Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Trajectory Tracking Control of Mobile Robot Using Adaptive Neuro - Fuzzy Inference System (ANFIS)

A thesis submitted to Addis Ababa Institute of Technology, School of Graduate Studies, Addis Ababa University

in partial fulfillment of the requirement for the Degree of Master of Science in Electrical Engineering (**Electrical Control Engineering**).

By:

Nigusu Teshome Tufa

Advisor: Dr. Dereje Shiferaw Negash

Addis Ababa, Ethiopia

May 2018

Addis Ababa University
Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Trajectory Tracking Control of Mobile Robot Using Adaptive Neuro - Fuzzy Inference System (ANFIS)

By:

Nigusu Teshome Tufa

APPROVED BY BOARD OF EXAMINERS

_____	_____
Chairman, Department of Graduate Committee	Signature
_____	_____
Advisor's Name	Signature
_____	_____
Internal Examiner's Name	Signature
_____	_____
External Examiner's Name	Signature

DECLARATION

I, the undersigned, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Nigusu Teshome Tufa _____

Name Signature

Place: Addis Ababa Institute of Technology, Addis Ababa University, Addis Ababa

Date of Submission: May 22, 2018

This thesis has been submitted for examination with my approval as a university advisor.

Dr. Dereje Shiferaw Negash _____

Advisor's Name Signature

ACKNOWLEDGEMENT

My first thanks are to the Almighty GOD as all is and has been possible through him. I would like to extend my heartfelt indebtedness and gratitude to Dr. Dereje Shiferaw for his kindness in providing me an opportunity to work under his supervision and guidance. During this period, without his endless efforts, immense knowledge, deep patience, invaluable guidance and answers to my numerous questions, this research would have never been possible. I am especially obliged to him for teaching me both research and writing skills, which have been proven beneficial for my current research and future career. He showed me different ways to approach a research problem and the need to be persistent to accomplish any goal. It has been a great honour and pleasure for me to do research under the supervision of Dr. Dereje Shiferaw.

I am highly indebted to the Addis Ababa University for allowing me to join the postgraduate program. I am very grateful to the staff members of the Department of Electrical and Computer Engineering for all their cooperation.

My deep appreciation, grateful and thanks also go to Dr. Saravanakumar Gurusamy for his kind help and support by providing necessary resources as well as for his devoted assistance, thoroughly reading the manuscript, making comments and giving valuable suggestions during the period of my research work.

I would like to thank all my friends for their encouragement and understanding. Their help and lots of lovely memory with them can never be captured in words.

Finally, I thank my family for their unlimited support and continuous encouragement in my studies.

'Thank you all!!!'

ABSTRACT

In autonomous trajectory tracking navigation of mobile robots in the static environment is a source of problems. Because it is not possible to model all the possible conditions, the key point in the robot control is to design a system that is adaptable to different conditions and robust in static environments.

The subject of this thesis primarily addresses the trajectory tracking control of a differential drive mobile robot based on Adaptive Neuro-Fuzzy Inference System (ANFIS) controller. ANFIS has two layers like input fuzzy layer and the following neural network layer. The hybrid system combines the advantages of fuzzy logic, which deal with explicit knowledge that can be explained and understood, then also neural network, which deal with implicit knowledge, which can be acquired by learning. The kinematic modeling is developed for non-holonomic wheeled mobile robot systems. A learning algorithm based on neural network technique is developed to tune the parameters of fuzzy membership functions, which smooth the trajectory tracking with minimal error for the given path. Using the developed ANFIS controller, the mobile robots can be able to track the trajectory, and reach the target successfully in cluttered environments. The control objective has been to make the mobile robot actuator traces desired trajectory using ANFIS based controller. This has been done through the simulation of the robot model using the software MATLAB/Simulink version R2017a for a Pioneer 3-DX mobile robot.

TABLE OF CONTENTS

Contents Pages

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviation	Description
MR	Mobile Robots
AMR	Autonomous Mobile Robots
WMR	Wheeled Mobile Robots
DDWMR	Differential-Drive Wheeled Mobile Robots
DOF	Degrees Of Freedom
ICR	Instantaneous Center of Rotation
CW	Clock Wise
CCW	Counter Clock Wise
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Controller
MF	Membership Function
SMC	Sliding Mode Control
MPC	Model Predictive Control
AI	Artificial Intelligence
NN	Neural Networks
ANFIS	Adaptive Neuro-Fuzzy Inference System
GA	Genetic Algorithm
LSE	Least Square Estimate
TRS	Teaching/Learning Robotics with a Simulator
VR	Virtual Reality
ROS	Robotics Operating System
V-REP	Virtual Robot Experimentation Platform
GUI	Graphical User Interface
CUI	Custom User Interface
API	Application Program Interface

LIST OF SYMBOLS

Symbol	Signification
x	Input of the neuro-fuzzy controller
y	Input of the neuro-fuzzy controller
f	Output of the neuro-fuzzy controller
$A_{11}.....A_{m1}$	fuzzification of x by Gaussian membership function
$A_{21}....A_{m2}$	fuzzification of y by Gaussian membership function
μ_{ij}	fuzzy inference of T-norm
$w_1, ..., w_m$	output of the fuzzy controller
$x_1, ..., x_n$	a control system with 'n' inputs
R_i	linguistic rules
w_i	fuzzification of y by Gaussian membership function
$V(s)$	criterion function
$f(t)$	Output of the neuro-fuzzy controller
$f_d(t)$	Desired output
a_{ij}	parameter of the adaptation of the learning algorithm
b_{ij}	parameter of the adaptation of the learning algorithm
f_i	parameter of the adaptation of the learning algorithm
Γ_a	predefined constant
Γ_b	predefined constant
Γ_f	predefined constant
q	Vector of independent velocity variables
ξ	the chassis posture vector
θ	angle from X-axis to the robot's motion direction
V_x	represent respectively the instantaneous horizontal velocities
V_y	represent respectively the instantaneous vertical velocities
V	represent the intensity of the longitudinal velocity
ω	angular velocity of the robot
Ω_L	angular velocity of the left wheels

Ω_R	angular velocity of the right wheels
R_a	radius of the wheels
L	distance between the two wheels
q	State vectors
u	control vector
P_d	the set of point of the reference
$x_d\,(t)$	represent the x- coordinates of the robot according to the reference frame
$y_d\,(t)$	represent the y- coordinates of the robot according to the reference frame
$\theta_d\,(t)$	trajectory's curvature at each step time t
P(x, y, θ)	robot's track
e_θ	input of the Orientation ANFIS controller
ΔΩ	absolute value of angular velocity of difference between two wheels
$f_\theta(t)$	output of the Orientation ANFIS controller
w_i^{θ}	Adaptive parameter of the Orientation ANFIS controller
e_x	difference between the robot's position and the position's coordinates along with X-axis
e_y	difference between the robot's position and the position's coordinates along with Y-axis
Ω_p^{left}	speed of the left wheel
Ω_p^{right}	speed of the right wheel
$f_p(t)$	output of the position ANFIS controller
η	learning rate

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

In recent years, Robot gains an extensive interest and plays many important roles in the world. This is due to the wish to replace the humans with robots in dangerous tasks, and to use the robots in classical everyday tasks and in industry. Particularly, Robots can liberate humans from hazardous or dangerous works such as nuclear-waste cleaning, mining, forestry, agriculture, military duty, fire-fighting, Sea-exploration, and space-exploration, robotic wars and etc. Robots begin to affect our daily life like office automation and surgery. As well, robots may help the handicapped people in their daily life activities. Among these, Autonomous robots have a major influence on human life in the future.

A wheeled mobile robot (WMR) is a wheeled vehicle which is capable of an autonomous motion without external human drivers, because it is equipped, for its motion, with motors that are driven by an embedded computer. The WMR is considered in this thesis due to its wide usage among mobile robots, fast maneuvering, simple controllers and energy saving characteristics [1, 2].

The trajectory tracking control of wheeled mobile robots have been, and still are, the subjects of numerous research studies. The aim of trajectory tracking is to control wheeled mobile robots to follow a given predefined trajectory (reference trajectory) with reduced (minimal) error for the given path. In particular, that is the case of non-holonomic robot where the tracking trajectories and the control of motion are not independents because of non-holonomic constraints. For example, a two wheeled mobile robot can go anywhere but can’t follow any trajectories. Originally in robotics, the motion control was based on the problem of "how to move a piano from one room to another in a house without hitting anything" [3]. Generally, in the robotic domain, the trajectory planning for non-holonomic robot consists to find trajectories from an initial point to a final point by taking account of mechanical constraints and to control the robot in order that it follows the desired trajectories. Two main approaches to controlling wheeled mobile robots are trajectory tracking and posture stabilization.

Generally, a good control method must satisfy the robustness of stabilization and fast convergence. Thus, the control problems are need to find a suitable control technique to ensure the

robustness as well as the fast convergence. Obviously, the outstanding advantage of the classical feedback control techniques is robust in the linear system. However, in the nonlinear case, the fast convergence must be compromised with the stability. Alternatively, in artificial intelligence, the terms planning and AI planning refers to solve a discrete control problem [3]. In this case, instead of moving a piano through a continuous space, the task is to find a set of discrete actions allow to move the object from the initial point to the goal point.

The goal of this thesis work is to propose a new concept to control the motion of a mobile robot to track and reach the goal target. In fact, it is considered that all trajectories can be decomposed in two elementary motions: the linear motion and the rotation (nonlinear) motion. The proposed approach, which is based on Machine-Learning that uses an Adaptive Neural Fuzzy Inference System (ANFIS) control allows that the mobile robot moves towards the desired position, desired orientation and also can track any reference trajectories (linear or nonlinear). Neuro-fuzzy systems add the advantages of fuzzy reasoning to neural networks, which learn fast without needing symbolic representation of the system. In this study, a non-holonomic mobile robot control system is modeled and simulated. The system uses the adaptive neuro-fuzzy method to design the controller for trajectory tracking.

1.2 Statement of the Problem

This thesis work includes the development of Adaptive Neuro-Fuzzy Inference System (ANFIS) based trajectory tracking controllers for a non-holonomic differential drive mobile robot. Solving the trajectory tracking problem is a key question in robotics systems. The objective of trajectory tracking is to control a robot to track a desired trajectory by controlling position and orientation of the robot i.e., the error between the desired trajectory and the actual trajectory converges to zero. The error between the reference pose and the current pose of the robots (or error in the pose) can't be avoided i.e., the robot deviates the desired trajectory, which is caused by the slippage, disturbances, noise, vehicle-terrain interaction and measure errors of sensors.

The direct implementation of control in the control system of a real actuator is impossible without obtain correct kinematic model. After obtaining the correct kinematic model, it is possible to apply the control law to the trajectory tracking problem to make the robot traces desired trajectory. There are infinite number of path to move from one point to the next and following a desired trajectory

is still a challenging task. Therefore, a robot motion control to precisely tracking a given trajectory is a crucial problem. Thus, the problem of following a desired path is investigated.

Many traditional approaches were employed to solve the problem of trajectory tracking. Unfortunately, most approaches triggered speed jump when error change suddenly. To resolve the problem, ANFIS based trajectory tracking controller is proposed in this thesis. Therefore, Adaptive Neuro Fuzzy Inference Systems (ANFIS) have been developed to overcome limitations on the trajectory tracking problem of mobile robots by combining the advantages of both neural networks and fuzzy inference systems.

1.3 Objective of the Thesis

1.3.1 General Objectives

The main objective in this thesis is to make the mobile robot traces desired trajectory using adaptive neuro- fuzzy inference system.

1.3.2 Specific Objectives

- To design ANFIS based trajectory tracking controller for mobile robot.
- To simulate the developed ANFIS based controller using MATLAB SIMULINK.
- To evaluate the performance of the controller in terms of trajectory tracking.

1.4 Contributions of the Thesis

This thesis in general contributes to the growing field of mobile robot systems. Specific contributions of this thesis are mainly the design and simulation of novel trajectory tracking controllers for non-holonomic mobile robots. The Adaptive Neural Fuzzy Inference System (ANFIS) has been applied to implement the trajectory tracking control of wheeled mobile robot.

1.5 Organization of the Thesis

The thesis is organized as follows. Chapter 1 clarify the background and motivation of the thesis and gives a synthetic overview of the state of art relating the research in mobile robots system. Chapter 2 introduces the literature reviews. Chapter 3 covers the model of robot control system. Chapter 4 introduces ANFIS and trajectory control strategy. Chapter 5 gives information on the simulation results. A simulation is designed to verify the control strategy. Finally, chapter 6 concludes this thesis and indicates future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

In this chapter, the survey of literatures that are already reported by various researchers is discussed. Assuming that a path is generated through a path planner. The objective of the trajectory tracking controller is to control the mobile robots to follow the known path by controlling the robot linear and angular velocities. Due to the slippage, disturbances, noise, vehicle-terrain interaction and measure errors of sensors including external and internal sensors; the error between the reference path and the current position of the robots can't be avoided. Therefore, a robot movement control to precisely tracking a given trajectory is the crucial problem. There has been enormous activity in the study of path tracking in recent years. Much has been written on solving the problem of motion under non-holonomic constraints and so many different types of nonlinear trajectory tracking controllers have been proposed in the literature for the trajectory tracking of such robots.

2.2 Backstepping Kinematic Based Approaches

The dynamic work which is the integration of a kinematic controller and a torque controller for non-holonomic mobile robots was first proposed in [4]. A combined kinematic/torque controller is developed using backstepping and the stability is guaranteed using Lyapunov theory. The proposed control algorithm can be applied to three robot navigation problems: path following, tracking a reference trajectory and stabilization about a reference posture. This control algorithm provides a rigorous method of taking into account the specific vehicle dynamics to convert the steering system command into control input for the actual vehicle. First, feedback velocity control inputs are designed for the kinematic steering system to make the position error asymptotically stable. Then, a computed torque controller is designed such that the mobile robot velocities converge to the given velocity inputs. This control approach can be applied to a class of smooth kinematic system control velocity inputs and therefore, the same design procedure works for all of the three basic navigation problems.

This control law which is capable of dealing with the three basic non-holonomic navigation problems and considers the complete dynamics of the mobile robot is derived using the

backstepping approach. The method is valid as long as the velocity control inputs are smooth and bounded and the dynamics of the actual vehicles are completely known. In fact, the perfect knowledge of mobile robot parameters is unattainable. For example friction is very difficult to model in these systems.

To confront this problem a robust adaptive controller should be designed so that it can deal with uncertainties and unknown unmodeled disturbances. Therefore, no need of a priori information of the dynamic parameters of the mobile robot because the controller will learn them online. Robust control methods such as sliding mode control and adaptive intelligent control methods such as fuzzy logic and neural network controllers are the possible solutions to this problem and will be explained in the rest sections of this chapter.

2.3 Sliding Mode Based Approaches

Sliding mode is also called sliding error surface or a hyper surface that is a function of observer error function. Once the tracking error trajectories have reached this hyper surface, the error based action makes the observation errors "slide" to zero. A robust tracking control for a non-holonomic wheeled mobile robot is proposed in [5, 6]. A novel sliding control law was proposed for asymptotically stabilizing the mobile robot to a desired trajectory. It is shown that this proposed scheme is robust to bounded external disturbances.

A large difficulty in controlling non-holonomic mobile robots is that in the real world there are uncertainties in their modeling. Taking into account intrinsic characteristics of mobile robots such as actual vehicle dynamics, inertia and power limits of actuators and localization errors, their dynamic equations could not be described as a simplified mathematical model. The advantages of using sliding mode control for the solution include being fast, having a good transient response and robustness with regard to parameter variations. The dynamic model of the robot is used in this work to describe their behavior with bounded disturbances. By means of a computed torque method, error dynamics of the mobile robot are linearized and sliding mode control law is applied for stabilizing the robot to a reference trajectory and compensating for the existing disturbances. So basically, the proposed control scheme uses the computed torque method for feedback linearization of the dynamic equation and a theory of sliding mode for the robust control.

This proposed control scheme has the ability to solve the trajectory tracking problem based on dynamic modeling when the reference trajectory is not given as a closed form and it is shown that

by applying the sliding mode control, the controller behavior of the mobile robot is robust against initial condition errors and external disturbances such as integration errors, noise in control signal, localization errors and etc. [5].

The main problem with this control approach is that it again needs the precise dynamic model of the robot for the computed torque part. When we want to do the feedback linearization of the nonlinear dynamic model of the system, we need to know the exact robot dynamics. This disadvantage of this control law leads us to use an adaptive or intelligent controller with learning ability so that we can learn the robot dynamics online without knowing for the controller design. So many different methods of adaptive and intelligent control can be used to deal with this problem. Some of them will be explained in the next sections of this chapter.

2.4 Fuzzy Logic Based Approaches

A robust tracking control for a non-holonomic wheeled mobile robot using fuzzy logic control is proposed in [7, 8, 9, and 10].

The proposed algorithm in [7] deals with development of a controller of fuzzy-genetics algorithm for 2-DOF Wheeled Mobile Robot. The global inputs to the WMR are a reference position and a reference velocity which are time variables. The output of WMR which is a current position of WMR is estimated by dead-reckoning algorithm. Dead-reckoning algorithm determines the present position of WMR in real time by adding up the increased position data to the previous one in sampling period. The tracking controller makes position error to be converged to zero. In order to reduce position error, a compensation velocities on the track of trajectory is necessary. Therefore, a controller using fuzzy-genetic algorithm is proposed to give velocity compensation in this system. Input variables of two fuzzy logic controllers (FLCs) are position errors in every sampling time. The output values of FLCs are compensation velocities. Genetic algorithms (GAs) are implemented to adjust the output gain of fuzzy logic.

In [8] a control algorithm based on the errors in postures of mobile robot which generates correction signals for the left and right motor speeds is proposed. The control strategy is based on a feed forward velocity profile and the correcting signal in the velocity generated from the FLC according to the postures errors. Simulation results demonstrate the effectiveness of the proposed

algorithm, which proved the good tracking results and showed the robustness of the algorithm against the uncertainties in the system model.

The fuzzy logic controller proposed in [9] is based on a backstepping approach to ensure asymptotic stabilization of the robots position and orientation around the desired trajectory by taking into account the kinematics and dynamics of the vehicle. Mamdani inference system is used to construct the controller; with nine if-then rules and the centroid of area method as our deffuzification strategy where the input torques and velocities are considered as linguistic variables. Many researchers used only the kinematic model (steering system) to solve the tracking control problem, where the velocity, used as input control, is assumed to be supplied by the mobile robot whose dynamic of actuators is neglected. Real prototype have actuated wheels whose slip rate, rolling, inertia moment and mass distribution contribute to the forces exerted on the structure of the vehicle thus affecting the accuracy and full maneuverability of the robot. Motivated by this fact, the dynamic model of the robot is used in this wok to convert the steering system into the actual vehicle. The triangle and trapezoidal-shaped membership functions are used in this design along with three fuzzy partitions and nine rules.

The controller in [10] presents a new tracking method for a mobile robot by combining predictive control and fuzzy logic control. Trajectory tracking of autonomous mobile robots usually has non-linear time-varying characteristics and is often perturbed by additive noise. To overcome the time delay caused by the slow response of the sensor, the algorithm uses predictive control, which predicts the position and orientation of the robot. In addition, fuzzy control is used to deal with the non-linear characteristics of the system.

2.5 Adaptive Based Approaches

Adaptive control methods for trajectory tracking of a wheeled mobile robot are proposed in [11, 12, 13, and 14].

A dual adaptive dynamic controller for trajectory tracking of non-holonomic wheeled mobile robots is presented in [11]. The controller is developed entirely in discrete-time and the robot's nonlinear dynamic functions are assumed to be unknown. A Gaussian radial basis function neural network is employed for function approximation, and its weights are estimated stochastically in real-time. In contrast to adaptive certainty equivalence controllers hitherto published for mobile robots, the proposed control law takes into consideration the estimates' uncertainty, thereby leading

to improved tracking performance. The proposed method is verified by realistic simulations and Monte Carlo analysis.

A novel simple adaptive tracking controller is presented in [13] based on the kinematics models. An artificial potential field is used to navigate the wheeled mobile robot in the controller. Easy design, fast convergence, and adaptability to other non-holonomic mobile are obvious advantages. Stability of the rule is proved through the use of a Lyapunov function.

In [14] an adaptive control rules, at the dynamics level, for the non-holonomic mobile robots with unknown dynamic parameters is proposed. Adaptive controls are derived for mobile robots, using backstepping technique, for tracking of a reference trajectory and stabilization to a fixed posture. For the tracking problem, the controller guarantees the asymptotic convergence of the tracking error to zero. For stabilization, the problem is converted to an equivalent tracking problem, using a time varying error feedback, before the tracking control is applied. The designed controller ensures the asymptotic zeroing of the stabilization error. The proposed control laws include a velocity/acceleration limiter that prevents the robot's wheels from slipping.

2.6 Neural Network Based Approaches

Nowadays, the neural networks have been proved to be a promising approach to solve complex control problems. The neural network controllers are generally based on the function approximation property and learning ability of the neural network. The use of neural network controllers for trajectory tracking of mobile robots has been proposed in [15, 16, 17 and 18].

A wavelet neural network based controller for mobile robots is proposed in [15]. The work presents a predictive control scheme for mobile robots that possess complexity, nonlinearity and uncertainty. A multi-layer back-propagation neural network is employed as a model for nonlinear dynamics of the robot. The neural network is constructed by the wavelet orthogonal decomposition to form a wavelet neural network that can overcome the problems caused by local minima of optimization. The wavelet network is also helpful to determine the number of the hidden nodes and the initial value of the weights.

The neural network trajectory tracking controller proposed in [16] uses the learning property of the neural network to make an adaptive controller which adapts the backstepping controller gains. The proposed control approach has the properties to quickly drive the position error to zero and to

indicate better smooth movement in the tracking performance process. This novel control approach integrated the backstepping controller with compound orthogonal networks and improves its performance by using the learning property of the neural network.

The neural network controller proposed in [17] is based on the neural network function approximation property and can deal with unmodeled bounded disturbances and unstructured unmodeled dynamics of the mobile robot. The neural network is combined with the backstepping controller to learn the full dynamics of the mobile robot and convert the velocity output of the backstepping controller to a torque input for the actual vehicle. The advantage of having neural networks in this approach is that there is no need to know the dynamic model of the robot and the neural network will learn it online without a priori knowledge of the dynamics.

CHAPTER 3

MODEL OF WHEELED MOBILE ROBOT

3.1 Introduction

This chapter briefly outlines the modeling of a differential drive non-holonomic WMR to obtain the equations used in this development. The development and control of a robotic system require an understanding and a suitable representation of a "model" of the system. Any model is an idealization of the actual system. The modeling of a differential drive mobile robot platform consists of kinematic modeling; which is the basic study of the mechanical systems. Kinematic modeling deals with the geometric relationships that govern the system and studies the mathematics of motion without considering the forces causing the motion.

The process of understanding the motions of a robot begins with the process of describing the contribution each wheel provides for motion. Each wheel has a role in enabling the whole robot to move. By the same token, each wheel also imposes constraints on the robot's motion; for example, refusing to skid laterally. The notation that allows expression of robot motion in a global reference frame as well as the robot's local reference frame is introduced in the following section. Then, the construction of simple forward kinematic models of motion, describing the robot movement as a whole as a function of its geometry and individual wheel behavior is demonstrated by using this notation.

3.2 Kinematic Modeling of the Mobile Robot

Deriving a model for the whole robot's motion is a bottom-up process. Each individual wheel contributes to the robot's motion and at the same time, imposes constraints on the robot's motion. Wheels are tied together based on robot chassis geometry, and therefore their constraints combine to form constraints on the overall motion of the robot chassis. But the forces and constraints of each wheel must be expressed with respect to a clear and consistent reference frame. This is particularly important in mobile robotics because of its self-contained and mobile nature; a clear mapping between global and local frames of reference is required. It begins by defining these reference frames formally, then using the resulting formalism to annotate the kinematics of individual wheels and whole robots.

3.2.1 Mobile Robot Position Representation

Throughout this analysis the robot is modeled as a rigid body on wheels, operating on a horizontal plane. The total degree of freedom (DOF) of this robot chassis on the plane is three; two for position in the plane and one for orientation along the vertical axis, which is orthogonal to the plane. Of course, there are additional degrees of freedom and flexibility due to the wheel axles, wheel steering joints, and wheel castor joints. However, by robot chassis only the rigid body of the robot is referred, ignoring the joints and degrees of freedom internal to the robot and its wheels. The first step for the kinematic modeling is to define appropriate coordinate systems for the platform which will be described in the next section.

3.2.1.1 Coordinate Systems

The main function of the coordinate systems is to represent the position of the robot. Two coordinates are exist for derivation model: Cartesian or Polar coordinates. Hence, because of popularity of Cartesian coordinates, the modeling done in Cartesian coordinates. The following two Cartesian coordinate systems are used for mobile robot modeling and control purposes:

a) Global Inertial Frame $\{X_I, Y_I\}$: This is the fixed coordinate system in the plane of the robot. It define an arbitrary inertial basis on the plane as the global reference frame from some origin *O*.

b) Local Robot Frame $\{X_R, Y_R\}$: This is the coordinate system attached to the robot and defines two axes relative to *P* on the robot chassis.

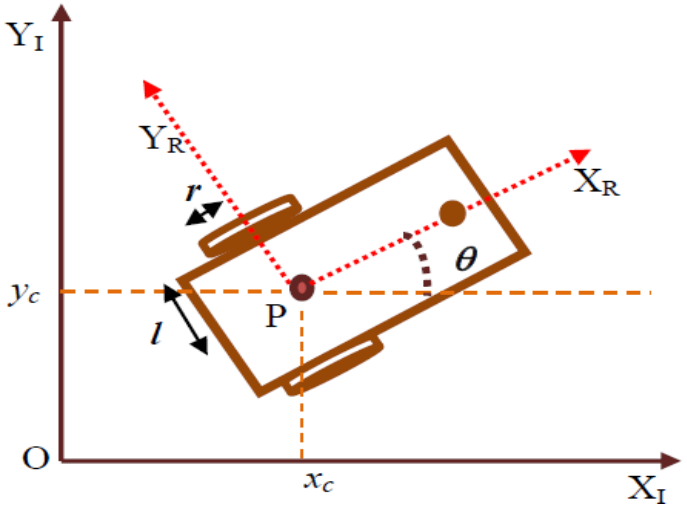


Figure 3.1: The global reference frame and the robot local frame coordinate systems

In order to specify the position of the robot on the plane a relationship between the global reference frame of the plane and the local reference frame of the robot is established as in figure 3.1. To specify the position of the robot, choose a point *P* on the robot chassis as its position reference point. The position of *P* in the global reference frame is specified by coordinates x_c and y_c, and the angular difference between the global and local reference frames is given by θ. The pose of the robot can be described as a vector with these three elements. Note, the use of the subscript *I* and *R* to clarify the basis of this pose as the inertial reference frame and robot's local reference frame respectively:

$$q_I = [x_I \quad y_I \quad \theta_I]^T \tag{3.1}$$

$$q_R = [x_R \quad y_R \quad \theta_R]^T \tag{3.2}$$

The important issue that needs to be explained is the mapping between these two frames. To describe robot motion in terms of component motions, it will be necessary to map motion along the axes of the global reference frame to motion along the axes of the robot's local reference frame. The mapping between these two frames is accomplished using the standard orthogonal rotation transformation:

$$\dot{q}_R = R(\theta)\, \dot{q}_I \tag{3.3}$$

Where

$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

This matrix is known as the orthogonal rotation matrix and can be used to map motion in the global reference frame in terms of the local reference frame motion.

3.2.2 Differential Drive Kinematics

Many mobile robots use a drive mechanism known as differential drive. It consists of 2 drive wheels mounted on a common axis, and each wheel can independently being driven either forward or backward.

One of the simplest mobile robot constructions is a chassis with two fixed wheels. Understanding this construction helps to grasp some basic kinematics of the robots. Usually differential drive mobile robots have an additional castor wheel which is usually used for stability. Sometimes roller-balls can be used but from the kinematics point of view, there are no differences in calculations.

As it can rotate freely in all directions, the calculation of the castor wheel can be omitted because it only has a very little influence over the robot's kinematics. In case of differential drive, to avoid slippage and have only a pure rolling motion, the robot must rotate around a point that lies on the common axis of the two driving wheels. This point is known as the instantaneous center of rotation (ICR) or the instantaneous center of curvature (ICC). By changing the velocities of the two wheels, the ICR will move and different trajectories will be followed as shown in Figure 3.2.

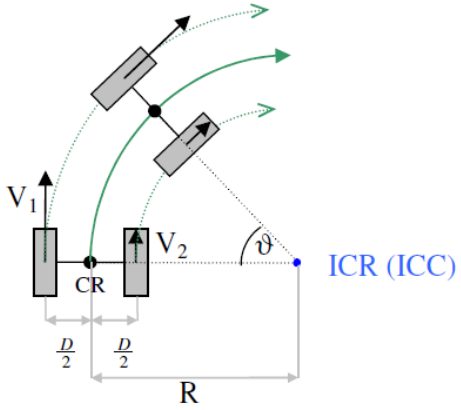


Figure 3.2: The differential drive motion example [44].

At each moment in time the left and right wheels follow a path (Figure 3.2) that moves around the ICR with the same angular rate $\omega = d\theta/dt$. Because the rate of rotation ω about the ICR must be the same for both wheels, the following equations can be written as:

$$V_{CR} = \omega \, R \tag{3.5}$$

Therefore, based on equation (3.5), the following equations can be derived as:

$$V_1 = \omega \left(R + \frac{D}{2}\right) = \left(V_{CR} + \frac{D}{2}\,\omega\right) \tag{3.6}$$

$$V_2 = \omega \left(R - \frac{D}{2}\right) = \left(V_{CR} - \frac{D}{2}\,\omega\right) \tag{3.7}$$

Where

V_1 and V_2 are the linear velocities of the left and right wheels respectively.

V_{CR} is the linear velocity of the *CR* point of the platform

ω is the rotational velocity of the platform

D is the distance between the two wheels, and

R is the signed distance from the *ICC* to the midpoint between the two wheels.

Note that: V_1, V_2, and *R* are all functions of time.

The linear velocity of the *CR* point, which is the midpoint between the two wheels, can be calculated as the average of the velocities V_1 and V_2:

$$V_{CR} = \frac{V_1 + V_2}{2} = \frac{R}{2}(\Omega_1 + \Omega_2) \quad (3.8)$$

At any moment in time ω and R can be solved as:

$$\omega = \frac{V_2 - V_1}{D} = \frac{R}{D}(\Omega_2 - \Omega_1) \text{ and } \quad R = \frac{D}{2}\frac{V_2 + V_1}{V_2 - V_1} \quad (3.9)$$

Therefore, from equations (3.8) and (3.9), the angular wheel speed can be:

$$\Omega_1 = \frac{1}{R}(V_{CR} - \frac{D}{2}\omega) \quad (3.10)$$

$$\Omega_2 = \frac{1}{R}(V_{CR} + \frac{D}{2}\omega) \quad (3.11)$$

Where Ω_1 and Ω_2 are the rotational velocities of the left and right wheels respectively.

There are three interesting cases with these kinds of drives.

1. If $V_1 = V_2$, then we have forward linear motion in a straight line. R becomes infinite, and there is effectively no rotation, i.e., ω is zero (Figure 3.3a).

2. For different values of V_1 and V_2, the mobile robot does not move in a straight line but rather follows a curved trajectory around a point located at a distance R from CR (Figure 3.3b and 3.3c – turning around one of the wheels), changing both the robot's position and orientation.

3. If $V_1 = -V_2$, then $R = 0$, and we have rotation about the midpoint of the wheel axis (Figure 3.3d).

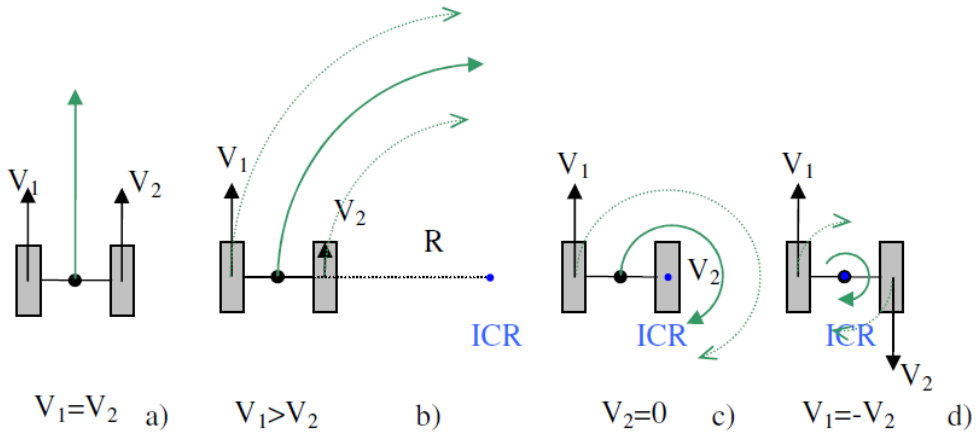


Figure 3.3. The different moving possibilities for differential drive [44]:

a) Motion forward or backward, R is infinite, b) turning R>D/2, c) turning R=D/2, d) turning R=0;

A differential drive mobile robot is very sensitive to the relative velocity of the two wheels. Small errors in the relative velocities between the wheels can affect the robot trajectory. They are also very sensitive to small variations in the ground plane, and may need extra castor wheels for balance support.

3.2.2.1 The Differential Drive Odometry

Let's try to find formulas by means of which is possible to compute the actual position of the robot.

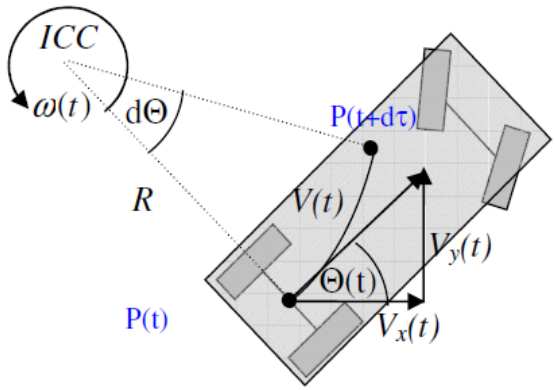


Figure 3.4. Odometry for differential drive [44].

Suppose that a differential drive robot is rotating around the point *ICC* with an angular velocity $\omega(t)$. During the infinite short time $d\tau$ the robot will travel the distance from the point $P(t)$ to $P(t+d\tau)$ with a linear velocity $V(t)$. $V(t)$ has two perpendicular components, one along the *X*- axis: $V_x(t)$, and the other along the *Y*- axis: $V_y(t)$. For infinite short time the robot is moving along a straight line tangent in the point $P(t)$ to the real trajectory of the robot. Based on the two components of the velocity $V(t)$, the traveled distance in each direction can be calculated as [44]:

$$dx = V_x(t)d\tau \quad (3.12)$$

$$dy = V_y(t)d\tau \quad (3.13)$$

where

$$V_x(t) = V(t)\cos\theta(t) \quad (3.14)$$

$$V_y(t) = V(t)\sin\theta(t) \quad (3.15)$$

Similarly, the angle of the rotation can be obtained as follows:

$$d\theta = \omega(t)d\tau \quad (3.16)$$

By integrating equations (3.12), (3.13) and (3.16), the time domain can be:

$$x(t) = \int_0^t V_x(\tau)d\tau + x_o$$

$$y(t) = \int_0^t V_y(\tau)d\tau + y_o \qquad (3.17)$$

$$\theta(t) = \int_0^t \omega(\tau)\, d\tau + \theta o$$

Where

(x_0, y_0, θ_0) are the initial pose.

Using equations (3.14) and (3.15), the equation (3.17) can be rewritten as:

$$x(t) = \int_0^t V(\tau)\cos\theta(\tau)\ d\tau + x_o$$

$$y(t) = \int_0^t V(\tau)\sin\theta(\tau)\ d\tau + y_o \qquad (3.18)$$

$$\theta(t) = \int_0^t \omega(\tau)\, d\tau + \theta o$$

Formulas in equation (3.18) are valid for all robots capable of moving in a particular direction $\theta(t)$ at a given velocity $V(t)$. For the special case of differential drive robot, based on equations (3.8) and (3.9), equation (3.18) can be rewritten as:

$$x(t) = \frac{1}{2}\int_0^t [V_1(\tau) + V_2(\tau)]\cos\theta(\tau)\ d\tau + x_o \qquad (3.19)$$

$$y(t) = \frac{1}{2}\int_0^t [\,V_1(\tau) + V_2(\tau)]\sin\theta(\tau)\ d\tau + y_o \qquad (3.20)$$

$$\theta(t) = \frac{1}{D}\int_0^t [V_2(\tau) - V_1(\tau)]\, d\tau + \theta o \qquad (3.21)$$

3.2.3 Forward Kinematic Models

In the simplest cases, the mapping described by equation (3.3) is sufficient to generate a formula that captures the forward kinematics of the mobile robot. More formally, consider the example shown in Figure 3.5. The goal of the robot forward kinematic modeling is to find the robot speed in the global inertial frame as a function of the wheels speeds and the geometric parameters of the robot.

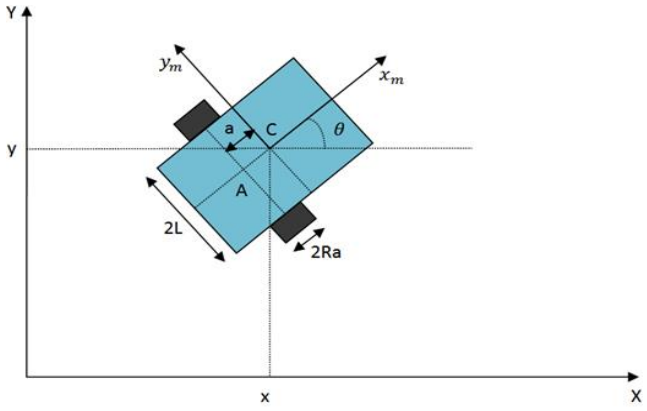


Figure 3.5: The differential drive mobile robot model

The following notations will be used in this thesis:

A: The intersection of the axis of symmetry with the driving wheels axis

C: The center of mass of the platform

a: The distance between the center of mass and driving wheels axis in *x*-direction

L: The distance between each driving wheel and the robot axis of symmetry in *y*-direction

R_a: The radius of each driving wheel

Ω_R: The rotational velocity of the right wheel

Ω_L: The rotational velocity of the left wheel

v: The translational velocity of the platform in the local frame

ω: The rotational velocity of the platform in the local and global frames

The forward kinematic problem can be described as the problem of finding the robots overall speed in the global reference frame as the following function:

$$\dot{q}_I = [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T = f(wheels\ speed, geometric\ parameters)$$

$$\dot{q}_I = [\dot{x} \quad \dot{y} \quad \dot{\theta}]^T = f(\Omega_R, \Omega_L, L, R_a, \theta) \qquad (3.22)$$

The strategy will be to first compute the contribution of each of the two wheels in the local reference $\dot{q}_R$. First, consider the contribution of each wheel's spinning speed to the translation speed at *A* in the direction of $+x_m$. If one wheel spins while the other wheel contributes nothing and is stationary, since *A* is halfway between the two wheels, it will move instantaneously with half the speed: $\dot{x}_{mR} = \frac{R_a}{2}\Omega_R$ and $\dot{x}_{mL} = \frac{R_a}{2}\Omega_L$. In a differential drive robot, these two contributions

can simply be added to calculate the $\dot{x}_R$ component of $\dot{q}_R$. Neither wheel can contribute to make sideways motion in the robot's reference frame, so $\dot{y}_R$ is always zero.

Once again, the contributions of each wheel can be computed independently and just added for computing rotational component $\dot{\theta}_R$. Consider the right wheel, forward spin of this wheel results in counterclockwise (CCW) rotation at point A. If the right wheel spins alone, the robot pivots around the left wheel. The rotation velocity ω_R at A can be computed because the wheel is instantaneously moving along the arc of a circle of radius 2L: $\omega_R = \frac{R_a}{2L}\Omega_R$. The same calculation applies to the left wheel, with the exception that forward spin results in clockwise (CW) rotation at point A: $\omega_L = -\frac{R_a}{2L}\ \Omega_L$.

Therefore by combining these individual formulas yields the translational speed and the rotational velocity in the robot reference frame as follows:

$$\dot{x}_R = \frac{R_a}{2}(\Omega_R + \ \Omega_L) \tag{3.23}$$

$$\dot{y}_R = 0 \tag{3.24}$$

$$\dot{\theta}_R = \frac{R_a}{2L}(\Omega_R - \ \Omega_L) \tag{3.25}$$

From equation (3.3), it is known that the robot's motion in the global reference frame from motion in its local reference frame can be computed as: $\dot{q}_I = R(\theta)^{-1}\dot{q}_R$

Therefore, the forward kinematic model for the differential-drive mobile robot in the inertial frame is:

$$\dot{q}_I = R(\theta)^{-1}\begin{bmatrix}\dot{x}_R\\ \dot{y}_R\\ \dot{\theta}_R\end{bmatrix} \tag{3.26}$$

$$\dot{q}_I = R(\theta)^{-1}\ \frac{R_a}{2}\begin{bmatrix}\Omega_R + \ \Omega_L\\ 0\\ \frac{\Omega_R - \Omega_L}{L}\end{bmatrix} \tag{3.27}$$

The inverse of the rotation matrix is:

$$R(\theta)^{-1} = \begin{bmatrix}cos\,\theta & -sin\,\theta & 0\\ sin\,\theta & cos\,\theta & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3.28}$$

Therefore, the robot velocity in the global inertial frame is:

$$\dot{q}_I = \begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \frac{R_a}{2} \begin{bmatrix} \Omega_R + \Omega_L \\ 0 \\ \frac{\Omega_R - \Omega_L}{L} \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{R_a}{2} cos\,\theta & \frac{R_a}{2} cos\,\theta \\ \frac{R_a}{2} sin\,\theta & \frac{R_a}{2} sin\,\theta \\ \frac{R_a}{2L} & -\frac{R_a}{2L} \end{bmatrix} \begin{bmatrix} \Omega_R \\ \Omega_L \end{bmatrix} \qquad (3.29)$$

The above equation is the general forward kinematic equation for a differential drive mobile robot. This approach to kinematic modeling can provide information about the motion of a robot given its component wheel speeds in straightforward cases. However, to determine the space of possible motions for each robot chassis design, it is must go further, describing formally the constraints on robot motion imposed by each wheel.

3.2.4 Types of Wheel

A wheeled mobile robot is a vehicle which is capable of an autonomous motion (without external human driver) because it is equipped with motors that are driven by output from on boarded microcontroller based on sensor information. Two constraints can be observed for every wheel type while a wheeled robot is in motion. The first constraint enforces the concept of rolling contact as represented in Figure 3.6 (a).The second constraint enforces the concept of no lateral slippage, that the wheel must not slide orthogonal to the wheel plane as shown in Figure 3.6 (b). The initial stage of a kinematic model of the robot is to express constraints on the motions of individual wheels. Thereby, the movement of the entire robot can be computed by combining the motions of individual wheels.

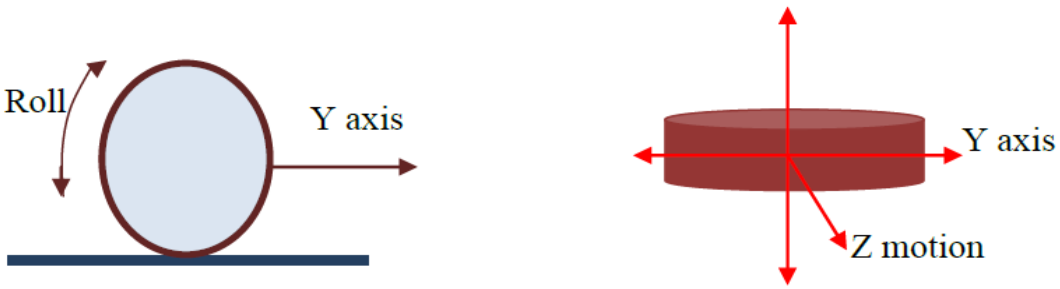


Figure 3.6 Types of wheels: a) Rolling motion b) Lateral slip

Based on the geometrical constraints, wheel can be categorized into five basic types: Conventional Fixed standard wheel, Steered standard wheel, Castor wheel, Swedish wheel and Spherical wheels. The castor wheel, Swedish wheel and spherical wheel impose no kinematic constraints on the robot chassis, since can range freely in all of these cases owing to the internal wheel degrees of freedom. Only fixed standard wheels and steerable standard wheels have impact on robot chassis kinematics and therefore, require consideration when computing the robot's kinematic constraints.

The fixed standard conventional wheels are the most widely used among wheel mobile robots with wheeled locomotion. These wheels are simple to construct, require less maintenance, provide smooth motion, offer high load carrying capacity and are cheap.

3.2.5 Wheel Kinematic Constraints

The first step to a kinematic model of the robot is to express constraints on the motions of individual wheels. Usually, the mechanical mobile robot solution namely "two-wheel differential drive mobile robot" has three wheels minimum. Two separately controlled "drive wheels" have a common horizontal axis which is fixed (regarding its body) during robot operation. By their angular velocities, these "drive wheels" assure the mobility of the mobile robot. One free wheel, (namely "castor" wheel which is a passive one) assure the robot equilibrium, is mounted independently on a vertical axis not on a driven axis of the mobile robot body. In consequence, a castor wheel is automatically and free aligned on the route as a result of the forces developed by only the two "drive wheels" [33].

The speed difference between both two independently-driven coaxial wheels results in a rotation of the vehicle about the center of the axle while the wheels act in concert to produce motion in the forward or reverse direction. Such a robot can rotate on the spot (i.e., without moving the midpoint between the wheels), provided that the angular velocities of the two wheels are equal and opposite. Mobile robots operate at relatively low speeds and assume vertical motion is absent.

However, several important assumptions will simplify the analysis. Assume that, the wheel always remains in vertical during the motion of a robot and there is no sliding at the single point of contact between the ground plane and the wheel. It means that the wheel is in motion under only pure rolling conditions and rotation about the vertical axis through the contact point.

There is no vertical axis of rotation or steering for the fixed standard wheel. It means the angle between the chassis and the wheel axis is fixed, therefore it is limited to move the robot back and forth along the wheel plane and rotation around its contact point with the ground plane.

The following assumptions about the wheel motion will cause the robot kinematic constraints:

- ✓ Movement on a horizontal plane.
- ✓ Point contact between the wheels and ground. Assume that the plane of wheel always remains vertical and that there is in all cases one single point of contact between the wheel and the ground plane.
- ✓ Wheels are not deformable.
- ✓ Pure rolling which means that we have the instantaneous center of zero velocity at the contact point of the wheels and ground.
- ✓ No slipping, skidding or sliding.
- ✓ No friction for rotation around contact points.
- ✓ Steering axes orthogonal to the surface.
- ✓ Wheels are connected by a rigid frame (chassis).

Considering the above assumptions about the wheel motion, the robot will have a special kind of constraint called Non-holonomic constraint. A non-holonomic constraint is a constraint on the feasible velocities of a body. For the case of the differential drive mobile robot, it can simply mean that the robot can move in some directions (Forward and backward) but not others (Sideward) as is shown in figure 3.7.

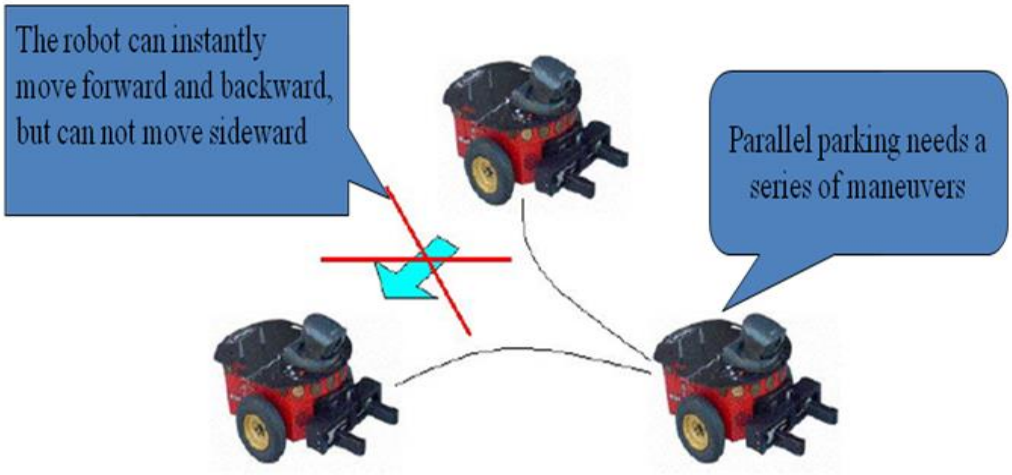


Figure 3.7: The non-holonomic constraint on the robot motion

3.2.6 Holonomicity of Mobile Robot

In the robotics community, when describing the path space of a mobile robot, often the concept of holonomy is used. The term holonomy has broad applicability to several mathematical areas, including differential equations, functions and constraint expressions. In mobile robotics, the term refers specifically to the kinematic constraints of the robot chassis.

Holonomic versus Non-holonomic:

- A non-holonomic kinematic constraint requires a differential relationship, such as the derivative of a position variable. Furthermore, it cannot be integrated to provide a constraint in terms of the position variables only. A holonomic kinematic constraint can be expressed as an explicit function of position variables only. For example, in the case of a mobile robot with a single fixed standard wheel, a holonomic kinematic constraint would be expressible using α, $l, r, \varphi, x, y, \theta$ only. Such a constraint may not use derivatives of these values, such as $\dot{\varphi}$ or $\dot{\xi}$.
- A non-holonomic mobile robot configuration is described by more than three coordinates. Three values are needed to describe the location and orientation of the robot, while others are needed to describe the internal geometry. However, a holonomic mobile robot can be described by three coordinates. The internal geometry does not appear in the kinematic equations of the abstract mobile robot, so it can be ignored. The robot can instantly develop a wrench or accelerate in an arbitrary combination of directions X, Y, θ.
- Non-holonomic robots are most prevalent because of their simple design and ease of control. By their nature, non-holonomic mobile robots have fewer degrees of freedom than holonomic mobile robots. These few actuated degrees of freedom in non-holonomic mobile robots are often independently controllable or mechanically decoupled, further simplifying the low-level control of the robot. Since they have fewer degrees of freedom, there are certain motions they cannot perform. This creates difficult problems for motion planning and implementation of reactive behaviors.
- Holonomicity, offers full mobility with the same number of degrees of freedom as the environment. This makes path planning easier because there aren‘t constraints that need to be integrated. Implementing reactive behaviors is easy because there are no constraints which limit the directions in which the robot can accelerate.

- In case of non-holonomic mobile robot, the wheels rotate in the forward direction and then backward to its previous angular position, the robot will not necessarily arrive in the same location due to slippage or any other conditions.
- In case of holonomic mobile robot, the wheels rotate in the forward direction and then backward to its previous angular position, the robot will arrive in the same location. So, holonomic robot can perform both Forward Kinematics (The angular rate difference between both wheels determines position & orientation of robot) and Inverse Kinematics (The position and orientation of a robot determines the angular rate difference between both wheels).

CHAPTER 4

ANFIS FOR TRAJECTORY TRACKING OF MOBILE ROBOT

4.1 Introduction

In this chapter, for a given desired trajectory, how to make the mobile robot track it in a stable and robust way is discussed. Then, the trajectory tracking of wheeled mobile robot based on an Adaptive Neural Fuzzy Inference System (ANFIS) control allowing the mobile robot moves towards the desired pose and can track any reference trajectories is discussed in section below. In the next section, after a review of the ANFIS controller (sections 4.2, 4.3, 4.4 and 4.5), the trajectory control strategy used for mobile robot is described in section 4.6.

4.2 Adaptive Networks

An adaptive network is a kind of multi-layer feed forward network. Some of the nodes in this network are adaptive, which means that each output of these nodes depends on the parameter(s) held in these nodes, and the learning rule specifies how these parameters should be updated to minimize the error. Each node performs a particular node function on incoming signals and parameters. The function of the node may vary from node to node, and the choice of each node function depends on the overall input-output function, which the adaptive network is required to carry out.

The links in an adaptive network only shows the flow direction of signals between nodes i.e., no weights are associated with these links. For indicating different capabilities in an adaptive network, circles and squares are used. A square node (adaptive node) has parameters while a circle node (fixed node) has no parameters. In order to achieve a desired input-output mapping, these parameters are updated according to given training data and a learning procedure. An example for adaptive networks is given in Figure 4.1.

The basic learning rule of adaptive networks is based on the gradient descent and the chain rule, which was proposed by Werbos [28] in the 1970's. But the gradient method is slow and has tendency to become trapped in local minima.

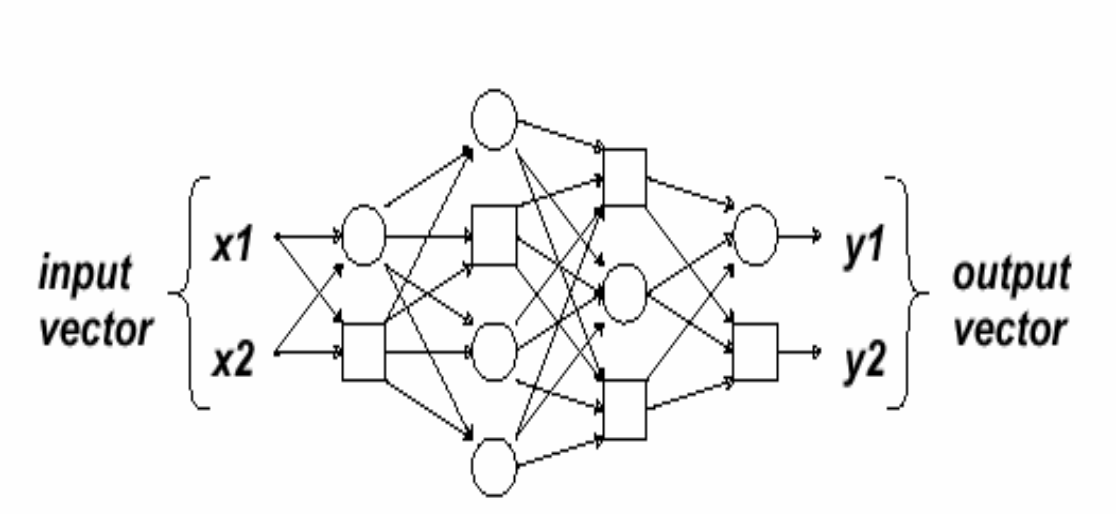


Figure 4.1 Adaptive Network [27]

4.3 Fuzzy Reasoning Mechanisms

The leading theory in quantifying uncertainty in scientific models from the late 19th century until the late 20th century had been probability theory. However, the gradual evolution of the expression of uncertainty using probability theory was challenged first in 1937 by Max Black with his studies in vagueness, then with the introduction of fuzzy sets by Zadeh [23].

Uncertainty can be manifested in many forms: it can be fuzzy (not sharp, unclear, imprecise, or approximate), it can be vague (not specific, amorphous), it can be ambiguous (too many choices, contradictory), it can be of the form of ignorance (not knowing something), or it can be a form due to natural variability (conflicting, random, chaotic, or unpredictable).

Zadeh extended the notion of binary membership to accommodate various "degrees of membership" on the real continuous interval [0 1], where the endpoints of 0 and 1 conform to no membership and full membership respectively. The sets on the universe X that can accommodate "degrees of membership" were termed by Zadeh as "fuzzy sets". A fuzzy set is a function that maps its elements to a real number value on the interval [0 1]. The functional mapping is given by $\mu_A(x) \in [0\ 1]$ and the symbol $\mu_A(x)$ is the degree of membership of an element x in the fuzzy set A. Therefore, $\mu_A(x)$ is a value on the unit interval that measures the degree to which an element x belongs in a fuzzy set A [23].

Fuzzy logic is different from predicate logic. In predicate logic assertions about the world are true or false which are referred as crisp values and have exact meaning. In fuzzy logic inputs have values according to how much they belong to a fuzzy set. Fuzzy sets are defined with membership functions, which measure the degree of similarity of an instance to the set as a numerical value.
Fuzzy control systems produce actions according to fuzzy rules based on fuzzy logic. The basic units of the fuzzy logic controller are:

1. Fuzzification
2. Fuzzy Rule Base
3. Fuzzy Inference Engine and
4. Defuzzification.

4.3.1. Fuzzification

Fuzzification is the process of making a crisp quantity ‘fuzzy’. In fuzzier, crisp input values are mapped to fuzzy sets using membership functions. This is done by simply recognizing that many of the quantities that is considered to be crisp and deterministic are actually not deterministic at all: they carry considerable uncertainty. If the form of uncertainty happens to arise because of imprecision, ambiguity, or vagueness, then the variable is probably fuzzy and can be represented by a membership function. Since the membership function essentially embodies all fuzziness for a particular fuzzy set, its description is the essence of a fuzzy property or operation. Just as there are an infinite number of ways to characterize fuzziness, there are an infinite number of ways to graphically depict the membership functions that describe this fuzziness.

4.3.2. Fuzzy Rule Base

Fuzzy rule base contains the IF-THEN rules which specifies the behavior of the system in a premise-consequent form that sets the foundation for approximate (imprecise) reasoning.

4.3.3. Fuzzy Inference Engine

Fuzzy inference engine maps input fuzzy sets to output fuzzy sets using fuzzy rule base.

4.3.4. Defuzzification

Defuzzifier maps the fuzzy output sets to crisp output value. Despite the fact that the bulk of the information assimilated every day is fuzzy, most of the actions or decisions implemented by humans or machines are crisp or binary. Therefore, in various applications and engineering

scenarios, there is a need to "defuzzify" the fuzzy results generated through a fuzzy systems analysis. In other words, it may eventually find a need to convert the fuzzy results to crisp results. Defuzzification is the conversion of a fuzzy quantity to a precise quantity, just as fuzzification is the conversion of a precise quantity to a fuzzy quantity. Some of the techniques used in the defuzzification process are: Center of Largest Area, Max membership Function, Centroid Method, Weighted Average Method, Mean Max Principle, Center of Sums and First (or Last) of Maxima [21]. The structure of a fuzzy control system is given in the Figure 4.2:

In a fuzzy control system of a mobile autonomous robot, firstly crisp sensor values is translated into linguistic classes in the fuzzifier, and then appropriate rules are fired in the fuzzy inference engine which generates a fuzzy output value, finally this value is translated into crisp turning angle or speed in the defuzzifier. This final result is passed to actuator as a command.

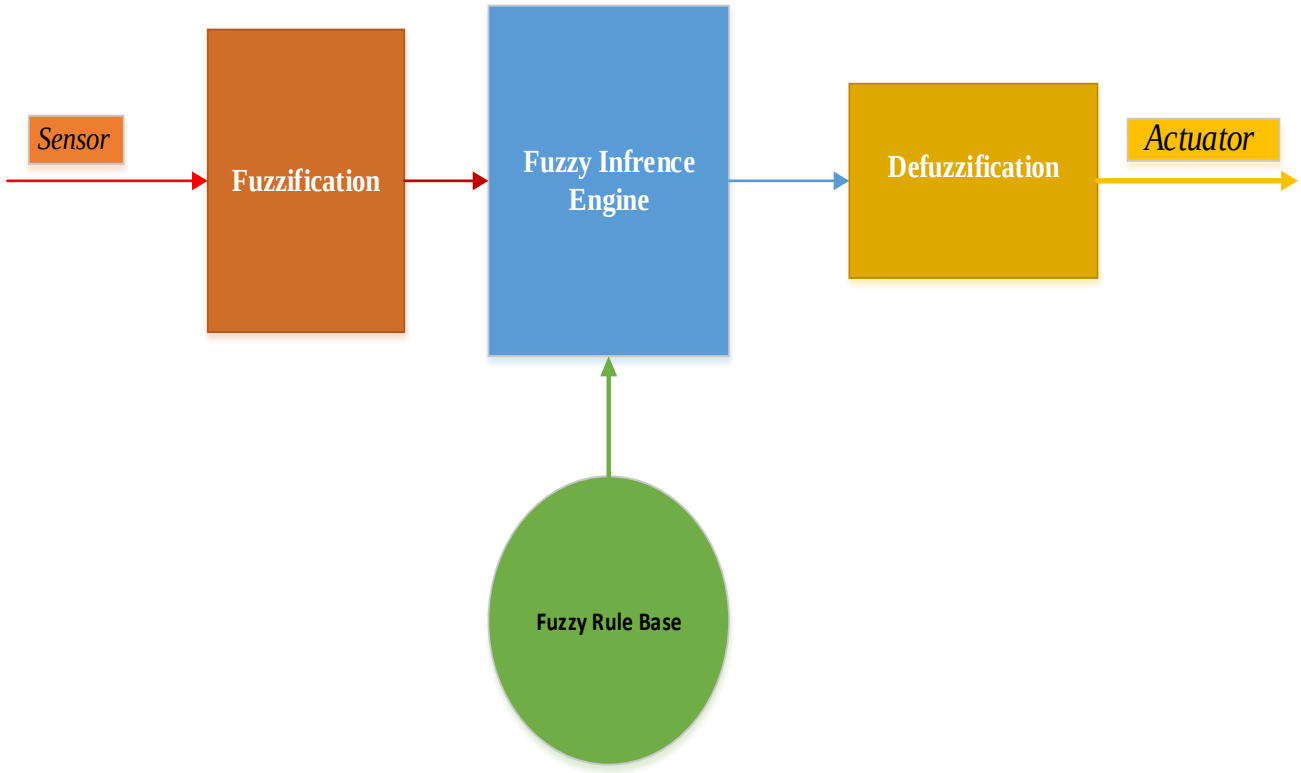


Figure 4.2. Fuzzy Logic Control System Architecture

4.3.5. Principles of Fuzzy Control Design

A number of assumptions are implicit in a fuzzy control system design. Six basic assumptions are commonly made whenever a fuzzy rule-based control policy is selected.

i. The plant is observable and controllable: state, input, and output variables are usually available for observation and measurement or computation.
ii. There exists a body of knowledge comprised of a set of linguistic rules, engineering common sense, intuition, or a set of input–output measurements data from which rules can be extracted.
iii. A solution exists.
iv. The control engineer is looking for a "good enough" solution, not necessarily the optimum one.
v. The controller will be designed within an acceptable range of precision.
vi. The problems of stability and optimality are not addressed explicitly; such issues are still open problems in fuzzy controller design.

The steps in designing a simple fuzzy control system are as follows:

a. Identify the variables (inputs, states, and outputs) of the plant.
b. Partition the universe of discourse or the interval spanned by each variable into a number of fuzzy subsets, assigning each a linguistic label.
c. Assign or determine a membership function for each fuzzy subset
d. Assign the fuzzy relationships between the inputs 'or states' fuzzy subsets on the one hand and the outputs 'fuzzy subsets' on the other hand, thus forming the rule-base.
e. Choose appropriate scaling factors for the input and output variables in order to normalize the variables to the [0, 1] or the [−1, 1] interval.
f. Fuzzify the inputs to the controller.
g. Use fuzzy approximate reasoning to infer the output contributed from each rule.
h. Aggregate the fuzzy outputs recommended by each rule.
i. Apply defuzzification to form a crisp output

Depending on the types of fuzzy reasoning and fuzzy if-then rules employed, most fuzzy inference systems can be classified into three types as shown in Figure 4.3.

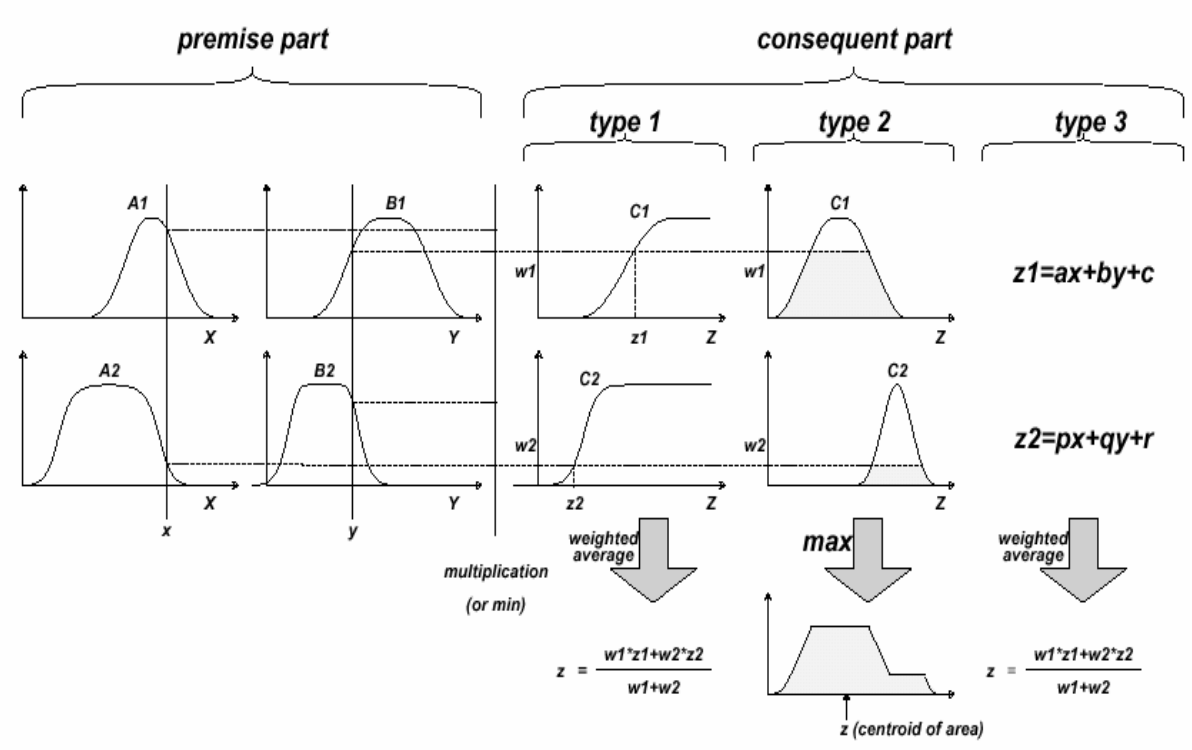


Figure 4.3. Commonly used fuzzy reasoning mechanisms (adopted from [27])

The differences between them lie in the consequents of their fuzzy rules, and thus their aggregation and deffuzification procedures differ accordingly.

Type 1: Tsukamoto Fuzzy Inference System

In type 1 systems, the overall output is calculated as weighted average of each rule's crisp output, which is represented by the rule's firing strength and output membership functions. The rules firing strength can be determined by the product or minimum of the output of input membership functions. The output membership functions used must be monotonically non-decreasing [29].

Type 2: Mamdani Fuzzy Inference System

In type 2 systems, which is called Mamdani model, standard fuzzy sets are used in both the antecedents and consequents of the rules, and a deffuzification procedure is utilized to obtain a crisp value from the output. The overall fuzzy output is calculated by applying "max" function to the fuzzy outputs of each rule. For final crisp output several approaches might be used like: center of area, bisector of area, mean of maxima, maximum criterion, etc. [30, 31].

Type 3: Sugeno Fuzzy Inference System

In type 3 systems Sugeno's fuzzy if-then rules are used [32]. The output of each rule is a linear combination of input variables plus a constant term, and the final output is the weighted average of each rule's output.

4.4. ANFIS (Adaptive Neuro- Fuzzy Inference System)

Fuzzy controller (FC) has various applications in both industry and household. For the complex (or ill-defined) systems that are not easily controlled by conventional control schemes, FC's provides a feasible alternative since they can easily capture the approximate, qualitative aspects of human knowledge and reasoning. The two important factors which restrict the FC's are the soundness of knowledge acquisition techniques and the availability of human expert. The attribute of fuzzy systems is expressing the knowledge in the form of linguistic rules, which can implement the expert human knowledge and the experience. But the shortcoming is the lack of a systematic methodology for their design. Normally, it is a time-consuming task that to update parameters of membership functions. The use of neural network learning techniques can make the process automatically, and significantly reduce the development time with a better performance. Consequently, the merger of neural networks and fuzzy logic led to the design of neuro-fuzzy controllers which are one of the most popular research fields today. Some architectures for neuro-fuzzy controllers are proposed for example by Jang [34], Abraham [35] and so on. In this section, the basics of adaptive networks, called ANFIS, is introduced in detail and it works as a fuzzy controller [36].

ANFIS, which is based on both neural networks and fuzzy inference systems, belongs to a class of Adaptive Fuzzy Inference System [37]. ANFIS can be also considered as a Neural Network close to Radial Basis Function. Its wide applications are including: the nonlinear function modeling [40, 27], the time series prediction [38, 39, 27], the online parameter identification for control systems [27], and the fuzzy controller design [34, 40]. The proposed ANFIS architecture can identify the near-optimal membership functions and other parameters of a rule base to acquire a desired input-output mapping, as it concerns the automatic elicitation of knowledge in the forms of fuzzy "*if-then*" rules. The basics of the ANFIS architecture are introduced in detail in the paper [34]. An application of ANFIS to the avoidance problem of the non-holonomic can be found in [41].

4.4.1 Architecture of ANFIS

ANFIS architecture for type 3 reasoning system is given in Figure 4.4 below: For simplicity, assume that the fuzzy inference system has two inputs *x* and *y* and one output *f*. Then, suppose that the rule base contains two fuzzy if- then rules of Takagi and Sugeno's type:

Rule 1: If *x* is A_1 and *y* is B_1, then $f_1 = p_1 x + q_1 y + r_1$,

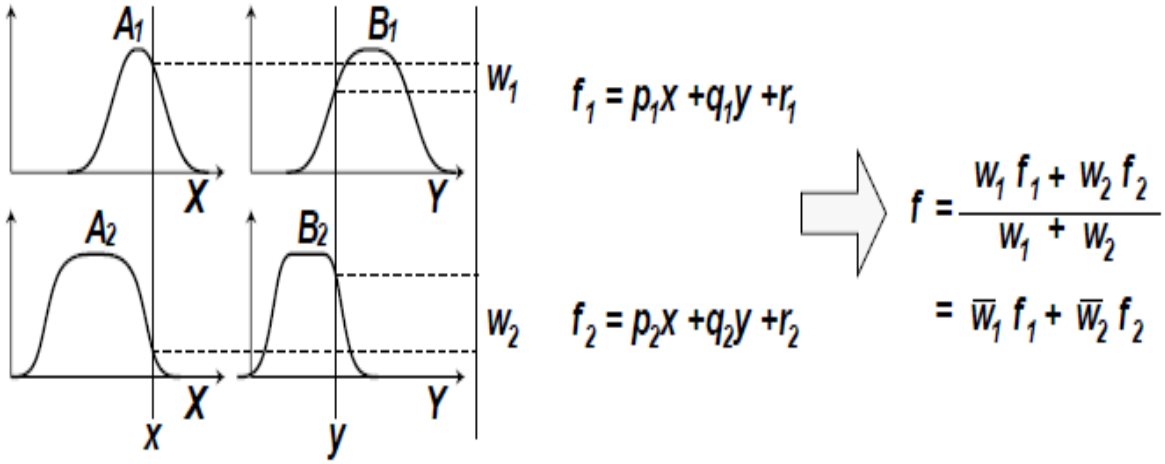
Rule 1: If *x* is A_2 and *y* is B_2, then $f_2 = p_2 x + q_2 y + r_2$.

Then the type- 3 fuzzy reasoning is illustrated in Figure 4.4(a), and the corresponding equivalent ANFIS architecture is shown in Figure 4.4(b), where nodes of the same layer have similar functions, as described next (Here the output of the i^{th} node in layer *l* is denoted as $O_{l,i}$).

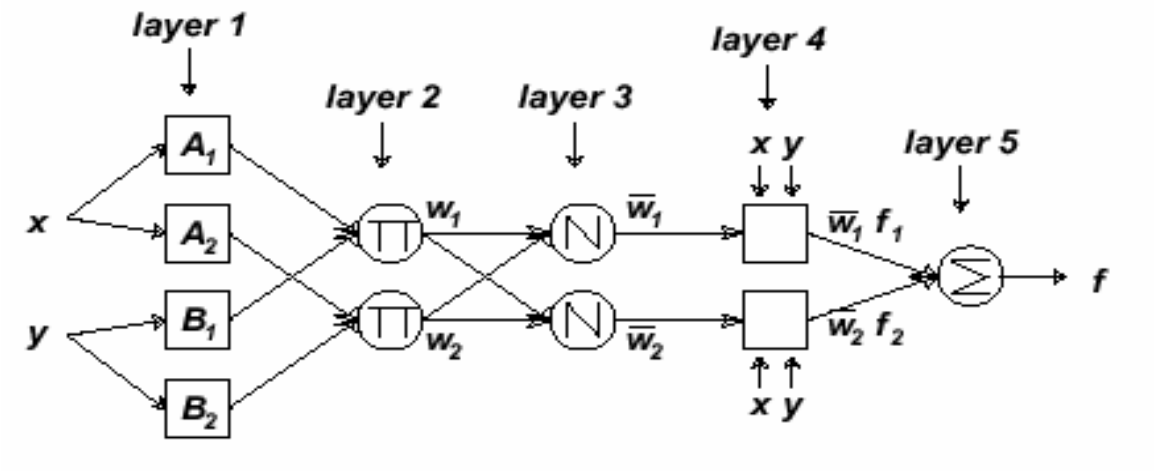
Layer 1: This layer represents the membership functions. Every node *i* in this layer is a square (or an adaptive) node with a node function:

$$O_{1,i} = \mu_{A_i}(x) \qquad (4.1)$$

Where *x* is the input to node *i* and A_i is the linguistic terms associated with this node function. In other words, $O_{1,i}$ is the membership function of $A_i(x)$ and it specifies the degrees to which the given *x* satisfies the quantifier A_i. After the fuzzification (see in Figure 4.4), we can get $A_1 \ldots A_i$ and $B_1 \ldots B_i$ respectively.



(a)



(b)

Figure 4.4 (a) Type-3 fuzzy reasoning; (b) equivalent type-3 ANFIS Architecture [27]

The Gaussian membership function is defined as:

$$\mu_i(x) = \exp\{-\frac{1}{2}\left(\frac{x-a_i}{b_i}\right)^2\} \qquad (4.2)$$

Where a_i and b_i are parameters of this node. As these parameters change the numerous membership functions can be represented. Any continuous and piecewise differentiable function, such as commonly used trapezoidal or triangular shaped membership functions, might be used in this layer as node functions. Parameters in this layer are referred to as premise parameters.

Layer 2: This layer represents the rules of the system. Every node in this layer is a circle node (has no parameters) labeled as *Π*. In these nodes incoming inputs are multiplied and result is sent as output. Each node output represents the firing strength of a rule. Then, use the fuzzy inference Π-norm to combine each pair of $\mu_{A_i}(x)$ and $\mu_{B_i}(y)$ to get the output of the nodes in this layer w_i.

$$O_{2,i} = w_i = \mu_{A_i}(x)\,\mu_{B_i}(y) \qquad i = 1, 2 \qquad (4.3)$$

But, they are not the real outputs, they must be defuzzified to obtain the real output *f* at last.

Layer 3: This layer calculates the ratio of a rule's firing strength to the sum of all rules' firing strengths. Every node in this layer is a circle node labeled as *N*. The output of this layer is called normalized firing strengths and calculated with:

$$O_{3,i} = \bar{w}_i = \frac{w_i}{w_1 + w_2} \qquad i = 1, 2 \qquad (4.4)$$

Layer 4: This layer calculates the output of each rule. Every node in this layer is a square node with a node function:

$$O_{4,i} = \bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i) \quad (4.5)$$

Where $\bar{w}_i$ is the output of third layer and $\{p_i, q_i, r_i\}$ are the parameters of the node. Parameters in this layer are called consequent parameters.

Layer 5: Node in this layer combines the output of each rule with vector summation. This single node is a circle node labeled as $\sum$ that computes the overall output as the summation of all incoming signals. The output of this neural network is given by the following equation:

$$O_{5,1} = f = \sum_i \bar{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad (4.6)$$

4.4.2 Hybrid Learning Algorithm of ANFIS

From the proposed type-3 ANFIS architecture (Figure 4.4b), it is observed that given the values of premise parameters, the overall output can be expressed as a linear combinations of the consequent parameters. More precisely, the output *f* in Figure 4.4 (b) can be rewritten as:

$$\begin{aligned} f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\ &= \bar{w}_1 (p_1 x + q_1 y + r_1) + \bar{w}_2 (p_2 x + q_2 y + r_2) \quad (4.7) \\ &= (\bar{w}_1 x) p_1 + (\bar{w}_1 y) q_1 + (\bar{w}_1) r_1 + (\bar{w}_2 x) p_2 + (\bar{w}_2 y) q_2 + (\bar{w}_2) r_2 \end{aligned}$$

Which is linear in the consequent parameters (p_1, q_1, r_1, p_2, q_2 *and* r_2). From this observation, let:

S = the total parameters in premise and consequent parts.

S_1 = the parameters in premise (nonlinear) part.

S_2 = the parameters in consequent (linear) part.

In ANFIS hybrid learning which combines the gradient descent method and least square estimate (LSE) is used to identify parameters. The ANFIS network that has only one output can be represented as:

$$Output = F(\vec{I}, S) \quad (4.8)$$

Where $\vec{I}$ represents the vector of input variables, *S* represents the parameters of the network, and *F* represents the overall function implemented by the adaptive network. If there exists a function

H such that composite function $H \circ F$ is linear in some of the elements of S, then these elements can be found by using least squares method. If the parameter set S can be divided into two sets S_1 and S_2 such that $H \circ F$ is linear in the elements of S_2, then after applying H to Equation (4.8), we get the following, which is linear in S_2:

$$H\,(Output) = H \circ F\,(\vec{I}\,, S) \qquad (4.9)$$

Now, given values of elements of S_1, training data P can be added to the Equation (4.9) and the following matrix equation is obtained:

$$AX = B \qquad (4.10)$$

Where X is an unknown vector whose elements are parameters in S_2. If dimension of S_2 is M then dimension of A is PxM, dimension of B is Px1 and dimension of X is Mx1. Since number of training data (P) is greater than number of linear parameters (M), this is an over determined problem and there is no exact solution. Instead, LSE of X, X^* can be found. The most well-known formula for X^* is:

$$X^* = (A^T A)^{-1} A^T B \qquad (4.11)$$

Where A^T is the transpose of A and $(A^T A)^{-1} A^T$ is pseudo inverse of A if $A^T A$ is nonsingular. Because the Equation (4.11) is expensive in computation and it becomes a problem when $A^T A$ is singular, sequential method of LSE used is given in the following equation:

$$X_{i+1} = X_i + S_{i+1}\, a_{i+1}\, (b_{i+1}^T - a_{i+1}^T X_i) \qquad (4.12)$$

$$S_{i+1} = S_i - \frac{S_i\, a_{i+1}\, a_{i+1}^T\, S_i}{1 + a_{i+1}^T\, S_i\, a_{i+1}} \qquad i = 0, 1, \ldots, P\text{-}1 \qquad (4.13)$$

where a_i^T is the i^{th} row vector of matrix A, b_i^T is the i^{th} element of matrix B. S_i is covariance matrix and least square estimate X^* is X_P. Initially $X_0 = 0$ and $S_0 = \gamma I$; where I is identity matrix with dimension M x M and γ is a positive large number.

In ANFIS hybrid-learning procedure is composed of a forward pass and a backward pass. In the forward pass, input data is given to network and functional signals go forward to calculate each node output until the A and B given in the Equation (4.10) is obtained and then the parameters in S_2 are identified by the sequential least squares formulas in Equations (4.12) and (4.13). After parameters in S_2 are identified, the functional signals go forward and the output is calculated.

With using the actual and target output values the error is calculated. In the backward pass the error is propagated from output end to input end and parameters in S_1 is updated by gradient descent method.

Assuming the training data set has P entries, the error measure can be defined for the p^{th} ($1 \leq p \leq P$) entry of the training data as the sum of squared errors:

$$E_p = \sum (T_p - O_p)^2 \qquad (4.14)$$

Where T_p is the target (desired) output and O_p is the actual output.
Then, the overall error measure is:

$$E = \sum_{p=1}^{P} E_p \qquad (4.15)$$

To implement gradient descent in E over parameter space, firstly the error rate $\frac{\partial E_p}{\partial O}$ for the p^{th} training data and for each node output must be calculated. The error rate for the output node at layer L can be calculated from Equation (4.14):

$$\frac{\partial E_p}{\partial O} = -2\,(T_p - O_p^L) \qquad (4.16)$$

For the internal node at (k, i) layer k index i, error rate can be derived by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{k+1} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_p^k} \qquad (4.17)$$

Where $1 \leq k \leq L-1$. That is the error rate of an internal node can be expressed as a linear combination of the error rates of the nodes in the next layer. Therefore, for all $1 \leq k \leq L$ *and* $1 \leq i \leq k$, $\frac{\partial E_p}{\partial O_{i,p}^k}$ can be found by using Equations (4.16) and (4.17).

If α is a parameter of the given adaptive network, then

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O \in S} \frac{\partial E_p}{\partial O} \frac{\partial O}{\partial \alpha} \qquad (4.18)$$

Where S is the set of nodes whose output depends on α. Then the derivative of the overall error measure E with respect to α is,

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha} \qquad (4.19)$$

Accordingly, the update formula for the parameter α is:

$$\Delta\alpha = - \eta \frac{\partial E}{\partial \alpha} \quad (4.20)$$

In which η is a learning rate which can be expressed as:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha}\left(\frac{\partial E}{\partial \alpha}\right)^2}} \quad (4.21)$$

Where *k* is step size, the length of each gradient transition in the parameter space. The value of *k* change to vary speed of convergence.

4.4.3 Parameters Updating Algorithm of ANFIS

Now, based on ANFIS architecture and hybrid system, define *S* as the set of all parameters to adapt in the neural network:

$$S = a_1, \ldots, a_i, b_1, \ldots, b_i, f_1, \ldots, f_i \quad (4.22)$$

And *V(s)* the function to minimize using LSE will be:

$$V(s) = \frac{1}{2}\left(f_d\,(t) - f(t)\right)^2 \quad (4.23)$$

Where $f_d\,(t)$ is the desired output and $f(t)$ is the actual output of the neural network. In this case, Godjevac [42] showed it as possible to use an iterative procedure to update parameters in order to minimize the function *V(s)*. The three kinds of parameters a_i, b_i and f_i may be updated using hybrid combination of back propagation training algorithm that's based on gradient descent method and forward propagation training algorithm that's based on LSE by the following equations:

$$a_i(t+1) = a_i(t) - \Gamma_a \frac{\mu_i}{\sum_{i=1}^{m}\mu_i}\left(f_d - f\right)\left(f_i - f\right)\frac{(x-a_i\,(t))}{b^2{}_i(t)} \quad (4.24)$$

$$b_i(t+1) = b_i(t) - \Gamma_b \frac{\mu_i}{\sum_{i=1}^{m}\mu_i}\left(f_d - f\right)\left(f_i - f\right)\frac{(x-a_i\,(t))^2}{b^3{}_i\,(t)} \quad (4.25)$$

$$f_i(t+1) = f_i(t) - \Gamma_f \frac{\mu_i}{\sum_{i=1}^{m}\mu_i}\left(f_d - f\right) \quad (4.26)$$

Where a_i, b_i, and f_i are the parameters of the adaptation of the learning algorithm. Γ_a, Γ_b *and* Γ_f are the predefined constants. The iterative procedure for the adaptation of parameters and for the minimization of the criterion function can be summarized as follows [42]:

1. *Initialization of parameters.*

a) Consequent values fi, are random numbers.

b) Choice of premise parameters a_i and b_i *(all membership functions on the universe of discourse are regularly distributed and have the same width).*

2. *Data input* (x, y, f_d).
3. *Fuzzy inference.*
4. *Adaptation of consequent parameters* f_i.
5. *Adaptation of parameters* a_i and b_i.
6. *Evaluation of the criterion function V(s).*
7. *Repeat the steps 3 to 6 until V is smaller than one threshold value.*

The algorithm is more efficient if the steps 3 and 4 are repeated twice in the same iteration loop. Namely, the weights *fi* are adapted twice in one iteration [43].

4.5 ANFIS Controller for non-holonomic WMR

The purpose of control is to input appropriate wheel speeds (both left and right), to keep the robot track with the desired given reference. That is, to make the difference between the robot's track $P(x(t), y(t), \theta(t))$ and reference $P_d(x_d(t), y_d(t), \theta_d(t))$ is converged to zero (0).

It is difficult to obtain the accurate parameters from robot in the real. Therefore, a new approach based on adaptive neuro-fuzzy and depending on the kinematic models (3.8), (3.9), (3.14) and (3.15) is proposed. The goal of the neuro-fuzzy is to control the velocity of each wheel in order to minimize the two errors, the error between desired position and actual position ($x_d - x$, $y_d - y$) and the error between desired orientation and actual orientation ($\theta_d - \theta$).

4.5.1 Position Control

The position control allows the robot to follow the target point ($x_d(t)$, $y_d(t)$) with a desired path. The crucial thing is to make robots reach to a given position from the initial position to the final one without changing its orientation. The two input variables e_x and e_y are the differences between the robot's position and the position's coordinates along with *X-axis* and *Y-axis* respectively at each time step, which express as (4.27) and (4.28). Both of the two wheels must have the same angular velocity so as to drive the robot to go forwards. Therefore, the speed of the left wheel must be equal with the speed of the right wheel.

The position control block diagram is as follow:

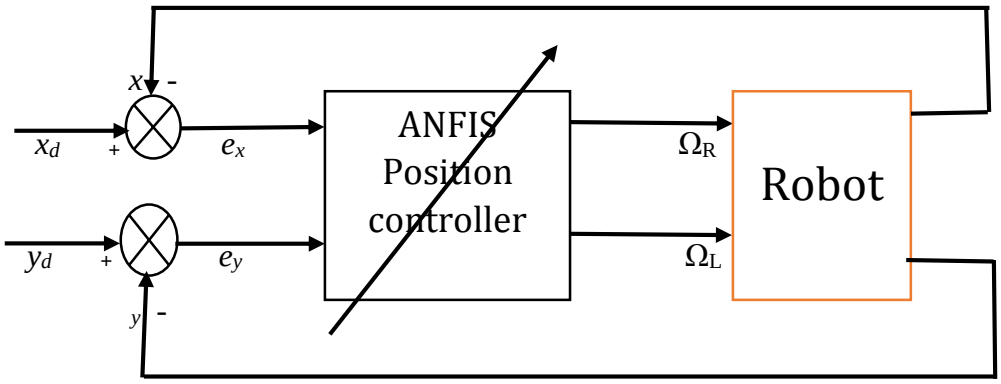


Figure 4.5: Position control block diagram

In the position control, the neural network needs two inputs e_x and e_y which are given by equations (4.27) and (4.28) respectively. The position control diagram is shown in Figure 4.5 and the single output is expressed in equation (4.29).

$$e_x(t) = x_d(t) - x(t) \qquad (4.27)$$

$$e_y(t) = y_d(t) - y(t) \qquad (4.28)$$

Where

$x(t)$ and $y(t)$ correspond to the coordinates of the robot;

$x_d(t)$ and $y_d(t)$ a correspond to the desired coordinates of the robot.

The neuro fuzzy have only one output $f_p(t)$:

$$f_p(t) = \frac{\sum_{i=1}^{m} w_i^p f_i^p}{\sum_{i=1}^{m} w_i^p} \qquad (4.29)$$

4.5.2 Orientation Control

Similar as the position control, the orientation control allows the robot to rotate around itself by following the target angle. Consequently, the ANFIS needs one input e_θ which is the difference between the robot's desired angle θ_d and the robot's directions θ at each time step. By the fuzzy inference, the neural networks and the defuzzification, the single output is an angular velocity. The input is showed in equation (4.30) and the output is showed in equation (4.31).

The orientation control block diagram is described as follow:

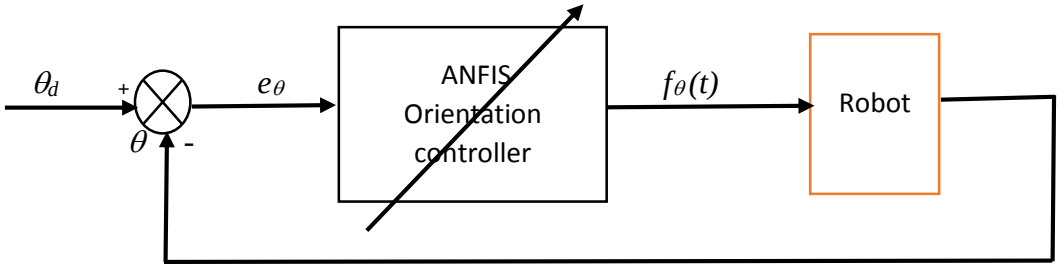


Figure 4.6: Orientation control block diagram

The loss function and parameters update are expressed in equation (4.32). The diagram of control has showed in Figure 4.6.

$$e_\theta(t) = \theta_d(t) - \theta(t) \qquad (4.30)$$

$$f_\theta(t) = \frac{\sum_{i=1}^{m} w_i^\theta f_i^\theta}{\sum_{i=1}^{m} w_i^\theta} \qquad (4.31)$$

At each step time, the parameters f_i^θ are updated in order to minimize the following equation:

$$V_\theta(t) = (\theta_d(t) - \theta(t))^2 \qquad (4.32)$$

4.5.3 Trajectory Control strategy

Generally, the control of wheeled mobile robot consists in doing a follow of reference path and supposes to measure both the position and orientation with respect to a fixed frame. This approach is composed of two neuro-fuzzy controller, both position and orientation control, allowing to track these desired trajectories.

If we combine the position's control with the orientation's control, we will get the trajectory's control which can make the robot follow a desired trajectory. Inputs e_x, e_y and e_θ are the differences between the real position of the robot given by $P = (x, y, \theta)$ and the desired position $P_d = (x_d, y_d, \theta_d)$. The linguistic variable "distance" is obtained by using equation (4.33).

$$distance = \sqrt{e_x^2 + e_y^2} \qquad (4.33)$$

The linguistic variable "alpha" which is the angle of orientation that the MR forms between the straight line of actual and target position is calculated by using equation (4.34).

$$alpha = \tan^{-1}\left(\frac{e_y}{e_x}\right) - \theta \qquad (4.34)$$

v and w are given by ANFIS position control, and w is given by ANFIS orientation control. In this case, the angular velocity of two wheels (Ω_R and Ω_L) are given by equations (4.35) and (4.36).

$$\Omega_R = \frac{1}{R_a}(v + \mathrm{L}\, w) \qquad (4.35)$$

$$\Omega_L = \frac{1}{R_a}(v - \mathrm{L}\, w) \qquad (4.36)$$

The following diagram show the global trajectory control:

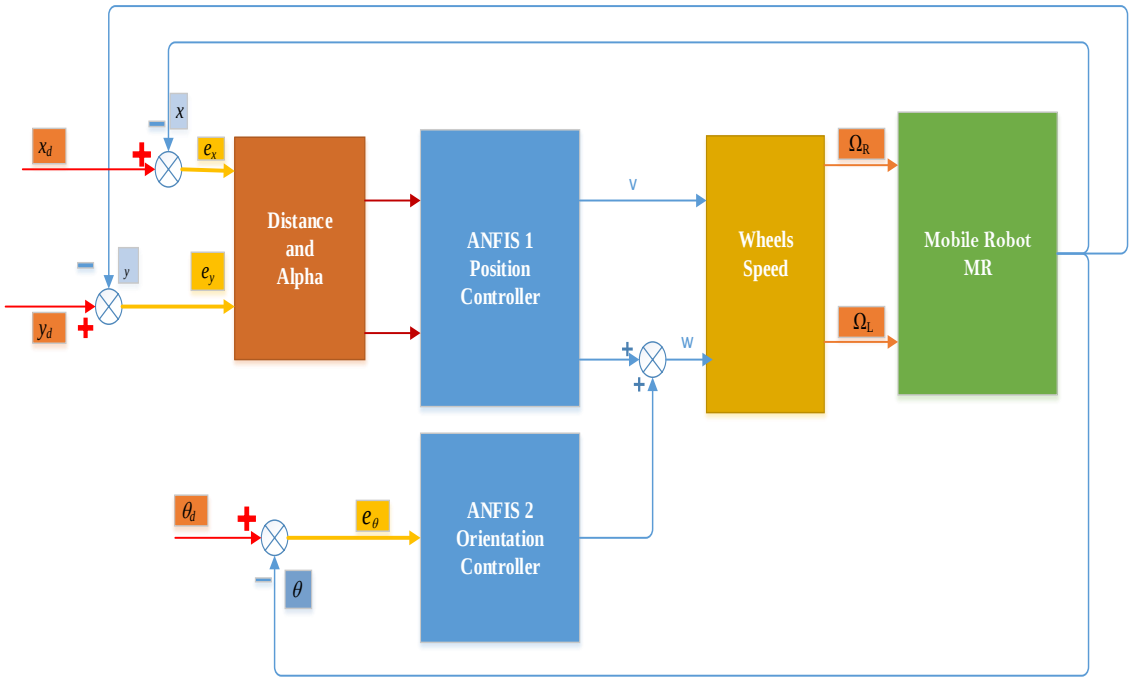


Figure 4.7: Trajectory control block diagram: Two neuro-fuzzy controllers for both the position's control and the orientation's control.

The algorithm implemented in the ANFIS controller block for pose regulation has two steps synchronized with the help of the summing junction (⊗) that is shown in Figure 4.7. The steps are:

Step 1: Using ANFIS1, the MR is oriented from the actual position and orientation to the target position regardless final orientation, in this step the MR rotates the α ($alpha$) angle and it moves to the target position.

Step 2: Once the MR has reached the target position, using ANFIS2 the MR is oriented according the target pose.

CHAPTER 5

SIMULATION RESULTS AND DISCUSSIONS

5.1 Introduction

In this section, the simulation results about the problem described in the previous sections are presented. Simulation has been recognized as an important research tool since the beginning of the 20^{th} century and now the simulation is a powerful visualization, planning, and strategic tool in different areas of research and development. The simulation has also a very important role in robotics. The controller has been designed with the software MATLAB/ Simulink version R2017a. Simulation has been performed by using the virtual robot Pioneer 3-DX. Knowing the coordinates of the robot, the current navigational controller can thus calculate the distances and heading angle of the robot.

5.2 ANFIS Based Trajectory Tracking

The acronym ANFIS derives its name from *adaptive neuro-fuzzy inference system*. Using a given input/output data set, the toolbox function *anfis* constructs Sugeno-type fuzzy inference system (FIS) whose membership function parameters are tuned (adjusted) using a back propagation algorithm alone or in combination with a least squares type of method. This adjustment allows fuzzy systems to learn from the data they are modeling.

The Neuro-Fuzzy Designer includes four distinct areas to support a typical workflow. This application helps to perform the following tasks:

1. Loading, Plotting, and Clearing the Data
2. Generating or Loading the Initial FIS Structure
3. Training the FIS
4. Validating the Trained FIS

The Neuro-Fuzzy Designer Structure is given in Figure 5.1 below.

Loading, Plotting, and Clearing the Data

To train an FIS, it must begin by loading a Training data set that contains the desired input/output data of the system to be modeled. Any data set that is to be loaded must be an array with the data arranged as column vectors, and the output data in the last column.

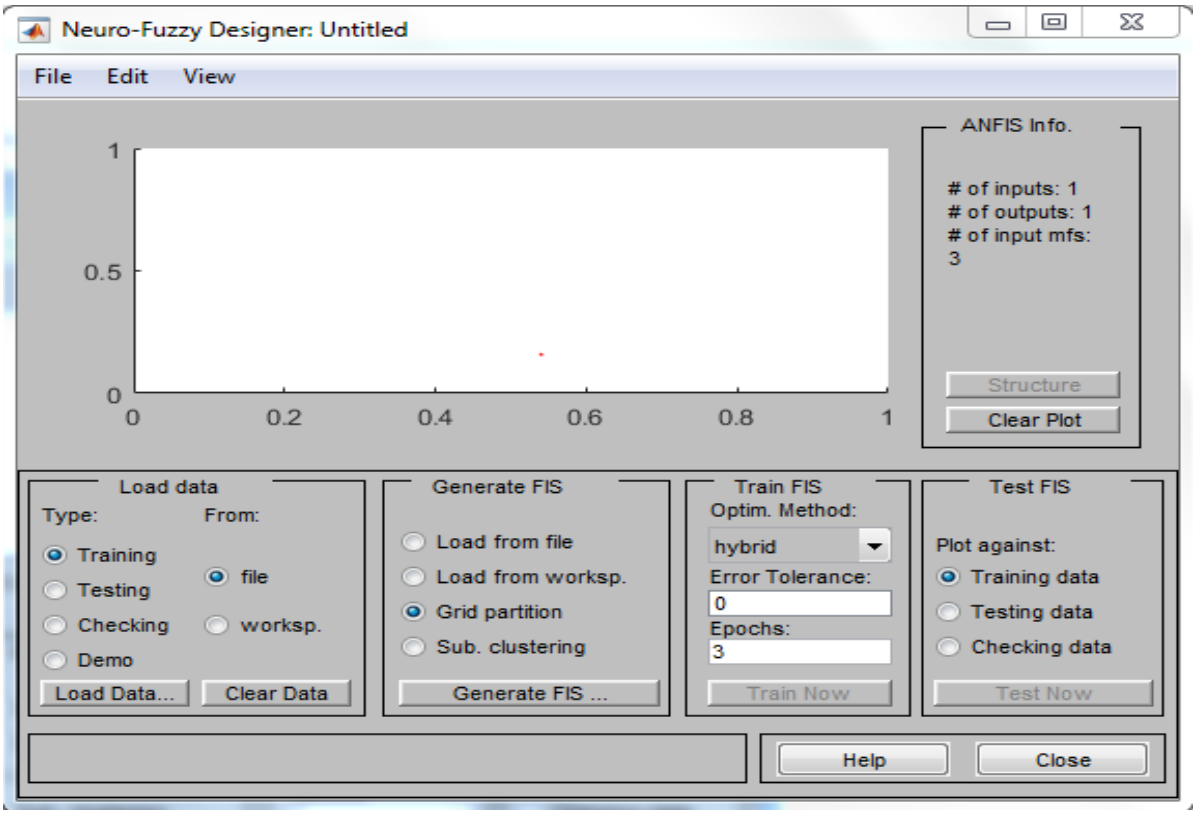


Figure 5.1: The Neuro-Fuzzy Designer Structure.

To load a data set using the **Load Data** portion of the designer:

1. Specify the data **Type**.
2. Select the data from a **file** or the MATLAB **workshop**.
3. Click **Load Data**.

After the data is loaded, it displays in the plot. The training, testing and checking data are annotated in blue as *circles*, *diamonds*, and *pluses* respectively.

To clear a specific data set from the designer:

1. In the **Load Data** area, select the data **Type**.
2. Click **Clear Data**.

This action also removes the corresponding data from the plot.

Generating or Loading the Initial FIS Structure

Before to start the FIS training, an initial FIS model structure must be specified. To specify the model structure, perform one of the following tasks:

- Load a previously saved Sugeno-type FIS structure from a **file** or the MATLAB **workspace.**
- Generate the initial FIS model by choosing one of the following partitioning techniques:
  - ✓ Grid partitioning: - Generates a single-output Sugeno-type FIS by using grid partitioning on the data.
  - ✓ Sub.clustering:- Generates an initial model for ANFIS training by first applying subtractive clustering on the data.

To view a graphical representation of the initial FIS structure, click **Structure**.

Training the FIS

After loading the training data and generating the initial FIS structure, start training the FIS. The following steps show how to train the FIS.

1. In **Optim. Method**, choose **hybrid** or **backpropaga** as the optimization method. The optimization methods train the membership function parameters to emulate the training data.

   Note: The hybrid optimization method is a combination of least-squares and back propagation gradient descent method.

2. Enter the number of training **Epochs** and the training **Error Tolerance** to set the stopping criteria for training.
   The training process stops whenever the maximum epoch number is reached or the training error goal is achieved.

3. Click **Train Now** to train the FIS. This action adjusts the membership function parameters and displays the error plots.

Validating the Trained FIS

After the FIS is trained, validate the model using a **Testing** or **Checking** data that differs from the one used to train the FIS. To validate the trained FIS:

1. Select the validation data set and click **Load Data**.

2. Click **Test Now**.
   This action plots the test data against the FIS output (shown in red) in the plot.

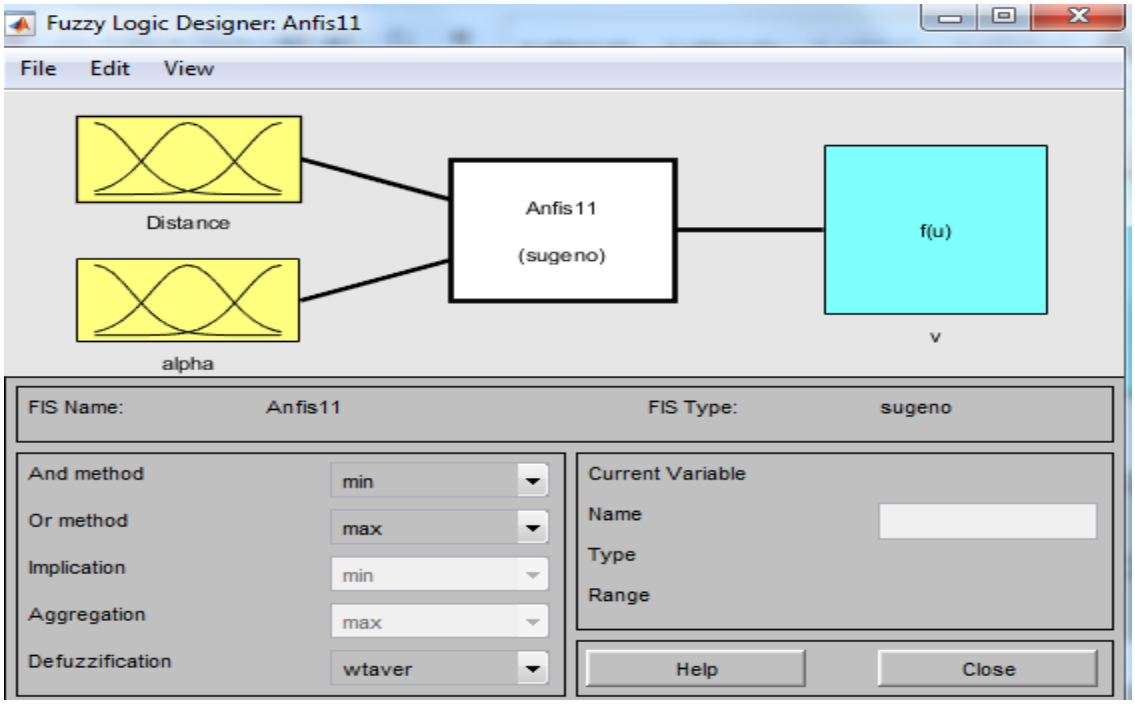


Figure 5.2: Anfis11 Position Controller Structure.

Figure 5.2 shows the position controller Anfis11 has two linguistic variables as inputs: the "*distance"* from the actual position to the target position and the "*alpha"* which represents the angle of deviation of the actual position to the target position. And it has one linguistic variables as outputs: linear speed "*v*". The model structure view, their membership functions, fuzzy *if-then* rule base, fuzzy rule view and surface view are shown in Figures below as follows:

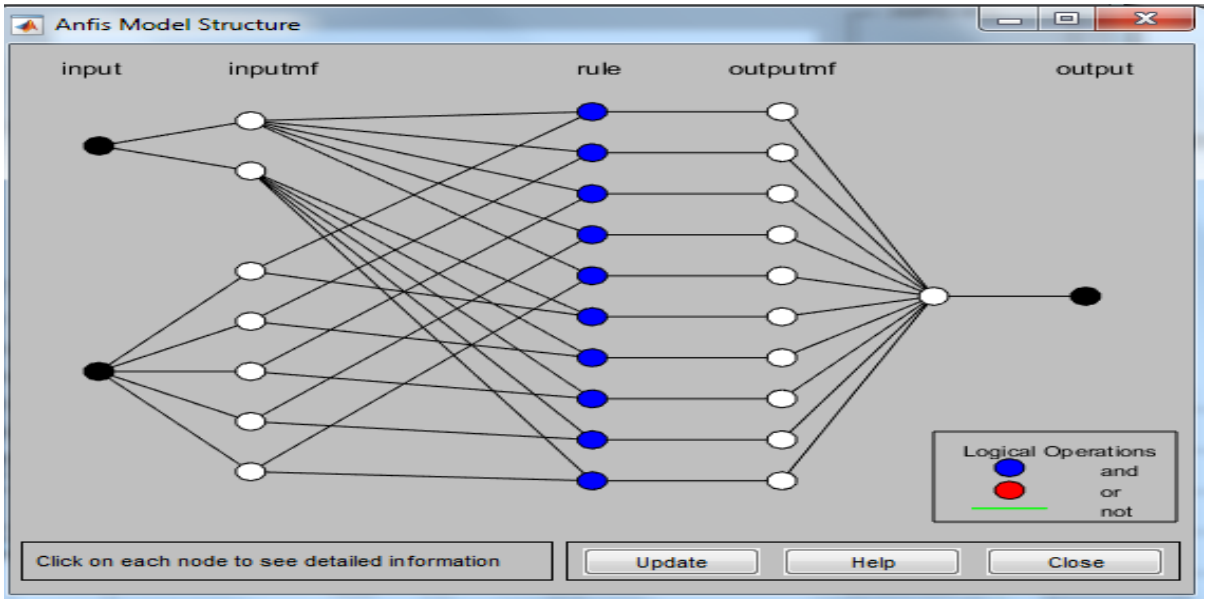


Figure 5.3: Anfis11 Position Controller Model Structure View.

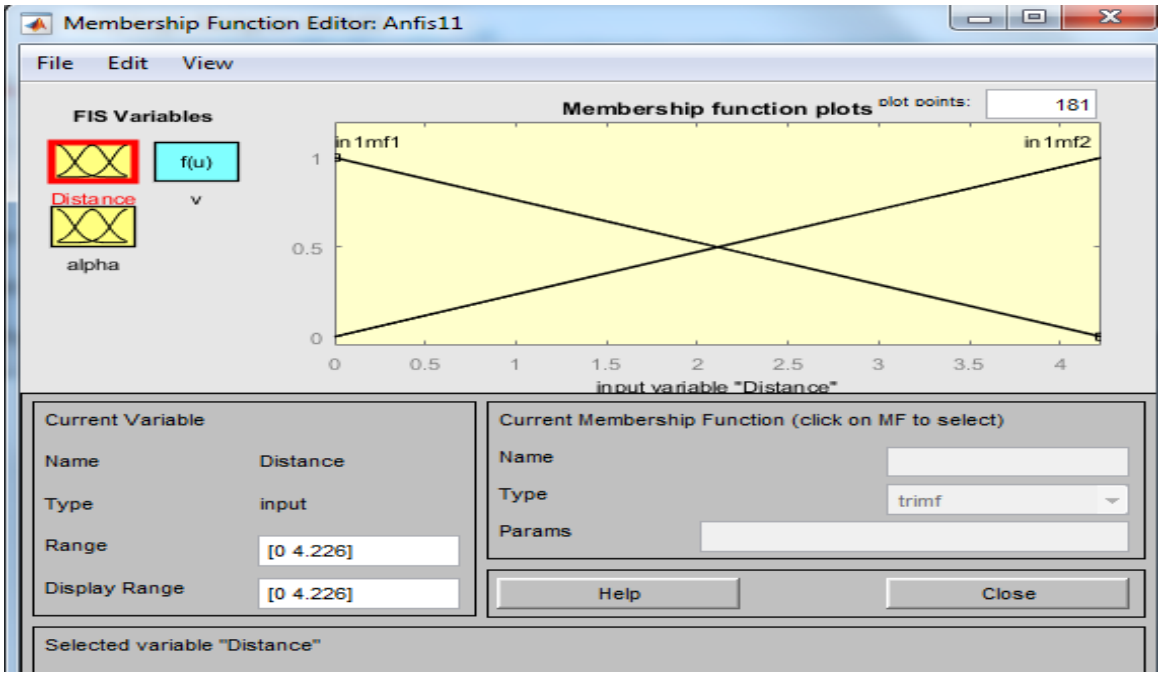


Figure 5.4a: Membership function for input linguistic variable "*distance*".

Figure 5.4a shows the input linguistic variable "*distance*" with its linguistic terms: mf1 and mf2 in the universe of discourse [0, 4.226].

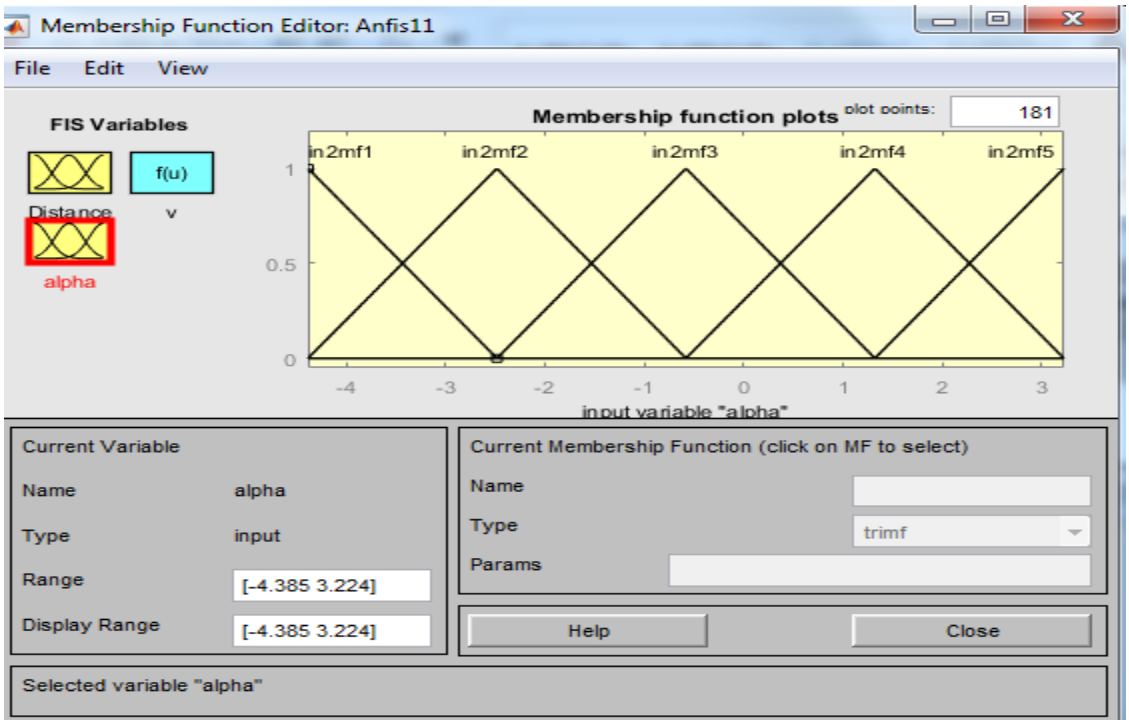


Figure 5.4b: Membership function for input linguistic variable "*alpha*".

Figure 5.4b shows the input linguistic variable "*alpha*" with its linguistic terms: mf1, mf2, mf3, mf4 and mf5 in the interval [-4.385, 3.224].

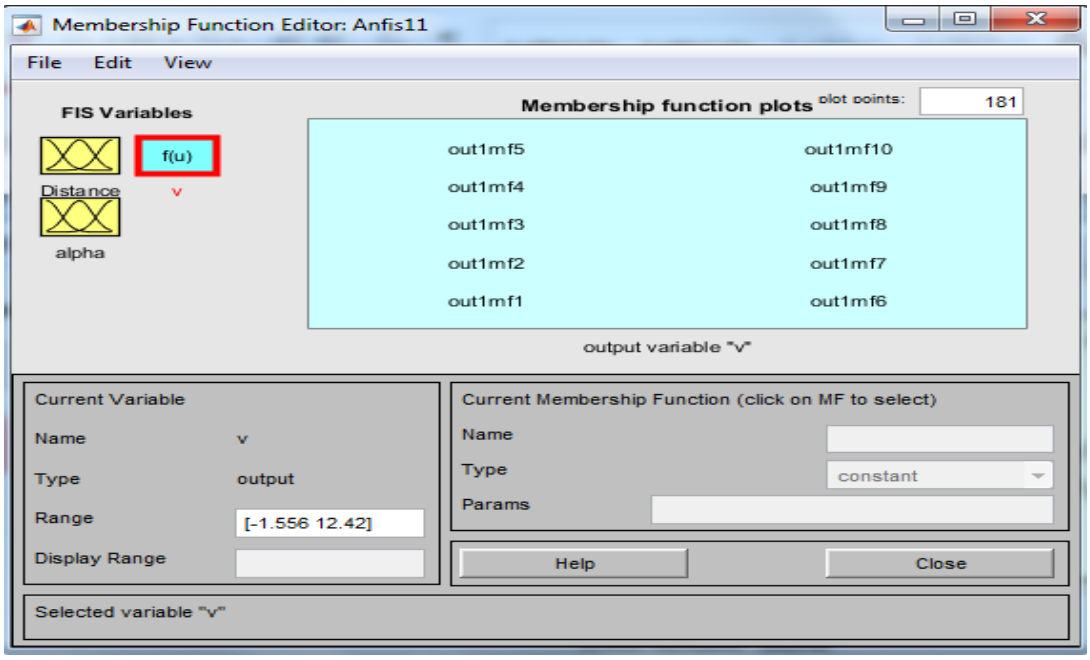


Figure 5.4c: Membership function for output linguistic variable "*v*".

Figure 5.4: Anfis11 membership function for position controller.

Figure 5.4c shows the output linguistic variable linear speed "*v*" has ten (10) linguistic terms in the interval [-1.556 12.42].

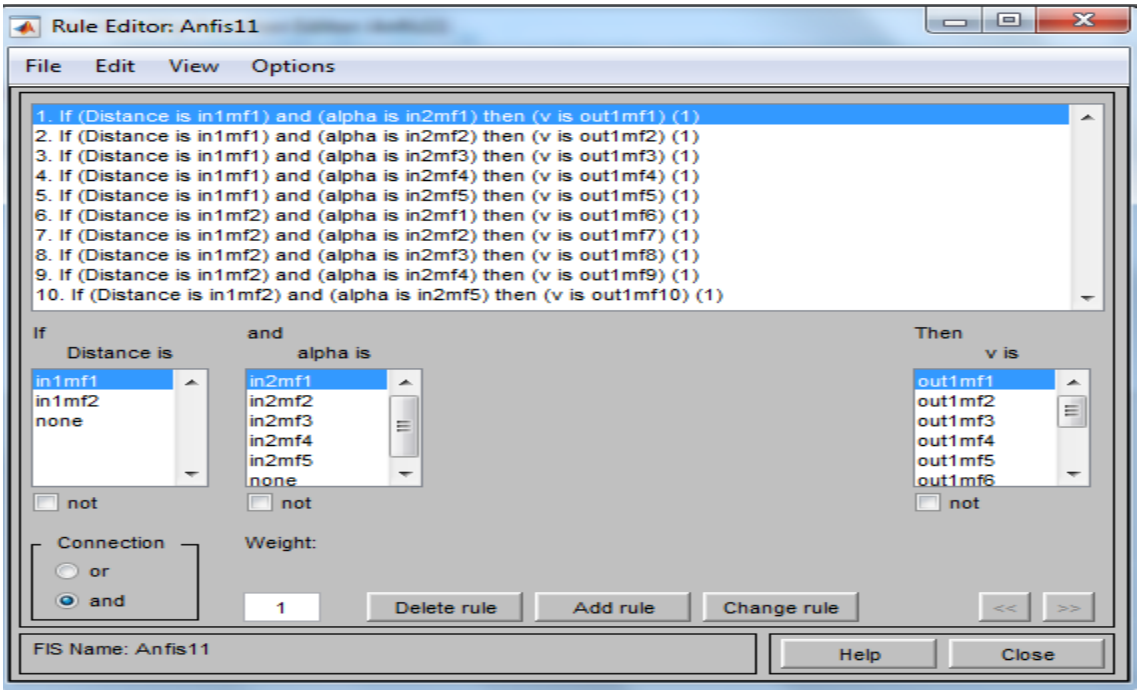


Figure 5.5: Anfis11 position controller rule.

Figure 5.5 shows Anfis11 position controller has 10 *if-then* rule base and the fuzzy rule view is shown in Figure 5.6 with their correspondence value.

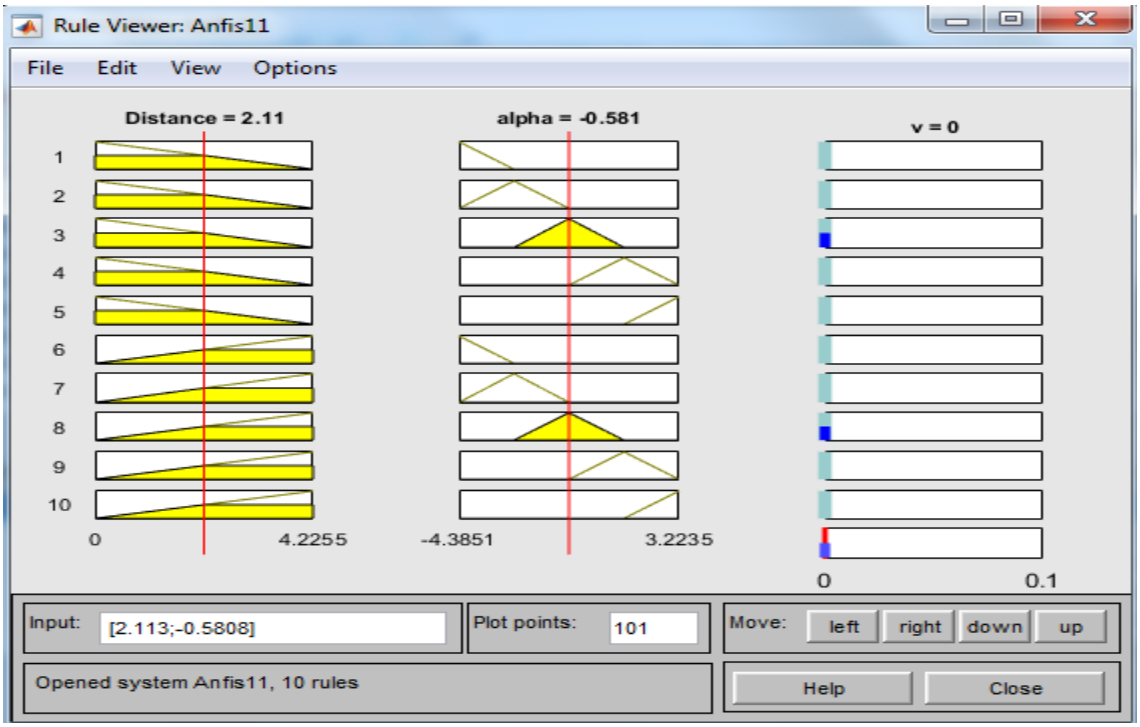


Figure 5.6: Anfis11 position controller rule view.

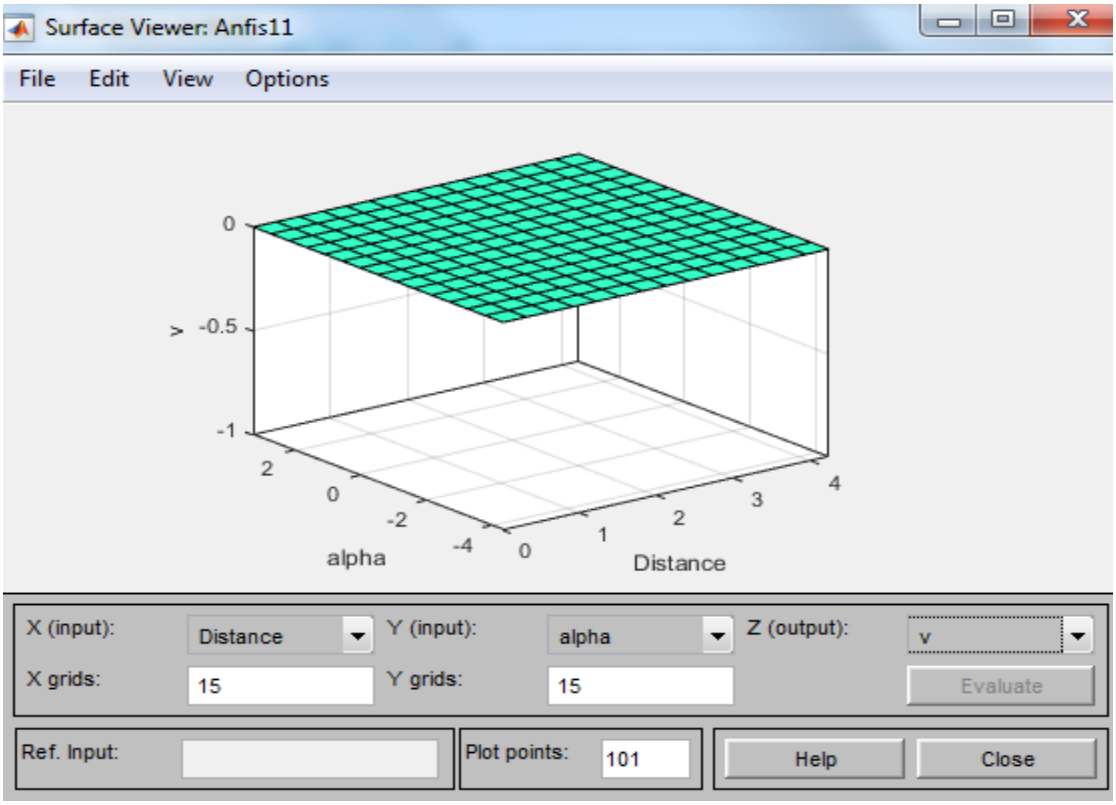


Figure 5.7: Anfis11 position controller surface view.

Figure 5.7 shows the 3D surface view of Anfis11 position controller in *Distance*, *alpha*, and *v*.

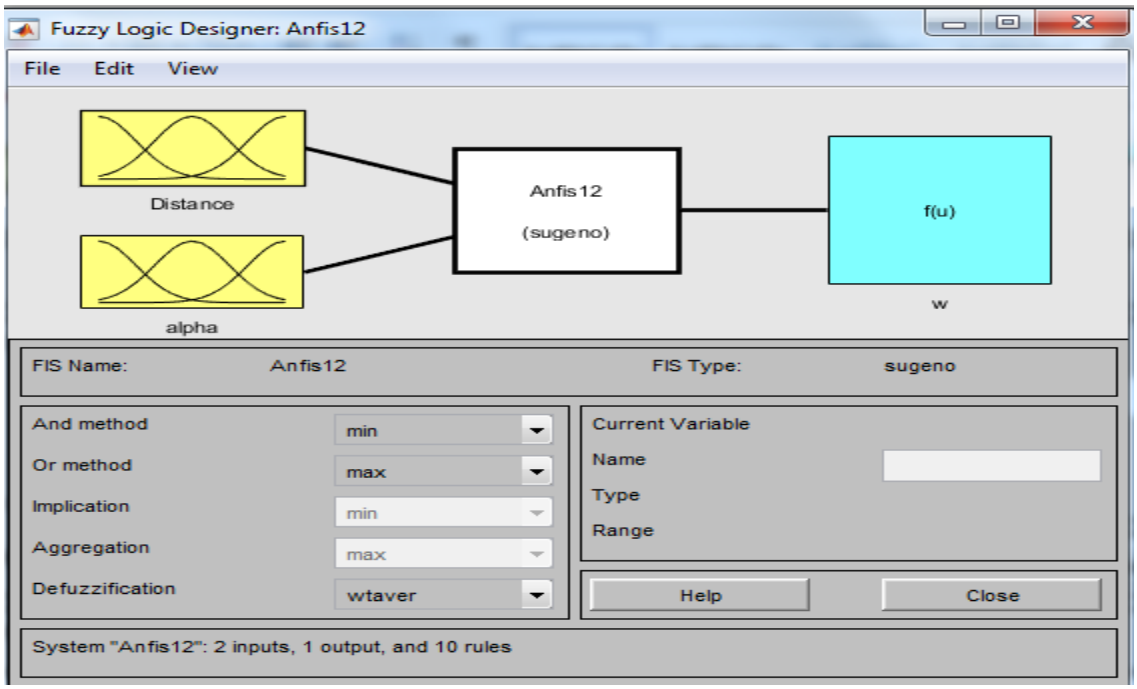


Figure 5.8: Anfis12 Position Controller Structure.

Figure 5.8 shows the position controller Anfis12 has two linguistic variables as inputs: the "*distance"* from the actual position to the target position and the "*alpha"* which represents the angle of deviation of the actual position to the target position. And it has one linguistic variables as outputs: angular speed "*w*". The model structure view, their membership functions, fuzzy *if-then* rule base, fuzzy rule view and surface view are shown in Figures below as follows:

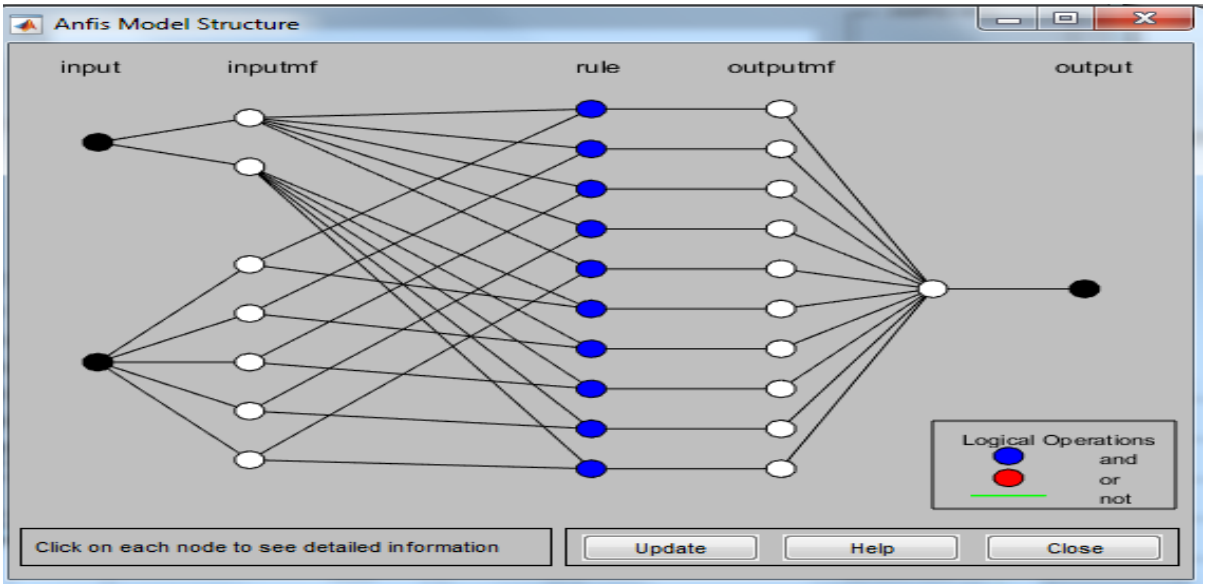


Figure 5.9: Anfis12 Position Controller Model Structure View.

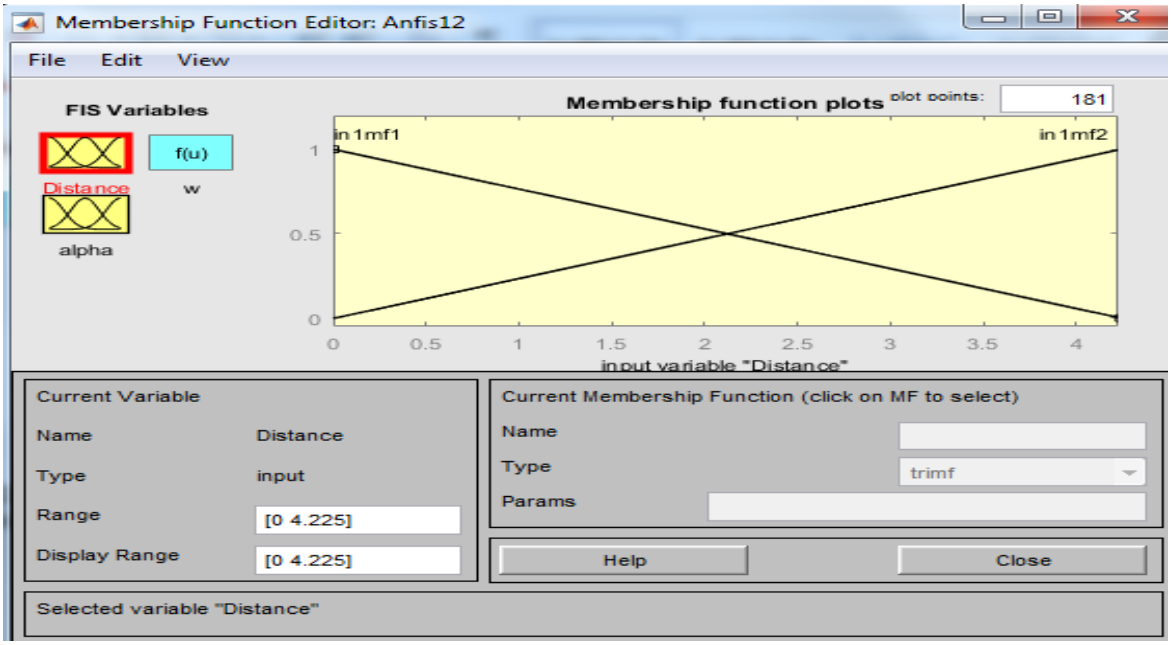


Figure 5.10a: Membership function for input linguistic variable "*Distance*".

Figure 5.10a shows the input linguistic variable "*distance*" with its linguistic terms: mf1 and mf2 in the universe of discourse [0, 4.225].

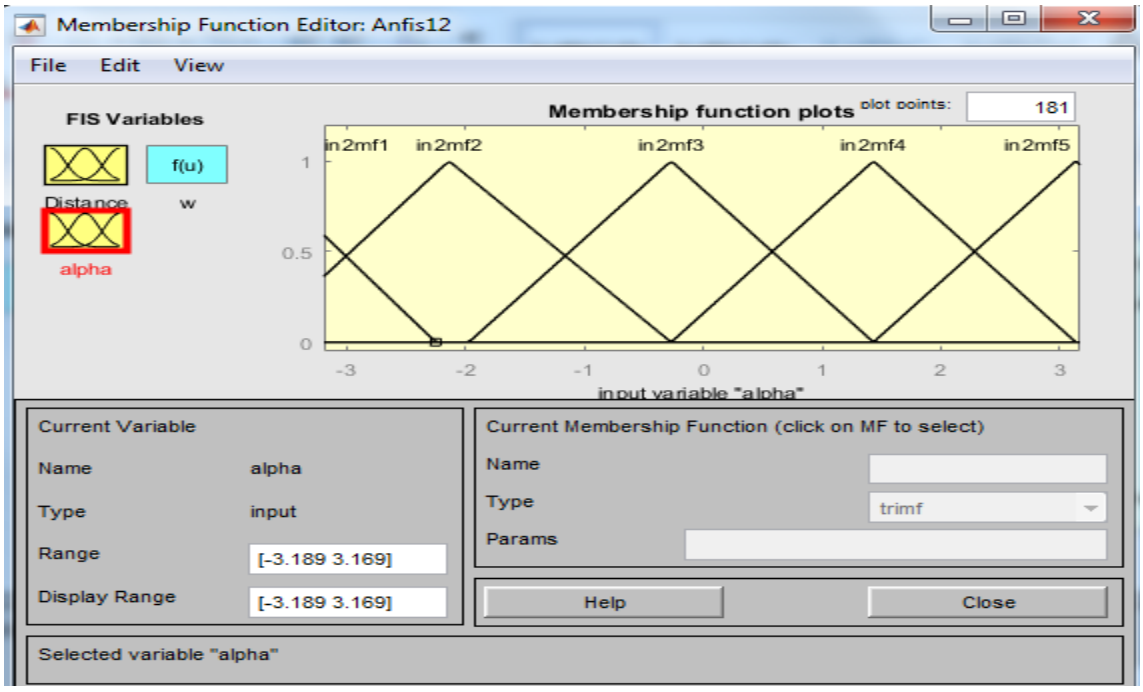


Figure 5.10b: Membership function for input linguistic variable "*alpha*".

Figure 5.10b shows the input linguistic variable "*alpha*" with its linguistic terms: mf1, mf2, mf3, mf4 and mf5 in the interval [-3.189, 3.169].

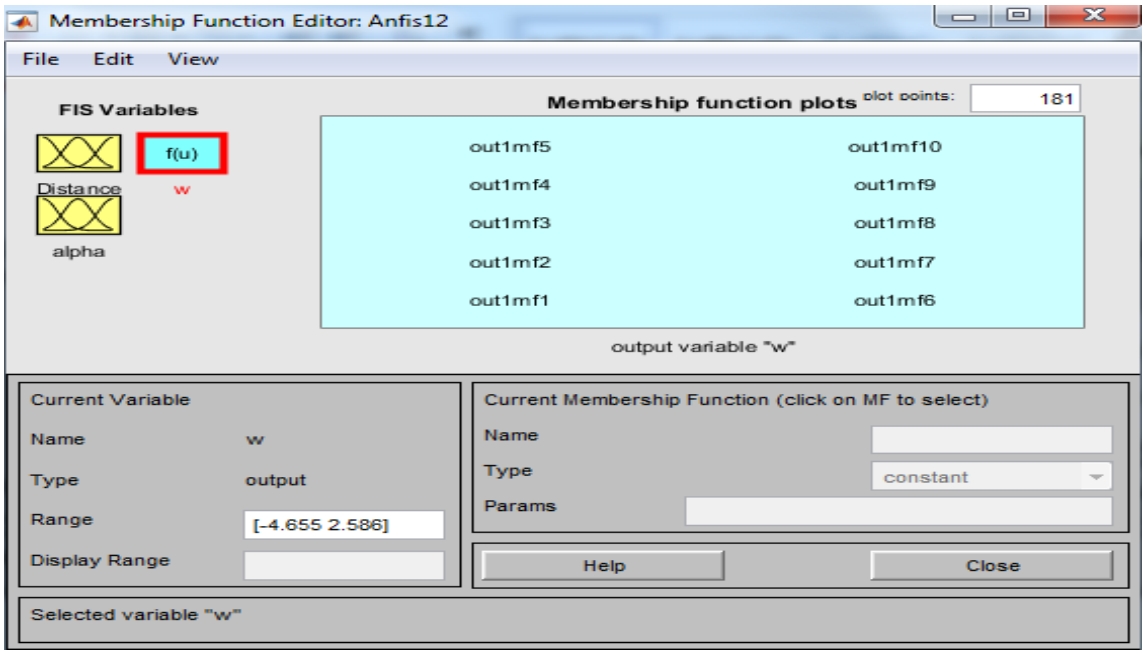


Figure 5.10c: Membership function for output linguistic variable "*w*".

Figure 5.10: Anfis12 membership function for position controller.

Figure 5.10c shows the output linguistic variable angular speed "*w*" has ten (10) linguistic terms in the interval [-4.655 2.586].

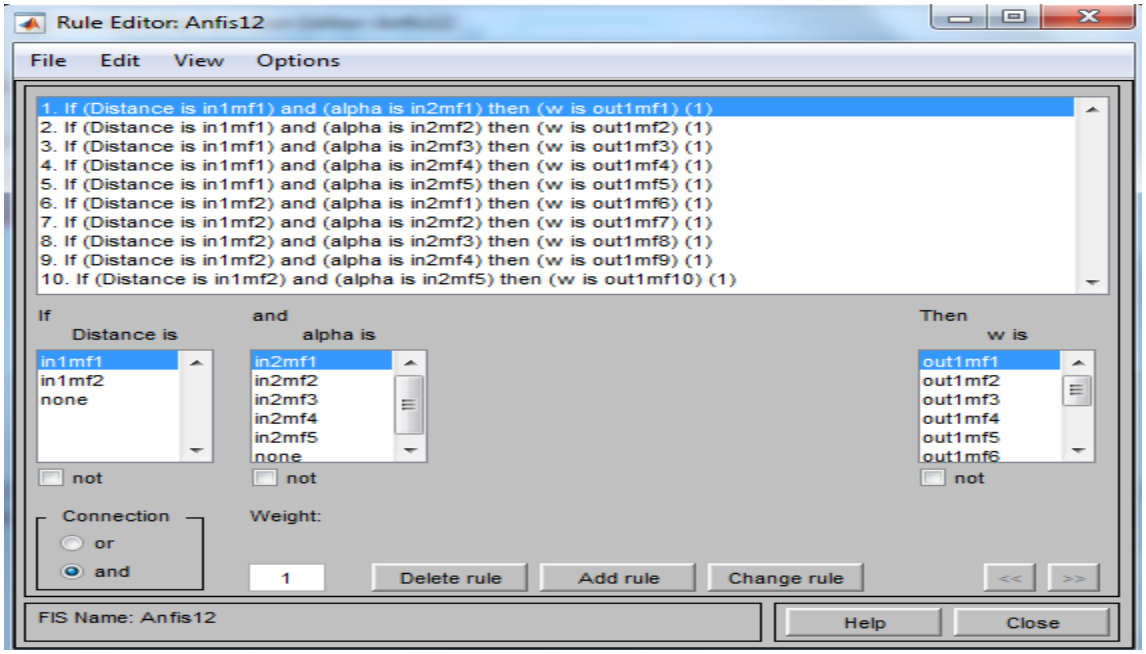


Figure 5.11: Anfis12 position controller rule.

Figure 5.11 shows Anfis12 position controller has ten (10) *if-then* rule base and the fuzzy rule view is shown in Figure 5.12 with their correspondence value.

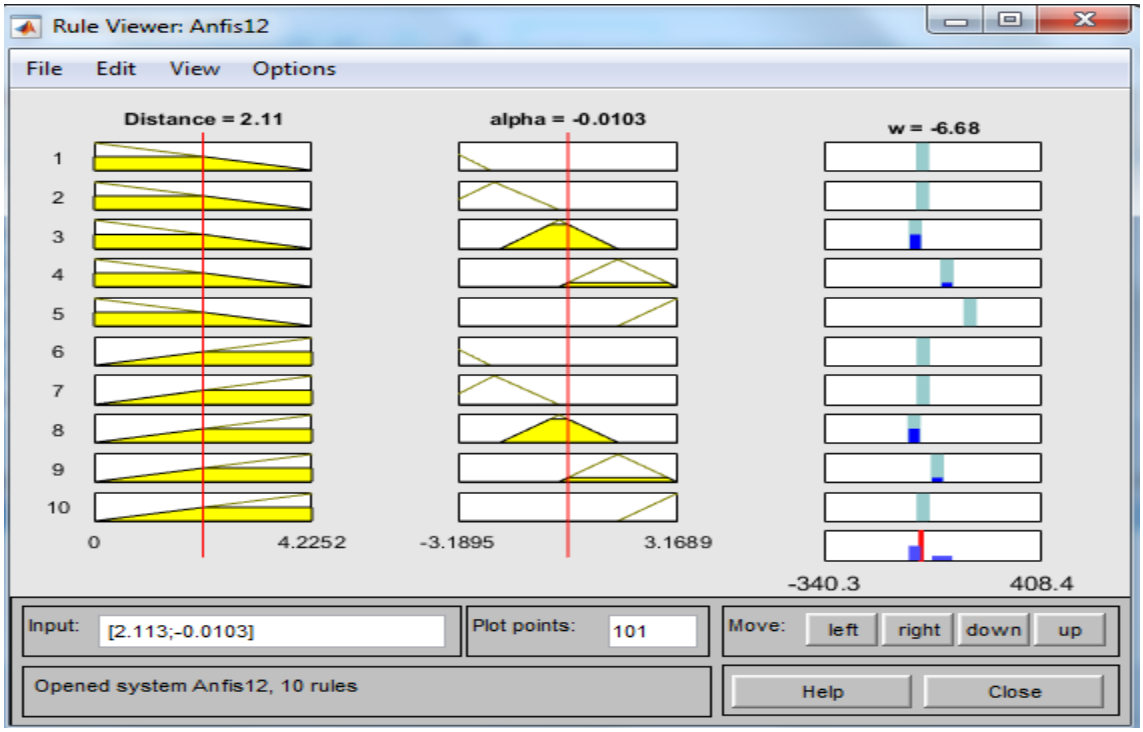


Figure 5.12: Anfis12 position controller rule view.

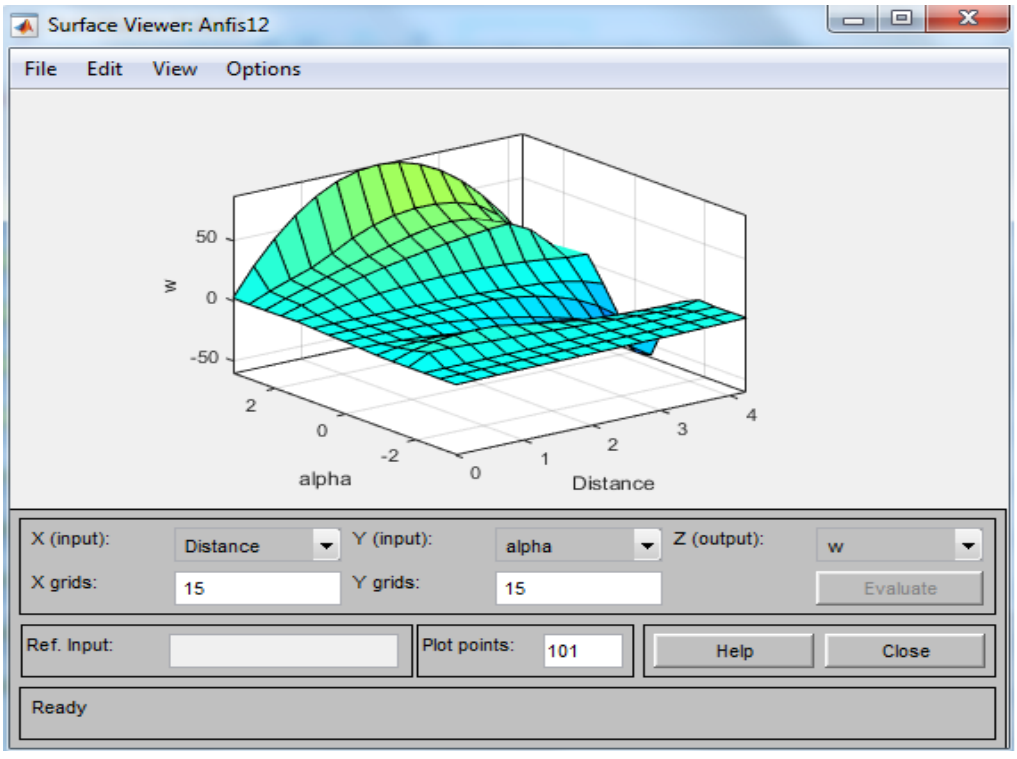


Figure 5.13: Anfis12 position controller surface view.

Figure 5.13 shows the 3D surface view of Anfis12 position controller in *Distance*, *alpha*, and *w*.

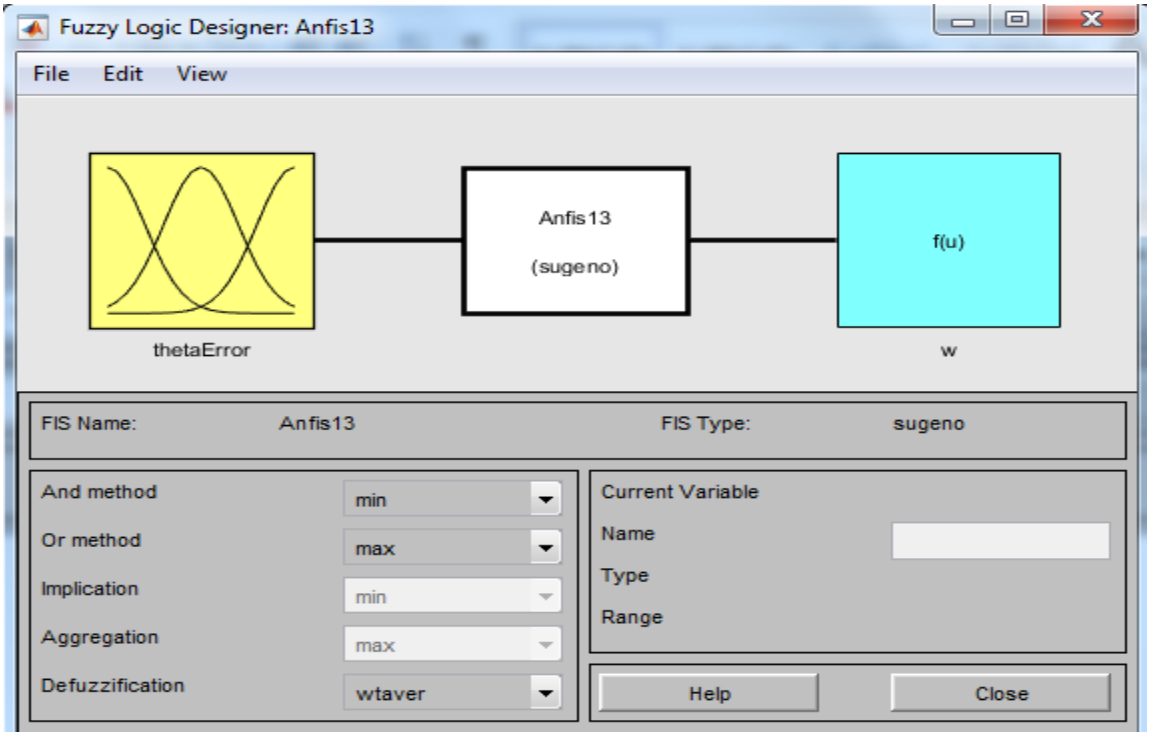


Figure 5.14: Anfis13 Orientation Controller Structure.

Figure 5.14 shows the orientation controller Anfis13 has one linguistic variable as inputs: "*thetaError"* and one linguistic variable as output: angular speed "*w*". The model structure view, their membership functions, fuzzy *if-then* rule base, fuzzy rule view and surface view are shown in Figures below as follow:

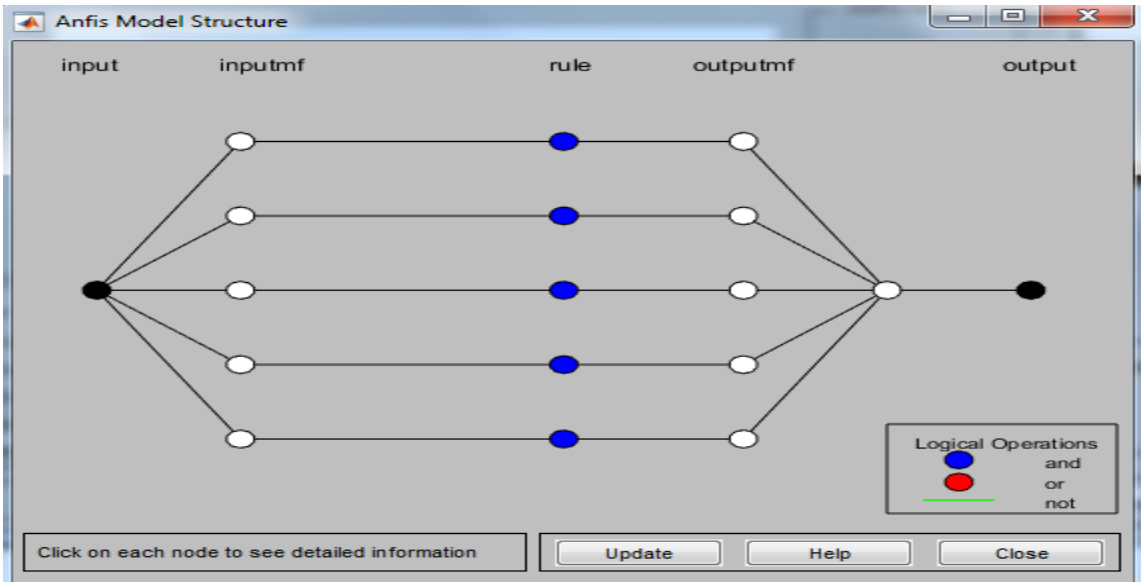


Figure 5.15: Anfis13 Orientation Controller Model Structure View.

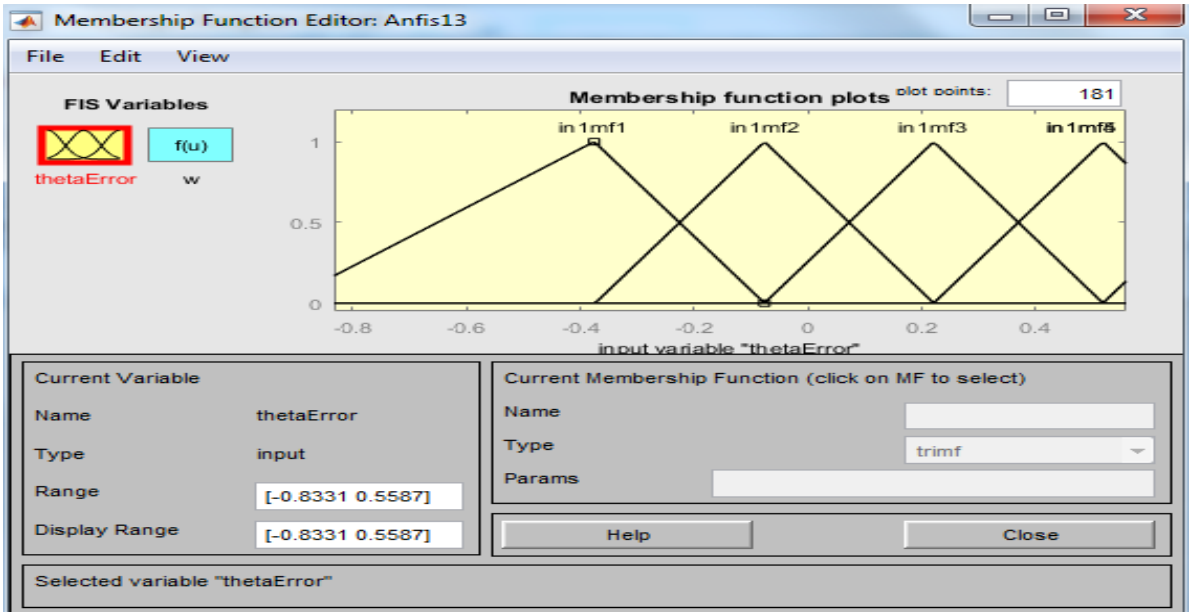


Figure 5.16a: Membership function for input linguistic variable "*thetaError*".

Figure 5.16a shows the input linguistic variable "*thetaError*" with its five linguistic terms: mf1, mf2, mf3, mf4 and mf5 in the interval [-0.8331, 0.5587].

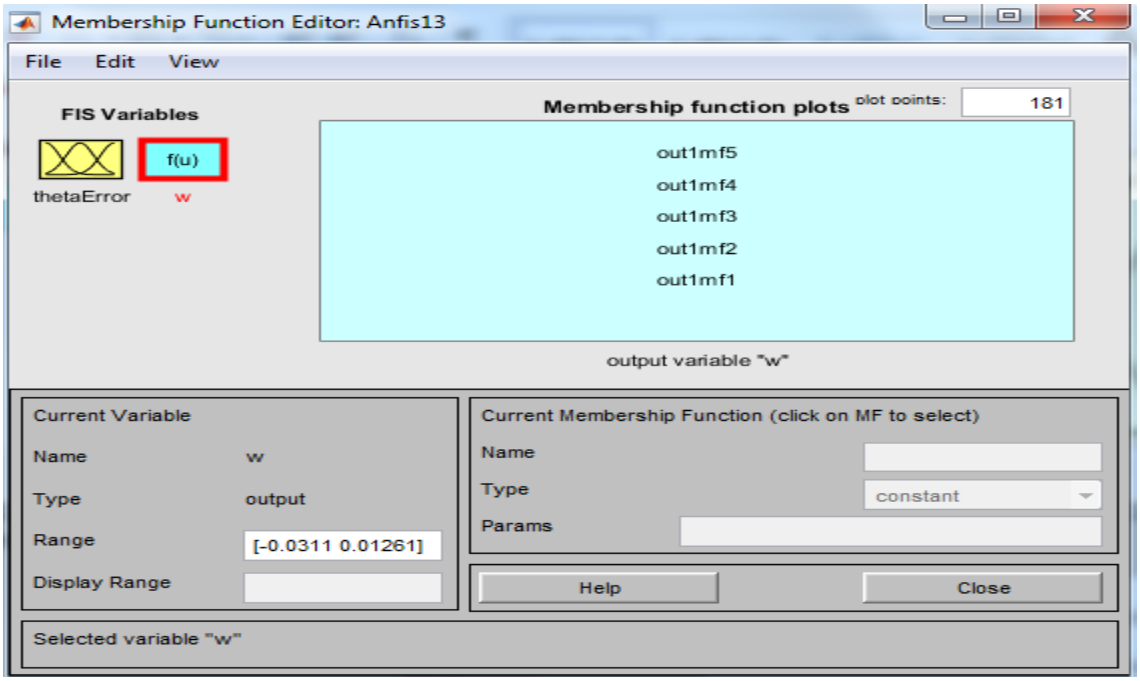


Figure 5.16b: Membership function for output linguistic variable "*w*".

Figure 5.16: Anfis13 membership function for orientation controller.

Figure 5.16b shows the output linguistic variable angular speed "*w*" with its five linguistic terms: mf1, mf2, mf3, mf4 and mf5 in the interval [-0.0311 0.0126].

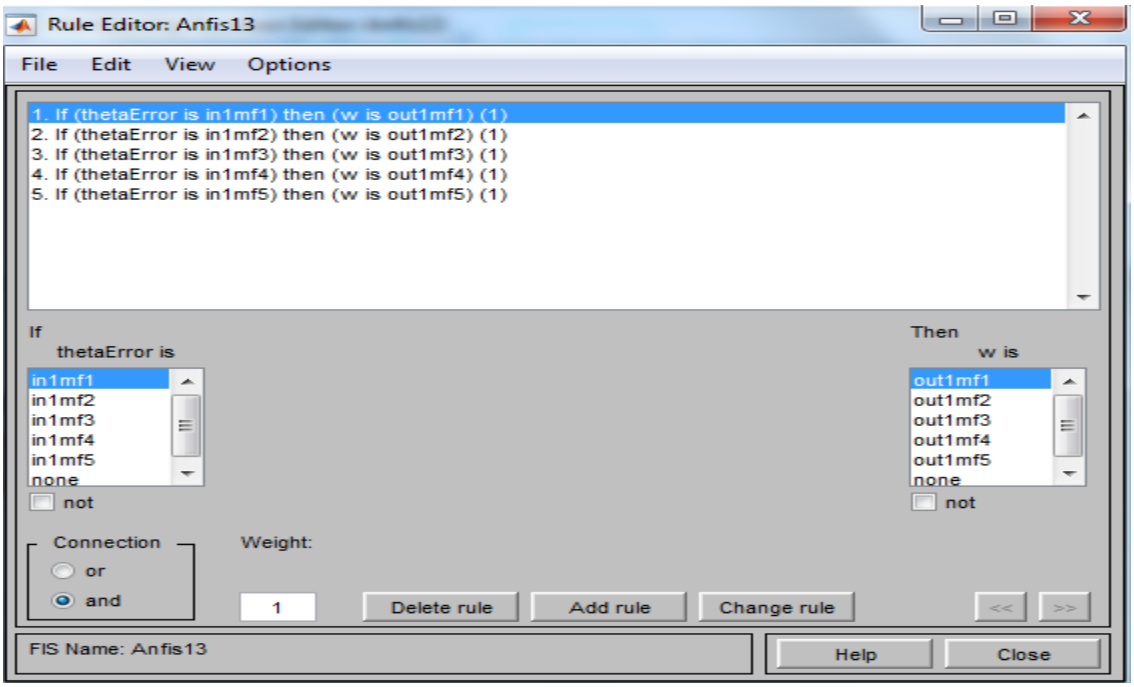


Figure 5.17: Anfis13 orientation controller rule.

Figure 5.17 shows Anfis13 orientation controller has five *if-then* rule base and the fuzzy rule view is shown in Figure 5.18 with their correspondence value.

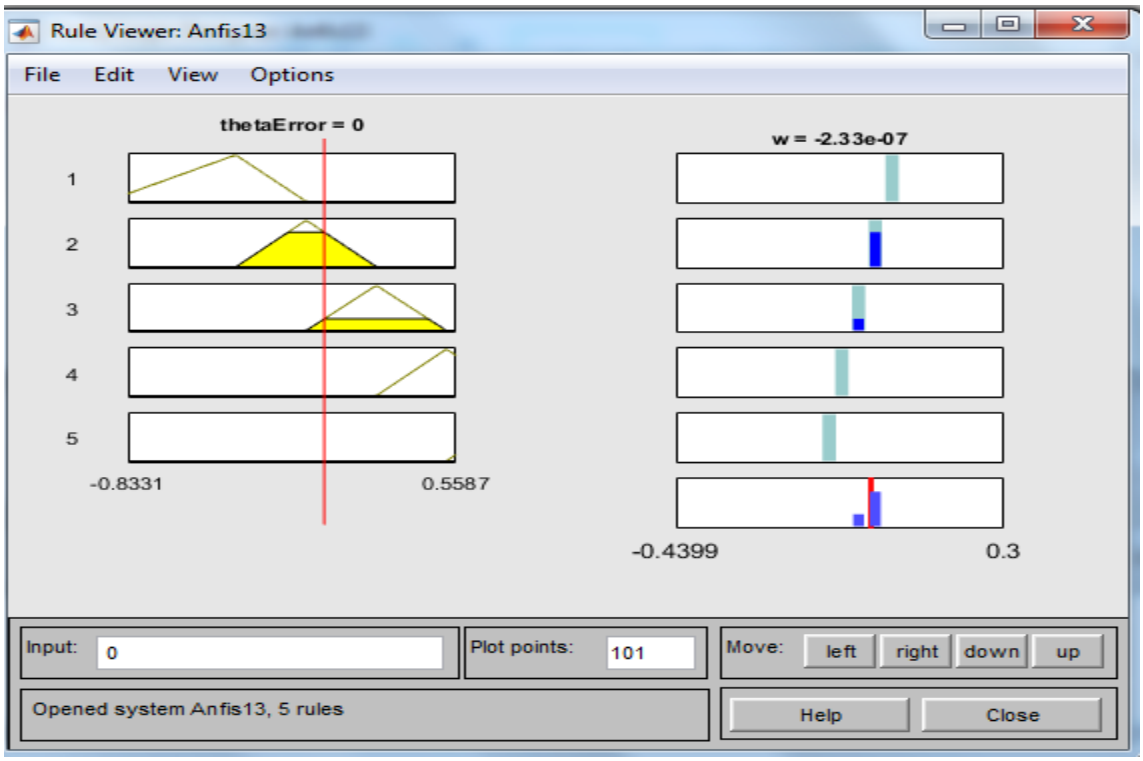


Figure 5.18: Anfis13 orientation controller rule view.

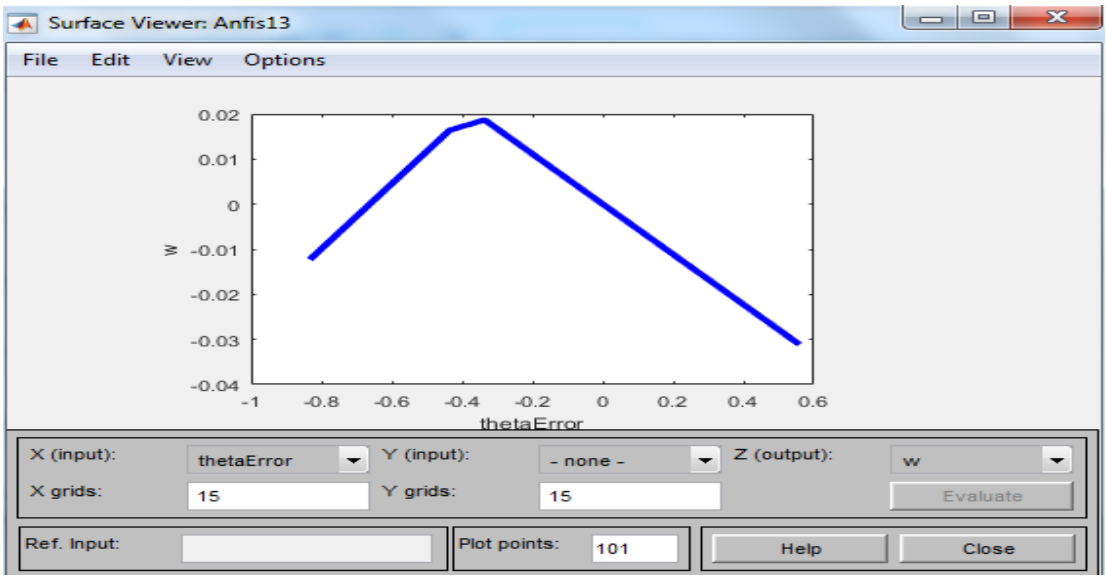


Figure 5.19: Anfis13 orientation controller surface view.

Figure 5.19 shows the 2D surface view of ANFIS12 position controller with *thetaError* and *w*.

Accordingly, different mobile robot trajectory tracking responses with different desired trajectory types are shown. The basic idea is to propose the desired (target) poses.

i) For repeating sequence desired trajectory [1 2 3 4]

The MATLAB Simulink block diagram to simulate the system response for this control structure is shown in Figure 5.20 below.

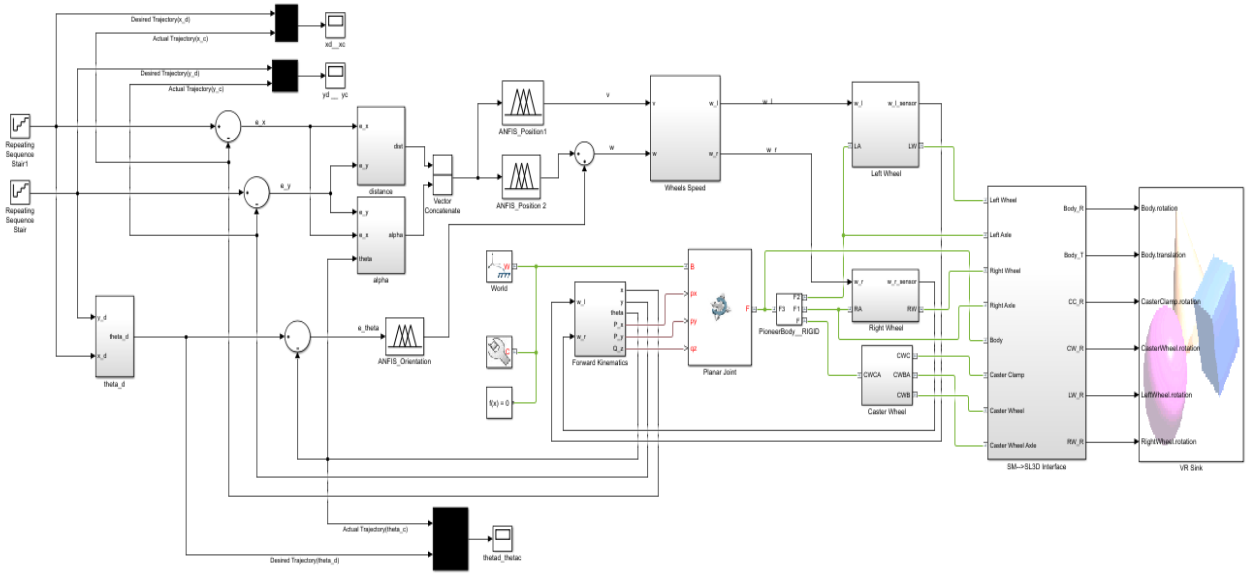


Figure 5.20: The Simulink block diagram to simulate the repeating sequence reference response.

ANFIS mainly adjusts robots motion direction and quickly moves it towards the target pose. Using the above block diagram we can test the MR response and is shown in Figures below as follows:

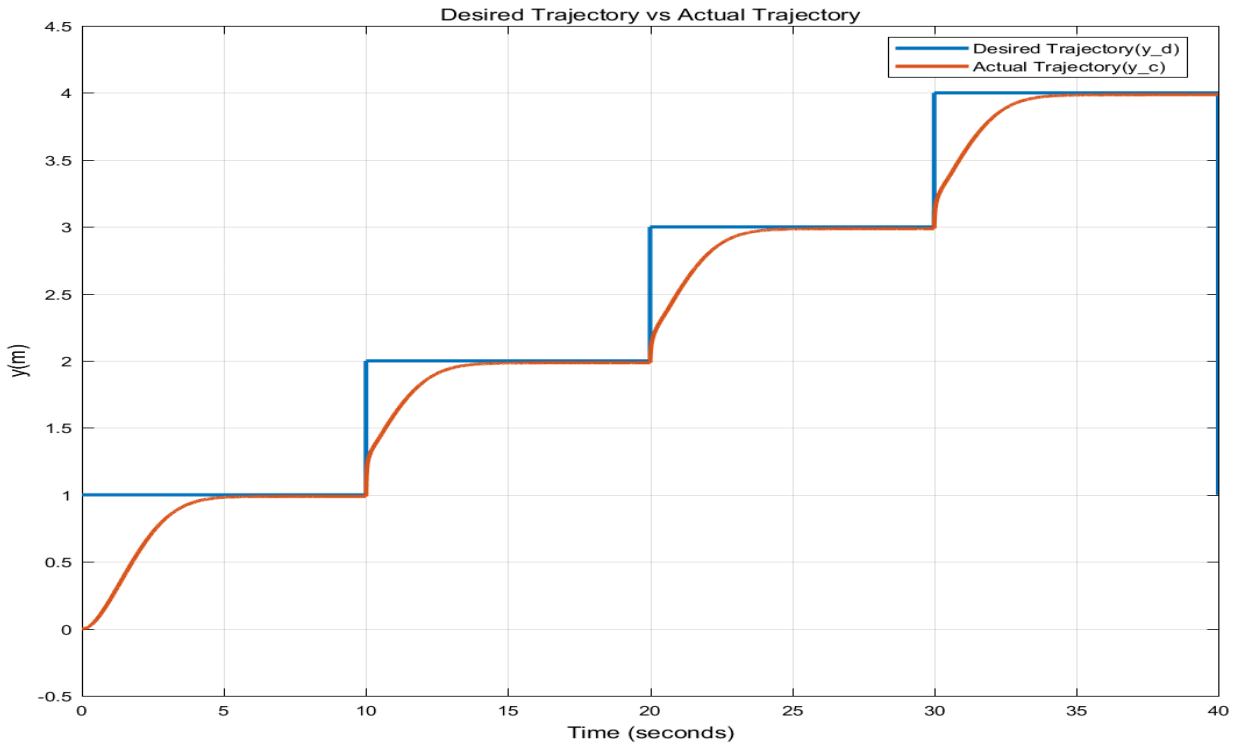


Figure 5.21a: Position in y-direction with time.

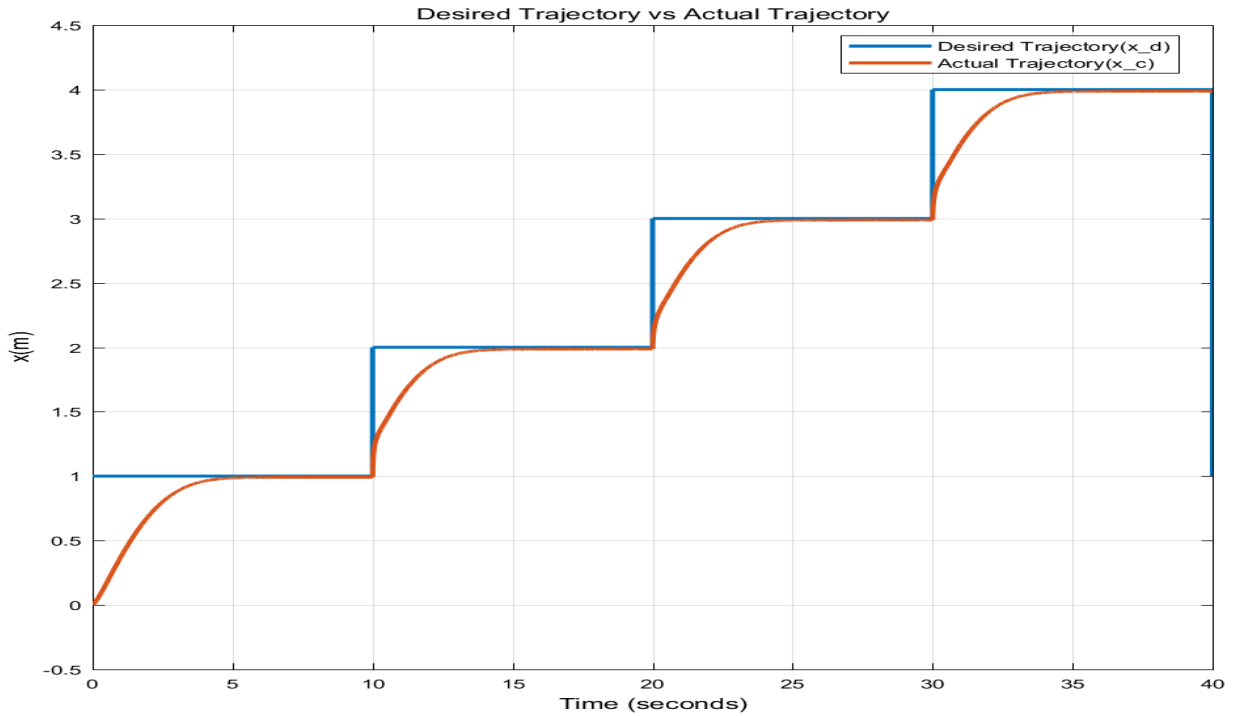


Figure 5.21b: Position in x-direction with time.

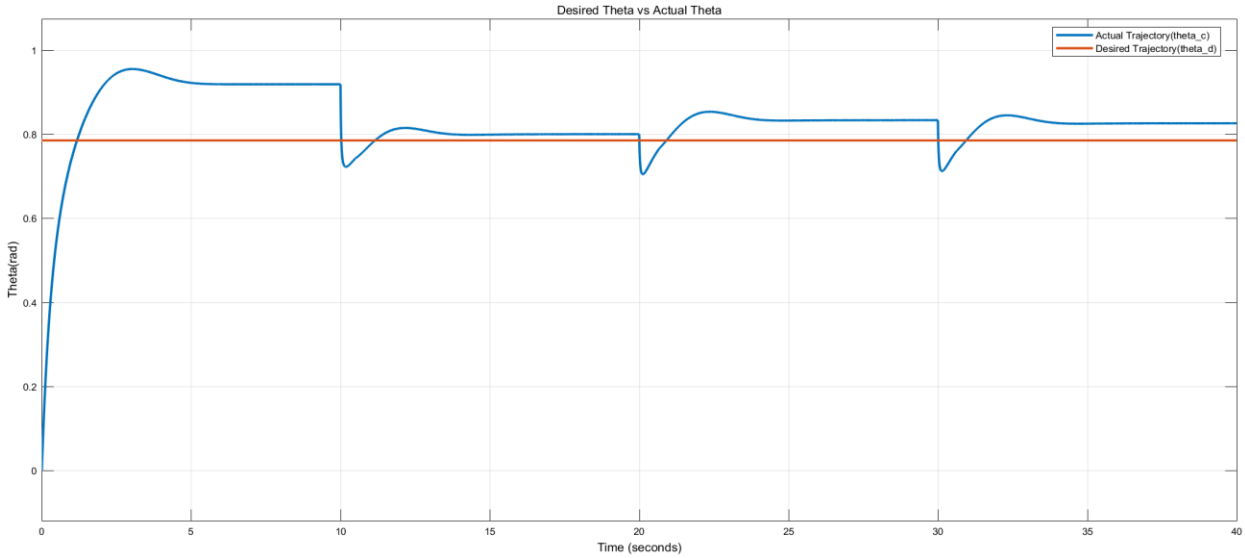


Figure 5.21c: Orientation in theta with time.

Figure 5.21: The robot reference and actual trajectory tracking.

Figure 5.21a and Figure 5.21b show the time in seconds that the MR lasted to orientate and arrive to the desired position (target), this process was achieved using Anfis11 and Anfis12. Figure 5.21c shows the time in seconds that the MR lasted to achieve the desired orientation (target), this process was achieved using Anfis13.

The angular speed needed from the robot wheel motors to provide such a tracking response is shown in Figure 5.22 below.

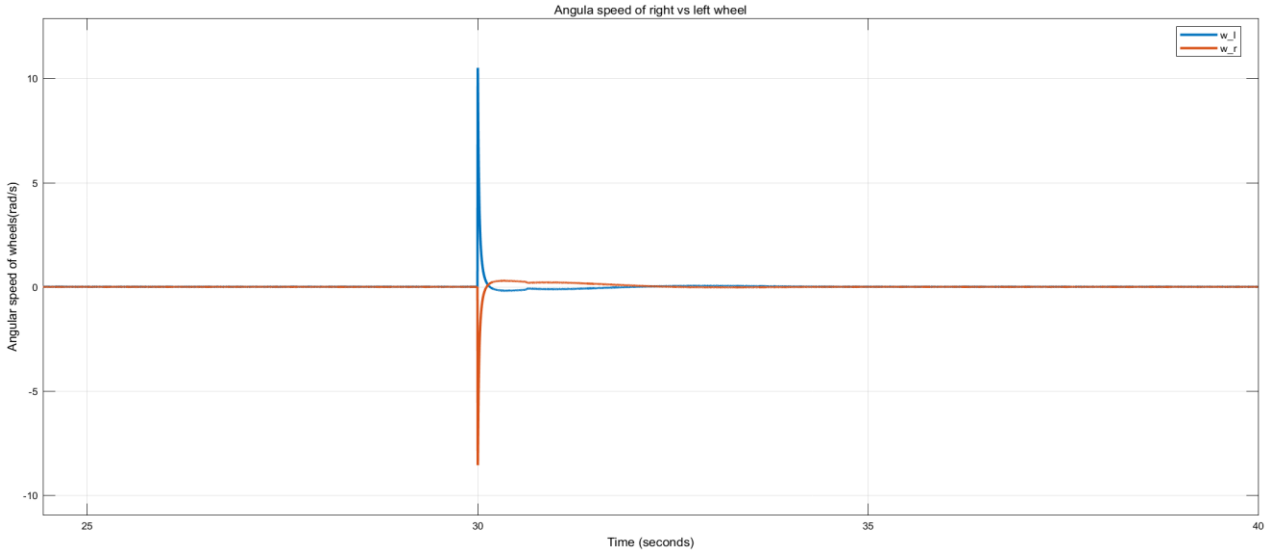


Figure 5.22: The angular speed of the right and left wheels response with time.

ii) For a nonlinear desired trajectory

The MATLAB Simulink block diagram to simulate the system response for this control structure is shown in Figure 5.23 below.

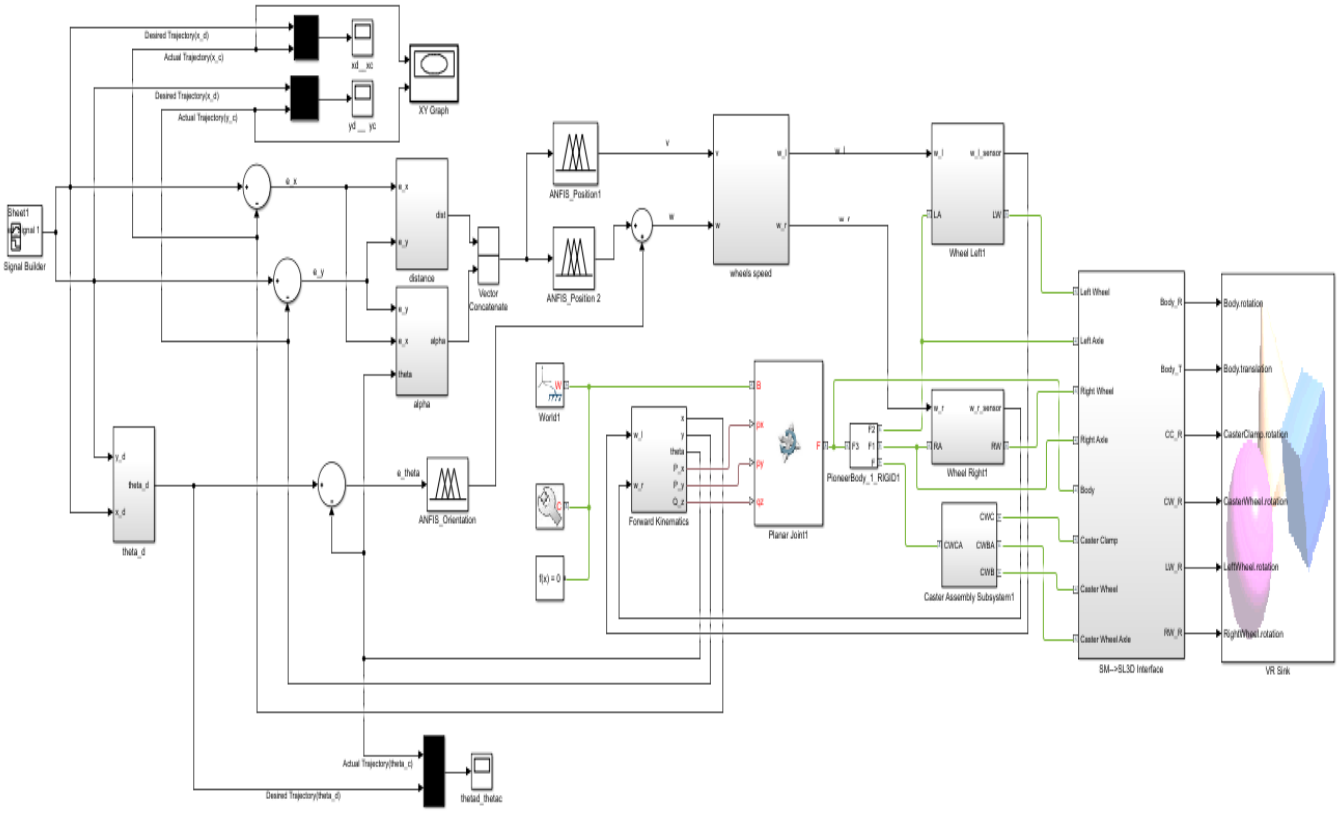


Figure 5.23: The MATLAB Simulink block diagram to simulate a nonlinear reference response.

The robot motion in response to such input using the above block diagram is shown in Figures below as follows:

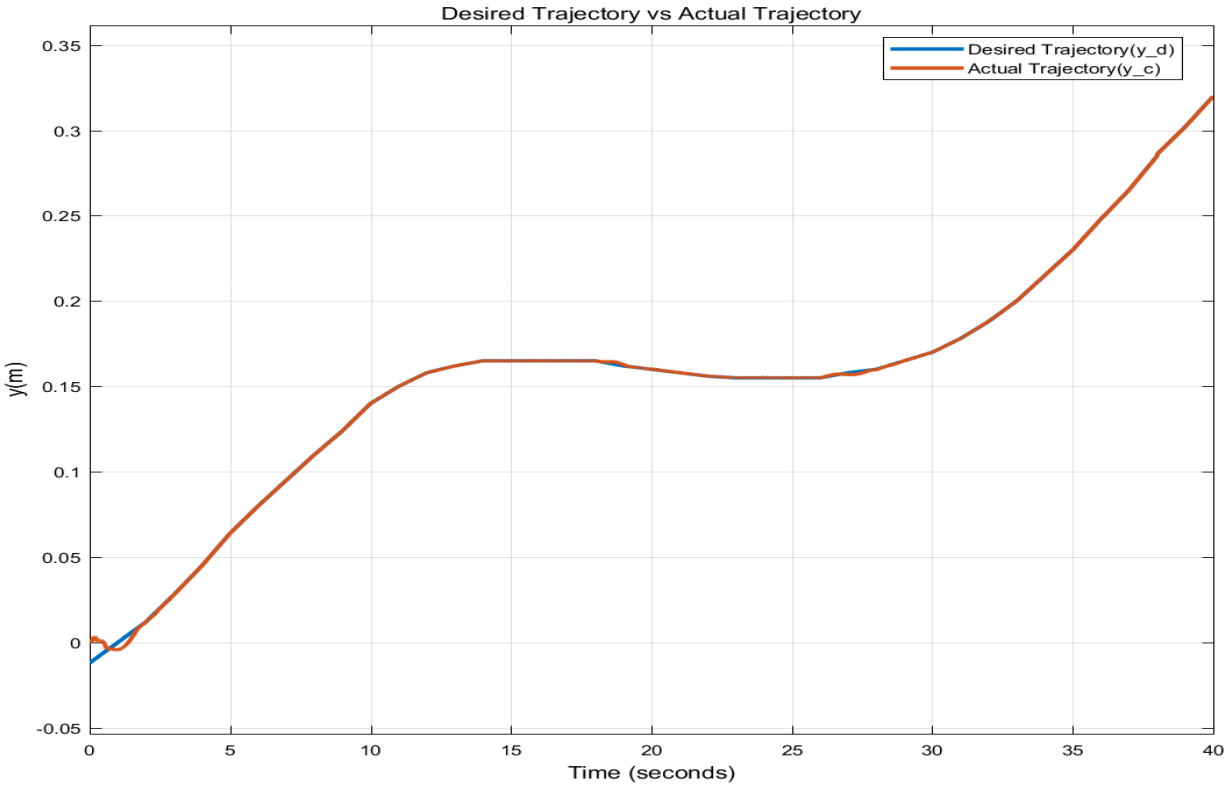


Figure 5.24a: Position in y-direction with time.

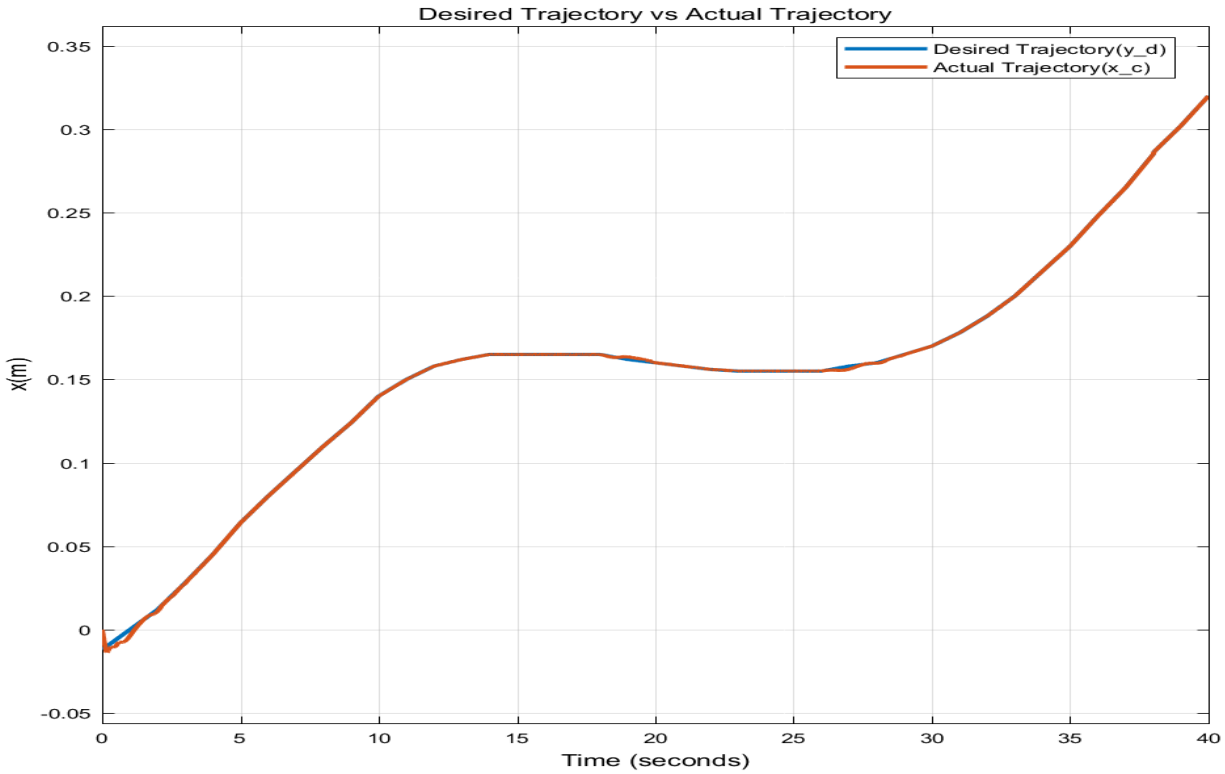


Figure 5.24b: Position in x-direction with time.

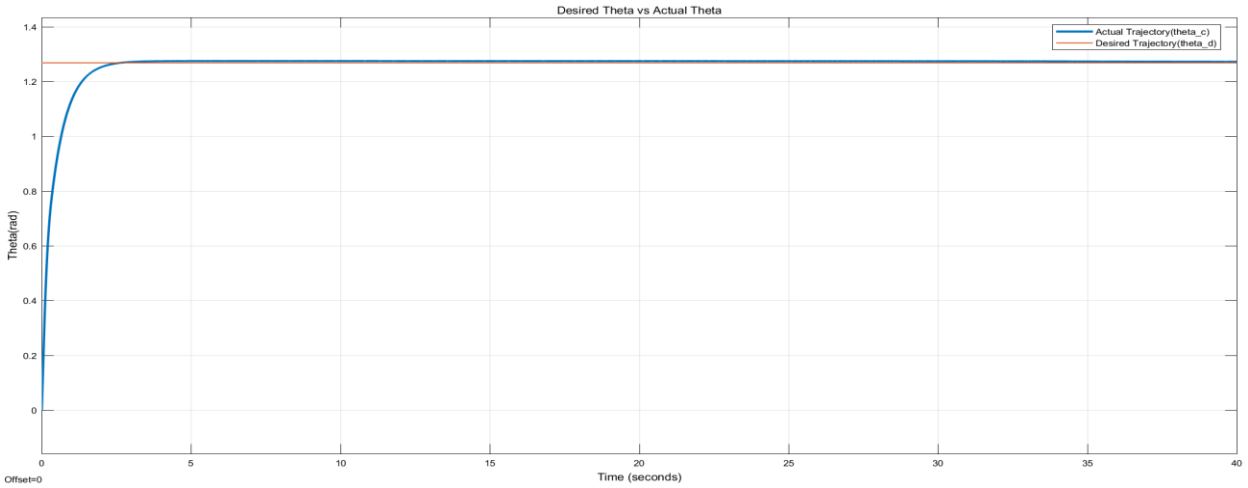


Figure 5.24c: Orientation in theta with time.

Figure 5.24: The robot reference and actual trajectory tracking.

Figure 5.24a and Figure 5.24b show the time in seconds that the MR lasted to orientate and arrive to the desired position (target), this process was achieved using ANFIS_Position1 (Anfis11) and ANFIS_Position2 (Anfis12). Figure 5.24c shows the time in seconds that the MR lasted to achieve the desired orientation (target), this process was achieved using ANFIS_Orientation (Anfis13). ANFIS mainly adjusts robots motion direction and quickly moves it towards the target pose accordingly.

The angular speed needed from the robot wheel motors to provide such a tracking response is shown in Figure 5.25 below.

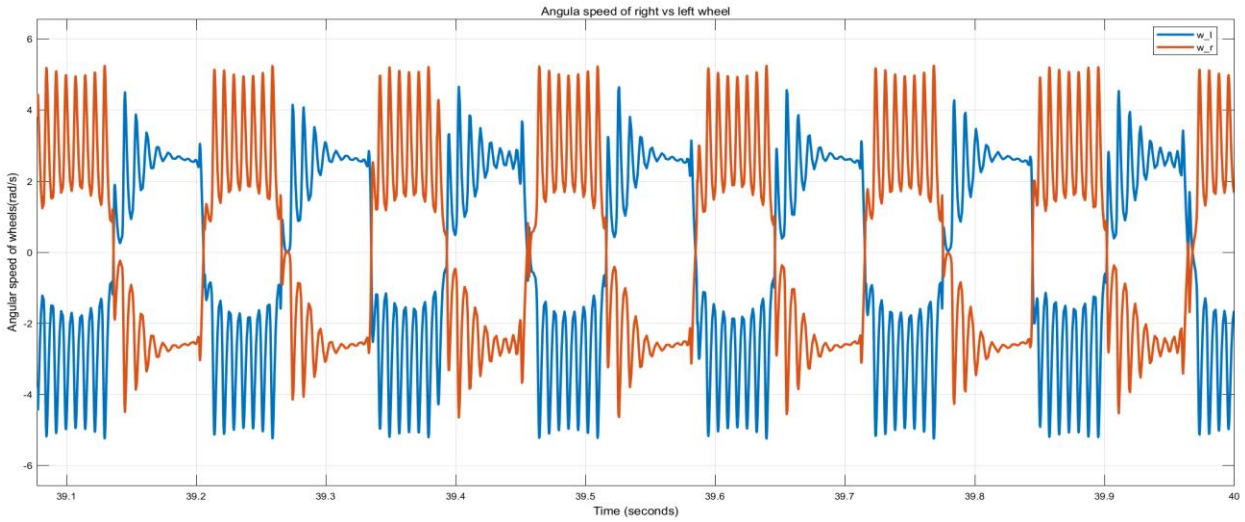


Figure 5.25: The angular speed of the right and left wheels response with time.

The above figure shows the trajectory of the robot with different desired reference trajectory. The motion of the robot is because of its non-holonomic constraint which imposes the no-lateral motion. Therefore, the designed controller is applicable to the real platform. The main purpose of the ANFIS controller is to make sure that the robot velocities will follow the velocities produced by the fuzzy controller. The advantage of having neural network in combination with FIS is that there is no need to know the dynamic model of the robot.

Simulation results demonstrate the effectiveness of the proposed algorithm, which proved the good tracking results and showed the robustness of the algorithm against the uncertainties in the system model.

5.3 Comparative analysis for Mamdani fuzzy controller and ANFIS controller

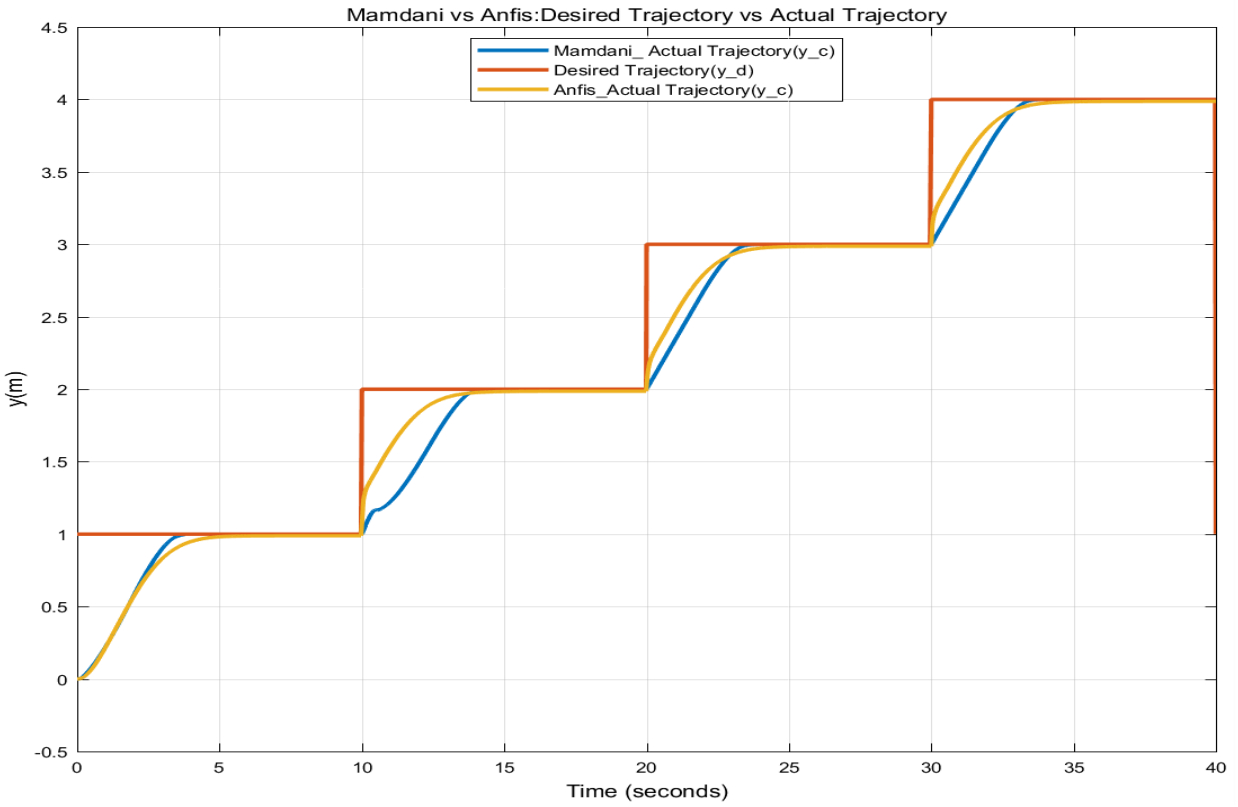
After training the ANFIS model, we can simulate the performance of the adaptive gain tuning controller using the Simulink block diagram. Comparison analysis between simple Mamdani based fuzzy controller and the ANFIS-based adaptive controller is done to show the advantages of the proposed controller. The following features of the controllers should be discussed to have a perfect comparison:

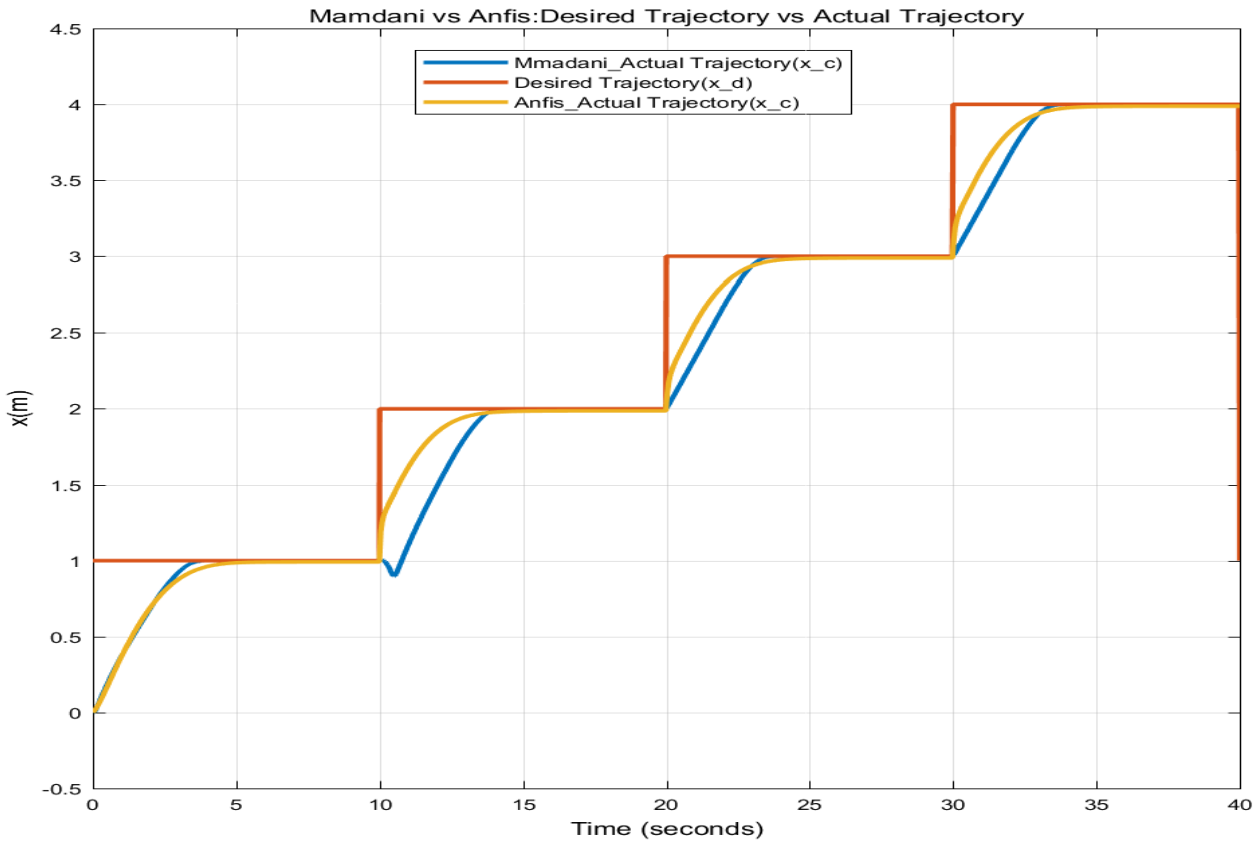
- ✓ The smoothness of the trajectories tracking.
- ✓ The trajectory tracking error.
- ✓ Energy saving.
- ✓ Time consuming limitation (fastness in running time).
- ✓ Performance of trajectory tracking.
- ✓ Trajectory tracking efficiency.

Comparison between the performances of the controllers in the above categories results in a comprehensive analysis and shows the advantages and restrictions of each controller. The simulation results of the Mamdani fuzzy controller and ANFIS controller in response to predefined reference trajectories will be shown in the remaining of this section and discussion will be made about the performance of each controller.

i) Comparison for repeating sequence reference trajectory

The time response of the robot output states $x(t)$, $y(t)$ and $\theta(t)$, the trajectory errors $e_x(t)$, $e_y(t)$ and $e_\theta(t)$ and the right and left wheel angular speeds are shown in the following figures.


Mamdani vs Anfis:Desired Trajectory vs Actual Trajectory
Mamdani_ Actual Trajectory(y_c)
Desired Trajectory(y_d)
Anfis_Actual Trajectory(y_c)
y(m)
Time (seconds)


Mamdani vs Anfis:Desired Trajectory vs Actual Trajectory
Mmadani_Actual Trajectory(x_c)
Desired Trajectory(x_d)
Anfis_Actual Trajectory(x_c)
x(m)
Time (seconds)

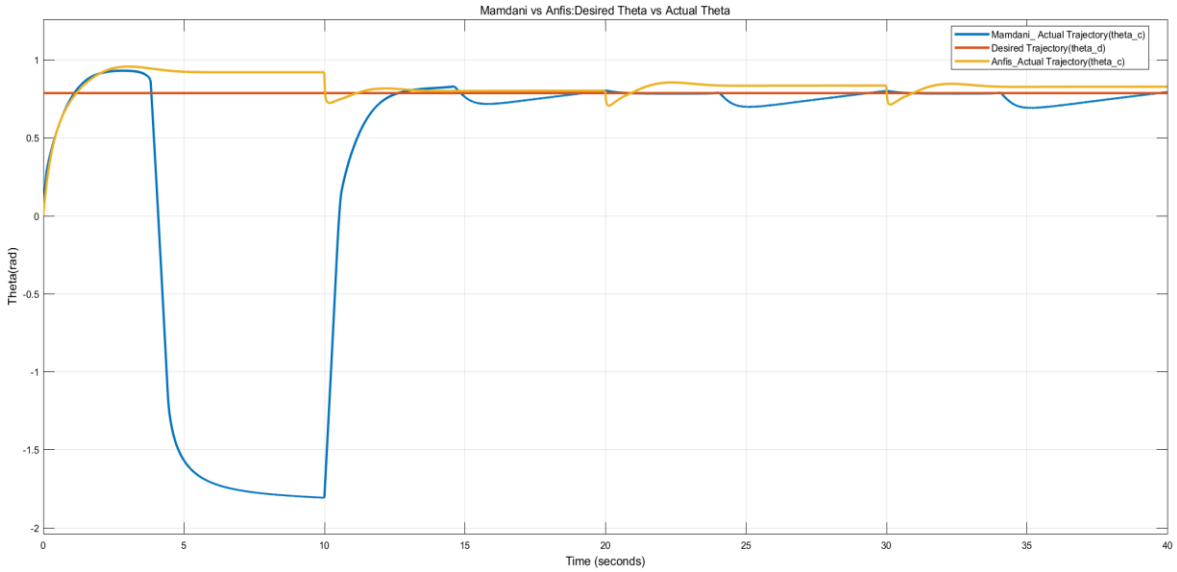
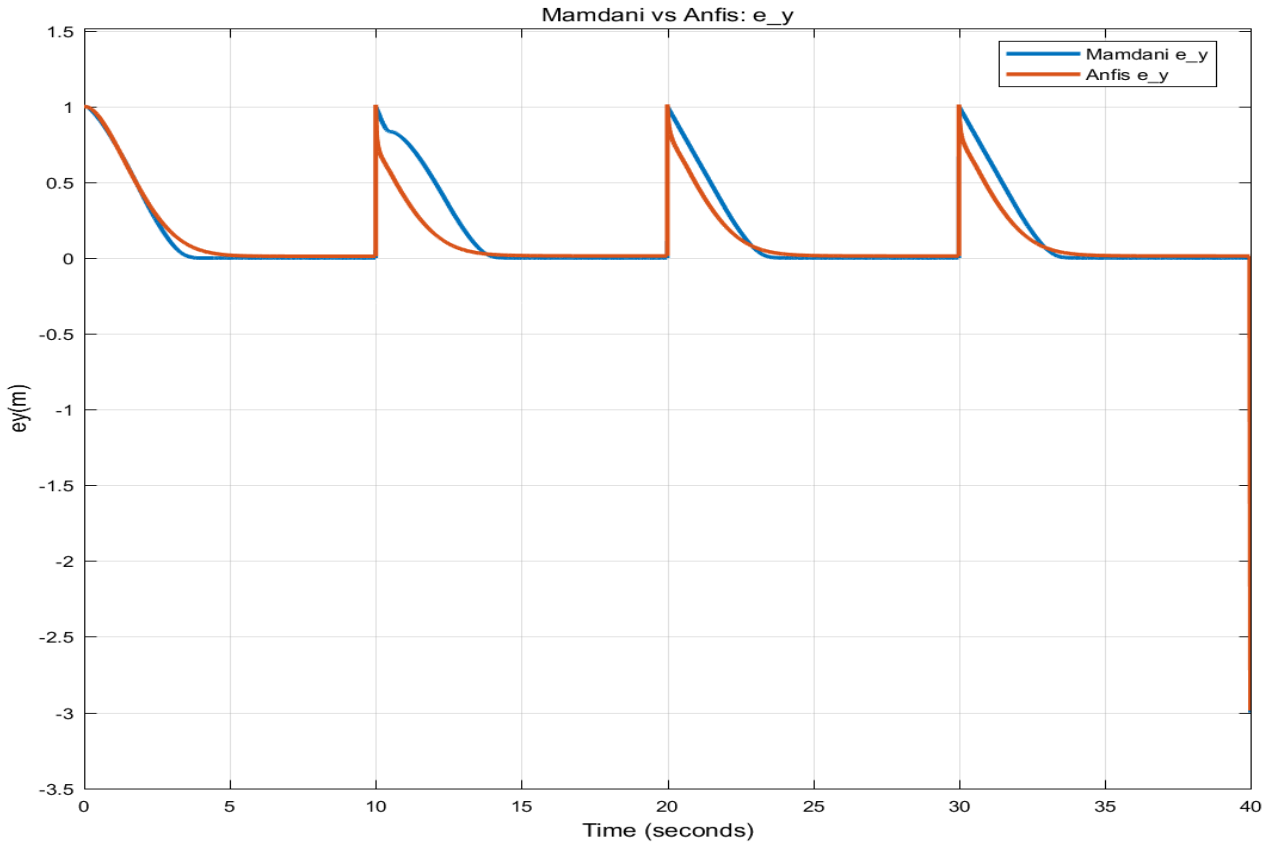


Figure 5.26: The robot output states time response vs reference trajectories, Comparison between the Mamdani fuzzy controller and ANFIS controller (Repeating sequence reference trajectory)



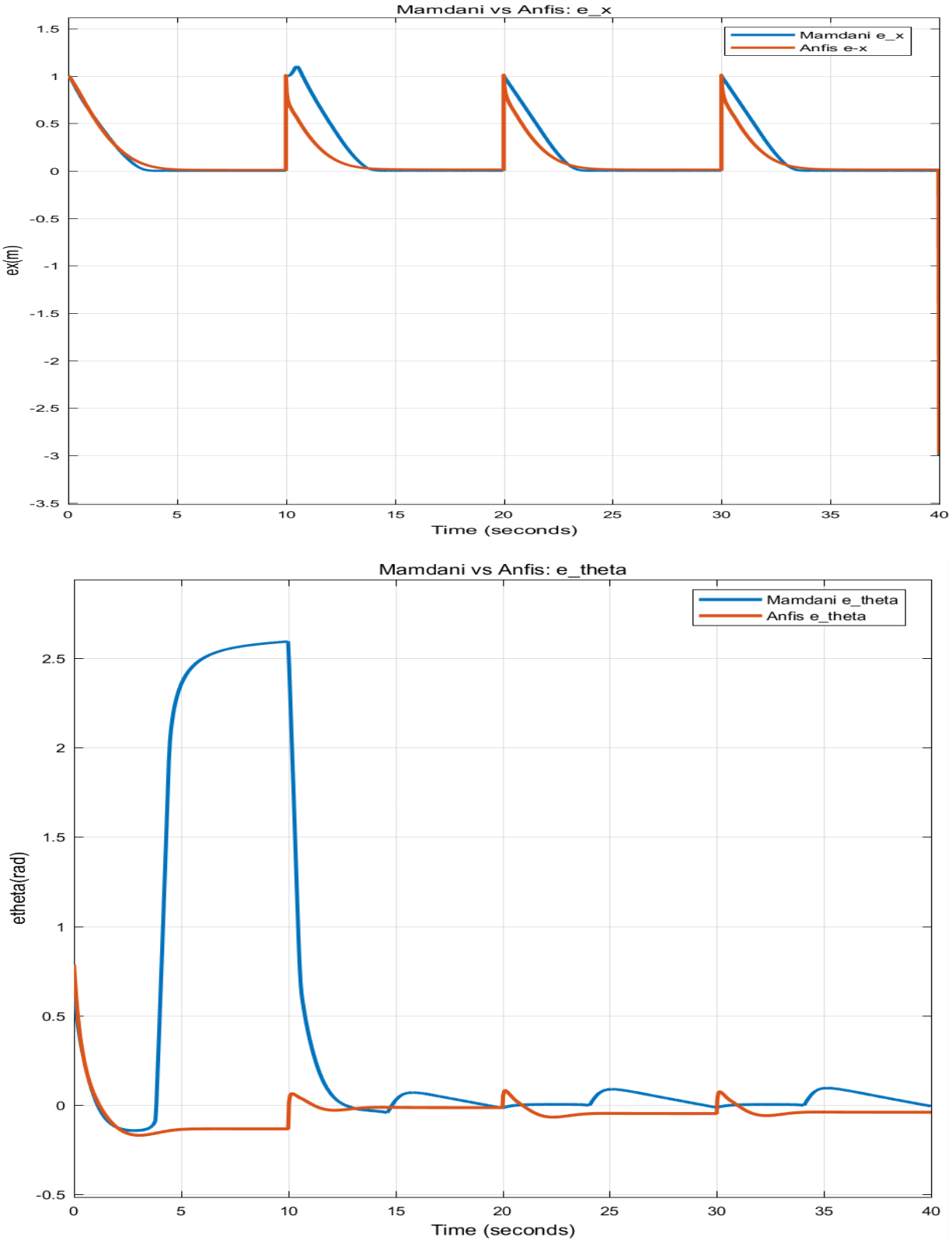


Figure 5.27: The robot trajectory errors time response, Comparison between the Mamdani fuzzy controller and ANFIS controller (Repeating sequence reference trajectory)

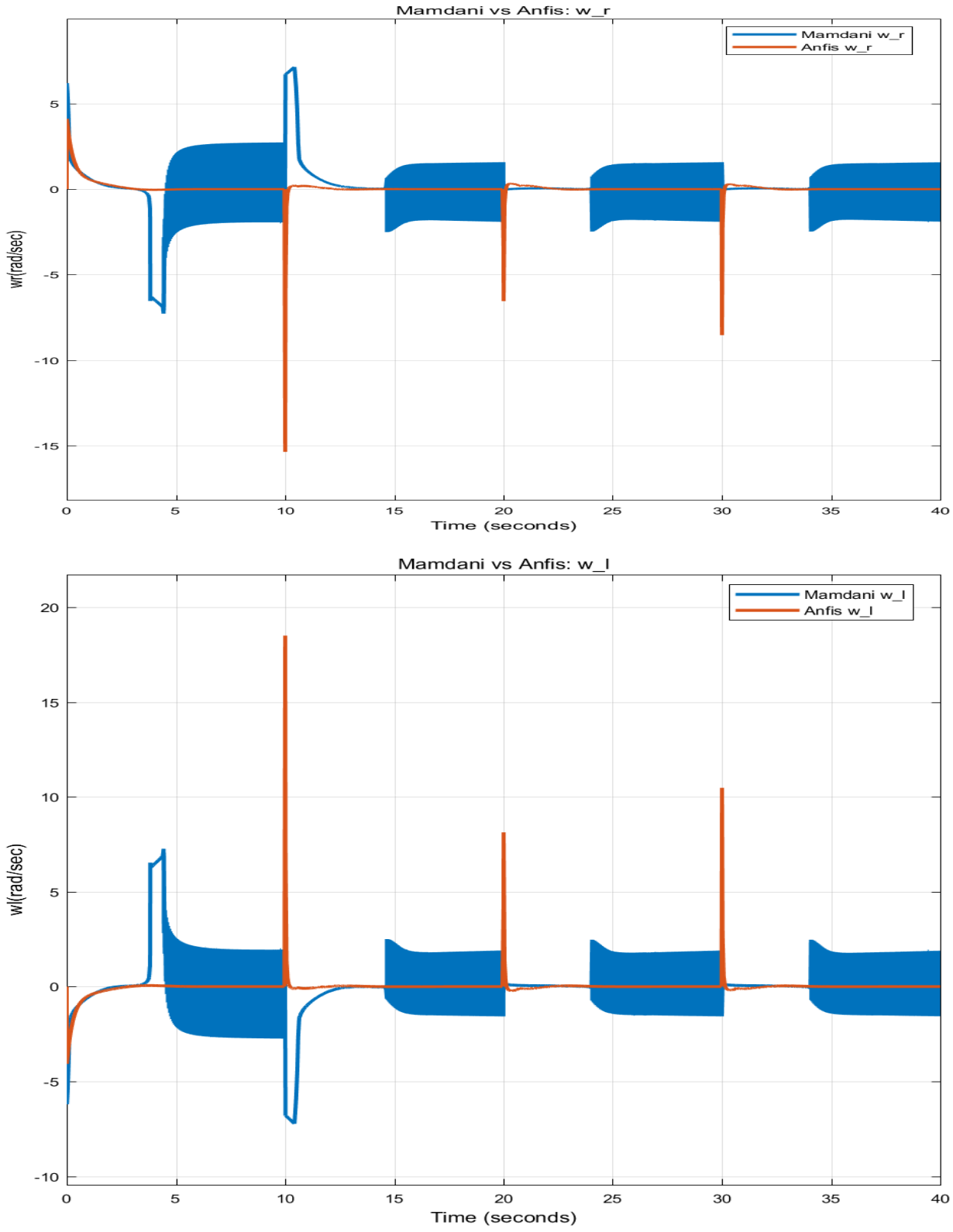


Figure 5.28: The right and left wheel angular speed, Comparison between the Mamdani fuzzy controller and ANFIS controller (Repeating sequence reference trajectory)

According to the above figure, the energy saving, time consuming limitation and performance of trajectory tracking of the ANFIS controller is better and the trajectory tracking error and the trajectory reach time is much smaller and the robot trajectory is much smoother in comparison with the Mamdani fuzzy controller. The tracking performance of Neuro-fuzzy controller in comparison to the Mamdani fuzzy controller is shown. Note that the Mamdani fuzzy controller is a time consuming process to get an accurate membership function. Making the controller ANFIS solves this problem and makes the robot track the given trajectory. The Mean Square Error (MSE) of the position controller and orientation controller for repeated sequence reference is shown in Table 5.1 to compare the deviation of simulated actual path from desired path between the Mamdani FIS and ANFIS.

MSE	Mamdani FIS	ANFIS
MSE_x (MSE in position x) (m)	0.0013	0.0008
MSE_y (MSE in position y) (m)	0.0012	0.0010
MSE_{theta} (MSE in orientation theta) (rad)	1.8556	0.0064

Table 5.1: Deviation from desired path comparison between Mamdani FIS and ANFIS for repeated sequence reference.

Among all these cases, ANFIS technique with Safe Boundary Algorithm is most suited for trajectory tracking of wheeled mobile robot (WMR), as shown in Table 5.1.

5.4 Discussions

The above comparison analysis shows that this proposed controller has the following advantages over the simple Mamdani based fuzzy controller:

- ✓ This controller can deal with unmodeled bounded disturbances and unstructured unmodeled dynamics in the vehicle which are a part of any environment that robot wants to perform the trajectory tracking. Thus, no knowledge of the system dynamics and parameters is needed in order to compensate for the nonlinear terms in the system.
- ✓ The other trajectory tracking controller that can be designed to improve the performance of the above controllers is the one that can adapt itself i.e., self-learning and self-tuning.

CHAPTER 6

CONCLUSIONS, RECOMMENDATIONS AND FUTURE WORKS

A trajectory tracking controller for differential drive mobile robot has been designed and simulated on Pioneer 3_DX robot platform.

6.1. Conclusions

This thesis has discussed the trajectory tracking for a differential drive wheeled mobile robot using adaptive neuro-fuzzy inference system (ANFIS). Adaptive neuro-fuzzy inference system (ANFIS) can be implemented to learn FIS and to identify and refine the various parameters in membership functions and fuzzy rules using training dataset values. The current adaptive tracking controller learns and develops the required information for mobile robot to track a desired trajectory. Different simulation results were performed to show the capability of the trajectory controller. This neuro-fuzzy controller makes it possible the MR track a given predefined reference trajectories. In order to compare the proposed method, a comparison is made with Mamdani based fuzzy approach. MSE is calculated for the Mamdani and ANFIS based controller. As an outcome, the ANFIS based controller is 38.46%, 16.67% and 99.65% more accurate than Mamdani based controller while tracking the *x*, *y* and *theta* values, respectively. Thus, it was found that the proposed adaptive neuro-fuzzy controller can be adaptable to any kind of predefined reference trajectories and has better performance than fuzzy controller in terms of pose accuracy.

6.2. Recommendations and Future Works

The recommendation of this research can be the real time implementation. It will be more preferable if we compare simulation results with real time experimental results to prove the authenticity of the proposed algorithm. In addition, it will be recommended in the future to develop more robust hybrid technique for better trajectory control.

The Future work can be in the field of path planning. Addition of the implemented trajectory tracking controllers to a path planning algorithm will complete the robot motion planning task with a very precise performance. The other future research on this mobile robot platform can be extended for a single mobile robot trajectory tracking. It will be more interesting if we can be used multiple mobile robots instead of a single mobile robot.

REFERENCES

[1] E. Wijn, "Unstable behavior of a unicycle mobile robot with tracking control," Report No. DCT 2004.63, Eindhoven, Eindhoven University of technology, 2004.

[2] R. Barzamini and A. Afshar, "Dynamic adaptive tracking control for wheeled mobile robots," AmirKabir University of Technology, Tehran, 2006.

[3] S. M. LaValle, "Planning Algorithms." Cambridge: Cambridge University Press. ISBN 0521862051 (Available online at http://planning.cs.uiuc.edu/), 2006

[4] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot: Backstepping kinematics into dynamics," Texas: Automation and Robotics institute, The University of Texas at Arlington, 1996.

[5] J. M. Yang and J. H. Kim, "Sliding mode control for trajectory tracking of non-holonomic wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, 578-587, 1999

[6] A. Bloch and S. Drakunov, "Stabilization of a nonholonomic system via sliding modes," *IEEE international conference on Decision Control*, pp. 2961-2963, 1994.

[7] K. Sangwon and P. Chongkug, "Optimal tracking controller for an autonomous wheeled mobile robot using fuzzy-genetic algorithm," *SPIE, the International Society for Optical Engineering, 2006.*

[8] K. N. Faress, M. T. El hargy and A. A. El kosy, " Trajectory tracking control for a wheeled mobile robot using fuzzy logic controller," *Proceedings of the 9th WSEAS International Conference on Systems*, Athens, Greece,2005.

[9] O. Castillo and L. T. Aguilar, "Fuzzy logic tracking control for unicycle mobile robots," *Advanced online publications*, 2006.

[10] X. Jiang and M. Yung, "Predictive fuzzy logic controller for trajectory tracking of a mobile robot," *IEEE Mid-Summer Workshop on Soft Computing in Industrial Applications*, pp. 29 – 32, 2005

[11] M. K. Bugeja and S. G. Fabri, "Dual Adaptive Control for Trajectory Tracking of Mobile Robots," *IEEE international conference on robotics and automation*, pp. 2215 – 2220, 2007.

[12] L. Ge, R. Li, D. Yu and L. Zhao, "A Nonlinear Adaptive Variable Structure Trajectory Tracking Algorithm for Mobile Robots," *intelligent robotics and applications*, pp. 892-900, 2008

[13] Z. Y. Liu, R. H. Jing, X. Q. Ding and J. H. Li, " Trajectory Tracking Control of Wheeled Mobile Robots Based on the Artificial Potential Field," *International conference on neuro computing*, pp. 382-387,2008

[14] F. Pourboghrat and M. P. Karlsson, "Adaptive control of dynamic mobile robots with nonholonomic constraints," Carbondale, IL 62901-6603, USA: Department of Electrical and Computer Engineering, Southern Illinois University, 2002

[15] D. Gu and H. Hu, "Wavelet Neural Network based predictive control for Mobile robots," Dept. of Engineering science, University of Essex, 2000.

[16] J. Ye, "Tracking control for nonholonomic mobile robots: Integrating the analog neural network into backstepping technique," *Neurocomputing*, 3373-3378. (2007).

[17] R. Fierro and F. L. Lewis, "Control of a nonholonomic mobile robot using neural networks," *IEEE transactions of neural networks*, 589-600, 1998.

[18] J. Velagic, N. Osmic and B. Lacevic, "Neural Network controller for mobile robot motion control," *International journal of intelligent systems and technologies, 2008.*

[19] G. Campion, G. Bastin and B. D'Andrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics and Control*, 47-62, 1996.

[20] G. Campion and G. Bastin, "On adaptive linearizing control of Omni-directional mobile robots," *MTNS*, (pp. 531-538), Amsterdam, 1989

[21] R. C. Arkin. *Behavior-Based Robotics*. MIT Press, Cambridge, Ma, 1998.

[22] C. Dongkyoung, S. Jin, K. Pyojae and J. Young, "Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots," *IEEE Transactions of Robotic and Automation*, 578-587, 1999.

[23] L. A. Zadeh. Fuzzy Sets. *Information and Control*, No 8, pp. 338-353, June 1965.

[24] R. M. DeSantis, "Modeling and path-tracking control of a mobile wheeled robot with differential drive," *Robotica*, volume 13, pp. 401-410, Cambridge university press, 1995.

[25] Y. kanayama and Y. Matsuo, "A Stable tracking control method for an autonomous mobile robot*," IEEE Transactions on Robotics and Control,* Arizona, 1990

[26] Rev. E, "ERA-MOBI user's manual", videre design, 2006

[27] Jyh-Shing R. Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Trans. Systems, Man & Cybernetics*, Vol. 23, pp. 665-685, 1993.

[28] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. PhD Thesis, Harvard University, 1974.

[29] Y. Tsukamoto. An approach to fuzzy reasoning method. *Advances in Fuzzy Set Theory and Applications.* pp. 137–149. North-Holland, Amsterdam, 1979.

[30] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller-part 1. *IEEE Trans. on Systems, Man, and Cybernetic*s, Vol. 20(2), pp. 404–418, 1990.

[31] C.C. Lee. Fuzzy logic in control systems: fuzzy logic controller-part 2. *IEEE Trans. on Systems, Man, and Cybernetic*s, Vol. 20(2), pp. 419–435, 1990.

[32] T. Takagi and M. Sugeno. Derivation of fuzzy control rules from human operator's control actions. *Process of the IFAC Symp on Fuzzy Information, Knowledge Representation and Decision Analysis*, pp. 55–60, July1983.

[33] Campion Guy, Bastin Georges, and Brigitte D' AndrCa-Novel, ―Structural properties and classification of kinematic and dynamic models of wheeled mobile robots, IEEE Transactions on Robotics and Automation 12 (1) (1996) 47–62.

[34] J. –S. Roger Jang. Anfis: adaptive-network-based fuzzy inference systems. IEEE Trans. On systems, Man and Cybernetics, 1992.

[35] Ajith Abraham. Adaptation of Fuzzy Inference System Using Neural Learning, Fuzzy System Engineering: Theory and Practice, Nadia Nedjah et al. (Eds.), Studies in Fuzziness and Soft Computing, Springer Verlag Germany, ISBN 3-540-25322-X, Chapter 3, pp. 53-83, 2005.

[36] J. –S. Roger Jang. Self-learning fuzzy controller based on the temporal back-propagation. IEEE Trans. On Neural Networks, 3(5): 714-723, September 1992.

[37] J.-S.R Jang, and C.T. Sun. "Neuro-fuzzy modeling and control" Proceedings of the IEEE, 1995, pp. 378-406.

[38] J.-S.R Jang, and C.T. Sun. Functional equivalence between radial basis function networks and fuzzy inference systems. IEEE Trans. On Neural Networks, 4(1): 156-159, January 1993.

[39] J.-S.R Jang, and C.T. Sun. Proceeding chaotic time series with fuzzy if-then rules. In Proc. Of IEEE international conference on fuzzy systems, San Francisco, March1993.

[40] J. –S. Roger Jang. Fuzzy modeling using generalized neural networks and Kalman filter algorithm. In Proc. Of the Ninth National Conference on Artificial Intelligence (AAAI-91), pp: 762-767, July 1991.

[41] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturization: A tool for investigation in control algorithms. Informatik, pages 17-20, February, 1994.

[42] J. Godjevac. "A Learning Procedure for a Fuzzy System: Application to Obstacle Avoidance". In Proceedings of the International Symposium on Fuzzy Logic, 1995, pp. 142-148.

[43] H. Nomura, I. Hayashi, and N. Wakami. A learning method of fuzzy inference rules by descent method. In Proceedings of IEEE Int. Conf. on Fuzzy Systems, pp: 203-210, San Diego, 1992.

[44] M. Pascal and S. Claude. "Motion Control of Wheeled Mobile Robots" Handbook of Robotics, S. Bruno, K.S Oussama (Eds), Springer-Verlag Berlin Heidelberg, 2008, pp.799-825.

[45] R.W. Brockett, R. H, "Asymptotic stability and feedback stabilization," Boston, MA 1983.

APPENDIX

APPENDIX A: ROBOT PARAMETERS SPECIFICATIONS

SPECIFICATIONS	Pioneer 3-DX

Physical Characteristics

Length (cm)	44.5
Width (cm)	39.3
Height (cm)	23.7
Clearance (cm)	6.0
Clearance bumpers (cm)	3.5
Weight (kg)	9
Payload (kg)	25

Power

Batteries 12VDC lead-acid	3
Charge (wtt-hrs)	252
Run time (hrs)	8–10
with PC (hrs)	3-4
Recharge time hr/battery std charger	6
High-Speed (3 batteries)	2.4

Mobility

Wheels	2 foam-filled
tread	Knobby
diam (mm)	195.3
width (mm)	47.4
Caster (mm)	75
Steering	Differential
Gear ratio	38.3:1

Swing (cm)	26.7
Turn (cm)	0
Translate speed max (mm/sec)	1,400
Rotate speed max (deg/sec)	300
Traversable step max (mm)	20
Traversable gap max (mm)	89
Traversable slope max (grade)	25%
Traversable terrains	Wheel-chair accessible

Sensors		Pioneer 3-AT	Performance PeopleBot
Sonar Front Array (one each side, six forward @ 20° intervals)	8	8 optional	8
Rear Sonar Array (one each side, six rear @ 20° intervals)	8 optional	8 optional	8
Top Deck Sonar (one each side, six forward @ 20° intervals)	na	na	8
Rear Deck Sonar (one each side, six forward @ 20° intervals)	na	na	8 optional
Encoders (2 ea) counts/rev	76,600	34,000	76,600
counts/mm	128	49	128
counts/rotation	33,500	22,500	33,500
Bumpers	Optional	Optional	Standard

Controls and Ports			
Main Power	Standard	Standard	Standard
Charge	Standard	Standard	Standard

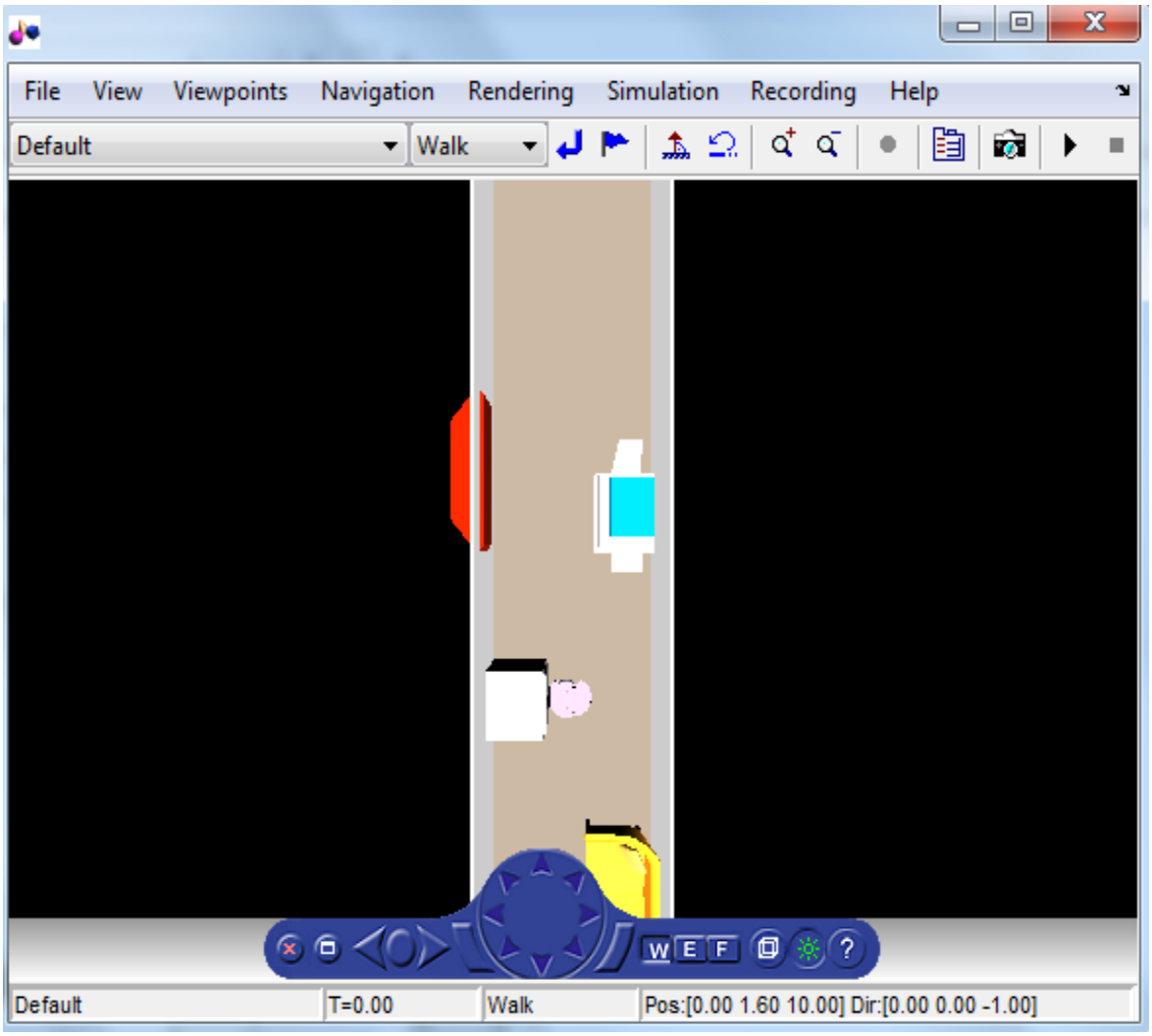
Joydrive	Optional	Standard	Standard
Motor Stop	Optional	Standard	Standard
Aux1 Power	5 & 12 VDC sw'd	5 & 12 VDC sw'd	5 & 12 VDC sw'd
Aux2 Power	5 & 12 VDC sw'd	5 & 12 VDC sw'd	5 & 12 VDC sw'd
System Serial	Standard	Standard	Standard
Motors	Standard	Standard	Standard
Microcontroller Reset	Standard	Standard	Standard

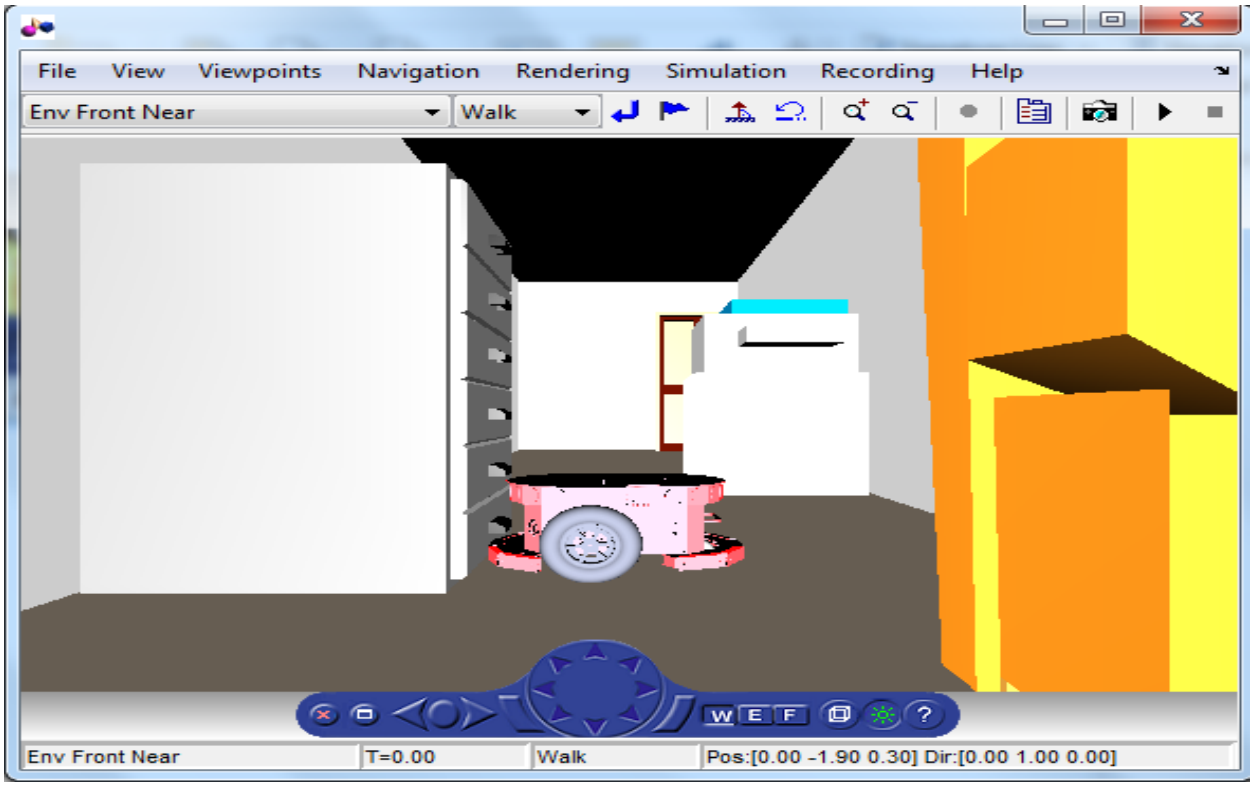
APPENDIX B: MATLAB CODE FOR 3RD ORDER TRAJECTORY

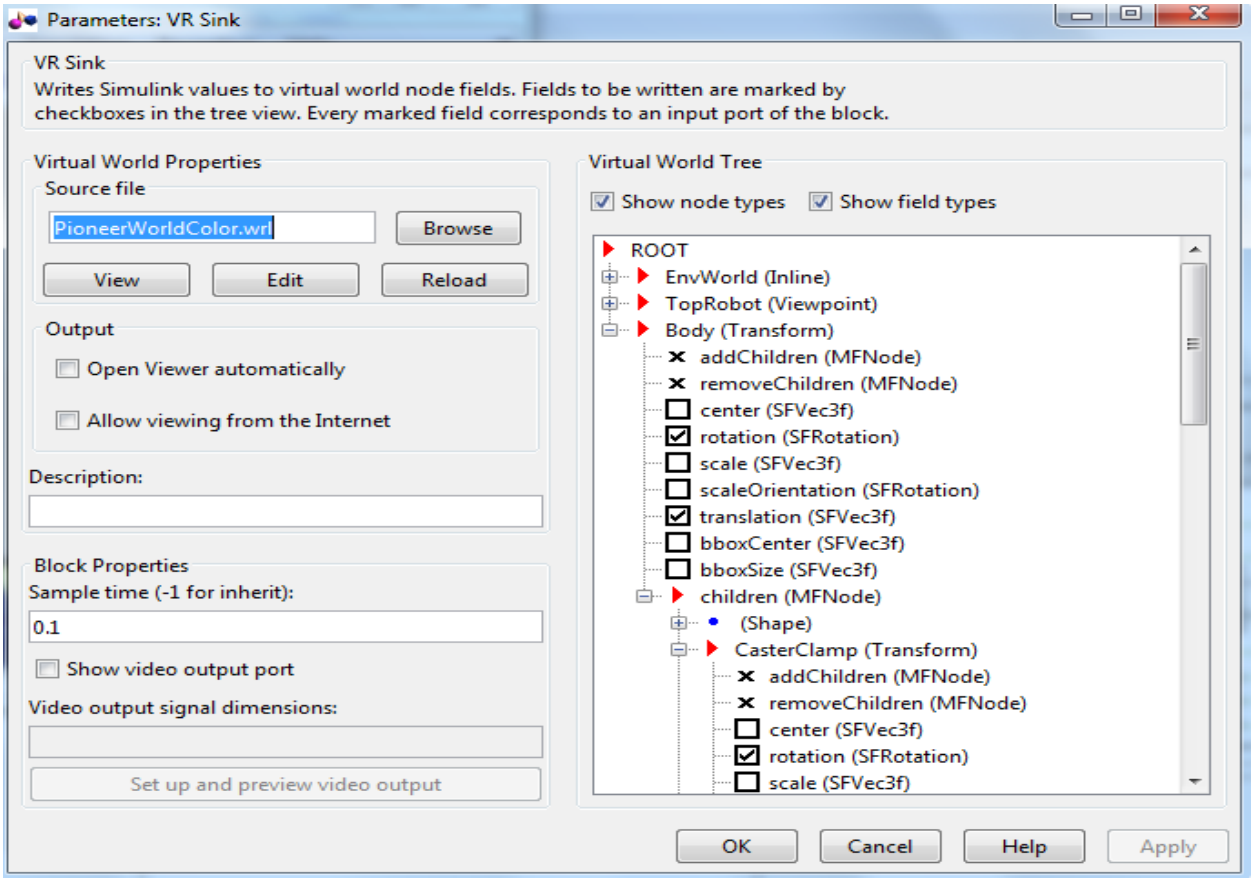
```
clear all
close all
clc
%% Trajectory 3rd Order
theta0 = 5; % Deg
thetaf = 35;
thetad0 = 0;     % Deg/Sec
thetadf = 0;
tstart = 0;      % seconds
tfinal = 100;
[a3,a2,a1,a0] = createTraj3(theta0,thetaf,thetad0,thetadf,tstart,tfinal);
p3=[a3,a2,a1,a0];
t = linspace(tstart,tfinal,150);
% Evaluate the polynomial: Position
pd3= polyder(p3);
pos3 = polyval(pd3,t);
plot(pos3,'r','linewidth',3)
pos1=[1:size(pos3,2); pos3]
save trj_pos pos1;
%% Anfis Training
load trj_pos;
trn_data=pos1(:, 1:100)';
chk_data=pos1(:, 101:150)';
%% Anfis Options
opt = anfisOptions('InitialFIS',4,'EpochNumber',30);
opt.DisplayANFISInformation = 0;
opt.DisplayErrorValues = 0;
opt.DisplayStepSize = 0;
opt.DisplayFinalResults = 0;
opt.ValidationData = chk_data;
[fis,trainError,stepSize,chkFIS,chkError] = anfis(trn_data,opt);
%% Output of Anfis
```

```
figure
out_fis=evalfis(1:100,fis)
plot(out_fis,'r','linewidth',2)
hold on
plot(trn_data(:,2),'b','linewidth',5)
xlabel('X(m)');
ylabel('y(m)');
title('The reference trajectory in xy plane using ANFIS controller');
grid on;
```

APPENDIX C: VR SINK OF THE MOBILE ROBOT (PIONEER 3-DX)




File View Viewpoints Navigation Rendering Simulation Recording Help
Env Front Near
Walk
W E F
Env Front Near
T=0.00
Walk
Pos:[0.00 -1.90 0.30] Dir:[0.00 1.00 0.00]


Parameters: VR Sink
VR Sink
Writes Simulink values to virtual world node fields. Fields to be written are marked by checkboxes in the tree view. Every marked field corresponds to an input port of the block.
Virtual World Properties
Source file
PioneerWorldColor.wrl
Browse
View
Edit
Reload
Output
Open Viewer automatically
Allow viewing from the Internet
Description:
Block Properties
Sample time (-1 for inherit):
0.1
Show video output port
Video output signal dimensions:
Set up and preview video output
Virtual World Tree
Show node types
Show field types
ROOT
EnvWorld (Inline)
TopRobot (Viewpoint)
Body (Transform)
addChildren (MFNode)
removeChildren (MFNode)
center (SFVec3f)
rotation (SFRotation)
scale (SFVec3f)
scaleOrientation (SFRotation)
translation (SFVec3f)
bboxCenter (SFVec3f)
bboxSize (SFVec3f)
children (MFNode)
(Shape)
CasterClamp (Transform)
addChildren (MFNode)
removeChildren (MFNode)
center (SFVec3f)
rotation (SFRotation)
scale (SFVec3f)
OK
Cancel
Help
Apply

APPENDIX D: MATLAB FUNCTIONS FOR DATA TRAINING

Function 1: Data Training MATLAB function for ANFIS_Position1

```
function p1 = importfile(filename, startRow, endRow)
%IMPORTFILE Import numeric data from a text file as a matrix.
%   P1 = IMPORTFILE(FILENAME) Reads data from text file FILENAME for the
%   default selection.
%
%   P1 = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME.
%
% Example:
%   p1 = importfile('p1.dat', 1, 28085);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2018/07/06 11:42:00

%% Initialize variables.
delimiter = ',';
if nargin<=2
    startRow = 1;
    endRow = inf;
end

%% Format for each line of text:
%   column1: double (%f)
%   column2: double (%f)
%   column3: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%f%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1,
'Delimiter', delimiter, 'TextType', 'string', 'EmptyValue', NaN,
'HeaderLines', startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-
startRow(block)+1, 'Delimiter', delimiter, 'TextType', 'string',
'EmptyValue', NaN, 'HeaderLines', startRow(block)-1, 'ReturnOnError', false,
'EndOfLine', '\r\n');
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end
```

```
%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Create output variable
p1 = table(dataArray{1:end-1}, 'VariableNames',
{'VarName1','VarName2','e18'});
```

Function 2: Data Training MATLAB function for ANFIS_Position2

```
function p2 = importfile(filename, startRow, endRow)
%IMPORTFILE Import numeric data from a text file as a matrix.
%   P2 = IMPORTFILE(FILENAME) Reads data from text file FILENAME for the
%   default selection.
%
%   P2 = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME.
%
% Example:
%   p2 = importfile('p2.dat', 1, 28085);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2018/07/06 11:57:12

%% Initialize variables.
delimiter = ',';
if nargin<=2
    startRow = 1;
    endRow = inf;
end

%% Format for each line of text:
%   column1: double (%f)
%   column2: double (%f)
%   column3: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%f%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');

%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1,
'Delimiter', delimiter, 'TextType', 'string', 'EmptyValue', NaN,
'HeaderLines', startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
for block=2:length(startRow)
```

```
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-
startRow(block)+1, 'Delimiter', delimiter, 'TextType', 'string',
'EmptyValue', NaN, 'HeaderLines', startRow(block)-1, 'ReturnOnError', false,
'EndOfLine', '\r\n');
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Create output variable
p2 = table(dataArray{1:end-1}, 'VariableNames',
{'VarName1','VarName2','e18'});
```

Function 3: Data Training MATLAB function for ANFIS_Orientation

```
function p3 = importfile(filename, startRow, endRow)
%IMPORTFILE Import numeric data from a text file as a matrix.
%   P3 = IMPORTFILE(FILENAME) Reads data from text file FILENAME for the
%   default selection.
%
%   P3 = IMPORTFILE(FILENAME, STARTROW, ENDROW) Reads data from rows
%   STARTROW through ENDROW of text file FILENAME.
%
% Example:
%   p3 = importfile('p3.dat', 1, 28085);
%
%    See also TEXTSCAN.

% Auto-generated by MATLAB on 2018/07/06 11:57:37

%% Initialize variables.
delimiter = ',';
if nargin<=2
    startRow = 1;
    endRow = inf;
end

%% Format for each line of text:
%   column1: double (%f)
%   column2: double (%f)
% For more information, see the TEXTSCAN documentation.
formatSpec = '%f%f%[^\n\r]';

%% Open the text file.
fileID = fopen(filename,'r');
```

```
%% Read columns of data according to the format.
% This call is based on the structure of the file used to generate this
% code. If an error occurs for a different file, try regenerating the code
% from the Import Tool.
dataArray = textscan(fileID, formatSpec, endRow(1)-startRow(1)+1,
'Delimiter', delimiter, 'TextType', 'string', 'EmptyValue', NaN,
'HeaderLines', startRow(1)-1, 'ReturnOnError', false, 'EndOfLine', '\r\n');
for block=2:length(startRow)
    frewind(fileID);
    dataArrayBlock = textscan(fileID, formatSpec, endRow(block)-
startRow(block)+1, 'Delimiter', delimiter, 'TextType', 'string',
'EmptyValue', NaN, 'HeaderLines', startRow(block)-1, 'ReturnOnError', false,
'EndOfLine', '\r\n');
    for col=1:length(dataArray)
        dataArray{col} = [dataArray{col};dataArrayBlock{col}];
    end
end

%% Close the text file.
fclose(fileID);

%% Post processing for unimportable data.
% No unimportable data rules were applied during the import, so no post
% processing code is included. To generate code which works for
% unimportable data, select unimportable cells in a file and regenerate the
% script.

%% Create output variable
p3 = table(dataArray{1:end-1}, 'VariableNames', {'VarName1','e17'});
```