# FUZZY LOGIC CONTROL OF A TWO DEGREE OF FREEDOM PARALLEL ROBOT

by

Angelo Liadis

A thesis submitted to the faculty of graduate studies
Lakehead University
in partial fulfillment of the requirements for the degree of
Masters of Science in Control Engineering

Department of Electrical Engineering

Lakehead University

September 2010

# Lakehead

UNIVERSITY

OFFICE OF GRADUATE STUDIES

| | |
|---|---|
| NAME OF STUDENT: | Angelo Liadis |
| DEGREE AWARDED: | Masters of Science in Control Engineering |
| ACADEMIC UNIT: | Department of Electrical Engineering |
| TITLE OF THESIS: | **FUZZY LOGIC CONTROL OF A TWO DEGREE OF FREEDOM PARALLEL ROBOT** |

This thesis have been prepared
under my supervision
and the candidate has complied
with the Master's regulations.

Signature of Supervisor

Date

# Acknowledgments

I would like to thank my supervisor Dr. Xiaoping Liu for all his help in the development and simulation of the parallel robot; ensuring that the results I obtained were accurate and logical. I would also like to express gratitude to Andrew Cudowski for his assistance with the MATLAB coding along with the drafting of the mechanical structure of the parallel robot and Longjun Qian for his unique insight while working on the controller derivations. I would also like to thank Kailash Bhatia for his aid with the machinery in the mechanical workshop at Lakehead University and Haibin Wang for his help with the C coding. Finally, I would like to thank all my family and friends for all their support throughout these two years by always keeping me focused on my goals.

# Abstract

Parallel robots differ greatly from their serial counterparts in terms of performance and mechanical ability. Serial robots are open chained mechanisms since each link is connected to an adjacent one and each joint is actuated, while parallel robots implement a closed chained structure. This type of structure consists of having the endpoint of each kinematic chain connected to one another, hence called the common point. There can be a multitude of kinematic chains, but each chain utilizes only one actuator located near the base of the system to perform the desired operation. This brings up the dilemma between parallel and serial robots. Serial robots allow for a greater control precision and a simpler dynamic model, yet they are generally more expensive and their load capacity is limited due to their large mass. Parallel robots have a higher load capacity due to their smaller mass, much faster accelerations at the end effector and boast a high mechanical stiffness to weight ratio. Their drawbacks entail a complex dynamic model and they generally have many singular regions that must be avoided in order to achieve stability.

There has been substantial research aimed at improving the performance of parallel robots by implementing PID and adaptive controllers' and so on, but due to the variations in the dynamic models of each system, it is nearly impossible to conclusively determine the most appropriate controller to design. Therefore, this thesis compares the simulation and experimental results of four non-fuzzy logic controllers, namely the non-adaptive and adaptive PD and backstepping controllers along with four fuzzy logic controllers, namely the fuzzy PD, indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers on a planar two degrees of freedom parallel robot in order to determine which controller would yield the best control performance.

By comparing the simulation results for the joint angles error and the end effector trajectory error plots for the non-fuzzy and fuzzy logic controllers, the adaptive backstepping and the fuzzy adaptive backstepping controllers held the potential to be the most likely candidate controllers to implement on the physical structure of the two degrees of freedom parallel robot.

After the eight controllers elaborated in this thesis were utilized on the parallel robot structure, the controller that outputted the most impressive experimental results were found in the fuzzy adaptive backstepping controller. The joint angles error and end effector trajectory deviation yielded particularly low results, but it is the tracking performance which differentiates this controller from the rest. It has the smoothest tracking performance when compared between the non-fuzzy and fuzzy logic controllers discussed in this thesis without conceding a large displacement error and it tracks an acutely symmetrical circle. Another significant advantage of the fuzzy adaptive backstepping controller over any other control techniques employed in this thesis is the low computation time required to generate the control signals. This is very important since a lower computation time allows the control performance to increase dramatically.

Therefore, the recommended control technique to be employed on the planar two degrees of freedom parallel robot is the fuzzy adaptive backstepping controller.

# Table of Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Background

There are essentially two types of robot manipulators: serial and parallel. Serial manipulators consist of a number of links connected in series to one another to form a kinematic chain. Each joint of the kinematic chain is usually actuated. This type of structure is known as an open chained mechanism. Parallel manipulators, on the other hand, consist of a number of kinematic chains connected in parallel to one another. The kinematic chains work in unison to move a common point. This common point usually consists of a manipulator that performs a certain task. For the purpose of the two degrees of freedom planar parallel robot system described in this thesis, the common point will also be referred to as the end effector. Since the kinematic chains are eventually connected to a common point, a parallel manipulator is considered a closed chained mechanism. The actuators in parallel manipulators are usually located at the base or close to the base of the system, which is in stark contrast to serial manipulators which have actuators at every joint. The advantages of this type of configuration include the fact that it could achieve a higher load capacity due to the decrease in the mass of the overall system, it can produce high accelerations at the end effector and it has a high mechanical stiffness to weight ratio. The disadvantages of this type of configuration include the fact that the dynamic model is quite complex in nature and there are many instances of singularities that must be mapped out and avoided in order to maintain control of the system. Parallel robots come in a wide variety of designs and applications ranging from the Stewart platform, which is used in aircraft motion simulators to the Delta robot, which is used in packaging plants. This endows the fact that there cannot be a conclusive result as to which controller best suits the functionality of all parallel robots. Therefore, it is logical to experiment with various control techniques to observe upon which controller would garner the most satisfactory results based on a specific mechanical system.

This thesis presents the reader with the simulation and experimental results obtained from the implementation of non-adaptive and adaptive PD and backstepping controllers along with four fuzzy controllers on a planar two degrees of freedom parallel robot. The parameters of the dynamic model of this system are derived in detail followed by the derivation of the inverse kinematics of the mechanical model. The non-singular region is then defined based on the results obtained in the inverse kinematics. It is important to map out the non-singular region since it is the only location in which the parallel robot is able to operate under stable conditions. If the parallel robot were to enter a singular region, it would render the controller ineffective and cause the entire system to become unstable. It is impossible to adequately design any controllers for the parallel robot without a clear understanding of the dynamic model and the inverse kinematics of the mechanical model.

The eight controllers discussed in this thesis are separated into two main categories: non-fuzzy logic controllers and fuzzy logic controllers. The major difference between the two is the fact that the fuzzy logic controllers employ an ingenious concept known as natural language. Natural language essentially describes the process of human communication to solve a specific scenario. The various parameters involved in a scenario allow an individual to make a reasonable decision, but the deduction methods of another person may reach a different conclusion. This type of reasoning is not found in non-fuzzy logic controllers since the results of non-fuzzy logic controllers are generated utilizing a crisp boundary, such as binary code. This is known as a classical set. Fuzzy logic controllers employ the concept of fuzzy sets, which is a set without a clearly defined crisp boundary. It can contain elements with only a partial degree of membership. An excellent resource comparing the differences between the implementation of a non-fuzzy logic controller and a fuzzy logic controller to establish the tip that should be given based on the service received is given in [13].

Another concept that is deliberated in this thesis is the difference in control performance between adaptive and non-adaptive controllers. Non-adaptive controllers are generally less computationally intensive due to the fact that the system parameters do not change over time. In terms of the two degrees of freedom parallel robot discussed in this thesis, this pertains to the mass of each link, the length of each link, the moment of inertia about each link and the distance to the centre of mass for each link. The foremost issue concerning the use of this approach is that the moment of inertia and centre of mass of each link is not consistent throughout the operation of the robotic structure. The adaptive method counteracts this effect by estimating all the system parameters online. In order to impartially quantify the results between non-adaptive and adaptive controllers, the PD and backstepping controllers discussed in this thesis are implemented using both methods.

This thesis will provide the reader with a comprehensive comparison of the effects that various controllers would inflict on a planar two degrees of freedom parallel robot structure designed by the writer. The pros and cons of the simulation and experimental results of the robotic system will also be discussed in detail.

The overall purpose of the implementation of such a wide array of controllers is to demonstrate the robustness that such a system could achieve and ultimately choose a single controller that yields the most satisfactory results.

## 1.2 Literature Review

The word *robot* was first introduced to the world by the Czech playwright Karel Capek in his 1920 play called *Rossum's Universal Robots*. This play described a group of artificially created people that were used to serve mankind, but they eventually rebelled and exterminated the human populous. Karel initially wanted to name these artificial people *labori*, but thought the name sounded silly, so he looked to his older brother Josef for some guidance. Thus, Josef coined the word robot, which in Czech the word *robota* literally means forced labourers [77]. Following Karel Capek's play, many science fiction writers began to conjure up dramatic stories of the roles robots could play in the future world of mankind, whether destructive or beneficial. Authors such as Isaac Asimov defined a moral code for robots called the *Three Laws of Robotics*, while Gene Roddenberry created the fictitious positronic based android named Data. Even though the concept of robots is relatively modern, the actual implementation of robotic devices has been around for many centuries.

A robot is defined as an automatically guided machine that is able to do tasks on its own [42]. In this context, automata of the sort have been around since the 1st century with devices such as the coin operated machine and the aeolipile. The 18th and 19th centuries produced more elaborate automatons such as a mechanical duck that would simulate the ingestion of food and toys that would serve tea, yet it was not until the 20th century that genuine robots became more sophisticated and achieved practical industrial applications.

Robots can be more readily identifiable by the amount of relative agency. The more a control system seems to have agency of its own, the more likely the machine is to be called a robot. The most common classification of robots pertains to the anthropomorphic behaviour of the system, most notably either mentally anthropomorphic or physically anthropomorphic. A mentally anthropomorphic robot would describe a roboticized task that a human being could potentially do. A perfect example that fits this criterion would be a device such as a self guided automobile. The operator would input specific information such as the destination into a computer and in turn, the computer would automatically control the speed, direction and other important factors that the automobile would require to arrive at the destination safely. Physically anthropomorphic robots refer to machines that simulate human movement in some way or form. For example, an industrial robot welding a door to an automobile essentially simulates the movement of the human arm gracefully moving the welding rod to the desired position, while a biped robot gingerly simulates the walking patterns of a human being. It is the latter of these two anthropomorphic robots that will be discussed in the remainder of this literature review.

The first robot ever utilized in industrial applications was Unimate, which was designed in 1954 by George Devol. In 1961, the General Motors Corporation installed Unimate in their New Jersey assembly plant to transport die castings from an assembly line and spot weld them onto auto bodies [36]. This task was initially performed by trained human workers, yet even the most skilled worker could potentially seriously injure themselves due to the high dexterity involved and the inherently dangerous nature of the job. Unimate proved that it was possible for robots to perform certain tasks safer and more efficiently than human beings could.

This ultimately led to a large influx of more multi-functional and flexible robotic structures such as the Stanford Arm. Designed by Victor Scheinman in 1969, the Stanford Arm has the privilege of being the first electrically powered, computer controlled robotic arm. Scheinman went on to improve his design and created the famous six degrees of freedom PUMA robot in 1978 [58]. This robot is so reliable and robust that it is still in use in industrial applications today.

All these industrial robots discussed thus far have the same type of architecture; that is, they are all considered serial robots. Serial robots consist of a number of rigid bodies connected in series, each linked together by a one degree of freedom joint that may be actuated. This type of open loop mechanical structure has been proven to be extremely effective at manipulating objects in Cartesian space, hence its widespread use in the assembly and manufacturing industry. However, this type of mechanical architecture is not suitable for all tasks. One major drawback of this configuration is the fact that the absolute accuracy and the load capacity to robot mass ratio are relatively low. Absolute accuracy is defined as the distance between the desired and actual position of the end effector. This value decreases significantly as the number of degrees of freedom increases due to structural effects such as flexural deformations, complex high velocity motions and motor backlash. The absolute accuracy can be improved by implementing sophisticated internal sensors and increasing the quality of the geometric realization, but it is generally accepted that the absolute accuracy of a serial robot is poor [33]. In terms of manipulating heavy loads, each link has to support the weight of the following segment in addition to the load; hence the links are subject to large flexure torques. To compensate for this effect the links must be stiffened, which in turn increases the overall mass of the robot. This increases the cost of the robot substantially due to the higher quality of materials required to counteract the forces acting on the links as well as more powerful actuators to adequately operate the apparatus. Therefore, serial robots are unsuitable for tasks requiring the manipulation of heavy loads or good positioning accuracy. In order to solve these issues a completely different type of mechanical architecture was pioneered, known as parallel robots.

Parallel robots vary significantly from their serial counterparts, but their most apparent difference arises in the layout of the mechanical structure. Parallel robots consist of a number of kinematic chains connected in parallel to one another, which work in concert to move the end effector to the desired position. This type of closed chained structure was first practically postulated by Dr. Eric Gough while working for the Dunlop Rubber Company in Birmingham, England, in 1947. A universal machine was needed to determine the properties of tires under certain load conditions, hence his variable length strut octahedral hexapod robot proved to be the solution. Dr. Gough successfully built a fully functional prototype of this multi-simulation table in 1955. The hexagonal platform end effector is manipulated with six degrees of freedom utilizing six linear actuators connected to each point of the hexagon. By varying the length of each actuator, the end effector could achieve the desired effect against the tire [18]. Curiously, it is not Dr. Gough who is epitomized as the father of modern parallel robotics; instead it is a researcher by the name of D. Stewart.

Mr. Stewart had a dilemma concerning the suitable needs of simulating flight conditions for pilots in training. An apparatus was needed to provide all the ranges of motion associated with flight, yet attain a high load capacity to system mass ratio without compromising the price point. Mr. Stewart's solution was a six degree of freedom parallel robot that consisted of a triangular platform end effector in which the simulator would sit atop of. A linear actuator is connected to each point of the triangle along with three secondary linear actuators connected to each of the previous three actuators. This type of setup would allow for six degrees of freedom due to the reason that all the linear actuators utilize a two axis joint connecting them to the foundation of the system. Although the theory itself is sound, no one has physically built a prototype of Stewart's proposed model. The main reason Stewart's name is associated with all types of parallel robots that employ an octahedral assembly of struts is because of Gough's fully functioning tire testing parallel robot, which was discussed and shown in the reviewers' remarks section of the paper Mr. Stewart published in 1965 [53]. The first flight simulator was actually invented by an American engineer by the name of Klaus Cappel in 1962 while working for the Franklin Institute Research Laboratories in Philadelphia. The mechanism Mr. Cappel conjured up was exactly the same as the variable length strut octahedral hexapod robot already being implemented by Dr. Gough even though he had no prior knowledge that such a system existed [5]. The history of the first parallel robots between academia and the industry can be described as very isolated to say the least, but the contributions of all three men allowed for the popularity and applications of parallel robots to rise substantially.

The amount of parallel robot structures that have been developed over the past few decades is austerely mindboggling. Unlike their serial robot counterparts, parallel robots can comprise of a very large variety of closed loop mechanisms with each new topology affecting the overall performance of the robot. This has led to a classification system based on the amount of degrees of freedom and by the type of joints or actuators that parallel robots employ. The number of degrees of freedom is determined by the amount of actuators present on the mechanical structure. Each actuator allows the coupled kinematic chain to operate in a specific direction. Every kinematic chain consists of at least one of the following joints or actuators: revolute, prismatic, universal or ball-and-socket. An example of a revolute joint would be a shaft that permits the link to rotate along a planar surface. A prismatic joint could represent a linear actuator that allows the link to increase or decrease in length. A universal joint is a joint in a rigid rod that allows the rod to rotate in any direction. It consists of a pair of hinges that are located close together and oriented at 90 degrees from one another, connected by a cross shaft. Ball-and-socket joints are spherical in nature and allow rotation in an infinite number of axes, albeit they are physically limited by the design constraints of the node. The most common mechanical structures of parallel robots consist of three or six degrees of freedom with identical kinematic chains. This is mainly due to the fact that these types of mechanical structures have been shown to work extremely well at manipulating various devices located at the end effector with a high degree of repeatability.

Three degrees of freedom parallel robots have been studied quite extensively over the years, resulting in innovations such as the Tricept robot [6] and an unnamed prismatic-universal-universal configuration defined in [73]. The most famous of them is the three translational degrees of freedom Delta robot. The Delta robot was conceived by Dr. Reymond Clavel in the early 1980s [9]. There are certain versions of the Delta robot which make use of an extra revolute degree of freedom. A shaft is connected from the base of the parallel robot that allows the end effector to rotate 360 degrees. The purpose of its creation was to achieve an apparatus which could manipulate small, light objects at very high speeds. The forward kinematics for such a robotic structure are defined in [34], while the fundamental guidelines for the optimal design of the Delta parallel robot based on the genetic algorithm approach are defined in [52]. Applications of the Delta robot can be found all over the industrial sector of society ranging from the packaging industry to the pharmaceutical industry. The same architecture has been used in more unconventional scenarios, such as neurosurgery [10] and playing table tennis [50].

Six degrees of freedom parallel robots are generally referred to as a hexapod robot in reference to the Stewart platform. As previously described, this type of mechanical structure has been used in the aerospace industry as flight simulators and in the entertainment industry as motion simulators. The inverse and forward kinematics of the Stewart platform have been studied quite extensively, yet Mr. Gregorio proposed a method to calculate the forward kinematics of the last remaining architecture yet to be fully understood, namely the spherical-prismatic prismatic-spherical revolute-spherical six degrees of freedom parallel robot [11]. Another popular six degrees of freedom parallel robot is the Hexa. The Hexa robot is a derivative of the Delta robot, which was designed by Dr. Pierrot in 1991 [40]. The rationalization behind this structural design was the fact that it has been shown that Delta parallel robots can achieve high speed and precision, yet they are limited by their number of degrees of freedom. The Hexa parallel robot has similar dynamic properties as its predecessor, yet it also has greater manoeuvrability. Hence, it is more suitable for tasks such as laser cutting. The inverse kinematics and inverse dynamics of the Hexa robot structure have been defined in [1].

This leads to the structure that will be implemented in this thesis; a two degree of freedom parallel robot. One of the first two degrees of freedom parallel robot apparatuses was based off the original Delta robot design. Dr. Ghorbel proposed a methodology to derive the equations of motion of a planar version of the Delta parallel robot [16]. He went a step further to analyze the experimental effects on closed chained mechanisms by building the Rice planar Delta robot. Dr. Ghorbel proved that the Rice planar Delta robot could satisfy a skew symmetry property, which guaranteed local asymptotic stability by utilizing a proportional derivative controller with gravity compensation [17]. In the most recent work on the Rice planar Delta robot, the control of closed kinematic chains using a singularly perturbed dynamic model have been described in [66]. Another two degrees of freedom parallel structure that has been proposed is the parallel translating robot. It utilizes a passive mechanism which can translate freely along a circular path based on the universal-prismatic-universal parallel manipulator in the singularity configuration. There are two compound limbs that connect a moving platform to a fixed platform. Each compound limb consists of an actuated linear slide and the passive mechanism. The motions of the two actuated linear slides are parallel to one another in a plane [39]. The planar two degrees of freedom parallel robot described in this thesis is a derivative of the Rice planar Delta robot. The main focus is to compare the differing controller results implemented on a similar mechanical structure. Before an analysis on the various types of controllers is presented, the dynamic equations and singular regions of any parallel robot structure must be well understood.

As Tsai noted in [60], research on parallel modelling has been more focused on the kinematics structure than the dynamics structure. Kinematics describes the study of the motion of objects without the consideration of the causes leading to motion, while dynamics describes the study of the causes of motion. The inverse kinematic equations of a parallel robot structure will yield the angles of all the robot's joints when given the desired position of the end effector. It could also solve for the velocity and acceleration if the desired velocity and acceleration of the end effector is known. Alternatively, the dynamic equations can solve for the robot's joint angles without any knowledge of the desired position of the end effector, but its main purpose is to solve for the actual position, velocity and acceleration required by each actuator wherever possible. It employs the system constraints to approximate a mathematical model for the parallel robot structure. Generally, the inverse kinematics of a parallel robot is relatively simple to solve since the kinematic chains rarely implement more than three links. The dynamic equations are usually quite complex in nature due to the constraints found in any parallel robot. Dr. Kovecses et al. addressed the methods for dynamic modelling of constrained robotic systems based on the differential variational principles of constrained dynamic systems [25]. A dynamics based trajectory planning technique was also well documented in [31]. Regardless of how efficient the modelling technique of a parallel robot is, all would be for naught if the desired trajectory of the system approached or passed through a singular point.

A singular point is a location that can achieve no solution with the given constraints; hence the parallel robot would become out of control. Singular points can be located in the reachable region of the end effector of any parallel robot, which is why it is crucial that they be mapped out and avoided. There has been substantial research in preventing such a scenario to occur [32], yet an ingenious proposal has been made to completely avoid the existence of singular regions. A comprehensive and straightforward design strategy that guarantees a singularity free workspace is presented in [71]. In this paper, Mr. Yang et al. proved that a three degree of freedom translational universal-prismatic-universal parallel robot structure can achieve a contiguous singularity free workspace. Mr. Kotlarski et al. also proved that a three degree of freedom revolute-revolute-revolute parallel robot structure can accomplish a singularity free workspace using an interval based approach [24].

An in depth analysis and history of the control techniques implemented in this thesis will be presented in the latter portions where the specified controller is derived. This literature review will focus strictly on the implementation of various controllers on other parallel robot structures. There are vast arrays of controllers that have been simulated and experimented on parallel robot apparatuses. The most commonly utilized are PID related controllers [30], [70], [45]. Robust controllers [12] and adaptive variants of these controllers [21] have also been studied quite extensively, but the most interesting type of control technique pertains to fuzzy logic control.

Fuzzy logic controllers are exemplified by the use of linguistic variables to represent part of the range of an ordinary crisp variable. It allows for better modelling and control of real world nonlinear systems. Dr. Zadeh invented the first fuzzy system as a graduate student for Columbia University in the 1960s. His technique spawned an entire collection of works based on improving his method, which can be found in all aspects of society. A specifically noteworthy example is the subway system in Sendai, Japan. It was the first subway structure to replace the human operator with a fuzzy system in 1988 [23].

In terms of robotics, many fuzzy systems are employed using the Takagi-Sugeno modelling approach. This interest relies on the fact that dynamic Takagi-Sugeno fuzzy models are easily obtained by the linearization of the nonlinear plant around different operating points. Once the Takagi-Sugeno models are obtained, a linear control methodology can be used to design the desired controllers for each linear model [55]. Dr. Tanaka proved the stability of the Takagi-Sugeno model in [56] and Dr. Sugeno described how effective the Takagi-Sugeno model is at defining a global functional structure for a nonlinear process [54]. However, a multitude of papers have been written proving the stability of this model with differing Lyapunov functions [57], [41], [69], [27]. There have also been a few researchers who have employed the linear matrix inequality to simplify the stability analysis and control design problems. This design methodology was exploited by Mr. Wang et al. to successfully balance and swing up an inverted pendulum on a cart [62]. Similarly, Mr. Khaber et al. designed a state feedback controller utilizing the linear matrix inequality approach to simplify the Takagi-Sugeno fuzzy controller [22].

The implementation of fuzzy controllers on parallel robots has also been explored recently; in many instances, they have been employed as a combination of multiple control techniques. A cable driven auto levelling parallel robot was designed by Mr. Yu et al. that developed a hierarchical fuzzy logic controller. The hierarchical fuzzy controller contained two layers: the low level layer which generated two outputs for levelling adjustment and the high level layer which coordinated the two outputs from the low level layer [72]. Mr. Zeinali et al. proposed a fuzzy model based adaptive robust control scheme for the tracking control of a four degree of freedom parallel manipulator with uncertain dynamics [75]. An interesting controller technique pertains to the adaptation of fuzzy sliding mode control. The sliding mode portion of the controller is based on variable structure control, but the author proved that variable structure control is unstable for physical sampled systems. In order to compensate for this fact, he employed a fuzzy logic controller to stabilize the six degrees of freedom parallel robot structure [3]. Any mechanical system is always susceptible to derivative information such as perturbations, yet most fuzzy systems do not have the ability to apprehend this type of data. Mr. Salgado et al. proposed a perturbed fuzzy system that is capable of modelling the derivative information, while maintaining the inference mechanism and structure model of a traditional fuzzy system [43].

The final segment that will conclude this literature review is the implementation of a network controller. Network control is by far the most important type of controller technique available in our globalized society. Robotic mechanisms are not only limited to industrial manufacturing applications or the simulation of anthropomorphic motion; they play a substantial role in allowing accessibility to areas that are either inaccessible or extremely hazardous. Doctors can now perform complex surgical operations without ever physically touching the patient. A precise robotic arm can simulate the movements of the doctors' arm and send the feedback signals of these movements back to the operator through TCP/IP based Internet communication to garner the appropriate responses [47]. A local area network controlled manipulator was proposed by Alan Mutka et al. that could inspect the welds inside a nuclear reactor [35]. This enables an operator to perform the entire inspection procedure remotely over a network, thus avoiding exposure to the dangerous radioactive particles which are commonly present in nuclear reactor environments. Rovers are also very popular robots that are controlled via a network. The most prevalent example pertains to the rovers that NASA sent to Mars, namely: Sojourner, Spirit and Opportunity. Humans, as of yet, have not set foot on Martian soil, so these rovers are controlled over an extremely large wireless network in order to explore the unknown. Vishwanath Chukkala et al. proposed a way to model the radio frequency environment of Mars to ensure reliable and efficient communication between any future rovers on Mars along with the operators here on the Earth [8]. Relating to parallel robots, Changfeng Li et al. have been experimenting with a six degree of freedom parallel robot utilized in inertial confinement fusion. This ingenious method employs an extremely precise micro-motion manipulator that can operate in a very confined chamber to control the nuclear fusion reaction. Their robotic apparatus has an inherently high autoimmunization; hence they used network control to simplify the control algorithms [29]. For the record, there has not been a successful inertial confinement fusion reaction that generates more energy than was induced in the system to date.

The following section will discuss the thesis overview.

## 1.3 Thesis Overview

The purpose of this thesis is to determine the most appropriate controller to implement on a planar two degrees of freedom parallel robot apparatus. Chapter 2 will discuss and derive the equations for the modelling of the parallel robot using the dynamic equations of the constrained system and the inverse kinematics of the mechanical structure. Chapter 3 will consist of the derivations of four non-fuzzy logic controllers based on the system parameters. The non-fuzzy logic controllers discussed in this thesis consist of the non-adaptive and adaptive PD and backstepping controllers. Chapter 4 will consist of the derivations of four fuzzy logic controllers. The fuzzy logic controllers discussed in this thesis consist of a fuzzy PD controller implementing a singleton fuzzifier, the product inference engine and a centre average defuzzifier along with an indirect adaptive fuzzy controller, a direct adaptive fuzzy controller and a fuzzy adaptive backstepping controller. Chapter 5 will compare and analyze the simulation results of each controller utilizing MATLAB. The plots of the joint angles and end effector trajectory along with their respective errors and torque will be compared between all the controllers and a generalized conclusion of these simulation results will be garnered. Chapter 6 will comprise of the electrical and mechanical specifications of the planar two degrees of freedom parallel robot. The schematics and tables associated with the specifications for the DSP board and the motor driver board can be found in Appendix A and Appendix B, respectively. Chapter 7 will detail the practical experimentation results of the parallel robot using the controllers discussed in Chapter 3 and Chapter 4. As with the simulated system, the plots of the joint angles and end effector trajectory along with their respective errors and PWM will be compared between all the controllers. Chapter 8 will entail the overall recommendation of the candidate controller which best suits the needs of the parallel robot system. A description of the improvements or additions that can be executed in future research endeavours will be investigated to conclude this thesis.

# Chapter 2

# Modelling

This chapter will describe the methodology of determining how to calculate the unknown constraints of the parallel robot that are needed to accurately model the movements of the parallel robot.

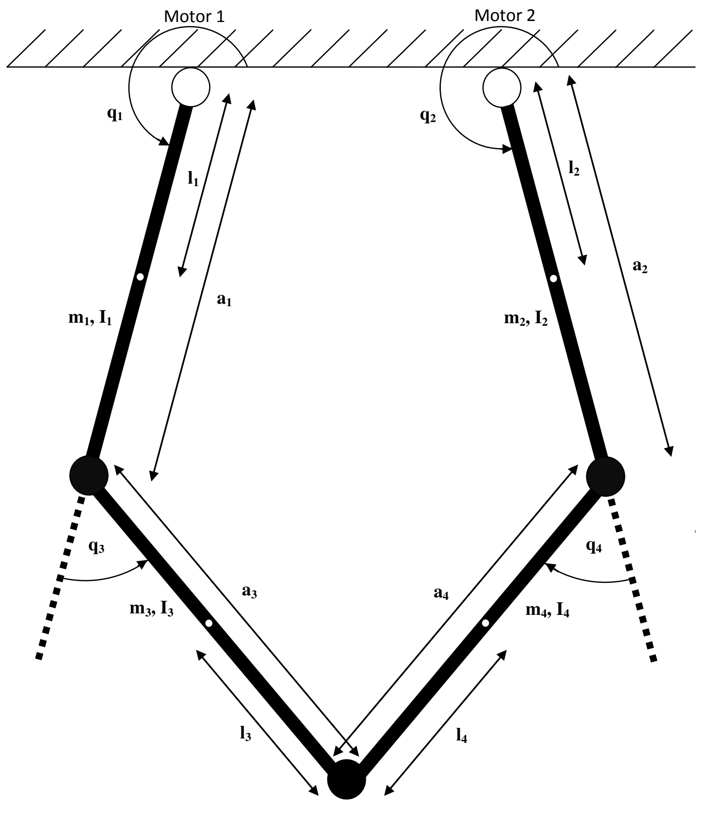


Figure 2.1: Planar Two Degrees of Freedom Parallel Robot

## 2.1 Dynamic Equations

The constrained two degrees of freedom system studied in this report is shown in Figure 2.1. It should be noted that the parameters of each individual link are known, which include: mass ($m_i$), distance to the centre of mass ($l_i$), moment of inertia ($I_i$), the overall length of the link ($a_i$), the force of gravity ($g$), which is equal to 9.8 metres per second squared and where $i$ represents the link number.

The differential equation described in [63] defines the dynamic model of the reduced model constrained system as:

$$D'(q')\ddot{q}' + C'(q',\dot{q}')\dot{q}' + g'(q') = u' \tag{2.1}$$
$$\phi(q') = 0 \tag{2.2}$$

where:

the constraint $\phi(q')$ is at least twice continuously differentiable;

$q' = \begin{bmatrix} q_1 & q_2 & q_3 & q_4 \end{bmatrix}^T$ describes the actual angles at each joint;

$u' = \begin{bmatrix} u_1 & u_2 & 0 & 0 \end{bmatrix}^T$ describes the torque applied at the actuated joints;

$D'(q') \in \mathbb{R}^4$ describes the inertia matrix;

$C'(q',\dot{q}')\dot{q}' \in \mathbb{R}^4$ describes the centrifugal and Coriolis terms and

$g'(q') \in \mathbb{R}^4$ describes the gravity vector

Now that the dynamic equations for a constrained system have been defined, the equations of motion for the parallel robot need to be solved.

The equations of motion of the constrained system expressed in terms of the independent generalized coordinates as described in [15] are defined as:

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u \tag{2.3}$$

which are obtained by combining the following formulae:

$$\begin{aligned} &D(q')\ddot{q} + C(q',\dot{q}')\dot{q} + g(q') = u && (2.4)\\ &\dot{q}' = \rho(q')\dot{q} && (2.5)\\ &q' = \sigma(q) && (2.6)\end{aligned}$$

where:

$$\begin{aligned} D(q') &= \rho(q')^T D'(q')\rho(q') && (2.7)\\ C(q',\dot{q}') &= \rho(q')^T C'(q',\dot{q}')\rho(q') + \rho(q')^T D'(q')\dot{\rho}(q',\dot{q}') && (2.8)\\ g(q') &= \rho(q')^T g'(q') && (2.9)\\ \sigma(q) &= \left[\begin{array}{cccc} q_1 & q_2 & q_3 & q_4 \end{array}\right]^T \end{aligned}$$

with $q_3$ and $q_4$ defined in equations (2.19) and (2.20), respectively.

The actuated joints are represented as:

$$q = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} q' \quad (2.10)$$

$$\dot{q} = \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \dot{q}' \quad (2.11)$$

$$\ddot{q} = \begin{bmatrix} \ddot{q}_1 \\ \ddot{q}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \ddot{q}' \quad (2.12)$$

Recall that the constraint equations must also be accounted for:

$$\phi(q') = \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = 0 \quad (2.13)$$

where:

$$\begin{aligned} \phi_1 &= a_1\cos q_1 + a_3\cos(q_1+q_3) - c - a_2\cos q_2 - a_4\cos(q_2+q_4) = 0 && (2.14)\\ \phi_2 &= a_1\sin q_1 + a_3\sin(q_1+q_3) - a_2\sin q_2 - a_4\sin(q_2+q_4) = 0 && (2.15)\end{aligned}$$

Using the Lagrangian method, the following matrices can be derived:

$$D'(q') = \begin{bmatrix} d_{11} & 0 & d_{13} & 0 \\ 0 & d_{22} & 0 & d_{24} \\ d_{31} & 0 & d_{33} & 0 \\ 0 & d_{42} & 0 & d_{44} \end{bmatrix} \tag{2.16}$$

$$C'(q', \dot{q}') = \begin{bmatrix} c_{11} & 0 & c_{13} & 0 \\ 0 & c_{22} & 0 & c_{24} \\ c_{31} & 0 & 0 & 0 \\ 0 & c_{42} & 0 & 0 \end{bmatrix} \tag{2.17}$$

$$g'(q') = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \tag{2.18}$$

where:

$$\begin{aligned}
d_{11} &= m_1 l_1^2 + m_3(a_1^2 + l_3^2 + 2a_1 l_3 \cos q_3) + I_1 + I_3 \\
d_{22} &= m_2 l_2^2 + m_4(a_2^2 + l_4^2 + 2a_2 l_4 \cos q_4) + I_2 + I_4 \\
d_{13} &= m_3(l_3^2 + a_1 l_3 \cos q_3) + I_3 \\
d_{24} &= m_4(l_4^2 + a_2 l_4 \cos q_4) + I_4 \\
d_{31} &= d_{13} \\
d_{42} &= d_{24} \\
d_{33} &= m_3 l_3^2 + I_3 \\
d_{44} &= m_4 l_4^2 + I_4 \\
h_1 &= -m_3 a_1 l_3 \sin q_3 \\
h_2 &= -m_4 a_2 l_4 \sin q_4 \\
c_{11} &= h_1 \dot{q}_3 \\
c_{22} &= h_2 \dot{q}_4 \\
c_{13} &= h_1(\dot{q}_1 + \dot{q}_3) \\
c_{24} &= h_2(\dot{q}_2 + \dot{q}_4) \\
c_{31} &= -h_1 \dot{q}_1 \\
c_{42} &= -h_2 \dot{q}_2 \\
g_1 &= g((m_1 l_1 + m_3 a_1) \cos q_1 + m_3 l_3 \cos(q_1 + q_3)) \\
g_2 &= g((m_2 l_2 + m_4 a_2) \cos q_2 + m_4 l_4 \cos(q_2 + q_4)) \\
g_3 &= g(m_3 l_3 \cos(q_1 + q_3)) \\
g_4 &= g(m_4 l_4 \cos(q_2 + q_4))
\end{aligned}$$

In the case of the physical structure of the parallel robot, the joint angles $q_1$ and $q_2$ are actuated. To improve performance, a safety net was employed while simulating the system which limited the range of the angles for the actuated joints to never traverse outside the span of -30 and -150 degrees. Therefore, at any given point in the operation of the robot, these angles are known. Although, to properly simulate the control of this system, the joint angles $q_3$ and $q_4$ must be known. In order to achieve this, it is possible to manipulate equations (2.14) and (2.15) to obtain the following results:

$$q_4 = -\tan^{-1}\left[\frac{\sqrt{\tilde{A}^2 + \tilde{B}^2 - \tilde{C}^2}}{\tilde{C}}\right] + \tan^{-1}\left[\frac{\tilde{B}}{\tilde{A}}\right] - q_2 - \pi \tag{2.19}$$

$$q_3 = \tan^{-1}\left[\frac{\tilde{\mu} + a_4 \sin(q_2 + q_4)}{\tilde{\lambda} + a_4 \cos(q_2 + q_4)}\right] - q_1 \tag{2.20}$$

where:

$\tilde{\lambda} = a_2 \cos q_2 - a_1 \cos q_1 + c$
$\tilde{\mu} = a_2 \sin q_2 - a_1 \sin q_1$
$\tilde{A} = 2a_4\tilde{\lambda}$
$\tilde{B} = 2a_4\tilde{\mu}$
$\tilde{C} = a_3^2 - a_4^2 - \tilde{\lambda}^2 - \tilde{\mu}^2$

Now by combining equations (2.13) and (2.10) and differentiating with respect to $q'$ the result would become:

$$\psi_{q'}(q') = \begin{bmatrix} \psi_{q'_{11}} & \psi_{q'_{12}} & \psi_{q'_{13}} & \psi_{q'_{14}} \\ \psi_{q'_{21}} & \psi_{q'_{22}} & \psi_{q'_{23}} & \psi_{q'_{24}} \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.21}$$

where:

$\psi_{q'_{11}} = -a_1 \sin(q_1) - a_3 \sin(q_1 + q_3)$
$\psi_{q'_{12}} = a_2 \sin(q_2) + a_4 \sin(q_2 + q_4)$
$\psi_{q'_{13}} = -a_3 \sin(q_1 + q_3)$
$\psi_{q'_{14}} = a_4 \sin(q_2 + q_4)$
$\psi_{q'_{21}} = a_1 \cos(q_1) + a_3 \cos(q_1 + q_3)$
$\psi_{q'_{22}} = -a_2 \cos(q_2) - a_4 \cos(q_2 + q_4)$
$\psi_{q'_{23}} = a_3 \cos(q_1 + q_3)$
$\psi_{q'_{24}} = -a_4 \cos(q_2 + q_4)$

By differentiating equation (2.21) with respect to time, it is possible to achieve:

$$\dot{\psi}_{q'}(q',\dot{q}') = \begin{bmatrix} \dot{\psi}_{q'_{11}} & \dot{\psi}_{q'_{12}} & \dot{\psi}_{q'_{13}} & \dot{\psi}_{q'_{14}} \\ \dot{\psi}_{q'_{21}} & \dot{\psi}_{q'_{22}} & \dot{\psi}_{q'_{23}} & \dot{\psi}_{q'_{24}} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.22}$$

where:

$$\begin{aligned}
\dot{\psi}_{q'_{11}} &= -a_1\dot{q}_1\cos(q_1) - a_3(\dot{q}_1+\dot{q}_3)\cos(q_1+q_3)\\
\dot{\psi}_{q'_{12}} &= a_2\dot{q}_2\cos(q_2) + a_4(\dot{q}_2+\dot{q}_4)\cos(q_2+q_4)\\
\dot{\psi}_{q'_{13}} &= -a_3(\dot{q}_1+\dot{q}_3)\cos(q_1+q_3)\\
\dot{\psi}_{q'_{14}} &= a_4(\dot{q}_2+\dot{q}_4)\cos(q_2+q_4)\\
\dot{\psi}_{q'_{21}} &= -a_1\dot{q}_1\sin(q_1) - a_3(\dot{q}_1+\dot{q}_3)\sin(q_1+q_3)\\
\dot{\psi}_{q'_{22}} &= a_2\dot{q}_2\sin(q_2) + a_4(\dot{q}_2+\dot{q}_4)\sin(q_2+q_4)\\
\dot{\psi}_{q'_{23}} &= -a_3(\dot{q}_1+\dot{q}_3)\sin(q_1+q_3)\\
\dot{\psi}_{q'_{24}} &= a_4(\dot{q}_2+\dot{q}_4)\sin(q_2+q_4)
\end{aligned}$$

Finally, the last two expressions for the dynamic equations are defined as:

$$\rho(q') = \psi_{q'}^{-1}(q') \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ \rho_{31} & \rho_{32} \\ \rho_{41} & \rho_{42} \end{bmatrix} \tag{2.23}$$

$$\dot{\rho}(q',\dot{q}') = -\psi_{q'}^{-1}(q')\dot{\psi}_{q'}(q',\dot{q}')\rho(q') = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \dot{\rho}_{31} & \dot{\rho}_{32} \\ \dot{\rho}_{41} & \dot{\rho}_{42} \end{bmatrix} \tag{2.24}$$

It should be noted that $\dot{D}(q') - 2C(q',\dot{q}')$ is skew symmetric [14]. It should also be noted that the reduced model is an implicit model since the parameterization of $q' = \sigma(q)$ is implicit and it is only valid locally due to the presence of the singularity. The following factors are not taken into account in the modelling of the parallel robot system: friction between joints, motor dynamics, gear train backlash, and link elasticity.

## 2.2 Inverse Kinematics

The purpose of determining the inverse kinematics of this parallel robot is to accurately model the angle produced at each joint at a specific location of the end effector. This is advantageous for two main reasons; the first being that it is relatively simple to define any reasonable trajectory for the end effector to traverse and secondly, it can track different trajectories in a non-singular region.

The constrained two degrees of freedom system shown in Figure 2.1 will also be applicable in this section. It should be noted that the parameters of the overall system are known, which include: the range of the desired angles for $q_1$ and $q_2$ respectively, the overall length of each link ($a$), the desired location of the end effector in the $x$ and $y$ axis respectively and the horizontal distance between the two motor shafts ($c$).

From Figure 2.1, the following equations can be derived:

$$x = a_1 \cos(q_1) + a_3 \cos(q_1 + q_3) \tag{2.25}$$
$$y = a_1 \sin(q_1) + a_3 \sin(q_1 + q_3) \tag{2.26}$$
$$x = a_2 \cos(q_2) + a_4 \cos(q_2 + q_4) + c \tag{2.27}$$
$$y = a_2 \sin(q_2) + a_4 \sin(q_2 + q_4) \tag{2.28}$$

By implementing the summing of squares method on equations (2.25) and (2.26), it is possible to achieve the following result:

$$x^2 + y^2 = a_1^2 + a_3^2 + 2a_1a_3 \cos(q_3) \tag{2.29}$$

Now it is quite simple to isolate $q_3$ from equation (2.29) to obtain:

$$q_3 = \cos^{-1}\left[\frac{x^2 + y^2 - a_1^2 - a_3^2}{2a_1a_3}\right] \tag{2.30}$$

Similarly, the application of the summing of squares method on equations (2.27) and (2.28) produces:

$$(x-c)^2 + y^2 = a_2^2 + a_4^2 + 2a_2a_4\cos(q_4) \tag{2.31}$$

Isolating $q_4$ from equation (2.31) allows the result to become:

$$q_4 = -\cos^{-1}\left[\frac{(x-c)^2 + y^2 - a_2^2 - a_4^2}{2a_2a_4}\right] \tag{2.32}$$

Since the range of $q_1$ and $q_2$ is constantly between -30 and -150 degrees, the trigonometric identity: $\sin^2 x + \cos^2 x = 1$ can be rearranged to produce:

$$\sin(q_1) = -\sqrt{1-\cos^2(q_1)} \tag{2.33}$$

$$\sin(q_2) = -\sqrt{1-\cos^2(q_2)} \tag{2.34}$$

By implementing another trigonometric identity: $\cos(x+y) = \cos x\cos y - \sin x\sin y$, it is possible to insert equations (2.33) and (2.34) into equations (2.25) and (2.27) respectively to obtain:

$$-\left(\overline{A}_1\cos^2(q_1) + \overline{B}_1\cos(q_1) + \overline{C}_1\right) = 0 \tag{2.35}$$

$$-\left(\overline{A}_2\cos^2(q_2) + \overline{B}_2\cos(q_2) + \overline{C}_2\right) = 0 \tag{2.36}$$

where:

$\overline{A}_1 = a_1^2 + a_3^2 + 2a_1a_3\cos(q_3)$
$\overline{B}_1 = -2x(a_1 + a_3\cos(q_3))$
$\overline{C}_1 = -a_3^2\sin^2(q_3) + x^2$
$\overline{A}_2 = a_2^2 + a_4^2 + 2a_2a_4\cos(q_4)$
$\overline{B}_2 = -2(x-c)(a_2 + a_4\cos(q_4))$
$\overline{C}_2 = -a_4^2\sin^2(q_4) + (x-c)^2$

Finally, solving equations (2.35) and (2.36) for $q_1$ and $q_2$ will yield:

$$q_1 = -\cos^{-1}\left[\frac{-\overline{B}_1 - \sqrt{\overline{B}_1^2 - 4\overline{A}_1\overline{C}_1}}{2\overline{A}_1}\right] \tag{2.37}$$

$$q_2 = -\cos^{-1}\left[\frac{-\overline{B}_2 + \sqrt{\overline{B}_2^2 - 4\overline{A}_2\overline{C}_2}}{2\overline{A}_2}\right] \tag{2.38}$$

Now that all the inverse kinematic equations have been derived, it is desirable to define the non-singular region. The purpose of the non-singular region is to determine the areas in which the parallel robot is controllable versus the singular region which determines where the parallel robot is uncontrollable. The parameter values used for the two degrees of freedom parallel robot discussed in this report are stated in Table 2.1. It should be noted that the horizontal distance between the two motor shafts ($c$) is 0.198 metres.

| Link | $m$ (kg) | $a$ (m) | $l$ (m) | $I$ (kg•m$^2$) |
|---|---|---|---|---|
| 1 | 0.0821 | 0.22688 | 0.11344 | 0.00063138 |
| 2 | 0.0855 | 0.22688 | 0.11344 | 0.00061735 |
| 3 | 0.12555 | 0.22688 | 0.11344 | 0.00053855 |
| 4 | 0.14413 | 0.22688 | 0.11344 | 0.00061825 |

Table 2.1: Parallel Robot Parameters

If a point satisfies the condition $\det[\psi_{q'}(q')] = \sin(q_1 + q_3 - q_2 - q_4) = 0$, then the end effector cannot be tracked, hence producing a singular point. Those trajectories in the non-singular region that do not cross or approach the singular points can be tracked. The solid black line shown in Figure 2.2 denotes the boundary of the reachable region of the end effector, while the white area outside this region is the unreachable region of the end effector. The shaded area defines a separate non-singular region that is isolated from the white non-singular region due to the presence of the singular points located at the base of the shaded region. It should be noted that motor one is located at the origin of the plot.

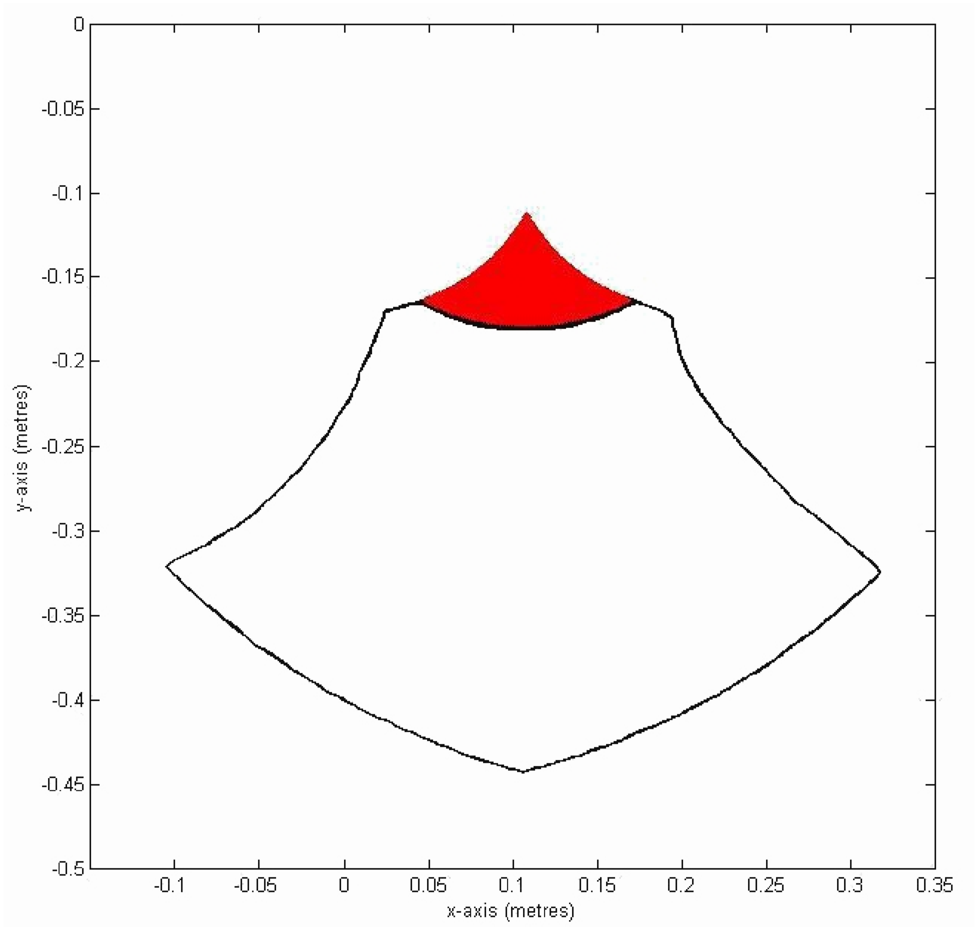


Figure 2.2: Non-Singular Region of the End Effector

# Chapter 3

# Non-Fuzzy Logic Controller Design

This chapter will describe the derivation of the non-adaptive and adaptive PD and backstepping controllers. A thorough mathematical analysis will be presented concerning the generation of the controller equation and the stability of the closed loop system will be justified. The adaptive controllers will also encompass an estimator equation that is used to estimate the parameters of the system.

## 3.1 Controller Background

The goal of implementing any type of controller is to observe the output response it would generate based on the inputted conditions. In order to achieve this, it is necessary to solve for the control input ($u$) of the system, which is essentially manipulating equation (2.4) into a suitable form. Each controller has a different method pertaining to how this equation is obtained, but the initial steps to reach this point are all similar.

The end effector of the two degrees of freedom parallel robot will follow a circular based trajectory; hence for tracking control it is appropriate to set the error and change in error as: $x_1 = q_{1d} - q_1$, $x_2 = q_{2d} - q_2$, $x_3 = \dot{q}_{1d} - \dot{q}_1$ and $x_4 = \dot{q}_{2d} - \dot{q}_2$, where: $q_{1d}$ and $q_{2d}$ are the desired angles; $q_1$ and $q_2$ are the actual angles; $\dot{q}_{1d}$ and

$\dot{q}_{2d}$ are the desired angular velocities; $\dot{q}_1$ and $\dot{q}_2$ are the actual angular velocities; $\ddot{q}_{1d}$ and $\ddot{q}_{2d}$ are the desired angular accelerations. The following system is in lower triangular form, which can be produced by differentiating $x_1$, $x_2$, $x_3$ and $x_4$.

$$
\begin{aligned}
\dot{x}_1 &= x_3 &(3.1)\\
\dot{x}_2 &= x_4 &(3.2)\\
\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} &= \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + D^{-1}(q')\left(-u + C(q', \dot{q}')\begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + g(q')\right) &(3.3)
\end{aligned}
$$

One aspect that constantly appears when implementing the appropriate controller is the feed forward term $u_d$. This term represents the desired control input required in the overall system operation. In theory, the actual and desired control input should be identical, but due to system disturbances and the force of gravity, this is known not to be the case. By adding $u_d$ into the specified controller, improved control performance can be achieved. It is defined as:

$$u_d = D(q_d')\ddot{q}_d + C(q_d', \dot{q}_d{}')\dot{q}_d + g(q_d') \tag{3.4}$$

Notice that this desired control input equation is the same as the actual control input in equation (2.4) when $q$, $\dot{q}$ and $\ddot{q}$ are the same as $q_d$, $\dot{q}_d$ and $\ddot{q}_d$, respectively. It should also be noted that $D(q_d')$, $C(q_d', \dot{q}_d{}')$ and $g(q_d')$ are the matrices calculated using equations (2.7), (2.8) and (2.9), respectively. To calculate $\ddot{q}_d$ and $\dot{q}_d$, the simplest method involves taking the first and second derivative of equations (2.25), (2.26), (2.27) and (2.28) found in the inverse kinematics section and then isolating them for the desired parameters. The following controllers never directly employ equation (3.4), but there is a clear resemblance that can be seen by setting the control and error parameters equal to zero.

Now it is possible to apply the lower triangular system described in equations (3.1), (3.2) and (3.3) towards the four non-fuzzy controllers utilized in this thesis, namely the non-adaptive and adaptive backstepping and PD controllers.

## 3.2 Non-Adaptive Backstepping Controller Design

Backstepping is a recursive Lyapunov based scheme proposed in 1990 by Petar Kokotovic. This technique is described in detail in [26]. The idea of backstepping is to design a controller recursively by considering some of the state variables as virtual controls and designing them for intermediate control laws. Backstepping achieves the goals of stabilization and tracking, which is crucial for the two degrees of freedom parallel robot described in this thesis. In order to achieve the desired results, a Lyapunov function is constructed for the entire system including the parameter estimates. Let this function candidate be:

$$V_1 = 0.5x_1^2 + 0.5x_2^2 \tag{3.5}$$

Differentiate $V_1$ to get:

$$\begin{aligned} \dot{V}_1 &= x_1x_3 + x_2x_4 = x_1(x_3 - \alpha_1 + \alpha_1) + x_2(x_4 - \alpha_2 + \alpha_2) \\ &= -c_1x_1^2 - c_2x_2^2 + x_1(x_3 - \alpha_1) + x_2(x_4 - \alpha_2) \end{aligned} \tag{3.6}$$

with $\alpha_1 = -c_1x_1$ and $\alpha_2 = -c_2x_2$ called virtual controllers [76], where $c_1$ and $c_2$ are positive constants.

Now it is prudent to implement the second Lyapunov function candidate as:

$$V_2 = V_1 + 0.5 \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} D(q') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \tag{3.7}$$

The derivative of $V_2$ with respect to time yields:

$$\begin{aligned}\dot{V}_2 &= \dot{V}_1 + \begin{bmatrix} (x_3-\alpha_1) & (x_4-\alpha_2) \end{bmatrix} D(q') \begin{bmatrix} (\dot{x}_3-\dot{\alpha}_1) \\ (\dot{x}_4-\dot{\alpha}_2) \end{bmatrix} \\ &\quad +0.5 \begin{bmatrix} (x_3-\alpha_1) & (x_4-\alpha_2) \end{bmatrix} \dot{D}(q') \begin{bmatrix} (x_3-\alpha_1) \\ (x_4-\alpha_2) \end{bmatrix} \end{aligned} \qquad (3.8)$$

As previously stated, $\dot{D}(q') - 2C(q', \dot{q}')$ is skew symmetric [14]; hence:

$$\begin{aligned} &0.5 \begin{bmatrix} (x_3-\alpha_1) & (x_4-\alpha_2) \end{bmatrix} \dot{D}(q') \begin{bmatrix} (x_3-\alpha_1) \\ (x_4-\alpha_2) \end{bmatrix} \\ &= \begin{bmatrix} (x_3-\alpha_1) & (x_4-\alpha_2) \end{bmatrix} C(q', \dot{q}') \begin{bmatrix} (x_3-\alpha_1) \\ (x_4-\alpha_2) \end{bmatrix} \end{aligned} \qquad (3.9)$$

Therefore, by substituting equations (3.3), (3.6) and (3.9) into equation (3.8), it is possible to achieve:

$$\begin{aligned}\dot{V}_2 &= -c_1x_1^2 - c_2x_2^2 + \begin{bmatrix} (x_3-\alpha_1) & (x_4-\alpha_2) \end{bmatrix} \\ &\quad \Bigg( -u + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D(q') \begin{bmatrix} (\ddot{q}_{1d}-\dot{\alpha}_1) \\ (\ddot{q}_{2d}-\dot{\alpha}_2) \end{bmatrix} \\ &\quad +C(q', \dot{q}') \begin{bmatrix} (\dot{q}_{1d}-\alpha_1) \\ (\dot{q}_{2d}-\alpha_2) \end{bmatrix} + g(q') \Bigg) \end{aligned} \qquad (3.10)$$

The derivatives of the virtual controllers are defined as: $\dot{\alpha}_1 = -c_1x_3$ and $\dot{\alpha}_2 = -c_2x_4$. Now, the controller can be determined. The control effort must satisfy the condition of convergence and it must ensure that the output response is stable. The following controller was chosen to accomplish these requirements:

$$\begin{aligned} u &= \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} c_3(x_3-\alpha_1) \\ c_4(x_4-\alpha_2) \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &\quad +D(q') \begin{bmatrix} (\ddot{q}_{1d}-\dot{\alpha}_1) \\ (\ddot{q}_{2d}-\dot{\alpha}_2) \end{bmatrix} + C(q', \dot{q}') \begin{bmatrix} (\dot{q}_{1d}-\alpha_1) \\ (\dot{q}_{2d}-\alpha_2) \end{bmatrix} + g(q') \end{aligned} \qquad (3.11)$$

By substituting equation (3.11) into equation (3.10), the following result will occur:

$$\dot{V}_2 = -c_1 x_1^2 - c_2 x_2^2 - c_3(x_3 - \alpha_1)^2 - c_4(x_4 - \alpha_2)^2 \tag{3.12}$$

This equation states that the function of $\dot{V}_2$ is negative semi-definite. Therefore, the corresponding closed loop system is stable. It should be noted that $c_3$ and $c_4$ represent positive constant gains.

## 3.3 Adaptive Backstepping Controller Design

The adaptive method concerns solving the output response of the controller using estimated system parameters instead of constant parameters. The estimation of the system parameters is a much more precise method than the utilization of constant parameters mainly due to the fact that any system in motion will be subjected to variations in the inertial and gravitational forces along with the changes in the location of the centre of mass. These forces and the centre of mass can be calculated initially, but over time it is increasingly more difficult to justify that these parameters are continuously correct. These slight discrepancies could potentially introduce a significant amount of error in the system, which could play a large role when the precision of the end effector is of paramount importance. This technique is described in detail in [2]. In order to determine whether adaptive controllers produce more impressive results than their non-adaptive counterparts, a Lyapunov function must be chosen to ensure the stability of the closed loop system. Let this candidate function be:

$$\begin{aligned} V_2 \;=\; & V_1 + 0.5 \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} D(q') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \\ & + 0.5 \left(\Theta - \hat{\Theta}\right)^T \Gamma \left(\Theta - \hat{\Theta}\right) \end{aligned} \tag{3.13}$$

where:

$$\Gamma = \begin{bmatrix} \gamma_1 & 0 & \cdots & 0 \\ 0 & \gamma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_{10} \end{bmatrix} \tag{3.14}$$

$$\Theta = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_{10} \end{bmatrix}^T \tag{3.15}$$

with:

$\theta_1 = m_1 l_1^2 + m_3 a_1^2 + I_1$
$\theta_2 = m_2 l_2^2 + m_4 a_2^2 + I_2$
$\theta_3 = m_3 l_3^2 + I_3$
$\theta_4 = m_4 l_4^2 + I_4$
$\theta_5 = m_3 a_1 l_3$
$\theta_6 = m_4 a_2 l_4$
$\theta_7 = g(m_1 l_1 + m_3 a_1)$
$\theta_8 = g(m_2 l_2 + m_4 a_2)$
$\theta_9 = m_3 l_3 g$
$\theta_{10} = m_4 l_4 g$

It should be noted that: $\Gamma$ is a positive definite matrix, $\hat{\Theta}$ is the estimation of $\Theta$ which defines the constant system parameters, $D(q')$ is a positive definite matrix defined in equation (2.7), $V_1$ is a Lyapunov function candidate defined in equation (3.5) and $\alpha_1 = -c_1 x_1$ and $\alpha_2 = -c_2 x_2$ are the virtual controllers with $c_1$ and $c_2$ being positive constants.

The time derivative of $V_2$ becomes:

$$\begin{aligned} \dot{V}_2 &= \dot{V}_1 + \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} D(q') \begin{bmatrix} (\dot{x}_3 - \dot{\alpha}_1) \\ (\dot{x}_4 - \dot{\alpha}_2) \end{bmatrix} \\ &\quad + 0.5 \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \dot{D}(q') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \\ &\quad - \dot{\hat{\Theta}}^T \Gamma \left( \Theta - \hat{\Theta} \right) \end{aligned} \tag{3.16}$$

The derivatives of the virtual controllers are defined as: $\dot{\alpha}_1 = -c_1 x_3$ and $\dot{\alpha}_2 = -c_2 x_4$.

By substituting equations (3.3), (3.6) and (3.9) into (3.16), it is possible to achieve:

$$\begin{aligned}\dot{V}_2 &= -c_1x_1^2 - c_2x_2^2 + \begin{bmatrix} (x_3-\alpha_1) & (x_4-\alpha_2) \end{bmatrix} \\ &\left( \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + D(q') \begin{bmatrix} (\ddot{q}_{1d}-\dot{\alpha}_1) \\ (\ddot{q}_{2d}-\dot{\alpha}_2) \end{bmatrix} + C(q',\dot{q}') \begin{bmatrix} (\dot{q}_{1d}-\alpha_1) \\ (\dot{q}_{2d}-\alpha_2) \end{bmatrix} \right. \\ &\left. + g(q') - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) - \dot{\hat{\Theta}}^T\Gamma\left(\Theta-\hat{\Theta}\right) \end{aligned} \quad (3.17)$$

$D(q')$, $C(q',\dot{q}')$ and $g(q')$ are comprised of many parameters that can be extracted in order to simpify the expression. Therefore, let:

$$\Xi = D(q') \begin{bmatrix} (\ddot{q}_{1d}-\dot{\alpha}_1) \\ (\ddot{q}_{2d}-\dot{\alpha}_2) \end{bmatrix} + C(q',\dot{q}') \begin{bmatrix} (\dot{q}_{1d}-\alpha_1) \\ (\dot{q}_{2d}-\alpha_2) \end{bmatrix} + g(q') = \Xi_o\Theta \quad (3.18)$$

where:

$$\Xi_o = \begin{bmatrix} D_{o11}(\ddot{q}_{1d}-\dot{\alpha}_1) + D_{o12}(\ddot{q}_{2d}-\dot{\alpha}_2) + C_{o11}(\dot{q}_{1d}-\alpha_1) + C_{o12}(\dot{q}_{2d}-\alpha_2) + g_{o1} \\ D_{o21}(\ddot{q}_{1d}-\dot{\alpha}_1) + D_{o22}(\ddot{q}_{2d}-\dot{\alpha}_2) + C_{o21}(\dot{q}_{1d}-\alpha_1) + C_{o22}(\dot{q}_{2d}-\alpha_2) + g_{o2} \end{bmatrix} \quad (3.19)$$

with:

$$\begin{aligned}
D_{o11} &= \begin{bmatrix} 1 & 0 & (1+\rho_{31})^2 & \rho_{41}^2 & 2(1+\rho_{31})\cos(q_3) & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
D_{o12} &= \begin{bmatrix} 0 & 0 & \rho_{32}(1+\rho_{31}) & \rho_{41}(1+\rho_{42}) & \rho_{32}\cos(q_3) & \rho_{41}\cos(q_4) & 0 & 0 & 0 & 0 \end{bmatrix} \\
D_{o21} &= \begin{bmatrix} 0 & 0 & \rho_{32}(1+\rho_{31}) & \rho_{41}(1+\rho_{42}) & \rho_{32}\cos(q_3) & \rho_{41}\cos(q_4) & 0 & 0 & 0 & 0 \end{bmatrix} \\
D_{o22} &= \begin{bmatrix} 0 & 1 & \rho_{32}^2 & (1+\rho_{42})^2 & 0 & 2(1+\rho_{42})\cos(q_4) & 0 & 0 & 0 & 0 \end{bmatrix} \\
C_{o11} &= [\ 0 \quad 0 \quad \dot{\rho}_{31}(1+\rho_{31}) \quad \rho_{41}\dot{\rho}_{41} \quad \dot{\rho}_{31}\cos(q_3) - \dot{q}_3(1+\rho_{31})\sin(q_3) \\
&\quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ] \\
C_{o12} &= [\ 0 \quad 0 \quad \dot{\rho}_{32}(1+\rho_{31}) \quad \rho_{41}\dot{\rho}_{42} \quad \dot{\rho}_{32}\cos(q_3) - (\dot{q}_1+\dot{q}_3)\rho_{32}\sin(q_3) \\
&\quad \dot{q}_2\rho_{41}\sin(q_4) \quad 0 \quad 0 \quad 0 \quad 0\ ] \\
C_{o21} &= [\ 0 \quad 0 \quad \rho_{32}\dot{\rho}_{31} \quad \dot{\rho}_{41}(1+\rho_{42}) \quad \dot{q}_1\rho_{32}\sin(q_3) \\
&\quad \dot{\rho}_{41}\cos(q_4) - (\dot{q}_2+\dot{q}_4)\rho_{41}\sin(q_4) \quad 0 \quad 0 \quad 0 \quad 0\ ] \\
C_{o22} &= [\ 0 \quad 0 \quad \rho_{32}\dot{\rho}_{32} \quad \dot{\rho}_{42}(1+\rho_{42}) \quad 0 \quad \dot{\rho}_{42}\cos(q_4) - \dot{q}_4(1+\rho_{42})\sin(q_4) \\
&\quad 0 \quad 0 \quad 0 \quad 0\ ] \\
g_{o1} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \cos(q_1) & 0 & (1+\rho_{31})\cos(q_1+q_3) & \rho_{41}\cos(q_2+q_4) \end{bmatrix} \\
g_{o2} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cos(q_2) & \rho_{32}\cos(q_1+q_3) & (1+\rho_{42})\cos(q_2+q_4) \end{bmatrix}
\end{aligned}$$

Replacing the parameters in equation (3.17) with the procedure detailed in equation (3.18) yields:

$$\dot{V}_2 = -c_1x_1^2 - c_2x_2^2 + \left[\begin{array}{cc} (x_3-\alpha_1) & (x_4-\alpha_2) \end{array}\right]\left(\left[\begin{array}{c} x_1 \\ x_2 \end{array}\right] + \Xi - \left[\begin{array}{c} u_1 \\ u_2 \end{array}\right]\right) - \dot{\hat{\Theta}}^T\Gamma\left(\Theta - \hat{\Theta}\right) \tag{3.20}$$

Now it is possible to determine the controller equation. Let the controller be:

$$\left[\begin{array}{c} u_1 \\ u_2 \end{array}\right] = \left[\begin{array}{c} c_3(x_3-\alpha_1) \\ c_4(x_4-\alpha_2) \end{array}\right] + \left[\begin{array}{c} x_1 \\ x_2 \end{array}\right] + \Xi_o\hat{\Theta} \tag{3.21}$$

In substituting the newly defined controller into equation (3.20), the resultant will become:

$$\begin{aligned} \dot{V}_2 &= -c_1x_1^2 - c_2x_2^2 - c_3(x_3-\alpha_1)^2 - c_4(x_4-\alpha_2)^2 \\ &\quad + \left[\begin{array}{cc} (x_3-\alpha_1) & (x_4-\alpha_2) \end{array}\right]\Xi_o\left(\Theta - \hat{\Theta}\right) - \dot{\hat{\Theta}}^T\Gamma\left(\Theta - \hat{\Theta}\right) \end{aligned} \tag{3.22}$$

Let the unknown parameters updating law be:

$$\dot{\hat{\Theta}} = \Gamma^{-1}\Xi_o^T\left[\begin{array}{c} (x_3-\alpha_1) \\ (x_4-\alpha_2) \end{array}\right] \tag{3.23}$$

With the substitution of equation (3.23) into equation (3.22) the final result will yield:

$$\dot{V}_2 = -c_1x_1^2 - c_2x_2^2 - c_3(x_3-\alpha_1)^2 - c_4(x_4-\alpha_2)^2 \tag{3.24}$$

This equation states that the function of $\dot{V}_2$ is negative semi-definite. Therefore, the corresponding closed loop system is stable. It should be noted that $c_3$ and $c_4$ represent positive constant gains.

## 3.4 Non-Adaptive PD Controller Design

The PD controller is the short form notation of the proportional derivative controller. The proportional term is utilized to compare the actual trajectory of the end effector with the desired trajectory, which is also known as the error. The derivative term is the attempt to see how far a process variable has been from the desired trajectory in the past and anticipating where the trajectory will need to be in the future, which is also known as the change in error. This technique was first developed in 1911 by the American inventor Elmer Sperry for automatic ship steering [38]. The control law that is now commonly associated with PD controllers was created by Nicholas Minorsky in 1922, which ironically was also used for automatic ship steering . In 1923, he successfully tested the automatic steering gear on the American battleship USS New Mexico [4]. Mr. Minorsky's control technique has been so successful that it is the most widely used controller in industrial applications. The simplicity of the design, yet the robustness of the control scheme have been the major reasons for its longevity. The non-adaptive PD controller will be implemented similarly to the non-adaptive backstepping technique. That is, a Lyapunov function is necessary in order to achieve the desired results. Let this function candidate be:

$$\begin{aligned} V &= 0.5 \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &\quad +0.5 \begin{bmatrix} x_3 & x_4 \end{bmatrix} D(q') \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \end{aligned} \tag{3.25}$$

It should be noted that $K_{P1}$ and $K_{P2}$ represent the proportional gains of motors one and two, respectively, while $D(q')$ is a positive definite matrix defined in equation (2.7).

The derivative of $V$ with respect to time becomes:

$$\begin{aligned}\dot{V} &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \\ &+0.5 \begin{bmatrix} x_3 & x_4 \end{bmatrix} \dot{D}(q') \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} x_3 & x_4 \end{bmatrix} D(q') \begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} \end{aligned} \quad (3.26)$$

As previously stated, $\dot{D}(q') - 2C(q', \dot{q}')$ is skew symmetric [14]; hence:

$$0.5 \begin{bmatrix} x_3 & x_4 \end{bmatrix} \dot{D}(q') \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_3 & x_4 \end{bmatrix} C(q', \dot{q}') \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \quad (3.27)$$

Therefore, by substituting equations (3.3) and (3.27) into equation (3.26), it is possible to achieve:

$$\begin{aligned}\dot{V} &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \\ &+ \begin{bmatrix} x_3 & x_4 \end{bmatrix} \left( D(q') \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + C(q', \dot{q}') \begin{bmatrix} \dot{q}_{1d} \\ \dot{q}_{2d} \end{bmatrix} + g(q') - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \end{aligned} \quad (3.28)$$

With all the appropriate data defined, it is now possible to determine the equation for the controller. The control effort must satisfy the condition of convergence and it must ensure that the output response is stable. The following controller was chosen to accomplish these requirements:

$$\begin{aligned}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} &= \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} K_{D1} & 0 \\ 0 & K_{D2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \\ &+D(q') \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + C(q', \dot{q}') \begin{bmatrix} \dot{q}_{1d} \\ \dot{q}_{2d} \end{bmatrix} + g(q') \end{aligned} \quad (3.29)$$

It should be noted that $K_{D1}$ and $K_{D2}$ represent the derivative gains of motors one and two, respectively.

By substituting equation (3.29) into equation (3.28), the following result will occur:

$$\dot{V} = -K_{D1}x_3^2 - K_{D2}x_4^2 \tag{3.30}$$

This equation states that the function of $\dot{V}$ is negative semi-definite. Therefore, the corresponding closed loop system is stable.

## 3.5 Adaptive PD Controller Design

The adaptive PD controller will be implemented similarly to the adaptive back-stepping technique. That is, a Lyapunov function is necessary in order to achieve the desired results. Let this function candidate be:

$$\begin{aligned} V &= 0.5 \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &\quad + 0.5 \begin{bmatrix} x_3 & x_4 \end{bmatrix} D(q') \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + 0.5 \left(\Theta - \hat{\Theta}\right)^T \Gamma \left(\Theta - \hat{\Theta}\right) \end{aligned} \tag{3.31}$$

It should be noted that: $K_{P1}$ and $K_{P2}$ represent the proportional gains of motors one and two, respectively, $\Gamma$ is a positive definite matrix defined in equation (3.14), $\hat{\Theta}$ is the estimation of $\Theta$ which states the constant system parameters defined in equation (3.15) and $D(q')$ is a positive definite matrix defined in equation (2.7).

The time derivative of $V$ becomes:

$$\begin{aligned} \dot{V} &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + 0.5 \begin{bmatrix} x_3 & x_4 \end{bmatrix} \dot{D}(q') \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \\ &\quad + \begin{bmatrix} x_3 & x_4 \end{bmatrix} D(q') \begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} - \dot{\hat{\Theta}}^T \Gamma \left(\Theta - \hat{\Theta}\right) \end{aligned} \tag{3.32}$$

By substituting equations (3.3) and (3.27) into (3.32), it is possible to achieve:

$$\begin{aligned}\dot{V} &= \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} - \dot{\hat{\Theta}}^T \Gamma \left( \Theta - \hat{\Theta} \right) \\ &\quad + \begin{bmatrix} x_3 & x_4 \end{bmatrix} \left( D(q') \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + C(q', \dot{q}') \begin{bmatrix} \dot{q}_{1d} \\ \dot{q}_{2d} \end{bmatrix} + g(q') - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \end{aligned} \tag{3.33}$$

$D(q')$, $C(q', \dot{q}')$ and $g(q')$ are comprised of many parameters that can be extracted in order to simpify the expression. Therefore, let:

$$\Xi = D(q') \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + C(q', \dot{q}') \begin{bmatrix} \dot{q}_{1d} \\ \dot{q}_{2d} \end{bmatrix} + g(q') = \Xi_o \Theta \tag{3.34}$$

where:

$$\Xi_o = \begin{bmatrix} D_{o11}\ddot{q}_{1d} + D_{o12}\ddot{q}_{2d} + C_{o11}\dot{q}_{1d} + C_{o12}\dot{q}_{2d} + g_{o1} \\ D_{o21}\ddot{q}_{1d} + D_{o22}\ddot{q}_{2d} + C_{o21}\dot{q}_{1d} + C_{o22}\dot{q}_{2d} + g_{o2} \end{bmatrix} \tag{3.35}$$

The sub-parameters of $D_{o11}$, $D_{o12}$, $D_{o21}$, $D_{o22}$, $C_{o11}$, $C_{o12}$, $C_{o21}$, $C_{o22}$, $g_{o1}$ and $g_{o2}$ are defined in equation (3.19). Replacing the parameters in equation (3.33) with the procedure detailed in equation (3.34) yields:

$$\dot{V} = \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} x_3 & x_4 \end{bmatrix} \left( \Xi - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) - \dot{\hat{\Theta}}^T \Gamma \left( \Theta - \hat{\Theta} \right) \tag{3.36}$$

Now it is possible to determine the controller equation. Let the controller be:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} K_{P1} & 0 \\ 0 & K_{P2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} K_{D1} & 0 \\ 0 & K_{D2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} + \Xi_o \hat{\Theta} \tag{3.37}$$

It should be noted that $K_{D1}$ and $K_{D2}$ represent the derivative gains of motors one and two, respectively.

In substituting the newly defined controller into equation (3.36), the resultant will become:

$$\dot{V} = \begin{bmatrix} x_3 & x_4 \end{bmatrix} \Xi_o \left(\Theta - \hat{\Theta}\right) - \begin{bmatrix} x_3 & x_4 \end{bmatrix} \begin{bmatrix} K_{D1} & 0 \\ 0 & K_{D2} \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} - \dot{\hat{\Theta}}^T \Gamma \left(\Theta - \hat{\Theta}\right) \tag{3.38}$$

Let the unknown parameters updating law be:

$$\dot{\hat{\Theta}} = \Gamma^{-1} \Xi_o^T \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} \tag{3.39}$$

With the substitution of equation (3.39) into equation (3.38) the final result will yield:

$$\dot{V} = -K_{D1} x_3^2 - K_{D2} x_4^2 \tag{3.40}$$

This equation states that the function of $\dot{V}$ is negative semi-definite. Therefore, the corresponding closed loop system is stable.

This concludes the derivation of the four non-fuzzy logic controllers. All the aforementioned controllers have been proven to make $\dot{V}_2$ or $\dot{V}$ negative semi-definite; hence the closed loop systems are stable. These derivations ensure that the controllers will be reproduced in simulation and experimentally. The following chapter pertains to the derivation of the four fuzzy logic controllers.

# Chapter 4

# Fuzzy Logic Controller Design

This chapter will describe the derivation of the PD, indirect adaptive and direct adaptive fuzzy logic controllers along with a fuzzy adaptive backstepping controller. A thorough mathematical analysis will be presented concerning the generation of the controller equation and the stability of the closed loop system will be justified. The adaptive controllers will encompass an estimator equation that is used to estimate the parameters of the system, while specific fuzzy systems will be introduced to solve the fuzzy logic problem.

## 4.1 Controller Foundation

The controller foundation will build on the knowledge previously explained in section 3.1 by the addition of fuzzy logic. Fuzzy logic is a concept first described by Lotfi A. Zadeh in 1965 [74]. It pertains to the implementation of human knowledge to adequately simulate the output response of a system. The main purpose of a fuzzy system is to map an input space to an output space. An input space is commonly referred to as the universe of discourse. The universe of discourse defines all the possible input data that can be utilized in the fuzzy logic system, whether the data is vague, imprecise or accurate. The imprecision found in the universe of discourse is known as the linguistic variable [51]. The output space is simply the response generated from the fuzzy system. It is important to realize that there is a trade-off between significance and precision when employing a fuzzy system. Depending on the importance of the input data being sent into a fuzzy system, it may be more beneficial to achieve a precise result. This desired result would stipulate for more membership functions in order to ensure the most accurate output at a cost of a higher computational time. A system that requires a faster response time would desire a more significant result; hence the number of membership functions would be smaller and yield a less precise outcome. A membership function is a curve that defines how each point in the input space is mapped to the degree of membership between zero and one. Unlike non-fuzzy logic controllers which yield crisp results, variables in the universe of discourse can achieve membership in a wide array of membership functions. It depends solely on the number and range of membership functions defined. There are five common membership function shapes namely: triangular, trapezoidal, sigmoidal, Gaussian and S or Z shaped. Any combination of the aforementioned shapes can be implemented in any order; it is strictly up to the designer. All the fuzzy logic controllers will utilize eleven membership functions for both the error and the change in error, that is $x_1$, $x_2$ and $x_3$, $x_4$, respectively, along with the controller output. Nine of the membership functions will be of triangular shape, while the remaining two will be of Z and S shape. Figure 5.1 portrays the membership functions utilized for the error, while the same structure with different values for the centre of each membership function are used for the remaining two.

Now that many of the terms pertaining to fuzzy logic have been depicted, the fuzzy system will be discussed. Figure 5.2 represents the general structure of a fuzzy system. There are four crucial components that comprise any fuzzy system: the fuzzifier, the fuzzy rule base, the fuzzy inference engine, and the defuzzifier.

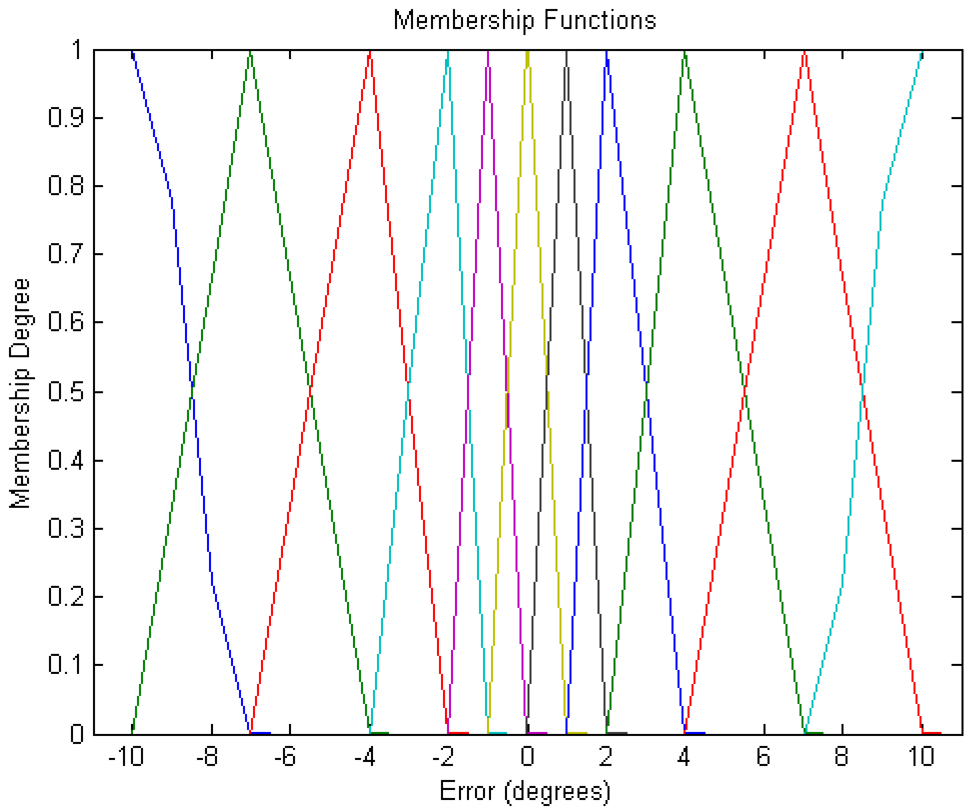


Figure 4.1: Membership Functions for Error

The centre of each of the eleven membership functions from left to right is defined as: [-10, -7, -4, -2, -1, 0, 1, 2, 4, 7, 10].

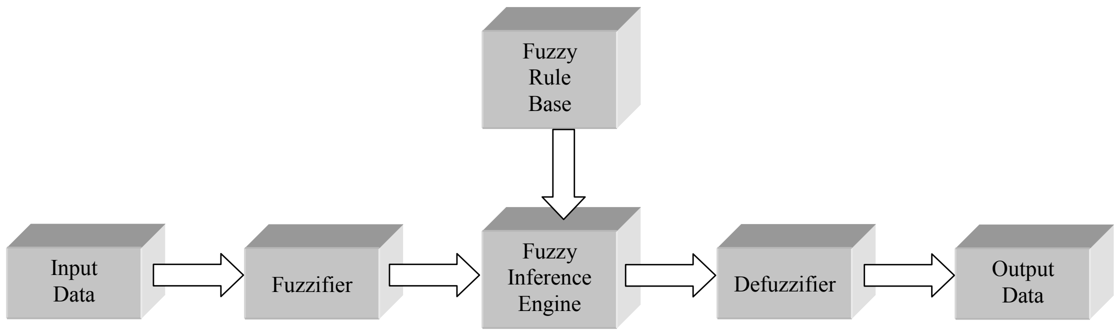


Figure 4.2: Fuzzy System

The fuzzifier is the first step in the fuzzy system process. Its purpose is to convert the inputted data into a fuzzy set. There are multiple techniques that are available to achieve this, but the one that will be discussed in this thesis is the singleton fuzzifier. The singleton fuzzifier maps a real valued point $x^* \in U \subset \mathbb{R}^n$ into a fuzzy set $A'$ in $U$, which has a membership value of 1 at $x^*$ and 0 at all other points in $U$. Therefore:

$$\mu_{A'}(x) = \begin{cases} 1 & x = x^* \\ 0 & x \neq x^* \end{cases} \tag{4.1}$$

The middle two blocks are the most important parts of the fuzzy system. The first of which is known as the fuzzy rule base. Fuzzy sets and fuzzy operators are the subjects and verbs of fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic. The general form for the fuzzy rule base is as follows:

$$Rule\ k:\ If\ x_1\ is\ A_1^k\ and\ ...\ and\ x_n\ is\ A_n^k,\ then\ y\ is\ B^k. \tag{4.2}$$

where:

$k$ is the $k^{th}$ rule in the fuzzy rule base;

$A_i^k$, known as the antecedent and $B^k$, known as the consequent, are the fuzzy sets in $U_i \subset \mathbb{R}$ and $V \subset \mathbb{R}$, respectively;

$x = (x_1, x_2, ..., x_n)^T \in U$ and $y \in V$ are the input and output linguistic variables of the fuzzy system, respectively.

There are two vital conditions that must be followed when defining any rule in the fuzzy rule base. The first condition states that a set of fuzzy if-then rules is complete if for any $x \in U$ there exists at least one rule in the fuzzy rule base such that: $\mu_{A_i^k}(x_i) \neq 0$ for all $i = 1, 2, ..., n$. The second condition states that a set of fuzzy if-then rules is consistent if there are no rules with the same $if$ parts, but different *then* parts. The greater the number of rules that are defined in the fuzzy rule base, the more precise will the result generated by the fuzzy system become. Although, the drawbacks of defining too many rules is that the computational time required to generate a response will also increase. The lesser the number of rules, the less precise will the result generated by the fuzzy system become. Although, the advantage of defining a limited number of rules is that the result will become more significant. Therefore, the objective in designing any fuzzy system is to balance the number of rules to achieve a result that is both significant and precise enough to garner the desired output response. The fuzzy systems employed in this thesis all utilize 121 distinct rules.

The other crucial component in the block diagram presented in Figure 5.2 is the fuzzy inference engine. The fuzzy inference engine is the heart of any fuzzy system. It is the complex calculation of an input space to an output space utilizing the rules generated in the fuzzy rule base. There are a multitude of inference methods describing how to solve any fuzzy system. Some of the more popular techniques such as: Mamdani's method, Larsen's method, Tsukamoto's method and Takagi, Sugeno and Kang's method are defined in [28]. These methods are then used to define various fuzzy inference engines, for instance: the product inference engine, the minimum inference engine and the Zadeh inference engine. The fuzzy inference engine that is implemented in this thesis is the product inference engine. The product inference engine uses the algebraic product for the t-norm operator, the maximum for the s-norm operator, Mamdani's product for implication and the individual rule based inference with the union combination. Mathematically, the product inference engine is given by:

$$\mu_{B'}(y) = \max_{k} \left\{ \max_{x \in U} \left[ \mu_{A'}\left(x_1, \ldots, x_n\right) \prod_{i=1}^{n} \mu_{A_i^k}(x_i) \mu_{B^k}(y) \right] \right\} \tag{4.3}$$

In this thesis, the fuzzy set $A'$ is a fuzzy singleton, which is shown in equation (4.1). Therefore, the product inference engine described in equation (4.3) can be simplifed to:

$$\mu_{B'}(y) = \max_{k} \left\{ \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*) \mu_{B^k}(y) \right\} \tag{4.4}$$

The final process of the fuzzy system is the defuzzifier. It converts the fuzzy set $B'$ in $V \subset \mathbb{R}^n$ into a crisp output data point $y^* \in V$. The defuzzifier that will be employed in this thesis is known as the centre average defuzzifier. The formula for the centre average defuzzifier is defined as:

$$y^* = \frac{\sum\limits_{k=1}^{M} \overline{y}^k w_k}{\sum\limits_{k=1}^{M} w_k} \tag{4.5}$$

where: $\overline{y}^k$ is the centre and $w_k$ is the height of the $k^{th}$ fuzzy set; $M$ is the number of fuzzy sets. It should be noted that this method is restricted to symmetrical membership functions.

Now it is possible to develop various fuzzy logic controllers based on a fuzzy system comprised of a singleton fuzzier, the product inference engine and a centre average defuzzifier.

## 4.2 Fuzzy Logic PD Controller Design

The fuzzy logic PD controller will employ a fuzzy system with a singleton fuzzifier, the product inference engine and a centre average defuzzifier. The formulas of this fuzzy system are found in equations (4.1), (4.4) and (4.5), respectively.

It can be proven that: $\mu_{B^{k'}}(y) = \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*)\mu_{B^k}(y)$ has the same centre as $B^k$, which is $\overline{y}^k$. It can also be shown that the height of $\max_y \left\{ \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*)\mu_{B^k}(y) \right\} = \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*)\mu_{B^k}(\overline{y}^k) = \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*)$ because $B^k$ is a normal fuzzy set and $\mu_{B^k}(\overline{y}^k) = 1$. Therefore, after combining these formulas together, the final result will yield:

$$y^* = \frac{\sum_{k=1}^{M} \overline{y}^k \left( \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*) \right)}{\sum_{k=1}^{M} \left( \prod_{i=1}^{n} \mu_{A_i^k}(x_i^*) \right)} \tag{4.6}$$

As shown in Figure 4.1, there are eleven membership functions that are utilized for both the error and change in error generated by the system when solving for equations (3.1), (3.2) and (3.3). This means that there are 121 distinct rules that must be defined for the fuzzy system. Therefore, the centre average defuzzifier in equation (4.6) can be rewritten to represent the planar two degrees of freedom parallel robot as:

$$y_j^* = \frac{\sum_{k=1}^{121} \overline{y}_j^k \left( \mu_{A_{x_j}^k}(x_j^*)\mu_{A_{x_{j+2}}^k}(x_{j+2}^*) \right)}{\sum_{k=1}^{121} \left( \mu_{A_{x_j}^k}(x_j^*)\mu_{A_{x_{j+2}}^k}(x_{j+2}^*) \right)} \tag{4.7}$$

It should be noted that $j$ is either one or two, which represents the terms attached with motor one or motor two, respectively, and $\overline{y}^k$ is the control surface defined in Table 4.1. The final procedure concerning this defuzzified variable is its multiplication by the scaling factor ($c_{out}$) in order for the solution to be large enough that it can be implemented as the controller output ($u$). In order to tune the fuzzy PD controller more effectively, the scaling factors ($c_e$) and ($c_{\dot{e}}$) were introduced for the error and change in error, respectively.

| $\boldsymbol{x_j/x_{j+2}}$ | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 1 | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 |
| **2** | 1 | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 7 |
| **3** | 2 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 7 |
| **4** | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 7 | 8 |
| **5** | 3 | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 7 | 8 | 8 |
| **6** | 3 | 4 | 4 | 5 | 5 | 6 | 7 | 7 | 8 | 8 | 9 |
| **7** | 4 | 4 | 5 | 5 | 6 | 7 | 7 | 8 | 8 | 9 | 9 |
| **8** | 4 | 5 | 5 | 6 | 7 | 7 | 8 | 8 | 9 | 9 | 10 |
| **9** | 5 | 5 | 6 | 7 | 7 | 8 | 8 | 9 | 9 | 10 | 10 |
| **10** | 5 | 6 | 7 | 7 | 8 | 8 | 9 | 9 | 10 | 10 | 11 |
| **11** | 6 | 7 | 7 | 8 | 8 | 9 | 9 | 10 | 10 | 11 | 11 |

Table 4.1: Control Surface

The numbers in bold for error and change in error represent the membership function number from left to right defined in Figure 5.1. The numbers that form the eleven by eleven matrix represent the control surface for all 121 possible rules in this fuzzy system. The values for these numbers are the centre of each membership function from left to right, which is also defined in Figure 5.1.

## 4.3 Indirect Adaptive Fuzzy Logic Controller Design

Indirect adaptive fuzzy logic control is defined as a fuzzy controller that comprises of a number of fuzzy systems that are initially constructed from the plant knowledge. The plant is simply the system that is being studied, which in this case is the planar two degrees of freedom parallel robot. Mr. Tong described in [59] that an unknown nonlinear first order system can successfully become semi-globally uniformly bounded using an indirect adaptive fuzzy logic approach. Therefore, this section will prove that the second order nonlinear parallel robot system described in this thesis can achieve stability based on the derivation method presented in [64].

The parallel robot model described in equations (3.1), (3.2) and (3.3) can be written as:

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} &= \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + \begin{bmatrix} F_1(X) \\ F_2(X) \end{bmatrix} + \begin{bmatrix} G_{11}(X) & G_{12}(X) \\ G_{21}(X) & G_{22}(X) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \end{aligned} \tag{4.8}$$

where:

$$\begin{bmatrix} F_1(X) \\ \\ F_2(X) \end{bmatrix} = D^{-1}(q') \left( C(q', \dot{q}') \begin{bmatrix} \dot{q}_1 \\ \\ \dot{q}_2 \end{bmatrix} + g(q') \right)$$

$$\begin{bmatrix} G_{11}(X) & G_{12}(X) \\ \\ G_{21}(X) & G_{22}(X) \end{bmatrix} = -D^{-1}(q')$$

$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \in \mathbb{R}$ is the state vector of the system that is available for measurement and $\begin{bmatrix} \ddot{q}_{1d} & \ddot{q}_{2d} \end{bmatrix}^T$ consists of the desired angular accelerations.

There are interconnection terms prevailent in various robotic systems which have been analyzed and solved [44]. For the purpose of simplifying the controller design, the interconnection terms $G_{12}$ and $G_{21}$ are neglected. This results in the following system:

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} &= \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + \begin{bmatrix} F_1(X) \\ F_2(X) \end{bmatrix} + \begin{bmatrix} G_{11}(X) & 0 \\ 0 & G_{22}(X) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}
\end{aligned} \tag{4.9}$$

To simplify the derivation, let $G_{11}(X) = G_1(X)$ and $G_{22}(X) = G_2(X)$. In order for equation (4.9) to be controllable, it is required that $G_1(X) \neq 0$ and $G_2(X) \neq 0$. As a matter of fact, the positive definiteness of the $D(q')$ matrix guarantees that the two inequalities will be true: $G_1(X) > 0$ and $G_2(X) > 0$. The control objective is to design a feedback controller $u = u(X \mid \hat{\Theta})$ based on fuzzy systems and an adaptive law for adjusting the parameter vector $(\hat{\Theta})$, such that the actual output follows the desired output. The fuzzy rule base that will describe the input and output behaviour of $F_1(X)$, $F_2(X)$, $G_1(X)$ and $G_2(X)$ is defined in equation (4.2).

If the nonlinear functions $F_1(X)$, $F_2(X)$, $G_1(X)$ and $G_2(X)$ are known, it is possible to choose $(u)$ such that the nonlinearity will cancel out. The advantage of this is the fact that the controller can be designed based on a linear control theory such as pole placement. Therefore, let:

$$K = \begin{bmatrix} k_{11} & 0 & k_{12} & 0 \\ 0 & k_{21} & 0 & k_{22} \end{bmatrix}$$

represent all the coefficients of the stable polynomial $s^2 + k_{j2}s + k_{j1}$ with $k_{11}$, $k_{12}$, $k_{21}$ and $k_{22}$ being positive constants; $j$ is either one or two, which represents the terms attached with motor one and motor two, respectively.

The control law can now be defined as:

$$\begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix} = \begin{bmatrix} \frac{1}{G_1(X)} & 0 \\ 0 & \frac{1}{G_2(X)} \end{bmatrix} \left( \begin{bmatrix} -F_1(X) - \ddot{q}_{1d} \\ -F_2(X) - \ddot{q}_{2d} \end{bmatrix} - KX \right) \tag{4.10}$$

Substituting equation (4.10) into equation (4.9) yields a closed loop system governed by:

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \dot{x}_3 &= -k_{11}x_1 - k_{12}x_3 \\ \dot{x}_4 &= -k_{21}x_2 - k_{22}x_4 \end{aligned} \tag{4.11}$$

Unfortunately, $F_1(X)$, $F_2(X)$, $G_1(X)$ and $G_2(X)$ are unknown, hence the ideal controller defined in equation (4.10) cannot be implemented. However, the $F_1(X)$, $F_2(X)$, $G_1(X)$ and $G_2(X)$ functions can be replaced by the fuzzy systems $\hat{F}_1(X)$, $\hat{F}_2(X)$, $\hat{G}_1(X)$ and $\hat{G}_2(X)$, which are constructed based on the fuzzy rule base defined in equation (4.2). To improve the approximation accuracy of $\hat{F}_1(X)$, $\hat{F}_2(X)$, $\hat{G}_1(X)$ and $\hat{G}_2(X)$, it is beneficial to leave some parameters in $\hat{F}_1(X)$, $\hat{F}_2(X)$, $\hat{G}_1(X)$ and $\hat{G}_2(X)$ free in order for them to change during the online operation of the system. Let $\hat{\Theta}_{F_1} \in \mathbb{R}^{M_{F_1}}$, $\hat{\Theta}_{F_2} \in \mathbb{R}^{M_{F_2}}$, $\hat{\Theta}_{G_1} \in \mathbb{R}^{M_{G_1}}$ and $\hat{\Theta}_{G_2} \in \mathbb{R}^{M_{G_2}}$ be the free parameters in $\hat{F}_1(X)$, $\hat{F}_2(X)$, $\hat{G}_1(X)$ and $\hat{G}_2(X)$, respectively. Now it is possible to denote the functions as: $\hat{F}_1(X) = \hat{F}_1(X \mid \hat{\Theta}_{F_1})$, $\hat{F}_2(X) = \hat{F}_2(X \mid \hat{\Theta}_{F_2})$, $\hat{G}_1(X) = \hat{G}_1(X \mid \hat{\Theta}_{G_1})$ and $\hat{G}_2(X) = \hat{G}_2(X \mid \hat{\Theta}_{G_2})$. Therefore, equation (4.10) can be rewritten as the fuzzy controller:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} u_{1d} \\ u_{2d} \end{bmatrix} = \begin{bmatrix} \frac{1}{\hat{G}_1(X|\hat{\Theta}_{G_1})} & 0 \\ 0 & \frac{1}{\hat{G}_2(X|\hat{\Theta}_{G_2})} \end{bmatrix} \left( \begin{bmatrix} -\hat{F}_1(X \mid \hat{\Theta}_{F_1}) - \ddot{q}_{1d} \\ -\hat{F}_2(X \mid \hat{\Theta}_{F_2}) - \ddot{q}_{2d} \end{bmatrix} - KX \right) \tag{4.12}$$

where $u_{1d}$ and $u_{2d}$ are the ideal controllers of $u_1^*$ and $u_2^*$, respectively.

To implement the controller defined in equation (4.12), the fuzzy functions of $\hat{F}_1(X \mid \hat{\Theta}_{F_1})$, $\hat{F}_2(X \mid \hat{\Theta}_{F_2})$, $\hat{G}_1(X \mid \hat{\Theta}_{G_1})$ and $\hat{G}_2(X \mid \hat{\Theta}_{G_2})$ must be described in detail. By using the singleton fuzzifier, the product inference engine and the centre average defuzzifier defined in equations (4.1), (4.4) and (4.5), respectively, it is possible to obtain:

$$\hat{F}_j(X \mid \hat{\Theta}_{F_j}) = \frac{\sum_{k=1}^{M_{F_j}} \overline{y}^k_{F_j} \left( \mu_{A^k_{x_j}}(x^*_j) \mu_{A^k_{x_{j+2}}}(x^*_{j+2}) \right)}{\sum_{k=1}^{M_{F_j}} \left( \mu_{A^k_{x_j}}(x^*_j) \mu_{A^k_{x_{j+2}}}(x^*_{j+2}) \right)} \tag{4.13}$$

$$\hat{G}_j(X \mid \hat{\Theta}_{G_j}) = \frac{\sum_{k=1}^{M_{G_j}} \overline{y}^k_{G_j} \left( \mu_{E^k_{x_j}}(x^*_j) \mu_{E^k_{x_{j+2}}}(x^*_{j+2}) \right)}{\sum_{k=1}^{M_{G_j}} \left( \mu_{E^k_{x_j}}(x^*_j) \mu_{E^k_{x_{j+2}}}(x^*_{j+2}) \right)} \tag{4.14}$$

where: $M_{F_j}$ or $M_{G_j}$ is the number of fuzzy rules, which is the product of the number of fuzzy sets in $x_1$ and $x_3$ or $x_2$ and $x_4$, respectively; $\mu_{A^k_{x_j}}$ and $\mu_{A^k_{x_{j+2}}}$ or $\mu_{E^k_{x_j}}$ and $\mu_{E^k_{x_{j+2}}}$ are the fuzzy sets for $x_1$ and $x_3$ or $x_2$ and $x_4$, respectively; $\overline{y}^k_{F_j}$ and $\overline{y}^k_{G_j}$ define the centre of the $k^{th}$ fuzzy set; $k$ is the $k^{th}$ rule in the fuzzy rule base.

Let $\overline{y}^k_{F_j}$ and $\overline{y}^k_{G_j}$ be the free parameters that are collected into $\hat{\Theta}_{F_j} \in \mathbb{R}^{\prod_{i=1}^{n} M_{F_j}}$ and $\hat{\Theta}_{G_j} \in \mathbb{R}^{\prod_{i=1}^{n} M_{G_j}}$, respectively, in order to rewrite equations (4.13) and (4.14) as:

$$\hat{F}_j(X \mid \hat{\Theta}_{F_j}) = \hat{\Theta}^T_{F_j} \xi_j(X) \tag{4.15}$$

$$\hat{G}_j(X \mid \hat{\Theta}_{G_j}) = \hat{\Theta}^T_{G_j} \eta_j(X) \tag{4.16}$$

where:

$$\xi_j = \begin{bmatrix} \xi_{j1} & \cdots & \xi_{jM_{F_j}} \end{bmatrix}^T \tag{4.17}$$

$$\eta_j = \begin{bmatrix} \eta_{j1} & \cdots & \eta_{jM_{G_j}} \end{bmatrix}^T \tag{4.18}$$

with:

$$\xi_{jk}(X) = \frac{\mu_{A^k_{x_j}}(x_j^*)\mu_{A^k_{x_{j+2}}}(x_{j+2}^*)}{\sum_{k=1}^{M_{F_j}} \left(\mu_{A^k_{x_j}}(x_j^*)\mu_{A^k_{x_{j+2}}}(x_{j+2}^*)\right)} \tag{4.19}$$

$$\eta_{jk}(X) = \frac{\mu_{E^k_{x_j}}(x_j^*)\mu_{E^k_{x_{j+2}}}(x_{j+2}^*)}{\sum_{k=1}^{M_{G_j}} \left(\mu_{E^k_{x_j}}(x_j^*)\mu_{E^k_{x_{j+2}}}(x_{j+2}^*)\right)} \tag{4.20}$$

The indirect adaptive fuzzy controller will utilize 121 distinct rules due to the number of membership functions defined in Figure 5.1. Therefore, $M_{F_j}$ and $M_{G_j}$, which are found in equations (4.19) and (4.20), respectively, are set to 121 to reflect the fuzzy system employed on the two degrees of freedom parallel robot.

This concludes the design portion of the fuzzy controller. The next task is to design an adaptive law for $\hat{\Theta}_{F_j}$ and $\hat{\Theta}_{G_j}$ such that the tracking error of $x_1$ and $x_2$ is minimized.

Substituting equation (4.12) into (4.9) yields the closed loop dynamics of the fuzzy control system as:

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} &= \begin{bmatrix} F_1(X) - \hat{F}_1(X \mid \hat{\Theta}_{F_1}) \\ F_2(X) - \hat{F}_2(X \mid \hat{\Theta}_{F_2}) \end{bmatrix} + \begin{bmatrix} \left(G_1(X) - \hat{G}_1(X \mid \hat{\Theta}_{G_1})\right) u_{1d} \\ \left(G_2(X) - \hat{G}_2(X \mid \hat{\Theta}_{G_2})\right) u_{2d} \end{bmatrix} - KX \end{aligned} \tag{4.21}$$

Let: $\Xi = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_{11} & 0 & -k_{12} & 0 \\ 0 & -k_{21} & 0 & -k_{22} \end{bmatrix}$, $B_1 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ and $B_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$ with $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$. Therefore, the dynamic equation (4.21) can be rewritten as:

$$\dot{X} = \Xi X + B \left\{ \begin{bmatrix} F_1(X) - \hat{F}_1(X \mid \hat{\Theta}_{F_1}) \\ F_2(X) - \hat{F}_2(X \mid \hat{\Theta}_{F_2}) \end{bmatrix} + \begin{bmatrix} \left(G_1(X) - \hat{G}_1(X \mid \hat{\Theta}_{G_1})\right) u_{1d} \\ \left(G_2(X) - \hat{G}_2(X \mid \hat{\Theta}_{G_2})\right) u_{2d} \end{bmatrix} \right\} \tag{4.22}$$

It should be noted that: $\dot{X} = [\ \dot{x}_1 \quad \dot{x}_2 \quad \dot{x}_3 \quad \dot{x}_4\ ]^T$. Now it is appropriate to define the optimal parameters as:

$$\Theta_{F_j} = \arg \min_{\hat{\Theta}_{F_j} \in \mathbb{R}^{\prod_{i=1}^{n} M_{F_j}}} \left[ \sup_{X \in \mathbb{R}^n} \left| F_j(X) - \hat{F}_j(X \mid \hat{\Theta}_{F_j}) \right| \right] \tag{4.23}$$

$$\Theta_{G_j} = \arg \min_{\hat{\Theta}_{G_j} \in \mathbb{R}^{\prod_{i=1}^{n} M_{G_j}}} \left[ \sup_{X \in \mathbb{R}^n} \left| G_j(X) - \hat{G}_j(X \mid \hat{\Theta}_{G_j}) \right| \right] \tag{4.24}$$

By implementing the fuzzy systems stated in equations (4.13) and (4.14) with approximators in equations (4.23) and (4.24), the minimum approximation error can be defined as:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} F_1(X) - \hat{F}_1(X \mid \Theta_{F_1}) \\ F_2(X) - \hat{F}_2(X \mid \Theta_{F_2}) \end{bmatrix} + \begin{bmatrix} \left( G_1(X) - \hat{G}_1(X \mid \Theta_{G_1}) \right) u_{1d} \\ \left( G_2(X) - \hat{G}_2(X \mid \Theta_{G_2}) \right) u_{2d} \end{bmatrix} \tag{4.25}$$

Therefore, by substituting equations (4.15), (4.16) and (4.25) into equation (4.22), the following closed loop dynamic equation can be realized:

$$\dot{X} = \Xi X + B \begin{bmatrix} \left( \Theta_{F_1} - \hat{\Theta}_{F_1} \right)^T \xi_1(X) + \left( \Theta_{G_1} - \hat{\Theta}_{G_1} \right)^T \eta_1(X) u_{1d} + \omega_1 \\ \left( \Theta_{F_2} - \hat{\Theta}_{F_2} \right)^T \xi_2(X) + \left( \Theta_{G_2} - \hat{\Theta}_{G_2} \right)^T \eta_2(X) u_{2d} + \omega_2 \end{bmatrix} \tag{4.26}$$

The goal of the adaptive law is to determine an adjusting mechanism for $\hat{\Theta}_{F_j}$ and $\hat{\Theta}_{G_j}$ in order for the tracking errors, $x_1$ and $x_2$ and the parameter errors, $(\Theta_{F_j} - \hat{\Theta}_{F_j})$ and $(\Theta_{G_j} - \hat{\Theta}_{G_j})$ to have a diminishing effect on the overall system.

To achieve such an outcome, consider the Lyapunov function candidate:

$$\begin{aligned} V &= 0.5\left(\Theta_{F_1}-\hat{\Theta}_{F_1}\right)^T\Gamma_1\left(\Theta_{F_1}-\hat{\Theta}_{F_1}\right)+0.5\left(\Theta_{F_2}-\hat{\Theta}_{F_2}\right)^T\Gamma_1\left(\Theta_{F_2}-\hat{\Theta}_{F_2}\right) \\ &\quad +0.5\left(\Theta_{G_1}-\hat{\Theta}_{G_1}\right)^T\Gamma_2\left(\Theta_{G_1}-\hat{\Theta}_{G_1}\right)+0.5\left(\Theta_{G_2}-\hat{\Theta}_{G_2}\right)^T\Gamma_2\left(\Theta_{G_2}-\hat{\Theta}_{G_2}\right) \\ &\quad +0.5X^TPX \end{aligned} \tag{4.27}$$

where:

$$\Gamma = \begin{bmatrix} \gamma_1 & 0 & \cdots & 0 \\ 0 & \gamma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \gamma_{121} \end{bmatrix} \tag{4.28}$$

$$\Theta = \begin{bmatrix} \theta_1 & \theta_2 & \cdots & \theta_{121} \end{bmatrix}^T \tag{4.29}$$

It should be noted that: $\Gamma_1$ and $\Gamma_2$ are positive definite matrices defined in equation (4.28), $\hat{\Theta}_{F_j}$ and $\hat{\Theta}_{G_j}$ are the estimations of $\Theta_{F_j}$ and $\Theta_{G_j}$, respectively, which are defined in equation (4.29) and $P$ is a positive definite matrix satisfying the Lyapunov equation: $\Xi^TP+P\Xi=-Q$, where $Q$ is an arbitrary four by four positive definite matrix.

The time derivative of $V$ becomes:

$$\begin{aligned} \dot{V} &= X^TP\dot{X}-\Gamma_1\dot{\hat{\Theta}}_{F_1}\left(\Theta_{F_1}-\hat{\Theta}_{F_1}\right)^T-\Gamma_1\dot{\hat{\Theta}}_{F_2}\left(\Theta_{F_2}-\hat{\Theta}_{F_2}\right)^T \\ &\quad -\Gamma_2\dot{\hat{\Theta}}_{G_1}\left(\Theta_{G_1}-\hat{\Theta}_{G_1}\right)^T-\Gamma_2\dot{\hat{\Theta}}_{G_2}\left(\Theta_{G_2}-\hat{\Theta}_{G_2}\right)^T \end{aligned} \tag{4.30}$$

It can be shown that $X^TP\Xi X=(X^TP\Xi X)^T$ since the result of $X^TP\Xi X$ will always yield a scalar number. Therefore, by setting up the formula as: $X^TP\Xi X = 0.5X^TP\Xi X+0.5(X^TP\Xi X)^T$, it is possible to achieve the relationship:

$$X^TP\Xi X=-0.5X^TQX \tag{4.31}$$

By substituting equations (4.26) and (4.31) into equation (4.30), the time derivative of $V$ will be revised as:

$$\begin{aligned}\dot{V} &= -0.5X^TQX + X^TPB_1\omega_1 + X^TPB_2\omega_2 \\ &\quad + \left(\Theta_{F_1} - \hat{\Theta}_{F_1}\right)^T \left[-\Gamma_1\dot{\hat{\Theta}}_{F_1} + X^TPB_1\xi_1(X)\right] \\ &\quad + \left(\Theta_{F_2} - \hat{\Theta}_{F_2}\right)^T \left[-\Gamma_1\dot{\hat{\Theta}}_{F_2} + X^TPB_2\xi_2(X)\right] \\ &\quad + \left(\Theta_{G_1} - \hat{\Theta}_{G_1}\right)^T \left[-\Gamma_2\dot{\hat{\Theta}}_{G_1} + X^TPB_1\eta_1(X)u_{1d}\right] \\ &\quad + \left(\Theta_{G_2} - \hat{\Theta}_{G_2}\right)^T \left[-\Gamma_2\dot{\hat{\Theta}}_{G_2} + X^TPB_2\eta_2(X)u_{2d}\right] \qquad (4.32)\end{aligned}$$

It is highly desirable that the minimum approximation error ($\omega_j$) be very small in order for the value to be negligible. This can be achieved by tuning the fuzzy systems to generate such a result. The final objective concerns the determination of two adaptive laws that would essentially eliminate the remaining unstable components of the system. Let the unknown parameters updating law be:

$$\dot{\hat{\Theta}}_{F_j} = \Gamma_1^{-1}\left(X^TPB_j\xi_j(X)\right) \qquad (4.33)$$

$$\dot{\hat{\Theta}}_{G_j} = \Gamma_2^{-1}\left(X^TPB_j\eta_j(X)u_{jd}\right) \qquad (4.34)$$

With the substitution of equations (4.33) and (4.34) into equation (4.32), the final result will yield:

$$\dot{V} = -0.5X^TQX + X^TPB_1\omega_1 + X^TPB_2\omega_2 \qquad (4.35)$$

This equation states that the function of $\dot{V}$ is negative semi-definite. Therefore, the corresponding closed loop system is stable.

## 4.4 Direct Adaptive Fuzzy Logic Controller Design

Direct adaptive fuzzy logic control is defined as a fuzzy controller that is a single fuzzy system comprised initially from the control knowledge. Mr. Shaocheng et al. described in [48] that an unknown nonlinear first order system can successfully become semi-globally uniformly bounded using a direct adaptive fuzzy logic approach. Therefore, this section will prove that the second order nonlinear parallel robot system described in this thesis can achieve stability based on the derivation method presented in [65].

The parallel robot model in described in equations (3.1), (3.2) and (3.3) can be written as:

$$\begin{aligned}
\dot{x}_1 &= x_3 \\
\dot{x}_2 &= x_4 \\
\begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} &= \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} + \begin{bmatrix} F_1(X) \\ F_2(X) \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}
\end{aligned} \tag{4.36}$$

where:

$$\begin{bmatrix} F_1(X) \\ F_2(X) \end{bmatrix} = D^{-1}(q') \left( C(q', \dot{q}') \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} + g(q') \right)$$

$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$ is an unknown positive constant matrix;

$X = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \in \mathbb{R}$ is the state vector of the system that is available for measurement and $\begin{bmatrix} \ddot{q}_{1d} & \ddot{q}_{2d} \end{bmatrix}^T$ consists of the desired angular accelerations.

There are interconnection terms prevailent in various robotic systems which have been analyzed and solved [37]. For the purpose of simplifying the controller design, the interconnection terms $b_{12}$ and $b_{21}$ are neglected. This results in the following system:

$$\begin{aligned}\dot{x}_1 &= x_3\\ \dot{x}_2 &= x_4\\ \begin{bmatrix}\dot{x}_3\\ \dot{x}_4\end{bmatrix} &= \begin{bmatrix}\ddot{q}_{1d}\\ \ddot{q}_{2d}\end{bmatrix}+\begin{bmatrix}F_1(X)\\ F_2(X)\end{bmatrix}+\begin{bmatrix}b_{11} & 0\\ 0 & b_{22}\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix}\end{aligned}\tag{4.37}$$

To simplify the derivation, let $b_{11} = b_1$ and $b_{22} = b_2$. The control objective is to design a feedback controller $u = u(X \mid \hat{\Theta})$ based on fuzzy systems and an adaptive law for adjusting the parameter vector $(\hat{\Theta})$, such that the actual output follows the desired output. The fuzzy rule base that will describe the behaviour of $(u)$ is defined in equation (4.2).

To incorporate the fuzzy rule base, it is logical to choose a fuzzy system for each control input. This fuzzy controller can be represented as:

$$u_j = u_{D_j}(X \mid \hat{\Theta}_j)\tag{4.38}$$

where: $j$ is either one or two, which represents the terms attached with motor one and motor two, respectively, $u_{D_j}$ is a fuzzy system and $\hat{\Theta}$ is the collection of adjustable parameters.

By using the singleton fuzzifier, the product inference engine and the centre average defuzzifier defined in equations (4.1), (4.4) and (4.5), respectively, it is possible to obtain:

$$u_{D_j}(X \mid \hat{\Theta}_j) = \frac{\sum_{k=1}^{M_{u_j}} \overline{y}_{u_j}^k \left(\mu_{A_{x_j}^k}(x_j^*)\mu_{A_{x_{j+2}}^k}(x_{j+2}^*)\right)}{\sum_{k=1}^{M_{u_j}} \left(\mu_{A_{x_j}^k}(x_j^*)\mu_{A_{x_{j+2}}^k}(x_{j+2}^*)\right)}\tag{4.39}$$

where: $M_{u_j}$ is the number of fuzzy rules, which is the product of the number of fuzzy sets in $x_1$ and $x_3$ or $x_2$ and $x_4$, respectively; $\mu_{A^k_{x_j}}$ and $\mu_{A^k_{x_{j+2}}}$ are the fuzzy sets for $x_1$ and $x_3$ or $x_2$ and $x_4$, respectively; $\overline{y}^k_{u_j}$ is the centre of the $k^{th}$ fuzzy set; $k$ is the $k^{th}$ rule in the fuzzy rule base.

Let $\overline{y}^k_{u_j}$ be the free parameters that are collected into $\hat{\Theta}_j \in \mathbb{R}^{\prod_{i=1}^{n} M_{u_j}}$ in order to rewrite the fuzzy controller in equation (4.39) as:

$$u_{D_j}(X \mid \hat{\Theta}_j) = \hat{\Theta}_j^T \xi_j(X) \tag{4.40}$$

where:

$$\xi_j = \begin{bmatrix} \xi_{j1} & \cdots & \xi_{jM_{F_j}} \end{bmatrix}^T \tag{4.41}$$

with:

$$\xi_{jk}(X) = \frac{\mu_{A^k_{x_j}}(x^*_j)\mu_{A^k_{x_{j+2}}}(x^*_{j+2})}{\sum\limits_{k=1}^{M_{u_j}} \left(\mu_{A^k_{x_j}}(x^*_j)\mu_{A^k_{x_{j+2}}}(x^*_{j+2})\right)} \tag{4.42}$$

The direct adaptive fuzzy controller will utilize 121 distinct rules due to the number of membership functions defined in Figure 5.1. Therefore, $M_{u_j}$, which is found in equation (4.42), is set to 121 to reflect the fuzzy system employed on the two degrees of freedom parallel robot.

This concludes the design portion of the fuzzy controller. The next task is to design an adaptive law for $\hat{\Theta}$ such that the tracking error, namely $x_1$ and $x_2$, is minimized.

Let the ideal controller be defined as:

$$\begin{bmatrix} u^*_1 \\ u^*_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{b_1} & 0 \\ 0 & \frac{1}{b_2} \end{bmatrix} \left( \begin{bmatrix} -F_1(X) - \ddot{q}_{1d} \\ -F_2(X) - \ddot{q}_{2d} \end{bmatrix} - KX \right) \tag{4.43}$$

where:

$$K = \begin{bmatrix} k_{11} & 0 & k_{12} & 0 \\ 0 & k_{21} & 0 & k_{22} \end{bmatrix}$$

represents all the coefficients of the stable polynomial $s^2 + k_{j2}s + k_{j1}$ with $k_{11}$, $k_{12}$, $k_{21}$ and $k_{22}$ being positive constants.

By substituting equation (4.38) into equation (4.37) and rearranging, the result will yield:

$$\begin{aligned} \dot{x}_1 &= x_3 \\ \dot{x}_2 &= x_4 \\ \begin{bmatrix} \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} &= \begin{bmatrix} b_1\left(u_1^* - u_{D1}(X \mid \hat{\Theta}_1)\right) \\ b_2\left(u_2^* - u_{D2}(X \mid \hat{\Theta}_2)\right) \end{bmatrix} - KX \end{aligned} \tag{4.44}$$

Let: $\Xi = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -k_{11} & 0 & -k_{12} & 0 \\ 0 & -k_{21} & 0 & -k_{22} \end{bmatrix}$, $B_1 = \begin{bmatrix} 0 \\ 0 \\ b_1 \\ 0 \end{bmatrix}$ and $B_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ b_2 \end{bmatrix}$ with $B = \begin{bmatrix} B_1 & B_2 \end{bmatrix}$. Therefore, the closed loop dynamic equation (4.44) can be rewritten as:

$$\dot{X} = \Xi X + B \begin{bmatrix} u_1^* - u_{D1}(X \mid \hat{\Theta}_1) \\ u_2^* - u_{D2}(X \mid \hat{\Theta}_2) \end{bmatrix} \tag{4.45}$$

It should be noted that: $\dot{X} = [\ \dot{x}_1 \quad \dot{x}_2 \quad \dot{x}_3 \quad \dot{x}_4\ ]^T$. Now it is appropriate to define the optimal parameters as:

$$\Theta_j = \arg \min_{\hat{\Theta}_j \in \mathbb{R}^{\prod_{i=1}^{n} M_{u_j}}} \left[ \sup_{X \in \mathbb{R}^n} \left| u_{Dj}(X \mid \hat{\Theta}_j) - u_j^* \right| \right] \tag{4.46}$$

By implementing the fuzzy system in equation (4.39) with the approximator stated in equation (4.46), the minimum approximation error can be defined as:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} u_{D1}(X \mid \Theta_1) - u_1^* \\ u_{D2}(X \mid \Theta_2) - u_2^* \end{bmatrix} \tag{4.47}$$

Therefore, by substituting equations (4.40) and (4.47) into equation (4.45), the following closed loop dynamic equation can be realized:

$$\dot{X} = \Xi X + B \begin{bmatrix} \left(\Theta_1 - \hat{\Theta}_1\right)^T \xi_1(X) - \omega_1 \\ \left(\Theta_2 - \hat{\Theta}_2\right)^T \xi_2(X) - \omega_2 \end{bmatrix} \tag{4.48}$$

Similar to the indirect adaptive fuzzy controller approach, a Lyapunov function candidate is needed to minimize the effect of the tracking errors $x_1$ and $x_2$ and the parameter errors $(\Theta_j - \hat{\Theta}_j)$. To achieve this sought result, let this candidate be:

$$V = 0.5X^T PX + 0.5\left(\Theta_1 - \hat{\Theta}_1\right)^T \Gamma \left(\Theta_1 - \hat{\Theta}_1\right) + 0.5\left(\Theta_2 - \hat{\Theta}_2\right)^T \Gamma \left(\Theta_2 - \hat{\Theta}_2\right) \tag{4.49}$$

It should be noted that: $\Gamma$ is a positive definite matrix defined in equation (4.28), $\hat{\Theta}_j$ is the estimation of $\Theta_j$, which is defined in equation (4.29) and $P$ is a positive definite matrix satisfying the Lyapunov equation: $\Xi^T P + P\Xi = -Q$, where $Q$ is an arbitrary four by four positive definite matrix.

The time derivative of $V$ becomes:

$$\dot{V} = X^T P\dot{X} - \Gamma\dot{\hat{\Theta}}_1 \left(\Theta_1 - \hat{\Theta}_1\right)^T - \Gamma\dot{\hat{\Theta}}_2 \left(\Theta_2 - \hat{\Theta}_2\right)^T \tag{4.50}$$

By substituting equations (4.31) and (4.48) into equation (4.50), the end result will become:

$$\begin{aligned}\dot{V} &= -0.5X^TQX - X^TPB_1\omega_1 - X^TPB_2\omega_2 \\ &\quad + \left(\Theta_1 - \hat{\Theta}_1\right)^T \left[X^TPB_1\xi_1(X) - \Gamma_1\dot{\hat{\Theta}}_1\right] \\ &\quad + \left(\Theta_2 - \hat{\Theta}_2\right)^T \left[X^TPB_2\xi_2(X) - \Gamma_2\dot{\hat{\Theta}}_2\right] \qquad (4.51)\end{aligned}$$

It is highly desirable that the minimum approximation error ($\omega_j$) be very small in order for the value to be negligible. This can be achieved by tuning the fuzzy system to generate such a result. The final objective concerns the determination of the adaptive law that would essentially eliminate the remaining unstable components of the system. Let the unknown parameters updating law be:

$$\dot{\hat{\Theta}}_j = \Gamma^{-1}\left[X^TPB_j\xi_j(X)\right] \qquad (4.52)$$

With the substitution of equation (4.52) into equation (4.51), the final result will yield:

$$\dot{V} = -0.5X^TQX - X^TPB_1\omega_1 - X^TPB_2\omega_2 \qquad (4.53)$$

This equation states that the function of $\dot{V}$ is negative semi-definite. Therefore, the corresponding closed loop system is stable.

## 4.5 Fuzzy Logic Adaptive Backstepping Controller Design

The controller to be examined in this section combines all the individual control techniques implemented in the prior controllers discussed in this thesis, namely: fuzzy control, adaptive control and backstepping control. The amalgamation of all three into one has proven to yield very compelling results depending on how the designer proves that a system can achieve stability. Sheng et al. discussed an adaptive fuzzy backstepping controller for a single input, single output nonlinear system with a strict feedback structure and proved that the closed loop system is semi-globally stable [49]. Wei et al. achieved asymptotic stability on a servo system that employed a similar controller to compensate the nonlinear friction that is present in an X-Y table [67]. Another example pertains to the novel approach Hsu et al. presented to solve the traditional problem of model reference adaptive control for a class of single input, single output minimum phase uncertain nonlinear system. The system was proven to converge asymptotically [20]. All the systems described in these papers were improvements to a previously implemented controller; hence it is prudent to determine whether a fuzzy adaptive backstepping controller would yield better results than the previously described controllers on the two degrees of freedom parallel robot discussed in this thesis. In order to ensure the stability of the closed loop system, a Lyapunov function must be chosen. Let this candidate function be:

$$V_1 = 0.5x_1^2 + 0.5x_2^2 \tag{4.54}$$

By using the techniques described in [76], it is possible to define the virtual controllers as: $\alpha_1 = -c_1x_1$ and $\alpha_2 = -c_2x_2$, where $c_1$ and $c_2$ are positive constants.

The derivative of $V_1$ with respect to time then becomes:

$$\dot{V}_1 = -c_1 x_1^2 - c_2 x_2^2 + x_1(x_3 - \alpha_1) + x_2(x_4 - \alpha_2) \tag{4.55}$$

Now it is sensible to delineate the second Lyapunov function candidate as:

$$W = V_1 + 0.5 \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} D(q') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \tag{4.56}$$

The time derivative of $W$ becomes:

$$\begin{aligned} \dot{W} &= \dot{V}_1 + \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} D(q') \begin{bmatrix} (\dot{x}_3 - \dot{\alpha}_1) \\ (\dot{x}_4 - \dot{\alpha}_2) \end{bmatrix} \\ &\quad + 0.5 \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \dot{D}(q') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \end{aligned} \tag{4.57}$$

As previously stated, $\dot{D}(q') - 2C(q', \dot{q}')$ is skew symmetric [14]; hence:

$$\begin{aligned} &0.5 \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \dot{D}(q') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \\ &= \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} C(q', \dot{q}') \begin{bmatrix} (x_3 - \alpha_1) \\ (x_4 - \alpha_2) \end{bmatrix} \end{aligned} \tag{4.58}$$

The derivatives of the virtual controllers are defined as: $\dot{\alpha}_1 = -c_1 x_3$ and $\dot{\alpha}_2 = -c_2 x_4$. Substituting equations (3.3) and (4.58) into (4.57) yields:

$$\begin{aligned} \dot{W} &= \dot{V}_1 + \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \Bigg( D(q') \begin{bmatrix} (\ddot{q}_{1d} - \dot{\alpha}_1) \\ (\ddot{q}_{2d} - \dot{\alpha}_2) \end{bmatrix} \\ &\quad + C(q', \dot{q}') \begin{bmatrix} (\dot{q}_{1d} - \alpha_1) \\ (\dot{q}_{2d} - \alpha_2) \end{bmatrix} + g(q') - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \Bigg) \end{aligned} \tag{4.59}$$

In order to condense the following derivation, let

$$\begin{aligned} \begin{bmatrix} z_1 & z_2 \end{bmatrix} &= \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} && (4.60) \\ \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} &= D(q') \begin{bmatrix} (\ddot{q}_{1d} - \dot{\alpha}_1) \\ (\ddot{q}_{2d} - \dot{\alpha}_2) \end{bmatrix} + C(q', \dot{q}') \begin{bmatrix} (\dot{q}_{1d} - \alpha_1) \\ (\dot{q}_{2d} - \alpha_2) \end{bmatrix} + g(q') && (4.61) \end{aligned}$$

Therefore, by substituting equations (4.60) and (4.61) into equation (4.59), the result can be rewritten as:

$$\dot{W} = \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \left( \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \tag{4.62}$$

The subsequent procedure consists of estimating $f_i$ with the fuzzy system approximator defined in [7]:

$$f_i = \kappa_i^T S_i + \delta_i \tag{4.63}$$

where: $\kappa_i$ are unknown parameters; $S_i = \xi_i(X_i)$ is the fuzzy set approximator with $X_i = [x_i, x_{i+2}]^T$; $\delta_i$ is the estimation error; $i$ is the number of actuators in the system, which is two for this parallel robot.

By substituting equation (4.63) into equation (4.62), it is possible to obtain:

$$\dot{W} = \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \left( \begin{bmatrix} \kappa_1^T S_1 + \delta_1 \\ \kappa_2^T S_2 + \delta_2 \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \tag{4.64}$$

The next step consists of applying Young's inequality to aid in the development of the controller equation. This inequality will be implemented three distinct ways in order to show the differing controllers produced by a similar control technique.

### 4.5.1 Method 1

Young's inequality can be written as:

$$z_i(\kappa_i^T S_i + \delta_i) \leq z_i(\kappa_i^T S_i + 0.5 z_i) + 0.5\delta_i \tag{4.65}$$

Let the third Lyapunov function candidate be:

$$V_2 = W + (\kappa_1 - \hat{\kappa}_1)^T \Gamma_1 (\kappa_1 - \hat{\kappa}_1) + (\kappa_2 - \hat{\kappa}_2)^T \Gamma_2 (\kappa_2 - \hat{\kappa}_2) \tag{4.66}$$

where $\hat{\kappa}_1$ and $\hat{\kappa}_2$ are the estimations of $\kappa_1$ and $\kappa_2$, respectively and $\Gamma_1$ and $\Gamma_2$ are positive definite matrices.

Differentiating $V_2$ with respect to time produces:

$$\dot{V}_2 = \dot{W} - (\kappa_1 - \hat{\kappa}_1)^T \Gamma_1 \dot{\hat{\kappa}}_1 - (\kappa_2 - \hat{\kappa}_2)^T \Gamma_2 \dot{\hat{\kappa}}_2 \tag{4.67}$$

By substituting equations (4.64) and (4.65) into equation (4.67) it is possible to achieve:

$$\begin{aligned}\dot{V}_2 \quad \leq \quad & \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} \kappa_1^T S_1 + 0.5 z_1 - u_1 \\ \kappa_2^T S_2 + 0.5 z_2 - u_2 \end{bmatrix} - (\kappa_1 - \hat{\kappa}_1)^T \Gamma_1 \dot{\hat{\kappa}}_1 \\ & - (\kappa_2 - \hat{\kappa}_2)^T \Gamma_2 \dot{\hat{\kappa}}_2 + 0.5\delta_1^2 + 0.5\delta_2^2 \end{aligned} \tag{4.68}$$

To simplify the expression further, it is beneficial to add and subtract the estimator $\hat{\kappa}_i$ inside the $z_i$ matrix. This would yield:

$$\begin{aligned}\dot{V}_2 \quad \leq \quad & \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} \hat{\kappa}_1^T S_1 + 0.5 z_1 - u_1 \\ \hat{\kappa}_2^T S_2 + 0.5 z_2 - u_2 \end{bmatrix} - (\kappa_1 - \hat{\kappa}_1)^T (\Gamma_1 \dot{\hat{\kappa}}_1 - z_1 S_1) \\ & - (\kappa_2 - \hat{\kappa}_2)^T (\Gamma_2 \dot{\hat{\kappa}}_2 - z_2 S_2) + 0.5\delta_1^2 + 0.5\delta_2^2 \end{aligned} \tag{4.69}$$

The controller can now be defined as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \begin{bmatrix} c_3 + 0.5 \\ c_4 + 0.5 \end{bmatrix} + \begin{bmatrix} x_1 + \hat{\kappa}_1^T S_1 \\ x_2 + \hat{\kappa}_2^T S_2 \end{bmatrix} \tag{4.70}$$

It should be noted that $c_3$ and $c_4$ represent positive constant gains. By substituting equations (4.55), (4.60) and (4.70) into (4.69), it is possible to attain the following solution:

$$\begin{aligned} \dot{V}_2 \leq\; & -c_1 x_1^2 - c_2 x_2^2 - c_3(x_3 - \alpha_1)^2 - c_4(x_4 - \alpha_2)^2 \\ & -(\kappa_1 - \hat{\kappa}_1)^T \left[\Gamma_1 \dot{\hat{\kappa}}_1 - (x_3 - \alpha_1) S_1\right] \\ & -(\kappa_2 - \hat{\kappa}_2)^T \left[\Gamma_2 \dot{\hat{\kappa}}_2 - (x_4 - \alpha_2) S_2\right] + 0.5\delta_1^2 + 0.5\delta_2^2 \end{aligned} \tag{4.71}$$

Let the unknown parameters updating law be:

$$\begin{bmatrix} \dot{\hat{\kappa}}_1 \\ \dot{\hat{\kappa}}_2 \end{bmatrix} = \begin{bmatrix} \Gamma_1^{-1}(x_3 - \alpha_1) S_1 \\ \Gamma_2^{-1}(x_4 - \alpha_2) S_2 \end{bmatrix} \tag{4.72}$$

Therefore by substituting equation (4.72) into equation (4.71), the final result will yield:

$$\dot{V}_2 \leq -c_1 x_1^2 - c_2 x_2^2 - c_3(x_3 - \alpha_1)^2 - c_4(x_4 - \alpha_2)^2 + 0.5\delta_1^2 + 0.5\delta_2^2 \tag{4.73}$$

This equation states that the function of $\dot{V}_2$ is negative semi-definite as long as $\delta_1$ and $\delta_2$ are chosen to be small values. Therefore, the corresponding closed loop system is stable.

### 4.5.2 Method 2

Note that $\kappa_i$ can be rewritten as:

$$\kappa_i = \|\kappa_i\| \frac{\kappa_i}{\|\kappa_i\|} = \|\kappa_i\| \kappa_i^*$$

with $\kappa_i^* = \frac{\kappa_i}{\|\kappa_i\|}$. It can be shown that $\kappa_i^{*T}\kappa_i^* = 1$. Therefore, Young's inequality can be written as:

$$\begin{aligned} z_i(\kappa_i^T S_i + \delta_i) &= z_i \|\kappa_i\| \kappa_i^{*T} S_i + z_i\delta_i \\ &\leq \frac{z_i^2 \|\kappa_i\|^2 S_i^T S_i}{2b_i^2} + \frac{b_i^2}{2}\kappa_i^{*T}\kappa_i^* + 0.5z_i^2 + 0.5\delta_i^2 \\ &\leq \frac{1}{2b_i^2} z_i^2 \|\kappa_i\|^2 S_i^T S_i + \frac{b_i^2}{2} + 0.5z_i^2 + 0.5\delta_i^2 \end{aligned} \tag{4.74}$$

It should be noted that $b_i$ is a positive constant. Set $\theta_i = \|\kappa_i\|^2$ and let the third Lyapunov function candidate be:

$$V_2 = W + 0.5\Gamma_1(\theta_1 - \hat{\theta}_1)^2 + 0.5\Gamma_2(\theta_2 - \hat{\theta}_2)^2 \tag{4.75}$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the estimations of $\theta_1$ and $\theta_2$, respectively and $\Gamma_1$ and $\Gamma_2$ are positive constants.

Differentiating $V_2$ with respect to time produces:

$$\dot{V}_2 \leq \dot{W} - \Gamma_1(\theta_1 - \hat{\theta}_1)\dot{\hat{\theta}}_1 - \Gamma_2(\theta_2 - \hat{\theta}_2)\dot{\hat{\theta}}_2 \tag{4.76}$$

By substituting equations (4.64) and (4.74) into equation (4.76) it is possible to achieve:

$$\begin{aligned} \dot{V}_2 \leq \ & \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} \frac{1}{2b_1^2} z_1\theta_1 S_1^T S_1 + 0.5z_1 - u_1 \\ \frac{1}{2b_2^2} z_2\theta_2 S_2^T S_2 + 0.5z_2 - u_2 \end{bmatrix} - \Gamma_1(\theta_1 - \hat{\theta}_1)\dot{\hat{\theta}}_1 \\ & -\Gamma_2(\theta_2 - \hat{\theta}_2)\dot{\hat{\theta}}_2 + \frac{b_1^2}{2} + \frac{b_2^2}{2} + 0.5\delta_1^2 + 0.5\delta_2^2 \end{aligned} \tag{4.77}$$

To simplify the expression further, it is beneficial to add and subtract the estimator $\hat{\theta}_i$ inside the $z_i$ matrix. This would yield:

$$\begin{aligned}\dot{V}_2 \leq\ & \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \begin{bmatrix} \frac{1}{2b_1^2} z_1 \hat{\theta}_1 S_1^T S_1 + 0.5 z_1 - u_1 \\ \frac{1}{2b_2^2} z_2 \hat{\theta}_2 S_2^T S_2 + 0.5 z_2 - u_2 \end{bmatrix} \\ & -(\theta_1 - \hat{\theta}_1)(\Gamma_1 \dot{\hat{\theta}}_1 - \frac{1}{2b_1^2} z_1^2 S_1^T S_1) \\ & -(\theta_2 - \hat{\theta}_2)(\Gamma_2 \dot{\hat{\theta}}_2 - \frac{1}{2b_2^2} z_2^2 S_2^T S_2) + \frac{b_1^2}{2} + \frac{b_2^2}{2} + 0.5\delta_1^2 + 0.5\delta_2^2 \end{aligned} \tag{4.78}$$

The controller can now be defined as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \begin{bmatrix} c_3 + 0.5 + \frac{1}{2b_1^2}(x_3 - \alpha_1)\hat{\theta}_1 S_1^T S_1 \\ c_4 + 0.5 + \frac{1}{2b_2^2}(x_4 - \alpha_2)\hat{\theta}_2 S_2^T S_2 \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.79}$$

It should be noted that $c_3$ and $c_4$ represent positive constant gains. By substituting equations (4.55), (4.60) and (4.79) into (4.78), it is possible to attain the following solution:

$$\begin{aligned}\dot{V}_2 \leq\ & -c_1 x_1^2 - c_2 x_2^2 - c_3(x_3 - \alpha_1)^2 - c_4(x_4 - \alpha_2)^2 \\ & -(\theta_1 - \hat{\theta}_1)(\Gamma_1 \dot{\hat{\theta}}_1 - \frac{1}{2b_1^2}(x_3 - \alpha_1)^2 S_1^T S_1) + \frac{b_1^2}{2} + \frac{b_2^2}{2} \\ & -(\theta_2 - \hat{\theta}_2)(\Gamma_2 \dot{\hat{\theta}}_2 - \frac{1}{2b_2^2}(x_4 - \alpha_2)^2 S_2^T S_2) + 0.5\delta_1^2 + 0.5\delta_2^2 \end{aligned} \tag{4.80}$$

Let the unknown parameters updating law be:

$$\begin{bmatrix} \dot{\hat{\theta}}_1 \\ \dot{\hat{\theta}}_2 \end{bmatrix} = \begin{bmatrix} \Gamma_1^{-1} \frac{1}{2b_1^2}(x_3 - \alpha_1)^2 S_1^T S_1 \\ \Gamma_2^{-1} \frac{1}{2b_2^2}(x_4 - \alpha_2)^2 S_2^T S_2 \end{bmatrix} \tag{4.81}$$

Therefore by substituting equation (4.81) into equation (4.80), the final result will yield:

$$\dot{V}_2 \leq -c_1x_1^2 - c_2x_2^2 - c_3(x_3-\alpha_1)^2 - c_4(x_4-\alpha_2)^2 + \frac{b_1^2}{2} + \frac{b_2^2}{2} + 0.5\delta_1^2 + 0.5\delta_2^2 \quad (4.82)$$

This equation states that the function of $\dot{V}_2$ is negative semi-definite as long as $b_1$, $b_2$, $\delta_1$ and $\delta_2$ are chosen to be small values. Therefore, the corresponding closed loop system is stable.

## 4.5.3 Method 3

Young's inequality can be written to achieve:

$$z_i\kappa_i^T S_i + z_i\delta_i \leq \frac{1}{2b_i^2} z_i^2 \left\|\kappa_i\right\|^2 + 0.5b_i^2 \left\|S_i\right\|^2 + 0.5z_i^2 + 0.5\left\|\delta_i\right\|^2 \quad (4.83)$$

It should be noted that $b_i$ is a positive constant. Let $\theta_i = \left\|\kappa_i\right\|^2$. It can be seen that $\|S_i\| \leq 1$ and $\delta_i$ is a scalar number, hence the result shown in equation (4.83) can be rewritten as:

$$z_i\kappa_i^T S_i + z_i\delta_i \leq \frac{1}{2b_i^2} z_i^2\theta_i + 0.5b_i^2 + 0.5z_i^2 + 0.5\delta_i^2 \quad (4.84)$$

Let the third Lyapunov function candidate be:

$$V_2 = W + 0.5(\theta_1 - \hat{\theta}_1)^T\Gamma_1(\theta_1 - \hat{\theta}_1) + 0.5(\theta_2 - \hat{\theta}_2)^T\Gamma_2(\theta_2 - \hat{\theta}_2) \quad (4.85)$$

where $\hat{\theta}_1$ and $\hat{\theta}_2$ are the estimations of $\theta_1$ and $\theta_2$, respectively and $\Gamma_1$ and $\Gamma_2$ are positive constants.

Differentiating $V_2$ with respect to time produces:

$$\dot{V}_2 = \dot{W} - \dot{\hat{\theta}}_1^T \Gamma_1(\theta_1 - \hat{\theta}_1) - \dot{\hat{\theta}}_2^T \Gamma_2(\theta_2 - \hat{\theta}_2) \tag{4.86}$$

By substituting equations (4.64) and (4.84) into equation (4.86) it is possible to achieve:

$$\begin{aligned} \dot{V}_2 \leq\ & \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \left( \begin{bmatrix} \frac{1}{2b_1^2} z_1\theta_1 + 0.5z_1 \\ \frac{1}{2b_2^2} z_2\theta_2 + 0.5z_2 \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \\ & + \left(0.5b_1^2 + 0.5\delta_1^2\right) + \left(0.5b_2^2 + 0.5\delta_2^2\right) \\ & - \dot{\hat{\theta}}_1^T \Gamma_1 \left(\theta_1 - \hat{\theta}_1\right) - \dot{\hat{\theta}}_2^T \Gamma_2 \left(\theta_2 - \hat{\theta}_2\right) \end{aligned} \tag{4.87}$$

To simplify the expression further, it is beneficial to add and subtract the estimator $\hat{\theta}_i$ inside the $z_i$ matrix. This would yield:

$$\begin{aligned} \dot{V}_2 \leq\ & \dot{V}_1 + \begin{bmatrix} z_1 & z_2 \end{bmatrix} \left( \begin{bmatrix} \frac{1}{2b_1^2} z_1\hat{\theta}_1 + 0.5z_1 \\ \frac{1}{2b_2^2} z_2\hat{\theta}_2 + 0.5z_2 \end{bmatrix} - \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \right) \\ & + \left(0.5b_1^2 + 0.5\delta_1^2\right) + \left(0.5b_2^2 + 0.5\delta_2^2\right) \\ & + \left( \frac{1}{2b_1^2} z_1^2 - \dot{\hat{\theta}}_1^T \Gamma_1 \right) \left(\theta_1 - \hat{\theta}_1\right) + \left( \frac{1}{2b_2^2} z_2^2 - \dot{\hat{\theta}}_2^T \Gamma_2 \right) \left(\theta_2 - \hat{\theta}_2\right) \end{aligned} \tag{4.88}$$

The controller can now be defined as:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} (x_3 - \alpha_1) & (x_4 - \alpha_2) \end{bmatrix} \begin{bmatrix} \left(c_3 + 0.5 + \frac{1}{2b_1^2}\hat{\theta}_1\right) \\ \left(c_4 + 0.5 + \frac{1}{2b_2^2}\hat{\theta}_2\right) \end{bmatrix} + \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.89}$$

It should be noted that $c_3$ and $c_4$ represent positive constant gains. By substituting equations (4.55), (4.60) and (4.89) into (4.88), it is possible to attain the following solution:

$$
\begin{aligned}
\dot{V}_2 \leq\; & -c_1x_1^2 - c_2x_2^2 - c_3(x_3-\alpha_1)^2 - c_4(x_4-\alpha_2)^2 \\
& + \left(0.5b_1^2 + 0.5\delta_1^2\right) + \left(0.5b_2^2 + 0.5\delta_2^2\right) \\
& + \left(\frac{1}{2b_1^2}(x_3-\alpha_1)^2 - \dot{\hat{\theta}}_1^T\Gamma_1\right)\left(\theta_1 - \hat{\theta}_1\right) \\
& + \left(\frac{1}{2b_2^2}(x_4-\alpha_2)^2 - \dot{\hat{\theta}}_2^T\Gamma_2\right)\left(\theta_2 - \hat{\theta}_2\right)
\end{aligned}
\tag{4.90}
$$

Let the unknown parameters updating law be:

$$
\begin{bmatrix} \dot{\hat{\theta}}_1 \\ \dot{\hat{\theta}}_2 \end{bmatrix} = \begin{bmatrix} \Gamma_1^{-1}\frac{1}{2b_1^2}(x_3-\alpha_1)^2 \\ \Gamma_2^{-1}\frac{1}{2b_2^2}(x_4-\alpha_2)^2 \end{bmatrix}
\tag{4.91}
$$

Therefore by substituting equation (4.91) into equation (4.90), the final result will yield:

$$
\dot{V}_2 \leq -c_1x_1^2 - c_2x_2^2 - c_3(x_3-\alpha_1)^2 - c_4(x_4-\alpha_2)^2 + \left(0.5b_1^2 + 0.5\delta_1^2\right) + \left(0.5b_2^2 + 0.5\delta_2^2\right)
\tag{4.92}
$$

This equation states that the function of $\dot{V}_2$ is negative semi-definite as long as $b_1$, $b_2$, $\delta_1$ and $\delta_2$ are chosen to be small values. Therefore, the corresponding closed loop system is stable.

For the purposes of this thesis, only Method 3 will be employed for the simulation and experimentation of the parallel robot system. The reasoning behind this is the fact that the first two methods comprise of 121 fuzzy sets for each motor that must be solved in order to determine the controller input. By eliminating the fuzzy sets in Method 3, the computation time will be greatly reduced, while potentially generating an equally accurate control signal.

Now that all the fuzzy logic controllers have been proven to be stable, the next chapter will discuss the simulation results of the eight derived controllers.

# Chapter 5

# Controller Simulation

Each controller discussed in this chapter will contain the simulation results for: the error between the desired and actual actuated joint angles, the location of the desired and actual end effector trajectory in Cartesian space along with their respective positional output error and the overall system torque required to achieve the actual results. A preliminary conclusion will then be drawn based on the pros and cons of each control technique.

## 5.1 Trajectory Generation

The end effector of the two degrees of freedom parallel robot was simulated to follow a circular trajectory based on the implementation of the desired controller. The tracking speed utilized is defined by the angular velocity formula: $\omega = 2\pi f$, where $f$ is the tracking frequency of the end effector. The location of the circular trajectory is based on the coordinate system defined in Figure 2.2. In these simulation results, the origin of the circle based on the Cartesian coordinate system in metres is defined as (0.1059, -0.3769). The radius of the circular trajectory is 0.03 metres and the frequency implemented is 0.5 Hertz. It should be noted that the trajectory defined in this report never impedes or approaches any singular point.

## 5.2 Controller Gains

The gains of each controller were obtained through trial and error, with the figures portrayed in this thesis garnering the most desirable results. Table 5.1 will list the gains utilized for the non-adaptive and adaptive PD and backstepping controllers, while Table 5.2 will list the gains utilized for the fuzzy PD, indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers.

| *Non Fuzzy Logic Controllers* | $K_{P1}$ | $K_{P2}$ | $K_{D1}$ | $K_{D2}$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $\Gamma$ |
|---|---|---|---|---|---|---|---|---|---|
| Non-Adaptive PD | 224 | 274 | 10 | 10 | - | - | - | - | - |
| Adaptive PD | 224 | 274 | 10 | 10 | - | - | - | - | 1000 |
| Non-Adaptive Backstepping | - | - | - | - | 44 | 54 | 5 | 5 | - |
| Adaptive Backstepping | - | - | - | - | 44 | 54 | 5 | 5 | 1000 |

Table 5.1: Non-Fuzzy Logic Controller Gains

| *Fuzzy Logic Controllers* | $k_1$ | $k_2$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $b_1$ | $b_2$ | $\Gamma$ |
|---|---|---|---|---|---|---|---|---|---|
| Indirect Adaptive Fuzzy | 100 | 100 | - | - | - | - | - | - | 0.1 |
| Direct Adaptive Fuzzy | $10^4$ | $10^4$ | - | - | - | - | - | - | 0.01 |
| Fuzzy Adaptive Backstepping | - | - | 30 | 30 | 5 | 5 | 0.05 | 0.05 | 0.001 |

| *Fuzzy Logic Controllers* | $c_{out1}$ | $c_{out2}$ | $c_{e1}$ | $c_{e2}$ | $c_{\dot{e}1}$ | $c_{\dot{e}2}$ |
|---|---|---|---|---|---|---|
| Fuzzy PD | 300 | 300 | 2 | 2 | 2.5 | 2.5 |

Table 5.2: Fuzzy Logic Controller Gains

The subsequent figures will illustrate the MATLAB simulation results generated by all eight controllers derived in the previous two chapters. The findings will be split up into two sections, namely: non-fuzzy logic controller simulations and fuzzy logic controller simulations. This is done in order to achieve an impartial conclusion between the two very different controller variants. It should be noted that the desired angle and torque outputs for both motors are shown in Figure 5.1 and Figure 5.2, respectively. The constant variables $c_{out1}$, $c_{out2}$, $c_{e1}$, $c_{e2}$, $c_{\dot{e}1}$ and $c_{\dot{e}2}$ represent the scaling factors of motor one and motor two, respectively.

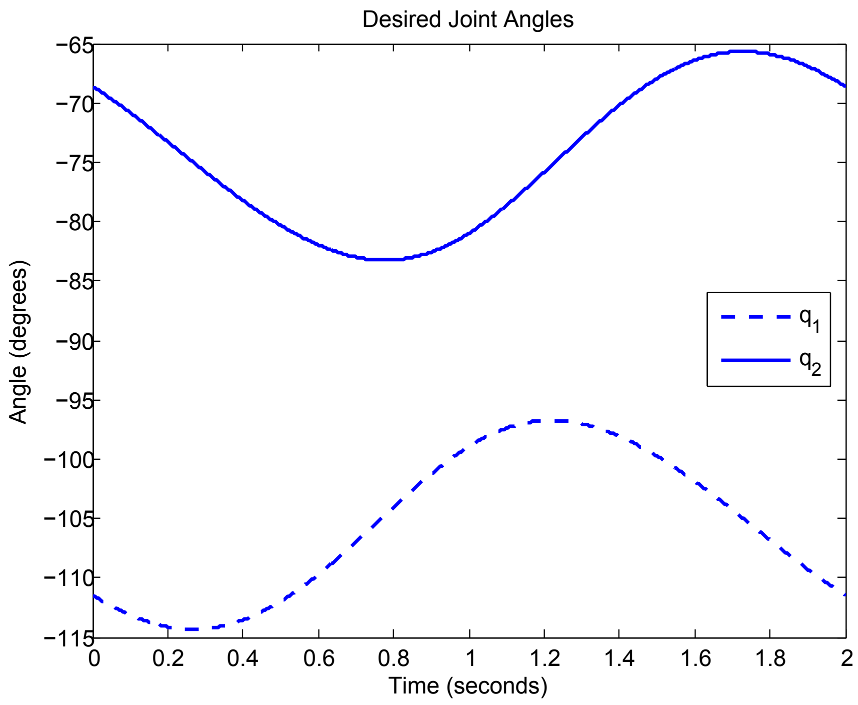


Figure 5.1: Desired Joint Angles

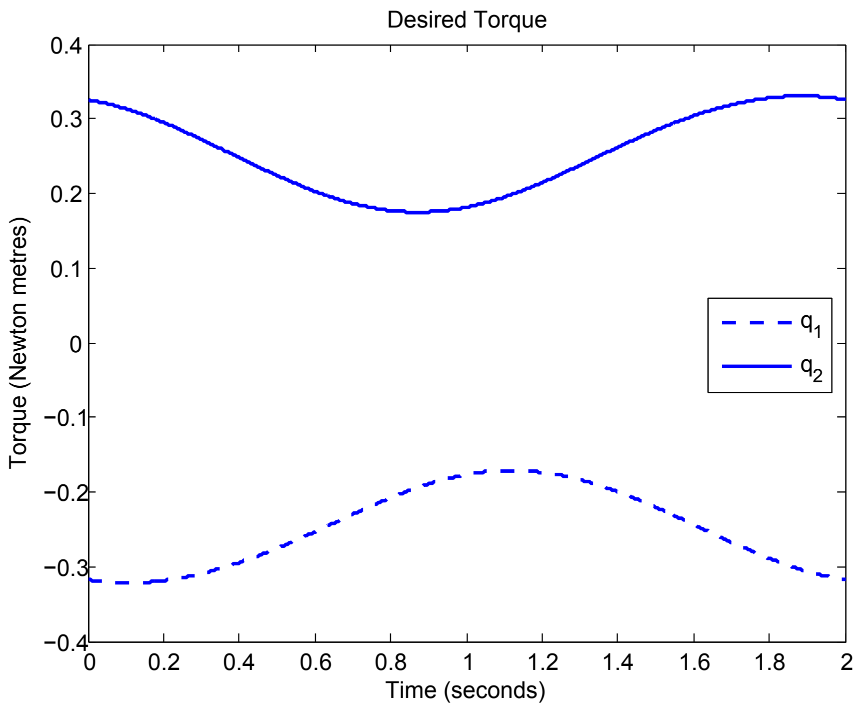


Figure 5.2: Desired Torque

## 5.3 Simulations of Non-Fuzzy Logic Controllers

To achieve accurate simulation findings, the initial error was chosen to be the same for all the non-fuzzy logic controllers. The initial error for $q_1$ is 2 degrees, while the initial error for $q_2$ is 1.5 degrees. The initial error for both $\dot{q}_1$ and $\dot{q}_2$ is zero. Both the adaptive PD and backstepping controllers require an initial value for $\hat{\Theta}$ as shown in Table 5.3. This value is defined using the solutions generated by the formulas in equation (3.15). It should be noted that the value of $\Gamma$ defined in Table 5.1 for the adaptive PD and backstepping controllers applies to all the parameters of $\Gamma$ defined in each respective controller.

| $\Theta$ | *Initial Conditions* |
|---|---|
| $\theta_1$ | 0.008150521 |
| $\theta_2$ | 0.009136644 |
| $\theta_3$ | 0.002154209 |
| $\theta_4$ | 0.002473008 |
| $\theta_5$ | 0.003231314 |
| $\theta_6$ | 0.003709512 |
| $\theta_7$ | 0.370800420 |
| $\theta_8$ | 0.415937470 |
| $\theta_9$ | 0.139717866 |
| $\theta_{10}$ | 0.160394552 |

Table 5.3: Initial Conditions for the Parameters

### 5.3.1 Joint Angles

The following figures depict the error between the desired and actual joint angles of $q_1$ and $q_2$, respectively. These are the only two angles directly controllable by the actuators of the parallel robot. It is crucial that the difference between the desired angles and the actual angles of $q_1$ and $q_2$ be as small as possible, in order for the end effector error to be minimized.

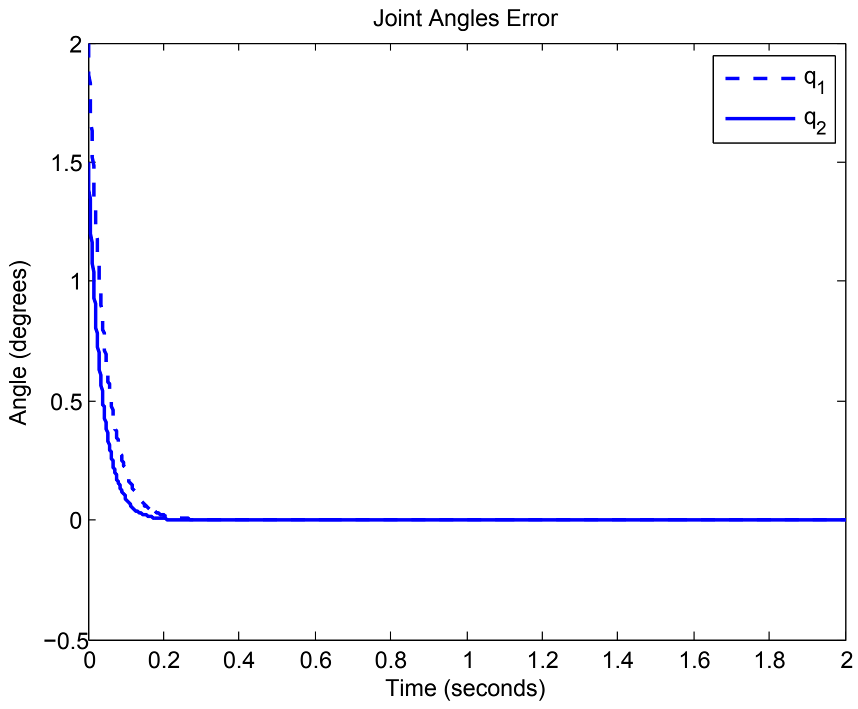


Figure 5.3: Joint Angles Error for Non-Adaptive PD

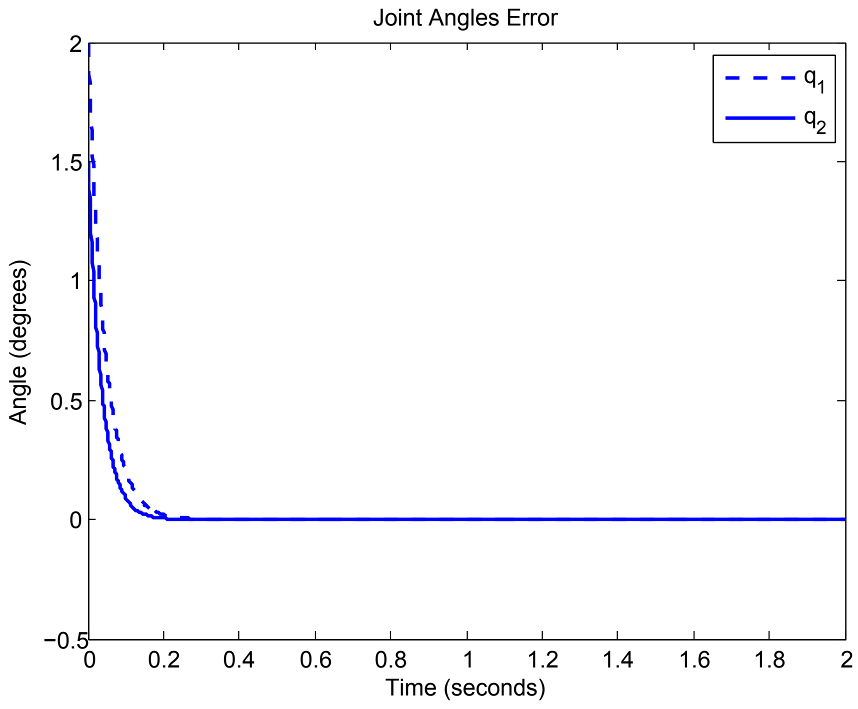


Figure 5.4: Joint Angles Error for Adaptive PD

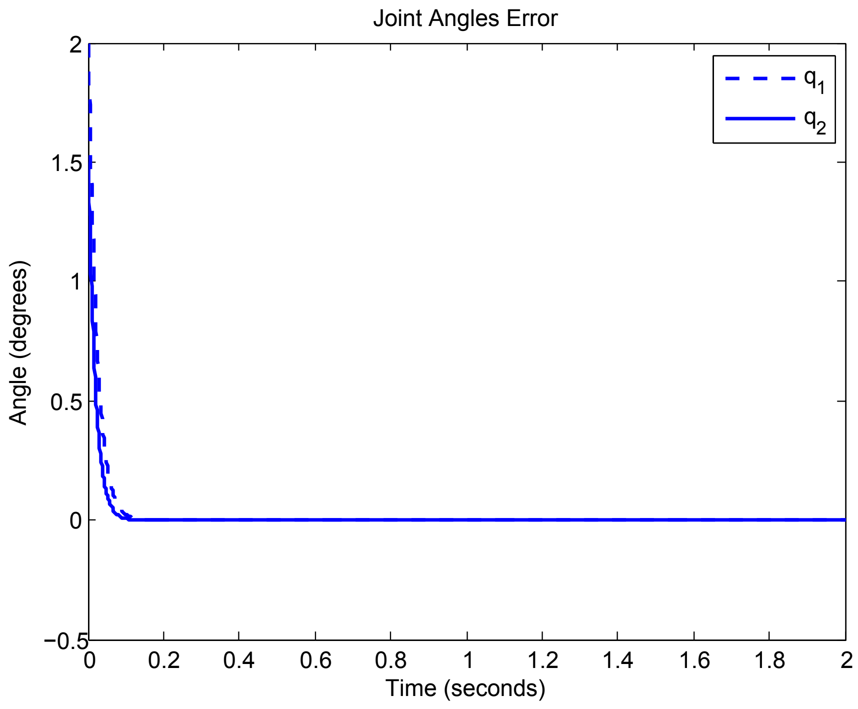


Figure 5.5: Joint Angles Error for Non-Adaptive Backstepping

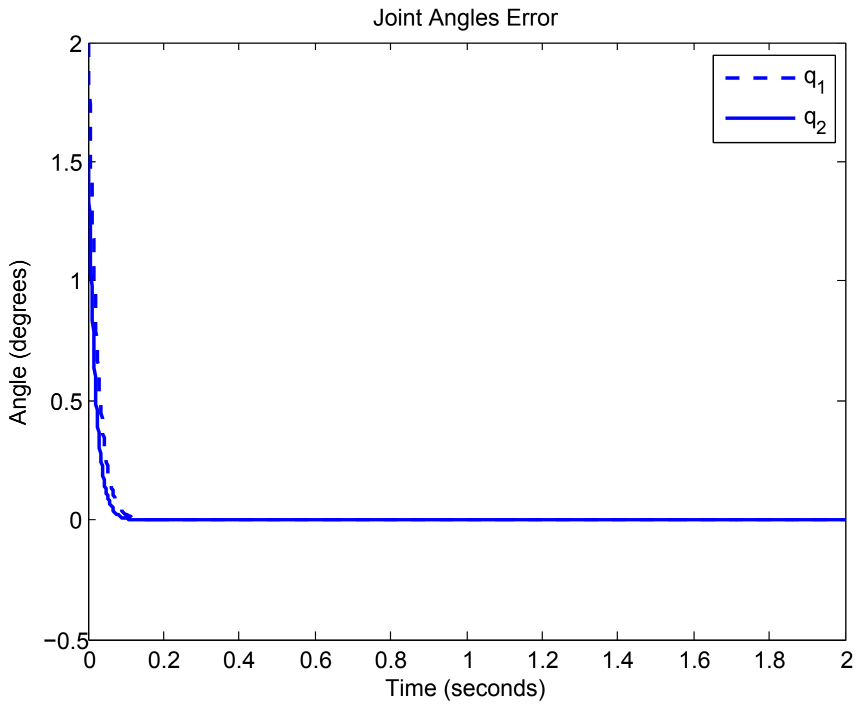


Figure 5.6: Joint Angles Error for Adaptive Backstepping

The purpose of the preceding joint angle error plots for $q_1$ and $q_2$ are to determine which controller can converge to the desired angle the quickest. The constraint that ensured unbiased results was the fact that the motors utilized in the physical model of the parallel robot could output a maximum torque of 7.5 Newton metres. This will be confirmed in the subsequent torque plots. It is clear from these figures that the non-adaptive and adaptive backstepping controllers produced the quickest convergence, which is approximately 0.1 seconds quicker than their non-adaptive and adaptive PD controller counterparts.

### 5.3.2 End Effector Trajectory

The next set of figures portray the trajectory tracking of the end effector along with its respective error between its desired and actual position. In order to determine the actual location of the end effector, equations (2.19) and (2.20) were employed to solve for $q_4$ and $q_3$, respectively. The forward kinematics equations were then applied based on all the available data to definitively determine the actual location of the end effector in Cartesian coordinates. The forward kinematics equations utilized for the two degrees of freedom planar parallel robot are based upon the works defined in [68] and [46]. The results from the joint angles of $q_1$ and $q_2$ play a crucial role in the determination of the actual location of the end effector, since a small angular error would not cause a large deviation when compared to the desired trajectory. The plots of the end effector error in the x-axis and y-axis are shown in order to easily identify the severity of the absolute accuracy.

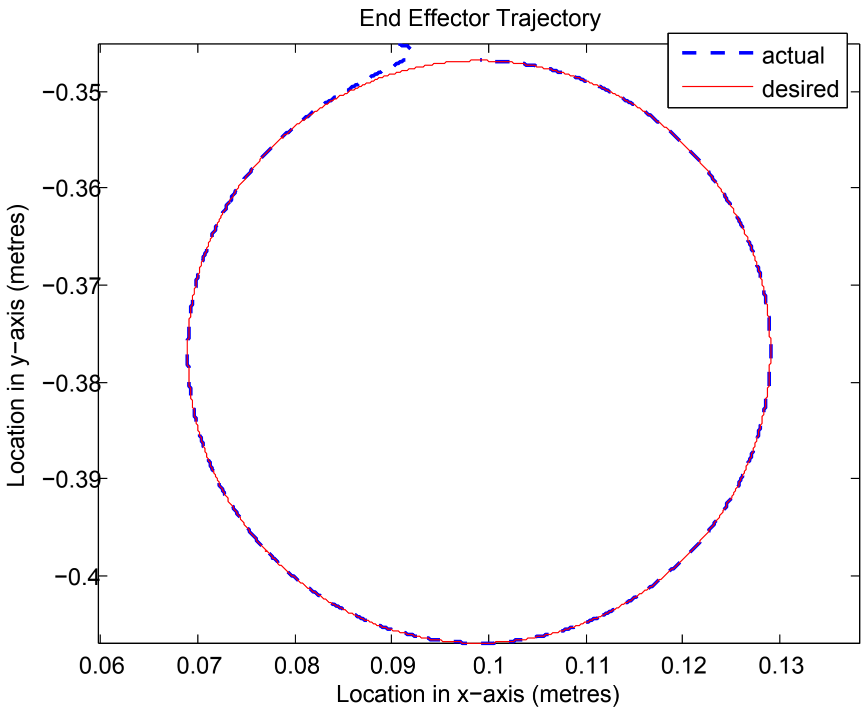


Figure 5.7: End Effector Trajectory for Non-Adaptive PD

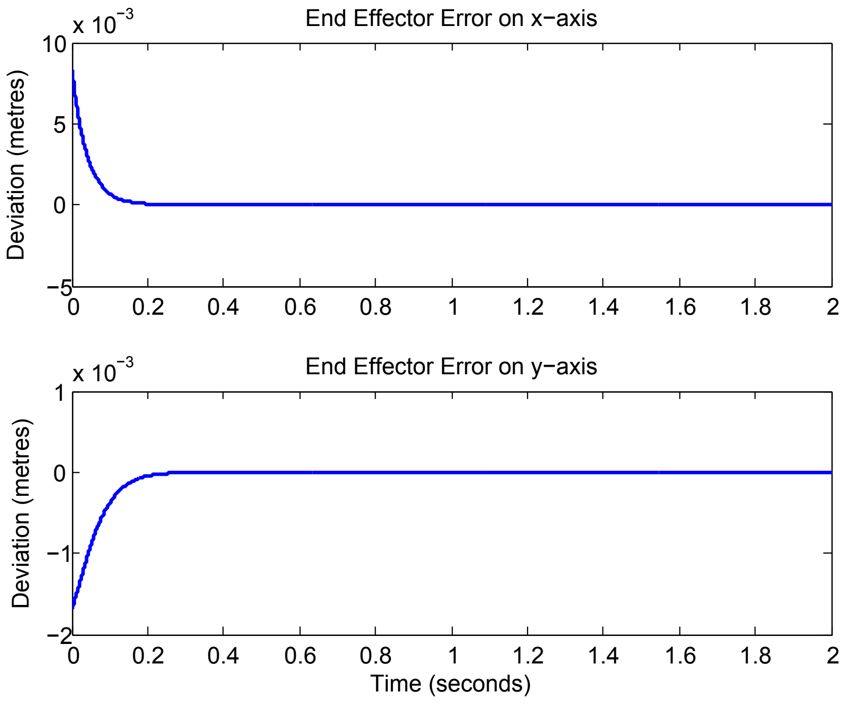


Figure 5.8: End Effector Trajectory Error for Non-Adaptive PD

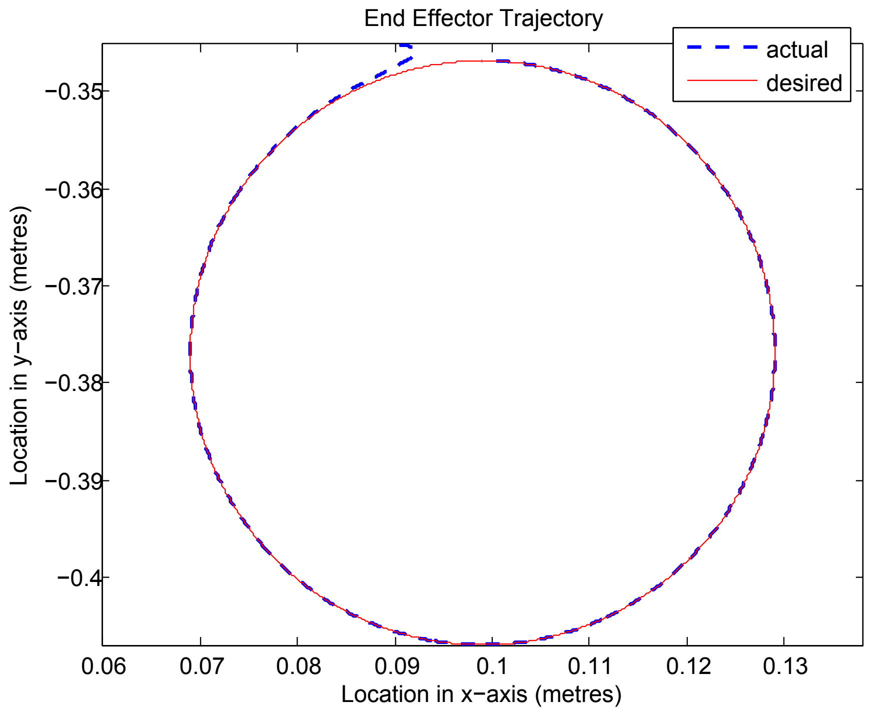


Figure 5.9: End Effector Trajectory for Adaptive PD

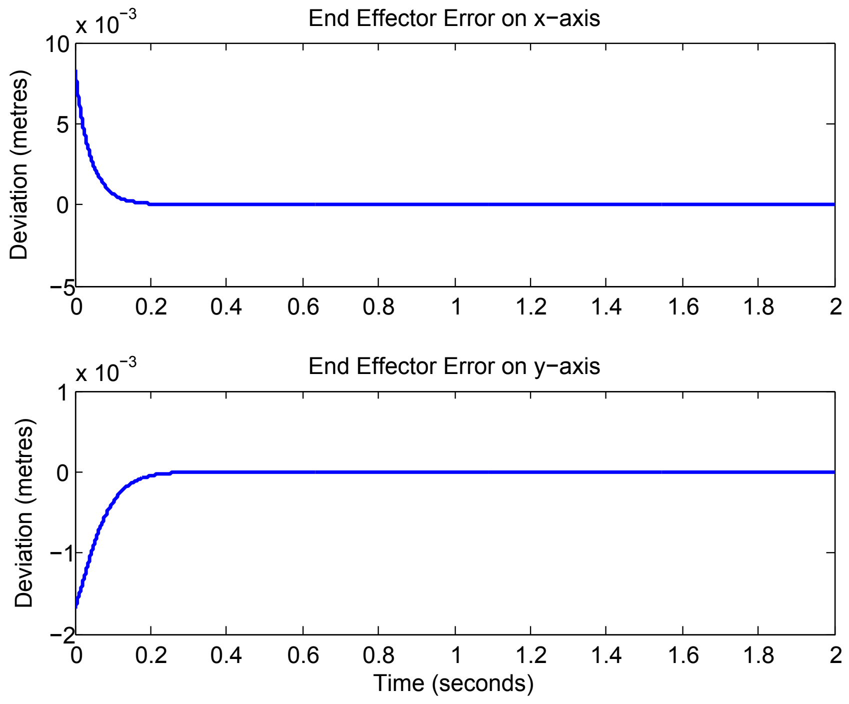


Figure 5.10: End Effector Trajectory Error for Adaptive PD

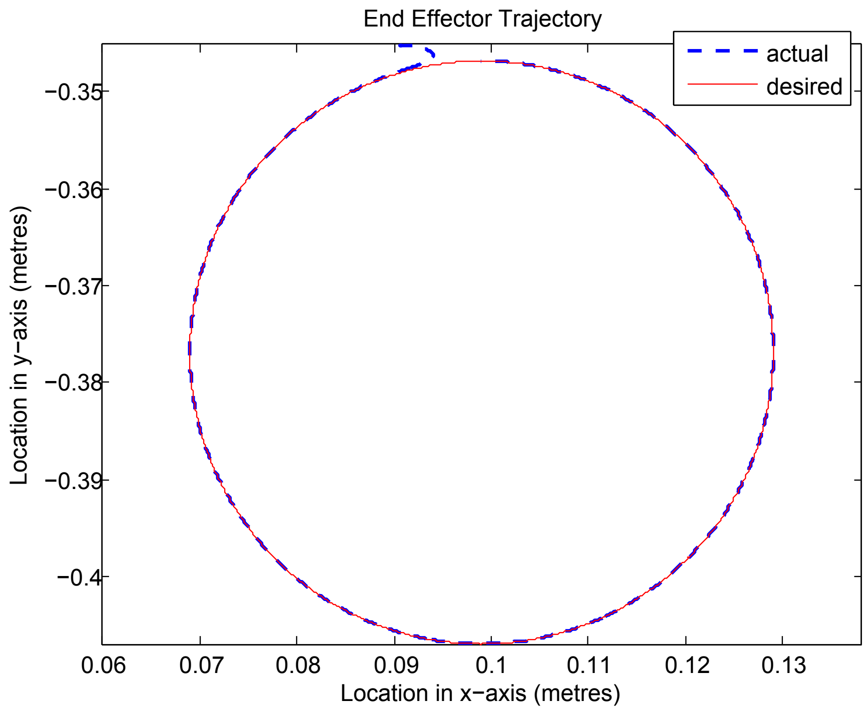


Figure 5.11: End Effector Trajectory for Non-Adaptive Backstepping

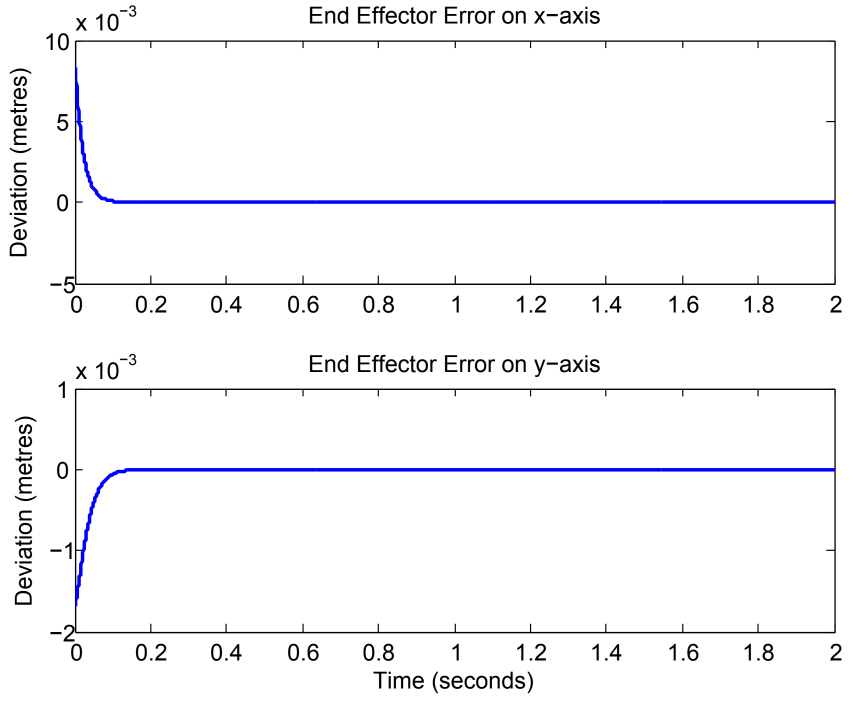


Figure 5.12: End Effector Trajectory Error for Non-Adaptive Backstepping

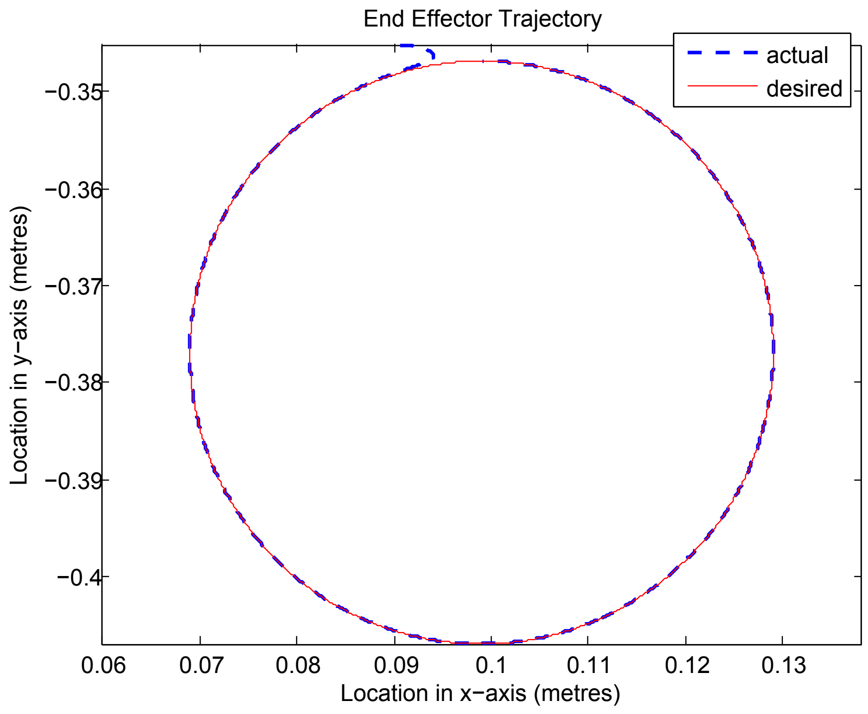


Figure 5.13: End Effector Trajectory for Adaptive Backstepping

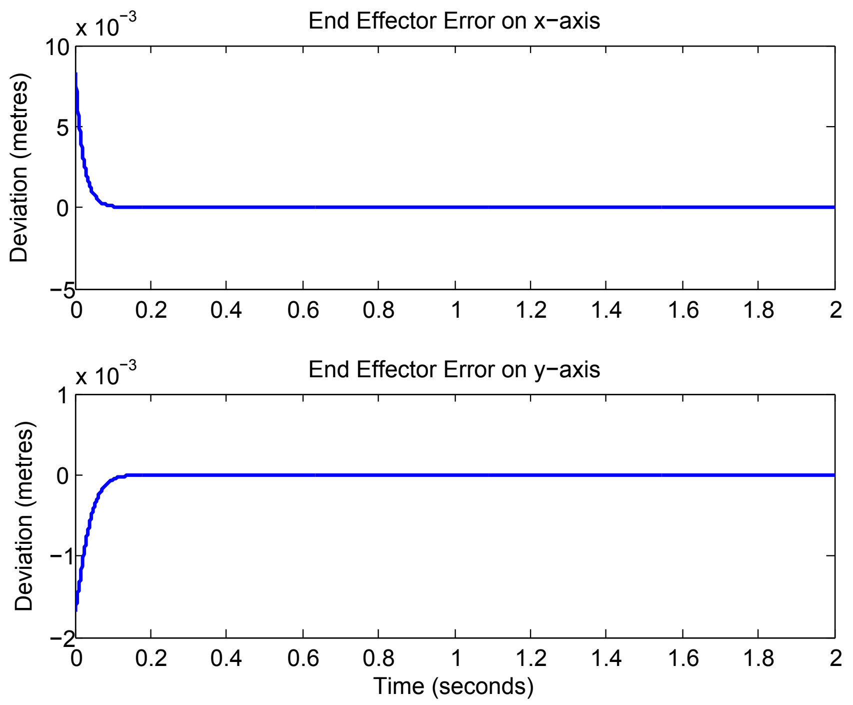


Figure 5.14: End Effector Trajectory Error for Adaptive Backstepping

Similar to the conclusion achieved in the joint angles portion of this section, the non-adaptive and adaptive backstepping controllers confirmed that the actual trajectory follows the desired trajectory approximately 0.1 seconds faster than the non-adaptive and adaptive PD controllers. It can be seen from all the end effector trajectory error figures that the x-axis error converges quicker than the y-axis error. This is due to the fact that the y-axis must overcome the force of gravity, which is more prevalent in the y-axis.

### 5.3.3 Controller Output - Torque

The last set of figures for this section illustrates the torque generated by the motors which will move the links to their actual position. To calculate the controller output required for the non-adaptive and adaptive backstepping and PD controllers, equations (3.11), (3.21), (3.29) and (3.37) were utilized, respectively. The ordinary differential equations for all four controllers and the unknown parameter updating laws for the adaptive controllers were solved using the Runge-Kutta *ode45* method in MATLAB [19].

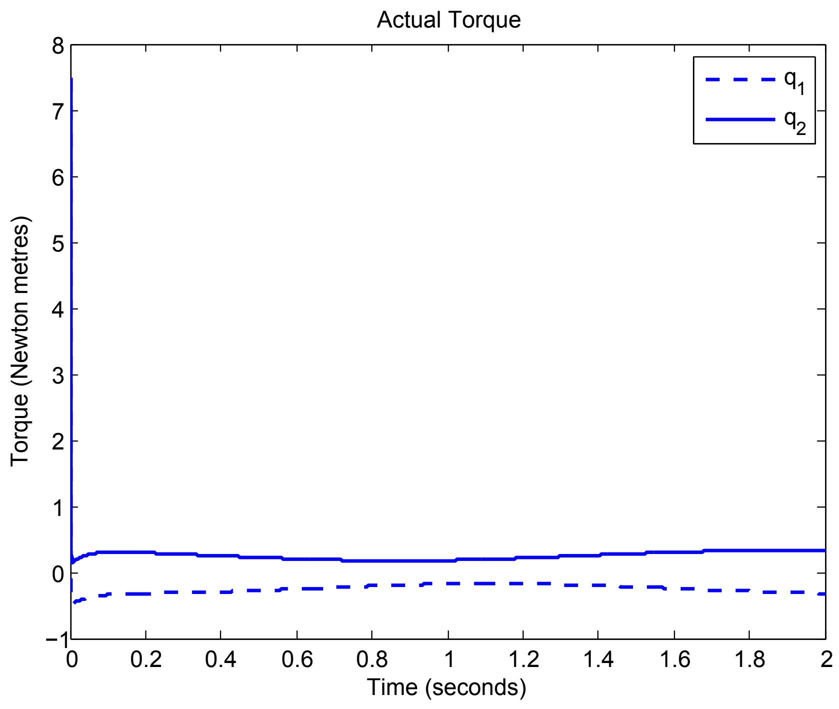


Figure 5.15: Torque for Non-Adaptive PD

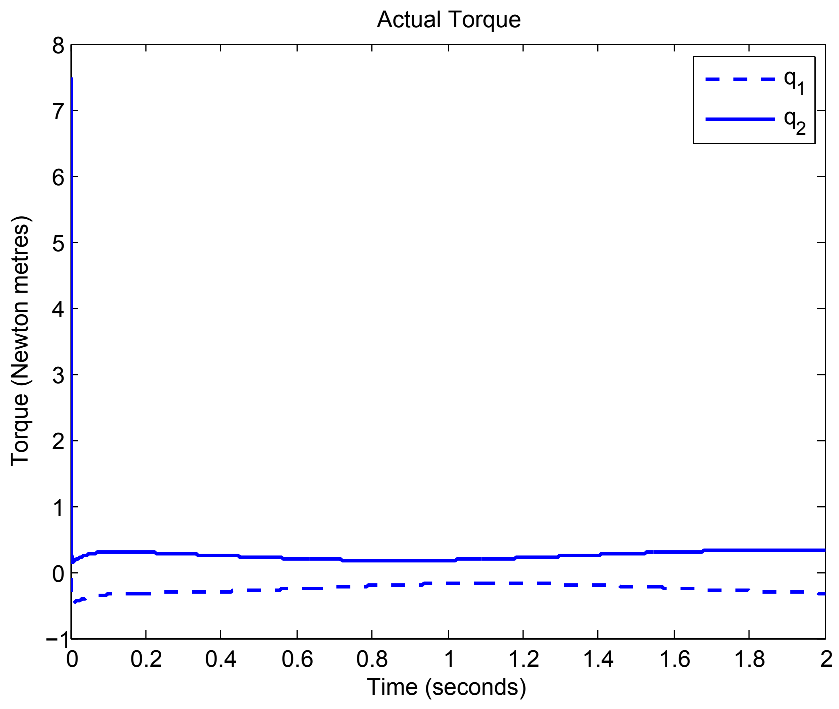


Figure 5.16: Torque for Adaptive PD

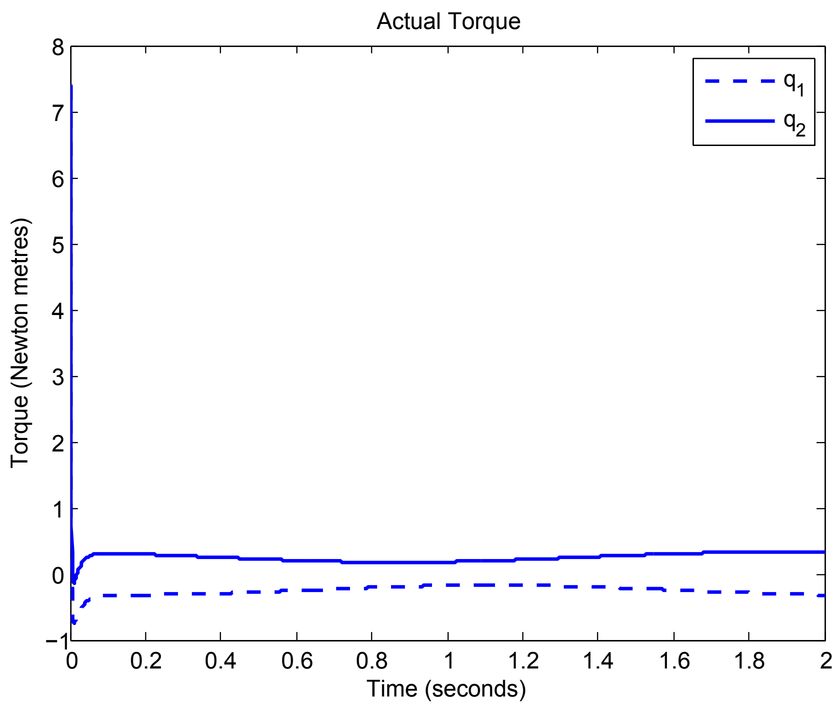


Figure 5.17: Torque for Non-Adaptive Backstepping

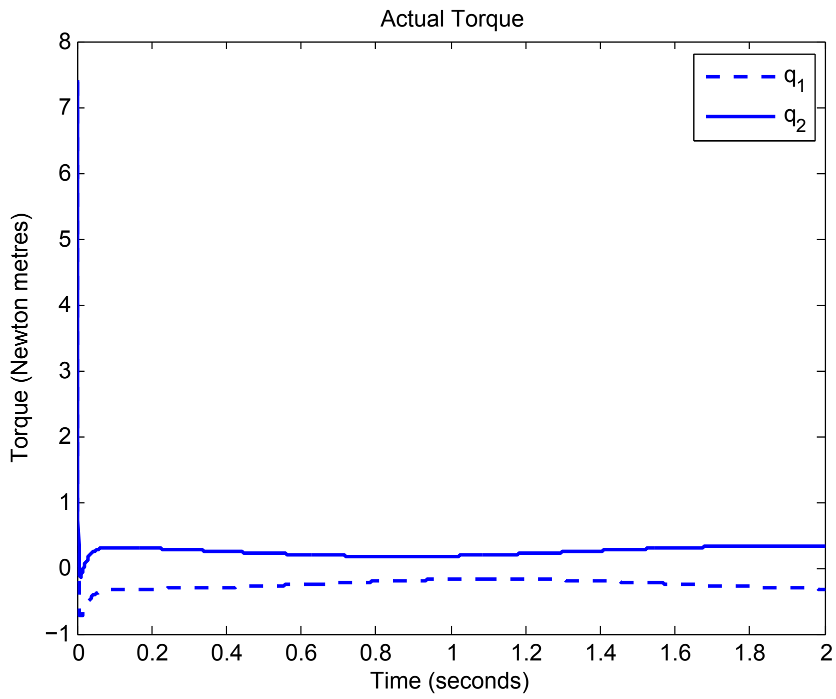


Figure 5.18: Torque for Adaptive Backstepping

As was initially stated in the joint angles portion of this section, the mechanical model of the parallel robot can achieve a peak torque of 7.5 Newton metres. The goal was to ensure that the controllers could converge to zero as quickly as possible without violating this constraint. This is the reason for the large initial torque found in the first few iterations of the torque plots. The subsequent torque data falls well under the maximum continuous torque that the motor can achieve, which is 6 Newton metres. The desired torque found in Figure 5.2 adequately portrays a detailed representation of the actual torque realized after these first few iterations.

## 5.4 Simulations of Fuzzy Logic Controllers

To achieve accurate simulation findings, the initial error for $q_1$, $q_2$, $\dot{q}_1$ and $\dot{q}_2$ was chosen to be zero for all the fuzzy logic controllers. It should be noted that the value of $\Gamma$ defined in Table 5.2 applies to all the parameters of $\Gamma$ defined in each respective controller. The indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers require an initial value for $\hat{\Theta}$. This value is defined as 0.1 for all the parameters listed in the specified controllers. Based on a 90 degree operating point, the value of $b$ in the direct adaptive fuzzy controller is 104.4766 and 99.3706 for motors one and two, respectively.

### 5.4.1 Joint Angles

The following figures depict the error between the desired and actual joint angles of $q_1$ and $q_2$, respectively. These are the only two angles directly controllable by the actuators of the parallel robot. It is crucial that the difference between the desired angles and the actual angles of $q_1$ and $q_2$ be as small as possible, in order for the end effector error to be minimized.

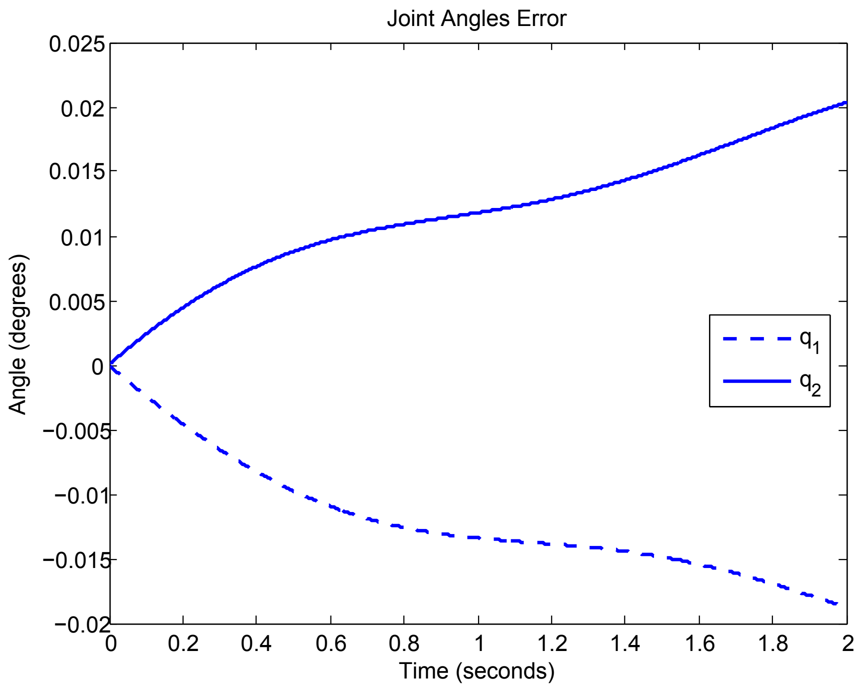


Figure 5.19: Joint Angles Error for Fuzzy PD

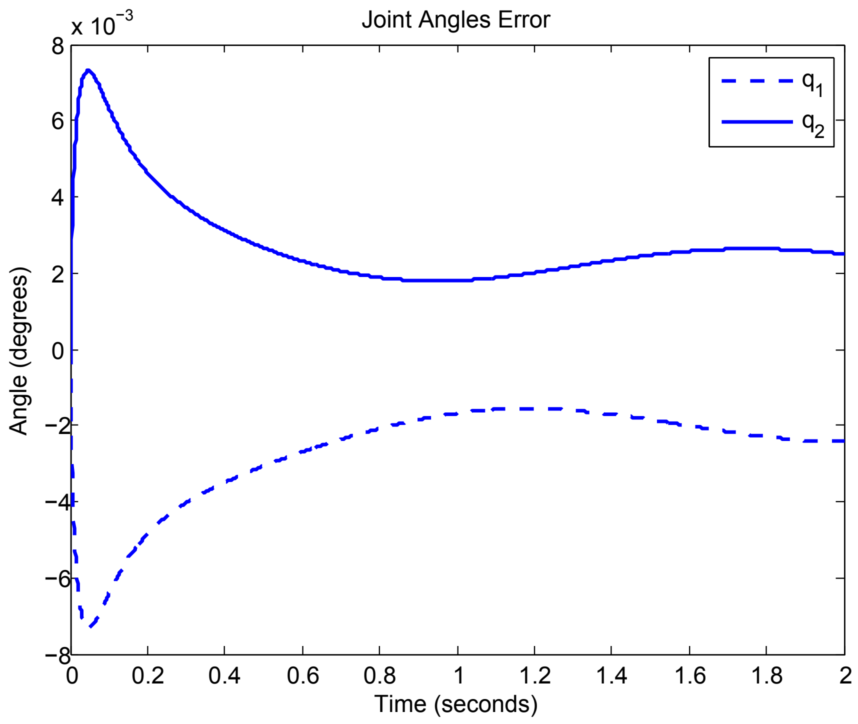


Figure 5.20: Joint Angles Error for Fuzzy Adaptive Backstepping

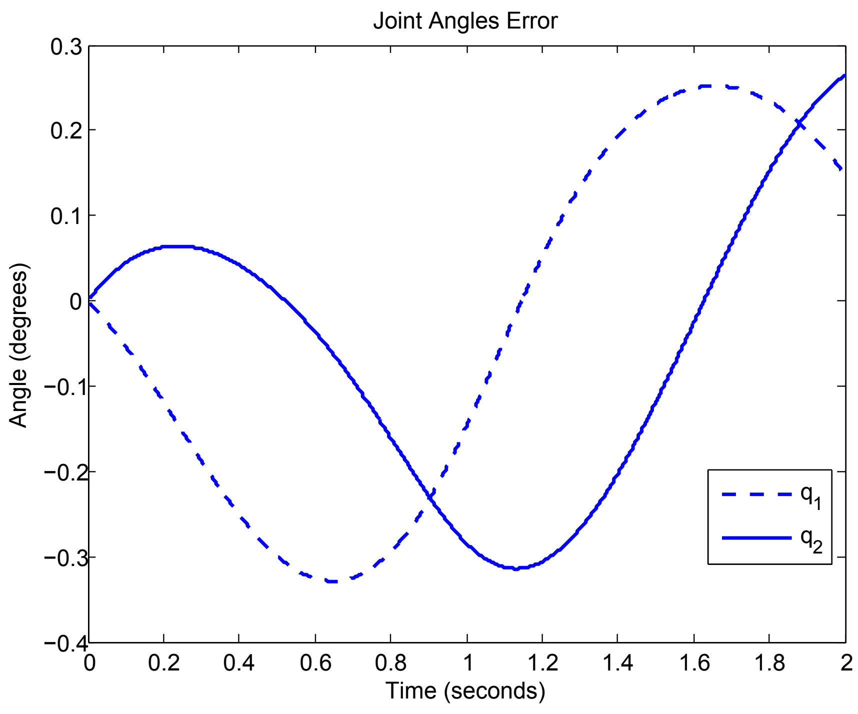


Figure 5.21: Joint Angles Error for Indirect Adaptive Fuzzy

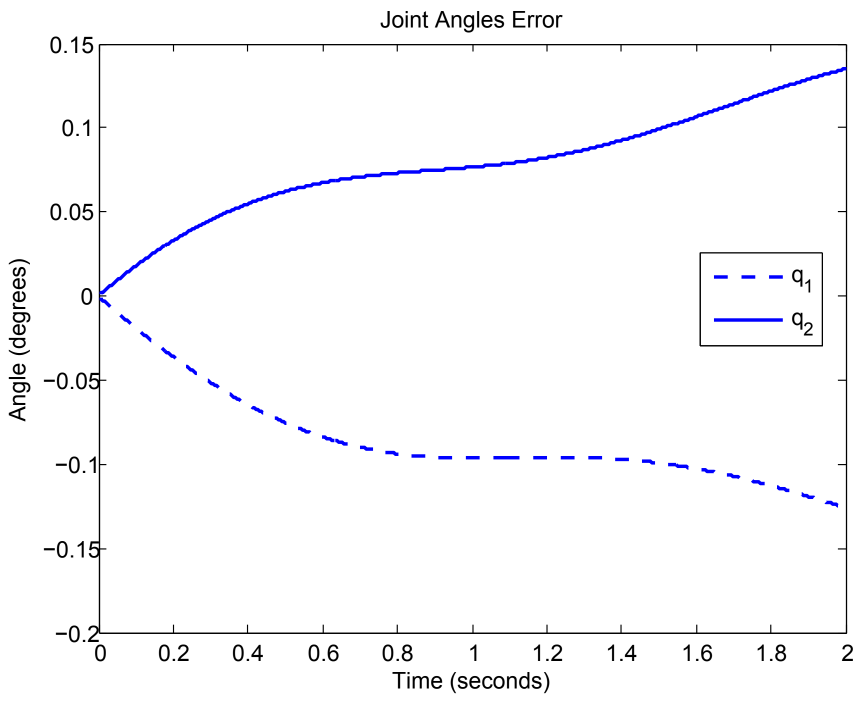


Figure 5.22: Joint Angles Error for Direct Adaptive Fuzzy

It is clear from the simulation results that the fuzzy adaptive backstepping controller achieved the most impressive results, with the indirect adaptive fuzzy controller garnering the least. The reason behind the failure of the indirect adaptive fuzzy controller will be examined in the controller output portion of this section. It is quite interesting to note that even though the fuzzy adaptive backstepping controller had the most parameters to individually tune; it accomplished the lowest joint angles error with less than 0.008 degrees.

### 5.4.2 End Effector Trajectory

The next set of figures portray the trajectory tracking of the end effector along with its respective error between its desired and actual position. In order to determine the actual location of the end effector, equations (2.19) and (2.20) were employed to solve for $q_4$ and $q_3$, respectively. The forward kinematics equations were then applied based on all the available data to definitively determine the actual location of the end effector in Cartesian coordinates. The forward kinematics equations utilized for the two degrees of freedom planar parallel robot are based upon the works defined in [68] and [46]. The results from the joint angles of $q_1$ and $q_2$ play a crucial role in the determination of the actual location of the end effector, since a small angular error would not cause a large deviation when compared to the desired trajectory. The plots of the end effector error in the x-axis and y-axis are also shown in order to easily identify the severity of the error.

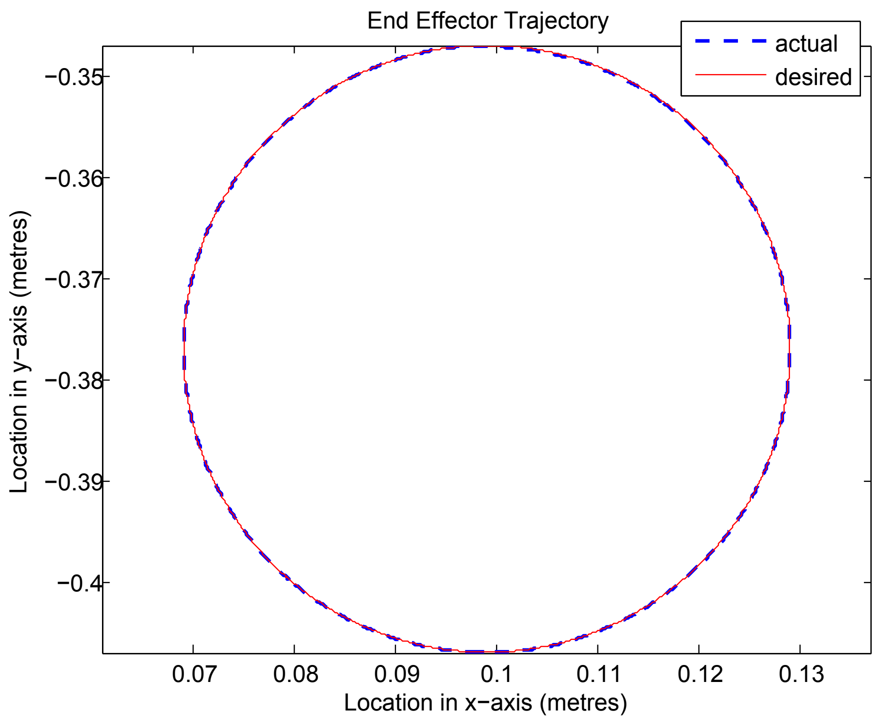


Figure 5.23: End Effector Trajectory for Fuzzy PD

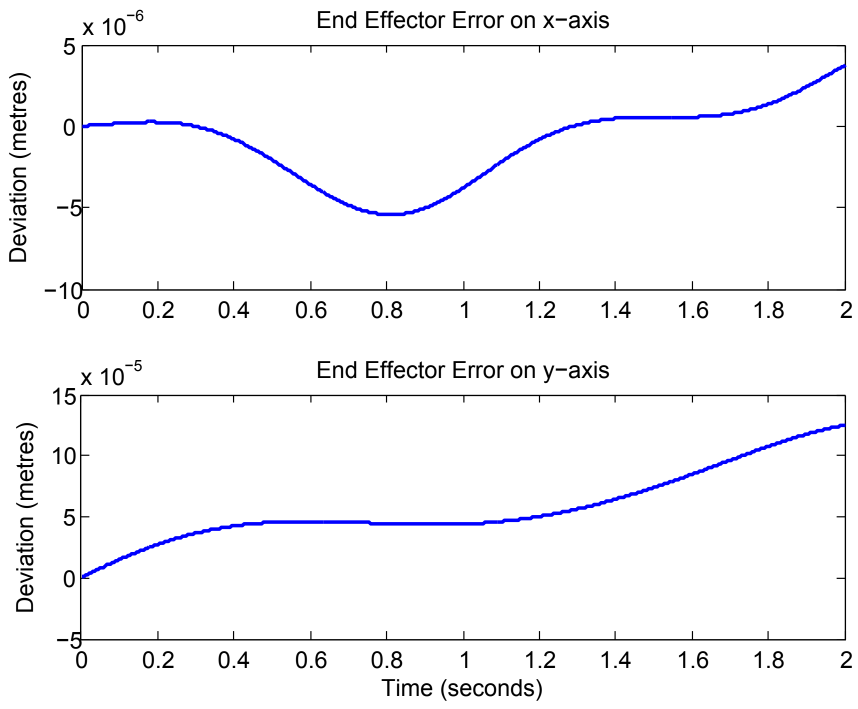


Figure 5.24: End Effector Trajectory Error for Fuzzy PD

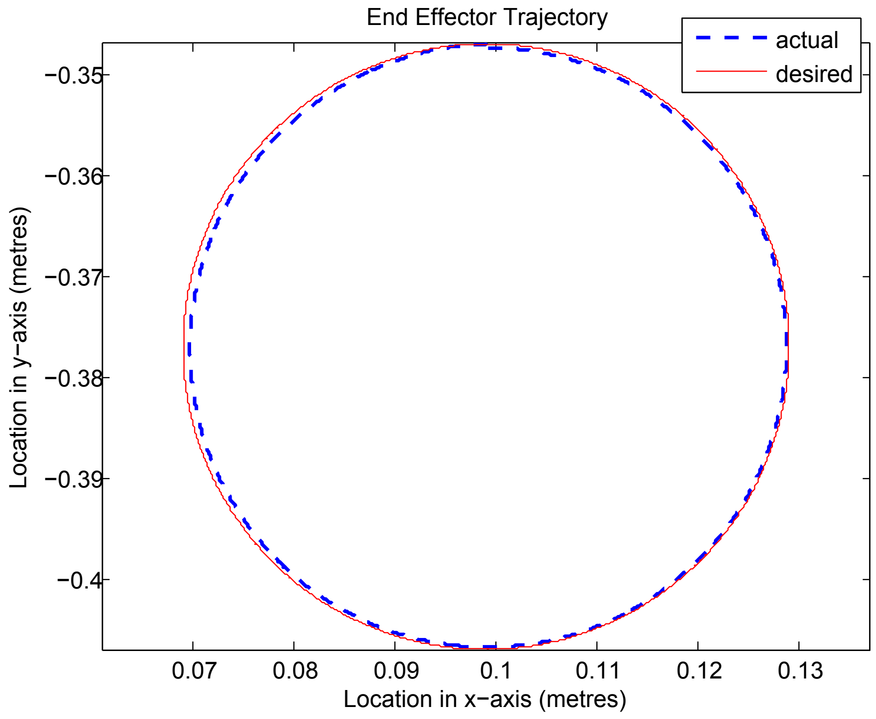


Figure 5.25: End Effector Trajectory for Indirect Adaptive Fuzzy

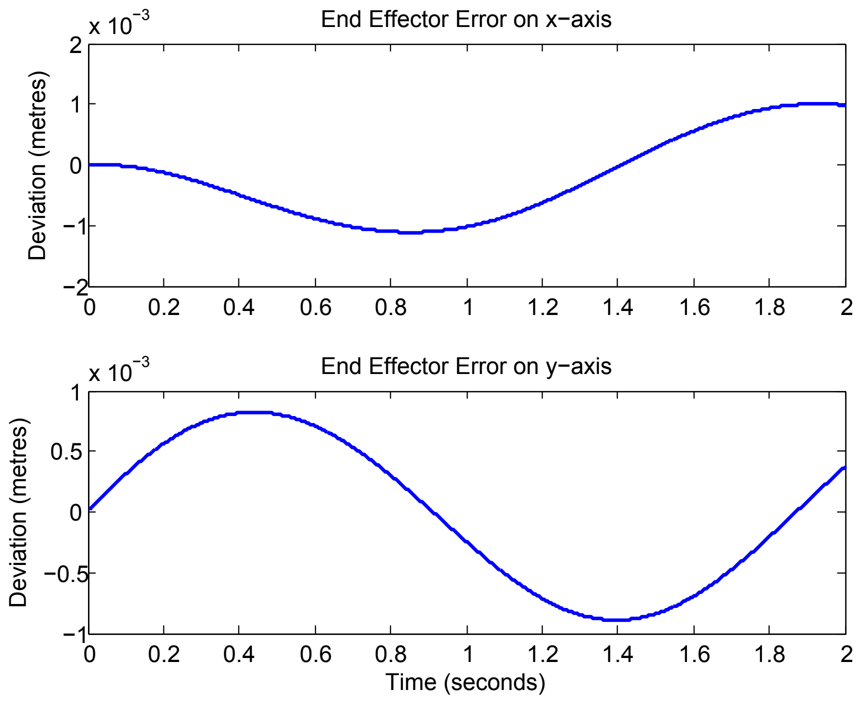


Figure 5.26: End Effector Trajectory Error for Indirect Adaptive Fuzzy

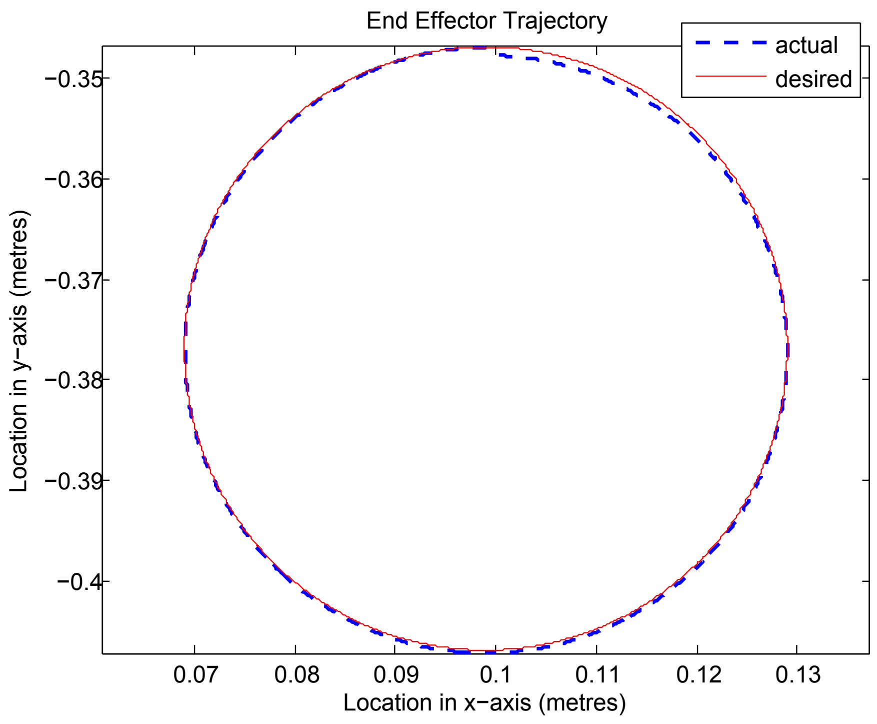


Figure 5.27: End Effector Trajectory for Direct Adaptive Fuzzy

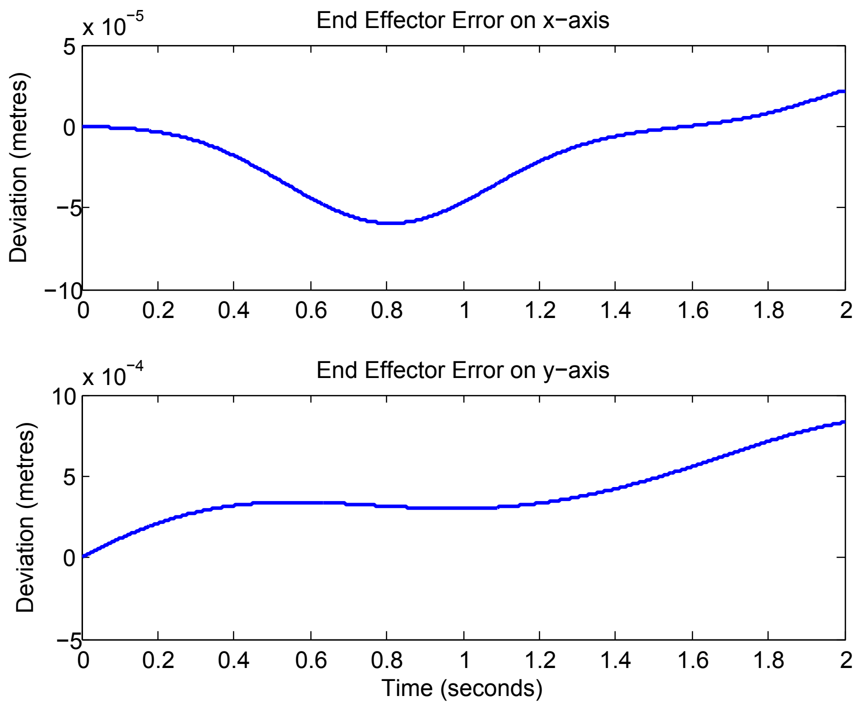


Figure 5.28: End Effector Trajectory Error for Direct Adaptive Fuzzy

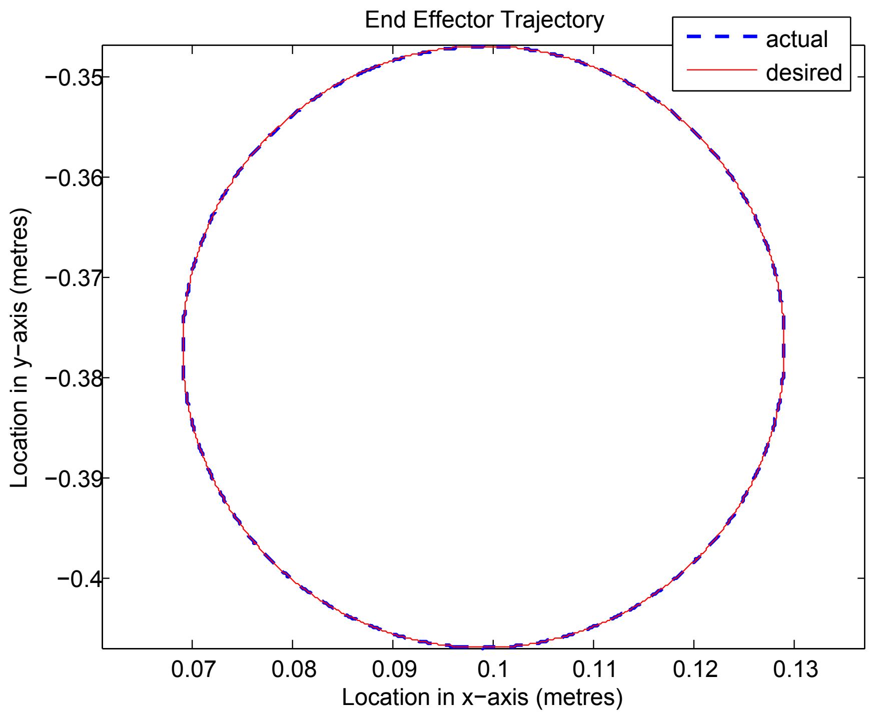


Figure 5.29: End Effector Trajectory for Fuzzy Adaptive Backstepping

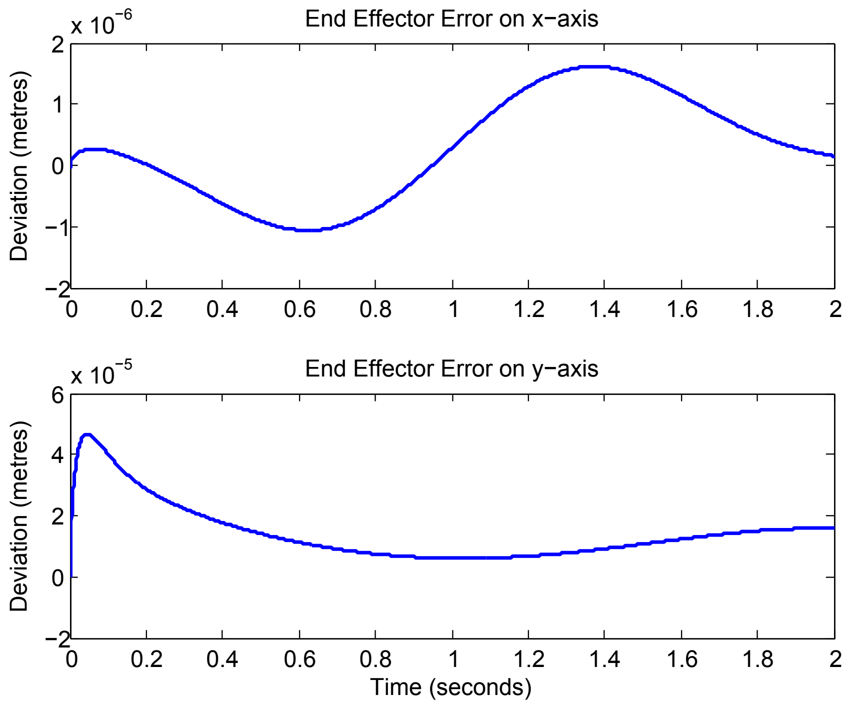


Figure 5.30: End Effector Trajectory Error for Fuzzy Adaptive Backstepping

Similar to the conclusion achieved in the joint angles portion of this section, the fuzzy adaptive backstepping controller tracked the desired end effector trajectory more accurately than any other controller. It accomplished a maximum of a mere 2 micrometre deviation in the x-axis and a 50 micrometre deviation in the y-axis. It can be seen from all the end effector trajectory error figures that the x-axis error is less than the y-axis error. This is due to the fact that the y-axis must overcome the force of gravity, which is more prevalent in the y-axis.

### 5.4.3 Controller Output - Torque

The last set of figures for this section illustrates the torque generated by the motors which will move the links to their actual position. To calculate the torque required for the fuzzy PD, indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers, equations (4.7), (4.12), (4.43) and (4.89) were utilized, respectively. The ordinary differential equations for all four controllers and the unknown parameter updating laws for the adaptive controllers were solved using the Runge-Kutta *ode45* method in MATLAB [19].

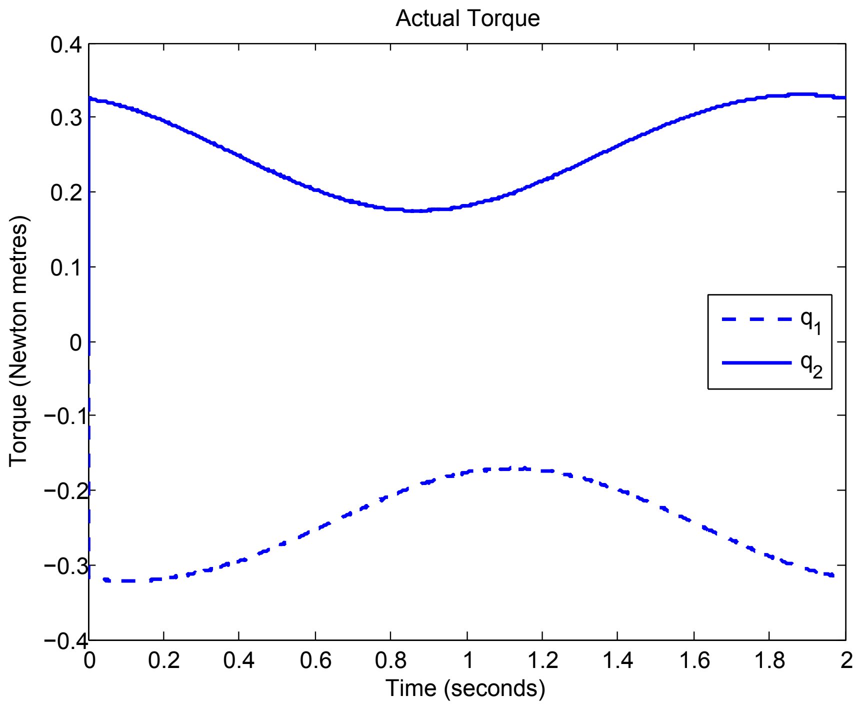


Figure 5.31: Torque for Fuzzy PD

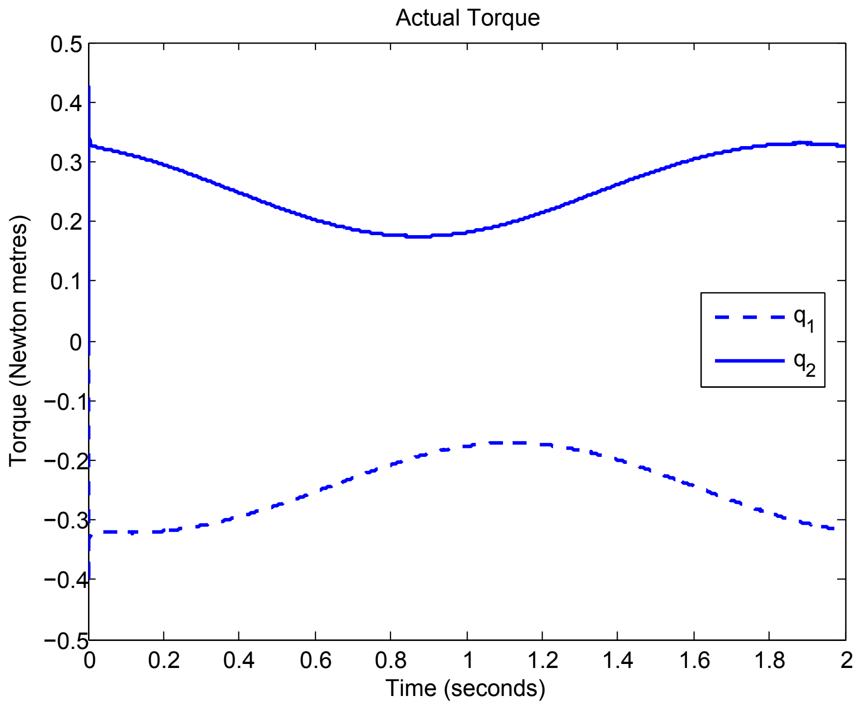


Figure 5.32: Torque for Fuzzy Adaptive Backstepping

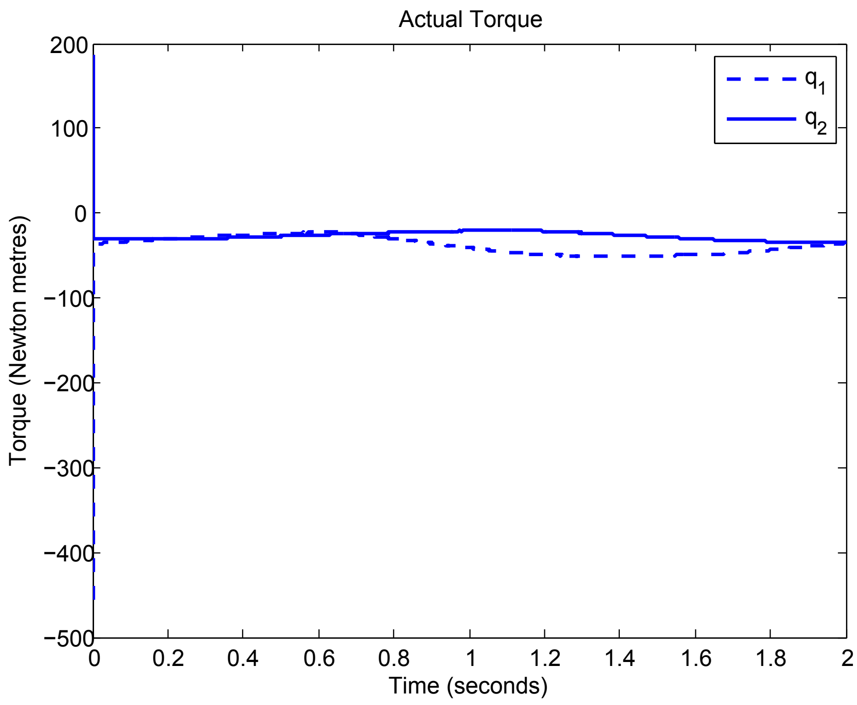


Figure 5.33: Torque for Indirect Adaptive Fuzzy

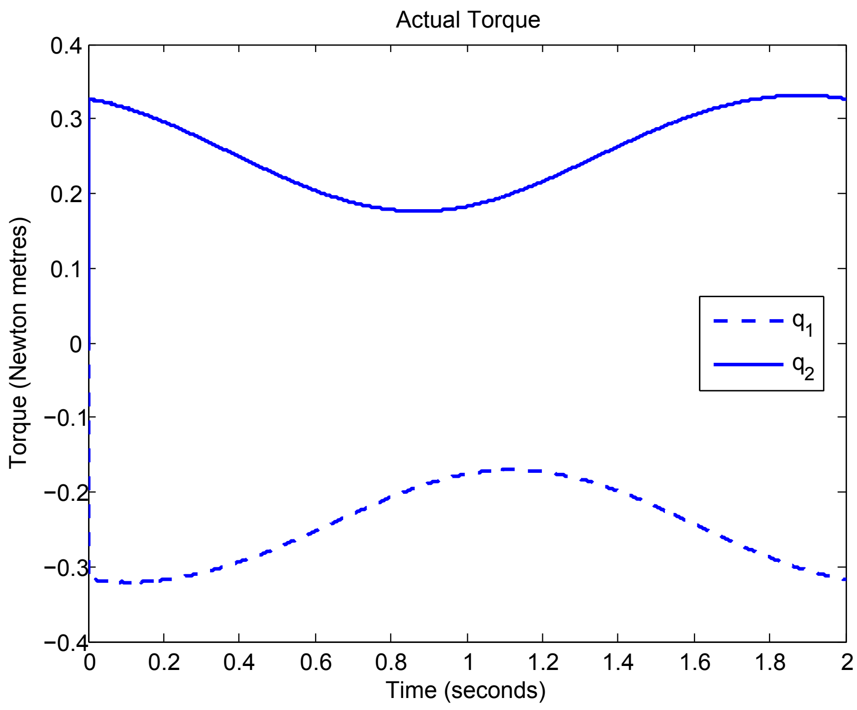


Figure 5.34: Torque for Direct Adaptive Fuzzy

The actual torque figures for most of the fuzzy logic controllers adequately follows the desired torque. The major exception is the indirect adaptive fuzzy controller. The mechanical model of the parallel robot can achieve a peak torque of 7.5 Newton metres and a maximum continuous torque of 6 Newton metres. Neither of these conditions is satisfied for the indirect adaptive fuzzy controller. The most likely reason for this failure is due to the fact that the indirect adaptive fuzzy controller is not calculated the same way as the rest of the controllers described in this section. The fuzzy PD, direct adaptive fuzzy and fuzzy adaptive backstepping controllers all utilize the system equation defined in (3.3) to solve for the control signals, while the indirect adaptive fuzzy controller estimates the system equation and then solves for the control signals. This system estimation procedure potentially increases the overall system error, thus translating to a much larger torque value than can be allotted.

The fuzzy adaptive backstepping controller holds the most promise going forward in determining the experimental results on the mechanical model of the planar two degrees of freedom parallel robot. The following chapter will discuss the electrical and mechanical design of the physical system.

# Chapter 6

# Parallel Robot Design

This chapter will describe the electrical and mechanical design of the two degrees of freedom planar parallel robot implemented to determine the controller performance in the experimental operation of the system.

## 6.1 Electrical Design

The two degrees of freedom parallel robot described in this thesis utilized two distinct printed circuit boards (PCBs) to achieve the desired functionality. These boards are designated: digital signal processing (DSP) and motor driver. The pinout table, schematic diagram and PCB layout for the DSP board and the motor driver board are found in Appendix A and Appendix B, respectively. The block diagram concerning the communication between the two circuit boards, the appropriate host computers and the motors is shown in Figure 6.1. The physical representations of the DSP board and the motor driver board are shown in Figure 6.2 and Figure 6.3, respectively.

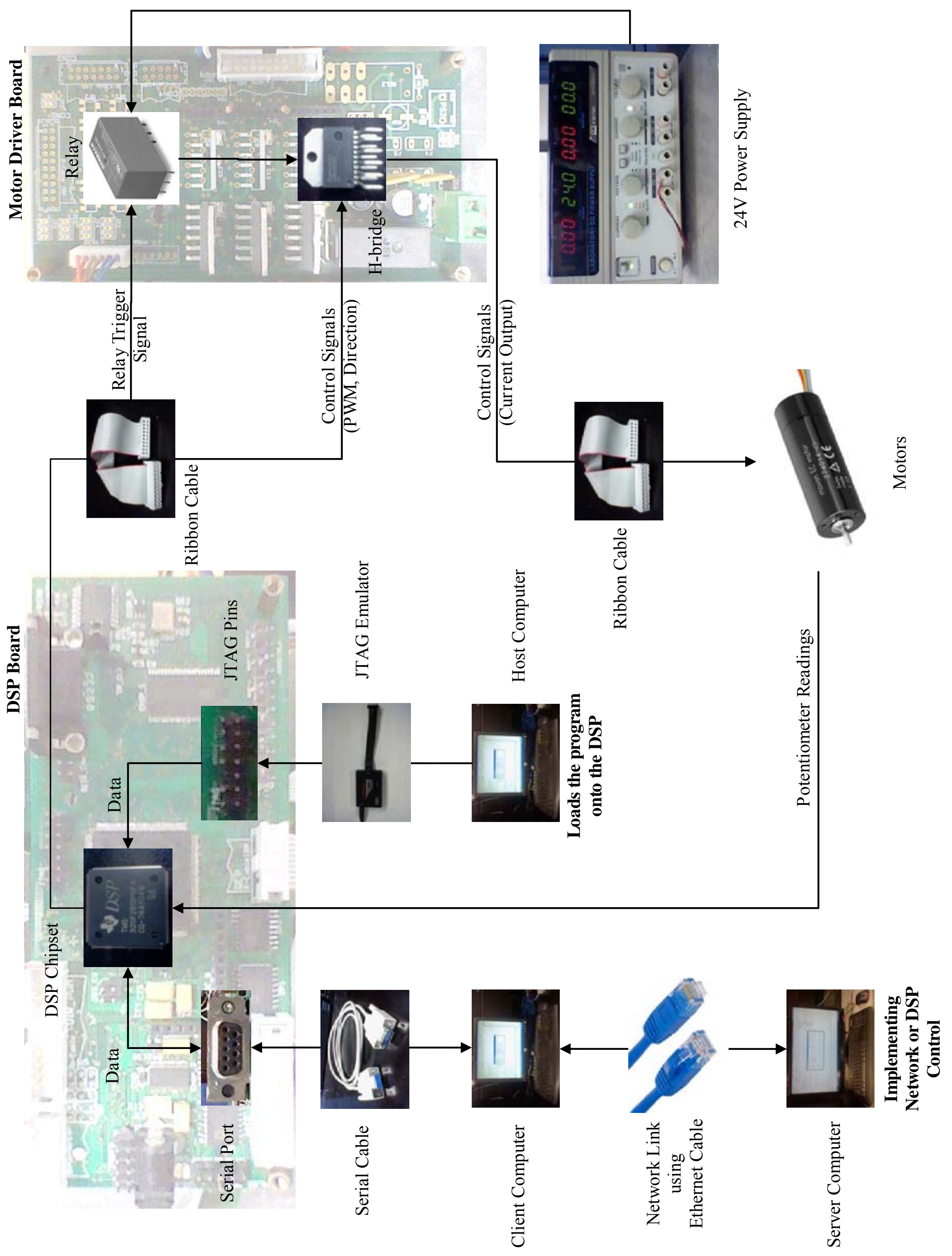


Figure 6.1: Block Diagram for Electrical Design

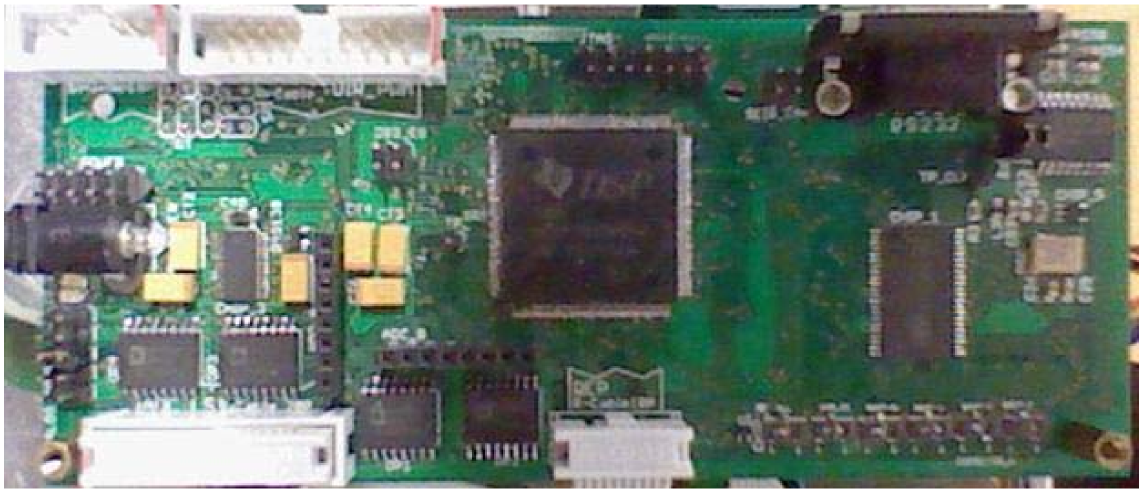


Figure 6.2: DSP Circuit Board Layout

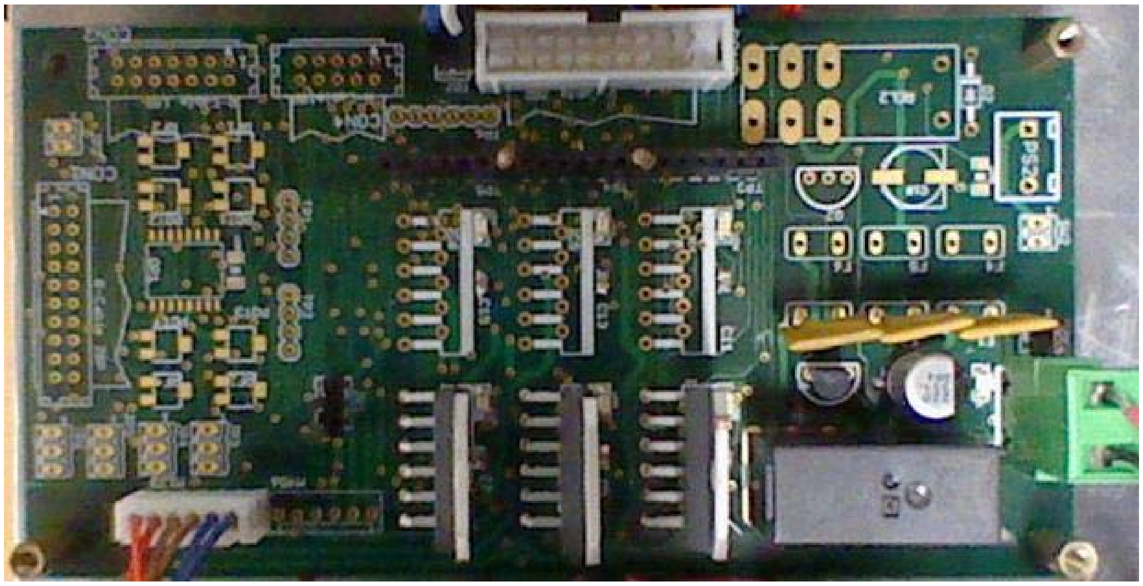


Figure 6.3: Motor Driver Circuit Board Layout

The DSP board is the most crucial component in the entire practical design of the parallel robot. Its purpose is to send the pulse width modulation (PWM) signals to the motor driver board and receive the potentiometer readings from the motors in order to determine the actual location of the actuated joints at any given time. The main chipset of the DSP board is the TMS320F2812 digital signal processor manufactured by Texas Instruments. The TMS320F2812 was programmed using Code Composer Studio through a joint test action group (JTAG) emulator connected to the JTAG pins found on the DSP board. The programmer sent instructions to the TMS320F2812 concerning the appropriate general purpose input/output (GPIO) pins that would be used to send or receive the desired data. These pinout relationships can be found in Table A.1 and Table A.2 in Appendix A. Regardless of the control technique, the server computer would calculate the desired position, velocity and acceleration of the joints in order to determine the desired trajectory of the end effector. It would send this data along with the designated controller gains to the client computer, which in turn would send all the data to the DSP board. If the parallel robot was implementing DSP control, the DSP chipset would calculate the controller output based on the potentiometer readings and send the PWM signals to the motor driver board. The DSP board would receive the potentiometer readings and send this data back to the server computer to be analyzed. On the other hand, if the parallel robot implemented network control, the server computer would calculate the controller output based on the potentiometer feedback signals from the DSP. It would relay the PWM signals to the client computer followed by the DSP board, which in turn would relay this data to the motor driver board. The potentiometer readings generated by this procedure would be available for analysis on the server computer. For both control techniques, the server computer would relay the data over the network to the client computer, followed by the transmission of the data through the serial cable to the DSP board. The data from the serial cable was processed by the MAXIM multichannel RS-232 driver/receiver. The server and client computer both use Visual Studio .NET 2003 to code the controller design and the serial communication protocol, respectively. The DSP board is not directly responsible in actuating the motors of the parallel robots. This procedure was handled by sending the specific GPIO signals to the motor driver board.

The sole purpose of the motor driver board is to actuate the motors in order for them to achieve their desired trajectory. The motor driver board employs the LMD18200 H-bridge chipset - manufactured by National Semiconductor - for each motor. Using the PWM and direction signals generated by the DSP board, the motor driver board can appropriately drive the motors in the desired direction. For the safety of the motors and the mechanical structure of the two degrees of freedom parallel robot, the DSP board must trigger a one-poled relay on the motor driver board to send the power to actuate the motors. The pinout relationship concerning the transmission of data throughout the motor driver board can be found in Table B.1 in Appendix B.

The links of this parallel robot are driven by two 270994 motors manufactured by Maxon Motor. The motors have a no load speed of 8780 revolutions per minute. The gear ratio ($G_R$) is 246 while the efficiency is slated at 0.87 due to the motor ball bearings and the gear trains. The back electromotive force constant ($K_V$) is 0.0258 volts per radian per second and the armature resistance ($R_a$) is rated at 2.36 ohms. The maximum continuous torque that the motor can handle is 6 Newton metres; hence the system is limited to this critical constraint while trying to achieve the desired trajectory of the system. The source voltage ($V_S$) of the system is 24 volts.

The DSP board sends PWM signals to the motor driver board; therefore it is important to convert the output generated by the controllers into a format that is understandable by the DSP board. This can be done by converting output signal into a duty ratio. Let Figure 6.4 approximate the equivalent circuit design of the permanent magnet DC motor:

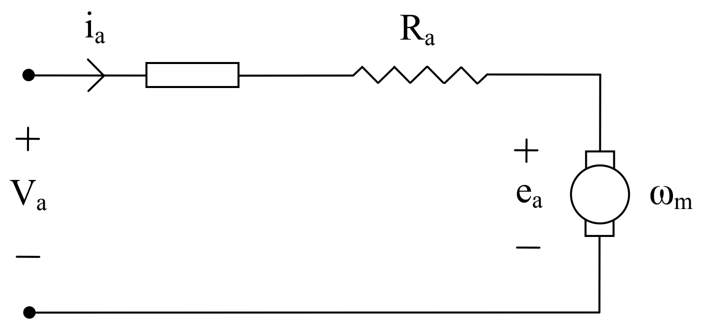


Figure 6.4: Equivalent Circuit of the Permanent Magnet DC Motor

where: $V_a$ is the armature voltage, $i_a$ is the armature current, $\omega_m$ is the angular velocity and $e_a$ is the motor voltage. The goal is to find the duty ratio ($D_R$) which is defined as:

$$D_R = V_a/V_S \tag{6.1}$$

To accomplish this, the following formula can be defined based on the electrical design of the motor:

$$V_a = i_a R_a + K_V \omega_m \tag{6.2}$$

By rearranging equation (6.2) and solving for $i_a$, the following can be obtained:

$$i_a = \frac{-K_V \omega_m + V_a}{R_a} \tag{6.3}$$

The angular velocity is defined as: $\omega_m = G_R q$. The real torque, which is equal to $(u)$, is defined as: $\tau = G_R \tau_m$, with the motor torque defined as: $\tau_m = K_V i_a$. Therefore, the real torque can be rewritten as:

$$u = G_R K_V i_a \tag{6.4}$$

By substituting equation (6.3) into equation (6.4) and isolating for $V_a$, it is possible to achieve:

$$V_a = \frac{R_a u + q(K_V G_R)^2}{K_V G_R} \tag{6.5}$$

All the variables are known at any given point except for the real torque $(u)$ which is generated differently for each controller. After substituting all the known variables into equation (6.5) and solving for the armature voltage, it is possible to determine the corresponding duty ratio.

## 6.2 Mechanical Design

The mechanical design of this planar two degrees of freedom parallel robot was drafted utilizing Solidworks 2009. Figure 6.5 portrays this representation in detail.

The entire structure of the parallel robot is comprised of aluminium angles, which were employed due to their robustness and lightweight attributes. The parameters of each link are found in Table 2.1. The physical representation of the overall system are shown in Figure 6.6 and Figure 6.7.

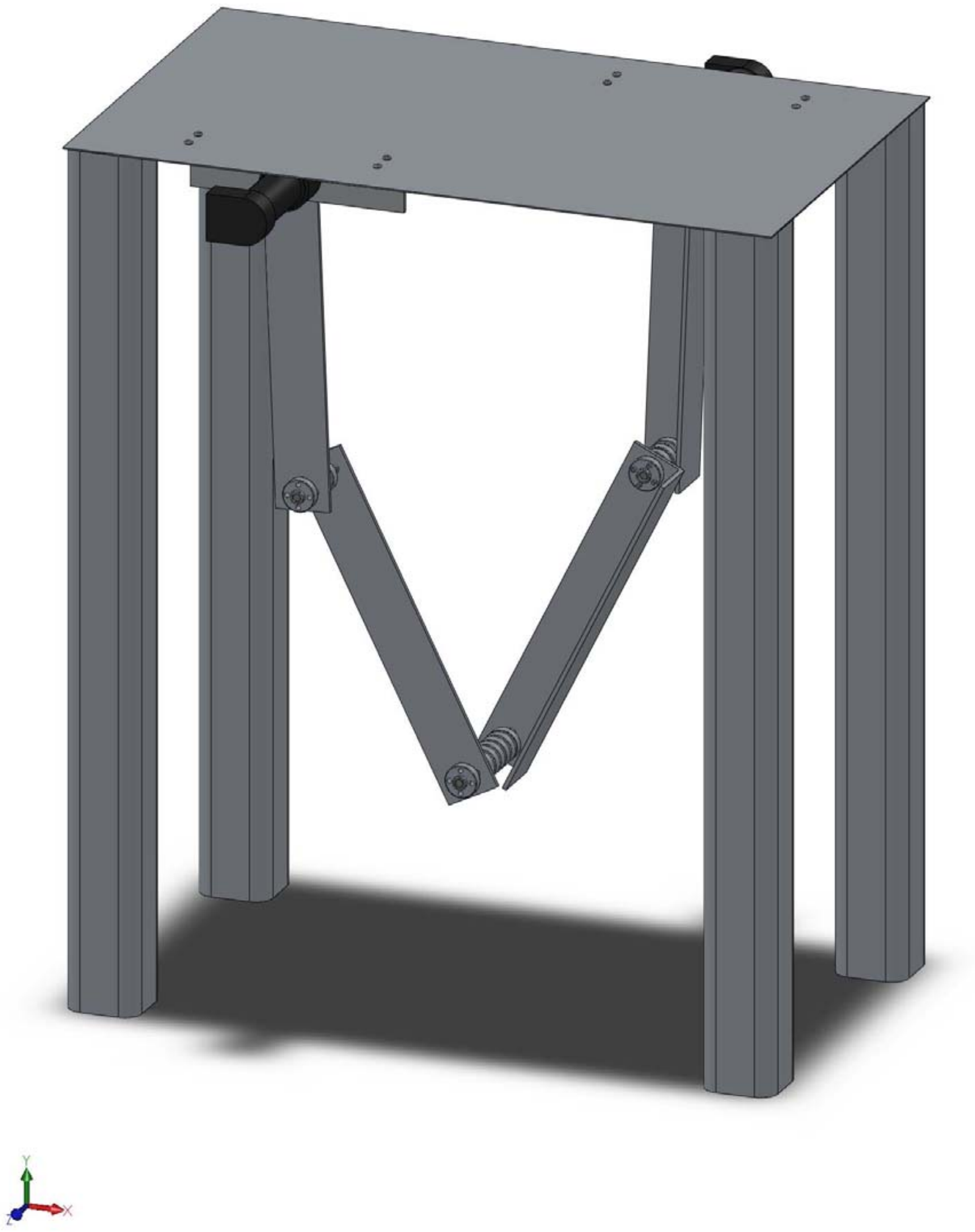


Figure 6.5: Solidworks Model of the Two Degrees of Freedom Parallel Robot

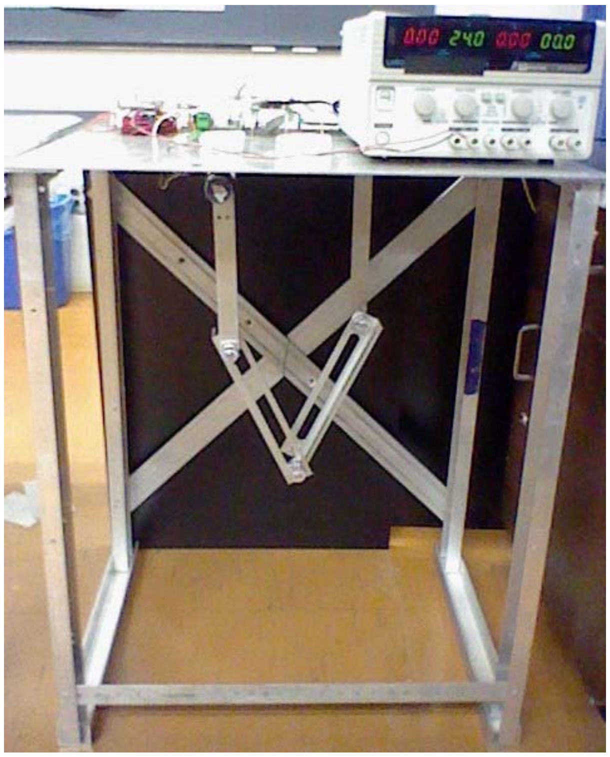


Figure 6.6: Physical Structure of the Two Degrees of Freedom Parallel Robot

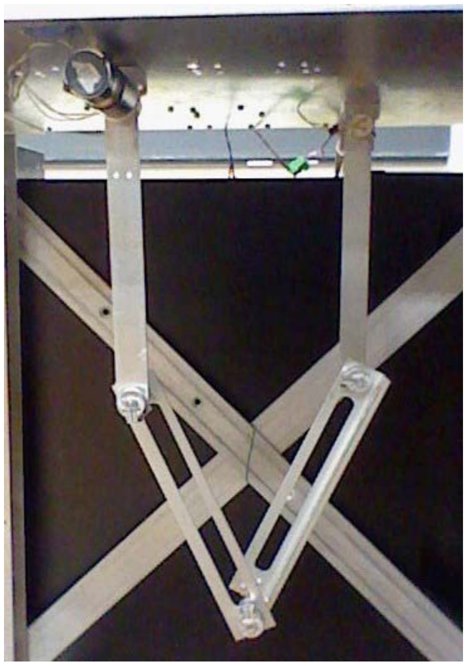


Figure 6.7: Close-up View of the Two Degrees of Freedom Parallel Robot

# Chapter 7

# Controller Experimentation

Each controller discussed in this chapter will contain the experimentation results for: the desired and actual actuated joint angles along with their respective error, the location of the desired and actual end effector trajectory in Cartesian space along with their respective positional output error, the control signal computation time and the overall transmission time, and the PWM signals required to achieve the actual results. A final conclusion will then be drawn based on the pros and cons of each control technique.

## 7.1 Trajectory Generation

The end effector of the two degrees of freedom parallel robot was programmed to follow a circular trajectory using the network control interface. This allowed for the programming of the parallel robot to occur on a server computer, which would transmit the controller output over the Lakehead University network to a client computer that was serially connected to the DSP board of the parallel robot. The block diagram of this process is shown in Figure 6.1. A detailed explanation of the operation of the network control technique used on the planar two degrees of freedom parallel robot described in this thesis can be found in [61]. The DSP chipset is limited in the amount of memory that it could allocate for a specified program; hence the following figures will portray the results of the eight controllers based on the network control technique.

The simulation of the end effector trajectory was conducted strictly by performing the circular trajectory once and gathering the desired results. This was clearly not attainable in the experimentation of the parallel robot mechanism since the initial position of the end effector varied at any given trial. To compensate for the unknown initial angles of the actuators, a potentiometer was attached to each motor shaft. The DSP board read the voltage readings from the potentiometers in order to conclusively determine the joint angle at any given point of the trajectory tracking. Another issue was the fact that the end effector had to reach the location of the desired circular trajectory, which was on average three centimetres higher in the y-axis than its initial position. Instead of linearly translating the end effector to reach its initial position on the circular trajectory, it was found that by programming the end effector to follow a progressive arc path, the amount of oscillation present in the system operation could be reduced. The end effector initially performed a progressive arc path until it reached the desired location to track the circular trajectory. Then, it performed two rotations around the desired circular trajectory and proceeded on a progressive arc path until $q_1$ and $q_2$ were perpendicular to the x-axis. It should be noted that the feedback signals from the potentiometers generated noisy results; hence a second order Butterworth filter with a cut-off frequency of 3 Hertz was programmed to filter the noise from the feedback signals.

The tracking speed utilized is defined by the angular velocity formula: $\omega = 2\pi f$, where $f$ is the tracking frequency of the end effector. The location of the circular trajectory is based on the coordinate system defined in Figure 2.2. In these experimentation results, the origin of the circle based on the Cartesian coordinate system in metres is defined as (0.1059, -0.3769). The radius of the circular trajectory is 0.03 metres and the frequency implemented is 0.5 Hertz. It should be noted that the trajectory defined in this report never impedes or approaches any singular point.

## 7.2 Controller Gains

The gains of each controller were determined utilizing the gains obtained in the simulation results as a reference and then tuning them by trial and error to achieve the figures garnering the most desirable results. Table 7.1 will list the gains utilized for the non-adaptive and adaptive PD and backstepping controllers, while Table 7.2 will list the gains utilized for the fuzzy PD, indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers.

| *Non Fuzzy Logic Controllers* | $K_{P1}$ | $K_{P2}$ | $K_{D1}$ | $K_{D2}$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $\Gamma$ | $e_{ff}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Non-Adaptive PD | 800 | 850 | 74 | 75 | - | - | - | - | - | 0.035 |
| Adaptive PD | 800 | 850 | 74 | 75 | - | - | - | - | 100 | 0.035 |
| Non-Adaptive Backstepping | - | - | - | - | 75 | 45 | 11 | 21 | - | 0.35 |
| Adaptive Backstepping | - | - | - | - | 75 | 45 | 11 | 21 | 20 | 0.35 |

Table 7.1: Non-Fuzzy Logic Controller Gains

| *Fuzzy Logic Controllers* | $k_1$ | $k_2$ | $c_1$ | $c_2$ | $c_3$ | $c_4$ | $b_1$ | $b_2$ | $\Gamma$ | $e_{ff}$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Indirect Adaptive Fuzzy | 95 | 7 | - | - | - | - | - | - | 10 | 0.35 |
| Direct Adaptive Fuzzy | 100000 | 5000 | - | - | - | - | - | - | 0.01 | 0.175 |
| Fuzzy Adaptive Backstepping | - | - | 5 | 5 | 120 | 140 | 0.5 | 0.5 | 0.01 | 0.35 |

| *Fuzzy Logic Controllers* | $c_{out1}$ | $c_{out2}$ | $c_{e1}$ | $c_{e2}$ | $c_{\dot{e}1}$ | $c_{\dot{e}2}$ | $e_{ff}$ |
|---|---|---|---|---|---|---|---|
| Fuzzy PD | 870 | 970 | 2.5 | 2 | 2 | 2.5 | 0.35 |

Table 7.2: Fuzzy Logic Controller Gains

The subsequent figures that have been generated utilizing MATLAB will illustrate the experimental data obtained by the implementation of all eight controllers on the two degrees of freedom parallel robot mechanism. The findings will be split up into two sections, namely: non-fuzzy logic controller experimentations and fuzzy logic controller experimentations.

It should be noted that $e_{ff}$ is the percent efficiency of the motors that can potentially decrease the overall system error. By utilizing the duty ratio formula defined in equation (6.1) and dividing it by the percent efficiency, it is possible to achieve a more accurate result. The objective is to obtain a result where the absolute error value is at its minimum. The constant variables $c_{out1}$, $c_{out2}$, $c_{e1}$, $c_{e2}$, $c_{\dot{e}1}$ and $c_{\dot{e}2}$ represent the scaling factors of motor one and motor two, respectively.

## 7.3 Experimentations of Non-Fuzzy Logic Controllers

The experimentation results for all the controllers discussed in this section follow the same circular trajectory regardless of the initial position. Both the adaptive PD and backstepping controllers require an initial value for $\hat{\Theta}$ as shown in Table 5.3. This value is defined using the solutions generated by the formulas in equation (3.15). It should be noted that the value of $\Gamma$ defined in Table 7.1 for the adaptive PD and backstepping controllers applies to all the parameters of $\Gamma$ defined in each respective controller.

### 7.3.1 Joint Angles

The following figures depict the error between the desired and actual joint angles of $q_1$ and $q_2$, respectively. These are the only two angles directly controllable by the actuators of the parallel robot. It is crucial that the difference between the desired angles and the actual angles of $q_1$ and $q_2$ be as small as possible, in order for the end effector error to be minimized.

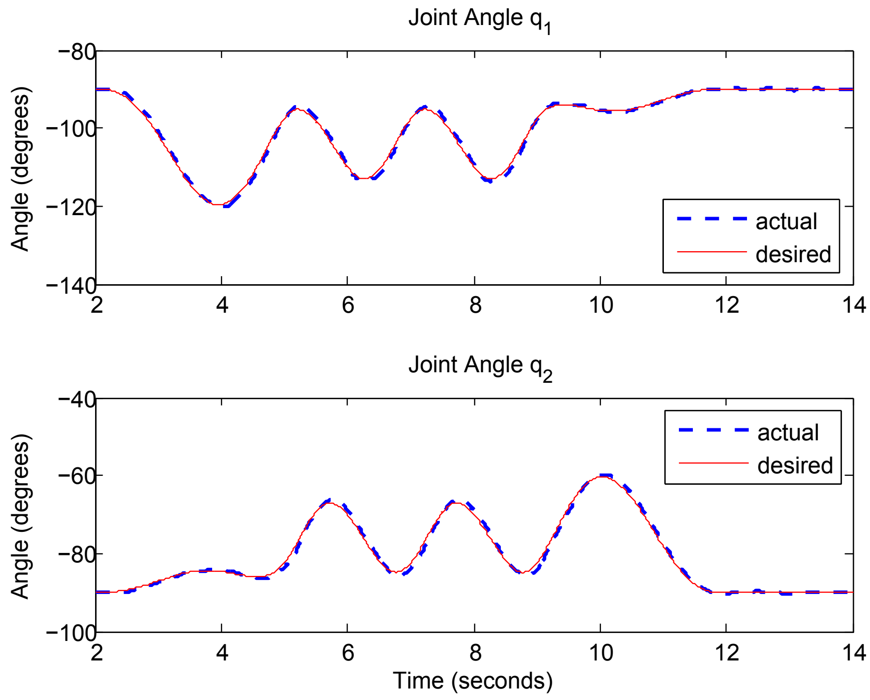


Figure 7.1: Joint Angles for Non-Adaptive PD

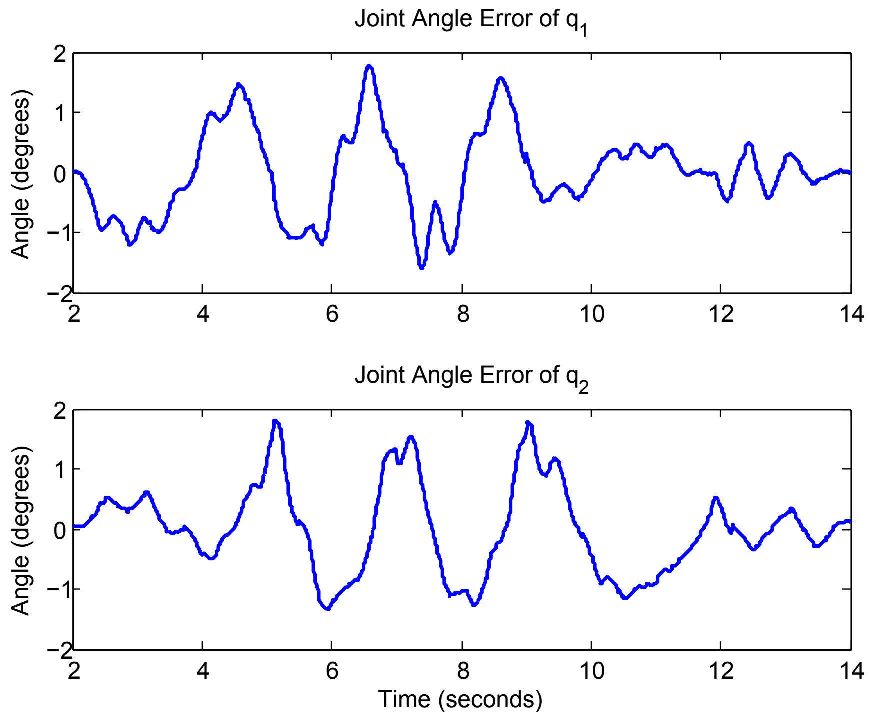


Figure 7.2: Joint Angles Error for Non-Adaptive PD

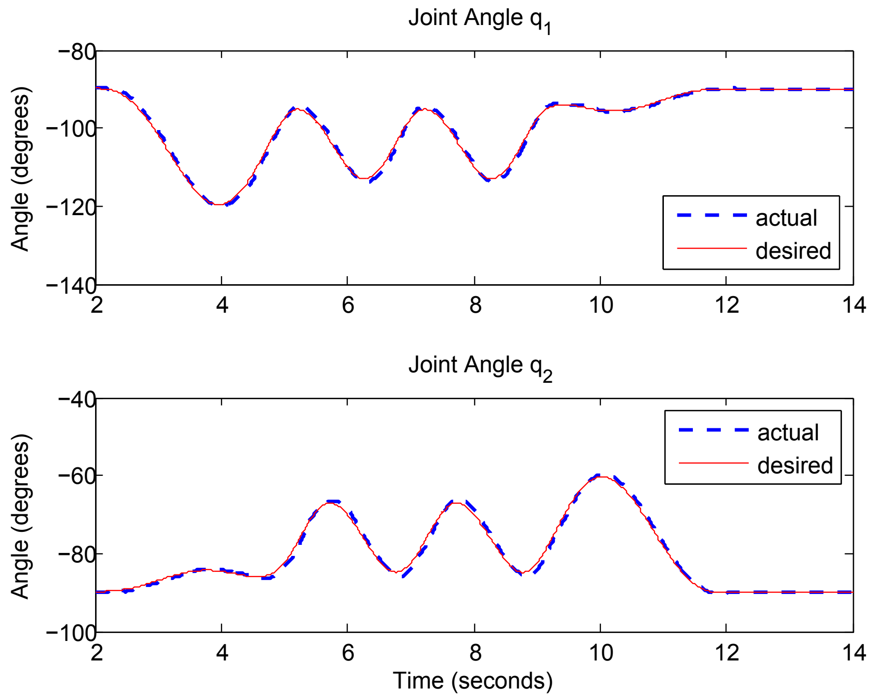


Figure 7.3: Joint Angles for Adaptive PD

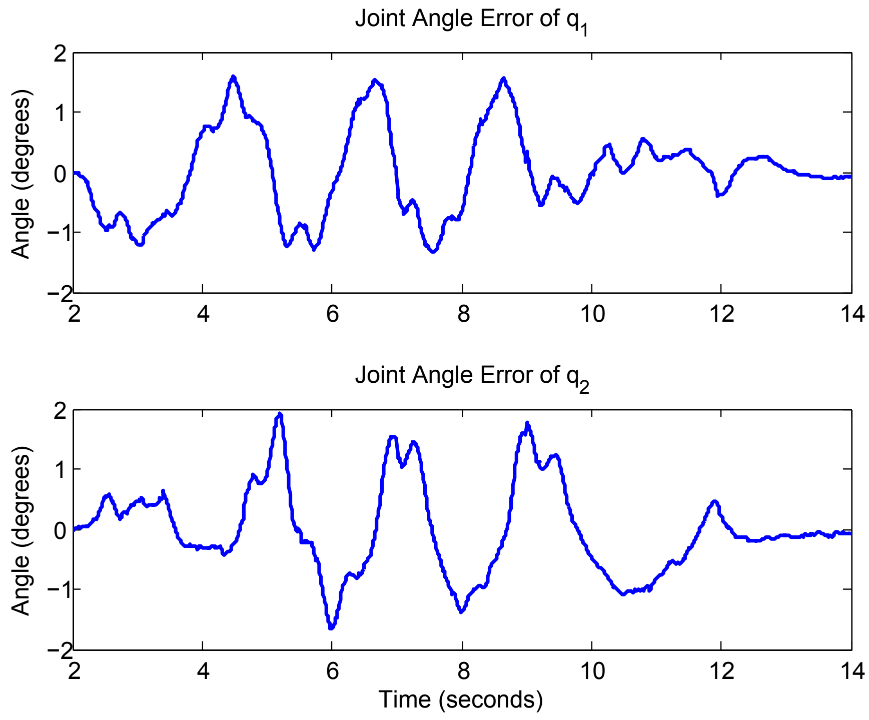


Figure 7.4: Joint Angles Error for Adaptive PD

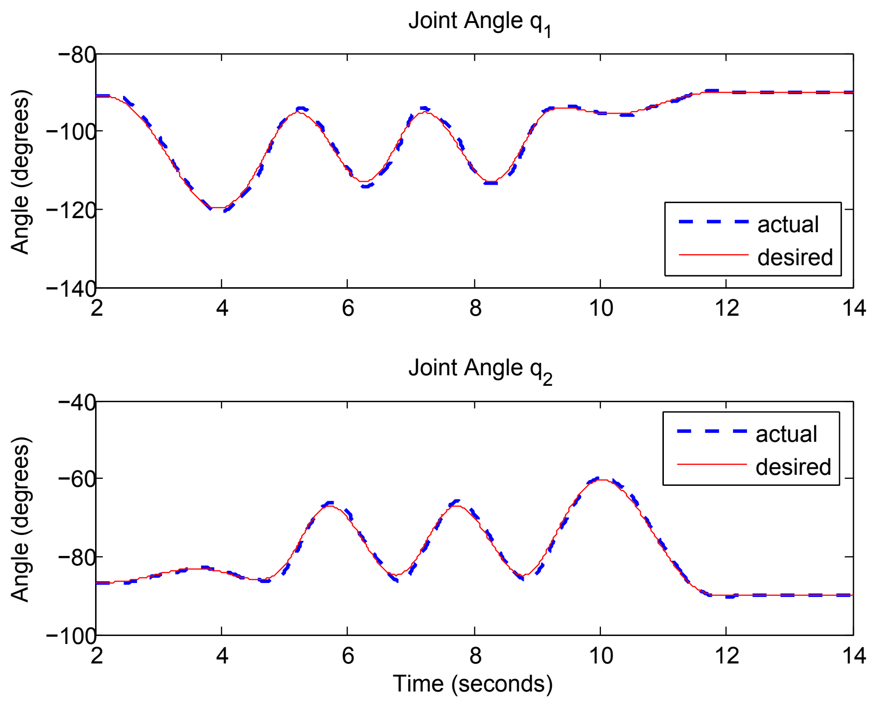


Figure 7.5: Joint Angles for Non-Adaptive Backstepping

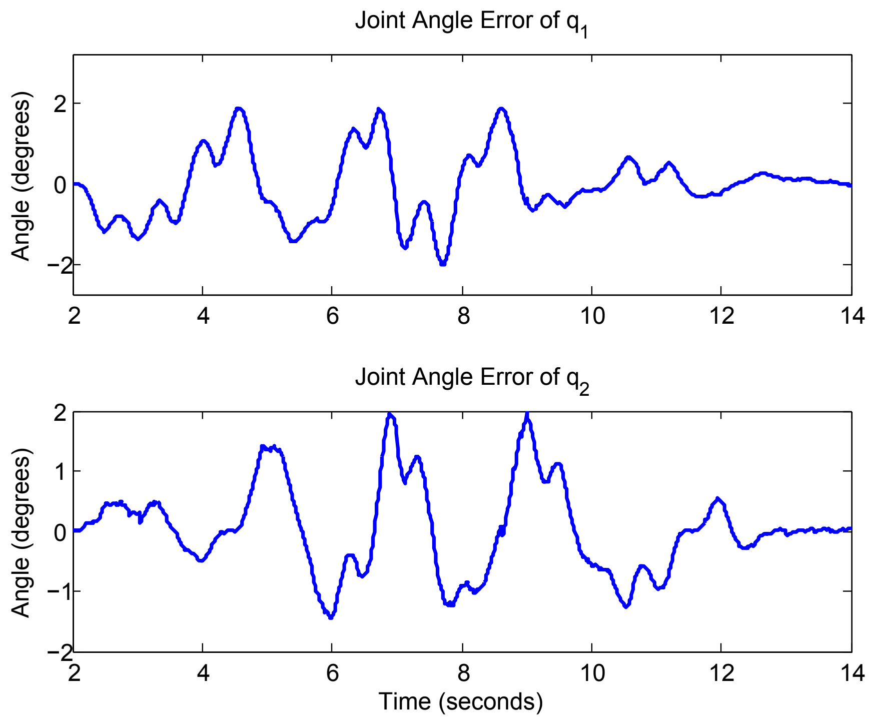


Figure 7.6: Joint Angles Error for Non-Adaptive Backstepping

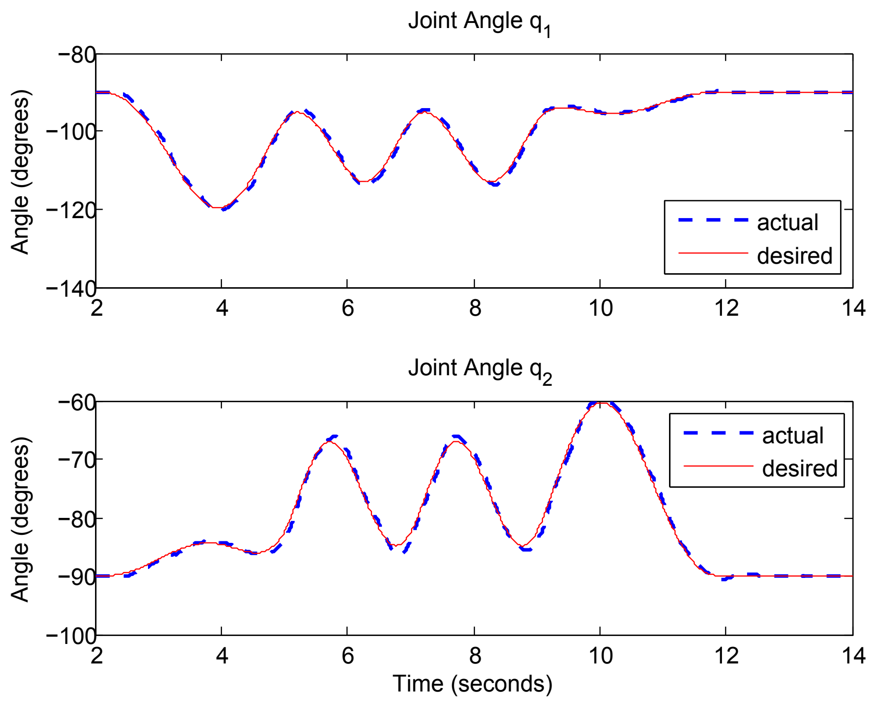


Figure 7.7: Joint Angles for Adaptive Backstepping

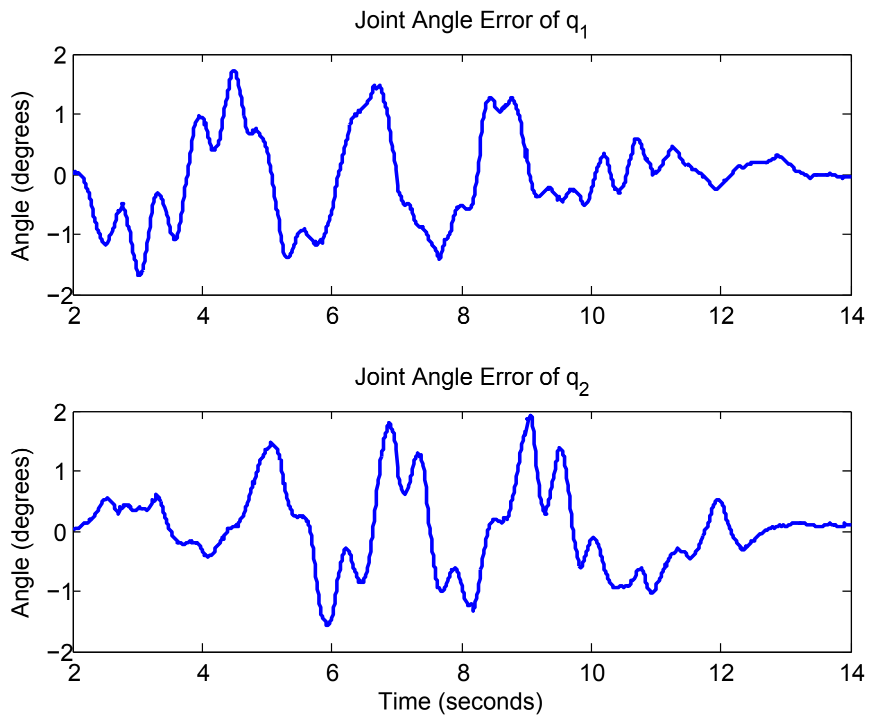


Figure 7.8: Joint Angles Error for Adaptive Backstepping

It is quite difficult to determine the most desirable controller based on the preceding plots, but it is clear that the adaptive PD and backstepping controllers achieved less error than their non-adaptive counterparts. Compared to the non-adaptive controllers, it can be seen that the oscillations in the adaptive controllers are smoother and the trajectory tracking procedure consistently terminates without any oscillations present. The differences between the adaptive PD and backstepping controllers appear to be very minute in this comparison. The peak, mean and RMS joint angle errors in degrees is shown in Table 7.3.

| | *Motor One* | | | *Motor Two* | | |
|---|---|---|---|---|---|---|
| $Non-Fuzzy\ Logic\ Controller$ | **Peak** | **Mean** | **RMS** | **Peak** | **Mean** | **RMS** |
| Non-Adaptive PD | 1.7776 | 0.5017 | 0.6695 | 1.8135 | 0.5136 | 0.6896 |
| Adaptive PD | 1.5929 | 0.4978 | 0.6623 | 1.9261 | 0.5185 | 0.6929 |
| Non-Adaptive Backstepping | 1.7943 | 0.5581 | 0.7138 | 1.9489 | 0.5223 | 0.6883 |
| Adaptive Backstepping | 1.7265 | 0.5462 | 0.7092 | 1.9359 | 0.5118 | 0.6783 |

Table 7.3: Peak, Mean and RMS Non-Fuzzy Logic Controller Joint Angle Errors

### 7.3.2 End Effector Trajectory

The next set of figures portray the trajectory tracking of the end effector along with its respective error between its desired and actual position. In order to determine the actual location of the end effector, equations (2.19) and (2.20) were employed to solve for $q_4$ and $q_3$, respectively. The forward kinematics equations were applied based on all the available data to definitively determine the actual location of the end effector in Cartesian coordinates. The forward kinematics equations utilized for the two degrees of freedom planar parallel robot are based upon the works defined in [68] and [46]. The results from the joint angles of $q_1$ and $q_2$ play a crucial role in the determination of the actual location of the end effector, since a small angular error would not cause a large deviation when compared to the desired trajectory. The plots of the end effector output along with their respective error in the x-axis and y-axis are also shown in order to easily identify the severity of the error.

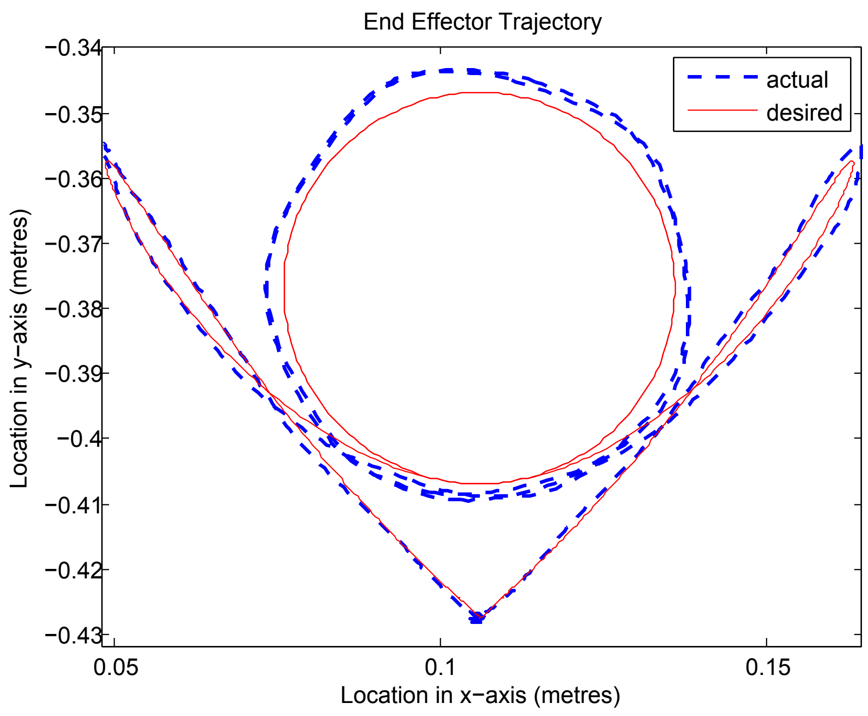


Figure 7.9: End Effector Trajectory for Non-Adaptive PD

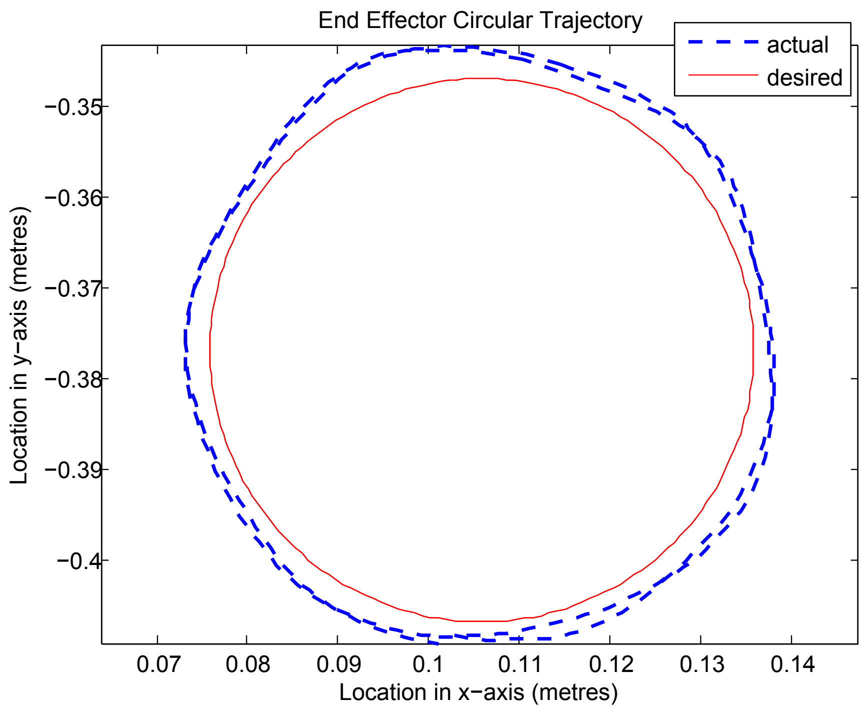


Figure 7.10: End Effector Circular Trajectory for Non-Adaptive PD

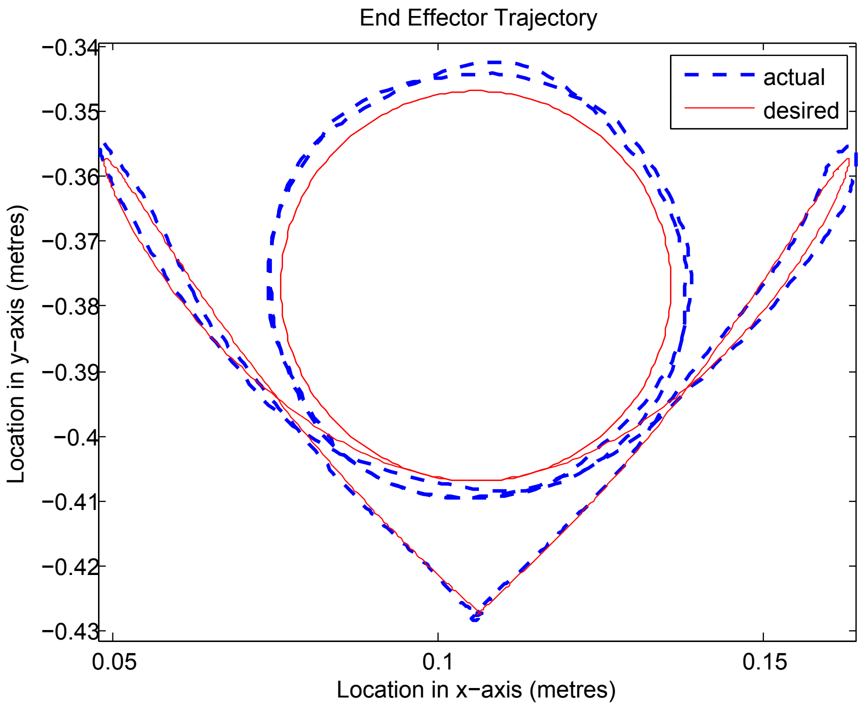


Figure 7.11: End Effector Trajectory for Adaptive PD

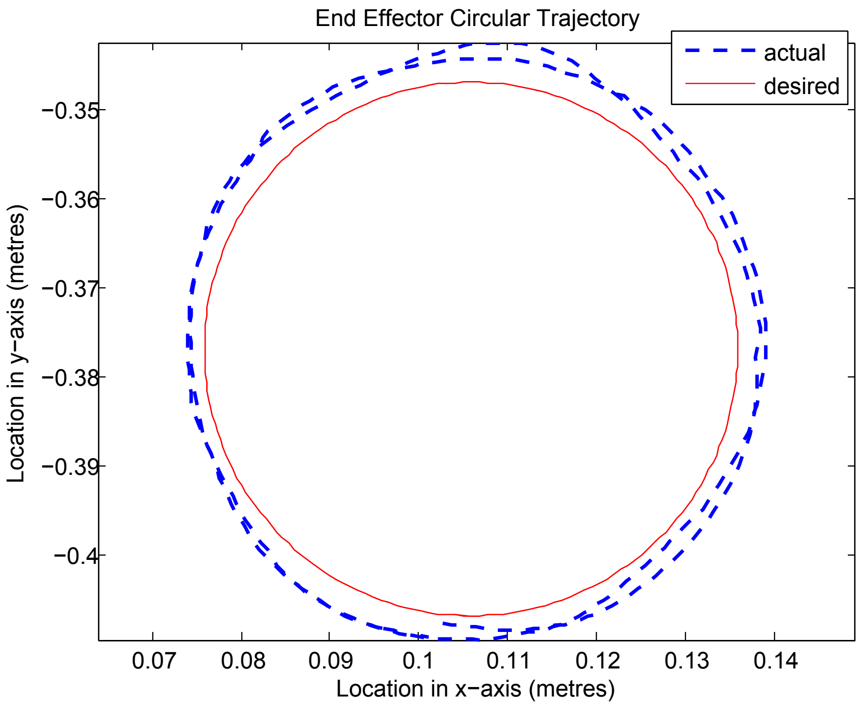


Figure 7.12: End Effector Circular Trajectory for Adaptive PD

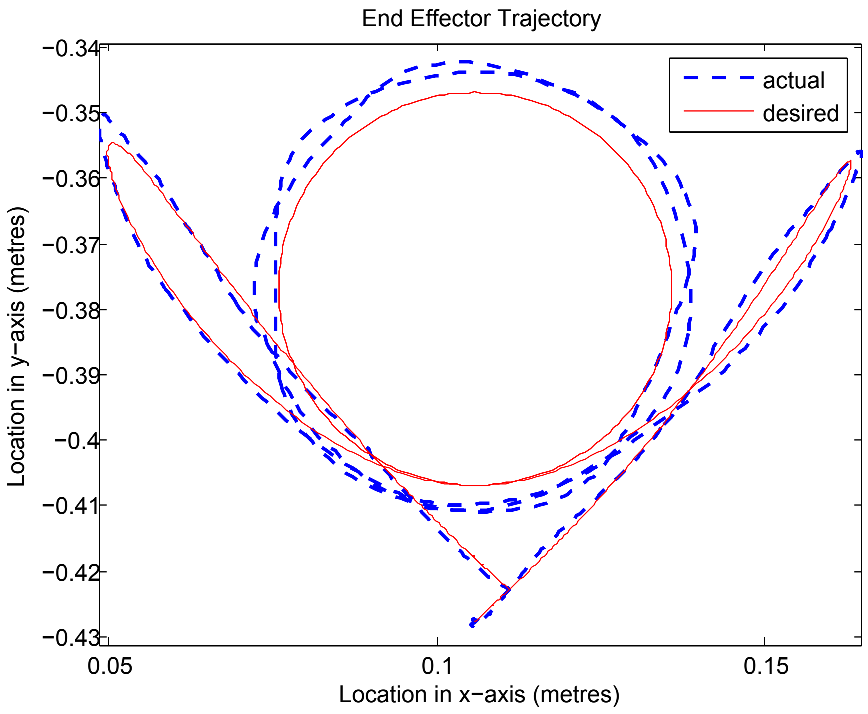


Figure 7.13: End Effector Trajectory for Non-Adaptive Backstepping

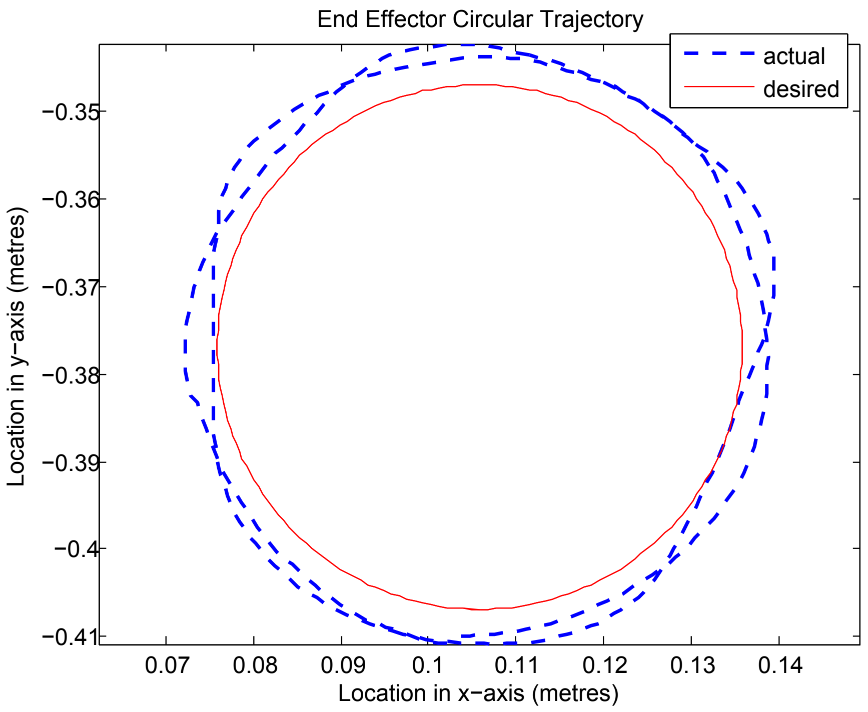


Figure 7.14: End Effector Circular Trajectory for Non-Adaptive Backstepping

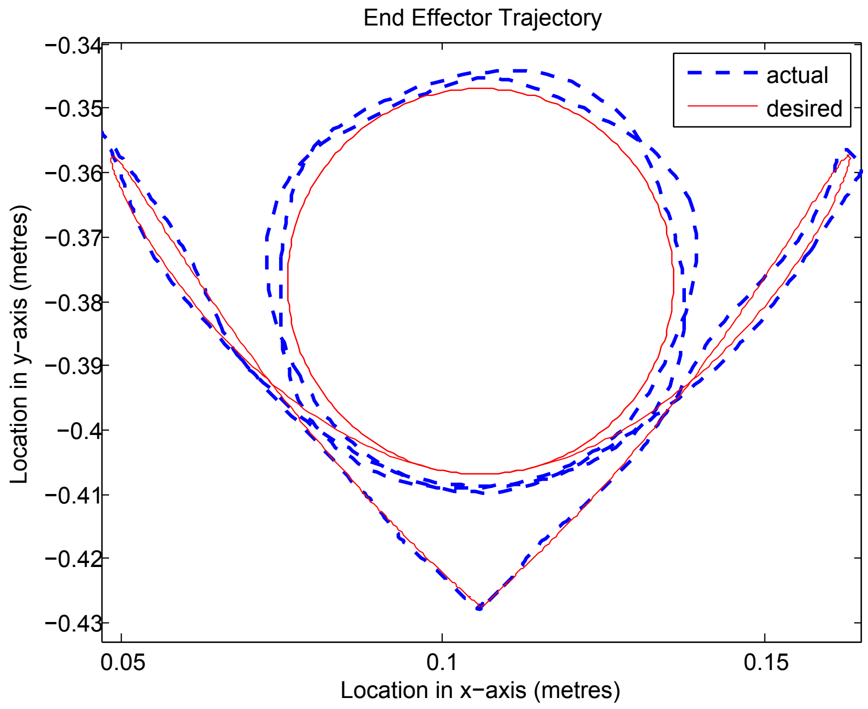


Figure 7.15: End Effector Trajectory for Adaptive Backstepping

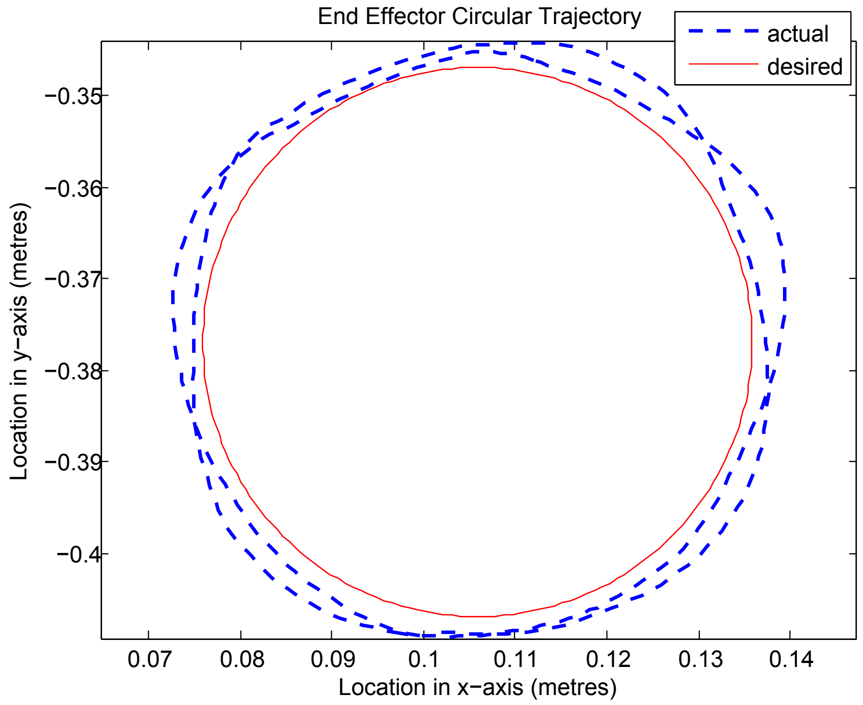


Figure 7.16: End Effector Circular Trajectory for Adaptive Backstepping

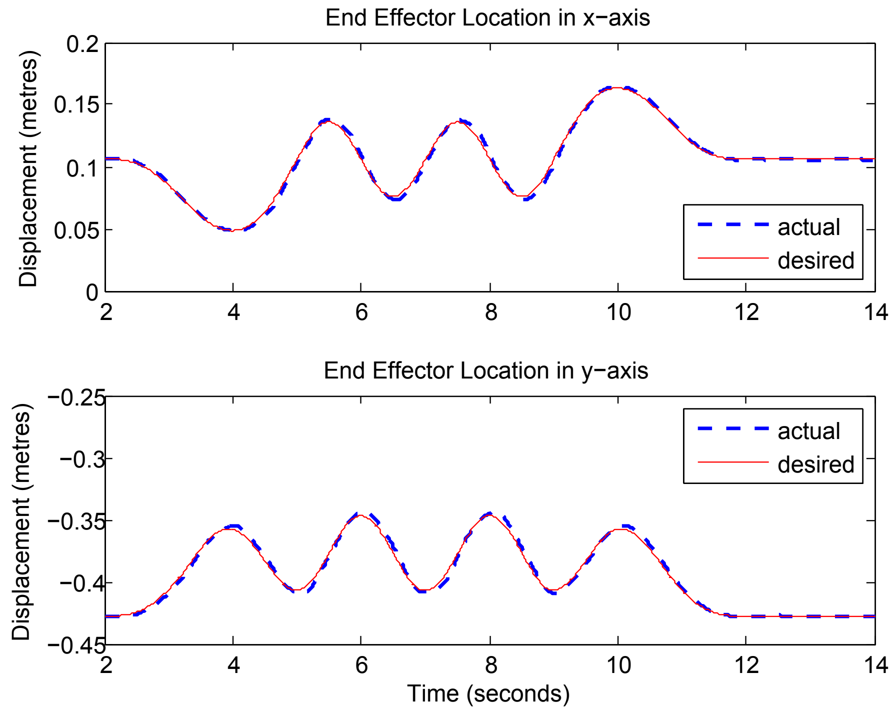


Figure 7.17: End Effector Output for Non-Adaptive PD

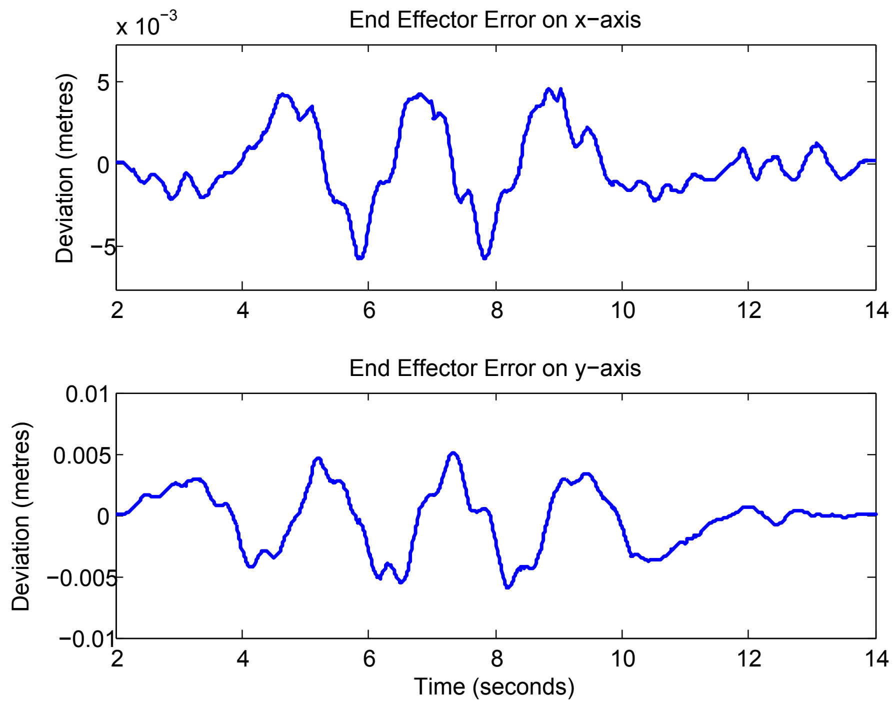


Figure 7.18: End Effector Output Error for Non-Adaptive PD

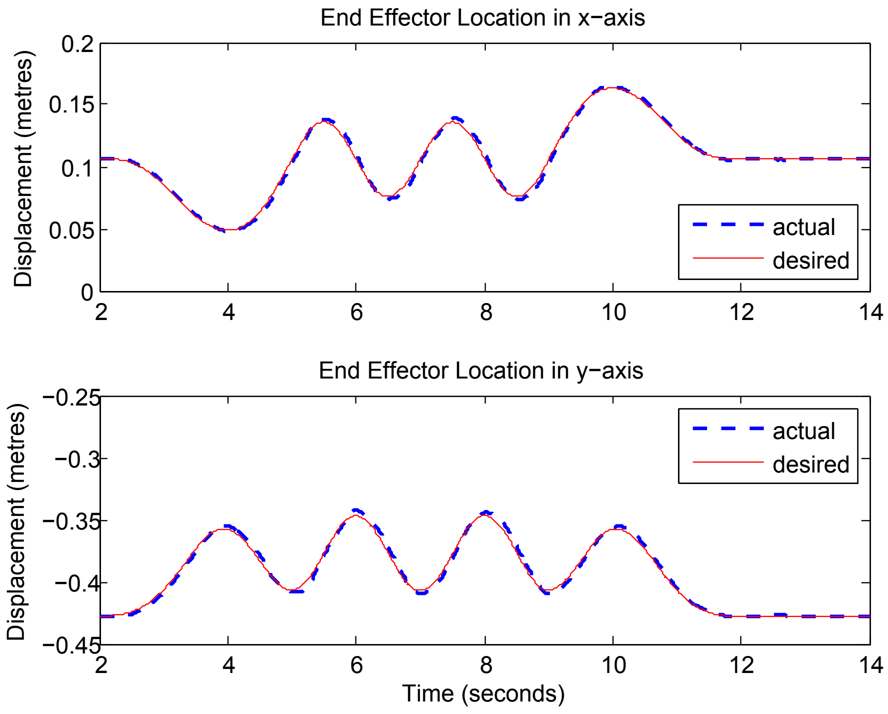


Figure 7.19: End Effector Output for Adaptive PD

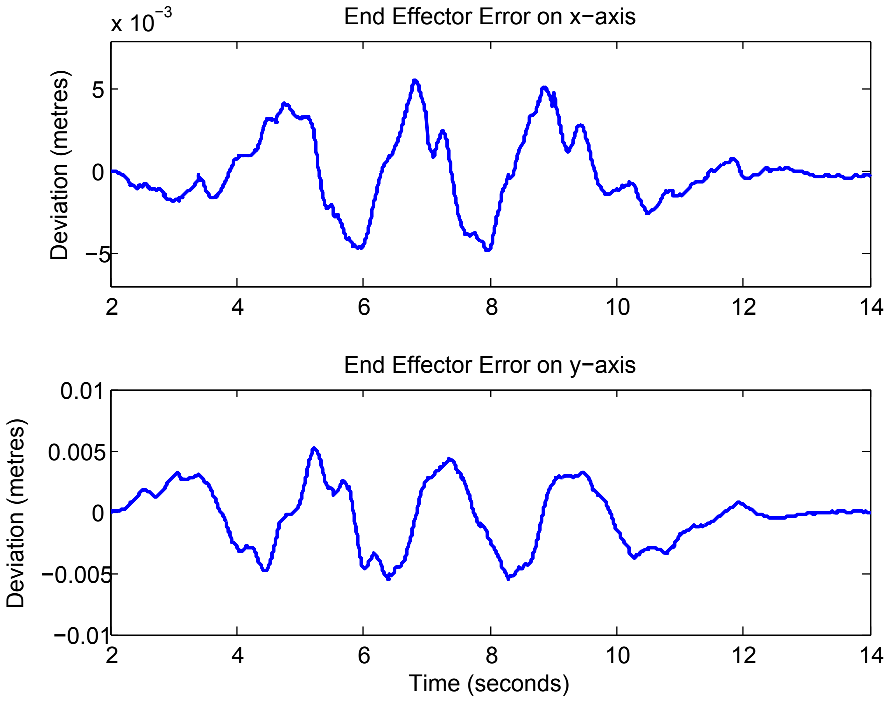


Figure 7.20: End Effector Output Error for Adaptive PD

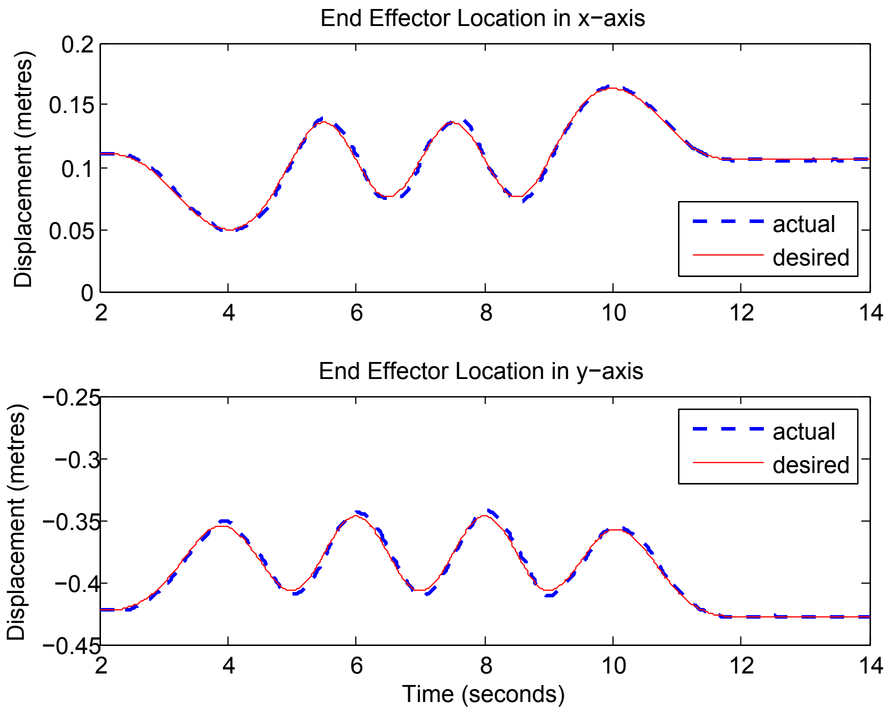


Figure 7.21: End Effector Output for Non-Adaptive Backstepping

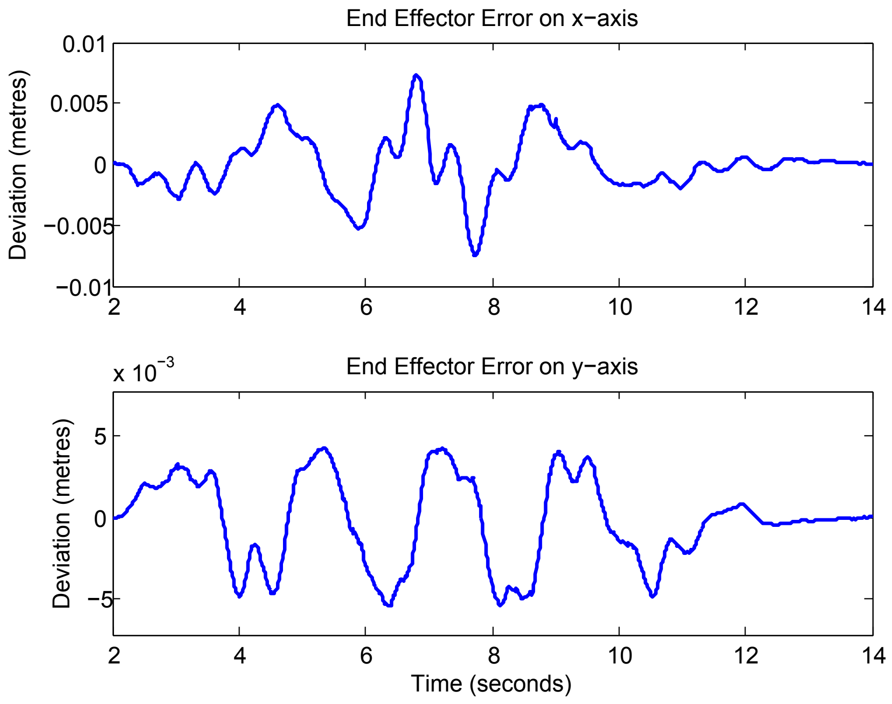


Figure 7.22: End Effector Output Error for Non-Adaptive Backstepping

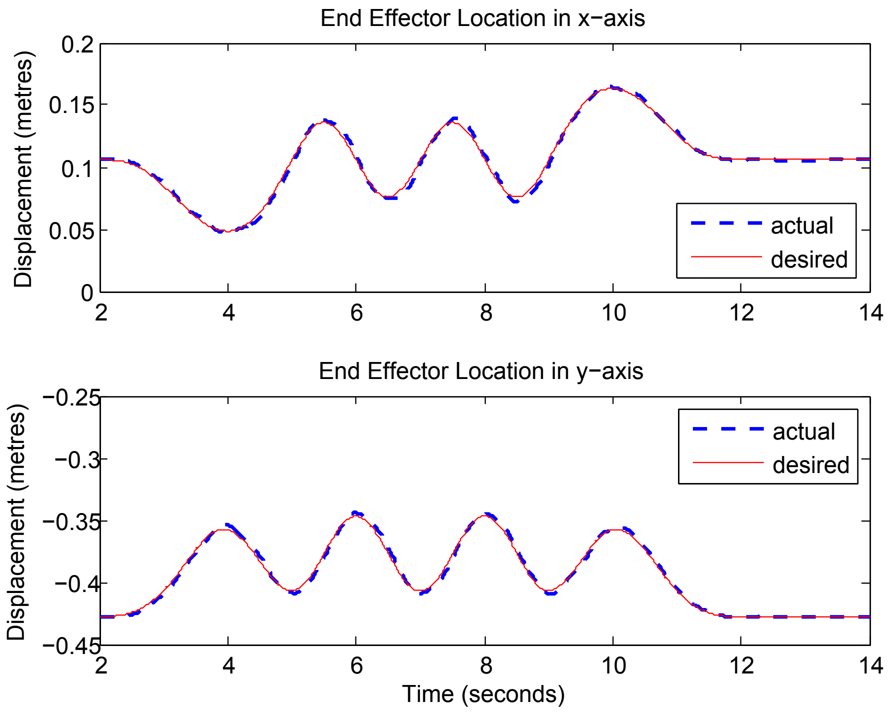


Figure 7.23: End Effector Output for Adaptive Backstepping

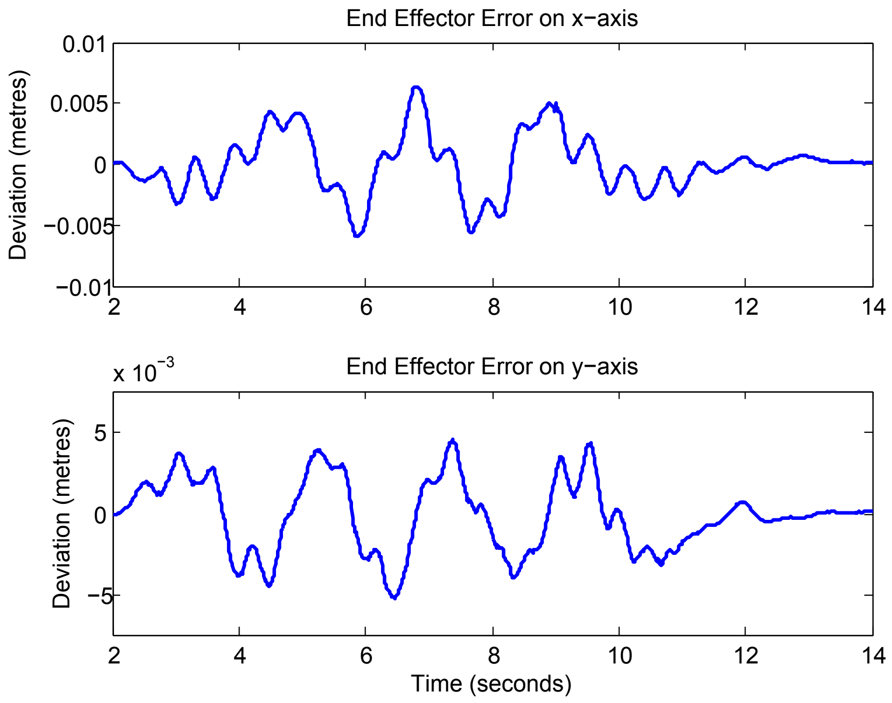


Figure 7.24: End Effector Output Error for Adaptive Backstepping

Similar to the previous observation, the adaptive PD and backstepping controllers procured the most desirable end effector tracking. On average, the end effector output error figures for all four controllers are situated within an error range of 0.006 metres for the x-axis and y-axis. This makes it very difficult to reach a conclusion based on these plots alone; hence, it is much easier to notice the differences between the circular end effector trajectory figures. The most impressive trajectory tracking controller is the adaptive PD controller. It achieves an acutely symmetrical circle without conceding a large displacement error. The adaptive backstepping controller is also quite proficient due to the low displacement error, yet the symmetry of the circle is not as desirable as the adaptive PD controller.

### 7.3.3 Computation and Transmission Time

The following plots portray the computation time, which is defined as the time for the server computer to calculate the control signals. There are also figures for the transmission time, which is defined as the time between the moment at which the DSP board sends the feedback signals over the Lakehead University network to the server and the moment at which the server receives this data plus the time between the moment at which the server sends the control signals to the DSP board over the Lakehead University network and the moment at which the DSP board receives this data.

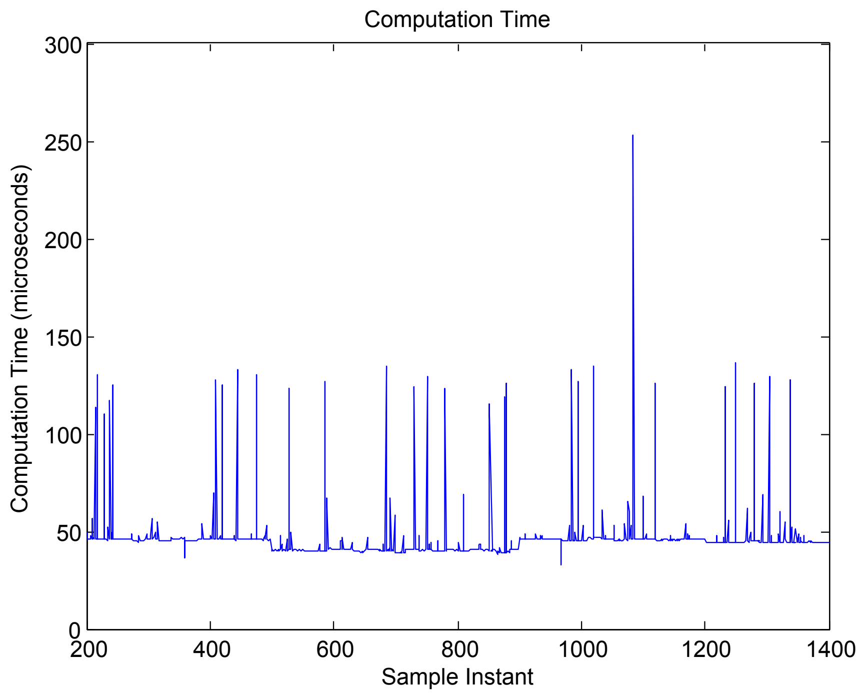


Figure 7.25: Computation Time for Non-Adaptive PD

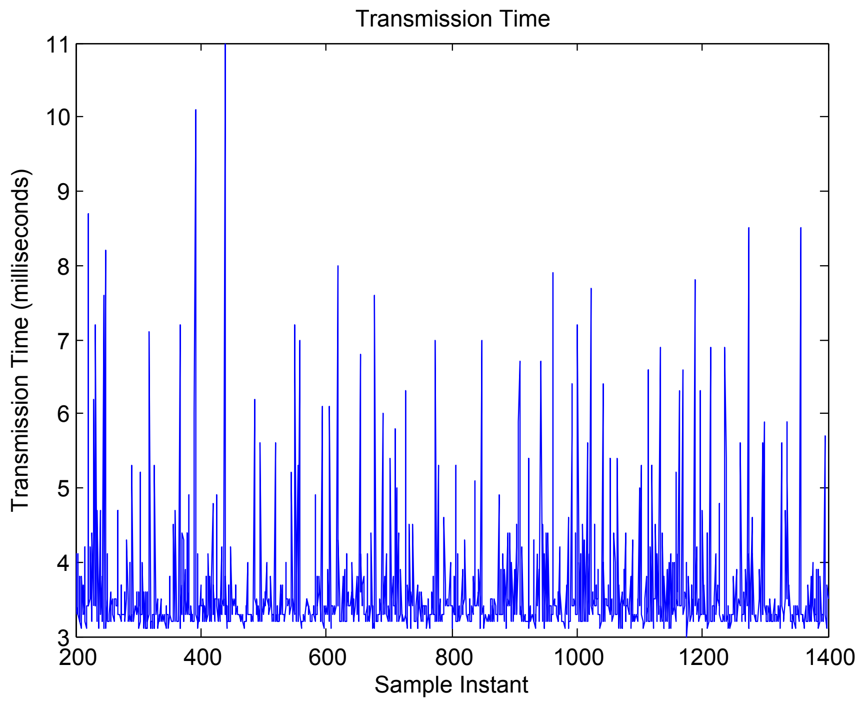


Figure 7.26: Transmission Time for Non-Adaptive PD

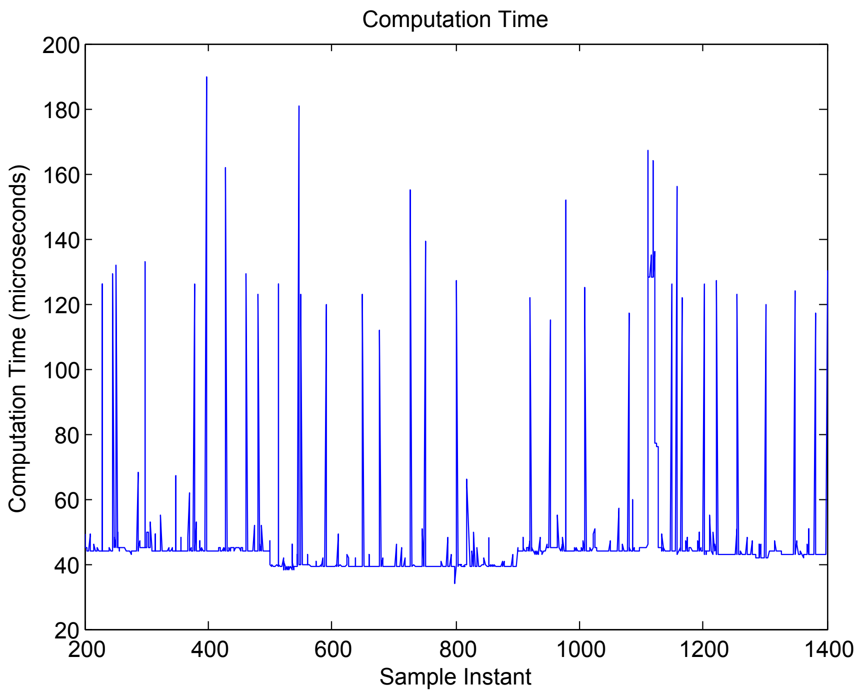


Figure 7.27: Computation Time for Adaptive PD

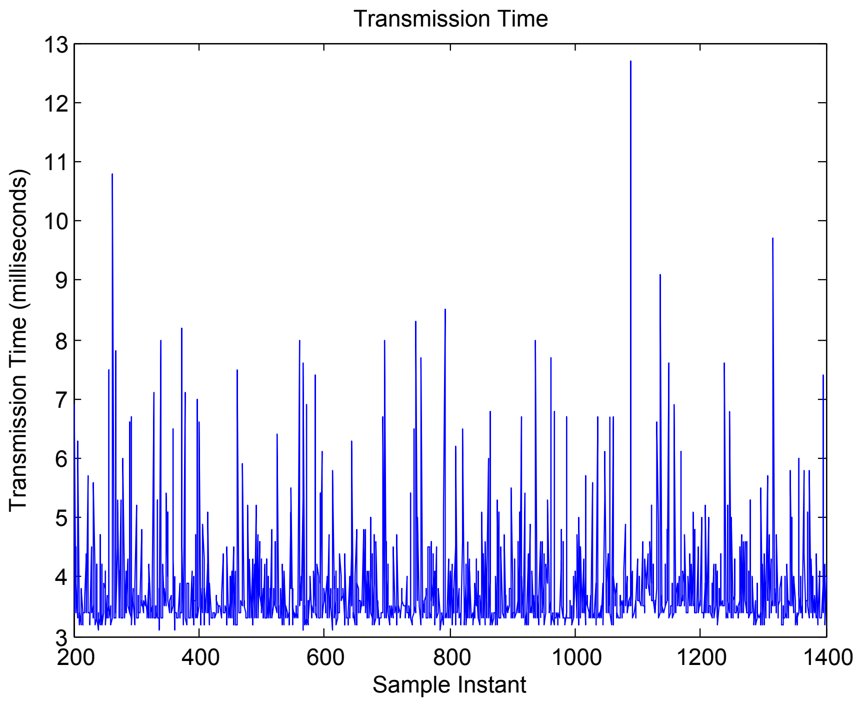


Figure 7.28: Transmission Time for Adaptive PD

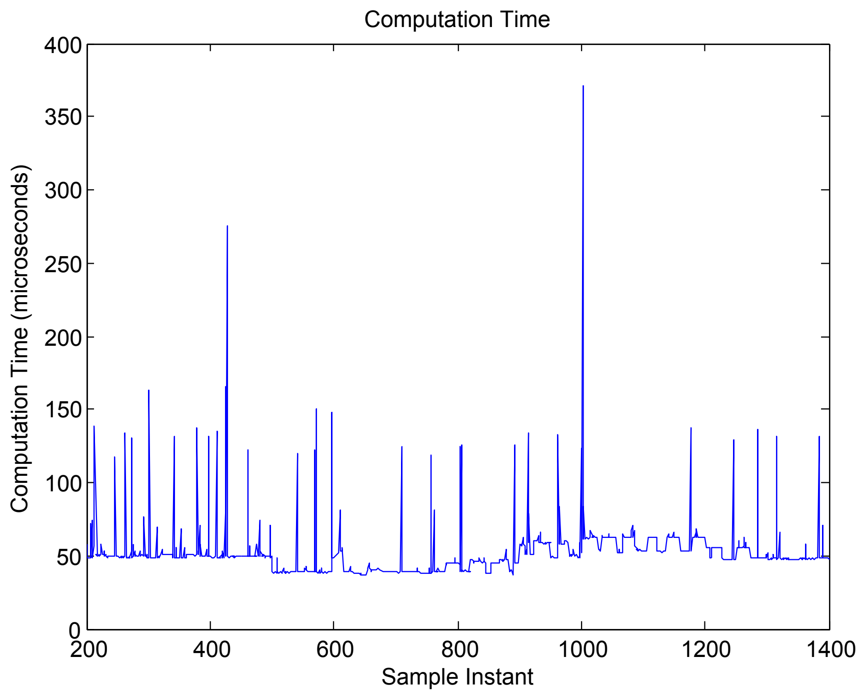


Figure 7.29: Computation Time for Non-Adaptive Backstepping

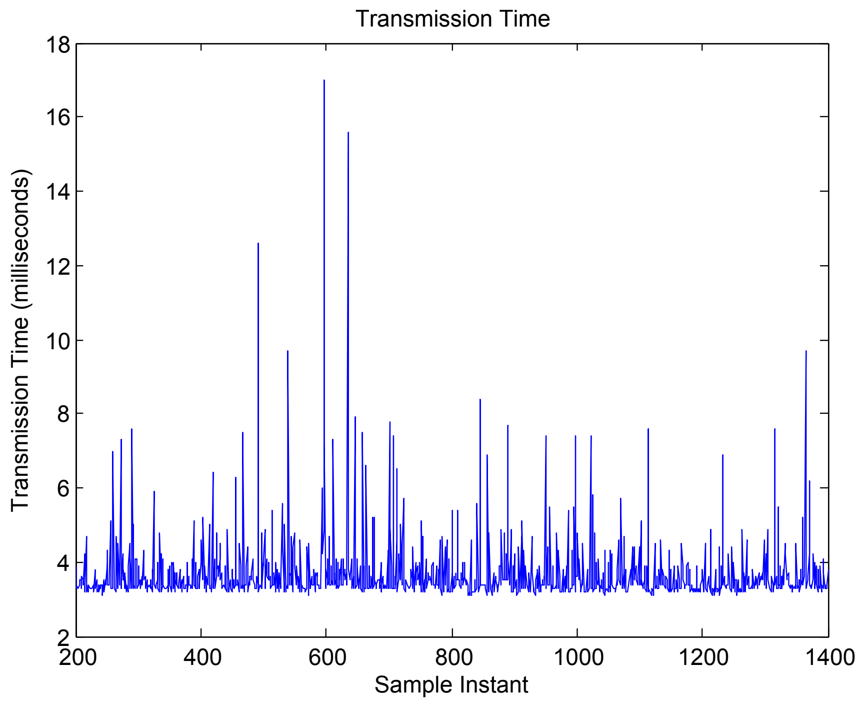


Figure 7.30: Transmission Time for Non-Adaptive Backstepping

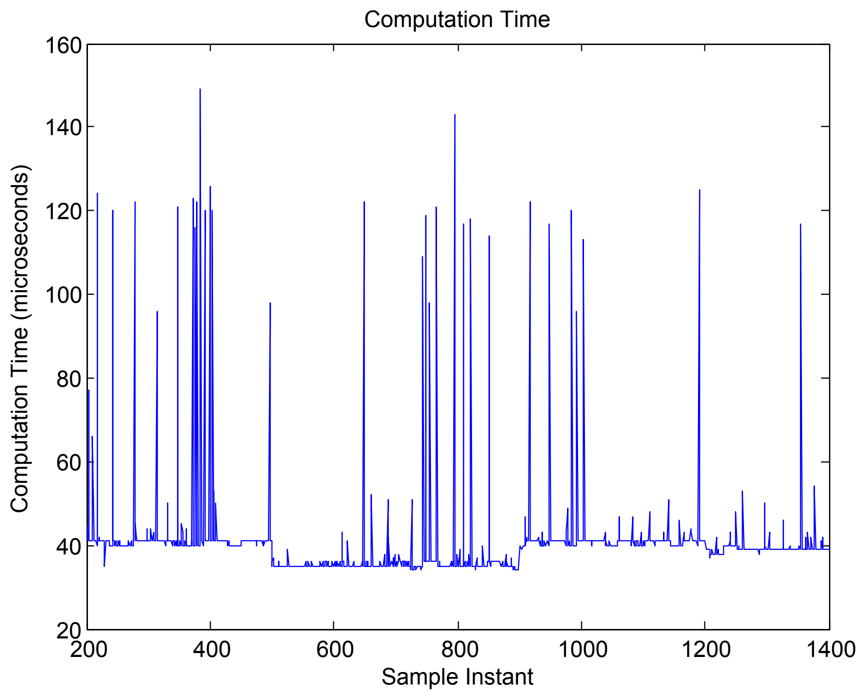


Figure 7.31: Computation Time for Adaptive Backstepping

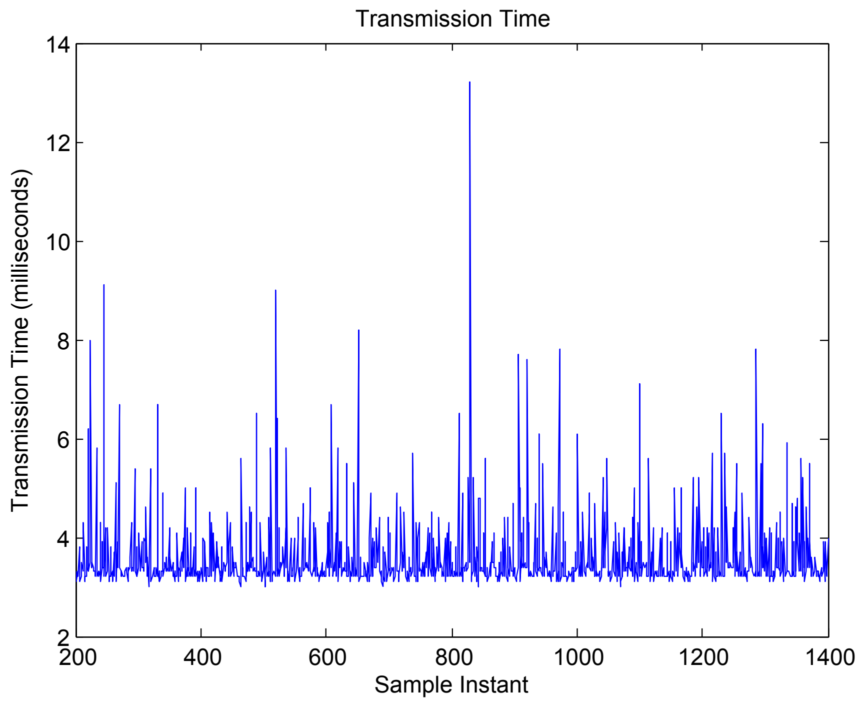


Figure 7.32: Transmission Time for Adaptive Backstepping

The preceding figures show the transmission time based on the Lakehead University network. On average, the overall transmission time for each controller was approximately 3.4 milliseconds. The main differences lay with the computation time plots. It can be seen that there is a clear decrease in computation time during the circular trajectory tracking procedure of the end effector. This is due to the fact that the desired circular trajectory is calculated offline by the server computer, so the server computer is not required to perform this operation online. The desired trajectory tracking of the progressive arc path that occurs prior and subsequent to this moment is calculated online, hence the larger computation time. Overall, the adaptive backstepping controller calculated the control signals for the end effector trajectory the quickest. The trajectory tracking of the progressive arc path needed approximately 41 microseconds to compute each sample, while the circular trajectory tracking needed approximately 35 microseconds to compute each sample. It is also important to note that the adaptive PD and backstepping controllers procured quicker computation time results than their non-adaptive counterparts. The reasoning behind this result relates to the multitude of mathematical operations present in the non-adaptive controllers due to definitive calculation of the $D(q')$, $C(q', \dot{q}')$ and $g(q')$ matrices, while the adaptive controllers estimate the value of $\Theta$ attached to these matrices.

### 7.3.4 Controller Output - PWM

The subsequent set of figures will illustrate the PWM signals generated by the DSP board to control the motors of the parallel robot system in order to achieve the desired trajectory. To calculate the PWM signals, the control signals for the non-adaptive and adaptive backstepping and PD controllers were calculated in equations (3.11), (3.21), (3.29) and (3.37), respectively.

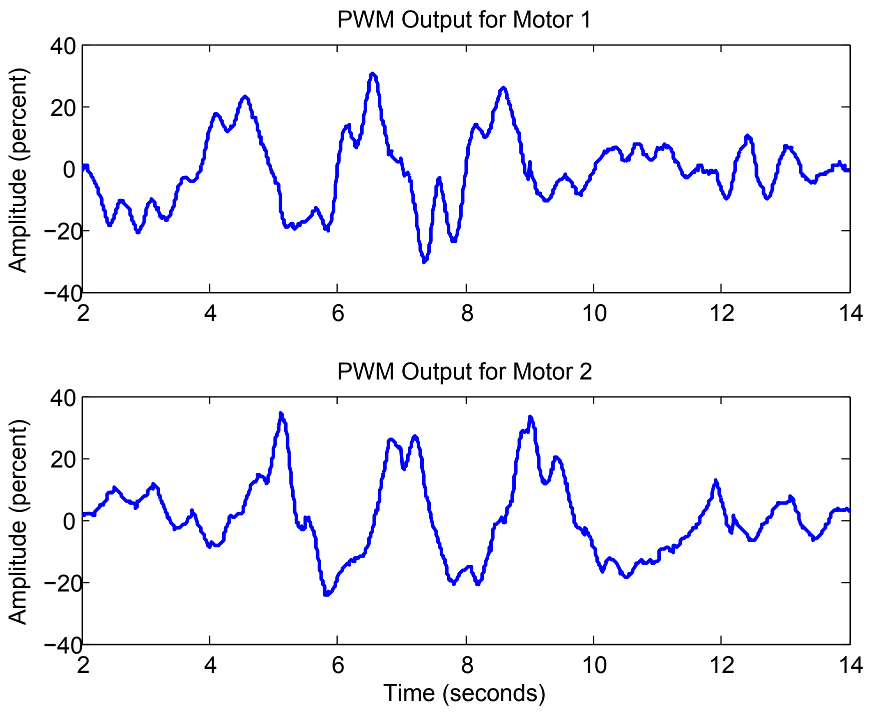


Figure 7.33: PWM Signal for Non-Adaptive PD

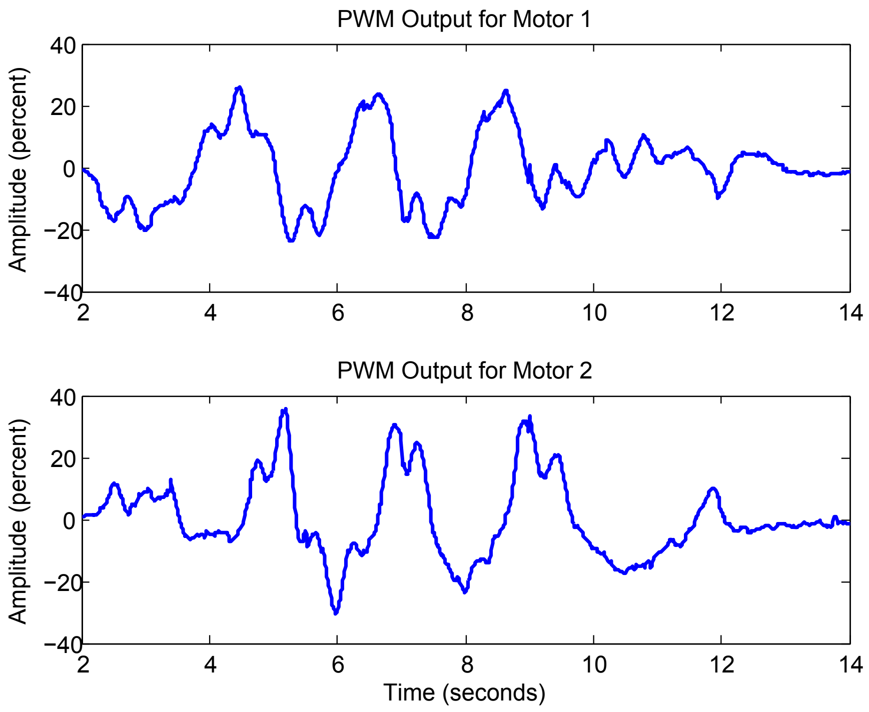


Figure 7.34: PWM Signal for Adaptive PD

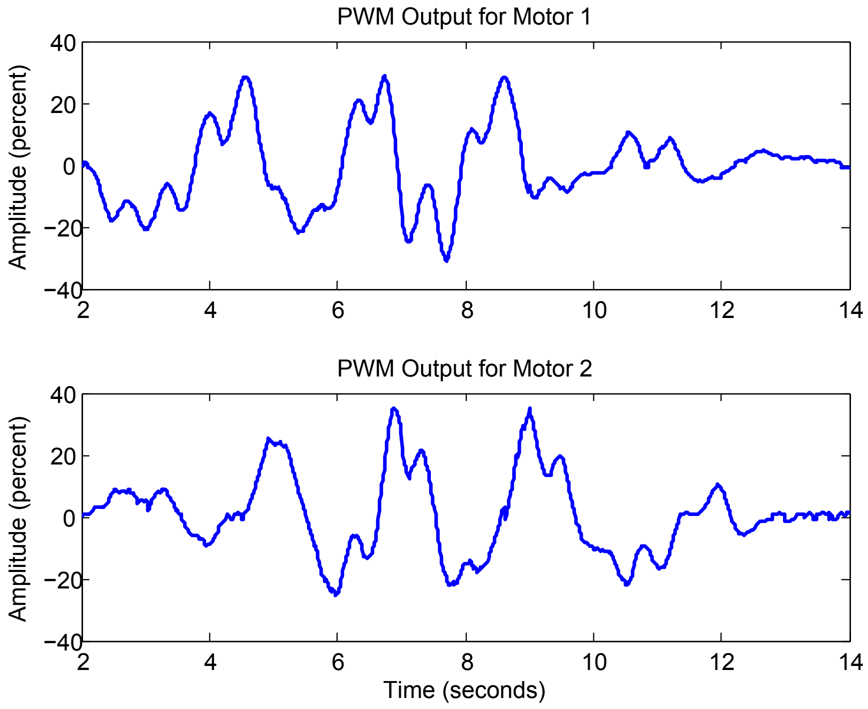


Figure 7.35: PWM Signal for Non-Adaptive Backstepping

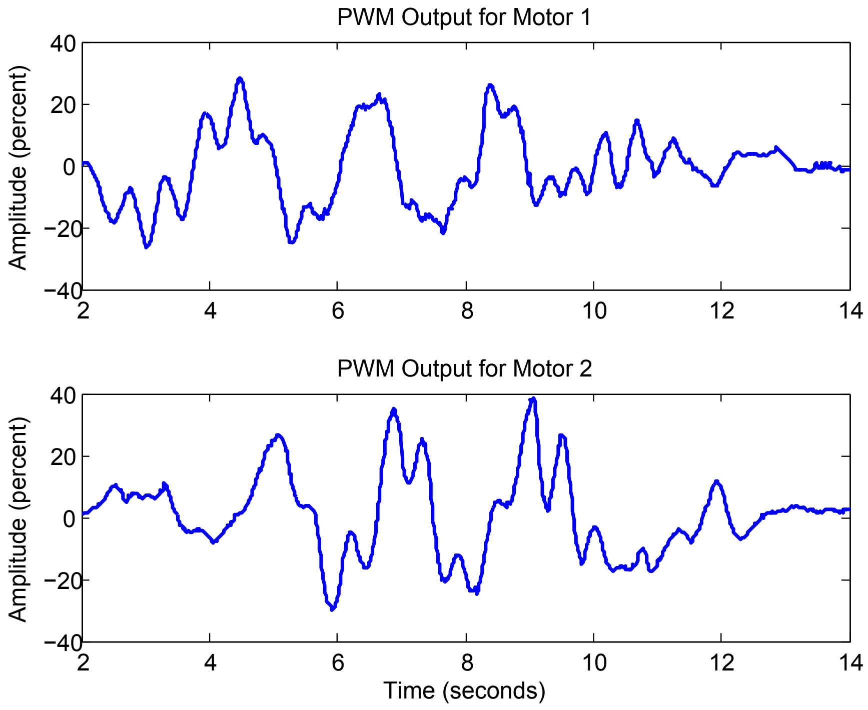


Figure 7.36: PWM Signal for Adaptive Backstepping

## 7.4 Experimentations of Fuzzy Logic Controllers

The experimentation results for all the controllers discussed in this section follow the same circular trajectory regardless of the initial position. It should be noted that the value of $\Gamma$ defined in Table 7.2 applies to all the parameters of $\Gamma$ defined in each respective controller. The indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers require an initial value for $\hat{\Theta}$. This value is defined as 0.1 for all the parameters listed in the specified controllers. Based on a 90 degree operating point, the value of $b$ in the direct adaptive fuzzy controller is 104.4766 and 99.3706 for motors one and two, respectively.

### 7.4.1 Joint Angles

The following figures depict the error between the desired and actual joint angles of $q_1$ and $q_2$, respectively. These are the only two angles directly controllable by the actuators of the parallel robot. It is crucial that the difference between the desired angles and the actual angles of $q_1$ and $q_2$ be as small as possible, in order for the end effector error to be minimized.

| | *Motor One* | | | *Motor Two* | | |
|---|---|---|---|---|---|---|
| *Fuzzy Logic Controller* | **Peak** | **Mean** | **RMS** | **Peak** | **Mean** | **RMS** |
| Fuzzy PD | 2.4519 | 0.7168 | 0.9240 | 2.1646 | 0.7268 | 0.8907 |
| Indirect Adaptive Fuzzy | 1.4248 | 0.3413 | 0.4635 | 1.5077 | 0.3730 | 0.5122 |
| Direct Adaptive Fuzzy | 1.4505 | 0.4513 | 0.6027 | 1.5287 | 0.4522 | 0.6211 |
| Fuzzy Adaptive Backstepping | 1.5920 | 0.5131 | 0.7013 | 1.5995 | 0.4764 | 0.6579 |

Table 7.4: Peak, Mean and RMS Fuzzy Logic Controller Joint Angle Errors

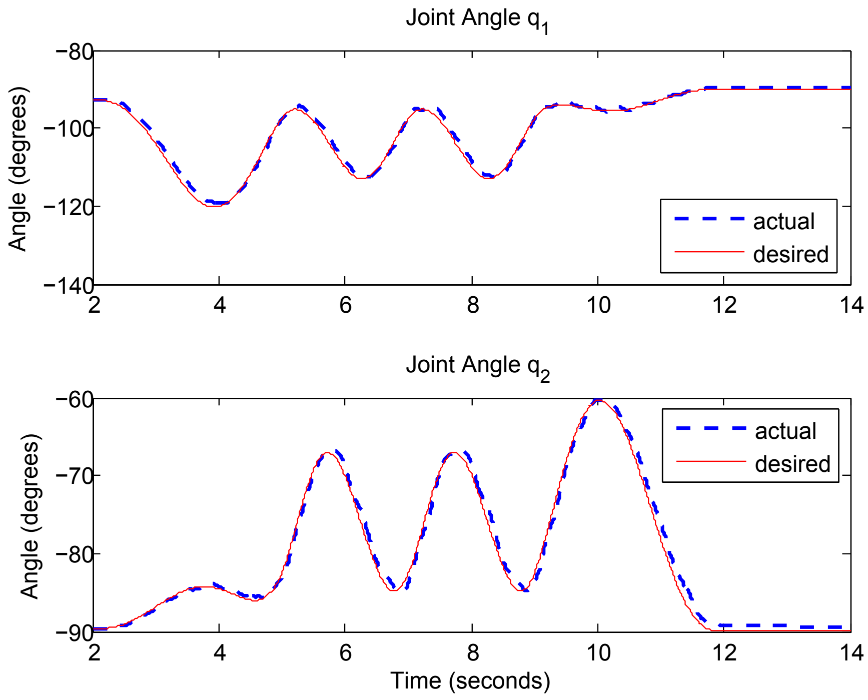


Figure 7.37: Joint Angles for Fuzzy PD

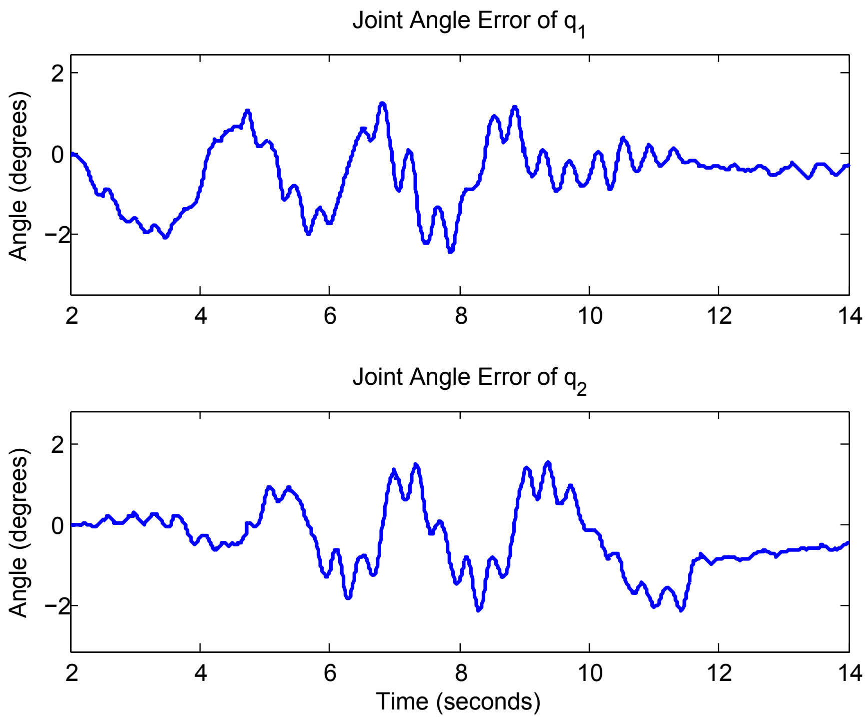


Figure 7.38: Joint Angles Error for Fuzzy PD

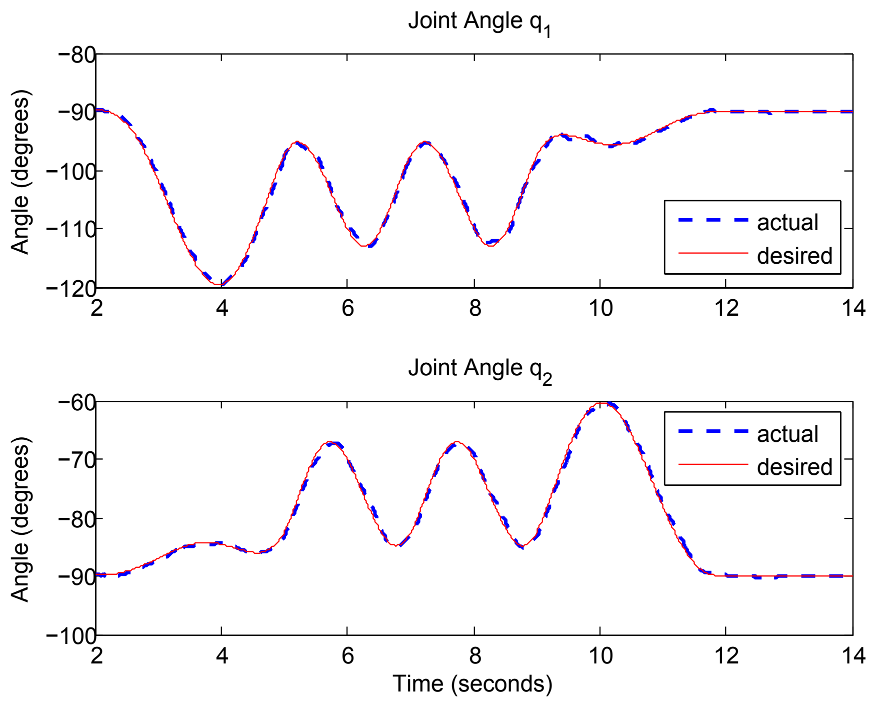


Figure 7.39: Joint Angles for Indirect Adaptive Fuzzy

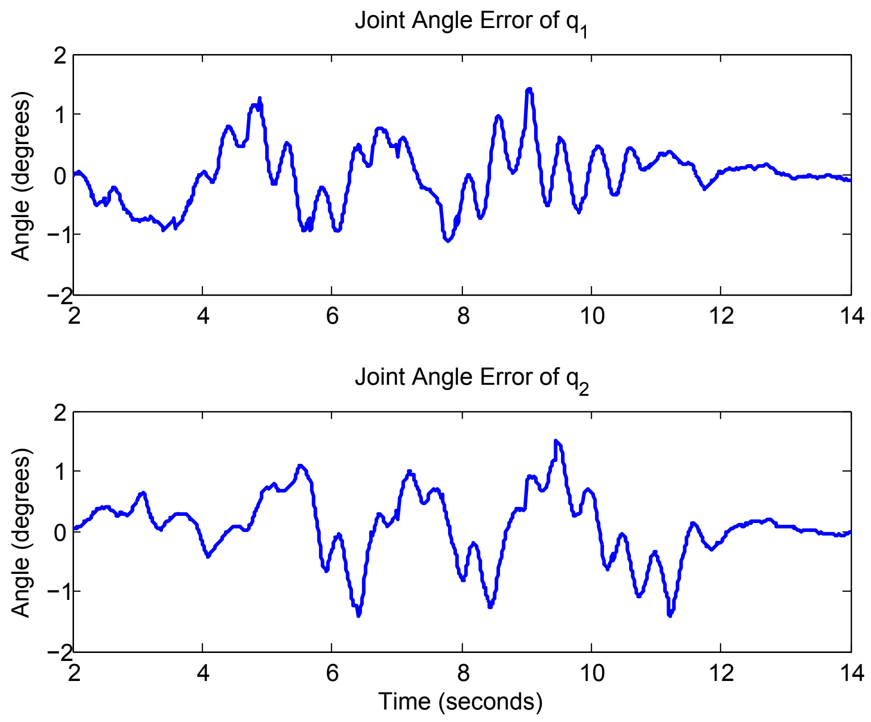


Figure 7.40: Joint Angles Error for Indirect Adaptive Fuzzy

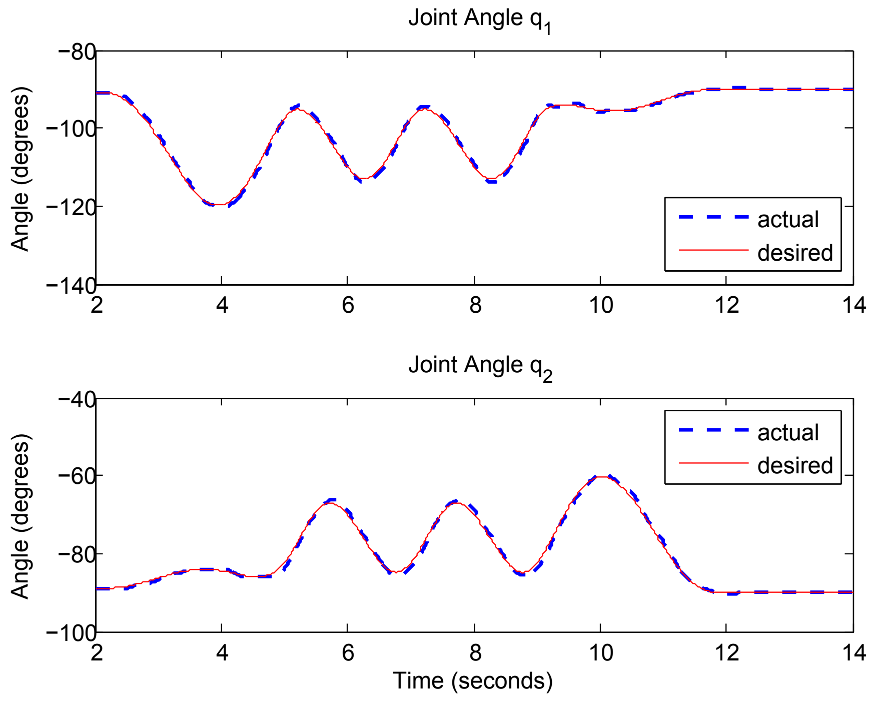


Figure 7.41: Joint Angles for Direct Adaptive Fuzzy

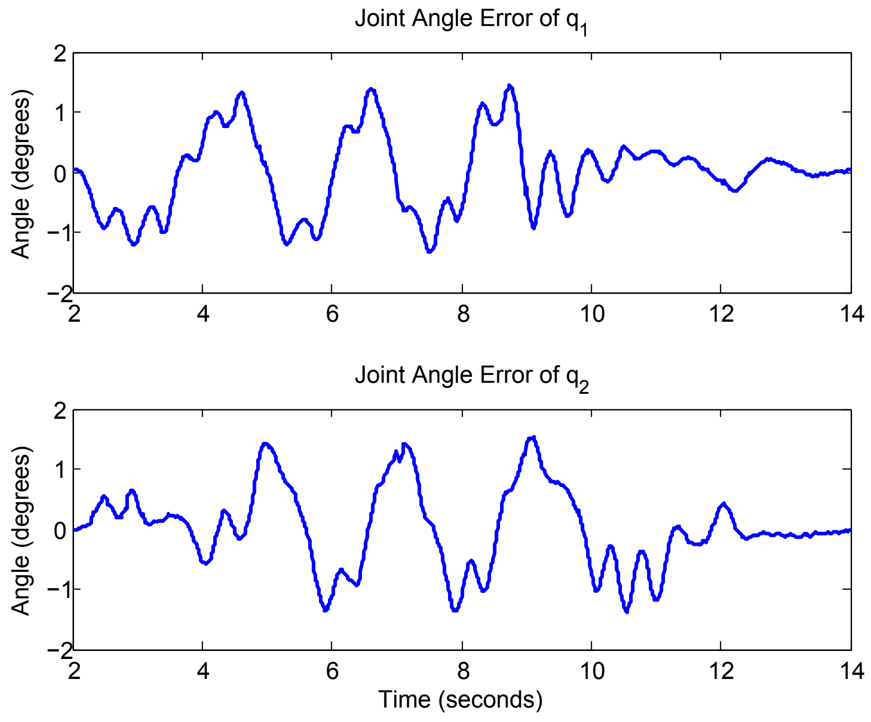


Figure 7.42: Joint Angles Error for Direct Adaptive Fuzzy

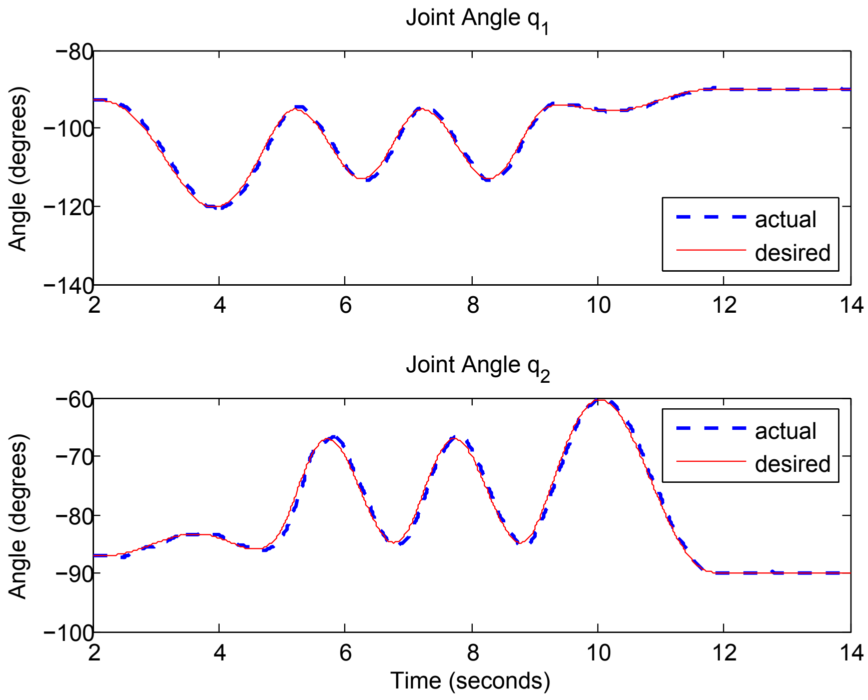


Figure 7.43: Joint Angles for Fuzzy Adaptive Backstepping

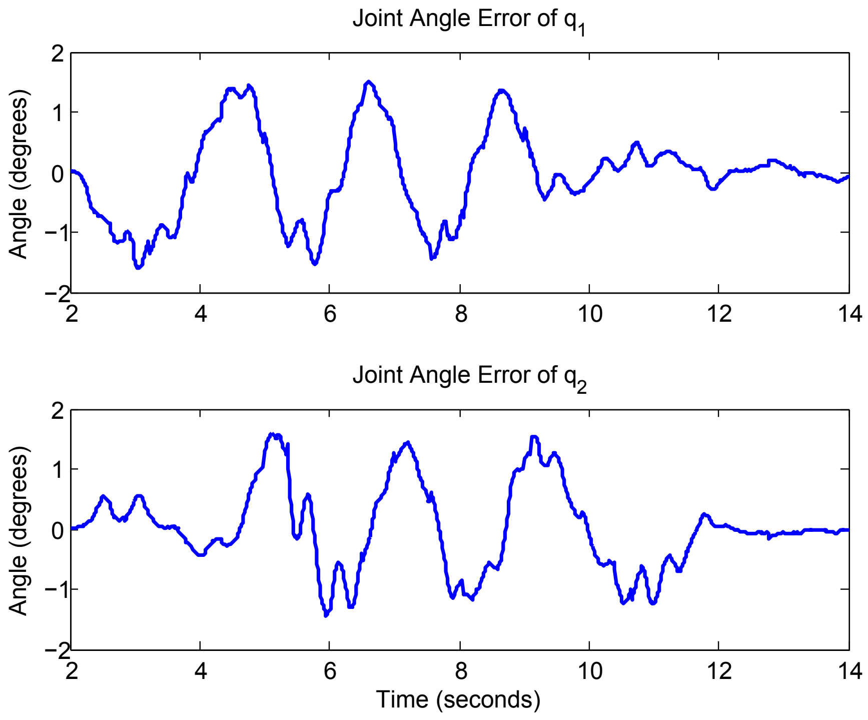


Figure 7.44: Joint Angles for Fuzzy Adaptive Backstepping

The fuzzy logic controller simulations predicted that the indirect fuzzy adaptive controller would yield the least desirable results and the fuzzy adaptive backstepping controller would produce the most desirable. Surprisingly, even though the indirect fuzzy adaptive controller was the most complex controller implemented on the parallel robot structure, it generated less error than the simpler fuzzy PD controller. The reasoning behind this is evidently due to the fact that the fuzzy PD controller could not compensate for the complexities of the parallel robot structure without introducing serious oscillations to the trajectory tracking. The remaining three controllers produced similar joint angles error plots with the results consistently less than 1.6 degrees; hence more data is required to determine the most satisfactory control technique. The peak, mean and RMS joint angle errors in degrees is shown in Table 7.4.

### 7.4.2 End Effector Trajectory

The next set of figures portray the trajectory tracking of the end effector along with its respective error between its desired and actual position. In order to determine the actual location of the end effector, equations (2.19) and (2.20) were employed to solve for $q_4$ and $q_3$, respectively. The forward kinematics equations were applied based on all the available data to definitively determine the actual location of the end effector in Cartesian coordinates. The forward kinematics equations utilized for the two degrees of freedom planar parallel robot are based upon the works defined in [68] and [46]. The results from the joint angles of $q_1$ and $q_2$ play a crucial role in the determination of the actual location of the end effector, since a small angular error would not cause a large deviation when compared to the desired trajectory. The plots of the end effector output along with their respective error in the x-axis and y-axis are also shown in order to easily identify the severity of the error.

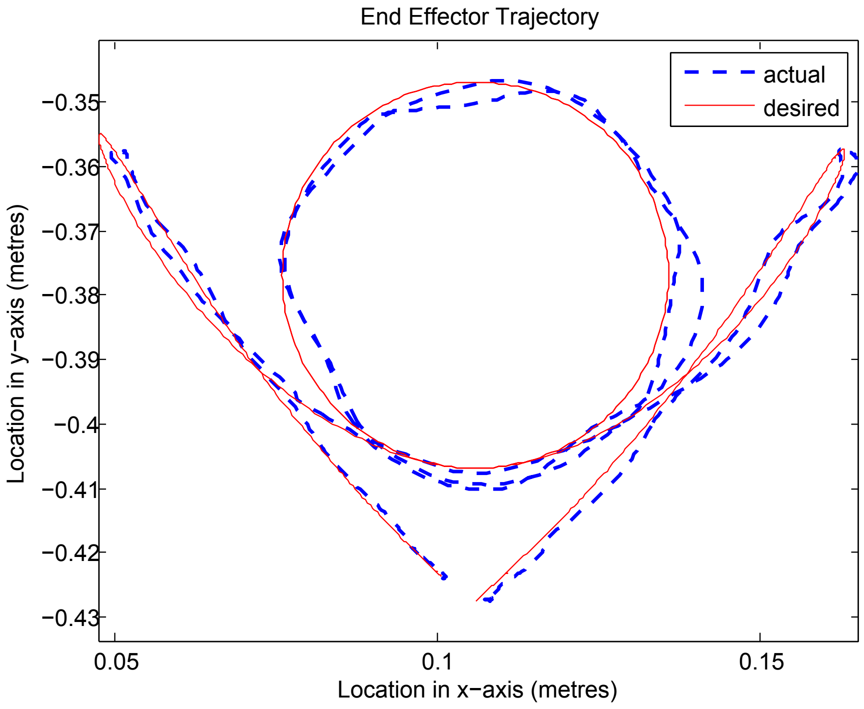


Figure 7.45: End Effector Trajectory for Fuzzy PD

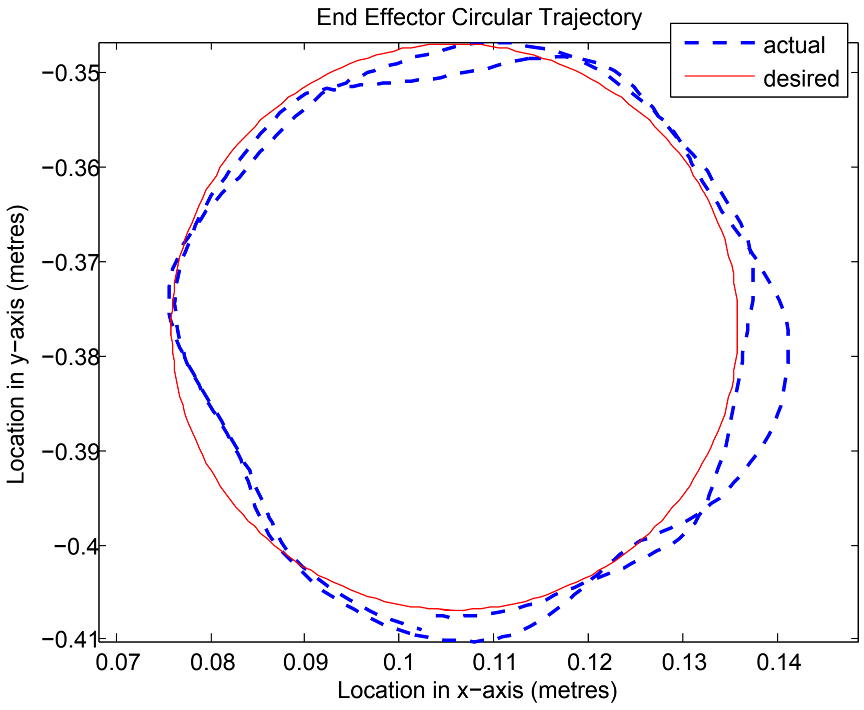


Figure 7.46: End Effector Circular Trajectory for Fuzzy PD

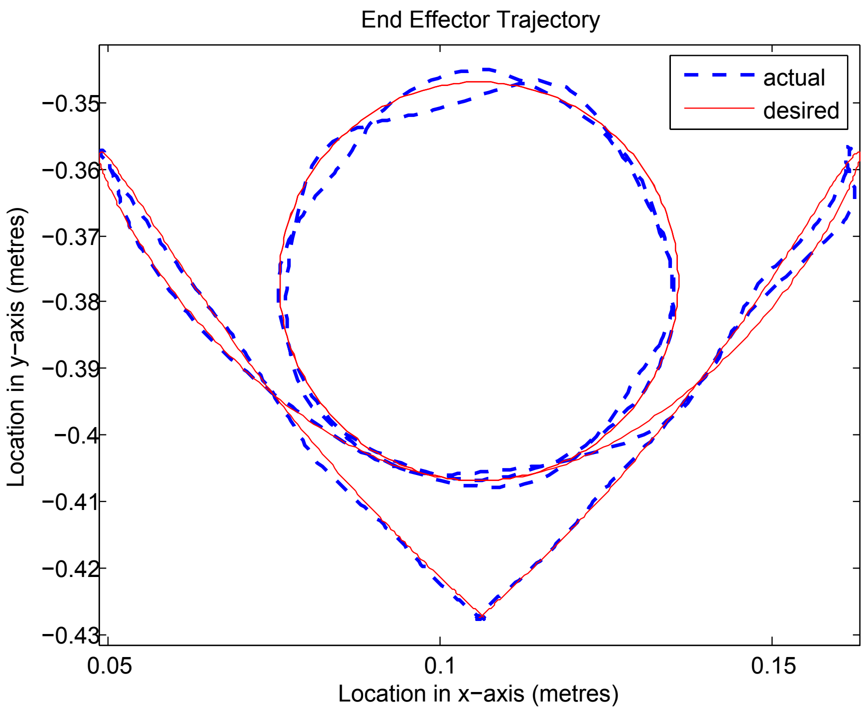


Figure 7.47: End Effector Trajectory for Indirect Adaptive Fuzzy

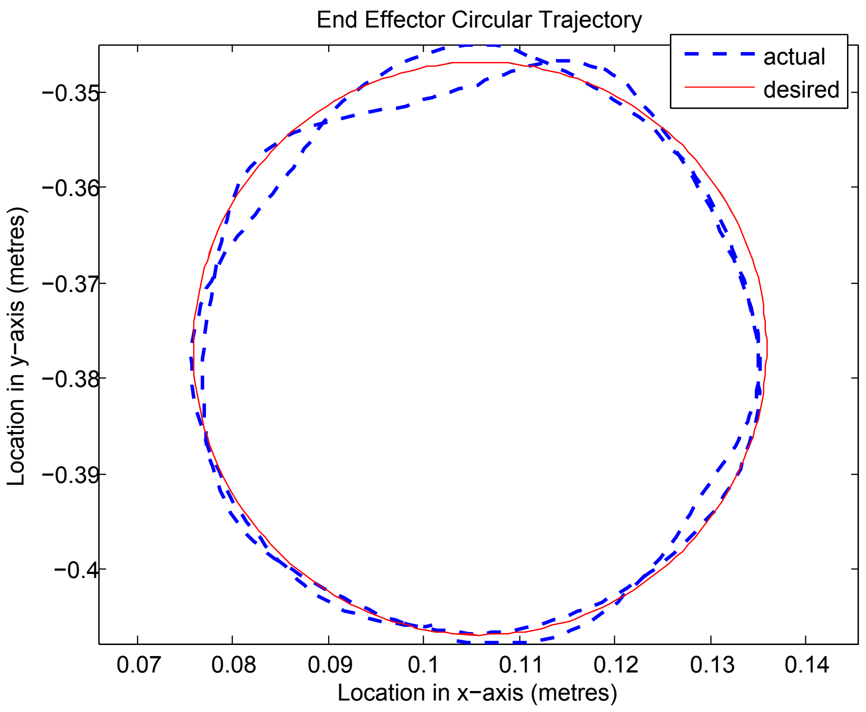


Figure 7.48: End Effector Circular Trajectory for Indirect Adaptive Fuzzy

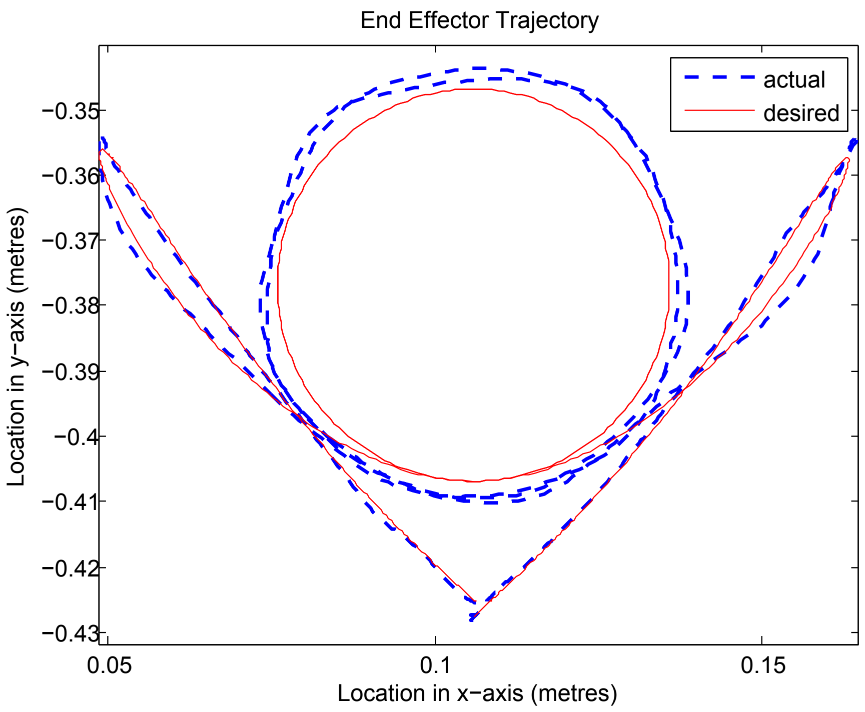


Figure 7.49: End Effector Trajectory for Direct Adaptive Fuzzy

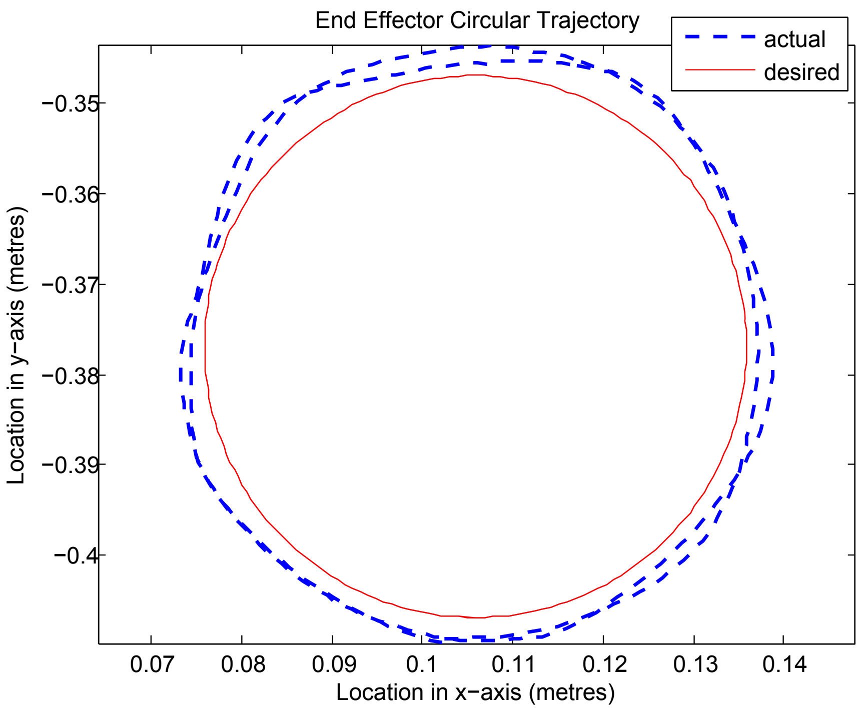


Figure 7.50: End Effector Circular Trajectory for Direct Adaptive Fuzzy

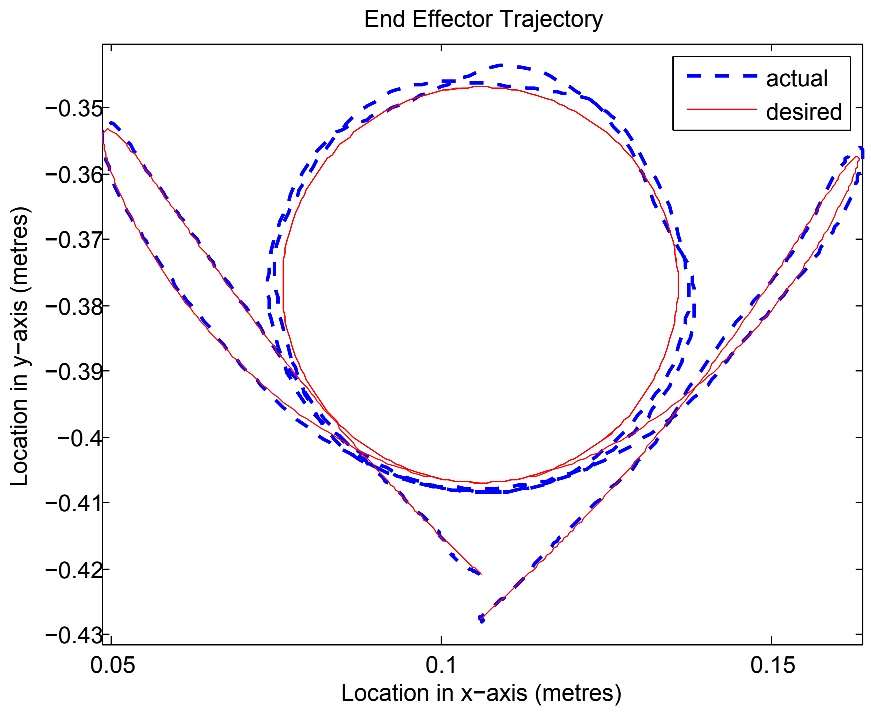


Figure 7.51: End Effector Trajectory for Fuzzy Adaptive Backstepping

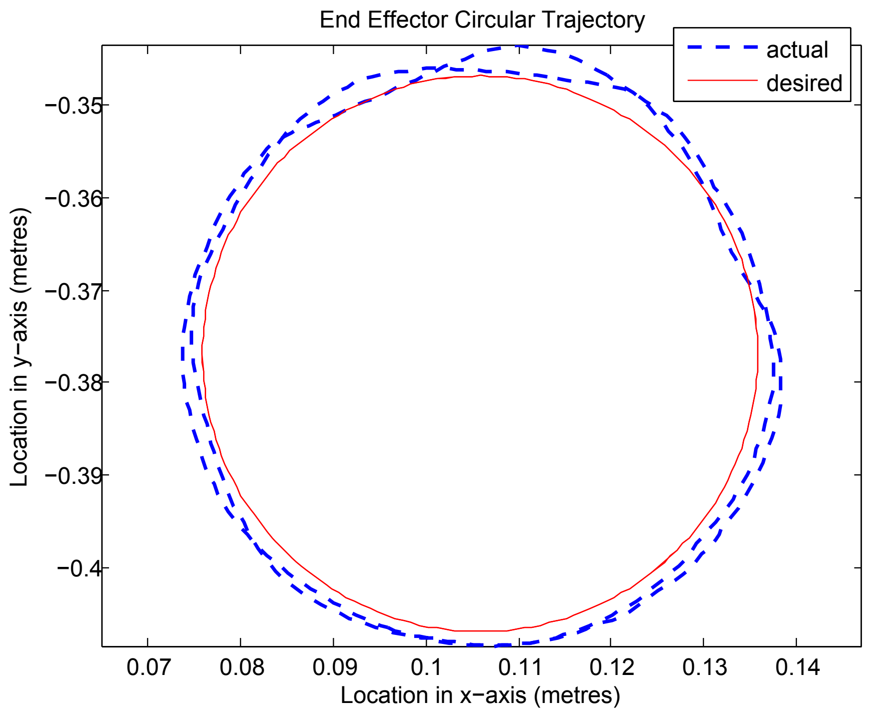


Figure 7.52: End Effector Circular Trajectory for Fuzzy Adaptive Backstepping

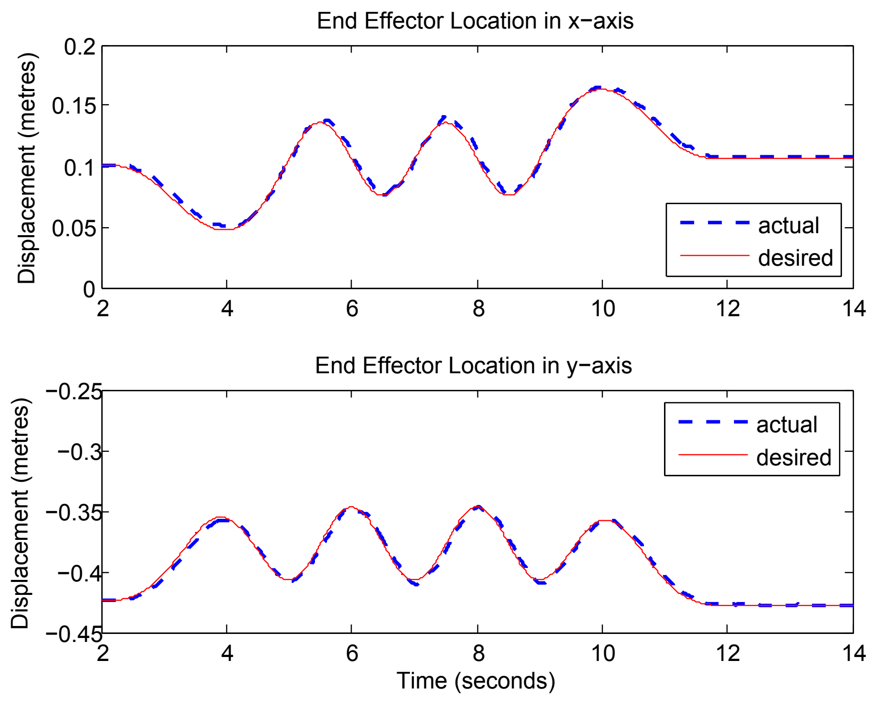


Figure 7.53: End Effector Output for Fuzzy PD

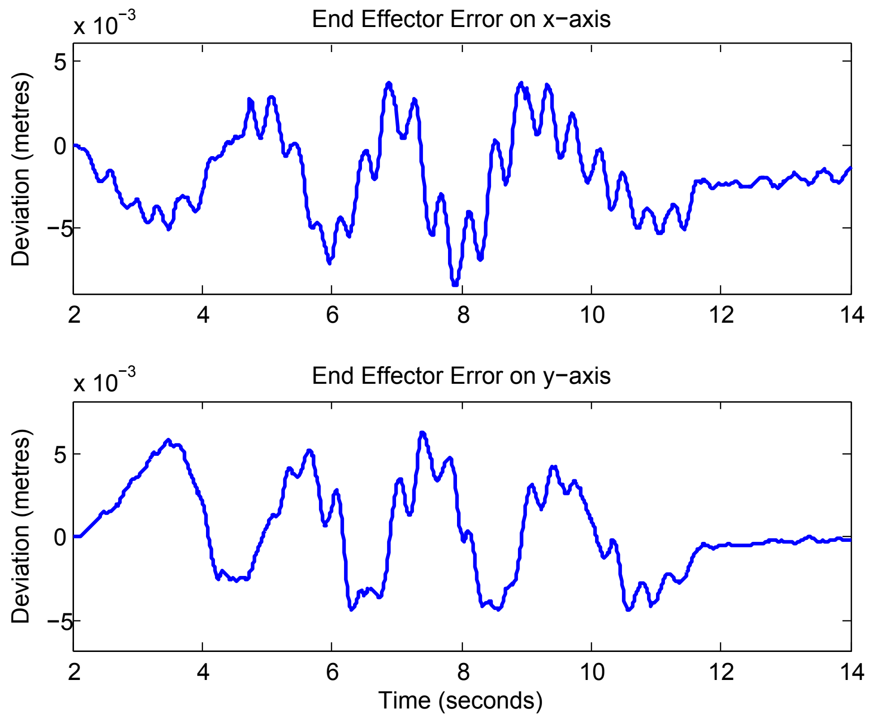


Figure 7.54: End Effector Output Error for Fuzzy PD

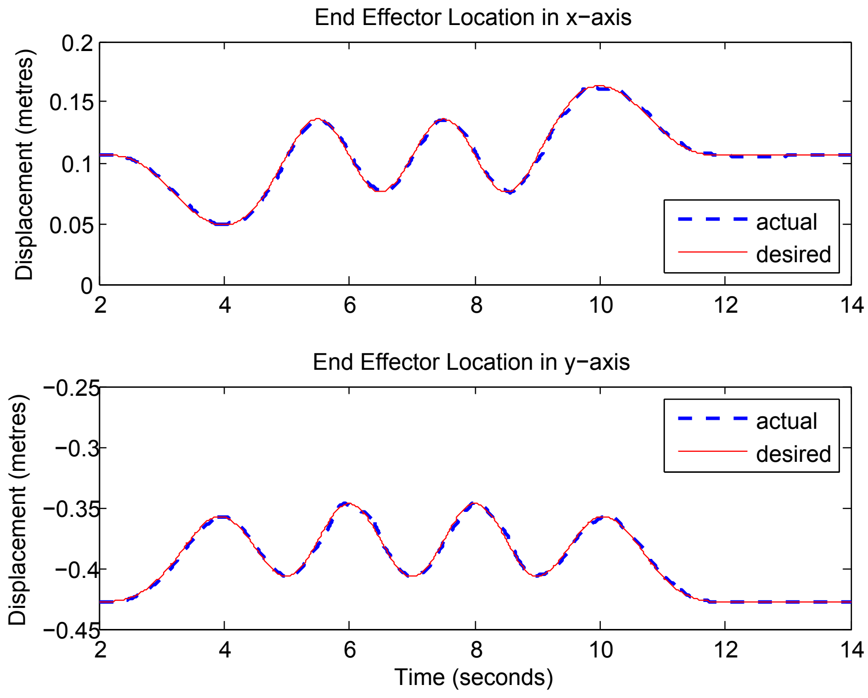


Figure 7.55: End Effector Output for Indirect Adaptive Fuzzy

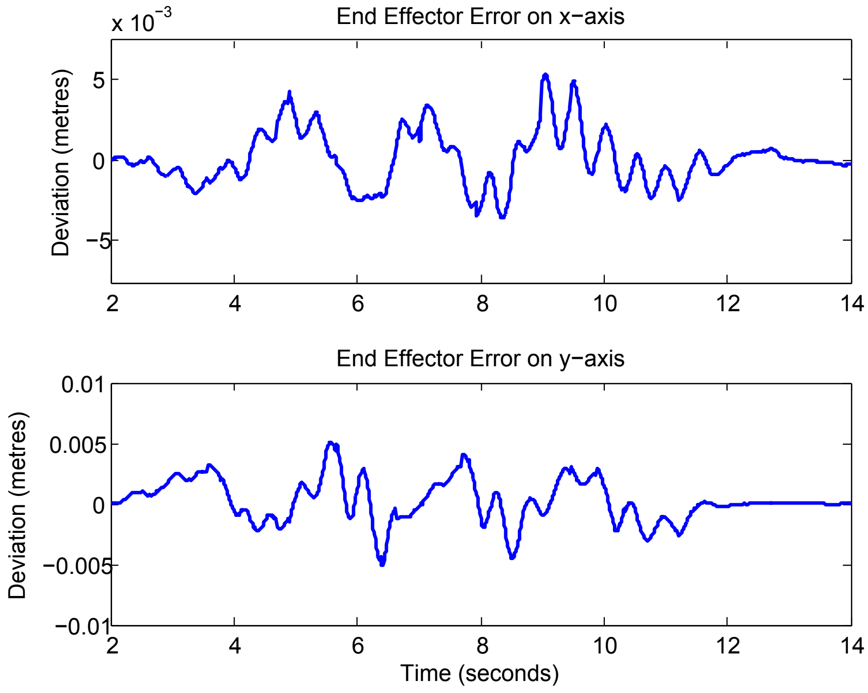


Figure 7.56: End Effector Output Error for Indirect Adaptive Fuzzy

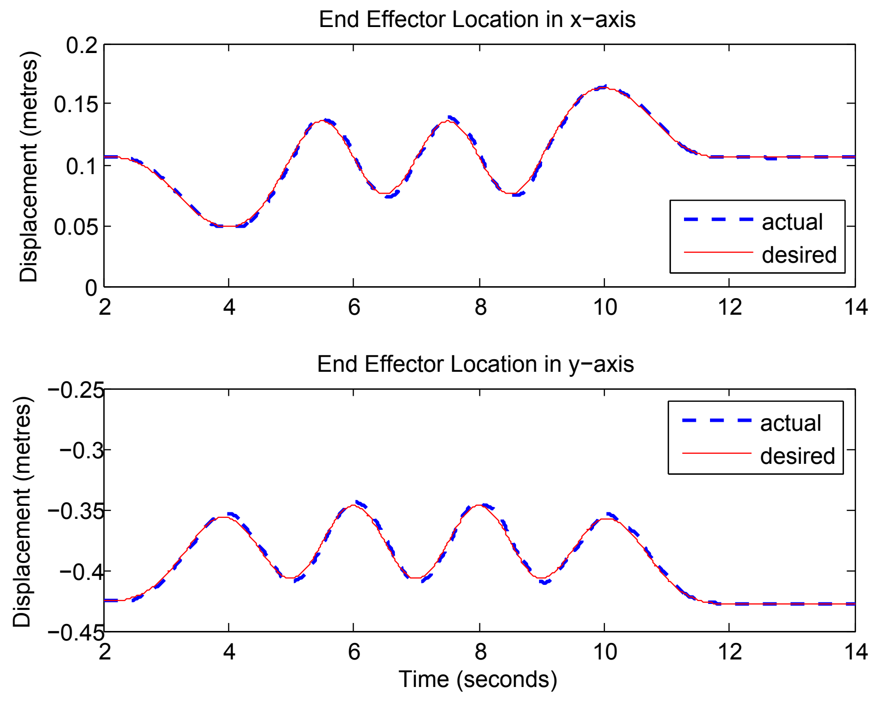


Figure 7.57: End Effector Output for Direct Adaptive Fuzzy

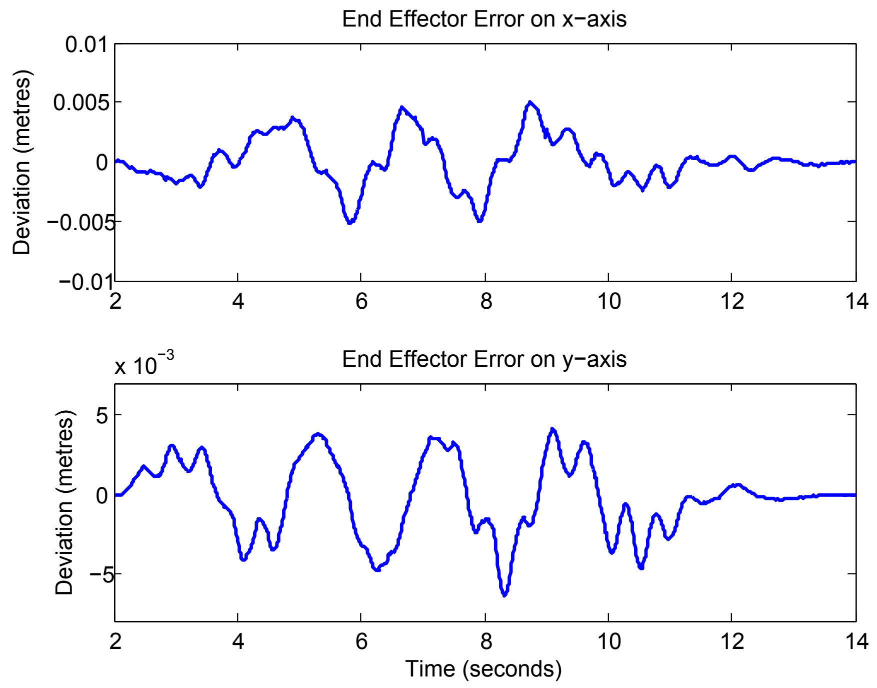


Figure 7.58: End Effector Output Error for Direct Adaptive Fuzzy

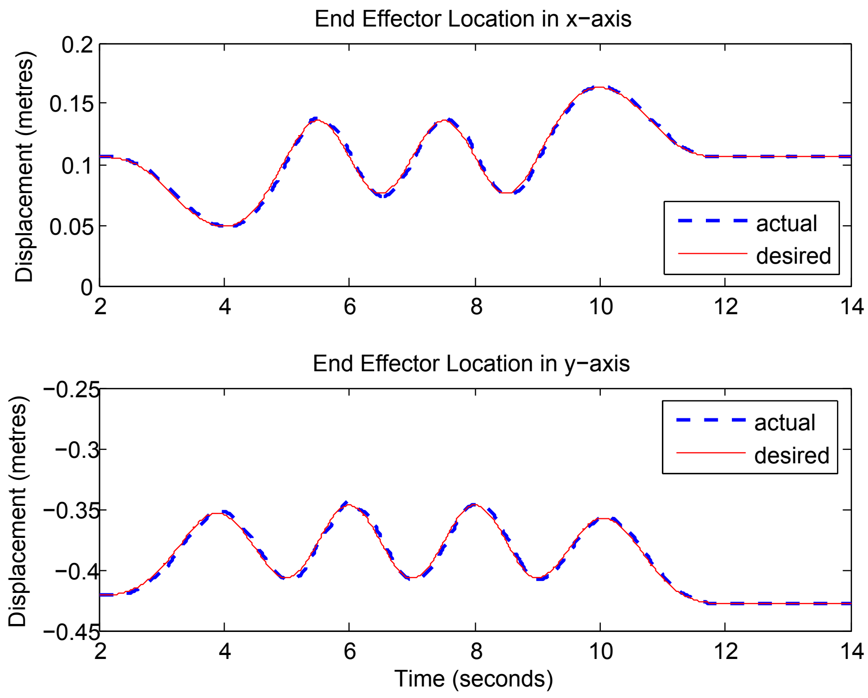


Figure 7.59: End Effector Output for Fuzzy Adaptive Backstepping

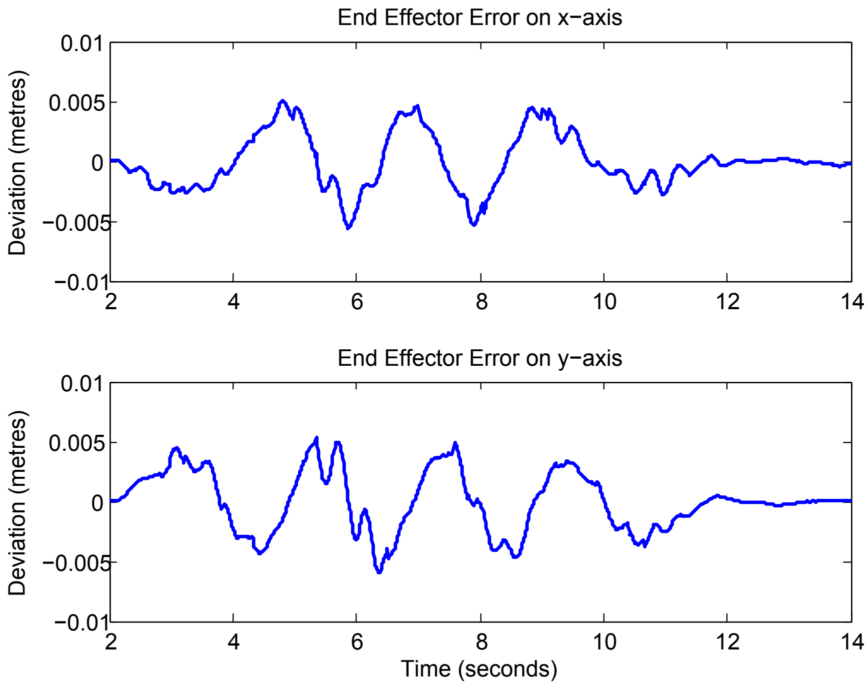


Figure 7.60: End Effector Output Error for Fuzzy Adaptive Backstepping

Similar to the previous observation, the difficulties of the fuzzy PD controller while tracking the circular trajectory is clearly visible. The end effector error in the x-axis deviates over 0.008 metres at certain instances, compared to the other three controllers which rarely achieve a deviation of 0.006 metres in either axis. The controller which achieves the least amount of end effector deviation error is the indirect adaptive fuzzy controller, but it is also more prone to oscillations when compared to the direct adaptive fuzzy controller and the fuzzy adaptive backstepping controller. The most compelling evidence concerning the most proficient circular trajectory tracking can be seen with the fuzzy adaptive backstepping controller. It has a smoother tracking performance when compared to all the other controllers discussed in this section without conceding a large displacement error and it tracks an acutely symmetrical circle. The direct adaptive fuzzy controller is also quite proficient due to the low displacement error, yet the symmetry of the circle is not as desirable as the fuzzy adaptive backstepping controller.

### 7.4.3 Computation and Transmission Time

The following plots portray the computation time, which is defined as the time for the server computer to calculate the control signals. There are also figures for the transmission time, which is defined as the time between the moment at which the DSP board sends the feedback signals over the Lakehead University network to the server and the moment at which the server receives this data plus the time between the moment at which the server sends the control signals to the DSP board over the Lakehead University network and the moment at which the DSP board receives this data.

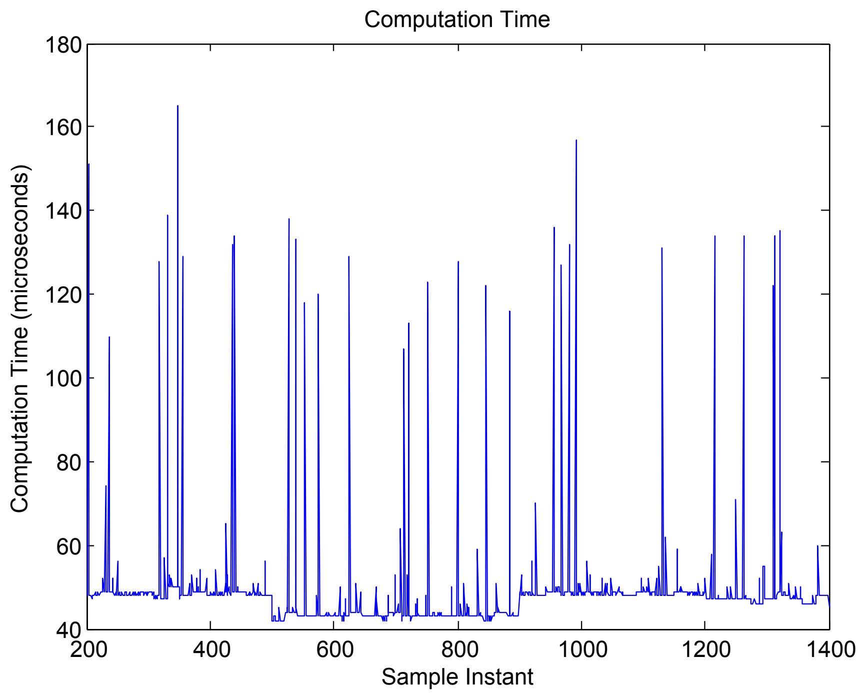


Figure 7.61: Computation Time for Fuzzy PD

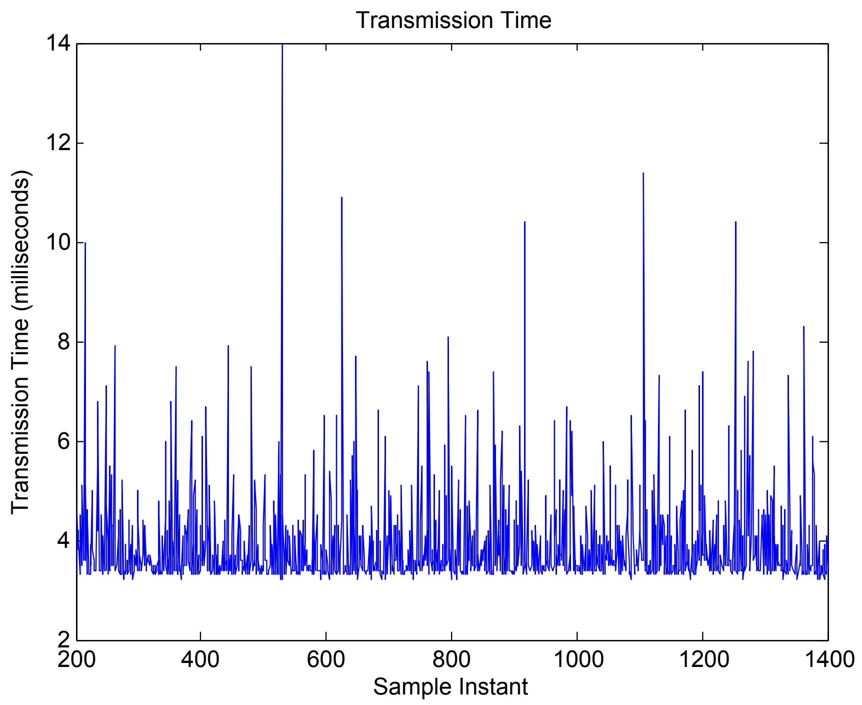


Figure 7.62: Transmission Time for Fuzzy PD

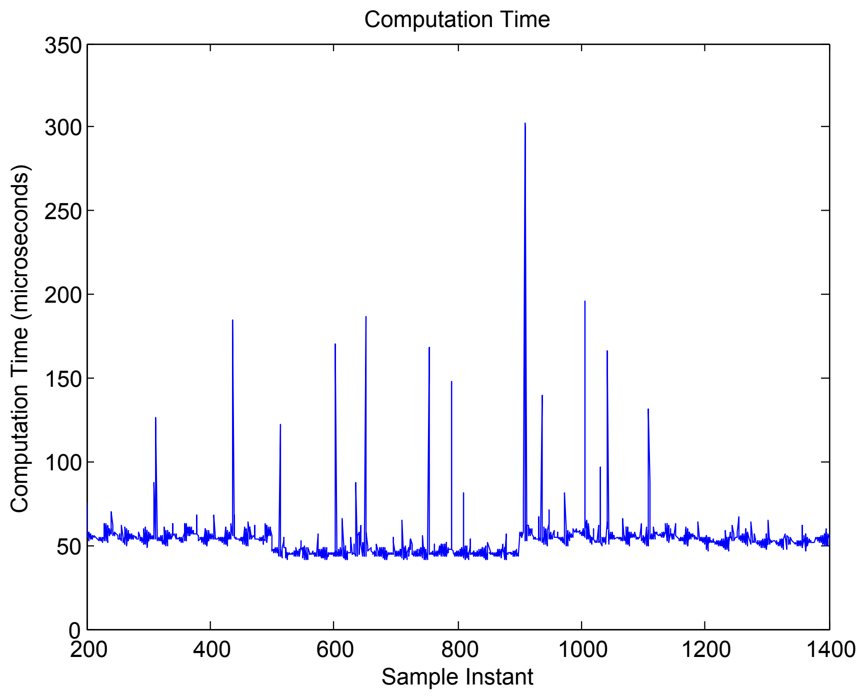


Figure 7.63: Computation Time for Indirect Adaptive Fuzzy

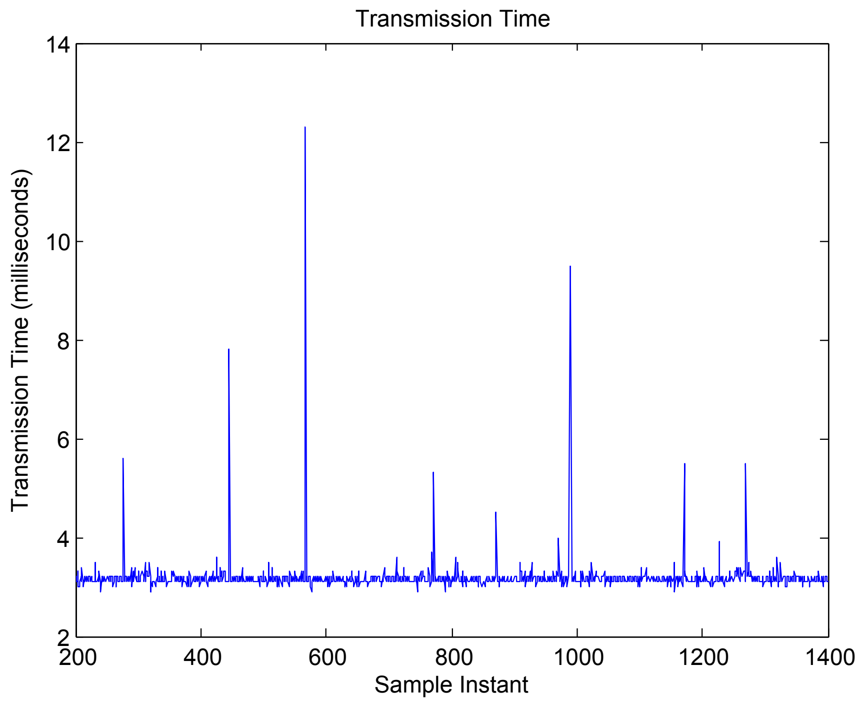


Figure 7.64: Transmission Time for Indirect Adaptive Fuzzy

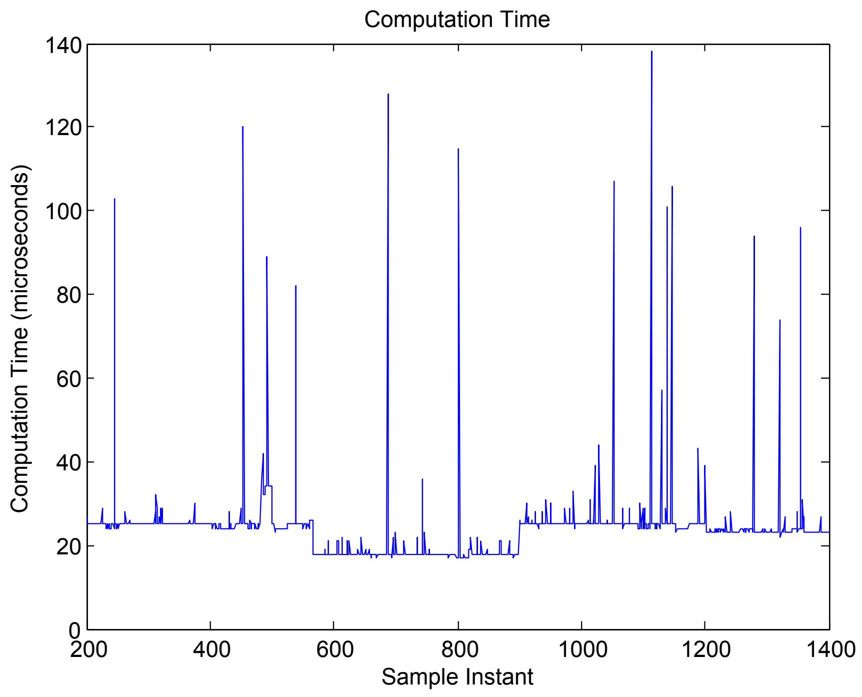


Figure 7.65: Computation Time for Direct Adaptive Fuzzy

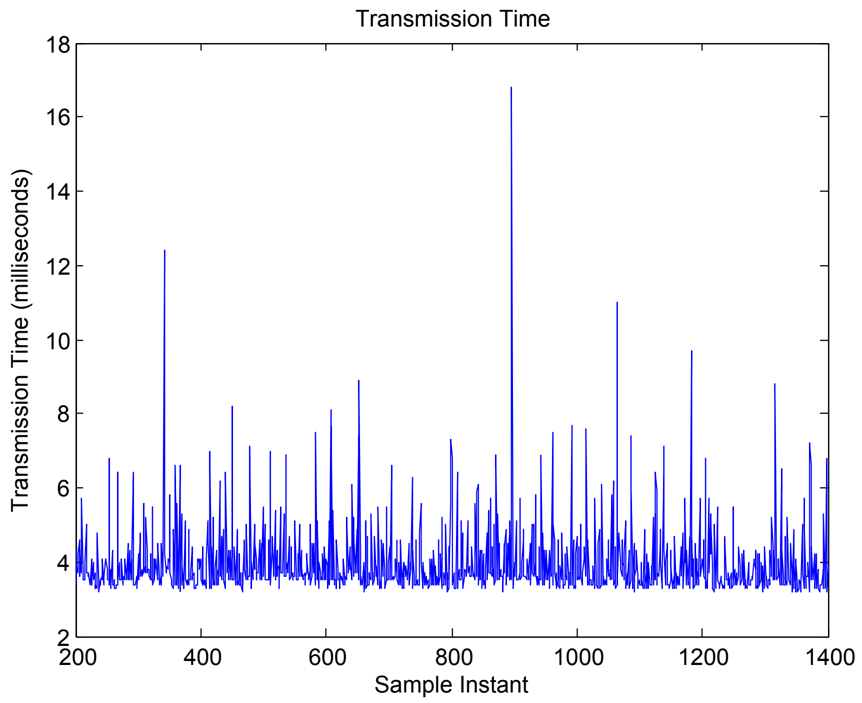


Figure 7.66: Transmission Time for Direct Adaptive Fuzzy

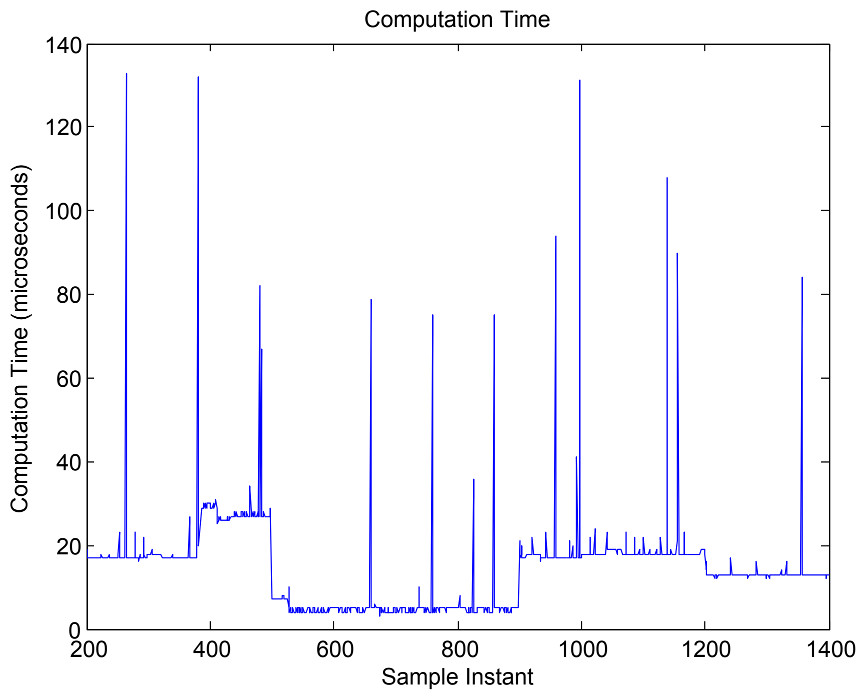


Figure 7.67: Computation Time for Fuzzy Adaptive Backstepping

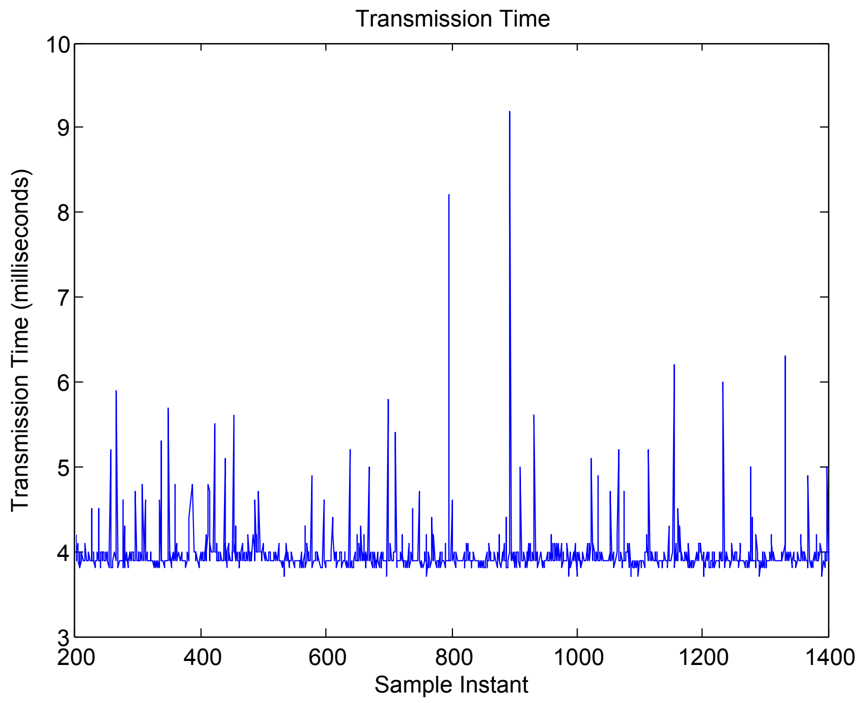


Figure 7.68: Transmission Time for Fuzzy Adaptive Backstepping

The preceding figures show the transmission time based on the Lakehead University network. On average, the overall transmission time for each controller was approximately 3.8 milliseconds, except for the indirect adaptive fuzzy controller, which achieved a result of approximately 3.15 milliseconds. The main differences lay with the computation time plots. It can be seen that there is a clear decrease in computation time during the circular trajectory tracking procedure of the end effector. This is due to the fact that the desired circular trajectory is calculated offline by the server computer, so the server computer is not required to perform this operation online. The desired trajectory tracking of the progressive arc path that occurs prior and subsequent to this moment is calculated online, hence the larger computation time. Overall, the fuzzy adaptive backstepping controller calculated the control signals for the end effector trajectory the quickest. The trajectory tracking of the progressive arc path needed approximately 17 microseconds to compute each sample, while the circular trajectory tracking needed approximately 5 microseconds to compute each sample. It is also important to note that the indirect adaptive fuzzy controller took the longest to calculate the control signals, which makes sense since it is the most computationally intensive control technique. The direct adaptive fuzzy controller attained respectable results; needing approximately 25 microseconds to compute each sample the trajectory tracking of the progressive arc path and approximately 18 microseconds for the circular trajectory tracking.

### 7.4.4 Controller Output - PWM

The subsequent set of figures will illustrate the PWM signals generated by the DSP board to control the motors of the parallel robot system in order to achieve the desired trajectory. To calculate the PWM signals, the control signals for the fuzzy PD, indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers were calculated in equations (4.7), (4.12), (4.43) and (4.89), respectively.

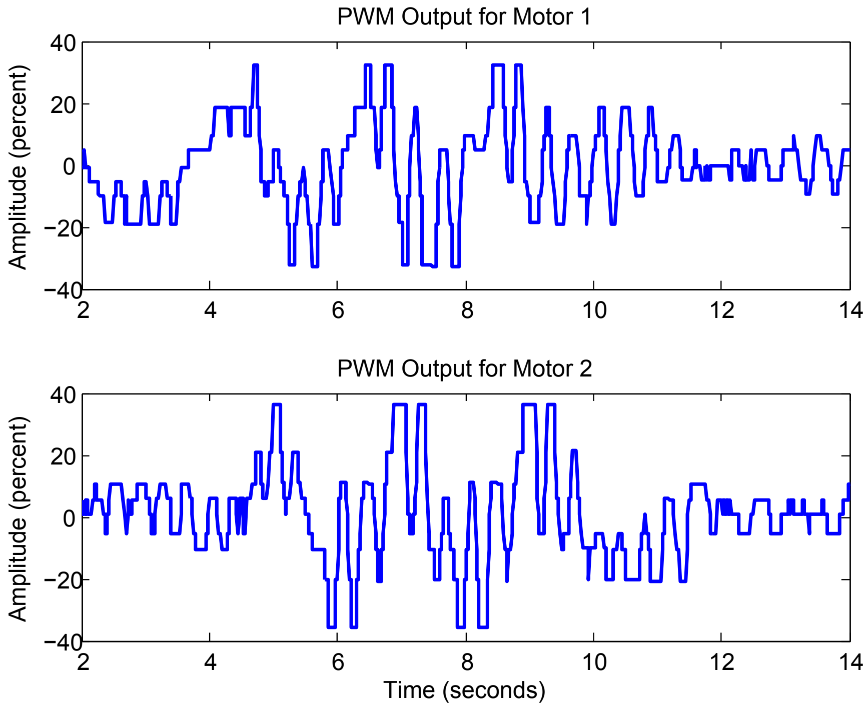


Figure 7.69: PWM Signal for Fuzzy PD

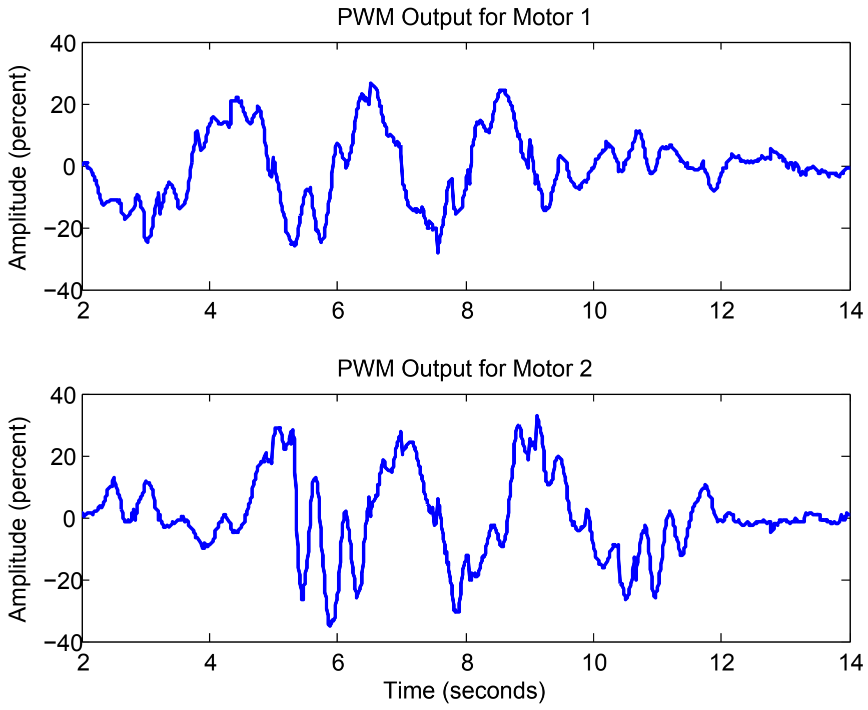


Figure 7.70: PWM Signal for Fuzzy Adaptive Backstepping

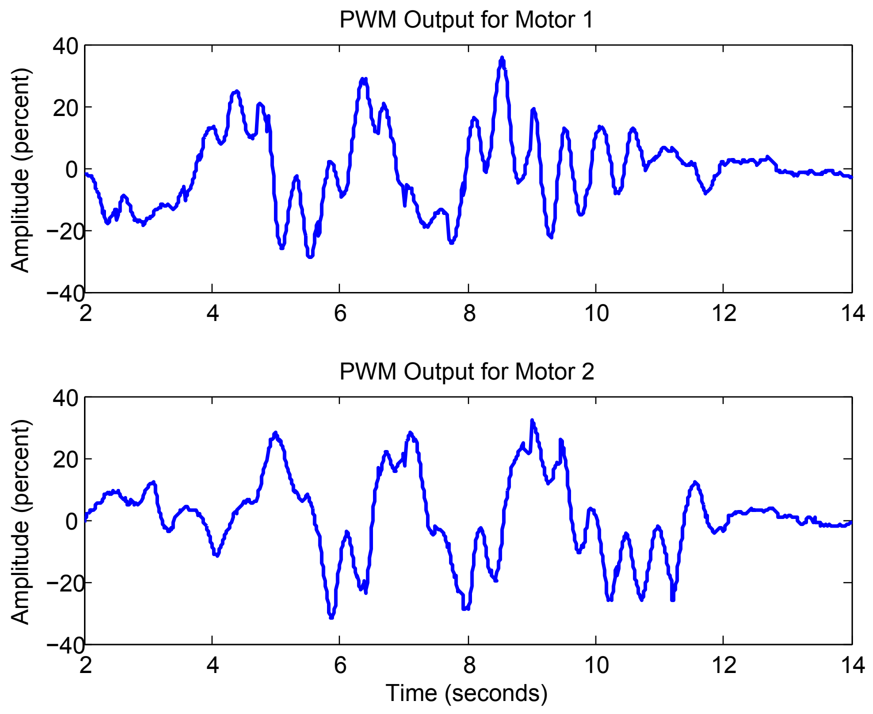


Figure 7.71: PWM Signal for Indirect Adaptive Fuzzy

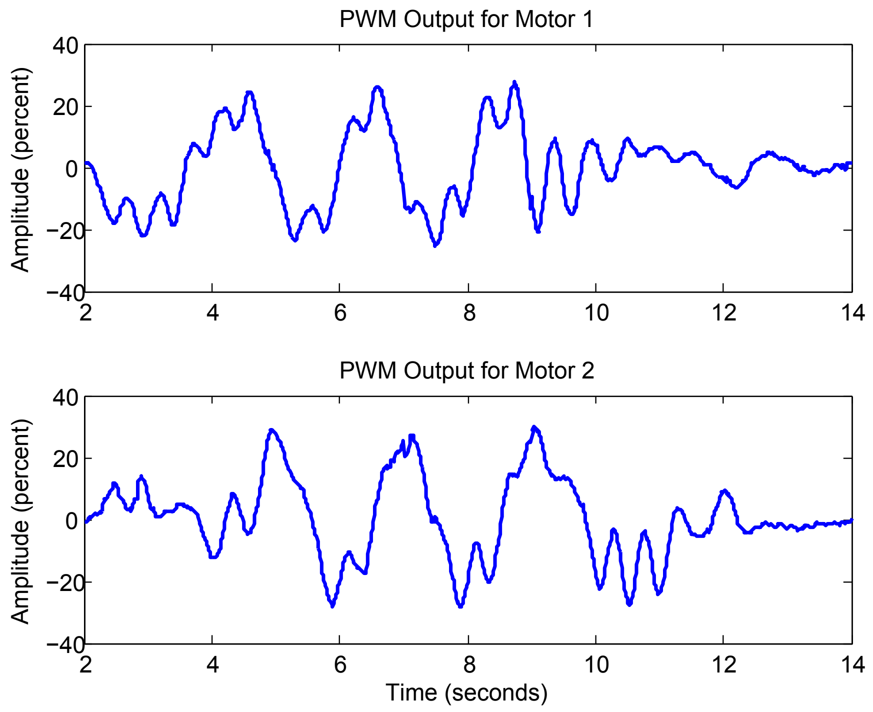


Figure 7.72: PWM Signal for Direct Adaptive Fuzzy

## 7.5 Controller Recommendation

Eight controllers have been simulated on MATLAB and experimented on the planar two degrees of freedom parallel robot structure. The joint angles, trajectory tracking and controller output figures were presented and analyzed between one another for both the simulation and experimental results; hence by using all the available data, a final conclusion will affirm the most appropriate controller to implement on the parallel robot structure.

The simulation results of the non-fuzzy controllers portrayed the non-adaptive and adaptive backstepping controllers as the most likely candidate controllers to achieve the most adequate tracking performance due to their quick convergence. The simulation results of the fuzzy controllers portrayed the fuzzy adaptive backstepping controller as the most proficient control technique due to the minimal circular trajectory errors, while the indirect adaptive fuzzy controller portrayed the most undesirable controller results due to the system estimation error prevalent in its controller output plot. Therefore, the next logical step consisted of the implementation of all the simulated controllers on the experimental apparatus to examine whether the results would concur with one another.

The experimentation results of the non-fuzzy controllers yielded certain dissimilar outcomes when compared to their simulation results. The adaptive PD and backstepping controllers had less profound oscillations while performing the trajectory tracking and the computation time was significantly less than their non-adaptive counterparts. Yet, it was the adaptive PD controller that held a slight edge against the adaptive backstepping controller due to the repeatability of tracking an acutely symmetrical circle while performing the circular end effector trajectory. The most exciting results lay in the experimentation of the fuzzy logic controllers. Contrary to the simulation results, the indirect adaptive fuzzy controller performed much better in terms of the PWM signal generated for the motors, while the fuzzy PD controller yielded slightly worse trajectory tracking plots. It is more than likely that the reason the indirect adaptive fuzzy controller achieved better findings during the experimental trials than its simulation results is due to the multitude of calculations involved while approximating the unknown parameters in MATLAB. The ordinary differential equation solver is accredited with significantly increasing the overall simulation time due the computation time required to estimate the large number of unknown parameters that must be solved in this control technique. This observation is prevalent in the experimental results since the indirect adaptive controller suffers from the longest computation time out of all the control techniques. The indirect adaptive fuzzy controller realized the lowest end effector deviation when compared to all eight control techniques, but it was also more prone to slight oscillations while performing the desired trajectory, hence causing the circular trajectory tracking to yield unsatisfactory results. As for the fuzzy PD controller, it could not compensate for the complexities of the parallel robot structure without introducing serious oscillations to the trajectory tracking. The direct adaptive fuzzy controller performed nearly exactly as the simulation results predicted. It achieved very low displacement error, yet the symmetry of the circle was not as desirable as the fuzzy adaptive backstepping controller.

Overall, the controller that outputted the most impressive experimental results were found in the fuzzy adaptive backstepping controller. The joint angles yielded a maximum angular error of 1.5 degrees, which is approximately 0.5 degrees smaller than any of the non-fuzzy logic controllers and on par with the indirect and direct adaptive fuzzy controllers. The end effector trajectory figures portrayed a maximum of 0.006 metres of deviation in the x-axis and y-axis, but it is the tracking performance which sets this controller apart from the rest. It has the smoothest tracking performance when compared to all the other controllers discussed in this thesis without conceding a large displacement error and it tracks an acutely symmetrical circle. Another significant advantage of the fuzzy adaptive backstepping controller when compared to all the other control techniques is the computation time required to generate the control signals. The trajectory tracking of the progressive arc path needed approximately 17 microseconds to compute each sample, while the circular trajectory tracking needed approximately 5 microseconds to compute each sample. The trajectory tracking of the progressive arc path is 8 microseconds quicker and the circular trajectory tracking is 13 microseconds faster than the direct adaptive fuzzy controller.

Therefore, based on the generated simulation and experimental results along with the comparisons between all eight controllers, the candidate controller which best suits the needs of the planar two degrees of freedom parallel robot is the fuzzy adaptive backstepping controller. The simulation and experimental results both show how accurate this control technique is, which is exemplified by the fact that it contains the greatest number of tuneable parameters of any of the controllers discussed in this thesis without being computationally intensive.

# Chapter 8

# Conclusion

## 8.1 Conclusion

The purpose of this thesis was to compare the simulation and experimental results of the non-adaptive and adaptive PD and backstepping controllers along with the fuzzy PD, indirect adaptive fuzzy, direct adaptive fuzzy and fuzzy adaptive backstepping controllers and garner a recommendation for the most suitable control technique to employ on the planar two degrees of freedom parallel robot structure. A summary of the differences between serial and parallel robot structures introduced the background of robotics, while the literature review provided a detailed account of the beginning of robotics. It focused on the main contributors in the field of parallel robots, whom began an unstoppable force of ingenuity that produced the vast varieties of parallel robot structures that play key roles in our society today. The planar two degrees of freedom parallel robot was introduced and modelled using the dynamic equations, inverse kinematics and non-singular region in order to adequately define the parameters of the non-linear system. The derivations of eight controllers were solved to ensure stability of the closed loop system, which all eight controllers achieved a negative semi-definite solution for $\dot{V}_2$ or $\dot{V}$. This led to the simulation of the non-fuzzy and fuzzy controllers in MATLAB to analyze whether the actual circular trajectory could satisfactorily track the desired circular trajectory. Once this task was completed, the electrical and mechanical design of the physical parallel robot structure was discussed in detail. The final portion consisted of the experimentation of the eight controllers on the planar two degrees of freedom parallel robot in order to conclusively determine the most appropriate controller to employ on the physical structure.

In conclusion, the fuzzy adaptive backstepping controller yielded the most remarkable results. The controller attained accurate end effector tracking results without compromising the amount of computation time and control effort usually found in more complex control techniques. It is highly recommended that the fuzzy adaptive backstepping control technique be utilized in various parallel robot structures to determine if similar results can be achieved.

## 8.2 Future Work

Due to the success of the fuzzy adaptive backstepping controller, the next step would comprise of programming the common point to perform more complex manoeuvres. This would lead to the implementation of a proper end effector that could manipulate an object by the pick and place procedure.

Another issue that must be addressed concerns the proof for the stability of the indirect and direct adaptive fuzzy controllers using the interconnection terms. The performance of the adaptive controllers can also be improved by implementing the term $-\sigma\hat{\Theta}$ to prevent the parameters from going to infinity, which would destabilize the system over a long duration of operation.

# References

[1] Ahmadi, M., Dehghani, M., Eghtesad, M., and Khayatian, A., *Inverse Dynamics of Hexa Parallel Robot Using Lagrangian Dynamics Formulation.*

[2] Astrom, K. J., and Wittenmark, B., *Adaptive Control - Second Edition.* Addison-Wesley, New York, 1995.

[3] Begon, P., Pierrot, F., and Dauchez, P., *Fuzzy Sliding Mode Control of a Fast Parallel Robot.* Proceedings of the 1995 IEEE International Conference on Robotics and Automation, pp. 1178-1183, 1995.

[4] Bennettis, S., *Nicolas Minorsky and the Automatic Steering of Ships.* Control Systems Magazine, pp. 10-15, November 1984.

[5] Bonev, I., *The True Origins of Parallel Robots.* ParalleMIC. January 23, 2003. May 23, 2010. http://www.parallemic.org/Reviews/Review007.html.

[6] Caccavale, F., Siciliano, B., and Villani, L., *The Tricept Robot : Dynamics and Impedance Control.* IEEE/ASME Transactions on Mechatronics, Volume 8, No. 2, pp. 263-268, June 2003.

[7] Chen, B., Liu, X., Liu, K., and Lin, C., *Direct Adaptive Fuzzy Control of Nonlinear Strict – Feedback Systems.* Automatica, Volume 45, pp. 1530-1535, 2009.

[8] Chukkala, V., De Leon, P., Horan, S., and Velusamy, V., *Modeling the Radio Frequency Environment of Mars for Future Wireless, Networked Rovers and Sensor Webs*. Proceedings of the 2004 IEEE Aerospace Conference, pp. 1329-1336, November 2003.

[9] Clavel, R., *Delta, a Fast Robot with Parallel Geometry*. 18th International Symposium on Industrial Robot, pp. 91-100, April 1988.

[10] Deblaise, D., and Maurine, P., *Effective Geometrical Calibration of a Delta Parallel Robot Used in Neurosurgery*.

[11] Di Gregorio, R., *Forward Position Analysis of the $SP-PS-RS$ Architectures*. International Journal of Robotics and Automation, Volume 21, No. 4, pp. 295-301, 2006.

[12] Fu, K., and Mills, J., K., *Robust Control Design for a Planar Parallel Robot*. International Journal of Robotics and Automation, Volume 22, No. 2, pp. 139-147, 2007.

[13] *Fuzzy Logic Toolbox User's Guide*. The MathWorks Inc., Version 2.1.1, June 2001.

[14] Ghorbel, F., *Modeling and PD Control of Closed − Chain Mechanical Systems*. Proceedings of the 34th Conference on Decision and Control, pp. 540-542, December 1995.

[15] Ghorbel, F., and Gunawardana, R., *A Validation Study of PD Control of a Closed − Chain Mechanical System*. Proceedings of the 36th IEEE Conference on Decision and Control, pp. 1998-2004, 1997.

[16] Ghorbel, F., Chetelat, O., and Longchamp, R., *A Reduced Model for Constrained Rigid Bodies with Application to Parallel Robots*. Proceedings of the IFAC Symposium of Robot Control, pp. 57-62, September 1994.

[17] Ghorbel, F., Chetelat, O., Gunawardana, R., and Longchamp, R., *Modeling and Set Point Control of Closed – Chain Mechanisms : Theory and Experiment*. IEEE Transactions on Control Systems Technology, pp. 801-815, 2000.

[18] Gough, V. E. and Whitehall, S.G., *Universal Tyre Test Machine*. Proceedings of the FISITA Ninth International Technical Congress, pp. 117-137, May 1962.

[19] Hahn, B., and Valentine, D.T., *Essential MATLAB for Engineers and Scientists*. Butterworth-Heinemann, Italy, 2007.

[20] Hsu, F., Y., and Fu, L., C., *A Novel Adaptive Fuzzy Variable Structure Control for a Class of Nonlinear Uncertain Systems via Backstepping*. Proceedings of the 37th IEEE Conference on Decision and Control, pp. 2228-2233, 1998.

[21] Ioannou, P. A. and Sun, J., *Robust Adaptive Control*. Prentice Hall, New Jersey, 1996.

[22] Khaber, F., Zehar, K., and Hamzaoui, A., *State Feedback Controller Design via Takagi – Sugeno Fuzzy Model : LMI Approach*. International Journal of Computational Intelligence, Volume 2, No. 3, pp. 148-153, 2006.

[23] Kosko, B., and Isaka, S., *Fuzzy Logic*. Scientific American, July 1993.

[24] Kotlarski, J., de Nijs, R., Abdellatif, H., and Heimann, B., *New Interval – Based Approach to Determine the Guaranteed Singularity – Free Workspace of Parallel Robots*. Proceedings of the 2009 IEEE International Conference on Robotics and Automation, pp. 1256-1261, May 2009.

[25] Kovecses, J., Piedboeuf, J., C., and Lange, C., *Dynamics Modeling and Simulation of Constrained Robotic Systems*. IEEE/ASME Transactions on Mechatronics, Volume 8, No. 2, pp. 165-177, June 2003.

[26] Krstic, M., Kanellakopoulos, I., and Kokotovic, P.V., *Nonlinear and Adaptive Control Design*. Wiley, New York, 1995.

[27] Lan, L., H., *Stability Analysis for a Class of Takagi – Sugeno Fuzzy Control Systems with PID Controllers*. International Journal of Approximate Reasoning, Volume 46, pp. 109-119, 2007.

[28] Lee, K., H., *First Course on Fuzzy Theory and Applications*. Springer, Berlin, 2005.

[29] Li, C., Sun, L., Qu, D., and Liu, Y., *Development of Robot System for Sensor Adjustment in ICF Research*. Proceedings of the 2006 IEEE International Conference on Mechatronics and Automation, pp. 20482052, June 2006.

[30] Li, Q., and Wu, F., X., *Control Performance Improvement of a Parallel Robot via the Design for Control Approach*. Mechatronics, Volume 14, pp. 947-964, 2004.

[31] Lou, Y., Feng, F., and Li, Z., *Dynamics Based Trajectory Planning for Parallel Manipulators*. Proceedings of the 2009 IEEE International Conference on Information and Automation, pp. 295-300, June 2009.

[32] Lu, Y., and Hu, B., *Determining Singularity of Parallel Manipulators with n Linear Active Legs by CAD Variation Geometry*. International Journal of Robotics and Automation, Volume 23, No. 3, pp. 160-167, 2008.

[33] Merlet, J. P., *Parallel Robots - Second Edition*. Springer, Netherlands, 2006.

[34] Mustafa, M., Misuari, R., and Daniyal, H., *Forward Kinematics of* 3 *Degree of Freedom Delta Robot*. 5th Student Conference on Research and Development, December 2007.

[35] Mutka, A., Draganjac, I., Kovacic, Z., Postruzin, Z., and Munk, R., *Control System for Reactor Vessel Inspection Manipulator*. Proceedings of the 18th IEEE International Conference on Control Applications, pp. 1312-1318, July 2009.

[36] Nof, S. Y., *Handbook of Industrial Robotics - Second Edition*. John Wiley and Sons, New Jersey, 1999.

[37] Ordonez, R., and Passino, K., *Stable Multi−Input Multi−Output Adaptive Fuzzy/Neural Control*. IEEE Transactions on Fuzzy Systems, Volume 7, No. 3, pp. 345-353, June 1999.

[38] *Our Fascinating Look at Building Automation History Starts in* 270 *B.C.* Building Automation Consultants. 2008. June 17, 2010. http://www.building-automation-consultants.com/building-automation-history.html.

[39] Peng, B., Zeng, L., Sun, Y., and Chen, X., *A Novel High Rigid* 2 − *DOF Parallel Translating Robot*. 2nd International Conference on Intelligent Computation Technology and Automation, pp. 375-379, 2009.

[40] Pierrot, F., Fournier, A., and Dauchez, P., *Towards a Fully − Parallel* 6 *DOF Robot for High − Speed Applications*. Proceedings of the 1991 IEEE International Conference on Robotics and Automation, pp. 1288-1293, April 1991.

[41] Rhee, B., J., and Won, S., *A New Fuzzy Lyapunov Function Approach for a Takagi−Sugeno Fuzzy Control System Design*. Fuzzy Sets and Systems, Volume 157, pp. 1211-1228, 2006.

[42] Robot. *Merriam − Webster Online*. 2010. May 3, 2010. http://www.merriam-webster.com/dictionary/robot.

[43] Salgado, P., and Gouveia, F., *Derivative Information from Fuzzy Models*. Springer, Berlin, pp. 61-68, 2007.

[44] Sanz, A., and Etxebarria, V., *Interconnection* and *Damping Assignment Passivity−Based Experimental Control of a Single−Link Flexible Robot Arm*. Proceedings of the 2006 IEEE International Conference on Control Applications, pp. 2504-2509, October 2006.

[45] Sartori Natal, G., Chemori, A., Pierrot, F., and Company O., *Nonlinear Dual Mode Adaptive Control of PAR*2 : *A* 2 − *DOF Planar Parallel Manipulator*, *with Real − Time Experiments*. Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2114-2119, October 2009.

[46] Sciavicco, L., and Siciliano, B., *Modeling and Control of Robot Manipulators*. McGraw-Hill, New York, 1996.

[47] Seung, S., Kang, B., Je, H., Park, J., Kim, K., and Park, S., *Tele−Operation Master−Slave System for Minimal Invasive Brain Surgery*. Proceedings of the 2009 IEEE International Conference on Robotics and Biomimetics, pp. 177-182, December 2009.

[48] Shaocheng, T., and Yongming, L., *Direct Adaptive Fuzzy Backstepping Control for Nonlinear Systems*. Proceedings of the 1st International Conference on Innovative Computing, Information and Control, 2006.

[49] Sheng, N., Tong, S., and Zhang, W., *Adaptive Fuzzy Backstepping Control for a Class of Strict − Feedback Nonlinear Systems with Unknown Time − Delay*. Chinese Control and Decision Conference, pp. 105-110, 2009.

[50] Silva, L., A., Sebastian, J., M., Saltaren, R., Aracil R., and Sanpedro, J., *RoboTenis : Optimal Design of a Parallel Robot with High Performance*.

[51] Sivanandam, S., N., Sumathi, S., and Deepa, S., N., *Introduction to Fuzzy Logic using MATLAB*. Springer, Berlin, 2007.

[52] Stan, S., D., Maties, V., Balan R., and Lapusan, C., *Genetic Algorithms to Optimal Design of a 3 DOF Parallel Robot*. IEEE 2008.

[53] Stewart, D., *A Platform with Six Degrees of Freedom*. Proceedings of the Institution of Mechanical Engineers, Volume 180, Part 1, No. 15, pp. 371-385, 1965-66.

[54] Sugeno, M., and Kang, G., T., *Structure Identification of Fuzzy Model*. Fuzzy Sets and Systems, Volume 28, pp. 15-33, 1988.

[55] Takagi, T., and Sugeno, M., *Fuzzy Identification of Systems and its Applications to Modeling and Control*. IEEE Transactions on Systems, Man and Cybernetics, Volume 15, pp. 116-132, 1985.

[56] Tanaka, K., and Sugeno, M., *Stability Analysis and Design of Fuzzy Control Systems*. Fuzzy Sets and Systems, Volume 45, pp. 135-156, 1992.

[57] Tanaka, K., Hori, T., and Wang, H., O., *A Multiple Lyapunov Function Approach to Stabilization of Fuzzy Control Systems*. IEEE Transactions on Fuzzy Systems, Volume 11, No. 4, pp. 582-589, August 2003.

[58] *Timeline of Computer History - Robots and Artificial Intelligence*. Computer History Museum. 2006. May 8, 2010. http://www.computerhistory.org/timeline/?category=rai.

[59] Tong, S., H., *Indirect Adaptive Fuzzy Backstepping Control for Nonlinear Systems*. Proceedings of the 5th International Conference on Machine Learning and Cybernetics, pp. 468-473, August 2006.

[60] Tsai, L., W., *Solving the Inverse Dynamics of a Stewart – Gough Manipulator by the Principle of Virtual Work*. ASME Journal of Mechanical Design, Volume 122, pp. 3-9, 2000.

[61] Wang, H., Liu, X., and Liadis, A., *A Network Controlled Planar Parallel Robot*. Proceedings of the 2010 IEEE International Conference on Mechatronics and Automation, August 2010.

[62] Wang, H., O., Tanaka, K., and Griffin, M., F., *An Approach to Fuzzy Control of Nonlinear Systems - Stability and Design Issues*. IEEE Transactions on Fuzzy Systems, Volume 4, No. 1, pp. 14-23, February 1996.

[63] Wang, L., *Adaptive Control of a Parallel Robot via Backstepping Technique*. Masters thesis, Lakehead University, Thunder Bay, Ontario, Canada, 2006.

[64] Wang, L., X., *A Course in Fuzzy Systems and Control*. Prentice Hall, New Jersey, 1997.

[65] Wang, L., X., *Adaptive Fuzzy Systems and Control*. Prentice Hall, New Jersey, 1994.

[66] Wang, Z., and Ghorbel, F., *Control of Closed Kinematic Chains using a Singularly Perturbed Dynamic Model*. Proceedings of the 43rd IEEE Conference on Decision and Control, pp. 317-322, December 2004.

[67] Wei, L., X., XIA Wang, X., Wang, H., R., and Wang, P., G., *Adaptive Backstepping Fuzzy Control for* $X-Y$ *Table with Friction*. Proceedings of the Third International Conference on Machine Leaming and Cybernetics, pp. 683-686, 2004.

[68] Wolovich, W., *Robotics : Basic Analysis and Design*. CBS College Publishing, New York, 1987.

[69] Xiu, Z., H., and Ren, G., *Stability Analysis and Systematic Design of Takagi − Sugeno Fuzzy Control Systems*. Fuzzy Sets and Systems, Volume 151, pp. 119-138, 2005.

[70] Yang, C., He, J., Jiang, H., and Han, J., *Modeling and Simulation of 6 − DOF Parallel Manipulator Based on PID Control with Gravity Compensation in Simulink/ADAMS*. International Workshop on Modelling, Simulation and Optimization, pp. 391-395, 2008.

[71] Yang, Y., and OBrien, J., F., *A Geometric Approach for the Design of Singularity − Free Parallel Robots*. Proceedings of the 2009 IEEE International Conference on Robotics and Automation, pp. 1801-1806, May 2009.

[72] Yu, Y., Yi, J., Li, C., Zhao, D., and Zhang, J., *Fuzzy Logic Based Adjustment Control of a Cable−Driven Auto−Leveling Parallel Robot*. The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2102-2107, October 2009.

[73] Yun, Y., and Li, Y., *Performance Analysis and Optimization of a Novel Large Displacement* 3 − *DOF Parallel Manipulator*. Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics, pp. 246-251, February 2009.

[74] Zadeh, L., A., *Fuzzy Sets*. Information and Control. Volume 8, No. 3, pp. 338-353, June 1965.

[75] Zeinali, M., and Notash, L., *Fuzzy Model−Based Adaptive Robust Control for Parallel Manipulators*. 12th IFToMM World Congress, June 2007.

[76] Zhou, J., and Wen, C., *Adaptive Backstepping Control of Uncertain Systems*. Springer, Berlin, 2008.

[77] Zunt, D., *Who did actually invent the word robot and what does it mean*? Karel Capek. 2004. May 3, 2010. http://capek.misto.cz/english/robot.html.

# Appendix A

# DSP Circuit Board Layout

This appendix will entail specific details about the DSP board. It will consist of the pinout tables of the DSP board followed by the schematic and PCB layout of the DSP board using the EAGLE Layout Editor (Version 5.4). The schematic diagram of the DSP board will be split up into smaller segments in order to accurately see the connections between the components. The PCB layout for the DSP board will be split up into three segments in order to visualize the trace connections between the components.

**ADC_LS**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 174 | 172 | 170 | 168 | 2 | 4 | 6 | 8 | - | 1, 13, 14, 166 |
| ADCINA0 | ADCINA2 | ADCINA4 | ADCINA6 | ADCINB0 | ADCINB2 | ADCINB4 | ADCINB6 | - | $V_{DDAIO}$, AVDDREF, $V_{DDA1}$, $V_{DDA2}$ |
| PF1 | PF3 | PF5 | FF1 | FF3 | CF1 | CF3 | CF5 | LS | 3.3V |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| PF 2 | PF 4 | PF 6 | FF2 | FF4 | CF2 | CF4 | CF6 | AGND | 3.3V |
| ADCINA1 | ADCINA3 | ADCINA5 | ADCINA7 | ADCINB1 | ADCINB3 | ADCINB5 | ADCINB7 | AVSSREF, $V_{SSA1}$, $V_{SSA2}$, $V_{SSAIO}$ | $V_{DDAIO}$, AVDDREF, $V_{DDA1}$, $V_{DDA2}$ |
| 173 | 171 | 169 | 167 | 3 | 5 | 7 | 9 | 12, 15, 165, 176 | 1, 13, 14, 166 |

PF = Position Feedback; FF = Force Feedback; CF = Current Feedback; LS = Limit Switch; AGND = Analog Ground; 3.3V = Analog Power

**DIR_PWM**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 116 | 122 | 124 | 92 | 98 | 47 | 28 | 26 | 22 | 72 |
| TDIRA (GPIOA11) | \|C1TRIP\| (GPIOA13) | \|C3TRIP\| (GPIOA15) | PWM1 (GPIOA0) | PWM5 (GPIOA4) | PWM9 (GPIOB2) | MCLKXA (GPIOF8) | MFSXA (GPIOF10) | MDXA (GPIOF12) | TCLKINB (GPIOB12) |
| DIR1 | DIR3 | DIR5 | PWM1 | PWM5 | PWM9 | TF1 | TF3 | TF5 | Break |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| DIR2 | DIR4 | DIR6 | PWM3 | PWM7 | PWM11 | TF2 | TF4 | TF6 | Relay Out |
| TCLKINA (GPIOA12) | \|C2TRIP\| (GPIOA14) | TDIRB (GPIOB11) | PWM3 (GPIOA2) | PWM7 (GPIOB0) | PWM11 (GPIOB4) | MCLKRA (GPIOF9) | MFSRA (GPIOF11) | MDRA (GPIOF13) | - - |
| 117 | 123 | 71 | 94 | 45 | 49 | 25 | 29 | 20 | - |

PWM = Pulse Width Modulation; DIR = Direction; TF = Thermal Flag;

**QEP**

| | | | | |
|---|---|---|---|---|
| 106 | 109 | 59 | 31, 64, 81, 114, 145 | 69 |
| CAP1_QEP1 (GPIOA8) | CAP3_QEPI1 (GPIOA10) | CAP5_QEP4 (GPIOB9) | $V_{DDIO}$ | $V_{DD3VFL}$ |
| QEP1 | QEPI1 | QEP4 | 3.3V | 3.3V |
| 1 | 3 | 5 | 7 | 9 |
| 2 | 4 | 6 | 8 | 10 |
| QEP2 | QEP3 | QEPI2 | GND | GND |
| CAP2_QEP2 (GPIOA9) | CAP4_QEP3 (GPIOB8) | CAP6_QEPI2 (GPIOB10) | $V_{SS}$ | TESTSEL, $V_{SS1}$ |
| 107 | 57 | 60 | 19, 32, 38, 52, 58, 70, 78, 86, 99, 105, 113, 120, 129, 142, 153 | 134, 163 |

3.3V and $V_{SS}$ = Digital Power; QEP = Quadrature Encoder Pulse; GND = Digital Ground;

Table A.1: DSP Board Pinout 1

**SPI_RT**

| | | | | |
|---|---|---|---|---|
| 40 | 34 | 53 | 61 | 115 |
| SPISIMOA (GPIOF0) | SPICLKA (GPIOF2) | T3PWM_T3CMP (GPIOB6) | \|C4TRIP\| (GPIOB13) | \|T2CTRIP\| / \|EVASOC\| (GPIOD1) |
| SPISIMOA | SPICLKA | SPISTEA2 | RT1 | RT3 |
| 1 | 3 | 5 | 7 | 9 |
| 2 | 4 | 6 | 8 | 10 |
| SPISOMIA | SPISTEA1 | SPISTEA3 | RT2 | RT4 |
| SPISOMIA (GPIOF1) | SPISTEA (GPIOF3) | T4PWM_T4CMP (GPIOB7) | \|T1CTRIP_PDPINTA\| (GPIOD0) | \|T3CTRIP\| / \|PDPINTB\| (GPIOD5) |
| 41 | 35 | 55 | 110 | 79 |

RT = Relay Trigger

**JTAG**

| | | | | | | |
|---|---|---|---|---|---|---|
| 126 | 131 | Same as QEP | 127 | 136 | 136 | 137 |
| TMS | TDI | $V_{DDIO}$ | TDO | TCK | TCK | EMU0 |
| TMS | TDI | PD | TDO | TCK_RET | TCK | EMU0 |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 |
| TRST | TMS/TDI | GND | GND | GND | GND | EMU1 |
| \|TRST\| | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | EMU1 |
| 135 | Same as QEP | Same as QEP | Same as QEP | Same as QEP | Same as QEP | 146 |

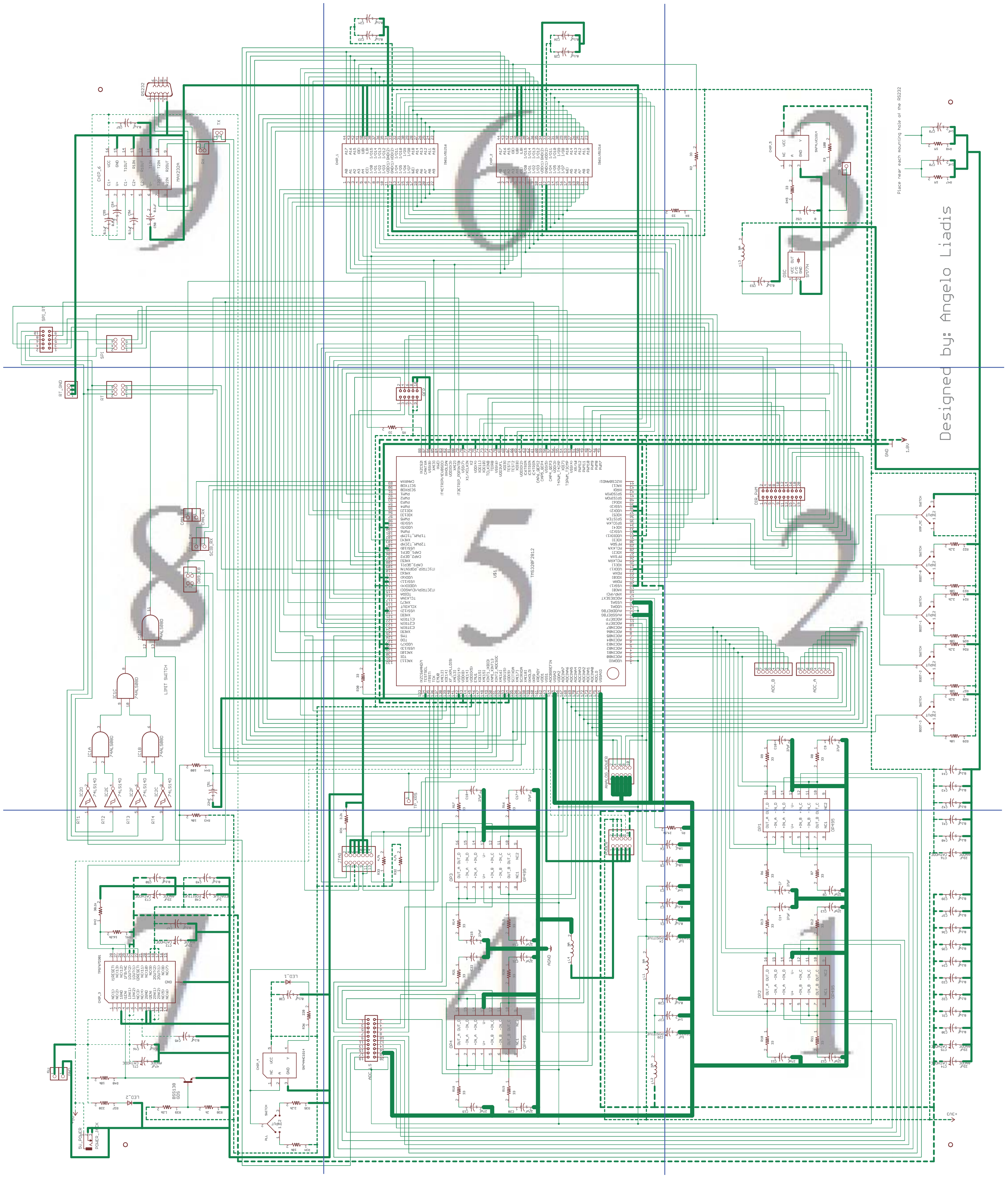
**POWER**

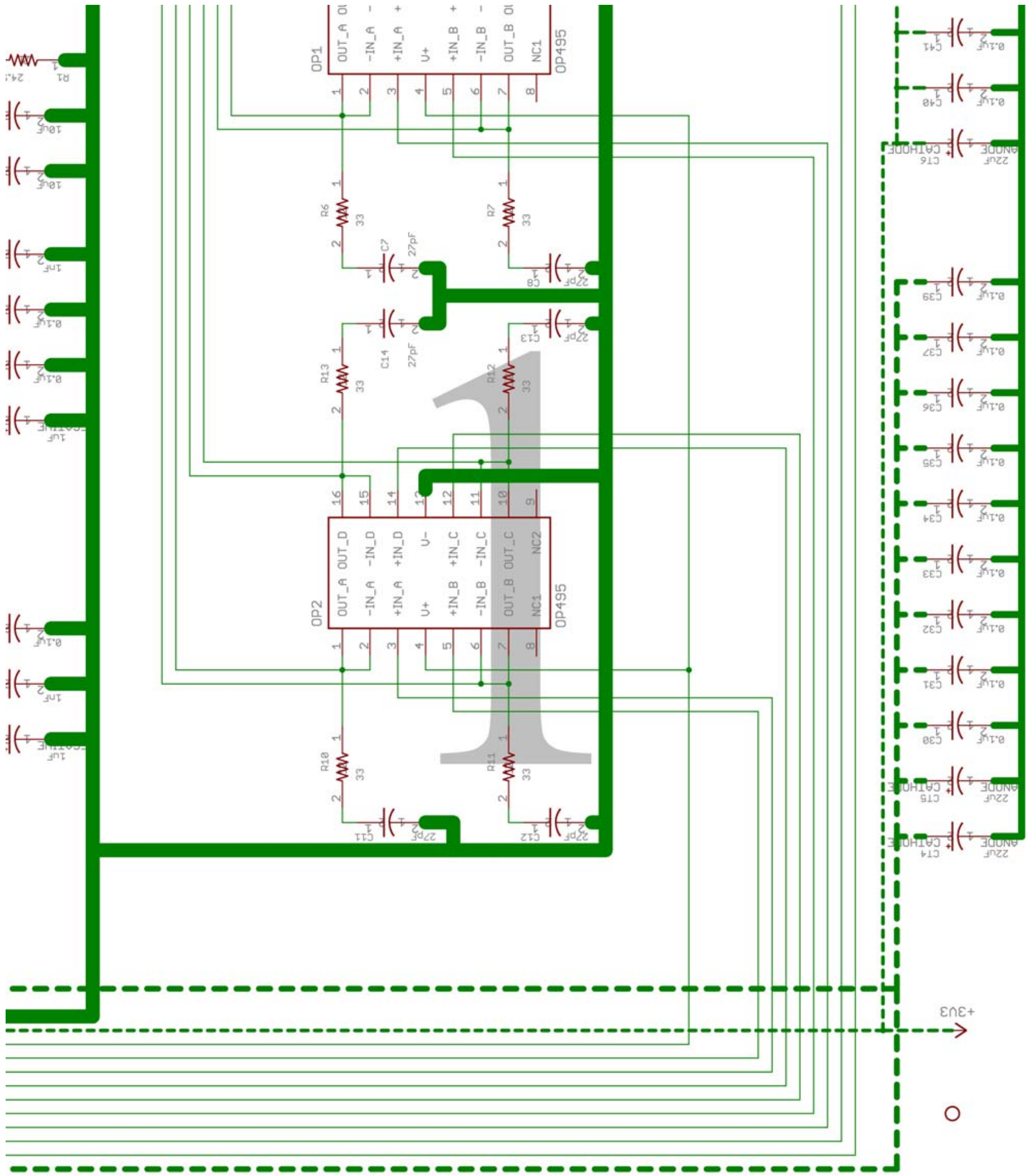
| | | | | |
|---|---|---|---|---|
| Same as QEP | Same as QEP | Same as QEP | Same as QEP | Same as QEP |
| $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ | $V_{SS}$ |
| GND | GND | GND | GND | GND |
| 1 | 3 | 5 | 7 | 9 |
| 2 | 4 | 6 | 8 | 10 |
| 3.3V | 3.3V | 3.3V | 5V | 1.8V |
| $V_{DDIO}$ | $V_{DDIO}$ | $V_{DDIO}$ | - | $V_{DD}$, $V_{DD1}$ |
| Same as QEP | Same as QEP | Same as QEP | - | 23, 37, 56, 75, 100, 112, 128, 143, 154, 162 |

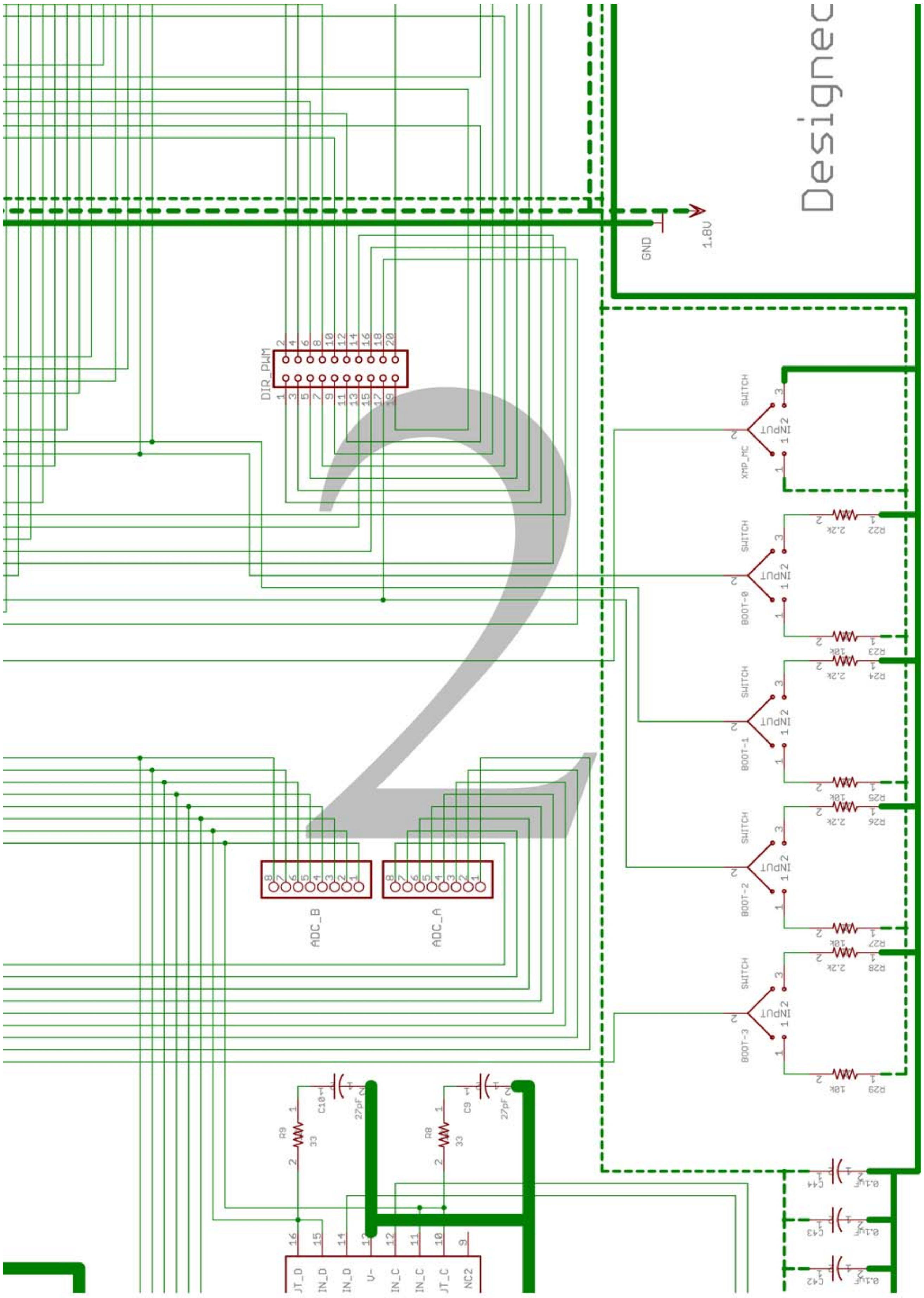
**ANALOG_POWER**

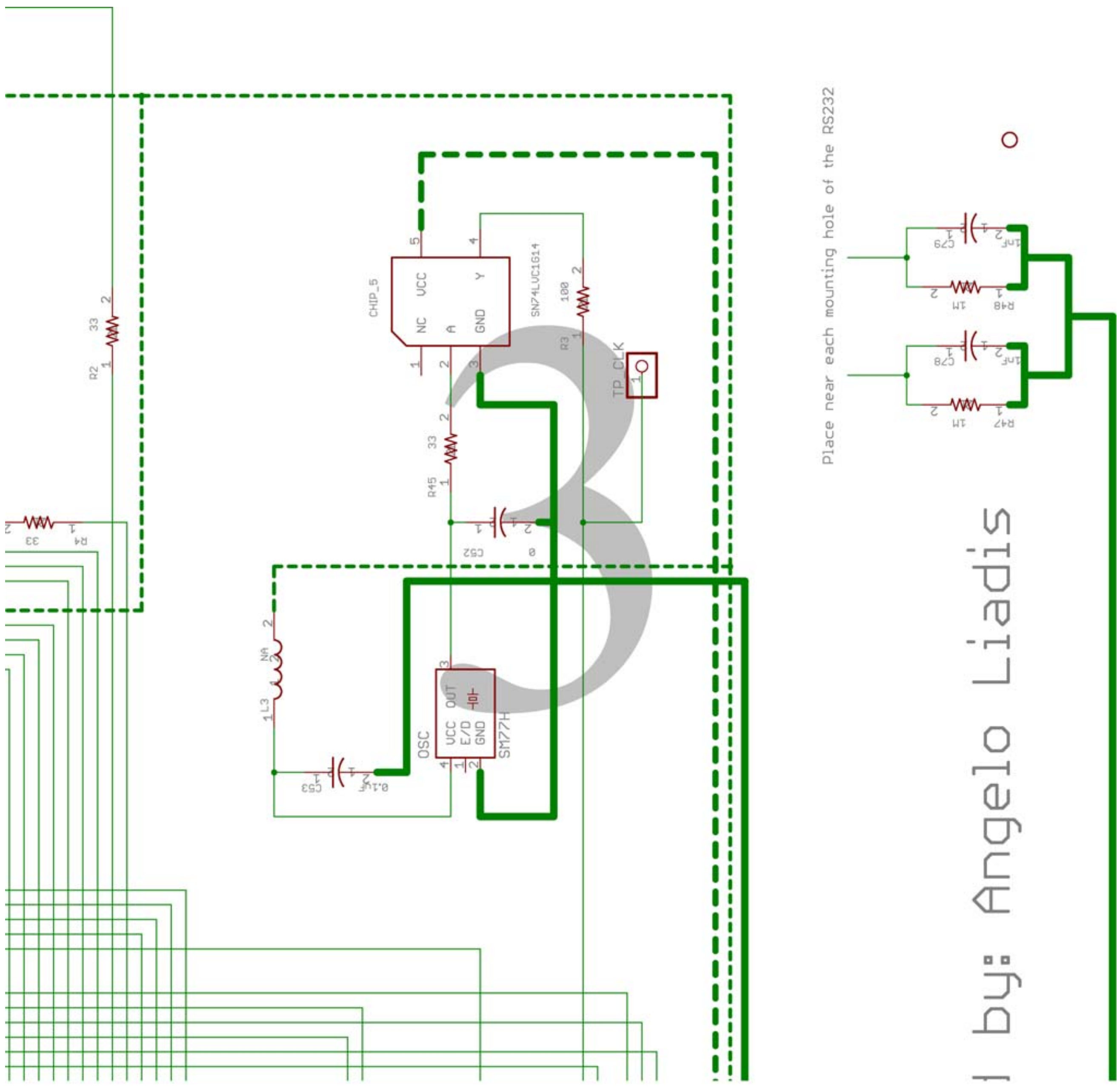
| | | | | |
|---|---|---|---|---|
| 12, 15, 165, 176 | 12, 15, 165, 176 | 12, 15, 165, 176 | 12, 15, 165, 176 | 12, 15, 165, 176 |
| AVSSREF, $V_{SSA1}$, $V_{SSA2}$, $V_{SSAIO}$ | AVSSREF, $V_{SSA1}$, $V_{SSA2}$, $V_{SSAIO}$ | AVSSREF, $V_{SSA1}$, $V_{SSA2}$, $V_{SSAIO}$ | AVSSREF, $V_{SSA1}$, $V_{SSA2}$, $V_{SSAIO}$ | AVSSREF, $V_{SSA1}$, $V_{SSA2}$, $V_{SSAIO}$ |
| AGND | AGND | AGND | AGND | AGND |
| 1 | 3 | 5 | 7 | 9 |
| 2 | 4 | 6 | 8 | 10 |
| 3.3V | 3.3V | VREFLO | ADCREFM | ADCREFP |
| $V_{DDAIO}$, AVDDREF, $V_{DDA1}$, $V_{DDA2}$ | $V_{DDAIO}$, AVDDREF, $V_{DDA1}$, $V_{DDA2}$ | ADCLO | ADCREFM | ADCREFP |
| 1, 13, 14, 166 | 1, 13, 14, 166 | 175 | 10 | 11 |

Table A.2: DSP Board Pinout 2

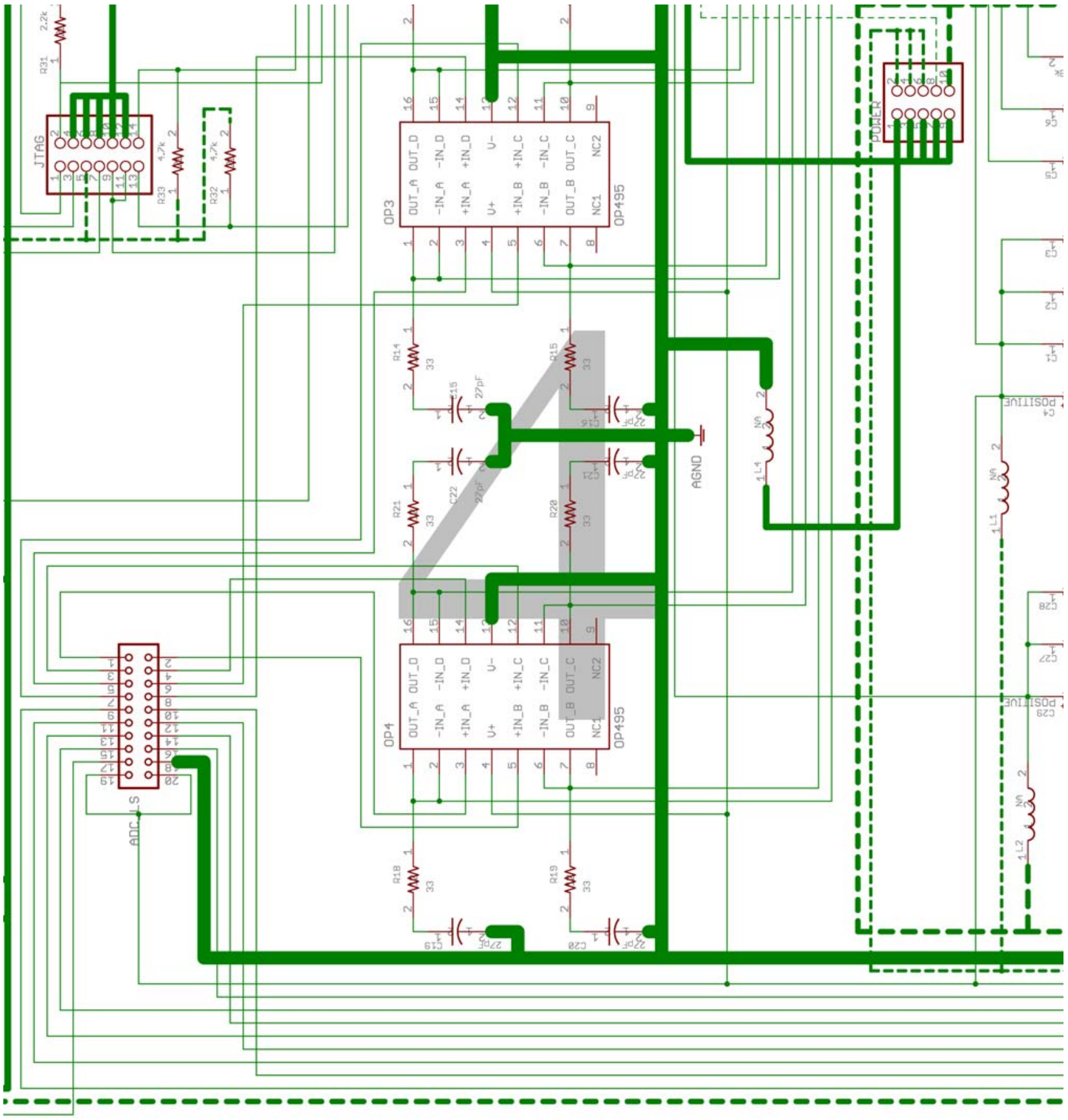

Designed by: Angelo Liadis
Place near each mounting hole of the RS232
TMS320F2812
MAX232N
RS232
LIMIT SWITCH
JTAG
SPI
1.8V
+3V3
GND


OP1
OP495
OUT_A
-IN_A
+IN_A
V+
+IN_B
-IN_B
OUT_B
NC1
R6
R7
R13
R12
R10
R11
33
C7
C14
27pF
C8
C13
C11
C12
22pF
OP2
OUT_D
-IN_D
+IN_D
V-
+IN_C
-IN_C
OUT_C
NC2
R1
10uF
1nF
0.1uF
1uF
C41
C40
C16
C39
C37
C36
C35
C34
C33
C32
C31
C30
C15
C14
22uF
ANODE
CATHODE
+3V3

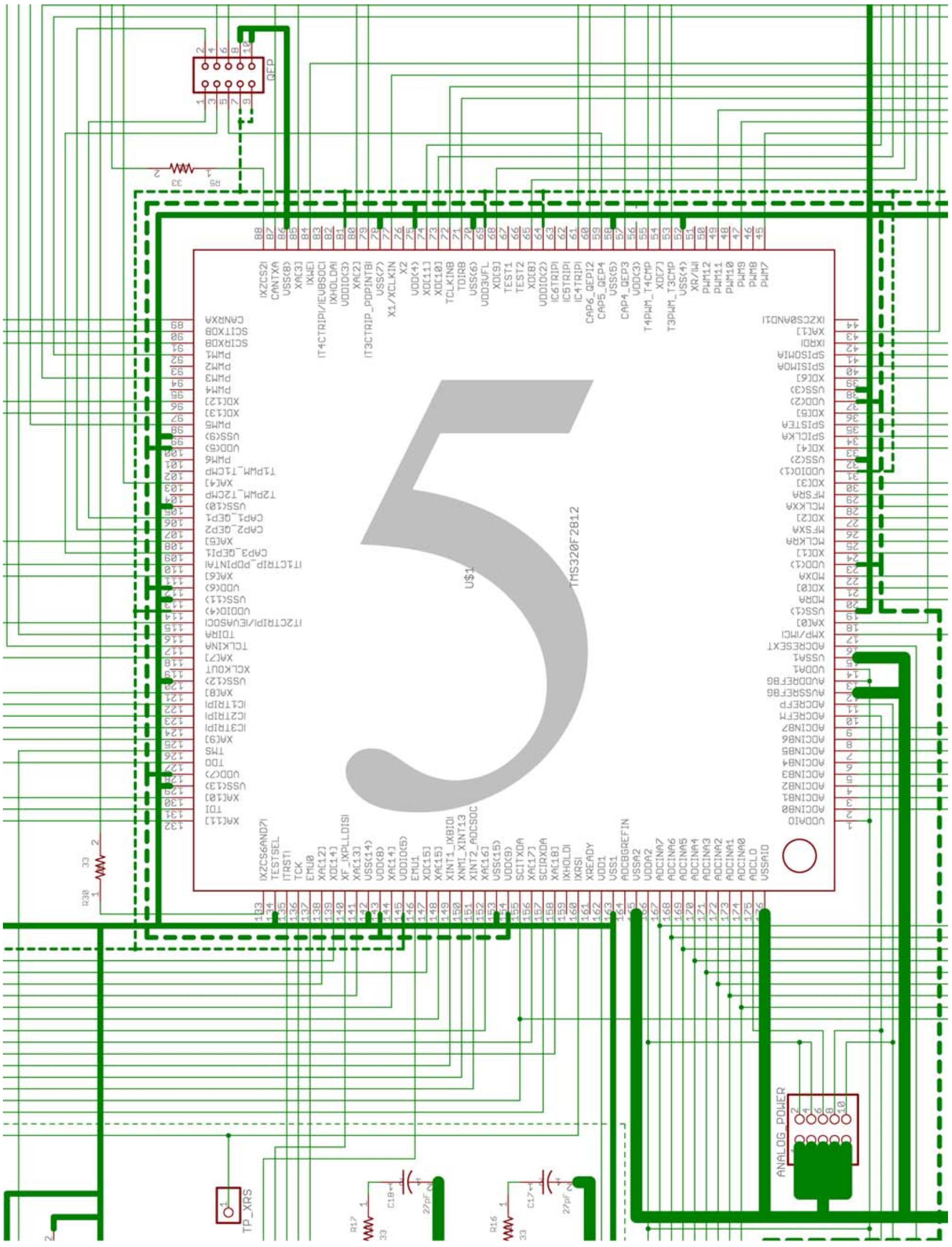

Designec
GND
1.8V
DIR_PWM
ADC_B
ADC_A
SWITCH
INPUT
XMP_MC
BOOT-0
BOOT-1
BOOT-2
BOOT-3
R22
R23
R24
R25
R26
R27
R28
R29
2.2k
10k
R8
R9
33
C9
C10
27pF
C42
C43
C44
0.1uF
JT_D
IN_D
V-
IN_C
JT_C
NC2



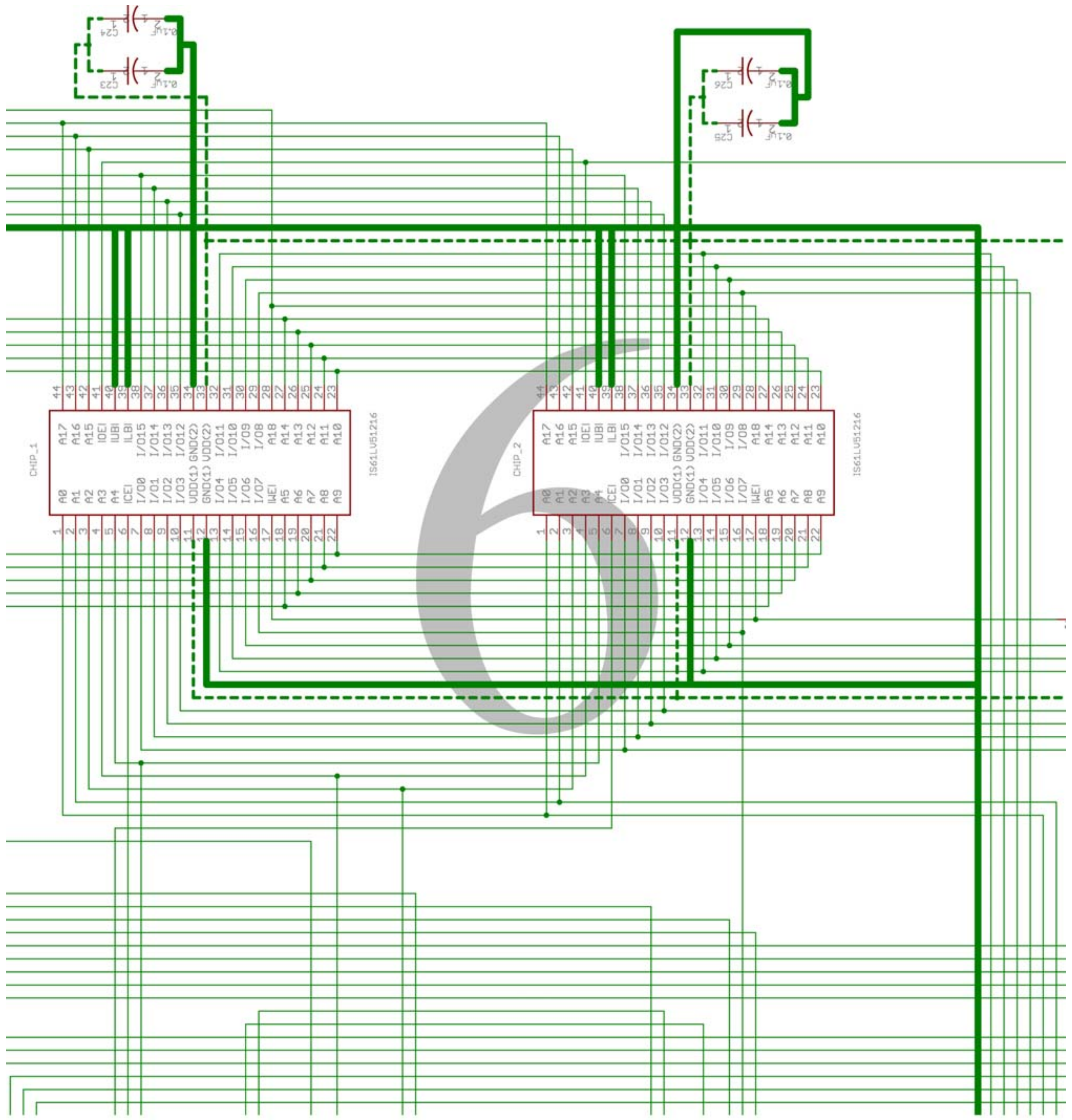
Place near each mounting hole of the RS232
l by: Angelo Liadis
CHIP_5
NC
VCC
A
GND
Y
SN74LVC1G14
R2
33
R3
100
R4
33
R45
33
C52
0
TP_CLK
OSC
VCC
E/D
GND
OUT
SM77H
L3
NA
C53
0.1uF
C79
1nF
C78
1nF
R48
1M
R47
1M
3

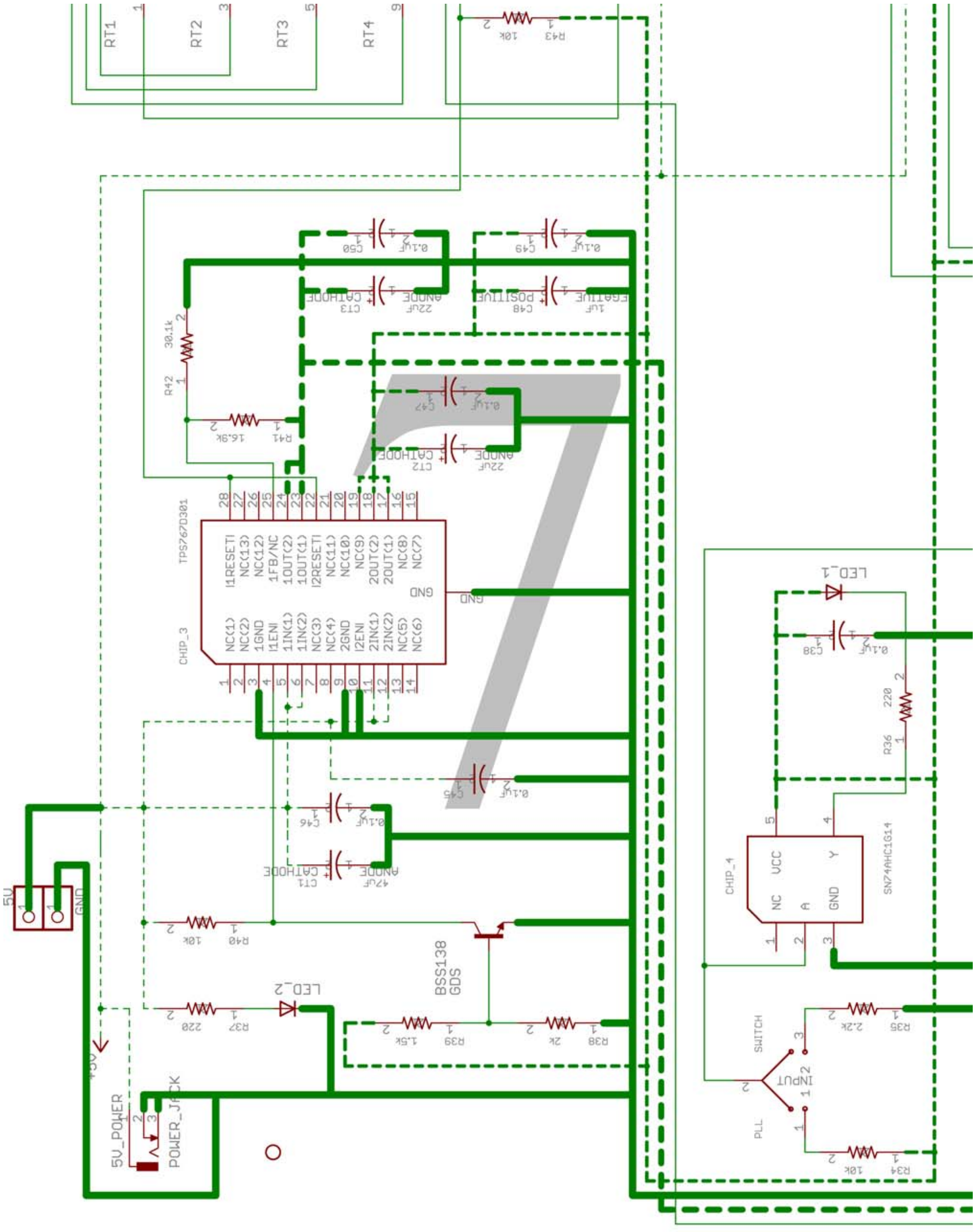


JTAG
R31
2.2k
R33
4.7k
R32
4.7k
OP3
OP495
OP4
OP495
OUT_A
-IN_A
+IN_A
V+
+IN_B
-IN_B
OUT_B
NC1
OUT_D
-IN_D
+IN_D
V-
+IN_C
-IN_C
OUT_C
NC2
R14
R15
R21
R20
R18
R19
33
C15
C16
C22
C21
C19
C20
27pF
22pF
AGND
L4
L1
L2
NA
POWER
ADC_LS
POSITIVE
C4
C29
C28
C27
C1
C2
C3
C5
C6

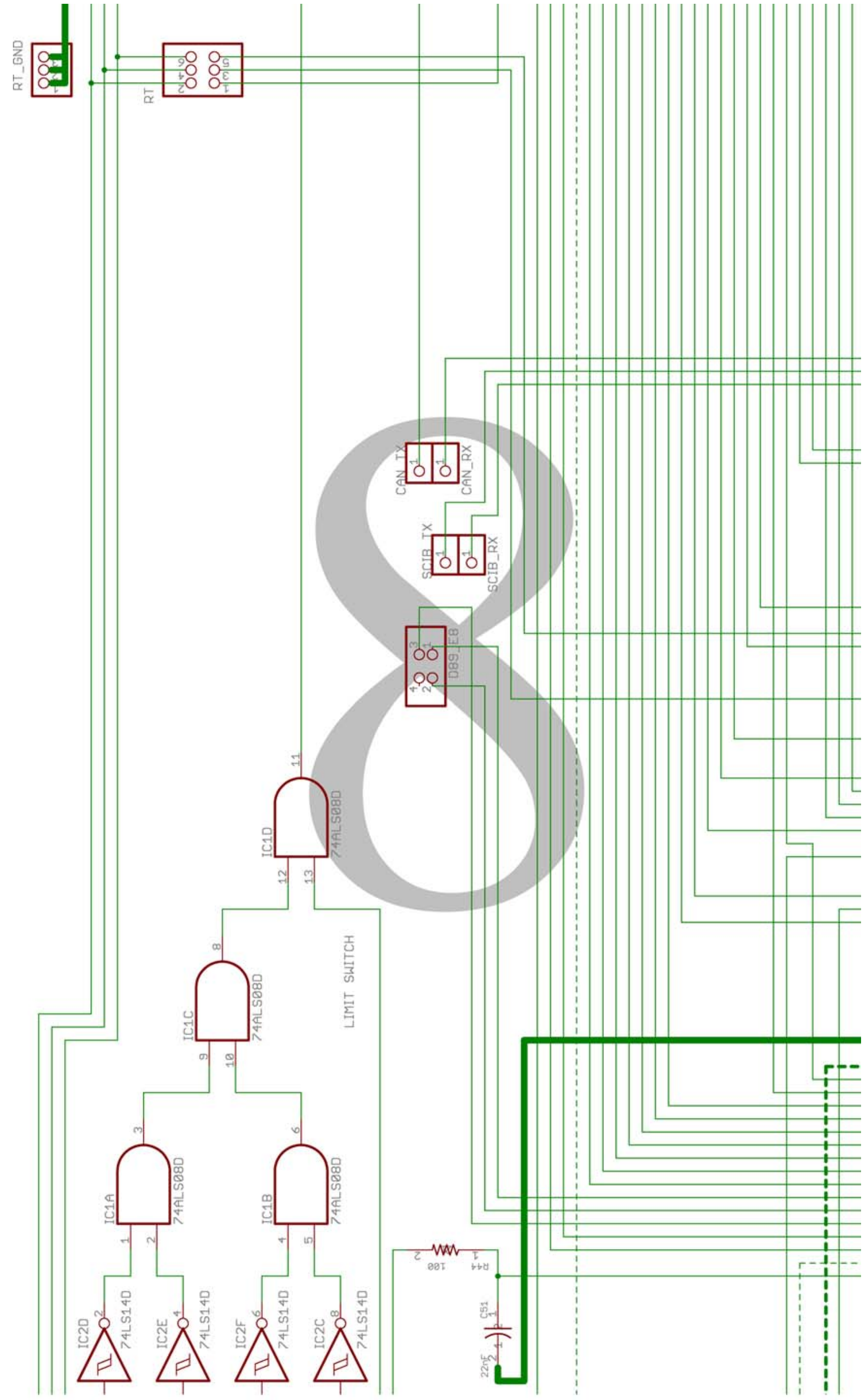


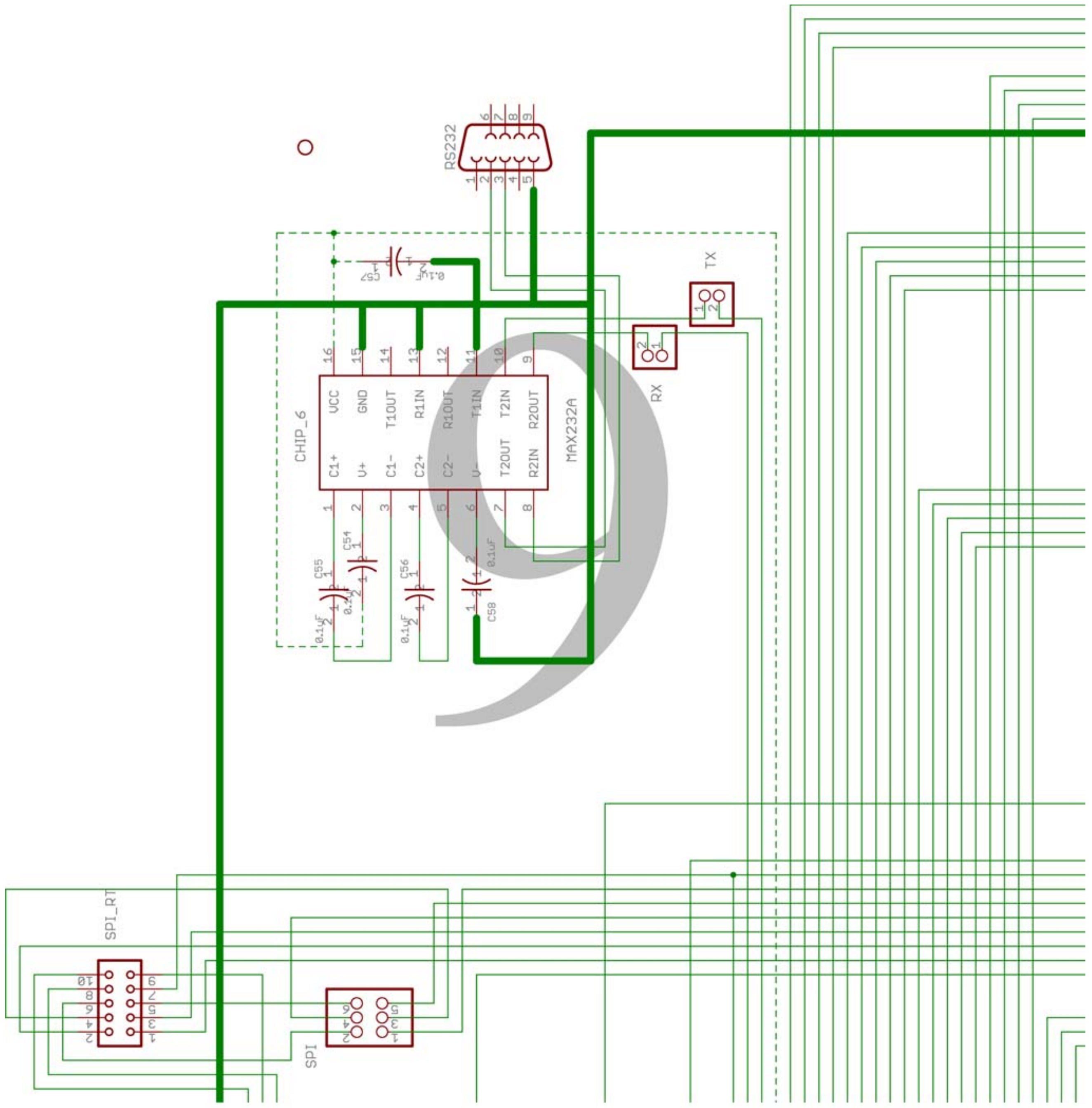
QFP
TMS320F2812
U$1
5
R5
R30
R16
R17
C17
C18
27pF
TP_XRS
ANALOG_POWER


C24
0.1uF
C23
0.1uF
C26
0.1uF
C25
0.1uF
CHIP_1
IS61LV51216
CHIP_2
IS61LV51216
A17 A16 A15 !OE! !UB! !LB! I/O15 I/O14 I/O13 I/O12 GND(2) VDD(2) I/O11 I/O10 I/O9 I/O8 A18 A14 A13 A12 A11 A10
A0 A1 A2 A3 A4 !CE! I/O0 I/O1 I/O2 I/O3 VDD(1) GND(1) I/O4 I/O5 I/O6 I/O7 !WE! A5 A6 A7 A8 A9

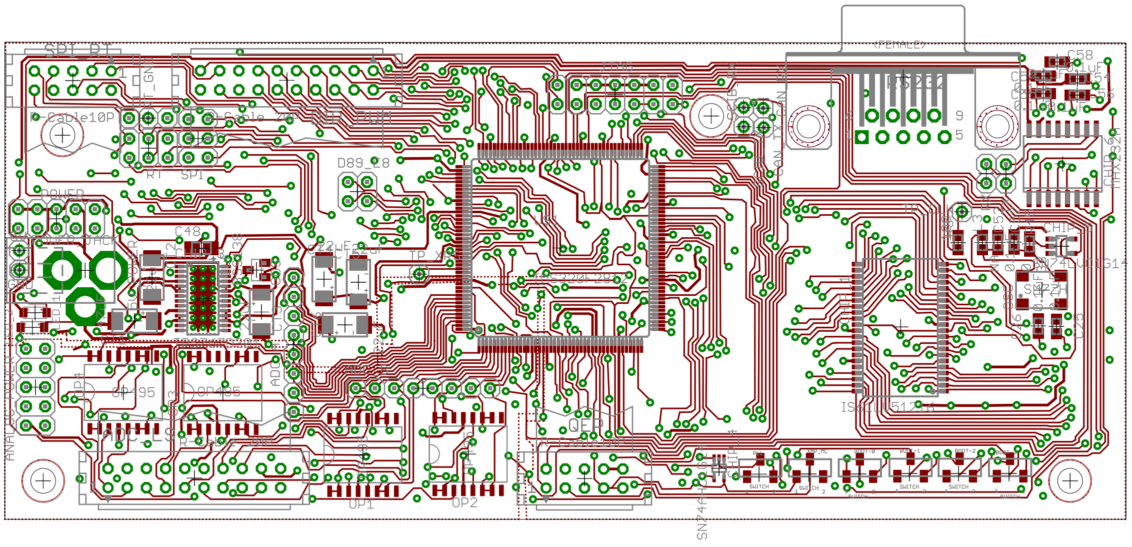


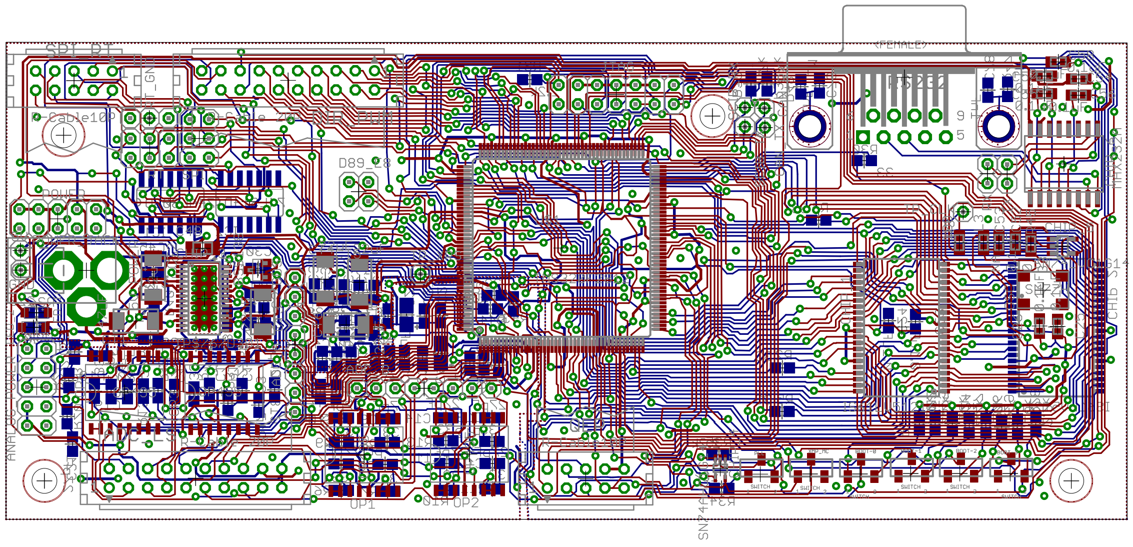
RT1
RT2
RT3
RT4
R43
10k
C50
0.1uF
C49
CT3
22uF
CATHODE
ANODE
C48
1uF
POSITIVE
NEGATIVE
R42
30.1k
R41
16.9k
C47
CT2
TPS767D301
CHIP_3
I1RESETI
NC(13)
NC(12)
1FB/NC
1OUT(2)
1OUT(1)
I2RESETI
NC(11)
NC(10)
NC(9)
2OUT(2)
2OUT(1)
NC(8)
NC(7)
GND
NC(1)
NC(2)
1GND
I1ENI
1IN(1)
1IN(2)
NC(3)
NC(4)
2GND
I2ENI
2IN(1)
2IN(2)
NC(5)
NC(6)
C45
C46
CT1
47uF
5V
GND
R40
BSS138
GDS
R37
220
LED_2
R39
1.5k
R38
2k
+5V
5V_POWER
POWER_JACK
LED_1
C38
R36
SN74AHC1G14
CHIP_4
VCC
NC
A
GND
Y
R35
2.2k
SWITCH
INPUT
PLL
R34

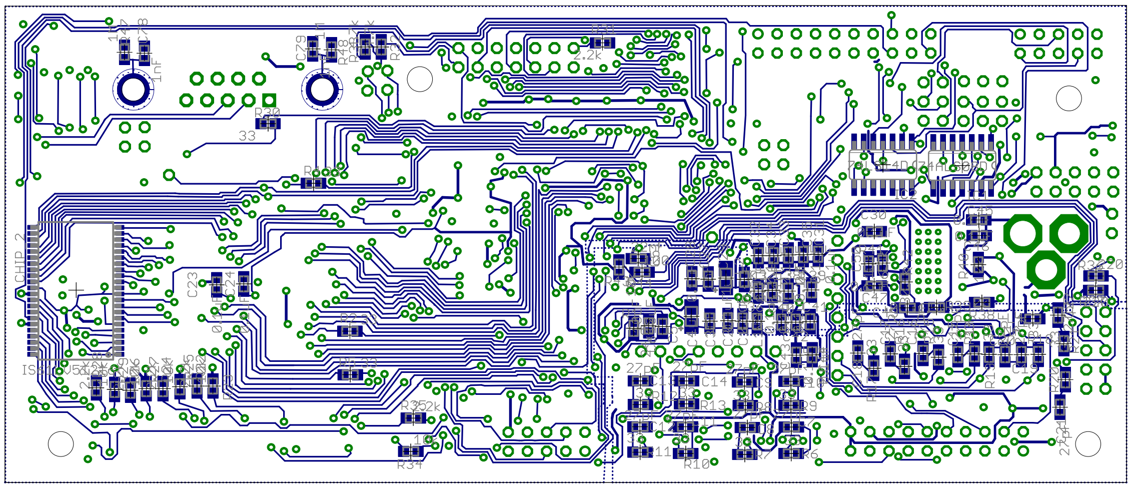

RT_GND
RT
CAN_TX
CAN_RX
SCIB_TX
SCIB_RX
DB9_E8
IC1D
74ALS08D
IC1C
74ALS08D
IC1A
74ALS08D
IC1B
74ALS08D
LIMIT SWITCH
IC2D
74LS14D
IC2E
74LS14D
IC2F
74LS14D
IC2C
74LS14D
R44
100
C51
22nF



RS232
TX
RX
CHIP_6
MAX232A
VCC
GND
T1OUT
R1IN
R1OUT
T1IN
T2IN
R2OUT
C1+
V+
C1-
C2+
C2-
V-
T2OUT
R2IN
C54
C55
C56
C57
C58
0.1uF
SPI_RT
SPI







The PCB layouts are as followed: Top Layer; Both Layers; Bottom Layer.

# Appendix B

# Motor Driver Circuit Board Layout

This appendix will entail specific details about the motor driver board. It will consist of the pinout tables of the motor driver board followed by the schematic and PCB layout of the motor driver board using the EAGLE Layout Editor (Version 5.4). The schematic diagram of the motor driver board will be split up into smaller segments in order to accurately see the connections between the components. The PCB layout for the motor driver board will be split up into three segments in order to visualize the trace connections between the components.

**DIR_PWM_TF_BRK_REL (CON3)**

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| IC1 - #3 | IC3 - #3 | IC5 - #3 | IC1 - #5 | IC3 - #5 | IC5 - #5 | IC1 - #9 | IC3 - #9 | IC5 - #9 | 72 |
| DIR1 (GPIOA11) | DIR3 (GPIOA13) | DIR5 (GPIOA15) | PWM1 (GPIOA0) | PWM3 (GPIOA4) | PWM5 (GPIOB2) | TF1 (GPIOF8) | TF3 (GPIOF10) | TF5 (GPIOF12) | Break (GPIOB12) |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 |
| 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| DIR2 (GPIOA12) | DIR4 (GPIOA14) | DIR6 (GPIOB11) | PWM2 (GPIOA2) | PWM4 (GPIOB0) | PWM6 (GPIOB4) | TF2 (GPIOF9) | TF4 (GPIOF11) | TF6 (GPIOF13) | Relay Output |
| IC2 - #3 | IC4 - #3 | IC6 - #3 | IC2 - #5 | IC4 - #5 | IC6 - #5 | IC2 - #9 | IC4 - #9 | IC6 - #9 | - |

PWM = Pulse Width Modulation; DIR = Direction; TF = Thermal Flag

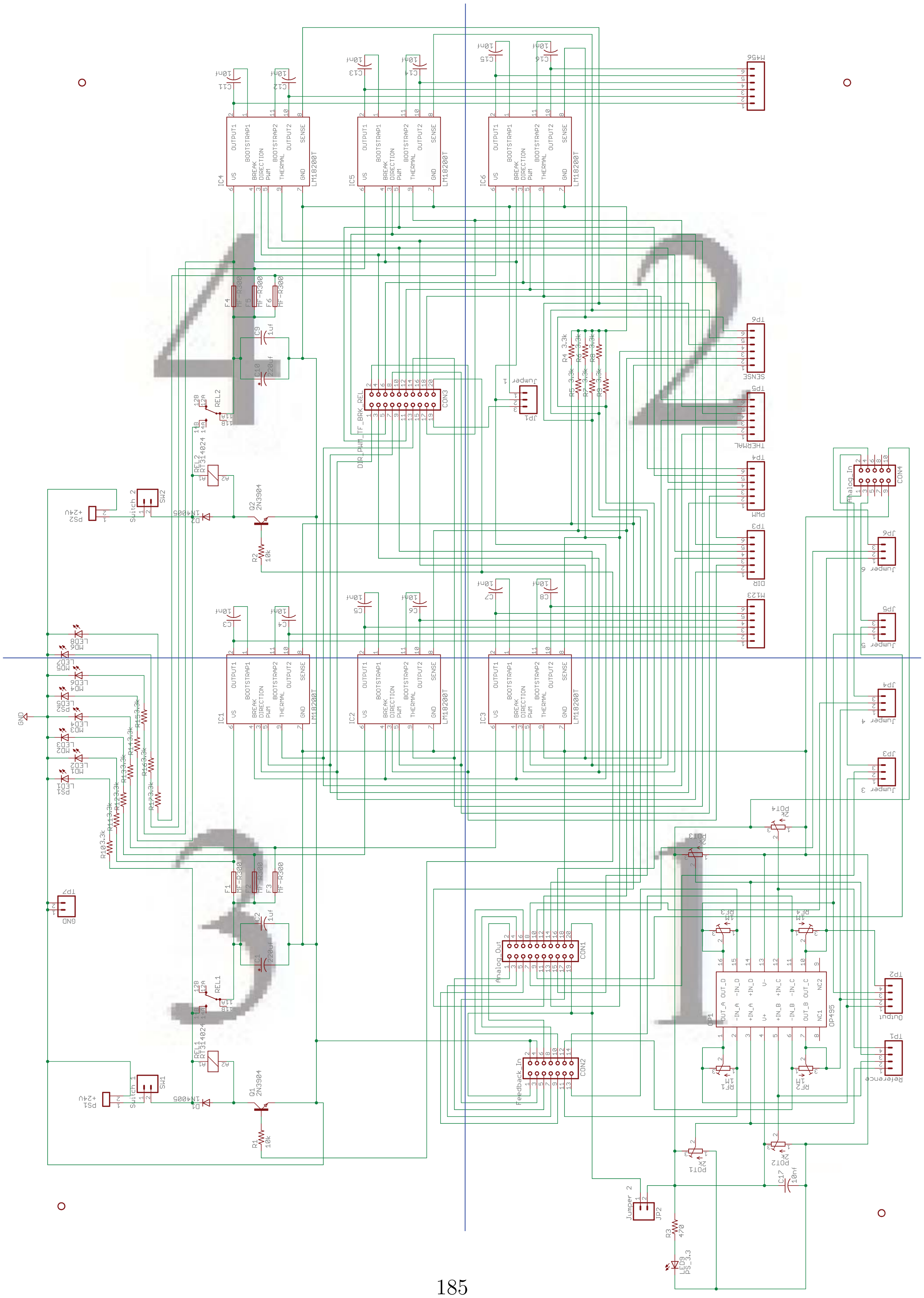
**M123**

| | | | | | |
|---|---|---|---|---|---|
| Motor 1+ | Motor 1- | Motor 2+ | Motor 2- | Motor 3+ | Motor 3- |
| 1 | 2 | 3 | 4 | 5 | 6 |

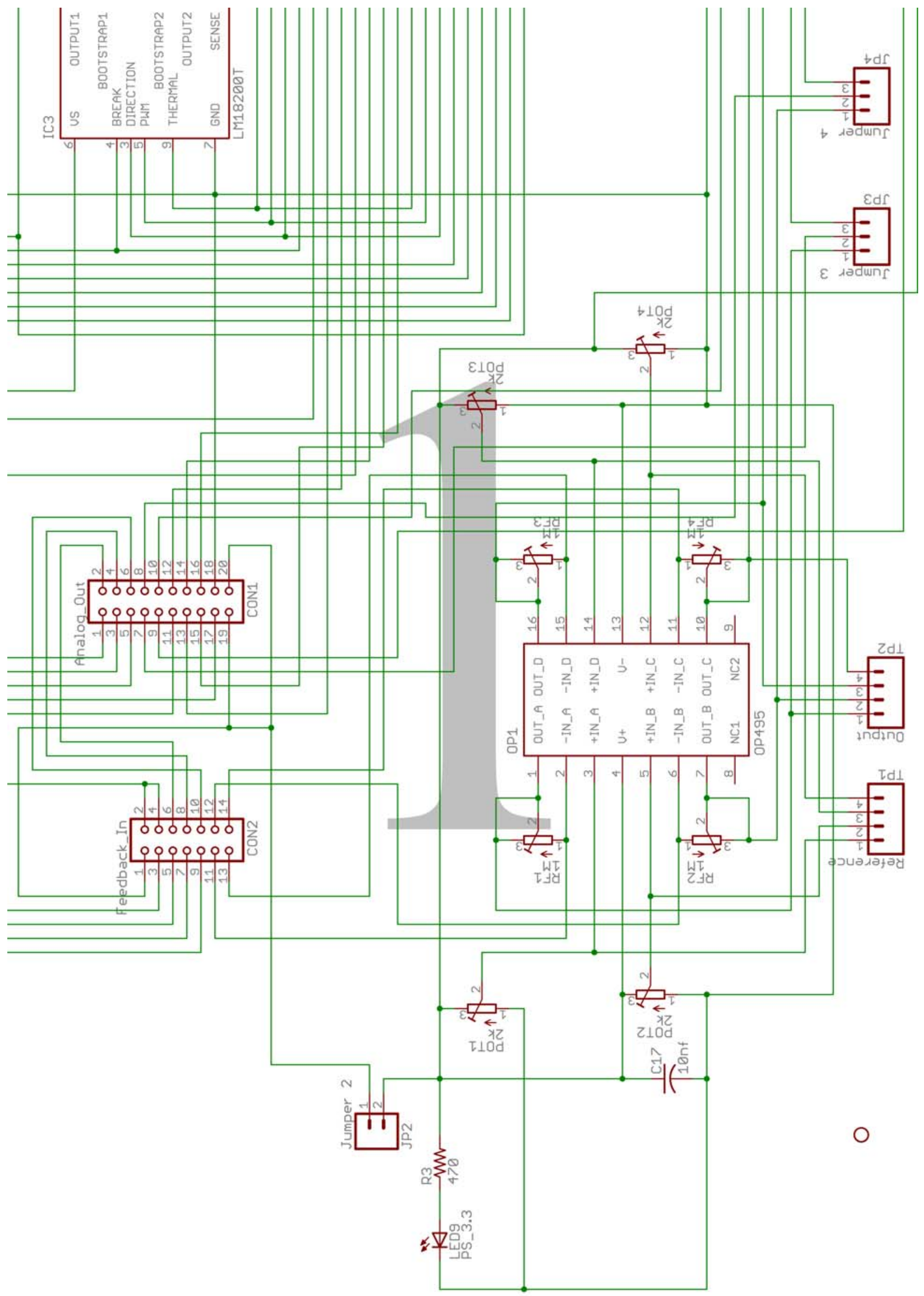
**M456**

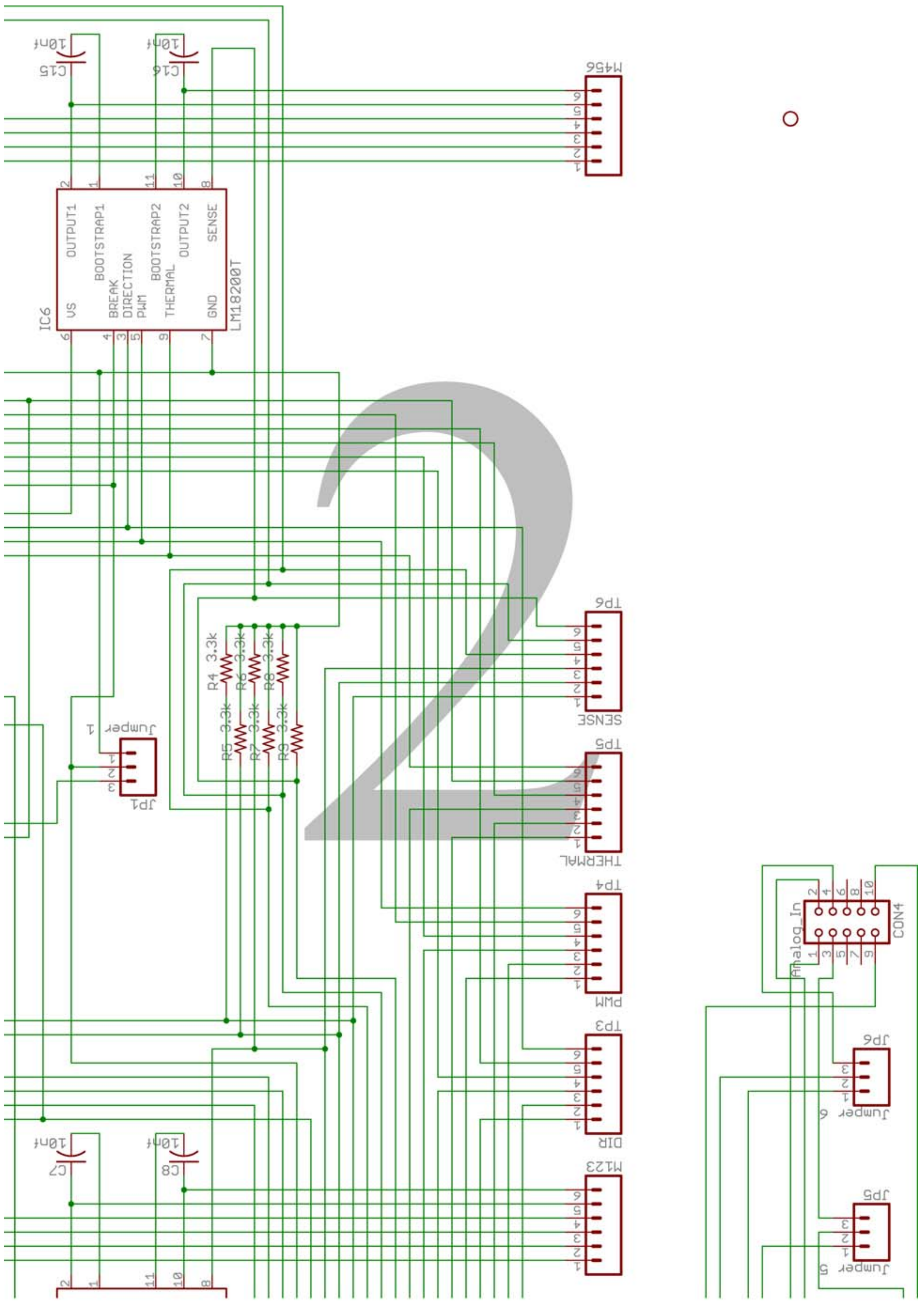
| | | | | | |
|---|---|---|---|---|---|
| Motor 4+ | Motor 4- | Motor 5+ | Motor 5- | Motor 6+ | Motor 6- |
| 1 | 2 | 3 | 4 | 5 | 6 |

Table B.1: Motor Driver Board Pinout

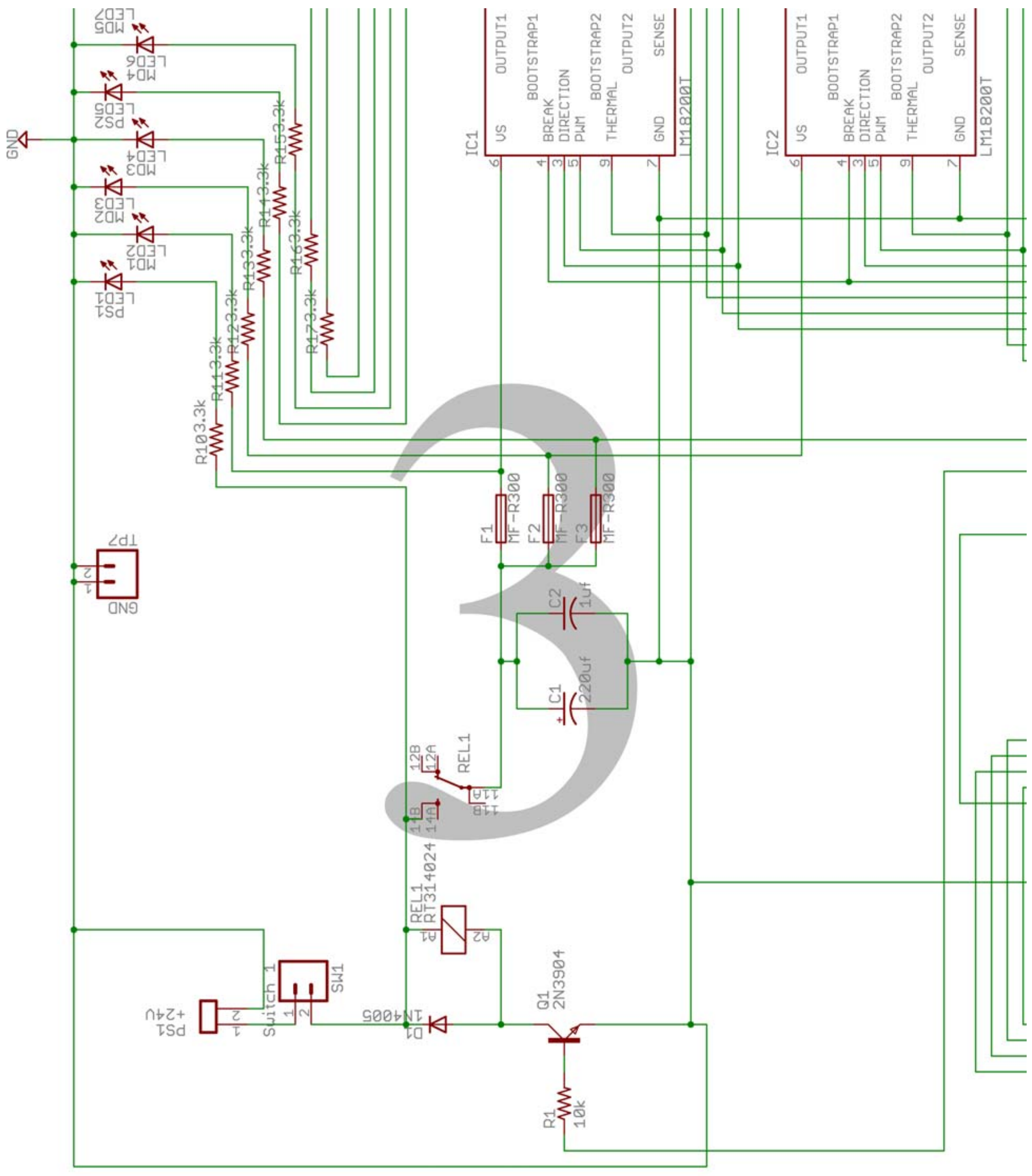


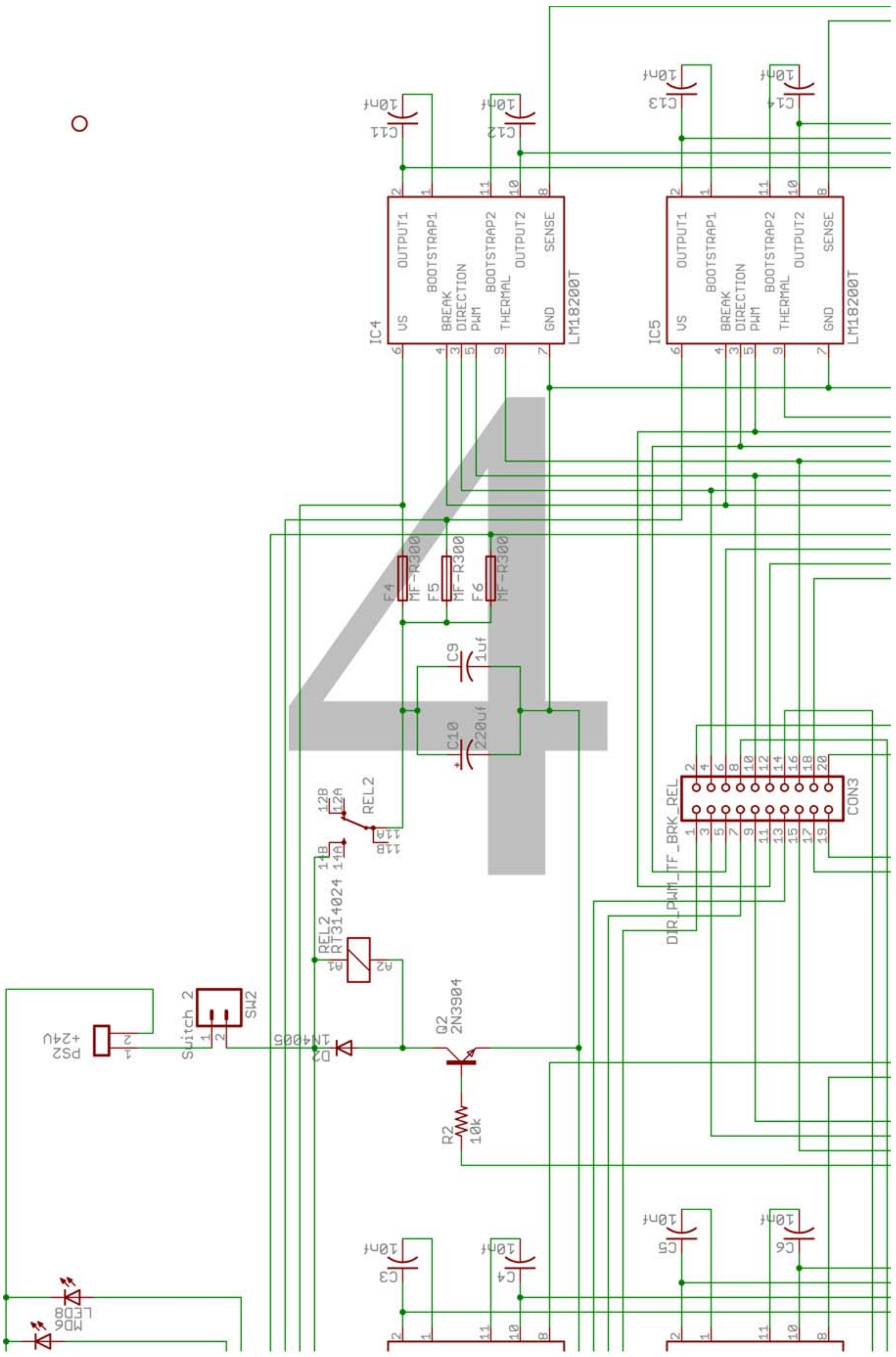
LM18200T
OP495
Analog_Out
CON1
Feedback_In
CON2
DIR_PWM_TF_BRK_REL
CON3
Analog_In
CON4
M123
M456
SENSE
THERMAL
PWM
DIR
Reference
Output
GND
+24V
2N3904
1N4005
RT314024
MF-R300


IC3
LM18200T
VS
OUTPUT1
BOOTSTRAP1
BREAK
DIRECTION
PWM
BOOTSTRAP2
THERMAL
OUTPUT2
GND
SENSE
JP4
Jumper 4
JP3
Jumper 3
POT4
2K
POT3
2K
RF3
1M
RF4
1M
RF1
1M
RF2
1M
Analog_Out
CON1
Feedback_In
CON2
OP1
OP495
OUT_A
-IN_A
+IN_A
V+
+IN_B
-IN_B
OUT_B
NC1
OUT_D
-IN_D
+IN_D
V-
+IN_C
-IN_C
OUT_C
NC2
TP2
Output
TP1
Reference
POT1
2K
POT2
2K
C17
10nf
Jumper 2
JP2
R3
470
LED9
PS_3.3

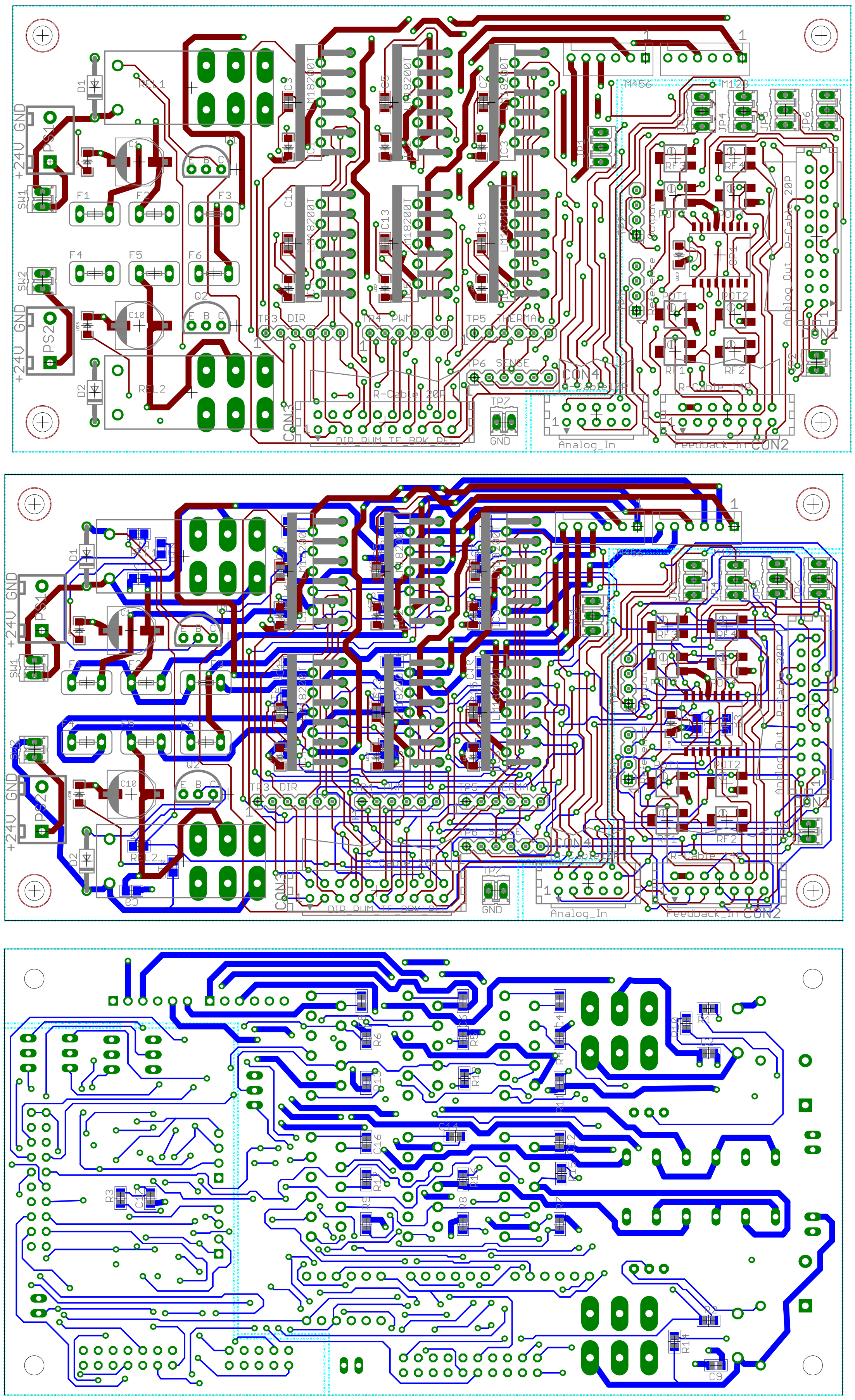


IC6
LM18200T
OUTPUT1
BOOTSTRAP1
BOOTSTRAP2
OUTPUT2
SENSE
VS
BREAK
DIRECTION
PWM
THERMAL
GND
C15
C16
C7
C8
10nf
M456
TP6
SENSE
TP5
THERMAL
TP4
PWM
TP3
DIR
M123
R4 3.3k
R5 3.3k
R6 3.3k
R7 3.3k
R8 3.3k
R9 3.3k
JP1
Jumper 1
CON4
Analog_In
JP6
Jumper 6
JP5
Jumper 5


GND
LED1 PS1
LED2 MD1
LED3 MD2
LED4 MD3
LED5 PS2
LED6 MD4
LED7 MD5
R10 3.3k
R11 3.3k
R12 3.3k
R13 3.3k
R14 3.3k
R15 3.3k
R16 3.3k
R17 3.3k
IC1
IC2
VS
BREAK
DIRECTION
PWM
THERMAL
GND
OUTPUT1
BOOTSTRAP1
BOOTSTRAP2
OUTPUT2
SENSE
LM18200T
F1 MF-R300
F2 MF-R300
F3 MF-R300
TP7
GND
C2 1uf
C1 220uf
REL1
12B
12A
11B
11A
14B
14A
REL1 RT314024
A1
A2
PS1 +24V
Switch 1
SW1
D1 1N4005
Q1 2N3904
R1 10k



IC4
LM18200T
VS
BREAK
DIRECTION
PWM
THERMAL
GND
OUTPUT1
BOOTSTRAP1
BOOTSTRAP2
OUTPUT2
SENSE
IC5
LM18200T
C11
C12
C13
C14
10nf
F4
F5
F6
MF-R300
C9
1uf
C10
220uf
REL2
12B
12A
11A
11B
14B
14A
RT314024
A1
A2
CON3
DIR_PWM_TF_BRK_REL
Q2
2N3904
R2
10k
D2
1N4005
SW2
Switch 2
PS2
+24V
C3
C4
C5
C6
LED8
MD6
4



The PCB layouts are as followed: Top Layer; Both Layers; Bottom Layer.