2015

Conversion of neural network models to state-space models for model-based control design

Sai Susheel Praneeth Kode

CONVERSION OF NEURAL NETWORK MODELS TO STATE-SPACE MODELS FOR MODEL-BASED CONTROL DESIGN

by

SAI SUSHEEL PRANEETH KODE

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
in
The Department of Mechanical and Aerospace Engineering
to
The School of Graduate Studies
of
The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2015

In presenting this thesis in partial fulfillment of the requirements for a master's degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this thesis.

K. Susheel Praneeth

Sai susheel praneeth Kode

3/31/2015

(date)

THESIS APPROVAL FORM

Submitted by Susheel Praneeth Sai Kode in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Mechanical Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the thesis committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this thesis. We further certify that we have reviewed the thesis manuscript and approve it in partial fulfillment of the requirements for the degree of Master of Science in Engineering in Mechanical Engineering.

3/31/2015
Dr. Farbod Fahimi (Date) Committee Chair

04/01/2015
Dr. Mark Lin (Date)

3/20/2015
Dr. Chang-kwon Kang (Date)

4/2/15
Dr. Keith Hollingsworth (Date) Department Chair

4/13/15
Dr. Shankar Mahalingam for S. Mahalingam (Date) College Dean

4/13/15
Dr. David Berkowitz (Date) Graduate Dean

ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Master of Science in Engineering College/Dept. Engineering/Mechanical and Aerospace Engineering

Name of Candidate Sai susheel praneeth Kode

Title Conversion of Neural Network Models to State-Space Models for Model-Based Control Design

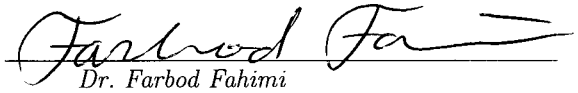
The objective of this research is to implement an artificial neural network into a closed-loop model based control law that requires a state space model. Model based control laws are strongly preferred due to the existence of mathematical proofs for their performance. With mathematical proof of performance, the controller is guaranteed to work correctly within a designed domain of operation. However, deriving physics based mathematical models and accurately identifying the parameters of such models are cumbersome, or sometimes impossible, for complex nonlinear systems. This limits the application domain of model based controllers.

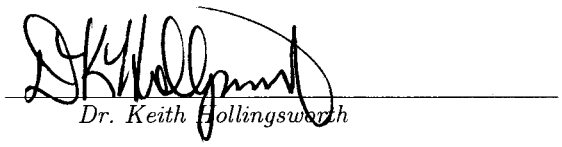
Neural networks are a great tool for modeling nonlinear systems, because they eliminate the physics based modeling and ease the identification process. However, controllers that have been introduced for neural network model lack mathematical proof of performance. Without mathematical proof of performance, the domain of operation in which the controller works correctly is unknown. So, unexpected controller behaviors may emerge.

The methodology proposed in this thesis combines the ease of modeling using neural networks with the mathematical proof of performance using model based control design.

Identification data is collected from the system by applying known inputs and recording the system response. A neural network is fitted to the collected data to generate a neural network model. Using the proposed method in this thesis, a state-space model is extracted from the neural network model. A model based controller uses the extracted state-space model to generate control commands.

In this thesis, a mobile robot is used as an example to demonstrate the implementation of the proposed methodology. First, as a proof-of-concept, the control commands are applied to the simulated neural network model representing the mobile robot. The results show that the robot successfully executes any user-defined command. Next, the methodology will be tested on a real mobile robot, and needed improvements will be made.

Abstract Approval: Committee Chair 
Dr. Farbod Fahimi

Department Chair 
Dr. Keith Hollingsworth

Graduate Dean 
Dr. David Berkowitz

ACKNOWLEDGMENTS

I am very thankful to Dr. Farbod Fahimi, for providing me a wonderful opportunity to work with him. He had dedicated his valuable time to give proper suggestions for my thesis. He stood as a backbone for my research work and his encouragement gives me enough strength to step forward with confidence, when I encountered some hurdles. Without his presence, I could not accomplish my research goal. He is the finest person I have ever seen in my life. I profusely grateful to him as it is been honor working with him. I would like to thank my examination committee Dr. Mark Lin and Dr. Chang-kwon Kang guided me in writing my thesis more efficiently. I would like to thank Dr. Keith Hollingsworth to support me with a Teaching Assistantship without which I wouldn not have a chance to accomplish my research. I would like to thank Claudia Meyering for her wonderful support in guiding me about the course work. She is one of the kindest persons I met in my life. I personally thank Dr. Huggy Ann and Dr. Kim Ann being very supportive in my life without them I cannot even think of my journey in University of Alabama in Huntsville (UAH).

Without mentioning about Phanidra Reddy, I would have drowned in the ocean of sadness. I would be thankful to him for my entire life. I would like to thank James Steele, UAH technical editor without him I would not have an opportunity to showcase my research with Dr. Fahimi to outside world. I am thankful to Tirumal, Sujatha and their son Akshaj for their love and wonderful support. I need to thank

my friends Rajan and Tripty to part of my journey who boosted me with lot of confidence. I sincerely thank Poornesh for helping me in means of transport. I would like to thank Surya Teja for his encouragement boost me up to accomplish my project successfully. I would like to thank Jerry for helping me with my research. I sincerely thank my roommates Prem, Vinay and Arjun for their time to time guidance and leave me with unforgettable moments in my life. I am indebeted to Shankarappa and his son Naresh Kumar, who are responsible for my higher studies in the United States of America. I would like to thank my seniors Malikarjuna, Karun, Harish, Nishanth, Nikhil for their immense support. Somu, Sowmya, Avadhani, Srinivas and Mahesh for being with me in tough times. Bharath, Venu, Venky, Harshinya, Yamini drived me to come out of the darkness. I would like to thank my sister Supreetha for taking care of my parents in my absence. I wish to show my gratitude to my parents, who endured several sufferings in their lives for the well being of their children. I would like to promise my parents that I will look after them at my best. I would also thank my grand parents for their love and support. Heartful thanks to UAH for everything, without which I would not have dreamt of achieving anything in my life. I would like to thank God for giving me enough strength to write this dissertation with everyone's support.

TABLE OF CONTENTS

PAGE

LIST OF FIGURES

FIGURE PAGE

LIST OF TABLES

TABLE PAGE

LIST OF SYMBOLS

SYMBOL	DEFINITION
a	PWM input slope, intercept
a_G	Acceleration due to gravity
b	PWM input slope, intercept
B	State matrix
C	Center of axle
C_{eq}	Damping coefficient associated with the robot
$\vec{e}_1$	Body axis of the robot
$\vec{e}_2$	Body axis of the robot
$\vec{E}$	Error vector used in sliding mode control law
e_v	Error in velocity
e_w	Error in angular velocity
F_L	Force in left wheel of the robot
F_R	Force in right wheel of the robot
f_f	Frictional force
$f(x)$	Smooth function used in NN
$\vec{f}(\vec{q}(t))$	Unsystematic disturbance

G	Center of gravity
H_D	State inversion matrix
H	State matrix
I_{eq}	Moment of inertia of the robot
K_d	Derivative gain
K_p	Proportional gain
K	Gain matrix
$\vec{k}$	Unit vector
l	Distance between center of axle and center of gravity
L	Hidden layers used in NN generation
M_{eq}	Equivalent mass of the robot
M_f	Moment of force
N_D	State matrix obtained from NN model
$\vec{q}$	Current state vector
$\dot{\vec{q}}$	First derivative of state vector
$\vec{q}^{d}(t)$	Desired state vector
$\vec{q}(t+\delta t)$	Future state vector
$\vec{r}_{G/C}$	Radius vector with respect to center of gravity and axle
T	Track of the robot

t	Time step
t_{shift}	Time shift
$\vec{U}(t)$	Control input vector
U_L	Speed corresponding to left wheel of the robot
U_R	Speed corresponding to right wheel of the robot
V_G	Velocity due to gravity
V_L	Velocity corresponding to left wheel of the robot
V_R	Velocity corresponding to right wheel of the robot
V_L^d	Desired velocity of robot in left wheel
V_R^d	Desired velocity of robot in right wheel
$\dot{V}_R$	Acceleration of the robot in right wheel
V_C	Actual velocity of robot
$\dot{V}_C$	Acceleration of robot
V_C^d	Desired velocity of robot
V^T	Matrix of first-layer weights
V_0	Constant velocity
V_{ci}	Robot's model velocity
V_{ni}	NN model velocity
W^T	Vector activation functions

x^1	Body axis orientation of analytical robot
y^1	Body axis orientation of analytical robot
δu	For small control input
δt	For a small time interval
ΔV_i	Velocity difference
$\Delta\omega_i$	Angular velocity difference
$\epsilon(x)$	Neural network functional approximation error
η_i	Safety factor for stability analysis
λ	Gain matrix used in sliding mode control law
σ	Gain matrix used in sliding mode control law
θ	Orientation(angle)
τ	Time constant
ϕ	Control gain matrix
ω_{ci}	Robot's model angular velocity
ω_{ni}	NN model angular velocity
ω_c	Actual angular velocity
ω_c^d	Desired angular velocity
ω	Angular velocity
ω_0	Constant angular velocity

I want to dedicate my thesis to my Dad and Mom, without whom I would be nothing.

Stay hungry, stay foolish.

—Steve Jobs

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

Traditionally, a system may be automated by implementing a mathematical closed-loop feedback control law. A control law is capable of driving one or more actuators in a way that will manipulate the state, or situation, of the system by using information of the current state, desired state, and physical characteristics of the system. Thus, the stability of the closed-loop system may be analyzed given knowledge of its physical characteristics, desired behavior, and anticipated disturbances from the environment. Problems with this traditional approach to automation design include unknown or variable system characteristics, lack of concise mathematical model of the system, or inefficiency in computing the control commands resulting from an extremely complex or non-linear system.

1.2 Proposed Method

In machine learning and cognitive science, artificial neural networks are a family of statistical learning algorithms inspired by biological neural networks (the central nervous systems of animals, in particular the brain) and are used to estimate or ap-

proximate functions that can depend on a large number of inputs and are generally unknown. Artificial neural networks are generally presented as systems of interconnected neurons which can compute values from inputs, and are capable of machine learning as well as the pattern recognition. Examinations of the human's central nervous system inspired the concept of neural networks. One method of modeling the behavior of a system is to use an artificial neural network. Thus neural networks are capable of learning the way a robot operates and the characteristics of its operating environment. Additionally, a neural network is more computationally efficient than the conventional control methods. Some of the conventional control methods includes as proportional-derivative control, feedback liberalization, adaptive feedback liberalization control. A neural network may be obtained by training the neural network based on detected action-reaction response from open loop tests. By modeling the dynamics of a system, a neural network inherently includes information on the physical characteristics of the system such as mass, moment of inertia, etc. This information may be extracted from the neural network mathematically and then applied to the motion control law. The extraction technique described in this thesis is new, fairly simple, and well formulated for the problem at hand. By understanding the relationship between a system's current state, control input and future state, the so called state-space representation of the system may be algebraically constructed. The goal of the research is to demonstrate the feasibility of using an artificial neural network embedded in a closed-loop model-based control law that requires a state-space model to control the motion of a system.

1.3 Categories of Existing Methodologies

Lewis et al. [1] review multiple strategies for incorporating the neural network feedback control laws. Feedback control involves the measurement of output signals from a dynamical system or plant, and the use of the difference between the measured values and certain prescribed desired values to compute system inputs that cause the measured values to follow or track the desired values. In feedback control design it is crucial to guarantee by rigorous means both the tracking performance and the internal stability or boundedness of all variables. Failure to do so can cause serious problems in the closed loop system, including instability and unboundedness of signals that can result in system failure or destruction.

The use of NN in control systems was first proposed by Werbos [2] and Narendra [3]. NN control has had two major thrusts: Approximate Dynamic programming, which uses NN to approximately solve the optimal control problem, and NN in closed-loop feedback control. Many researchers have contributed to the development of these fields. Several methodologies are illustrated [4] in the next section.

1.3.1 Indirect Method

In recent years, there has been a great deal of effort design feedback control systems that mimic the functions of living biological systems [5]. There has been great interest recently in universal model-free controllers that do not need a mathematical model of the controlled plant, but mimic the functions of the biological process to learn about the systems they are controlling on-line, so that performance improves

automatically. Techniques include fuzzy logic control, which mimics the linguistic and reasoning functions, artificial neural networks. It is generally understood that NN provide an elegant extension of control techniques to learning systems. Neural networks have been used in feedback control law as both system identifiers and also as controllers. System identifier on other side trains the network and builds a computational model in real time. System information is typically passed on to Neural network controller, which helps to generate the control input as shown in Figure 1.1. Solid lines shown in the figure represents the control signal where as the dotted line represents the training signal [6].

1.3.2 Direct Method

A somewhat more straightforward method of using a NN in feedback control is to combine the tasks of system identification and control into single NN. Figure 1.2 shows that system identifier and neural network controller are combined [7]. Current methods are mathematically complex due to training real time and undesirable due to uncertain stability margins.

1.3.3 Feedback/Feedforward Method

The challenge in using NN for feedback control purposes is to select a suitable control system structure, and then to demonstrate using mathematically acceptable techniques how the NN weighs can be tuned so that closed-loop stability and performance are guaranteed. Figure 1.3 represents the block diagram of feedback/feedforward method of neural network control.

1.3.4 Model-based Control Design Method

Control methodology approach is a combination of neural network model and model based controller, this proposed method was shown in Figure 1.4. Controller design using NN may be simplified by using the static NN as system identifier and separate mathematical control law. The proposed method is much simpler than previous approaches because the tuning the network in real time is not required.

1.3.5 Advantages and Disadvantages of all Categories

Direct control is more efficient, and involves directly tuning the parameters of an adjustable NN controller. The challenge in using NN for feedback control purposes is to select a suitable control system structure, and then to demonstrate using mathematically acceptable techniques how the NN weighs can be tuned so that closed-loop stability and performance are guaranteed [8]. The neural network does not accurately model the system during training in Indirect method. There is no Mathematical guarantee that controller can drive the error to zero is the limitation of the direct method. By offline training, we definitely have an accurate and verified NN model. Using Model-based control design, we can address our certain stability margins of current methods.

1.4 Objectives and Approach

The objective of this research is to implement an artificial neural network into a closed-loop model based control law that requires a state-space model. To

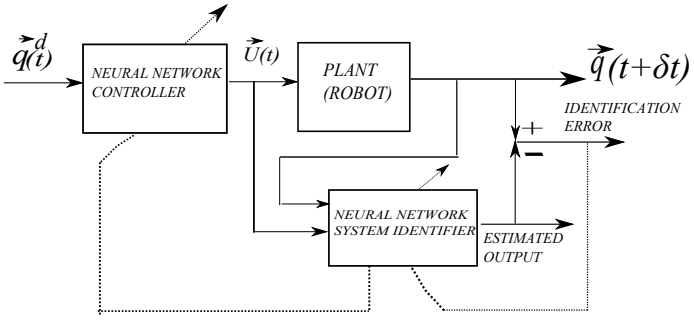


Figure 1.1: Indirect method of neural network control

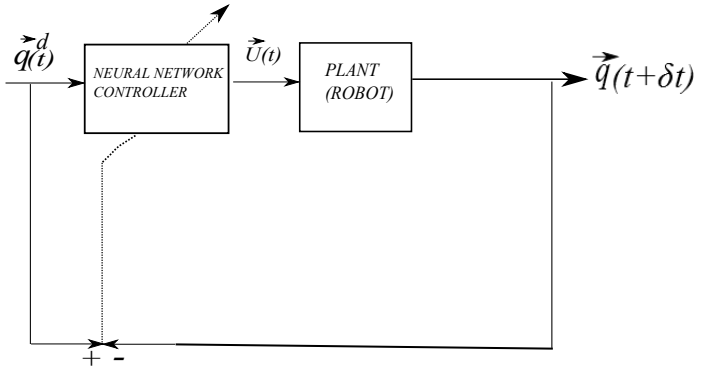


Figure 1.2: Direct method of neural network control

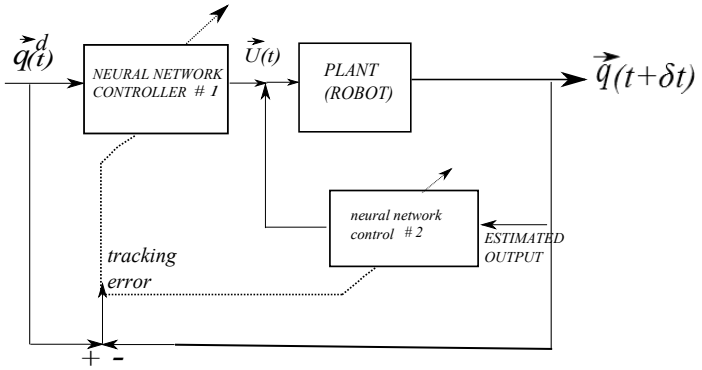


Figure 1.3: Feedback/feedforward method of neural network control

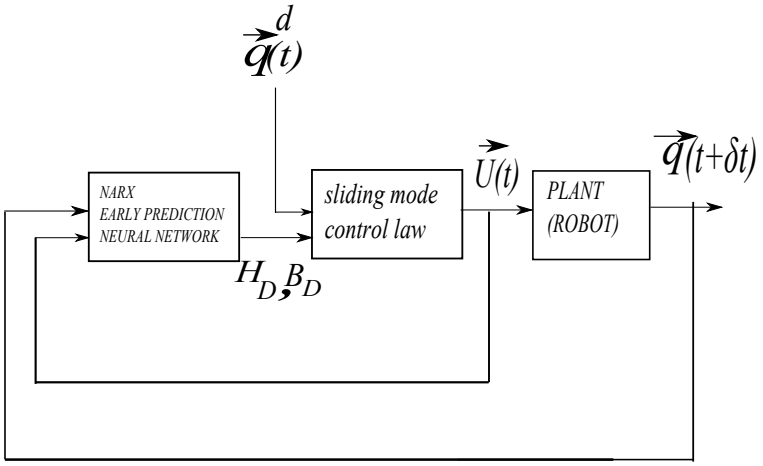


Figure 1.4: Model-based neural network control

demonstrate the implementation of the proposed method, which includes a state space extraction, the proposed method is validated in simulation on a ground robot. The suggested approach is as follows.

To validate the proposed extraction technique, a dynamical system must be defined for testing purposes. The system under consideration in this research is a Hilare-type robot, and an analytical dynamical robot will be derived to allow for simulation testing and validation. Procedure is discussed using following steps.

Step 1: Derivation of an analytical dynamic model. The physical parameters of an example autonomous ground robot will be defined. The kinematic and dynamic equations of motions will be derived and implemented in a state-space model. The dynamic model acts as a virtual robot with unknown dynamic model that needs to be controlled. The model will also be used as a reference to compare

with the predictions of the artificial neural network and as the plant in simulation of the control law.

Step 2: Formation of an artificial neural network dynamic model estimator. To closely approximate the action-reaction response of the virtual real robot, a neural network will be formulated from data produced in simulation by the analytical dynamical model. The data used consists solely of the state and control input information from a carefully selected open loop simulation. The neural network does not require information on the physical parameters of the robot. The validity of the neural network as an early prediction state estimator will be tested against the analytical dynamical model. Finally, the neural network will be implemented in a closed-loop control law for the virtual robot and simulated in software.

Step 3: Derivation of a model-based state-space control law. A control law will be defined and used in the series with the neural network. The control law will use the extracted state space model from the neural network to produce control input to drive the robot's actuators in a way that maintains desirable path or motion. Desirable path may be a straight, curved or any as per the user specifications.

Step 4: Verifications with simulations. To demonstrate the effectiveness of the proposed approach, the motion of the robot with the newly implemented closed-loop control law will be simulated. The Matlab/Simulink software [9] will be used to simulate the motion of the robot and demonstrate the effectiveness of the control method.

1.5 Summary

The research presented will propose a new method of motion control of a robot and validate its performance. The new method uses a neural network in the loop with a mathematical control law to provide estimations of the motion of the robot. This new technique does not require information on system parameters, does not requires information from a mathematical dynamical model of the system, and may be more computationally efficient than the traditional control approaches for systems that are complex and non-linear. In the summary, this thesis presents a new approach to implementing an artificial neural network into closed loop model based state space control law to govern the motion of an autonomous robot.

CHAPTER 2

DYNAMIC MODEL OF THE ROBOT WITH THE SPEED REGULATOR

2.1 Introduction

In this chapter, a dynamic model of the Hilare-type robot is developed with a speed regulator based on the free body and kinetic diagrams. In automatic control, a speed regulator is a device which has the function of maintaining a designated characteristic. It performs the activity of managing or maintaining a range of values in a robot [10]. Using these Figure 2.1 and Figure 2.2 the equations of motion can be derived. For dynamic modeling of the robot with speed controller, one must assume that the speed commands given to the speed controller is the input. The main objective of this chapter is to derive the equations of motion in state space form. Simulation results of the dynamic model is described later in the chapter.

2.1.1 Equations of Motion when Center of Gravity and Center of Axle Coincide

Consider a Hilare type of robot shown in the Figure 2.1, Figure 2.2 with local x^1 and y^1 axes, where C and G representing the center of axle and center of gravity.

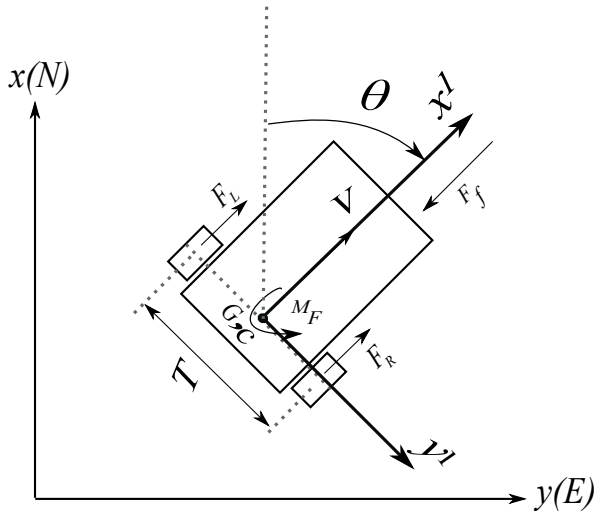


Figure 2.1: Free-body diagram

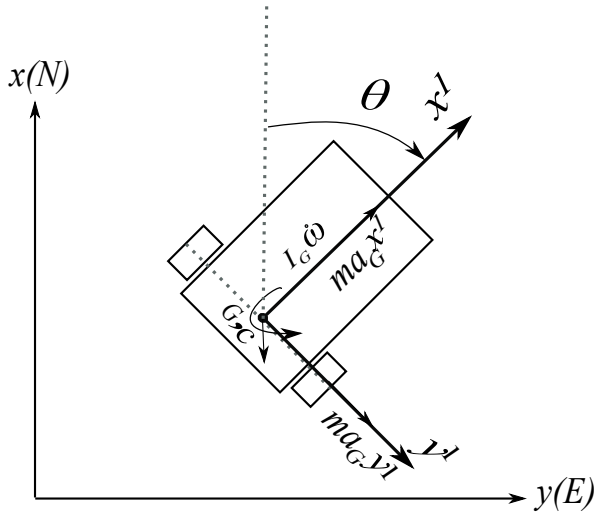


Figure 2.2: Kinetic diagram

F_L and F_R represent the forces of left and right wheels and F_f is the frictional force where V and ω represent the linear and angular velocity of the robot. The distance between the centers of two wheels is represented by T. The orientation of the robot θ is the angle made by the line with the x^1 axis. All the forces of the robot are balanced with the inertial forces. All the moments are balanced to the inertial moments.

$$m_{eq}\dot{V} = F_L + F_R - (C_{eq}V_L + C_{eq}V_R) - F_f \tag{2.1}$$

where m_{eq}, C_{eq} and I_{eq} represent the mass, damping ratio of wheel/bearing and center of gravity and moment of inertia of the robot, respectively.

$$I_{eq}\dot{\omega} = (T/2)(F_R - F_L) - (T/2)(C_{eq}V_L + C_{eq}V_R) - M_f \tag{2.2}$$

but

$$V_L = V + (T/2)\omega \tag{2.3}$$

$$V_R = V - (T/2)\omega \tag{2.4}$$

V_L and V_R represent the left and right wheel velocities of the robot, respectively. Now by substituting the Eqs.(2.3) and (2.4) into the (2.1) and (2.2), we get

$$m_{eq}\dot{V} = F_L + F_R - (2C_{eq}V) - F_f \tag{2.5}$$

$$I_{eq}\dot{\omega} = (T/2)(F_R - F_L) - (T/2)C_{eq}(2(T/2)\omega) - M_f \tag{2.6}$$

It is assumed that the speed controller acts as a proportional-derivative(PD)regulator. The speed controller determines the equivalent forces on the wheel using the control law

$$F_L = -K_p(V_L - V_L^d) - K_d\dot{V}_L \tag{2.7}$$

$$F_R = -K_p(V_R - V_R^d) - K_d\dot{V}_R \tag{2.8}$$

V_L^d and V_R^d represent the desired velocity of left and right wheel, respectively and K_p and K_d are the proportional and derivative control gains.

But the V_L^d and V_R^d are commanded by PWM signals U_L and U_R to the PD speed regulator. Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off.

$$V_L^d = a + bU_L \tag{2.9}$$

$$V_R^d = a + bU_R \tag{2.10}$$

Here a and b are constants. Substituting Eqns. (2.3), (2.4), (2.9) and (2.10) into (2.7), (2.8) results in (2.1) and (2.2) yields

$$\dot{V} = \frac{-2(K_p + C_{eq})V}{m + 2K_d} + \frac{2K_p a - F_f}{m + 2K_d} + \frac{K_p(b)}{m + 2K_d}(U_L + U_R) \tag{2.11}$$

$$\dot{\omega} = \frac{-T^2(K_p + C_{eq})\omega}{2I_{eq} + K_d T^2} - \frac{2M_f}{2I_{eq} + K_d T^2} + \frac{K_p bT}{2I_{eq} + K_d T^2}(U_L - U_R) \tag{2.12}$$

Linear and angular acceleration are obtained from the above equations. Using the one step integration, linear velocity and angular velocity of the robot can be found. Using these velocities with respect to the desired velocities we analyze the simulation results.

Equations (2.11) and (2.12) are written in the matrix form

$$\dot{\vec{q}} = \vec{H_1}(V, \omega) + B_1 \vec{u} \tag{2.13}$$

where

$$\dot{\vec{q}} = \begin{bmatrix} \dot{V} \\ \\ \dot{\omega} \end{bmatrix} \tag{2.14}$$

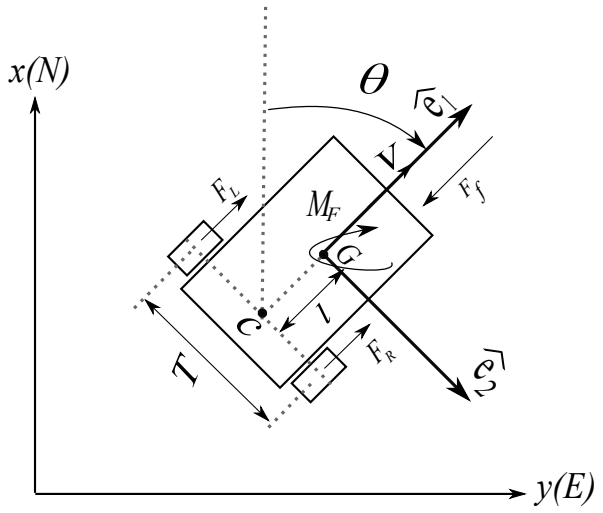


Figure 2.3: Free body diagram

$$\vec{H}_1(V,\omega) = \begin{bmatrix} \frac{-2(K_p+C_{eq})V}{m+2K_d} + \frac{2K_pa-F_f}{m+2K_d} \\ \frac{-T^2(K_p+C_{eq})\omega}{2I_{eq}+K_dT^2} - \frac{2M_f}{2I_{eq}+K_dT^2} \end{bmatrix} \tag{2.15}$$

$$\vec{U} = \begin{bmatrix} U_L \\ U_R \end{bmatrix} \; B_1 = \begin{bmatrix} \frac{k_pb}{m+2k_d} & \frac{k_pb}{m+2k_d} \\ \frac{K_p(b)T}{2I_{eq}+K_dT^2} & \frac{-K_p(b)T}{2I_{eq}+K_dT^2} \end{bmatrix} \tag{2.16}$$

Matrix form generated here used in the next chapter to develop the model based controller.

2.1.2 Equations of Motion when Center of Gravity is not Coinciding with the Center of Axle

Consider in Figure 2.3 and Figure 2.4 is a Hilare type of robot with its body axis as $\hat{e_1}$ and $\hat{e_2}$. Here in the Figure 2.3 it is shown that the center of gravity and

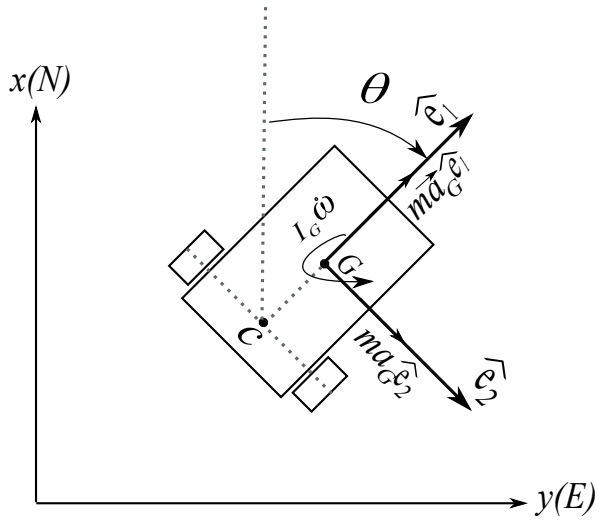


Figure 2.4: Kinetic diagram

center of axle is separated by a small distance l. By balancing the forces from the free body diagram and balancing the moments from the kinetic diagrams, we came up with the equations of motion.

Force/inertia in $\hat{e_1}$ direction is given by

$$m_{eq}a_{GL} = F_L + F_R - (C_{eq}V_L + C_{eq}V_R) - F_f \tag{2.17}$$

From the Figure 2.3, the velocity of the point G can also be written in terms the velocity of the center of axle of the robot($\vec{V_c}$) [11].

$$\vec{V_G} = \vec{V_c} + \vec{\omega} \times \vec{r}_{G/C} \tag{2.18}$$

$$= V_c\hat{e_1} + \omega\hat{k} \times \vec{r}_{G/C} \tag{2.19}$$

$$= V_c\hat{e_1} + \omega l\hat{e_2} \tag{2.20}$$

By differentiating the Eqns.(2.18), (2.19) and (2.20) we get the following:

$$\frac{d\vec{V_G}}{dt} = \dot{V_c}\hat{e_1} + \dot{V_c}\dot{\hat{e}_1} + \dot{\omega}l\hat{e_2} + \omega l\dot{\hat{e}_2} \tag{2.21}$$

$$= \dot{V_c}\hat{e_1} + \dot{V_c}\omega\dot{\hat{e}_2} + \dot{\omega}l\hat{e_2} - \omega^2 l\dot{\hat{e}_1} \tag{2.22}$$

Mathematically acceleration vector at G is written in the component form with respect to $\hat{e_1}$ and $\hat{e_2}$

$$\vec{a_G} = (\dot{V_c} - \omega^2 l)\hat{e_1} + (\dot{\omega}l + \dot{V_c}\omega)\hat{e_2} \tag{2.23}$$

Now by substituting the $\hat{e_1}$ component of (2.23) into Eq.(2.17) we obtain

$$m_{eq}(\dot{V_c} - \omega^2 l) = F_L + F_R - (C_{eq}V_L + C_{eq}V_R) - F_f \tag{2.24}$$

This should be formatted and the resulting moment of inertia in $\hat{k}$ direction is given by

$$I_{eq}\dot{\omega}\hat{k} + \vec{r}_{G/C} \times m\vec{a}_G = ((T/2)(F_L - F_R) - (T/2)(C_{eq}V_L + C_{eq}V_R) - M_f)\hat{k} \tag{2.25}$$

Now by substituting the Eq.(2.23) into Eq.(2.25) and we obtain

$$I_{eq}\dot{\omega} + m(\dot{\omega}l + \dot{V_c}\omega)l = (T/2)(F_L - F_R) - (T/2)(C_{eq}V_L + C_{eq}V_R) - M_f \qquad (2.26)$$

Equations (2.3), (2.4), (2.7) ,(2.8), (2.9) and (2.10) are substituted in Eqs. (2.24) and (2.26)

The terms are rearranged for $\dot{V_c}$and $\dot{\omega_c}$. This yields:

$$\dot{V_c} = \frac{-2(K_p + C_{eq})V_c}{m + 2K_d} + \frac{ml\omega^2 + 2K_p a - F_f}{m + 2K_d} + \frac{K_p(b)}{m + 2K_d}(U_L + U_R) \qquad (2.27)$$

$$\dot{\omega} = \frac{-((K_p + C_e q) - T^2 + mlV_c)W}{I_{eq} + ml^2 + (K_d T^2)/2} - \frac{M_f}{I_{eq} + ml^2 + K_d T^2} + \frac{K_p bT/2}{Ieq + ml^2 + K_d T^2}(U_L - U_R)$$

By writing the above equations in the matrix form we obtain

$$\dot{\vec{q}} = \vec{H}(V, \omega) + B\vec{U} \qquad (2.28)$$

$$\dot{\vec{q}} = \begin{bmatrix} \dot{V} \\ \\ \dot{\omega} \end{bmatrix} \tag{2.29}$$

$$\vec{H}(V,\omega) = \begin{bmatrix} \frac{-((K_p+C_{eq})V_c)}{m+2K_d} + \frac{ml\omega^2+2K_pa-F_f}{m+2K_d} \\ \frac{-T^2+m(l)V_c(K_p+C_{eq})\omega}{2I_{eq}+ml^2+K_dT^2} - \frac{2M_f}{2I_{eq}+ml^2+K_dT^2} \end{bmatrix} \tag{2.30}$$

$$\vec{U} = \begin{bmatrix} U_L \\ \\ U_R \end{bmatrix} \; B = \begin{bmatrix} \frac{k_pb}{m+2k_d} & \frac{k_pb}{m+2k_d} \\ \\ \frac{K_p(b)T/2}{I_{eq}+ml^2+K_dT^2} & \frac{-K_p(b)T/2}{I_{eq}+ml^2+K_dT^2} \end{bmatrix} \tag{2.31}$$

This matrix form of equations of motion will be used in the next chapter. We will discuss how the control methodology leads to the generation of the model based controller.

2.2 Simulations and Discussions

For this simulation we have assumed the l and frictional force to be zero and also kept the derivative control gain to be very small. We have the used the equations when the center of gravity does not coincides with the center of axle. Parameters used for simulation are shown in the following table.

Table 2.1: Table of parameters

paramaters	values used	units
V_c	0.9	m/s
ω	0.5	rad/s
a	3.5285	no units
b	-0.0025	no units
T	0.42	m
K_p	10	no units
K_d	0.01	no units
C_{eq}	0	Newton-s/m
m	13.6078	kg
I_{eq}	0.5443	kgm2
l	0.2	m

Desired velocities are assumed to be constant. These constant velocities are used with the equations (2.9) and (2.10) to determine the PWM signals. The PWM signals were fed to a robot model in SIMULINK. These simulations were run for 100 seconds. Results were analyzed by plotting the desired and robotic velocities with respect to time. We see that the robot's velocity takes some time to follow the desired velocity. It slowly picked up and after a couple of seconds it followed the desired velocity. Even the angular velocity showed similar fashion as linear velocity to follow the desired angular velocity.

Next, the desired velocity is assumed to be as a function of time. Desired velocity followed a sinusoidal sort of motion from the beginning till the end of simulation. As of before the desired velocity and robotic velocity are not overlapping. They maintain a small difference between them at all times. This simulation was also ran for 100 seconds.

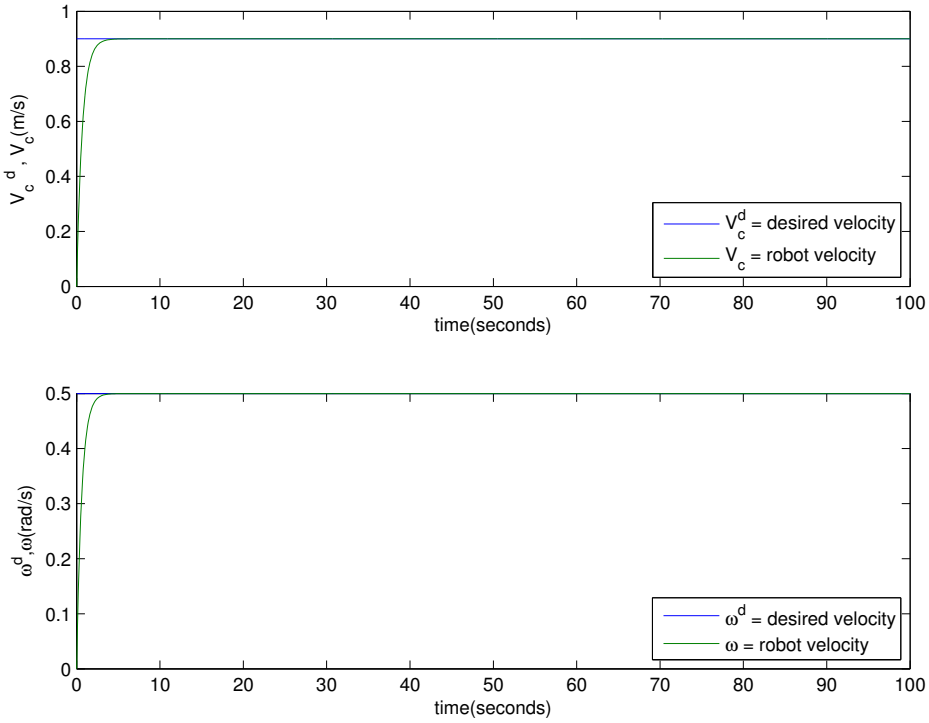


Figure 2.5: Simulation results for desired velocity kept as constant

2.3 Conclusion

Analyzing the Figure 2.5 and Figure 2.6, we can understand that simulation ran with constant desired velocity model response is as expected. However when the velocity is made a function of time desired velocities are not accurately followed. Hence we need a controller to eliminate the errors using the feedback. Thereby making the desired velocity and robot's velocity to overlapped.

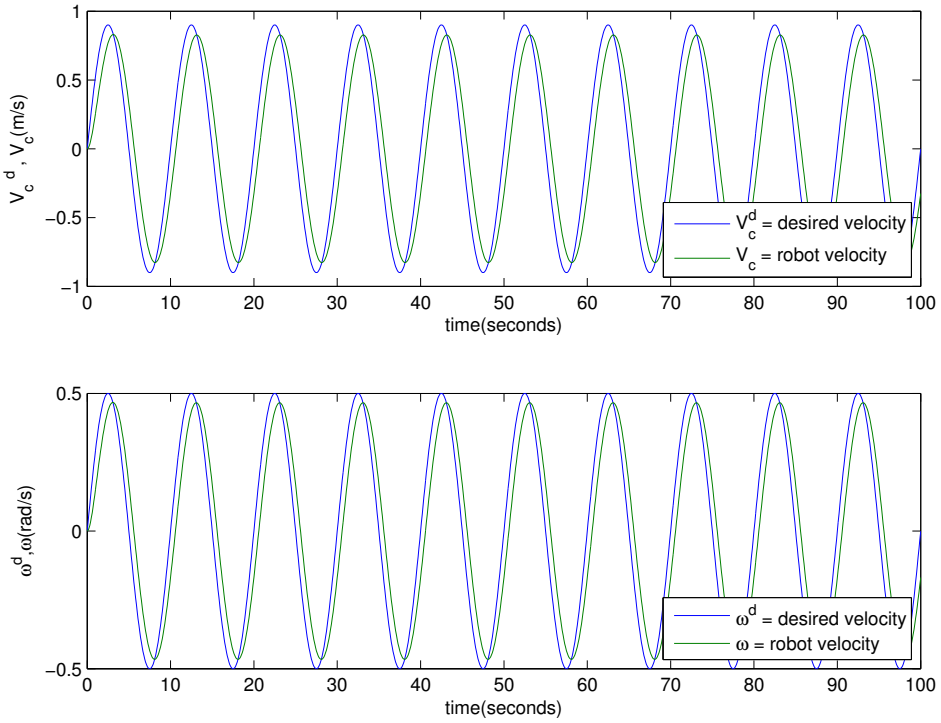


Figure 2.6: Simulation results for desired velocity made as a function of time

CHAPTER 3

CONTROL METHODOLOGY

3.1 Introduction

A controller design using the NN may be simplified by using static NN as system identifier and separate mathematical control law. Here for the first time, a Neural network model and a model based controller are combined. In our approach first a neural network is fitted to the data from the real system. Real-time Control System (RCS) is a reference model architecture, suitable for many software-intensive, real-time control problem domains [12]. RCS is a reference model architecture that defines the types of functions that are required in a real-time intelligent control system, and how these functions are related to each other. RCS is not a system design, nor is it a specification of how to implement specific systems. RCS prescribes a hierar-

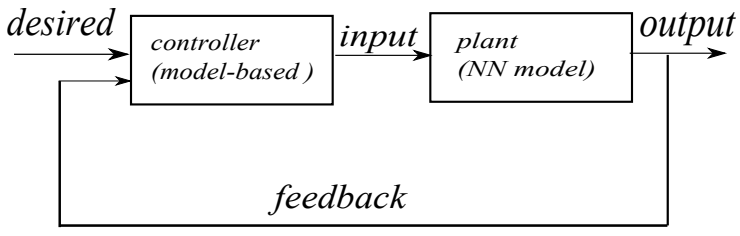


Figure 3.1: Control methodology

chical control model based on a set of well-founded engineering principles to organize system complexity. Also RCS provides a comprehensive methodology for designing, engineering, integrating, and testing control systems. Second, the matrices of the dynamic model of the system are extracted from the neural network model. Third, a model based controller is designed based on the extracted matrices. There might be varied approaches in developing the control methodologies. Following sections highlight how the NN model is generated. It will be shown how the extraction of state space model is achieved. Figure 3.1 represents a block diagram consisting of desired block, model based controller block, plant (neural network model), output, input and feedback.

3.2 Neural Network Model

3.2.1 Introduction to Neural Networks

The multilayered NN is modeled based on the structure of biological nervous systems. Nervous system cell is shown in Figure 3.2. This provides a non-linear mapping from an input space R^n into outer space R^m [1]. Its properties include function approximation, learning, generalization, classification, etc. It is known that the 2-layer NN has sufficient generality for closed loop control purposes. The 2-layer NN shown in Figure 3.3 consists of two layers of weights and thresholds and has L neurons, and the output layer has m neurons.

One may describe the NN mathematically as

$$y = W^T \sigma(V^T x) \tag{3.1}$$

where V is a matrix of first-layer weights and W is a matrix of second-layer weights. The second-layer thresholds are included as the first column of the matrix W^T augmenting the vector activation functions $\sigma(.)$ by 1 in the first position. Similarly, the first-layer thresholds are included as the first column of matrix V^T by augmenting vector x by 1 in the first position.

The main property of NN we are concerned with for control and estimation purposes is the function approximation property [13]. Let $f(x)$ be a smooth function from $R^n \to R^m$. Then it can be shown that if the activation functions are suitably selected and x is restricted to a compact set $S \in R^n$, then for some sufficiently large number L of hidden-layer neurons, there exist weights and thresholds [14] such one has

$$f(x) = W^T \sigma(V^T x) + \epsilon(x) \tag{3.2}$$

with $\epsilon(x)$ suitably small. $\epsilon(x)$ is called the neural network functional approximation error. In fact, for any choice of a positive number ϵ_n, one can find a neural network of large enough size L such that $\epsilon(x) \leq \epsilon_n$ for all $x \in S$. Finding a suitable NN for approximation involves adjusting the parameters V and W to obtain a good fit to $f(x)$. Note that tuning of the weights includes tuning of the thresholds as well. The neural net is nonlinear in the parameters V, which makes adjustment of these parameters difficult and was initially one of the major hurdles to be overcome in closed-loop feedback control applications [15]. If the first layer weights V are fixed,

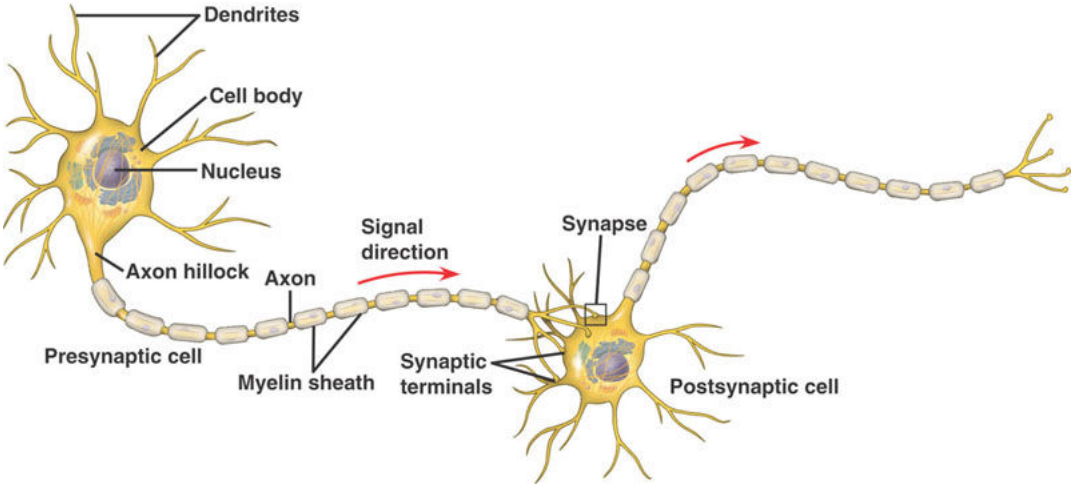


Figure 3.2: Nervous system cell

then the NN is linear in the adjustable parameters $W(LIP)$. It has been shown that, if the first-layer weights V are suitably fixed, then the approximation property can be satisfied by selecting only the output weights W for good approximation. For this to occur, $\sigma(V^T x)$ must provide a basis. It is not always straightforward to pick a basis $\sigma(V^T x)$. It has been shown that cerebellar model articulation controller (CMAC) [16], radial basis function (RBF) [17], fuzzy logic [18], and other structured NN approaches allow one to choose a basis by suitably partitioning the compact set S. However, this can be tedious. If one selects the activation functions suitably, e.g. as sigmoids, then it was shown in [19] that $\sigma(V^T x)$ is almost always a basis if V is selected randomly.

3.2.2 Generation of NN model

Generated output data is used for fitting the NN model i.e NARX time series prediction. For fitting the data into the NN model both the input and output vectors

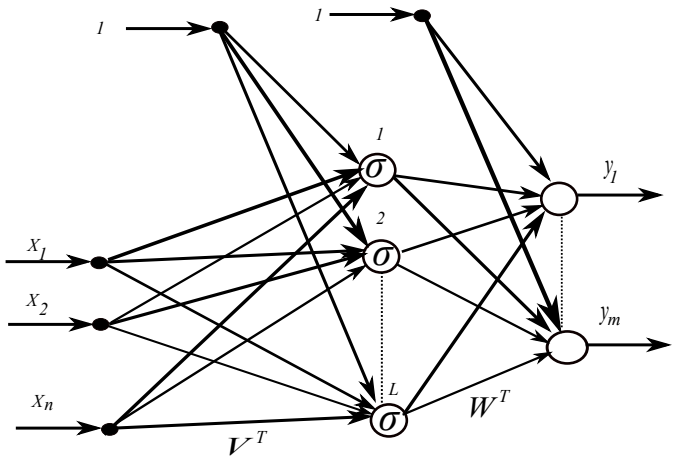


Figure 3.3: Two layer neural network

should be of same dimension. As discussed in the introduction of neural networks workflow has to be followed in order to generate the NN model. We are using the NARX type of network in order to generate the neural network model [20]. In this type of network, we use the input signal and output target from outside the neural network toolbox software. We use simulink model to obtain the input and target output to the network .i.e PWM and velocity vectors respectively. PWM is a pulse wave modulation signal which is a function of desired velocity and desired velocity is a function of time. PWM signals are fed to the robot there by to the PD speed regulator to generate the output velocity. Designed simulink model generally consists of desired block, input signal block, robot (plant), feedback block, controller, output block. We run the simulink model and the output is saved in the workspace. We simulate the model for about 100 sec with the sample time of 0.01 sec, there by we generate the output data.

Neural network tool box of Matlab has been used. After the data is collected, feedforward network has to be created. Next configuration, initialization, network analysis and validation has to be done. Then use the network if it is accurate go for it otherwise reinitialize the weights and biases. After the workflow is completed, a Matlab code is generated. Finally, the code is executed.

We generate three different type of simulink models .i.e open loop, close loop and predict one step ahead. Here we observe the performances of each model on the command window and we check that these performances should be as low as possible in order to have the accurate NN model. By using some of the essential commands as gensim we can generate the NN model. The generated NN model helps in the extraction of state space model.

3.3 Extraction of State Space Model from NN

In this section we discuss on how the state space model is extracted and also the way we present the approach [21].

Equation (2.31) has the following general form.

$$\begin{bmatrix} \dot{V}_c \\ \dot{\omega} \end{bmatrix} = H(V_c, \omega) + \begin{bmatrix} \frac{k_p b}{m+2k_d} & \frac{k_p b}{m+2k_d} \\ \frac{K_p(b)T/2}{I_{eq}+ml^2+K_dT^2} & \frac{-K_p(b)T/2}{I_{eq}+ml^2+K_dT^2} \end{bmatrix} \begin{bmatrix} U_L \\ U_R \end{bmatrix} \tag{3.3}$$

The above matrix form is expressed as

$$\dot{\vec{q}} = H(\vec{q}) + B\vec{U} \tag{3.4}$$

The discretized model is :

$$\vec{q}(t+\delta t) = \vec{q}(t) + \dot{\vec{q}}(t)\delta t \tag{3.5}$$

$$= \vec{q}(t) + [H(\vec{q}(t)) + B\vec{U}(t)]\delta t \tag{3.6}$$

$$= \vec{q}(t) + H(\vec{q}(t))\delta t + (B\delta t)\vec{U}(t) \tag{3.7}$$

$$\vec{q}(t+\delta t) = H_D(\vec{q}(t)) + B_D\vec{U}(t) \tag{3.8}$$

$$\vec{q}(t+\delta t) = N_D(\vec{q}(t), \vec{U}(t)) \tag{3.9}$$

Note that a neural network time series that is fit to the robot represent Eq. (3.9).

However, the individual terms H_D and B_D are not available. Here, the procedure will be proposed that can be used to calculate them using the NN. Note that N_D is obtained from the neural network that is fit to the data.

The NN model and the state space model must be equivalent so one can write:

$$H_D(\vec{q}(t)) + B_D(\vec{U}(t)) = N_D(\vec{q}(t), \vec{U}(t)) \tag{3.10}$$

We can write the following equations only because the approach in obtaining B matrix is initialized.

$$\vec{q}(t+\delta t)_1 = H(\vec{q}(t)) + B(\vec{q}(t))\vec{U}(t) \tag{3.11}$$

$$\vec{q}(t+\delta t)_2 = H(\vec{q}(t)) + B(\vec{q}(t))(\vec{U} + \delta\vec{U}) \tag{3.12}$$

Equation (3.12) is subtracted from Eq. (3.11) :

$$\vec{q}(t+\delta t)_2 - \vec{q}(t+\delta t)_1 = B(\vec{q}(t))\delta\vec{U} \tag{3.13}$$

To isolate the first column of B matrix,the following is assumed.

$$\delta\vec{U} = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \tag{3.14}$$

Equation (3.13) becomes

$$\vec{q}(t+\delta t)_2 - \vec{q}(t+\delta t)_1 = B(\vec{q}(t)) \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \tag{3.15}$$

$$\tag{3.16}$$

Right hand side of Eq. (3.15) can be calculated using the neural network model:

$$N(\vec{q}(t), U + \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}) - N(\vec{q}(t), U) = B(\vec{q}(t)) \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \tag{3.17}$$

But the right hand side of Eq. (3.17) represents the first column of B. So

$$\begin{bmatrix} B_{11} \\ \\ B_{12} \end{bmatrix} = N(\vec{q}(t), U + \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix}) - N(\vec{q}(t), U) \tag{3.18}$$

Similarly

$$\begin{bmatrix} B_{21} \\ \\ B_{22} \end{bmatrix} = N(\vec{q}(t), \vec{U} + \begin{bmatrix} 0 \\ \\ 1 \end{bmatrix}) - N(\vec{q}(t), \vec{U}) \tag{3.19}$$

Now that B is calculated H can be found from Eq. (3.11)

$$H(\vec{q}(t)) = N(\vec{q}(t), \vec{U}) - B\vec{U} \tag{3.20}$$

3.4 Conclusion

Knowing the $H(\vec{q})$ and B, we can design a model-based controller. As we see from the open loop we are just using PD speed regulator and actual does not follow the desired. For improving this we need a closed-loop controller. This closed-loop controller uses the sliding mode control law. Now this control law makes the error to reach the surfaces and thereby slides over the surface to make it zero. Hence, with the use of closed-loop controller we can eliminate the errors and thereby makes the actual to follow the desired. In the next chapter, the derivation of controller is discussed.

CHAPTER 4

CONTROL DESIGN

4.1 Introduction

In this chapter, a controller is developed based on the equations (2.1.2) and (2.31) that calculate H and B matrices from the previous chapter. Design of the closed-loop control is proposed to improve the limitations (actual not following the desired), we faced by using the open-loop control. The model-based controller is tested in two different stages; once via simulations and once via experimentation. This chapter mainly focus on the proof of concept of the proposed approach. First, we use a simulated robot as if it were a real robot and collect numerical data. The collected numerical data from the simulation is used to derive a neural network model. Next, we use a real robot to collect data for the derivation of a neural network model.

4.2 Proof of Concept

A proof of concept is a realization of a certain method or idea to demonstrate its feasibility. Here, we are using the robot model as if it were a real robot to explain the idea. We try to simulate the model as a trial to see that the approach works in simulation. This simulation is set to run for 100 seconds. Now, we collect the data

and use them to generate the NN model. Later, NN model was generated. We overlap the system response to NN response. Figure 4.1 and Figure 4.2 illustrates the result of overlapping of analytical system response from Eq. (2.12) with the NN model.

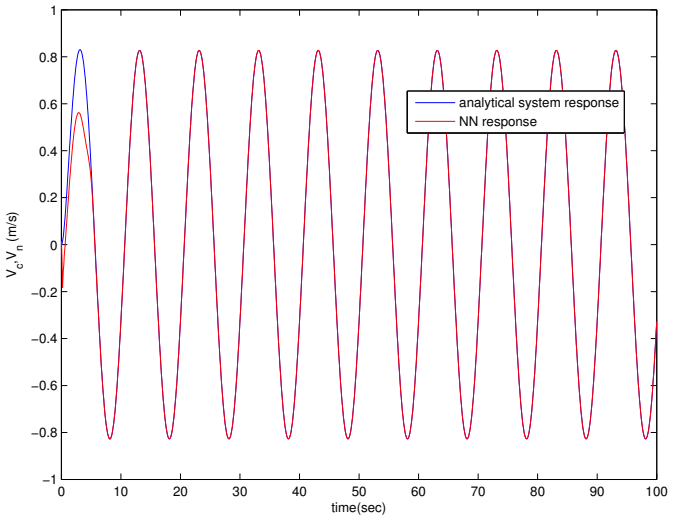


Figure 4.1: Linear velocity overlap of analytical system vs NN model

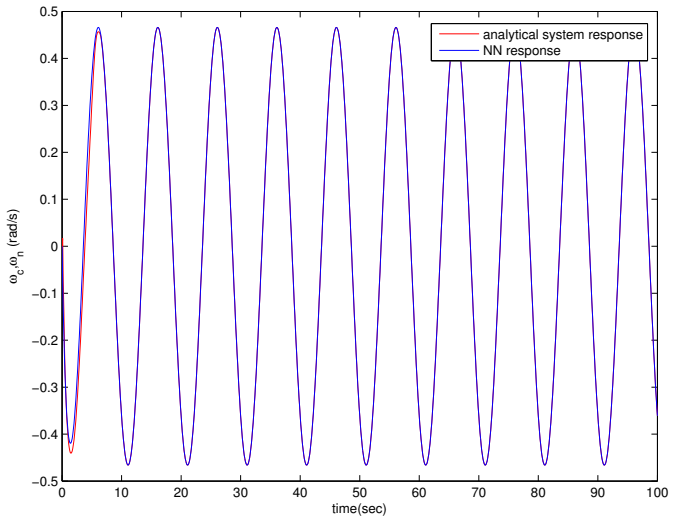


Figure 4.2: Angular velocity overlap of analytical system vs NN model

Now the generated NN of robotic model is used for the extraction of model of H_D and B_D matrices that are used with the model-based controller.

4.2.1 Design of a Model-Based Controller

The equations that calculate H and B matrices from the previous chapter help in the design of model-based controller. The idea behind the design came from the approach of control methodology. This calculation approach extracted the state space model and there by the design of model-based controller [22].

$$N(\vec{q}(t), \vec{U}(t)) = H_D(\vec{q}(t)) + B_D\vec{U} \tag{4.1}$$

This standardized Eq. (4.1) led to the design of the model based controller. Here, from the equation we can find the H_D and B_D matrix. The H_D and B_D matrices are found with the help of generated NN model.

The dynamics of the system under consideration may be described using the nonlinear state space representation introduced in section 3.3. The future state is given by the equation

$$\vec{q}(t + \delta t) = H_D(\vec{q}(t)) + B_D\vec{U}(t) + \vec{f}(\vec{q}(t)) \tag{4.2}$$

where $\vec{q}$ is the state vector, t denotes the time step, H_D is the state transition vector, B_D is the input matrix, $\vec{U}$ is the control input vector, and $\vec{f}(\vec{q}(t))$ is an unsystematic disturbance not accounted for in the dynamics of the robot. We assume an error function $\vec{\sigma}$ described as a sliding surface.

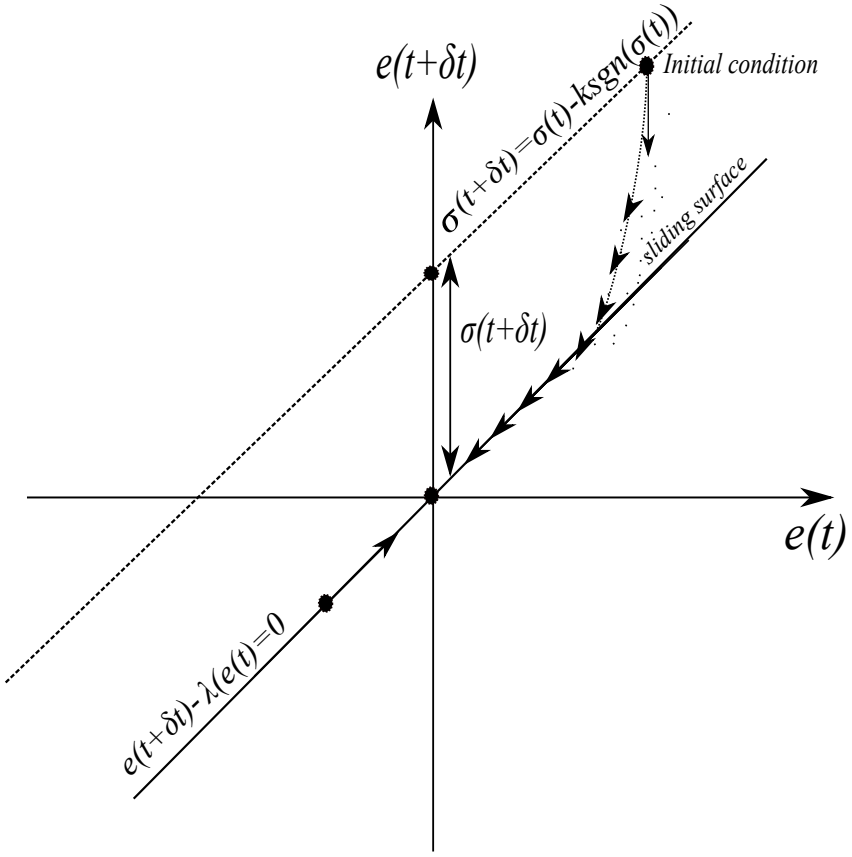


Figure 4.3: Sliding surfaces

$$\vec{\sigma}(t+\delta t) = \vec{q}(t+\delta t) - \vec{q}^{d}(t+\delta t) - \lambda(\vec{q}(t) - \vec{q}^{d}(t)) = \vec{\sigma}(t) - K.\mathrm{sgn}(\vec{\sigma}(t)) \qquad (4.3)$$

or

$$\vec{\sigma}(t+\delta t) = \vec{e}(t+\delta t) - \lambda\vec{e}(t) = \vec{\sigma}(t) - K.\mathrm{sgn}(\vec{\sigma}(t)) \qquad (4.4)$$

where

$$\vec{e}(t) = \vec{q}(t + \delta t) - \vec{q}^{d}(t + \delta t) \tag{4.5}$$

where d denotes a desired value, and λ and K are diagonal matrices. According to Figure 4.3 the term $\vec{\sigma}(t) - K.\text{sgn}(\vec{\sigma}(t))$, causes $\vec{\sigma}(t)$ to vanish. At that point $\vec{\sigma}(t + \delta t) = \vec{e}(t + \delta t) - \lambda\vec{e}(t) = 0$, causes $\vec{e}(t)$ to vanish.

$$0 < \lambda_{11}, \lambda_{22} < 1 \tag{4.6}$$

$$0 < K_{11}, K_{22} < 1 \tag{4.7}$$

By choosing the gain matrices λ and K carefully, the controller will dictate the behavior that drives the system to the sliding surface, and then slides along it until the error is zero. The matrix λ is indicative of how fast the system will slide along the surface after reaching it. The matrix K governs how quickly the system will reach the surface given an initial error or unsystematic disturbance. Following section examines the stability of the system in terms of gains.

By substituting Eq. (4.2) into Eq. (4.3), we are able to incorporate the state transition vector and input matrix into the equation and remove the next future vector $\vec{q}(t + \delta t)$. Neglecting the disturbance,

$$\vec{f}(\vec{q}(t)) = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix} \tag{4.8}$$

.

$$\vec{q}^{d}(t+\delta t)+\lambda(\vec{q}(t)-\vec{q}^{d}(t))+\vec{\sigma}(t)-K.sgn(\vec{\sigma}(t))=H_D(\vec{q}(t))+B_D\vec{U}(t) \qquad (4.9)$$

Rearranging algebraically, we arrive at an expression for the control input vector $\vec{U}(t)$.

$$\vec{U}(t)=B_D^{-}1[-H_D+\vec{q}^{d}(t+\delta t)+\lambda(\vec{q}(t)-\vec{q}^{d}(t))+\vec{\sigma}(t)-K.sgn(\vec{\sigma}(t))] \qquad (4.10)$$

This is the sliding mode control law.

4.2.2 Stability Analysis

Substitute Eq. (4.10) into Eq. (4.2).

$$\vec{q}(t+\delta t)=H_D+B_DB_D^{-}1[-H_D+\vec{q}^{d}(t+\delta t)+\lambda(\vec{q}(t)-\vec{q}^{d}(t))+\vec{\sigma}(t)-K.sgn(\vec{\sigma}(t))]+\vec{f}(\vec{q}(t)) \qquad (4.11)$$

$$\vec{q}(t+\delta t)=[\vec{q}^{d}(t+\delta t)+\lambda(\vec{q}(t)-\vec{q}^{d}(t))+\vec{\sigma}(t)-K.sgn(\vec{\sigma}(t))]+\vec{f}(\vec{q}(t)) \qquad (4.12)$$

It is desired that the control law drive the error to zero by ensuring that the error of the next time step is less than the current error.

$$|\vec{\sigma}(t+\delta t)| < |\sigma(\vec{t})| \tag{4.13}$$

$$|\vec{q}(t+\delta t) - \vec{q}^{d}(t+\delta t)| - |\lambda(\vec{q}(t) - \vec{q}^{d}(t))| < |\vec{\sigma}(t)| \tag{4.14}$$

$$|\vec{\sigma}(t) - K.sgn(\vec{\sigma}(t))] + \vec{f}| < |\vec{\sigma}(t)| \tag{4.15}$$

To demonstrate the stability of the control law, we consider the cases where $\sigma_i(t) > 0$ and $\sigma_i(t) < 0$. For the control law to drive the error to zero,

$$|\sigma_i(t) - K_i.sgn(\sigma_i(t))] + f_i| < |\sigma_i(t)| \tag{4.16}$$

where i denotes an element of the vector. If $\sigma_i(t) > 0$:

$$\sigma_i(t) - K_i + f_i < \sigma_i(t) \tag{4.17}$$

$\implies K_i > |f_i|$

$$K_i = F_i + \eta_i \tag{4.18}$$

where $F_i \geq |f_i|$ and $\eta_i > 0$. If $\sigma_i(t) < 0$:

$$\sigma_i(t) - K_i + f_i > \sigma_i(t) \tag{4.19}$$

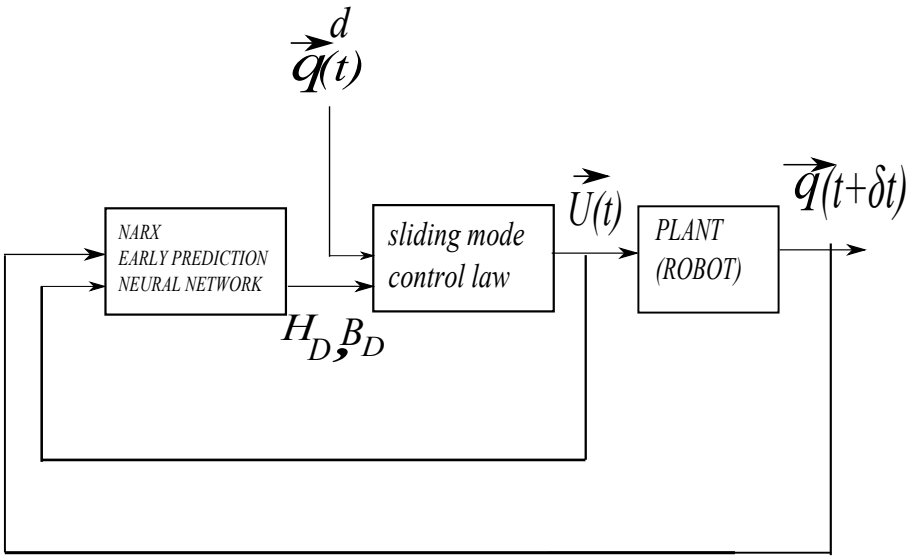


Figure 4.4: A block diagram of the feedback control law with the embedded neural network

$$\implies K_i < |f_i|$$

$$K_i = F_i + \eta_i \tag{4.20}$$

Thus, if K_i is chosen such that the condition in Eq. (4.18) (or Eq. (4.20) is met, then $\sigma_i(t)$ approaches zero as $t \to \infty$ despite the existence of the bounded disturbance or model mismatch included in $\vec{f}(\vec{q}(t))$. After $\vec{\sigma}(t)$ vanishes, $\vec{q}(t)$ approaches $\vec{q}^d(t)$ as $t \to \infty$, as suggested by equation (4.3).

4.2.3 Simulation Results for Closed-Loop Control

In order to eliminate the errors and have the feedback attached to the simulation model, we proposed closed-loop control. Figure 4.4 illustrates the block diagram of the feedback control law. As an extension to the model based controller we went

with the design of closed-loop control. Control gains used for the simulation of closed-loop control is being illustrated in the equations below.

Here, the simulation ran for 50 seconds and after that we plot the results for the desired versus robot velocities. We can even run for 100 seconds as of before simulations. Following equations represents the control gains used for the closed-loop control simulation.

$$\phi = \begin{bmatrix} 0.1 \\ \\ 0.28 \end{bmatrix} \tag{4.21}$$

$$\lambda = \begin{bmatrix} 0.5 & 0 \\ \\ 0 & 0.5 \end{bmatrix} \tag{4.22}$$

$$K = \begin{bmatrix} 0.1 + E_1 & 0 \\ \\ 0 & 0.1 + E_2 \end{bmatrix} \tag{4.23}$$

$$\tag{4.24}$$

Error mentioned in Eq. (4.23) is calculated as follows

$$\vec{E} = |(H(\vec{q}(t)\vec{U}(t) - N(\vec{q}(t), \vec{U}(t)))| \tag{4.25}$$

From Eq. (4.1) we get N matrix.

From Figure 4.5, we observed that results of the closed-loop control matched with the desired velocities. Figure 4.7 plots the error between the robot and desired

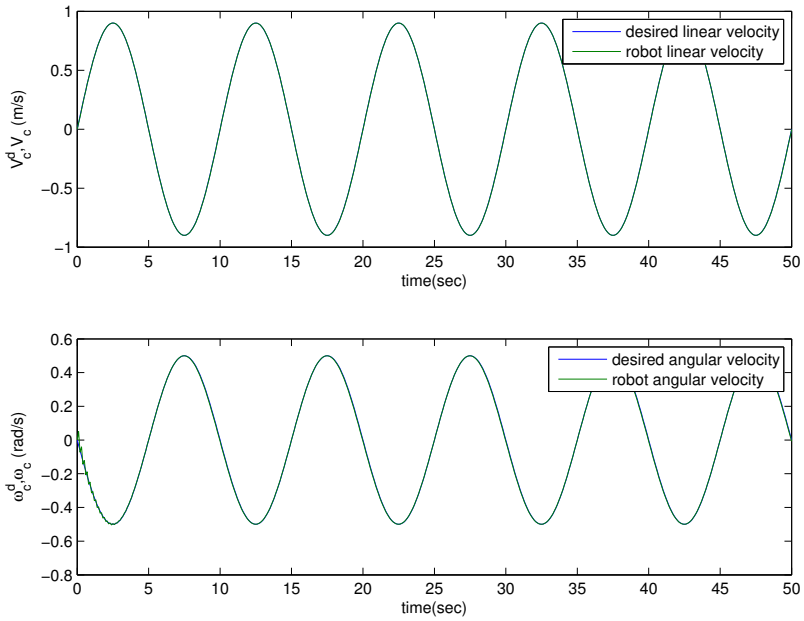


Figure 4.5: Closed loop control simulation results

velocities.From, this we can also come to the conclusion that even proof of concept worked. Refer to Figure 4.7 for error. The max, min error in velocity is $0.001, -0.008$ and max, min error in angular velocity is $0.05, -0.02$. This further concludes that we can go one step forward and test the real robot and can obtain the similar results as of before.

4.3 Experimentation

4.3.1 Description of Hilare-type Robot

In general, a Hilare-type robot has two wheels, left and right, driven by two independent motors. If the velocity of both the left and the right wheels are same, then the robot moves in a straight path. By increasing the right wheel velocity when

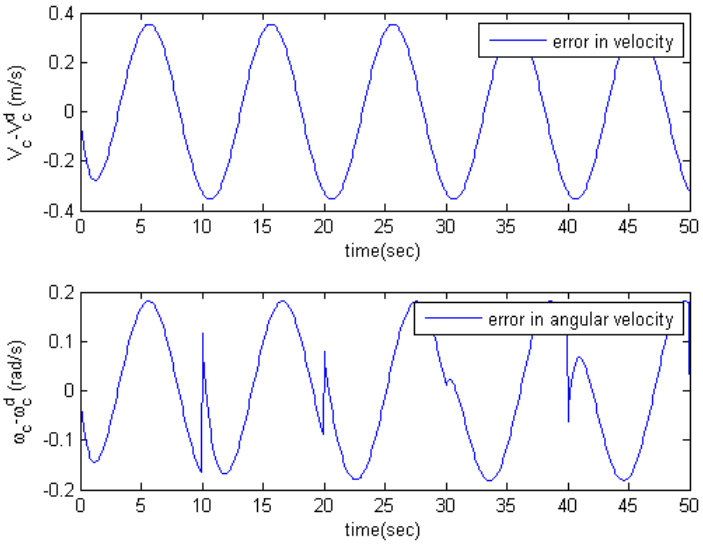


Figure 4.6: Error in open-loop

compared to the left wheel velocity, the robot makes a left turn. In contrast, by increasing the left wheel velocity when compared to the right wheel velocity, the robot makes the right turn. The Hilare-type robot have an unique ability of making a zero radius turn. In other words, if the robot is provided with the equal left and right wheel velocities but in opposite direction, then the robot turns around with zero radius.

Figure 4.8 and Figure 4.9 represents the front view, the top view of the Hilare-type robot respectively. The rectangular box on top of the robot consists of a navigational sensor, computer, servo controller safety switch and the modem. These components are used to control the robot in real-time.

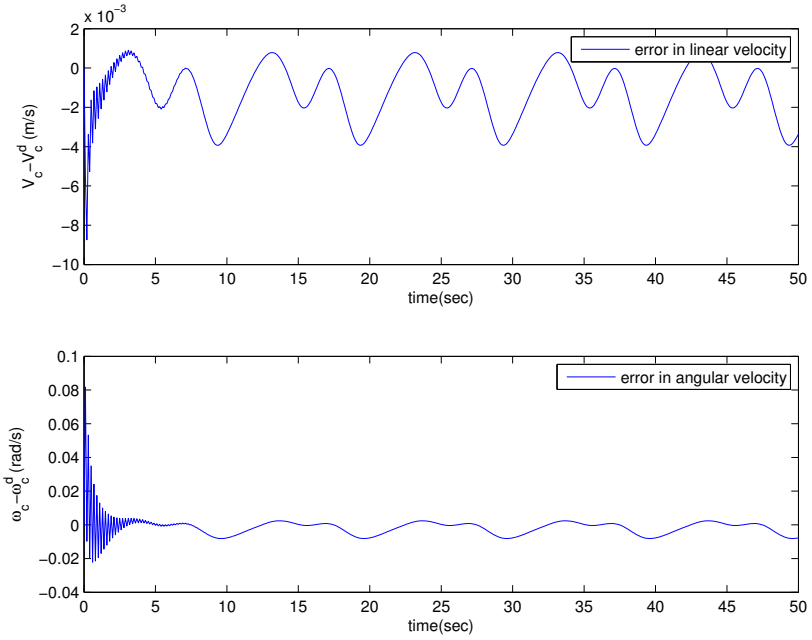


Figure 4.7: Error in closed-loop

4.3.2 Components Used for Experimentation

A research Hilare-type robot is purchased and the left and the right wheels of the robot are set to operate independently. This is because the control inputs given to the robot from the controller are the left and the right wheel velocities. In general, the Hilare-type robots that are available in the market are controlled by using two joysticks on the transmitter. One joystick is used to control the linear velocity of the robot and another is used to control the angular velocity of the robot. So, the transmitter is also need to be programmed, such that, one of the joysticks can be used to control the left wheel of the robot and another joystick can be used to control the right wheel of the robot. Moreover, the transmitter can be used to select either the manual mode or the automatic mode, by carefully programming the transmitter. This

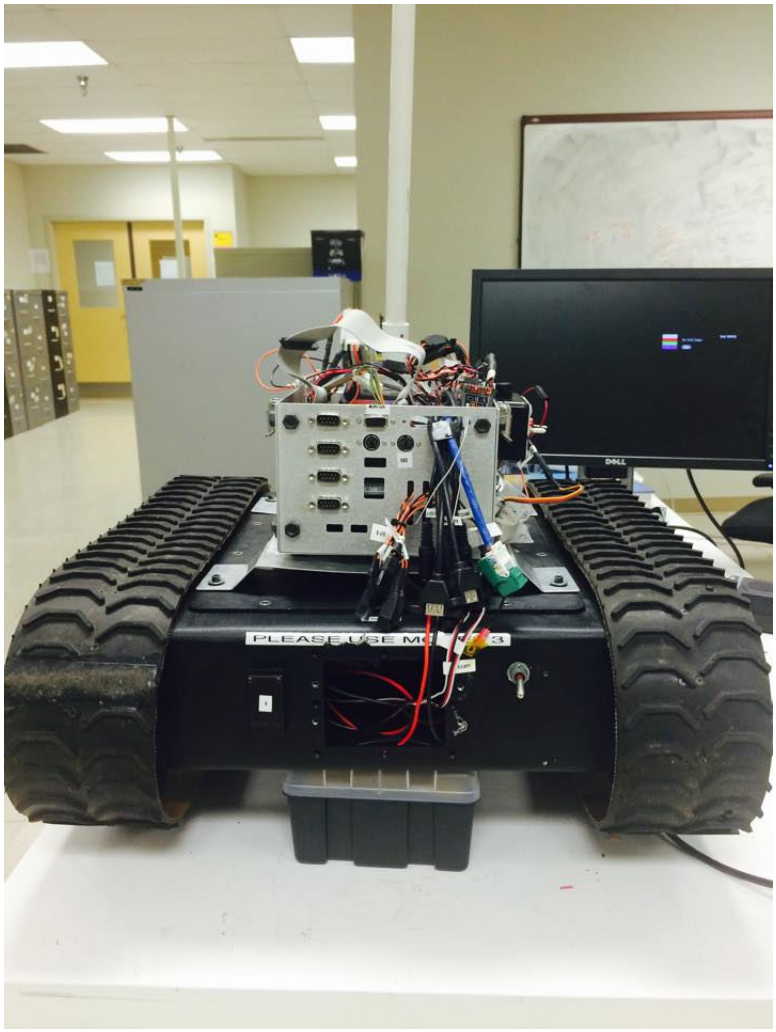


Figure 4.8: Front view of Hilare-type robot

can be achieved if the transmitter has a two level switch, which could be programmed to work as a mode selector. In the manual mode, the user can give the inputs to the robot using the transmitter, whereas, in the automatic mode, the robot follows the predefined user maneuvers. The detailed description of this mechanism is given as follows.

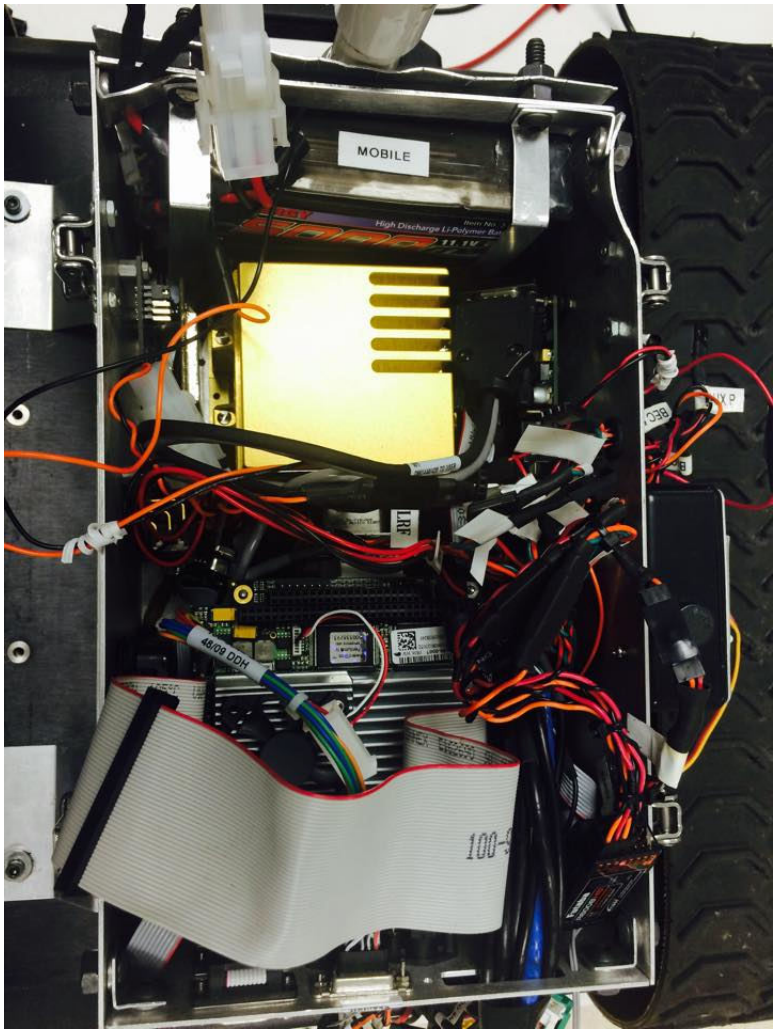


Figure 4.9: Top view of Hilare-type robot

Figure 4.10 shows the signal flow diagram for the Hilare-type robot. TSR represents the transmitter, REC represents the receiver, SSC represents the servo switch controller, PC 104 represents the computer and NAV represents the navigational sensor. The manual or the automatic mode can be selected by using the transmitter. The dotted line indicates the signal flow in the manual mode, where as, the semi dotted line indicates the signal flow in the automatic mode. In the manual mode, the

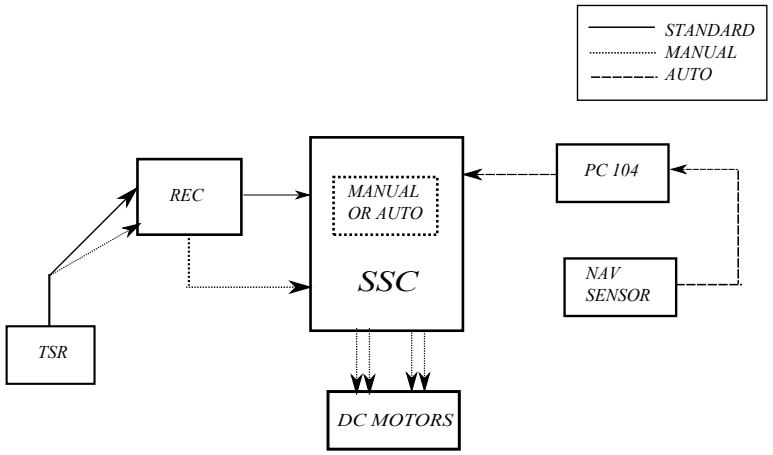


Figure 4.10: Signal flow diagram for the Hilare-type robot

servo switch controller takes the control inputs from the transmitter via receiver and gives to the DC motors of the robot. In other words, the control inputs to the robot are given by the user in real-time using transmitter.

In the automatic mode, the servo switch controller takes the control inputs from the controller that is already embedded in the computer PC104. The robot will then track the predefined user maneuvers. While tracking, the controller consider the real-time values such as position of the robot, Euler angels, longitudinal and lateral velocities of the robot etc., using navigational sensor to trace the desired trajectory as perfect as possible. But of course, the accuracy of trajectory tracking depends on the performance of the controller [23].

4.3.3 Procedure Followed for Collecting Data

After the trial is successful we performed the test on real robot. First, we need to collect velocity and angular velocity data from the robot. The data is used to fit a neural network model, which represents the behavior of the actual robot.

To collect the data, a program is written using Matlab in the desktop computer. Then, the program is compiled for the robot's hardware using the X-PC-target tool box of Matlab. Later, we send the compiled program to the robot's computer using an Ethernet connection. For this, robot computer is communicated to the desktop computer with the help of LAN network. Now, we used the X-PC-target commands to run the robot. With these commands, we could initiate the robot. The robot executes the written program [24].

$$V_c^d = V_0 \sin(2\pi(t)/\tau) \tag{4.26}$$

$$\omega_c^d = \omega_0 \sin(2\pi(t - t_{shift})/\tau) \tag{4.27}$$

where V_c^d and ω_c^d are desired linear and angular velocities of the robot. V_0 and ω_0 are the constant linear and angular velocities and t, τ represents, time of simulation and a constant value respectively. Where as t_{shift} represents, the time shift. This changes for some stipulated interval of time depending on the user requirements. Table 4.1 illustrates about the constants used for calculating the desired velocities of the robot.

Table 4.1: Constants used in calculating desired velocities of the robot

V_0	0.9
ω_0	0.5
τ	10
t_{shift}	0 to 9

We tested the robot for various time-shifts and collected the data. The time shifts are programmed in such a way that for every 10 seconds of simulation it transfers to next time shift. This continues until the end of simulation. As, the total simulation time is set to 100 seconds, so it starts with 0 and ends with 9. Similar procedure is adopted as of the robot model and generated NN model with the experimental data. We have prepared a table with the error in velocity, angular velocity for different hidden layers.

Velocity difference is calculated as

$$\Delta V_i = V_{ci} - V_{ni} \tag{4.28}$$

Here V_{ci} and V_{ni} are robot's model velocity and NN model velocity, respectively at a data point i. Sum of the square of the velocity difference is calculated as

$$V_{sum}^2 = \sum_{i=1}^{n} (\Delta V)^2 \tag{4.29}$$

where n is the total number of data points. The average error in velocity is calculated as

$$e_v = \sqrt{V_{sum}^2}/n \tag{4.30}$$

The same procedure as the velocity should be followed to obtain the error in the angular velocity difference is calculated as,

$$\Delta\omega_i = \omega_{ci} - \omega_{ni} \tag{4.31}$$

Here ω_{ci} and ω_{ni} are robot's model velocity and NN model velocity, respectively at a data point i. Sum of the square of the angular velocity difference is calculated as

$$\omega_{sum}^2 = \sum_{i=1}^{n} (\Delta\omega)^2 \tag{4.32}$$

where n is the total number of data points. The average error in velocity is calculated as

$$e_\omega = \sqrt{\omega_{sum}^2}/n \tag{4.33}$$

where L represents the hidden layers from 5, 10, 15 and 20.

Table 4.2: Error in the velocity (e_v)

$L5$	$L10$	$L15$	$L20$
3.389E-05	1.5276E-05	1.8177E-05	1.6822E-05

Table 4.3: Error in the angular velocity (e_ω)

$L5$	$L10$	$L15$	$L20$
6.1319E-05	5.4648E-05	5.7599E-05	5.5752E-05

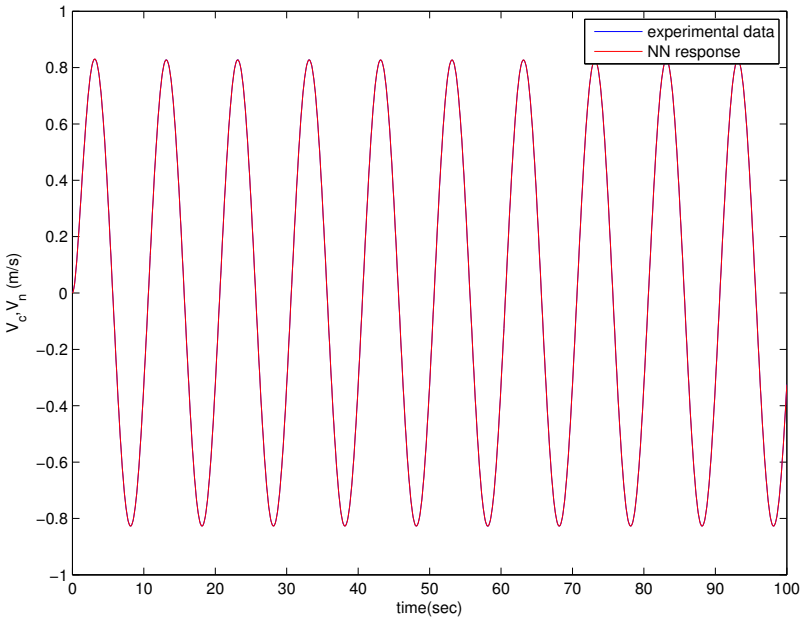


Figure 4.11: System response vs NN response

By analyzing Table 4.2 and Table 4.3 and taking the least error from all the hidden layers (L10) amongst all the other layers seems to be best fit. Hence from Table 4.3, $L10$ can be used further for the experimentation in order to the design the model based controller.

From Figure 4.11 and Figure 4.12, we can tell that the result of fitting the neural network to the actual data is satisfactory, because the NN response overlaps with the data collected from the experiments.Figure 4.12 indicates that there is small

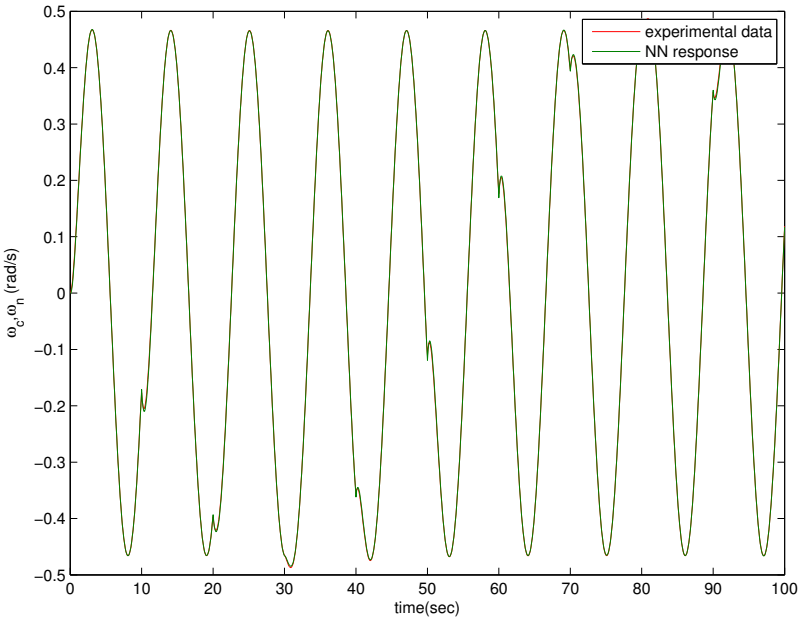


Figure 4.12: System response vs NN response

turn, this is mainly because of the desired angular velocity as it is a function of time-shift. A function in time that is time-shifted by a certain constant time period T. For convenience, we will denote this function as x(t - T) is known as a time-shift. As a desired velocity is not a function of time-shift, we do not see any turns and it is smooth throughout the simulation.

Using the NN model now we can find the $H_D(\vec{q})$ and B_D in real time.

4.3.4 Simulation Results for Closed-Loop Controller

As of before with the design of model-based controller we proposed the closed-loop control. When we plot the results for PWM versus velocity, we get the following result.

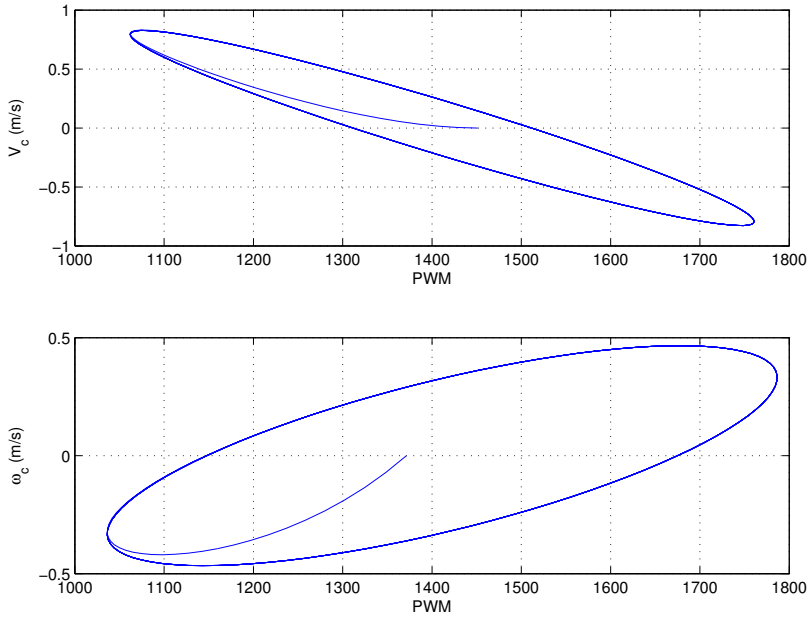


Figure 4.13: PWM vs velocity for simulation model

From Figure 4.13, we can observe a closed loop for both linear and angular velocity plots. This explains that there is no dead-zone arising between the axes. This result was achieved when we ran the simulation for robot model. Plot should be always a closed ellipse and when we try to draw a line from one end of ellipse to other on major axis, it should be a straight line. This implies model is linear. This proves that we can use the standardized form of Eq. (4.1) in which the equations are linear with respect to the PWM inputs. Experimental results are illustrated through Figure 4.14.

There are some differences between the modeled and actual robot. For modeled robot there occurs no slip when turning. The reason is analytical robot derived assumed has just two wheels. But the actual robot has a track on it thereby slips

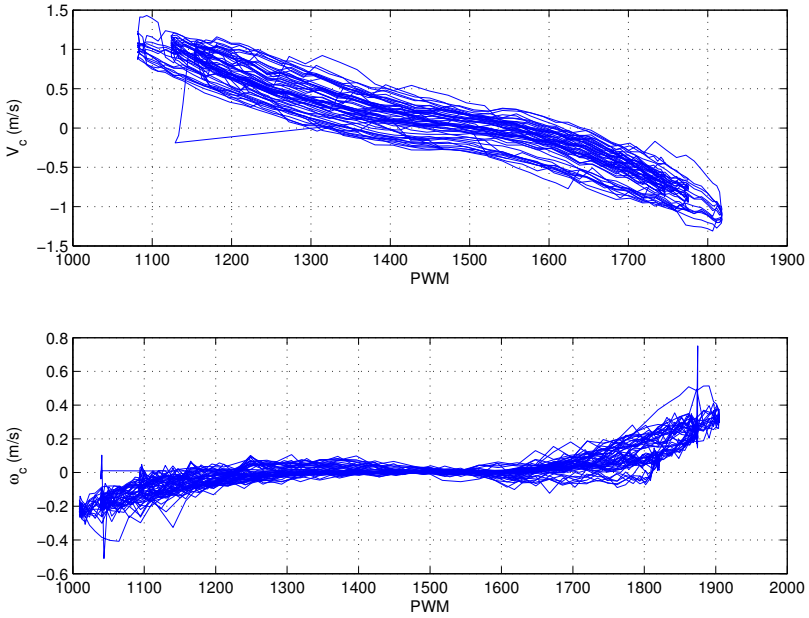


Figure 4.14: PWM vs velocity for real robot

especially when turning. The experimental results obtained are not as accurate as we had before in the simulation model. In order to find out the reason, we plotted the velocity and the angular velocity of the robot versus PWM inputs. When we closely observe at the results it seems that the plot has a dead zone in between the axis. This dead zone is not present when we observe the simulation model results. From this we can conclude that, the proposed standardized Eq. (4.1) as of the robotic model does not represent the actual robot in low speeds.

$$N(\vec{q}(t), \vec{U}(t)) = H_D(\vec{q}(t)) + B_D(\vec{U})\vec{U} \tag{4.34}$$

Equation (4.34) might be more complicated than the previous model [25]. Hence our approach does not apply to the real robot. In future, students need to come up with a new model in order to overcome this limitation.

4.4 Conclusion

It concludes that proof of concept works perfectly well for the simulation model in turn leads us to simulate the real robot. Designing of closed-loop control helped us to eliminate the errors and provided feedback to the simulation model. Simulation of the robot worked well using the model-based controller. However, since the model of the actual robot does not conform to the form assumed in Eq. (4.1) the proposed controller is not suitable for the actual robot. Hence, future work be coming up with the new model which more exactly represents the actual robot.

CHAPTER 5

CONCLUSION

An analytical dynamical model is derived with the physical parameters of robot taken into consideration. The kinematic and dynamic equations of motion are derived and implemented in a state space model. The model is used as a reference to compare with the predictions of the artificial neural network and as the plant in simulation control law. The formulation of an artificial neural network dynamic model is estimated. Finally, neural network is implemented in a closed loop control law for the virtual real robot and simulated in software. The control law is used and extracted state space information from the neural network to produce a control input. The Matlab/Simulink software is used to simulate the motion of the robot and demonstrated the effectiveness of the control method.

By analyzing the simulation results, constant desired velocity model's response is as expected. However, when the velocity is made a function of time desired velocities are not accurately followed. A proposed control methodology yields in designing the model-based controller. The proof of concept works perfect for the simulation model in turn leads to simulate the real robot. Designing the closed loop control helped us to eliminate the errors and provided feedback to the simulation model. The proposed

model is not suitable for the real robot. In future, a more sophisticated model would be developed which more exactly represents the actual robot.

This thesis proposes a new method of motion of control of a robot and validate its performance. The new method uses a neural network in the loop with a mathematical control law to provide estimations of the motion of the robot. This new technique does not require information on system parameters,does not require information from a mathematical dynamical model of the system, and may be more computationally efficient than traditional control approaches for systems that are complex or non linear. Hence, this thesis presents a new approach to implementing an artificial neural network into a closed-loop model-based state space control law to govern the motion of an autonomous robot.

REFERENCES

[1] F. Lewis and S. Ge, "Neural networks in feedback control systems," *Mechanical Engineer's Handbook*, 2005.

[2] P. J. Werbos, "Neural networks for control and system identification," in *Proc. IEEE Conf. Decision and Control*, (Fla.), 1989.

[3] K. Narendra and F. Lewis, "Special issue on neural network feedback control," *Automatica*, vol. 37, no. 8, Aug. 2001.

[4] Y. D. Landau, *Adaptive Control*. Marcel Dekker, 1979.

[5] L. V. Bertalanffy, *General System Theory*. Braziller, 1968.

[6] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–27, Mar. 1990.

[7] D. Haisloop and B. Holt, "A neural network structure for system identification," in *American Control Conference*, (San Diego, CA), pp. 2460–2465, 1990.

[8] J. Campos and F. L. Lewis, "Adaptive critic neural network for feedforward compensation," in *Proceedings of the American Control Conference*, (San Diego, CA), June. 1995.

[9] M. H. M. Beale and H. Demuth, *MATLAB Neural Network Toolbox User's Guide*, 2014.

[10] M. Egerstedt, *Motion Planning and Control of Mobile Robots*. PhD thesis, Royal Institute of Technology Sweden, Sweden, April 2000.

[11] F. Fahimi, *Autonomous Robots - Modeling, Path Planning, and Control*. Springer Science + Business Media, 2009.

[12] C. S. . Jim Albus, Tony Barbera, "Combining the strengths of software engineering and cognitive systems," in *AAAI Conference: Workshop on Intelligent Agent Architectures*, (San Jose, CA), 2004.

[13] G. Cybenko, "Approximation by superposition of a sigmoidal function," *Mathematics of Control signals and Systems*, vol. 2, pp. 303–314, 1989.

[14] H. Sussmann, "uniqueness of the weights for minimal feedforward nets with a given input-output map,," *Neural Networks*, vol. 5, pp. 589–593, 1992.

[15] J. Donald and N. Bhat, "Optimization neural net based predictive control," in *American Control Conference*, (San Diego, CA), pp. 2466–2471, 1990.

[16] J.S.Albus, "A new approach to manipulator control: the cerebellar model articulation controller equations CMAC," *Trans. ASME J. Dynam. Syst. Meas. Control*, vol. 97, pp. 220–227, 1975.

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

[18] L.-X. Wang *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*, 1994.

[19] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and functional-link net," *IEEE Trans. Neural Networks*, vol. 6, pp. 1320–1329, Nov. 1995.

[20] H. Z. V.S. Kasparian, C. Batur and J. Padovan, "Davidon least squares based learning algorithm for feedforward neural networks," *International Journal. of Neural Network*, vol. 7(4), pp. 187–197, 1994.

[21] I. Rivals and L. Personnaz, "Black-box modeling with state-space neural networks," *World Scientific*, 1995.

[22] D. P. amd L.A. Feldkamp, "Analyzing for lyapunov stability with adaptive critics," in *Proc. Int. Conf. Systems, Man, Cybernetics*, (Dearborn, MI), pp. 1658–1661, 1998.

[23] C. B. Panathula, "Solving practical problems in hilare-type mobile robot trajectory tracking control via application of nonlinear model predictive control," Master's thesis, University of Alabama, Huntsville, Huntsville, Alabama, 2012.

[24] H. D.-R. P. Gil, J. Henriques and A. Dourado, "State-space neural networks and the unscented kalman filter in on-line nonlinear system identification," in *IASTED International Conference on Intelligent Systems and Control*, 2001.

[25] F.-C. Chen and H. Khalil, "Adaptive control of nonlinear systems using neural networks," *Int. J. Control*, vol. 55, no. 6, pp. 1299–1317, 1992.