INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

MECHATRONICS AND INFORMATION TECHNOLOGIES
GRADUATE PROGRAM

A COMPARISON OF CONTROL SCHEMES FOR AN ARTICULATED 2 DEGREE-OF-FREEDOM ROBOT MANIPULATOR OPTIMIZED VIA GENETIC ALGORITHMS

THESIS

PRESENTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
IN INTELLIGENT SYSTEMS

JONATHAN ARRIAGA GONZALEZ

MONTERREY, N. L., MAY 2010

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY

CAMPUS MONTERREY

MECHATRONICS AND INFORMATION TECHNOLOGIES
GRADUATE PROGRAM

A COMPARISON OF CONTROL SCHEMES FOR AN ARTICULATED 2 DEGREE-OF-FREEDOM ROBOT MANIPULATOR OPTIMIZED VIA GENETIC ALGORITHMS

THESIS

PRESENTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE
IN INTELLIGENT SYSTEMS

BY

JONATHAN ARRIAGA GONZÁLEZ

MONTERREY, N.L. MAY 2010

A comparison of control schemes for an articulated 2 degree-of-freedom Robot Manipulator optimized via Genetic Algorithms

by

Jonathan Arriaga González

Thesis

Presented to the Graduate Program in

Mechatronics and Information Technologies

in partial fulfillment of the requirements for the degree of

Master of Science

in

Intelligent Systems

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

May 2010

To my parents, whose faith on me I never doubted.

Acknowledgments

This research work was possible with the help of many people I would like to thank.

To my parents Lic. Rodrigo Arriaga and Lic. Imla González, and brothers David Arriaga and Azael Arriaga, who taught me to dream and realize these dreams, to be constant, to be honest and to give my best.

To Dr. Rogelio Soto for being my advisor, for giving me the chance to attend to the conferences IEEE-SMC 2009 and MICAI 2009, for bringing me his guidance, suggestions, encouragement, experience, friendship and trust on me all the time in the develop of this research work. To Dr. José Luis Meza and Dr. Ernesto Rodríguez for bringing me their valuable experience, support and knowledge. To Dr. Manuel Valenzuela, who greatly inspired me to realize this research with his courses, experience, comments, ideas and work.

To Dr. Jose Luis Gordillo for accepting me in the eRobots research group and his encouragement and friendship all the time. To Dr. Hugo Terashima for his help to get the scholarship from ITESM and financial aid from CONACyT. To Dr. Ramón Brena, Dr. Leonardo Garrido, Dr. Santiago Conant, Dr. Arturo Galván and Dr. Aldo Díaz who also inspired me in the development of this work through their courses and experience.

To all my classmates and friends.

To ITESM Campus Monterrey and CONACyT for giving me the chance to receive, create and share knowledge and experience.

JONATHAN ARRIAGA GONZÁLEZ

Instituto Tecnológico y de Estudios Superiores de Monterrey
May 2010

A comparison of control schemes for an articulated 2 degree-of-freedom Robot Manipulator optimized via Genetic Algorithms

Jonathan Arriaga González
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2010

Thesis Advisor: Dr. Rogelio Soto

Robot manipulators play an important role in actual industrial processes. The trajectory following of robot manipulators is a non-linear problem that still requires much research [34]. This research work focuses on the control of the dynamics of an articulated robot manipulator.

A 2 degree-of-freedom (DOF) articulated robot manipulator is simulated, each of the two links of the robot having its respective controller. Two different kind of control objectives for the robot's links are considered, position control and velocity control. Four control schemes for the robot's dynamics were selected. For position, a PID Controller [24] and a Fuzzy Self-tuning (FST) PID Controller [14] are considered. On the other hand, for velocity control the FST PD+ Controller [35] and the Fuzzy Sliding Mode (FSM) Controller [12, 16, 28] were chosen. Controller's performance and robustness in relevant tasks are evaluated and compared in order to determine which control scheme fits best for each task.

Empirical adjustment of most controller's parameters always depend on the time and tests invested in tuning the controller, it is time consuming and subject to human error. As a fair comparison is intended, controller's parameters are optimized via Genetic Algorithms [22]. With this method, the tuning of parameters is not subtle to human error and the comparison can avoid possible erroneous conclusions.

Optimization of parameters of all controllers was carried out successfully. Parameters set by the GA are interpreted and show several details about the structure of the considered controllers. Performance comparison of controllers is discussed and conclusions about the complexity of controllers and its equivalence when performing some tasks is presented.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robot manipulators are commonly used in industrial process to develop tasks like painting, pick and place, arc welding and others. Actually, industrial robots are capable to develop a great variety of activities with high precision and repeatability. However, research about robotics is relevant given that there are still many applications that commercial robots are unable to perform correctly, like handling of fragile objects and imitation of human moves. The trajectory following of a robot manipulator is a non-linear control problem that still requires much research and is a very important field for production process. It is also an interesting control problem from the academic point of view.

This thesis focuses on the control of the dynamics of robot manipulators. The desired control objective is to make the manipulator asymptotically follow a given trajectory within a finite interval of time. This objective is not possible in general since asymptotic stability can be achieved only as time approaches to infinity [34] from the theoretical point of view.

Actual industrial robots have gears in its joints which transmit power from the actuators. When gear ratio is big enough, the robot's dynamics simplifies greatly to the point that the non-linearities can be neglected [34]. The robot's dynamics is given thus only by the dynamics of the actuators. The counterpart of using gears is that the mechanical complexity of the system increases and the mass does as well.

Thus, if no gears are used, the primary difficulty for the control of the robot's dynamics is that the robotic system is highly non-linear and presents inherently unknown dynamics [31]. A non-linearity in a system is a non-proportional relation between cause and effect. Physical systems are inherently non-linear, thus all control systems are non-linear from a certain point of view. Non-linearities can be classified as *inherent* (*natural*) and *intentional* (*artificial*) [5]. Inherent non-linearities are those which naturally come with the system's hardware and motion. Centripetal forces in rotational motion and Coulomb friction between contacting surfaces are examples of non-linearities.

The unknown dynamics are due to imperfect mathematical modelling of friction, dead time, loads, temperature, maintenance and so on. Therefore, is vital that the controller has the capability to adapt to different robot working conditions. For exam-

ple, when a robot takes an object, due to the object's mass, the apparent mass of the robot's last link is increased. The other links are affected by this parameter change because the robot is a linked mechanism. Adaptable control laws are heavily studied with the aim to minimize the perturbations with time.

If a controller that handles well all this issues is developed along with the needed actuators, it could be possible to design lighter and faster robots, with less or without gears. However, the mechanical design of robots is beyond the scope of this document, which focuses mainly on the study of the control of robot's dynamics.

With the speed of actual processors many complex control techniques are possible to be implemented. Actual actuators have also many capabilities which weren't available when robots from the last two decades were designed, thus the design of even actual robots has limitations not valid for today's technology. In the research presented in this document, performance of two controllers of the same kind for the same task is compared. A simulation model of a 2 degree-of-freedom (DOF) articulated robot is used to compare the different control schemes and decide which performs better for different kind of tasks.

1.1 Problem Formulation

When controlling the dynamics of articulated robot manipulators, the principal problem is the inherent non-linearities in the system. The logical choice when a non-linear system will be controlled is to use a non-linear controller which absorbs the non-linearities; while if the system is linear, then a linear controller would be the best choice. If the operating point of a given control system is small and the non-linearities are smooth, the control system may be approximated by a linear system with good accuracy.

Many of the most common controllers are linear because its application does not require improved performance or the operating point can be approximated as a linear system. An articulated robot manipulator is a non-linear system and its operating point is the whole range of movement, therefore its dynamics cannot be approximated to a linear system [34]. From this point in the document, the term *robot manipulator* will refer to *articulated robot manipulator.*

As a robot manipulator is subject to an environment with high variability, interacts with objects of different sizes and weights and its parameters can change with time as the robot ages. The dynamic model of a robot manipulator can be derived mathematically or by analyzing system data, but uncertainties of the actual real system with respect to the system model will always exist.

When controllers are to be built without having an accurate mathematical model of the system to be controlled, two problems arise: first, how to *establish the structure*

of the controller, and second, how to *set numerical values* of the controller parameters [39]. The first problem can be solved based in the approximated system model and the task type. For the second problem, depending on the controller there may be a method to adjust its parameters.

Controlling the dynamics of robot manipulators means to make the robot's links asymptotically approach to a reference. It is an interesting problem and represents some challenges in the design of the controller's structure and optimization due to its non-linearities and model uncertainties.

For robot manipulators, position tasks means regulating the system in a static position reference. Velocity tasks, in the other hand, means tracking of position trajectories. The tracking problem can be defined as follows [5]: Given a non-linear dynamics system and a desired output trajectory, find a control law for the input such that starting from any initial state within a bounded region, the tracking errors go to zero while the whole state $\boldsymbol{x}$ remains bounded.

Velocity controllers are defined as follows [10]: Assume that joint position $\boldsymbol{q}$ and joint velocity $\dot{\boldsymbol{q}}$ are available for measurement. Let the desired joint position $\boldsymbol{q}_d$ be a twice differentiable vector function. The controller will determine the actuator torques $\boldsymbol{\tau}$ in such a way that the following control aim be achieved

$$\lim_{t\to\infty} \boldsymbol{q}(t) = \boldsymbol{q}_d(t) \tag{1.1}$$

The control system is said to be globally asymptotically stable if the control aim is guaranteed irrespective of the robot initial configuration $\boldsymbol{q}(0)$ and $\dot{\boldsymbol{q}}(0)$.

For this research, two different controllers will be optimized for position tasks:

PID Controller The conventional PID controller is widely used in industrial process control because of its simplicity and its rather satisfactory performance. This controller was introduced in 1922 as a three-term controller [24]. It grow popular and was heavily analyzed to date. It is commonly used in applications in which the process parameters are well known and do not vary substantially [14].

Fuzzy Self-Tuning PID Controller This controller dates from 1985 [14]. When the process parameters are not known accurately or if they vary under different operating conditions, a conventional PID controller may not behave satisfactorily. The system performance can be improved by applying more elaborate nonfuzzy control schemes or by using a fuzzy controller. Instead of replacing the conventional PID controller by a fuzzy controller, it is desirable to maintain the PID controller and to add a higher level of control to it to improve the performance of the system. In this control scheme, the knowledge, experience and intuition of a designer is formulated linguistically and, by applying fuzzy subset theory, transformed into an extended fuzzy algorithm for tuning the parameters of the PID

controller, which is known as Fuzzy Self-tuning (FST) PID controller. Stability about this controller has been studied in [36, 21]. In [22] Membership Functions (MF) for this controller have been optimized via Genetic Algorithms and compared against an empirical tuning.

Similarly, for velocity tasks two controllers will be considered:

Fuzzy Self-Tuning PD+ Controller This controller was introduced first in [17] in 1998 as Fuzzy Self-tuning Computed-torque Control based in the Computed-torque Method [29]. It was later handled as Fuzzy PD+ in [35], which is also based in the controller introduced in [10] known as PD+. The structure of PD+ control consists of a linear PD feedback plus a specific compensation of the robot dynamics. In the Fuzzy PD+ control scheme, the gains of the PD+ are allowed to vary according to a fuzzy logic system which depends on the robot state. This is an important ingredient to deal with practical specifications such as keeping asymptotically the tracking error within prescribed bounds without saturating the actuators. Stability for this controller is analyzed in [17, 35, 33].

Fuzzy Sliding Mode Controller The Fuzzy-Sliding Mode (FSM) Controller has the characteristics of both a fuzzy logic controller and a variable structure controller (VSC) or also known as sliding mode controller. It was introduced in 1992 in [12, 16] and was further studied in [28]. The FSMC has a fuzzy-sliding motion similar to the conventional sliding motion of the VSC. In the FSMC the switching hyper plane is a fuzzy set, while in the sliding mode controller is a crisp set. Membership functions and fuzzy rules can easily be determined to prespecify the behavior of control system. In the conventional sliding mode controller, when the system enters into sliding mode, the state trajectories of the controlled system will be kept on a specified switching hyperplane. Due to the sampling action of digital implementation, delay of switch device and its operation at discrete instants, a sliding mode controller may have chattering. This phenomenon is undesirable in most control applications because high frequency components in the system not considered in the model could be excited and make the system unstable. One of the principal motivations to develop the FSMC was the attenuation of chattering. A simulation of a 6 DOF robot manipulator implementing this controller is reported in [3].

The more complex is the system and the controller, the more complex will be the adjustment procedure of the controller's parameters settings. Adjusting the controller's parameters can be a tedious task and many times the adjustment will not be the optimal even if is carried out by experienced operators. Empirical adjustment is an iterative process which can be costly and time consuming. An example of this case are the

PID controllers, which are very popular and effective for industrial applications but its performance depends heavily on the operating parameters of the system. Once these parameters change, a significant amount of effort is required to manually tune the PID controllers [22].

An approach based in Genetic Algorithms (GA) is considered for this task. GAs have been applied to control problems in approximating the system's model and for the design and optimization of the controller for the system. In [39] a Genetic Algorithm is implemented as an estimator for discrete time and continuous systems. Genetic Algorithms are used in [11] to estimate both continuous and discrete time systems and for identifying poles and zero or physical parameters of a system, in order to design an adaptive controller in combination with a pole placement scheme.

A comparison of Fuzzy Control against PID Control for a Position and Force Control of a Robot hand is presented in [8] and resulted that the Fuzzy Controller outperformed the PID Controller in tests of pure position control and pure force control. In that paper, both controllers were empirically adjusted, which lacks of a formal proof to determine if one controller has a better performance than another one. A PID controller has been successfully optimized with a GA comparing this optimization with that of a Simulated Annealing (SA) method and empirically adjusted settings in [13], the GA outperformed the optimization by SA and the empirical method. The last comparison is much better than the first one because the tuning of parameters in this case is not subtle to human error, which could have a negative effect in the controller's performance.

Therefore, GAs will be used to optimize the controller's parameters to work in relevant operating points of the robot's links. After optimization, controller's performance criterions will be measured and compared to determine which fits best for each kind of task. It is always important to make fair comparisons, in this case conditions of the system and design of controllers must be the same. Conditions of the system can easily be simulated to be the same, the difficulty relies in performing a fair tuning of controllers. Most controllers can have excellent performance if well tuned, while if tuning is poor then performance is also.

The common technique to tune controllers, even in industrial processes, is by empirical adjustment based on the operator's knowledge of the system. Empirical adjustment of parameters always depend on the time and tests invested in tuning the controller, it is time consuming and subject to human error. The use of GAs to optimize the controllers intends to overcome this problem and stay apart from possible erroneous conclusions.

1.2 Hypothesis

The principal hypothesis is that controllers for the dynamics or robots manipulators can be successfully optimized by Genetic Algorithms. Different controller schemes can then be compared fairly when optimized by this method. The comparison and the parameters of the controllers optimized provide relevant knowledge about the control of the system.

1.3 Objectives

The first objective of this research work is to optimize the proposed controllers to develop specific tasks. Once the controllers have been optimized, a comparison of performance criterions will be made. Optimization before performing a comparison is vital because is the most important part of the design. A controller is not useful if it is not well tuned, but is very useful if all parameters are set so the system develops the specified task.

A particular objective, regarding to position controllers is to determine if a non-linear controller handles the system much better than a linear controller. This will tell if the extra controller-complexity makes a real difference or is enough with a linear well-tuned controller .

1.4 Methodology

Given a physical system to be controlled, a common procedure for control design is as follows, with possibly a few iterations [19]:

1. Specify the desired behavior, and select actuators and sensors.
2. Model the physical plant by a set of differential equations.
3. Design a control law for the system.
4. Analyze and simulate the resulting control system.
5. Implement the control system in hardware.

For this thesis, only steps 1 to 4 are followed for each controller, implementation of the control system is left as a future work. Steps 3 and 4 are done for each of the 4 controllers considered to analyze.

An additional step is implemented also for each controller, the optimization of the control system's parameters, and is carried out via Genetic Algorithms. Finally,

having each controller designed and optimized, a comparison of performance criterions will determine which controller is better for each particular task.

Steps for the optimization and performance evaluation process are listed below:

1. **Optimization.** In this step the parameters for the controller of each link are optimized with the GA.

2. **Performance tests.** Once optimized, both controllers for each task are tested in normal operation conditions.

3. **Robustness tests.** The robustness of each controller is evaluated by modifying the operation conditions for which the controller was optimized.

1.5 Document Organization

The present document is organized as follows. In chapter 2 actual Artificial Intelligence (AI) methods are described, which are Fuzzy Logic, an important component of controllers and Genetic Algorithms, the optimization method chosen. Chapter 3 presents the Robot's mechanics, a dynamic model to implement a simulation is developed. The proposed controllers for comparison are presented in chapter 4. A methodology for the optimization and comparison process is introduced in chapter 5, comparison after optimization is also reported in this chapter. Finally, chapter 6 presents the conclusions of the research.

Chapter 2

Artificial Intelligence Methods

In this section two popular Artificial Intelligence methods are described. One is Fuzzy Logic, a method to represent knowledge and has been used broadly in control, decision-making techniques, prediction, etc. The other is Genetic algorithms (GA), which is an optimization method based in the mechanics of natural evolution of a population.

2.1 Fuzzy Logic

In the early 1960's, Lofti A. Zadeh from the University of California at Berkley introduced the idea and concept of grade of membership for the elements of a set. In 1965, he published his seminar paper on fuzzy sets [40] which lead the emergence of the fuzzy logic technology. The first fuzzy logic controller was published by [20], used to control a simple dynamic plant.

Fuzzy logic is an extensively studied field, many concepts and many applications have been developed. In this section a brief explanation of the most important concepts for understanding fuzzy logic most of these reported in [2].

2.1.1 Fuzzy Sets and Fuzzy Subsets

A fuzzy set is a generalization to classical set where the elements have degrees of membership. A fuzzy set is characterized by having a set of elements and a *membership function* (MF) that maps these elements of a universe of discourse to their corresponding membership values. The MF of a fuzzy set A is denoted by μ_A. The logic operators of intersection and union have multiple choices for the fuzzy conjunction (AND) and the fuzzy disjunction (OR) operators.

An **ordinary subset** A of a set X can be identified with a function $X \to \{0,1\}$ from X to the 2-element set $\{0,1\}$ as follows

$$A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{2.1}$$

This function is called the characteristic function.

Different from ordinary subsets, **fuzzy subsets** of X can have varying degrees of membership from 0 to 1. Using fuzzy subsets, the value of $A(x)$ is thought as the degree of membership of x in A. This function is called **membership function** of A and can be expressed as $\mu_A : X \rightarrow [0, 1]$. An special case is when the MF only takes values 0 or 1, A is known in this case a **crisp subset** of X.

Commonly the set X is called the **universe of discourse**. Since data is generally numerical, the universe of discourse is most often an interval of real numbers. The shape of a MF depends on the notion the set is intended to describe and on the particular application involved. Common MFs for control applications are triangular, trapezoidal, Gaussian and sigmoidal Z- and S-functions.

Triangular MF are used in applications very often, see figure 2.1. Equation 2.2 characterizes the degree of membership of x in this kind of MF. Graphical representations and operations with these fuzzy sets are simple. A triangular MF A with end points $(a, 0)$ and $(b, 0)$, and high point (c, α) is defined by:

$$A(x) = \begin{cases} \alpha\left(\frac{x-a}{c-a}\right) & \text{if } a \leq x < c \\ \alpha\left(\frac{x-b}{c-b}\right) & \text{if } c \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

For the case of trapezoidal MF, end points $(a, 0)$ and $(b, 0)$, and high points (c, α) and (d, α) define a trapezoidal MF A as follows:

$$A(x) = \begin{cases} \alpha\left(\frac{x-a}{c-a}\right) & \text{if } a \leq x < c \\ \alpha & \text{if } c \leq x < d \\ \alpha\left(\frac{x-b}{d-b}\right) & \text{if } d \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

MFs are useful for processing numeric input data. A wider explanation of fuzzy subsets can be found in [25].

2.1.2 Linguistic Variables

When a variable is described with numbers as values, it can be formulated with math theory. But when a variable takes words as values, there isn't a formal structure to formulate the problem with classic math theory [15]. The linguistic variables concept was introduced in order to provide a formal structure to deal with words as values. Linguistic terms are useful for communicating ideas and knowledge with human beings, is the most important element in the representation of human knowledge.

Fuzzy subsets are associated with a linguistically meaningful term; for example "high" error and "low" error. The word error is the linguistic variable and adjectives

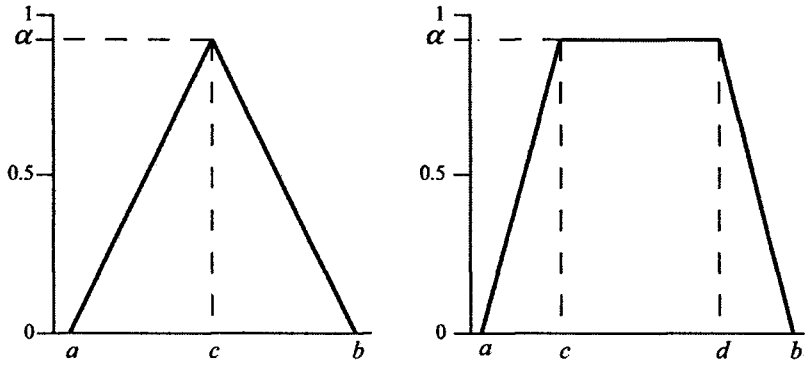


Figure 2.1: Triangular and Trapezoidal MFs.

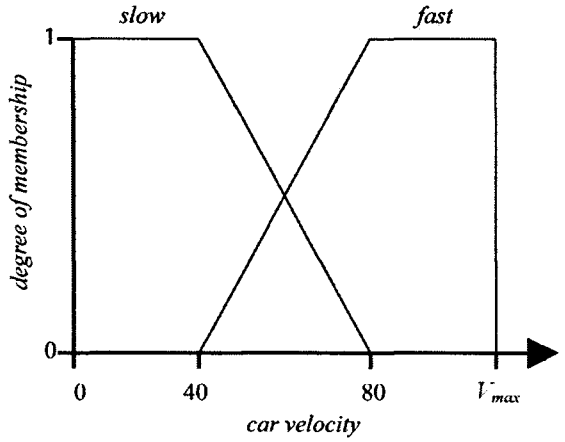


Figure 2.2: MFs for *speed.*

"high" and "low" are linguistic values. A linguistic variable is the equivalent of a symbolic variable in AI and a numeric variable in science and engineering. In general, the values of a linguistic variable can be linguistic expressions of terms and modifiers such as "very", "more" and connectives such as "and", "or".

For example, the velocity of a car is x, which can take values in the interval $[0, V_{max}]$, where V_{max} is the car maximum velocity. Two MFs are defined "*slow*" and "*fast*" within $[0, V_{max}]$ as shown in 2.2. If x is considered as a linguistic variable, it can take values as "*slow*" and "*fast*". Thus, it can be said that "x *is slow*" or "x *is fast*".

Formally, a linguistic variable is characterized by (X, T, U, M), where:

X is the name of the linguistic variable. From figure 2.2, X is the car speed.

T is the set of linguistic values X can take. From figure 2.2 is inferred that $T = slow, fast$.

U is the physical domain in which X can take quantitative values. From figure 2.2, $U = [0, V_{max}]$

M is a semantic rule relating each linguistic value in T with a fuzzy subset in U. From the last example, M relates $T = slow,\ fast$ with the MFs in figure 2.2.

2.1.3 Fuzzifiers and Defuzzifiers

The *fuzzification* process is defined as a mapping from a real-valued point $\boldsymbol{x}^* \in U \subset R^n$ to a fuzzy set A' in U. While designing a fuzzifier, one should consider that the input is at the crisp point $\boldsymbol{x}^*$, the fuzzy set A' should have a large membership value at $\boldsymbol{x}^*$. If the input to the fuzzy system is corrupted by noise, it is desirable that the fuzzifier should suppress the noise. Maybe one of the most important points when designing a fuzzifier is the computations involved in the inference engine. If the fuzzifier is ruled by a complex formula then performance will be greatly affected, furthermore the knowledge with a complex function will be harder to understand by a human being than when using a simpler function.

In [15], three fuzzifiers are proposed:

Singleton fuzzifier The *singleton fuzzifier* maps a real-valued point $\boldsymbol{x}^* \in U$ into a fuzzy singleton A' in U, which has a membership value 1 at $\boldsymbol{x}^*$ and 0 at other points int U; it is expressed as

$$\mu_{A'}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} = \boldsymbol{x}^* \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Triangular fuzzifier The *Triangular fuzzifier* maps $\boldsymbol{x}^* \in U$ into a fuzzy set A' in U which has the following triangular MF:

$$\mu_{A'}(\boldsymbol{x}) = \begin{cases} (1 - \frac{|x_1 - x_1^*|}{b_1}) \star \cdots \star (1 - \frac{|x_n - x_n^*|}{b_n}) & \text{if } |x_i - x_i^*| \leq b_i,\ i = 1, 2, \cdots, n \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

where b_i are positive parameters and $\star$ describes the t-norm (described in [15, 25]), which is usually an algebraic product or *min* product. In order to use trapezoidal MFs (which are an extension of triangular MFs), little changes should be implemented in the last equation.

Gaussian fuzzifier The *Gaussian fuzzifier* maps $\boldsymbol{x}^* \in U$ into a fuzzy set A' in U which has the following Gaussian MF:

$$\mu_{A'}(\boldsymbol{x}) = e^{-(\frac{x_1 - x_1^*}{a_1})^2} \star \cdots \star e^{-(\frac{x_n - x_n^*}{a_n})^2} \tag{2.6}$$

This fuzzifier method is broadly used in many applications, however the application described in chapter 5 only uses trapezoidal and singleton fuzzifiers.

The *defuzzification* process is a mapping from fuzzy set B' in $V \subset R$ to a crisp point $y^* \in V$. In other words, the defuzzifier specifies a point in V that best represents the fuzzy set B', which is similar to the mean value or a random variable. The fuzzy set B' is defined in a special way and there are a number of methods to determine its representing point. The design of the fuzzy subset B' along with the defuzzification process should consider the same points as when designing the fuzzification process.

Also in [15] the center of gravity defuzzifier, center average defuzzifier, maximum defuzzifier are defined. For the application in this document, only the center average defuzzifier process is reported.

Center average defuzzifier As the fuzzy set B' is the union or intersection of M fuzzy sets, an approximation of the centers of the M fuzzy sets with the weights equal the heights of the corresponding fuzzy sets. Let y^{-l} be the center of the $l'th$ fuzzy set and w_l be its height, the center average defuzzifier determines y^* as:

$$y^* = \frac{\sum_{l=1}^{M} y^{-l} w_l}{\sum_{l=1}^{M} w_l} \tag{2.7}$$

This defuzzifier is the most commonly used defuzzifier in fuzzy systems and fuzzy control. It is computationally simple and intuitive plausible, small changes in y^{-l} and w_l result in small changes in y^*.

2.1.4 Fuzzy IF-THEN rules

Fuzzy systems are knowledge-based systems or rule-based systems. The core of a fuzzy system is a knowledge base composed of fuzzy IF-THEN rules. A fuzzy IF-THEN rule is a relation that transforms conditions about linguistic variables to conclusions.

The following is an example of a fuzzy IF-THEN rule [15]:

IF velocity of car is high
THEN apply less force to the accelerator pedal

in which words "*high*" and "*less*" are characterized by MFs. Fuzzy rules are commonly generated from a human expert in the system or from knowledge obtained while operating the system.

The main feature of this kind of knowledge representation is its partial matching capability that enables an inference even when the rule conditions are partially satisfied. To infer a conclusion in a fuzzy rule, the conditions based on the match degree of the input data and the consequent are combined. The higher is the matching degree, the closer is the inferred conclusion to the consequence of the rule.

2.1.5 Fuzzy Inference Engine

A typical rule base is of the form

$$R_j: \text{ If } \boldsymbol{x} \text{ is } A_j \text{ then } u \text{ is } B_j,\ j = 1, 2, \cdots, r \tag{2.8}$$

where $\boldsymbol{x} = (x_1, \cdots, x_n) \in X$ and $u \in U$, and $A_j(\boldsymbol{x}) = \min_{i=1\cdots n}\{A_{ji}(x_i)\}$ for

$$\begin{aligned} A_j &: A_{1j} \times A_{2j} \times \cdots \times A_{nj} : X = X_1 \times X_2 \times \cdots \times X_n \to [0,1] \\ B_j &: U \to [0,1] \end{aligned} \tag{2.9}$$

A common interpretation for R_j is a fuzzy relation $X \times U$. This means that when the input is x, the degree to which a value $u \in U$ is consistent with the meaning of R_j is

$$C_j(x,u) = T(A_j(x), B_j(u)) \tag{2.10}$$

where T is some t-norm [25]. Thus, C_j is a fuzzy subset of $X \times U$. For a given input x, C_j induces a fuzzy subset of U

$$C_j^x(x) : u \to C_j(x,u), u \in U,\ j = 1, 2, \cdots, N \tag{2.11}$$

This implies that the output of each rule R_j is a fuzzy subset of U. The Mamdani method [20] uses the minimum of the t-norm:

$$C_j^x(u) = C_j(x,u) = A_j(x) \wedge B_j(u) \tag{2.12}$$

Having translated each rule R_j into C_j^x, the next task is to fuse all the rules together. The problem is that given N fuzzy subsets, $C_1^x, \cdots, C_N^x$ of U, it is needed to combine them to produce and overall output. What is wanted is a single fuzzy subset of U from the C_j^x, $j = 1, \cdots, N$. The Mamdani method to accomplish this task is for each $\boldsymbol{x} \in X = X_1 \times \cdots \times X_n$, this gives the fuzzy subset

$$C^x(u) = C(u|x) = \max_{j=1,\cdots,N}\left(\min_{i=1,\cdots,n}\{A_{ji}(x_i), B_j(u)\}\right) \tag{2.13}$$

of U. This last equation is known as **Mamdani synthesis**.

The overall output is, for each x, a fuzzy subset C^x of U. The meaning of C^x is that when the input x is given, each control action u is compatible with degree $C^x(u)$.

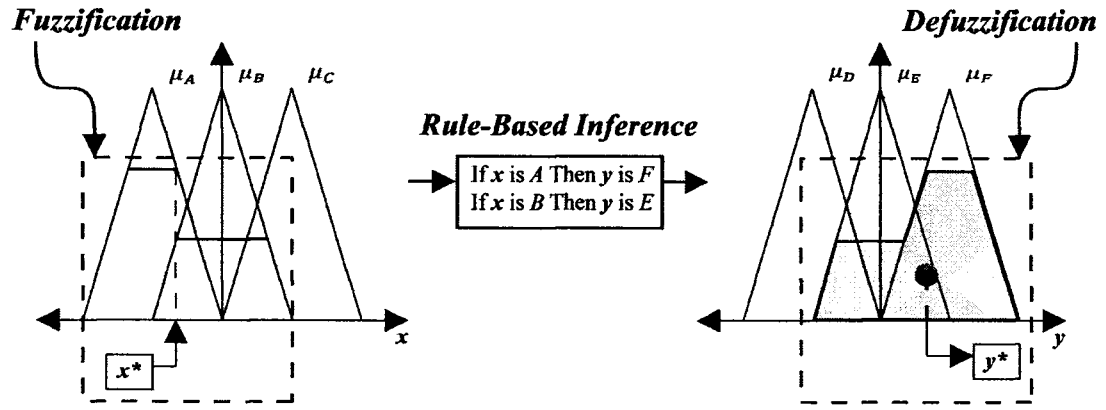


Figure 2.3: Example of fuzzy processing of variables.

In most control applications, a single numerical (crisp) output is needed $u^* = u^*(x)$ for the control law. A crisp output is obtained from C^x using any of the *defuzzification* methods described above.

2.1.6 Fuzzy Processing of Variables

The fuzzy processing of crisp variables can be summarized as follows:

Fuzzification Mapping from a real-valued point (crisp point) to an input fuzzy set in the universe of discourse.

Rule-Based Inference Transformation of input fuzzy sets into output fuzzy sets by means of a set of rules and an inference engine.

Defuzzification Mapping from output fuzzy sets to a real-valued point.

The process above summarized is illustrated in figure 2.3. This figure shows an example of a single-input single-output fuzzy system, however, the number of input and output variables a fuzzy system can process is not limited.

2.2 Genetic Algorithms

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics [6]. They combine survival of the fittest among string structures with a structured yet randomized information exchange to form a search algorithm. In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old, an occasional new part is tried for good

measure. Genetic algorithms are no simple random walk, they exploit the historical information to speculate on new search points with expected improved performance.

The principal differences of GAs from more normal optimization and search procedures are:

- GAs work with a coding of the parameters, not the parameters themselves.
- GAs perform a parallel search, exploring many points in the search space at the same time.
- GAs use only the result from the objective function to guide the search.
- GAs use probabilistic transition rules, not deterministic rules.

2.2.1 The Algorithm Description

The core of the GA is the codification of the solution to the problem, usually handled as a binary string of finite length called **chromosome**. Each solution is known as an **individual**, there is a number of individuals (solutions) that compose a **population**. Every cycle of the GA, individuals are evaluated with a **fitness function** and with some operators (that will be described next) a new population or **offspring** is created. The cycles in which a new population is created are called **generations**. So, the GA works with a population of solutions that explore the search space at the same time (within a generation), therefore is said that GAs perform a parallel search, regardless of the fact that only one solution is evaluated at a time.

To run a GA, an initial population is needed, it can be random (setting every bit of the chromosome of every individual randomly to 0 or 1) or defined, maybe as the last population of a previous run. Within a population, every individual is codified into a string representing its chromosome.

A codification with *real* numbers has also been developed and implemented in [23], the best representation is the one which is more *natural* to the problem. Computer languages in which individual bits cannot be accessed the *real* codification is encouraged. However, there is an option to this case, the virtual gene genetic algorithm (vgGA) [38]. In the vgGA, traditional crossover and mutation are implemented as arithmetic functions. This implementation allows the generalization to virtual chromosomes of alphabets of any cardinality. This codification is more efficient than using an integer number to represent a single bit, in terms of computer memory.

To give every individual an estimate of how good or bad it behaves, its chromosome is decodified into an instance of the solution to the problem. The problem is then simulated and its performance is evaluated with the fitness function defined, the result of the evaluation is assigned to each individual. Individuals are evaluated along the

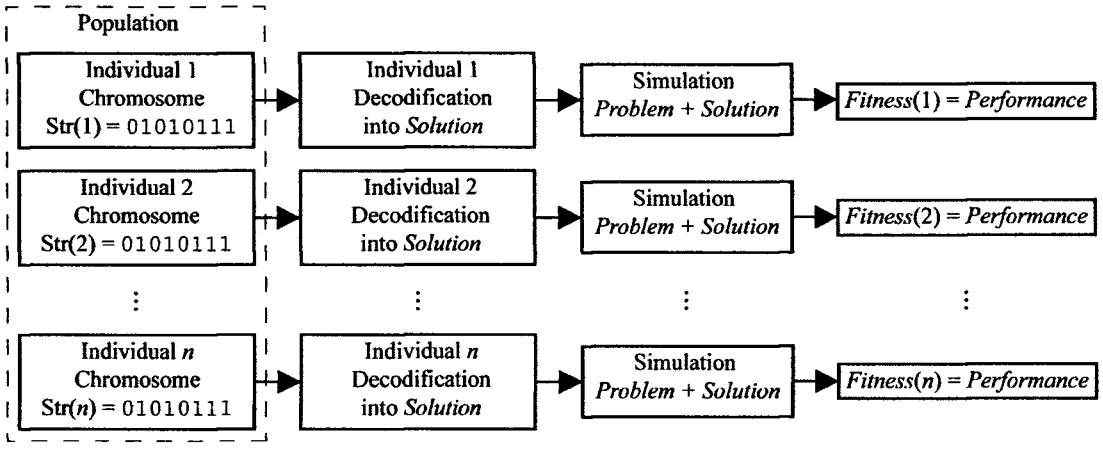


Figure 2.4: Fitness assignment to a population of n individuals.

generations with the same fitness function. This sub procedure is illustrated in figure 2.4.

Having received an evaluation with the fitness function, individuals are **reproduced** in a way that those with higher fitness receive more copies and those with lower fitness receive less copies. A popular method for selecting individuals to reproduce based in its fitness is roulette-wheel selection. In roulette-wheel selection, fitness for all individuals compose a wheel with every individual having a piece of the wheel representing its relative fitness with respect to the total population. A pointer points randomly to some place in the wheel and the chosen individual receive a copy in a mating pool. This is performed a number of times equal to the size of the population.

After the selection process, stronger individuals (with higher fitness) are more likely to have more copies in the mating pool than the weaker ones. Once the mating pool is created each copy of each individual mates with another one chosen randomly. A crossing point is selected within the string bits for each mate to perform **crossover** after that point, this is called single-point crossover. Multiple-point crossover has been analyzed in [37], results suggest that two-point crossover performs better than single-point crossover. With crossover, a new offspring is generated, two new individuals for the next generation are created from two individuals in the current generation. Crossover of two individuals with cross point after the second bit is illustrated below.

Before crossover		After crossover
00|00	$\rightarrow$	00|11
11|11		11|00

After crossover, each bit of each individual is **mutated** with a very low probability, usually 2%. The result is a new population with possibly better individuals than their parents in the last generation.

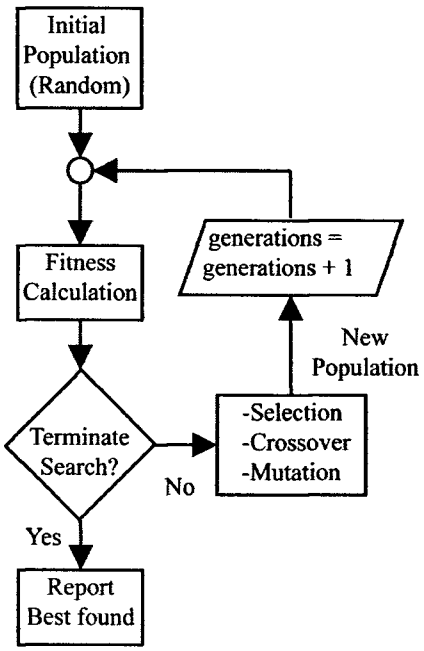


Figure 2.5: Flow chart diagram of steps the Genetic Algorithm performs.

This is a brief description of GAs, a pseudo code of the whole algorithm is rather complicated. Instead, a flow chart diagram of a simple GA is illustrated in figure 2.5. A pseudocode version has already been reported in [6].

2.2.2 Simple example of Genetic Algorithm

In this section a GA is followed step by step to illustrate it's mechanics. First, consider the function $f(x) = \sqrt{x}$, when $x \in [1\ 64]$ illustrated in fig. 2.6. The GA will try to find the value for x that maximizes the function $f(x)$.

If the shape of the function is not known or where the maximum value can be found, a wide range for the involved variables must be chosen in order to let the GA find the optimal values for that function. This is a weak point for the GA, it can only explore points within the given range of variables.

As the GA only works with binary numbers, x must be codified in a binary string, which for this example will be of length 4. The four digit string can represent values from 0 (0000) to 16 (1111), these digits must be decodified to transform them into valid values for x. The following expression is used for the decodification of the binary unsigned integer representation of the string.

$$d(i) = i\frac{(x_{max} - x_{min})}{2^l - 1} + x_{min} \tag{2.14}$$

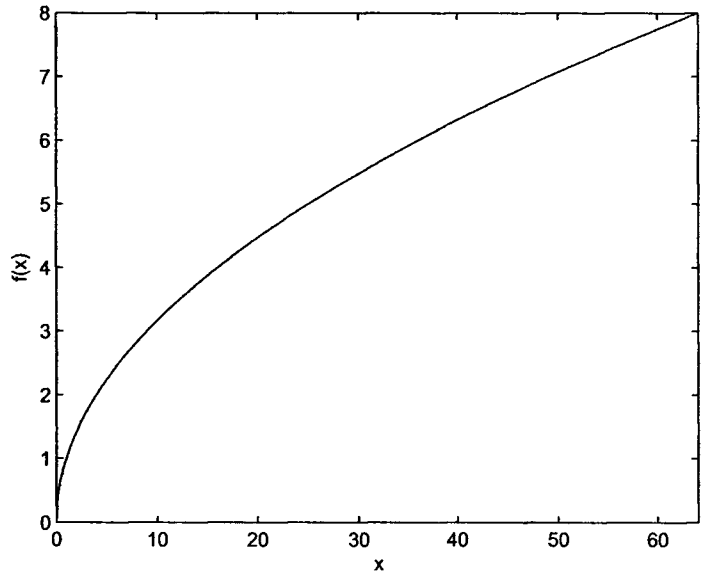


Figure 2.6: Function $f(x) = \sqrt{x}$.

were $d(i)$ is the decodified value of the integer representation of the string i, x_{max} and x_{min} are the limits for x and l is the length of the string. For example, the string (1111), with unsigned integer given by $i = 16$ is decodified into $d(i) = 64$ using equation 2.14. This is one of many ways to decode a string, depending on the application other methods for decodification can be used.

Until now, only the codification and decodification methods have been defined. The following step is to create a population of individuals, for this case the population will be composed only by four individuals. The population is created by giving each individual a string with each bit value randomly selected as 0 or 1. Table 2.1 illustrates the initial population, values for fitness are obtained by evaluating each individual (which is an instance of x) with the function $f(x)$. The parameter $\frac{f_i}{\sum f}$ is the probability of selection for that individual, the higher the value for this parameter, the higher will be its probability to obtain a copy in the mating pool. The expected count $\frac{f_i}{\bar{f}}$ is an estimate of the number of copies each individual will have in the mating pool after selection.

Table 2.2 shows the population after the first cycle. Cross points were randomly selected and resulted in 3 for the first mate and 2 for the second mate. Mutation is set to 2% and acts along all bits in the chromosomes of all individuals, in this case only the last bit in the chromosome of the last individual was mutated. It is known from figure 2.6 that the maximum of $f(x)$ will be when $x = 64$ by string 1111, having fitness equal to $f(x) = 8$.

Table 2.1: Initial random population.

#	Parents (Last Gen.)	Individual String	i	$d(i)$	$f(x)$	$\frac{f_i}{\sum f}$	$\frac{f_i}{\bar{f}}$	Actual Count
1	-	1110	14	59.8	7.73	0.35	1.38	2
2	-	0101	5	22.0	4.69	0.21	0.84	1
3	-	1001	9	38.8	6.22	0.28	1.12	1
4	-	0011	3	13.6	3.68	0.16	0.66	0
Sum					22.32	1	4	4
Average					5.58	0.25	1	1

Table 2.2: Population after the first cycle.

#	Parents (Last Gen.)	Individual String	i	$d(i)$	$f(x)$	$\frac{f_i}{\sum f}$	$\frac{f_i}{\bar{f}}$	Actual Count
1	[1,2]	111|1	15	64.0	8.00	0.30	1.21	2
2	[1,2]	010|0	4	17.8	4.22	0.16	0.64	0
3	[1,3]	11|01	13	55.6	7.46	0.28	1.12	1
4	[1,3]	10|11	11	47.2	6.87	0.26	1.03	1
Sum					26.55	1	4	4
Average					6.64	0.25	1	1

If the solution of the problem requires an improved accuracy for the involved variables, the codifying strings for those variables must be larger and a bigger population must be used. If the evaluation of every individual is costly and the solution requires a good resolution, the algorithm run will take much time. Knowledge about the optimal solution of the problem and its behavior can improve the algorithm's performance greatly.

In the second step the GA has already found the optimal with only eight evaluations of the objective function out of the 16 possible values x can be decodified into. This may not sound surprising because the example is very simple, but for bigger problems with solutions codified in larger strings, the GA works with higher efficiency.

It is known for advance which is the optimal value of x for the fitness function $f(x)$ and the algorithm can be stopped right now. However, in a more complex problem the optimal value or values for the function to be optimized might not be known. Different criterions for stopping the algorithm must be considered in this case, like number of generations, average fitness of the individuals in the current population, variance in the fitness of the individuals, etc.

Chapter 3

Robot Mechanics

3.1 Robot Kinematics

Manipulator-type robots have multiple DOF, are three dimensional, are open loop and are chain mechanisms. Kinematics analysis of robots have been extensively studied and reported in the literature. Analysis for robot manipulators are can be found in [26, 34, 9].

For a kinematic analysis, a three-dimensional reference frame is placed in the base of the robot, denoted by the coordinates $[x\ y\ z]^T$. Links are numbered consecutively from the base (link 0) to the end effector (link n). Joints are contact points between the links and are numbered in such a way that joint i connects the links i and $i-1$. The axis of motion of joint i is assigned to the z axis and identified by z_i. The generalized articular coordinate, denoted by q_i, is the angular displacement around z_i if joint i is rotational, or the lineal displacement over z_i if joint i is translational.

Articular positions for every link are measured with sensors placed in the actuators, which are commonly located in the joints. For a n DOF robot, the articular positions vector $\mathbf{q}$ will have n elements:

$$\boldsymbol{q} = [q_1\ q_2\ \cdots\ q_n]^T \tag{3.1}$$

The position and orientation of the end effector of the robot is essential for most of the applications because is the element that performs the desired task. This position and orientation is expressed in terms of the reference coordinate frame located in the base of the robot. The $\boldsymbol{x}$ vector of Cartesian positions which contains coordinates and angles can be written as follows:

$$\boldsymbol{x} = [x_1\ x_2\ \cdots\ x_m]^T \tag{3.2}$$

where $m \leq n$.

For the case of a 6 DOF robot manipulator, the end effector can be in any position and orientation in the Euclidean three dimensional space, thus $m = 6$. This means

that three components of the $\boldsymbol{x}$ vector refer to the position and the other three to the orientation of the end effector.

The forward kinematics model of a robot describes the relationship between the articular positions $\boldsymbol{q}$ and the position and orientation $\boldsymbol{x}$ of the robot's end effector, as represented by eq. 3.3.

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{q}) \tag{3.3}$$

This model can be obtained following a methodology and involving simple trigonometric expressions and identities. The inverse kinematics model is just the inverse relation of the forward kinematics model as shown below:

$$\boldsymbol{q} = \boldsymbol{f}^{-1}(\boldsymbol{x}) \tag{3.4}$$

Different from the forward kinematics model, obtaining the inverse kinematics of a mechanism is a complicated process and in many cases can exist none or multiple solutions.

One of the more common methods to derive the forward and inverse kinematics of robots is the Denavit-Hartenberg representation, which serves for all possible configuration of robots. For a description of this method refer to [26].

3.2 Robot Dynamics

While the kinematic analysis of robot serves to determine the estimated position of the links and eventually its velocities, the dynamic model of robots relates to the interaction of the system's internal and external forces. Dynamic analysis can determine accelerations of the robot's links based on loads, masses and inertias. This document focus on the control of the dynamics of a 2 DOF robot, the simulation results presented in chapter 5 are obtained using the dynamic model developed in this section.

To be able to accelerate a mass and take it near or to a determined position it is needed to exert a force on it. The same analogy can be made to angular positions, to cause an angular acceleration in a rotating body a torque must be applied on it. In the case of robots, to accelerate any link, it is necessary to have actuators that are capable of exerting large enough forces and torques on the links and joints to move them at a desired acceleration and velocity. To be able to calculate how strong each actuator must be, it is necessary to determine the dynamic relationships that govern the motions of the robot, i.e. the force-mass-acceleration and the torque-inertia-angular acceleration relationships.

The dynamic equations may be used to find the equations of motion of mechanisms. Knowing the forces and torques, it can be determined how a mechanism will

move. A dynamic model of a robot consists of a vectorial differential equation that can be expressed in terms of $\boldsymbol{q}$ as:

$$\boldsymbol{f}(\boldsymbol{q}, \dot{\boldsymbol{q}}, \ddot{\boldsymbol{q}}, \tau) = 0 \tag{3.5}$$

In the last equation, $\boldsymbol{f}$ is not the same function which appears in equation 3.3. Obtaining the dynamic model is helpful when developing a computer simulation of the robot.

Techniques such as Newtonian mechanics can be used to find the dynamic equations for robots. However, due to the fact that robots are three-dimensional, multiple DOF mechanisms with distributed masses, it is very difficult to use Newtonian mechanics. Lagrangian mechanics is based on energy terms and for complex mechanisms as robots, it is easier to use.

3.2.1 Lagrange equations of motion

As commented before, the dynamic equations of motion for a robot manipulator can be obtained directly from Newton's equations. This is possible, but for a robot with n DOF, the analysis becomes mucho more complex. For this case, the use of the Lagrange equations of motion is a more convenient way when obtaining a dynamic model. This section only describes the part of the Lagrange analysis that leads to the development of a dynamic model for a 2 DOF robot manipulator. A deeper analysis and derivation of this equations can be found in [7].

Consider a n DOF robot, the total energy of the robot $\mathcal{E}$ is given by the sum of its kinetic $\mathcal{K}$ and potential energy $\mathcal{U}$. This relation can be expressed as shown in eq. 3.6.

$$\mathcal{E}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = \mathcal{K}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) + \mathcal{U}(\boldsymbol{q}(t)) \tag{3.6}$$

where $\boldsymbol{q} = [q_1\ q_2\ \cdots\ q_n]^T$.

The Lagrangian $\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is defined as the difference between the kinetic energy of the system and the potential energy of the system, as follows:

$$\mathcal{L}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = \mathcal{K}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) - \mathcal{U}(\boldsymbol{q}(t)). \tag{3.7}$$

For a robot manipulator, the Lagrange equations of motion can be written for each joint as:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{q}_i}\right] - \frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial q_i} = \tau_i \tag{3.8}$$

where $i = 1, \cdots, n$ and τ_i are the external forces and torques applied to the robot, as those exerted by the actuators, friction forces, loads, etc.

Equation 3.8 can also be written in vectorial form as:

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{\boldsymbol{q}}}\right] - \frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \boldsymbol{q}} = \boldsymbol{\tau}. \tag{3.9}$$

3.2.2 Lagrangian for a 2 DOF robot manipulator

Consider the robot manipulator shown in figure 3.1. The robot is composed by 2 rigid links with length l_1 and l_2 and masses m_1 and m_2 respectively. Joints 1 and 2 are rotational so the displacements are along the vertical plane $x - y$. Distance between the rotary axis and the center of mass (COM) for each link is denoted by l_{c1} and l_{c2} respectively. The moments of inertia are given by I_1 and I_2 for link 1 and 2 respectively. The angular positions are measured for q_1 starting from the negative y axis and for q_2 from the extension of link 1 to link 2. Both positions are positive clockwise.

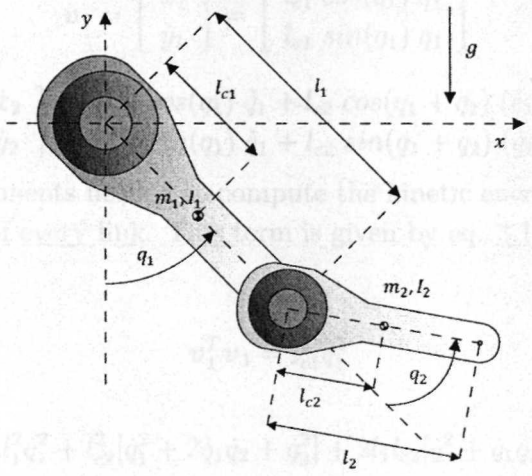


Figure 3.1: Sketch of a 2 DOF Robot Manipulator

For this case, the articular positions vector $\boldsymbol{q}(t)$ is defined as follows:

$$\boldsymbol{q} = [q_1(t)\ q_2(t)]^T. \tag{3.10}$$

As the robot is composed of two links, the total kinetic energy of the system $\mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ is given by the sum of the kinetic energy of each link:

$$\mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{K}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathcal{K}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}). \tag{3.11}$$

The coordinates in the $x - y$ plane for the COM of link 1 are easily derived and found to be:

$$\begin{aligned} x_1 &= l_{c1}\, sin(q_1) \\ y_1 &= -l_{c1}\, cos(q_1) \end{aligned} \tag{3.12}$$

similarly, coordinates for the COM of link 2 are derived and shown below:

$$\begin{aligned} x_2 &= l_1\, sin(q_1) + l_{c2}\, sin(q_1 + q_2) \\ y_2 &= -l_1\, cos(q_1) - l_{c2}\, cos(q_1 + q_2). \end{aligned} \tag{3.13}$$

For this configuration of robot, it is a relatively simple task to find the expressions above. Links move within a bidimensional plane.

The velocity terms can be found by derivation of the expressions for the coordinates of the COM for each link with respect to time. These expressions are given by eq. 3.14 and eq. 3.15 for link 1 and link 2 respectively.

$$\boldsymbol{v}_1 = \begin{bmatrix} \dot{x}_1 \\ \dot{y}_1 \end{bmatrix} = \begin{bmatrix} l_{c1}\, cos(q_1)\, \dot{q}_1 \\ l_{c1}\, sin(q_1)\, \dot{q}_1 \end{bmatrix} \tag{3.14}$$

$$\boldsymbol{v}_2 = \begin{bmatrix} \dot{x}_2 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} l_1\, cos(q_1)\, \dot{q}_1 + l_{c2}\, cos(q_1 + q_2)\, (\dot{q}_1 + \dot{q}_2) \\ l_1\, sin(q_1)\, \dot{q}_1 + l_{c2}\, sin(q_1 + q_2)\, (\dot{q}_1 + \dot{q}_2) \end{bmatrix} \tag{3.15}$$

One of the components needed to compute the kinetic energy is the square of the velocity of the COM of every link. This term is given by eq. 3.16 and eq. 3.17 for link 1 and 2 respectively.

$$\boldsymbol{v}_1^T \boldsymbol{v}_1 = l_{c1}^2 \dot{q}_1^2 \tag{3.16}$$

$$\boldsymbol{v}_2^T \boldsymbol{v}_2 = l_1^2 \dot{q}_1^2 + l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2] + 2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] cos(q_2) \tag{3.17}$$

With the last equations derived, the kinetic energy for link 1 can be expressed as:

$$\begin{aligned} \mathcal{K}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \tfrac{1}{2} m_1 \boldsymbol{v}_1^T \boldsymbol{v}_1 + \tfrac{1}{2} I_1 \dot{q}_1^2 \\ &= \tfrac{1}{2} m_1 l_{c1}^2 \dot{q}_1^2 + \tfrac{1}{2} I_1 \dot{q}_1^2 \end{aligned} \tag{3.18}$$

and for link 2:

$$\begin{aligned} \mathcal{K}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \tfrac{1}{2} m_2 \boldsymbol{v}_2^T \boldsymbol{v}_2 + \tfrac{1}{2} I_2 [\dot{q}_1 + \dot{q}_2]^2 \\ &= \tfrac{1}{2} m_2 l_1^2 \dot{q}_1^2 + \tfrac{1}{2} m_2 l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1 \dot{q}_2 + \dot{q}_2^2] \\ &\quad + m_2 l_1 l_{c2} [\dot{q}_1^2 + \dot{q}_1 \dot{q}_2] cos(q_2) \\ &\quad + \tfrac{1}{2} I_2 [\dot{q}_1 + \dot{q}_2]^2. \end{aligned} \tag{3.19}$$

The total potential energy of the system is also composed of the sum of the potential energy of each link:

$$\mathcal{U}(\boldsymbol{q}) = \mathcal{U}_1(\boldsymbol{q}) + \mathcal{U}_2(\boldsymbol{q}). \tag{3.20}$$

For link 1, the potential energy is expressed by eq. 3.21 and for link 2 by eq.3.22.

$$\mathcal{U}_1(\boldsymbol{q}) = -m_1 l_{c1}\, g\, cos(q_1) \tag{3.21}$$

$$\mathcal{U}_2(\boldsymbol{q}) = -m_2 l_1\, g\, cos(q_1) - m_2 l_{c2}\, g\, cos(q_1 + q_2) \tag{3.22}$$

Having determined the kinetic and potential energies, the Lagrangian of the system is found to be:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{U}(\boldsymbol{q}) \\
&= \mathcal{K}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \mathcal{K}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{U}_1(\boldsymbol{q}) - \mathcal{U}_2(\boldsymbol{q}) \\
&= \tfrac{1}{2}[m_1 l_{c1}^2 + m_2 l_1^2]\dot{q}_1^2 + \tfrac{1}{2} m_2 l_{c2}^2 [\dot{q}_1^2 + 2\dot{q}_1\dot{q}_2 + \dot{q}_2^2] \\
&\quad + m_2 l_1 l_{c2}[\dot{q}_1^2 + \dot{q}_1\dot{q}_2] cos(q_2) \\
&\quad + [m_1 l_{c1} + m_2 l_1]\, g\, cos(q_1) \\
&\quad + m_2 l_{c2}\, g\, cos(q_1 + q_2) \\
&\quad + \tfrac{1}{2} I_1 \dot{q}_1^2 + \tfrac{1}{2} I_2 [\dot{q}_1 + \dot{q}_2]^2.
\end{aligned} \tag{3.23}$$

Derivatives of the robot's Lagrangian are now found in order to derive the equation of motion for each link in the system. For link 1 are expressed by eq. 3.24, eq. 3.25 and eq. 3.26.

$$\frac{\partial \mathcal{L}}{\partial q_1} = -[m_1 l_{c1} + m_2 l_1] g\, sin(q_1) - m_2\, g\, l_{c2} sin(q_1 + q_2). \tag{3.24}$$

$$\begin{aligned}
\tfrac{\partial \mathcal{L}}{\partial \dot{q}_1} &= [m_1 l_{c1}^2 + m_2 l_1^2]\dot{q}_1 + m_2 l_{c2}^2 \dot{q}_1 + m_2 l_{c2}^2 \dot{q}_2] \\
&\quad + 2 m_2 l_1 l_{c2} \dot{q}_1 cos(q_2) + m_2 l_1 l_{c2} \dot{q}_2 cos(q_2) \\
&\quad + I_1 \dot{q}_1 + I_2[\dot{q}_1 + \dot{q}_2].
\end{aligned} \tag{3.25}$$

$$\begin{aligned}
\tfrac{d}{dt}\left[\tfrac{\partial \mathcal{L}}{\partial \dot{q}_1}\right] &= [m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2 m_2 l_1 l_{c2} cos(q_2)]\ddot{q}_1 \\
&\quad + [m_2 l_{c2}^2 + m_2 l_1 l_{c2} cos(q_2)]\ddot{q}_2 \\
&\quad - 2 m_2 l_1 l_{c2} sin(q_2)\dot{q}_1\dot{q}_2 - m_2 l_1 l_{c2} sin(q_2)\dot{q}_2^2 \\
&\quad + I_1 \ddot{q}_1 + I_2[\ddot{q}_1 + \ddot{q}_2].
\end{aligned} \tag{3.26}$$

For link 2 by eq. 3.27, eq. 3.28 and eq. 3.29.

$$\frac{\partial \mathcal{L}}{\partial q_2} = -m_2 l_1 l_{c2} sin(q_2)[\dot{q}_1\dot{q}_2 + \dot{q}_1^2] - m_2\, g\, l_{c2} sin(q_1 + q_2). \tag{3.27}$$

$$\frac{\partial \mathcal{L}}{\partial \dot{q}_2} = m_2 l_{c2}^2 \dot{q}_1 + m_2 l_{c2}^2 \dot{q}_2 + m_2 l_1 l_{c2} cos(q_2)\dot{q}_1 + I_2[\dot{q}_1 + \dot{q}_2]. \tag{3.28}$$

$$\begin{aligned} \frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{q}_2}\right] &= m_2 l_{c2}^2 \ddot{q}_1 + m_2 l_{c2}^2 \ddot{q}_2 \\ &+ m_2 l_1 l_{c2} cos(q_2)\ddot{q}_1 - m_2 l_1 l_{c2} sin(q_2)\dot{q}_1 \dot{q}_2 \\ &+ I_2[\ddot{q}_1 + \ddot{q}_2]. \end{aligned} \tag{3.29}$$

Having derived the last equations, the final step is to merge those equations in the Lagrange equation of motion for each link, eq. 3.8 as shown below for $i = 1, 2$.

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial \dot{q}_i}\right] - \frac{\partial \mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}})}{\partial q_i} = \tau_i \tag{3.30}$$

For link 1, the dynamic equation of motion is given by eq. 3.31. While for link 2 by eq. 3.32.

$$\begin{aligned} \tau_1 &= [m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2 m_2 l_1 l_{c2} cos(q_2) + I_1 + I_2]\ddot{q}_1 \\ &+ [m_2 l_{c2}^2 + m_2 l_1 l_{c2} cos(q_2) + I_2]\ddot{q}_2 \\ &- 2 m_2 l_1 l_{c2} sin(q_2)\dot{q}_1 \dot{q}_2 - m_2 l_1 l_{c2} sin(q_2)\dot{q}_2^2 \\ &+ [m_1 l_{c1} + m_2 l_1] g\ sin(q_1) + m_2\ g\ l_{c2} sin(q_1 + q_2). \end{aligned} \tag{3.31}$$

$$\begin{aligned} \tau_2 &= [m_2 l_{c2}^2 + m_2 l_1 l_{c2} cos(q_2) + I_2]\ddot{q}_1 + [m_2 l_{c2}^2 + I_2]\ddot{q}_2 \\ &+ m_2 l_1 l_{c2} sin(q_2)\dot{q}_1^2 + m_2\ g\ l_{c2} sin(q_1 + q_2) \end{aligned} \tag{3.32}$$

3.2.3 Dynamic model of Robot Manipulators

This section make use of the dynamic equations of motion found in the last section to obtain the dynamic model of a 2 DOF robot. The kinetic energy $\mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ of a mechanical articulated device composed by n links can be expressed as:

$$\mathcal{K}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.33}$$

where $M(\boldsymbol{q})$ for a is a symmetric positive definite $n \times n$ inertia matrix. For the case of the potential energy $\mathcal{U}(\boldsymbol{q})$, a specific expression is not known, but it depends of the articular positions vector $\boldsymbol{q}$.

The Lagrangian as defined in eq. 3.7 now using eq. 3.33 turns to be:

$$\mathcal{L}(\boldsymbol{q}(t), \dot{\boldsymbol{q}}(t)) = \frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}} - \mathcal{U}(\boldsymbol{q}(t)) \tag{3.34}$$

and thus, the Lagrange equation of motion can be rewritten as follows:

$$\frac{d}{dt}\left[\frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right]\right] - \frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right] + \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{\tau}. \tag{3.35}$$

It can be noted that:

$$\frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right] = M(\boldsymbol{q})\dot{\boldsymbol{q}} \tag{3.36}$$

and thus:

$$\frac{d}{dt}\left[\frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\frac{1}{2}\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\dot{\boldsymbol{q}}\right]\right] = M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}}. \tag{3.37}$$

So eq. 3.35 can be written again as:

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\right]\dot{\boldsymbol{q}} + \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \dot{\boldsymbol{q}}} = \boldsymbol{\tau}. \tag{3.38}$$

Substituting expressions in 3.39 and 3.40 into the last equation, the dynamic equation of motion for n DOF robots can be expressed compactly as eq. 3.41:

$$C(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{M}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\frac{\partial}{\partial \dot{\boldsymbol{q}}}\left[\dot{\boldsymbol{q}}^T M(\boldsymbol{q})\right]\dot{\boldsymbol{q}} \tag{3.39}$$

$$g(\boldsymbol{q}) = \frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \dot{\boldsymbol{q}}} \tag{3.40}$$

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{3.41}$$

where $C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}$ is a $n \times 1$ vector known as the Coriolis and centrifugal forces vector. The term $\boldsymbol{g}(\boldsymbol{q})$ is a $n \times 1$ vector of gravity forces and $\boldsymbol{\tau}$ is also a $n \times 1$ vector which contains the external forces to the system.

An important part of a real mechanical system is friction, which for this model can be described as the function $\boldsymbol{f}(\dot{\boldsymbol{q}})$. Therefore, the system model can be re-written as equation 3.42.

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{f}(\dot{\boldsymbol{q}}) = \boldsymbol{\tau} \tag{3.42}$$

Assuming that friction is modeled by viscous and Coulomb effects, the friction terms in the robot dynamics can be described by:

$$\boldsymbol{f}(\dot{\boldsymbol{q}}) = +F_v\dot{\boldsymbol{q}} + F_c\,\mathrm{sign}(\dot{\boldsymbol{q}}) \tag{3.43}$$

where F_v and F_c are $n \times n$ diagonal matrices whose positive entries denote the viscous and Coulomb coefficients of each joint. The term $\mathrm{sign}(\dot{\boldsymbol{q}})$ is a $n \times 1$ vector expressed as follows:

$$\text{sign}(\dot{\boldsymbol{q}}) = \begin{bmatrix} \text{sign}(\dot{q}_1) \\ \text{sign}(\dot{q}_2) \\ \vdots \\ \text{sign}(\dot{q}_n) \end{bmatrix} \tag{3.44}$$

and $\text{sign}(x)$ is the sign function given by:

$$\begin{aligned} \text{sign}(x) &= +1 \quad \text{if} \quad x > 0 \\ \text{sign}(x) &= -1 \quad \text{if} \quad x < 0. \end{aligned} \tag{3.45}$$

Furthermore, to implement a computer simulation of the model, equation 3.42 can be solved for $\ddot{\boldsymbol{q}}$ as shown below.

$$\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q})\left(\boldsymbol{\tau} - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{f}(\dot{\boldsymbol{q}})\right) \tag{3.46}$$

where variables $\dot{\boldsymbol{q}}$ and $\boldsymbol{q}$ are obtained by integration of $\ddot{\boldsymbol{q}}$.

3.2.4 Dynamic model of a 2 DOF Robot Manipulator

For a 2 DOF robot manipulator as shown in fig. 3.1, eq. 3.42 is composed as follows:

$$\begin{aligned} M(\boldsymbol{q}) &= \begin{bmatrix} M_{11}(\boldsymbol{q}) & M_{12}(\boldsymbol{q}) \\ M_{21}(\boldsymbol{q}) & M_{22}(\boldsymbol{q}) \end{bmatrix} \\ C(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \begin{bmatrix} C_{11}(\boldsymbol{q}, \dot{\boldsymbol{q}}) & C_{12}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ C_{21}(\boldsymbol{q}, \dot{\boldsymbol{q}}) & C_{22}(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} \\ \boldsymbol{g}(\boldsymbol{q}) &= \begin{bmatrix} g_1(\boldsymbol{q}) \\ g_2(\boldsymbol{q}) \end{bmatrix} \\ \boldsymbol{f}(\dot{\boldsymbol{q}}) &= \begin{bmatrix} f_1(\dot{\boldsymbol{q}}) \\ f_2(\dot{\boldsymbol{q}}) \end{bmatrix} \\ \boldsymbol{\tau} &= \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} \end{aligned} \tag{3.47}$$

where each matrix component is defined as:

$$
\begin{aligned}
M_{11}(\boldsymbol{q}) &= m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2 m_2 l_1 l_{c2} cos(q_2) + I_1 + I_2 \\
M_{12}(\boldsymbol{q}) &= m_2 l_{c2}^2 + m_2 l_1 l_{c2} cos(q_2) + I_2 \\
M_{21}(\boldsymbol{q}) &= m_2 l_{c2}^2 + m_2 l_1 l_{c2} cos(q_2) + I_2 \\
M_{22}(\boldsymbol{q}) &= m_2 l_{c2}^2 + I_2 \\
C_{11}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= -m_2 l_1 l_{c2} sin(q_2) \dot{q}_2 \\
C_{12}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= -m_2 l_1 l_{c2} sin(q_2) [\dot{q}_2 + \dot{q}_2] \\
C_{21}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= m_2 l_1 l_{c2} sin(q_2) \dot{q}_1 \\
C_{22}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= 0 \\
g_1(\boldsymbol{q}) &= [m_1 l_{c1} + m_2 l_1] g \; sin(q_1) + m_2 \; g \; l_{c2} sin(q_1 + q_2) \\
g_2(\boldsymbol{q}) &= m_2 \; g \; l_{c2} sin(q_1 + q_2) \\
f_1(\dot{\boldsymbol{q}}) &= f_{v_1} \dot{q}_1 + f_{c_1} \text{sign}(\dot{q}_1) \\
f_2(\dot{\boldsymbol{q}}) &= f_{v_2} \dot{q}_2 + f_{c_2} \text{sign}(\dot{q}_2)
\end{aligned}
\tag{3.48}
$$

These expressions are used to develop a computer simulation of a 2 DOF robot for which the control algorithms presented in chapter 4 are tested and compared.

3.2.5 Dynamic model parameters

A real experimental robot with these parameters has been built at CICESE research center. Parameters used for the simulation of a 2DOF robot as shown in fig. 3.1 are reported next:

$$
\begin{aligned}
g &= 9.81 \; m/sec^2 \\
m_1 &= 23.90 \; kg \\
m_2 &= 3.88 \; kg \\
lc_1 &= 0.091 \; m \\
lc_2 &= 0.048 \; m \\
l_1 &= 0.45 \; m \\
l_2 &= 0.45 \; m \\
I_1 &= 1.266 \; kg \; m^2 \\
I_2 &= 0.093 \; kg \; m^2. \\
f_{c_1} &= 7.17 \; N \; m. \\
f_{c_2} &= 1.734 \; N \; m.
\end{aligned}
$$

$$
\begin{aligned}
f_{v_1} &= 2.288\ N\ m\ seg/^\circ. \\
f_{v_2} &= 0.175\ N\ m\ seg/^\circ.
\end{aligned}
$$

These parameters used for the simulation where found by experimental evaluation of three identification schemes to determine the dynamic parameters of a 2 DOF direct-drive robot [32]. Actuators are direct-drive motors, are DM series motors from Parker Compumotor. Motors are operated in torque mode so they act as torque source and accept an analog voltage as reference for torque signal. According to the manufacturer, the direct-drive motors are able to supply torques within the following bounds:

$$
\begin{aligned}
\tau_1^{max} &= 150[Nm] \\
\tau_2^{max} &= 15[Nm].
\end{aligned}
\tag{3.49}
$$

The physical control algorithm is executed at 2.5 msec sampling rate in a control board (based on a DSP 32-bit floating point microprocessor) [18]. For purpose of this thesis, it is implemented in MATLAB-Simulink along with the robot dynamical model.

Chapter 4

Controller design

In the last chapter, the dynamic model of the robot was obtained as equation 3.46 and shown below again.

$$\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q})\left(\boldsymbol{\tau} - C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q}) - \boldsymbol{f}(\dot{\boldsymbol{q}})\right) \tag{4.1}$$

As commented before, $\boldsymbol{\tau}$ is a $n \times 1$ vector which contains the external forces or torques to the system. If the robot's link are specified to move towards some desired angular position $\boldsymbol{q}_d$ or to follow a trajectory with a given angular velociy $\dot{\boldsymbol{q}}_d$, these torques must be calculated in order to make the links move as specified. The task of applying the necessary torques to the system is accomplished with a controller defined by a control law. Depending of the reference and the actual position, the controller applies a torque τ to the link.

This section deals with two kind of robot's links controllers, which are used for different tasks. The first kind are position controllers, links are given a desired angular position $\boldsymbol{q}_d$ and the controller must be able to make the link approach to that position without overshoot and steady state error. Position controllers doesn't care about the velocity of the link, the goal is to move the link towards some desired angular position and keep it there until a new desired position is set. Two position controllers are introduced in this chapter, one is the Proportional Integral Derivative (PID) Controller, the other is the Fuzzy Self-tuning (FST) PID Controller, which is an extension of the former.

The second kind are velocity controllers, by its name these controllers regulate the position and velocity of each link. The desired position is given by a trajectory and the desired velocity is just the derivative of the position, furthermore the desired acceleration can also be computed. Also two velocity controllers are presented, the first is the Fuzzy Self-tuning PD+ (FST PD+) Controller and the second is a Fuzzy-Sliding Mode (FSM) Controller.

4.1 Position Controllers

4.1.1 PID Controller

This section describes the structure of the PID and its application for the control of the robot's arm links. Many of the commercial and industrial robot arms are still PID controllers, although numerous methods like adaptive control, neural control and fuzzy control have been studied. The PID controller is very robust if parameters are adjusted properly and its performance can be enough to complete the tasks of many of the more common applications.

The introduction of this popular controller is claimed by the Taylor Instrument Company in 1936 when *preact*, that is, derivative action, was added to their *double response* controller. The use of derivative and integral action was, in the 1930s not new: many controllers using it had been designed and used throughout the nineteenth century. A consequence of the gradual introduction of such controllers into the process industries was a growing interest in the dynamics of various typical processes and attempts to analyze the behavior of controllers [1].

However, Nicolas Minorsky in 1922 in his paper "Directional stability of automatically steered bodies" had already analyzed and discussed the properties of the three-term controller [24]. This paper stands as one of the early formal discussions of control theory.

Structure for this controller is illustrated in figure 4.1. The equation for a PID controller can be described as follows:

$$\boldsymbol{\tau} = \boldsymbol{K}_p\tilde{\boldsymbol{q}} + \boldsymbol{K}_v\dot{\tilde{\boldsymbol{q}}} + \boldsymbol{K}_i\int_0^t \tilde{\boldsymbol{q}}\,dt \tag{4.2}$$

where $\tilde{\boldsymbol{q}} = \boldsymbol{q}_d - \boldsymbol{q}$ is the error of the angular position of the link with respect to the desired angular position, $\dot{\tilde{\boldsymbol{q}}} = \dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}$ is the error of the angular velocity of the link with respect to the desired angular velocity. Terms $\boldsymbol{K}_p$, $\boldsymbol{K}_v$ and $\boldsymbol{K}_i$ are 2×2 diagonal matrices with constants defining the PIDs for both links.

This is the simplest of the controllers that will be defined in this document. Although its simple structure, it is difficult to optimize the parameter settings of this controller for robot arms because these systems have serious non-linearities and strong couplings. Chapter 5 describes how this controller's parameters will be modified by a GA (an optimization method) to compare its performance with a more complex controller, the FST PID controller, which is described next. For all controllers, chapter 5 describes how its parameter's settings will be optimized.

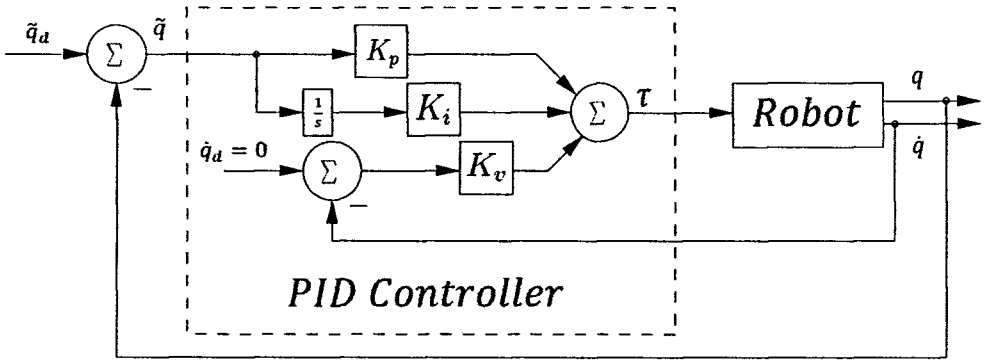


Figure 4.1: PID block diagram

4.1.2 Fuzzy Self-Tuning PID Controller

The PID Controller described in the last section is a linear controller and can handle most linear systems with great performance. Real-world object's dynamics are commonly non-linear and therefore a linear controller may handle the system but performance can be seriously affected in some cases. This control scheme has been successfully applied to a flying robot in [30]. Also, in [22] this controller has been optimized with a Genetic Algorithm for the same robot manipulator model, resulting in an enhanced response accuracy and speed.

The FST PID Controller is an extension to the PID Controller for non-linear systems. It consists of a fuzzy system tuning online the PID controller settings depending on $\tilde{q}_i$ and $\dot{\tilde{q}}_i$. Equation of this controller is expressed as:

$$\boldsymbol{\tau} = K_p(\tilde{\boldsymbol{q}})\tilde{\boldsymbol{q}} + K_v(\tilde{\boldsymbol{q}})\dot{\tilde{\boldsymbol{q}}} + K_i(\tilde{\boldsymbol{q}}) \int_0^t \tilde{\boldsymbol{q}}\, dt \tag{4.3}$$

where $K_p(\tilde{\boldsymbol{q}})$, $K_i(\tilde{\boldsymbol{q}})$ and $K_v(\tilde{\boldsymbol{q}})$ are 2×2 diagonal matrices with entries $K_{p_i}(\tilde{q}_i)$, $K_{i_i}(\tilde{q}_i)$ and $K_{v_i}(\tilde{q}_i)$ respectively.

Figure 4.2 shows a block diagram of the FST PID Controller.

In order to tune the gains according to the input, a conceptual Fuzzy Logic Tuner (FLT) is defined. The conceptual FLT is composed by one input $|x|$ and the corresponding output y (see figure 4.2), which can be seen as a static mapping H defined by

$$\begin{array}{rccc} H: & \mathbb{R}_+ & \rightarrow & \mathbb{R} \\ & |x| & \mapsto & y\,. \end{array} \tag{4.4}$$

The universes of discourse of $|x|$ and y are partitioned into three fuzzy sets: B (*Big*), M (*Medium*) and S (*Small*) each described by a MF. Trapezoidal MFs are used for input variables and singleton MFs for output variables, this is illustrated in figure

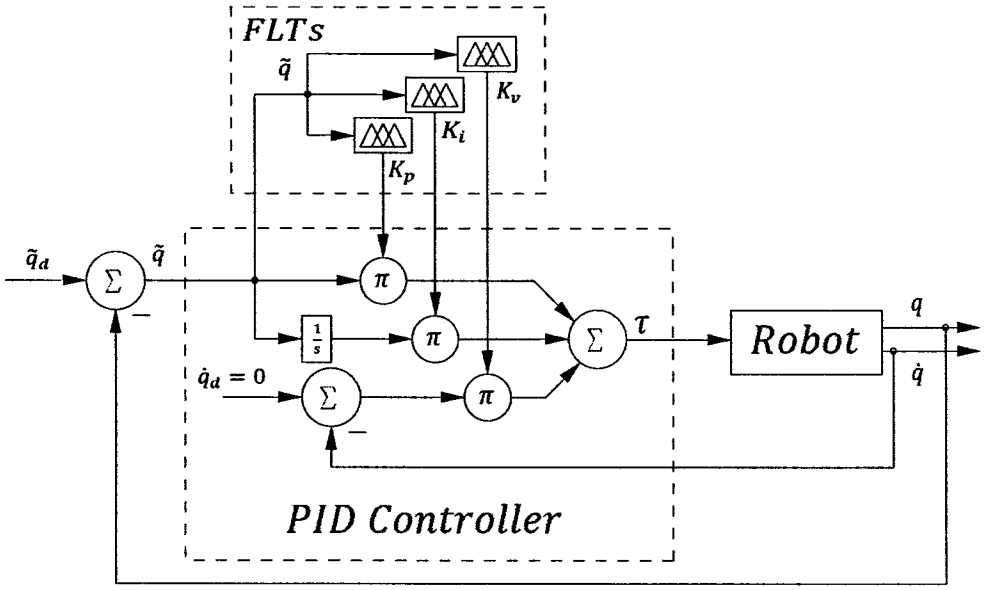


Figure 4.2: FST PID block diagram

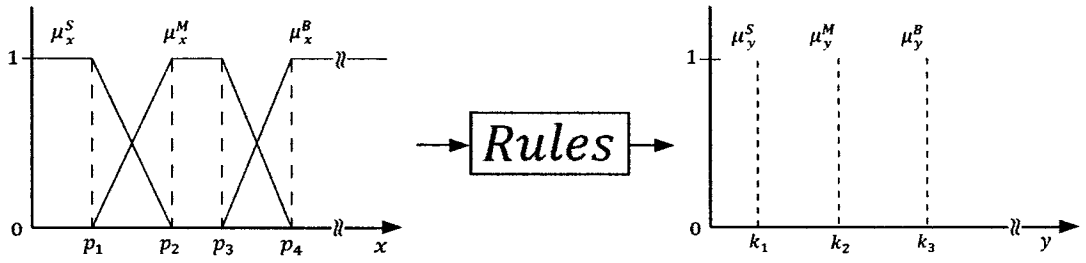


Figure 4.3: FLT parameters.

4.3. The corresponding *Small*, *Medium* and *Big* MFs for the input variable $|x|$ are denoted by

$$\boldsymbol{\mu}_x(|x|\,;\boldsymbol{p}) = \begin{bmatrix} \mu_x^B(|x|\,;p_3,p_4) \\ \mu_x^M(|x|\,;p_1,p_2,p_3,p_4) \\ \mu_x^S(|x|\,;p_1,p_2) \end{bmatrix} \tag{4.5}$$

For the output variable y, the corresponding singleton MFs to *Small*, *Medium* and *Big* are represented by

$$\boldsymbol{\mu}_y(\cdot;\boldsymbol{k}) = \begin{bmatrix} \mu_y^S(\cdot;k_1) \\ \mu_y^M(\cdot;k_2) \\ \mu_y^B(\cdot;k_3) \end{bmatrix} \tag{4.6}$$

For the case of FST controllers the FLT is a single-input, single-output fuzzy logic system. For the controller of each link there are three FLTs, one for each gain of the PID portion of the controller. The input for each FLT (denoted by $|x|$) is always the absolute error of the current link $|\tilde{q}_i|$. The output of the FLT (denoted by y) is the gain of each component of the PID. The defuzzification method used is Weighted-Average and the fuzzy model is Sugeno.

This controller is much more complex than the PID, to optimize the fuzzy systems many parameters must be set correctly and no theoretical method is known to accomplish this task in one step. It is an iterative procedure of trial and error and therefore a GA is a well suited method to search the optimal parameters or at least the closest possible in a finite time search.

4.2 Velocity Controllers

A basic problem in control of robot's dynamics is the so-called motion control, where a manipulator is requested to track a desired position trajectory. This section describes the velocity controllers, which accomplish this task. Two different velocity controllers are presented in this section, the FST PD+ controller and the FSM Controller. In chapter 5 the methodology to optimize parameters settings and performance comparison of both controllers is presented.

4.2.1 Fuzzy Self-Tuning PD+ Controller

When nonlinearities are not severe, local linearization can be used to derive linear models which are approximations of the nonlinear equations near the operating point [9]. The manipulator control problem cannot be linearized because no linearization valid for all regions can be found. Instead of linearizing for an operating point, the controller can be designed to cancel the non-linearity by compensating the robot's dynamics using the developed model of the system.

Computed Torque Method

Using the system model in the control law to linearize the system is known as the **computed torque method**, first proposed in [29]. A control scheme of this type might be called a linearizing control law, since it uses a non-linear control term to

cancel a non-linearity in the controlled system such that the overall closed loop system is linear. In a linearizing control law scheme, there exists a servo law (commonly a linear controller) and a model-based portion with the model of the non-linearity of the system.

The problem of controlling a robot manipulator is a multi-input, multi-output (MIMO) problem. A vector of desired joint positions, velocities and accelerations are given and the control law must compute a vector of joint actuator signals. The linearizing control law scheme is applicable and appears in a matrix-vector form. Equation 4.7 shows a generic linearizing control law:

$$F = \alpha F' + \beta \tag{4.7}$$

where, for a system with n controlled variables, F, F' and β are $n \times 1$ vectors and α is an $n \times n$ matrix. The matrix α is chosen to decouple the n dynamic equations of each controlled variable. If the system model is accurate, the linearization and decoupling will compensate correctly the system and the servo law can be defined in a simpler way. An example of using a PD for the servo law is given by the equation below:

$$F' = \ddot{X}_d + K_p E + K_v \dot{E} \tag{4.8}$$

where K_p and K_v are $n \times n$ diagonal matrices of constant gains, E and $\dot{E}$ are $n \times 1$ vectors of errors in position and velocity respectively.

Ideally, the non-linearities in the system would be cancelled with the model-based portion of the linearizing control law and the servo law would make the system a linear closed loop system. A practical problem arises when the parameters and the structure of the non-linear system are not known accurately.

Fuzzy Self-Tuning PD+ Control for Robot Manipulators

Taking only the internal forces of the system (ignoring friction terms), the manipulator dynamics can be written as equation 4.9.

$$M(\boldsymbol{q})\ddot{\boldsymbol{q}} + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) = \boldsymbol{\tau} \tag{4.9}$$

As has been described in the last section, the linearizing control law must have the following shape:

$$\boldsymbol{\tau} = \alpha \boldsymbol{\tau}' + \beta. \tag{4.10}$$

where $\boldsymbol{\tau}$ is the $n \times 1$ vector of joint torques. Terms for the model-based portion of the control law are chosen from equation 4.9 as follows:

$$\alpha = M(\boldsymbol{q}) \tag{4.11}$$

$$\beta = C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \tag{4.12}$$

The servo law can be defined similarly as equation 4.8 for the robot manipulator in matrix form:

$$\boldsymbol{\tau}' = \ddot{\boldsymbol{q}}_d + K_v\dot{\tilde{\boldsymbol{q}}} + K_p\tilde{\boldsymbol{q}} \tag{4.13}$$

finally, substituting equation 4.13 into 4.10 the linearizing control law is given by the next equation:

$$\boldsymbol{\tau} = M(\boldsymbol{q})\left(\ddot{\boldsymbol{q}}_d + K_v\dot{\tilde{\boldsymbol{q}}} + K_p\tilde{\boldsymbol{q}}\right) + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_d + \boldsymbol{g}(\boldsymbol{q}). \tag{4.14}$$

So in the closed loop system with the linearizing control law, the next equation describes the system:

$$\ddot{\tilde{\boldsymbol{q}}}_d + K_v\dot{\tilde{\boldsymbol{q}}} + K_p\tilde{\boldsymbol{q}} = 0 \tag{4.15}$$

which ideally behaves as a second-order linear system.

With both portions of the linearizing control law, the controller just defined is known as Computed-torque Control. It was introduced in [29] and a fuzzy supervisor scheme was presented later in [17]. This controller is robust and can handle very well some tasks, but its performance is enhanced when gains are allowed to vary according with a fuzzy logic system which depends on the robot's states. This controller was introduced in [35], with the name of Fuzzy PD+. Control law for this controller is simplified to:

$$\boldsymbol{\tau} = K_p(\tilde{\boldsymbol{q}})\tilde{\boldsymbol{q}} + K_v(\tilde{\boldsymbol{q}})\dot{\tilde{\boldsymbol{q}}} + M(\boldsymbol{q})\ddot{\boldsymbol{q}}_d + C(\boldsymbol{q}, \dot{\boldsymbol{q}})\dot{\boldsymbol{q}}_d + \boldsymbol{g}(\boldsymbol{q}). \tag{4.16}$$

where $K_p(\tilde{\boldsymbol{q}})$ and $K_v(\tilde{\boldsymbol{q}})$ are $n \times n$ diagonal positive definite matrices whose diagonal entries are denoted by $K_{p_i}(\tilde{q}_i)$ and $K_{v_i}(\tilde{q}_i)$ respectively.

This controller is similar as the FST PID, both use the same kind of FLT for tuning online the servo control law. For convenience and standardization, this controller will be named Fuzzy Self-tuning PD+ in this document, given its similarities with the structure of the FST PID. The input for each FLT is the absolute error of the current link $|\tilde{q}_i|$ and its output is one of the two gains of the PD portion of the controller for each link, therefore two FLTs are needed for this controller.

A block diagram of this controller is shown in 4.4. In this figure the FLTs are represented by a single one-input, two-output block, because the input for both fuzzy systems is the same and fuzzy logic systems can have as many inputs and output as needed.

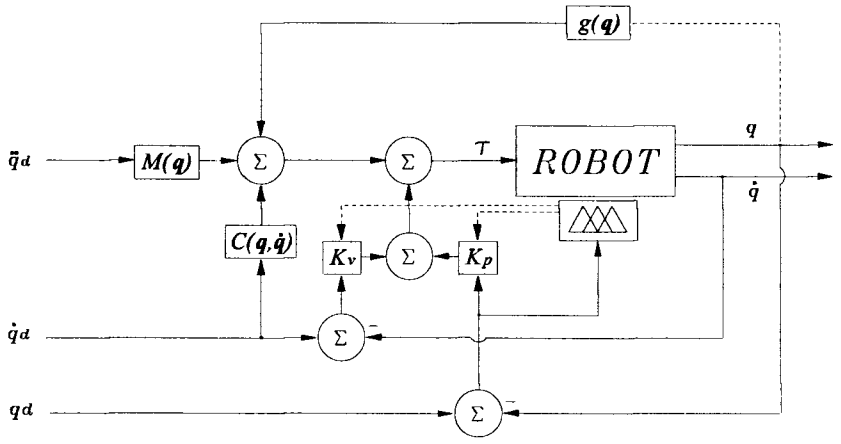


Figure 4.4: FST PD+ block diagram

4.2.2 Fuzzy-Sliding Mode Controller

The sliding mode control is a method derived from phase plane analysis. Phase plane analysis is a graphical method for studying dynamic systems, which was introduced in the late 1800's by mathematicians such as Henri Poincare. The idea of the method is to generate, in the state space of a second-order dynamic system, motion trajectories corresponding to various initial conditions and then to examine the qualitative features of the trajectories.

The phase plane is a graphical representation of the state space of dynamic systems. For the case of second-order systems, the phase plane is a two-dimensional plane with the process variable in the x axis and its derivative in the y axis. A further description and analysis of phase planes and sliding surfaces in non-linear systems can be found in [5].

Sliding Mode Control for Robot Manipulators

The vector containing the controlled variable (angular position) states for a single joint of a robot manipulator is defined by

$$\boldsymbol{q}_s = [q\ \dot{q}\ \cdots\ q^{(n-1)}]^T \tag{4.17}$$

and let $\tilde{q} = q_d - q$ be the tracking error in the variable q, thus

$$\tilde{\boldsymbol{q}}_s = \boldsymbol{q}_{ds} - \boldsymbol{q}_s = [\tilde{q}\ \dot{\tilde{q}}\ \cdots\ \tilde{q}^{(n-1)}]^T \tag{4.18}$$

represents the tracking error vector of the system states (controlled variable and its further derivatives). A time-varying surface $S(t)$ is defined in the state-space $\boldsymbol{R}^{(n)}$ by the scalar equation $s(\boldsymbol{q}_s; t) = 0$, where

$$s(\boldsymbol{q}_s;t) = -\left(\frac{d}{dt}+\lambda\right)^{n-1}\tilde{q} \tag{4.19}$$

and λ is a strictly positive constant, which is interpreted as a line in the phase plane of slope $-\lambda$ and containing the point $\boldsymbol{q}_{ds} = [q_d\ \dot{q}_d]^T$, as illustrated in figure 4.5.

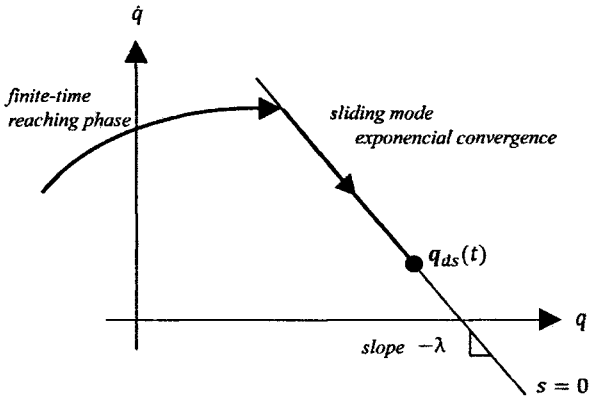


Figure 4.5: Sliding surface in a phase plane.

Notice that $\boldsymbol{q}_s$ refers to a vector of system states of a single variable, while $\boldsymbol{q}$ used in earlier sections is a vector of angular positions for all joints (in this case are two). If only two states are considered for control ($n = 2$) i.e. position and velocity of link 1 and 2, then the following expression for a 2 DOF manipulator is valid

$$\boldsymbol{s} = -\dot{\tilde{\boldsymbol{q}}} - \boldsymbol{\lambda}\tilde{\boldsymbol{q}} \tag{4.20}$$

where $\boldsymbol{s}$ is a vector of sliding surfaces for each joint; $\dot{\tilde{\boldsymbol{q}}}$ and $\tilde{\boldsymbol{q}}$ are 2×1 vectors containing the angular velocity and position errors respectively for joint 1 and 2; $\boldsymbol{\lambda}$ is a 2×2 diagonal matrix with constants defining slopes in the phase planes for joint 1 and 2.

Given initial conditions $\boldsymbol{q}_d(0) = \boldsymbol{q}(0)$, the problem of tracking $\boldsymbol{q} \equiv \boldsymbol{q}_d$ is equivalent to that of remaining in the surface $S(t)$ for all $t > 0$. When $s \equiv 0$, it represents a linear differential equation whose unique solution is $\tilde{\boldsymbol{q}} \equiv 0$. Therefore, the problem of tracking the $n-$dimensional vector $\boldsymbol{q}_d$ can be reduced to that of keeping the scalar quantity s at zero. The simplified $1^{st}-$order problem of keeping the scalar s at zero can now be achieved by choosing the control law such that outside $S(t)$ the following condition is satisfied

$$\frac{1}{2}\frac{d}{dt}s^2 \leq -\eta|s| \tag{4.21}$$

where η is a strictly positive constant. The last equation states that the "distance" to the surface (s^2) decreases along all system trajectories. Thus, all trajectories point

towards the surface $S(t)$, once on the surface the system trajectories remain on the surface. The system's behavior once on the surface is called *sliding regime* or *sliding mode*. It can be said that the sliding condition makes the surface an invariant set.

The necessary condition for the dynamics to be in sliding mode is

$$\dot{\boldsymbol{s}} = [0\ 0]^T \tag{4.22}$$

which helps satisfy in part equation 4.21. Equation 4.1 can be rewritten as follows

$$\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q})\boldsymbol{\tau} + M^{-1}(\boldsymbol{q})\left(-C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\right) \tag{4.23}$$

thus, the condition for the dynamics to be in sliding mode using equation 4.23 are given by

$$\begin{aligned} \dot{\boldsymbol{s}} &= -\ddot{\tilde{\boldsymbol{q}}} - \lambda\dot{\tilde{\boldsymbol{q}}} \\ &= \ddot{\boldsymbol{q}} - \ddot{\boldsymbol{q}}_d - \lambda\dot{\tilde{\boldsymbol{q}}} \\ &= M^{-1}(\boldsymbol{q})\boldsymbol{\tau} + M^{-1}(\boldsymbol{q})\left(-C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} - \boldsymbol{g}(\boldsymbol{q})\right) - \ddot{\boldsymbol{q}}_d - \lambda\dot{\tilde{\boldsymbol{q}}} \\ &= [0\ 0]^T \end{aligned} \tag{4.24}$$

Solving equation 4.24 for the control input, an expression for $\boldsymbol{\tau}$ called equivalent control $\boldsymbol{\tau}_{eq}$ is obtained:

$$\boldsymbol{\tau}_{eq} = M(\boldsymbol{q})(\ddot{\boldsymbol{q}}_d + \lambda\dot{\tilde{\boldsymbol{q}}}) + C(\boldsymbol{q},\dot{\boldsymbol{q}})\dot{\boldsymbol{q}} + \boldsymbol{g}(\boldsymbol{q}) \tag{4.25}$$

if the dynamics were exactly known, $\boldsymbol{\tau}_{eq}$ would maintain $\dot{\boldsymbol{s}} = [0\ 0]^T$. It can easily seen that control laws given by equations 4.14 and 4.25 are pretty similar.

In order to satisfy sliding condition 4.21 in the presence of uncertainty on the system model, a discontinuous term $\boldsymbol{\tau}'$ is added to $\boldsymbol{\tau}_{eq}$ across the surface $s = 0$:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{eq} - \boldsymbol{\tau}' \tag{4.26}$$

where $\boldsymbol{\tau}' = \boldsymbol{k}\ \mathrm{sign}(\boldsymbol{s})$. In the last equation, $\boldsymbol{k}$ is a 2×2 diagonal matrix with gains k_i for the controller of each link, term $\mathrm{sign}(\boldsymbol{s})$ returns a 2×1 vertical vector containing the sign of s_i for each link. The term $\mathrm{sign}(x)$ denotes the sign function in equation 3.45.

this is illustrated in figure 4.6. By choosing k_i to be large enough, equation 4.21 is satisfied. Equation 4.26 can be seen as an intuitive feedback control strategy that means "if error is negative, push hard enough in the positive direction".

Fuzzy-Sliding Mode Control for Robot Manipulators

When the sign function is used, chattering is commonly found when the system is close to the sliding surface, see figure 4.7. Chattering is undesirable in most applications

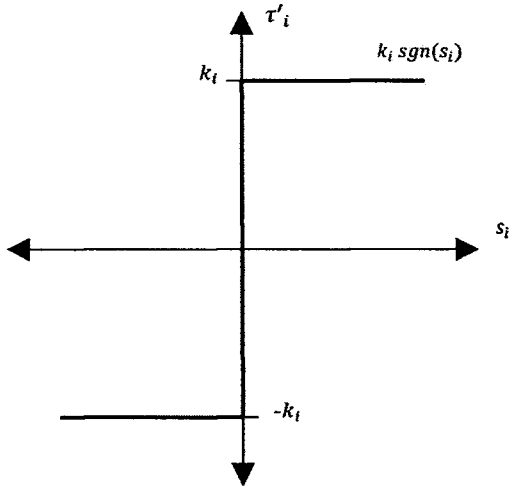


Figure 4.6: Discontinuous term of equation 4.26.

since it involves high control activity, it could excite high frequency components not considered in the system model and make it unstable in the worst case.

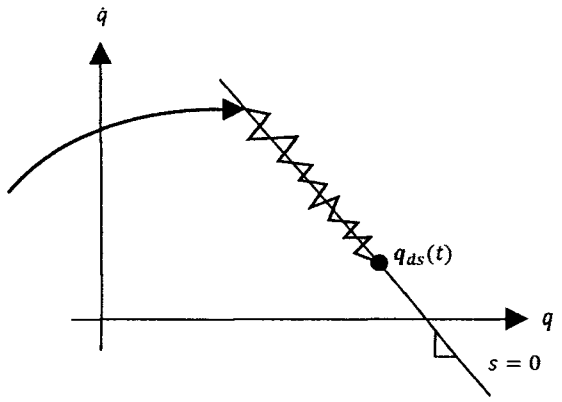


Figure 4.7: Presence of chattering as the result of control switching.

For this reason, a fuzzy system softens the output of the controller so the system remains within the sliding surface without drastic moves from the controller. This controller was introduced in 1992 in [12, 16] with the name of fuzzy-sliding mode controller. It was further studied in [28]. This controller is illustrated in figure 4.8. In [3] a FSM controller is used to control in simulation a 6-DOF robot manipulator.

The fuzzy system is composed by one input x and the corresponding output y, which can be seen as a static mapping H defined by equation 4.27, slightly different from that of the FLT.

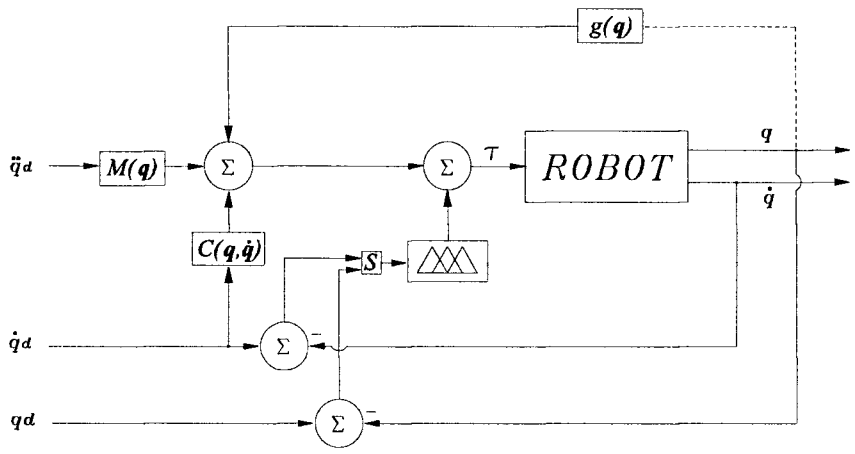


Figure 4.8: FSM block diagram

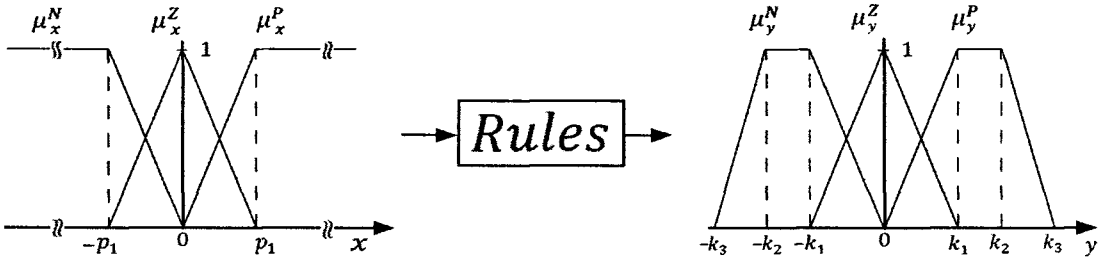


Figure 4.9: FSM controller parameters.

$$\begin{array}{rccc} H: & \mathbb{R}_+ & \rightarrow & \mathbb{R} \\ & x & \mapsto & y\,. \end{array} \tag{4.27}$$

The universes of discourse of x and y are partitioned into three fuzzy sets: N (*Negative*), Z (*Zero*) and P (*Positive*) each described by a MF. Trapezoidal and triangular MFs are used for input and output variables, this is illustrated in figure 4.9. The corresponding *Negative*, *Zero* and *Positive* MFs for the input variable x are denoted by

$$\boldsymbol{\mu}_x(x;\boldsymbol{p}) = \begin{bmatrix} \mu_x^N(x;-p_1) \\ \mu_x^Z(x;-p_1,p_1) \\ \mu_x^P(x;p_1) \end{bmatrix} \tag{4.28}$$

For the output variable y, the corresponding singleton MFs to *Negative*, *Zero* and *Positive* are represented by

$$\boldsymbol{\mu}_y(\cdot;\boldsymbol{k}) = \begin{bmatrix} \mu_y^N(\cdot;-k_1,-k_2,-k_3) \\ \mu_y^Z(\cdot;-k_1,k_1) \\ \mu_y^P(\cdot;k_1,k_2,k_3) \end{bmatrix} \tag{4.29}$$

The fuzzy inference method is the Mamdani model and the defuzzification method is the center of mass. For the fuzzy system just described, the inference rules are designed such that if the control variable is above the sliding surface, the control signal is negative; if it is below the sliding surface, the control signal is positive. Rules in the fuzzy system are defined as follows:

If $\mu_x^N(x;-p_1)$	Then	$\mu_y^N(\cdot;-k_1,-k_2,-k_3)$
If $\mu_x^Z(x;-p_1,p_1)$	Then	$\mu_y^Z(\cdot;-k_1,k_1)$
If $\mu_x^P(x;p_1)$	Then	$\mu_y^P(\cdot;k_1,k_2,k_3)$

Term $\boldsymbol{\tau}'$ in equation 4.26 is now given by $K_{Fuzz}(\boldsymbol{s})$. It is a vertical vector of two elements, the output of the fuzzy systems for both links. Input for the fuzzy system in this case is the value s_i for a single link and the output is the discontinuous term τ_i' of the torque applied by the controller. The FSM control law can be expressed as follows:

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{eq} - K_{Fuzz}(\boldsymbol{s}) \tag{4.30}$$

which is an extension to the sliding mode with boundary layer [4]. The function $K_{Fuzz_i}(s_i)$ with MFs defined by 4.28 and 4.29 may have a shape like figure 4.10. Boundary values for τ_i' are given as an effect of the trapezoidal output MFs.

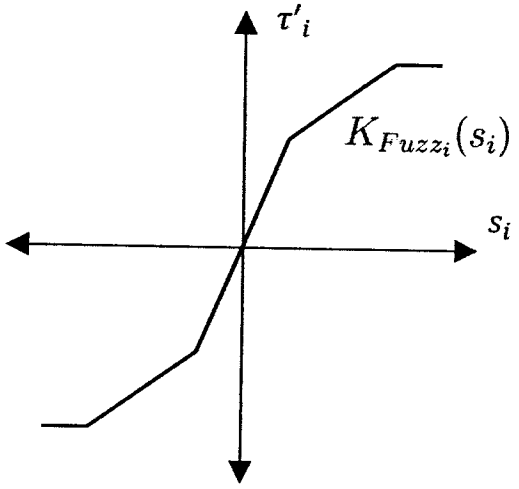


Figure 4.10: Possible shape of function $K_{Fuzz_i}(s_i)$.

Chapter 5

Simulation Results

This section presents the results of the simulation for the controllers that are being compared. The equations of the robot's dynamics were modeled in MATLAB-Simulink and the Fuzzy Systems defined were simulated using the MATLAB-Fuzzy Logic Toolbox.

As established before, two position controllers and two velocity controllers are compared for similar tasks. In order to make a comparison as fair as possible for the controllers of each kind, a methodology for the optimization and testing process is proposed. This process is described as follows:

1. **Optimization.** In this step the parameters for the controller of each link are optimized with the GA. As the intention is to use the controller in all the motion range, it will be optimized to work in the regions known to be hard to handle. This is when any link is in the positions 90° and 270° respect to the negative y axis. The controller of each link is optimized separately.

   The GA will try to maximize the negative of the Integral Absolute Error (IAE) criterion for all controllers. The fitness function for all controllers is thus given by the following expression:

   $$\text{Fitness} = -\text{IAE} \tag{5.1}$$

   Maximizing the negative of the IAE means try to make it as closer to zero as possible. For position controllers, the error for the IAE calculation is the difference of the response of a model reference with respect to the response of the link. The error for the IAE calculation for the optimization of velocity controllers is the difference of the desired position trajectory with respect to the position of the link.

   Figure 5.1 shows a block diagram illustrating the fitness evaluation for controllers. The chromosome of each individual is first decodified, then a simulation of the respective link of the robot arm with the controller defined by the decodified

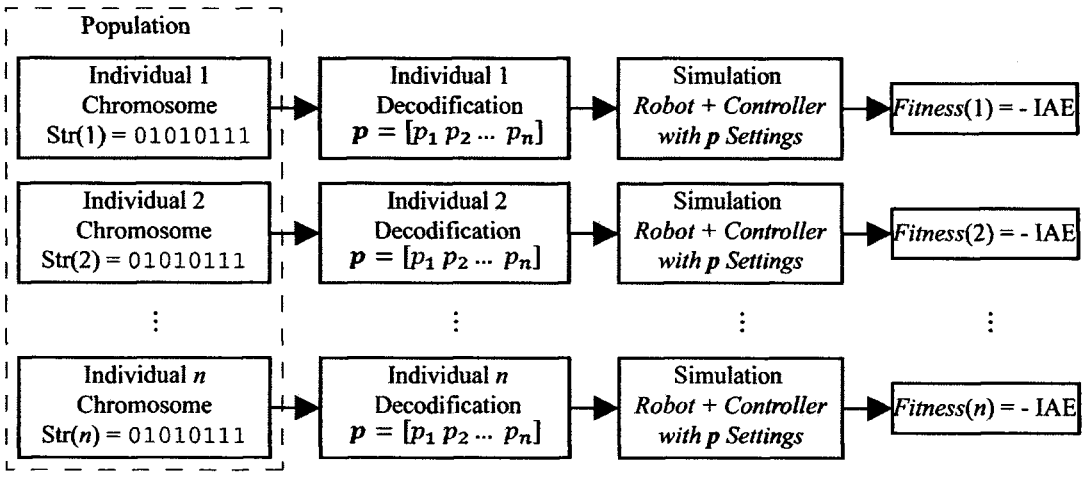


Figure 5.1: Fitness assignment for optimization of controllers.

parameters is done and finally performance of the system is measured. The fitness is assigned for each individual in the population.

2. **Performance tests.** Once optimized, both controllers are tested in normal operation conditions, i.e. no parameter variation and no perturbations. The references given for both links are along all the motion range and both links are moving at the same time. This is the main test for both controllers because following the references given in this step is the least expected performance.

3. **Robustness tests.** In this last step, the robustness of each controller is evaluated by modifying the operation conditions for which the controller was optimized. Two kind of tests are performed, one is varying of the robot's internal parameters and the other is by giving the system an external perturbation.

For all controllers, the final value of parameters optimized is reported in Appendix A. A test to illustrate the sliding surface in a phase plane for the FSM controller and the PID controller with unsaturated torque signal can also be found in Appendix A.

5.1 Position Controllers

This section presents the simulation results for the PID and FST PID controllers. The performance of both controllers is compared one against the other, so the methodology described above is followed. Figure 5.2 shows a block diagram of the steps to compare the two position controllers considered.

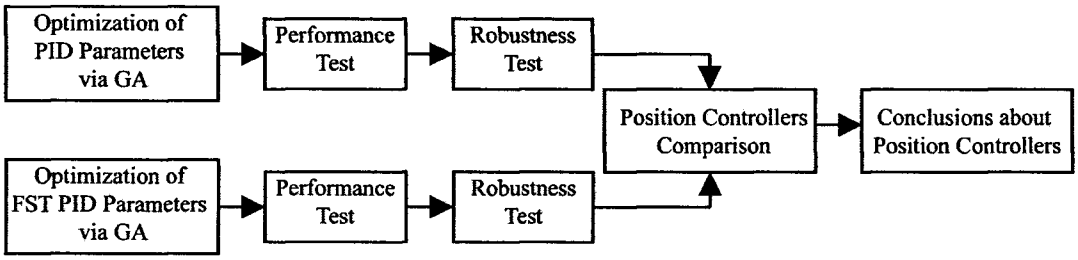


Figure 5.2: Methodology applied to compare Position controllers.

5.1.1 Model Reference for Position Controllers

One thing that can be noted from the robot's dynamic model is that it can be approximated to a second-order system in small regions of the motion range. This means that small moves tend to behave as a linear second-order system, but in the large, the system is non-linear. So a linearization the system and design of a controller for a linear system is not recommended. Instead, the controller can be adjusted to achieve a linear behavior of the system for all the motion range.

Based in this idea, the controller is optimized by the GA so the system behaves the most closely to a second-order linear system. The closed loop transfer function of a second-order system is given by equation 5.2. For a deeper description of second-order linear systems refer to [27].

$$\frac{C(s)}{R(s)} = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{5.2}$$

Another desired behavior of the system is to have very little or no overshoot, so the second-order reference system is designed to be *critically damped*, this means a damping ratio of $\zeta = 1$. The undamped natural frequency is arbitrarily chosen to be $w_n = 5$ for link 1 and $w_n = 7$ for link 2.

When the GA optimizes position controllers, the system performance is evaluated using the IAE with respect to the system model as follows:

$$\text{IAE} = \int_0^t |c(t) - q_i(t)|dt \tag{5.3}$$

this is different to the error defined previously for the controller of each link, which remains as $\tilde{q}_i = q_{d_i} - q_i$.

The goal of the optimization of position controllers is that the system shows a transient response close to the response of a second-order linear system. If the error for the IAE calculation were evaluated with respect to the position reference, the transient response may be faster and more accurate but also more likely to have overshoot or

Table 5.1: Parameters of the GA used to optimize the PID controllers.

Population size	40
Mutation probability	2%
Number of generations	50
Crossover probability	100%
Chromosome length	24 bits

Table 5.2: Parameters adjusted by the GA for the PID controller of each link.

Parameter	String length
K_p	8 bit
K_v	8 bit
K_i	8 bit

oscillation when approaching the reference. Overshoot and oscillation are not desired for this application, therefore must be minimized. To avoid torque saturation is not one of the goals of the optimization.

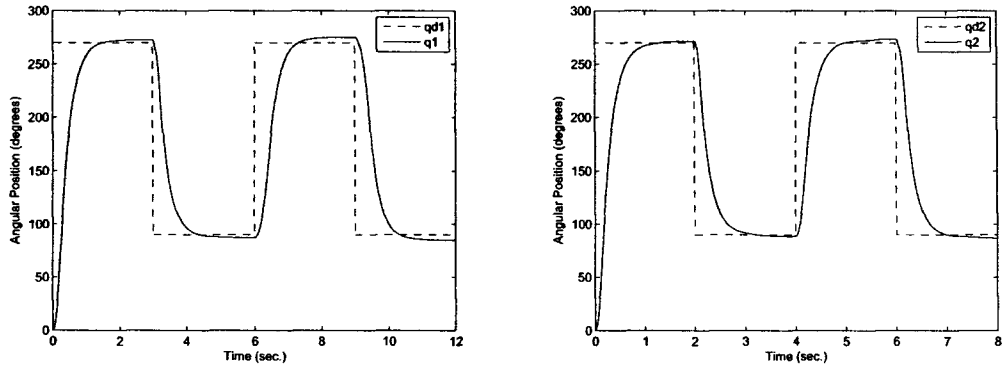
5.1.2 Simulation Results for the PID Controller

Optimization of Parameters for the PID Controller

As described above, the first step in the methodology for comparing position controllers is the optimization of the controller's parameters. For this purpose a GA with the parameters shown in table 5.1 is used to optimize controller for link 1 and link 2. Table 5.2 reports the parameters modified by the GA and its string length for the PID controller. For each PID controller the GA adjusted the K_p, K_v and K_i gains, which for simplicity were defined by the same string-length. The decodification for these parameters was done using equation 2.14.

The references for which the controller is optimized are intended to be so that the controller handles the most non-linear regions of the motion range, this references are given by equations 5.4 and 5.5.

$$q_{d1}(t) = \begin{cases} 270° & \text{if} \quad 0 \leq t < 3 \; sec. \\ 90° & \text{if} \quad 3 \leq t < 6 \; sec. \\ 270° & \text{if} \quad 6 \leq t < 9 \; sec. \\ 90° & \text{if} \quad t > 9 \; sec. \end{cases} \tag{5.4}$$



a) b)

Figure 5.3: PID with optimized parameters. Response of: a) link 1; b) link 2.

$$q_{d2}(t) = \begin{cases} 270° & \text{if} \quad 0 \leq t < 2 \; sec. \\ 90° & \text{if} \quad 2 \leq t < 4 \; sec. \\ 270° & \text{if} \quad 4 \leq t < 6 \; sec. \\ 90° & \text{if} \quad t > 6 \; sec. \end{cases} \tag{5.5}$$

Also, to ensure the controller can also handle parametric uncertainties, the mass of link 2 (m_2) changes from 100% to 300% in $t = 6$ *sec.* while optimizing link 1 and in $t = 4$ *sec.* while optimizing link 2. Controllers for each link are optimized separately, this means that each link's controller is optimized while the other link remains static but coupled.

Performance of the controller for each link achieved after optimization is shown in figure 5.3. Both links show a transient response close to the model reference but have steady-state error and overshoot.

Performance tests for the PID Controller

The next step after the optimization of the controller is to test it in normal operation conditions for a wider operation range. This is intended to see if now that the controller has been optimized to handle hard regions, it generalizes its behavior to other points in space. One important detail is that in normal operation conditions both links will probably move coupled at the same time, so these tests are performed giving references for both links simultaneously. As commented before, no external perturbation or parameter variation is present in these tests.

The simulation results for this step are shown in figure 5.4. The reason to make link 2 follow the same references twice is because it can accomplish following the references in less time than link 1 and still needs to be moving while link 1 is moving. References for this test are given by equations 5.6 and equations 5.7 for link 1 and link 2 respectively.

$$q_{d1}(t) = \begin{cases} 90° & \text{if} \quad 0 \leq t < 4 \; sec. \\ 180° & \text{if} \quad 4 \leq t < 8 \; sec. \\ 360° & \text{if} \quad 8 \leq t < 12 \; sec. \\ 180° & \text{if} \quad 12 \leq t < 16 \; sec. \\ 90° & \text{if} \quad 16 \leq t < 20 \; sec. \\ 0° & \text{if} \quad t > 20 \; sec. \end{cases} \tag{5.6}$$

$$q_{d2}(t) = \begin{cases} 90° & \text{if} \quad 0 \leq t < 2 \; sec. \\ 180° & \text{if} \quad 2 \leq t < 4 \; sec. \\ 360° & \text{if} \quad 4 \leq t < 6 \; sec. \\ 180° & \text{if} \quad 6 \leq t < 8 \; sec. \\ 90° & \text{if} \quad 8 \leq t < 10 \; sec. \\ 0° & \text{if} \quad 10 \leq t < 12 \; sec. \\ 90° & \text{if} \quad 12 \leq t < 14 \; sec. \\ 180° & \text{if} \quad 14 \leq t < 16 \; sec. \\ 360° & \text{if} \quad 16 \leq t < 18 \; sec. \\ 180° & \text{if} \quad 18 \leq t < 20 \; sec. \\ 90° & \text{if} \quad 20 \leq t < 22 \; sec. \\ 0° & \text{if} \quad t >= 22 \; sec. \end{cases} \tag{5.7}$$

From these results it can be noted that link 1 and link 2 approach to the given references but accuracy is not the best. Steady state error (e_{ss}) is 2.9° for link 1 and 1.6° for link 2. A good accuracy would be $e_{ss} \leq 1°$, but for both links this was not possible. This inaccuracy is mainly because the GA did not optimize the controllers to be accurate, but to have an optimal behavior considering references in all the motion range.

The torque plots show that controller for link 1 makes soft moves and its torque signal is saturated in few occasions. In the other case, controller for link 2 saturates the actuator more times and makes more drastic moves than link 1. Appendix A shows the torque signal if the actuators were not saturated in simulation.

Robustness tests for the PID Controller

The robustness tests are performed to see if the controller is able to handle different conditions that can be present while performing some task. Simulation results for the PID controllers performing these tests are presented in figure 5.5.

The first robustness test is an external perturbation as shown in figure 5.5(a,b). An external perturbation of 20° @ $t = 2$ *sec.* is being applied to both links. The references are $q_{d1} = 90°$ and $q_{d2} = 0°$ all the time for link 1 and link 2 respectively. Both links start the simulation in the same positions of its references and move coupled at the same time.

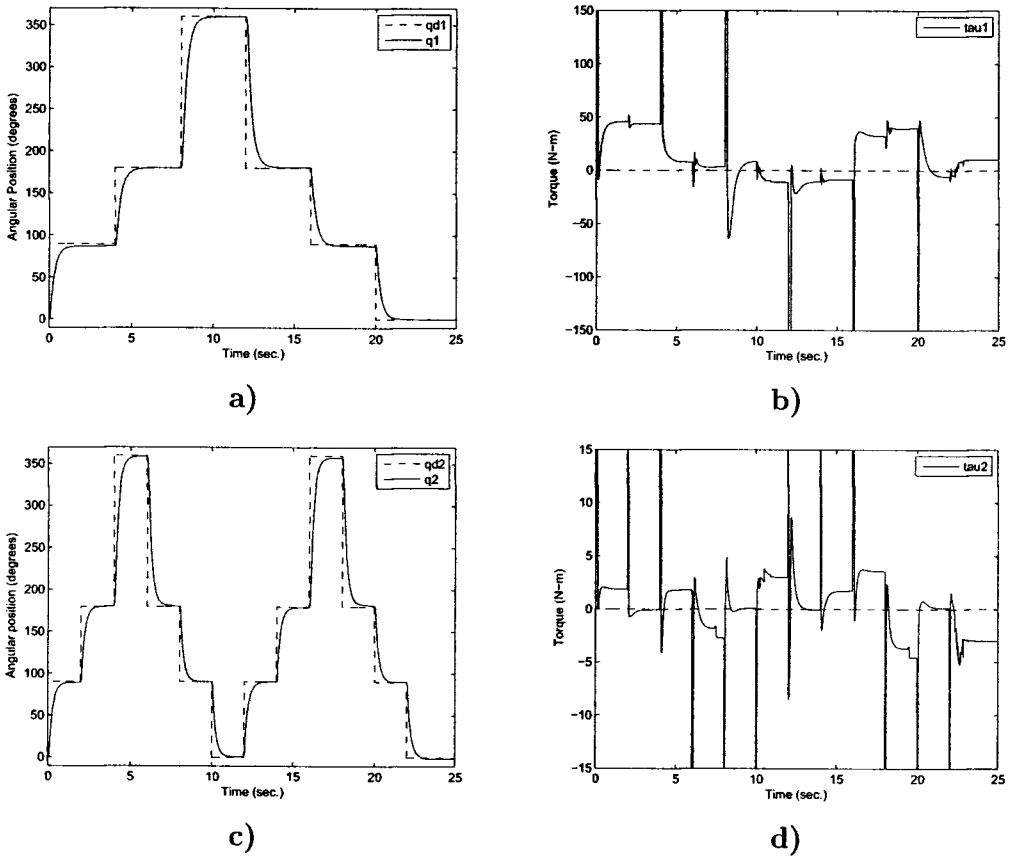


Figure 5.4: PID performance test. Response of: a) link 1; c) link 2. Torque for b) link 1; d) link 2.

The second kind of robustness test is a parameter variation, this is accomplished by changing link's 2 mass m_2. For this test, m_2 can be described as a function of time given by equation 5.8. The references are given the same for both links and defined generically by equation 5.9. The same function is applied for testing controller of link 1 and link 2. Different from the perturbation test, now each link moves while the other one remains static. Figure 5.5(c,d) show performance of controllers for link 1 and link 2 respectively for this test.

$$m_2(t) = \begin{cases} 3.88\ kg & \text{if} \quad 0 \leq t < 8\ sec. \\ 7.76\ kg & \text{if} \quad 8 \leq t < 16\ sec. \\ 11.64\ kg & \text{if} \quad t > 16\ sec. \end{cases} \tag{5.8}$$

$$q_d(t) = \begin{cases} 90° & \text{if} \quad 0 \leq t < 2\ sec. \\ 180° & \text{if} \quad 2 \leq t < 4\ sec. \\ 90° & \text{if} \quad 4 \leq t < 6\ sec. \\ 0° & \text{if} \quad 6 \leq t < 8\ sec. \\ 90° & \text{if} \quad 8 \leq t < 10\ sec. \\ 180° & \text{if} \quad 10 \leq t < 12\ sec. \\ 90° & \text{if} \quad 12 \leq t < 14\ sec. \\ 0° & \text{if} \quad 14 \leq t < 16\ sec. \\ 90° & \text{if} \quad 16 \leq t < 18\ sec. \\ 180° & \text{if} \quad 18 \leq t < 20\ sec. \\ 90° & \text{if} \quad 20 \leq t < 22\ sec. \\ 0° & \text{if} \quad t >= 22\ sec. \end{cases} \tag{5.9}$$

In the perturbation test, performance of this controller for link 1 is poor, given by $e_{ss} = 2.6°$ (being $e_{ss} \leq 1°$ a good accuracy range). Controller for link 2 is much better, having an accuracy within the range $e_{ss} \leq 1°$. Both links separate from its reference at the start of the simulation because position controllers don't compensate the control signal automatically with the gravity matrix as the velocity controllers do.

For the parameter variation test, performance for the controller of link 1 is poor near the 90° region, in the worst case $e_{ss} = 5.2°$. As m_2 increases, the accuracy of controllers for both links decrease heavily. Link 2 is less affected than link 1 when m_2 changes, the worst case was $e_{ss} = 3.3°$. From these tests, it can be observed that controller for link 1 doesn't show good robustness, controller for link 2 show better performance for the perturbation test but not so good in the parameter variation test.

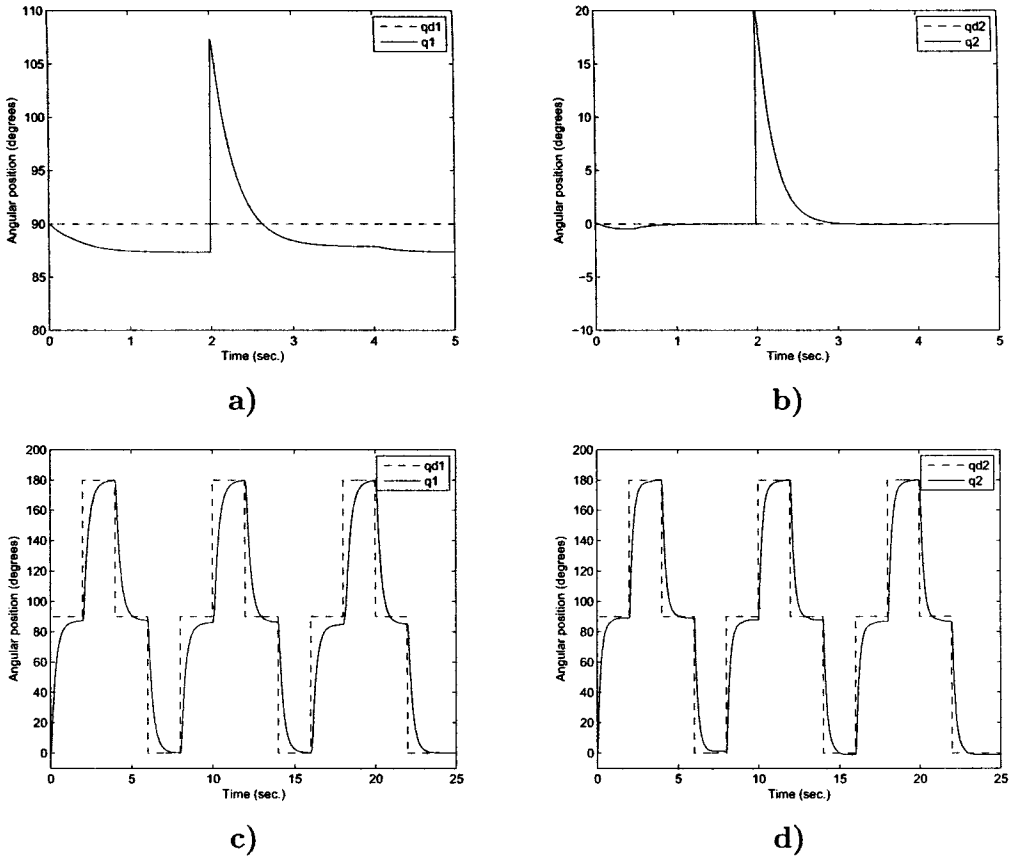


Figure 5.5: PID robustness test. External perturbation for: a) link 1; b) link 2. Parametric variations for: c) link 1; d) link 2.

Table 5.3: Parameters that define a FLT.

	Parameter	String length
Input MF	p_1	5 bit
	p_2	5 bit
	p_3	5 bit
	p_4	5 bit
Rules	*RuleOrder*	1 bit
Output MF	k_1	5 bit
	k_2	5 bit
	k_3	8 bit

5.1.3 Simulation Results for the FST PID Controller

Optimization of Parameters for the FST PID Controller

Following the methodology proposed, the FST PID controller is optimized with a GA. Table 5.3 contains a list of parameters that define a FLT which are modified by the GA. The input MFs are defined by $\boldsymbol{p} = \{p_1, p_2, p_3, p_4\}$ and the output singleton MFs by $\boldsymbol{k} = \{k_1, k_2, k_3\}$. These parameters are coded each in a substring and concatenated into a string to be handled by the GA.

In order to decode the parameters from the string, first p_1 is obtained using equation 2.14 and the next three parameters are decoded and adjusted as

$$p_n = p_{n-1} + d_{p_n} \tag{5.10}$$

where d_{p_n} is the decodified value of each parameter and $n = 2, 3, 4$. This results in $p_1 < p_2 < p_3 < p_4$, which is a necessary condition because the MFs must cover the entire universe of discourse.

For the output singleton MFs, it is needed to obtain gains that satisfy the restriction $k_3 > k_2 > k_1$. Therefore the decodification and adjustment for these parameters is

$$\begin{aligned} k_3 &= d_{k_3} \\ k_2 &= \tfrac{d_{k_2}}{100} k_3 \\ k_1 &= \tfrac{d_{k_1}}{100} k_2 \end{aligned} \tag{5.11}$$

where $d_{k_2}, d_{k_1} \in [0, 100]$ are the decodified values using equation 2.14.

The *RuleOrder* parameter, refers to the order of the rules, which can be selected as shown in table 5.4. In [22] the same FLT is optimized by a GA, but rules were fixed

Table 5.4: Rules selected from *RuleOrder*.

RuleOrder	Rules Selected		
0	If $\mu^S(\|x\|;p_1,p_2)$	Then	$\mu_y^S(\cdot;k_1)$
	If $\mu^M(\|x\|;p_1,p_2,p_3,p_4)$	Then	$\mu_y^M(\cdot;k_2)$
	If $\mu^B(\|x\|;p_3,p_4)$	Then	$\mu_y^B(\cdot;k_3)$
1	If $\mu^S(\|x\|;p_1,p_2)$	Then	$\mu_y^B(\cdot;k_3)$
	If $\mu^M(\|x\|;p_1,p_2,p_3,p_4)$	Then	$\mu_y^M(\cdot;k_2)$
	If $\mu^B(\|x\|;p_3,p_4)$	Then	$\mu_y^S(\cdot;k_1)$

Table 5.5: Parameters of the GA used to optimize the FST PID controllers.

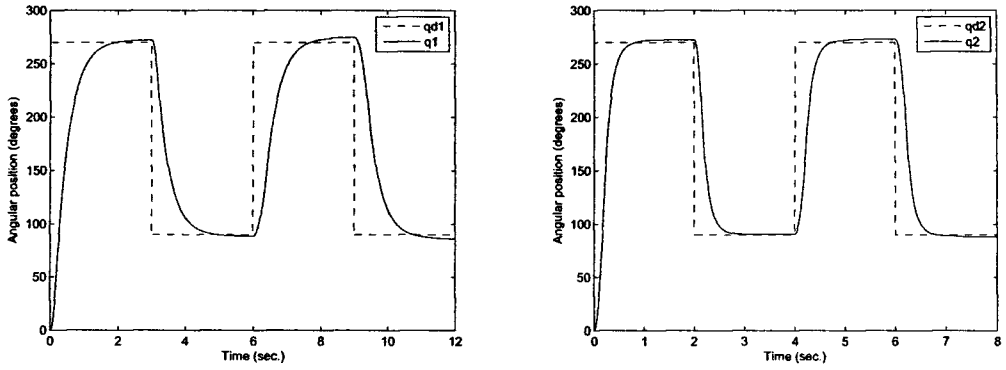
Population size	80
Mutation probability	2%
Number of generations	50
Crossover probability	100%
Chromosome length	117 bits

following the next reasoning. For a big position error we need to apply a small k_p in order to avoid torque saturation. The first rule specifies that for a small position error we should apply a big k_p in order to reduce this error. For the derivative gain k_v, a similar criterion is used taking into account that for big position errors it is suitable to have small damping, to avoid an oscillatory response. In the case of integral gains a inverse criterion is used taking into account that for big $|\tilde{q}|$ it is suitable to have big k_i to reduce the position error. As the intention is not to limit optimization, rules were also free to be modified by the GA.

So, for the Controller of a single link there are 8 parameters for each of the three FLT. The resulting string that codifies parameters for the three FLTs of each joint has $39 \times 3 = 117$ bits.

Parameters of this GA are slightly different from that used for the PID, see table 5.5. A larger population size is used because there are more parameters to be adjusted, so more solutions (individuals in the population) must be processed in parallel to find the optimal. More parameters to be adjusted mean an increase in the string length handled by the GA and so in the complexity of the problem.

For the optimization of the FST PID, the same conditions are used as for the PID, i.e. references and changes in m_2. The GA evaluates the performance of each solution using the IAE of the system with respect to the model reference. Performance of this controller after optimization is shown in figure 5.6.



a) b)

Figure 5.6: FST PID with optimized parameters. Response of: a) link 1; b) link 2.

Performance tests for the FST PID Controller

As done for the PID, the FST PID is now tested in normal operation conditions with no external perturbations or parameters change. The same references are given for this controller as shown in figure 5.7. Now link 1 reached the specified references with much more accuracy ($e_{ss} = 1.6°$) than the PID controller ($e_{ss} = 2.9°$), but still is out of the range $e_{ss} \leq 1°$ of a considered good accuracy. Controller for link 1 show better performance criterions than the PID. Performance for link 2 is good but no better than when using the PID controller for all criterions in this test. For both links the torque plots show more control signal activity but actuators are saturated almost the same times.

Robustness tests for the FST PID Controller

After the performance tests, robustness tests are performed for the FST PID. The same test done for the PID is performed for the FST PID, figure 5.8 show the simulation results for this controller.

Performance in the presence of an external perturbation was almost the same for link 1 than when using the PID (see table 5.6). For link 2 performance is worse than when the PID was tested, having an increase of the e_{ss} from 0° to 1.1°. This test shows that the PID is much better in this case for link 2. When varying m_2, link 1 and 2 show better evaluation than the PID in all performance criterions.

5.1.4 Position Controllers Comparison

This section summarizes the results obtained for tests performed on the position controllers. From every test four different criterions are obtained to make a comparison

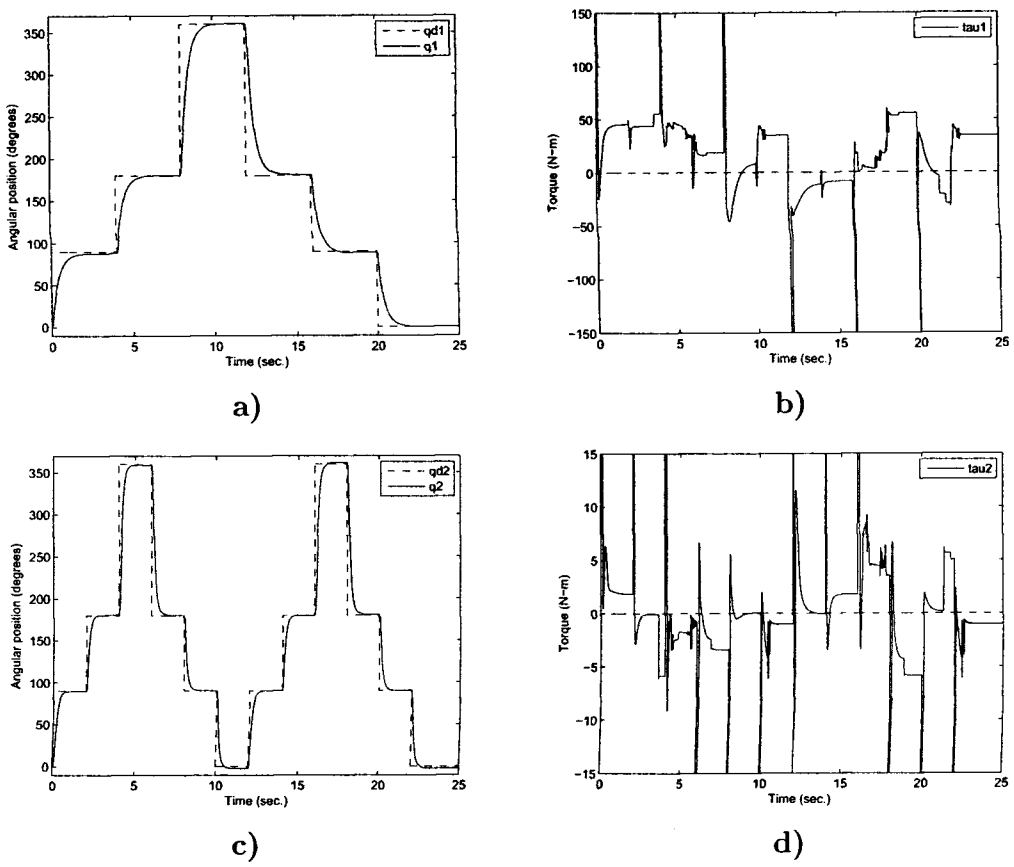


Figure 5.7: FST PID performance test. Response of: a) link 1; c) link 2. Torque for b) link 1; d) link 2.

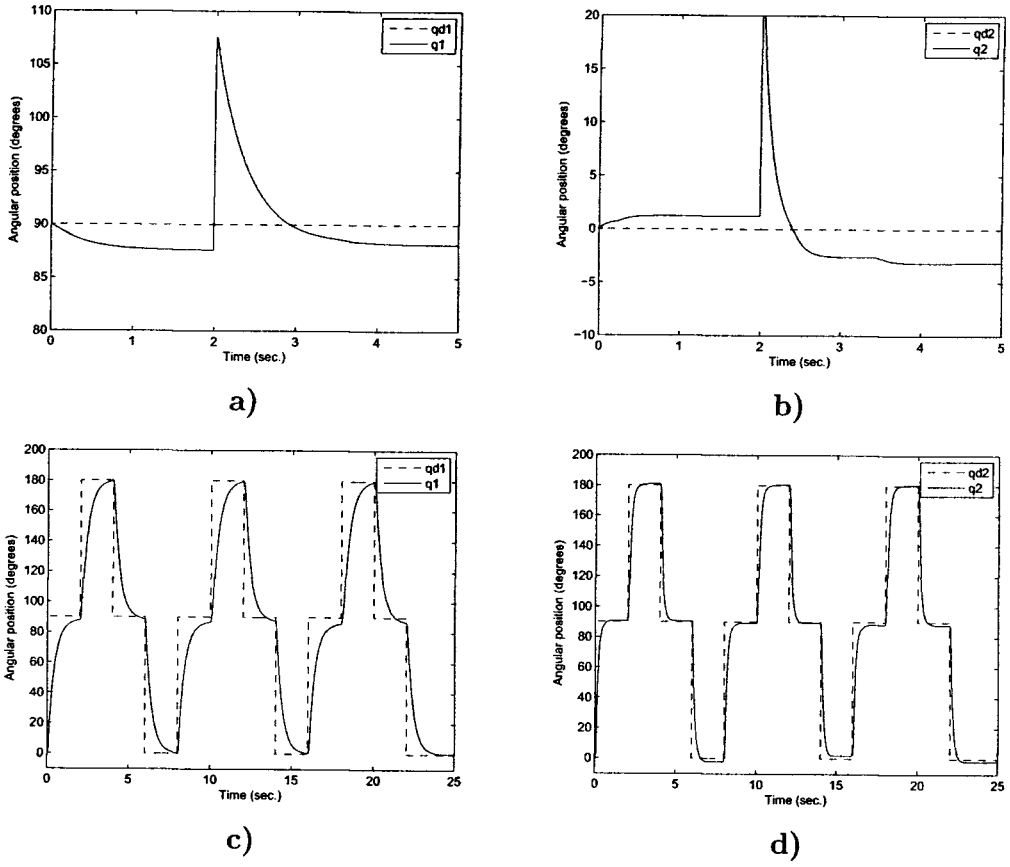


Figure 5.8: FST PID robustness test. External perturbation for: a) link 1; b) link 2. Parametric variations for: c) link 1; d) link 2.

Table 5.6: Performance Comparison of Position Controllers

		PID		FST PID		
		Link 1	Link 2	Link 1	Link 2	Figures
Optimization	IAE	393.5	239.9	376.7	257.5	5.3,5.6
Normal Operation	IAE	270.2	372.4	260.0	403.4	5.4,5.7
	Overshoot (deg.)	2.4	1.2	1.6	3.2	
	e_{ss} (deg.)	2.9	2.1	1.6	3.1	
External Perturbation	t_s @ 0.5 deg. (sec.)	3.0	0.7	3.0	3.0	5.5,5.8
	e_{ss} (deg.)	2.6	0.0	2.4	1.1	
Parameter Variation	IAE	400.5	281.3	395.2	296.9	
	Overshoot (deg.)	4.9	3.3	2.8	2.0	
	e_{ss} (deg.)	5.2	3.3	4.0	2.1	

between the controllers. For the e_{ss} units are degrees, unlike commonly expressed (as a percentage), because changes in references are not the same every time. Therefore the maximum e_{ss} along the simulation time is reported instead. Similar to e_{ss}, the Settling time (t_s) is taken when the link remains within the 0.5° region close to the reference (commonly taken when the link remains within 2% of its reference). This last parameter is only used for the robustness test. Table 5.6 reports values for the evaluation criterions.

Based in the table, for link 1 the FST PID controller is superior to the PID in every test. For link 2 the performance is better for the FST PID than the PID only in the parameter variation test. This suggests that a FST PID controller is better for link 1, while for link 2 is the PID.

5.2 Velocity Controllers

Similar to the Position Controllers, a comparison of velocity controllers is presented in this section. Now a FST PD+ controller and a FSM controller will be compared one against the other following the methodology proposed. As for Position controllers, the methodology applied for Velocity controllers is illustrated in figure 5.9.

As commented before, in the optimization step the GA will try to set the controller's parameters so that the negative of the IAE is maximized. The error for the IAE calculation in the optimization of velocity controllers, is given by the difference between the desired position trajectory and the angular position of each link as follows:

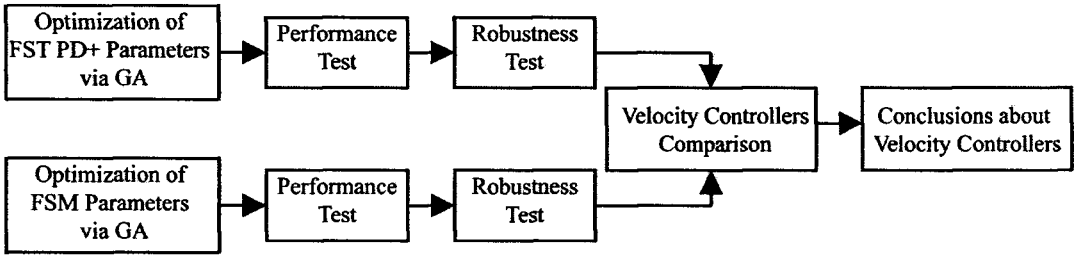


Figure 5.9: Methodology applied to compare Velocity controllers.

Table 5.7: Parameters of the GA used to optimize the FST PD+ controllers.

Population size	50
Mutation probability	2%
Number of generations	50
Crossover probability	100%
Chromosome length	78 bits

$$\text{IAE} = \int_0^t |q_{d_i}(t) - q_i(t)|dt \tag{5.12}$$

5.2.1 Simulation Results for the FST PD+ Controller

Optimization of Parameters for the FST PD+ Controller

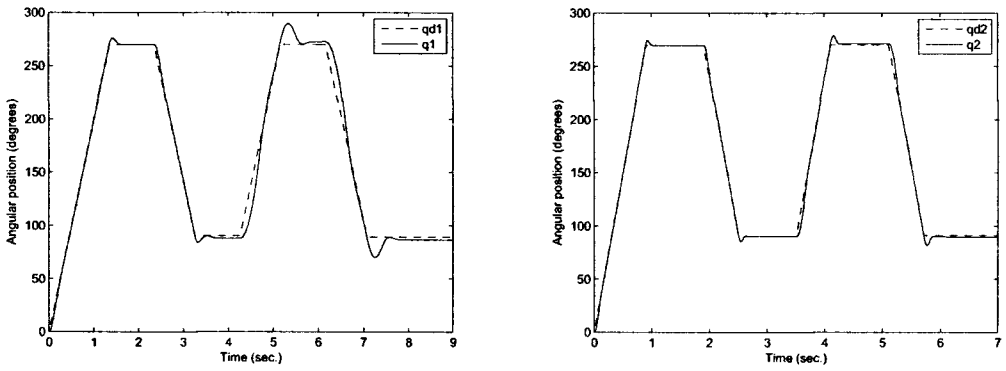
As for the PID and the FST PID, an optimization of the FST PD+ controller is done in the first step. This controller is similar to the FST PID in that it uses the same FLT structure, (see fig. 4.3, table 5.3 and 5.4) but only two are needed to adjust each of the PD gains (k_p and k_v) in real time. Table 5.7 lists the parameters of the GA used to optimize this controller.

References are also similar as those used for the optimization of position controllers (90° and 270°), but instead of a drastic reference change, a ramp is used to move the references from the actual point to the next. Ramps with velocity of $200°/sec.$ and $300°/sec.$ were used for the references of link 1 and link 2 respectively. Mass of link 2 m_2 changes from 100% to 300% in $t = 3.5sec.$ when optimizing both links. References for both links are listed in table 5.8.

Response of the system using this controller after optimization is shown in figure 5.10. Both links present overshoot no matter if m_2 is increased and e_{ss} is less compared to the results of the optimization of position controllers.

Table 5.8: References for velocity controllers in the optimization process.

Link 1 References	1. Ramp to 270° @ 200°/*sec.* 2. 270° for 1*sec.* 3. Ramp to 90° @ 200°/*sec.* 4. 90° for 1*sec.* 5. Ramp to 270° @ 200°/*sec.* 6. 270° for 1*sec.* 7. Ramp to 90° @ 200°/*sec.* 8. 90° for 3*sec.*
Link 2 References	1. Ramp to 270° @ 300°/*sec.* 2. 270° for 1*sec.* 3. Ramp to 90° @ 300°/*sec.* 4. 90° for 1*sec.* 5. Ramp to 270° @ 300°/*sec.* 6. 270° for 1*sec.* 7. Ramp to 90° @ 300°/*sec.* 8. 90° for 3*sec.*



a) b)

Figure 5.10: FST PD+ with optimized parameters. Response of: a) link 1; b) link 2.

Table 5.9: References for velocity controllers in performance tests.

Link 1 References	1. Ramp to 90° @ 50°/*sec.* 2. 90° for 1*sec.* 3. Ramp to 180° @ 50°/*sec.* 4. 180° for 1*sec.* 5. Ramp to 360° @ 50°/*sec.* 6. 360° for 1*sec.* 7. Ramp to 180° @ 50°/*sec.* 8. 180° for 1*sec.* 9. Ramp to 90° @ 50°/*sec.* 10. 90° for 1*sec.* 11. Ramp to 0° @ 50°/*sec.* 12. 0° for 1*sec.*
Link 2 References	1. Ramp to 90° @ 100°/*sec.* 2. 90° for 0.5*sec.* 3. Ramp to 180° @ 100°/*sec.* 4. 180° for 0.5*sec.* 5. Ramp to 360° @ 100°/*sec.* 6. 360° for 0.5*sec.* 7. Ramp to 180° @ 100°/*sec.* 8. 180° for 0.5*sec.* 9. Ramp to 90° @ 100°/*sec.* 10. 90° for 0.5*sec.* 11. Ramp to 0° @ 100°/*sec.* 12. 0° for 0.5*sec.*

Performance tests for the FST PD+ Controller

As for the position controllers, this controller is tested in normal operation conditions for references different from which it was optimized. Links also move coupled at the same time with no external perturbation or parameter variation. Ramps for references in this test are slower than for the optimization (50°/*sec.* for link 1 and 100°/*sec.* for link 2), the intention is that the conditions of the optimization are harder to handle. Table 5.9 details the references for link 1 and link 2, the former executes the cycle only once and the latter does it twice. The reason for which link 2 follows the references twice is the same as for the position controllers.

Figure 5.11 shows the results for this test, both links follow accurately the references, with $e_{ss} = 0.0°$ for link 1 and $e_{ss} = 0.1°$ for link 2, which is a good accuracy ($e_{ss} \leq 1°$). Overshoot in the worst case was 0.6° for link 1 and 1.1° for link 2. Torque plots show drastic moves and the actuators are saturated in several ocassions.

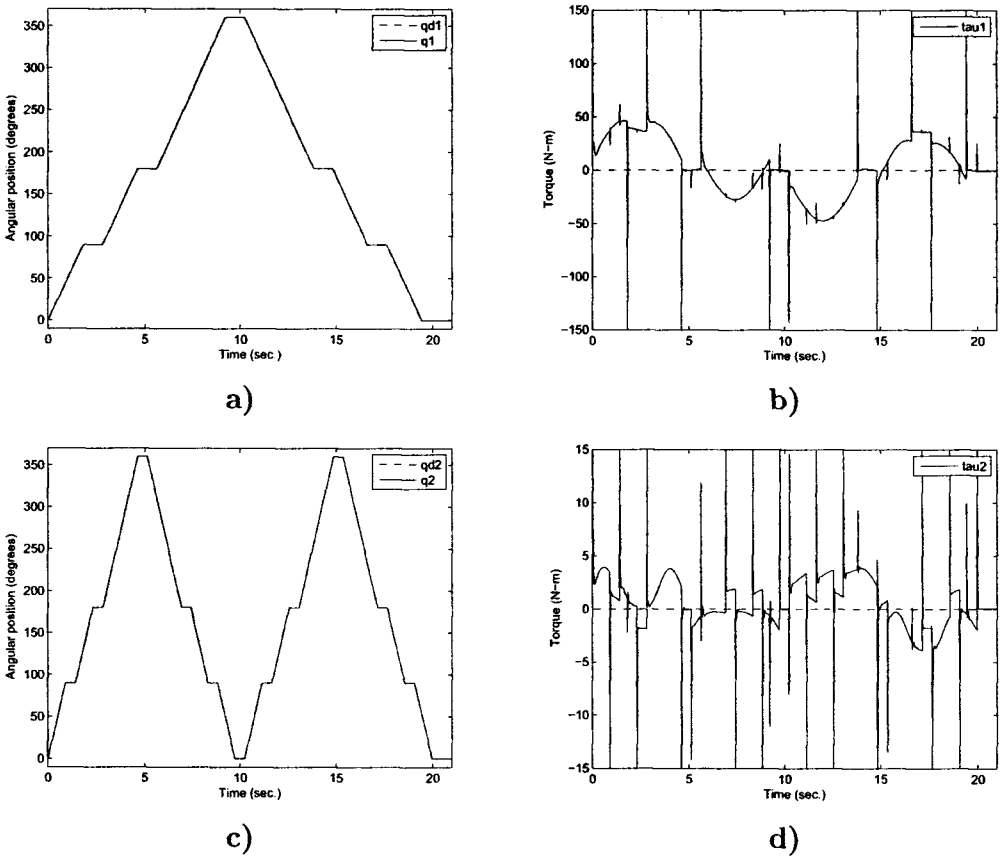


Figure 5.11: FST PD+ performance test. Response of: a) link 1; c) link 2. Torque for b) link 1; d) link 2.

Table 5.10: References for velocity controllers in robustness tests.

Link 1 References	1. Ramp to 90° @ 50°/*sec.* 2. 90° for 1*sec.* 3. Ramp to 180° @ 50°/*sec.* 4. 180° for 1*sec.* 5. Ramp to 90° @ 50°/*sec.* 6. 90° for 1*sec.* 7. Ramp to 0° @ 50°/*sec.* 8. 0° for 1*sec.*
Link 2 References	1. Ramp to 90° @ 100°/*sec.* 2. 90° for 1*sec.* 3. Ramp to 180° @ 100°/*sec.* 4. 180° for 1*sec.* 5. Ramp to 90° @ 100°/*sec.* 6. 90° for 1*sec.* 7. Ramp to 0° @ 100°/*sec.* 8. 0° for 1*sec.*

Robustness tests for the FST PD+ Controller

The last step for evaluating this controller is the robustness test, which is also similar to that made for the position controllers.

The external perturbation test is exactly the same performed to earlier controllers, 20° @ $t = 2$ *sec.* being applied to both links at the same time. References are static and defined as $q_{d1} = 90°$ and $q_{d2} = 0°$ along the simulation time for link 1 and link 2 respectively. Figure 5.12(a,b) show results for this test.

For the parameter variation test similar references are given as for the position controllers. Table 5.10 lists the references for this test which are repeated three times, one cycle for each value of m_2. The same velocity as the normal operation conditions test (50°/*sec.* for link 1 and 100°/*sec.* for link 2) is used. Link's 2 mass m_2 is a function of time defined by equation 5.13 for link 1 and equation 5.14 for link 2. Figure 5.12(c,d) show performance of controllers for link 1 and link 2 respectively.

$$m_2(t) = \begin{cases} 3.88\ kg & \text{if} \quad 0 \leq t < 10.5\ sec. \\ 7.76\ kg & \text{if} \quad 10.5 \leq t < 22.3\ sec. \\ 11.64\ kg & \text{if} \quad t > 22.3\ sec. \end{cases} \tag{5.13}$$

$$m_2(t) = \begin{cases} 3.88\ kg & \text{if} \quad 0 \leq t < 7.3\ sec. \\ 7.76\ kg & \text{if} \quad 7.3 \leq t < 15\ sec. \\ 11.64\ kg & \text{if} \quad t > 15\ sec. \end{cases} \tag{5.14}$$

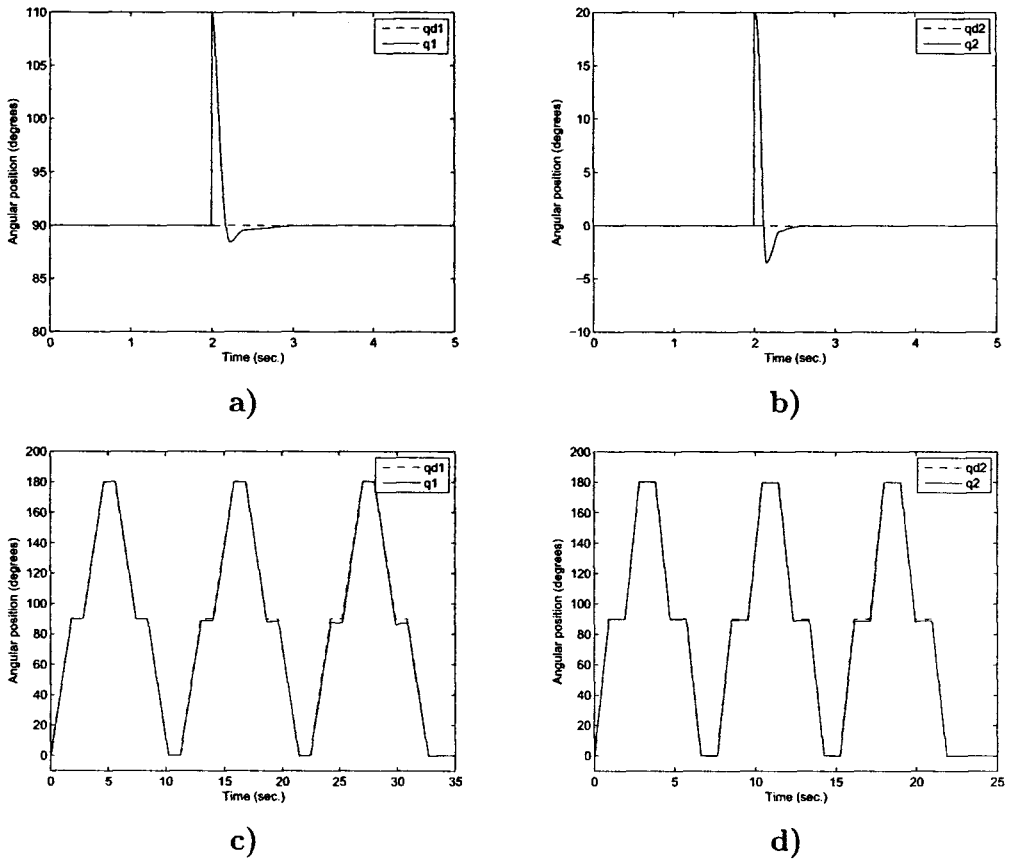


Figure 5.12: FST PD+ robustness test. External perturbation for: a) link 1; b) link 2. Parametric variations for: c) link 1; d) link 2.

This controller shows a superior performance than the position controllers in the external perturbation test. The settling time is very low compared to position controllers, $t_s = 0.4sec.$ for link 1 and $t_s = 0.3sec.$ for link 2. Accuracy is very good, having $e_{ss} = 0.0°$ for both links.

In the parameter variation test, steady state error and overshoot are present when m_2 is 200% and 300% of its original value. For link 1 overshoot was 4.0° and $e_{ss} = 2.8°$ which are high, while for link 2 overshoot was 2.1° and $e_{ss} = 1.5°$ These last results on performance criterions are high, but also the parameter change.

5.2.2 Simulation Results for the FSM Controller

Optimization of Parameters for the FSM Controller

The FSM controller is composed of a sliding surface defined by λ and a fuzzy system defined by p_1, k_1, k_2 and k_3 as shown in figure 4.9. These parameters are listed

Table 5.11: Parameters adjusted by the GA for the FSM controllers.

	Parameter	String length
Sliding surface	λ	6 bit
Fuzzy system	p_1	6 bit
	k_1	6 bit
	k_2	5 bit
	k_3	5 bit

Table 5.12: Parameters used by the GA to optimize the FSM controllers.

Population size	40
Mutation probability	2%
Number of generations	50
Crossover probability	100%
Chromosome length	28 bits

in table 5.11 along with the length of the string that codifies it in the GA.

The decodification and adjustment for k_1, k_2 and k_3 is similar as for the FLTs. It is done by the following equation:

$$\begin{aligned} k_3 &= d_{k_3} \\ k_2 &= \tfrac{d_{k_2}}{100} k_3 \\ k_1 &= \tfrac{d_{k_1}}{100} k_2 \end{aligned} \tag{5.15}$$

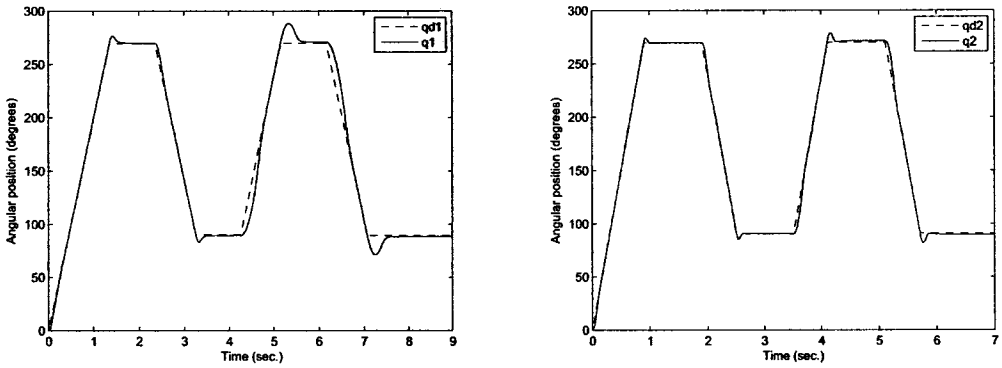
where $d_{k_2}, d_{k_1} \in [0, 100]$ are the decodified values using equation 2.14.

This controller is optimized using the GA with parameters listed in table 5.12. The same conditions as for the FST PD+ are used, i.e. references and m_2 increment. After the PID, this controller is the one with less parameters and therefore its chromosome length and population size are smaller than for the FST PID and the FST PD+.

Response of the system after optimization is shown in 5.13. The controller makes the system move very close to its reference. The overall performance is very similar to the FST PD+ controller at least in this step.

Performance tests for the FSM Controller

Once optimized, the next step according to the methodology is testing the controller in normal operation conditions. The same references are given to the system and



a) b)

Figure 5.13: FSM with optimized parameters. Response of: a) link 1; b) link 2.

the response is reported in figure 5.14. This test also shows that this controller behaves almost the same as the FST PD+ controller, with no change in e_{ss} and overshoot only 0.1° less for both links. Torque plots are also very similar.

Robustness tests for the FSM Controller

This section presents the robustness tests performed to the FSM controller. Conditions and references are the same as for the FST PD+ controller, shown in fig.5.15. Results are very similar than for the FST PD+ controller. When an external perturbation is presented t_s is only enhanced for link 1 in $0.1sec.$, for link 2 remains the same, accuracy is also very good with $e_{ss} = 0.0°$ for both links.

For link 1, the overshoot is enhanced in the parameter variation test, going from 4.0° when using the FST PD+ to 2.8° with the FSM, for link 2 only decreases 0.1°. Also, accuracy is enhanced for link 1 going from a $e_{ss} = 2.8°$ when using the FST PD+ to a $e_{ss} = 2.0°$ with the FSM, for link 2 is also enhanced but only a change of 0.1° less.

5.2.3 Velocity Controllers Comparison

A comparison of the velocity controllers is presented in 5.13. The same criterions measured for the position controllers are measured for the velocity controllers. From this table, the FSM controller has a better evaluation than the FST PD+ in every criterion for both links. Although this difference, behavior is almost the same and both controllers seem to be much more robust than position controllers, as the external perturbation test suggests (which is the same for both type of controllers).

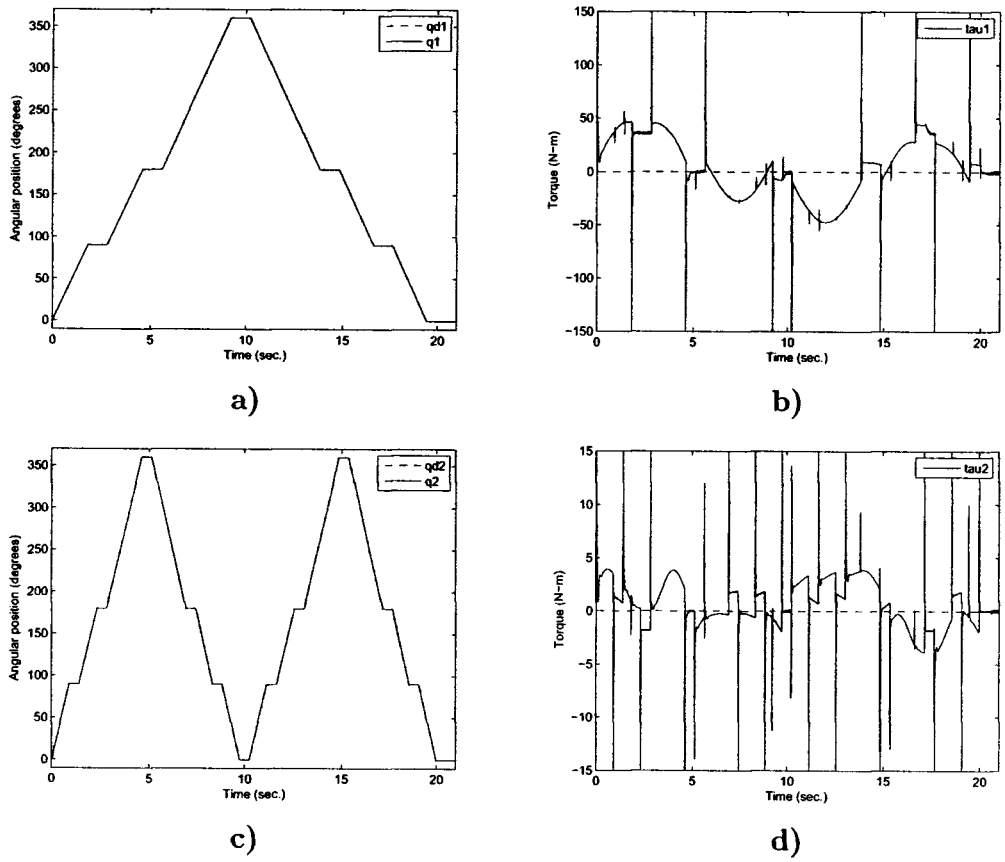


Figure 5.14: FSM performance test. Response of: a) link 1; c) link 2. Torque for b) link 1; d) link 2.

Table 5.13: Performance Comparison of Velocity Controllers

		FST PD+		FSM		
		Link 1	Link 2	Link 1	Link 2	Figures
Optimization	IAE	43.2	11.2	40.1	11.1	5.10,5.13
Normal Operation	IAE	10.9	12.3	9.7	11.9	5.11,5.14
	Overshoot (deg.)	0.6	1.1	0.5	1.0	
	e_{ss} (deg.)	0.0	0.1	0.0	0.1	
External Perturbation	t_s @ 0.5 deg. (sec.)	0.4	0.3	0.3	0.3	5.12,5.15
	e_{ss} (deg.)	0.0	0.0	0.0	0.0	
Parameter Variation	IAE	35.9	14.7	28.0	14.3	
	Overshoot (deg.)	4.0	2.1	2.8	2.0	
	e_{ss} (deg.)	2.8	1.5	2.0	1.4	

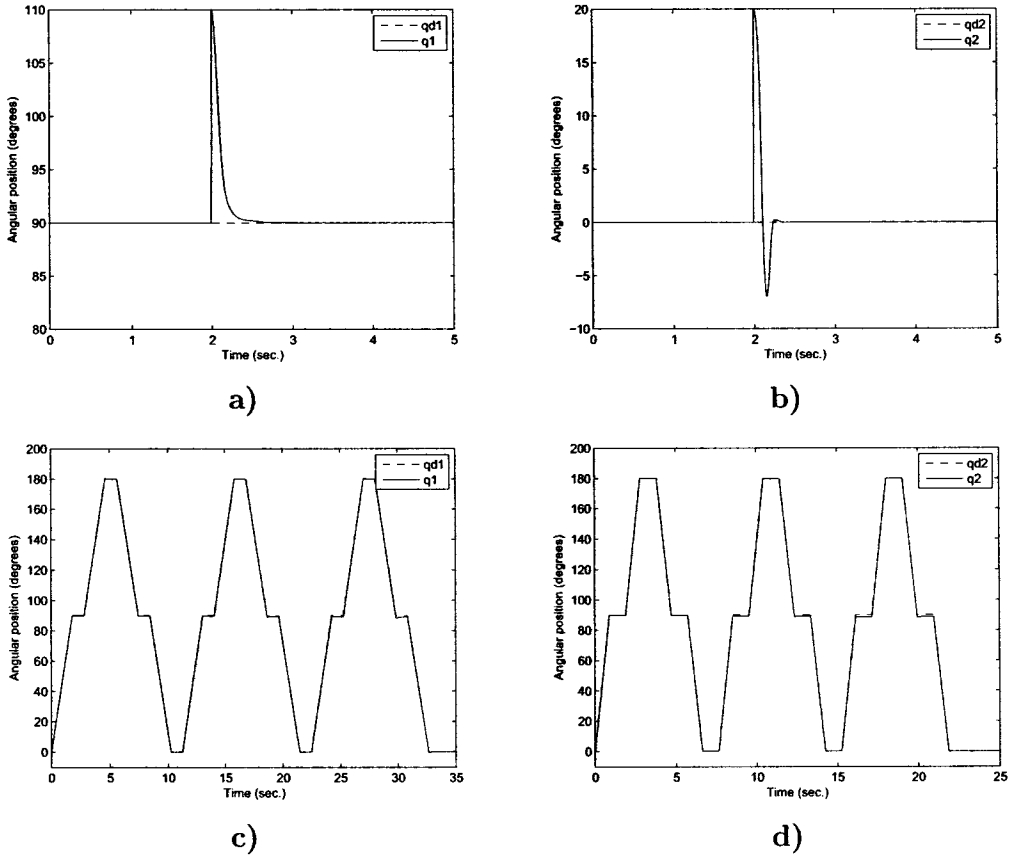


Figure 5.15: FSM robustness test. External perturbation for: a) link 1; b) link 2. Parametric variations for: c) link 1; d) link 2.

Chapter 6

Conclusions

Results obtained after optimization of controllers presented in the last chapter show that the GA successfully optimized both kind of controllers. The GA started from random parameter settings for all controllers, knowing nothing about how to set these parameters correctly. Setting the controller's parameters empirically would have been a hard job and doubts about the good setting of the parameters would remained. Controllers comparison was done as fair as possible for each kind of controllers, i.e. same tuning procedure, same optimization objective and same testing conditions.

Position controllers show less accuracy and more error compared with velocity controllers, but its structure does not involve the compensation of dynamics based in the model of the robot. Compensation of the robot's dynamics involve matrix multiplication, which can be costly in terms of computer operations. Knowledge of the robot's parameters is also needed to compute the compensation, but is not always available or precise. However, velocity controllers show good robustness in the presence of model uncertainties.

Even, velocity controllers have more information about the desired output of the system than position controllers. Velocity controllers receive the desired angular position $\boldsymbol{q}_d$, the desired angular velocity $\dot{\boldsymbol{q}}_d$ and the desired angular acceleration $\ddot{\boldsymbol{q}}_d$. Position controllers are only guided by the desired angular position $\boldsymbol{q}_d$ (the desired angular velocity $\dot{\boldsymbol{q}}_d$ is always zero).

6.1 Discussion about Position Controllers

It is shown from table 5.6, that the FST PID controller is better for link 1 than the PID controller in all cases. While for link 2 works better the PID controller when the robot works in normal operation conditions, but when parameters vary the FST PID performs better.

These results can be justified according to the following points:

- Link 1 is heavier and bigger than link 2, its non-linearities in the dynamics are harder and a non-linear controller is neccesary to control this link.

- Using a non-linear controller for link 2 only increases the total complexity of the system (*robot dynamics + controller*), making it more difficult to be optimized. A non-linearity is added to the system from the point of view of the GA.

An interesting point about the parameters of position controllers after optimization (see appendix A) is that the integral gain K_i in both PIDs ended with the minimum possible value, within the specific range. In the FST PID, the $K_{i_{fuzz}}$ gain also has the minimum value in the output MF μ_y^B for the controller of link 1, while for link 2 is not the minimum but is small (2.1 in the range [0.1 250]).

For the integral gains in the FST PID, the order of rules can be chosen by the GA with parameter *RuleOrder*. This parameter is decodified from a single bit (1 bit from a 117-bit string), therefore it is hard for the GA to test the different combinations of this parameter with the others, which could lead to a quick convergence into a local maximum.

Maybe $RuleOrder = 0$ for the case of integral gains and different values for the parameters of MFs could lead to a better performance, but only empirical knowledge supports this inference. However, from the simulation results the GA suggests that a PD-based controller may be a better choice for position tasks, when the desired deynamic behavior of the links of this particular robot is similar to a second-order linear system.

6.2 Discussion about Velocity Controllers

Table 5.13 shows the performance criterions comparison for the two velocity controllers considered. It can be seen that both controllers show almost the same performance for both links in all tests, except in the parameter variation test, in which the FSM controller was better than the FST PD+ controller.

From the performance criterions table, at least two relevant conclusions can be drawn:

- The GA successfully optimized both controllers for the specific task. If the performance criterions were significantly better for one controller than the other, some doubts would remain about the reliability of the optimization method. However, the GA successfully optimized these controllers for the specific objective because the final result, regardless of the control structure, was similar. This is also one of the Hypothesis intended to be proved.

- Both controllers have the same capabilities or are equivalent when its specific parameters are set in a certain way, in this specific application. Its structure is similar, but both are derived from different theories.

6.3 Contributions of this research work

In the literature, there are no reports of comparison of different controllers previously optimized for the same task. Most of the reports compare a single controller tuned by different methods or two or more controllers optimized empirically. Optimization of different controllers of the same kind for the same tasks leads to a fair testing and comparison of their capabilities. So the comparison tests presented in this research are a contribution to the study of robot control.

In the case of position controllers, the GA suggested that a PD-based controller is a better choice if the desired behavior of the system is similar to a second-order system. This is also a contribution because there are no reports that achieve the dynamics of the robot's links for position tasks to behave this way.

For velocity controllers, the contribution is the optimization of the FST PD+ controller and the FSM controller for this application. Also the experimental results concluding that both controllers can have the same capabilities, given that performance reached after optimization for both was similar. However, the FSM controller is encouraged to be used in this application because there are less parameters to be adjusted, being a simpler problem to be optimized and having the same capabilities than a more complex controller, the FST PD+.

6.4 Further Work

As a further work, it might be interesting to develop a controller derived from the Accuracy-Based Classifier System (XCS). An XCS is an adaptable system whichs learns from the payoff it receives from the enviromnent after performing an action given some condition. With this system in mind, Robot's states could be the condition and controller gains could be the action. A system model could also be derived from this method by using the XCS to learn the system model in parallel with the operation.

The design and construction of an experimental 2 DOF robot manipulator is also considered to test the proposed controllers after they have been optimized. It could also be used as a test bed for other AI methods.

Appendix A

Final Value of Parameters modified by the GA

A.1 Final Values for Parameters of Controllers

Final values for parameters modified by the GA are reported in the next tables. For the PID controller, tables A.1 and A.2 show the value of parameters after optimization for link 1 and link 2 respectively.

Final values of parameters optimized for the FST PID controller of link 1 and link 2 are shown in A.3 and A.4 respectively. Figures A.1 and A.2 show the shape of the input and output MFs represented by these values for link 1 and link 2 respectively. In tables A.5 and A.6 results for the FST PD+ controller are shown for link 1 and link 2 respectively. For this controller, figures A.3 and A.4 illustrate the MFs of the FLTs for link 1 and link 2 respectively.

Tables in which parameters for FLTs are defined, the final value of the parameters may not be within the specific range for that parameter. This is because the decodification of p_1, p_2, p_3 and p_4 is done using equation 5.10 shown below again:

$$p_n = p_{n-1} + d_{p_n} \tag{A.1}$$

in which the decodified value of the current parameter is added to the value of the last parameters decodified. For the parameters k_1, k_2 and k_3 the decodification is done by equation 5.11 as shown below again:

$$\begin{aligned} k_3 &= d_{k_3} \\ k_2 &= \tfrac{d_{k_2}}{100} k_3 \\ k_1 &= \tfrac{d_{k_1}}{100} k_2 \end{aligned} \tag{A.2}$$

where $d_{k_2}, d_{k_1} \in [0, 100]$ are the decodified values using equation 2.14. By using this decodification k_1 and k_2 can have final value higher than its range max.

For the FSM Controller, the final value of parameters for link 1 and link 2 are shown in tables A.7 and A.8 respectively. The MFs for the fuzzy systems used in this controller are illustrated in figures A.5 and A.6 for link 1 and link 2 respectively.

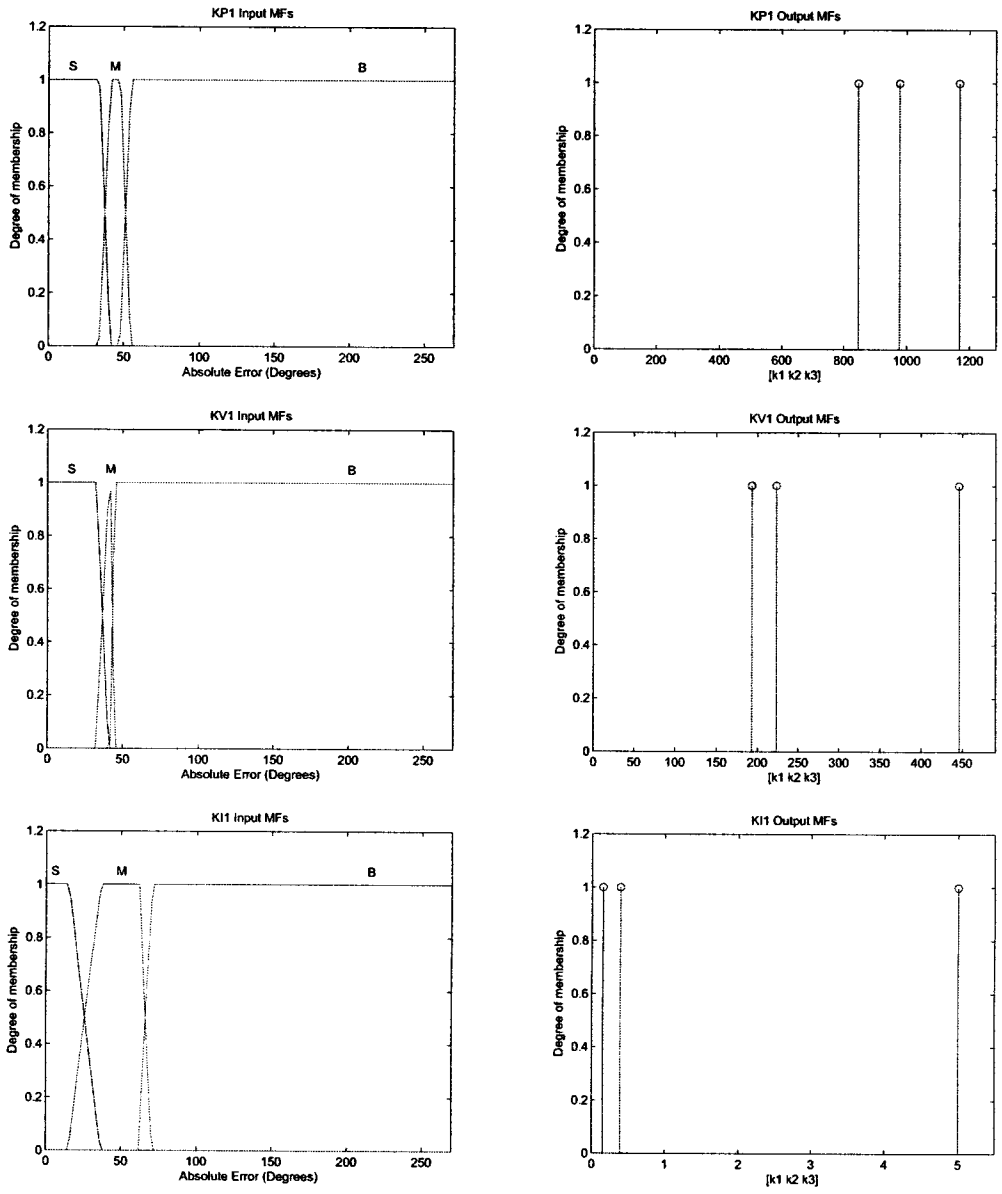


Figure A.1: MFs of FLTs of FST PID Controller for link 1.

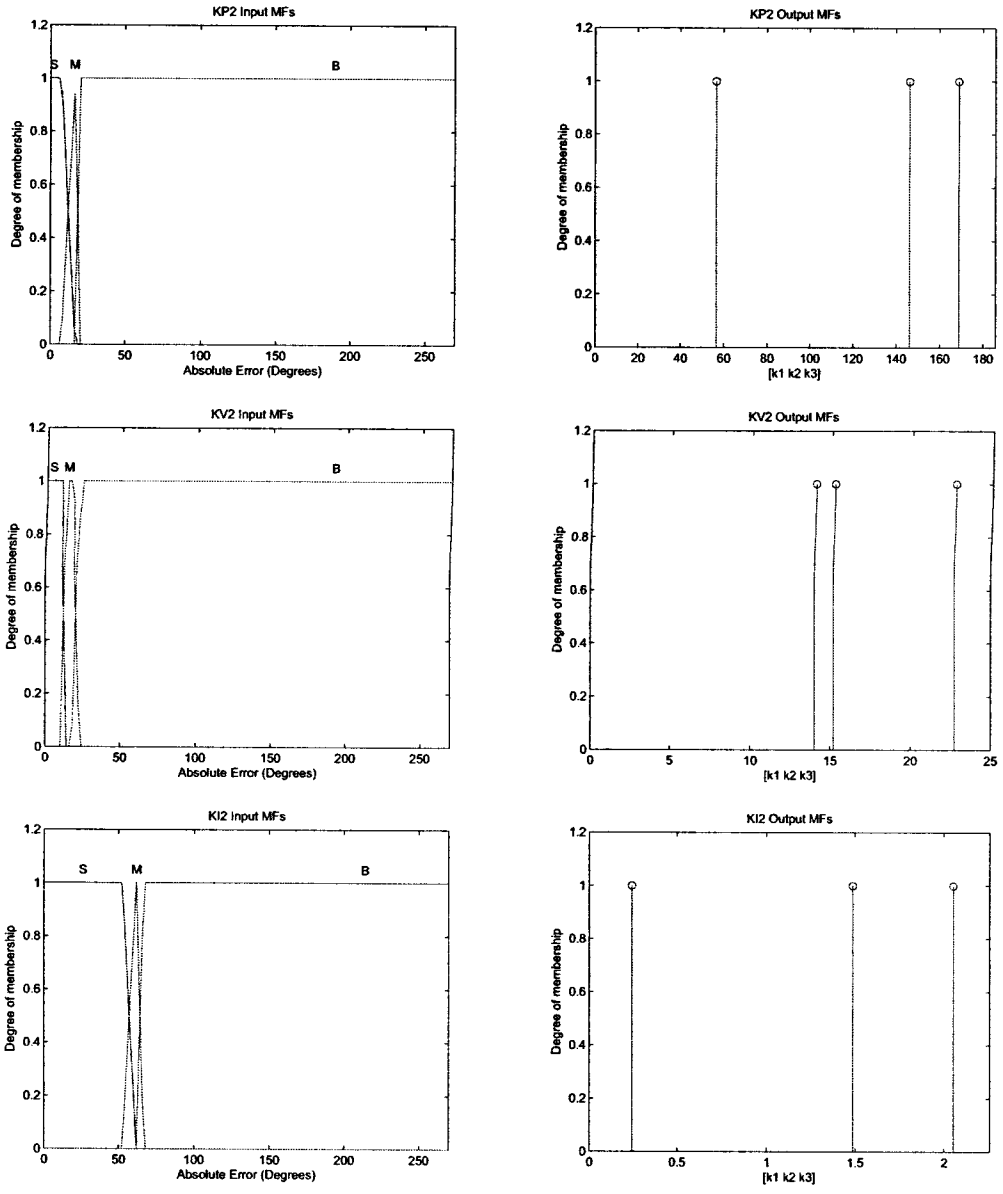


Figure A.2: MFs of FLTs of FST PID Controller for link 2.

Table A.1: Final Values for the PID Controller of Link 1

Parameter	Range Min	Range Max	String Length	Final Value
K_{p_1}	0.1	1000	8	859.4
K_{v_1}	0.1	500	8	248.1
K_{i_1}	0.1	2000	8	0.1

Table A.2: Final Values for the PID Controller of Link 2

Parameter	Range Min	Range Max	String Length	Final Value
K_{p_2}	0.1	100	8	92.6
K_{v_2}	0.1	50	8	19.0
K_{i_2}	0.1	200	8	0.1

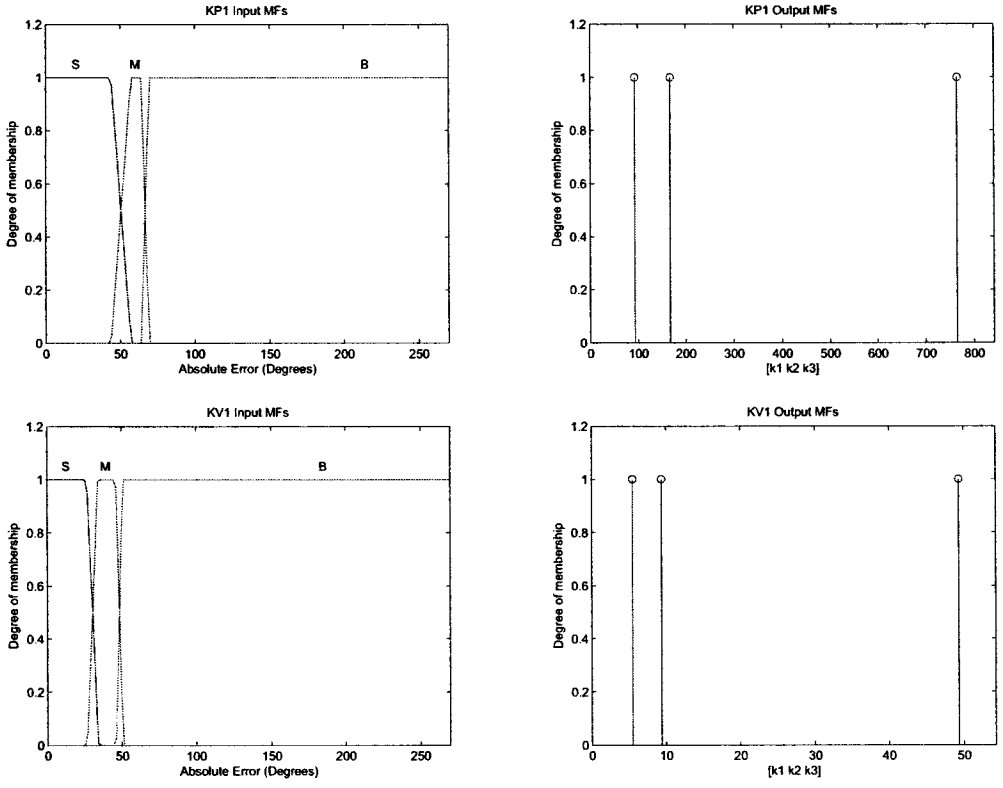


Figure A.3: MFs of FLTs of FST PD+ Controller for link 1.

Table A.3: Final Values for the FST PID Controller of Link 1

Parameter		Range Min	Range Max	String Length	Final Value
$K_{p_1}(\tilde{q}_1)$	p_1	0.1	45	5	33.8
	p_2	0.1	15	5	41.3
	p_3	0.1	20	5	47.6
	p_4	0.1	15	5	54.7
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	846.7
	k_2	5	95	5	978.2
	k_3	500	1500	8	1168.0
$K_{v_1}(\tilde{q}_1)$	p_1	0.1	45	5	32.4
	p_2	0.1	15	5	40.9
	p_3	0.1	20	5	41.9
	p_4	0.1	15	5	44.8
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	193.1
	k_2	5	95	5	223.0
	k_3	50	700	8	446.1
$K_{i_1}(\tilde{q}_1)$	p_1	0.1	60	5	15.1
	p_2	0.1	30	5	36.7
	p_3	0.1	30	5	62.0
	p_4	0.1	30	5	70.5
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	0.2
	k_2	5	95	5	0.4
	k_3	5	2500	8	5.0

Table A.4: Final Values for the FST PID Controller of Link 2

Parameter		Range Min	Range Max	String Length	Final Value
$K_{p_2}(\tilde{q}_2)$	p_1	0.1	45	5	7.1
	p_2	0.1	15	5	16.5
	p_3	0.1	20	5	16.6
	p_4	0.1	15	5	19.5
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	56.7
	k_2	5	95	5	146.2
	k_3	1	200	8	168.9
$K_{v_2}(\tilde{q}_2)$	p_1	0.1	45	5	11.3
	p_2	0.1	15	5	12.8
	p_3	0.1	20	5	17.6
	p_4	0.1	15	5	22.8
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	14.0
	k_2	5	95	5	15.2
	k_3	0.1	100	8	22.7
$K_{i_2}(\tilde{q}_2)$	p_1	0.1	60	5	52.5
	p_2	0.1	30	5	62.0
	p_3	0.1	30	5	62.1
	p_4	0.1	30	5	66.8
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	0.2
	k_2	5	95	5	1.5
	k_3	0.1	250	8	2.1

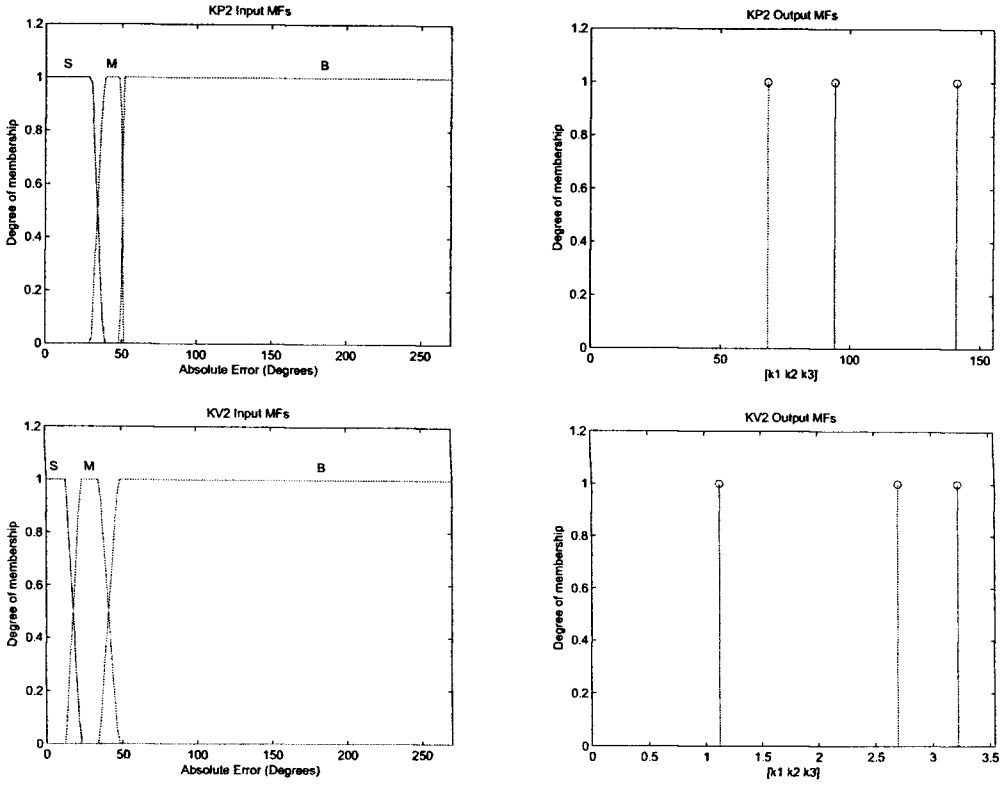


Figure A.4: MFs of FLTs of FST PD+ Controller for link 2.

Table A.5: Final Values for the FST PD+ Controller of Link 1

Parameter		Range Min	Range Max	String Length	Final Value
$K_{p_1}(\tilde{q}_1)$	p_1	0.1	45	5	43.6
	p_2	0.1	15	5	57.2
	p_3	0.1	20	5	64.1
	p_4	0.1	15	5	69.4
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	93.1
	k_2	5	95	5	167.3
	k_3	50	800	8	764.8
$K_{v_1}(\tilde{q}_1)$	p_1	0.1	45	5	29.6
	p_2	0.1	15	5	38.0
	p_3	0.1	20	5	51.8
	p_4	0.1	15	5	57.0
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	5.5
	k_2	5	95	5	9.4
	k_3	10	250	8	49.4

Table A.6: Final Values for the FST PD+ Controller of Link 2

Parameter		Range Min	Range Max	String Length	Final Value
$K_{p_2}(\tilde{q}_2)$	p_1	0.1	45	5	33.8
	p_2	0.1	15	5	42.7
	p_3	0.1	20	5	55.9
	p_4	0.1	15	5	56.4
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	68.3
	k_2	5	95	5	94.2
	k_3	1	200	8	140.9
$K_{v_2}(\tilde{q}_2)$	p_1	0.1	45	5	14.1
	p_2	0.1	15	5	24.9
	p_3	0.1	20	5	38.7
	p_4	0.1	15	5	52.8
	RuleOrder	0	1	5	1.0
	k_1	5	95	5	1.1
	k_2	5	95	5	2.7
	k_3	0.1	100	8	3.2

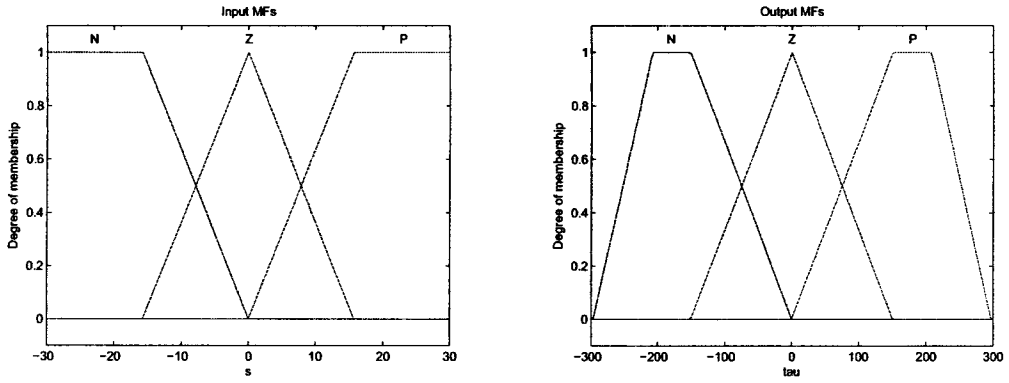


Figure A.5: MFs of the FSM Controller for link 1.

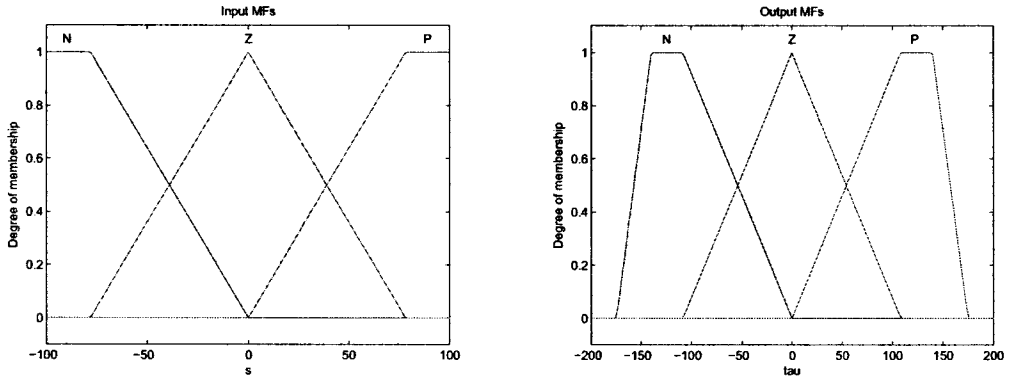


Figure A.6: MFs of the FSM Controller for link 2.

Parameter p_1 is decoded with the use of equation 2.14. For parameters k_1, k_2 and k_3 the decodification is the same as for the FLTs.

Shapes of functions $K_{Fuzz_1}(s_1)$ and $K_{Fuzz_2}(s_2)$ are illustrated in figures A.7 and A.8 for the controller of link 1 and link 2 respectively.

A.2 Sliding Surface of the FSM Controller

In order to illustrate the slope of the sliding surfaces in the phase planes for the FSM Controller, the following test was performed on the system. References were given for both links as shown in table A.9, initial references were $\boldsymbol{q}_d(0) = [0°\ 0°]^T$ and $\dot{\boldsymbol{q}}_d(0) = [0°\ 0°]^T$. Initial positions for link 1 and 2 were $\boldsymbol{q}(0) = [20°\ 20°]^T$.

The FSM Controller after optimization shows the response in the phase plane of $\tilde{q}$ illustrated in figure A.9 for link 1 and link 2. In these figures the sliding surfaces are illustrated as dashed lines that cross the origin, which represent zero error in angular

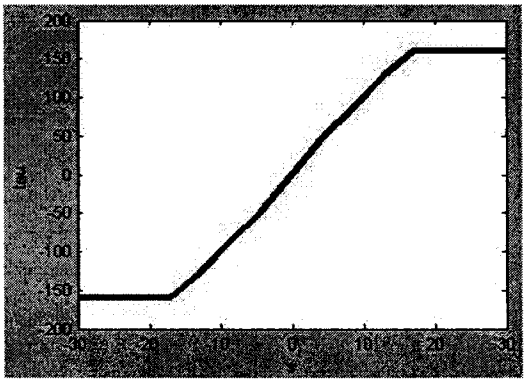


Figure A.7: Function $K_{Fuzz_1}(s_1)$.

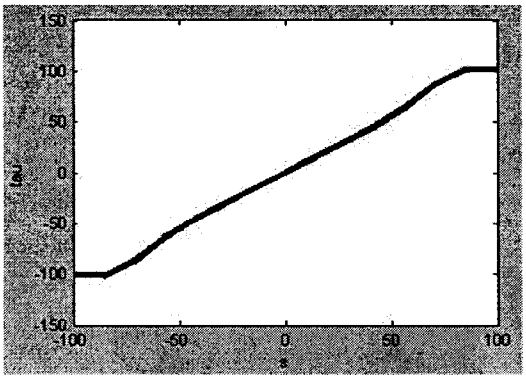


Figure A.8: Function $K_{Fuzz_2}(s_2)$.

Table A.7: Final Values for the FSM Controller of Link 1

Parameter		Range Min	Range Max	String Length	Final Value
s_1	λ	0.1	200	6	12.4
$K_{Fuzz_1}(s_1)$	p_1	0.1	100	6	15.7
	k_1	10	90	5	150.7
	k_2	10	90	5	207.8
	k_3	10	350	6	296.9

Table A.8: Final Values for the FSM Controller of Link 2

Parameter		Range Min	Range Max	String Length	Final Value
s_2	λ	0.1	200	6	50.9
$K_{Fuzz_2}(s_2)$	p_1	0.1	100	6	78.1
	k_1	10	90	5	108.3
	k_2	10	90	5	139.8
	k_3	10	350	6	174.7

Table A.9: References to illustrate sliding surfaces.

Link 1 References	1. Ramp to 100° @ 50°/*sec.* 2. 100° for 0.5*sec.*
Link 2 References	1. Ramp to 100° @ 50°/*sec.* 2. 100° for 0.5*sec.*

position and velocity, or $\tilde{\boldsymbol{q}}_{ds} = [0\ 0]^T$. Slope for these sliding surfaces is given by the parameter λ reported in tables A.7 and A.8 for link 1 and link 2 respectively.

The same test was performed using the discontinuous function from equation 4.26 (shown below again) instead of the fuzzy system, as the classical sliding mode controller was defined. Figure A.10 shows the response of the system in the phase plane for link 1 and link 2 using the sliding mode controller with this function. It can be seen how *chattering* is present due to the hard non-linearity the relay function adds to the system. Gains for the sgn function were set empirically as $k_1 = 50$ for the controller of link 1 and $k_2 = 5$ for the controller of link 2.

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{eq} - k\ \mathrm{sgn}(\boldsymbol{s}) \tag{A.3}$$

where sgn is the sign function:

$$\begin{aligned} \mathrm{sgn}(s) &= +1 \quad \text{if} s > 0 \\ \mathrm{sgn}(s) &= -1 \quad \text{if} s < 0. \end{aligned} \tag{A.4}$$

Transient response of the system is shown in figures A.11 and A.12 for the FSM Controller and the classical Sliding Mode Controller respectively. In torque plots applied by the classical Sliding Mode Controller (see figure A.12-c,d) it can be seen that chattering in the control signal is present in both links.

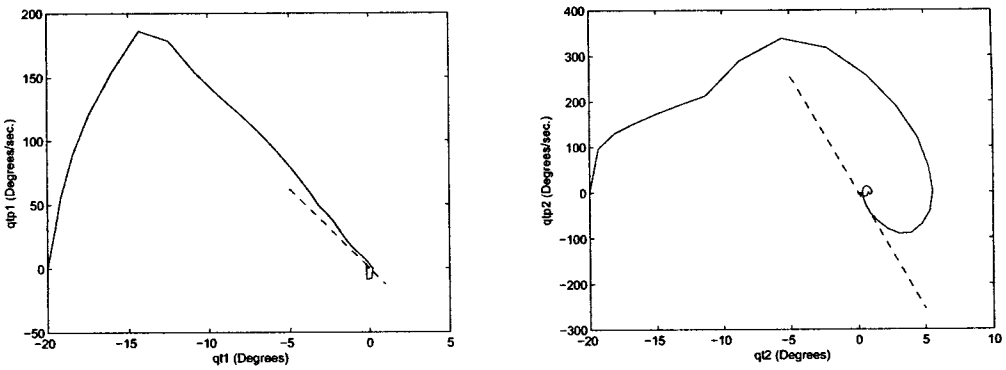


Figure A.9: Phase plane response of the FSM for: a) link 1; b) link 2.

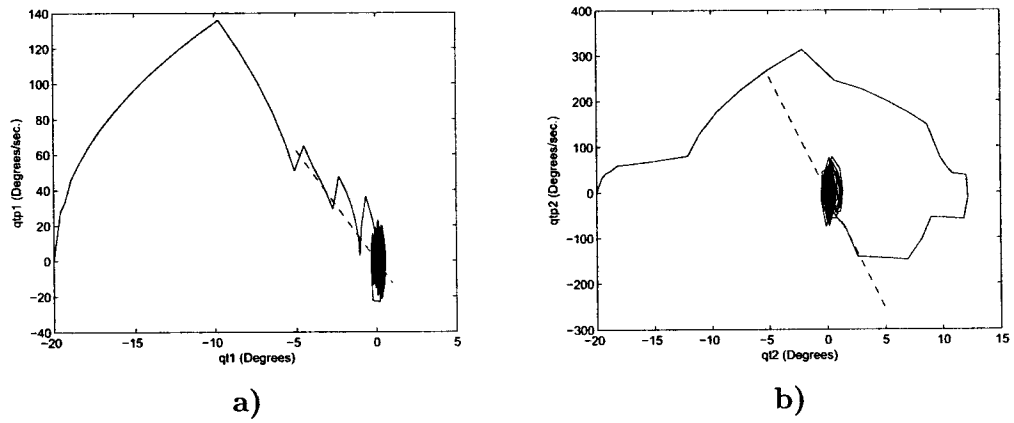


Figure A.10: Phase plane response of the classical Sliding Mode for: a) link 1; b) link 2.

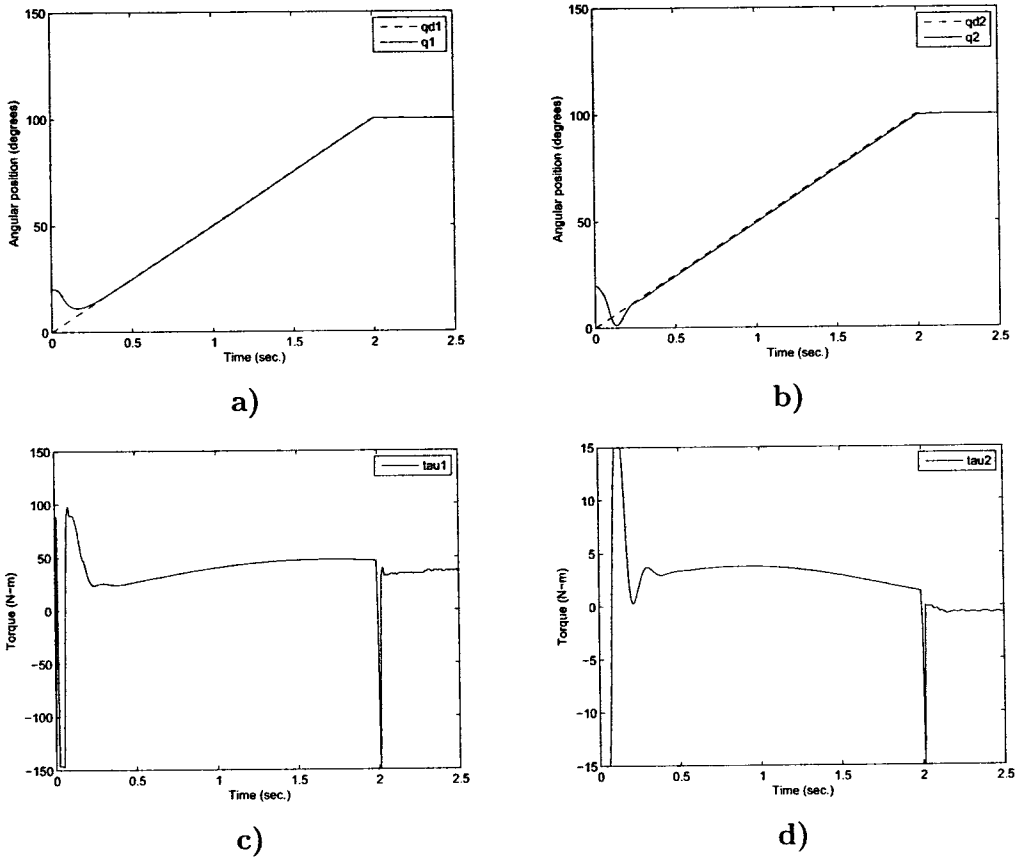


Figure A.11: FSM test. Response of a) link 1; b) link 2. Torque for: c) link 1; d) link 2.

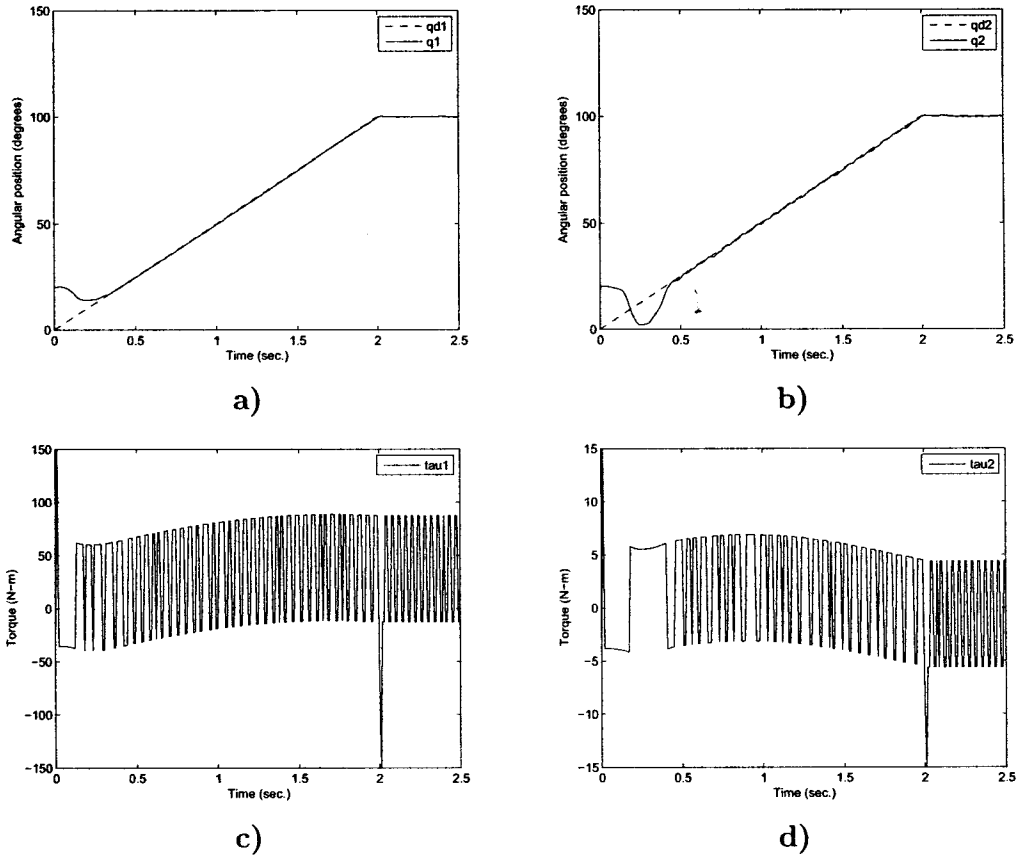


Figure A.12: Classical Sliding Mode. Response of: a) link 1; b) link 2. Torque for: c) link 1; d) link 2.

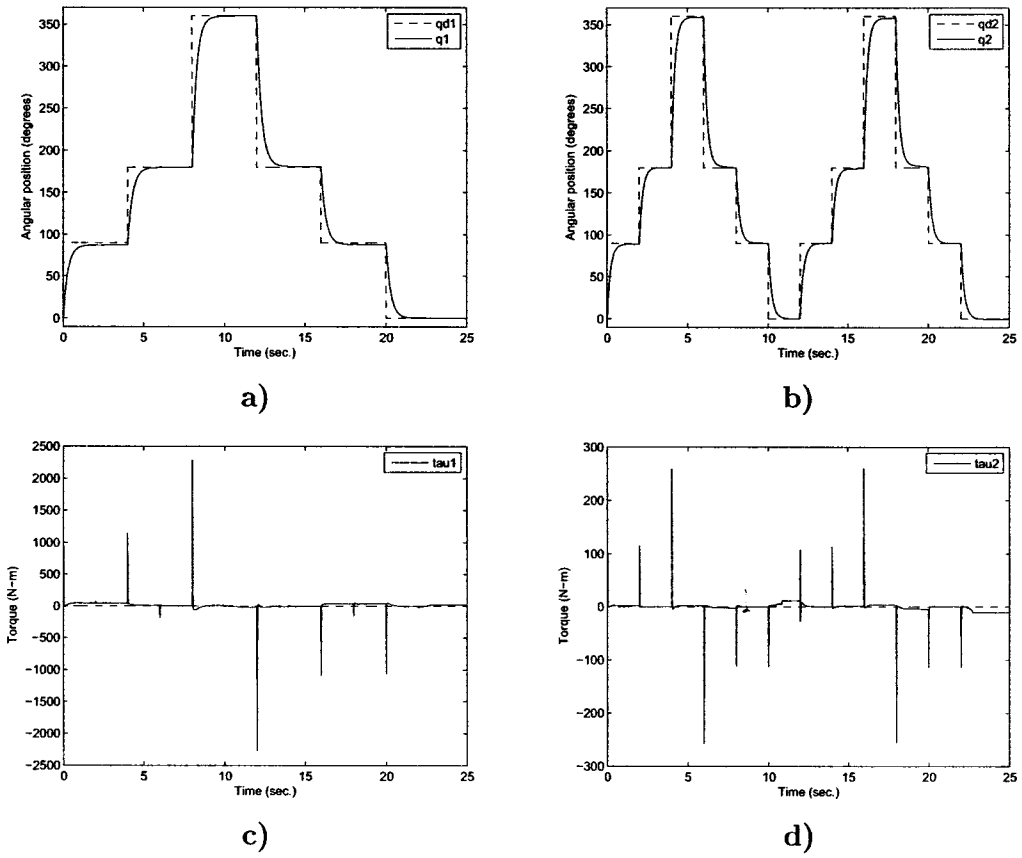


Figure A.13: PID with unsaturated torque. Response of: a) link 1; b) link 2. Torque for: c) link 1; d) link 2.

A.3 PID with Unsaturated Torque Signal

In order to illustrate the performance of a controller without torque signal satuaration, the PID experiment was modified for the test shown below. The PID controller with saturated torque signal is now compared with the same PID controller but with unsaturated torque signal for both links. References are the same as in the performance test for the PID controller (see chapter 5). Parameters of the controller are the resulting after optimization.

Figure A.13 (a, c) show the transient response and the torque applied for link 1. For link 2, figure A.13 (b, d) show the transient response and the torque applied respectively.

Table A.10 compares performance of the PID with saturated and unsaturated torque signal. The saturated torque signal is according to the capabilities of real direct-drive motors, is the experiment reported in chapter 5 for the performance of the PID

Table A.10: Performance comparison of PID with saturated and unsaturated torque.

		Saturated		**Unsaturated**		
		Link 1	Link 2	Link 1	Link 2	Figures
Normal Operation	IAE	270.2	372.4	233.0	322.5	5.4,5.7,A.12
	Overshoot (deg.)	2.4	1.2	2.4	0.1	
	e_{ss} (deg.)	2.9	2.1	2.9	2.1	

in normal operation conditions. For link 1 almost the same performance is obtained, IAE is better if torque is unsaturated. For link 2, if torque is unsaturated, performance is enhanced in the overshoot criterion and IAE.

Altough performance of the controller is better with unsaturated control signal, is not a significant performance enhancement. From torque plots, it can be seen that torque requested by the controller to the actuators is beyond the capabilities of real direct-drive motors.

Bibliography

[1] S. Bennett. Nicholas minorsky and the automatic steering of ships. *Control Systems Magazine, IEEE*, 4(4):10 – 15, nov 1984.

[2] Rafael Ernesto Bourguet Díaz. *Qualitative Knowledge Acquisition Using Fuzzy Logic and System Dynamics.* PhD thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, N.L., México, 2003.

[3] Edgar Noriega Cristobal. Control difuso en modo deslizante para un manipulador de seis grados de libertad. Master's thesis, Instituto Tecnológico y de Estudios Superiores de Monterrey, Monterrey, N.L., México, 2008.

[4] D. Driankov, H. Hellendoor, and M. Reinfrank. *An Introduction to Fuzzy Control, second edition.* Springer, Berlin; New York, 1996.

[5] Jean-Jacques E. Slotine and Weiping Li. *Applied nonlinear control.* Prentice Hall, Englewood Cliffs, N.J., 1991.

[6] David Edward Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, Michigan, 1989.

[7] Walter Hauser. *Introducción a los principios de mecánica.* UTEHA, México, 1969.

[8] Dimitris Hristu, Jon Babb, Harjit Singh, and Susan Gottschlich. Position and force control of a multifingered hand: A comparison of fuzzy logic to traditional pid control. *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems. Advanced Robotic Systems and the Real World, IROS*, pages 1391–1398, 1994.

[9] John J. Craig. *Introduction to robotics: mechanics and control.* Pearson/Prentice Hall, One Lake Street, Upper Saddle River, New Jersey 07458, 2005.

[10] D. E. Koditschek. Natural motion for robot arms. *Proceedings of the IEEE 23th Conference on Decision and Control, Las Vegas, NV*, pages 733–735, 1984.

[11] Kristin Kristinsson and A. Guy Dumont. System identification and control using genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics, Vol. 22, No. 5*, pages 1033–1046, 1992.

[12] C.-C. Kung and S.-C. Lin. A fuzzy-sliding mode controller design. *Systems Engineering, 1992., IEEE International Conference on*, pages 608 –611, sep 1992.

[13] D.P. Kwok and Fang Sheng. Genetic algorithm and simulated annealing for optimal robot arm pid control. *IEEE Conference on Evolutionary Computation*, pages 707–713, 1994.

[14] H. R. Van Nauta Lemke and Wang De-zhao. Fuzzy pid supervisor. *Decision and Control, 1985 24th IEEE Conference on*, 24:602 –608, dec. 1985.

[15] Wang Li-Xin. *A course in fuzzy systems and control.* Prentice Hall, PTR, Upper Saddle River, N.J., 1997.

[16] Sinn-Cheng Lin and Chung-Chun Kung. A linguistic fuzzy-sliding mode controller. *American Control Conference, 1992*, pages 1904 –1905, june 1992.

[17] M. A. Llama, V. Santibañez, Rafael Kelly, and Jesus Flores. Stable fuzzy self-tuning computed-torque control of robot manipulators. *IEEE International Conference on Robotics and Automation*, pages 2369–2374, May 1998.

[18] Miguel A. Llama, Rafael Kelly, and Victor Santibañez. A stable motion control system for manipulators via fuzzy self-tuning. *Fuzzy Sets and Systems*, 124(2):133 – 154, 2001.

[19] S. Hutchinson M. Spong and M. Vidyasagar. *Robot Modeling and Control.* John Wiley and Sons, Hoboken, NJ, 2006.

[20] E. H. Mamdani. Applications of fuzzy algorithms for control of a simple dynamic plant. *IEEE Proceedings, vol. 121, no. 12*, pages 1585–1588, 1974.

[21] J.L. Meza, V. Santibanez, R. Soto, and M.A. Llama. Stable fuzzy self-tuning pid control of robot manipulators. *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pages 2624 –2629, oct. 2009.

[22] J.L. Meza, R. Soto, and J. Arriaga. An optimal fuzzy self-tuning pid controller for robot manipulators via genetic algorithm. *Artificial Intelligence, 2009. MICAI 2009. Eighth Mexican International Conference on*, pages 21 –26, nov. 2009.

[23] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs.* Springer-Verlag, Berlin; New York, 1999.

[24] N. Minorsky. Directional stability of automatically steered bodies. *J. Amer. Soc. of Naval Engineers*, 34:280 - 309, 1922.

[25] Hung T. Nguyen, Nadipuram R. Prasad, Carol L. Walker, and Elbert A. Walker. *A first course in fuzzy and neural control.* Chapman, Hall/CRC Press, Boca Raton, FL, 2003.

[26] Saeed B. Niku. *Introduction to Robotics, Analysis, Systems, Applications.* Pearson/Prentice Hall, One Lake Street, Upper Saddle River, New Jersey 07458, 2001.

[27] Katsuhiko Ogata. *Modern Control Engineering.* Pearson/Prentice Hall, One Lake Street, Upper Saddle River, New Jersey 07458, 2010.

[28] R. Palm. Sliding mode fuzzy control. *Fuzzy Systems, 1992., IEEE International Conference on*, pages 519 -526, mar 1992.

[29] Richard Paul Collins Paul. *Modelling, trajectory calculation and servoing of a computer controlled arm.* PhD thesis, Stanford University, Stanford, CA, USA, 1972.

[30] Sukon Puntunan and Manukid Parnichkun. Self-tuning precompensation of pid based heading control of a flying robot. *IEEE Workshop on Advanced Robotics and its Social Impacts.*, pages 226–230, 2005.

[31] Zhihua Qu, John Dorsey, Darren M. Dawson, and Roger W. Johnson. A new learning control scheme for robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1463–1468, 1991.

[32] Fernando Reyes and Rafael Kelly. Experimental evaluation of identification schemes on a direct drive robot. *Robotica*, 15(5):563–571, 1997.

[33] V. Santibáñez, R. Kelly, and M.A. Llama. Global asymptotic stability of a tracking sectorial fuzzy controller for robot manipulators. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(1):710 - 718, feb. 2004.

[34] Victor Santibañez and Rafael Kelly. *Control de Movimiento de Robots Mainpuladores.* Pearson/Prentice Hall, Madrid, España, 2003.

[35] Victor Santibañez, Rafael Kelly, and Miguel A. Llama. Fuzzy pd+ control for robot manipulators. *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, San Francisco, CA*, pages 2112–2117, 2000.

[36] V. Santibanez, R. Kelly, and M.A. Llama. A novel global asymptotic stable setpoint fuzzy controller with bounded torques for robot manipulators. *Fuzzy Systems, IEEE Transactions on*, 13(3):362 - 372, june 2005.

[37] William M. Spears and Kenneth A. De Jong. An analysis of multi-point crossover, 1991.

[38] M. Valenzuela-Rendón. The virtual gene genetic algorithm. *Genetic and Evolutionary Computation (GECCO)*, pages 1457–1468, 2003.

[39] Alen Varsek, Tanja Urbancic, and Bodgan Filipic. Genetic algorithms in controller design and tuning. *IEEE Transactions on Systems, Man and Cybernetics, Vol. 23, No. 5*, pages 1330–1339, 1993.

[40] L.A. Zadeh. Fuzzy sets. *Information and Control, vol. 8*, pages 338–353, 1965.