Design of a Cognitive Neural Predictive Controller for Mobile Robot

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD) to:
Electronic and Computer Engineering
School of Engineering and Design
Brunel University
United Kingdom

by:
Ahmed Sabah Al-Araji
B.Sc. (Hons), M.Sc. (Hons)

September
2012

Design of a Cognitive Neural Predictive Controller for Mobile Robot

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD) to:

Electronic and Computer Engineering
School of Engineering and Design
Brunel University
United Kingdom

by:

Ahmed Sabah Al-Araji

B.Sc. (Hons), M.Sc. (Hons)

Supervised by:

Prof. Hamed S. Al-Raweshidy

Dr. Maysam F. Abbod

September
2012

ABSTRACT

In this thesis, a cognitive neural predictive controller system has been designed to guide a nonholonomic wheeled mobile robot during continuous and non-continuous trajectory tracking and to navigate through static obstacles with collision-free and minimum tracking error. The structure of the controller consists of two layers; the first layer is a neural network system that controls the mobile robot actuators in order to track a desired path. The second layer of the controller is cognitive layer that collects information from the environment and plans the optimal path. In addition to this, it detects if there is any obstacle in the path so it can be avoided by re-planning the trajectory using particle swarm optimisation (PSO) technique.

Two neural networks models are used: the first model is mdified Elman recurrent neural network model that describes the kinematic and dynamic model of the mobile robot and it is trained off-line and on-line stages to guarantee that the outputs of the model will accurately represent the actual outputs of the mobile robot system. The trained neural model acts as the position and orientation identifier. The second model is feedforward multi-layer perceptron neural network that describes a feedforward neural controller and it is trained off-line and its weights are adapted on-line to find the reference torques, which controls the steady-state outputs of the mobile robot system. The feedback neural controller is based on the posture neural identifier and quadratic performance index predictive optimisation algorithm for N step-ahead prediction in order to find the optimal torque action in the transient to stabilise the tracking error of the mobile robot system when the trajectory of the robot is drifted from the desired path during transient state.

Three controller methodologies were developed: the first is the feedback neural controller; the second is the nonlinear PID neural feedback controller and the third is nonlinear inverse dynamic neural feedback controller, based on the back-stepping method and Lyapunov criterion.The main advantages of the presented approaches are to plan an optimal path for itself avoiding obstructions by using intelligent (PSO) technique as well as the analytically derived control law, which has significantly high computational accuracy with predictive optimisation technique to obtain the optimal torques control action and lead to minimum tracking error of the mobile robot for different types of trajectories.

The proposed control algorithm has been applied to monitor a nonholonomic wheeled mobile robot, has demonstrated the capability of tracking different trajectories with continuous gradients (lemniscates and circular) or non-continuous gradients (square) with bounded external disturbances and static obstacles. Simulations results and experimental work showed the effectiveness of the proposed cognitive neural predictive control algorithm; this is demonstrated by the minimised tracking error to less than (1 cm) and obtained smoothness of the torque control signal less than maximum torque (0.236 N.m), especially when external disturbances are applied and navigating through static obstacles.

Results show that the five steps-ahead prediction algorithm has better performance compared to one step-ahead for all the control methodologies because of a more complex control structure and taking into account future values of the desired one, not only the current value, as with one step-ahead method. The mean-square error method is used for each component of the state error vector to compare between each of the performance control methodologies in order to give better control results.

To

My Respected Parents and Parents-in-Law.

My Beautiful Beloved Wife "Shaimaa".

&

My Life Flowers "Danya" and "Yousif".

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and gratitude to my supervisor, Prof Hamed S. Al-Raweshidy, for his guidance, support and encouragement throughout my thesis work.

I would also like to express my gratitude and thank my advisor, Dr. Maysam F. Abbod for his explicit support and help throughout my thesis organisation.

Thanks are also due to the Electronic and Computer Engineering Department, School of Engineering and Design at Brunel University, for providing all the requirements for this research course, especially laboratories.

I would further like to thank Prof Kais Al-Ibrahimy, Control and Systems Engineering Department, University of Technology, Iraq for his introduction and encouragement for studying for my PhD in the UK.

Finally, I owe the greatest debt of gratitude to my family. I would like to thank my parents and parents-in-law for a lifetime of support and encouragement.

I would like to thank my wife "Shaimaa" for supporting me at all times.

TABLE OF CONTENTS

LIST OF SYMBOLS

Symbol	Definition
α	The feedback gain of the self-connections at the context layer.
β	The connection weight from the hidden units to the context units at the context layer for posture identifier.
δ_m	The degree of mobility for mobile robot.
δ_M	The degree of manoeuvrability for mobile robot.
δ_s	The steerable standard wheels for mobile robot.
$\Delta\sigma$	The variable parameter that allows identification of optimal trajectory.
$\Delta\psi$	The variable parameter that allows identification of optimal trajectory.
$(\Delta k_x, \Delta k_y, \Delta k_\theta)$	The change of the nonlinear feedback acceleration controller gains.
Δt	The sampling period between two sampling times.
ξ	Desired damping coefficient.
η	The learning rate.
θ	The robotic orientation angle.
$\theta_{mt,N}$	Orientation posture identifier future values for N step-ahead prediction.
$\theta_{rt,N}$	Orientation desired future values for N step-ahead prediction.
λ	The vector of constraint forces.
σ	Constant gain.
τ	Input torque vector.
τ_1	The feedback left torque control action in the transient period.
τ_2	The feedback right torque control action in the transient period.
τ_a	Angular torque of mobile robot.
τ_d	Bounded unknown disturbances.
τ_I	Linear torque of mobile robot.
τ_L	The torque of left wheel of the mobile robot.
τ_R	The torque of right wheel of the mobile robot.
$\tau_{ref1}(k)$	The left reference torque.
$\tau_{ref2}(k)$	The right reference torque.
$\tau'_{1t,N}$	The feedback left torque control action for N step-ahead prediction.
$\tau'_{2t,N}$	The feedback right torque control action for N step-ahead prediction.
φ	The points number for the deleted track.
ϕ_{lm}	The bias of the obstacle neural network model.
$A(q)$	The matrix associated with the constraints.
a	The neuron in the hidden layer of feedforward controller.
a_i, b_i, c_i, d_i	The cubic spline equation parameters.
$B(q)$	The input transformation matrix.
C	The number of the context nodes of posture identifier.
CA_{Fit}	The fitness function of the collision avoidance.
CA_{Fit1}	Collision avoidance via-point y_i should not be in the obstacle region.

CA_{Fit2}	Collision avoidance section $y_i y_{i+1}$ should not intersect obstacle region.
C_o	The output of the obstacle neural network model.
$C(q,\dot{q})$	The centripetal and carioles matrix.
c	The centre of mass of the mobile robot.
c_1 and c_2	The acceleration constants.
D	The estimated length of the two-segment track.
d	The number of the y-axis points
E	Cost function of posture identifier learning.
Ec	Cost function of feedforward controller learning.
$EX_{m,t,N}$	Error X-coordinate posture identifier future values for N step-ahead prediction.
$EY_{m,t,N}$	Error Y-coordinate posture identifier future values for N step-ahead prediction.
$E\theta_{m,t,N}$	Error orientation posture identifier future values for N step-ahead prediction.
$f(.)$	Neural activation function.
f and g	Two vectors of Jacobi-Lie-Bracket.
Fit	The final fitness function.
$\overline{G}$	The input vector of posture identifier.
$G(q)$	The gravitational torques vector.
$gbest_d$	The best particle among all particles in the population.
H	Denotes nonlinear node with sigmoid function.
$h_c^o(k)$	The output of the context unit of posture identifier.
$h_c(k)$	The output of the hidden unit of posture identifier.
hc_a	The output of the neuron of hidden layer of feedforward controller.
h_j	The output of the neuron of hidden layer.
I	Inertia of the mobile robot.
i	The number of the input nodes of posture identifier.
IH_m	The weighted input of neuron of the middle layer.
J	The quadratic performance index for multi input/output system.
J_1	Objective function for N step-ahead prediction
j	The number of neuron in the hidden layer of posture identifier.
K	Control gains k_x, k_y, k_θ .
K_d	The derivative gain of the PID controller.
K_i	The integral gain of the PID controller.
K_p	The proportional gain of the PID controller.
k	The number of samples for mobile robot.
$(k_x^*, k_y^*, k_\theta^*)$	Gains are determined by comparing the actual and the desired characteristic polynomial equations.
L	The distance between the two wheels for mobile robot.
Li	The linear node.
L_{ob}	The obstacle's length.
L_{robot}	The length of the mobile robot cart.
M	The mass of the mobile robot.

MD_{Fit}	The fitness function of the minimum distance.
$M(q)$	Symmetric positive definite inertia matrix.
m	The number of the neurons in the middle layer.
N	The number of steps-ahead.
Np	The number of pulses.
n	The number of the input nodes of feedforward controller.
nh	The number of the hidden nodes of posture identifier.
nhc	The number of the hidden nodes of feedforward controller.
np	The number of patterns.
npc	The number of feedforward controller patterns.
$[O, X, Y]$	The global coordinate frame.
(ob_x, ob_y)	The obstacle centre point.
Oc_b	The output of the neuron of output layer of feedforward controller.
OH_m	The output of neuron of the middle layer.
O_k	The output of the neuron of output layer.
p	The parameter of curve shape of activation function.
$pbest_i$	The best previous weight of i^{th} particle.
pop	The population of the particle.
q	The pose vector in the surface.
q_e	The configuration error vector of mobile robot.
$\dot{q}_e$	The derivation configuration error vector for the mobile robot.
q_{em}	The posture identifier configuration error vector.
q_m	The outputs of the posture identifier model.
q_o	The initial pose of the mobile robot.
q_r	Virtual desired posture trajectory vector of the mobile robot.
R and Q	Positive weighting factors.
R	Instantaneous curvature radius of the robot trajectory (distance from the ICR or ICC to the midpoint between the two wheels (c).
R_o	The transformation matrix.
r	Radius of the wheel for mobile robot.
r_1 and r_2	The random numbers function.
S	Standard form of the controllability.
Sg	The number of segments.
T	The time needed to travel the two segments track.
T_{CL}	The time of PULSOUT command left wheel.
T_{Co}	The time of Code overhead and it is equal to 1.6msec.
T_{CR}	The time of PULSOUT command right wheel.
T_{Pd}	The time of PUASE duration and it is equal to 20msec.
T_s	Sampling time.
T_{TIL}	The total time of instructions loop.
t	Real time.
u(t)	The output of the PID controller in time domain.
V	Lyapunov functions.
$\dot{V}$	Time derivative of Lyapunov functions.

$\overline{Vb}$	Weight vector of the posture identifier hidden layers.
$\overline{Vbc}$	Weight vector of the hidden layers of feedforward controller.
VC	Weight matrix of the posture identifier context layers.
$Vcont$	Weight matrix of the hidden layers of feedforward controller.
VH	Weight matrix of the posture identifier hidden layers.
V_I	Linear velocity of the mobile robot (m/sec).
$\dot{V}_I$	The linear acceleration of the differential wheeled mobile robot.
V_{IMax}	The maximum allowed mobile robot linear velocity.
$V_{i,d}^k$	The velocity of i^{th} particle at k iteration.
V_L	Linear velocity of the left wheel (m/sec).
Vp_m	The vertices points of the obstacle.
V_R	Linear velocity of the right wheel (m/sec).
V_W	Angular velocity of the mobile robot (rad/sec).
$\dot{V}_W$	The angular acceleration of the differential wheeled mobile robot.
v_c	The velocity control vector.
v_r	The reference linear velocity.
W	Weight matrix of the output posture identifier layer.
$\overline{Wb}$	Weight vector of the output posture identifier layer.
$\overline{Wbc}$	Weight vector of the output layer of feedforward controller.
$Wcont$	Weight matrix of the output layer of feedforward controller.
W_{ob}.	The obstacle's width.
W_{robot}	The width of the mobile robot cart.
w_n	The characteristic frequency.
w_r	The reference angular velocity.
w_{rMax}	The maximum allowed mobile robot angular velocity.
$X_{mt,N}$	X-coordinate posture identifier future values for N step-ahead prediction.
$X_{rt,N}$	X-coordinate desired future values for N step-ahead prediction.
x and y	Coordinates of point c for mobile robot.
xw_m	The weight of the obstacle neural network for x_i input.
$Y_{mt,N}$	Y-coordinate posture identifier future values for N step-ahead prediction.
$Y_{rt,N}$	Y-coordinate desired future values for N step-ahead prediction.
$y_{i,d}^k$	The position of i^{th} particle at k iteration.
yw_m	The weight of the obstacle neural network for y_i input.
Z^{-1}	One delay sample in Z-domain.

LIST OF ABBREVIATIONS

Abbreviation	Definition
ACO-MH	Ant Colony Optimisation Meta-Heuristic.
AI	Artificial Intelligence.
ANFIS	Adaptive Neural Fuzzy Inference System.
ANNs	Artificial Neural Networks.
APRBS	Amplitude Modulated Pseudo Random Binary Sequence.
Boe-Bot	Board of Education Robotics.
BPA	Back Propagation Algorithm.
CBAPPA	Cognitive Based Adaptive Path Planning Algorithm.
CCW	Counter-Clockwise Direction.
CW	Clockwise Direction.
EBNN	Explanation-Based Neural Network Learning Algorithm.
FFNC	The Feedforward Neural Controller
GA	Genetic Algorithm.
H_∞	H-infinity.
ICC	Instantaneous Centre of Curvature.
ICR	Instantaneous Centre of Rotation.
MIMO	Multi-Input Multi-Output System.
MLP	Multi-layer Perceptron.
MPC	Model Predictive Control.
MSE	Mean Square Error.
NIDFC	Nonlinear Inverse Dynamic Feedback Controller.
NNs	Neural Networks.
PIC	Perpherial Interface Controller.
PID	Proportion, Integral and Derivative controller.
PRBS	Pseudo Random Binary Sequence.
PSO	Particle Swarm Optimisation.
PWM	Pulse Width Modulation.
RBF	Radial Basis Function.
RNNs	Recurrent Neural Networks.
RPF	Repulsive Penalty Function.
RPM	Revolution Per Minute.
USB	Universal Serial Bus.

LIST OF TABLES

LIST OF FIGURES

Chapter One

Introduction to Mobile Robots

1.1. Introduction

In general, there are many definitions of robots. There seems to be some difficulty in suggesting an accurate meaning or definition for the word ‘robot’, and various alternatives exist, differing according to points of view. Some view a robot through the aspect of reprogram ability, while others are more concerned with the manipulation of robot behaviours (intelligence). The popular understanding of the term ‘robot’ generally connotes some anthropomorphic (human-like) appearance, as reflected in spin-off terms like robot ‘arms’ for welding. Lexically, the word ‘robot’ is actually derived from the Czech word ‘*robota*’, which is loosely translated as ‘menial laborer’ [1]. *Robota* connotes the compulsory service (i.e. slavery) of a physical agent, which can generate an intelligent connection between perception and action. However, the current notation of robot includes programmable, mechanical capable and flexible.

Basically, robots can be divided into two categories, fixed and mobile robots. Fixed robots are mounted on a fixed surface and materials are brought to the workspace near the robot. A fixed robot is normally used in mass production, as in car factories, for welding or stamping. Mobile robots have the capability to move around in their environment and are not fixed to one physical location; therefore, the mobile robot can be defined as a mechanical device that performs automated tasks, whether according to direct human supervision, a pre-defined program, or a set of general guidelines, using artificial intelligence (AI) techniques [2].

Mobility is the robot's capability to move from one place to another in unstructured environments to a desired target. Mobile robots can be categorized into wheeled, tracked or legged robots, and they are more useful than fixed robots. Mobile robots are increasingly used in industry, in service robotics, for factories (e.g. in delivering components between assembly stations) and in difficult to access or dangerous areas such

as space, military environments, nuclear-waste cleaning and for personal use in the forms of domestic vacuum cleaners and lawn mowers [2, 3, 4].

Over the last decade, the design and engineering of mobile robot systems acting autonomously in complex, dynamic and uncertain environments has remained a challenge. Such systems have to be able to perform multiple tasks, and therefore must integrate a variety of knowledge-intensive information processes on different levels of abstractions guaranteeing real-time execution, robustness, adaptability and scalability [5]. Cognitive control methodologies have been proven to be a source of inspiration and guidance to overcome current limitations in the controller for more complex and adaptive systems, and these methodologies have been utilising mobile robot systems as demonstrators, serving as an important proof of the concept for cognitive models [6, 7, 8]. Cognitive technical systems are capable to perceive the environment as well as to aggregate knowledge and to structure this knowledge autonomously. Enhancing a complex technical system by cognitive capabilities enables the system to interact in open and unknown environments. According to the definition in [9], cognitive systems (biological or technical) are characterised by their learning capabilities, representation of relevant aspects of the environment and ability to realize situated behaviour.

1.2. Motivations

Ever since the advent of mobile robots and throughout the years the problems facing robot control have been the subject of continuous and vigorous research, and while the basic prediction problems in mapping, path planning and trajectory tracking were well understood and solved, plenty of open problems are still there waiting to be addressed. These include computational complexity, linearisation effect, association of measurements to features, detection of loops in the robot's path and maintaining topological consistency as the maps are getting very large [10, 11, 12, 13, 14, 15, 16, 17, 18].

The fundamental essence of the motivation for this work is to generate an optimal path for mobile robots in order to avoid static obstacles and track the desired trajectory with minimum tracking error and save the battery energy of the robot through the design of an adaptive robust controller.

1.3. Aim and Objectives of the Research

The aim of the research is to construct a cognitive neural predictive control methodology that gives a control signal to the actuators of the nonholonomic wheeled mobile robot, which achieves the following objectives:

1- Detect the obstacle in the path and plan the optimum path for collision-free path between a starting and target location.

2- Overcome the challenge in modelling and identifying the position and orientation of the mobile robot for N step-ahead prediction.

3- The motion of the mobile robot model will track the desired lemniscates, circular and square trajectories with minimum tracking error in the steady state and controlled transient behaviour in order to minimise the travel time and travelling distance of the mobile robot.

4- Minimisation of a weighted quadratic cost function in the errors which are defined as the difference between the desired trajectory (position and orientation) and the actual outputs of the mobile robot, also quadratic in the control signals with a finite number of time steps-ahead in order to reduce the control effort, with vanishes spikes action for saving the energy of batteries of the mobile robot driving circuit, and maintaining the smoothness and continuity of the motion of the mobile robot without slippage or deviation from the desired trajectory.

5- Investigation of the controller robustness performance through adding boundary unknown disturbances and static obstacles.

6- Verification of the controller adaptation performance by changing the initial pose state and changing the static obstacles locations.

7- Validation of the controller capability of tracking any trajectories with continuous and non-continuous gradients to avoid the static obstacles in the path.

1.4. Contributions to Knowledge

The fundamental essence of the contribution of this work is to modify and improve the performance of traditional PID and modern controllers by employing the theory of cognitive neural network topology as a basis for a learning and adapting system with the capability of planning fairly optimal trajectories that are desired to guide and manoeuvre a nonholonomic wheeled mobile robot through pre-defined trajectories (Lemniscates and Circular as a continuous and Square as a non-continuous gradients path) with collision-free navigation. This is done by finding the optimal torque control action that will minimise the tracking error (the travel time and travelling distance) of the mobile robot by utilising an optimisation predictive algorithm that works to curtail the error between desired trajectory and actual mobile robot trajectory, in addition to reducing the control effort (i.e. reducing the spike of torque control signals and thus saving battery energy of the mobile robot system) encountered, even in the presence of obstacles in the path.

1.5. Achievements

Cognitive neural predictive controller plans the desired trajectory of optimal smoothness for the mobile robot and executes the trajectory tracking by generating optimal torque control actions using the analytically derived control law, which has significantly high computational accuracy with five steps-ahead predictive optimisation technique. Simulation results and experimental work show the effectiveness of the three proposed control methodologies, which achieved excellent tracking for Lemniscates and Circular as a continuous and Square as a non-continuous gradients trajectories with collision-free path for the actual mobile robot and reduced the tracking error to less than 1cm. The actions of the proposed controller were small smooth values of the torques for right and left wheels without sharp spikes and less than maximum torque (0.235N.m); therefore, the velocity of the actual mobile robot does not exceed the maximum value (0.165m/sec).

In addition to that, the proposed control algorithm achieves a collision-free path by re-planning the primary path to generate the smoothness desired trajectory without overshooting. The minimum number of the segments based on cubic spline interpolation technique and particle swarm optimisation with kinematic constraints on velocity enables

the mobile robot to avoid static obstacles and to keep the tracking error at less than 1cm with minimum distance.

1.6. Thesis Organisation

The remainder of the thesis is organised as follows:

Chapter two reviews previous studies related to this field.

Chapter three describes the kinematics and dynamics mathematical model of the nonholonomic wheeled mobile robot.

Chapter four describes the use of modified Elman recurrent neural networks to learn and to act as posture neural identifier that overcome the challenge in modelling and identifying the position and orientation of the mobile robot for N step-ahead prediction and simulation results for modelling and identifying of the mobile robot.

Chapter five represents the core of the controller. In this chapter, it is suggested that using an adaptive neural predictive controller with three control methodologies and optimisation predictive algorithm attains specific benefits towards a systematic engineering design procedure for adaptive neural predictive control system and simulation results.

Chapter six represents the core of the cognitive neural controller. In this chapter, it is suggested that cognitive neural predictive controller be used with two proposed path planning algorithms for detection of obstacles in the desired trajectory and re-planning the desired path with simulation results.

Chapter seven presents experimental work and simulation results of the proposed cognitive neural predictive controller.

Finally, chapter eight contains the conclusions of the entire work and suggestions for future work.

Chapter Two

Overview of Control Methodologies for Mobile Robots

2.1. Introduction

There are many elements of the mobile robot that are critical to robust mobility, such as the kinematics of locomotion, sensors for determining the robot's environmental context and techniques for localising with respect to its map. The general control scheme for the mobile robots navigation is shown in Figure 2.1 [2].

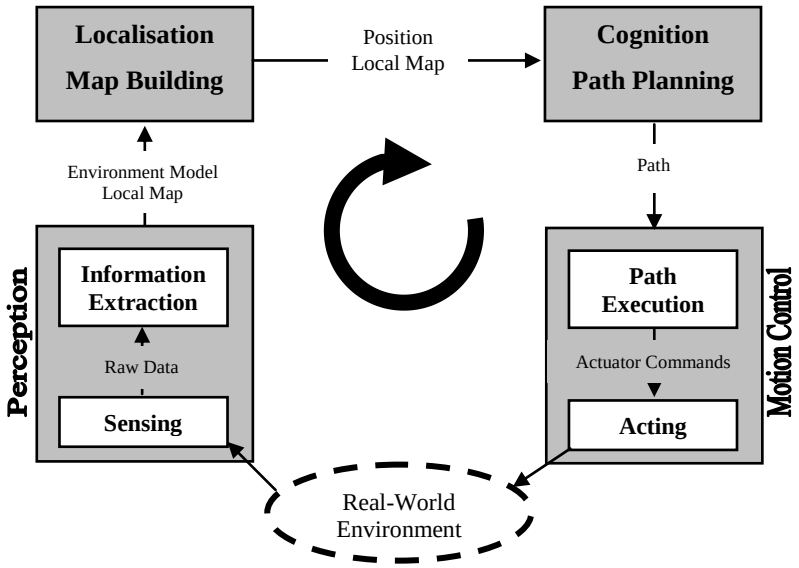


Figure (2.1): The general control scheme for the mobile robots navigation [2].

It consists of four blocks: perception - the robot must interpret its sensors to extract meaningful data; localization - the robot must determine its position in the environment; cognition - the robot must decide how to act to achieve its goals; and motion control - the robot must modulate its motor outputs to achieve the desired trajectory. If all four blocks can be verified, navigation will be successful [2].

2.2. Control Problems of the Mobile Robot

Most wheeled mobile robots can be classified as nonholonomic mechanical systems. Control problems stem from the motion of a wheeled mobile robot in a plane possessing three degrees of freedom, while it has to be controlled using only two control inputs under the nonholonomic constraint. These control problems have recently attracted considerable attention in the control community [19]. During the past few years, many methods have been developed to solve mobile robot control problems which can be classified into three categories:

The first category is the sensor-based control approach for navigation problems of the mobile robot on interactive motion planning in dynamic environments and obstacle motion estimation [20, 21, 22]. Since the working environment for mobile robots is unstructured and may change with time, the robot must use its on-board sensors to cope with dynamic environmental changes, while for proper motion planning (such as environment configuration prediction and obstacle avoidance motion estimation) it uses sensory information [23, 24, 25, 26].

The second category for navigation problems of the mobile robot is path planning. The path is generated based on a prior map of the environment and used certain optimisation algorithms based on a minimal time, minimal distance and minimal energy performance index. Many methods have been developed for avoiding both static and moving obstacles, as presented in [27, 28, 29, 30].

The third category for the navigation problems of mobile robot is designing and implementing the motion control that mobile robot needs to execute the desired path accurately and to minimise tracking error.

2.3. Tracking Error of the Mobile Robot

The trajectory planning of mobile robot aims to provide an optimal path from an initial pose to a target pose [31]. Optimal trajectory planning for a mobile robot provides a path, which has minimal tracking error and shortest driving time and distance. Tracking errors of mobile robots cause collisions with obstacles due to deviations from the planned path, which also causes the robot to fail to accomplish the mission successfully. It also causes

an increase in traveling time, as well as the travel distance, due to the additional adjustments needed to satisfy the driving states. There are three major reasons for increasing tracking error for mobile robots:

The first major reason for tracking error is the discontinuity of the rotation radius on the path of the differential driving mobile robot. The rotation radius changes at the connecting point of the straight line route and curved route, or at a point of inflection. At these points it can be easy for differential driving mobile robot to secede from its determined orbit due to the rapid change of direction [32]. Therefore, in order to decrease tracking error, the trajectory of the mobile robot must be planned so that the rotation radius is maintained at a constant, if possible.

The second reason for increasing of tracking error due is that the small rotation radius interferes with the accuracy of driving the mobile robot. The path of the mobile robot can be divided into curved and straight-line segments. While tracking error is not generated in the straight-line segment, significant error is produced in the curved segment due to centrifugal and centripetal forces, which cause the robot to slide over the surface [33].

The third of the major reason for increasing tracking error due to the rotation radius is not constant such as complex curvature or random curvature (i.e. the points of inflection exist at several locations, necessitating that the mobile robot wheel velocities need to be changed whenever the rotation radius and travelling direction are changed [33, 34].

In fact, the straight-line segment can be considered as a curved segment whose rotation radius is infinity. As the tracking error becomes larger at the curved segment, the possibility of a tracking error increases with the decrease of the rotation radius of the curved path. Note that a relatively small error occurs at the straight-line path. Tracking error can be reduced by applying the control methodologies.

2.4. Control Strategies and Methodologies for Mobile Robot

Control system development is necessary to guarantee success in tracking the mobile robot on the desired trajectory. While there is an abundance of control methodologies for trajectory tracking that can be applied to track the mobile robot, the main aim is to control the system cheaply and effectively without sacrificing the robustness and reliability of the

controller. The difference in the tracking control strategy implemented depends mostly on how the system is modelled and how the information is obtained. The control strategies for such a system can be classified into two distinct sections, namely a linear control model and a nonlinear control model. Linear control strategies use linearised dynamics behaviour for a certain operating point, depending on the mathematical model of the mobile robot system. Nonlinear control strategies use the dynamics model of the mobile robot system in designing a controller with variables parameters depending on the mathematical model of the robot system. Many researchers and designers have recently showed an active interest in the development and applications of nonlinear control methodologies for three main reasons [35]:

a- Improvement of existing control system.

b- Analysis of hard nonlinearities.

c- Dealing with model uncertainties.

2.4.1. Previous Works Related to Artificial Intelligent Techniques

The traditional control methods for path tracking the mobile robot used linear or non-linear feedback control, while AI controllers were carried out using neural networks or fuzzy inference [36, 37]. Neural networks (NNs) are recommended for AI control as a part of a well-known structure with adaptive critic [38]. Much research has been done on the applications of neural networks for control of nonlinear dynamics model of mobile robot systems and has been supported by two of the most important capabilities of neural networks: their ability to learn and their good performance for the approximation of nonlinear functions [39, 40, 41]. The neural network based control of mobile robots has recently been the subject of intense research [42]. It is usual to work with kinematic models of mobile robot to obtain stable motion control laws for path following or goal reaching [43, 44]. Two novel dual adaptive neural control schemes were proposed for dynamics control of nonholonomic mobile robots [45]. The first scheme was based on Gaussian radial basis function artificial neural networks (ANNs) and the second on sigmoid multi-layer perceptron (MLP). ANNs were employed for real-time

approximation of the robot's nonlinear dynamics functions, which were assumed to be unknown. The tracking control of nonholonomic wheeled mobile robot using two cascade controllers were proposed in [46]. The first stage was fuzzy controller and second stage was adaptive neural fuzzy inference system (ANFIS) controller for the solution of path tracking problem of mobile robots.

2.4.2. Previous Works Related to Sliding Mode Technique

A trajectory tracking control for a nonholonomic mobile robot by the integration of a kinematics controller and neural dynamics controller based on the sliding mode theory was presented in [47]. A discrete-time sliding model control for the trajectory tracking problem of nonholonomic wheeled mobile robot was presented in [48], in which the control algorithm was designed in discrete-time domain in order to avoid problems caused by discretisation of continuous-time controller. A new trajectory tracking control system of nonholonomic wheeled mobile robot was presented in [49] using sliding-mode control and torque control based on radial basis function (RBF) neural networked control.

2.4.3. Previous Works Related to Back-Stepping Technique

Integrating the neural networks into back-stepping technique to improve learning algorithm of analogue compound orthogonal networks and novel tracking control approach for nonholonomic mobile robots was proposed in [50]. Rotation error transformation and back-stepping technique were exploited to achieve the control law for solving the problem of trajectory tracking for nonholonomic wheeled mobile robot for tracking the desired trajectory was explained in [51]. Using the idea of back-stepping in the feedback control law of nonholonomic mobile robot, which employs the disturbance observer control approach to design an auxiliary wheel velocity controller, in order to make the tracking errors as small as possible in consideration of unknown bounded disturbance in the kinematics of the mobile robot, was proposed in [52].

2.4.4. Previous Works Related to Predictive Controller

There are other techniques for trajectory tracking controllers, such as predictive control technique. Predictive approaches to trajectory tracking seem to be very promising because

the desired trajectory is known beforehand. Model predictive trajectory tracking control was applied to a mobile robot, whereby linearised tracking error dynamics were used to predict future system behaviour and a control law was derived from a quadratic cost function, penalizing the system tracking error and the control effort [53, 54]. In addition, an adaptive trajectory-tracking controller based on the robot kinematics and dynamics was proposed in [55, 56, 57, 58] and its stability property was proved using the Lyapunov theory.

2.4.5. Previous Works Related to PID Controller

An adaptive controller of nonlinear PID-based neural networks was developed for the velocity and orientation tracking control of a nonholonomic mobile robot [59]. PID controller and simple linearised model of the mobile robot were used as a simple and effective solution for the trajectory tracking problem of a mobile robot [60, 61]. A self-tuning PID control strategy based on a deduced model was proposed for implementing a motion control system that stabilises the two-wheeled vehicle and follows the reference motion commands [62, 63].

2.4.6. Previous Works Related to Different Types of Controllers

A variable structure control algorithm was proposed to study the trajectory tracking control based on the kinematics model of a 2-wheel differentially driven mobile robot by using of the back-stepping method and virtual feedback parameter with the sigmoid function [64]. There are other techniques for path-tracking controllers, such as the trajectory-tracking controllers designed by pole-assignment approach for mobile robot model presented in [65]. The model of the mobile robot from combination of kinematics and robust H-infinity (H_∞) dynamics tracking controller to design the kinematics tracking controller by using the Lyapunov stability theorem was proposed in [66].

2.4.7. Previous Works Related to Path Planning Algorithms

In addition to that, one of the main tasks for mobile robot is to decide how to plan to reach the target point according to some optimal standards in unknown, partly unknown

or known environment with collision-free navigation. In recent years, many studies on robot motion planning used various approaches, such as the explanation-based neural network learning algorithm (EBNN), an approach to learn indoor robot navigation tasks through trial-and-error method by applying EBNN in the context of reinforcement learning, which allows the robot to learn control using dynamic programming as explained in [67]. A mobile robot's local path-planning algorithm based on the human's heuristic method was explained in [68], and used a new laser finder, one of the active vision systems for a free-ranging mobile robot. The problem of generating smooth trajectories for a fast-moving mobile robot in a cluttered environment was proposed in [69] by using smoothing polynomial curves methods. The cognitive based adaptive path planning algorithm (CBAPPA) was proposed in [7] for solving the path planning problems for autonomous robotic applications. [12] explained a method to solve the problem of path planning for mobile robots based on Ant Colony Optimisation Meta-Heuristic (ACO-MH) to find the best route according to certain cost functions. A comparative study of three proposed algorithms that used occupancy grid map of environments to find a path for mobile robot from the given start to the goal was explained in [70]. The use of Hopfield-type neural network dynamics for real-time collision-free robot path generation in an arbitrarily varying environment was presented in [71, 72]. This was also used in neural networks in the algorithm for multi-path planning in unknown environment for mobile robot based on genetic optimisation algorithm, as proposed in [73, 74, 75, 76]. In addition, the genetic algorithm (GA), particle swarm optimisation (PSO) algorithm is widely used in the mobile robot path planning in order to find the optimal path and to avoid the static or dynamic obstacles [77, 78 79, 80].

2.5. Summary

This chapter described the main points of navigation problems for mobile robots and the methods that have been developed for solving mobile robot control problems, and explained the major reasons for increasing tracking error for mobile robots and many methods related to this work. In addition, the chapter presented some of the neural networks methodologies for path planning of mobile robots that have used optimisation algorithms, such as genetic algorithm and particle swarm optimisation techniques.

Chapter Three

Locomotion, Kinematics and Dynamics of Differential Wheeled Mobile Robots

3.1. Introduction

This chapter describes the basic concept of locomotion for wheeled mobile robot and explains the kinematics and dynamics model for the nonholonomic wheeled mobile robot under the nonholonomic constraint of pure rolling and non-slipping.

3.2. Locomotion of Differential Wheeled Mobile Robots

A mobile robot needs locomotion mechanisms to move unbounded throughout its environment. However, there are large varieties of ways to move, and so the section of a robot's approach to locomotion is an important aspect of mobile robot design. The most popular locomotion mechanism in mobile robotics and fabricated vehicles is the wheel. It can achieve good efficiencies and needs a relatively simple mechanical implementation. While designing a wheeled mobile robot, the main features concern the type of wheels, their arrangement and their actuation systems (eventual steering mechanisms). These parameters define the mobility characteristic of the robot. Some robots are omni-directional [81, 82, 83]; that is, they can instantaneously move in any direction along the plane not considering their orientation around the vertical axis.

However, these kinds of mobile robots are uncommon because they need particular wheels or mechanical structures. Other kinds of wheeled robots have a car-like configuration, that is, they have four wheels (two of them on a steering mechanism) [84, 85], that permit a translation in the frontal direction of the vehicle and a rotation around a

point that depends on the wheels' steering angle. It is easy to understand that these kinds of robots are not omni-directional; in fact, supposing that the wheels do not slide on the floor, a car-like robot can not slip in its lateral direction.

The two-wheel differential-drive robot is the most popular kind of mobile robot [36, 86, 87]; a robot with two wheels actuated by two independent motors with a coincident rotation axis. However, because the mobile robots need three ground contact points for stability, one or two additional passive castor wheels or slider points may be used for balancing for differential-drive robot.

There is a very large range of possible wheel configurations when one considers possible techniques for mobile robot locomotion, as there are four major different wheel types [2]:

a- Standard wheel: two degrees of freedom; rotation around the (motorised) wheel axle and the contact point.

b- Caster wheel: two degrees of freedom; rotation around an offset steering joint.

c- Swedish wheel: three degrees of freedom; rotation around the (motorised) wheel axle, around the rollers, and around the contact point.

d- Ball or spherical wheel: realization technically difficult.

In the differential wheeled mobile robot there are two types of wheel, the standard wheel and the castor wheel, each of which has a primary axis of rotation and is thus highly directional, as shown in Figure 3.1 [2].

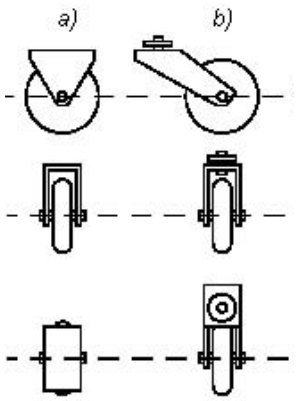


Figure 3.1: The two basic wheel types. (a) standard wheel and (b) castor wheel [2].

To move in a different direction, the wheel must be steered first along a vertical axis. The key difference between these two wheels is that the standard wheel can accomplish this steering motion with no side-effects, as the centre of rotation passes through the contact patch with the ground, whereas the castor wheel rotates around an offset axis, causing a force to be imparted to the robot chassis during steering [2]. The number of variations in wheel configuration for rolling mobile robots is quite large. However, there are three fundamental characteristics of a robot are governed by these choices: stability, maneuverability and controllability.

3.2.1. Stability

Stability requires a minimum number of wheels (three) in order to guarantee stable balance, with the additional important proviso that the centre of gravity must be contained within the triangle formed by the ground contact points of the wheels. Nevertheless, two wheeled differential-drive robot can achieve stability if the centre of mass is below the wheel axle [53].

3.2.2. Manoeuvrability

The manoeuvrability of a robot is a combination of the mobility δ_m available based on the kinematic sliding constraints of the standard wheels, plus the additional freedom contributed by steering and spinning the steerable δ_s standard wheels. These are depicted in Figure 3.2.

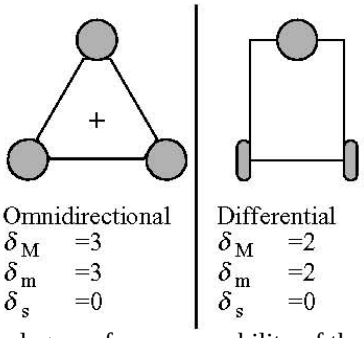


Figure 3.2: The degree of manoeuvrability of the mobile robot [2].

Some robots are omni-directional [81, 82, 83], meaning that they can be move at any time in any direction along the ground plane (x,y) regardless of the orientation of the robot around its vertical axis. Therefore the degree of maneuverability δ_M of omni-

directional mobile robot is equal to three because the degree of mobility δ_m is equal to three and the degree of steer ability is equal to zero. In differential drive mobile robot [36, 86], the degree of manoeuvrability is equal to two because the result of degree of mobility is equal to two and the degree of steer ability is equal to zero.

3.2.3. Controllability

There is generally an inverse correlation between controllability and manoeuvrability because the controllability problem for mobile robot systems is subject to kinematic constraints on the velocity and its application to collision-free path planning. In a differential drive mobile robot, the two motors attached to the two wheels must be driven along exactly the same velocity profile, which can be challenging considering variations between wheels, motors and environmental differences. Controlling an omni-directional robot for specific direction of travel is more difficult and often less accurate when compared to less manoeuvrable designs [2].

3.3. Kinematics Models of Differential Wheeled Mobile Robots

Kinematics is most basic study of how mechanical systems behave. In mobile robotics the mechanical behaviour of the robot must be understood, both in order to design suitable mobile robots for tasks and to understand how to generate control software, for instance mobile robot hardware [88, 89]. In terms of the motion without considering the forces that affect it, the study of the mathematics of motion is called kinematics, and it deals with the geometric relationships that control the mobile robot system. To explain the term nonholonomic wheeled mobile robot see appendix A. The differential drive robot platform as a rigid body on wheels, operating on a horizontal plane, is shown in Figure 3.3 [90].

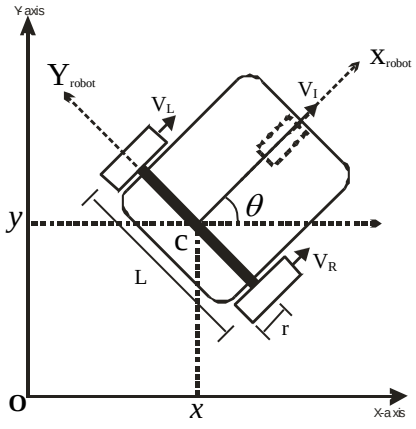


Figure 3.3: Schematic of the nonholonomic mobile robot [90].

The total dimensionality 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. The mobile robot platform has two identical parallel, non-deformable rear wheels, two independent DC motors, which are controlled by two independent analogous DC motors left and right wheels for motion and orientation and one castor front wheel for stability. The two wheels have the same radius denoted by r and L as the distance between the two wheels. The centre of mass of the mobile robot is located at point c centre of axis of wheels.

The pose of mobile robot in the global coordinate frame $[O, X, Y]$ and the pose vector in the surface is defined as $q = (x, y, \theta)^T$, where x and y are coordinates of point c and θ is the robotic orientation angle measured from X-axis and these three generalized coordinates can describe the configuration of the mobile robot. Sometimes roller-balls can be used but from the kinematics point of view, there are no differences in calculations because it can rotate freely in all directions. It is assumed that the masses and inertias of the wheels are negligible and that the centre of mass of the mobile robot is located in the middle of the axis connecting the rear wheels.

The plane of each wheel is perpendicular to the ground and the contact between the wheels and the ground is ideal for rolling without skidding or slipping; the velocity of the centre of mass of the mobile robot is orthogonal to the rear wheels' axis. Under these assumptions the wheel has two constraints. The first constraint enforces the concept of rolling contact. This means that the wheel must roll in the appropriate direction motion. The second constraint enforces the concept of no lateral slippage. This means that the wheel must not slide orthogonally to the wheel plane [2]. An idealized rolling wheel is shown in Figure 3.4 [91].

.

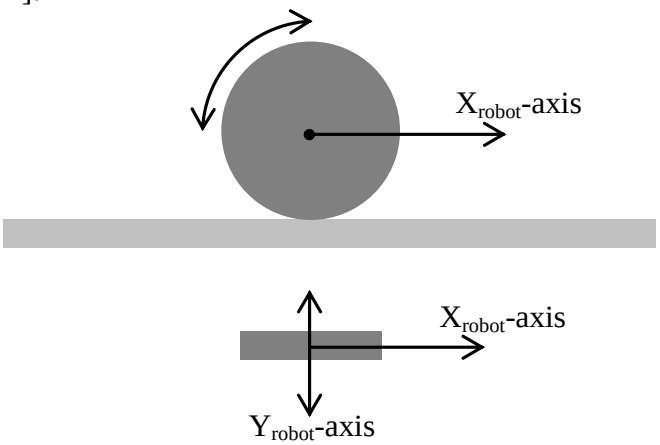


Figure 3.4: Idealized rolling wheel [91].

The wheel is free to rotate about its axis (Y_{robot} axis). The robot exhibits preferential rolling in one direction (X_{robot} axis) and a certain amount of lateral slip. The parameters of the kinematics model are shown in Table 3.1.

Table 3.1: Parameters of the model's kinematics.

r	Radius of each wheel (m).
L	Distance between the driving wheels along the Y_{robot} (m).
V_w	Angular velocity of the robot (rad/sec).
V_I	Linear velocity of the robot along the X_{robot} (m/sec).
V_L	Linear velocity of the left wheel (m/sec).
V_R	Linear velocity of the right wheel (m/sec).
c	Centre of axis of rear wheels.
$[O,X,Y]$	The pose of mobile robot in the global coordinate frame.
$(x,y,\theta)^T$	The pose vector in the surface (the current position and orientation).
R	Instantaneous curvature radius of the robot trajectory (distance from the ICR or ICC to the midpoint between the two wheels (c).

At each instant in time the left and right wheels follow a path as show in Figure 3.5 [91] that moves around the instantaneous centre of curvature (ICC) or the instantaneous centre of rotation (ICR) with the same angular rate [92].

$$V_w(t) = \frac{d\theta(t)}{dt} \tag{3.1}$$

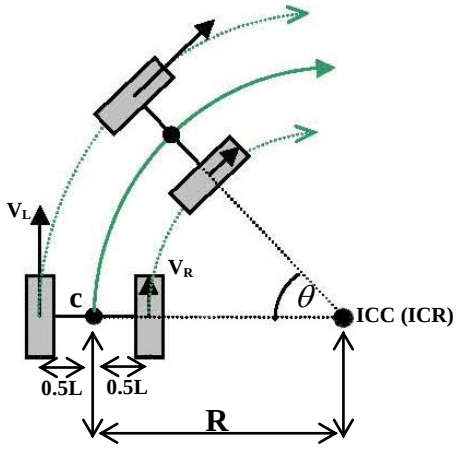


Figure 3.5: Instantaneous centre of rotation (ICR) [91].

thus:

$$V_I(t) = V_w(t)R(t) \tag{3.2}$$

$$V_L(t) = (R(t) + \frac{L}{2})V_w(t) \tag{3.3}$$

$$V_R(t) = (R(t) - \frac{L}{2})V_w(t) \tag{3.4}$$

By solving equations 3.3 and 3.4, the instantaneous curvature radius of the robot trajectory relative to the mid-point axis is given as equation 3.5.

$$R(t) = \frac{L(V_L(t) + V_R(t))}{2(V_L(t) - V_R(t))} \tag{3.5}$$

The angular velocity of the mobile robot is given as equation 3.6.

$$V_w(t) = \frac{V_L(t) - V_R(t)}{L} \tag{3.6}$$

The linear velocity of the mobile robot is given as equation 3.7.

$$V_I(t) = \frac{V_L(t) + V_R(t)}{2} \tag{3.7}$$

By changing the velocities of the two wheels, the instantaneous centre of rotation will move and different trajectories will be followed, as shown in Figure 3.6.

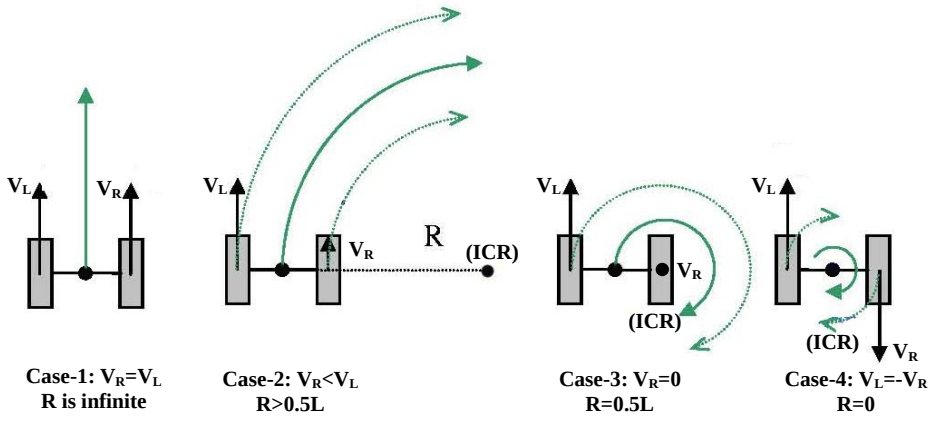


Figure 3.6: The different moving possibilities for differential drive [91].

A differential drive mobile robot is very sensitive to the relative velocity of the two wheels. Small differences between the velocities provided to each wheel cause different trajectories. The kinematics equations in the world frame can be represented as follows [92, 93, 94]:

$$\dot{x}(t) = V_I(t)\cos\theta(t) \tag{3.8}$$

$$\dot{y}(t) = V_I(t)\sin\theta(t) \tag{3.9}$$

$$\dot{\theta}(t) = V_w(t) \tag{3.10}$$

Integrating equations 3.8, 3.9 and 3.10 and they were obtained as follows:

$$x(t) = \int_0^t V_I(\tau)\cos\theta(\tau)d\tau + x_o \tag{3.11}$$

$$y(t) = \int_0^t V_I(\tau)\sin\theta(\tau)d\tau + y_o \tag{3.12}$$

$$\theta(t) = \int_0^t V_w(\tau)d\tau + \theta_o \tag{3.13}$$

where (x_o, y_o, θ_o) is the initial pose.

Formulas equations 3.11, 3.12 and 3.13 are valid for all robots capable of moving in a particular direction $\theta(t)$ at a given velocity $V_I(t)$. For the special case of differential drive robot, based on equations 3.6 and 3.7, it can be concluded that:

$$x(t) = 0.5\int_0^t [V_R(\tau) + V_L(\tau)]\cos\theta(\tau)d\tau + x_o \tag{3.14}$$

$$y(t) = 0.5\int_0^t [V_R(\tau) + V_L(\tau)]\sin\theta(\tau)d\tau + y_o \tag{3.15}$$

$$\theta(t) = \frac{1}{L}\int_0^t [V_L(\tau) - V_R(\tau)]d\tau + \theta_o \tag{3.16}$$

For a practical realization, the formula equations 3.14, 3.15 and 3.16 can be rewritten for discrete timing thus:

$$x(k) = 0.5\sum_{i=1}^{k} [V_R(i) + V_L(i)]\cos\theta(i)\Delta t + x_o \tag{3.17}$$

$$y(k) = 0.5\sum_{i=1}^{k} [V_R(i) + V_L(i)]\sin\theta(i)\Delta t + y_o \tag{3.18}$$

$$\theta(k) = \frac{1}{L}\sum_{i=1}^{k} [V_L(i) - V_R(i)]\Delta t + \theta_o \tag{3.19}$$

where $x(k), y(k), \theta(k)$ are the components of the pose at the k step of the movement and Δt is the sampling period between two sampling times. In the computer simulation, the currently form of the pose equations as follows:

$$x(k) = 0.5[V_R(k) + V_L(k)]\cos\theta(k)\Delta t + x(k-1) \tag{3.20}$$

$$y(k) = 0.5[V_R(k) + V_L(k)]\sin\theta(k)\Delta t + y(k-1) \tag{3.21}$$

$$\theta(k) = \frac{1}{L}[V_L(k) - V_R(k)]\Delta t + \theta(k-1) \tag{3.22}$$

These are the equations used to build a model of the mobile robot and used to simulate the mobile robot in Matlab program.

The kinematics equations 3.8, 3.9 and 3.10 can be represented as follows [64, 95, 96]:

$$\dot{q} = S(q)V \tag{3.23}$$

where $q(t)$ and $\dot{q}(t) \in \Re^{3\times1}$ are defined as:

$$q = (x, y, \theta)^T,\ \dot{q} = (\dot{x}, \dot{y}, \dot{\theta})^T \tag{3.24}$$

and the velocity vector $V(t) \in \Re^2$ is defined as:

$$V = \begin{bmatrix} V_I & V_w \end{bmatrix}^T \tag{3.25}$$

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_I(t) \\ V_w(t) \end{bmatrix} \tag{3.26}$$

The mobile robot can be steered to any position in a free workspace when it is assumed that the mobile robot wheels are ideally installed in each away that they have ideal rolling without slipping [90]. But the wheel of a mobile robot cannot move sideways; the freedom of motion is limited because no lateral slippage is possible, and the wheel must not slide orthogonally to the wheel plane, and the velocity of the point c of the mobile robot must be in the direction of the axis of symmetry (x-axis), which is referred as the nonholonomic constraint [2, 64, 95, 96], as shown in equation 3.27:

$$-\dot{x}(t)\sin\theta(t) + \dot{y}(t)\cos\theta(t) = 0 \tag{3.27}$$

Thus the constraint in equation 3.27 for mobile robot can be expressed as matrix:

$$A^T(q)\dot{q} = 0 \tag{3.28}$$

where

$$A^T(q) = [-\sin\theta(t) \quad \cos\theta(t) \quad 0] \tag{3.29}$$

To check controllability of the nonlinear time variant system in equation 3.30, the accessibility rank condition is globally satisfied and has implied controllability [96].

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} 0.5\cos\theta(t) & 0.5\cos\theta(t) \\ 0.5\sin\theta(t) & 0.5\sin\theta(t) \\ 1/L & -1/L \end{bmatrix} \begin{bmatrix} V_L \\ V_R \end{bmatrix} \tag{3.30}$$

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix}=\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ 1/L\end{bmatrix}V_L(t)+\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ -1/L\end{bmatrix}V_R(t) \tag{3.31}$$

Let

$$f=\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ 1/L\end{bmatrix} \text{ and } \quad g=\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ -1/L\end{bmatrix} \tag{3.32}$$

By using Jacobi-Lie-Bracket of f and g to find $[f,g]$ see appendix B.

$$[f,g]^i=\sum_{j=1}^{n}(f^j\frac{\partial g^i}{\partial q^j}-g^j\frac{\partial f^i}{\partial q^j}) \tag{3.33}$$

$$[f,g]=\begin{bmatrix}[f,g]^1\\ [f,g]^2\\ [f,g]^3\end{bmatrix}=\begin{bmatrix}\frac{-1}{L}\sin\theta(t)\\ \frac{1}{L}\cos\theta(t)\\ 0\end{bmatrix} \tag{3.34}$$

$$rank\{f,g,[f,g]\}=rank\begin{bmatrix}0.5\cos\theta(t) & 0.5\cos\theta(t) & \frac{-1}{L}\sin\theta(t)\\ 0.5\sin\theta(t) & 0.5\sin\theta(t) & \frac{1}{L}\cos\theta(t)\\ 1/L & -1/L & 0\end{bmatrix} \tag{3.35}$$

If the determent of the matrix in equation 3.35 is equal to $(1/L^2)\neq 0$, then the rank of matrix is equal to 3; therefore, the system in equation 3.31 is controllable. For investigation of trajectory tracking stability for the mobile robot, that the desired trajectory can be described by a virtual desired robot with a state vector $q_r=[x_r,y_r,\theta_r]^T$, an input vector $u_r=[V_{Ir},V_{Wr}]^T$ as shown in Figure 3.7 [64], under the assumption that the virtual desired robot is not at rest ($V_{Ir}=0, V_{Wr}=0$) when $t\to\infty$, has to find a control law $u=[V_I,V_w]^T$, such that $\lim_{t\to\infty}(q_r-q)=0$, with any initial robot posture $q(0)$.

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix}=\begin{bmatrix}\cos\theta_r(t) & 0\\ \sin\theta_r(t) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}V_{Ir}(t)\\ V_{wr}(t)\end{bmatrix} \tag{3.36}$$

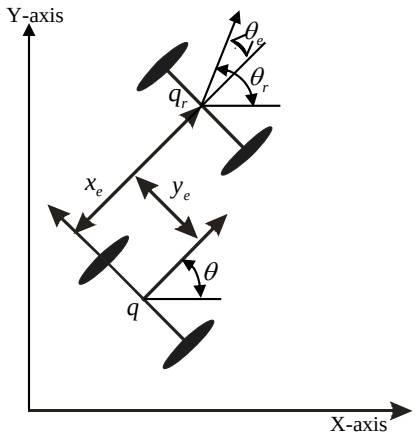


Figure 3.7: Configuration error of mobile robot [64].

In the local coordinates with respect to the body of the mobile robot, the configuration error $q_e = [x_e, y_e, \theta_e]^T$ can be presented by $q_e = R_o(q_r - q)$:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{3.37}$$

where R_o is the transformation matrix.

By taking the time derivative of equation 3.37 and rearranging with equations 3.26 and 3.36, the configuration error for the robot becomes (see appendix C):

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} V_w y_e - V_I + V_{Ir}\cos\theta_e \\ -V_w x_e + V_{Ir}\sin\theta_e \\ V_{wr} - V_w \end{bmatrix} \tag{3.38}$$

To reform equation (3.38) in the standard form, u_e will be defined

$$u_e = \begin{bmatrix} u1_e \\ u2_e \end{bmatrix} = \begin{bmatrix} -V_I + V_{Ir}\cos\theta_e \\ V_{wr} - V_w \end{bmatrix} \tag{3.39}$$

Then the configuration error model of equation 3.38 becomes

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & V_w & 0 \\ -V_w & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} 0 \\ V_{Ir}\sin\theta_e \\ 0 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u1_e \\ u2_e \end{bmatrix} \tag{3.40}$$

By linearising the configuration error model of equation 3.40 about the equilibrium point, the following is obtained:

$$\begin{bmatrix} \dot{x}_e \\ \dot{y}_e \\ \dot{\theta}_e \end{bmatrix} = \begin{bmatrix} 0 & V_{wr} & 0 \\ -V_{wr} & 0 & V_{Ir} \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u1_e \\ u2_e \end{bmatrix} \tag{3.41}$$

The controllability of the system (3.41) can easily checked, by using standard form of the controllability $S = [B \quad AB \quad A^2B]$ [97], and checking that the determent is not equal to zero $|S| \neq 0$.

$$|S| = (1 + (V_{wr})^4 + (V_{wr} \times V_{Ir})^2) \times ((V_{wr})^2 + (V_{Ir})^2) \tag{3.42}$$

However, from equation 3.42, when the virtual desired robot stops ($V_{wr} = 0, V_{Ir} = 0$), the controllable property is lost.

3.4. Dynamics Model of Differential Wheeled Mobile Robots

In order to produce motion, forces must be applied to the mobile robot model. These forces are modelled by studying of the motion of the dynamic model of the differential wheeled mobile robot shown in Figure 3.2. It deals with mass, forces and speed associated with this motion. The dynamics model can be described by the following dynamic equations based on Euler Lagrange formulation [48, 56, 59, 98].

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) + \tau_d = B(q)\tau - A(q)\lambda \tag{3.43}$$

where q is the three dimensional vector of configuration variables equation 3.24.

$M(q) \in \Re^{n \times n}$ is a symmetric positive definite inertia matrix.

$C(q,\dot{q}) \in \Re^{n \times n}$ is the centripetal and carioles matrix.

$G(q) \in \Re^{n}$ is the gravitational torques vector

$\tau_d \in \Re^{n \times 1}$ denotes bounded unknown disturbances including unstructured un modeled dynamics.

$B(q) \in \Re^{n \times r}$ is the input transformation matrix.

$\tau \in \Re^{r \times 1}$ is input torque vector.

$A(q) \in \Re^{n \times m}$ is the matrix associated with the constraints of equation 3.29.

$\lambda \in \Re^{m \times 1}$ is the vector of constraint forces.

Remark 1: The plane of each wheel is perpendicular to the ground and the contact between the wheels and the ground is pure rolling and non-slipping, hence the velocity of

the centre of the mass of the mobile robot is orthogonal to the rear wheels' axis. The trajectory of mobile robot base is constrained to the horizontal plane, therefore, $G(q)$ is equal to zero.

Remark 2: In this dynamic model, the passive self-adjusted supporting wheel influence is not taken into consideration as it is a free wheel. This significantly reduces the complexity of the model for the feedback controller design. However, the free wheel may be a source of substantial distortion, particularly in the case of changing its movement direction. This effect is reduced if the small velocity of the robot is considered [56, 59].

Remark 3: The centre of mass for mobile robot is located in the middle of axis connecting the rear wheels in c point, as shown in Figure 3.3, therefore $C(q,\dot{q})$ is equal to zero.

The dynamical equation of the differential wheeled mobile robot can be expressed as:

$$\begin{bmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & I \end{bmatrix}\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} + \tau_d = \frac{1}{r}\begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ \frac{L}{2} & \frac{-L}{2} \end{bmatrix}\begin{bmatrix} \tau_L \\ \tau_R \end{bmatrix} + \begin{bmatrix} -\sin\theta \\ \cos\theta \\ 0 \end{bmatrix}\lambda \tag{3.44}$$

where τ_L and τ_R are the torques of left and right motors respectively.

M and I present the mass and inertia of the mobile robot respectively.

$$M\ddot{x} + \tau d = \frac{\cos\theta}{r}\tau_L + \frac{\cos\theta}{r}\tau_R - \lambda\sin\theta \tag{3.45}$$

$$M\ddot{y} + \tau d = \frac{\sin\theta}{r}\tau_L + \frac{\sin\theta}{r}\tau_R + \lambda\cos\theta \tag{3.46}$$

$$I\ddot{\theta} + \tau d = \frac{L}{2r}\tau_L - \frac{L}{2r}\tau_R \tag{3.47}$$

Assume

$$\tau_I = \frac{\tau_L + \tau_R}{r} \quad \text{and} \quad \tau_a = \frac{L}{2r}(\tau_L - \tau_R) \tag{3.48}$$

where τ_I and τ_a are linear and angular torques respectively.

$$\ddot{x} = \frac{\cos\theta}{M}\tau_I - \frac{\sin\theta}{M}\lambda - \frac{\tau_d}{M} \tag{3.49}$$

$$\ddot{y} = \frac{\sin\theta}{M}\tau_I + \frac{\cos\theta}{M}\lambda - \frac{\tau_d}{M} \tag{3.50}$$

$$\ddot{\theta} = \frac{\tau_a}{I} - \frac{\tau_d}{I} \tag{3.51}$$

In order to reach the normal form, the following transformation is used by differentiating equation 3.23 as:

$$\ddot{q} = \dot{S}(q)V + S(q)\dot{V} \tag{3.52}$$

Therefore

$$\ddot{x} = -\sin(\theta)\dot{\theta}V_I + \cos(\theta)\dot{V}_I \tag{3.53}$$

$$\ddot{y} = \cos(\theta)\dot{\theta}V_I + \sin(\theta)\dot{V}_I \tag{3.54}$$

$$\ddot{\theta} = \dot{V}_W \tag{3.55}$$

Comparing equations 3.53, 3.54 and 3.55 with equations 3.49, 3.50 and 3.51 respectively, the followings can be written:

$$-\sin(\theta)\dot{\theta}V_I + \cos(\theta)\dot{V}_I = \frac{\cos\theta}{M}\tau_I - \frac{\sin\theta}{M}\lambda - \frac{\tau_d}{M} \tag{3.56}$$

$$\cos(\theta)\dot{\theta}V_I + \sin\theta\dot{V}_I = \frac{\sin\theta}{M}\tau_I + \frac{\cos\theta}{M}\lambda - \frac{\tau_d}{M} \tag{3.57}$$

$$\dot{V}_W = \frac{\tau_a}{I} - \frac{\tau_d}{I} \tag{3.58}$$

Multiplying equation 3.56 by $\cos\theta$ and equation 3.57 by $\sin\theta$

$$-\sin(\theta)\cos(\theta)\dot{\theta}V_I + (\cos(\theta))^2\dot{V}_I = \frac{(\cos(\theta))^2}{M}\tau_I - \frac{\sin(\theta)\cos(\theta)}{M}\lambda - \frac{\tau_d\cos(\theta)}{M} \tag{3.59}$$

$$\cos(\theta)\sin(\theta)\dot{\theta}V_I + (\sin(\theta))^2\dot{V}_I = \frac{(\sin(\theta))^2}{M}\tau_I + \frac{\sin(\theta)\cos(\theta)}{M}\lambda - \frac{\tau_d\sin(\theta)}{M} \tag{3.60}$$

Adding equations 3.59 and 3.60 the following is obtained:

$$\dot{V}_I = \frac{\tau_I}{M} - \frac{\tau_d(\sin(\theta) + \cos(\theta))}{M} \tag{3.61}$$

Assume τ_d is bounded unknown disturbances.

$$\dot{V}_I = \frac{\tau_I}{M} + \tau_d \tag{3.62}$$

$$\dot{V}_W = \frac{\tau_a}{I} + \tau_d \tag{3.63}$$

where $\dot{V}_I$ and $\dot{V}_W$ are the linear and angular acceleration of the differential wheeled mobile robot.

The dynamics and the kinematics model structure of the differential wheeled mobile robot can be shown in Figure 3.8.

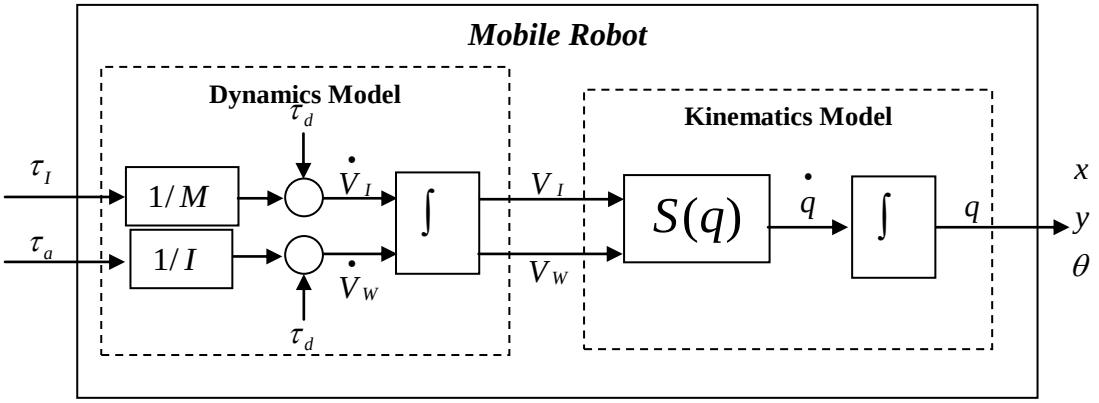


Figure 3.8: The dynamics and the kinematics model structure of the differential wheeled mobile robot.

3.5. Summary

This chapter describes the mathematical model of the nonholonomic wheeled mobile robot that has depended on the kinematics and dynamics analysis under the nonholonomic constraint of pure rolling and non-slipping. Also, it has described the basic concept of the locomotion of the wheeled mobile robot and the fundamental characteristics of the mobile robot (stability, manoeuvrability and controllability).

Chapter Four

Modelling of the Mobile Robot Based on Modified Elman Recurrent Neural Networks Identification

4.1. Introduction

The main goal of this chapter is to describe approaches to neural-network based modelling and identifying that are found to be practically applicable to a reasonably wide class of unknown nonlinear systems. Modelling and identification of nonlinear dynamics systems is a challenging task because nonlinear processes are unique in the sense that they do not share many properties. Also, system identification is one of the important and integral parts of a control system design [99]. The system identification is necessary to establish a model based on which the controller can be designed, and it is useful for tuning and simulation before applying the controller to the real system. System identification is relevant in many applications, including control system design, modelling, simulation, prediction, optimization, supervision, fault detection and diagnosis components [99, 100, 101, 102, 103, 104, 105].

Kinematics and dynamics mobile robot system identification and modelling is a very important step in control applications, since it is a pre-requisite for analysis and control design. Due to the nonlinear nature of most of the systems encountered in many engineering applications, there has been extensive research covering the field of nonlinear system identification. However, mathematical models are the most useful in this respect [106]. To build a mathematical model of the system, one can use the physical laws that govern the system's behaviour. Alternatively, one can observe the signals produced by the

system to known inputs and find a model that best reproduces the observed output. The former approach is called mathematical modelling, and the latter is called identification. If there is insufficient information about the kinematics and dynamics behaviour of the mobile robot system for N step-ahead prediction of the position and orientation, identification is necessary to make accurate model.

This chapter focuses on nonlinear kinematics and dynamics mobile robot system identification to overcome the challenge in identifying the position and orientation of the mobile robot by using modified Elman recurrent neural networks with two stages for learning off-line and on-line and used two-configuration serial-parallel and parallel with back propagation algorithm for learning neural networks.

4.2. Neural Networks and System Identification

The neural networks are a technique for using physical hardware or computer software to model computational properties analogous to some that have been postulated for real networks of nerves, such as the ability to learn and store relationships [107]. Therefore, neural networks have become a very fashionable area of research with a range of potential applications that spans AI, engineering and science [101, 102, 103, 104, 105, 107, 108]. All these applications are dependent upon training the network and adjusting the weights, which define the strength of connection between the neurons in the network.

There are several capabilities for neural networks, the first of which are perhaps the most significant [109, 110]:

- Neural networks are best suited for the control of nonlinear systems because of the flexibly and arbitrarily map nonlinear functions that they contain.

- Particularly well-suited to multivariable applications due to map interactions and cross-couplings readily whilst incorporating many inputs and outputs.

- Learned off-line and subsequently employed either on- or off-line, or they can be learned on-line as part of an adaptive control scheme, or simply a real-time system identifier.

In an attempt to accurately nonlinear model using identification techniques, a wide variety of techniques have been developed for multi-layer perceptron (MLP) in [39, 101, 103, 111, 112] while recurrent neural networks model (RNN) was researched in [113, 114].

Linear models are well established for system identification; however, nonlinear system identification has not received as much attention as linear system identification, due to the difficulty of devising proper models and algorithms to estimate their parameters [115]. Hence, nonlinear systems are normally approximated by using linear models by restricting the range of perturbation to fixed and small range. However, such a model is restricted to system operations within that range, and to develop a globally valid model (valid for all inputs) for nonlinear system, a nonlinear model has to be used.

4.3. Neural Network Topology for Modelling Approach

To describe the kinematics and dynamics model of the mobile robot by using artificial neurons as the basic building element for the development of multi-layered and higher order neural network, the five basic steps shown in Figure 4.1 are used in order to overcome the challenge in the identification and modelling of the mobile robot system.

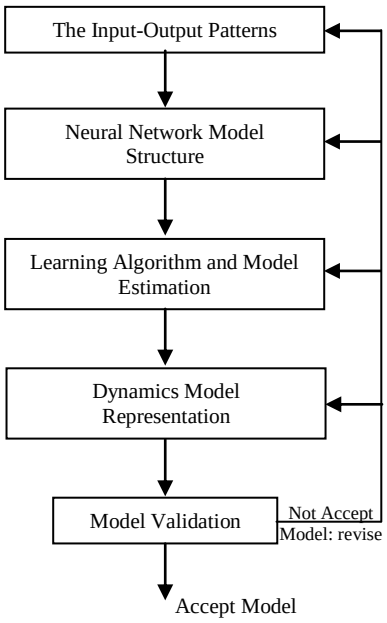


Figure 4.1: Steps of modelling and identifying for mobile robot system.

4.3.1. The Input-Output Patterns

The neural networks can be seen as a system transforming a set of input patterns into a set of output patterns, and such a network can be trained to provide a desired response to a given input. The network achieves such behaviour by adapting its weights during the learning phase on the basis of some learning rules. The training of neural networks often requires the existence of a set of input and output patterns, called the training set, and this kind of learning is called supervised learning [116].

The learning set of input-output patterns for the nonlinear system is collected during the simulations test. One of the most crucial tasks in system identification is the design of appropriate excitation signals for gathering identification data, because nonlinear dynamics model is significantly more complex, and thus the data must contain considerably more information. Consequently, for identification of nonlinear kinematics and dynamics model of mobile robot system, the requirements of a suitable data set are very high. Therefore, an amplitude modulated pseudo random binary signal (APRBS) is used, which is capable of describing the system in all relevant operating conditions. The parameters of this signal, whose spectrum can be easily derived, are chosen according to the dynamics of the system. Frequency properties of the excitation signal and the amplitudes have to be chosen properly to cover all operating regions in order to be able to assess the reliability of the model's predictions [99].

4.3.2. Neural Networks Model Structure

This section focuses on nonlinear MIMO system identification of nonholonomic wheeled mobile robot (position and orientation) using the modified Elman recurrent neural network structure to construct the position and orientation neural networks identifier, as shown in Figure 4.2, which shows that the structure of modified Elman recurrent neural networks consists of the nodes of input layer, hidden layer, context layer and output layer [113, 114]. The input and output units interact with the outside environment, while the hidden and context units do not. The input units are only buffer units, which pass the signal without changing them. The output units are linear units which add the signals fed to them. The hidden units are non-linear activation functions.

The context units are used only to memorise the previous activations of the hidden units in order to increase the speed of learning and minimise the numbers of nodes in hidden layer. The context units can be considered to function as one-step time delay.

The structure shown in Figure 4.2 is based on the following equations [113, 114]:

$$h(k) = F\{VH\overline{G}(k), VC\overline{h^o}(k), bias\overline{Vb}\} \tag{4.1}$$

$$O(k) = (Wh(k), bias\overline{Wb}) \tag{4.2}$$

where *VH*, *VC*, and *W* are weight matrices, $\overline{Vb}$ and $\overline{Wb}$ are weight vectors and *F* is a non-linear vector function. The multi-layered modified Elman neural network, shown in Figure 4.2 is composed of many interconnected processing units called neurons or nodes [113, 114].

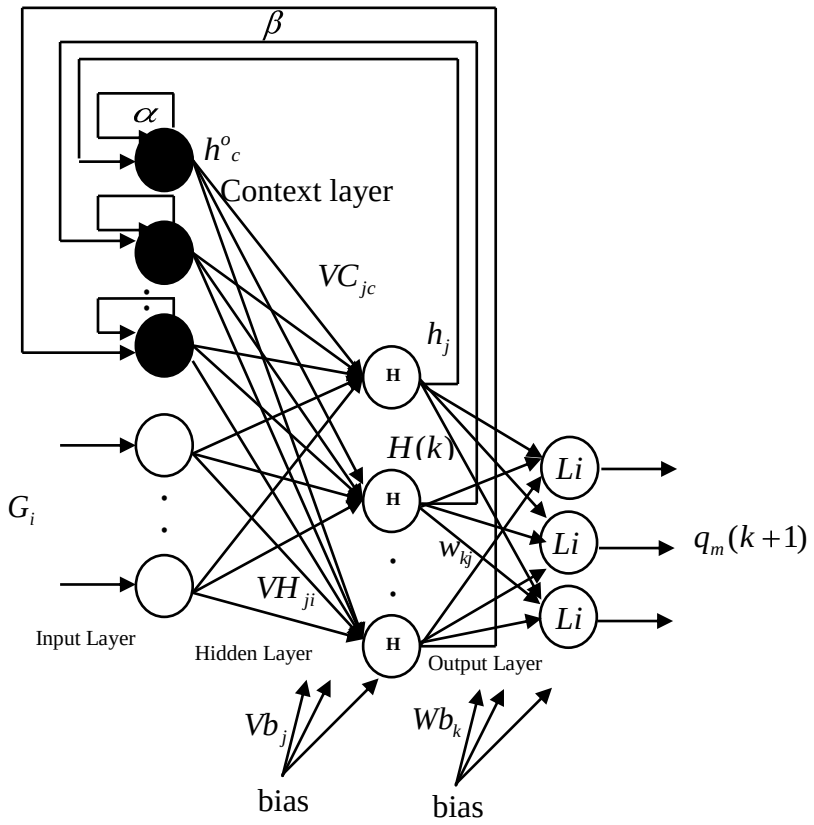


Figure 4.2: The Modified Elman Recurrent Neural Networks [108, 109].

The network weights are denoted as follows:

VH : Weight matrix of the hidden layers.

VC : Weight matrix of the context layers.

$\overline{Vb}$: Weight vector of the hidden layers.

W : Weight matrix of the output layer.

$\overline{Wb}$: Weight vector of the output layer.

Li : Denotes linear node.

H : Denotes nonlinear node with sigmoid function.

Modified Elman recurrent neural network was used as opposed to conventional neural networks in the posture identifier for these reasons as follows [113, 114, 117]:

1- To increase the speed of learning and minimise the numbers of nodes in hidden layer because it has the context units that are used only to memorise the previous activations of the hidden units.

2- To improve the network memory ability, self-connections (α fixed value) are introduced into the context units of the network in order to give these units a certain amount of inertia.

3- To increase the order of the neural model for matching with actual model through self-connection in the context units for the Elman network.

4- To reduces the output oscillation and minimises the error between the actual output system and neural network output, because of the order of neural networks have approached from the actual system order.

The output of the c^{th} context unit in the modified Elman network is given by:

$$h_c^o(k) = \alpha h_c^o(k-1) + \beta h_j(k-1) \tag{4.3}$$

where $h_c^o(k)$ and $h_c(k)$ are respectively the output of the context unit and hidden unit and α is the feedback gain of the self-connections and β is the connection weight from the hidden units j^{th} to the context units c^{th} at the context layer. The value of α and β are selected randomly between (0 and 1), not modified by the training algorithm [113, 114, 117]. To explain these calculations, consider the general j^{th} neuron in the hidden layer shown in Figure 4.3 [113, 114], and the inputs to this neuron consist of an i – dimensional vector and (i^{th}, is the number of the input nodes). Each of the inputs has a weight *VH* and *VC* associated with it.

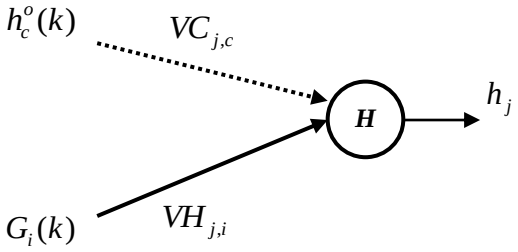


Figure (4.3): Neuron j in the hidden layer.

Here, $\overline{Vb}$ is weight vector for the bias input that it is important input and it's equal to -1 to prevent the neurons being quiescent when all inputs of neural may be equal to zero [118]. The first calculation within the neuron consists of calculating the weighted sum net_j of the inputs as [113, 114]:

$$net_j = \sum_{i=1}^{nh} VH_{ji} \times G_i + \sum_{c=1}^{C} VC_{jc} \times h_c^o + bias \times \overline{Vb}_j \tag{4.4}$$

where

$j=c$. $nh=C$ number of the hidden nodes and context nodes and $\overline{G}$ is the input vector.

The activation function of the hidden nodes is sigmoid function and the activation function of the output nodes is linear function as shown in Figures 4.4a and 4.4b respectively [118].

Next the output of the neuron h_j is calculated as the continuous sigmoid function of the net_j as:

$$h_j = \mathrm{H}(net_j) \tag{4.5}$$

$$\mathrm{H}(net_j) = \frac{2}{1+e^{-net_j}} - 1 \tag{4.6}$$

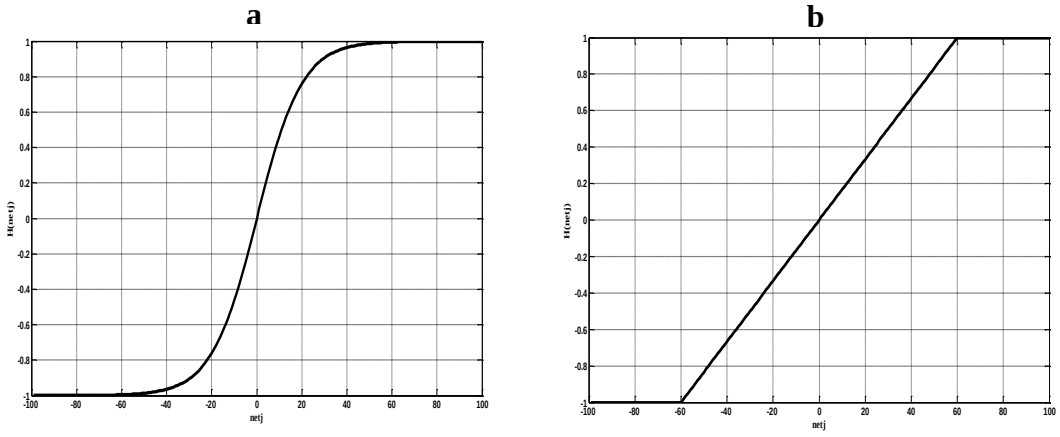


Figure 4.4: Activation function of neural networks (a) Sigmoid function; (b) linear function.

Once the outputs of the hidden layer are calculated, they are passed to the output layer. In the output layer, three linear neurons are used to calculate the weighted sum (neto) of its inputs.

$$\mathrm{neto}_k = \sum_{j=1}^{nh} W_{kj} \times h_j + bias \times \overline{Wb}_k \tag{4.7}$$

where W_{kj} is the weight between the hidden neuron h_j and the output neuron. $\overline{Wb}$ is the weight vector for the output neuron. The three linear neurons then pass the sum ($neto_k$) through a linear function of slope 1 (another slope can be used to scale the output) as:

$$O_k = L(neto_k) \tag{4.8}$$

These outputs of the identifier are the modelling pose vector in the surface is defined as:

$$q_m = (x_m, y_m, \theta_m)^T \tag{4.9}$$

where x_m and y_m are modelling coordinates and θ_m is the modelling orientation angle.

4.3.3. Learning Algorithm and Model Estimation

The dynamic back propagation algorithm is used to adjust the weights of the dynamics modified Elman recurrent neural networks. The sum of the square of the differences between the desired outputs $q = (x, y, \theta)^T$ and network outputs $q_m = (x_m, y_m, \theta_m)^T$ is chosen as criterion for estimating the model performance:

$$E = \frac{1}{np}\sum_{i=1}^{np}((x - x_m)^2 + (y - y_m)^2 + (\theta - \theta_m)^2) \tag{4.10}$$

where *np* is number of patterns and the factor 0.5 is used to help in differentiating the equation 4.10.

The use of the minus sign accounts for the gradient-descent method in weight-space and reflects the search for the direction of the weight change that reduces the value of E objective cost function. The learning rate η has a small value for smoothness weights changing while a large integration step size means that oscillations may occur and stability may be lost. The connection matrix between hidden layer and output layer is W_{kj}, using the chain rule differentiation as follows:

$$\Delta W_{kj}(k+1) = -\eta \frac{\partial E}{\partial W_{kj}} \tag{4.11}$$

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial W_{kj}} \tag{4.12}$$

$$\frac{\partial E}{\partial W_{kj}} = \frac{\partial E}{\partial q_m(k+1)} f'(net_k) h_j \tag{4.13}$$

$$\frac{\partial E}{\partial q_m(k+1)} = \frac{\partial \frac{1}{2}\sum((x-x_m)^2+(y-y_m)^2+(\theta-\theta_m)^2)}{\partial q_m(k+1)} \tag{4.14}$$

$$\Delta W_{kj}(k+1) = \eta \times h_j \times e_k \tag{4.15}$$

$$W_{kj}(k+1) = W_{kj}(k) + \Delta W_{kj}(k+1) \tag{4.16}$$

The connection matrix between input layer and hidden layer is VH_{ji}

$$\Delta VH_{ji}(k+1) = -\eta \frac{\partial E}{\partial VH_{ji}} \tag{4.17}$$

$$\frac{\partial E}{\partial VH_{ji}} = \frac{\partial E}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial VH_{ji}} \tag{4.18}$$

$$\frac{\partial E}{\partial VH_{ji}} = \frac{\partial E}{\partial q_m(k+1)} \sum_{k=1}^{K} W_{kj} f(net_j)' \times G_i \tag{4.19}$$

$$\frac{\partial E}{\partial q_m(k+1)} = \frac{\partial \frac{1}{2}\sum((x-x_m)^2+(y-y_m)^2+(\theta-\theta_m)^2)}{\partial q_m(k+1)} \tag{4.20}$$

$$\frac{\partial E}{\partial VH_{ji}} = f(net_j)' \times G_i \sum_{k=1}^{K} e_k W_{kj} \tag{4.21}$$

$$\Delta VH_{ji}(k+1) = \eta \times f(net_j)' \times G_i \sum_{k=1}^{K} e_k W_{kj} \tag{4.22}$$

$$VH_{ji}(k+1) = VH_{ji}(k) + \Delta VH_{ji}(k+1) \tag{4.23}$$

The connection matrix between context layer and hidden layer is VC_{ji}

$$\Delta VC_{jc}(k+1) = -\eta \frac{\partial E}{\partial VC_{jc}} \tag{4.24}$$

$$\frac{\partial E}{\partial VC_{jc}} = \frac{\partial E}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial h_j} \frac{\partial h_j}{\partial net_c} \frac{\partial net_c}{\partial VC_{jc}} \tag{4.25}$$

$$\frac{\partial E}{\partial VC_{jc}} = \frac{\partial E}{\partial q_m(k+1)} \sum_{k=1}^{K} W_{kj} f(net_j)' \times h_c^o \tag{4.26}$$

$$\frac{\partial E}{\partial q_m(k+1)} = \frac{\partial \frac{1}{2}\sum((x-x_m)^2+(y-y_m)^2+(\theta-\theta_m)^2)}{\partial q_m(k+1)} \tag{4.27}$$

$$\frac{\partial E}{\partial VC_{jc}} = f(net_j)' \times h_c^o \sum_{k=1}^{K} e_k W_{kj} \tag{4.28}$$

$$\Delta VC_{jc}(k+1) = \eta \times f(net_j)' \times h_c^o \sum_{k=1}^{K} e_k W_{kj} \tag{4.29}$$

$$VC_{jc}(k+1) = VC_{jc}(k) + \Delta VC_{jc}(k+1) \tag{4.30}$$

The connection vector between input bias and output layer is $\overline{Wb}_k$

$$\Delta \overline{Wb}_k(k+1) = -\eta \frac{\partial E}{\partial \overline{Wb}_k} \tag{4.31}$$

$$\frac{\partial E}{\partial \overline{Wb}_k} = \frac{\partial E}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial \overline{Wb}_k} \tag{4.32}$$

$$\frac{\partial E}{\partial \overline{Wb}_k} = \frac{\partial E}{\partial q_m(k+1)} \times bias \tag{4.33}$$

$$\frac{\partial E}{\partial q_m(k+1)} = \frac{\partial \frac{1}{2} \sum ((x - x_m)^2 + (y - y_m)^2 + (\theta - \theta_m)^2)}{\partial q_m(k+1)} \tag{4.34}$$

$$\Delta \overline{Wb}_k(k+1) = \eta \times bias \times e_k \tag{4.35}$$

$$\overline{Wb}_k(k+1) = \overline{Wb}_k(k) + \Delta \overline{Wb}_k(k+1) \tag{4.36}$$

The connection vector between input bias and hidden layer is $\overline{Vb}_j$

$$\Delta \overline{Vb}_j(k+1) = -\eta \frac{\partial E}{\partial \overline{Vb}_j} \tag{4.37}$$

$$\frac{\partial E}{\partial \overline{Vb}_j} = \frac{\partial E}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial o_k} \frac{\partial o_k}{\partial net_k} \frac{\partial net_k}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \overline{Vb}_j} \tag{4.38}$$

$$\frac{\partial E}{\partial \overline{Vb}_j} = \frac{\partial E}{\partial q_m(k+1)} \sum_{k=1}^{K} W_{kj} f(net_j)' \times bias \tag{4.39}$$

$$\frac{\partial E}{\partial q_m(k+1)} = \frac{\partial \frac{1}{2} \sum ((x - x_m)^2 + (y - y_m)^2 + (\theta - \theta_m)^2)}{\partial q_m(k+1)} \tag{4.40}$$

$$\frac{\partial E}{\partial \overline{Vb}_j} = f(net_j)' \times bias \sum_{k=1}^{K} e_k W_{kj} \tag{4.41}$$

$$\Delta \overline{Vb}_j(k+1) = \eta \times f(net_j)' \times bias \sum_{k=1}^{K} e_k W_{kj} \tag{4.42}$$

$$\overline{Vb}_j(k+1) = \overline{Vb}_j(k) + \Delta \overline{Vb}_j(k+1) \tag{4.43}$$

4.3.4. Dynamics Model Representation

In analogy to nonlinear system identification, a nonlinear dynamic model can be used in two configurations: prediction for one-step and simulation for N-step prediction. Prediction means that the neural networks model and the actual system model receive the same external inputs, but the output of the actual system becomes as input to the neural network model in order to affect the dynamic behaviour of the neural networks model by the actual system model; the model predicts one step into the future. The one-step prediction configuration is called a series-parallel model, and is shown in Figure 4.5 [39, 99, 100, 104, 107, 119, 120, 121].

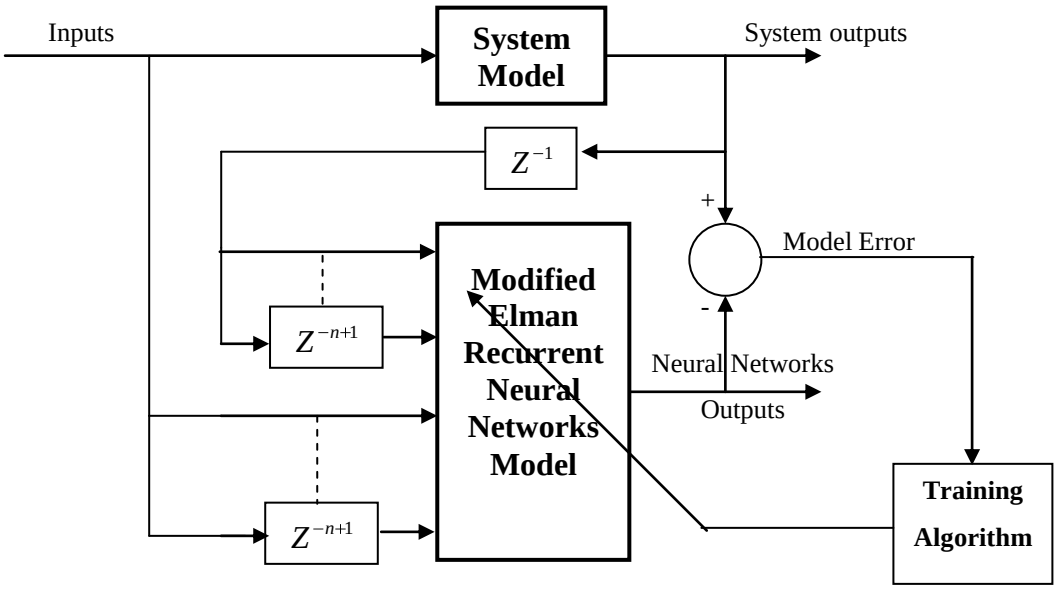


Figure 4.5: The series-parallel structure model [39, 99, 100, 104, 107, 119, 120, 121].

For N-step-ahead prediction, simulation means that the neural networks model and the actual system model receive the same external inputs; the outputs of the actual system model are not used as inputs to the neural networks model. The neural networks model output itself can be fed-back and used as part of the neural networks input. The simulation configuration is called a parallel model, and is shown in Figure 4.6 [39, 99, 100, 119, 120, 121].

However, if the parallel structure model is employed after using series-parallel structure model, it can be guaranteed that the learning neural networks model of the weights will

converge or the error between the output of the system model and that of the neural networks model will lend to zero and $y_m \approx y_p$.

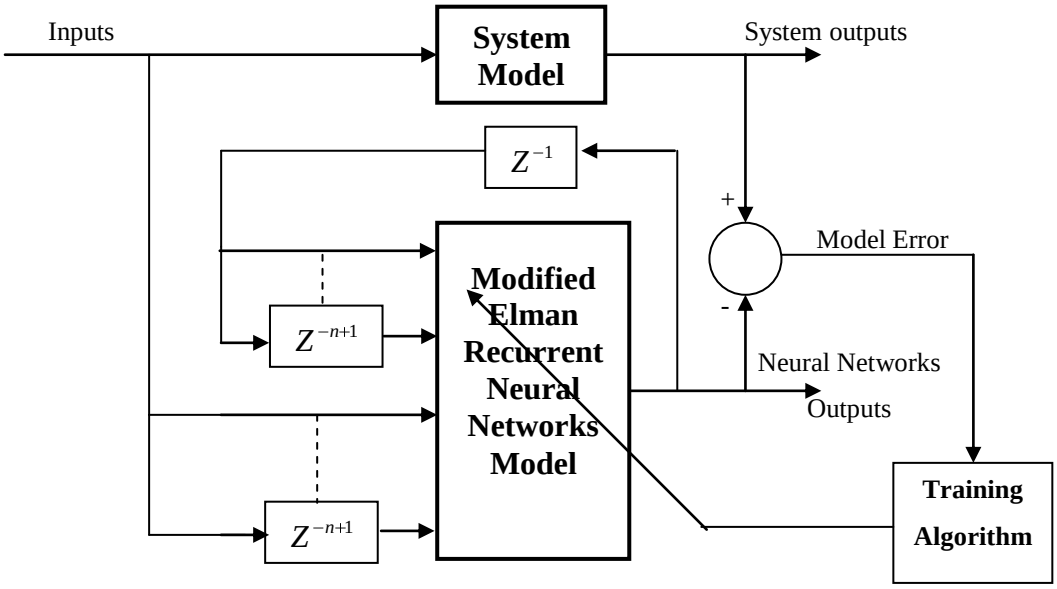


Figure 4.6: The parallel structure model [39, 99, 100, 119, 120, 121].

In this way the neural networks model can be used independently of the actual system model. Back-propagation algorithm is used for learning the two models configuration series-parallel and parallel identification structure.

4.3.5. Model Validation

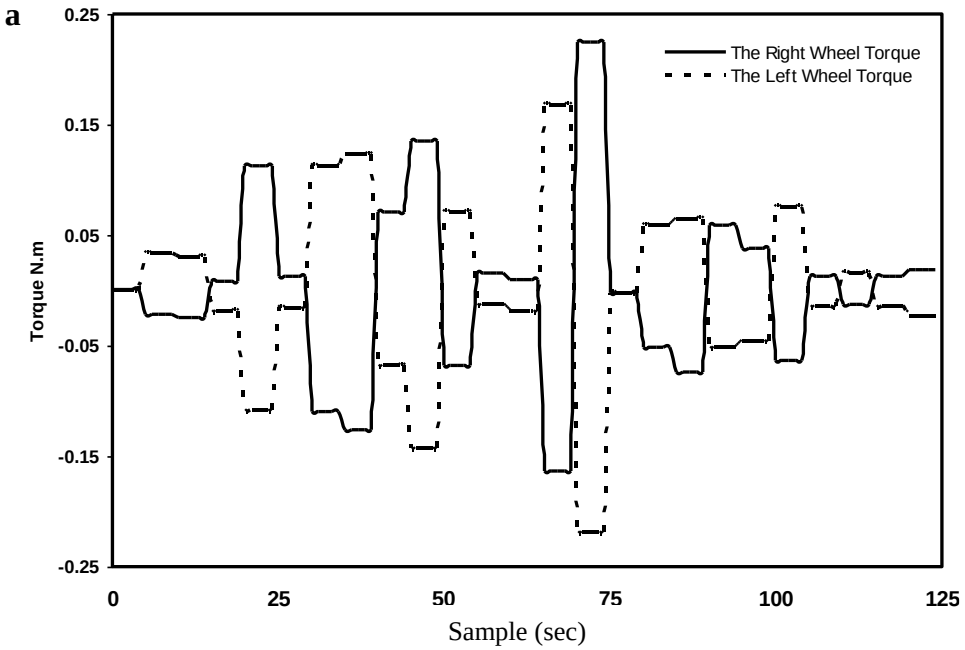
The end task of identification system is validating the neural network model and the model quality. The validation is to check the model quality by using another data set called a testing data set. The test data set should excite the system and the neural model. The validation process is performed using two different approaches. The first is the prediction error between the actual output of the system and neural network model output. The second validation can be achieved through visualization of the prediction. This visualization is given as graphic representation of the actual outputs and the predictions calculated by the neural network model [99, 122].

Alone of the most important problems that are discovered during validating the model is over-learning problem which means the neural network has learned one region and lost another region for learning. To solve this problem, the identification process is repeated

with the removal of one hidden node until no over-learning problem occurs [123]. Another problem is an insufficient training data or poor model generalization behaviour. Rather it can be concluded that model is not flexible enough to describe the underlying relationships [99]. In addition to that, if the number of the learning set is high, the convergence of the error back-propagation learning algorithm will be slow; therefore, it must use the momentum method to accelerate the convergence of the error back-propagation learning algorithm [118].

4.4. Simulation Results for the Mobile Robot Modelling

The first stage in the proposed control methodology is to set the position and orientation neural network identifier (posture identifier) in the neural network topology layer. This task is performed using series-parallel and parallel identification technique configuration with modified Elman recurrent neural networks model. The identification scheme of the nonlinear MIMO mobile robot system is needed to input-output training data pattern to provide enough information about the kinematic and dynamic mobile robot model to be modelled. This can be achieved by injecting a sufficiently rich input signal to excite all process modes of interest, while also ensuring that the training patterns adequately covers the specified operating region. A hybrid excitation signal has been used for the robot model. Figures 4.7a and 4.7b show the input signals $\tau_R(k)$ and $\tau_L(k)$, right and left wheel torques respectively, and τ_I and τ_a, linear and angular torques respectively.



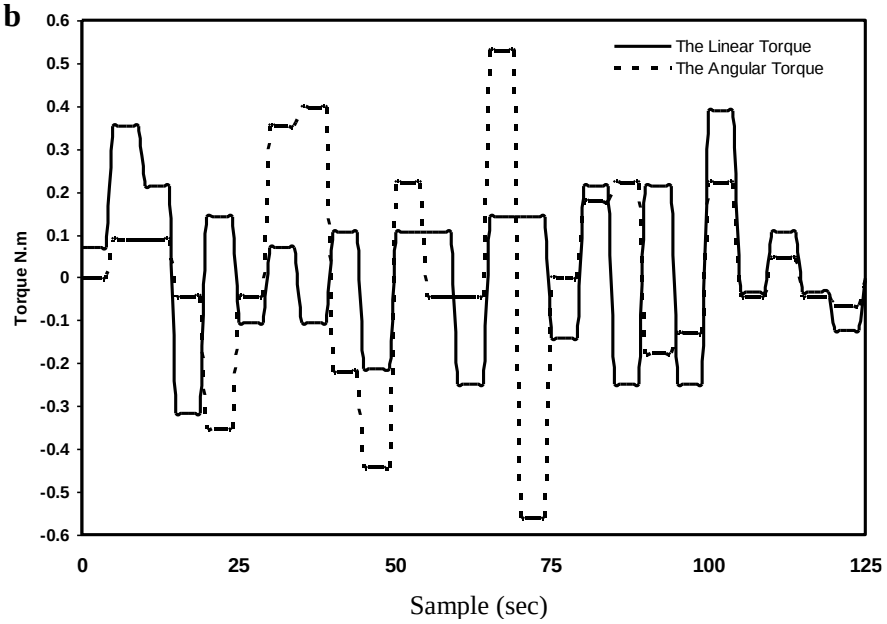
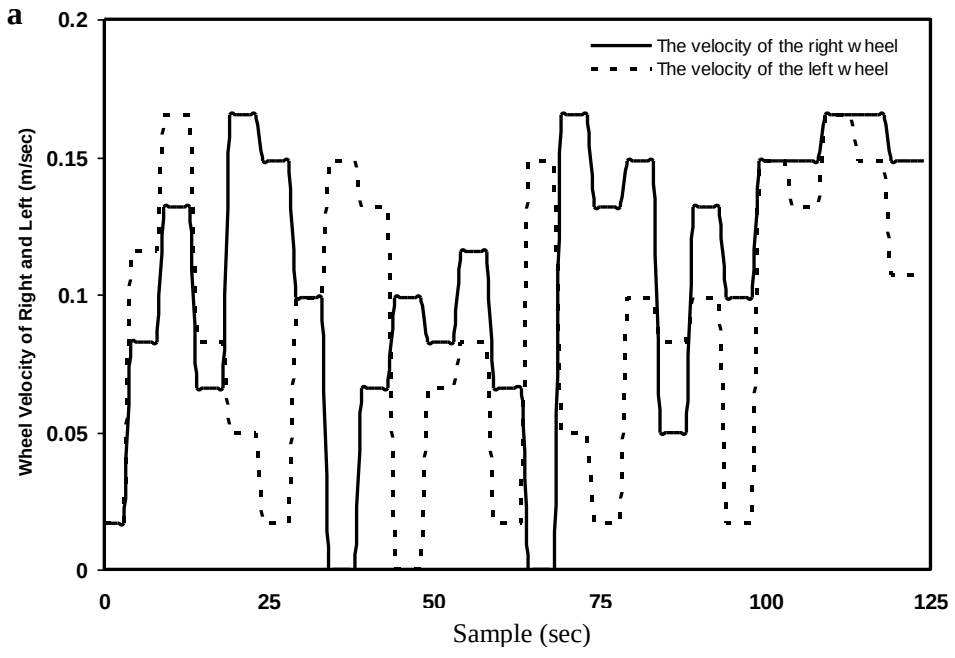


Figure 4.7: The PRBS input torque signals used to excite the mobile robot model: (a) the right and left wheel torque; and (b) linear and angular torque.

Figures 4.8a and 4.8b show the input signals $V_R(k)$ and $V_L(k)$, right and left wheel velocity respectively and V_I and V_W, linear and angular velocity respectively. The training set is generated by feeding a pseudo random binary sequence (PRBS) signals, with sampling time (T_s) of 0.5 second, to the model and measuring its corresponding outputs, position x and y and orientation θ.



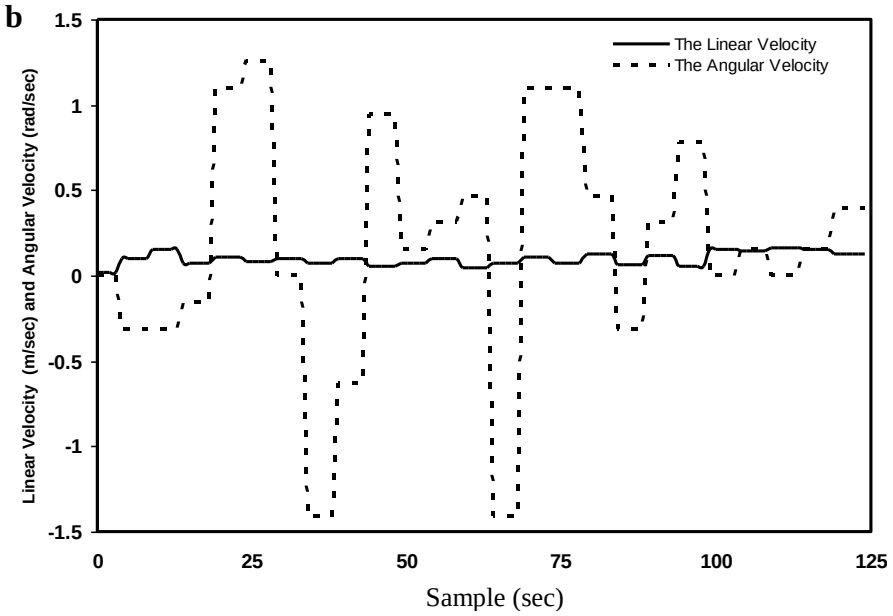
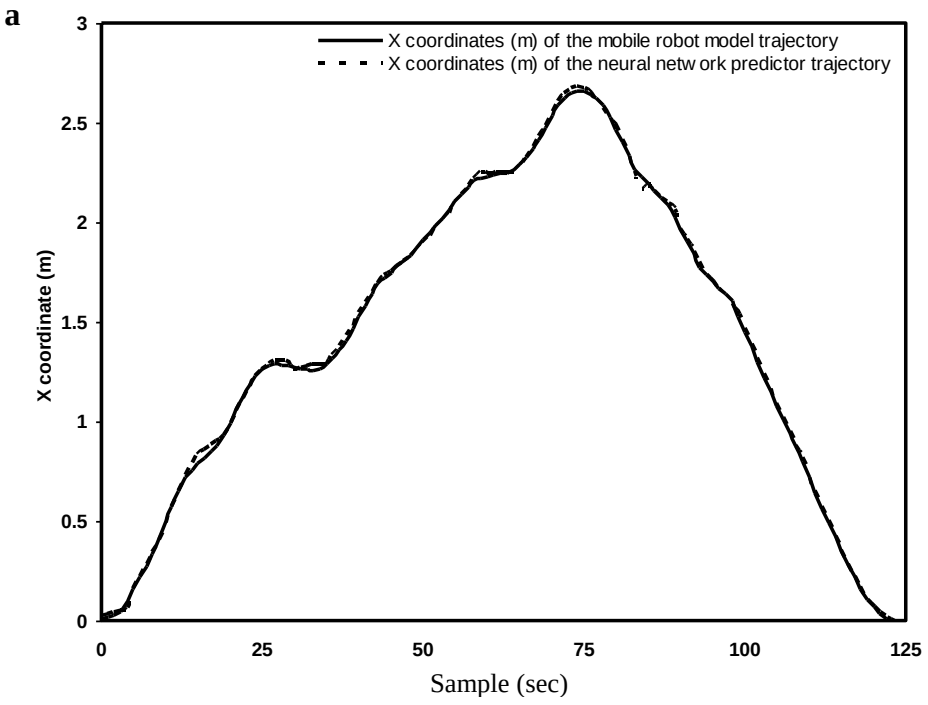


Figure 4.8: The velocity signals to the mobile robot model:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

The minimum number of the nodes in the hidden layer is equal to the number of nodes in the input layer, while the number of the nodes in the context layer is equal to the nodes in the hidden layer; therefore, back propagation learning algorithm is used with the modified Elman recurrent neural network of the structure (5-6-6-3). The number of nodes in the input, hidden, context and output layers are 5, 6, 6 and 3 respectively, as shown in Figure 4.2. The learning algorithm can be developed as in appendix D.



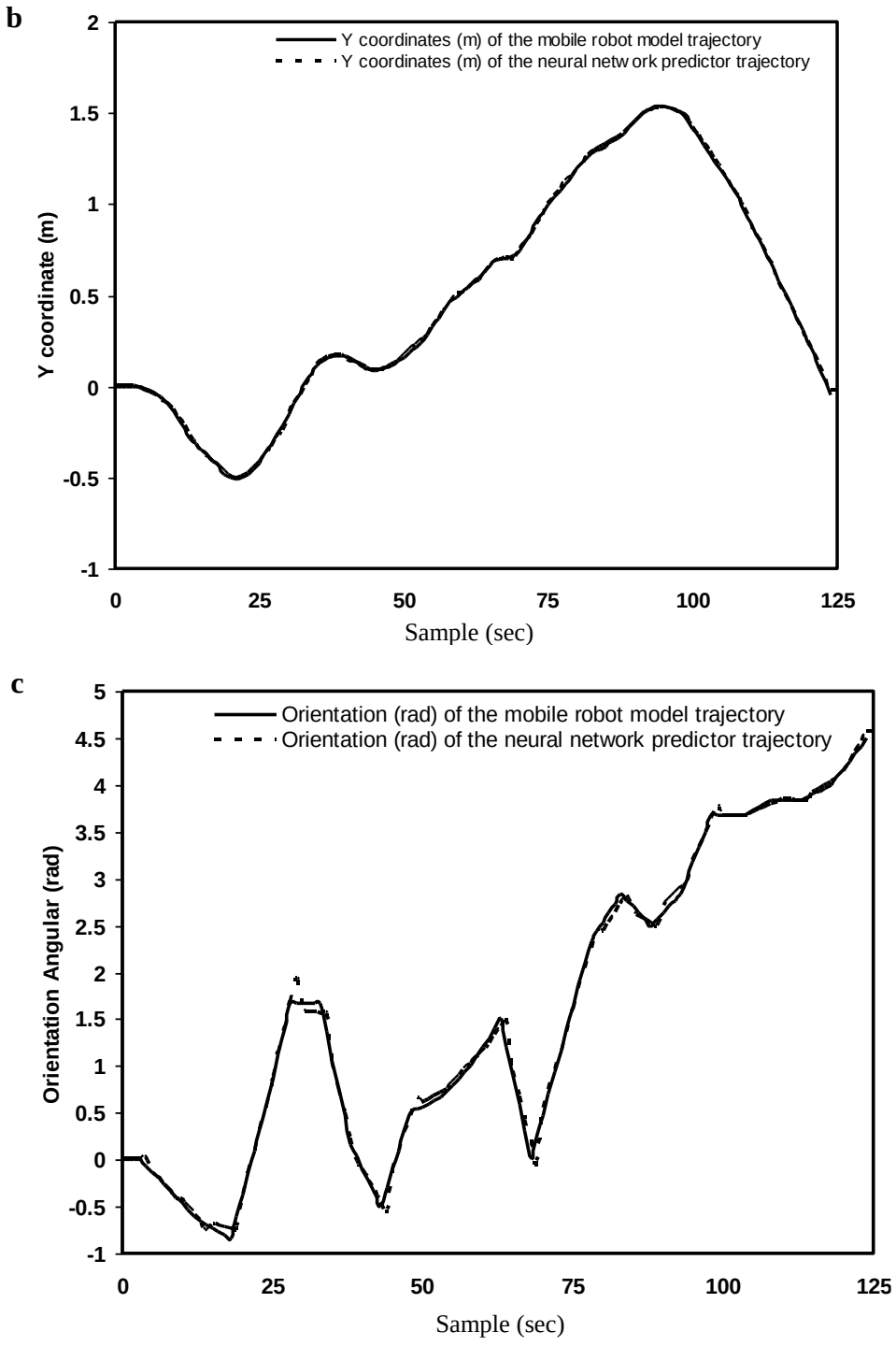


Figure 4.9: The response of the identifier with the actual mobile robot model output: (a) in the X-coordinate; (b) in the Y-coordinate; (c) in the θ-orientation.

A training set of 125 patterns was used with a learning rate of 0.1. After 2230 epochs, the identifier outputs of the neural network, position x, y and orientation θ, are approximated to the actual outputs of the model trajectory, as shown in Figures 4.9a, 4.9b and 4.9c.The objective cost function mean square error (MSE) is less than 0.00045, as shown in Figure 4.10.

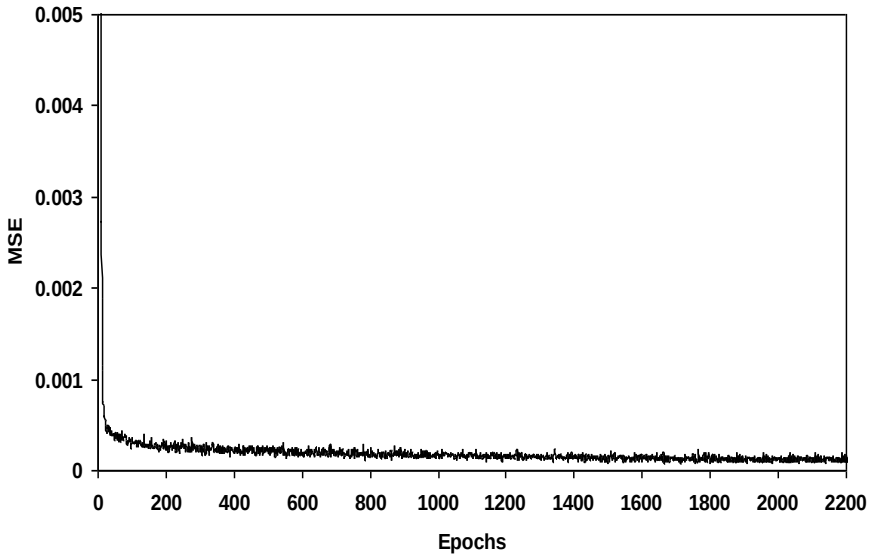


Figure 4.10: The objective cost function MSE.

Parallel configuration is used to guarantee the similarity between the outputs of the neural network identifier and the actual outputs of the mobile robot model trajectory. At 3766 epochs the same training set patterns has been achieved with an MSE less than 6.9×10^{-6}. The neural network identifier position and orientation outputs and the mobile robot model trajectory are shown in Figure 4.11.

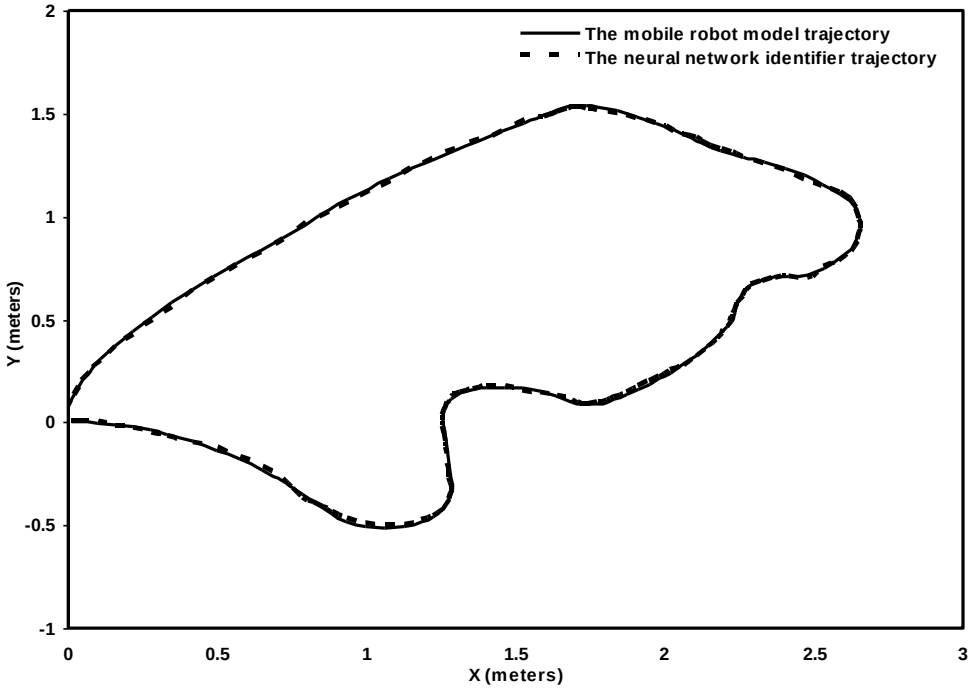


Figure 4.11: The response of the modified Elman neural network model with the actual mobile robot model outputs for the training patterns.

For testing set, Figure 4.12 shows the input signals $\tau_R(k)$ and $\tau_L(k)$, right and left wheel torques respectively. The system has been identified with almost identical position and

orientation of the mobile robot and the neural posture model for testing signal, as shown in Figure 4.13, therefore the neural posture identifier will be used for the on-line estimate for the output of the model position and orientation. To ensure the output of the neural posture identifier model will be equal to the output of the mobile robot, an on-line update of the weights of the neural model will be done by back propagation algorithm. The weights of the posture neural network identifier are shown in appendix E.

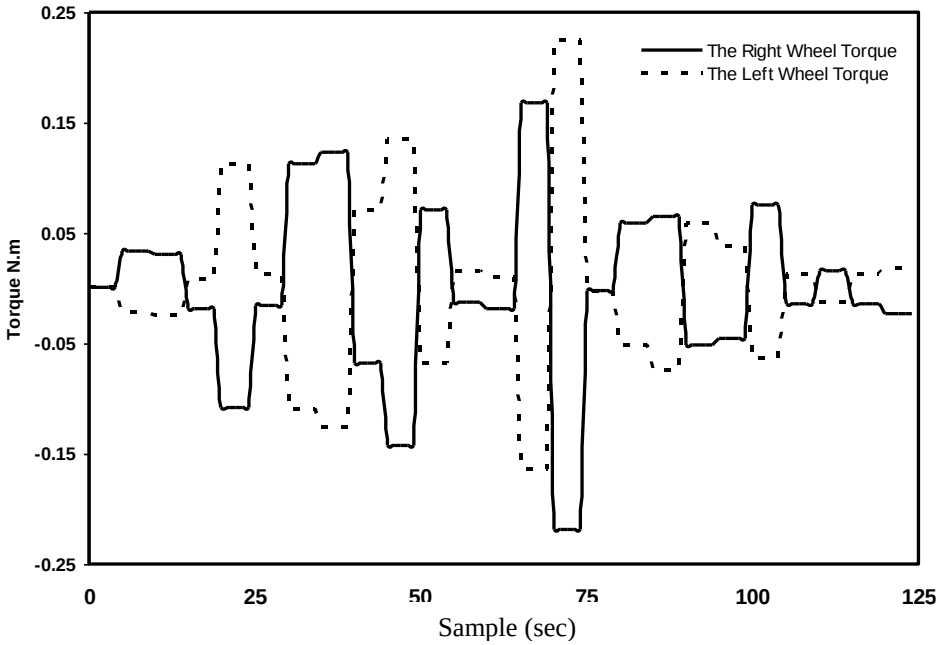


Figure 4.12: The PRBS input torque signals for testing.

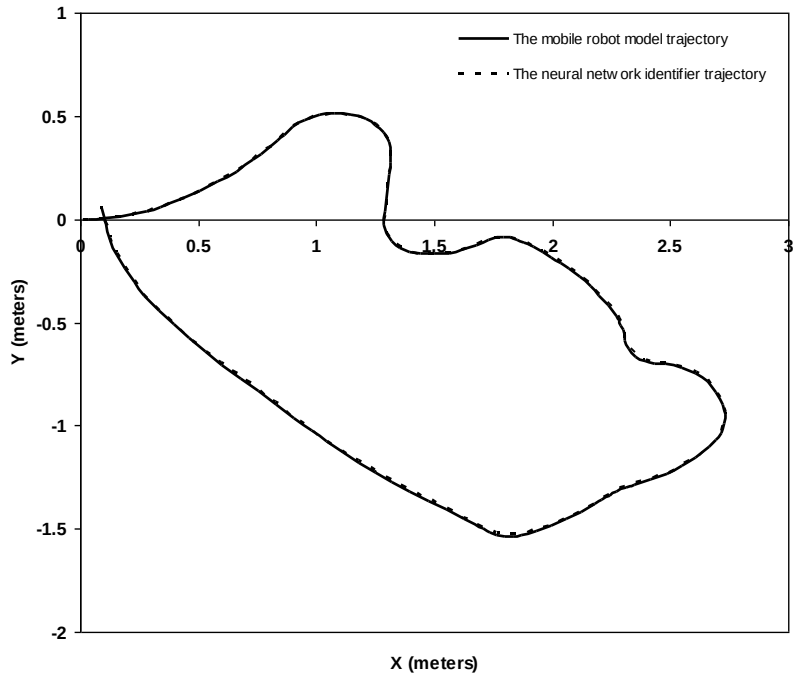


Figure 4.13: The response of the modified Elman neural network model with the actual mobile robot model outputs for the testing patterns.

4.5. Summary

In this chapter, the use of modified Elman recurrent neural networks with relation to input-output model of nonholonomic wheeled mobile robot kinematics and dynamics is outlined in order to predict the posture (position and orientation) of the mobile robot for N step-ahead prediction. The focus has been on the network architecture used to present the modified Elman recurrent neural networks model and the learning mechanism of that network by using back propagation algorithm with two configurations: series-parallel and parallel model.

Chapter Five

Adaptive Neural Predictive Controller

5.1. Introduction

An adaptive neural predictive control of nonlinear multi-input multi-output MIMO mobile robot system is considered in this chapter. The approach used to control the trajectory tracking of nonholonomic wheeled mobile robot depends on the information available about the mobile robot, using three control methodologies. These control methodologies are based on the minimisation of a quadratic performance index function of the error between the desired trajectory input and the actual outputs identifier model (position and orientation of mobile robot model). The performance of the proposed adaptive neural control with three control methodologies will be compared with them in terms of minimum tracking error and in generating an optimal torque control action and the capability of tracking any trajectories with continuous and non-continuous gradients, despite the presence of bounded external disturbances.

5.2. Model Predictive Control

The basic structure of the model predictive control is shown in Figure 5.1 [124].

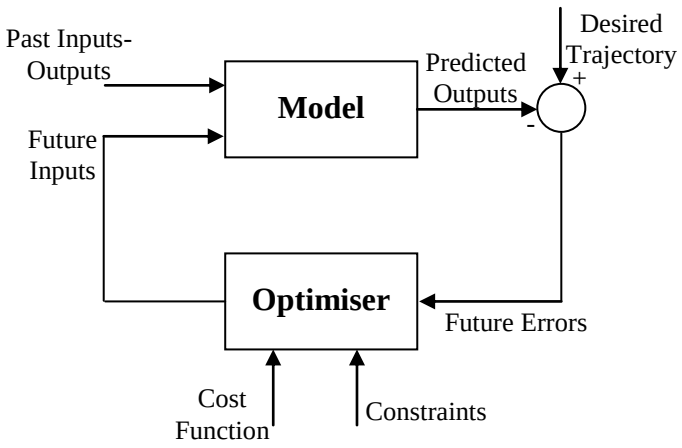


Figure 5.1: Basic structure of model predictive controller [124].

The strategy of this methodology is [124]:

a- Explicit use of a model to predict the system output at future time instants.

b- Calculation of a control sequence minimising an objective function.

c- Receding strategy, so that at each instant the system output at future time instant is displaced towards the future, which involves the application of the first control signal of the sequence calculated at each step.

A model is used to predict the future system outputs, based on past and current values and on the proposed optimal future control actions. The model is considered as modified Elman recurrent neural networks model and acts as posture identifier to predict the position and orientation of the mobile robot. The actions are calculated by the optimiser, taking into account the cost function (where the future tracking error is considered) as well as the constraints (the amount of computation required is ever higher). To apply the idea of predictive optimisation algorithm that minimises the difference between the predicted error and the desired robot trajectory in certain interval. The first step in the procedure of the control structure is the identification of the kinematics and dynamics mobile robot model from the input-output data. In the second step, a feedforward neural controller is designed using feedforward multi-layer perceptron neural networks to find reference torques that control the steady-state outputs of the mobile robot trajectory. The final step uses a robust feedback predictive controller.

5.3. Adaptive Neural Predictive Controller Structure

The proposed structure of the mobile robot actuators controller is an adaptive neural predictive controller and can be given in the form of the block diagram (shown in Figure 5.2).

It consists of:

1- Position and orientation neural network identifier (see chapter four).

2- Feedforward neural controller.

3- Feedback neural controller.

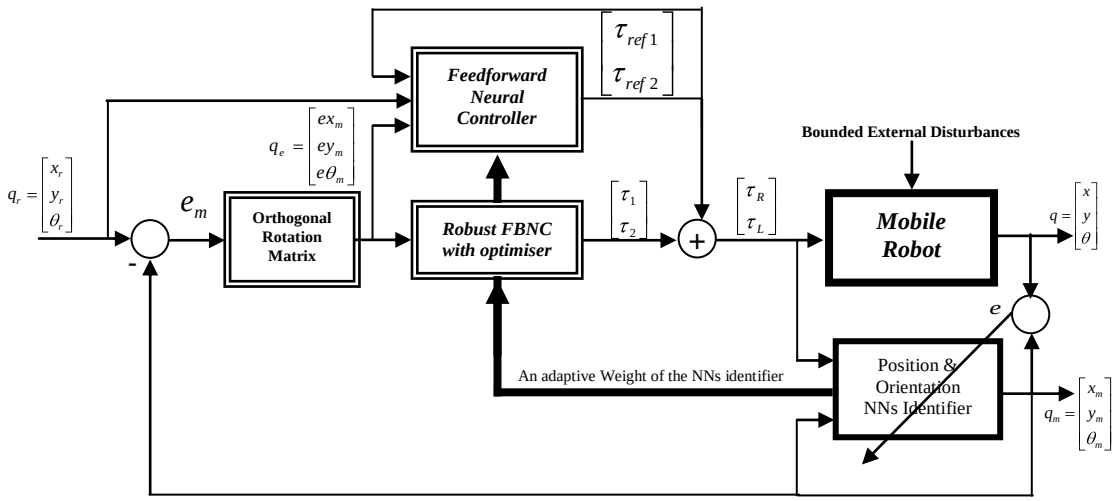


Figure 5.2: The proposed structure of the adaptive neural predictive controller for the mobile robot actuators system.

Three kinds of the control methodologies are proposed and applied.

First control methodology - the feedback neural controller that consists of position and orientation neural network identifier with predictive optimisation algorithm.

Second control methodology - the nonlinear PID neural feedback controller that consists of self-tuning nonlinear PID neural parameters with posture identifier and predictive optimisation algorithm.

Third control methodology - nonlinear inverse-dynamic neural predictive feedback controller, which consists of the nonlinear feedback acceleration control equation based on Lyapunov criterion stability method and posture neural network identifier with optimisation predictive algorithm. These control methodologies are based on the minimisation of a quadratic performance index function of the error between the desired trajectory input and the actual outputs identifier model (position and orientation of mobile robot model).

The integrated adaptive control structure, which consists of an adaptive feedforward neural controller and feedback neural controller with an optimisation predictive algorithm, brings together the advantages of the adaptive neural method with the robustness of feedback for N-step-ahead prediction. In the following sections, each part of the proposed actuators controller is explained in detail.

5.3.1. Feedforward Neural Controller

The feedforward neural controller (FFNC) is of prime importance in the structure of the controller due to its necessity in keeping the steady-state tracking error at zero. This means that the actions of the FFNC, $\tau_{ref1}(k)$ and $\tau_{ref2}(k)$ are used as the reference torques of the steady state outputs of the mobile robot. Hence, the FFNC is supposed to learn the adaptive inverse model of the mobile robot system with off-line and on-line stages to calculate mobile robot's reference input torques drive. Reference input torques keep the robot on a desired trajectory in the presence of any disturbances or initial state errors.

To achieve FFNC, a multi-layer perceptron model is used, as shown in Figure 5.3 [118]. The system is composed of many interconnected processing units called neurons or nodes.

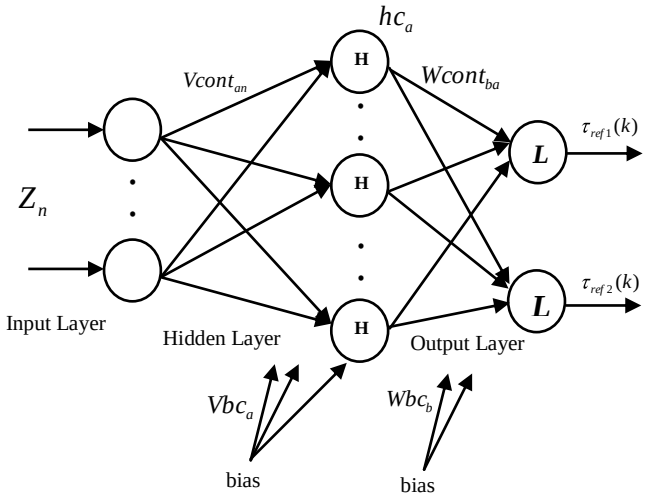


Figure 5.3: The multi-layer perceptron neural networks act as the feedforward neural controller.

The network notations are as follows:

$Vcont$: Weight matrix of the hidden layers.

$\overline{Vbc}$: Weight vector of the hidden layers.

$Wcont$: Weight matrix of the output layer.

$\overline{Wbc}$: Weight vector of the output layer.

To explain these calculations, consider the general a^{th} neuron in the hidden layer shown in Figure 5.3. The inputs to this neuron consist of an n-dimensional vector (n^{th} is the number of the input nodes). Each of the inputs has a weight $Vcont$ associated with it. The first

calculation within the neuron consists of calculating the weighted sum $netc_a$ of the inputs as [118]:

$$netc_a = \sum_{a=1}^{nhc} Vcont_{an} \times Z_n + bias \times \overline{Vbc_a} \tag{5.1}$$

where *nhc* is number of the hidden nodes.

Next the output of the neuron hc_a is calculated as the continuous sigmoid function of the $netc_a$ as:

$$hc_a = \mathrm{H}(netc_a) \tag{5.2}$$

$$\mathrm{H}(netc_a) = \frac{2}{1+e^{-netc_a}} - 1 \tag{5.3}$$

Once the outputs of the hidden layer are calculated, they are passed to the output layer. In the output layer, two linear neurons are used to calculate the weighted sum (*netco*) of its inputs (the output of the hidden layer as in equation 5.4).

$$netco_b = \sum_{a=1}^{nhc} Wcont_{ba} \times hc_a + bias \times \overline{Wbc_b} \tag{5.4}$$

where $Wcont_{ba}$ is the weight between the hidden neuron hc_a and the output neuron.

The two linear neurons then pass the sum $(netco_b)$ through a linear function of slope 1 (another slope can be used to scale the output) as:

$$Oc_b = L(netco_b) \tag{5.5}$$

The outputs of the feedforward neural network controller represent right and left wheels torques, $\tau_{ref1}(k)$ and $\tau_{ref2}(k)$ respectively.

The training of the feedforward neural controller is performed off-line as shown in Figure 5.4, in which the weights are adapted on-line in order to keep the robot on a desired trajectory in the presence of any disturbances or initial state errors. The posture neural network identifier finds the mobile robot Jacobian through the neural identifier model. The indirect learning approach is currently considered as one of the better approaches that

can be followed to overcome the lack of initial knowledge, such as initial pose and disturbances effect [116].

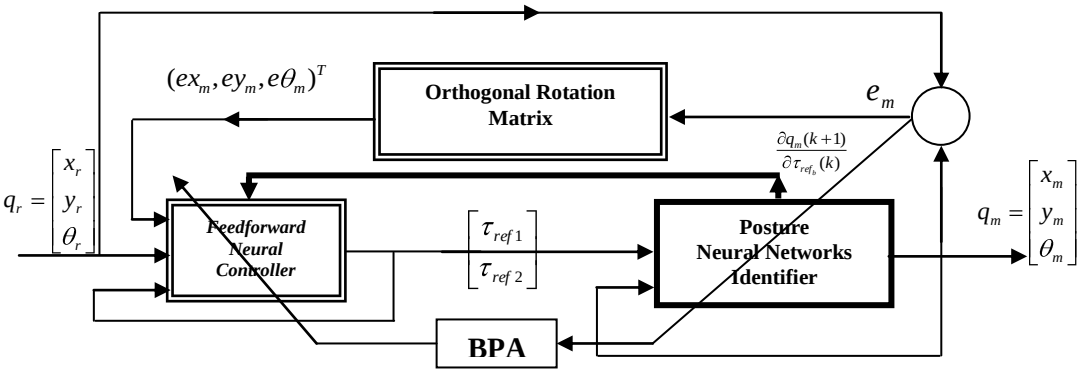


Figure 5.4: The feedforward neural controller structure for mobile robot model.

The dynamic back propagation algorithm is employed to realise the training of the weights of feedforward neural controller. The sum of the square of the differences between the desired posture $q_r = (x_r, y_r, \theta_r)^T$ and network posture $q_m = (x_m, y_m, \theta_m)^T$ is

$$Ec = \frac{1}{npc}\sum_{i=1}^{npc}((x_r - x_m)^2 + (y_r - y_m)^2 + (\theta_r - \theta_m)^2) \tag{5.6}$$

where *npc* is number of patterns.

The use of the minus sign accounts for the gradient-descent in weight-space and reflects the search for the direction of the weight change that reduces the value of Ec. The learning rate η has a small value for smoothness weights changing. Using the chain rule differentiation:

The connection matrix between hidden layer and output layer is $Wcont_{ba}$

$$\Delta Wcont_{ba}(k+1) = -\eta \frac{\partial Ec}{\partial Wcont_{ba}} \tag{5.7}$$

$$\frac{\partial Ec}{\partial Wcont_{ba}} = \frac{\partial Ec}{\partial netc_b} \times \frac{\partial netc_b}{\partial Wcont_{ba}} \tag{5.8}$$

$$\frac{\partial Ec}{\partial Wcont_{ba}} = \frac{\partial Ec}{\partial oc_b} \times \frac{\partial oc_b}{\partial netc_b} \times \frac{\partial netc_b}{\partial Wcont_{ba}} \tag{5.9}$$

$$\frac{\partial Ec}{\partial Wcont_{ba}} = \frac{\partial Ec}{\partial \tau_{ref\,b}(k)} \times \frac{\partial \tau_{ref\,b}(k)}{\partial oc_b} \times \frac{\partial oc_b}{\partial netc_b} \times \frac{\partial netc_b}{\partial Wcont_{ba}} \tag{5.10}$$

$$\frac{\partial Ec}{\partial Wcont_{ba}} = \frac{\partial Ec}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} \frac{\partial \tau_{ref\,b}(k)}{\partial oc_b} \frac{\partial oc_b}{\partial netc_b} \frac{\partial netc_b}{\partial Wcont_{ba}} \tag{5.11}$$

$$\frac{\partial Ec}{\partial Wcont_{ba}} = \frac{\partial Ec}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} \frac{\partial \tau_{ref\,b}(k)}{\partial oc_b} f'(netc_b) hc_a \tag{5.12}$$

$$\frac{\partial Ec}{\partial q_m(k+1)} = \frac{\partial \frac{1}{2}\sum((x_r - x_m)^2 + (y_r - y_m)^2 + (\theta_r - \theta_m)^2)}{\partial q_m(k+1)} \tag{5.13}$$

This is achieved in the local coordinates with respect to the body of the mobile robot, which is the same output of the position and orientation neural networks identifier. The configuration error can be represented by using a transformation matrix as:

$$\begin{bmatrix} ex_m \\ ey_m \\ e\theta_m \end{bmatrix} = \begin{bmatrix} \cos\theta_m & \sin\theta_m & 0 \\ -\sin\theta_m & \cos\theta_m & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x_m \\ y_r - y_m \\ \theta_r - \theta_m \end{bmatrix} \tag{5.14}$$

where x_r, y_r and θ_r are the reference position and orientation of the mobile robot.

$$Jacobain = \frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} \tag{5.15}$$

where the outputs of the identifier are $q_m = (x_m, y_m, \theta_m)^T$

$$\frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} = \frac{\partial q_m(k+1)}{\partial o_k(k)} \frac{\partial o_k(k)}{\partial net_k} \frac{\partial net_k}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_{ref\,b}(k)} \tag{5.16}$$

For linear activation function in the outputs layers:

$$\frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} = \frac{\partial net_k}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_{ref\,b}(k)} \tag{5.17}$$

For nonlinear activation function in the hidden layers:

$$\frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} = \sum_{k=1}^{K} W_{kj} f(net_j)' \frac{\partial net_j}{\partial \tau_{ref\,b}(k)} \tag{5.18}$$

$$\frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} = \sum_{j=1}^{nh} f(net_j)' VH_{jb} \sum_{k=1}^{K} W_{kj} \tag{5.19}$$

Substituting equations 5.19 and 5.13 into equation 5.12, $\Delta Wcont_{ba}(k+1)$ becomes:

$$\Delta Wcont_{ba}(k+1) = \eta hc_a \times \sum_{j=1}^{nh} f(net_j)' VH_{jb}((ex_m(k+1)W_{1j}) + (ey_m(k+1)W_{2j}) + (e\theta_m(k+1)W_{3j})) \tag{5.20}$$

$$Wcont_{ba}(k+1) = Wcont_{ba}(k) + \Delta Wcont_{ba}(k+1) \tag{5.21}$$

The connection matrix between input layer and hidden layer is $Vcont_{an}$

$$\Delta Vcont_{an}(k+1) = -\eta \frac{\partial Ec}{\partial Vcont_{an}} \tag{5.22}$$

$$\frac{\partial Ec}{\partial Vcont_{an}} = \frac{\partial Ec}{\partial netc_b} \times \frac{\partial netc_b}{\partial Wcont_{ba}} \tag{5.23}$$

$$\frac{\partial Ec}{\partial Vcont_{an}} = \frac{\partial Ec}{\partial oc_b} \times \frac{\partial oc_b}{\partial netc_b} \times \frac{\partial netc_b}{\partial Vcont_{an}} \tag{5.24}$$

$$\frac{\partial Ec}{\partial Vcont_{an}} = \frac{\partial Ec}{\partial \tau_{ref\,b}(k)} \times \frac{\partial \tau_{ref\,b}(k)}{\partial oc_b} \times \frac{\partial oc_b}{\partial netc_b} \times \frac{\partial netc_b}{\partial hc_a} \times \frac{\partial hc_a}{\partial netc_a} \times \frac{\partial netc_a}{\partial Vcont_{an}} \tag{5.25}$$

$$\frac{\partial Ec}{\partial Vcont_{an}} = \frac{\partial Ec}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} \frac{\partial \tau_{ref\,b}(k)}{\partial oc_b} \times \frac{\partial oc_b}{\partial netc_b} \times \frac{\partial netc_b}{\partial hc_a} \times \frac{\partial hc_a}{\partial netc_a} \times \frac{\partial netc_a}{\partial Vcont_{an}} \tag{5.26}$$

$$\frac{\partial Ec}{\partial Vcont_{an}} = \frac{\partial Ec}{\partial q_m(k+1)} \frac{\partial q_m(k+1)}{\partial \tau_{ref\,b}(k)} \times \sum_{b=1}^{B} Wcont_{ba} \times f(netc_a)' \times Z_n \tag{5.27}$$

Substituting equations 5.19 and 5.13 into equation 5.27, $\Delta Vcont_{an}(k+1)$ becomes:

$$\Delta Vcont_{an}(k+1) = \eta Z_n f(netc_a)' \sum_{b=1}^{B} Wcont_{ba} \sum_{j=1}^{nh} f(net_j)' \sum_{i=1}^{I} VH_{ji} \times$$

$$((ex_m(k+1)W_{1j}) + (ey_m(k+1)W_{2j}) + (e\theta_m(k+1)W_{3j})) \tag{5.28}$$

The *B* and *I* are equal to two because there are two outputs in the feedforward neural controller.

$$Vcont_{an}(k+1) = Vcont_{an}(k) + \Delta Vcont_{an}(k+1) \tag{5.29}$$

Once the feedforward neural controller has learned, it generates the torque control action to keep the output of the mobile robot at the steady state reference value and to overcome any external disturbances during trajectory. The torques will be known as the reference torques of the right and left wheels $\tau_{ref\,1}$ and $\tau_{ref\,2}$ respectively.

5.3.2. Feedback Neural Controller

The feedback neural controller is essential to stabilise the tracking error of the mobile robot system when the trajectory of the robot drifts from the desired trajectory during transient state. The feedback neural controller generates an optimal torque control action

that minimises the cumulative error between the desired input trajectory and the output trajectory of the mobile robot. The weighted sum of the torque control signal can be obtained by minimising a quadratic performance index. The feedback neural controller consists of the adaptive weights of the position and orientation neural networks identifier and an optimisation algorithm. The quadratic performance index for multi-input/multi-output system can be expressed as:

$$J = \frac{1}{2}\sum_{k=1}^{N} Q(q_r(k+1) - q(k+1))^2 + R((\tau_{ref1}(k) - \tau_R(k))^2 + (\tau_{ref2}(k) - \tau_L(k))^2) \quad (5.30)$$

Hence

$$q_r(k+1) = [x_r(k+1), y_r(k+1), \theta_r(k+1)]^T \quad (5.31)$$

$$q(k+1) = [x(k+1), y(k+1), \theta(k+1)]^T \quad (5.32)$$

$$\tau_R(k) = \tau_{ref1}(k) + \tau_1(k) \quad (5.33)$$

$$\tau_L(k) = \tau_{ref2}(k) + \tau_2(k) \quad (5.34)$$

(Q, R) are positive weighting factors.

N is the number of steps ahead.

Then J will be given as follows:

$$J = \frac{1}{2}\sum_{k=1}^{N} Q((x_r(k+1) - x(k+1))^2 + (y_r(k+1) - y(k+1))^2 + (\theta_r(k+1) - \theta(k+1))^2) + R((\tau_1(k))^2 + (\tau_2(k))^2) \quad (5.35)$$

The quadratic cost function will not only force the mobile robot output to follow the desired trajectory by minimising the cumulative error for N steps-ahead, but also forces the torque control actions ($\tau_1(k)$ and $\tau_2(k)$) in the transient period to be as close as possible to the reference torque control signals ($\tau_{ref1}(k)$ and $\tau_{ref2}(k)$).

In addition, J depends on Q and R factors and chooses a set of values of the weighting factors Q and R to determine the optimal control action by observing the system behaviour [97]. The on-line position and orientation neural networks identifier is used to obtain the predicted values of the outputs of the mobile robot system $q_m(k+1)$ for N steps ahead, instead of running the mobile robot system itself $q(k+1)$ for N steps. This is performed to find the optimal torque control actions by using the posture identifier

weights and optimisation algorithm depending on the quadratic cost function. Therefore, it can be said that:

$$q_m(k+1) \approx q(k+1) \tag{5.36}$$

and the performance index of equation 5.35 can be stated as:

$$J = \frac{1}{2}\sum_{k=1}^{N} Q((x_r(k+1) - x_m(k+1))^2 + (y_r(k+1) - y_m(k+1))^2 + (\theta_r(k+1) - \theta_m(k+1))^2) + R((\tau_1(k))^2 + (\tau_2(k))^2) \tag{5.37}$$

To achieve equations 5.36 and 5.37, a modified Elman neural network will be used as posture identifier. This task is carried out using an identification technique based on series-parallel and parallel configuration with two stages to learn the posture identifier.

The first stage is an off-line identification, while the second stage is an on-line modification of the weights of the obtained position and orientation neural identifier. The on-line modifications are necessary to keep tracking any possible variation in the kinematics and dynamics parameters of the mobile robot system. Back propagation algorithm (BPA) is used to adjust the weights of the posture neural identifier to learn the kinematics and dynamics of the mobile robot system, by applying a simple gradient decent rule.

5.3.2.1. Neural Feedback Control Methodology

The first control methodology of the feedback controller is neural predictive feedback controller which consists of position and orientation neural network identifier with predictive optimisation algorithm.

In this section, the two feedback right and left wheels torque control signals, $\tau_1(k)$ and $\tau_2(k)$ respectively, will be derived for one-step-ahead when N=1.

where

$$\tau_1(k+1) = \tau_1(k) + \Delta\tau_1(k+1) \tag{5.38}$$

$$\tau_2(k+1) = \tau_2(k) + \Delta\tau_2(k+1) \tag{5.39}$$

The control law is obtained by minimising the quadratic cost function as follows:

$$\Delta\tau_1(k+1) = -\eta \frac{\partial J}{\partial \tau_1(k)} \tag{5.40}$$

The use of the minus sign accounts for the gradient-descent in weight-space and reflects the search for the direction of the weight change, which reduces the value of *J*. Using the chain rule differentiation, it has:

$$-\eta \frac{\partial J}{\partial \tau_1(k)} = -\eta \frac{\frac{1}{2} Q \partial((x_r(k+1) - x_m(k+1))^2}{\partial \tau_1(k)} - \eta \frac{\frac{1}{2} Q \partial((y_r(k+1) - y_m(k+1))^2}{\partial \tau_1(k)}$$

$$-\eta \frac{\frac{1}{2} Q \partial((\theta_r(k+1) - \theta_m(k+1))^2}{\partial \tau_1(k)} - \eta \frac{\frac{1}{2} R \partial(\tau_1(k))^2}{\partial \tau_1(k)} \tag{5.41}$$

$$\Delta\tau_1(k) = \eta \frac{Q e x_m(k+1) \partial x_m(k+1)}{\partial \tau_1(k)} + \eta \frac{Q e y_m(k+1) \partial y_m(k+1)}{\partial \tau_1(k)}$$

$$+\eta \frac{Q e \theta_m(k+1) \partial \theta_m(k+1)}{\partial \tau_1(k)} - \eta R \tau_1(k) \tag{5.42}$$

where $ex_m(k+1), ey_m(k+1), e\theta_m(k+1)$ can be calculated from equation 5.14.

The modified Elman neural network identifier shown in Figure 5.5 has:

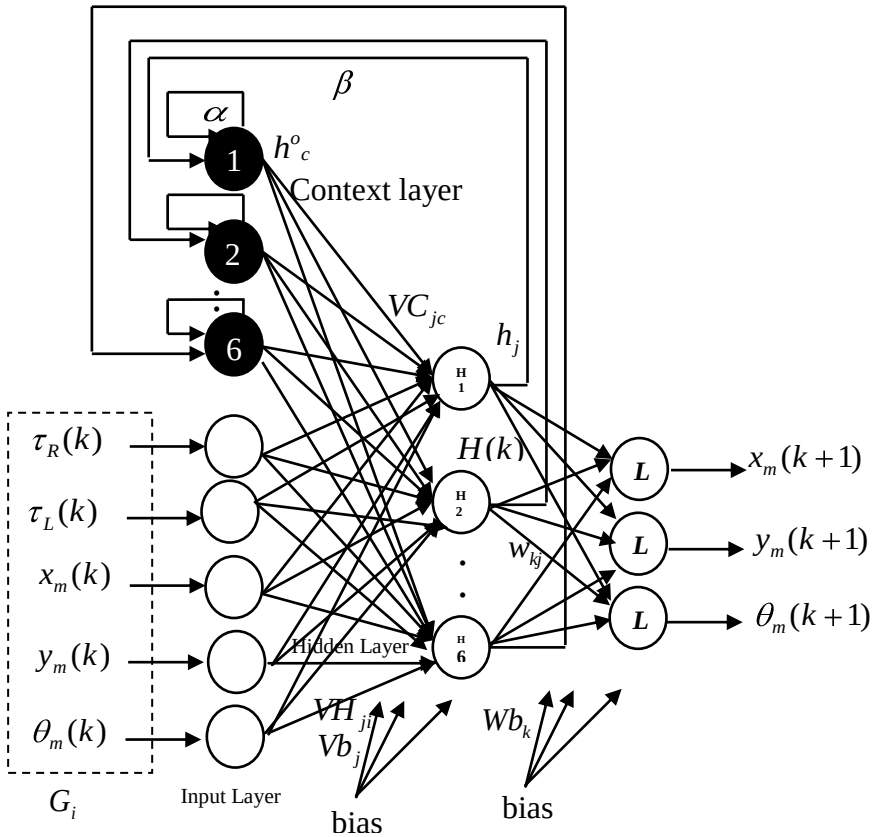


Figure 5.5: Elman neural networks act as the posture identifier.

$$\frac{\partial q_m(k+1)}{\partial \tau_1(k)} = \frac{\partial q_m(k+1)}{\partial o_k}\frac{\partial o_k}{\partial net_k}\frac{\partial net_k}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_R(k)}\frac{\partial \tau_R(k)}{\partial \tau_1(k)} \tag{5.43}$$

For the output with a linear activation function

$$\frac{\partial q_m(k+1)}{\partial \tau_1(k)} = \frac{\partial net_k}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_R(k)}\frac{\partial \tau_R(k)}{\partial \tau_1(k)} \tag{5.44}$$

$$\frac{\partial q_m(k+1)}{\partial \tau_1(k)} = \frac{\partial net_k}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_R(k)} \tag{5.45}$$

Figure 5.5 shows that $\tau_R(k)$ is linked to the exciting nodes, VH_{j1} and $\tau_L(k)$ is linked to the exciting nodes VH_{j2} then:

$$\frac{\partial x_m(k+1)}{\partial \tau_1(k)} = \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} \tag{5.46}$$

$$\frac{\partial y_m(k+1)}{\partial \tau_1(k)} = \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j1} \tag{5.47}$$

$$\frac{\partial \theta_m(k+1)}{\partial \tau_1(k)} = \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j1} \tag{5.48}$$

$$\Delta\tau_1(k+1) = \eta Q(ex_m(k+1)\sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} + ey_m(k+1)\sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j1}$$
$$+ e\theta_m(k+1)\sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j1}) - \eta R\tau_1(k) \tag{5.49}$$

and for $\Delta\tau_2(k+1)$ is:

$$\Delta\tau_2(k+1) = \eta Q(ex_m(k+1)\sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j2} + ey_m(k+1)\sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j2}$$
$$+ e\theta_m(k+1)\sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2}) - \eta R\tau_2(k) \tag{5.50}$$

The total control action of the nonlinear neural controller became as:

$$\tau_R(k+1) = \tau_{ref1}(k+1) + \tau_1(k+1) \tag{5.51}$$

$$\tau_L(k+1) = \tau_{ref2}(k+1) + \tau_2(k+1) \tag{5.52}$$

This is calculated at each sample time k and applied to mobile robot system and the position and orientation identifier model. Then we continue to apply this procedure at the

next sampling time (k+1) until the error between the desired trajectory and the identifier model output becomes lower than a specified value.

The weights of the position and orientation neural network identifier and the weights of the feedforward neural controller are updated after each sampling time in order to minimise the error between $q=[x,y,\theta]^T$ & $q_m=[x_m,y_m,\theta_m]^T$ using back propagation learning algorithm. For N steps estimation of the two feedback neural controller actions $\tau_1(k)\,\&\,\tau_2(k)$ the techniques of generalized predictive control theory will be used. The N steps estimation of $\tau_1(k)\,\&\,\tau_2(k)$ will be calculated for each sample. The position and orientation in the identifier model, shown in Figure 5.5, represent the kinematics and dynamics model of the mobile robot system and will be controlled asymptotically. Therefore, they can be used to predict future values of the model outputs for the next N steps and can be used to find the optimal value of $\tau_1(k)\,\&\,\tau_2(k)$ using an optimisation algorithm. For this purpose, let N be a pre-specified positive integer that is denoted such that the future values of the set point are:

$$X_{rt,N}=[x_r(t+1),x_r(t+2),x_r(t+3),...,x_r(t+N)] \tag{5.53}$$

$$Y_{rt,N}=[y_r(t+1),y_r(t+2),y_r(t+3),...,y_r(t+N)] \tag{5.54}$$

$$\theta_{rt,N}=[\theta_r(t+1),\theta_r(t+2),\theta_r(t+3),...,\theta_r(t+N)] \tag{5.55}$$

where t represents the time instant.

Then the predicted outputs of the robot model used the neural identifier, shown in Figure 5.5, are:

$$X_{mt,N}=[x_m(t+1),x_m(t+2),x_m(t+3),...,x_m(t+N)] \tag{5.56}$$

$$Y_{mt,N}=[y_m(t+1),y_m(t+2),y_m(t+3),...,y_m(t+N)] \tag{5.57}$$

$$\theta_{mt,N}=[\theta_m(t+1),\theta_m(t+2),\theta_m(t+3),...,\theta_m(t+N)] \tag{5.58}$$

Equations 5.59, 5.60 and 5.61 implement equation 5.14 to calculate the error vectors.

$$EX_{m,t,N}=[ex_m(t+1),ex_m(t+2),ex_m(t+3)...,ex_m(t+N] \tag{5.59}$$

$$EY_{m,t,N}=[ey_m(t+1),ey_m(t+2),ey\ (t+3)...,ey_m(t+N] \tag{5.60}$$

$$E\theta_{m,t,N}=[e\theta_m(t+1),e\theta_m(t+2),e\theta_m(t+3)...,e\theta_m(t+N] \tag{5.61}$$

Two-feedback control signals can be determined by:

$$\tau'_{1t,N} = [\tau'_1(t), \tau'_1(t+1), \tau'_1(t+2), ..., \tau'_1(t+N-1)] \tag{5.62}$$

$$\tau'_{2t,N} = [\tau'_2(t), \tau'_2(t+1), \tau'_2(t+2), ..., \tau'_2(t+N-1)] \tag{5.63}$$

Assuming the following objective function:

$$J_1 = \frac{1}{2} Q[(EX_{m,t,N} EX_{m,t,N}{}^T) + (EY_{m,t,N} EY_{m,t,N}{}^T) + (E\theta_{m,t,N} E\theta_{m,t,N}{}^T)]$$

$$+ \frac{1}{2} R[(\tau'_{1t,N} \tau'_{1t,N}{}^T) + (\tau'_{2t,N} \tau'_{2t,N}{}^T)] \tag{5.64}$$

then it is aimed to find τ'_1 and τ'_2 such that J_1 is minimised using the gradient descent rule. The new control actions will be given by:

$$\tau'_{1t,N}{}^{K+1} = \tau'_{1t,N}{}^K + \Delta\tau'_{1t,N}{}^K \tag{5.65}$$

$$\tau'_{2t,N}{}^{K+1} = \tau'_{2t,N}{}^K + \Delta\tau'_{2t,N}{}^K \tag{5.66}$$

where *k* here indicates that calculations are performed at the k^{th} sample; and

$$\Delta\tau'_{1t,N}{}^K = -\eta \frac{\partial J_1}{\partial \tau'_{1t,N}{}^K} = [\Delta\tau'_1(t), \Delta\tau'_1(t+1), \Delta\tau'_1(t+2), ...\Delta\tau'_1(t+N-1)] \tag{5.67}$$

$$\Delta\tau'_{2t,N}{}^K = -\eta \frac{\partial J_1}{\partial \tau'_{2t,N}{}^K} = [\Delta\tau'_2(t), \Delta\tau'_2(t+1), \Delta\tau'_2(t+2), ...\Delta\tau'_2(t+N-1)] \tag{5.68}$$

$$-\eta \frac{\partial J_1}{\partial \tau'_{1t,N}{}^K} = \eta QEX_{m,t,N} \frac{\partial X_{mt,N}}{\partial \tau'_{1t,N}{}^K} + \eta QEY_{m,t,N} \frac{\partial Y_{mt,N}}{\partial \tau'_{1t,N}{}^K}$$

$$+ \eta QE\theta_{m,t,N} \frac{\partial \theta_{mt,N}}{\partial \tau'_{1t,N}{}^K} - \eta R\partial\tau'_{1t,N}{}^K \tag{5.69}$$

$$-\eta \frac{\partial J_1}{\partial \tau'_{2t,N}{}^K} = \eta QEX_{m,t,N} \frac{\partial X_{mt,N}}{\partial \tau'_{2t,N}{}^K} + \eta QEY_{m,t,N} \frac{\partial Y_{mt,N}}{\partial \tau'_{2t,N}{}^K}$$

$$+ \eta QE\theta_{m,t,N} \frac{\partial \theta_{mt,N}}{\partial \tau'_{2t,N}{}^K} - \eta R\partial\tau'_{2t,N}{}^K \tag{5.70}$$

where

$$\frac{\partial X_{mt,N}}{\partial \tau'_{1t,N}{}^K} = \left[\frac{\partial x_m(t+1)}{\partial \tau'_1(t)} \quad \frac{\partial x_m(t+2)}{\partial \tau'_1(t+1)} \quad ... \quad \frac{\partial x_m(t+N)}{\partial \tau'_1(t+N-1)} \right] \tag{5.71}$$

$$\frac{\partial X_{mt,N}}{\partial \tau'_{2t,N}{}^K} = \left[\frac{\partial x_m(t+1)}{\partial \tau'_2(t)} \quad \frac{\partial x_m(t+2)}{\partial \tau'_2(t+1)} \quad ... \quad \frac{\partial x_m(t+N)}{\partial \tau'_2(t+N-1)} \right] \tag{5.72}$$

$$\frac{\partial Y_{m_{t,N}}}{\partial \tau'^{\,K}_{1_{t,N}}} = \left[\frac{\partial y_m(t+1)}{\partial \tau'_1(t)} \quad \frac{\partial y_m(t+2)}{\partial \tau'_1(t+1)} \quad \cdots \quad \frac{\partial y_m(t+N)}{\partial \tau'_1(t+N-1)} \right] \tag{5.73}$$

$$\frac{\partial Y_{m_{t,N}}}{\partial \tau'^{\,K}_{2_{t,N}}} = \left[\frac{\partial y_m(t+1)}{\partial \tau'_2(t)} \quad \frac{\partial y_m(t+2)}{\partial \tau'_2(t+1)} \quad \cdots \quad \frac{\partial y_m(t+N)}{\partial \tau'_2(t+N-1)} \right] \tag{5.74}$$

$$\frac{\partial \theta_{m_{t,N}}}{\partial \tau'^{\,K}_{1_{t,N}}} = \left[\frac{\partial \theta_m(t+1)}{\partial \tau'_1(t)} \quad \frac{\partial \theta_m(t+2)}{\partial \tau'_1(t+1)} \quad \cdots \quad \frac{\partial \theta_m(t+N)}{\partial \tau'_1(t+N-1)} \right] \tag{5.75}$$

$$\frac{\partial \theta_{m_{t,N}}}{\partial \tau'^{\,K}_{2_{t,N}}} = \left[\frac{\partial \theta_m(t+1)}{\partial \tau'_2(t)} \quad \frac{\partial \theta_m(t+2)}{\partial \tau'_2(t+1)} \quad \cdots \quad \frac{\partial \theta_m(t+N)}{\partial \tau'_2(t+N-1)} \right] \tag{5.76}$$

It can be seen that each element in the above vectors can be obtained such that:

$$net_j = \sum_{i=1}^{nh} VH_{ji} \times G_i + \sum_{c=1}^{C} VC_{jc} \times h_c^o + bias \times \overline{Vb_j} \tag{5.77}$$

where $j=c$ and $nh=C$ are the number of the hidden and context nodes respectively and $\overline{G}$ is the input vector such as

$$G = [\tau_R(t), \tau_L(t), x_m(t), y_m(t), \theta_m(t)]^T \tag{5.78}$$

$$h_j = \frac{2}{1+e^{-net_j}} - 1 \tag{5.79}$$

$$f(net_j)' = 0.5(1-h_j^2) \tag{5.80}$$

$$\frac{\partial x_m(t+1)}{\partial \tau'_1(t)} = \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} \tag{5.81}$$

$$\frac{\partial y_m(t+1)}{\partial \tau'_1(t)} = \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j1} \tag{5.82}$$

$$\frac{\partial \theta_m(t+1)}{\partial \tau'_1(t)} = \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j1} \tag{5.83}$$

$$\frac{\partial x_m(t+1)}{\partial \tau'_2(t)} = \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j2} \tag{5.84}$$

$$\frac{\partial y_m(t+1)}{\partial \tau'_2(t)} = \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j2} \tag{5.85}$$

$$\frac{\partial \theta_m(t+1)}{\partial \tau'_2(t)} = \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} \tag{5.86}$$

Equations 5.71 to 5.76 are the well-known Jacobian vectors, which must be calculated using equations 5.81 to 5.86 every time a new control signal has to be determined. This

could result in a large computation for a large N. Therefore, recursive methods for calculating the Jacobian vectors are developed so that the algorithm can be applied to real-time systems. After completing the procedure from n=1 to N the new control actions for the next sample will be:

$$\tau_R(k+1) = \tau_{ref1}(k+1) + \tau_1'^K(t+N) \tag{5.87}$$

$$\tau_L(k+1) = \tau_{ref2}(k+1) + \tau_2'^K(t+N) \tag{5.88}$$

where $\tau_1'^k(t+N) \& \tau_2'^k(t+N)$ are the final values of the feedback-controlling signals calculated by the optimisation algorithm. This is calculated at each sample time k so that $\tau_R(k+1) \& \tau_L(k+1)$ are torque control actions of the right and the left wheels respectively. These actions will be applied to the mobile robot system and the position and orientation identifier model at the next sampling time. The application of this procedure will continue at the next sampling time (k+1) until the error between the desired input and the actual output becomes lower than a pre-specified value.

5.3.2.2. Nonlinear PID Neural Control Methodology

Second control methodology of the feedback controller is nonlinear PID neural predictive feedback controller which consists of nonlinear PID neural networks and posture identifier with predictive optimisation algorithm. A PID controller consists of three terms: proportional, integral and derivative. The standard form of a PID controller is given in the s-domain as equation 5.89 [125].

$$Gc(s) = P + I + D = K_p + \frac{K_i}{s} + K_d s \tag{5.89}$$

where K_p, K_i and K_d are called the proportional gain, the integral gain and the derivative gain respectively. In time domain, the output of the PID controller u(t) can be described as follows [125]:

$$u(t) = K_p e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \tag{5.90}$$

where e(t) is the input to the controller.

For MIMO nonlinear system cannot use the classical PID controller, therefore the nonlinear PID neural controller with self-tuning parameters techniques is necessary to use

with this MIMO nonlinear system in order to overcome the external disturbances and parameter variations which are unpredictable and cannot be modelled accurately. The proposed structure of the nonlinear PID neural controller is shown in Figure 5.6.

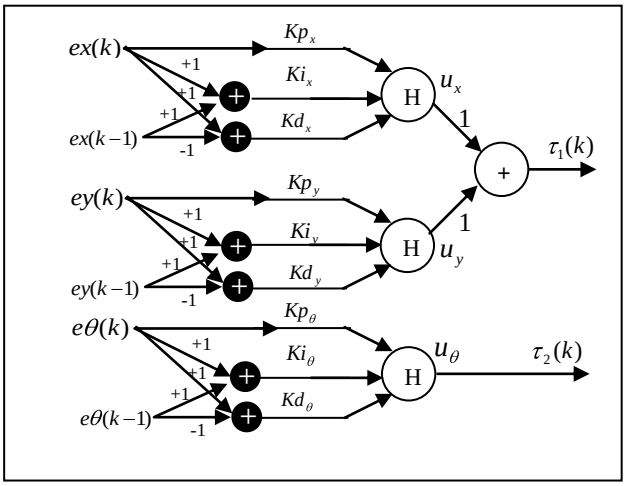


Figure 5.6: The nonlinear PID neural feedback controller structure.

The proposed structure of the nonlinear PID neural consists of position and orientation nonlinear PID controllers. The position nonlinear PID controller depends on the x-coordinate error and y-coordinate error while the orientation nonlinear PID controller depends on the θ-angular error.

The proposed control law of the feedback torque of right and left wheel (τ_1 and τ_2) respectively can be proposed as follows:

$$\tau_1 = u_x + u_y \tag{5.91}$$

$$\tau_2 = u_\theta \tag{5.92}$$

where u_x, u_y and u_θ are the outputs of the neural networks that can be obtained from sigmoid activation function which has nonlinear relationship as presented in the following function:

$$u_x = \frac{2}{1+e^{-netu_x}} - 1 \tag{5.93}$$

$$u_y = \frac{2}{1+e^{-netu_y}} - 1 \tag{5.94}$$

$$u_\theta = \frac{2}{1+e^{-netu_\theta}} - 1 \tag{5.95}$$

$netu_x, netu_y$ and $netu_\theta$ are calculated from these equations

$$netu_x = ex(k)Kp_x + (ex(k) + ex(k-1))Ki_x + (ex(k) - ex(k-1))Kd_x \quad (5.96)$$

$$netu_y = ey(k)Kp_y + (ey(k) + ey(k-1))Ki_y + (ey(k) - ey(k-1))Kd_y \quad (5.97)$$

$$netu_\theta = e\theta(k)Kp_\theta + (e\theta(k) + e\theta(k-1))Ki_\theta + (e\theta(k) - e\theta(k-1))Kd_\theta \quad (5.98)$$

The control parameters Kp, Ki and Kd are the proportional, integral, and derivative gains respectively. The control parameters Kp, Ki and Kd of the self-tuning nonlinear PID for position and orientation controllers are adjusted using the gradient-descent delta rule method for one-step-ahead when N=1.

The update rules for these control parameters are expressed by:

$$Kp_\gamma(k+1) = Kp_\gamma(k) + \Delta Kp_\gamma(k+1) \quad (5.99)$$

$$Ki_\gamma(k+1) = Ki_\gamma(k) + \Delta Ki_\gamma(k+1) \quad (5.100)$$

$$Kd_\gamma(k+1) = Kd_\gamma(k) + \Delta Kd_\gamma(k+1) \quad (5.101)$$

$$\Delta(Kp_\gamma, Ki_\gamma, Kd_\gamma)(k+1) = -\eta \frac{\partial J}{\partial(Kp_\gamma, Ki_\gamma, Kd_\gamma)(k)} \quad (5.102)$$

where γ is x, y, θ for each time.

The use of the minus sign accounts for the gradient-descent in weight-space reflecting, the seek of the direction for weight, change that reduces the value of *J* in equation 5.37. The learning rate η is small value for smoothness weights changing of the PID controllers parameters.

By applying the chain rule, the terms $\frac{\partial J}{\partial(Kp_x, Ki_x, Kd_x)(k)}$ for position x-coordinate error is represented as follows:

$$\Delta Kp_x(k+1) = -\eta \frac{\partial J}{\partial Kp_x(k)} = -\eta \frac{\frac{1}{2}Q\partial((x_r(k+1) - x_m(k+1))^2}{\partial Kp_x(k)} - \eta \frac{\frac{1}{2}R\partial(\tau_1(k))^2}{\partial Kp_x(k)} \quad (5.103)$$

$$\Delta Kp_x(k+1) = \eta Qex(k+1)\frac{\partial x_m(k+1)}{\partial Kp_x(k)} - \eta R \frac{\partial \tau_1(k)}{\partial Kp_x(k)} \quad (5.104)$$

$$\frac{\partial x_m(k+1)}{\partial Kp_x(k)} = \frac{\partial x_m(k+1)}{\partial o_1}\frac{\partial o_1}{\partial net_1}\frac{\partial net_1}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_R(k)}\frac{\partial \tau_R(k)}{\partial \tau_1(k)}\frac{\partial \tau_1(k)}{\partial u_x(k)}\frac{\partial u_x(k)}{\partial netu_x(k)}\frac{\partial netu_x(k)}{\partial Kp_x(k)} \quad (5.105)$$

$$\frac{\partial x_m(k+1)}{\partial Kp_x(k)} = \frac{\partial net_1}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_R(k)}\frac{\partial \tau_R(k)}{\partial \tau_1(k)}\frac{\partial \tau_1(k)}{\partial u_x(k)}\frac{\partial u_x(k)}{\partial netu_x(k)}\frac{\partial netu_x(k)}{\partial Kp_x(k)} \tag{5.106}$$

$$\tau_R(k) = \tau_{ref\,1}(k) + \tau_1(k) \tag{5.107}$$

$$\frac{\partial x_m(k+1)}{\partial Kp_x(k)} = \frac{\partial net_1}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_R(k)}\frac{\partial u_x(k)}{\partial netu_x(k)}\frac{\partial netu_x(k)}{\partial Kp_x(k)} \tag{5.108}$$

Figure 5.5 shows that $\tau_R(k)$ is linked to the exciting nodes, VH_{j1}

$$\frac{\partial x_m(k+1)}{\partial Kp_x(k)} = ex(k)\, f(netu_x)' \sum_{j=1}^{nh} W_{1j}\, f(net_j)' VH_{j1} \tag{5.109}$$

$$\frac{\partial \tau_1(k)}{\partial Kp_x(k)} = \frac{\partial \tau_1(k)}{\partial u_x(k)}\frac{\partial u_x(k)}{\partial netu_x(k)}\frac{\partial netu_x(k)}{\partial Kp_x(k)} \tag{5.110}$$

$$\frac{\partial \tau_1(k)}{\partial Kp_x(k)} = f(netu_x)' ex(k) \tag{5.111}$$

$$\Delta Kp_x(k+1) = ex(k)\, f(netu_x)' (\eta Q ex(k+1) \sum_{j=1}^{nh} W_{1j}\, f(net_j)' VH_{j1} - \eta R) \tag{5.112}$$

and for $\Delta Ki_x(k+1)$ and $\Delta Kd_x(k+1)$ are

$$\Delta Ki_x(k+1) = (ex(k) + ex(k-1))\, f(netu_x)' (\eta Q ex(k+1) \sum_{j=1}^{nh} W_{1j}\, f(net_j)' VH_{j1} - \eta R) \tag{5.113}$$

$$\Delta Kd_x(k+1) = (ex(k) - ex(k-1))\, f(netu_x)' (\eta Q ex(k+1) \sum_{j=1}^{nh} W_{1j}\, f(net_j)' VH_{j1} - \eta R) \tag{5.114}$$

By applying the chain rule, the terms $\frac{\partial J}{\partial (Kp_y, Ki_y, Kd_y)(k)}$ for position y-coordinate error is represented as follows:

$$\Delta Kp_y(k+1) = -\eta \frac{\partial J}{\partial Kp_y(k)} = -\eta \frac{\frac{1}{2} Q \partial ((y_r(k+1) - y_m(k+1))^2}{\partial Kp_y(k)} - \eta \frac{\frac{1}{2} R \partial (\tau_1(k))^2}{\partial Kp_y(k)} \tag{5.115}$$

$$\Delta Kp_y(k+1) = ey(k)\, f(netu_y)' (\eta Q ey(k+1) \sum_{j=1}^{nh} W_{2j}\, f(net_j)' VH_{j1} - \eta R) \tag{5.116}$$

$$\Delta Ki_y(k+1) = (ey(k) + ey(k-1))\, f(netu_y)' (\eta Q ey(k+1) \sum_{j=1}^{nh} W_{2j}\, f(net_j)' VH_{j1} - \eta R) \tag{5.117}$$

$$\Delta Kd_y(k+1) = (ey(k) - ey(k-1))\, f(netu_y)' (\eta Q ey(k+1) \sum_{j=1}^{nh} W_{2j}\, f(net_j)' VH_{j1} - \eta R) \tag{5.118}$$

By applying the chain rule, the terms $\frac{\partial J}{\partial(Kp_\theta, Ki_\theta, Kd_\theta)(k)}$ for θ-orientation error is represented as follows:

$$\Delta Kp_\theta(k+1) = -\eta \frac{\partial J}{\partial Kp_\theta(k)} = -\eta \frac{\frac{1}{2} Q \partial((\theta_r(k+1) - \theta_m(k+1))^2}{\partial Kp_\theta(k)} - \eta \frac{\frac{1}{2} R \partial(\tau_2(k))^2}{\partial Kp_\theta(k)} \tag{5.119}$$

$$\Delta Kp_\theta(k+1) = \eta Qe\theta(k+1) \frac{\partial \theta_m(k+1)}{\partial Kp_\theta(k)} - \eta R \frac{\partial \tau_2(k)}{\partial Kp_\theta(k)} \tag{5.120}$$

$$\frac{\partial \theta_m(k+1)}{\partial Kp_\theta(k)} = \frac{\partial \theta_m(k+1)}{\partial o_3} \frac{\partial o_3}{\partial net_3} \frac{\partial net_3}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_L(k)}{\partial \tau 2(k)} \frac{\partial \tau 2(k)}{\partial u_\theta(k)} \frac{\partial u_\theta(k)}{\partial netu_\theta(k)} \frac{\partial netu_\theta(k)}{\partial Kp_\theta(k)} \tag{5.121}$$

$$\frac{\partial \theta_m(k+1)}{\partial Kp_\theta(k)} = \frac{\partial net_3}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_L(k)}{\partial \tau_2(k)} \frac{\partial \tau_2(k)}{\partial u_\theta(k)} \frac{\partial u_\theta(k)}{\partial netu_\theta(k)} \frac{\partial netu_\theta(k)}{\partial Kp_\theta(k)} \tag{5.122}$$

$$\tau_L(k) = \tau_{ref\,2}(k) + \tau_2(k) \tag{5.123}$$

$$\frac{\partial \theta_m(k+1)}{\partial Kp_\theta(k)} = \frac{\partial net_3}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial u_\theta(k)}{\partial netu_\theta(k)} \frac{\partial netu_\theta(k)}{\partial Kp_\theta(k)} \tag{5.124}$$

Figure 5.5 shows that $\tau_L(k)$ is linked to the exciting nodes VH_{j2}

$$\frac{\partial \theta_m(k+1)}{\partial Kp_\theta(k)} = e\theta(k) f(netu_\theta)' \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} \tag{5.125}$$

$$\frac{\partial \tau_2(k)}{\partial Kp_\theta(k)} = \frac{\partial \tau_2(k)}{\partial u_\theta(k)} \frac{\partial u_\theta(k)}{\partial netu_\theta(k)} \frac{\partial netu_\theta(k)}{\partial Kp_\theta(k)} \tag{5.126}$$

$$\frac{\partial \tau_2(k)}{\partial Kp_\theta(k)} = f(netu_\theta)' e\theta(k) \tag{5.127}$$

$$\Delta Kp_\theta(k+1) = e\theta(k) f(netu_\theta)' (\eta Qe\theta(k+1) \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} - \eta R) \tag{5.128}$$

while for $\Delta Ki_\theta(k+1)$ and $\Delta Kd_\theta(k+1)$ can be obtained as follows:

$$\Delta Ki_\theta(k+1) = (e\theta(k) + e\theta(k-1)) f(netu_\theta)' (\eta Qe\theta(k+1) \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} - \eta R) \tag{5.129}$$

$$\Delta Kd_\theta(k+1) = (e\theta(k) - e\theta(k-1)) f(netu_\theta)' (\eta Qe\theta(k+1) \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} - \eta R) \tag{5.130}$$

After finding these control parameters, $netu_x, netu_y$ and $netu_\theta$ can be calculated, enabling the identification of values u_x, u_y and u_θ. Finally, the feedback control action $\tau_1(k+1) \& \tau_2(k+1)$ for the next sample can be calculated as the proposed law:

$$\tau_1(k+1) = u_x + u_y \tag{5.131}$$

$$\tau_2(k+1) = u_\theta \tag{5.132}$$

The total control action of the nonlinear controller became as:

$$\tau_R(k+1) = \tau_{ref1}(k+1) + \tau_1(k+1) \tag{5.133}$$

$$\tau_L(k+1) = \tau_{ref2}(k+1) + \tau_2(k+1) \tag{5.134}$$

This is calculated at each sample time k and applied to mobile robot system and the position and orientation identifier model. Then we continue to apply this procedure at the next sampling time (k+1) until the error between the desired trajectory and the identifier model output becomes lower than a specified value. The weights of the position and orientation neural network identifier and the weights of the feedforward neural controller are updated after each sampling time in order to minimise the error between $q = [x, y, \theta]^T$ & $q_m = [x_m, y_m, \theta_m]^T$ using back propagation learning algorithm. For N steps estimation of the two feedback PID neural controller actions $\tau_1(k) \& \tau_2(k)$ the techniques of generalized predictive control theory will be used. The N steps estimation of $\tau_1(k) \& \tau_2(k)$ will be calculated for each sample. The position and orientation in the identifier model, shown in Figure 5.5, represent the kinematics and dynamics model of the mobile robot system and will be controlled asymptotically. Therefore, they can be used to predict future values of the model outputs for the next N steps and can be used to find the optimal value of $\tau_1(k) \& \tau_2(k)$ using an optimisation algorithm.

For this purpose, let N be a pre-specified positive integer that is denoted such that the future values of the set point are:

$$X_{rt,N} = [x_r(t+1), x_r(t+2), x_r(t+3), ..., x_r(t+N)] \tag{5.135}$$

$$Y_{rt,N} = [y_r(t+1), y_r(t+2), y_r(t+3), ..., y_r(t+N)] \tag{5.136}$$

$$\theta_{rt,N} = [\theta_r(t+1), \theta_r(t+2), \theta_r(t+3), ..., \theta_r(t+N)] \tag{5.137}$$

where t represents the time instant.

Then the predicted outputs of the robot model used the neural identifier, shown in Figure 5.5, are:

$$X_{mt,N} = [x_m(t+1), x_m(t+2), x_m(t+3),..., x_m(t+N)] \tag{5.138}$$

$$Y_{mt,N} = [y_m(t+1), y_m(t+2), y_m(t+3),..., y_m(t+N)] \tag{5.139}$$

$$\theta_{mt,N} = [\theta_m(t+1), \theta_m(t+2), \theta_m(t+3),..., \theta_m(t+N)] \tag{5.140}$$

Then define the following error vector:

$$EX_{m,t,N} = [ex_m(t+1), ex_m(t+2), ex_m(t+3)..., ex_m(t+N] \tag{5.141}$$

$$EY_{m,t,N} = [ey_m(t+1), ey_m(t+2), ey\ (t+3)..., ey_m(t+N] \tag{5.142}$$

$$E\theta_{m,t,N} = [e\theta_m(t+1), e\theta_m(t+2), e\theta_m(t+3)..., e\theta_m(t+N] \tag{5.143}$$

Two-feedback control signals can be determined by:

$$\tau'_{1t,N} = [\tau'_1(t), \tau'_1(t+1), \tau'_1(t+2),..., \tau'_1(t+N-1)] \tag{5.144}$$

$$\tau'_{2t,N} = [\tau'_2(t), \tau'_2(t+1), \tau'_2(t+2),..., \tau'_2(t+N-1)] \tag{5.145}$$

Assuming the following objective function:

$$J_1 = \frac{1}{2} Q[(EX_{m,t,N} EX_{m,t,N}{}^T) + (EY_{m,t,N} EY_{m,t,N}{}^T) + (E\theta_{m,t,N} E\theta_{m,t,N}{}^T)]$$

$$+ \frac{1}{2} R[(\tau'_{1t,N} \tau'_{1t,N}{}^T) + (\tau'_{2t,N} \tau'_{2t,N}{}^T)] \tag{5.146}$$

Then our purpose is to find $Kp_\gamma, Ki_\gamma,$ and Kd_γ for position nonlinear PID controller and orientation nonlinear PID controller such that J_1 is minimised using the gradient descent rule,

$$Kp'_{\gamma t,N}{}^{K+1} = Kp'_{\gamma t,N}{}^{K} + \Delta Kp'_{\gamma t,N}{}^{K} \tag{5.147}$$

$$Ki'_{\gamma t,N}{}^{K+1} = Ki'_{\gamma t,N}{}^{K} + \Delta Ki'_{\gamma t,N}{}^{K} \tag{5.148}$$

$$Kd'_{\gamma t,N}{}^{K+1} = Kd'_{\gamma t,N}{}^{K} + \Delta Kd'_{\gamma t,N}{}^{K} \tag{5.149}$$

then it is aimed to find τ'_1 and τ'_2 ; the new control actions will be given by:

$$\tau'_{1t,N}{}^{K+1} = U_{xt,N}{}^{K} + U_{yt,N}{}^{K} \tag{5.150}$$

$$\tau'_{2t,N}{}^{K+1} = U_{\theta t,N}{}^{K} \tag{5.151}$$

where *k* here indicates that calculations are performed at the k^{th} sample; and

$$\Delta Kp'_{xt,N}{}^{K} = -\eta \frac{\partial J1}{\partial Kp'_{xt,N}{}^{K}} = [\Delta Kp'_x(t), \Delta Kp'_x(t+1), \Delta Kp'_x(t+2), ...\Delta Kp'_x(t+N-1)] \tag{5.152}$$

$$-\eta \frac{\partial J1}{\partial Kp'_{xt,N}{}^{K}} = \eta QEX_{m,t,N} \frac{\partial X_{mt,N}}{\partial Kp'_{xt,N}{}^{K}} - \eta R \frac{\partial \tau'_{1t,N}{}^{K}}{\partial Kp'_{xt,N}{}^{K}} \tag{5.153}$$

$$\frac{\partial X_{mt,N}}{\partial Kp'_{xt,N}{}^{K}} = \left[\frac{\partial x_m(t+1)}{\partial Kp'_x(t)} \quad \frac{\partial x_m(t+2)}{\partial Kp'_x(t+1)} \quad \frac{\partial x_m(t+3)}{\partial Kp'_x(t+2)} \quad \cdots \quad \frac{\partial x_m(t+N)}{\partial Kp'_x(t+N-1)} \right] \tag{5.154}$$

where:

$$\frac{\partial x_m(t+1)}{\partial Kp_x(t)} = ex_m(t) f(netu_x)' \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} \tag{5.155}$$

$$\frac{\partial \tau'_{1t,N}{}^{K}}{\partial Kp'_{xt,N}{}^{K}} = \left[\frac{\partial \tau'_1(t)}{\partial Kp'_x(t)} \quad \frac{\partial \tau'_1(t+1)}{\partial Kp'_x(t+1)} \quad \frac{\partial \tau'_1(t+2)}{\partial Kp'_x(t+2)} \quad \cdots \quad \frac{\partial \tau'_1(t+N-1)}{\partial Kp'_x(t+N-1)} \right] \tag{5.156}$$

$$\frac{\partial \tau_1(t)}{\partial Kp_x(t)} = f(netu_x)' ex_m(t) \tag{5.157}$$

$$\Delta Ki'_{xt,N}{}^{K} = -\eta \frac{\partial J1}{\partial Ki'_{xt,N}{}^{K}} = [\Delta Ki'_x(t), \Delta Ki'_x(t+1), \Delta Ki'_x(t+2), ...\Delta Ki'_x(t+N-1)] \tag{5.158}$$

$$-\eta \frac{\partial J1}{\partial Ki'_{xt,N}{}^{K}} = \eta QEX_{m,t,N} \frac{\partial X_{mt,N}}{\partial Ki'_{xt,N}{}^{K}} - \eta R \frac{\partial \tau'_{1t,N}{}^{K}}{\partial Ki'_{xt,N}{}^{K}} \tag{5.158}$$

$$\frac{\partial X_{mt,N}}{\partial Ki'_{xt,N}{}^{K}} = \left[\frac{\partial x_m(t+1)}{\partial Ki'_x(t)} \quad \frac{\partial x_m(t+2)}{\partial Ki'_x(t+1)} \quad \frac{\partial x_m(t+3)}{\partial Ki'_x(t+2)} \quad \cdots \quad \frac{\partial x_m(t+N)}{\partial Ki'_x(t+N-1)} \right] \tag{5.159}$$

where:

$$\frac{\partial x_m(t+1)}{\partial Ki'_x(t)} = (ex_m(t) + ex_m(t-1)) f(netu_x)' \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} \tag{5.160}$$

$$\frac{\partial \tau'_{1t,N}{}^{K}}{\partial Ki'_{xt,N}{}^{K}} = \left[\frac{\partial \tau'_1(t)}{\partial Ki'_x(t)} \quad \frac{\partial \tau'_1(t+1)}{\partial Ki'_x(t+1)} \quad \frac{\partial \tau'_1(t+2)}{\partial Ki'_x(t+2)} \quad \cdots \quad \frac{\partial \tau'_1(t+N-1)}{\partial Ki'_x(t+N-1)} \right] \tag{5.161}$$

$$\frac{\partial \tau_1(t)}{\partial Ki'_x(t)} = f(netu_x)'(ex_m(t) + ex_m(t-1)) \tag{5.162}$$

$$\Delta Kd'_{xt,N}{}^{K} = -\eta \frac{\partial J1}{\partial Kd'_{xt,N}{}^{K}} = [\Delta Kd'_x(t), \Delta Kd'_x(t+1), \Delta Kd'_x(t+2), ...\Delta Kd'_x(t+N-1)] \tag{5.163}$$

$$-\eta \frac{\partial J1}{\partial Kd'_{xt,N}{}^{K}} = \eta QEX_{m,t,N} \frac{\partial X_{mt,N}}{\partial Kd'_{xt,N}{}^{K}} - \eta R \frac{\partial \tau'_{1t,N}{}^{K}}{\partial Kd'_{xt,N}{}^{K}} \tag{5.164}$$

$$\frac{\partial X_{mt,N}}{\partial Kd'_{xt,N}{}^{K}} = \left[\frac{\partial x_m(t+1)}{\partial Kd'_x(t)} \quad \frac{\partial x_m(t+2)}{\partial Kd'_x(t+1)} \quad \frac{\partial x_m(t+3)}{\partial Kd'_x(t+2)} \quad \cdots \quad \frac{\partial x_m(t+N)}{\partial Kd'_x(t+N-1)} \right] \tag{5.165}$$

where:

$$\frac{\partial x_m(t+1)}{\partial Kd'_x(t)} = (ex_m(t) - ex_m(t-1))\, f\,(netu_x)' \sum_{j=1}^{nh} W_{1j}\, f\,(net_j)' VH_{j1} \tag{5.166}$$

$$\frac{\partial \tau'_{1_{t,N}}{}^{K}}{\partial Kd'_{x_{t,N}}{}^{K}} = \left[\frac{\partial \tau'_1(t)}{\partial Kd'_x(t)} \quad \frac{\partial \tau'_1(t+1)}{\partial Kd'_x(t+1)} \quad \frac{\partial \tau'_1(t+2)}{\partial Kd'_x(t+2)} \quad \cdots \quad \frac{\partial \tau'_1(t+N-1)}{\partial Kd'_x(t+N-1)} \right] \tag{5.167}$$

$$\frac{\partial \tau_1(t)}{\partial Kd'_x(t)} = f\,(netu_x)'(ex_m(t) - ex_m(t-1)) \tag{5.168}$$

$$\Delta Kp'_{y_{t,N}}{}^{K} = -\eta \frac{\partial J1}{\partial Kp'_{y_{t,N}}{}^{K}} = [\Delta Kp'_y(t), \Delta Kp'_y(t+1), \Delta Kp'_y(t+2), \ldots \Delta Kp'_y(t+N-1)] \tag{5.169}$$

$$-\eta \frac{\partial J1}{\partial Kp'_{y_{t,N}}{}^{K}} = \eta QEY_{m,t,N} \frac{\partial Y_{m,t,N}}{\partial Kp'_{y_{t,N}}{}^{K}} - \eta R \frac{\partial \tau'_{1_{t,N}}{}^{K}}{\partial Kp'_{y_{t,N}}{}^{K}} \tag{5.170}$$

$$\frac{\partial Y_{m_{t,N}}}{\partial Kp'_{y_{t,N}}{}^{K}} = \left[\frac{\partial y_m(t+1)}{\partial Kp'_y(t)} \quad \frac{\partial y_m(t+2)}{\partial Kp'_y(t+1)} \quad \frac{\partial y_m(t+3)}{\partial Kp'_y(t+2)} \quad \cdots \quad \frac{\partial y_m(t+N)}{\partial Kp'_y(t+N-1)} \right] \tag{5.171}$$

where:

$$\frac{\partial y_m(t+1)}{\partial Kp_y(t)} = ey_m(t)\, f\,(netu_y)' \sum_{j=1}^{nh} W_{2j}\, f\,(net_j)' VH_{j1} \tag{5.172}$$

$$\frac{\partial \tau'_{1_{t,N}}{}^{K}}{\partial Kp'_{y_{t,N}}{}^{K}} = \left[\frac{\partial \tau'_1(t)}{\partial Kp'_y(t)} \quad \frac{\partial \tau'_1(t+1)}{\partial Kp'_y(t+1)} \quad \frac{\partial \tau'_1(t+2)}{\partial Kp'_y(t+2)} \quad \cdots \quad \frac{\partial \tau'_1(t+N-1)}{\partial Kp'_y(t+N-1)} \right] \tag{5.173}$$

$$\frac{\partial \tau_1(t)}{\partial Kp_y(t)} = f\,(netu_y)' ey_m(t) \tag{5.174}$$

$$\Delta Ki'_{y_{t,N}}{}^{K} = -\eta \frac{\partial J1}{\partial Ki'_{y_{t,N}}{}^{K}} = [\Delta Ki'_y(t), \Delta Ki'_y(t+1), \Delta Ki'_y(t+2), \ldots \Delta Ki'_y(t+N-1)] \tag{5.175}$$

$$-\eta \frac{\partial J1}{\partial Ki'_{y_{t,N}}{}^{K}} = \eta QEY_{m,t,N} \frac{\partial Y_{m_{t,N}}}{\partial Ki'_{y_{t,N}}{}^{K}} - \eta R \frac{\partial \tau'_{1_{t,N}}{}^{K}}{\partial Ki'_{y_{t,N}}{}^{K}} \tag{5.176}$$

$$\frac{\partial Y_{m_{t,N}}}{\partial Ki'_{y_{t,N}}{}^{K}} = \left[\frac{\partial y_m(t+1)}{\partial Ki'_y(t)} \quad \frac{\partial y_m(t+2)}{\partial Ki'_y(t+1)} \quad \frac{\partial y_m(t+3)}{\partial Ki'_y(t+2)} \quad \cdots \quad \frac{\partial y_m(t+N)}{\partial Ki'_y(t+N-1)} \right] \tag{5.177}$$

where:

$$\frac{\partial y_m(t+1)}{\partial Ki'_y(t)} = (ey_m(t) + ey_m(t-1))\, f\,(netu_y)' \sum_{j=1}^{nh} W_{2j}\, f\,(net_j)' VH_{j1} \tag{5.178}$$

$$\frac{\partial \tau'_{1_{t,N}}{}^{K}}{\partial Ki'_{y_{t,N}}{}^{K}} = \left[\frac{\partial \tau'_1(t)}{\partial Ki'_y(t)} \quad \frac{\partial \tau'_1(t+1)}{\partial Ki'_y(t+1)} \quad \frac{\partial \tau'_1(t+2)}{\partial Ki'_y(t+2)} \quad \cdots \quad \frac{\partial \tau'_1(t+N-1)}{\partial Ki'_y(t+N-1)} \right] \tag{5.179}$$

$$\frac{\partial \tau_1(t)}{\partial Ki'_y(t)} = f(netu_y)'(ey_m(t) + ey_m(t-1)) \tag{5.180}$$

$$\Delta Kd'_{y_{t,N}}{}^{K} = -\eta \frac{\partial J1}{\partial Kd'_{y_{t,N}}{}^{K}} = [\Delta Kd'_y(t), \Delta Kd'_y(t+1), \Delta Kd'_y(t+2), ...\Delta Kd'_y(t+N-1)] \tag{5.181}$$

$$-\eta \frac{\partial J1}{\partial Kd'_{y_{t,N}}{}^{K}} = \eta QEY_{m,t,N} \frac{\partial Y_{mt,N}}{\partial Kd'_{y_{t,N}}{}^{K}} - \eta R \frac{\partial \tau'_{1t,N}{}^{K}}{\partial Kd'_{y_{t,N}}{}^{K}} \tag{5.182}$$

$$\frac{\partial Y_{mt,N}}{\partial Kd'_{y_{t,N}}{}^{K}} = \left[\frac{\partial y_m(t+1)}{\partial Kd'_y(t)} \quad \frac{\partial y_m(t+2)}{\partial Kd'_y(t+1)} \quad \frac{\partial y_m(t+3)}{\partial Kd'_y(t+2)} \quad ... \quad \frac{\partial y_m(t+N)}{\partial Kd'_y(t+N-1)} \right] \tag{5.183}$$

where:

$$\frac{\partial y_m(t+1)}{\partial Kd'_y(t)} = (ey_m(t) - ey_m(t-1)) f(netu_y)' \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j1} \tag{5.184}$$

$$\frac{\partial \tau'_{1t,N}{}^{K}}{\partial Kd'_{y_{t,N}}{}^{K}} = \left[\frac{\partial \tau'_1(t)}{\partial Kd'_y(t)} \quad \frac{\partial \tau'_1(t+1)}{\partial Kd'_y(t+1)} \quad \frac{\partial \tau'_1(t+2)}{\partial Kd'_y(t+2)} \quad ... \quad \frac{\partial \tau'_1(t+N-1)}{\partial Kd'_y(t+N-1)} \right] \tag{5.185}$$

$$\frac{\partial \tau'_1(t)}{\partial Kd'_y(t)} = f(netu_y)'(ey_m(t) - ey_m(t-1)) \tag{5.186}$$

$$\Delta Kp'_{\theta_{t,N}}{}^{K} = -\eta \frac{\partial J1}{\partial Kp'_{\theta_{t,N}}{}^{K}} = [\Delta Kp'_\theta(t), \Delta Kp'_\theta(t+1), \Delta Kp'_\theta(t+2), ...\Delta Kp'_\theta(t+N-1)] \tag{5.187}$$

$$-\eta \frac{\partial J1}{\partial Kp'_{\theta_{t,N}}{}^{K}} = \eta QE\theta_{m,t,N} \frac{\partial \theta_{mt,N}}{\partial Kp'_{\theta_{t,N}}{}^{K}} - \eta R \frac{\partial \tau'_{2t,N}{}^{K}}{\partial Kp'_{\theta_{t,N}}{}^{K}} \tag{5.188}$$

$$\frac{\partial \theta_{mt,N}}{\partial Kp'_{\theta_{t,N}}{}^{K}} = \left[\frac{\partial \theta_m(t+1)}{\partial Kp'_\theta(t)} \quad \frac{\partial \theta_m(t+2)}{\partial Kp'_\theta(t+1)} \quad \frac{\partial \theta_m(t+3)}{\partial Kp'_\theta(t+2)} \quad ... \quad \frac{\partial \theta_m(t+N)}{\partial Kp'_\theta(t+N-1)} \right] \tag{5.189}$$

where:

$$\frac{\partial \theta_m(t+1)}{\partial Kp_\theta(t)} = e\theta_m(t) f(netu_\theta)' \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} \tag{5.190}$$

$$\frac{\partial \tau'_{2t,N}{}^{K}}{\partial Kp'_{\theta_{t,N}}{}^{K}} = \left[\frac{\partial \tau'_2(t)}{\partial Kp'_\theta(t)} \quad \frac{\partial \tau'_2(t+1)}{\partial Kp'_\theta(t+1)} \quad \frac{\partial \tau'_2(t+2)}{\partial Kp'_\theta(t+2)} \quad ... \quad \frac{\partial \tau'_2(t+N-1)}{\partial Kp'_\theta(t+N-1)} \right] \tag{5.191}$$

$$\frac{\partial \tau'_2(t)}{\partial Kp_\theta(t)} = f(netu_\theta)' e\theta_m(t) \tag{5.192}$$

$$\Delta Ki'_{\theta_{t,N}}{}^{K} = -\eta \frac{\partial J1}{\partial Ki'_{\theta_{t,N}}{}^{K}} = [\Delta Ki'_\theta(t), \Delta Ki'_\theta(t+1), \Delta Ki'_\theta(t+2), ...\Delta Ki'_\theta(t+N-1)] \tag{5.193}$$

$$-\eta\frac{\partial J1}{\partial Ki'_{\theta_{t,N}}{}^{K}}=\eta QE\theta_{t,N}\frac{\partial\theta_{m_{t,N}}}{\partial Ki'_{\theta_{t,N}}{}^{K}}-\eta R\frac{\partial Ufb'_{2_{t,N}}{}^{K}}{\partial Ki'_{\theta_{t,N}}{}^{K}} \tag{5.194}$$

$$\frac{\partial\theta_{m_{t,N}}}{\partial Ki'_{\theta_{t,N}}{}^{K}}=\left[\frac{\partial\theta_m(t+1)}{\partial Ki'_\theta(t)}\quad\frac{\partial\theta_m(t+2)}{\partial Ki'_\theta(t+1)}\quad\frac{\partial\theta_m(t+3)}{\partial Ki'_\theta(t+2)}\quad\ldots\quad\frac{\partial\theta_m(t+N)}{\partial Ki'_\theta(t+N-1)}\right] \tag{5.195}$$

where:

$$\frac{\partial\theta_m(t+1)}{\partial Ki'_\theta(t)}=(e\theta_m(t)+e\theta_m(t-1))f(netu_\theta)'\sum_{j=1}^{nh}W_{3j}f(net_j)'VH_{j2} \tag{5.196}$$

$$\frac{\partial\tau'_{2_{t,N}}{}^{K}}{\partial Ki'_{\theta_{t,N}}{}^{K}}=\left[\frac{\partial\tau'_2(t)}{\partial Ki'_\theta(t)}\quad\frac{\partial\tau'_2(t+1)}{\partial Ki'_\theta(t+1)}\quad\frac{\partial\tau'_2(t+2)}{\partial Ki'_\theta(t+2)}\quad\ldots\quad\frac{\partial\tau'_2(t+N-1)}{\partial Ki'_\theta(t+N-1)}\right] \tag{5.197}$$

$$\frac{\partial\tau_2(t)}{\partial Ki'_\theta(t)}=f(netu_\theta)'(e\theta_m(t)+e\theta_m(t-1)) \tag{5.198}$$

$$\Delta Kd'_{\theta_{t,N}}{}^{K}=-\eta\frac{\partial J1}{\partial Kd'_{\theta_{t,N}}{}^{K}}=[\Delta Kd'_\theta(t),\Delta Kd'_\theta(t+1),\Delta Kd'_\theta(t+2),\ldots\Delta Kd'_\theta(t+N-1)] \tag{5.199}$$

$$-\eta\frac{\partial J1}{\partial Kd'_{\theta_{t,N}}{}^{K}}=\eta QE\theta_{m,t,N}\frac{\partial\theta_{m_{t,N}}}{\partial Kd'_{\theta_{t,N}}{}^{K}}-\eta R\frac{\partial\tau'_{2_{t,N}}{}^{K}}{\partial Kd'_{\theta_{t,N}}{}^{K}} \tag{5.200}$$

$$\frac{\partial\theta_{m_{t,N}}}{\partial Kd'_{\theta_{t,N}}{}^{K}}=\left[\frac{\partial\theta_m(t+1)}{\partial Kd'_\theta(t)}\quad\frac{\partial\theta_m(t+2)}{\partial Kd'_\theta(t+1)}\quad\frac{\partial\theta_m(t+3)}{\partial Kd'_\theta(t+2)}\quad\ldots\quad\frac{\partial\theta_m(t+N)}{\partial Kd'_\theta(t+N-1)}\right] \tag{5.201}$$

where:

$$\frac{\partial\theta_m(t+1)}{\partial Kd'_\theta(t)}=(e\theta_m(t)-e\theta_m(t-1))f(netu_\theta)'\sum_{j=1}^{nh}W_{3j}f(net_j)'VH_{j2} \tag{5.202}$$

$$\frac{\partial\tau'_{2_{t,N}}{}^{K}}{\partial Kd'_{\theta_{t,N}}{}^{K}}=\left[\frac{\partial\tau'_2(t)}{\partial Kd'_\theta(t)}\quad\frac{\partial\tau'_2(t+1)}{\partial Kd'_\theta(t+1)}\quad\frac{\partial\tau'_2(t+2)}{\partial Kd'_\theta(t+2)}\quad\ldots\quad\frac{\partial\tau'_2(t+N-1)}{\partial Kd'_\theta(t+N-1)}\right] \tag{5.203}$$

$$\frac{\partial\tau_2(t)}{\partial Kd'_\theta(t)}=f(netu_\theta)'(e\theta_m(t)-e\theta_m(t-1)) \tag{5.204}$$

After completing the procedure from n=1 to N, the new control actions for the next sample will be:

$$\tau_R(k+1)=\tau_{ref1}(k+1)+\tau_1'^{K}(t+N) \tag{5.205}$$

$$\tau_L(k+1)=\tau_{ref2}(k+1)+\tau_2'^{K}(t+N) \tag{5.206}$$

where $\tau_1'^{k}(t+N)\,\&\,\tau_2'^{k}(t+N)$ are the final values of the feedback-controlling signals calculated by the optimisation algorithm.

This is calculated at each sample time k so that $\tau_R(k+1)\,\&\,\tau_L(k+1)$ are torque control actions of the right and the left wheels respectively. These actions will be applied to the mobile robot system and the position and orientation identifier model at the next sampling time. The application of this procedure will continue at the next sampling time (k+1) until the error between the desired input and the actual output becomes lower than a pre-specified value.

5.3.2.3 Nonlinear Inverse-Dynamic Neural Control Methodology

The third control methodology of the feedback controller is nonlinear inverse dynamic predictive feedback neural controller, which consists of the nonlinear feedback acceleration control equation based on back-stepping technique and Lyapunov stability method and posture neural network identifier with optimisation predictive algorithm. This methodology has converted the steering system commands (velocities) to torques commands and has taken into account the parameters of the actual mobile robot.

The proposed nonlinear inverse dynamic feedback equation is:

$$\tau_{1,2} = f_{\tau_{1,2}}(q_m, \frac{df_c(q_{em}, v_r, w_r, K)}{dt}) = B(q_m)^{-1}[M(q_m)\dot{v}_c + \tau_d] \tag{5.207}$$

where $q_m = (x_m, y_m, \theta_m)^T$ is identifier network posture state vector.

The structure of the nonlinear inverse dynamic feedback controller can be shown in Figure 5.7.

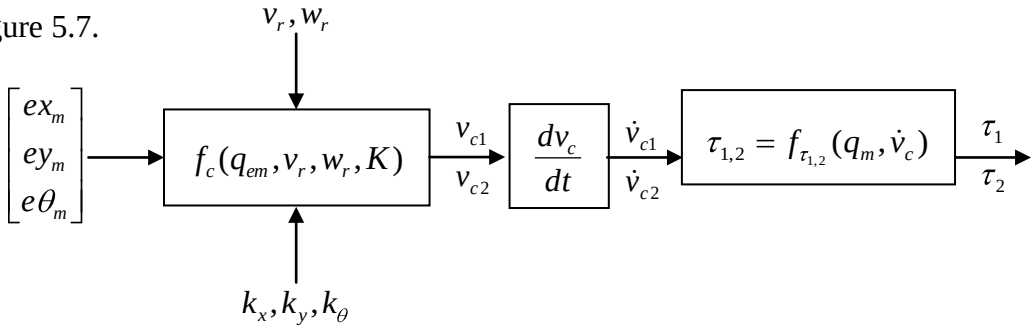


Figure 5.7: The nonlinear inverse-dynamic feedback controller (NIDFC) structure.

To facilitate the closed loop tracking error system development and stability analysis, a global invertible transformation equation 5.14 was used. The configuration error with state vector is $q_{em} = [ex_m, ey_m, e\theta_m]^T$. The desired trajectory can be described by a virtual desired robot with a state vector $q_r = [x_r, y_r, \theta_r]^T$ generated by desired mobile robot whose equations of motion are:

$$\dot{x}_r = v_r \cos\theta_r \tag{5.208}$$

$$\dot{y}_r = v_r \sin\theta_r \tag{5.209}$$

$$\dot{\theta}_r = w_r \tag{5.210}$$

After taking the time derivative of equation 5.14, the derivation configuration error for the mobile robot becomes as follows (see appendix C):

$$\begin{bmatrix} e\dot{x}_m \\ e\dot{y}_m \\ e\dot{\theta}_m \end{bmatrix} = \begin{bmatrix} v_w ey_m - v_i + v_r \cos e\theta_m \\ -v_w ex_m + v_r \sin e\theta_m \\ w_r - v_w \end{bmatrix} \tag{5.211}$$

The reference linear velocity and the reference angular velocity are given by equations 5.212 and 5.213 respectively [94].

$$v_r = \sqrt{(\dot{x}_r)^2 + (\dot{y}_r)^2} \tag{5.212}$$

$$w_r = \frac{\ddot{y}_r \dot{x}_r - \ddot{x}_r \dot{y}_r}{(\dot{x}_r)^2 + (\dot{y}_r)^2} \tag{5.213}$$

with $v_r > 0$ and $w_r > 0$, for all t, determine a smooth velocity control law $v_c = f_c(q_{em}, v_r, w_r, K)$ such that $\lim_{t\to\infty}(q_r - q_m) = 0$ is asymptotically stable.

where q_{em}, v_r, w_r and K are the tracking posture model error, the reference linear and angular velocity and control gain respectively.

The objective of the nonlinear feedback controller is to design a controller for the transformed kinematics model of the mobile robot that forces the actual Cartesian position and orientation of the neural network identifier to a constant desired position and orientation. Based on this control objective, a differentiable, time-varying controller is proposed as follows:

$$v_c = \begin{bmatrix} v_i \\ v_w \end{bmatrix} = \begin{bmatrix} v_r \cos e\theta_m + k_x ex_m \\ w_r + k_y v_r ey_m + k_\theta v_r \sin e\theta_m \end{bmatrix} \tag{5.214}$$

where $k_x, k_y, k_\theta > 0$ are design control gain parameters.

The proposed nonlinear feedback acceleration control input is the time derivative of v_c as follows:

$$\dot{v}_c = \begin{bmatrix} \dot{v}_r \cos e\theta_m \\ \dot{w}_r + k_y \dot{v}_r ey_m + k_\theta \dot{v}_r \sin e\theta_m \end{bmatrix} + \begin{bmatrix} k_x & 0 & -vr \sin e\theta_m \\ 0 & k_y v_r & k_\theta v_r \cos e\theta_m \end{bmatrix} \begin{bmatrix} e\dot{x}_m \\ e\dot{y}_m \\ e\dot{\theta}_m \end{bmatrix} \tag{5.215}$$

Assuming that the linear and angular reference velocities are constants obtains:

$$\dot{v}_c = \begin{bmatrix} k_x & 0 & -vr \sin e\theta_m \\ 0 & k_y v_r & k_\theta v_r \cos e\theta_m \end{bmatrix} \begin{bmatrix} e\dot{x}_m \\ e\dot{y}_m \\ e\dot{\theta}_m \end{bmatrix} \tag{5.216}$$

The Lyapunov based nonlinear are the simplest but also successful methods in kinematics stabilisation. A constructive Lyapunov functions method is considered based on [126, 127] as follows:

$$V = \frac{1}{2}(ex_m^2 + ey_m^2) + \frac{1}{k_y}(1 - \cos e\theta_m) \tag{5.217}$$

Time derivative of equation 5.217 becomes:

$$\dot{V} = e\dot{x}_m ex_m + e\dot{y}_m ey_m + \frac{1}{k_y}\dot{\theta}_m \sin\theta_m \tag{5.218}$$

Substituting equation 5.214 in equation 5.211, the derivative state vector error becomes as follows:

$$\begin{bmatrix} e\dot{x}_m \\ e\dot{y}_m \\ e\dot{\theta}_m \end{bmatrix} = \begin{bmatrix} (w_r + v_r(k_y ey_m + k_\theta \sin e\theta_m))ey_m - k_x ex_m \\ -(w_r + k_y v_r ey_m + k_\theta v_r \sin e\theta_m)ex_m + v_r \sin e\theta_m \\ -v_r(k_y ey_m + k_\theta \sin e\theta_m) \end{bmatrix} \tag{5.219}$$

Then

$$\begin{aligned} \dot{V} &= ((w_r + v_r(k_y ey_m + k_\theta \sin e\theta_m))ey_m - k_x ex_m)ex_m \\ &+ (-(w_r + k_y v_r ey_m + k_\theta v_r \sin e\theta_m)ex_m + v_r \sin e\theta_m)ey_m \\ &+ \frac{1}{k_y}\sin e\theta_m(-v_r(k_y ey_m + k_\theta \sin e\theta_m)) \end{aligned} \tag{5.220}$$

$$\dot{V} = -k_x ex_m^2 - v_r \frac{k_\theta}{k_y}\sin^2 e\theta_m \le 0 \tag{5.221}$$

Clearly, $V \ge 0.$ If $q_{em} = 0, V = 0.$ If $q_{em} \ne 0, V > 0.$ and $\dot{V} \le 0.$ If $q_{em} = 0, \dot{V} = 0.$ If $q_{em} \ne 0, \dot{V} < 0.$ Then, V becomes a Lyapunov function.

So the closed loop system is globally asymptotically stable with three weighting parameters of error variables $(k_x, k_y, k_\theta) > 0$. The controller gains (k_x, k_y, k_θ) are determined by two stages as follows:

$$k_x = (k_x^* + \Delta k_x) > 0 \tag{5.222}$$

$$k_y = (k_y^* + \Delta k_y) > 0 \tag{5.223}$$

$$k_\theta = (k_\theta^* + \Delta k_\theta) > 0 \tag{5.224}$$

where $(k_x^*, k_y^*, k_\theta^*)$ are determined by comparing the actual and the desired characteristic polynomial equations, while $(\Delta k_x, \Delta k_y, \Delta k_\theta)$ are determined by using the gradient-descent delta rule method in order to adjust the parameters of the nonlinear feedback acceleration controller.

The desired characteristic polynomial takes the following form:

$$(Z + z_1)(Z + z_2)(Z + z_3) = 0 \tag{5.225}$$

where

$$z_1 = -e^{-2\xi w_n T_s} \tag{5.226}$$

$$z_{2,3} = -e^{-\xi w_n T_s \pm j w_n \sqrt{1-\xi^2} T_s} \tag{5.227}$$

The desired damping coefficient $\xi \in (0,1)$ and the characteristic frequency $w_n >> |w_{rMax}|$ are selected.

where w_{rMax} is the maximum allowed mobile robot angular velocity.

The characteristic polynomial of the closed loop control law after linearization of the equation 5.219 is:

$$\det(ZI - A) = Z^3 + (T_s k_x^* + T_s v_r k_\theta^* - 3)Z^2 + (T_s^2 k_x^* v_r k_\theta^* + T_s^2 v_r^2 k_y^* + T_s^2 w_r^2 - 2T_s v_r k_\theta^* - 2T_s k_x^* + 3)Z$$
$$+ (-T_s k_x^* - T_s v_r k_\theta^* - T_s^2 k_x^* v_r k_\theta^* - T_s^2 w_r^2 - T_s^2 v_r^2 k_y^* + T_s^3 w_r^2 v_r k_\theta^* + T_s^3 k_x v_r^2 k_y^* - 1). \tag{5.228}$$

For more details, see appendix F.

Comparing coefficients at the same power of Z in equation 5.225 and equation 5.228 results in the following:

$$T_s k_x^* - 3 + T_s\, v_r k_\theta^* = z_1 + z_2 + z_3 \tag{5.229}$$

$$T_s^2 k_x^* v_r k_\theta^* + T_s^2 v_r^2 k_y^* + T_s^2 w_r^2 - 2T_s\, v_r k_\theta^* - 2T_s k_x^* + 3 = z_1 z_2 + z_1 z_3 + z_2 z_3 \tag{5.230}$$

$$-T_s k_x^* - T_s v_r k_\theta^* - T_s^2 k_x^* v_r k_\theta^* - T_s^2 w_r^2 - T_s^2 v_r^2 k_y^* + T_s^3 w_r^2 v_r k_\theta^* + T_s^3 k_x v_r^2 k_y^* - 1 = z_1 z_2 z_3 \tag{5.231}$$

Let $k_x^* = k_\theta^*$ to find the solution of equation 5.229, as follows:

$$k_x^* = k_\theta^* = \frac{(z_1 + z_2 + z_3 + 3)}{T_s(1 + v_r)} \tag{5.232}$$

and k_y is determined from equation 5.230 or equation 5.231 as follows:

$$k_y^* = \frac{z_1 z_2 + z_1 z_3 + z_2 z_3 - 3 - T_s^2 w_r^2 + 2(z_1 + z_2 + z_3 + 3) - T_s\, v_r (\frac{z_1 + z_2 + z_3 + 3}{1 + v_r})^2}{T_s^2 v_r^2} \tag{5.233}$$

or

$$k_y^* = \frac{z_1 z_2 z_3 + 4 + z_1 + z_2 + z_3 + v_r (\frac{z_1 + z_2 + z_3 + 3}{1 + v_r})^2 + T_s^2 w_r^2 - T_s^2 w_r^2 v_r (\frac{z_1 + z_2 + z_3 + 3}{1 + v_r})}{T_s^2 v_r^2 (\frac{z_1 + z_2 + z_3 + 3}{1 + v_r}) - T_s^2 v_r^2} \tag{5.234}$$

When v_r is close to zero or T_s is very small sampling time, k_y^* goes to infinity and the stability of the mobile robot system will lose, therefore, $v_r > 0$, as proven in the Lyapunov function, and to avoid small sampling time, k_y^* can be chosen as gain scheduling in [128], as in equation 5.235.

$$k_y^* = \sigma \times |v_r(k)| \tag{5.235}$$

where σ is constant gain.

So the closed loop system is globally asymptotically stable.

The control parameters (k_x, k_y, k_θ) of nonlinear feedback acceleration controllers are adjusted by using the gradient-descent delta rule method in order to apply optimisation predictive algorithm. For one step-ahead when N=1, the update rules for these control parameters are expressed by:

$$k_x(k+1) = k_x^*(k+1) + \Delta k_x(k+1) \tag{5.236}$$

$$k_y(k+1) = k_y^*(k+1) + \Delta k_y(k+1) \tag{5.237}$$

$$k_\theta(k+1) = k_\theta^*(k+1) + \Delta k_\theta(k+1) \tag{5.238}$$

$$\Delta(k_\gamma)(k+1) = -\eta \frac{\partial J}{\partial k_\gamma(k)} \tag{5.239}$$

where γ is x, y, θ for each time.

The use of the minus sign accounts for the gradient-descent in weight-space reflecting the seek of the direction for weight, change that reduces the value of *J* in equation 5.37. The learning rate η is a small value to smooth weights changing of nonlinear feedback acceleration controller's parameters.

By applying the chain rule, the term $\frac{\partial J}{\partial k_x(k)}$ for position x-coordinate error is represented as:

$$\Delta k_x(k+1) = -\eta \frac{\partial J}{\partial k_x(k)} = -\eta \frac{\frac{1}{2} Q \partial((x_r(k+1) - x_m(k+1))^2}{\partial k_x(k)} - \eta \frac{\frac{1}{2} R \partial(\tau_1(k))^2}{\partial k_x(k)} \tag{5.240}$$

$$\Delta k_x(k+1) = \eta Q e x_m(k+1) \frac{\partial x_m(k+1)}{\partial k_x(k)} - \eta R \frac{\partial \tau_1(k)}{\partial k_x(k)} \tag{5.241}$$

$$\frac{\partial x_m(k+1)}{\partial k_x(k)} = \frac{\partial x_m(k+1)}{\partial o_1} \frac{\partial o_1}{\partial net_1} \frac{\partial net_1}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_R(k)} \frac{\partial \tau_R(k)}{\partial \tau_1(k)} \frac{\partial \tau_1(k)}{\partial \dot{v}_{c1}(k)} \frac{\partial \dot{v}_{c1}(k)}{\partial v_{c1}(k)} \frac{\partial v_{c1}(k)}{\partial k_x(k)} \tag{5.242}$$

$$\frac{\partial x_m(k+1)}{\partial k_x(k)} = \frac{\partial net_1}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_R(k)} \frac{\partial \tau_R(k)}{\partial \tau_1(k)} \frac{\partial \tau_1(k)}{\partial \dot{v}_{c1}(k)} \frac{\partial \dot{v}_{c1}(k)}{\partial v_{c1}(k)} \frac{\partial v_{c1}(k)}{\partial k_x(k)} \tag{5.243}$$

$$\tau_R(k) = \tau_{ref\,1}(k) + \tau_1(k) \tag{5.244}$$

$$\frac{\partial x_m(k+1)}{\partial k_x(k)} = \frac{\partial net_1}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_R(k)} \frac{\partial \tau_1(k)}{\partial \dot{v}_{c1}(k)} \frac{\partial \dot{v}_{c1}(k)}{\partial v_{c1}(k)} \frac{\partial v_{c1}(k)}{\partial k_x(k)} \tag{5.245}$$

Figure 5.5 shows that $\tau_R(k)$ is linked to the exciting nodes, VH_{j1}

$$\frac{\partial x_m(k+1)}{\partial k_x(k)} = Mex_m(k) \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} \tag{5.246}$$

$$\frac{\partial \tau_1(k)}{\partial k_x(k)} = \frac{\partial \tau_1(k)}{\partial \dot{v}_{c1}(k)} \frac{\partial \dot{v}_{c1}(k)}{\partial v_{c1}(k)} \frac{\partial v_{c1}(k)}{\partial k_x(k)} = Mex_m(k) \tag{5.247}$$

$$\Delta k_x(k+1) = \eta Q e x_m(k+1) Mex_m(k) \sum_{j=1}^{nh} W_{1j} f(net_j)' VH_{j1} - \eta R Mex_m(k) \tag{5.248}$$

By applying the chain rule, the term $\frac{\partial J}{\partial k_y(k)}$ for position y-coordinate error is represented as:

$$\Delta k_y(k+1) = -\eta \frac{\partial J}{\partial k_y(k)} = -\eta \frac{\frac{1}{2} Q \partial((y_r(k+1) - y_m(k+1))^2}{\partial k_y(k)} - \eta \frac{\frac{1}{2} R \partial(\tau_2(k))^2}{\partial k_y(k)} \tag{5.249}$$

$$\Delta k_y(k+1) = \eta Q e y_m(k+1) \frac{\partial y_m(k+1)}{\partial k_y(k)} - \eta R \frac{\partial \tau_2(k)}{\partial k_y(k)} \tag{5.250}$$

$$\frac{\partial y_m(k+1)}{\partial k_y(k)} = \frac{\partial y_m(k+1)}{\partial o_2} \frac{\partial o_2}{\partial net_2} \frac{\partial net_2}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_L(k)}{\partial \tau_2(k)} \frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)} \frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)} \frac{\partial v_{c2}(k)}{\partial k_y(k)} \tag{5.251}$$

$$\frac{\partial y_m(k+1)}{\partial k_y(k)} = \frac{\partial net_2}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_L(k)}{\partial \tau_2(k)} \frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)} \frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)} \frac{\partial v_{c2}(k)}{\partial k_y(k)} \tag{5.252}$$

$$\tau_L(k) = \tau_{ref\,2}(k) + \tau_2(k) \tag{5.253}$$

$$\frac{\partial y_m(k+1)}{\partial k_y(k)} = \frac{\partial net_2}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)} \frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)} \frac{\partial v_{c2}(k)}{\partial k_y(k)} \tag{5.254}$$

Figure 5.5 shows that $\tau_L(k)$ is linked to the exciting nodes, VH_{j2}

$$\frac{\partial y_m(k+1)}{\partial k_y(k)} = Iv_r(k) e y_m(k) \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j2} \tag{5.255}$$

$$\frac{\partial \tau_2(k)}{\partial k_y(k)} = \frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)} \frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)} \frac{\partial v_{c2}(k)}{\partial k_y(k)} = Iv_r(k) e y_m(k) \tag{5.256}$$

$$\Delta k_y(k+1) = \eta Q e y_m(k+1) Iv_r(k) e y_m(k) \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j2} - \eta R Iv_r(k) e y_m(k) \tag{5.257}$$

By applying the chain rule, the term $\frac{\partial J}{\partial k_\theta(k)}$ for orientation error is represented as:

$$\Delta k_\theta(k+1) = -\eta \frac{\partial J}{\partial k_\theta(k)} = -\eta \frac{\frac{1}{2} Q \partial((\theta_r(k+1) - \theta_m(k+1))^2}{\partial k_\theta(k)} - \eta \frac{\frac{1}{2} R \partial(\tau_2(k))^2}{\partial k_\theta(k)} \tag{5.258}$$

$$\Delta k_\theta(k+1) = \eta Q e \theta_m(k+1) \frac{\partial y_\theta(k+1)}{\partial k_\theta(k)} - \eta R \frac{\partial \tau_2(k)}{\partial k_\theta(k)} \tag{5.259}$$

$$\frac{\partial \theta_m(k+1)}{\partial k_\theta(k)} = \frac{\partial \theta_m(k+1)}{\partial o_3} \frac{\partial o_3}{\partial net_3} \frac{\partial net_3}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_L(k)}{\partial \tau_2(k)} \frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)} \frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)} \frac{\partial v_{c2}(k)}{\partial k_\theta(k)} \tag{5.260}$$

$$\frac{\partial \theta_m(k+1)}{\partial k_\theta(k)} = \frac{\partial net_3}{\partial h_j} \frac{\partial h_j}{\partial net_j} \frac{\partial net_j}{\partial \tau_L(k)} \frac{\partial \tau_L(k)}{\partial \tau_2(k)} \frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)} \frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)} \frac{\partial v_{c2}(k)}{\partial k_\theta(k)} \tag{5.261}$$

$$\tau_L(k) = \tau_{ref\,2}(k) + \tau_2(k) \tag{5.262}$$

$$\frac{\partial \theta_m(k+1)}{\partial k_\theta(k)} = \frac{\partial net_3}{\partial h_j}\frac{\partial h_j}{\partial net_j}\frac{\partial net_j}{\partial \tau_L(k)}\frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)}\frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)}\frac{\partial v_{c2}(k)}{\partial k_\theta(k)} \tag{5.263}$$

Figure 5.5 shows that $\tau_L(k)$ is linked to the exciting nodes, VH_{j2}

$$\frac{\partial \theta_m(k+1)}{\partial k_\theta(k)} = Iv_r(k)\sin e\theta_m(k)\sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} \tag{5.264}$$

$$\frac{\partial \tau_2(k)}{\partial k_\theta(k)}\frac{\partial \tau_2(k)}{\partial \dot{v}_{c2}(k)}\frac{\partial \dot{v}_{c2}(k)}{\partial v_{c2}(k)}\frac{\partial v_{c2}(k)}{\partial k_\theta(k)} = Iv_r(k)\sin e\theta_m(k) \tag{5.265}$$

$$\Delta k_\theta(k+1) = \eta Qe\theta_m(k+1)Iv_r(k)\sin e\theta_m(k)\sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} - \eta RIv_r(k)\sin e\theta_m(k) \tag{5.266}$$

After identifying these control parameters, the feedback control action $\tau_1(k+1)\,\&\,\tau_2(k+1)$ for the next sample can be calculated.

The total control action of the nonlinear controller became as:

$$\tau_R(k+1) = \tau_{ref1}(k+1) + \tau_1(k+1) \tag{5.267}$$

$$\tau_L(k+1) = \tau_{ref\,2}(k+1) + \tau_2(k+1) \tag{5.268}$$

This is calculated at each sample time k and applied to mobile robot system and the position and orientation identifier model. Then this procedure is continually applied at the next sampling time (k+1) until the error between the desired trajectory and the identifier model output becomes lower than a specified value.

The weights of the position and orientation neural network identifier and the weights of the feedforward neural controller are updated after each sampling time in order to minimise the error between $q = [x, y, \theta]^T$ & $q_m = [x_m, y_m, \theta_m]^T$ using back propagation learning algorithm.

For N steps estimation of the nonlinear inverse dynamic feedback controller actions $\tau_1(k)\,\&\,\tau_2(k)$ the techniques of generalized predictive control theory are used. The N steps estimation of $\tau_1(k)\,\&\,\tau_2(k)$ is calculated for each sample. The position and orientation in the identifier model, shown in Figure 5.5, represent the kinematics and dynamics model of the mobile robot system and are controlled asymptotically.

Therefore, it can be used to predict future values of the model outputs for the next N steps and can be used to find the optimal value of $\tau_1(k)\,\&\,\tau_2(k)$ using an optimisation predictive algorithm.

For this purpose, let N be a pre-specified positive integer that is denoted such that the future values of the set point are:

$$X_{rt,N} = [x_r(t+1), x_r(t+2), x_r(t+3), ..., x_r(t+N)] \tag{5.269}$$

$$Y_{rt,N} = [y_r(t+1), y_r(t+2), y_r(t+3), ..., y_r(t+N)] \tag{5.270}$$

$$\theta_{rt,N} = [\theta_r(t+1), \theta_r(t+2), \theta_r(t+3), ..., \theta_r(t+N)] \tag{5.271}$$

$$v_{rt,N} = [v_r(t+1), v_r(t+2), v_r(t+3), ..., v_r(t+N)] \tag{5.272}$$

$$w_{rt,N} = [w_r(t+1), w_r(t+2), w_r(t+3), ..., w_r(t+N)] \tag{5.273}$$

where t represents the time instant. Then the predicted outputs of the robot model used the neural identifier, shown in Figure 5.5, are:

$$X_{mt,N} = [x_m(t+1), x_m(t+2), x_m(t+3), ..., x_m(t+N)] \tag{5.274}$$

$$Y_{mt,N} = [y_m(t+1), y_m(t+2), y_m(t+3), ..., y_m(t+N)] \tag{5.275}$$

$$\theta_{mt,N} = [\theta_m(t+1), \theta_m(t+2), \theta_m(t+3), ..., \theta_m(t+N)] \tag{5.276}$$

Then define the following error vector:

$$EX_{m,t,N} = [ex_m(t+1), ex_m(t+2), ex_m(t+3) ..., ex_m(t+N] \tag{5.277}$$

$$EY_{m,t,N} = [ey_m(t+1), ey_m(t+2), ey_m(t+3) ..., ey_m(t+N] \tag{5.278}$$

$$E\theta_{m,t,N} = [e\theta_m(t+1), e\theta_m(t+2), e\theta_m(t+3) ..., e\theta_m(t+N] \tag{5.279}$$

Two-feedback control signals can be determined by:

$$\tau'_{1_{t,N}} = [\tau'_1(t), \tau'_1(t+1), \tau'_1(t+2), ..., \tau'_1(t+N-1)] \tag{5.280}$$

$$\tau'_{2_{t,N}} = [\tau'_2(t), \tau'_2(t+1), \tau'_2(t+2), ..., \tau'_2(t+N-1)] \tag{5.281}$$

Assuming the following objective function:

$$J_1 = \frac{1}{2} Q[(EX_{m,t,N} EX_{m,t,N}{}^T) + (EY_{m,t,N} EY_{m,t,N}{}^T) + (E\theta_{m,t,N} E\theta_{m,t,N}{}^T)]$$

$$+ \frac{1}{2} R[(\tau'_{1t,N} \tau'_{1t,N}{}^T) + (\tau'_{2t,N} \tau'_{2t,N}{}^T)] \quad (5.282)$$

Then our purpose is to adjust the control parameters (k_x, k_y, k_θ) of nonlinear feedback acceleration controllers such that J_1 is minimised using the gradient descent rule.

$$\Delta(k_\gamma)(k+1) = -\eta \frac{\partial J_1}{\partial k_\gamma(k)} \quad (5.283)$$

where γ is x, y, θ for each time.

$$k'_{x,t,N}{}^{K+1} = k'^{*}_{x,t,N}{}^{K} + \Delta k'_{x,t,N}{}^{K} \quad (5.284)$$

$$k'_{y,t,N}{}^{K+1} = k'^{*}_{y,t,N}{}^{K} + \Delta k'_{y,t,N}{}^{K} \quad (5.285)$$

$$k'_{\theta,t,N}{}^{K+1} = k'^{*}_{\theta,t,N}{}^{K} + \Delta k'_{\theta,t,N}{}^{K} \quad (5.286)$$

$$\Delta k'_{xt,N}{}^{K} = -\eta \frac{\partial J_1}{\partial k'_{xt,N}{}^{K}} = [\Delta k'_x(t), \Delta k'_x(t+1), \Delta k'_x(t+2), ... \Delta k'_x(t+N-1)] \quad (5.287)$$

$$-\eta \frac{\partial J_1}{\partial k'_{xt,N}{}^{K}} = \eta Q E X_{m,t,N} \frac{\partial X_{mt,N}}{\partial k'_{xt,N}{}^{K}} - \eta R \frac{\partial \tau'_{1t,N}{}^{K}}{\partial k'_{xt,N}{}^{K}} \quad (5.288)$$

$$\frac{\partial X_{mt,N}}{\partial k'_{xt,N}{}^{K}} = \begin{bmatrix} \frac{\partial x_m(t+1)}{\partial k'_x(t)} & \frac{\partial x_m(t+2)}{\partial k'_x(t+1)} & \frac{\partial x_m(t+3)}{\partial k'_x(t+2)} & \cdots & \frac{\partial x_m(t+N)}{\partial k'_x(t+N-1)} \end{bmatrix} \quad (5.289)$$

where:

$$\frac{\partial x_m(t+1)}{\partial k'_x(t)} = M e x_m(t) \sum_{j=1}^{nh} W_{1j} f(net_j)' V H_{j1} \quad (5.290)$$

and

$$\frac{\partial \tau'_{1t,N}{}^{K}}{\partial k'_{xt,N}{}^{K}} = \begin{bmatrix} \frac{\partial \tau'_1(t)}{\partial k'_x(t)} & \frac{\partial \tau'_1(t+1)}{\partial k'_x(t+1)} & \frac{\partial \tau'_1(t+2)}{\partial k'_x(t+2)} & \cdots & \frac{\partial \tau'_1(t+N-1)}{\partial k'_x(t+N-1)} \end{bmatrix} \quad (5.291)$$

where:

$$\frac{\partial \tau_1(t)}{\partial k'_x(t)} = M e x_m(t) \quad (5.292)$$

$$\Delta k'_{y_{t,N}}{}^{K} = -\eta \frac{\partial J_1}{\partial k'_{y_{t,N}}{}^{K}} = [\Delta k'_y(t), \Delta k'_y(t+1), \Delta k'_y(t+2), ... \Delta k'_y(t+N-1)] \quad (5.293)$$

$$-\eta \frac{\partial J_1}{\partial k'_{y_{t,N}}{}^{K}} = \eta Q E Y_{m,t,N} \frac{\partial Y_{mt,N}}{\partial k'_{y_{t,N}}{}^{K}} - \eta R \frac{\partial \tau'_{2t,N}{}^{K}}{\partial k'_{y_{t,N}}{}^{K}} \quad (5.294)$$

$$\frac{\partial Y_{mt,N}}{\partial k_{y_{t,N}}^{'\ K}} = \left[\frac{\partial y_m(t+1)}{\partial k_y^{'}(t)} \quad \frac{\partial y_m(t+2)}{\partial k_y^{'}(t+1)} \quad \frac{\partial y_m(t+3)}{\partial k_y^{'}(t+2)} \quad \cdots \quad \frac{\partial y_m(t+N)}{\partial k_y^{'}(t+N-1)} \right] \tag{5.295}$$

where:

$$\frac{\partial y_m(t+1)}{\partial k_y^{'}(t)} = Iv_r(t) ey_m(t) \sum_{j=1}^{nh} W_{2j} f(net_j)' VH_{j2} \tag{5.296}$$

and

$$\frac{\partial \tau_{2t,N}'^{\ K}}{\partial k_{y_{t,N}}^{'\ K}} = \left[\frac{\partial \tau_2'(t)}{\partial k_y^{'}(t)} \quad \frac{\partial \tau_2'(t+1)}{\partial k_y^{'}(t+1)} \quad \frac{\partial \tau_2'(t+2)}{\partial k_y^{'}(t+2)} \quad \cdots \quad \frac{\partial \tau_2'(t+N-1)}{\partial k_y^{'}(t+N-1)} \right] \tag{5.297}$$

where:

$$\frac{\partial \tau_1(t)}{\partial k_y^{'}(t)} = Iv_r(t) ey_m(t) \tag{5.298}$$

$$\Delta k_{\theta_{t,N}}^{'\ K} = -\eta \frac{\partial J_1}{\partial k_{\theta_{t,N}}^{'\ K}} = [\Delta k_\theta^{'}(t), \Delta k_\theta^{'}(t+1), \Delta k_\theta^{'}(t+2), ... \Delta k_\theta^{'}(t+N-1)] \tag{5.299}$$

$$-\eta \frac{\partial J_1}{\partial k_{\theta t,N}^{'\ K}} = \eta QE\theta_{m,t,N} \frac{\partial \theta_{mt,N}}{\partial k_{\theta t,N}^{'\ K}} - \eta R \frac{\partial \tau_{2t,N}'^{\ K}}{\partial k_{\theta t,N}^{'\ K}} \tag{5.300}$$

$$\frac{\partial \theta_{mt,N}}{\partial k_{\theta_{t,N}}^{'\ K}} = \left[\frac{\partial \theta_m(t+1)}{\partial k_\theta^{'}(t)} \quad \frac{\partial \theta_m(t+2)}{\partial k_\theta^{'}(t+1)} \quad \frac{\partial \theta_m(t+3)}{\partial k_\theta^{'}(t+2)} \quad \cdots \quad \frac{\partial \theta_m(t+N)}{\partial k_\theta^{'}(t+N-1)} \right] \tag{5.301}$$

where:

$$\frac{\partial \theta_m(t+1)}{\partial k_\theta^{'}(t)} = Iv_r(t) \sin e\theta_m(t) \sum_{j=1}^{nh} W_{3j} f(net_j)' VH_{j2} \tag{5.302}$$

and

$$\frac{\partial \tau_{2t,N}'^{\ K}}{\partial k_{\theta_{t,N}}^{'\ K}} = \left[\frac{\partial \tau_2'(t)}{\partial k_\theta^{'}(t)} \quad \frac{\partial \tau_2'(t+1)}{\partial k_\theta^{'}(t+1)} \quad \frac{\partial \tau_2'(t+2)}{\partial k_\theta^{'}(t+2)} \quad \cdots \quad \frac{\partial \tau_2'(t+N-1)}{\partial k_\theta^{'}(t+N-1)} \right] \tag{5.302}$$

where:

$$\frac{\partial \tau_2(t)}{\partial k_\theta^{'}(t)} = Iv_r(t) \sin e\theta_m(t) \tag{5.303}$$

After completing the procedure from n=1 to N the new control actions for the next sample will be:

$$\tau_R(k+1) = \tau_{ref1}(k+1) + \tau_1'^K(t+N) \tag{5.304}$$

$$\tau_L(k+1) = \tau_{ref2}(k+1) + \tau_2'^K(t+N) \tag{5.305}$$

where $\tau_1''^{k}(t+N)\,\&\,\tau_2'^{k}(t+N)$ are the final values of the feedback-controlling signals calculated by the optimisation algorithm. This is calculated at each sample time k so that $\tau_R(k+1)\,\&\,\tau_L(k+1)$ are torque control actions of the right and the left wheels respectively. These actions will be applied to the mobile robot system and the position and orientation identifier model at the next sampling time. The application of this procedure will continue at the next sampling time (k+1) until the error between the desired input and the actual output becomes lower than a pre-specified value.

5.4. Simulation Results

In this section, several desired trajectories are tracked from nonholonomic wheeled mobile robot in order to clarify the features of the adaptive neural predictive controller explained in this chapter. The performance of the proposed adaptive neural control with three control methodologies will be compared with them in terms of minimum tracking error and in generating an optimal torque control action and the capability of tracking any trajectories with continuous and non-continuous gradients, despite the presence of bounded external disturbances. This comparison has to be made under identical conditions in order to specify the differences and similarities (and consequently the advantages and disadvantages) of the proposed control methodology. The simulation results in this chapter are implemented using Matlab program.

5.4.1. Case Study 1

The desired lemniscates trajectory, which has explicitly continuous gradient with rotation radius changes, can be described by the following equations:

$$x_r(t) = 0.75 + 0.75 \times \sin(\frac{2\pi t}{50}) \tag{5.306}$$

$$y_r(t) = \sin(\frac{4\pi t}{50}) \tag{5.307}$$

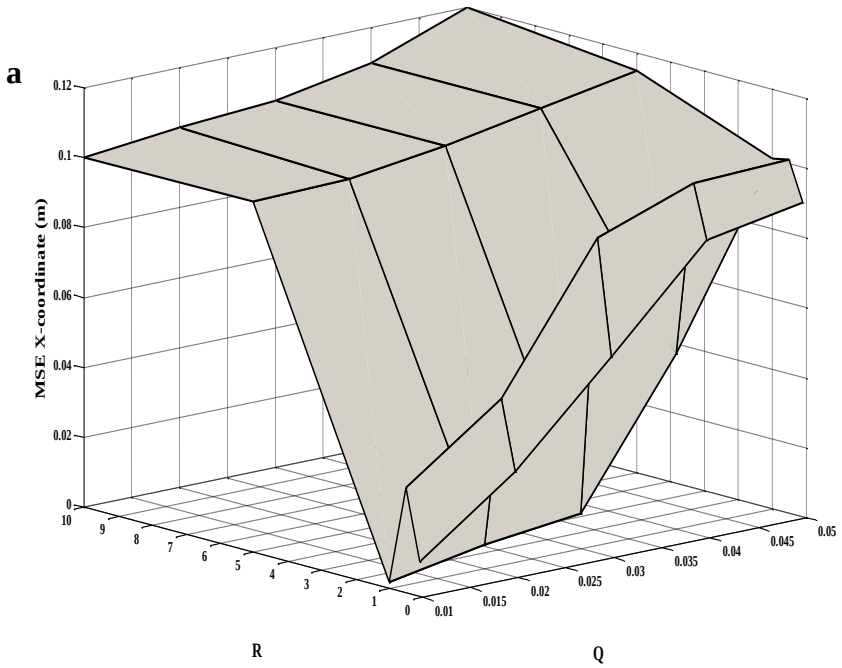
$$\theta_r(t) = 2\tan^{-1}(\frac{\Delta y_r(t)}{\sqrt{(\Delta x_r(t))^2 - (\Delta y_r(t))^2} + \Delta x_r(t)}) \tag{5.308}$$

The structure of the feedforward neural controller is multi-layer perceptron neural network (8-16-2), as shown in Figure 5.3, where the maximum number of the nodes in the hidden layer can be expressed as 2n+1, where n is number of nodes in the input layer [39,

116] (see appendix G). The trajectory has been learned by the feedforward neural controller with off-line and on-line adaptation stages using back propagation algorithm, as shown in Figure 5.4, to find the suitable reference torque control action at steady state. Finally the case of tracking a lemniscates trajectory for robot model, as shown in Figure 5.2, is demonstrated with optimisation algorithm for N-step-ahead prediction. For simulation purposes, the desired trajectory is chosen as described in the equations 5.306, 5.307 and 5.308. The robot model starts from the initial posture $q(0) = [0.75, -0.25, \pi/2]$ as its initial conditions.

A disturbance term $\tau d = 0.01\sin(2t)$ [56, 59, 62] is added to the robot system as unmodelled kinematics and dynamics disturbances in order to prove the adaptation and robustness ability of the proposed controller. The feedback neural controller seems to require more tuning effort of its two parameters (Q and R). Q is the sensitivity weighting matrix to the corresponding error between the desired trajectory and identifier trajectory, while the weighting matrix R defines the energy of the input torque signals of right and left wheels.

Investigating the feedback control performance of the neural predictive controller can easily be obtained by changing the ratio of the weighting matrices (Q and R) in order to reduce the tracking error in x-coordinate, y-coordinate and θ-orientation using mean square error method, as shown in Figures 5.8a, 5.8b and 5.8c respectively.



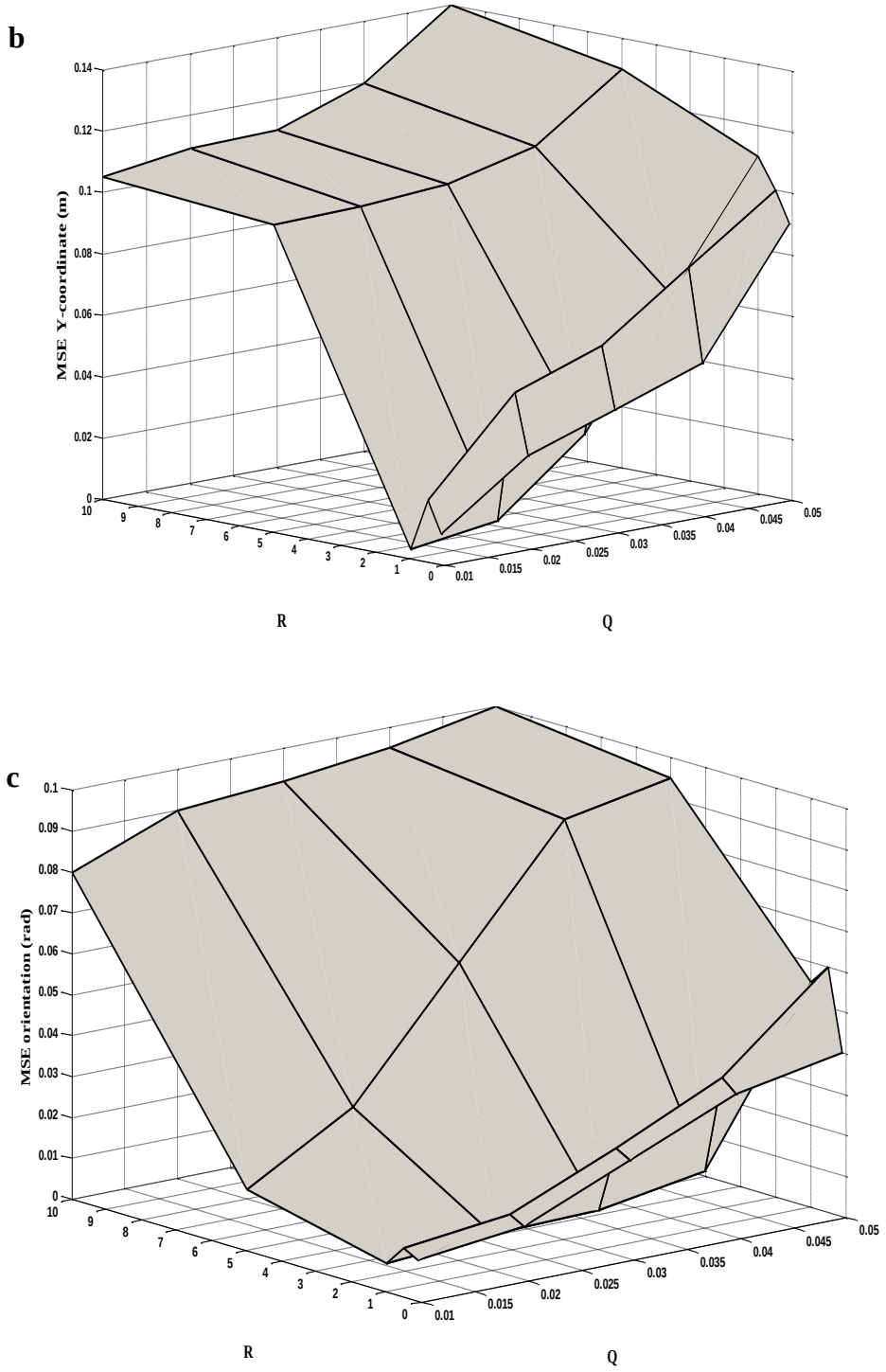


Figure 5.8: Changing (Q and R) parameters then calculating the mean square error: (a) in x-coordinate; (b) y-coordinate; (c) in orientation

This also gives the designer the possibility of obtaining more optimised control action depending on MSE of the position and orientation, which is more difficult to obtain in other controllers.

To investigate and demonstrate the effects of (Q and R) parameters on the performance of the neural predictive controller and the tracking error of the mobile robot, Q and R have taken 0.05 and 10 respectively and the results of control action and the trajectory tracking of the mobile robot are shown in Figures 5.9 and 5.10 respectively.

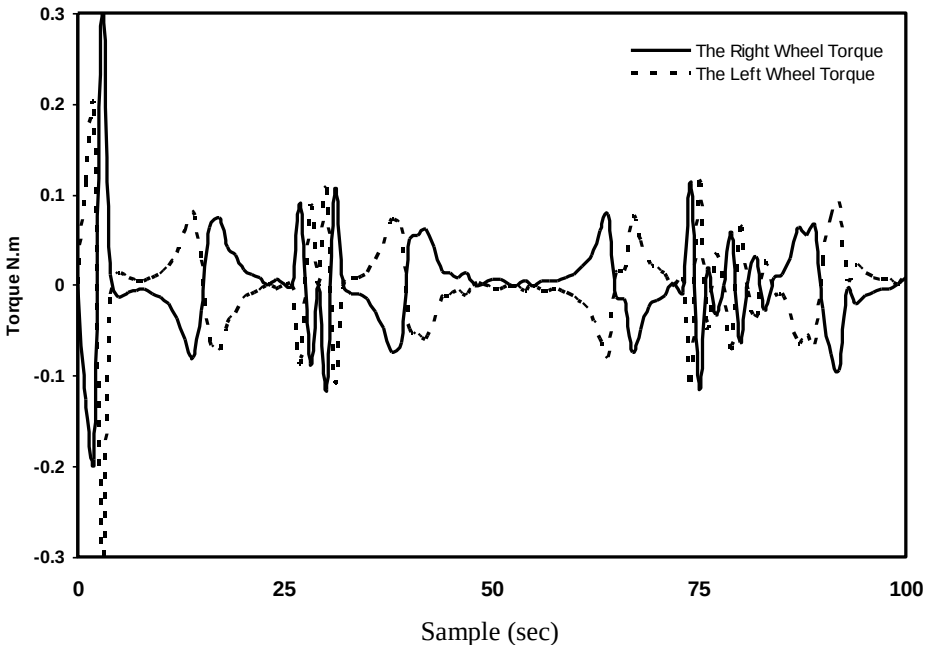


Figure 5.9: The right and left wheel torque control signal when the feedback parameters Q=0.05 and R=10.

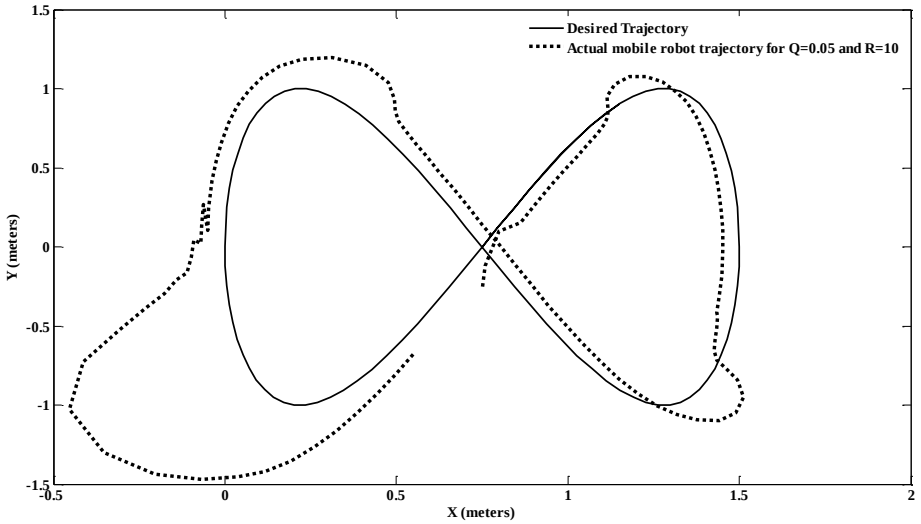


Figure 5.10: Actual trajectory of mobile robot and desired trajectory for Q=0.05 and R=10.

Therefore, the best value of Q parameter is equal to 0.01 and best value of R parameter is equal to 1 for obtaining more optimised control action, as shown in Figure 5.8.

After picking the best values of Q and R for N is equal to one-step-ahead, it becomes necessary to choose the best count for N step-ahead prediction. This is accomplished by carrying out the simulation of the desired lemniscates trajectory of the mobile robot with optimisation algorithm for different Ns (1 to 10), then calculating the position and orientation mean square error for each case in order to select the best count N for smallest position and orientation MSE.

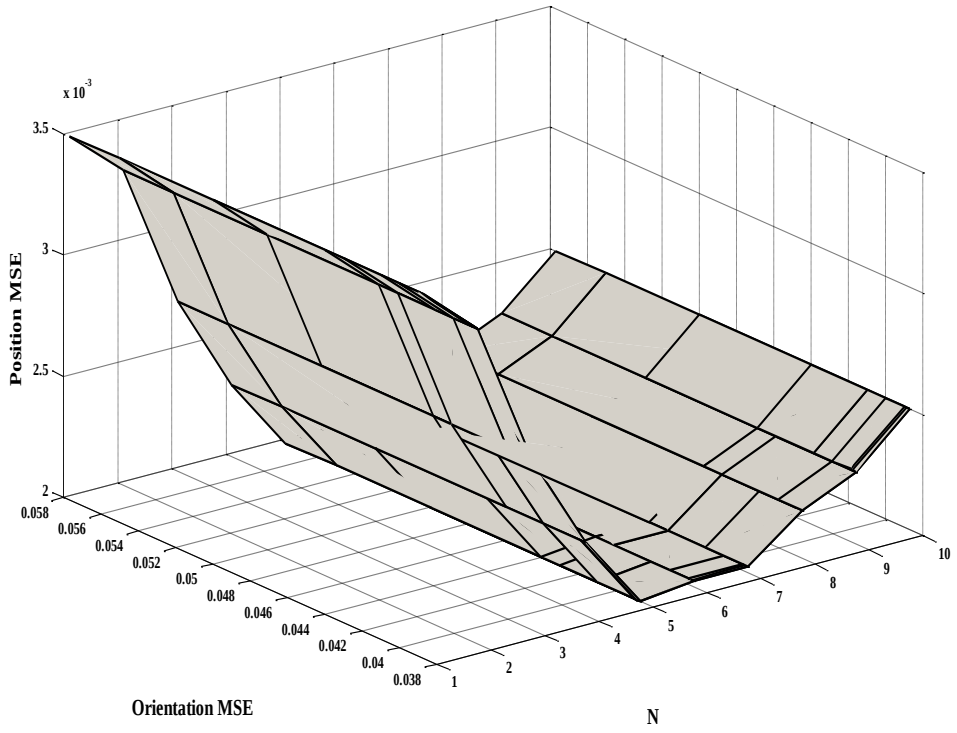


Figure 5.11: The MSE of position and orientation with N parameters.

Figure 5.11 shows that the best count for N step-ahead is equal to 5, although the position and orientation mean square error are the same in case N =6 and N=7; however; it is necessary to take the execution time of the simulation when N=6 or N=7 as it takes a long time to calculate the optimal control action, so the best count for N step-ahead is equal to 5.

The robot trajectory tracking obtained by the proposed adaptive neural predictive controller is shown in Figures 5.12a and 5.12b. These Figures demonstrate well position and orientation tracking performance for the five steps-ahead predictions in comparison with results of one step-ahead.

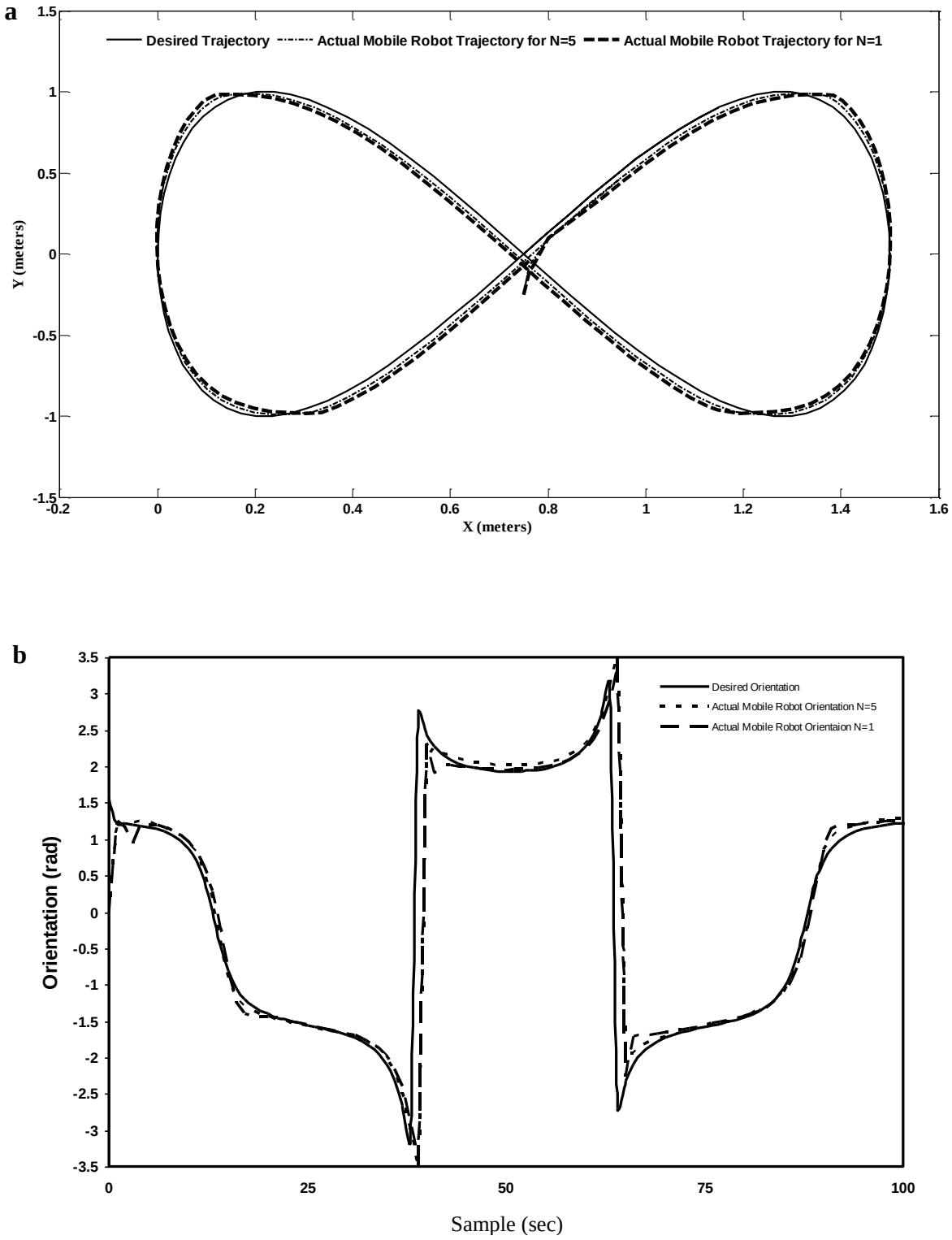


Figure 5.12: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired lemniscates trajectory; (b) actual orientation of mobile robot and desired orientation.

The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small values of the control input torques for right and left wheels with small sharp spikes, as shown in Figures 5.13a and 5.13b.

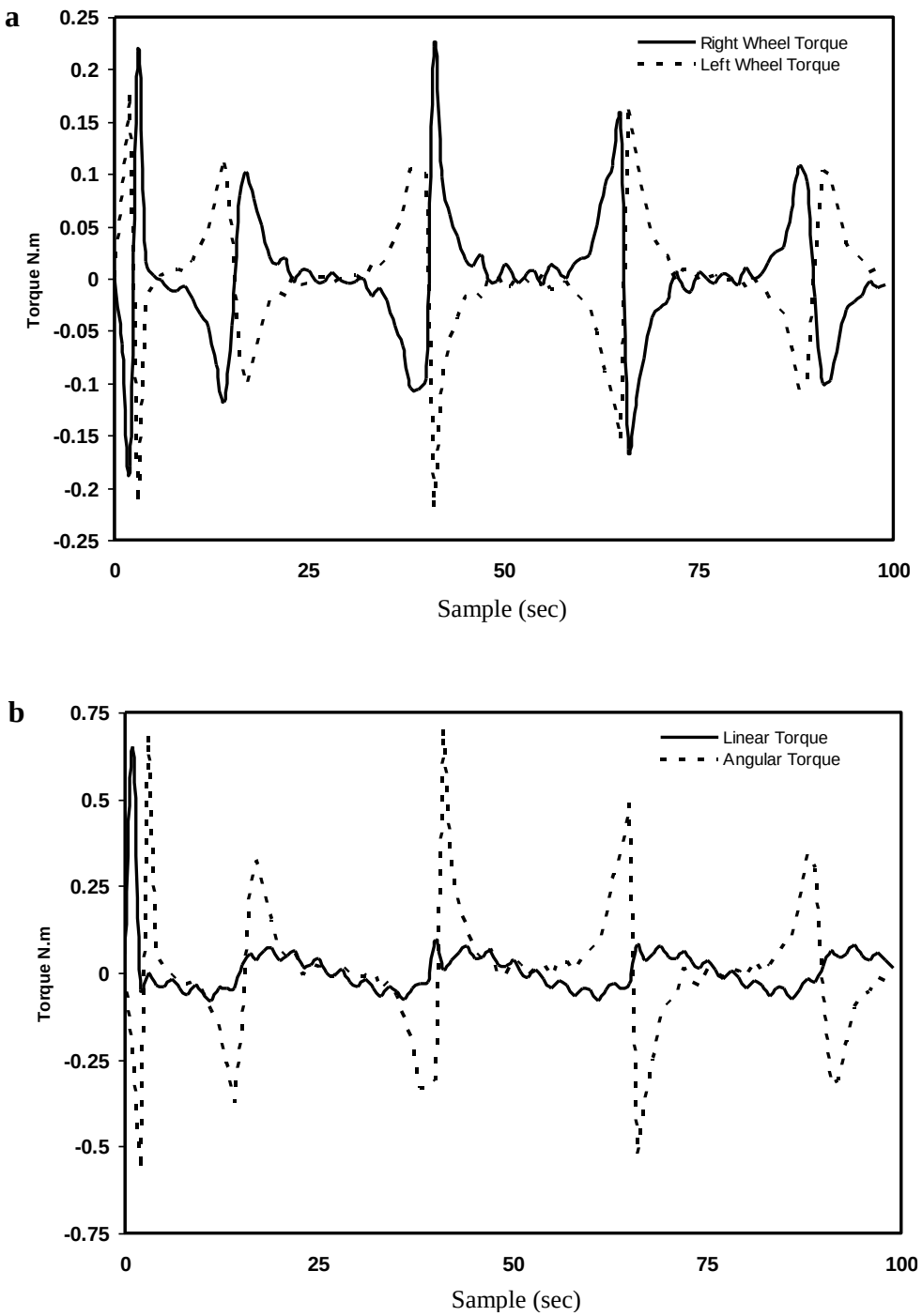


Figure 5.13: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the right wheel and the left wheel were smooth values without sharp spikes, as shown in Figure 5.14a. The mean of linear velocity of the mobile robot is equal to 0.135m/sec, and maximum angular velocity is equal to ± 0.75rad/sec, as shown in Figure 5.14b.

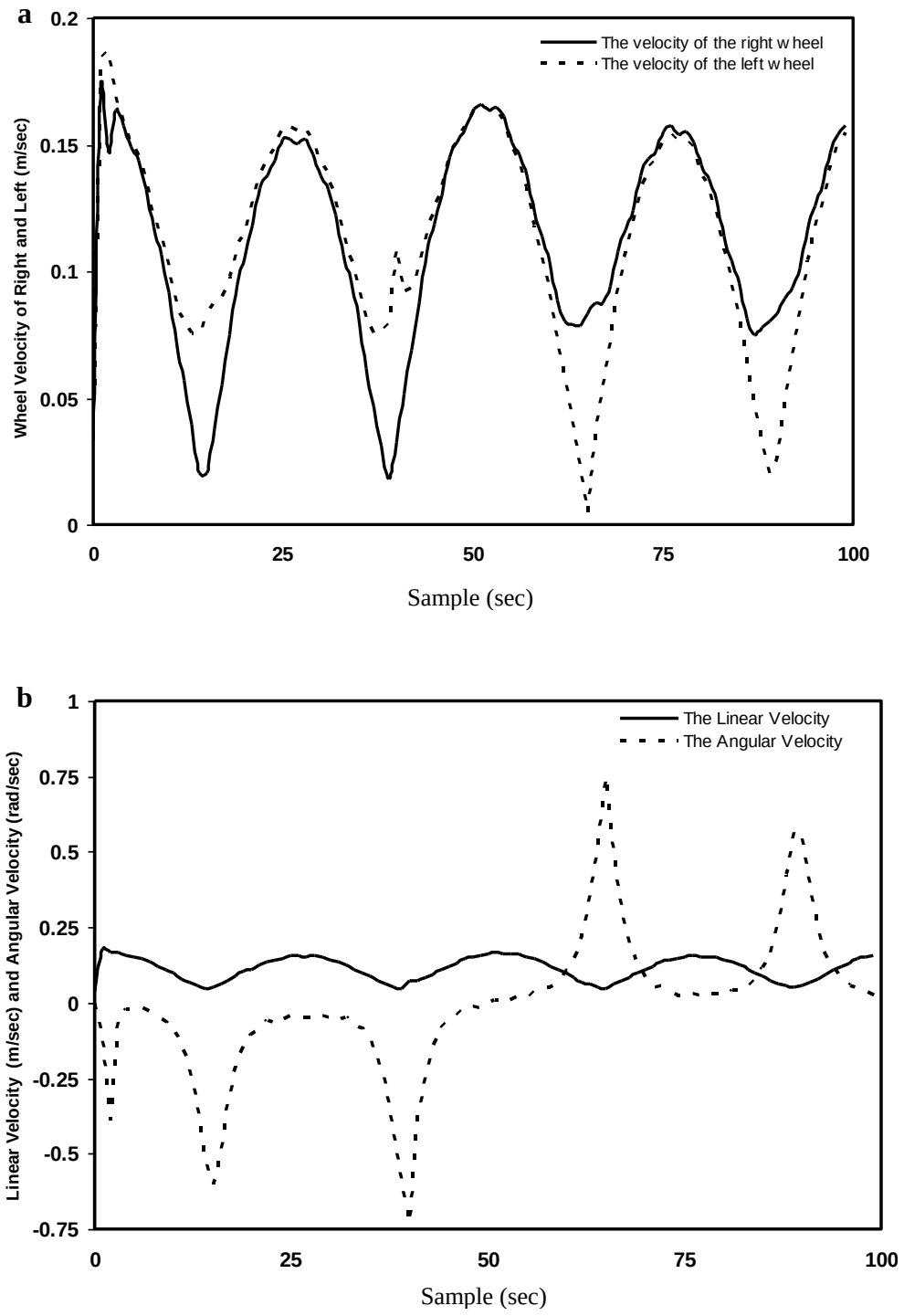


Figure 5.14: Velocity action for N=5: (a) the right and left wheel velocity; (b) the linear and angular velocity.

The effectiveness of the proposed first methodology of the adaptive neural predictive controller with predictive optimisation algorithm is clear by showing the convergence of the position and orientation trajectory error for the robot model motion, as shown in Figures 5.15a , 5.15b and 5.16 respectively for N=1 and 5 steps-ahead-predictive.

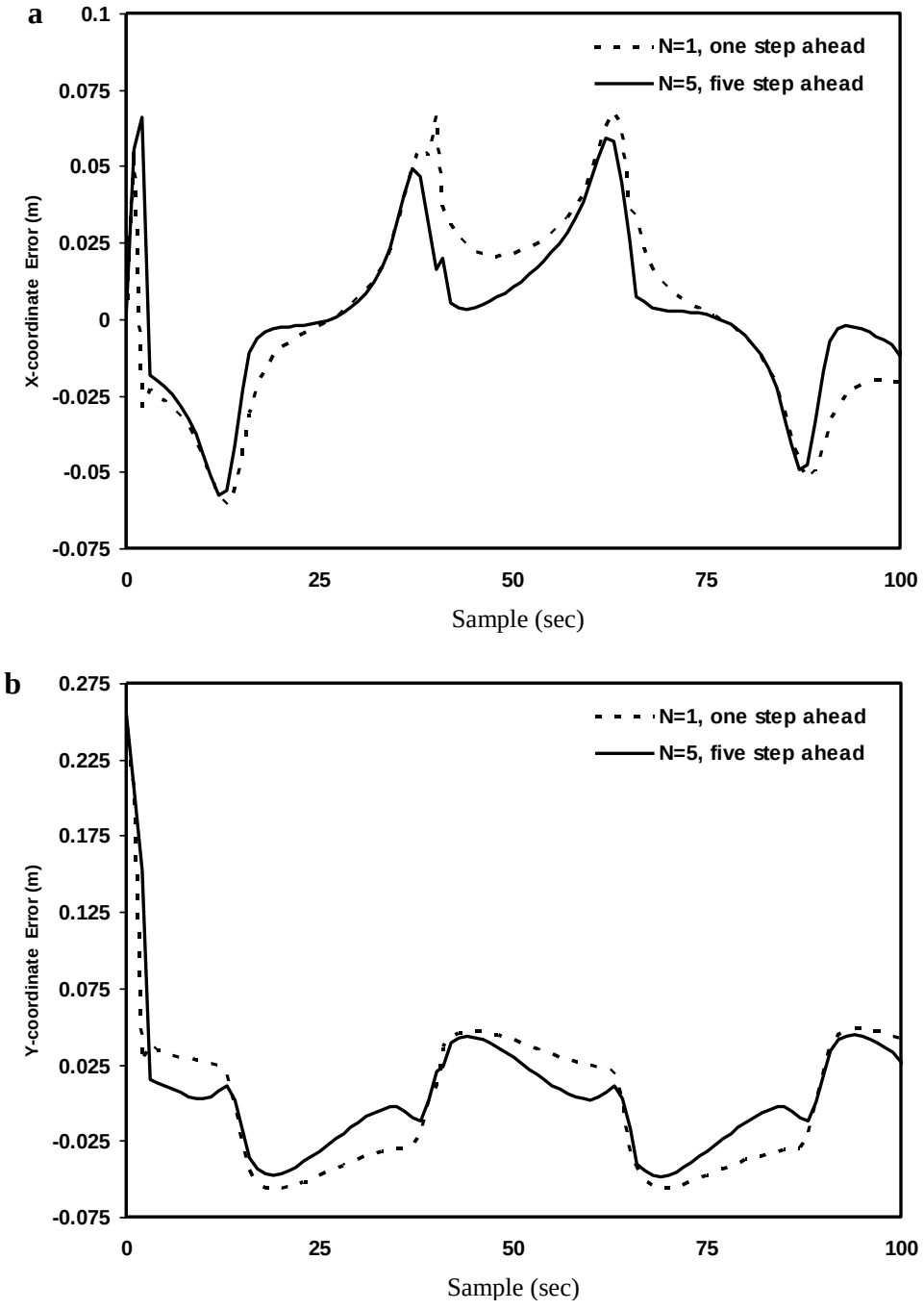


Figure 5.15: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

The tracking error in the X-coordinate trajectory is between -0.06m and 0.065m for one-step ahead while for five steps-ahead the absolute X- coordinate error is less than 0.06m. For Y-coordinate tracking error is between -0.065m and 0.035m for one step-ahead while for five steps-ahead the maximum absolute Y-coordinate errors are less than 0.035m. The maximum tracking error in the orientation of the trajectory is equal to ± 0.7 rad for one step-ahead but it is equal to ± 0.5rad for the five steps-ahead predictions.

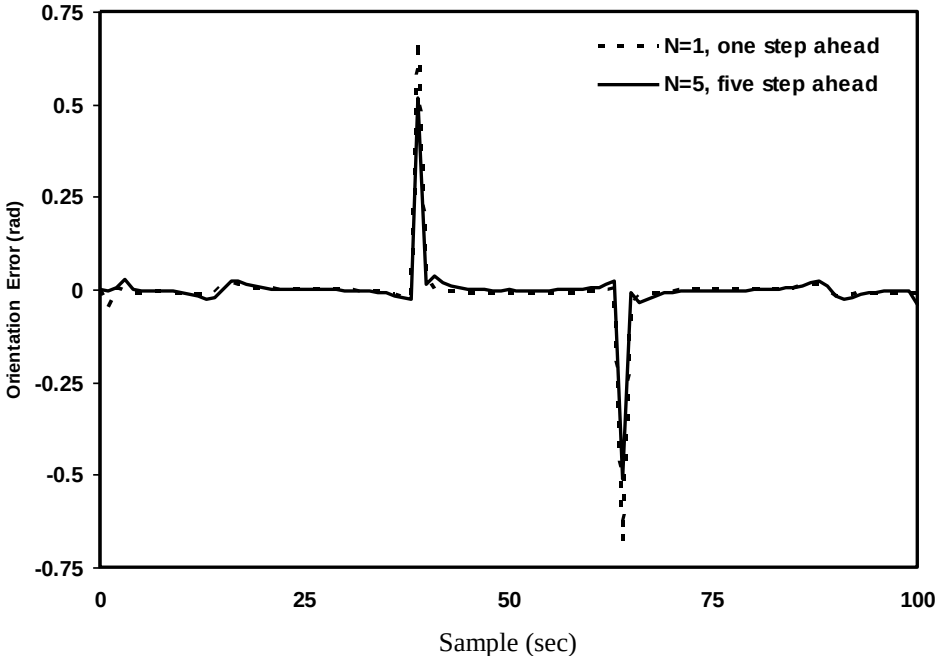


Figure 5-16: Orientation tracking error for two cases N=1, 5.

The second control methodology is the nonlinear PID neural feedback controller, which consists of eleven parameters: nine PID parameters $(Kp_\gamma, Ki_\gamma, Kd_\gamma)$ where γ is x, y, θ; and the additional two parameters Q and R. Therefore, the investigation of best parameters (Q and R) cannot be easily undertaken. However, the same values of Q and R in the first control methodology can be used in order to make the comparison results of the simulation under identical conditions to specify the differences and similarities. Therefore, the value of Q parameter is equal to 0.01 and the value of R parameter is equal to 1 for obtaining more optimised control action. Figure 5.17 shows that the best count for N is equal to five, because of minimum position and orientation errors.

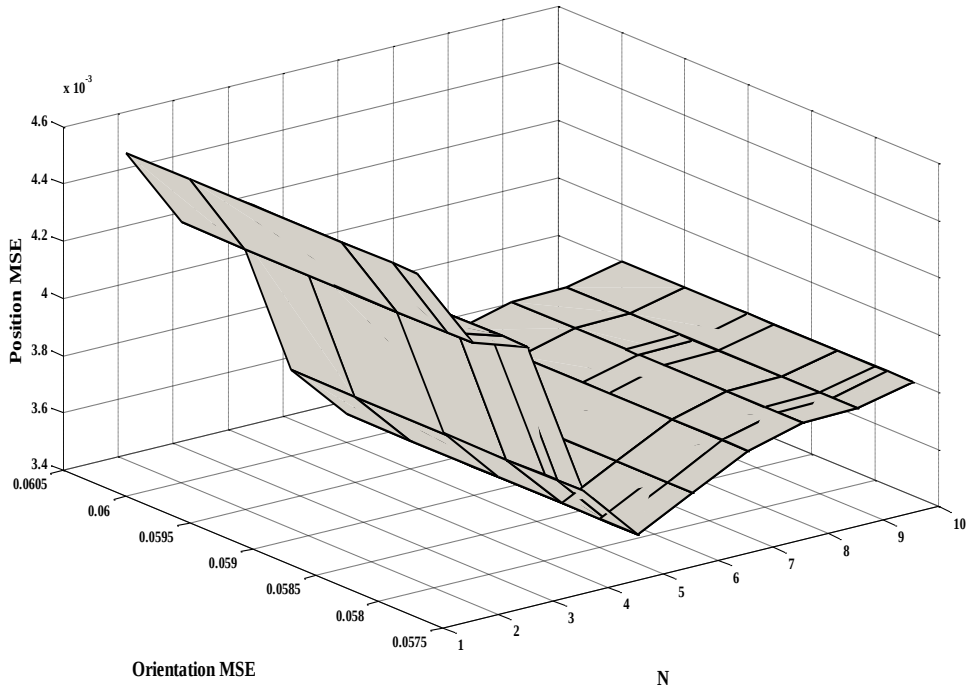
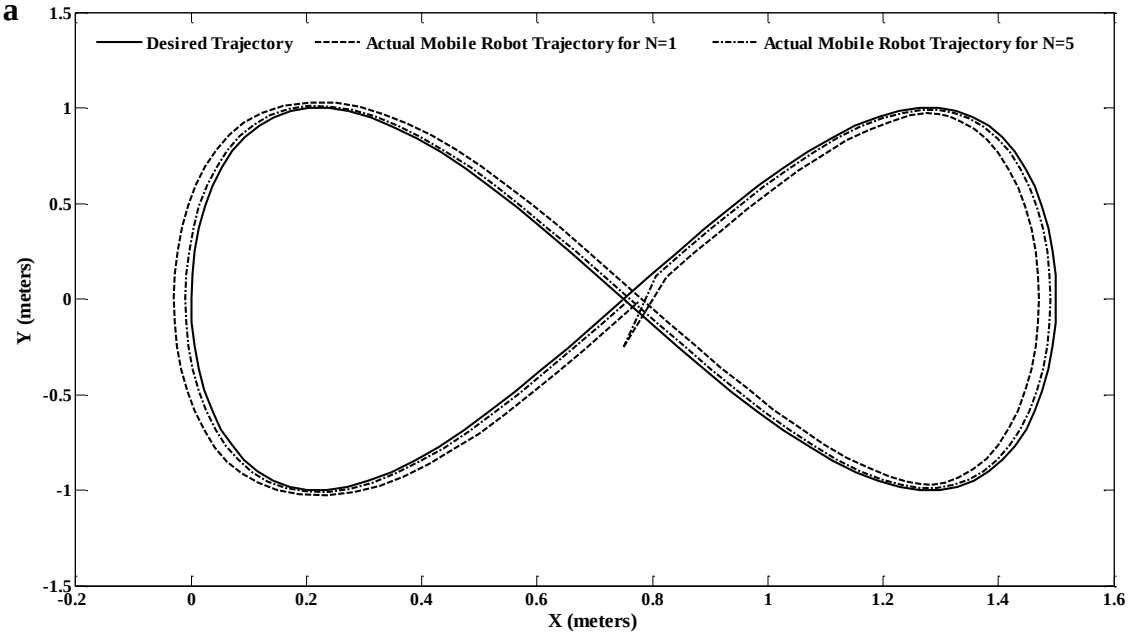


Figure 5.17: The MSE of position and orientation with N parameters.

The PID parameters can be tuned using many methods; one of these methods is trial-and-error tuning method, which is used to find the initial PID parameters, and for tuning on-line, which employs optimised-auto-tuning with identification of the mobile robot model as equations 5.99, 5.100 and 5.101. The mobile robot trajectory tracking obtained by the proposed adaptive neural predictive controller is shown in Figures 5.18a and 5.18b. These Figures demonstrate good position and orientation tracking performance for the five steps-ahead predictions in comparison with the results of one step-ahead.

a



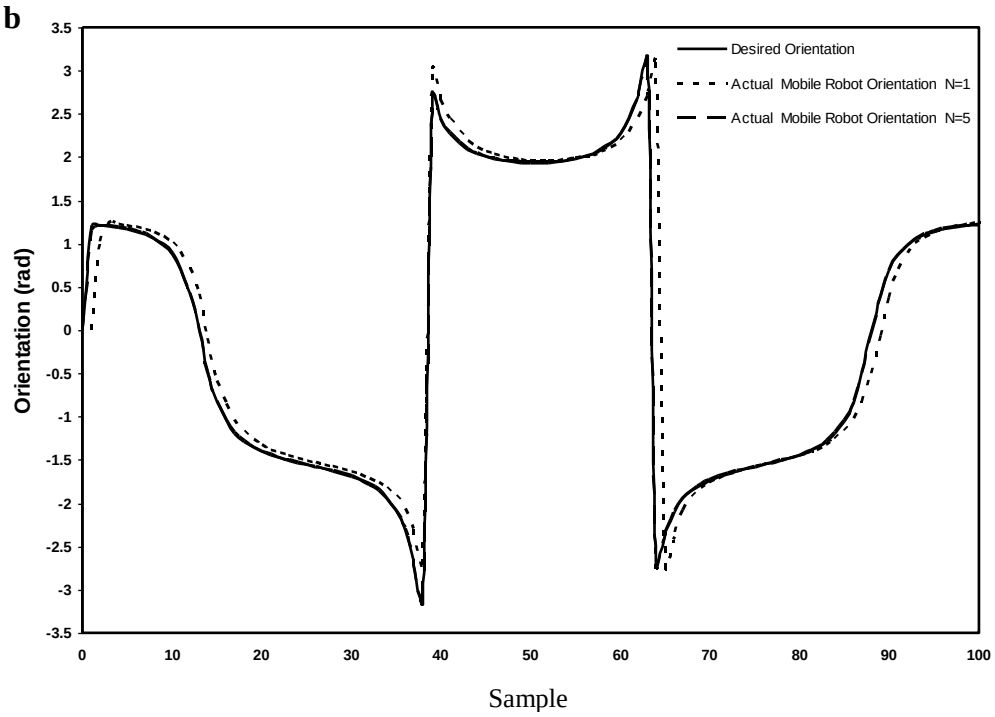
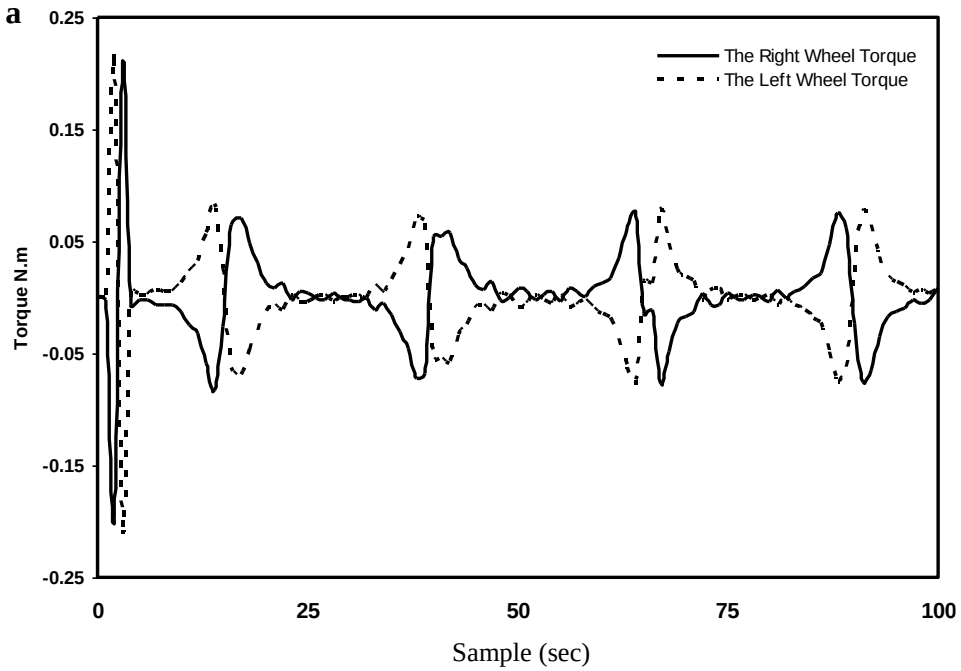


Figure 5.18: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired lemniscates trajectory; (b) actual orientation of mobile robot and desired orientation.

In spite of the existence of bounded disturbances, the adaptive learning and robustness of neural controller with optimisation algorithm show small effects of these disturbances. The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small values of the control input torques for right and left wheels with small sharp spikes, as shown in Figures 5.19a and 5.19b.



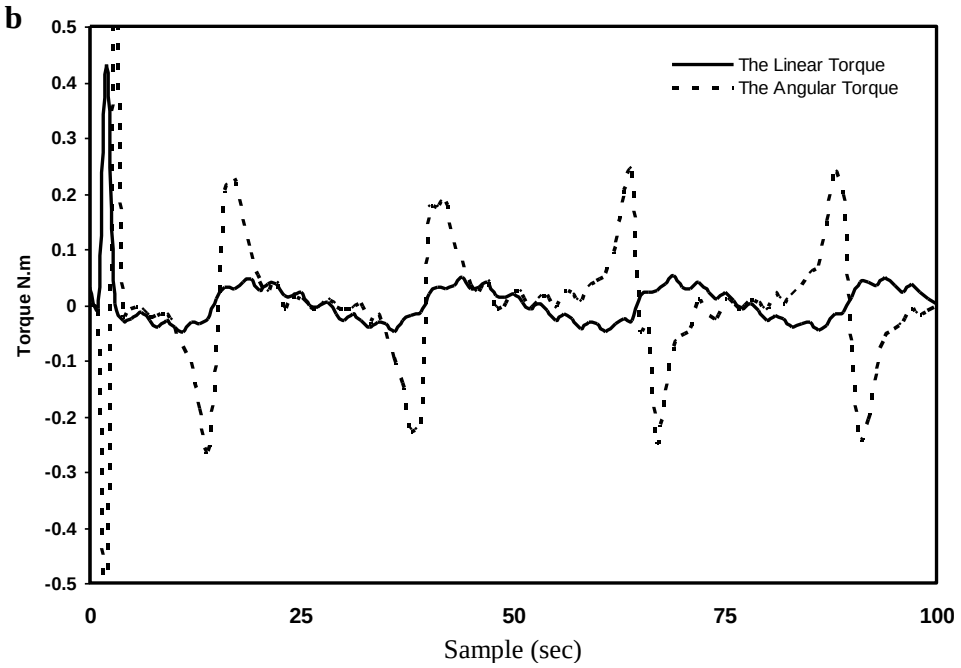
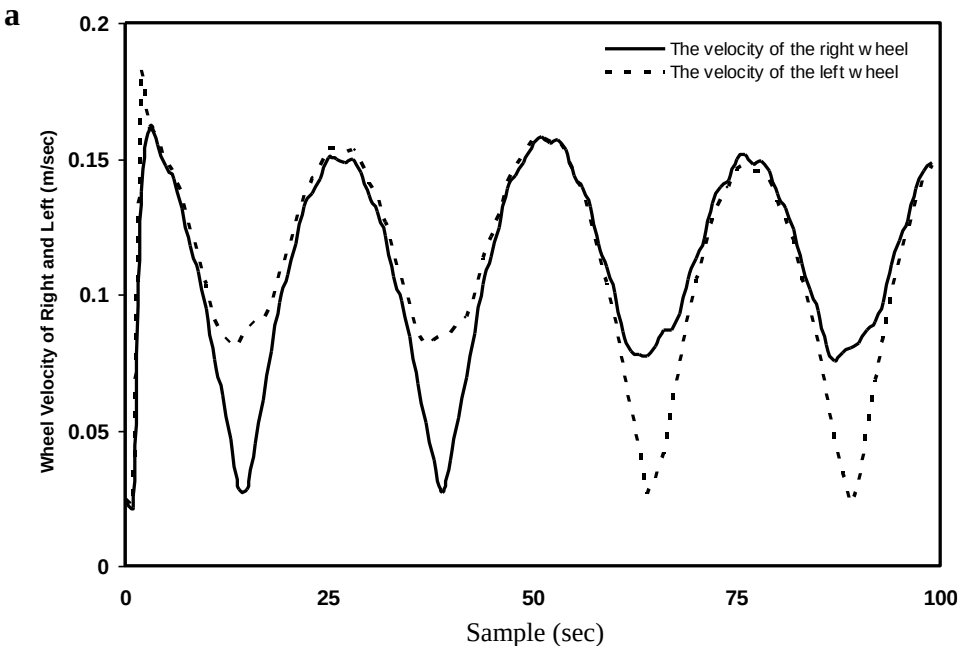


Figure 5-19: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the simulation results for right and left wheels were smooth values without sharp spikes, as shown in Figure 5.20a. The mean of linear velocity of the mobile robot is equal to 0.135m/sec, and maximum angular velocity is equal to ± 0.5rad/sec, as shown in Figure 5.20b.



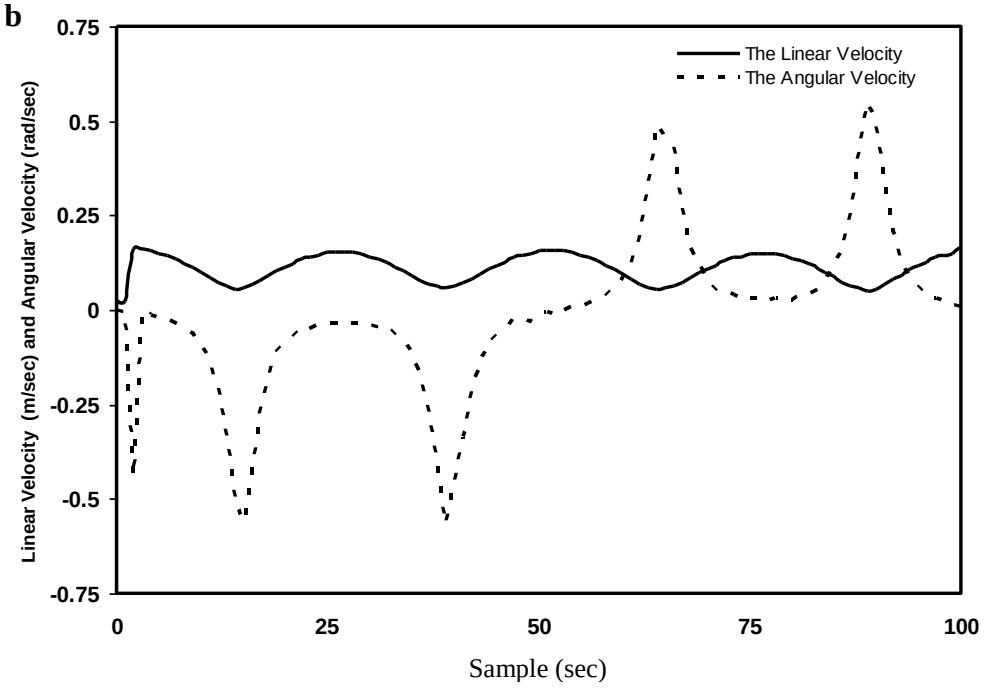
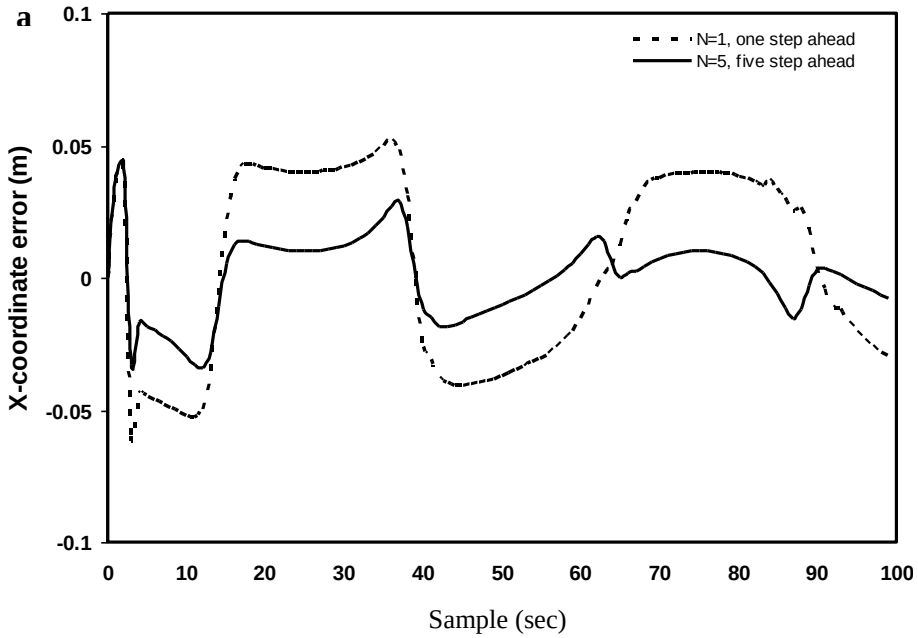


Figure 5.20: Velocity action for N=5: (a) the right and left wheel velocity; (b) the linear and angular velocity.

The effectiveness of the proposed second methodology of the adaptive neural predictive controller with predictive optimisation algorithm is clear by showing the convergence of the position and orientation trajectory error for the robot model motion as shown in Figures 5.21a, 5.21b and 5.22 respectively for N=1 and 5 steps-ahead-predictive.

The absolute maximum tracking error in the X-coordinate trajectory is less than 0.05m for one step-ahead while for five steps-ahead the X- coordinate error is less than ± 0.03m.



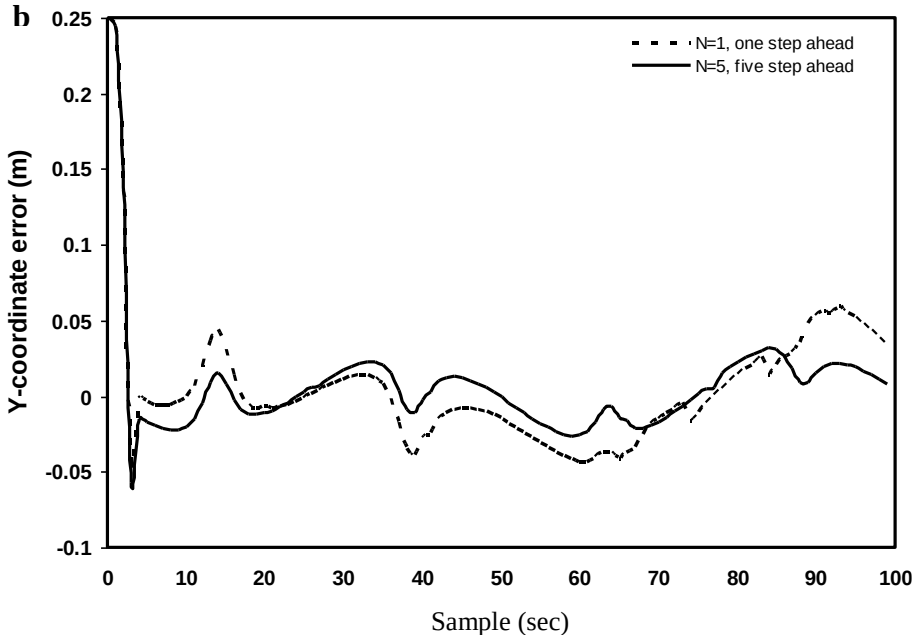


Figure 5.21: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

Tracking error is less than ± 0.05m for one step-ahead for Y-coordinate, while for five steps-ahead the error has declined to less than ± 0.025m. The maximum tracking error in the orientation of the trajectory is equal to ± 0.65rad for one step-ahead but it is equal to ± 0.34rad for five steps-ahead

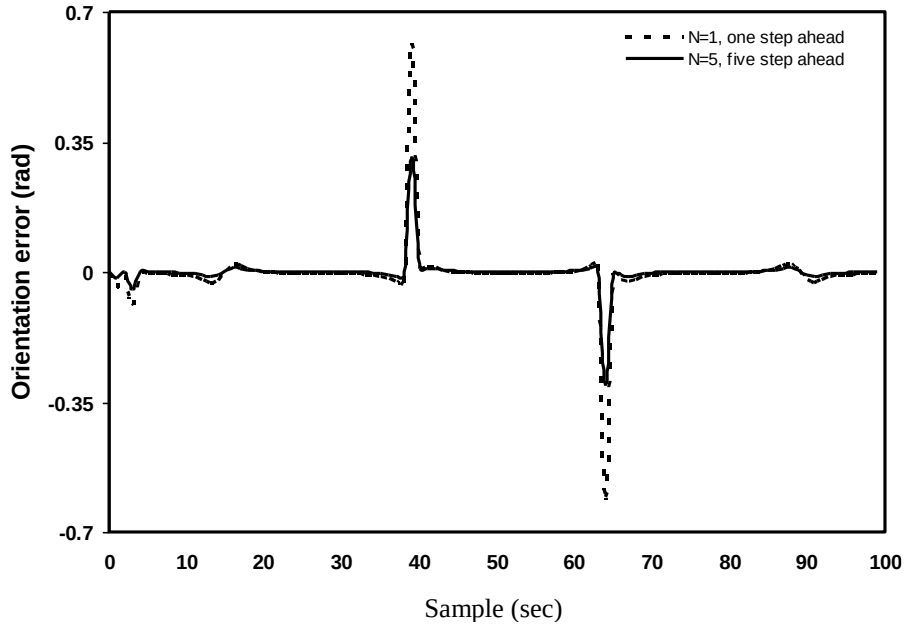


Figure 5.22: Orientation tracking error for two cases N=1, 5.

The parameters of the position and orientation nonlinear PID controller $(Kp_\gamma, Ki_\gamma, Kd_\gamma)$ are tuned by using optimised-auto-tuning with posture identifier model of the mobile robot model, as shown in Figures 5.23 and 5.24 for one step-ahead and five steps-ahead respectively, with initial values chosen from trial-and-error tuning method, which are 1, 0.1 and 0.2 for $(Kp_\gamma, Ki_\gamma, Kd_\gamma)$ respectively.

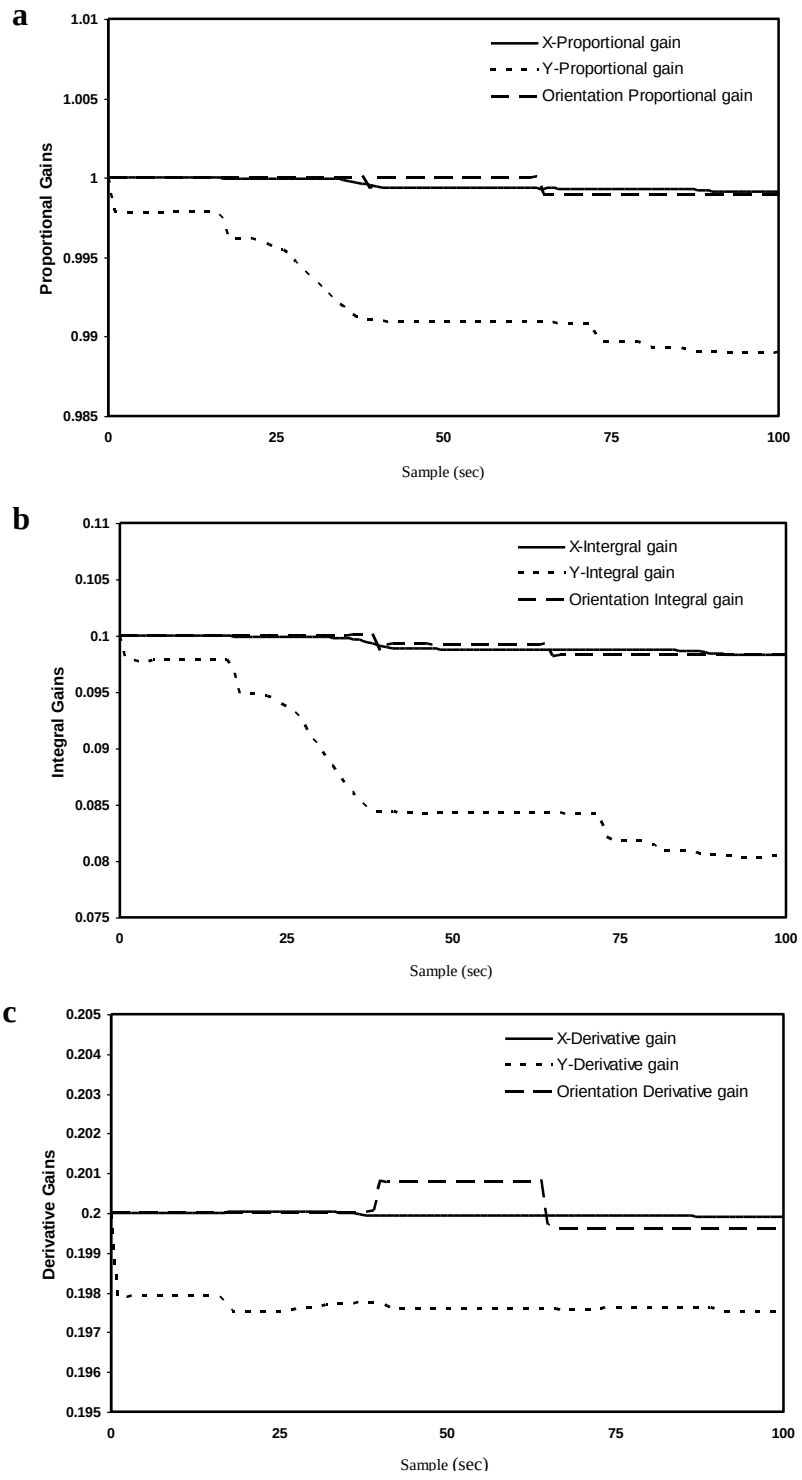


Figure 5.23: The position and orientation PID controller parameters for N=1:
(a) proportion gain; (b) integral gain; (c) derivative gain.

The optimal value of the orientation derivative gain at sample 40 to 64 is 0.201 in order to track the desired orientation and reduce the orientation error to 0.65rad based on the equation (5.130).

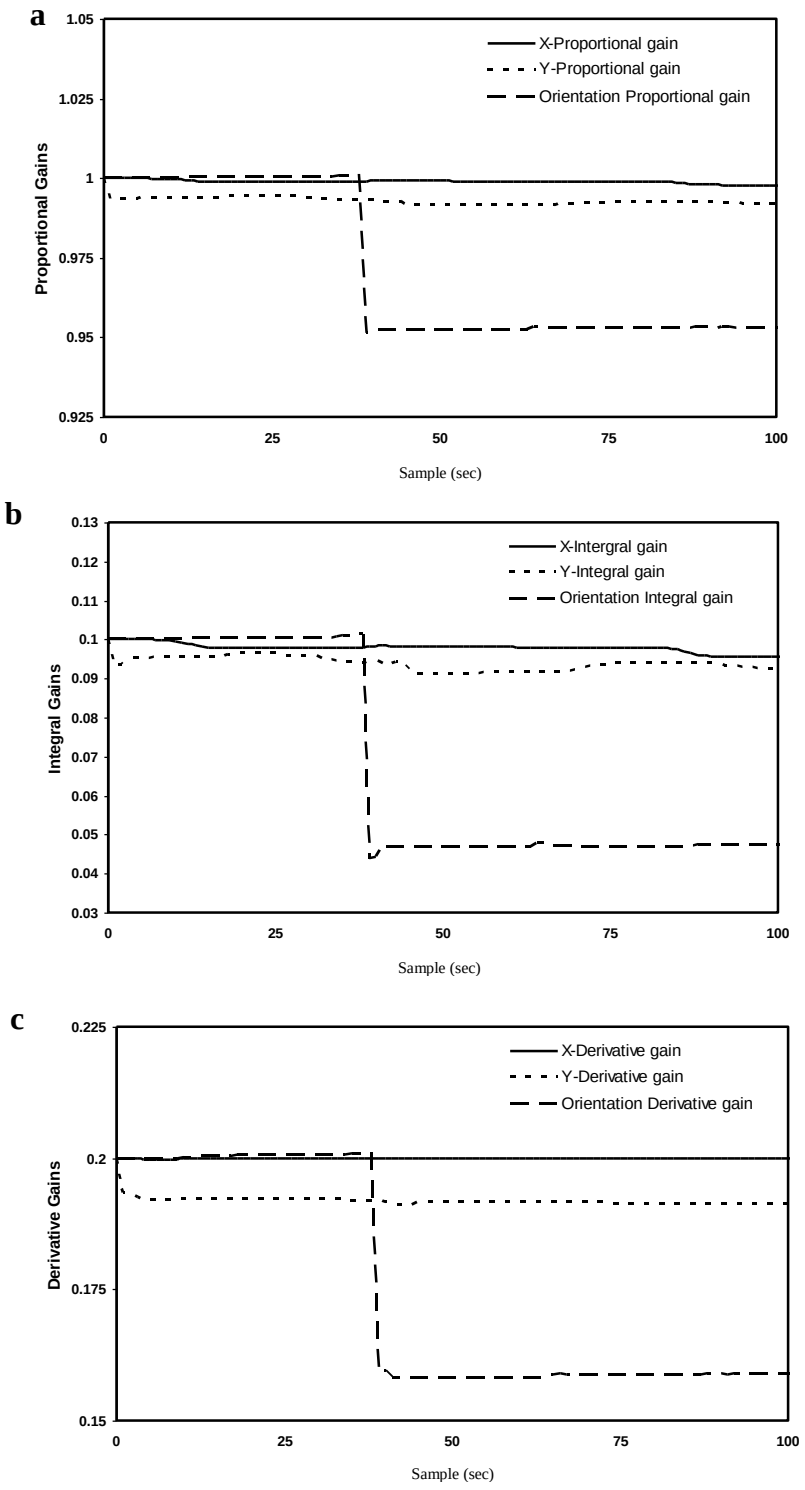


Figure 5.24: The position and orientation PID controller parameters for N=5:
(a) proportion gain; (b) integral gain; (c) derivative gain.

The third control methodology is nonlinear inverse dynamic neural feedback controller, which consists of five parameters for tuning effort $(k_x, k_y, k_\theta, Q, R)$. The same values have been used in the first and second control methodology (Q equal to 0.01 and R equal to 1) in order to make the comparison results of the simulation under identical conditions to specify the differences and similarities. The values of the parameters (k_x, k_y, k_θ) of the inverse dynamic neural controller is found using equations 5.240 and 5.243, and tuned by using equations 5.244, 5.245 and 5.246. The sampling time is equal to 0.5sec and the desired damping coefficient is equal to $\xi = 0.2$ for non-oscillation and fast response of mobile robot. The characteristic frequency w_n is equal to 5rad/sec with constant scheduling gain equal to $\sigma = 10$. Figure 5.25 shows that the best count for N is equal to five because of minimum position and orientation mean square errors.

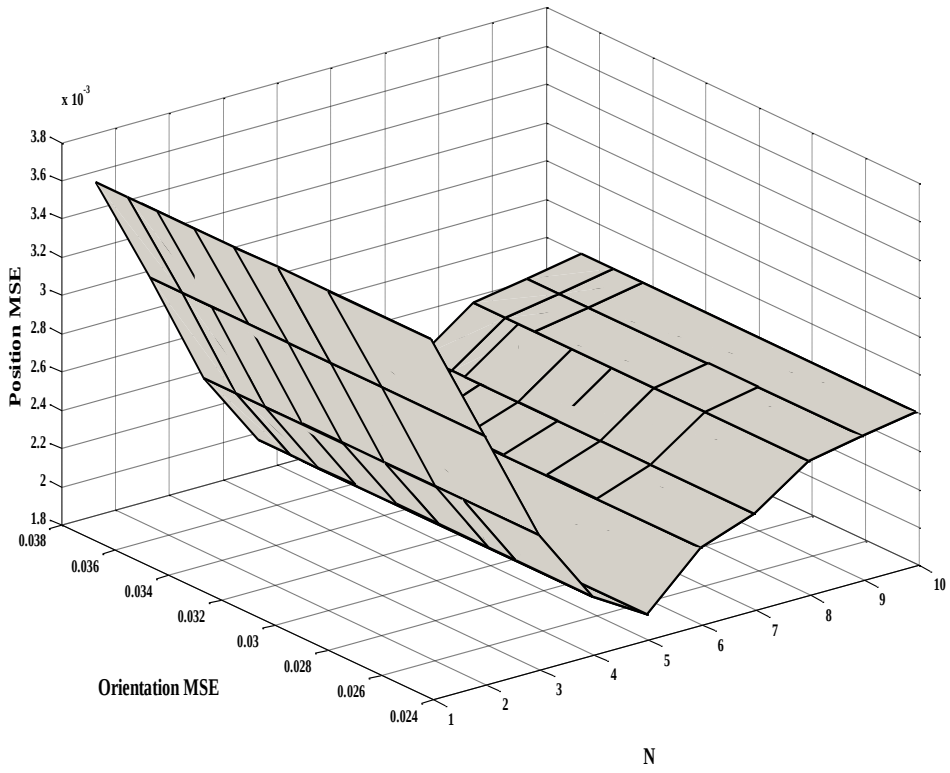


Figure 5.25: The MSE of position and orientation with N parameters.

The robot trajectory tracking obtained by the proposed adaptive neural predictive controller is shown in Figures 5.26a and 5.26b. These Figures demonstrate excellent position and orientation tracking performance for the five steps-ahead predictions in comparison with results of one step-ahead. In spite of the existence of bounded disturbances the adaptive learning and robustness of neural controller with optimisation algorithm show very small effects of these disturbances.

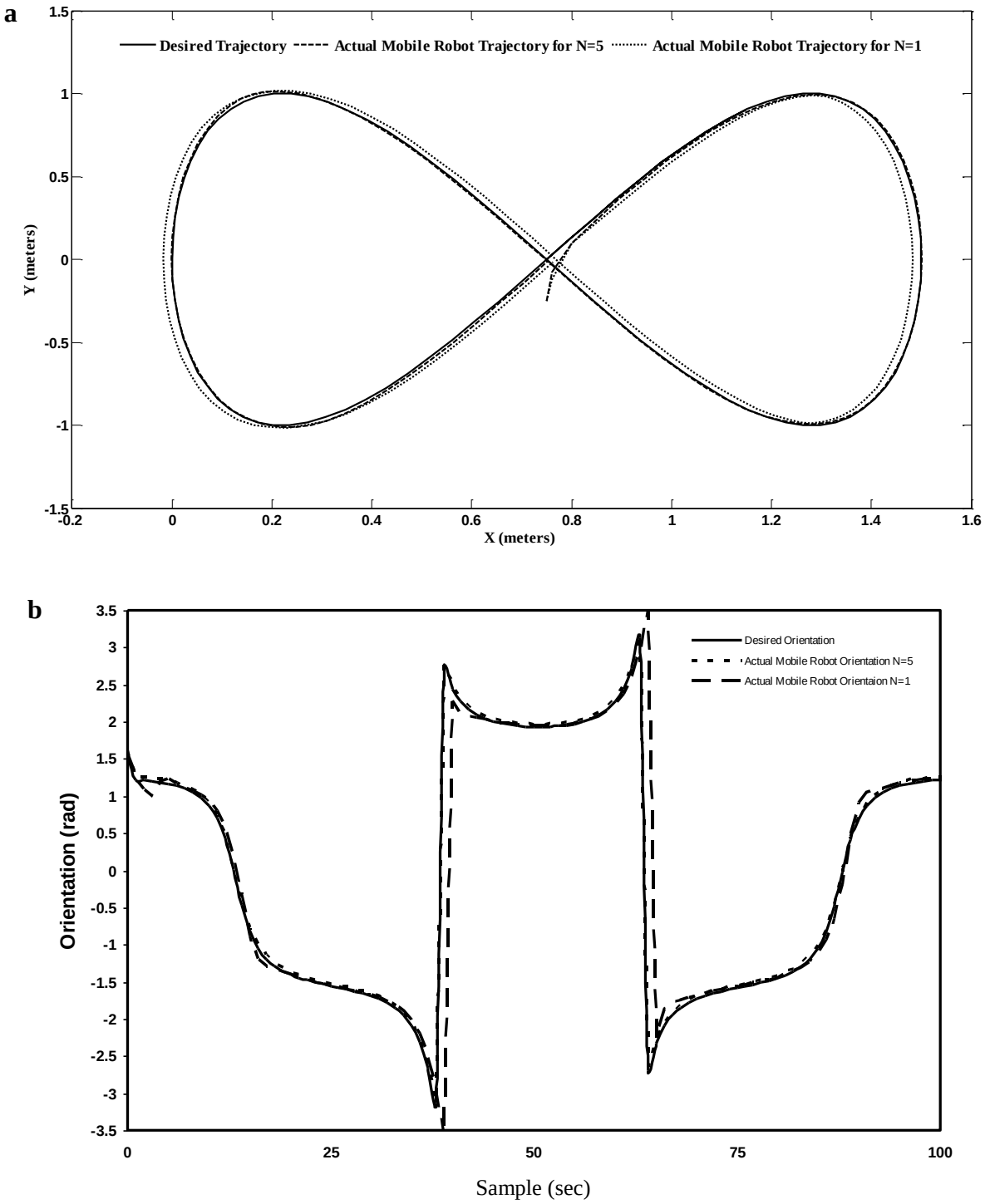


Figure 5.26: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired lemniscates trajectory; (b) actual orientation of mobile robot and desired orientation.

The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small values of the control input torques for right and left wheels with small sharp spikes, as shown in Figures 5.27a and 5.27b.

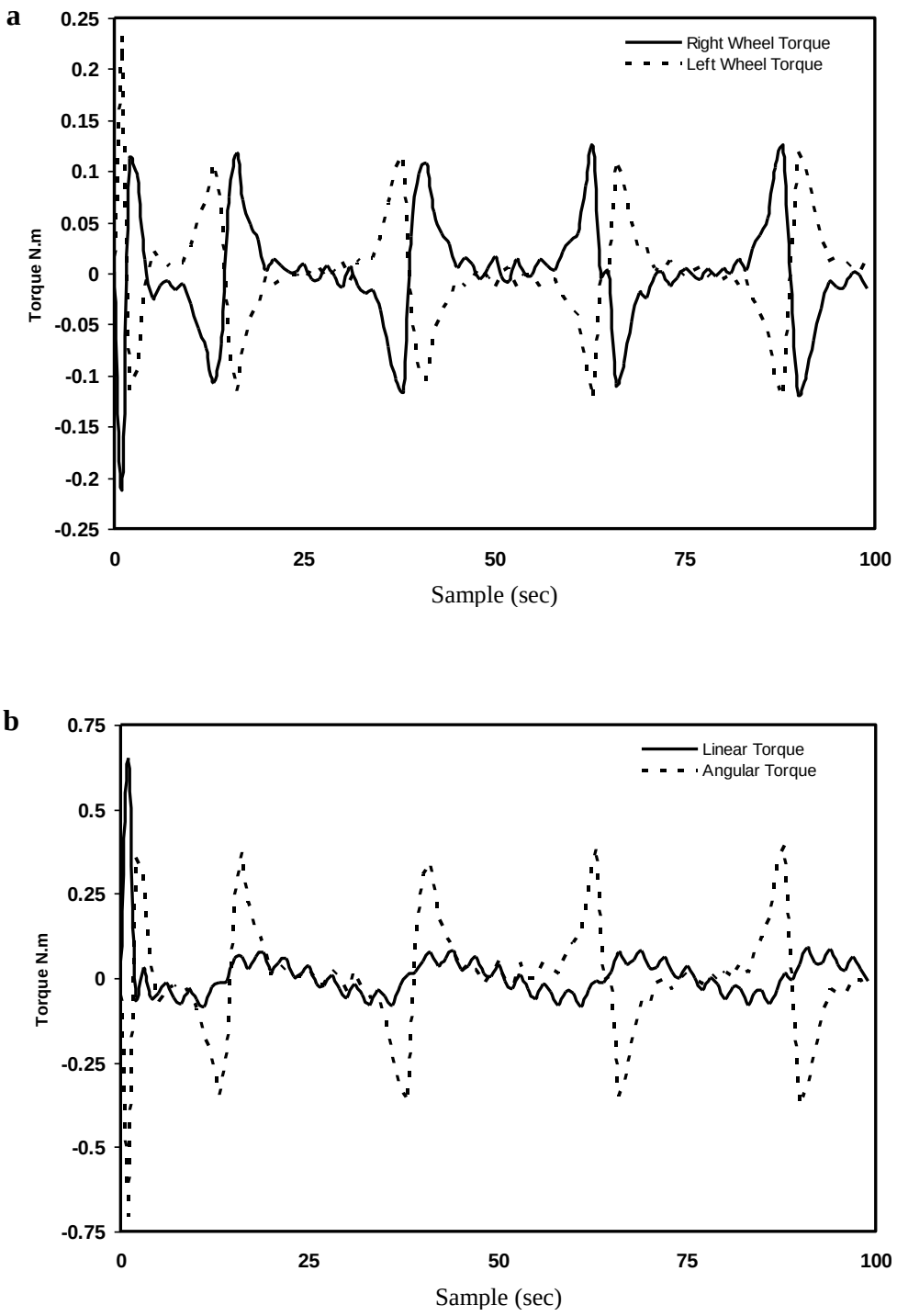


Figure 5.27: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the simulation results for right and left wheels were smooth values without sharp spikes, as shown in Figure 5.28a. The mean linear velocity of the mobile robot is equal to 0.135m/sec, and maximum angular velocity is equal to ± 0.65rad/sec, as shown in Figure 5.28b.

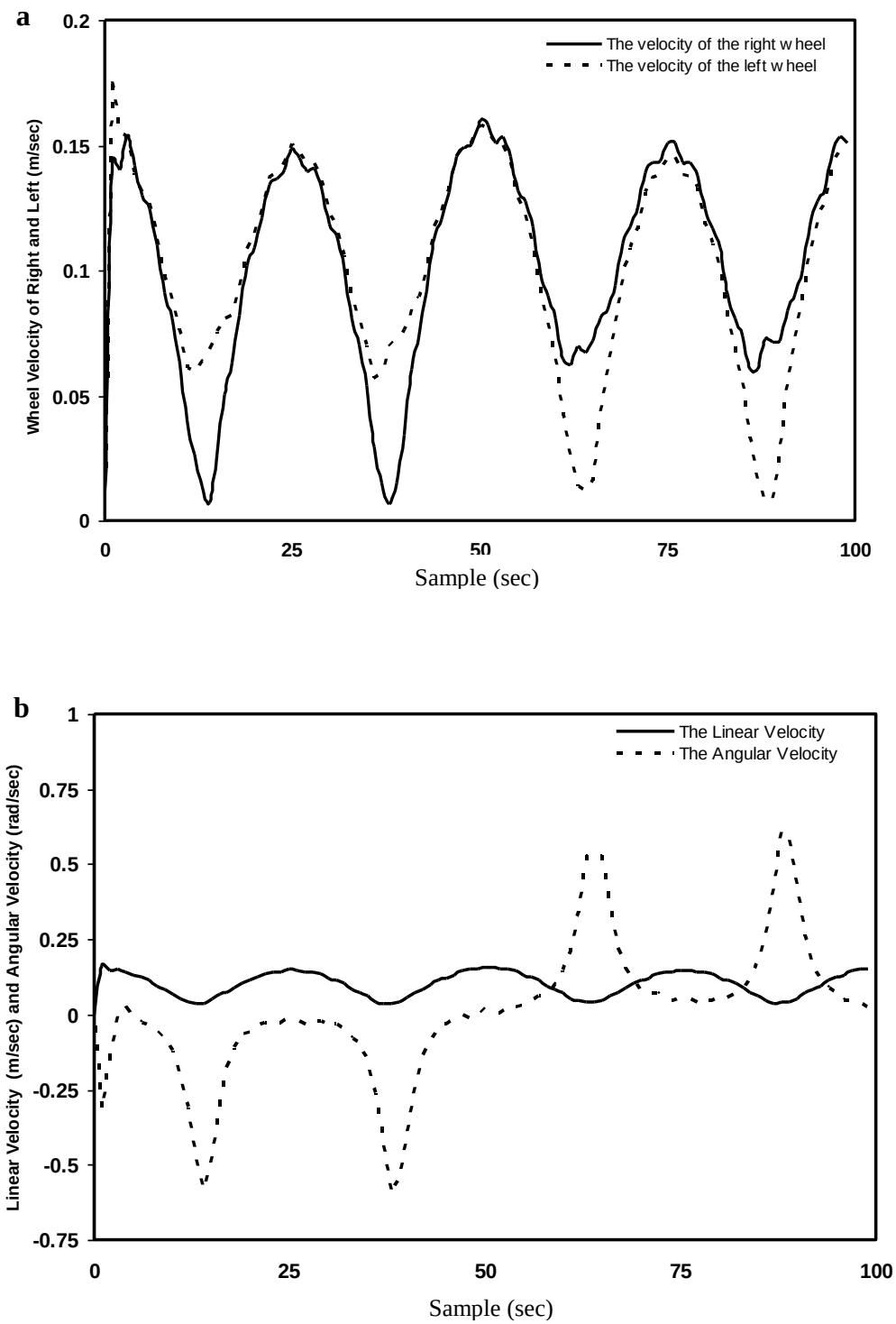


Figure 5.28: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

The effectiveness of the proposed third methodology of the adaptive neural predictive controller with predictive optimisation algorithm is clear by showing the convergence of the position and orientation trajectory error for the robot model motion as shown in Figures 5.29a, 5.29b and 5.30 respectively for N=1 and 5 steps-ahead-predictive. The

maximum tracking error in the X-coordinate trajectory is equal to 0.02m for one-step ahead while for the five steps ahead the X- coordinate error is less than 0.0075m. The maximum tracking error in Y-coordinate is equal to 0.03m for one-step ahead while for the five steps ahead the error has declined to less than 0.00125m.

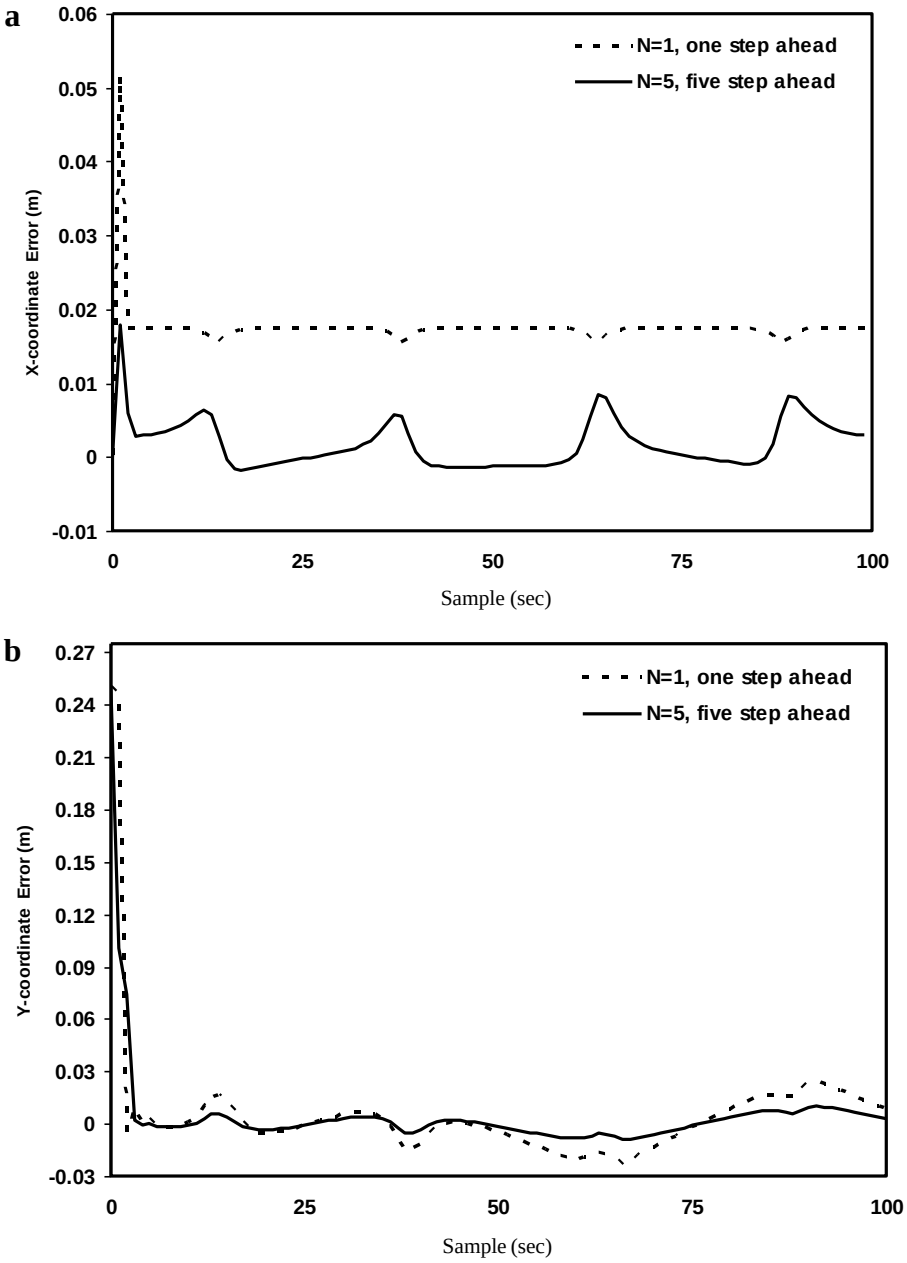


Figure 5.29: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

The maximum tracking error in the orientation of the trajectory is equal to ± 0.5rad for one step-ahead but it is equal to ± 0.25rad for the five steps-ahead.

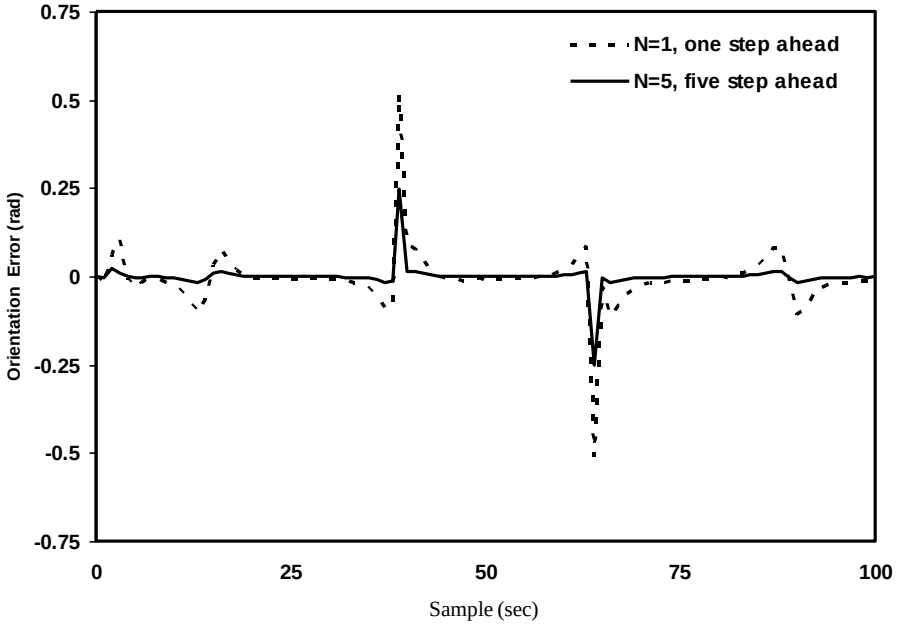
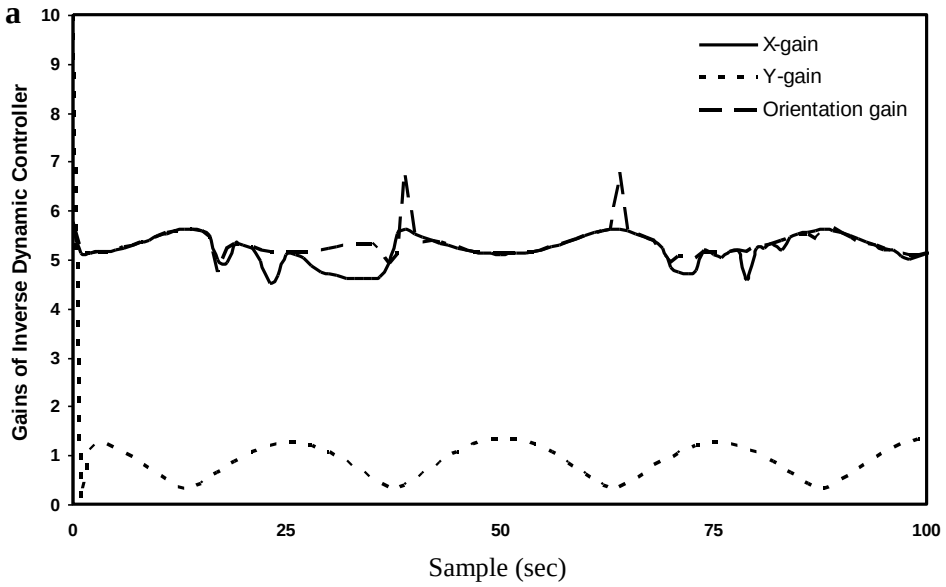


Figure 5.30: Orientation tracking error for two cases N=1, 5.

The optimised-auto-tuning with identification of the mobile robot model is used for tuning the parameters of the inverse dynamic controller (k_x, k_y, k_θ) for one step-ahead and five steps-ahead, as shown in Figures 5.31a and 5.31b respectively.



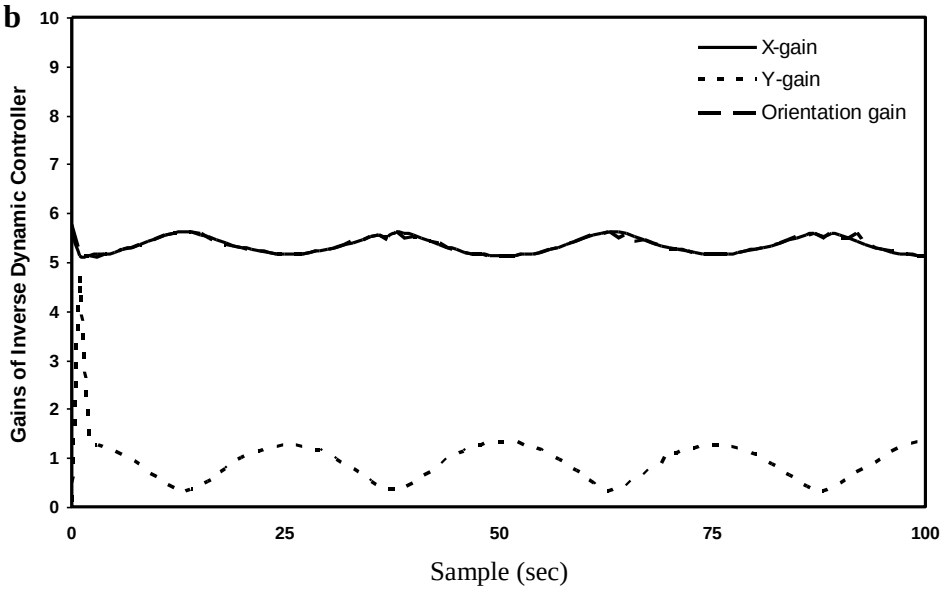


Figure 5.31: The inverse dynamic controller parameters: (a) N=1; (b) N=5.

The mean-square error for each component of the state error $(q_r - q) = (e_x, e_y, e_\theta)$, for the five steps-ahead predictive control of each methodology can be shown in Table 5.1.

Table 5.1: MSE of each methodology for five steps-ahead prediction of desired lemniscates trajectory.

$MSE(q_r - q) = (e_x, e_y, e_\theta)$	Methodology1	Methodology2	Methodology3
MSE of X-coordinate	0.00018	0.00017	0.00015
MSE of Y-coordinate	0.00081	0.00068	0.00061
MSE of Orientation	0.00523	0.00232	0.00125

The third control methodology has smaller mean square errors for the position (X-coordinate, Y-coordinate) and the Orientation than first and second control methodologies, which have high MSE for the posture, as shown in Table 5.1. Therefore, it can be considered that the third control methodology was the best methodology for the desired lemniscates trajectory.

5.4.2 Case Study 2

The desired circular trajectory, which has explicitly continuous gradient with rotation radius constant, can be described by the following equations:

$$x_r(t) = \cos(\frac{t}{10}) \tag{5.309}$$

$$y_r(t) = \sin(\frac{t}{10}) \tag{5.310}$$

$$\theta_r(t) = \frac{\pi}{2} + \frac{t}{10} \tag{5.311}$$

The mobile robot model starts from the initial position and orientation $q(0) = [1.1, -0.5, \pi/2]$ as its initial conditions with the external disturbance. The best values of Q and R parameters in this case are equal to 0.025 and 1 respectively for minimising the position and orientation mean square errors and obtaining more optimised control action. Using the same stages of the proposed controller of the first methodology, the robot trajectory tracking obtained by the proposed controller is shown in Figures 5.32a and 5.32b, which is good pose tracking performance for the five steps-ahead prediction in comparison with results of one step-ahead prediction.

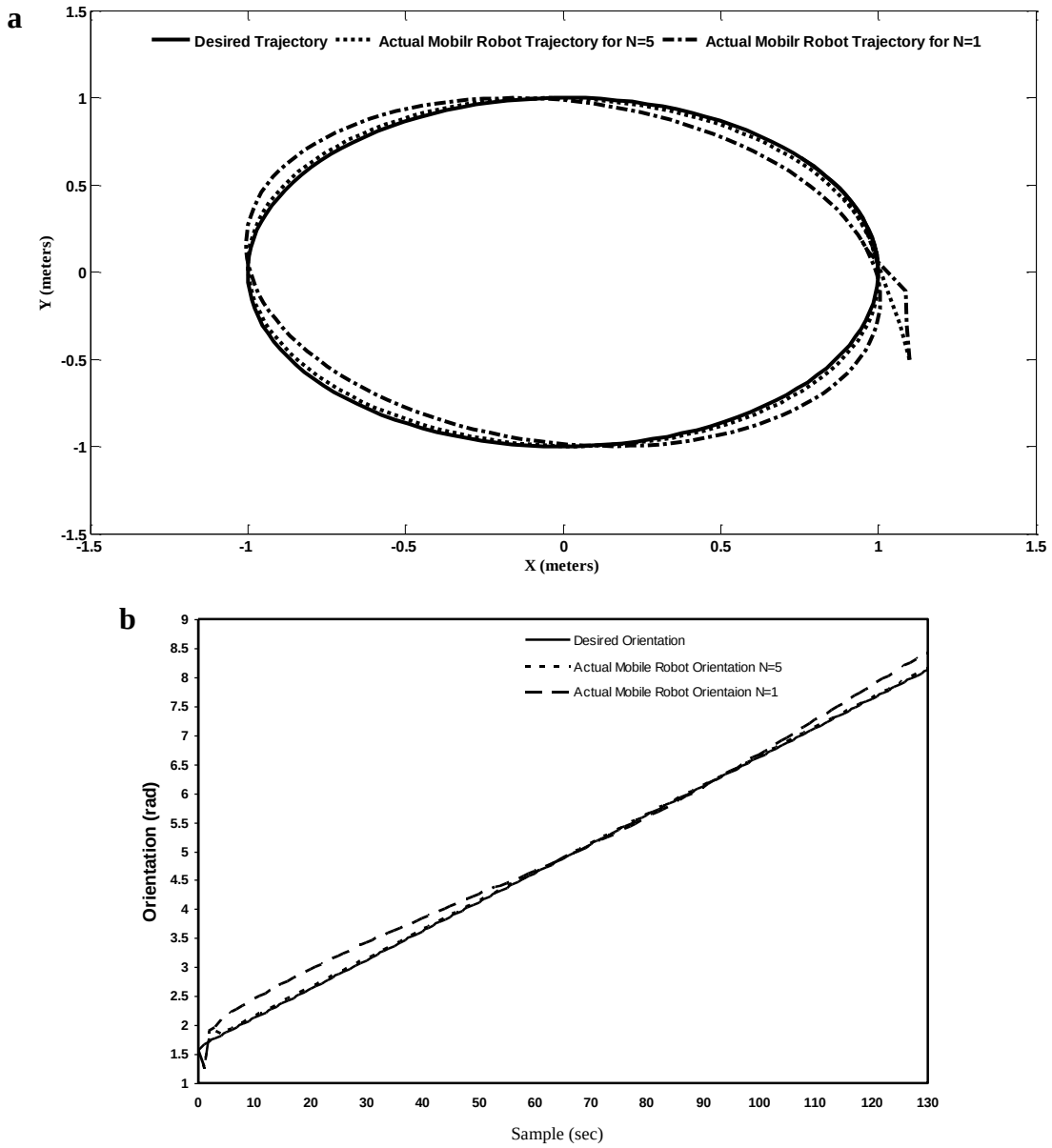


Figure 5.32: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired circular trajectory; (b) actual orientation of mobile robot and desired orientation.

The behaviour of the control action torques for right and left wheels is small, with smoothness values without sharp spikes, as shown in Figures 5.33a and 5.33b.

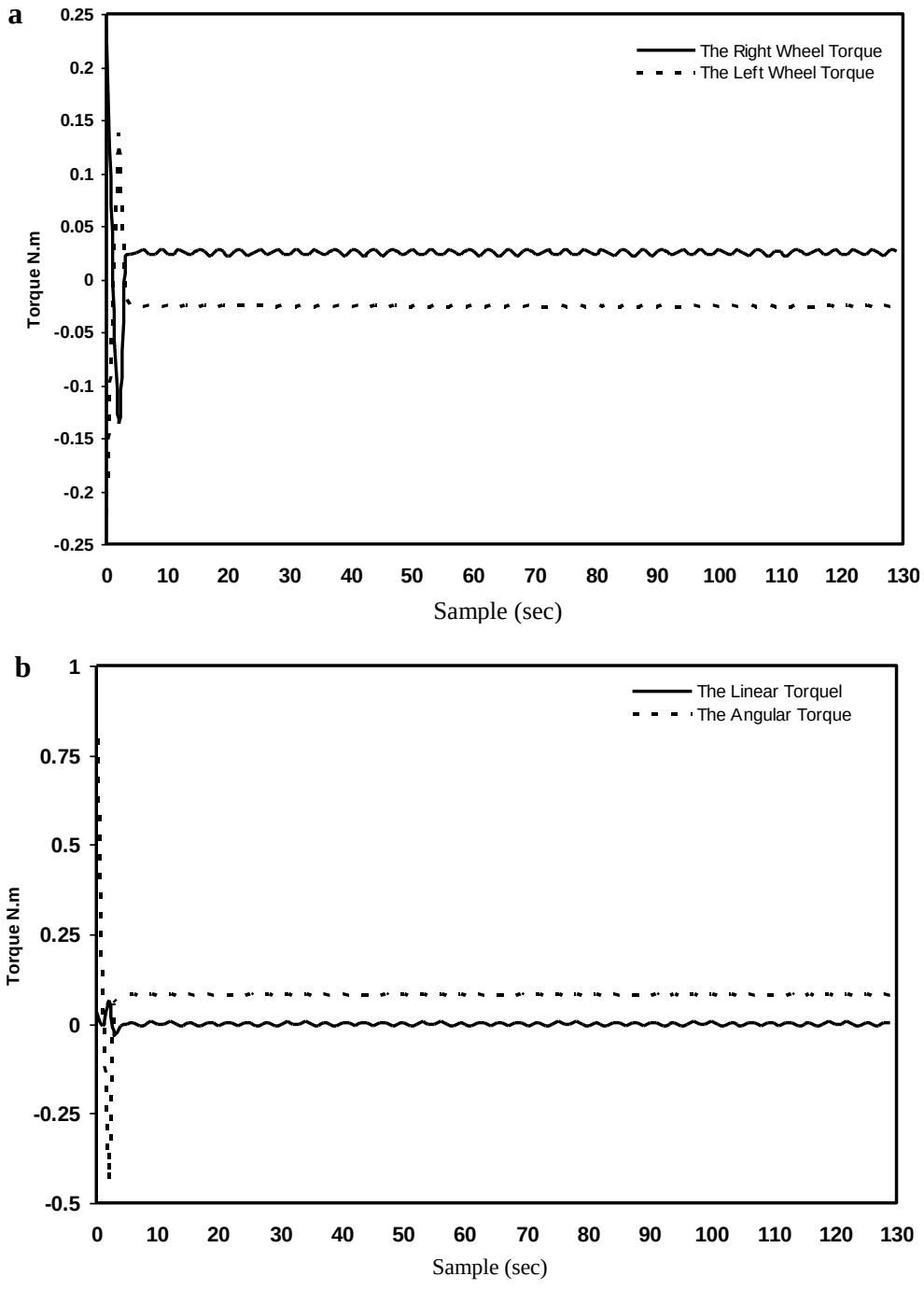


Figure 5.33: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the right wheel and the left wheel were smooth values without sharp spikes, as shown in Figure 5.34a. The mean of linear velocity of the mobile robot is equal to 0.0975m/sec, and the rate of the angular velocity is equal to 0.75rad/sec, as shown in Figure 5.34b.

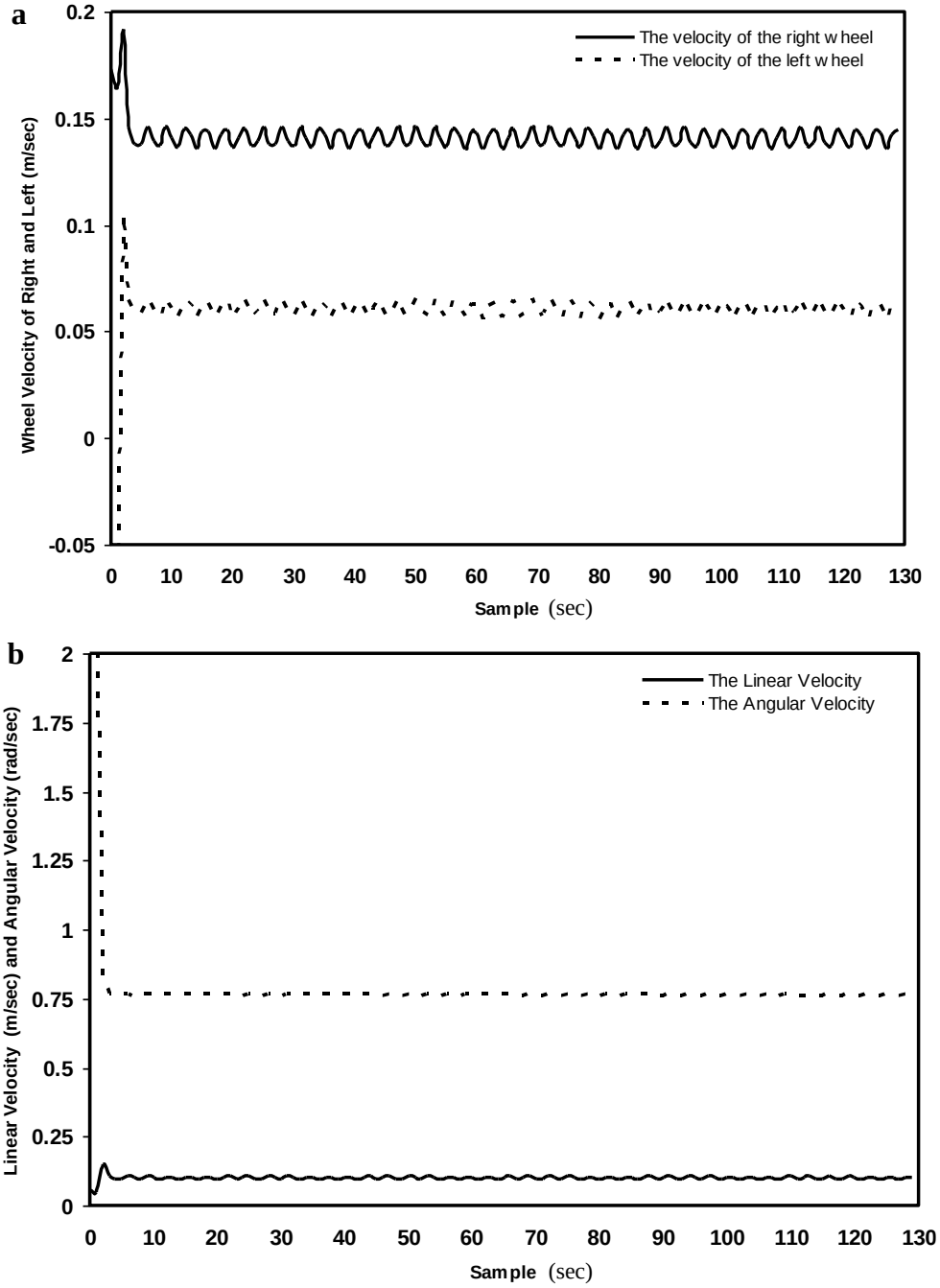


Figure 5.34: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

In addition, the convergence of the position and orientation trajectory error for the mobile robot model motion is very evident as illustrated in Figures 5.35a, 5.35b and 5.36 respectively for one and five steps-ahead prediction. The maximum tracking error in the X-coordinate trajectory is equal to ± 0.125m for one step-ahead while for the five steps-ahead the maximum X coordinate error in the circular trajectory is sloped less than ± 0.025m.

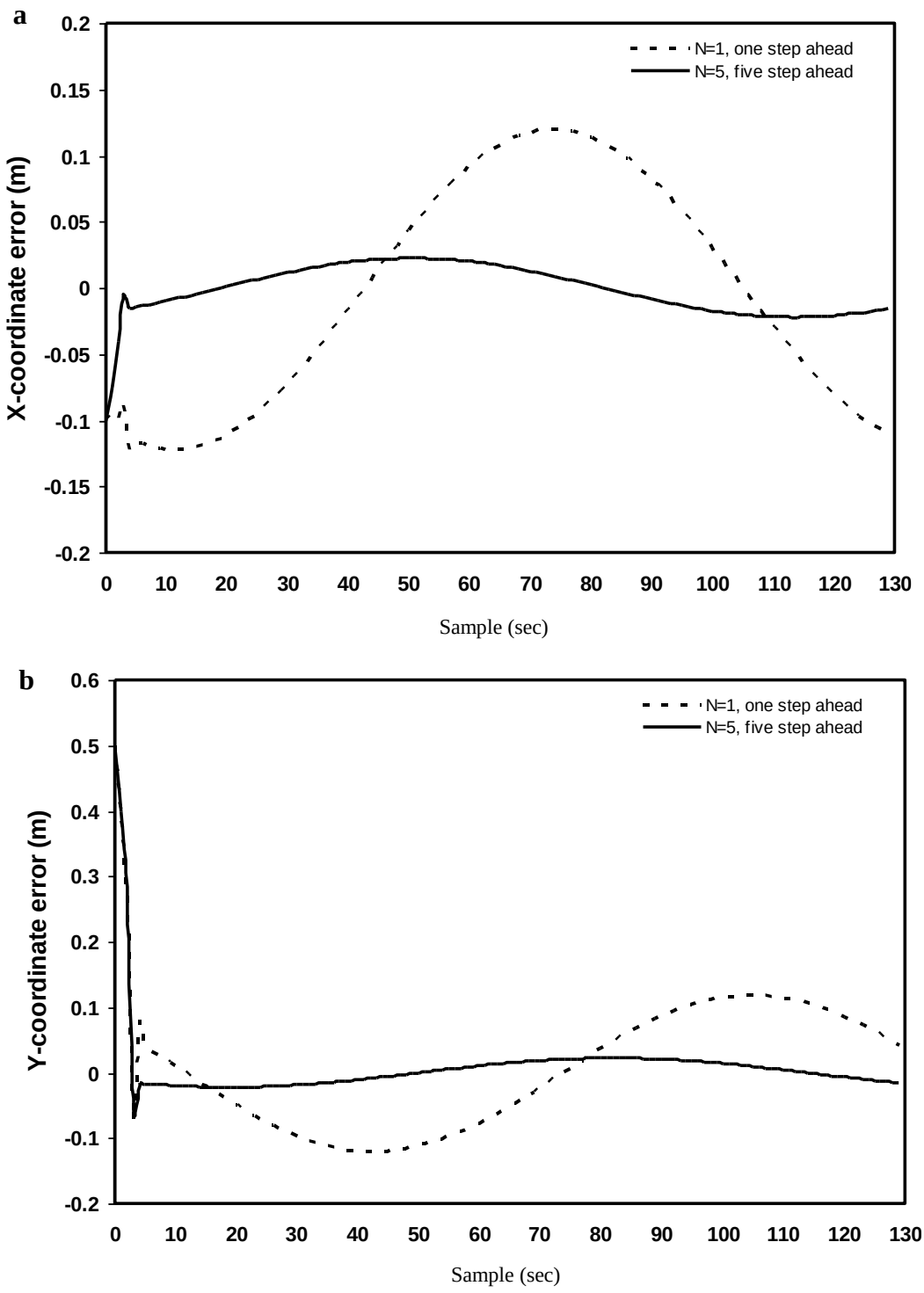


Figure 5.35: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

For Y-coordinate tracking error is equal to ± 0.1m for one step-ahead while for the five steps-ahead the error has declined to less than ± 0.013m. The maximum tracking error in the orientation of the trajectory is equal to 0.46rad for one step-ahead, but it is equal to 0.43rad for five steps-ahead

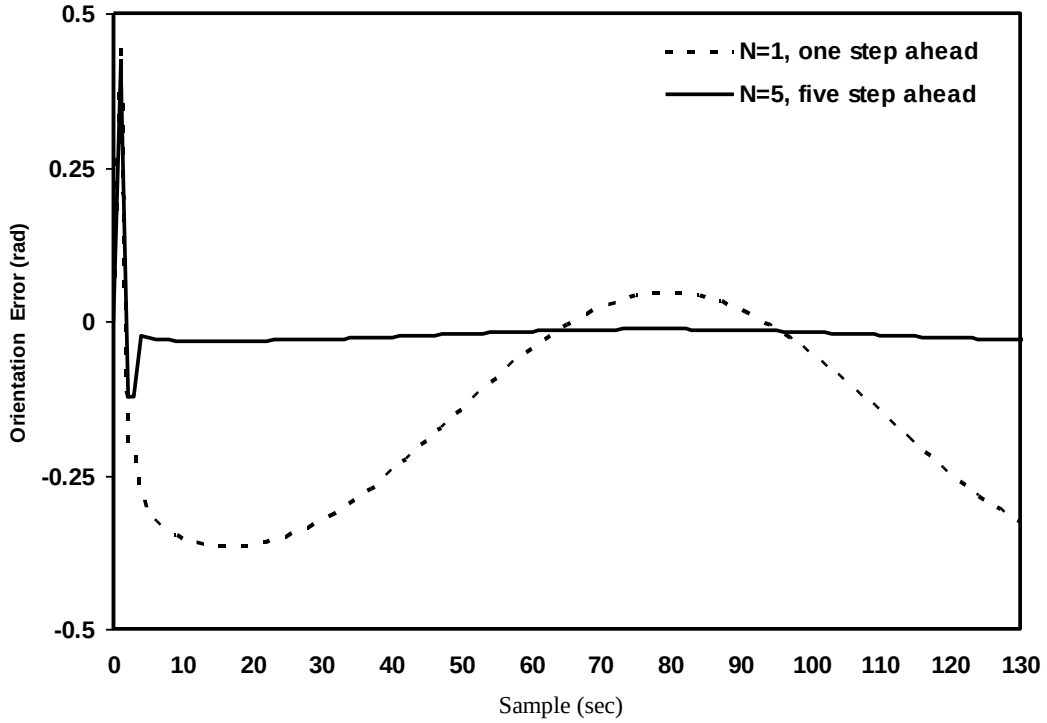
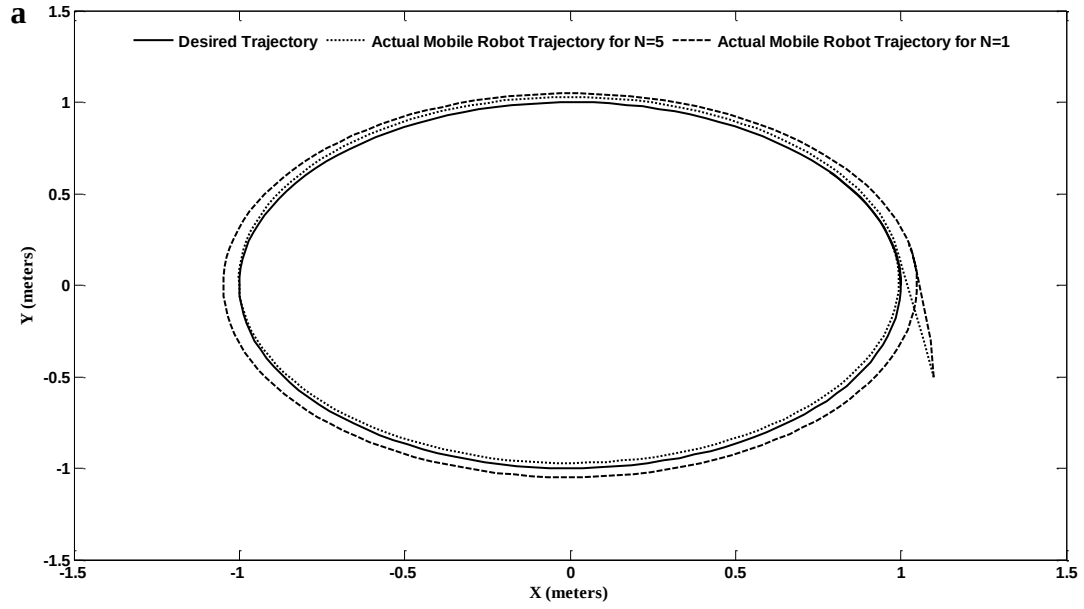


Figure 5.36: Orientation tracking error for two cases N=1, 5.

For the second control methodology, the nonlinear PID neural feedback controller is used with the initial nonlinear PID parameters $(Kp_\gamma, Ki_\gamma, Kd_\gamma)$ equal to 1, 0.1 and 0.2 respectively. The mobile robot trajectory tracking obtained by the proposed adaptive neural predictive controller is shown in Figures 5.37a and 5.37b. These Figures demonstrate excellent position and orientation tracking performance for the five steps-ahead predictions in comparison with results of one step-ahead.



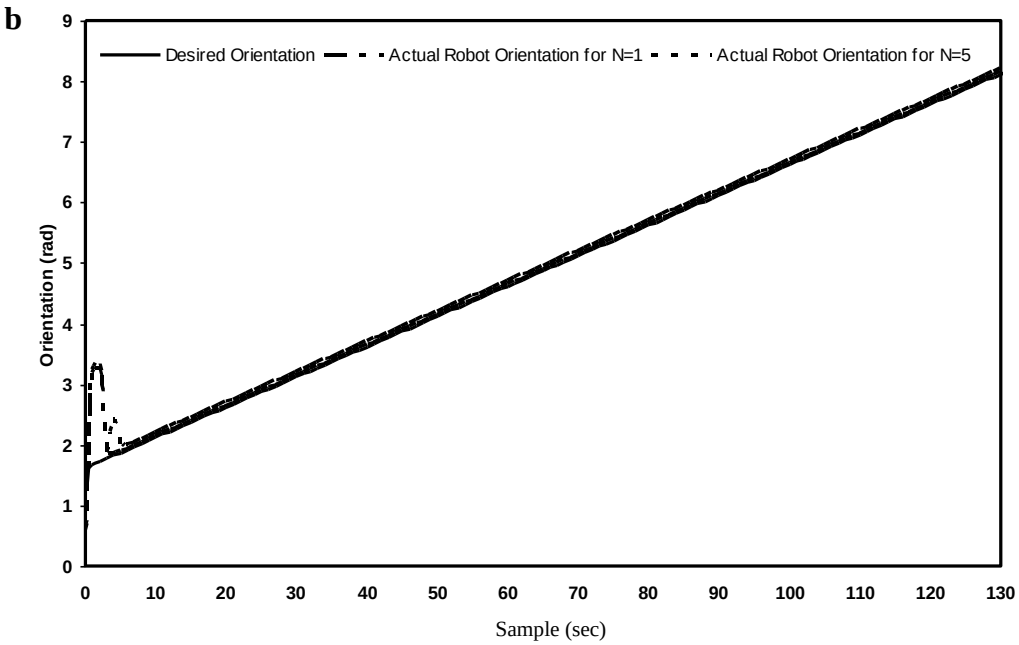
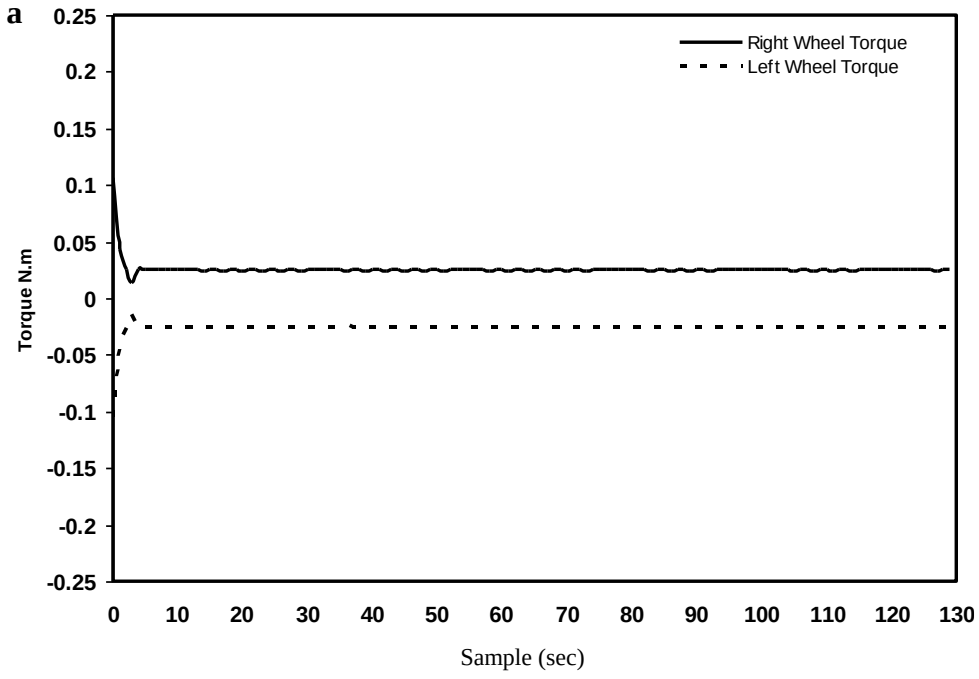


Figure 5.37: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired circular trajectory; (b) actual orientation of mobile robot and desired orientation.

In spite of the existence of bounded disturbances the adaptive learning and robustness of neural controller with optimisation algorithm show small effect of these disturbances. The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small smooth values of the control input torques for right and left wheels without sharp spikes. The actions described in Figures 5.38a and 5.38b show that smaller power is required to drive the DC motors of the mobile robot model.



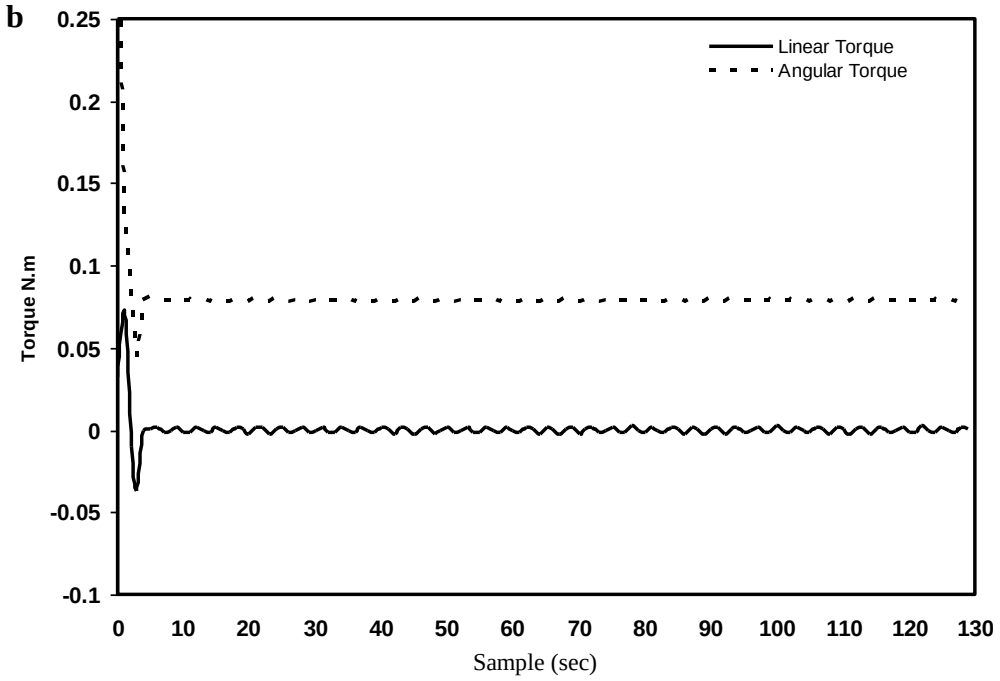
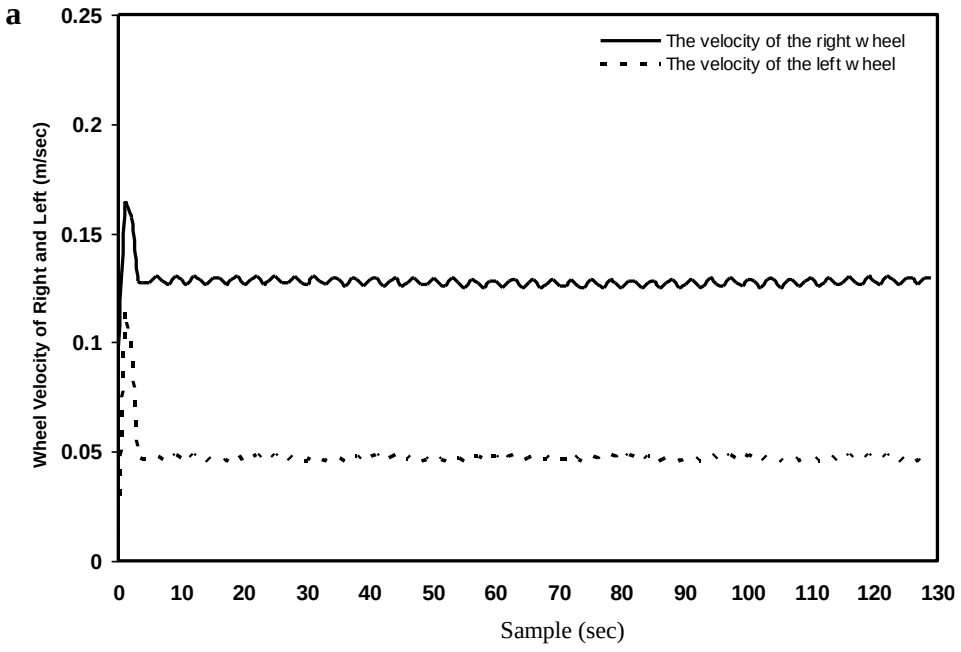


Figure 5.38: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the simulation results for right and left wheels were smooth values without sharp spikes, as shown in Figure 5.39a. The mean of linear velocity of the mobile robot is equal to 0.0975m/sec, and the rate of the angular velocity is equal to 0.77rad/sec, as shown in Figure 5.39b.



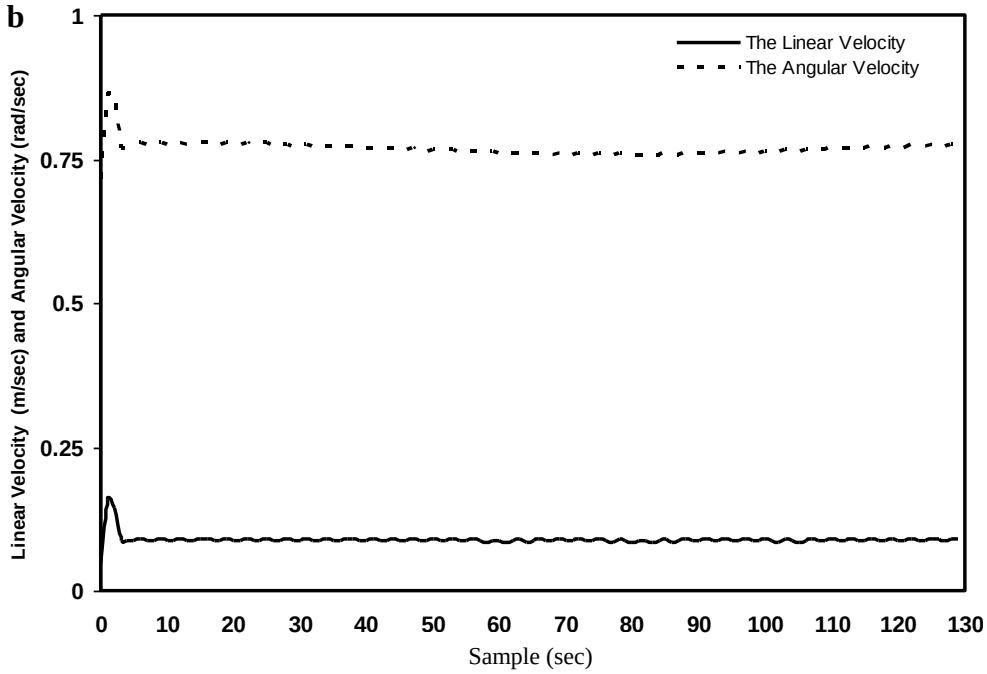
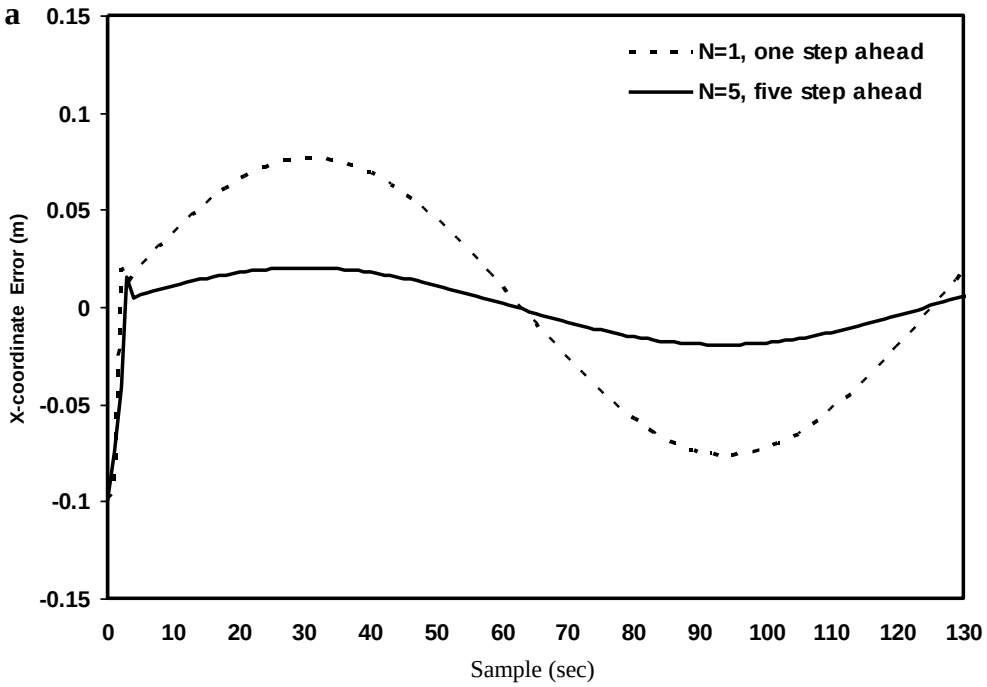


Figure 5.39: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

The effectiveness of the proposed adaptive neural predictive control with predictive optimisation algorithm is clear by showing the convergence of the position and orientation trajectory errors for the robot model motion, as shown in Figures 5.40a, 5.40b and 5.41 respectively for N=1 and 5 steps ahead prediction. The maximum tracking error in the X-coordinate trajectory is equal to ± 0.085m for one step-ahead and for five steps-ahead the X- coordinate error is less than ± 0.015m.



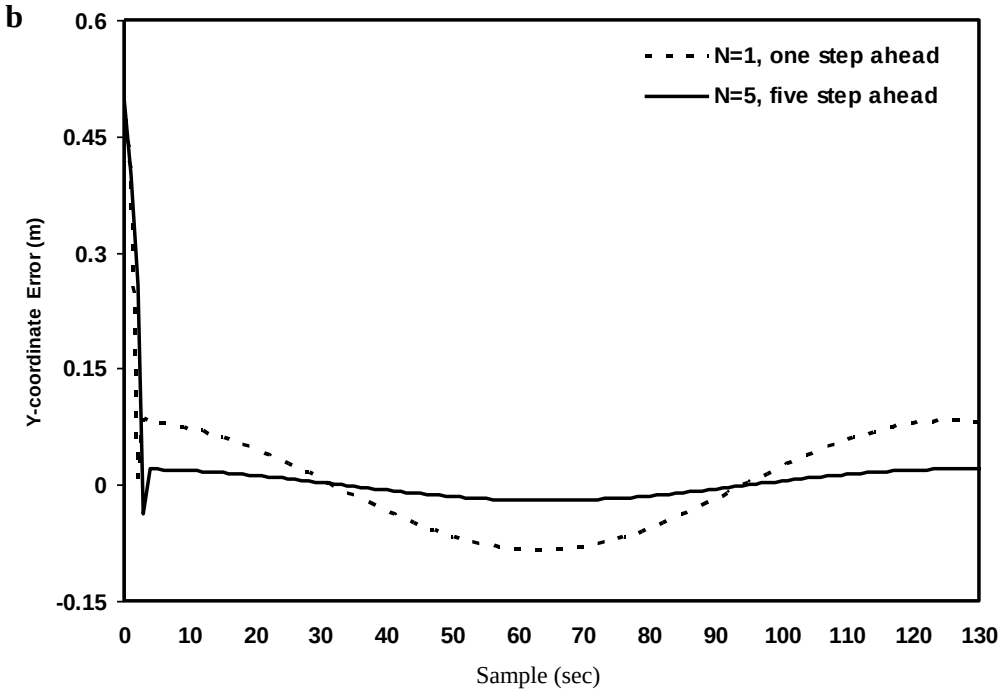


Figure 5.40: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

For Y-coordinate tracking error is equal to ± 0.1m for one step-ahead and for five steps-ahead the error has declined to less than 0.01m. The maximum tracking error in the orientation of the trajectory is equal to -1.65rad for one step-ahead but it is equal to -1.5rad for five steps-ahead.

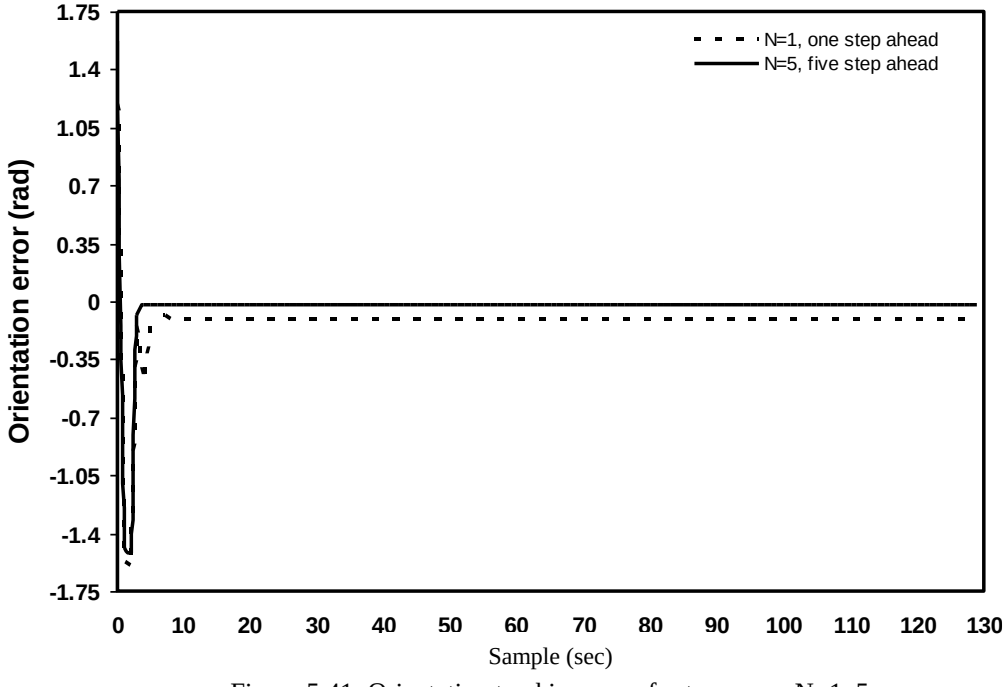


Figure 5.41: Orientation tracking error for two cases N=1, 5.

The parameters of the position and orientation nonlinear PID controller $(Kp_\gamma, Ki_\gamma, Kd_\gamma)$ are demonstrated, as shown in Figures 5.42 and 5.43.

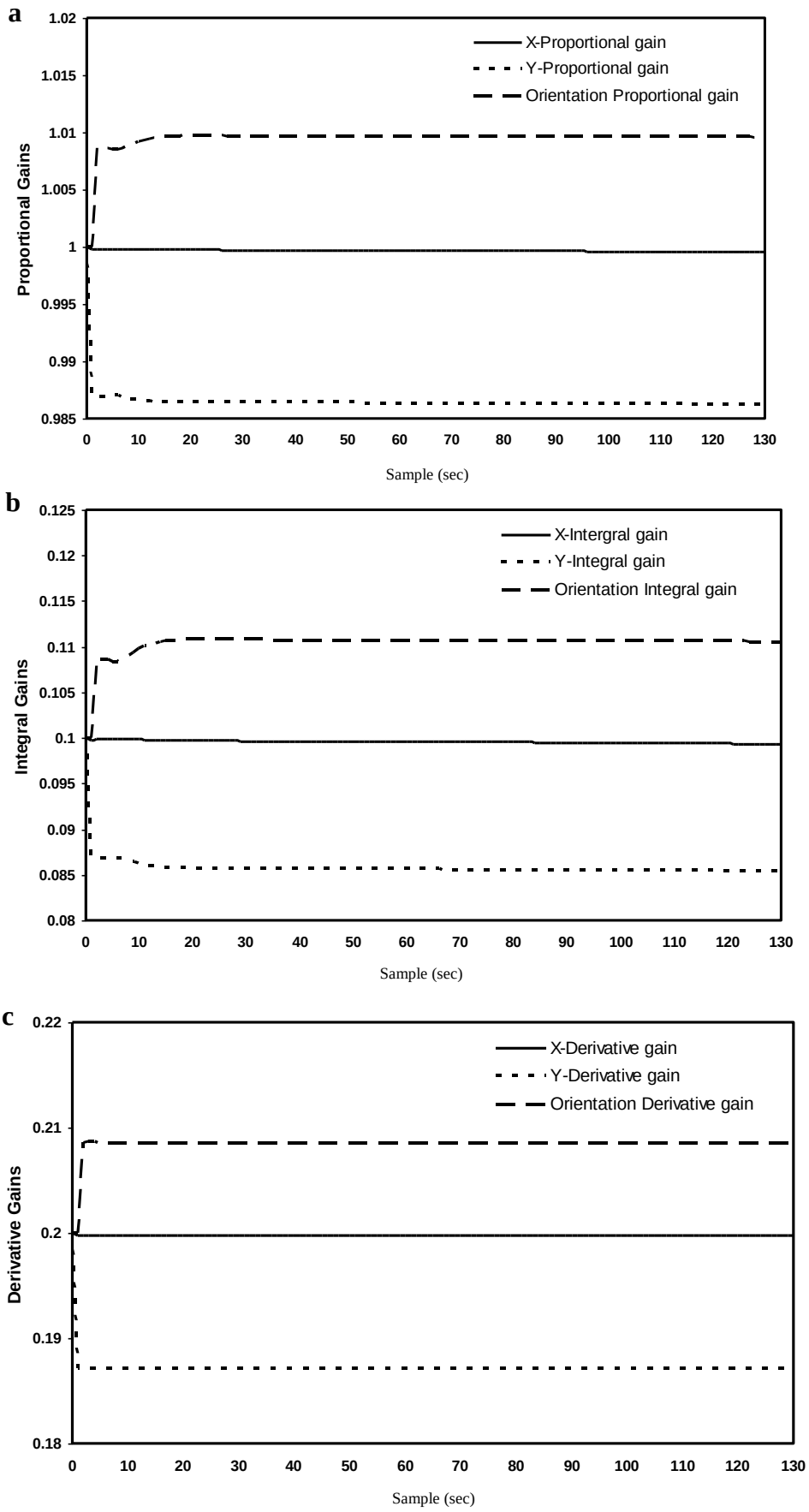


Figure 5.42: The position and orientation PID controller parameters for N=1: (a) proportion gain; (b) integral gain; (c) derivative gain.

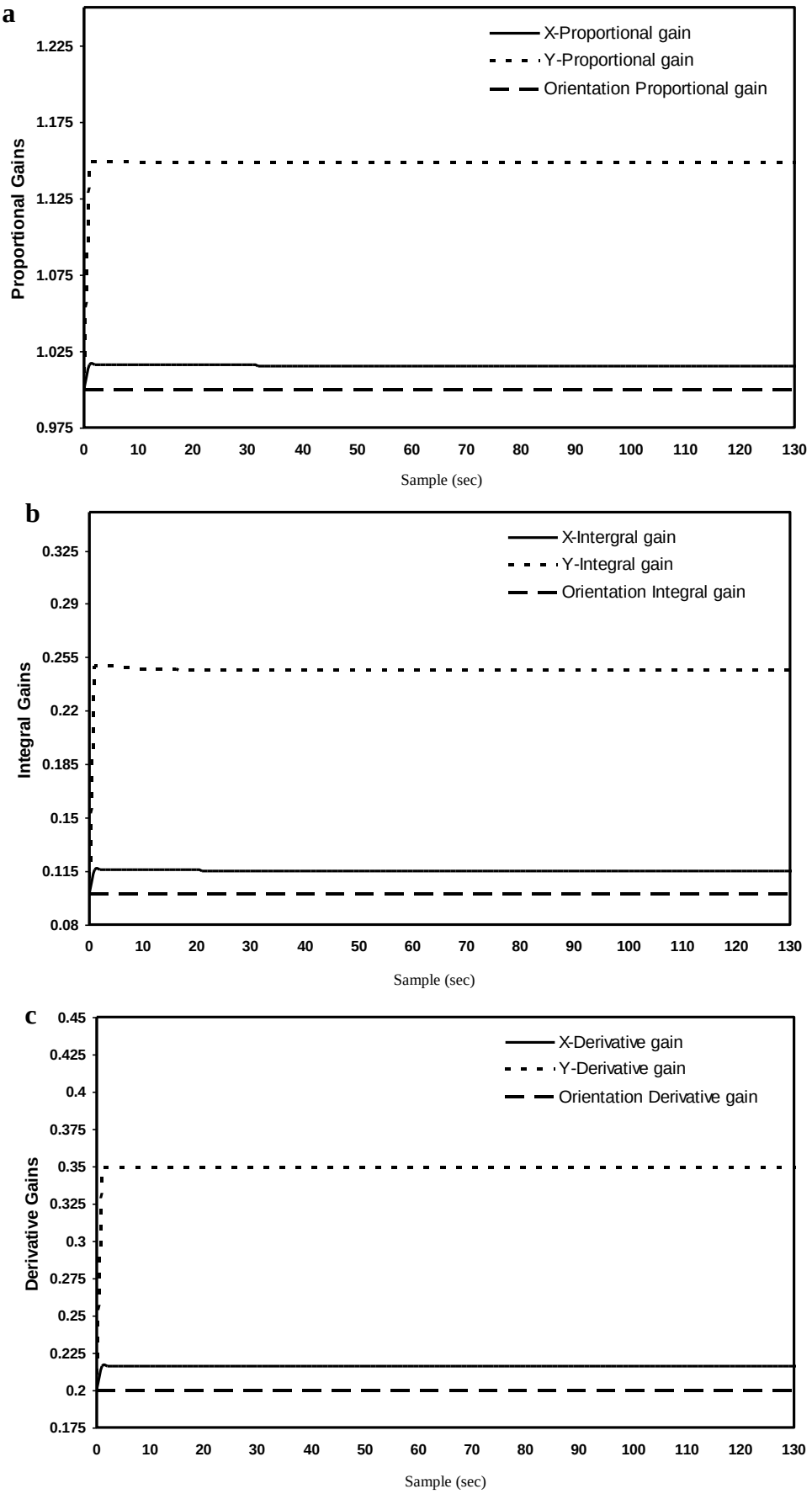


Figure 5.43: The position and orientation PID controller parameters for N=5: (a) proportion gain; (b) integral gain; (c) derivative gain.

For the third control methodology the nonlinear inverse dynamic neural feedback controller is used, which has five parameters $(k_x, k_y, k_\theta, Q, R)$ for tuning effort. The robot trajectory tracking obtained by the proposed adaptive neural predictive controller is shown in Figures 5.44a and 5.44b.

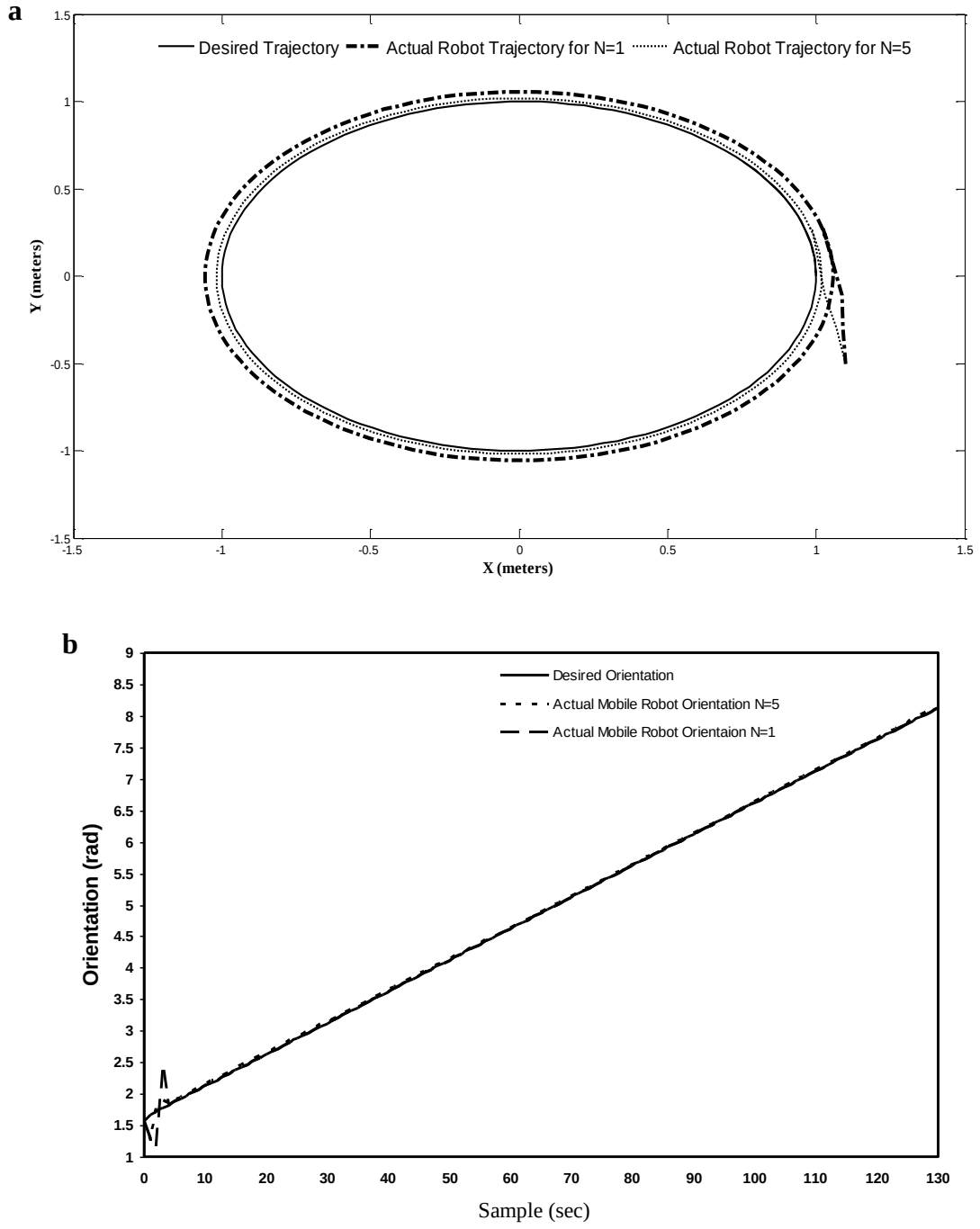


Figure 5.44: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired circular trajectory; (b) actual orientation of mobile robot and desired orientation.

These Figures demonstrate very good position and orientation tracking performance for five steps-ahead predictions in comparison with results of one step-ahead. In spite of the

existence of bounded disturbances the adaptive learning and robustness of neural controller with optimisation algorithm show that these disturbances have a negligible effect. The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small smooth values of the control input torques for right and left wheels without sharp spikes. The actions described in Figures 5.45a and 5.45b show that smaller power is required to drive the DC motors of the mobile robot model

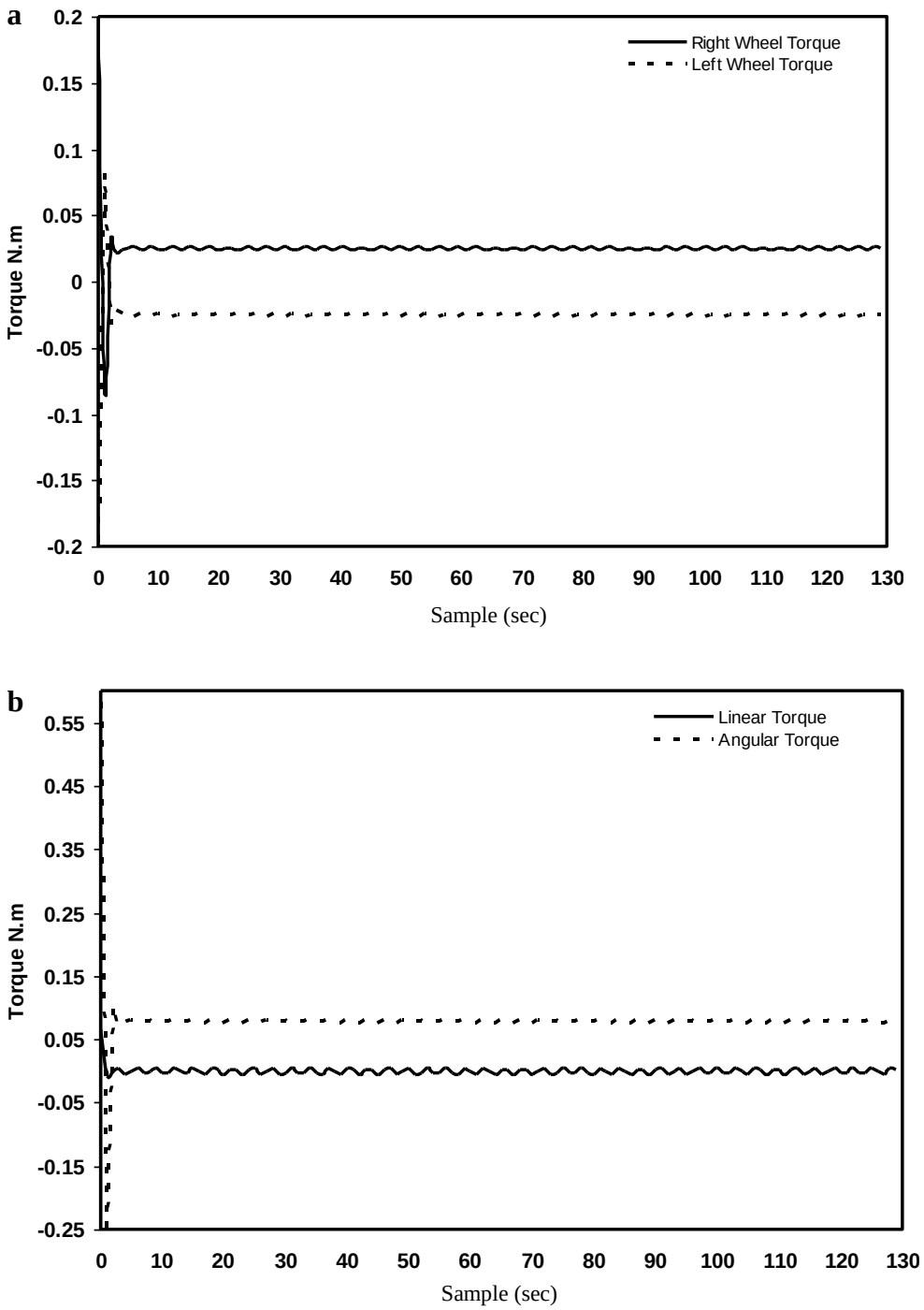


Figure 5-45: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the simulation results for right and left wheels were smooth values without sharp spikes, as shown in Figure 5.46a. The mean linear velocity of the mobile robot is equal to 0.0975m/sec, and the rate of the angular velocity is equal to 0.75rad/sec, as shown in Figure 5.46b.

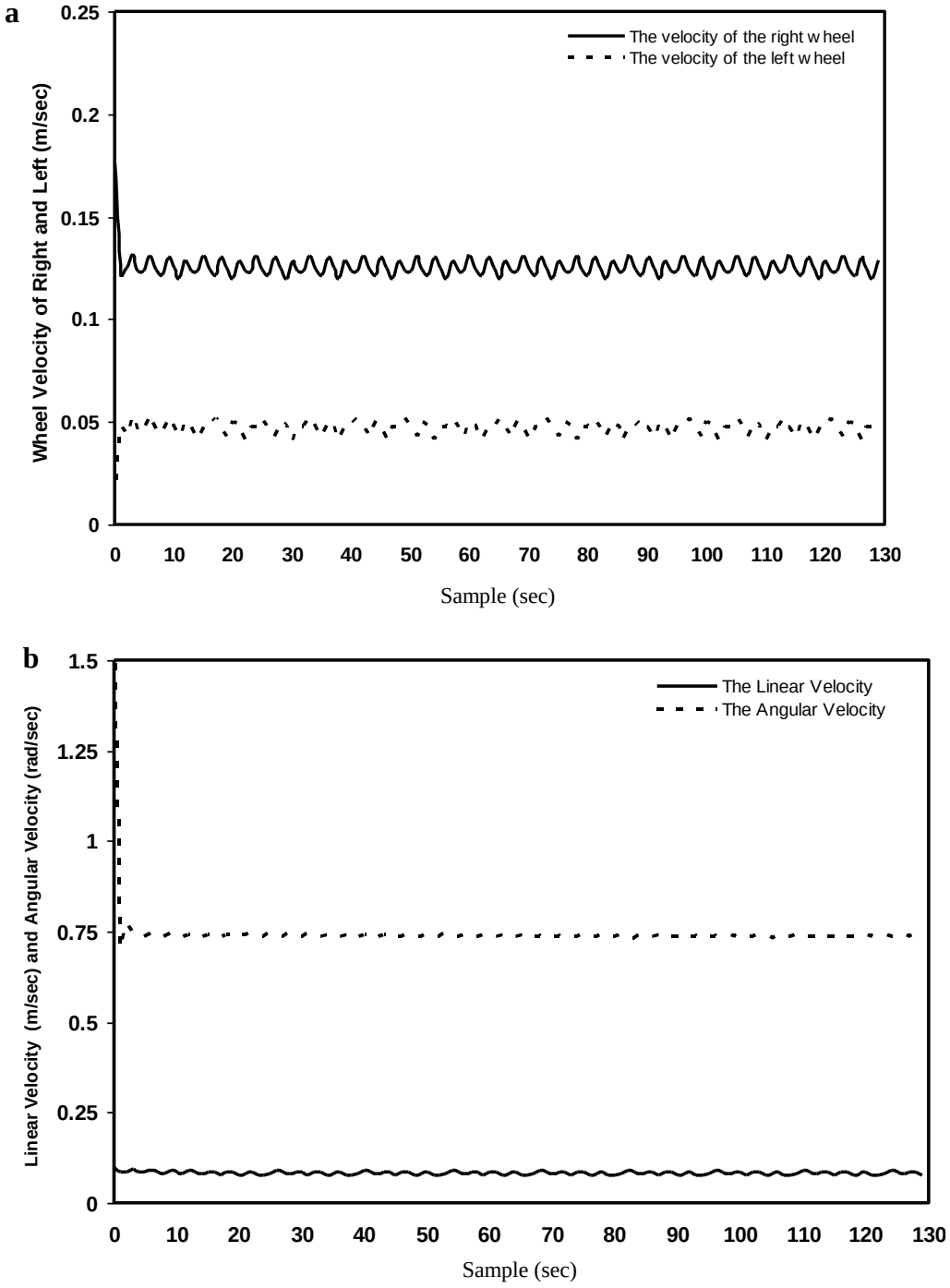


Figure 5.46: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

The effectiveness of the proposed adaptive neural predictive control with predictive optimisation algorithm is clear by showing the convergence of the pose trajectory error for the robot model motion for N=1 and 5 steps-ahead, as shown in Figures 5.47a and 5.47b for position tracking error and Figure 5.48 for orientation tracking error. The maximum tracking error in the X-coordinate trajectory is equal to ± 0.05m for one step-ahead while for the five steps-ahead the X-coordinate error is equal to ± 0.025m.

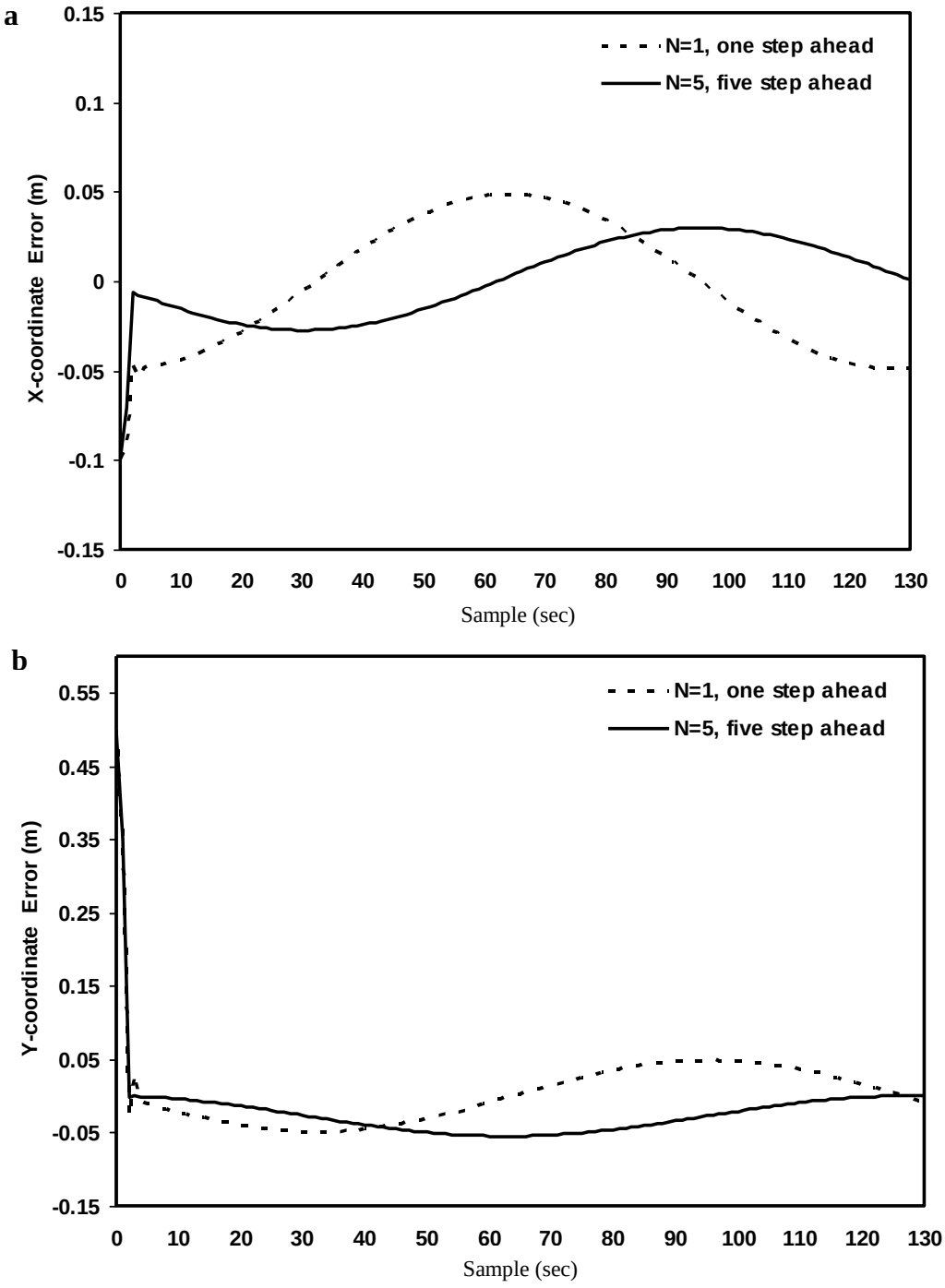


Figure 5.47: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

For Y-coordinate tracking error is equal to ± 0.05m for one step-ahead and for five steps-ahead the error has declined to less than 0.05m. The maximum tracking error in the orientation of the trajectory is equal to ± 0.5rad for one step-ahead but it is equal to 0.35rad for five steps-ahead.

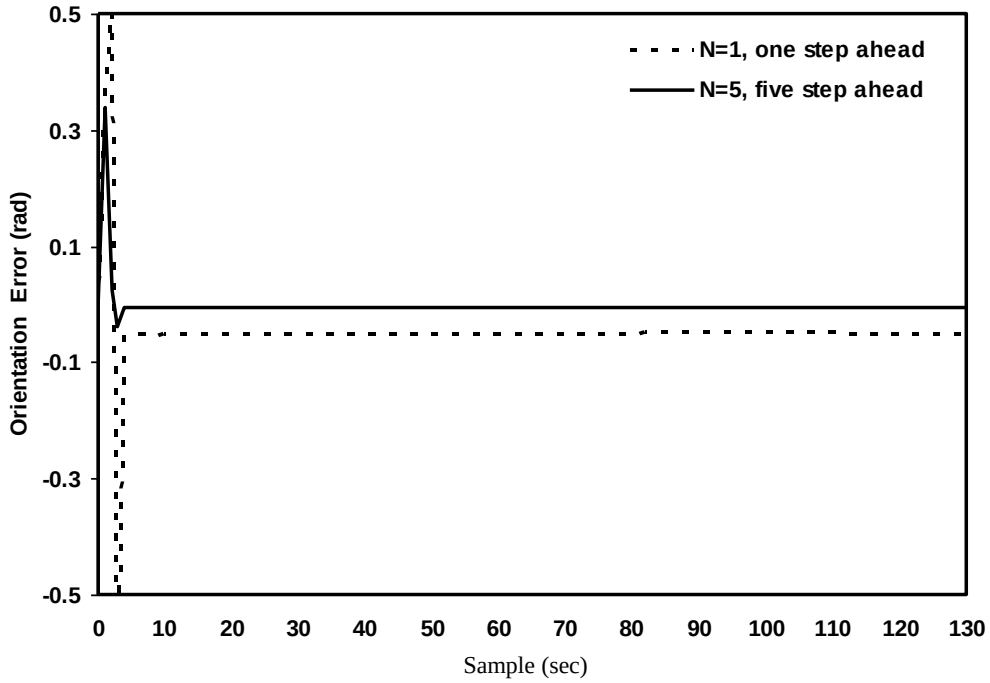
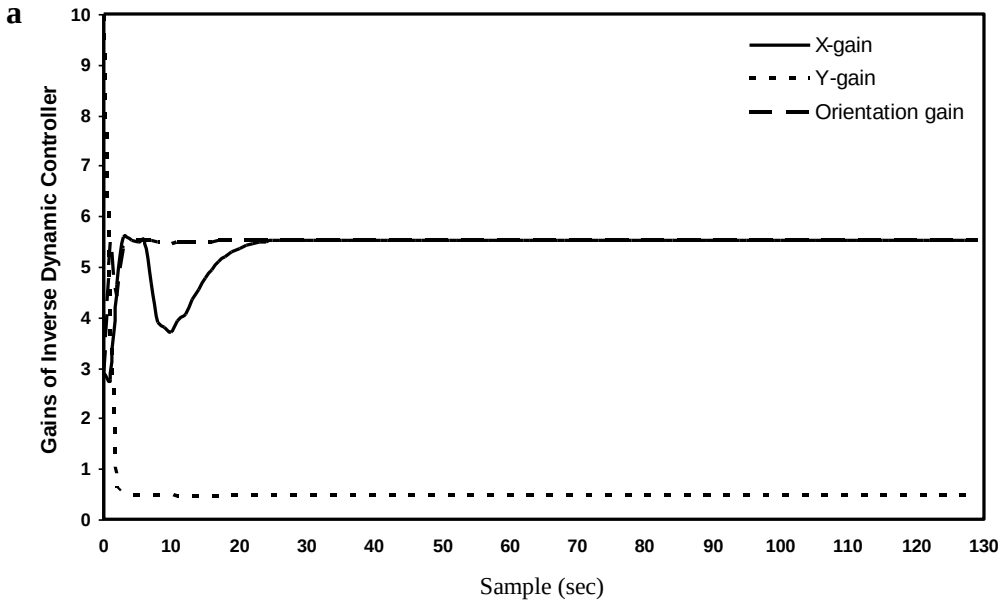


Figure 5.48: Orientation tracking error for two cases N=1, 5.

By using optimised-auto-tuning with posture neural identifier of the mobile robot model, the parameters of the inverse dynamic controller (k_x, k_y, k_θ) can be demonstrated, as shown in Figures 5.49a and 5.49b.



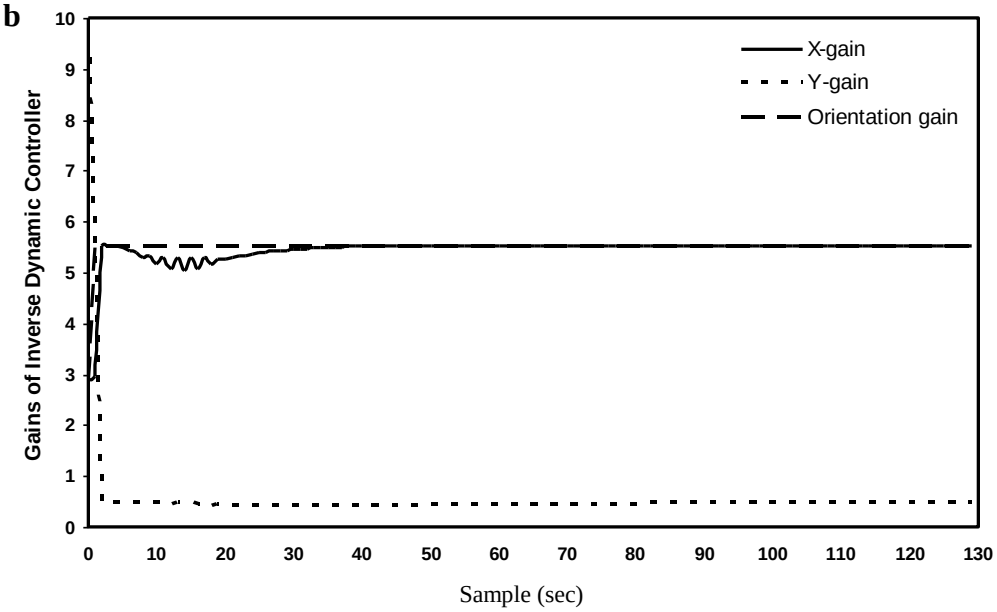


Figure 5.49: The inverse dynamic controller parameters: (a) N=1; (b) N=5.

As shown in Table 5.2, the mean-square error for each component of the state error $(q_r - q) = (e_x, e_y, e_\theta)$ is used for the five steps-ahead predictive control of each methodology.

Table 5.2: MSE of each methodology for the five steps-ahead prediction of desired circular trajectory

$MSE(q_r - q) = (e_x, e_y, e_\theta)$	Methodology1	Methodology2	Methodology3
MSE of X-coordinate	0.000889	0.000284	0.000653
MSE of Y-coordinate	0.000211	0.000201	0.000209
MSE of Orientation	0.00248	0.00130	0.00169

As can be seen from the results of Table 5.2, the second control methodology has the smallest MSE of the X-coordinate and the MSE of Y-coordinate is less than for the first and third control methodologies. Therefore, it can be stated that second control methodology was the best methodology for the desired circular trajectory.

5.3.3. Case Study 3

Simulation is also carried out for desired square trajectory, which has explicitly non-continuous gradient for verification the capability of the proposed adaptive neural predictive controller performance. The mobile robot model starts from the initial position and orientation $q(0) = [0, -0.1, 0]$ as its initial posture with the same external disturbance used in case one and case two, and uses the same stages of the proposed controller in the

first methodology, but the best value of Q parameter is equal to 0.02 and best value of R parameter is equal to 1 for minimising the position and orientation mean square errors and obtaining more optimised control action. Figure 5.50a shows that the mobile robot tracks the square desired trajectory quite accurately, but at the end of one side of the square there is a sudden increase in position errors of the mobile robot against the desired trajectory at the corners of the square, because the desired orientation angle changes suddenly at each corner, as shown in Figure 5.50b, therefore the mobile robot takes a slow turn.

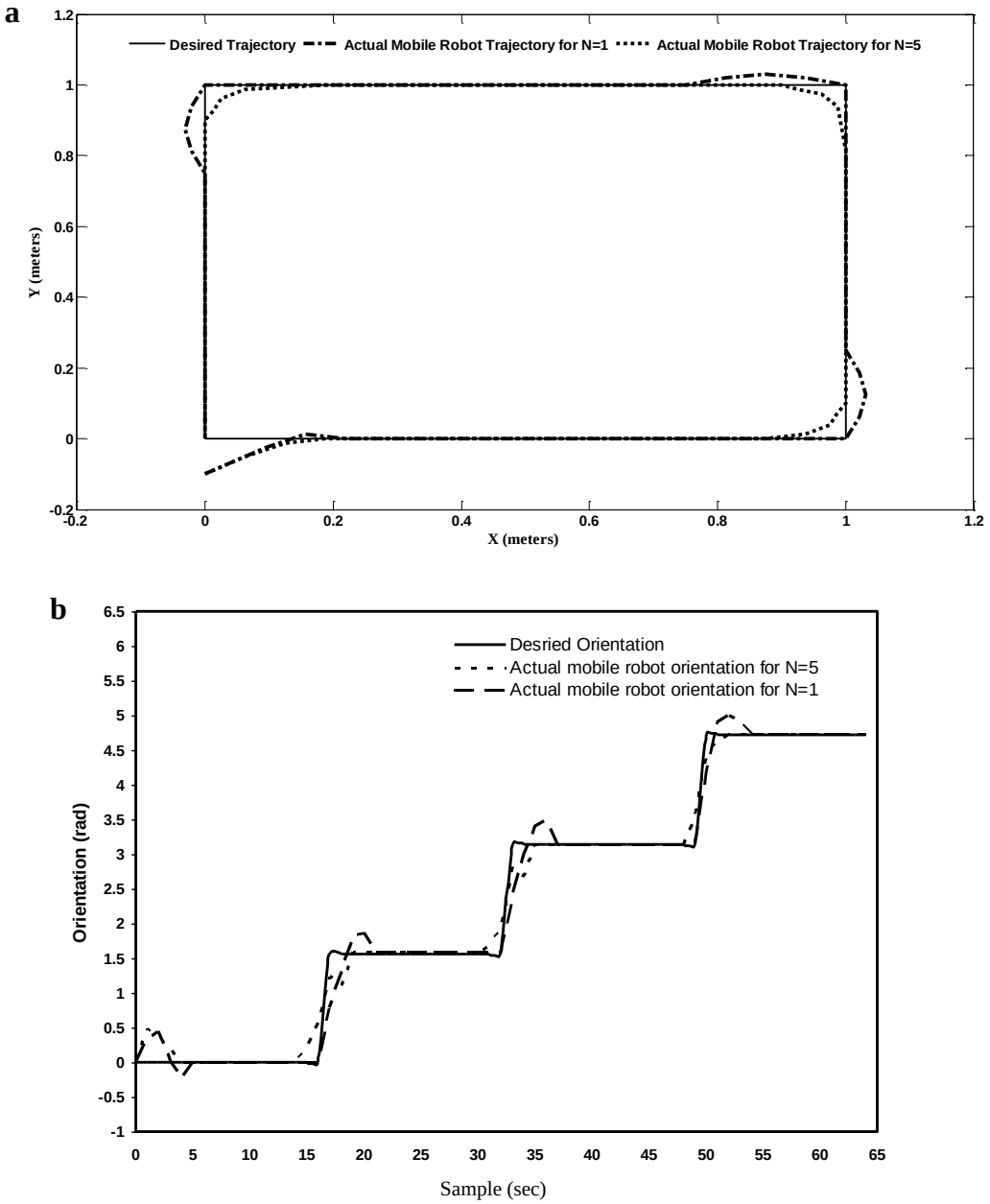


Figure 5.50: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired square trajectory; (b) actual orientation of mobile robot and desired orientation.

In Figures 5.51a and 5.51b, the behaviour of the control action torques for right and left wheels is smooth values with sharp spikes, when the desired orientation angle changes suddenly at each corner.

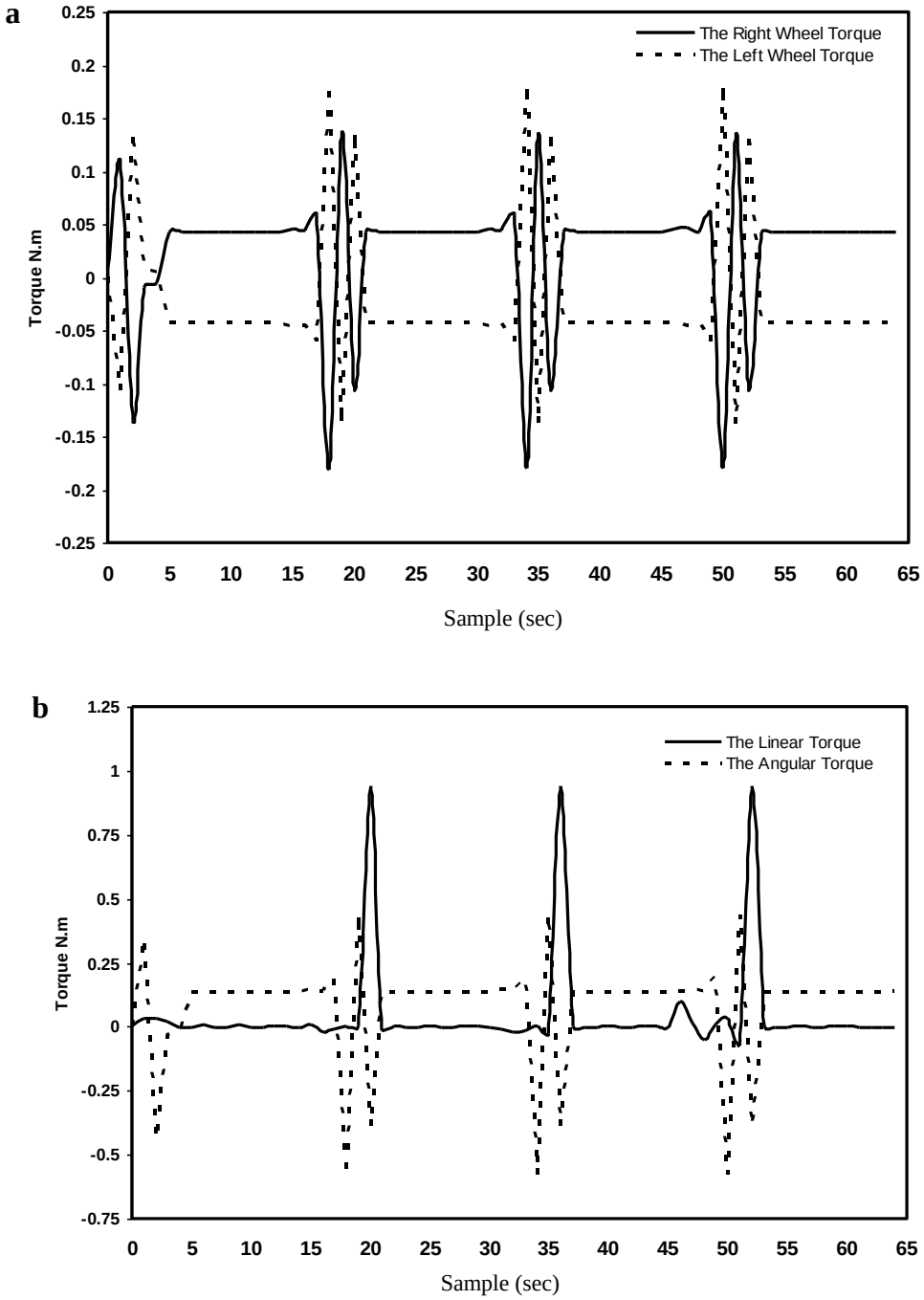


Figure 5.51: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the right wheel and the left wheel were smooth values without sharp spikes, as shown in Figure 5.52a. The mean linear velocity of the mobile robot is equal to 0.125m/sec, and the maximum peak of the angular velocity is equal to 1.25rad/sec, as shown in Figure 5.52b.

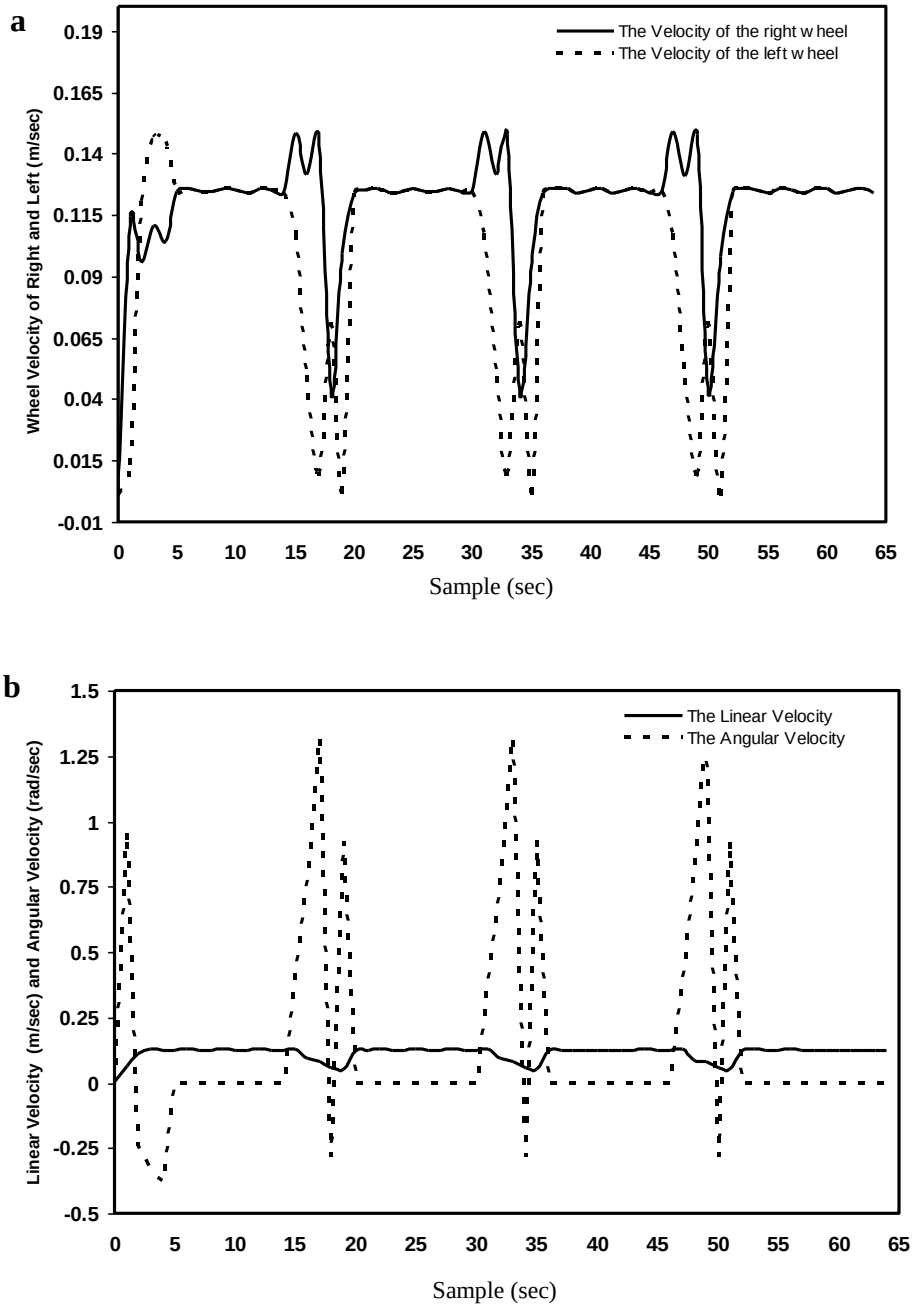


Figure 5.52: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

In addition to that, the robot tracks the right side of the square desired trajectory and the tracking errors sharply drop to small values as shown in Figures 5.53a and 5.53b, for

position tracking errors. The maximum X-coordinate error in the square trajectory is equal to ±0.03m for one step-ahead predication while for five steps-ahead prediction the maximum error in the X-coordinate is equal to 0.04m.

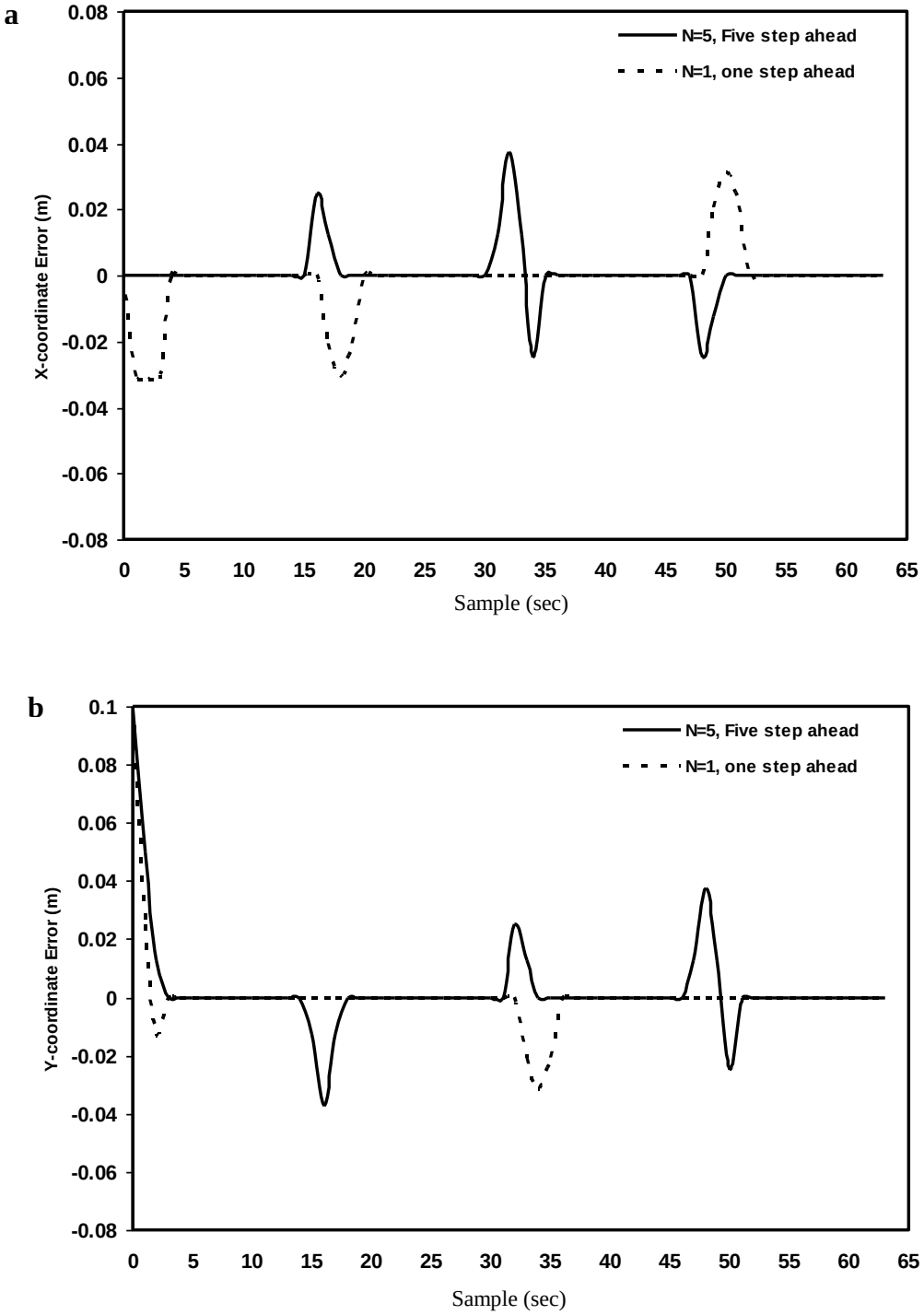


Figure 5.53: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

The maximum Y-coordinate error in the square trajectory is equal to -0.04m for one step-ahead predication while for five steps-ahead prediction the maximum error in the Y-

coordinate is equal to ±0.04m. Along any one side of the square, the desired orientation angle is constant, therefore the orientation error is equal to zero, but at the end of one side of the square trajectory the desired orientation angle changes suddenly, therefore the orientation error of the mobile robot against the desired trajectory at the corners of the square is increasing, as shown in Figure 5.54.

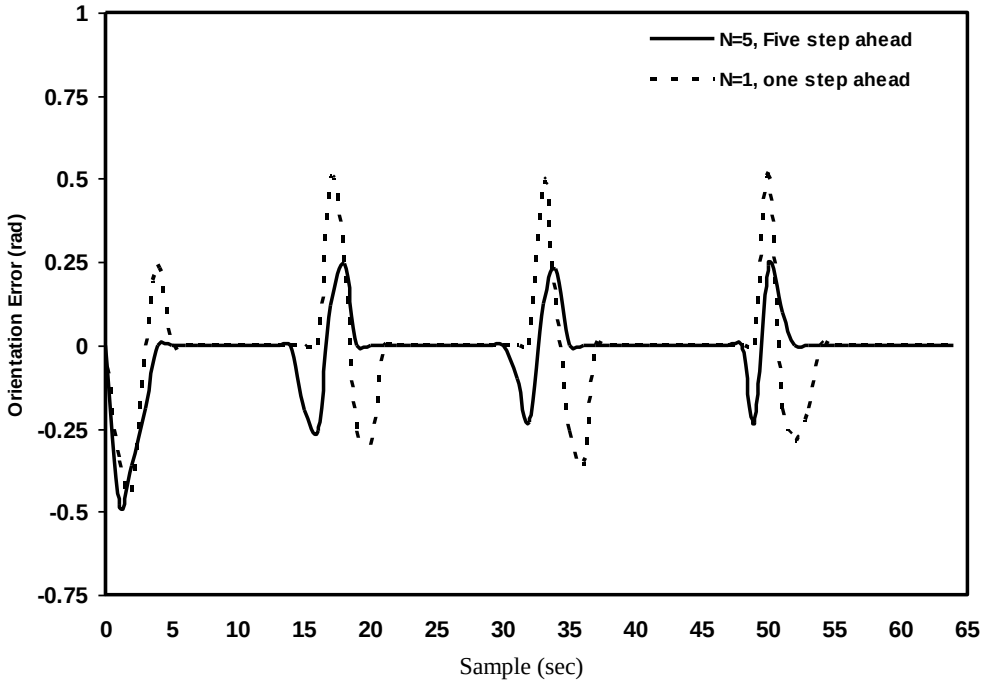
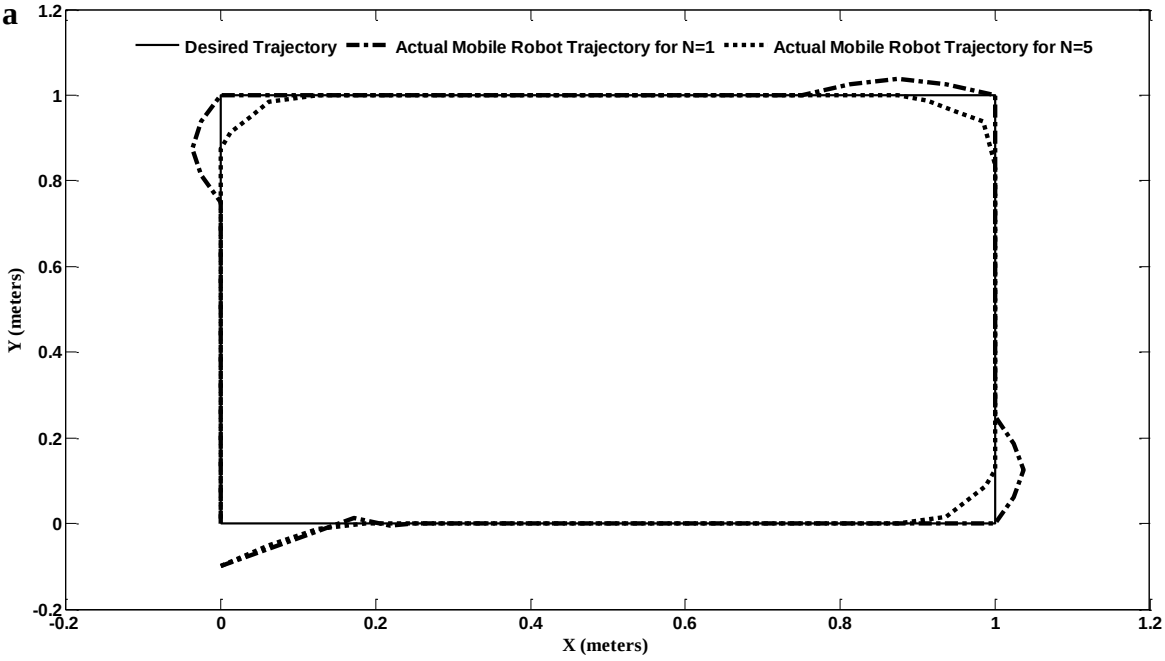


Figure 5.54b: Orientation tracking error for N= 1 and 5.

For the second control methodology of the adaptive neural predictive controller, the mobile robot trajectory tracking is obtained, as shown in Figures 5.55a and 5.55b.

a



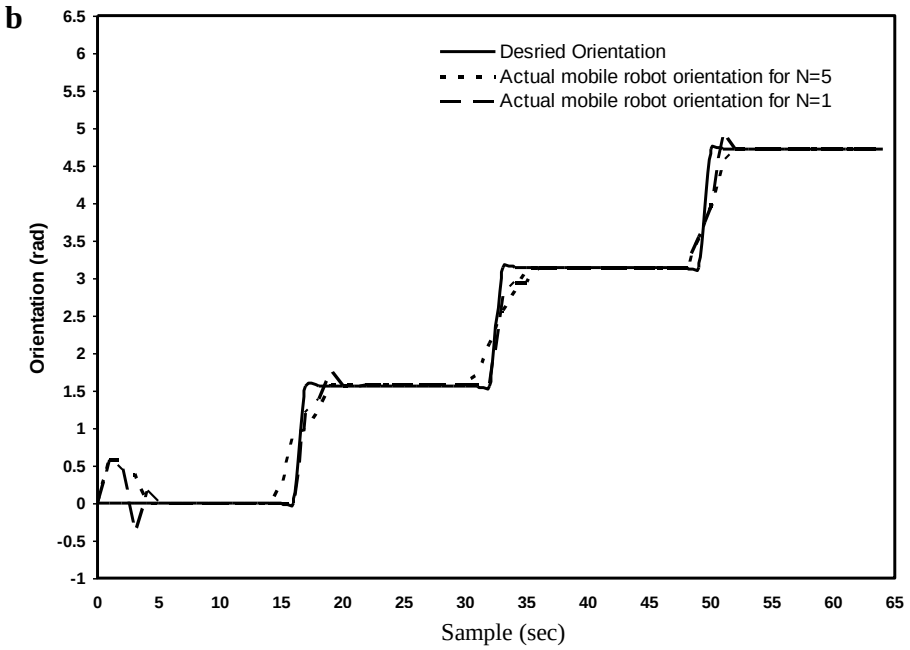
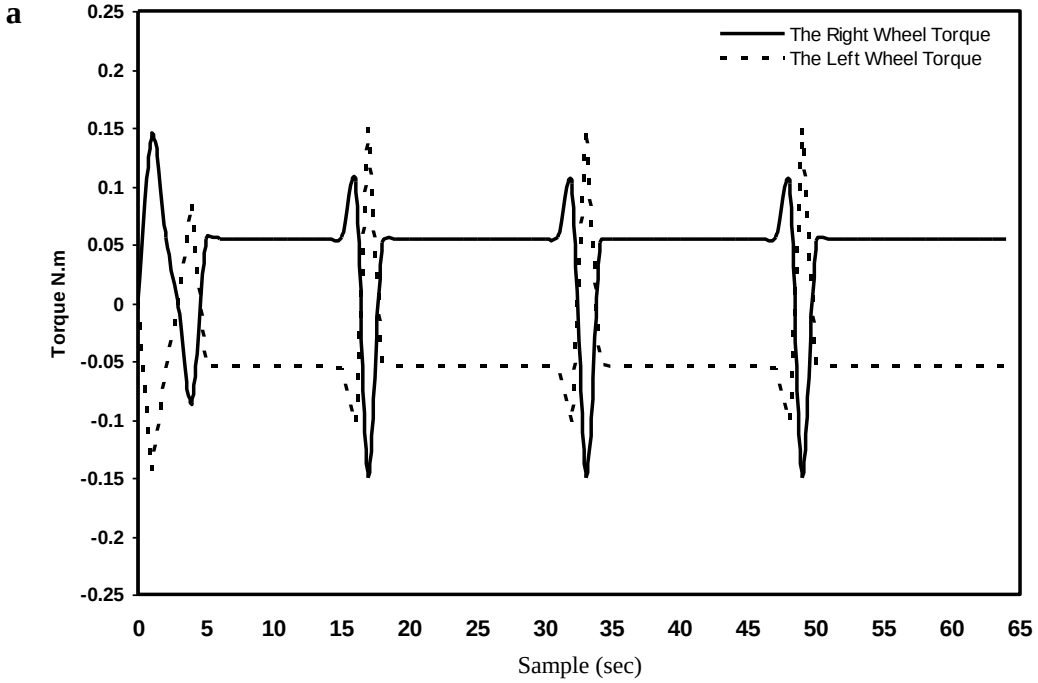


Figure 5.55: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired square trajectory; (b) actual orientation of mobile robot and desired orientation.

These Figures demonstrate excellent position and orientation tracking performance for the five steps-ahead prediction in comparison with results of one step-ahead. In spite of the existence of bounded disturbances, the adaptive learning and robustness of neural controller with optimisation algorithm shows a small effect of these disturbances. The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small smooth values of the control input torques for right and left wheels without sharp spikes.



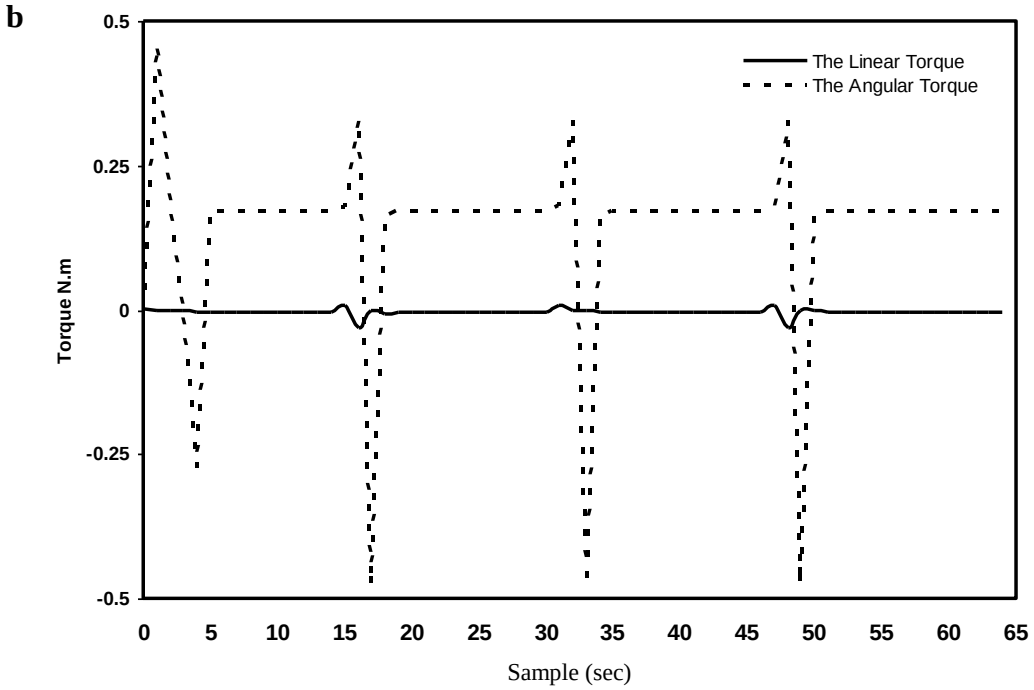
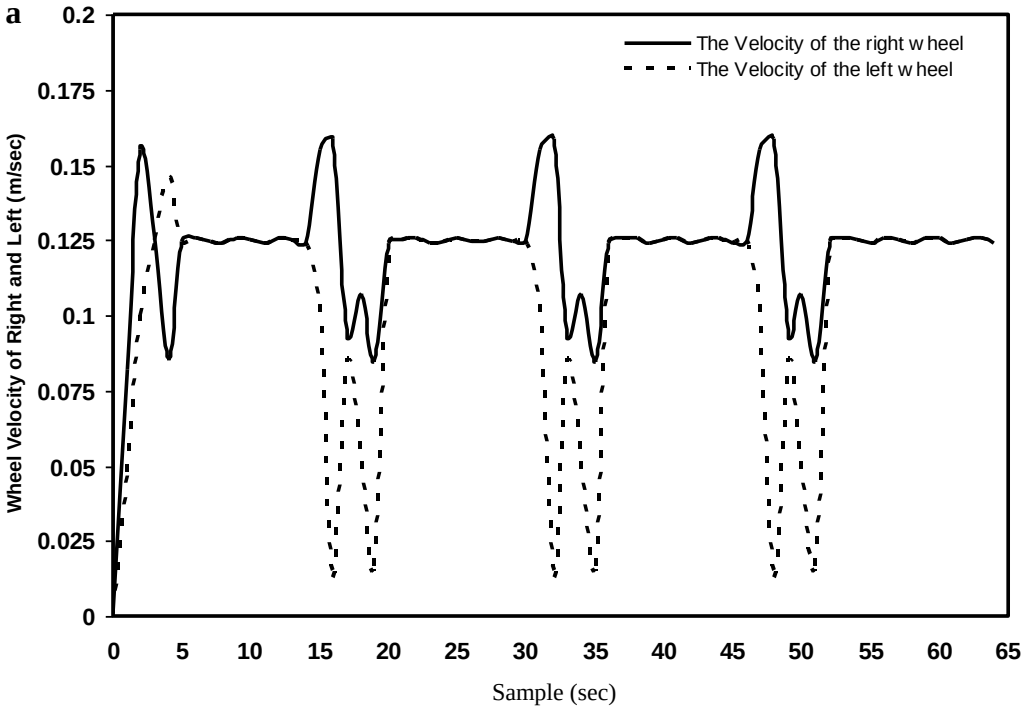


Figure 5.56: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small smooth values of the control input torques for right and left wheels without sharp spikes. The actions described in Figures 5.56a and 5.56b show that smaller power is required to drive the DC motors of the mobile robot model. The velocities of the simulation results for right and left wheels were smooth values without sharp spikes are shown in Figures 5.57a and 5.57b.



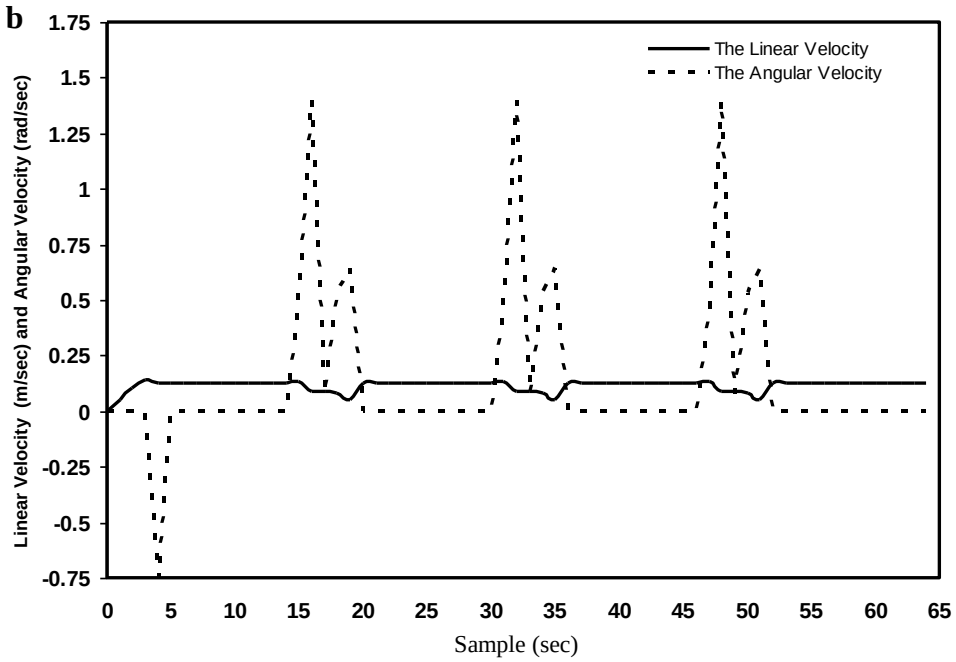
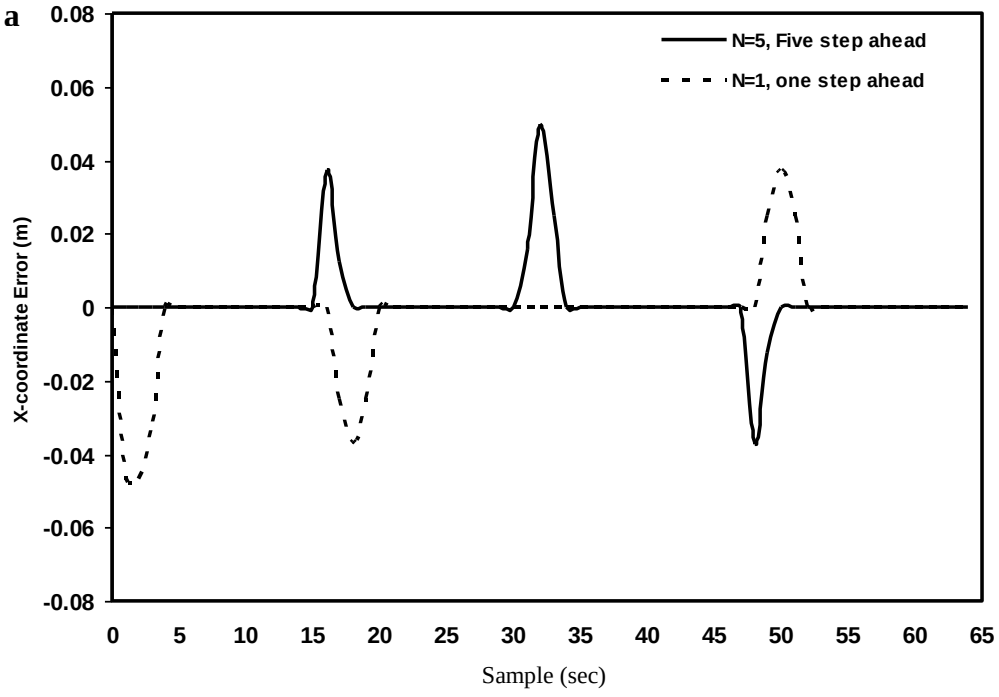


Figure 5.57: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

As shown in Figure 5.58a, the maximum X-coordinate error in the square trajectory is equal to -0.045m for one step-ahead predication while for five steps-ahead prediction the maximum error in the X-coordinate is equal to 0.05m. The maximum Y-coordinate error in the square trajectory is equal to -0.04m for one step-ahead predication while for five steps-ahead prediction the maximum error in the Y-coordinate is equal to ±0.05m, as shown in Figure 5.58b.



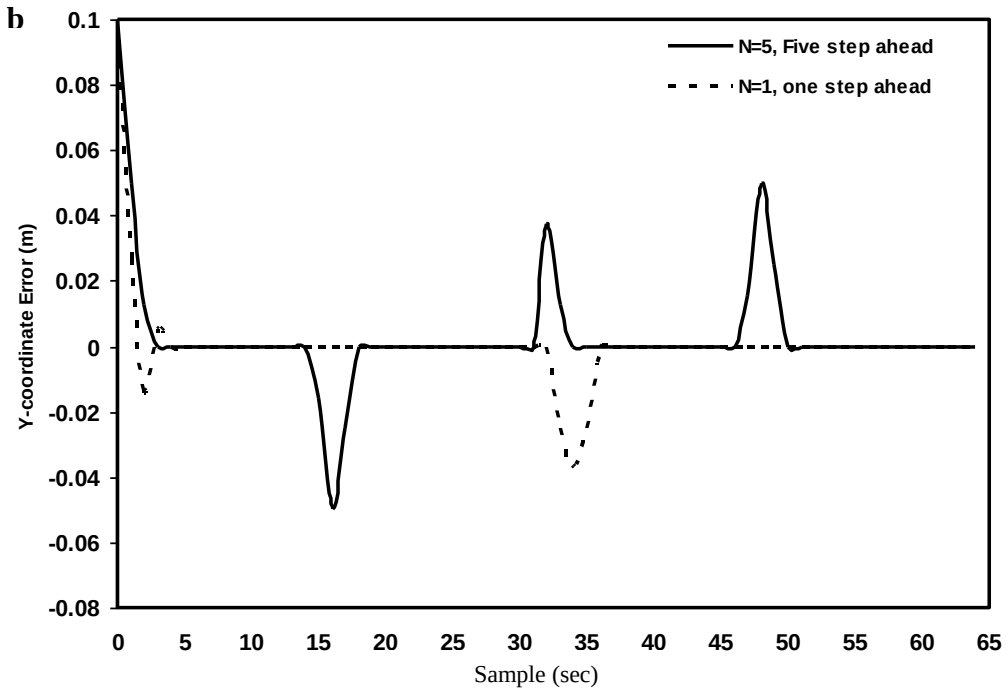


Figure 5.58: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

The orientation errors for one step-ahead is equal to 0.35rad and for five steps-ahead the error is equal to 0.25rad, as shown in Figure 5.59.

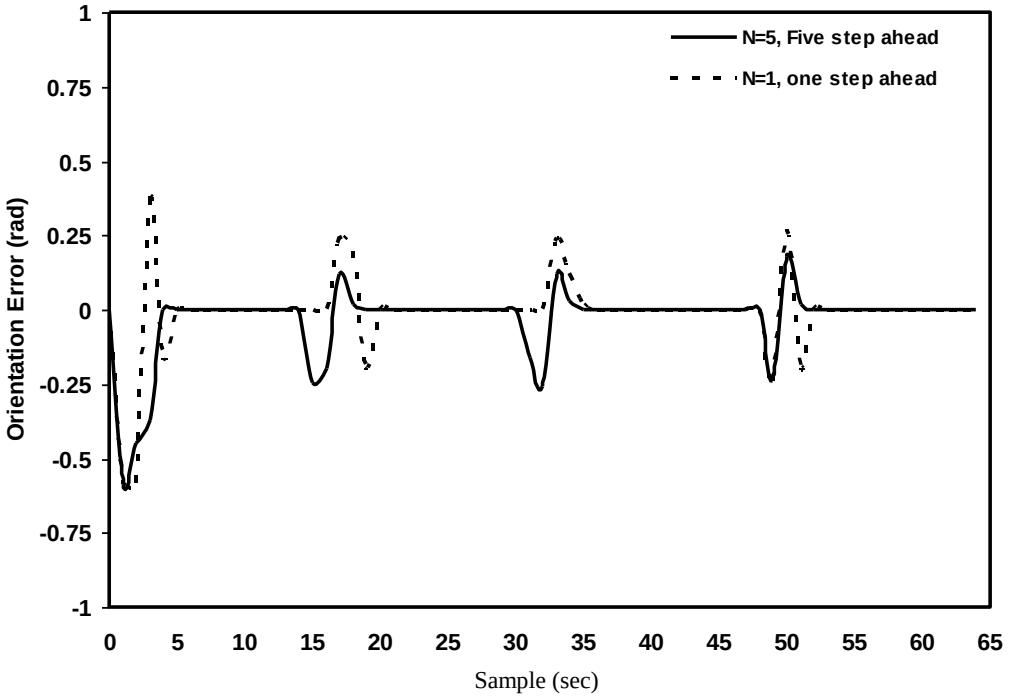


Figure 5.59: Orientation tracking error for N= 1 and 5.

The initial values of $(Kp_\gamma, Ki_\gamma, Kd_\gamma)$ are chosen from trial-and-error tuning method, giving 1, 0.1 and 0.2 respectively, and the parameters of the position and orientation

nonlinear PID controller can be demonstrated, as shown in Figures 5.60 and 5.61 for one and five steps-ahead respectively.

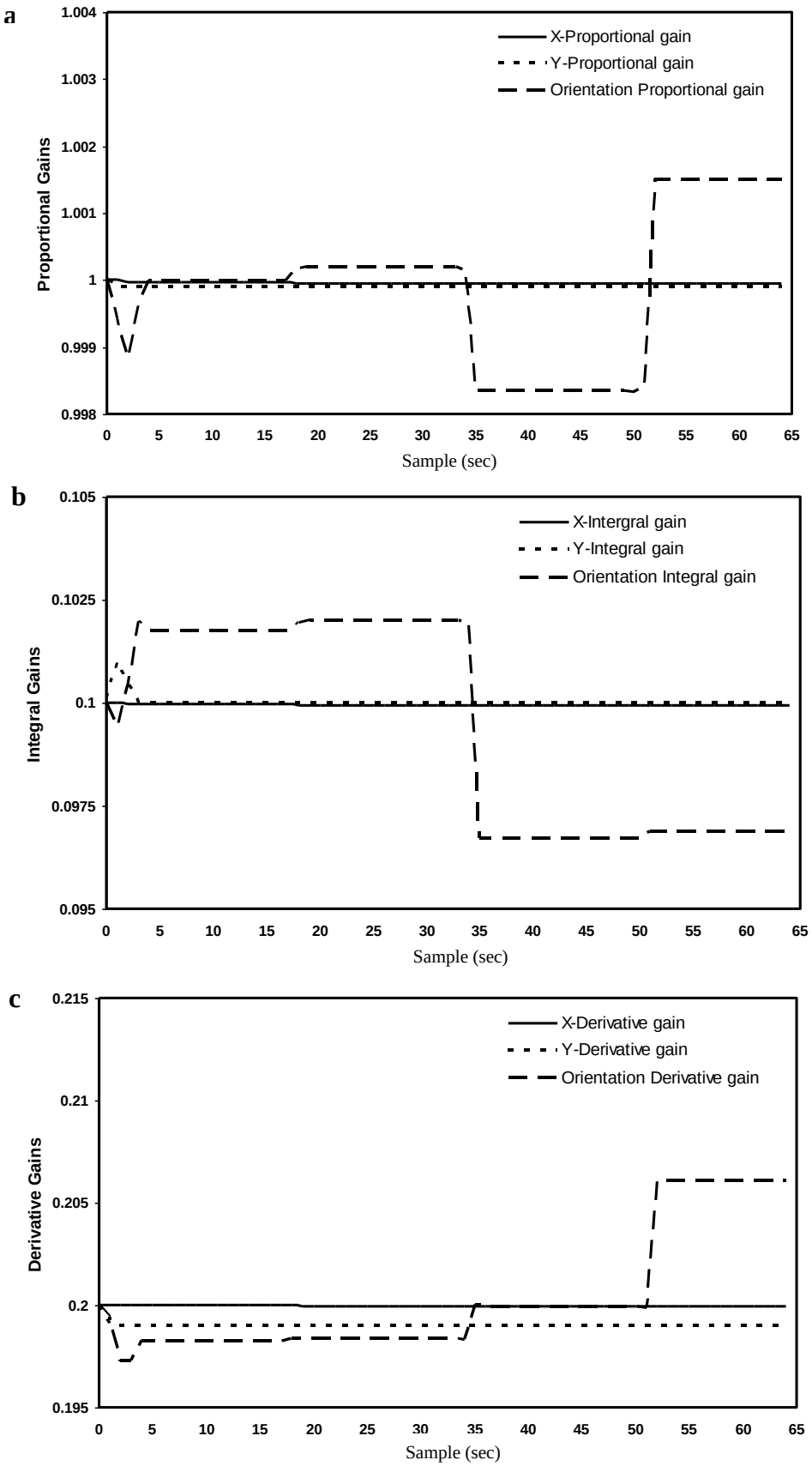


Figure 5.60: The position and orientation PID controller parameters for N=1: (a) proportion gain; (b) integral gain; (c) derivative gain.

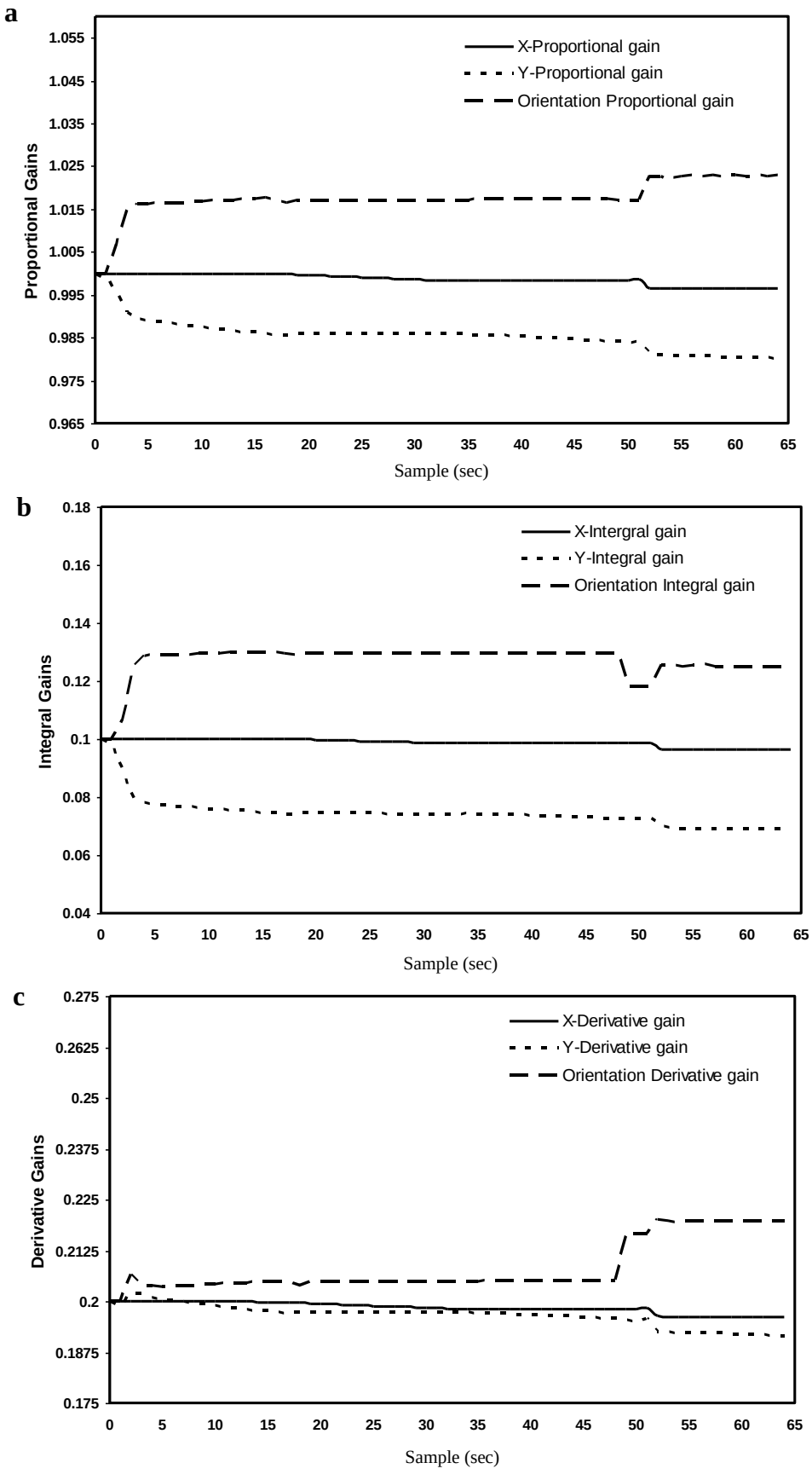


Figure 5.61: The position and orientation PID controller parameters for N=5: (a) proportion gain; (b) integral gain; (c) derivative gain.

The third control methodology of the adaptive neural controller is nonlinear inverse dynamic neural feedback controller with five parameters $(k_x, k_y, k_\theta, Q, R)$ for tuning effort. The same values of Q equal to 0.02 and R equal to 1 have been used in the first and second methodology in order to make the comparison results of the simulation under identical conditions to specify the differences and similarities.

The mobile robot trajectory tracking obtained by the proposed adaptive neural predictive controller are shown in Figures 5.62a and 5.62b.

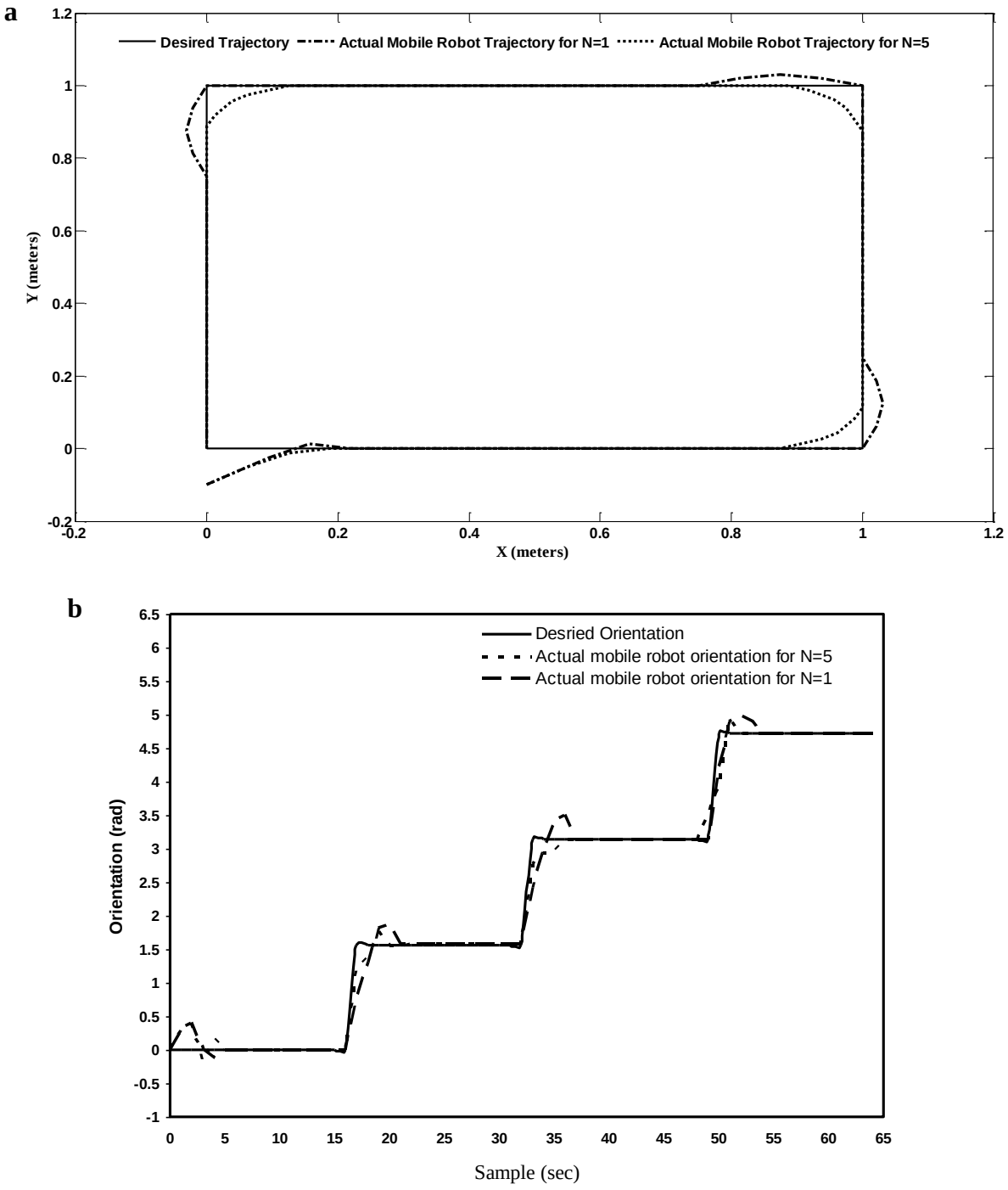


Figure 5.62: Simulation results for one and five steps-ahead predictive: (a) actual trajectory of mobile robot and desired square trajectory; (b) actual orientation of mobile robot and desired orientation.

These Figures demonstrate excellent position and orientation tracking performance for five steps-ahead predictions in comparison with results of one step-ahead. In spite of the existence of bounded disturbances the adaptive learning and robustness of neural controller with optimisation algorithm show small effect of these disturbances. The simulation results demonstrated the effectiveness of the proposed controller by showing its ability to generate small smooth values of the control input torques for right and left wheels without sharp spikes. The actions described in Figures 5.63a and 5.63b show that smaller power is required to drive the DC motors of the mobile robot model.

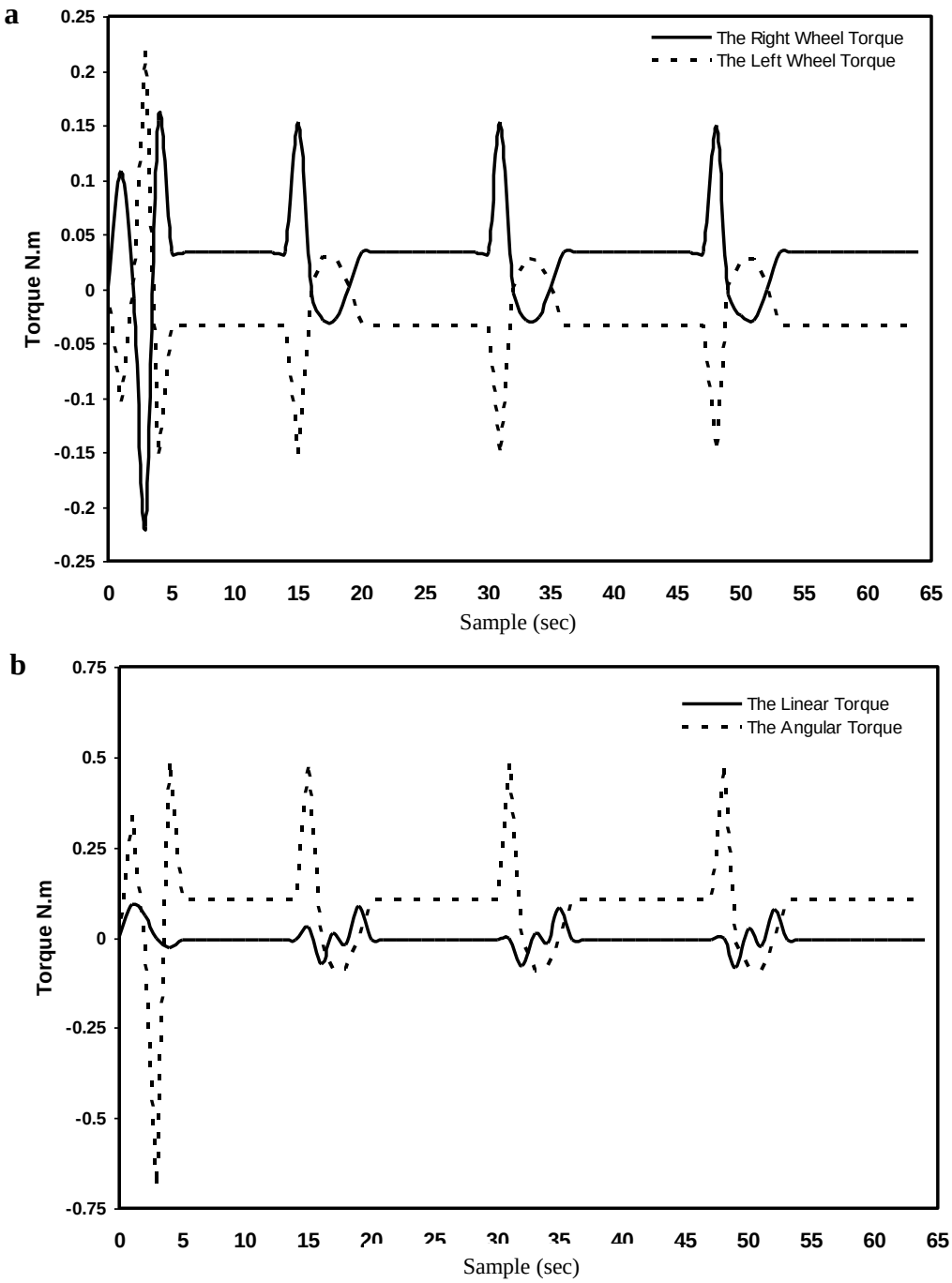


Figure 5.63: Torque action for N=5:
(a) the right and left wheel torque; (b) the linear and angular torque.

The velocities of the simulation results for right and left wheels were smooth values without sharp spikes are shown in Figures 5.64a and 5.64b.

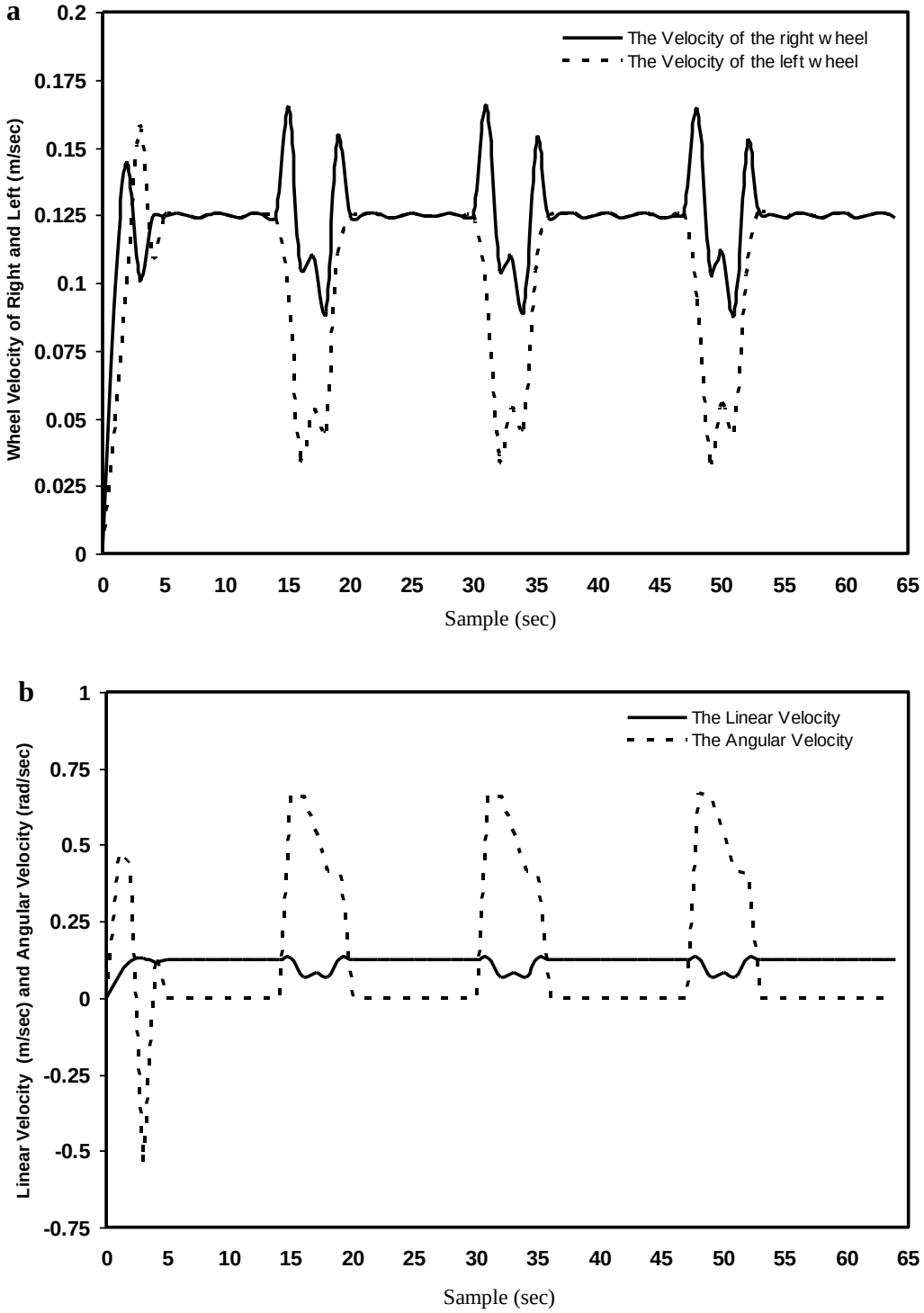


Figure 5.64: Velocity action for N=5:
(a) the right and left wheel velocity; (b) the linear and angular velocity.

The effectiveness of the proposed adaptive neural predictive control with predictive optimisation algorithm is clear by showing the convergence of the position trajectory error for the robot model motion for N=1 and 5 steps-ahead, as shown in Figures 5.65a and 5.65b, while the orientation tracking error is shown in Figure 5.66. The maximum X-coordinate error in the square trajectory is equal to ± 0.03m for one step-ahead predication while for five steps-ahead predictions the maximum error in the X-coordinate is equal to ± 0.04m.

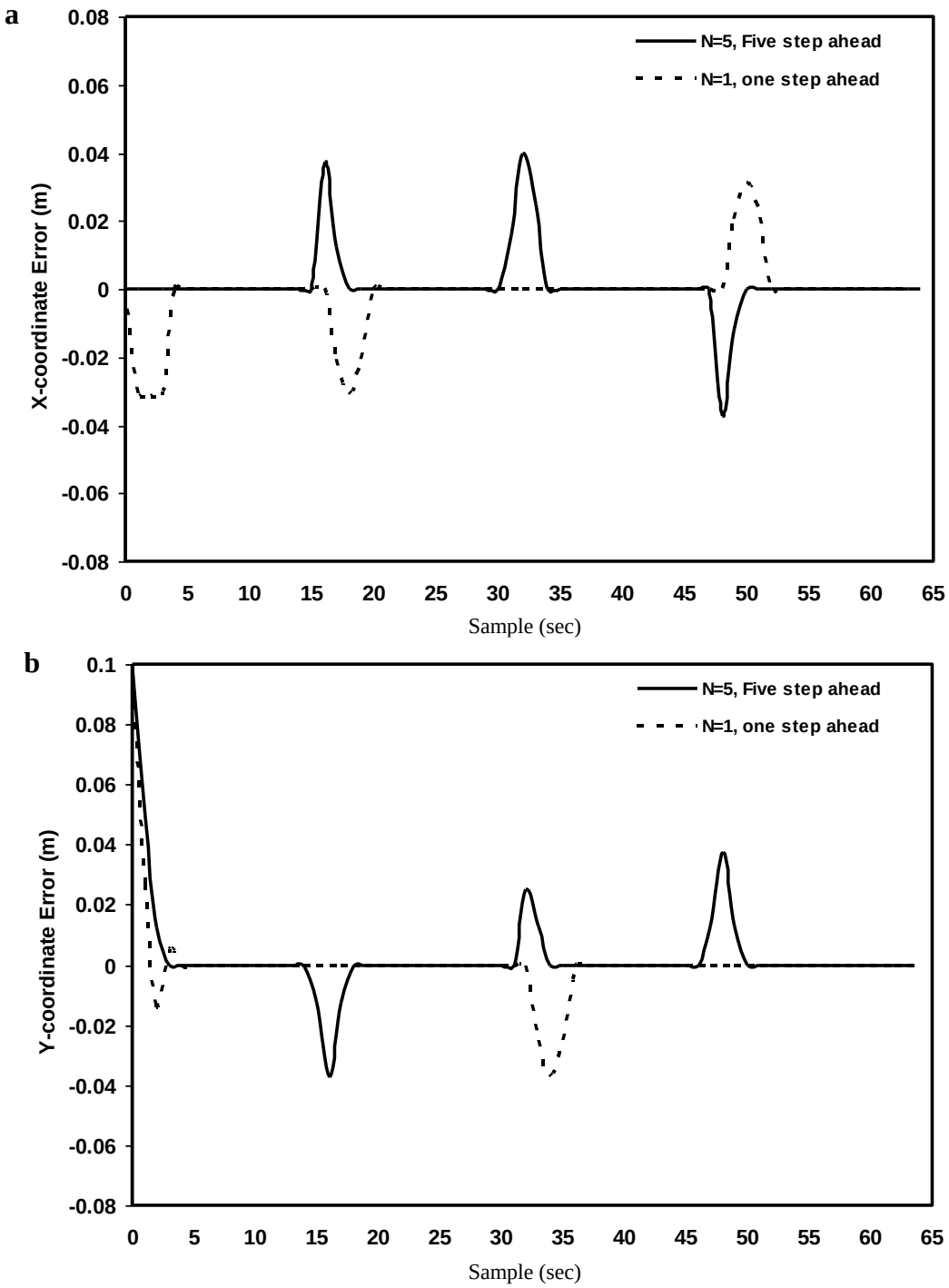


Figure 5.65: Position tracking error for two cases N=1 and 5:
(a) in X-coordinate; (b) in Y-coordinate.

The maximum Y-coordinate error in the square trajectory is equal to -0.04m for one step-ahead predication while for five steps-ahead predictions the maximum error in the Y-coordinate is equal to ±0.04m. The orientation tracking error for one step-ahead is equal to 0.5rad and for five steps-ahead the error is equal to -0.25rad.

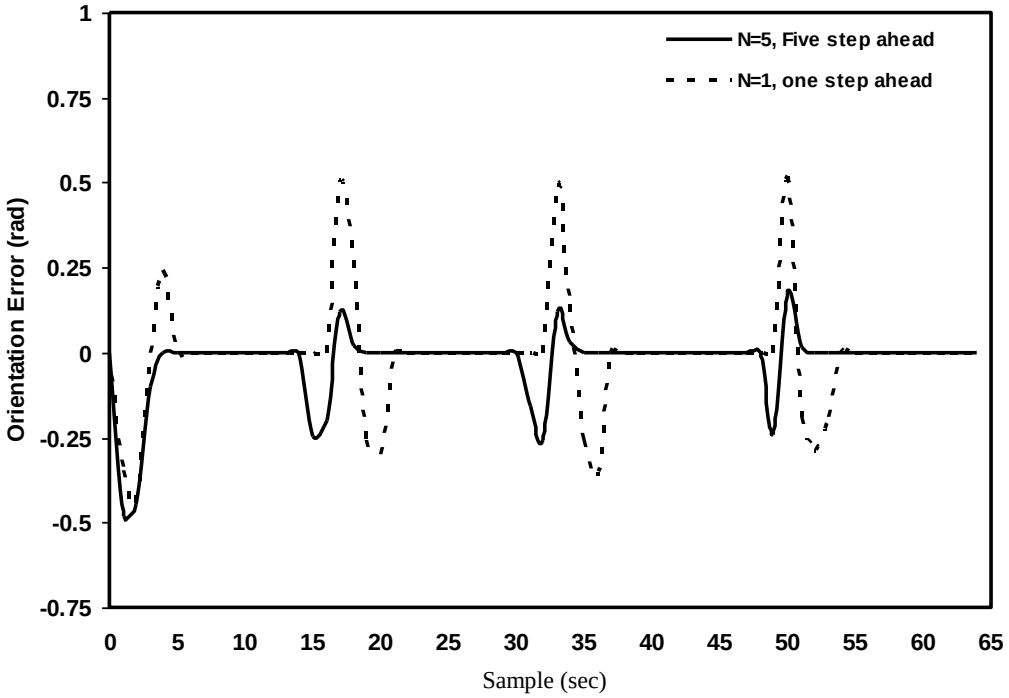
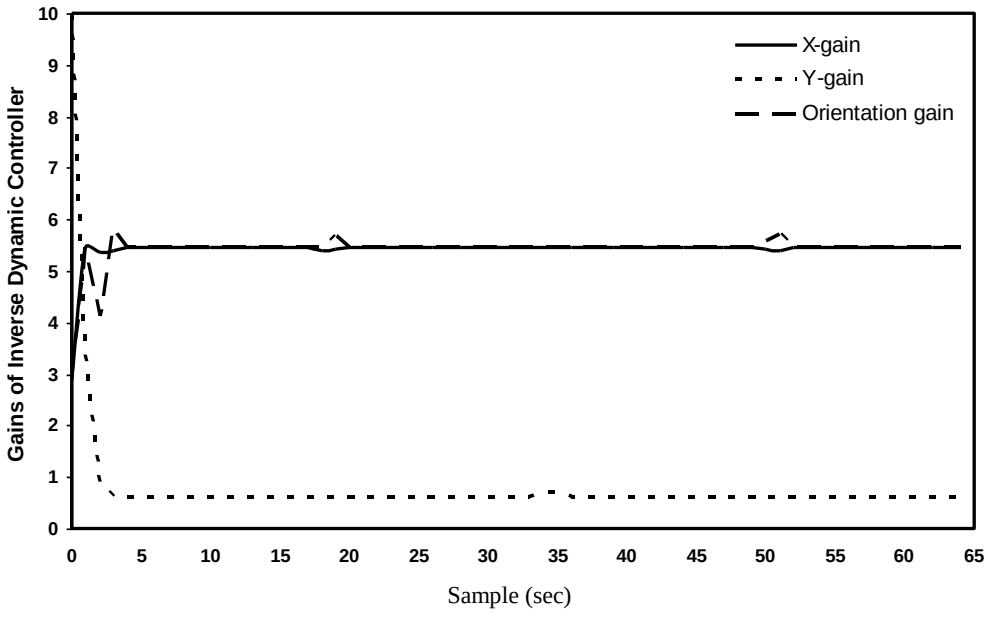


Figure 5.66: Orientation tracking error for N= 1 and 5.

The optimised-auto-tuning with identification of the mobile robot model are used for tuning the parameters of the inverse dynamic controller (k_x, k_y, k_θ) as shown in Figures 5.67a and 5.67b.

a



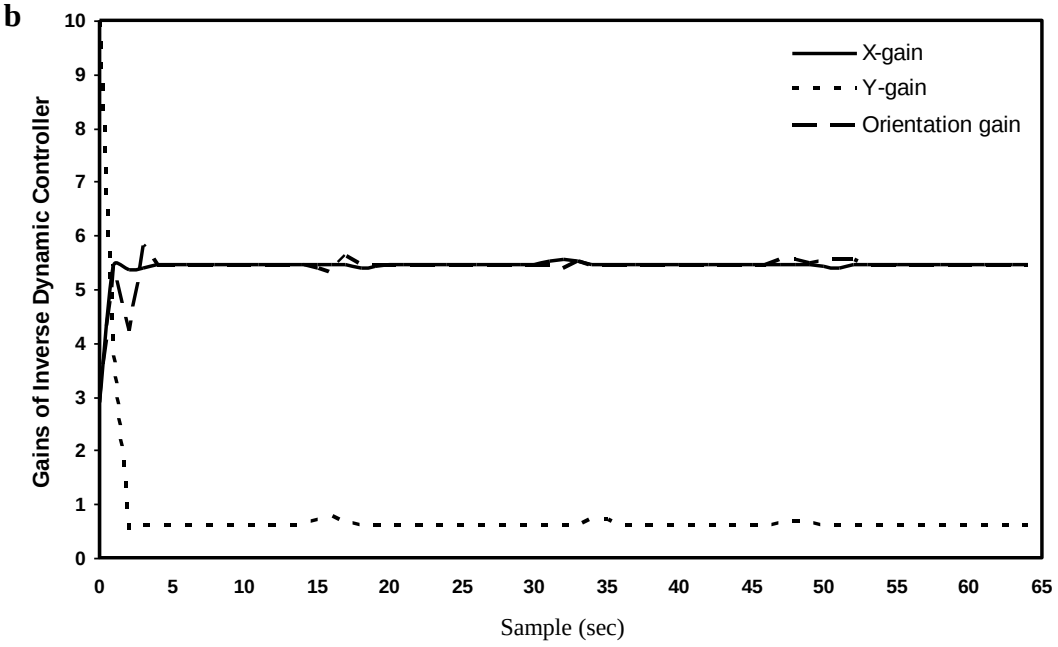


Figure 5.67: The inverse dynamic controller parameters: (a) N=1; (b) N=5.

The mean-square errors for each component of the state error $(q_r - q) = (e_x, e_y, e_\theta)$ for the five steps-ahead predictive control of each methodology are shown in Table 5.3.

Table 5.3: MSE of each methodology for the five steps-ahead-prediction of desired square trajectory

$MSE(q_r - q) = (e_x, e_y, e_\theta)$	Methodology1	Methodology2	Methodology3
MSE of X-coordinate	0.000433	0.000871	0.000240
MSE of Y-coordinate	0.000215	0.000255	0.000209
MSE of Orientation	0.00961	0.00914	0.00745

From the results of Table 5.3, the third control methodology has the smallest MSE of the X-coordinate and Y-coordinate compared to the first and second methodologies, and the MSE of orientation for third control methodology is less than for the first and second control methodologies.

5.5. Summary

This chapter gives an overview of the basic concepts of the adaptive neural predictive control structure, consisting of posture neural identifier, a feedforward neural controller and feedback predictive controller with three kinds of control predictive methodologies. The first methodology is the feedback neural controller, the second is the nonlinear PID

neural controller and the third is nonlinear inverse dynamic predictive controller. The performance of the proposed adaptive neural control with three control methodologies was tested using Matlab program and the simulation results are compared with them in terms of minimum tracking error and in generating an optimal torque control action and the capability of tracking any trajectories with continuous and non-continuous gradients, despite the presence of bounded external disturbances.

Chapter Six

Cognitive Neural Predictive Controller

6.1. Introduction

The cognitive neural predictive controller for mobile robot system is considered in this chapter. The approach is used to detect the static obstacle in the desired path of the mobile robot based on neural network model and to re-plan optimal smoothness desired path for mobile robot by using AI technique in order to avoid the static obstacle with minimum distance and to track the desired trajectory by using an adaptive neural predictive control methodology.

6.2. Cognitive Neural Predictive Controller Structure

The proposed controller can be given in the form of the block diagram shown in Figure 6.1.

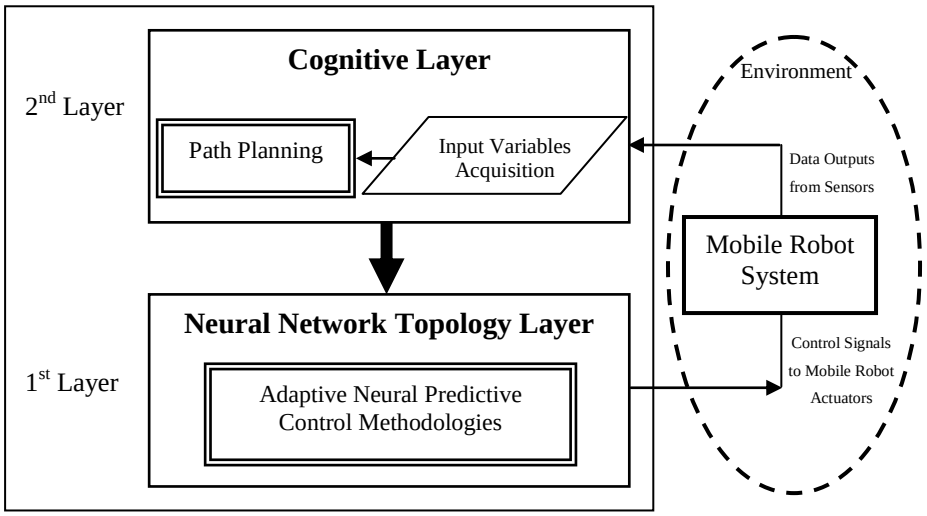


Figure 6.1: The proposed structure of the cognitive neural predictive controller for the mobile robot system.

It consists of:

1- Neural Network Topology Layer.

2- Cognitive Layer.

Neural network topology layer is the execution layer, because it controls the mechanical actuators in order to follow the mobile robot's desired trajectory (fed from the cognitive layer). The general structure of this layer is explained in chapter five.

6.2.1. Cognitive Layer

Cognitive layer is the planning layer, which collects all the information from the environment by using sensors such as IR, 2D laser scanner, ultrasound, GPS and camera. It contains built-in parameters and constraints on cognition to facilitate a priori predictions about behaviours and plan smoothness desired trajectory to minimise the travelling time and travel distance, to save the battery energy of the mobile robot. The cognitive layer can also re-plan the desired path if any static obstacle in the trajectory is detected, in order to avoid the mobile robot colliding with entities in the environment, ensuring that the trajectory tracking of the mobile robot allows collision-free navigation.

6.2.1.1. Cognition Path Planning Algorithm

Path planning aims to provide the optimum collision-free path between a starting point and target locations; therefore, the second layer in the cognitive neural predictive control methodology plans the trajectory of the mobile robot. The planned path is usually decomposed into line segments between ordered sub-goals or way points, which the mobile robot follows toward the target, as shown in Figure 6.2a. It needs to re-plan the primary path if there is any obstacle in the path, as shown in Figure 6.2b.

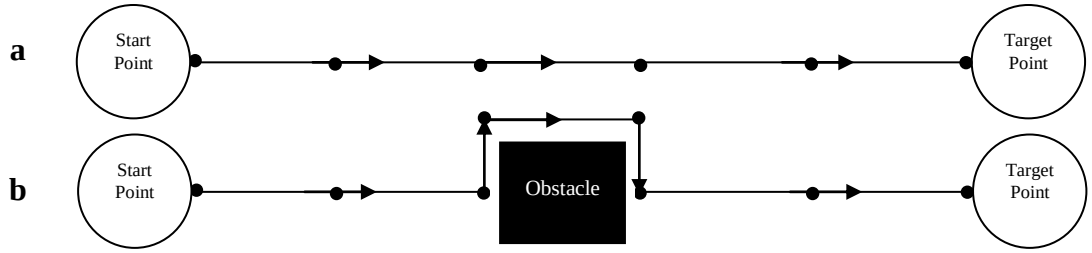


Figure 6.2: (a) the normal path; (b) the path around an obstacle.

To apply the cognition path planning for the mobile robot, it needs a model of the obstacle in the cognitive layer to determine the obstacle's dimensions in order to avoid the accident between the mobile robot cart and the obstacle. It also needs an AI method to re-plan the path with minimum distance to avoid the obstacle and reach the desired path.

The neural network can be described the obstacle model in the path, as shown in Figure 6.3 [74, 129]. The structure shown in Figure 6.3 is based on the following equations:

$$IH_m = xw_m x_i + yw_m y_i + \phi_{\mathrm{Im}} \tag{6.1}$$

$$OH_m = f(IH_m) \tag{6.2}$$

$$C_o = f(\sum_{m=1}^{M} OH_m - \phi_o) \tag{6.3}$$

$$f(r) = \frac{1}{1+e^{-r/p}} \tag{6.4}$$

where x_i and y_i are the coordinate of i^{th} points of the desired path; xw_m and yw_m are the network weights for x_i and y_i respectively; and ϕ_{Im} is the bias, which is equal to the free element in the equation expressing the shape of the obstacle.

IH_m is the weighted input of the m^{th} neuron of the middle layer and the neural activation function is $f(.)$ and p is a parameter that controls the steep of curve shape.

m is the number of the neurons in the middle layer and it is equal to the number of vertices of the obstacle.

OH_m is the output of the m^{th} neuron of the middle layer.

It is used a repulsive penalty function (RPF) in the output neuron and ϕ_o is a bias which is equal to the number of the vertices of the obstacle decreased by 0.5.

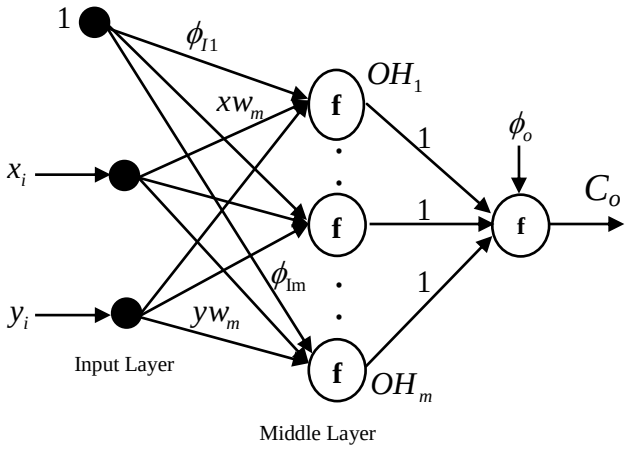


Figure 6.3: The obstacle neural network model [74, 129].

C_o is the output of the obstacle neural network model of each point in the workspace is equal to 0 or 1. If the output is equal to 1, that the coordinate (x_i,y_i) is in the obstacle region, otherwise the point is not in it. The dimensions of the mobile robot cannot be ignored, therefore the vertices of the obstacle Vp_m will increase, as shown in Figure 6.4.

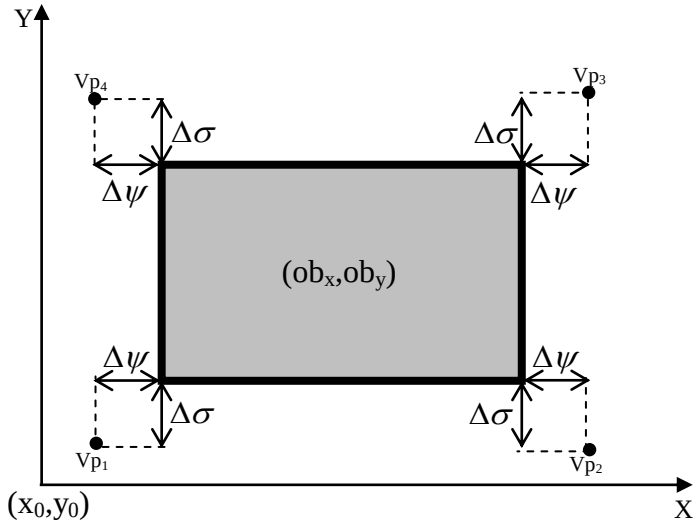


Figure (6.4): The obstacle vertices.

where $\Delta\psi$ and $\Delta\sigma$ are variable parameters that allows identification of optimal trajectory, as follows: $\frac{L_{robot}}{2} \le \Delta\psi < L_{robot}$ and $\frac{W_{robot}}{2} \le \Delta\sigma < W_{robot}$ respectively.

W_{robot} is the cart width of the mobile robot.

L_{robot} is the cart length of the mobile robot.

To build an optimal and robust path planning algorithm for manoeuvring the mobile robot to avoid the obstacle in the environment while minimising costs such as time, energy and distance, the following proposed algorithm is used:

- Re-plan the path using the primary path as a guide to anticipate the object that will be encountered.
- Determine the localisation of the obstacle centre point (ob_x,ob_y) with respect to the reference point (x_0,y_0) and determine the dimensions of the obstacle as length L_{ob} and width W_{ob}.
- To avoid the obstacle in the desired path, the start point is required, and a return to the desired path after obstacle avoidance, requiring the end point, as shown in Figure 6.5. To calculate these main points, neural network obstacle model is used (as shown in Figure 6.3) with calculation of the vertices points, as shown in Figure

6.4, as a condition for minimum or maximum distance between mobile robot centre point and obstacle centre point using equations 6.1, 6.2, 6.3 and 6.4.

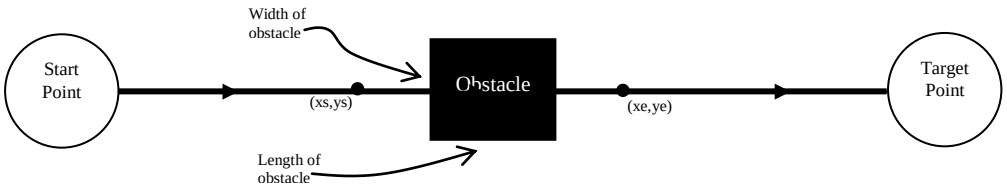


Figure 6.5: Obstacle with two main points

- The detection of the start point is undertaken by applying (x_i,y_i) for each coordinate point of the desired trajectory in the neural network obstacle model and finding the output of the model; if the model output C_o=1, the start point in the algorithm is detected as (x_{i-1},y_{i-1}), which means that the mobile robot is approaching the obstacle body. After finding the start point, (x_j,y_j) is applied for the coordinate point of the desired trajectory starting from start point in the neural network obstacle model, and finding the output of the model if the model output is changed from C_o=1 to C_o=0, the end point in the algorithm is detected as (x_j,y_j), that means the mobile robot is away from the obstacle body and it returns to the desired trajectory and the number of points between start and end points is denoted as φ.
- To find the optimal side of the obstacle that will re-plan the desired path, the positions of the three points are used (start point, obstacle centre point and end point), as shown in Figure 6.6.

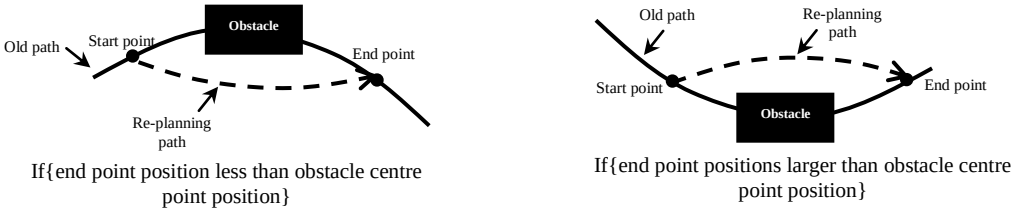


Figure 6.6: Optimal side of the obstacle.

- After determining the start and end points and the optimal side for re-planning the path, numerical techniques such as cubic spline interpolation polynomial or AI technique (e.g. particle swarm optimisation (PSO)) are applied to plan the optimal smoothness path without overshooting between the start and end points with minimum distance.

6.2.1.2. Cubic Spline Interpolation Technique

There are numerous techniques of interpolation, such as Lagrange interpolation polynomial [130, 131], Newton interpolation polynomial [132, 133], Neville interpolation polynomial [134], Piecewise interpolation polynomial [135, 136] and cubic spline interpolation polynomial [137, 138]. Cubic spline interpolation polynomial was used in this study due to its smoothness, predictability, non-overshooting behaviour and simplicity. The cubic spline interpolation is third-degree polynomial of the Piecewise interpolation, where the higher the degree of spline is the smoother the curve. The cubic spline function $f_i(x)$ as third order polynomials for each segment is as follows:

$$f_i(x) = a_i + b_i x + c_i x^2 + d_i x^3 \tag{6.5}$$

where *i* is the number of segments (*i*=1, 2, 3…Sg).

To calculate the parameters of the polynomial in equation 6.5 for each segment see appendix H.

The same number of points is used for the deleted track in order to keep the same travelling time for the mobile robot, denoted as φ. Therefore, T, which is the travelling time between all segments track, is calculated using equation 6.6:

$$T = \varphi \times T_s \tag{6.6}$$

where φ is the number of points. T_s is the sampling time.

It is necessary to calculate the estimation distance of the track segments as equation 6.7 [92, 94]:

$$D = \sum_{j=1}^{\varphi-1} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \tag{6.7}$$

To investigate the optimal travelling time for the mobile robot during to track the optimal path, the desired linear velocity of the mobile robot should be not exceed to the V_{Imax}. This can be calculated as equation 6.8:

$$V_I = \frac{D}{T} \rightarrow V_I < V_{\mathrm{Im}ax} \tag{6.8}$$

6.2.1.3. Particle Swarm Optimisation Technique

Many AI methods have been used to find the optimal path and to avoid static or dynamic obstacles, such as genetic algorithm (GA) and PSO algorithm [74, 77]. PSO is a kind of algorithm to search for the best solution by simulating the movement and flocking of birds. PSO algorithms use a population of individuals (called particles) whose positions represent the potential solutions for the studied problem, with velocities are randomly initialized in the search space. The search for optimal solution is performed by updating the particle velocities and positions in each iteration [77,139, 140].

The aim of the algorithm is to determine the points (x_i,y_i) (i=1, 2, 3$\ldots\varphi$) that constitute the optimal smoothness path from the starting point to the end point. In order to reduce the length of the point's string, the point's x_i is determined by using the x-axes of the start and end points. Therefore, y_i becomes the search space for each via-point of the mobile robot trajectory and the via-point candidates are specified by one-dimensional data.

The conventional evolutionary equations of particle swarm optimisation are as follows [77, 139, 140, 141]:

$$V_{i,d}^{k+1} = V_{i,d}^{k} + c_1 r_1 (pbest_{i,d}^{k} - y_{i,d}^{k}) + c_2 r_2 (gbest_{d}^{k} - y_{i,d}^{k}) \tag{6.9}$$

$$y_{i,d}^{k+1} = y_{i,d}^{k} + V_{i,d}^{k+1} \tag{6.10}$$

$$i = 1,2,3,.....pop$$

$$d = 1,2,3,.....\varphi$$

where:

pop is number of particles.

$V_{i,d}^{k}$ is the velocity of the i^{th} particle at k iteration.

$y_{i,d}^{k}$ is the position of the i^{th} particle at k iteration.

c_1 and c_2 are the acceleration constants with positive values equal to 2.

r_1 and r_2 are random numbers between 0 and 1.

$pbest_i$ is best previous weight of i^{th} particle.

$gbest_d$ is best particle among all the particle in the population.

The previous best value is called *pbest*. *pbest* is related only to a particular particle. It also has another value called *gbest*, which is the best value of all the *pbest* particles in the swarm. The particles are evaluated using a fitness function to see how close they are to the optimal solution and the stability of the PSO algorithm. Two evaluation functions must be integrated into a fitness function, the collision avoidance and the shortest distance. Collision avoidance is essential to path planning and makes the mobile robot travel in the workspace safely.

Collision avoidance can be described in two main points:

1- The via-point y_i should not be in the obstacle region.

$$CA_{Fit1} = \begin{cases} 1 & if \quad C_o = 0 \\ 0 & others \end{cases} \tag{6.11}$$

2- The section $y_i y_{i+1}$ should not intersect obstacle region.

$$CA_{Fit2} = \begin{cases} 0 & y_i y_{i+1} \cap obstacle \\ 1 & others \end{cases} \tag{6.12}$$

The fitness function of the collision avoidance can be expressed as equation 6.13:

$$CA_{Fit} = CA_{Fit1} \,\&\, CA_{Fit2} \tag{6.13}$$

The minimum distance is the second fitness function which makes the mobile robot travel in the workspace with minimum travelling time and travel distance and can be expressed as follows [92, 94]:

$$MD_{Fit} = \sum_{j=1}^{\varphi-1} \sqrt{(x_{j+1} - x_j)^2 + (y_{j+1} - y_j)^2} \tag{6.14}$$

The final fitness function is constructed as shown in equation 6.15:

$$Fit = MD_{Fit} / CA_{Fit} \tag{6.15}$$

When the final fitness function reaches the minimum value, the global optimal smoothness path is found. To investigate whether the new desired trajectory is optimal travelling time for the mobile robot, the desired linear velocity of the mobile robot while tracking the optimal path in order to avoid the static obstacle should not exceed the V_{Imax} and can be calculated using equation 6.16:

$$V_I = \frac{MD_{Fit}}{T} \rightarrow V_I < V_{\mathrm{Im}ax} \tag{6.16}$$

Therefore, T must be calculated, which is the travelling time of the tracking between the start and end points, using equation 6.6.

The steps of PSO for re-planning path for the mobile robot can be described as follows:

- **Step1** Initial searching points y_1^0 and V_1^0 of each particle are usually generated randomly within the allowable range. Note that the dimension of search space is consists of all the number of points between start and end points. The current searching point is set to *pbest* for each particle. The best-evaluated value of *pbest* is set to *gbest* and the particle number with the best value is stored.
- **Step2** The objective function value is calculated for each particle by using equation 6.15. If the *pbest* value is better than the current *pbest* of the particle, the *pbest* value is replaced by the current value. If the best value of *pbest* is better than the current *gbest, gbest* is replaced by the best value and the particle number with the best value is stored.
- **Step3** The current searching point of each particle is updated by using equations 6.9 and 6.10.
- **Step4** The current iteration number reaches the predetermined maximum iteration number, then exit; otherwise, return to step 2.

6.3. Simulation Results

In this section, continuous and non-continuous gradients desired trajectories are tracked from nonholonomic wheeled mobile robot with existence static obstacle in the path in order to clarify the features of the cognitive neural predictive controller. The performance of the proposed cognitive neural control with the best control methodology (outlined in chapter five) in terms of minimum mean square errors in the position and orientation tracking error are taken in order to detect the static obstacle and re-plan the desired path to avoid the obstacle, and to reach and follow the original path collision-free. This comparison has to be made under identical conditions in order to specify the differences and similarities and consequently the advantages and disadvantages each of the proposed path planning methodologies. The simulation results in this chapter are implemented using Matlab program.

6.3.1. Case Study 1

The desired lemniscates trajectory, which has explicitly continuous gradient with rotation radius changes, and the equations of the desired lemniscates trajectory, are stored in the cognitive layer to generate the desired position and orientation at each time interval and to detect and determine the distance between the mobile robot and the obstacle in order to re-plan the path for collision-free navigation. As shown in Table 5.1, the third control methodology has smaller MSE of the position (X-coordinate, Y-coordinate) and the orientation than start and second control methodologies; therefore, it can be considered that the third control methodology was the best methodology for the desired lemniscates trajectory.

A static obstacle is placed in the desired path in order to verify the robustness and adaptation of the cognitive neural predictive controller and its capability to track any trajectory through re-planning the desired path to avoid the static obstacle with minimum tracking error. To re-plan the desired lemniscates path many tasks in the cognitive layer of the controller are executed (as a proposed algorithm for obstacle avoidance) as follows:

From the Figure 6.7, obstacle centre point position (ob_x,ob_y) is determined as (1.2965,0.998), then the length L_{ob} = 0.18m and width W_{ob}=0.16m of the obstacle body are determined, in order to calculate the vertices points of the obstacle, as shown in Figure 6.4, and built into the obstacle neural network model as follows:

$$IH_1 = x_i - 1.1465 \tag{6.17}$$

$$IH_2 = -x_i + 1.446 \tag{6.18}$$

$$IH_3 = y_i - 0.848 \tag{6.19}$$

$$IH_4 = -y_i + 1.148 \tag{6.20}$$

where

$\Delta\psi$ is equal to $\frac{L_{robot}}{2}$ = 0.06m

$\Delta\sigma$ is equal to $\frac{W_{robot}}{2}$ =0.07m

L_{robot} is the length of the mobile robot cart and it's equal to 0.12m.

W_{robot} is the width of the mobile robot cart and it's equal to 0.14m.

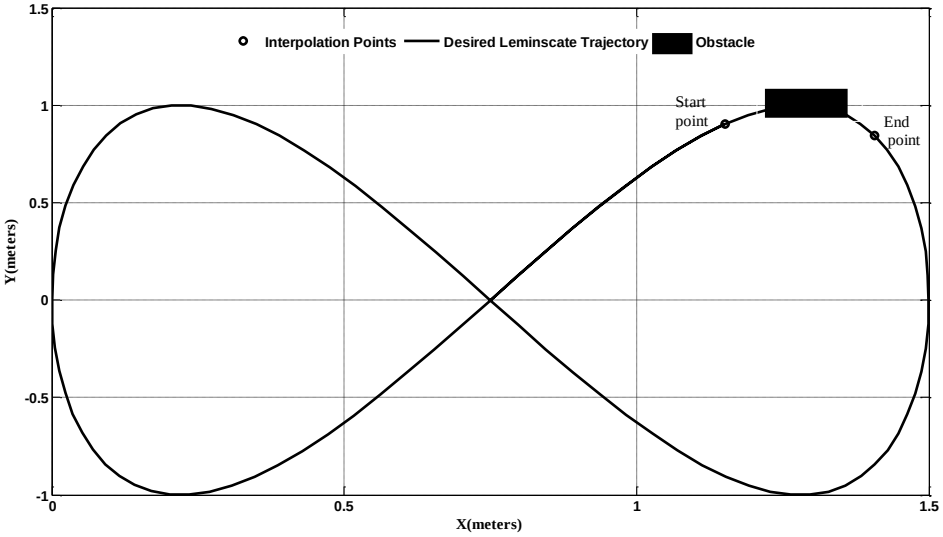


Figure 6.7: The obstacle in the desired lemniscates path.

The new path that the mobile robot will track to avoid the obstacle consists of two main points; start point and end point. These two points are determined depending on the obstacle neural network model, here determined as follows: start point (1.1417, 0.844) and end point (1.407, 0.8349).

The best side to re-plan the path is below the obstacle, because this achieves minimum distance (depending on the proposed algorithm) as shown in Figure 6.6. Then the cubic spline interpolation technique is applied to find the new smoothness path with non-overshooting behaviour that will reduce the tracking error as follows:

For first segment, i=1, $1.1517 \le x_i \le 1.2787$:

$$y_i(x_i) = 4.6533 - 5.7279x_i + 2.1472x_i^2 + 3\times10^{-6}x_i^3 \quad (6.21)$$

For second segment, i=2, $1.2787 \le x_i \le 1.407$:

$$y_i(x_i) = 4.6533 - 5.728x_i + 2.1473x_i^2 - 9\times10^{-6}x_i^3 \quad (6.22)$$

After planning the new path in the cognitive layer, as shown in Figure 6.8, the number of points should be determined that the mobile robot must track, and the linear velocity should not exceed V_{Imax} equal to 0.165m/sec, as follows:

It is known the number of points of the deleted track φ is equal to seven points with start and end points. Equation 6.6 is then applied to find the travelling time of the two

segments track (equal to 3.5 sec). Then the distance *D* between start-point and end-point (two segments track) is determined using equation 6.7, equal to 0.3007m. Then the linear velocity of the mobile robot is checked to see whether it exceeds V_{Imax} or not, using equation 6.8; it is equal to 0.08591m/sec.

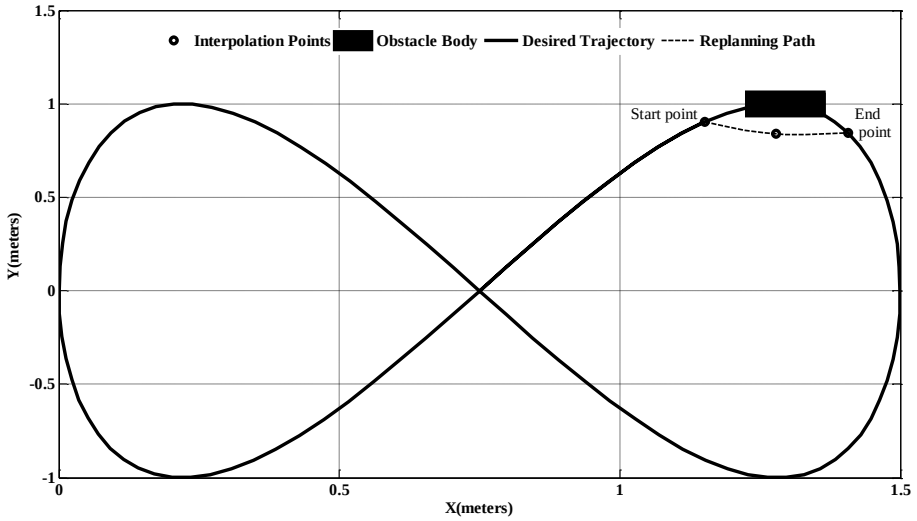
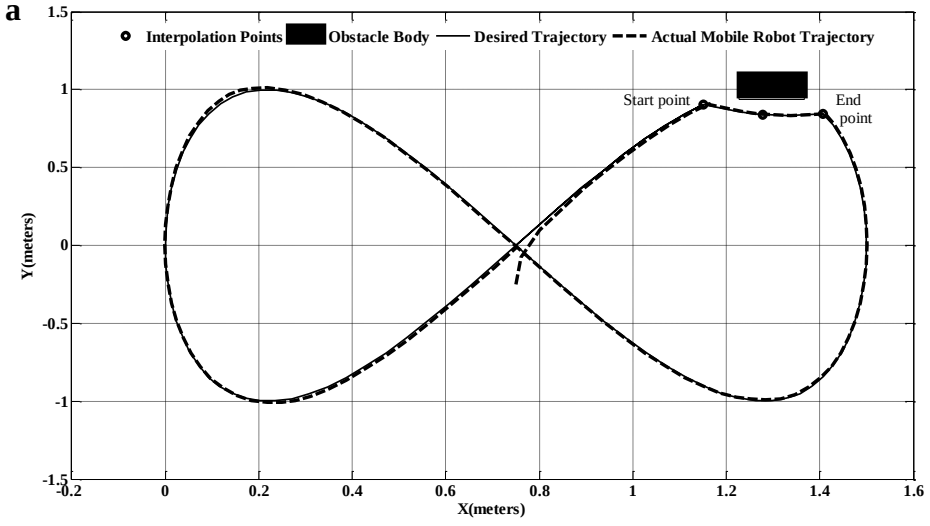


Figure 6.8: Re-planning the desired lemniscates trajectory of the mobile robot.

The third control methodology with five steps-ahead prediction has the smallest MSE of the position (X-coordinate, Y-coordinate) and the orientation, as shown in Table 5.1, for trajectory tracking of the mobile robot with obstacle in the lemniscates path. After re-planning the lemniscates path by the cognitive neural predictive controller, the trajectory tracking of the mobile robot is obtained as shown in Figures 6.9a and 6.9b.



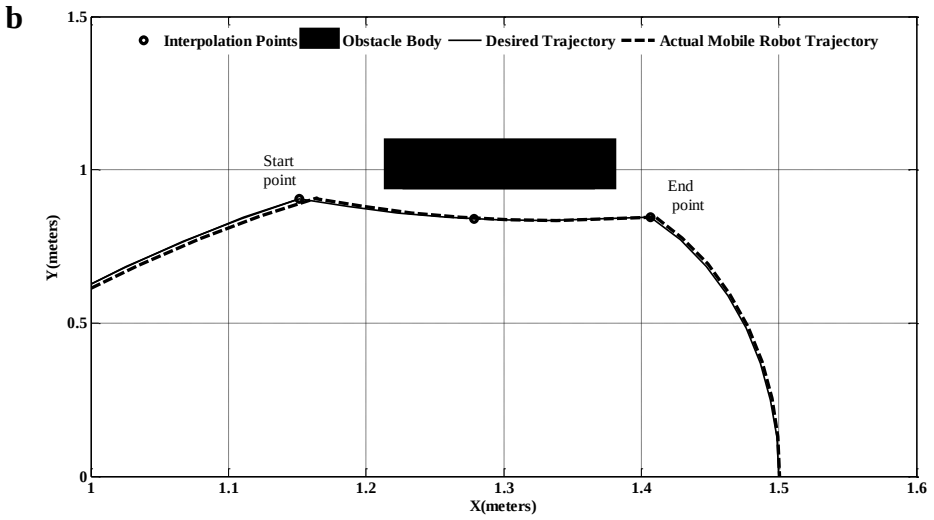


Figure 6.9: Actual trajectory of mobile robot and desired lemniscates trajectory with obstacle avoidance: (a) overall trajectory tracking; (b) zoom in during obstacle avoidance.

These Figures demonstrate very good position and orientation tracking performance for obstacle avoidance and minimum tracking error in spite of the existence of bounded disturbances. The simulation results demonstrated the effectiveness of the proposed cognitive neural predictive controller by showing its ability to generate on-line small values of the control input torques for right and left wheels with small sharp spikes from twelve to eighteen samples, in order to track the new desired path and to avoid the static obstacle, as shown in Figure 6.10.

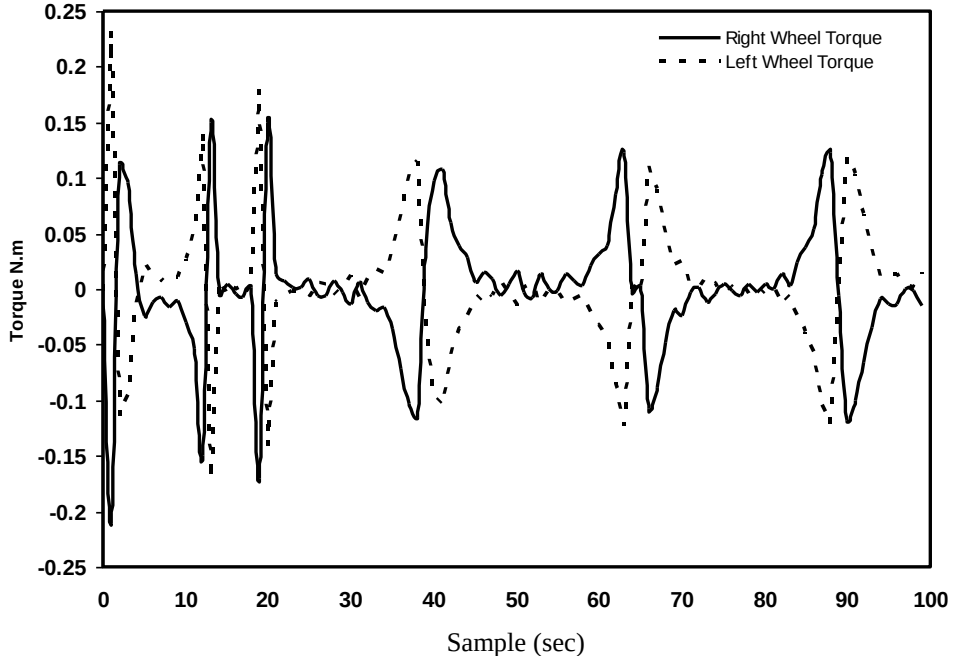


Figure 6.10: The torque of the right and left wheel action.

The velocities of the right and left wheels are changed during seven samples from 12 to 18 samples, as shown in Figure 6.11a. The mean linear velocity of the mobile robot is equal to 0.08591m/sec, as shown in Figure 6.11b and maximum angular velocity is equal to -1rad/sec during these seven samples of avoiding the static obstacle. The overall mean linear velocity of the mobile robot is equal to 0.135m/sec.

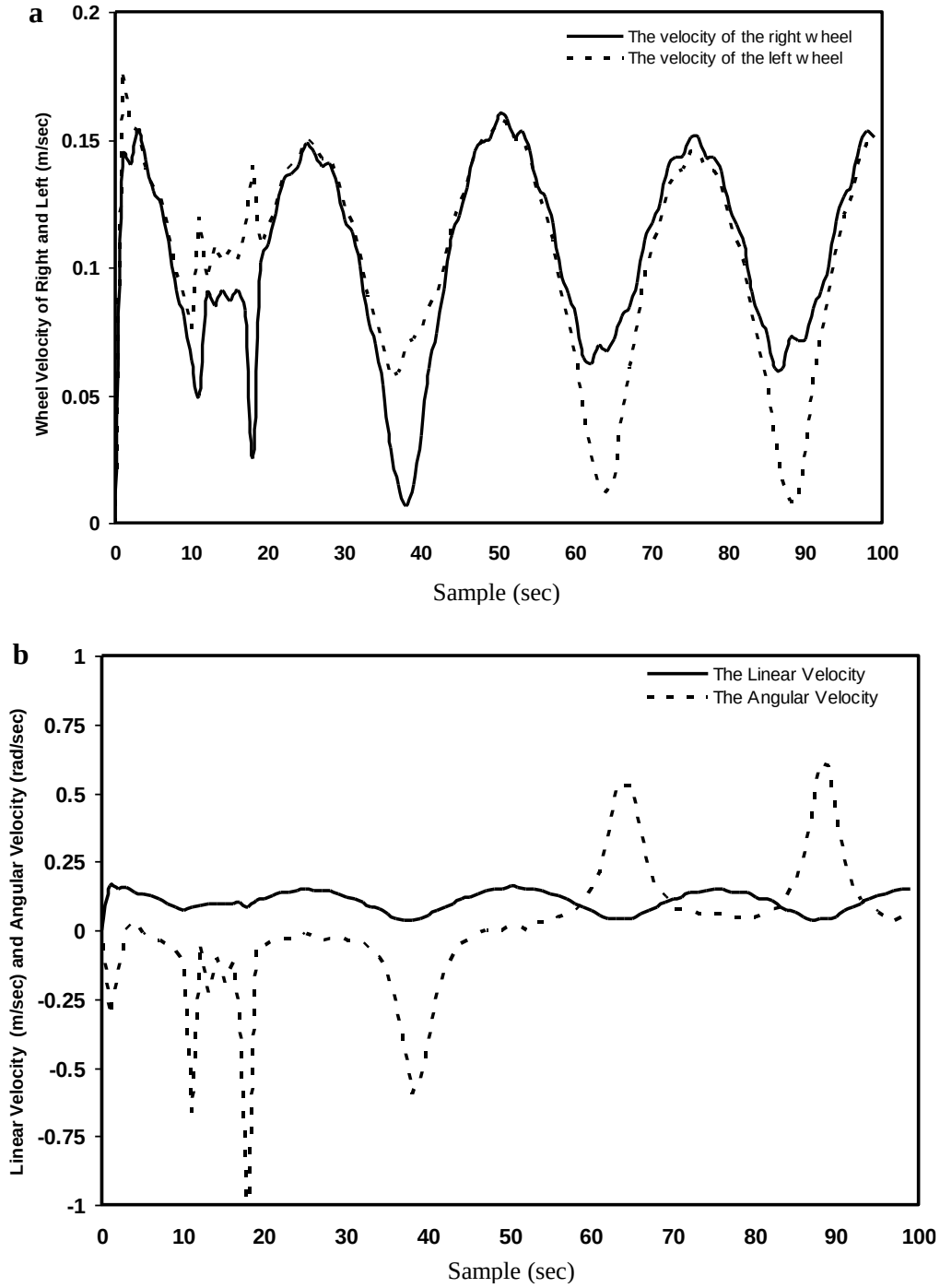


Figure 6.11: Mobile robot velocity: (a) the right and left wheel velocity action; (b) the linear and angular velocity action.

PSO technique steps are used to find the new smoothness path without overshooting behaviour and minimum distance as shown in Figure 6.12, depending on fitness function as in equation 6.15, from the start point (1.1417, 0.846) to the end point (1.407, 0.8349). The size of the particle dimension is equal to five and the population of particles is equal to 100.

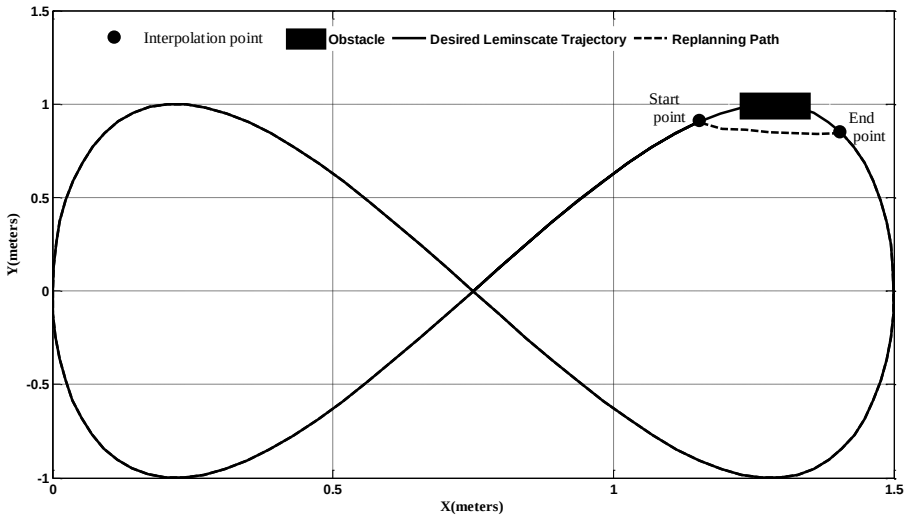
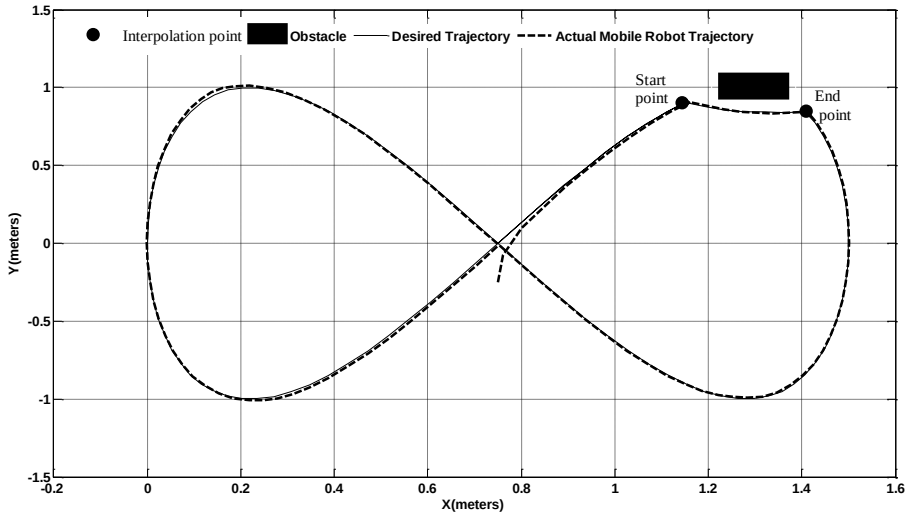


Figure 6.12: Re-planning the desired lemniscates trajectory of the mobile robot.

The distance *D* between start point and end point is equal to 0.293m using equation 6.14. The linear velocity of the mobile robot is then checked to see whether it exceeds V_{Imax} or not using equation 6.16; it is equal to 0.08378m/sec. After the cognitive neural predictive controller re-plans the lemniscates path using the third control methodology, the trajectory tracking of the mobile robot is obtained, as shown in Figures 6.13a and 6.13b.

a



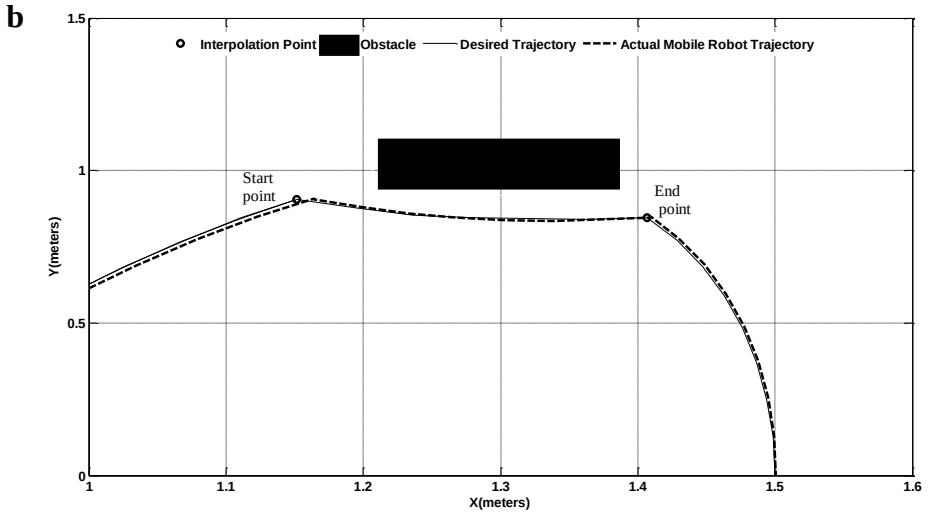


Figure 6.13: Actual trajectory of mobile robot and desired lemniscates trajectory with obstacle avoidance: (a) overall trajectory tracking; (b) zoom in during obstacle avoidance.

These Figures demonstrate excellent position and orientation tracking performance for obstacle avoidance and minimum tracking error in spite of the existence of bounded disturbances. The simulation results demonstrated the effectiveness of the proposed cognitive neural predictive controller by showing its ability to generate on-line small values of the control input torques for right and left wheels with small sharp spikes from twelve to eighteen samples in order to track the new desired path and to avoid the static obstacle, as shown in Figure 6.14.

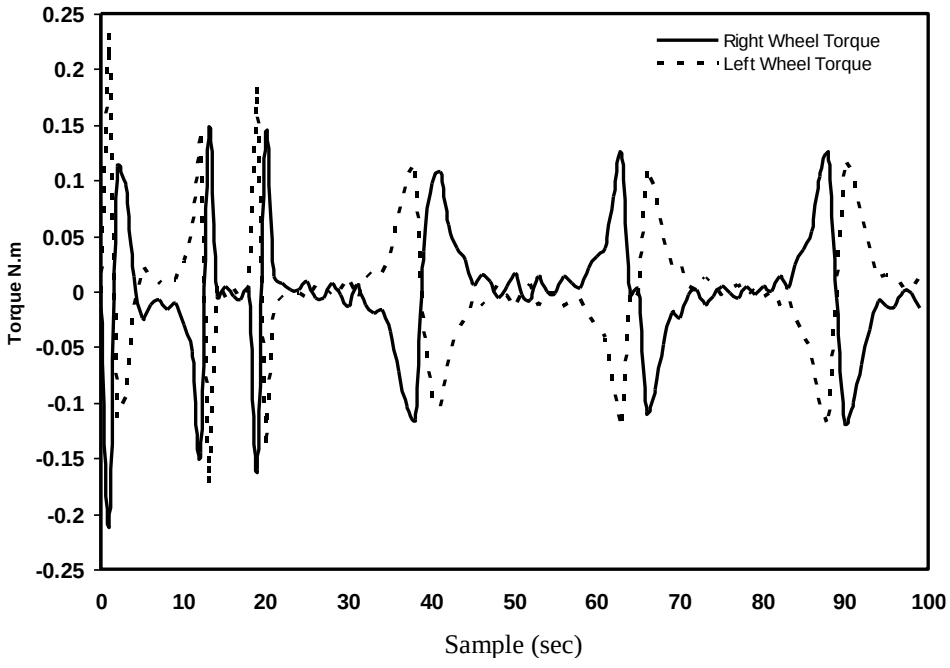


Figure 6.14: The torque of the right and left wheel action.

The velocities of the right and left wheels are changed during seven samples from 12 to 18 samples, as shown in Figure 6.15a.

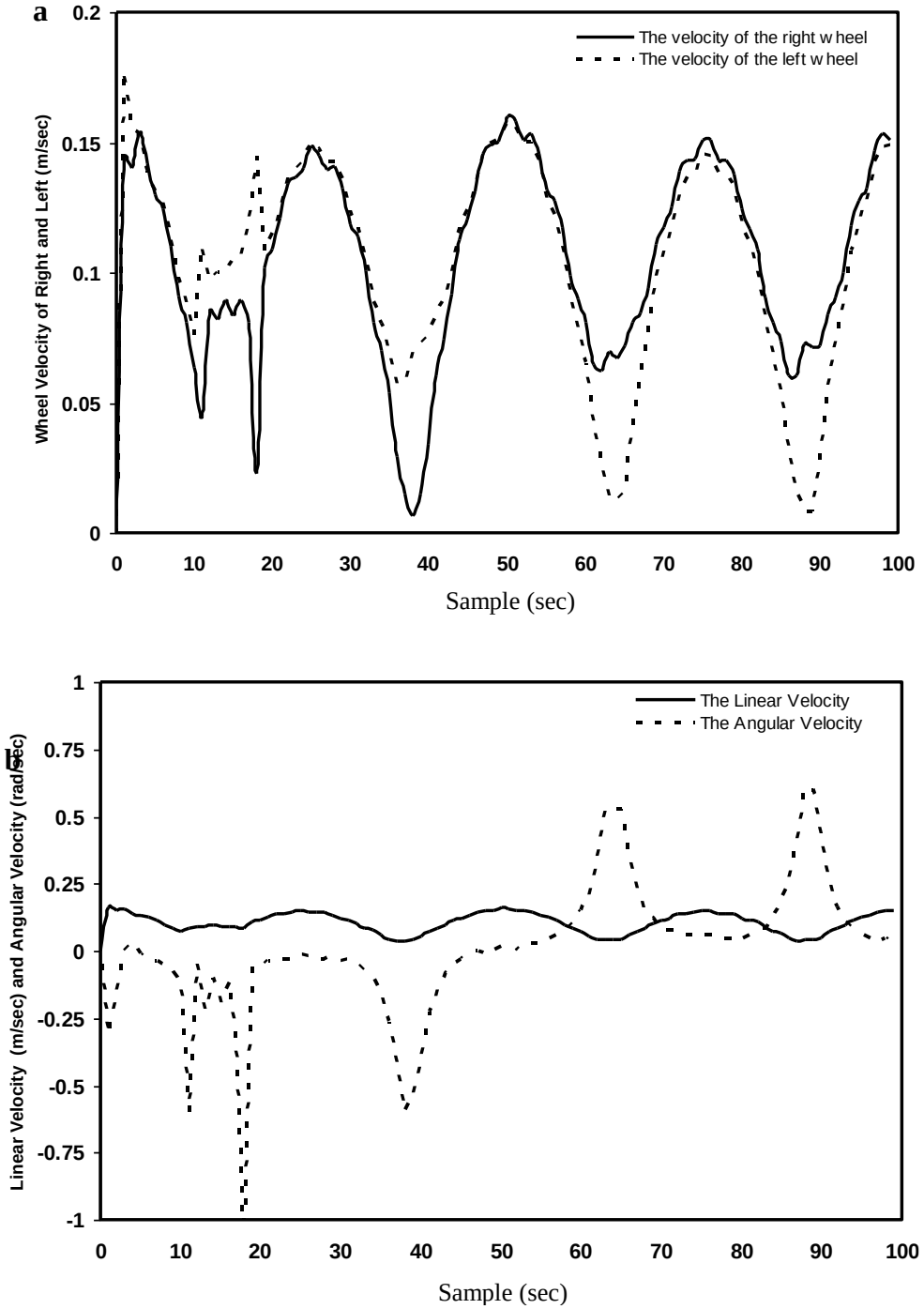


Figure 6.15: Mobile robot velocity: (a) the right and left wheel velocity action; (b) the linear and angular velocity action.

The mean linear velocity of the mobile robot is equal to 0.08378m/sec during seven samples from 12 to 18 samples is shown in Figure 6.15b. The maximum angular velocity

is equal to -0.96rad/sec during these seven samples (avoiding the static obstacle). The overall mean linear velocity of the mobile robot is equal to 0.135m/sec.

The comparison between two techniques (numerical and intelligent) to re-plan the desired lemniscates trajectory is shown in Table 6.1.

Table 6.1: Numerical and intelligent techniques for minimum distance and linear velocity in the lemniscates path.

Techniques	Distance (m)	Velocity (m/sec)	Time (sec)	MSE of X-coordinate	MSE of Y-coordinate	MSE of Orientation
Cubic spline interpolation	0.3007	0.08591	3.5	1.77×10^{-4}	5.1×10^{-5}	1.39×10^{-3}
PSO technique	0.2930	0.08378	3.5	9.2×10^{-5}	6.85×10^{-8}	1.21×10^{-4}

The PSO technique is better than cubic spline interpolation technique because the former re-plans the path to avoid the static obstacle with minimum distance and minimum linear velocity, as well as minimum mean square error for each component of the state error for the seven samples from 12 to 18, as shown in Table 6.1.

6.3.2. Case Study-2

The desired circular trajectory which has explicitly continuous gradient with rotation radius constant and the equations of the desired circular trajectory are stored in the cognitive layer to generate the desired position and orientation at each time interval and to detect and calculate the distance between the mobile robot and the obstacle in order to re-plan the path for collision-free navigation.

Applying the tasks of the cognitive layer to detect and re-plan the desired circular path to avoid the static obstacle is undertaken as follows. From Figure 6.16, the obstacle centre point position (ob_x,ob_y) is determined as (0,1), after which the vertices points of the obstacle are determined, as shown in Figure 6.4, and the obstacle neural network model is built using the following equations:

$$IH_1 = x_i - 0.15 \tag{6.23}$$

$$IH_2 = -x_i + 0.15 \tag{6.24}$$

$$IH_3 = y_i - 0.85 \tag{6.25}$$

$$IH_4 = -y_i + 1.15 \tag{6.26}$$

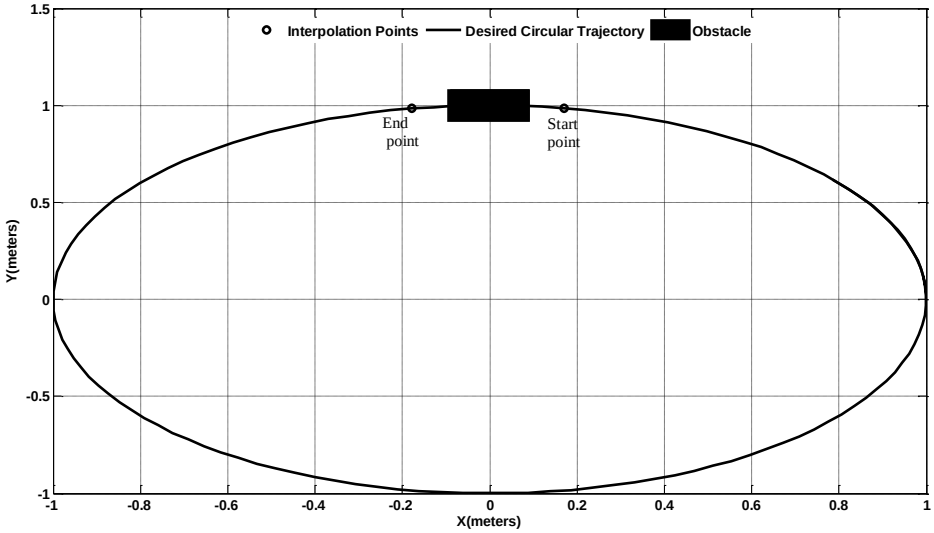


Figure 6.16: The obstacle in the desired circular path.

The new path that the mobile robot will track to avoid the obstacle consists of two main points; start point and end point. These two points are determined depending on the obstacle neural network model, and they are found as follows: start-point (0.17, 0.9854) and end-point (-0.1782, 0.984). The best side to re-plan the path is below the obstacle, because it will achieve minimum distance depending on the proposed algorithm, as shown in Figure 6.6. Then the cubic spline interpolation technique is applied to find the new smoothness path with non-overshooting behaviour that will reduce the tracking error as follows:

For first segment, i=1, $0.17 \geq x_i \geq -0.0052$:

$$y_i(x_i) = 0.8038 + 0.0536x_i + 5.9716x_i^2 - 4\times10^{-6}x_i^3 \quad (6.27)$$

For second segment, i=2, $-0.0052 \geq x_i \geq -0.178$:

$$y_i(x_i) = 0.8038 + 0.0536x_i + 5.9716x_i^2 - 5\times10^{-6}x_i^3 \quad (6.28)$$

After planning the new path in the cognitive layer, as shown in Figure 6.17, the number of points that the mobile robot must be track and the linear velocity that should not exceed V_{Imax} (equal to 0.165m/sec) should be determined, as follows. It is known that the number of points of the deleted track φ is equal to seven points, with start and end points. Equation 6.6 is then applied to find the travelling time of the two segments track; it is equal to 3.5 sec. The distance *D* between start-point and end-point (two segments track) is then determined using equation 6.7; it is equal to 0.525m. Then the linear velocity of the

mobile robot is checked to see whether it exceeds V_{Imax} or not using equation 6.8; it is equal to 0.15m/sec.

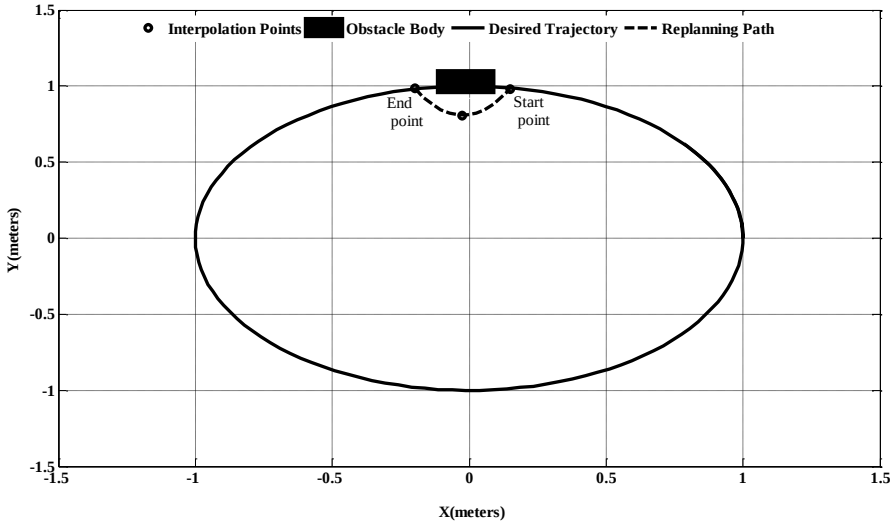
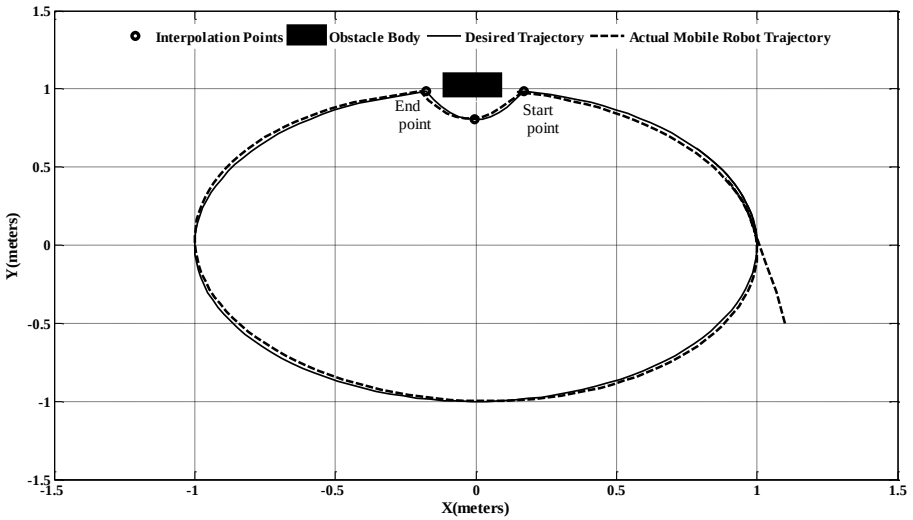


Figure 6-17: Re-planning the desired circular trajectory of the mobile robot.

The mobile robot model starts from the initial position and orientation $q(0) = [1.1, -0.5, \pi/2]$ as its initial conditions with the external disturbance. As shown in Table 5.2, the second control methodology with five steps-ahead predictions has the smallest MSE of the position (X-coordinate, Y-coordinate) and the orientation, for trajectory tracking of the mobile robot with obstacle in the circular path. After re-planning the circular path by the cognitive neural predictive controller, the trajectory tracking of the mobile robot is obtained as shown in Figures 6.18a and 6.18b.

a



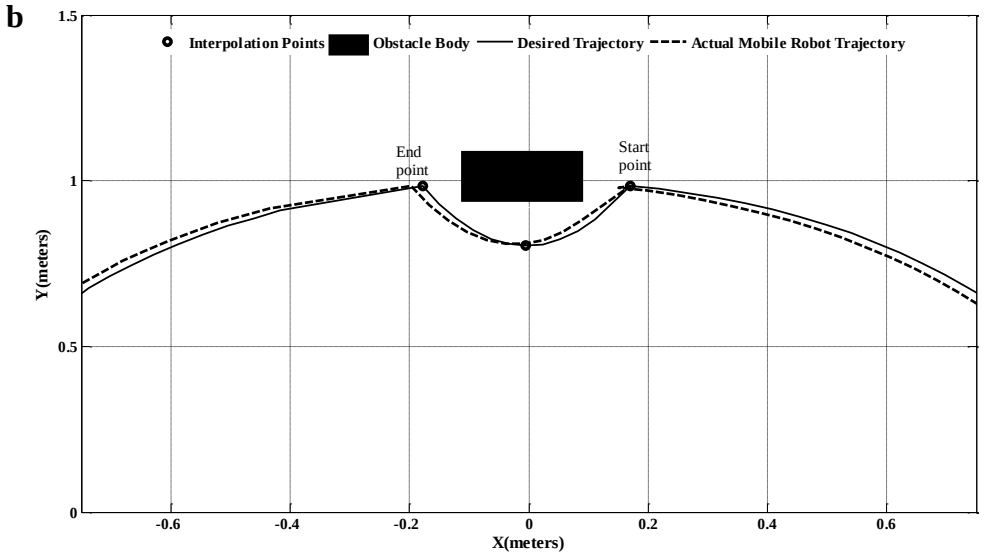


Figure 6.18: Actual trajectory of mobile robot and desired circular trajectory with obstacle avoidance: (a) overall trajectory tracking; (b) zoom in during obstacle avoidance.

These Figures demonstrate the position and orientation tracking performance for obstacle avoidance and minimum tracking error in spite of the existence of bounded disturbances. The simulation results demonstrated the effectiveness of the proposed cognitive neural predictive controller by showing its ability to generate on-line small values of the control input torques for right and left wheels with small sharp spikes from thirty to thirty-six samples in order to track the new desired path and to avoid the static obstacle, as shown in Figure 6.19.

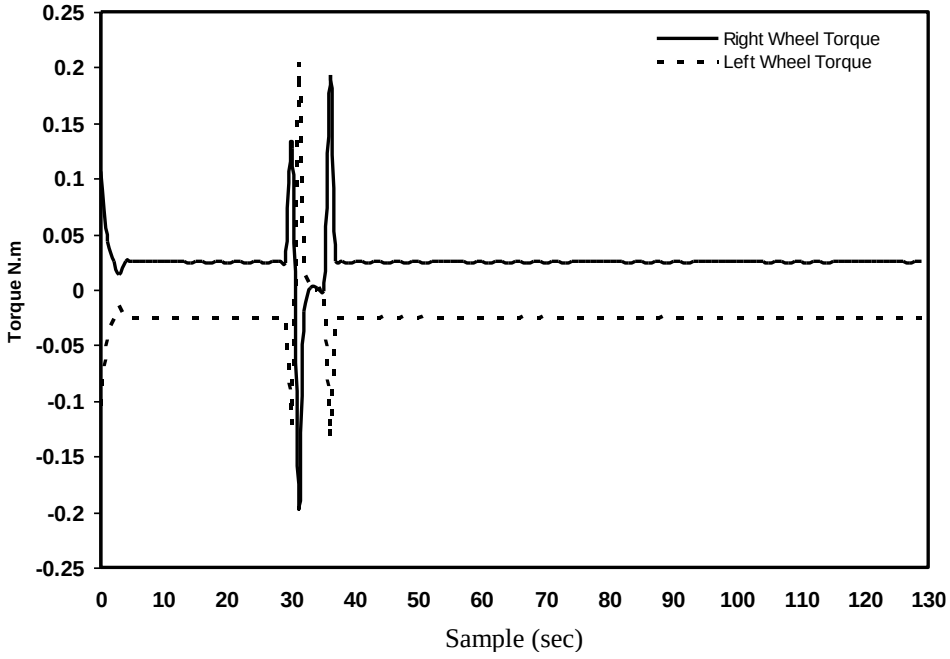


Figure 6-19: The torque of the right and left wheel action.

The velocities of the right and left wheels are changed during seven samples from 30 to 36 samples, as shown in Figure 6.20a. The mean linear velocity of the mobile robot is equal to 0.15m/sec, as shown in Figure 6.20b, and maximum angular velocity is equal to 0.7rad/sec during these seven samples; it avoided the static obstacle. The overall mean linear velocity of the mobile robot is equal to 0.0975m/sec.

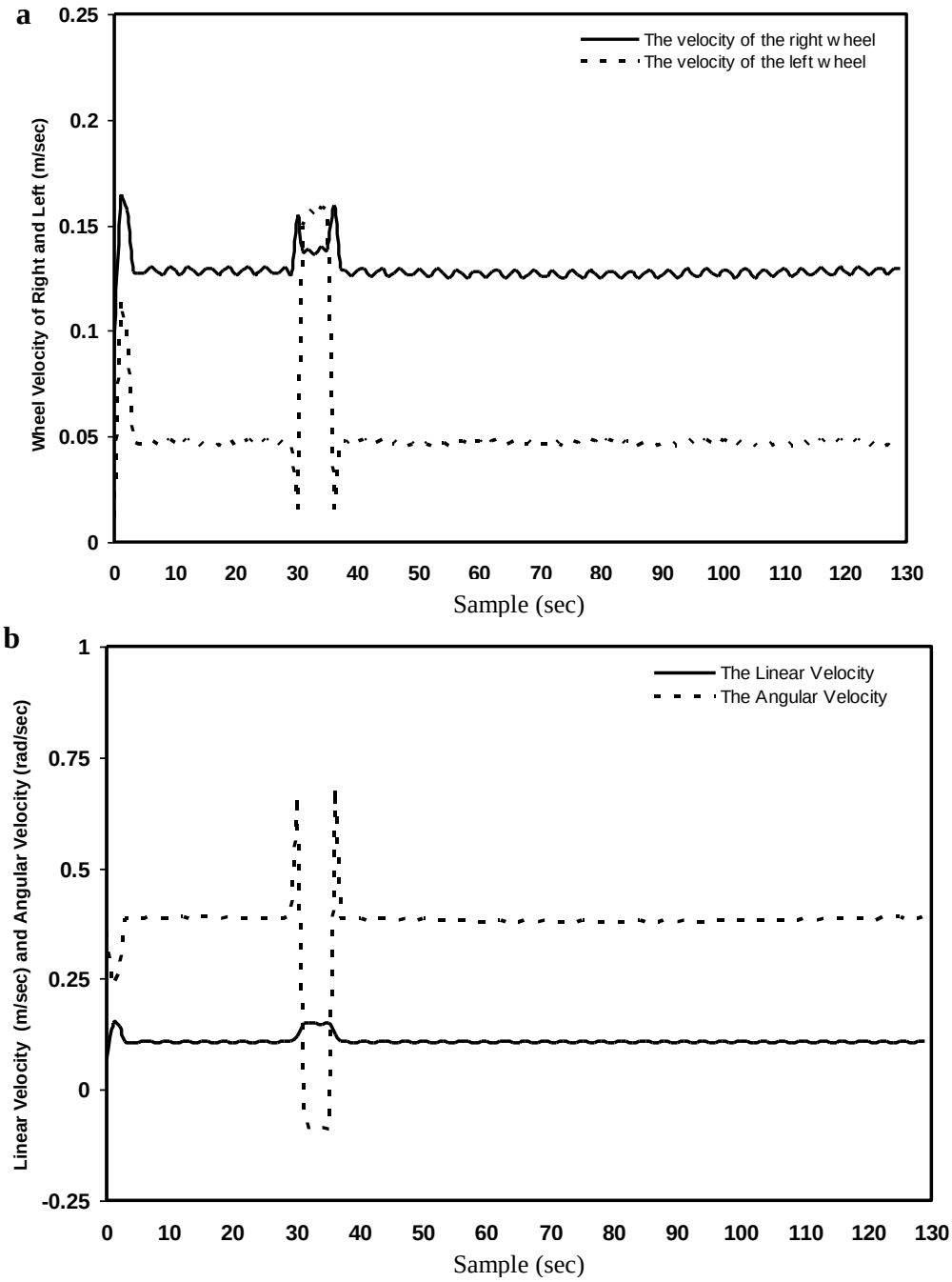


Figure 6.20: Mobile robot velocity: (a) the right and left wheel velocity action; (b) the linear and angular velocity action.

By using particle swarm optimisation technique steps to find the new smoothness path without overshooting behaviour and minimum distance, as shown in Figure 6.21, fitness function is used (as in equation 6.15) from the start point (0.17, 0.9854) to the end point (-0.1782, 0.984). The size of the particle dimension is equal to five, and the population of particles is equal to 100.

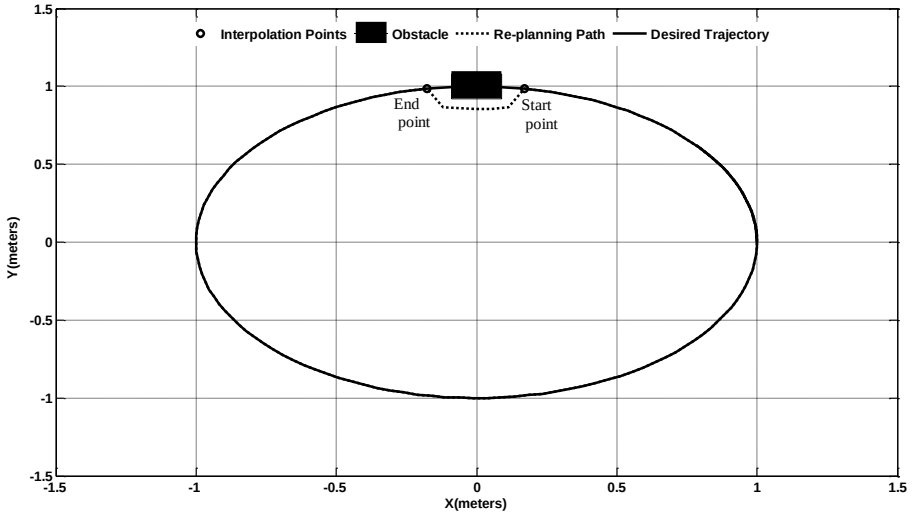
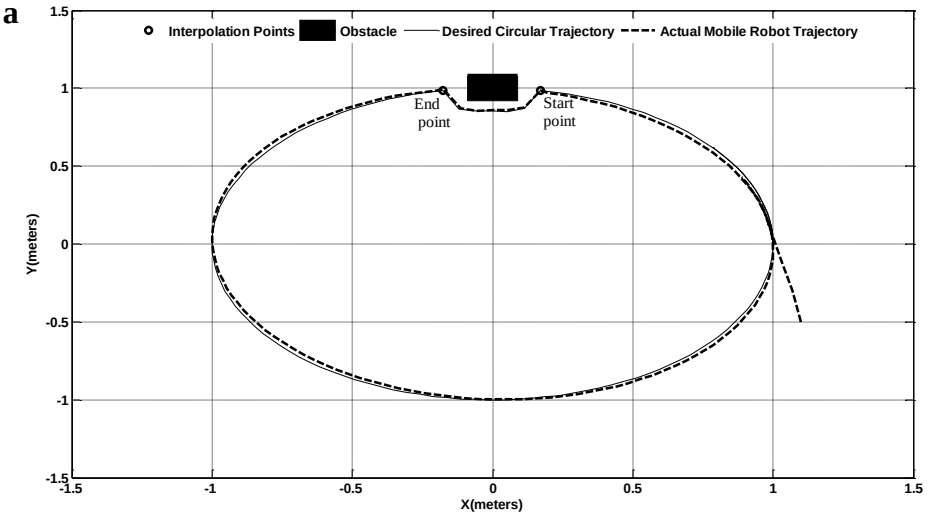


Figure 6.21: Re-planning the desired circular trajectory of the mobile robot.

The distance between start-point and end-point is equal to 0.4983m using equation 6.14. The linear velocity of the mobile robot was checked using equation 6.16 to ensure that it did not exceed V_{Imax} and it was found to be equal to 0.1423m/sec. After re-planning the circular path by the cognitive neural predictive controller with second control methodology, the trajectory tracking of the mobile robot is obtained, as shown in Figures 6.22a and 6.22b.

a



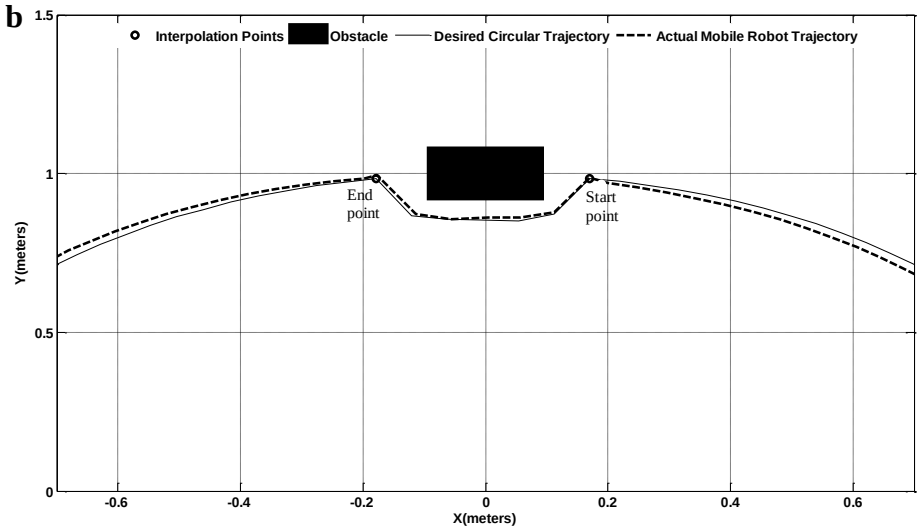


Figure 6.22: Actual trajectory of mobile robot and desired circular trajectory with obstacle avoidance: (a) overall trajectory tracking; (b) zoom in during obstacle avoidance.

These Figures demonstrate excellent position and orientation tracking performance for obstacle avoidance and minimum tracking error in spite of the existence of bounded disturbances. The simulation results demonstrated the effectiveness of the proposed cognitive neural predictive controller by showing its ability to generate on-line small values of the control input torques for right and left wheels with small sharp spikes from thirty to thirty-six samples in order to track the new desired path and to avoid the static obstacle, as shown in Figure 6.23.

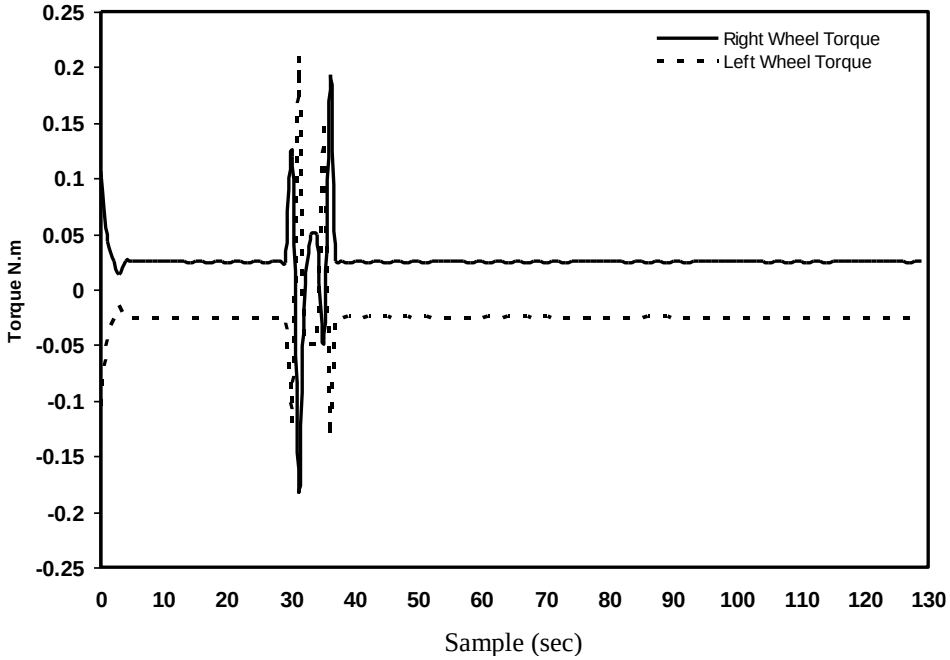


Figure (6-23): The torque of the right and left wheel action.

The velocities of the right and left wheels are changed during seven samples from 30 to 36 samples, as shown in Figure 6.24a.

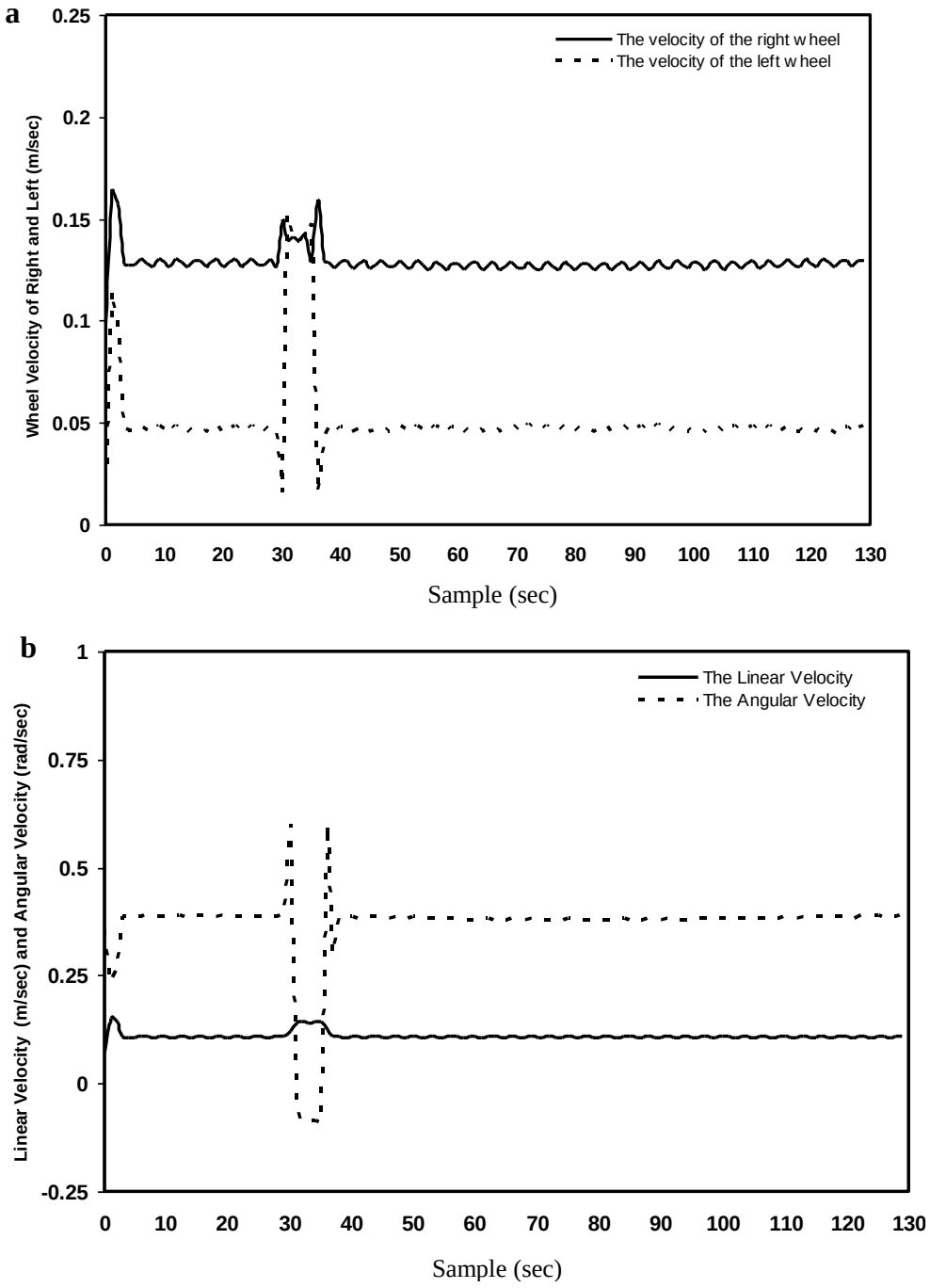


Figure 6.24: Mobile robot velocity: (a) the right and left wheel velocity action; (b) the linear and angular velocity action.

The mean of linear velocity of the mobile robot is equal to 0.1423m/sec, as shown in Figure 6.24b, and maximum angular velocity is equal to 0.63rad/sec during these seven samples; the static obstacle was avoided. The overall mean linear velocity of the mobile robot is equal to 0.0975m/sec.

The comparison between two techniques (numerical and intelligent) to re-plan the desired circular trajectory is shown in Table 6.2.

Table 6.2: Numerical and intelligent techniques for minimum distance and linear velocity in the circular path.

Techniques	Distance (m)	Velocity (m/sec)	Time (sec)	MSE of X-coordinate	MSE of Y-coordinate	MSE of Orientation
Cubic spline interpolation	0.5250	0.150	3.5	7.7×10^{-3}	8.3×10^{-3}	6.94×10^{-2}
PSO technique	0.4983	0.1423	3.5	8.7×10^{-4}	2.4×10^{-4}	2.33×10^{-3}

The PSO technique is better than cubic spline interpolation technique because the former is re-planning the path to avoid the static obstacle with the minimum distance and minimum linear velocity as well as the minimum mean square error for each component of the state error for the seven samples from 30 to 36 samples, as shown in Table 6.2.

6.3.3. Case Study 3

The desired square trajectory which has explicitly non-continuous gradient is carried to verify the capability of the proposed cognitive neural predictive controller to re-plan the desired path in order to avoid the static obstacle as follows. From Figure 6.25 it can be seen that the obstacle centre point position (ob_x,ob_y) is (0.5,0). The length L_{ob} = 0.18m and width W_{ob}=0.16m of the obstacle body are used to calculate the vertices points of the obstacle, as shown in Figure 6.4, and built by the obstacle neural network model:

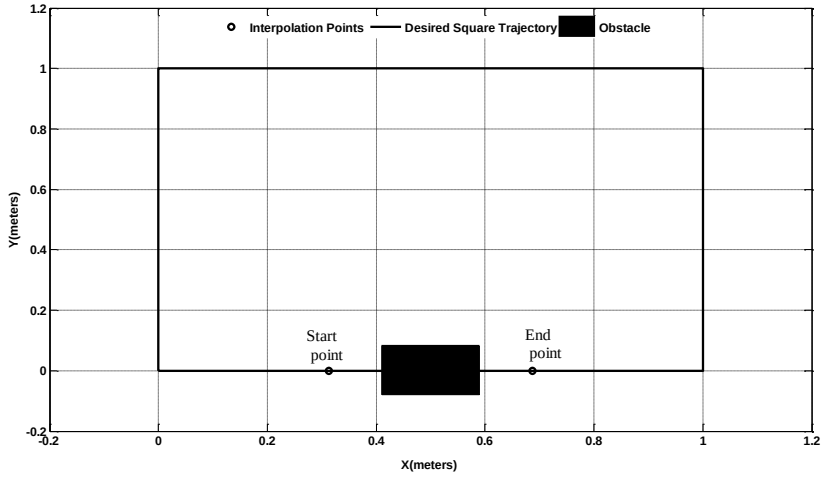


Figure 6.25: The obstacle in the desired square path.

$$IH_1 = x_i - 0.35 \tag{6.29}$$

$$IH_2 = -x_i + 0.65 \tag{6.30}$$

$$IH_3 = y_i - 0.15 \tag{6.31}$$

$$IH_4 = -y_i + 0.15 \tag{6.32}$$

The new path that the mobile robot will track to avoid the obstacle consists of two main points; start point and end point. These two points are determined depending on the obstacle neural network model and they are found as follows: start-point (0.3125, 0) and end-point (0.6875, 0). The best side to re-plan the path is below the obstacle because it will achieve the minimum distance depending on the proposed algorithm, as shown in Figure 6.6. Cubic spline interpolation technique is then applied to find the new smoothness path with non-overshooting behaviour (to reduce the tracking error) as follows:

For first segment, i=1, $0.3125 \le x_i \le 0.499$

$$y_i(x_i) = -1.0695 + 4.977x_i - 4.9778x_i^2 + 3\times10^{-7}x_i^3 \tag{6.33}$$

For second segment, i=2, $0.499 \le x_i \le 0.6875$

$$y_i(x_i) = -1.0695 + 4.9778x_i - 4.9778x_i^2 - 5\times10^{-7}x_i^3 \tag{6.34}$$

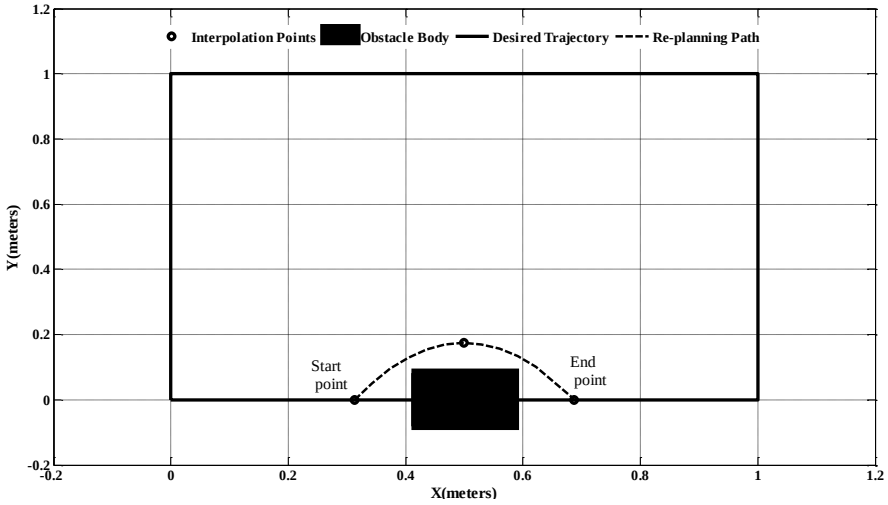


Figure 6.26: Re-planning the desired square trajectory of the mobile robot.

After planning the new path in the cognitive layer (as shown in Figure 6.26), the number of points that the mobile robot must track and the linear velocity that does not exceed the V_{Imax} (equal to 0.165m/sec) should be determined as follows. It is known that the number of points of the deleted track φ is equal to seven points, with start and end points, then the

equation 6.6 is applied to find the travelling time of the two segments track; it is equal to 3.5 sec. Then the distance *D* between start point and end point (two segments track) is determined using equation 6.7; it is equal to 0.536m. Then the linear velocity of the mobile robot is checked to see if it exceeds V_{Imax} or not using equation 6.8; it is equal to 0.153m/sec. The mobile robot model starts from the initial position and orientation $q(0) = [0,-0.1,0]$ as its initial posture and uses the third control methodology with the smallest MSE of the posture. After re-planning the square path using the cognitive neural predictive controller, the trajectory tracking of the mobile robot is obtained as shown in Figures 6.27a and 6.27b.

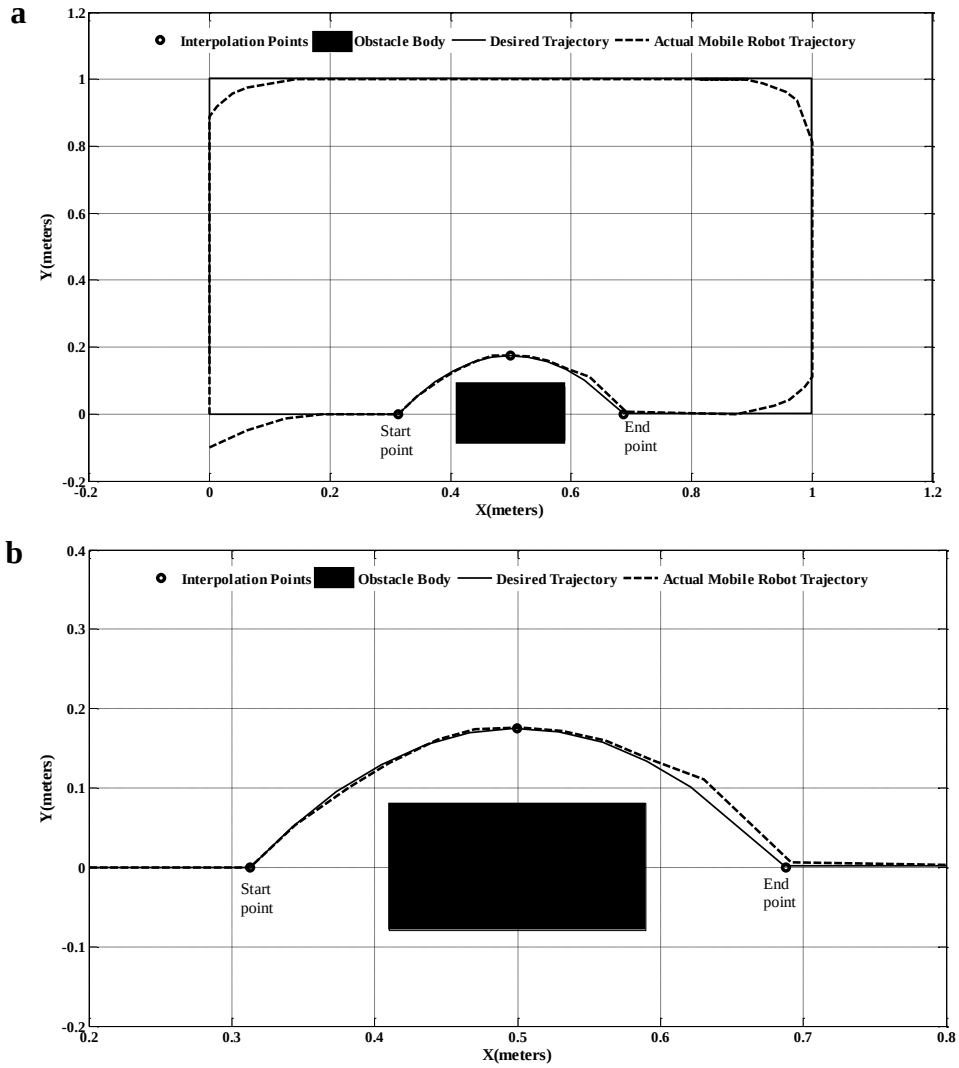


Figure 6.27: Actual trajectory of mobile robot and desired square trajectory with obstacle avoidance: (a) overall trajectory tracking; (b) zoom in during obstacle avoidance.

These Figures demonstrate very good position and orientation tracking performance for obstacle avoidance and minimum tracking error in spite of the existence of bounded disturbances.

The simulation results demonstrated the effectiveness of the proposed cognitive neural predictive controller by showing its ability to generate on-line small values of the control input torques for right and left wheels with small sharp spikes from seven to thirteen samples in order to track the new desired path and to avoid the static obstacle, as shown in Figure 6.28.

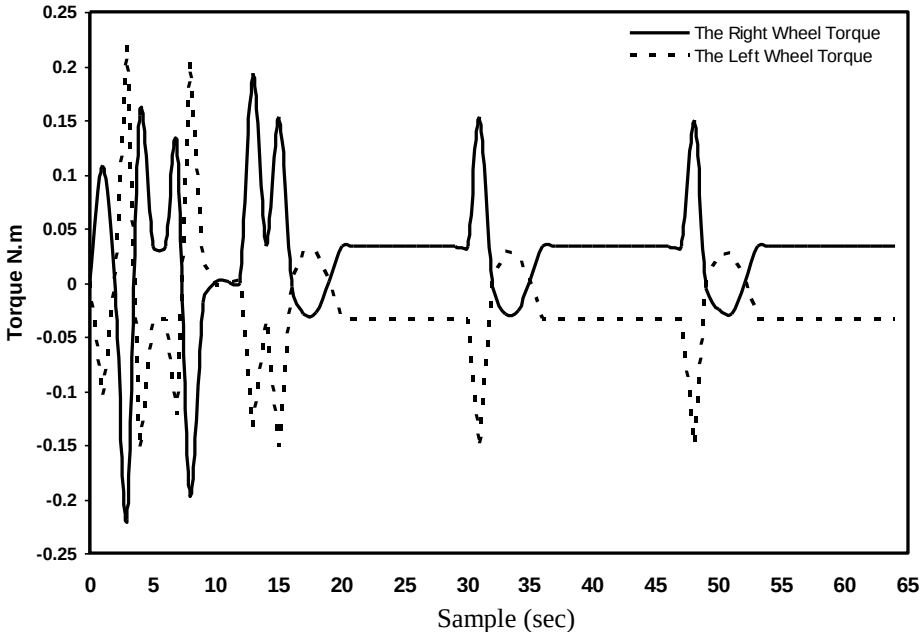
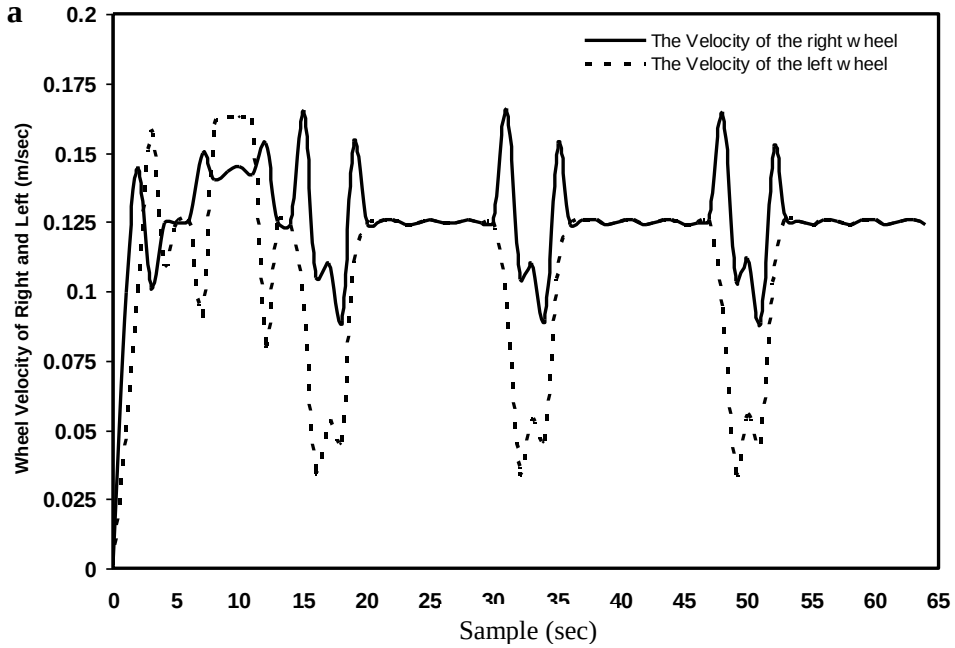


Figure 6-28: The torque of the right and left wheel action.

The velocities of the right and left wheels are changed during seven samples from 7 to 13 samples, as shown in Figure 6.29a. The mean linear velocity of the mobile robot is equal to 0.153m/sec, as shown in Figure 6.29b, and maximum angular velocity is equal to 0.7rad/sec during these seven samples (the static obstacle was avoided). The overall mean linear velocity of the mobile robot is equal to 0.125m/sec.



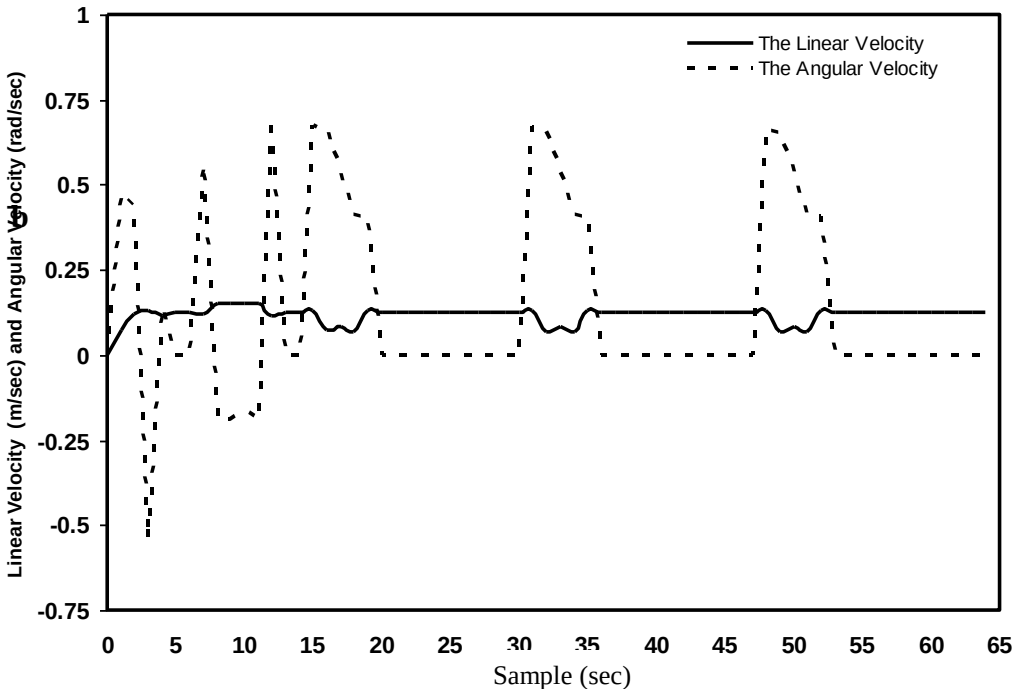


Figure 6.29: Mobile robot velocity: (a) the right and left wheel velocity action; (b) the linear and angular velocity action.

Using PSO technique to find the new smoothness path without overshooting behaviour and minimum distance is shown in Figure 6.30, depending on fitness function as in equation 6.15 from the start point (0.3125, 0) to the end point (0.6875, 0). The size of the particle dimension is equal to five, and the population of particles is equal to 100.

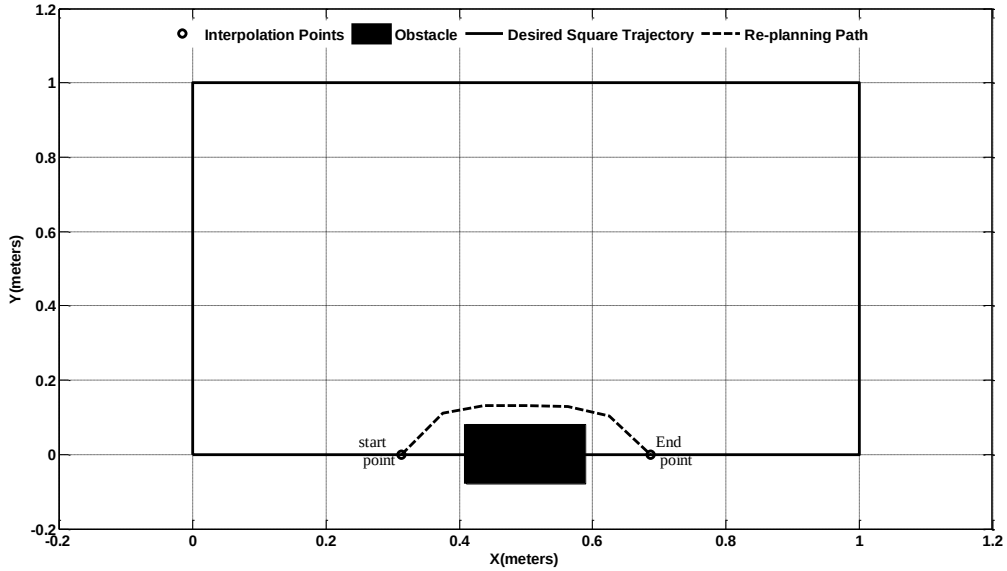


Figure 6-30: Re-planning the desired trajectory of the mobile robot.

The distance between start-point and end-point is equal to 0.5144m using equation 6.14. The linear velocity of the mobile robot is then checked to see whether it exceeds V_{Imax} or not using equation 6.16; it is equal to 0.1469m/sec. After re-planning the square path by the cognitive neural predictive controller with third control methodology, the trajectory tracking of the mobile robot is obtained, as shown in Figures 6.31a and 6.31b.

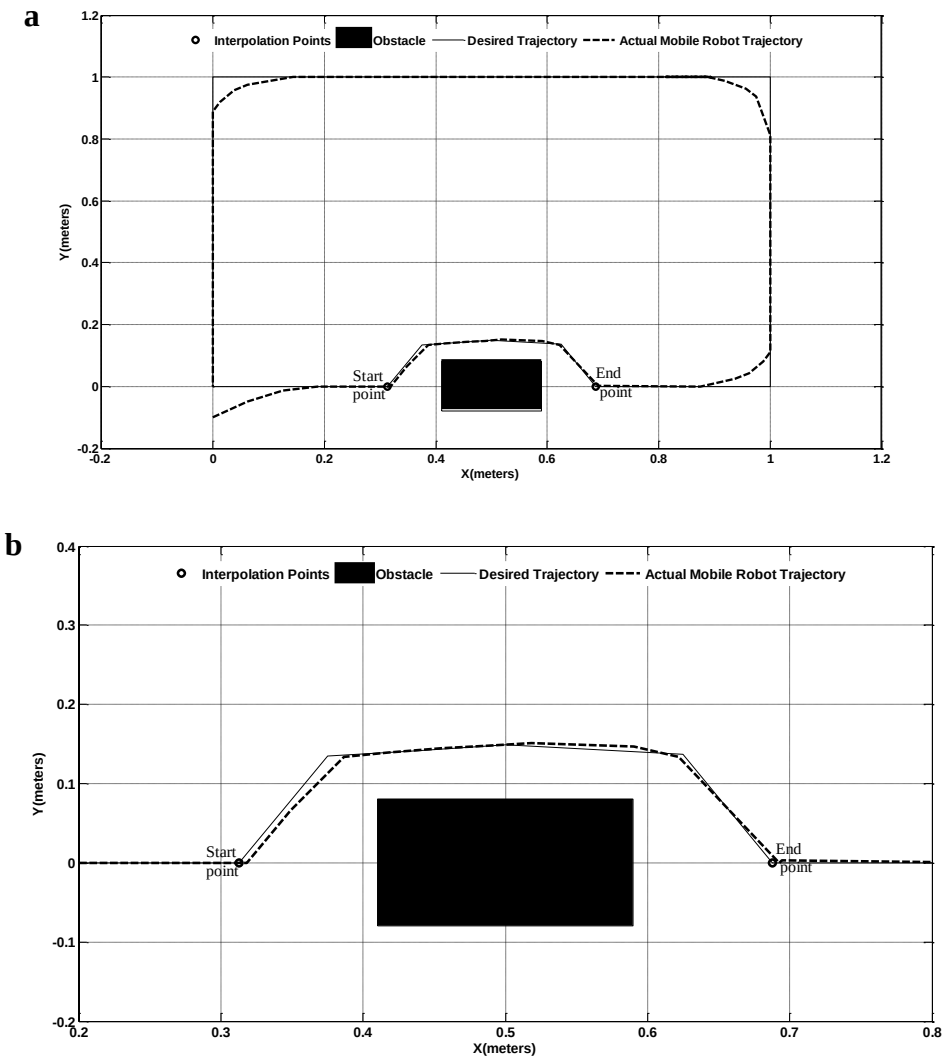


Figure 6.31: Actual trajectory of mobile robot and desired square trajectory with obstacle avoidance: (a) overall trajectory tracking; (b) zoom in during obstacle avoidance.

These Figures demonstrate excellent position and orientation tracking performance for obstacle avoidance and minimum tracking error in spite of the existence of bounded disturbances.

The simulation results demonstrated the effectiveness of the proposed cognitive neural predictive controller by showing its ability to generate on-line small values of the control

input torques for right and left wheels with small sharp spikes from seven to thirteen samples in order to track the new desired path and to avoid the static obstacle, as shown in Figure 6.32.

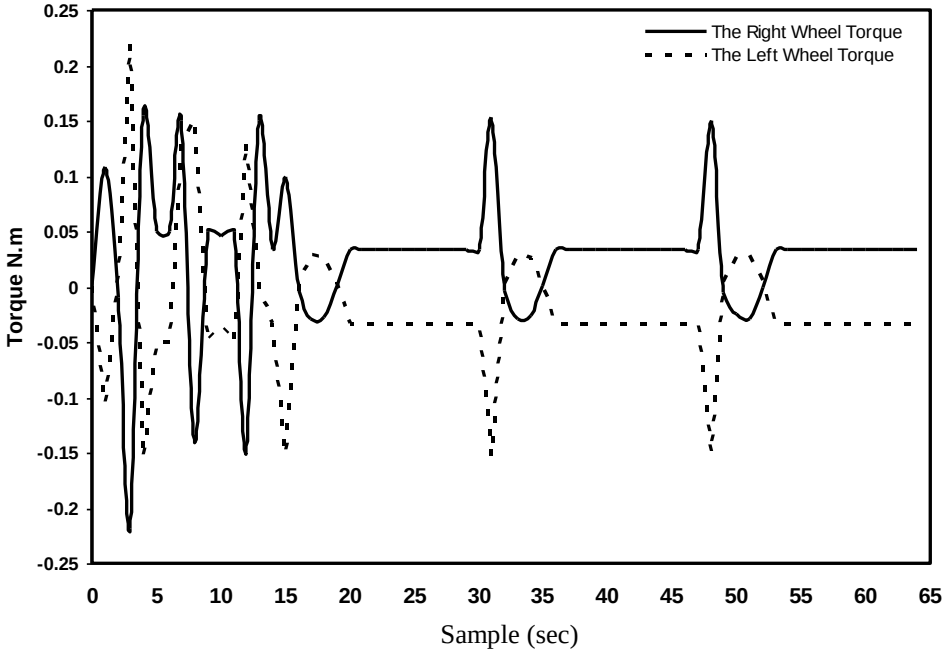
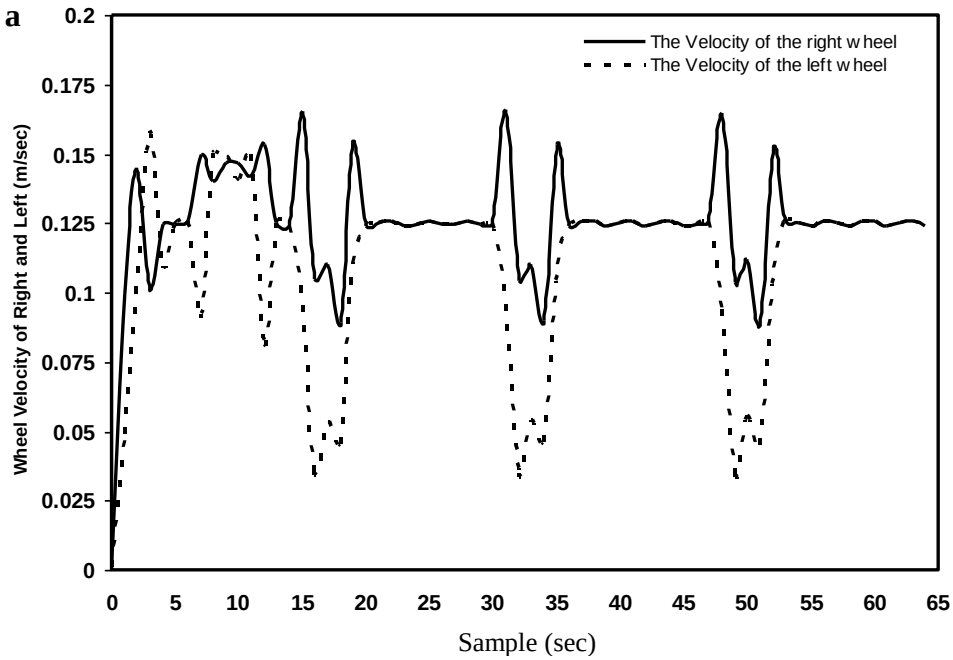


Figure 6-32: The torque of the right and left wheel action.

The velocities of the right and left wheels are changed during seven samples from 7 to 13 samples, as shown in Figure 6.33a.



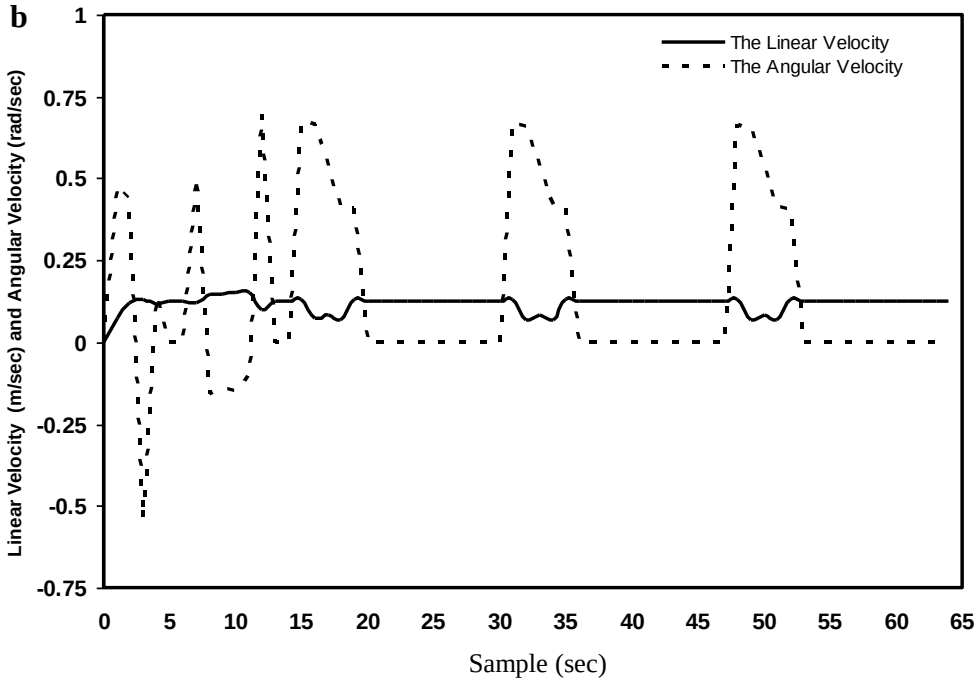


Figure 6.33: Mobile robot velocity: (a) the right and left wheel velocity action; (b) the linear and angular velocity action.

The mean linear velocity of the mobile robot is equal to 0.1469m/sec as shown in Figure 6.33b and maximum angular velocity is equal to 0.75rad/sec during these seven samples; the static obstacle was avoided. The overall mean linear velocity of the mobile robot is equal to 0.125m/sec.

The comparison between the two techniques (numerical and intelligent) to re-plan the desired square trajectory is shown in Table 6.3.

Table 6.3: Numerical and intelligent techniques for minimum distance and linear velocity in the square path.

Techniques	Distance (m)	Velocity (m/sec)	Time (sec)	MSE of X-coordinate	MSE of Y-coordinate	MSE of Orientation
Cubic spline interpolation	0.536	0.153	3.5	9.7×10^{-3}	8.9×10^{-3}	7.89×10^{-2}
PSO technique	0.5144	0.1469	3.5	4.6×10^{-4}	1.7×10^{-4}	2.56×10^{-3}

The PSO technique is better than cubic spline interpolation technique because the former is re-planning the path to avoid the static obstacle with minimum distance and minimum linear velocity as well as minimum mean square error for each component of the state error for the seven samples from 7 to 13 samples as shown in Table 6.3.

6.4. Summary

This chapter gives an overview of the basic concepts of the cognitive neural predictive control structure that are based on two layers; neural network topology layer and cognitive layer. Neural network topology layer is the execution layer because it controls the mechanical actuators in order to follow the mobile robot's desired trajectory fed from the cognitive layer. Cognitive layer is the planning layer, which collects all the information from the environment using multi-sensors in order to detect the obstacle in the desired path and re-plan optimal smoothness desired trajectory to avoid the obstacle. The performance of the proposed cognitive neural control with best control methodology is implemented using Matlab program using two techniques for re-planning the path: cubic spline interpolation as a numerical technique and PSO as an intelligent technique, and they were compared in terms of minimum distance and minimum linear velocity as well as in generating an optimal torque control action despite the presence of bounded external disturbances.

Chapter Seven

Experimental Work

7.1. Introduction

In this chapter, several desired trajectories are tracked from nonholonomic wheeled mobile robot in order to validate the applicability of the proposed cognitive neural control methodologies explained in chapter five. Experiments were executed using an actual mobile robot.

7.2. Boe-Bot Mobile Robot

The laboratory experiments have been conducted using a Boe-Bot mobile robot type nonholonomic wheeled mobile robot (V3), as shown in Figure 7.1.

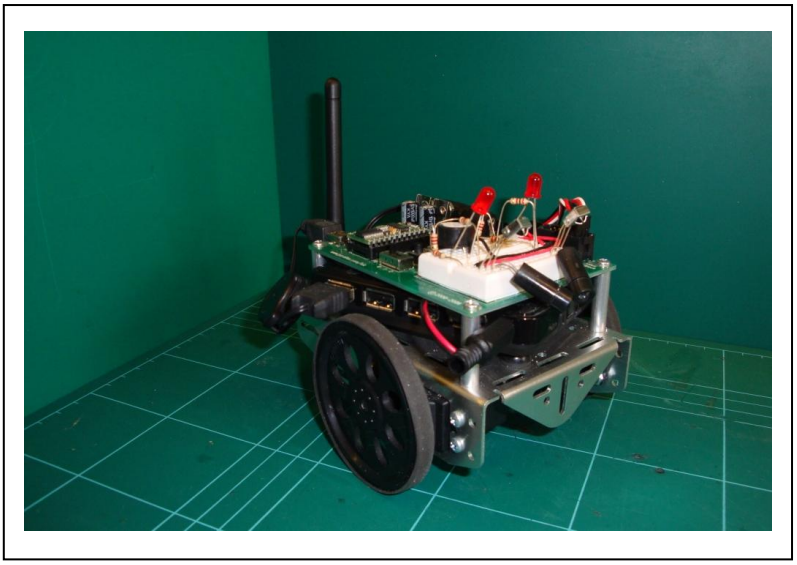


Figure 7.1: Boe Bot mobile robot for the experiments.

The wheeled mobile robot is equipped with BASIC Stamp 2 programmable (BS2) microcontroller type (PIC16C57c) consisting of EEPROM 2kByte, a decoding logic unit, infrared sensors, PWM generator for differential control of the robot [142, 143]. The hardware specifications of the Boe-Bot mobile robot model are summarised in Table 7.1

[142, 143]. The wheel radius includes the o-ring used to prevent slippage; the rubber is stiff enough so that point contact with the ground can be assumed.

Table 7.1: Hardware specifications of the mobile robot.

List	**Specification**
Size (m)	$0.14 \times 0.12 \times 0.1$ (L×W×H)
Weight	0.65kg
Inertia	0.36 kg.m^2
Distance between Wheels	0.105m
Radius of wheel	0.033m

7.2.1. Introducing the Continuous Rotation Servo

The Parallax Continuous Rotation servo shown in Figure 7.2 includes the motors that will make the Boe-Bot's wheels turn. This Figure points out the servos' external parts [142].

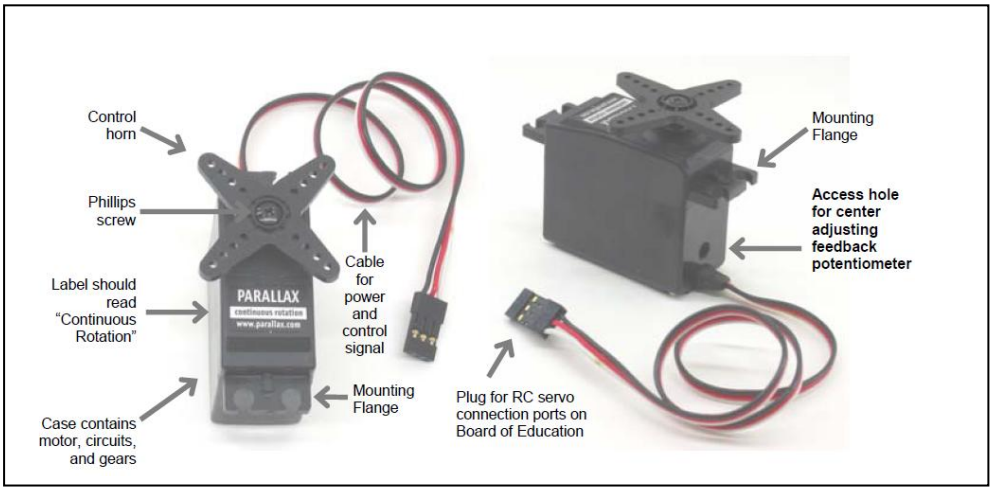


Figure 7.2: Parallax continuous rotation servo [142].

The pulse width controls speed and direction, determined by the duration of the pulse signals that have to be sent to the servo connected to pin 12 or 13 of the BS2 microcontroller. In order for smooth rotation, the servo needs a 20 ms pause between pulses. As the length of the pulse decreases from 1.5 ms, the servo will gradually rotate faster in an anti-clockwise direction, as can be seen in Figure 7.3a. Figure 7.3b is a sample timing diagram for a centred servo. Likewise, as the length of the pulse increases from 1.5 ms, the servo will gradually rotate faster in the clockwise direction, as can be seen in Figure 7.3c.

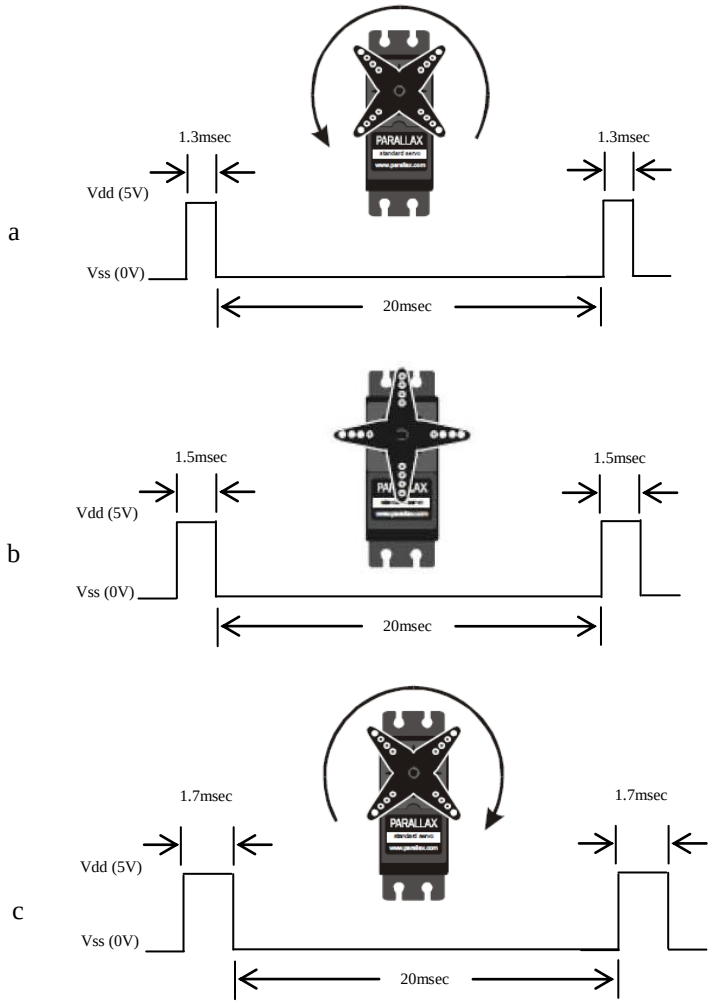


Figure 7.3: The duration of the pulse signals in: (a) anti-clockwise direction; (b) centre servo direction; (c) clockwise direction.

In order to generate these signals from BS2 microcontroller, a PULSOUT and a PAUSE command are used. Figuring out the PAUSE command from the timing diagram is easy; it is PAUSE 20 for the 20msec between pulses and for PULSOUT command, it needs the duration argument as stated in equation 7.1:

$$\text{duration argument} = \frac{Pulse\_Duration}{2\mu\text{sec}} \tag{7.1}$$

Table 7.2 shows the following PULSOUT duration.

Table 7.2: PULSOUT duration

Pulse Duration	Duration Argument	Description
1.3msec	650	Full Speed anti-cw
1.5msec	750	Centre No Rotation
1.7msec	850	Full Speed cw

The general structure of the pulse width controls speed and direction for the mobile robot, as shown in Figure 7.4.

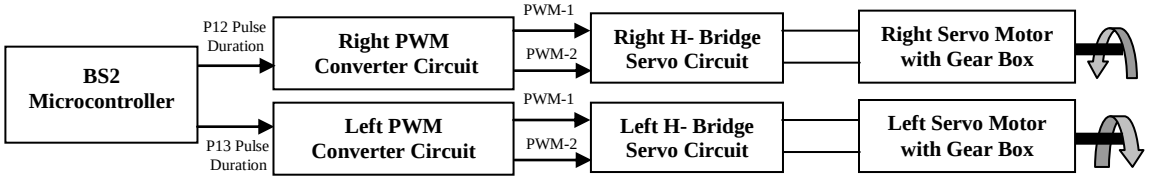


Figure 7.4: The pulse width controls speed and direction structure.

To generate fix sampling time 0.5sec with variable speed of the wheels, equations 7.2 and 7.3 are proposed:

$$T_{TIL}=T_{CL}+T_{CR}+T_{Pd}+T_{Co} \quad (7.2)$$

$$N_P=T_S/T_{TIL} \quad (7.3)$$

where

T_{TIL} is the total time of instructions loop.

T_{CL} is the time of PULSOUT command left wheel.

T_{CR} is the time of PULSOUT command right wheel.

T_{Pd} is the time of PAUSE duration equal to 20msec.

T_{Co} is the time of code overhead equal to 1.6msec.

Np is the number of pulses.

T_S is the sampling time equal to 0.5sec.

However, for each sampling time, a new number of pulses is generated in order to achieve the same sampling time; this is equal to 0.5sec. Figure 7.5 shows the practical transfer curve for the continuous rotation servo of the actual mobile robot after both servo motors are calibrated to make the servo stop turning at 1.5msec pulse, as shown in Figure 7.3b. The horizontal axis shows the pulse width in msec, and the vertical axis shows the rotational velocity in RPM.

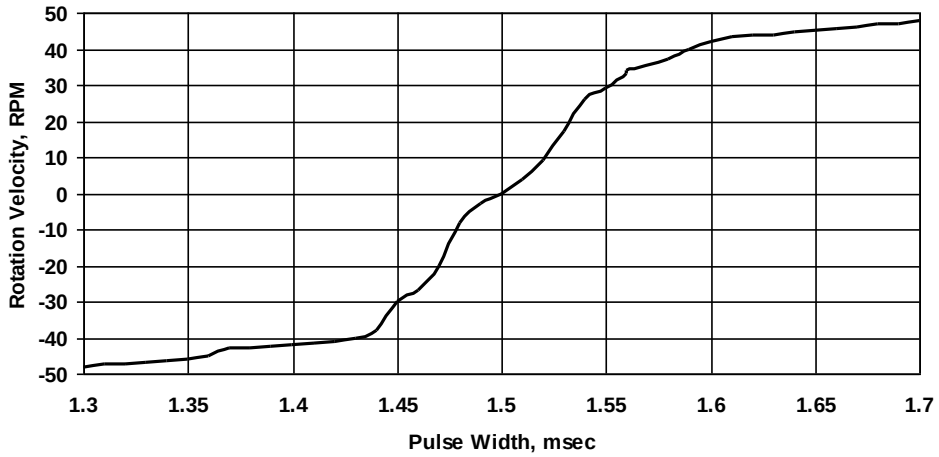


Figure 7.5: The rotation velocity vs. pulse width for servo motor.

In this graph, anti-clockwise direction is negative and clockwise direction is positive. This particular servo's transfer curve ranges from about -48RPM to 48RPM over the range of test pulse widths that range from 1.3msec to 1.7msec.

Figure 7.6 shows the transfer curve for the converting simulation velocity of the left and right wheels to the actual PULSOUT time command for continuous rotation servo. The horizontal axis shows the rotational velocity in RPM and the vertical axis shows the pulse width in msec. In this graph, anti-clockwise direction is negative and clockwise direction is positive. This rotation servo transfer curve ranges from about pulse widths that range from 1.3msec to 1.7msec over the range of -48RPM to 48RPM velocity of the servo motor.

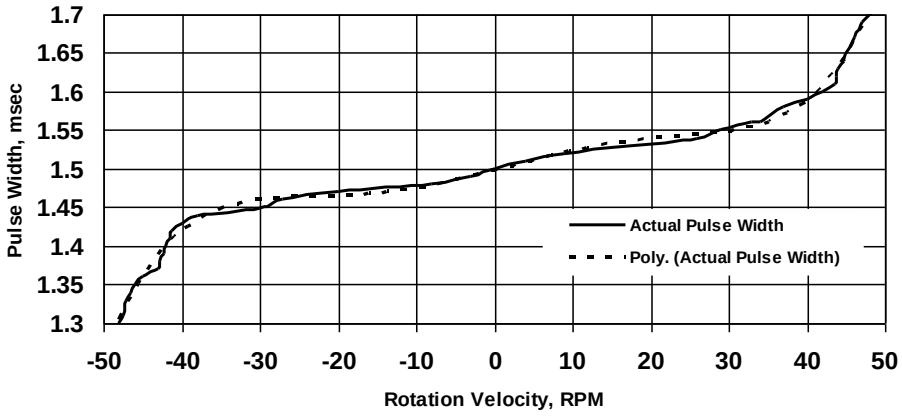


Figure 7.6: The pulse width vs. rotation velocity for servo motor.

In order to convert the velocity of the right and left wheels of the mobile robot to pulse width value, it is proposed the fourth order polynomial equation that describes the variable pulse width (msec) versus rotation velocity (RPM) for the servo motor is as follows:

$$T(V) = (-6\times10^{-7}V^4 + 0.0002V^3 + 0.0014V^2 - 0.0134V + 149.98)/100 \qquad (7.4)$$

A personal computer carries out the cognitive neural predictive control algorithm using MATLAB program then transmits the control data to BASIC Stamp Editor Software version 2.5 in order to convert it to the Boe-Bot mobile robot action format which admits right wheel velocity and left wheel velocity as lookup input duration argument in order to convert them to suitable pulse duration control signals. From the pulse duration, it can be determined the duration argument as equation 7.1, and the number of pulses sent by the

computer are coded messages which are recognised by the microcontroller. Based on received characters, the microcontroller creates control actions for servo motors. The output voltages of the two IR sensors for measuring the distance are converted to coded messages by microcontroller and sent to the personal computer.

The data transmitting between the Boe-Bot robot and main computer is modified from wire to wireless communication using wireless USB Hub and adapter that has radio speed of up to 480Mbps and which is forty times faster than wireless Internet (802.11b) protocol [144]. In addition to that, wireless USB was created for laptop users, allowing high speed wireless connections with little impact on battery life [144]. Lab experiments are carried out by tracking a desired position (x, y) and orientation angle (θ) with a lemniscates, circular and square trajectories in the tracking control of the Boe-Bot robot and re-planning the desired trajectory in order to avoid the static obstacle, rendering collision-free navigation.

7.3. Experiments and Simulation Results

To validate the applicability of the proposed cognitive neural predictive control methodology, the experiments were carried out using the actual Boe-Bot mobile robot in order to track three different types of desired trajectories.

7.3.1. Tracking Lemniscates Trajectory Case Study 1

From simulation results for the tracking desired lemniscates trajectory, which has explicitly continuous gradient with rotation radius changes, the third control methodology has the smallest MSE of the position (X-coordinate, Y-coordinate) and the orientation, compared to the first and second control methodologies, which have the highest MSE for the posture, as shown in Table 5.1. Therefore, it can be considered that the third control methodology was the best methodology for tracking the desired lemniscates trajectory. In the experiments, the best control data action of the simulations was the five steps-ahead action of third control methodology, which has the smallest MSE for posture. The simulation velocity control data action converted from linear velocity (m/sec) to rotation velocity revolution per minute (RPM) is shown in Figure 7.7, using equations 7.5 and 7.6 for the right and left wheels respectively:

$$VR_{RPM} = \frac{V_R}{r} \times \frac{60}{2\pi} \tag{7.5}$$

$$VL_{RPM} = -\frac{V_L}{r} \times \frac{60}{2\pi} \tag{7.6}$$

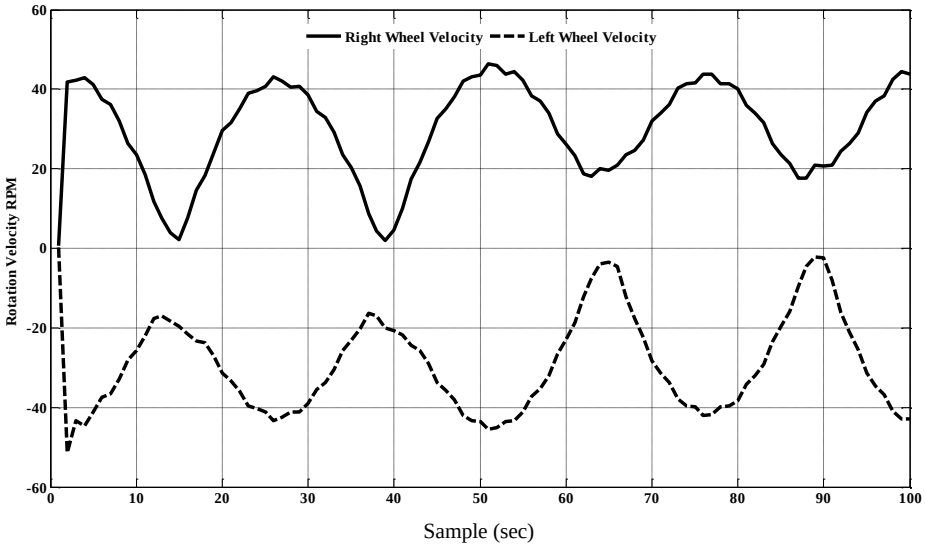


Figure 7.7: The rotation velocity (RPM) for the right and left wheels.

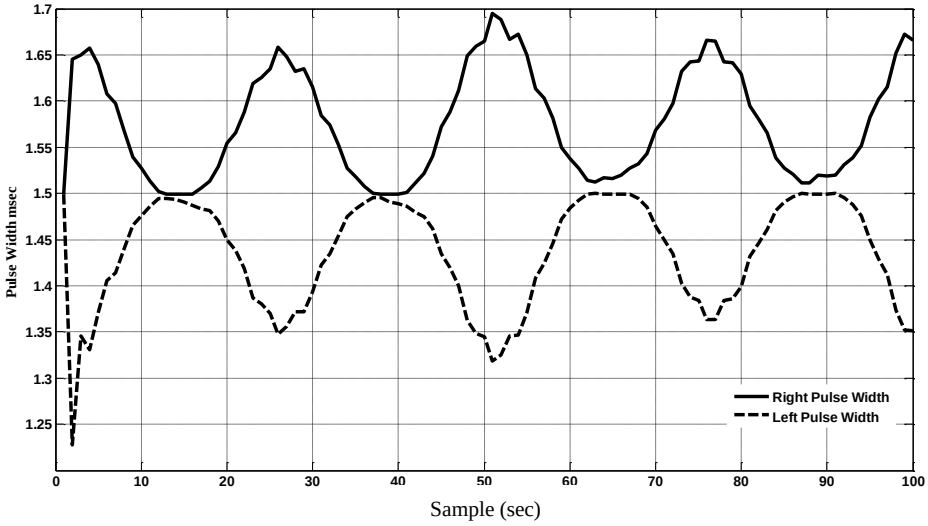


Figure 7.8: The pulse width duration (msec) for right and left PWM converter circuit.

The suitable pulse duration is calculated using equation 7.4, as shown in Figure 7.8, and finally duration argument is found using equation 7.1, as shown in Figure 7.9. Figure 7.10 shows the number of pulses for each sample. However, the number of pulses should be an integer number; therefore, it is equal to 20, in order to keep the sampling time equal to 0.5sec.

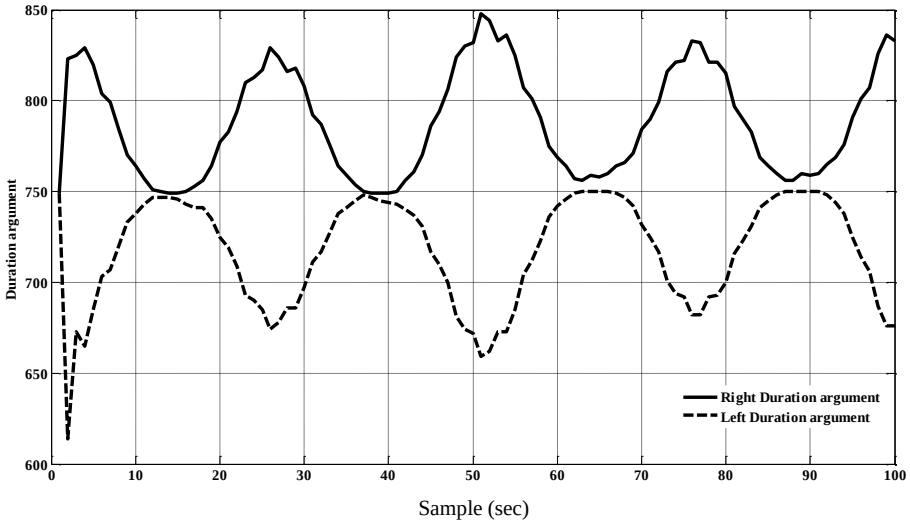


Figure 7.9: The duration argument for the right and left PULSOUT command.

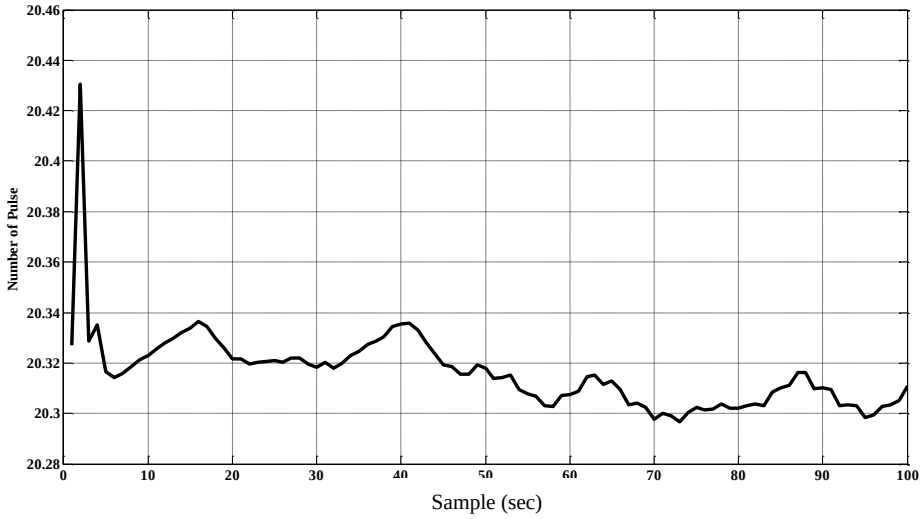


Figure 7.10: The number of pulses to keep sampling time 0.5sec.

The data format was converted from MATLAB file of simulations to BASIC Stamp Editor Software version 2.5 format as a lookup table, and transmitted to the Boe-Bot mobile robot model, which admits right wheel velocity and left wheel velocity as input reference signals by using wireless USB hub communication. The duration argument of the simulation results for right and left wheels downloaded to the memory of the Boe-Bot mobile robot as velocity commands, which have smooth values without sharp spikes, as shown in Figure 7.9. The initial pose for the Boe-Bot mobile robot starts at position (0.75-0.25m) and orientation 1.57rad, and should follow the desired lemniscates trajectory as shown in Figure 7.11.

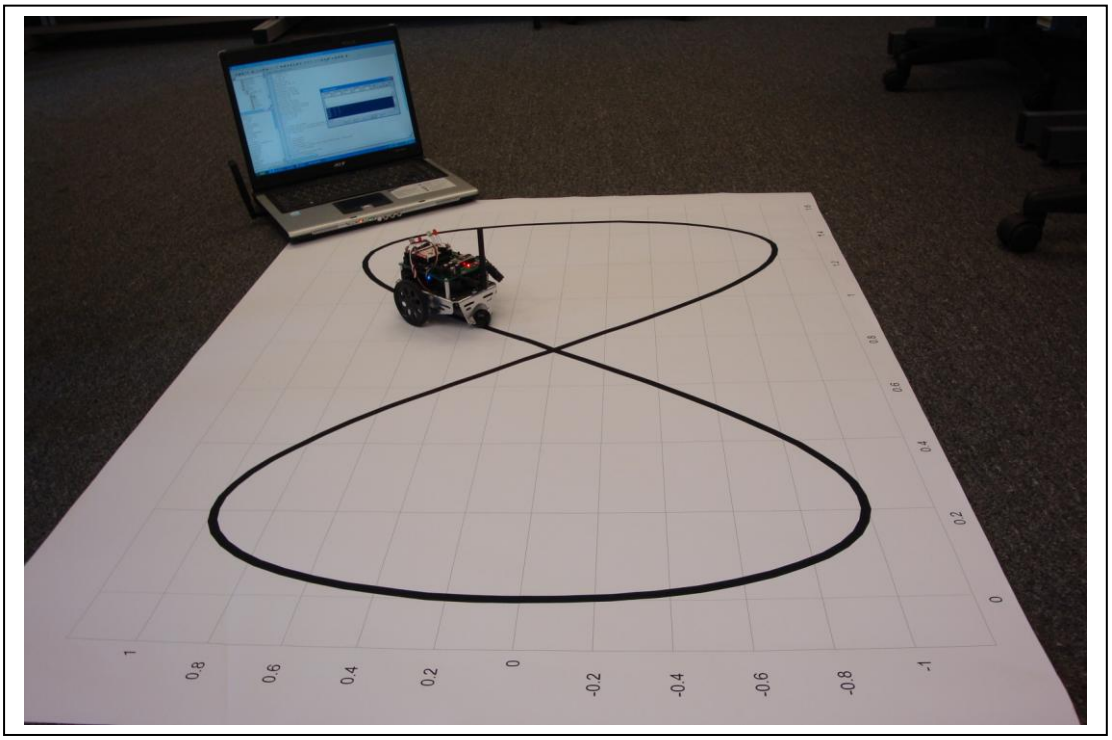


Figure 7.11: Real set-up experiment of Boe-Bot robot for lemniscates trajectory tracking.

The desired trajectory starts at position (0.75, 0m). After 50sec the mobile robot has finished the tracking of the desired path and the tracking was reasonably accurate because the maximum tracking error in the (X-Y) coordinate trajectory were equal to ± 0.0175m, while the maximum tracking error in the orientation of the trajectory was equal to ± 0.5rad and the mean-square error for each component of the state error vector $(q_r - q) = (e_x, e_y, e_\theta)$, is $MSE(q_r - q) = (0.00018, 0.00073, 0.00136)$ respectively.

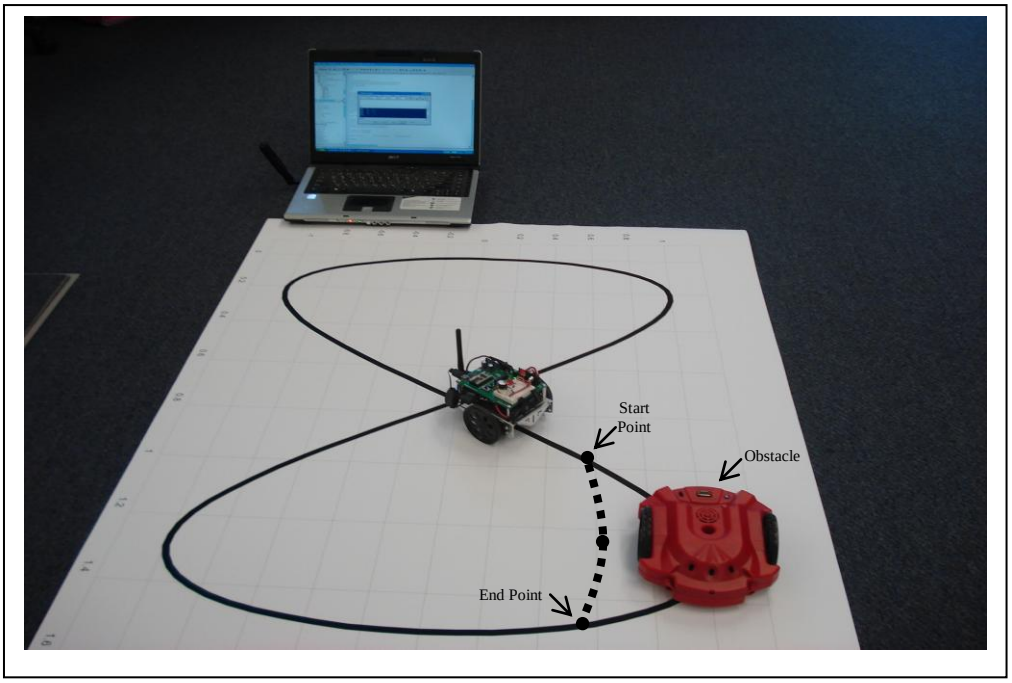


Figure 7.12: Real set-up experiment of Boe-Bot robot for lemniscates trajectory tracking with obstacle avoidance.

A static obstacle is placed in the desired path in order to verify the robustness and adaptation of the cognitive neural predictive controller and its capability to make the mobile robot track any trajectory through re-planning the desired path to avoid the static obstacle with minimum tracking error. Experiments were conducted using an actual mobile robot required to track the desired lemniscates trajectory with obstacle avoidance, as shown in Figure 7.12. The desired trajectory started at position (0.75, 0m) and after 50sec the mobile robot finished the tracking of the desired path. The tracking was reasonably accurate, especially during new path track to avoid the static obstacle and the mean-square error, is $MSE(q_r - q) = (0.00019, 0.00076, 0.00139)$. The duration argument of the simulation results for right and left wheels that make the mobile robot to avoid the static obstacle have downloaded to the memory of the Boe-Bot mobile robot as velocity commands, which have smooth values without sharp spikes, and can be shown in Figure 7.13.

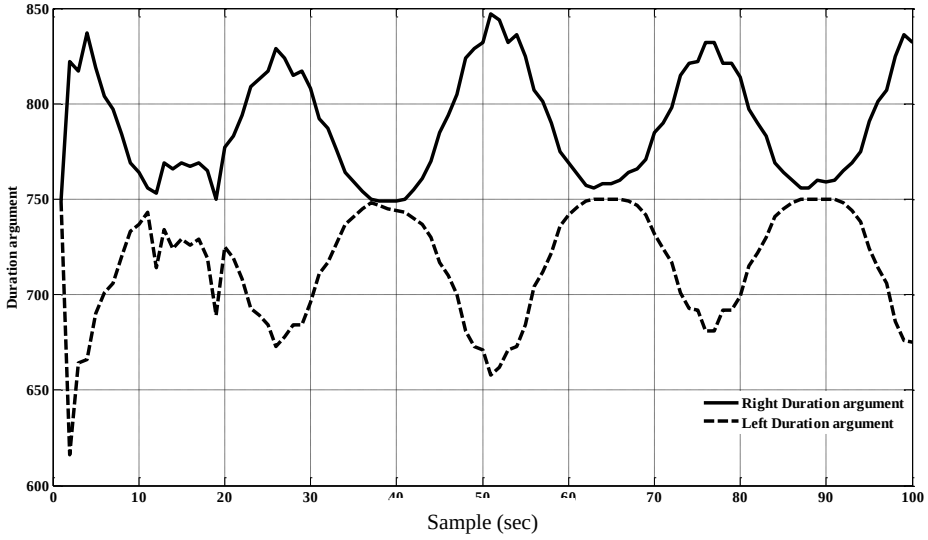


Figure 7.13: The duration argument for the right and left PULSOUT command for obstacle avoidance.

7.3.2. Tracking Circular Trajectory Case Study 2

In order to verify the applicability of the proposed control methodology to track the continuous gradient with rotation radius constant, an experiment was conducted for desired circular trajectory by using Boe-Bot mobile robot. According to the simulation results of Table 5.2 in chapter five, the second control methodology has the smallest MSE of the (X-coordinate and Y-coordinate).

Therefore, the second control methodology was the best methodology for the desired circular trajectory. In the experiments, the best velocities control data action of simulation was the five steps-ahead of the second control methodology, which has the smallest MSE. These control data were transmitted to the Boe-Bot mobile robot model, which admits right wheel velocity and left wheel velocity as input reference signals and pulse width duration, as shown in Figures 7.14 and 7.15 respectively.

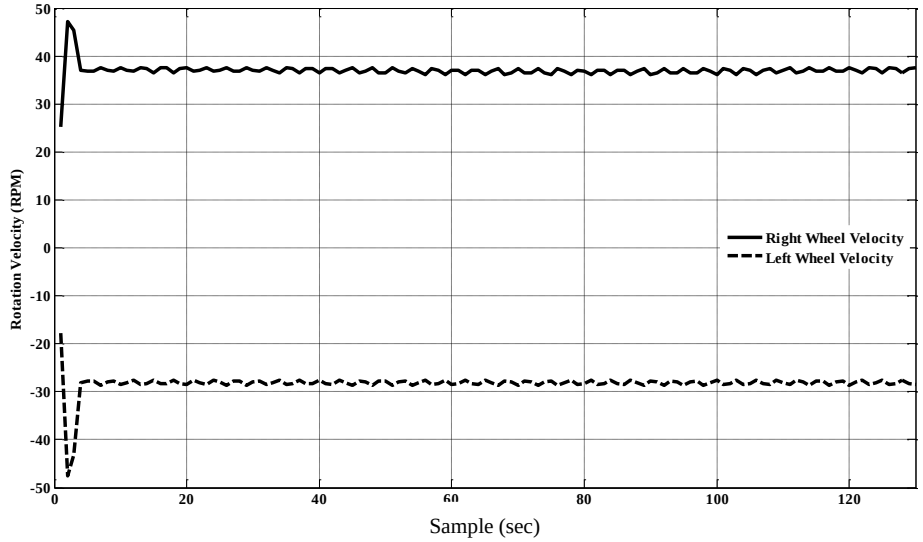


Figure 7.14: The rotation velocity for the right and left wheels.

Wireless USB communication was used to download the memory of the mobile robot after it converted the data format from MATLAB file of simulations to BASIC Stamp Editor Software version 2.5 format, as a lookup table for duration argument data, as shown in Figure 7.16.

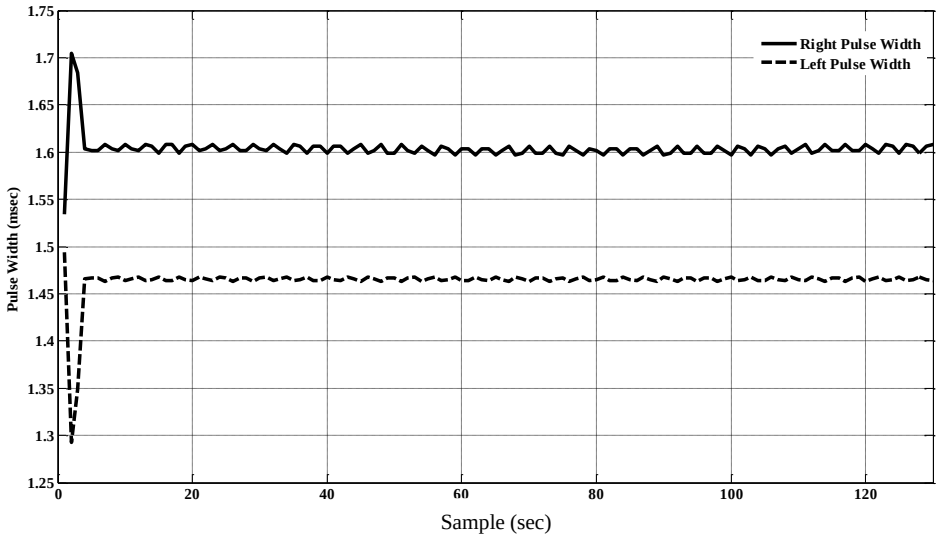


Figure 7.15: The pulse width duration for right and left PWM converter circuit.

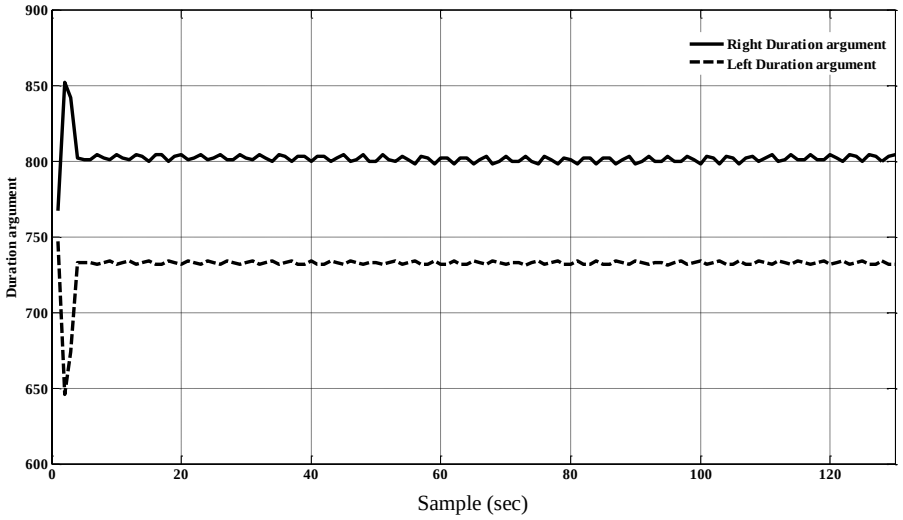


Figure 7.16: The Duration argument for the right and left PULSOUT command.

The initial pose for the Boe-Bot mobile robot starts at position (1.1, -0.5m) and orientation 1.57rad, and should follow desired circular trajectory as show in Figure 7.17. The desired trajectory starts at position (1, 0m). After 65sec the mobile robot has finished the tracking of the desired trajectory with good performance tracking, because the maximum tracking error in the (X-Y) coordinate trajectory was equal to ± 0.01m, while the maximum tracking error in the orientation of the trajectory was equal to ± 0.25rad and the mean-square error for each component of the state error $(q_r - q) = (e_x, e_y, e_\theta)$ was $MSE(q_r - q) = (0.000302, 0.000233, 0.00145)$ respectively.

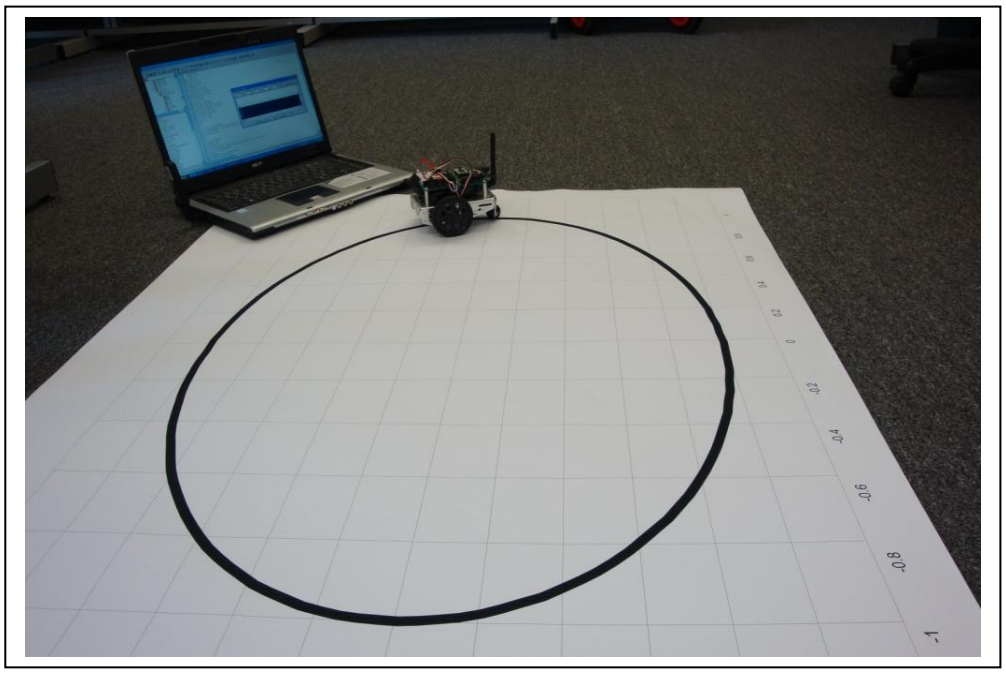


Figure 7.17: Real set-up experiment of Boe-Bot robot for circular trajectory tracking.

To investigate the applicability of the proposed control methodology for obstacle avoidance and tracking the desired circular trajectory, experiments were carried out as shown in Figure 7.18. The desired trajectory starts at position (1, 0m) and after 65sec the mobile robot finished tracking the desired path and the tracking was reasonably accurate, especially during new path track to avoid the static obstacle and the mean-square error for each component of the state error was $MSE(q_r - q) = (0.000311, 0.000237, 0.00149)$.

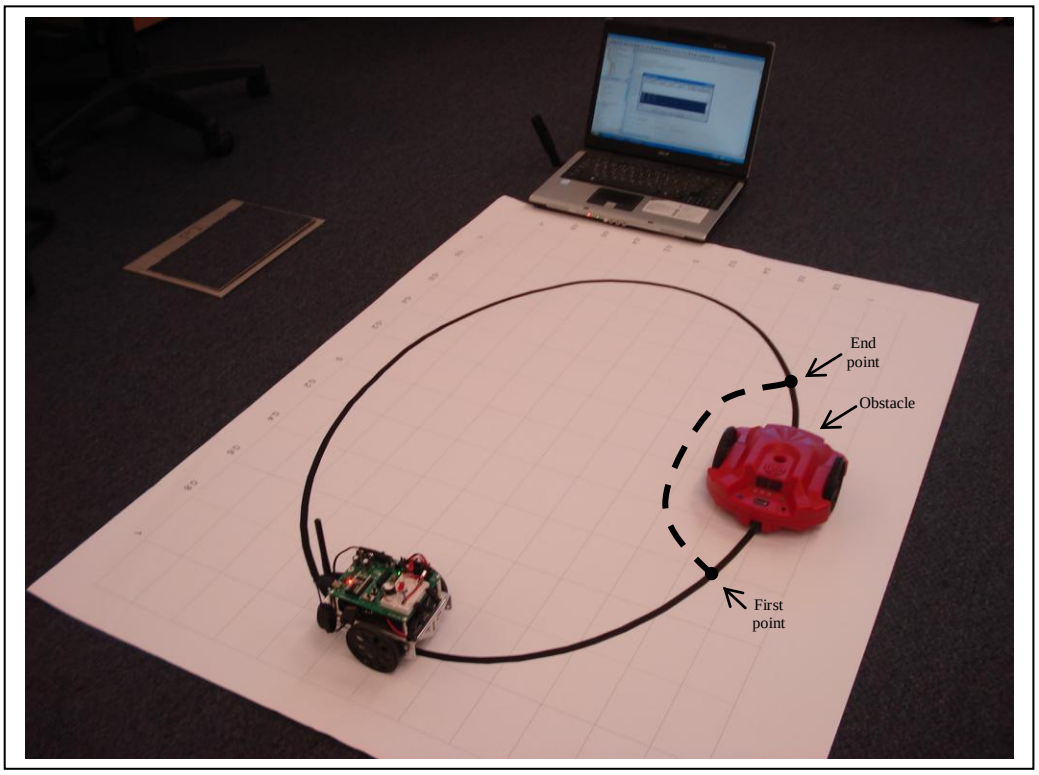


Figure 7.18: Real set-up experiment of Boe-Bot robot for circular trajectory tracking with obstacle avoidance.

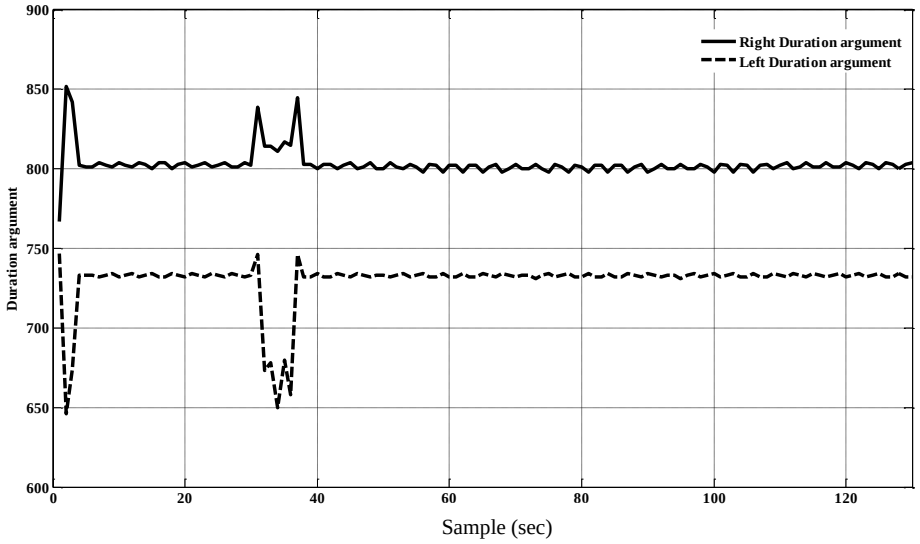


Figure 7.19: The duration argument for the right and left PULSOUT command for obstacle avoidance.

The duration argument of the simulation results for right and left wheels that make the mobile robot avoid the static obstacle have downloaded to the memory of the Boe-Bot mobile robot as velocity commands which have smooth values without sharp spikes, as shown in Figure 7.19.

7.3.3. Tracking Square Trajectory Case Study 3

From simulation results for the tracking desired square trajectory which has explicitly non-continuous gradient, the second and the third control methodologies have the smallest MSE of the position (X-coordinate, Y-coordinate) and the orientation compared to the first control methodology, which has high MSE for the posture, as shown in Table 5.3. The third control methodology was used to validate the applicability of the proposed cognitive neural predictive controller in the experimental work, in which the mobile robot was required to track desired square trajectory. The best control data action of simulations was the five steps-ahead action of third control methodology, as shown in Figure 7.20, which transmitted to the Boe-Bot mobile robot model, which admits right wheel velocity and left wheel velocity as input reference signals as lookup table duration argument data, as shown in Figure 7.21. In the experiment, the Boe-Bot mobile robot started at the initial position 0 and -0.1 meter and initial orientation 0rad, was to follow the desired square trajectory shown in Figure 7.22.

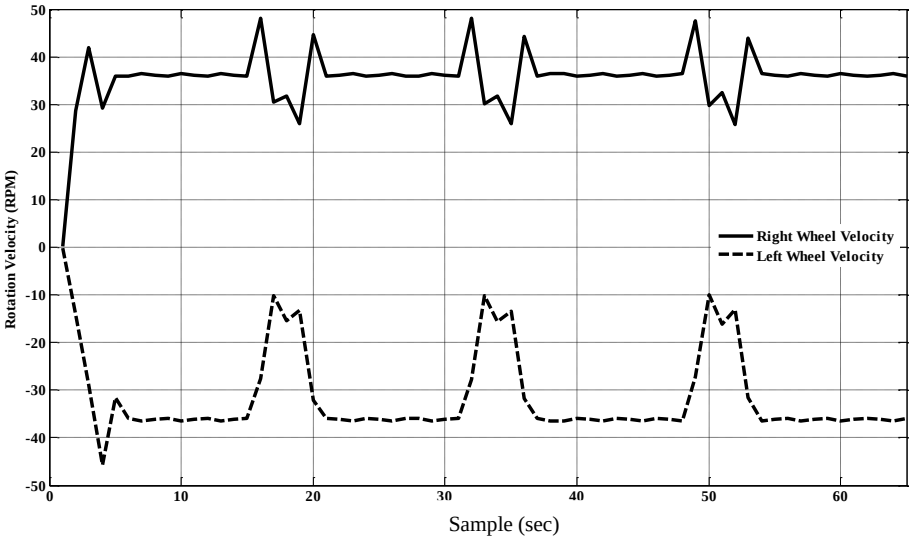


Figure 7.20: The rotation velocity (RPM) for the right and left wheels.

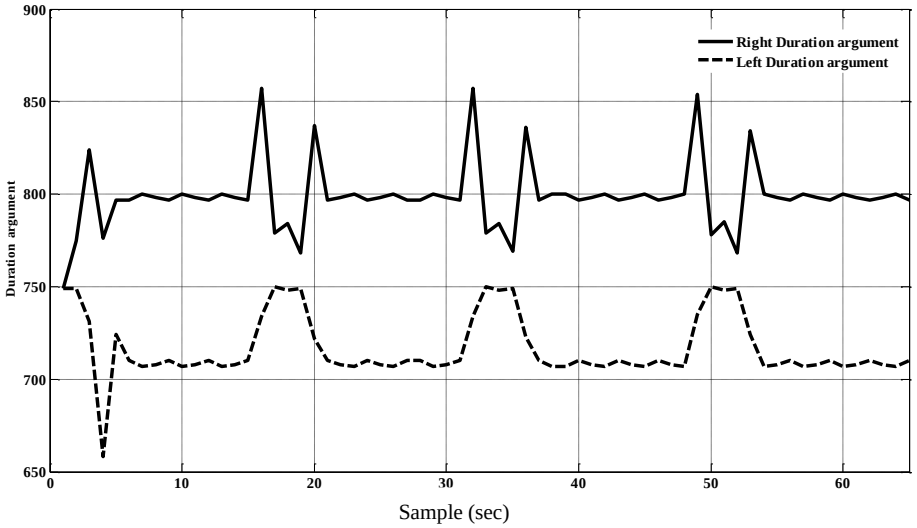


Figure 7.21: The Duration argument for the right and left PULSOUT command for obstacle avoidance.

The desired trajectory starts at position (0, 0m). After 32.5sec the mobile robot finished the tracking of the desired trajectory with good performance tracking, because the maximum tracking error in the (X-Y) coordinate trajectory was equal to ± 0.05m at the end of one side of the square trajectory while the maximum tracking error in the orientation of the trajectory was equal to ± 0.5rad because the desired orientation angle changes suddenly at each corner. The mean-square error for each component of the state error $(q_r - q) = (e_x, e_y, e_\theta)$ was *MSE(*q_r*-q)*=(0.000258, 0.000227, 0.00787) respectively.

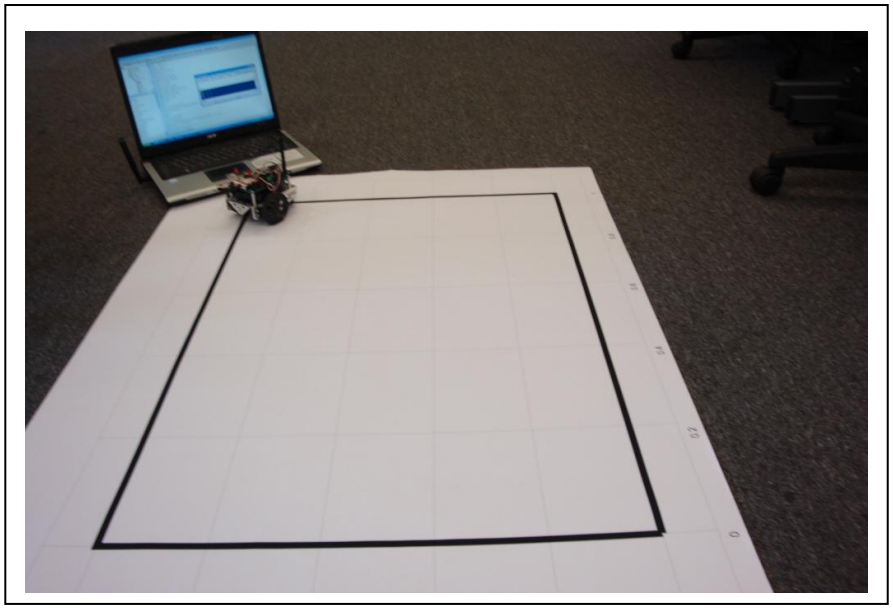


Figure 7.22: Real set-up experiment of Boe-Bot robot for square trajectory tracking.

In order to investigate the applicability of the proposed cognitive neural control methodology, experiments were carried out using the actual mobile robot required to track the desired square trajectory with obstacle avoidance, as shown in Figure 7.23.

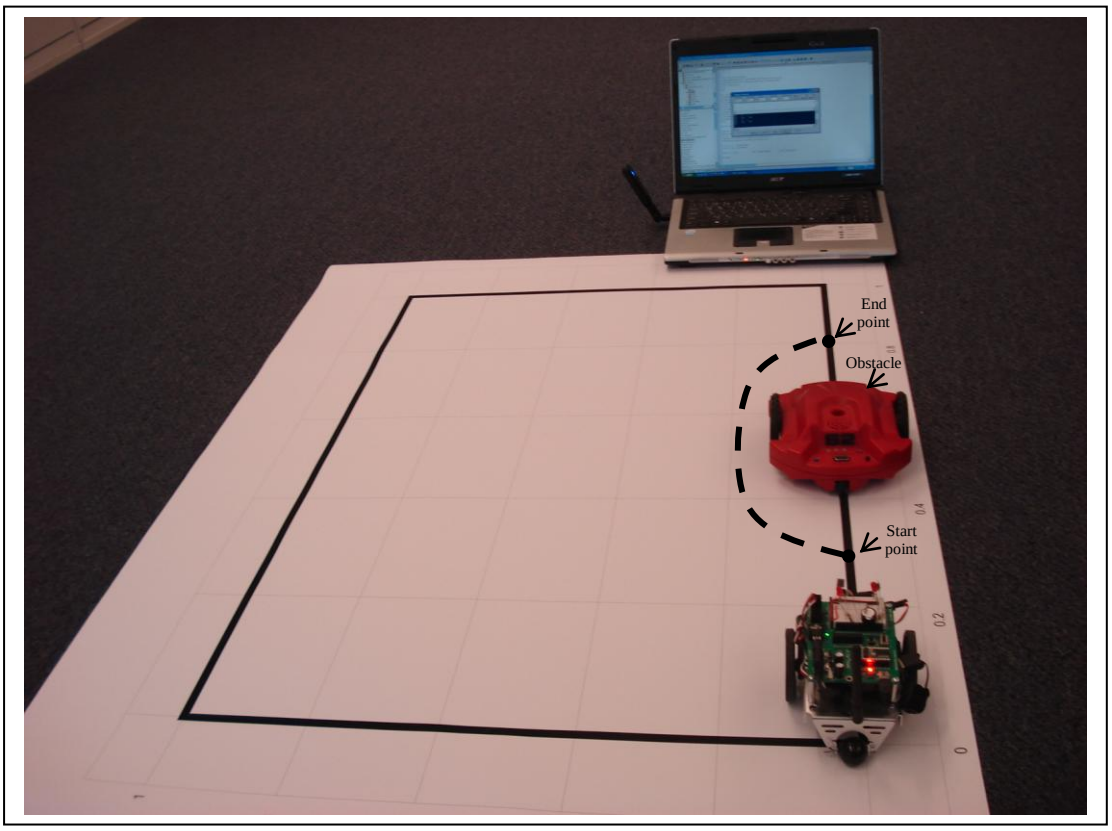


Figure 7.23: Real set-up experiment of Boe-Bot robot for square trajectory tracking with obstacle avoidance.

The desired trajectory starts at zero position and after 32.5sec the mobile robot finished the tracking of the desired path; the tracking was reasonably accurate, especially during new path track to avoid the static obstacle and The mean-square error for each component of the state error was $MSE(q_r\text{-}q)$=(0.000263, 0.000236, 0.00799).

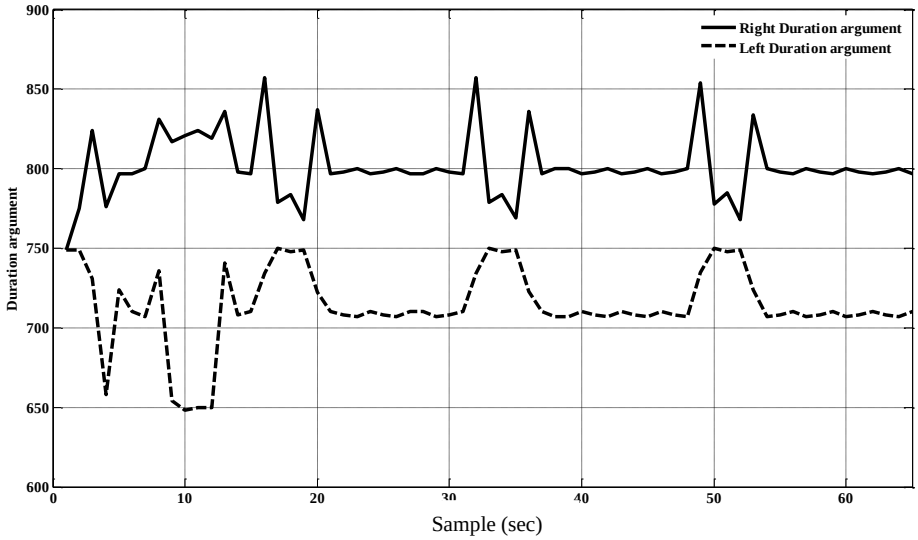


Figure 7.24: The Duration argument for the right and left PULSOUT command for obstacle avoidance.

The duration argument of the simulation results for right and left wheels that make the mobile robot avoid the static obstacle downloaded to the memory of the Boe-Bot mobile

robot as velocity commands with smooth values and without sharp spikes, as shown in Figure 7.24

The difference between simulations results and experimental results caused the residual errors in the experimental results due to the inherent friction present in the real system, especially during tracking the non-continuous gradient path and modelling errors, due to the difficulty of estimating or measuring the geometric, kinematics or inertial parameters, or from incomplete knowledge of the components of the system.

In addition to that, calibration and alignment of the IR sensors for reading X-Y coordinate of the mobile robot trajectory caused some error readings which were not presented in the simulation. From the simulation results and lab experiments, the five steps-ahead predictive for each control methodology gives better control results, which is expected because of the more complex control structure, and also due to taking into account future values of the desired, not only the current value, as with one step-ahead.

The percentage of the mean square error between simulation results for five steps-ahead predictions and experimental work for three different types of the desired trajectories without static obstacle in the path, as shown in Table 7.3.

Table 7.3: The percentage of MSE between simulation results and experimental work without obstacle.

Desired Trajectory	**Lemniscates Trajectory**	**Circular Trajectory**	**Square Trajectory**
(MSE of X-coordinate) 100%	16.6%	5.9%	6.9%
(MSE of Y-coordinate) 100%	16.6%	13.7%	7.9%
(MSE of Orientation) 100%	8.1%	10.3%	5.3%

The percentage of the mean square error between simulation results for five steps-ahead predictions and experimental work for three different types of the desired trajectories with static obstacle in the desired path, as shown in Table 7.4.

Table 7.4: The percentage of MSE between simulation results and experimental work with static obstacle in the desired path.

Desired Trajectory	**Lemniscates Trajectory**	**Circular Trajectory**	**Square Trajectory**
(MSE of X-coordinate) 100%	21.5%	8.6%	8.7%
(MSE of Y-coordinate) 100%	19.7%	17.9%	11.4%
(MSE of Orientation) 100%	10.1%	12.7%	6.7%

7.4. Summary

This chapter introduced the rotation servo motor of the actual Boe-Bot mobile robot from Parallax Inc used in the experimental work with three case studies. The first case study was based on lemniscates trajectory, which has explicitly continuous gradient with rotation radius changes. The second case study was based on circular trajectory, which has explicitly continuous gradient with rotation radius constant. The third case study was based on square trajectory, which has explicitly non-continuous gradient with collision free navigation. In the experiments, the best control data action of simulations of the best control methodology was transmitted to the actual mobile robot model, which admits right wheel velocity and left wheel velocity using wireless USB communication to download in the memory of the mobile robot after converting the data format from MATLAB file of simulation to BASIC Stamp Editor Software version 2.5 format.

Chapter Eight

Conclusions and Suggested Future Work

8.1. Conclusions

A cognitive neural predictive trajectory tracking control for nonholonomic wheeled mobile robot has been presented in this thesis. The proposed controller structure consists of two layers: the execution layer and cognition path planning layer. The execution layer is a neural network system that controls the mobile robot actuators in order to track a desired path, which consisted of two neural networks models that describe the kinematic and dynamic mathematical model with velocity constraints of the nonholonomic wheeled mobile robot system and a feedforward neural controller. The models are modified Elman recurrent neural network and feedforward multi-layer perceptron respectively. The position and orientation identifier based on the modified Elman recurrent neural network model is trained off-line and on-line stages to guarantee that the outputs of the model will accurately represent the actual outputs of the mobile robot system and the number of nodes in the input, hidden, context and output layers are 5, 6, 6 and 3 respectively.

The feedforward neural controller structure is based on multi-layer perceptron neural network and the number of nodes in the input, hidden and output layers are 8, 16 and 2 respectively. It is trained off-line and its weights are adapted on-line to find the reference torques, which control the steady-state outputs of the mobile robot system. The feedback neural controller is based on the posture neural identifier and quadratic performance index predictive optimisation algorithm for N step-ahead prediction in order to find the optimal torque action in the transient-state to stabilise the tracking error of the mobile robot system when the trajectory of the robot drifts from the desired path during transient state.

The three different types of proposed control methodologies were used in the structure of the cognitive neural predictive controller as follows. The first control methodology of the feedback controller is neural predictive feedback controller, which consists of position and orientation neural network identifier with predictive optimisation algorithm. The second control methodology of the feedback controller is nonlinear PID neural predictive feedback controller, which consists of position and orientation nonlinear PID controllers. The position nonlinear PID controller depends on the x-coordinate error and y-coordinate error, while the orientation nonlinear PID controller depends on the θ-angular error and posture identifier with predictive optimisation algorithm. The third control methodology of the feedback controller is nonlinear inverse dynamic predictive feedback neural controller, which consists of the nonlinear feedback acceleration control equation based on Lyapunov stability method and posture neural network identifier with optimisation predictive algorithm.

The second layer in the structure of the proposed controller is cognition path planning layer, which collects all the information from the environment and plans the optimal smoothness desired path. In addition to this, it detects if there is any obstacle in the desired path in order to avoid the static obstacle by re-planning the desired trajectory based on two techniques - spline interpolation numerical technique and artificial particle swarm optimisation technique - then feeds the desired optimal posture to the first layer (neural control layer).

The main advantages of the presented approach are to plan an optimal or feasible path, avoiding obstructions, and the incorporation of AI neural networks structure in each of the traditional and modern control structures that lead to the development and improvement of the performance of these controllers through adaptation of the control gains parameters, such as (k_p, k_i, k_d) for PID controller and (k_x, k_y, k_θ) for inverse dynamics controller, because of the analytically derived control law which has significantly high computational accuracy with predictive optimisation technique. Therefore, the second and third control methodologies are better than first control methodology, because they generated smoothness optimal torques control action and led to minimum tracking errors of the mobile robot for different types of trajectory.

The proposed control algorithm for the nonholonomic wheeled mobile robot has the capability of tracking any trajectory, such as lemniscates trajectory, which has explicitly continuous gradient with rotation radius changes; and circular trajectory, which has explicitly continuous gradient with rotation radius constant and non-continuous gradients square trajectory.

From simulation results and experimental work, the five steps-ahead prediction control action was better than the one step-ahead for the trajectory tracking of the mobile robot for continuous and non-continuous gradient path with the generation of small smooth values of the control input torques for right and left wheels without sharp spikes, through using actual wireless mobile robot type Boe-Bot robotics. The proposed control algorithm has the capability of static obstacle avoidance by re-planning path with optimal smoothness trajectory based on cubic spline interpolation technique and particle swarm optimisation technique for minimum tracking error for mobile robot, in order to reduce the travelling time and travel distance without exceeding the maximum velocity of the mobile robot's wheels.

8.2. Suggested Future Work

The following points are suggested for the development of the work done in this thesis such as:

1- Simulation or an on-line application of the proposed controller to another type of mobile robot, such as a holonomic wheeled mobile robot and using dynamic obstacles instead of static obstacles in the environment.

2- Using fast back propagation algorithm to learn the off-line and on-line posture identifier of the kinematics and dynamics model of the mobile robot under investigation to obtain higher learning speeds for the neural networks needed for other types of mobile robot that have smaller sampling times that are less than 1msec.

3- Another method can be worked upon by selecting a different performance index to minimise the tracking error and to find optimal control effort.

4- Neural-fuzzy, wavelet neural network and genetic algorithm may be useful for building the posture identifier model.

The following points are suggested for future work:

1- Using multiple mobile robots in master and slave configuration for trajectory tracking control system and obstacles avoidance based on hardware and software. The hardware consists of two mobile robots type NI RIO 9632 while Labview software package is used to build the robust neural predictive controller in order to guide the master and slave mobile robots implementing intelligent path planning method.

2- Swarm of mobile robots capable of self-assembling, self-organizing and self-planning in order to solve problems that a single mobile robot is unable to solve. These mobile robots should combine the power of cognitive neural networks topology with the flexibility of self reconfiguration and the capability of re-planning optimal trajectories as aggregate swarm-mobile robots can dynamically change their structure to match environmental variations.

3- A cognitive controller for soccer mobile robots; this controller will be implemented on several intelligent robotics that will be based on hybrid neural networks layer architecture with two levels: the execution level, in which the information is handled numerically for decision making; and the knowledge level, whereby the environment is described by a knowledge base containing static facts (background knowledge) provided by the designer in addition to dynamic facts (symbolic information) corresponding to the data acquired through the robot's sensor during its mission.

4- Developing a mobile robot interface between the human and intelligent environment based on robot localisation and navigation by using a cognitive controller for mobile robots with environment representations based on swarm technique.

5- Applying cognitive algorithm and neural networks topology in the design of an adaptive nonlinear controller for mobile robot to carry out the missions in different placements, such as in hospitals as a nurse or a servant in hotels.

References

[1] R. Murphy, Introduction to Artificial Intelligent Robotics. Cambridge, MA: MIT Press, 2000.

[2] R. Siegwart and I. R. Nourbakhah, Introduction to Autonomous Mobile Robots. Cambridge, MA: MIT Press, 2004.

[3] A. Filipescu, V. Minzu, B. Dumitrascu and E. Minca, Trajectory Tracking and Discrete-Time Sliding-Mode Control of Wheeled Mobile Robot. Proceedings of the IEEE International Conference on Information and Automation, Shenzhen, China, 2011, pp. 27-32.

[4] B. Krose, Environment Representations for Cognitive Robot Companions. ERCM News, No. 53, 2003, pp. 29-30.

[5] M. Hulse and M. Hild, Informatics for Cognitive Robots. Advanced Engineering Information, Vol. 24, 2010, pp.2-3.

[6] A. Farinelli, G. Grisetti and L. Locchi, Design and Implementation of Modular Software for Programming Mobile Robot. International Journal Advance Robotics Systems, Vol. 3, 2006, pp. 37-42.

[7] A.A. Razavian and J. Sun, Cognitive Based Adaptive Path Planning Algorithm for Autonomous Robotic Vehicles. Proceedings of the IEEE Southeast Con, 2005, pp. 153-160.

[8] C. Beak and H. Neumann, A Neuroinspired Cognitive Behavioural Control Architecture for Visually Driven Mobile Robotics. Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, Bangkok, Thailand, Feb. 21-26, 2009, pp. 2190-2196.

[9] E. Ahle and D. Soffker, A Cognitive-Oriented Architecture to Realize Autonomous Behaviour-Part II: Application to Mobile Robotics. Proceedings of the 2006 IEEE International Conference on Systems, Man and Cybernetics, Taipei, Taiwan, Oct 8-11, 2006, pp. 2221-2227.

[10] J. Velagic, B. Lacevic and B. Perunicic, A 3-Level Autonomous Mobile Robot Navigation System Designed by Using Reasoning/Search Approaches. Robotics Autonomous Systems. Vol. 54, No. 12, 2006, pp. 989-1004.

[11] C. Ordonez, E.G. Collins Jr., M.F. Selekwa and D.D. Dunlap, The Virtual Wall Approach to Limit Cycle Avoidance for Unmanned Ground Vehicles. Robotics Autonomous Systems. Vol. 56, No. 8, 2007, pp. 645-657.

[12] M.A. Porta, O. Montiel, O. Castillo, R. Sepulveda and P. Melin, Path Planning for Autonomous Mobile Robot Navigation with Ant Colony Optimization and Fuzzy Cost Function Evaluation. Applied Soft Computing, Vol. 9, 2009, pp. 1102-1110.

[13] V. Mohan, P. Morasso, G. Metta and S. Kasderidis, The Distribution of Rewards in Sensorimotor Maps Acquired by Cognitive Robots through Exploration. Neurocomputing, Vol. 74, 2011, pp. 3440-3455.

[14] O. Motlagh, S. H. Tang, N. Ismail and A. R. Ramli, an Exoert Fuzzy Cognitive Map for Reactive Navigation of Mobile Robot. Fuzzy Sets and Systems, Vol. 201, 2012, pp. 105-121.

[15] H. Wei, B. Wang, Y. Wang, Z. Shao and K. C. Chan, Staying-alive Path Planning with Energy Optimization for Mobile Robot. Expert Systems with Applications, Vol. 39, 2012, pp. 3559-3571.

[16] Real-time Navigation of Mobile Robots in Problems of Border Patrolling and Avoiding Collisions with Moving and Deforming Obstacle. Robotics and Autonomous Systems, Vol. 60, 2012, pp. 769-788.

[17] X. Linlin, T. Changxing, Y. Xuedong, K. Yunhan and F. Yan, 2-Dimensional SVFC Application in Path Planning of Mobile Robots. Energy Procedia, Vol. 17, 2012, pp. 1563-1569.

[18] N. Ghita and M. Kloetzer, Trajectory Planning for a Car-like Robot by Environment Abstraction. Robotics and Autonomous Systems, Vol. 60, 2012, pp. 609-619.

[19] T-C. Lee, K-T. Song, C-H. Lee and C-C. Teng, Tracking Control of Unicycle-Modelled Mobile Robots Using a Saturation Feedback Controller. IEEE Transactions on Control Systems Technology, Vol. 9, No. 2, 2001, pp. 305-318.

[20] S. Nazlibilek, Autonomous Navigation of Robotic Units in Mobile Sensor Network, Measurement Vol. 45, 2012, pp. 938–949.

[21] J. M. Toibero, F. Roberti, R. Carlli and P. Fiorini, Switching Control Approach for Stable Navigation of Mobile Robots in Unknown Environment. Robotics and Computer-Integrated Manufacturing, Vol. 27, 2011, pp. 558-568.

[22] W. Yaonan, Y. Yimin, Y. Xiaofang, Z. Yuanli, Y. Feng and T. Lei, Autonomous Mobile Robot Navigation System Designed in Dynamic Environment Based on Transferable Belief Model. Measurement Vol. 44, 2011, pp. 1389-1405.

[23] C-Y. Tsai and K-T. Song, Visual Tracking Control of a Wheeled Mobile Robot with System Model and Velocity Quantization Robustness. IEEE Transactions on Control Systems Technology, Vol. 17, No. 3, 2009, pp. 520-527.

[24] T. Hirukawa, S. Komada and J. Hirai, Image Feature Based Navigation of Nonholonomic Mobile Robots with Active Camera. SICE Annual Conference, Kagawa University-Japan, 12-20 Sept. 2007, pp. 2502-2506.

[25] Y. Liu, J. Gao, D. Liu and Z. Wang, The Design of Control System and Study on Control Algorithm of Laser Navigation Mobile Robot. Proceedings of the 3rd International Congress on Image and Signal Processing CISP, 2010, pp. 4276-4280.

[26] Y-C. Chang, Y.Y. Lwin and Y. Yamamoto, Sensor-Based Trajectory Planning Strategy for Nonholonomic Mobile Robot with Laser Range Sensors. Proceedings of the IEEE International Symposium on Industrial Electronics ISIE 2009, Seoul Olympic Parktel, Seoul, South Korea, 5-8 July, 2009, pp. 1755-1760.

[27] A. Kokosy, F-O. Defaux and W. Perruquetti, Autonomous Navigation of a Nonholonomic Mobile Robot in a Complex Environment. Proceedings of the 2008 IEEE International Workshop on Safety, Security and Rescue Robotics, Sendai, Japan, October 2008, pp. 102-108.

[28] D. Pedrosa, A. Medeiros and P. Alsina, Point-to-Point Paths Generation for Wheeled Mobile Robots. Proceedings of the IEEE International Conference on Robotics and Automation, Taipei, Taiwan, 14-9 Sept. 2003, pp. 3752-3757.

[29] G. Yasuda and H. Takai, Sensor-Based Path Planning and Intelligent Steering Control of Nonholonomic Mobile Robots. Proceedings of the 27th Annual Conference of the IEEE Industrial Electronics Society IECON, 2001, pp. 317-322.

[30] T. Sahin and E. Zergeroglu, A Computationally Efficient Path Planner for a Collection of Wheeled Mobile Robots with Limited Sensing Zones. Proceedings of the IEEE International Conference on Robotics and Automation, Roma, Italy, 10-14 April 2007, pp. 1074-1079.

[31] W. Shangming and Z. Milo, Smooth Path Planning and Control for Mobile Robots. Proceedings of the IEEE International Conference on Network Sense Control, March 2005, pp. 894-899

[32] T. Hague, N. Tillet, Navigation and Control of an Autonomous Horticultural Robot. Mechatronics, Vol. 6, No. 2, 1996, pp. 165-180.

[33] H. Choi and S. Ryew, Robotic System with Active Steering Capability for Internal Inspection of Urban Gas Pipelines. Mechatronics, Vol. 12, No. 5, 2002, pp. 713-736.

[34] F. Takanori, N. Kiroshi and A. Norihiko, Adaptive Tracking Control of a Nonholonomic Mobile Robot. IEEE Trans Robotic Autonomous, Vol 16, No. 2, 2000, pp. 609-615.

[35] J. E. Slotine and W. Li, Applied Nonlinear Control. Upper Saddle River, NJ: Prentice Hall, 1991.

[36] R-J Wai and C-M Liu, Design of Dynamic Petri Recurrent Fuzzy Neural Network and Its Application to Path-Tracking Control of Nonholonomic Mobile Robot. IEEE Transactions on Industrial Electronics, Vol. 56, No.7, 2009, pp. 2667-2683.

[37] F. Mnif and F. Touati, An Adaptive Control Scheme for Nonholonomic Mobile Robot with Parametric Uncertainty. International Journal of Advanced Robotic Systems, Vol. 2, No. 1, 2005, pp 59-63.

[38] Z. Hendzel, An Adaptive Critic Neural Networks for Motion Control of Wheeled Mobile Robot. Nonlinear Dynamics, Vol. 50, No. 4, 2007, pp. 849-855.

[39] K. S. Narenda and K. Patgasarathy, Identification and Control of Dynamical Systems using Neural Networks. IEEE Transactions on Neural Networks Vol. 1, No. 1, 1990, pp. 4-27.

[40] D. Nguye and B. Widrow, Neural Networks for Self-Learning Control Systems, IEEE Control System Magazine, Vol. 10, No. 1, 1990, pp. 18-23.

[41] K. Horlink, M. Stinchombe and H. White, Universal Approximation of an Unknown Mapping and Its Derivatives Using Multilayer Feedforward Networks. Neural Networks, Vol. 3, 1990, pp. 551-560.

[42] M. Corradini, G. Ippoliti and S. Longhi, Neural Networks Based Control of Mobile Robots: Development and Experimental Validation. Journal of Robotic Systems, Vol. 20, No. 10, 2003, pp. 587-600.

[43] Z. P. Jiang, E. Lefeber and H. Nijmeijer, Saturated Stabilization and Tracking of a Nonholonomic Mobile Robot. Systems and Control Letters, Vol. 42, 2001, pp. 327-332.

[44] G. Ramiez and S. Zeghloul, A New Local Path Planner for Nonholonomic Mobile Robot Navigation in Cluttered Environments. Proceedings of the IEEE Conference on Robotics and Automation, 2000, pp. 2058-2063.

[45] M.K. Bugeja, S.G. Fabri and L. Camilleri, Dual Adaptive Dynamic Control of Mobile Robots Using Neural Networks. IEEE Transactions on Systems, Man, and Cybernetics-Part B: CYBERNETICS, Vol. 39,No. 1, 2009, pp.129-141.

[46] M. Imen, M. Mansouri and M. A. Shoorehdeli, Tracking Control of Mobile Robot Using ANFIS. Proceedings of the IEEE International Conference on Mechatronics and Automation, Beijing, Chain, 7-10 August, 2011, pp. 422-427.

[47] N.A. Martins, D.W. Bertol, E.R. De Pieri, E.B. Castelan and M.M. Dias, Neural Dynamic Control of a Nonholonomic Mobile Robot Incorporating the Actuator Dynamics. CIMCA 2008, IAWTIC 2008, and ISA 2008, IEEE Computer Society 2008, pp. 563-568.

[48] A. Filipescu, V. Minzu, B. Dumitrascu and E. Minca, Discrete-Time Sliding Mode Control of Wheeled Mobile Robots. Proceedings of the 8th Asian Control Conference ASCC, Kaohsiung, Taiwan, May 15-18, 2011, pp. 771-776.

[49] Y. Liu, Y. Zhand and H. Wang, Tracking Control of Wheeled Mobile Robots Based on Sliding-Mode Control. Proceedings of the 2nd International Conference on Artificial Intelligence, Management Science and Electronic Commerce AIMSEC, 2011, pp. 1787-1790.

[50] J. Ye, Tracking Control for Nonholonomic Mobile Robots: Integrating the Analog Neural Network into the Backstepping Technique. Neurocomputing, Vol. 71, 2008, pp. 3373-3378.

[51] H. Guangxin and Z. Yanhui, Global Trajectory Tracking Control of Nonholonomic WMRs with Driving Motor Dynamics Being Considered. Proceedings of the 2nd International Conference on Intelligent Control and Information Processing ICICIP, 2011, pp. 640-643.

[52] L. Cheng, L. Cao, H. Wu, Q. Zhu, W. Xu and L. Liu, Trajectory Tracking Control of Nonholonomic Mobile Robots by Back-Stepping. Proceedings of the International Conference on Modelling, Identification and Control ICMIC, Shanghai, China, 26-29 June, 2011, pp. 134-139.

[53] G. Klancar and I. Skrjanc, Tracking Error Model-Based Predictive Control for Mobile Robots in Real Time. Robotics and Autonomous System, Vol. 55, 2007, pp. 460-469.

[54] I. Maurovic, M. Baotic and I. Petrovic, Explicit Model Predictive Control for Trajectory Tracking with Mobile Robots. Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics AIM 2011, Budapest, Hungary, 3-7 July, 2011, pp. 712-717

[55] F.N. Martins, W.C. Celesta, R. Carelli, M. Sarcinelli–Filho, and T.F. Bastos–Filho, An Adaptive Dynamic Controller for Autonomous Mobile Robot Trajectory Tracking. Control Engineering Practice, Vol. 16, 2008, pp. 1354-1363.

[56] T. Das, I.N. Kar and S. Chaudhury, Simple Neuron-Based Adaptive Controller for a Nonholonomic Mobile Robot Including Actuator Dynamics. Neurocomputing, Vol. 69, 2006, pp. 2140-2151.

[57] T. Y. Wang, C. Tsai, Adaptive Trajectory Tracking Control of a Wheeled Mobile Robot via Lyapunov Techniques. Proceedings of the 30th Annual Conference of the IEEE Industrial Electronics Society, Busan, South Korea, November 2-5, 2004. pp. 389-394.

[58] J. Wang, Z. Lu, W. Chen and X. Wu, An Adaptive Trajectory Tracking Control of Wheeled Mobile Robots. Proceedings of the 6th IEEE conference on Industrial Electronics and Applications, 2011, pp. 1156-1160.

[59] J. Ye, Adaptive Control of Nonlinear PID-Based Analogue Neural Network for a Nonholonomic Mobile Robot. Neurocomputing, Vo. 71, 2008, pp. 1561-1565.

[60] J. E. Normey-Rico, I. Alcala, J. Gomez-Ortega and E. F. Camacho, Mobile Robot Path Tracking Using a Robust PID Controller. Control Engineering Practice, Vol. 9, 2001, pp. 1209-1214.

[61] M.S. Saidonr, H. Desa and R. M, Noor, A Differential Steering Control with Proportional Controller for an Autonomous Mobile Robot. Proceedings of the 7th IEEE International Colloquium on Signal Processing and its Applications, 2011, pp.90-94.

[62] R. Tsai-Jiun, C. Tien-Chi and C. Chun-Jung, Motion Control for a Two-Wheeled Vehicle Using a Self-Tuning PID Controller. Control Engineering Practice, Vol. 16, 2008, pp. 365-375.

[63] P.k. Padhy, T. Sasaki, S. Nakamura and H. Hashimoto, Modeling and Position Control of Mobile Robot. Proceedings of the 11th IEEE International Workshop on Advanced Motion Control, Nagaoka-Japan, 21-24 March, 2010, pp.100-105.

[64] H-X Zhang, G-J Dai and H Zeng, A Trajectory Tracking Control Method for Nonholonomic Mobile Robot. Proceedings of the International Conference on Wavelet Analysis and Pattern Recognition Beijing, China, 2-4 Nov. 2007, pp. 7-11.

[65] S. Sun, Designing Approach on Trajectory Tracking Control of Mobile Robot. Robotics and Computer Integrated Manufacturing, Vol. 21, 2005, pp. 81-85.

[66] C-K Hwang, B-S Chen and Y-T Chang, Combination of Kinematical and Robust Dynamical Controllers for Mobile Robotics Tracking Control: I Optimal H_∞ Control. Proceedings of the IEEE International Conference on Control Applications, 2004, pp. 1205-1210.

[67] S. Thrun, An Approach to Learning Mobile Robot Navigation. Robotics and Autonomous Systems, Vol. 15, 1995, pp. 301-319.

[68] Y.Y. Cha, Navigation of Free-Ranging Mobile Robot Using Heuristic Local Path-Planning Algorithm. Proceedings of the Robotics and Computer-Integrated Manufacturing, Vol. 13, No. 2, 1997, pp. 145-156.

[69] J. Reuter, Mobile Robots Trajectories with Continuously Differentiable Curvature: An Optimal Control Approach. Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Victoria, B.C., Canada, Oct 1998, pp. 38-43.

[70] M. Cikes, M. Dakulovic and I. Petrovic, The Path Planning Algorithms for a Mobile Robot Based on the Occupancy Grid Map of the Environment – A Comparative Study. Proceedings of International Symposium on Information, Communication and Automation Technologies, 2011, pp. 1-8.

[71] Y. Zhong, B. Shirinzadah and Y. Tain, A New Neural Network for Robot Path Planning. Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Xian, China, July 2-5, 2008, pp. 1361-1366.

[72] N. Sadati and J. Taheri, Solving Robot Motion Planning Problem Using Hopfield Neural Network in a Fuzzified Environment. Proceedings of the IEEE International Conference on Fuzzy Systems, 2002, pp. 1144-1149.

[73] M. Li and J. Hu and L. Li, Multipath Planning Based on Neural Network Optimized with Adaptive Niche in Unknown Environment. Proceedings of the International Conference on Intelligent Computation Technology and Automation, 2010, pp. 761-764.

[74] H. Chen, X. Du and W. Gu, Path Planning Method Based on Neural Network and Genetic Algorithm. Proceedings of the International Conference on Intelligent Mechatronics and Automation, Chendgu, China, 2004, pp. 667-671.

[75] M. Gao, J. Xu, J. Tian and H. Wu, Path Planning for Mobile Robot Based on Chaos Genetic Algorithm. Proceedings of the 4th International Conference on Natural Computation, 2008, pp. 409-413.

[76] S. Yang and C. Luo, A Neural Network Approach to Complete Coverage Path Planning. Proceedings of the IEEE Transactions on Systems, Man and Cybernetics, Part B, Vol. 34, No. 1, 2004, pp. 718-724.

[77] Y-H. Xue and H-P. Liu, Optimal Path Planning for Service Robot in Indoor Environment. Proceedings of the International Conference on Intelligent Computation Technology and Automation, 2010, pp. 850-853.

[78] E. Masehian and D. Sedighizadeh, A Multi-Objective PSO-based Algorithm for Robot Path Planning. Proceedings of the IEEE International Conference on Industrial Technology, 2010, pp. 465-470.

[79] W. Li and G-Y. Wang, Application of Improved PSO in Mobile Robotic Path Planning. Proceedings of the International Conference on Intelligent Computing and Integrated Systems, 2010, pp. 45-48.

[80] F. Wang and B. Lu, The Robot Path Planning Based on PSO Algorithm Integrated with PID. Proceedings of the International Conference on Information Engineering and Computer Science, 2009, pp. 1-4.

[81] J. A. Batlle, J. M. Font-Llagunes and A. Barjau, Calibration for Mobile Robots with an Invariant Jacobian. Robotics and Autonomous Systems, Vol. 58, 2010, pp. 10-15.

[82] K. Kanj and A. Zell, Path Following for an Omnidirectional Mobile Robot Based on Model Predictive Control. Proceedings of the IEEE International Conference on Robotics and Automation, Kobe international Conference Centre, May 12-17, 2009, pp. 3341-3346.

[83] Y. Liu, X. Wu, J. Jim Zhu, and J. Lew, Omni-Directional Mobile Robot Controller Design by Trajectory Linearization. Proceedings of the American Control Conference, Denver, Colorado, Jun. 2003, pp. 3423-3428.

[84] G. V. Raffo, G. K. Gomes and J. E. Normey-Rico, A Predictive Controller for Autonomous Vehicle Path Tracking. IEEE Transactions on Intelligent Transportation Systems, Vol. 10, No.1, 2009, 92-102.

[85] D. Gu and H. Hu, Neural Predictive Control for a Car-Like Mobile Robot. Robotics and Autonomous Systems, Vol. 39 2002, pp. 73-86.

[86] K. Shojaei, A. M. Shahri and A. Tarakameh, Adaptive Feedback Linearizing Control of Nonholonomic Wheeled Mobile Robots in Presence of Parametric and Nonparametric Uncertainties. Robotics and Computer-Integrated Manufacturing Vol. 27, 2011, pp. 194-204.

[87] R. Martinez, O. Castillo and L. Aguilar, Optimization of Interval Type-2 Fuzzy Logic Controllers For a Perturbed Autonomous Wheeled Mobile Robot Using Genetic Algorithms. Information Sciences, Vol. 179, 2009, pp. 2158-2174

[88] J.J. Craig, Introduction to Robotics: Mechanical and Control. 2nd edition. Boston. Addison-Wesley, 1989.

[89] L. Sciavicco, L. Siciliano, Modeling and Control of Robot Manipulators. New York. McGraw-Hill, 1996.

[90] W.E. Dixon, D.M. Dawson, E. Zergeroglu and A. Behal, Nonlinear Control of Wheeled Mobile Robots. London: Springer-Verlag, 2001.

[91] Y. Wang, Y. Chen and M. Lin, Dynamic Object Tracking Control for Nonholonomic Wheeled Autonomous Robot. Journal of Science and Engineering, Vol. 12, No. 3, 2009, pp. 339-350.

[92] S. Han, B. Choi and J. Lee, A Precise Curved Motion Planning for a Differential Driving Mobile Robot. Mechatronics, Vol. 18, 2008, pp. 486-494.

[93] G. Scaglia, A. Rosales, L. Quintero, V. Mut and R. Agarwal, A Linear-Interpolation-Based Controller Design for Trajectory Tracking of Mobile Robots. Control Engineering Practice, Vol. 18, 2010, pp. 318-329.

[94] G. Klancar, D. Matko and S. Blazic, A Control Strategy for Platoons of Differential Drive Wheeled Mobile Robot. Robotics and Autonomous Systems, Vol. 59, 2011, pp. 57-64.

[95] S.X. Yang, H. Yang, Q. Max and H. Meng, Neural Dynamics Based Full-State Tracking Control of a Mobile Robot. Proceedings of the IEEE International Conference on Robotics & Automation, New Orleans, LA, 2004, pp. 4614-4619.

[96] A.M. Bloch, Nonholonomic Mechanics and Control. New York: Springer-Verlag, 2003.

[97] K. Ogata, Modern Control Engineering. 4th Edition, by Addison- Wesley Publishing Company, Inc. 2003.

[98] B.S. Park, S.J. Yoo, J.B. Park and Y.H. Choi, Adaptive Neural Sliding Mode Control of Nonholonomic Wheeled Mobile Robots with Model Uncertainty. IEEE Transactions on Control Systems Technology, Vol. 17, No. 1, 2009, pp.207-214.

[99] O. Nells, Nonlinear System Identification. London: Springer-Verlag, 2001.

[100] M. Norgaard, O. Ravn, N. K. Poulsen and L. K. Hansen, Neural Networks for Modeling and Control of Dynamic Systems. London: Springer-Verlag, 2000.

[101] N. Dong, D. Liu and Z. Chen, Data Based Predictive Control Using Neural Networks and Stochastic Approximation. Proceedings of the International Conference on Modeling, Identification and Control, China, June 26-29, 2011, pp. 256-260.

[102] K.R. Chernyshov, An Information Theoretic Approach to Neural Network Based System Identification. Proceedings of the Siberian Conference on Control and Communications, 2009, pp.100-107.

[103] Y. Chetouani, Using Artificial Neural Networks for the Modeling of a Distillation Column. International Journal of Computer Science and Applications, Vol. 4, Issue 3, 2007, pp. 119-133.

[104] W. Zhang, On Training Optimization of the Generalized ADLINE Neural Network for Time Varying System Identification. Proceedings of the Chinese Control and Decision Conference, 2009, pp. 775-780.

[105] A. Jain, A.S. Thoke, R. N. Patel and E. Koley, Intercircuit and Cross-Country Fault Detection and Classification Using Artificial Neural Network. Annual IEEE India Conference, 2010, pp. 1-4.

[106] G.W. Ng, Application of Neural Networks to Adaptive Control of Nonlinear System. London: Springer-Verlag, 1997.

[107] O. Omidvar and D. L. Elliott, Neural Systems for Control. London: Academic Press, 1997.

[108] G.W. Irwin, K. Warwick and K. J. Hunt, Neural Network Application in Control. London: The Institution of Electrical Engineers, 1995.

[109] A. Zalzala and A. Morris, Neural Networks for Robotic Control Theory and Applications. London: Ellis Horwood, 1996.

[110] C.H. Dagli, Artificial Neural Networks for Intelligent Manufacturing. London: Chapman and Hall, 1994.

[111] S. Chen, E.S. Chng and K. Alkadhimi, Regularized Orthogonal Least Squares Algorithm for Constructing Radial Basis Function Networks. International Journal of Control, Vol. 64, No. 5, 1996, pp. 829-309.

[112] P.C. Panchariya, A.K. Palit, D. Popovic and A.L. Sharrna, Nonlinear System Identification Using Takagi-Sugeno Type Neuro-Fuzzy Model. Proceedings of the 2nd International IEEE Conference on Intelligent Systems, 2004, pp. 76-81.

[113] L.R. Medsker and L.C. Jain, Recurrent Neural Networks Design and Applications. CRC Press LLC 2001.

[114] D. T. Pham and X. Lin, Neural Network for Identification, Prediction, and Control. London: Springer-Verlag, 1995.

[115] P.P. Kanjilal, Adaptive Prediction and Predictive Control. London: Peter Peregrinus, 1995.

[116] S. Omatu, M. Khalid, and R. Yusof, Neuro- Control and Its Applications. London: Springer-Velag, 1995.

[117] J. Xu, M. Zhang and J. Zhang, Kinematic Model Identification of Autonomous Mobile Robot Using Dynamical Recurrent Neural Networks. Proceedings of the IEEE International Conference on Mechatronics & Automation, Niagara Falls, Canada, July 2005, pp. 1447-1450.

[118] J.M. Zurada, Introduction to Artificial Neural Systems. USA: Jaico Publishing House, 1992.

[119] K.S. Narendra and K. parthasarathy, Gradient Methods for the Optimization of Dynamical Systems Containing Neural Networks. IEEE Trans. Neural Networks, Vol. 2, No. 2, 1991, pp. 252-262.

[120] A.U. Levin and K.S. Narendra, Control of Nonlinear Dynamical Systems Using Neural Networks-Part II: Obsrvability, Identification, and Control. IEEE Trans. Neural Networks, Vol. 7, No. 1, 1996, pp. 30-42.

[121] M. Amina, V.S. Kodogiannis, I.P. Petrounias, J.N. Lygouras and G-J. E. Nychas, Identification of the Listeria Monocytogenes Survival Curves in UHT Whole Milk Utilising Local Linear Wavelet Neural Networks. Expert Systems with Applications, Vol. 39, 2012, pp. 1435-1450.

[122] I.E. Putro, A. Budiyono, K.J. Yoon and D.H. Kim, Modeling of Unmanned Small Scale Rotorcraft Based on Neural Network Identification. IEEE International Conference on Robotics and Biomimetics, Bangkok, Thailand, Feb 21-26, 2009, pp. 1938-1943.

[123] E.P. Nahas, M.A. Henson and D.E. Sebrog, Nonlinear Internal Model Control Strategy for Neural Network Models. Computer Chem. Eng. Vol. 16, No. 12, 1992, pp. 1039-1057.

[124] E. F. Camacho and C. Bordons, Model Predictive Control. London: Springer-Verlag, 1999.

[125] Q. Zhong, Robust Control of Time-delay Systems. London: Springer-Verlag, 2006.

[126] S. Sastry, Nonlinear Systems Analysis, Stability and Control. New York: Springe-Verlag, 1999.

[127] M. Vidyasagar, Nonlinear Systems Analysis. 2nd Edition Upper Saddle River, NJ: Prentice Hall, 1993.

[128] A. Luca, G. Oriolo, Modelling and Control on Nonholonomic Mechanical System. Vienna: Springer-Verlag, 1995.

[129] Z. Yongjie, C. Jiang and W. Shuguo, A New Path-Planning Algorithm for Mobile Robot Based on Neural Network. Proceedings of the IEEE International Conference on Computers, communications, Control Andrews Powerhouse Engineering, 2002, pp. 1570-1573.

[130] R. Al-Jarrah, On the Lagrange Interpolation Polynomials of Entire Functions. Journal of Approximation Theory, Vol. 41, 1984, pp. 170-l 78.

[131] S.B. Damelina and H.S. Jung, Pointwise convergence of derivatives of Lagrange interpolation polynomials for exponential weights. Journal of Computational and Applied Mathematics, Vol. 173, 2005, pp. 303–319.

[132] F. Haslinger, On Newton's Interpolation Polynomials. Journal of Approximation Theory, Vol. 22, 1978, pp. 352-355.

[133] M. Zhang and W. Xiao, Construct and Realization of Newton Interpolation Polynomial Based on Matlab7. Procedia Engineering, Vol. 15, 2011, pp. 3831-3835.

[134] C. Boor, Multivariate polynomial interpolation: Aitken–Neville sets and generalized principal lattices. Journal of Approximation Theory, Vol. 161, 2009, pp. 411–420.

[135] P. Henrici, Essential of Numerical Analysis. New York: John Wiley and Sons, 1982.

[136] T. Horsch and B. Jiittler, Cartesian Spline Interpolation for Industrial Robots, Computer-Aided Design**,** Vol. 30, No. 3, 1998, pp. 217-224.

[137] M Egerstedt and C. F.Martin, Optimal Trajectory Planning and Smoothing Splines, Automatica, Vol. 37, 2001, pp.1057-1064.

[138] T. Maekawa, T. Noda, S. Tamura, T. Ozaki and K-I. Machida, Curvature Continuous Path Generation for Autonomous Vehicle using B-Spline Curves, Computer-Aided Design, Vol. 42, 2010, pp. 350_359.

[139] J. Derrac, S. Garc, D. Molina and F. Herrera , A Practical Tutorial on the Use of Nonparametric Statistical Tests as a Methodology for Comparing Evolutionary and Swarm Intelligence Algorithms. Journal of Swarm and Evolutionary Computation, Vol. 1, 2011, pp. 3–18.

[140] J. Zhou, Z. Duan, Y. Li, J. Deng and D. Yu, PSO-Based Neural Network Optimization and Its Utilization in a Boring Machine. Journal of Materials Processing Technology, Vol. 178, 2006, pp. 19–23.

[141] Y.S. Lee, S.M. Shamsuddin, and H.N. Hamed, Bounded PSO Vmax Function in Neural Network Learning. Proceedings of the 8th International Conference on Intelligent Systems Design and Applications, 2008, pp. 474-479.

[142] Internet website http://www.parallax.com/, Robotics with the Boe-Bot Text Manual v3.0. Accessed Oct 2011.

[143] K-H. Su, Y-Y. Chen and S-F. Su, Design of Neural-Fuzzy-Based Controller for Two Autonomously Driven Wheeled Robot. Neurocomputing, Vol. 73, 2010, pp. 2478-2488.

[144] Internet website http://www.iogear.com/product/GUWH104KIT, Wireless USB Hub and Adapter Text Manual. Accessed Oct 2011.

Appendix A

Holonomic and Nonholonomic Wheeled Mobile Robot

The term holonomic has broad applicability to several mathematical areas, including differential equations, functions and constraint expressions [2]. In mobile robotics, the term refers to the kinematic constraints of the robot chassis.

A nonholonomic mobile robot has the following properties:

1- The robot configuration is described by three coordinates. Three values are needed to describe the location and orientation of the robot.

2- The robot has two DOF, or three DOF with singularities.

A holonomic robot is a robot that has zero nonholonomic kinematics constraints. Conversely, a nonholonomic robot is a robot with one or more nonholonomic kinematics constraints [2].

A holonomic kinematics constraint can be expressed as an explicit function of position variables (x, y, θ) only, not using derivatives of these values, such as ($\dot{x}$, $\dot{y}$, and $\dot{\theta}$).

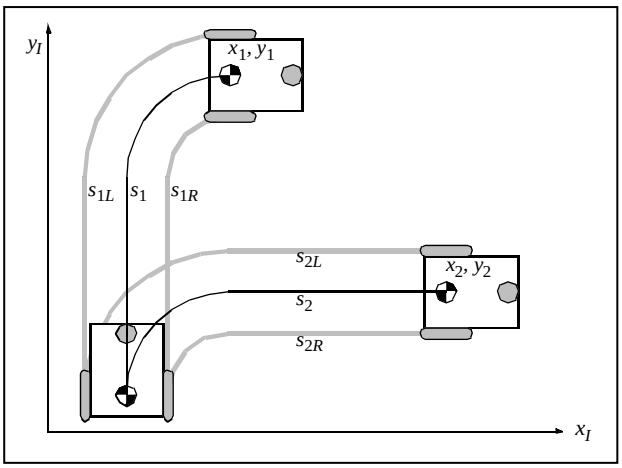


Figure A-1.

A nonholonomic kinematics 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.

Because of this latter point of view, nonholonomic systems are often called non-intergrable systems, so the differential equations are not integrable to the final position, and the measure of the traveled distance of each wheel is not sufficient to calculate the final position of the robot as shown in Figure A-1, where $S_1=S_2$, $S_{1R}=S_{2R}$, $S_{1L}=S_{2L}$, but $x_1 \neq x_2$ and $y_1 \neq y_2$.

To check that the mobile robot is nonholonomic, a mobile robot as shown in Figure A-2, is running along a trajectory $s(t)$, and every instant of the movement its velocity $v(t)$ is:

$$v(t) = \frac{\partial s}{\partial t} = \frac{\partial x}{\partial t}\cos\theta + \frac{\partial y}{\partial t}\sin\theta \tag{A-1}$$

$$ds = dx\cos\theta + dy\sin\theta \tag{A-2}$$

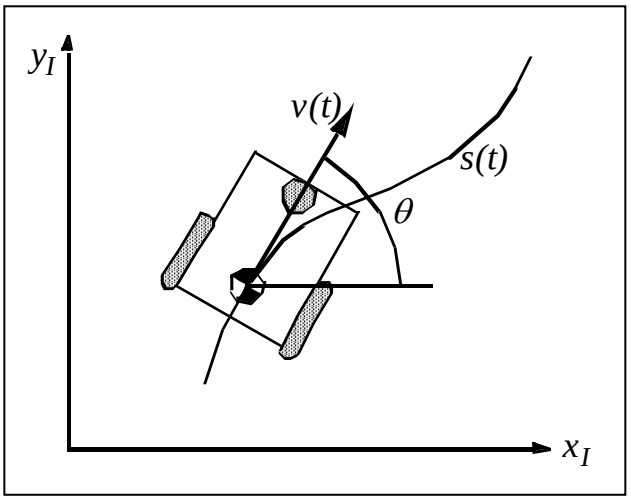


Figure A-2.

The function $v(t)$ is said to be integrable (holonomic) if a trajectory function $s(t)$ exists, which can be described by the values $x, y,$ and θ only.

The condition for integrable function is [100]:

$$\frac{\partial^2 s}{\partial x \partial y} = \frac{\partial^2 s}{\partial y \partial x}; \frac{\partial^2 s}{\partial x \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial x}; \frac{\partial^2 s}{\partial y \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial y} \tag{A-3}$$

with $s = s(x, y, \theta)$,

Then ds becomes as follows:

$$ds = \frac{\partial s}{\partial x}dx + \frac{\partial s}{\partial y}dy + \frac{\partial s}{\partial \theta}d\theta \tag{A-4}$$

$$\frac{\partial s}{\partial x} = \cos\theta, \frac{\partial s}{\partial y} = \sin\theta, \frac{\partial s}{\partial \theta} = 0 \tag{A-5}$$

After applying the condition for an integrable function as in equation A-3:

$$\frac{\partial^2 s}{\partial x \partial y} = \frac{\partial^2 s}{\partial y \partial x} = 0 \tag{A-6}$$

$$\frac{\partial^2 s}{\partial x \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial x} = -\sin\theta = 0 \tag{A-7}$$

$$\frac{\partial^2 s}{\partial y \partial \theta} = \frac{\partial^2 s}{\partial \theta \partial y} = \cos\theta = 0 \tag{A-8}$$

Therefore, the differential equation is not integrable and the mobile robot is a nonholonomic system.

A holonomic mobile robot has the following properties:

1- The robot configuration is described by three coordinates.

2- The robot has three DOF without singularities.

3- The robot can instantly develop a wrench in an arbitrary combination of directions (x, y and θ).

Holonomic mobile robot offers full mobility with the same number of degrees of freedom as the environment. This makes the trajectory tracking easier, because there are no constraints that need to be integrated. Implementing reactive behaviours is easy because there are no constraints to limit the directions in which the robot can accelerate.

Appendix B

Jacobi-Lie-Bracket

By using Jacobi-Lie-Bracket of f and g to find $[f,g]$:

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix}=\begin{bmatrix}0.5\cos\theta(t) & 0.5\cos\theta(t)\\ 0.5\sin\theta(t) & 0.5\sin\theta(t)\\ 1/L & -1/L\end{bmatrix}\begin{bmatrix}V_L\\ V_R\end{bmatrix} \tag{B-1}$$

$$\begin{bmatrix}\dot{x}(t)\\ \dot{y}(t)\\ \dot{\theta}(t)\end{bmatrix}=\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ 1/L\end{bmatrix}V_L(t)+\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ -1/L\end{bmatrix}V_R(t) \tag{B-2}$$

Let

$$f=\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ 1/L\end{bmatrix}\text{ and }\ g=\begin{bmatrix}0.5\cos\theta(t)\\ 0.5\sin\theta(t)\\ -1/L\end{bmatrix} \tag{B-3}$$

$$[f,g]^i=\sum_{j=1}^{n}\left(f^j\frac{\partial g^i}{\partial q^j}-g^j\frac{\partial f^i}{\partial q^j}\right) \tag{B-4}$$

$$\begin{bmatrix}|f,g|^1\\ |f,g|^2\\ |f,g|^3\end{bmatrix}=\begin{bmatrix}f_1\frac{\partial g_1}{\partial x}-g_1\frac{\partial f_1}{\partial x}+f_2\frac{\partial g_1}{\partial y}-g_2\frac{\partial f_1}{\partial y}+f_3\frac{\partial g_1}{\partial\theta}-g_3\frac{\partial f_1}{\partial\theta}\\ f_1\frac{\partial g_2}{\partial x}-g_1\frac{\partial f_2}{\partial x}+f_2\frac{\partial g_2}{\partial y}-g_2\frac{\partial f_2}{\partial y}+f_3\frac{\partial g_2}{\partial\theta}-g_3\frac{\partial f_2}{\partial\theta}\\ f_1\frac{\partial g_3}{\partial x}-g_1\frac{\partial f_3}{\partial x}+f_2\frac{\partial g_3}{\partial y}-g_2\frac{\partial f_3}{\partial y}+f_3\frac{\partial g_3}{\partial\theta}-g_3\frac{\partial f_3}{\partial\theta}\end{bmatrix} \tag{B-5}$$

$$\begin{bmatrix}|f,g|^1\\ |f,g|^2\\ |f,g|^3\end{bmatrix}=\begin{bmatrix}\frac{-1}{L}\sin\theta(t)\\ \frac{1}{L}\cos\theta(t)\\ 0\end{bmatrix} \tag{B-6}$$

$$[f,g]=\begin{bmatrix}[f,g]^1\\ [f,g]^2\\ [f,g]^3\end{bmatrix}=\begin{bmatrix}\frac{-1}{L}\sin\theta(t)\\ \frac{1}{L}\cos\theta(t)\\ 0\end{bmatrix} \tag{B-7}$$

$$rank\{f,g,[f,g]\}=rank\begin{bmatrix}0.5\cos\theta(t) & 0.5\cos\theta(t) & \frac{-1}{L}\sin\theta(t)\\ 0.5\sin\theta(t) & 0.5\sin\theta(t) & \frac{1}{L}\cos\theta(t)\\ 1/L & -1/L & 0\end{bmatrix} \tag{B-8}$$

$$rank\{f,g,[f,g]\}=[\frac{0.5\cos^2\theta(t)}{L^2}+\frac{0.5\sin^2\theta(t)}{L^2}]-[\frac{-0.5\cos^2\theta(t)}{L^2}+\frac{-0.5\sin^2\theta(t)}{L^2}] \tag{B-9}$$

$$rank\{f,g,[f,g]\}=\frac{1}{L^2} \tag{B-10}$$

Appendix C

Time Derivative of the State Error Vector

$$q = (x, y, \theta)^T \tag{C-1}$$

$$\dot{q} = (\dot{x}, \dot{y}, \dot{\theta})^T \tag{C-2}$$

$$V = \begin{bmatrix} V_I & V_w \end{bmatrix}^T \tag{C-3}$$

$$q_r = [x_r, y_r, \theta_r]^T \tag{C-4}$$

$$u_r = [V_{Ir}, V_{Wr}]^T \tag{C-5}$$

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} \cos\theta(t) & 0 \\ \sin\theta(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_I(t) \\ V_w(t) \end{bmatrix} \tag{C-6}$$

$$\begin{bmatrix} \dot{x}_r(t) \\ \dot{y}_r(t) \\ \dot{\theta}_r(t) \end{bmatrix} = \begin{bmatrix} \cos\theta_r(t) & 0 \\ \sin\theta_r(t) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} V_{Ir}(t) \\ V_{wr}(t) \end{bmatrix} \tag{C-7}$$

$$q_e = [x_e, y_e, \theta_e]^T \tag{C-8}$$

$$q_e = R_o(q_r - q) \tag{C-9}$$

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_r - x \\ y_r - y \\ \theta_r - \theta \end{bmatrix} \tag{C-10}$$

1- The time derivative of the $x_e = x_r\cos(\theta) - x\cos(\theta) + y_r\sin(\theta) - y\sin(\theta)$ (C-11)

$$\dot{x}_e = \dot{x}_r\cos(\theta) - x_r\sin(\theta)\dot{\theta} - \dot{x}\cos(\theta) + x\sin(\theta)\dot{\theta} + \dot{y}_r\sin(\theta) + y_r\cos(\theta)\dot{\theta} - \dot{y}\sin(\theta) - y\cos(\theta)\dot{\theta} \tag{C-12}$$

$$\dot{x}_e = V_{Ir}\cos(\theta_r)\cos(\theta) - x_r\sin(\theta)\dot{\theta} - V_I\cos(\theta)\cos(\theta) + x\sin(\theta)\dot{\theta} + V_{Ir}\sin(\theta_r)\sin(\theta) + y_r\cos(\theta)\dot{\theta} - V_I\sin(\theta)\sin(\theta) - y\cos(\theta)\dot{\theta} \tag{C-13}$$

$$\dot{x}_e = V_{Ir}\cos(\theta_r)\cos(\theta) + V_{Ir}\sin(\theta_r)\sin(\theta) - x_r\sin(\theta)\dot{\theta} + x\sin(\theta)\dot{\theta} + y_r\cos(\theta)\dot{\theta} - y\cos(\theta)\dot{\theta} - V_I(\cos(\theta)\cos(\theta) + \sin(\theta)\sin(\theta)) \quad \text{(C-14)}$$

$$\dot{x}_e = V_{Ir}\cos\theta_e + yeV_w - V_I \quad \text{(C-15)}$$

2- The time derivative of the $y_e = -x_r\sin(\theta) + x\sin(\theta) + y_r\cos(\theta) - y\cos(\theta)$ (C-16)

$$\dot{y}_e = -\dot{x}_r\sin(\theta) - x_r\cos(\theta)\dot{\theta} + \dot{x}\sin(\theta) + x\cos(\theta)\dot{\theta} + \dot{y}_r\cos(\theta) - y_r\sin(\theta)\dot{\theta} - \dot{y}\cos(\theta) + y\sin(\theta)\dot{\theta} \quad \text{(C-17)}$$

$$\dot{y}_e = -V_{Ir}\cos(\theta_r)\sin(\theta) - x_r\cos(\theta)\dot{\theta} + V_I\cos(\theta)\sin(\theta) + x\cos(\theta)\dot{\theta} + V_{Ir}\sin(\theta_r)\cos(\theta) - y_r\sin(\theta)\dot{\theta} - V_I\sin(\theta)\cos(\theta) + y\sin(\theta)\dot{\theta} \quad \text{(C-18)}$$

$$\dot{y}_e = -V_{Ir}\cos(\theta_r)\sin(\theta) + V_{Ir}\sin(\theta_r)\cos(\theta) - x_r\cos(\theta)\dot{\theta} + x\cos(\theta)\dot{\theta} - y_r\sin(\theta)\dot{\theta} + y\sin(\theta)\dot{\theta} + V_I\cos(\theta)\sin(\theta) - V_I\sin(\theta)\cos(\theta) \quad \text{(C-19)}$$

$$\dot{y}_e = -V_{Ir}\cos(\theta_r)\sin(\theta) + V_{Ir}\sin(\theta_r)\cos(\theta) - x_r\cos(\theta)\dot{\theta} + x\cos(\theta)\dot{\theta} - y_r\sin(\theta)\dot{\theta} + y\sin(\theta)\dot{\theta} + V_I\cos(\theta)\sin(\theta) - V_I\sin(\theta)\cos(\theta) \quad \text{(C-20)}$$

$$\dot{y}_e = V_{Ir}\sin\theta_e - x_eV_w \quad \text{(C-21)}$$

3- The time derivative of the $\theta_e = \theta_r - \theta$ (C-22)

$$\dot{\theta}_e = \dot{\theta}_r - \dot{\theta} = V_{wr} - V_w \quad \text{(C-23)}$$

Appendix D

Back Propagation for Modified Elman Recurrent Neural Network

Given are np training pairs {($\overline{G^r}$), ($q^r(k+1)$)} (input), (output).

where r=1, 2, 3…np

$\overline{G}$ = [$\tau_R(k), \tau_L(k), x(k), y(k), \theta(k)$] ; $q^r(k+1) = [x(k+1), y(k+1), \theta(k+1)]$

Start

Initialize VH_{ji}, VC_{jc} and W_{kj} with random values.

where VH is ($nh \times (ni+1)$), VC is ($nh \times nh$) and W is ($3 \times nh$)

Select η

1- r=0 (r: counter initialization)

E_{cycle} =0 (cycle error initialization)

2- $r \leftarrow r+1$

$\overline{G} \leftarrow \overline{G^r}$

$q(k+1) \leftarrow q(k+1)^r$

$h_c^\circ(k) = \alpha h_c^\circ(k-1) + \beta h_c(k-1)$

For (c=1, 2, …C)

$$h_j = H[\sum_{i=1}^{ni} V_{j,i} \times G_i + \sum_{c=1}^{C} V_{j,c} \times h_c^\circ + bais \times V_{j,ni+1}]$$

For (j=1, 2,…nh)

$$q_{mk}(k+1) = L[\sum_{j=1}^{nh} W_{kj} \times h_j + bias \times W_{nh+1}]$$

For (k=1, 2 & 3)

For (j=1 ,2, … nh)

3- Error value is computed.

$$e_m(k+1) = q(k+1) - q_m(k+1)$$

4- Error signals vector δ_m and δ_h of both layers are computed.

$$\delta_k = e_k(k+1) = ex_m(k+1), ey_m(k+1), e\theta_m(k+1)$$

$$\delta_{hj} = \frac{1}{2}(1-h_j^2)\sum_{k=1}^{3}\delta_k W_{kj}$$

For (j=1,2,…nh)

5- Output layer weights are adjusted:

$$W_{kj} = W_{kj} + \eta \times \delta_k h_j$$

For (k=1, 2, & 3)

For (j=1, 2, 3,…nh)

$$W_{nh+k} = W_{nh+k} + \eta \times \delta_k bais$$

6- Hidden layer weights are adjusted:

$$VH_{j,i} = VH_{j,i} + \eta \times \delta_{hj} \times G_i$$

For (j=1, 2, …nh)

For (i=1, 2, …ni)

$$VH_{j,nh+1} = VH_{j,nh+1} + \eta \times \delta_{hj} \times bais$$

7- Context layer weights are adjusted:

$$VC_{j,c} = VC_{j,c} + \eta \times \delta_{hc} \times h_c^{\circ}$$ For (j=1, 2,…nh)

For (c=1,2,…C)

8- Calculate error cycle:

$$E_{cycle} = E_{cycle} + \frac{1}{2}[ex_m^2(k+1) + ex_m^2(k+1) + e\theta_m^2(k+1)]$$

if r < Z; Then Goto Step 2

if $\frac{E_{cycle}}{Z} > \text{MSE}_{\max}$; Then Goto Step 1

where the $\text{MSE}_{\max}$ is the upper bound of the MSE.

STOP

Appendix E

Weights of the Posture Neural Network Identifier

$$VH_{ji} = \begin{bmatrix} 0.2336 & 1.445 & -0.2432 & 2.12079 & -0.21533 \\ -0.2834 & 0.1739 & -2.224 & 3.32414 & 0.10547 \\ -0.1406 & -0.5004 & -0.5491 & -3.11113 & -1.21674 \\ 2.0338 & -0.121 & 0.7542 & 1.70228 & 0.54603 \\ 0.3098 & 1.4439 & -0.8321 & 2.15473 & -0.21843 \\ -0.3247 & -0.6631 & -0.4912 & 1.3116 & 0.40267 \end{bmatrix}$$

$$VC_{ic} = \begin{bmatrix} 0..2575 & 0.9423 & -0.49179 & -0.14935 & -1.31367 & -0.51667 \\ -0.3234 & -0.37358 & 0.27136 & 0.5211 & -0.14366 & -0.3241 \\ 0.4321 & -0.3231 & 0.6542 & 0.19932 & 1.07638 & 0.16778 \\ 0.967 & -0.00058 & -0.47344 & -0.5592 & 0.9952 & 0.21532 \\ -0.396 & -0.4353 & 0.9732 & 1.25627 & -0.221 & 1.2591 \\ 0.761 & -0.1521 & -1.35358 & 0.42745 & 1.7623 & -1.021 \end{bmatrix}$$

$$\overline{Vb}_j^T = \begin{bmatrix} -1.571 & 0.949 & -0.239 & -2.627 & 0..7321 & -0.3274 \end{bmatrix}^T$$

$$W_{kj} = \begin{bmatrix} 0.4553 & 0.477 & -0.321 & 1.023 & 0.6185 & 0.753 \\ -0.3229 & -1.817 & -0.306 & -1.0618 & -0.04945 & 2.071 \\ -1.1631 & 1.501 & 4.331 & 1.917 & 0.197 & -3.344 \end{bmatrix}$$

$$\overline{Wb}_k^T = \begin{bmatrix} -1.489 & -0.732 & 0.9117 \end{bmatrix}^T$$

Appendix F

Linearisation of the Closed Loop Characteristic Equation

$$\begin{bmatrix} e\dot{x}_m \\ e\dot{y}_m \\ e\dot{\theta}_m \end{bmatrix} = \begin{bmatrix} (w_r + v_r(k_y ey_m + k_\theta \sin e\theta_m))ey_m - k_x ex_m \\ -(w_r + k_y v_r ey_m + k_\theta v_r \sin e\theta_m)ex_m + v_r \sin e\theta_m \\ -v_r(k_y ey_m + k_\theta \sin e\theta_m) \end{bmatrix} \quad \text{(F-1)}$$

By linearization equation F-1 as follows:

$\sin\theta = \theta$.

$$\begin{bmatrix} e\dot{x}_m \\ e\dot{y}_m \\ e\dot{\theta}_m \end{bmatrix} = \begin{bmatrix} -k_x^* & w_r & 0 \\ -w_r & 0 & v_r \\ 0 & -k_y^* v_r & -k_\theta^* v_r \end{bmatrix} \begin{bmatrix} ex_m \\ ey_m \\ e\theta_m \end{bmatrix} \quad \text{(F-2)}$$

The closed loop characteristic of the system can be determined by using equation F-3.

$$\det[SI - A] = 0 \quad \text{(F-3)}$$

$$\det[SI - A] = \begin{bmatrix} S + k_x^* & -w_r & 0 \\ w_r & S & -v_r \\ 0 & k_y^* v_r & S + k_\theta^* v_r \end{bmatrix} \quad \text{(F-4)}$$

$$S^3 + (k_x^* + v_r k_\theta^*)S^2 + (k_x^* v_r k_\theta^* + v_r^2 k_y^* + w_r^2 + v_r k_\theta^*)S + w_r^2 v_r k_\theta^* + k_x v_r^2 k_y^* \quad \text{(F-5)}$$

To convert a continuous-time characteristic equation S to the discrete-time characteristic equation Z, the forward difference method is used, as follows:

$$c/cs(z) = c/cs(s)\big|_{s=\frac{z-1}{T}} \quad \text{(F-6)}$$

$$c/cs(z) = Z^3 + (T_s k_x^* + T_s v_r k_\theta^* - 3)Z^2 + (T_s^2 k_x^* v_r k_\theta^* + T_s^2 v_r^2 k_y^* + T_s^2 w_r^2 - 2T_s\, v_r k_\theta^* - 2T_s k_x^* + 3)Z$$

$$+ (-T_s k_x^* - T_s v_r k_\theta^* - T_s^2 k_x^* v_r k_\theta^* - T_s^2 w_r^2 - T_s^2 v_r^2 k_y^* + T_s^3 w_r^2 v_r k_\theta^* + T_s^3 k_x v_r^2 k_y^* - 1). \quad \text{(F-7)}$$

Appendix G

Determination of the Optimal Neural Network Size and its Validation

Basically, the number of nodes required in the hidden layer depends on the inputs of the network as well as the number of patterns to be learnt. For J number of input nodes, 2J+1 number of hidden nodes are usually used; again for learning p different patterns, maximum p-1 number of hidden nodes is required [39, 116, 119, 120]. However, to obtain the optimum number of hidden layer nodes the following method can be used:

1- Divide the database into two sets, one for learning (the learning set must have a sufficient number of patterns that adequately cover the specific operating region) and the other for testing the neural network model after learning.

2- Use the learning set to train the neural network until it has a sufficiently small MSE, defined as follows:

$$MSE = \frac{1}{p}\sum_{i=1}^{p}(y^{(i)} - \hat{y}^{(i)})^2 \qquad \text{(G-1)}$$

where $\hat{y}$ is the predicted value of the observation y, and p is the number of patterns in the learning set. Plot the observations (y)s along with the predictions ($\hat{y}$)s of the neural network based on the learning and test sets. Now, if the error curve ($(y^{(i)} - \hat{y}^{(i)})$; i = 1,2,…p) obtained based on the learning set is more suppressed than that based on the test set (which indicates that the network has learnt the learning patterns in a point wise fashion rather than learning the nonlinear functionality that they represent), eliminate one hidden node and repeat step 2. If the error curve obtained based on the learning set is approximately the same (in an overall sense) as that obtained based on the test set, the number of the hidden nodes will be the optimum one and the neural network has a good generalization capability.

Appendix H

Cubic spline Interpolation Technique

The curves are third order polynomials,

$$f_i(x) = a_i + b_i x_i + c_i x_i^2 + d_i x_i^3 \tag{H-1}$$

$$f'(x_i) = \frac{2}{\frac{x_{i+1} - x_i}{y_{i+1} - y_i} + \frac{x_i - x_{i-1}}{y_i - y_{i-1}}} \tag{H-2}$$

$$f_1'(x_0) = \frac{3(y_1 - y_0)}{2(x_1 - x_0)} - \frac{f'(x_1)}{2} \tag{H-3}$$

$$f_n'(x_n) = \frac{3(y_n - y_{n-1})}{2(x_n - x_{n-1})} - \frac{f'(x_{n-1})}{2} \tag{H-4}$$

$$f_i''(x_{i-1}) = \frac{2[f_i'(x_i) + 2f_i'(x_{i-1})]}{(x_i - x_{i-1})} + \frac{6(y_i - y_{i-1})}{(x_i - x_{i-1})^2} \tag{H-5}$$

$$f_i''(x_i) = \frac{2[2f_i'(x_i) + f_i'(x_{i-1})]}{(x_i - x_{i-1})} - \frac{6(y_i - y_{i-1})}{(x_i - x_{i-1})^2} \tag{H-6}$$

$$d_i = \frac{f_i''(x_i) + f_i''(x_{i-1})}{6(x_i - x_{i-1})} \tag{H-7}$$

$$c_i = \frac{x_i f_i''(x_{i-1}) - x_{i-1} f_i''(x_i)}{2(x_i - x_{i-1})} \tag{H-8}$$

$$b_i = \frac{(y_i - y_{i-1}) - c_i(x_i^2 - x_{i-1}^2) - di(x_i^3 - x_{i-1}^3)}{(x_i - x_{i-1})} \tag{H-9}$$

$$a_i = y_{i-1} - b_i x_{i-1} - c_i x_{i-1}^2 - d_i x_{i-1}^3 \tag{H-10}$$

LIST OF PUBLICATIONS

The following publications originated from this thesis:

[1] **Ahmed S. Al-Araji**, Maysam F. Abbod and Hamed S. Al-Raweshidy, Design of a neural predictive controller for nonholonomic mobile robot based on posture identifier. Proceedings of the 13th IASTED International Conference on Intelligent Systems and Control (ISC 2011). Cambridge, United Kingdom, July 11 - 13, 2011. pp. 198-207.

[2] **Ahmed S. Al-Araji**, Maysam F. Abbod and Hamed S. Al-Raweshidy, Design of an adaptive nonlinear PID controller for nonholonomic mobile robot based on posture identifier. Proceedings of the 2011 IEEE International Conference on Control System, Computing and Engineering (ICCSCE 2011). Penang, Malaysia, November 25-27, 2011, pp. 337-342.

[3] **Ahmed S. Al-Araji**, Maysam F. Abbod and Hamed S. Al-Raweshidy, Neural Autopilot predictive controller for nonholonomic wheeled mobile robot based on a pre-assigned posture identifier in the presence of disturbance. Proceedings of the 2nd International Conference on Control, Instrumentation and Automation (ICCIA 2011). Shiraz, Iran, Dec 27-29, 2011.

[4] **Ahmed S. Al-Araji**, Maysam F. Abbod and Hamed S. Al-Raweshidy, Design of an adaptive predictive nonlinear controller for nonholonomic mobile robot system based on posture identifier in the presence of disturbance. International Journal of Simulation, System, Science and Technology (IJSSST). Vol. 12, No. 3, 2012, pp. 17-28.

[5] **Ahmed S. Al-Araji**, Maysam F. Abbod and Hamed S. Al-Raweshidy, Applying posture identifier and back-stepping method in the design of an adaptive nonlinear predictive controller for nonholonomic mobile robot. Proceedings of the 31st IASTED Asian Conference on Modeling, Identification and Control (AsiaMIC 2012). Phuket, Thailand, April 2-4, 2012, pp. 110-117.

[6] **Ahmed S. Al-Araji**, Maysam F. Abbod and Hamed S. Al-Raweshidy, Applying posture identifier in designing an adaptive nonlinear predictive controller for nonholonomic mobile robot. Accepted in Neurocomputing Elsevier Journal, 8-6-2012 and available on-line 24-8-2012.