was implemented in Matlab using the back-propagation algorithm and a linear in parameter neural network.

Figure 2.6 shows the output of the implemented estimating one layer Artificial neural network*.It has been noticed from the graphs that the while the output was accurately estimated, the internal states were either shifted, amplified or both,which is due to the fact that this kind of observer focus on minimizing the output error and create what is referred as generalized states[†].

This enables the designer to use those states as the plant's actual states since a linear mapping does not affect the poles or the behaviour of the system.

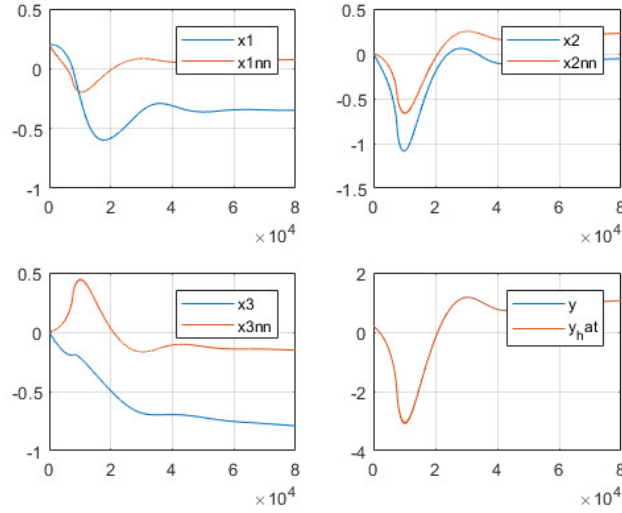
NOTE:

Although figure 2.6 depictss almost perfect identification, it's important to acknowledge that indirect methods carry a high risk of failure, typically because of the difficulty in finding the appropriate Hurwitz[‡] matrix to minimize the feedback gain L.

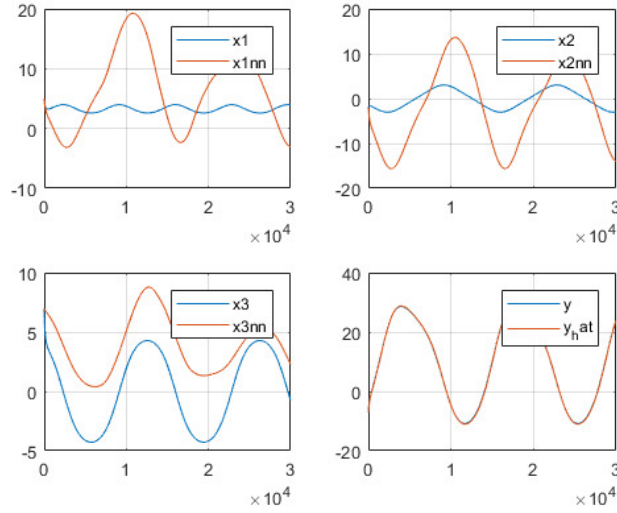
*it is worth mentioning that these system identification and observers implemented using a simple sigmoid function with a single layer LPNN,thus it can only observe up till 3rd order systems.With an increase in complexity there is an increase in the number of layers.

[†]The generalized states are a linear mapping of the actual states

[‡]the hurwitz matrix plays a huge role in minimizing the error and ensuring the stability of an indirect approach type of neural network observer: It defines the singularity of the observation matrix



(a) estimation of a nonlinear system



(b) estimation of a nonlinear limit cycle

Figure 2.6: Estimation using Dynamic systems ANN with one layer

2.3.1.2 The direct approach

To implement the direct approach, it is necessary to know the numbers of layers as well as the activation function of the neural network used for the training. However, unlike the indirect approach, it has almost no limitation other than the hardware limitations such as the saturation of components and maximum voltage and current output. That is mainly due to the fact that the direct approach does not depend on any feedback gain, which ensures that only the characteristics of the neural network affects the estimations.

Taking into consideration the system described by eq.(2.13), where f is a nonlinear function. The observer would approximate the states as follows :

$$\begin{cases} \hat{\dot{X}} = \hat{W}\sigma(V[\hat{X}, U]^T) \\ Y = CX \end{cases} \quad (2.18)$$

Figure 2.7 and figure 2.8 depicts the block diagram of the neural network-based observer, derived from equations eq.(2.13) and eq.(2.18)

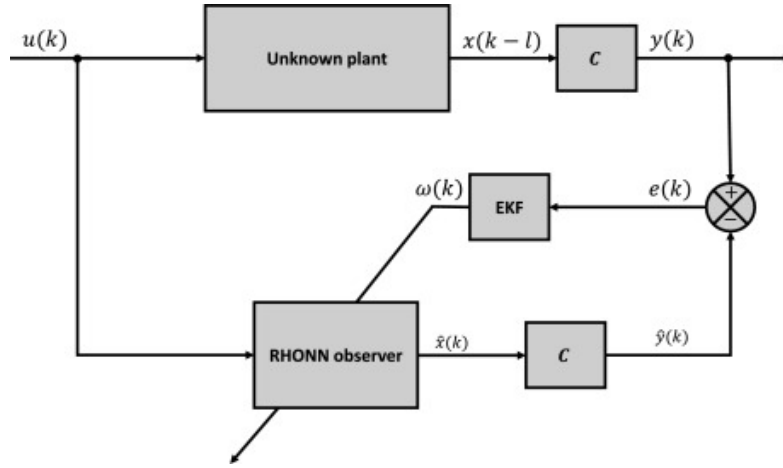


Figure 2.7: Connection of NEURAL NETWORK observer of a motor

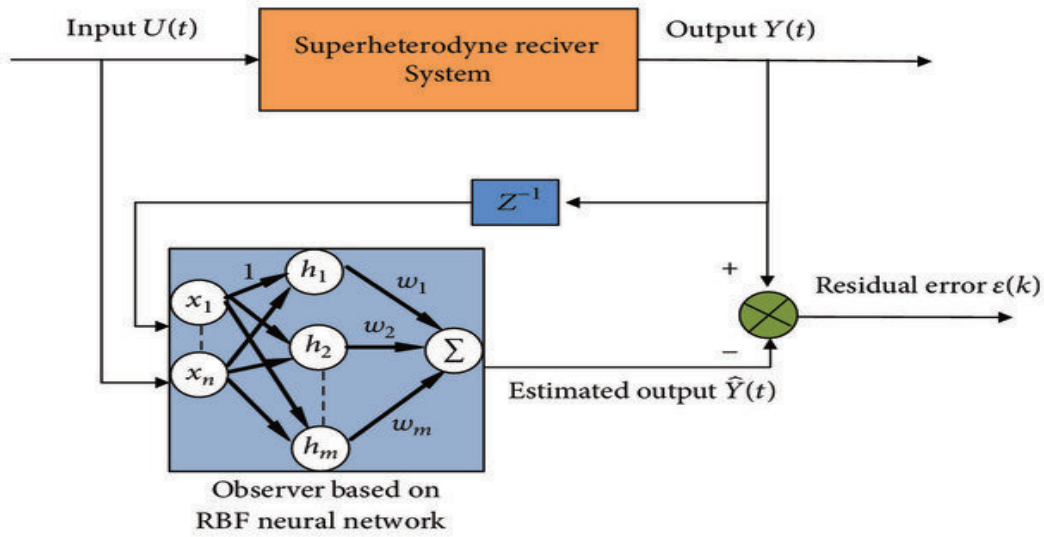


Figure 2.8: Observer based on R.B.F neural-network [5]

Using these approximated states it is possible to design a proper control signal to operate the plant as desired, which would open a window of infinite possibilities.

It's noteworthy that the direct approach to neural modeling of a system may pose greater theoretical complexity. Nevertheless, adjusting our variables offers the potential to considerably reduce the computational complexity of the algorithm.

There are two main methods, which have been popular in system identification and plant's modeling using neural networks:

The first one is the standard approach using the system state space description :

By considering the back-propagation error to be $E = \frac{1}{2}(\dot{y} - \hat{\dot{y}})^T(\dot{y} - \hat{\dot{y}})$. It is possible to express the update equation of the weights as follows:

$$\begin{cases} \dot{W} = (\eta(\dot{y} - \hat{\dot{y}})^T C)^T (\sigma(V^T [\hat{x} \ u]^T))^T \\ \dot{V} = (\eta(\dot{y} - \hat{\dot{y}})^T C)^T [\hat{x} \ u] (\text{diag}(\sigma_d(V^T [\hat{x} \ u]^T))) W^T \end{cases} \quad (2.19)$$

The equations 2.19 can be implemented by using the simplified algorithm 2

The algorithm was tested with the system described by:

$$\ddot{\Phi} = -10 \sin \Phi - 2\dot{\Phi} + u(t) \quad (2.20)$$

Different inputs were tested and the results are shown in figure 2.9

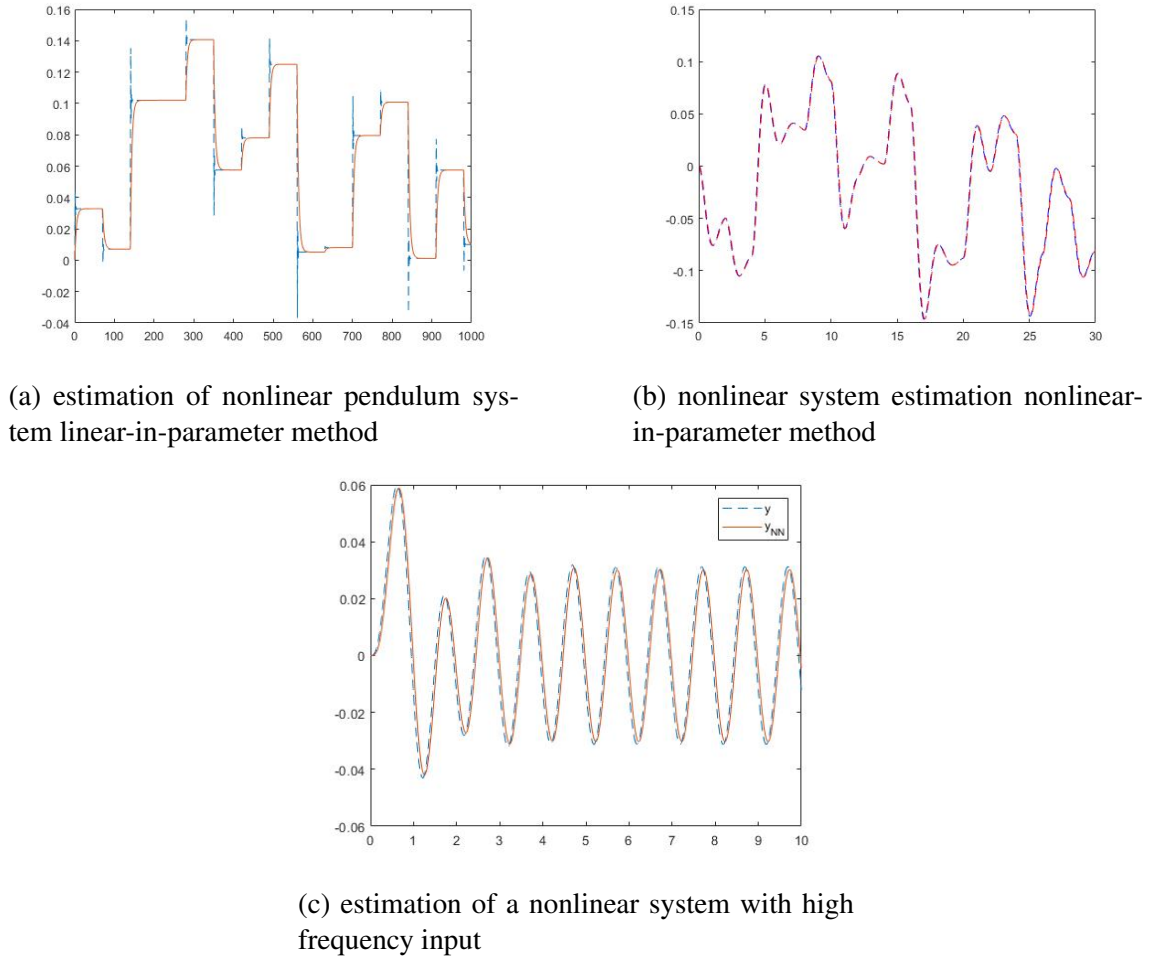


Figure 2.9: Estimation of Dynamic systems using one layer online ANN

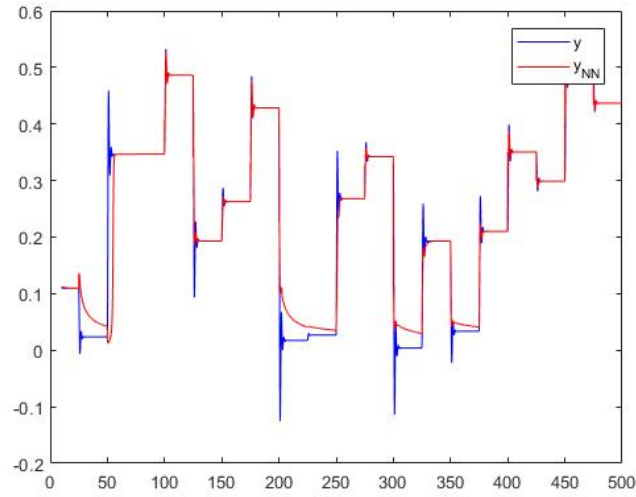
The second popular approach considers modeling a system by using the input-output relation of the system. Let's consider the system described by the eq.(2.20). It is possible to describe the future values of such system as a function of the previous outputs as well as the current and past inputs.

$$y[n+1] = f(y[n], y[n-1], \dots, y[n-n_y], u[n], u[n-1], \dots, u[n-n_u]) \quad (2.21)$$

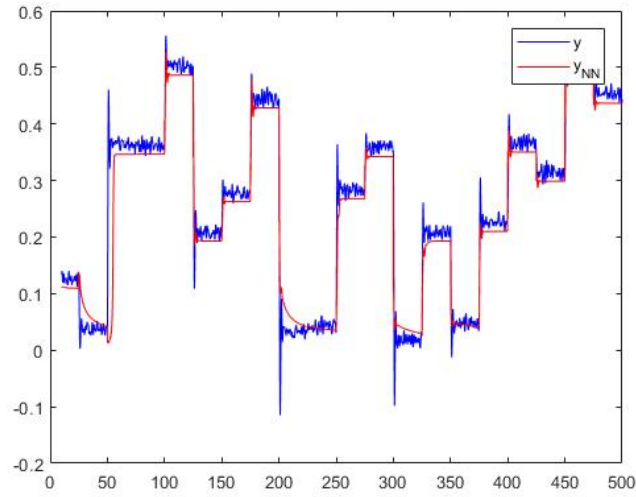
eq.(2.21) describes the behaviour of the system depending on the inputs and outputs over an specific interval of time. As seen in this chapter. Any nonlinear function can be estimated using a dynamic(online) neural network. However in this case an estimator of the form eq.(2.22) is considered :

$$\hat{y}[n+1] = \lambda \sigma(W^T \sigma(V^T \bar{x})) \quad (2.22)$$

This dynamic neural network was implemented by following the algorithm 3. the results can be seen in figure 2.10 which represent the plant model estimation using the algorithm 3



(a) nonlinear system estimation



(b) estimation of system with noise

Figure 2.10: Estimation of Dynamic systems using input-output relationship

NOTE:

While neuro-observers generalize the states, is robust, can handle non-linearities, and is capable of adapting parallel processing. It has some limitations such as its great computational resources' requirement as well as the necessity for a high speed processor. Moreover, the number of layers depends on the complexity of the system.

A prime example of the downside and the requirement for training will be mentioned later in the chapter about neural network model reference adaptive controller in chapter 3

Algorithm 2: Weighted Function Algorithm

Input: input, output
Output: \hat{y}

```

1 Initialize  $W, C, p, V$ , learning_rate, and step_time;
2  $y\_hat \leftarrow output(1)$ ;
3 for  $i \leftarrow 1$  to  $size(output)$  do
4   if  $i > 1$  then
5      $\dot{y} \leftarrow \frac{output(i) - output(i-1)}{step\_time}$ ;
6   end
7   else
8      $\dot{y} \leftarrow \frac{output(1)}{step\_time}$ ;
9   end
10   $x \leftarrow output(i)$ ;
11   $u \leftarrow input(i)$ ;
12   $hidden\_layer \leftarrow activation(V' \times [x, u]')$ ;
13   $x\_hat\_dot \leftarrow W' \times hidden\_layer$ ;
14   $y\_hat\_dot \leftarrow C \times x\_hat\_dot$ ;
15   $p(i, 1) \leftarrow y\_hat + step\_time \times y\_hat\_dot$ ;
16   $E \leftarrow errored(y\_dot, y\_hat\_dot)$ ;
17   $y\_hat \leftarrow p(i, 1)$ ;
18   $V \leftarrow V + step\_time \times (learning\_rate \times E' \times C)' \times [x, u] \times$ 
     $derivative(V' \times [x, u]') \times W'$ ;
19   $W \leftarrow W + step\_time \times (learning\_rate \times E' \times C)' \times (hidden\_layer)'$ ;
20 end

```

2.4 Training and Learning Algorithms

There are four types of training when it comes to machine learning and neural networks[15].

1. supervised learning

This type of training is based on the fact that the algorithm is supplied with a set of data., which consists of inputs and output; these data are then generalized by the algorithm to respond accordingly to the input.

2. unsupervised learning

This type of training has no type of support and is not provided with a target;It focuses on finding similarities between inputs.The input are then categorised based on these similarities

3. reinforcement learning

This type of training is a mixture of both supervised and unsupervised learning, no data is provided to the algorithm however, if the output is wrong, it is told that those data were not the 'correct' ones.

4. evolutionary learning

biological organisms adapt to improve their survival rates.This type of training tries to mimic this biological trait of evolution;however it is outside of the scope of this project report.Which focuses on reinforced training mostly.

Algorithm 3: NN_estimator(input, output)

Input : input, output**Output:** p

```

1  $W \leftarrow \text{zeros}(7, 1);$ 
2  $C \leftarrow [1 \ 0];$ 
3  $p \leftarrow \text{zeros}(\text{size}(\text{output}), 1);$ 
4  $V \leftarrow \text{zeros}(7, 7);$ 
5  $p(1) \leftarrow \text{output}(1);$ 
6  $p(2) \leftarrow \text{output}(2);$ 
7  $p(3) \leftarrow \text{output}(3);$ 
8  $p(4) \leftarrow \text{output}(4);$ 
9  $p(5) \leftarrow \text{output}(5);$ 
10  $\text{learning\_rate} \leftarrow 0.98;$ 
11  $\text{step\_time} \leftarrow 0.1;$ 
12  $\text{lamb} \leftarrow 5;$ 
13 for  $i \leftarrow 5$  to  $\text{size}(\text{output}) - 1$  do
14    $x1 \leftarrow \text{output}(i);$ 
15    $x2 \leftarrow \text{output}(i - 1);$ 
16    $x3 \leftarrow \text{output}(i - 2);$ 
17    $x4 \leftarrow \text{output}(i - 3);$ 
18    $x5 \leftarrow \text{output}(i - 4);$ 
19    $u1 \leftarrow \text{input}(i);$ 
20    $u2 \leftarrow \text{input}(i - 1);$ 
21    $x \leftarrow [x1; x2; x3; x4; x5; u1; u2];$ 
22    $\text{net0} \leftarrow V^\top \cdot x;$ 
23    $\text{net1} \leftarrow \text{activation}(\text{net0});$ 
24    $\text{net} \leftarrow W^\top \cdot \text{net1};$ 
25    $R \leftarrow \text{derivative}(\text{net});$ 
26    $Q \leftarrow \text{derivative}(\text{net0});$ 
27    $p(i + 1) \leftarrow \text{lamb} \cdot \text{activation}(\text{net});$ 
28    $e \leftarrow \text{output}(i + 1) - p(i + 1);$ 
29    $V \leftarrow V + \text{step\_time} \cdot \text{learning\_rate} \cdot e^\top \cdot \text{lamb} \cdot R \cdot Q \cdot W \cdot x^\top;$ 
30    $W \leftarrow W + \text{step\_time} \cdot \text{learning\_rate} \cdot e^\top \cdot \text{lamb} \cdot R \cdot \text{net1};$ 
31 end
32 return p;

```

The neural networks used in control engineering are dynamic, that is to say that they vary with time, however such networks also require training before hand, which is why the neural networks that have been used in the controllers as well as in the observers training algorithms can be classified as reinforced training.

2.5 Stability of a Neural Network

To study the stability of a neural observer, it might be necessary to re-define the definition of a stable observer.

Let us define the stability of the neural observer to be the behaviour of the error between the observed outputs and the actual output and how it converge due to a sudden change in the inputs and states.

In the previous sections, the direct and indirect method were introduced to design an observer. However since the two methods are equivalent, only one case will be studied concerning the stability.

Consider the following descriptions :

$$\begin{cases} \dot{e} = Ae + \tilde{W}\sigma(\bar{x}) + w(t) \\ \dot{\tilde{W}} = \eta(ec^T c)^T \sigma^T(\bar{x}) + \rho \tilde{W} \\ e = X - \hat{X} \\ \tilde{W} = W_{ideal} - \hat{W} \end{cases} \quad (2.23)$$

By considering the equations 2.23. One can check the stability of the neural network by considering the lyapunov candidate $V(X) = \frac{1}{2}X^T P X + \frac{1}{2\rho}tr(\tilde{W}^T \tilde{W})$ Where :

- * e : represents the error between the observed outputs and the actual outputs.
- * \tilde{W} : represents the difference between the ideal weights W_{ideal} and the estimated weights \hat{W}
- * $w(t)$: represents some external disturbance plus the NN error.
- * P : is a positive definite matrix such that $A^T P + P A = -Q$ with Q being a positive definite matrix.
- * ρ : small positive constant to avoid oscillation of the estimated weights.

Of course by considering the constraints on the weights as well as the maximum values defined for the activation functions of the neuron. the stability condition may be developed even further as follows:

$$\begin{aligned} \dot{V}(x) &= -\frac{1}{2}\tilde{x}^T Q \tilde{x} + \tilde{x}^T P (\tilde{W}\sigma(x) + w) + tr(\tilde{W}^T \dot{\tilde{W}}) \\ &\leq -\frac{1}{2}\lambda_{min}(Q)\|\tilde{x}\|^2 + \|\tilde{x}\| \|P\| (\|\tilde{W}\sigma_{max} + \|w\|\| + \sigma_{max}\|\tilde{W}\| \|m\| \|\tilde{x}\| \\ &\quad + (\|W\| \|\tilde{W}\| - \|\tilde{W}\|^2) \|c\| \|\tilde{x}\| \end{aligned} \quad (2.24)$$

With

$$* m = \eta \rho^{-1} A^{-T} c^T c$$

according to lyapunouv condition of stability. it is required for $\dot{V}(x)$ to be negative, to do so, solving the inequality for $||\tilde{W}||$ using the quadratic formula.

$$||\tilde{x}|| \geq \frac{2||P||||w|| + \frac{(\sigma_{max}||P||+||W||||c||+\sigma_{max}||m||)^2}{2}}{\lambda_{min}(Q)} = b \quad (2.25)$$

equation 2.25 shows that the neural network observer provide an estimation within an error of b and is stable in the sense of lyapunouv [4].

such concept may be extended to all dynamic neural networks. That is to say, any neural network with a valid design is stable within a "n-dimensional" ball.

Chapter 3

Study of Three Neural Network Controller

3.1 Introduction

While the non-linear controllers introduced in Chapter 1 showcased remarkable accuracy in terms of results, it is necessary to note that such controllers require knowledge or at least a close estimation of the system behaviors, as well as access to the internal states. On the other hand, using neural networks, which only require training data to function, is more beneficial and flexible.

Throughout this chapter, three distinct neural network controllers have been introduced, exploring their principles, training methodologies, and applications in control engineering.

3.2 Model Reference Neural Network Controller

3.2.1 Fundamental Concept

Neural network based-Model reference controller has been relatively popular due to two reasons, one of, which being its simplicity and the other reason is the intuitive theoretical explanation behind its behavior. This controller requires a reference model similar to the MRAC controller, on the other hand, the way the neural network controller is design differs significantly from the MIT and lyapounov methods.

Chapter 1 was a detailed study of some controllers. Those controlled will be a basis for defining the functionality of the neural network based model reference controllers. Now, let's delve into the process of analyzing such controllers using the linearization feedback analysis method.

$$\begin{cases} \dot{X} = AX + Br \\ y = Cx \end{cases} \quad (3.1)$$

$$\begin{cases} \dot{X} = f(x, u) \\ y = x_1 \end{cases} \quad (3.2)$$

Systems described by eq.(3.1) and eq.(3.2) represent the reference system and the plant model respectively. To simplify the analysis of such a plant, one might use the lie-differentiation used in the linearization feedback analysis method seen in Chapter(1) as follows :

$$\begin{cases} \dot{Z} = \begin{bmatrix} 0 & 1 & \dots & 0 & 0 \\ \vdots & 0 & 1 & \vdots & 0 \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix} Z + \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix} v \\ y = z_1 \end{cases} \quad (3.3)$$

Where $z_1 = y, z_2 = \dot{y}, \dots, z_r = y^{(r-1)}$ and $v = y^{(r)} = \alpha(x) + \beta(x)u$.

By that definition, one can write the control signal u as $u = \frac{v + K r_{ref} - \alpha(x)}{\beta(x)}$; As seen in chapter(2). Any equation can be approximated using an artificial neural network as long as there are enough data to train the network. and the control signal $u = \frac{v + K r_{ref} - \alpha(x)}{\beta(x)}$ is no exception to the rule.

The desired output of the neural network is an estimate of the control signal obtained from using the feedback linearization or at least an equivalent function.

The question would be how to approximate the control signal using a neural network?

A popular approach relies on using the back-propagation method with a slight parameters manipulation.

1. the first step is to use the estimator seen in chapter(2) to estimate the plant with a small degree of error.
2. the second step is to create another neural network such as :

$$\hat{u} = \hat{W} \sigma_k \left(\begin{bmatrix} x \\ u \\ r \end{bmatrix} \right) \quad (3.4)$$

Where r is the reference signal. Eq.(3.4) represent the ANN to estimate a control signal in such a way that the plant model behave in a similar way to the reference model.

3. the third step is to define the error as $e = y_{ref} - \hat{y}$ and $J = \frac{1}{2} e^T e$. This particular modification enables us to train the network using the training algorithm :

$$\hat{W} = -\eta \frac{\partial J}{\partial \hat{W}} = -\eta \frac{\partial J}{\partial e} \frac{\partial e}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \hat{u}} \frac{\partial \hat{u}}{\partial \hat{W}} \quad (3.5)$$

Eq.(3.5) is currently the most efficient approach in designing the control signal using back-propagation.

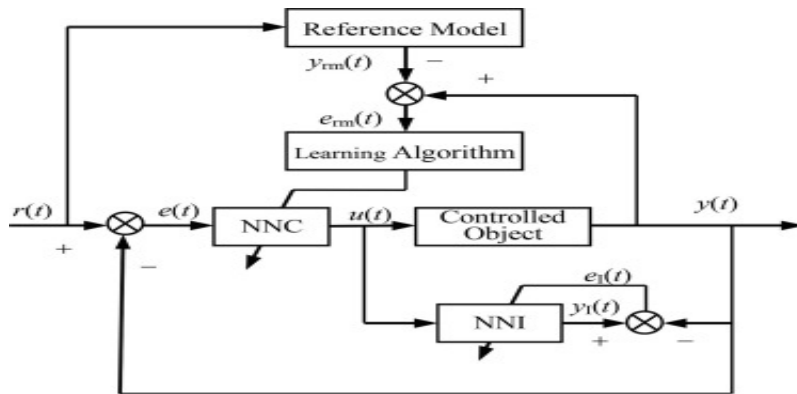


Figure 3.1: NN based MRAC controller block diagram[6]

The algorithm is as follows :

Algorithm 4: Model Reference Adaptive Control with Neural Network (MRAC-NN)

Inputs: Plant estimator from Chapter 1;

Neural network architecture;

Training data: state variables (x), control signal (u), reference signal (r), reference output (y_{ref});

Parameters: Learning rate (η);

Number of training iterations;

Output: Trained neural network weights (\hat{W});

```

1 while not converged do
2   Estimate the plant using the estimator from Chapter 1;
3   Initialize neural network weights randomly;
4   for  $i \leftarrow 1$  to Number of training iterations do
5     Compute the estimated control signal:
6        $\hat{u} = \hat{W} * \text{activation\_function}([x, u, r]);$ 
7     Compute the output of the plant using the estimated control signal:
8        $y_{\text{hat}} = \text{Plant\_estimator}([x, \hat{u}]);$ 
9     Calculate the error between the reference output and the estimated output:
10       $e = y_{\text{ref}} - y_{\text{hat}};$ 
11     Calculate the cost function:  $J = \frac{1}{2}e^T e;$ 
12     Compute the gradients of the cost function with respect to the neural
13       network weights:  $dJ_{\hat{W}} = -\eta \cdot \frac{\partial J}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{u}} \cdot \frac{\partial \hat{u}}{\partial \hat{W}};$ 
14     Update the neural network weights using gradient descent:
15        $\hat{W} = \hat{W} + dJ_{\hat{W}};$ 
16   end
17 end
18 Output the trained neural network weights ( $\hat{W}$ );

```

3.2.2 NN-based MRAC Applications and Illustrations

Let's consider three different illustrations :

1. **illustration 1:** the first illustration is provided by matlab

$$\ddot{\theta} = -10 \sin \theta - 2\dot{\theta} + \tau \quad (3.6)$$

Illustration depicted in eq.(3.6) is shown in figure 3.2

The goal is to control the arm is such way that the arm has exactly an angle equal to that of the reference.

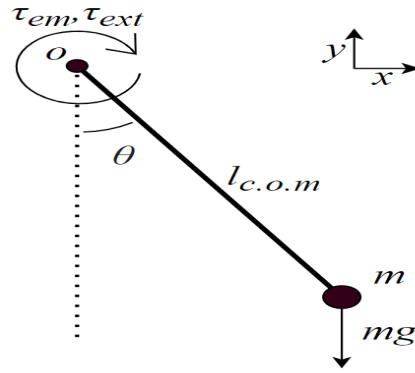
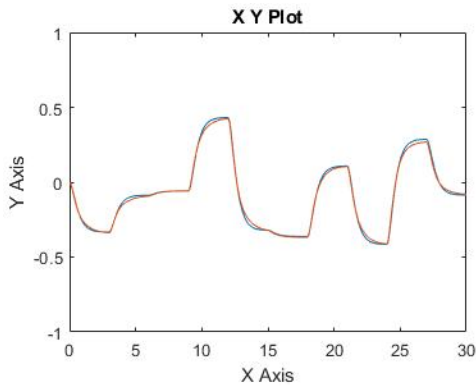
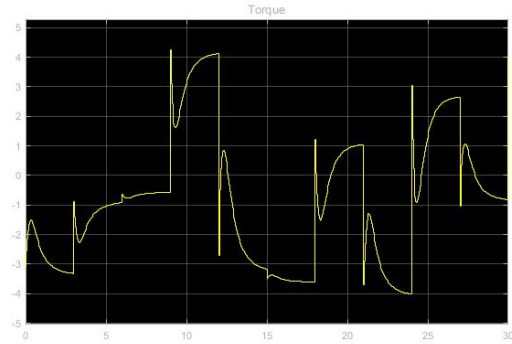


Figure 3.2: Pendulum model diagram



(a) reference model vs output of the controlled system



(b) control signal input to the pendulum

Figure 3.3: Pendulum control

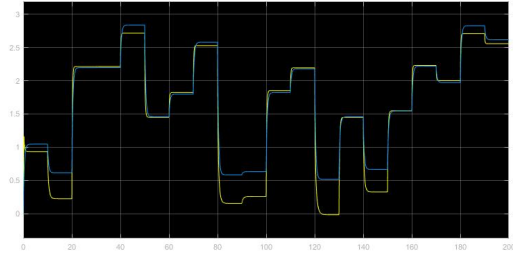
2. Illustration 2:

$$\ddot{x} = 20u - \dot{x}^2 \text{sign}(\dot{x}) - 0.5 \quad (3.7)$$

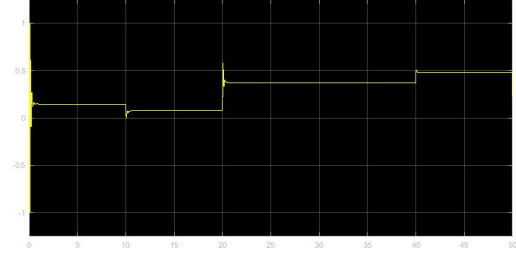
Expression eq.(3.7) describes the behaviour of a small sized car in one dimension under an input force.

Neural network controllers depends highly on the data set used during the training, figure(3.4a) shows how the plant behaves if a reference signal that was not part of the training is

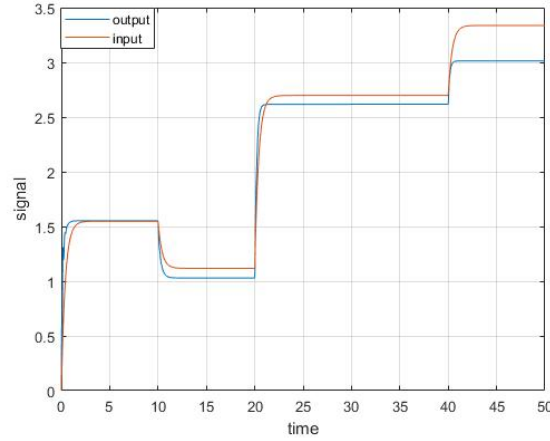
Neural network controllers are significantly influenced by the dataset utilized during their training process. figure 3.4a provides insight into the behavior of the plant when confronted with a reference signal that was not included in the training dataset. While figure(3.4c) represent the plant behavior for when the reference signal was part of the training.



(a) system response when the controller is untrained for input less than 1



(b) control signal input to 1D car



(c) ideal output vs output speed signal of the 1D car

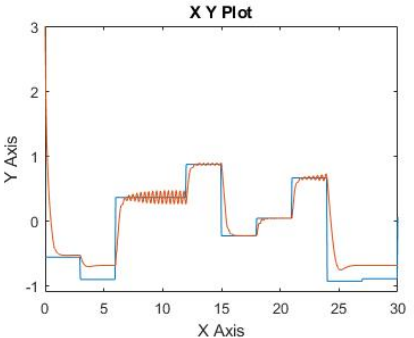
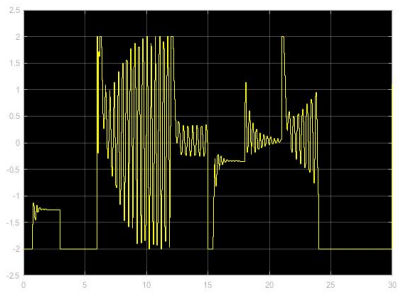
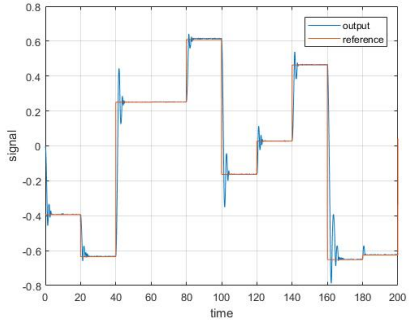
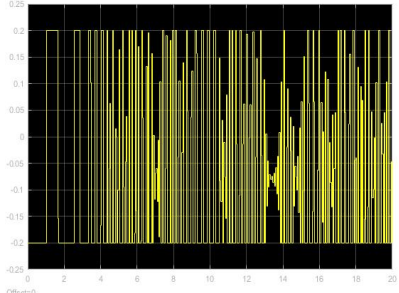
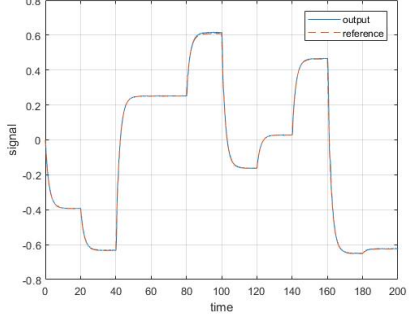
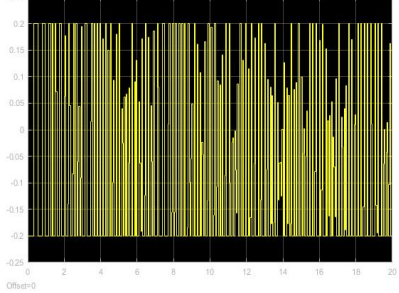
Figure 3.4: 1D car control

3. Illustration 3:

$$\begin{cases} \dot{x}_1 = x_1(3 - x_1 - 2x_2) + u \\ \dot{x}_2 = x_2(2 - x_1 - x_2) - u \\ y = x_1 \end{cases} \quad (3.8)$$

Since this is the last illustration of the model reference neural network control; three different references are considered to assess how well the neural network controller can handle different types of systems commonly encountered in control engineering applications : the first reference is over-damped second order response, second reference has been chosen as a first order system and the third reference was an oscillatory second order system.

Table 3.1: Model Reference Neural Network Control illustration

System Reference Order	Reference vs Plant Output	Control Signal
Oscillatory Second Order		
under-damped Second Order		
first order system		

Note:

One important method used in NN-based MRAC is the cumulative error approach. This method uses the concept of tracking error to then apply a mixture method linearization feedback and linear state feedback control method.

$$r(k) = \lambda_{n-1}(x_1(k) - \bar{x}_1(k)) + \lambda_{n-1}(x_2(k) - \bar{x}_2(k)) + \dots + (x_{n-1}(k) - \bar{x}_{n-1}(k)) \quad (3.9)$$

The eq.(3.9) is applied to the system eq.(3.2) and eq.(3.1). By forcing the tracking error $r(k+1) = K_r r(k)$, one can obtain a control policy of the form :

$$u(k) = Ax(\bar{k}) + Br_{ref}(k) - f(x(k), u(k)) - \Lambda E(k) + K_r r(k) \quad (3.10)$$

Where $E(k) = \begin{bmatrix} e_{n-1} \\ e_{n-2} \\ \vdots \\ e_2 \end{bmatrix}$ and $\Lambda = \begin{bmatrix} \lambda_1 & \lambda_2 & \dots & \lambda_{n-1} \end{bmatrix}$

While this method is popular due to its accuracy, the computational power required is several folds larger than the approach discussed in this section., which is why it is not suitable to use it in more complex and MIMO systems.

3.3 NARMA L2 Controllers

3.3.1 Fundamental Concept

Nonlinear auto-regressive moving average is a more general and wider type estimated model than the traditional auto-regressive moving average linear model. The principle idea is to create a neural network that would estimate the plant. let's consider a Nonlinear auto-regressive moving average estimation to the plant.

$$y[k+1] = N(y[k], y[k-1], \dots, u[k], u[k-1], u[k-2], \dots) \quad (3.11)$$

Eq.(3.11) is any nonlinear function that may approximate the behavior of the system and it is assume to exist. applying taylor series mentioned in chapter (2).eq.(3.11) can be re-written as :

$$\begin{aligned} y[k+1] &= N(y[k], \dots, 0, 0, 0, \dots) + \sum_{i=0}^p \frac{\partial N}{\partial u[k-i]} \Big|_{u=0} u[k-i] \\ &= f(y[k], \dots) + \sum_{i=0}^p g_i(y[k], \dots) u[k-i] \end{aligned} \quad (3.12)$$

eq.(3.12) has been named NARMA Level 1 or NARMA L1 for short, issue with the first level of the NARMA is that there are as many neural networks as there are input delays, this would require a relatively large computational power to design a controller and thus is not suitable for application. Henceforth the second level of the NARMA was invented and is design in a similar way to the first level. let's consider the effect of the control input on the output of the system only :

$$\begin{aligned} y[k+1] &= N(y[k], \dots, 0, u[k-1], u[k-2], \dots) + \frac{\partial N}{\partial u[k]} \Big|_{u[k]=0} u[k] \\ &= f(y[k], \dots, u[k-1], \dots) + g(y[k], \dots, u[k-1], \dots) u[k] \end{aligned} \quad (3.13)$$

eq.(3.13) has been named NARMA Level 2 or NARMA L2 for short, which only requires two Neural networks in order to estimate a nonlinear function. However what is desired is to controlled the plant, to do so $y[k+1] = y_r$ is considered as the desired response, which implies that $u[k]$ is the required control signal for the system to behave as desired.

$$u[k] = \frac{y_r - f(y[k], \dots, u[k-1], \dots)}{g(y[k], \dots, u[k-1], \dots)} \quad (3.14)$$

eq.(3.14) represent the control policy of the input signal.

figure(3.5) represents the block diagram and control signal adjustment of NARMA L2 neural network controller.

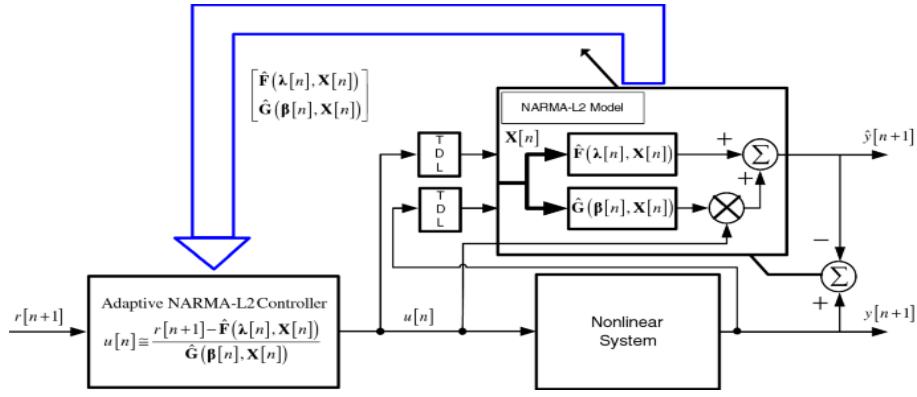


Figure 3.5: NN based NARMA-L2 controller block diagram [5]

the algorithm is as follow :

Algorithm 5: NARMA-L2 Neural Network Control Algorithm

Inputs: Plant estimator from Chapter 1; Neural network architecture; Training data: state variables (x), control signal (u)

Parameters: Learning rate (η); Number of training iterations

Output: Trained neural network weights

```

1 while not converged do
2   Estimate the plant using the estimator from Chapter 1;
3   Initialize neural network weights randomly;
4   for  $i \leftarrow 1$  to Number of training iterations do
5     Compute the estimated control signal:
6        $\hat{y} = \hat{W}_f * \text{activation\_function}([x, u]) + \hat{W}_g * \text{activation\_function}([x, u])u$ ;
7     Calculate the error between the reference output and the estimated output:
8        $e = y_{\text{ref}} - \hat{y}$ ;
9     Calculate the cost function:  $J = \frac{1}{2}e^T e$ ;
10    Compute the gradients of the cost function with respect to the neural
11      network weights:  $d\hat{W}_f = -\eta \cdot \frac{\partial J}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{W}_f}$ ;
12       $d\hat{W}_g = -\eta \cdot \frac{\partial J}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{W}_g}$ ;
13    Update the neural network weights using gradient descent:
14       $\hat{W}_f = \hat{W}_f + d\hat{W}_f$ ;
15       $\hat{W}_g = \hat{W}_g + d\hat{W}_g$ ;
16  end
17 end
  
```

3.3.2 NARMA-L2 Applications and Illustrations

Let's consider three different illustrations :

1. **illustration 1:** this illustration is provided by matlab :

$$\ddot{y} = -9.81 + \frac{15i^2(t)}{y} \text{sign}(i) - \frac{12\dot{y}}{3} \quad (3.15)$$

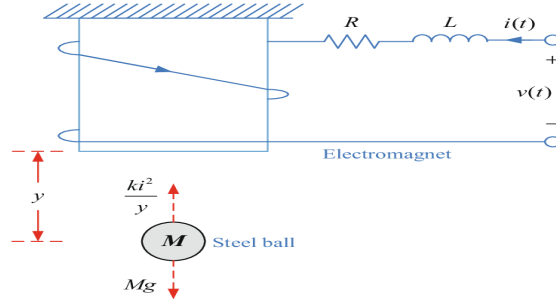
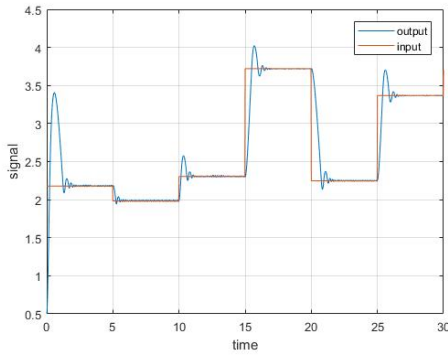
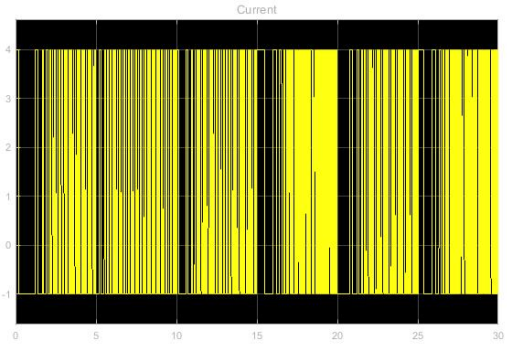


Figure 3.6: Diagram of maglev



(a) magnetic levitation controlled



(b) control signal input to the magnetic levitation

Figure 3.7: Magnetic levitation

2. **Illustration 2:** Expression eq.(3.16) describes the behaviour of a small sized car in one dimension under an input force.

$$\ddot{x} = 20u - \dot{x}^2 \text{sign}(\dot{x}) - 0.5 \quad (3.16)$$

NARMA-L2 controllers depends highly on the variation that the inputs have on the plant's output as well as the data set's elements included during the training process. Thus by extension the size of the data set. figure 3.8a provides an insight to the behaviour of the plant when the data set training elements are focused on the extremum of the plant's output.

Considering figure 3.8a and zooming around the first period of time before the reference changes, figure 3.9 is obtained. It is possible to define figure 3.9 as representing the response of the NARMA-L2 controller to a step input signal.

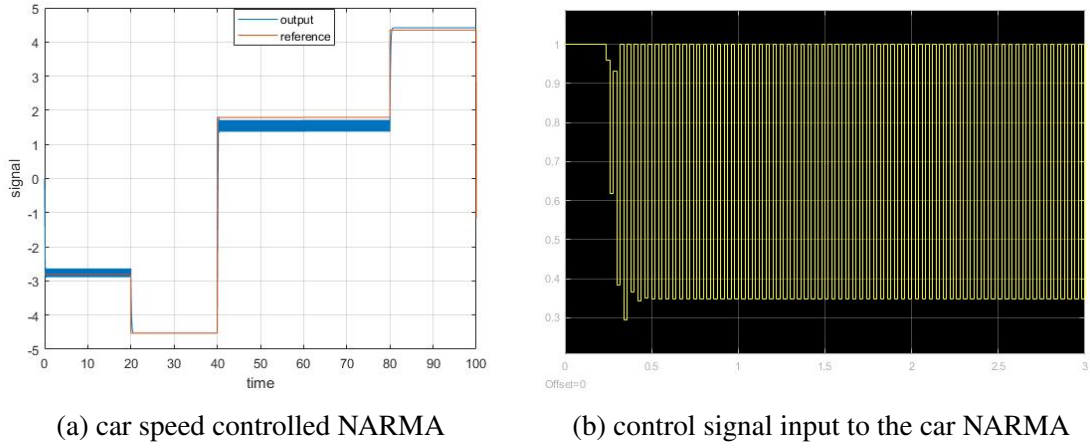


Figure 3.8: CAR SPEED CONTROL with NARMA-L2

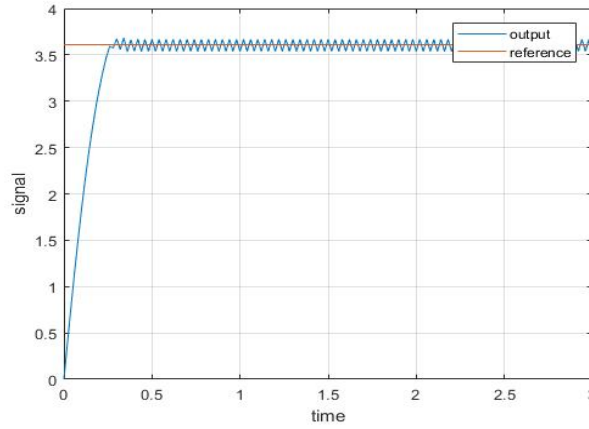


Figure 3.9: Step response of car speed with NARMA

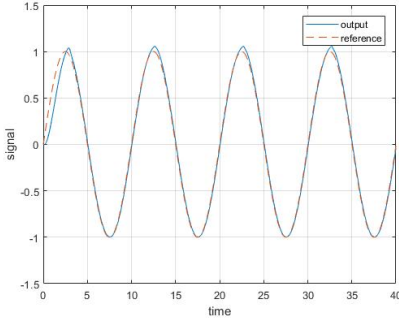
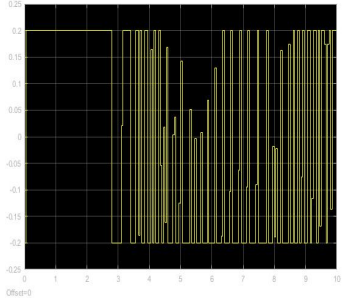
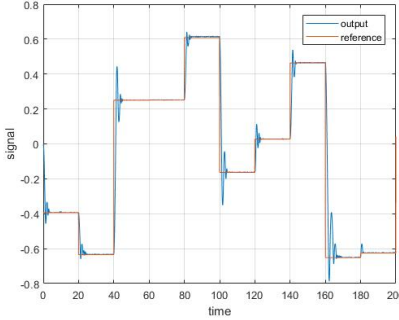
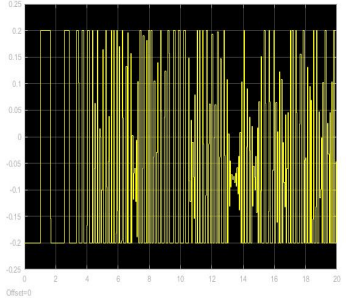
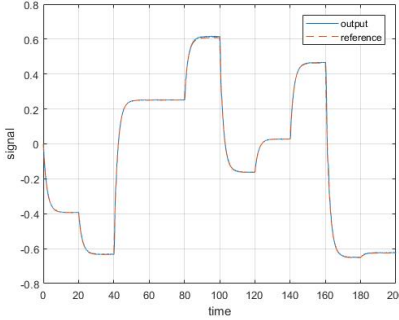
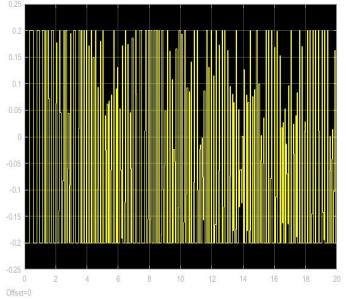
3. **Illustration 3:** eq3.17 is commonly found in mechanical engineering, robotics and mechatronics.

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = 5u - x_2 \cos x_1 - x_1 - x_1^3 x_2 \end{cases} \quad (3.17)$$

Since this is the last illustration of the NARMA-L2 neural network controller; three different references are considered to assess how well the neural network controller can handle different types of systems commonly encountered in control engineering applications : the references are over-damped second order response, first order system ,and an pure oscillatory second order system. The results of the simulation are summarized in table3.2

One important aspect to notice from the results of table 3.2 is that the NARMA-L2 controller has a significant trade-off between the raise time and the overshoot, Moreover the rising time is maximized when the reference undergoes a *jump*. Similarly to the MRAC neural network seen in the section 3.2, these limitations depends on the hardware used and more importantly on the characteristics of the neural network built.

Table 3.2: NARMA-L2 Control illustration

System Reference Order	Reference vs Plant Output	Control Signal
pure Oscillatory Second Order		
under-damped Second Order		
first order system		

3.4 Predictive Neural Network Controller

3.4.1 Fundamental Concept

Neural network predictive controller is optimal control based controller that uses the characteristic of the neural network to estimate the output and predict the optimal control that minimizes a given performance measure, which is why the neural network predictive controller is referred to as neural network based model predictive controller.

Chapter 1 investigated the model predictive controllers theory. However, in this section, the systems are considered to be in discrete-time.

The first step of any neural network based controller is the plant estimation that was developed in chapter 2. Which is then followed by designing the performance measure as follows :

$$J(k) = \sum_{i=1}^{N_2} (\hat{y}(k+i) - y_{ref}(k+i))^2 + \rho \sum_{j=1}^{N_u} (u(k+j-2) + u(k+j-1))^2 \quad (3.18)$$

Eq.(3.18) is designed in MATLAB by varying the constants N_u , N_2 and ρ , which represent the control horizon, cost horizon and the control weighting factor (penalty) as seen

in chapter(1).Moreover, the minimization algorithm as well as its characteristics can be varied depending of the desired behaviour of the system as well as personal preferences.

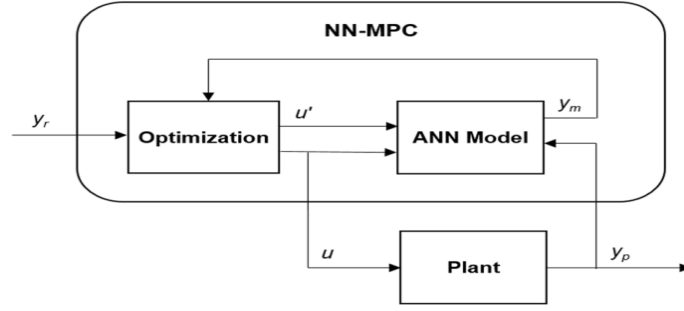


Figure 3.10: Neural network predictive controller [5]

Figure(3.10) represent a diagram of the neural network predictive control.

One of the major drawbacks of the NNMPC is the computational power required at each stage. The neural network has a complexity relative to the number of layers and nodes at each layer as mentioned in chapter(2), on top of that, at each stage it is necessary for the optimization block shown in figure(3.10) to execute a complex algorithm that finds the optimal control signal to minimize the performance measure $J(k)$.

the algorithms are as follow :

Algorithm 6: MPC Neural Network Control Algorithm

Inputs: Plant estimator from Chapter 1; Neural network architecture; Training data: state variables (x), control signal (u)

Parameters: Learning rate (η); Number of training iterations; Cost horizon; Control horizon; Effort weight

Output: Trained neural network weights

```

1 while not converged do
2   Estimate the plant using the estimator from Chapter 1;
3   Initialize neural network weights randomly;
4   for  $i \leftarrow 1$  to Number of training iterations do
5     Compute the estimated control signal:
6        $\hat{y} = \hat{W}_f * \text{activation\_function}([x, u]);$ 
7     Calculate the error between the reference output and the estimated output:
8        $e = y_{\text{ref}} - \hat{y};$ 
9     Calculate the cost function:  $J = \frac{1}{2} e^T e;$ 
10    Compute the gradients of the cost function with respect to the neural
11    network weights:  $d\hat{W} = -\eta \cdot \frac{\partial J}{\partial e} \cdot \frac{\partial e}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \hat{W}};$ 
12    Update the neural network weights using gradient descent:
13       $\hat{W} = \hat{W} + d\hat{W};$ 
14  end
15 end

```

3.4.2 NN Based MPC Applications and Illustrations

For the illustrations of the MPC, two different scenarios are considered due required computational power:

Algorithm 7: CSRCHBAC Algorithm

Inputs: Initial solution \mathbf{x}_0 , Step size α , Backtracking factor β , Maximum iterations N_{\max} , Tolerance ϵ

Output: Optimal solution \mathbf{x}^*

```

1 Initialize solution  $\mathbf{x} \leftarrow \mathbf{x}_0$ ;
2 for  $n \leftarrow 1$  to  $N_{\max}$  do
3   Generate random search direction  $\mathbf{d}$ ;
4   Steer search direction  $\mathbf{d}$ ;
5   Compute trial solution  $\mathbf{x}_{\text{trial}} = \mathbf{x} + \alpha \cdot \mathbf{d}$ ;
6   Evaluate trial solution's performance  $f_{\text{trial}}$ ;
7   while  $f_{\text{trial}} > f(\mathbf{x})$  do
8     Backtrack:  $\alpha \leftarrow \beta \cdot \alpha$ ;
9     Compute new trial solution  $\mathbf{x}_{\text{trial}} = \mathbf{x} + \alpha \cdot \mathbf{d}$ ;
10    Re-evaluate trial solution's performance  $f_{\text{trial}}$ ;
11  end
12  Update solution:  $\mathbf{x} \leftarrow \mathbf{x}_{\text{trial}}$ ;
13  if  $\|\nabla f(\mathbf{x})\| < \epsilon$  then
14    break;
15  end
16 end
Output: Optimal solution  $\mathbf{x}^*$ 

```

1. **illustration 1** eq.(3.19) represents the state space representation of a continuous steering tank reactor that was provided by MATLAB.

$$\begin{cases} \dot{h}(t) = w_1(t) + w_2(t) - 0.2\sqrt{h(t)} \\ \dot{C}_b(t) = (C_{b1} - C_b(t))\frac{w_1(t)}{h(t)} + (C_{b2} - C_b(t))\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{(1 + k_2 C_b(t))^2} \end{cases} \quad (3.19)$$

where $w_2 = 0.1, C_{b1} = 24.9, C_{b2} = 0.1, K_1 = 1$, and $K_2 = 1$

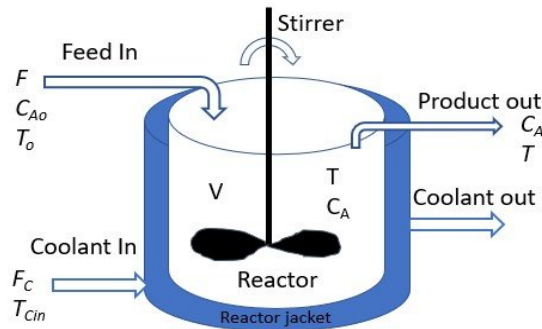
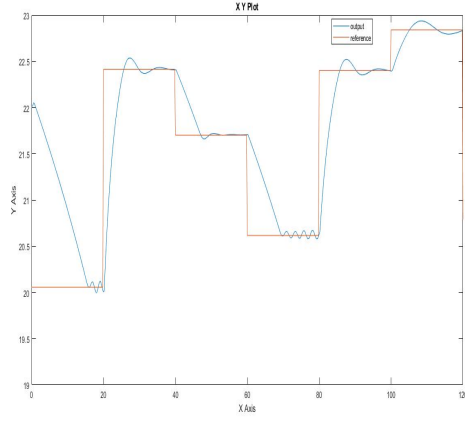
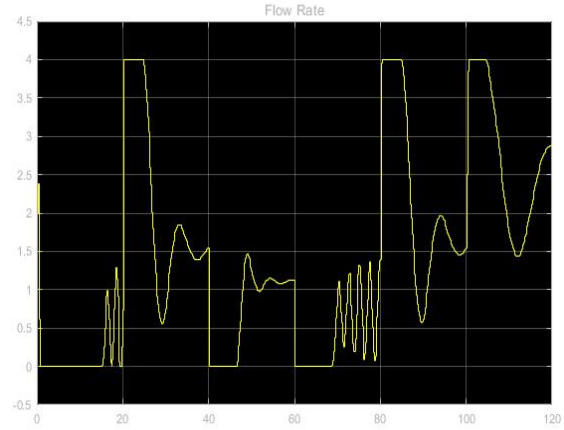


Figure 3.11: Continuous steering tank reactor [5]

Figure 3.12a depicts the response of neural network based model predictive control of the Continuous Steering Tank Reactor with a control signal figure 3.12b while figure 3.13a represents the response of the Continuous Steering Tank Reactor when the reference has a first order behaviour with the control signal depicted in figure 3.13b.

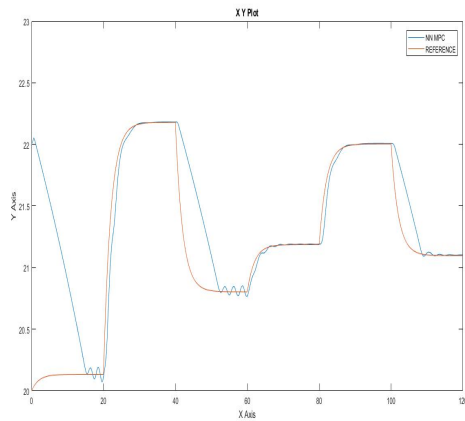


(a) concentration of the steering tank output vs reference

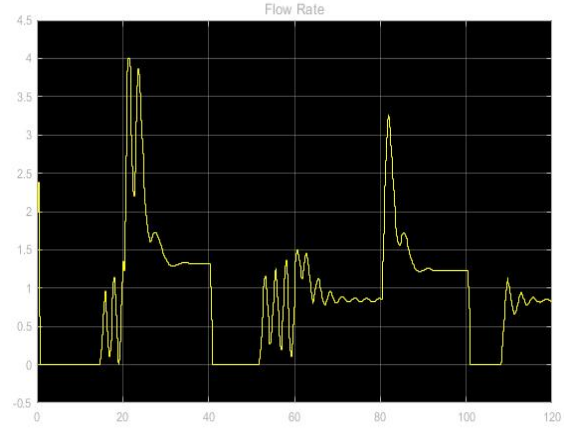


(b) control signal input to the steering tank

Figure 3.12: Continuous steering tank control with NN-MPC



(a) concentration of the steering tank output vs reference with a 1st order reference



(b) control signal input to the steering tank with a first order reference

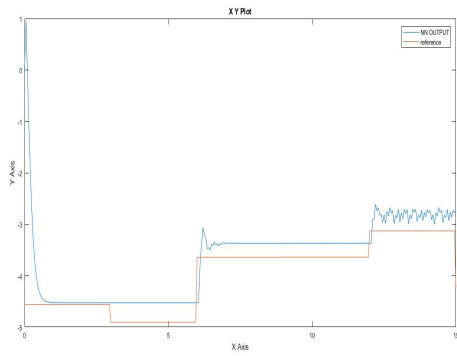
Figure 3.13: Continuous steering tank control with NN-MPC 1st order ref

2. **illustration 2** This example represent the system behaviour of a 1 dimensional car damped with a nonlinear damping force and a mechanical damping constant, which was taken equivalent to the worst case scenario.

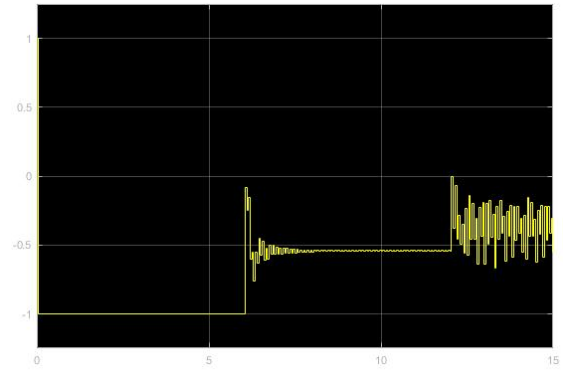
$$\ddot{x} = 20u - \dot{x}^2 \text{sign}(\dot{x}) - 0.5 \quad (3.20)$$

figure 3.14a represent the output signal of the controlled car with a control signal represented in figure 3.14b.

NN predictive controllers depends highly on the reference signal as well as their variations. Moreover, the system's output and the data set's elements included during the training process have a significant impact on the convergence speed.



(a) speed control of a 1D car



(b) control signal of the 1D car

Figure 3.14: Car in 1 dimension system

Finally the parameters used within the optimizer program determine the form and characteristics as well as the behaviour of the system's response.

figure 3.14 provides an insight to the behaviour of the plant when the data set training elements are focused on the extremum of the plant's output, while using the Cyclic Steered Random Search with Backtracking (CSRCHBAC) algorithm to minimize the performance measure.

Chapter 4

Simulation and Results

4.1 Introduction

The preceding chapter (chapter 3) extensively explored the theories behind the considered controllers, each sharing a similar architecture but employing distinct techniques. Thus, it is crucial to compare their respective performances.

Before delving into this investigation, it is essential to clarify the concept of objective truth. Experimental results attain objectivity only when multiple researchers consistently achieve the same outcomes. To ensure clarity and mitigate ambiguities, our comparison focuses on two distinct nonlinear SISO models.

The performance metrics under scrutiny include disturbance rejection, dynamic error, control effort, response times and overshoot.

4.2 Case Study 1 : Liquid Level Control

4.2.1 Modeling

Figure 4.1 depicts the diagram of a liquid level control of a tank, which was the first system considered.

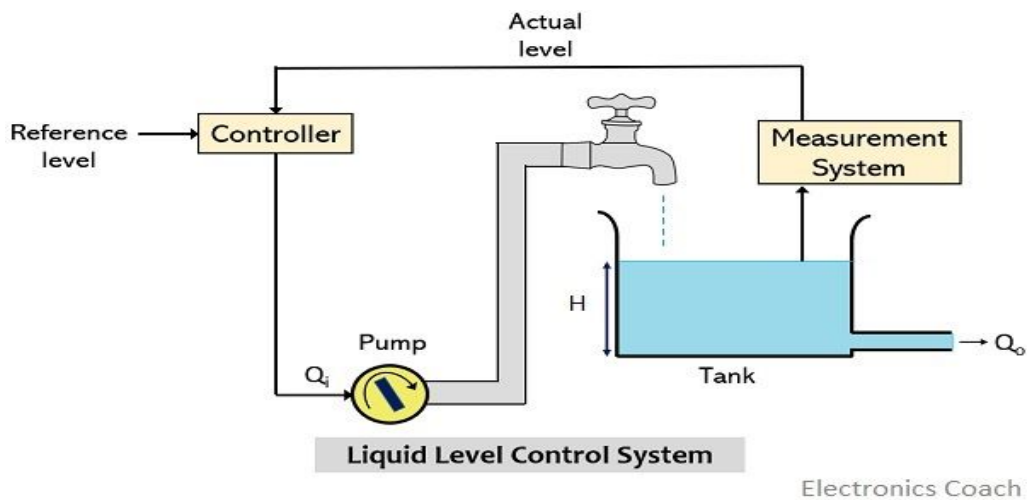


Figure 4.1: Liquid level control system [5]

Applying fluid mechanics to figure 4.1 provides a significant insight about the system studied. As commonly known in the physics field : The energy of a system is conserved

throughout an isolated system, thus it can be written mathematically such as $E_{initial} = E_{final}$. An imaginary portion of the water is taken into consideration, which enables us to apply the physical formulas as if it was a solid object, therefore:

$$\begin{cases} E_{initial} = \rho V g H = E_{final} = \frac{1}{2} \rho V v^2 \\ v = \sqrt{2gH} \end{cases} \quad (4.1)$$

with ρ being the density of the water

V is the volume taken into consideration

$E_{initial}$ is the initial energy

E_{final} is the final energy

$g = 9.81$ is the gravitational acceleration

H is the height of the system

v is the speed output from the tank.

Controlling the level(height) of the tank was the purpose of figure 4.1, thus the volume variation is considered :

$$\begin{cases} \dot{V} = A_{tank} \dot{H} = Q_{in} - Q_{out} \\ Q_{out} = A_{opening} v = A_{opening} \sqrt{2gH} \\ Q_{in} = u \end{cases} \quad (4.2)$$

with

A_{tank} represents the cross section area of the tank.

$A_{opening}$ represents the cross section area of the hole from which the water flows out.

Q_{in} is the in-flow of the water(control input)

Q_{out} is the out-flow of the water

eq.(4.2) can be simplified to a system equation of the form of the eq.(4.3):

$$\dot{H} = \alpha u - \beta \sqrt{H} \quad (4.3)$$

Taking $A_{tank} = 2m^2$ and $A_{opening} = 0.25m^2$, therefore eq.(4.3) becomes :

$$\dot{H} = 0.5u - 0.125\sqrt{H} \quad (4.4)$$

4.2.2 Simulation and Results

The uncontrolled tank level system has a nonlinear relationship between the flow-rate of the water and the level system; Figure 4.2 shows the relation between them.

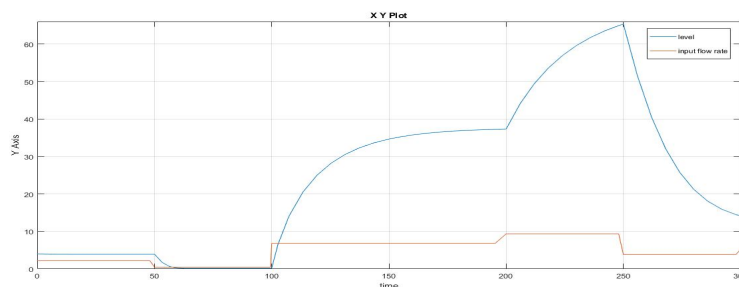
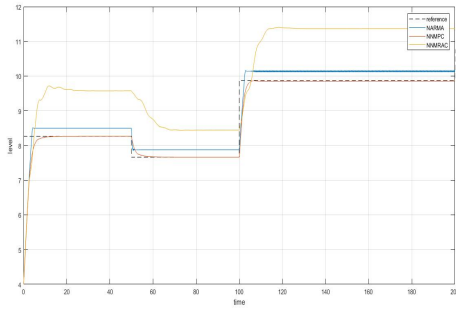
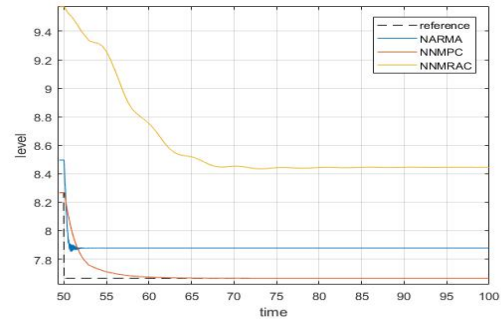


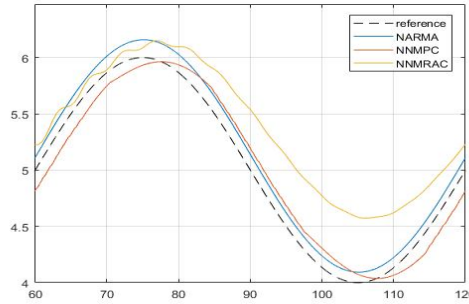
Figure 4.2: Input flow rate vs tank level of an uncontrolled system



(a) step reference output with different neural network controllers



(b) zoomed output with MPC, MRAC and NARMA-L2 controllers



(c) sinusoidal tank level control with NN controller

Figure 4.3: Tank level control with NN controller

Applying the neural network controllers seen in chapter 3:

Figure 4.3 depicts the behaviour of the response due to neural network controller trained with a input flow rate between 4 and 10 m^3/min .

Moreover, by considering a sinusoidal input reference, the following results-represented in figure 4.3c are obtained

4.2.3 Performance Study

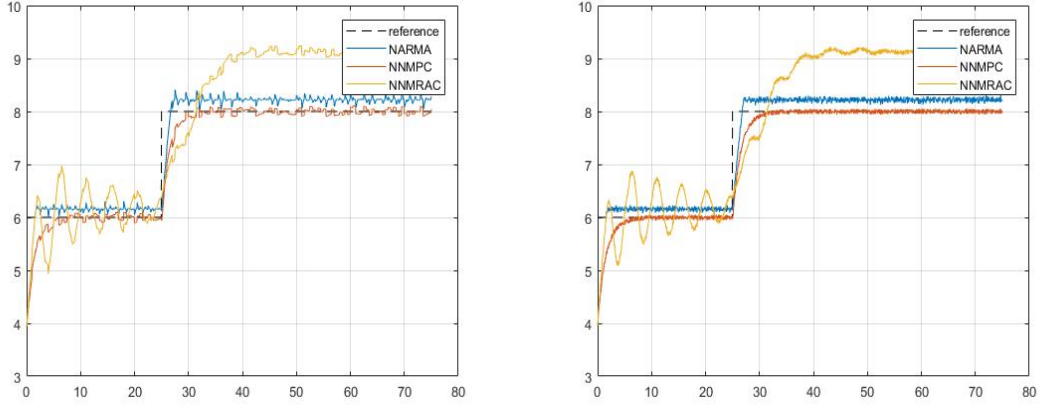
As mentioned in the introduction of this chapter. The performance studied of the system cover several aspects, ranging from the noise disturbance rejection to response time and overshoot.

4.2.3.1 Noise Disturbance Rejection

To study the effect of the disturbance on the output of the system; it is imperative to specify the kind of noise affecting the system, thus three cases were considered :

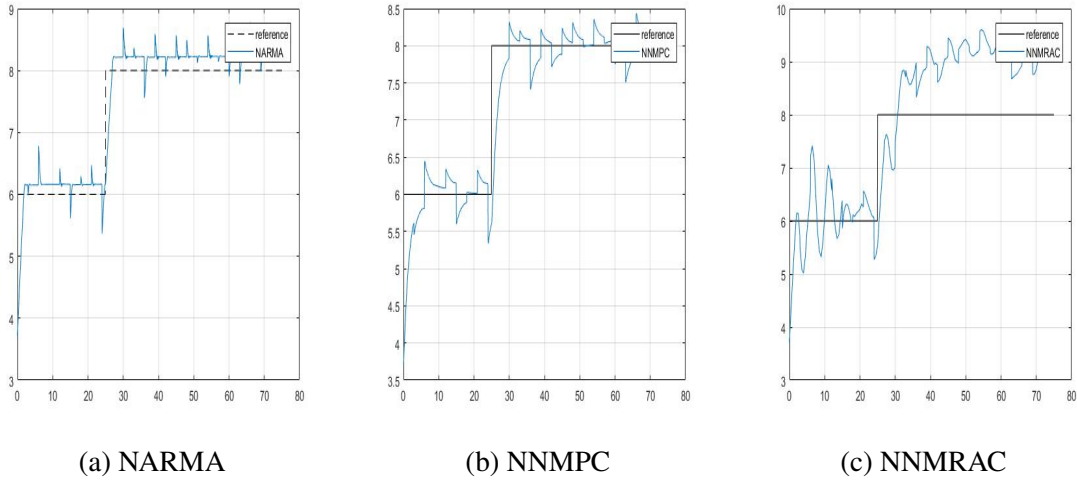
1. Figure 4.4a, which shows the effect of high frequency and high amplitude noise on the system
2. Figure 4.4b, which represents the effect of high frequency and low amplitude noise on the system
3. Figure 4.5 that depicts low frequency high amplitude noise effect.

Summarizing these results in table 4.1 using a Matlab script, that took as input data with noise, without noise and the noise itself.



(a) step response with high amplitude high frequency noise (b) response with low amplitude high frequency noise

Figure 4.4: High frequency noise on the tank system



(a) NARMA

(b) NNMP

(c) NNMRAC

Figure 4.5: High amplitude low frequency noise on the tank system

From the table 4.1, it can be seen that while NARMA L2 has the best high frequency disturbance rejection, it also has the worst low frequency disturbance rejection. On the other hand, NNMP has the best high frequency disturbance rejection but is unreliable in rejecting low frequencies.

Discussion :

The results of this simulation were as follows :

1. NNMRAC reduced the noise disturbance by half at both high and low frequencies (table 4.1) : that is due to the architecture of the neural network which consists of a plant estimator as well as a control signal estimator.

This was unexpected since modern MRAC might be more effective at reducing certain types of low frequency noise compared to high frequency noise

2. NNMP reduced the noise disturbance by 40% at high frequencies while it is unable to reduce the low frequency noise (table 4.1) : that is due to the approach used during the design, the minimization algorithm reduces the performance measure between a reference value and the output of the NN estimator.

Table 4.1: Tank's noise rejection

Noise Attenuation	Low Frequency (1Hz)	High Frequency(250Hz)
NNMRAC	50%	50%
NNMPC	0%	60%
NARMA L2	80%	0.8%

The NN estimator would consider the noisy output of the system to be the undisturbed. Therefore no noise reduction would occur.

3. NARMA L2 reduces the low frequency noise by 80% while the high frequency noise was reduced only by 0.8% (table 4.1) : that is due to the NARMA L2 design. Where a slow variation of a signal causes minimum effect on the controller response on the other hand, a sudden change may cause an even larger error.

This phenomenon can be explained by the NARMA formula :

$$\hat{y} = f(0) + \frac{\partial f}{\partial u}(0)u$$

4.2.3.2 Dynamic Error

The graphs represented in figures 4.6a and 4.6b depict the errors of the output of the system represented on figures 4.3a and 4.3b respectively

Moreover, the error of the sinusoidal input (figure 4.6c) provides a deeper understanding of the system behaviour

Table 4.2: Error comparison of the tank

Controller	maximum error	steady state error for a step input from 8.3 to 7.7	approximate dynamic error for the sinusoidal reference
NNMRAC	1.8	0.8	$e = 0.35 + 0.25\sin(\frac{2\pi}{56}t)$
NNMPC	0	0	$e = 0.2\sin(\frac{2\pi}{60}t)$
NARMA L2	0.2	0.2	$e = 0.08\sin(\frac{2\pi}{60}t)$

Table 4.2 shows in details the error of different responses for each type of controller, while it can be seen that the error of sinusoidal reference are close approximation, though they still provide an idea of the limits of each controller.

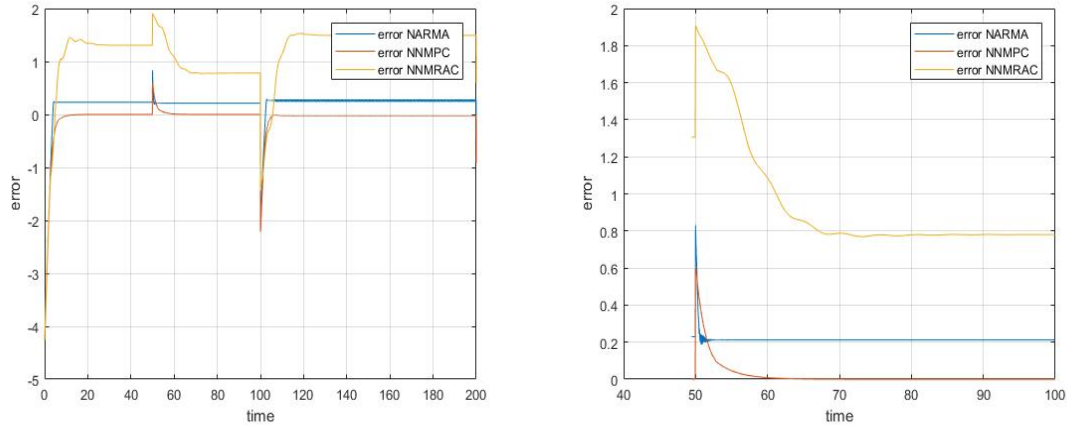
Discussion :

The results of this simulation were as follows :

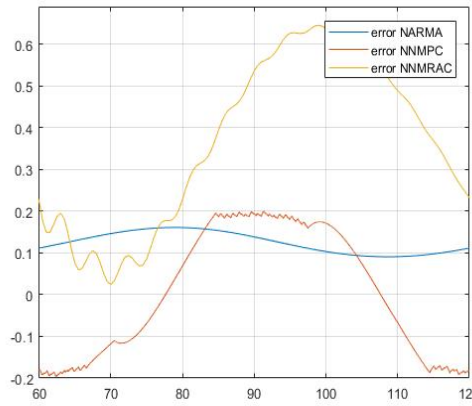
1. NNMRAC produced relatively large error compared to the other neural network controllers. it had a maximum error $e = 1.8$ throughout all the simulations of the tank control, while the dynamic error for a sinusoidal response was approximated to $e = 0.35 + 0.25\sin(\frac{2\pi}{56}t)$ (table 4.2).

these results were as expected due to the complexity of the controller. Since the two neural controllers were interconnected and dependent on each other which amplifies the error produced by the second network.

2. NNMPC had a non existent steady state error throughout the simulations. The dynamic error for a sinusoidal response was approximated to $e = 0.20\sin(\frac{2\pi}{60}t)$ (table 4.2).



(a) error of MPC,MRAC and NARMA-L2 controllers (b) zoomed error with MPC,MRAC and NARMA-L2



(c) error of the sinusoidal response of the tank

Figure 4.6: Error tank level control with NN controller

The steady state error as well as the dynamic error for sinusoidal response were expected to be close to zero as a result of using the minimization algorithm to minimize the error between the reference and the output of the system.

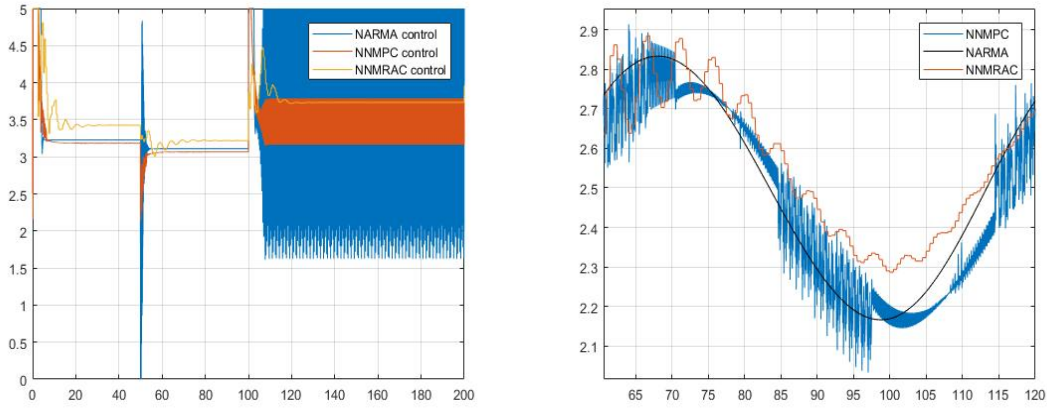
3. NARMA L2 had a maximum steady state error $e = 0.2$ throughout the simulations with a dynamic error for a sinusoidal response was approximated to $e = 0.08\sin(\frac{2\pi}{60}t)$ (table 4.2).

Both the steady state error and the dynamic error for sinusoidal input were less than expected.

4.2.3.3 Control Effort ΔU

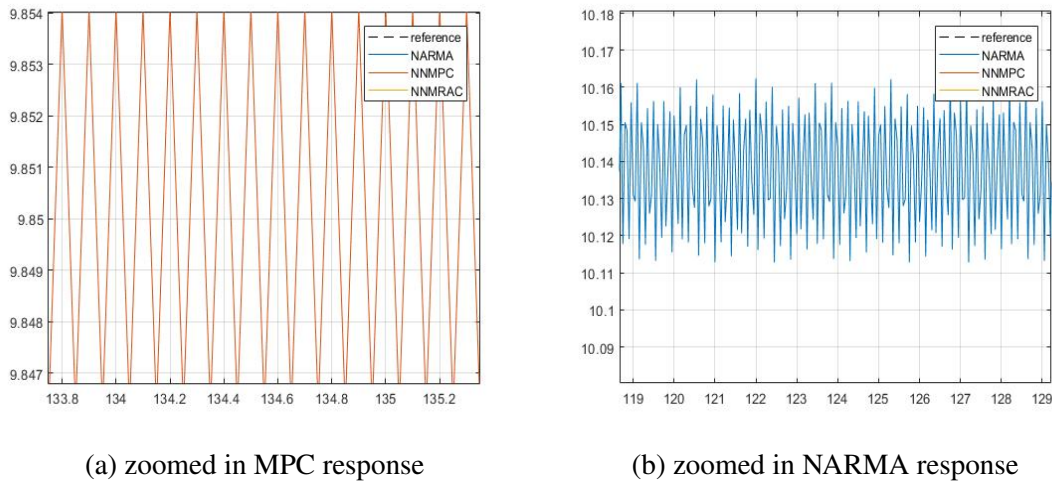
Figures 4.7a and 4.7b shows the control efforts of each controllers under different situations. figure 4.7a shows the control efforts of the response represented by figure 4.3a, thus enabling us to conclude that probably the less trained areas of the network may oscillate as shown in figure 4.8. which in return causes the control effort to increase dramatically.

Overcoming the variations of the control effort can be done by using a smooth reference input for the NARMA L2 and by over training the system for the NNMP as shown in figures 4.7a and 4.7b.



(a) control signals for step response of different controllers (b) control signal for the sinusoidal reference of different NN controllers

Figure 4.7: Tank control signal



(a) zoomed in MPC response

(b) zoomed in NARMA response

Figure 4.8: Tank output signal in under-trained areas

The NNMP can reduce the control effort by increasing their importance in the performance measure that is being minimized.

Table 4.3 summarizes the characteristics of the controllers' control effort under different circumstances.

Discussion :

The results of this simulation were as follows :

1. it has been witnessed that NNMRAC produced minimum amount of control efforts whether the it was for a step response as shown in figure 4.7a or a smooth response as shown in figure 4.7b. The control effort of this type of NN controller was similar to the control effort produced by a linear controller

These results were relatively unexpected, since the controller is forcing a plant to behave in a similar fashion to a reference plant.

2. NNMP it has been witnessed that NNMP produced a large amount of control effort specifically for a step response of under-trained areas of the NN controllers as 4.7a and a smooth response as shown in figure 4.7b. The control effort of the well trained area for the step response were seen to be close to non-existent.

Table 4.3: Tank's Control Effort

Response	step reference	Sinusoidal Reference
NNMRAC	Minimum over all the range of study	Minimum over all the range of study
NNMPC	Medium in under-trained area and non-existent in well-trained areas	Maximum in continuous varying references
NARMA L2	Maximum in under-trained area and non-existent in well-trained areas	Non-existent

This result was expected, which was a reason as to why the minimization algorithm had a parameter that can be modified to reduce the effort produced by the controller.

3. NARMA produced large effort only under two specific situations : The first situation is the sudden change for the step response. The second situation is for step response of under-trained areas.

This result was expected, since the NARMA L2 cannot handle sudden changes in the reference.

4.2.3.4 Response Time & Overshoot

Table 4.4 summarizes the system's performances depicted in figure 4.9 when different controllers are employed.

In that regards, NNMPC has an overwhelming better performance than the other neural controllers.

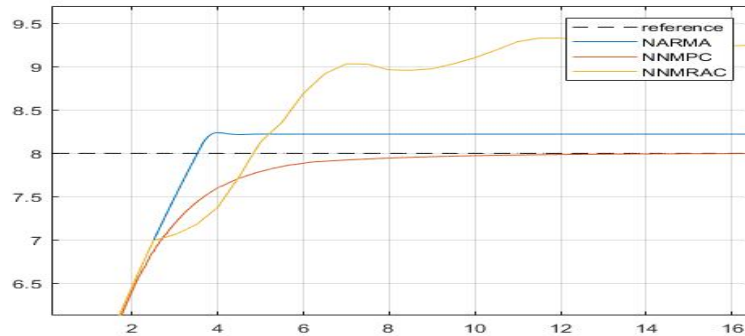


Figure 4.9: Tank step response

Table 4.4: Tank's Performance

Controller	Overshoot	Rising Time	Settling time
NNMRAC	$\frac{9.4-9.25}{9.25} \times 100 = 1.62\%$	$t_{6.7} = 2.1s$	$T_{ss} = 9s$
NARMA L-2	$\frac{8.3-8.2}{8.2} \times 100 = 1.21\%$	$t_{6.7} = 2.3s$	$T_{ss} = 3.6s$
NNMPC	0%	$t_{6.7} = 2.3s$	$T_{ss} = 3.6s$

Discussion :

The results of this simulation were as follows :

1. NNMRAC step response depends entirely on the reference model used. with a small error margin. the NNMRAC reference model was designed to have 0% overshoot, 2 seconds rising time and 3 seconds settling time. The results however were different with an overshoot of 1.62%, a rising time of 2.1 seconds which was relatively close to the desired value and a settling time of 9s.

These results were as unexpected. The NN controller was design to produce specific results but the performance obtained was different.

2. NNMPC produced excellent results with a 0 % overshoot, a rising time of 2.3 seconds which was relatively and a settling time of 3.6s.

While the overshoot was expected to be close to zero, the settling time as well as the rising time was expected to be larger due to the complexity of the minimization algorithm.

3. NARMA L2 produced excellent results with a 1.21 % overshoot, a rising time of 2.3 seconds which was relatively and a settling time of 3.6s.

These results were as expected. with a rising time of 2.3 second and a settling time of 3.6 seconds. However the overshoot was expected to be over 10 % due to the inability of the NARMA L2 to handle sudden changes.

4.3 Case Study 2: Motor Nonlinear Control System

4.3.1 Modeling

Figure 4.10 depicts the diagram of a motor nonlinear circuit, which was the second system considered. Applying Kirchhoff voltage law on the motor provides an insight about the

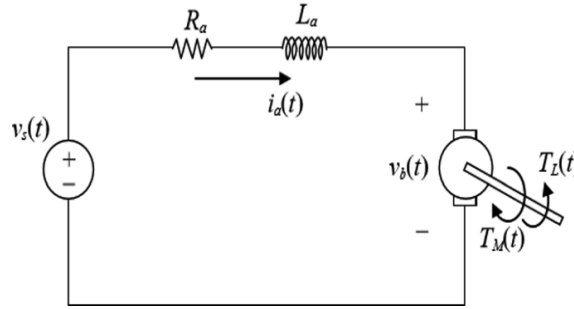


Figure 4.10: Motor nonlinear circuit system[7]

system's behaviour, thus it can be written mathematically as $\sum(V) = 0$. Moreover, by applying the torque law on the motor's load such as $\sum(\tau_i) = \tau$. It can be deduced that the state space representation is written as follows :

$$\begin{cases} v_a - R_a i_a - L_a \frac{di_a}{dt} - v_b = 0 \\ J\dot{\omega} = T_M - D\omega - T_L \end{cases} \quad (4.5)$$

where

ω - the angular speed

v_a - the input voltage

L_a - the total inductance 0.0917H

R_a - the total resistance 7.2Ω

D - the viscous-friction coefficient $0.0004Nm/rad/s$.

J - moment of inertia $0.0007046kgm^2$

i_a -motor current

[7] provides an example of a nonlinear motor with the values mentioned before. Moreover, it specified that : $v_b = K_m L_f i_a \omega$ and $T_M = K_m L_f i_a^2$.

where :

L_f -field winding inductance

K_m -torque/back EMF constant

$K_m L_f = 0.1236Nm/WbA$

4.3.2 Simulation and Results

The uncontrolled motor speed system has a nonlinear relationship between the motor current and the motor speed system;figure 4.11 shows the relation between them.

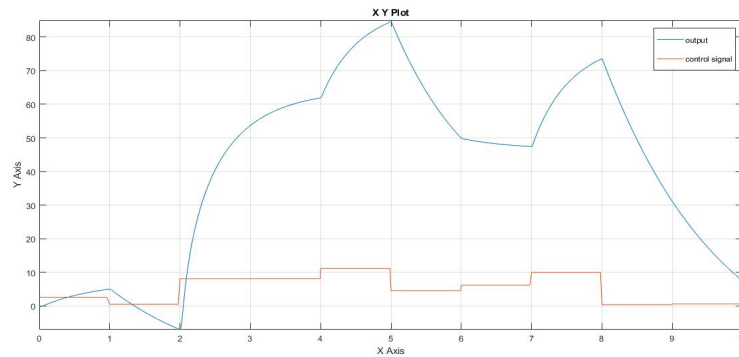


Figure 4.11: Input voltage vs motor speed of an uncontrolled system

Applying the neural network controllers seen in chapter 3:

Figure 4.12a and 4.12b depicts the step response behaviour due to different neural network controllers trained with an input voltage V_a between 0 and 15 V. Moreover, by considering a sinusoidal input reference, the following results-represented in figure 4.12c are obtained :

4.3.3 Performance study

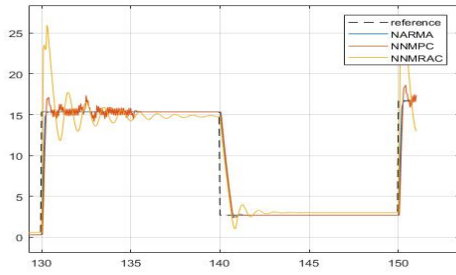
Once again. Similarly to the tank performance analysis; the performance studied of the system covers a wide area of analysis-ranging from the disturbance rejection until the response time and overshoot.

4.3.3.1 Noise Disturbance Rejection

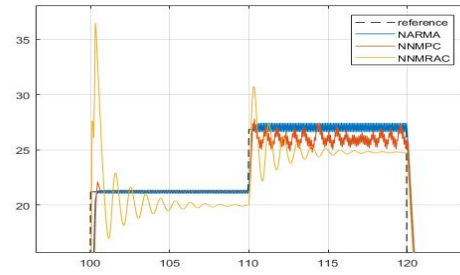
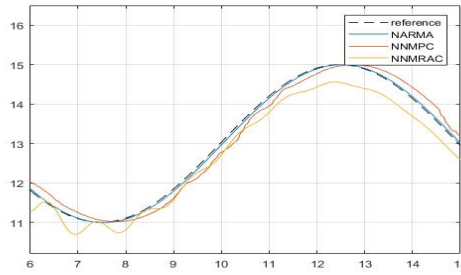
To study the effect of the noise disturbance on the system output; it is imperative to specify the kind of noise affecting the system, thus two cases were considered:

1. low frequency noise
2. high frequency noise

Figures 4.13 and 4.14 show the behaviours of the system when an outside disturbance is applied at the output.



(a) step response of the system with different controllers

(b) step response of the system in under-trained areas $V \notin [0, 15]$ 

(c) sinusoidal response with sine reference

Figure 4.12: Response of the system

Table 4.5: Motor's noise rejection

noise attenuation	low frequency(0.23Hz)	high frequency(1KHz)
NNMRAC	100 %	0%
NNMPC	100 %	0 %
NARMA L2	100 %	0 %

From table 4.5, all three controllers had the exact same behaviour to a sudden disturbance at the output. As a result, the low frequency noise were completely rejected while the high frequency noise could not be reduced.

One important observation seen during this experiment is that the low frequency noise caused the NARMA L2 and MPC to revise their control signal, thus producing an even better step response.

Discussion :

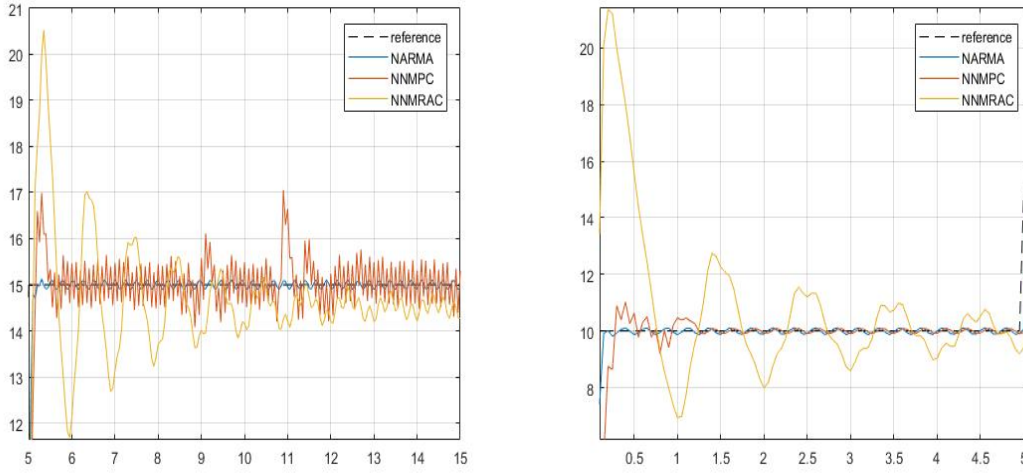
The results of this simulation were as follows :

1. NNMRAC completely rejects low frequency noise (table 4.5) but cannot handle high frequency noise : that is due to the architecture of the neural network which consists of a plant estimator as well as a control signal estimator.

This was expected since modern MRAC might be more effective at reducing certain types of low frequency noise compared to high frequency noise

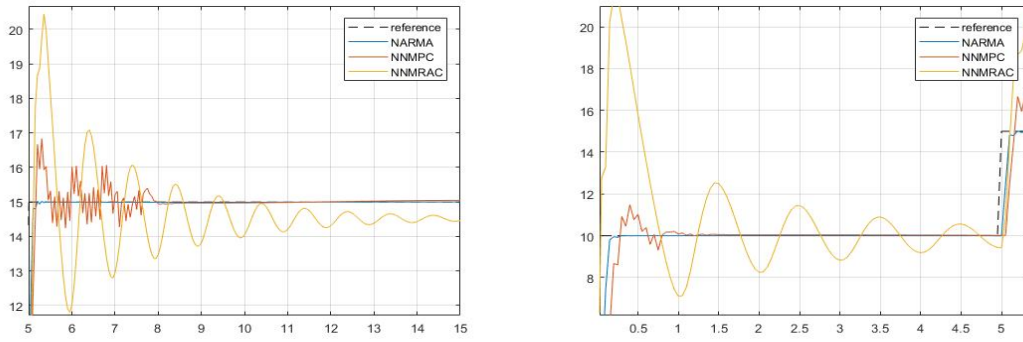
2. NNMPC reduced the noise disturbance by 100% at low frequencies while it is unable to reduce the high frequency noise (table 4.5) : that is due to the approach used during the design, the minimization algorithm reduces the performance measure between a reference value and the output of the NN estimator.

The NN estimator would consider the noisy output of the system to be the undisturbed. Therefore no noise reduction would occur, one way to solve such issue is to reduce the sampling time during the training stage.



(a) response with high amplitude high frequency noise (b) response with low amplitude high frequency noise

Figure 4.13: High frequency noise on the motor



(a) response with high amplitude low frequency noise on the motor (b) response with low amplitude low frequency noise

Figure 4.14: Low frequency noise on the motor

3. NARMA L2 reduces the low frequency noise by 100% while the high frequency noise was reduced only by 0% (table 4.5) : that is due to the NARMA L2 design. Where a slow variation of a signal causes minimum effect on the controller response on the other hand, a sudden change may cause an even larger error.

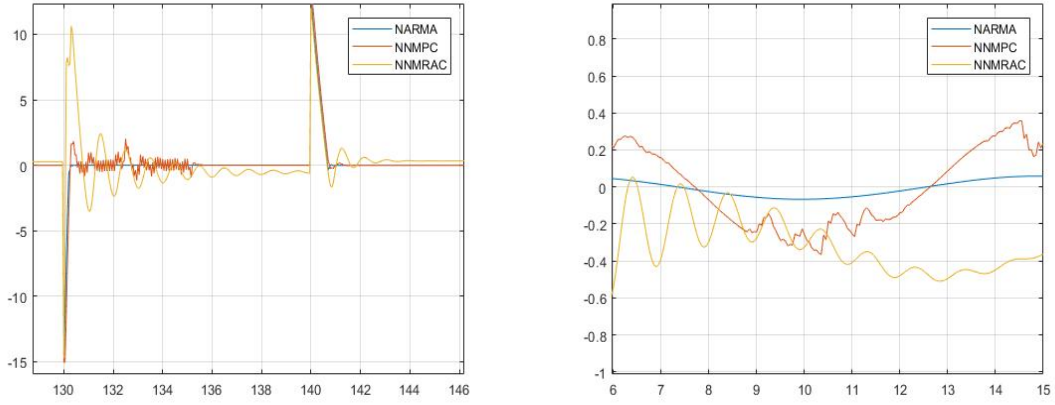
This phenomenon can be explained by the NARMA formula :

$$\hat{y} = f(0) + \frac{\partial f}{\partial u}(0)u$$

4.3.3.2 Dynamic Error

The graph represented on figure 4.15a depicts the errors of the output of the system for the step response of figure 4.12a, while the graphs represented on figure 4.15b depicts the errors of the output of the system for the sinusoidal response of 4.12c.

Table 4.6 shows in details the error of different responses for each type of controller, while it can be seen that the error of sinusoidal reference are close approximation, though they still provide an idea of the limits of each controller.



(a) dynamic error of MPC,MRAC and (b) dynamic error of the sinusoidal response of NARMA-L2 controllers the motor

Figure 4.15: Motor's speed dynamic error with different NN controller

Table 4.6: Error comparison of the motor

controller	max error	steady state error for step input from 15.34 to 2.67	approximate dynamic error for sinusoidal reference
NNMRAC	2	0.4	$e = -0.3 + 0.3\sin(\frac{2\pi}{16}t)$
NNMPC	0.05	0.003	$e = 0.2\sin(\frac{2\pi}{8}t)$
NARMA L2	0	0.006	$e = 0.05\sin(\frac{2\pi}{11}t)$

One significant observation would be that the NARMA L2 provides the best results when it comes to error minimization.

Discussion :

The results of this simulation were as follows :

1. NNMRAC produced relatively large error compared to the other neural network controllers. it had a maximum error $e = 2$ throughout all the simulations of the tank control, while the dynamic error for a sinusoidal response was approximated to $e = -0.3 + 0.3\sin(\frac{2\pi}{16}t)$ (table 4.2).

these results were as expected due to the complexity of the controller. Since the two neural controllers were interconnected and dependent on each other which amplifies the error produced by the second network.

2. NNMPAC had almost a non-existent steady state error throughout the simulations. The dynamic error for a sinusoidal response was approximated to $e = 0.2\sin(\frac{2\pi}{8}t)$ (table 4.2).

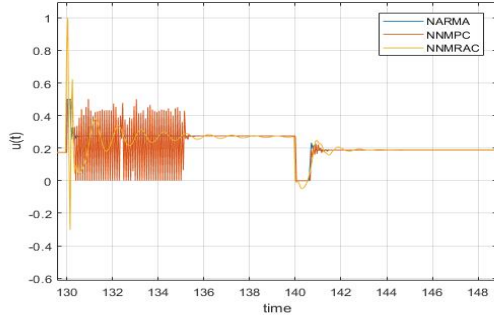
The steady state error as well as the dynamic error for sinusoidal response were expected to be close to zero as a result of using the minimization algorithm to minimize the error between the reference and the output of the system.

3. NARMA L2 had a maximum steady state error $e = 0.2$ throughout the simulations with a dynamic error for a sinusoidal response was approximated to $e = 0.05\sin(\frac{2\pi}{11}t)$ (table 4.2).

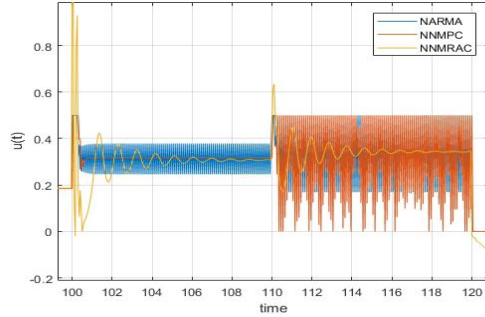
Both the steady state error and the dynamic error for sinusoidal input were less than expected.

4.3.3.3 Control Effort ΔU

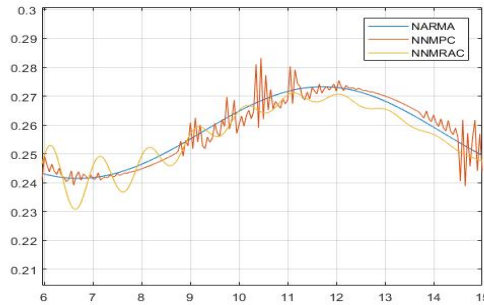
Figures 4.16 show the control efforts for the step response of each controller under different situations: Figure 4.16a shows the control effort for the step response represented in figure 4.12a, while figure 4.16b represents the control efforts for the step response shown in figure 4.12b, thus enabling us to conclude that probably the less trained areas of the network may oscillate. Therefore causing the control effort to increase dramatically.



(a) control effort



(b) control effort under-trained areas



(c) control effort with sine reference

Figure 4.16: Speed control signal of a motor

Overcoming the increase of the control effort could be done by using a smooth reference input as shown in figure 4.16c.

Table 4.7: Motor Control Effort

Response	Step reference	Sinusoidal Reference
NNMRAC	Minimum over all the range of study	Minimum over all the range of study
NNMPC	maximum during rising time and non-existent in the steady states and Medium in under-trained areas	Medium in continuous varying references
NARMA L2	Maximum in under-trained area and non-existent in well-trained areas	Non-existent

Table 4.7 summarizes the characteristics of the controllers' control effort under different circumstances.

Discussion :

The results of this simulation were as follows :

1. it has been witnessed that NNMRAC produced minimum amount of control efforts whether the it was for a step response as shown in figure 4.16a or a smooth response as shown in figure 4.16c. The control effort of this type of NN controller was similar to the control effort produced by a linear controller

These results were relatively unexpected, since the controller is forcing a plant to behave in a similar fashion to a reference plant.

2. NNMPc it has been witnessed that NNMPc produced a large amount of control effort specifically for a step response of under-trained areas of the NN controllers as 4.16a and a smooth response as shown in figure 4.16c. The control effort of the well trained area for the step response were seen to be close to converge to zero as the system stabilizes.

This result was expected. Moreover it can be improved by increasing the importance of the control effort in the performance measure of the controller.

3. NARMA produced large effort only under two specific situations : The first situation is the sudden change for the step response. The second situation is for step response of under-trained areas as shown in figure 4.16b. Otherwise the effort was minimum throughout the simulation.

This result was expected, since the NARMA L2 cannot handle sudden changes in the reference.

4.3.3.4 Response Time & Overshoot

Table 4.8 summarizes the system's performances depicted in figure 4.17 when different controllers are employed.

In that regards, NARMA L2 has an overwhelming better performance than the other neural controllers.

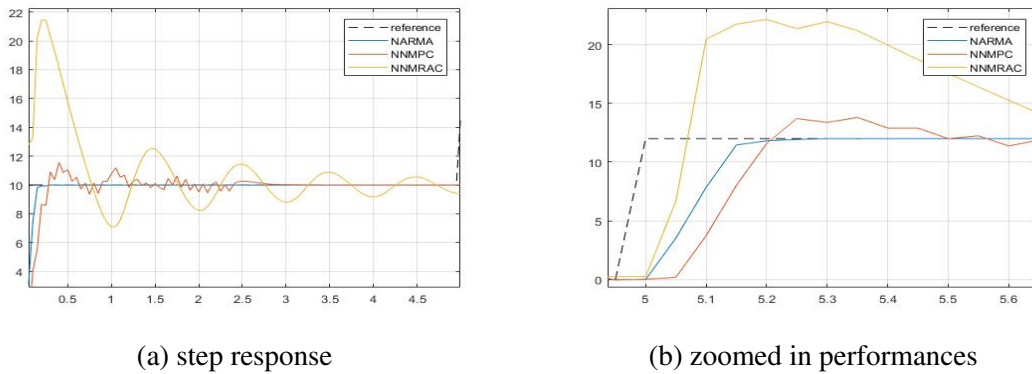


Figure 4.17: Speed response of a motor

Table 4.8: Motor's Performance

Controller	Overshoot	Rising Time	steady state time
NNMRAC	$\frac{22.14-12}{12} \times 100 = 84.5\%$	$t_{10.8} = 0.065s$	$T_{ss} = 4.5s$
NARMA L-2	0%	$t_{10.8} = 0.14s$	$T_{ss} = 0.28s$
NNMPc	$\frac{16.5-15}{15} \times 100 = 10\%$	$t_{10.8} = 0.189s$	$T_{ss} = 2.4s$

Discussion :

The results of this simulation were as follows :

1. NNMRAC step response depends entirely on the reference model used. with a small error margin. the NNMRAC reference model was designed to have 10% overshoot, 0.022 seconds rising time and 0.05 seconds settling time. The results however were different with an overshoot of 84.5%, a rising time of 0.065 seconds which was relatively close to the desired value and a settling time of 4.5s.

These results were as unexpected. The NN controller was design to produce specific results but the performance obtained was different.

2. NN MPC produced excellent results with a 10 % overshoot, a rising time of 0.189 seconds which was relatively and a settling time of 2.4s.

the results are close to the first case study, however the overshoot seems to be larger for fast systems.

3. NARMA L2 produced excellent results with a 0% overshoot, a rising time of 0.14 seconds which was relatively and a settling time of 0.28s.

These results were as expected. However was non-existent which goes against the idea that NARMA L2 is unable to handle sudden changes.

4.4 Conclusion

Throughout the simulations. It has been noticed that each controller has their own application areas. As a result of considering two systems, which can be characterized by their settling time : motor system is a fast system, while the tank system considered was 7 times slower.

The results were as follows:

1. slow systems (tank system):

A decent choice would be the neural predictive controller whose results were distinguished during the simulation, it proved to have non-existent overshoot, minimum rising time, the best settling time, and the best noise rejection at low frequencies and a medium noise and disturbance rejection at high frequencies. Moreover the error was almost non-existent.

The worst would be the neural network model reference controller whose results were proved to have the worst settling time, significant error increase and average disturbance rejection. However, it was the fastest in terms of rise time.

2. fast systems (motor system):

A decent choice would be the NARMA L2 controller whose results were distinguished during the simulation, it proved to have non-existent overshoot, minimum rising time, the best settling time, and a good noise and disturbance rejection. However, the downside were observed in its inability to handle high frequency noises.

The worst would be the neural network model reference controller whose results were proved unable to handle sudden changes in references causing an overshoot proportional to the change in reference, the worst settling time.

Slow system such as a tank system or a motion planning project are best controlled using the MPC, which utilizes the optimal control theories to provide a control policy. Fast

systems such as plane system dynamic control or a motor speed control are best controlled using the NARMA L2.

Neural network model reference on the other hand is more suited to control linear systems or non-linear system around a specific fixed point. this is due to the fact that NNMRAC has two neural controllers connected together in such a way that the second network is entirely dependent on the first network.

Conclusion

The design and implementation of advanced control techniques, specifically nonlinear controllers and neural networks, can significantly improve the performance and efficiency of a plant's responses. This thesis has explored the theoretical study and evaluations of three different neural network controllers: the Neural Network Predictive Controller, the Neural Network Model Reference Controller, and the Nonlinear Auto-Regressive Moving Average Controller.

After evaluating the response and behavior of the controlled plant's responses and accuracy, our findings indicate that neural network controllers can significantly enhance plant performance and response. However, there are challenges, such as the extensive data requirements for controller training and the substantial computational power needed, which suggest the need for further optimization of these algorithms.

Throughout the simulations, the results demonstrated that while neural network controllers provided enhanced responses with minimal information about the plants, each type studied has specific fields of application and limitations under certain conditions.

The key contributions of this work are:

1. A comprehensive analysis of the theoretical foundations and practical implementations of nonlinear and neural network-based controllers.
2. The design and implementation of neural network-based observers and the neural network controllers of interest.
3. Simulations of various plants and their compatibility with different control techniques in general, and with neural network-based controllers specifically.
4. An objective evaluation of the performance of three distinct neural network controllers, highlighting their potential benefits and challenges.
5. Insights into the necessary optimizations needed and future research directions for neural network controllers in control engineering.

In conclusion, this thesis has demonstrated the potential of neural network controllers to revolutionize modern control engineering by enhancing performance measures and increasing plant efficiency. However, to fully realize their potential, future research should focus on developing more optimal algorithms and general neural network controllers that are applicable regardless of the plant's behavior, or on developing stronger processors capable of handling vast amounts of data within the required response time. By addressing these limitations and challenges, we can create more robust and efficient control systems in the future.

Future Work

The main contribution of this report has been to investigate and compare the performance of three neural network controllers on nonlinear SISO systems. Through this project,

significant insights and understanding were obtained about the strengths and limitations of each controller, highlighting their areas of application in the real world.

However, further research is required in this field of study, such as extending and generalizing from SISO plant systems to MIMO plant systems. By addressing this issue, it is possible to improve our understanding of neural network control techniques. Moreover, researching other techniques for adjusting the parameters of the neural network controllers to improve their performance is essential.

Another interesting field of investigation involves adaptive control strategies that allow the neural network controllers to adjust their parameters online under unforeseen changes in the system or operating conditions. Additionally, exploring the integration and creation of neural network-based controllers with other types of nonlinear controllers could provide valuable advancements.

Bibliography

- [1] Hazlina Selamat and Rubiyah Yusof. *Introduction to Adaptive and Self-Tuning Control*. Penerbit UTM Press, 1st edition, 2014.
- [2] Real-time comparison of a number of predictive controllers. *ISA Transactions*, 46(3):411–418, July 2007.
- [3] Keinosuke Fukunaga. *The Artificial Neural Network Book*. Academic Press, 1990.
- [4] Heidar A. Talebi, Farzaneh Abdollahi, Rajni V. Patel, and Khashayar Khorasani. *Neural Network-Based State Estimation of Nonlinear Systems: Application to Fault Detection*, volume 395 of *Lecture Notes in Control and Information Sciences*. Springer, Berlin, Heidelberg, 2010.
- [5] Google. Google images. <https://www.google.com/imghp>. Accessed: 2024-06-25.
- [6] Direct adaptive control based on improved neural network for omni-directional mobile robot. *IEEE*.
- [7] S. Mehta and J. Chiasson. Nonlinear control of a series dc motor: theory and experiment. *IEEE Transactions on Industrial Electronics*, 45(1):134–141, Feb 1998.
- [8] Donald E. Kirk. *Optimal Control Theory: An Introduction*. Dover Publications, 2004.
- [9] Vadim Utkin. *Control Systems, Robotics, and Automation*. Springer Verlag, 1978.
- [10] R. Decarlo and S. Žak. Quick introduction to sliding mode control and its applications. *IEEE*, 2008.
- [11] Shankar Sastry. *Nonlinear System Analysis, Stability, and Control*. Springer Verlag, 1999.
- [12] E. F. Camacho and C. Bordón. *Advanced Textbook in Control and Signal Processing Model Predictive Control*. Springer Verlag, 2nd edition, 1999.
- [13] Seborg et al. *Process Dynamics and Control*. Wiley, 2010.
- [14] Steven H. Strogatz. *Nonlinear Dynamics and Chaos with Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2nd edition, 2015.
- [15] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. CRC Press Taylor & Francis Group, 2015.
- [16] Kenneth Hunt, George Irwin, and Kevin Warwick. Neural network engineering in dynamic control systems. *Advances in Industrial Control*, 1995.

- [17] George Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989.