UNIVERSITY OF CINCINNATI

September 23, 20 03

I, Souma Mahmoud Alhaj Ali **,**
hereby submit this as part of the requirements for the degree of:

DOCTORATE OF PHILOSOPHY (Ph.D.)

in:

Industrial Engineering

It is entitled:

Technologies for Autonomous Navigation in Unstructured
Outdoor Environments

Approved by:

Dr. Ernest L. Hall
Dr. Thomas R. Huston
Dr. Ronald L. Huston
Dr. Richard L. Shell
Dr. Issam A. Minkarah

Technologies for Autonomous Navigation in Unstructured Outdoor Environments

A dissertation submitted to the

Division of Research and Advanced Studies
of the University of Cincinnati

in partial fulfillment of the
requirements for the degree of

DOCTORATE OF PHILOSOPHY (Ph.D.)

In the Department of Mechanical, Industrial and Nuclear Engineering
of the College of Engineering

2003

by

Souma Mahmoud Alhaj Ali

B.S., University of Jordan, 1995
M.S., University of Jordan, 1998

Committee Chair: Dr. Ernest L. Hall

ABSTRACT

Robots have been used in manufacturing and service industries to improve productivity, quality, and flexibility. Robots are usually mounted on a fixed plate, or on rails, and can move in a limited manner. The success of robots in these environments encourages the use of mobile robots in other applications where the environments are not structured, such as in outdoor environments.

This dissertation presents the development of an autonomous navigation and obstacle avoidance system for a Wheeled Mobile Robot (WMR) operating in unstructured outdoor environments. The algorithm produces the robot's path positioned within the road boundaries and avoids any fixed obstacles along the path. The navigation algorithm was developed from a feedforward multilayer neural network. The network used a quasi-Newton backpropagation algorithm for training.

Proportional-plus-derivative computed-torque, proportional-plus-integral-plus-derivative computed-torque, digital, and adaptive controllers were developed to select suitable control torques for the motors, which cause the robot to follow the desired path from the navigation algorithm.

Simulation software permitting easy investigation of alternative architectures was developed by using Matlab and C++. The simulation software for the controllers was developed for two case studies. The first case study is the two-link robot manipulator, and the second is a navigation controller for the WMR. The simulation software for the WMR navigation controller used the Bearcat III dynamic model, developed in this dissertation.

Simulation results verify the effectiveness of the navigation algorithm and the controllers. The navigation algorithm was able to produce a path with a small mean square error, compared

to the targeted path, which was developed by using an experienced driver. The algorithm also produced acceptable results when tested with different kinds of roads and obstacles. The controllers found suitable control torques, permitting the robot to follow these paths. The digital controller produced the best results.

The significance of this work is the development of a dynamic system model and controllers for WMR navigation, rather than robot manipulators, which is a new research area. In addition, the navigation system can be utilized in numerous applications, including various defense, industrial and medical robots.

To:

my deceased father,

my mother,

my husband, Ayman,

and my sons, Mahmoud and Omar

ACKNOWLEDGEMENTS

The author wish to express her appreciation and gratitude towards her advisor: Dr. Ernest L. Hall for his support and guidance. His continued guidance has helped the author in directing this dissertation towards achieving valuable contributions. The author conveys her heartiest thanks to the other dissertation committee members: Dr. Richard L. Shell, Dr. Issam A. Minkarah, Dr. Ronald L. Huston, and Dr. Thomas R. Huston for their continued support that enabled the student to achieve the objectives of her research.

The author expresses her gratitude to all who provided her with continued support: her mother, husband, brothers and sisters.

Table of Contents

List of Figures

[1] J. Burnstein, http://www.robotics.org/public/articles/articles.cfm?cat=201

List of Tables

List of symbols

ξ	The robot posture.
x, y, θ	The coordination and the orientation of the center of gravity of the robot as shown in Fig. 3.2.
$\alpha, \beta, l, \varphi$	As shown in Fig. 3.3-4.3.
r	Wheel radius.
β_s, β_c	The orientation angles of the steering wheels and the castor wheels respectively.
$J_{1f}, J_{1s}, J_{1c}, J_{1sw}$	Matrices of size $N_f \times 3$, $N_s \times 3$, $N_c \times 3$, $N_{sw} \times 3$ respectively, their

	forms derived directly from the mobility constrains in Eq. (3.3)-(6.3).
J_2	Constant $(N \times N)$ matrix, whose diagonal entries are the radii of the wheels, except for the radii of the Swedish wheel, those need to be multiplied by cosine the angle of the contact point.
C_{1f}, C_{1s}, C_{1c}	Matrices of size $N_f \times 3$, $N_s \times 3$, $N_c \times 3$, respectively, their forms derived directly from the non-slipping constrains in Eq. (4.3) and (6.3) respectively.
δ_m	The degree of mobility.
δ_s	The degree of steerability.
η_1	Robot velocity component along Ym as shown in Fig. 3.5.
η_2	Robot angular velocity ω.
$\dot{x}$, $\dot{y}$	The rate of change of the position of the robot.
$\dot{\theta}$	The rate of change of the orientation of the robot.
M	The degree of nonholonomy of a mobile robot
τ_φ, τ_c, and τ_s	The torques that can be potentially applied to the rotation and orientation of robot wheels
F_r	The reaction force applied to the right wheel by the rest of the robot.
f_r	The friction force between the right wheel and the ground.
m_w	The mass of the wheel.
τ_r	The torque acting on the right wheel that is provided by the right motor.
J_w	The inertia of the wheel.

μ	The maximum static friction coefficient between the wheel and the ground.
P	The reaction force applied on the wheel by the rest of the robot.
F_l	The reaction force applied to the left wheel by the rest of the robot.
f_l	The friction force between the left wheel and the ground.
τ_l	The torque acting on the left wheel that is provided by the left motor.
f_f	The reaction force applied to the robot by the front wheel (castor wheel).
f_n	The resultant normal force.
m	The mass of the robot excluding the wheels.
J_c	The inertia of the robot excluding the wheels.
x_c, y_c, θ	The coordination and the orientation of the center of gravity of the robot.
$M(q)$	The inertia matrix.
$V_m(q,\dot{q})$	The Coriolis/centripetal matrix.
$F(\dot{q})$	The friction terms.
$G(q)$	The gravity vector
τ_d	The torque resulted from the disturbances.
τ	The control input torque.
r_e	The exterior radius of the disc.
r_i	The interior radius of the disc.

N	The normal force between the wheel and the ground.
f_c	The centrifugal force.
υ	The velocity of the robot in the direction toward the center of the circular path.
ρ	The radius of the circular path.
$x_1(t),...,x(t)_n$	ANN cell inputs.
$y(t)$	ANN cell output.
$v_1,....,v_n$	ANN dendrite or input weights.
v_0	ANN firing threshold or bias weight.
f	ANN cell function or activation function
f_1, f_2, f_3, f_4, f_5	The activation function for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.
$n1, n2, n3, n4, n5$	The number of input signals for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.
L	The number of outputs for layer 5.
$v_{n2n1}, v_{n3n2}, v_{n4n3}, v_{n5n4}, v_{Ln5}$	The input weights for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.
$v_{n20}, v_{n30}, v_{n40}, v_{n50}, v_{L0}$	The bias weights for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.
b	Margin vector.
d^k	Desired target vector associated with input vector x.
$f(net)$	Differentiable activation function, usually a sigmoid function.

i^*	Label of a wining neuron in a competitive net.
$J(w)$	Objective (criterion) function.
net	Weigted-sum or $w^T x$.
$p(x)$	Probability density function of x.
T	Threshold constant.
w	Weight vector
w_i^*	Weight vector of wining unit *i*.
w^*	Solution weight vector
α	Momentum rate.
ε	Positive constant, usually arbitrary small.
ε^k	Error vector at the kth step of the AHK algorithm
ρ	Learning rate.
r	Positive vector of a unit in a neural field.
$r^k, r(y, x^k)$	Reinforcement signal.
i^*	Label of a wining neuron in a competitive net.
$\Phi(r, r^*)$	Continuous neighborhood function with center at r^* .
q_1	The actual orientation of the first link of the manipulator.
q_2	The actual orientation of the second link of the manipulator.
m_1	The mass of the first link of the manipulator.
m_2	The mass of the second link of the manipulator.
τ_1	The torque of the first link of the manipulator.

τ_2	The torque of the second link of the manipulator.
q_{1d}	The desired orientation of the first link of the manipulator.
q_{2d}	The desired orientation of the second link of the manipulator.
k_p	A proportional gain matrix.
k_v	A derivative gain matrix.
k_i	An integrator gain matrix.
$q_d(t)$	The desired motion trajectory.
x_{cd}	The x-axis component of the desired position of the WMR center of gravity.
y_{cd}	The y-axis component of the desired position of the WMR center of gravity.
θ_d	The desired orientation of the WMR.
$q(t)$	The actual path.
x_c	The x-axis component of the actual position of the WMR center of gravity.
y_c	The y-axis component of the actual position of the WMR center of gravity.
θ	The actual orientation of the WMR.
T	The digital controller sample period.
ε	The integer value.
Λ	A positive definite design parameter matrix, commonly selected as a diagonal matrix with large positive entries for the approximation-based

	adaptive controller.
$\sigma_{\min}$	The minimum singular value of Λ, and the 2-norm is used.
q_b	A scalar bound.
$W(x)$	A matrix of known robot functions.
$\boldsymbol{\Phi}$	A vector of unknown parameters, such as masses and friction coefficients.
Γ	A tuning parameter matrix, usually selected as a diagonal matrix with positive elements.

Chapter 1: Introduction

1.1 Introduction

Intelligent machines that can perform tedious, routine, or dangerous tasks offer great promise to augment human activities. Automated Guided Vehicles (AGV's) are currently used in manufacturing where the environment can be structured, as in the case of handling raw materials, transporting work in process, and delivering finished products inside manufacturing and service plants. For example, AGV's are currently being used to transport heavy metal plates between workstations for the construction of machine tools at the Mazak facility in Florence, Kentucky. Also, personal robots, such as the Helpmate by Transitions Research Corporation, are being used to deliver food to patients in hospitals. Many single task mobile robots have also been studied in the defense arena. However, the development of useful autonomous vehicles presents a challenge to researchers when the environment is not structured, as when the environment is dynamically changing with multiple obstacles. Today's robots cannot be used in those environments without direct human supervision for safety. In unstructured environments, such as in outdoor applications, it is currently difficult or dangerous for robots to perform the task. In some environments, the task is left undone, because it is too dangerous for a human operator to perform. This is the case in detecting and disarming minefields, for example. The need for robots to work in these unstructured environments is clear.

Navigation in an unstructured environment includes such problems as obstacle avoidance, avoidance of hazards, such as holes, boulders, or dangerous locations. Another problem is the ability to move to a desired location and sometimes search an entire region. Various scenarios can easily be envisioned where problems are extremely difficult to solve by autonomous vehicles.

This research is concerned with developing an autonomous navigation and obstacle avoidance system for a WMR operation in an unstructured outdoor environment. The navigation system developed in this research uses a Differential Global Positioning System (DGPS) receiver, in addition to the vision system. The navigation algorithm uses input from a DGPS receiver and a laser scanner. The algorithm uses Artificial Neural Networks (ANN) to produce a path for the robot. The robot path is positioned within the road boundaries and avoids obstacles if there are any in the road.

Proportional-plus-Derivative (PD) Computed-Torque (CT), Proportional-plus-Integral-plus-Derivative (PID) CT, digital, and adaptive controllers have been developed in order to select the suitable control torques to the motors that causes the robot to follow the desired path that was produced from the navigation algorithm.

To permit the widest exploration of the proposed algorithm and controller, a simulation approach is followed. From the initial problem formulation, simulation software is developed, which permits an easy investigation of alternative architectures. The software is developed by using Matlab and C++.

1.2 Research Goals:

1. To develop a navigation and obstacle avoidance algorithm for unstructured outdoor environments. This algorithm will cover fixed obstacles.
2. To develop simulation software to test alternative navigation scenarios and permit the widest exploration of the proposed algorithm.
3. To develop a kinematic and dynamic model for WMRs.
4. To develop controllers that will select suitable control torques, so that the robot will follow the desired path produced by the navigation algorithm.

5. To develop simulation software to test the efficiency and effectiveness of the controllers on different paths for two case studies, namely, the two-link robot manipulator and WMR navigation.

1.3 Significance

Navigation in an unstructured environment is an open area for research. In the current literature, there is no information about reliable robot operations in an outdoor or unstructured environment. Most of the previous research in this area covered the navigation in a structured indoor environment. However, in the proposed research, and in many other applications, the environment is no longer structured. In most cases, there is no prior knowledge of the environment or the path, presence of wild terrain, and unpredictable obstacles. The available literature does not investigate these applications.

Robot utilization in defense applications is very important, since it can reduce casualties and risk to military personnel. Moreover, it will improve the performance and efficiency of the military missions. Most of these applications are in an outdoor unstructured environment. Therefore, developing a navigation system for unstructured environments helps utilize robots in defense applications.

Some of the major challenges for this research are:

- An effective navigation and obstacle avoidance algorithm for an unstructured outdoor environment.
- Sensory devices capable of giving a proper description of the environment to the controller.
- Controllers capable of producing the robot control torque that will produce the targeted robot path.

Similar operations performed by humans are very limited, for the following reasons:

- The limited capabilities of humans in terms of force, speed of operation, and tolerance to poisonous gas or low-level radiation, besides the psychological issues.
- Some of the conditions are dangerous to humans.

In addition, the navigation algorithm and the controller design developed in this research can be utilized in numerous applications. They include the defense robot, discovery or explorer robot, rescue robot, undersea vehicles, passenger transport in urban areas, and personal robots. The navigation algorithm and the controller can also be used in many unstructured outdoor applications, as in construction and agriculture. Actually, they are also needed for any robot working in real time applications, since all of these applications have varying degrees of uncertainty.

1.4 Contribution and relation to the present state of knowledge in the field

Many navigation methods have been developed. However, most of these are limited to structured environments. Despite the huge applications of autonomous navigation in unstructured environments, this is an area open to research.

Expected contributions to the existing body of knowledge in this area of research are summarized below:

1. The development of simulation for an autonomous navigation and obstacle avoidance system. This system will enable the robot to navigate in an outdoor unstructured environment, such as a road marked by boundaries and cluttered with obstacles. The navigation system uses an algorithm that receives information from a DGPS receiver and a laser scanner. The system produces the robot path that is positioned within the road boundaries and avoids any fixed obstacles along the path. The proposed algorithm uses

ANN to generate a path, which will enable the robot to learn its environment and improve its performance.

2. The development of a kinematic and dynamic model for WMR.
3. The development of simulation for PD CT, PID CT, digital, and adaptive controllers for WMR navigation. The controller task is to select robot control torque needed to produce the desired path developed by the navigation algorithm.

The broader impact of this research is that autonomous navigation in an outdoor unstructured environment has numerous applications that will help society, such as personal robots and passenger transporters. Success in developing the navigation algorithm and the controllers in this research will open the way to invent robots for other applications.

1.5 Methodology

Step 1: Review the literature in the following areas

1. Navigation and obstacle avoidance systems.
2. Control techniques.
3. Robot modeling.
4. Sensory devices available.
5. Simulation.

Step 2: Develop the navigation and obstacle avoidance system

1. Develop the mathematical formulations.
2. Choose the appropriate sensory devices.
3. Determine the inputs and outputs.
4. Develop the navigation and obstacle avoidance algorithm.

Step 3: Develop the simulation software for the navigation algorithm

1. Develop the model assumptions.
2. Build the model.
3. Verify the model.
4. Validate the model.

Step 4: Conduct experiments on the navigation algorithm using the simulation software

1. Test different alternative methods of navigation.
2. Vary the parameters.
3. Analyze the results and modify the algorithm accordingly.

Step 5: Develop the kinematic and dynamic model for the robot

1. Study the types and structures available for the WMR.
2. Develop the mathematical formulations for robot kinematics.
3. Customize the kinematic model for Bearcat III.
4. Analyze all the forces affecting the robot motion.
5. Develop the mathematical formulations for robot dynamics.
6. Customize the dynamic model for Bearcat III.

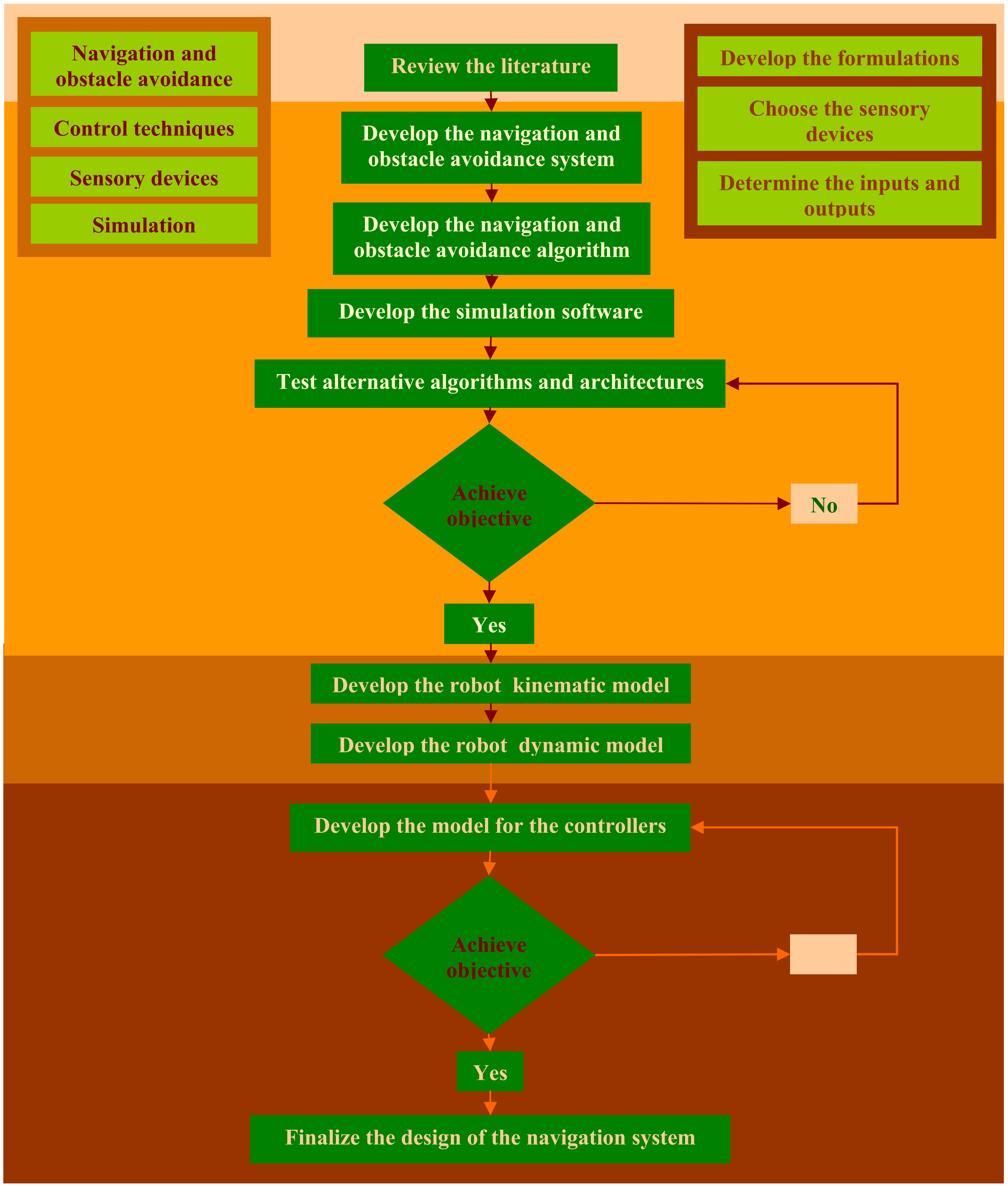
Step 6: Develop the formulation for controllers

1. Study the available controllers.
2. Develop formulations and models for controllers.

Step 7: Develop the simulation software for controllers

1. Build the software.
2. Verify the model.
3. Validate the model.

The methodology is shown in Fig 1.1.


Navigation and obstacle avoidance
Control techniques
Sensory devices
Simulation
Review the literature
Develop the navigation and obstacle avoidance system
Develop the navigation and obstacle avoidance algorithm
Develop the simulation software
Test alternative algorithms and architectures
Develop the formulations
Choose the sensory devices
Determine the inputs and outputs
Achieve objective
No
Yes
Develop the robot kinematic model
Develop the robot dynamic model
Develop the model for the controllers
Achieve objective
Yes
Finalize the design of the navigation system
Figure 1.1: The research methodology.

Chapter 2: Background

2.1 Introduction

A robot is a reprogrammable machine that is capable of processing some human-like activities such as judgment, reasoning, learning, and vision [1].

Another definition for the industrial robot is “a reprogrammable multi-functional manipulator designed to move material, parts, tools or specialized devices, through variable programmed motions for the performance of a variety of tasks” as defined by the Robot Industries Association (RIA). Robots or manipulators are usually modeled as an open kinematic chain of rigid bodies called links, which are interconnected by joints. Some of them have closed kinematic chains such as four bar mechanisms for some links, others mounted on a fixed pedestal base that is connected to other links. Robots manipulate objects and perform tasks by their end-effecter that is attaches to the free end. The advantages that robots offer over hard automation or Computer Numerical Control (CNC) machines are their flexibility and the variety of tasks that they can perform. However, robots are still limited in their sensory capabilities both the vision and tactile, flexibility, and in their ability to adapt, learn and having some creativity [1]. Current research is being conducted to develop a robot that is able to respond to changes to its environment [2].

While robotics research has mainly been concerned with vision (eyes) and tactile, some problems regarding adapting, reasoning, and responding to changed environment have been solved with the help of artificial intelligence using heuristic methods such as ANN. Neural computers have been suggested to provide a higher level of intelligence that allows the robot to plan its action in a normal environment as well as to perform non-programmed tasks [1]. A well established field in the discipline of control systems is the intelligent control, which represents a generalization of the concept of control, to include autonomous anthropomorphic interactions of a machine with the environment [3].

Meystel and Albus [3] defined intelligence as “the ability of a system to act appropriately in an uncertain environment, where an appropriate action is that which increases the probability of success, and success is the achievement of the behavioral sub goals that support the system’s ultimate goal”. The intelligent systems act so as to maximize this probability. Both goals and success criteria are generated in the environment external to the intelligent system. At a minimum, the intelligent system had to be able to sense the environment which can be achieved by the use of sensors, then perceive and interpret the situation in order to make decisions by the use of analyzers, and finally implements the proper control actions by using actuators or drives. Higher levels of intelligence require the abilities to: recognize objects and events, store and use knowledge about the world, learn, and to reason about and plan for future. Advanced forms of intelligence have the ability to perceive and analyze, to plan and scheme, to choose wisely, and plan successfully in a complex, competitive, and hostile world [3].

Flexibility of automation systems had been enhanced due to the robot’s reprogram ability.

First commercially marketed in 1956 by a firm named Unimation, industrial robots improve plant productivity, flexibility, and it improved the products quality. The use of robots is justified based on the productive and reliable performance they offer. Social impacts on labor skills and employment opportunities are still to be adequately addressed [4].

Not until recently, that the US manufacturers have realized robots significant impact on productivity and flexibility [1].

Robots are currently in use in many applications in the manufacturing industry including: welding, sealing, painting, material handling, assembly, and inspection; and in service industry where the environments can be structured [5]. In order for the robot to operate successfully machine sensing, perception, cognition, and action are required. The task is complex because of the dynamic environment. Both temporal and spatial relationships must be represented and used in planning the action. Furthermore, some linking or communications or cooperation may be required between the robot and other objects in the environment.

According to the (RIA)[2], the robot industry in the US starts to recover from the year 1993; the actual number of robots delivered to the customer from the US manufacturers for the years 1984-2001 are shown in Fig. 2.1. Where the numbers of years 1984-1995 are based on actual reports from IRA member companies plus an estimate of robot sales done by IRA, while the numbers for the years 1995-2001 are the actual reports from IRA member companies.

A robot can be viewed as a set of integrated subsystems [1, 6-7]:

1. **Manipulator:** The manipulators are a series of links that are connected by joints. Their motion is accomplished by using actuators as pneumatic cylinders.
2. **Sensory or feedback devices**: Sensors that monitor the position, velocity, and acceleration of various linkages and joints and feed this information to the controller.
3. **Controller:** Computer used to analyze the sensory information and generate signals for the drive system so as to take the proper actions.
4. **Power source:** Power systems that may be of electric, pneumatic, or hydraulic sources used to provide and regulate the energy needed for the manipulator's actuators and drives.

[2] J. Burnstein (2001, July). Robotics Industry Statistics. Robotic Industries Association, Robotics online Ann Arbor, MI. [Online]. Available: http://www.robotics.org/public/articles/articles.cfm?cat=201

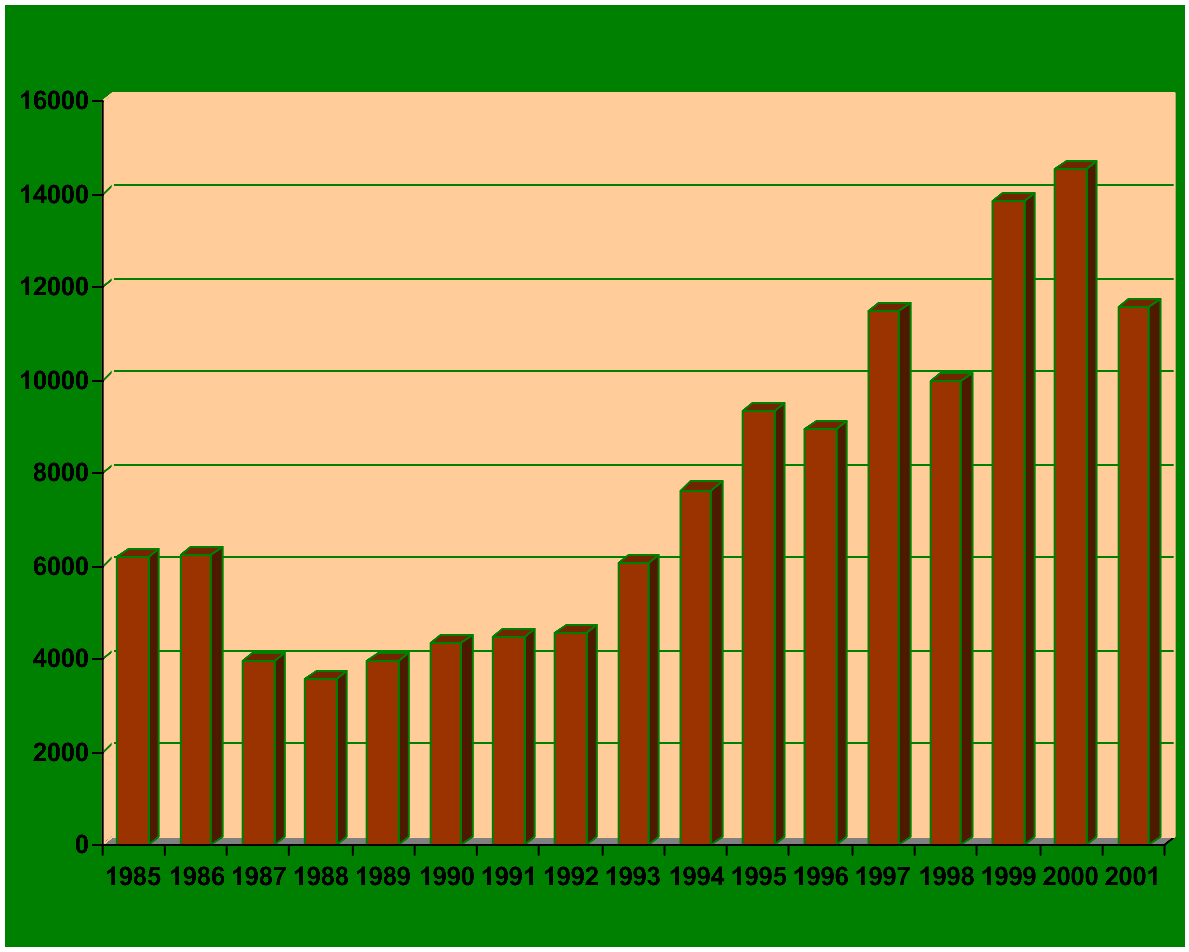


Figure 2.1: Actual number of robots delivered to the customer from the US manufacturers[3].

Based on the kinematic structure of the various joints and links and their relationships with each other, different manipulators configuration are available. To position and orient an object in a three-dimensional space there are six basic degrees of freedom. The major links (arm and body motions) perform the gross manipulation tasks which are positioning, as in arc welding, spray painting, and water jet cutting applications. The last three links (wrist movements) which are called the minor links perform the fine manipulation tasks. If having more than six axes of motion, robots are called redundant degree of freedom robots. These axes

[3] J. Burnstein (2001, July). Robotics Industry Statistics. Robotic Industries Association, Robotics online Ann Arbor, MI. [Online]. Available: http://www.robotics.org/public/articles/articles.cfm?cat=201

are used for greater flexibility. Typical joints are revolute (R) joints, or prismatic (P) joints. R joints provide rotational motion, while P joints provide sliding motion [1].

The major mechanical configurations commonly used for robots are cartesian, cylindrical, spherical, articulated, and selective compliance articulated robot for assembly. To select a particular robot for an application consideration such as workplace coverage, particular reach, and collision avoidance have to taken into account. For further details refer to Lasky and Hsia [8].

In this section a review of the literature for the main challenges in this research will be provided.

2.2 Navigation

A navigation system is the method for guiding a vehicle. Since the vehicle is in continuous motion, the navigation system should extract a representation of the world from the moving images and other sensory information that it receives. Several capabilities are needed for autonomous navigation. One is the ability to execute elementary goal achieving actions such as going to a given location or following a leader. Another is the ability to react to unexpected events in real time such as avoiding a suddenly appearing obstacle. Another capability is map formation such as building, maintaining, and using a map of the environment. Another is learning which might include noting the location of an obstacle on the map so that they could be avoided in the future. Other learning capabilities might focus on the three-dimensional nature of the terrain and adapt the drive torque to the inclination of hills. Another capability is planning such as formation of plans that accomplish specific goals or avoid certain situations such as traps [1].

Previously the research community has taken two quite different approaches to mobile robot design. In autonomous vehicle research, one goal has been to develop a vehicle that can navigate at relatively high speed in outdoor environments such as on roadways or in open terrain. Such vehicles require massive computational power to process visual information in real time and provide sensory feedback for control. The second direction taken has been to design mobile robots with relatively simple sensors such as sonar and communications capability for cooperative behavior similar to swarms of bees or ants. In both approaches, there is a need for real time decision-making based on sensed information from the environment and for learning. This may be contrasted to the stationary industrial robots in which most of the decision making may be preprogrammed or planned off-line or by teach programming [9].

During the last fifteen years, a great deal of research has been done on the interpretation of motion fields as the information they contain about the 3-D world. In general, the problem is compounded by the fact that the information that can be derived from the sequence of images is not the exact projection of the 3D-motion field, but rather only information about the movement of light patterns, optical flow [1].

About 40 years ago, Hubel and Wiesel [10] studied the visual cortex of the cat and found simple, complex, and hyper complex cells. In fact, vision in animals is connected with action in two senses: “vision for action” and “action for vision” [11].

The general theory for mobile robotic navigation is based on a simple premise. For a mobile robot to operate it must sense the known world, be able to plan its operations, and then act based on this model. This theory of operation has become known as SMPA (Sense, Map, Plan, and Act) [1-2].

SMPA was accepted as the normal theory until around 1984 when a number of people started to think about the more general problem of organizing intelligence. There was a requirement that intelligence be reactive to dynamic aspects of the unknown environments, that a mobile robot operate on time scales similar to those of animals and humans, and that intelligence be able to generate robust behavior in the face of uncertain sensors, unpredictable environments, and a changing world. This led to the development of the theory of reactive navigation by using Artificial Intelligence (AI) [12].

2.2.1 Systems and Methods for mobile robot navigation

2.2.1.1 Odometry and Other Dead-Reckoning Methods

Odometry is one of the most widely used navigation methods for mobile robot positioning. It provides good short-term accuracy, an inexpensive facility, and allows high sampling rates. This method uses encoders to measure wheel rotation and/or steering orientation. Odometry has the advantage that it is totally self-contained, and it is always capable of providing the vehicle with an estimate of its position. The disadvantage of odometry is that the position error grows without bound unless an independent reference is used periodically to reduce the error [13].

2.2.1.2 Inertial Navigation

This method uses gyroscopes and sometimes accelerometers to measure the rate of rotation and acceleration. Measurements are integrated once (or twice) to yield position. Inertial navigation systems also have the advantage that they are self-contained. On the downside, inertial sensor data drifts with time because of the need to integrate rate data to yield position; any small constant error increases without bound after integration. Inertial sensors are thus unsuitable for accurate positioning over an extended period of time. Another problem with inertial navigation is the high equipment cost. For example, highly accurate gyros, used in airplanes, are prohibitively expensive. Very recently fiber-optic gyros (also called laser gyros), which have been developed and said to be very accurate, have fallen dramatically in price and have become a very attractive solution for mobile robot navigation [14-15].

2.2.1.3 Active Beacon Navigation Systems

This method computes the absolute position of the robot from measuring the direction of incidence of three or more actively transmitted beacons. The transmitters, usually using light or radio frequencies must be located at known sites in the environment [16-17].

2.2.1.4 Landmark Navigation

In this method distinctive artificial landmarks are placed at known locations in the environment. The advantage of artificial landmarks is that they can be designed for optimal detect-ability even under adverse environmental conditions. As with active beacons, three or more landmarks must be "in view" to allow position estimation. Landmark positioning has the advantage that the position errors are bounded, but detection of external landmarks and real-time position fixing may not always be possible. Unlike the usually point-shaped beacons, artificial landmarks may be defined as a set of features, e.g., a shape or an area. Additional information, for example, distance, can be derived from measuring the geometric properties of the landmark, but this approach is computationally intensive and not very accurate [18].

2.2.1.5 Map-based Positioning

In this method information acquired from the robot's onboard sensors is compared to a map or world model of the environment. If features from the sensor-based map and the world model map match, then the vehicle's absolute location can be estimated. Map-based positioning often includes improving global maps based on the new sensory observations in a dynamic environment and integrating local maps into the global map to cover previously unexplored areas. The maps used in navigation include geometric and topological maps. Geometric maps represent the world in a global coordinate system, while topological maps represent the world as a network of nodes and arcs [19-20].

2.2.1.6 Global Positioning System (GPS)

GPS is a worldwide radio-navigation system formed from a constellation of 24 satellites and their ground stations [21]. GPS is funded and controlled by the U. S. Department of Defense (DOD). Originally, it was designed for and operated by the U. S. military. The system provides specially coded satellite signals that can be processed in a GPS receiver, enabling it to compute position, velocity and time. Four GPS satellite signals are used to compute positions in three dimensions and the time offset in the receiver clock [22] as shown in Fig. 2.2.

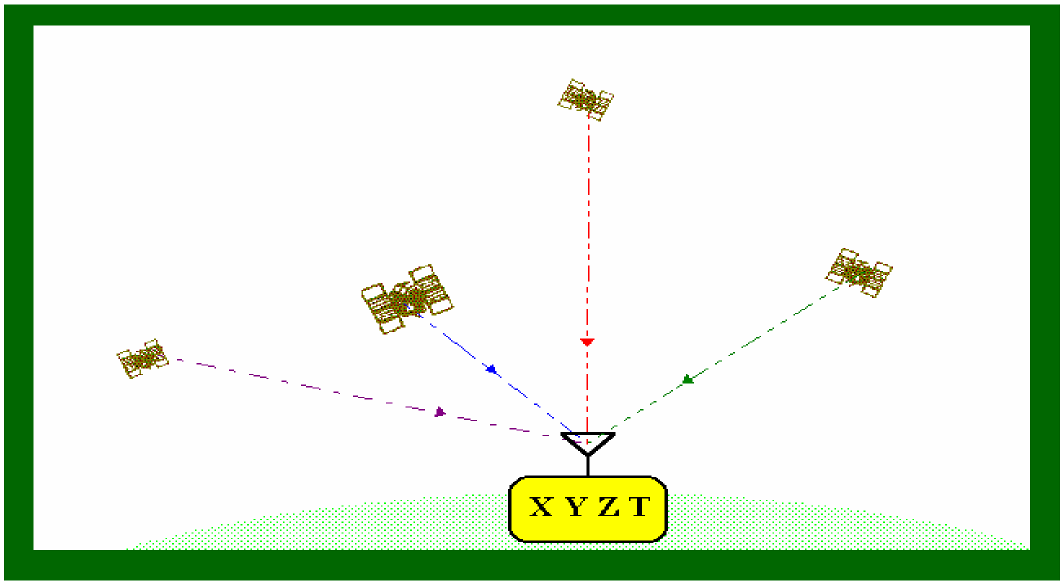


Figure 2.2: Measurements of code-phase arrival times from at least four satellites to estimate the position (X, Y, Z) and the GPS time (T) [22].

The space segment of the system consists of the 24 satellites that orbit the earth in 12 hours. These Space Vehicles (SVs) send radio signals from space. There are often more than 24 operational satellites as new ones are launched to replace older satellites. The satellite orbits repeat almost the same ground track (as the earth turns beneath them) once each day. The orbit altitude is such that the satellites repeat the same track and configuration over any point approximately each 24 hours (4 minutes earlier each day). There are six equally spaced orbital planes and inclined at about fifty-five degrees with respect to the equatorial plane. This constellation provides the user with between five and eight SVs visible from any point on the earth [22] as shown in Fig. 2.3.

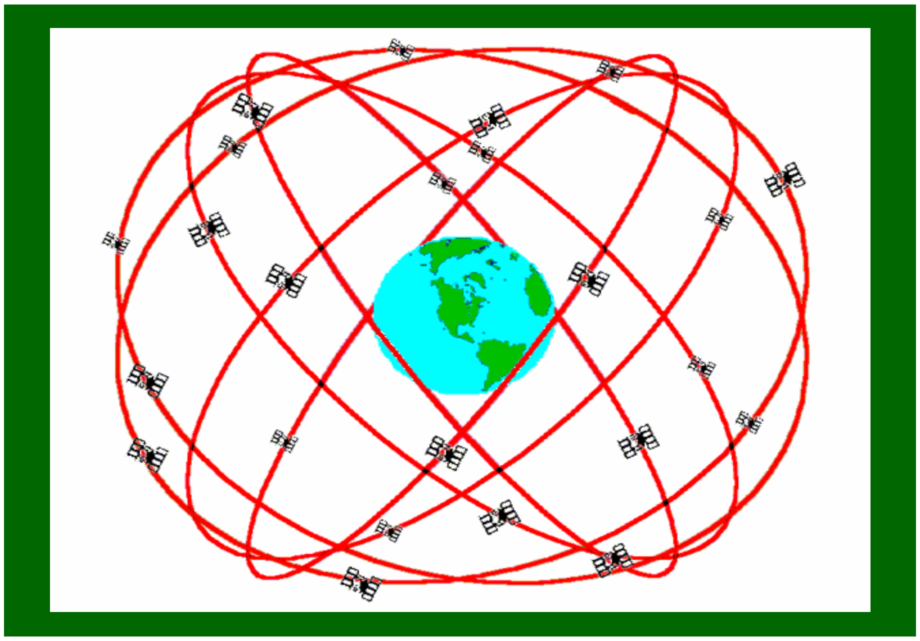


Figure 2.3: GPS nominal constellation, twenty-four satellites in six orbital planes (Four satellites in each plane) [22].

The control segment consists of a system of tracking stations located around the world, which measure signals from the SVs that are incorporated into orbital models for each satellite. The models compute precise orbital data (ephemeris) and SV clock corrections for each satellite. The master control station uploads ephemeris and clock data to the SVs. The SVs then send subsets of the orbital ephemeris data to GPS receivers over radio signals. The GPS user segment consists of the GPS receivers and the user community. GPS receivers convert SV signals into position, velocity, and time estimates [22]. The basis of GPS is triangulation (trilateration or resection) from satellites. That the GPS receiver measures distance using the travel time of radio signals.

The SVs transmit two microwave carrier signals L1 and L2. While the L1 frequency (1575.42 MHz) carries the navigation message and the SPS code signal, the L2 frequency (1227.60 MHz) is used to measure the ionospheric delay by PPS equipped receivers. Three binary codes shift the L1 and/or L2 carrier phase [22]:

1. The Coarse Acquisition (C/A) Code modulates the L1 carrier phase.
2. The Precise (P) Code modulates both the L1 and L2 carrier phases.
3. The Navigation Message modulates the L1-C/A code signal.

The GPS navigation message consists of time-tagged data bits marking the time of transmission of each sub frame at the time they are transmitted by the SV. A data bit frame consists of 1500 bits divided into five 300-bit sub frames. A data frame is transmitted every thirty seconds, the complete navigation message is composed of a set of twenty-five frames (125 sub frames) and sent over a 12.5 minute period [22].

The Pseudo Random Noise Code (PRNC) is a fundamental part of GPS. It is a complicated sequence of "on" and "off" pulses. The signal is so complicated that it almost looks like random electrical noise. Each satellite has its own unique PRNC; hence, it is guaranteed that the receiver won't accidentally pick up another satellite's signal. Therefore, all the satellites can use the same frequency without jamming each other. In addition, it makes it more difficult for a hostile force to jam the system. The PRNC also makes it possible to use information theory to amplify the GPS signal, that's why the GPS receivers don't need big satellite dishes to receive the GPS signals [21].

In simple words the GPS measure the distance as follows [21]:

1. Distance to a satellite is determined by measuring how long a radio signal takes to reach the receiver from that satellite.
2. To make the measurement it is assumed that both the satellite and the receiver are generating the same PRN at exactly the same time.
3. The travel time is determined by comparing how late the satellite's PRNC appears compared to the receiver's code.
4. The distance is then calculated by multiplying the travel time by the speed of light.

GPS receivers are used for navigation, positioning, time dissemination, and other research. Navigation receivers can be used for aircraft, ships, ground vehicles, and even for individuals [22].

The GPS signal contains some errors; the errors are a combination of noise, bias, and blunders. Noise errors are due to the noise of the PRNC and the noise within the receiver where each of them is around 1 meter. While bias errors result from Selective Availability (SA)[4] and other factors [22].

In order to correct bias errors the DGPS is evolved, where bias errors are corrected in the location of interest with measured bias errors at a known position. A reference receiver, or base station, computes corrections for each satellite signal.

---

[4] Selective Availability (SA) is the intentional degradation of the Standard Positioning Service signals by a time varying bias. SA is controlled by the DOD to limit accuracy for non-U. S. military and government users [22].

DGPS implementations require software in the reference receiver that can track all SVs in view and form individual pseudo-range corrections for each SV. These corrections however, has limited effect at useful ranges because both receivers would have to be using the same set of SVs in their navigation solutions and have identical geometric dilution of precision term [22].

2.3 Literature Review

The previous research on mobile robots has been divided into three topics by Wichert [23]:

- Systems that navigate with conventional distance sensors and use vision to find objects to be manipulated.
- Systems that directly couple the sensor output to motor controllers in a supervised learning process. Typical goals are road following or docking.
- Systems using landmark vision systems. Certain basic features such as edges, lines, and regions that are expected to be found in the environment are modeled. Then the model is matched to the environments by matching the landmarks.

Learning from the environment is also important for intelligent behavior. One approach that allows a robot to learn a model of its interaction with its operating environment in terms of experienced dynamics is described by Michaud and Mataric [24]. Another approach in which the robot adapts to environmental changes by efficiently transferring a learned behavior in previous environments into a new one and effectively modifying it to cope with the new environment is described by Minato and Asada [25]. In the following a brief review for the literature in these systems will be presented

2.3.1 Vision and sensor based navigation

For an intelligent robot that must adapt to environmental changes in situations in which humans thrive, vision sensing is necessary. Vision-based navigation had been presented in the literature [26-30]. Fork and Kabuka [31] present a navigation system for automatic guided vehicles that uses an efficient double heuristic search algorithm for path location. Beccari, et al. [32] described a vision-based line tracking system. A sensor composed of a fish-eye lens with a TV camera has been used by Kurata, et al. [33], they used a reference target on a ceiling and hybrid image processing circuits, the experimental testing showed that the proposed system was valid for an indoor navigation.

Sensor based navigation systems that rely on sonar or laser scanners that provide one dimensional distance profiles have been used for collision and obstacle avoidance. A general adaptable control structure is also required. The mobile robot must make decisions on its navigation tactics; decide which information to use to modify its position, which path to follow around obstacles, when stopping is the safest alternative, and which direction to proceed when no path is given. In addition, sensors information can be used for constructing maps of the environment for short term reactive planning and long-term environmental learning.

To detect any sensor failures or faults, Nebot and Durrant-Whyte [34] presented decentralized estimation architecture for multiple sensor systems, the architecture consists of several local loops where information filters are implemented, the information are calculated in each loop and communicated to other loops.

2.3.2 Use of fuzzy logic

Fuzzy logic and fuzzy languages has also been used in navigation algorithms for mobile robots as described in [35-41]. A fuzzy logic based real time navigation controller is described by Mora and Sanchez [42]. Lin and Wang [43] propose a fuzzy logic approach to guide an AGV from a starting point toward the target without colliding with any static obstacle as well as moving obstacles; they also study other issues as sensor modeling and trap recovery. Kim and Hyung [44] used fuzzy multi-attribute decision-making in deciding which via-point the robot should proceed to at each step. The via-point is a local target point for the robot's movement at each decision step. A set of candidate via-points is constructed at various headings and velocities. Watanabe, et al. [45] described a method using a fuzzy logic model for the control of a time varying rotational angle in which multiple linear models are obtained by utilizing the original non-linear model at some representative angles. New navigation strategies for intelligent mobile robot are described by Choi, et al. [46].

2.3.3 Use of ANN

ANN has also been applied to mobile robot navigation. It had been considered for applications that focus on recognition and classification of path features during navigation. Kurd and Oguchi [47] propose the use of neural network controller that was trained using supervised learning as an indirect-controller to obtain the best control parameters for the main controller in use with respect to the position of the AGV. A method that uses incremental learning and classification based on a self-organizing ANN is described by Vercelli and Morasso [48]. Xue and Cheung [49] proposed a neural network control scheme for controlling active suspension. The presented controller used a multi-layer back propagation neural network and a prediction-correction method for adjusting learning parameters. Dracopoulos [50] present the application of multi-layer perceptrons to the robot path planning problem and in particular to the task of maze navigation. Zhu, et al. [51] present recent results of integrating omni-directional view image analysis and a set of adaptive back propagation networks to understand the outdoor road scene by a mobile robot.

To navigate and recognize where it is, a mobile robot must be able to identify its current location. The more the robot knows about its environment, the more efficiently it can operate [52-55]. Grudic and Lawrence [56] used a nonparametric learning algorithm to build a robust mapping between an image obtained from a mobile robot's on-board camera, and the robot's current position. It used the learning data obtained from these raw pixel values to automatically choose a structure for the mapping without human intervention, or any prior assumptions about what type of image features should be used.

In order to localize itself, a mobile robot tries to match its sensory information at any instant against a prior environment model or map [57]. A probabilistic map can be regarded as a model that stores at each robot configuration the probability density function of the sensor readings at that robot configuration. Vlassis, et al. [58] described a novel sensor model and a method for maintaining a probabilistic map in cases of dynamic environments.

2.3.4 Use of neural integrated fuzzy controller

A neural integrated fuzzy controller (NiF-T) that integrates the fuzzy logic representation of human knowledge with the learning capability of neural networks has been developed for nonlinear dynamic control problems. Ng and Trivedi [59], Daxwanger and Schmidt [60] presented their neuro-fuzzy approach to visual guidance of a mobile robot vehicle.

2.3.5 Map-based navigation

While roadmaps has been used for robot path planning in a static structured environment, Piggio and Zaccaria [61] had combine the use of dynamic analogical representation of the environment with a road map extraction method to guide the robot navigation and to classify the different regions of space in which the robot moves in order to use this approach in some real situations. Flexibility maps had been proposed based on the link between the static and the kinematics theory and landmark based navigation, the robot motion defined by flexibility maps are shown to coincide with the equipotential lines from potential functions, these potential functions can be straightforwardly derived from the definition of the map, the map is used to derive collision- free robot motion that pass through particular checkpoints [62].

2.3.6 Biological navigation

Biological navigation also is being tried; Franz and Mallot [63] showed that biomimetic systems provide a real world test of biological navigation behaviors besides making new navigation mechanisms available for indoor robots where simpler insect navigation behaviors have been implemented successfully.

Smith, et al. [64] conducted research on biologically-inspired approaches to the development of intelligent adaptive systems. Quinn, et al. [65] reported a recent work in using artificial evolution in the design of neural network controllers for small, homogeneous teams of mobile autonomous robots. The robots used were only equipped with infrared sensors and performed a formation movement task from random starting positions. They described a successful evolved team where the robots achieved their task by adopting and maintaining a functionally distinct roles. Smith and Husbands [66] proposed a model for visual homing, and extended it to an algorithm capable of autonomous exploration and navigation through large-scale environments, the environments uses a waypoint selection during the construction of multi-leg routes.

2.3.7 New methods proposed

Some new methods are proposed based on recent web technologies such as browsers, Java language and socket communication. They allow users to connect to robots through a web server, using their hand-held computers, and to monitor and control the robots via various input devices. Ohwada, et al. [67] described a web-based method for communication with and control of heterogeneous robots in a unified manner, including mobile robots and vision sensors.

2.3.8 Navigation in unstructured environment

Some research has been conducted regarding robotics in unstructured environment. Martínez and Torras [68] presented visual procedures especially tailored to the constraints and requirements of a legged robot that works with an un-calibrated camera, with pan and zoom, freely moving towards a stationary target in an unstructured environment that may contain independently-moving objects. Kurazume and Hirose [69] proposed a new method called Cooperative Positioning System (CPS). The main concept of CPS is to divide the robots into two groups, A and B, where group A remains stationary and acts as a landmark while group B moves. Then group B stops and acts as a landmark for group A. This process is repeated until the target position is reached. Their application was a floor-cleaning robot system. Torras [70] reviewed neural learning techniques for making robots well adapted to their surroundings. Yahja, et al. [71] propose an on-line path planner for outdoor mobile robots using a framed-quadtrees data structure and an optimal algorithm to incrementally re-plan optimal paths. They showed that the use of framed-quadtrees leads to paths that are shorter and more direct compared to the other representations; however, their results indicate that starting with partial information is better than starting with no information at all. Baratoff, et al. [72] designed a space-variant image transformation, called the polar sector map, which is ideally suited to the navigational tasks. Yu, et al. [73] presented a hybrid evolutionary motion planning simulation system for mobile robots operating in unstructured environments, based on a new obstacle representation method named cross-line, a follow boundary repair approach, and a hybrid evolutionary motion planning algorithm. Yan, et al. [74] presents an attempt to devise and develop a domain-independent reasoning system scheme for handling dynamic threats, and uses the scheme for automated route planning of defense vehicles in an unstructured environment.

Computer vision and image sequence techniques were proposed for obstacle detection and avoidance for autonomous land vehicles that can navigate in an outdoor road environment. The object shape boundary is first extracted from the image, after the translation from the vehicle location in the current cycle to that in the next cycle is estimated, the position of the object shape in the image of the next cycle is predicted, then it is matched with the extracted shape of the object in the image of the next cycle to decide whether the object is an obstacle [75]. Jarvis [76] report some preliminary work regarding an autonomous outdoor robotic vehicle navigation using flux gate compass, DGPS and range sensing, and distance transform based path planning. An adaptive navigation method suited for the complex natural environments had been proposed based on a multi-purpose perception system that manages different terrain representations, the method focuses on the functions that deal with the navigation planning and the robot self-localization which have been integrated within the robot control system [77]. Krishna and Kalra [78] proposed incorporating cognition and remembrance capabilities in a sensor-based real-time navigation algorithm, they stated that these features enhance the robots performance by providing a memory-based reasoning whereby the robot’s forthcoming decisions are also affected by its previous experiences during the navigation, which is apart from the current range inputs, the suggested robot navigates in a concave maze-like unstructured altered environment which has been modeled by classifying temporal sequences of special sensory patterns. A fuzzy classification scheme coupled to Kohonen's self-organizing map and fuzzy network [78]. Marco et al. [79] developed a hybrid controller for semi-autonomous and autonomous underwater vehicles in which the missions imply multiple task robot behavior. They proposed the use of Prolog as a computer language for the specification of the discrete event system (DES) aspects of the mission control, and made the connections between a Prolog specification and the more common Petri Net graphical representation of a DES.

2.4 Controllers for mobile robots autonomous navigation

Robots and robots manipulators have complex nonlinear dynamics that make their accurate and robust control difficult. On the other hand, they fall in the class of Lagrangian dynamical systems, so that they have several extremely nice physical properties that make their control straight forwarded [80].

Different controllers had been developed for the motion of robot manipulators, however, not until recently where there has been an interest in moving the robot itself, not only its manipulators.

Shim and Sung [81] proposed a WMR asymptotic control with driftless constraints based on empirical practice using the WMR kinematic equations. They showed that with the appropriate selection of the control parameters, the numerical performance of the asymptotic control could be effective. The trajectory control of a wheeled inverse pendulum type robot had been discussed by Yun-Su and Yuta [82], their control algorithm consists of balance and velocity control, steering control, and straight line tracking control for navigation in a real indoor environments.

Rajagopalan and Barakat [83] developed a computed torque control scheme for Cartesian velocity control of WMRs. Their control structure can be used to control any mobile robot if its inverse dynamic model exists. A discontinuous stabilizing controller for WMRs with nonholonomic constraints where the state of the robot asymptotically converges to the target configuration with a smooth trajectory was presented by Zhang and Hirschorn [84]. A path tracking problem was formulated by Koh and Cho [85] for a mobile robot to follow a virtual target vehicle that is moved exactly along the path with specified velocity. The driving velocity control law was designed based on bang-bang control considering the acceleration bounds of driving wheels and the robot dynamic constraints in order to avoid wheel slippage or mechanical damage during navigation. Zhang, et al. [86] employed a dynamic modeling to design a tracking controller for a differentially steered mobile robot that is subject to wheel slip and external loads.

A sliding mode control was used to develop a trajectory tracking control in the presence of bounded uncertainties [87]. A solution for the trajectory tracking problem for a WMR in the presence of disturbances that violate the nonholonomic constraint is proposed later by the same authors based on discrete-time sliding mode control [88-89].

An electromagnetic approach for path guidance of a mobile-robot-based automatic transport service system with a PD control algorithm was investigated by Wu, et al. [90]. Jiang, et al. [91] developed a model-based control design strategy that deals with global stabilization and global tracking control for the kinematic model with a nonholonomic WMR in the presence of input saturations. An adaptive robust controller was proposed for the global tracking problem for the dynamic of the non-holonomic systems with unknown dynamics [92]. However, real time adaptive controls are not common in practical applications due partly to the stability problems associated with them [93].

A fuzzy logic controller had been tried for WMRs navigation. Montaner and Ramirez-Serrano [94] developed a fuzzy logic controller that can deal with the sensors inputs uncertainty and ambiguity for direction and velocity maneuvers. A locomotion control structure was developed based on the integration of an adaptive fuzzy-net torque controller with a kinematic controller to deal with unstructured unmodeled robot dynamics for a non-holonomic mobile robot cart [95]. Toda, et al. [96] employed a sonar-based mapping of crop rows and fuzzy logic control-based steering for the navigation of a WMR in an agricultural environment. They constructed a crop row map from the sonar readings and transferred it to the fuzzy logic control system, which steers the robot along the crop row. A local guidance control method for WMR using fuzzy logic for guidance, obstacle avoidance and docking of a WMR was proposed by Vázquez and Garcia [97], the method provide a smooth but not necessary optimal solution.

2.5 Defense Robot

Defense robots which may be though as a science fiction can be the future of the army. The argument that defense robots can reduce the risks to soldiers is difficult to counter. While the hazards of war are increasing by developing nuclear, biological, and chemical weapons, autonomous robots that could fight in these conditions, or aid in detection and decontamination, must be seriously considered. On the other hand, there are very risky uses of autonomous weapons that will be subject to open debate. The potential for errors and failure of an autonomous weapon must be seriously considered [98].

The defense look to robotics to increase the performance and operational benefit, reduce the risk to personnel, and improve the efficiencies through the saving of resources, manpower, or system requirements [99].

The major agencies that fund research in robotics and smart weapons development are the Defense Advanced Research Projects Agency (DARPA) and the research offices of the Navy, Army, and Air Force. The Pentagon spends more than $300 million a year on autonomous weapons development. Table 2.1 tracked the use of automated machines in defense since the last years of the First World War [98].

Time Period	**Event**
1918-1919	The U.S. Navy and Sperry Gyroscope Co. perform the first experiments on guided missiles during the last years of the First World War. (Gurney, G.: Rocket and Missile Technology; Franklin Watts, New York, 1964 as cited in [98]).
1945	War-weary Germans learn of the development of the V-2, the first operational ballistic missile, in their movie newsreels (Jan. 21).
1950's	AIM-7 Sparrow (predecessor to the contemporary AIM-120A AMRAAM) is developed for use by the U.S. Air Force. The 1950's Sparrow model employs a target illumination technique, whereby energy reflected from the target is received by the missile and used to locate the target. This marks the beginning of "fire-and-forget" weapons systems.
1961-1975	During the Vietnam War the AQM-34L reconnaissance drone (Firebee), is used to perform day and night reconnaissance, high-altitude surveillance, electronic intelligence, and distribution of propaganda leaflets [98].
1961-1975	Smart bombs are employed for the first time during the Vietnam War.

	These are bombs that are laser guided.
1973	During the Yom Kippur War, Israel successfully employs harassment drones known as remotely piloted vehicles (RPVs) to confuse enemy air defenses.
1982	RPVs are used to ferret out Syrian SA-6 missiles in Lebanon's Valley.
1983	The $600-million Strategic Computing Program (SCP) creates three applications for pulling the technology-generation process by creating carefully selected technology interactions with challenging defense applications: an autonomous land vehicle, a pilot's associate, and a battle management system. These developments explicitly connect the three armed services to further AI developments.
1983	Tomahawk cruise missiles deployed by the U.S. Navy for first time. Tomahawk autonomously finds its way to the target by using terrain contour matching (TERCOM) system, in which actual terrain beneath missile's flight path is compared to a stored map containing information about the terrain. The missile, in a sense, navigates itself towards the target.
March 1983	Ronald Reagan proposes implementing the Strategic Defense Initiative (SDI), a program intending to create a "leak proof shield" over the United States protecting it from ICBM's. The idea behind the system was to use sensors to detect missile launches and then to intercept the warheads while they were still in outer space.
July 1988	With the aid of the Aegis Spy-1 radar system (a radar which

	automatically identifies friend/foe status of oncoming aircraft), the U.S.S. Vincennes mistakenly identifies an Iranian civilian airbus, Flight 655, as an attacking Iranian F-14. The ship opens fire on the airplane, killed 290 passengers. The radar detection system is later criticized as being too complicated for use in combat situations.
February 1991	Smart bombs are used extensively to selectively destroy enemy targets in Kuwait and Iraq during the Gulf War. These bombs are praised for their effectiveness and precision. Videos of the bombs entering their targeted buildings are shown on public media. Also used in the Gulf War is the Navy's Tomahawk missile, as well as the Air Force's LANTIRN targeting system, which uses infrared detectors to locate targets in the dark or in bad weather.
September 1991	AIM-120 Advanced Medium Range Air-to-Air Missile (AMRAAM) is deployed by the U.S. Navy and Air Force. This missile is the successor to the AIM-7 Sparrow. AIM-120 guides itself to its target by using its own radar system. It marks a great advancement in the notion of "fire-and-forget".
January 1992	Testing begins on Autonomous Guidance for Conventional Weapons (AGCW) to be used in glide bombs. The AGCW uses an IR seeker to search for a generic type of target and automatically selects an aim point to fire at the target.

Table 2.1: The history for autonomous robots in defense applications for the years 1918-1992 [98].

So far, robot development has largely been restricted to such tasks as reconnaissance or bomb disposal and information gathering applications which are usually teleoperator controlled [78]. Remote-controlled surveillance aircraft are already in service, and land based machines that rely on remote control rather than true artificial intelligence are doing well on proving grounds[5].

Defense robots control available ranges from direct operator control where human operators control the machines while they are having some physical contact with the machine, to the teleoperator control where the unit is still controlled by a human, but at a distance. The next level is those under preprogrammed operation. These units are obviously programmed to perform a certain task, and simply follow a predetermined set of orders on their own on the battlefield [98]. Approaching truly autonomous technology is structured control. Units in this category work in conjunction with artificial vision or sensor systems to respond in a rudimentary way to environmental stimuli [98].

2.5.1 Remote-controlled robots

Teleoperator control or remote control operation robots been developed for various defense applications including ground vehicles, undersea vehicles, and air vehicles. The Space and Naval Warfare Systems Center, San Diego (SSC San Diego) [100] have been involved in various aspects of these robotics, one of their Unmanned Ground Vehicles is Man Portable Robotic System (MPRS/URBOT). The MPRS program goal is to develop lightweight, man-portable mobile robots for operation in urban environments. The technical strategy calls for optimizing a realistic and robust solution to an appropriate set of articulated user requirements, using predominantly off-the-shelf components. The capabilities and sophistication of these

[5] B. Pokorny. (1987, March 9). Creating the ideal soldier; U.S. seeks a PFC. Robot. The Record, Section: News. [Online]. pp. A01. Available: http://www-cse.stanford.edu/classes/cs201/Projects/autonomous-weapons/articles/robot-soldiers.txt

systems will expand with time as new technologies become available from various sources. The initial MPRS system was implemented under a reflexive teleoperated control interface (developed on ROBART III), supported by ultrasonic and near-infrared collision avoidance sensors. The ROBART series of research prototypes has served the US Navy in developing the component technologies needed in support of the MDARS program. While ROBART I could only detect a potential intruder, ROBART II could both detect and assess, thereby increasing its sensitivity with a corresponding reduction in nuisance alarms. As the third-generation prototype, ROBART III is specifically intended to demonstrate the feasibility of automated response, using (for purposes of illustration only) a pneumatically powered six-barrel Gatling-style weapon that fires simulated tranquilizer darts or rubber bullets. The ROBART series is shown in Fig 4.2.

Under the same category, the SSC San Diego [100] has been involved in developing an unmanned air vehicle and unmanned undersea vehicles. One of the unmanned air vehicles that had been developed is the Air Mobile Ground Security and Surveillance System (AMGSSS).

AMGSSS development started in 1992 to provide a rapidly deployable, extended-range surveillance capability for force protection and tactical security. It consisted of three air-mobile remote sensing platforms and a control station. Later on, the program took on expanded to include the sensor package that may operate as stand-alone units or integrated with Cypher air-mobile vehicles. New mission areas include: support to counter drug and border patrol operations, signal/communications relays, detection and assessment of barriers, remote assessment of suspected contaminated areas, and re-supply of small quantities of critical items [100]. Initial taxi tests for the first robot plane designed specifically to carry weapons into combat have been completed and a maiden flight is planned for early 2002 as reported by Nando Media[6].

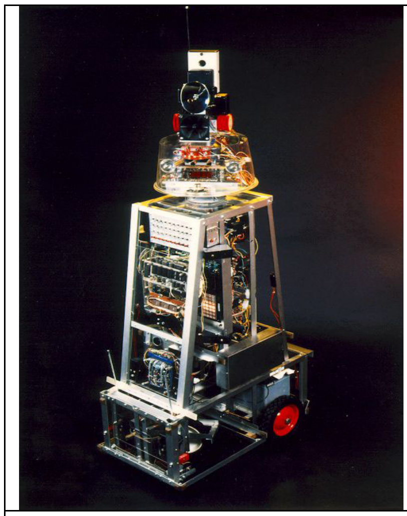
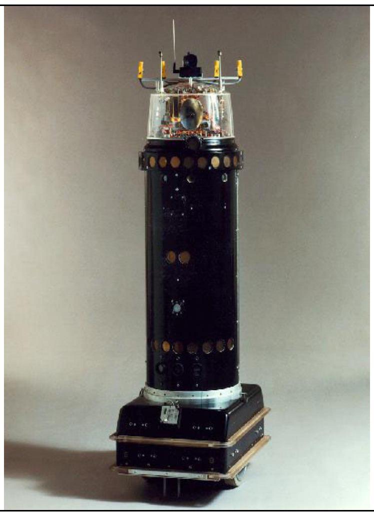
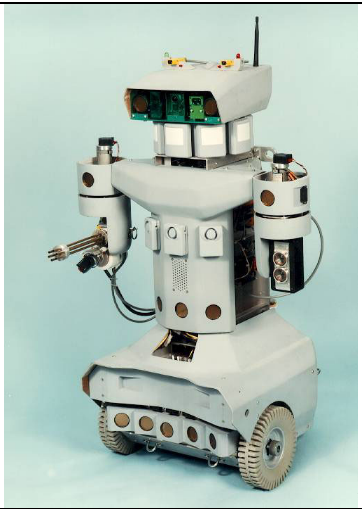
		
ROBART I (1980-1982)	ROBART II (1982-1992)	ROBART III (1992-)

Figure 2.4: The ROBART series [100].

Aerial and underwater unmanned vehicles had been used also for defense purposes, usually with some kinds of remote control [101]. The Distributed Surveillance Sensor Network (DSSN) program falls under the Unmanned Undersea Vehicles, DSSN purpose is to investigate the applicability of small, inexpensive undersea vehicles to surveillance applications and submarine connectivity. DSSN is based on the concept of a fleet of autonomous undersea vehicles which gather surveillance data and communicate acoustically. The docking station is self powered and is not connected to shore or ship by communications cable. The data is retrieved by means of a Flying Plug, which is a remotely controlled vehicle guided to the docking station by means of a high-bandwidth channel by which the data is recovered and instructions are downloaded to be disseminated to the surveillance fleet. Figure 2.5 shows unmanned air vehicle and unmanned undersea vehicle [100].

[6] Nando Times. (2001, Dec.). Health & Science: U.S. military tests new robot attack plane. *Nando Times*. [Online]. Available: http://www.nandotimes.com/healthscience/story/206710p-1995152c.html

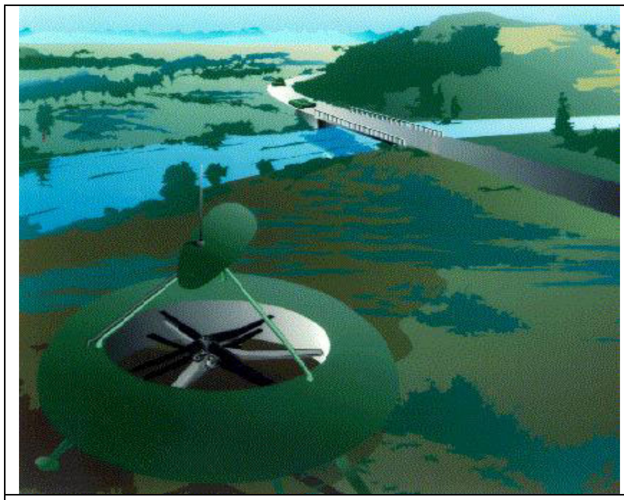
	
Unmanned air vehicle	Unmanned undersea vehicle

Figure 2.5: Unmanned air vehicle and unmanned undersea vehicle [100].

Army Research Labs with NIST support demonstrated defense robot vehicles, called Experimental Unmanned Vehicles (XUV) that developed at a cost of approximately $50 million over the past four years. It used technology from leading robotics laboratories in the US and Germany. XUV performed autonomous scout missions in difficult off-road terrain. Running with general goal points and mission profiles given by Army scouts. XUV navigated through woods and fields to find and report enemy targets. During the demonstration, the XUVs drove autonomously over difficult terrain including dirt roads, trails, tall grass, weeds, brush, and woods. Using on board sensors, the XUVs were able to detect and avoid both positive and negative obstacles. The Demo III XUVs have repeatedly navigated kilometers of difficult off-road terrain with only high-level mission commands provided by an operator from a remote location. Figure 2.6 shows the XUV in action at Ft. Indiantown Gap [102].

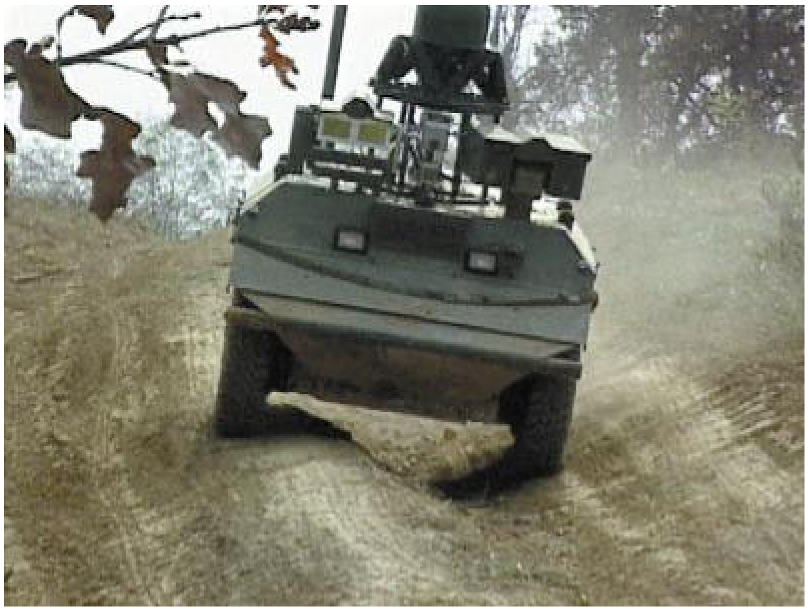


Figure 2.6: Experimental Unmanned Vehicle in action at Ft. Indiantown Gap. Photo courtesy of the Army Research Labs [102].

A small intruder robot has been developed for recon, delivery, and for drooping some bombs, the robot can work over one mile range and has real-time color video [103]. Laird, et al. [104] presented the Man Portable Robotic System (MPRS) that had been developed to validate the concept of employing small robots to conduct tunnel, sewer, and bunker reconnaissance in urban combat. MPRS had been developed in order to reduce the need for soldiers to descend below the surface into a hostile environment to search for explosives. This system is controlled by an operator control unit.

A reporter robot is being built at the Massachusetts Institute of Technology Media Lab. The robot news hound is the Afghan Explorer that can improve the information reported during wars. The first one should be ready in two months. It uses technology such as the internet, satellite communications, and global positioning satellite systems to send a robot into enemy territory and controlled remotely using a personal computer and a web browser. A web camera would provide two-way video conferencing[7].

2.5.2 Autonomous Robots

As may be expected, different designs of autonomous robots or fully automated vehicles have been researched, some has been developed; however, these had not been finalized or proved practically. Pokorny[8] stated that fully automated vehicles are at least a decade away, scientists say: true battlefield robots need far more computing power, better sensors, and stronger communication links than are available today.

Pransky [105] describes two types of mobile robots designed for the US military. The mobile detection assessment response system (MDARS) and the spiral track autonomous robot (STAR). MDARS is an automated robotic security and inventory system capable of patrolling interior and exterior warehouses and storage sites for the Department of Defense. MDARS-I utilizes the Cybermotion K2A Navmaster mobility base developed by Cybermotion, Inc., Salem, Virginia. While MDARS-E employs a diesel-powered, four wheel, hydrostatic-drive vehicle about the size of a golf cart, built by Robotic Systems Technology (RST), Westminster, Maryland. Both MDARS-I and MDARS-E can operate in a supervised-autonomous mode and are equipped with autonomous navigation and collision avoidance, intruder detection capabilities, and a product assessment system. Most of the real-world evaluation for MDARS-I has taken place at a warehouse test-site at Camp Elliott, San Diego, California. The exterior vehicle recently demonstrated autonomous path execution under differential GPS control at the RST development facility in Maryland.

[7] K. Maney. (2002, March). Roving reporter on battlefields could be a robot. *USA Today*. [Online]. Available: http://www.usatoday.com/money/columns/maney/2002-03-20-maney.htm
[8] B. Pokorny. (1987, March 9). Creating the ideal soldier; U.S. seeks a PFC. Robot. The Record, Section: News. [Online]. pp. A01. Available: http://www-cse.stanford.edu/classes/cs201/Projects/autonomous-weapons/articles/robot-soldiers.txt

STAR is a multi-terrain military vehicle used to reduce risk to military personnel and equipment during surveillance, reconnaissance, and infiltration missions. The STAR is capable of being operated autonomously by preprogramming a start point and an end-point or remotely using a wireless data link and control system software resident in a laptop computer. For both remote and autonomous operations, ultrasonic sensors are mounted around the external perimeter of the robot to provide collision avoidance capabilities. All power is placed on board to allow for missions involving distant travel. The electronics enclosure contains motion controller and a sensor card. The STAR is also equipped with a complete on-board electronic control system and wireless data/video links for high-level decision making, motion control, autonomous path planning, and execution. The control system and software provide the STAR with enough intelligence to execute decisions which are typically required of their human counterparts. STAR has progressed in multiple phases, and it is currently in progress [105-106].

2.5.3 Other research trends in defense robot

Scientists created tiny solar-powered robots using scorpions as a design model as part of biomimetics. biomimetics used the concepts of animal neural mechanisms and biological processes as blueprints for building better machines. These robots are endowed with computers, sensors, and transmitters. The goal of building solar-powered robots is that one day these robots will scurry unaided across rugged terrain while performing a variety of tasks, including defense reconnaissance[9]. Georgia Tech Research Institute developed a tiny insect-like robots that are able to fly unnoticed through a building and attach themselves to a wall, in order to be used in some spying activities [107]. Another tiny insect-like robot is being developed by two mechanical

[9] K. Hearn. (2001, March). Scorpion is model for new military robot. *Spark notes*. [Online]. Available: http://cgi.sparknotes.com/newsfeed.mpl?nid=8051

engineering professors from Vanderbilt University. The robot is about one-third of the size of a credit card, and expected to have applications for defense and intelligence-gathering missions[10].

Rosenblum and Gothard [108] developed a scene understanding analyzer and a reactive planner to enable the robot that is working in defense applications to avoid hazards and run over traversable and hugs features. The system also works in a wide range of lighting conditions. The scene understanding system developed based on a multi-sensor system that uses an "operator-trained" rule base to analyze the pixel level attributes across the set of diverse phenomenology imaging sensors, each of these sensors is registered to range of information so it will be possible to know what features and where are they in the environment.

When having more than one robot working together, coordination and cooperation between these robots is essential. Singh and Thayer [109] surveyed the state-of-the art in autonomous multi-robot work systems and investigated the relative strengths and weaknesses of each approach with respect to defense applications. They found that current multi-robot coordination and cooperation architectures alone are not fully sufficient to handle the scale and scope of future combat. Hence, new methods are needed. Durfee, et al. [110] suggested inserting an agent technology between the operator and the unmanned ground vehicles that being used in the defense applications.

Blitch [111] discussed the application of robotic systems to Urban Search And Rescue (USAR) Activities and the development of a knowledge-based system for efficient management of automated search assets. He concluded that robotics platforms and decision support tools can improve the efficiency for the USAR community which is important for defense and rescue applications.

[10] Robot books.com. Developing Robotic Insects. *Robot News* Nashville, Tenn. Vanderbilt University Engineers. [Online]. Available: http://www.robotbooks.com/robot-insects.htm

Chapter 3: Robot Modeling

3.1 Introduction

de Wit, et al. [112] define the WMR as "wheeled vehicle which is capable of an autonomous motion (without external human driver) because it is equipped, for its motion, with actuators that are driven by an embarked computer".

This chapter will analyze the kinematic and dynamic modeling and the structural properties for WMRs. A general kinematic and dynamic model for all WMRs will be presented.

To provide a focus for the presented model, a particular configuration will be addressed: the kinematic and dynamic model for Bearcat III robot will be derived using the Newton-Euler formulations. Bearcat III is a WMR with two driven wheels and a caster wheel. The driven wheels are of a fixed-wheel type. Bearcat III is shown in Fig. 3.1.

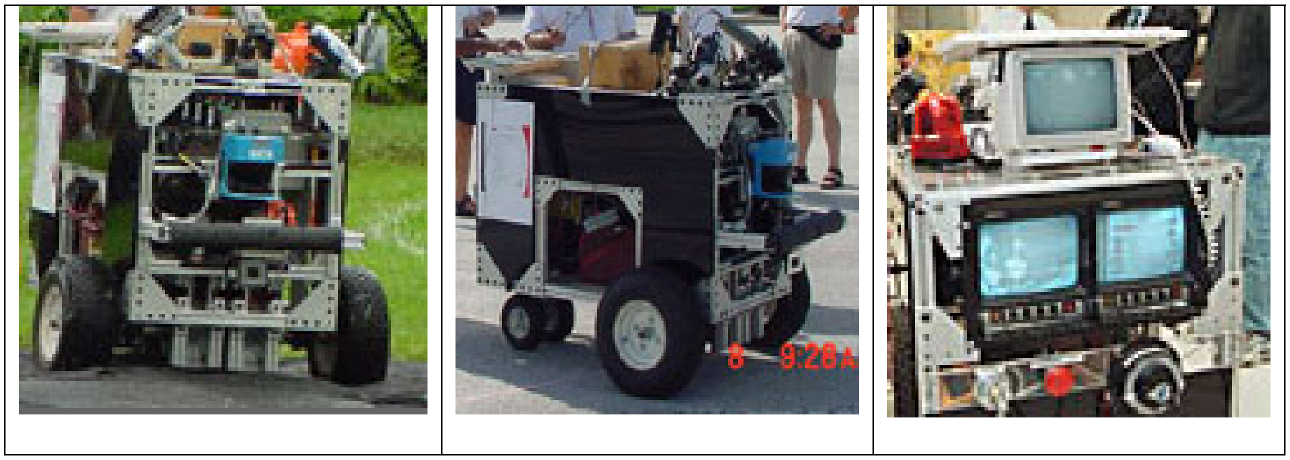


Figure 3.1: Bearcat III.

3.2 Robot kinematics

3.2.1 Robot description

Before deriving the robot kinematics, some assumptions need to be stated [112]:

1. The robot is made up of a rigid cart equipped with non-deformable wheels.
2. The wheels are moving on a horizontal plane and rotating about its horizontal axes.

3. The contact between the wheels and the ground is reduced to a single point of the plane.
4. No slip occurs in the orthogonal direction of rolling (non-slipping).
5. No translational slip occurs between the wheels and the floor (pure rolling).

The position of the WMR in the plane is shown in Fig. 3.2, where an arbitrary inertial base frame b is fixed in the plane of motion, and a frame m is attached to the WMR.

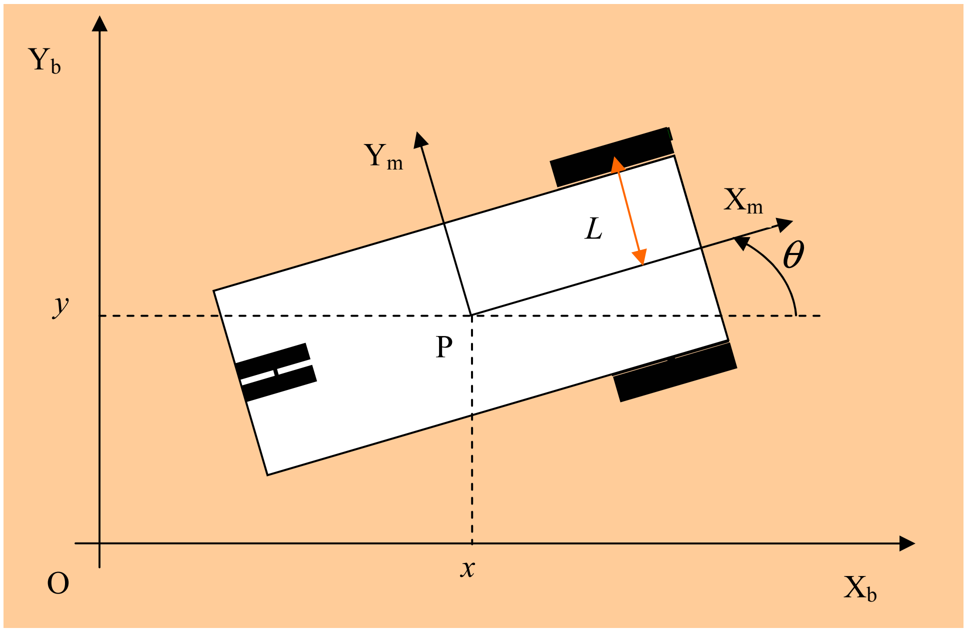


Figure 3.2 WMR position coordinates.

The robot posture can be described in terms of the origin P of the moving frame coordinates and the orientation angle θ, with respect to the base frame with origin O. Hence, the robot posture is given by [112]:

$$\xi = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} \tag{3.1}$$

The rotation matrix expressing the orientation of the base frame with respect to the moving frame is [112]:

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

The two wheels in front of the robot (driving wheels) are of the fixed type that can only rotate about its horizontal axle, whose orientation is fixed with respect to the robot (β is constant in Fig 3.3), the position of each wheel can be described by four constants: α, β, l, r and a time varying angle φ as shown in Fig. 3.3. The components of the velocity of the contact point are computed and the following constrains can be deduced [112]:

- On the wheel plane:

$$\left(-\sin(\alpha+\beta) \quad \cos(\alpha+\beta) \quad l\cos\beta\right)R(\theta)\dot{\xi} + r\dot{\varphi} = 0; \tag{3.3}$$

- Orthogonal to the wheel plane:

$$\left(\cos(\alpha+\beta) \quad \sin(\alpha+\beta) \quad l\sin\beta\right)R(\theta)\dot{\xi} = 0; \tag{3.4}$$

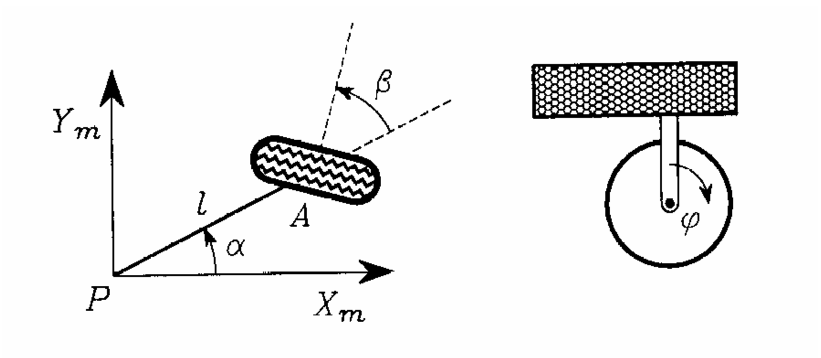


Figure 3.3: Fixed wheel structure [112].

In Bearcat III $\alpha_f = \beta_f = 45^0$, l_f=19.8 in, r_f (the radius of the fixed wheel) = 9 in, and the two wheels have the same radius.

The third wheel in Bearcat II is a castor wheel which is orientable with respect to the robot, but the orientation of the wheel plane is about a vertical axle that does not pass through the center of the wheel. As shown in Fig. 3.4, the position of this wheel is represented by four constants α, l, r, d and its motion is described by two time-varying angles $\beta(t)$ and $\varphi(t)$. Hence, the angle β is varying in this wheel compared to the fixed wheels in front of the robot. This wheel constrains are as follows [112]:

- On the wheel plane:

$$\left(-\sin(\alpha+\beta) \quad \cos(\alpha+\beta) \quad l\cos\beta\right)R(\theta)\dot{\xi} + r\dot{\varphi} = 0; \tag{3.5}$$

- Orthogonal to the wheel plane:

$$\left(\cos(\alpha+\beta) \quad \sin(\alpha+\beta) \quad d + l\sin\beta\right)R(\theta)\dot{\xi} + d\dot{\beta} = 0; \tag{3.6}$$

As mentioned before this wheel is not powered and just moving freely, hence, it did not affect the kinematics of the robot.

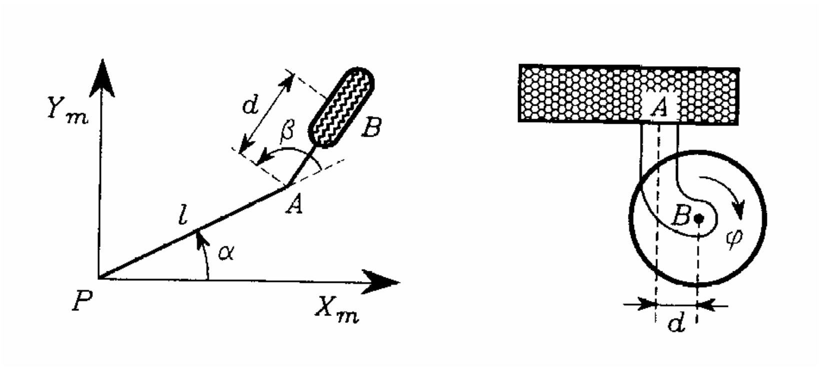


Figure 3.4: Castor wheel structure [112].

In Bearcat III α_c=0, β_c is variable, l_c=15.25 in, and r_c (the radius of the castor wheel) = 4.5 in.

The set of postures, orientation, and rotation coordinates ξ, β, φ is termed the set of configuration coordinates in the sequel. The total number of configuration coordinates is

$N_f+2N_s+2N_c+N_{sw}+3$, where: N_f is the number of fixed wheel in the robot, N_s is the number of steering wheels, N_c is the number of castor wheels, and N_{sw} is the number of Swedish wheels in the robot. For Bearcat III the total number of configuration coordinates is seven.

By adopting the subscript 'f' for fixed wheel, 's' for steering wheels, 'c' for castor wheels, and 'sw' for Swedish wheels. The constrains on robot mobility equations (Eq. (3.3)-(3.6)) can be written in more general form [112]:

$$J_1(\beta_s,\beta_c)R(\varphi)\dot{\xi}+J_2\dot{\varphi}=0 \qquad (3.7)$$
$$C_1(\beta_s,\beta_c)R(\varphi)\dot{\xi}+C_2\dot{\beta}_c=0$$
$$J_1(\beta_s,\beta_c)=\begin{pmatrix} J_{1f} \\ J_{1s}(\beta_s) \\ J_{1c}(\beta_c) \\ J_{1sw} \end{pmatrix}, C_1(\beta_s,\beta_c)=\begin{pmatrix} C_{1f} \\ C_{1s}(\beta_s) \\ C_{1c}(\beta_c) \end{pmatrix}, C_2=\begin{pmatrix} 0 \\ 0 \\ C_{2c} \end{pmatrix}$$

where:

β_s,β_c: are the orientation angles of the steering wheels and the castor wheels respectively.

$J_{1f},J_{1s},J_{1c},J_{1sw}$: are matrices of size $N_f\times 3$, $N_s\times 3$, $N_c\times 3$, and $N_{sw}\times 3$ respectively, their forms derived directly from the mobility constrains in Eq. (3.3)-(3.6).

J_2: is a constant $(N\times N)$ matrix, whose diagonal entries are the radii of the wheels, except for the radii of the Swedish wheel, those need to be multiplied by cosine the angle of the contact point.

C_{1f},C_{1s},C_{1c}: are matrices of size $N_f\times 3, N_s\times 3$, and $N_c\times 3$ respectively, their forms derived directly from the non-slipping constrains in Eq. (3.4) and (3.6) respectively.

Each robot structure can be characterized by it is degree of mobility and steerability. The degree of mobility, δ_m, of a mobile robot can de defined as follows [112]:

$\delta_m = \dim(N(C_1^*(\beta_s))) = 3 - rank(C_1^*(\beta_s))$, $1 \le \delta_m \le 3$ (3.8)

The degree of steerability, δ_s , of a mobile robot is the number of steering wheels that can be oriented independently in order to steer the robot, it can de defined as follows [112]:

$\delta_s = rank(C_{1s}(\beta_s))$, $0 \le \delta_s \le 2$ (3.9)

From the conditions of the inequality of Eq. (3.8) and (3.9), there are only five nonsingular robot structures that are of practical interest, these are shown in Table 3.1.

δ_m	3	2	2	1	1
δ_s	0	0	1	1	2

Table 3.1: Degree of mobility and degree of steerability combinations for possible wheeled mobile robots [112].

Each robot structure is designated by using a denomination of the form "Type (δ_m, δ_s) robot". Based on the pre-mentioned equations, Bearcat III is Type (2, 0) robot which is typically referred to as the unicycle robot, The mobility of Bearcat III is restricted in a sense that, for any admissible trajectory $\xi(t)$, the velocity $\dot{\xi}(t)$ is constrained to belong to the 2-dimentional distribution spanned by the vector fields $R^T(\theta)s_1$ and $R^T(\theta)s_2$ where s_1 and s_2 are two constant vectors spanning $N(C_{1f})$ [112].

Returning back to Fig. 3.2, the characteristic constants for type (2, 0) robot are specified in Table 3.2.

Wheels	α	β	l
1_f (right fixed wheel)	0	0	L
2_f (left fixed wheel)	π	0	L

3_c (The rear castor wheel)	$3\pi/2$	-	L

Table 3.2: The characteristic constants of Type (2, 0) robot [112].

Substituting the values of the characteristics constants from Table 3.2 into Eq. (3.7), the constraints matrices J_1, J_2, C_1, C_2 for type (2, 0) robot are as follows [112]:

$$J_1 = \begin{pmatrix} J_{1f} \\ J_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} 0 & 1 & L \\ 0 & -1 & L \\ \cos\beta_{c3} & \sin\beta_{c3} & L\cos\beta_{c3} \end{pmatrix} \tag{3.10}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} C_{1f} \\ C_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ \sin\beta_{c3} & -\cos\beta_{c3} & d + L\sin\beta_{c3} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 \\ C_{2c} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix}$$

For Bearcat III L= 13 in, substituting the values of L, r, and d for Bearcat III, the constraints matrices J_1, J_2, C_1, C_2 for Bearcat III are:

$$J_1 = \begin{pmatrix} J_{1f} \\ J_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 13 \\ 0 & -1 & 13 \\ \cos\beta_{c3} & \sin\beta_{c3} & 13\cos\beta_{c3} \end{pmatrix} \tag{3.11}$$

$$J_2 = \begin{pmatrix} 9 & 0 & 0 \\ 0 & 9 & 0 \\ 0 & 0 & 9 \end{pmatrix}$$

$$C_1 = \begin{pmatrix} C_{1f} \\ C_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ \sin\beta_{c3} & -\cos\beta_{c3} & 3.25 + 13\sin\beta_{c3} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 \\ C_{2c} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 3.25 \end{pmatrix}$$

3.2.2 Posture kinematic model

de Wit, et al. [112] define the posture kinematic model as "the simplest state space model able to give a global description of wheeled mobile robots". It can be applied to the five classes of the WMR described in Table 3.1. This model has a particular generic structure which allows understanding the maneuverability properties of the robot.

As shown previously, whatever the type of mobile robot, the velocity is restricted as follows [112]:

$$\dot{\xi}(t) \in \Delta_c = span\{col(R^T(\theta)\Sigma(\beta_s))\} \quad \forall t \tag{3.12}$$

Where the columns of the matrix $\Sigma(\beta_s)$ form a basis of $N(C_1^*(\beta_s))$ as follows [112]:

$$N(C_1^*(\beta_s)) = span\{col(\Sigma(\beta_s))\} \tag{3.13}$$

Which can be simplified as: for all t, there exists a time-varying vector $\eta(t)$ such that:

$$\dot{\xi}(t) \in \Delta_c = R^T(\theta)\Sigma(\beta_s)\eta \tag{3.14}$$

For type (2, 0) robot where there are no steering wheels ($\beta_s = 0$), hence, Eq. (3.14) can be further simplified into:

$$\dot{\xi}(t) \in \Delta_c = R^T(\theta)\Sigma\eta \tag{3.15}$$

The posture kinematic model for type (2, 0) robot can be described as follows [112]:

$$\Sigma = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad (3.16)$$

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}$$

where:

η_1 : is the robot velocity component along Y_m as shown in Fig. 3.5.

η_2 : is the angular velocity ω.

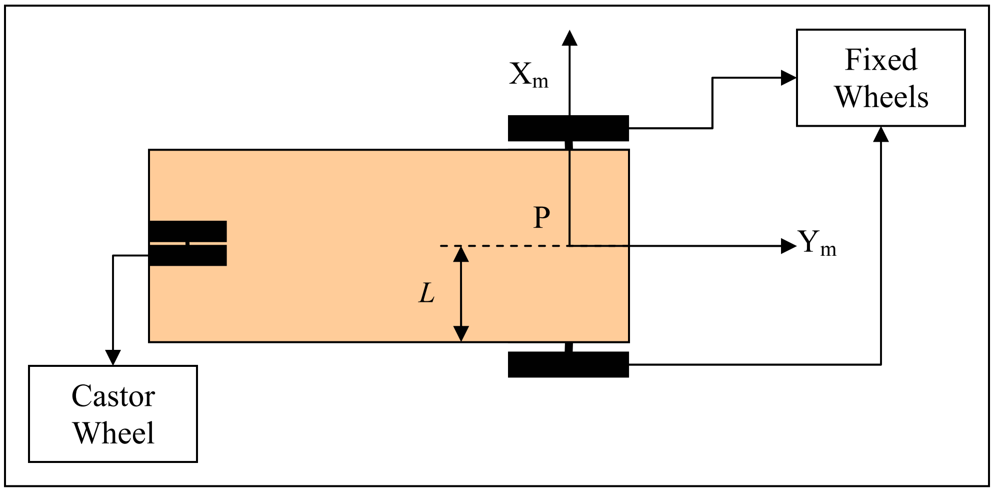


Figure 3.5: The structure of type (2, 0) robot (Bearcat III).

Eq. (3.16) derived based on the assumption that both the fixed driving wheels have the same radius, which is the common case, however, if these two wheels have different radius, the kinematics will be as follows [113]:

$$\dot{x} = v \cdot \sin\theta \qquad (3.17)$$

$$\dot{y} = v \cdot \cos\theta$$

$$\dot{\theta} = \omega$$

where:

$\dot{x}$, $\dot{y}$: is the rate of change of the robot position.

$\dot{\theta}$: is the rate of change of the robot orientation.

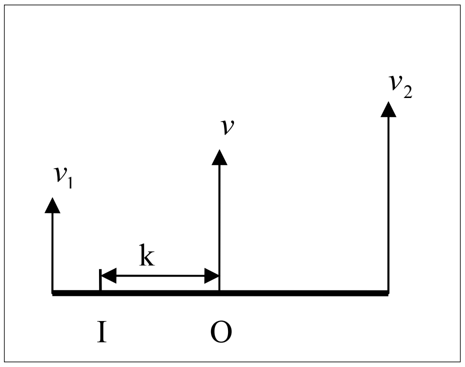


Figure 3.6: The structure of the two wheels.

The linear speed of the first and second wheel can be derived from Fig. 3.6 as shown in the following equation, where I considered to be the center of rotation [113]:

$$v_1 = -(\frac{L}{2} - k) \cdot \omega \tag{3.18}$$

$$v_2 = (\frac{L}{2} + k) \cdot \omega$$

Solving Eq. (3.18) for k and ω [113]:

$$\omega = \frac{v_2 - v_1}{L} \tag{3.19}$$

$$k = \frac{v_1 + v_2}{v_2 - v_1} \cdot \frac{L}{2}$$

From Eq. (3.17), the linear speeds v, v_1, v_2 can be defined as:

$$v = \omega \cdot k \tag{3.20}$$

$$v_1 = \omega_1 \cdot R_1$$

$v_2 = \omega_2 \cdot R_2$

Where subscript ‘1’ refers to the first wheel and subscript ‘2’ refers to the second wheel.

From Eq. (3.18), v can also be defined as [113]:

$$v = \frac{v_1 + v_2}{2} \tag{3.21}$$

Substituting the value of v from Eq. (3.21) and the value of ω from Eq. (3.19) into Eq. (3.20), $\dot{x}, \dot{y},$ and $\dot{\theta}$ can be defined as [113]:

$$\dot{x} = \frac{\omega_1 \cdot R_1 + \omega_2 \cdot R_2}{2} \cdot \sin\theta \tag{3.22}$$

$$\dot{y} = \frac{\omega_1 \cdot R_1 + \omega_2 \cdot R_2}{2} \cdot \cos\theta$$

$$\dot{\theta} = \frac{\omega_2 \cdot R_2 - \omega_1 \cdot R_1}{L}$$

However, in Bearcat III both of the two fixed wheels have the same radius.

The posture kinematic model can be written in the following compact form [112]:

$$\dot{z} = B(z)u \tag{3.23}$$

For type (2, 0) robot, δ_s =0, then:

$z = \xi$

$B(z) = R^T(\theta)\Sigma$

$u = \eta$

The maneuverability of the robot depend on δ_m and δ_s, the sum of these two numbers is called the degree of maneuverability δ_M, however, the maneuverability of the WMR depends not only on δ_M, but also on the way these δ_M degrees of freedom are partitioned into δ_m and δ_s.

In other words two WMR with the same δ_M but with different δ_m are not equivalent on the kinematic basis, as for assigning the position of the instantaneous center of rotation (ICR).

The kinematics developed in this section is irreducible, please refer to [112] for further details regarding irreducibility, controllability, and stability of the posture kinematic model described in this section.

3.2.3 Configuration kinematic model

The configuration kinematic model for WMR will be developed in order to analyze WMR within the framework of the theory of nonholonomic systems.

From the mobility constraints equations (Eq. (3.5)-(3.7)) and including the variables of the castor wheel (refer to Fig. 3.4), $\dot{\beta}_c$ and $\dot{\varphi}$ can be defined as [112]:

$$\dot{\beta}_c = -C_{2c}^{-1} C_{1c}(\beta_c) R(\theta)\dot{\xi} \tag{3.24}$$

$$\dot{\varphi} = -J_2^{-1} J_1(\beta_s, \beta_c) R(\theta)\dot{\xi}$$

Combining these equations with the posture kinematic model (Eq. (3.15) and (3.16)), the state equations for β_c and φ become [112]:

$$\dot{\beta}_c = D(\beta_c)\Sigma(\beta_s)\eta \tag{3.25}$$

$$\dot{\varphi} = E(\beta_s, \beta_c)\Sigma(\beta_c)$$

where:

$$D(\beta_c) = -C_{2c}^{-1} C_{1c}(\beta_c)$$

$$E(\beta_s, \beta_c) = -J_2^{-1} J_1(\beta_s, \beta_c)$$

Defining the vector of configuration coordinates q to include the castor wheel variables as [112]:

$$q = \begin{pmatrix} \xi \\ \beta_s \\ \beta_c \\ \varphi \end{pmatrix} \tag{3.26}$$

From Eq. (3.15), (3.16), and (3.26), the configuration kinematic model $\dot{q}$ can be defined as [112]:

$$\dot{q} = S(q)u \tag{3.27}$$

where:

$$S(q) = \begin{pmatrix} R^T(\theta)\Sigma(\beta_s) & 0 \\ 0 & I \\ D(\beta_c)\Sigma(\beta_s) & 0 \\ E(\beta_s, \beta_c)\Sigma(\beta_s) & 0 \end{pmatrix} \quad u = \begin{pmatrix} \eta \\ \zeta \end{pmatrix}$$

Where ζ is angle related to the Swedish wheel.

Reproducibility of Eq. (3.27) is related to the dimension of the involutive closure of the distribution Δ_1 spanned in local coordinates q by the columns of the matrix $S(q)$ [112]:

$$\Delta_1(q) = span\{col(S(q))\} \tag{3.28}$$

It follows immediately that:

$$\delta_m + N_s = \dim(\Delta_1) \le \dim(inv\Delta_1)) \le \dim(q) = 3 + N + N_c + N_s \tag{3.29}$$

Then the degree of nonholonomy, M, of a mobile robot can be defined as [112]:

$$M = \dim(inv(\Delta_1)) - (\delta_m + N_s) \tag{3.30}$$

de Wit, et al. [112] describe the degree of nonholonomy as it represents the number of velocity constrains that are integrable and therefore cannot be eliminated, whatever the choice of the generalized coordinates. This number depends on the particular structure of the robot. The configuration kinematic model described in this section (Eq. (3.27)) for all types of WMR is nonholonomic, i.e., $\dim(q) \rangle \dim(inv(\Delta_1))$. This, however, does not contradict what has been stated before regarding the irreducibility of the posture kinematic state space model, since reducibility of Eq. (3.27) means that there exist at least one smooth function of $\xi, \beta_c, \varphi, \beta_s$ involving explicitly at least one of the variables β_c, φ, which is constant along the trajectories of the system compatible with all the mobility constrains (Eq. (3.7)).

Returning back to Bearcat III which is type (2, 0) robot, δ_m=2 and the configuration kinematic model is [112]:

$$\dot{q} = S(q)\eta \tag{3.31}$$

$$q = \begin{pmatrix} x \\ y \\ \theta \\ \beta_{c3} \\ \varphi_1 \\ \varphi_2 \\ \varphi_3 \end{pmatrix}$$

$$S(q) = \begin{pmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \\ \frac{1}{3.25}\cos\beta_{c3} & -\frac{1}{3.25}(3.25 + 13\sin\beta_{c3}) \\ -\frac{1}{9} & -\frac{13}{9} \\ \frac{1}{9} & -\frac{13}{9} \\ -\frac{1}{9}\sin\beta_{c3} & -\frac{13}{9}\cos\beta_{c3} \end{pmatrix}$$

$$\eta = \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix}$$

The degree of nonholonomy for Bearcat III is as follows:

$$\dim(\Delta_1) = 2, \dim(inv(\Delta_1)) = 6, M = 6 - 2 = 4 \tag{3.32}$$

Hence, the number of coordinates that can be eliminated is 7-6=1. From Eq. (3.31):

$$\dot{\varphi}_1 + \dot{\varphi}_2 = -\frac{2L}{r}\dot{\theta} \tag{3.33}$$

Which means that $(\varphi_1 + \varphi_2 + 2L\theta / r)$ has a constant value along any trajectory compatible with the constrains [112].

3.3 Robot dynamics using Lagrange formulation

3.3.1 Configuration dynamic model

In this section a general configuration dynamic model will be developed using Lagrange formulation that gives a complete description of the dynamics of the system including the generalized forces provided by the actuators, and applicable to all WMRs.

The torques provided by the actuators are denoted by τ_φ for the rotation of the wheels, τ_c for the orientation of the castor wheels, and τ_s for the orientation of the steering wheels respectively. Using the Lagrange formulation, the dynamics of the mobile robot is described by the following (3+N$_c$+N+N$_s$) Lagrange's equations [112]:

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\xi}}\right)^T - \left(\frac{\partial T}{\partial \xi}\right)^T = R^T(\theta)J_1^T(\beta_s,\beta_c)\lambda + R^T(\theta)C_1^T(\beta_s,\beta_c)\mu \tag{3.34}$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\beta}_c}\right)^T - \left(\frac{\partial T}{\partial \beta_c}\right)^T = C_2^T\mu + \tau_c$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\varphi}}\right)^T - \left(\frac{\partial T}{\partial \varphi}\right)^T = J_2^T\lambda + \tau_\varphi$$

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\beta}_s}\right)^T - \left(\frac{\partial T}{\partial \beta_s}\right)^T = \tau_s$$

Where *T* represents the kinetic energy and λ, μ are the Lagrange multipliers associated with motion constrains.

After eliminating the Lagrange multipliers, the system becomes [112]:

$$\begin{gathered}\Sigma^T(\beta_s)R(\theta)[T]_\xi + D(\beta_c)[T]_{\beta_c} + E(\beta_s,\beta_c)[T]_\varphi = \\ \Sigma^T(\beta_s)(D(\beta_c)\tau_c + E^T(\beta_s,\beta_c)\tau_c)\end{gathered} \tag{3.35}$$

$$[T]_{\beta_s} = \tau_s$$

where the compact notation:

$[T]\psi = \frac{d}{dt}\left(\frac{\partial T}{\partial \dot{\psi}}\right)^T - \left(\frac{\partial T}{\partial \psi}\right)^T$ has been used

The kinetic energy of WMR can be expressed as [112]:

$$T = \dot{\xi}^T R^T(\theta)(M(\beta_c)R(\theta)\dot{\xi} + 2V(\beta_c)\dot{\beta}_c + 2W\dot{\beta}_c) + \dot{\beta}_c^T I_c \dot{\beta}_c + \dot{\varphi}^T I_\varphi \dot{\varphi} + \dot{\beta}_s^T I_s \dot{\beta}_s \tag{3.36}$$

Where $M(\beta_c), V(\beta_c), W, I_c, I_\varphi$, and I_s depends on the mass distribution and the inertia moments of the various rigid bodies (cart and wheels) that constitute the robot. Therefore, the general configuration dynamic model of WMR in the state space is as follows [112]:

$$\dot{\xi} = R^T(\theta)\Sigma(\beta_s)\eta \tag{3.37}$$

$$\dot{\beta}_s = \zeta$$

$$\dot{\beta}_c = D(\beta_c)\Sigma(\beta_s)\eta$$

$$\begin{gathered}H_1(\beta_s,\beta_c)\dot{\eta} + \Sigma^T(\beta_s)V(\beta_c)\dot{\zeta} + f_1(\beta_s,\beta_c,\eta,\zeta) = \\ \Sigma^T(\beta_s)(D^T(\beta_c)\tau_c + E^T(\beta_s,\beta_c)\tau_\varphi)\end{gathered}$$

$$V^T(\beta_c)\Sigma(\beta_s)\dot{\eta} + I_s\dot{\zeta} + f_2(\beta_s,\beta_c,\eta,\zeta) = \tau_s$$

$$\dot{\varphi} = E(\beta_c, \beta_s)\Sigma(\beta_s)\eta$$

where:

$$H_1(\beta_s, \beta_c) = \Sigma^T(\beta_s)(M(\beta_c) + D^T(\beta_c) + V(\beta_c)D(\beta_c)$$
$$+ D^T(\beta_c)I_c D(\beta_c) + E^T(\beta_s, \beta_c)I_\varphi E(\beta_s, \beta_c))\Sigma(\beta_s)$$

The model in Eq. (3.37) is very general where it has all the torques (τ_φ, τ_c, and τ_s) that can be potentially applied to the rotation and orientation of robot wheels, in practice a limited number of actuators may be used which delete many components of these torques. However, the WMR had to have actuator configurations that allow a full maneuverability. To ensure a full robot mobility, N_m additional actuators (with $N_m \geq \delta_m$) must be provided for the WMR for either the rotation of some wheels or the orientation of the castor wheels. The vector of theses torques is denoted by τ_m where:

$$\begin{pmatrix} \tau_c \\ \tau_\varphi \end{pmatrix} = P\tau_m \qquad (3.38)$$

Where P is an $((N_c + N) \times N_m)$, using Eq. (3.38), the fourth equation of the general configuration dynamic model became [112]:

$$H_1(\beta_s, \beta_c)\dot{\eta} + \Sigma^T(\beta_s)V(\beta_c)\dot{\zeta} + f_1(\beta_s, \beta_c, \eta, \zeta) =$$
$$B(\beta_s, \beta_c)P\tau_m \qquad (3.39)$$

where:

$$B(\beta_s, \beta_c) = \Sigma^T(\beta_s)(D^T(\beta_c) \qquad E^T(\beta_s, \beta_c))$$

After developing the general configuration dynamic model, this model can be applied to Bearcat III which is type (2, 0) robot by defining the *B* matrix and the *P* matrix.

For Bearcat III, the only wheel that rotates is the castor wheel; hence, the *B* matrix is just for β_c, while the β_s components are zero since there are no steering wheels. The matrix $B(\beta_c)$ is reduced to [112]:

$$B(\beta_{c3}) = \begin{pmatrix} \frac{1}{d}\cos\beta_{c3} & -\frac{1}{r} & \frac{1}{r} & -\frac{1}{r}\sin\beta_{c3} \\ -\frac{1}{d}(d + L\sin\beta_{c3}) & -\frac{L}{r} & -\frac{L}{r} & -\frac{L}{r}\cos\beta_{c3} \end{pmatrix} \tag{3.40}$$

Here we must have two actuators, however, different configurations are admissible as having two rotating actuators in wheel number one and wheel number two, or having one actuator for the orientation of wheel number three and one actuator for the rotation of wheel number one or wheel number two, or alternatively having the two actuators just for the third wheel which is the castor wheel. However, in Bearcat III the two actuators are for the rotation for the two first wheels, one actuator for each of them, while the third wheel which is the castor wheel is freely moving and not activated. Thus, the *P* matrix will be as follows [112]:

$$P = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{3.41}$$

3.3.2 Posture dynamic model

The general posture dynamic model can be developed by writing the general configuration dynamic model in a more compact form [112]:

$$\dot{q} = S(q)u \tag{3.42}$$

$$H(\beta)\dot{u} + f(\beta, u) = F(\beta)\tau_0$$

where:

$$\beta = \begin{pmatrix} \beta_s \\ \beta_c \end{pmatrix},\ q = \begin{pmatrix} \xi \\ \beta \\ \varphi \end{pmatrix},\ u = \begin{pmatrix} \eta \\ \xi \end{pmatrix},\ H(\beta) = \begin{pmatrix} H_1(\beta_s, \beta_c) & \Sigma^T(\beta_s)V(\beta_c) \\ V^T(\beta_{0c})\Sigma(\beta_s) & I_s \end{pmatrix}$$

$$f(\beta, u) = \begin{pmatrix} f_1(\beta_s, \beta_c, \eta, \xi) \\ f_2(\beta_s, \beta_c, \eta, \xi) \end{pmatrix},\ F(\beta) = \begin{pmatrix} B(\beta_s, \beta_c)P & 0 \\ 0 & I \end{pmatrix},\ \tau_0 = \begin{pmatrix} \tau_m \\ \tau_s \end{pmatrix}$$

For Bearcat III this model could be simplified by deleting all the terms related to the steering wheels.

3.4 Robot dynamics using Newton-Euler method

3.4.1 Dynamic analysis

For Bearcat III robot, more simplified kinematic and dynamic model can be achieved by using the Newton-Euler method and just considering the velocity along the x and y axis and the angular velocity, with the robot center of mass as a reference point [114]. Bearcat III structure and dynamic analysis is shown in Fig. 3.7.

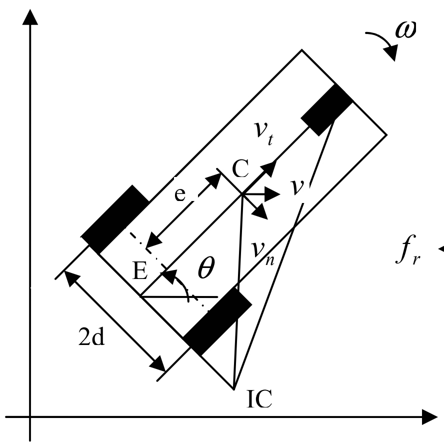
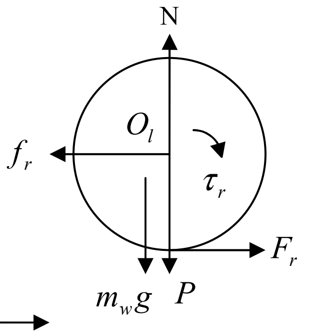
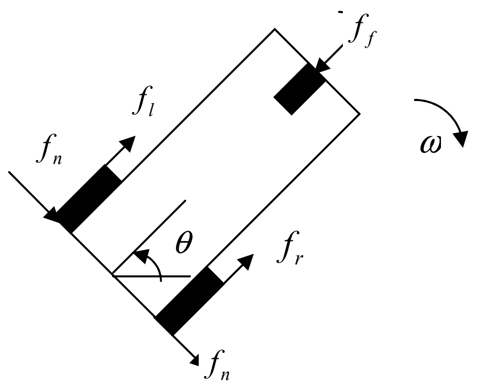
 ω v_t C e v E θ v_n 2d IC		 f_f f_l f_n ω θ f_r f_n
a. Robot Structure	b. Dynamic analysis for the right wheel	c. Dynamic analysis for the robot

Figure 3.7: Robot dynamic analysis [114].

According to Fig. 3.7 and to the kinematic derivation described earlier, the kinematic model with respect to the robot center of gravity (Point C in Fig. 3.7 a.) can be described as follows [114]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ \sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_t \\ v_n \\ \omega \end{bmatrix} \tag{3.43}$$

Where v_t, v_n, ϖ can be defined in terms of the angular velocity of the robot left wheel ω_l and the angular velocity of the robot right wheel ω_r as follows [114]:

$$\begin{bmatrix} v_t \\ v_n \\ w \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{er}{2d} & \frac{-er}{2d} \\ \frac{r}{2d} & \frac{-r}{2d} \end{bmatrix} \begin{bmatrix} \omega_l \\ \omega_r \end{bmatrix} \tag{3.44}$$

However, Eq. 3.43 can be simplified by utilizing that $v_n = e\omega$ as follows [114]:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & e\sin\theta \\ \sin\theta & -e\cos\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_t \\ \omega \end{bmatrix} \tag{3.45}$$

The nonholonomic constraint can be obtained directly from Eq. 3.45 as [114]:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = \omega e \tag{3.46}$$

For the center of the wheel axes (Point E in Fig. 3.7 a.) $e = 0$ and hence Eq. 3.46 reduces to [114]:

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0 \tag{3.47}$$

This means that there is no motion in the direction of the wheel axis.

Another constraint for the kinematic model comes from the inertial structure of the robot where the robot path cannot exceed the minimum turning radius or the maximum curvature [114]:

$$\rho \geq R_{\min imum} \text{ , or } k \leq K_{\max imum} \tag{3.48}$$

From Fig. 3.7 b., the Newton-Euler equation for the right wheel can be described as [114]:

$$F_r - f_r = m_w \ddot{x}_r \tag{3.49}$$

$$\tau_r - F_r \cdot r = J_w \dot{\omega}_r$$

where:

F_r: is the reaction force applied to the right wheel by the rest of the robot.

f_r: is the friction force between the right wheel and the ground.

m_w: is the mass of the wheel.

τ_r: is the torque acting on the right wheel that is provided by the right motor.

r: is the radius of the wheel.

J_w: is the inertia of the wheel.

Note that the coriolis part had been deleted since it is negligible due to the fact that the wheel inertia is much smaller compared to the robot inertia.

Returning back to pure rolling and non-slipping assumptions that was stated in section 3.1.2.1, the following can be stated [114]:

$$\dot{x}_r = r \cdot \omega_r \tag{3.50}$$

$$F_r \leq \mu(P + m_w g)$$

hence:

$$\tau_r \leq \frac{\mu(P+m_w g)(m_w r^2 + J_w) - f_r \cdot J_w}{m_w r}$$

where:

μ : is the maximum static friction coefficient between the wheel and the ground.

P : is the reaction force applied on the wheel by the rest of the robot.

The dynamic analysis for the left wheel can be obtained in the same way [114]:

$$F_l - f_l = m_w \ddot{x}_l \tag{3.51}$$

$$\tau_l - F_l \cdot r = J_w \dot{\omega}_l$$

$$\dot{x}_l = r \cdot \omega_l$$

where:

F_l : is the reaction force applied to the left wheel by the rest of the robot.

f_l : is the friction force between the left wheel and the ground.

τ_l : is the torque acting on the left wheel that is provided by the left motor.

From Fig. 3.7 c., the robot Newton-Euler equation can be derived as [114]:

$$(f_l + f_r - f_f)\cos\theta + 2f_n \sin\theta = m\ddot{x}_c \tag{3.52}$$

$$(f_l + f_r - f_f)\sin\theta + 2f_n \cos\theta = m\ddot{y}_c$$

$$f_l \cdot d - f_r \cdot d - 2f_n \cdot e = J_c \ddot{\theta}$$

where:

f_f : is the reaction force applied to the robot by the front wheel (castor wheel).

f_n : is the resultant normal force.

m : is the mass of the robot excluding the wheels.

J_c: is the inertia of the robot excluding the wheels.

x_c, y_c, θ: are the coordination and the orientation of the center of gravity of the robot.

The dynamic model of the robot can be obtained from Eq. 3.43-3.52 in terms of ξ as follows [114]:

$$N(\xi)\ddot{\xi} + C(\xi,\dot{\xi})\dot{\xi} + \eta(\xi) = I(\xi)\tau \tag{3.53}$$

where:

$$\xi = \begin{bmatrix} x_c \\ y_E \\ \theta \end{bmatrix}, \; N(\xi) = \begin{bmatrix} \frac{(mr^2 + 2J_0\cos^2\theta)}{r} & \frac{2J_0\sin\theta\cos\theta}{r} & -mre\sin\theta \\ \frac{2J_0\sin\theta\cos\theta}{r} & \frac{(mr^2 + 2J_0\sin^2\theta)}{r} & mre\sin\theta \\ 0 & 0 & \frac{(J_c r^2 + 2d^2 J_0)}{r^2} \end{bmatrix}$$

$$C(\xi,\dot{\xi}) = \begin{bmatrix} \frac{-2\dot{\theta}J_0\sin\theta\cos\theta}{r} & \frac{2\dot{\theta}J_0\cos^2\theta}{r} & -\dot{\theta}mre\cos\theta \\ \frac{-2\dot{\theta}J_0\sin^2\theta}{r} & \frac{2\dot{\theta}J_0\sin\theta\cos\theta}{r} & -\dot{\theta}mre\cos\theta \\ 0 & 0 & 0 \end{bmatrix}, \; I(\xi) = \begin{bmatrix} \cos\theta & \cos\theta \\ \sin\theta & \sin\theta \\ \frac{d}{r} & \frac{-d}{r} \end{bmatrix}$$

$$\eta(\xi) = \begin{bmatrix} -2f_n r\sin\theta \\ 2f_n r\cos\theta \\ -2f_n e \end{bmatrix}, \; \tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}, \; J_0 = J_w + m_w r^2$$

It can be noted that Eq. 3.53 form is similar to the form of the dynamics of the robot manipulator, which is usually represented as [80]:

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau \tag{3.54}$$

where:

$M(q)$: is the inertia matrix.

$V_m(q,\dot{q})$: is the Coriolis/centripetal matrix.

$F(\dot{q})$: is the friction term.

$G(q)$: is the gravity vector

τ_d: is the torque from the disturbances.

τ: is the control input torque.

Except for the presence of the matrix $I(\xi)$ in the right hand side of the equation. In order to use this formulations for the approximation-based controller, τ need to be alone in the left side of the equation. This can be done by pre-multiply all the equation by the inverse of $I(\xi)$ matrix. However, the $I(\xi)$ matrix is not square, thus, the Moore-Penrose inverse need to be calculated for this matrix.

3.4.2 Calculation of the Pseudo-inverse matrix (Moore-Penrose generalized inverse)

The Pseudo-inverse matrix or the Moore-Penrose generalized inverse is a generalization of the inverse to rectangular matrices $A \in R^{m \times n}$ where $m \neq n$. It was defined independently by Moore in 1920 and Penrose 1955 [115].

For every matrix $A \in R^{n \times m}$, a unique matrix $P \in R^{n \times m}$ exists, P is a pseudo-inverse of A if it satisfies the following four Moore-Penrose conditions [116]:

1. $APA = A$ (3.55)
2. $PAP = P$
3. $(AP)^T = AP$
4. $(PA)^T = PA$

P can be calculated as follows [117]:

- If $A \in R^{m \times n}, m \leq n, rank(A) = m$ then $P = A^T(AA^T)^{-1}$ (3.56)

- If $A \in R^{m \times n}, n \leq m, rank(A) = n$ then $P = (A^T A)^{-1} A^T$
- If $A \in R^{n \times n}, rank(A) = n$ then $P = A^{-1}$

Based on Eq. 3.56 I^{-1} is:

$$I^{-1}(\xi) = \begin{bmatrix} \frac{\cos\theta}{2} & \frac{\sin\theta}{2} & \frac{r}{2d} \\ \frac{\cos\theta}{2} & \frac{\sin\theta}{2} & \frac{-r}{2d} \end{bmatrix} \tag{3.57}$$

Pre-multiplying Eq. 3.53 by I^{-1}:

$$I^{-1}(\xi)N(\xi)\ddot{\xi} + I^{-1}(\xi)C(\xi,\dot{\xi})\dot{\xi} + I^{-1}(\xi)\eta(\xi) = \tau \tag{3.58}$$

Let $M(\xi) = I^{-1}(\xi)N(\xi)$, then $M(\xi)$ is defined as:

$$M(\xi) = \begin{bmatrix} \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2ed\sin^2\theta - mr^2ed\sin\theta\cos\theta + J_cr^2 + 2J_0d^2)}{2rd} \\ \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2ed\sin^2\theta - mr^2ed\sin\theta\cos\theta - J_cr^2 - 2J_0d^2)}{2rd} \end{bmatrix} \tag{3.59}$$

Let $J(\xi,\dot{\xi}) = I^{-1}(\xi)C(\xi,\dot{\xi})$, then $J(\xi,\dot{\xi})$ is defined as:

$$J(\xi,\dot{\xi}) = \begin{bmatrix} \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \\ \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \end{bmatrix} \tag{3.60}$$

Let $F = I^{-1}(\xi)\eta(\xi)$, then F is defined as:

$$F = \begin{bmatrix} \frac{-f_n er}{d} \\ \frac{-f_n er}{d} \end{bmatrix} \tag{3.61}$$

The dynamic model of the robot can be defined as:

$$M(\xi)\ddot{\xi} + J(\xi,\dot{\xi})\dot{\xi} + F = \tau$$

where:

$$\xi = \begin{bmatrix} x_c \\ y_E \\ \theta \end{bmatrix}$$

$$M(\xi) = \begin{bmatrix} \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2 ed\sin^2\theta - mr^2 ed\sin\theta\cos\theta + J_c r^2 + 2J_0 d^2)}{2rd} \\ \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2 ed\sin^2\theta - mr^2 ed\sin\theta\cos\theta - J_c r^2 - 2J_0 d^2)}{2rd} \end{bmatrix}$$

$$J(\xi, \dot{\xi}) = \begin{bmatrix} \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \\ \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \end{bmatrix}$$

$$F = \begin{bmatrix} \frac{-f_n er}{d} \\ \frac{-f_n er}{d} \end{bmatrix}, \ \tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

(3.62)

This dynamic model is in terms of ξ which is defined as follows:

$$\xi = \begin{bmatrix} x_c \\ y_E \\ \theta \end{bmatrix} \qquad (3.63)$$

Where point C and E as shown in Fig. 3.7 a., however, it is more convenient to develop the dynamic model using the motion of point C, which is the robot center of gravity. The relation between point C and point E in Fig. 3.7 a. is given by [114]:

$$x_c = x_E + e\cos\theta \qquad (3.64)$$

$$y_c = y_E + e\sin\theta$$

Working with the robot dynamic model in Eq. 3.62 and defining ζ as:

$$\zeta = \begin{bmatrix} x_c \\ y_c \\ \theta \end{bmatrix} \tag{3.65}$$

$\dot{\xi}$ and $\ddot{\xi}$ can be defined as:

$$\xi = \begin{bmatrix} x_c \\ y_E \\ \theta \end{bmatrix} = \begin{bmatrix} x_c \\ y_c - e\sin\theta \\ \theta \end{bmatrix} \tag{3.66}$$

$$\dot{\xi} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_E \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \dot{x}_c \\ \dot{y}_c - e\dot{\theta}\cos\theta \\ \dot{\theta} \end{bmatrix}$$

$$\ddot{\xi} = \begin{bmatrix} \ddot{x}_c \\ \ddot{y}_E \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \ddot{x}_c \\ \ddot{y}_c - (e\ddot{\theta}\cos\theta - e\dot{\theta}^2\sin\theta) \\ \ddot{\theta} \end{bmatrix}$$

Substituting $\dot{\xi}$ and $\ddot{\xi}$ into Eq. 3.62, the robot dynamic model can be defined as a function of ζ as follows:

$$M(\zeta)\ddot{\zeta} + J(\zeta,\dot{\zeta})\dot{\zeta} + G(\zeta,\dot{\zeta},\ddot{\zeta}) = \tau$$

where:

$$\zeta = \begin{bmatrix} x_c \\ y_c \\ \theta \end{bmatrix}, \ \tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

$$M(\zeta) = \begin{bmatrix} \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2ed\sin^2\theta - mr^2ed\sin\theta\cos\theta + J_cr^2 + 2J_0d^2)}{2rd} \\ \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2ed\sin^2\theta - mr^2ed\sin\theta\cos\theta - J_cr^2 - 2J_0d^2)}{2rd} \end{bmatrix}$$

$$J(\zeta,\dot{\zeta}) = \begin{bmatrix} \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\,\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \\ \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\,\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \end{bmatrix}$$

$$G(\zeta,\dot{\zeta},\ddot{\zeta}) = \begin{bmatrix} \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r}(e\dot{\theta}^2\sin\theta - e\ddot{\theta}\cos\theta) - \frac{J_0\dot{\theta}\cos\theta}{r}(e\dot{\theta}\cos\theta) - \frac{f_n er}{d} \\ \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r}(e\dot{\theta}^2\sin\theta - e\ddot{\theta}\cos\theta) - \frac{J_0\dot{\theta}\cos\theta}{r}(e\dot{\theta}\cos\theta) - \frac{f_n er}{d} \end{bmatrix}$$

(3.67)

3.4.3 Properties of the dynamic model

1. The robot dynamic model form is similar to the form of the dynamics of the robot manipulator. This can be deduced from comparing Eq. 3.67 to Eq. 3.54.
2. The inertia matrix $M(\zeta)$, or $M(\xi)$ is always positive.
3. The matrices $M(\zeta), J(\zeta,\dot{\zeta}), G(\zeta,\dot{\zeta},\ddot{\zeta})$ are only function of $\theta, \dot{\theta}$, and $\ddot{\theta}$ and not of x_c, y_c, or y_e.
4. To check whether the term $\dot{M}(\xi) - 2J(\xi,\dot{\xi})$ or $\dot{N}(\xi) - 2C(\xi,\dot{\xi})$ are skew symmetry as in the case of the robot manipulator, the following had been calculated:

- For $\dot{M}(\xi) - 2J(\xi,\dot{\xi})$:

Let $SI = \dot{M}(\xi) - 2J(\xi,\dot{\xi})$

where $M(\xi)$ and $J(\xi,\dot{\xi})$ are as given in Eq. 60.3

$$\dot{M}(\xi) = \begin{bmatrix} \frac{(-mr^2\dot{\theta}\sin\theta - 2J_o\dot{\theta}\sin\theta)}{2r} & \frac{(mr^2\dot{\theta}\cos\theta + 2J_o\dot{\theta}\cos\theta)}{2r} & \frac{(2mr^2ed\dot{\theta}\sin\theta\cos\theta - mr^2ed\dot{\theta}\cos^2\theta + mr^2ed\dot{\theta}\sin^2\theta}{2rd} \\ \frac{(-mr^2\dot{\theta}\sin\theta - 2J_o\dot{\theta}\sin\theta)}{2r} & \frac{(mr^2\dot{\theta}\cos\theta + 2J_o\dot{\theta}\cos\theta)}{2r} & \frac{(2mr^2ed\dot{\theta}\sin\theta\cos\theta - mr^2ed\dot{\theta}\cos^2\theta + mr^2ed\dot{\theta}\sin^2\theta}{2rd} \end{bmatrix}$$

$$-2J(\xi,\dot{\xi}) = \begin{bmatrix} \frac{2J_0\dot{\theta}\sin\theta}{r} & \frac{-2J_0\dot{\theta}\cos\theta}{r} & mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta) \\ \frac{2J_0\dot{\theta}\sin\theta}{r} & \frac{-2J_0\dot{\theta}\cos\theta}{r} & mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta) \end{bmatrix}$$

$$SI = \dot{M}(\xi) - 2J(\xi,\dot{\xi}) =$$

$$\begin{bmatrix} \frac{(-mr^2\dot{\theta}\sin\theta + 2J_o\dot{\theta}\sin\theta)}{2r} & \frac{(mr^2\dot{\theta}\cos\theta - 2J_o\dot{\theta}\cos\theta)}{2r} & \frac{(2mr^2ed\dot{\theta}\sin\theta\cos\theta - mr^2ed\dot{\theta}\cos^2\theta + mr^2ed\dot{\theta}\sin^2\theta + 2mr^2ed\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2rd} \\ \frac{(-mr^2\dot{\theta}\sin\theta + 2J_o\dot{\theta}\sin\theta)}{2r} & \frac{(mr^2\dot{\theta}\cos\theta - 2J_o\dot{\theta}\cos\theta)}{2r} & \frac{(2mr^2ed\dot{\theta}\sin\theta\cos\theta - mr^2ed\dot{\theta}\cos^2\theta + mr^2ed\dot{\theta}\sin^2\theta + 2mr^2ed\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2rd} \end{bmatrix}$$

$$SI = \begin{bmatrix} \frac{(-mr^2\dot{\theta}\sin\theta + 2J_o\dot{\theta}\sin\theta)}{2r} & \frac{(mr^2\dot{\theta}\cos\theta - 2J_o\dot{\theta}\cos\theta)}{2r} & \frac{(4mre\dot{\theta}\sin\theta\cos\theta + mre\dot{\theta})}{2} \\ \frac{(-mr^2\dot{\theta}\sin\theta + 2J_o\dot{\theta}\sin\theta)}{2r} & \frac{(mr^2\dot{\theta}\cos\theta - 2J_o\dot{\theta}\cos\theta)}{2r} & \frac{(4mre\dot{\theta}\sin\theta\cos\theta + mre\dot{\theta})}{2} \end{bmatrix} \tag{3.68}$$

If SI is a skew symmetric, then it must satisfy $SI^T = -SI$, however, $SI \in R^{2\times3}$ while $SI^T \in R^{3\times2}$ so it cannot be proved that it is a skew symmetric but it is very clear that there is some kind of symmetry where the first raw equals the second row and $SI_{11} = -SI_{12}$.

- For $\dot{N}(\xi) - 2C(\xi,\dot{\xi})$:

Let $SII = \dot{N}(\xi) - 2C(\xi,\dot{\xi})$

where $N(\xi)$ and $C(\xi,\dot{\xi})$ as given in Eq. 3.53:

$$\dot{N}(\xi) = \begin{bmatrix} \frac{-4J_o\dot{\theta}\sin\theta\cos\theta}{r} & \frac{2J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta}{r} & -mre\dot{\theta}\cos\theta \\ \frac{2J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta}{r} & \frac{4J_o\dot{\theta}\sin\theta\cos\theta}{r} & mre\dot{\theta}\cos\theta \\ 0 & 0 & 0 \end{bmatrix}$$

$$-2C(\xi,\dot{\xi}) = \begin{bmatrix} \frac{\dot{\theta}J_0\sin\theta\cos\theta}{r} & \frac{-\dot{\theta}J_0\cos^2\theta}{r} & 2\dot{\theta}mre\cos\theta \\ \frac{\dot{\theta}J_0\sin^2\theta}{r} & \frac{-\dot{\theta}J_0\sin\theta\cos\theta}{r} & 2\dot{\theta}mre\cos\theta \\ 0 & 0 & 0 \end{bmatrix}$$

$$SII = \dot{N}(\xi) - 2C(\xi,\dot{\xi}) =$$

$$\begin{bmatrix} \frac{-4J_0\dot{\theta}\sin\theta\cos\theta + \dot{\theta}J_0\sin\theta\cos\theta}{r} & \frac{2J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta - \dot{\theta}J_0\cos^2\theta}{r} & -mre\dot{\theta}\cos\theta + 2\dot{\theta}mre\cos\theta \\ \frac{2J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta + \dot{\theta}J_0\sin^2\theta}{r} & \frac{4J_o\dot{\theta}\sin\theta\cos\theta - \dot{\theta}J_0\sin\theta\cos\theta}{r} & mre\dot{\theta}\cos\theta + 2\dot{\theta}mre\cos\theta \\ 0 & 0 & 0 \end{bmatrix}$$

$$SII = \begin{bmatrix} \dfrac{-3J_0\dot{\theta}\sin\theta\cos\theta}{r} & \dfrac{J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta}{r} & mre\dot{\theta}\cos\theta \\ \dfrac{2J_0\dot{\theta}\cos^2\theta - J_0\dot{\theta}\sin^2\theta}{r} & \dfrac{3J_0\dot{\theta}\sin\theta\cos\theta}{r} & 3mre\dot{\theta}\cos\theta \\ 0 & 0 & 0 \end{bmatrix}$$

(3.69)

If SII is a skew symmetric, then it must satisfy $SII^T = -SII$:

(3.70)

$$SII = \begin{bmatrix} \dfrac{-3J_0\dot{\theta}\sin\theta\cos\theta}{r} & \dfrac{J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta}{r} & mre\dot{\theta}\cos\theta \\ \dfrac{2J_0\dot{\theta}\cos^2\theta - J_0\dot{\theta}\sin^2\theta}{r} & \dfrac{3J_0\dot{\theta}\sin\theta\cos\theta}{r} & 3mre\dot{\theta}\cos\theta \\ 0 & 0 & 0 \end{bmatrix}$$

$$SII^T = \begin{bmatrix} \dfrac{-3J_0\dot{\theta}\sin\theta\cos\theta}{r} & \dfrac{2J_0\dot{\theta}\cos^2\theta - J_0\dot{\theta}\sin^2\theta}{r} & 0 \\ \dfrac{J_0\dot{\theta}\cos^2\theta - 2J_0\dot{\theta}\sin^2\theta}{r} & \dfrac{3J_0\dot{\theta}\sin\theta\cos\theta}{r} & 0 \\ mre\dot{\theta}\cos\theta & 3mre\dot{\theta}\cos\theta & 0 \end{bmatrix}$$

$$-SII = \begin{bmatrix} \dfrac{3J_0\dot{\theta}\sin\theta\cos\theta}{r} & \dfrac{-J_0\dot{\theta}\cos^2\theta + 2J_0\dot{\theta}\sin^2\theta}{r} & -mre\dot{\theta}\cos\theta \\ \dfrac{-2J_0\dot{\theta}\cos^2\theta + J_0\dot{\theta}\sin^2\theta}{r} & \dfrac{-3J_0\dot{\theta}\sin\theta\cos\theta}{r} & -3mre\dot{\theta}\cos\theta \\ 0 & 0 & 0 \end{bmatrix}$$

hence, $SII^T \neq -SII$

Hence, $SII^T \neq -SII$ and the matrix $\dot{N}(\xi) - 2C(\xi,\dot{\xi})$ is not skew symmetric.

3.4.4 Bearcat III dynamic model

To customize the dynamic model for Bearcat III, we need to substitute the values for $m, r, J_0, e, d, J_c, f_n$ in the Eq. 3.62. According to Fig. 3.7 for Bearcat III, $m = 306.18kg$, $r = 0.2095m$, $e = 0.338m$, $d = 0.432m$, and J_0, J_c, f_n need to be calculated.

3.4.4.1 Calculation of the moment of inertia

The second moment or moment of inertia is the name that had been given to rotational inertia, the rotational analog of mass for linear motion [118]. The moment of inertia concept is illustrated in Fig. 3.8.

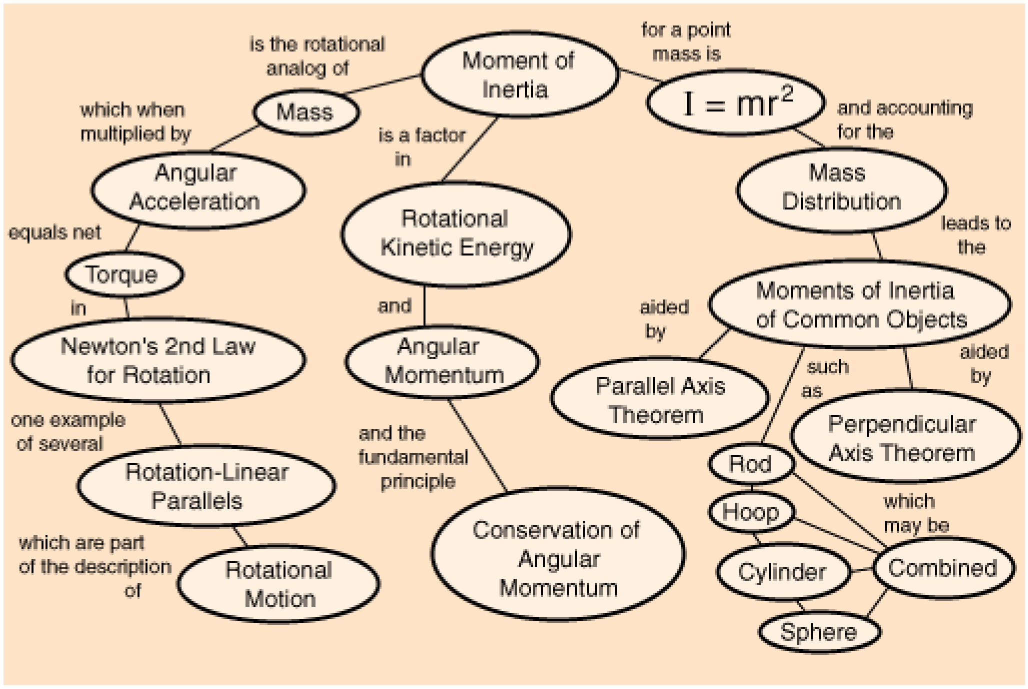


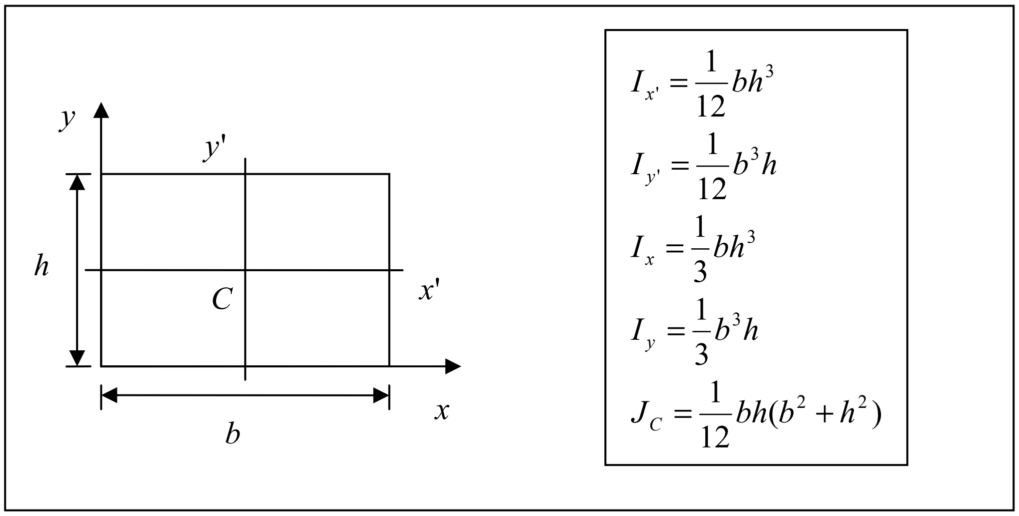
Figure 3.8: Moment of inertia [118].

The moment of inertia for an area with respect to the x and y-axis is defined as follows [119]:

$$I_x = \int y^2 dA \tag{3.71}$$

$$I_y = \int x^2 dA$$

For a rectangle, the moment of inertia can be calculated according to Fig. 3.9 [119]:



$$I_{x'} = \frac{1}{12} bh^3$$

$$I_{y'} = \frac{1}{12} b^3 h$$

$$I_x = \frac{1}{3} bh^3$$

$$I_y = \frac{1}{3} b^3 h$$

$$J_C = \frac{1}{12} bh(b^2 + h^2)$$

Figure 3.9: Moment of inertia of a rectangle [119].

Polar moment of inertia can be calculated based on the following integral [119]:

$$J = \int r^2 dA \tag{3.72}$$

For a body of mass m rotating about an axis AA' the moment of inertia can be calculated as [119]:

$$I = \int r^2 dm \tag{3.73}$$

To calculate the moment of inertia of a body with respect to a coordinate axis x, y, z, the same equation can be modified by substituting r by the distance of the element of mass to that axis as follows [119]:

$$I_x = \int (y^2 + z^2) dm \tag{3.74}$$

$$I_y = \int (z^2 + x^2) dm$$

$$I_z = \int (x^2 + y^2) dm$$

Bearcat III has a rectangular prism shape, thus, Bearcat III mass moment of inertia (J_c in Eq. 60.3) can be calculated according to the Fig. 3.10 to be 43 kgm^2.

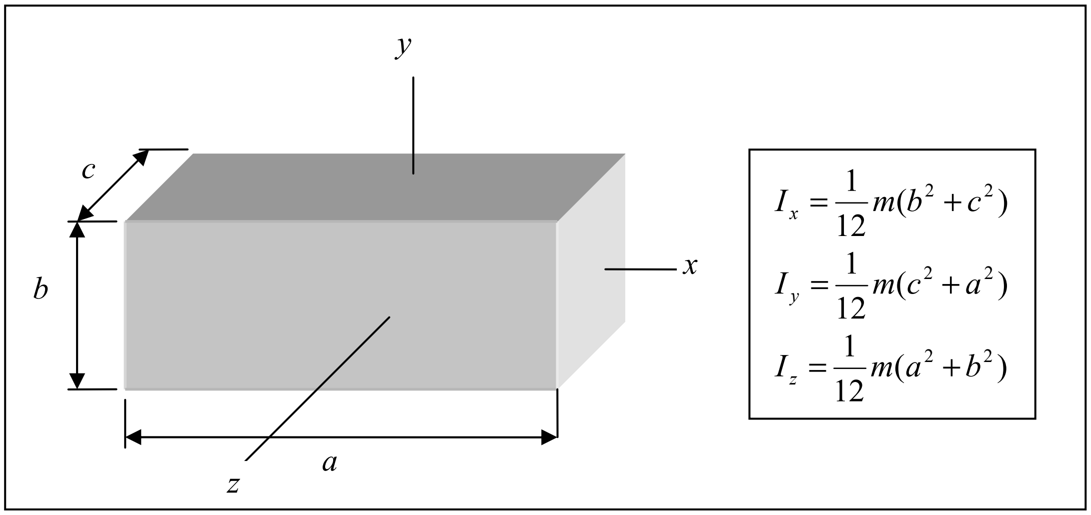

y
c
b
x
a
z
$I_x = \frac{1}{12}m(b^2 + c^2)$
$I_y = \frac{1}{12}m(c^2 + a^2)$
$I_z = \frac{1}{12}m(a^2 + b^2)$

Figure 3.10: Mass moment of inertia of a rectangular prism [119].

For a thin disc the mass moment of inertia can be calculated based on the following figure.

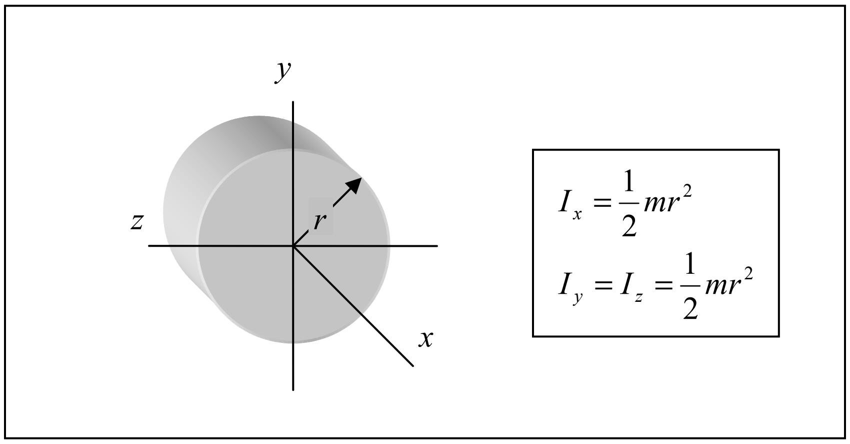


Figure 3.11: Mass moment of inertia of a thin disc [119].

If the disc is hollow then the mass moment of inertia is calculated by subtracting the outer diameter from the inner diameter as follows:

$$I = \frac{1}{2} m(r_e^{\,2} - r_i^2) \tag{3.75}$$

where:

r_e : is the exterior radius of the disc.

r_i : is the interior radius of the disc.

For a body consisting of more than one component, the moment of inertia for this body can be calculated by adding the moment of inertia of each of it's components with respect to a given axis.

To calculate the moment of inertia for Bearcat III wheel, J_w, Eq. 3.75 can be used, however, the wheel consists of two components, namely, a rubber tire and a metal rim, the moment of inertia for the robot wheel is calculated as:

$$J_w = \frac{1}{2} m_t (r_{te}^{\,2} - r_{ti}^2) + \frac{1}{2} m_r (r_{re}^2 - r_{ri}^2) = 0.055 kgm^2 \tag{3.76}$$

J_0 in Eq. 3.62 is defined as [114]:

$$J_0 = J_w + m_w r^2 \tag{3.77}$$

where:

J_w : is the wheel inertia.

m_w : is the total mass of the wheel.

Substituting the value of J_w from Eq. 3.76 for Bearcat III and $m_w = 5kg$, J_0 is calculated to be $0.274 kgm^2$.

3.4.4.2 Calculation of the resultant normal force f_n

f_n the force in Fig. 3.7 c. has two main components. The first component represent the reaction to the normal friction force between the wheel and the ground, while the second component is the centrifugal force which represent the tendency of the wheel to leave its curved path.

f_n can be calculated as follows:

$$f_n = \mu N + f_c \tag{3.78}$$

where:

μ : is the friction coefficient between the ground and the wheel.

N : is the normal force between the wheel and the ground.

f_c : is the centrifugal force.

The normal force is the reaction to the gravitational forces that is also two components (as shown in Fig. 3.7 b.): the first component is the weight of the wheel while the second component is the portion of the weight of the robot that is carried by the wheel.

Bearcat III has three wheels and it is assumed that the weight of the robot is equally distributed on the three wheels; hence, N can be calculated as follows:

$$N = P + m_w g \tag{3.79}$$

$$P = \frac{1}{3} mg$$

where:

m_w : is the total mass of the wheel.

m : is the mass of the robot excluding the wheels.

The centrifugal force, f_c, can be calculated as follows:

$$f_c = \frac{m\upsilon^2}{\rho} \tag{3.80}$$

where:

υ: is the velocity of the robot in the direction toward the center of the circular path.

ρ: is the radius of the circular path.

However, υ, the velocity of the robot toward the center of the circular path (IC as shown in Fig. 3.7 a.), is very small, and the radius of the circular path ρ is very large, hence, the centrifugal force f_c is very small and can be neglected.

Hence, the resultant normal force f_n is only attributed to the reaction to the normal force.

$$f_n = \mu(\frac{1}{3}mg + m_w g) \tag{3.81}$$

The value of the frictional coefficient μ between the ground and the wheel depend of the type of the surface of the ground, for grass 0.6 is common, while for concrete 0.9 is usually used. Bearcat III usually moves on grass, therefore, 0.6 was used in the calculations. Substituting the parameters for bearcat III into equation 3.81, f_n is calculated to be 629.45 N.

Substituting these values into Eq. 3.67, Bearcat III dynamic model is:

$$M_B(\zeta)\ddot{\zeta} + J_B(\zeta,\dot{\zeta})\dot{\zeta} + G_B(\zeta,\dot{\zeta},\ddot{\zeta}) = \tau \tag{3.82}$$

where:

$$\zeta = \begin{bmatrix} x_c \\ y_c \\ \theta \end{bmatrix}, \tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

$$M_B(\zeta) = \begin{bmatrix} 33.454\cos\theta & 33.454\sin\theta & 10.866\sin^2\theta - 10.866\sin\theta\cos\theta + 11.014 \\ 33.454\cos\theta & 33.454\sin\theta & 10.866\sin^2\theta - 10.866\sin\theta\cos\theta - 11.014 \end{bmatrix}$$

$$J_B(\zeta,\dot{\zeta}) = \begin{bmatrix} -1.305\dot{\theta}\sin\theta & 1.305\dot{\theta}\cos\theta & -10.866\dot{\theta}\cos\theta(\sin\theta + \cos\theta) \\ -1.305\dot{\theta}\sin\theta & 1.305\dot{\theta}\cos\theta & -10.866\dot{\theta}\cos\theta(\sin\theta + \cos\theta) \end{bmatrix}$$

$$G_B(\zeta,\dot{\zeta},\ddot{\zeta}) = \begin{bmatrix} 11.308\dot{\theta}^2\sin^2\theta - 0.441\dot{\theta}^2\cos^2\theta - 11.308\ddot{\theta}\sin\theta\cos\theta - 103.422 \\ 11.308\dot{\theta}^2\sin^2\theta - 0.441\dot{\theta}^2\cos^2\theta - 11.308\ddot{\theta}\sin\theta\cos\theta - 103.422 \end{bmatrix}$$

Chapter 4: ANN

4.1 Introduction

Hassoun [120] defines ANN as “ parallel computational models comprised of densely interconnected adaptive processing units, they are viable computational models for a wide range of problems including pattern classification, speech synthesis and recognition, adaptive interfaces, function approximation, image compression, associative memory, clustering, forecasting and prediction, combinatorial optimization, nonlinear system modeling, and control”. ANN can also be defined as a complex nonlinear interconnected systems that have a broad range of properties.

Neural network are closely modeled based on biological processes for processing information. Specifically, the nervous system and the neuron, where the signals propagate in a form of potential differences between the inside and outside of the cell. The neuronal cell is shown in Fig. 4.1 [80].

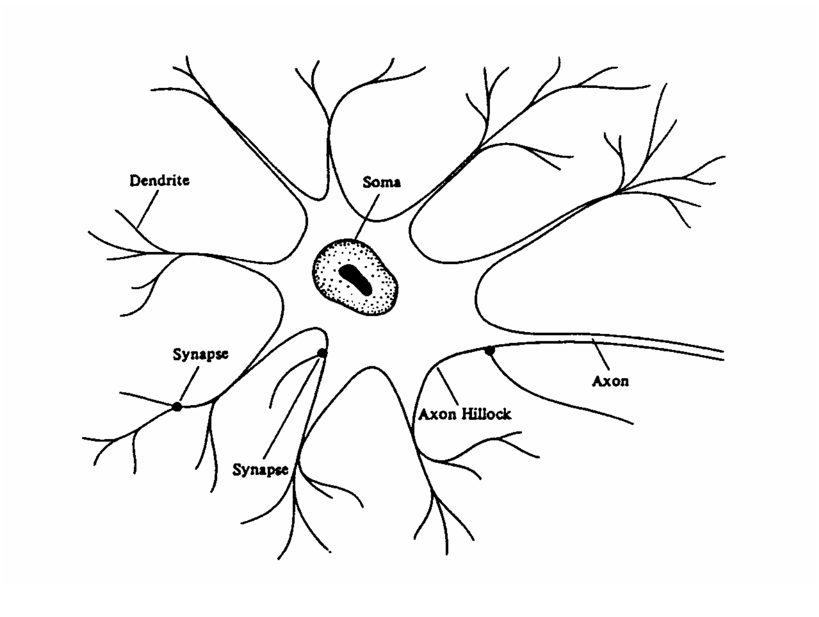


Figure 4.1: Neuronal cell [80].

The artificial neuron is a simplified abstraction of the biological neuronal cell, the output of the first neuron may serve as the input to another neuron and hence the name, the artificial neural network (ANN) [121].

Neuroengineering is the field that studies ANN, it can be defined as: "A field which tries to copy over the known capabilities of the brain into computer hardware and software. More precisely, it develops mathematical designs, which could be embodied directly in hardware or simulated in software" [122].

There are three subdivision of neuroengineering where the researchers try to map out brain capabilities. In the first subdivision researchers aim to understand specific pathways and connections for the brain that are responsible for implementing specific abilities without caring about the procedure of learning these abilities. The second subdivision researchers however, focus on the learning ability of the brain. Three kinds of learning had been built: supervised, reinforcement learning, and unsupervised learning. The third subdivision of neuroengineering is the one related to the Hopfield nets where researchers notice that certain ANN, that were derived in the biologically-based literature, could also be utilized to solve complicated static optimization problems, or minimize complex quadratics [122]. Since ANN is an extensive topic that had been described in many books, only the terms important to this research will be described here.

4.2 ANN mathematical model

The neuron can be modeled mathematically as shown in Fig. 4.2, where:

- The cell inputs are $x_1(t),...,x(t)_n$
- The cell output is $y(t)$
- The dendrite or input weights are $v_1,....,v_n$
- The firing threshold or bias weight is v_0

- The cell function or activation function is f

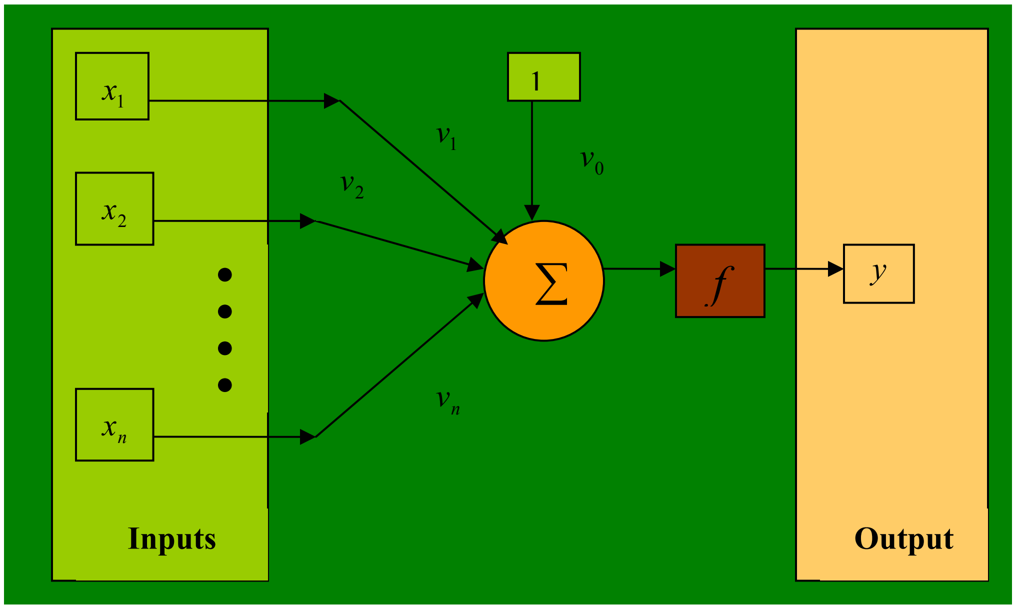


Figure 4.2: Neuron mathematical model.

The output can be expressed as [80]:

$$y(t) = f(\sum_{j=1}^{n} v_j x_j(t) + v_0). \tag{4.1}$$

The activation function can be linear or nonlinear; some common activation functions are shown in Fig. 4.3 [80].

Eq. (4.1) can be streamlined by defining the augmented column vector of the cell inputs $\bar{x}(t) \in \Re^{n+1}$ and the input weights $\bar{v}(t) \in \Re^{n+1}$ as [80]:

$$\bar{x}(t) = [1 \quad x_1 \quad x_2 \quad \dots \quad x_n]^T, \; \bar{v}(t) = [v_0 \quad v_1 \quad v_2 \quad \dots \quad v_n]^T \tag{4.2}$$

Then Eq. (4.1) can be written in matrix notation as [80]:

$$y = f(v^T x). \tag{4.3}$$

One layer ANN is commonly used which has L cells, all fed by the same input signals $x_j(t)$, and producing one output per neuron $y_l(t)$. This network can be modeled as [80]:

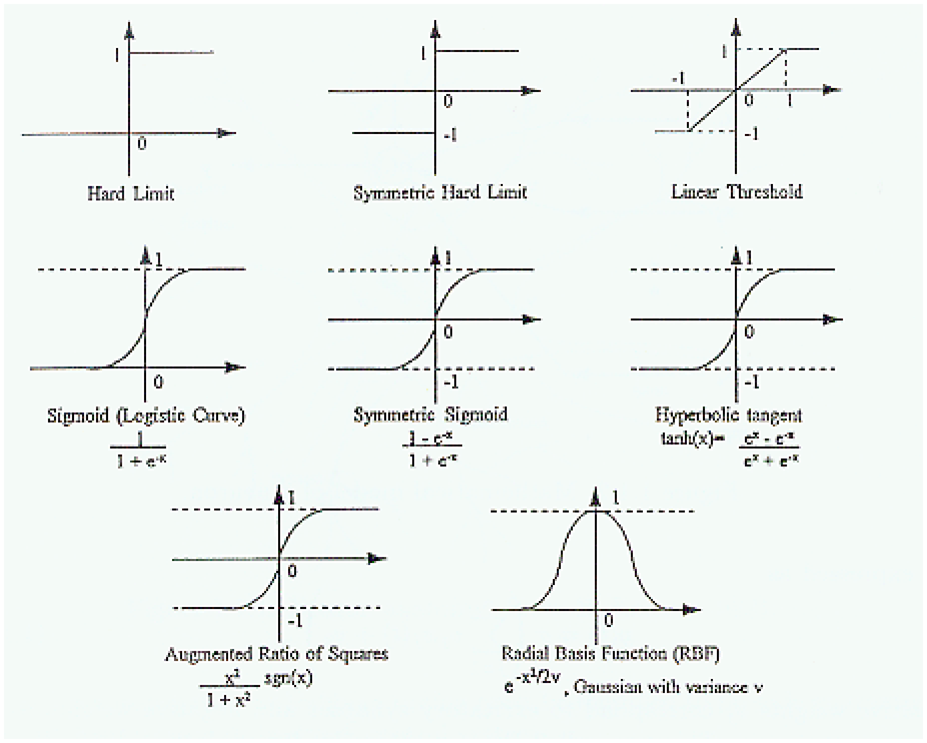


Figure 4.3: Common ANN activation functions [80].

$$y_l = f(\sum_{j=1}^{n} v_{lj} x_j + v_{l0}); l = 1,2,...,L. \tag{4.4}$$

The navigation algorithm developed in this research uses an ANN that consists of five layers as shown in Fig. 4.4 where the second layer input is the first layer output and so on.

Based on Eq. (4.4) the output for that network can be written as follows:

$$y_l = f_5(\sum_{j=1}^{n5} v_{lj} f_4(\sum_{k=1}^{n4} v_{jk} f_3(\sum_{i=1}^{n3} v_{ki} f_2(\sum_{o=1}^{n2} v_{io} f_1(\sum_{p=1}^{n1} v_{op} x_p + v_{o0}) + v_{i0}) + v_{k0}) + v_{j0}) + v_{l0})$$

where:

f_1, f_2, f_3, f_4, f_5: are the activation functions for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.

$n1, n2, n3, n4, n5$: are the number of input signals for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.

L : is the number of outputs for layer 5.

$v_{n2n1}, v_{n3n2}, v_{n4n3}, v_{n5n4}, v_{Ln5}$: are the input weights for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.

$v_{n20}, v_{n30}, v_{n40}, v_{n50}, v_{L0}$: are the bias weights for layer 1, layer 2, layer 3, layer 4, and layer 5 respectively.

$l = 1,2,...,L.$

(4.5)

A refinement can be achieved by defining the augmented matrix of layers weights as:

$$V_{n2n1}^T = \begin{bmatrix} v_{10} & v_{11} & v_{12} & \cdots & v_{1n1} \\ v_{20} & v_{21} & v_{22} & \cdots & v_{2n1} \\ \vdots & \vdots & \vdots & & \vdots \\ v_{n20} & v_{n21} & v_{n22} & \cdots & v_{n2n1} \end{bmatrix} \tag{4.6}$$

$$V_{n3n2}^T = \begin{bmatrix} v_{10} & v_{11} & v_{12} & \cdots & v_{1n2} \\ v_{20} & v_{21} & v_{22} & \cdots & v_{2n2} \\ \vdots & \vdots & \vdots & & \vdots \\ v_{n30} & v_{n31} & v_{n32} & \cdots & v_{n3n2} \end{bmatrix} \tag{4.7}$$

$$V_{n4n3}^T = \begin{bmatrix} v_{10} & v_{11} & v_{12} & \cdots & v_{1n3} \\ v_{20} & v_{21} & v_{22} & \cdots & v_{2n3} \\ \vdots & \vdots & \vdots & & \vdots \\ v_{n40} & v_{n41} & v_{n42} & \cdots & v_{n4n3} \end{bmatrix} \tag{4.8}$$

$$V_{n5n4}^T = \begin{bmatrix} v_{10} & v_{11} & v_{12} & \cdots & v_{1n4} \\ v_{20} & v_{21} & v_{22} & \cdots & v_{2n4} \\ \vdots & \vdots & \vdots & & \vdots \\ v_{n50} & v_{n51} & v_{n52} & \cdots & v_{n5n4} \end{bmatrix} \tag{4.9}$$

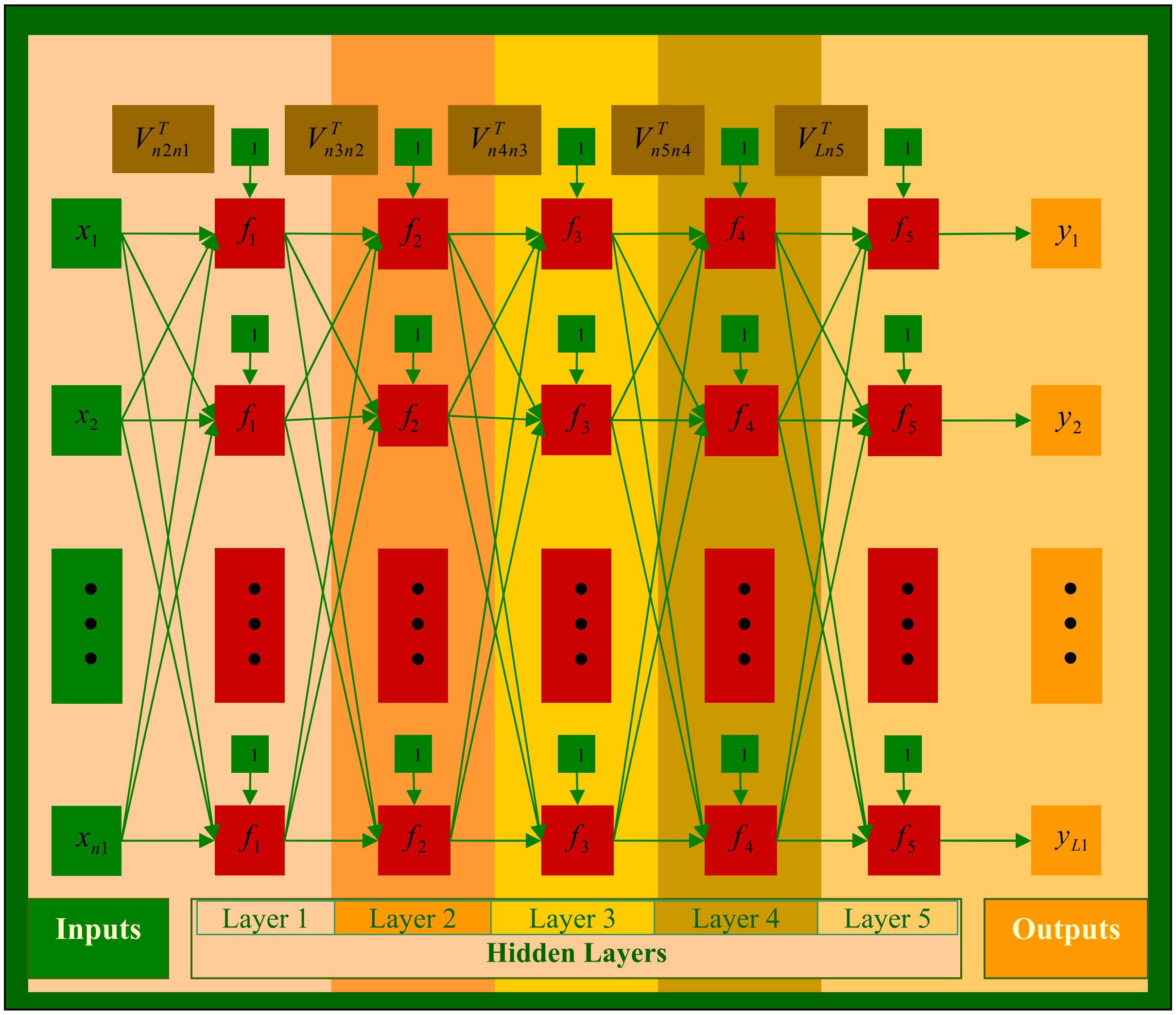


Figure 4.4: The ANN for the developed navigation algorithm.

$$V^T_{Ln5} = \begin{bmatrix} v_{10} & v_{11} & v_{12} & \cdots & v_{1n5} \\ v_{20} & v_{21} & v_{22} & \cdots & v_{2n5} \\ \vdots & \vdots & \vdots & & \vdots \\ v_{L0} & v_{L1} & v_{L2} & \cdots & v_{Ln5} \end{bmatrix} \tag{4.10}$$

Hence, the network can be written as follows:

$$y_l = f_5(V^T_{Ln5} f_4(V^T_{n5n4} f_3(V^T_{n4n3} f_2(V^T_{n3n2} f_1(V^T_{n2n1} x))))), \tag{4.11}$$

$l = 1,2,\ldots, L.$

For the navigation algorithm developed for fixed obstacles, there were three inputs and only one output; the variables of Eq. (4.11) were set as shown in Table 4.1.

Variable	**Description**
y	The robot path, x and y coordinates.
x_1	The DGPS points for the lower boundary of the path
x_2	The DGPS points for the upper boundary of the path
x_3	The position of the fixed obstacles a long the path
f_1	Log-sigmoid, which is: $f(x)=\frac{1}{1+e^{-x}}$
f_2	Log-sigmoid, which is: $f(x)=\frac{1}{1+e^{-x}}$
f_3	Log-sigmoid, which is: $f(x)=\frac{1}{1+e^{-x}}$
f_4	Log-sigmoid, which is: $f(x)=\frac{1}{1+e^{-x}}$
f_5	Linear, which is $f(x)=x$
$n1$	3
$n2$	7
$n3$	12
$n4$	8
$n5$	1
L	1

Table 4.1: The inputs and outputs for the navigation algorithm.

4.3 ANN learning rules

4.3.1 Supervised learning

Supervised learning, or learning with a teacher, or associative learning is where the network is supplied with a training data set that includes a set of inputs and its corresponding output or target, ANN adjusts their weights using one observation at a time. Learning is achieved by minimizing a criterion function.

Two basic rules are known: the error-correction rules and the gradient-descent-based rules [120]. Error- correction rules are ad hoc rules that drive the output error to zero. Among the error correction rules are the perceptron learning rule, the May's rule, the Butz's rule, and the Widrow-Hoff rule (also known as α -LMS rule) [120].

The gradient-descent-based learning rules optimize an appropriate criterion function iteratively by a gradient search procedure. The common gradient-descent-based learning rules are: the μ -Least mean square (LMS), the stochastic μ -LMS rule, the correlation rule, the delta rule, the Minkowski-r delta rule, the relative entropy delta rule, the adaptive Ho-Kashyap (AHK) I, the AHK II, the AHK III , and the delta rule for stochastic units. The criterion function, learning vector, condition, and activation functions are summarized for each of these rules in Table 4.2 according to Hassoun [120]. For more information regarding these learning rules, please refer to [120].

Criterion function[a]	**Learning vector**[b]	**Conditions**	**Activation function**[c]
Perceptron rule			
$J(w) = -\sum_{z^T w \le 0} z^T w$	$\begin{cases} z^k & if\,(z^k)^T w^k \le 0 \\ 0 & otherwise \end{cases}$	$\rho > 0$	$f(net) = \operatorname{sgn}(net)$
Remarks: Finite convergence time if the training set is linearly separable.			

Perceptron rule with variable learning rate and fixed margin			
$J(w) = -\sum_{z^T w \le b} (z^T w - b)$	$\begin{cases} z^k & if\,(z^k)^T w^k \le 0 \\ 0 & otherwise \end{cases}$	$b > 0$ $\rho^k\,satisfies:$ $1.\rho^k \ge 0$ $2.\sum_{k=1}^{\infty} \rho^k = \infty$ $3.\frac{\sum_{k=1}^{\infty} (\rho^k)^2}{\left(\sum_{k=1}^{\infty} \rho^k\right)^2}$	$f(net) = \text{sgn}(net)$
Remarks: Converges to $z^T w > b$ if the training set is linearly separable. Finite convergence if $\rho^k = \rho$, where ρ is a finite positive constant.			
Mays' rule			
$J(w) = \frac{1}{2} \sum_{z^T w \le b} \frac{(z^T w - b)^2}{\lVert z \rVert^2}$	$\begin{cases} \frac{b - (z^k)^T w^k}{\lVert z^k \rVert^2} z^k & if\,(z^k)^T w^k \le b \\ 0 & otherwise \end{cases}$	$0 < \rho < 2$ $b > 0$	$f(net) = \text{sgn}(net)$
Remarks: Finite converges to $z^T w > b > 0$ if the training set is linearly separable.			
Butz's rule			
$J(w) = -\sum_i (z^i)^T w$	$\begin{cases} z^k & if\,(z^k)^T w^k \le b \\ \gamma z^k & otherwise \end{cases}$	$0 \le \gamma \le 1$ $\rho > 0$	$f(net) = \text{sgn}(net)$
Remarks: Finite convergence if the training set is linearly separable. For non separable cases: place w in a region that tends to minimize the error probability.			
Widrow-Hoff rule (α -LMS)			
$J(w) = \frac{1}{2} \sum_i \frac{\left[d^i - (x^i)^T w\right]^2}{\lVert x^i \rVert^2}$	$\left[d^k - (x^k)^T w^k\right] x^k$	$\rho^k = \frac{\alpha}{\lVert x^k \rVert^2}$ $0 < \alpha < 2$	$f(net) = net$

Remarks: Converges in the mean square to the minimum SSE or LMS solution if $\lVert x^i \rVert = \lVert x^j \rVert$ for all *i, j*.			
μ-LMS			
$J(w) = \frac{1}{2}\sum_i \left[d^i - (x^i)^T w\right]^2$	$\left[d^k - (x^k)^T w^k\right] x^k$	$0 < \rho < \frac{2}{3\left\langle \lVert x \rVert^2 \right\rangle}$	$f(net) = net$
Remarks: Converges in the mean square to the minimum SSE or LMS solution.			
Stochastic μ-LMS			
$J(w) = \frac{1}{2}\left\langle \left[d^i - (x^i)^T w\right]^2 \right\rangle$	$\left[d^k - (x^k)^T w^k\right] x^k$	$\rho^k\, satisfies:$ $1. \rho^k \geq 0$ $2. \sum_{k=1}^{\infty} \rho^k = +\infty$ $3. \sum_{k=1}^{\infty} \left(\rho^k\right)^2 < \infty$	$f(net) = net$
Remarks: $\langle . \rangle \equiv$ mean operator. Converges in the mean square to the minimum SSE or LMS solution.			
Correlation rule			
$J(w) = -\sum_i d^i (x^i)^T w$	$d^k x^k$	$\rho > 0$	$f(net) = net$
Remarks: If the vectors x^k are mutually orthonormal it converges to the minimum SSE.			
Delta rule			
$J(w) = \frac{1}{2}\sum_i (d^i - y^i)^2$ $y^i \overset{\Delta}{=} (x^i)^T w$	$(d^k - y^k) f'(net^k) x^k$	$0 < \rho < 1$	$y = f(net)$ f is a sigmoid function

Remarks: Extends the μ-LMS rule to cases with differentiable nonlinear activations.			
Minkowski-*r* delta rule			
$J(w) = \frac{1}{2}\sum_i \left\|d^i - y^i\right\|^r$	$\operatorname{sgn}(d^k - y^k)\left\|d^k - y^k\right\|^{r-1} f'(net^k)x^k$	$0 < \rho < 1$	$y = f(net)$ f is a sigmoid function
Remarks: $1 < r < 2$ for pseudo-Gaussian distribution $p(x)$ with pronounced tails. $r = 2$ gives delta rule. $r = 2$ arises when $p(x)$ is a Laplace distribution.			
Relative entropy delta rule			
$J(w) =$ $\frac{1}{2}\sum_i \left[(1+d^i)\ln\left(\frac{1+d^i}{1+y^i}\right) + (1+d^i)\ln\left(\frac{1-d^i}{1-y^i}\right)\right]$	$\beta(d^k - y^k)x^k$	$0 < \rho < 1$	$y = \tanh(\beta net)$
Remarks: Eliminates the flat spot suffered by delta rule and converges to a linearly separable solution if one exists.			
AHK I			
$J(w,b) = \frac{1}{2}\sum_i \left[(z^i)^T w - b_i\right]^2$ with margin vector *b>0*	Margin: $\Delta b_i = \begin{cases} \rho_1 \varepsilon_i^k & if \varepsilon_i^k > 0 \\ 0 & otherwise \end{cases}$ Weight vector: $\begin{cases} \rho_2(\rho_1 - 1)\varepsilon_i^k z^i & if \varepsilon_i^k > 0 \\ -\rho_2 \varepsilon_i^k z^i & if \varepsilon_i^k \le 0 \end{cases}$ $\varepsilon_i^k = (z^i)^T w^k - b_i^k$	$b^1 > 0$ $0 < \rho_1 < 2$ $0 < \rho_2 < \frac{2}{\max_i \left\|z^i\right\|^2}$	$f(net) = \operatorname{sgn}(net)$

Remarks: b_i values can only increase from their initial values. Converges to a robust solution for linearly separable problems.			
AHK II			
$J(w,b) = \frac{1}{2}\sum_i \left[(z^i)^T w - b_i\right]^2$ with margin vector *b>0*	Margin $\Delta b_i^k = \begin{cases} \rho_1 \varepsilon_i^k & if \varepsilon_i^k > \frac{-b_i^k}{\rho_1} \\ 0 & otherwise \end{cases}$ Weight vector: $\begin{cases} \rho_2(\rho_1 - 1)\varepsilon_i^k z^i & if \varepsilon_i^k > \frac{-b_i^k}{\rho_1} \\ -\rho_2 \varepsilon_i^k z^i & if \varepsilon_i^k \le \frac{-b_i^k}{\rho_1} \end{cases}$ $\varepsilon_i^k = (z^i)^T w^k - b_i^k$	$b^1 > 0$ $0 < \rho_1 < 2$ $0 < \rho_2 < \frac{2}{\max_i \lVert z^i \rVert^2}$	$f(net) = \mathrm{sgn}(net)$
Remarks: b_i values can take any positive values. Converges to a robust solution for linearly separable problems.			
AHK III			
$J(w,b) = \frac{1}{2}\sum_i \left[(z^i)^T w - b_i\right]^2$ with margin vector *b>0*	Margin $\Delta b_i^k = \begin{cases} \rho_1 \varepsilon_i^k & if \varepsilon_i^k > \frac{-b_i^k}{\rho_1} \\ 0 & otherwise \end{cases}$ Weight vector: $\begin{cases} \rho_2(\rho_1 - 1)\varepsilon_i^k z^i & if \varepsilon_i^k > \frac{-b_i^k}{\rho_1} \\ 0 & otherwise \end{cases}$ $\varepsilon_i^k = (z^i)^T w^k - b_i^k$	$b^1 > 0$ $0 < \rho_1 < 2$ $0 < \rho_2 < \frac{2}{\max_i \lVert z^i \rVert^2}$	$f(net) = \mathrm{sgn}(net)$
Remarks: Converges for linearly and nonlinearly separable cases.			

Delta rule for stochastic units			
$J(w)=\frac{1}{2}\sum_i (d^i - \langle y^i \rangle)^2$	$\beta\left[d^k - \tanh(\beta net^k)\right]$ $\left[1-\tanh^2(\beta net^k)\right]x^k$	$0<\rho<1$	Stochastic activation: $y=\begin{cases}+1 & with \quad P(y=+1)\\ 1 & with \quad 1-P(y=+1)\end{cases}$ $P(y=1)=$ $\frac{1}{1+e^{-2\beta net}}$
Remarks: Performance in average is equivalent to the delta rule applied to a unit with deterministic activation: $f(net)=\tanh(\beta net)$			

Comments:

a. Note: $z^k = \begin{cases} x^k & if \quad d^k = +1 \\ -x^k & if \quad d^k = -1 \end{cases}$

b. The general form of the learning equation $w^{k+1} = w^k + \rho^k s^k$, where ρ^k is the learning rate and s^k is the learning vector.

c. $net = x^T w$

Symbols:

b : Margin vector.

d^k : Desired target vector associated with input vector x.

$f(net)$: Differentiable activation function, usually a sigmoid function.

i^* : Label of a wining neuron in a competitive net.

$J(w)$: Objective (criterion) function.

net : Weigted-sum or $w^T x$.

$p(x)$: Probability density function of x.

T : Threshold constant.

w : Weight vector
w_i^* : Weight vector of wining unit *i*.
w^* : Solution weight vector
y : Unit (neuron) output.
α : Momentum rate.
ε : Positive constant, usually arbitrary small.
ε^k : Error vector at the kth step of the AHK algorithm
ρ : Learning rate.

Table 4.2: ANN supervised learning rules [120].

4.3.2 Reinforcement learning

The learning process in reinforcement learning is designed to maximize the expected value of a criterion function by using trial and error. That is if the action improves the state of affairs then the tendency to produce this action is reinforced while if it deteriorate the state of affairs, then the tendency to produce it is weakened [120].

The training data does not includes a target; instead, it includes a performance judge or a utility function that reports how good the current network output is [122]. This serves as an evaluative signal or a critic that is not associated with the inputs but informs the unit being trained about the appropriateness of the output i.e., the unit performance due to certain input. However, it gives no indication about what the output should be for this input compared to the case of supervised learning [120].

Barto and Anandan (1985) [as cited in 120] developed a reinforcement learning rule called associative reward-penalty algorithm, the criterion function, learning vector conditions, and activation function for this rule is summarized in Table 4.3.

Criterion function[a]	Learning vector[b]	Conditions	Activation function[c]
$J(w) = -\langle r \rangle$	$r^k(y^k - \langle y^k \rangle)x^k$	$\rho > 0$ $r^k =$ $\begin{cases} +1 & reward \\ -1 & penalize \end{cases}$	Stochastic activation: $y = \begin{cases} +1 & with & P(y=+1) \\ 1 & with & 1-P(y=+1) \end{cases}$ $P(y=1) = \frac{1}{1+e^{-2\beta net}}$
Remarks: w^k evolves so as to maximize the average reinforcement signal $\langle r \rangle$. a. The general form of the learning equation $w^{k+1} = w^k + \rho^k s^k$, where ρ^k is the learning rate and s^k is the learning vector. b. $net = x^T w$.			
Symbols: r : Positive vector of a unit in a neural field. $r^k, r(y, x^k)$: Reinforcement signal.			

Table 4.3: Associative reward-penalty reinforcement learning rule [120].

4.3.3 Unsupervised learning

In this case the network is adapted without giving it any kind of directive feed back, in other words there are no target information in the training data or a performance judge. The unsupervised learning objective is to find out the features inherent in the training data [122-125].

There are three main categories of unsupervised learning [122]:

- Associative memories or autoassociative memories networks: These networks reconstruct the entire pattern when provided with part of the pattern that had been seen before.
- Self-organization maps (SOM) or competitive learning: which are networks designed to extract features and learn to categorize the input vectors presented to it. SOM can also

learn the topology of their input vectors. Competitive networks learn the inputs distribution by dedicating more neurons to classify certain input parts that have higher densities. SOM allow neighbor neurons to the winning neuron to output values so that the transition of output vectors is smoother than the one obtained from competitive layers, where only one neuron has an output at a time [126]. SOM map a signal pattern of arbitrary dimensionality onto a one-dimensional or two-dimensional array, where the input array is replaced by a weighted sum of primary signals and every unit has its own adaptive input weights. Generally, the unit array receives its input signals through a replying network [122].

- System identification networks: Are networks designed to model the dynamic model for the variables they observe.

Accordingly, three classes of unsupervised learning rules had been developed. The first class is the Hebbian learning rules that includes the Hebbian rule, the Oja's rule, the Yuille, et al. rule, the Linsker's rule, and the Hassoun's rule.

The second class considers the competitive learning rules that can handle clustering of unlabeled data or vector quantization cases, where a binary-valued output network is used to categories the input by considering the correlations in the input data. This class includes the standard competitive learning rule.

The third class is the Kohonen's self-organization feature map rule that captures the topology and probability distribution of the input data [120]. The criterion function, learning vector conditions, and activation function for all the unsupervised learning rules are presented in Table 4.4. For more information regarding theses learning rules please refer to [120].

Criterion function[a]	**Learning vector**[b]	**Conditions**	**Activation function**[c]

Hebbian rule			
$J(w) = -\frac{1}{2}\left\langle y^2 \right\rangle$	$y^k x^k$	$\rho \ge 0$	$f(net) = net$
Remarks: $\lVert w^* \rVert \to \infty$ with w^* pointing in the direction of $c^{(1)}$ (see comment d).			
Oja's rule			
Does not have an exact $J(w)$. However, this rule tends to maximize $\left\langle y^2 \right\rangle$ subject to the constraint $\lVert w \rVert = 1$.	$y^k x^k - (y^k)^2 w^k$	$0 < \rho < \frac{1}{\lambda_{\max}}$	$f(net) = net$
Remarks: Converges in the mean to $w^* = c^{(1)}$ which maximizes $\left\langle y^2 \right\rangle$			
Yuille et al. rule			
$J(w) = -\frac{1}{2}\left\langle y^2 \right\rangle + \frac{\lVert w \rVert^4}{4}$	$y^k x^k - \lVert w^k \rVert^2 w^k$	$0 < \rho < \frac{1}{\lambda_{\max}}$	$f(net) = net$
Remarks: Converges to $\lVert w^* \rVert = \lambda_{\max}$ with w^* in the direction of $c^{(1)}$. w^* maximizes $\left\langle y^2 \right\rangle$.			
Linsker's rule			
$J(w) = -\frac{1}{2}\left\langle y^2 \right\rangle + \frac{\lambda}{2}\left(1 - \sum_j w_j\right)^2$	$y^k x^k + \lambda(1 - \sum_j w_j^k)1$	$\sum_j w_j = 1$ $0 < \rho << 1$ $w^- \le w_i^k \le w^+$ $\lambda > 0$	$f(net) = net$

Remarks: Maximizes $\langle y^2 \rangle$ subject to $\sum_j w_j = 1$. Converges to w^* whose components are clamped at w^- or w^+, when λ is large.			
Hassoun's rule			
$J(w) = -\frac{1}{2}\langle y^2 \rangle + \frac{\lambda}{2}(1 - \lVert w \rVert)^2$	$y^k x^k - \lambda w^k (1 - \frac{1}{\lVert w^k \rVert})$ For $\rho\lambda = 1$ this rule reduces to: $w^{k+1} = \frac{w^k}{\lVert w^k \rVert} + \rho y^k x^k$	$\lambda >> \lambda_{\max}$ $0 < \rho \le \frac{2}{\lambda}$	$f(net) = net$
Remarks: Converges in the mean to $\lVert w^* \rVert = \frac{\lambda}{\lambda - \lambda_{\max}}$ with w^* parallel to $c^{(1)}$. For $\lambda >> \lambda_{\max}$, w^+ approaches $c^{(1)}$.			
Standard competitive learning rule			
$J(w) = \frac{1}{2}\sum_k \lVert x^k - w_{i*} \rVert^2$ i^* is the index of the wining unit: $w_{i*}^T x^k \ge w_j^T x^k$	$x^k - w_{i*}$	ρ^k satisfies: 1.$\rho^k \ge 0$ 2.$\sum_{k=1}^{\infty} (\rho^k)^2 < \infty$ 3.$\sum_{k=1}^{\infty} \rho^k = \infty$	$f(net) = net$
Remarks: Converges to a local minimum of J representing some clustering configuration.			
Kohonen's feature map rule			
$J(w) = \frac{1}{2}\sum_{k,i} \Phi(r, r_{i*}) \lVert x^k - w_i \rVert^2$ e.g.,	$\Phi(r, r_{i*})(x^k - w_i)$	ρ^k and σ^k evolves according to: $k^{-\alpha}$ with	$f(net) = net$

$\Phi(r, r_{i^*}) = e^{-\frac{\lVert r_i - r_{i^*}\rVert^2}{2\sigma^2}}$		$0 < \alpha \le 1$, or $a\left(\frac{b}{a}\right)^{k/k_{\max}}$ with $0 < b < a < 1$	
Remarks: The weight vectors evolve to a solution which tends to preserve the topology of the input space. The local point density of this solution is of the form $g(p(x))$, where g is a continuous monotonically increasing function and $p(x)$ is a stationary probability density function governing x^k.			
a. Note: $z^k = \begin{cases} x^k & if \quad d^k = +1 \\ -x^k & if \quad d^k = -1 \end{cases}$ b. The general form of the learning equation $w^{k+1} = w^k + \rho^k s^k$, where ρ^k is the learning rate and s^k is the learning vector. c. $net = x^T w$ d. $c^{(i)}$ is the ith normalized eigenvector of the autocorrelation matrix C with a corresponding eigenvalue $\lambda_i (\lambda_1 \ge \lambda_2 \ge ... \ge \lambda_n \ge 0$ and $\lambda_{\max} = \lambda_1)$.			
Symbols: i^* : Label of a wining neuron in a competitive net. $\Phi(r, r^*)$: Continuous neighborhood function with center at r^*.			

Table 4.4: ANN unsupervised learning rules [120].

Chapter 5: The navigation system

5.1 Introduction

The navigation system is responsible for guiding the robot. The navigation system should extract a representation of the world from the moving images and other sensory information received. Several capabilities are required for autonomous navigation. Over the years, all kinds of technologies were tried to simplify the task but each one had it's own disadvantages regarding working in unstructured environments [21]:

Landmarks: It is subject to movement by environmental factors.

Dead Reckoning: The accuracy of it depends on measurement tools. The errors accumulate quickly.

Celestial: Only works in nights in good weather and has limited precision.

OMEGA: It is based on relatively few radio direction beacons. It also has limited accuracy and subject to radio interference.

LORAN: Has a limited coverage (mostly coastal), variable accuracy, gets affected by geographic situation, and it is easy to jam or disturb.

SatNav: It is based on low-frequency doppler measurements so it is sensitive to small movements at the receiver. In addition, there are only a few satellites so updates are infrequent.

The navigation system that has been developed here is suitable for unstructured outdoor environment, which is usually an open outdoor wild terrain with no landmarks and some stationary or moving obstacles. Hence, the above methods cannot be used. On the other hand, GPS works in any open area; thus, DGPS is suggested to be used for the navigation system.

It is suggested to supply the robot with treads, which makes it capable of navigating through any open wild terrain.

In this chapter, the navigation system and navigation algorithm will be presented, followed by an overview of the developed simulation software and the results of running the navigation algorithm.

5.2 The sensory devices

Point or line tracking can be achieved through the use of a vision system consists of digital Charge Coupled Device (CCD) cameras and an automatic video tracking device for image processing. This system enables the robot to navigate through the landmarks.

Just as with a human being, the robot may have some obstacles on its path while navigating. In order for the robot to reach its target safely, it needs to sense these obstacles and avoid them. Obstacle avoidance is one of the key problems in computer vision and mobile robots. There has been a considerable research devoted to the obstacle detection problem for mobile robot platforms and intelligent vehicles.

Laser scanners have been used for many years for obstacle detection and are found to be most reliable and provide accurate results. They operate by sweeping a laser beam across a scene and at each angle, measuring the range and returned intensity. The laser provides fast single-line laser scans and is used to map the location and size of possible obstacles. With these inputs, the central control system can control the steering speed of the robot on an obstacle course [127].

The proposed system is designed to perform mainly with fixed obstacles, for the case of moving obstacles, which are assumed not so many; the robot will stop and wait until these obstacles will free the robot path. If these obstacles stop and remain stand still the robot will avoid them using the same algorithm it uses for a fixed obstacle.

The robot needs to be supplied with a DGPS receiver in order to make it capable to navigate in the places where there are no landmarks.

5.3 Navigation algorithm

During navigation in an unstructured environment there are two possible cases: there are landmarks or there are no landmarks. There may be some stationary obstacles and a low possibility for moving obstacles.

For the first case, the robot will find its path using the digital CCD camera and the image processing will be done by an automatic video tracking device. For the second case, which is the common case in unstructured outdoor environment, the robot will use the DGPS receiver to capture information about its path. DGPS receiver position information will be used to plan the robot path. The developed navigation algorithm first connects the DGPS receiver points for the path between the starting point and the end point. Then subtracts half the amount of the road width to find out the lower boundary of the road. The road upper boundary is calculated by adding half the amount of the road width to the DGPS path. ANN is then used to find the best path for the robot.

ANN was used to give the robot some intelligence and learning capabilities, that, when applying different roads to the robot, the algorithm will find the suitable path without being trained.

The goal of the navigation algorithm is to find the best path that satisfies the following constrains:

1. Falls within the road boundaries.
2. Avoid any obstacle along the path.

Simulation software was developed to test different navigation scenarios and architectures. The simulation was based on the kinematics of the robot.

5.3.1 ANN architecture

5.3.1.1 ANN input

The inputs to the ANN are:

1. The lower boundary of the road.
2. The upper boundary of the road.
3. The position of the fixed obstacles along the road collected from the laser scanner.

5.3.1.2 ANN output

The output of the ANN will be the robot path that needs to be within the lower and upper road boundaries and avoids any obstacles.

5.3.1.3 ANN layers architecture

The architecture for ANN was selected based on simple rules of thump and trial and error. Trials were conducted on simulation software developed using Matlab for robot navigation. The ANN used was a feedforward backpropagation network that consists of five layers, the layers architecture are shown in Table 5.1.

Layer	**Number of neurons**	**Activation function**
First layer	3	Log-sigmoid
Second layer	7	Log-sigmoid
Third layer	12	Log-sigmoid
Fourth layer	8	Log-sigmoid
Fifth layer (Output layer)	1	Linear

Table 5.1: ANN layers architecture.

5.3.2 ANN training

The network was trained using backpropagation training algorithm. Backpropagation was developed by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks. The backpropagation training algorithm used was a Broyden, Fletcher, Goldfarb, and Shanno (BFGS) quasi-Newton algorithm, which is a high performance algorithm that can converge from ten to one hundred times faster than the common gradient descent algorithms. The Newton's method is an alternative to the conjugate gradient methods for fast optimization; it uses standard numerical optimization techniques [126].

The basic step of Newton's method uses the Hessian matrix (second derivatives) of the performance index at the current values of weights and biases. Newton's method often converges faster than conjugate gradient methods. Unfortunately, it is complex and expensive to compute the Hessian matrix for feedforward neural networks. There is a class of algorithms that is based on Newton's method, but which doesn't require calculation of the second derivative. These are called quasi-Newton (or secant) methods. They update an approximate Hessian matrix at each iteration of the algorithm. The update is computed as a function of the gradient. The quasi-Newton method that has been most successful in published studies is the Broyden, Fletcher, Goldfarb, and Shanno (BFGS) update [126]. This algorithm has been implemented in this algorithm. The performance function used is the mean square error (MSE). MSE is the average squared error between the network outputs and the target. The weights and biases of the network were automatically initialized to small random numbers by the software.

The training data was a simulated road boundary that was developed using typical DGPS receiver data. Two sets of experiment were tried. In the first set six obstacles were imposed to the road, while seventeen obstacles were used in the second set of experiments.

The target used for training the network is developed based on the judgment of an experienced driver on how to navigate the road without hitting any of the obstacles. The road boundaries are represented by lines, while the obstacles are represented by circles. The robot path is represented by a line, which represents the position of the robot center of gravity and a distance of half the width of the robot. Some clearance always need to be kept between the robot path and the obstacles or the road boundary during navigation. The road consists of 1000 points. The simulated road boundary with the two sets of obstacles and the target are shown in Fig. 5.1 and Fig. 5.2.

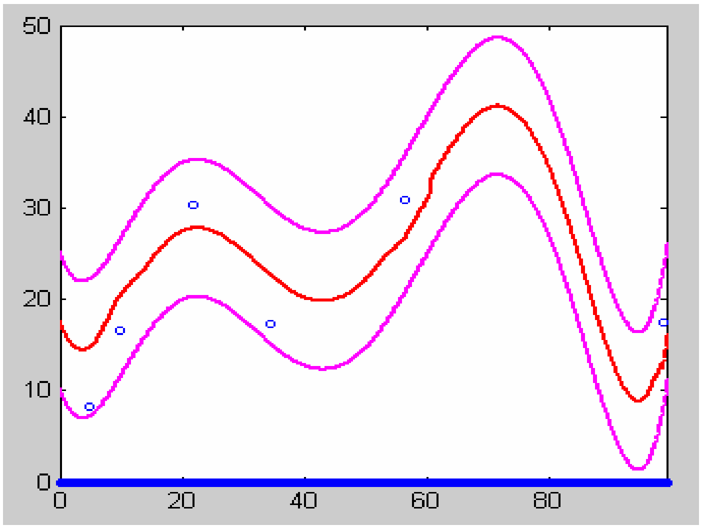


Figure 5.1: The simulated road boundary with the first set of obstacles (six obstacles) and the target.

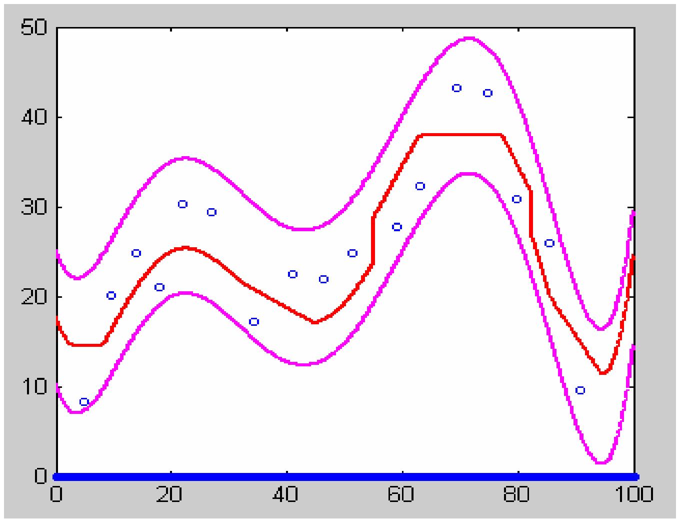


Figure 5.2: The simulated road boundary with the second set of obstacles (seventeen obstacles) and the target.

The network trained in a very small time and was able to produce the target with a MSE of 1.33 $*10^{-4}$ in the first case and $4.86*10^{-4}$ in the second case. The training performance is shown in Fig. 5.3 and Fig. 5.4 respectively.

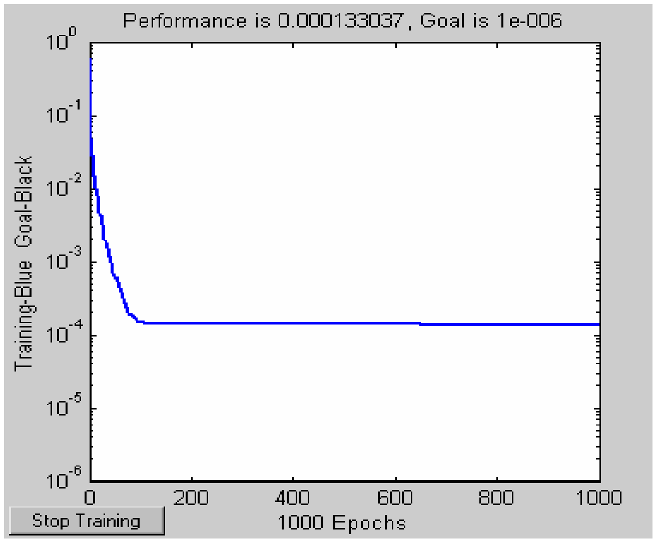


Figure 5.3: The training performance of the network versus epochs[11] for the road with six obstacles.

[11] An epoch is defined as one complete run through all input/output pairs [126].

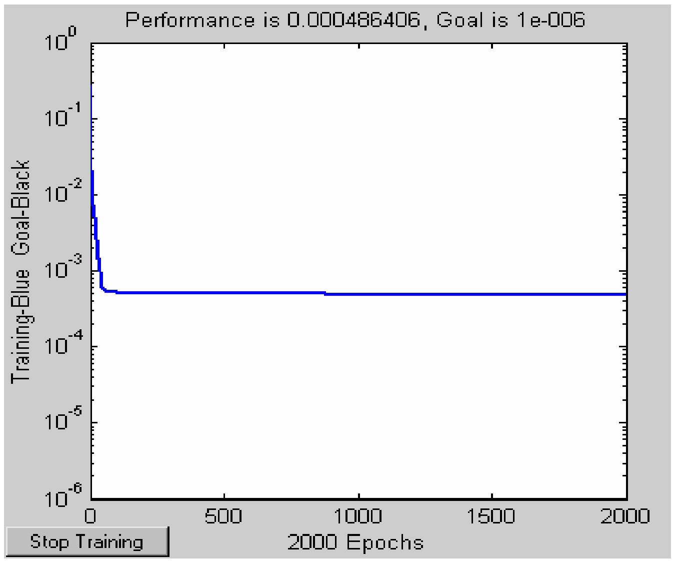


Figure 5.4: The training performance of the network versus epochs for the road with seventeen obstacles.

5.3.3 Results

After training the network, the same road applied to the network, the output from the network is shown in Fig. 5.5 and Fig. 5.6 respectively.

As shown in the figures the network was able to produce a good path that is within the road boundaries and avoiding all the obstacles, actually, in the second case the network output is even smoother than the target.

To test the learning capability of the network, two different roads were given to the network; one with few obstacles and another one with many obstacles. The roads are shown in Fig. 5.7 and Fig. 5.8 respectively.

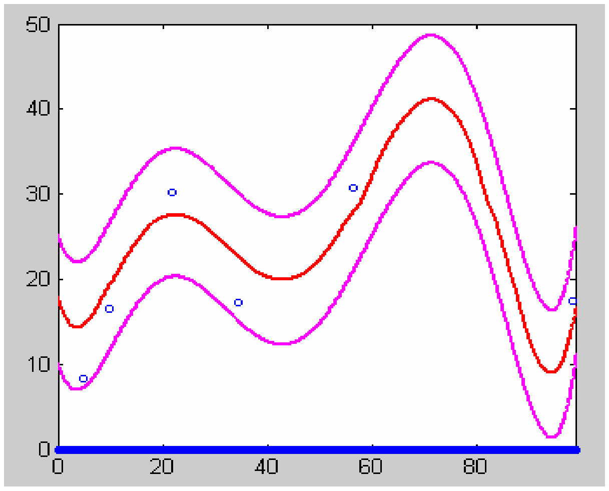


Figure 5.5: The output of the network for the original road with six obstacles.

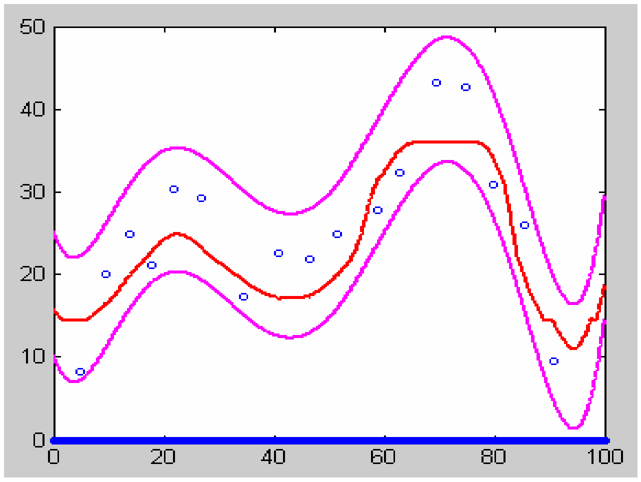


Figure 5.6: The output of the network for the original road with seventeen obstacles.

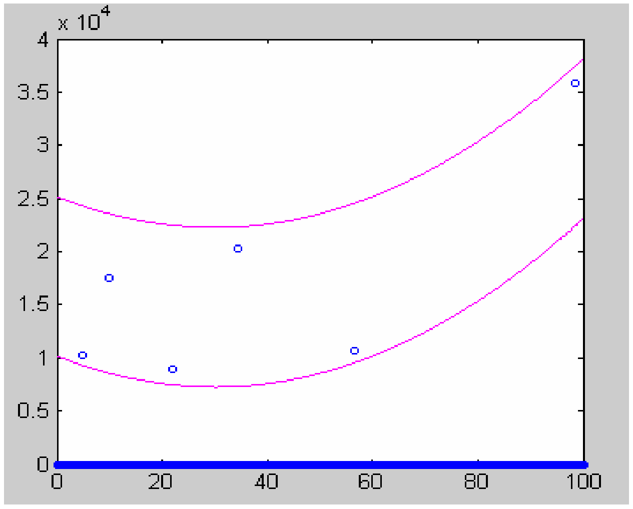


Figure 5.7: The second road boundary applied to the network.

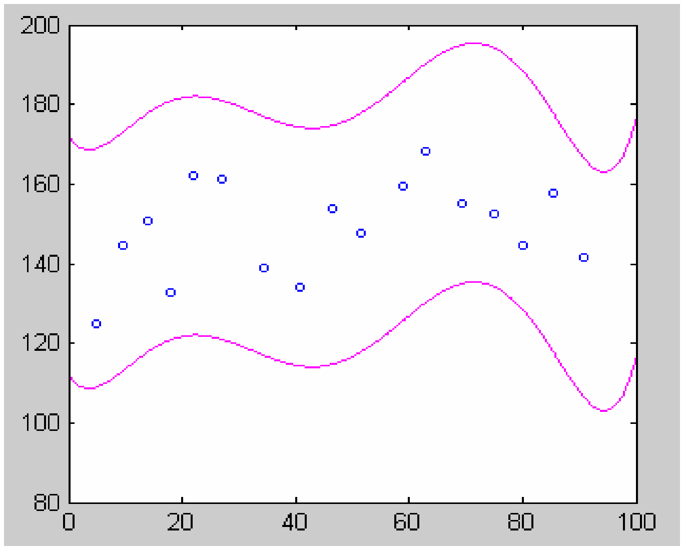


Figure 5.8: The third road boundary applied to the network.

The output of the network for these roads are shown in Fig 5.9 and Fig. 5.10 respectively.

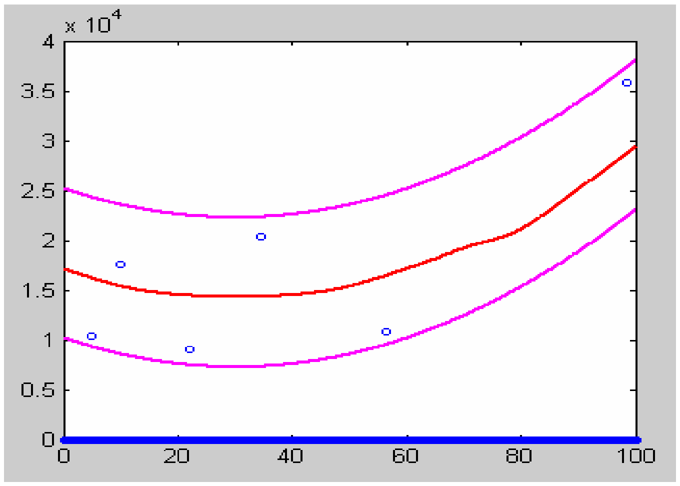


Figure 5.9: The output of the network for the second road boundary.

As shown in the figures the network performance was excellent for the second road where a few number of obstacles are present. For the third road, the network succeeds in finding a good path but it hits one of the obstacles.

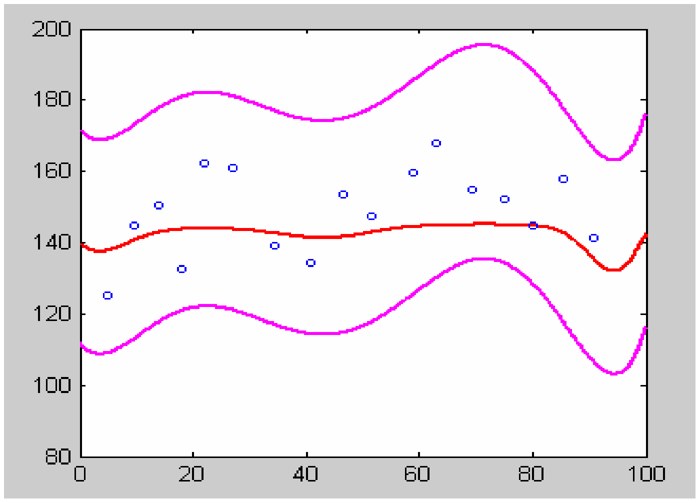


Figure 5.10: The output of the network for the third road boundary.

5.3.4 Conclusion

In this chapter the development of an autonomous navigation system suitable for robot navigation in outdoor unstructured environment is presented. It is suggested that the navigation system uses a vision system consists of digital CCD cameras and a video tracking device. In cases where there are no landmarks the robot is suggested to have a DGPS receiver. A laser scanner is suggested to be used to detect the presence of an obstacle.

The navigation algorithm was developed using a feedforward neural network with five layers; the network was trained using quasi-Newton backpropagation algorithm. Software has been developed to simulate the performance and efficiency of the algorithm. The network was trained on road with less number of obstacles and on another road with many obstacles and the network was able to produce the output within a MSE of $1.33 *10^{-4}$ and $4.86*10^{-4}$ from the target that was developed using an experienced driver. The network has been tested under different kind of roads and obstacles and produced very good results on paths with a small number of obstacles; for a very curved paths with many obstacles the network had some difficulties in dealing with the obstacles in some parts of the path. However, this is acceptable even when compared to experienced human drivers.

Supervised learning was used since for this particular application, where the training data includes the inputs and the target, neither the unsupervised learning nor the reinforcement learning can produce better results than the supervised learning. In other cases where the training data does not include a target; instead, it includes a performance judge or a utility function it is better to use the reinforcement learning. In the pattern recognition, features extraction, or input categorization cases it will be more appropriate to use unsupervised learning as was mentioned in Chapter 4.

Chapter 6: Dynamic simulation for real time motion control of the robot

6.1 Introduction

Robots and robot manipulators have complex nonlinear dynamics that make their accurate and robust control difficult. On the other hand, they fall in the class of Lagrangian dynamical systems, so that they have several extremely good physical properties that make their control straight forward [80].

This chapter presents the development of three major types of controllers. The first type is the Computed-Torque (CT) controller, from which a PD CT controller and a PID CT controller are developed. The second type of controllers, developed in this chapter, is the digital controller. The third type is the filtered-error approximation-based controller, based on approximation of the unknown nonlinear functions that can be used to derive adaptive, robust, and learning controllers.

A dynamic simulation is conducted from a framework developed by Lewis, et al. [80] for two case studies. The first case study is the two-link robot manipulator, and the second is a navigation controller for the WMR, which takes as input the desired robot path from the navigation algorithm described in chapter 5 and produces the suitable control torques. The simulation is developed by using Matlab and C++.

The Lewis framework was formulated for a trajectory of robot manipulators. However, this research is oriented toward robot navigation. Hence, the Lewis framework is adjusted, as needed, to control the robot path, instead of the manipulators trajectories.

6.2 CT controllers

The dynamics of the robot can be formulated by:

$$M(q)\ddot{q} + J(q,\dot{q})\dot{q} + F = \tau \tag{6.1}$$

In Eq. (6.1), $M, J, F, and \tau$ were previously defined in equation (3.62) or (3.67), where q equals ξ in equation (3.62) or (3.67). The dynamics of the robot manipulator can be formulated by [80]:

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + R(\dot{q}) + G(q) + \tau_d = \tau \tag{6.2}$$

where:

- $M(q)$: is the inertia matrix;
- $V_m(q,\dot{q})$: is the Coriolis/centripetal matrix;
- $R(\dot{q})$: is the friction term;
- $G(q)$: is the gravity vector;
- τ_d: is the torque from the disturbances;
- τ: is the control input torque.

The similarities between Eqs. (6.1) and (6.2) make it possible to derive a CT controller suitable for both cases. The general rules for different types of CT controllers are developed and followed by customizing these rules for a WMR motion control and a two-link robot manipulator.

Eqs. (6.1) and (6.2) can be written as follows [80]:

$$M(q)\ddot{q} + N(q,\dot{q}) = \tau \tag{6.3}$$

or, in the case of the existence of unknown disturbances τ_d:

$$M(q)\ddot{q} + N(q,\dot{q}) + \tau_d = \tau$$

where $N(q,\dot{q})$ represents the nonlinear terms.

The objective of a motion controller is to move the robot or the robot manipulator according to the desired motion trajectory $q_d(t)$. The actual motion trajectory is defined as $q(t)$. The tracking error, in this case, can be defined as [80]:

$$e(t) = q_d(t) - q(t) \tag{6.4}$$

The Brunovsky canonical form can be developed by differentiating $e(t)$ twice and writing it in the terms of the state x [80]:

$$\frac{d}{dt}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix}\begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} u \tag{6.5}$$

where:

$$u \equiv \ddot{q}_d + M^{-1}(q)(N(q,\dot{q}) - \tau)$$

$$x = \begin{bmatrix} e^T \\ \dot{e}^T \end{bmatrix}$$

To develop the CT controller, a linear system design procedure is used to select the feedback control signal $u(t)$, which stabilizes the tracking error equation. Then the torques needed for the motors are computed by using the inverse of the dynamic equation for the WMR or the two-link manipulator [80]:

$$\tau = M(q)(\ddot{q}_d - u) + N(q,\dot{q}) \tag{6.6}$$

Eq. (6.6) represents a nonlinear feedback control law, which guarantees tracking the desired motion trajectory $q_d(t)$. Eq. (6.6) is termed feedback linearization, based on computing the motor torques making the nonlinear dynamic systems (Eq. (6.1) and (6.2)) equivalent to the linear dynamics in Eq. (6.5). Two types of CT controllers are developed here, namely, the PD CT controller and the PID CT controller.

6.2.1 PD CT controller

A PD feedback for $u(t)$, with a derivative gain matrix K_v and a proportional gain matrix K_p, produces the PD CT controller, where the motor torques equal [80]:

$$\tau = M(q)(\ddot{q}_d + K_v\dot{e} + K_p e) + N(q,\dot{q}) \tag{6.7}$$

which has the tracking error dynamics $\ddot{e} = -K_v\dot{e} - K_p e$

The gain matrices need to be selected positive definite to keep the tracking error dynamics stable.

6.2.2 PID CT controller

The PD CT controller can easily be adjusted to a PID CT controller by adding an integrator gain matrix K_i to $u(t)$, as follows [80]:

$$\tau = M(q)(\ddot{q}_d + K_v\dot{e} + K_p e + K_i(\int e)) + N(q,\dot{q}) \tag{6.8}$$

which has the tracking error dynamics $\ddot{e} = -K_v\dot{e} - K_p e$

The integrator gain cannot be too large, in order to keep the tracking error stable.

The PD CT and PID CT controllers are illustrated in Fig. 6.1, where $K = [K_p \quad K_v]$ for a PD CT controller, and $K = [K_p \quad K_v \quad K_i]$ for a PID CT controller. As shown in Fig. 6.1, these controllers have an outer tracking loop and an inner feedback linearization loop, which consists of the nonlinear component $N(q,\dot{q})$.

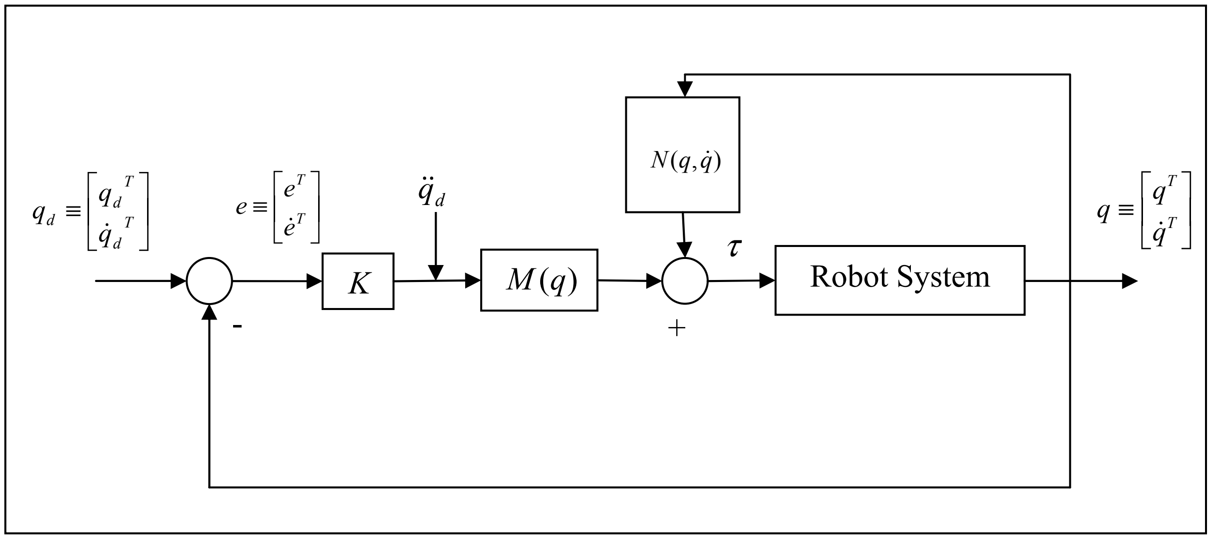

$q_d \equiv \begin{bmatrix} q_d^T \\ \dot{q}_d^T \end{bmatrix}$
$e \equiv \begin{bmatrix} e^T \\ \dot{e}^T \end{bmatrix}$
$\ddot{q}_d$
$N(q,\dot{q})$
$q \equiv \begin{bmatrix} q^T \\ \dot{q}^T \end{bmatrix}$
K
$M(q)$
τ
Robot System
-
+

Figure 6.1: PD and PID CT controllers block diagram [80].

6.2.3 Simulation for the PD CT controller for the two-link manipulator

6.2.3.1 Simulation architecture

This controller function is to select the suitable motor torques, so that the robot manipulator having two-links will follow the desired motion trajectory $q_d(t)$. The desired motion trajectory is already available from the description of the task the manipulator needs to perform. The two-link manipulator is shown in Fig. 6.2:

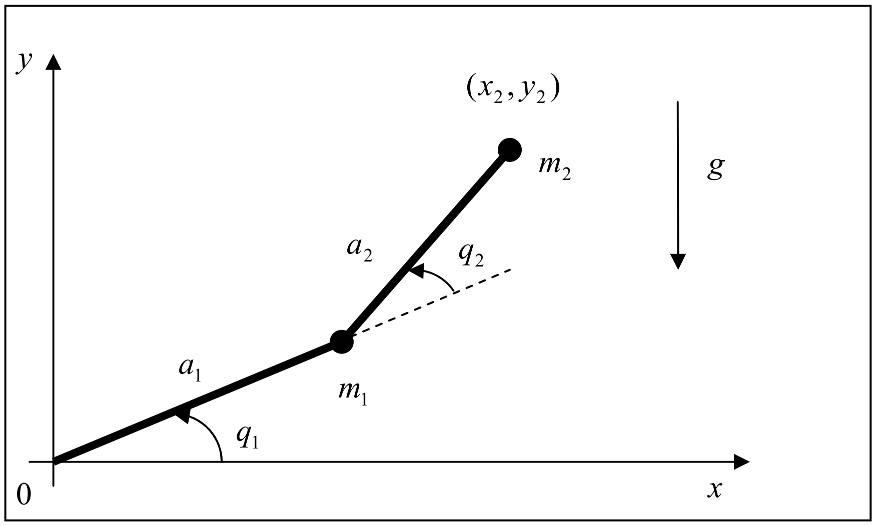


Figure 6.2: Two-link manipulator [80].

Lewis, et al. [80] developed the two-link manipulator dynamics as follows:

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau \tag{6.9}$$

where:

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}$$

$$M(q) = \begin{bmatrix} (m_1 + m_2)a_1^2 + m_2 a_2^2 + 2m_2 a_1 a_2 \cos q_2 & m_2 a_2^2 + m_2 a_1 a_2 \cos q_2 \\ m_2 a_2^2 + m_2 a_1 a_2 \cos q_2 & m_2 a_2^2 \end{bmatrix}$$

$$V(q,\dot{q}) = \begin{bmatrix} -m_2 a_1 a_2 (2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2) \sin q_2 \\ m_2 a_1 a_2 \dot{q}_1^2 \sin q_2 \end{bmatrix}$$

$$G(q) = \begin{bmatrix} (m_1 + m_2) g a_1 \cos q_1 + m_2 g a_2 \cos(q_1 + q_2) \\ m_2 g a_2 \cos(q_1 + q_2) \end{bmatrix}$$

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix}$$

q_1 : is the actual orientation of the first link of the manipulator;

q_2 : is the actual orientation of the second link of the manipulator;

m_1 : is the mass of the first link of the manipulator;

m_2 : is the mass of the second link of the manipulator;

τ_1 : is the torque of the first link of the manipulator;

τ_2 : is the torque of the second link of the manipulator.

The two-link manipulator dynamic model can be written, as illustrated in Eq. (6.3), in terms of linear and nonlinear components:

$$M(q)\ddot{q} + N(q,\dot{q}) = \tau \tag{6.10}$$

where $N(q,\dot{q})$ represents the nonlinear terms:

$$N(q,\dot{q}) \equiv V(q,\dot{q})\dot{q} + G(q)$$

The simulation program developed in this section has the following main components:

- The first component computes the desired manipulator trajectory from the input of the path planner or the design engineer. The desired trajectory $q_d(t)$ is given as follows:

$$q_d = \begin{bmatrix} q_{1d} \\ q_{2d} \end{bmatrix} \tag{6.11}$$

where:

q_{1d} : is the desired orientation of the first link of the manipulator;

q_{2d} : is the desired orientation of the second link of the manipulator.

- The second component calculates the controller input by: (1) calculating the tracking error between the desired trajectory $q_d(t)$ and the actual trajectory $q(t)$, and (2) computing the inertia term $M(q)$ and the nonlinear term $N(q,\dot{q})$, based on the dynamic model described in Eq. (6.9). Finally, the motor torques are calculated by using Eq. (6.7).
- The third component calculates the new position of the each link of the manipulator by using the state-space equation $\dot{x} = f(x,u)$, where the state-space position/velocity form is used [80]:

$$x \equiv \begin{bmatrix} q^T \\ \dot{q}^T \end{bmatrix} \tag{6.12}$$

$$\dot{x} = \begin{bmatrix} \dot{q} \\ -M^{-1}(q)N(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}(q) \end{bmatrix}\tau$$

The state-space form is used to update the position of the each link of the manipulator.

In adapting this controller to different types of manipulators, one needs to: (1) replace the dynamic model of the two-link manipulator with that of the new manipulator, (2) define the parameters of the desired path for the new manipulator, and (3) adjust the controller equations accordingly.

The inputs to the PD CT controller simulation program are:

1. The desired motion trajectory $q_d(t)$;
2. Manipulator parameters;
3. Controller parameters k_p and k_v.

The outputs of the PD CT controller simulation program are:

1. The motor torques;
2. The actual path $q(t)$.

6.2.3.2 Simulation results

Two experiments are conducted here: (1) without any torque disturbances and (2) with constant unknown torque disturbances. The results of the two experiments will be demonstrated in this section.

The simulation parameters for the first experiment are shown in the following table:

Arm parameters	
m_1	1 kg
m_2	1 kg
a_1	1 m
a_2	1 m
Desired trajectory	

$q_{d1}(t)$	$0.1\sin t$
$q_{d2}(t)$	$0.1\cos t$
Controller parameters	
k_p	100
k_v	20
Note:	m_1, m_2, a_1, and a_2 are according to Fig. 6.2.

Table 6.1: Simulation parameters for a PD CT controller.

The results of this experiment are shown in Figs. 6.3-6.4. Fig. 6.3 shows the tracking errors for the joints, while Fig. 6.4 shows the desired versus the actual joint angles. As shown in both figures, the tracking error for joint 1 starts from -0.1 and drops to zero in less than one time unit. Similarly, the tracking error for joint 2 starts from 0.1 and drops to zero. The actual joint angles are exactly equal to the desired joint angles after just one time unit.

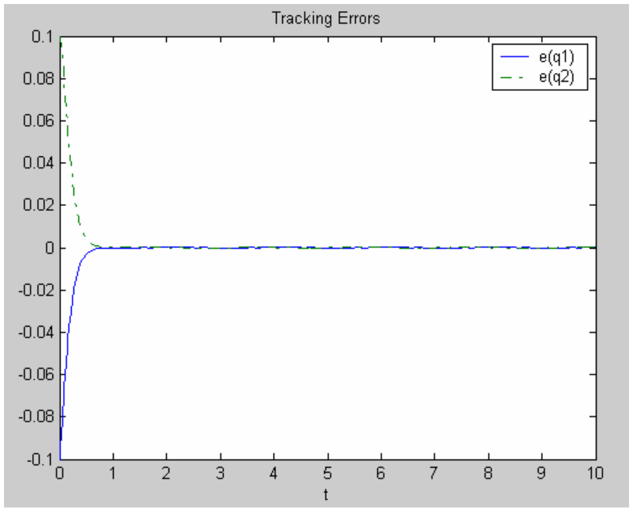


Figure 6.3: Tracking errors for a two-link manipulator using a PD CT controller.

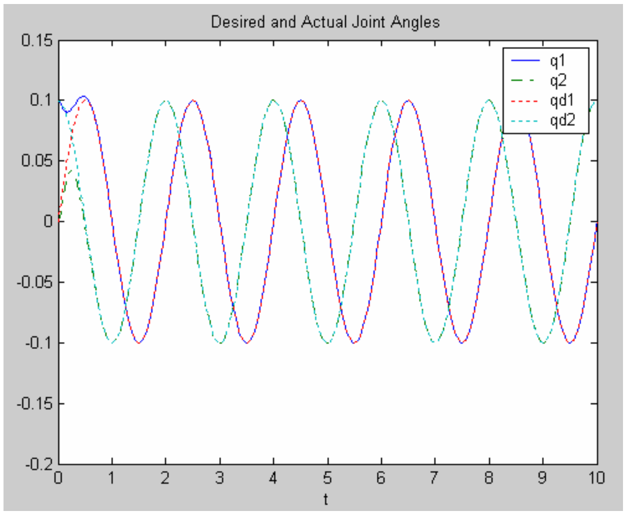


Figure 6.4: Desired versus actual joint angles for a two-link manipulator using a PD CT controller.

In the second experiment, the simulation parameters shown in Table 6.1 are used. However, in this case, a constant torque disturbance τ_d is added to the manipulator dynamics as follows:

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) + \tau_d = \tau \qquad (6.13)$$

where:

$M(q), V(q,\dot{q}), G(q), \tau$: are as given in Eq. (6.9).

The torque disturbance for the first link, τ_{d1} and for the second link, τ_{d2} are set at 10 Nm.

The results of this experiment are shown in Figs. 6.5-6.6, where Fig. 6.5 shows the tracking errors for the joints, while Fig. 6.6 shows the desired versus actual joint angles. High tracking errors are detected, and the tracking errors no longer drop to zero. The tracking error for joint 1 oscillates about -0.1, while the tracking error for joint 2 starts at 0.1 and increases to 0.3. The actual joint angles fail to follow the desired joint angles. This result shows that the CT PD controller is not suitable in the presence of torque disturbances.

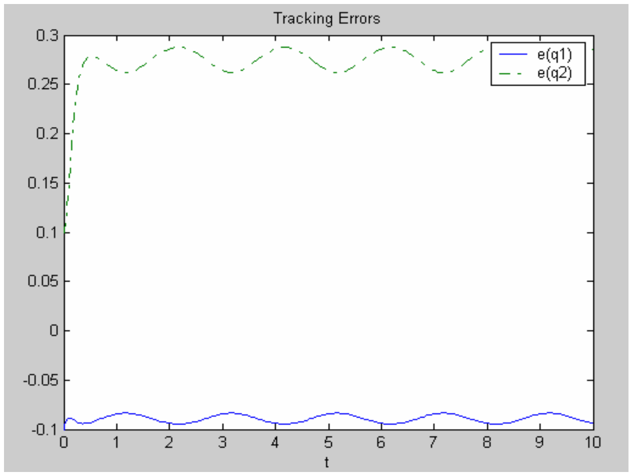


Figure 6.5: Tracking errors of a two-link manipulator using a PD CT controller with a constant torque disturbance.

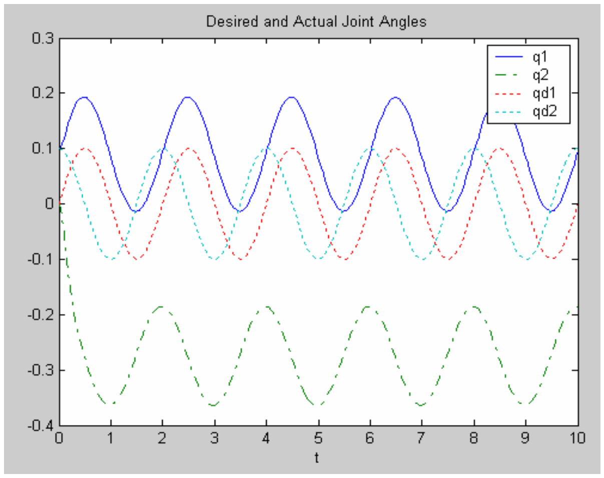


Figure 6.6: Desired versus actual joint angles of a two-link manipulator using a PD CT controller with a constant torque disturbance.

An approximation of the PD CT controllers is the PD-gravity controller, which can be implemented easily by adding a nonlinear gravity term to the torque equation , as follows [80]:

$$\tau = k_v \dot{e} + k_p e + G(q) \tag{6.14}$$

Despite neglecting the nonlinear Coriolis/centripetal terms, the PD-gravity controller performs well in many applications.

For a two-link manipulator, using the simulation parameters in Table 6.1, the results for the PD-gravity controller simulation are shown in Figs. 6.7-6.8:

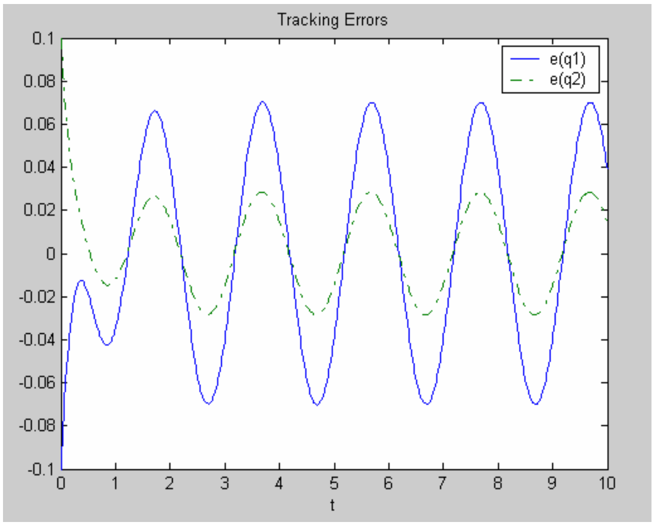


Figure 6.7: Tracking errors of a two-link manipulator using a PD-gravity controller.

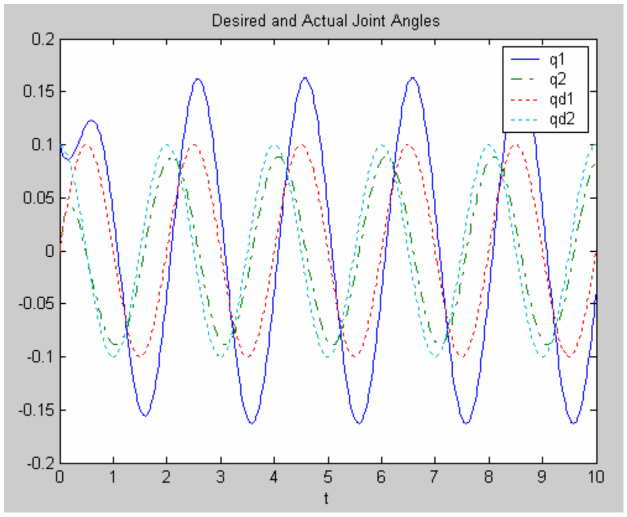


Figure 6.8: Desired versus actual joint angles of a two-link manipulator using a PD-gravity controller.

Figs. 6.7 and 6.8 show that the errors are small and the actual joint angles are close to the desired joint angles. However, the PD CT controller gave better results, which leads to the following conclusion: Whenever there is a good dynamic model, do not use approximations. Controllers using approximation perform well in cases where it is difficult to obtain a good dynamic model.

6.2.4 Simulation of the PD CT controller for the WMR navigation

6.2.4.1 Simulation architecture

The function of this controller is to select the suitable motor torques, so that the WMR will follow the desired motion trajectory, $q_d(t)$. For a two-link manipulator, $q_d(t)$ is already available from the description of the task the manipulator needs to perform. However, for a WMR, $q_d(t)$ is not available from the task description. Instead, the $q_d(t)$ used is the desired path produced from the navigation system described in chapter 5.

The dynamic model described in Eq. (6.1) can be written, as illustrated in Eq. (6.3), in terms of linear and nonlinear components:

$$M(q)\ddot{q} + N(q,\dot{q}) = \tau \tag{6.15}$$

where $N(q,\dot{q})$ represent the nonlinear terms:

$$N(q,\dot{q}) \equiv J(q,\dot{q})\dot{q} + F$$

The WMR simulation program is very similar to the one for the two-link manipulator. It has the following main components:

- The first component computes the desired WMR trajectory from the input from the navigation system. The desired trajectory, $q_d(t)$, is:

$$q_d = \begin{bmatrix} x_{cd} \\ y_{cd} \\ \theta_d \end{bmatrix} \tag{6.16}$$

where:

x_{cd}: is the x-axis component of the desired position of the WMR center of gravity;

y_{cd}: is the y-axis component of the desired position of the WMR center of gravity;

θ_d: is the desired orientation of the WMR.

- The second component calculates the controller input from the tracking error between the desired trajectory , $q_d(t)$, and the actual trajectory , $q(t)$, where $q(t)$ is:

$$q = \begin{bmatrix} x_c \\ y_c \\ \theta \end{bmatrix} \tag{6.17}$$

where:

x_c: is the x-axis component of the actual position of the WMR center of gravity;

y_c: is the y-axis component of the actual position of the WMR center of gravity;

θ: is the actual orientation of the WMR.

Then the inertia term $M(q)$ and the nonlinear term $N(q,\dot{q})$ are computed from the WMR dynamic model, described in Eq. (6.1). Finally, the motor torques are calculated by using Eq. (6.7).

- The third component calculates the new position of the WMR by using the state-space equation, $\dot{x} = f(x,u)$, where the state-space position/velocity form is used [80]:

$$x \equiv \begin{bmatrix} q^T \\ \dot{q}^T \end{bmatrix} \tag{6.18}$$

$$\dot{x} = \begin{bmatrix} \dot{q} \\ -M^{-1}(q)N(q,\dot{q}) \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}(q) \end{bmatrix} \tau$$

This form is used to update the WMR actual position. Eq. (3.62) shows that $M(q)$ is not square. Hence, the Moore-Penrose generalized inverse is used.

The inputs to the PD CT controller simulation program are:

1. The desired motion trajectory, $q_d(t)$;
2. Robot parameters;

3. Controller parameters k_p and k_v.

The outputs of the PD CT controller simulation program are:

1. The motor torques, $\tau(t)$;
2. The actual path, $q(t)$.

6.2.4.2 Simulation results

Several experiments are conducted on the PD CT simulation software. The robot parameters are according to Bearcat III and are shown in Table 6.2:

Bearcat III parameters	
m	306.180 kg
r	0.210 m
J_0	0.274 kgm^2
J_c	42.997 kgm^2
e	0.338 m
d	0.432 m
f_n	629.451 N

Table 6.2: Bearcat III parameters.

Different controller parameters are tested by using a sinusoidal desired motion trajectory:

$$q_d = \begin{bmatrix} x_{cd} \\ y_{cd} \\ \theta_d \end{bmatrix} = \begin{bmatrix} c \cdot \sin t \\ c \cdot \cos t \\ c \cdot \sin t \end{bmatrix} \qquad (6.19)$$

where c is a constant.

Moreover, the results are:

Case I: Equal controller parameters for x, y, and θ

In the first set of experiments, the same k_p and k_v are used for the three components of the motion trajectory, $q(x\ ,\ y$ and $\theta)$, and the results are:

- Starting with $k_p = k_v = 0$, the tracking errors are in the range of 0.00-6.20, as shown in Fig. 6.9. The desired and the actual motion trajectory paths are shown in Fig. 6.10. As shown in the figures, the tracking errors are very high for both x and y, while the tracking error is about 0.10 for θ.

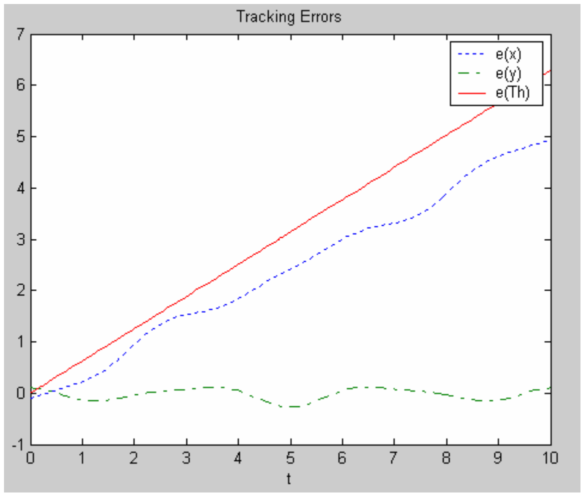


Figure 6.9: Tracking errors for WMR navigation using a PD CT controller.

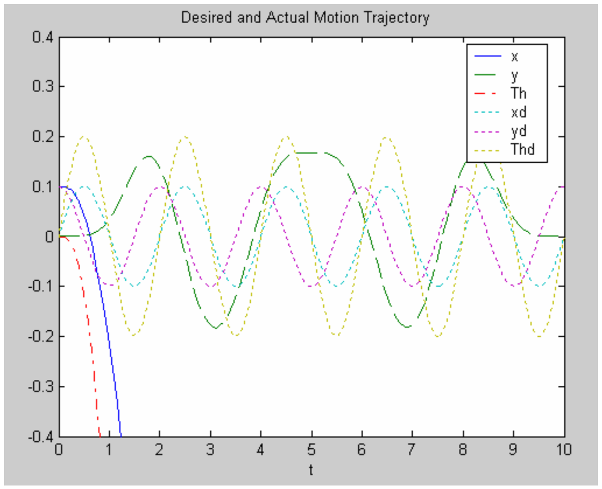


Figure 6.10: Desired versus actual motion trajectories for WMR navigation using a PD CT controller.

- After increasing k_p to 4, and k_v to 10, the results of the tracking errors are shown in Fig. 6.11. While the desired and actual motion trajectories are shown in Fig. 6.12, both figures display smaller tracking errors than in the first case. The tracking errors in the second case are in the range of 0.00-4.25. The tracking error of θ is zero, while the tracking error for x is very small, oscillating around zero. However, y is observed to have a high tracking error.

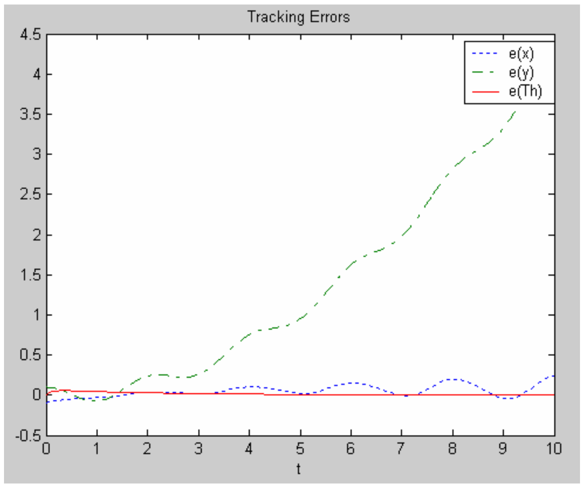


Figure 6.11: Tracking errors for WMR navigation using a PD CT controller.

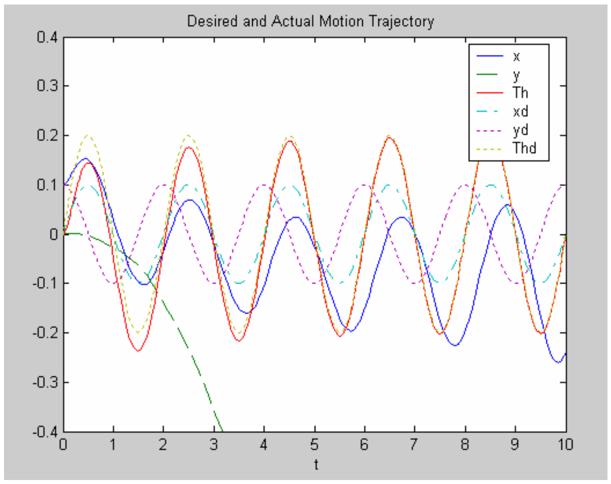


Figure 6.12: Desired and actual motion trajectories for WMR navigation using a PD CT controller.

- After increasing k_p to 10, while reducing k_v to 1, the tracking errors are reduced, as shown in Fig. 6.13. Fig. 6.14 shows the desired and the actual motion trajectories.

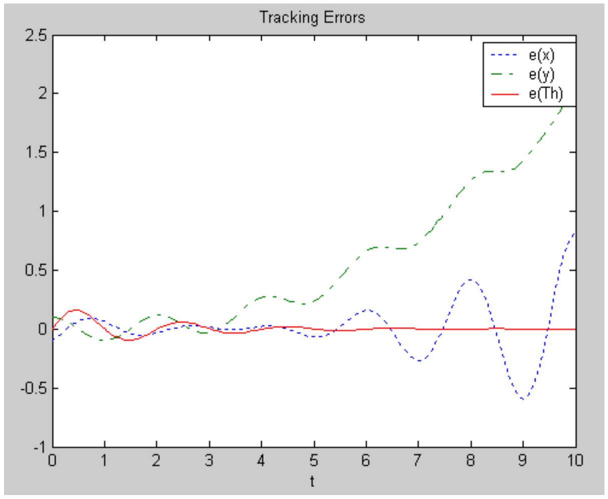


Figure 6.13: Tracking errors for WMR navigation using a PD CT controller.

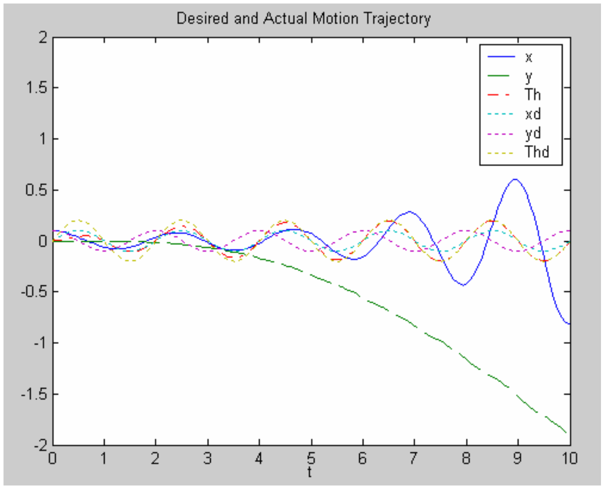


Figure 6.14: Desired and actual motion trajectories for WMR navigation using a PD CT controller.

As shown in these figures, the tracking errors are in the range of 0.00-2.00. The tracking error of θ is still zero. However, x has a greater oscillation about zero. In this case, the tracking error of y is reduced to 2.00, compared to 4.25 in the previous trial.

- Increasing the value of k_p to 100 and the value of k_v to 10, does not make the solution any better, as shown in Figs. 6.15 and 6.16.

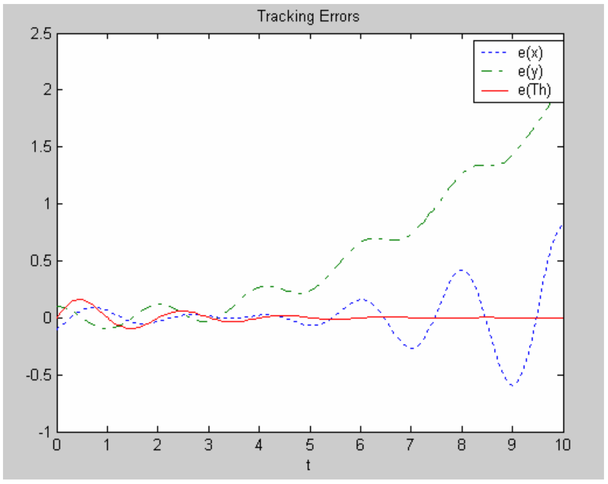


Figure 6.15: Tracking errors for WMR navigation using a PD CT controller.

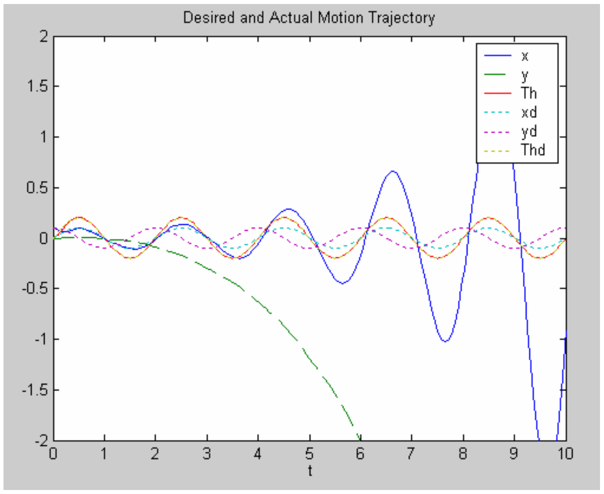


Figure 6.16: Desired and actual motion trajectories for WMR navigation using a PD CT controller.

Case II: Different controller parameters for x, y, and θ

In this set of experiments, different values for k_p and k_v are used for each components of the motion trajectory q. The two parameters k_{p1} and k_{v1} are used to control x. While the two parameters k_{p2} and k_{v2} are used to control y, k_{p3} and k_{v3} are used to control θ. The results are:

- Starting with k_{p1}=2, k_{v1}=1, k_{p2}=0, k_{v2}=10, k_{p3}=2, and k_{v3}=1, the tracking errors are in the range of 0.00-0.04, as shown in Fig. 6.17. The desired versus the actual motion trajectory is shown in Fig. 6.18. These figures display good results, because the tracking errors are zero for both x and θ. Hence, the actual follow the desired motion trajectories for these two components. The tracking error for y oscillates around 0.02.

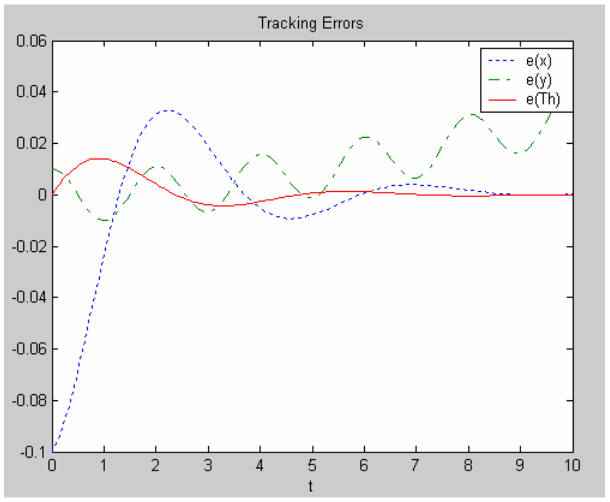


Figure 6.17: Tracking errors for WMR navigation using a PD CT controller.

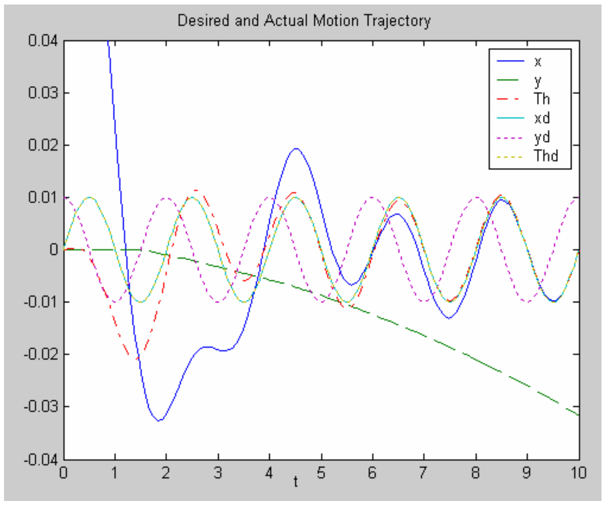


Figure 6.18: Desired and actual motion trajectories for WMR navigation using a PD CT controller.

- The previous experiment shows that the gain values k_{p1}=2, k_{v1}=1, k_{p3}=2, and k_{v3}=1 are good, since they produce zero errors. Therefore, the same values are used in this trial. To observe the effect of increasing the controller gain for y, k_{p2} is set at 20, and k_{v2} is set at 100. The tracking errors are in the range of 0.00-0.04, as shown in Fig. 6.19. The desired versus the actual motion trajectories are shown in Fig. 6.20. Results are very close to the previous ones for y and θ. Therefore, the actual follows the desired orientation, with no improvement in the performance of the actual y trajectory. However, a deterioration in the performance of x is noticed:

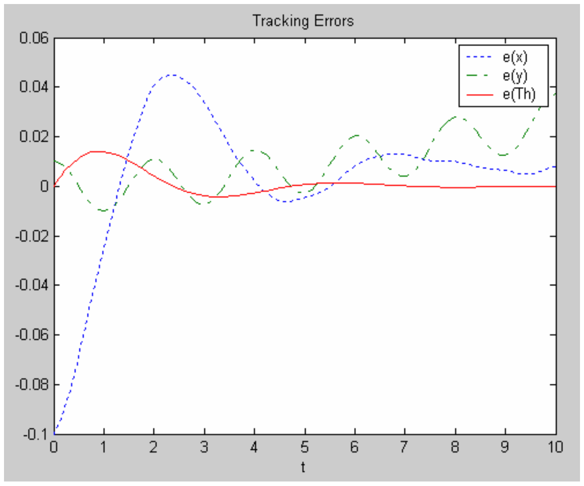


Figure 6.19: Tracking errors for WMR navigation using a PD CT controller.

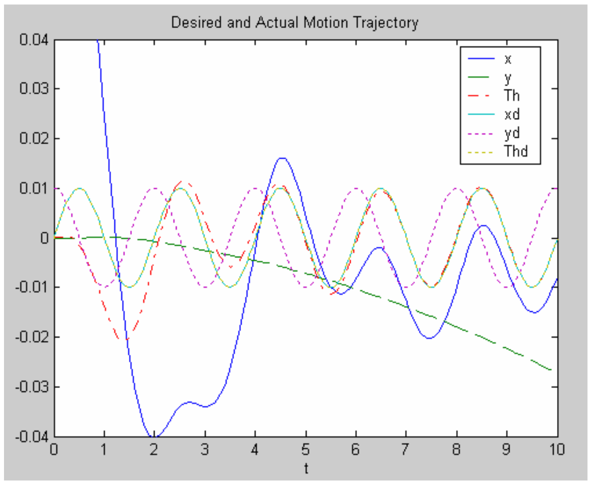


Figure 6.20: Desired and actual motion trajectories for WMR navigation using a PD CT controller.

- Keeping k_{p1}=2, k_{v1}=1, k_{p2}=20, k_{p3}=2, and k_{v3}=1 and further increasing k_{v2} to 1000, the results of the simulation are shown in Figs. 6.21 and 6.22.

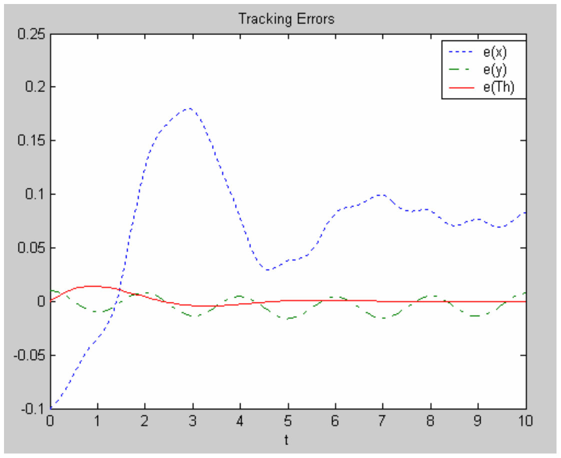


Figure 21.6: Tracking errors for WMR navigation using a PD CT controller.

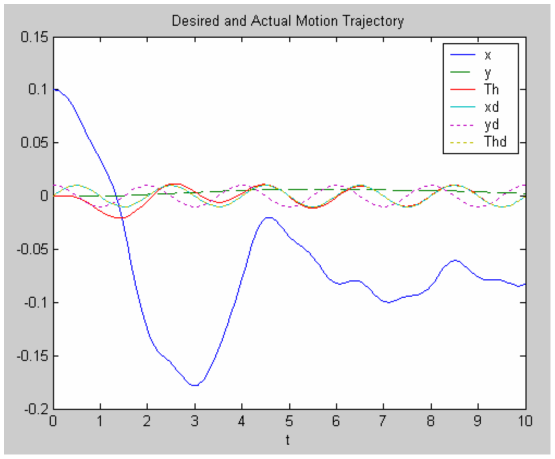


Figure 22.6: Desired and actual motion trajectories for WMR navigation using a PD CT controller.

These results are observed to be worse than those in the previous two experiments, because the obtained tracking errors are in the range of 0.00 to 0.08. Although the actual follows the desired orientation, both x and y actual trajectories are far from the desired trajectories.

6.2.4.3 Conclusion

As shown in the above figures, it is clear that k_p and k_v need to be different for each component of the motion trajectory q. Small positive values are needed in order to get good results. The gain parameters k_{p1}=2, k_{v1}=1, k_{p3}=2, and k_{v3}=1 are observed to give good results. The values of parameters k_{p2} and k_{v2} should not be too large. However, setting the values of k_{p2} at 0 and k_{v2} at 10 produces the best results.

The selection process of proper parameters for the controller is a challenging task, since there are six parameters to be changed at each trial. Although x, y, and θ are independent of each other, they are, nevertheless, used to update the new values of the actual motion trajectories. Hence, they have an impact upon each other. Therefore, keeping the same value of k_p and k_v for a particular motion trajectory component does not imply that the actual motion trajectory will remain the same. Finding a good method for calculating the best values for the controller parameters is recommended.

6.2.5 Simulation for the PID CT controller for the two-link manipulator

6.2.5.1 Simulation architecture

The PID CT controller simulation program for the two-link manipulator is developed in the same way that the PD CT controller simulation program was developed in section 6.2.3. However, Eq. (6.8) is used in the calculation of the motor torques.

The inputs to the PID CT controller simulation program are:

1. The desired motion trajectory, $q_d(t)$;
2. Robot parameters;
3. Controller parameters, k_p, k_v, and k_i.

The outputs of the PID CT controller simulation program are:

1. The motor torques, $\tau(t)$;
2. The actual path, $q(t)$.

6.2.5.2 Simulation results

Two experiments are conducted with the PID CT controller: one without any torque disturbances and the other with constant torque disturbances. The simulation parameters for the two experiments are shown in the following table:

Arm parameters	
m_1	1 kg
m_2	1 kg
a_1	1 m
a_2	1 m
Desired trajectory	
$q_{d1}(t)$	$0.1\sin t$
$q_{d2}(t)$	$0.1\cos t$
Controller parameters	
k_p	100

k_v	20
k_i	50
Note:	m_1, m_2, a_1, and a_2 are according to Fig. 2.6.

Table 6.3: Simulation parameters for the PID CT controller.

The results of this experiment are shown in Figs. 6.23-6.24. Fig. 6.23 shows the tracking errors for the joint angles, while Fig. 6.24 shows the desired versus actual joint angles. As shown in these figures, the tracking error for joint 1 starts at -0.1 and drops to zero in five time units. Similarly, the tracking error for joint 2 starts at 0.1 and drops to zero. The actual joint angles are equal to the desired joint angles.

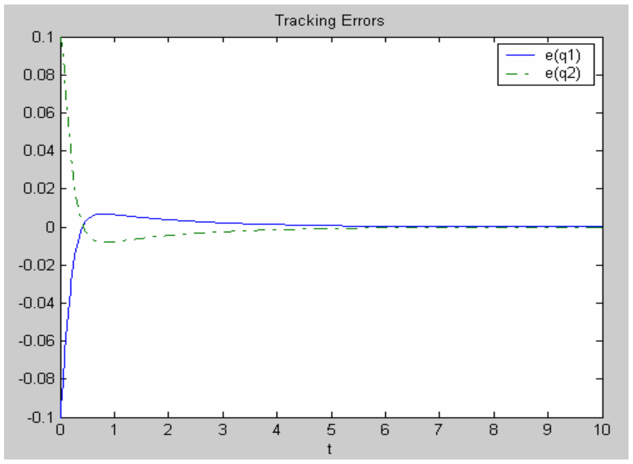


Figure 6.23: Tracking errors for a two-link manipulator using a PID CT controller.

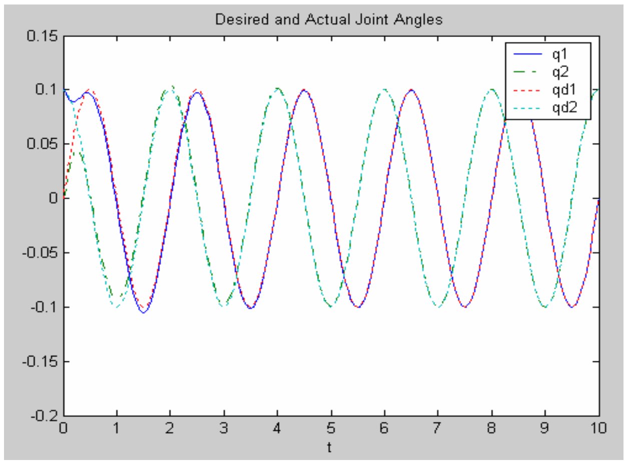


Figure 6.24: Desired versus actual joint angles for a two-link manipulator.

In the second experiment a constant torque disturbance, τ_d, is added to the manipulator dynamics. The torque disturbance for the first link, τ_{d1}, and for the second link, τ_{d2}, are set at 10 Nm. The remaining simulation parameters are shown in Table 6.3.

The results of this experiment are shown in Figs. 6.25-6.26, where Fig. 6.25 shows the tracking errors for the joint angles, while Fig. 6.26 shows the desired versus actual joint angles.

As shown in these figures, the controller performance is very acceptable, and it is not affected by the torque disturbance added. Hence, the PID CT controller can be used, even with existing constant torque disturbances.

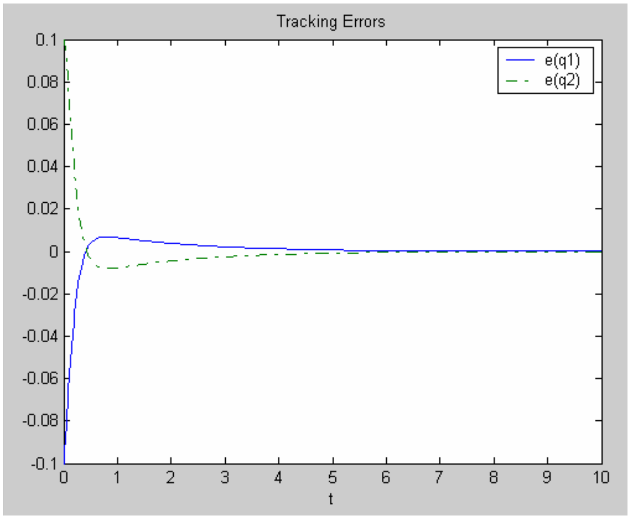


Figure 6.25: Tracking errors for a two-link manipulator using a PID CT controller with a constant torque disturbance.

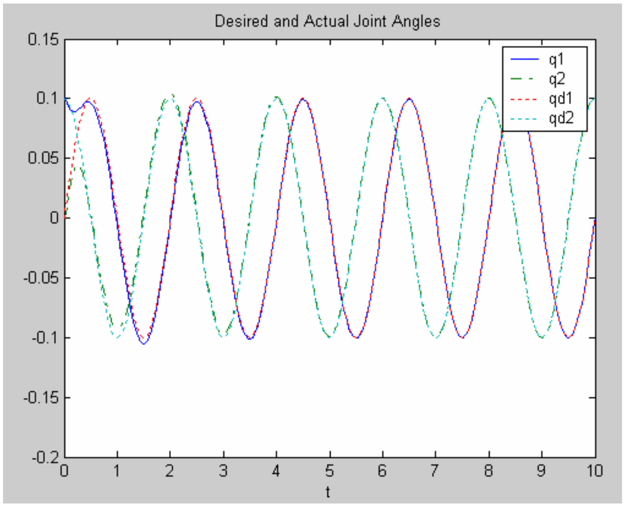


Figure 6.26: Desired versus actual joint angles for a two-link manipulator using a PID CT controller with a constant torque disturbance.

6.2.6 Simulation for the PID CT controller for the WMR navigation

6.2.6.1 Simulation architecture

The PID CT controller simulation program for the WMR motion is developed in the same way that the PD CT controller simulation program was developed in section 6.2.4, except that Eq. (6.8) is used in calculating the motor torques.

The inputs to the PID CT controller simulation program are:

1. The desired motion trajectory, $q_d(t)$;
2. Robot parameters;
3. Controller parameters, k_p, k_v, and k_i.

The outputs of the PID CT controller simulation program are:

1. The motor torques, $\tau(t)$;
2. The actual path, $q(t)$.

6.2.6.2 Simulation results

Several experiments are conducted on the PID CT simulation software. The robot parameters are according to Bearcat III, shown in Table 6.2. Different trajectories and controller parameters are tried, and the results are:

Case I: Using a sinusoidal desired motion trajectories:

$$q_d = \begin{bmatrix} x_{cd} \\ y_{cd} \\ \theta_d \end{bmatrix} = \begin{bmatrix} c \cdot \sin t \\ c \cdot \cos t \\ c \cdot \sin t \end{bmatrix} \tag{6.20}$$

where c is a constant.

- The first experiment in this set starts with small positive values for k_p, k_v, and k_i, where k_p=2, k_v=1, and k_i=1. The tracking errors are in the range of 0.00-0.04, as shown in Fig. 6.27. The desired versus the actual motion trajectories are shown in Fig. 6.28. As shown in these figures, the tracking errors oscillate around zero for both x and θ, while the error oscillates around .04 for y. Although this is an acceptable margin of error, more experiments will be conducted to reduce the error in the y trajectory.

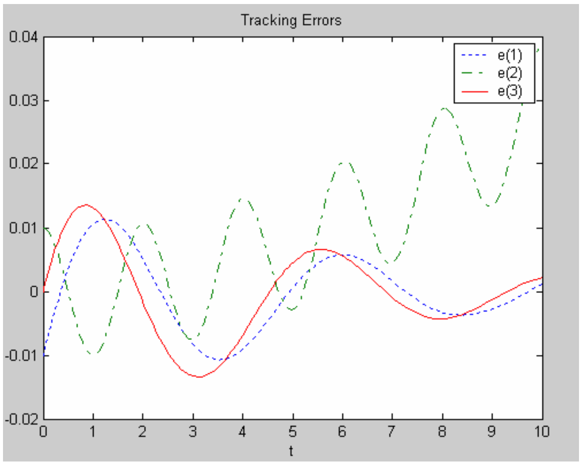


Figure 6.27: Tracking errors for WMR navigation using a PID CT controller.

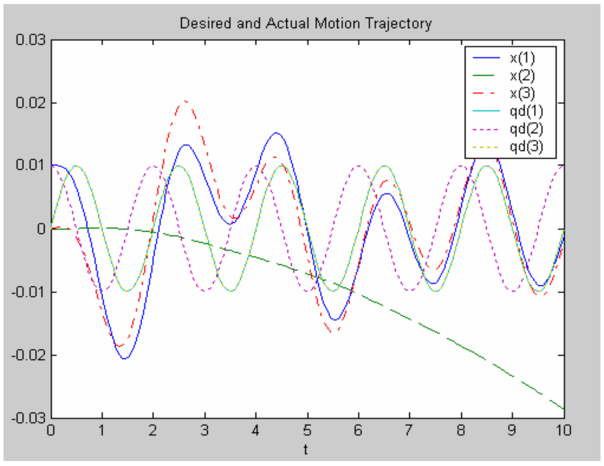


Figure 6.28: Desired and actual motion trajectories for WMR navigation.

- The second experiment is conducted by increasing k_v to 5, while keeping the same values for k_p and k_i (k_p=2 and k_i=1). The tracking errors are shown in Fig. 6.29, and the desired versus the actual motion trajectories are shown in Fig. 6.30. As shown in these figures, the tracking errors now are in the range of 0.00-3.75. There is less tracking error of θ, and a greater tracking error of y. However, the tracking error of y is less oscillatory than in the first case.

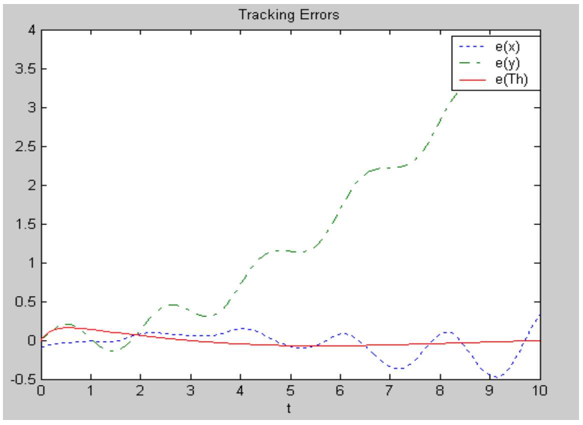


Figure 6.29: Tracking errors for WMR navigation using a PID CT controller.

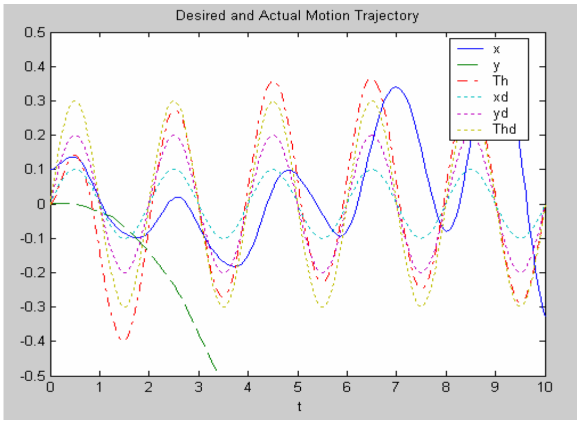


Figure 6.30: Desired versus actual motion trajectories for WMR navigation.

- The third experiment is conducted by increasing k_v to 10, while keeping the same values for k_p and k_i (k_p=2 and k_i=1). The tracking errors are shown in Fig. 6.31, and the desired versus the actual motion trajectories are shown in Fig. 6.32. As shown in these figures, the tracking errors now are in the range of 0.00-4.20. There is greater improvement in the tracking error of θ and x, and a greater tracking error of y. Again, the tracking error of y has less oscillation than in both the first and the second case.

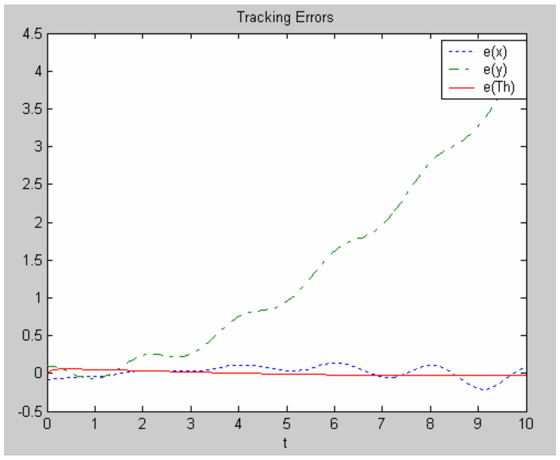


Figure 6.31: Tracking errors for WMR navigation using a PID CT controller.

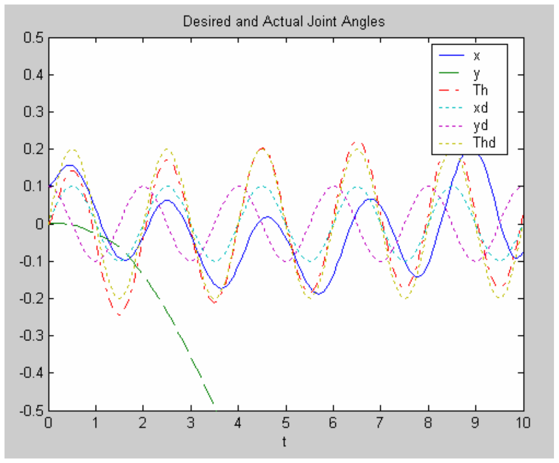


Figure 6.32: Desired versus actual motion trajectories for WMR navigation using a PID CT controller.

- The fourth experiment is conducted by setting the value of k_p at 0 and both k_v and k_i at 10. The tracking errors are in the range of 0.00-2.75 as shown in Fig. 6.33. The desired versus the actual motion trajectories are shown in Fig. 6.34. As shown in these figures, the tracking error oscillates around zero for θ. The tracking error oscillates between 0.00 and -0.90 for x, and between 0.00 and 2.75 for y.

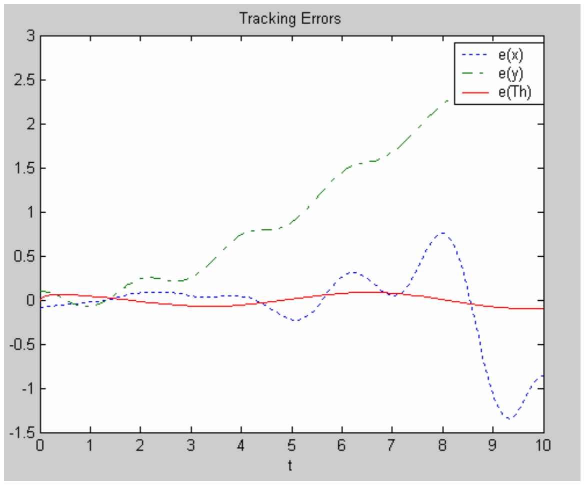


Figure 6.33: Tracking errors for WMR navigation using a PID CT controller.

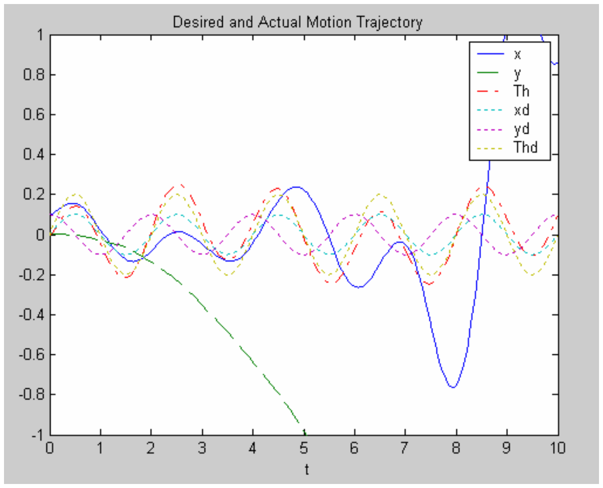
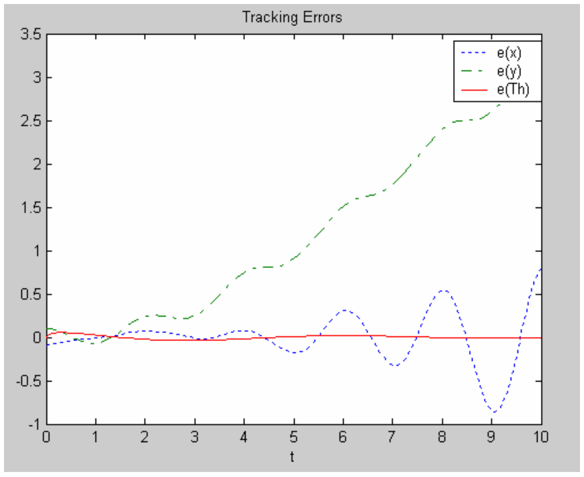


Figure 6.34: Desired versus actual motion trajectories for WMR navigation using a PID CT controller.

- The fifth experiment in this set is conducted by increasing the value of k_p to 10, while keeping the same values for k_v and k_i (k_v=k_i=10). The tracking errors are shown in Fig. 6.35, while the desired versus the actual motion trajectories are shown in Fig. 6.36. The tracking errors now are in the range of 0.00-3.15, compared to 0.00-2.75 in the fourth experiment. Noticeable is an improvement in the tracking error of θ and x, and a small increase in the tracking error of y. However, the tracking error of y oscillates less than in the first case.



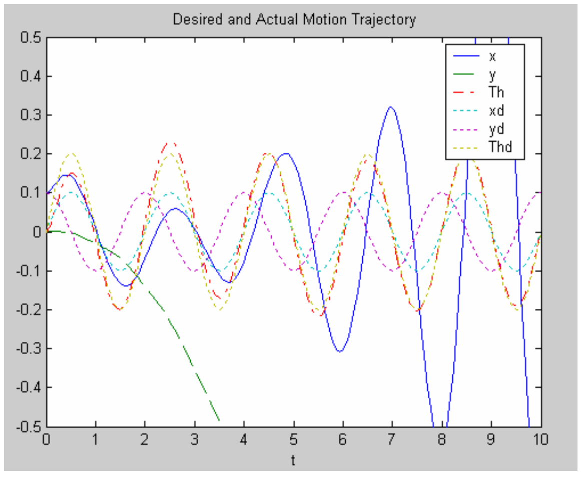
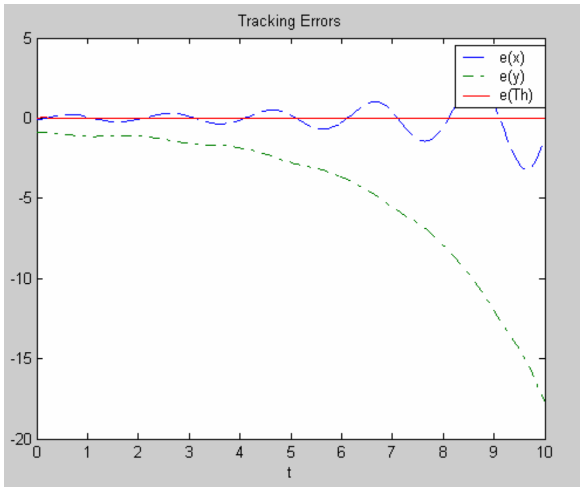


Figure 6.35: Tracking errors for WMR navigation using a PID CT controller.

Figure 6.36: Desired versus actual motion trajectories for WMR navigation.

- The sixth experiment is conducted by increasing k_p to 100, while keeping k_v at 10 and reducing k_i to 0. The tracking errors are shown in Fig. 6.37, and the desired versus the actual motion trajectories are shown in Fig. 6.38. As shown in these figures, the tracking errors now are extremely high (in the range of -17.00-2.50).



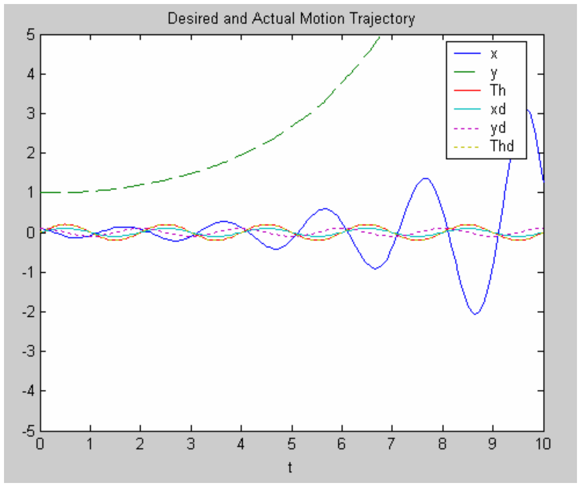


Figure 6.37: Tracking errors for WMR navigation using a PID CT controller.

Figure 6.38: Desired versus actual motion trajectories for WMR navigation using a PID CT controller.

Case II: Using the following desired motion trajectories:

$$q_d = \begin{bmatrix} x_{cd} \\ y_{cd} \\ \theta_d \end{bmatrix} = \begin{bmatrix} c_1 \cdot t^2 \\ c_1 \cdot t^2 + c_2 \cdot t \\ c_3 \cdot \sin t \end{bmatrix} \tag{6.21}$$

where c_1, c_2, and c_3 are constants.

- An experiment is conducted by using k_p=2, k_v=1, and k_i=1 for this desired motion trajectory. The tracking errors are in the range of -0.25-0.00, as shown in Fig. 6.39. The desired versus the actual motion trajectories are shown in Fig. 6.40. As shown in these figures, the tracking errors for x and θ are very small, oscillating around zero. For y, the tracking error starts at zero and increases to around 0.25. However, it is still small.

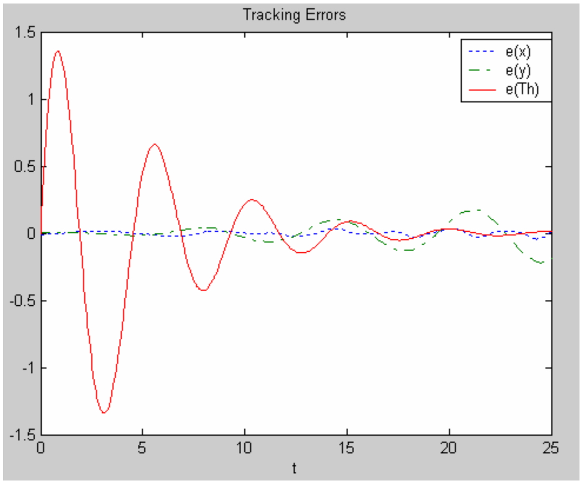


Figure 6.39: Tracking errors for WMR navigation using a PID CT controller.

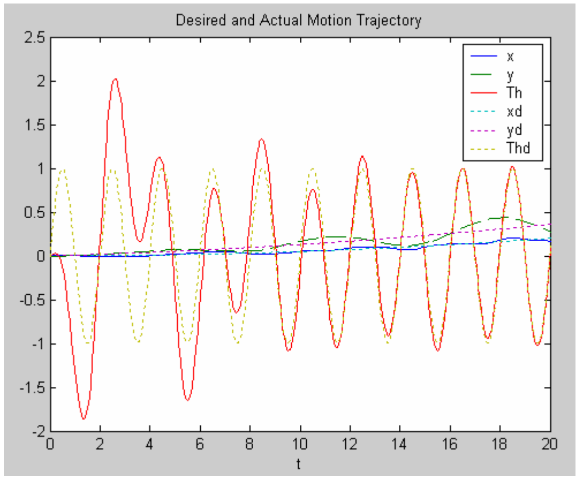


Figure 6.40: Desired versus actual motion trajectories for WMR navigation using a PID CT controller.

Case III: Using the following desired motion trajectory:

$$q_d = \begin{bmatrix} x_{cd} \\ y_{cd} \\ \theta_d \end{bmatrix} = \begin{bmatrix} c_1 \cdot t \\ c_2 \cdot t \\ c_3 \cdot \sin t \end{bmatrix} \tag{6.22}$$

where c_1, c_2, and c_3 are constants.

- An experiment is conducted by using the same controller parameters as those in case II (k_p=2, k_v=1, and k_i=1). The tracking errors are in the range of -0.01-0.35, as shown in Fig. 6.41. The desired versus the actual motion trajectories are shown in Fig. 6.42. As shown in these figures, the tracking error for θ is zero. For x it oscillates around zero, and for y it starts at zero and increases to 0.35.

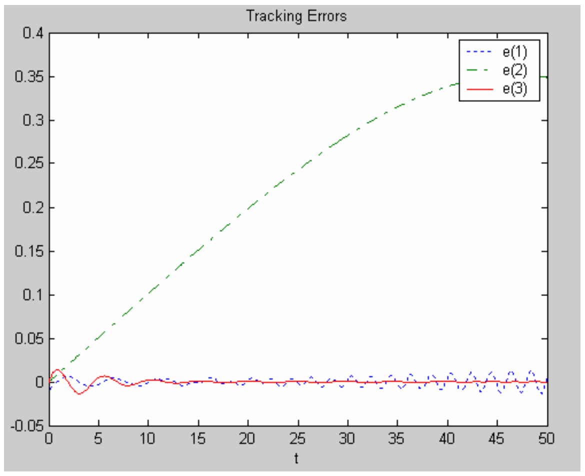


Figure 6.41: Tracking errors for WMR navigation using a PID CT controller.

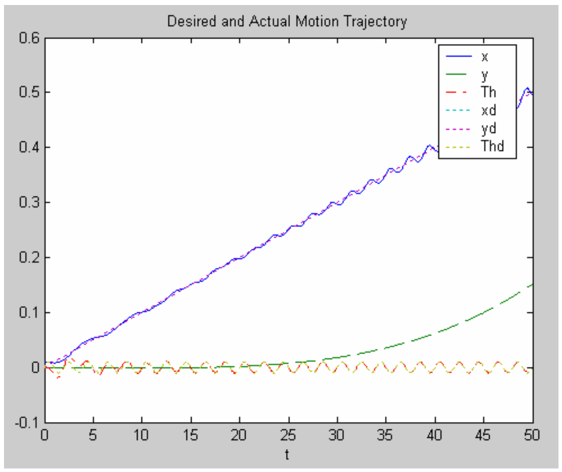


Figure 6.42: Desired versus actual motion trajectories for WMR navigation using a PID CT controller.

6.2.6.3 Conclusion

As shown in the figures, it is clear that the values of k_p, k_v, and k_i need to be small positive numbers to obtain good results. It is also noticeable that increasing k_p and fixing k_v

and k_i, or increasing k_v and fixing k_p and k_i, reduces the tracking errors for θ and x, while it increases the tracking error of y. However, there is a limit to this increase, which is about 10.

Using very large, or zero, values for k_p, k_v, or k_i is not recommended. Remember that the value of k_i must not be too large, as a condition for having a stable tracking error.

k_p=2, k_v=1, and k_i=1 seem to give very reasonable results. Hence, using these values is recommended. However, this depends on the type of application. It is up to the simulation users' discretion to choose the values that best fit their interests.

6.3 Digital controller

Significant progress has been made in the analysis and design of discrete-data and digital control systems in recent years. Due to the rapid advances in digital control theory, the availability of high performance low-cost microprocessors, and digital signal processors, digital control systems gained popularity and importance in industry. However, stability and sampling period selection are important factors to consider [128].

Three design approaches for digital control systems exist: the direct digital design, the digital redesign, and the direct sampled-data. The direct digital design approach discretizes the analog plant and then determines a digital controller. The digital redesign approach pre-designs an analog controller for the analog plant and then carries out the digital redesign for the analog controller. The direct sampled-data approach directly designs a digital controller for the analog plant [129].

The PD digital controller output can be calculated by using the following equation [80]:

$$\tau_\varepsilon = M(q_\varepsilon)(\ddot{q}_{d\varepsilon} + K_v\dot{e}_\varepsilon + K_p e_\varepsilon) + N(q_\varepsilon, \dot{q}_\varepsilon) \tag{6.23}$$

The control input can be calculated only at certain sample times, $t_\varepsilon = \varepsilon T$, where:

T : is the sample period;

ε : is the integer value.

Care should be taken with some of the problems inherent to digital controllers, such as stability, actuator saturation, and antiwindup [80].

6.3.1 Simulation of the digital controller for the two-link manipulator

Simulation software for the digital controller of a two-link manipulator is developed, as follows, using Eq. 23.6 and the two-link manipulator dynamics:

- The controller input is only updated at times εT .
- The controller computes the first control output by using the initial robot dynamics state values. The integrator then holds this value for its calculations over one sample time.
- For the next sample period, the final robot dynamics state values are assigned as the new initial robot dynamics state values. The integrator holds this value for its calculations over this sample time period, etc.

The simulation software for the digital controller is tested for different sample periods. The first experiment is conducted by using the simulation parameters in Table 6.1 and a sample period equal to 20 msec. The results of the digital controller are shown in Fig. 6.43, Fig. 6.44, and Fig. 6.45.

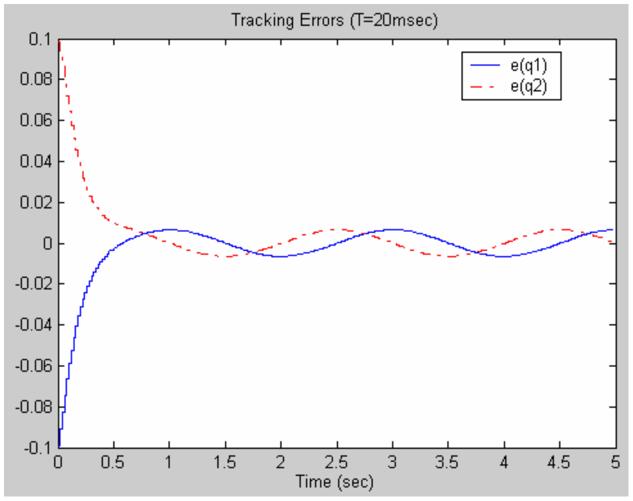


Figure 6.43: Tracking errors for a two-link manipulator, using a digital controller.

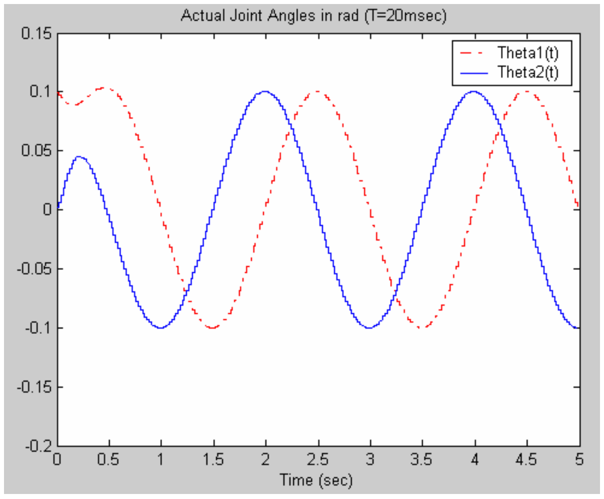


Figure 6.44: Actual joint angles for a two-link manipulator, using a digital controller.

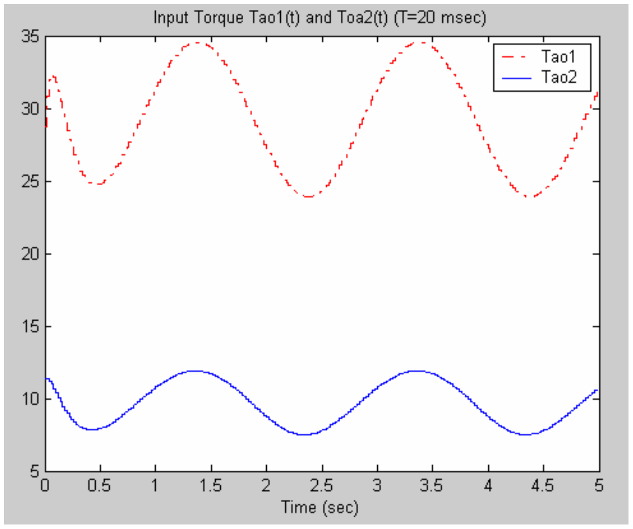


Figure 6.45: Control torques for link 1 and link 2, using a digital controller.

As shown in Fig. 6.43, the tracking errors are very small and close to zero. Fig. 6.44 shows that the actual joint angles are very close to the desired sinusoidal joint angles. Fig. 6.45 shows that the control torques are smooth.

Increasing the sample period above 20 msec impairs the performance of the digital controller, due to windup problems or actuator saturation. For example, for the sample period equal to 100 msec, the results of the digital controller simulation are shown below in Figs. 6.46, 6.47, and 6.48.

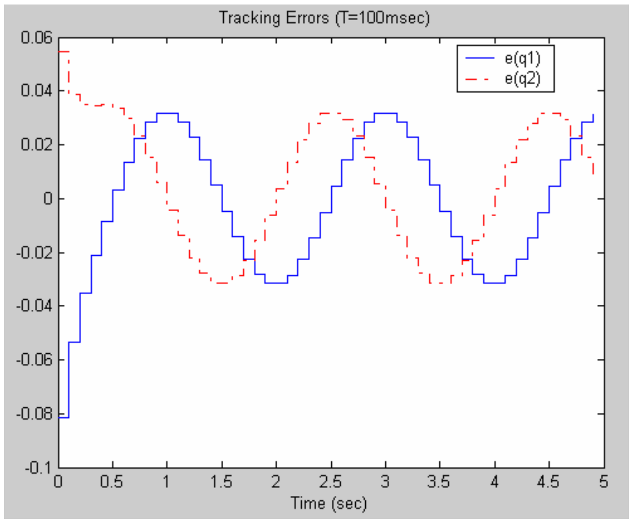


Figure 6.46: Tracking errors for a two-link manipulator, using a digital controller.

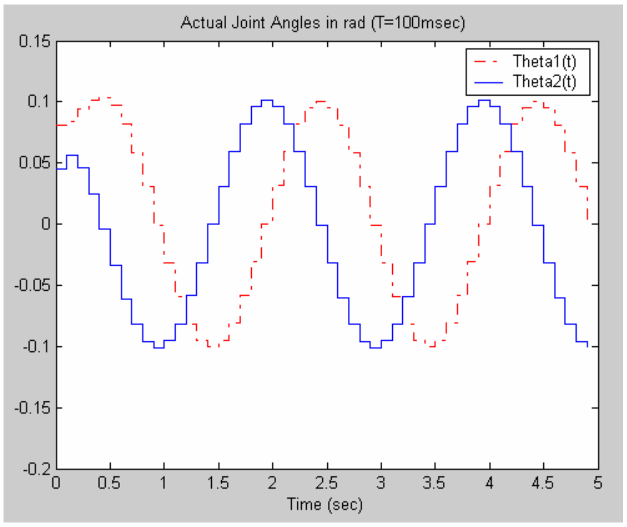


Figure 6.47: Actual joint angles for a two-link manipulator, using a digital controller.

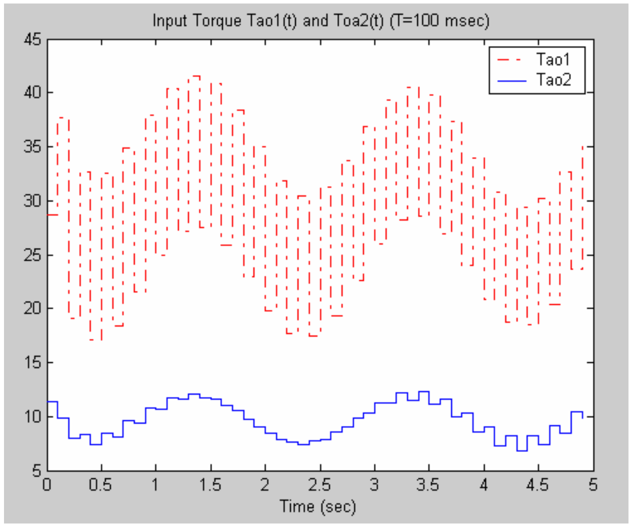


Figure 6.48: Input torques for link 1 and link 2, using a digital controller.

Although the tracking errors are still small, and the actual joint angles are close to the desired joint angles. Fig. 6.48 shows that the input torque for link 1 behavior has entered a limit cycle, which is not acceptable. It is also worth mentioning that very small sampling periods are not recommended either.

6.3.2 Simulation of the digital controller for the WMR motion

Simulation software is developed for the digital controller of a WMR by using Eq. (6.23) and the WMR dynamics. The WMR parameters are according to Bearcat III, which was shown in table 6.2. Different controller parameters and different trajectories are tried and the results are:

Case I: In the first experiment, the following simulation parameters are used:

Desired trajectory	
$q_{d1}(t)$	$0.1\sin t$
$q_{d2}(t)$	$0.1\cos t$
$q_{d3}(t)$	$0.1\sin t$
Controller parameters	
k_p	2
k_v	0
sample period	20 msec

Table 6.4: Simulation parameter for case I.

The results are shown in Figs. 6.49-6.50. As shown in Fig. 6.49, the tracking errors are in the range of -0.8-0.2. The actual motion trajectories are smooth, but far from the desired motion trajectories.

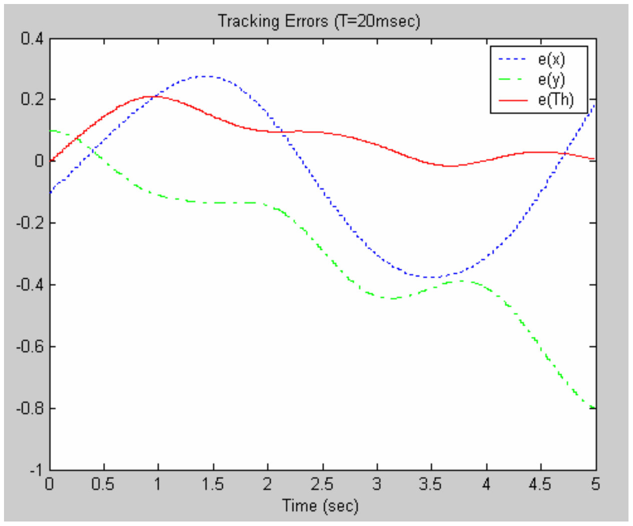


Figure 6.49: Tracking errors for WMR navigation, using a digital controller.

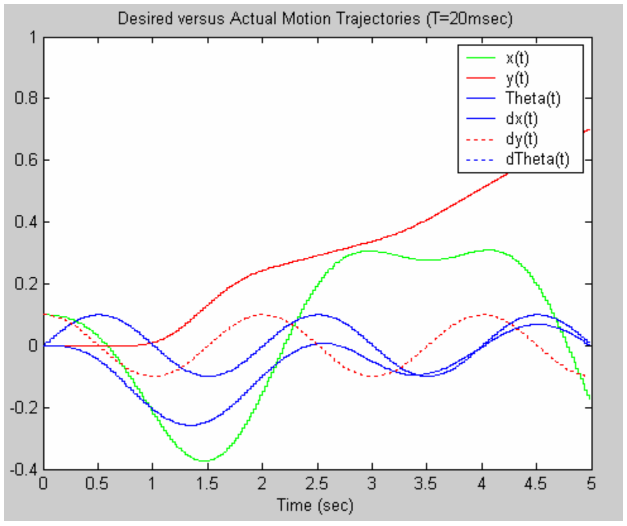


Figure 6.50: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

Case II: To study the effect of increasing the derivative gain, another experiment is conducted, with the following simulation parameters:

Desired trajectory	
$q_{d1}(t)$	$0.01\sin t$
$q_{d2}(t)$	$0.01\cos t$
$q_{d3}(t)$	$0.01\sin t$
Controller parameters	
k_p	2
k_v	1
sample period	20 msec

Table 6.5: Simulation parameters for case II.

The results are shown in Figs. 6.51-6.52. As shown in Fig. 6.51, the tracking errors are very small in this experiment. The tracking errors for x and θ are almost zero. For y, the tracking error oscillates between 0.000 and -0.025. The actual motion trajectories are smooth and match the desired motion trajectories for x and θ. For y, the actual motion trajectory is also smooth, but higher than the desired motion trajectory.

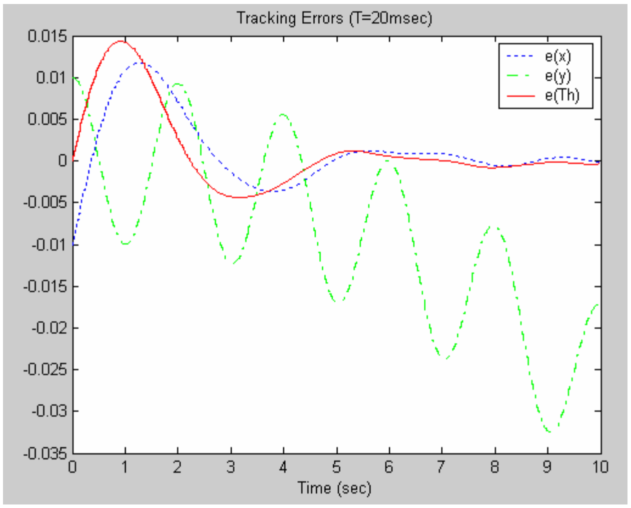


Figure 6.51: Tracking errors for WMR navigation, using a digital controller.

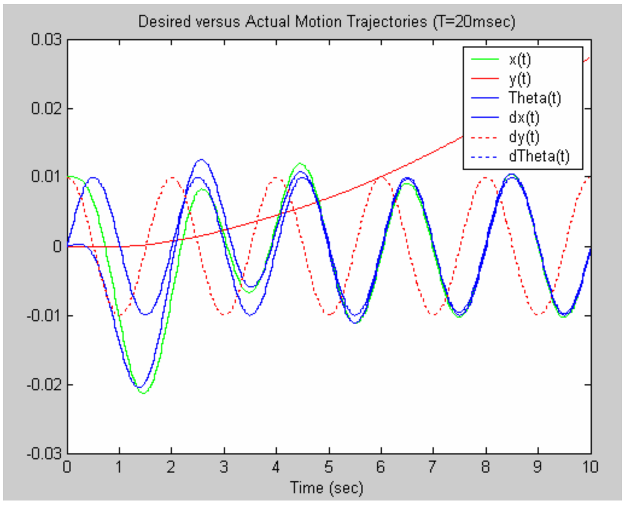


Figure 6.52: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

Further increasing the derivative gain, k_v, from 1 to 100 does not improve the results, as shown in the following figures. The remaining simulation parameters are shown in Table 6.5.

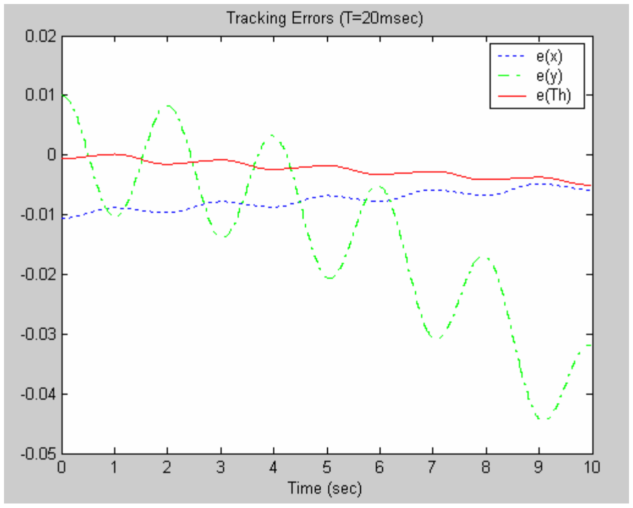


Figure 6.53: Tracking errors for WMR navigation, using a digital controller.

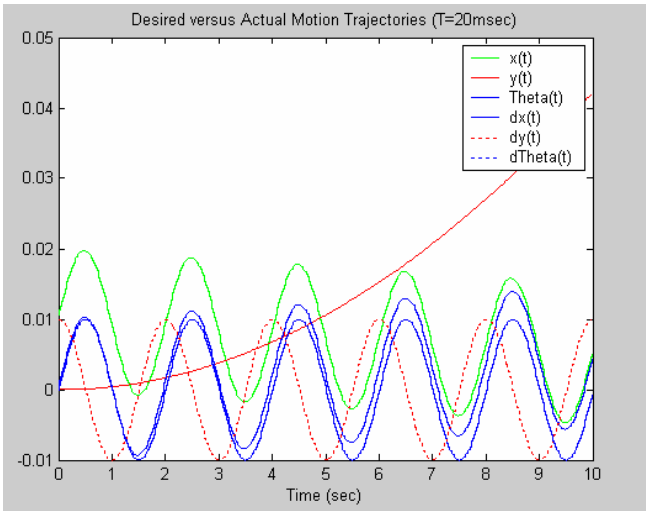


Figure 6.54: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

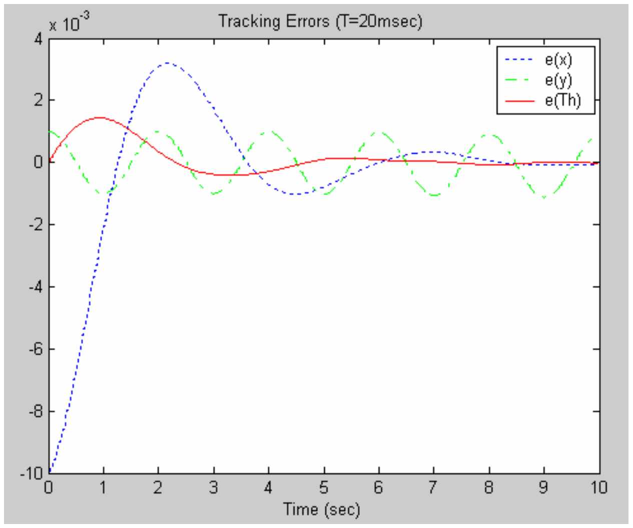
Although these results are still acceptable, the ones for k_v=1 are much better.

Case III: To study the effect of the different amplitudes for the desired motion trajectories, another experiment is conducted with the following simulation parameters:

Desired trajectory	
$q_{d1}(t)$	$0.001\sin t$
$q_{d2}(t)$	$0.001\cos t$
$q_{d3}(t)$	$0.001\sin t$
Controller parameters	
k_p	2
k_v	1
sample period	20 msec

Table 6.6: Simulation parameters for case III.

The results are shown in Figs. 6.55-6.56. As shown in Fig. 6.55, the tracking errors are very small in this experiment. The tracking errors for x and θ are almost zero, while the tracking error for y oscillates around zero. The actual motion trajectories are smooth and match the desired motion trajectories for x and θ. For y, the actual motion trajectory is equal to zero. Comparing these results with the results of case II reveals that both results are the same for x and θ. For y, less oscillation is observed, and the center of the oscillation does not shift. This result is actually interesting, since it shows that even the amplitudes of the desired motion trajectory have an effect on the tracking errors.



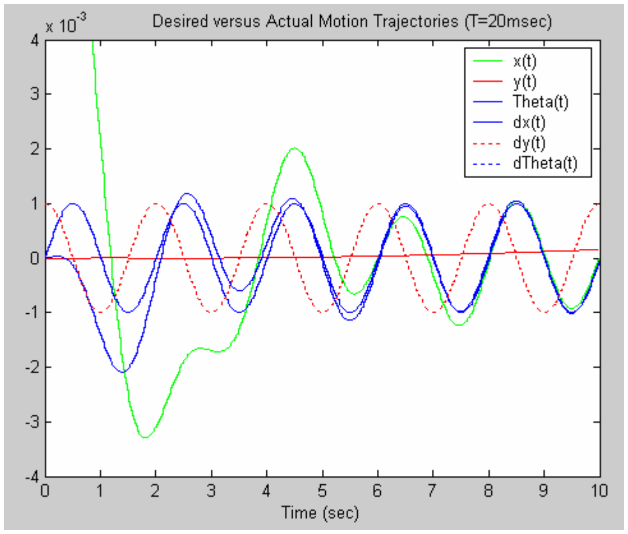


Figure 6.55: Tracking errors for WMR navigation, using a digital controller.

Figure 6.56: Desired versus actual motion trajectories for WMR navigation.

Case IV: To study the effect of different desired motion trajectories, another experiment is conducted, with the following simulation parameters:

Desired trajectory	
$q_{d1}(t)$	$0.0005t^2$
$q_{d2}(t)$	$0.0005t^2 + .008t$

$q_{d3}(t)$	$0.001\sin t$
Controller parameters	
k_p	2
k_v	1
sample period	20 msec

Table 6.7: Simulation parameters for case IV.

The results are shown in Figs. 6.57-6.58. As shown in Fig. 6.57, the tracking errors are very small in this experiment. The tracking errors for x and θ are zero, while the tracking error for y increases from 0.00 to 0.13. The actual motion trajectories are smooth and match the desired motion trajectories for x and θ. For y, the actual motion trajectory equals zero. Comparing these results with the results of case II shows that the desired motion trajectory has an effect on the performance of the tracking errors.

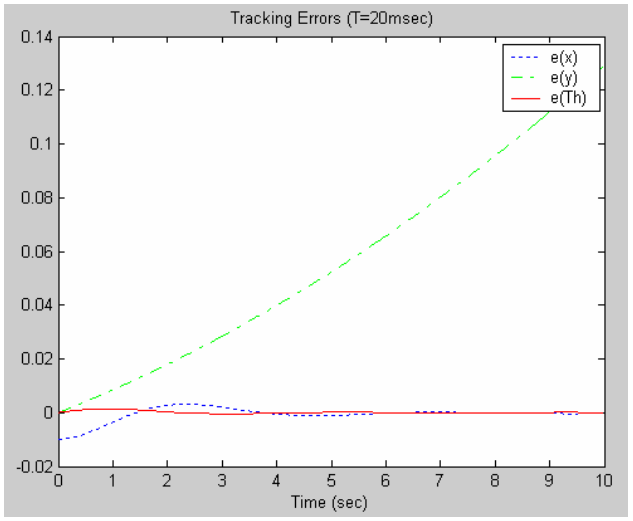


Figure 6.57: Tracking errors for WMR navigation, using a digital controller.

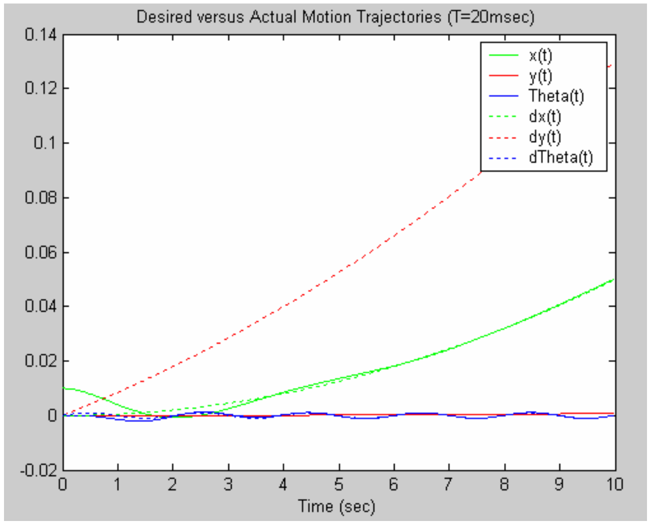


Figure 6.58: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

Using a different amplitude for θ improves the tracking error of y, as shown in the following figures for $q_{d3}(t) = 0.01 \sin t$, while the remaining simulation parameters are according to Table 6.7. As shown in Fig. 6.59, the tracking errors are almost zero for x and θ. For y, the tracking error increases from 0 to 0.045.

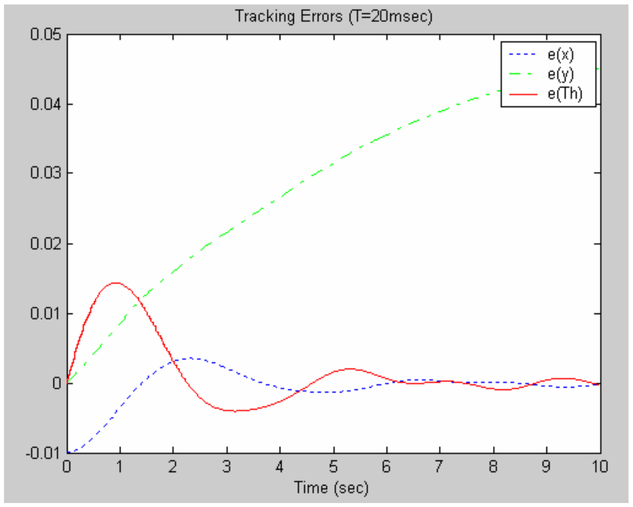


Figure 6.59: Tracking errors for WMR navigation, using a digital controller.

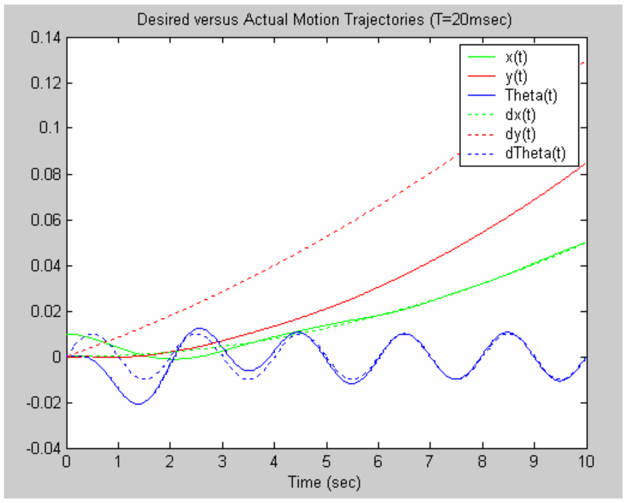


Figure 6.60: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

Case V: To study the effect of increasing the proportional gain, another experiment is conducted, with the following simulation parameters:

Desired trajectory	
$q_{d1}(t)$	$0.0005t^2$
$q_{d2}(t)$	$0.0005t^2 + .008t$
$q_{d3}(t)$	$0.01 \sin t$
Controller parameters	
k_p	50

k_v	1
sample period	20 msec

Table 6.8: Simulation parameters for case V.

The results are shown in Figs. 6.61-6.62. As shown in Fig. 6.61, the tracking errors are small, but oscillating, compared to the previous case. The tracking errors for x and θ oscillate around zero, while the tracking error for y increases from 0 to 0.027. The actual motion trajectories are smooth and match the desired motion trajectory for θ. The actual motion trajectory oscillates around the desired motion trajectory for x. For y, the actual motion trajectory is close to the desired motion trajectory. Comparing these results with the results of case IV shows that increasing the proportional gain reduces the tracking error for y and increases the tracking errors for x and θ.

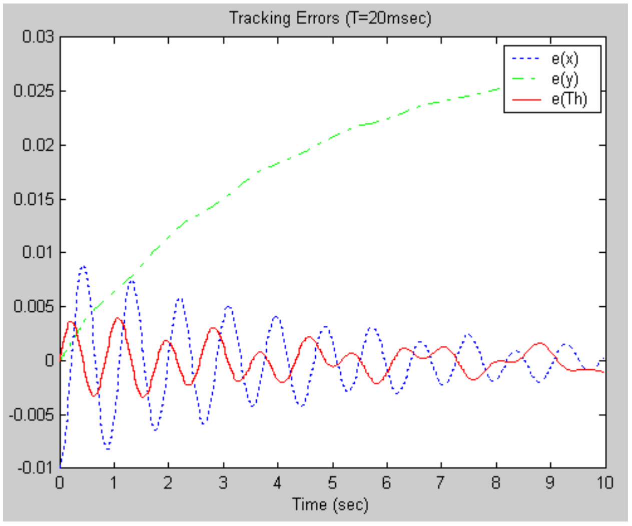


Figure 6.61: Tracking errors for WMR navigation, using a digital controller.

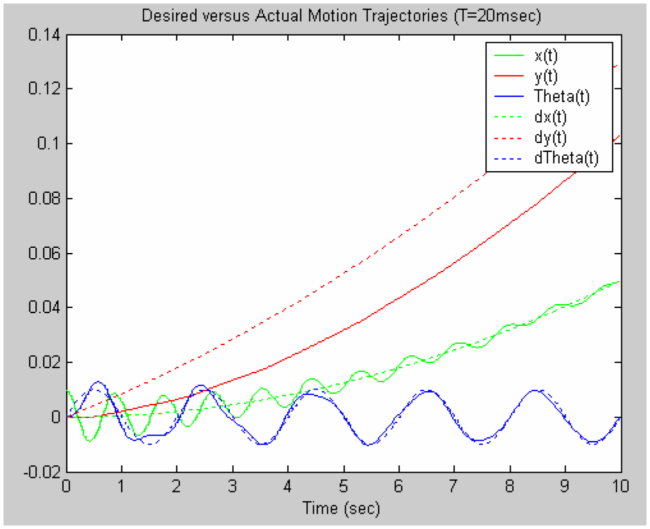


Figure 6.62: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

Further increasing the proportional gain to 100 makes the system unstable, as shown in Figs. 6.63-6.64. Fig. 6.63 shows that the tracking errors are very high, and the actual motion trajectories are very far from those desired.

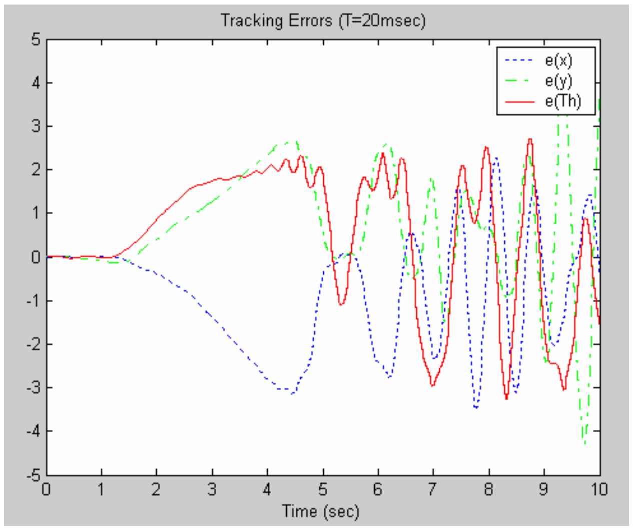


Figure 6.63: Tracking errors for WMR navigation, using a digital controller.

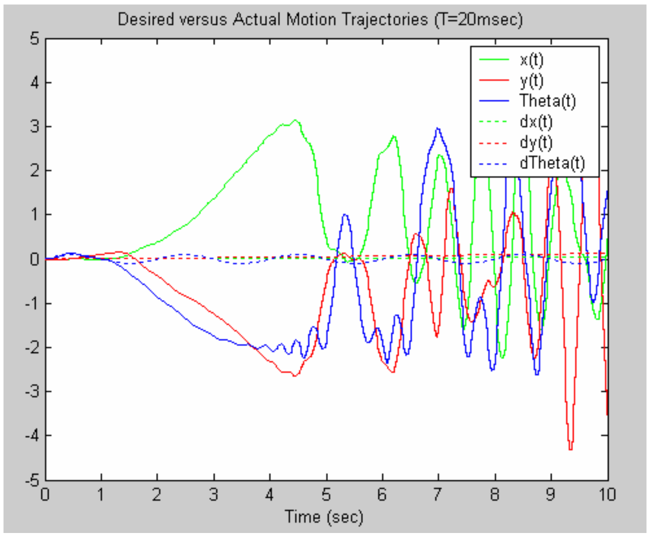


Figure 6.64: Desired versus actual motion trajectories for WMR navigation, using a digital controller.

6.3.2.1 Conclusion

As shown in the above figures, it is clear that k_p=2 and k_v=1 are the best controller parameters. Increasing the value of k_v does not improve results. However, increasing the value of k_p does, provided its value does not exceed 50.

Comparing the performance of the three controllers discussed so far, namely, PD CT, PID CT, and the digital controller, it can be shown that the digital controller provides the best results. Hence, its use is recommended for this application.

6.4 Approximation-based adaptive controller

Approximation-based controllers are needed where the system dynamic model, or one of its terms, is not known for certain. In these situations, the CT controllers developed earlier will not perform well. In real life applications, even having the dynamic model for the robot, some parameters, as friction coefficients, are unknown and may change with time. In these cases, approximation is needed to account for the nonlinear terms inherent in the real time applications.

Other control applications use simplified CT controllers and increase PD gain to account for all the nonlinear terms not included in the torque formulations. However, this may result in a large control signal and instability problems. Instead, this framework approximates the unknown robot function and terms. The controllers using this framework provide good performance, even in the presence of unknown dynamics and disturbances. Performance and stability can be mathematically proven and, thus, relied upon in applications. These techniques can be extended to more complicated control objectives, such as force control for grinding, polishing, and other applications, where straight PD methods are inadequate. More information may be found in Lewis, et al. [80].

In this section, an approximation-based adaptive controller will be developed for two case studies, namely, a two-link robot manipulator and a WMR, based on the filtered tracking error approximation. The effectiveness of the controller will be ascertained by developing dynamic simulation software for each of the two cases.

6.4.1 The tracking problem

The objective is to have the robot or robot manipulator follow a desired and defined trajectory or path, expressed in joint space $q_d(t)$. The tracking design problem can be described as finding a control input $\tau(t)$ to the motor that causes the robot or robot manipulator to follow the desired trajectory. A general framework for tracking control, which includes many adaptive, robust, learning, and neural network techniques, is the approximation-based technique, which will be presented in this section.

Given the desired trajectory, $q_d(t)$, define the tracking error, $e(t)$, and filtered tracking error, $r(t)$, by [80]:

$$e = q_d - q \tag{6.24}$$

$$r = \dot{e} + \Lambda e$$

where:

Λ: is a positive definite design parameter matrix, commonly selected as a diagonal matrix with large positive entries.

As long as the controller guarantees that the filtered error, $r(t)$, is bounded, the tracking error, $e(t)$, is bounded, and Eq. (6.24) is a stable system. It may be shown that:

$$\|e\| \le \frac{\|r\|}{\sigma_{\min}(\Lambda)} \tag{6.25}$$

$$\|\dot{e}\| \le \|r\|$$

where:

$\sigma_{\min}$: is the minimum singular value of Λ, and the 2-norm is used.

For controlling the robot manipulator, the desired trajectory, $q_d(t)$, is specified by the path planner or the design engineer. However, for controlling the navigation of a WMR, the desired trajectory is the best path to satisfy the requirements of the navigation task. In each case, the desired trajectory needs to satisfy the following boundedness assumptions [80]:

$$\begin{Vmatrix} q_d(t) \\ \dot{q}_d(t) \\ \ddot{q}_d(t) \end{Vmatrix} \leq q_b \qquad (6.26)$$

where:

q_b : is a scalar bound.

Differentiating Eq. (6.24) gives:

$$\dot{r} = \frac{d}{dt}(\dot{e} - \Lambda e) = \ddot{e} - \Lambda \dot{e} = \ddot{q}_d - \ddot{q} + \Lambda(\dot{q}_d - \dot{q}) \qquad (6.27)$$

Then multiplying by M:

$$M\dot{r} = M\ddot{e} - M\Lambda\dot{e} = M\ddot{q}_d - M\ddot{q} + M\Lambda\dot{e} \qquad (6.28)$$

Since:

$$\ddot{e} = \ddot{q}_d - \ddot{q} \qquad (6.29)$$

This can also be written as:

$$M\dot{r} = -V_m r + f(x) + \tau_d - \tau \qquad (6.30)$$

$$M\dot{r} = M\ddot{q}_d + M\Lambda\dot{e} - M\ddot{q}$$

$$M\dot{r} = M(\ddot{q}_d + \Lambda\dot{e}) + V_m\dot{q} + R\dot{q} + G + \tau_d - \tau$$

And defining $f(x)$ to represent the nonlinear robot function as [80]:

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + R(\dot{q}) + G(q) \qquad (6.31)$$

Since:

$$V_m\dot{q} = V_m\dot{q}_d - V_m\dot{e} \qquad (6.32)$$

$$V_m r = V_m\dot{e} + V_m\Lambda e$$

Adding these two equations gives:

$$V_m\dot{q} + V_m r = V_m\dot{q}_d + V_m\Lambda e = V_m(\dot{q}_d + \Lambda e) - V_m r = V_m\dot{q} - V_m(\dot{q}_d + \Lambda e) \qquad (6.33)$$

Hence, the filtered error is expressed as:

$$M\dot{r} = -V_m r + f(x) + \tau_d - \tau \tag{6.34}$$

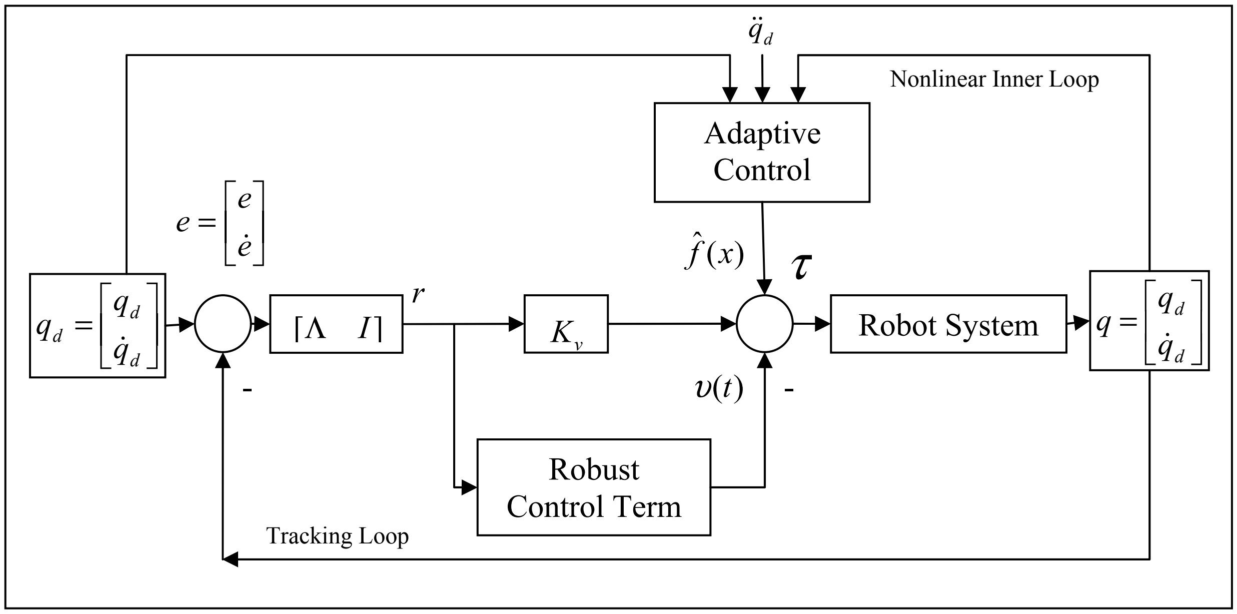
Where the vector x contains all the time signals needed to compute $f(.)$. Vector x can be defined as [80]:

$$x = \begin{bmatrix} e^T \\ \dot{e}^T \\ q_d^T \\ \dot{q}_d^T \\ \ddot{q}_d^T \end{bmatrix} \tag{6.35}$$

Hence, $f(x)$ contains all the potential unknown robot parameters, except for the term $V_m r$, which cancels out in Lyapunov proofs controller stability. It is important to note that in the approximation-based control approach described here, the correct version of V_m needs to be stated in the skew-symmetric form [80]. The approximation-based controller is shown in Fig. 65.6. A block diagram for the approximation-based controller can be constructed as follows [80]:

- The input is the desired position and velocity. The position is $q_d(t)$. These are compared with the actual position and velocity. The difference is the error signal and derivative. The error vector is filtered, or compensated, to give the filtered error, $r(t)$. Note that this is analogous to the PD compensator. Starting from the left side of Fig. 6.65, the error computation is:

$$\mathbf{e} = \begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \mathbf{q}_d - \mathbf{q} = \begin{bmatrix} q_d \\ \dot{q}_d \end{bmatrix} - \begin{bmatrix} q \\ \dot{q} \end{bmatrix} \tag{6.36}$$



$\ddot{q}_d$
Nonlinear Inner Loop
Adaptive Control
$e = \begin{bmatrix} e \\ \dot{e} \end{bmatrix}$
$\hat{f}(x)$
τ
r
$q_d = \begin{bmatrix} q_d \\ \dot{q}_d \end{bmatrix}$
$[\Lambda \quad I]$
K_v
Robot System
$q = \begin{bmatrix} q_d \\ \dot{q}_d \end{bmatrix}$
-
$\upsilon(t)$
-
Robust Control Term
Tracking Loop

Figure 6.65: Filtered error approximation-based controller [80].

- This error signal is filtered, or compensated, by the linear filter block to compute the filtered term, $r(t)$:

$$r = [\Lambda \quad I]\begin{bmatrix} e \\ \dot{e} \end{bmatrix} = \dot{e} + \Lambda e \tag{6.37}$$

- Now the architecture depends on the type of controller approximation. A general type of approximation-based controllers is derived by setting the torque equal to the estimate of the nonlinear robot function, $\hat{f}$, plus the filtered error multiplied by a gain, and minus a robust control term, as shown below [80]:

$$\tau = \hat{f} + K_v r - \upsilon(t) \tag{6.38}$$

where:

$$K_v r = K_v \dot{e} + K_v \Lambda e$$

- The robustifing signal, $\upsilon(t)$, is an auxiliary signal, which is added to provide robustness to counteract disturbances and modeling errors. The estimates of $\hat{f}$ and $\upsilon(t)$ are defined differently for adaptive, robust controller, fuzzy logic, and neural network controllers.

6.4.2 Error Dynamics

Lewis, et al. [80] used nonlinear stability proofs, based on Lyapunov, or passivity techniques, to show that tracking error stability can be assured by selecting one of a variety of specific controllers. The controllers and proofs of stability are derived from closed-loop error dynamics, which, in turn, are derived by substituting the approximation-based controller equation into the filtered error equation to give [80]:

$$M\dot{r} = -V_m r + f(x) + \tau_d - (\hat{f} + K_v r - \upsilon(t)) \tag{6.39}$$

Defining the function approximation error as [80]:

$$f_e = f - \hat{f} \tag{6.40}$$

$$M\dot{r} = -V_m r - K_v r + f_e + \tau_d + \upsilon(t)$$

Note that the tracking error dynamics are disturbed by the functional approximation error. The controller design problem is to keep the error dynamics stable by the proper selection of the estimate of $f(x)$, $\hat{f}(x)$, and the robust term, $\upsilon(t)$. Then the filtered tracking error is bounded, which implies that the tracking error will be bounded. Consequently, the WMR, or robot manipulator, follows the desired trajectory, $q_d(t)$. Several specific controllers assuring stable tracking can be implemented [80].

If the nonlinear function, $f(x)$, is known, it is better to use that instead of the $\hat{f}(x)$ in Eq. (6.38):

$$\tau = K_v r + M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + R(\dot{q}) + G(q) \tag{6.41}$$

Which is equivalent to the CT control equations described in an earlier section.

6.4.3 Adaptive Controller

By performing on-line tuning of parameters, adaptive controllers were successful in dealing with modeling uncertainties in general linear and nonlinear systems. The model-reference approach, hyper-stability techniques, self-tuning regulators, and gradient-based techniques were used in adaptive control. Some adaptive control applications rely on the linear-in-the-parameters (LIP) assumption [80].

The LIP assumption is stated by Lewis, et al. [80] as: "The nonlinear robot function is linear in the unknown parameters such as masses and friction coefficients so that one can write":

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + R(\dot{q}) + G(q) = W(x)\Phi \tag{6.42}$$

where:

$W(x)$: is a matrix of known robot functions;

Φ: is a vector of unknown parameters, such as masses and friction coefficients.

One adaptive controller given by Slotine as cited in Lewis, et al. [80] is:

$$\tau = W(x)\hat{\Phi} + K_v r \tag{6.43}$$

$$\hat{\Phi} = \Gamma W^T(x) r$$

where:

Γ: is a tuning parameter matrix, usually selected as a diagonal matrix with positive elements.

Dynamic on-line tuning is used by adaptive controller manufacturers to estimate the unknown parameter vector, Φ. Therefore, the controller has its own dynamics. $\hat{\Phi}$ is used in the estimate of the nonlinear function, $\hat{f}(x)$, as [80]:

$$\hat{f}(x) = W(x)\hat{\Phi} \tag{6.44}$$

The adaptive controller, having the same structure described earlier in Fig. 6.65, can be modeled according Eq. (6.43), based on the following assumptions [80]:

1. No torque disturbances ($\tau_d(t)$=0).

2. The desired trajectory, $q_d(t)$, is bounded.
3. The LIP assumption holds.
4. The unknown vector parameter, Φ, is constant.

Then the tracking error, $r(t)$, goes to zero, and the estimate for the unknown vector parameter, $\hat{\Phi}$, is uniform ultimate bounded. For the proof, please refer to Lewis, et al. [80].

However, in real life applications, although there may be some torque disturbances and a small tracking error, $r(t)$, the result of the theorem is still adequate. It is important to note that the performance of the adaptive controller depends on the regression matrix, $W(x)$. Hence, if $W(x)$ is not precisely known, the adaptive controller will not perform well [80].

6.4.4 Developing an adaptive controller for the two-link manipulator

6.4.4.1 Controller architecture

The filtered-error approximation-based adaptive controller for a two-link manipulator can be developed from Eq. (6.43). The regression matrix, $W(x)$, can be derived by using the two-link manipulator dynamics. Lewis, et al. [80] developed the regression matrix for the two-link manipulator, based on the dynamics described in Eq. (6.9):

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) = W(x)\Phi \tag{6.45}$$

$$\hat{f}(x) = W(x)\hat{\Phi}$$

$$W(x) = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

$$\begin{aligned}
&W_{11} = a_1^2(\ddot{q}_{d1} + \lambda_1\dot{e}_1) + a_1 g\cos q_1 \\
&W_{12} = (a_2^2 + 2a_1a_2\cos q_2 + a_1^2)(\ddot{q}_{d1} + \lambda_1\dot{e}_1) + (a_2^2 + a_1a_2\cos q_2)(\ddot{q}_{d2} + \lambda_2\dot{e}_2) \\
&-a_1a_2\dot{q}_2(\dot{q}_{d1}\sin q_2 + \lambda_1 e_1) - a_1a_2(\dot{q}_1 + \dot{q}_2)(\dot{q}_{d2}\sin q_2 + \lambda_2 e_2) + a_2 g\cos(q_1 + q_2) + a_1 g\cos q_1 \\
&W_{21} = 0 \\
&W_{22} = (a_2^2 + a_1a_2\cos q_2)(\ddot{q}_{d1} + \lambda_1\dot{e}_1) + a_2^2(\ddot{q}_{d2} + \lambda_2\dot{e}_2) + a_1a_2(\dot{q}_{d1} + \lambda_1 e_1)\sin q_2 + a_2 g\cos(q_1 + q_2)
\end{aligned}$$

$$\hat{\Phi} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$

6.4.4.2 Simulation of the adaptive controller for the two-link manipulator

Simulation software for the adaptive controller is developed for the two-link manipulator by using Lewis, et al. [80] architecture. The simulation parameters are shown in Table 6.9:

Two-link manipulator parameters	
a_1	1 m
a_2	1 m
m_1	0.8 kg
m_2	2.3 kg
Controller parameters	
K_v	$\begin{bmatrix} 20 & 0 \\ 0 & 20 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$
Desired trajectory	
$q_{d1}(t)$	$\sin t$
$q_{d2}(t)$	$\cos t$

Table 6.9: Adaptive controller simulation parameters for the two-link manipulator.

Two experiments are conducted: one, for the correct dynamic model, and a second, using incorrect dynamics. The results are:

Case I: The first experiment uses the two-link dynamic model, shown in Eq. (6.9). The results are shown in Figs. 6.66-6.68. As shown in Fig. 6.66, the tracking errors start at 1, and reach zero by the third time unit. Thus, the actual joint angles match the desired joint angles by the third time unit, as shown in Fig. 6.67. Fig. 6.68 shows that the adaptive controller succeeds in estimating m_1 at around 0.7 kg, which is close to the actual value (0.8 kg). However, the adaptive controller estimates m_2 at around 1.5 kg, while the actual value is 2.3 kg. In general, the performance of the adaptive controller for the two-link manipulator is good, and the results are acceptable.

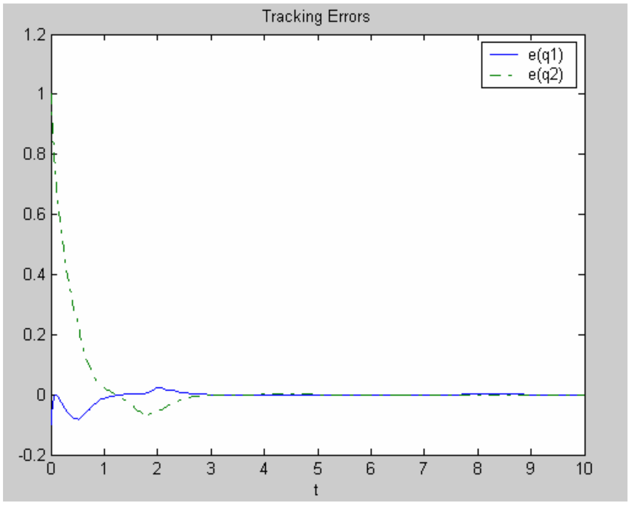


Figure 6.66: Adaptive controller tracking errors.

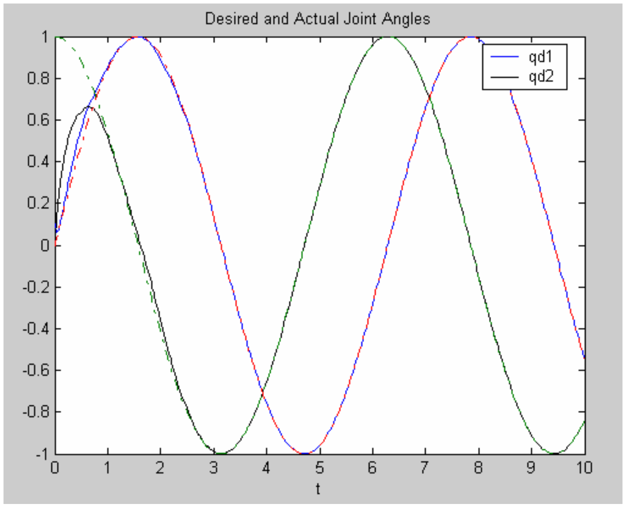


Figure 6.67: Adaptive controller desired versus actual join angles.

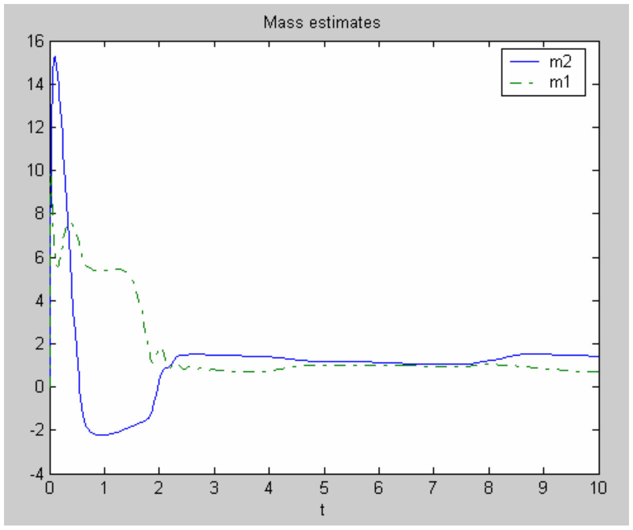


Figure 6.68: Adaptive controller parameters estimate.

Case II: The results of case I show that the adaptive controller performance is good when using the correct dynamic model. In this experiment, a simulation uses an incorrect regression model, as shown in the following equation [80]:

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + G(q) = W(x)\Phi \tag{6.46}$$

$$\hat{f}(x) = W(x)\hat{\Phi}$$

$$W(x) = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}$$

$$\begin{aligned}
W_{11} &= a_1^2(\ddot{q}_{d1} + \lambda_1\dot{e}_1) \\
W_{12} &= (a_2^2 + 2a_1a_2\cos q_2 + a_1^2)(\ddot{q}_{d1} + \lambda_1\dot{e}_1) + (a_2^2 + a_1a_2\cos q_2)(\ddot{q}_{d2} + \lambda_2\dot{e}_2) \\
&- a_1a_2\dot{q}_2(\dot{q}_{d1}\sin q_2 + \lambda_1 e_1) - a_1a_2(\dot{q}_1 + \dot{q}_2)(\dot{q}_{d2}\sin q_2 + \lambda_2 e_2) + a_2 g\cos(q_1 + q_2) + a_1 g\cos q_1 \\
W_{21} &= 0 \\
W_{22} &= (a_2^2 + a_1a_2\cos q_2)(\ddot{q}_{d1} + \lambda_1\dot{e}_1) + a_2^2(\ddot{q}_{d2} + \lambda_2\dot{e}_2) + a_1a_2(\dot{q}_{d1} + \lambda_1 e_1)\sin q_2 + a_2 g\cos(q_1 + q_2)
\end{aligned}$$

$$\hat{\Phi} = \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}$$

Where the second term of W_{11} has been deleted. The results of the simulation are shown in Figs. 6.69-6.71. As shown in Fig. 6.69, the tracking errors are high and the actual joint angles do not match the desired joint angles, as shown in Fig. 6.70. Fig. 6.71 shows that the adaptive controller cannot estimate m_1 and m_2. Previous research was conducted to improve the performance of the adaptive controller in dealing with unmodelled dynamics, as reported by Lewis, et al. [80]. They suggested using *e*-modification, σ-modification, and dead zone techniques. For more information, please refer to Lewis, et al. [80].

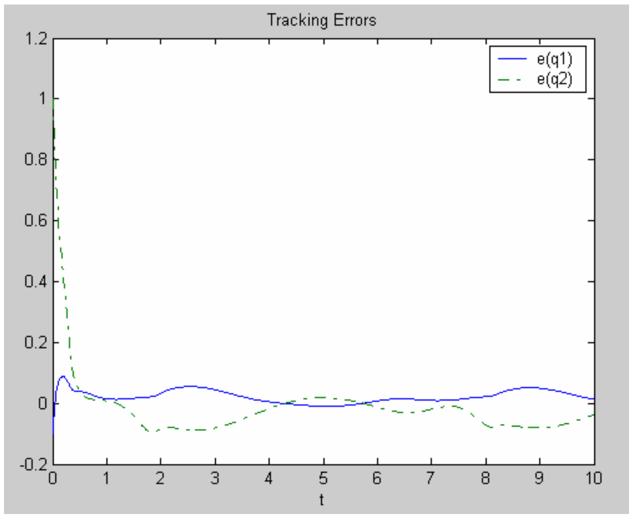


Figure 6.69: Adaptive controller tracking errors.

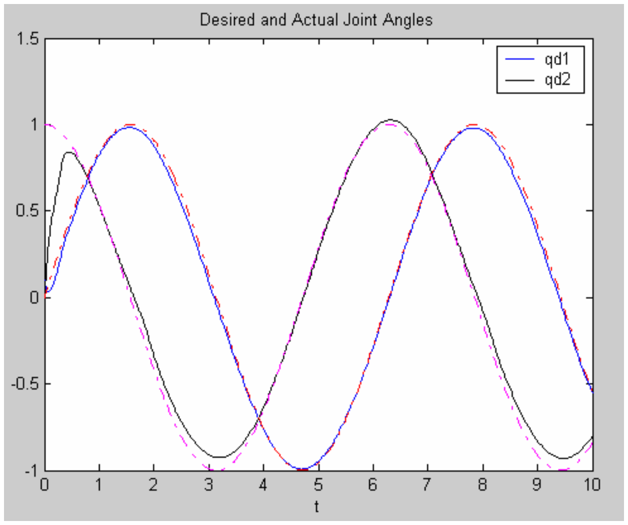


Figure 6.70: Desired versus actual joint angles.

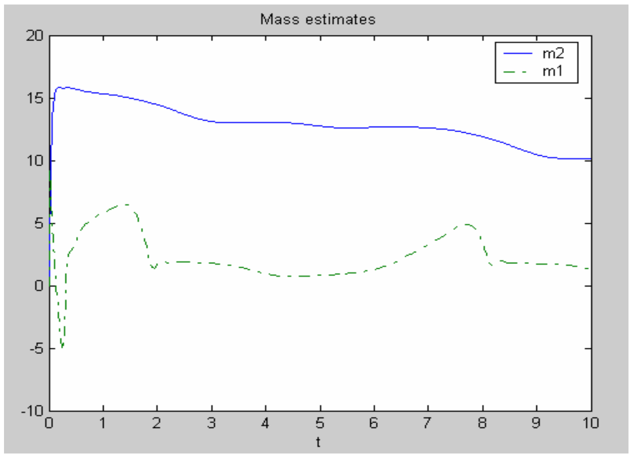


Figure 6.71: Parameters estimate.

6.4.5 Developing an adaptive controller for WMR navigation

6.4.5.1 Controller architecture

The filtered-error approximation-based adaptive controller for WMR navigation is developed from Eq. (6.43). The regression matrix, $W(x)$, can be derived from the WMR dynamics, presented in chapter 3. The dynamic model is described in Eq. (3.62):

$M(q)\ddot{q} + J(q,\dot{q})\dot{q} + F = \tau$

where:

$$q = \begin{bmatrix} x_c \\ y_e \\ \theta \end{bmatrix}$$

$$M(q) = \begin{bmatrix} \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2 ed\sin^2\theta - mr^2 ed\sin\theta\cos\theta + J_c r^2 + 2J_0 d^2)}{2rd} \\ \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2 ed\sin^2\theta - mr^2 ed\sin\theta\cos\theta - J_c r^2 - 2J_0 d^2)}{2rd} \end{bmatrix}$$

$$J(q,\dot{q}) = \begin{bmatrix} \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \\ \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta + \cos\theta)}{2} \end{bmatrix}$$

$$F = \begin{bmatrix} \frac{-f_n er}{d} \\ \frac{-f_n er}{d} \end{bmatrix}$$

$$\tau = \begin{bmatrix} \tau_r \\ \tau_l \end{bmatrix}$$

(6.47)

The regression matrix, $WR(x)$, for WMR navigation needs to be developed from Eq. (6.42) and WMR dynamics:

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{E}) + J(q,\dot{q})(\dot{q}_d + \Lambda E) + F = WR(x)\psi \quad (6.48)$$

$$\hat{f}(x) = WR(x)\hat{\psi}$$

where:

$$x = \begin{bmatrix} E^T \\ \dot{E}^T \\ q_d^T \\ \dot{q}_d^T \\ \ddot{q}_d^T \end{bmatrix}$$

Since $M(q)$ is a 2x3 matrix and $(\ddot{q}_d + \Lambda\dot{E})$ is a 3x1 matrix, $M(q)(\ddot{q}_d + \Lambda\dot{E})$ is a 2x1 matrix. The same applies for the $J(q,\dot{q})(\dot{q}_d + \Lambda E)$ and the F matrices. Thus, $f(x)$ is a 2x1 matrix. The regression matrix and the vector of the unknown robot parameters, $WR(x)\hat{\psi}$, need to be a 2x1 matrix. There are many options for that. However, it is best to choose the $WR(x)$ to be a 2x2 matrix and $\hat{\psi}$ to be a 2x1 matrix. The most important unknown robot parameter affecting the whole dynamic model is its mass m. Hence, $\hat{\psi}$ can be set as follows:

$$\hat{\psi} = \begin{bmatrix} m \\ 1 \end{bmatrix} \quad (6.49)$$

While the regression matrix, $WR(x)$, needs to be:

$$WR(x) = \begin{bmatrix} WR_{11} & WR_{12} \\ WR_{21} & WR_{22} \end{bmatrix} \quad (6.50)$$

Now, the $WR_{11}, WR_{12}, WR_{21}, WR_{22}$ matrices must satisfy the following:

$$f(x) = WR(x)\psi$$

where:

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{E}) + J(q,\dot{q})(\dot{q}_d + \Lambda E) + F$$

$$q = \begin{bmatrix} x_c \\ y_e \\ \theta \end{bmatrix}$$

$$M(q) = \begin{bmatrix} \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2 ed\sin^2\theta - mr^2 ed\sin\theta\cos\theta + J_c r^2 + 2J_0 d^2)}{2rd} \\ \frac{(mr^2\cos\theta + 2J_0\cos\theta)}{2r} & \frac{(mr^2\sin\theta + 2J_0\sin\theta)}{2r} & \frac{(mr^2 ed\sin^2\theta - mr^2 ed\sin\theta\cos\theta - J_c r^2 - 2J_0 d^2)}{2rd} \end{bmatrix}$$

$$J(q,\dot{q}) = \begin{bmatrix} \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2} \\ \frac{-J_0\dot{\theta}\sin\theta}{r} & \frac{J_0\dot{\theta}\cos\theta}{r} & \frac{-mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2} \end{bmatrix}$$

$$F = \begin{bmatrix} \frac{-f_n er}{d} \\ \frac{-f_n er}{d} \end{bmatrix}$$

(6.51)

To find the values of the $WR_{11}, WR_{12}, WR_{21}, WR_{22}$ matrices, set $(\dot{q}_d + \Lambda E)$ as matrix S. Hence, $(\ddot{q}_d + \Lambda\dot{E})$ is matrix $\dot{S}$, where both S and $\dot{S}$ are:

$$S = (\dot{q}_d + \Lambda E) = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} \dot{q}_{d1} + \lambda_1 e_1 \\ \dot{q}_{d2} + \lambda_2 e_2 \\ \dot{q}_{d3} + \lambda_3 e_3 \end{bmatrix} \tag{6.52}$$

$$\dot{S} = (\ddot{q}_d + \Lambda\dot{E}) = \begin{bmatrix} \dot{S}_1 \\ \dot{S}_2 \\ \dot{S}_3 \end{bmatrix} = \begin{bmatrix} \ddot{q}_{d1} + \lambda_1 \dot{e}_1 \\ \ddot{q}_{d2} + \lambda_2 \dot{e}_2 \\ \ddot{q}_{d3} + \lambda_3 \dot{e}_3 \end{bmatrix}$$

Hence, $\hat{f}(x)$ can be defined as:

$$\hat{f}(x) = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \end{bmatrix} \cdot \begin{bmatrix} \dot{S}_1 \\ \dot{S}_2 \\ \dot{S}_3 \end{bmatrix} + \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \end{bmatrix} \cdot \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} + \begin{bmatrix} F_1 \\ F_2 \end{bmatrix} \tag{6.53}$$

Expanding $\hat{f}(x)$:

$$\hat{f}(x)=\begin{bmatrix}\hat{f}_1\\ \hat{f}_2\end{bmatrix}=\begin{bmatrix}M_{11}\dot{S}_1+M_{12}\dot{S}_2+M_{13}\dot{S}_3+J_{11}S_1+J_{12}S_2+J_{13}S_3+F_1\\ M_{21}\dot{S}_1+M_{22}\dot{S}_2+M_{23}\dot{S}_3+J_{21}S_1+J_{22}S_2+J_{23}S_3+F_2\end{bmatrix}=WR(x)\hat{\Psi} \tag{6.54}$$

Therefore:

$$\begin{bmatrix}M_{11}\dot{S}_1+M_{12}\dot{S}_2+M_{13}\dot{S}_3+J_{11}S_1+J_{12}S_2+J_{13}S_3+F_1\\ M_{21}\dot{S}_1+M_{22}\dot{S}_2+M_{23}\dot{S}_3+J_{21}S_1+J_{22}S_2+J_{23}S_3+F_2\end{bmatrix}=\begin{bmatrix}WR_{11} & WR_{12}\\ WR_{21} & WR_{22}\end{bmatrix}\cdot\begin{bmatrix}\hat{\Psi}_1\\ \hat{\Psi}_2\end{bmatrix}=\begin{bmatrix}WR_{11}\hat{\Psi}_1+WR_{12}\hat{\Psi}_2\\ WR_{21}\hat{\Psi}_1+WR_{22}\hat{\Psi}_2\end{bmatrix}$$

$$\begin{aligned}M_{11}\dot{S}_1+M_{12}\dot{S}_2+M_{13}\dot{S}_3+J_{11}S_1+J_{12}S_2+J_{13}S_3+F_1&=WR_{11}\hat{\Psi}_1+WR_{12}\hat{\Psi}_2\\ M_{21}\dot{S}_1+M_{22}\dot{S}_2+M_{23}\dot{S}_3+J_{21}S_1+J_{22}S_2+J_{23}S_3+F_2&=WR_{21}\hat{\Psi}_1+WR_{22}\hat{\Psi}_2\end{aligned} \tag{6.55}$$

Substituting the values of $\hat{\Psi}_1$ and $\hat{\Psi}_2$:

$$\begin{aligned}M_{11}\dot{S}_1+M_{12}\dot{S}_2+M_{13}\dot{S}_3+J_{11}S_1+J_{12}S_2+J_{13}S_3+F_1&=WR_{11}m+WR_{12}\\ M_{21}\dot{S}_1+M_{22}\dot{S}_2+M_{23}\dot{S}_3+J_{21}S_1+J_{22}S_2+J_{23}S_3+F_2&=WR_{21}m+WR_{22}\end{aligned} \tag{6.56}$$

Substituting the values of $M_{11},M_{12},M_{13},M_{21},M_{22},M_{23}$, $J_{11},J_{12},J_{13},J_{21},J_{22},J_{23}$ and F_1,F_2:

$$\begin{aligned}&\frac{(mr^2\cos\theta+2J_0\cos\theta)}{2r}\dot{S}_1+\frac{(mr^2\sin\theta+2J_0\sin\theta)}{2r}\dot{S}_2+\frac{(mr^2ed\sin^2\theta-mr^2ed\sin\theta\cos\theta+J_cr^2+2J_0d^2)}{2rd}\dot{S}_3+\\ &\frac{-J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2+\frac{-mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3+\frac{-f_ner}{d}=WR_{11}m+WR_{12}.\\ &\frac{(mr^2\cos\theta+2J_0\cos\theta)}{2r}\dot{S}_1+\frac{(mr^2\sin\theta+2J_0\sin\theta)}{2r}\dot{S}_2+\frac{(mr^2ed\sin^2\theta-mr^2ed\sin\theta\cos\theta-J_cr^2-2J_0d^2)}{2rd}\dot{S}_3+\\ &\frac{-J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2+\frac{-mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3+\frac{-f_ner}{d}=WR_{21}m+WR_{22}.\end{aligned} \tag{6.57}$$

More simplification:

$$\frac{mr^2\cos\theta}{2r}\dot{S}_1+\frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{mr^2\sin\theta}{2r}\dot{S}_2+\frac{2J_0\sin\theta}{2r}\dot{S}_2+\frac{mr^2ed\sin^2\theta-mr^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3+$$
$$\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3-\frac{f_ner}{d}=WR_{11}m+WR_{12}.$$
$$\frac{mr^2\cos\theta}{2r}\dot{S}_1+\frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{mr^2\sin\theta}{2r}\dot{S}_2+\frac{2J_0\sin\theta}{2r}\dot{S}_2+\frac{mr^2ed\sin^2\theta-mr^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-$$
$$\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3-\frac{f_ner}{d}=WR_{21}m+WR_{22}.$$

(6.58)

It is now clear that:

$$\frac{mr^2\cos\theta}{2r}\dot{S}_1+\frac{mr^2\sin\theta}{2r}\dot{S}_2+\frac{mr^2ed\sin^2\theta-mr^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-\frac{mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3=WR_{11}m.$$
$$\frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{2J_0\sin\theta}{2r}\dot{S}_2+\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{f_ner}{d}=WR_{12}.$$
$$\frac{mr^2\cos\theta}{2r}\dot{S}_1+\frac{mr^2\sin\theta}{2r}\dot{S}_2+\frac{mr^2ed\sin^2\theta-mr^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-\frac{mre\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3=WR_{21}m.$$
$$\frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{2J_0\sin\theta}{2r}\dot{S}_2-\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{f_ner}{d}=WR_{22}.$$

(6.59)

Thus, the $WR_{11},WR_{12},WR_{21},WR_{22}$ are:

$$WR_{11}=\frac{r^2\cos\theta}{2r}\dot{S}_1+\frac{r^2\sin\theta}{2r}\dot{S}_2+\frac{r^2ed\sin^2\theta-r^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-\frac{re\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3$$
$$WR_{12}=\frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{2J_0\sin\theta}{2r}\dot{S}_2+\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{f_ner}{d}$$
$$WR_{21}=\frac{r^2\cos\theta}{2r}\dot{S}_1+\frac{r^2\sin\theta}{2r}\dot{S}_2+\frac{r^2ed\sin^2\theta-r^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-\frac{re\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3$$
$$WR_{22}=\frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{2J_0\sin\theta}{2r}\dot{S}_2-\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{f_ner}{d}$$

(6.60)

Hence, the regression matrix for WMR, $WR(x)$ is:

$$WR(x)=\begin{bmatrix}\frac{r^2\cos\theta}{2r}\dot{S}_1+\frac{r^2\sin\theta}{2r}\dot{S}_2+\frac{r^2ed\sin^2\theta-r^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-\frac{re\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3 & \frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{2J_0\sin\theta}{2r}\dot{S}_2+\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{f_ner}{d}\\ \frac{r^2\cos\theta}{2r}\dot{S}_1+\frac{r^2\sin\theta}{2r}\dot{S}_2+\frac{r^2ed\sin^2\theta-r^2ed\sin\theta\cos\theta}{2rd}\dot{S}_3-\frac{re\dot{\theta}\cos\theta(\sin\theta+\cos\theta)}{2}S_3 & \frac{2J_0\cos\theta}{2r}\dot{S}_1+\frac{2J_0\sin\theta}{2r}\dot{S}_2-\frac{J_cr^2+2J_0d^2}{2rd}\dot{S}_3-\frac{J_0\dot{\theta}\sin\theta}{r}S_1+\frac{J_0\dot{\theta}\cos\theta}{r}S_2-\frac{f_ner}{d}\end{bmatrix}$$

(6.61)

6.4.5.2 Simulation of the adaptive controller for WMR navigation

Simulation software for the adaptive controller is developed for WMR navigation. Using the dynamic model and the controller model presented earlier, the desired trajectory taken is the output of the navigation algorithm, described in chapter 5. The robot parameters used in the simulation are shown in Table 6.2, which are according to Bearcat III.

6.4.5.2.1 Simulation results

Two sets of experiments are performed. In the first set, a sinusoidal trajectory is used, whereas a quadratic trajectory is used for the second set.

Case I: The simulation parameters used in this experiment are shown in Table 6.10:

Controller parameters	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$
Desired trajectory	
$q_{d1}(t)$	$\sin t$
$q_{d2}(t)$	$\cos t$
$q_{d3}(t)$	$\sin t$

Table 6.10: Adaptive controller simulation parameters for WMR navigation.

The results of the experiment are shown in Figs. 6.72-6.74.

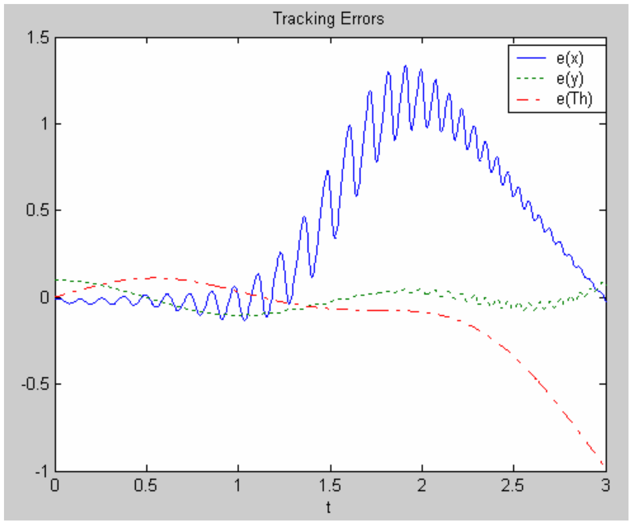


Figure 6.72: Adaptive controller tracking errors.

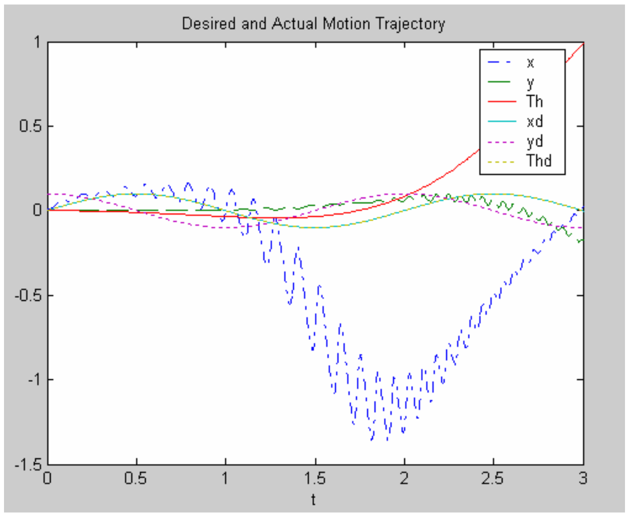


Figure 6.73: Adaptive controller desired versus actual motion trajectories.

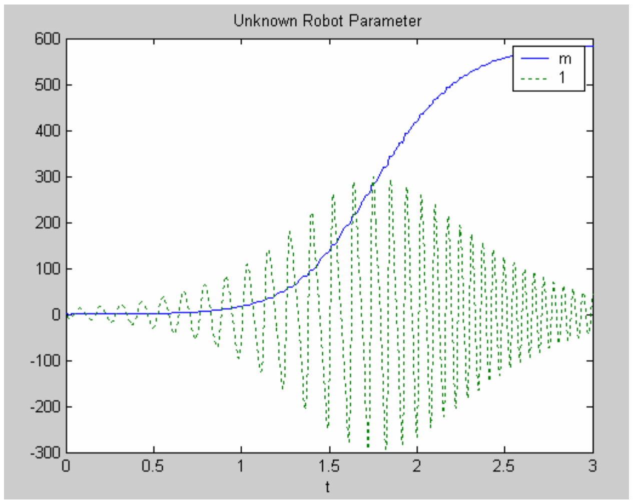


Figure 6.74: Adaptive controller parameters estimate.

The first thing to note is that the adaptive controller runs only for 3 time units, instead of 10, as was specified by the simulation program. This is due to the integrator function, which is function ode23. Function ode 45 is used, also, but similar results are achieved where the integrator is not able to continue running for the entire time interval. Matlab service personnel were contacted for this problem. They said that this is due to the complexity of the calculations.

This is unfortunate, since better results would be obtained if more time were given to the controller.

The controller was able to approximate the unknown robot parameters. The first unknown robot parameter (mass) increases from zero to around 600 kg, and the second unknown robot parameter (1) oscillates around one, which is its actual value.

The tracking errors are in the range of -1-1.4, which is not too high. The actual motion trajectories follow the desired motion trajectories for part of the time.

Case II: A quadratic trajectory is used in this set of experiments to test the performance of the controller for a different kind of path and to simplify the calculations for the integrator function, so it will work longer. The desired motion trajectories are shown in Table 6.11:

Desired trajectory	
$q_{d1}(t)$	$.01t^2$
$q_{d2}(t)$	$.01t^2 + 0.01t$
$q_{d3}(t)$	$\sin t$

Table 6.11: Adaptive controller desired motion trajectories for case II.

- In the first experiment of this set, the following controller parameters are used:

Controller parameters	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$

Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

Table 6.12: Adaptive controller simulation parameters for WMR navigation.

The results of the experiment are shown in Figs. 6.75-6.77. The integrator works for six time units for this path. In the first figure, the tracking errors are around zero for the first two time units. Then they start to increase to 1.6. However, it appears that they begin reducing again after five time units. First, the actual motion trajectories follow the desired motion trajectories. Then they move far apart from each other. In the final stage, they follow each other again. The controller is able to approximate the unknown robot parameters, where the first unknown robot parameter (mass) increases from zero to around 600 kg, while the second unknown robot parameter (1) oscillates around one, which is its actual value.

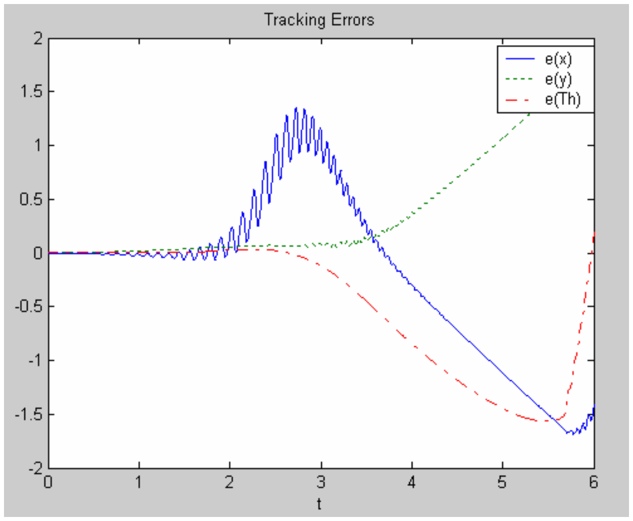


Figure 6.75: Adaptive controller tracking errors.

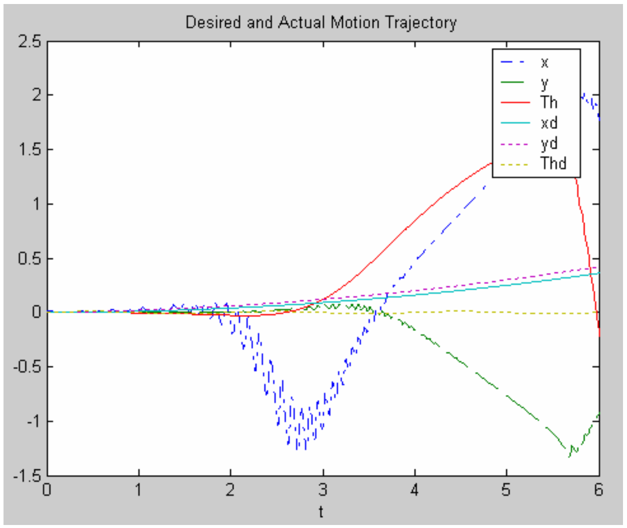


Figure 6.76: Adaptive controller desired versus actual motion trajectories.

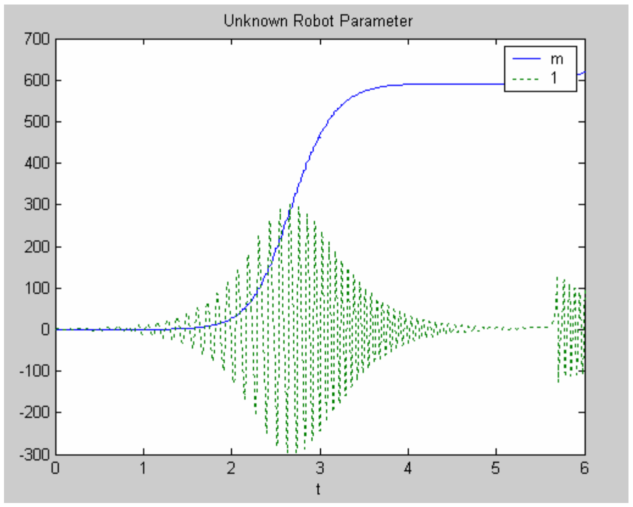


Figure 6.77: Adaptive controller parameters estimate.

- To study the effect of increasing the gain, K_v, a second experiment in this set is performed by using the following controller parameters:

Controller parameters	
K_v	$\begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

Table 6.13: Adaptive controller simulation parameters for WMR navigation.

The results of the experiment are shown in Figs. 6.78-6.80. The integrator works for 3.5 time units for this path. In the first figure, the tracking errors are around zero for the first two time units. They begin increasing, but start reducing again after three time units. First, the actual motion trajectories follow the desired motion trajectories. Then they move far apart from each other. In the final stage, they follow each other again. The controller can approximate the unknown robot parameters, where the first unknown robot parameter (mass) increases from zero to around 520 kg, while the second unknown robot parameter (1) oscillates around one, which is its actual value. This result is very close to the result of the first experiment, which means that increasing the value of K_v does not have that much effect on the performance of the controller.

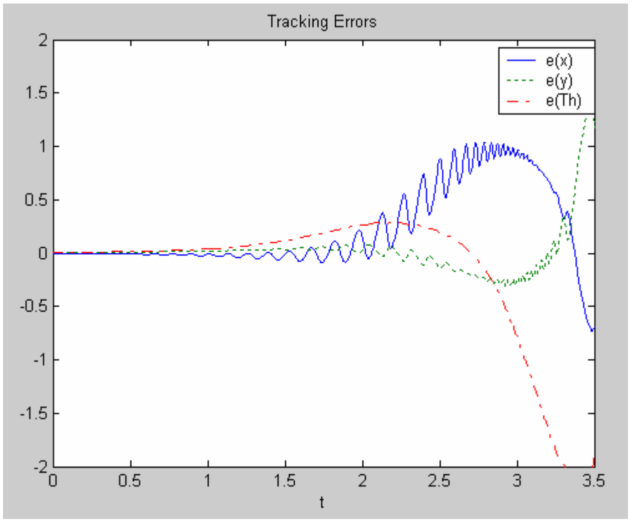


Figure 6.78: Adaptive controller tracking errors.

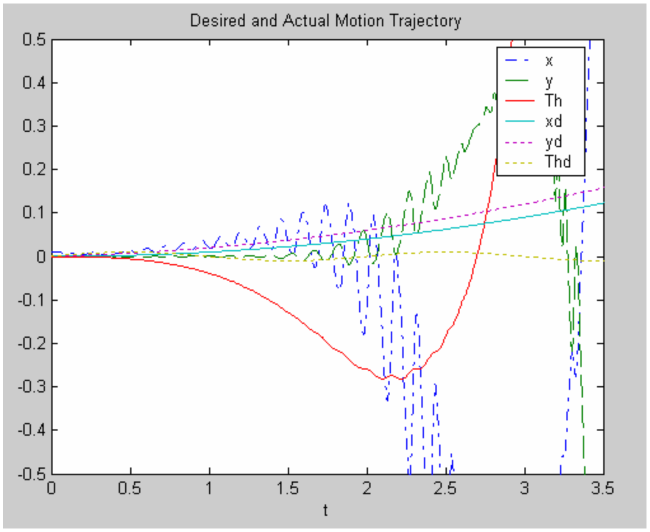


Figure 6.79: Adaptive controller desired versus actual motion trajectories.

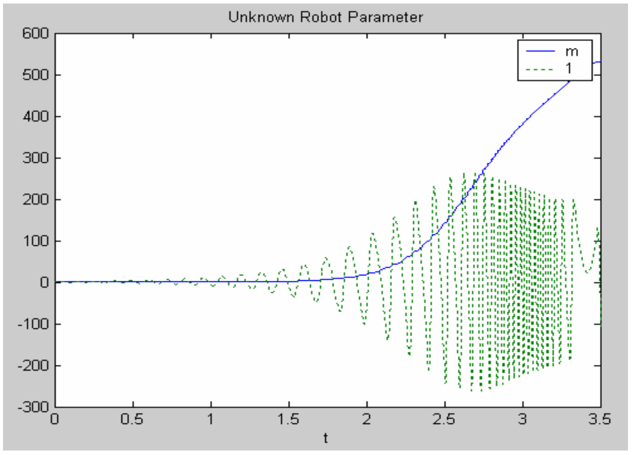


Figure 6.80: Parameters estimate.

- To study the effect of parameter Λ, a third experiment in this set is performed with lower values of Λ, shown in the following table:

Controller parameters	
K_v	$\begin{bmatrix} 20 & 0 & 0 \\ 0 & 20 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

Table 6.14: Adaptive controller simulation parameters for WMR navigation.

The results of this experiment are shown in Figs. 6.81-6.83. The integrator works for three and a half time units. In the first figure, the tracking errors increase from zero to around 0.8. First, the actual motion trajectories follow the desired motion trajectories. Then they move far apart from each other. The controller cannot approximate the unknown robot parameters in the case where the first unknown robot parameter (mass) remains at zero, while the second unknown robot parameter (1) oscillates around zero. This result is much worse than the results of the first two experiments. This tells us that the values of $\lambda_1, \lambda_2, \lambda_3$ should remain at 5, instead of 1.

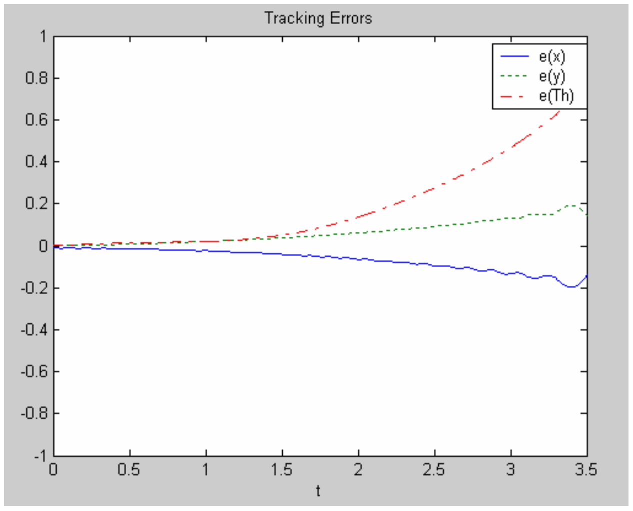


Figure 6.81: Adaptive controller tracking errors.

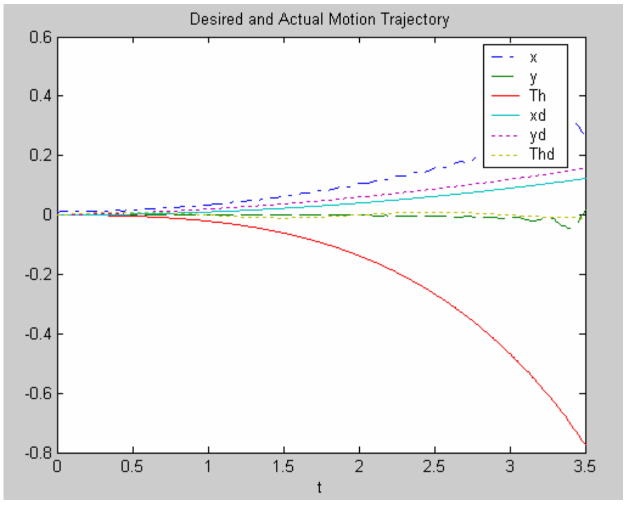


Figure 6.82: Adaptive controller desired versus actual motion trajectories.

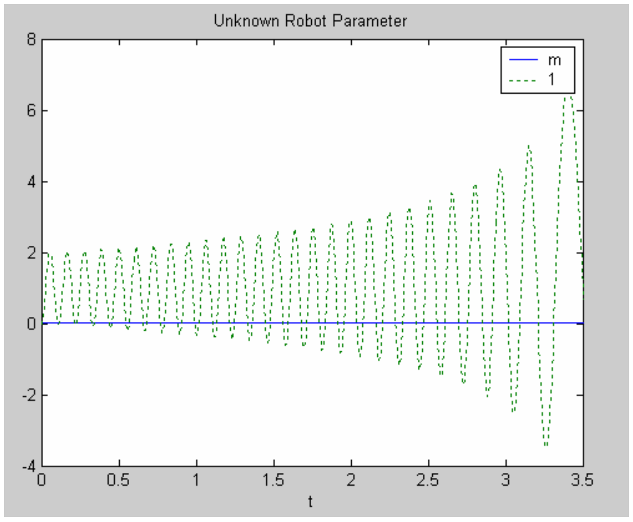


Figure 6.83: Parameters estimate.

- To study the effect of parameters K_v and Λ together, another experiment in this set is performed, with the following controller parameters:

Controller parameters	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$

Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

Table 6.15: Adaptive controller simulation parameters for WMR navigation.

The results of this experiment are shown in Figs. 6.84-6.86. The integrator works for seven time units in this case. In the first figure, the tracking errors increase from zero to around 0.8. First, the actual motion trajectories follow the desired motion trajectories. Then they move far apart from each other. The controller cannot approximate the unknown robot parameters where the first unknown robot parameter (mass) remains at zero, while the second unknown robot parameter (1) oscillates around zero. This result illustrates that decreasing the value of K_v does not improve the performance of the controller when the values of $\lambda_1, \lambda_2, \lambda_3$ equal 1.

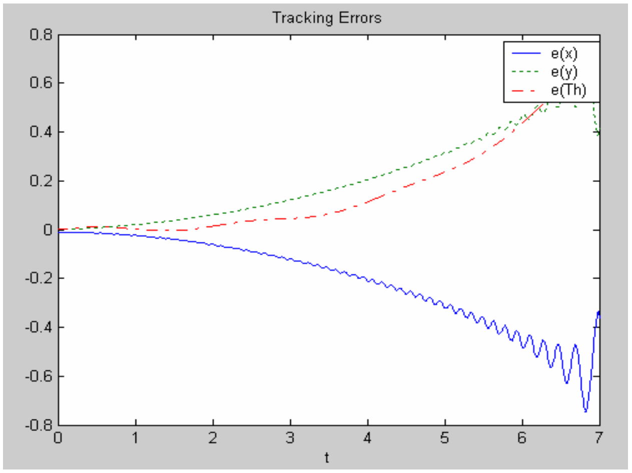


Figure 6.84: Adaptive controller tracking errors.

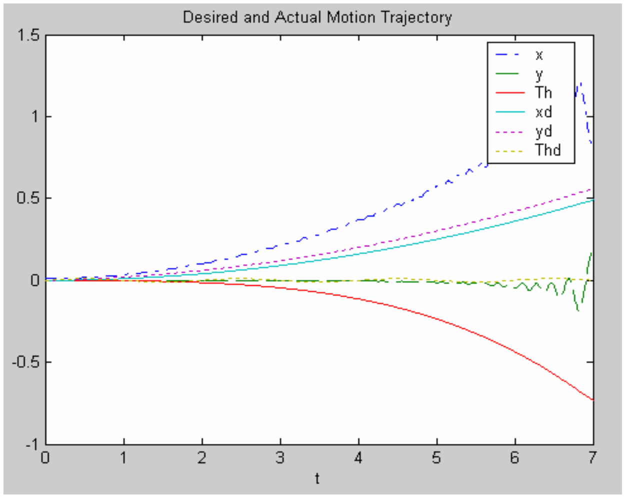


Figure 6.85: Adaptive controller desired versus actual motion trajectories.

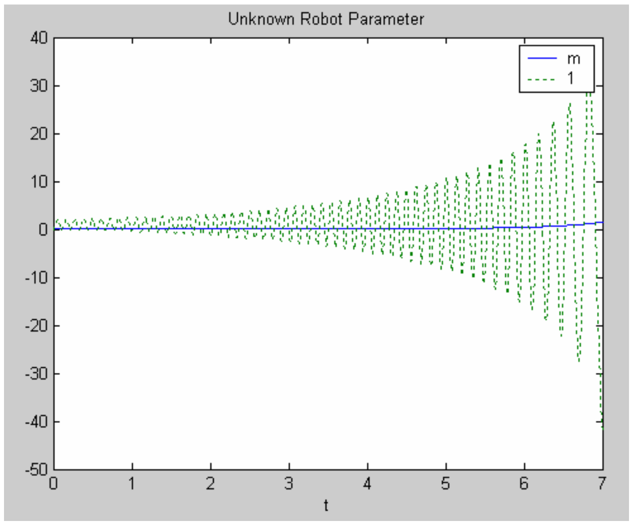


Figure 6.86: Adaptive controller parameters estimate.

- To study the effect of parameter Γ, another experiment in this set is performed with the best values for K_v and Λ and a lower value for Γ. The controller parameters are shown in table 6.16:

Controller parameters	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix}$

Table 6.16: Adaptive controller simulation parameters for WMR navigation.

The results of this experiment are shown in Figs. 6.87-6.89. The integrator works for five time units. In the first figure, the tracking errors are small in the first four time units.

Then the tracking errors for x and y increase rapidly to around 230. In the beginning, the actual motion trajectories follow the desired motion trajectories. Then the actual motion trajectories for x and y move far apart from the desired motion trajectories. However, the actual orientation, θ, continues to follow the desired orientation, θ_d. The controller cannot approximate the unknown robot parameters. This result illustrates that decreasing the value of Γ does not improve the performance of the controller.

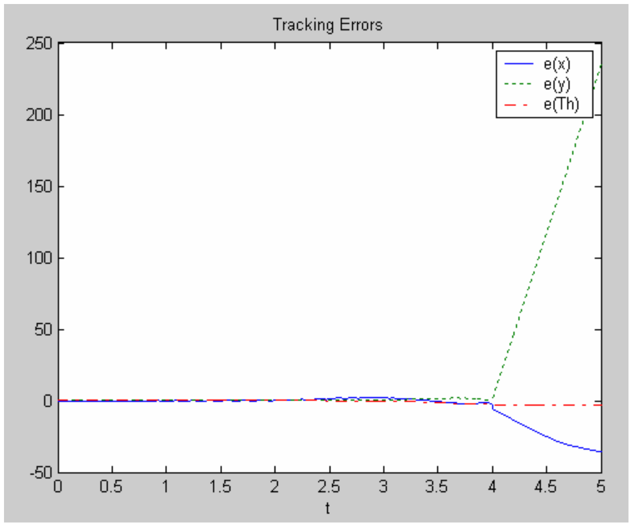


Figure 6.87: Adaptive controller tracking errors.

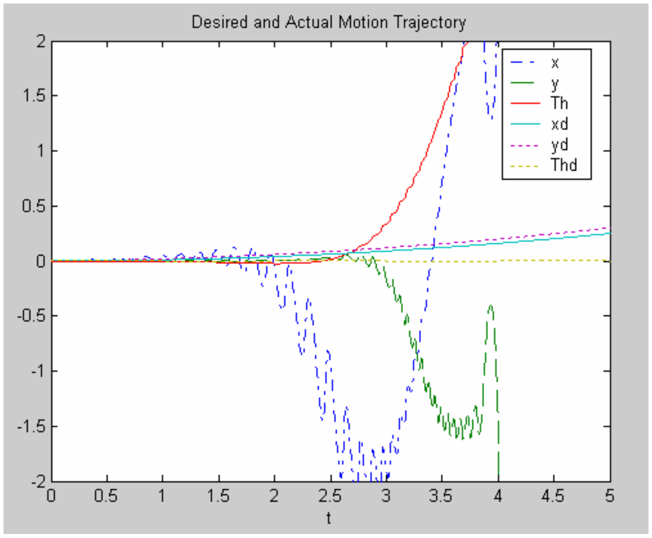


Figure 6.88: Adaptive controller desired versus actual motion trajectories.

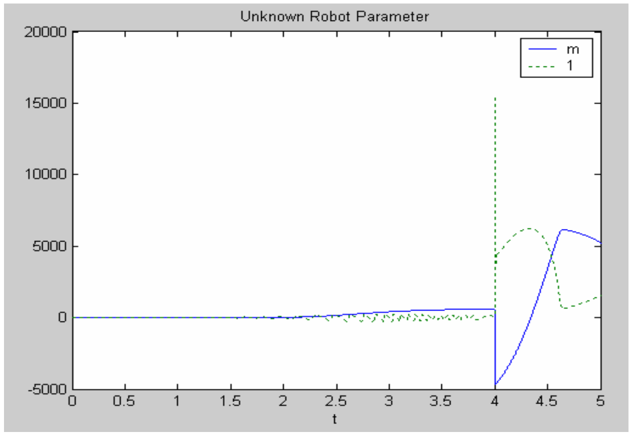


Figure 6.89: Parameters estimate.

- The last experiment shows that reducing parameter Γ will not improve the performance of the controller. However, what is the effect of increasing Γ on the performance of the controller? To study this, another experiment is performed with a higher value of Γ, as shown in table 6.17:

Controller parameters	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 15 & 0 \\ 0 & 15 \end{bmatrix}$

Table 6.17: Adaptive controller simulation parameters for WMR navigation.

The results of this experiment are shown in Figs. 6.90-6.92. The integrator works for six time units. The results are very similar to the results of the first experiment in this set, with the value of the Γ diagonal equal to 10, instead of 15. However, this experiment shows greater tracking errors. Also, the controller can approximate the unknown robot parameters. The results of this experiment illustrate that higher values of Γ can be used without impairing the performance of the controller.

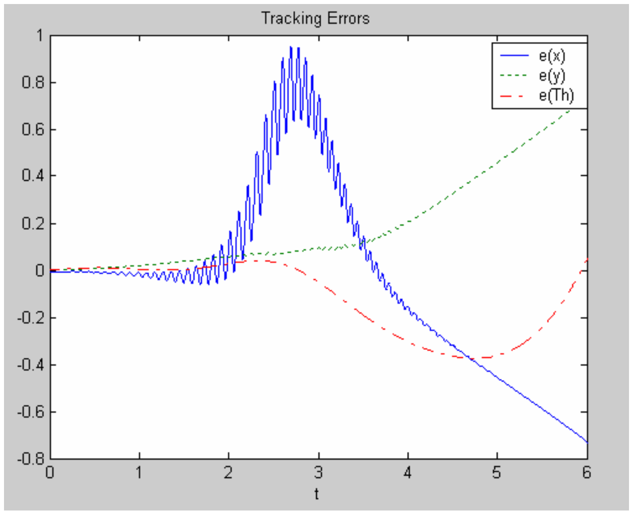


Figure 6.90: Adaptive controller tracking errors versus time.

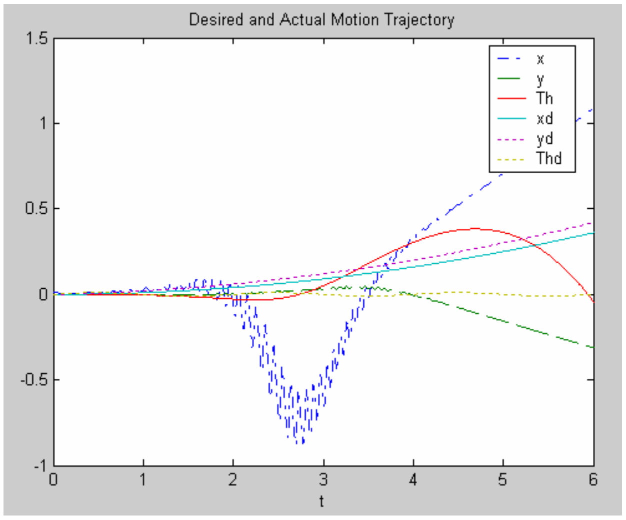


Figure 6.91: Adaptive controller desired versus actual motion trajectory.

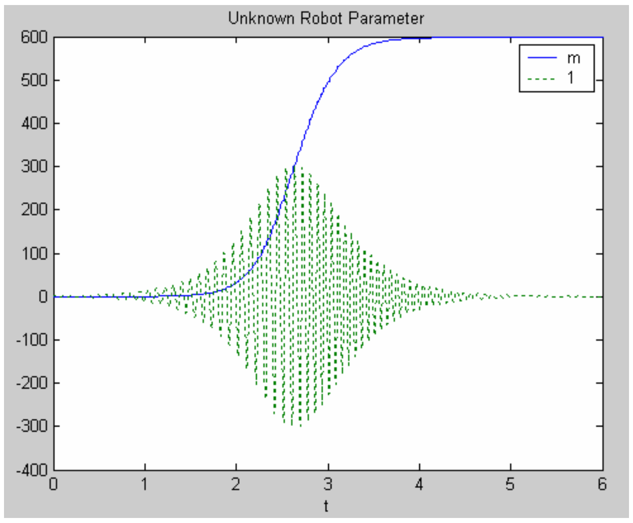


Figure 6.92: Parameters estimate.

- The last experiment shows that higher values of Γ can be used without impairing the performance of the controller. However, to study how much higher Γ can be raised, four experiments are performed with the same values for K_v (diagonal 2) and Λ (diagonal 5), and with a higher value for Γ. For Γ=diagonal 30, a similar performance was detected, as shown in Figs. 6.93-6.94. Further increasing the value of Γ to diagonal 50 does not affect the results, as shown in Fig. 6.95 and Fig. 6.96. Increasing the value of Γ to diagonal 100 reduces the tracking errors, as shown in Fig. 6.97 and Fig. 6.98 below. Finally, increasing the value of Γ to diagonal 1000 further reduces the tracking errors, as shown in Figs. 6.99-6.101.

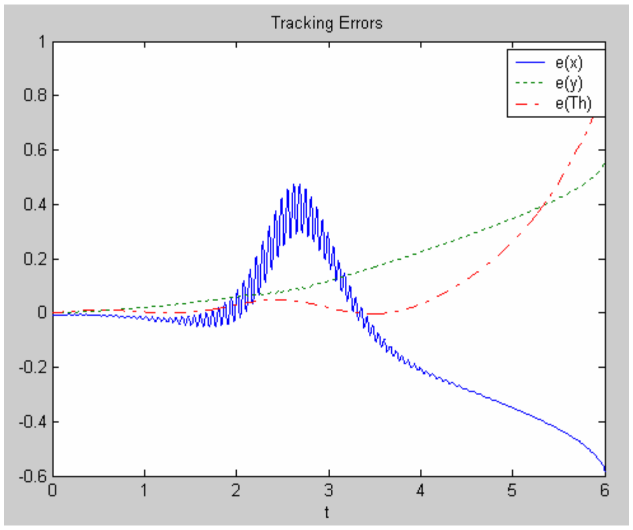


Figure 6.93: Adaptive controller tracking errors.

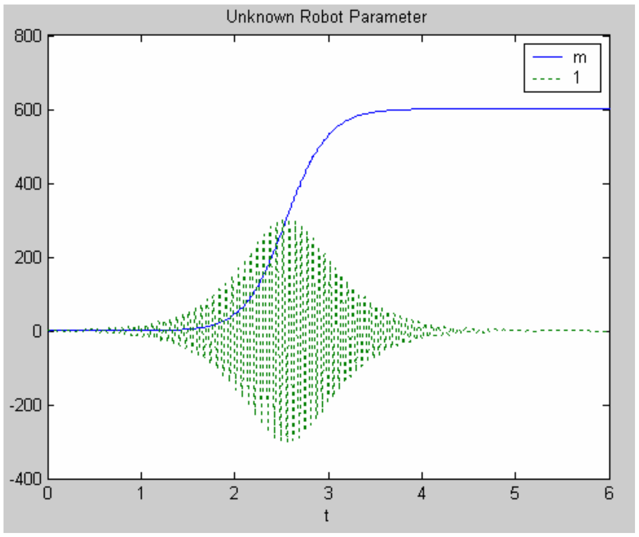


Figure 6.94: Adaptive controller parameters estimate.

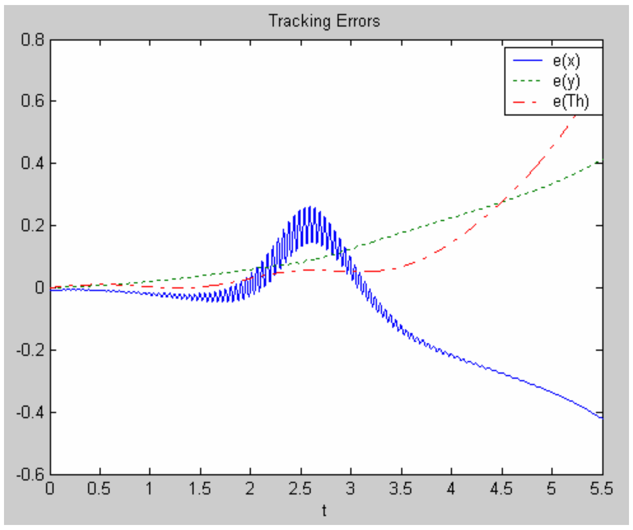


Figure 6.95: Adaptive controller tracking errors.

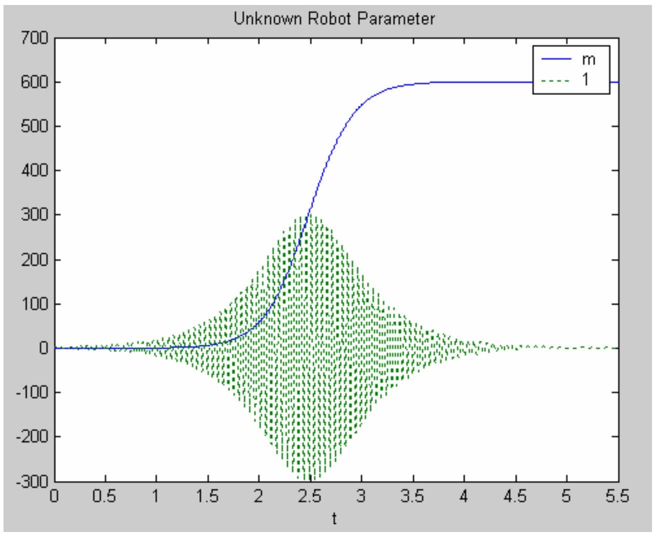


Figure 6.96: Adaptive controller parameters estimate.

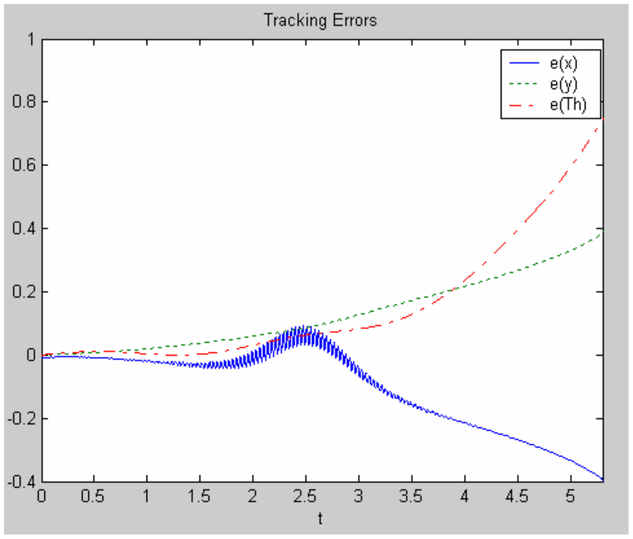


Figure 6.97: Adaptive controller tracking errors.

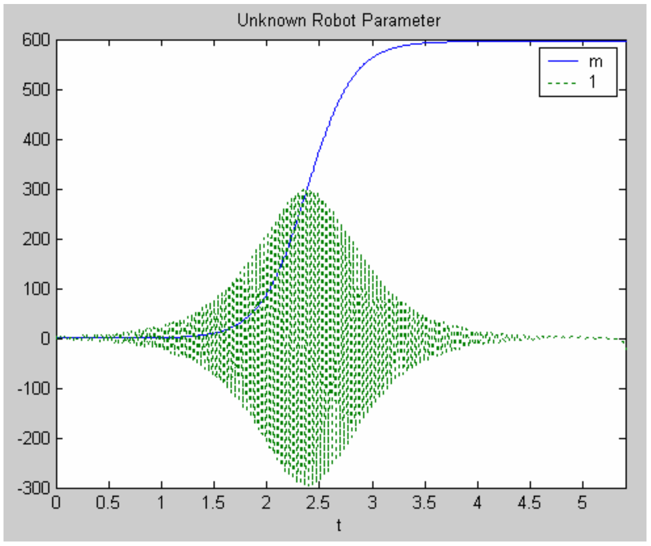


Figure 6.98: Adaptive controller parameters estimate.

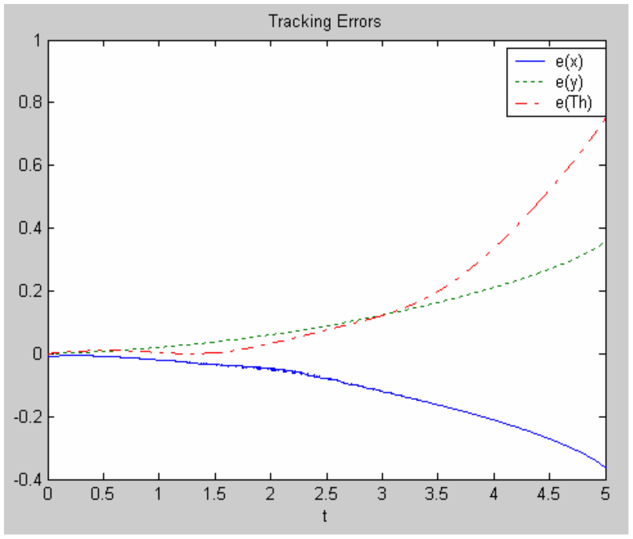


Figure 6.99: Adaptive controller tracking errors.

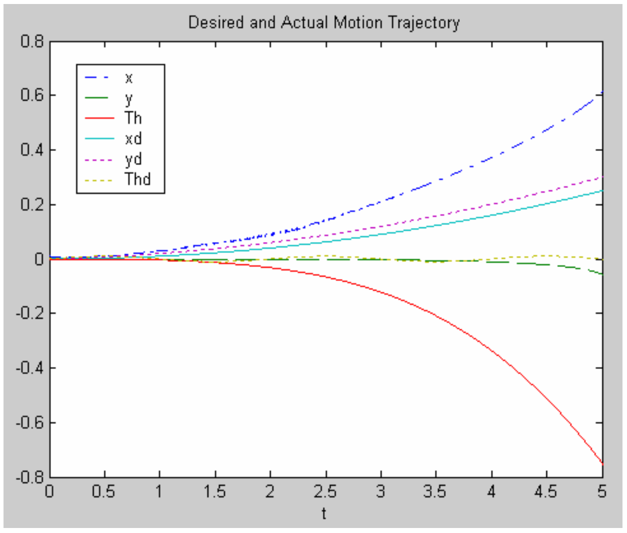


Figure 6.100: Adaptive controller desired versus actual motion trajectories.

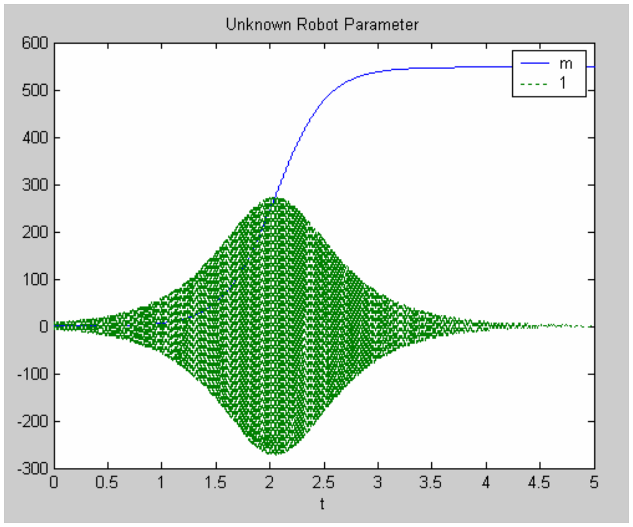


Figure 6.101: Adaptive controller parameters estimate.

6.4.5.2.2 Conclusion

Based on the results of the experiments conducted on the approximation-based adaptive controller for WMR navigation, the following controller parameters are recommended to provide the best performance:

Controller parameters	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

Table 6.18: Recommended approximation-based adaptive controller parameters for WMR navigation.

Increasing the value of K_v does not have much effect on the performance of the controller.

Regarding the values of $\lambda_1, \lambda_2, \lambda_3$, the experiments show that reducing the values of $\lambda_1, \lambda_2, \lambda_3$ makes the controller unable to approximate the unknown robot parameters. Thus, using values for $\lambda_1, \lambda_2, \lambda_3$ lower than 5 is not recommended.

Experiments show that decreasing the value of Γ to less than 10 makes the controller unable to approximate the unknown robot parameters and does not improve the performance of the controller. However, increasing the value of Γ improves the performance of the controller. It seems that there is no limit to how high the value of Γ can be.

Comparing the performance of the approximation-based adaptive controller with that of the CT controllers and the digital controller shows that both the CT controllers and the digital controller provide better results. This can be justified by the fact that we have a very good dynamic model for the WMR. Hence, approximation is not needed, and controllers using the exact model will provide better results. In other cases where the dynamic model cannot be derived, or only an approximate dynamic model is available, the approximation-based adaptive controller will produce better results.

Chapter 7: Conclusions and recommendations for future work

7.1 Conclusions

Intelligent machines that can perform tedious, routine, or dangerous tasks offer great promise to augment human activities. Robots have been used in manufacturing and service industries. They prove to improve productivity, quality, and flexibility. Robots are usually mounted on a fixed plate or on rails and can move in a limited manner. The success of robots in these environments encourage the use of mobile robots in other applications where the environments are not structured, such as in outdoor environments.

Navigation in an unstructured outdoor environment encounters such problems as obstacle avoidance, the ability to move to a desired location, and the ability to search an entire region. Various outdoor navigation scenarios pose problems extremely difficult to solve by an autonomous vehicle.

This dissertation presents the development of an autonomous navigation and obstacle avoidance system for WMR operation in unstructured outdoor environments.

It is suggested that the navigation system use a vision system consisting of digital CCD cameras and a video tracking device. Where there are no landmarks, a DGPS receiver is recommended. Use of a laser scanner is suggested to detect obstacles. The navigation algorithm was developed by using a feedforward neural network consisting of five layers. The network used a quasi-Newton backpropagation algorithm for training. Software was developed to simulate the performance and efficiency of the algorithm. The network was trained on a road with few obstacles and on a road with many obstacles. The network was able to produce output within a MSE of $1.33 * 10^{-4}$ and $4.86 * 10^{-4}$ from the target that was developed by an experienced driver. The network was tested on different roads with various obstacles and produced very good results on roads with a small number of obstacles. For a road with sharp curves and many obstacles, the network had some difficulties in dealing with obstacles on some parts of the road. However, this is very acceptable when compared with an experienced human driver.

PD CT, PID CT, digital, and adaptive controllers were developed to select suitable control torques for the motors causing the robot to follow the desired path, which was constructed by the navigation algorithm.

Dynamic simulation software was developed to test the performance and efficiency of the controllers. The simulation software was conducted from a framework constructed by Lewis, et al. [80] for two case studies. The first case study is the two-link robot manipulator, and the second is a navigation controller for the WMR. The WMR navigation controller takes the desired robot path from the navigation algorithm and produces the suitable control torques. The Lewis framework was formulated for trajectories of robot manipulators. This framework is adjusted as needed to control the robot path, rather than the manipulators' trajectories.

To conduct the dynamic simulation, a dynamic model was needed. A general kinematic and dynamic model for all WMRs was analyzed and presented. The general posture dynamic model for all WMRs was developed by using the Lagrange formulation.

For the Bearcat III robot, a more simplified kinematic and dynamic model was derived by considering only the velocity along the x and y axis and the angular velocity, with the robot center of mass as a reference point. Bearcat III robot dynamic model was derived by using the Newton-Euler method.

Several experiments were conducted on the PD CT, PID CT, digital and approximation-based adaptive controllers simulation software. The robot parameters were from Bearcat III. Various controller parameters were tested by different desired motion trajectories.

Experiments conducted on a PD CT controller for WMR navigation show that k_p and k_v must be different for each component of motion trajectory q. Small positive values are needed to obtain good results. The recommended controller parameters for WMR navigation is shown in Table 7.1.

The selection process of proper parameters for the PD CT controller is a challenging task, since six parameters must be changed in each trial. Although x, y, and θ are independent of each other, they are, nevertheless, used to update the new values of the actual motion trajectories. Thus, they have an impact upon each other. Therefore, keeping the same value of k_p and k_v for a particular motion trajectory component does not imply that the actual motion trajectory will remain the same.

Experiments conducted on a PID CT controller for WMR navigation show that the values of k_p, k_v, and k_i need to be small positive numbers to obtain good results. It is also noteworthy that increasing k_p and fixing k_v and k_i, or increasing k_v and fixing k_p and k_i, reduces the tracking errors for θ and x, while it increases the tracking error of y. However, there is a limit to this increase, which is about 10.

Using very large, or zero, values for k_p, k_v, or k_i is not recommended. As a condition for having a stable tracking error, the value of k_i must not be too large. The recommended PID CT controller parameters for WMR navigation is shown in Table 7.1.

Experiments conducted on a digital controller for WMR navigation show that k_p=2 and k_v=1 are the best controller parameters. Increasing the value of k_v does not improve results. However, increasing the value of k_p does, provided its value does not exceed 50.

Comparing the performance of the three controllers discussed so far, namely, PD CT, PID CT, and the digital controller, it can be shown that the digital controller provides the best results. Thus, its use is recommended for this application.

Based on the results of the experiments conducted on the approximation-based adaptive controller for WMR navigation, it is found out that increasing the value of K_v does not have much effect on the performance of the controller.

Regarding the values of $\lambda_1, \lambda_2, \lambda_3$, experiments show that reducing the values of $\lambda_1, \lambda_2, \lambda_3$ makes the controller unable to approximate the unknown robot parameters. Thus, using values for $\lambda_1, \lambda_2, \lambda_3$ lower than 5 is not recommended.

Experiments on the adaptive controller show that decreasing the value of Γ to less than 10 makes the controller unable to approximate the unknown robot parameters and does not improve the performance of the controller. However, increasing the value of Γ does improve the performance of the controller. It seems that there is no limit to how high the value of Γ can be.

Based on the results of the experiments conducted on the PD CT, PID CT, digital, and adaptive controllers for WMR navigation, the following controllers parameters are recommended to provide the best performance:

PD CT controller	
k_{p1}	2
k_{v1}	1
k_{p2}	0
k_{v2}	10
k_{p3}	2

k_{v3}	1
PID CT controller	
k_p	2
k_v	1
k_i	1
Digital controller	
k_p	2
k_v	1
Approximation-based adaptive controller	
K_v	$\begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix}$
Λ	$\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 5 \end{bmatrix}$
Γ	$\begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$

Table 7.1: Recommended controllers parameters for WMR navigation.

Comparing the performance of the approximation-based adaptive controller with that of the CT controllers and the digital controller shows that both the CT controllers and the digital controller provide better results. This can be justified by the fact that we have a very good dynamic model for the WMR. Hence, approximation is not needed, and controllers using the exact model will provide better results. In other cases where the dynamic model cannot be derived, or only an approximate dynamic model is available, the approximation-based adaptive controller will produce better results.

7.2 Recommendations for future work

Regardless of all research that has been conducted, autonomous navigation in unstructured environments is still an open area of research. There are many problems that need to be solved in this area.

With regard to the navigation algorithm, the performance of the network presented in this dissertation can be further modified by incorporating rules concerning how to deal with obstacles. This can be done by incorporating fuzzy logic into the neural network. Neuro-fuzzy systems combine the advantages of fuzzy systems and neural networks. Fuzziness can be introduced to the neural nets by using either a fuzzy input or a fuzzy output, or by using fuzzy training data. Neural network learning provides a good method for adjusting the expert's knowledge and automatically generates additional fuzzy rules and membership functions to meet certain specifications. It also reduces design time and costs.

The navigation algorithm deals only with fixed obstacles. It can be enhanced by incorporating rules that enable the algorithm to avoid moving obstacles. This will make the algorithm more effective in dealing with unpredictable real life situations. Moving obstacles can be incorporated by adding a fourth dimension to the problem, which is time. The robot must monitor the position of the obstacle in each time interval and then predict the next position of the obstacle, based upon the trajectory of the obstacle. The robot must then avoid that position.

The simulation developed for the navigation algorithm is two dimensional. Adding a third dimension will improve the software appearance, making it appear more like a real life scenario.

These simulations help in choosing the best network parameters and in testing the effectiveness and efficiency of the algorithm. The algorithm now needs to be implemented on Bearcat III. This can be done by modifying the simulation software, so that inputs are taken from the robot vision system, DGPS receiver, and the laser scanner, and output is sent to the main controller.

Concerning the CT and digital controllers, their performance can be enhanced by finding a good method for calculating the best k_p, k_v, and k_i values.

Filtered-error approximation-based controllers can be used to develop adaptive, robust, and learning controllers. Only the adaptive controller is covered in this dissertation. Hence, robust and learning controllers can also be developed from the same principles.

Once again, it would be beneficial if these controllers were built into Bearcat III. This can be done by using navigation system outputs as inputs to the controller and connecting controller outputs to Bearcat III motors.

References

1. W. Golnazarian and E. L. Hall, “Intelligent industrial robots,” in Handbook of Industrial Automation, New York: Marcel Dekker, Inc., 2000, pp. 499-527.
2. E.L. Hall and B.C. Hall, *Robotics: A User-Friendly Introduction*. Orlando, FL: Saunders College Publishing, Holt, Rinehart and Winston, 1985, pp. 66-75.
3. A. M. Meystel and J. S. Albus, Intelligent Systems: Architecture, Design, and Control, New York: John Wiley & Sons, Inc., 2002, pp. 3-7.
4. P. J. McKerrow, *Introduction to Robotics.* Reading, MA: Addison-Wesley, 1991, pp. 18-20.
5. M. Weil, “New competitiveness spurs record robot sales,” *Managing Automation*, Vol. 9(6), part 2 (of 2), pp. 5-8, June 1994.
6. C. R. Asfahl, *Robots and Manufacturing Automation.* 2nd ed., New York: John Wiley & Sons, 1992, pp. 137-177.
7. R. D. Klafter, T. A. Chmielewski, and M. Negin, *Robotic Engineering An Integrated Approach*. Englewood Cliffs New Jersey, NY: Prentice Hall, 1989, pp. 84-89.
8. T. A. Lasky and T. C. Hsia, “Robotics: Robot configuration,” in The Electrical Engineering Handbook. Boca Raton Florida: R.C., CRC Press, Inc., 1993, pp. 2154-2162.
9. A. Kosaka and J. Pan, “Purdue experiments in model-based vision for hallway navigation,” in *Proc. of Workshop on Vision for Robots in IROS’95 conference,* 1995, pp. 87-96.
10. D. H. Hubel and T. N. Wiesel, “Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex,” *J. Physiol.*, pp. 106-154, Jan. 1962.
11. Y. Aloimonos, *Visual Navigation, From Biological Systems to Unmanned Ground Vehicles.* New Jersey, NY: Lawrence Erlbaum Associates Publishers, 1997, pp. 6-17.

12. P. Andreou and A. Charalambides. (1996, May). Exploring mars using intelligent robots. *Surprise* [Online]. *4*, pp. 3-9. Available: http://www-dse.doc.ic.ac.uk/~nd/surprise_95/journal/vol4/pma/report.html.

13. R. Araujo, and A. T. de Almeida, "Learning sensor-based navigation of a real mobile robot in unknown worlds," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 29, No. 2, pp. 164-178, April 1999.

14. A. De la Escalera, L. Moreno, M. A. Salichs, and J. M. Armingol, "Continuous mobile robot localization by using structured light and a geometric map," *International Journal of Systems Science*, Vol. 27, No. 8, pp.771-782, Aug. 1996.

15. J. Borenstein, H. R. Everett, L. Feng, and D. Wehe, "Mobile robot positioning: Sensors and techniques," *Journal of Robotic Systems*, Vol. 14, Issue 4, pp.231-249, April 1997.

16. J. A. Janet, R. C. Luo, and M. G. Kay, "Autonomous mobile robot global motion planning and geometric beacon collection using traversability vectors," *IEEE Transactions on Robotics and Automation*, Vol. 13, No. 1, pp. 132-140, Feb. 1997.

17. S. Premvuti and J. Wang, "Relative position localizing system for multiple autonomous mobile robots in distributed robotic system: System design and simulation," *Robotics and Autonomous Systems*, Vol. 18, Issue 3, pp. 319-326, August 1996.

18. M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *IEEE Transactions on Robotics and Automation,* Vol. 13, No. 2, pp. 251-263, April 1997.

19. A. Zelinsky and Y. Kuniyoshi, "Learning to coordinate behaviors for robot navigation," *Advanced Robotics*, Vol. 10, No. 2, pp. 143-159, 1996.

20. S. Kotani, K. Nishikawa, and H. Mori, "Empirical learning in mobile robot navigation," *Advanced Robotics*, Vol. 12, No. 4, pp. 317-333, 1998.

21. Trimble Navigation Limited. (2002). All about GPS. [Online]. Available: http://www.trimble.com/gps/

22. P. H. Dana, (1994, Sep.). The geographer's craft project, Department of Geography. The University of Colorado at Boulder. [Online]. Available: http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html

23. G. von Wichert, "Self organizing visual perception for mobile robot navigation," in *Proceedings of the First Euromicro Workshop on Advanced Mobile Robot*, 1996 pp. 194-200.

24. F. Michaud and M. J. Mataric, "Learning from history for adaptive mobile robot control," *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, 1998, pp. 1865-1870.

25. T. Minato and M. Asada, "Environmental change adaptation for mobile robot navigation," *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, 1998, pp. 1859-1864.

26. J. Lee, S. Choi, C. Lee, Y. Lee, and K. Lee, "A study for AGV steering control and identification using vision system," *Proceedings of IEEE International Symposium on Industrial Electronics*, Vol. 3, 2001, pp. 1575-1578.

27. K. Tomita and S. Tsugawa, "Visual navigation of a vehicle along roads: The algorithm and experiments," *Proceedings of Vehicle Navigation and Information Systems Conference*, 1994, pp. 419-424.

28. H. Frohn and W. Seelen, "VISOCAR: An autonomous industrial transport vehicle guided by visual navigation," *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, 1989, pp. 1155-1159.

29. K. Storjohann, T. Zielke, H. A. Mallot, and W. Seelen, “Visual Obstacle Detection for Automatically Guided Vehicles,” *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 2, 1990, pp. 761-766.

30. M. R. Kabuka, S. Harjadi, and A. Younis, “A fault-tolerant architecture for an automatic vision-guided vehicle,” *IEEE Transactions on Systems, Man and Cybernetics,* Vol. 20, Issue 2, pp. 380-394, March/April 1990.

31. K. Y. Fok and M. R. Kabuka, “An automatic navigation system for vision guided vehicles using a double heuristic and a finite state machine,” *IEEE Transactions on Robotics and Automation*, Vol. 7, Issue 1, pp. 181-189, Feb. 1991.

32. G. Beccari, S. Caselli, F. Zanichell, and A. Calafiore, “Vision-based line tracking and navigation in structured environments,” *Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation*, 1997, pp. 406-411.

33. J. Kurata, K. T. V. Grattan, and H. Uchiyama, “Navigation system for a mobile robot with a visual sensor using a fish-eye lens,” *American Institute of Physics*, Vol. 69, Issue 2, pp. 585-590, Feb. 1998.

34. E. M. Nebot and H. Durrant-Whyte, “A high integrity navigation architecture for outdoor autonomous vehicles,” *Robotics and Autonomous Systems*, Vol. 26, Issue 2-3, pp. 81-97, Feb. 1999.

35. W. Wijesoma, P. P. Khaw, and E. K. Teoh, “Control and navigation of an outdoor AGV using fuzzy reasoning,” *Proceedings of IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, 1999, pp. 544-549.

36. H. Roth and K. Schilling, “Control strategies for mobile robots based on fuzzy logic,” *Proceedings of 1995 IEEE International Conference on Fuzzy Systems, International Joint*

Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium, Vol. 1, 1995, pp. 89-96.

37. S. K. Tso, Y. H. Fung, and Y. P. Cheung, "Fuzzy-logic control for differential-wheel-drive AGVs using linear opto-sensor arrays," *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 3, 1996, pp. 2816-2821.
38. G. Castellano, G. Attolico, E. Stella, and A. Distante, "Automatic generation of rules for a fuzzy robotic controller," *Proceedings of the 1996 IEEE/RSJ International Conference on Intelligent Robots and Systems '96, IROS 96*, Vol. 3, 1996, pp. 1179-1186.
39. P. Baranyi, A. Martinovics, S. Kovacs, D. Tikk, and Y. Yam, "A general extension of fuzzy svd rule base reduction using arbitrary inference algorithm," *IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 3, 1998, pp. 2785-2790.
40. Y. H. Fung and S. K. Tso, *"*Analysis of linguistic fuzzy control for curved-path-following autonomous vehicles," *Proceedings of IEEE International Conference on Robotics and Automation*, Vol. 3, 1998, pp. 2506-2511.
41. E. T. Lee, "Applying fuzzy logic to robot navigation," *Kybernetes*, Vol. 24, No. 6, pp. 38-43, July 1995.
42. T. E. Mora and E. N. Sanchez, "Fuzzy logic-based real-time navigation controller for a mobile robot," *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 1, 1998, pp. 612-617.
43. C. H. Lin and L. L. Wang, "Intelligent collision avoidance by fuzzy logic control", *Robotics and Autonomous Systems*, Vol. 20, pp. 61-83, April 1997.

44. K. H. C. Kim and S. Hyung, , “Mobile robot navigation based on optimal via-point selection method,” *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, 1998, pp. 1242-1247.

45. K. L. Watanabe, K. Izumi, J. Tang, and F. Han, “Rotational control of an omnidirectional mobile robot using a fuzzy servo controller,” *Advanced Robotics*, Vol. 12, Issue 3, pp. 171-189, 1998.

46. J. W. Choi, S. H. Kwon, H. Y. Lee, and S.G. Lee, “Navigation strategy of an intelligent mobile robot using fuzzy logic,” *Proceedings of the 1998 IEEE International Conference on Fuzzy Systems*, Vol. 1, 1998, pp. 602-605.

47. S. Kurd and K. Oguchi, *‘*Neural network control for automatic guided vehicles using discrete reference markers,” *Industry Applications Society, Thirty-Second IAS Annual Meeting,* Vol. 2, 1997, pp. 886-891.

48. G. Vercelli and P. Morasso, “Recognition and classification of path features with self-organizing maps during reactive navigation,” *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, 1998, pp. 1437-1442.

49. H. Xue and E. Cheung, “learning control for position tracking of active suspension of high-speed AGV via neural network,” *Proceedings of 1996 IEEE Conference on Emerging Technologies and Factory Automation*, Vol. 2, 1996, pp. 523-527.

50. D. C. Dracopoulos, “Robot path planning for maze navigation,” *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol. 3, 1998, pp. 2081-2085.

51. Z. Y. Zhu, S. Yang, D. Shi, and G. Xu, , “Better road following by integrating omni-view images and neural nets,” *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol. 2, 1998, pp. 974-979.

52. G. Cicirelli, D. Distante, T. D. Orazio, and G. Attolico, “Learning actions from vision-based positioning in goal-directed navigation,” *Proceedings of the 1998 IEEE.RSJ International Conference on Intelligent Robots and Systems*, Vo. 3, 1998, pp. 1715-1720.

53. E. Kruse and F.M. Wahl, “Camera-based monitoring system for mobile robot guidance,” *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems,* Vol. 2, 1998, pp. 1248-1253.

54. J. J. Little, J. Lu, and D. R. Murray, “Selecting stable image features for robot localization using stereo,” *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, 1998, pp. 1072-1077.

55. A. R. Arsenio and M. Isabel, “Active range sensing for mobile robot localization,” *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems,* Vol. 2, 1998, pp. 1066-1071.

56. G. Z. Grudic and P. D. Lawrence, “Nonparametric learning approach to vision based mobile robot localization,” *Proceedings of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, 1998, pp. 724-729.

57. H. J. S. Feder, J. J. Leonard, and C. M. Smith, “Adaptive concurrent mapping and localization using sonar,” *Proceeding of the 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, 1998, pp. 892-898.

58. N. S. Vlassis, G. Papakonstantinou, and P. Tsanakas, “Dynamic sensory probabilistic maps for mobile robot localization,” *Proceedings of the 1998 IEEE.RSJ International Conference on Intelligent Robots and Systems*, Vol. 2, 1998, pp. 718-723.

59. K. C. Ng and M. M. Trivedi, “Neuro-fuzzy controller for mobile robot navigation and multirobot convoying,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 28, No. 6, pp. 829-840, Dec. 1998.

60. W. A. Daxwanger and G. Schmidt, “Neuro-fuzzy posture estimation for visual vehicle guidance,” *Proceedings of the 1998 IEEE International Joint Conference on Neural Networks*, Vol. 3, 1998, pp. 2086-2091.

61. M. Piggio and R. Zaccaria, “Using roadmaps to classify regions of space for autonomous robot navigation,” *Robotics and Autonomous Systems*, Vol. 25, pp. 209-217, Nov. 1999.

62. P. Veelaert and H. Peremans, “Flexibility maps: A formalization of navigation behaviors,” *Robotics and Autonomous Systems*, Vol. 27, pp. 151-169, May 1999.

63. M. O. Franz and H. A. Mallot, “Biomimetic robot navigation,” *Robotics and Autonomous Systems*, Vol. 30, Issue 1-2, pp. 133-153, Jan. 2000.

64. T.M.C. Smith, P. Husbands, A. Philippides, and M. O'Shea, “Evaluating the effectiveness of biologically-inspired robot control networks through operational analysis,” in *EPSRC/BBSRC International Workshop on Biologically-Inspired Robotics: The Legacy of W. Grey Walter*, 2002, pages 280-287.

65. M. Quinn, L. Smith, G. Mayley, and P. Husbands, “Evolving teamwork and role allocation for real robots,” in *Proceedings of 8th International Conference on Artificial Life*, 2002, pages 302-311.

66. L. Smith and P. Husbands, “Visual landmark navigation through large-scale environments,” in *EPSRC/BBSRC International Workshop on Biologically-Inspired Robotics: The Legacy of W. Grey Walter*, 2002, pages 272-279.

67. H. Hiraishi, H. Ohwada, and F. Mizoguchi, “Web-based communication and control for multiagent robots,” *Proceedings of the 1998 IEEE.RSJ International Conference on Intelligent Robots and Systems*, Vo. 1, 1998, pp. 120-125.

68. E. Martinez and C. Torras, “Qualitative vision for the guidance of legged robots in unstructured environments,” *Pattern Recognition*, Vol. 34, Issue 8, pp. 1585-1599, August 2001.

69. R. Kurazume and S. Hirose, “Development of a cleaning robot system with cooperative positioning system”, *Autonomous Robots,* Vol. 9, Issue 3, pp. 237-246, Dec. 2000.

70. C. Torras, “Robot adaptivity,” *Robotics and Autonomous systems*, Vol. 15, Issue 1-2, pp. 11-23, July 1995.

71. A. Yahja, S. Sanjiv, and A. Stentz, “An efficient on-line path planner for outdoor mobile robots,” *Robotics and Autonomous systems*, Vol. 32, Issue 2-3, pp. 129-143, Aug. 2000.

72. G. Baratoff, C. Toepfer, and H. Neumann, “Combined space-variant maps for optical-flow-based navigation,” *Biological Cybernetics*, Vol. 83, Issue 3, pp. 199-209, Aug. 2000.

73. H. Yu, C. Chi, T. Su, and Q. Bi, “Hybrid evolutionary motion planning using follow boundary repair for mobile robots,” *Journal of Systems Architecture*, Vol. 47, Issue 7, pp. 635-647, July 2001.

74. X. Yan, S. S. lyengar, and N. E. Brener, “An event driven integration reasoning scheme for handling dynamic threats in an unstructured environment,” *Artificial Intelligence*, Vol. 95, Issue 1, pp. 169-186, August 1997.

75. K. H. Chen and W. H. Tsai, “Vision-based obstacle detection and avoidance for autonomous land vehicle navigation in outdoor roads,” *Automation in Construction*, Vol. 10, Issue 1, pp. 1-25, Nov. 2000.

76. R. Jarvis, "An all-terrain intelligent autonomous vehicle with sensor-fusion-based navigation capabilities," *Control Eng. Practice*, Vol. 4, No. 4, pp. 481-486, April 1996.

77. M. Devy, R. Chatila, P. Fillatreau, S. Lacroix, and F. Nashashibi, "On autonomous navigation in a natural environment," *Robotics and Autonomous Systems*, Vol. 16, pp. 5-16, Nov. 1995.

78. K. M. Krishna and P. M. Kalra, "Perception and remembrance of the environment during real-time navigation of a mobile robot," *Robotics and Autonomous Systems*, Vol. 37, pp. 25-51, Oct. 2001.

79. D. B. Marco, A. J. Healey, and R. B. McGhee, "Autonomous underwater vehicles: Hybrid control of mission and motion," *Autonomous Robots*, Vol. 3, Issue 2, pp. 169-186, May 1996.

80. F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Padstow, UK: Taylor and Francis Ltd, T. J. International Ltd, 1999, pp. 147-167.

81. H.-S. Shim and Y.-G. Sung, "Asymptotic control for wheeled mobile robots with driftless constraints," *Robotics and Autonomous Systems*, Vol. 43, Issue 1, pp. 29-37, April 2003.

82. H. Yun-Su and S. Yuta, "Trajectory tracking control for navigation of the inverse pendulum type self-contained mobile robot," *Robotics and Autonomous Systems*, Vol. 17, Issue 1-2, pp. 65-80, April 1996.

83. R. Rajagopalan and N. Barakat, "Velocity control of wheeled mobile robots using computed torque control and its performance for a differentially driven robot," *Journal of Robotic Systems*, Vol. 14, Issue 4, pp. 325-340, April 1997.

84. M. Zhang and R. M. Hirschorn, "Discontinuous feedback stabilization of nonholonomic wheeled mobile robots," *Dynamics and Control*, Vol. 7, Issue 2, pp. 155-169, April 1997.

85. K. C. Koh and H. S. Cho, “A smooth path tracking algorithm for wheeled mobile robots with dynamic constraints,” *Journal of Intelligent and Robotic Systems*, Vol. 24, Issue 4, pp. 367-385, April 1999.

86. Y. Zhang, J. H. Chung, and S. A. Velinsky, “Variable structure control of a differentially steered wheeled mobile robot,” *Journal of Intelligent and Robotic Systems*, Vol. 36, Issue 3, pp. 301-314, March 2003.

87. M. L. Corradini and G. Orlando, “Robust tracking control of mobile robots in the presence of uncertainties in the dynamical model,” *Journal of Robotic Systems*, Vol. 18, Issue 6, pp. 317-323, June 2001.

88. M. L. Corradini and G. Orlando, “Control of mobile robots with uncertainties in the dynamical model: a discrete time sliding mode approach with experimental results,” *Control Engineering Practice*, Vol. 10, Issue 1, pp. 23-34, January 2002.

89. M. L. Corradini, T. Leo, and G. Orlando, “Experimental testing of a discrete-time sliding mode controller for trajectory tracking of a wheeled mobile robot in the presence of skidding effects,” *Journal of Robotic Systems*, Vol. 19, Issue 4, pp. 177-188, April 2002.

90. S.-F. Wu, J.-S. Mei, and P.-Y. Niu, “Path guidance and control of a guided wheeled mobile robot,” *Control Engineering Practice*, Vol. 9, Issue 1, pp. 97-105, January 2001.

91. Z.-P. Jiang, E. Lefeber, and H. Nijmeijer, “Saturated stabilization and tracking of a nonholonomic mobile robot,” *Systems and Control Letters*, Vol. 42, Issue 5, pp. 327-332, April 2001.

92. W. Dong, W. Liang Xu, and W. Huo, “Trajectory tracking control of dynamic non-holonomic systems with unknown dynamics,” *International Journal of Robust and Nonlinear Control*, Vol. 9, Issue 13, pp. 905-922, November 1999.

93. P. J. Werbos, “New designs for universal stability in classical adaptive control and reinforcement learning,” *Proceedings of the International Joint Conference on Neural Networks IJCNN, USA,* Vol. 4, 1999, pp. 2292-2295.

94. M. B. Montaner and A. Ramirez-Serrano, “Fuzzy knowledge-based controller design for autonomous robot navigation,” *Expert Systems with Applications*, Vol. 14, Issue 1-2, pp. 179-186, January 1998.

95. A. V. Topalov, J.-H. Kim, and T. P. Proychev, “Fuzzy-net control of non-holonomic mobile robot using evolutionary feedback-error-learning,” *Robotics and Autonomous Systems*, Vol. 23, Issue 3, pp. 187-200, April 1998.

96. M. Toda, O. Kitani, T. Okamoto, and T. Torii, “Navigation method for a mobile robot via sonar-based crop row mapping and fuzzy logic control,” *Journal of Agricultural Engineering Research*, Vol. 72, Issue 4, pp. 299-309, April 1999.

97. F. Vázquez and E. Garcia, “A local guidance method for low-cost mobile robots using fuzzy logic,” *Annual Review in Automatic Programming*, Vol. 19, pp. 203-207, 1994.

98. S. Chen, T. Hsieh, J. Kung, and V. Beffa. (2000, June). Autonomous Weapons. Stanford Computer Science Education, Stanford University, Stanford, CA. [Online]. Available: http://cse.stanford.edu/classes/cs201/projects-95-96/autonomous-weapons/

99. J. Hollingum, “Robots for the dangerous tasks,” *Industrial Robot*, Vol. 26, No. 3, pp. 178-183, May 1999.

100. Space and Naval Warfare Systems Center, San Diego (SSC San Diego). (2002, Sep.). Robotics at space and naval warfare systems center. The Space and Naval Warfare Systems Center, San Diego. [Online]. Available: http://www.spawar.navy.mil/robots

101. J. Hollingum, “Unmanned vehicles go to war,” *Industrial Robot*, Vol. 25, Issue 6, pp. 379-383, Nov. 1998.

102. L. Greenhouse and J. Norris. (2002, Nov.) Control system architecture for military robotics. The NIST Virtual Timeline, Information Services Division, National Institute of Standards and Technology. [Online]. Available: http://museum.nist.gov/exhibits/timeline/printerFriendly.cfm?itemId=38

103. D. Golding. (2002, July). Intruder military robot. Angelus Research Corporation. Garden Grove, California. [Online]. Available: http://www.angelusresearch.com/intruder.htm

104. R. T. Laird, M. H. Bruch, M. B. West, D. A. Ciccimaro, and H. R. Everett. (2000, April). Issues in vehicle teleoperation for tunnel and sewer reconnaissance. Presented at IEEE International Conference on Robotics and Automation. [Online]. Available: http://vrai-group.epfl.ch/icra2000/papers/laird.pdf

105. J. Pransky, “Mobile robots: Big benefits for us military,” *Industrial Robot*, Vol. 24, Number 2, pp. 126-130, May 1997.

106. M. L. Perez. (1997, Oct.). A Low-cost multi-terrain mobile robot for hostile environments. Lawrence Livermore National Laboratory. [Online]. Available: http://www.llnl.gov/automation-robotics/1_star.html

107. J. Hollingum, “Military look to flying insect robots,” *Industrial Robot*, Vol. 25, Issue 2, pp. 124-128, March 1998.

108. M. Rosenblum and B. Gothard, “A High Fidelity Multi-sensor Scene Understanding System for Autonomous Navigation,” *Proceedings of the IEEE Intelligent Vehicles Symposium 2000, Dearborn (MI), USA,* 2000, pp. 637-643.

109. S. P. N. Singh and S. M. Thayer. (2001, July). ARMS (Autonomous Robots for Military Systems): A Survey of Collaborative Robotics Core Technologies and Their Military Applications. Tech. report CMU-RI-TR-01-16, Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania. [Online]. Available: http://www.ri.cmu.edu/pub_files/pub3/singh_surya_2001_2/singh_surya_2001_2.pdf

110. E. H. Durfee, P. G. Kenny, and K. C. Kluge, "Integrated permission planning and execution for unmanned ground vehicles," *Autonomous Robots*, Vol. 5, Issue 1, pp. 97-110, March 1998.

111. J. G. Blitch, "Artificial intelligence technologies for robot assisted urban search and rescue,"*Expert Systems with Applications* , Vol. 11, Issue 2, pp. 109-124, 1996.

112. C. C. de Wit, B. Siciliano, and G. Bastin, *Theory of Robot Control*. Britain: Springer-Verlag London Limited, 1996, pp. 265-303.

113. G. Lazea, E. Lupu, and M. Patko, "Aspects on kinematic modelling and simulation of wheeled robots," *International Symposium on Systems Theory, Robotics, Computers and Process Informatics, SINTES 8, Craiova*, 1996, pp. 150.

114. W. Wu, H. Chen, and P. Woo, "Time optimal path planning for a wheeled mobile robot," *Journal of Robotic Systems*, Vol. 17, Issue 11, pp.585-591, Oct. 2000.

115. E. W. Weisstein. (2003). Moore-Penrose Matrix Inverse. Eric Weisstein's World of Mathematics. [Online]. Available: http://mathworld.wolfram.com/Moore-PenroseMatrixInverse.html

116. G. H. Golub and C. F. Van Loan, *Matrix Computations*. 3rd ed., Baltimore, Maryland: The Johns Hopkins University Press, 1996, pp. 257-258.

117. K. Kameyama. (2002). Pseudoinverse matrix (Moore-Penrose generalized inverse), Class notes, Graduate School of Systems and Information Engineering, University of Tsukuba, Tokyo, Japan. [Online]. Available: http://risk.tara.tsukuba.ac.jp/~kame/sc2/sc2-020419-projection.pdf

118. C. R. Nave. (March, 2000). Moment of Inertia, Class notes, Goergia State University, Atlanta, Georgia. [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/mi.html#mig

119. F. P. Beer and E. R. Johnston, Jr., *Vector Mechanics for Engineers: Statics and Dynamics*. 6th ed., USA: McGraw Hill, 1997, pp. 496-501.

120. M. H. Hassoun, *Fundementals of Artificial Neural Netwoks*. USA: Massachusetts Institute of Technology Press, 1995, pp. 57-134.

121. R. Kothari, "Neural Network," University of Cincinnati, OH, course notes, Dec. 2000.

122. K. Sethi and A. K. Jain, *Artificial Neural Networks and statistical Pattern Recognition: Old and New Connections*. Amsterdam, Netherlands: Elsevier Science Publishers B. V., 1991, pp. 12-14.

123. T. Kohnen, *Self-Organization and Associative Memory*. 2th ed., Berlin Heidelberg, Germany: Springer-Verlag, 1988, pp. 119-122.

124. E. Micheli-Tzanakou, *Supervised and Unsupervised Pattern Recognition: Feature Extraction and Computational Intelligence*. USA: CRC Press LLC, 2000, pp. 14-18.

125. M. H. Hassoun, *Associative Neural Memories: Theory and Implementation*. New York, USA: Oxford University Press, Inc., 1993, pp. 3-9.

126. Matlab 6. (1994-2002). Neural network toolbox. The MathWorks, Inc. Natick, MA. [Online]. Available: http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/nnet.shtml?BB=1

127. M. Saxena, *Obstacle Avoidance using Laser Scanner for Bearcat III*. Cincinnati, OH: University of Cincinnati, Master thesis, 2001, pp. 22-23.

128. B. C. Kuo, *Digital Control Systems.* 2nd ed., Ft. Worth: Saunders College Publishing, 1992, pp. 1-3.

129. L. S. Shie, W. M. Wang, and M. K. Appu Panicker, "Design of PAM and PWM digital controllers for cascaded analog systems**,"** *ISA Transactions(R),* Vol. 37, Issue 3, pp. 201-213, July 1998.

Bibliography

1. A. M. S. Zalzala and X. Liu, *Neural Networks for Identification, Prediction and Control*. Great Britain: Springer- Verlag London Limited, 1995.
2. D. T. Pham and A. S. Morris, *Neural Networks for Robotic Control, Theory and Applications*. Great Britain: Ellis Horwood Limited, 1996.
3. G. Campion, G. Bastin, and B. D'Andrea-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE transactions on robotics and automation,* Vol. 12, No. 1, pp. 47-62, Feb. 1996.
4. L. Sciavicco and B. Siciliano, *Modeling and Control of Robot Manipulators*. New York: McGraw-Hill, 1996, pp. 119-167.
5. M. Chester, *Neural Network: A Tutorial*. Englewood Cliffs, New Jersey: PTR Prentice Hall, 1993, pp. 1-17.
6. O. Omidvar and D. L. Elliott, *Neural Systems for Control*. USA: Academic Press, 1997.
7. P. F. Muir and C. P. Neuman, "Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot," *IEEE international conference on robotics and automation*, Vol. 3, 1987, pp. 1172-1178.
8. P. F. Muir and C. P. Neuman, "Kinematic modeling of wheeled mobile robots," *Journal of robotic systems*, Vol. 4, No. 2, pp. 281-340, 1987.
9. P. J. Antsaklis and K. M. Passino, *An Introduction to Intelligent and Autonomous Control*. Boston, MA: Kluwer Academic Publishers, 1993.
10. P. J. Werbos and A. J. Pellionisz, "Neurocontrol and neurobiology: new developments and connections," *International Joint Conference on Neural Networks (IJCNN)*, Vol. 3, 1992, pp. 373-378.

11. P. J. Werbos, “Backpropagation and neurocontrol: a review and prospectus,” *International Joint Conference on Neural Networks (IJCNN)*, Vol. 1, 1989, pp. 209-216.
12. P. J. Werbos, “Neural networks for control and system identification,” *Proceedings of the 28th IEEE Conference on Decision and Control*, Vol. 1, 1989, pp. 260-265.
13. P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, Vol. 78, Issue 10, 1990, pp. 1550-1560.
14. P. J. Werbos, “An overview of neural networks for control,” *IEEE Control Systems Magazine*, Vol. 11, Issue 1, 1991, pp. 40-41.

Index